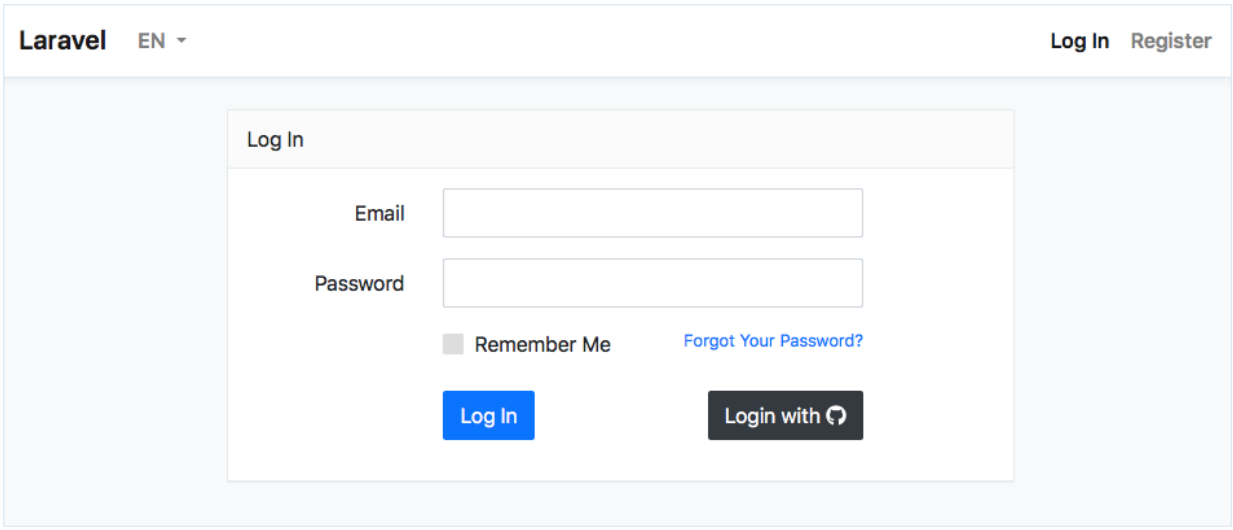
Table of Contents

Laravel-Vue SPA 2

> A Laravel-Vue SPA starter kit.



Features

- Laravel 8
- Vue + VueRouter + Vuex + Vuel18n + ESlint
- Pages with dynamic import and custom layouts
- Login, register, email verification and password reset
- Authentication with JWT
- Socialite integration
- Bootstrap 5 + Font Awesome 5

Installation

- `composer create-project --prefer-dist cretueusebiu/laravel-vue-spa`
- Edit `.env` and set your database connection details
- (When installed via git clone or download, run `php artisan key:generate` and `php artisan jwt:secret`)
- `php artisan migrate`
- `npm install`

Usage

Development

```
npm run dev
```

Production

```
npm run build
```

Socialite

This project comes with GitHub as an example for Laravel Socialite.

To enable the provider create a new GitHub application and use `https://example.com/api/oauth/github/callback` as the Authorization callback URL.

Edit `.env` and set `GITHUB_CLIENT_ID` and `GITHUB_CLIENT_SECRET` with the keys form your GitHub application.

For other providers you may need to set the appropriate keys in `config/services.php` and redirect url in `OAuthController.php` .

Email Verification

To enable email verification make sure that your `App\User` model implements the `Illuminate\Contracts\Auth\MustVerifyEmail` contract.

Testing

```
# Run unit and feature tests
vendor/bin/phpunit

# Run PhpStan tests
php vendor/bin/phpstan analyze

# Run PhpCs tests
php vendor/bin/phpcs
# all replace CRLF to LF
git config core.autocrlf false
git rm --cached -r .
git reset --hard

# Run Dusk browser tests
php artisan dusk
```

Changelog

Please see CHANGELOG for more information what has changed recently.

Введение

Данная система была разработана для учета деятельности компании, которая занимается аутсорсингом персонала, которая бы выполняла следующие функции:

- Заведение и учет заявок в систему
- Заведение и учет заказчиков в систему
- Заведение и учет сотрудников в систему
- Формирование бригад и контроль над ними
- Учет заработной платы
- Отслеживание начислений и выплат
- Заведение пользователей системы и управление их доступами

Ссылка на прототип по которому разрабатывалась система - https://6r6vj4.axshare.com/

Ссылка на ТЗ по которому разрабатывалась система - https://docs.google.com/document/d/1xQfU8UhANyJhXJbbidt1GJScCR693MR3/edit

Система реализована с использованием следующих технологий:

- Laravel 11
- Vue + VueRouter + Vuex + VueI18n
- PostgreSQL 14

Настройки проекта

Система реализована на технологиях следующих версий:

- Backend: php-8.2, Laravel 10
- Frontend: VueJS 2.0, npm 6.14.8, nodeJS v14.15.1
- База данных: PostgreSQL 14.0
- Composer v2.*

Установка проекта

Установку рекомендуется начать с клонирования проекта в папку сайта на сервере с git репозитория стандартной командой: git clone https://gitlab.com/longcatdev/rezervsil.

Далее требуется выполнить следующие действия:

- Создать базу данных любым удобным способом
- Настроить конфигурационный файл .env

```
APP_NAME="Rezervsil"
APP_ENV=local
APP_KEY=base64:7Fee3cCt1YeXfq7PwgUr7pbCEk3KDppcibQpNhOPELw=
APP_DEBUG=true
APP_LOG_LEVEL=debug
APP_URL=http://rezervsil

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=pgsql
DB_HOST=127.0.0.1
DB_PORT=5432
DB_DATABASE=rezervsil
DB_USERNAME=postgres
DB_PASSWORD=postgres

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
```

```
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"

JWT_TTL=1440
JWT_SECRET=Mw7wc1EW4rKRwaQCZwmQ7cHda8P4LI9kQXADSnQM2E3gdzyV0P3G4JB5lrcOGAWj

API_KEY_MANGO=
API_SECRET_MANGO=
API_URL_MANGO=

JWT_ALGO=HS256
```

Запустить следующие команды по порядку

- composer install
- php artisan key:generate

> После выполнения команды в файле .env должен обновится параметр APP_KEY Пример: APP_KEY=base64:my1tfgTTN7MH7GJaTjIRMbYbW18y81irnCLB7/cbUxY=

- php artisan jwt:secret - генерация JWT ключа

> После выполнения команды в файле .env должен обновится параметр JWT_SECRET Пример: JWT_SECRET=TpO4KsHnm3D5pBZ1qIBcTRWcJQdGl9L3TYyTsIA5inJS2LQYwqbjaf74pSd9XjmE

- php artisan migrate --seed
- php artisan storage:link
- npm ci
- php artisan cache:clear
- php artisan config:clear
- php artisan optimize:clear
- php artisan config:cache
- npm run dev (для локальной разработки)
- npm run build (для развороте на сервере)

Service layer

Описание

Сервисный слой (Service layer) — это шаблон проектирования, который инкапсулирует бизнес логику вашего приложения и определяет границу и набор допустимых операций с точки зрения взаимодействующих с ним клиентов.

Расположение

Расположение сервис классов: app/Services

Структура: каждый класс хранится в директории соответствующей сущности для которой он разрабатывался. Для модуля "Сотрудники" директория "Employees/EmployeeService.php"

Пример использования

Инициализация

Инициализация класса в базовом использовании происходит в конструкторе класса контроллера

```
use App\Services\Clients\ClientService as Service;
use App\Services\Contacts\ContactService;
use App\Services\LiveFeeds\LiveFeedService;
...
public function __construct(
    protected Service $service,
    protected ContactService $contactService,
    protected LiveFeedService $liveFeedService
) {
...
}
```

Взаимодействие

Для взаимодействия с сервисом используется назначенное свойство сервиса "$this->service" или необходимое Методы используемы из метода контроллера должны иметь область видимости public

```
public function store(EmployeeStoreRequest $request): void
{
    $this->service->store(EmployeeRequestDTO::fromRequest($request));
}
```

Использование

app/Services/EmployeeService::class Название метода сервиса обычно(по возможности) соответствует названию метода контроллера от куда он вызывается. Принимает в себя типизированные параметры и отображает тип возвращаемого ответа Метод содержит реализацию основной бизнес логики для конкретной задачи, в данном случае происходит сохранение модели Клиентов с данными заранее прошедшими валидацию(Можно указать ссылку на доку с валидацией), и с установленными типами данных с помощью паттерна DTO(ссылка на пример использования технологий DTO)

```
public function store(EmployeeRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

Использование в модулях

Controllers

Пример вызова метода store(сохранение) у заранее инициализированного класса сервиса

```
public function store(ClientStoreRequest $request): void
{
    $this->service->store(ClientRequestDTO::fromRequest($request));
}
```

Jobs

Пример вызова метода getTodayBirthdays из ClientService при выполнении Job

```
public function handle(): mixed
{
    return app(ClientService::class)->getTodayBirthdays();
}
```

Console commands

Бизнес логика выполнения команды реализуется в сервис классах Структура: каждый класс хранится в директории "Commands" и в дальнейшем соответствующей сущности для которой он разрабатывался. Для модуля "Клиенты" директория "app/Services/Commands/Clients/ParseClientService.php" Инициализация класса сервиса происходит в конструкторе с дальнейшей реализацией в методе handle

```
public function __construct(private QueueSimActionService $service)
{
    parent::__construct();
}
...
public function handle()
{
    $this->all = $this->option('all');
    $this->class = $this->option('class');
    $this->simId = $this->option('simId');
    $this->personalAccountId = $this->option('personalAccountId');

    return $this->service->processing(
        class:              $this->class,
        personalAccountId:  $this->personalAccountId,
        simId:              $this->simId,
        all:                $this->all
    );
}
```

Dependency Injection

Описание

Реализовывает слабосвязанную архитектуру, чтобы получить более тестируемый, сопровождаемый и расширяемый код.

Пример использования

Инициализация

В конструктор класса передается объект, который необходимо связать с создаваемым. Тем самым создавая объект, мы связываем этот объект с переданным ClientService. Так задается связанность между этими двумя классами. Данный подход позволяет при необходимости подменить EmployeeService на другой с таким же набором методов, что позволяет сделать код более гибким. Данный подход часто используется в других паттернах, например: Service layer, Repository

```
public function __construct(protected EmployeeService $service)
{
    //
}
```

Взаимодействие

После создания объекта с зависимостью, можно вызывать метод обращаясь к классу как к свойству объекта. Метод create должен отдавать данные для создания модели, так чтобы наш класс не был зависим от реализации мы используем DI и получаем данные вызовом $this->service->create()

```
public function create(): JsonResponse
{
    return response()->json($this->service->create());
}
```

Datatable

Описание

Пакет yajra/laravel-datatables используется для обработки данных через фасад DataTables. Он позволяет получать набор данных, фильтровать их, сортировать и изменять формат вывода

Расположение

Используем 3 метода для работы с данным пакетом. В Service классе в котором необходимо получить набор данных для таблицы создаем три метода: datatable - собирает данные через фасад DataTables query - возвращает запрос для получения данных, который отдается в datatable applyFilters - фильтрует данные по реквесту пользователя

Пример использования

Взимодействие

Для получения готовых данных вызываем метод datatable из севрис класса в контроллере и готовим его для ответа методом toJson. Полученный реквест преобразуем в DTO для фильтров

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request))->toJson();
}
```

Использование

В методе сервиса datatable используем фасад DataTables и вызываем один из методов, в зависимости от типа данных, который возвращает query: Eloquent (Eloquent Builder), Query (Query Builder), Collection (Collection). Метод setTransformer формирует полученный набор данных из запроса в то, что вернет данные метод. Возвращаем ресурс DatatableResource тем самым данные будут в формате описанным в ресурсе. Метод orderColumn задает порядок сортировки в зависимости реквеста. Первым параметром передается названия поля для сортировки, вторым замыкание в котором сортируется запрос Методы filter фильтрует запрос по реквесту. Параметром передается замыкание в котором вызывается метод applyFilters. В него передаем запрос и дто с фильтрами

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
            'workingShifts' => function ($query) {
                $query->select('id', 'name');
            }
        ])->select('employees.*'))
        ->setTransformer(function ($model) {
            return EmployeeDTResource::make($model)->resolve();
        })
        ->with('counter', function () {
            return $this->model->count();
        })
        ->filterColumn('phones', function ($query, $keyword) use ($request) {
            if (preg_match("/^\d+$/", phone_system_format($request->search['value']))) {
                $query->where('phones', 'LIKE', "%" . phone_system_format($request->search[
                    ->orWhere('phones', 'LIKE', "%" . $request->search['value'] . "%");
            }
        })
        ->filterColumn('gender', function ($query, $keyword) {
            $query->enumSearch($keyword);
        })
        ->filterColumn('birthdate', function ($query, $keyword) use ($request) {
            foreach (['Y-m-d', 'd.m.Y'] as $format) {
                try {
                    $date = Carbon::createFromFormat($format, $request->search['value']);
                    if ($date && $date->format($format) === $request->search['value']) {
                        $query->where('birthdate', 'LIKE', $date->format('Y-m-d'));
                    }
                } catch (\Throwable $th) {
                }
            }
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}

public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->working_shift)) {
        $query->whereHas('workingShifts', function ($query) use ($request) {
            $query->whereId($request->working_shift);
```

```
        });
    }
    if (isset($request->gender)) {
        $query->where('gender', $request->gender);
    }
}
```

В методе сервиса applyFilters фильтруется запрос $query в зависимости от переданного $dto. В данном случае, если пришел параметр user_id, то ограничиваем запрос и получаем клиентов только по переданному user_id. Если пришел параметр type, то получаем клиентов определенного типа

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->working_shift)) {
        $query->whereHas('workingShifts', function ($query) use ($request) {
            $query->whereId($request->working_shift);
        });
    }
    if (isset($request->gender)) {
        $query->where('gender', $request->gender);
    }
}
```

DTO

Описание

DTO представляют собой особый тип объектов, которые используются для передачи данных между различными слоями приложения или между разными приложениями. Они содержат только необходимые поля и методы для передачи данных и обладают простыми структурами. Для DTO используем пакет spatie/laravel-data

Расположение

Классы дто располагаются в app\DTO с добавлением в папку, связанной с логикой данных. Класс следует называть по набору данных, который он возвращает. Например дто, который будет хранить данные для сохранения модели Client, будет называться ClientStoreDTO и располагаться в папке Clients

Пример испозользования

Инициализация

В классе дто указываются типизированные свойства для данных, которые он будет хранить

```
class UserStoreDTO extends Data
{
    public string $full_name;

    public string $email;

    public Collection $phones;

    public string $role_name;

    public string $password;
    ...
}
```

Так же создается статический метод fromRequest который будет формировать данные и вернет дто. Такой метод необходимо создавать, если дто предназначено для реквеста

```
public static function fromRequest(FormRequest $request): static
{
    $dto = static::from([
        'full_name'     => $request->validated('full_name'),
        'email'         => $request->validated('email'),
        'phones'        => array_to_dto(ContactStoreDTO::class, $request->validated('phones
        'role_name'     => $request->validated('role_name'),
        'password'      => $request->validated('password')
    ]);
    $dto->phonesIsUnique();
    return $dto;
}
```

Использование

Вызов статического метода ClientTableDTO::fromRequest($request) вернет объект дто ClientTableDTO с набором данных переданным в $request

```
public function datatable(ClientTableRequest $request): JsonResponse
{
    return $this->service->datatable(ClientTableDTO::fromRequest($request))->toJson();
}
```

Также дто ClientTableDTO можно создать передав массив данных в конструктор. Но в этом случае ключи переданного массива должны совпадать с названием свойств дто

```
public function datatable(ClientTableRequest $request): JsonResponse
{
    return $this->service->datatable(new ClientTableDTO($request->validated()))->toJson();
}
```

Использование в модели

Если в модели присутствует json поле то для его описания следует создать класс дто. После добавить аттрибут в модель на это поле с методоми get и set. get - преобразует полученный json из бд в дто ColorDTO set - при сохранение модели дто ColorDTO будет преобразовано в json строку Так при работе с моделью у которой есть json поле, всегда будем получать вместо него класс DTO

```php
public function color(): Attribute
{
    return Attribute::make(
        get: fn (string $color) => ColorDTO::from(json_decode($color, true)),
        set: fn (ColorDTO $color) => json_encode($color->toArray(), JSON_UNESCAPED_UNICODE)
    );
}
```

Enums

Описание

Перечисления — это ограничивающий слой над классами и константами классов, предназначенный для предоставления способа определения закрытого набора возможных значений для типа. Используются перечисления, представленные php

Расположение

Перечисления располагаются в app\Enums с добавлением в папку, связанной с логикой данных. Перечисление следует называет по набору данных, который он хранит. Например для модели Client перечисление для хранения его типов будет называться EmployeeTypesEnum и располагаться в папке Employees

Пример использования

Инициализация

В case задаются возможные варианты перечисления и указывается их значение

```
final class EmployeeStatusEnum extends Enum implements LocalizedEnum
{
    public const ACTIVE = 'active';
    public const NO_ACTIVE = 'no_active';
    ...
}
```

BrigadeEmployeeRequestDTO

App\DTO\Brigades

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?array $working_shifts;
public string $full_name;
public string $birthdate;
public array $phones;
public ?string $passport_serial_number;
public ?string $issued;
public ?string $issue_date;
public ?string $registration_address;
public ?string $comment;
```

Список методов

fromRequest

Параметры:

- BrigadeEmployeeStoreRequest $request - валидированные входящие данные

Ответ : `return \App\DTO\Brigades\BrigadeEmployeeRequestDTO`

```
public static function fromRequest(BrigadeEmployeeStoreRequest $request): BrigadeEmployeeRe
{
    return new BrigadeEmployeeRequestDTO(
        working_shifts:  $request->validated('working_shifts'),
        full_name:  $request->validated('full_name'),
        birthdate:  $request->validated('birthdate'),
        phones:  $request->validated('phones'),
        passport_serial_number:  $request->validated('passport_serial_number'),
        issued:  $request->validated('issued'),
        issue_date:  $request->validated('issue_date'),
        registration_address:  $request->validated('registration_address'),
        comment:  $request->validated('comment')
    );
}
```

BrigadeFilterRequestDTO

App\DTO\Brigades

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?string $working_shift_date;
public ?array $customer;
```

Список методов

fromRequest

Параметры:

- BrigadeFilterRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Brigades\BrigadeFilterRequestDTO
```

```
public static function fromRequest(BrigadeFilterRequest $request): BrigadeFilterRequestDTO
{
    return new BrigadeFilterRequestDTO(
        working_shift_date: $request->validated('working_shift_date'),
        customer: $request->validated('customer')
    );
}
```

BrigadeRequestDTO

App\DTO\Brigades

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $customer_id;
public ?int $employee_id;
public int $subdivision_id;
```

Список методов

fromRequest

Параметры:

- BrigadeStoreRequest|BrigadeUpdateRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Brigades\BrigadeFilterRequestDTO
```

```
public static function fromRequest(BrigadeStoreRequest|BrigadeUpdateRequest $request): Brig
{
    return new BrigadeRequestDTO(
        customer_id: $request->validated('customer_id'),
        employee_id: $request->validated('employee_id'),
        subdivision_id: $request->validated('subdivision_id')
    );
}
```

BrigadeStatusRequestDto

App\DTO\Brigades

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $brigade_status_id;
public ?string $day_off_start_date;
public ?string $day_off_end_date;
```

Список методов

fromRequest

Параметры:

- BrigadeEmployeeStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Brigades\BrigadeStatusRequestDTO
```

```
public static function fromRequest(BrigadeUpdateStatusRequest $request): BrigadeStatusReque
{
    return new BrigadeStatusRequestDTO(
        brigade_status_id: $request->validated('brigade_status_id'),
        day_off_start_date: $request->validated('day_off_start_date'),
        day_off_end_date: $request->validated('day_off_end_date')
    );
}
```

WorkShiftRequestDTO

App\DTO\Brigades

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $employee_id;
public int $subdivision_id;
```

Список методов

fromRequest

Параметры:

- WorkShiftRequestDTO $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Brigades\WorkShiftRequest
```

```
public static function fromRequest(WorkShiftRequest $request): WorkShiftRequestDTO
{
    return new WorkShiftRequestDTO(
        employee_id: $request->validated('employee_id'),
        subdivision_id: $request->validated('subdivision_id')
    );
}
```

CustomerRequestDTO

App\DTO\Customers

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public string $actual_location;
public ?int $working_hours_count;
public ?int $employees_involved_count;
public string $current_contract_number;
public ?string $contract_start_date;
public ?string $contract_end_date;
public ?string $current_additional_agreement_number;
public int $billing_period_id;
public array $rate_customers;
public array $rate_employees;
public int $cost_transport_there_customer;
public int $cost_transport_back_customer;
public int $cost_transport_round_trip_customer;
public int $cost_transport_there_tk;
public int $cost_transport_back_tk;
public int $cost_transport_round_trip_tk;
public int $brigadiers_x1_rub;
public int $brigadiers_x2_rub;
public int $brigadiers_personal_rub;
public int $legal_entity_id;
public string $active;
public ?string $comment;
```

Список методов

fromRequest

Параметры:

- CustomerStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Customers\CustomerRequestDTO
```

```
public static function fromRequest(CustomerStoreRequest $request): CustomerRequestDTO
{
    return new CustomerRequestDTO(
        name: $request->validated('name'),
        actual_location: $request->validated('actual_location'),
        working_hours_count: $request->validated('working_hours_count'),
        employees_involved_count: $request->validated('employees_involved_count'),
        current_contract_number: $request->validated('current_contract_number'),
        contract_start_date: $request->validated('contract_start_date'),
        contract_end_date: $request->validated('contract_end_date'),
        current_additional_agreement_number: $request->validated('current_additional_agreem
        billing_period_id: $request->validated('billing_period_id'),
        rate_employees: $request->validated('rate_employees'),
        rate_customers: $request->validated('rate_customers'),
        cost_transport_there_customer: $request->validated('cost_transport_there_customer')
        cost_transport_back_customer: $request->validated('cost_transport_back_customer'),
        cost_transport_round_trip_customer: $request->validated('cost_transport_round_trip_
        cost_transport_there_tk: $request->validated('cost_transport_there_tk'),
        cost_transport_back_tk: $request->validated('cost_transport_back_tk'),
        cost_transport_round_trip_tk: $request->validated('cost_transport_round_trip_tk'),
        brigadiers_x1_rub: $request->validated('brigadiers_x1_rub'),
        brigadiers_x2_rub: $request->validated('brigadiers_x2_rub'),
        brigadiers_personal_rub: $request->validated('brigadiers_personal_rub'),
        legal_entity_id: $request->validated('legal_entity_id'),
        active: $request->validated('active'),
        comment: $request->validated('comment')
    );
}
```

CustomerRequisiteRequestDTO

App\DTO\Customers

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $customer_id;
public ?string $name;
public ?string $inn;
public ?string $kpp;
public ?string $orgn;
public ?string $legal_address;
public ?string $phone;
public ?string $email;
public ?string $director_name;
public ?string $director_position;
public ?string $foundation;
public ?array $bank_requisites;
public ?array $contacts;
```

Список методов

fromRequest

Параметры:

- CustomerRequisiteStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Customers\CustomerRequisiteRequestDTO
```

```php
public static function fromRequest(CustomerRequisiteStoreRequest $request): CustomerRequisi
{
    return new CustomerRequisiteRequestDTO(
        customer_id: $request->validated('customer_id'),
        name: $request->validated('name'),
        inn: $request->validated('inn'),
        kpp: $request->validated('kpp'),
        orgn: $request->validated('orgn'),
        legal_address: $request->validated('legal_address'),
        phone: $request->validated('phone'),
        email: $request->validated('email'),
        director_name: $request->validated('director_name'),
        director_position: $request->validated('director_position'),
        foundation: $request->validated('foundation'),
        bank_requisites: $request->validated('bank_requisites'),
        contacts: $request->validated('contacts'),
    );
}
```

DashboardFilterRequestDTO

App\DTO\Dashboard

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?array $period;
public ?array $customer;
public ?array $user;
public ?array $legal_entity;
public ?string $data_type;
public ?string $type;
```

Список методов

fromRequest

Параметры:

- DashboardFilterRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Customers\DashboardFilterRequestDTO
```

```
public static function fromRequest(DashboardFilterRequest $request): DashboardFilterRequest
{
    return new DashboardFilterRequestDTO(
        period: $request->validated('period'),
        customer: $request->validated('customer'),
        user: $request->validated('user'),
        data_type: $request->validated('data_type'),
        type: $request->validated('type'),
        legal_entity: $request->validated('legal_entity')
    );
}
```

DatatableColumnStoreRequestDTO

App\DTO\Datatables

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?array $period;
public ?array $customer;
public ?array $user;
public ?array $legal_entity;
public ?string $data_type;
public ?string $type;
```

Список методов

fromRequest

Параметры:

- DatatableColumnStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Datatables\DatatableColumnStoreRequestDTO
```

```
public static function fromRequest(DatatableColumnStoreRequest $request): DatatableColumnSt
{
    return new DatatableColumnStoreRequestDTO(
        columns: $request->validated('columns')
    );
}
```

DatatableRequestDTO

App\DTO\Datatables

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?bool $datatable;
public ?int $length;
public ?int $start;
public ?int $draw;
public ?array $columns;
public ?array $order;
public ?array $search;
public ?array $period;
public ?bool $status;
public ?array $role;
public ?array $user;
public ?array $customer;
public ?string $section;
public ?string $gender;
public ?int $rating;
public ?int $working_shift;
public ?int $employee_status;
public ?int $documentable_id;
public ?string $documentable_type;
public ?string $month;
public ?string $working_shift_date;
public ?int $order_id;
public ?int $employee_id;
public ?int $employee;
public ?int $employee_rating;
public ?string $last_object;
```

Список методов

fromRequest

Параметры:

- DatatableRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Datatables\DatatableRequestDTO
```

```
public static function fromRequest(DatatableRequest $request): DatatableRequestDTO
{
    return new DatatableRequestDTO(
        datatable: $request->validated('datatable'),
        length: $request->validated('length'),
        start: $request->validated('start'),
        draw: $request->validated('draw'),
        columns: $request->validated('columns'),
        order: $request->validated('order'),
        search: $request->validated('search'),
        period: $request->validated('period'),
        status: $request->validated('status'),
        role: $request->validated('role'),
        user: $request->validated('user'),
        customer: $request->validated('customer'),
        section: $request->validated('section'),
        gender: $request->validated('gender'),
        rating: $request->validated('rating'),
        working_shift: $request->validated('working_shift'),
        documentable_id: $request->validated('documentable_id'),
        documentable_type: $request->validated('documentable_type'),
        employee_status: $request->validated('employee_status'),
        month: $request->validated('month'),
        working_shift_date: $request->validated('working_shift_date'),
        order_id: $request->validated('order_id'),
        employee_id: $request->validated('employee_id'),
        employee: $request->validated('employee'),
        employee_rating: $request->validated('employee_rating'),
        last_object: $request->validated('last_object')
    );
}
```

DocumentRequestDTO

App\DTO\Documents

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public ?UploadedFile $file;
public ?string $comment;
public int $documentable_id;
public string $documentable_type;
```

Список методов

fromRequest

Параметры:

- $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Documents\DocumentStoreRequest
```

```
public static function fromRequest(DocumentStoreRequest $request): DocumentRequestDTO
{
    return new DocumentRequestDTO(
        name: $request->validated('name'),
        file: $request->validated('file'),
        comment: $request->validated('comment'),
        documentable_id: $request->validated('documentable_id'),
        documentable_type: $request->validated('documentable_type'),
    );
}
```

EmployeeImportRequestDTO

App\DTO\Employees

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public UploadedFile $file;
```

Список методов

fromRequest

Параметры:

- EmployeeImportRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Employees\EmployeeImportRequestDTO
```

```
public static function fromRequest(EmployeeImportRequest $request): EmployeeImportRequestDT
{
    return new EmployeeImportRequestDTO(
        file:  $request->validated('file')
    );
}
```

DocumentRequestDTO

App\DTO\Employees

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```php
public ?UploadedFile $avatar;
public ?bool $delete_avatar;
public string $card_created_at;
public string $user_id;
/** @var array<int, int> */
public array $working_shifts;
public string $full_name;
public string $birthdate;
/** @var array<int, string> */
public array $phones;
/** @var array<int, string> */
public ?array $additional_phones;
public string $gender;
public string $passport_serial_number;
public string $issued;
public string $issue_date;
public string $registration_address;
public string $fact_address;
public ?string $job_resource_id;
public ?string $whence;
public ?string $skills;
public string $health_restrictions;
public string $medical_book;
public string $self_employed;
public ?string $inn;
public ?string $payment_account;
public ?string $correspondent_account;
public ?string $bik;
public ?string $bank_name;
public ?bool $active;
public ?string $comment;
```

Список методов

fromRequest

Параметры:

- EmployeeStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Employees\EmployeeRequestDTO
```

```php
public static function fromRequest(EmployeeStoreRequest $request): EmployeeRequestDTO
{
    return new EmployeeRequestDTO(
        avatar: $request->validated('avatar'),
        delete_avatar: $request->validated('delete_avatar'),
        card_created_at: $request->validated('card_created_at'),
        user_id: $request->validated('user_id'),
        working_shifts: $request->validated('working_shifts'),
        full_name: $request->validated('full_name'),
        birthdate: $request->validated('birthdate'),
        phones: $request->validated('phones'),
        additional_phones: $request->validated('additional_phones'),
        gender: $request->validated('gender'),
        passport_serial_number: $request->validated('passport_serial_number'),
        issued: $request->validated('issued'),
        issue_date: $request->validated('issue_date'),
        registration_address: $request->validated('registration_address'),
        fact_address: $request->validated('fact_address'),
        job_resource_id: $request->validated('job_resource_id'),
        whence: $request->validated('whence'),
        skills: $request->validated('skills'),
        health_restrictions: $request->validated('health_restrictions'),
        medical_book: $request->validated('medical_book'),
        self_employed: $request->validated('self_employed'),
        inn: $request->validated('inn'),
        payment_account: $request->validated('payment_account'),
        correspondent_account: $request->validated('correspondent_account'),
        bik: $request->validated('bik'),
        bank_name: $request->validated('bank_name'),
        active: $request->validated('active'),
        comment: $request->validated('comment')
    );
}
```

EventHistoryDataRequestDTO

App\DTO\Employees

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```php
/** @var array<string, mixed>|null */
public ?array $working_shift;
/** @var array<string, mixed>|null */
public ?array $customer;
/** @var array<string, mixed>|null */
public ?array $employee_status;
/** @var array<string, mixed>|null */
public ?array $employee_rating;
public ?bool $comment;
public ?bool $payment;
public ?string $search;
```

Список методов

fromRequest

Параметры:

- EventHistoryDataRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Employees\EventHistoryDataRequestDTO
```

```
public static function fromRequest(EventHistoryDataRequest $request): EventHistoryDataReque
{
    return new EventHistoryDataRequestDTO(
        working_shift: $request->validated('working_shift'),
        customer: $request->validated('customer'),
        employee_status: $request->validated('employee_status'),
        employee_rating: $request->validated('employee_rating'),
        comment: $request->validated('comment'),
        payment: $request->validated('payment'),
        search: $request->validated('search')
    );
}
```

IncomingRequestDTO

App\DTO\Incomings

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $payment_date;
public int $legal_entity_id;
public int $customer_id;
public int $sum;
public ?string $comment;
```

Список методов

fromRequest

Параметры:

- IncomingStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Incomings\IncomingRequestDTO
```

```
public static function fromRequest(IncomingStoreRequest $request): IncomingRequestDTO
{
    return new IncomingRequestDTO(
        payment_date: $request->validated('payment_date'),
        legal_entity_id: $request->validated('legal_entity_id'),
        customer_id: $request->validated('customer_id'),
        sum: $request->validated('sum'),
        comment: $request->validated('comment'),
    );
}
```

InvoicePaymentRequestDTO

App\DTO\Invoices

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $payment_date;
    public int $sum;
```

Список методов

fromRequest

Параметры:

- InvoicePaymentRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Invoices\InvoicePaymentRequestDTO
```

```
public static function fromRequest(InvoicePaymentRequest $request): InvoicePaymentRequestDT
    {
        return new InvoicePaymentRequestDTO(
            payment_date: $request->validated('payment_date'),
            sum: $request->validated('sum')
        );
    }
```

InvoiceRequestDTO

App\DTO\Invoices

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $payment_date;
public int $legal_entity_id;
public int $customer_id;
public int $sum;
public ?string $comment;
```

Список методов

fromRequest

Параметры:

- InvoiceStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Invoices\InvoiceRequestDTO
```

```
public static function fromRequest(InvoiceStoreRequest $request): InvoiceRequestDTO
{
    return new InvoiceRequestDTO(
        date: $request->validated('date'),
        customer_id: $request->validated('customer_id'),
        invoice: $request->validated('invoice'),
        legal_entity_id: $request->validated('legal_entity_id'),
        sum: $request->validated('sum'),
        paymented: $request->validated('paymented'),
        paymented_date: $request->validated('paymented_date'),
        comment: $request->validated('comment'),
    );
}
```

LegalEntityRequestDTO

App\DTO\LegalEntities

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?string $name;
public ?string $inn;
public ?string $kpp;
public ?string $orgn;
public ?string $legal_address;
public ?string $phone;
public ?string $email;
public ?string $director_name;
public ?string $director_position;
public ?string $foundation;
public ?array $bank_requisites;
public ?array $contacts;
```

Список методов

fromRequest

Параметры:

- $request - валидированные входящие данные

Ответ :

```
return \App\DTO\LegalEntities\LegalEntityRequestDTO
```

```
public static function fromRequest(LegalEntityStoreRequest $request): LegalEntityRequestDTO
{
    return new LegalEntityRequestDTO(
        name: $request->validated('name'),
        inn: $request->validated('inn'),
        kpp: $request->validated('kpp'),
        orgn: $request->validated('orgn'),
        legal_address: $request->validated('legal_address'),
        phone: $request->validated('phone'),
        email: $request->validated('email'),
        director_name: $request->validated('director_name'),
        director_position: $request->validated('director_position'),
        foundation: $request->validated('foundation'),
        bank_requisites: $request->validated('bank_requisites'),
        contacts: $request->validated('contacts'),
    );
}
```

MainFilterRequestDTO

App\DTO\Main

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?string $name;
public ?string $inn;
public ?string $kpp;
public ?string $orgn;
public ?string $legal_address;
public ?string $phone;
public ?string $email;
public ?string $director_name;
public ?string $director_position;
public ?string $foundation;
public ?array $bank_requisites;
public ?array $contacts;
```

Список методов

fromRequest

Параметры:

- MainFilterRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Main\MainFilterRequestDTO
```

```
public static function fromRequest(MainFilterRequest $request): MainFilterRequestDTO
{
    return new MainFilterRequestDTO(
        working_shift_date: $request->validated('working_shift_date'),
        customer_id: $request->validated('customer_id')
    );
}
```

NotificationStoreDTO

App\DTO\Notifications

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $message;
public int $user_id;
```

Список методов

fromRequest

Параметры:

- array $request - валидированные входящие данные

Ответ :

```
return App\DTO\Notifications\NotificationStoreDTO
```

```
public static function fromRequest(array $request): NotificationStoreDTO
{
    return new NotificationStoreDTO(
        message: $request['message'],
        user_id: $request['user_id']
    );
}
```

OrderEmployeeStoreRequestDTO

App\DTO\Orders

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $order_id;
public string $working_shift_date;
public string $subdivision_id;
public string $customer_id;
public string $working_shift_id;
```

Список методов

fromRequest

Параметры:

- OrderEmployeeStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Orders\OrderEmployeeStoreRequestDTO
```

```
public static function fromRequest(OrderEmployeeStoreRequest $request): OrderEmployeeStoreR
{
    return new OrderEmployeeStoreRequestDTO(
        order_id: $request->validated('order_id'),
        working_shift_date: $request->validated('working_shift_date'),
        subdivision_id: $request->validated('subdivision_id'),
        customer_id: $request->validated('customer_id'),
        working_shift_id: $request->validated('working_shift_id')
    );
}
```

OrderEmployeeUpdateRequestDTO

App\DTO\Orders

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $employee_id;
public string $subdivision_id;
```

Список методов

fromRequest

Параметры:

- OrderEmployeeUpdateRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Orders\OrderEmployeeUpdateRequestDTO
```

```
public static function fromRequest(OrderEmployeeUpdateRequest $request): OrderEmployeeUpdat
{
    return new OrderEmployeeUpdateRequestDTO(
        employee_id: $request->validated('employee_id'),
        subdivision_id: $request->validated('subdivision_id')
    );
}
```

OrderRequestDTO

App\DTO\Orders

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $customer_id;
public ?bool $is_master_order;
public string $type;
public string $order_date;
public string $working_shift_date;
public ?string $working_shift_end_date;
public array $order_items;
```

Список методов

fromRequest

Параметры:

- OrderStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Orders\OrderRequestDTO
```

```
public static function fromRequest(OrderStoreRequest $request): OrderRequestDTO
{
    return new OrderRequestDTO(
        customer_id: $request->validated('customer_id'),
        is_master_order: $request->validated('is_master_order'),
        type: $request->validated('type'),
        order_date: $request->validated('order_date'),
        working_shift_date: $request->validated('working_shift_date'),
        working_shift_end_date: $request->validated('working_shift_end_date'),
        order_items: $request->validated('order_items')
    );
}
```

OrderUpdateRequestDTO

App\DTO\Orders

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $customer_id;
public string $order_date;
public string $working_shift_date;
public ?string $working_shift_end_date;
public string $cost_transportation;
public string $deadline;
public string $male;
public string $female;
public string $number_buses_there;
public string $number_buses_back;
public string $number_buses_there_back;
public string $place_shipment;
public string $plan;
public string $rate_employee;
public string $working_hour_id;
public string $subdivision_id;
public string $working_shift_id;
public string $working_shift_time;
public string $working_shift_start_time;
public string $working_shift_end_time;
```

Список методов

fromRequest

Параметры:

- OrderUpdateRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Orders\OrderUpdateRequestDTO
```

```php
public static function fromRequest(OrderUpdateRequest $request): OrderUpdateRequestDTO
{
    return new OrderUpdateRequestDTO(
        customer_id: $request->validated('customer_id'),
        order_date: $request->validated('order_date'),
        working_shift_date: $request->validated('working_shift_date'),
        working_shift_end_date: $request->validated('working_shift_end_date'),
        cost_transportation: $request->validated('cost_transportation'),
        deadline: $request->validated('deadline'),
        male: $request->validated('male'),
        female: $request->validated('female'),
        number_buses_there: $request->validated('number_buses_there'),
        number_buses_back: $request->validated('number_buses_back'),
        number_buses_there_back: $request->validated('number_buses_there_back'),
        place_shipment: $request->validated('place_shipment'),
        plan: $request->validated('plan'),
        rate_employee: $request->validated('rate_employee'),
        working_hour_id: $request->validated('working_hour_id'),
        subdivision_id: $request->validated('subdivision_id'),
        working_shift_id: $request->validated('working_shift_id'),
        working_shift_time: $request->validated('working_shift_time'),
        working_shift_start_time: $request->validated('working_shift_start_time'),
        working_shift_end_time: $request->validated('working_shift_end_time')
    );
}
```

EmployeeBonusRequestDTO

App\DTO\References\EmployeeBonuses

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $working_shift_count;
public string $bonus;
public ?array $subdivisions;
```

Список методов

fromRequest

Параметры:

- EmployeeBonusStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\EmployeeBonuses\EmployeeBonusRequestDTO
```

```
public static function fromRequest(EmployeeBonusStoreRequest $request): EmployeeBonusReques
{
    return new EmployeeBonusRequestDTO(
        working_shift_count: $request->validated('working_shift_count'),
        bonus: $request->validated('bonus'),
        subdivisions: $request->validated('subdivisions'),
    );
}
```

EmployeeRatingRequestDTO

App\DTO\References\EmployeeRatings

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public string $absence_count;
public string $days_count;
```

Список методов

fromRequest

Параметры:

- EmployeeRatingStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\EmployeeRatings\EmployeeRatingRequestDTO
```

```
public static function fromRequest(EmployeeRatingStoreRequest $request): EmployeeRatingRequ
{
    return new EmployeeRatingRequestDTO(
        name: $request->validated('name'),
        absence_count: $request->validated('absence_count'),
        days_count: $request->validated('days_count'),
    );
}
```

EmployeeStatusRequestDTO

App\DTO\References\EmployeeStatuses

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public string $description;
```

Список методов

fromRequest

Параметры:

- $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\EmployeeStatuses\EmployeeStatusRequestDTO
```

```
public static function fromRequest(EmployeeStatusStoreRequest $request): EmployeeStatusRequ
{
    return new EmployeeStatusRequestDTO(
        name: $request->validated('name'),
        description: $request->validated('description'),
    );
}
```

PenaltyRequestDTO

App\DTO\References\Penalties

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public string $description;
```

Список методов

fromRequest

Параметры:

- PenaltyStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\Penalties\PenaltyRequestDTO
```

```
public static function fromRequest(PenaltyStoreRequest $request): PenaltyRequestDTO
{
    return new PenaltyRequestDTO(
        name: $request->validated('name')
    );
}
```

PercentageBonusRequestDTO

App\DTO\References\PercentageBonuses

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public string $absence;
public string $bonus;
```

Список методов

fromRequest

Параметры:

- PercentageBonusStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\PercentageBonuses\PercentageBonusRequestDTO
```

```
public static function fromRequest(PercentageBonusStoreRequest $request): PercentageBonusRe
{
    return new PercentageBonusRequestDTO(
        name: $request->validated('name'),
        absence: $request->validated('absence'),
        bonus: $request->validated('bonus'),
    );
}
```

SubdivisionRequestDTO

App\DTO\References\Subdivisions

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public int $customer_id;
```

Список методов

fromRequest

Параметры:

- SubdivisionStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\Subdivisions\SubdivisionRequestDTO
```

```
public static function fromRequest(SubdivisionStoreRequest $request): SubdivisionRequestDTO
{
    return new SubdivisionRequestDTO(
        name: $request->validated('name'),
        customer_id: $request->validated('customer_id'),
    );
}
```

WorkingHourRequestDTO

App\DTO\References\WorkingHours

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $name;
```

Список методов

fromRequest

Параметры:

- WorkingHourStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\WorkingHours\WorkingHourRequestDTO
```

```
public static function fromRequest(WorkingHourStoreRequest $request): WorkingHourRequestDTO
{
    return new WorkingHourRequestDTO(
        name: $request->validated('name'),
    );
}
```

ReferenceRequestDTO

App\DTO\References

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
```

Список методов

fromRequest

Параметры:

- ReferenceStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\References\ReferenceRequestDTO
```

```
public static function fromRequest(ReferenceStoreRequest $request): ReferenceRequestDTO
{
    return new ReferenceRequestDTO(
        name: $request->validated('name'),
    );
}
```

RoleModelDTO

App\DTO\RoleUsers

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $id;
public string $display_name;
public array $permissions;
```

Список методов

fromRequest

Параметры:

-
- int $id - Идентификатор роли
- string $display_name - Имя роли
- array $permissions - Досутпы

Ответ :

```
return \App\DTO\RoleUsers\RoleModelDTO
```

```
public static function fromModel(int $id, string $display_name, array $permissions): RoleMo
{
    return new RoleModelDTO(
        id: $id,
        display_name: $display_name,
        permissions: $permissions
    );
}
```

RoleRequestDTO

App\DTO\RoleUsers

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $display_name;
public string $color;
public array $modules;
```

Список методов

fromRequest

Параметры:

- RoleRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\RoleUsers\RoleRequest
```

```
public static function fromRequest(RoleRequest $request): RoleRequestDTO
{
    return new RoleRequestDTO(
        display_name: $request->validated('display_name'),
        color: $request->validated('color'),
        modules: $request->validated('modules')
    );
}
```

AccrualRequestDTO

App\DTO\Salary\Accruals

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $employee_id;
public string $date;
public int $sum;
```

Список методов

fromRequest

Параметры:

- array $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Salary\Accruals\AccrualRequestDTO
```

```
public static function fromRequest(array $request): AccrualRequestDTO
{
    return new AccrualRequestDTO(
        employee_id: $request['employee_id'],
        date: $request['date'],
        sum: $request['sum'],
    );
}
```

PayoutRequestDTO

App\DTO\Salary\Payouts

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $employee_id;
public string $date;
public int $sum;
```

Список методов

fromRequest

Параметры:

- PayoutStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Salary\Payouts\PayoutRequestDTO
```

```
public static function fromRequest(PayoutStoreRequest $request): PayoutRequestDTO
{
    return new PayoutRequestDTO(
        employee_id: $request->validated('employee_id'),
        date: $request->validated('date'),
        sum: $request->validated('sum'),
    );
}
```

StatementFileRequestDTO

App\DTO\Salary\Statements

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $name;
public UploadedFile $file;
```

Список методов

fromRequest

Параметры:

- StatementFileStoreRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Salary\Statements\StatementFileRequestDTO
```

```
public static function fromRequest(StatementFileStoreRequest $request): StatementFileReques
{
    return new StatementFileRequestDTO(
        name: $request->validated('name'),
        file: $request->validated('file')
    );
}
```

StatementRequestDTO

App\DTO\Salary\Statements

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?bool $received;
public ?int $received_sum;
public ?string $received_date;
public ?string $comment;
```

Список методов

fromRequest

Параметры:

- StatementUpdateRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Salary\Statements\StatementRequestDTO
```

```
public static function fromRequest(StatementUpdateRequest $request): StatementRequestDTO
{
    return new StatementRequestDTO(
        received: $request->validated('received'),
        received_sum: $request->validated('received_sum'),
        received_date: $request->validated('received_date'),
        comment: $request->validated('comment')
    );
}
```

WorkedShiftFilterRequestDTO

App\DTO\Salary\WorkedShifts

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?string $date;
public ?array $customer;
```

Список методов

fromRequest

Параметры:

- WorkedShiftFilterRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Salary\WorkedShifts\WorkedShiftFilterRequestDTO
```

```
public static function fromRequest(WorkedShiftFilterRequest $request): WorkedShiftFilterReq
{
    return new WorkedShiftFilterRequestDTO(
        date: $request->validated('date'),
        customer: $request->validated('customer')
    );
}
```

WorkedShiftRequestDTO

App\DTO\Salary\WorkedShifts

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $employee_id;
public string $date;
public int $sum;
```

Список методов

fromRequest

Параметры:

- WorkedShiftFilterRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Salary\WorkedShifts\WorkedShiftUpdateRequest
```

```
public static function fromRequest(WorkedShiftUpdateRequest $request): WorkedShiftRequestDT
{
    return new WorkedShiftRequestDTO(
        employee_id: $request->validated('employee_id'),
        date: $request->validated('date'),
        sum: $request->validated('sum'),
    );
}
```

BaseSearchRequestDTO

App\DTO\Search

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public ?int $page;
public ?string $q;
```

Список методов

fromRequest

Параметры:

- BaseSearchRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Search\BaseSearchRequestDTO
```

```
public static function fromRequest(BaseSearchRequest $request): BaseSearchRequestDTO
{
    return new BaseSearchRequestDTO(
        page: $request->validated('page'),
        q: $request->validated('q')
    );
}
```

UserRequestDTO

App\DTO\Users

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public string $full_name;
public ?string $password;
public string $login;
public string $phone;
public ?string $image;
public ?string $card_created_at;
public ?string $employment_date;
public ?string $working_position;
public ?string $birthdate;
public ?string $additional_phone;
public ?string $dismissal_date;
public ?string $comment;
public ?string $coefficient;
public ?string $salary;
public ?string $active;
public ?UploadedFile $avatar;
public ?int $role_id;
public ?int $delete_avatar;
```

Список методов

fromRequest

Параметры:

- UserRequest $request - валидированные входящие данные

Ответ :

```
return \App\DTO\Users\UserRequestDTO
```

```php
public static function fromRequest(UserRequest $request): UserRequestDTO
{
    return new UserRequestDTO(
        full_name: $request->validated('full_name'),
        password: $request->validated('password'),
        login: $request->validated('login'),
        phone: $request->validated('phone'),
        image: $request->validated('image'),
        card_created_at: $request->validated('card_created_at'),
        employment_date: $request->validated('employment_date'),
        working_position: $request->validated('working_position'),
        birthdate: $request->validated('birthdate'),
        additional_phone: $request->validated('additional_phone'),
        dismissal_date: $request->validated('dismissal_date'),
        comment: $request->validated('comment'),
        coefficient: $request->validated('coefficient'),
        salary: $request->validated('salary'),
        active: $request->validated('active'),
        avatar: $request->validated('avatar'),
        role_id: $request->validated('role_id'),
        delete_avatar: $request->validated('delete_avatar'),
    );
}
```

WorkScheduleStoreRequestDTO

App\DTO\WorkSchedules

> Данный класс наследует класс DataTransferObject который предоставляется библиотекой Spatie/data-transfer-object

Поля

```
public int $user_id;
public array $period;
public array $schedule;
```

Список методов

fromRequest

Параметры:

- WorkScheduleStoreRequestDTO $request - валидированные входящие данные

Ответ :

```
return \App\DTO\WorkSchedules\WorkScheduleFormDataRequestDTO
```

```
public static function fromRequest(WorkScheduleStoreRequest $request): WorkScheduleStoreReq
{
    return new WorkScheduleStoreRequestDTO(
        user_id: $request->validated('user_id'),
        period: $request->validated('period'),
        schedule: $request->validated('schedule')
    );
}
```

ActionEnum

App\Enums\ActionHistories

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве возможных действий пользователя.

Константы

```
public const USERS_CREATE = 'users_create';
public const USERS_EDIT = 'users_edit';
public const USERS_DELETE = 'users_delete';
```

SectionEnum

App\Enums\ActionHistories

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве секций системы.

Константы

```
public const ORDERS = 'orders';
public const BRIGADES = 'brigades';
public const CUSTOMERS = 'customers';
public const EMPLOYEES = 'employees';
public const SALARIES = 'salary';
public const USERS = 'users';
public const LEGAL_ENTITIES = 'legal_entities';
public const REFERENCES = 'references';
public const DASHBOARDS = 'dashboards';
```

SubsectionEnum

App\Enums\ActionHistories

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве подсекций системы.

Константы

```
public const ROLES = 'roles';
public const WORK_SCHEDULES = 'work_schedules';
public const PAYOUTS = 'payouts';
public const SHIFT_HISTORY = 'shift_history';
public const PAYMENT_HISTORY = 'payment_history';
public const FACTS = 'facts';
public const INVOICES = 'invoices';
public const PLAN = 'plans';
public const ACCRUALS = 'accruals';
public const STATEMENT = 'statements';
public const SUMMARIES = 'summaries';
public const DASHBOARD_EMPLOYEES = 'dashboard_employees';
public const DASHBOARD_MANAGER_ACTIVITY = 'dashboard_manager_activity';
public const DASHBOARD_ORDERS = 'dashboard_orders';
public const DASHBOARD_PAYMENTS = 'dashboard_payments';
public const DASHBOARD_SEASONAL_ACTIVITY = 'dashboard_seasonal_activity';
public const TRANSPORT_REPORTS = 'transport_reports';
public const INCOMINGS = 'incomings';
public const GENERAL_REPORTS = 'general_reports';
public const WORKED_SHIFTS = 'worked_shifts';
```

BrigadeStatusEnum

App\Enums\Brigades

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве статусов бригад.

Константы

```
public const CAME_OUT = 'came_out';
public const NOT_CAME_OUT = 'not_came_out';
public const DAY_OFF = 'day_off';
public const NOT_CONFIRMED = 'not_confirmed';
```

EmployeeFormsEnum

App\Enums\Employees

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве данных формы сотрудников.

Константы

```
public const INDIVIDUALS = 'individuals';
public const SELF_EMPLOYED = 'self_employed';
```

EmployeeStatusEnum

App\Enums\Employees

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве статусов сотрудников.

Константы

```
public const ACTIVE = 'active';
public const NO_ACTIVE = 'no_active';
```

EventHistoryEnum

App\Enums\Employees

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве тпиов действий пользователя.

Константы

```
public const EMPLOYEE_CHANGE_STATUS = 'employee_change_status';
public const EMPLOYEE_CHANGE_RATING = 'employee_change_rating';
public const EMPLOYEE_SET_ACCRUED = 'employee_set_accrued';
public const EMPLOYEE_SET_BRIGADIERS = 'employee_set_brigadiers';
public const EMPLOYEE_SET_REWARD = 'employee_set_reward';
public const EMPLOYEE_SET_PENALTY = 'employee_set_penalty';
public const EMPLOYEE_PAYOUT = 'employee_payout';
public const EMPLOYEE_SET_WORKING_SHIFT = 'employee_set_working_shift';
public const EMPLOYEE_REMOVE_WORKING_SHIFT = 'employee_remove_working_shift';
public const EMPLOYEE_CHANGE_BRIGADE_STATUS = 'employee_change_brigade_status';
public const GENERATE_EXCEL_SHIFT_HISTORY = 'generate_excel_shift_history';
public const GENERATE_EXCEL_PAYMENT_HISTORY = 'generate_excel_payment_history';
```

GenderEnum

App\Enums\Genders

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве возможных полов у пользователей.

Константы

```
public const MALE = 'male';
public const FEMALE = 'female';
```

NotificationEnum

namespace

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве возможных типов уведомлений.

Константы

```
public const ENTITY_ORDER = 'orders';
public const ENTITY_BRIGADE = 'brigades';
public const ENTITY_WORKED_SHIFT = 'worked_shifts';

public const EMPLOYEE_ALREADY_WORKED_6_SHIFTS = 'employee_already_worked_6_shifts';
public const EMPLOYEE_ALREADY_WORKED_7_SHIFTS = 'employee_already_worked_7_shifts';
public const EMPLOYEE_ABSENT_LAST_SHIFT = 'employee_absent_last_shift';
public const EMPLOYEE_BLACK_LIST = 'employee_black_list';
public const EMPLOYEE_NEVER_WORKED_CUSTOMER = 'employee_never_worked_customer';
public const ALREADY_BEEN_COLLECTED_CUSTOMER = 'already_been_collected_customer';
public const WARN_EMPLOYEE_PENALTIES = 'warn_employee_penalties';
public const IMPOSSIBLE_CHARGE_PENALTIES = 'impossible_charge_penalties';
public const PENALTY_CHARGED_PART = 'penalty_charged_part';
public const EMPLOYEE_BIRTHDAY_TODAY = 'employee_birthday_today';
public const EMPLOYEE_BIRTHDAY_TOMORROW = 'employee_birthday_tomorrow';
```

WorkedEnum

App\Enums\Salary\WorkedShifts

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве статусов смен.

Константы

```
public const NO = 'no';
public const YES = 'yes';
public const PARTLY = 'partly';
public const NO_SELECTED = 'not_selected';
```

SettingEnum

App\Enums\Settings

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве статусов смен.

Константы

```
public const SETTING_LOGO = 'logo';
public const SETTING_IMAGE_BACKGROUND = 'image_background';
public const SETTING_NAME_COMPANY = 'name_company';
```

BooleanEnum

App\Enums

> Данный класс наследует класс Enum который предоставляется библиотекой BenSampo/laravel-enum Так же имплементирует интерфейс LocalizedEnum который служит для переводов значений Переводы значений хранятся по пути /lang/ru/enums.php

Константы

Данный класс содержит в себе константы которые выступают в качестве статусов ДА и НЕТ.

Константы

```
public const YES = true;
public const NO = false;
```

title

App\Exports\Datatables\BaseDatatableExport::class

```
public function __construct(protected array $data, protected array $headings)
{
}
```

array

Данный метод возвращает данные, необходимые для генерации данных в выгрузке `return array`

```
public function array(): array
{
    return $this->data;
}
```

headings

Данный метод возвращает данные, необходимые для формирования шапки заголовков в перовой строке выгрузки `return array`

```
public function array(): array
{
    return $this->data;
}
```

bindValue

Данный метод изменяет значение ячеек в необходимый формат Функционал:

- Если данные являются числом, конвертируем его в строку для корректного отображения в ячейке
- Если дынные являются датой, форматируем под необходимый формат, и конвертируем в строку для корректного отображения `return bool`

```
public function bindValue(Cell $cell, $value): bool
{
  if (is_numeric($value)) {
      $cell->setValueExplicit($value, DataType::TYPE_STRING);
      return true;
  }
  if (DateTime::createFromFormat('Y-m-d\TH:i:s.u\Z', $value) !== false) {
      $formattedDate = Carbon::parse($value)->format('d.m.Y');
      $cell->setValueExplicit($formattedDate, DataType::TYPE_STRING);
      return true;
  }
  return parent::bindValue($cell, $value);
}
```

registerEvents

Метод для регистрации событий, конкретно в этом случае служит применения стилей к ячейкам, а именно первая строка заголовков выделяется жирным шрифтом `return bool`

```
public function registerEvents(): array
{
  return [
      AfterSheet::class => function (AfterSheet $event) {
          $event->sheet->getStyle('A1:ZZ1')->applyFromArray([
              'font' => [
                  'bold' => true
              ]
          ]);
      },
  ];
}
```

LoginController

App\Http\Controllers\Auth

> Данный класс содержит в себе следующий трейт: AuthenticatesUsers

Список методов

__construct

Данный метод регистрирует middleware по умолчанию для всех маршрутов за исключением выхода из системы

```
public function __construct()
{
    $this->middleware('guest')->except('logout');
}
```

attemptLogin

Данный метод вызывается при попытке пользователя авторизоваться в системе

Параметры:

- Request $request - данные , необходимые для аутентификации пользователя (логин/email/пароль)

В случае если пользователь ввел валидный email в строку логина, то происходит попытка аутентификации с помощью пары email/пароль

```


*Ответ*
: return bool
```

```
protected function attemptLogin(Request $request): bool
{
    /**
     * Авторизация по логину
     * Если строка не является почтой по умолчанию считаем ее логином
     */
    $dataLogin = $this->credentials($request);
    if (!filter_var($dataLogin['email'], FILTER_VALIDATE_EMAIL)) {
        $dataLogin['login'] = $dataLogin['email'];
        unset($dataLogin['email']);
    }
    $token = $this->guard()->attempt($dataLogin);

    if (! $token) {
        return false;
    }

    $user = $this->guard()->user();
    if ($user instanceof MustVerifyEmail && ! $user->hasVerifiedEmail()) {
        return false;
    }

    $this->guard()->setToken($token);

    return true;
}
```

sendLoginResponse

Данный метод является стандартным и вызывается после успешной аутентификации
Устанавливает токены доступа и очищает количество попыток входа в систему

Параметры:

- Request $request - Проверенные данные

```
return \Illuminate\Http\JsonResponse
```

```
protected function sendLoginResponse(Request $request)
{
    $this->clearLoginAttempts($request);

    $token = (string) $this->guard()->getToken();
    $expiration = $this->guard()->getPayload()->get('exp');

    return response()->json([
        'token' => $token,
        'token_type' => 'bearer',
        'expires_in' => $expiration - time(),
    ]);
}
```

sendFailedLoginResponse

Данный метод является стандартным, и вызывается при неудачной попытке аутентификации пользователя

Параметры:

- Request $request - Проверенные данные

```
return void
```

```
protected function sendFailedLoginResponse(Request $request)
{
    $user = $this->guard()->user();

    if ($user instanceof MustVerifyEmail && ! $user->hasVerifiedEmail()) {
        throw VerifyEmailException::forUser($user);
    }

    throw ValidationException::withMessages([
        $this->username() => [trans('auth.failed')],
    ]);
}
```

logout

Данный метод является стандартным, и вызывается при выходе пользователя из системы

Параметры:

- Request $request - Проверенные данные

```
return \Illuminate\Http\JsonResponse
```

```
public function logout(Request $request)
{
    $this->guard()->logout();

    return response()->json(null, 204);
}
```

UserController

App\Http\Controllers\Auth

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel

Свойства

Данное поле хранит в себе сервайс подключаемый через конструктор контроллера

```
protected Service $service;
```

Список методов

__construct

Конструктор класса определяющий сервис класс содержащий бизнес логику

Параметры:

- Service $service - класс сервиса UserService ```php

public function __construct(protected Service $service) { }

```
### current
Данный метод предоставляет данные об авторизованном пользователе
Параметры:
* Request $request - данные, необходимые для предоставления информации об авторизованном по

*Ответ*
:
```

\Illuminate\Http\JsonResponse

```
```php
public function current(Request $request): \Illuminate\Http\JsonResponse
{
    return response()->json(UserCurrentDTO::fromModel($request->user()->load([
        'roles', 'permissions'
    ])));
}
```
```

index

Данный метод обращается к текущему сервайсу, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ :

```
\Illuminate\Http\JsonResponse
```

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Данный метод обращается к текущему сервайсу, и возвращает данные для построения таблицы
Параметры:

- DatatableRequest $request - параметры DataTable

Ответ :

```
\Illuminate\Http\JsonResponse
```

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Метод отвечает за предоставление данных на страницу создания пользователя

Ответ :

```
\Illuminate\Http\JsonResponse
```

```
public function create(): JsonResponse
{
    return response()->json($this->service->getCreateData());
}
```

store

Метод отвечает за создание пользователя в системе Параметры:

- UserRequest $request - параметры DataTable

Ответ :

```
\Illuminate\Http\JsonResponse
```

```
public function store(UserRequest $request): JsonResponse
{
    return response()->json($this->service->store(UserRequestDTO::fromRequest($request)));
}
```

edit

Метод отвечает за предоставление данных на страницу редактирования пользователя

Ответ :

```
\Illuminate\Http\JsonResponse
```

```
public function edit(User $user): JsonResponse
{
    return response()->json($this->service->getEditData($user));
}
```

update

Метод отвечает за обновление данных существующего пользователя в системе Параметры:

- User $user - модель для обновления
- UserRequest $request- валидированные входящие данные

```
public function update(User $user, UserRequest $request): void
{
  $this->service->update($user, UserRequestDTO::fromRequest($request));
}
```

destroy

Метод отвечает за "мягкое" удаление пользователя из системы Параметры:

- User $user - модель для удаления

Ответ :

```
\Illuminate\Http\JsonResponse
```

```
public function destroy(User $user): JsonResponse
{
    if ($this->service->destroy($user)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

LoginController

App\Http\Controllers\Brigades

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности бригады

Список методов

__construct

Используем DI для $service Параметры:

- protected Service $service,

```
public function __construct(protected Service $service)
{
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных
- Customer $customer - Модель заказчика

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request, Customer $customer): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request), $customer)
}
```

customers

Метод для получения списка заказчиков Параметры

- BrigadeFilterRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function customers(BrigadeFilterRequest $request): JsonResponse
{
    return response()->json($this->service->customers(BrigadeFilterRequestDTO::fromRequest(
}
```

additionalDatatable

Метод формирования данных для дополнительного DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных
- Customer $customer - Модель заказчика

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function additionalDatatable(DatatableRequest $request, Customer $customer): JsonRes
{
    $this->authorize('viewAny', Brigade::class);
    return $this->service->additionalDatatable(DatatableRequestDTO::fromRequest($request),
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getDatatableFilters());
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->getCreateData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- BrigadeStoreRequest $request - данные прошедшие валидацию

```
public function store(BrigadeStoreRequest $request): void
{
    $this->authorize('create', Brigade::class);
    $this->service->store(BrigadeRequestDTO::fromRequest($request));
}
```

storeEmployee

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- BrigadeEmployeeStoreRequest $request - данные прошедшие валидацию

```
public function storeEmployee(BrigadeEmployeeStoreRequest $request): JsonResponse
{
    $this->authorize('create', Employee::class);
    return response()->json($this->service->storeEmployee(BrigadeEmployeeRequestDTO::fromRe
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- BrigadeUpdateRequest $request - данные прошедшие валидацию
- Brigade $brigade - бригада

```
public function update(BrigadeUpdateRequest $request, Brigade $brigade): void
{
    $this->authorize('update', $brigade);
    $this->service->update(BrigadeRequestDTO::fromRequest($request), $brigade);
}
```

updateStatus

Метод обновляет статус модели, вызывает аналогичный метод сервиса Параметры:

- BrigadeUpdateStatusRequest $request - данные прошедшие валидацию
- Brigade $brigade - бригада

```
public function updateStatus(BrigadeUpdateStatusRequest $request, Brigade $brigade): void
{
    $this->authorize('update', $brigade);
    $this->service->updateStatus(BrigadeStatusRequestDTO::fromRequest($request), $brigade);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Brigade $brigade - бригада

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function destroy(Brigade $brigade): JsonResponse
{
    $this->authorize('delete', $brigade);
    if ($this->service->destroy($brigade)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

WorkShiftController

App\Http\Controllers\Brigades

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности бригады

Список методов

__construct Используем DI для $service Параметры:

- protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getDatatableFilters());
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- WorkShiftRequest $request - данные прошедшие валидацию
- Order $order - бригада

```
public function update(WorkShiftRequest $request, Order $order): void
{
    $this->authorize('update', $order);
    $this->service->update(WorkShiftRequestDTO::fromRequest($request), $order);
}
```

CustomerController

App\Http\Controllers\Customers

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности заказчик

Список методов

__construct

Используем DI для $service Параметры:

- protected Service $service

```
public function __construct(protected Service $service)
{
}
### index
Данный метод обращается к текущему сервису, который возвращает данные, необходимые для рабо

*Ответ*
:
return \Illuminate\Http\JsonResponse

```php
public function index(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных
- Customer $customer - Модель заказчика

Ответ : `return \Illuminate\Http\JsonResponse`
```

```
public function datatable(DatatableRequest $request, Customer $customer): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request), $customer)
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getDatatableFilters());
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    $this->authorize('create', Customer::class);
    return response()->json($this->service->getCreateData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- CustomerStoreRequest $request - данные прошедшие валидацию

```
public function store(CustomerStoreRequest $request): void
{
    $this->authorize('create', Customer::class);
    $this->service->store(BrigadeRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Customer $customer - модель заказчика

```
public function edit(Customer $customer): JsonResponse
{
    $this->authorize('update', $customer);
    return response()->json($this->service->editData($customer));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- CustomerStoreRequest $request - данные прошедшие валидацию
- Customer $customer - модель заказчика

```
public function update(CustomerStoreRequest $request, Customer $customer): void
{
    $this->authorize('update', $customer);
    $this->service->update(CustomerRequestDTO::fromRequest($request), $customer);
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Customer $customer - модель заказчика

```php
public function destroy(Customer $customer): JsonResponse
{
    $this->authorize('delete', $customer);
    if ($this->service->destroy($customer)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

CustomerRequisiteController

App\Http\Controllers\Customers

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности бригады
>
> ## Список методов
>
> ## __construct
>
> Используем DI для $service Параметры:
>
> - protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json($this->service->getDatatableFilters());
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Brigade::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- WorkShiftRequest $request - данные прошедшие валидацию
- Order $order - бригада

```
public function update(WorkShiftRequest $request, Order $order): void
{
    $this->authorize('update', $order);
    $this->service->update(WorkShiftRequestDTO::fromRequest($request), $order);
}
```

EmployeeController

App\Http\Controllers\Dashboard\Employees

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы страницы сотрудники в дашбордах
>
> ## Список методов
>
> ### __construct
>
> Используем DI для $service Параметры:
>
> - protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля Параметры

- DashboardFilterRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(DashboardFilterRequest $request): JsonResponse
{
    return response()->json($this->service->getIndexData(DashboardFilterRequestDTO::fromReq
}
```

ManagerActivityController

App\Http\Controllers\Dashboard\ManagerActivity

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы страницы активность менеджеров в дашбордах
>
> ## Список методов
>
> ### __construct
>
> Используем DI для $service Параметры:
>
> - protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля Параметры

- DashboardFilterRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(DashboardFilterRequest $request): JsonResponse
{
    return response()->json($this->service->getIndexData(DashboardFilterRequestDTO::fromReq
}
```

OrderController

App\Http\Controllers\Dashboard\Orders

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы страницы заявки в дашбордах
>
> ## Список методов
>
> ### __construct
>
> Используем DI для $service Параметры:
>
> - protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля Параметры

- DashboardFilterRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(DashboardFilterRequest $request): JsonResponse
{
    return response()->json($this->service->getIndexData(DashboardFilterRequestDTO::fromReq
}
```

PaymentController

App\Http\Controllers\Dashboard\Payments

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы страницы оплата в дашбордах
>
> ## Список методов
>
> ## __construct
>
> Используем DI для $service Параметры:
>
> - protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля Параметры

- DashboardFilterRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(DashboardFilterRequest $request): JsonResponse
{
    return response()->json($this->service->getIndexData(DashboardFilterRequestDTO::fromReq
}
```

SeasonalActivityController

App\Http\Controllers\Dashboard\SeasonalActivity

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы страницы сезонная активность в дашбордах
>
> ## Список методов
>
> ### __construct
>
> Используем DI для $service Параметры:
>
> - protected Service $service

```
public function __construct(protected Service $service)
{
}
```

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля Параметры

- DashboardFilterRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(DashboardFilterRequest $request): JsonResponse
{
    return response()->json($this->service->getIndexData(DashboardFilterRequestDTO::fromReq
}
```

LoginController

App\Http\Controllers\Auth

> Данный класс содержит в себе следующий трейт: AuthenticatesUsers

Список методов

EmployeeController

App\Http\Controllers\Employees

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности сотрудник
>
> ## Методы
>
> ## index
>
> Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Employee::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Employee::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Employee::class);
    return response()->json($this->service->getDatatableFilters());
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    $this->authorize('create', Employee::class);
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeStoreRequest $request - данные прошедшие валидацию

```
public function store(EmployeeStoreRequest $request): void
{
    $this->authorize('create', Employee::class);
    $this->service->store(EmployeeRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Employee $employee - модель сотрудника

```
public function edit(Employee $employee): JsonResponse
{
    $this->authorize('update', $employee);
    return response()->json($this->service->editData($employee));
}
```

show

Данный метод обращается к основному сервису и возвращает данные, необходимые для отображения модели в системе Параметры:

- Employee $employee - модель сотрудника

```
public function show(Employee $employee): JsonResponse
{
    $this->authorize('viewAny', $employee);
    return response()->json($this->service->show($employee));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeStoreRequest $request - данные прошедшие валидацию
- Employee $employee - модель сотрудника

```
public function update(EmployeeStoreRequest $request, Employee $employee): void
{
    $this->authorize('update', $employee);
    $this->service->update(EmployeeRequestDTO::fromRequest($request), $employee);
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Employee::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Document $document - модель сотрудника

```
public function destroy(Employee $employee): JsonResponse
{
    $this->authorize('delete', $employee);
    if ($this->service->destroy($employee)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

generateExcel

Метод генерации excel документа из данных datatable, вызывающий аналогичный метод сервиса Параметры:

- DataTableRequest $request - Класс валидатора для проверки данных

Ответ : `return Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function generateExcel(DataTableRequest $request): BinaryFileResponse
{
    $this->authorize('viewAny', Employee::class);
    return $this->service->generateExcel(DatatableRequestDTO::fromRequest($request));
}
```

search

Метод для поиска записей по названию Параметры:

- BaseSearchRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function search(BaseSearchRequest $request): JsonResponse
{
    return response()->json($this->service->search(BaseSearchRequestDTO::fromRequest($reque
}
```

LoginController

App\Http\Controllers\GeneralReports

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности общий отчёт

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Customer::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Customer::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса Параметры

- DatatableColumnStoreRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Customer::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

generateExcel

Метод генерации excel документа из данных datatable, вызывающий аналогичный метод сервиса
Параметры:

- DataTableRequest $request - Класс валидатора для проверки данных

Ответ : `return Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function generateExcel(DataTableRequest $request): BinaryFileResponse
{
    $this->authorize('viewAny', Customer::class);
    return $this->service->generateExcel(DatatableRequestDTO::fromRequest($request));
}
```

IncomingController

App\Http\Controllers\Incomings

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности поступления

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Incoming::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Incoming::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Incoming::class);
    return response()->json($this->service->getDatatableFilters());
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    $this->authorize('create', Incoming::class);
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- IncomingStoreRequest $request - данные прошедшие валидацию

```
public function store(IncomingStoreRequest $request): void
{
    $this->authorize('create', Incoming::class);
    $this->service->store(IncomingRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Incoming $incoming - модель поступления

```
public function edit(Incoming $incoming): JsonResponse
{
    $this->authorize('update', $incoming);
    return response()->json($this->service->editData($incoming));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- IncomingStoreRequest $request - данные прошедшие валидацию
- Incoming $incoming - модель поступления

```
public function update(IncomingStoreRequest $request, Incoming $incoming): void
{
    $this->authorize('update', $incoming);
    $this->service->update(IncomingRequestDTO::fromRequest($request), $incoming);
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса Параметры:

- DatatableColumnStoreRequest $request - данные прошедшие валидацию

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Incoming::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Incoming $incoming - модель поступления

```
public function destroy(Incoming $incoming): JsonResponse
{
    $this->authorize('delete', $incoming);
    if ($this->service->destroy($incoming)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

generateExcel

Метод генерации excel документа из данных datatable, вызывающий аналогичный метод сервиса
Параметры:

- DataTableRequest $request - Класс валидатора для проверки данных

Ответ : `return Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function generateExcel(DataTableRequest $request): BinaryFileResponse
{
    $this->authorize('viewAny', Incoming::class);
    return $this->service->generateExcel(DatatableRequestDTO::fromRequest($request));
}
```

InvoiceController

App\Http\Controllers\Invoices

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности счёта

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Invoice::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Invoice::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Invoice::class);
    return response()->json($this->service->getDatatableFilters());
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    $this->authorize('create', Invoice::class);
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- IncomingStoreRequest $request - данные прошедшие валидацию

```
public function store(IncomingStoreRequest $request): void
{
    $this->authorize('create', Invoice::class);
    $this->service->store(IncomingRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Invoice $invoice- модель счёта

```
public function edit(Invoice $invoice): JsonResponse
{
    $this->authorize('update', $invoice);
    return response()->json($this->service->editData($invoice));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- InvoiceStoreRequest $request - данные прошедшие валидацию
- Invoice $invoice- модель счёта

```
public function update(IncomingStoreRequest $request, Invoice $invoice): void
{
    $this->authorize('update', $invoice);
    $this->service->update(InvoiceRequestDTO::fromRequest($request), $invoice);
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса Параметры:

- DatatableColumnStoreRequest $request - данные прошедшие валидацию

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Invoice::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Invoice $invoice - модель счёта

```
public function destroy(Invoice $invoice): JsonResponse
{
    $this->authorize('delete', $incoming);
    if ($this->service->destroy($incoming)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

generateExcel

Метод генерации excel документа из данных datatable, вызывающий аналогичный метод сервиса
Параметры:

- DataTableRequest $request - Класс валидатора для проверки данных

Ответ : `return Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function generateExcel(DataTableRequest $request): BinaryFileResponse
{
    $this->authorize('viewAny', Invoice::class);
    return $this->service->generateExcel(DatatableRequestDTO::fromRequest($request));
}
```

LegalEntityController

App\Http\Controllers\LegalEntities

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности юридического лица

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', LegalEntity::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', LegalEntity::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    $this->authorize('create', LegalEntity::class);
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- LegalEntityStoreRequest $request - данные прошедшие валидацию

```
public function store(LegalEntityStoreRequest $request): void
{
    $this->authorize('create', LegalEntity::class);
    $this->service->store(LegalEntityRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- LegalEntity $legalEntity - модель юридического лица

```
public function edit(LegalEntity $legalEntity): JsonResponse
{
    $this->authorize('update', $legalEntity);
    return response()->json($this->service->editData($legalEntity));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- LegalEntityStoreRequest $request - данные прошедшие валидацию
- LegalEntity $legalEntity - модель юридического лица

```
public function update(LegalEntityStoreRequest $request, LegalEntity $legalEntity): void
{
    $this->authorize('update', $legalEntity);
    $this->service->update(LegalEntityRequestDTO::fromRequest($request), $legalEntity);
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса Параметры:

- DatatableColumnStoreRequest $request - данные прошедшие валидацию

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', LegalEntity::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

search

Метод поиска юридического лица по его названию Параметры:

- BaseSearchRequest $request - данные прошедшие валидацию

```
public function search(BaseSearchRequest $request): JsonResponse
{
    return response()->json($this->service->search(BaseSearchRequestDTO::fromRequest($reque
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- LegalEntity $legalEntity - модель юридического лица

```
public function destroy(LegalEntity $legalEntity): JsonResponse
{
    $this->authorize('delete', $legalEntity);
    if ($this->service->destroy($legalEntity)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

LoginController

App\Http\Controllers\Main

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы главной страницы

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(MainFilterRequest $request): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return response()->json($this->service->getIndexData(MainFilterRequestDTO::fromRequest(
}
```

getDatatableFilters Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return response()->json($this->service->getDatatableFilters());
}
```

LoginController

App\Http\Controllers\Notifications

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности уведомления

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

OrderController

App\Http\Controllers\Auth

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности заявки

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод получения данных фильтров для DataTable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return response()->json($this->service->getDatatableFilters());
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    $this->authorize('create', Order::class);
    return response()->json($this->service->createData());
}
### store
Метод создаёт новую модель, вызывает аналогичный метод сервиса
Параметры:
* OrderStoreRequest $request - данные прошедшие валидацию

```php
public function store(OrderStoreRequest $request): void
{
    $this->authorize('create', Order::class);
    $this->service->store(OrderRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Order $order - модель заказа

```
public function edit(Order $order): JsonResponse
{
    $this->authorize('update', $order);
    return response()->json($this->service->editData($order));
}
```
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- OrderUpdateRequest $request - данные прошедшие валидацию
- Order $order - модель заказа

```
public function update(OrderUpdateRequest $request, Order $order): void
{
    $this->authorize('update', $order);
    $this->service->update(OrderUpdateRequestDTO::fromRequest($request), $order);
}
```

storeTableColumn

Данный метод сохраняет колонки datatable, вызывает аналогичный метод сервиса Параметры:

- DatatableColumnStoreRequest $request - данные прошедшие валидацию

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function storeTableColumn(DatatableColumnStoreRequest $request): JsonResponse
{
    $this->authorize('viewAny', Incoming::class);
    return response()->json(
        $this->service->datatableStoreColumn(DatatableColumnStoreRequestDTO::fromRequest($r
    );
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Order $order - модель заказа

```
public function destroy(Order $order): JsonResponse
{
    $this->authorize('delete', $order);
    if ($this->service->destroy($order)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

OrderEmployeeController

App\Http\Controllers\Orders

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сотрудников в заявках

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

search

Метод поиска записей по названию Параметры

- BaseSearchRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function search(BaseSearchRequest $request): JsonResponse
{
    return response()->json($this->service->search(BaseSearchRequestDTO::fromRequest($reque
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- OrderStoreRequest $request - данные прошедшие валидацию

```
public function store(OrderStoreRequest $request): void
{
    $this->authorize('update', Order::class);
    $this->service->store(OrderRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Order $order - модель заказа

```
public function edit(Order $order): JsonResponse
{
    $this->authorize('update', $order);
    return response()->json($this->service->editData($order));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- OrderEmployeeUpdateRequest $request - данные прошедшие валидацию
- Order $order - модель заказа

```
public function update(OrderEmployeeUpdateRequest $request, Order $order): void
{
    $this->authorize('update', $order);
    $this->service->update(OrderEmployeeUpdateRequestDTO::fromRequest($request), $order);
}
```

OrderRecommendationEmployeeController

App\Http\Controllers\Orders

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы с рекомендациями сотрудников в заявках

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод формирования данных для фильтров datatable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    $this->authorize('viewAny', Order::class);
    return response()->json($this->service->getDatatableFilters());
}
```

BillingPeriodController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника период оплаты

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- ReferenceStoreRequest $request - данные прошедшие валидацию

```
public function store(ReferenceStoreRequest $request): void
{
    $this->service->store(ReferenceRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- BillingPeriod $billingPeriod - модель периода оплаты

```
public function edit(BillingPeriod $billingPeriod): JsonResponse
{
    return response()->json($this->service->editData($billingPeriod));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- ReferenceStoreRequest $request - данные прошедшие валидацию
- BillingPeriod $billingPeriod - модель периода оплаты

```
public function update(ReferenceStoreRequest $request, BillingPeriod $billingPeriod): void
{
    $this->service->update(ReferenceRequestDTO::fromRequest($request), $billingPeriod);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- BillingPeriod $billingPeriod - модель счёта

```
public function destroy(BillingPeriod $billingPeriod): JsonResponse
{
    if ($this->service->destroy($billingPeriod)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

EmployeeBonusController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника бонусы сотрудникам

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeBonusStoreRequest $request - данные прошедшие валидацию

```
public function store(EmployeeBonusStoreRequest $request): void
{
    $this->service->store(EmployeeBonusRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- EmployeeBonus $employeeBonus - модель бонусы сотрудникам

```
public function edit(EmployeeBonus $employeeBonus): JsonResponse
{
    return response()->json($this->service->editData($employeeBonus));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeBonusStoreRequest $request - данные прошедшие валидацию
- EmployeeBonus $employeeBonus - модель бонсы сотрудникам

```
public function update(EmployeeBonusStoreRequest $request, EmployeeBonus $employeeBonus): v
{
    $this->service->update(EmployeeBonusRequestDTO::fromRequest($request), $employeeBonus);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeBonus $employeeBonus - модель бонусов сотрудникам

```
public function destroy(EmployeeBonus $employeeBonus): JsonResponse
{
    if ($this->service->destroy($employeeBonus)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

EmployeeRatingController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника рейтинг сотрудников

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeRatingStoreRequest $request - данные прошедшие валидацию

```
public function store(EmployeeRatingStoreRequest $request): void
{
    $this->service->store(EmployeeRatingRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- EmployeeRating $employeeRating - модель рейтинг сотрудников

```
public function edit(EmployeeRating $employeeRating): JsonResponse
{
    return response()->json($this->service->editData($employeeRating));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeRatingStoreRequest $request - данные прошедшие валидацию
- EmployeeRating $employeeBonus - модель рейтинг сотрудникам

```
public function update(EmployeeRatingStoreRequest $request, EmployeeRating $employeeRating)
{
    $this->service->update(EmployeeRatingRequestDTO::fromRequest($request), $employeeRating
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeRating $employeeRating - модель бонусов сотрудникам

```
public function destroy(EmployeeRating $employeeRating): JsonResponse
{
    if ($this->service->destroy($employeeRating)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

EmployeeStatusController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника статусы сотрудников

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeStatusStoreRequest $request - данные прошедшие валидацию

```
public function store(EmployeeStatusStoreRequest $request): void
{
    $this->service->store(EmployeeStatusRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- EmployeeStatus $employeeStatus - модель статус сотрудников

```
public function edit(EmployeeStatus $employeeStatus): JsonResponse
{
    return response()->json($this->service->editData($employeeStatus));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeStatusStoreRequest $request - данные прошедшие валидацию
- EmployeeStatus $employeeStatus - модель статус сотрудникам

```
public function update(EmployeeStatusStoreRequest $request, EmployeeStatus $employeeStatus)
{
    $this->service->update(EmployeeStatusRequestDTO::fromRequest($request), $employeeStatus
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- EmployeeStatus $employeeStatus - модель статусы сотрудников

```
public function destroy(EmployeeStatus $employeeStatus): JsonResponse
{
    if ($this->service->destroy($employeeStatus)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

JobResourceController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника откуда узнали о работе

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- ReferenceStoreRequest $request - данные прошедшие валидацию

```
public function store(ReferenceStoreRequest $request): void
{
    $this->service->store(ReferenceRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- JobResource $jobResource - модель откуда узнал о работе

```
public function edit(JobResource $jobResource): JsonResponse
{
    return response()->json($this->service->editData($jobResource));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- ReferenceStoreRequest $request - данные прошедшие валидацию
- JobResource $jobResource - модель откуда узнал о работе

```
public function update(ReferenceStoreRequest $request, JobResource $jobResource): void
{
    $this->service->update(ReferenceRequestDTO::fromRequest($request), $jobResource);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- JobResource $jobResource - модель откуда узнал о работе

```
public function destroy(JobResource $jobResource): JsonResponse
{
    if ($this->service->destroy($jobResource)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

PenaltyController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника штрафы
>
> ## index
>
> Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- ReferenceStoreRequest $request - данные прошедшие валидацию

```
public function store(PenaltyStoreRequest $request): void
{
    $this->service->store(PenaltyRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- Penalty $penalty - модель откуда узнал о работе

```
public function edit(Penalty $penalty): JsonResponse
{
    return response()->json($this->service->editData($penalty));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- PenaltyStoreRequest $request - данные прошедшие валидацию
- Penalty $penalty - модель штрафы

```
public function update(PenaltyStoreRequest $request, Penalty $penalty): void
{
    $this->service->update(PenaltyRequestDTO::fromRequest($request), $penalty);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Penalty $penalty - модель штрафы

```
public function destroy(Penalty $penalty): JsonResponse
{
    if ($this->service->destroy($penalty)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

PercentageBonusController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника процент премии

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- PercentageBonusStoreRequest $request - данные прошедшие валидацию

```
public function store(PercentageBonusStoreRequest $request): void
{
    $this->service->store(PercentageBonusRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- PercentageBonus $percentageBonus - модель процент премии

```
public function edit(PercentageBonus $percentageBonus): JsonResponse
{
    return response()->json($this->service->editData($percentageBonus));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- PenaltyStoreRequest $request - данные прошедшие валидацию
- PercentageBonus $percentageBonus - модель процент премии

```
public function update(PercentageBonusStoreRequest $request, PercentageBonus $percentageBon
{
    $this->service->update(PercentageBonusRequestDTO::fromRequest($request), $percentageBon
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- PercentageBonus $percentageBonus - модель процент премии

```
public function destroy(PercentageBonus $percentageBonus): JsonResponse
{
    if ($this->service->destroy($percentageBonus)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

SubdivisionController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника подразделения

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- SubdivisionStoreRequest $request - данные прошедшие валидацию

```
public function store(SubdivisionStoreRequest $request): void
{
    $this->service->store(SubdivisionRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры

- Subdivision $subdivision - модель подразделения

```
public function edit(Subdivision $subdivision): JsonResponse
{
    return response()->json($this->service->editData($subdivision));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- SubdivisionStoreRequest $request - данные прошедшие валидацию
- Subdivision $subdivision - модель подразделения

```
public function update(SubdivisionStoreRequest $request, Subdivision $subdivision): void
{
    $this->service->update(SubdivisionRequestDTO::fromRequest($request), $subdivision);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- Subdivision $subdivision - модель подразделения

```
public function destroy(Subdivision $subdivision): JsonResponse
{
    if ($this->service->destroy($subdivision)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

WorkingHourController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника количество часов

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- WorkingHourStoreRequest $request - данные прошедшие валидацию

```
public function store(WorkingHourStoreRequest $request): void
{
    $this->service->store(WorkingHourRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры

- WorkingHour $workingHour - модель количество часов

```
public function edit(WorkingHour $workingHour): JsonResponse
{
    return response()->json($this->service->editData($workingHour));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- WorkingHourStoreRequest $request - данные прошедшие валидацию
- WorkingHour $workingHour - модель количество часов

```
public function update(WorkingHourStoreRequest $request, WorkingHour $workingHour): void
{
    $this->service->update(WorkingHourRequestDTO::fromRequest($request), $workingHour);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- WorkingHour $workingHour - модель количество часов

```
public function destroy(WorkingHour $workingHour): JsonResponse
{
    if ($this->service->destroy($workingHour)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

WorkingShiftController

App\Http\Controllers\References

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы справочника смен

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- WorkingHourStoreRequest $request - данные прошедшие валидацию

```
public function store(WorkingHourStoreRequest $request): void
{
    $this->service->store(WorkingHourRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры

- WorkingHour $workingHour - модель количество часов

```
public function edit(WorkingHour $workingHour): JsonResponse
{
    return response()->json($this->service->editData($workingHour));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- WorkingHourStoreRequest $request - данные прошедшие валидацию
- WorkingHour $workingHour - модель количество часов

```
public function update(WorkingHourStoreRequest $request, WorkingHour $workingHour): void
{
    $this->service->update(WorkingHourRequestDTO::fromRequest($request), $workingHour);
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- WorkingHour $workingHour - модель количество часов

```
public function destroy(WorkingHour $workingHour): JsonResponse
{
    if ($this->service->destroy($workingHour)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

SettingTableController

App\Http\Controllers\Settings

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы сущности настройки

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

ActionHistoryController

App\Http\Controllers\Users

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы модуля история действий

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

getDatatableFilters

Метод предоставляет данные для фильтров datatable

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function getDatatableFilters(): JsonResponse
{
    return response()->json($this->service->getDatatableFilters());
}
```

RoleController

App\Http\Controllers\Users

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы модуля роли

index

Данный метод обращается к текущему сервису, который возвращает данные, необходимые для работы стартовой страницы модуля

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function index(): JsonResponse
{
    return response()->json($this->service->getIndexData());
}
```

datatable

Метод формирования данных для DataTable Параметры

- DatatableRequest $request - Класс валидатора для проверки данных

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function datatable(DatatableRequest $request): JsonResponse
{
    return $this->service->datatable(DatatableRequestDTO::fromRequest($request));
}
```

create

Данный метод обращается к основному сервису и возвращает данные, необходимые для создания модели в системе

Ответ : `return \Illuminate\Http\JsonResponse`

```
public function create(): JsonResponse
{
    return response()->json($this->service->createData());
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- RoleRequest $request - данные прошедшие валидацию

```
public function store(RoleRequest $request): JsonResponse
{
    return $this->service->store($request->validated());
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- RoleUser $roleUser - модель роли

```
public function edit(RoleUser $roleUser): JsonResponse
{
    return response()->json($this->service->getEditData($roleUser));
}
```

update

Метод обновляет модель, вызывает аналогичный метод сервиса Параметры:

- RoleRequest $request - данные прошедшие валидацию
- RoleUser $roleUser - модель роли

```
public function update(RoleRequest $request, RoleUser $roleUser): JsonResponse
{
    return response()->json($this->service->update($roleUser, $request->validated()));
}
```

destroy

Метод удаляет модель, вызывает аналогичный метод сервиса Параметры:

- RoleUser $roleUser - модель роли

```
public function destroy(RoleUser $roleUser): JsonResponse
{
    if ($this->service->destroy($roleUser)) {
        return response()->json([
            'success' => true
        ]);
    }
    return response()->json(['message' => __('message.destroy_element')], 422);
}
```

WorkScheduleController

App\Http\Controllers\WorkSchedules

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фреймворком Laravel Данный контроллер отвечает за логику работы модуля график работы

formData

Метод отдаёт данные для формы, вызывает аналогичный метод сервиса Параметры:

- WorkScheduleFormDataRequest $request - данные прошедшие валидацию

```
public function formData(WorkScheduleFormDataRequest $request): JsonResponse
{
    $this->authorize('create', WorkSchedule::class);
    return response()->json($this->service->formData(WorkScheduleFormDataRequestDTO::fromRe
}
```

store

Метод создаёт новую модель, вызывает аналогичный метод сервиса Параметры:

- WorkScheduleStoreRequest $request - данные прошедшие валидацию

```
public function store(WorkScheduleStoreRequest $request): void
{
    $this->authorize('create', WorkSchedule::class);
    $this->service->store(WorkScheduleStoreRequestDTO::fromRequest($request));
}
```

edit

Данный метод обращается к основному сервису и возвращает данные, необходимые для редактирования модели в системе Параметры:

- WorkSchedule $workSchedule - модель периода оплаты

```
public function edit(WorkSchedule $workSchedule): JsonResponse
{
    $this->authorize('update', $workSchedule);
    return response()->json($this->service->editData($workSchedule));
}
```

DownloadFileController

App\Http\Controllers

> Данный контроллер наследует базовый функционал стандартного контроллера Controller, поставляемого с фрейворком Laravel Данный контроллер отвечает за загрузку файлов из системы

Список методов

privateFile

Данный метод позволяет загрузить закрытый от просмотра из вне файл, по его uuid Параметры:

- string $uuid - UUID файла

Ответ : `return \Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function privateFile(string $uuid): \Symfony\Component\HttpFoundation\BinaryFileResp
{
    $file = Media::whereUuid($uuid)->firstOrFail();
    return response()->file($file->getPath(), [
        'Content-Disposition' =>  'attachment; filename="'.$file->file_name.'"'
    ]);
}
```

privatePreview

Данный метод позволяет посмотреть закрытый от просмотра из вне файл, по его uuid Параметры:

- string $uuid - UUID файла

Ответ : `return \Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function privatePreview(string $uuid): BinaryFileResponse
{
    $file = Media::whereUuid($uuid)->firstOrFail();
    return response()->file(
        $file->getPath('preview'),
        ['Content-Disposition' =>  'attachment; filename="' . $file->file_name . '"']
    );
}
```

publicFile

Данный метод позволяет загрузить открытый для загрузки файл, по его uuid Параметры:

- string $uuid - UUID файла

Ответ : `return \Symfony\Component\HttpFoundation\BinaryFileResponse`

```
public function publicFile(string $uuid): \Symfony\Component\HttpFoundation\BinaryFileRespo
{
    $file = Media::where('collection_name', 'public')->whereUuid($uuid)->firstOrFail();
    return response()->file($file->getPath(), [
        'Content-Disposition' =>  'attachment; filename="'.$file->file_name.'"'
    ]);
}
```

BrigadeEmployeeStoreRequest

App\Http\Requests\Brigades

> Данный класс отвечает за валидацию входящих данных, необходимых для создания бригады

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'employee_id' => 'bail|required|numeric|integer|exists:employees,id,deleted_at,NULL
        'subdivision_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:subdivisions,id,deleted_at,NULL'
        ]
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

BrigadeStoreRequest

App\Http\Requests\Brigades

> Данный класс отвечает за валидацию входящих данных, необходимых для добавления сотрудника в бригады

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'working_shifts' => 'bail|nullable|array',
        'working_shifts.*' => 'bail|nullable|numeric|integer|exists:working_shifts,id,delet
        'full_name' => 'bail|required|string|max:255',
        'birthdate' => 'bail|required|date',
        'phones' => 'bail|required|array',
        'phones.*' => ['bail', 'required', 'string', new PhoneNumberRule()],
        'passport_serial_number' => 'bail|nullable|string|max:11',
        'issued' => 'bail|nullable|string|max:255',
        'issue_date' => 'bail|nullable|date',
        'registration_address' => 'bail|nullable|string|max:255',
        'comment' => 'bail|nullable|string|max:255'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.employees');
}
```

BrigadeUpdateRequest

App\Http\Requests\Brigades

> Данный класс отвечает за валидацию входящих данных, необходимых для обновления бригады

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'subdivision_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:subdivisions,id,deleted_at,NULL'
        ]
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

BrigadeUpdateStatusRequest

App\Http\Requests\Brigades

> Данный класс отвечает за валидацию входящих данных, необходимых для обновления статуса бригады

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'brigade_status_id' => 'bail|required|numeric|integer|exists:brigade_statuses,id,de
        'days' => 'bail|required_if:brigade_status_id,3|numeric|integer|min:1|max:255',
        'day_off_start_date' => 'bail|nullable|date',
        'day_off_end_date' => 'bail|nullable|date'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.brigades');
}
```

WorkShiftRequest

App\Http\Requests\Brigades

> Данный класс отвечает за валидацию входящих данных, необходимых для добавления пользователей в бригады

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'employee_id' => 'bail|required|numeric|integer|exists:employees,id,deleted_at,NULL
        'subdivision_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:subdivisions,id,deleted_at,NULL'
            // Rule::unique('orders')->where(function ($query) {
            //     return $query->where('customer_id', $this->input('customer_id'))
            //         ->where('working_shift_date', $this->input('working_shift_date'))
            //         ->where('working_hour_id', $this->input('order_items.*.working_hour_
            //         ->whereNull('deleted_at');
            // })
        ]
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

CustomerRequisiteStoreRequest

App\Http\Requests\Customers

> Данный класс отвечает за валидацию входящих данных, необходимых для добавления реквезитов заказчика

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'name' => 'bail|required|string|max:255',
        'inn' => [
            'bail',
            'nullable',
            'numeric',
            'integer',
            new InnValidationRule()
        ],
        'kpp' => 'bail|nullable|numeric|integer|digits:9',
        'orgn' => 'bail|nullable|numeric|integer|digits:13',
        'legal_address' => 'bail|nullable|string|max:255',
        'phone' => [
            'bail', 'nullable', 'string', 'max:255', new PhoneNumberRule()
        ],
        'email' => 'bail|nullable|string|email|max:255',
        'director_name' => 'bail|nullable|string|max:255',
        'director_position' => 'bail|nullable|string|max:255',
        'foundation' => 'bail|nullable|string|max:255',
        'contacts' => 'bail|nullable|array',

        'bank_requisites' => 'bail|nullable|array',
        'bank_requisites.*.payment_account' => 'bail|nullable|string|digits:20',
        'bank_requisites.*.bank_name' => 'bail|nullable|string|max:255',
        'bank_requisites.*.correspondent_account' => 'bail|nullable|string|digits:20',
        'bank_requisites.*.bik' => 'bail|nullable|string|digits:9',
        'bank_requisites.*.bank_address' => 'bail|nullable|string|max:255',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.customer_requisites');
}
```

CustomerStoreRequest

App\Http\Requests\Customers

> Данный класс отвечает за валидацию входящих данных, необходимых для создания заказчика

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:50',
        'actual_location' => 'bail|required|string|max:50',
        'working_hours_count' => 'bail|nullable|numeric|integer|digits_between:1,15',
        'employees_involved_count' => 'bail|nullable|numeric|integer|digits_between:1,15',
        'current_contract_number' => 'bail|required|string|max:50',
        'contract_start_date' => 'bail|nullable|date',
        'contract_end_date' => 'bail|nullable|string|max:255',
        'current_additional_agreement_number' => 'bail|nullable|string|max:50',
        'billing_period_id' => 'bail|required|numeric|integer|exists:billing_periods,id,del
        'rate_customers' => 'bail|required|array',
        'rate_customers.*.working_hour_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:working_hours,id,deleted_at,NULL'
        ],
        'rate_customers.*.sum' => 'bail|required|numeric|integer|digits_between:1,15',
        'rate_employees' => 'bail|required|array',
        'rate_employees.*.working_hour_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:working_hours,id,deleted_at,NULL'
        ],
        'rate_employees.*.sum' => 'bail|required|numeric|integer|digits_between:1,15',
        'cost_transport_there_customer' => 'bail|required|numeric|integer|digits_between:1,
        'cost_transport_back_customer' => 'bail|required|numeric|integer|digits_between:1,1
        'cost_transport_round_trip_customer' => 'bail|required|numeric|integer|digits_betwe
        'cost_transport_there_tk' => 'bail|required|numeric|integer|digits_between:1,15',
        'cost_transport_back_tk' => 'bail|required|numeric|integer|digits_between:1,15',
        'cost_transport_round_trip_tk' => 'bail|required|numeric|integer|digits_between:1,1
        'brigadiers_x1_rub' => 'bail|required|numeric|integer|digits_between:1,15',
        'brigadiers_x2_rub' => 'bail|required|numeric|integer|digits_between:1,15',
        'brigadiers_personal_rub' => 'bail|required|numeric|integer|digits_between:1,15',
        'legal_entity_id' => 'bail|required|numeric|integer|exists:legal_entities,id,delete
        'active' => 'bail|required|string|max:255',
        'comment' => 'bail|nullable|string|max:256',
```

```
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.customers');
}
```

DashboardFilterRequest

App\Http\Requests\Dashboard

> Данный класс отвечает за валидацию входящих данных, необходимых для фильтрации данных в дашбордах

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'period' => 'bail|nullable|array|max:2',
        'period.*' => 'bail|nullable|date',
        'customer' => 'bail|nullable|array',
        'customer.id' => 'bail|nullable|exists:customers,id,deleted_at,NULL',
        'user' => 'bail|nullable|array',
        'user.id' => 'bail|nullable|exists:users,id,deleted_at,NULL',
        'data_type' => 'bail|nullable|string|in:payment,employee,order',
        'type' => 'bail|nullable|string|in:month,year',
        'legal_entity' => 'bail|nullable|array',
        'legal_entity.id' => 'bail|nullable|exists:legal_entities,id,deleted_at,NULL'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

DatatableColumnStoreRequest

App\Http\Requests\Datatables

> Данный класс отвечает за валидацию входящих данных, необходимых для сохранения столбцов таблицы

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'columns' => 'required|array|min:1',
    ];
}
```

BrigadeEmployeeStoreRequest

App\Http\Requests\Datatables

> Данный класс отвечает за валидацию входящих данных, необходимых для фильтрации данных в таблице

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'datatable'             => 'nullable|boolean',
        'length'                => 'nullable|integer',
        'start'                 => 'nullable|integer',
        'draw'                  => 'nullable|integer',
        'columns'               => 'nullable|array',
        'order'                 => 'nullable|array',
        'search'                => 'nullable|array',
        'search.value'          => 'nullable|string|max:150',
        'period'                => 'bail|nullable|array|max:2',
        'status'                => 'bail|nullable|boolean',
        'role'                  => 'bail|nullable|exists:roles,name,deleted_at,NULL',
        'user'                  => 'bail|nullable|array',
        'user.id'               => 'bail|nullable|exists:users,id,deleted_at,NULL',
        'customer'              => 'bail|nullable|array',
        'customer.id'           => 'bail|nullable|exists:customers,id,deleted_at,NULL',
        'section'               => 'bail|nullable|string|max:255',
        'working_shift_date'    => 'bail|nullable|date',
        'gender' => [
            'bail',
            'nullable',
            'string',
            Rule::in(GenderEnum::getValues())
        ],
        'rating'                => 'bail|nullable|numeric|integer|exists:employee_ratings,i
        'working_shift'         => 'bail|nullable|numeric|integer|exists:working_shifts,id,
        'employee_status'       => 'bail|nullable|numeric|integer|exists:employee_statuses,
        'documentable_id'       => 'bail|nullable|numeric|integer',
        'documentable_type'     => 'bail|nullable|string|max:255',
        'month'                 => 'bail|nullable|date',
        'order_id'              => 'bail|nullable|numeric|integer|exists:orders,id,deleted_
        'employee_id'           => 'bail|nullable|numeric|integer|exists:employees,id,delet
        'employee'              => 'bail|nullable|numeric|integer|exists:employees,id,delet
        'employee_rating'       => 'bail|nullable|numeric|integer|exists:employee_ratings,i
        'last_object'           => 'bail|nullable|numeric|integer|exists:customers,id,delet
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.datatables');
}
```

DocumentStoreRequest

App\Http\Requests\Datatables

> Данный класс отвечает за валидацию входящих данных, необходимых для сохранения документов

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255',
        'file' => [
            'bail',
            'required',
            'file',
            'max:102400',
            'mimes:txt,doc,docx,pdf,xls,xlsx,zip,rar,jpeg,jpg,png,mp3,wav,mp4,avi,bmp,gif,t
        ],
        'comment' => 'bail|nullable|string|max:256',
        'documentable_id' => 'bail|nullable|numeric|integer',
        'documentable_type' => 'bail|nullable|string|max:255',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.documents');
}
```

DocumentUpdateRequest

App\Http\Requests\Datatables

> Данный класс отвечает за валидацию входящих данных, необходимых для фильтрации данных в таблице

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    $rules = parent::rules();
    $rules['file'] = [
        'bail',
        'nullable',
        'file',
        'max:102400',
        'mimes:txt,doc,docx,pdf,xls,xlsx,zip,rar,jpeg,jpg,png,mp3,wav,mp4,avi,bmp,gif,tif'
    ];
    return $rules;
}
```

EmployeeStoreRequest

App\Http\Requests\Employees

> Данный класс отвечает за валидацию входящих данных, необходимых для создания сотрудника

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'avatar' => 'bail|nullable|image|max:10240',
        'delete_avatar' => 'bail|nullable|boolean',
        'card_created_at' => 'bail|required|date',
        'user_id' => 'bail|required|numeric|integer|exists:users,id,deleted_at,NULL',
        'working_shifts' => 'bail|required|array',
        'working_shifts.*' => 'bail|nullable|numeric|integer|exists:working_shifts,id,delet
        'full_name' => 'bail|required|string|max:255',
        'birthdate' => 'bail|required|date',
        'phones' => 'bail|required|array',
        'phones.*' => ['bail', 'required', 'string', new PhoneNumberRule()],
        'additional_phones' => 'bail|nullable|array',
        'additional_phones.*' => ['bail', 'required', 'string', 'max:255', new PhoneNumberR
        'additional_phones.0' => ['bail', 'nullable', 'string', 'max:255', new PhoneNumberR
        'gender' => [
            'bail',
            'required',
            'string',
            Rule::in(GenderEnum::getValues())
        ],
        'passport_serial_number' => 'bail|required|string|max:11',
        'issued' => 'bail|required|string|max:255',
        'issue_date' => 'bail|required|date',
        'registration_address' => 'bail|required|string|max:255',
        'fact_address' => 'bail|required|string|max:255',
        'job_resource_id' => 'bail|nullable|numeric|integer|exists:job_resources,id,deleted
        'whence' => [
            'bail',
            'nullable',
            sprintf('required_if:job_resource_id,%d', JobResource::where('alias', 'other')-
            'string',
            'max:255'
        ],
        'skills' => 'bail|nullable|string|max:255',
        'health_restrictions' => 'bail|required|string|max:255',
        'medical_book' => 'bail|required|string|max:255',
        'self_employed' => 'bail|required|string|max:255',
        'inn' =>  [
            'bail',
            'nullable',
            'required_if:self_employed,1',
            'numeric',
```

```
            'integer',
            'digits:12',
            new InnValidationRule()
        ],
        'payment_account' => 'bail|nullable|required_if:self_employed,1|string|digits:20',
        'correspondent_account' => 'bail|nullable|required_if:self_employed,1|string|digits
        'bik' => 'bail|nullable|required_if:self_employed,1|string|digits:9',
        'bank_name' => 'bail|nullable|required_if:self_employed,1|string|max:255',
        'active' => 'bail|nullable|required_if:self_employed,1|string|max:255',
        'comment' => 'bail|nullable|string|max:255',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.employees');
}
```

IncomingStoreRequest

App\Http\Requests\Incomings

> Данный класс отвечает за валидацию входящих данных, необходимых для создания поступлений

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'payment_date' => 'bail|required|date',
        'legal_entity_id' => 'bail|required|numeric|integer|exists:legal_entities,id,delete
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'sum' => 'bail|required|numeric|integer|digits_between:1,15',
        'comment' => 'bail|nullable|string|max:256'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.incomings');
}
```

InvoiceStoreRequest

App\Http\Requests\Invoices

> Данный класс отвечает за валидацию входящих данных, необходимых для создания счетов

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'date' => 'bail|required|date',
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'invoice' => 'bail|required|string|max:50',
        'legal_entity_id' => 'bail|required|numeric|integer|exists:legal_entities,id,delete
        'sum' => 'bail|required|numeric|integer|digits_between:1,15',
        'paymented' => 'bail|nullable|numeric|integer|digits_between:1,15',
        'paymented_date' => 'bail|nullable|date',
        'comment' => 'bail|nullable|string|max:256'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.invoices');
}
```

LegalEntityStoreRequest

App\Http\Requests\LegalEntities

> Данный класс отвечает за валидацию входящих данных, необходимых для создания юридических лиц

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255',
        'inn' => [
            'bail',
            'nullable',
            'numeric',
            'integer',
            new InnValidationRule()
        ],
        'kpp' => 'bail|nullable|numeric|integer|digits:9',
        'orgn' => 'bail|nullable|numeric|integer|digits:13',
        'legal_address' => 'bail|nullable|string|max:255',
        'phone' => [
            'bail', 'nullable', 'string', 'max:255', new PhoneNumberRule()
        ],
        'email' => 'bail|nullable|string|email|max:255',
        'director_name' => 'bail|nullable|string|max:255',
        'director_position' => 'bail|nullable|string|max:255',
        'foundation' => 'bail|nullable|string|max:255',
        'contacts' => 'bail|nullable|array',

        'bank_requisites' => 'bail|nullable|array',
        'bank_requisites.*.payment_account' => 'bail|nullable|string|digits:20',
        'bank_requisites.*.bank_name' => 'bail|nullable|string|max:255',
        'bank_requisites.*.correspondent_account' => 'bail|nullable|string|digits:20',
        'bank_requisites.*.bik' => 'bail|nullable|string|digits:9',
        'bank_requisites.*.bank_address' => 'bail|nullable|string|max:255',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.legal_entities');
}
```

MainFilterRequest

App\Http\Requests\Main

> Данный класс отвечает за валидацию входящих данных, необходимых для фильтрации данных на главной странице

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'working_shift_date' => 'bail|nullable|date',
        'customer_id' => 'bail|nullable|exists:customers,id,deleted_at,NULL'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.orders');
}
```

OrderEmployeeStoreRequest

App\Http\Requests\Orders

> Данный класс отвечает за валидацию входящих данных, необходимых для добавления сотрудника в заявку

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'order_id' => 'bail|required|numeric|integer|exists:orders,id,deleted_at,NULL',
        'order_date' => 'bail|required|date',
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'subdivision_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:subdivisions,id,deleted_at,NULL'

        ]
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

OrderEmployeeUpdateRequest

App\Http\Requests\Orders

> Данный класс отвечает за валидацию входящих данных, необходимых для обновления сотрудника в заявке

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'employee_id' => 'bail|required|numeric|integer|exists:employees,id,deleted_at,NULL
        'subdivision_id' => 'bail|required|numeric|integer|exists:subdivisions,id,deleted_a
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

OrderStoreRequest

App\Http\Requests\Orders

> Данный класс отвечает за валидацию входящих данных, необходимых для создания заявки

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'is_master_order' => 'bail|required|boolean',
        'type' => [
            'bail',
            'required',
            'string',
            Rule::in(OrderTypeEnum::getValues())
        ],
        'order_date' => 'bail|required|date',
        'working_shift_date' => 'bail|required|date',
        'working_shift_end_date' => 'bail|nullable|after_or_equal:working_shift_date',
        'order_items' => 'bail|required|array',
        'order_items.*.cost_transportation' => 'bail|required|numeric|integer|min:0|max:429
        'order_items.*.deadline' => 'bail|required|date_format:d.m.Y H:i',
        'order_items.*.male' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'order_items.*.female' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'order_items.*.number_buses_there' => 'bail|required|numeric|integer|min:0|max:4294
        'order_items.*.number_buses_back' => 'bail|required|numeric|integer|min:0|max:42949
        'order_items.*.number_buses_there_back' => 'bail|required|numeric|integer|min:0|max
        'order_items.*.place_shipment' => 'bail|required|string|min:0|max:20',
        'order_items.*.plan' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'order_items.*.rate_employee' => 'bail|required|numeric|integer|min:0|max:429496729
        'order_items.*.working_hour_id' => 'bail|required|numeric|integer|exists:working_ho
        'order_items.*.subdivision_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:subdivisions,id,deleted_at,NULL',
            Rule::unique('orders')->where(function ($query) {
                return $query->where('customer_id', $this->input('customer_id'))
                    ->where('working_shift_date', $this->input('working_shift_date'))
                    ->where('working_hour_id', $this->input('order_items.*.working_hour_id'
                    ->whereNull('deleted_at');
            })
        ],
        'order_items.*.working_shift_id' => 'bail|required|integer|exists:working_shifts,id
        'order_items.*.working_shift_time' => 'bail|required|date_format:H:i',
        'order_items.*.working_shift_start_time' => 'bail|required|date_format:H:i',
        'order_items.*.working_shift_end_time' => 'bail|required|date_format:H:i|after:work
```

```
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.orders');
}
```

OrderUpdateRequest

App\Http\Requests\Orders

> Данный класс отвечает за валидацию входящих данных, необходимых для создания заявки

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'customer_id' => 'bail|required|numeric|integer|exists:customers,id,deleted_at,NULL
        'order_date' => 'bail|required|date',
        'working_shift_date' => 'bail|required|date',
        'working_shift_end_date' => 'bail|nullable|after_or_equal:working_shift_date',
        'cost_transportation' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'deadline' => 'bail|required|date_format:d.m.Y H:i',
        'male' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'female' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'number_buses_there' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'number_buses_back' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'number_buses_there_back' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'place_shipment' => 'bail|required|string|min:0|max:20',
        'plan' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'rate_employee' => 'bail|required|numeric|integer|min:0|max:4294967295',
        'working_hour_id' => 'bail|required|numeric|integer|exists:working_hours,id,deleted
        'subdivision_id' => [
            'bail',
            'required',
            'numeric',
            'integer',
            'exists:subdivisions,id,deleted_at,NULL',
            Rule::unique('orders')->ignore($this->order_id)->where(function ($query) {
                return $query->where('customer_id', $this->input('customer_id'))
                    ->where('working_shift_date', $this->input('working_shift_date'))
                    ->where('working_hour_id', $this->input('order_items.*.working_hour_id'
                    ->whereNull('deleted_at');
            })
        ],
        'working_shift_id' => 'bail|required|numeric|integer|exists:working_shifts,id,delet
        'working_shift_time' => 'bail|required|date_format:H:i',
        'working_shift_start_time' => 'bail|required|date_format:H:i',
        'working_shift_end_time' => 'bail|required|date_format:H:i|after:working_shift_star
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.orders');
}
```

EmployeeBonusStoreRequest

App\Http\Requests\References\EmployeeRatings

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике статусы сотрудников

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'working_shift_count' => 'bail|required|numeric|integer|digits_between:1,12',
        'bonus' => 'bail|required|decimal:0,2|max:9999999999.99',
        'subdivisions' => 'bail|nullable|array',
        'subdivisions.*' => 'bail|nullable|numeric|integer|exists:subdivisions,id,deleted_a
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
     return __('attributes.references.employee_bonuses');
}
```

EmployeeRatingStoreRequest

App\Http\Requests\References\EmployeeRatings

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике рейтинг сотрудников

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
    {
        return [
            'name' => 'bail|required|string|max:255',
            'absence_count' => 'bail|required|numeric|integer|digits_between:1,12',
            'days_count' => 'bail|required|numeric|integer|digits_between:1,12',
        ];
    }
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.references.employee_ratings');
}
```

EmployeeStatusStoreRequest

App\Http\Requests\References\EmployeeStatuses

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике статусы сотрудников

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255',
        'description' => 'bail|nullable|string|max:3000',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
     return __('attributes.references.employee_statuses');
}
```

PenaltyStoreRequest

App\Http\Requests\References\Penalties

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике штрафы

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.references.penalties');
}
```

PercentageBonusStoreRequest

App\Http\Requests\References\PercentageBonuses

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике проценты премии

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255',
        'absence' => 'bail|required|decimal:0,2|max:9999999999.99',
        'bonus' => 'bail|required|decimal:0,2|max:9999999999.99',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.references.percentage_bonuses');
}
```

SubdivisionStoreRequest

App\Http\Requests\References\Subdivisions

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике подразделения

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255',
        'customer_id' => 'bail|bail|required|exists:customers,id,deleted_at,NULL',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.references.subdivisions');
}
```

WorkingHourStoreRequest

App\Http\Requests\References\WorkingHours

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочнике количество часов

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'name' => 'bail|required|integer|max:255',
    ];
}
```

ReferenceStoreRequest

App\Http\Requests\References

> Данный класс отвечает за валидацию входящих данных, необходимых для создание записей в справочниках

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'name' => 'bail|required|string|max:255',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.employees');
}
```

PayoutStoreRequest

App\Http\Requests\Salary\Payouts

> Данный класс отвечает за валидацию входящих данных, необходимых для создание выплаты на странице выплаты

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'employee_id' => 'bail|required|numeric|integer|exists:employees,id,deleted_at,NULL
        'date' => 'bail|required|date',
        'sum' => 'bail|required|numeric|integer|digits_between:1,15'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.salary.payouts');
}
```

WorkedShiftFilterRequest

App\Http\Requests\Salary\WorkedShifts

> Данный класс отвечает за валидацию входящих данных, необходимых для фильтрации отработанных смен

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'date' => 'bail|required|date',
        'customer' => 'bail|nullable|array',
        'customer.id' => 'bail|nullable|numeric|integer|exists:id,deleted_at,NULL',
    ];
}
```

BaseSearchRequest

App\Http\Requests\Search

> Данный класс отвечает за валидацию входящих данных, необходимых для поиска

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```
public function rules(): array
{
    return [
        'page' => 'nullable|integer',
        'q' => 'nullable|string|max:255'
    ];
}
```

RoleRequest

App\Http\Requests\Users

> Данный класс отвечает за валидацию входящих данных, необходимых для добавления сотрудника в бригады

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'display_name'  => [
            'bail',
            'required',
            'regex:/(^[a-zA-Za-яA-Я0-9 .]+$)+/iu',
            'string',
            'max:255',
            'min:3',
            Rule::unique('roles', 'display_name')
                ->ignore($this->role_user)
                ->whereNull('deleted_at')
        ],
        'color'         => 'bail|required|string|max:7|hex_color',
        'modules'       => 'required|array',
        'modules.*.*'   => 'bail|nullable|exists:permissions,id',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.roles');
}
```

UserRequest

App\Http\Requests\Users

> Данный класс отвечает за валидацию входящих данных, необходимых для создание пользователя

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
            'login' => ['bail','required', 'string', 'max:255',
                Rule::unique('users', 'login')
                    ->whereNull('deleted_at')
                    ->ignore($this->id)],
            'password' => [
                'bail',
                (!isset($this->user) ? 'required' : 'nullable'),
                'string',
                'max:255',
                'min:8',
                new PasswordRule()
            ],
            'full_name' => 'bail|required|string|max:50',
            'role_id' => 'bail|required|numeric|integer|exists:roles,id',
            'phone' => [
                'bail',
                'required',
                new PhoneNumberRule(),
                Rule::unique('users', 'phone')
                    ->whereNull('deleted_at')
                    ->ignore($this->id),
            ],
            'avatar' => 'bail|nullable|image|max:10240',
            'delete_avatar' => 'bail|nullable|boolean',
            'card_created_at' => 'bail|required|date',
            'employment_date' => 'bail|nullable|date',
            'working_position' => 'bail|nullable|string|max:50',
            'birthdate' => 'bail|required|date',
            'additional_phone' => ['bail', 'nullable', new PhoneNumberRule()],
            'dismissal_date' => 'bail|nullable|date',
            'comment' => 'bail|nullable|string|max:256',
            'coefficient' => 'bail|required|numeric|integer|digits_between:1,18',
            'salary' => 'bail|required|numeric|integer|digits_between:1,18',
            'active' => 'bail|required|boolean',
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```
public function attributes(): array
{
    return __('attributes.users');
}
```

WorkScheduleFormDataRequest

App\Http\Requests\Users

> Данный класс отвечает за валидацию входящих данных, необходимых для создание рабочего графика

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```php
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'user_id' => 'bail|nullable|numeric|integer|exists:users,id,deleted_at,NULL',
        'period' => 'bail|required|array',
        'period.*' => 'bail|required|date'
    ];
}
```

WorkScheduleStoreRequest

App\Http\Requests\Users

> Данный класс отвечает за валидацию входящих данных, необходимых для создание рабочего графика

Список методов

authorize

Метод для проверки доступа пользователя к этому действию, всегда возвращает значение true, так как данный функционал реализован в политиках фреймворка `return array`

```
public function authorize(): bool
{
    return true;
}
```

rules

Метод содержит правила валидации для входящих полей `return array`

```php
public function rules(): array
{
    return [
        'user_id' => 'bail|required|numeric|integer|exists:users,id,deleted_at,NULL',
        'period' => ['bail', 'required', 'array', new IsOneWeekRule()],
        'period.*' => 'bail|required|date',
        'schedule' => 'bail|required|array',
        'schedule.*.morning_person' => 'bail|required|array',
        'schedule.*.morning_person.active' => 'bail|nullable|boolean',
        'schedule.*.morning_person.name' => 'bail|nullable|string|max:50',
        'schedule.*.morning_phone' => 'bail|nullable|boolean',
        'schedule.*.office' => 'bail|required|array',
        'schedule.*.office.from' => 'bail|nullable|string|max:50',
        'schedule.*.office.to' => 'bail|nullable|string|max:50',
        'schedule.*.office.in_home' => 'bail|nullable|boolean',
        'schedule.*.evening_person' => 'bail|required|array',
        'schedule.*.evening_person.active' => 'bail|nullable|boolean',
        'schedule.*.evening_person.name' => 'bail|nullable|string|max:50',
        'schedule.*.evening_phone' => 'bail|nullable|boolean'
    ];
}
```

attributes

Метод возвращающий массив перевода ошибок для данного класса `return array`

```php
public function attributes(): array
{
    return __('attributes.work_schedules');
}
```

BrigadeAdditionalDTResource

App\Http\Resources\Brigades

> Данный ресурс отвечает за преобразование входящих данных о бригаде в массив, для их последующего отображения в виде дополнительной таблицы на странице бригады

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'name' => $this->name,
        'plan' => $this->plan,
        'fact' => $this->fact,
        'came_out' => $this->getAttribute('came_out'),
        'not_came_out' => $this->getAttribute('not_came_out'),
        'day_off' => $this->getAttribute('day_off'),
        'not_confirmed' => $this->getAttribute('not_confirmed')
    ];
}
```

BrigadeCustomerResource

App\Http\Resources\Brigades

> Данный ресурс отвечает за преобразование входящих данных о заказчике бригады в массив, для их последующего отображения на странице бригады

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'subdivisions' => JsonResource::collection($this->whenLoaded('subdivisions'))
    ];
}
```

BrigadeDTResource

App\Http\Resources\Brigades

> Данный ресурс отвечает за преобразование входящих данных о бригаде в массив, для их последующего отображения в виде таблицы на странице бригады

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'brigade_status_id' => $this->brigade_status_id,
        'subdivision' => new JsonResource($this->whenLoaded('subdivision')),
        'employee' => new JsonResource($this->whenLoaded('employee')),
        'actions' => ['edit']
    ];
}
```

WorkShiftDTResource

App\Http\Resources\Brigades

> Данный ресурс отвечает за преобразование входящих данных о смене сотрудника бригады в массив, для их последующего отображения в виде таблицы на странице редактирования смен

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'order_date' => $this->order_date->format('d.m.Y'),
        'working_shift_date' => $this->working_shift_date->format('d.m.Y'),
        'subdivision' => new JsonResource($this->whenLoaded('subdivision')),
        'customer' => new JsonResource($this->whenLoaded('customer')),
        'workingShift' => new JsonResource($this->whenLoaded('workingShift')),
        'working_shift_time' => Carbon::parse($this->working_shift_time)->format('H:i'),
        'working_shift_start_time' => Carbon::parse($this->working_shift_start_time)->forma
        'working_shift_end_time' => Carbon::parse($this->working_shift_end_time)->format('H
        'workingHour' => new JsonResource($this->whenLoaded('workingHour')),
        'rate_employee' => $this->rate_employee,
        'plan' => $this->plan,
        'brigades_count' => $this->brigades_count,
        'male' => $this->male,
        'female' => $this->female,
        'deadline' => $this->deadline->format('H:i d.m.Y'),
        'orderStatus' => new JsonResource($this->whenLoaded('orderStatus')),
        'brigades' => JsonResource::collection($this->whenLoaded('brigades')),
        'actions' => $this->brigades->isEmpty() ? ['edit'] : ['destroy']
    ];
}
```

CustomerDTResource

App\Http\Resources\Customers

> Данный ресурс отвечает за преобразование входящих данных о заказчиках в массив, для их последующего отображения в виде таблицы на странице заказчики

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'actual_location' => $this->actual_location,
        'current_contract_number' => $this->current_contract_number,
        'current_additional_agreement_number' => $this->current_additional_agreement_number
        'legalEntity' => new JsonResource($this->whenLoaded('legalEntity')),
        'customerRequisite' => new JsonResource($this->whenLoaded('customerRequisite')),
        'billingPeriod' => new JsonResource($this->whenLoaded('billingPeriod')),
    ];
}
```

CustomerExportAllResource

App\Http\Resources\Customers

> Данный ресурс отвечает за преобразование входящих данных о заказчике в массив, для их последующего экспорта

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```php
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'actual_location' => $this->actual_location,
        'working_hours_count' => $this->working_hours_count,
        'employees_involved_count' => $this->employees_involved_count,
        'current_contract_number' => $this->current_contract_number,
        'current_additional_agreement_number' => $this->current_additional_agreement_number
        'billing_period_id' => $this->billingPeriod?->name,
        'rate_customers' => $this->rate_customers,
        'rate_employees' => $this->rate_employees,
        'cost_transport_there_customer' => $this->cost_transport_there_customer,
        'cost_transport_back_customer' => $this->cost_transport_back_customer,
        'cost_transport_round_trip_customer' => $this->cost_transport_round_trip_customer,
        'cost_transport_there_tk' => $this->cost_transport_there_tk,
        'cost_transport_back_tk' => $this->cost_transport_back_tk,
        'cost_transport_round_trip_tk' => $this->cost_transport_round_trip_tk,
        'brigadiers_x1_rub' => $this->brigadiers_x1_rub,
        'brigadiers_x2_rub' => $this->brigadiers_x2_rub,
        'brigadiers_personal_rub' => $this->brigadiers_personal_rub,
        'legal_entity_id' => $this->legalEntity?->name,
        'active' => $this->active,
        'contract_start_date' => $this->contract_start_date,
        'contract_end_date' => $this->contract_end_date,
        'comment' => $this->comment,
    ];
}
```

CustomerExportRequisitesResource

App\Http\Resources\Customers

> Данный ресурс отвечает за преобразование входящих данных о заказчике в массив, для их последующего экспорта

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $bank_requisites = '';
    $contacts = '';

    foreach ($this->customerRequisite?->bank_requisites ?? [] as $key => $requisite) {
        $bank_requisites = sprintf(
            '%s%s %d%s',
            $bank_requisites,
            __('attributes.bank_requisites.title'),
            $key + 1,
            PHP_EOL
        );
        foreach ($requisite as $elementKey => $element) {
            $bank_requisites = sprintf(
                '%s%s: %s%s',
                $bank_requisites,
                __('attributes.bank_requisites.' . $elementKey),
                $element,
                PHP_EOL
            );
        }
    }
    foreach ($this->customerRequisite?->contacts ?? [] as $key => $contact) {
        $contacts = sprintf(
            '%s%s %d%s',
            $contacts,
            __('attributes.contacts.title'),
            $key + 1,
            PHP_EOL
        );
        foreach ($contact as $elementKey => $element) {
            if ($elementKey == 'phones') {
                $element = implode(',', $element);
            }
            $contacts = sprintf(
                '%s%s: %s%s',
                $contacts,
                __('attributes.contacts.' . $elementKey),
                $element,
                PHP_EOL
            );
        }
    }
```

```
        return [
            'id' => $this->id,
            'customer_id' => $this->name,
            'name' => $this->customerRequisite?->name,
            'inn' => $this->customerRequisite?->inn,
            'kpp' => $this->customerRequisite?->kpp,
            'orgn' => $this->customerRequisite?->orgn,
            'legal_address' => $this->customerRequisite?->legal_address,
            'phone' => $this->customerRequisite?->phone,
            'email' => $this->customerRequisite?->email,
            'director_name' => $this->customerRequisite?->director_name,
            'director_position' => $this->customerRequisite?->director_position,
            'foundation' => $this->customerRequisite?->foundation,
            'bank_requisites' => $bank_requisites,
            'contacts' => $contacts,
        ];
    }
```

CustomerFormResource

App\Http\Resources\Customers

> Данный ресурс отвечает за преобразование входящих данных о заказчике в массив, для их последующего отображения в форме странице создание/обновление заказчика

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'class' => Customer::class,
        'id' => $this->id,
        'name' => $this->name,
        'actual_location' => $this->actual_location,
        'working_hours_count' => $this->working_hours_count,
        'employees_involved_count' => $this->employees_involved_count,
        'current_contract_number' => $this->current_contract_number,
        'contract_start_date' => $this->contract_start_date ? $this->contract_start_date->f
        'contract_end_date' => $this->contract_end_date,
        'current_additional_agreement_number' => $this->current_additional_agreement_number
        'billing_period_id' => $this->billing_period_id,
        'rate_customers' => $this->rate_customers,
        'rate_employees' => $this->rate_employees,
        'cost_transport_there_customer' => $this->cost_transport_there_customer,
        'cost_transport_back_customer' => $this->cost_transport_back_customer,
        'cost_transport_round_trip_customer' => $this->cost_transport_round_trip_customer,
        'cost_transport_there_tk' => $this->cost_transport_there_tk,
        'cost_transport_back_tk' => $this->cost_transport_back_tk,
        'cost_transport_round_trip_tk' => $this->cost_transport_round_trip_tk,
        'brigadiers_x1_rub' => $this->brigadiers_x1_rub,
        'brigadiers_x2_rub' => $this->brigadiers_x2_rub,
        'brigadiers_personal_rub' => $this->brigadiers_personal_rub,
        'legal_entity_id' => $this->legal_entity_id,
        'active' => $this->active,
        'comment' => $this->comment,
        'customerRequisite' => new JsonResource($this->whenLoaded('customerRequisite'))
    ];
}
```

CustomerRequisiteDTResource

App\Http\Resources\Customers

> Данный ресурс отвечает за преобразование входящих данных о заказчике в массив, для их последующего отображения в виде таблице на странице заказчика

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'legal_address' => $this->legal_address,
        'phone' => $this->phone,
        'email' => $this->email,
        'director_name' => $this->director_name,
        'actions' => [ 'edit', 'destroy' ]
    ];
}
```

CustomerRequisiteFormResource

App\Http\Resources\Customers

> Данный ресурс отвечает за преобразование входящих данных о заказчике в массив, для их последующего отображения в форме на странице заказчика

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'customer_id' => $this->customer_id,
        'name' => $this->name,
        'inn' => $this->inn,
        'kpp' => $this->kpp,
        'orgn' => $this->orgn,
        'legal_address' => $this->legal_address,
        'phone' => $this->phone,
        'email' => $this->email,
        'director_name' => $this->director_name,
        'director_position' => $this->director_position,
        'foundation' => $this->foundation,
        'bank_requisites' => $this->bank_requisites,
        'contacts' => $this->contacts,
    ];
}
```

EmployeeResource

App\Http\Resources\Dashboard\Employees

> Данный ресурс отвечает за преобразование входящих данных о сотрудниках в массив, для их последующего отображения в виде диаграммы на странице дашборды

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'alias' => $this->alias,
        'employees_count' => $this->employees_count
    ];
}
```

ManagerActivityResource

App\Http\Resources\Dashboard\ManagerActivity

> анный ресурс отвечает за преобразование входящих данных об активности менеджеров в массив, для их последующего отображения в виде диаграммы на странице дашборды

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->full_name,
        'brigades_count' => $this->brigades_count,
        'brigades_count_came_out' => $this->getAttribute('brigades_count_came_out')
    ];
}
```

OrderResource

App\Http\Resources\Dashboard\Orders

> Данный ресурс отвечает за преобразование входящих данных о заказах в массив, для их последующего отображения в виде диаграммы на странице дашборды

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'color' => $this->color,
        'name' => $this->name,
        'orders_count' => $this->orders_count
    ];
}
```

PaymentResource

App\Http\Resources\Dashboard\Payments

> Данный ресурс отвечает за преобразование входящих данных о выплатах в массив, для их последующего отображения в виде диаграммы на странице дашборды

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'date' => Carbon::parse($this->getAttribute('date'))->format('d.m.Y'),
        'sum' => $this->getAttribute('sum')
    ];
}
```

SeasonalActivityResource

App\Http\Resources\Dashboard\SeasonalActivity

> Данный ресурс отвечает за преобразование входящих данных о сезонной активности в массив, для их последующего отображения в виде диаграммы на странице дашборды

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'date' => $request->type === 'month' ?
            Carbon::create(null, $this->getAttribute('date'))->getTranslatedMonthName('MMM'
            : $this->getAttribute('date'),
        'total' => $this->getAttribute('total'),
        'fact_total' => $this->getAttribute('fact_total'),
        'plan_total' => $this->getAttribute('plan_total')
    ];
}
```

DocumentDTResource

App\Http\Resources\Documents

> Данный ресурс отвечает за преобразование входящих данных о документах в массив, для их последующего отображения в виде таблице на странице документы

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $file = $this->getMedia('file')->first();
    return [
        'id' => $this->id,
        'name' => $this->name,
        'comment' => $this->comment,
        'created_at' => $this->created_at->format('Y-m-d'),
        'media_file' => new MediaDTResource($file),
        'actions' => [ 'edit', 'destroy' ]
    ];
}
```

DocumentFormResource

App\Http\Resources\Documents

> Данный ресурс отвечает за преобразование входящих данных о документах в массив, для их последующего отображения в форме странице документы

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $file = $this->getMedia('file')->first();
    return [
        'id' => $this->id,
        'name' => $this->name,
        'comment' => $this->comment,
        'created_at' => $this->created_at->format('Y-m-d'),
        'media_file' => new MediaDTResource($file),
    ];
}
```

EmployeeDTResource

App\Http\Resources\Employees

> Данный ресурс отвечает за преобразование входящих данных о сотрудниках в массив, для их последующего отображения в виде таблице на странице сотрудники

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'full_name' => $this->full_name,
        'gender' => GenderEnum::getDescription($this->gender),
        'employeeRating' => new JsonResource($this->whenLoaded('employeeRating')),
        'employeeStatus' => new JsonResource($this->whenLoaded('employeeStatus')),
        'phones' => $this->phones,
        'workingShifts' => JsonResource::collection($this->whenLoaded('workingShifts')),
        'where_worked' => null,
        'working_shift_last_date' => null,
        'comment' => $this->comment,
        'birthdate' => $this->birthdate->format('Y-m-d'),
        'actions' => [ 'edit', 'destroy' ]
    ];
}
```

EmployeeFormResource

App\Http\Resources\Employees

> Данный ресурс отвечает за преобразование входящих данных о документах в массив, для их последующего отображения в форме создания/редактирования сотрудников

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```php
public function toArray(Request $request): array
{
    $avatar = $this->getMedia('avatar')->first();
    return [
        'media_avatar' => new MediaDTResource($avatar),
        'card_created_at' => $this->card_created_at,
        'user_id' => $this->user_id,
        'user' => new JsonResource($this->whenLoaded('user')),
        'working_shifts' => $this->working_shifts['ids'] ?? [],
        'full_name' => $this->full_name,
        'birthdate' => $this->birthdate->format('Y-m-d'),
        'gender' => $this->gender,
        'phones' => $this->phones,
        'additional_phones' => $this->additional_phones,
        'passport_serial_number' => $this->passport_serial_number,
        'issued' => $this->issued,
        'issue_date' => $this->issue_date->format('Y-m-d'),
        'registration_address' => $this->registration_address,
        'fact_address' => $this->fact_address,
        'job_resource_id' => $this->job_resource_id,
        'whence' => $this->whence,
        'skills' => $this->skills,
        'health_restrictions' => $this->health_restrictions,
        'medical_book' => $this->medical_book,
        'self_employed' => $this->self_employed,
        'inn' => $this->inn,
        'payment_account' => $this->payment_account,
        'correspondent_account' => $this->correspondent_account,
        'bik' => $this->bik,
        'bank_name' => $this->bank_name,
        'active' => $this->active,
        'comment' => $this->comment,
    ];
}
```

GeneralReportDTResource

App\Http\Resources\GeneralReports

> Данный ресурс отвечает за преобразование входящих данных в массив, для их последующего отображения в виде таблице на странице общий отчёт

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'working_shift_date' => $this->working_shift_date,
        'cost_transportation' => $this->cost_transportation,
        'rate_customer' => $this->getAttribute('rate_customer'),
        'working_shift_duration' => round((float)$this->getAttribute('working_shift_duratio
        'brigades_count' => $this->getAttribute('brigades_count'),
        'total_working_hours' => round((float)$this->getAttribute('total_working_hours'), 1
        'cost_work' => round((float)$this->getAttribute('cost_work'), 2),
        'transfer' => $this->getAttribute('transfer'),
        'total' => round((float)$this->getAttribute('total'), 2),
        'employee_payment' => $this->getAttribute('employee_payment'),
        'income' => round((float)$this->getAttribute('income'), 2),
        'customer' => new JsonResource($this->whenLoaded('customer')),
        'workingShift' => new JsonResource($this->whenLoaded('workingShift'))
    ];
}
```

IncomingDTResource

App\Http\Resources\Incomings

> последующего отображения в виде таблице на странице поступления

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'payment_date' => $this->payment_date,
        'customer' => new JsonResource($this->whenLoaded('customer')),
        'indebtedness' => null,
        'legalEntity' => new JsonResource($this->whenLoaded('legalEntity')),
        'sum' => $this->sum,
        'comment' => $this->comment,
        'actions' => [ 'edit', 'destroy' ]
    ];
}
```

IncomingFormResource

App\Http\Resources\Incomings

> Данный ресурс отвечает за преобразование входящих данных о поступлениях в массив, для их последующего отображения в форме создания/редактирования поступлений

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'payment_date' => $this->payment_date,
        'legal_entity_id' => $this->legal_entity_id,
        'customer_id' => $this->customer_id,
        'sum' => $this->sum,
        'comment' => $this->comment
    ];
}
```

InvoiceDTResource

App\Http\Resources\Invoices

> Данный ресурс отвечает за преобразование входящих данных о счетах в массив, для их последующего отображения в виде таблице на странице счета

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'date' => $this->date,
        'customer' => new JsonResource($this->whenLoaded('customer')),
        'invoice' => $this->invoice,
        'legalEntity' => new JsonResource($this->whenLoaded('legalEntity')),
        'sum' => $this->sum,
        'paymented' => $this->paymented,
        'paymented_date' => $this->paymented_date,
        'comment' => $this->comment,
        'actions' => [ 'edit', 'destroy' ]
    ];
}
```

InvoiceFormResource

App\Http\Resources\Invoices

> Данный ресурс отвечает за преобразование входящих данных о счетах в массив, для их последующего отображения в форме создания/редактирования счетов

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'payment_date' => $this->paymented_date,
        'legal_entity_id' => $this->legal_entity_id,
        'customer_id' => $this->customer_id,
        'sum' => $this->sum,
        'comment' => $this->comment
    ];
}
```

LegalEntityDTResource

App\Http\Resources\LegalEntities

> Данный ресурс отвечает за преобразование входящих данных о юридических лицах в массив, для их последующего отображения в виде таблице на юридических лиц

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'legal_address' => $this->legal_address,
        'phone' => $this->phone,
        'email' => $this->email,
        'director_name' => $this->director_name,
        'actions' => [ 'edit', 'destroy' ]
    ];
}
```

LegalEntityFormResource

App\Http\Resources\LegalEntities

> анный ресурс отвечает за преобразование входящих данных о юридических лицах в массив, для их последующего отображения в форме создания/редактирования юридических лиц

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'class' => LegalEntity::class,
        'id' => $this->id,
        'name' => $this->name,
        'inn' => $this->inn,
        'kpp' => $this->kpp,
        'orgn' => $this->orgn,
        'legal_address' => $this->legal_address,
        'phone' => $this->phone,
        'email' => $this->email,
        'director_name' => $this->director_name,
        'director_position' => $this->director_position,
        'foundation' => $this->foundation,
        'bank_requisites' => $this->bank_requisites,
        'contacts' => $this->contacts,
    ];
}
```

MainResource

App\Http\Resources\Main

> Данный ресурс отвечает за преобразование входящих данных о заявках в массив, для их последующего отображения на главной странице

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'plan' => $this->plan,
        'fact' => $this->brigades_count,
        'working_shift_date' => $this->working_shift_date->format('Y-m-d'),
        'deadline_date' => $this->deadline->format('d.m.Y'),
        'deadline_time' => $this->deadline->format('H:i'),
        'customer' => new JsonResource($this->whenLoaded('customer')),
        'subdivision' => new JsonResource($this->whenLoaded('subdivision')),
        'orderStatus' => new JsonResource($this->whenLoaded('orderStatus')),
        'workingShift' => new JsonResource($this->whenLoaded('workingShift'))
    ];
}
```

MediaDTResource

App\Http\Resources\Media

> Данный ресурс отвечает за преобразование входящих данных о медиа файлах в массив, для их последующего отображения

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'file_name' => $this->file_name,
        'uuid' => $this->uuid,
        'url' => route('file_private', $this->uuid),
        'previewUrl' => $this->hasGeneratedConversion('preview')
            ? route('file_private_preview', $this->uuid)
            : null,
    ];
}
```

NotificationDTResource

App\Http\Resources\Notifications

> Данный ресурс отвечает за преобразование входящих данных о уведомлениях в массив, для их последующего отображения

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'created_at' => $this->created_at->format('Y-m-d'),
        'message' => $this->message,
        'read_at' => $this->created_at->format('Y-m-d')
    ];
}
```

BrigadeAllOrderResource

App\Http\Resources\Brigades

> Данный ресурс отвечает за преобразование входящих данных о бригадах в последней заявке в массив, для их последующего использования в заявках

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'order' => new OrderResource($this->whenLoaded('order'))
    ];
}
```

BrigadeLatestOrderResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о бригадах в последней заявке в массив, для их последующего использования в заявках

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'status' => $this->brigade_status_id,
        'latestOrder' => new OrderResource($this->whenLoaded('latestOrder'))
    ];
}
```

OrderDTResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о заявках в массив, для их последующего отображения в виде таблицы на странице заявки

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'order_date' => $this->order_date->format('d.m.Y'),
        'working_shift_date' => $this->working_shift_date->format('d.m.Y'),
        'subdivision' => new JsonResource($this->whenLoaded('subdivision')),
        'working_shift_time' => Carbon::parse($this->working_shift_time)->format('H:i'),
        'working_shift_start_time' => Carbon::parse($this->working_shift_start_time)->forma
        'working_shift_end_time' => Carbon::parse($this->working_shift_end_time)->format('H
        'workingHour' => new JsonResource($this->whenLoaded('workingHour')),
        'rate_employee' => $this->rate_employee,
        'brigades_count' => $this->brigades_count,
        'plan' => $this->plan,
        'male' => $this->male,
        'female' => $this->female,
        'deadline' => $this->deadline->format('H:i d.m.Y'),
        'orderStatus' => new JsonResource($this->whenLoaded('orderStatus')),
        'actions' => ['destroy']
    ];
}
```

OrderEmployeeDateResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о сотрудниках из заявок в массив, для их последующего отображения в заявках

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'order_date' => $this->order_date->format('d.m.Y'),
        'subdivisions' => OrderEmployeeSubdivisionResource::collection(Order::select('subdi
            ->distinct()
            ->with('subdivision')
            ->where('customer_id', $this->customer_id)
            ->where('order_date', $this->order_date)
            ->get())
    ];
}
```

OrderEmployeeDTResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о сотрудниках из заявок в массив, для их последующего отображения в виде таблицы в заявках

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'employee' => new EmployeeDTResource($this->whenLoaded('employee')),
        'subdivision' => new JsonResource($this->whenLoaded('subdivision')),
        'actions' => ['edit', 'destroy']
    ];
}
```

OrderEmployeeFormResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о сотрудниках из заявок в массив, для их последующего отображения в форме редактирования заявки

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $avatar = $this->getMedia('avatar')->first();
    return [
        'id' => $this->id,
        'media_avatar' => new MediaDTResource($avatar),
        'full_name' => $this->full_name,
        'birthdate' => (int)$this->birthdate->diffInYears(Carbon::now()),
        'gender' => $this->gender ? GenderEnum::getDescription($this->gender) : null,
        'phones' => $this->phones,
        'fact_address' => $this->fact_address,
        'latestEvent' => new BrigadeLatestOrderResource($this->whenLoaded('latestBrigade'))
        'employeeRating' => new JsonResource($this->whenLoaded('employeeRating')),
        'employeeStatus' => new JsonResource($this->whenLoaded('employeeStatus')),
        'comment' => $this->comment
    ];
}
```

OrderEmployeeRecommendationDTResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о рекомендованных сотрудниках для заявок в массив, для их последующего отображения в виде таблице на странице редактирования заявки

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'full_name' => $this->full_name,
        'gender' => GenderEnum::getDescription($this->gender),
        'workingShifts' => JsonResource::collection($this->whenLoaded('workingShifts')),
        'employeeRating' => new JsonResource($this->whenLoaded('employeeRating')),
        'employeeStatus' => new JsonResource($this->whenLoaded('employeeStatus')),
        'customers' => BrigadeAllOrderResource::collection($this->whenLoaded('brigades')),
        'comment' => $this->comment,
        'actions' => ['edit']
    ];
}
```

OrderEmployeeSubdivisionResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о подразделениях сотрудников из заявок в массив, для их последующего использования

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->subdivision->id,
        'name' => $this->subdivision->name
    ];
}
```

OrderFormResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о заявках в массив, для их последующего отображения в форме создания заявок

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```php
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'order_date' => $this->order_date->format('Y-m-d'),
        'working_shift_date' => $this->working_shift_date->format('Y-m-d'),
        'subdivisions' => new JsonResource($this->whenLoaded('subdivision')),
        'working_shift_time' => Carbon::parse($this->working_shift_time)->format('H:i'),
        'working_shift_start_time' => Carbon::parse($this->working_shift_start_time)->forma
        'working_shift_end_time' => Carbon::parse($this->working_shift_end_time)->format('H
        'workingHour' => new JsonResource($this->whenLoaded('workingHour')),
        'rate_employee' => $this->rate_employee,
        'plan' => $this->plan,
        'male' => $this->male,
        'female' => $this->female,
        'fact' => $this->getAttribute('fact_male') + $this->getAttribute('fact_female'),
        'fact_male' => $this->getAttribute('fact_male'),
        'fact_female' => $this->getAttribute('fact_female'),
        'deadline' => $this->deadline->format('d.m.Y H:i'),
        'orderStatus' => new JsonResource($this->whenLoaded('orderStatus')),
        'number_buses_there' => $this->number_buses_there,
        'number_buses_back' => $this->number_buses_back,
        'number_buses_there_back' => $this->number_buses_there_back,
        'place_shipment' => $this->place_shipment,
        'customer_id' => $this->customer_id,
        'subdivision_id' => $this->subdivision_id,
        'working_hour_id' => $this->working_hour_id,
        'working_shift_id' => $this->working_shift_id,
        'workingShift' => new JsonResource($this->whenLoaded('workingShift')),
        'other_orders' => $this->other_orders,
        'customer' => new JsonResource($this->whenLoaded('customer'))
    ];
}
```

OrderResource

App\Http\Resources\Orders

> Данный ресурс отвечает за преобразование входящих данных о заявках в массив, для их последующего использования

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'working_shift_date' => $this->working_shift_date->format('d.m.Y'),
        'customer' => new JsonResource($this->whenLoaded('customer'))
    ];
}
```

EmployeeBonusDTResource

App\Http\Resources\References\EmployeeBonuses

> Данный ресурс отвечает за преобразование входящих данных о премиях сотрудников в массив, для их последующего отображения в виде таблице в справочнике премии сотрудников

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'working_shift_count' => $this->working_shift_count,
        'bonus' => $this->bonus,
        'subdivisionModels' => JsonResource::collection($this->whenLoaded('subdivisionModel
        'actions' => isset($this->alias) ? [] : [ 'edit', 'destroy' ]
    ];
}
```

EmployeeBonusFormResource

App\Http\Resources\References\EmployeeBonuses

> Данный ресурс отвечает за преобразование входящих данных о премиях сотрудников в массив, для их последующего отображения в форме создания/редактирования в справочнике премии сотрудников

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'working_shift_count' => $this->working_shift_count,
        'bonus' => $this->bonus,
        'subdivisions' => $this->subdivisions['ids'] ?? []
    ];
}
```

EmployeeRatingDTResource

App\Http\Resources\References\EmployeeRatings

> Данный ресурс отвечает за преобразование входящих данных о рейтинге сотрудников в массив, для их последующего отображения в виде таблицы в справочнике рейтинг сотрудников

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'absence_count' => $this->absence_count,
        'days_count' => $this->days_count,
        'actions' => isset($this->alias) ? [] : [ 'edit', 'destroy' ]
    ];
}
```

EmployeeRatingFormResource

App\Http\Resources\References\EmployeeRatings

> Данный ресурс отвечает за преобразование входящих данных о рейтинге сотрудников в массив, для их последующего отображения в форме создания/редактирования в справочнике рейтинг сотрудников

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'absence_count' => $this->absence_count,
        'days_count' => $this->days_count,
    ];
}
```

EmployeeStatusDTResource

App\Http\Resources\References\EmployeeStatuses

> Данный ресурс отвечает за преобразование входящих данных о статусах сотрудников в массив, для их последующего отображения в виде таблицы в справочнике статусы сотрудников

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'description' => $this->description,
        'actions' => isset($this->alias) ? [] : [ 'edit', 'destroy' ]
    ];
}
```

PenaltyDTResource

App\Http\Resources\References\Penalties

> Данный ресурс отвечает за преобразование входящих данных о штрафах в массив, для их последующего отображения в виде таблицы в справочнике штрафы

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'sum' => $this->sum,
        'actions' => isset($this->alias) ? [] : [ 'edit', 'destroy' ]
    ];
}
```

PenaltyFormResource

App\Http\Resources\References\Penalties

> Данный ресурс отвечает за преобразование входящих данных о штрафах в массив, для их последующего отображения в форме создания/редактирования в справочнике штрафы

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'sum' => $this->sum,
    ];
}
```

PercentageBonusDTResource

App\Http\Resources\References\PercentageBonuses

> Данный ресурс отвечает за преобразование входящих данных о проценте премии в массив, для их последующего отображения в виде таблицы в справочнике процент премии

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'absence' => $this->absence,
        'bonus' => $this->bonus,
        'actions' => isset($this->alias) ? [] : [ 'edit', 'destroy' ]
    ];
}
```

PercentageBonusFormResource

App\Http\Resources\References\PercentageBonuses

> Данный ресурс отвечает за преобразование входящих данных о премиях в массив, для их последующего отображения в форме создания/редактирования в справочнике процент премии

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'absence' => $this->absence,
        'bonus' => $this->bonus,
    ];
}
```

SubdivisionDTResource

App\Http\Resources\References\Subdivisions

> Данный ресурс отвечает за преобразование входящих данных о подразделениях в массив, для их последующего отображения в виде таблицы в справочнике подразделения

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'customer' => $this->customer,
        'actions' => isset($this->alias) ? [] : [ 'edit', 'destroy' ]
    ];
}
```

SubdivisionFormResource

App\Http\Resources\References\Subdivisions

> Данный ресурс отвечает за преобразование входящих данных о подразделениях, для их последующего отображения в форме создания/редактирования в справочнике подразделения

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name,
        'customer_id' => $this->customer_id,
        'customer' => new JsonResource($this->whenLoaded('customer')),
    ];
}
```

AccrualDTResource

App\Http\Resources\Salary\Accruals

> Данный ресурс отвечает за преобразование входящих данных о начислениях в массив, для их последующего отображения в виде таблице на странице начисления

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $data = [
        'id' => $this[0]->id,
        'full_name' => $this[0]->full_name,
        'total' => $this[0]->total
    ];
    foreach ($this->resource as $key => $day) {
        $date = Carbon::parse($day->date);
        $data[$date->format('Y-m-d')] = [
            'id' => $day->accrual_id,
            'date' => $date->format('Y-m-d'),
            'sum' => $day->sum,
            'employee' => ['id' => $day->employee_id, 'full_name' => $this[0]->full_name],
            'employee_id' => $day->employee_id,
        ];
    }
    return $data;
}
```

FactDTResource

App\Http\Resources\Salary\Facts

> Данный ресурс отвечает за преобразование входящих данных о фактическом количество в массив, для их последующего отображения в виде таблице на странице факт

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $data = [
        'date' => $this[0]->date,
        'total' => $this[0]->fact_total,
        'in_plan' => $this[0]->plan_total,
        'delta' => abs($this[0]->plan_total - $this[0]->fact_total),
        'plan_completion' =>
            $this[0]->plan_total > 0 ? round(($this[0]->fact_total / $this[0]->plan_total)
    ];
    foreach ($this->resource as $key => $plan) {
        if ($plan->subdivision_id) {
            $data['subdivision-' . $plan->subdivision_id] = $plan->fact_sum;
        }
    }
    return $data;
}
```

PayoutDTResource

App\Http\Resources\Salary\Payouts

> Данный ресурс отвечает за преобразование входящих данных о выплатах в массив, для их последующего отображения в виде таблице на странице выплаты

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $data = [
        'id' => $this[0]->id,
        'full_name' => $this[0]->full_name,
        'total' => $this[0]->total
    ];
    foreach ($this->resource as $key => $day) {
        $date = Carbon::parse($day->date);
        $data[$date->format('Y-m-d')] = [
            'id' => $day->payout_id,
            'date' => $date->format('Y-m-d'),
            'sum' => $day->sum,
            'employee' => ['id' => $day->employee_id, 'full_name' => $this[0]->full_name],
            'employee_id' => $day->employee_id,
        ];
    }
    return $data;
}
```

PlanDTResource

App\Http\Resources\Salary\Plans

> Данный ресурс отвечает за преобразование входящих данных о плановом количестве в массив, для их последующего отображения в виде таблице на странице план

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $data = [
        'date' => $this[0]->date,
        'total' => $this[0]->total
    ];
    foreach ($this->resource as $key => $plan) {
        if ($plan->subdivision_id) {
            $data['subdivision-' . $plan->subdivision_id] = $plan->sum;
        }
    }
    return $data;
}
```

BrigadeAdditionalDTResource

App\Http\Resources\Salary\Summaries

> Данный ресурс отвечает за преобразование входящих данных о сводах в массив, для их последующего отображения в виде таблице на странице свод

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    foreach ($this->resource as $key => $cusomer) {
        foreach ($cusomer as $subdivisionKey => $subdivision) {
            // dump($subdivision);
            $data['subdivision-' . $subdivision->subdivision_id][] = [
                'sum' => $subdivision->sum,
            ];
        }
    }
    // dd($data);
    return $data;
}
```

WorkedShiftDTResource

App\Http\Resources\Salary\WorkedShifts

> Данный ресурс отвечает за преобразование входящих данных о отработанные смены в массив, для их последующего отображения в виде таблице на странице отработанные смены

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'employee' => new JsonResource($this->whenLoaded('employee')),
        'date' => $this->date,
        'order_id' => $this->order_id,
        'comment' => $this->comment,
        'worked_id' => $this->worked_id,
        'accrued' => $this->accrued,
        'brigadiers' => $this->brigadiers
    ];
}
```

WorkedShiftFormResource

App\Http\Resources\Salary\WorkedShifts

> Данный ресурс отвечает за преобразование входящих данных о отработанные смены в массив, для их последующего отображения в форме создания смен

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return parent::toArray($request);
}
```

ActionHistoryDTResource

App\Http\Resources\Users

> Данный ресурс отвечает за преобразование входящих данных истории действий пользователя в массив, для их последующего отображения в виде таблицы на странице история действий

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'created_at' => $this->created_at->format('Y-m-d H:i:s'),
        'description' => $this->description,
        'subject_id' => $this->subject_id,
        'properties' => $this->properties,
        'causer' => $this->causer,
        'causer_name' => $this->causer_name,
        'section' => isset($this->properties['section'])
            ? SectionEnum::getDescription($this->properties['section'])
            : null,
        'subsection' => isset($this->properties['subsection'])
            ? SubsectionEnum::getDescription($this->properties['subsection'])
            : null
    ];
}
```

RoleElementResource

App\Http\Resources\Users

> Данный ресурс отвечает за преобразование входящих данных о роли пользователя в массив, для их последующего отображения

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id'            => $this->id,
        'color'         => $this->color,
        'display_name'  => $this->display_name,
        'modules'       => $this->when($this->relationLoaded('permissions'), $this->prepare
    ];
}
```

preparePermissions

Метод для подготовки доступов Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```php
private function preparePermissions(): array
{
    $config = config('roles') ;
    $modules = [];
    $notView = Arr::collapse($config['permissions_not_view']);
    $permissions = $this->permissions->pluck('id', 'name')->groupBy(function ($item, $key)
        return explode('_', $key)[0];
    })->toArray();

    foreach ($config['permissions'] as $key => $item) {
        if (in_array($key, $notView)) {
            continue;
        }
        $modules[$key] = isset($permissions[$key]) ? $permissions[$key] : [];
    }

    return $modules;
}
```

RoleTableResource

App\Http\Resources\Users

> Данный ресурс отвечает за преобразование входящих данных о роли пользователя в массив, для их последующего отображения в виде таблице

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id'            => $this->id,
        'name'          => $this->name,
        'display_name'  => $this->display_name,
        'created_at'    => $this->created_at,
        'color'         => $this->color,
        'actions'       => $this->id !== self::ADMIN ? [ 'edit', 'destroy' ] : []
    ];
}
```

UserAuthResource

App\Http\Resources\Users

> Данный ресурс отвечает за преобразование входящих данных об авторизованном пользователи в массив, для их последующего использования

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $roles = new Collection($this->whenLoaded('roles'));
    /**
     * @var RoleUser $role
     */
    $role = $this->roles->first();
    return [
        'id' => $this->id,
        'avatar' => $this->image ? route('file_private', $this->image) : null,
        'avatarPreview' => $this->image ? route('file_private_preview', $this->image) : nul
        'full_name' => $this->full_name,
        'email' => $this->email,
        'roles' => $roles->pluck('name'),
        'permissions' => $this->whenLoaded('permissions') ? $this->getAllPermissions()->plu
        'roleRu' => $role->display_name
    ];
}
```

UserDTResource

App\Http\Resources\Users

> Данный ресурс отвечает за преобразование входящих данных о пользователе в массив, для их последующего отображения в виде таблице на странице пользователи

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    $actions = [];
    if ($this->hasRole('admin') && Auth::user()->id == $this->id) {
        $actions = [ 'edit', 'destroy' ];
    } else {
        $actions = [ 'edit', 'destroy' ];
    }
    return [
        'id' => $this->id,
        'birthdate' => $this->birthdate,
        'full_name' => $this->full_name,
        'avatar' => $this->image ? route('file_private', $this->image) : null,
        'avatarPreview' => $this->image ? route('file_private_preview', $this->image) : nul
        'login' => $this->login,
        'working_position' => $this->working_position,
        'phone' => $this->phone,
        'comment' => $this->comment,
        'role' => $this->whenLoaded('roles') ? new RoleTableResource($this->roles->first())
        'workSchedule' => new WorkScheduleFormResource($this->whenLoaded('workSchedule')),
        'actions' => $actions
    ];
}
```

UserElementResource

App\Http\Resources\Users

> Данный ресурс отвечает за преобразование входящих данных о пользователе в массив, для их последующего использования

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    /**
     * @var RoleUser $role
     */
    $role = $this->roles->first();
    return [
        'id' => $this->id,
        'login' => $this->login,
        'full_name' => $this->full_name,
        'role_id' => $role->id,
        'phone' => $this->phone,
        'email' => $this->email,
        'avatar' => $this->image ? route('file_private', $this->image) : null,
        'avatarPreview' => $this->image ? route('file_private_preview', $this->image) : nul
        'card_created_at' => $this->card_created_at,
        'employment_date' => $this->employment_date,
        'working_position' => $this->working_position,
        'birthdate' => $this->birthdate,
        'additional_phone' => $this->additional_phone,
        'dismissal_date' => $this->dismissal_date,
        'comment' => $this->comment,
        'coefficient' => $this->coefficient,
        'salary' => $this->salary,
        'active' => $this->active,
    ];
}
```

WorkScheduleFormResource

App\Http\Resources\WorkSchedules

> Данный ресурс отвечает за преобразование входящих данных о графике пользователя в массив, для их последующего отображения в виде таблицы на странице графи пользователя

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'period' => [$this->start_date, $this->end_date],
        'user' => new JsonResource($this->whenLoaded('user')),
        'user_id' => $this->user_id,
        'schedule' => json_decode($this->schedule),
    ];
}
```

BaseResource

App\Http\Resources

> Данный ресурс отвечает за преобразование входящих данных в массив, для их последующего использования

Список методов

toArray

Метод для формализации данных Параметры:

- Request $request - элемент коллекции

Ответ : `return array`

```
public function toArray(Request $request): array
{
    return [
        'id' => $this->id,
        'name' => $this->name
    ];
}
```

Brigade

App\Models\Brigades

> Данный класс содержит логику работы модели "Бригады", которая используется для взаимодействия с таблицей БД "brigades" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'order_id',
    'employee_id',
    'subdivision_id',
    'brigade_status_id',
    'day_off_start_date',
    'day_off_end_date',
    'user_id'
];

protected $casts = [
    'day_off_start_date' => 'date',
    'day_off_end_date' => 'date'
];
```

Список методов

boot

Статичный метод инициализирующий обсервер BrigadeObserver в момент создания класса модели

```
public static function boot()
{
    parent::boot();
    self::observe(new BrigadeObserver());
}
```

employee

Метод реализации обратной связи один ко многим с моделью Employee

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```php
public function employee(): BelongsTo
{
    return $this->belongsTo(Employee::class);
}
```

subdivision

Метод реализации обратной связи один к одному с моделью Subdivision

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```php
public function subdivision(): BelongsTo
{
    return $this->belongsTo(Subdivision::class);
}
```

brigadeStatus

Метод реализации обратной связи один к одному с моделью BrigadeStatus

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```php
public function brigadeStatus(): BelongsTo
{
    return $this->belongsTo(BrigadeStatus::class);
}
```

order

Метод реализации обратной связи один ко многим с моделью Order

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```php
public function order(): BelongsTo
{
    return $this->belongsTo(Order::class);
}
```

latestOrder

Метод реализации связи с последней заявкой Order

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```php
public function latestOrder(): BelongsTo
{
    return $this->order()->latest();
}
```

Brigade

App\Models\Brigades

> Данный класс содержит логику работы модели "Статусы бригад", которая используется для взаимодействия с таблицей БД "brigade_statuses" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'color'
];
```

Customer

App\Models\Customers

> Данный класс содержит логику работы модели "Заказчик", которая используется для взаимодействия с таблицей БД "customers" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'name',
    'actual_location',
    'working_hours_count',
    'employees_involved_count',
    'current_contract_number',
    'current_additional_agreement_number',
    'billing_period_id',
    'rate_customers',
    'rate_employees',
    'cost_transport_there_customer',
    'cost_transport_back_customer',
    'cost_transport_round_trip_customer',
    'cost_transport_there_tk',
    'cost_transport_back_tk',
    'cost_transport_round_trip_tk',
    'brigadiers_x1_rub',
    'brigadiers_x2_rub',
    'brigadiers_personal_rub',
    'legal_entity_id',
    'active',
    'contract_start_date',
    'contract_end_date',
    'comment',
    ];

protected $casts = [
    'rate_customers' => 'json',
    'rate_employees' => 'json',
    'contract_start_date' => 'date'
];
```

Список методов

workingShift

Метод реализации обратной связи один к одному с моделью WorkingShift

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```
public function workingShift(): BelongsTo
{
    return $this->belongsTo(WorkingShift::class);
}
```

workedShifts

Метод реализации связи один ко многим с моделью WorkedShift

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function workedShifts(): HasMany
{
    return $this->hasMany(WorkedShift::class);
}
```

legalEntity

Метод реализации обратной связи один к одному с моделью LegalEntity

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function legalEntity(): BelongsTo
{
    return $this->belongsTo(LegalEntity::class);
}
```

billingPeriod

Метод реализации обратной связи один к одному с моделью BillingPeriod

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function billingPeriod(): BelongsTo
{
    return $this->belongsTo(BillingPeriod::class);
}
```

customerRequisite

Метод реализации связи один к одному с моделью CustomerRequisite

Ответ

`return Illuminate\Database\Eloquent\Relations\HasOne`

```
public function customerRequisite(): HasOne
{
    return $this->hasOne(CustomerRequisite::class);
}
```

incomings

Метод реализации связи один ко многим с моделью Incoming

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function incomings(): HasMany
{
    return $this->hasMany(Incoming::class);
}
```

invoices

Метод реализации связи один ко многим с моделью Invoice

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function invoices(): HasMany
{
    return $this->hasMany(Invoice::class);
}
```

subdivisions

Метод реализации связи один к ко многим с моделью Subdivision

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```php
public function subdivisions(): HasMany
{
    return $this->hasMany(Subdivision::class);
}
```

orders

Метод реализации связи один к ко многим с моделью Order

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```php
public function orders(): HasMany
{
    return $this->hasMany(Order::class);
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```php
public function getCountRelationShip(): bool
{
    $model = $this->loadCount(['incomings', 'invoices', 'subdivisions', 'workedShifts']);
    $count = collect($model)->map(function ($item, $key) {
        return str_contains($key, '_count') ? $item : 0;
    })->sum();
    return $count === 0;
}
```

CustomerRequisite

App\Models\Customers

> Данный класс содержит логику работы модели "Реквизиты клиента", которая используется для взаимодействия с таблицей БД "customer_requisites" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'customer_id',
    'name',
    'inn',
    'kpp',
    'orgn',
    'legal_address',
    'phone',
    'email',
    'director_name',
    'director_position',
    'foundation',
    'bank_requisites',
    'contacts',
];

protected $casts = [
    'bank_requisites' => 'json',
    'contacts' => 'json'
];
```

Список методов

phone

Метод преобразует номер телефона в правильный формат

```
protected function phone(): Attribute
{
    return Attribute::make(
        get: fn (string|null $value) =>
            isset($value)
                ? (is_phone($value) ? phone_display_format($value) : $value)
                : null,
        set: fn (string|null $value) =>
            isset($value)
                ? (is_phone($value) ? phone_system_format($value) : $value)
                : null,
    );
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```
public function getCountRelationShip(): bool
{
    return true;
}
```

Document

App\Models\Documents

> Данный класс содержит логику работы модели "Документы", которая используется для взаимодействия с таблицей БД "documents" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, InteractsWithMedia

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'documentable_id',
    'documentable_type',
    'comment'
];
```

Employee

App\Models\Employees

> Данный класс содержит логику работы модели "Сотрудник", которая используется для взаимодействия с таблицей БД "employees" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, InteractsWithMedia, HasJsonRelationships

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'card_created_at',
    'user_id',
    'working_shifts',
    'full_name',
    'birthdate',
    'phones',
    'additional_phones',
    'gender',
    'passport_serial_number',
    'issued',
    'issue_date',
    'registration_address',
    'fact_address',
    'job_resource_id',
    'whence',
    'skills',
    'health_restrictions',
    'medical_book',
    'self_employed',
    'inn',
    'payment_account',
    'correspondent_account',
    'bik',
    'bank_name',
    'active',
    'comment',
    'employee_status_id',
    'employee_rating_id',
];

protected $casts = [
    'working_shifts' => 'json',
    'phones' => 'json',
    'additional_phones' => 'json',
    'birthdate' => 'date',
    'issue_date' => 'date'
];
```

Список методов

boot

Статичный метод инициализирующий обсервер EmployeeObserver в момент создания класса модели

```
public static function boot()
{
    parent::boot();
    self::observe(new EmployeeObserver());
}
```

registerMediaConversions

Метод конвертации изображения

```
public function registerMediaConversions(?Media $media = null): void
{
    $this->addMediaConversion('preview')
        ->fit(Fit::Crop, 300, 300);
}
```

phones

Метод для форматирования номера телефона

```
protected function phones(): Attribute
{
    return Attribute::make(
        get: function (string|null $value) {
            $formatedPhones = [];
            foreach (json_decode($value) as $key => $phone) {
                array_push($formatedPhones, phone_display_format($phone));
            }
            return $formatedPhones;
        }
    );
}
```

workingShifts

Метод реализации обратной json связи один ко многим с моделью WorkingShift

Ответ

`return Staudenmeir\EloquentJsonRelations\Relations\BelongsToJson`

```
public function workingShifts(): BelongsToJson
{
    return $this->belongsToJson(WorkingShift::class, 'working_shifts->ids');
}
```

user

Метод реализации обратной связи один ко многим с моделью User

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function user(): BelongsTo
{
    return $this->belongsTo(User::class);
}
```

payouts

Метод реализации связи один ко многим с моделью Payout

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function payouts(): HasMany
{
    return $this->hasMany(Payout::class);
}
```

scopeEnumSearch

Метод реализации поиск по enum

Ответ

`return Illuminate\Database\Eloquent\Builder`

```
public function scopeEnumSearch(Builder $query, string $keyword): Builder
{
    $enums = [
        'gender' => GenderEnum::asSelectArray()
    ];
    foreach ($enums as $column => $elements) {
        foreach (enumSearch($elements, $keyword) as $key => $value) {
            $query->orWhere($column, 'LIKE', "{$key}");
        }
    }
    return  $query;
}
```

brigades

Метод реализации обратной связи один ко многим с моделью Brigade

Ответ

```
return Illuminate\Database\Eloquent\Relations\HasMany
```

```
public function brigades(): HasMany
{
    return $this->hasMany(Brigade::class);
}
```

latestBrigade

Метод реализации обратной связи один к одному с моделью Brigade

Ответ

```
return Illuminate\Database\Eloquent\Relations\HasOne
```

```
public function latestBrigade(): HasOne
{
    return $this->brigades()->one()->ofMany([
        'id' => 'max'
    ], function (Builder $query) {
        $query->whereHas('brigadeStatus', function ($query) {
            $query->where('alias', BrigadeStatusEnum::CAME_OUT);
        });
    });
}
```

employeeRating

Метод реализации обратной связи один к одному с моделью EmployeeRating

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function employeeRating(): BelongsTo
{
    return $this->belongsTo(EmployeeRating::class);
}
```

employeeStatus

Метод реализации обратной связи один к одному с моделью EmployeeStatus

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function employeeStatus(): BelongsTo
{
    return $this->belongsTo(EmployeeStatus::class);
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```
public function getCountRelationShip(): bool
{
    return true;
}
```

Incoming

App\Models\Incomings

> Данный класс содержит логику работы модели "Поступления", которая используется для взаимодействия с таблицей БД "incomings" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'payment_date',
    'customer_id',
    'legal_entity_id',
    'sum',
    'comment',
];
```

Список методов

customer

Метод реализации обратной связи один ко многим с моделью Customer

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```
public function customer(): BelongsTo
{
    return $this->belongsTo(Customer::class);
}
```

legalEntity

Метод реализации связи один ко многим с моделью LegalEntity

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```php
public function legalEntity(): BelongsTo
{
    return $this->belongsTo(LegalEntity::class);
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```php
public function getCountRelationShip(): bool
{
    return true;
}
```

Invoice

App\Models\Invoices

> Данный класс содержит логику работы модели "Счета", которая используется для взаимодействия с таблицей БД "invoices" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'date',
    'customer_id',
    'invoice',
    'legal_entity_id',
    'sum',
    'paymented',
    'paymented_date',
    'comment'
];
```

Список методов

customer

Метод реализации обратной связи один ко многим с моделью Customer

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function customer(): BelongsTo
{
    return $this->belongsTo(Customer::class);
}
```

legalEntity

Метод реализации связи один ко многим с моделью LegalEntity

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```php
public function legalEntity(): BelongsTo
{
    return $this->belongsTo(LegalEntity::class);
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```php
public function getCountRelationShip(): bool
{
    return true;
}
```

LegalEntity

App\Models\LegalEntities

> Данный класс содержит логику работы модели "Юридическое лицо", которая используется для взаимодействия с таблицей БД "legal_entities" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'name',
    'inn',
    'kpp',
    'orgn',
    'legal_address',
    'phone',
    'email',
    'director_name',
    'director_position',
    'foundation',
    'bank_requisites',
    'contacts',
];

protected $casts = [
    'bank_requisites' => 'json',
    'contacts' => 'json',
];
```

Список методов

phone

Метод для форматирования номера телефона

```
protected function phone(): Attribute
{
    return Attribute::make(
        get: fn (string|null $value) =>
            isset($value)
                ? (is_phone($value) ? phone_display_format($value) : $value)
                : null,
        set: fn (string|null $value) =>
            isset($value)
                ? (is_phone($value) ? phone_system_format($value) : $value)
                : null,
    );
}
```

customers

Метод реализации связи один ко многим с моделью Customer

Ответ

```
return Illuminate\Database\Eloquent\Relations\HasMany
```

```
public function customers(): HasMany
{
    return $this->hasMany(Customer::class);
}
```

incomings

Метод реализации связи один ко многим с моделью Incoming

Ответ

```
return Illuminate\Database\Eloquent\Relations\HasMany
```

```
public function incomings(): HasMany
{
    return $this->hasMany(Incoming::class);
}
```

invoices

Метод реализации связи один ко многим с моделью Invoice

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```php
public function invoices(): HasMany
{
    return $this->hasMany(Invoice::class);
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```php
public function getCountRelationShip(): bool
{
    $model = $this->loadCount(['customers', 'incomings', 'invoices']);
    $count = collect($model)->map(function ($item, $key) {
        return str_contains($key, '_count') ? $item : 0;
    })->sum();
    return $count === 0;
}
```

LegalEntityDocument

App\Models\LegalEntities

> Данный класс содержит логику работы модели "Документы юридического лица", которая используется для взаимодействия с таблицей БД "legal_entity_documents" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, InteractsWithMedia

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'comment'
];
```

Notification

App\Models\Notifications

> Данный класс содержит логику работы модели справочника "Период выставления счетов" которая используется для взаимодействия с таблицей БД "notifications" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'user_id',
    'message',
    'read_at'
];
```

Order

App\Models\Orders

> Данный класс содержит логику работы модели "Заявка", которая используется для взаимодействия с таблицей БД "orders" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'customer_id',
    'subdivision_id',
    'working_shift_id',
    'working_hour_id',
    'order_status_id',
    'is_master_order',
    'type',
    'order_date',
    'working_shift_date',
    'place_shipment',
    'rate_employee',
    'plan',
    'fact',
    'male',
    'female',
    'number_buses_there',
    'number_buses_back',
    'number_buses_there_back',
    'cost_transportation',
    'working_shift_time',
    'working_shift_start_time',
    'working_shift_end_time',
    'deadline',
    'other_orders'
];

protected $casts = [
    'is_master_order' => 'boolean',
    'order_date' => 'date',
    'working_shift_date' => 'date',
    'deadline' => 'datetime',
    'other_orders' => 'json'
];
```

Список методов

boot

Статичный метод инициализирующий обсервер OrderObserver в момент создания класса модели

```
public static function boot()
{
    parent::boot();
    self::observe(new OrderObserver());
}
```

customer

Метод реализации обратной связи один ко многим с моделью Employee

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```
public function customer(): BelongsTo
{
    return $this->belongsTo(Customer::class);
}
```

subdivision

Метод реализации обратной связи один к одному с моделью Subdivision

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```
public function subdivision(): BelongsTo
{
    return $this->belongsTo(Subdivision::class);
}
```

workingShift

Метод реализации обратной связи один ко многим с моделью WorkingShift

Ответ

```
return Illuminate\Database\Eloquent\Relations\BelongsTo
```

```
public function workingShift(): BelongsTo
{
    return $this->belongsTo(WorkingShift::class);
}
```

workingHour

Метод реализации обратной связи один ко многим с моделью WorkingHour

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function workingHour(): BelongsTo
{
    return $this->belongsTo(WorkingHour::class);
}
```

orderStatus

Метод реализации обратной связи один ко многим с моделью OrderStatus

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function orderStatus(): BelongsTo
{
    return $this->belongsTo(OrderStatus::class);
}
```

brigades

Метод реализации связи один ко многим с моделью Brigade

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function brigades(): HasMany
{
    return $this->hasMany(Brigade::class);
}
```

workedShifts

Метод реализации связи один ко многим с моделью WorkedShift

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function workedShifts(): HasMany
{
    return $this->hasMany(WorkedShift::class);
}
```

BillingPeriod

App\Models\References

> Данный класс содержит логику работы модели справочника "Период выставления счетов" которая используется для взаимодействия с таблицей БД "billing_periods" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $hidden - скрытые поля
- $fillable - поля разрешенные для массового заполнения
- $relations - список связей модели

```
protected $hidden = [
    'password',
    'remember_token',
];

protected $fillable = [
    'name',
    'email',
    'password',
    'is_block',
    'image',
    'phones',
    'manager_lvl_id',
    'created_by'
];

protected $relations = [
    'customers'
];
```

Связи

customers

Метод реализации связи один ко многим с моделью Customer

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function customers(): HasMany
{
    return $this->hasMany(Customer::class);
}
```

EmployeeBonus

App\Models\References

> Данный класс содержит логику работы модели справочника "Премии сотрудникам" которая используется для взаимодействия с таблицей БД "employee_bonuses" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait, HasJsonRelationships Свойства
>
> - $hidden - скрытые поля
> - $fillable - поля разрешенные для массового заполнения
> - $relations - список связей модели

```
protected $fillable = [
    'working_shift_count',
    'bonus'
];

protected $casts = [
    'subdivisions' => 'json'
];

protected $relations = [
    'subdivisions'
];
```

Связи

subdivisions

Метод реализации обратной json связи один ко многим с моделью Subdivision

Ответ

`return Staudenmeir\EloquentJsonRelations\Relations\BelongsToJson`

```
public function subdivisions(): BelongsToJson
{
    return $this->belongsToJson(Subdivision::class, 'subdivisions->ids');
}
```

subdivisions

Метод реализации обратной json связи один ко многим с моделью Subdivision

Ответ

`return Staudenmeir\EloquentJsonRelations\Relations\BelongsToJson`

```
public function subdivisionModels(): BelongsToJson
{
    return $this->belongsToJson(Subdivision::class, 'subdivisions->ids');
}
```

EmployeeRating

App\Models\References

> Данный класс содержит логику работы модели справочника "Рейтинг сотрудника" которая используется для взаимодействия с таблицей БД "employee_ratings" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'absence_count',
    'days_count',
];
```

EmployeeStatus

App\Models\References

> Данный класс содержит логику работы модели справочника "Статус сотрудника" которая используется для взаимодействия с таблицей БД "employee_statuses" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения
- $relations - список связей модели

```
protected $fillable = [
    'name',
    'description',
];

protected $relations = [
    'employees'
];
```

Связи

employees

Метод реализации связи один ко многим с моделью Employee

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function employees(): HasMany
{
    return $this->hasMany(Employee::class);
}
```

JobResource

App\Models\References

> Данный класс содержит логику работы модели справочника "Откуда узнал о работе" которая используется для взаимодействия с таблицей БД "job_resources" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения
- $relations - список связей модели

```
protected $fillable = [
    'name'
];

protected $relations = [
    'employees'
];
```

Связи

employees

Метод реализации связи один ко многим с моделью Employee

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function employees(): HasMany
{
    return $this->hasMany(Employee::class);
}
```

OrderStatus

App\Models\References

> Данный класс содержит логику работы модели справочника "Статус заявки" которая используется для взаимодействия с таблицей БД "order_statuses" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения
- $relations - список связей модели

```
protected $fillable = [
    'name',
    'color'
];

protected $relations = [
    'orders'
];
```

Связи

orders

Метод реализации связи один ко многим с моделью Order

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function orders(): HasMany
{
    return $this->hasMany(Order::class);
}
```

Penalty

App\Models\References

> Данный класс содержит логику работы модели справочника "Штрафы" которая используется для взаимодействия с таблицей БД "Penalties" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'sum'
];
```

PercentageBonus

App\Models\References

> Данный класс содержит логику работы модели справочника "Проценты премии" которая используется для взаимодействия с таблицей БД "percentage_bonuses" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'sum'
];
```

Subdivision

App\Models\References

> Данный класс содержит логику работы модели справочника "Подразделения" которая используется для взаимодействия с таблицей БД "subdivisions" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait, HasJsonRelationships

Свойства

- $fillable - поля разрешенные для массового заполнения
- $relations - список связей модели

```
protected $fillable = [
    'name',
    'customer_id'
];

protected $relations = [
    'subdivisions'
];
```

Связи

customer

Метод реализации обратной связи один к одному с моделью Customer

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function customer(): BelongsTo
{
    return $this->belongsTo(Customer::class);
}
```

orders

Метод реализации связи один ко многим с моделью Order

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```php
public function orders(): HasMany
{
    return $this->hasMany(Order::class);
}
```

subdivisions

Метод реализации json связи один ко многим с моделью EmployeeBonus

Ответ

```
return Staudenmeir\EloquentJsonRelations\Relations\HasManyJson
```

```php
public function subdivisions(): HasManyJson
{
    return $this->hasManyJson(EmployeeBonus::class, 'subdivisions->ids');
}
```

Worked

App\Models\References

> Данный класс содержит логику работы модели "Отработанных смен" которая используется для взаимодействия с таблицей БД "workeds" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'color'
];
```

WorkingHour

App\Models\References

> Данный класс содержит логику работы модели справочника "Количество часов" которая используется для взаимодействия с таблицей БД "working_hours" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name'
];
```

WorkingShift

App\Models\References

> Данный класс содержит логику работы модели справочника "Смены" которая используется для взаимодействия с таблицей БД "working_shifts" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, RelationshipCheckTrait, HasJsonRelationships

Свойства

- $fillable - поля разрешенные для массового заполнения
- $relations - список связей модели

```
protected $fillable = [
    'name'
];

protected $relations = [
    'employees'
];
```

Связи

customer

Метод реализации json связи один ко многим с моделью Employee

Ответ

`return Staudenmeir\EloquentJsonRelations\Relations\HasManyJson`

```
public function employees(): HasManyJson
{
    return $this->hasManyJson(Employee::class, 'working_shifts->ids');
}
```

Schedule

App\Models\Schedules

> Данный класс содержит логику работы модели "График", которая используется для взаимодействия с таблицей БД "schedules" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory

Setting

App\Models\Settings

> Данный класс содержит логику работы модели "Настройки", которая используется для взаимодействия с таблицей БД "settings" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: InteractsWithMedia

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'key',
    'value'
];
```

Список методов

addImage

Метод для добавления картинки к модели

Ответ

`return Spatie\MediaLibrary\MediaCollections\Models\Media`

```
public function addImage(UploadedFile $photo): Media
{
    return $this->addMedia($photo)->preservingOriginal()->toMediaCollection('public');
}
```

deleteImage

Метод для удаления картинки

Ответ

`return mixed`

```
public function deleteImage(): mixed
{
    return $this->getMedia('public')->last()->delete();
}
```

SettingTable

App\Models\Settings

> Данный класс содержит логику работы модели "Настройки таблицы", которая используется для взаимодействия с таблицей БД "setting_tables" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'table',
    'column',
    'user_id'
];
```

Список методов

column

Возвращает коллекцию полученную из json. Сохраняет json, полученный из переданного массива

Ответ

`return Illuminate\Database\Eloquent\Casts\Attribute`

```
public function column(): Attribute
{
    return new Attribute(
        get: fn ($value) => new Collection(json_decode($value, true)),
        set: fn ($value) => json_encode($value),
    );
}
```

ActionHistory

App\Models\Users

> Данный класс содержит логику работы модели "История действий", которая используется для взаимодействия с таблицей БД "action_histories" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'action',
    'link',
    'section',
    'subsection',
    'user_id'
];
```

Список методов

customer

Метод реализации обратной связи один ко многим с моделью User

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function user(): BelongsTo
{
    return $this->belongsTo(User::class);
}
```

scopeEnumSearch Метод реализации поиск по секции в enum

Ответ

`return Illuminate\Database\Eloquent\Builder`

```
public function scopeEnumSearch(Builder $query, string $keyword): Builder
{
    $enums = [
        'section' => SectionEnum::asSelectArray(),
    ];
    foreach ($enums as $column => $elements) {
        foreach (enumSearch($elements, $keyword) as $key => $value) {
            $query->orWhere($column, 'LIKE', "%{$key}%");
        }
    }
    return  $query;
}
```

RoleUser

App\Models\Users

> Данный класс содержит логику работы модели "Роли пользователя", которая используется для взаимодействия с таблицей БД "role_users" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory

Свойства

- $fillable - поля разрешенные для массового заполнения

```
protected $fillable = [
    'name',
    'display_name',
    'guard_name',
    'color'
];
```

Список методов

scopeWithoutAdminScope

Метод отображает всех пользователей, кроме пользователей с ролью admin

Ответ

`return void`

```
public function scopeWithoutAdminScope(Builder $query): void
{
    $query->where('name', '!=', 'admin');
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```php
public function getCountRelationShip(): bool
{
    $model = $this->loadCount(['users']);
    $count = collect($model)->map(function ($item, $key) {
        return str_contains($key, '_count') ? $item : 0;
    })->sum();
    return $count === 0;
}
```

WorkSchedule

App\Models\WorkSchedules

> Данный класс содержит логику работы модели "График работы", которая используется для взаимодействия с таблицей БД "work_schedules" Данный класс расширяет базовый класс фреймворка Model Данный класс содержит в себе следующие трейты: HasFactory, SoftDeletes, LogsActivity

Свойства

- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $fillable = [
    'user_id',
    'start_date',
    'end_date',
    'schedule'
];

protected $casts = [
    'schedule' => AsArrayObject::class,
    'schedule->evening_phone' => 'boolean'
];
```

Список методов

getActivitylogOptions

Метод для получения истории действий пользователя с графиком работы

Ответ

`return Spatie\Activitylog\LogOptions`

```
public function getActivitylogOptions(): LogOptions
{
    return LogOptions::defaults()
        ->logOnly([
            'id',
            'user_id',
            'start_date',
            'end_date',
            'schedule'
        ])
        ->logOnlyDirty()
        ->dontSubmitEmptyLogs();
}
```

tapActivity

Метод записи активности пользователя

Ответ

`return void`

```
public function tapActivity(Activity $activity, string $eventName): void
{
    $activity->properties = $activity->properties->merge([
        'section' => SectionEnum::USERS,
        'subsection' => SubsectionEnum::WORK_SCHEDULES,
    ]);
}
```

schedule

Метод возвращает график работы

Ответ

`return Illuminate\Database\Eloquent\Casts\Attribute`

```
protected function schedule(): Attribute
{
    return Attribute::make(
        get: fn (string|null $value) => $value,
        set: fn (array $value) => json_encode($value),
    );
}
```

user

Метод реализации обратной связи один к многим с моделью User

Ответ

`return Illuminate\Database\Eloquent\Relations\BelongsTo`

```
public function user(): BelongsTo
{
    return $this->belongsTo(User::class);
}
```

getCountRelationShip

Метод реализации проверку модели на наличие связей

Ответ

`return bool`

```
public function getCountRelationShip(): bool
{
    return true;
}
```

User

App\Models

> Данный класс расширяет стандартный класс Authenticatable Данный класс имплементирует следующие интерфейсы: JWTSubject, HasMedia Данный класс содержит в себе следующие трейты: Notifiable, HasFactory, SoftDeletes, HasRoles, InteractsWithMedia, LogsActivity

Свойства

- $hidden - скрытые поля
- $fillable - поля разрешенные для массового заполнения
- $casts - поля, которые необходимо преобразовать при получении из БД

```
protected $hidden = [
    'password',
    'remember_token',
];

protected $fillable = [
    'full_name',
    'email',
    'password',
    'login',
    'phone',
    'card_created_at',
    'employment_date',
    'working_position',
    'birthdate',
    'additional_phone',
    'dismissal_date',
    'comment',
    'coefficient',
    'salary',
    'active'
];

protected $casts = [
    'email_verified_at' => 'datetime',
];
```

Методы

addAvatar

Данный метод добавляет аватар пользователю Параметры:

- UploadedFile $photo - изображение

Ответ : `return \Spatie\MediaLibrary\MediaCollections\Models\Media`

```
public function addAvatar(UploadedFile $photo): \Spatie\MediaLibrary\MediaCollections\Model
{
    return $this->addMedia($photo)->preservingOriginal()->toMediaCollection('avatar');
}
```

deleteAvatar

Данный метод удаляет аватар пользователя

```
public function deleteAvatar(): mixed
{
    return $this->getMedia('avatar')->last()->delete();
}
```

Данный метод используется для получения идентификатора пользователя для аутентификации JWT

```
public function getJWTIdentifier(): mixed
{
    return $this->getKey();
}
```

getJWTCustomClaims

```
public function getJWTCustomClaims(): array
{
    return [];
}
```

setPasswordAttribute

Данный метод используется для хеширования пароля пользователя

```
public function setPasswordAttribute(string|null $value): void
{
    if ($value !== null) {
        $this->attributes['password'] = Hash::make($value);
    }
}
```

getAllPermissionsAttribute

Данный метод используется для получения всех разрешений пользователя

```
public function getAllPermissionsAttribute(): ?\Illuminate\Support\Collection
{
    return $this->getAllPermissions()->isNotEmpty() ? $this->getAllPermissions()->pluck('na
}
```

oauthProviders

Связь One To Many с OAuthProvider

```
public function oauthProviders(): HasMany
{
    return $this->hasMany(OAuthProvider::class);
}
```

settingTables

Метод реализации связи один ко многим с моделью SettingTable

Ответ

```
return Illuminate\Database\Eloquent\Relations\HasMany
```

```
public function settingTables(): HasMany
{
    return $this->hasMany(SettingTable::class);
}
```

workSchedule

Метод реализации связи один ко многим с моделью WorkSchedule

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function workSchedule(): HasOne
{
    return $this->hasOne(WorkSchedule::class);
}
```

employees

Метод реализации связи один ко многим с моделью Employee

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function employees(): HasMany
{
    return $this->hasMany(Employee::class);
}
```

brigades

Метод реализации связи один ко многим с моделью Brigade

Ответ

`return Illuminate\Database\Eloquent\Relations\HasMany`

```
public function brigades(): HasMany
{
    return $this->hasMany(Brigade::class);
}
```

role

Метод возвращает роль пользователя

Ответ

`return Illuminate\Database\Eloquent\Model|null`

```
public function role(): Model|null
{
    return $this->roles->first();
}
```

getCountRelationShip

```php
public function getCountRelationShip(): bool
{
    $model = $this->loadCount(['employees']);
    $count = collect($model)->map(function ($item, $key) {
        return str_contains($key, '_count') ? $item : 0;
    })->sum();
    return $count === 0;
}
```

BrigadeObserver

App\Observers\Brigades

> Данный наблюдатель служит для отслеживания событий модели бригады

Методы класса

created

Данный метод вызывается при создании модели Brigade, и обновляет статус связанной заявки
Параметры:

- Brigade $brigade - сериализованная модель которую необходимо обновить

```
public function created(Brigade $brigade): void
{
    $order = $brigade->order;
    if ($order->plan === $order->brigades->count()) {
        $order->order_status_id = OrderStatus::where('alias', OrderStatusEnum::DONE)->first
        $order->saveQuietly();
    }
}
```

EmployeeObserver

App\Observers\Employees

> Данный наблюдатель служит для отслеживания событий модели сотрудники

Методы класса

updating

Данный метод вызывается при обновлении модели Employee и форматирует номера телефона
Параметры:

- Employee $employee - сериализованная модель которую необходимо обновить

```php
public function updating(Employee $employee): void
{
    $formatedPhones = [];
    foreach ($employee->phones as $key => $phone) {
        array_push($formatedPhones, phone_system_format($phone));
    }
    $employee->phones = $formatedPhones;
}
```

OrderObserver

App\Observers\Orders

> Данный наблюдатель служит для отслеживания событий модели заявки

Методы класса

created

Данный метод вызывается при создании модели Order, и обновляет её статус Параметры:

- Order $order - сериализованная модель которую необходимо обновить

```
public function created(Order $order): void
{
    $deadline = Carbon::parse($order->deadline);
    $workingShiftEnd = Carbon::parse(
        $order->working_shift_date->format('Y-m-d') . ' ' . $order->working_shift_end_time
    );
    $brigadeCount = $order->brigades->count();
    $status = '';

    if ($workingShiftEnd->diffInHours(Carbon::now()) < 1) {
        $status = OrderStatusEnum::COMPLETED;
    } elseif ($order->plan === $brigadeCount) {
        $status = OrderStatusEnum::DONE;
    } elseif (Carbon::now()->diffInHours($deadline) <= 1 && $order->plan !== $brigadeCount)
        $status = OrderStatusEnum::OVERDUE;
    } elseif (Carbon::now()->diffInHours($deadline) <= 24 && $order->plan !== $brigadeCount
        $status = OrderStatusEnum::IN_WORK;
    } elseif (Carbon::now()->diffInHours($deadline) > 24 && $order->plan !== $brigadeCount)
        $status = OrderStatusEnum::PLANNED;
    }

    $order->order_status_id = OrderStatus::where('alias', $status)->first()->id;
    $order->saveQuietly();
}
```

BillingPeriodPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- BillingPeriod $billingPeriod - период

Ответ : `return bool`

```
public function update(User $user, BillingPeriod $billingPeriod): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- BillingPeriod $billingPeriod - период

Ответ : `return bool`

```
public function delete(User $user, BillingPeriod $billingPeriod): bool
{
    return $user->can('references_destroy');
}
```

BrigadePolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('brigades_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('brigades_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Brigade $brigade - бригада

Ответ : `return bool`

```
public function update(User $user, Brigade $brigade): bool
{
    return $user->can('brigades_create');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Brigade $brigade - бригада

Ответ : `return bool`

```
public function delete(User $user, Brigade $brigade): bool
{
    return $user->can('brigades_destroy');
}
```

CustomerPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('customers_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function update(User $user, Customer $customer): bool
{
    return $user->can('customers_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- $customer - заказчик

Ответ : `return bool`

```
public function update(User $user, Customer $customer): bool
{
    return $user->can('customers_create');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- $customer - заказчик

Ответ : `return bool`

```php
public function delete(User $user, Customer $customer): bool
{
    return $user->can('customers_destroy');
}
```

CustomerRequisitePolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('customers_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('customers_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- $customerRequisite - реквизиты заказчика

Ответ : `return bool`

```
public function update(User $user, CustomerRequisite $customerRequisite): bool
{
    return $user->can('customers_create');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- $customerRequisite - реквизиты заказчика

Ответ : `return bool`

```php
public function delete(User $user, CustomerRequisite $customerRequisite): bool
{
    return $user->can('customers_edit');
}
```

DocumentPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return true;
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('files_uploadAndDelete');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- $document - документ

Ответ : `return bool`

```
public function update(User $user, Document $document): bool
{
    return $user->can('files_uploadAndDelete');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- $document - документ

Ответ : `return bool`

```
public function delete(User $user, Document $document): bool
{
    return $user->can('files_uploadAndDelete');
}
```

EmployeeBonusPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- $employeeBonus - бонусы сотрудников

Ответ : `return bool`

```
public function update(User $user, EmployeeBonus $employeeBonus): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- $employeeBonus - бонусы сотрудников

Ответ : `return bool`

```
public function delete(User $user, EmployeeBonus $employeeBonus): bool
{
    return $user->can('references_destroy');
}
```

EmployeePolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('employees_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('employees_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- $employee - сотрудник

Ответ : `return bool`

```
public function update(User $user, Employee $employee): bool
{
    return $user->can('employees_create');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- $employee - сотрудник

Ответ : `return bool`

```
public function delete(User $user, Employee $employee): bool
{
    return $user->can('employees_destroy');
}
```

EmployeeStatusPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- EmployeeStatus $employeeStatus - статус сотрудника

Ответ : `return bool`

```
public function update(User $user, EmployeeStatus $employeeStatus): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- EmployeeStatus $employeeStatus - статус сотрудника

Ответ : `return bool`

```
public function delete(User $user, EmployeeStatus $employeeStatus): bool
{
    return $user->can('references_destroy');
}
```

FactPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('salaries_factRead');
}
```

IncomingPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('customers_incomingsRead');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('customers_incomingsEdit');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Incoming $incoming - поступление

Ответ : `return bool`

```
public function update(User $user, Incoming $incoming): bool
{
    return $user->can('customers_incomingsEdit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Incoming $incoming - поступление

Ответ : `return bool`

```
public function delete(User $user, Incoming $incoming): bool
{
    return $user->can('customers_incomingsEdit');
}
```

InvoicePolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('customers_invoicesRead');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('customers_invoicesEdit');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Invoice $invoice - счёт

Ответ : `return bool`

```
public function update(User $user, Invoice $invoice): bool
{
    return $user->can('customers_invoicesEdit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Invoice $invoice - счёт

Ответ : `return bool`

```
public function delete(User $user, Invoice $invoice): bool
{
    return $user->can('customers_invoicesEdit');
}
```

JobResourcePolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- JobResource $jobResource - модель справочника "откуда узнал о работе"

Ответ : `return bool`

```
public function update(User $user, JobResource $jobResource): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- JobResource $jobResource - модель справочника "откуда узнал о работе"

Ответ : `return bool`

```
public function delete(User $user, JobResource $jobResource): bool
{
    return $user->can('references_destroy');
}
```

LegalEntityDocumentPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('legalEntities_documentsRead');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('legalEntities_documentsRead');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- LegalEntityDocument $legalEntityDocument - модель документа юридического лица

Ответ : `return bool`

```
public function update(User $user, LegalEntityDocument $legalEntityDocument): bool
{
    return $user->can('legalEntities_documentsEditAndDelete');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- LegalEntityDocument $legalEntityDocument - модель документа юридического лица

Ответ : `return bool`

```
public function delete(User $user, LegalEntityDocument $legalEntityDocument): bool
{
    return $user->can('legalEntities_documentsEditAndDelete');
}
```

LegalEntityPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('legalEntities_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('legalEntities_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- LegalEntity $legalEntity - модель юридического лица

Ответ : `return bool`

```
public function update(User $user, LegalEntity $legalEntity): bool
{
    return $user->can('legalEntities_editAndDelete');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- LegalEntity $legalEntity - модель юридического лица

Ответ : `return bool`

```
public function delete(User $user, LegalEntity $legalEntity): bool
{
    return $user->can('legalEntities_editAndDelete');
}
```

OrderPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('orders_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('orders_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Order $order - модель заявки

Ответ : `return bool`

```
public function update(User $user, Order $order): bool
{
    return $user->can('orders_create');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Order $order - модель заявки

Ответ : `return bool`

```
public function delete(User $user, Order $order): bool
{
    return $user->can('orders_destroy');
}
```

PayoutPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('salaries_payoutRead');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('salaries_payoutCreate');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Payout $payout - модель выплаты

Ответ : `return bool`

```
public function update(User $user, Payout $payout): bool
{
    return $user->can('salaries_payoutEdit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Payout $payout - модель выплаты

Ответ : `return bool`

```
public function delete(User $user, Payout $payout): bool
{
    return $user->can('salaries_payoutEdit');
}
```

PenaltyPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Penalty $penalty - модель штрафов

Ответ : `return bool`

```
public function update(User $user, Penalty $penalty): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Penalty $penalty - модель штрафов

Ответ : `return bool`

```
public function delete(User $user, Penalty $penalty): bool
{
    return $user->can('references_destroy');
}
```

PercentageBonusPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- PercentageBonus $percentageBonus - модель процент премии

Ответ : `return bool`

```
public function update(User $user, PercentageBonus $percentageBonus): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- PercentageBonus $percentageBonus - модель процент премии

Ответ : `return bool`

```
public function delete(User $user, PercentageBonus $percentageBonus): bool
{
    return $user->can('references_destroy');
}
```

PlanPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('salaries_planRead');
}
```

SettingPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('settings_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('settings_create');
}
```

StatementPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('salaries_statementsRead');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('salaries_statementsEdit');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Statement $statement - модель ведомости

Ответ : `return bool`

```
public function update(User $user, Statement $statement): bool
{
    return $user->can('salaries_statementsEdit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Statement $statement - модель ведомости

Ответ : `return bool`

```
public function delete(User $user, Statement $statement): bool
{
    return $user->can('salaries_statementsEdit');
}
```

SubdivisionPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- Subdivision $subdivision - модель подразделения

Ответ : `return bool`

```
public function update(User $user, Subdivision $subdivision): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- Subdivision $subdivision - модель подразделения

Ответ : `return bool`

```
public function delete(User $user, Subdivision $subdivision): bool
{
    return $user->can('references_destroy');
}
```

WorkingHourPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- WorkingHour $workingHour - модель количества часов

Ответ : `return bool`

```
public function update(User $user, WorkingHour $workingHour): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- WorkingHour $workingHour - модель количества часов

Ответ : `return bool`

```
public function delete(User $user, WorkingHour $workingHour): bool
{
    return $user->can('references_destroy');
}
```

WorkingShiftPolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('references_read');
}
```

create

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function create(User $user): bool
{
    return $user->can('references_create');
}
```

update

Параметры:

- User $user - авторизованный пользователь
- WorkingShift $workingShift - модель смены

Ответ : `return bool`

```
public function update(User $user, WorkingShift $workingShift): bool
{
    return $user->can('references_edit');
}
```

delete

Параметры:

- User $user - авторизованный пользователь
- WorkingShift $workingShift - модель смены

Ответ : `return bool`

```
public function delete(User $user, WorkingShift $workingShift): bool
{
    return $user->can('references_destroy');
}
```

WorkSchedulePolicy

App\Policies

Методы

viewAny

Параметры:

- User $user - авторизованный пользователь

Ответ : `return bool`

```
public function viewAny(User $user): bool
{
    return $user->can('users_chartEdit');
}
```

PasswordRule

Правило валидации приходящего в запросе пароля на корректность

Список методов класса

passes

Данный метод производит проверку на со *Ответ* ствие пароля заданному шаблону: "В пароле должна быть хотя бы одна цифра, одна буква в нижнем регистре и одна буква в верхнем регистре"

```
public function passes($attribute, $value)
{
  return preg_match('/((?=.*\\d)(?=.*[a-za-я])(?=.*[A-ZA-Я]).{6,20})/', $value);
}
```

message

Данное поле возвращает разъяснение ошибки валидации пользователю системы в случае, если проверка закончилась неудачей

```
public function message()
{
    return __('validation.password_regex');
}
```

InnValidationRule

Правило валидации приходящего в запросе ИНН на корректность

```
public function validate(string $attribute, mixed $value, Closure $fail): void
{
    if (!is_numeric($value)) {
        $fail('validation.inn')->translate();
    }
    switch (strlen($value)) {
        case 12:
            self::isInnIp($value) ? self::isInnIp($value) : $fail('validation.inn')->transl
            break;
        case 10:
            self::isInnJur($value) ? self::isInnJur($value) : $fail('validation.inn')->tran
            break;
        default:
            $fail('validation.inn')->translate();
            break;
    }
}
```

isInnIp

Данный метод выполняет проверку корректности ИНН для индивидуальных предпринимателей

```
public static function isInnIp(string $value): bool
{
        $s = (7 * (float)$value[0] +
                2 * (float)$value[1] +
                4 * (float)$value[2] +
                10 * (float)$value[3] +
                3 * (float)$value[4] +
                5 * (float)$value[5] +
                9 * (float)$value[6] +
                4 * (float)$value[7] +
                6 * (float)$value[8] +
                8 * (float)$value[9]) % 11;
        if ($s == 10) {
            $s = 0;
        }
        $s2 = (3 * (float)$value[0] +
                7 * (float)$value[1] +
                2 * (float)$value[2] +
                4 * (float)$value[3] +
                10 * (float)$value[4] +
                3 * (float)$value[5] +
                5 * (float)$value[6] +
                9 * (float)$value[7] +
                4 * (float)$value[8] +
                6 * (float)$value[9] +
                8 * (float)$value[10]) % 11;
        if ($s2 == 10) {
            $s2 = 0;
        }
        if ($s != $value[10] || $s2 != $value[11]) {
            return false;
        }

        return true;
}
```

isInnJur

Данный метод выполняет проверку корректности ИНН для юридических лиц

```
public static function isInnJur(string $value): bool
{
        $s = (2 * (float)$value[0] +
                4 * (float)$value[1] +
                10 * (float)$value[2] +
                3 * (float)$value[3] +
                5 * (float)$value[4] +
                9 * (float)$value[5] +
                4 * (float)$value[6] +
                6 * (float)$value[7] +
                8 * (float)$value[8]) % 11;
        if ($s == 10) {
            $s = 0;
        }
        if ($s != $value[9]) {
            return false;
        }

        return true;
}
```

IsOneWeekRule

Правило валидации приходящего в запросе дня недели на корректность

Список методов класса

validate

Данный метод производит проверку на со *Ответ* ствие значения дню недели.

```
public function validate(string $attribute, mixed $value, Closure $fail): void
{
    if (
        !Carbon::create($value[0])->startOfWeek()->eq(Carbon::create($value[0])) ||
        !Carbon::create($value[1])->endOfWeek()->eq(Carbon::create($value[1])->endOfDay())
        !(Carbon::create($value[0])->diffInDays(Carbon::create($value[1])) == 6)
    ) {
        $fail('validation.is_one_week')->translate();
    }
}
```

PhoneNumberRule

Правило валидации приходящего в запросе номера телефона на корректность

Список методов класса

validate

Данный метод производит проверку на со *Ответ* ствие значения корректному номеру телефона.

```
public function validate(string $attribute, mixed $value, Closure $fail): void
{
    if (!(bool)preg_match('/^((8|\+7)[\- ]?)?(\(?\d{3}\)?[\- ]?)?[\d\- ]{10}/', $value)) {
        $fail(__('validation.phone_number'))->translate();
    }
}
```

BrigadeService

App\Services\Brigades

> Данный сервис отвечает за основную логику работы с модулем "Бригады" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Brigade $model,
    protected string $className = __CLASS__
) {
}
```

getDatatableFilters

Данный метод возвращает фильтры для дататейбла

Ответ

`return array`

```
public function getDatatableFilters(): array
{
    return [
        'brigadeStatuses' => JsonResource::collection(
            BrigadeStatus::select('id', 'name', 'alias', 'color')
                ->orderBy('id')
                ->get()
        )
    ];
}
```

customers

Данный метод возвращает отфильтрованный список заказчиков для страницы бригад Принимает BrigadeFilterRequestDTO $request

Ответ

`return Illuminate\Http\Resources\Json\JsonResource`

```
public function customers(BrigadeFilterRequestDTO $request): JsonResource
{
    return JsonResource::collection(
        Customer::select('id', 'name')
            ->whereHas('orders', function ($query) use ($request) {
                $query->when(isset($request->working_shift_date), function ($query) use ($r
                    $query->where('working_shift_date', $request->working_shift_date);
                });
            })
            ->get()
    );
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице "Бригад" Для формирования *Ответ* а используется DatatableRequestDTO и Customer $customer

```
public function datatable(DatatableRequestDTO $request, Customer $customer): JsonResponse
{
    return DataTables::eloquent($this->model->with([
        'subdivision' => function ($query) {
            $query->select('id', 'name');
        },
        'employee' => function ($query) {
            $query->select('id', 'full_name', 'comment');
        },
        'brigadeStatus' => function ($query) {
            $query->select('id', 'alias');
        }
    ])
        ->whereHas('order', function ($query) use ($customer) {
            $query->where('customer_id', $customer->id);
        })
        ->leftJoin('orders', 'orders.id', 'brigades.order_id')
        ->select('brigades.*'))
        ->setTransformer(function ($model) {
            return BrigadeDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

additionalDatatable

Данный метод возвращает JSON объект, который используется при построении итоговых таблицы на индексной странице "Бригад" Для формирования *Ответ* а используется DatatableRequestDTO и Customer $customer

```
public function additionalDatatable(DatatableRequestDTO $request, Customer $customer): Json
{
    return DataTables::eloquent(
        Subdivision::leftJoin('brigades', 'subdivisions.id', 'brigades.subdivision_id')
            ->leftJoin('orders', 'orders.id', 'brigades.order_id')
            ->where('orders.customer_id', $customer->id)
            ->groupBy(['subdivisions.id'])
            ->select(
                'subdivisions.name as name',
                DB::raw('SUM(orders.plan) as plan'),
                DB::raw('SUM(CASE WHEN brigade_status_id = 1 THEN 1 ELSE 0 END) as not_came
                DB::raw('SUM(CASE WHEN brigade_status_id = 2 THEN 1 ELSE 0 END) as came_out
                DB::raw('SUM(CASE WHEN brigade_status_id = 3 THEN 1 ELSE 0 END) as day_off'
                DB::raw('SUM(CASE WHEN brigade_status_id = 4 THEN 1 ELSE 0 END) as not_conf
                DB::raw('COUNT(*) as fact')
            )
    )
        ->setTransformer(function ($model) {
            return BrigadeAdditionalDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод служит для фильтрации данных, перед их отдачей методом datatable

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->customer)) {
        $query->where('orders.customer_id', $request->customer['id']);
    }
    if (isset($request->working_shift_date)) {
        $query->where('working_shift_date', $request->working_shift_date);
    }
}
```

formData

Данный метод служит для предоставления данных для дальнейшего использования

```
protected function formData(): array
{
    return [
        'working_shifts' => JsonResource::collection(WorkingShift::select('id', 'name', 'al
        'employees' => JsonResource::collection(Employee::select('id', 'full_name as name')
        'customers' => BrigadeCustomerResource::collection(
            Customer::select('id', 'name')
                ->with('subdivisions')
                ->whereHas('orders', function ($query) {
                    $query->where('order_date', Carbon::now());
                })
                ->get()
        )
    ];
}
```

createData

Данный метод служит для предоставления данных на страницу создания бригады

```
public function createData(): array
{
    return [...$this->formData()];
}
```

store

Данный метод реализует логику сохранения бригад В качестве параметра метод принимает BrigadeRequestDTO $request

```
public function store(BrigadeRequestDTO $request): void
{
    Order::where('customer_id', $request->customer_id)
        ->where('order_date', Carbon::now())
        ->first()
        ->brigades()
        ->create(array_merge(
            $request->toArray(),
            [
                'brigade_status_id' => BrigadeStatus::where('alias', BrigadeStatusEnum::NOT
                'user_id' => Auth::user()->id
            ]
        ));
}
```

storeEmployee

Данный метод реализует логику добавления сотрудников в бригады В качестве параметра метод принимает BrigadeEmployeeRequestDTO $request

```
public function storeEmployee(BrigadeEmployeeRequestDTO $request): JsonResource
{
    $employee = Employee::create($request->toArray());
    $employee->workingShifts()->sync($request->working_shifts ?? [])->save();
    return JsonResource::make($employee);
}
```

update

Данный метод реализует логику обновления бригад В качестве первого аргумента метод принимает BrigadeRequestDTO $request В качестве второго аргумента метод принимает модель Brigade $brigade

```
public function update(BrigadeRequestDTO $request, Brigade $brigade): void
{
    $order = Order::where('customer_id', $request->customer_id)
        ->where('order_date', Carbon::now())
        ->first();

    $brigade->update([
        'order_id' => $order->id,
        'subdivision_id' => $request->subdivision_id
    ]);
}
```

updateStatus

Данный метод реализует логику обновления статусов бригад В качестве первого аргумента метод принимает BrigadeStatusRequestDTO $request В качестве второго аргумента метод принимает модель Brigade $brigade

```
public function updateStatus(BrigadeStatusRequestDTO $request, Brigade $brigade): void
{
    $brigade->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Brigade $brigade

Ответ

`return bool`

```
public function destroy(Brigade $brigade): bool
{
    return $brigade->getCountRelationShip() ? $brigade->delete() : false;
}
```

WorkShiftService

App\Services\Brigades

> Данный сервис отвечает за основную логику работы с модулем "Бригады" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Order $model,
    protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

getDatatableFilters

Данный метод возвращает фильтры для дататейбла

Ответ

`return array`

```
public function getDatatableFilters(): array
{
    return [
        'customers' => JsonResource::collection(Customer::select('id', 'name')->get())
    ];
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице "Бригад" Для формирования *Ответ* а используется DatatableRequestDTO

```php
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
        'brigades' => function ($query) use ($request) {
            $query->select('id', 'employee_id', 'order_id')
                ->where('employee_id', $request->employee);
        },
        'customer' => function ($query) {
            $query->select('id', 'name');
        },
        'subdivision' => function ($query) {
            $query->select('id', 'name');
        },
        'workingShift' => function ($query) {
            $query->select('id', 'name');
        },
        'workingHour' => function ($query) {
            $query->select('id', 'name');
        },
        'orderStatus' => function ($query) {
            $query->select('id', 'alias');
        }
    ])->withCount('brigades'))
        ->setTransformer(function ($model) {
            return WorkShiftDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->withQuery('counts', function (Builder $query) {
            return [
                'plan' => $query->sum('plan'),
                'fact' => $query->get()->sum('brigades_count'),
                'male' => $query->sum('male'),
                'female' => $query->sum('female')
            ];
        })
        ->toJson();
}
```

applyFilters

Данный метод служит для фильтрации данных, перед их отдачей методом datatable

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->customer)) {
        $query->where('customer_id', $request->customer);
    }
    if (isset($request->working_shift_date)) {
        $query->where('working_shift_date', $request->working_shift_date);
    }
    if (isset($request->employee_id)) {
        $query->whereHas('brigades', function ($query) use ($request) {
            $query->where('employee_id', $request->employee_id);
        });
    }
}
```

update

Данный метод реализует логику обновления периода бригад В качестве первого аргумента метод принимает WorkShiftRequestDTO $request В качестве второго аргумента метод принимает модель Order $order

```
public function update(WorkShiftRequestDTO $request, Order $order): void
{
    $order->brigades()
        ->create(array_merge(
            $request->toArray(),
            [
                'brigade_status_id' => BrigadeStatus::where('alias', BrigadeStatusEnum::NOT
                'user_id' => Auth::user()->id
            ]
        ));
}
```

CustomerRequisiteService

App\Services\Customers

> Данный сервис отвечает за основную логику работы с модулем "Заказчики" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected CustomerRequisite $customerRequisite,
    protected string $className = __CLASS__
) {
    $this->model = $customerRequisite->query();
}
```

getIndexData

Данный метод возвращает список столбцов для дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице "Заказчики" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return CustomerRequisiteDTResource::make($model)->resolve();
        })
        ->filterColumn('phone', function ($query, $keyword) use ($request) {
            if (preg_match("/^\d+$/", phone_system_format($request->search['value']))) {
                $query->where('phone', 'LIKE', "%" . phone_system_format($request->search['
            }
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

createData

Данный метод служит для предоставления данных на страницу создания бригады

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод служит для предоставления данных на страницу редактирования реквизитов заказчика

```
public function editData(CustomerRequisite $customerRequisite): array
{
    return [
        ...$this->formData(),
        'element' => CustomerRequisiteFormResource::make($customerRequisite)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения реквизитов заказчика В качестве параметра метод принимает CustomerRequisiteRequestDTO $request

```
public function store(CustomerRequisiteRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления реквизитов заказчика В качестве первого аргумента метод принимает CustomerRequisiteRequestDTO $request В качестве второго аргумента метод принимает модель CustomerRequisite $customerRequisite

```
public function update(CustomerRequisiteRequestDTO $request, CustomerRequisite $customerReq
{
    $customerRequisite->update($request->toArray());
}
```

CustomerService

App\Services\Customers

> Данный сервис отвечает за основную логику работы с модулем "Заказчики" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Customer $customer,
    protected string $className = __CLASS__
) {
    $this->model = $customer->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице "Реквезитов заказчика" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
            'customerRequisite',
            'legalEntity' => function ($query) {
                $query->select('id', 'name');
            },
            'billingPeriod' => function ($query) {
                $query->select('id', 'name');
            }
        ])->select('customers.*'))
        ->setTransformer(function ($model) {
            return CustomerDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [
        'billing_periods' => JsonResource::collection(BillingPeriod::select('id', 'name')->
        'working_hours' => JsonResource::collection(WorkingHour::select('id', 'name')->get(
        'legal_entities' => JsonResource::collection(LegalEntity::select('id', 'name')->get
    ];
}
```

createData

Данный метод отдает информацию для страницы создания заказчика

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования заказчика

Ответ

`return array`

```
public function editData(Customer $customer): array
{
    return [
        ...$this->formData(),
        'element' => new CustomerFormResource($customer->load([
            'customerRequisite'
        ]))
    ];
}
```

store

Данный метод отдает информацию для страницы редактирования заказчика

Ответ

`return array`

```
public function store(CustomerRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает CustomerRequestDTO $request В качестве второго аргумента метод принимает модель Customer $customer

```
public function update(CustomerRequestDTO $request, Customer $customer): void
{
    $customer->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Customer $customer

Ответ

`return bool`

```
public function destroy(Customer $customer): bool
{
    return $customer->getCountRelationShip() ? $customer->delete() : false;
}
```

search

Данный метод реализует поиск заказчика по названию В качестве параметра метод принимает BaseSearchRequestDTO $request

Ответ

`return bool`

```
public function search(BaseSearchRequestDTO $request): LengthAwarePaginator
{
    $keywords = prepare_keyword($request->q ?? '');

    $data = $this->model
    ->select(['id', 'name'])
    ->when(count($keywords), function ($query) use ($keywords) {
        foreach ($keywords as $keyword) {
            $query->where(function ($query) use ($keyword) {
                $query->where('name', 'like', "%$keyword%");
            });
        }
    })
    ->orderBy('name')
    ->paginate(15);

    return $data;
}
```

generateExcel

Метод генерации excel документа Параметры

- DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

`return bool`

```
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->reject(fn ($item) => $item['data'] === 'actions')
        ->pluck('text', 'name')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'customers.xlsx'
    );
}
```

generateExcelAll

Метод генерации excel документа Параметры

- DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

`return bool`

```
public function generateExcelAll(DataTableRequestDTO $request): BinaryFileResponse
{
    $customer = new Customer();
    $columns = ['id' => 'ID'];
    foreach ($customer->getFillable() as $key => $column) {
        $columns[$column] = __("attributes.customers.$column");
    }
    $arrayQueryDT = json_decode($this->datatableExportAll($request)->content(), true);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'customers_all.xlsx'
    );
}
```

generateExcelRequisites

Метод генерации excel документа Параметры

- DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

`return bool`

```php
public function generateExcelRequisites(DataTableRequestDTO $request): BinaryFileResponse
{
    $customer = new CustomerRequisite();
    $columns = ['id' => 'ID'];
    foreach ($customer->getFillable() as $key => $column) {
        $columns[$column] = __("attributes.customer_requisites.$column");
    }
    $arrayQueryDT = json_decode($this->datatableExportRequisites($request)->content(), true
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'customers_requisites.xlsx'
    );
}
```

EmployeeService

App\Services\Dashboard\Employees

> Данный сервис отвечает за основную логику работы с модулем "Сотрудники" в дашбордах
> Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected EmployeeStatus $model,
    protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает данные для индексной странице Параметры DashboardFilterRequestDTO $request - данные прошедшие валидацию

Ответ

```
return array
```

```
public function getIndexData(DashboardFilterRequestDTO $request): JsonResource
{
    return EmployeeResource::collection($this->model
        ->withCount('employees')
        ->when(isset($request->customer), function ($query) use ($request) {
            $query->whereHas('employees.brigades.order', function ($query) use ($request) {
                $query->where('customer_id', $request->customer['id']);
            });
        })
        ->when(isset($request->period[0]), function ($query) use ($request) {
            $query->whereHas('employees.brigades.order', function ($query) use ($request) {
                $query->where('working_shift_date', '>=', Carbon::parse($request->period[0]
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('working_shift_date', '<=', Carbon::parse($requ
                            ->endOfDay());
                    });
            });
        })
        ->get());
}
```

ManagerActivityService

App\Services\Dashboard\ManagerActivity

> Данный сервис отвечает за основную логику работы с модулем "Активность менеджеров" в дашбордах Класс содержит в себе следующие трейты: DataTableStoreColumnTrait
>
> ## Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected User $model,
    protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает данные для индексной странице Параметры DashboardFilterRequestDTO $request - данные прошедшие валидацию

Ответ

```
return array
```

```php
public function getIndexData(DashboardFilterRequestDTO $request): JsonResource
{
    return ManagerActivityResource::collection($this->model
        ->withCount(['brigades', 'brigades as brigades_count_came_out' => function ($query)
            $query->whereHas('brigadeStatus', function ($query) {
                $query->where('alias', BrigadeStatusEnum::CAME_OUT);
            });
        }])
        ->when(isset($request->user), function ($query) use ($request) {
            $query->whereHas('brigades', function ($query) use ($request) {
                $query->where('user_id', $request->user['id']);
            });
        })
        ->when(isset($request->customer), function ($query) use ($request) {
            $query->whereHas('brigades.order', function ($query) use ($request) {
                $query->where('customer_id', $request->customer['id']);
            });
        })
        ->when(isset($request->period[0]), function ($query) use ($request) {
            $query->whereHas('brigades.order', function ($query) use ($request) {
                $query->where('working_shift_date', '>=', Carbon::parse($request->period[0]
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('working_shift_date', '<=', Carbon::parse($requ
                            ->endOfDay());
                    });
            });
        })
        ->get());
}
```

OrderService

App\Services\Dashboard\Orders

> Данный сервис отвечает за основную логику работы с модулем "Заявки" в дашбордах
> Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected OrderStatus $model,
    protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает данные для индексной странице Параметры

- DashboardFilterRequestDTO $request - данные прошедшие валидацию

Ответ

```
return array
```

```
public function getIndexData(DashboardFilterRequestDTO $request): JsonResource
{
    return OrderResource::collection($this->model
        ->where('alias', '!=', OrderStatusEnum::COMPLETED)
        ->withCount('orders')
        ->when(isset($request->customer), function ($query) use ($request) {
            $query->whereHas('orders', function ($query) use ($request) {
                $query->where('customer_id', $request->customer['id']);
            });
        })
        ->when(isset($request->period[0]), function ($query) use ($request) {
            $query->whereHas('orders', function ($query) use ($request) {
                $query->where('working_shift_date', '>=', Carbon::parse($request->period[0]
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('working_shift_date', '<=', Carbon::parse($requ
                            ->endOfDay());
                    });
            });
        })
        ->get());
}
```

PaymentService

App\Services\Dashboard\Payments

> Данный сервис отвечает за основную логику работы с модулем "Оплата" в дашбордах
> Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поля класса

```
public function __construct(
    protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает данные для индексной странице Параметры

- DashboardFilterRequestDTO $request - данные прошедшие валидацию

Ответ

`return array`

```
public function getIndexData(DashboardFilterRequestDTO $request): array
{
    return [
        'incomings' => PaymentResource::collection(Incoming::select('payment_date as date',
            ->when(isset($request->customer), function ($query) use ($request) {
                $query->where('customer_id', $request->customer['id']);
            })
            ->when(isset($request->legal_entity), function ($query) use ($request) {
                $query->where('legal_entity_id', $request->legal_entity['id']);
            })
            ->when(isset($request->period[0]), function ($query) use ($request) {
                $query->where('payment_date', '>=', Carbon::parse($request->period[0])->sta
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('payment_date', '<=', Carbon::parse($request->p
                    });
            })->get()),
        'payouts' => PaymentResource::collection(Payout::when(
            isset($request->customer),
            function ($query) use ($request) {
                $query->whereHas('employee.brigades.order', function ($query) use ($request
                    $query->where('customer_id', $request->customer['id']);
                });
            }
        )
            ->when(isset($request->legal_entity), function ($query) use ($request) {
                $query->whereHas('employee.brigades.order.customer', function ($query) use
                    $query->where('legal_entity_id', $request->legal_entity['id']);
                });
            })
            ->when(isset($request->period[0]), function ($query) use ($request) {
                $query->where('date', '>=', Carbon::parse($request->period[0])->startOfDay(
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('date', '<=', Carbon::parse($request->period[1]
                    });
            })->get()),
        'accruals' => PaymentResource::collection(Accrual::when(
            isset($request->customer),
            function ($query) use ($request) {
                $query->whereHas('employee.brigades.order', function ($query) use ($request
                    $query->where('customer_id', $request->customer['id']);
                });
            }
        )
            ->when(isset($request->legal_entity), function ($query) use ($request) {
```

```
                $query->whereHas('employee.brigades.order.customer', function ($query) use
                    $query->where('legal_entity_id', $request->legal_entity['id']);
                });
            })
            ->when(isset($request->period[0]), function ($query) use ($request) {
                $query->where('date', '>=', Carbon::parse($request->period[0])->startOfDay(
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('date', '<=', Carbon::parse($request->period[1]
                    });
            })->get())
    ];
}
```

SeasonalActivityService

App\Services\Dashboard\SeasonalActivity

> Данный сервис отвечает за основную логику работы с модулем "Сезонная активность" в дашбордах Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Order $model,protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает данные для индексной странице Параметры DashboardFilterRequestDTO $request - данные прошедшие валидацию

Ответ

`return array`

```
public function getIndexData(DashboardFilterRequestDTO $request): JsonResource
{
    return SeasonalActivityResource::collection($this->getQuery($request)->get());
}
```

generateExcelRequisites

Данный метод возвращает нужный запрос в зависимости от переданных данных Параметры DashboardFilterRequestDTO $request - данные прошедшие валидацию

Ответ

`return array`

```
private function getQuery(DashboardFilterRequestDTO $request): Builder
{
    $query = $this->model->query();
    if (isset($request->data_type) && $request->data_type === 'payment') {
        $query = Payout::select(DB::raw('SUM(COALESCE(sum, 0)) as total'))
            ->when(isset($request->type), function ($query) use ($request) {
                $query->addSelect(DB::raw("date_part('$request->type', date) as date"))
                    ->groupBy(DB::raw("date_part('$request->type', date)"));
            })
            ->when(isset($request->customer), function ($query) use ($request) {
                $query->whereHas('employee.brigades.order', function ($query) use ($request
                    $query->where('customer_id', $request->customer['id']);
                });
            })
            ->when(isset($request->period[0]), function ($query) use ($request) {
                $query->where('date', '>=', Carbon::parse($request->period[0])->startOfDay(
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('date', '<=', Carbon::parse($request->period[1]
                    });
            });
    } elseif (isset($request->data_type) && $request->data_type === 'employee') {
        $query->select(
            DB::raw('SUM(COALESCE(plan, 0)) as plan_total'),
            DB::raw('COUNT(brigades.id) as fact_total')
        )
            ->leftJoin('brigades', 'brigades.order_id', 'orders.id')
            ->whereNull('brigades.deleted_at')
            ->when(isset($request->type), function ($query) use ($request) {
                $query->addSelect(DB::raw("date_part('$request->type', order_date) as date"
                    ->groupBy(DB::raw("date_part('$request->type', order_date)"));
            })
            ->when(isset($request->customer), function ($query) use ($request) {
                $query->where('customer_id', $request->customer['id']);
            })
            ->when(isset($request->period[0]), function ($query) use ($request) {
                $query->where('working_shift_date', '>=', Carbon::parse($request->period[0]
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('working_shift_date', '<=', Carbon::parse($requ
                            ->endOfDay());
                    });
            });
    } elseif (isset($request->data_type) && $request->data_type === 'order') {
        $query->select(DB::raw('COUNT(*) as total'))
            ->when(isset($request->type), function ($query) use ($request) {
```

```
                $query->addSelect(DB::raw("date_part('$request->type', order_date) as date"
                    ->groupBy(DB::raw("date_part('$request->type', order_date)"));
            })
            ->when(isset($request->customer), function ($query) use ($request) {
                $query->where('customer_id', $request->customer['id']);
            })
            ->when(isset($request->period[0]), function ($query) use ($request) {
                $query->where('working_shift_date', '>=', Carbon::parse($request->period[0]
                    ->when(isset($request->period[1]), function ($item) use ($request) {
                        return $item->where('working_shift_date', '<=', Carbon::parse($requ
                            ->endOfDay());
                    });
            });
    }

    return $query;
}
```

DocumentService

App\Services\Documents

> Данный сервис отвечает за основную логику работы с модулем "Документы"

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Document $model
) {
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->select('documents.*'))
        ->setTransformer(function ($model) {
            return DocumentDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный применяет фильтры к запросу для datatable

Ответ

return void

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->period[0])) {
        $query->where('created_at', '>=', Carbon::parse($request->period[0])->startOfDay())
            ->when(isset($request->period[1]), function ($item) use ($request) {
                return $item->where('created_at', '<=', Carbon::parse($request->period[1])-
            });
    }
    if (isset($request->documentable_id)) {
        $query->where('documentable_id', $request->documentable_id);
    }
    if (isset($request->documentable_type)) {
        $query->where('documentable_type', $request->documentable_type);
    }
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(Document $document): array
{
    return [
        ...$this->formData(),
        'element' => DocumentFormResource::make($document)->resolve()
    ];
}
```

store

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function store(DocumentRequestDTO $request): void
{
    $model = $this->model->create($request->toArray());
    $this->saveFile($request, $model);
}
```

update

Данный метод реализует логику обновления модели В качестве первого аргумента метод принимает DocumentRequestDTO $request В качестве второго аргумента метод принимает модель Document $document

```
public function update(DocumentRequestDTO $request, Document $document): void
{
    $document->update($request->toArray());
    $file = $document->getMedia('file')->first();
    if (isset($request->file) && isset($file)) {
        $file->delete();
    }
    $this->saveFile($request, $document);
}
```

generateExcelRequisites

Данный метод сохраняет файл В качестве параметра метод принимает DocumentRequestDTO $request В качестве параметра метод принимает модель Document $document

Ответ

return void

```
public function saveFile(DocumentRequestDTO $request, Document $document): void
{
    if (isset($request->file)) {
        $document->addMedia($request->file)->preservingOriginal()->toMediaCollection('file'
    }
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Document $document

Ответ

return bool

```
public function destroy(Document $document): bool
{
    return $document->delete();
}
```

EmployeeService

App\Services\Employees

> Данный сервис отвечает за основную логику работы с модулем "Сотрудники" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait
>
> ## Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Employee $model,protected string $className = __CLASS__
) {
}
```

getDatatableFilters

Данный метод возвращает фильтры для datatable

Ответ

`return array`

```
public function getDatatableFilters(): array
{
    return [
        'genders' => enumDataFormat(GenderEnum::asSelectArray()),
        'ratings' => JsonResource::collection(EmployeeRating::select('id', 'name')->get()),
        'working_shifts' => JsonResource::collection(WorkingShift::select('id', 'name')->ge
        'employee_statuses' => JsonResource::collection(EmployeeStatus::select('id', 'name'
    ];
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```php
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```php
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
            'workingShifts' => function ($query) {
                $query->select('id', 'name');
            }
        ])->select('employees.*'))
        ->setTransformer(function ($model) {
            return EmployeeDTResource::make($model)->resolve();
        })
        ->with('counter', function () {
            return $this->model->count();
        })
        ->filterColumn('phones', function ($query, $keyword) use ($request) {
            if (preg_match("/^\d+$/", phone_system_format($request->search['value']))) {
                $query->where('phones', 'LIKE', "%" . phone_system_format($request->search[
                    ->orWhere('phones', 'LIKE', "%" . $request->search['value'] . "%");
            }
        })
        ->filterColumn('gender', function ($query, $keyword) {
            $query->enumSearch($keyword);
        })
        ->filterColumn('birthdate', function ($query, $keyword) use ($request) {
            foreach (['Y-m-d', 'd.m.Y'] as $format) {
                try {
                    $date = Carbon::createFromFormat($format, $request->search['value']);
                    if ($date && $date->format($format) === $request->search['value']) {
                        $query->where('birthdate', 'LIKE', $date->format('Y-m-d'));
                    }
                } catch (\Throwable $th) {
                }
            }
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->working_shift)) {
        $query->whereHas('workingShifts', function ($query) use ($request) {
            $query->whereId($request->working_shift);
        });
    }
    if (isset($request->gender)) {
        $query->where('gender', $request->gender);
    }
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [
        'genders' => enumDataFormat(GenderEnum::asSelectArray()),
        'working_shifts' => JsonResource::collection(WorkingShift::select('id', 'name', 'al
        'job_resources' => JsonResource::collection(JobResource::select('id', 'name', 'alia
    ];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(Employee $employee): array
{
    return [
        ...$this->formData(),
        'element' => new EmployeeFormResource($employee->load(
            ['user' => function ($query) {
                $query->select('id', 'full_name');
            }]
        ))
    ];
}
```

store

Данный метод создаёт новую модель

Ответ

`return array`

```
public function store(EmployeeRequestDTO $request): void
{
    $this->saveData($request, $this->model->create($request->toArray()));
}
```

show Данный метод отдаёт данные на страницу просмотра

Ответ

`return array`

```
public function show(Employee $employee): JsonResource
{
    return JsonResource::make($employee);
}
```

saveData Данный метод сохраняет дополнительные данные модели в системе В качестве параметра метод принимает EmployeeRequestDTO $request В качестве параметра метод принимает модель Employee $employee

Ответ

return void

```
public function saveData(EmployeeRequestDTO $request, Employee $employee): void
{
    $employee->workingShifts()->sync($request->working_shifts ?? [])->save();
    if (isset($request->avatar)) {
        $employee->addMedia($request->avatar)->preservingOriginal()->toMediaCollection('ava
    }
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает EmployeeRequestDTO $request В качестве второго аргумента метод принимает модель Employee $employee

```
public function update(EmployeeRequestDTO $request, Employee $employee): void
{
    $employee->update($request->toArray());
    $avatar = $employee->getMedia('avatar')->first();
    if ((isset($request->avatar) && isset($avatar)) || isset($request->delete_avatar)) {
        $avatar->delete();
    }
    $this->saveData($request, $employee);
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Employee $employee

Ответ

return bool

```
public function destroy(Employee $employee): bool
{
    return $employee->getCountRelationShip() ? $employee->delete() : false;
}
```

search

Данный метод реализует поиск по названию В качестве параметра метод принимает BaseSearchRequestDTO $request

Ответ

return Illuminate\Pagination\LengthAwarePaginator

```
public function search(BaseSearchRequestDTO $request): LengthAwarePaginator
{
    $keywords = prepare_keyword($request->q ?? '');

    return $this->model
        ->select(['id', 'full_name'])
        ->when(count($keywords), function ($query) use ($keywords) {
            foreach ($keywords as $keyword) {
                $query->where(function ($query) use ($keyword) {
                    $query->where('full_name', 'ILIKE', "%$keyword%");
                });
            }
        })
        ->orderBy('full_name')
        ->paginate(15);
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

return bool

```php
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->reject(fn ($item) => $item['data'] === 'actions')
        ->reject(fn ($item) => $item['data'] === 'edit_working_shift')
        ->pluck('text', 'name')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'employees.xlsx'
    );
}
```

GeneralReportService

App\Services\GeneralReports

> Данный сервис отвечает за основную логику работы с модулем "Отчёт общий" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Order $order,protected string $className = __CLASS__
) {
    $this->model = $order->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
        'customer' => function ($query) {
            $query->select('id', 'name');
        },
        'workingShift' => function ($query) {
            $query->select('id', 'name');
        }
    ])
        ->leftJoin('customers', 'customers.id', 'orders.customer_id')
        ->leftJoin('working_hours', 'working_hours.id', 'orders.working_hour_id')
        ->leftJoin(
            DB::raw('LATERAL jsonb_array_elements(rate_customers::jsonb) AS jsonb_element')
            function ($join) {
                $join->on(DB::raw("(jsonb_element->>'working_hour_id')::int"), 'working_hou
            }
        )
        ->select([
            'orders.*',
            DB::raw('(select COALESCE(sum(accrual_sum), 0) from summaries
                where orders.customer_id = summaries.customer_id and working_shift_date = d
                as employee_payment'),
            DB::raw("(select count(*) from worked_shifts left join workeds on worked_shifts
                where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', 'part
                as brigades_count"),
            DB::raw("COALESCE((jsonb_element->>'sum')::int, 0) / working_hours.name AS rate_
            'working_hours.name as working_shift_duration',
            DB::raw(
                '(cost_transport_there_customer * number_buses_there +
                cost_transport_back_customer * number_buses_back +
                cost_transport_round_trip_customer * number_buses_there_back) as transfer'
            ),
            DB::raw("(working_hours.name *
                (select count(*) from worked_shifts left join workeds on worked_shifts.work
                    where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', '
                as total_working_hours"),
            DB::raw("(working_hours.name *
                (select count(*) from worked_shifts left join workeds on worked_shifts.work
                    where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', '
                (COALESCE((jsonb_element->>'sum')::int, 0) / working_hours.name)) as cost_w
            DB::raw("(working_hours.name *
                (select count(*) from worked_shifts left join workeds on worked_shifts.work
                    where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', '
```

```
                (COALESCE((jsonb_element->>'sum')::int, 0) / working_hours.name) +
                (cost_transport_there_customer * number_buses_there +
                cost_transport_back_customer * number_buses_back +
                cost_transport_round_trip_customer * number_buses_there_back)) as total"),
            DB::raw("(working_hours.name *
                (select count(*) from worked_shifts left join workeds on worked_shifts.work
                    where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', '
                (COALESCE((jsonb_element->>'sum')::int, 0) / working_hours.name) +
                (cost_transport_there_customer * number_buses_there +
                cost_transport_back_customer * number_buses_back +
                cost_transport_round_trip_customer * number_buses_there_back) -
                (select COALESCE(sum(accrual_sum), 0) from summaries
                    where orders.customer_id = summaries.customer_id and working_shift_date
                cost_transportation) as income"),
        ]))
        ->filterColumn('rate_customer', function ($query, $keyword) {
            $query->where(
                DB::raw("(COALESCE((jsonb_element->>'sum')::int, 0) / working_hours.name)")
                'LIKE',
                "%{$keyword}%"
            );
        })
        ->filterColumn('working_shift_duration', function ($query, $keyword) {
            $query->where(DB::raw('(working_hours.name)::text'), 'LIKE', "%{$keyword}%");
        })
        ->filterColumn('brigades_count', function ($query, $keyword) {
            $query->where(
                DB::raw("(select count(*) from
                worked_shifts left join workeds on worked_shifts.worked_id = workeds.id
            where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', 'partly')
                'LIKE',
                "%{$keyword}%"
            );
        })
        ->filterColumn('total_working_hours', function ($query, $keyword) {
            $query->where(
                DB::raw("(working_hours.name *
                (select count(*) from worked_shifts left join workeds on worked_shifts.work
                    where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', '
                'LIKE',
                "%{$keyword}%"
            );
        })
        ->filterColumn('cost_work', function ($query, $keyword) {
            $query->where(DB::raw("(working_hours.name *
```

```
                (select count(*) from worked_shifts left join workeds on worked_shifts.work
                    where orders.id = worked_shifts.order_id and workeds.alias IN ('yes', '
                (COALESCE((jsonb_element->>'sum')::int, 0) / working_hours.name))"), 'LIKE'
        })
        ->filterColumn('transfer', function ($query, $keyword) {
            $query->where(DB::raw(
                '(cost_transport_there_customer * number_buses_there +
                cost_transport_back_customer * number_buses_back +
                cost_transport_round_trip_customer * number_buses_there_back)'
            ), 'LIKE', "%{$keyword}%");
        })
        ->setTransformer(function ($model) {
            return GeneralReportDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->period[0])) {
        $query->where('working_shift_date', '>=', Carbon::parse($request->period[0])->start
            ->when(isset($request->period[1]), function ($item) use ($request) {
                return $item->where('working_shift_date', '<=', Carbon::parse($request->per
            });
    }
    if (isset($request->customer)) {
        $query->where('customer_id', $request->customer['id']);
    }
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

return bool

```
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->reject(fn($item) => $item['data'] === 'actions')
        ->pluck('text', 'name')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'general-reports.xlsx'
    );
}
```

IncomingService

App\Services\Incomings

> Данный сервис отвечает за основную логику работы с модулем "Поступления" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Incoming $incoming,
    protected string $className = __CLASS__
) {
    $this->model = $incoming->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
            'customer' => function ($query) {
                $query->select('id', 'name');
            },
            'legalEntity' => function ($query) {
                $query->select('id', 'name');
            }
        ])->select('incomings.*'))
        ->setTransformer(function ($model) {
            return IncomingDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->period[0])) {
        $query->where('payment_date', '>=', Carbon::parse($request->period[0])->startOfDay(
            ->when(isset($request->period[1]), function ($item) use ($request) {
                return $item->where('payment_date', '<=', Carbon::parse($request->period[1]
            });
    }
    if (isset($request->customer)) {
        $query->where('customer_id', $request->customer['id']);
    }
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [
        'legal_entities' => JsonResource::collection(LegalEntity::select('id', 'name')->get
    ];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(Incoming $incoming): array
{
    return [
        ...$this->formData(),
        'element' => new IncomingFormResource($incoming->load([
            'customerRequisite'
        ]))
    ];
}
```

store

Данный метод создаёт новую модель

Ответ

`return array`

```
public function store(IncomingRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает IncomingRequestDTO $request В качестве второго аргумента метод принимает модель Incoming $incoming

```
public function update(IncomingRequestDTO $request, Incoming $incoming): void
{
    $incoming->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Incoming $incoming

Ответ

return bool

```
public function destroy(Incoming $incoming): bool
{
    return $incoming->getCountRelationShip() ? $incoming->delete() : false;
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

return bool

```php
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->reject(fn ($item) => $item['data'] === 'actions')
        ->pluck('text', 'name')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'incomings.xlsx'
    );
}
```

InvoiceService

App\Services\Invoices

> Данный сервис отвечает за основную логику работы с модулем "Счета" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Invoice $invoice,
    protected string $className = __CLASS__
) {
    $this->model = $invoice->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
            'customer' => function ($query) {
                $query->select('id', 'name');
            },
            'legalEntity' => function ($query) {
                $query->select('id', 'name');
            }
        ])->select('invoices.*'))
        ->setTransformer(function ($model) {
            return InvoiceDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->period[0])) {
        $query->where('date', '>=', Carbon::parse($request->period[0])->startOfDay())
            ->when(isset($request->period[1]), function ($item) use ($request) {
                return $item->where('date', '<=', Carbon::parse($request->period[1])->endOf
            });
    }
    if (isset($request->customer)) {
        $query->where('customer_id', $request->customer['id']);
    }
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [
        'legal_entities' => JsonResource::collection(LegalEntity::select('id', 'name')->get
    ];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(Invoice $invoice): array
{
    return [
        ...$this->formData(),
        'element' => new InvoiceFormResource($invoice->load([
            'customerRequisite'
        ]))
    ];
}
```

store

Данный метод создаёт новую модель

Ответ

`return array`

```
public function store(InvoiceRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает InvoiceRequestDTO $request В качестве второго аргумента метод принимает модель Invoice $invoice

```
public function update(InvoiceRequestDTO $request, Invoice $invoice): void
{
    $invoice->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Invoice $invoice

Ответ

return bool

```
public function destroy(Invoice $invoice): bool
{
    return $invoice->getCountRelationShip() ? $invoice->delete() : false;
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

return bool

```php
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->reject(fn ($item) => $item['data'] === 'actions')
        ->pluck('text', 'name')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'invoices.xlsx'
    );
}
```

LegalEntityService

App\Services\LegalEntities

> Данный сервис отвечает за основную логику работы с модулем "Юридические лица" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected LegalEntity $legalEntity,
    protected string $className = __CLASS__
) {
    $this->model = $legalEntity->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return LegalEntityDTResource::make($model)->resolve();
        })
        ->filterColumn('phone', function ($query, $keyword) use ($request) {
            if (preg_match("/^\d+$/", phone_system_format($request->search['value']))) {
                $query->where('phone', 'LIKE', "%" . phone_system_format($request->search['
            }
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(LegalEntity $legalEntity): array
{
    return [
        ...$this->formData(),
        'element' => LegalEntityFormResource::make($legalEntity)->resolve()
    ];
}
```

store

Данный метод создаёт новую модель

Ответ

`return array`

```
public function store(LegalEntityRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает LegalEntityRequestDTO $request В качестве второго аргумента метод принимает модель LegalEntity $legalEntity

```
public function update(LegalEntityRequestDTO $request, LegalEntity $legalEntity): void
{
    $legalEntity->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель LegalEntity $legalEntity

Ответ

return bool

```
public function destroy(LegalEntity $legalEntity): bool
{
    return $legalEntity->getCountRelationShip() ? $legalEntity->delete() : false;
}
```

search

Данный метод реализует поиск по названию В качестве параметра метод принимает BaseSearchRequestDTO $request

Ответ

return bool

```
public function search(BaseSearchRequestDTO $request): LengthAwarePaginator
{
    $keywords = prepare_keyword($request->q ?? '');

    return $this->model
        ->select(['id', 'name'])
        ->when(count($keywords), function ($query) use ($keywords) {
            foreach ($keywords as $keyword) {
                $query->where(function ($query) use ($keyword) {
                    $query->where('name', 'like', "%$keyword%");
                });
            }
        })
        ->orderBy('name')
        ->paginate(15);
}
```

MainService

App\Services\Main

> Данный сервис отвечает за основную логику работы с модулем "Главная страница" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait
>
> ## Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Order $model,
    protected string $className = __CLASS__
) {
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(MainFilterRequestDTO $request): JsonResource
{
    return MainResource::collection($this->model->with([
        'orderStatus' => function ($query) {
            $query->select('id', 'alias');
        },
        'customer' => function ($query) {
            $query->select('id', 'name');
        },
        'subdivision' => function ($query) {
            $query->select('id', 'name');
        },
        'workingShift' => function ($query) {
            $query->select('id', 'name');
        }
    ])
        ->withCount('brigades')
        ->whereHas('orderStatus', function ($query) {
            $query->where('alias', '!=', OrderStatusEnum::COMPLETED);
        })
        ->when(isset($request->customer_id), function ($query) use ($request) {
            $query->whereId($request->customer_id);
        })
        ->when(isset($request->working_shift_date), function ($query) use ($request) {
            $query->where('working_shift_date', $request->working_shift_date);
        })
        ->get());
}
```

getDatatableFilters

Данный метод возвращает данные для фильтров в дататейбле

Ответ

`return array`

```
public function getDatatableFilters(): array
{
    return [
        'customers' => JsonResource::collection(Customer::select('id', 'name')->get())
    ];
}
```

NotificationService

App\Services\Notifications

> Данный сервис отвечает за основную логику работы с модулем "Уведомления"

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Notification $model
) {
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->select('notifications.*'))
        ->setTransformer(function ($model) {
            return NotificationDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры на запрос для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->period[0])) {
        $query->where('created_at', '>=', Carbon::parse($request->period[0])->startOfDay())
            ->when(isset($request->period[1]), function ($item) use ($request) {
                return $item->where('created_at', '<=', Carbon::parse($request->period[1])-
            });
    }
}
```

OrderEmployeeService

App\Services\Orders

> Данный сервис отвечает за основную логику работы с модулем "Заявки"

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Employee $model,
    protected string $className = __CLASS__
) {
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent(Brigade::with([
        'employee.workingShifts' => function ($query) {
            $query->select('id', 'name');
        },
        'employee.employeeRating' => function ($query) {
            $query->select('id', 'name');
        },
        'employee.employeeStatus' => function ($query) {
            $query->select('id', 'name');
        },
        'subdivision'
    ])->select('brigades.*')
        ->where('order_id', $request->order_id))
        ->setTransformer(function ($model) {
            return OrderEmployeeDTResource::make($model)->resolve();
        })
        ->toJson();
}
```

search

Данный осуществляет поиск по названию

Ответ

`return array`

```
public function search(BaseSearchRequestDTO $request): mixed
{
    $keywords = prepare_keyword($request->q ?? '');
    return OrderEmployeeFormResource::collection($this->model
        ->with([
            'employeeRating' => function ($query) {
                $query->select('id', 'name');
            },
            'employeeStatus' => function ($query) {
                $query->select('id', 'name');
            },
            'latestBrigade.latestOrder.customer' => function ($query) {
                $query->select('id', 'name');
            }
        ])->select('employees.*')
        ->when(count($keywords), function ($query) use ($keywords) {
            foreach ($keywords as $keyword) {
                $query->where(function ($query) use ($keyword) {
                    $query->where('full_name', 'ILIKE', "%$keyword%");
                });
            }
        })
        ->orderBy('full_name')
        ->paginate(15))
        ->response()
        ->getData(true);
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(Order $order): array
{
    return [
        'customer_working_shift_date' => OrderEmployeeDateResource::collection(
            Order::select('working_shift_date', 'customer_id', 'working_shift_id')
                ->distinct()
                ->where('customer_id', $order->customer_id)
                ->where('working_shift_id', $order->working_shift_id)
                ->get()
        )
    ];
}
```

store

Данный метод создаёт новую модель

Ответ

`return array`

```
public function store(OrderEmployeeStoreRequestDTO $request): void
{
    $brigades = Brigade::where('subdivision_id', $request->subdivision_id)
        ->whereHas('order', function ($query) use ($request) {
            $query->where('working_shift_date', $request->working_shift_date)
                ->where('customer_id', $request->customer_id)
                ->where('working_shift_id', $request->working_shift_id);
        })
        ->get()
        ->map(function ($item) {
            $brigade = $item->replicate();
            $brigade->brigade_status_id = BrigadeStatus::where('alias', BrigadeStatusEnum::
                ->first()
                ->id;
            $brigade->user_id = Auth::user()->id;

            return $brigade;
        });

    $order = Order::find($request->order_id);
    $order->brigades()->delete();
    $order->brigades()->saveMany($brigades);
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает OrderEmployeeUpdateRequestDTO $request В качестве второго аргумента метод принимает модель Order $order

```
public function update(OrderEmployeeUpdateRequestDTO $request, Order $order): void
{
    $order->brigades()->create(
        array_merge(
            $request->toArray(),
            [
                'brigade_status_id' => BrigadeStatus::where('alias', BrigadeStatusEnum::NOT
                'user_id' => Auth::user()->id
            ]
        )
    );
}
```

OrderRecommendationEmployeeService

App\Services\Orders

> Данный сервис отвечает за основную логику работы с модулем "Заявки"

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Employee $model,
    protected string $className = __CLASS__
) {
}
```

getDatatableFilters

Данный метод возвращает данные для фильтров datatable

```
public function getDatatableFilters(): array
{
    return [
        'genders' => enumDataFormat(GenderEnum::asSelectArray()),
        'employee_statuses' => JsonResource::collection(EmployeeStatus::select('id', 'name'
        'employee_ratings' => JsonResource::collection(EmployeeRating::select('id', 'name')
        'subdivisions' => JsonResource::collection(Subdivision::select('id', 'name')->get()
    ];
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
        'workingShifts' => function ($query) {
            $query->select('id', 'name');
        },
        'employeeRating' => function ($query) {
            $query->select('id', 'name');
        },
        'employeeStatus' => function ($query) {
            $query->select('id', 'name');
        },
        'brigades.order.customer' => function ($query) {
            $query->select('id', 'name');
        }
    ])->select('employees.*')
        ->whereHas('brigades', function ($query) use ($request) {
            $query->where('order_id', '!=', $request->order_id);
        }))
        ->setTransformer(function ($model) {
            return OrderEmployeeRecommendationDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```php
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->gender)) {
        $query->where('gender', $request->gender);
    }
    if (isset($request->employee_status)) {
        $query->whereHas('employeeStatus', function ($query) use ($request) {
            $query->whereId($request->employee_status);
        });
    }
    if (isset($request->employee_rating)) {
        $query->whereHas('employeeRating', function ($query) use ($request) {
            $query->whereId($request->employee_rating);
        });
    }
    if (isset($request->last_object)) {
        $query->whereHas('brigades.latestOrder.customer', function ($query) use ($request)
            $query->whereId($request->last_object);
        });
    }
}
```

OrderService

App\Services\Orders

> Данный сервис отвечает за основную логику работы с модулем "Заявки" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Order $model,
    protected string $className = __CLASS__
) {
}
```

BillingPeriodService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected BillingPeriod $billingPeriod
) {
    $this->model = $billingPeriod->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Период выставления счетов" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return ReferenceDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

editData

Данный метод отдает информацию для страницы редактирования периода выставления счетов Параметры BillingPeriod $billingPeriod - Модель периода выставления счетов

Ответ

`return array`

```
public function editData(BillingPeriod $billingPeriod): array
{
    return [
        ...$this->formData(),
        'element' => ReferenceFormResource::make($billingPeriod)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения периода выставления счетов В качестве параметра метод принимает ReferenceRequestDTO $request

```
public function store(ReferenceRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления периода выставления счетов В качестве первого аргумента метод принимает ReferenceRequestDTO $request В качестве второго аргумента метод принимает модель BillingPeriod $billingPeriod

```
public function update(ReferenceRequestDTO $request, BillingPeriod $billingPeriod): void
{
    $billingPeriod->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель BillingPeriod $billingPeriod

Ответ

return bool

```
public function destroy(BillingPeriod $billingPeriod): bool
{
    return $billingPeriod->getCountRelationShip() ? $billingPeriod->delete() : false;
}
```

EmployeeBonusService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected EmployeeBonus $model
) {
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Бонусы сотрудникам" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with(['subdivisionModels' => function ($query
                return $query->select(['id', 'name']);
    }]))
    ->setTransformer(function ($model) {
        return EmployeeBonusDTResource::make($model)->resolve();
    })
    ->filter(function (Builder $query) use ($request) {
        $this->applyFilters($query, $request);
    }, true)
    ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [
        'subdivisions' => JsonResource::collection(Subdivision::select('id', 'name')->get()
    ];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования периода выставления счетов Параметры EmployeeBonus $employeeBonus - Модель периода выставления счетов

Ответ

`return array`

```
public function editData(EmployeeBonus $employeeBonus): array
{
    return [
        ...$this->formData(),
        'element' => EmployeeBonusFormResource::make($employeeBonus)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения бонусов сотрудникам В качестве параметра метод принимает EmployeeBonusRequestDTO $request

```
public function store(EmployeeBonusRequestDTO $request): void
{
    $employeeBonus = $this->model->create($request->toArray());
    $this->saveSubdivisions($request, $employeeBonus);
}
```

update

Данный метод реализует логику обновления бонусов сотрудникам В качестве первого аргумента метод принимает EmployeeBonusRequestDTO $request В качестве второго аргумента метод принимает модель EmployeeBonus $employeeBonus

```
public function update(EmployeeBonusRequestDTO $request, EmployeeBonus $employeeBonus): voi
{
    $employeeBonus->update($request->toArray());
    $this->saveSubdivisions($request, $employeeBonus);
}
```

saveSubdivisions

Данный метод реализует логику бонусов к подразделениям В качестве первого аргумента метод принимает EmployeeBonusRequestDTO $request В качестве второго аргумента метод принимает модель EmployeeBonus $employeeBonus

```
protected function saveSubdivisions(EmployeeBonusRequestDTO $request, EmployeeBonus $employ
{
    $employeeBonus->subdivisions()->sync($request->subdivisions ?? [])->save();
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель EmployeeBonus $employeeBonus

Ответ

return bool

```
public function destroy(EmployeeBonus $employeeBonus): bool
{
    return $employeeBonus->getCountRelationShip() ? $employeeBonus->delete() : false;
}
```

EmployeeRatingService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected EmployeeRating $model
) {
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Рейтинг сотрудника" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return EmployeeRatingDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметры EmployeeRating $employeeRating - Модель рейтинга сотрудников

Ответ

`return array`

```
public function editData(EmployeeRating $employeeRating): array
{
    return [
        ...$this->formData(),
        'element' => EmployeeRatingFormResource::make($employeeRating)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения рейтинга В качестве параметра метод принимает EmployeeRatingRequestDTO $request

```php
public function store(EmployeeRatingRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления рейтинга В качестве первого аргумента метод принимает EmployeeRatingRequestDTO $request В качестве второго аргумента метод принимает модель EmployeeRating $employeeRating

```php
public function update(EmployeeRatingRequestDTO $request, EmployeeRating $employeeRating):
{
    $employeeRating->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель EmployeeRating $employeeRating

Ответ

return bool

```php
public function destroy(EmployeeRating $employeeRating): bool
{
    return $employeeRating->getCountRelationShip() ? $employeeRating->delete() : false;
}
```

EmployeeStatusService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected EmployeeStatus $employeeStatus
) {
    $this->model = $employeeStatus->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Статус сотрудника" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return EmployeeStatusDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр EmployeeStatus $employeeStatus - Модель статуса сотрудников

Ответ

`return array`

```
public function editData(EmployeeStatus $employeeStatus): array
{
    return [
        ...$this->formData(),
        'element' => EmployeeStatusFormResource::make($employeeStatus)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения статуса сотрудника В качестве параметра метод принимает EmployeeStatusRequestDTO $request

```php
public function store(EmployeeStatusRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления статуса сотрудника В качестве первого аргумента метод принимает EmployeeStatusRequestDTO $request В качестве второго аргумента метод принимает модель EmployeeStatus $employeeStatus

```php
public function update(EmployeeStatusRequestDTO $request, EmployeeStatus $employeeStatus):
{
    $employeeStatus->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель EmployeeStatus $employeeStatus

Ответ

return bool

```php
public function destroy(EmployeeStatus $employeeStatus): bool
{
    return $employeeStatus->getCountRelationShip() ? $employeeStatus->delete() : false;
}
```

JobResourceService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected JobResource $jobResource
) {
    $this->model = $jobResource->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Откуда узнал о работе" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return ReferenceDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр JobResource $jobResource - Модель откуда узнал о работе

Ответ

`return array`

```
public function editData(JobResource $jobResource): array
{
    return [
        ...$this->formData(),
        'element' => ReferenceFormResource::make($jobResource)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения модели В качестве параметра метод принимает ReferenceRequestDTO $request

```
public function store(ReferenceRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления модели В качестве первого аргумента метод принимает ReferenceRequestDTO $request В качестве второго аргумента метод принимает модель JobResource $jobResource

```
public function update(ReferenceRequestDTO $request, JobResource $jobResource): void
{
    $jobResource->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель EmployeeStatus $employeeStatus

Ответ

return bool

```
public function destroy(EmployeeStatus $employeeStatus): bool
{
    return $employeeStatus->getCountRelationShip() ? $employeeStatus->delete() : false;
}
```

PenaltyService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Penalty $penalty
) {
    $this->model = $penalty->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Штрафы" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return PenaltyDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр Penalty $penalty - Модель штрафы

Ответ

`return array`

```
public function editData(Penalty $penalty): array
{
    return [
        ...$this->formData(),
        'element' => PenaltyFormResource::make($penalty)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения штрафа В качестве параметра метод принимает ReferenceRequestDTO $request

```
public function store(ReferenceRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления штрафа В качестве первого аргумента метод принимает PenaltyRequestDTO $request В качестве второго аргумента метод принимает модель Penalty $penalty

```
public function update(PenaltyRequestDTO $request, Penalty $penalty): void
{
    $penalty->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Penalty $penalty

Ответ

return bool

```
public function destroy(Penalty $penalty): bool
{
    return $penalty->getCountRelationShip() ? $penalty->delete() : false;
}
```

PercentageBonusService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected PercentageBonus $percentageBonus
) {
    $this->model = $percentageBonus->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Процент премии" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return PercentageBonusDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр PercentageBonus $percentageBonus - Модель штрафы

Ответ

`return array`

```
public function editData(PercentageBonus $percentageBonus): array
{
    return [
        ...$this->formData(),
        'element' => PercentageBonusFormResource::make($percentageBonus)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения бонуса В качестве параметра метод принимает PercentageBonusRequestDTO $request

```php
public function store(PercentageBonusRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления бонуса В качестве первого аргумента метод принимает PercentageBonusRequestDTO $request В качестве второго аргумента метод принимает модель PercentageBonus $percentageBonus

```php
public function update(PercentageBonusRequestDTO $request, PercentageBonus $percentageBonus
{
    $percentageBonus->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель PercentageBonus $percentageBonus

Ответ

return bool

```php
public function destroy(PercentageBonus $percentageBonus): bool
{
    return $percentageBonus->getCountRelationShip() ? $percentageBonus->delete() : false;
}
```

SubdivisionService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Subdivision $subdivision
) {
    $this->model = $subdivision->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Подразделения" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
            'customer' => function ($query) {
                $query->select('id', 'name');
            }
        ]))
        ->setTransformer(function ($model) {
            return SubdivisionDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр Subdivision $subdivisio - Модель подразделения

Ответ

`return array`

```
public function editData(Subdivision $subdivision): array
{
    return [
        ...$this->formData(),
        'element' => SubdivisionFormResource::make($subdivision->load([
            'customer' => function ($query) {
                $query->select('id', 'name');
            }
        ]))->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения подразделения В качестве параметра метод принимает SubdivisionRequestDTO $request

```
public function store(SubdivisionRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления подразделения В качестве первого аргумента метод принимает SubdivisionRequestDTO $request В качестве второго аргумента метод принимает модель Subdivision $subdivision

```
public function update(SubdivisionRequestDTO $request, Subdivision $subdivision): void
{
    $subdivision->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель Subdivision $subdivision

Ответ

return bool

```
public function destroy(Subdivision $subdivision): bool
{
    return $subdivision->getCountRelationShip() ? $subdivision->delete() : false;
}
```

WorkingHourService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected WorkingHour $workingHour)
{
    $this->model = $workingHour->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Количество часов" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return ReferenceDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр WorkingHour $workingHour - Модель количества рабочих часов

Ответ

`return array`

```
public function editData(WorkingHour $workingHour): array
{
    return [
        ...$this->formData(),
        'element' => ReferenceFormResource::make($workingHour)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения количества часов В качестве параметра метод принимает WorkingHourRequestDTO $request

```
public function store(WorkingHourRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления количества часов В качестве первого аргумента метод принимает WorkingHourRequestDTO $request В качестве второго аргумента метод принимает модель WorkingHour $workingHour

```
public function update(WorkingHourRequestDTO $request, WorkingHour $workingHour): void
{
    $workingHour->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель WorkingHour $workingHour

Ответ

return bool

```
public function destroy(WorkingHour $workingHour): bool
{
    return $workingHour->getCountRelationShip() ? $workingHour->delete() : false;
}
```

WorkingShiftService

App\Services\References

Свойства Данное поле хранит модель приходящую из конструктора класса protected Builder $model;

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected WorkingShift $workingShift
) {
    $this->model = $workingShift->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице справочника "Смены" Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return ReferenceDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

```
return array
```

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметр WorkingShift $workingShift - Модель смен

Ответ

`return array`

```
public function editData(WorkingShift $workingShift): array
{
    return [
        ...$this->formData(),
        'element' => ReferenceFormResource::make($workingShift)->resolve()
    ];
}
```

store

Данный метод реализует логику сохранения смены В качестве параметра метод принимает ReferenceRequestDTO $request

```
public function store(ReferenceRequestDTO $request): void
{
    $this->model->create($request->toArray());
}
```

update

Данный метод реализует логику обновления смены В качестве первого аргумента метод принимает ReferenceRequestDTO $request В качестве второго аргумента метод принимает модель WorkingShift $workingShift

```
public function update(ReferenceRequestDTO $request, WorkingShift $workingShift): void
{
    $workingShift->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель WorkingShift $workingShift

Ответ

return bool

```
public function destroy(WorkingShift $workingShift): bool
{
    return $workingShift->getCountRelationShip() ? $workingShift->delete() : false;
}
```

AccrualService

App\Services\Salary\Accruals

> Данный сервис отвечает за основную логику работы с модулем "Начисления" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait
>
> ## Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Accrual $accrual
) {
    $this->model = $accrual->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    $startDate = Carbon::parse($request->month)->startOfMonth();
    $endDate = Carbon::parse($request->month)->endOfMonth();
    $rawResults = DB::table(DB::raw("generate_series('$startDate', '$endDate', interval '1
        ->select(
            DB::raw('CAST(gs AS date) as date'),
            'employees.id',
            'employees.full_name',
            'accruals.id as accrual_id',
            'accruals.employee_id',
            DB::raw('COALESCE(accruals.sum, 0) as sum'),
            DB::raw('SUM(COALESCE(accruals.sum, 0)) OVER (PARTITION BY employees.id) as tot
        )
        ->crossJoin('employees')
        ->leftJoin('accruals', function ($join) {
            $join->on('accruals.date', '=', DB::raw('CAST(gs AS date)'))
                ->on('accruals.employee_id', '=', 'employees.id');
        })
        ->when($request->search['value'], function ($query) use ($request) {
            $query->where('employees.full_name', 'like', '%' . $request->search['value'] .
        })
        ->whereNull('employees.deleted_at')
        ->orderBy('employees.full_name')
        ->orderBy('date')
        ->get()
        ->groupBy('full_name');
    return DataTables::of($rawResults)
        ->setTransformer(function ($model) {
            return AccrualDTResource::make($model)->resolve();
        })
        ->toJson();
}
```

FactService

App\Services\Salary\Facts

> Данный сервис отвечает за основную логику работы с модулем "Факт" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Fact $fact
) {
    $this->model = $fact->query();
}
```

datatableHeaders

Данный метод возвращает массив с заголовками таблицы которые используются при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatableHeaders(DatatableRequestDTO $request): array
{
    $result = [];
    $result['customers'] = Fact::select([
            'facts.id',
            'subdivisions.id as subdivision_id',
            'subdivisions.name as subdivision_name',
            'customers.name as customer_name',
        ])
        ->where('date', Carbon::create($request->month)->startOfMonth())
        ->leftJoin('subdivisions', 'facts.subdivision_id', '=', 'subdivisions.id')
        ->leftJoin('customers', 'subdivisions.customer_id', '=', 'customers.id')
        ->get()
        ->groupBy('customer_name')
        ->toArray();
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    $startDate = Carbon::parse($request->month)->startOfMonth();
    $endDate = Carbon::parse($request->month)->endOfMonth();
    $rawResults = DB::table(DB::raw("generate_series('$startDate', '$endDate', interval '1
        ->select(
            DB::raw('CAST(gs AS date) as date'),
            'subdivisions.id as subdivision_id',
            'subdivisions.name as subdivision_name',
            DB::raw('COALESCE(facts.sum, 0) as fact_sum'),
            DB::raw('COALESCE(plans.sum, 0) as plan_sum'),
            DB::raw('SUM(COALESCE(facts.sum, 0)) OVER (PARTITION BY facts.date) as fact_tot
            DB::raw('SUM(COALESCE(plans.sum, 0)) OVER (PARTITION BY plans.date) as plan_tot
        )
        ->leftJoin('facts', function ($join) {
            $join->on('facts.date', '=', DB::raw('CAST(gs AS date)'));
        })
        ->leftJoin('plans', function ($join) {
            $join->on('plans.date', '=', DB::raw('CAST(gs AS date)'));
            $join->on('plans.subdivision_id', '=', 'facts.subdivision_id');
        })
        ->leftJoin('subdivisions', function ($join) {
            $join->on('facts.subdivision_id', '=', 'subdivisions.id');
        })
        ->get()
        ->groupBy('date');
    return DataTables::of($rawResults)
        ->setTransformer(function ($model) {
            return FactDTResource::make($model)->resolve();
        })
        ->toJson();
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Отвем

return bool

```
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->pluck('text', 'data')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'facts.xlsx'
    );
}
```

PayoutService

App\Services\Salary\Payouts

> Данный сервис отвечает за основную логику работы с модулем "Выплаты" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Payout $payout
) {
    $this->model = $payout->query();
}
```

datatableHeaders

Данный метод возвращает массив с заголовками таблицы которые используются при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatableHeaders(DatatableRequestDTO $request): array
{
    $result = [];
    $result['customers'] = Fact::select([
            'facts.id',
            'subdivisions.id as subdivision_id',
            'subdivisions.name as subdivision_name',
            'customers.name as customer_name',
        ])
        ->where('date', Carbon::create($request->month)->startOfMonth())
        ->leftJoin('subdivisions', 'facts.subdivision_id', '=', 'subdivisions.id')
        ->leftJoin('customers', 'subdivisions.customer_id', '=', 'customers.id')
        ->get()
        ->groupBy('customer_name')
        ->toArray();
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    $startDate = Carbon::parse($request->month)->startOfMonth();
    $endDate = Carbon::parse($request->month)->endOfMonth();
    $rawResults = DB::table(DB::raw("generate_series('$startDate', '$endDate', interval '1
        ->select(
            DB::raw('CAST(gs AS date) as date'),
            'employees.id',
            'employees.full_name',
            'payouts.id as payout_id',
            'payouts.employee_id',
            DB::raw('COALESCE(payouts.sum, 0) as sum'),
            DB::raw('SUM(COALESCE(payouts.sum, 0)) OVER (PARTITION BY employees.id) as tota
        )
        ->crossJoin('employees')
        ->leftJoin('payouts', function ($join) {
            $join->on('payouts.date', '=', DB::raw('CAST(gs AS date)'))
                ->on('payouts.employee_id', '=', 'employees.id');
        })
        ->when($request->search['value'], function ($query) use ($request) {
            $query->where('employees.full_name', 'like', '%' . $request->search['value'] .
        })
        ->whereNull('employees.deleted_at')
        ->orderBy('employees.full_name')
        ->orderBy('date')
        ->get()
        ->groupBy('full_name');
    return DataTables::of($rawResults)
        ->setTransformer(function ($model) {
            return PayoutDTResource::make($model)->resolve();
        })
        ->toJson();
}
```

store

Метод генерации excel документа Параметры PayoutRequestDTO $request - провалидированные данные

Ответ

return bool

```
public function store(PayoutRequestDTO $request): void
{
    $model = $this->model
        ->where('employee_id', $request->employee_id)
        ->where('date', $request->date)
        ->first();
    if ($model) {
        $model->increment('sum', $request->sum);
    } else {
        $this->model->create($request->toArray());
    }
}
```

update

Данный метод реализует логику обновления модели В качестве первого аргумента метод принимает PayoutRequestDTO $request В качестве второго аргумента метод принимает модель Payout $payout

Ответ

return bool

```
public function update(PayoutRequestDTO $request, Payout $payout): void
{
    $payout->update($request->toArray());
}
```

PlanService

App\Services\Salary\Plans

> Данный сервис отвечает за основную логику работы с модулем "План" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Plan $plan
) {
    $this->model = $plan->query();
}
```

datatableHeaders

Данный метод возвращает массив с заголовками таблицы которые используются при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatableHeaders(DatatableRequestDTO $request): array
{
    $result = [];
    $result['customers'] = Plan::select([
            'plans.id',
            'subdivisions.id as subdivision_id',
            'subdivisions.name as subdivision_name',
            'customers.name as customer_name',
        ])
        ->where('date', Carbon::create($request->month)->startOfMonth())
        ->leftJoin('subdivisions', 'plans.subdivision_id', '=', 'subdivisions.id')
        ->leftJoin('customers', 'subdivisions.customer_id', '=', 'customers.id')
        ->get()
        ->groupBy('customer_name')
        ->toArray();
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    $startDate = Carbon::parse($request->month)->startOfMonth();
    $endDate = Carbon::parse($request->month)->endOfMonth();
    $rawResults = DB::table(DB::raw("generate_series('$startDate', '$endDate', interval '1
        ->select(
            DB::raw('CAST(gs AS date) as date'),
            'subdivisions.id as subdivision_id',
            'subdivisions.name as subdivision_name',
            DB::raw('COALESCE(plans.sum, 0) as sum'),
            DB::raw('SUM(COALESCE(plans.sum, 0)) OVER (PARTITION BY plans.date) as total')
        )
        ->leftJoin('plans', function ($join) {
            $join->on('plans.date', '=', DB::raw('CAST(gs AS date)'));
        })
        ->leftJoin('subdivisions', function ($join) {
            $join->on('plans.subdivision_id', '=', 'subdivisions.id');
        })
        ->orderBy('plans.date')
        ->get()
        ->groupBy([
            'date'
            ]);
    return DataTables::of($rawResults)
        ->setTransformer(function ($model) {
            return PlanDTResource::make($model)->resolve();
        })
        ->toJson();
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

return bool

```
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->pluck('text', 'data')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'plans.xlsx'
    );
}
```

StatementService

App\Services\Salary\Plans

> Данный сервис отвечает за основную логику работы с модулем "Ведомость" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Statement $statement
) {
    $this->model = $statement->query();
}
```

getIndexData

Данный метод возвращает массив с данных для индексной странице

Ответ

`return array`

```
public function getIndexData(): array
{
    return [
        'employeeForms' => enumDataFormat(EmployeeFormsEnum::asSelectArray()),
    ];
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return StatementDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

SummaryService

App\Services\Salary\Summaries

> Данный сервис отвечает за основную логику работы с модулем "Свод" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait
>
> ## Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Summary $summary
) {
    $this->model = $summary->query();
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```php
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    $startDate = Carbon::parse($request->month)->startOfQuarter();
    $endDate = Carbon::parse($request->month)->endOfQuarter();
    $rawResults = DB::table(DB::raw("generate_series('$startDate', '$endDate', interval '1
        ->select(
            DB::raw('CAST(gs AS date) as date'),
            'employees.id',
            'employees.full_name',
            'summaries.id as summary_id',
            'summaries.employee_id',
            DB::raw('COALESCE(summaries.accrual_sum, 0) as accrual_sum'),
            DB::raw('COALESCE(summaries.incoming_sum, 0) as incoming_sum'),
            DB::raw('SUM(COALESCE(summaries.accrual_sum, 0)) OVER (PARTITION BY employees.i
            DB::raw('SUM(COALESCE(summaries.incoming_sum, 0)) OVER (PARTITION BY employees.
        )
        ->crossJoin('employees')
        ->leftJoin('summaries', function ($join) use ($request) {
            $join->on('summaries.date', '=', DB::raw('CAST(gs AS date)'))
                ->on('summaries.employee_id', '=', 'employees.id')
                ->where('summaries.customer_id', $request->customer['id']);
        })
        ->when($request->search['value'], function ($query) use ($request) {
            $query->where('employees.full_name', 'like', '%' . $request->search['value'] .
        })
        ->whereNull('employees.deleted_at')
        ->orderBy('employees.full_name')
        ->orderBy('date')
        ->get()
        ->groupBy('full_name');
    return DataTables::of($rawResults)
        ->setTransformer(function ($model) {
            return SummaryDTResource::make($model)->resolve();
        })
        ->toJson();
}
```

WorkedShiftService

App\Services\Salary\WorkedShifts

> Данный сервис отвечает за основную логику работы с модулем "Отработанные смены"
> Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Incoming $incoming,
    protected string $className = __CLASS__
) {
    $this->model = $incoming->query();
}
```

customers

Данный метод возвращает массив с заказчиками необходимых для отображения дататейблов на индексной странице Параметры WorkedShiftFilterRequestDTO $request - провалидированные данные

Ответ

`return array`

```
public function customers(WorkedShiftFilterRequestDTO $request): array
{
    return [
        'customers' => WorkedShift::select([
            'worked_shifts.id',
            'customers.id as customer_id',
            'customers.name',
        ])
        ->where('date', Carbon::create($request->date)->startOfMonth())
        ->leftJoin('customers', 'worked_shifts.customer_id', '=', 'customers.id')
        ->get()
    ];
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model)
        ->setTransformer(function ($model) {
            return WorkedShiftDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->month)) {
        // dd($request->month);
    }
}
```

formData

Данный метод отдает информацию для использования в формах

Ответ

`return array`

```
protected function formData(): array
{
    return [];
}
```

createData

Данный метод отдает информацию для страницы создания

Ответ

`return array`

```
public function createData(): array
{
    return [...$this->formData()];
}
```

editData

Данный метод отдает информацию для страницы редактирования

Ответ

`return array`

```
public function editData(WorkedShift $workedShift): array
{
    return [
        ...$this->formData(),
        'element' => WorkedShiftFormResource::make($workedShift)->resolve()
    ];
}
```

update

Данный метод реализует логику обновления периода заказчика В качестве первого аргумента метод принимает WorkedShiftRequestDTO $reques В качестве второго аргумента метод принимает модель WorkedShift $workedShift

```
public function update(WorkedShiftRequestDTO $request, WorkedShift $workedShift): void
{
    $workedShift->update($request->toArray());
}
```

destroy

Данный метод реализует удаление сущности из системы В качестве параметра метод принимает модель WorkedShift $workedShift

Ответ

return bool

```php
public function destroy(WorkedShift $workedShift): bool
{
    return $workedShift->getCountRelationShip() ? $workedShift->delete() : false;
}
```

TransportReportService

App\Services\TransportReports

> Данный сервис отвечает за основную логику работы с модулем "Отчёт транспорт" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Order $order,
    protected string $className = __CLASS__
) {
    $this->model = $order->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

datatable

Данный метод возвращает JSON объект, который используется при построении таблицы на индексной странице Для формирования *Ответ* а используется DatatableRequestDTO

```
public function datatable(DatatableRequestDTO $request): JsonResponse
{
    return DataTables::eloquent($this->model->with([
        'customer' => function ($query) {
            $query->select(
                'id',
                'name',
                'cost_transport_there_tk',
                'cost_transport_back_tk',
                'cost_transport_round_trip_tk'
            );
        },
        'workingShift' => function ($query) {
            $query->select('id', 'name');
        }
    ])->select('orders.*'))
        ->orderColumn('total', function ($query, $order) {
            $query->leftJoin('customers', 'customers.id', 'orders.customer_id')
                ->orderBy(
                    DB::raw('(cost_transport_there_tk * number_buses_there
                        + cost_transport_back_tk * number_buses_back
                        + cost_transport_round_trip_tk * number_buses_there_back)'),
                    $order
                );
        })
        ->setTransformer(function ($model) {
            return TransportReportDTResource::make($model)->resolve();
        })
        ->filter(function (Builder $query) use ($request) {
            $this->applyFilters($query, $request);
        }, true)
        ->toJson();
}
```

applyFilters

Данный метод применяет фильтры к запросу для datatable

Ответ

`return array`

```
public function applyFilters(Builder $query, DatatableRequestDTO $request): void
{
    if (isset($request->period[0])) {
        $query->where('working_shift_date', '>=', Carbon::parse($request->period[0])->start
            ->when(isset($request->period[1]), function ($item) use ($request) {
                return $item->where('working_shift_date', '<=', Carbon::parse($request->per
            });
    }
    if (isset($request->customer)) {
        $query->where('customer_id', $request->customer['id']);
    }
}
```

generateExcel

Метод генерации excel документа Параметры DataTableRequest $request - Данные прошедшие валидацию, со *Ответ* ствуют данным для метода datatable и со *Ответ* ствует параметрам отображения данных на странице заказчики

Ответ

return bool

```
public function generateExcel(DataTableRequestDTO $request): BinaryFileResponse
{
    $columns = (new Collection($request->columns))
        ->reject(fn($item) => $item['data'] === 'actions')
        ->pluck('text', 'name')
        ->toArray();
    $arrayQueryDT = json_decode($this->datatable($request)->content(), true);
    $arrayQueryDT['data'] = sorting_keys($arrayQueryDT['data'], $columns);
    return Excel::download(
        new BaseDatatableExport($arrayQueryDT['data'], $columns),
        'transport-reports.xlsx'
    );
}
```

WorkScheduleService

App\Services\WorkSchedules

> Данный сервис отвечает за основную логику работы с модулем "График работы" Класс содержит в себе следующие трейты: DataTableStoreColumnTrait

Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected WorkSchedule $invoice,
    protected string $className = __CLASS__
) {
    $this->model = $invoice->query();
}
```

getIndexData

Данный метод возвращает столбцы дататейбла

Ответ

`return array`

```
public function getIndexData(): array
{
    $result = [];
    if (Auth::check()) {
        $headers = Auth::user()->settingTables()->where('table', __CLASS__)->first();
        $result['headers'] = !empty($headers) ? $headers->column : null;
    }
    return $result;
}
```

store

Данный метод создаёт новую модель Принимает параметр WorkScheduleStoreRequestDTO $request

Ответ

return void

```
public function store(WorkScheduleStoreRequestDTO $request): void
{
    $data = $request->toArray();
    $data['start_date'] = $request->period[0];
    $data['end_date'] = $request->period[1];

    $this->model->updateOrCreate([
        'user_id' => $data['user_id'],
        'start_date' => $data['start_date'],
        'end_date' => $data['end_date'],
    ], $data);
}
```

formData

Данный отдаёт данные для формы создания/редактирования Параметры Принимает параметр WorkScheduleFormDataRequestDTO $request

Ответ

`return array`

```
public function formData(WorkScheduleFormDataRequestDTO $request): array
{
    $workSchedule = $this->model->with('user')
        ->where('user_id', $request->user_id)
        ->where('start_date', $request->period[0])
        ->where('end_date', $request->period[1])->first();
    return [
        'element' => $workSchedule ? new WorkScheduleFormResource($workSchedule) : null
    ];
}
```

editData

Данный метод отдает информацию для страницы редактирования Параметры Принимает параметр WorkSchedule $workSchedule

Ответ

`return array`

```
public function editData(WorkSchedule $workSchedule): array
{
    return [
        'element' => new WorkScheduleFormResource($workSchedule->load([
            'user'
        ]))
    ];
}
```

SettingService

App\Services

> Данный сервис отвечает за основную логику работы с модулем "Настройки" Класс содержит в себе следующие трейты: ConstantTrait
>
> ## Список методов класса

__construct

Данный метод инициализирует поле класса $model

```
public function __construct(
    protected Setting $setting)
{
    $this->model = $setting->query();
}
```

outputData

Данный метод возвращает данные настроек

Ответ

return Illuminate\Support\Collection

```
public function outputData(): Collection
{
    /**
     * @var Collection<string, mixed> $config
     */
    $config = new Collection(config('main_settings'));
    $setting = Setting::whereIn('key', self::$settingConst)
        ->pluck('value', 'key');
    /**
     * @var Collection<string, mixed> $files
     */
    $files = new Collection(self::$settingInput);
    $files->filter(fn ($value) => $value == 'image')->keys();
    $setting = $setting->map(function ($value, $key) use ($files) {
        if ($files->contains($key)) {
            return route('file_public', $value);
        }
        return $value;
    });
    return $config->merge($setting);
}
```

getIndexData

Метод возвращает данные необходимые для настройки системы

Ответ

`return array`

```
public function getIndexData(): array
{
    return [
        'inputs' => self::$settingInput,
        'elements' => $this->outputData()->toArray()
    ];
}
```

store

Метод сохраняет настройки системы

Ответ

`return array`

```
public function store(array $request): array
{
    foreach ($request['elements'] as $key => $item) {
        $setting = Setting::firstorCreate(
            [
                'key' => $key
            ],
            [
                'value' => null
            ]
        );
        $value = $setting->value;
        if (self::$settingInput[$key] === 'image') {
            if (!empty($item) || (isset($request["delete_$key"])) && $request["delete_$key"
                $setting->getMedia('public')->isNotEmpty() ? $setting->deleteImage() : fals
                $value = null;
            }
            if (!empty($item)) {
                $file = $setting->addImage($item);
                $value = $file->uuid;
            }
        }
        if (!empty($value)) {
            $setting->update(['value' => $value]);
        } else {
            $setting->delete();
        }
    }
    return $this->getIndexData();
}
```