INFOtec Informatikai és Információtechnikai Kft

Identity Hungary Kft. - GINOP-2.1.7-15-2016-02650 GarNET – egységes garanciakezelő kártya rendszer

FELADAT 3

SZERVER OLDALI FEJLESZTÉSI FELADATOK

(Back end fejlesztői dokumentáció)

Packages

A framework felépítése során használt csomagok:

- Laravel : https://github.com/laravel/laravel
- ACL : https://github.com/spatie/laravel-permission
- Passport : https://github.com/laravel/passport
- ApiDoc : https://github.com/mpociot/laravel-apidoc-generator
- Telescope : https://github.com/laravel/telescope
- CORS : https://github.com/barryvdh/laravel-cors
- Predis : https://github.com/nrk/predis

Hasznos link(ek):

- https://medium.com/modulr/create-api-authentication-with-passport-of-laravel-5-6-1dc2d400a7f

API Doc:

- Új dokumentum generálás : php artisan apidoc:generate
- Meglévő dokumentum újragenerálás : php artisan apidoc:rebuild

CORS:

- Módosított érték(ek):
  - 'allowedMethods' => ['GET', 'POST', 'PUT', 'PATCH', 'DELETE']
  - 'maxAge' => 1728000,

Szükséges header-ek:

- Accept : application/json
- X-Requested-With : XMLHttpRequest
- Content-Type : application/json

Amennyiben az adott endpoint bejelentkezéshez kötött, abban az esetben token küldése szükséges a következő formátumban

- Authentication : tokenType token [Bearer {token here}]

Végpontok:

- [POST] : /api/login
- [POST] : /api/user/signup
- [GET] : /api/user/signup/activate/{token}
- [POST] : /api/user/details
- [POST] : /api/user/logout
- [POST] : /api/user/password/reset/create
- [GET] : /api/user/password/reset/find/{token}
- [POST] : /api/user/password/reset
- [GET] : /api/test

Telepítés

Telepítés:

```
* git clone [ from git repository ]
* docker-compose up
* a build után: docker exec -ti [container name] bash
* composer update
* php artisan key:generate
* php artisan migrate
* php artisan passport:install
* php artisan db:seed
* chmod -R 777 storage/
```

DB migráció után (php artisan migrate:fresh –seed) mindenképpen szükséges a következő artisan parancs futtatása: php artisan passport:install

Alapértelmezett felhasználói csoportok:

```
* Admin         [ role ID : 1 ]
* User          [ role ID : 3 ]
* Seller        [ role ID : 4 ]
* Service       [ role ID : 5 ]
```

Alapértelmezett jogosultságok:

```
* Not exists
```

Unit test

10%

https://blog.pusher.com/tests-laravel-applications/

https://semaphoreci.com/community/tutorials/getting-started-with-phpunit-in-laravel

A tesztelést az éles (production) adatbázistól teljesne független teszt adatbázissal szükséges elvégezni, ezzel elkerülendő az éles adatok nem várt módosítását, illetve az adatbázis szükségtelen méretnövelését. A teszteléshez egy külön erre a célra docker configurációs file-al kell elindítani a rendszert.

A tesztesetek a `./tests` mappában találhatóak.

Teszteléskor a `.env.testing` és `docker-compose-unit-test.yml` file-ok vannak használatban.

A teszt rendszer indítása

- docker-compose -f docker-compose-unit-test.yml up
- docker exec -ti < container > bash
- php artisan migrate --seed --env=testing

Amennyiben nem teljesen új telepítésről van szó, abban az esetben a `php artisan migrate --seed --env=testing` parancs helyett a `php artisan migrate:fresh --seed --env=testing` futtatása szükséges.

A teszt indítása:

- docker-compose -f docker-compose-unit-test.yml up
- docker exec -ti < container > bash
- ./vendor/bin/phpunit

Sikeres futás válasz:

```
root@7fbe8e7c502a:/app# ./vendor/bin/phpunit
PHPUnit 7.5.1 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 78 ms, Memory: 16.00MB

OK (2 tests, 2 assertions)
```

Sikertelen futás válasz:

```
root@7fbe8e7c502a:/app# ./vendor/bin/phpunit
PHPUnit 7.5.2 by Sebastian Bergmann and contributors.

.F.....                                                             7 / 7 (100%)

Time: 838 ms, Memory: 26.00MB

There was 1 failure:

1) Tests\Unit\AuthTest::testLogin
Expected status code 200 but received 401.
Failed asserting that false is true.

/app/vendor/laravel/framework/src/Illuminate/Foundation/Testing/TestResponse.php:133
/app/tests/Unit/Auth/AuthTest.php:51

FAILURES!
Tests: 7, Assertions: 9, Failures: 1.
```

Tesztek:

File	Name	Required response
AuthRoutesTest	`testFakeLogin()`	401
AuthRoutesTest	`testValidLogin()`	200
AuthRoutesTest	`testRegistration()`	200
AuthRoutesTest	`testLogoutWithFakeToken())`	401
AuthRoutesTest	`testLogoutWithUnauthenticatedUser()`	401
AuthRoutesTest	`testWholeAuthProcessWithGeneratedUSer()`	200
CustomerRoutesTest	`testGetAllWithAuthenticatedUser()`	200
CustomerRoutesTest	`testGetSpecifiedWithAuthenticatedUser()`	200
CustomerRoutesTest	`testStoreWithAuthenticatedUser()`	200
CustomerRoutesTest	`testUpdateWithAuthenticatedUser()`	200
CustomerRoutesTest	`testUpdateStatusWithAuthenticatedUser()`	200
CustomerRoutesTest	`testGetAllWithUnauthenticatedUser()`	401
CustomerRoutesTest	`testGetSpecifiedWithUnauthenticatedUser()`	401
CustomerRoutesTest	`testStoreWithUnauthenticatedUser()`	401
CustomerRoutesTest	`testUpdateWithUnauthenticatedUser()`	401
CustomerRoutesTest	`testUpdateStatusWithUnauthenticatedUser()`	401
SellerRoutesTest	`testGetAllWithAuthenticatedUser()`	200
SellerRoutesTest	`testGetSpecifiedWithAuthenticatedUser()`	200
SellerRoutesTest	`testStoreWithAuthenticatedUser()`	200
SellerRoutesTest	`testUpdateWithAuthenticatedUser()`	200
SellerRoutesTest	`testUpdateStatusWithAuthenticatedUser()`	200
SellerRoutesTest	`testGetAllWithUnauthenticatedUser()`	401
SellerRoutesTest	`testGetSpecifiedWithUnauthenticatedUser()`	401
SellerRoutesTest	`testStoreWithUnauthenticatedUser()`	401
SellerRoutesTest	`testUpdateWithUnauthenticatedUser()`	401
SellerRoutesTest	`testUpdateStatusWithUnauthenticatedUser()`	401

- **Common/Database**

Adatbázis migráció:

100%

Alapok:

A garNet backend rendszer úgynevezett migrációs és seeder file-okat hazsnál az adatábizsban használtl táblák létrehozására / módosítására, illetve a táblában tárolt alapértelmezett adatok kezelésére.

A migrációs file-ban `up()` függvény meghívásával van lehetőség létrehozni / módosítani az file-ban definiált tábla adatait. A `down()` függvény meghívásával törölni lehet a definiált táblát.

Példa migrációs file:

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return  void
    */
    public function up()
    {

        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->boolean('active')->default(false);
            $table->string('activation_token')->nullable();
            $table->string('revoke_token')->nullable();
            $table->rememberToken();
            $table->string('default_language')->nullable()->default("hu");
            $table->timestamps();
            $table->softDeletes();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return  void
    */
    public function down()
    {
        Schema::dropIfExists('users');

    }
}
```

A táblák létrehozása után bizonyos esetekben szükséges alapértlemezett adatokkal feltölteni azokat. Az adatok feltötését a backend rendszer úgynevezett seeder file-ok segítségével valósítja meg. A rendszernek szüksége van egy default Seeder file-ra, amelynek neve `DatabaseSeeder.php`, ahol a `run()` függvényben definiálva van az összes olyan Seeder file-amit migrációkor a rendszernek le kell futtatnia.

DatabaseSeeder.php:

```php
<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return  void
    */
    public function run()
    {
        $this->call([
            CreateRolesTableSeeder::class,
            CreateUsersTableSeeder::class,
            CreateCustomerTableSeeder::class,
            CreateAddressesTableSeeder::class,
            CreateContactPersonsTableSeeder::class,
            CreateItemTypesTableSeeder::class,
            CreatePaymentTypesTableSeeder::class,
            CreateSellersTableSeeder::class,
            CreateWarrantyItemsTableSeeder::class,
            CreateWarrantiesTableSeeder::class,
            CreateDummyTableSeeder::class

        ]);
    }
}
```

Példa Seeder file:

```php
<?php
use Illuminate\Database\Seeder;
use Spatie\Permission\Models\Role;
use App\User;

class CreateUsersTableSeeder extends Seeder
{
    /**
    * Run the database seeds.
    *
    * @return  void
    */

    public function run()
    {
        $user                       = [];
        $user['email']              = 'test.seller1@identity-hungary.hu';
        $user['email_verified_at']  = now();
        $user['password']           = '$2y$12$fnvEaXTfGyzT7YZwaD5sEO9S413Jw0wHo0gGIlcFi1ETOji.IUZrW';
        $user['active']             = true;
        $user['activation_token']   = null;
        $user['revoke_token']       = null;
        $user['remember_token']     = str_random(10);
        $user['default_language']   = 'hu';

        $data = User::create($user);
        $role = Role::where('id', 4)->first();
        $data->assignRole($role);
    }
}
```

1. Telepítéskor szükséges parancsok:

A rendszer telepítésekor szükséges a migráció lefuttatására / a szükséges adatbázis táblák létrehozására, illetve az alapértlemezett adatok importálására.

Installálásakor a következő parancsok futtatása szükséges:

- docker exec -ti [container name] bash
- php artisan migrate
- php artisan db:seed

2. Frissítéskor szükséges parancsok:

Amennyiben változtatás történt a migrációs / seeder file-okban, abban az esetben szükséges a migráció újbóli lefuttatása / az adatbázis frissítése.

Frissítéskor a következő parancsok futtatása szükséges:

- docker exec -ti [container name] bash
- php artisan migrate:fresh --seed

Adatbázis struktúra

100%

Table	Documentation	Structure is final
address	DONE	YES
contact_person	DONE	YES
customer	DONE	YES
customer_invoice	DONE	YES
customer_invoice_item	DONE	YES
customer_invoice_extra_item	DONE	YES
customer_notification_log	DONE	YES
customer_upload	DONE	YES
form_groups	DONE	YES
form_items	DONE	YES
migrations	NOT RELEVANT	NOT RELEVANT
model_has_permissions	DONE	YES
model_has_roles	DONE	YES
oauth_access_tokens	DONE	YES
password_resets	DONE	YES
permissions	DONE	YES
product_category	DONE	YES
role_has_permissions	DONE	YES
roles	DONE	YES
seller	NOT RELEVANT	NOT RELEVANT
seller_product_category	NOT RELEVANT	NOT RELEVANT
telescope_entries	NOT RELEVANT	NOT RELEVANT
telescope_entries_tag	NOT RELEVANT	NOT RELEVANT
telescope_monitoring	NOT RELEVANT	NOT RELEVANT
users	DONE	YES

Tábla: users

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
email	VARCHAR(191)		X	X					
email_verified_at	TIMESTAMP	X	X			X		X	
password	VARCHAR(191)		X						
active	TINYINT(1)		X						
activation_token	VARCHAR(191)								
revoke_token	VARCHAR(191)								
remember_token	VARCHAR(191)								
default_language	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: address

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
postal_code	VARCHAR(191)		X						
city	VARCHAR(191)		X						
street	VARCHAR(191)		X						
street_number	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: contact_person

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
name	VARCHAR(191)		X						
email	VARCHAR(191)		X						
mobile_phone	VARCHAR(191)								
wired_phone	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: customer

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
user_id	INT(10)		X			X			
first_name	VARCHAR(191)								
last_name	VARCHAR(191)								
mobile_phone	VARCHAR(191)								
wired_phone	VARCHAR(191)								
home_address_id	INT(10)		X			X			
home_address_door_number	VARCHAR(191)								
postal_address_id	INT(10)		X			X			
postal_address_door_number	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: customer_invoice

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
invoice_id	INT(10)		X						
user_id	INT(10)		X						
seller_id	INT(10)		X						
purchased	DATE		X						
app_no	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: customer_invoice_item

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
invoice_id	INT(10)		X			X			
name	VARCHAR(191)		X						
product_category_id	INT(10)		X			X			
product_subcategory_id	INT(10)		X			X			
manufacture	VARCHAR(191)		X						
type_no	VARCHAR(191)		X						
barcode	VARCHAR(191)		X						
price	INT(10)		X						
warranty_in_month	INT(10)		X			X			
warranty	TINYINT(1)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: customer_invoice_extra_item

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
invoice_item_id	INT(10)		X			X			
form_group_id	INT(10)		X						
form_item_id	INT(10)		X						
form_item_value	VARCHAR(191)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: customer_notification_log

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
user_id	INT(10)		X						
invoice__id	INT(10)		X						
invoice_item_id	INT(10)		X			X			
expire_date	INT(10)		X						
notification_date	INT(10)		X						
type	VARCHAR(191)		X						
cron_at	DATETIME								
read_at	DATETIME								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: customer_upload

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
filename	VARCHAR(191)		X						
extension	VARCHAR(191)		X						
path	VARCHAR(191)		X						
size	INT(10)		X						
invoice_id	INT(10)		X			X			
invoice_item_id	INT(10)		X			X			
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: form_groups

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
product_category_id	INT(11)		X						
name	VARCHAR(191)		X						
description	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: form_items

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
form_group_id	INT(11)		X						
type	VARCHAR(191)		X						
name	VARCHAR(191)		X						
label	VARCHAR(191)		X						
value	VARCHAR(191)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
deleted_at	TIMESTAMP								

Tábla: model_has_permissions

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
permission_id	INT(10)	X	X			X			
model_type	VARCHAR(191)	X	X						
model_id	BIGINT(20)	X	X			X			

Tábla: model_has_roles

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
role_id	INT(10)	X	X			X			
model_type	VARCHAR(191)	X	X						
model_id	BIGINT(20)	X	X			X			

Tábla: oauth_access_tokens

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	VARCHAR(100)	X	X						
user_id	INT(11)								
client_id	INT(11)		X						
name	VARCHAR(191)								
scopes	TEXT								
revoked	TINYINT(1)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								
expires_at	DATETIME								

Tábla: password_resets

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
email	VARCHAR(191)		X						
token	VARCHAR(191)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								

Tábla: permissions

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
name	VARCHAR(191)		X						
guard_name	VARCHAR(191)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								

Tábla: product_category

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
name	VARCHAR(191)		X						
parent_id	INT(11)								
created_at	TIMESTAMP								
updated_at	TIMESTAMP								

Tábla: role_has_permissions

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
permisison_id	INT(10)	X	X			X			
role_id	INT(10)	X	X			X			

Tábla: roles

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
name	VARCHAR(191)		X						
guard_name	VARCHAR(191)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								

Tábla: seller_product_category

Column	DataType	PK	NN	UQ	BIN	UN	ZF	AI	G
id	INT(10)	X	X			X		X	
seller_id	INT(11)		X						
product_category_id	INT(11)		X						
created_at	TIMESTAMP								
updated_at	TIMESTAMP								

- **Common/Files**

Form requests:

100%

FormRequest files:

Az `App\Http\Request\` mappában találahtóak az úgynevezett fromRequest file-ok. A formRequest file-ok szerepe, hogy az adott GET / POST hívások során küldött paraméterek / adatok ellenőrzését / módosítását megfelelő bizontásggal el lehessen végezni. A garNet rendszerben minden endpointhoz tartzik egy formRequest file.

A formRequest file-ok létrehozása a `php artisan make:request {RequestName}` parancs segítségével történik. Ebben ez esetben a rendszer generál egy request file-t az alábbi előre definiált formátumban. Az `authorize()` függvény segítségével szabályozható, hogy az adott request csak authentikált felhazsnálók számára érhető-e el. A `rules()` függvény segítségével szabályozható, hogy a küldött adatoknak milyen validálási szabályoknak kell megfelelnie.

A `withValidator()` függvény nem alapértelmezett függvény, viszont minden olyna esetben hasznos, amikor az alapértelmezett validálás után egyéb, összetett feltételeknek is meg kell felelnie a küldött adat(ok)nak.

```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class TestRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     *
     * @return  bool
     */
    public function authorize()
    {
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            //
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            if ('custom validation is failed') {
                return $validator->errors()->add('id', 'Custom error message');
            }
        });
        return;
    }
}
```

- **Admin/Auth**

Bejelentkezés

100%

A bejelentkezés email / jelszó páros megadásával történik. Sikeres bejelentkezés után a garNet backend rendszer egy, a bejelentkezett felhasználó adatait tartalmazó objektummal tér vissza frontend rendszer részére.Ez az objektum tartalmaz egy úgynevezett AccessToken-t, amellyel a token lejárati dátumáig a felhasználó azonosítható és a különböző védett kérésekhez a hozzáférése szabályozható. Az authentikációt a Passport nevű open source csomag segítségével valósítjuk meg, amely a Google OAuth2 authentikációs folyamatra épül. Az adatok megadása után a frontend a szükséges endpoint felé továbbítja az adatokat.

Endpoint:

Method	Endpoint	Headers
`POST`	`api/user/login`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
email	`string`	`required`	Email
password	`string`	`required`	Jelszó
remember_me	`boolean`	`optional`	Emlékeztessen később

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Unauthorized, security problem	`422`
Internal Server Error	`500`

Regisztráció

100%

1. Regisztráció

A regisztráció folyamatának első lépéseként a regisztrációhoz szükséges adatok megadása szükséges. Az adatok megadása után a garNet backend rendszer a regisztráció során megadott postafiók címre kiküld egy emailt, amiben a regisztráció véglegesítéséhez szükséges link található. A linkben szerepel egy 60 karakter hosszúságú speciális hash, ami a regisztrált felhasználóhoz kötődik és egyértelműen azonosítja őt. Az adatok megadása után a frontend a szükséges endpoint felé továbbítja az adatokat.

Regisztrációhoz szükséges adatok:

- email
- jelszó
- jelszó ismétlése
- jogosultsági szint

A regisztrációkor elvégzett biztonsági ellenőrzések:

- az összes szükséges mező kitöltése megtörtént
- az email mezőben szereplő adat valós email (formátum szempontjából)
- az email nem szerepel még az adatbázisban
- a jelszó mezőben szereplő adat minimum 8 karakter hosszúságú hash, ami tartalmaz kisbetűt, nagybetűt, számot
- a jelszó és jelszó ismétlése mezőben szereplő hash megegyezik

Endpoint:

Method	Endpoint	Headers
`POST`	`api/user/signup`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
email	`string`	`required`	Email
password	`string`	`required`	Jelszó
password_confirmation	`string`	`required`	Jelszó ismét
role_id	`integer`	`required`	Jogosultsági csoport

Handled responses:

Status	StatusCode
Success	`200`
Internal Server Error	`500`

Sample success response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": {
        "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6Ijc0ZTA5ZDgyMGQwNzgyMTA4NWNkODNkODBhNjE2ZmNmNmQyYmNhYzMxOGQ2ZWEwNTg0N2EzMTQzMDk4ZDliNmM0M
    }
}
```

2. Megerősítés

A linkre kattintva a felhasználó véglegesítheti a regisztrációt (amennyiben a biztonsági feltételeknek megfelel). Amíg a regisztráció megerősítése nem történik meg, addig a felhasználó nem tud bejelentkezni a garNet rendszerbe. Az adatok megadása után a frontend a szükséges endpoint felé továbbítja az adatokat.

Regisztráció véglegesítésekor elvégzett biztonsági ellenőrzések:

- a linkben szereplő aktivációs kód szerepel az adatbázisban
- a linkben szereplő aktivációs kód valós aktivációs kód
- a linkben szereplő aktivációs kód érvényes

A garNet backend rendszerben a felhasználó egy adott típusú felhasználóként kerül rögzítésre. A regisztráció alapadatai egy közös adatbázis táblába (users tábla), a különböző típusú felhasználókhoz tartozó speciális adatok külön – külön adatbázis táblákban kerülnek rögzítésre.

Method	Endpoint	Headers
`GET`	`api/user/signup/activate/{token}`	Alapértelmezett

Query parameters:

Param	Type	Status	Description
token	`string`	`required`	Token

Handled responses:

Status	StatusCode
Success	`200`
Internal Server Error	`500`

Sample success response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "id": 16,
    "email": "john.doe@example.com",
    "email_verified_at": {
        "date": "2019-01-22 09:23:47.007925",
        "timezone_type": 3,
        "timezone": "UTC"
    },
    "active": true,
    "default_language": "hu",
    "created_at": "2019-01-22 08:17:39",
    "updated_at": "2019-01-22 09:23:47",
    "deleted_at": null
}
```


Kijelentkezés:

100%

Kijelentkezés esetén a garNet backend rendszer a bejelentkezett felhasználó AccessToken-ét törli a rendszerből, ezzel biztosítva a sikeres kijelentkezést. Az adatok megadása után a frontend a szükséges endpoint felé továbbítja az adatokat.

Endpoint:

Method	Endpoint	Headers
`POST`	`api/user/logout`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
---	---	---	---

Handled responses:

Status	StatusCode
Success	`200`
Internal Server Error	`500`

Sample responses:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "message": "Successfully logged out"
}
```

Elfelejtett jelszó:

100%

1. Új jelszó igénylés

Amennyiben a felhasználó elfelejtette a jelszavát, abban az esetben lehetősége van új jelszót igényelni. Biztonsági okokból "clearText" jelszót a rendszer nem tárol és nem küld ki a felhasználónak email-ben. A folyamat első lépéseként a felhasználó által megadott postafiókra küld a garNet rendszer egy emailt, amiben a jelszócseréhez szükséges link található.

Endpoint:

Method	Endpoint	Headers
`POST`	`user/password/reset/create`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
email	`string`	`required`	Email

Handled responses:

Status	StatusCode
Success	`200`
Internal Server Error	`500`

Sample success response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "message": "We have e-mailed your password reset link."
}
```

2. Megerősítés

A linkben szerepel egy 60 karakter hosszúságú speciális hash, ami a regisztrált felhasználóhoz kötődik és egyértelműen azonosítja őt.

A jelszócserekor elvégzett biztonsági ellenőrzések:

- a linkben szereplő hash szerepel az adatbázisban
- a linkben szereplő hash valós hash
- a linkben szereplő hash érvényes

Endpoint:

Method	Endpoint	Headers
`GET`	`api/user/password/reset/find/{token}`	Alapértelmezett

Query parameters:

Param	Type	Status	Description
token	`string`	`required`	Token

Handled responses:

Status	StatusCode
Success	`200`
Token is invalid	`422`
Internal Server Error	`500`

Sample success response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": {
        "id": 1,
        "email": "john.doe@example.com",
        "token": "0ofYUmTPHlZg0MF8UVxbKljWVPeibUgzHAO6lZhsmdxzRaxK49WKGfD4wJ6J",
        "created_at": "2019-01-22 11:55:39",
        "updated_at": "2019-01-22 11:55:39"
    }
}
```

3. Véglegesítés

Amennyiben a 2. pontban végzett biztonsági ellenőrzés során a garNet rendszer nem talál problémát, átirányítja a felhasználót a `/reset-password/{token}` URL-en található felületre, ahol meg tudja adni az új jelszavát. Az adatok megadása után a frontend a szükséges endpoint felé továbbítja az adatokat. Sikeres módosítás esetén a user objektummal tér vissza a backend rendszer, ezzel egyidőben pedig egy email kerül kiküldésre a felhasználónak a sikeres jelszócseréről.

A jelszócserekor elvégzett biztonsági ellenőrzések:

- az összes szükséges mező kitöltése megtörtént
- a jelszó mezőben szereplő adat minimum 8 karakter hosszúságú hash, ami tartalmaz kisbetűt, nagybetűt, számot, speciális karaktert
- a jelszó és jelszó ismétlése mezőben szereplő hash megegyezik.

Endpoint:

Method	Endpoint	Headers
`POST`	`user/password/reset`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
email	`string`	`required`	Email
password	`string`	`required`	Password
password_confirmation	`string`	`required`	Password again
token	`string`	`required`	Token

Handled responses:

Status	StatusCode
Success	`200`
Token is invalid	`422`
Unknown email	`422`
Internal Server Error	`500`

Sample success response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": {
        "id": 1,
        "email": "john.doe@example.com",
        "email_verified_at": "2019-01-03 07:57:19",
        "active": 1,
        "default_language": "hu",
        "created_at": "2019-01-03 07:57:19",
        "updated_at": "2019-01-22 12:30:42",
        "deleted_at": null
    }
}
```


- **Admin/Roles**

Függvények:

20%

index():

Az összes `role` (jogosultsági szint) listázása. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/roles`	Alapértelmezett
`GET`	`api/admin/roles?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "name": "Admin",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        },
        {
            "id": 2,
            "name": "Customer",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        },
        {
            "id": 3,
            "name": "Seller",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
```

```
        },
        {
            "id": 4,
            "name": "Service",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

Adott `role` listázása. A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/roles/{id}`	Alapértelmezett
`GET`	`api/admin/roles/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	Role ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
The given data was invalid	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": {
        "role": {
            "id": 1,
            "name": "Admin",
            "guard_name": "api",
            "created_at": "2019-02-27 09:31:39",
            "updated_at": "2019-02-27 09:31:39",
            "permissions": []
        },
        "rolePermissions": [],
        "allowedPermissions": []
    }
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


Status: `The given data was invalid` StatusCode: `422` Response: `JSON`


```
{
    "message": "The given data was invalid.",
    "errors": {
        "id": [
            "Data not found"
        ]
    }
}
```


store():

Új `role` rögzítése. A rögzítés több lépésben történik tranzakciókezelés használatával.

Rögzítés menete:

- a `role` alapadatinak rögzítése a `roles` táblába
- a `role` -hoz tartozó `permission` -ok rögzítése a `role_has_permissions` táblába

Method	Endpoint	Headers
`POST`	`api/admin/roles`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`name`	`string`	`required|string`	Name of role
`guard_name`	`string`	`required|string`	Type of guard
`permissions`	`array`	`required`	List of attached permissions

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

update():

Meglévő `role` adatainak módosítása. A módosítás több lépésben történik tranzakciókezelés használatával.

Módosítás menete:

- a `role` alapadatinak módosítása a `roles` táblába
- a `role` -hoz tartozó által `permission` -ök frissítése

Method	Endpoint	Headers
`PUT`	`api/admin/roles/{id}`	Alapértelmezett

Param	Type	Status	Description
`role_id`	`integer`	`required|integer`	Role ID
`name`	`string`	`required|string`	Name of role
`guard_name`	`string`	`required|string`	Type of guard
`permissions`	`array`	`required`	List of attached permissions

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

destroy():

Meglévő `role` törlése az adatbázisból. A törlés több lépésben történik tranzakciókezelés használatával. Adott `role` törlése végleges, tehát nem visszaállítható!

Módosítás menete:

- a `role` alapadatinak törlése a `roles` táblába
- a `role` -hoz tartozó által `permission` -ök törlése

Method	Endpoint	Headers
`DELETE`	`api/admin/roles/{id}`	Alapértelmezett

Param	Type	Status	Description
`role_id`	`integer`	`required|integer`	Role ID

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/admin/roles/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
    {
        return [];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/admin/roles/` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

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

    /**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "name"          => "required|string",
            "guard_name"    => "required|string",
            "permissions"   => "required"
        ];
    }
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[PUT] api/admin/roles/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "id"            => "required",
            "name"          => "required|string",
            "guard_name"    => "required|string",
            "permissions"   => "required|array"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
```

```
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

DeleteRequest:

Az `DeleteRequest` file tartalmazza az `[DELETE] api/admin/roles/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "id" => "required|integer"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

- **Admin/Permissions**

Függvények:

20%

index():

Az összes `permission` (jogosultsági szint) listázása. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/permissions`	Alapértelmezett
`GET`	`api/admin/permissions?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "name": "testPermission",
            "guard_name": "api",
            "created_at": "2019-02-28 12:32:55",
            "updated_at": "2019-02-28 12:32:55"
        },
        {
            "id": 2,
            "name": "testPermission2",
            "guard_name": "api",
            "created_at": "2019-02-28 12:32:55",
            "updated_at": "2019-02-28 12:32:55"
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

Adott `permission` listázása. A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/permissions/{id}`	Alapértelmezett
`GET`	`api/admin/permissions/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	Permission ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
The given data was invalid	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "name": "testPermission",
            "guard_name": "api",
            "created_at": "2019-02-28 12:32:55",
            "updated_at": "2019-02-28 12:32:55"
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```

Status: `The given data was invalid` StatusCode: `422` Response: `JSON`


```
{
    "message": "The given data was invalid.",
    "errors": {
        "id": [
            "Data not found"
        ]
    }
}
```


store():

Új `permission` rögzítése. A rögzítés tranzakciókezelés használatával történik.

Rögzítés menete:

- a `permission` alapadatinak rögzítése a `permissions` táblában

Method	Endpoint	Headers
`POST`	`api/admin/permissions`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`name`	`string`	`required|string`	Name of permission
`guard_name`	`string`	`required|string`	Type of guard

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

update():

Meglévő `permission` adatainak módosítása. A módosítás tranzakciókezelés használatával történik.

Módosítás menete:

- a `permission` alapadatinak módosítása a `permissions` táblában

Method	Endpoint	Headers
`PUT`	`api/admin/permissions/{id}`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`permission_id`	`integer`	`required|integer`	Id of permission
`name`	`string`	`required|string`	Name of role
`guard_name`	`string`	`required|string`	Type of guard

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

destroy():

Meglévő `permission` törlése az adatbázisból. A törlés több lépésben történik tranzakciókezelés használatával. Adott `permission` törlése végleges, tehát nem visszaállítható!

Módosítás menete:

- a `permission` alapadatinak törlése a `permissions` táblába
- a `permission` eltávolítása azokból a `role` -okból, amikhez csatolva van

Method	Endpoint	Headers
`DELETE`	`api/admin/permissions/{id}`	Alapértelmezett

Param	Type	Status	Description
`permission_id`	`integer`	`required|integer`	Id of permission

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/admin/permissions/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
    {
        return [];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/admin/permissions/` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

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

    /**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "name"          => "required|string",
            "guard_name"    => "required|string"
        ];
    }
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[PUT] api/admin/permissions/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```php
public function rules()
    {
        return [
            "id"            => "required|integer",
            "name"          => "required|string",
            "guard_name"    => "required|string"
        ];
    }
```

```php
/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        $data = $this->model->getById(
            $this->id,
            $with_trashed    = false,
            $with_paginate   = false,
            $with_relation   = false
        );

        if (!$data || $data->count() == 0) {
            $validator->errors()->add('id', 'Data not found');
        }
    });
    return;
}
```

DeleteRequest:

Az `DeleteRequest` file tartalmazza az `[DELETE] api/admin/permissions/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
    {
        return [
            "id" => "required|integer"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
      // checks user current password
      // before making changes
      $validator->after(function ($validator) {
          $data = $this->model->getById(
              $this->id,
              $with_trashed    = false,
              $with_paginate   = false,
              $with_relation   = false
          );

          if (!$data || $data->count() == 0) {
              $validator->errors()->add('id', 'Data not found');
          }
      });
      return;
  }
```

- **Admin/Users**

Függvények:

20%

index():

Az összes `user` (felhazsnáló) listázása. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/users`	Alapértelmezett
`GET`	`api/admin/users?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "name": "Admin",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        },
        {
            "id": 2,
            "name": "Customer",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        },
        {
            "id": 3,
            "name": "Seller",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        },
        {
            "id": 4,
            "name": "Service",
            "guard_name": "api",
            "created_at": "2019-02-19 10:46:23",
            "updated_at": "2019-02-19 10:46:23"
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```

show():

Adott `role` listázása. A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/roles/{id}`	Alapértelmezett
`GET`	`api/admin/roles/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	Role ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
The given data was invalid	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": {
        "role": {
            "id": 1,
            "name": "Admin",
            "guard_name": "api",
            "created_at": "2019-02-27 09:31:39",
            "updated_at": "2019-02-27 09:31:39",
            "permissions": []
        },
        "rolePermissions": [],
        "allowedPermissions": []
    }
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


Status: `The given data was invalid` StatusCode: `422` Response: `JSON`


```
{
    "message": "The given data was invalid.",
    "errors": {
        "id": [
            "Data not found"
        ]
    }
}
```


---

store():

Új `role` rögzítése. A rögzítés több lépésben történik tranzakciókezelés használatával.

Rögzítés menete:

- a `role` alapadatinak rögzítése a `roles` táblába
- a `role` -hoz tartozó `permission` -ok rögzítése a `role_has_permissions` táblába

Method	Endpoint	Headers
`POST`	`api/admin/roles`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`name`	`string`	`required|string`	Name of role
`guard_name`	`string`	`required|string`	Type of guard
`permissions`	`array`	`required`	List of attached permissions

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

update():

Meglévő `role` adatainak módosítása. A módosítás több lépésben történik tranzakciókezelés használatával.

Módosítás menete:

- a `role` alapadatinak módosítása a `roles` táblába
- a `role` -hoz tartozó által `permission` -ök frissítése

Method	Endpoint	Headers
`PUT`	`api/admin/roles/{id}`	Alapértelmezett

Param	Type	Status	Description
`role_id`	`integer`	`required|integer`	Role ID
`name`	`string`	`required|string`	Name of role
`guard_name`	`string`	`required|string`	Type of guard
`permissions`	`array`	`required`	List of attached permissions

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

destroy():

Meglévő `role` törlése az adatbázisból. A törlés több lépésben történik tranzakciókezelés használatával. Adott `role` törlése végleges, tehát nem visszaállítható!

Módosítás menete:

- a `role` alapadatinak törlése a `roles` táblába
- a `role` -hoz tartozó által `permission` -ök törlése

Method	Endpoint	Headers
`DELETE`	`api/admin/roles/{id}`	Alapértelmezett

Param	Type	Status	Description
`role_id`	`integer`	`required|integer`	Role ID

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/admin/roles/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
    {
        return [];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/admin/roles/` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

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

    /**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "name"          => "required|string",
            "guard_name"    => "required|string",
            "permissions"   => "required"
        ];
    }
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[PUT] api/admin/roles/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "id"            => "required",
            "name"          => "required|string",
            "guard_name"    => "required|string",
            "permissions"   => "required|array"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
```

```
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

DeleteRequest:

Az `DeleteRequest` file tartalmazza az `[DELETE] api/admin/roles/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "id" => "required|integer"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->id,
                $with_trashed    = false,
                $with_paginate   = false,
                $with_relation   = false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

- **Admin/Customers**

Függvények:

80%

index():

Az összes (aktív és törölt állapotban lévő) `customer` típusú felhasználó listázása. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/customers`	Alapértelmezett
`GET`	`api/admin/customers?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
"success": {
    "current_page": 1,
        "data": [
            {
            "id": 1,
            "user_id": 1,
            "first_name": "John",
            "last_name": "Doe",
            "mobile_phone": "36201234567",
            "wired_phone": null,
            "home_address_id": 1,
            "home_address_door_number": "2/5",
            "postal_address_id": 2,
            "postal_address_door_number": "1/4",
            "created_at": "2019-01-03 07:57:19",
            "updated_at": "2019-01-03 07:57:19",
            "deleted_at": null,
            "home_address": {
                "id": 1,
                "postal_code": 1234,
                "city": "Sample city",
                "street": "Sample street",
                "street_number": "1",
                "created_at": "2019-01-03 07:57:19",
                "updated_at": "2019-01-03 07:57:19",
                "deleted_at": null
            },
            "postal_address": {
                "id": 2,
                "postal_code": 1234,
                "city": "Sample2 city",
                "street": "Sample2 street",
                "street_number": "1",
                "created_at": "2019-01-03 07:57:19",
                "updated_at": "2019-01-03 07:57:19",
                "deleted_at": null
            }
        }
        ],
    "first_page_url": "http://localhost:8000/api/customers?page=1",
    "from": 1,
    "last_page": 1,
    "last_page_url": "http://localhost:8000/api/customers?page=1",
    "next_page_url": null,
    "path": "http://localhost:8000/api/customers",
    "per_page": 15,
    "prev_page_url": null,
    "to": 3,
    "total": 3
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

show():

Adott (aktív és törölt állapotban lévő) `customer` típusú felhasználó listázása. A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/customers/{id}`	Alapértelmezett
`GET`	`api/admin/customers/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Customer not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": {
        "id": 1,
        "user_id": 1,
        "first_name": "John",
        "last_name": "Doe",
        "mobile_phone": "36201234567",
        "wired_phone": null,
        "home_address_id": 1,
        "home_address_door_number": "2/5",
        "postal_address_id": 2,
        "postal_address_door_number": "1/4",
        "created_at": "2019-01-31 12:13:14",
        "updated_at": "2019-01-31 12:13:14",
        "deleted_at": null,
        "user": {
            "id": 1,
            "email": "sample.user@example.com",
            "email_verified_at": "2019-01-31 12:13:14",
            "active": 1,
            "default_language": "hu",
            "created_at": "2019-01-31 12:13:14",
            "updated_at": "2019-01-31 12:13:14",
            "deleted_at": null
        },
        "home_address": {
            "id": 1,
            "postal_code": 1234,
            "city": "Sample city",
            "street": "Sample street",
            "street_number": "11",
            "created_at": "2019-01-31 12:13:14",
            "updated_at": "2019-01-31 12:13:14",
            "deleted_at": null
        },
        "postal_address": {
            "id": 2,
            "postal_code": 6723,
            "city": "Sample city2",
            "street": "Sample street2",
            "street_number": "2",
            "created_at": "2019-01-31 12:13:14",
            "updated_at": "2019-01-31 12:13:14",
            "deleted_at": null
        }
    }
}
```

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

store():

Új Customer típusú felhasználó rögzítése. A rögzítés több lépésben történik tranzakciókezelés használatával.

Rögzítés menete:

- a felhasználó alapadatinak rögzítése a user táblába
- a felhasználó customer típusú adatainak rögzítése a customer táblába
- a felhasználó által megadott címek rögzítése az address táblába

Method	Endpoint	Headers
`POST`	`api/admin/customers`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
id	integer	required	Azonosító
email	string	required|unique:users	Postafiók
email_confirmation	string	required|same:email	Postafiók ismét
role_id	integer	required	Jogosultsági csoport
password	string	required|string|min:8|confirmed|regex :/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]) (?=.*?[#?!@$%^&*-]).{6,}$/	Jelszó
password_confirmation	string	required|same:password	Jelszó ismét
first_name	string	required	Keresztnév
last_name	string	required	Vezetkénév
wired_phone	string	optional	Vezetékes telefon
mobile_phone	string	required	Mobil telefon
home_address_postal_code	integer	required	Alapértelmezett cím irányítószám
home_address_city	string	required	Alapértelmezett cím város
home_address_street	string	required	Alapértelmezett cím utca
home_address_street_number	string	required	Alapértelmezett cím házszám
home_address_door_number	string	optional	Alapértelmezett cím emelet / ajtó
postal_address_postal_code	integer	required	Levelezési cím irányítószám
postal_address_city	string	required	Levelezési cím város
postal_address_street	string	required	Levelezési cím utca
postal_address_street_number	string	required	Levelezési cím házszám
postal_address_door_number	string	optional	Levelezési cím emelet / ajtó

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

updateStatus():

Meglévő Customer típusú felhasználó státuszának módosítása. A státusz aktív vagy inaktív lehet. inkatív státusz esetén a felhasználó nem tud bejelentkezni a garNet rendszerbe.

Method	Endpoint	Headers
`PUT`	`api/admin/customers/{id}/status`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`id`	`integer`	`required`	Azonosító
`status`	`boolean`	`required`	Állapot

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/admin/customers/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
{
    return [];
}

/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        $data = $this->model->findUserById(
            $this->id,
            true
        );

        if (!$data || $data->count() == 0) {
            $validator->errors()->add('id', 'Data not found');
        }
    });
    return;
}
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/admin/customers/` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```php
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "email"                       => "required|email|unique:users",
            "email_confirmation"          => "required|same:email",
            "role_id"                     => "required|integer",
            "password"                    => "required|string|min:8|confirmed|regex:/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$%^&*-]).{6,}$/"
            "password_confirmation"       => "required|same:password",
            "first_name"                  => "required|string",
            "last_name"                   => "required|string",
            "wired_phone"                 => "sometimes|string",
            "mobile_phone"                => "required|string",
            "home_address_postal_code"    => "required|integer",
            "home_address_city"           => "required|string",
            "home_address_street"         => "required|string",
            "home_address_street_number"  => "required|string",
            "home_address_door_number"    => "sometimes|string",
            "postal_address_postal_code"  => "required|integer",
            "postal_address_city"         => "required|string",
            "postal_address_street"       => "required|string",
            "postal_address_street_number" => "required|string",
            "postal_address_door_number"  => "sometimes|string"
        ];
    }
```

A `transformUserData()` és `transformCustomerData()` függvények segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

```
/**
     * Filter UserData
     *
     * @return  array
     */
    public function transformUserData($request)
    {
        $validated  = $request->validated();
        $data       = [];

        $data["email"]      = $validated["email"];
        $data["password"]   = bcrypt($validated['password']);
        $data["active"]     = 1;

        return $data;
    }

    /**
     * Filter CustomerData
     *
     * @return  array
     */
    public function transformCustomerData($request, $homeAddress, $postalAddress)
    {
        $validated  = $request->validated();
        $data       = [];

        $data["first_name"]                 = $validated["first_name"];
        $data["last_name"]                  = $validated["last_name"];
        $data["wired_phone"]                = isset($validated["wired_phone"]) && $validated["wired_phone"] != "" ?
        $data["mobile_phone"]               = isset($validated["mobile_phone"]) && $validated["mobile_phone"] != ""
        $data["home_address_id"]            = $homeAddress->id;
        $data["home_address_door_number"]   = isset($validated["home_address_door_number"]) && $validated["home_addr
        $data["postal_address_id"]          = $postalAddress->id;
        $data["postal_address_door_number"] = isset($validated["postal_address_door_number"]) && $validated["postal_

        return $data;
    }
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[POST] api/admin/customers/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```php
/**
    * Get the validation rules that apply to the request.
    *
    * @return  array
    */
   public function rules()
   {
       return [
           "user_id"                     => "required|integer",
           "email"                       => "sometimes|email|confirmed",
           "password"                    => "sometimes|string|min:8|confirmed|regex:/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[
           "first_name"                  => "required|string",
           "last_name"                   => "required|string",
           "wired_phone"                 => "sometimes|string",
           "mobile_phone"                => "required|string",
           "home_address_postal_code"    => "required|integer",
           "home_address_city"           => "required|string",
           "home_address_street"         => "required|string",
           "home_address_street_number"  => "required|string",
           "home_address_door_number"    => "sometimes|string",
           "postal_address_postal_code"  => "required|integer",
           "postal_address_city"         => "required|string",
           "postal_address_street"       => "required|string",
           "postal_address_street_number" => "required|string",
           "postal_address_door_number"  => "sometimes|string"
       ];
   }

   /**
    * Configure the validator instance.
    *
    * @param    \Illuminate\Validation\Validator  $validator
    * @return  void
    */
   public function withValidator($validator)
   {
       // checks user current password
       // before making changes
       $validator->after(function ($validator) {
           if ($this->id != $this->user_id) {
               return $validator->errors()->add('id', 'Security incident');
           }

           $data = $this->model->findUserById(
               $this->id,
               false
           );
```

```
            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

A `transformUserData()` és `transformCustomerData()` függvények segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

UpdateStatusRequest:

Az `UpdateStatusRequest` file tartalmazza az `[PUT] api/admin/customers/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "user_id"    => "required|integer",
            "active"     => "required|boolean"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
```

```
        // checks user current password
        // before making changes
        $validator->after(function ($validator) {
            if ($this->id != $this->user_id) {
                return $validator->errors()->add('id', 'Security incident');
            }

            $data = $this->model->findUserById(
                $this->user_id,
                true
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

A `transformUserData()` függvény segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

```
/**
     * Filter UserData
     *
     * @return  array
     */
    public function transformUserData($request)
    {
        $validated  = $request->validated();
        $data       = [];

        $data["active"]     = $validated['active'];
        $data["deleted_at"] = isset($validated['active']) && $validated['active'] == true ? null : Carbon::now();
        return $data;
    }
```

- **Admin/Sellers**

Függvények:

20%

index():

Az összes (aktív és törölt állapotban lévő) `seller` típusú felhasználó listázása. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/sellers`	Alapértelmezett
`GET`	`api/admin/sellers?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "user_id": 4,
            "name": "Sample Seller 1 Kft.",
            "tax_number": "12345678-12-1",
            "registration_number": "1234567-AB",
            "home_address_id": 1,
            "home_address_door_number": "2/3",
            "site_address_id": 2,
            "site_address_door_number": "3/4",
            "postal_address_id": 3,
            "postal_address_door_number": "4/5",
            "contact_id": 1,
            "authenticated_seller": 1,
            "created_at": "2019-02-19 10:46:24",
            "updated_at": "2019-02-19 10:46:24",
            "deleted_at": null
        },
        {
            "id": 2,
            "user_id": 5,
            "name": "Sample Seller 2 Kft.",
            "tax_number": "12345678-99-1",
            "registration_number": "1234567-AW",
            "home_address_id": 1,
            "home_address_door_number": "2/3",
            "site_address_id": 2,
            "site_address_door_number": "3/4",
            "postal_address_id": 3,
            "postal_address_door_number": "4/5",
            "contact_id": 1,
            "authenticated_seller": 1,
            "created_at": "2019-02-19 10:46:24",
            "updated_at": "2019-02-19 10:46:24",
            "deleted_at": null
        },
        {
```

```
            "id": 3,
            "user_id": 6,
            "name": "Sample Seller 3 Kft.",
            "tax_number": "12345678-54-3",
            "registration_number": "1234567-XY",
            "home_address_id": 3,
            "home_address_door_number": "5/13",
            "site_address_id": 2,
            "site_address_door_number": "3/24",
            "postal_address_id": 1,
            "postal_address_door_number": "9/55",
            "contact_id": 2,
            "authenticated_seller": 1,
            "created_at": "2019-02-19 10:46:24",
            "updated_at": "2019-02-19 10:46:24",
            "deleted_at": null
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

Adott (aktív és törölt állapotban lévő) `seller` típusú felhasználó listázása. A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/sellers/{id}`	Alapértelmezett
`GET`	`api/admin/sellers/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
The given data was invalid	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "user_id": 4,
            "name": "Sample Seller 1 Kft.",
            "tax_number": "12345678-12-1",
            "registration_number": "1234567-AB",
            "home_address_id": 1,
            "home_address_door_number": "2/3",
            "site_address_id": 2,
            "site_address_door_number": "3/4",
            "postal_address_id": 3,
            "postal_address_door_number": "4/5",
            "contact_id": 1,
            "authenticated_seller": 1,
            "created_at": "2019-02-19 10:46:24",
            "updated_at": "2019-02-19 10:46:24",
            "deleted_at": null
        }
    ]
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `The given data was invalid` StatusCode: `422` Response: `JSON`

```
{
    "message": "The given data was invalid.",
    "errors": {
        "id": [
            "Data not found"
        ]
    }
}
```

store():

Új Seller típusú felhasználó rögzítése. A rögzítés több lépésben történik tranzakciókezelés használatával.

Rögzítés menete:

- a felhasználó alapadatinak rögzítése a user táblába
- a felhasználó seller típusú adatainak rögzítése a customer táblába
- a felhasználó által megadott címek rögzítése az address táblába

Method	Endpoint	Headers
`POST`	`api/admin/sellers`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
email	string	required|unique:users	Postafiók
email_confirmation	string	required|same:email	Postafiók ismét
role_id	integer	required	Jogosultsági csoport
password	string	required|string|min:8|confirmed|regex :/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]) (?=.*?[#?!@$%^&*-]).{6,}$/	Jelszó
password_confirmation	string	required|same:password	Jelszó ismét
name	string	required	Eladó név
tax_number	string	required	Adószám
registration_number	string	optional	Cégjegyzékszám
home_address_postal_code	integer	required	Alapértelmezett cím irányítószám
home_address_city	string	required	Alapértelmezett cím város
home_address_street	string	required	Alapértelmezett cím utca
home_address_street_number	string	required	Alapértelmezett cím házszám
home_address_door_number	string	optional	Alapértelmezett cím emelet / ajtó
site_address_postal_code	integer	required	Központ cím irányítószám
site_address_city	string	required	Központ cím város
site_address_street	string	required	Központ cím utca
site_address_street_number	string	required	Központ cím házszám
site_address_door_number	string	optional	Központ cím emelet / ajtó
postal_address_postal_code	integer	required	Levelezési cím irányítószám

`postal_address_city`	`string`	`required`	Levelezési cím város
`postal_address_street`	`string`	`required`	Levelezési cím utca
`postal_address_street_number`	`string`	`required`	Levelezési cím házszám
`postal_address_door_number`	`string`	`optional`	Levelezési cím emelet / ajtó
`contact_person_name`	`string`	`optional`	Kapcsolattartó név
`contact_person_email`	`string`	`optional`	Kapcsolattartó email
`contact_person_mobile_phone`	`string`	`optional`	Kapcsolattartó mobiltelefon
`contact_person_wired_phone`	`string`	`optional`	Kapcsolattartó vezetékes telefon
`authenticated_seller`	`boolean`	`requires`	Hitelesített eladó

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

update():

Meglévő Seller típusú felhasználó adatainak módosítása. A módosítás több lépésben történik tranzakciókezelés használatával.

Módosítás menete:

- a felhasználó alapadatinak módosítása a user táblába
- a felhasználó seller típusú adatainak módosítása a customer táblába
- a felhasználó által megadott címek módosítása az address táblába

Method	Endpoint	Headers
`PUT`	`api/admin/sellers/{id}`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
user_id	integer	required	ID
email	string	required|same:email	Postafiók
password	string	required|string|min:8|confirmed|regex :/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9]) (?=.*?[#?!@$%^&*-]).{6,}$/	Jelszó
name	string	required	Eladó név
tax_number	string	required	Adószám
registration_number	string	optional	Cégjegyzékszám
home_address_postal_code	integer	required	Alapértelmezett cím irányítószám
home_address_city	string	required	Alapértelmezett cím város
home_address_street	string	required	Alapértelmezett cím utca
home_address_street_number	string	required	Alapértelmezett cím házszám
home_address_door_number	string	optional	Alapértelmezett cím emelet / ajtó
site_address_postal_code	integer	required	Központ cím irányítószám
site_address_city	string	required	Központ cím város
site_address_street	string	required	Központ cím utca
site_address_street_number	string	required	Központ cím házszám
site_address_door_number	string	optional	Központ cím emelet / ajtó
postal_address_postal_code	integer	required	Levelezési cím irányítószám
postal_address_city	string	required	Levelezési cím város
postal_address_street	string	required	Levelezési cím utca

`postal_address_street_number`	`string`	`required`	Levelezési cím házszám
`postal_address_door_number`	`string`	`optional`	Levelezési cím emelet / ajtó
`contact_person_name`	`string`	`optional`	Kapcsolattartó név
`contact_person_email`	`string`	`optional`	Kapcsolattartó email
`contact_person_mobile_phone`	`string`	`optional`	Kapcsolattartó mobiltelefon
`contact_person_wired_phone`	`string`	`optional`	Kapcsolattartó vezetékes telefon
`authenticated_seller`	`boolean`	`requires`	Hitelesített eladó

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "User not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/admin/sellers/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```php
public function rules()
{
    return [];
}

/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        $data = $this->model->findUserById(
            $this->id,
            true
        );

        if (!$data || $data->count() == 0) {
            $validator->errors()->add('id', 'Data not found');
        }
    });
    return;
}
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/admin/sellers/` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```php
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "email"                         => "required|email",
            "email_confirmation"            => "required|same:email",
            "role_id"                       => "required|integer",
            "password"                      => "required|string|min:8|confirmed|regex:/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0
            "password_confirmation"         => "required|same:password",
            "name"                          => "required|string",
            "tax_number"                    => "required|string",
            "registration_number"           => "required|string",
            "home_address_postal_code"      => "required|integer",
            "home_address_city"             => "required|string",
            "home_address_street"           => "required|string",
            "home_address_street_number"    => "required|string",
            "postal_address_door_number"    => "sometimes|string",
            "contact_person_name"           => "sometimes|string",
            "contact_person_email"          => "sometimes|email",
            "contact_person_mobile_phone"   => "sometimes|string",
            "contact_person_wired_phone"    => "sometimes|string",
            "authenticated_seller"          => "required|boolean"
        ];
    }
```

A `transformUserData()` és `transformSellerData()` függvények segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

```
/**
     * Filter UserData
     *
     * @return  array
     */
    public function transformUserData($request)
    {
        $validated  = $request->validated();
        $data       = [];

        $data["email"]      = $validated["email"];
        $data["password"]   = bcrypt($validated['password']);
        $data["active"]     = 1;

        return $data;
    }

    /**
     * Filter SellerData
     *
     * @return  array
     */
    public function transformSellerData($request, $homeAddress, $siteAddress, $postalAddress, $contactPerson)
    {
        $validated  = $request->validated();
        $data       = [];

        $data["name"]                           = $validated["name"];
        $data["tax_number"]                     = $validated["tax_number"];
        $data["registration_number"]            = $validated["registration_number"];
        $data["home_address_id"]                = $homeAddress->id;
        $data["home_address_door_number"]       = isset($validated["home_address_door_number"]) && $validated["home_addr
        $data["site_address_id"]                = $siteAddress->id;
        $data["site_address_door_number"]       = isset($validated["site_address_door_number"]) && $validated["site_addr
        $data["postal_address_id"]              = $postalAddress->id;
        $data["postal_address_door_number"]     = isset($validated["postal_address_door_number"]) && $validated["postal_
        $data["contact_id"]                     = isset($contactPerson) && $contactPerson != null ? $contactPerson->id :
        $data["authenticated_seller"]           = $validated["authenticated_seller"];

        return $data;
    }
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[PUT] api/admin/sellers/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "user_id"                       => "required|integer",
            "email"                         => "sometimes|email|confirmed",
            "password"                      => "sometimes|string|min:8|confirmed|regex:/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[
            "name"                          => "required|string",
            "tax_number"                    => "required|string",
            "registration_number"           => "required|string",
            "home_address_postal_code"      => "required|integer",
            "home_address_city"             => "required|string",
            "home_address_street"           => "required|string",
            "home_address_street_number"    => "required|string",
            "home_address_door_number"      => "sometimes|string",
            "site_address_postal_code"      => "required|integer",
            "site_address_city"             => "required|string",
            "site_address_street"           => "required|string",
            "site_address_street_number"    => "required|string",
            "site_address_door_number"      => "sometimes|string",
            "postal_address_postal_code"    => "required|integer",
            "postal_address_city"           => "required|string",
            "postal_address_street"         => "required|string",
            "postal_address_street_number"  => "required|string",
            "postal_address_door_number"    => "sometimes|string",
            "contact_id"                    => "sometimes|integer",
            "contact_person_name"           => "sometimes|string",
            "contact_person_email"          => "sometimes|email",
            "contact_person_mobile_phone"   => "sometimes|string",
            "contact_person_wired_phone"    => "sometimes|string",
            "authenticated_seller"          => "required|boolean"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
```

```
        // checks user current password
        // before making changes
        $validator->after(function ($validator) {
            if ($this->id != $this->user_id) {
                return $validator->errors()->add('id', 'Security incident');
            }

            $data = $this->model->findUserById(
                $this->user_id,
                true
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

A `transformUserData()` és `transformCustomerData()` függvények segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

UpdateStatusRequest:

Az `UpdateStatusRequest` file tartalmazza az `[PUT] api/admin/sellers/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
/**
     * Get the validation rules that apply to the request.
     *
     * @return  array
     */
    public function rules()
    {
        return [
            "user_id"    => "required|integer",
            "active"     => "required|boolean"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {
```

```
        // checks user current password
        // before making changes
        $validator->after(function ($validator) {
            if ($this->id != $this->user_id) {
                return $validator->errors()->add('id', 'Security incident');
            }

            $data = $this->model->findUserById(
                $this->user_id,
                true
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', 'Data not found');
            }
        });
        return;
    }
```

A `transformUserData()` függvény segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

```
/**
     * Filter UserData
     *
     * @return  array
     */
    public function transformUserData($request)
    {
        $validated  = $request->validated();
        $data       = [];

        $data["active"]     = $validated['active'];
        $data["deleted_at"] = isset($validated['active']) && $validated['active'] == true ? null : Carbon::now();
        return $data;
    }
```

- **Admin/Forms**

Függvények:

100%

index():

Az összes (aktív és törölt állapotban lévő) `formItem` típusú adat listázása. A garNet rendszerben rögzíthető termékek rendelkeznek alap és extra property-kel. Alap property-knek nevezzük azokat az property-ket amelyekkel minden termék rendelkezik. Bizonyos esetbekben (pl. mobiltelefon esetében) azonban szükség van extra adatok megadására is. Ezeket nevezzük extra propertyk-nek. A form modul ezen property-k adminisztrációját kezeli. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/formItems`	Alapértelmezett
`GET`	`api/admin/formItems?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "product_category_id": 21,
            "name": "Serial Number",
            "description": "Serial number description",
            "created_at": "2019-04-02 12:08:44",
            "updated_at": "2019-04-02 12:08:44",
            "deleted_at": null,
            "form_item": [
                {
                    "id": 1,
                    "form_group_id": 1,
                    "type": "text",
                    "name": "textTypeItemName",
                    "label": "textTypeItemFieldLabelName",
                    "value": null,
                    "created_at": "2019-04-02 12:08:44",
                    "updated_at": "2019-04-02 12:08:44",
                    "deleted_at": null
```

```
            }
        ]
    },
    {
        "id": 2,
        "product_category_id": 38,
        "name": "IMEI Number",
        "description": "IMEI number description",
        "created_at": "2019-04-02 12:08:44",
        "updated_at": "2019-04-02 12:08:44",
        "deleted_at": null,
        "form_item": [
            {
                "id": 2,
                "form_group_id": 2,
                "type": "text",
                "name": "imei_number",
                "label": "imei_number",
                "value": null,
                "created_at": "2019-04-02 12:08:44",
                "updated_at": "2019-04-02 12:08:44",
                "deleted_at": null
            }
        ]
    }
]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

Adott (aktív és törölt állapotban lévő) `formItem` típusú adat listázása.A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/formItems/{id}`	Alapértelmezett
`GET`	`api/admin/formItems/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Data not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "product_category_id": 21,
            "name": "Serial Number",
            "description": "Serial number description",
            "created_at": "2019-04-02 12:08:44",
            "updated_at": "2019-04-02 12:08:44",
            "deleted_at": null,
            "form_item": [
                {
                    "id": 1,
                    "form_group_id": 1,
                    "type": "text",
                    "name": "textTypeItemName",
                    "label": "textTypeItemFieldLabelName",
                    "value": null,
                    "created_at": "2019-04-02 12:08:44",
                    "updated_at": "2019-04-02 12:08:44",
                    "deleted_at": null
                }
            ]
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


Status: `not found` StatusCode: `404` Response: `JSON`


```
{
    "error": "Data not found."
}
```

store():

Új formItem típusú adat rögzítése. A rögzítés több lépésben történik tranzakciókezelés használatával.

Rögzítés menete:

- a formItem alapadatinak rögzítése a form_groups táblába
- a formItem típusú bejegyzés adatainak rögzítése a form_items táblába

Method	Endpoint	Headers
`POST`	`api/admin/formItems`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`form_group_product_subcategory_id`	`integer`	`required`	Form group termék azonosító
`form_group_name`	`string`	`required`	Form group megnezevése
`form_group_description`	`integer`	`required`	Form group megjegyzés
`form_item_type`	`integer`	`required`	Form elem típus
`form_item_name`	`string`	`required`	Form elem megnevezés
`form_item_label`	`string`	`required`	Form elem cimke

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

update():

Meglévő formItem típusú adat módosítása. A módosítás több lépésben történik tranzakciókezelés használatával.

Módosítás menete:

- a formItem alapadatinak módosítása a form_groups táblába
- a formItem típusú bejegyzés adatainak módosítása a form_items táblába

Method	Endpoint	Headers
`PUT`	`api/admin/formItems/{id}`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`form_group_id`	`integer`	`required`	Azonosító
`form_group_product_subcategory_id`	`integer`	`required`	Form group termék azonosító
`form_group_name`	`string`	`required`	Form group megnezevése
`form_group_description`	`integer`	`required`	Form group megjegyzés
`form_item_type`	`integer`	`required`	Form elem típus
`form_item_name`	`string`	`required`	Form elem megnevezés
`form_item_label`	`string`	`required`	Form elem cimke

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

updateStatus():

Meglévő formItem típusú adat státuszának módosítása. A státusz aktív vagy inaktív lehet.

Method	Endpoint	Headers
`PUT`	`api/admin/formItems/{id}/status`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`form_group_id`	`integer`	`required`	Azonosító
`active`	`boolean`	`required`	Állapot

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/admin/formItems/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```php
public function rules()
{
    return [];
}

/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        $data = $this->model->getById(
            $this->id
        );

        if (!$data || $data->count() == 0) {
            $validator->errors()->add('id', trans("messages.data not found"));
        }
    });
    return;
}
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/admin/formItems/` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
{
    return [
        "form_group_product_subcategory_id" => "required",
        "form_group_name"                   => "required",
        "form_group_description"            => "required",
        "form_item_type"                    => "required",
        "form_item_name"                    => "required",
        "form_item_label"                   => "required"
    ];
}
```

A `transformFormGroupData()` és `transformFormItemData()` függvények segítségével a küldött adatok formázása / átalakítása történik a megfelelő formára.

```
public function transformFormGroupData($request)
{
    $validated  = $request->validated();
    $data       = [];

    $data["product_category_id"]    = $validated["form_group_product_subcategory_id"];
    $data["name"]                   = $validated["form_group_name"];
    $data["description"]            = $validated["form_group_description"];

    return $data;
}

public function transformFormItemData($request, $form_group)
{
    $validated  = $request->validated();
    $data       = [];

    $data["form_group_id"]  = $form_group->id;
    $data["type"]           = $validated["form_item_type"];
    $data["name"]           = $validated["form_item_name"];
    $data["label"]          = $validated["form_item_label"];
    $data["value"]          = isset($validated["form_item_value"]) && $validated["form_item_value"] != "" ? $validated["form_item_value"] : null;

    return $data;
}
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[POST] api/admin/formItems/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
{
    return [
        "form_group_id"                     => "required",
        "form_group_product_subcategory_id" => "required",
        "form_group_name"                   => "required",
        "form_group_description"            => "required",
        "form_item_type"                    => "required",
        "form_item_name"                    => "required",
        "form_item_label"                   => "required"
    ];
}

public function withValidator($validator)
{
```

```
        // checks user current password
        // before making changes
        $validator->after(function ($validator) {
            if ($this->id != $this->form_group_id) {
                return $validator->errors()->add('id', trans("messages.security incident"));
            }

            $data = $this->model->getById(
                $this->form_group_id,
                false
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', trans("messages.data not found"));
            }
        });
        return;
    }
```

UpdateStatusRequest:

Az `UpdateStatusRequest` file tartalmazza az `[PUT] api/admin/formItems/{id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
    public function rules()
    {
        return [
            "form_group_id" => "required",
            "active"        => "required"
        ];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
    public function withValidator($validator)
    {

        $validator->after(function ($validator) {
            $data = $this->model->getById(
                $this->form_group_id
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', trans("messages.data not found"));
            }
        });
        return;
    }
```

- **Admin/Product categories**

Függvények:

100%

index():

Az összes `productCategory` listázása adminisztráció céljából. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/products`	Alapértelmezett
`GET`	`api/admin/products?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
  "success": [
      {
          "id": 1,
          "name": "Sample product1",
          "parent_id": null,
          "created_at": "2019-04-02 12:08:44",
          "updated_at": "2019-04-02 12:08:44",
          "deleted_at": null
      },
      {
          "id": 2,
          "name": "Sample product2",
          "parent_id": 1,
          "created_at": "2019-04-02 12:08:44",
          "updated_at": "2019-04-02 12:08:44",
          "deleted_at": null
      },
      {
```

```
            "id": 3,
            "name": "Sample product3",
            "parent_id": 1,
            "created_at": "2019-04-02 12:08:44",
            "updated_at": "2019-04-02 12:08:44",
            "deleted_at": null
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

Adott (aktív és törölt állapotban lévő) `productCategory` típusú adat listázása ID alapján. A garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot. Az alapértelmezett listázás `paginate` nélkül kerül megvalósításra. A `withPaginate` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten olyan formátumban adja vissza, ami tartalmaz minden szükséges adatot egy paginator létrehozásához. A `withTrashed` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten a törölt adatokkal együtt adja vissza. A `withRelation` query modifier használatával a garNet backend rendszer az eredményt alapértelmezetten az adott modellhez kapcsolódó relációkkal együtt adja vissza.

Method	Endpoint	Headers
`GET`	`api/admin/products/{id}`	Alapértelmezett
`GET`	`api/admin/products/{id}?withTrashed=true&withePaginate=true&withRelation=true`	With query modifiers

Query parameters:

Param	Type	Default	Status	Description
`id`	`integer`	---	`required`	ID
`$withTrashed`	`boolean`	`false`	`optional`	query modifier (add softDeleted rows to result)
`$withPaginate`	`boolean`	`false`	`optional`	query modifier (return with paginated result)
`$withRelation`	`boolean`	`false`	`optional`	query modifier (add relation(s) to result)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Data not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "name": "Sapmle product1",
            "parent_id": null,
            "created_at": "2019-05-08 09:50:13",
            "updated_at": "2019-05-08 09:50:13",
            "deleted_at": null
        }
    ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


Status: `not found` StatusCode: `404` Response: `JSON`


```
{
    "error": "Data not found."
}
```


store():

Új ProductCategory típusú adat rögzítése. A rögzítés tranzakciókezelés használatával történik.

Rögzítés menete:

- a productCategory típusú adat rögzítése a `product_category` táblába

Method	Endpoint	Headers
`POST`	`api/admin/products`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`name`	`string`	`required`	Category name
`parent_id`	`integer`	`required`	Parent Id

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

update():

Meglévő productCategory típusú adat módosítása. A módosítás tranzakciókezelés használatával történik.

Módosítás menete:

- a productCategory típusú adat módosítása a `product_category` táblába

Method	Endpoint	Headers
`PUT`	`api/admin/products/{id}`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`id`	`integer`	`required|integer`	Item Id
`name`	`string`	`required`	Category name
`parent_id`	`integer`	`required|integer`	Parent Id

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

updateStatus():

Meglévő ProductCategory típusú adat státuszának módosítása. A státusz aktív vagy inaktív lehet. inkatív státusz esetén nem kerül alapértlemezetten listázva.

Method	Endpoint	Headers
`PUT`	`api/admin/products/{id}/status`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`id`	`integer`	`required`	Azonosító
`active`	`boolean`	`required`	Állapot

Handled responses:

Status	StatusCode
Success	200
Unauthorized	401
Not found	404
Security error	422
Internal Server Error	500

Response:

Status: success StatusCode: 200 Response: JSON

```
{
    "success": true
}
```

Status: unauthenticated StatusCode: 401 Response: JSON

```
{
    "message": "Unauthenticated."
}
```

Status: not found StatusCode: 404 Response: JSON

```
{
    "error": "Data not found."
}
```

Status: securtiy problem StatusCode: 422 Response: JSON

```
{
    "message": "Ids are different, security problem."
}
```

- **Public/Customers**

Függvények:

100%

show():

A bejelentkezett felhasználó saját adatainak megjelenítése (Profil oldalhoz szükséges adatok listázása).

Method	Endpoint	Headers
`GET`	`api/public/customers/{id}/profile`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
"success": [
    {
        "id": 1,
        "user_id": 1,
        "first_name": "Sample first name",
        "last_name": "Sample last name",
        "mobile_phone": "36201234567",
        "wired_phone": null,
        "expire_alert_in_day": 30,
        "home_address_id": 1,
        "home_address_door_number": "2/5",
        "postal_address_id": 2,
        "postal_address_door_number": "1/4",
        "created_at": "2019-05-09 12:09:10",
        "updated_at": "2019-05-09 12:09:10",
        "deleted_at": null,
        "user": {
            "id": 1,
            "email": "sample.user1@identity-hungary.hu",
            "email_verified_at": "2019-05-09 12:09:10",
            "active": 1,
            "default_language": "hu",
```

```
            "created_at": "2019-05-09 12:09:10",
            "updated_at": "2019-05-09 12:09:10",
            "deleted_at": null
        },
        "postal_address": {
            "id": 2,
            "postal_code": 6722,
            "city": "Szeged",
            "street": "Sample utca",
            "street_number": "1",
            "created_at": "2019-05-09 12:09:10",
            "updated_at": "2019-05-09 12:09:10",
            "deleted_at": null
        },
        "home_address": {
            "id": 1,
            "postal_code": 6726,
            "city": "Szeged",
            "street": "Sample2 utca",
            "street_number": "11",
            "created_at": "2019-05-09 12:09:10",
            "updated_at": "2019-05-09 12:09:10",
            "deleted_at": null
        }
    }
  ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


update():

Bejelentkezett felhasználó saját adatainak módosítása. A módosítás tranzakciókezelés használatával történik.

Módosítás menete:

- a felhasználó alapadatinak módosítása a user táblába
- a felhasználó customer típusú adatainak módosítása a customer táblába
- a felhasználó által megadott címek módosítása az address táblába

Method	Endpoint	Headers
`PUT`	`api/public/customer/{id}/profile`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
user_id	integer	required	Azonosító
email	string	required|unique:users	Postafiók
password	string	required|string|min:8|confirmed|regex :/^(?=.*?[A-Z])(?=.*?[a-z])(?=.*?[0-9])(?=.*?[#?!@$%^&*-]).{6,}$/	Jelszó
password_confirmation	string	required|same:password	Jelszó ismét
default_language	string	required	Alapértelmezett nyelv
first_name	string	required	Keresztnév
last_name	string	required	Vezetkénév
wired_phone	string	optional	Vezetékes telefon
mobile_phone	string	required	Mobil telefon
home_address_postal_code	integer	required	Alapértelmezett cím irányítószám
home_address_city	string	required	Alapértelmezett cím város
home_address_street	string	required	Alapértelmezett cím utca
home_address_street_number	string	required	Alapértelmezett cím házszám
home_address_door_number	string	optional	Alapértelmezett cím emelet / ajtó
postal_address_postal_code	integer	required	Levelezési cím irányítószám
postal_address_city	string	required	Levelezési cím város
postal_address_street	string	required	Levelezési cím utca
postal_address_street_number	string	required	Levelezési cím házszám
postal_address_door_number	string	optional	Levelezési cím emelet / ajtó

Handled responses:

Status	StatusCode
Success	200
Unauthorized	401
Not found	404
Security error	422
Internal Server Error	500

Response:

Status: success StatusCode: 200 Response: JSON

```
{
    "success": true
}
```

Status: unauthenticated StatusCode: 401 Response: JSON

```
{
    "message": "Unauthenticated."
}
```

Status: not found StatusCode: 404 Response: JSON

```
{
    "error": "User not found."
}
```

Status: securtiy problem StatusCode: 422 Response: JSON

```
{
    "message": "Ids are different, security problem."
}
```

- **Public/Invoices**

Függvények:

100%

index():

A bejelentkezett felhasználó által rögzített számlák listázása.

Method	Endpoint	Headers
`GET`	`api/public/customers/{user_id}/invoices`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
"success": [
    {
        "id": 1,
        "invoice_no": "ABC123/2019",
        "user_id": 1,
        "seller_id": 1,
        "purchased": "2019-05-09",
        "ap_no": "AP12345678",
        "created_at": "2019-05-09 12:09:11",
        "updated_at": "2019-05-09 12:09:11",
        "deleted_at": null
    },
    {
        "id": 2,
        "invoice_no": "ABC12345/2019",
        "user_id": 1,
        "seller_id": 2,
        "purchased": "2019-05-09",
        "ap_no": "AP12345678",
        "created_at": "2019-05-09 12:09:11",
        "updated_at": "2019-05-09 12:09:11",
        "deleted_at": null
    }
]
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

A bejelentkezett felhasználó által rögzített adott számla megjelenítése ID alapján

Method	Endpoint	Headers
`GET`	`api/public/customers/{user_id}/invoices/{invoice_id}`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
 "success": [
     {
         "id": 1,
         "invoice_no": "ABC123/2019",
         "user_id": 1,
         "seller_id": 1,
         "purchased": "2019-05-09",
         "ap_no": "AP12345678",
         "created_at": "2019-05-09 12:09:11",
         "updated_at": "2019-05-09 12:09:11",
         "deleted_at": null
     }
 ]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


store():

Számla típusú adat rögzítése. A rögzítés tranzakciókezelés használatával történik.

Rögzítés menete:

- a számla típusú adat alapadatinak rögzítése a customer_invoice táblába

Method	Endpoint	Headers
`POST`	`api/public/customer/{user_id}/invoices`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
seller_id	string	required|integer	Eladó azonosító
invoice_no	string	required	Számla sorszám
user_id	string	required|integer	Felhazsnáló azonosító
purchased	string	required	Vásárlás dátuma
ap_no	string	sometimes	Pénztárgép azonosító
files	string	required	Csatolt file-ok

Handled responses:

Status	StatusCode
Success	200
Unauthorized	401
Internal Server Error	500

Response:

Status: success StatusCode: 200 Response: JSON

```
{
    "success": true
}
```

Status: unauthenticated StatusCode: 401 Response: JSON

```
{
    "message": "Unauthenticated."
}
```

update():

Számla típusú adat módosítása. A módosítás tranzakciókezelés használatával történik.

Rögzítés menete:

- a számla típusú adat adatinak módosítása a customer_invoice táblába

Method	Endpoint	Headers
PUT	api/public/customer/{user_id}/invoices/{invoice_id}	Alapértelmezett

Body parameters:

Param	Type	Status	Description
seller_id	string	required|integer	Eladó azonosító
invoice_id	string	required|integer	Számla azonosító
invoice_no	string	required	Számla sorszám
user_id	string	required|integer	Felhazsnáló azonosító
purchased	string	required	Vásárlás dátuma
ap_no	string	sometimes	Pénztárgép azonosító

Handled responses:

Status	StatusCode
Success	200
Unauthorized	401
Not found	404
Security error	422
Internal Server Error	500

Response:

Status: success StatusCode: 200 Response: JSON

```
{
    "success": true
}
```

Status: unauthenticated StatusCode: 401 Response: JSON

```
{
    "message": "Unauthenticated."
}
```

Status: not found StatusCode: 404 Response: JSON

```
{
    "error": "Data not found."
}
```

Status: securtiy problem StatusCode: 422 Response: JSON

```
{
    "message": "Ids are different, security problem."
}
```

destroy():

Meglévő számla típusú adat státuszának módosítása. A státusz aktív vagy inaktív lehet. inkatív státusz esetén a számla alapértelmezetten nem kerül listázásra a garNet rendszerbe.

Method	Endpoint	Headers
`PUT`	`api/admin/customers/{id}/status`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`invoice_id`	`integer`	`required`	Számla azonosító
`user_id`	`integer`	`required`	Felhasználó azonosító

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/public/customer/{user_id}/invoices/{invoice_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
    public function rules()
    {
        return [];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */

    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            if ($this->user()->id != $this->user_id) {
                return $validator->errors()->add('id', trans("messages.security incident"));
            }

            $data = $this->model->findByUserIdAndInvoiceId(
                            $this->user()->id,
                            $this->invoice_id
                        );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', trans("messages.data not found"));
            }

        });
        return;
    }
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/public/customer/{user_id}/invoices` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
    {
        $rules = [
            "seller_id"     => "required|integer",
            "invoice_no"    => "required",
            "user_id"       => "required|integer",
            "purchased"     => "required",
            "ap_no"         => "sometimes",
            "files"         => "required"
        ];
        return $rules;
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */
```

```
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        if ($this->user()->id != $this->user_id) {
            return $validator->errors()->add('id', trans("messages.security incident"));
        }
    });
    return;
}
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[PUT] api/public/customer/{user_id}/invoices/{invoice_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
{
    $rules = [
        "seller_id"     => "required|integer",
        "invoice_id"    => "required|integer",
        "invoice_no"    => "required",
        "user_id"       => "required|integer",
        "purchased"     => "required",
        "ap_no"         => "sometimes"
    ];
    return $rules;
}

/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */
public function withValidator($validator)
{
    $validator->after(function ($validator) {
        if ($this->user()->id != $this->user_id) {
            return $validator->errors()->add('id', trans("messages.security incident"));
        }

        $data = $this->model->findByUserIdAndInvoiceId(
            $this->user()->id,
            $this->invoice_id
                    );

        if (!$data || $data->count() == 0) {
            $validator->errors()->add('id', trans("messages.data not found"));
        } elseif ($data->deleted_at != "") {
            $validator->errors()->add('id', trans("messages.data is soft deleted"));
        }
    });
    return;
}
```

deleteRequest:

Az `UpdatestatusRequest` file tartalmazza az `[DELETE] api/public/customer/{user_id}/invoices/{invoice_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
{
    $rules = [
        "invoice_id"    => "required|integer",
        "user_id"       => "required|integer"
    ];
    return $rules;
}

/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */

public function withValidator($validator)
{
    $validator->after(function ($validator) {
        if ($this->user()->id != $this->user_id) {
            return $validator->errors()->add('id', trans("messages.security incident"));
        }


        $data = $this->model->findByUserIdAndInvoiceId(
            $this->user()->id,
            $this->invoice_id
                    );

        if (!$data || $data->count() == 0) {
            $validator->errors()->add('id', trans("messages.data not found"));
        } elseif ($data->deleted_at != "") {
            $validator->errors()->add('id', trans("messages.data is soft deleted"));
        }
    });
    return;
}
```

- **Public/InvoiceItems**

Függvények:

100%

index():

A bejelentkezett felhasználó által rögzített számlákhoz rendelt termékek listázása.

Method	Endpoint	Headers
GET	api/public/customer/{user_id}/invoices/{invoice_id}/items	Alapértelmezett

Handled responses:

Status	StatusCode
Success	200
Unauthorized	401
Internal Server Error	500

Response:

Status: success StatusCode: 200 Response: JSON


```
{
"success": [
    {
        "id": 1,
        "invoice_id": 1,
        "name": "Sample item1",
        "product_category_id": 1,
        "product_subcategory_id": 2,
        "manufacture": "11",
        "type_no": "11",
        "barcode": "111",
        "price": 10000,
        "warranty_in_month": 12,
        "warranty": 0,
        "created_at": "2019-05-14 10:38:44",
        "updated_at": "2019-05-14 10:38:44",
        "deleted_at": null
    },
    {
        "id": 2,
        "invoice_id": 1,
        "name": "Sample item2",
        "product_category_id": 1,
        "product_subcategory_id": 2,
        "manufacture": "11",
        "type_no": "11",
        "barcode": "111",
        "price": 20000,
        "warranty_in_month": 24,
        "warranty": 0,
        "created_at": "2019-05-14 10:38:44",
        "updated_at": "2019-05-14 10:38:44",
        "deleted_at": null
    }
]
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

A bejelentkezett felhasználó által rögzített adott számlához rendelt adott termék megjelenítése ID alapján

Method	Endpoint	Headers
`GET`	`api/public/customer/{user_id}/invoices/{invoice_id}/items/{item_id}`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
"success": [
    {
        "id": 1,
        "invoice_id": 1,
        "name": "Sample item1",
        "product_category_id": 1,
        "product_subcategory_id": 2,
        "manufacture": "11",
        "type_no": "11",
        "barcode": "111",
        "price": 10000,
        "warranty_in_month": 12,
        "warranty": 0,
        "created_at": "2019-05-14 10:38:44",
        "updated_at": "2019-05-14 10:38:44",
        "deleted_at": null
    }
]
}
```

getByUserId():

A bejelentkezett felhasználóhoz rendelt összes termék listázása user azonosító alapján

Method	Endpoint	Headers
`GET`	`api/public/customer/{user_id}/items/`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
    "success": [
        {
            "id": 1,
            "invoice_id": 1,
            "name": "Sample item1",
            "product_category_id": 1,
            "product_subcategory_id": 2,
            "manufacture": "11",
            "type_no": "11",
            "barcode": "23456789",
            "price": 10000,
            "warranty_in_month": 12,
            "warranty": 0,
            "created_at": "2019-05-14 10:38:44",
            "updated_at": "2019-05-14 10:38:44",
            "deleted_at": null
        },
        {
            "id": 2,
            "invoice_id": 2,
            "name": "Sample item2",
            "product_category_id": 1,
            "product_subcategory_id": 2,
            "manufacture": "11",
            "type_no": "11",
            "barcode": "23456789",
            "price": 10000,
            "warranty_in_month": 12,
            "warranty": 0,
            "created_at": "2019-05-14 10:38:44",
            "updated_at": "2019-05-14 10:38:44",
            "deleted_at": null
        }
    ]
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

getExpiredItems():

A bejelentkezett felhasználó által rögzített adott számlához tartozó lejáró termékek listázása.

Method	Endpoint	Headers
`GET`	`api/public/customer/{user_id}/expiredItems`	Alapértelmezett

store():

A bejelentkezett felhasználó által rögzített adott számlához történő termék rögzítése. A rögzítés tranzakciókezelés használatával történik.

Rögzítés menete:

- a számlához tarozó termák típusú adat alapadatinak rögzítése a customer_invoice_item táblába

Method	Endpoint	Headers
`POST`	`api/public/customer/{user_id}/invoices/{invoice_id}/items`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`invoice_id`	`integer`	`required`	Számla azonosító
`name`	`string`	`required`	Termék megnevezés
`product_category_id`	`integer`	`required`	Termék kategória azonosító
`product_subcategory_id`	`integer`	`required`	Termék alkategória azonosító
`manufacture`	`string`	`sometimes`	Gyártó megnevezés
`type_no`	`string`	`required`	Típusszám
`barcode`	`string`	`sometimes`	Vonalkód
`price`	`integer`	`required`	Ár
`warranty_in_month`	`integer`	`required`	Garancia hónapokban
`warranty`	`boolean`	`required`	Feltöltött dokumentum csatolva
`has_extra_form_item`	`boolean`	`required`	Extra form elem

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

update():

A bejelentkezett felhasználó által rögzített adott számlához történő termék adatainak módosítása. A módosítás tranzakciókezelés használatával történik.

Rögzítés menete:

- a számlához tarozó termák típusú adat alapadatinak módosítása a customer_invoice_item táblába

Method	Endpoint	Headers
`PUT`	`customer/{user_id}/invoices/{invoice_id}/items/{item_id}`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`invoice_id`	`integer`	`required`	Számla azonosító
`name`	`string`	`required`	Termék megnevezés
`product_category_id`	`integer`	`required`	Termék kategória azonosító
`product_subcategory_id`	`integer`	`required`	Termék alkategória azonosító
`manufacture`	`string`	`sometimes`	Gyártó megnevezés
`type_no`	`string`	`required`	Típusszám
`barcode`	`string`	`sometimes`	Vonalkód
`price`	`integer`	`required`	Ár
`warranty_in_month`	`integer`	`required`	Garancia hónapokban
`warranty`	`boolean`	`required`	Feltöltött dokumentum csatolva
`has_extra_form_item`	`boolean`	`required`	Extra form elem

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

destroy():

A bejelentkezett felhasználó által rögzített adott számlához történő termék státuszának módosítása. A státusz aktív vagy inaktív lehet. Inkatív státusz esetén a számlához tartozó termék alapértelmezetten nem kerül listázásra a garNet rendszerbe.

Method	Endpoint	Headers
`PUT`	`api/public/customer/{user_id}/invoices/{invoice_id}/items/{item_id}`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`user_id`	`integer`	`required`	Felhasználó azonosító
`invoice_id`	`integer`	`required`	Számla azonosító
`item_id`	`integer`	`required`	Termék azonosító

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Security error	`422`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
    "success": true
}
```

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Status: `not found` StatusCode: `404` Response: `JSON`

```
{
    "error": "Data not found."
}
```

Status: `securtiy problem` StatusCode: `422` Response: `JSON`

```
{
    "message": "Ids are different, security problem."
}
```

Form requests:

100%

ShowRequest:

A `ShowRequest` file tartalmazza az `[GET] api/public/customer/{user_id}/invoices/{invoice_id}/items/{item_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
    {
        return [];
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */

    public function withValidator($validator)
    {
      $validator->after(function ($validator) {
          if ($this->user()->id != $this->user_id) {
              return $validator->errors()->add('id', trans("messages.security incident"));
          }

          $data = $this->invoice->findByUserIdAndInvoiceId(
                          $this->user()->id,
                          $this->invoice_id
                      );

          if (!$data || $data->count() == 0) {
              $validator->errors()->add('id', trans("messages.data not found"));
          }
      });
      return;
    }
```

GetByUserIdRequest:

A `GetByUserIdRequest` file tartalmazza az `[GET] api/public/customer/customer/{user_id}/items` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
    public function rules()
        {
            return [];
        }

        /**
         * Configure the validator instance.
         *
         * @param    \Illuminate\Validation\Validator  $validator
        * @return  void
        */

        public function withValidator($validator)
        {
            $validator->after(function ($validator) {
                if ($this->user()->id != $this->user_id) {
                    return $validator->errors()->add('id', trans("messages.security incident"));
                }
            });
            return;
        }
```

ExpiredItemRequest:

A `ExpiredItemRequest` file tartalmazza az `[GET] api/public/customer/{user_id}/expiredItems` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
{
    return [];
}

/**
 * Configure the validator instance.
 *
 * @param    \Illuminate\Validation\Validator  $validator
 * @return  void
 */

public function withValidator($validator)
{
    $validator->after(function ($validator) {
        if ($this->user()->id != $this->user_id) {
            return $validator->errors()->add('id', trans("messages.security incident"));
        }
    });
    return;
}
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] customer/{user_id}/invoices/{invoice_id}/items` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
public function rules()
{
    $rules = [
        "invoice_id"               => "required|integer",
        "name"                     => "required|string",
        "product_category_id"      => "required|integer",
        "product_subcategory_id"   => "required|integer",
        "manufacture"              => "required|string",
        "type_no"                  => "required|string",
        "barcode"                  => "sometimes|string",
        "price"                    => "required|integer",
        "warranty_in_month"        => "required|integer",
        "warranty"                 => "required",
        "has_extra_form_item"      => "required",
    ];
    return $rules;
}
```

```
/**
* Configure the validator instance.
*
* @param    \Illuminate\Validation\Validator  $validator
* @return  void
*/

public function withValidator($validator)
{
    $validator->after(function ($validator) {
        if ($this->user()->id != $this->user_id) {
            return $validator->errors()->add('id', trans("messages.security incident"));
        }

        if ($this->warranty === "1") {
            if (!isset($this->files) || empty($this->files) || count($this->files) == 0) {
                return $validator->errors()->add('id', trans("messages.files is emtpy"));
            }
        }

        if ($this->has_extra_form_item === "1") {
            $extra_form_items = $this->extraFormItem->getById($this->product_subcategory_id);

            if (!$extra_form_items || count($extra_form_items) == 0) {
                return $validator->errors()->add('has_extra_form_item', trans("messages.missing extra items for this subcategory id"));
            } else {
```

```
                foreach ($extra_form_items as $keys => $items) {
                    foreach ($items->FormItem as $key => $item) {
                        $current_item = $this->{$item['name']};
                        if ($current_item === null) {
                            return $validator->errors()->add('id', trans("messages.item is required"));
                        }
                    }
                }
            }

            $data = $this->invoice->findByUserIdAndInvoiceId(
                $this->user()->id,
                $this->invoice_id
            );

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', trans("messages.data not found"));
            }
        });
        return;
    }
```

UpdateRequest:

Az `UpdateRequest` file tartalmazza az `[PUT] api/public/customer/{user_id}/invoices/{invoice_id}/items/{item_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
    public function rules()
    {
        $rules = [
            "invoice_id"              => "required|integer",
            "name"                    => "required|string",
            "product_category_id"     => "required|integer",
            "product_subcategory_id"  => "required|integer",
            "manufacture"             => "required|string",
            "type_no"                 => "required|string",
            "barcode"                 => "nullable|string",
            "price"                   => "required|integer",
            "warranty_in_month"       => "required|integer",
            "warranty"                => "required|boolean",
            "has_extra_form_item"     => "required"
        ];
        return $rules;
    }
```

```
    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */

    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            if ($this->user()->id != $this->user_id) {
                return $validator->errors()->add('id', trans("messages.security incident"));
            }

            $data = $this->model->findByUserIdAndInvoiceIdAnItemId(
                $this->user()->id,
                $this->invoice_id,
                $this->item_id
                    );

            return $data;

            if (!$data || $data->count() == 0) {
                $validator->errors()->add('id', trans("messages.data not found"));
            } elseif ($data->deleted_at != "") {
                $validator->errors()->add('id', trans("messages.data is soft deleted"));
            }
```

```
        });
        return;
    }
```

deleteRequest:

Az `UpdateStatusRequest` file tartalmazza az `[DELETE] api/public/customer/{user_id}/invoices/{invoice_id}/items/{item_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás után az egyéb ellenőrzések a `withValidator()` függvény segítségével történnek.

```
    public function rules()
    {
        $rules = [
            "user_id"       => "required|integer",
            "invoice_id"    => "required|integer",
            "item_id"       => "required|integer"
        ];
        return $rules;
    }

    /**
     * Configure the validator instance.
     *
     * @param    \Illuminate\Validation\Validator  $validator
     * @return  void
     */

    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            if ($this->user()->id != $this->user_id) {
                return $validator->errors()->add('id', trans("messages.security incident"));
            }
```

```
          $data = $this->model->findByUserIdAndInvoiceIdAnItemId(
              $this->user()->id,
              $this->invoice_id,
              $this->item_id
          );

          if (!$data || $data->count() == 0) {
              $validator->errors()->add('id', trans("messages.data not found"));
          } elseif ($data->deleted_at != "") {
              $validator->errors()->add('id', trans("messages.data is soft deleted"));
          }
      });
      return;
  }
```

- **Public/NotificationLogs**

Függvények:

80%

getExpireProductsByUserId():

A bejelentkezett felhasználóhoz tartozó lejáró termékek értesítésének listája

Method	Endpoint	Headers
`GET`	`api/public/customer/{user_id}/notifications/expire`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

updateReadAt():

A bejelentkezett felhasználó számára kiküldött értesítések olvásának kezelése.

Method	Endpoint	Headers
`POST`	`api/public/customer/{user_id}/notifications/{item_id}/status`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

Status: `unauthenticated` StatusCode: `401` Response: `JSON`

```
{
    "message": "Unauthenticated."
}
```

Form requests:

Nincs szükség fromRequets file(ok)-ra.

- **Public/Uploads**

Függvények:

100%

download():

Adott számlához és / vagy termékhez feltöltött dokumentum letöltése.

Method	Endpoint	Headers
`GET`	`api/public/customer/{user_id}/uploads/{file_id}/download/{disposition}`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
File not exist	`404`
Unauthorized	`401`
Internal Server Error	`500`

store():

Adott számlához és / vagy termékhez feltöltött dokumentum mentése.

Method	Endpoint	Headers
`POST`	`api/public/customer/{user_id}/uploads`	Alapértelmezett

Body parameters:

Param	Type	Status	Description
`invoice_id`	`integer`	`required`	Számla azonosító
`invoice_item_id`	`integer`	`sometimes`	Termék azonosító
`user_id`	`integer`	`required`	Felhasználó azonosító
`files`	`array`	`required`	Feltöltött file(ok)

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Unauthorized	`401`
Internal Server Error	`500`

destroy():

Adott számlához és / vagy termékhez feltöltött dokumentum törlése.

Method	Endpoint	Headers
`DELETE`	`api/public/customer/{user_id}/uploads/{file_id}`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Unauthorized	`401`
Internal Server Error	`500`

Form requests:

100%

DownloadRequest:

A `DownloadRequest` file tartalmazza az `[GET] api/public/customer/{user_id}/uploads/{file_id}/download/{disposition}` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
{
    return [];
}
```

StoreRequest:

A `StoreRequest` file tartalmazza az `[POST] api/public/customer/{user_id}/uploads` végpont meghívásakor küldött paraméter(ek) ellenőrzését. Alapértelmezett validálás nincs, a küldött `ID` paraméter ellenőrzése a `withValidator()` függvény segítségével történik.

```
public function rules()
{
    return [
        "invoice_id"        => "required|integer|min:1",
        "invoice_item_id"   => "sometimes|integer|min:1",
        "user_id"           => "required|integer|min:1",
        "files"             => "required"
    ];
}

/**
* Configure the validator instance.
*
* @param    \Illuminate\Validation\Validator  $validator
* @return  void
*/

public function withValidator($validator)
{
```

```
  $validator->after(function ($validator) {
      if ($this->user()->id != $this->user_id) {
          return $validator->errors()->add('id', trans("messages.security incident"));
      }
  });
  return;
}
```

DeleteRequest:

A `DeleteRequest` file tartalmazza az `[DELETET] api/public/customer/{user_id}/uploads/{file_id}` végpont meghívásakor küldött paraméter(ek) ellenőrzését.

```
    public function rules()
    {
        return [
            "file_id"   => "required|integer|min:1",
            "user_id"   => "required|integer|min:1",
        ];
    }

    /**
    * Configure the validator instance.
    *
    * @param    \Illuminate\Validation\Validator  $validator
    * @return  void
    */

    public function withValidator($validator)
    {
        $validator->after(function ($validator) {
            if ($this->user()->id != $this->user_id) {
                return $validator->errors()->add('id', trans("messages.security incident"));
            }
        });
        return;
    }
```

- **Public/ExtraFormItems**

Függvények:

100%

show():

Adott termékkategóriához tartozó extra form elem listázása

Method	Endpoint	Headers
`GET`	`api/public/extraFormItems/{id}`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`

```
{
"success": [
    {
        "id": 5,
        "product_category_id": 3,
        "name": "Sample 1",
        "description": "Sample 1 description",
        "created_at": "2019-05-29 11:30:37",
        "updated_at": "2019-05-29 11:30:37",
        "deleted_at": null,
        "form_item": [
            {
                "id": 5,
                "form_group_id": 5,
                "type": "text",
                "name": "Sample 1 group",
                "label": "Sample 1 label",
                "value": null,
                "created_at": "2019-05-29 11:30:37",
                "updated_at": "2019-05-29 11:30:37",
                "deleted_at": null
            }
        ]
    }
]
}
```


Form requests:

Nincs szükség fromRequets file(ok)-ra.

- **Public/ProductCategories**

Függvények:

100%

index():

A termékek rögzítésekor megjelenített kategória főcsoportok listázása

Method	Endpoint	Headers
GET	api/public/productCategories	Alapértelmezett

Handled responses:

Status	StatusCode
Success	200
Unauthorized	401
Not found	404
Internal Server Error	500

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
"success": [
    {
        "id": 1,
        "name": "Sample 1",
        "parent_id": null,
        "created_at": "2019-05-09 12:09:10",
        "updated_at": "2019-05-09 12:09:10",
        "deleted_at": null
    },
    {
        "id": 14,
        "name": "Sample 2",
        "parent_id": null,
        "created_at": "2019-05-09 12:09:11",
        "updated_at": "2019-05-09 12:09:11",
        "deleted_at": null
    }
]
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


show():

A termékek rögzítésekor a kiválasztott kategória főcsoporthoz tartozó alkategóriák listázása

Method	Endpoint	Headers
`GET`	`api/public/productCategories/{id}`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
"success": {
    "id": 1,
    "name": "Számítástechnika",
    "parent_id": null,
    "created_at": "2019-05-09 12:09:10",
    "updated_at": "2019-05-09 12:09:10",
    "deleted_at": null
}
}
```


Status: `unauthenticated` StatusCode: `401` Response: `JSON`


```
{
    "message": "Unauthenticated."
}
```


Form requests:

Nincs szükség fromRequets file(ok)-ra.

- **Public/Sellers**

Függvények:

100%

index():

termékek rögzítésekor megjelenített eladók listázása

Method	Endpoint	Headers
`GET`	`api/public/sellers`	Alapértelmezett

Handled responses:

Status	StatusCode
Success	`200`
Unauthorized	`401`
Not found	`404`
Unauthorized	`401`
Internal Server Error	`500`

Response:

Status: `success` StatusCode: `200` Response: `JSON`


```
{
"success": [
    {
        "id": 1,
        "name": "Media Markt Magyarország Kft."
    },
    {
        "id": 2,
        "name": "Euronics Magyarország Zrt."
    },
    {
        "id": 3,
        "name": "Tesco Global Zrt."
    }
]
}
```


FRONTEND

- **API**

Admin lekérdezések

Összes vásárló lekérése

secureFetch bemeneti értékei

Paraméter	Érték
`params`	`aktuális oldal száma`
`URL`	`/admin/customers${params}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getCustomers = (params) => {
  secureFetch(
    `/admin/customers${params}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getCustomers } from './api/queries'

getCustomers(current_page).then(response => response).catch(error => error)
```

Új vásárló létrehozása

secureFetch bemeneti értékei

Paraméter	Érték
`URL`	`/admin/customers`
`method`	`POST`
`headers`	`headersWidthAuth`
`body`	`data`

BODY paraméter értéke

```
const data = {
  "email"                       : "required|email|unique:users",
  "email_confirmation"          : "required|same:email",
  "role_id"                     : "required|integer",
  "password"                    : "required|string|min:8|confirmed|regex",
  "password_confirmation"       : "required|same:password",
  "first_name"                  : "required|string",
  "last_name"                   : "required|string",
  "wired_phone"                 : "sometimes|string",
  "mobile_phone"                : "required|string",
  "home_address_postal_code"    : "required|integer",
  "home_address_city"           : "required|string",
  "home_address_street"         : "required|string",
  "home_address_street_number"  : "required|string",
  "home_address_door_number"    : "sometimes|string",
  "postal_address_postal_code"  : "required|integer",
  "postal_address_city"         : "required|string",
  "postal_address_street"       : "required|string",
  "postal_address_street_number" : "required|string",
  "postal_address_door_number"  : "sometimes|string"
}
```

Függvény deklarációja

```
export const createCustomer = (data) => {
  secureFetch(
    '/admin/customers',
    'POST',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { createCustomer } from './api/queries'

createCustomer(data).then(response => response).catch(error => error)
```

Vásárló adatainak módosítása

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`vásárló id`
`URL`	`/admin/customers/${id}`
`method`	`PUT`
`headers`	`headersWidthAuth`
`body`	`data`

BODY paraméter értéke

```
const data = {
  "id"                          : "required|integer",
  "email"                       : "sometimes|email|confirmed",
  "password"                    : "sometimes|string|min:8|confirmed|regex",
  "first_name"                  : "required|string",
  "last_name"                   : "required|string",
  "wired_phone"                 : "sometimes|string",
  "mobile_phone"                : "required|string",
  "home_address_postal_code"    : "required|integer",
  "home_address_city"           : "required|string",
  "home_address_street"         : "required|string",
  "home_address_street_number"  : "required|string",
  "home_address_door_number"    : "sometimes|string",
  "postal_address_postal_code"  : "required|integer",
  "postal_address_city"         : "required|string",
   "postal_address_street"        : "required|string",
   "postal_address_street_number" : "required|string",
   "postal_address_door_number"   : "sometimes|string"
}
```

Függvény deklarációja

```
export const updateCustomer = (id, data) => {
  secureFetch(
    `/admin/customers/${id}`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
      ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { updateCustomer } from './api/queries'

updateCustomer(id, data).then(response => response).catch(error => error)
```

Vásárló adatainak lekérdezése id alapján

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`vásárló id`
`URL`	`/admin/customer/${id}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getCustomerById = (id) => {
  secureFetch(
    `/admin/customer/${id}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getCustomerById } from './api/queries'

getCustomerById(id).then(response => response).catch(error => error)
```

Vásárló státuszának állítása

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`felhasználó id`
`URL`	`/admin/customers/${id}/status`
`method`	`PUT`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
    "user_id": Number,
    "active": "boolean"
}
```

Függvény deklarációja

```
export const toggleStatusCustomer = (id, data) => {
  secureFetch(
    `/admin/customers/${id}/status`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { toggleStatusCustomer } from './api/queries';

toggleStatusCustomer(id, data).then(response => response);
```

Összes eladó lekérése

secureFetch bemeneti értékei

Paraméter	Érték
`PARAMS`	`aktuális oldal száma`
`URL`	`/admin/sellers${params}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getSellers = (params) => {
  secureFetch(
    `/admin/sellers${params}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getSellers } from './api/queries'

getSellers(current_page).then(response => response)
```

Új eladó létrehozása

secureFetch bemeneti értékei

Paraméter	Érték
URL	/admin/sellers
method	POST
headers	headersWidthAuth
body	data

BODY paraméter értéke

```
const data = {
  "email"                       : "required|email",
  "email_confirmation"          : "required|same:email",
  "role_id"                     : "required|integer",
  "password"                    : "required|string|min:8|confirmed|regex",
  "password_confirmation"       : "required|same:password",
  "name"                        : "required|string",
  "tax_number"                  : "required|string",
  "registration_number"         : "required|string",
  "home_address_postal_code"    : "required|integer",
  "home_address_city"           : "required|string",
  "home_address_street"         : "required|string",
  "home_address_street_number"  : "required|string",
  "home_address_door_number"    : "sometimes|string",
  "site_address_postal_code"    : "required|integer",
  "site_address_city"           : "required|string",
  "site_address_street"         : "required|string",
  "site_address_street_number"  : "required|string",
  "site_address_door_number"    : "sometimes|string",
  "postal_address_postal_code"  : "required|integer",
  "postal_address_city"         : "required|string",
  "postal_address_street"       : "required|string",
  "postal_address_street_number" : "required|string",
  "postal_address_door_number"  : "sometimes|string",
  "contact_person_name"         : "sometimes|string",
  "contact_person_email"        : "sometimes|email",
  "contact_person_mobile_phone" : "sometimes|string",
  "contact_person_wired_phone"  : "sometimes|string",
  "authenticated_seller"        : "required|boolean"
}
```

Függvény deklarációja

```
export const createSeller = (data) => {
  secureFetch(
    '/admin/sellers',
    'POST',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { createSeller } from './api/queries'

createSeller(data).then(response => response)
```

Eladó adatainak módosítása

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`eladó id`
`URL`	`/admin/sellers/${id}`
`method`	`PUT`
`headers`	`headersWidthAuth`
`body`	`data`

BODY paraméter értéke

```
const data = {
  "id"                          : "required|integer",
  "email"                       : "sometimes|email|confirmed",
  "password"                    : "sometimes|string|min:8|confirmed|regex",
  "name"                        : "required|string",
  "tax_number"                  : "required|string",
  "registration_number"         : "required|string",
  "home_address_postal_code"    : "required|integer",
  "home_address_city"           : "required|string",
  "home_address_street"         : "required|string",
  "home_address_street_number"  : "required|string",
  "home_address_door_number"    : "sometimes|string",
  "site_address_postal_code"    : "required|integer",
  "site_address_city"           : "required|string",
  "site_address_street"         : "required|string",
  "site_address_street_number"  : "required|string",
  "site_address_door_number"    : "sometimes|string",
  "postal_address_postal_code"  : "required|integer",
  "postal_address_city"         : "required|string",
  "postal_address_street"       : "required|string",
  "postal_address_street_number": "required|string",
  "postal_address_door_number"  : "sometimes|string",
  "contact_id"                  : "sometimes|integer",
  "contact_person_name"         : "sometimes|string",
  "contact_person_email"        : "sometimes|email",
  "contact_person_mobile_phone" : "sometimes|string",
  "contact_person_wired_phone"  : "sometimes|string",
  "authenticated_seller"        : "required|boolean"
}
```

Függvény deklarációja

```
export const updateSeller = (id, data) => {
  secureFetch(
    `/admin/sellers/${id}`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { updateSeller } from './api/queries'

updateSeller(id, data).then(response => response)
```

Eladó adatainak lekérdezése id alapján

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`eladó id`
`URL`	`/admin/seller/${id}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getSellerById = (id) => {
  secureFetch(
    `/admin/seller/${id}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getSellerById } from './api/queries'

getSellerById(id).then(response => response)
```

Eladó státuszának állítása

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`felhasználó id`
`URL`	`/admin/sellers/${id}/status`
`method`	`PUT`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "user_id": Number,
  "active": "boolean"
}
```

Függvény deklarációja

```
export const getSellerById = (id) => {
  secureFetch(
    `/admin/seller/${id}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { toggleStatusSeller } from './api/queries';

toggleStatusSeller(id, data).then(response => response);
```

Jogosultságok lekérdezése

secureFetch bemeneti értékei

Paraméter	Érték
`PARAMS`	`aktuális oldal száma`
`URL`	`/admin/permissions${params}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getPermissions = (params) => {
  secureFetch(
    `/admin/permissions${params}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getPermissions } from './api/queries';

getPermissions(params).then(response => response);
```

Új jogosultság létrehozása

secureFetch bemeneti értékei

Paraméter	Érték
`URL`	`/admin/permissions`
`method`	`POST`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "name"          : "required|string",
  "guard_name"    : "required|string"
}
```

Függvény deklarációja

```
export const createPermission = (data) => (
  secureFetch(
    '/admin/permissions',
    'POST',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
      ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
)

```

Függvény használata

```
import { createPermission } from './api/queries';

createPermission(data).then(response => response);
```

Jogosultság módosítása

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`jogosultság id`
`URL`	`/admin/permissions`
`method`	`PUT`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "id"            : "required|integer",
  "name"          : "required|string",
  "guard_name"    : "required|string"
}
```

Függvény deklarációja

```
export const updatePermission = (id, data) => {
  secureFetch(
    `/admin/permissions/${id}`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { updatePermission } from './api/queries';

updatePermission(id, data).then(response => response);
```

Szerepkörök lekérdezése

secureFetch bemeneti értékei

Paraméter	Érték
`PARAMS`	`akutális oldal száma`
`URL`	`/admin/roles${params}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getRoles = (params) => {
  secureFetch(
    `/admin/roles${params}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getRoles } from './api/queries';

getRoles(params).then(response => response).catch(error => error)
```

Szerepkör lekérdezése

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`szerepkör id`
`URL`	`/admin/roles/${id}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getRoleById = (id) => {
  secureFetch(
    `/admin/roles/${id}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getRoleById } from './api/queries';

getRoleById(id).then(response => response).catch(error => error)
```

Új szerepkör létrehozása

secureFetch bemeneti értékei

Paraméter	Érték
`URL`	`/admin/roles`
`method`	`POST`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "name"          : "required|string",
  "guard_name"    : "required|string",
  "permissions"   : "required"
}
```

Függvény deklarációja

```
export const createRole = (data) => {
  secureFetch(
    '/admin/roles',
    'POST',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { createRole } from './api/queries';

createRole(data).then(response => response).catch(error => error)
```

Szerepkör módosítása

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`szerepkör id`
`URL`	`/admin/roles/${id}`
`method`	`PUT`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "id"            : "required",
  "name"          : "required|string",
  "guard_name"    : "required|string",
  "permissions"   : "required|array"
}
```

Függvény deklarációja

```
export const updateRole = (id, data) => {
  secureFetch(
    `/admin/roles/${id}`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { updateRole } from './api/queries';

updateRole(id, data).then(response => response).catch(error => error)
```

Szerepkör törlése

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`szerepkör id`
`URL`	`/admin/roles/${id}`
`method`	`DELETE`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "id" : "required|integer"
}
```

Függvény deklarációja

```
export const deleteRole = (id, data) => {
  secureFetch(
    `/admin/roles/${id}`,
    'DELETE',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { deleteRole } from './api/queries';

deleteRole(id, data).then(response => response).catch(error => error)
```

Termékkategóriák lekérdezése

secureFetch bemeneti értékei

Paraméter	Érték
`PARAMS`	`aktuális oldal száma`
`URL`	`/admin/products${params}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
1  export const getProductCategories = (params) => {
2    secureFetch(
3      `/admin/products${params}`,
4      'GET',
5      headersWidthAuth(
6        localStorage.getItem('token_type'),
7        localStorage.getItem('access_token'),
8        localStorage.getItem('default_languages')
9      )
10   ).then(res => res)
11    .catch(error => error)
12 }
```

Függvény használata

```
1  import { getProductCategories } from './api/queries';
2
3  getProductCategories(params).then(response => response).catch(error => error)
```

Termékkategória lekérdezése

secureFetch bemeneti értékei

Paraméter	Érték
`ID`	`termékkategória id`
`URL`	`/admin/products/${id}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getProductCategoryById = (id) => (
  secureFetch(
    `/admin/products/${id}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
)

```

Függvény használata

```
import { getProductCategoryById } from './api/queries';

getProductCategoryById(id).then(response => response).catch(error => error)
```

Új termékkategória létrehozása

secureFetch bemeneti értékei

Paraméter	Érték
URL	/admin/products
method	POST
headers	headersWidthAuth
data	data

BODY paraméter értéke

```
const data = {
  "name"      : "required",
  "parent_id" : "sometimes"
}
```

Függvény deklarációja

```
export const createProductCategories = (data) => {
  secureFetch(
    '/admin/products',
    'POST',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { createProductCategories } from './api/queries';

createProductCategories(data).then(response => response).catch(error => error)
```

Termékkategória módosítása

secureFetch bemeneti értékei

Paraméter	Érték
ID	termékkategóri id
URL	/admin/products/${id}
method	PUT
headers	headersWidthAuth
data	data

BODY paraméter értéke

```
const data = {
  "id"        : "required|integer",
  "name"      : "required",
  "parent_id" : "sometimes"
}
```

Függvény deklarációja

```
export const updateProductCategories = (id, data) => {
  secureFetch(
    `/admin/products/${id}`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { updateProductCategories } from './api/queries';

updateProductCategories(id, data).then(response => response).catch(error => error)
```

Extra form mezők lekérdezése

secureFetch bemeneti értékei

Paraméter	Érték
`PARAMS`	`aktuális oldal száma`
`URL`	`/admin/formItems${params}`
`method`	`GET`
`headers`	`headersWidthAuth`

Függvény deklarációja

```
export const getExtraFormItems = (params) => {
  secureFetch(
    `/admin/formItems${params}`,
    'GET',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    )
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { getExtraFormItems } from './api/queries';

getExtraFormItems(params).then(response => response).catch(error => error)
```

Új extra form mező létrehozása

secureFetch bemeneti értékei

Paraméter	Érték
`URL`	`/admin/formItems`
`method`	`POST`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
    "form_group_product_subcategory_id" : "required",
    "form_group_name"                   : "required",
    "form_group_description"            : "required",
    "form_item_type"                    : "required",
    "form_item_name"                    : "required",
    "form_item_label"                   : "required"
}
```

Függvény deklarációja

```
export const createExtraFormItems = (data) => {
  secureFetch(
    '/admin/formItems',
    'POST',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { createExtraFormItems } from './api/queries';

createExtraFormItems(data).then(response => response).catch(error => error)
```

Extra form mező módosítása

secureFetch bemeneti értékei

Paraméter	Érték
`form_item_id`	`extra form mező id`
`URL`	`/admin/formItems/${form_item_id}`
`method`	`PUT`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "form_group_id"                    : "required",
  "form_group_product_subcategory_id" : "required",
  "form_group_name"                  : "required",
  "form_group_description"           : "required",
  "form_item_type"                   : "required",
  "form_item_name"                   : "required",
  "form_item_label"                  : "required"
}
```

Függvény deklarációja

```
export const updateExtraFormItems = (form_item_id, data) => {
  secureFetch(
    `/admin/formItems/${form_item_id}`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { updateExtraFormItems } from './api/queries';

updateExtraFormItems(form_item_id, data).then(response => response).catch(error => error)
```

Extra form mező státuszának állítása

secureFetch bemeneti értékei

Paraméter	Érték
`form_item_id`	`extra form mező id`
`URL`	`/admin/formItems/${form_item_id}/status`
`method`	`PUT`
`headers`	`headersWidthAuth`
`data`	`data`

BODY paraméter értéke

```
const data = {
  "form_group_id" : "required",
  "active"        : "required"
}
```

Függvény deklarációja

```
export const toggleStatusFormItems = (form_item_id, data) => {
  secureFetch(
    `/admin/formItems/${form_item_id}/status`,
    'PUT',
    headersWidthAuth(
      localStorage.getItem('token_type'),
      localStorage.getItem('access_token'),
      localStorage.getItem('default_languages')
    ),
    JSON.stringify(data)
  ).then(res => res)
   .catch(error => error)
}
```

Függvény használata

```
import { toggleStatusFormItems } from './api/queries';

toggleStatusFormItems(form_item_id, data).then(response => response).catch(error => error)
```