

# D O K U M E N T Á C I Ó

## Vizsgaremek

## SZOFTVERFEJLESZTŐ ÉS -TESZTELŐ TECHNIKUS SZAKMA

### Tartalomjegyzék

1. Bevezetés, a téma ismertetése, témaválasztás indoklása .....	1
2. Felhasználói dokumentáció .....	4
2.1 Telepítési útmutatás Windows eszközre .....	4
2.2 A Web App telepítése és konfigurálása .....	6
3. Fejlesztői dokumentáció .....	28
3.1 A fejlesztési környezet, használt technológiák, keretrendszerek bemutatása .....	28
3.2 A webes projektben használt technológiák és verziók .....	32
3.3 Az asztali alkalmazásban használt technológiák és funkciók .....	34
3.4 Adatbázis, adattáblák felépítése és diagram .....	36
3.5 Útvonalak listája .....	37
4. Tesztelés .....	43

### 1. Bevezetés, a téma ismertetése, témaválasztás indoklása

A megvalósított projekt egy hotel honlapot és annak foglalási rendszerét, valamint termékeinek kezelését valósítja meg. A téma választás oka egy barátunk kérése, mivel jelen pillanatban épít egy hotelt-panziót a Balaton-felvidéken. Kérésének megfelelően a prioritásunk egy full-stack web létrehozása volt.

Kérésére láttunk neki a projektnek, mint referencia értékű feladatnak, mivel mindannyian tudjuk, hogy a mai világban rendkívül könnyű egy WordPress sablonnal egy SPA-t vagy bármilyen honlap sablont létre hozni és publikálni, viszont valós idejű foglalásokat az airbnb.com, a szallas.hu és a Booking.com legalább 10%-os költséggel kezel a teljes foglalás árából, még csak nem is a szállásadó profitjából.

Ezért manapság már a legtöbb önmagára adó hotel tulajdonos a saját honlapján is szeretné, ha a vendégek, felhasználók tudnák a foglalásaikat kezelni.

Ez nem azt jelent, hogy nem hirdetnek a legfőbb platformokon és nem érkezik ott foglalásaik nagy része, hanem azt, hogy saját platformokon is szeretnék kiadni szobáikat a közvetítő cégek költségei nélkül. Ezért kulcsfontosságú volt számunkra a foglalások kezelése adminisztrátor felhasználók számára, ezután a 'User' -azaz vendég- felhasználók saját foglalásaik kezelése, majd mindezek mellett szeretnük volna, ha a szálloda termékeinek kezelés is megvalósul a web applikáción.

A termék kezelés is valóban live adatokkal történik a honlapon, valamint a termékek adatai elérhetőek 3rd party applikációk és API-k számára is.

Az adminisztrátor tud termékeket felvinni, módosítani, törölni és ahogy végre hatja bármelyik műveletet azt a bejelentkezett felhasználók is látják a honlapon valós időben.

Ez azért is font mert az összes modern üzlet árai gyorsan változnak a mai világban köszönhetően a hirtelen és magas inflációs nyomásnak, valamint a magyar Forint deviza elértéktelenedésének köszönhetően, hiszen napjainkban a magyar vállalkozások termékeik több mint 60-70%-át importálják, azaz euróban vagy dollárban fizetnek a beszerzés közben az áruért. Ha pedig Forint gyengül akkor a magyar cégeknek többet kell fizetni az árért dollárban ezért kénytelenek növelni az itthon eladott termékek árait.

Mindezek fényében fontos volt számunkra, hogy a hotel tulajdonos/ vezetőség / adminisztrátornak egy egyszerű mégis dinamikus és valós idejű lehetősége legyen a szálloda termékeinek kezelésére, főleg úgy, hogy későbbi terveink között szerepel a hotel foglalást is kibővíteni további funkciókkal, valamint a termék kezelést egy kisebb ERP (Enterprise Resource Planning - Vállalati Erőforrás Tervezés) rendszerré változtatni.

Fontos megemlítenem, hogy Vállalati Erőforrás Tervezés azaz ERP világában az SAP SE német cég piacvezető több évtizedes tapasztalattal, így nem kívánnánk ekkor konglomerátumokkal versenyezni, hanem a kisebb hoteleket, panziókat céloznánk meg, mint lehetséges jövőbeli megrendelők.

Fontos megemlítenünk, hogy az elkészült hotel projekt egyfajta sablonként is szolgál majd számunkra a jövőben, mivel terveink között szerepel fodrászatok, műkörmösök, autószerelők, orvosi rendelők foglалási rendszerinek is a kiszolgálása és az, hogy a hotelben napokat kezelünk szobánként, míg a többi iparban napok és órákat kezelünk per kiszolgáló személyzet nem igazán nagy különbség egy fejlesztő számára.

Mindezeknek a fényében logikusan esett választásunk a Laravel-re, mivel az iskolai oktatás Back end fejlesztés tananyagában is hosszasan tanulmányoztuk a framework-ot.

A Laravel egy PHP-alapú és nyílt forráskódú háttér-keretrendszer, amelyet egyéni webalkalmazások széles skálájának létrehozására használnak. Ez egy teljes mértékben szerveroldali keretrendszer, amely a Model-View-Controller (MVC) tervezés segítségével kezeli az adatokat, amely logikai részekre bontja az alkalmazás-háttér-architektúrát.

Asztali alkalmazásunkban- mivel az üzlet felhasználók több mint 70%-a Windows operációs rendszert használ, a magán személyek pedig még magasabb százalékban- a .NET framework-re esett választásunk, a Visual Studio IDE által nyújtott Windows-ra célzott Microsoft termékre.

Az asztali alkalmazás jelenlegi funkciója, hogy begyűjti a termék adatokat Http kliens-el, a Laravel REST API által adott json-t feldolgozza és az API által adott adatokat megjeleníti egy táblában, majd egy gomb megnyomásával az adatokat .csv formátumban el is lehet menteni.

## 2. Felhasználói dokumentáció

### 2.1 Telepítési útmutatás Windows eszközre

#### **Xampp telepítése:**

1. Nyissa meg az Apache Friends webhelyet:  
<https://www.apachefriends.org/download.html>
2. Kattintson a Letöltés gombra az XAMPP Windows-verziójához, és mentse a fájlt a számítógépére.
3. Gyors megjegyzés: A projekt a PHP 8.0.12 verzióval lett készítve és tesztelve. Amennyiben másik verzió van telepítve a gépen, telepítse a PHP 8.0.12 vagy 8.0.2 verziót, de mindenképpen a PHP 8.0.12-es verzió az ajánlott.
4. A telepítő elindításához kattintson duplán a letöltött fájlra.
5. Kattintson az OK gombra.
6. Kattintson a Tovább gombra.
7. Az XAMPP különféle telepíthető összetevőket kínál, mint például a MySQL, phpMyAdmin, PHP, Apache stb. A legtöbb esetben ezeket az összetevőket fogja használni, ami azt jelenti, hogy ajánlott hagyni az alapértelmezett beállításokat.
8. Kattintson a Tovább gombra.
9. Használja az alapértelmezett telepítési helyet. (Vagy válasszon másik mappát a szoftver telepítéséhez a „Mappa kiválasztása” mezőben.)
10. Kattintson a Tovább gombra.
11. Válassza ki az XAMPP vezérlőpult nyelvét.
12. Kattintson a Tovább gombra.

13. Törölje a További információ a Bitnami for XAMPP-ról lehetőséget.
14. Kattintson a Tovább gombra.
15. Kattintson ismét a Tovább gombra.
16. Kattintson a Hozzáférés engedélyezése gombra, hogy engedélyezze az alkalmazást a Windows tűzfalon (ha van).
17. Kattintson a Befejezés gombra.
18. A lépések végrehajtása után elindul az XAMPP Vezérlőpult, és megkezdheti a webservert környezet konfigurálását.

### **Az XAMPP konfigurálása Windows 10 rendszeren**

1. Az XAMPP vezérlőpult három fő részből áll. A Modulok részben megtalálja az összes elérhető webszolgáltatást. Az egyes szolgáltatásokat a Start gombra kattintva indíthatja el.
2. Amikor elindít néhány szolgáltatást, beleértve az Apache-t és a MySQL-t, a jobb oldalon, az egyes szolgáltatások által használt folyamatazonosító (PID) és TCP/IP-port (Port) száma is megjelenik. Például alapértelmezés szerint az Apache a 80-as és a 443-as TCP/IP portot, míg a MySQL a 3306-os TCP/IP portot használja.
3. Az Adminisztrálás gombra kattintva elérheti az egyes szolgáltatások adminisztrációs irányítópultját, és ellenőrizheti, hogy minden megfelelően működik-e.
4. Az alapértelmezett beállításoknak működniük kell a legtöbb felhasználó számára, aki XAMPP-t használ a webhely futtatásához szükséges tesztelési környezet létrehozásához. A beállítási konfigurációtól függően azonban előfordulhat, hogy módosítania kell az Apache szerver TCP/IP portszámát, az adatbázis feltöltési méretét, vagy be kell állítania a phpMyAdmin jelszavát.
5. A beállítások módosításához a megfelelő szolgáltatás Config gombját kell használnia. Például meg kell nyitnia a httpd.conf fájlt az Apache-kiszolgáló beállításainak módosításához, a my.ini fájlt pedig a MySQL-beállítások módosításához.

6. Xampp Control panel indítása
7. Apache és MySQL rákattintani a Start gombra
8. MYSQL gombon az admin gombra kattintva eljutunk a <http://localhost/phpmyadmin/> oldalra

## 2.2 A Web App telepítése és konfigurálása

### phpMyAdmin konfigurálás:

1. Új adatbázis létrehozása: "laravel" címen
2. Ez fontos, mivel az env állományban van meghatározva az adatbázis neve és a migrációs séma a projektben előre meg van határozva, azaz ebbe az adatbázisba fogjuk migrálni az adatbázis táblákat és azoknak a Laravel php-ban megadott relációs sémáját is.

### Composer telepítése:

1. Composer telepítése exe fájl formátumban az alábbi webhelyről:  
<https://getcomposer.org/download/>
2. Letöltött exe fájl futtatása és telepítése a legújabb php verzóval
3. Parancssor megnyitása és a "composer" szó beírásával majd az "Enter" szó leütésével ellenőrizzük, hogy a composer sikeresen települt a Windows 10 környezetben

### Laravel telepítése:

1. composer global require laravel/installer

### Projekt mappa készítése és Laravel telepítése:

1. composer create-project --prefer-dist laravel/laravel **ProjektNév**
2. Ezután cd az az be kell lépni a laravel **ProjektNév** projekt mappába és ott Git bash vagy Új terminál VS Code programban

### 3. php artisan migrate

#### **Jetstream telepítése:**

1. `composer require laravel/Jetstream`
2. `php artisan jetstream:install livewire`
3. `php artisan jetstream:install inertia`
4. `npm install`
5. `npm run dev`
6. `php artisan migrate`
7. `php artisan vendor:publish --tag=jetstream-views`
8. `npm run dev`

#### **Laravel Sanctum telepítése:**

1. `composer require laravel/sanctum`
2. `php artisan vendor:publish --  
provider="Laravel\Sanctum\SanctumServiceProvider"`
3. `php artisan migrate`

#### **Laravel Fortify telepítése**

1. `composer require laravel/fortify`
2. `php artisan vendor:publish --  
provider="Laravel\Fortify\FortifyServiceProvider"`
3. `php artisan migrate`

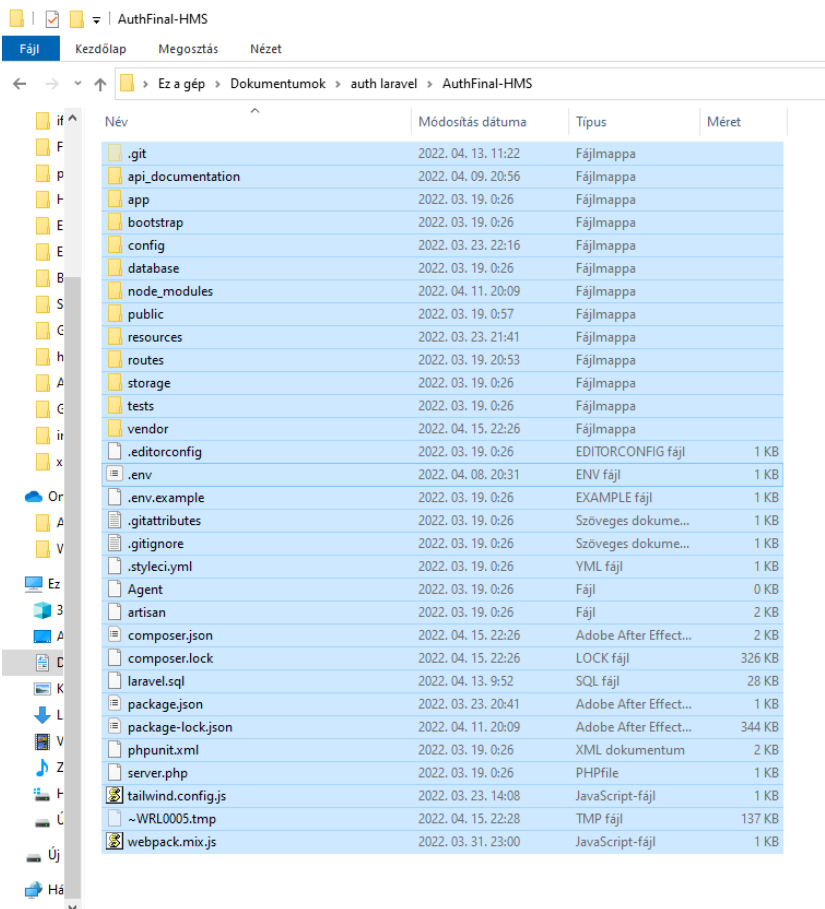
#### **Doctrine Dbal telepítése**

1. `composer require doctrine/dbal`

#### **Clone Laravel projekt:**

1. A file explorerben job egér gomb klikkel vagy Git bash here vagy open with VS code-al meg kell nyitni egy mappát és oda klónozni a `git clone` paranccsal a project mappát, avagy azt letölteni a Github URL-ről zip formátumban. A klónozáshoz új Terminál ablakot kell létesíteni a VS Code-ban vagy a projekt mappában job klikk és a `GIT bash here` paranccsal lehet Terminált létesíteni.
2. **Egy Terminálban beírni a következő parancsokat:**

3. git clone 'url'
4. ha megvan a projekt a lokális mappában akkor be kell lépni az AuthFinal-HMS klónozott mappába és annak az egész tartalmát Ctrl+ A paranccsal kijelölni és Ctrl-C / Ctrl +V billentyűk segítségével át kell másolni a korábban composer-el létrehozott **ProjektNév** mappába, ahol már a teljes környezet kiépítésre került (composer create-project --prefer-dist laravel/laravel **ProjektNév**)



5. Be kell lépni a laravel **ProjektNév** mappába, amit korábban megadott a Laravel project készítésénél
6. A **ProjektNév** mappát megnyitni VS Code alkalmazással
7. Itt **New Terminal** paranccsal új terminált létesítünk
8. **composer install**



## 9. npm install

## 10. npm run dev

## 11. Env file konfigurálása a lejjeb található env paraméterek bemásolása

### Env file konfigurálása:

Fontos az alábbi változók bemásolása az env fájlba (kivéve az APP\_KEY változót, mivel azt a "php artisan key:generate" parancs generálja majd) mivel ezek a paraméterek adják a környezet konfigurációját.

```
APP_NAME="HMS System"
APP_ENV=local
APP_KEY=base64:ET5PUEUNqm6u1V7I1eqgIsqTdkinieqkAhwTm6wLdKM=
APP_DEBUG=true
APP_URL=http://127.0.0.1:8000
ASSET_URL=/public/assets

LOG_CHANNEL=stack
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=

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

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
```

```
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=474d8b7be5ef44
MAIL_PASSWORD=6b25bed91ad7a9
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS= register@budapesthotels.com
MAIL_FROM_NAME="${APP_NAME}"

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

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

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

12.php artisan key:generate

13.php artisan migrate (migrációs hiba esetén törölni kell a duplikált migrációs táblát (database/migrations útvonalon a VS doce-banl) és a

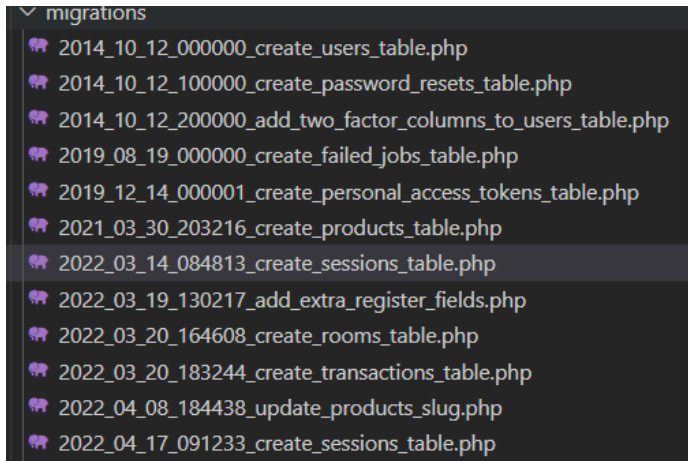
<http://localhost/phpmyadmin> oldalon, például ebben az esetben

```
Elhasználok@DESKTOP-KRFPV10 MINGW64 ~/Documents/auth_laravel TEST/TestHotel
php artisan migrate
igrating: 2021_03_30_203216_create_products_table
igrated: 2021_03_30_203216_create_products_table (31.96ms)
igrating: 2022_03_14_084813_create_sessions_table

Illuminate\Database\QueryException

SQLSTATE[42S01]: Base table or view already exists: 1050 Table 'sessions' already exists (SQL: create table `sessions` (`id` varchar(255) not null, `user_id` bigint unsigned null, `ip_address` varchar(45) null, `user_agent` text null, `payload` text not null, `last_activity` int not null) default character set utf8mb4 collate 'utf8mb4_unicode_ci')
```

a 'sessions' táblát, majd újra kell migrálni a 'php artisan migrate' parancsal



Duplikált session table migration esetén akár törölhetjük a korábbi verziót, persze előtte érdemes ellenőrizni, hogy tartalmilag megegyezik a két fájl.

14.php artisan migrate:fresh

15.npm run watch

16. Adatbázis feltöltése:

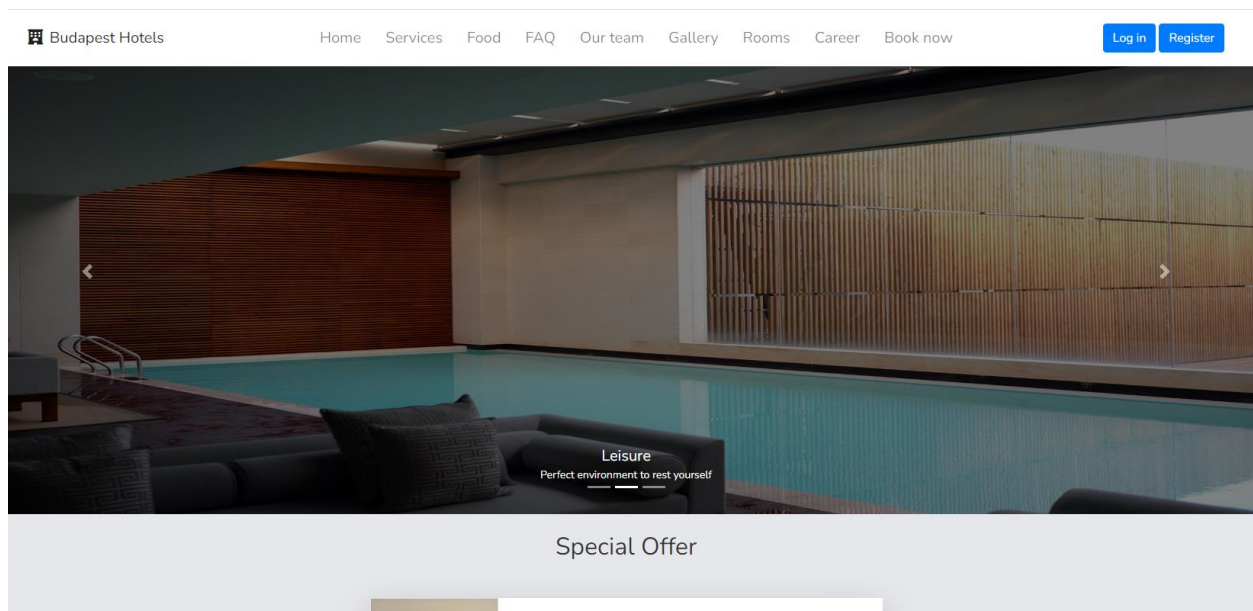
- a. <http://localhost/phpmyadmin/> oldalra látogatni:
- b. Drop database laravel az adatbázis eldobása
- c. Újra csinálni egy laravel nevű adatbázist
- d. A Laravel.sql fájlt a <http://localhost/phpmyadmin/> laravel adatbázisban az Import gombra kattintva, majd a 'Choose file' gombra kattintva be tudjuk illeszteni, majd a 'Go' gomb megnyomásával be is importáljuk az összes korábban bevitt, megadott és exportált adatot.
  - i. Opcionálisan: A user táblát fel tudjuk tölteni random user adatokkal a AuthFinal-HMS\database\seeders\DatabaseSeeder.php elérési útvonalon található fájl alapján a faker segítségével, a "php artisan db:seed" parancsal

17.A Laravel fejlesztési szerver indítása: php artisan serve

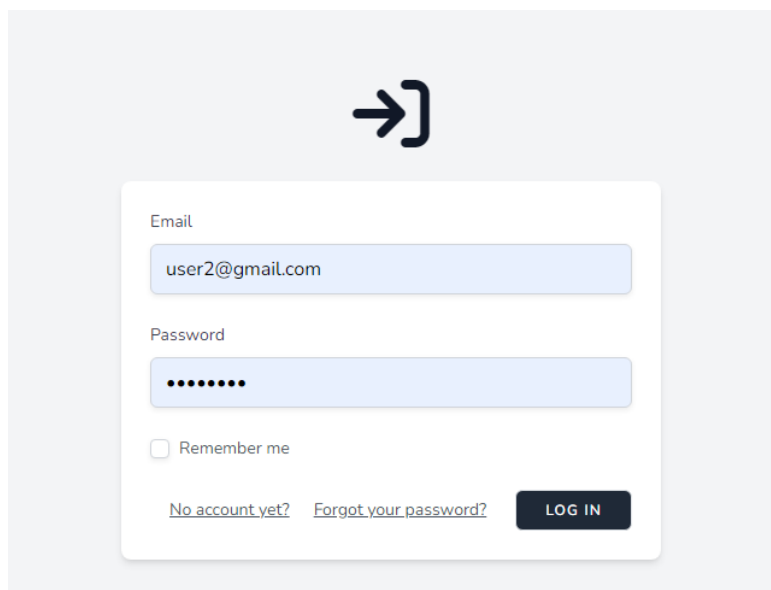
18.Xampp Control Panel start Apach és MySQL ellenőrzése

19. <http://127.0.0.1:8000/> URL-t beírni a böngészőbe

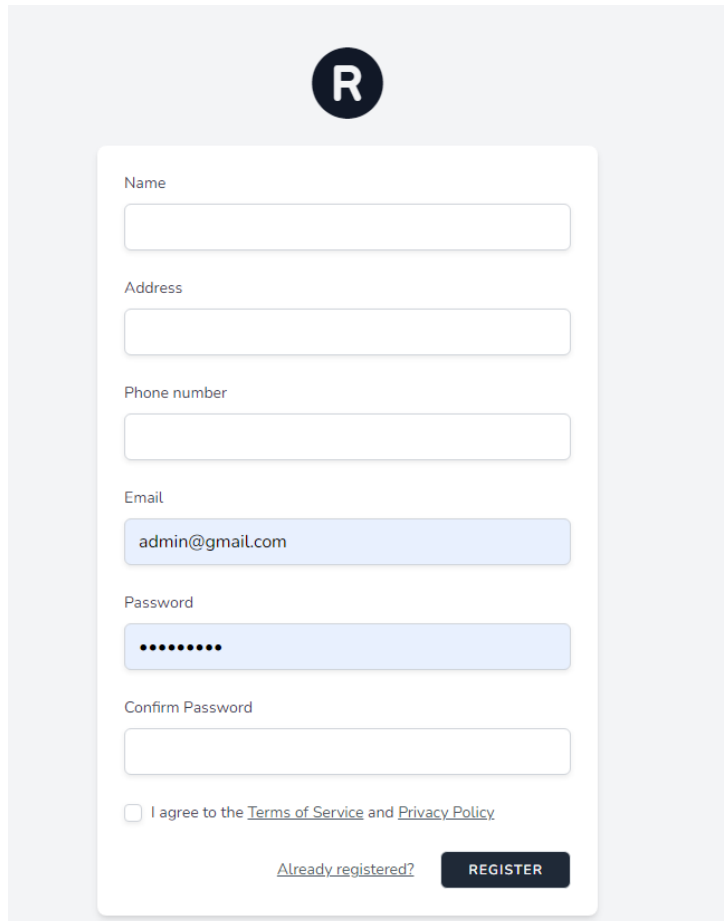
**Kezdőlap:**



A jobb felső sarokban található Login illetve Register gombokkal a /login:



Illetve a /register oldalra érkezik:



**R**

Name

Address

Phone number

Email

admin@gmail.com

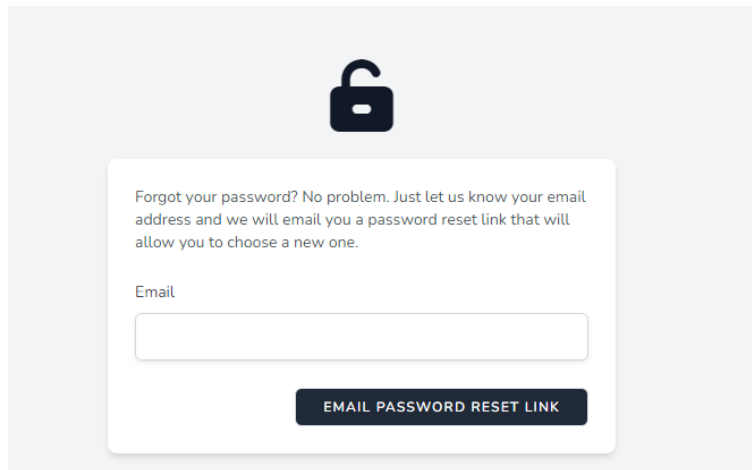
Password

Confirm Password

☐ I agree to the [Terms of Service](#) and [Privacy Policy](#)

[Already registered?](#) **REGISTER**

A /login oldalról a No account Yet? -re kattintva a /register oldalra tud eljutni a felhasználó, míg a Forgot your password? -re kattintás esetén a /forgot-password nézet jelenik meg:



Mind regisztráció mind jelszó elfelejtés esetén email-es jóváhagyásra van szükség.

### **Email küldés:**

A web appon történt regisztráció után a regisztrációt jóvá kell hagyni email-en keresztül, egy felhasználó csak azután tudja használni a programot, hogyha a regisztrációt követően igazolta email címét, valamint jelszó visszaállítás, helyreállítás is emailben küldött linken keresztül működik.

Az email funkciók teszteléséhez és használatához a **MailTrap** szolgáltatását vettük igénybe.

### **Mailtrap bemutatása:**

A Mailtrap hamis SMTP-kiszolgálót biztosít a fejlesztőcsapat számára, amellyel tesztelheti, megtekintheti és megoszthatja a gyártás előtti környezetekből küldött e-maileket, és valós adatokkal tesztelheti anélkül, hogy fennállna a valódi ügyfelek spamküldésének veszélye. A Railsware készítette, és számos fejlesztési feladathoz a Mailtrap használata ingyenes.

Lényegében regisztrál a Mailtrap szolgáltatásra, és minden e-mailt elküld a gyártás előtti környezetből a hamis Mailtrap SMTP-kiszolgálón keresztül.

A MailTrap segítségével e-maileket rögzíthet a tesztelési fejlesztői és állomásozó környezetekből

Ezután minden levele a Mailtraphez tartozik. Megtekintheti és hibakeresheti e-mailjeit a Mailtrap barátságos grafikus felületén.

A Mailtrap segítségével akár valódi felhasználói e-maileket tartalmazó kiíratásokat is elhelyezhet a termelési adatbázisában, tesztek segítségével az átmeneti kiszolgálón. Automatizált tesztjei ellentmondhatnak a valós adatoknak – ha e-mailt küldenek a Mailtrap-on keresztül, így kiküszöbölhető annak a kockázata, hogy a teszte-mailek valódi ügyfelek e-mail címére menjenek.

**URL:** <https://mailtrap.io/>

**Ahhoz, hogy regisztrálni tudjunk a honlapon, a fenti URL-en be kell jelentkezni az alábbi paraméterekkel.**

**A projekthez tartozó felhasználónév és jelszó, ezekkel kell belépni a honlapon.**

Felhasználónév: **emailsenderz53@gmail.com**

Jelszó: **Email23\$**

**Amennyiben Xampp Apache és MySQL még mindig aktív, valamint a Laravel development server is az akkor: <http://127.0.0.1:8000/> vagy a <http://127.0.0.1:8000/index> oldalakon be tud jelentkezni, mint felhasználó a honlapon vagy regisztrálni is tud. Az alábbi fiókokat is lehet használni tesztelésre:**

**Adminisztrátor fiók:**

felhasználónév/ email: [admin@gmail.com](mailto:admin@gmail.com)

jelszó: **Admin111\$**

**Vendég fiók:**

felhasználónév/ email: **user2@gmail.com**

jelszó: **User222\$**

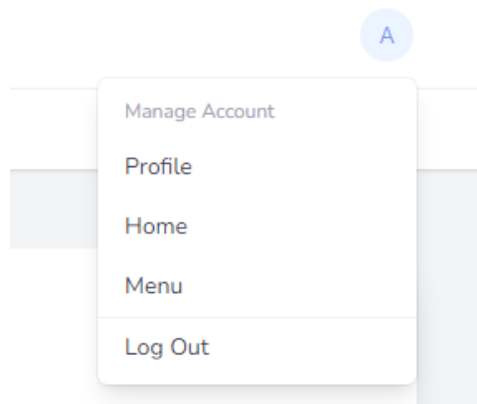
**Minden esetben új regisztráció esetén figyelni kell, hogy a jelszó megadásánál a jelszónak minimum 8 karakterből kell állnia, legyen benne betű, szám és speciális karakter és egy nagy betű is.**

Mindkét URL igazából egy welcome page, azaz ahol a felhasználók meg tudják ismerni a hotelt, tudnak képeket nézni és információt gyűjteni a hotelről ( a navigációs menü 'Gallery' szövegre kattintva) , szobákról ( a navigációs menü 'Room' szövegre kattintva), a szolgáltatásokról( a navigációs menü 'Services' szövegre kattintva) , az ételről ( a navigációs menü 'Food' szövegre kattintva), gyakran ismételt kérdésekről(GYIK) ( a navigációs menü 'FAQ' szövegre kattintva), a felső vezetésről ( a navigációs menü 'Our Team' szövegre kattintva), aktuális álláslehetőségről( a navigációs menü 'Career' szövegre kattintva),szobát foglalni ( a navigációs menü 'Book Now' szövegre kattintva).

A 'Book now' szövegre kattintva vagy be kell jelentkeznie, vagy regisztrálnia kell emailés validálással a felhasználónak.

Admin felhasználó esetén a következő lehetőségek állnak rendelkezésre:

A navigációs sávban a Profile-ra kattintva a profil beállítások válnak elérhetővé:



/user/profile a URL cím:

Itt akár Adminnak akár vendég, azaz sima User felhasználónak is lehetősége a profileban a nevét, email címét, lakcímét és telefonszámát változtatni.

Ezen az oldalon tud az összes felhasználó jelszavát változtatni.



Lejjebb Két kulcsos azonosítást tudnak a felhasználók bekapcsolni Google Authenticator Applikáció segítségével egy egyedi QR kódot tudnak beolvasni, valamint recovery kódokat elmenteni.

Az ezalatti konténerben pedig, ha esetleg több böngésző session lenne nyitva akkor a többi böngészőből a user ki tudja léptetni magát.

És végül, de nem utolsó sorban a felhasználók GDPR-nak megfelelően tudják a saját fiókjukat törölni is.

The screenshot displays the Google Account 'Profile' settings page. It is organized into several sections, each with a title, a brief description, and a set of controls.

- Profile Information:** Includes a photo placeholder with a blue 'A', a 'SELECT A NEW PHOTO' button, and input fields for Name (admin), Email (admin@gmail.com), Role (Admin), Address (Sunset street 123, Miami, Florida), and Phone number (+36201111111). A 'SAVE' button is at the bottom right.
- Update Password:** Features input fields for Current Password, New Password, and Confirm Password, with a 'SAVE' button at the bottom right.
- Two Factor Authentication:** Shows a message: 'You have not enabled two factor authentication. When two factor authentication is enabled, you will be prompted for a second, random token during authentication. You may retrieve this token from your phone's Google Authenticator application.' A 'ENABLE' button is present.
- Browser Sessions:** Includes a warning: 'If necessary, you may log out of all of your other browser sessions across all of your devices. Some of your recent sessions are listed below. However, this list may not be exhaustive. If you feel your account has been compromised, you should also update your password.' It lists one session: 'Windows - Chrome 127.0.0.1, This device'. A 'LOG OUT OTHER BROWSER SESSIONS' button is at the bottom.
- Delete Account:** Contains a warning: 'Once your account is deleted, all of its resources and data will be permanently deleted. Before deleting your account, please download any data or information that you wish to retain.' A red 'DELETE ACCOUNT' button is at the bottom.

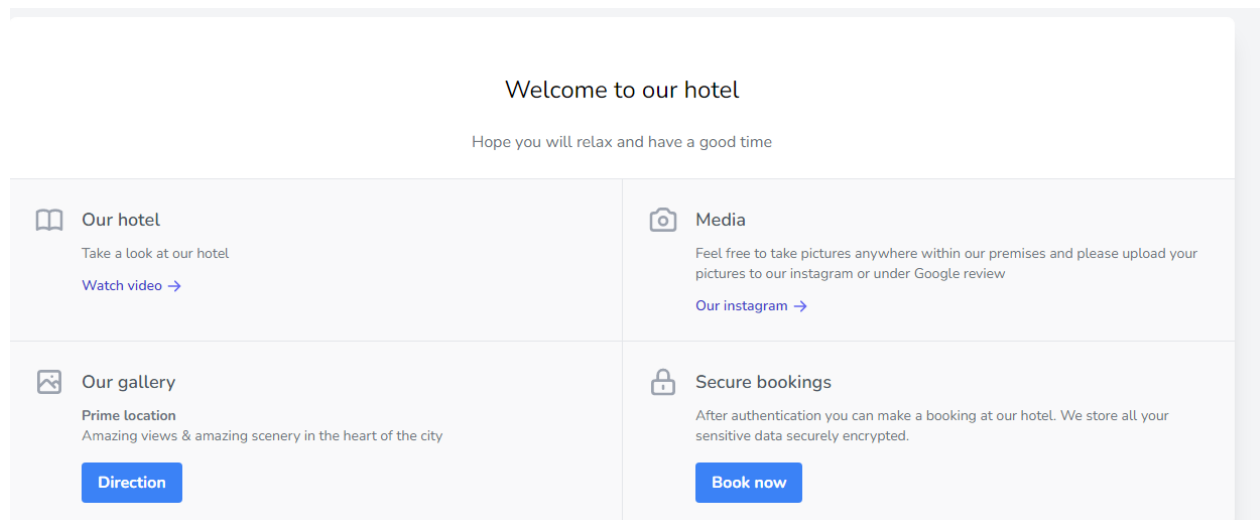
A home page a bejelentkezett felhasználóknak mindig /user/home URI a kezdőlap, ám ez másik nézetet ad vissza annak alapján, hogy a felhasználó jogosultsága admin vagy user.

Admin felhasználó esetén itt található három konténer.

Egy a foglalások kezelésére való átvitelért felelős (Bookings Admin Center konténer, azaz: /transactions útvonal), a második a termékek kezelésére való átvitelért felelős (Products Admin Center, azaz: /products útvonal), a harmadik pedig a felhasználók kezelésére való átvitelért felelős (Users Admin Center, azaz: /users útvonal).

Legfelül van egy admint üdvözlő üzenet, valamint két kék színű link, ahol az egyiken a program összes útvonala jelenik meg (Routes list), a másikra (Download Routes list in CSV) kattintva pedig egy .csv-ben letöltődik a webről a program által használt összes útvonal.

A navigációs menüben a Home page vissza visszavisz a logged-out home page-re, ebben az esetben a Dashboard ikon jelenik meg a logged out navigációs menü jobb felső sarkában a Login és Register helyett hiszen a felhasználó be van jelentkezve és ha a felhasználó a Dashboard-ra kattint akkor újra a logged in view-ban és navbaron lesz a felhasználó a /dashboard nézetben.



A dashboard nézetben van egy útvonal (Google Maps redirect link) választó gomb (Direction) és egy Book Now gomb, ahol a felhasználó a /my-bookings útvonalra jut el ahol csak magának tud foglalni szobát a belépett felhasználó a New gombra kattintva. A media konténer Our instagram az Instagramra visz át, míg a watch a video a YouTube-ra, mindkettő új oldalon jelenik meg.

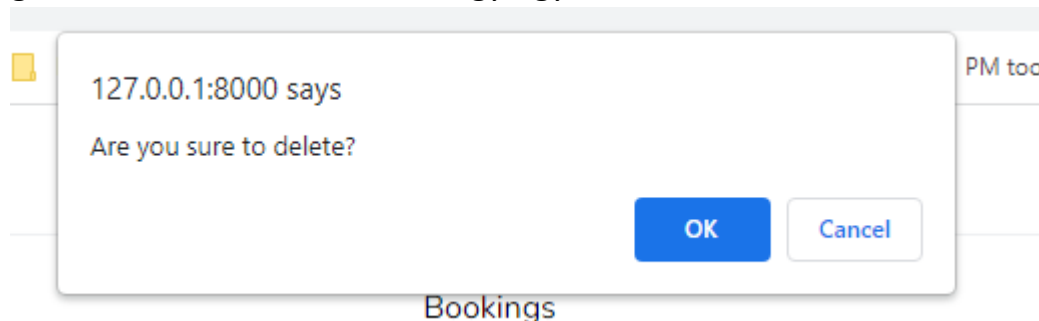
Amennyiben a bejelentkezett felhasználónak van foglalása akkor a táblában ez meg is jelenik:

New

Booking ID	User Name	Email	Phone	Room Name	Checkin	Checkout	Days	Bill	Halfboard	No of guests	Actions
27	admin	admin@gmail.com	+36201111111	Executive	2022-06-30	2022-07-07	7	490000	Yes	2	<div>EditDelete</div>

A user tudja módosítani, illetve törölni is saját foglalását az Edit illetve a Delete gombra kattintva.

Admin felhasználó a /transactions útvonalon tud foglalásokat módosítani az Edit gombbal, törölni a Delete gombbal és a New gombbal új foglalást rögzíteni. Delet gombra kattintás esetén van egy figyelmeztetés



Amíg az admin itt nem kattint az OK gombra addig nem törlődik a foglalás.

Edit gombra kattintva az összes választható paraméter (Guest Name, Guest email, Guest phone, Select Room, Checkin, Checkout, Halfboard, Number of guests) állíthatóvá válik az admin számára. kivéve a Days és a Bill field, mivel ezek auto generáltak a day a Check-in és Check-out napok különbége, míg a Bill ez a különbség megszorozva a napok számával:

Guest name

Toni Maki

Guest email:

joseph@gmail.com

Guest phone:

06206652342

Select Room

Luxury

Checkin:

03/25/2022

Checkout:

03/31/2022

Days:

5

Bill:

123000

Halfboard:

Yes

Number of guests:

4

Cancel

Save

Természetesen mind a `/transactions`, `/products`, `/users` middleware által vannak biztosítva és letiltva a vendégek számára, tehát csak jóváhagyott admin felhasználók tudják ezeket a szenzitív adatokat látni és állítani.

Az admin a /transactions oldalon a New gombra kattintva tud új foglalást rögzíteni.

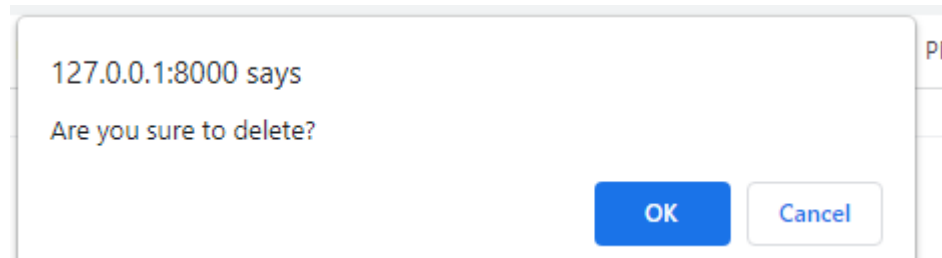
Itt amennyiben még nincs ilyen nevű felhasználó, mint akit az admin felvisz akkor a foglalás rögzítésével együtt a user (azaz vendég) neve, email címe és egy random generált bcrypt titkosítással ellátott jelszó is rögzítésre kerül a user táblában a transaction táblában pedig tároljuk az admin által megadott user telefonszámot.

A Search field-ben, azaz kereső mezőben lehetséges keresni avagy szűrni a foglalások között, szoba név(Room name) vagy email vagy telefonszám alapján.

Az összes foglalást foglalási idő alapján dátum szerint növekvő sorrendben mutatjuk alapértelmezve.

Az oldalon pagination van érvényben, jelenleg 10 találatot mutat egy oldal.

Az admin a /products oldalon tudja a termékeket kezelni. Fel tud vinni új terméket a New gombbal, a terméket tudja kategorizálni, tud leírást adni és árat is, a Save gombbal elmenteni ezt. A Delete gombbal tud törölni terméket az admin, itt is megjelenik egy pop-up warning



Amíg ezt le nem OK-za az admin addig nem törlődik a termék, Cancel gombra kattintva lehet visszalépni és nem törölni a terméket.

Az Edit gombra kattintva lehet a termék nevét, árát, leírását és kategóriáját változtatni (Name, Category, Description, Price).

Minden változtatás azonnal frissül az adatbázisban itt a web útvonalon megadva is, ahonnan a REST API is nyeri a termék adatokat.

Minden termék itt is pagination által jelenik meg 10 találatot mutatva egy oldalon, kategória név alapján növekvő sorrendben.

A Search field-ben, azaz kereső mezőben lehetséges keresni avagy szűrni a termékek között, név(name) vagy category(Kategória) vagy description(leírás) alapján.

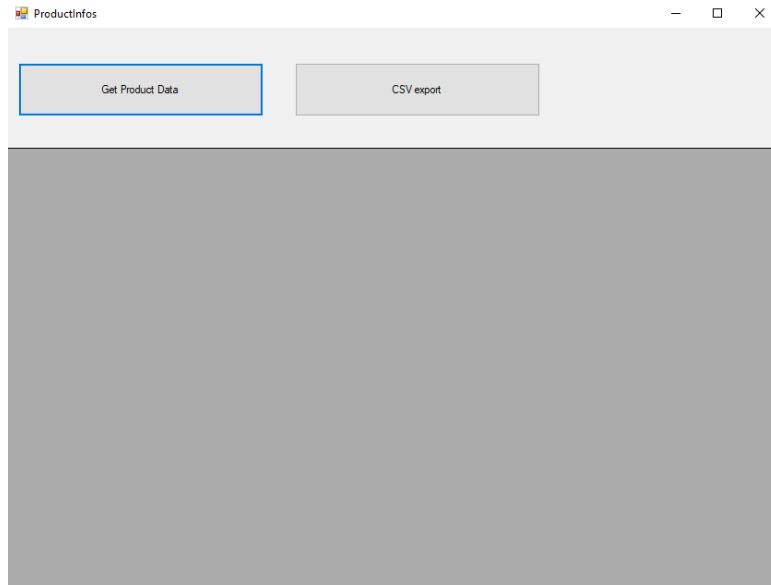
Minden terméket, ami az admin oldalon megjelenik az elérhető a vendégek számára is egy hasonló tábla nézetben csak teljes CRUD operátorok nélkül, azaz szerkeszteni, törölni, avagy új terméket itt felvinni nem lehet, csak Read opció adott, azaz a GET metódus, hogy tudják a vendégek élőben látni a termékeket és keresni közöttük, mindezt a /restaurantandbar útvonalon a Menu & bar navigációs gombra kattintva.

Minden termék itt is pagination által jelenik meg, itt 15 találatot mutatva egy oldalon, kategória név alapján növekvő sorrendben.

A Search field-ben, azaz kereső mezőben lehetséges keresni avagy szűrni a termékek között, név(name) vagy category(Kategória) vagy description(leírás) alapján.

Minden, ami látható egy szépen rendezett táblában a /restaurantandbar útvonalon azt egy nyers json formátumban is meg tudjuk kapni a /jsonproducts útvonalon a weben.

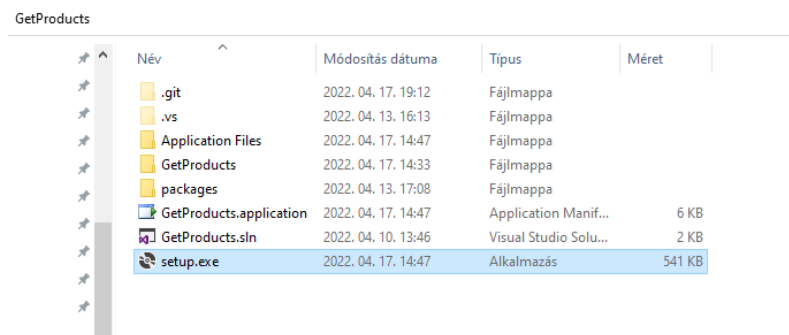
Az asztali alkalmazás az úgynevezett GetProducts is ezeket a adatok szedi le a webről egy Http kliens segítségével majd az asztali alkalmazás is táblába rendezi ezeket a Get Product data-ra kattintva:



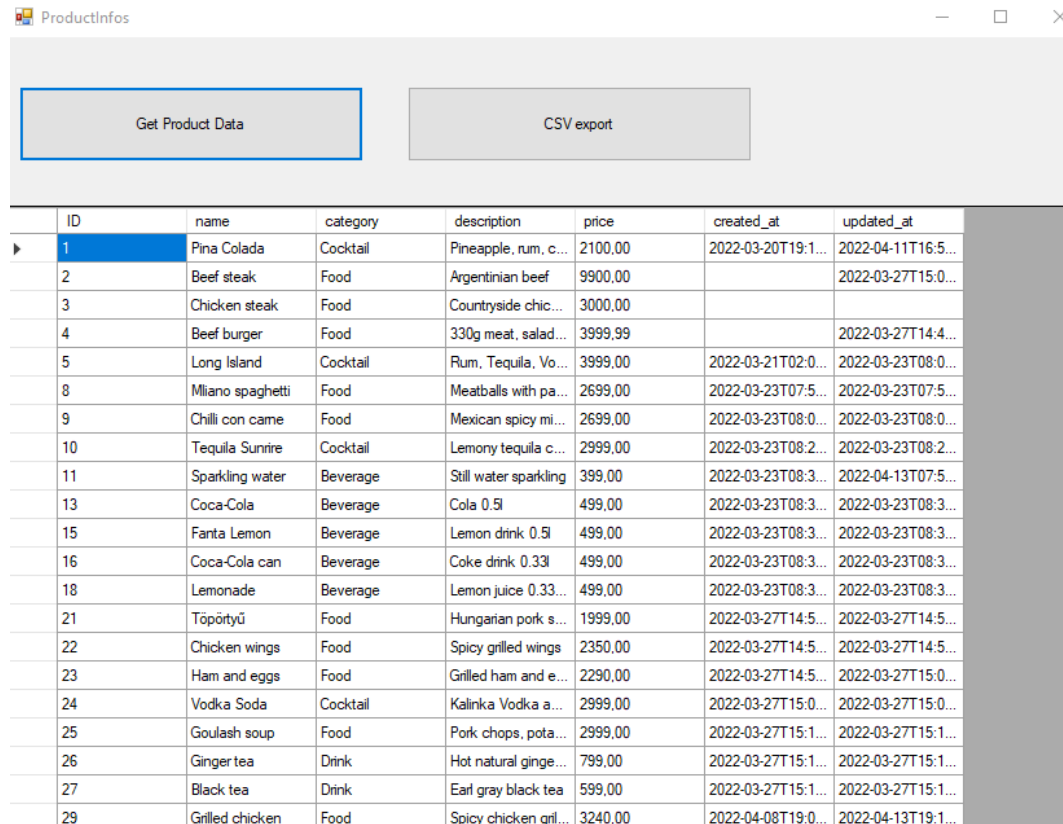
Az asztali alkalmazást a Products.exe fájlra kétszer kattintva tudjuk telepíteni és indítani a NET\_Desktop-master mappából.

Az indító alkalmazás a NET\_Desktop-master\GetProducts\bin\Debug\GetProducts.exe útvonalon érhető el. A letöltött zip fájl kibontása után.

Magát az asztali alkalmazást a <https://github.com/Citrom2021/AuthFinal-HMS> repositoryban található NET\_Desktop-master.zip néven. A zip fájlt le lehet tölteni és 7zip vagy Windows File Explorer segítségével kibontani és a kreált mappában az említett útvonalakon található exe fájlokkal telepíteni, illetve indítani az alkalmazást. Az asztali alkalmazás is jelenleg localhost-ra azaz a <http://127.0.0.1:8000/jsonproducts> útvonalra lett fejlesztve egyelőre.



A Get Product data az alábbi nézetet adja vissza:



ID	name	category	description	price	created_at	updated_at
1	Pina Colada	Cocktail	Pineapple, rum, c...	2100,00	2022-03-20T19:1...	2022-04-11T16:5...
2	Beef steak	Food	Argentinian beef	9900,00		2022-03-27T15:0...
3	Chicken steak	Food	Countryside chic...	3000,00		
4	Beef burger	Food	330g meat, salad...	3999,99		2022-03-27T14:4...
5	Long Island	Cocktail	Rum, Tequila, Vo...	3999,00	2022-03-21T02:0...	2022-03-23T08:0...
8	Milano spaghetti	Food	Meatballs with pa...	2699,00	2022-03-23T07:5...	2022-03-23T07:5...
9	Chilli con came	Food	Mexican spicy mi...	2699,00	2022-03-23T08:0...	2022-03-23T08:0...
10	Tequila Sunire	Cocktail	Lemony tequila c...	2999,00	2022-03-23T08:2...	2022-03-23T08:2...
11	Sparkling water	Beverage	Still water sparkling	399,00	2022-03-23T08:3...	2022-04-13T07:5...
13	Coca-Cola	Beverage	Cola 0.5l	499,00	2022-03-23T08:3...	2022-03-23T08:3...
15	Fanta Lemon	Beverage	Lemon drink 0.5l	499,00	2022-03-23T08:3...	2022-03-23T08:3...
16	Coca-Cola can	Beverage	Coke drink 0.33l	499,00	2022-03-23T08:3...	2022-03-23T08:3...
18	Lemonade	Beverage	Lemon juice 0.33...	499,00	2022-03-23T08:3...	2022-03-23T08:3...
21	Tőpörtyű	Food	Hungarian pork s...	1999,00	2022-03-27T14:5...	2022-03-27T14:5...
22	Chicken wings	Food	Spicy grilled wings	2350,00	2022-03-27T14:5...	2022-03-27T14:5...
23	Ham and eggs	Food	Grilled ham and e...	2290,00	2022-03-27T14:5...	2022-03-27T15:0...
24	Vodka Soda	Cocktail	Kalinka Vodka a...	2999,00	2022-03-27T15:0...	2022-03-27T15:0...
25	Goulash soup	Food	Pork chops, pota...	2999,00	2022-03-27T15:1...	2022-03-27T15:1...
26	Gingert tea	Drink	Hot natural ginge...	799,00	2022-03-27T15:1...	2022-03-27T15:1...
27	Black tea	Drink	Earl gray black tea	599,00	2022-03-27T15:1...	2022-03-27T15:1...
29	Grilled chicken	Food	Spicy chicken gril...	3240,00	2022-04-08T19:0...	2022-04-13T19:1...

Miután megkapta az asztali alkalmazás az adatokat és táblába rendezi azokat a CSV Export gombra kattintva tetszőleges néven lehet menteni az exportált csv fájlt.

Kétszer kattintva a tetszőleges néven mentett .csv fájlra megnyílik a Microsoft Excel és csv nézetben kapjuk vissza az összes termék adatot, amit tetszés szerint lehet szerkeszteni az Excel-ben.



25

POST <http://localhost:8000/api/login> Send

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies

none form-data **x-www-form-urlencoded** raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	name	Admin			
<input checked="" type="checkbox"/>	email	admin@gmail.com			
<input checked="" type="checkbox"/>	password	Admin111\$			
<input checked="" type="checkbox"/>	password_confirmation	Admin111\$			
	Key	Value	Description		

body Cookies Headers (10) Test Results Status: 200 OK Time: 859 ms Size: 788 B Save Response

Pretty Raw Preview Visualize JSON

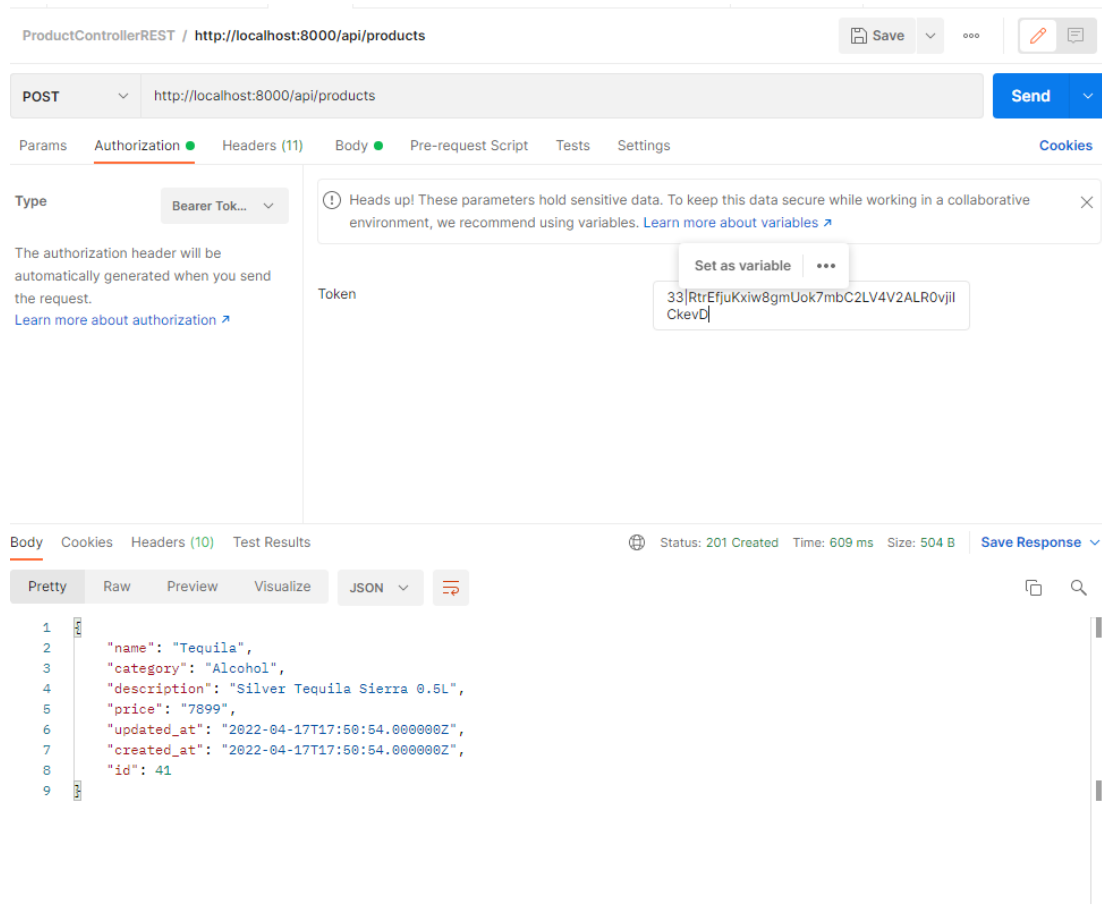
```
1 {
2   "user": {
3     "id": 1,
4     "name": "admin",
5     "email": "admin@gmail.com",
6     "role": "Admin",
7     "email_verified_at": "2022-03-20T19:07:07.000000Z",
8     "current_team_id": null,
9     "profile_photo_path": null,
10    "created_at": "2022-03-20T19:06:58.000000Z",
11    "updated_at": "2022-04-11T17:25:21.000000Z",
12    "address": "Sunset street 123, Miami, Florida",
13    "phone_number": "+36201111111",
14    "profile_photo_url": "https://ui-avatars.com/api/?name=a&color=7F9CF5&background=EBF4FF"
15  },
16  "token": "33|RtrEfjuKxiw8gmUok7mbC2LV4V2ALR0vjIckevD"
17 }
```

majd a következő http metódus pl. POST <http://localhost:8000/api/products>

Esetén az Authorization alatt a Type: Bearer Token-t kiválasztása után a Token field-be bemásolni.

A token( Sanctum) és / vagy Admin szerep által tiltott, avagy nem engedélyezett metódusok csak a token megadása után lesznek engedélyezve.

Az összes metódus ami POST, PUT, DELETE tehát a CRUD-on belül a READ-et kivéve mind szigorúan autorizációhoz vannak kötve



Visszatérve a web-re a /users útvonalon van lehetősége az adminoknak a felhasználó kezelésre.

Az admin a /users oldalon tudja a usereket kezelni. Fel tud vinni új user-t a New gombbal, a usereknek tud role-t adni (pl. Admin), tud nevet, emailt, címet és telefonszámot bevinni. A Save gombbal tudja elmenteni ezt. Ha az admin új usert visz fel akkor a bcrypt-el egy titok enkriptált jelszót is beinjektálunk az adatbázis user táblájába. Amennyiben egy user bajba kerülne és admin segítséget kérne mert jelszavát megszerezték, és a user nincs gép előtt, akkor az admin az Edit -> Reset Password gombra kattintva tud új bcrypt enkriptált jelszavat generálni.

Biztonsági okok miatt az adminok senkinek se tudják a jelszavát, és az adatbázisban is bcrypt technológia által enkriptált formátumban tároljuk a jelszavakat az adat biztonság nevében. A user táblában nincsen Delete gomb, az Admin nem tud

törölni usereket, csak a userek tudják magukat, biztonsági okokból. Az admin az Edit gombra kattintva tudja a userek adatait változtatni (name, role, email, address, phone number). Mint az összes többi edit pop-up modal boksznál úgy itt is a Save gombbal lehet menteni, míg a Cancel gombbal visszalépni és meg nem történté tenni a változtatásokat.

Minden a /users oldalon is pagination által jelenik meg, itt 20 találatot mutatva egy oldalon, ID alapján növekvő sorrendben.

A Search field-ben, azaz kereső mezőben lehetséges keresni, avagy szűrni a userek között, név(name), cím(address), szerep(role), email, telefonszám (phone number) alapján.

### 3. Fejlesztői dokumentáció

#### 3.1 A fejlesztési környezet, használt technológiák, keretrendszerek bemutatása

A fejlesztés a webes applikáció esetén az MVC(Model-View-Controller) modell alapján valósult meg a Laravel framework használatával. A weben a Livewire-t használva a Controllert a Livewire Component helyettesíti

A Laravel egy webalkalmazás-keretrendszer kifejező, elegáns szintaxissal. A webes keretrendszer struktúrát és kiindulópontot biztosít az alkalmazás létrehozásához, lehetővé téve, hogy valami csodálatos létrehozására összpontosítson, a fejlesztő ne a részleteken izzadjon.

A Laravel arra törekszik, hogy elképesztő fejlesztői élményt nyújtson, miközben olyan hatékony funkciókat biztosít, mint az alapos függőségi befecskendezés, a kifejező adatbázis-absztrakciós réteg, a várólisták és ütemezett feladatok, az egység- és integrációs tesztelés és még sok más.

Számunkra a Laravel tűnt a legjobb választás modern, full-stack webalkalmazások készítéséhez.

A Laravel robusztus eszközöket kínál a függőségi injekcióhoz, az egységtesztekhez, a várólistákhoz, a valós idejű eseményekhez és még sok másához. A Laravel finomhangolása professzionális webalkalmazások készítésére szolgál, és készen áll a vállalati munkaterhelés kezelésére.

Mivel a Laravel egy skálázható keretrendszer, ez rendkívül fontos az alkalmazás jövőbeli fejlesztése esetén akár aktuális is lehet, például:

A Laravel hihetetlenül méretezhető. A PHP méretezésbarát jellegének és a Laravel beépített támogatásának köszönhetően a gyors, elosztott gyorsítótár-rendszerekhez, mint például a Redis, a vízszintes méretezés a Laravel segítségével gyerekjáték. Valójában a Laravel-alkalmazások könnyen méretezhetők havonta több százmillió kérés kezelésére. Az olyan platformok, mint a Laravel Vapor, lehetővé teszik a Laravel alkalmazásának szinte korlátlan léptékű futtatását az AWS legújabb szerver nélküli technológiáján.

A Laravel által kínált Tailwind CSS let használva a front-end design UI-UX megvalósítására, valamint használtuk a Laravel Blade templating technológiát.

A blade az az egyszerű, de erőteljes sablonozó motor, amely a Laravelhez tartozik. Ellentétben néhány PHP sablonozó motorral, a Blade nem korlátozza a sima PHP kód használatát a sablonokban. Valójában az összes Blade-sablon sima PHP-kódba van fordítva, és a módosításig gyorsítótárban van, ami azt jelenti, hogy a Blade lényegében nulla többletköltséget jelent az alkalmazáshoz. A blade-sablonfájlok a `.blade.php` fájlkiterjesztést használják, és általában az erőforrások/nézetek könyvtárban tárolódnak.

A blade nézetek visszaadhatók útvonalakról vagy vezérlőkről a globális nézet segéd segítségével. Természetesen, amint azt a nézetek dokumentációjában említettük, az adatok átadhatók a Blade nézetnek a nézetsegéd második argumentumával.

Emellett Bootstrap 5-t egyes konténerek és a logged-out navigációs menü kezelésére, a FontAwesome library-t pedig az ikonok megjelenítésére.

Az autentikációs nézetek és bejelentkezett felhasználók kezelésének modern, reszponzív felhasználó élményt nyújtó megoldásáért a Laravel Jetstream frameworkot használtuk fel. A Laravel Jetstream egy gyönyörűen megtervezett alkalmazásindító készlet a Laravel számára, és tökéletes kiindulópontot biztosít a következő Laravel alkalmazáshoz. A Jetstream biztosítja az alkalmazás bejelentkezésének, regisztrációjának, e-mail-ellenőrzésének, kéttényezős hitelesítésének, munkamenet-kezelésének, a Laravel Sanctumon keresztüli API-nak és az opcionális csapatkezelési funkcióknak a megvalósítását. A Jetstream Tailwind CSS-t használ, és a Livewire vagy az Inertia állványok közül választhat.

A REST API-n Laravel Sanctum autentikációt használtunk fel Modellek és Controllerek megírásával. Bevezetés

A Laravel Sanctum pehelysúlyú hitelesítési rendszert biztosít SPA-k (egyoldalas alkalmazások), mobilalkalmazások és egyszerű, token alapú API-k számára. A Sanctum lehetővé teszi az alkalmazás minden felhasználójának, hogy több API tokent generáljon a fiókjához. Ezek a tokenek olyan képességeket/hatóköröket kaphatnak, amelyek meghatározzák, hogy a tokenek milyen műveleteket hajthatnak végre. A Laravel Sanctum két különálló probléma megoldására létezik. Először is, a Sanctum egy egyszerű csomag, amellyel API-tokeneket adhat ki a felhasználóknak az OAuth bonyolítása nélkül. Ezt a funkciót a GitHub és más alkalmazások ihlették, amelyek „személyes hozzáférési tokeneket” bocsátanak ki. Képzelje el például, hogy az alkalmazás „fiókbeállítási” között van egy képernyő, ahol a felhasználó API-tokent hozhat létre a fiókjához. A Sanctum segítségével létrehozhatja és kezelheti ezeket a tokeneket. Ezeknek a tokeneknek általában nagyon hosszú a lejárat ideje (év), de a felhasználó manuálisan bármikor visszavonhatja őket. A Laravel Sanctum ezt a szolgáltatást úgy kínálja, hogy egyetlen adatbázistáblában tárolja a felhasználói API-tokeneket, és hitelesíti a bejövő HTTP-kéréseket az engedélyezési fejlécen keresztül, amelynek tartalmaznia kell egy érvényes API tokent. Másodszor, a Sanctum azért létezik, hogy egyszerű módot kínáljon az egyoldalas alkalmazások (SPA-k) hitelesítésére, amelyeknek kommunikálniuk kell egy Laravel által üzemeltetett API-val. Ezek a SPA-k

létezhetnek ugyanabban a lerakatban, mint a Laravel-alkalmazás, vagy lehetnek teljesen különálló táruk, például a Vue CLI vagy egy Next.js alkalmazás segítségével létrehozott SPA. Ehhez a funkcióhoz a Sanctum nem használ semmilyen jelzőt. Ehelyett a Sanctum a Laravel beépített cookie-alapú munkamenet-hitelesítési szolgáltatásait használja. A Sanctum általában a Laravel webes hitelesítési órét használja ennek megvalósítására. Ez biztosítja a CSRF-védelem, a munkamenet-hitelesítés előnyeit, valamint védelmet nyújt a hitelesítési adatok XSS-en keresztüli kiszivárgása ellen. A Sanctum csak akkor kísérli meg a hitelesítést cookie-k használatával, ha a bejövő kérés az Ön saját SPA frontendjétől származik. Amikor a Sanctum megvizsgál egy bejövő HTTP-kérélmét, először ellenőrzi, hogy van-e hitelesítési cookie, és ha ilyen nincs, akkor a Sanctum megvizsgálja, hogy van-e érvényes API-token az engedélyezési fejlécben.

A weben egyes esetekben kontrollerek, de javában Modellek és Laravel Livewire Components használatával írtuk meg a kódot, amivel manipuláljuk azt, hogy a weben mi történjen nézetekkel és adatokkal.

A Laravel Livewire egy olyan könyvtár, amely egyszerűvé teszi modern, reaktív, dinamikus interfészek létrehozását a Laravel Blade sablonnyelvként. Ez egy nagyszerű halom, ha olyan alkalmazást szeretne létrehozni, amely dinamikus és reaktív, de nem érzi magát kényelmesen egy teljes JavaScript-keretrendszerbe, például a Vue-ba. A Livewire a kezdeti komponens kimenetet az oldallal együtt jeleníti meg (mint a Blade is). Így SEO-barát. Interakció esetén a Livewire AJAX kérést küld a szervernek a frissített adatokkal. A szerver újra rendereli az összetevőt, és az új HTML-kóddal válaszol. A Livewire ezután intelligensen mutálja a DOM-ot a megváltozott dolgoknak megfelelően. A Livewire-re vonatkozó összes kérés a szerver oldalon jelenik meg. A szerveroldali alkalmazásokkal a jelölés és az adatok összeállításra kerülnek a szerveren, mielőtt kiszolgálnák azokat a kliensnek.

A hiba üzenetek megjelenítésére flash error messageket írtunk és ezeket használtuk fel és jelenítetjük meg a felhasználó számára.

A Jetstreamben használt autentikációt a Fortify segítségével valósítottuk meg. A Laravel Fortify egy frontend agnosztikus hitelesítési háttér megvalósítás a Laravel számára. A Fortify regisztrálja a Laravel összes hitelesítési funkciójának megvalósításához szükséges útvonalakat és vezérlőket, beleértve a bejelentkezést,

regisztrációt, jelszó-visszaállítást, e-mail-ellenőrzést és még sok más. A motorháztető alatt a Jetstream hitelesítési részeit a Laravel Fortify hajtja, amely a Laravel front-end agnosztikus hitelesítési háttere. Lényegében a Fortify határozza meg az útvonalakat és a vezérlőket az alkalmazás hitelesítési funkcióinak megvalósításához, miközben a Jetstream felhasználói felület kéréseket küld ezekre az útvonalakra. A Jetstream telepítésekor a config/fortify.php konfigurációs fájl telepítésre kerül az alkalmazásba. Ebben a konfigurációs fájlban testreszabtuk a Fortify viselkedésének különböző aspektusait, például a használandó hitelesítési őr, ahová a felhasználókat a hitelesítés után át kell irányítani, és így tovább. A fortify konfigurációs fájlban a Fortify funkcióit engedélyeztük, például a profiladatok vagy jelszavak frissítésének lehetőségét:

```
'features' => [  
    Features::registration(),  
    Features::resetPasswords(),  
    Features::emailVerification(),  
    Features::updateProfileInformation(),  
    Features::updatePasswords(),  
    Features::twoFactorAuthentication([  
        'confirmPassword' => true,  
    ]),  
],  
];
```

## 3.2 A webes projektben felhasznált technológiák és verziók

### PHP:

PHP 8.0.12 (cli) (built: Oct 19 2021 11:21:05) ( ZTS Visual C++ 2019 x64 )

Copyright (c) The PHP Group

Zend Engine v4.0.12, Copyright (c) Zend Technologies



## Laravel:

### Laravel Framework 8.83.5

Többi felhasznált technológia és verzió:

```
"name": "laravel/laravel",
"type": "project",
"description": "The Laravel Framework.",
"keywords": ["framework", "laravel"],
"license": "MIT",
"require": {
    "php": "^7.3|^8.0",
    "doctrine/dbal": "^3.3",
    "fideloper/proxy": "^4.4",
    "fruitcake/laravel-cors": "^2.0",
    "guzzlehttp/guzzle": "^7.0.1",
    "kyslik/column-sortable": "^6.4",
    "laravel/framework": "^8.12",
    "laravel/jetstream": "^2.6",
    "laravel/sanctum": "^2.9",
    "laravel/tinker": "^2.5",
    "laravel/ui": "^3.0",
    "livewire/livewire": "^2.5"
},
"require-dev": {
    "facade/ignition": "^2.5",
    "fakerphp/faker": "^1.9.1",
    "laravel/sail": "^1.0.1",
    "mockery/mockery": "^1.4.2",
    "nunomaduro/collision": "^5.0",
    "phpunit/phpunit": "^9.3.3"
},
"autoload": {
    "psr-4": {
        "App\\": "app/",
        "Database\\Factories\\": "database/factories/",
        "Database\\Seeders\\": "database/seeders/"
    }
},
"autoload-dev": {
    "psr-4": {
        "Tests\\": "tests/"
    }
}
```

```
},
"scripts": {
  "post-autoload-dump": [
    "Illuminate\\Foundation\\ComposerScripts::postAutoloadDump",
    "@php artisan package:discover --ansi"
  ],
  "post-root-package-install": [
    "@php -r \"file_exists('.env') || copy('.env.example', '.env');\""
  ],
  "post-create-project-cmd": [
    "@php artisan key:generate --ansi"
  ]
},
"extra": {
  "laravel": {
    "dont-discover": []
  }
},
"config": {
  "optimize-autoloader": true,
  "preferred-install": "dist",
  "sort-packages": true
},
"minimum-stability": "dev",
"prefer-stable": true
}
```

### 3.3 Az asztali alkalmazásban használt technológiák és funkciók

.NET Framework 4.8-ban történt a fejlesztés, ami C# programnyelvben Visual Studio IDE által lett írva.

Használtuk a Data Grid view osztályt ami az adatokat testre szabható rácsban jeleníti meg a System.Windows.Forms namespace-en belül (A namespace (névtér) segítségével lehet globális tereket képezni a láthatósági feltételekhez, a könnyebb kódkarbantartáshoz/átláthatósághoz, és a globális típusdeklarációhoz. C#

esetében a .NET keretrendszer miatt ez egészen az operációs rendszer szintjéig megtalálható, és nyelvek közötti átjárásra is lehetőséget ad.)

#### namespace System.Windows.Forms;

A System.Windows.Forms egy UI keretrendszer Windows asztali alkalmazások létrehozásához. Ez az egyik legproduktívabb módja az asztali alkalmazások létrehozásának a Visual Studio látványtervezője alapján. Az olyan funkciók, mint a vizuális vezérlők fogd és vidd elhelyezése, megkönnyítik az asztali alkalmazások létrehozását

#### namesapce System.Net.Http;

A System.Net.Http Programozási felületet biztosít a modern HTTP-alkalmazásokhoz, beleértve azokat a HTTP-kliens-összetevőket is, amelyek lehetővé teszik az alkalmazások számára, hogy webszolgáltatásokat HTTP-n keresztül használhassanak, és HTTP-összetevőket is használhatnak az ügyfelek és a kiszolgálók HTTP-fejlécek elemzésére.

#### namespace System.Collections.Generic;

A System.Collections.Generic olyan interfészeket és osztályokat tartalmaz, amelyek általános gyűjteményeket határoznak meg, amelyek lehetővé teszik a felhasználók számára, hogy olyan erősen tipizált gyűjteményeket hozzanak létre, amelyek jobb típusbiztonságot és teljesítményt nyújtanak, mint a nem általános, erősen tipizált gyűjtemények.

#### namespace System.IO;

Olyan típusokat tartalmaz, amelyek lehetővé teszik a fájlok és adatfolyamok olvasását és írását, valamint olyan típusokat, amelyek alapvető fájl- és könyvtártámogatást biztosítanak

#### namespace System.Linq;

Osztályokat és felületeket biztosít, amelyek támogatják a nyelvbe integrált lekérdezést (LINQ) használó lekérdezéseket.

System.Text Namespace

Osztályokat tartalmaz, amelyek ASCII és Unicode karakterkódolást képviselnek; absztrakt alaposztályok karakterblokkok konvertálásához bájtblokkokká és bájtblokkokból; és egy segítő osztály, amely a String objektumokat úgy kezeli és formázza, hogy közben nem hoz létre String példányokat.

### 3.4 Adatbázis, adattáblák felépítése és diagram

A laravel adatbázisban a products tábla az, ami termékek tárolására van használva. Ennek oszlopai az

'id','name','category','description','price','created\_at','update\_at'. Ezek a 2021\_03\_30\_203216\_create\_products\_table Laravel migráció hozta létre, a Product model adja az alapját, a ProductController és az AuthController határozza meg hogy mi történhet ebben a táblában az api.php route útvonalon keresztül. A web route útvonalon keresztül az app\Http\Livewire\Products.php illetve a app\Http\Livewire\UserProduct.php Component-ek a weben renderelik a termékek adatait ezzel a táblával kommunikálva és a viewnak továbbadva azt a restaurantandbar.blade.php, products.blade.php, create.blade.php és vice versa. Ezt a táblát használja és szolgálja ki a REST API is az api.php route által.

A foglalások tárolására és azonosítására a transactions és rooms táblát használjuk. A transactions táblában a PRIMARY KEY az id, míg FOREIGN KEY-nek a user\_id-t használjuk ami követi a `users`.`id` azaz a users tábla PRIMARY KEY-ét és a room\_id-t ami használjuk ami követi a `rooms`.`id` azaz a rooms tábla id-t annak a PRIMARY KEY-ét. A `users`.`id` alapján a felhasználót kezeljük és azonosítjuk a transaction táblában, míg a `rooms`.`id` alapján a szobát és annak nevét illetve árát kezeljük és azonosítjuk.

A felhasználók, beleértve az adminok - adatai a users táblában vannak tárolva. A jelszavakat bcrypt titkosítással tároljuk.

A session táblát a Laravel generálja mivel a HTTP-alapú alkalmazások állapotatlanok, a munkamenetek lehetőséget biztosítanak a felhasználó információinak tárolására több kérésben. Ezeket a felhasználói információkat általában egy állandó tárolóban/háttérrendszerben helyezik el, amely ebből a táblából érhető el.

A personal\_access\_tokens táblában a személyes hozzáférési tokenek vannak tárolva, mivel a products táblát az API-n keresztül is lehet használni, így ezeknek a userknek a tokenjének tárolására használjuk, leginkább hitelesítésre. A tokeneket a Laravel Sanctum segítségével hozzuk létre.

### 3.5 Útvonalak listája

A `route:list` paranccsal megjeleníthető az alkalmazáshoz regisztrált összes útvonal lista. Ez a parancs megjeleníti a tartományt, metódust, URI-t, nevet, műveletet és köztes szoftvert(middleware) a generált táblázatban szereplő útvonalakhoz.

Természetesen ezt még formázni is kell. Mi a weben a

<http://127.0.0.1:8000/routes> útvonalon táblában jelenítjük meg ezeket, illetve

<http://127.0.0.1:8000/routestocsv> úton CSV formátumban le töltődik a fájl. A következő tábla bemutatja alkalmazásunk útvonalait.

A domain oszlop üres hiszen nincsen domain-ünk, csak localhostot használunk, a method bemutatja, hogy a CRUD operations-nek megfelelő melyik HTTP method van használva. A URI (Uniform Resource Identifier) megadja az elérést. Általában a URI két részhalmazt tartalmaz, az URN-t, amely megmondja a nevet, és az URL-t, amely megmondja a helyet.

A name oszlop nyilvánvalóan a nevét adja vissza.

Az action oszlop pedig azt, hogy melyik Controller melyik metódusa, funkciója avagy melyik Livewire Component van jelen.

A middleware oszlop pedig a köztes szoftvereket mutatja be. A laravelben ezek mechanizmust biztosítanak az alkalmazásba érkező HTTP-kérések ellenőrzéséhez és szűréséhez. Például a Laravel tartalmaz egy köztes szoftvert, amely ellenőrzi az alkalmazás felhasználójának hitelesítését. Ha a felhasználó nincs hitelesítve, a köztes szoftver átirányítja a felhasználót az alkalmazás bejelentkezési képernyőjére. Ha azonban a felhasználó hitelesített, a köztes szoftver lehetővé teszi, hogy a kérés továbbhaladjon az alkalmazásban. A Laravel keretrendszer számos köztes szoftvert tartalmaz, köztük a hitelesítést és a CSRF védelmet szolgáló köztes szoftvereket. Mindezek a köztes szoftverek az `app/Http/Middleware` könyvtárban találhatók.

domain	method	uri	name	action	middleware
	GET HEAD	/		Closure	web
	POST	api/login		App\Http\Controllers\AuthController@login	api
	POST	api/logout		App\Http\Controllers\AuthController@logout	api
	POST	api/logout		App\Http\Controllers\AuthController@logout	App\Http\Middleware\Authenticate:sanctum
	GET HEAD	api/products		App\Http\Controllers\ProductController@index	api
	POST	api/products		App\Http\Controllers\ProductController@store	api
	POST	api/products		App\Http\Controllers\ProductController@store	App\Http\Middleware\Authenticate:sanctum
	GET HEAD	api/products/search/{name}		App\Http\Controllers\ProductController@search	api
	GET HEAD	api/products/{id}		App\Http\Controllers\ProductController@show	api
	PUT	api/products/{id}		App\Http\Controllers\ProductController@update	api
	PUT	api/products/{id}		App\Http\Controllers\ProductController@update	App\Http\Middleware\Authenticate:sanctum
	DELETE	api/products/{id}		App\Http\Controllers\ProductController@destroy	api
	DELETE	api/products/{id}		App\Http\Controllers\ProductController@destroy	App\Http\Middleware\Authenticate:sanctum
	POST	api/register		App\Http\Controllers\AuthController@register	api
	GET HEAD	api/user		Closure	api
	GET HEAD	api/user		Closure	App\Http\Middleware\Authenticate:sanctum
	GET HEAD	dashboard	dashboard	Closure	web
	GET HEAD	dashboard	dashboard	Closure	App\Http\Middleware\Authenticate:sanctum
	GET HEAD	dashboard	dashboard	Closure	Illuminate\Auth\Middleware\EnsureEmailIsVerified
	POST	email/verification-notification	verification.send	Laravel\Fortify\Http\Controllers>EmailVerificationNotificationController@store	web
	POST	email/verification-notification	verification.send	Laravel\Fortify\Http\Controllers>EmailVerificationNotificationController@store	App\Http\Middleware\Authenticate:web
	POST	email/verification-notification	verification.send	Laravel\Fortify\Http\Controllers>EmailVerificationNotificationController@store	Illuminate\Routing\Middleware\ThrottleRequests:6,1
	GET HEAD	email/verify	verification.notice	Closure	web
	GET HEAD	email/verify	verification.notice	Closure	App\Http\Middleware\Authenticate
	GET HEAD	email/verify/{id}/{hash}	verification.verify	Laravel\Fortify\Http\Controllers\VerifyEmailController@__invoke	web
	GET HEAD	email/verify/{id}/{hash}	verification.verify	Laravel\Fortify\Http\Controllers\VerifyEmailController@__invoke	App\Http\Middleware\Authenticate:web
	GET HEAD	email/verify/{id}/{hash}	verification.verify	Laravel\Fortify\Http\Controllers\VerifyEmailController@__invoke	Illuminate\Routing\Middleware\ValidateSignature
	GET HEAD	email/verify/{id}/{hash}	verification.verify	Laravel\Fortify\Http\Controllers\VerifyEmailController@__invoke	Illuminate\Routing\Middleware\ThrottleRequests:6,1
	GET HEAD	forgot-password	password.request	Laravel\Fortify\Http\Controllers>PasswordResetLinkController@create	web
	GET HEAD	forgot-password	password.request	Laravel\Fortify\Http\Controllers>PasswordResetLinkController@create	App\Http\Middleware\RedirectIfAuthenticated:web
	POST	forgot-password	password.email	Laravel\Fortify\Http\Controllers>PasswordResetLinkController@store	web
	POST	forgot-password	password.email	Laravel\Fortify\Http\Controllers>PasswordResetLinkController@store	App\Http\Middleware\RedirectIfAuthenticated:web
	GET HEAD	index		Closure	web
	GET HEAD	jsonproducts		App\Http\Controllers\ProductController@index	web
	GET HEAD	livewire/livewire.js		Livewire\Controllers\LivewireJavaScriptAssets@source	
	GET HEAD	livewire/livewire.js.map		Livewire\Controllers\LivewireJavaScriptAssets@maps	
	POST	livewire/message/{name}	livewire.message	Livewire\Controllers\HttpConnectionHandler	web
	GET HEAD	livewire/preview-file/{filename}	livewire.preview-file	Livewire\Controllers\FilePreviewHandler@handle	web
	POST	livewire/upload-file	livewire.upload-file	Livewire\Controllers\FileUploadHandler@handle	web

Welsch Ádám Tamás Csizmár Dániel Ferenc Szikszi Ákos			Vizsgaremek		2022.04.18
POST	livewire/upload-file	livewire.upload-file	Livewire\Controllers\FileUploadHandler@handle	Illuminate\Routing\Middleware\ThrottleRequests:60,1	
GET HEAD	login	login	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@create	web	
GET HEAD	login	login	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@create	App\Http\Middleware\RedirectIfAuthenticated:web	
POST	login		Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@store	web	
POST	login		Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@store	App\Http\Middleware\RedirectIfAuthenticated:web	
POST	login		Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@store	Illuminate\Routing\Middleware\ThrottleRequests:login	
POST	logout	logout	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@destroy	web	
GET HEAD	my-bookings		App\Http\Livewire\UserTransactions	web	
GET HEAD	my-bookings		App\Http\Livewire\UserTransactions	App\Http\Middleware\Authenticate:sanctum	
GET HEAD	my-bookings		App\Http\Livewire\UserTransactions	Illuminate\Auth\Middleware\EnsureEmailIsVerified	
GET HEAD	privacy-policy	policy.show	Laravel\Jetstream\Http\Controllers\Livewire\PrivacyPolicyController@show	web	
GET HEAD	products		App\Http\Livewire\Products	web	
GET HEAD	products		App\Http\Livewire\Products	App\Http\Middleware\Authenticate	
GET HEAD	products		App\Http\Livewire\Products	App\Http\Middleware\AdminMiddleware	
GET HEAD	register	register	Laravel\Fortify\Http\Controllers\RegisteredUserController@create	web	
GET HEAD	register	register	Laravel\Fortify\Http\Controllers\RegisteredUserController@create	App\Http\Middleware\RedirectIfAuthenticated:web	
POST	register		Laravel\Fortify\Http\Controllers\RegisteredUserController@store	web	
POST	register		Laravel\Fortify\Http\Controllers\RegisteredUserController@store	App\Http\Middleware\RedirectIfAuthenticated:web	
POST	reset-password	password.update	Laravel\Fortify\Http\Controllers\NewPasswordController@store	web	
POST	reset-password	password.update	Laravel\Fortify\Http\Controllers\NewPasswordController@store	App\Http\Middleware\RedirectIfAuthenticated:web	
GET HEAD	reset-password/{token}	password.reset	Laravel\Fortify\Http\Controllers\NewPasswordController@create	web	
GET HEAD	reset-password/{token}	password.reset	Laravel\Fortify\Http\Controllers\NewPasswordController@create	App\Http\Middleware\RedirectIfAuthenticated:web	
GET HEAD	restaurantandbar		App\Http\Livewire\UserProducts	web	
GET HEAD	restaurantandbar		App\Http\Livewire\UserProducts	App\Http\Middleware\Authenticate:sanctum	
GET HEAD	restaurantandbar		App\Http\Livewire\UserProducts	Illuminate\Auth\Middleware\EnsureEmailIsVerified	
GET HEAD	routes		Closure	web	
GET HEAD	routes		Closure	App\Http\Middleware\Authenticate	
GET HEAD	routes		Closure	App\Http\Middleware\AdminMiddleware	
GET HEAD	routeestocsv		Closure	web	
GET HEAD	routeestocsv		Closure	App\Http\Middleware\Authenticate	
GET HEAD	routeestocsv		Closure	App\Http\Middleware\AdminMiddleware	
GET HEAD	sanctum/csrf-cookie		Laravel\Sanctum\Http\Controllers\CsrfCookieController@show	web	
GET HEAD	terms-of-service	terms.show	Laravel\Jetstream\Http\Controllers\Livewire\TermsOfServiceController@show	web	
GET HEAD	transactions		App\Http\Livewire\Transactions	web	
GET HEAD	transactions		App\Http\Livewire\Transactions	App\Http\Middleware\Authenticate	
GET HEAD	transactions		App\Http\Livewire\Transactions	App\Http\Middleware\AdminMiddleware	
GET HEAD	two-factor-challenge	two-factor.login	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@create	web	
GET HEAD	two-factor-challenge	two-factor.login	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@create	App\Http\Middleware\RedirectIfAuthenticated:web	
POST	two-factor-challenge		Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@store	web	
POST	two-factor-challenge		Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@store	App\Http\Middleware\RedirectIfAuthenticated:web	
POST	two-factor-challenge		Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@store	Illuminate\Routing\Middleware\ThrottleRequests:two-factor	



GET HEAD	user/confirm-password		Laravel\Fortify\Http\Controllers\ConfirmablePasswordController@show	web
GET HEAD	user/confirm-password		Laravel\Fortify\Http\Controllers\ConfirmablePasswordController@show	App\Http\Middleware\Authenticate:web
POST	user/confirm-password	password.confirm	Laravel\Fortify\Http\Controllers\ConfirmablePasswordController@store	web
POST	user/confirm-password	password.confirm	Laravel\Fortify\Http\Controllers\ConfirmablePasswordController@store	App\Http\Middleware\Authenticate:web
GET HEAD	user/confirmed-password-status	password.confirmation	Laravel\Fortify\Http\Controllers\ConfirmedPasswordStatusController@show	web
GET HEAD	user/confirmed-password-status	password.confirmation	Laravel\Fortify\Http\Controllers\ConfirmedPasswordStatusController@show	App\Http\Middleware\Authenticate:web
POST	user/confirmed-two-factor-authentication	two-factor.confirm	Laravel\Fortify\Http\Controllers\ConfirmedTwoFactorAuthenticationController@store	web
POST	user/confirmed-two-factor-authentication	two-factor.confirm	Laravel\Fortify\Http\Controllers\ConfirmedTwoFactorAuthenticationController@store	App\Http\Middleware\Authenticate:web
POST	user/confirmed-two-factor-authentication	two-factor.confirm	Laravel\Fortify\Http\Controllers\ConfirmedTwoFactorAuthenticationController@store	Illuminate\Auth\Middleware\RequirePassword
GET HEAD	user/home		App\Http\Controllers\HomeController@UservsAdmin	web
GET HEAD	user/home		App\Http\Controllers\HomeController@UservsAdmin	Illuminate\Auth\Middleware\EnsureEmailIsVerified
PUT	user/password	user-password.update	Laravel\Fortify\Http\Controllers>PasswordController@update	web
PUT	user/password	user-password.update	Laravel\Fortify\Http\Controllers>PasswordController@update	App\Http\Middleware\Authenticate:web
GET HEAD	user/profile	profile.show	Laravel\Jetstream\Http\Controllers\Livewire\UserProfileController@show	web
GET HEAD	user/profile	profile.show	Laravel\Jetstream\Http\Controllers\Livewire\UserProfileController@show	App\Http\Middleware\Authenticate:sanctum
GET HEAD	user/profile	profile.show	Laravel\Jetstream\Http\Controllers\Livewire\UserProfileController@show	Illuminate\Auth\Middleware\EnsureEmailIsVerified
PUT	user/profile-information	user-profile-information.update	Laravel\Fortify\Http\Controllers\ProfileInformationController@update	web
PUT	user/profile-information	user-profile-information.update	Laravel\Fortify\Http\Controllers\ProfileInformationController@update	App\Http\Middleware\Authenticate:web
POST	user/two-factor-authentication	two-factor.enable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@store	web
POST	user/two-factor-authentication	two-factor.enable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@store	App\Http\Middleware\Authenticate:web
POST	user/two-factor-authentication	two-factor.enable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@store	Illuminate\Auth\Middleware\RequirePassword
DELETE	user/two-factor-authentication	two-factor.disable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@destroy	web
DELETE	user/two-factor-authentication	two-factor.disable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@destroy	App\Http\Middleware\Authenticate:web
DELETE	user/two-factor-authentication	two-factor.disable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@destroy	Illuminate\Auth\Middleware\RequirePassword
GET HEAD	user/two-factor-qr-code	two-factor.qr-code	Laravel\Fortify\Http\Controllers\TwoFactorQrCodeController@show	web
GET HEAD	user/two-factor-qr-code	two-factor.qr-code	Laravel\Fortify\Http\Controllers\TwoFactorQrCodeController@show	App\Http\Middleware\Authenticate:web
GET HEAD	user/two-factor-qr-code	two-factor.qr-code	Laravel\Fortify\Http\Controllers\TwoFactorQrCodeController@show	Illuminate\Auth\Middleware\RequirePassword
GET HEAD	user/two-factor-recovery-codes	two-factor.recovery-codes	Laravel\Fortify\Http\Controllers\RecoveryCodeController@index	web
GET HEAD	user/two-factor-recovery-codes	two-factor.recovery-codes	Laravel\Fortify\Http\Controllers\RecoveryCodeController@index	App\Http\Middleware\Authenticate:web
GET HEAD	user/two-factor-recovery-codes	two-factor.recovery-codes	Laravel\Fortify\Http\Controllers\RecoveryCodeController@index	Illuminate\Auth\Middleware\RequirePassword
POST	user/two-factor-recovery-codes		Laravel\Fortify\Http\Controllers\RecoveryCodeController@store	web
POST	user/two-factor-recovery-codes		Laravel\Fortify\Http\Controllers\RecoveryCodeController@store	App\Http\Middleware\Authenticate:web
POST	user/two-factor-recovery-codes		Laravel\Fortify\Http\Controllers\RecoveryCodeController@store	Illuminate\Auth\Middleware\RequirePassword
GET HEAD	users		App\Http\Livewire\Users	web
GET HEAD	users		App\Http\Livewire\Users	App\Http\Middleware\Authenticate
GET HEAD	users		App\Http\Livewire\Users	App\Http\Middleware\AdminMiddleware



Egy egyszerűsített nézet a route list-ről Middleware nélkül:

METHOD	URI	NAME	ACTION
GET	_ignition/health-check	ignition.healthCheck	Facade\Ignition\Http\Controllers\HealthCheckController
POST	_ignition/execute-solution	ignition.executeSolution	Facade\Ignition\Http\Controllers\ExecuteSolutionController
POST	_ignition/share-report	ignition.shareReport	Facade\Ignition\Http\Controllers\ShareReportController
GET	_ignition/scripts/{script}	ignition.scripts	Facade\Ignition\Http\Controllers\ScriptController
GET	_ignition/styles/{style}	ignition.styles	Facade\Ignition\Http\Controllers\StyleController
GET	login	login	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@create
POST	login		Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@store
POST	logout	logout	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController@destroy
GET	forgot-password	password.request	Laravel\Fortify\Http\Controllers>PasswordResetLinkController@create
GET	reset-password/{token}	password.reset	Laravel\Fortify\Http\Controllers\NewPasswordController@create
POST	forgot-password	password.email	Laravel\Fortify\Http\Controllers>PasswordResetLinkController@store
POST	reset-password	password.update	Laravel\Fortify\Http\Controllers\NewPasswordController@store
GET	register	register	Laravel\Fortify\Http\Controllers\RegisteredUserController@create
POST	register		Laravel\Fortify\Http\Controllers\RegisteredUserController@store
GET	email/verify	verification.notice	Closure
GET	email/verify/{id}/{hash}	verification.verify	Laravel\Fortify\Http\Controllers\VerifyEmailController@__invoke
POST	email/verification-notification	verification.send	Laravel\Fortify\Http\Controllers>EmailVerificationNotificationController@store
PUT	user/profile-information	user-profile-information.update	Laravel\Fortify\Http\Controllers\ProfileInformationController@update
PUT	user/password	user-password.update	Laravel\Fortify\Http\Controllers>PasswordController@update
GET	user/confirm-password		Laravel\Fortify\Http\Controllers\ConfirmablePasswordController@show
GET	user/confirmed-password-status	password.confirmation	Laravel\Fortify\Http\Controllers\ConfirmedPasswordStatusController@show
POST	user/confirm-password	password.confirm	Laravel\Fortify\Http\Controllers\ConfirmablePasswordController@store
GET	two-factor-challenge	two-factor.login	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@create
POST	two-factor-challenge		Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController@store
POST	user/two-factor-authentication	two-factor.enable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@store
POST	user/confirmed-two-factor-authentication	two-factor.confirm	Laravel\Fortify\Http\Controllers\ConfirmedTwoFactorAuthenticationController@store
DELETE	user/two-factor-authentication	two-factor.disable	Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController@destroy
GET	user/two-factor-qr-code	two-factor.qr-code	Laravel\Fortify\Http\Controllers\TwoFactorQrCodeController@show
GET	user/two-factor-recovery-codes	two-factor.recovery-codes	Laravel\Fortify\Http\Controllers\RecoveryCodeController@index
POST	user/two-factor-recovery-codes		Laravel\Fortify\Http\Controllers\RecoveryCodeController@store
GET	terms-of-service	terms.show	Laravel\Jetstream\Http\Controllers\Livewire\TermsOfServiceController@show
GET	privacy-policy	policy.show	Laravel\Jetstream\Http\Controllers\Livewire\PrivacyPolicyController@show
GET	user/profile	profile.show	Laravel\Jetstream\Http\Controllers\Livewire\UserProfileController@show
GET	sanctum/csrf-cookie		Laravel\Sanctum\Http\Controllers\CsrfCookieController@show

POST	livewire/message/{ name }	livewire.message	Livewire\Controllers\HttpConnectionHandler
POST	livewire/upload-file	livewire.upload-file	Livewire\Controllers\FileUploadHandler@handle
GET	livewire/preview-file/{ filename }	livewire.preview-file	Livewire\Controllers\FilePreviewHandler@handle
GET	livewire/livewire.js		Livewire\Controllers\LivewireJavaScriptAssets@source
GET	livewire/livewire.js.map		Livewire\Controllers\LivewireJavaScriptAssets@maps
POST	api/login		App\Http\Controllers\AuthController@login
POST	api/register		App\Http\Controllers\AuthController@register
GET	api/products		App\Http\Controllers\ProductController@index
GET	api/products/{ id }		App\Http\Controllers\ProductController@show
GET	api/products/search/{ name }		App\Http\Controllers\ProductController@search
POST	api/products		App\Http\Controllers\ProductController@store
PUT	api/products/{ id }		App\Http\Controllers\ProductController@update
DELETE	api/products/{ id }		App\Http\Controllers\ProductController@destroy
POST	api/logout		App\Http\Controllers\AuthController@logout
GET	api/user		Closure
GET	/		Closure
GET	index		Closure
GET	products		App\Http\Livewire\Products
GET	transactions		App\Http\Livewire\Transactions
GET	restaurantandbar		App\Http\Livewire\UserProducts
GET	my-bookings		App\Http\Livewire\UserTransactions
GET	dashboard	dashboard	Closure
GET	user/home		App\Http\Controllers\HomeController@UservsAdmin
GET	users		App\Http\Livewire\Users
GET	routes		Closure
GET	roustocsv		Closure
GET	jsonproducts		App\Http\Controllers\ProductController@index

## 4. Tesztelés





### API dokumentációk elérése a weben:

1. Swagger:
  - a. [https://app.swaggerhub.com/apis-docs/wokres/product-controller\\_rest/1.0.0](https://app.swaggerhub.com/apis-docs/wokres/product-controller_rest/1.0.0)
2. Postman
  - a. <https://documenter.getpostman.com/view/19126249/UVyxRZv2#ea026aee-00a1-4287-9517-85a8d58b8977>

### ProductController OPENAPI 3.0 specifikáció:

```
openapi: 3.0.0
info:
  title: ProductControllerREST
  version: 1.0.0
servers:
  - url: http://localhost:8000
components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
    noauthAuth:
      type: http
      scheme: noauth
paths:
  /api/products:
    post:
      tags:
        - default
      summary: http://localhost:8000/api/products
      requestBody:
        content:
          application/x-www-form-urlencoded:
            schema:
              properties:
                name:
```

```
        type: string
        example: Gingerbread
    category:
        type: string
        example: Food
    description:
        type: string
        example: Sweet bread, allergic
    price:
        type: integer
        example: '955'
security:
  - bearerAuth: []
parameters:
  - name: Accept
    in: header
    schema:
      type: string
      example: application/json
responses:
  '201':
    description: Created
    headers:
      Host:
        schema:
          type: string
          example: localhost:8000
      Date:
        schema:
          type: string
          example: Fri, 08 Apr 2022 20:04:53 GMT
      Connection:
        schema:
          type: string
          example: close
      X-Powered-By:
        schema:
          type: number
          example: PHP/8.0.12
      Cache-Control:
        schema:
```

```
      type: string
      example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '58'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      name: Ketchup
      category: Food
      description: Small portion of ketchup
      price: '248'
      updated_at: '2022-04-08T20:04:53.000000Z'
      created_at: '2022-04-08T20:04:53.000000Z'
      id: 31
'401':
  description: Unauthorized
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Fri, 08 Apr 2022 19:46:25 GMT
```



```
Connection:
  schema:
    type: string
    example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      message: Unauthenticated.
'403':
  description: Forbidden
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Sat, 09 Apr 2022 18:40:36 GMT
  Connection:
    schema:
      type: string
      example: close
```

```
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '59'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      message: Forbidden
get:
  tags:
    - default
  summary: http://localhost:8000/api/products
  security:
    - bearerAuth: []
  parameters:
    - name: Accept
      in: header
      schema:
        type: string
```

```
    example: application/json
  responses:
    '200':
      description: OK
      headers:
        Host:
          schema:
            type: string
            example: localhost:8000
        Date:
          schema:
            type: string
            example: Fri, 08 Apr 2022 20:02:33 GMT
        Connection:
          schema:
            type: string
            example: close
        X-Powered-By:
          schema:
            type: number
            example: PHP/8.0.12
        Cache-Control:
          schema:
            type: string
            example: no-cache, private
        Content-Type:
          schema:
            type: string
            example: application/json
        X-RateLimit-Limit:
          schema:
            type: integer
            example: '60'
        X-RateLimit-Remaining:
          schema:
            type: integer
            example: '59'
        Access-Control-Allow-Origin:
          schema:
            type: string
            example: '*'
```

```
content:
  application/json:
    schema:
      type: object
    example:
      - id: 1
        name: Pina Colada
        category: Cocktail
        description: Pineapple, rum, creamy
cocktail
        price: '2399.00'
        created_at: '2022-03-
20T19:10:10.000000Z'
        updated_at: '2022-03-
31T21:44:06.000000Z'
      - id: 2
        name: Beef steak
        category: Food
        description: Argentinian beef
        price: '9900.00'
        created_at: null
        updated_at: '2022-03-
27T15:02:11.000000Z'
      - id: 3
        name: Chicken steak
        category: Food
        description: Countryside chicken
        price: '3000.00'
        created_at: null
        updated_at: null
      - id: 4
        name: Beef burger
        category: Food
        description: 330g meat, salad, mayo,
burger
        price: '3999.99'
        created_at: null
        updated_at: '2022-03-
27T14:46:37.000000Z'
      - id: 5
        name: Long Island
```

```
category: Cocktail
description: Rum, Tequila, Vodka, Coke
price: '3999.00'
created_at: '2022-03-
21T02:03:35.000000Z'
updated_at: '2022-03-
23T08:02:03.000000Z'
- id: 8
  name: Mliano spaghetti
  category: Food
  description: Meatballs with pasta
  price: '2699.00'
  created_at: '2022-03-
23T07:59:36.000000Z'
  updated_at: '2022-03-
23T07:59:36.000000Z'
- id: 9
  name: Chilli con carne
  category: Food
  description: Mexican spicy minced meat
  price: '2699.00'
  created_at: '2022-03-
23T08:00:06.000000Z'
  updated_at: '2022-03-
23T08:00:06.000000Z'
- id: 10
  name: Tequila Sunrize
  category: Cocktail
  description: Lemony tequila cocktail
  price: '2999.00'
  created_at: '2022-03-
23T08:25:54.000000Z'
  updated_at: '2022-03-
23T08:25:54.000000Z'
- id: 11
  name: Sparkling water
  category: Beverage
  description: Still water sparkling
  price: '499.00'
  created_at: '2022-03-
23T08:34:32.000000Z'
```

```
        updated_at: '2022-03-
23T08:34:32.000000Z'
      - id: 13
        name: Coca-Cola
        category: Beverage
        description: Cola 0.5l
        price: '499.00'
        created_at: '2022-03-
23T08:35:55.000000Z'
        updated_at: '2022-03-
23T08:35:55.000000Z'
      - id: 14
        name: Pepsi Coke
        category: Beverage
        description: Coke 0.5l
        price: '499.00'
        created_at: '2022-03-
23T08:36:09.000000Z'
        updated_at: '2022-03-
23T08:36:09.000000Z'
      - id: 15
        name: Fanta Lemon
        category: Beverage
        description: Lemon drink 0.5l
        price: '499.00'
        created_at: '2022-03-
23T08:36:36.000000Z'
        updated_at: '2022-03-
23T08:36:36.000000Z'
      - id: 16
        name: Coca-Cola can
        category: Beverage
        description: Coke drink 0.33l
        price: '499.00'
        created_at: '2022-03-
23T08:36:56.000000Z'
        updated_at: '2022-03-
23T08:36:56.000000Z'
      - id: 18
        name: Lemonade
        category: Beverage
```

```
description: Lemon juice 0.33 glass
price: '499.00'
created_at: '2022-03-
23T08:37:46.000000Z'
updated_at: '2022-03-
23T08:37:46.000000Z'
- id: 21
  name: Töpörtyű
  category: Food
  description: Hungarian pork specialty
  price: '1999.00'
  created_at: '2022-03-
27T14:52:15.000000Z'
  updated_at: '2022-03-
27T14:58:50.000000Z'
- id: 22
  name: Chicken wings
  category: Food
  description: Spicy grilled wings
  price: '2350.00'
  created_at: '2022-03-
27T14:53:50.000000Z'
  updated_at: '2022-03-
27T14:53:50.000000Z'
- id: 23
  name: Ham and eggs
  category: Food
  description: Grilled ham and eggs
  price: '2290.00'
  created_at: '2022-03-
27T14:56:18.000000Z'
  updated_at: '2022-03-
27T15:00:16.000000Z'
- id: 24
  name: Vodka Soda
  category: Cocktail
  description: Kalinka Vodka and Soda 2dl
  price: '2999.00'
  created_at: '2022-03-
27T15:01:10.000000Z'
```

```
        updated_at: '2022-03-
27T15:01:22.000000Z'
      - id: 25
        name: Goulash soup
        category: Food
        description: Pork chops, potato,
vegetable, thick soup
        price: '2999.00'
        created_at: '2022-03-
27T15:13:49.000000Z'
        updated_at: '2022-03-
27T15:13:49.000000Z'
      - id: 26
        name: Ginger tea
        category: Drink
        description: Hot natural ginger tea
        price: '799.00'
        created_at: '2022-03-
27T15:15:24.000000Z'
        updated_at: '2022-03-
27T15:15:24.000000Z'
      - id: 27
        name: Black tea
        category: Drink
        description: Earl gray black tea
        price: '599.00'
        created_at: '2022-03-
27T15:16:43.000000Z'
        updated_at: '2022-03-
27T15:16:56.000000Z'
      - id: 29
        name: Grilled chicken
        category: Food
        description: Spicy chicken grilled
        price: '2850.00'
        created_at: '2022-04-
08T19:07:48.000000Z'
        updated_at: '2022-04-
08T19:07:48.000000Z'
      - id: 30
        name: Honey
```



```
        category: Food
        description: small box of honey
        price: '699.00'
        created_at: '2022-04-
08T19:29:14.000000Z'
        updated_at: '2022-04-
08T19:30:29.000000Z'
    /api/logout:
      post:
        tags:
          - default
        summary: http://localhost:8000/api/logout
        requestBody:
          content:
            application/x-www-form-urlencoded:
              schema:
                properties:
                  name:
                    type: string
                    example: Andras2
                  email:
                    type: string
                    example: andras2@gmail.com
                  password:
                    type: string
                    example: xyz23
                  password_confirmation:
                    type: string
                    example: xyz23
        security:
          - bearerAuth: []
        parameters:
          - name: Accept
            in: header
            schema:
              type: string
              example: application/json
        responses:
          '200':
            description: OK
            headers:
```

```
Host:
  schema:
    type: string
    example: localhost:8000
Date:
  schema:
    type: string
    example: Fri, 08 Apr 2022 20:02:07 GMT
Connection:
  schema:
    type: string
    example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '59'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
```

```
      message: You have succesfully logged out
'401':
  description: Unauthorized
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Fri, 08 Apr 2022 20:02:23 GMT
    Connection:
      schema:
        type: string
        example: close
    X-Powered-By:
      schema:
        type: number
        example: PHP/8.0.12
    Cache-Control:
      schema:
        type: string
        example: no-cache, private
    Content-Type:
      schema:
        type: string
        example: application/json
    Access-Control-Allow-Origin:
      schema:
        type: string
        example: '*'
  content:
    application/json:
      schema:
        type: object
        example:
          message: Unauthenticated.
/api/login:
  post:
    tags:
```

```
- default
summary: http://localhost:8000/api/login
requestBody:
  content:
    application/x-www-form-urlencoded:
      schema:
        properties:
          name:
            type: string
            example: Admin
          email:
            type: string
            example: admin@gmail.com
          password:
            type: string
            example: Admin111$
          password_confirmation:
            type: string
            example: Admin111$
security:
  - bearerAuth: []
parameters:
  - name: Accept
    in: header
    schema:
      type: string
      example: application/json
responses:
  '201':
    description: Created
    headers:
      Host:
        schema:
          type: string
          example: localhost:8000
      Date:
        schema:
          type: string
          example: Fri, 08 Apr 2022 19:47:16 GMT
    Connection:
      schema:
```

```
      type: string
      example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '59'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      user:
        id: 1
        name: admin
        email: admin@gmail.com
        role: Admin
        email_verified_at: '2022-03-
20T19:07:07.000000Z'
        current_team_id: null
        profile_photo_path: null
```

```
      created_at: '2022-03-
20T19:06:58.000000Z'
      updated_at: '2022-03-
20T19:07:07.000000Z'
      address: Rakoczi utca 85
      phone_number: '+36201111111'
      profile_photo_url: >-
        https://ui-
avatars.com/api/?name=a&color=7F9CF5&background=EBF4FF
      token:
15|3XKkWLZA90yQ5sPjViOE6fUYKKNb6P4NnxPYMnCP
'401':
  description: Unauthorized
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Fri, 08 Apr 2022 19:48:13 GMT
    Connection:
      schema:
        type: string
        example: close
    X-Powered-By:
      schema:
        type: number
        example: PHP/8.0.12
    Cache-Control:
      schema:
        type: string
        example: no-cache, private
    Content-Type:
      schema:
        type: string
        example: application/json
    X-RateLimit-Limit:
      schema:
        type: integer
```

```
        example: '60'
      X-RateLimit-Remaining:
        schema:
          type: integer
          example: '60'
      Access-Control-Allow-Origin:
        schema:
          type: string
          example: '*'
    content:
      application/json:
        schema:
          type: object
          example:
            message: Wrong credentials entered
/api/products/4:
  get:
    tags:
      - default
    summary: http://localhost:8000/api/products/25
    security:
      - bearerAuth: []
    parameters:
      - name: Accept
        in: header
        schema:
          type: string
          example: application/json
    responses:
      '200':
        description: OK
        headers:
          Host:
            schema:
              type: string
              example: localhost:8000
          Date:
            schema:
              type: string
              example: Fri, 08 Apr 2022 20:02:54 GMT
        Connection:
```

```
    schema:
      type: string
      example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '58'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    examples:
      example-0:
        summary:
http://localhost:8000/api/products/25
        value:
          id: 25
          name: Goulash soup
          category: Food
          description: Pork chops, potato,
vegetable, thick soup
```



```
        price: '2999.00'
        created_at: '2022-03-
27T15:13:49.000000Z'
        updated_at: '2022-03-
27T15:13:49.000000Z'
        example-1:
          summary:
http://localhost:8000/api/products/4
          value:
            id: 4
            name: Beef burger
            category: Food
            description: 330g meat, salad, mayo,
burger
            price: '3999.99'
            created_at: null
            updated_at: '2022-03-
27T14:46:37.000000Z'
    /api/products/30:
      put:
        tags:
          - default
        summary: http://localhost:8000/api/products/30
        requestBody:
          content:
            application/x-www-form-urlencoded:
              schema:
                properties:
                  name:
                    type: string
                    example: Honey
                  category:
                    type: string
                    example: Food
                  description:
                    type: string
                    example: small box of honey
                  price:
                    type: integer
                    example: '499'
        security:
```

```
- bearerAuth: []
parameters:
- name: Accept
  in: header
  schema:
    type: string
    example: application/json
responses:
  '200':
    description: OK
    headers:
      Host:
        schema:
          type: string
          example: localhost:8000
      Date:
        schema:
          type: string
          example: Fri, 08 Apr 2022 20:06:41 GMT
      Connection:
        schema:
          type: string
          example: close
      X-Powered-By:
        schema:
          type: number
          example: PHP/8.0.12
      Cache-Control:
        schema:
          type: string
          example: no-cache, private
      Content-Type:
        schema:
          type: string
          example: application/json
      X-RateLimit-Limit:
        schema:
          type: integer
          example: '60'
      X-RateLimit-Remaining:
        schema:
```

```
      type: integer
      example: '59'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      id: 30
      name: Honey
      category: Food
      description: small box of honey
      price: '899'
      created_at: '2022-04-08T19:29:14.000000Z'
      updated_at: '2022-04-08T20:06:41.000000Z'
'401':
  description: Unauthorized
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Fri, 08 Apr 2022 20:06:13 GMT
    Connection:
      schema:
        type: string
        example: close
    X-Powered-By:
      schema:
        type: number
        example: PHP/8.0.12
    Cache-Control:
      schema:
        type: string
        example: no-cache, private
```

```
Content-Type:
  schema:
    type: string
    example: application/json
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      message: Unauthenticated.
'403':
  description: Forbidden
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Sat, 09 Apr 2022 18:39:54 GMT
    Connection:
      schema:
        type: string
        example: close
    X-Powered-By:
      schema:
        type: number
        example: PHP/8.0.12
    Cache-Control:
      schema:
        type: string
        example: no-cache, private
  Content-Type:
    schema:
      type: string
      example: application/json
```

```
    X-RateLimit-Limit:
      schema:
        type: integer
        example: '60'
    X-RateLimit-Remaining:
      schema:
        type: integer
        example: '58'
    Access-Control-Allow-Origin:
      schema:
        type: string
        example: '*'
  content:
    application/json:
      schema:
        type: object
      example:
        message: Forbidden
delete:
  tags:
    - default
  summary: http://localhost:8000/api/products/4
  security:
    - bearerAuth: []
  parameters:
    - name: Accept
      in: header
      schema:
        type: string
      example: application/json
  responses:
    '200':
      description: OK
      headers:
        Host:
          schema:
            type: string
          example: localhost:8000
        Date:
          schema:
            type: string
```

```
    example: Fri, 08 Apr 2022 19:49:14 GMT
Connection:
  schema:
    type: string
    example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Content-Type:
  schema:
    type: string
    example: text/html; charset=UTF-8
Cache-Control:
  schema:
    type: string
    example: no-cache, private
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '59'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  text/plain:
    schema:
      type: string
      example: '1'
'401':
  description: Unauthorized
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
```

```
Date:
  schema:
    type: string
    example: Fri, 08 Apr 2022 19:50:23 GMT
Connection:
  schema:
    type: string
    example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
      example:
        message: Unauthenticated.
'403':
  description: Forbidden
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Sat, 09 Apr 2022 18:39:31 GMT
```

```
Connection:
  schema:
    type: string
    example: close
X-Powered-By:
  schema:
    type: number
    example: PHP/8.0.12
Cache-Control:
  schema:
    type: string
    example: no-cache, private
Content-Type:
  schema:
    type: string
    example: application/json
X-RateLimit-Limit:
  schema:
    type: integer
    example: '60'
X-RateLimit-Remaining:
  schema:
    type: integer
    example: '59'
Access-Control-Allow-Origin:
  schema:
    type: string
    example: '*'
content:
  application/json:
    schema:
      type: object
    example:
      message: Forbidden
/api/products/search/long:
  get:
    tags:
      - default
    summary:
http://localhost:8000/api/products/search/chicken
security:
```



```
- noauthAuth: []
parameters:
- name: Accept
  in: header
  schema:
    type: string
    example: application/json
responses:
  '200':
    description: OK
    headers:
      Host:
        schema:
          type: string
          example: localhost:8000
      Date:
        schema:
          type: string
          example: Fri, 08 Apr 2022 19:32:46 GMT
      Connection:
        schema:
          type: string
          example: close
      X-Powered-By:
        schema:
          type: number
          example: PHP/8.0.12
      Cache-Control:
        schema:
          type: string
          example: no-cache, private
      Content-Type:
        schema:
          type: string
          example: application/json
      X-RateLimit-Limit:
        schema:
          type: integer
          example: '60'
      X-RateLimit-Remaining:
        schema:
```

```
        type: integer
        example: '57'
    Access-Control-Allow-Origin:
        schema:
            type: string
            example: '*'
    content:
        application/json:
            schema:
                type: object
            examples:
                example-0:
                    summary:
http://localhost:8000/api/products/search/chicken
                    value:
                        - id: 3
                          name: Chicken steak
                          category: Food
                          description: Countryside chicken
                          price: '3000.00'
                          created_at: null
                          updated_at: null
                        - id: 22
                          name: Chicken wings
                          category: Food
                          description: Spicy grilled wings
                          price: '2350.00'
                          created_at: '2022-03-
27T14:53:50.000000Z'
                          updated_at: '2022-03-
27T14:53:50.000000Z'
                        - id: 29
                          name: Grilled chicken
                          category: Food
                          description: Spicy chicken grilled
                          price: '2850.00'
                          created_at: '2022-04-
08T19:07:48.000000Z'
                          updated_at: '2022-04-
08T19:07:48.000000Z'
                    example-1:
```

```
summary:
http://localhost:8000/api/products/search/chicken
value:
  - id: 5
    name: Long Island
    category: Cocktail
    description: Rum, Tequila, Vodka,
Coke
    price: '3999.00'
    created_at: '2022-03-
21T02:03:35.000000Z'
    updated_at: '2022-03-
23T08:02:03.000000Z'
/api/register:
  post:
    tags:
      - default
    summary: http://localhost:8000/api/register
    requestBody:
      content:
        application/x-www-form-urlencoded:
          schema:
            properties:
              name:
                type: string
                example: tester
              email:
                type: string
                example: user@gmail.com
              password:
                type: string
                example: Testing111
              password_confirmation:
                type: string
                example: Testing111
      parameters:
        - name: Accept
          in: header
          schema:
            type: string
            example: application/json
```

```
responses:
  '201':
    description: Created
    headers:
      Host:
        schema:
          type: string
          example: localhost:8000
      Date:
        schema:
          type: string
          example: Fri, 08 Apr 2022 20:01:06 GMT
      Connection:
        schema:
          type: string
          example: close
      X-Powered-By:
        schema:
          type: number
          example: PHP/8.0.12
      Cache-Control:
        schema:
          type: string
          example: no-cache, private
      Content-Type:
        schema:
          type: string
          example: application/json
      X-RateLimit-Limit:
        schema:
          type: integer
          example: '60'
      X-RateLimit-Remaining:
        schema:
          type: integer
          example: '59'
      Access-Control-Allow-Origin:
        schema:
          type: string
          example: '*'
    content:
```

```
application/json:
  schema:
    type: object
  example:
    user:
      name: tester
      email: tester1234@gmail.com
      updated_at: '2022-04-
08T20:01:04.000000Z'
      created_at: '2022-04-
08T20:01:04.000000Z'
      id: 7
      profile_photo_url: >-
        https://ui-
avatars.com/api/?name=t&color=7F9CF5&background=EBF4FF
    token:
16|l3cQirhIWpgDpe7vi6ngDMAxnIsjXMOBGWmTtWCt
'422':
  description: Unprocessable Content
  headers:
    Host:
      schema:
        type: string
        example: localhost:8000
    Date:
      schema:
        type: string
        example: Fri, 08 Apr 2022 19:50:43 GMT
    Connection:
      schema:
        type: string
        example: close
    X-Powered-By:
      schema:
        type: number
        example: PHP/8.0.12
    Cache-Control:
      schema:
        type: string
        example: no-cache, private
    Content-Type:
```

```
    schema:
      type: string
      example: application/json
  X-RateLimit-Limit:
    schema:
      type: integer
      example: '60'
  X-RateLimit-Remaining:
    schema:
      type: integer
      example: '59'
  Access-Control-Allow-Origin:
    schema:
      type: string
      example: '*'
  content:
    application/json:
      schema:
        type: object
      examples:
        example-0:
          summary:
            http://localhost:8000/api/register
          value:
            message: The given data was invalid.
            errors:
              email:
                - The email has already been
taken.
              password:
                - The password confirmation does
not match.
        example-1:
          summary:
            http://localhost:8000/api/register
          value:
            message: The given data was invalid.
            errors:
              email:
                - The email has already been
taken.
```

