

DOKUMENTÁCIÓ

Vizsgaremek

SZOFTVERFEJLESZTŐ ÉS -TESZTELŐ TECHNIKUS SZAKMA

Tartalomjegyzék

1. Bevezetés, a téma ismertetése, témaválasztás indoklása
2. Felhasználói dokumentáció.
 - 2.1 Telepítési útmutatás Windows eszközre
 - 2.2 A Web App telepítése és konfigurálása
3. Fejlesztői dokumentáció
 - 3.1 A fejlesztési környezet, használt technológiák, keretrendszerök bemutatása
 - 3.2 A webes projektben felhasznált technológiák és verziók
 - 3.3 Az asztali alkalmazásban használt technológiák és funkciók
 - 3.4 Adatbázis, adattáblák felépítése és diagram
 - 3.5 Útvonalak listája
 - 3.6 Modellek bemutatása ..
 - 3.7 Controller-ek és AdminMiddleware bemutatása
 - 3.8 Livewire Component-ek bemutatása
 - 3.9 Front-end, UI/ UX bemutatása
 - 3.10 Asztali alkalmazás bemutatása
4. Tesztelés
5. Összefoglalás
6. Kiegészítés

1. Bevezetés, a téma ismertetése, téma választás 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 mindenkorban 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 foglalásaiat kezelni.

Ez nem azt jelent, hogy nem hirdetnek a legfőbb platformokon és nem érkezik ott foglalásaiak 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 foglalásaiat kezelése, majd mindenek mellett szerettük volna, ha a szálloda termékeinek kezelés is megvalósul a web applikáción.

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

Az adminisztrátor tud termékeket felvinni, módosítani, törleni 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 elértekkelenedésének 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 kényetlenek növelni az itthon eladott termékek árait.

Mindezek fényében fontos volt számunkra, hogy a hotel tulajdonos/ vezetőség / adminisztrátorak 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 kívánunk 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 foglalási rendszerinek is a kiszolgálása és az, hogy a hotelben napokat kezelünk szobánkként, 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 hosszan 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 gépén, telepítse a PHP 8.0.12 vagy 8.0.2 verziót, de mindenkorban 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 webszerver 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 folyamataazonosító (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 kíkkel vagy Git bash here vagy open with VS code-al meg kell nyitni egy mappát és oda klónozni a git clone parancsal 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 kíkk és a GIT bash here parancsal 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 parancssal 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**)

Fájl	Név	Módosítás dátuma	Típus	Méret
if	.git	2022. 04. 13. 11:22	Fájlmappa	
F	api_documentation	2022. 04. 09. 20:56	Fájlmappa	
P	app	2022. 03. 19. 0:26	Fájlmappa	
H	bootstrap	2022. 03. 19. 0:26	Fájlmappa	
E	config	2022. 03. 23. 22:16	Fájlmappa	
E	database	2022. 03. 19. 0:26	Fájlmappa	
B	node_modules	2022. 04. 11. 20:09	Fájlmappa	
S	public	2022. 03. 19. 0:57	Fájlmappa	
C	resources	2022. 03. 23. 21:41	Fájlmappa	
h	routes	2022. 03. 19. 20:53	Fájlmappa	
A	storage	2022. 03. 19. 0:26	Fájlmappa	
G	tests	2022. 03. 19. 0:26	Fájlmappa	
i	vendor	2022. 04. 15. 22:26	Fájlmappa	
x	.editorconfig	2022. 03. 19. 0:26	EDITORCONFIG fájl	1 KB
.	.env	2022. 04. 08. 20:31	ENV fájl	1 KB
Or	.env.example	2022. 03. 19. 0:26	EXAMPLE fájl	1 KB
A	.gitattributes	2022. 03. 19. 0:26	Szöveges dokume...	1 KB
V	.gitignore	2022. 03. 19. 0:26	Szöveges dokume...	1 KB
Ez	.styleci.yml	2022. 03. 19. 0:26	YML fájl	1 KB
3	Agent	2022. 03. 19. 0:26	Fájl	0 KB
A	artisan	2022. 03. 19. 0:26	Fájl	2 KB
C	composer.json	2022. 04. 15. 22:26	Adobe After Effect...	2 KB
K	composer.lock	2022. 04. 15. 22:26	LOCK fájl	326 KB
L	laravel.sql	2022. 04. 13. 9:52	SQL fájl	28 KB
V	package.json	2022. 03. 23. 20:41	Adobe After Effect...	1 KB
Z	package-lock.json	2022. 04. 11. 20:09	Adobe After Effect...	344 KB
H	phpunit.xml	2022. 03. 19. 0:26	XML dokumentum	2 KB
H	server.php	2022. 03. 19. 0:26	PHPfile	1 KB
H	tailwind.config.js	2022. 03. 23. 14:08	JavaScript-fájl	1 KB
Ú	~WRL0005.tmp	2022. 04. 15. 22:28	TMP fájl	137 KB
Ú	webpack.mix.js	2022. 03. 31. 23:00	JavaScript-fájl	1 KB

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** parancsal új terminált létesítünk
8. **composer install**
9. **npm install**
10. **npm run dev**
11. **Env file konfigurálása a lejebb 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 útvonaon a VS doce-banl) és a

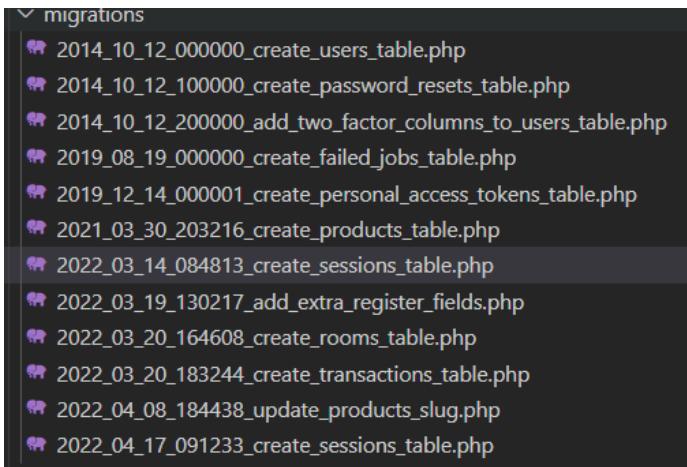
<http://localhost/phpmyadmin> oldalon, például ebben az esetben

```
e1hasznalo@DESKTOP-KRPPV10 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` va
char(255) not null, `user_id` bigint unsigned null, `ip_address` varchar(45) null, `user_agent` text null, `payload` text not nu
l, `last_activity` int not null) default character set utf8mb4 collate 'utf8mb4_unicode_ci')

at C:\Users\Elhasznalo\Documents\auth\laravel\TEST\TestHotel\vendor\laravel\framework\src\Illuminate\Database\Connection.php:711
```

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



The screenshot shows a dark-themed file explorer window with a tree view. The root node is 'migrations'. Under it, there are 17 sub-nodes, each represented by a small elephant icon and a file name. The file names are timestamped and follow a specific naming convention for Laravel migrations.

- ▼ migrations
 - 🐘 2014_10_12_000000_create_users_table.php
 - 🐘 2014_10_12_100000_create_password_resets_table.php
 - 🐘 2014_10_12_200000_add_two_factor_columns_to_users_table.php
 - 🐘 2019_08_19_000000_create_failed_jobs_table.php
 - 🐘 2019_12_14_000001_create_personal_access_tokens_table.php
 - 🐘 2021_03_30_203216_create_products_table.php
 - 🐘 2022_03_14_084813_create_sessions_table.php
 - 🐘 2022_03_19_130217_add_extra_register_fields.php
 - 🐘 2022_03_20_164608_create_rooms_table.php
 - 🐘 2022_03_20_183244_create_transactions_table.php
 - 🐘 2022_04_08_184438_update_products_slug.php
 - 🐘 2022_04_17_091233_create_sessions_table.php

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:

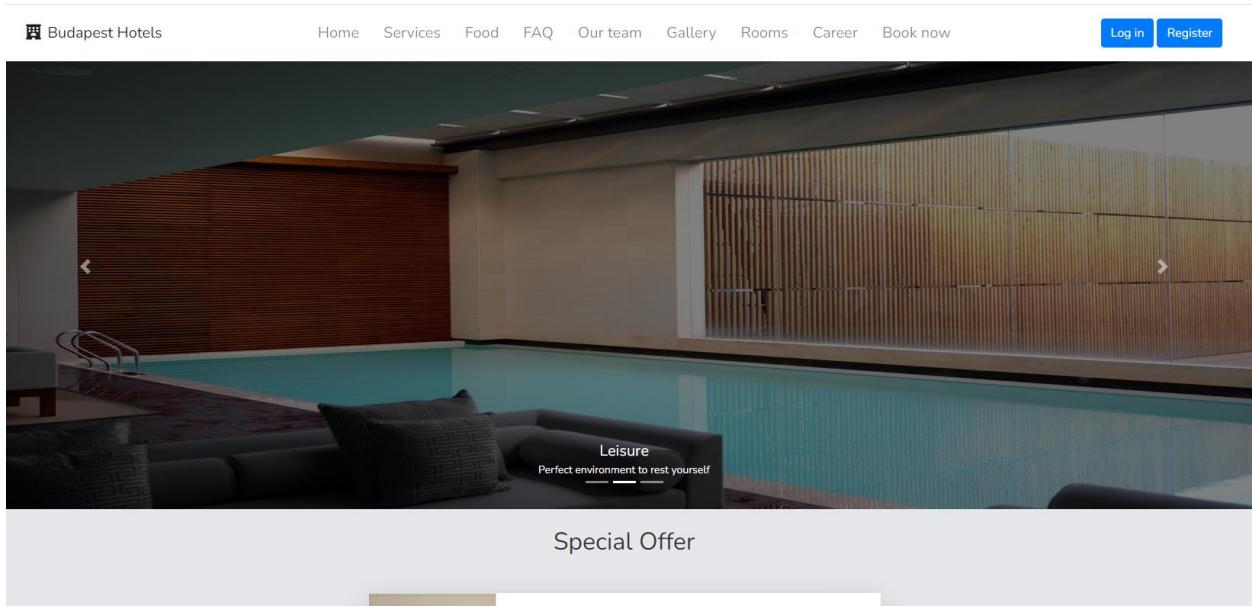
- a. <http://localhost/phpmyadmin/> oldalra látogatni:
- b. Drop database laravel az adatbázis eldobása
- c. Újra csinálni egy laravel nevű adatbázist
- d. A Laravel.sql fájlt a <http://localhost/phpmyadmin/> laravel adatbázisban az Import gombra kattintra, majd a 'Choose file' gombra kattintva be tudjuk 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

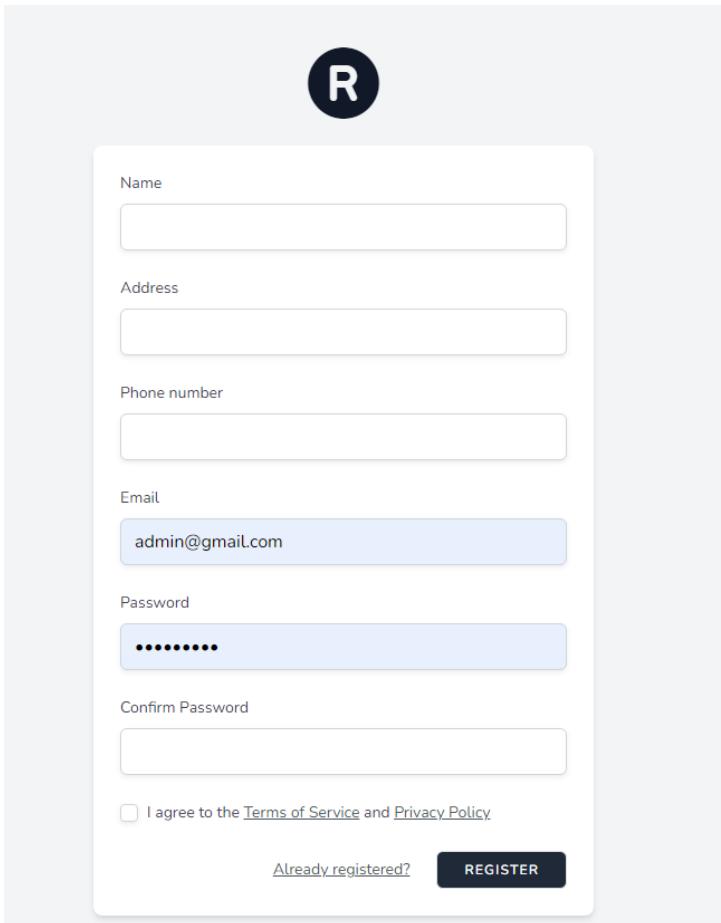
Kezdőlap:



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

The screenshot shows a login form. At the top right of the form is a large, stylized black arrow pointing to the right. The form itself has a light gray background. It contains fields for 'Email' (with the value 'user2@gmail.com') and 'Password' (with several dots indicating the password). Below these fields is a 'Remember me' checkbox followed by a checked checkbox. At the bottom of the form are three links: 'No account yet?', 'Forgot your password?', and a dark blue 'LOG IN' button. The entire form is contained within a white rounded rectangle.

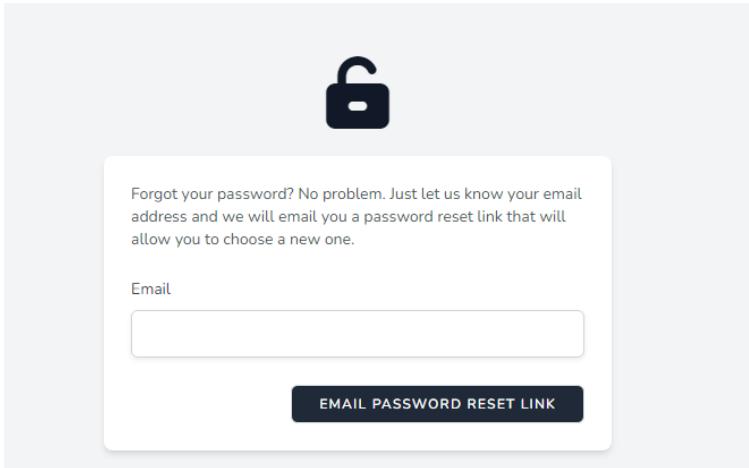
Illetve a /register oldalra érkezik:



The screenshot shows a registration form with the following fields and layout:

- Name:** Input field (empty)
- Address:** Input field (empty)
- Phone number:** Input field (empty)
- Email:** Input field containing "admin@gmail.com" (highlighted with a light blue background)
- Password:** Input field containing "*****" (highlighted with a light blue background)
- Confirm Password:** Input field (empty)
- Agreement:** A checkbox followed by the text "I agree to the [Terms of Service](#) and [Privacy Policy](#)".
- Buttons:** Two buttons at the bottom: "Already registered?" (link) and "REGISTER" (dark button).

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, helyre állítás 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 állomásoszó 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 kiküszöbölnének annak a kockázata, hogy a teszt- és mailek valódi ügyfelek e-mail címére menjenek.

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

Ahhoz, hogy regisztrálni tudunk 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

jelszó: **Admin111\$**

Vendég fiók:

felhasználónév/ email: user2@gmail.com

jelszó: **User222\$**

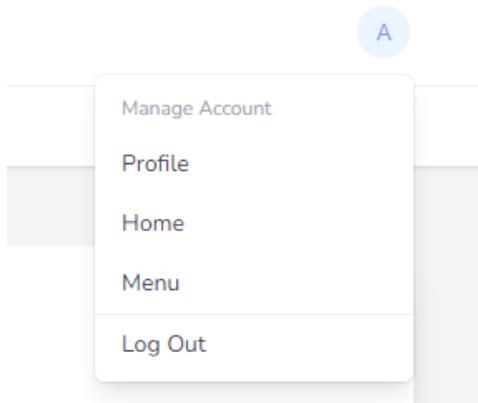
Minden esetben új regisztráció esetén figyelni kell, hogy a jelszó megadásánál a jelszónak minimum 8 karakterből kell állnia, legyen benne betű, szám és speciális karakter és egy nagy betű is.

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 kérdésekéről(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 emailes 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 jelszavat változtatni.

Lejjebb Két kulcsos azonosítást tudnak a felhasználók bekapcsolni Google Authenticator Applikáció segítségével egy egyedi QR kódót 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 felhasználok GDPR-nak megfelelően tudják a saját fiókjukat törölni is.

The screenshot displays a user profile management interface with the following sections:

- Profile Information:** Allows updating account details like Name (admin), Email (admin@gmail.com), Role (Admin), Address (Sunset street 123, Miami, Florida), and Phone number (+36201111111). A "SAVE" button is present.
- Update Password:** A section prompting the user to update their password if they are using a long, random password for 60 days. It includes fields for Current Password, New Password, and Confirm Password, along with a "SAVE" button.
- Two Factor Authentication:** A section stating that two-factor authentication is not enabled. It explains that when enabled, a security token will be required for login. A "ENABLE" button is available.
- Browser Sessions:** A section for managing active browser sessions. It shows a list for "Windows - Chrome" (127.0.0.1, This device) and a "LOG OUT OTHER BROWSER SESSIONS" button.
- Delete Account:** A section for permanently deleting the account. It states that all data will be permanently deleted. A "DELETE ACCOUNT" button is present.

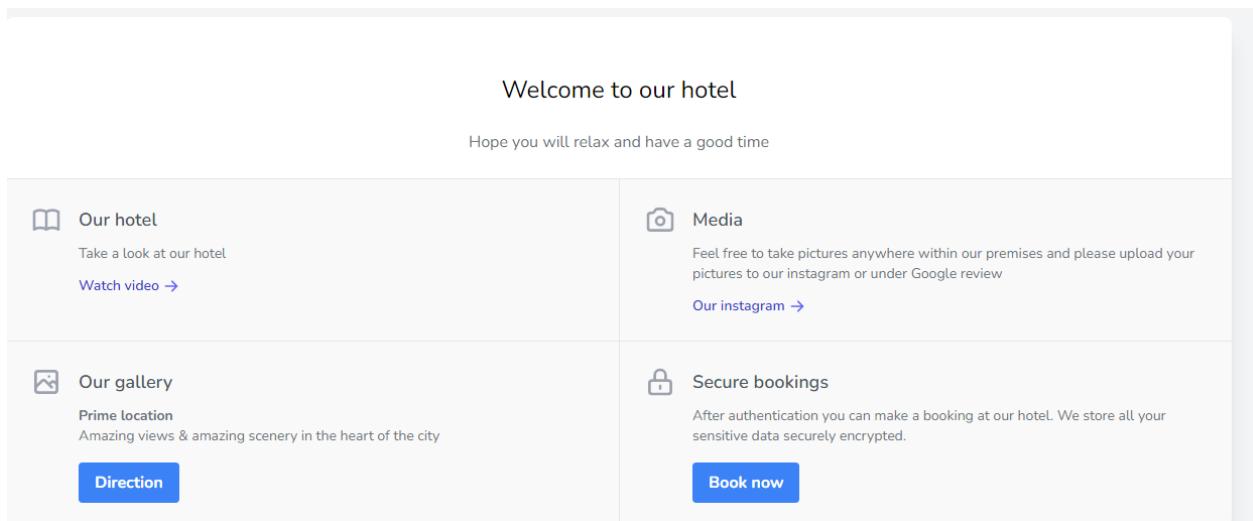
A home page a bejelentkezett felhasználóknak minden /user/home URI a kezdőlap, ám ez másik nézetet ad vissza annak alapján, hogy a felhasználó jogosultsága admin vagy user.

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 admin 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.



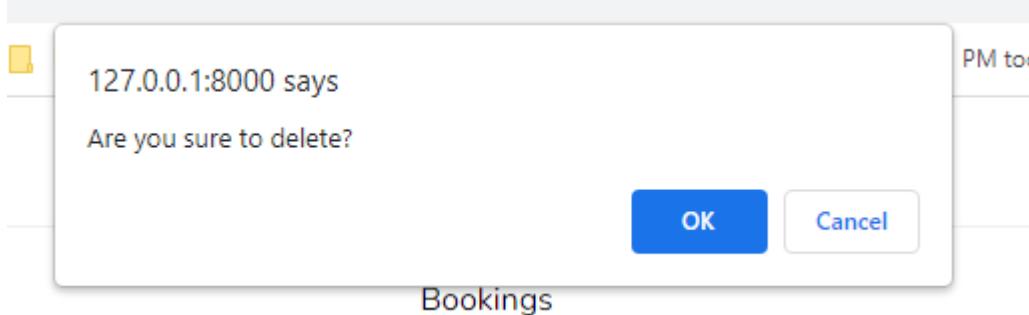
A dashboard nézetben van egy útvonal (Google Maps redirect link) választó gomb (Direction) és egy Book Now gomb, ahol a felhasználó a /my-bookings útvonalra jut el ahol csak magának tud foglalni szobát a belépett felhasználó a New gombra kattintva. A media konténer Our instagram az Instagramra visz át,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

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. Delete 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 különbsége, míg a Bill ez a különbség megszorozva a napok számával:

Guest name:
Toni Maki

Guest email:
joseph@gmail.com

Guest phone:
06206652342

Select Room

Checkin:

Checkout:

Days:

Bill:

Halfboard:

Number of guests:

Természetesen minden /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.

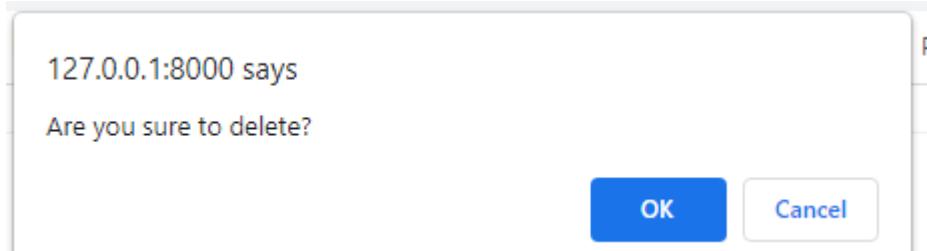
Itt amennyiben még nincs ilyen nevű felhasználó, mint akit az admin felvisz akkor a foglalás rögzítésével együtt a user (azaz vendég) neve, email címe és egy random generált bcrypt titkosítással ellátott jelszó is rögzítésre kerül a user táblában a transaction táblában pedig tároljuk az admin által megadott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

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 törlni 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 törlni 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.

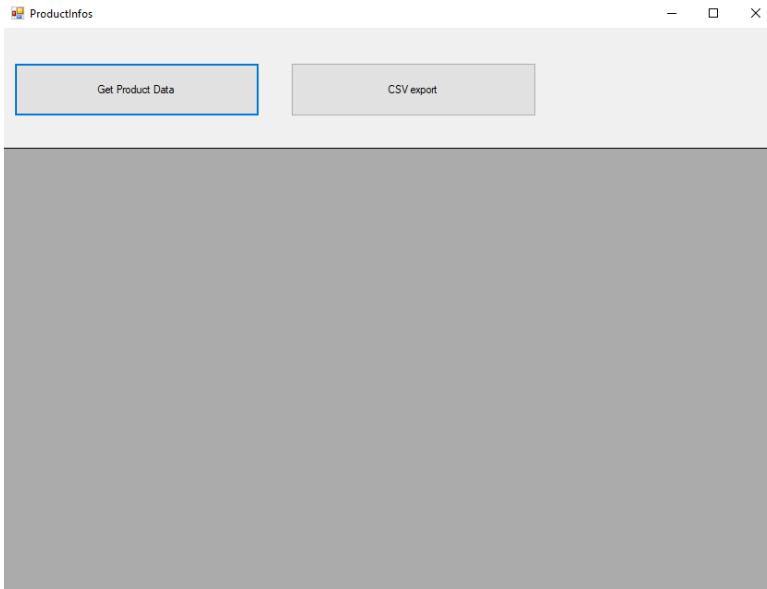
Minden terméket, ami az admin oldalon megjelenik az elérhető a vendégek számára is egy hasonló tábla nézetben csak teljes CRUD operátorok nélkül, azaz szerkeszteni, törlni, avagy új terméket itt felvinni nem lehet, csak Read opción adott, azaz a GET metódus, hogy tudják a vendégek előben látni a termékeket és keresni közöttük, mindez 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 adatait szed le a webről egy Http kliens segítségével majd az asztali alkalmazás is táblába rendezi ezeket a Get Product data-ra kattintva:



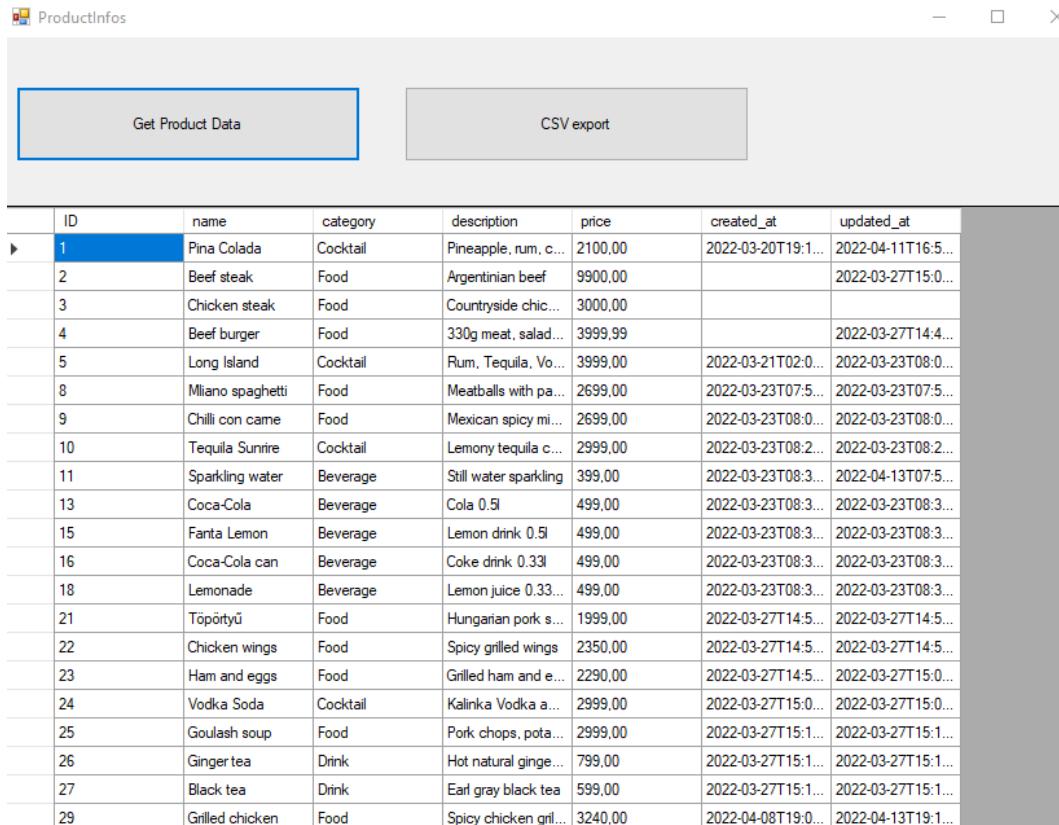
Az asztali alkalmazást a Products.exe fájlra kétszer kattintva tudjuk telepíteni és indítani a NET/Desktop-master mappából.

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.

Név	Módosítás dátuma	Típus	Méret
.git	2022. 04. 17. 19:12	Fájlmappa	
.vs	2022. 04. 13. 16:13	Fájlmappa	
Application Files	2022. 04. 17. 14:47	Fájlmappa	
GetProducts	2022. 04. 17. 14:33	Fájlmappa	
packages	2022. 04. 13. 17:08	Fájlmappa	
GetProducts.application	2022. 04. 17. 14:47	Application Manif...	6 KB
GetProducts.sln	2022. 04. 10. 13:46	Visual Studio Solu...	2 KB
setup.exe	2022. 04. 17. 14:47	Alkalmazás	541 KB

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 carne	Food	Mexican spicy mi...	2699.00	2022-03-23T08:0...	2022-03-23T08:0...
10	Tequila Sunrise	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 0.33l	499.00	2022-03-23T08:3...	2022-03-23T08:3...
21	Töpörtyű	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	Ginger 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 grill...	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.

A	B	C	D	E	F	G	H	I	J	K
ID	"name"	"category"	"description"	"price"	"created_at"	"updated_at"				
1	"Pina Colada"	"Cocktail"	"Pineapple, rum, creamy cocktail"	"2100,00"	"2022-03-20T19:10:10.000000Z"	"2022-04-11T16:50:16.000000Z"				
2	"Beef steak"	"Food"	"Argentinian beef"	"9900,00"	"",	"2022-03-27T15:02:11.000000Z"				
3	"Chicken steak"	"Food"	"Countryside chicken"	"3000,00"	"",					
4	"Beef burger"	"Food"	"330g meat, salad, mayo, burger"	"3999,99"	"",	"2022-03-27T14:46:37.000000Z"				
5	"Long Island"	"Cocktail"	"Rum, Tequila, Vodka, Coke"	"3999,00"	"2022-03-21T02:03:35.000000Z"	"2022-03-23T08:02:03.000000Z"				
8	"Mliano spaghetti"	"Food"	"Meatballs with pasta"	"2699,00"	"2022-03-23T07:59:36.000000Z"	"2022-03-23T07:59:36.000000Z"				
9	"Chilli con carne"	"Food"	"Mexican spicy minced meat"	"2699,00"	"2022-03-23T08:00:06.000000Z"	"2022-03-23T08:00:06.000000Z"				
10	"Tequila Sunrise"	"Cocktail"	"Lemon tequila cocktail"	"2999,00"	"2022-03-23T08:25:54.000000Z"	"2022-03-23T08:25:54.000000Z"				
11	"Sparkling water"	"Beverage"	"Still water sparkling"	"399,00"	"2022-03-23T08:34:32.000000Z"	"2022-04-13T07:58:55.000000Z"				
13	"Coca-Cola"	"Beverage"	"Cola 0.5l"	"499,00"	"2022-03-23T08:35:55.000000Z"	"2022-03-23T08:35:55.000000Z"				
15	"Fanta Lemon"	"Beverage"	"Lemon drink 0.5l"	"499,00"	"2022-03-23T08:36:36.000000Z"	"2022-03-23T08:36:36.000000Z"				
16	"Coca-Cola can"	"Beverage"	"Coke drink 0.33l"	"499,00"	"2022-03-23T08:36:56.000000Z"	"2022-03-23T08:36:56.000000Z"				
18	"Lemonade"	"Beverage"	"Lemon juice 0.33 glass"	"499,00"	"2022-03-23T08:37:46.000000Z"	"2022-03-23T08:37:46.000000Z"				
21	"TÁPÁRTYLÓ"	"Food"	"Hungarian pork specialty"	"1999,00"	"2022-03-27T14:52:15.000000Z"	"2022-03-27T14:58:50.000000Z"				
22	"Chicken wings"	"Food"	"Spicy grilled wings"	"2350,00"	"2022-03-27T14:53:50.000000Z"	"2022-03-27T14:53:50.000000Z"				
23	"Ham and eggs"	"Food"	"Grilled ham and eggs"	"2290,00"	"2022-03-27T14:56:18.000000Z"	"2022-03-27T15:00:16.000000Z"				
24	"Vodka Soda"	"Cocktail"	"Kalinka Vodka and Soda 2dl"	"2999,00"	"2022-03-27T15:01:10.000000Z"	"2022-03-27T15:01:22.000000Z"				
25	"Goulash soup"	"Food"	"Pork chops, potato, vegetable, thick soup"	"2999,00"	"2022-03-27T15:13:49.000000Z"	"2022-03-27T15:13:49.000000Z"				
26	"Ginger tea"	"Drink"	"Hot natural ginger tea"	"799,00"	"2022-03-27T15:15:24.000000Z"	"2022-03-27T15:15:24.000000Z"				
27	"Black tea"	"Drink"	"Earl gray black tea"	"599,00"	"2022-03-27T15:16:43.000000Z"	"2022-03-27T15:16:56.000000Z"				
29	"Grilled chicken"	"Food"	"Spicy chicken grilled"	"3240,00"	"2022-04-08T19:07:48.000000Z"	"2022-04-13T19:10:46.000000Z"				
32	"Acacia honey"	"Food"	"Sweet acacia honey"	"999,99"	"2022-04-09T18:42:02.000000Z"	"2022-04-13T08:04:08.000000Z"				
33	"Honey"	"Food"	"Small portion of honey"	"799,00"	"2022-04-09T18:42:15.000000Z"	"2022-04-09T18:42:15.000000Z"				
34	"Gingerbread"	"Food"	"Sweet bread, allergic"	"955,00"	"2022-04-09T18:42:49.000000Z"	"2022-04-09T18:42:49.000000Z"				
35	"Chicken soup"	"food"	"Hot spicy soup by chicken"	"1499,00"	"2022-04-10T18:54:26.000000Z"	"2022-04-10T18:54:26.000000Z"				
36	"Ice-cream can chocolate"	"Sweets"	"chocolate Magnum"	"3499,00"	"2022-04-13T12:45:43.000000Z"	"2022-04-13T12:45:43.000000Z"				
37	"Lollipop"	"Sweets"	"multiple fruit flavour"	"899,00"	"2022-04-13T12:46:41.000000Z"	"2022-04-13T12:46:41.000000Z"				
38	"Haribo frog"	"Sweets"	"200g Haribo Frog pack"	"899,00"	"2022-04-13T14:37:34.000000Z"	"2022-04-13T14:37:34.000000Z"				
39	"Gin"	"Alcohol"	"0,5L Gin"	"4999,00"	"2022-04-17T12:42:43.000000Z"	"2022-04-17T12:42:43.000000Z"				
40	"Boiled rice"	"Food"	"Boiled jasmine rice"	"699,00"	"2022-04-17T12:52:42.000000Z"	"2022-04-17T12:53:57.000000Z"				
41	"Tequila"	"Alcohol"	"Silver Tequila Sierra 0,5L"	"7899,00"	"2022-04-17T17:50:54.000000Z"	"2022-04-17T17:50:54.000000Z"				

Eddig csak a weben és az asztali alkalmazás kapcsán említettük a Termékeket (Products), ám ezeket az api útvonalon is elhelyeztük egy REST API-ban ami Postman testing tool-al tesztelhető, a Postman kollekció megtalálható a következő utvonalon a projekt mappában: api_documentation & docs\ ProductControllerREST.postman_collection.json

Itt ugyanazokat a műveleteket tudják a userek és adminok végre hajtani, mint a weben a /products útvonalon, annyi különbséggel, hogy a REST API tesztelése folyamán konfigurálni kell a Postman applikációt.

Legkönyebben ezt a Postman megnyitása után a Collections -> import gombra kattintva -> importálni kell a mappában található kollekciót.

Fontos megemlíteni, hogy a login után kapott token-t fontos másolni,

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|RtziEfjuKxiw8gmUok7mbC2LV4V2ALR0vjiICkevD"
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 minden szigorúan autorizációhoz vannak kötve

The screenshot shows a Postman interface with the following details:

- Request URL:** http://localhost:8000/api/products
- Method:** POST
- Authorization:** Bearer Token (set to "33|RtrEfjuKxiw8gmUok7mbC2LV4V2ALR0vjljCkevD")
- Response Status:** 201 Created
- Response Time:** 609 ms
- Response Size:** 504 B
- Response Body (Pretty JSON):**

```
1  {
2   "name": "Tequila",
3   "category": "Alcohol",
4   "description": "Silver Tequila Sierra 0.5L",
5   "price": "7899",
6   "updated_at": "2022-04-17T17:50:54.000000Z",
7   "created_at": "2022-04-17T17:50:54.000000Z",
8   "id": 41
9 }
```

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 user-t visz fel akkor a bcrypt-el egy titok enkriptált jelszót is beinjectálunk az adatbázis user táblájába. Amennyiben egy user bajba kerülne és admin segítséget kérne mert jelszavát megszereztek, 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 adatbiztonság névében. 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 boksz-nál 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, vagy 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, keretrendszerk 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 szintaxis-sal. A webes keretrendszer struktúrát és kiindulópontot biztosít az alkalmazás létrehozásához, lehetővé téve, hogy valami csodálatos létrehozására összpontosítson, a fejlesztő ne a részleteken izzadon.

A Laravel arra törekszik, hogy elkepesztő fejlesztői élményt nyújtson, miközben olyan hatékony funkciókat biztosít, mint az alapos függőségi befecskendezés, a kifejező adatbázis-absztraktiós 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 egységesztekhez, a várólistákhoz, a valós idejű eseményekhez és még sok másnak. 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 gyorsítótárrendszerekhez, 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 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 könyvtárában 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 authentikációs nézetek és és bejelentkezett felhasználók kezelésének modern, rezponzív 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 authentikációt 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/hatóköreket 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 „fiókbeállításai” között van egy képernyő, ahol a felhasználó API-tokkent 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 tárak, 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 őrét 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-kérelmet, 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 authentikációt 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 másat. 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 ōrt, 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 oszályt 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 legtermékenyebb 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 interfészket 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 temékek 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 ás 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 állapottalanok, 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.

Table of contents

1 failed_jobs	Page number: 2
2 migrations	Page number: 3
3 password_resets	Page number: 4
4 personal_access_tokens	Page number: 5
5 products	Page number: 6
6 rooms	Page number: 7
7 sessions	Page number: 8
8 transactions	Page number: 9
9 users	Page number: 10
10 Relational schema	Page number: 11

1 failed_jobs

Creation: Mar 20, 2022 at 08:04 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	bigint(20)	UNSIGNED	No		auto_increment			
uuid	varchar(255)		No					
connection	text		No					
queue	text		No					
payload	longtext		No					
exception	longtext		No					
failed_at	timestamp		No	current_timestamp()				

2 migrations

Creation: Mar 20, 2022 at 08:04 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	int(10)	UNSIGNED	No		auto_increment			
migration	varchar(255)		No					
batch	int(11)		No					

3 password_resets

Creation: Mar 20, 2022 at 08:04 PM
Last update: Apr 17, 2022 at 10:11 AM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
email	varchar(255)		No					
token	varchar(255)		No					
created_at	timestamp		Yes	NULL				

4 personal_access_tokens

Creation: Mar 20, 2022 at 08:04 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	bigint(20)	UNSIGNED	No		auto_increment			
tokenable_type	varchar(255)		No					
tokenable_id	bigint(20)	UNSIGNED	No					
name	varchar(255)		No					
token	varchar(64)		No					
abilities	text		Yes	NULL				
last_used_at	timestamp		Yes	NULL				
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				

5 products

Creation: Apr 08, 2022 at 08:45 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	bigint(20)	UNSIGNED	No		auto_increment			
name	varchar(255)		No					
category	varchar(255)		No					
description	varchar(255)		Yes	NULL				
price	decimal(7,2)		No					
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				

6 rooms

Creation: Mar 20, 2022 at 08:04 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	bigint(20)	UNSIGNED	No		auto_increment			
room_type	varchar(255)		No					
price	decimal(8,2)		No					
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				

7 sessions

Creation: Mar 20, 2022 at 08:04 PM
Last update: Apr 17, 2022 at 10:12 AM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	varchar(255)		No					
user_id	bigint(20)	UNSIGNED	Yes	NULL				
ip_address	varchar(45)		Yes	NULL				
user_agent	text		Yes	NULL				
payload	text		No					
last_activity	int(11)		No					

8 transactions

Creation: Apr 15, 2022 at 10:57 PM
Last update: Apr 17, 2022 at 10:12 AM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	bigint(20)	UNSIGNED	No		auto_increment			
user_id	bigint(20)	UNSIGNED	Yes	NULL		-> users.id ON UPDATE CASCADE ON DELETE CASCADE		
user_name	varchar(255)		Yes	NULL				
room_id	bigint(20)	UNSIGNED	Yes	NULL		-> rooms.id ON UPDATE CASCADE ON DELETE CASCADE		
room_name	varchar(255)		Yes	NULL				
checkin	date		No					
checkout	date		No					
days	int(11)		No					
bill	int(11)		No					
halfboard	varchar(255)		No					
number_of_guests	int(11)		No					
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				
email	varchar(255)		Yes	NULL				
phone	varchar(255)		Yes	NULL				

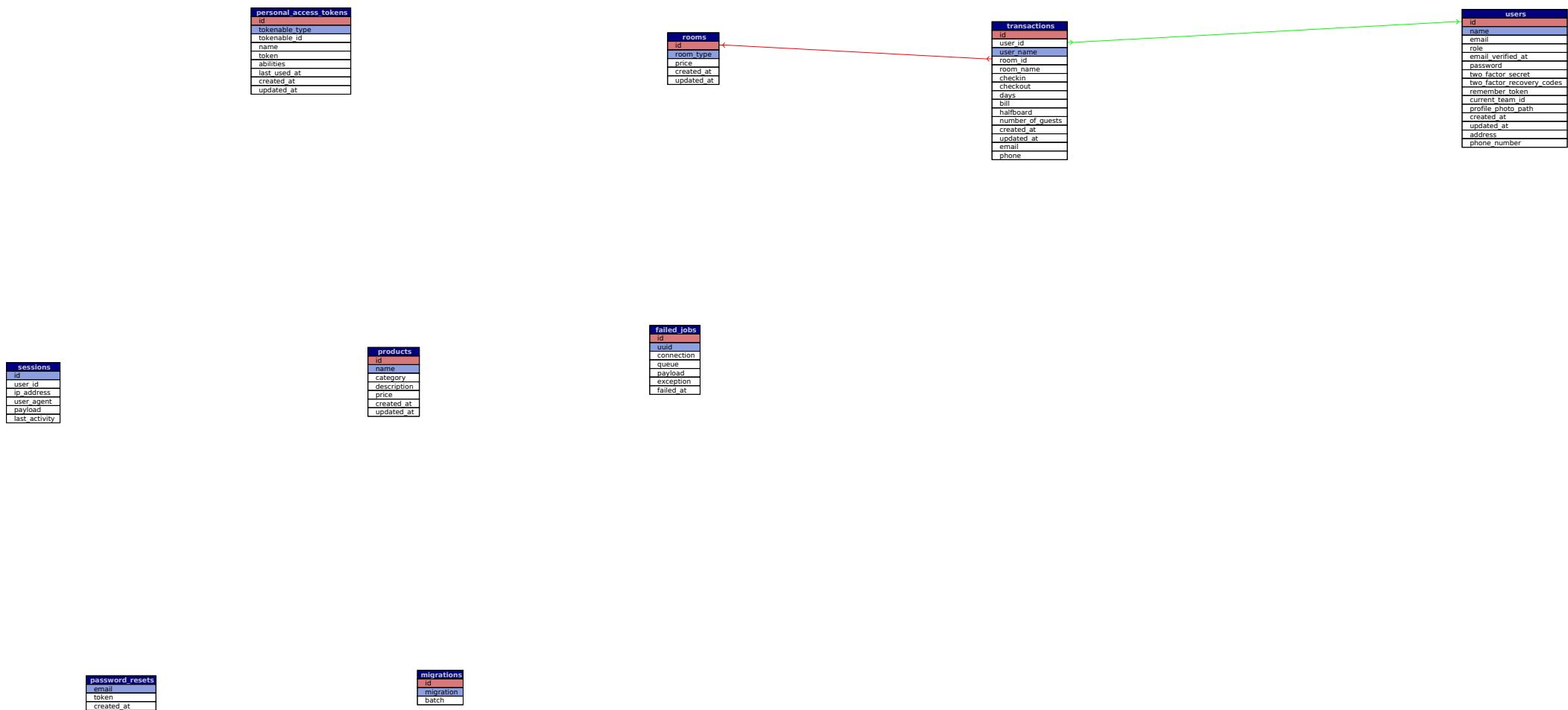
PDF export page

9 users

Creation: Mar 20, 2022 at 08:04 PM
Last update: Apr 17, 2022 at 10:12 AM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
id	bigint(20)	UNSIGNED	No		auto_increment			
name	varchar(255)		No					
email	varchar(255)		No					
role	varchar(255)		No	User				
email_verified_at	timestamp		Yes	NULL				
password	varchar(255)		No					
two_factor_secret	text		Yes	NULL				
two_factor_recovery_codes	text		Yes	NULL				
remember_token	varchar(100)		Yes	NULL				
current_team_id	bigint(20)	UNSIGNED	Yes	NULL				
profile_photo_path	varchar(2048)		Yes	NULL				
created_at	timestamp		Yes	NULL				
updated_at	timestamp		Yes	NULL				
address	varchar(255)		Yes	NULL				
phone_number	varchar(255)		Yes	NULL				

PDF export page



3.5 Útvonalak listája

A route:list parancssal 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 ellenőri 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

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	routestocsv		Closure	web
GET HEAD	routestocsv		Closure	App\Http\Middleware\Authenticate
GET HEAD	routestocsv		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	routestocsv		Closure
GET	jsonproducts		App\Http\Controllers\ProductController@index

3.6 Modellek bemutatása

Product model: a Product Model egy olyan osztály, amely a termékek kezelésére alapul szolgáló adattábla logikai szerkezetét és kapcsolatát képviseli. A Product Model lehetővé teszi az adatok lekérését, beillesztését és frissítését az adattáblázatba.

UserProducts model: a UserProduct Model egy olyan osztály, amely a vendégek számára a termékek lekérésére szolgáló adattábla logikai szerkezetét és kapcsolatát képviseli, a CRUD-ból csak a Read van ehhez a modelhez társítva a Controller/ Component-ben és a Viewban. A UserProduct Model lehetővé teszi az adatok lekérését adattáblázatból.

Room model: a Room Model egy olyan osztály, amely a szobák kezelésére alapul szolgáló adattábla logikai szerkezetét és kapcsolatát képviseli. A Laravelben minden adatbázistáblához tartozik egy megfelelő „Modell”, amely lehetővé teszi számunkra, hogy kapcsolatba léphessünk az adott táblával. A Room Model lehetővé teszi az adatok lekérését, beillesztését és frissítését a rooms adattáblázatba.

Transaction model: a Transaction Model egy olyan osztály, amely a foglalások kezelésére alapul szolgáló transactions adattábla logikai szerkezetét és kapcsolatát képviseli. A Transaction Model lehetővé teszi az admin számára a foglalások és adatok lekérését, beillesztését és frissítését a transactions adattáblázatba.

User model: a User Model egy olyan osztály, amely a felhasználók kezelésére alapul szolgáló adattábla logikai szerkezetét és kapcsolatát képviseli. A User Model lehetővé teszi a felhasználói adatok lekérését, beillesztését és frissítését a users adattáblázatba, illetve a login és register authentikáció is a users táblában történik.

UserTransaction model: a UserTransaction Model egy olyan osztály, amely a vendég foglalások kezelésére a vendégek(User role) számára a transactions táblába alapul szolgáló adattábla logikai szerkezetét és kapcsolatát képviseli. A UserTransaction Model lehetővé teszi a vendégek számára a sajét foglalásaik és annak adatainak lekérését, beillesztését és frissítését a transactions adattáblázatba.

3.7 Controller-ek és AdminMiddleware bemutatása

AuthController: ez a Controller határozza meg az admin illetve felhasználó hitelesítést a api route-n keresztül. Ez validálja a felhasználókat a regisztráció folyamán a **register** funkcióval, ahol meghatározza mi legyen a Request body-ja. A mi esetünkben ezek a

```
name' => 'required|string',
'email' => 'required|string|unique:users,email',
'password' => 'required|string|confirmed'
```

Amire sikeresség esetén králunk egy új Felhasználót az adatbázisban és visszaadunk egy bearer token-t egy 201-es válasszal.

A **login** funkció sikeresség esetén, ha az emailt és jelszót sikeresen validáltuk visszaad egy 200-as Http választ egy tokennel, míg sikertelenség esetén egy 401-es 'Wrong credentials entered' üzenetet. Az AuthController által adott Sanctum tokennel lehetséges a ProductController protected route-ait használni.

A ProductController felelős az api route-n a termékek kezeléséért a CRUD operátorokkal http metódusok által a **store, show, update, destroy** illetve a keresésért a **search** funkciókkal.

A HomeController felelős azért, hogy hitelesítse a felhasználót a weben az Auth osztály által majd azt a view-t adja vissza usernek amit mi szeretnénk a /user/home URI alatt ahogy beállítottuk a public const HOME = '/user/home';. Admin esetén ez a 'admin.dashboard' view fájl, míg user role esetén a 'dashboard' fájl.

A ProductController a webes route termékek kezésére írodott Inertia által.

A TransactionController a foglalások webes kezelésére írodott Inertia által a /transactions view-ra.

A UserProductsController a vendégek webes termék nézetére írodott Inertia által a /restaurantandbar view-ra.

Minden view ami Controller vagy Component által csak adminok részére van szánva AdminMiddleware-l van védve és logged in user esetén ha probálná eléri egy bejelentkezett felhasználó 403-as response-t kap mind a weben mind az api-n.

A UserTransactionController a vendégek saját foglalásainak webes kezelésére írodott Inertia által.

3.8 Livewire Component-ek bemutatása

A **Livewire/Products** Component felelős a /products viewen a termékek kezeléséért, ami admin felhasználók számára lett létrehozva Ez a komponens helyettesíti a Controllert a **render** funkció felelős a products view-nak való átadásának és ezen belül a **\$searchterm4** felelős a search bar míg a **->paginate()** a pagination működéséért

A **create** a pop-up modal-t kelti életre.

Az **openModal()** megnyitja a modalt egy igazra állítással.

A **closeModal()** bezárja a felugró modal window-t.

A **cleanupFields()** kiüríti a modalban található sorokat.

Az **edit(\$id)** a termék szerkesztését felelős, ami az Edit gomb megnyomásával kel életre a front-enden.

A **delete(\$id)** a termék törlését felelős, ami a Delete gomb megnyomásával kel életre a front-enden.

A **save()** pedig a mentésért felelős **updateOrCreate (['id'=>\$this->product_id],** metódussal valósítunk meg azaz a **save()**-n belül lehet frissítés vagy létrehozás is.

A sikeres végrehajtás üzeneteket is innen adjuk át a front-endnek a **session()->flash('message',..);** megoldással.

A **Livewire/Transactions** Component felelős a foglalások létrehozásáért, törléséért, szerkesztéséért, ami admin felhasználók számára lett létrehozva a /transactions view-n.

Ez a komponens helyettesíti a Controllert a `render` funkció felelős a tranzakciók view-nak való átadásának és ezen belül a `$transactionsToList` felelős a search bar míg a `->paginate()` a pagination működéséért. Ez adja tovább a frontenden lévő `foreach` ciklusnak a `$transactions`-okat.

A `create` a pop-up modal-t kelti életre.

Az `openModal()` megnyitja a modalt egy igazra állítással.

A `closeModal()` bezárja a felugró modal window-t.

A `cleanupFields()` kiüríti a modalban található sorokat.

Az `edit($id)` a foglalás szerkesztésért felelős, ami az Edit gomb megnyomásával kel életre a front-enden.

A `delete($id)` a foglalás törlésért felelős, ami a Delete gomb megnyomásával kel életre a front-enden.

A `save()` pedig a mentésért felelős `updateOrCreate` (`['id'=>$this->product_id]`, metódussal valósítunk meg azaz a `save()`-n belül lehet frissítés vagy létrehozás is.

A sikeres végrehajtás üzeneteket is innen adjuk át a front-endnek a `session()->flash('message',..);` megoldással.

A `self::CreateUser();` az adminos szobafoglalás felvitelnél létrehozza a user-t is, vagy updatejük (`updateOrCreate` metódus) ha már létezik.

`private function CreateUser()` az admmin által beírt user felvitele táblába, vagy név alapján updatelése /user id vissza a transactions táblába.

`public function GetRoom()` Név alapján lekérdezzük a szoba adatait

`public function UpdateDays()` Napok kiszámolása checkin és checkout datekból, ha a különbség negatív, akkor a checkout dátum a checkin előtt van akkor nem adunk vissza automatikus bill és day értéket amit ezekkel kalkulálni szeretnénk hanem flash error message: 'Checkout date must be after checkin!'

`public function UpdateBill()` -> Összeg kiszámolása szobafajtából és napok számából

`private function IsRoomAvailable()` -> itt megvizsgáljuk, hogy a kiválasztott időszakra a szoba foglalt-e

`function getRandomBytes ($nbBytes = 32)` randomByte generáló, amit a `function generatePassword($length)` használ, hogy random jelszót generálunk a foglalás alatt admin által létrehozott usernek

A **Livewire/UserProducts** Component felelős a /restaurantandbar viewen a termékek megjelenítéséért, ami vendég felhasználók számára lett létrehozva. Ez a komponens helyettesíti a Controllert a `render` funkció felelős a products view-nak való átadásának és ezen belül a `$searchterm3` felelős a search bar míg a `->paginate()` a pagination működéséért. Ez a **Livewire/Products** klónja csak itt a front end nem hív meg CRUD fülekötöket csak a Read metodus van kihelyezve a User role alatt bejelentkezett vendégeknek.

A **Livewire/UserTransactions** Component felelős azért hogy a vendégek saját foglalásaikat létre tudják hozni, törleni, szerkeszten. Ez a **Livewire/Transactions** klónja csak itt a userek csak saját foglalásaikat kezelik és látják a /my-bookings view-n.

Ez a komponens helyettesíti a Controllert a `render` funkció felelős a tranzakciók view-nak való átadásának és ezen belül, ez adja tovább a frontenden lévő `foreach` ciklusnak a `$transactions`-okat. Itt nincs search és pagination egyenlőre.

A `create` a pop-up modal-t kelti életre.

Az `openModal()` megnyitja a modalt egy igazra állítással.

A `closeModal()` bezárja a felugró modal window-t.

A `cleanupFields()` kiüríti a modalban található sorokat.

Az `edit($id)` a foglalás szerkesztéséért felelős, ami az Edit gomb megnyomásával kel életre a front-enden.

A `delete($id)` a foglalás törléséért felelős, ami a Delete gomb megnyomásával kel életre a front-enden.

A `save()` pedig a mentésért felelős `updateOrCreate ([id=>$this->product_id],` metódussal valósítunk meg azaz a `save()`-n belül lehet frissítés vagy létrehozás is.

A sikeres végrehajtás üzeneteket is innen adjuk át a front-endnek a `session()->flash('message',..);` megoldással.

A `self::CreateUser();` az adminos szobafoglalás felvitelnél létrehozza a user-t is, vagy updatejük (`updateOrCreate` metódus) ha már létezik.

`private function CreateUser()` az admin által beírt user felvitele táblába, vagy név alapján updatelelse /user id vissza a transactions táblába.

`public function GetRoom()` Név alapján lekérdezzük a szoba adatait

`public function UpdateDays()` Napok kiszámolása checkin és checkout datekből, ha a különbség negatív, akkor a checkout dátum a checkin előtt van akkor nem adunk vissza automatikus bill és day értéket amit ezekkel kalkulálni szeretnénk hanem flash error message: 'Checkout date must be after checkin!'

`public function UpdateBill()` -> Összeg kiszámolása szobafajtából és napok számából

`private function IsRoomAvailable()` -> itt megvizsgáljuk, hogy a kiválasztott időszakra a szoba foglalt-e

`function getRandomBytes ($nbBytes = 32)` randomByte generáló, amit a `function generatePassword($length)` használ, hogy random jelszót generálunk a foglalás alatt admin által létrehozott usernek

A **Livewire/Users** Component felelős a felhasználók/vendégek (User role) létrehozásáért, törléséért, szerkesztéséért, ami admin felhasználók számára lett létrehozva a /users view-n és a Users Modellel dolgozik együtt. Az admin felhasználó itt akár egy meglévő User role helyett tud adni Admin role-t is, valamint Password reset-et is meg tud valósítani.

`protected $rules` = felelős az email validálásáért hogy ne legyen exception mert az admin felvisz egy olyan emailt ami már benne van a users táblában

Ez a komponens helyettesíti a Controllert a `render` funkció felelős a user view-nak való átadásának és ezen belül a `$searchTerm2` felelős a search bar míg a -

>`paginate()` a pagination működéséért. Ez adja tovább a frontenden lévő `foreach` ciklusnak a `$users`-eket.

A `create` a pop-up modal-t kelti életre.

Az `openModal()` megnyitja a modalt egy igazra állítással.

A `closeModal()` bezárja a felugró modal window-t.

A `cleanupFields()` kiüríti a modalban található sorokat.

Az `edit($id)` a user szerkesztésért felelős, ami az Edit gomb megnyomásával kel életre a front-enden.

A `delete($id)` a user törlésért felelős, ami a Delete gomb megnyomásával kel életre a front-enden.

A `save()` pedig a mentésért felelős `updateOrCreate ([id'=>$this->product_id],` metódussal valósítunk meg azaz a `save()`-n beül lehet frissítés vagy létrehozása is a usernek.

A sikeres végrehajtás üzeneteket is innen adjuk át a front-endnek a `session()->flash('message',..);` megoldással.

`function getRandomBytes ($nbBytes = 32)` Kriptográfiailag biztonságos pszeudo-véletlen bájtokat generál amit

`public function generatePassword($length)` használ fel jelszó generáláshoz

`public function passwordReset()` -et használjuk, hogy random jelszót generálunk a Password reset gombra kattintva az Edit modal-on belül a `createusers.blade` view-ban, ezáltal az admin által `User::updateOrCreate` -n beül a `password' => bcrypt(self::generatePassword(12))`, -el bcrypt funkciót belül bcrypt titkosítással generáljuk a random stringet(12 karakteres jelszó felső/alsó/számok) a password oszlopba a users táblában.

3.9 Front-end, UI/ UX bemutatása

A Front enden blade templating technológiát `@extends('layouts.master')` is használunk Tailwind CSS és Boostrap 5-el tettük reszponzívvá a képeket és konténereket.

Minden Livewire Component-hez tartozik egy view is, amit a resources\views\livewire mappában lehet megtalálni. Ezekben a view-kban az az actionoket `wire:model` és `wire:click`, `wire:key` és `wire:defer`, `wire:click.prevent`, `wire:model.defer` akciókkal hívjuk meg, manipuláljuk a, illetve keltjük életre. Leginkább wire:modellt használva (pl.: \$xyz)nyilvános tulajdonságokat az összetevő osztályban, minden alkalommal, amikor egy bemeneti elem ezzel a direktívával frissül, a tulajdonságot szinkronizáljuk az értékével.

A home pagek az index.blade.php és a welcome.blade.php az `@include('home.social')` például tehát a home mappa fájlait include-áljuk per szekció a clean code névében. A date picker amit használunk az a `HTML5 input type=date` a táblákat a tailwind CSS-el formázzuk, a success messageket session és flash message, hibaüzenteket `@error` funkcióval hívjuk meg, a sessionoket pedig a `@if(session()->has(''))/{{ session('PasswordMessage') }}@endif` megolást használjuk

A tábla nézetekben(/users & /transactions & /my-bookings & /products) a felugró ablakot Tailwind Modal component-et használva valósítottunk meg. A képek egy része CDN-ről jelenik meg míg a másik része lokálisan a public\images mappában található meg, minden kép tudomásunk szerint MIT és free open source library-ből származik.

3.10 Asztali alkalmazás bemutatása

Az asztali alkalmazás a .NET http kliense által rácsatlakozik a <http://127.0.0.1:8000/jsonproducts> URL-ra ahonnan lehívja a json adatok és azokat WinForms-on belül DataGridView-ban megjeleníti a Get Data gomb megnyomására. A CSV export megnyomására pedig .csv állományban el lehet menteni. Mivel a /jsonproducts URL-t és az ott található json-okat a Laravel REST API szolgáltatja így ha a lokális Laravel devolpment szerver nem fut akkor a programban beállított URL nem elérhető így asztali alkalmazás nem tudja consume-olni a RESful service-t. `public void CreateCSVFile()` -ban CSV fájl feltöltése történik a data grid view adatokkal majd annak kiírása `using StreamWriter sw = new(this.FileName);` pedig a Streamwriter a kírást végzi a fájlnak

`class Products` ez az osztály határozza meg mik lesznek az oszlopokban mit is tartalmaz a Product osztály

`private void button1_Click` ez a gomb amin belül meghívjuk a URI-t majd azt átadjuk a datagridview-nak

`private void ButtonCSV_Click` ezen belül adjuk meg az extensiont:

```
using SaveFileDialog browser = new();
    browser.Filter = "CSV (*.csv)|*.csv";
```

majd pedig kiexportáljuk a CSV fájlt:

```
CSVExport export = new(browser.FileName, dataGridView1);
    export.CreateCSVFile();
```

4. Tesztelés

BUDAPEST HOTELS

ISMERT FUNKCIÓK TESZTELÉSE
(WEBALKALMAZÁS)

TESZTELÉSI TERV

KÉSZÍTETTE:
WELSCH ÁDÁM
SZIKSZAI ÁKOS
CSIZMÁR DÁNIEL FERENC

TESZTELÉSI TERV AZONOSÍTÓJA: BH-HMS 1.00

UTOLSÓ FRISSÍTÉS: 2022.04.18.

1. A TERMÉK ÉS VERZIÓI

VERZIÓ	DÁTUM	KÉSZÍTETTE
--------	-------	------------

1.0 2022.04.13.

W.Á.; Sz.Á.;Cs.D

2. BEVEZETÉS

A tesztelési terv célja a HMS rendszer alkalmazás **ismert funkcióinak részletes tesztelése, vizsgálata** webes felületen, illetve annak Api tesztelése Postman segítségével.

A HMS alkalmazás egy szálloda alkalmazása, amely azzal a céllal jött létre, hogy:

- az admin könnyedén tudjon módosításokat eszközölni (esetleges akciók, éttermi ajánlat, szoba árak, karrier lehetőségek publikálása), illetve a vendégek által leadott foglalások kezelése és azok nyomon követése.

- a vendégek foglalásokat tudjanak végezni (log-in felület regisztráció után), foglalásokat módosítani/törölni.

3. TESZTELÉSHEZ FELHASZNÁLT DOKUMENTUMOK

- Tesztelési terv (HMS-1.0_testplan.pdf)
- Tesztelési jegyzőkönyv (HMS-1.0_testcases.xlsx)

4. LESZÁLLÍTANDÓ TESZT DOKUMENTUMOK

Teszt forgatókönyvek, Teszteset struktúra és tesztesetek, Tesztelési jegyzőkönyv, Tesztkészletek, Hibajelentés

5. ELVÁRÁSOK

- a tesztelők ismerik az alapdokumentumokat, amelyek meghatározzák a rendszert
- a tesztelés folyamatának alapját ez a dokumentum, a Tesztelési terv képezi

6. TESZTELT TULAJDONSÁGOK ÉS FUNKCIÓK

Nem funkcionális tesztek:

- HMS alapfelület megjelenik hibaüzenet nélkül
- HMS Admin felületre való belépés
- HMS Felhasználói felületre való belépés

Funkcionális tesztek:

- oldalakon a gombok működnek - a megfelelő helyre navigál
- foglalás lehetséges a kritériumoknak megfelelően
- belépés Admin/Felhasználói felületre lehetséges
- regisztráció működik
- felhasználói login funkciók
- admin login funkciók

System Integration Teszt:

- a HMS rendszer szinkronban van az Mail kiszolgálóval [Mailtrap.io] (jelszó módosítás, visszaállítás, regisztrálásnál e-mailes visszaigazolás)

6.1 NEM FUNKCIIONÁLIS TESZTEK

- oldalak helyesen jelennek meg
- oldalakon a gombok elhelyezkedése egységes és a tervnek megfelelő
- képek megjelennek az oldalakon
- helyesírási hibák nincsenek
- az oldal reszponzív

6.2 FUNKCIIONÁLIS TESZTEK

- főoldalról a Login gomb megnyomásával a login ablakra navigált az alkalmazás
- ha admin email címet és jelszót adunk meg, akkor az admin felület megnyílik
- ha felhasználóval lépünk be a Felhasználói felület nyílik meg
- elfelejtett jelszó és e-mail cím megadása esetén e-mail értesítés érkezik a Mailtrap.io oldalra
- register oldal gomb megnyomására a regisztrációs oldal ugrik fel
- legyenek írhatóak a mezők
- lap alján az elfogadó checkbox legyen kitölthető (nélküle ne engedje a regisztrációt)
- register gomb megnyomásával navigáljon a verifikáló oldalra
- fő oldalról a "Book Now" gombra kattintva a felhasználói login felület ugrik fel, ahol a helyes felhasználó és jelszó megadása után a Secure Bookings opciónál van lehetőség folytatni a foglalást.

Felhasználói foglalás kezelés:

Secure Bookings:

- New: Új foglalást lehet leadni. minden adatot ki kell tölteni és helyesen kell eltárolnia -> adatbázisba kell kerülnie az adatoknak és más foglalást nem szabad engedni az adott időpontra ha be van fejezve a foglalás.
- Edit: A már leadott foglalást lehet módosítani hasonló módon kötelező az összes adat megadása, és ha már van foglalás az adott napra, akkor nem engedheti a módosítás befejezését.
- Delete: A foglalás törlését eredményezi. Adatbázis szintű törlés történik. Egy ismételt foglalás a korábban törölt időpontban működöképes kell, hogy legyen.

Media:

- az Our Instagram gomb megnyomása az Instagram bejelenkező felületére navigál (Nincs még Instagram profilja a Budapest Hotels-nek)
- Our Hotel - Watch video gomb megnyomása a szálloda youtube csatornáján levő bemutató videót indítja el egy új oldalon.

- Our gallery - Direction gomb megnyomása a googleMaps alkalmazást nyitja meg.

- MenuBar:

- HomePage: Nyitó oldalra navigál
- Menu & Bar: Ételek, Italok listája árakkal
- Dashboard: Kezelőfelület megnyitása (User Center)
- My Bookings: Foglalások és azoknak kezelése

Jobb felső sarok - Monogram

- Profil módosítás:

- Photo hozzáadása
- Jelszó frissítése
- Két faktoros authentikáció
- Browser Session
- Delete Account

- Home - Dashboard

- Menu - Ételek, Italok listája árakkal

- LogOut - Kijelentkezés

Admin felület

Menu Bar működése

- Home Page: Főoldal megnyitása
- Menu and Bar: Ételek, Italok listája árakkal (readonly) -> Kereső mező működik
- Dashboard: Dashboard megnyitása (User Center)
- My bookings: foglalások kezelőfelülete

Users Admin Center - Manage Users gomb megnyomásával a regisztrált felhasználók felülete nyílik meg.

- Search Bar működik (Name, Role, Email, Address, Phone Number)
- New gomb új felhasználó hozzáadását teszi lehetővé
- Edit: Felhasználó módosítása gomb - felhasználó módosítása

Reset Password: Email kiküldése működik

Save működik -> frissülnek az adatok az adatbázisban

Booking Admin Center - Manage Booking gomb megnyomása a foglalások kezelés felületére navigál.

- Back to Admin Center: Dashboard felületre navigál vissza
- Search Bar működik, ad találatot (User Name, Email, Phone, Room Name)
- Edit gomb: booking módosítása
- Delete: Foglalás törlése
- New: Új foglalás létrehozása (Recepciós ad-hoc foglalás esetén)

Product Admin Center - Manage Products gomb megnyomása Ételek, Italok listája árakkal oldalra ugrik -> Kereső mező működik és találatot ad (Name, Category, Description).

- New: Új termék felvétele a listára
- Edit: Termék módosítása
- Delete: Termék törlése

Az újonnan felvett, módosított vagy törölt termékek azonnal frissítésre kerülnek az adatbázisban és a felületen is.

6.3 SYSTEM INTEGRATION TESZTEK

Az alkalmazás jelenleg csak egy külső alkalmazást használ, ami a mailtrap.io.

Ennek működését igazolni és jóváhagyni szükséges a következő módokon:

- főoldalról regisztráció, végén email verifikáció szükséges
- Login felületen - Forgot your password? esetén email cím megadása -> email kiküldése

7. STRATÉGIA

- a tesztelők előkészítik a teszeseteket, ha készen van a specifikáció
- a tesztelők előkészítik a tesztadatokat
- fejlesztők integrációs teszteket készítenek
- manuálisan végezik a tesztelők a funkció/nem funkció teszteket és regressziós tesztet a teszt környezetben
- API tesztelésre és a Backend működésének ellenőrzésére Postman van használva, aminek "collection" file-ja a csomag része

- terheléses tesztek futtatása a tervezett kapacitás ellenőrzésére
- adatmentés és módosítás esetén visszaellenőrzés szükséges az adatbázisban

8. ELFOGADÁSI KRITÉRIUMOK

- funkció és regressziós tesztek 100%-a sikeres
- integrációs tesztek lefedik a kódbázist
- kritikus hiba nincs a rendszerben
- tesztelés azonnali felfüggesztése szükséges súlyos hiba esetén, vagy ha a tesztkörnyezet / hardver / szoftver nem elérhető - azonnali javítás a kódban
- tesztelés újrakezdhető, ha a fenti problémák elhárultak és a súlyos hibák javításra kerültek

9. SZÜKSÉGES ERŐFORRÁSOK

9.1 SZÜKSÉGES ERŐFORRÁSOK

- 3 fő fejlesztő, 3 fő tesztelő
- 3 fős csapat a fejlesztői és tesztelői gárda -> fontos szempont a független tesztelés

9.2 SZEREPKÖRÖK, FELADATOK

- tesztelő: tesztesetek írása, futtatása, és eredmények dokumentálása
- fejlesztő: fejlesztés kivitelezése, unit tesztek végrehajtása, hibajavítás
- projekt menedzser (Welsch Ádám): erőforrások biztosítása, ütemezésekkel, határidőkkel kapcsolatos döntések, elkészült tesztelés elfogadása

9.3 ESZKÖZ ERŐFORRÁSOK

- PC munkaállomás Microsoft Windows 10 operációs rendszerrel
- Chrome Verzió: 100.0.4896.88 (Hivatalos verzió) (64 bites)

9.4 TESZTKÖRNYEZET

- HMS Tesztrendszer tesztkörnyezet "Test" adatbázissal
- <http://127.0.0.1:8000/index>

9.5 TESZTADATOK

- a Budapest Hotels tesztadatainak felvétele manuálisan történik

10. KOCZKÁZATOK

- erőforráshiány lép fel a projekt haladása során
- a képzések meghiúsulnak, amelyeken a projekt résztvevők továbbképzése történik
- tesztelési feladatok időigényének alábecslése

11. MENETREND, HATÁRIDŐK

TESZTELÉSI FÁZIS	DÁTUM
Tesztesetek elkészítése	2022. március 16.
Funkcionális tesztelés	2022. április 4.
User Acceptance tesztek	2022. április 18.

5. Összefoglalás

Véleményünk szerint a projekt sikeres lett sok fejleszteni való van még hátra szeretnénk select dropdown filterrel kiegészíteni az össze tábla view-t szeretnénk PDF és CSV extractot a weben a táblák adataiból kinyerni és szeretnénk, ha a weben CSV upload-ot is kivitelezni tudnának a userek. Azaz a táblákat felölteni a CSV fájlból nyert adatokkal, amiket a weben szeretnénk feldolgozni. Szeretnénk csinálni egy room numbers táblát, ahol akár több száz hotel szobát is hozzá tudunk rendelni a rooms táblához ahol jelenleg szoba típusok és annak árai ID-ei vannak. Szeretnénk csinálni egy stock táblát, ami products tábla készletét tartaná számon és dinamikus azt jelenítenénk emg ami még elérhető termék ami kifogyott annak nincs view és egy auto trigger mail küldés lenne az adminok felé. Szeretnénk a front-enden egy contact us formot kitenni aminek megvan már a view-ja és azt PHPMailer Frameworkkel mail küldésére alkalmassá tenni. Az asztali alkalmazást is sok funkcióval szeretnénk bővíteni, hogy ERP funkciókat is akár el tudjon látni.

A projekt folyamán a Laravel framework illetve PHP end to end(több éves programozói tudás) tudásnak hiánya illetve ennek a megismerésének elmélyedése és állandó hibákba, bugokba futás jelentette a legnagyobb nehézséget de ezáltal is tanultunk a lehető legtöbbet.

A password reset generálás valamint a front end és back end összekötése jelentette a legnagyobb szakmai kihívást, valamint a sok különféle rendszer pl. Laravel, Jetstream, Fortify, Sanctum, .NET, Livewire, PHP ezeknek az együttes kezelése és megismerése.

A jövőben szeretnénk a web appot PWA-vá tenni azaz Progresszív Web Applikációjává, illetve átalakítani időpont foglalójával, hogy szerelők, fodrászok, orvosi irodák, fogászatok, műkörmösök is tudják hatékonyan használni.

Összességében mindenki egyetértünk, hogy egy modern, responszív, dinamikus full stack web app-et tudunk létrehozni ami nagy siker számunkra.

6. Kiegészítés

API dokumentációk elérése a weben:

1. Swagger:
 - a. 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 Sunrire
        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 successfully 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|13cQirhIWpgDpe7vi6ngDMAxnIsjXMObGWhTtwCt  
          '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.
```

LRD

Export to Postman (Open API 3.0.0)

Routes List

GET [_ignition/health-check](#)

HTTP Method

URL

Controller

Controller Method

Middleware 1

GET

http://127.0.0.1:8000/_ignition/health-check

Facade\Ignition\Http\Controllers\HealthCheckController

@__invoke

Facade\Ignition\Http\Middleware\IgnitionEnabled

Try

GET [_ignition/scripts/{script}](#)

HTTP Method

URL

Controller

Controller Method

Middleware 1

GET

http://127.0.0.1:8000/_ignition/scripts/{script}

Facade\Ignition\Http\Controllers\ScriptController

@__invoke

Facade\Ignition\Http\Middleware\IgnitionEnabled

Try

GET [_ignition/styles/{style}](#)

HTTP Method

URL

Controller

Controller Method

Middleware 1

GET

http://127.0.0.1:8000/_ignition/styles/{style}

Facade\Ignition\Http\Controllers\StyleController

@__invoke

Facade\Ignition\Http\Middleware\IgnitionEnabled

Try

GET login

HTTP Method

GET

URL

<http://127.0.0.1:8000/login>

Controller

Laravel\Fortify\Http\Controllers\AuthenticatedSessionController

Controller Method

@create

Middleware 1

web

Middleware 2

guest:web

Try**GET** forgot-password

HTTP Method

GET

URL

<http://127.0.0.1:8000/forgot-password>

Controller

Laravel\Fortify\Http\Controllers>PasswordResetLinkController

Controller Method

@create

Middleware 1

web

Middleware 2

guest:web

Try**GET** reset-password/{token}

HTTP Method

GET

URL

<http://127.0.0.1:8000/reset-password/{token}>

Controller

Laravel\Fortify\Http\Controllers\NewPasswordController

Controller Method

@create

Middleware 1

web

Middleware 2

guest:web

Try

GET register

HTTP Method

GET

URL

<http://127.0.0.1:8000/register>

Controller

Laravel\Fortify\Http\Controllers\RegisteredUserController

Controller Method

@create

Middleware 1

web

Middleware 2

guest:web

Try**GET** email/verify/{id}/{hash}

HTTP Method

GET

URL

<http://127.0.0.1:8000/email/verify/{id}/{hash}>

Controller

Laravel\Fortify\Http\Controllers\VerifyEmailController

Controller Method

@__invoke

Middleware 1

web

Middleware 2

auth:web

Middleware 3

signed

Middleware 4

throttle:6,1

Try**GET** user/confirm-password

HTTP Method

GET

URL

<http://127.0.0.1:8000/user/confirm-password>

Controller

Laravel\Fortify\Http\Controllers\ConfirmablePasswordController

Controller Method

@show

Middleware 1

web

Middleware 2

auth:web

Try

GET user/confirmed-password-status

HTTP Method

GET

URL

<http://127.0.0.1:8000/user/confirmed-password-status>

Controller

Laravel\Fortify\Http\Controllers\ConfirmedPasswordStatusController

Controller Method

@show

Middleware 1

web

Middleware 2

auth:web

Try**GET** two-factor-challenge

HTTP Method

GET

URL

<http://127.0.0.1:8000/two-factor-challenge>

Controller

Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController

Controller Method

@create

Middleware 1

web

Middleware 2

guest:web

No.	Attributes	Type	Nullable	Bail	Rules
1	code	String	Nullable		
2	recovery_code	String	Nullable		

Try**GET** user/two-factor-qr-code

HTTP Method

GET

URL

<http://127.0.0.1:8000/user/two-factor-qr-code>

Controller

Laravel\Fortify\Http\Controllers\TwoFactorQrCodeController

Controller Method

@show

Middleware 1

web

Middleware 2
Middleware 3

auth:web
password.confirm

Try

GET user/two-factor-recovery-codes

HTTP Method
URL
Controller
Controller Method
Middleware 1
Middleware 2
Middleware 3

GET
<http://127.0.0.1:8000/user/two-factor-recovery-codes>
Laravel\Fortify\Http\Controllers\RecoveryCodeController
@index
web
auth:web
password.confirm

Try

GET terms-of-service

HTTP Method
URL
Controller
Controller Method
Middleware 1

GET
<http://127.0.0.1:8000/terms-of-service>
Laravel\Jetstream\Http\Controllers\Livewire\TermsOfServiceController
@show
web

Try

GET privacy-policy

HTTP Method
URL
Controller
Controller Method

GET
<http://127.0.0.1:8000/privacy-policy>
Laravel\Jetstream\Http\Controllers\Livewire\PrivacyPolicyController
@show

Middleware 1

web

Try**GET** [user/profile](#)

HTTP Method

GET

URL

<http://127.0.0.1:8000/user/profile>

Controller

Laravel\Jetstream\Http\Controllers\Livewire\UserProfileController

Controller Method

@show

Middleware 1

web

Middleware 2

auth:sanctum

Middleware 3

verified

Try**GET** [sanctum/csrf-cookie](#)

HTTP Method

GET

URL

<http://127.0.0.1:8000/sanctum/csrf-cookie>

Controller

Laravel\Sanctum\Http\Controllers\CsrfCookieController

Controller Method

@show

Middleware 1

web

Try**GET** [livewire/preview-file/{filename}](#)

HTTP Method

GET

URL

<http://127.0.0.1:8000/livewire/preview-file/{filename}>

Controller

Livewire\Controllers\FilePreviewHandler

Controller Method

@handle

Middleware 1

web

[Try](#)**GET** [livewire/livewire.js](#)

HTTP Method

GET

URL

<http://127.0.0.1:8000/livewire/livewire.js>

Controller

Livewire\Controllers\LivewireJavaScriptAssets

Controller Method

@source

[Try](#)**GET** [livewire/livewire.js.map](#)

HTTP Method

GET

URL

<http://127.0.0.1:8000/livewire/livewire.js.map>

Controller

Livewire\Controllers\LivewireJavaScriptAssets

Controller Method

@maps

[Try](#)**GET** [api/products](#)

HTTP Method

GET

URL

<http://127.0.0.1:8000/api/products>

Controller

App\Http\Controllers\ProductController

Controller Method

@index

Middleware 1

api

[Try](#)

GET api/products/{id}

HTTP Method

URL

Controller

Controller Method

Middleware 1

GET

<http://127.0.0.1:8000/api/products/{id}>

App\Http\Controllers\ProductController

@show

api

Try

GET api/products/search/{name}

HTTP Method

URL

Controller

Controller Method

Middleware 1

GET

<http://127.0.0.1:8000/api/products/search/{name}>

App\Http\Controllers\ProductController

@search

api

Try

GET products

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

Middleware 3

GET

<http://127.0.0.1:8000/products>

App\Http\Livewire\Products

@__invoke

web

auth

admin

Try

GET transactions

HTTP Method	GET
URL	http://127.0.0.1:8000/transactions
Controller	App\Http\Livewire\Transactions
Controller Method	@__invoke
Middleware 1	web
Middleware 2	auth
Middleware 3	admin

Try**GET** restaurantandbar

HTTP Method	GET
URL	http://127.0.0.1:8000/restaurantandbar
Controller	App\Http\Livewire\UserProducts
Controller Method	@__invoke
Middleware 1	web
Middleware 2	auth:sanctum
Middleware 3	verified

Try**GET** my-bookings

HTTP Method	GET
URL	http://127.0.0.1:8000/my-bookings
Controller	App\Http\Livewire\UserTransactions
Controller Method	@__invoke
Middleware 1	web
Middleware 2	auth:sanctum
Middleware 3	verified

Try

GET user/home

HTTP Method	GET
URL	http://127.0.0.1:8000/user/home
Controller	App\Http\Controllers\HomeController
Controller Method	@UservsAdmin
Middleware 1	web
Middleware 2	verified

Try

GET users

HTTP Method	GET
URL	http://127.0.0.1:8000/users
Controller	App\Http\Livewire\Users
Controller Method	@__invoke
Middleware 1	web
Middleware 2	auth
Middleware 3	admin

Try

GET jsonproducts

HTTP Method	GET
URL	http://127.0.0.1:8000/jsonproducts
Controller	App\Http\Controllers\ProductController
Controller Method	@index
Middleware 1	web

Try

POST [_ignition/execute-solution](#)

HTTP Method	POST
URL	http://127.0.0.1:8000/_ignition/execute-solution
Controller	Facade\Ignition\Http\Controllers\ExecuteSolutionController
Controller Method	@__invoke
Middleware 1	Facade\Ignition\Http\Middleware\IgnitionEnabled
Middleware 2	Facade\Ignition\Http\Middleware\IgnitionConfigValueEnabled:enableRunnableSolutions

No.	Attributes	Type	Nullable	Bail	Rules
1	solution *required				
2	parameters	Array			

Try**POST** [_ignition/share-report](#)

HTTP Method	POST
URL	http://127.0.0.1:8000/_ignition/share-report
Controller	Facade\Ignition\Http\Controllers\ShareReportController
Controller Method	@__invoke
Middleware 1	Facade\Ignition\Http\Middleware\IgnitionEnabled
Middleware 2	Facade\Ignition\Http\Middleware\IgnitionConfigValueEnabled:enableShareButton

No.	Attributes	Type	Nullable	Bail	Rules
1	report *required				
2	tabs *required	Array			min:1
3	lineSelection				

Try**POST** [login](#)

HTTP Method	POST
-------------	-------------

URL	http://127.0.0.1:8000/login
Controller	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController
Controller Method	@store
Middleware 1	web
Middleware 2	guest:web
Middleware 3	throttle:login

No.	Attributes	Type	Nullable	Bail	Rules
1	email *required	String			
2	password *required	String			

Try**POST** [logout](#)

HTTP Method	POST
URL	http://127.0.0.1:8000/logout
Controller	Laravel\Fortify\Http\Controllers\AuthenticatedSessionController
Controller Method	@destroy
Middleware 1	web

Try**POST** [forgot-password](#)

HTTP Method	POST
URL	http://127.0.0.1:8000/forgot-password
Controller	Laravel\Fortify\Http\Controllers>PasswordResetLinkController
Controller Method	@store
Middleware 1	web
Middleware 2	guest:web

Try

POST reset-password

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

POST<http://127.0.0.1:8000/reset-password>

Laravel\Fortify\Http\Controllers\NewPasswordController

@store

web

guest:web

Try**POST** register

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

POST<http://127.0.0.1:8000/register>

Laravel\Fortify\Http\Controllers\RegisteredUserController

@store

web

guest:web

Try**POST** email/verification-notification

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

Middleware 3

POST<http://127.0.0.1:8000/email/verification-notification>

Laravel\Fortify\Http\Controllers\EmailVerificationNotificationController

@store

web

auth:web

throttle:6,1

Try

POST user/confirm-password

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

POST<http://127.0.0.1:8000/user/confirm-password>

Laravel\Fortify\Http\Controllers\ConfirmablePasswordController

@store

web

auth:web

Try**POST** two-factor-challenge

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

Middleware 3

POST<http://127.0.0.1:8000/two-factor-challenge>

Laravel\Fortify\Http\Controllers\TwoFactorAuthenticatedSessionController

@store

web

guest:web

throttle:two-factor

No.	Attributes	Type	Nullable	Bail	Rules
1	code	String	Nullable		
2	recovery_code	String	Nullable		

Try**POST** user/two-factor-authentication

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

Middleware 3

POST<http://127.0.0.1:8000/user/two-factor-authentication>

Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController

@store

web

auth:web

password.confirm

Try**POST** user/confirmed-two-factor-authentication

HTTP Method

POST

URL

<http://127.0.0.1:8000/user/confirmed-two-factor-authentication>

Controller

Laravel\Fortify\Http\Controllers\ConfirmedTwoFactorAuthenticationController

Controller Method

@store

Middleware 1

web

Middleware 2

auth:web

Middleware 3

password.confirm

Try**POST** user/two-factor-recovery-codes

HTTP Method

POST

URL

<http://127.0.0.1:8000/user/two-factor-recovery-codes>

Controller

Laravel\Fortify\Http\Controllers\RecoveryCodeController

Controller Method

@store

Middleware 1

web

Middleware 2

auth:web

Middleware 3

password.confirm

Try**POST** livewire/message/{name}

HTTP Method

POST

URL

<http://127.0.0.1:8000/livewire/message/{name}>

Controller

Livewire\Controllers\HttpConnectionHandler

Controller Method

@__invoke

Middleware 1

web

Try**POST** [livewire/upload-file](#)

HTTP Method

POST

URL

<http://127.0.0.1:8000/livewire/upload-file>

Controller

Livewire\Controllers\FileUploadHandler

Controller Method

@handle

Middleware 1

web

Try**POST** [api/login](#)

HTTP Method

POST

URL

<http://127.0.0.1:8000/api/login>

Controller

App\Http\Controllers\AuthController

Controller Method

@login

Middleware 1

api

Try**POST** [api/register](#)

HTTP Method

POST

URL

<http://127.0.0.1:8000/api/register>

Controller

App\Http\Controllers\AuthController

Controller Method

@register

Middleware 1

api

Try

POST api/products

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

POST<http://127.0.0.1:8000/api/products>

App\Http\Controllers\ProductController

@store

api

auth:sanctum

Try**POST** api/logout

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

POST<http://127.0.0.1:8000/api/logout>

App\Http\Controllers\AuthController

@logout

api

auth:sanctum

Try**PUT** user/profile-information

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

PUT<http://127.0.0.1:8000/user/profile-information>

Laravel\Fortify\Http\Controllers\ProfileInformationController

@update

web

auth:web

Try

PUT user/password

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

PUT<http://127.0.0.1:8000/user/password>

Laravel\Fortify\Http\Controllers\PasswordController

@update

web

auth:web

Try**PUT** api/products/{id}

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

PUT<http://127.0.0.1:8000/api/products/{id}>

App\Http\Controllers\ProductController

@update

api

auth:sanctum

Try**DELETE** user/two-factor-authentication

HTTP Method

URL

Controller

Controller Method

Middleware 1

Middleware 2

Middleware 3

DELETE<http://127.0.0.1:8000/user/two-factor-authentication>

Laravel\Fortify\Http\Controllers\TwoFactorAuthenticationController

@destroy

web

auth:web

password.confirm

Try

DELETE [api/products/{id}](#)

HTTP Method

DELETE

URL

<http://127.0.0.1:8000/api/products/{id}>

Controller

App\Http\Controllers\ProductController

Controller Method

@destroy

Middleware 1

api

Middleware 2

auth:sanctum

Try



ProductControllerREST

POST http://localhost:8000/api/products

```
http://localhost:8000/api/products
```

AUTHORIZATION

Bearer Token

Token

<token>

HEADERS

Accept

application/json

Content-Type

application/json

BODY raw

```
{
  "name": "Sex on the Beach",
  "category": "Cocktail",
  "description": "Hot fuzzy drink",
  "price": "2499",
}
```

Example Request

[http://localhost:8000/api/...](http://localhost:8000/api/products) ▾

```
curl --location --request POST 'http://localhost:8000/api/products' \
--header 'Accept: application/json' \
--data-urlencode 'name=Sonka' \
--data-urlencode 'slug=20' \
--data-urlencode 'description=Füstölt sonka husvétra' \
--data-urlencode 'price=100'
```

401 Unauthorized

Example Response

Body Header (8)

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

POST http://localhost:8000/api/logout

http://localhost:8000/api/logout

AUTHORIZATION

Bearer Token

Token

<token>

HEADERS

Accept

application/json

BODY urlencoded

name

Andras2

email

andras2@gmail.com

password

xyz23

password_confirmation

xyz23

Example Request

http://localhost:8000/api/... ▾

```
curl --location --request POST 'http://localhost:8000/api/logout' \  
--header 'Accept: application/json'
```

Example Response



Body Header (10)

```
{  
  "message": "You have successfully logged out"  
}
```

POST http://localhost:8000/api/login

```
http://localhost:8000/api/login
```

AUTHORIZATION

Bearer Token

Token

<token>

HEADERS

Accept

application/json

BODY urlencoded

name

Admin

email

admin@gmail.com

password

Admin111\$

password_confirmation

Admin111\$

Example Request

http://localhost:8000/api/... ▾

```
curl --location --request POST 'http://localhost:8000/api/login' \  
--header 'Accept: application/json' \  
--data-urlencode 'email=admin@gmail.com' \  
--data-urlencode 'password=Admin123'
```



Example Response

201 Created

Body Header (10)

```
{  
  "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, View More  
    "created_at": "2022-03-20T19:06:58.000000Z",  
  },
```

GET http://localhost:8000/api/products

```
http://localhost:8000/api/products
```

AUTHORIZATION

Bearer Token

Token

<token>

HEADERS

Accept

application/json

BODY urlencoded

name

Sonka

category

Food

description

Füstölt sonka husvétra

price

1999

Example Request

<http://localhost:8000/api/products>

```
curl --location --request GET 'http://localhost:8000/api/products' \
--header 'Accept: application/json'
```

Example Response

200 OK

Body Header (10)

```
[  
  {  
    "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"  
  },  
]
```

[View More](#)

GET <http://localhost:8000/api/products/25><http://localhost:8000/api/products/4>**AUTHORIZATION**

Bearer Token

Token

<token>

HEADERS**Accept**

application/json

BODY urlencoded**name**

Sonka

category

20

description

Füstölt sonka husvétra

price

00

Example Request

[http://localhost:8000/api/...](http://localhost:8000/api/products/25) ▾

```
curl --location --request GET 'http://localhost:8000/api/products/25' \
--header 'Accept: application/json'
```

Example Response

200 OK

Body Header (10)

```
{
  "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"
}
```

PUT http://localhost:8000/api/products/32<http://localhost:8000/api/products/32>**AUTHORIZATION**

Bearer Token

Token

<token>

HEADERS**Accept**

application/json

Content-Type

application/x-www-form-urlencoded

Content-Type

application/json

BODY urlencoded**name**

Honey

category

Food

description

small box of honey

price

499

Example Request

[http://localhost:8000/api/...](http://localhost:8000/api/) ▾

```
curl --location --request PUT 'http://localhost:8000/api/products/30' \
--header 'Accept: application/json' \
--data-urlencode 'name=Honey' \
--data-urlencode 'category=Food' \
--data-urlencode 'description=small box of honey' \
--data-urlencode 'price=245'
```

Example Response

401 Unauthorized

Body Header (8)

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

GET http://localhost:8000/api/products/search/chicken

```
http://localhost:8000/api/products/search/long
```

HEADERS**Accept**

application/json

BODY urlencoded**name**

Sonka

category

20

description

Füstölt sonka husvétra

price

100

Example Request

[http://localhost:8000/api/...](http://localhost:8000/api/products/search/chicken) ▾

```
curl --location --request GET 'http://localhost:8000/api/products/search/chicken' \
--header 'Accept: application/json'
```

Example Response

200 OK

Body Header (10)

```
[  
 {  
   "id": 3,  
   "name": "Chicken steak",  
   "category": "Food",  
   "description": "Countryside chicken",  
   "price": "3000.00",  
   "created_at": null,  
   "updated_at": null  
 },  
 ]
```

[View More](#)

POST <http://localhost:8000/api/register><http://localhost:8000/api/register>**HEADERS****Accept**

application/json

BODY urlencoded**name**

tester

email

user@gmail.com

password

Testing111

password_confirmation

Testing111

Example Request

[http://localhost:8000/api/...](http://localhost:8000/api/register) ▾

```
curl --location --request POST 'http://localhost:8000/api/register' \
--header 'Accept: application/json' \
--data-urlencode 'name=user' \
--data-urlencode 'email=joska@gmail.com' \
--data-urlencode 'password=Joska222' \
--data-urlencode 'password_confirmation=Joska211'
```

Example Response

422 Unprocessable Content

Body Header (10)

```
{
  "message": "The given data was invalid.",
  "errors": {
    "email": [
      "The email has already been taken."
    ],
    "password": [
      "The password confirmation does not match."
    ]
}
```

[View More](#)**DEL <http://localhost:8000/api/products/4>**<http://localhost:8000/api/products/30>**AUTHORIZATION**

Bearer Token

Token

<token>

HEADERS**Accept**

application/json

BODY urlencoded<https://documenter.getpostman.com/view/19126249/UVyxRZv1>

BODY urlencoded

name

Long Island

category

Cocktail

description

Rum, Tequila, Vodka, Coke

price

4999

Example Request

[http://localhost:8000/api/...](http://localhost:8000/api/products/28) ▾

```
curl --location --request DELETE 'http://localhost:8000/api/products/28' \
--header 'Accept: application/json' \
--data-urlencode 'name=Long Island' \
--data-urlencode 'category=Cocktail' \
--data-urlencode 'description=Rum, Tequila, Vodka, Coke' \
--data-urlencode 'price=4999'
```

Example Response

200 OK

Body Header (10)