BMSZC Neumann János Informatikai Technikum

1144. Budapest, Kerepesi út 124.

1. Vizsgaremek

EasyBook időpontfoglaló webapplikáció

06135013 Szoftverbiztonsági szakember

Benyújtás helye: Budapest Készítette:

Láng Richárd

Tartalom

[1 Bevezetés 3](#_Toc198735431)

[1.1 Feladat leírás 3](#_Toc198735432)

[1.1.1 Általános ismertetés 3](#_Toc198735433)

[2 Felhasználói dokumentáció 4](#_Toc198735434)

[2.1 A program általános specifikációja 4](#_Toc198735435)

[2.2 Rendszerkövetelmények 5](#_Toc198735436)

[2.2.1 Hardver követelmények 5](#_Toc198735437)

[2.2.2 Szoftver követelmények 6](#_Toc198735438)

[2.3 3. A program telepítése 6](#_Toc198735439)

[2.3.1 Manuális letöltés 6](#_Toc198735440)

[2.3.2 Alkalmazás futtatása 13](#_Toc198735441)

[2.4 A program használatának a részletes leírása 13](#_Toc198735442)

[2.5 GYÍK, FAQ (Gyakran Ismételt Kérdések) 17](#_Toc198735443)

[3 Fejlesztői dokumentáció 19](#_Toc198735444)

[3.1 Az alkalmazott fejlesztői eszközök 19](#_Toc198735445)

[3.1.1 Adatmodell leírása 22](#_Toc198735446)

[3.2 Részletes feladatspecifikáció, algoritmusok 25](#_Toc198735447)

[3.3 Backend logika 25](#_Toc198735448)

[3.3.1 Foglalás létrehozása (BookingController@store) 28](#_Toc198735449)

[3.4 Frontend oldalak 36](#_Toc198735450)

[3.5 Email küldés mükődése 37](#_Toc198735451)

[3.6 Logolás 39](#_Toc198735452)

[3.7 Tesztelési dokumentáció 40](#_Toc198735453)

[4 Összegzés 58](#_Toc198735454)

[Továbbfejlesztési lehetőségek 58](#_Toc198735455)

[5 Felhasznált irodalom 59](#_Toc198735456)

# Bevezetés

## Feladat leírás

### Általános ismertetés

Az én kis projektem egy időpontfoglalásra tervezett webes alkalmazás. Azért választottam ezt a témát, mert szerettem volna egy egyedibb, gyakorlatias célú webalkalmazást készíteni. Olyan típusú időpontfoglaló rendszert álmodtam meg, amilyennel még nem találkoztam korábban. Sok kisebb, helyi vállalkozásnál ma is telefonon vagy e-mailben kell időpontot egyeztetni, ami munkaidőben zavaró lehet számukra. Az általam tervezett rendszer ezt a folyamatot automatizálja, így csökkenti a megszakítások számát.

A weboldalon keresztül egyszerűen lehet időpontot foglalni, amelyet a szolgáltató online felületen keresztül tud jóváhagyni. A rendszer nem egy konkrét vállalkozásra van szabva: bárki regisztrálhat, és a felhasználók könnyedén megtalálhatják a számukra megfelelő szolgáltatót. Ugyanakkor a rendszer könnyen testreszabható akár egyetlen vállalkozás igényei szerint is.

Úgy érzem, ez a projekt kiváló lehetőség volt számomra arra, hogy új technológiákat ismerjek meg, és gyakorlati tapasztalatot szerezzek. Emellett jó alapot nyújtott ahhoz is, hogy bővítsem a GitHub portfóliómat.

# Felhasználói dokumentáció

## A program általános specifikációja

Az EasyBook egy időpontfoglaló webes alkalmazás, amely a szolgáltatók és vendégek számára nyújt kényelmes megoldást a foglalások kezelésére. Segítségével könnyedén lefoglalhatjuk a kívánt időpontot például kedvenc fodrászunkhoz, míg a szolgáltatók könnyedén nyomon követhetik és kezelhetik ügyfeleik foglalásait.

**Funkciók és jellemzők**

Általános jellemzők

* Modern, reszponzív felhasználói felület, mobil- és asztali eszközökre optimalizálva.
* Biztonságos bejelentkezés és regisztráció
  + Jelszóerősség figyelő, 2 faktoros hitelesítés és Google regisztráció

Szolgáltató oldali funkciók

* Profil létrehozása és testreszabásának lehetősége
  + Leírás, elérhetőség megadása
  + Munkaidő megadása, szerkesztése (teljes testreszabhatóság)
* Időpontfoglalások kezelése
  + Naptár nézet a foglalásokhoz
  + Időpontok elfogadás elutasítása
* Visszaszámláló a legközelebbi megerősített időponthoz a dashboardon
* Jelzés a dashboardon, ha van meg nem erősített foglalás
* Google Calendar integráció
  + Automatikus eseményrögzítés a naptárba (engedélyezés esetén)

Ügyfél oldali funkciók

* Szolgáltatók listázása, keresés lehetőséggel
* Szolgáltató adatlapának megtekintése
  + Név, cím, leírás, telefonszám
* Online időpontfoglalás a kiválasztott szolgáltatónál
  + Elérhető időpontok nézet naptárral
* Időpontok listázása, naptárral
* Visszaszámláló a legközelebbi megerősített időponthoz a dashboardon
* Email értesítés foglaláskor, illetve megerősítéskor
* Google Calendar integráció
  + Automatikus eseményrögzítés a naptárba (engedélyezés esetén)

## Fontos információ

A Google bejelentkezés jelenleg nem lehetséges tesztelni, mivel az OAuth kliens csak production környezeteben müködik és a Google Cloud Console beállításaiban kizárólag a saját Google fiókjaim vannak hozzáadva mint teszterként.

## Rendszerkövetelmények

### Hardver követelmények

Szerver oldalon:

Minimum:

* CPU: 2vCPU
* RAM: 4GB
* Tárhely: 20 GB (ssd ajánlott)

Ajánlott:

* CPU: 4vCPU
* RAM: 8 GB
* Tárhely: 50 GB (ssd ajánlott)

Kliens oldalon:

Minimum:

* CPU: Kétszálú processzor (pl. Intel Core i3 vagy AMD Athlon)
* RAM: 4 GB
* Tárhely: nincs különösebb igény (csak böngésző cache)
* Képernyőfelbontás: Legalább 1280 x 720
* Internetkapcsolat: stabil kapcsolat, legalább 5Mbps letöltési sebességgel

Ajánlott:

* CPU: Intel Core i5 vagy újabb
* RAM: 8 GB
* Képernyőfelbontás: 1920 x1080 (Full HD)

### Szoftver követelmények

Szerver oldalon:

* OS: Ajánlott (Ubuntu LTS, Debian, CentOs, Amazon Linux 2)
* Webszerver: Apache
* PHP: 8.2+
* Composer: legfrisebb stabil verzió
* Laravel: 10.x
* Adatbázis: MYSQL 8.x

Kliens oldaon:

* OS: Windows 10/11 vagy macOs 11 vagy újabb, Linux
* Böngésző: legfrisebb Opera, Chrome, Brave
* Böngésző technológiák:
  + Javascript - engedélyezve kell legyen
  + Cookies - engedélyezve kell legyen
  + SessionStorage - engedélyezve kell legyen

## 3. A program telepítése

### Docker letöltés

Előfeltételek:

* Git
* Docker

1. Projekt klónozása
   1. git clone <https://github.com/Yoclar/EasyBook.git>
   2. cd EasyBook
2. Környzezeti változók
   1. cp .env.example .env
   2. Az alábbi .env fájl részletet kell átmásolni az a projekten belül a .env-be

APP\_NAME=EasyBook

APP\_ENV=local

APP\_KEY=base64:JhYFTthEjomm/WAgyAs9V3l4DgftBIptzr59+KyUndE=

APP\_DEBUG=true

APP\_TIMEZONE=UTC

APP\_URL=http://localhost

LOG\_CHANNEL=daily

LOG\_STACK=single

LOG\_DEPRECATIONS\_CHANNEL=null

LOG\_LEVEL=debug

LOG\_DAILY\_DAYS=14

DB\_CONNECTION=mysql

DB\_HOST=mysql

DB\_PORT=3306

DB\_DATABASE=easybook

DB\_USERNAME=root

DB\_PASSWORD=rootpassword

MAIL\_MAILER=smtp

MAIL\_SCHEME=null

MAIL\_HOST=smtp.gmail.com

MAIL\_PORT=587

MAIL\_USERNAME=laravelmybeloved@gmail.com

MAIL\_PASSWORD=gcvmozerclufuktp

MAIL\_FROM\_ADDRESS="laravelmybeloved@gmail.com"

MAIL\_FROM\_NAME="${APP\_NAME}"

GOOGLE\_CLIENT\_ID=test-google-client-id

GOOGLE\_CLIENT\_SECRET=test-google-client-sercret

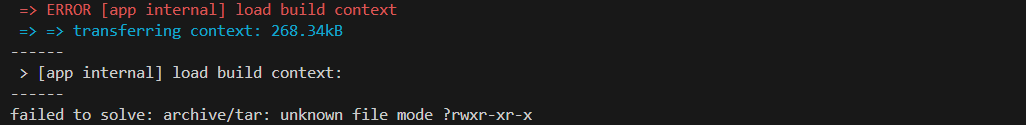
GOOGLE\_REDIRECT\_URI=http://localhost:8000/callback

(A Google OAuth2 bejelentkezéshez fel kell setupolni egy projektet a [https://console.cloud.google.com/](https://console.cloud.google.com/" \o "https://console.cloud.google.com/" \t "https://discord.com/channels/@me/_blank) oldalon onnnan kell a projektnek a Client ID-ja amit az APIs & Services->Credentials->Client IDs->Client részről tudunk megszerezni itt található a client secrets is)

Ha nem szeretnék az én teszt emailemet használni. Az email testreszabható: az alábbi lépésekkel (a weboldal által szemléltetve):

<https://dev.to/aoamusat/sending-email-in-laravel-app-using-gmail-account-iaj>

1. Docker
   1. docker-compose up --build -d
   2. Első indítás után
      1. docker-compose exec app php artisan key:generate **!** # ha az .env fileban nincs benne valamiért
      2. docker-compose exec app php artisan migrate
      3. docker-compose exec app php artisan storage:link
      4. docker-compose exec app php artisan db:seed # opcionális ha factory adatokat is szeretnénk
   3. , Alkalmazás elérése
      1. http://localhost:8000

Esetleges hibák: 

Ha esetleg hasonló hibát kapunk töröljük a szimbolikus linket

* rm public/storage

Vagy

* Remove-Item -Recurse -Force .\public\storage

### Manuális letöltés

Az alklmazás müködésére szükséges szoftverek.

1. Böngésző
2. Visual Studio Code
3. Xampp
4. Composer

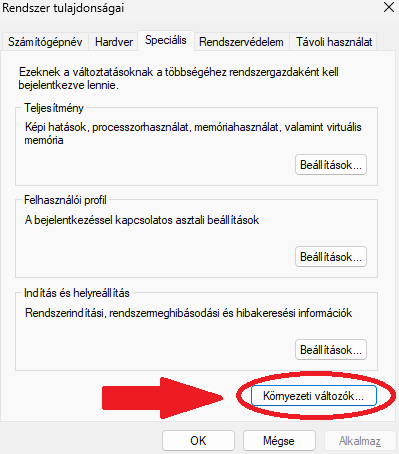
Alkalmazás telepítése

1. Xampp letöltése és telepítése: A xampp magával hozza a php fájljait ezért nem kell külön letölteni a php-t. Forrás: <https://www.apachefriends.org/hu/index.html>
2. Composer letöltése és telepítése. Forrás: <https://getcomposer.org>
3. Git letöltése: Tutorial: <https://www.youtube.com/watch?v=4xqVv2lTo40>
4. Parancssorban tesztelni, hogy a php és a node js verzióit a windows megtalálja és felismeri-e. php -v

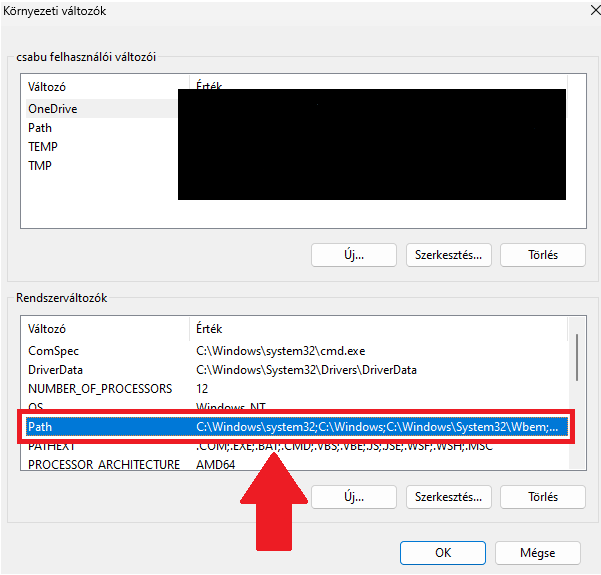
Amennyiben, nincs telepítve, sérültek a fájlok a telepítés közben vagy nincs helyes elérésiút állítva.

Ezt a következőképpen kell csinálni:

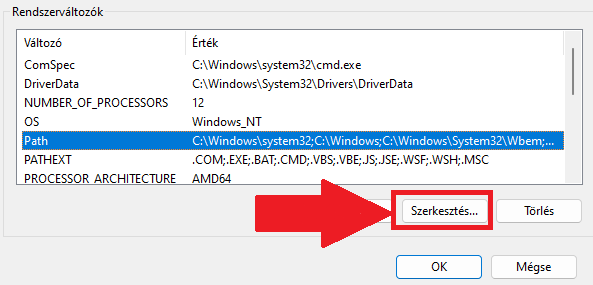
A számítógép keresőjébe írjuk be, hogy Környezeti változók.



Válassza ki a “Rendszerváltozók” menüből a “Path” menüpontot.

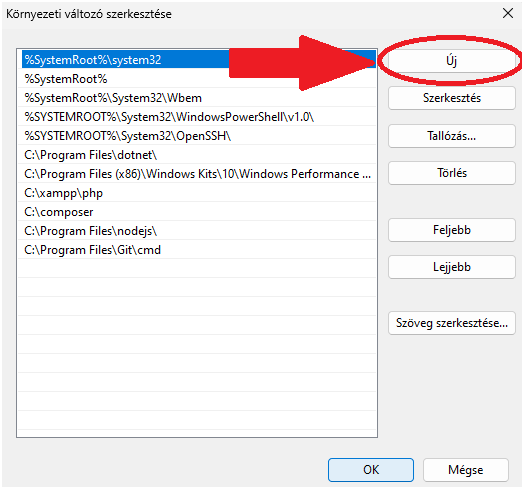


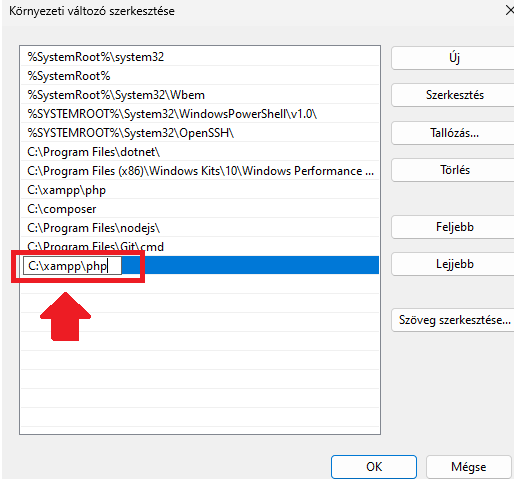
Nyomjon rá a “szerkesztés” gombot.

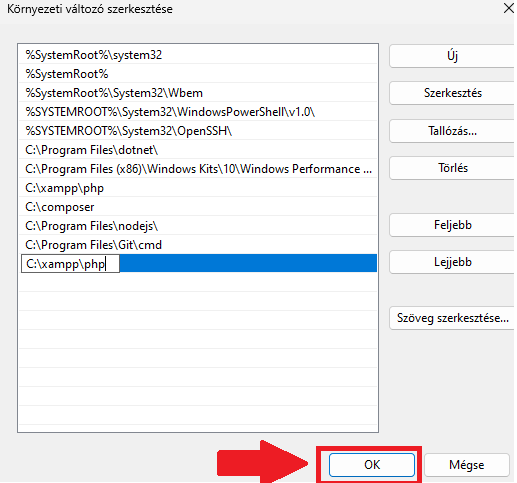


Hozzon létre új elérési utat az “Új” gombal, írja be a xampp elérési útját.

Xampp esetében: C:/xampp/php (a telepítés helye alapján eltérő lehet).

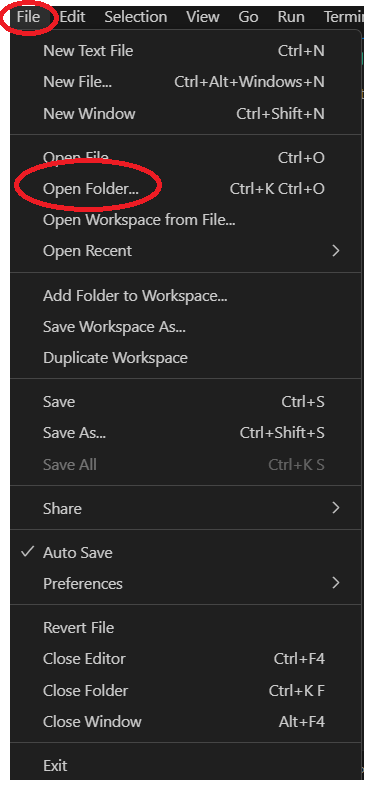






1. Visual studio code letöltése majd telepítése. Forrás: <https://code.visualstudio.com>
2. A projekt letöltése és telepítése github-ról.

* Nyissa meg a “Visual Studio Code” nevű alkalmazást.
* A felső menüsorban nyomjon rá a “File” gombra, majd nyomjon rá az ”Open folder” gombra, válassza ki a “xammp” mappán belüli “htdocs” nevű mappát. (Ez a gép -> Helyilemez (C:, vagy ahova a xammp telepítve lett) -> xammpp -> htdocs)



Kép beszúrása...

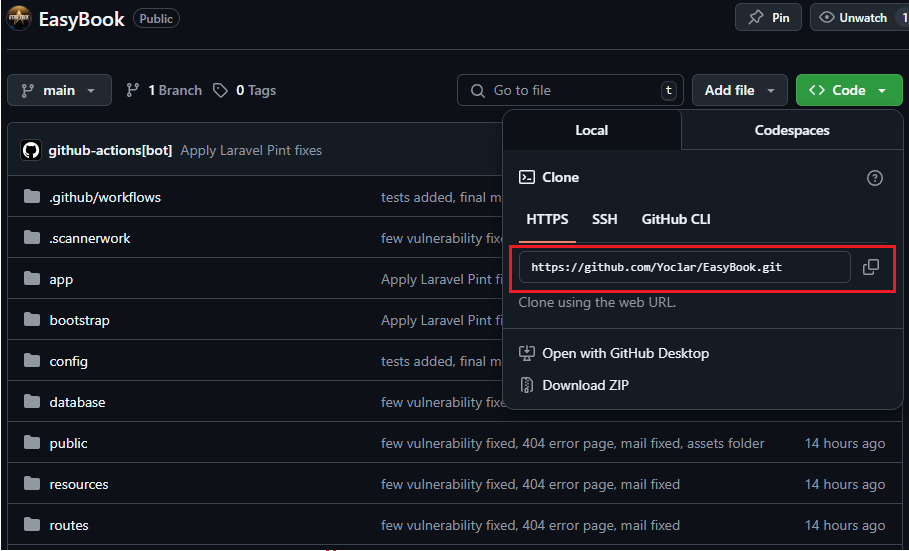
* Nyisson egy új terminált a “Ctrl+Shift+ö” billentyű kombinációval vagy a felső menüsorvan lévő “Terminal” gomra kattintva majd a “New terminal” gombra nyomva.

1. Futtasa a következő parancsot:
2. Hozzon létre új mappát “EasyBook” néven, a következő helyen:

Ez a gép -> Helyilemez (C:, vagy ahova a xammp telepítve lett) -> xammpp -> htdocs

1. Nyissa meg ezt a mappát a Visual Studio Codeban
2. Futtassa a következő parancsokat a Visual Studio Code termináljában:

* git clone (url)



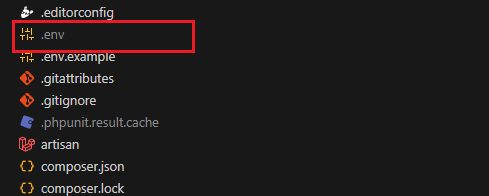
(Ezt az URL-t a képen láthatja)

(Amennyiben az URL-t nem találja, a GitHub repository url-je a következő:

<https://github.com/Yoclar/EasyBook.git>

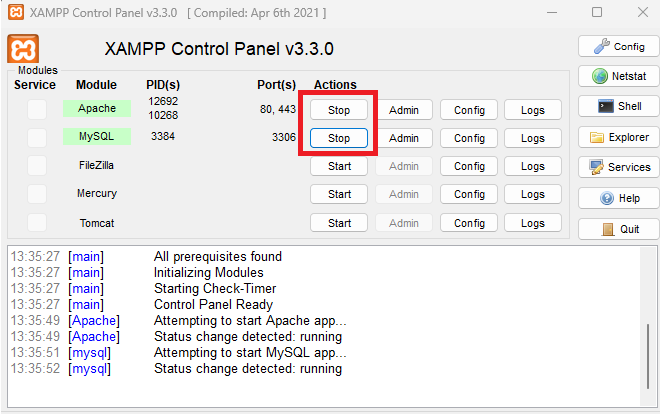
* Composer intall

1. Hozzon létre új fájlt “.env” néven.



1. A fentebb megadott .env fájl adatait másolja a meglévő .env fájlba, illetve a DB\_HOST változót mysql-ről 127.0.0.1-re kell változtatni.
2. Hozzon létre adatbázist a következő néven: easybook néven

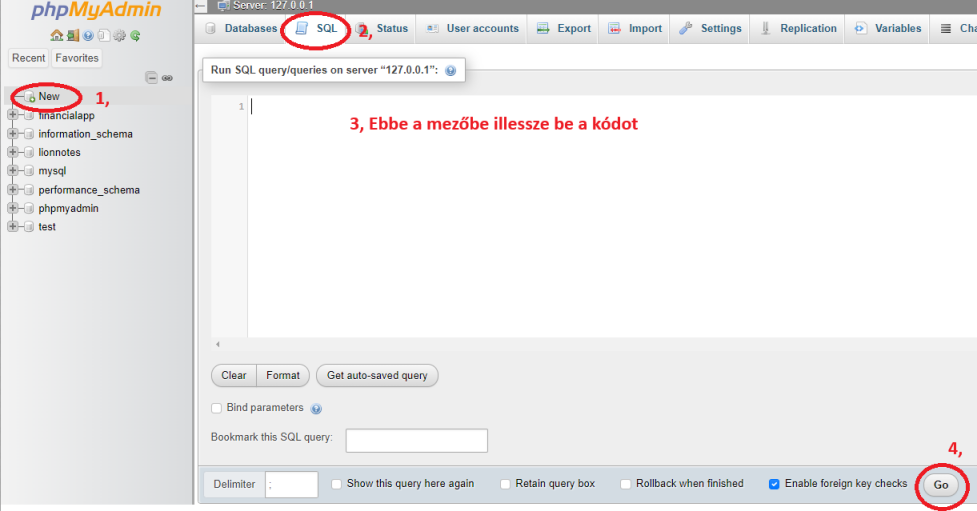
* Indítsa el a “Xammp” alkalmazást.
* Nyomjon rá a “Start” gombra a(z) “Apache” és “MySQL” module mellett.



* Nyomjon rá az “Admin” gombra a MySQL modul mellett
* Kattintson a “new” gombra majd az “SQL” gombra.
* Írja be az alábbi kódot:

CREATE DATABASE EasyBook CHARACTER SET utf8 COLLATE utf8\_hungarian\_ci;

* Ezt követően nyomja meg a “Go” gombot.



* A visual studio code termináljában futtassa le a következő parancsot, hogy az adatbázis fel legyen töltve: php artisan migrate:fresh –seed, illetve, ha így nem müködne, akkor először php artisan migrate:fresh, és utána a php artisan migrate:fresh –seed parancsot.

(Erre azért van szügség mert a Laravel adatbázis feltöltését használjuk)

1. Futtassa a következő kódokat a Visual Studio Code-ban

* php artisan migrate:fresh
* php artisan migrate:fresh --seed

1. Mappák létrehozása a képeknek, iconoknak és videóknak.
2. Link létrehozása a storage mappához, hogy a tárolt fájlokat elérjük, a következő parancsal (a visual studio code termináljában):

php artisan storage:link

### Alkalmazás futtatása

(Bizonyos parancsok és funkciók működését akadályozhatja VPN illetve vírusírtó)

Futtassa az adatbázist.

1. A Visual Studio Code alkalmazásban a projekt mappájának megnyitása után, nyisson két terminált.
2. Futtassa a köetkező parancsokat eltérő terminálokban:

* php artisan serve

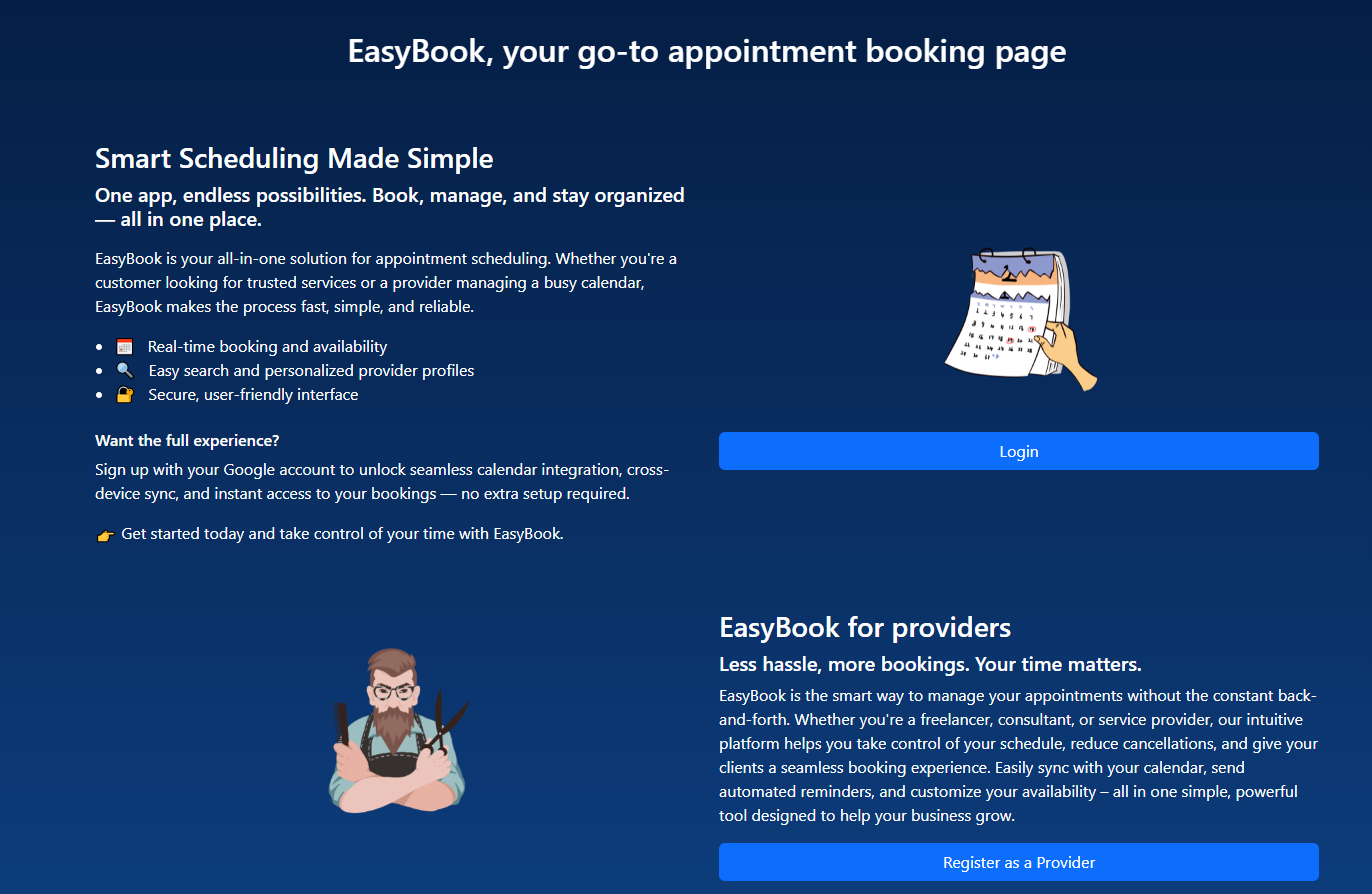
1. Nyissa meg a termináljában megjelenő linket vagy írja be egy böngészőbe hogy localhost:8000

A bejelentkezéshez az adatbázisban szereplő bármely tesztként létrehozott felhasználót használhatja, illetve regisztrálhat új felhasználót.

## A program használatának a részletes leírása

Program leírása

Az odal betöltését követően a fóoldalon (landing pagen) találjuk magunkat. Itt található a weboldal bemutatása,általánosan, szolgáltatóknak, illetve ügyfeleknek egyaránt. Ezen feül egy gyakori kérdések szekció egy hogyan foglalj időpontot (how to an appointment), illetve egy contact us szekció, ahol lehetőséget nyújtunk a felhasználóknak bármilyen kérdés esetén kapcsolatba tudnak lépni a fejleszővel.



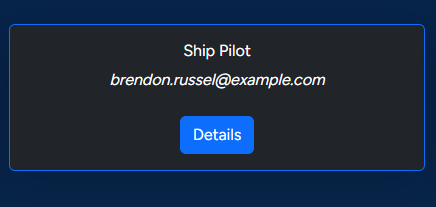
Ezután lehetőségünk lesz ügyfélként vagy szolgáltatóként regisztrálni a weboldalra. Mind a kettőhöz egy külön rész, illetve külön gombok találhatók.

A regisztrációs gombra történő kattintás után megjelenik a regisztráció aloldal. Itt a felhasználó eldöntheti, hogy manuálisan szeretne regisztrálni, vagy szeretne a Google fiókjával belépni az alkalmazásba. Ha a Google fiók mellett döntünk, akkor a Google engedélyt kér, mivel az alkalmazás képes az időpontot a Google Calendarba menteni, de erre csak akkor kerül sor, ha ezt a felhasználó az alkalmazáson belül engedélyezi. Belépés vagy regisztrációt követően a dashboardon találjuk magunkat.

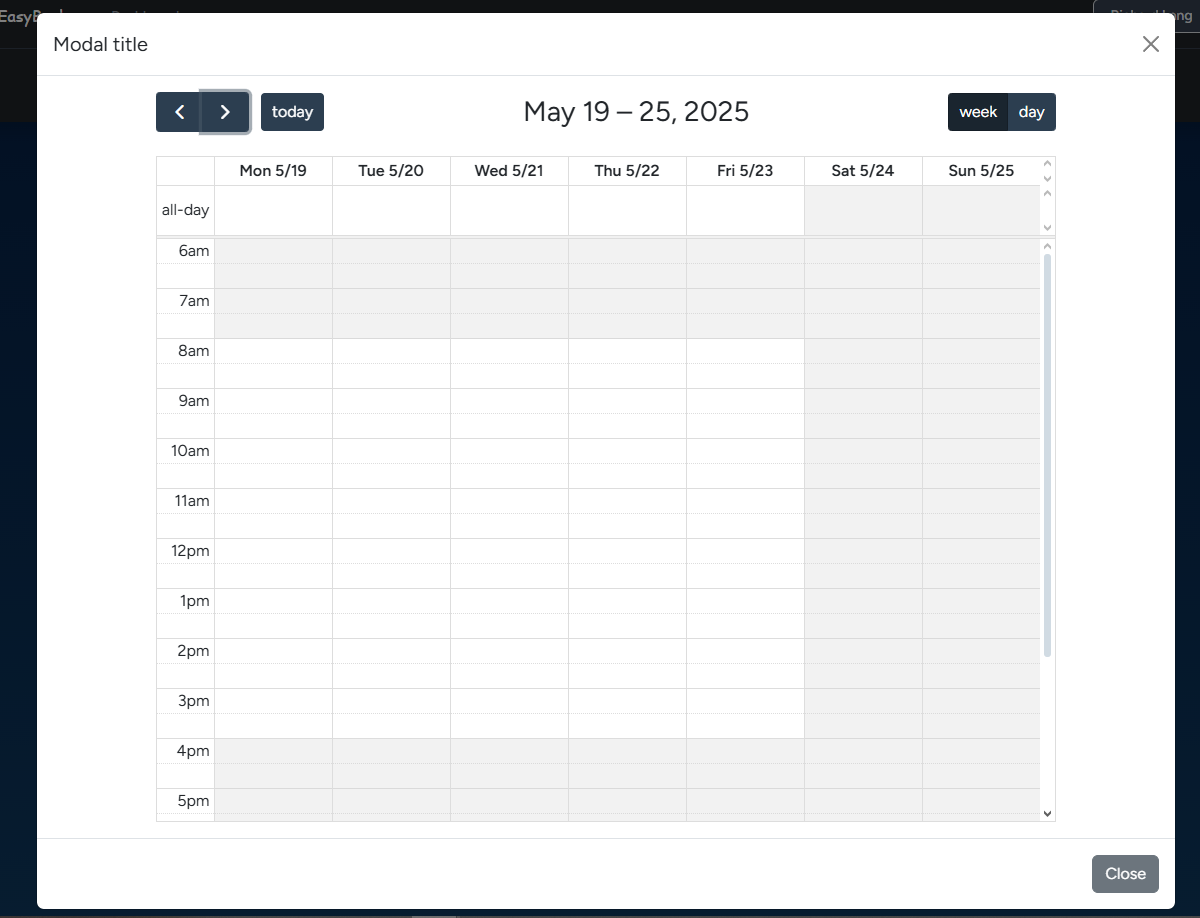
A dashboardon található egy visszaszámláló a következő megerősített időpontig, illetve az aloldal értesíti a szolgáltatót, ha még van meg nem erősített időpontja, ami megerősítésre vár.

Itt elérhetőek lesznek számunkra a profil, az (időpontjaim) my appointments, illetve, ha ügyfél regisztráció történt, akkor a szolgáltatók (providers), ahol az összes szolgáltató elérhetó, ahol foglalást tudunk kezdeményezni.

**Foglalás menete**: Foglalást úgy tudunk kezdeményezni, hogy a providers oldalon a kiválasztott szolgáltatónál ráklikkelünk a details gombra, ezt követően az adott szolgáltató összes megadott információja elérhetővé válik számunkra.



Ezt követően a Book now gombra kell kattintsunk és előugrik egy naptár (calendar modal), ahol láthatjuk az összes elérhető információt az adott szolgáltatóhoz.



Itt kizárólag a világos cellákra tudunk kattintani, mivel ezek a szabad időpontok, ezen felül a világos cellák tartoznak bele a szolgáltatónk munkaidejébe. Majd ha kiválasztottunk egy időpontot, akkor előugrik egy megerősítő rész, ahol az adatokat kell megadni. Itt be kell írjuk milyen szolgáltatást veszünk igénybe és az időpontot, ha esetleg rossz cellára klikkeltünk volna, vagy módosítanánk rajta. Ezt követően egy emailt kapunk, hogy sikeresen foglaltunk időpontot.

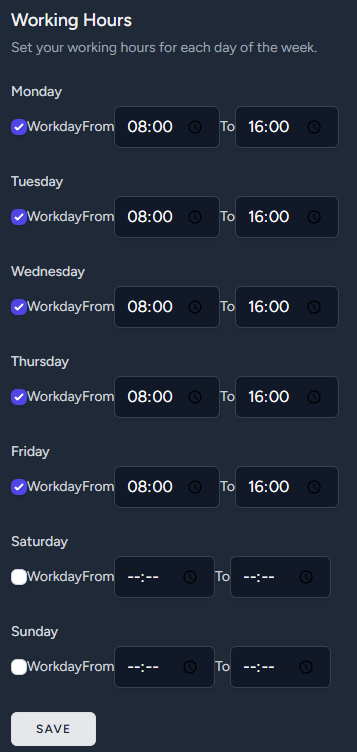
**Figyelem**: A rendszer csak akkor hagyja jóvá az időpontot, ha nem ütközik senkinek a foglalásával az adott időpont ÉS nekünk nincs már egy foglalásunk abban az időintervellumban.

**Foglalás megerősítése**: Hogyha egy ügyfél foglalást kezdeményez egy szolgáltatónál, akkor az függőben státuszt (pending) kap ameddig a szolgáltató meg nem erősíti azt. Ezt az időpontjaim (My appointments) részen lehet megtenni. Itt megtalálhatóak a még el nem következett időpontok státusszal és naptárral. Ahhoz, hogy ezt jóváhagyjuk a megerősítés (confirm) gombra kell kattintsunk. A megerősítés befejeztével az ügyfél emailt kap a megerősítésről, illetve ha be van kapcsolva akkor a Google Naptárjukba (Calendarjukba) is mentődik az időpont.

**Profil oldal**:

A profil oldalon lehetőség van személyes információkat cserélni, illetve be- vagy átállítani.

Ha szolgáltató ranggal rendelkezel akkor több beállítás áll rendelkezésedre. A 2 faktoros hitelesítés, profil információ csere, és a fiók törlése elérhető mindenki számára. A Google Calendar preferencia csak a Google fiókkal autentikált felhasználók számára érhető el. Ezen felül a szolgáltatók testre tudják szabni a cégük vagy székhelyük információit. Erre szükség is van, hogyha Google fiókkal jelentkezel be, mivel a cégnevet automatikus generáljuk (szóval azt nagyon érdemes átnevezni), és a többi információ viszont nem érhető el. Ezen kívül a szolgáltató be tudja állítani a munkaidejét és így csak a megadott munkaidőre lehet foglalni.



## GYÍK, FAQ (Gyakran Ismételt Kérdések)

* Szükséges regisztrálni ahhoz, hogy időpontot foglalhassunk?
  + Igen, jelenleg csak regisztrált ügyfelek foglalhatnak időpontot.
* Módosíthatom/törölhetem az időpontomat?
  + Sajnos ez az opció még fejlesztés alatt áll, Kérjük, legyen türelmes hamarosan elérhető lesz.
* Hogyan müködik a Google Calendar integráció?
  + Hogyha Google fiókkal vagy regisztrálva, akkor be tudod kapcsolni ezt az opciót a profil (profile) aloldalon. Ha a szolgáltatód jóváhagyja az időpontodat, akkor mentésre kerül a naptáradba.
* Valami nem müködik, mit tudok tenni?
  + Kérjük használja a lépjen kapcsolatba velünk (Contact us) szekciót a főoldalon és írjon egy emailt, mi ránézünk amilyen gyorsan csak lehet.

# Fejlesztői dokumentáció

## Az alkalmazott fejlesztői eszközök

Az alkalmazás Laravelben készült, ami egy php alapú keretrendszer.A Larvel nagyon megkönnyíti a webfejlesztést.

**Fejlesztői környezet**:

* IDE: Visual studio code

**Programozási nyelvek**:

* PHP, Javascript

**Operációs rendszer**:

* Windows 10

**Konténerizációs eszköz**:

* Docker

Könyvtárak és keretrendszerek:

**Backend keretrendszer**:

* Laravel 11+

**CSS keretrendszerek**:

* Tailwind CSS, Bootstrap

Harmadik féltől származó csomagok (packegek): FullCalendar, intl-tel-input (phone masking, Laravel fortify (2fa), Laravel socialite, Laravel UI, Google apiclient, Jeybin toastr, Ramsey uuid

**Tesztelés**: Unit és Feature tesztek (PHPunit)

**Kódbiztonság és minőségelemzé**s:

* Owasp zap
* SonarQube

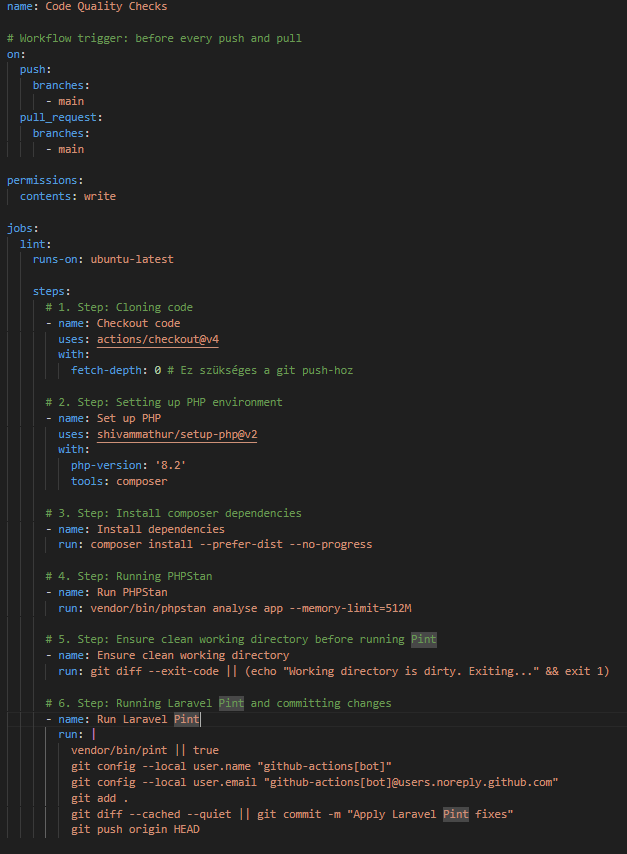
**Terhelhetőség:**

* Locust

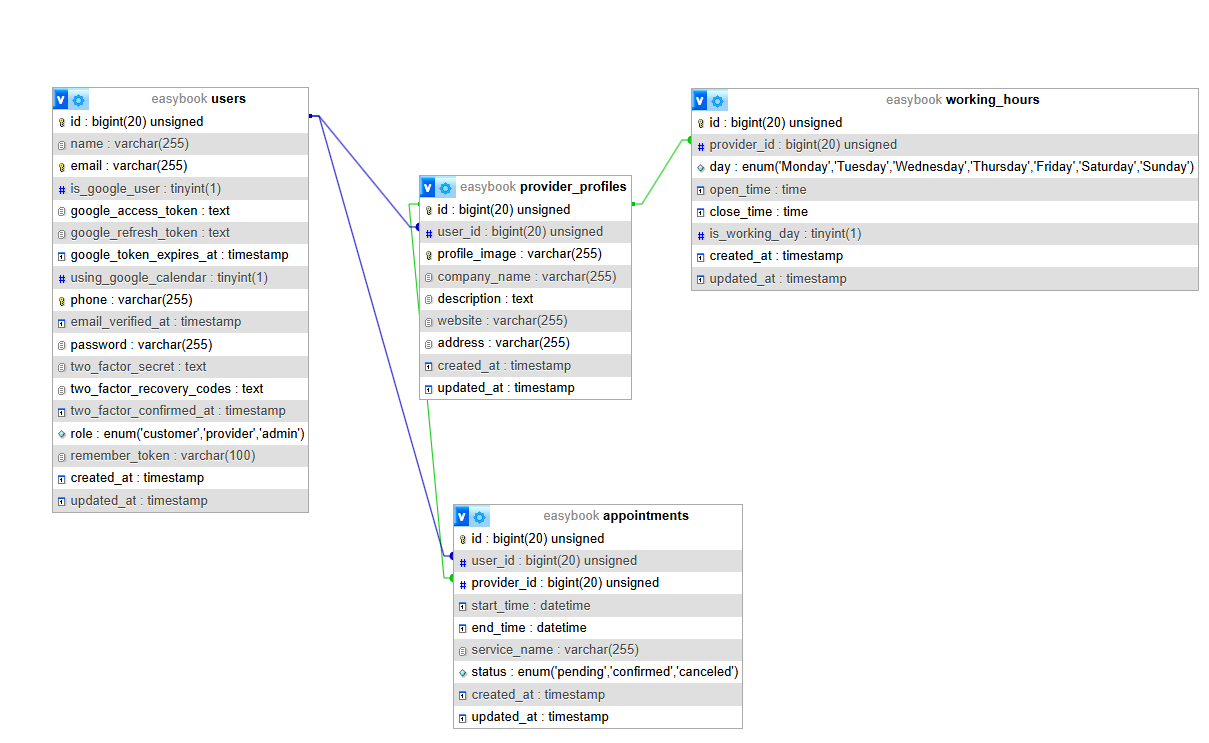
CI/CD eszközök:

* CI pipeline Github Workflow segítségével
  + PHPStan statikus kódelemzés a bugok megtalálására
  + Laravel Pint PHP kód automatikus formázása

Github Workflow:



### Adatmodell leírása



A projektemben MYSQL adatbázist használtam. Ezt a 4 táblát hozza létre az alkalmazásom induláskor.

Az adatbázis tábláit a Laravel által létrehozott database\migration mappában található fájlok hozzák létre, amiket a seedersben lévő fájlok töltenek fel adatokkal, a véletlenszerű adatokhoz a seederek a factories mappa tartalmát veszik igénybe.   
A factoriesben olyan fájlok vannak, amik egy táblában lévő oszlopoknak a követelményeinek megfelelően képes adatokat létrehozni, amiket az adott seeder feltölt.  
Egy táblához egy seeder és egy factory tartozik.

**User tábla** (model)

A User tábla tartalmaz minden hasznos információt a felhasználóról, beleértve az autentikációhoz szükséges adatokat és a személyes adatokat. Ilyenek például a név, email cím, jelszó (hashelve), a google bejelentkezéshez szükséges adatok, a 2faktoros hitelesítés adatai.

**ProviderProfile tábla** (model)

A ProviderProfile tábla egy kiegészítő táblája a user táblának. Akkor van rá szükség, ha valaki szolgáltatóként regisztrál. Ez a tábla tartalmazza az összes infórmációt a szolgáltatóról magáról. A táblához tartozik például a leírás, szolgáltató neve, illetve weboldala és címe, ha van és megadták.

**WorkingHour tábla** (model)

A WorkingHour (munkaidő) tábla felelős a szolgáltatók munkaidejéről. Ez automatikusan létrehozódik ha valaki szolgáltatóként regisztrál és beállít egy alapértelmezett (default) munkaidőt mindenkinek aki regisztrál. Ezt később teljesen egyénileg lehet kezelni, módosítani.

**Appointment tábla** (model)

Az appointment (időpontok) tábla az időpontokat tartalmazza. Ha egy ügyfél foglal egy időpontot, akkor az ebbe a táblába kerül be. Itt kötődik össze az ügyfél és a szolgáltató. A tábla tartalmaz egy user id-t és egy provider id-t . Ezen felül a foglalás nevét, időpontját és státuszát.

A projektemben a táblákat a Laravel a migrációkon keresztül foreignId-vel (idegenkulccsal) köti össze.

Ezen felül a kapcsolatokat a Laravel relationök által tudja kezelni. Ha meg szeretném tudni, hogy egy felhasználónak be van-e kapcsolva a Google Calendar beállítás, akkor azt nagyon könnyen megthetem. Például így:

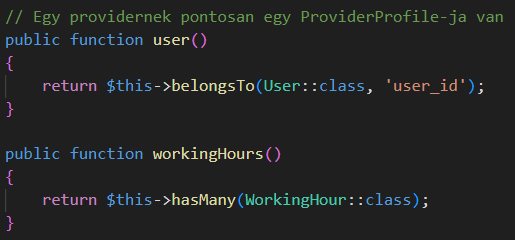


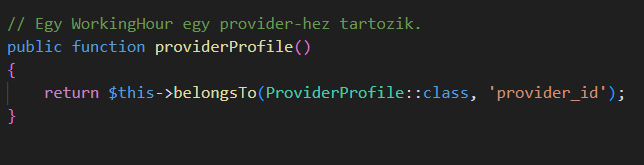
Itt vannak a kapcsolatok, amik ezt lehetővé teszik:

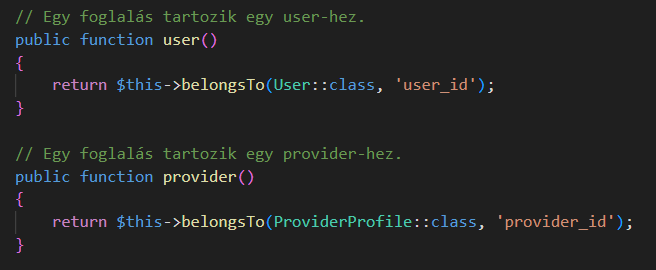
**User model**:



**ProviderProfile model**:



**WorkingHour model**:

**Appointment model**:  


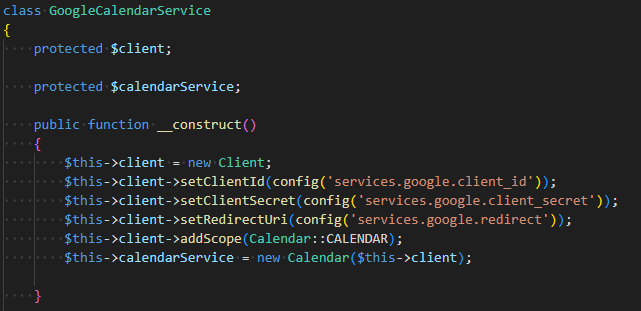
## Részletes feladatspecifikáció, algoritmusok

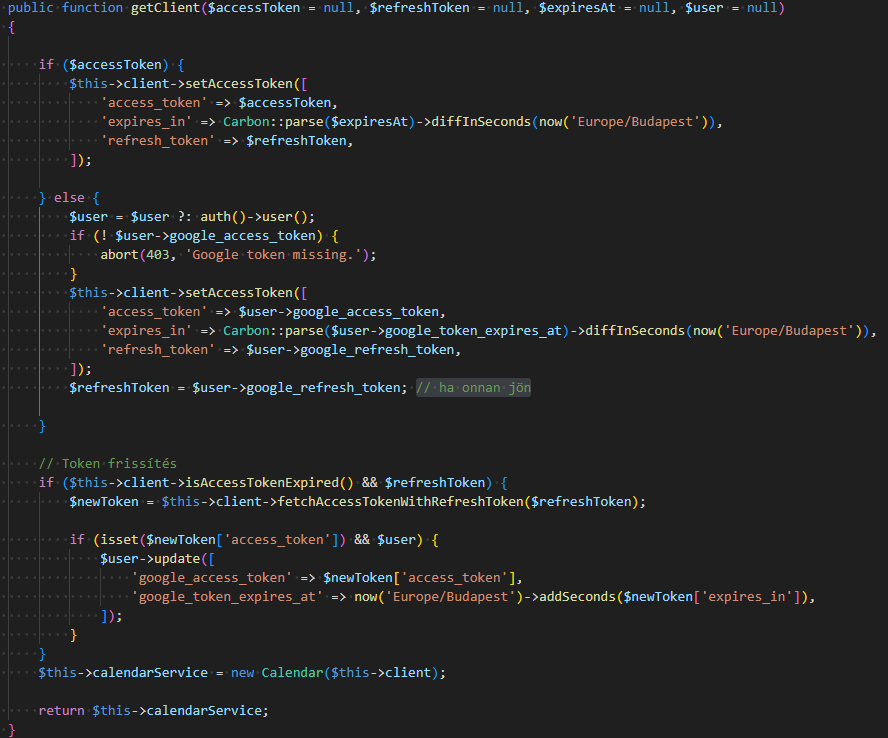
## Backend logika

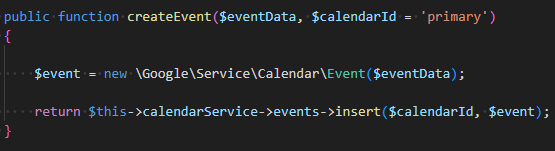
**GoogleCalendarService.php**

Ez az osztály felelős a Google Calendar API kezeléséért és integrációjáért a Laravel alkalmazásban. Biztosítja a hitelesített kliens létrehozását, a hozzáférési tokenek kezelését és az események létrehozását a Google Naptárban.

* **Kliens inicializálása**   
  Beállítja a Google API klienset a konfigurációs fájlban megadott client\_id, client\_secret és redirect\_uri értékekkel, valamint hozzáadja a naptárhoz szükséges jogosultságot (Calendar::CALENDAR).
* **Kliens visszaadása tokenekkel (**getClient**)**  
  Ez a metódus beállítja a klienshez a hozzáférési tokent és frissítő tokent, akár paraméterként, akár a jelenlegi felhasználó adatából. Ha a hozzáférési token lejárt, automatikusan megpróbálja frissíteni a tokent a refresh token segítségével, és ha sikeres, frissíti a felhasználó adatbázisában tárolt tokeneket.
* **Esemény létrehozása (**createEvent**)**  
  Paraméterként kap egy esemény adatokat tartalmazó tömböt ($eventData), majd ezt egy Google Calendar eseménnyé alakítva beszúrja a megadott naptárba (alapértelmezetten a primary naptárba).

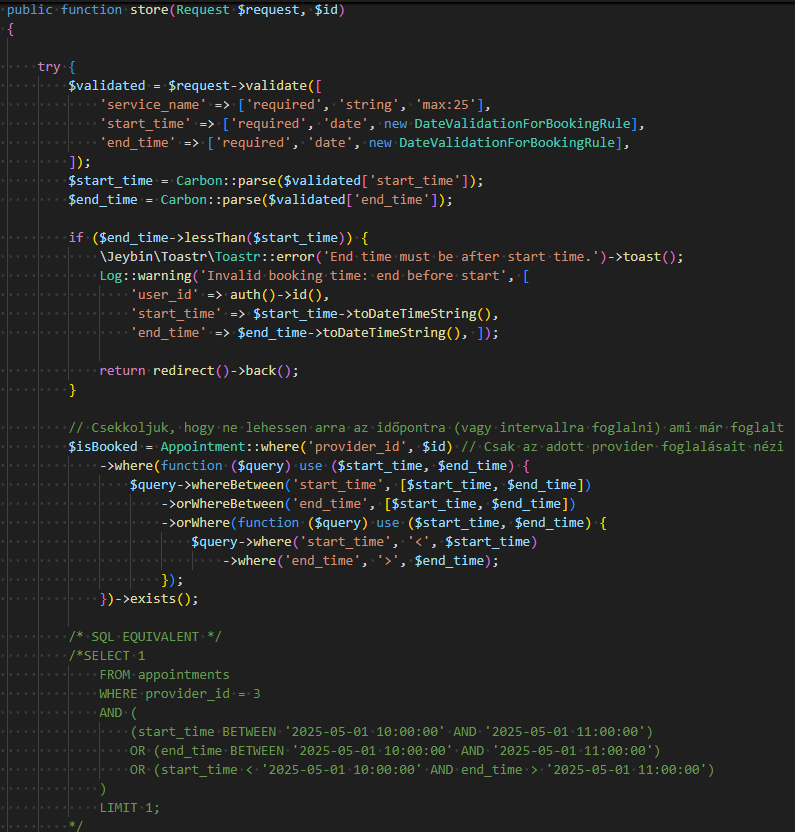




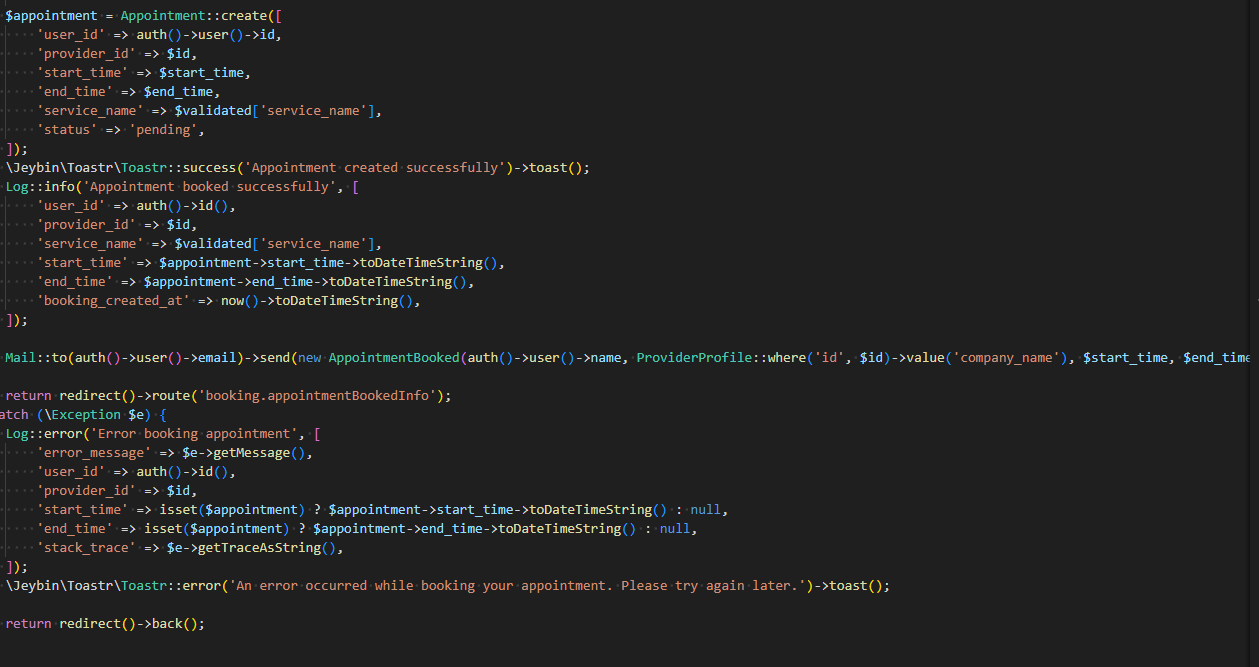


**BookingController.php**

A BookingController felelős a foglalásokért, beleértve az összes validáiót, illetve ez a controller küldi meg az adatokat a fullcalendarnak.







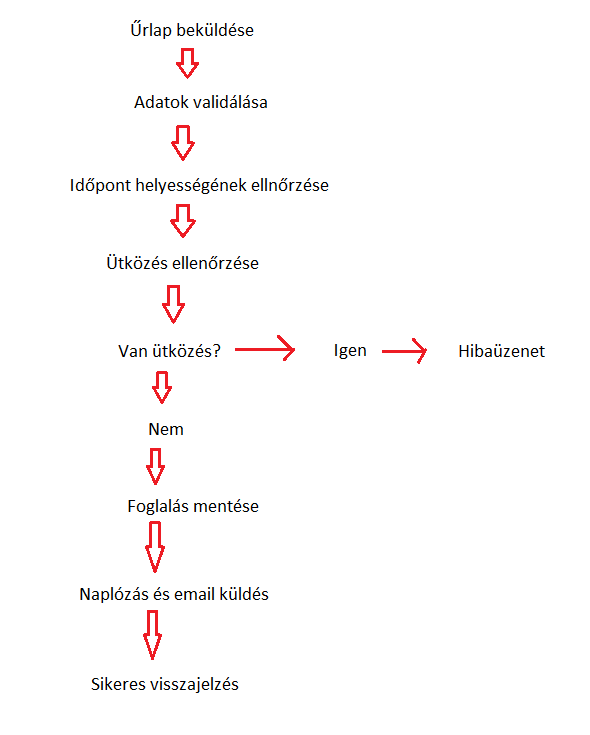
### ****Foglalás létrehozása (BookingController@store)****

Ez a metódus felelős egy új időpontfoglalás rögzítéséért. A következő lépésekben történik a folyamat:

1. **Bemeneti adatok fogadása** – A felhasználó megadja a szolgáltatót, időpontot és a szolgáltatás nevét.
2. **Validálás** – Ellenőrizzük a kötelező mezőket, az időpont formátumát, és hogy az a jövőre vonatkozzon.
3. **Ütközések ellenőrzése** – Megvizsgáljuk, hogy az adott időpont nem foglalt-e más ügyfélnél vagy szolgáltatónál, illetve az ügyfélnek nincs-e már másik foglalása ugyanakkor.
4. **Adatok mentése** – Ha nincs ütközés, létrejön egy új Appointment rekord „pending” státusszal.
5. **Értesítések és naplózás** – A rendszer e-mailt küld az ügyfélnek, és Log::info segítségével naplózza az eseményt.

Sikeres mentés után a felhasználó visszajelzést kap a foglalás rögzítéséről.

Itt egy folyamatábra a müködéséről.

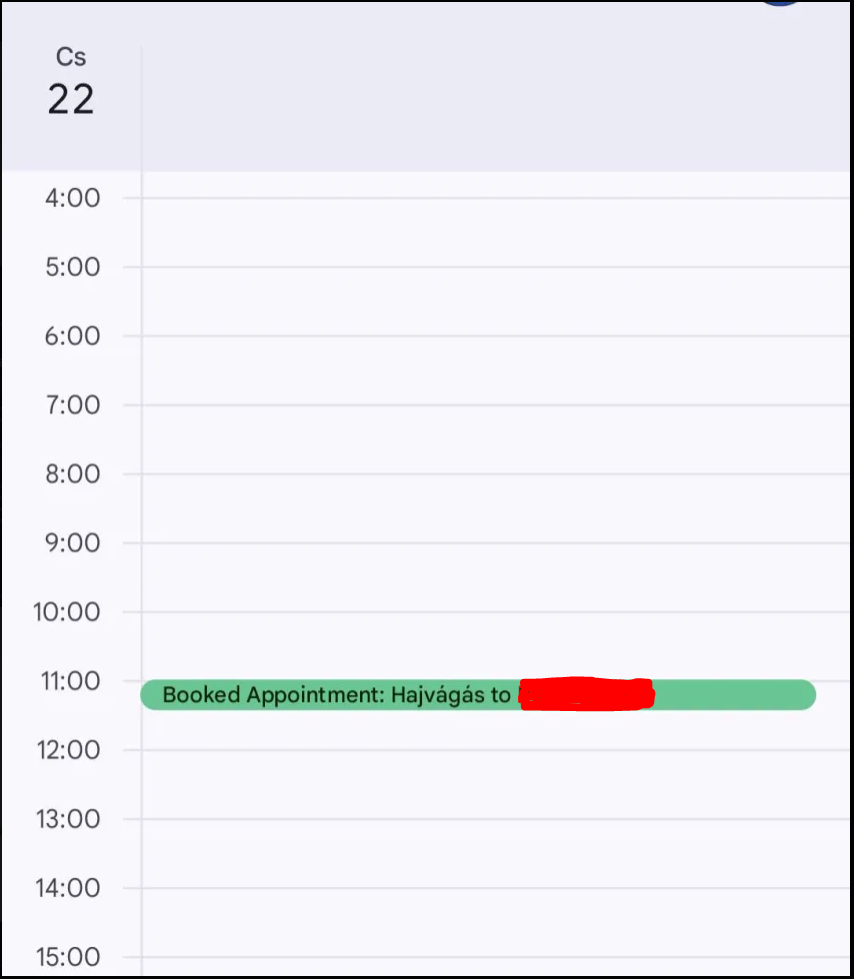


  
Itt történik meg az adatok lekérése, és átadása a fullcalendarnak. Ez alapján mutatja a fullcalendar, hol vannak már foglalások ezzel segítve a felhasználót.

**BookingManagementController.php**

Ez az osztály felel azért, hogy a szolgáltató jóváhagyhassa, vagy elutasíthassa az időpontot. Ezenfelül itt történik meg az időpont Google Calendarba rögzítése, amennyiben ez engedélyezve van.





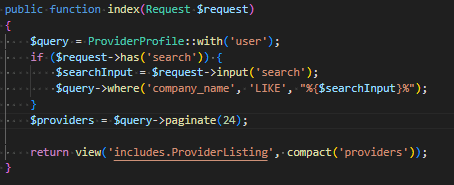
Így néz ki, amikor mentésre kerül a naptárba.

**ProfileController.php**

A ProfileController felelős a felhasználói profilok kezeléséért az alkalmazásban. Amikor egy felhasználó megnyitja a profil szerkesztő oldalát, a vezérlő lekéri a felhasználó alapadatait, és ha a felhasználó szolgáltatóként van regisztrálva, akkor hozzáfér a szolgáltatói profiljához és a munkarend adataihoz is, amelyeket továbbít a nézet felé. A profilfrissítés során a beérkező adatokat validálja és elmenti, figyelve arra, hogy ha az email cím megváltozik, akkor az új emailhez kapcsolódó igazolást újra kell végezni. Szolgáltatók esetén külön metódus kezeli a szolgáltatói profil módosítását, szintén adatellenőrzéssel és mentéssel.

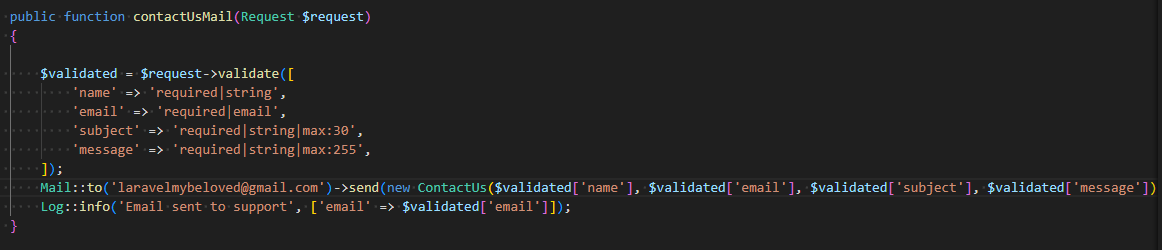
**ProviderListingController.php**

Ez a controller végzi a szolgáltatók listázását, illetve paginációját, hogy csak bizonyos mennyiségű szolgáltató legyen megjelenítve az oldalon, illetve kezeli a keresőmezőt.



**WelcomeController.php**

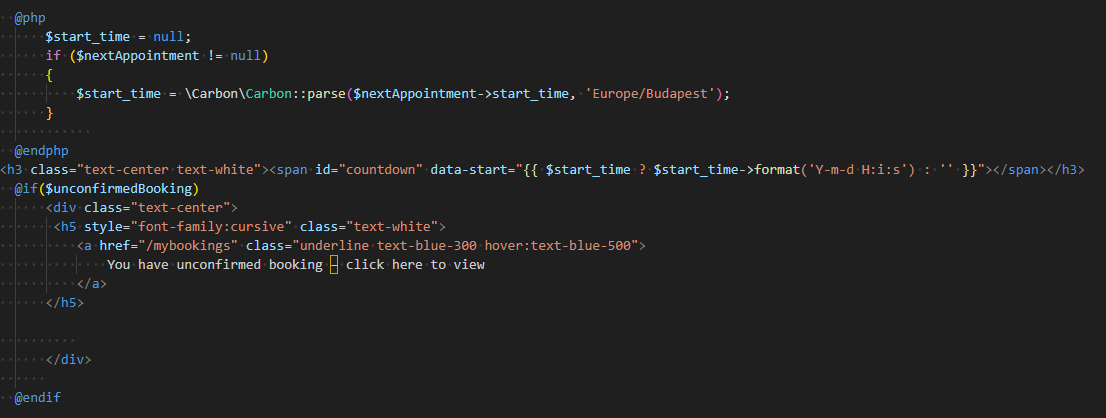
A welcomecontroller a legegyszerűbb. Itt csak annyi történik, hogy a contact us email küldésének logikáját végzi.



**DashboardController.php**

A dashboardon mutatjuk meg a következő megerősített időpontját, és javascript segítségével egy visszaszámlálás indul, illetve ha van a szolgáltatónak meg nem erősített időpontja akkor jelez.



Frontend oldalon:  
  




Ez a rész egy valós idejű visszaszámlálót jelenít meg a felhasználó számára a következő megerősített időpontjáig. A rendszer a szerveroldalon előkészíti a következő foglalás kezdési idejét, majd ezt JavaScript segítségével másodpercenként frissíti a felületen. Amennyiben nincs megerősített időpont, a rendszer ezt egy üzenetben jelzi. Ha az időpont már elkezdődött, a számláló rövid szünet után automatikusan újratölti az oldalt. A szkript minden időegységet (nap, óra, perc, másodperc) kezel, és a hátralévő idő szöveges formában jelenik meg.

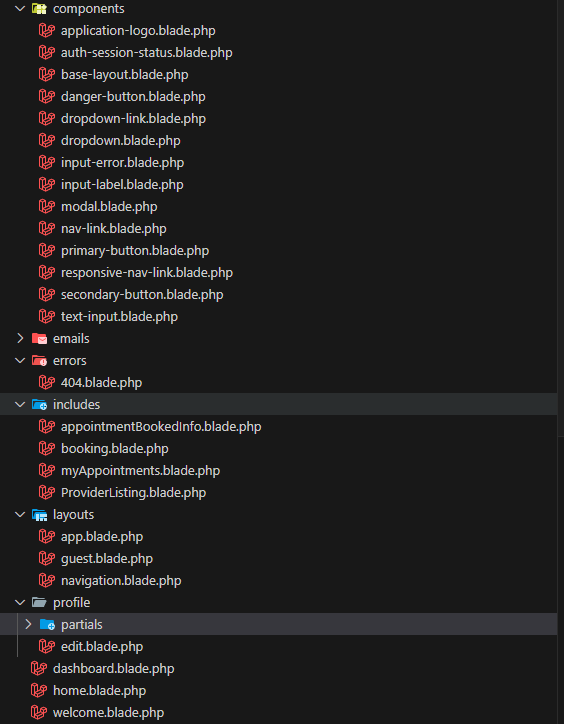
Egy külön figyelmeztető üzenet jelenik meg akkor is, ha a felhasználónak van megerősítetlen foglalása, amely egy linket tartalmaz a foglalás megtekintéséhez.

**AuthenticatedSessionController.php**

  
Ez a controller végzi a bejelentkezést és nézi, hogy a felhasználónak be van-e kapcsolva a 2 faktoros bejelentkezés, hogyha be van, akkor a two-factor login pagere redirectelünk. Ahol lehetőségünk van 2 faktoros hitelesítő kóddal belépni vagy recovery kóddal is van erre opciónk. Ezután a dashboardra.

## **Frontend oldalak**

Van 3 csontváz frontend oldalam, amit az összes többi frontend oldal kiegészít a saját tartalmával. És ezeket a csontváz oldalakat pedig premade komponensekből állnak.



Az összes külső library és third party extension a layouts fájlokba van betöltve, így elég volt csak ott linkelni őket.

## Email küldés mükődése

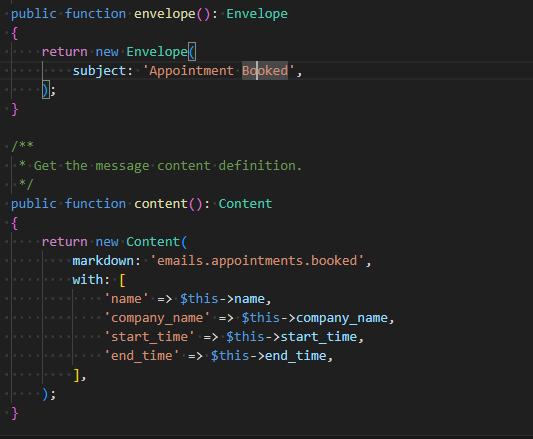
Az alkalmazásban az időpontfoglalást követően automatikus értesítő e-mail kerül kiküldésre. Ez a Laravel Mailable osztályán alapul. Az e-mail küldési logika az alábbi lépésekből:áll:

***Adatok átadása a konstruktorban***:

A Mailable osztály példányosításakor átadásra kerülnek a szükséges változók, mint például a felhasználó neve, a szolgáltató cégneve, valamint a foglalás kezdő- és befejező időpontja.

***Tárgy megadása az envelope() metódusban***  
Az e-mail tárgya (subject) az Envelope objektumban kerül meghatározásra, például: Appointment Booked.

***Tartalom összeállítása a content() metódusban***  
A Content objektumban megadjuk a markdown nézetfájlt (emails.appointments.booked), valamint az e-mailhez szükséges változókat (with kulcs alatt) továbbítjuk a nézetnek.



***Frontend sablon (Markdown)***  
Az emails.appointments.booked sablon tartalmazza az e-mail HTML struktúráját. A változók (pl. név, időpont) a nézetben sablonkóddal jelennek meg ({{ $name }}, {{ $start\_time }} stb.), így az e-mail személyre szabott tartalommal kerül kiküldésre.

## Logolás

Az alkalmazás naplózási (logolási) mechanizmusa a Laravel beépített Log szolgáltatásán alapul, amely a háttérben a Monolog könyvtárat használja. A rendszer minden jelentősebb eseményt automatikusan vagy manuálisan naplóz, annak érdekében, hogy a program működése átlátható, auditálható és hibák esetén könnyen visszakövethető legyen.

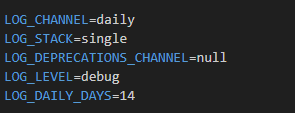
A logolás kulcsfontosságú része az alkalmazás üzemeltetésének és karbantartásának. A segítségével az alábbi célok érhetők el:

* **Hibakeresés:** ha az alkalmazás hibába ütközik, a naplófájlok alapján pontosan visszakereshető, hol és mi történt
* **Biztonság:** a rendszerbe való belépési próbálkozások, 2FA hitelesítések, fióktörlések és egyéb kritikus műveletek naplózása segíti a lehetséges támadások vagy jogosulatlan hozzáférések elemzését.
* **Működés követése:** a foglalások, megerősítések, Google Calendar integrációk és egyéb műveletek logolása támogatja az adminisztratív visszakereshetőséget és az esetleges felhasználói reklamációk kezelését.

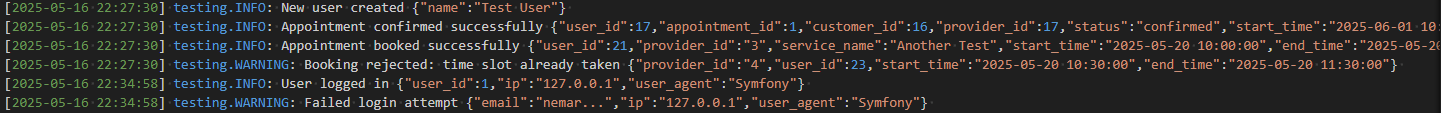
A logbejegyzések alapértelmezés szerint a storage/logs/laravel.log fájlban kerülnek rögzítésre, ahol időbélyeggel, log szinttel (info, warning, error, stb.) és részletes üzenettel látják el az eseményeket.

Példák naplózott eseményekre:

* Új időpont foglalása sikeres validáció után (Log::info('Sikeres foglalás', [...]))
* Google Calendar esemény mentése (Log::info('Google esemény mentve') vagy Log::error('Google API hiba'))
* Sikeres és sikertelen bejelentkezések
* 2 faktoros hitelesítés indítása vagy hibája
* Felhasználói fiók törlése

Az alkalmazás igény esetén bővíthető külön log csatornával is, például külön fájlba irányíthatók a foglalásokkal kapcsolatos naplók vagy a külső API hívások. Ez a Laravel konfigurációs fájljaiban (config/logging.php) könnyen testreszabható. Illetve az .env fájlban is lehetőségünk van ezt testreszabni.  


Példa a logfile egy részletéből:



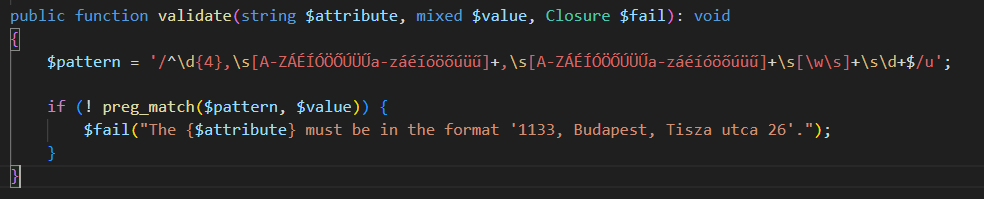
## Tesztelési dokumentáció

* Unit tesztek

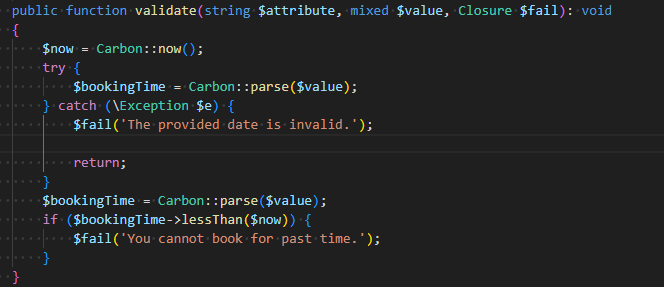
2 unit teszt osztályt hoztam létre azon belül 7 unit teszt functiont

Először teszteltem az AddressValidationRuleomat, ami a a regisztrációkor validálja a address inputot, hogy megfelelő formátumban kerül elmentésre az adatbázisban.

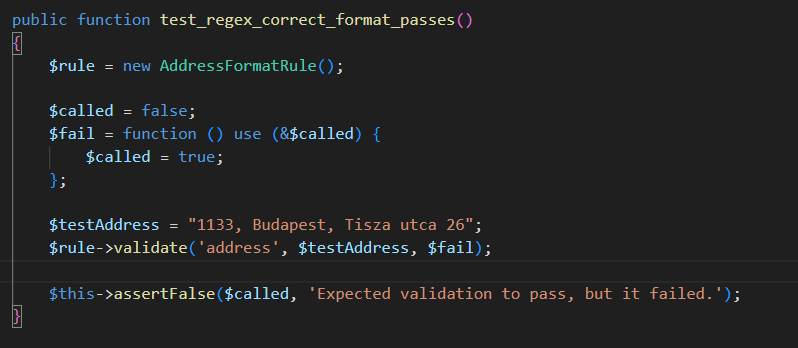
**AddressFormatRule.php**



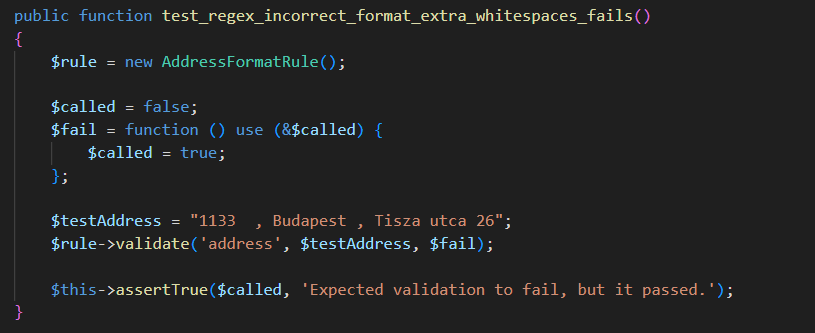
**DateValidationForBookingRule.php**

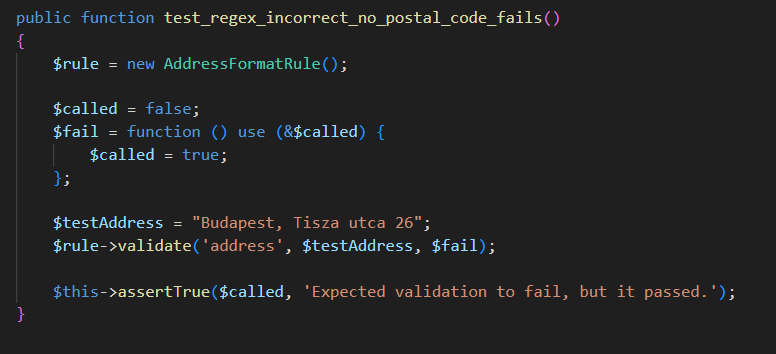
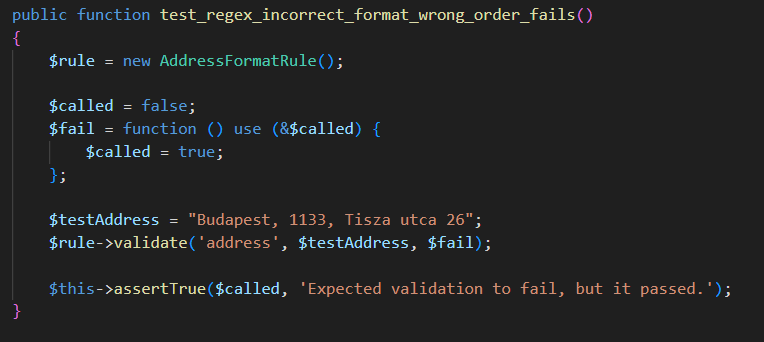


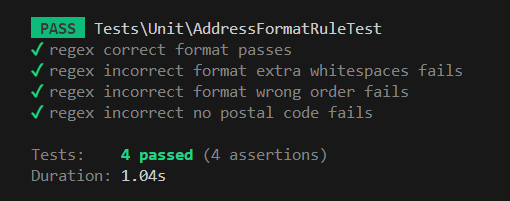
Először megnéztem, hogy átmegy-e a kívánt validáción.



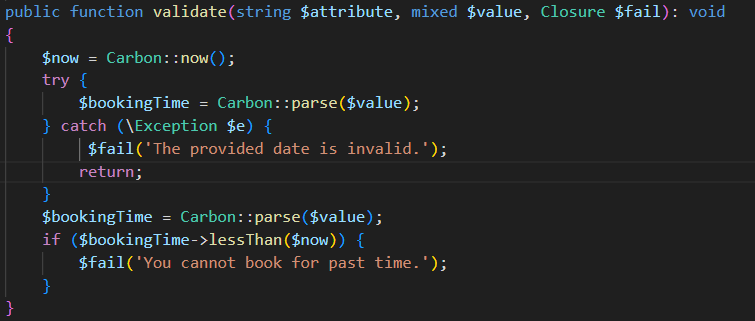
Ezután teszteltem különböző a helyes formátumtól eltérő inputot. Például, hogy több szóközt tartalmaz, nem tartalmaz irányítószámot vagy rossz sorrendben vannak az adatok.





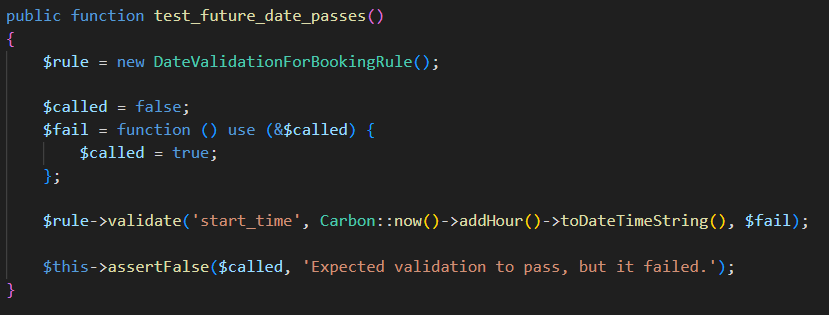
A tesztjeim sikeresen lefutottak és az összes passed.  


**DateValidationForBookingRuleTest.php**

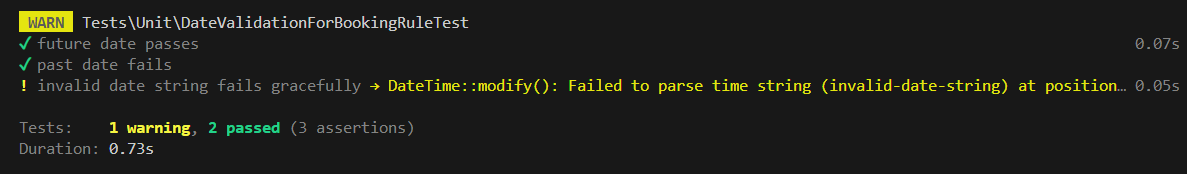
Ez a rule validálja le, hogy a foglalás date formátum legyen, illetve, hogy ne lehessen úgy foglalni, hogy később lenne a kezdőidőpont, mint, a befejező időpont.

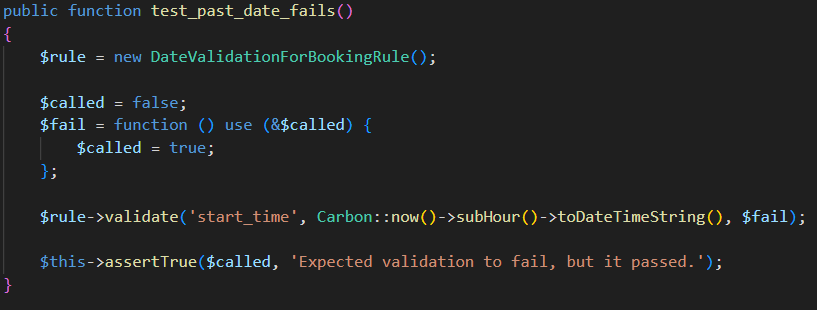
Ehhez a validációs szabályhoz is írtam unit teszteket.

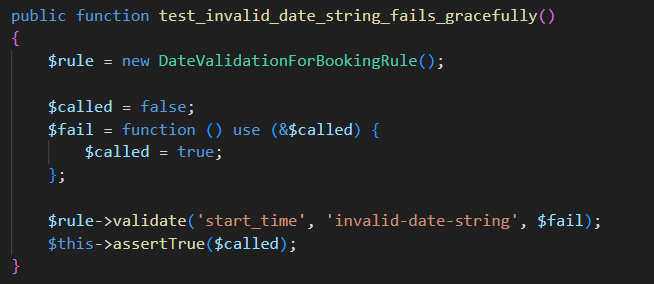
Első alkalommal megnézem, hogy a rule elvárt müködése helyes-e.



Ez után megnézem, hogy megfelelően validálja, hogyha valaki korábbi kezdő időt használ, mint végzőidőt, illetve, hogyha nem megfelő dátumot kapunk, akkor excpetiont dobunk és returnöl.





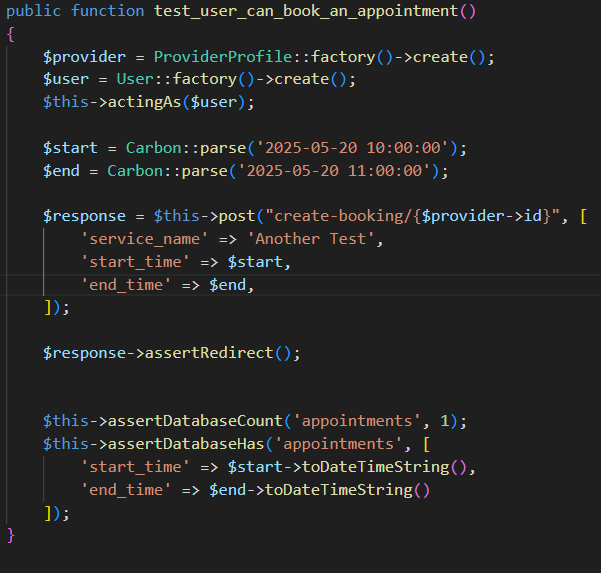


**Feature tesztek**

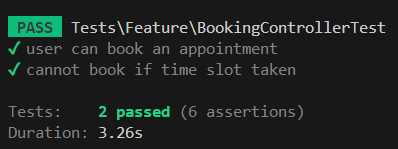
Feature tesztekből a phpunit-ot használtam, illetve 3 tesztosztályt hoztam létre, ezenfeül a laravel automatikusan létrehozott tesztjeit is használtam.

**BookingTest.php**

Ez a feature teszt, azért jelentős, mert ez ellenőrzi, hogy a foglalás logika megfelelően müködik, illetve a validáció, hogy ne lehessen ugyanarra az időpontra több embernek foglalnia. A tesztjeim, ahol fontos, hogy véletlenül se legyen ilyen adat az adatbázisba használják a RefreshDatabase trait-et, hogy egy üres adatbázisba tudjak tesztelni.

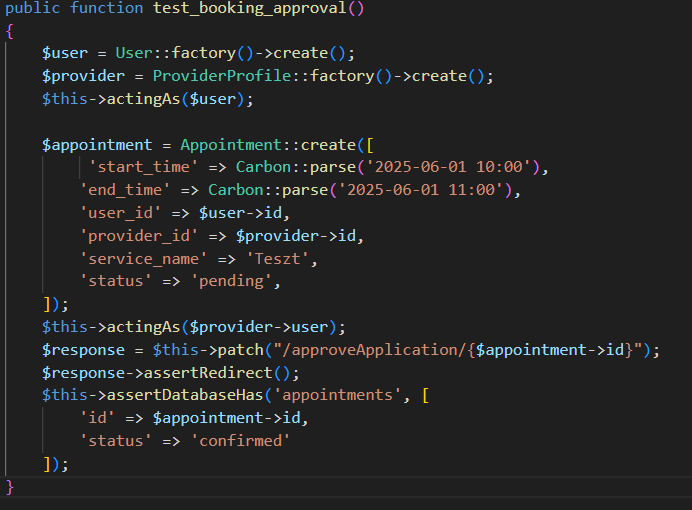




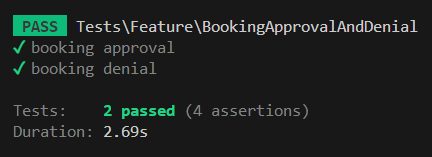


**BookingApprovalAndDenialTest.php**

Itt teszteltem le, hogy a szolgáltató (provider) jóvá tudja hagyni az időpontot, illetve elutasítani is tudja.

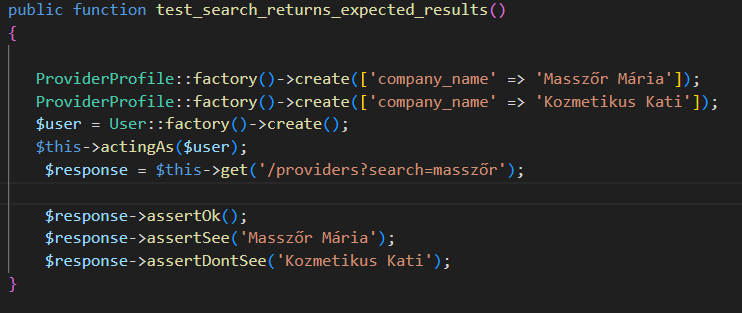


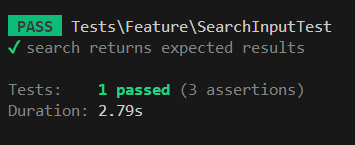




**SearchInputTest.php**

Itt a szolgáltatók kilistázásakor jelenlévő search input fieldet teszteltem le.



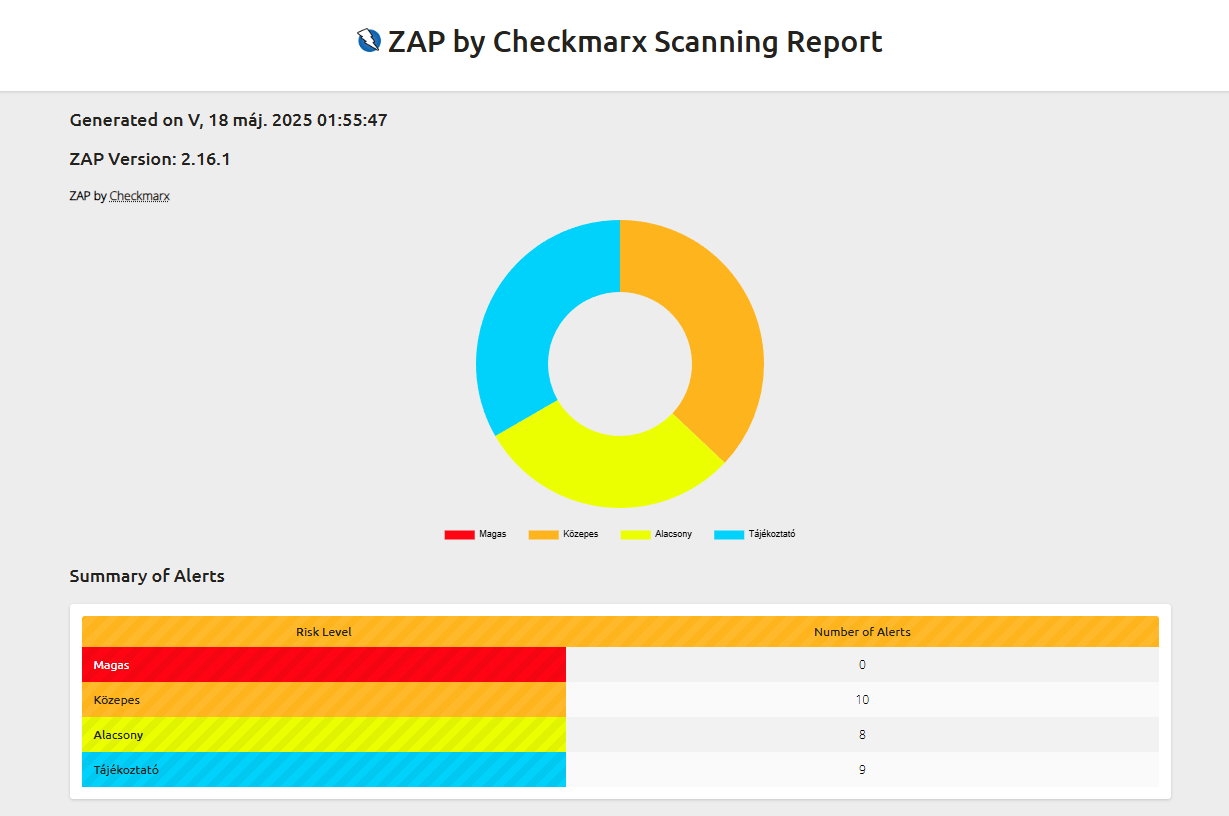


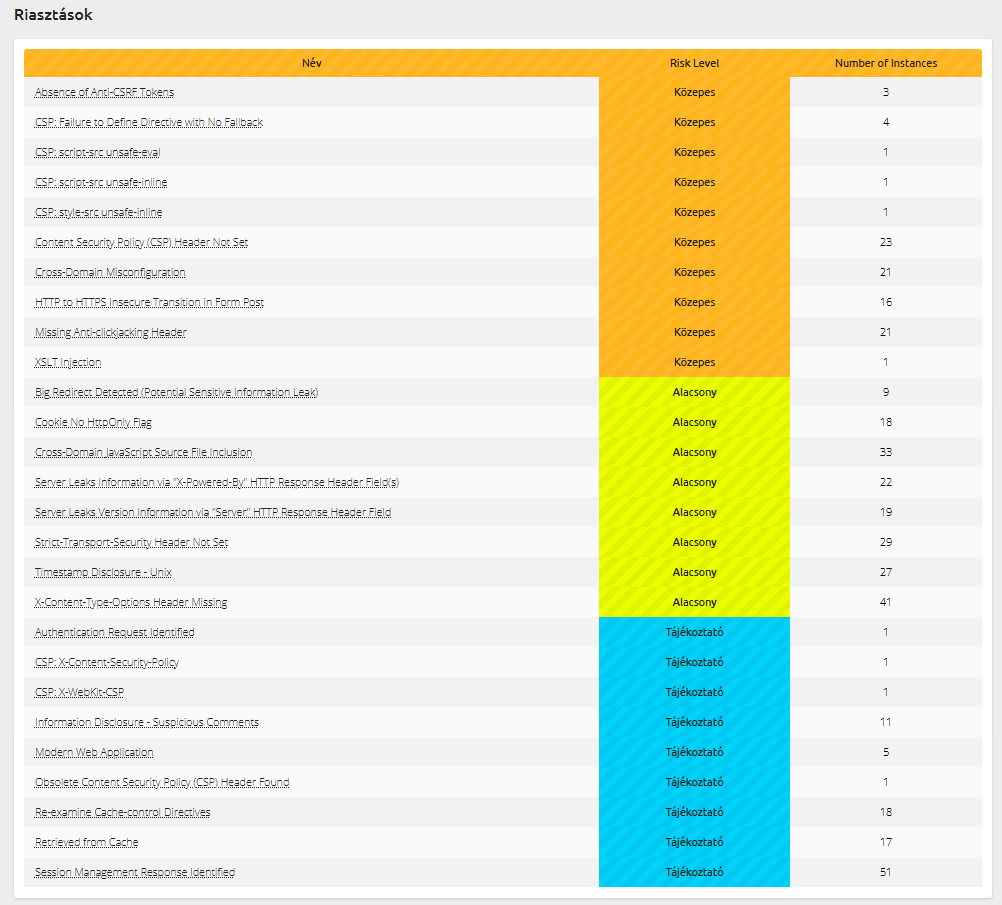
Ezeken a teszteken túl, volt lehetőségem tesztelni a laravel automatikusan létrehozott tesztjeivel az autentikációt és ehhez kapcsolodó routokat, és a profiloldalt.

Kódelemzés

**Owasp zap elemzés**

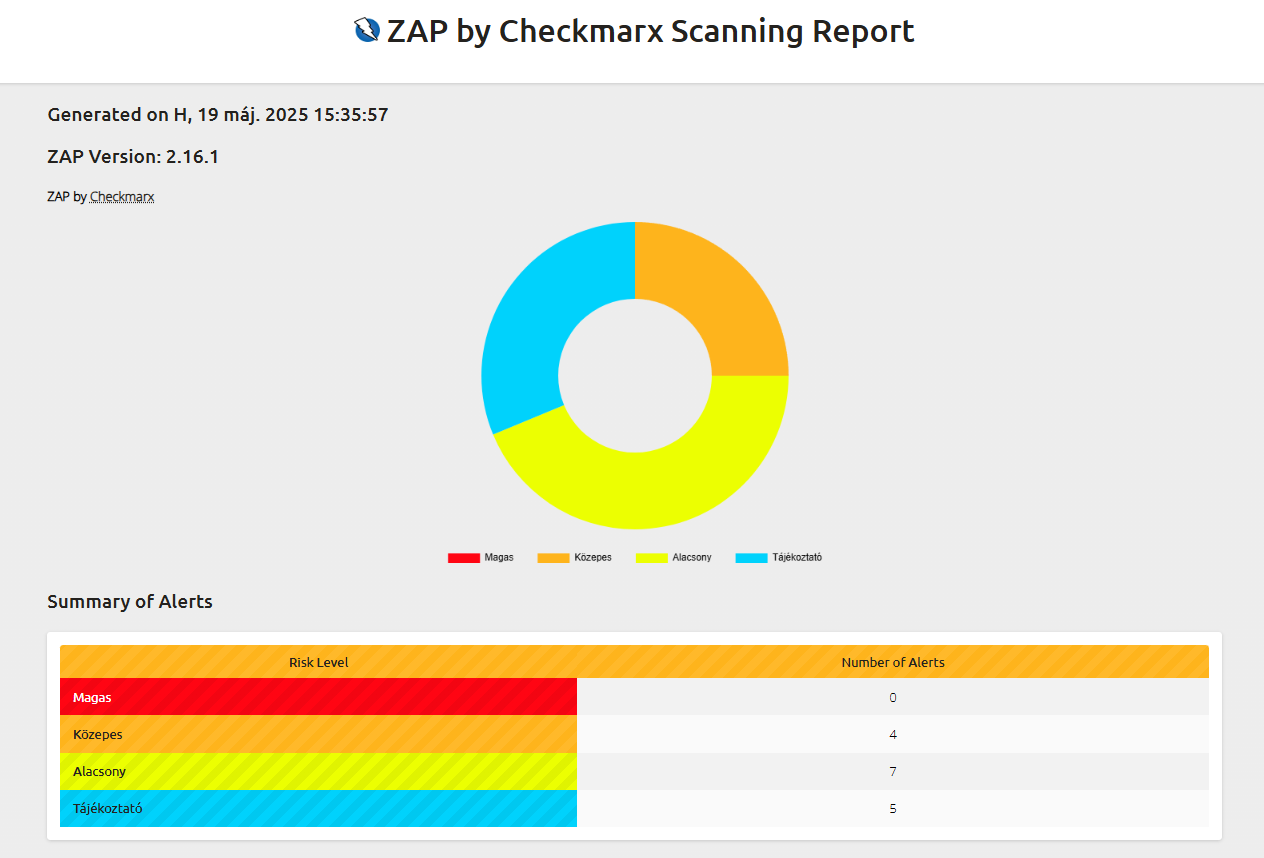
Első futattásra ezeket az alerteket kaptam az alkalmazásomhoz. Ez az első futattás egy felderítés volt számomra, hogy milyen sebezhetőségek vannak a projektemben, és hogyan tudok ezen segíteni.

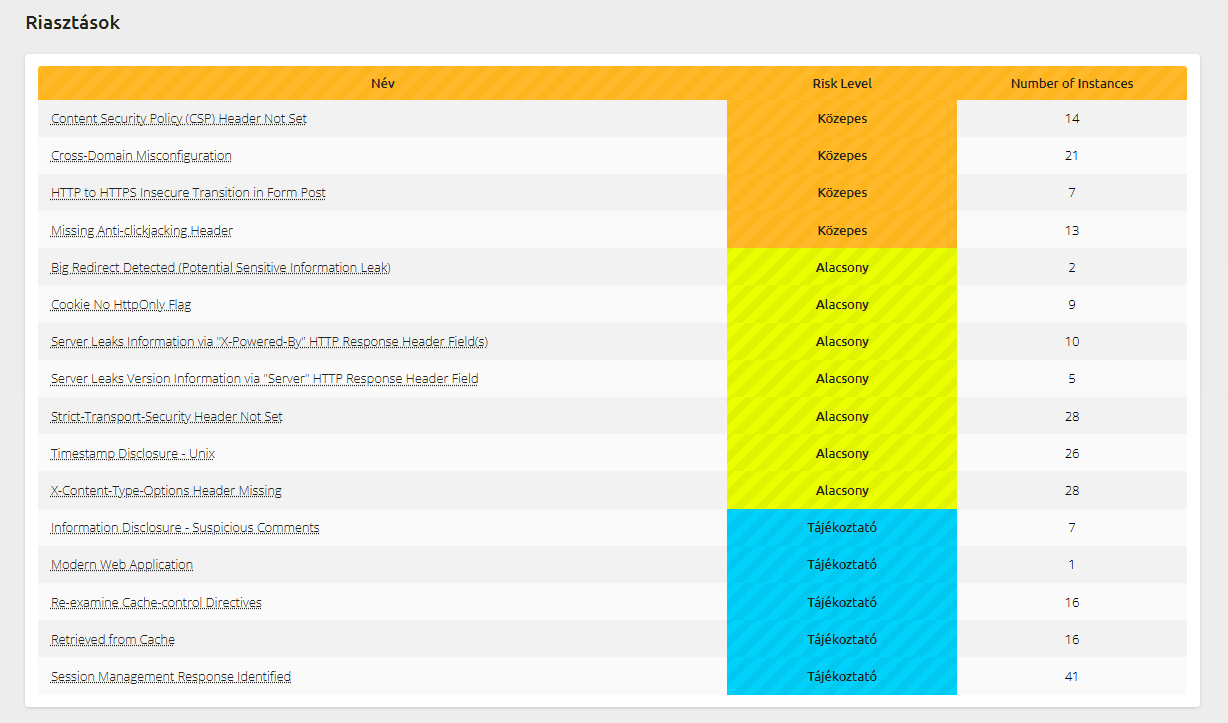


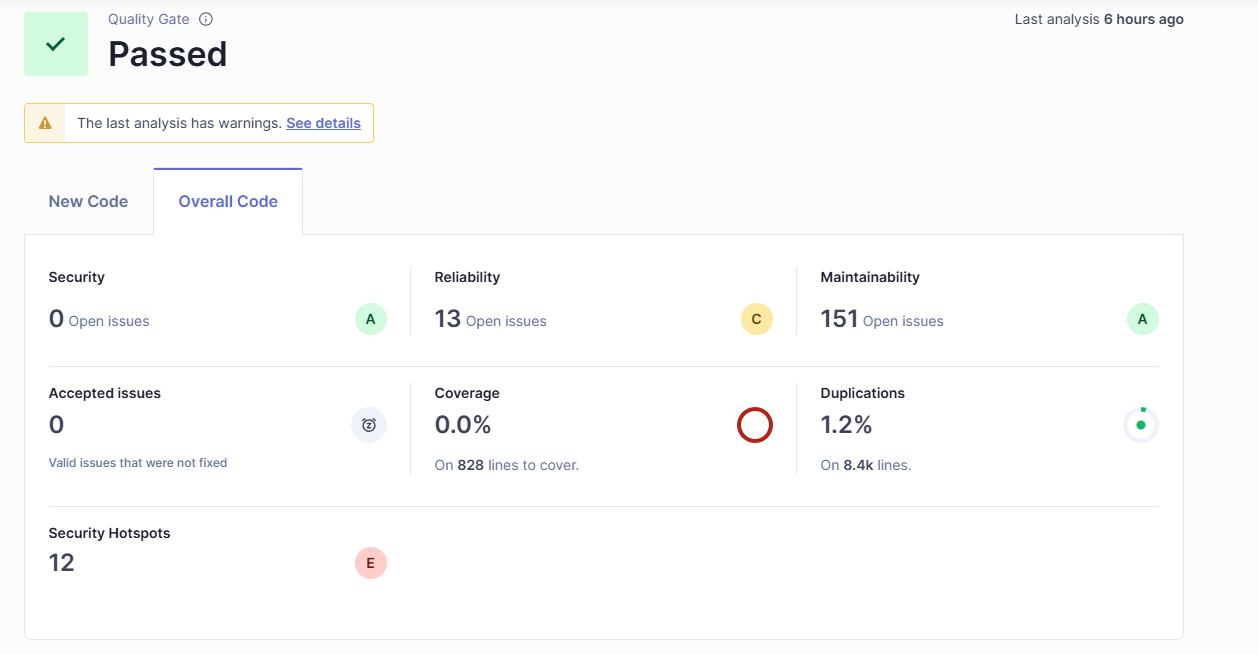


Zap második futattás (néhány sérülékenység kijavítva):

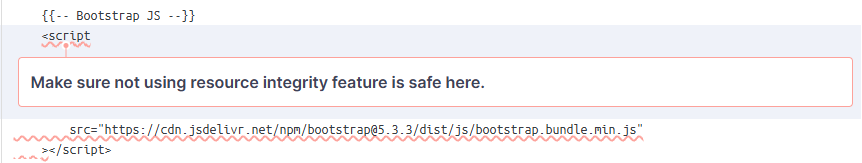
Sajnos időszükében nem volt időm minden sérülékenységet kijavítani, viszont az owasp zap egy remek eszköz a sérülékenységek felderítésében.

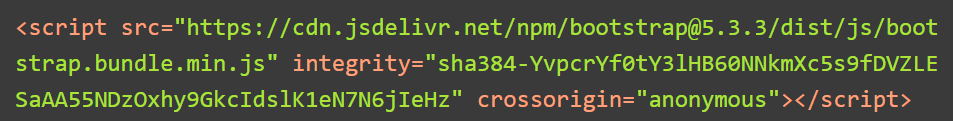




**Sonarqube elemzés**  


A Sonarqube elemzést követően javítottam biztonsági problémákat az alkalmazásomban.  
Köztük a külső Javascript és CSS fájlokat Subresource Integrity (SRI) használatával töltjük be, hogy az erőforrás tartalma ne változhasson meg észrevétlenül, például CDN manipuláció esetén.



Javítottam, hasht adtam hozzá.  


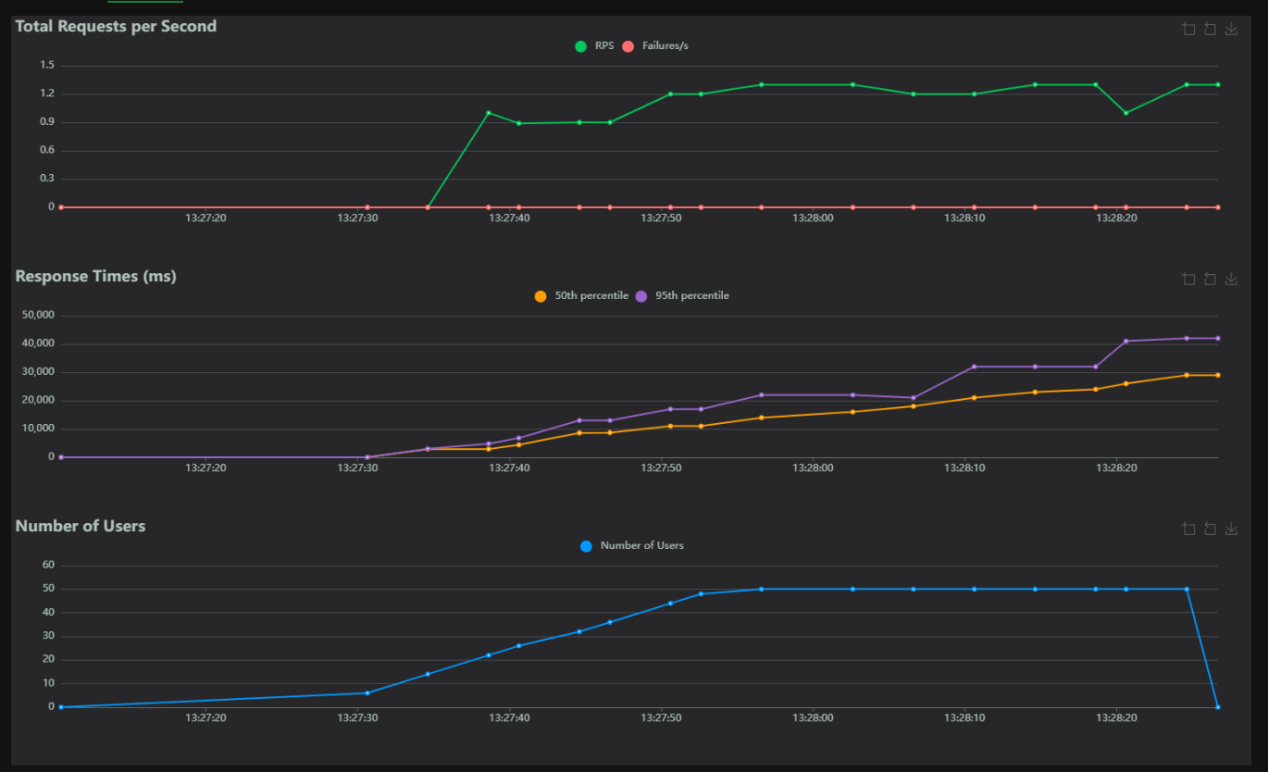
Ezen felül javítottam a reliabilityket  
required helyett required\_once, alt tag az imagehez  




Locust load testing (terherhetőség teszt):

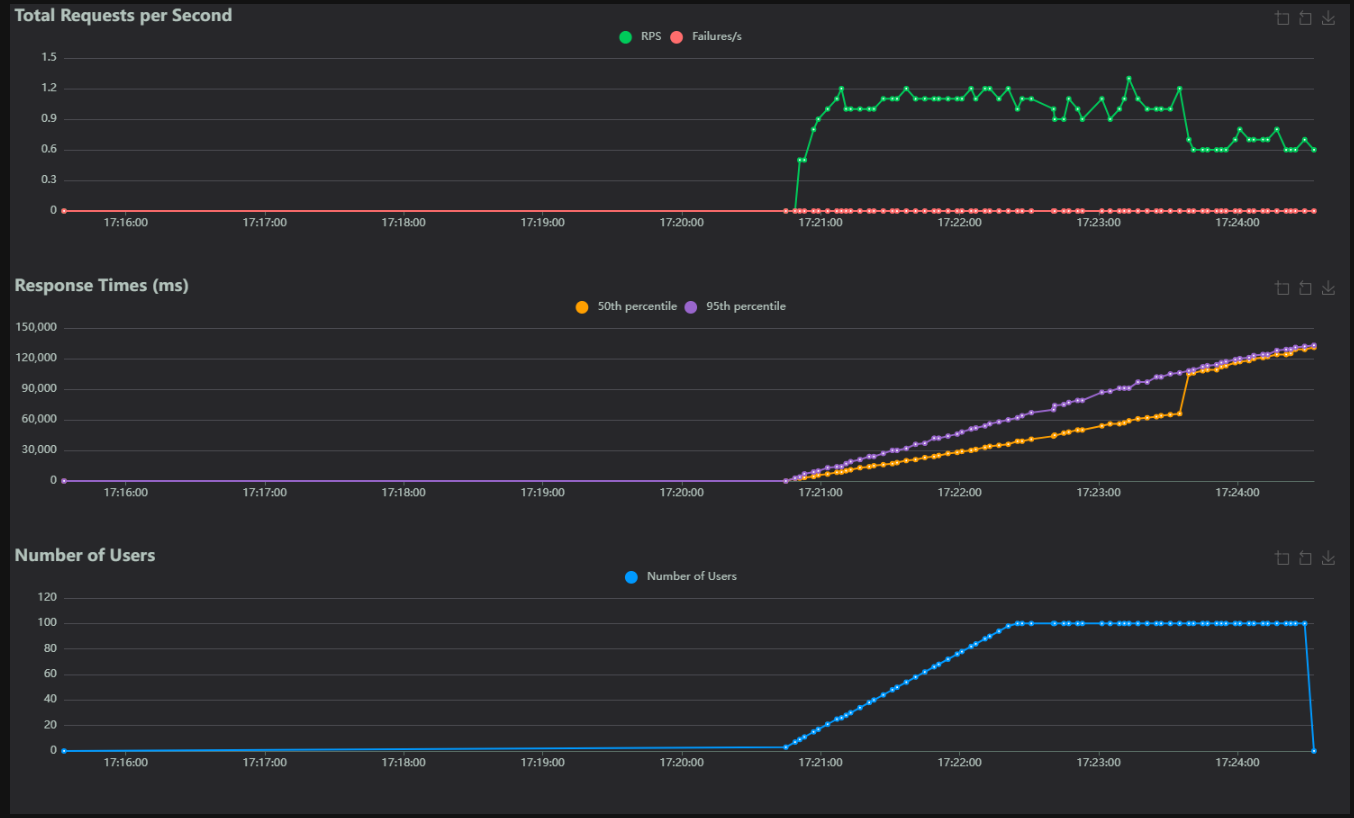
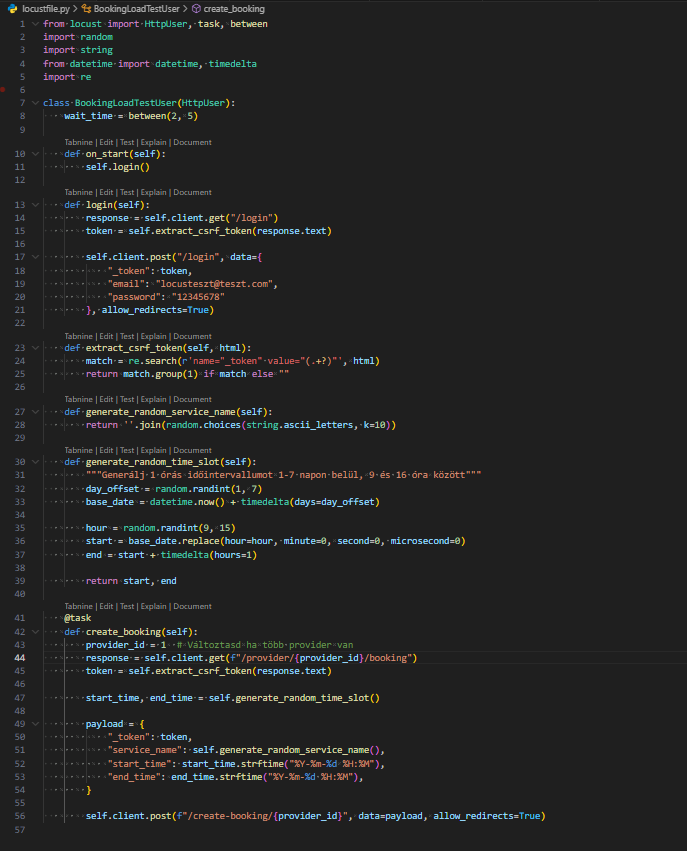
A tesztelés során volt alkalmam load testinget is csinálni az alkalmazásomon, ezt a locust nevű python framework segítségével hajtottam végre. A locustban minden user valójában egy virtuális felhasználó. Ezek nem valódi emberek, hanem párhuzamosan futó szimulált kliensek, amiket a locust futatt. Olyan mintha egy ember navigálna a webalkalmazásban.

Első loadteszt:



Itt 50 “virtuális“ felhasználó jelentkezik be, majd meglátogatták a dashboard oldalt, ez lövetően pedig a “Saját foglalások” (My Bookings) oldalt. A ramp-up idő másodpercenként 2 felhasználó, vagyis másodpercenként 2 új felhasználó kapcsolódott a rendszerhez.   


A második teszt a foglalás funkció válaszidejét vizsgálta 100 felhasználóval. A felhasználók egyesével kerültek be a tesztbe és a futásidő kürölbelül 4 perc volt.

**Megállapítások:**

A grafikonon kitűnően látszik, hogy ugyan a teszt nem futott hibára, ugyanakkor a foglalási algoritmusomnak még szüksége lesz a jövőben egy refaktorálásra, a mostani állapotában nincs optimalizálva jelentős felhasználószámra.   
Megfontolandó frissítések: query optimalizálás, Async vagy queue alapú feldolgozás

# Összegzés

## Továbbfejlesztési lehetőségek

* Kód optimalizálás (Locust load teszt alapján)
* Az esetlegesen megmaradt sérékenységek javítása
* Ünnepnapok, illetve szabadság beállítási lehetőség, hogy abban az intervallumban ne lehessen foglalni.
* Szolgáltatás név (service name) hozzáadása szolgáltatónként és így a foglaláskor egy legördülő menüből (dropdown) lehessen választani.
* Ha egy ügyfél minden héten megbeszélten ugyanakkor érkezik, akkor lehessen beállítani ismétlődő foglalást.
* Ha valamilyen oknál fogva szükséges lenne, akkor ügyfél letiltás.
* Időpont módosítási és törlési lehetőség appon belül (ügyfeleknek és szolgáltatóknak).
* Vendégként (Guest) való foglalási opció
* Appon belüli értesítések (notifications) szekció.
* Többnyelvűség beállítása az appon belűl.
* Több tartalom a dashboard oldalra
* Közeli szolgáltatók listázása (ha megadunk lakcímet)
* Több szűrés lehetőség a szolgáltatókra
* Szolgáltató értékelése lehetőség, később szűrés értékelés alapján
  + Esetleges statisztikák
  + Grafikonok
* Olyan ötletek, amelyeket még érdemes a jövőben megvalósítani
* Joker rész, minden amivel még jobb lehet a projekt majd, amivel még eladhatóbb, még többet tudhat majd stb.
* Itt azt kell prezentálni, hogy érted a piaci trendeket, elvárásokat.

# Felhasznált irodalom

Laravel dokumentáció: <https://laravel.com/docs/12.x/>

Stackoverflow: <https://stackoverflow.com/questions>

FullCalendar: <https://fullcalendar.io>

Bootstrap:<https://getbootstrap.com>

Intl-tel-input Team (2024) International phone input: <https://intl-tel-input.com>

SonarQube: <https://www.sonarsource.com/products/sonarqube/>

Zap: <https://www.zaproxy.org>

phpMyAmin: <https://www.phpmyadmin.net>

Xampp: <https://www.apachefriends.org/hu/index.html>

Docker: <https://www.docker.com>

Github: <https://github.com>

Animated CSS: <https://animate.style>

UIkit: <https://getuikit.com>

Locust: <https://locust.io>

Felhasznált cikkek:

Szerző: Hafiq Iqmal, A megjelenés éve: Feb 24, 2021, A cikk címe: Laravel Benchmark with Locust Url: <https://hafiqiqmal93.medium.com/laravel-benchmark-with-locust-6f891c16ddac>

Szerző: Gonzalo Gomez, A megjelenés éve: September 03, 2024, A cikk címe: How to integrate Google Calendar with Laravel, Url: <https://ggomez.dev/blog/how-to-integrate-google-calendar-with-laravel>

Szerző: [Shpëtim Islami](https://medium.com/@draodakum?source=post_page---byline--8bf832eeb471---------------------------------------), A megjelenés éve: Dec 22, 2021, A cikk címe: Performance testing your application, Url: <https://medium.com/laraveladvanced/performance-testing-your-application-8bf832eeb471>

Szerző: [Adedayo Adedoyin](https://medium.com/@joshuaadedoyin2?source=post_page---byline--db24df8a2e2c---------------------------------------), A megjelenés éve: Aug 25, 2023, A cikk címe: Dockerizing a Laravel Project: A step-by-step guide, Url: <https://medium.com/@joshuaadedoyin2/dockerizing-a-laravel-project-a-step-by-step-guide-db24df8a2e2c>

Szerző: [Akeem Amusat](https://dev.to/aoamusat), A megjelenés éve: 2021. okt. 6., A cikk címe: Sendint Email in Laravel App Using Gmail Account, Url: https://dev.to/aoamusat/sending-email-in-laravel-app-using-gmail-account-iaj

--------------------------------------------------------------------------------------------------------