# Tóth Levente Bálint, Pácz Barnabás, Prukner András

## Projekt:

A projektünk egy autóbérlési és versenypálya-kezelési rendszer, amelynek célja, hogy felhasználóbarát weboldalt biztosítson a felhasználók számára és asztali alkalmazást az adminok számára. A projekt két fő komponensből áll:

Laravel Weboldal: Az online felhasználók számára lehetővé teszi az autók foglalását.

MAUI Asztali Alkalmazás: Adminisztációs felületett biztosít az autók, pályák, események kezelésére

Mindkét rendszer egy közös REST API-t használ, amely az adatokat egy központi adatbázisból szolgáltatja.

## Használt rendszerek

### Laravel:

#### Funkciók:

\* Felhasználói regisztráció és bejelentkezés

\* Autó foglalás eseményekre

\* Asztali alkalmazás az felhasználók, foglalások, autók és pályák kezelésére

### MAUI Asztali Alkalmazás:

#### Funkciók:

\* Az API-n keresztül:

\* Versenypályák felvétele, modósítása, törlése

\* Autók felvétele, modósítása, törlése

\* Foglalások kezelés

\* Felhasználók kezelése

## REST API:

## Laravel alapú API

### Fő Funkciók:

\* Adatok elérhetővé tétele az adatbázisból

\* Hitelesítés és jogosultságkezelés

\* Foglalások kezelése

\* Adatok szűrése és keresése

\* Adatbázis struktúra:

\* Felhasználók (regisztrált ügyfelek és adminok)

\* Autók (elérhető bérleti járművek)

\* Versenypályák (foglalható helyszínek)

\* Foglalások (felhasználói és adminisztratív tranzakciók)

\* Jövőbeli Bővítési Lehetőségek:

\* Mobilalkalmazás fejlesztése Androidra és iOS-re

\* Támogatás többnyelvű kezelőfelülethez

#### Githgub Letrehozasa

\* Github link: https://github.com/LeventeTooth/Project.git

#### Az adatbázis megtervezése:

---

Az adatbázis tervezéséhez a `drawsql.app` nevű oldalt használtuk. Tervezéskor figyelembe vettünk rengeteg szmepontot, hogy milyen adatokra lesz szükségünk az ötleteink megvalósításához, mikre lehet szükség.

#### C# MAUI alkalmazás:

Az asztali (windows) felhasználásra tervezett programot a Mincrosoft .NET keretrendszerében; .NET MAUI- ban készítettük el. A MAUI lehetőséget biztosít, hogy letisztultan, átláthatóan tudnjunk MVVM `(Model View ViewModel)`struktúrával dolgozni.

\* A `Model` az adatok szerkezerte

\* A `View` a felhasználó által látható kezelőfelület

\* A `ViewModel` az pedig az az egység ami az elöző két részt összeköti

A három rész eggyüt alkott egy egész programot.

**Rendszer Architektúra**

1. Áttekintés

Ahogy korábban már említettük a projekt 2 fő részből áll egy laravel backend és webes felület, illete egy MAUI asztali alkalmazás és mindkét rendszer egy közös REST API-t használ.

* A Laravel backend biztosítja az összes adatlogikát
* A REST API-n keresztül kommunikál a MAUI alkalmazással és a weboldallal
* A backend felelős az adatok eléréséért, tárolásáért, ellenőrzéséért és feldolgozásáért

1. Útvonalak

2.1 Útvonalak amiket a frontend használ:

* Az események, autók, pályák, csapatok csak a laravel resource útvonalait használják.
* A felhasználók az auth resource útvonalait valamint az alábbi egyedi útvonalakat:
  + /login
  + /my-profile
  + /logout
  + /register
* A foglalás a resource útvonalain felül pedig ezeket az egyedi útvonalakat használja:
  + /rents/create/{event}/{car}
  + /rents/thank-you-page

2.2 Útvonalak amiket a REST API használ:

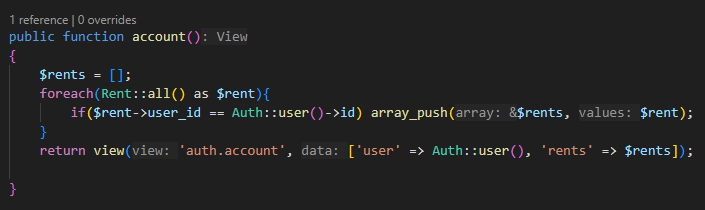
A REST API csak resource útvonalakat használ, a következő modellekhez:

* Foglalás
* Autó
* Pálya
* Esemény
* Csapat

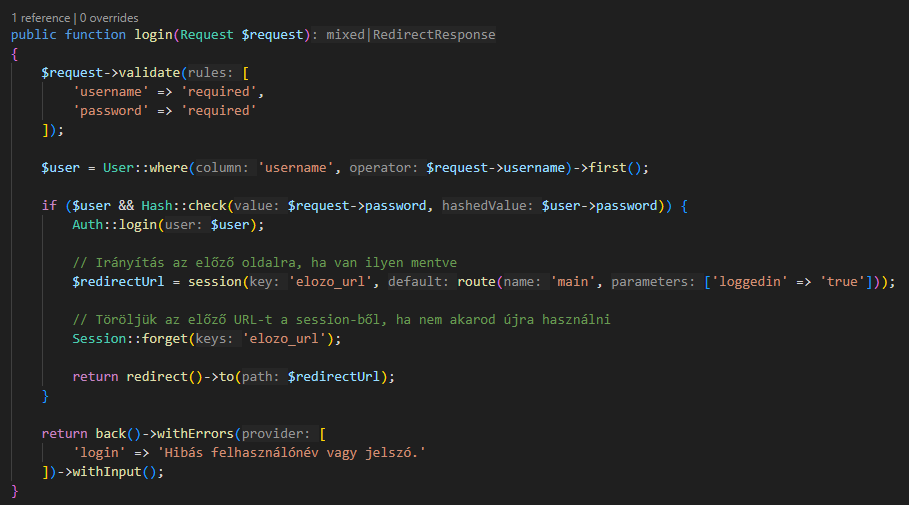
1. Controller logikák

3.1 Auth Controller

* index(): vissza adja azt a frontendet amin a felhasználók be tudnak jelentkezni
* logout(): Kijelentkezteti a felhasználót és visszairányít a kezdőlapra
* account(): Összegyűjti az adott felhasználóhoz tartozó rendeléseket és visszaadja azt a frontendet amin a felhasználó saját adatai jelennek meg és a hozzá tartozó foglalások.



* login(): lebonyolítja a bejelentkezés folyamatát:
  + validálja, hogy tényleg megadásra kerültek e a szükséges értékek az űrlapon
  + megkeresi felhasználónév alapján a User modellt
  + Ellenőrzi a hashelt jelszót
  + bejelentkezteti a felhasználót
  + visszairányít arra az oldalra honnan megnyitotta a felhasználó a bejelentkezést



* create(): visszaadja regisztrációs frontendet
* store(): lebonyolítja a regisztrációs folyamatot:
  + validálja, hogy tényleg megadásra kerültek e a szükséges értékek az űrlapon
  + hasheli a jelszót
  + létrehozza az új felhasználót
  + átirányít a sikeres regisztráció frontendre
* edit():
  + id alapján megkeresi az adott felhasználót és átirányít az adatmódosító frontendre
* update():
  + id alapján megkeresi az adott felhasználót
  + validálja, hogy tényleg megadásra kerültek e a szükséges értékek az űrlapon
  + frissíti a User modellt
  + átirányít a sikeres adat frissítés frontendre
* destroy():
  + id alapján megkeresi az adott felhasználót
  + törli a User modellt
  + átirányít a sikeres felhasználó törlés frontendre

3.2 Car Controller

* index(): Visszaadja az összes autó frontendet

3.3 Event Controller

* index(): Visszaadja az összes esemény frontendet
* show(): Visszaadja azt a frontendet ahonnan indul a foglalás
  + Megjeleníti az esemény adatait
  + Megjeleníti a foglalható autókat
    - Csak azokat az autókat lehet lefoglalni amik még szabadok

3.4 Group Controller

* index(): Visszaadja az összes csapat frontendet
* create(): index(): Visszaadja az a csapat létrehozás fontendet
* store():
  + validálja, hogy tényleg megadásra kerültek e a szükséges értékek az űrlapon
  + Létrehozza a csapat modellt
  + Visszairányít az összes csapat oldalra egy siker üzenettel
* destroy():
  + id alapján megkeresi a csapat modellt
  + megvizsgálja hogy a felhasználó melyik csapathoz tartozik
    - csak azt a csapatot engedi törölni amelyiknek a tagja
  + Visszairányít az összes csapat oldalra

3.5 Track Controller

* index(): Visszaadja az összes pálya frontendet

3.6 Rent Controller

* createRent(esemény, autó):
  + visszaadja a foglalás űrlap frontendjét aminek átadja a már kiválasztott eseményt és autót, illetve a foglalható időpontokat.
* store():
  + validálja, hogy tényleg megadásra kerültek e a szükséges értékek az űrlapon
  + létrehozza a foglalást
  + a sessionbe elmenti a foglalás adatait a thank you oldalnak
  + visszaadja a thank you page-et
* thankYouPage():
  + a sessionből kiszedi a foglalás adatait
  + megkeresi az esemény és az autó adatit
  + vissza adja a frontendet aminek átadja foglalás összes adatát amit megjelenít a frontend

3.7 API Controllerek

Minden modellhez a szükséges CRUD műveleteket tartalmazza az összes API controller.

* API Car Controller
* API Event Controller
* API Group Controller
* API Rent Controller
* API Track Controller