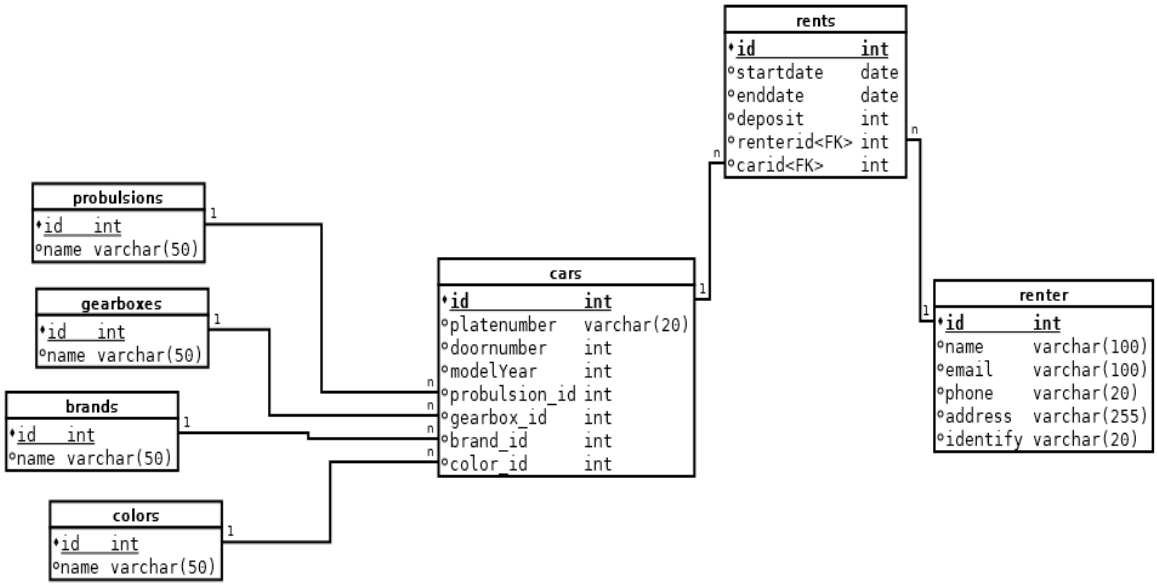
Tartalomjegyzék

Adatbázis

Terv:



Szerkezet:

Propulsions:

- name: meghajtás neve

Gearboxes:

- name: váltótípus neve

Brands:

- name: Márka neve

Colors:

- name: Szín neve

Cars:

- platenumber: rendszám
- doornumber: ajtók száma
- modelYear: évjárat
- propilsion_id: külső kulcs a propulsions táblához
- gearbox_id: külső kulcs a gearboxes táblához
- brand_id: külső kulcs a brands táblához
- color_id: külső kulcs a color táblához

Rents:

- startdate: kölcsönzés kezdeti dátuma
- enddate: kölcsönzés vége
- deposit: kölcsönzés letéti ára
- cars_id: külső kulcs a cars táblához
- renter_id: külső kulcs a renters táblához

Renters:

- name: kölcsönző neve
- email: kölcsönző email címe
- phone: kölcsönző telefonszáma
- address: kölcsönző lakcíme
- identify: kölcsönző személyi igazolványszáma

Adabázis létrehozása

Az adatbázis két módszerrel lehet létrehozni:

1. Az API beüzemelése után az env.example fájl másolása és env-re való átnevezése után a DB_DATABASE mezőnek megkell adni egy tetszőleges nevet. Majd az XAMPP indítása után a böngészőbe a http://localhost/phpmyadmin/ felületen a megadott névvel létrehozni egy táblát majd az API-ban futtatni a php artisan migrate parancsot. Ezzel létrejönnek a megtervezett táblák és a laravel saját táblái, és a projekt működésének tesztelése érdekében kreált tesztadatok.

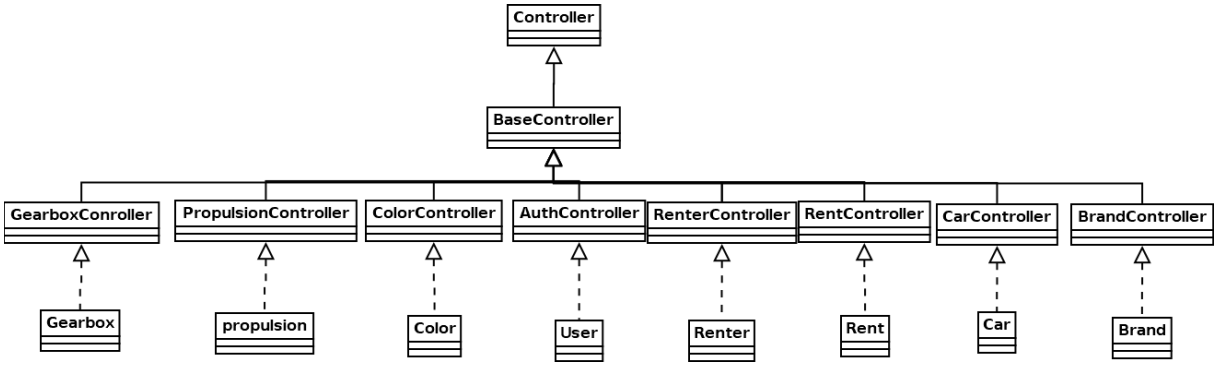
2. A projekt fő mappaszerkezetében található sql fájlt az XAMPP indítása után a böngészőben http://localhost/phpmyadmin/ felületen importáljuk. Ezzel tesztadatok nélkül jön létre az adatbázis

Az laravel által létrehozott táblák:

- migrations
- personal_acces_tokens
- failed_jobs
- password_resets
- users

REST API

UML ábra



Általános működés

A REST API http kéréseket fogad, melyek tartalmazzák a műveletekhez szükséges megfelelő adatokat. A kényes műveletek végpontjai védettek, autentikáció szükséges a használatukhoz. Ilyenek a kijelentkezés, autók módosítása, törlése stb…

Ugyanakkor a regisztrációhoz, bejelentkezéshez, az autók lekérdezéséhez nem kell autentikáció.

Az adatokat Json vagy formátumban fogadja és feldolgozza. A vezérlést kontrollerek valósítják meg, minden adatkezelési csoportnak külön kontrollere van, itt történik az adatfeldolgozás.

A kontrollerek modellekkel vannak kapcsolatban amelyek az adatkezelésért felelősek, itt történik az adatok adatbázisból kiolvasása, illetve a az adatok kiírása adatbázisba.

A modellek adatbázis táblákkal vannak kapcsolatban, melyek az adatok tárolásáért felelősek. Az adatbázis táblák adatait a modellek kezelik.

Végpont neve	Metódus típusa	Végpont eléréséhez szükséges Autentikáció?	Végpont leírása
/api/login	POST	Nem	Bejelentkezés
/api/register	POST	Nem	Regisztráció
/api/logout	POST	Igen	Kijelentkezés
/api/cars	GET	Nem	Autók lekérdezése
/api/cars/{id}	GET	Nem	Az adott autó lekérdezése

/api/cars	POST	Igen	Új autó felvétele
/api/cars/{id}	PUT	Igen	Az adott autó adatainak módosítása
/api/cars/{id}	DELETE	Igen	Az adott autó törlése
/api/renters	GET	Nem	Kölcsönzök lekérdezése
/api/renters/{id}	GET	Nem	Az adott kölcsönző lekérdezése
/api/renters	POST	Igen	Új kölcsönző felvétele
/api/renters/{id}	PUT	Igen	Az adott kölcsönző adatainak módosítása
/api/renters/{id}	DELETE	Igen	Az adott kölcsönző törlése
/api/rents	GET	Nem	Kölcsönzések lekérdezése
/api/rents/{id}	GET	Nem	Az adott kölcsönzés lekérdezése
/api/rents	POST	Igen	Új kölcsönzés felvétele
/api/rents/{id}	PUT	Igen	A kölcsönzés adatainak módosítása
/api/rents/{id}	DELETE	Igen	Az adott kölcsönzés törlése
/api/brands	GET	Nem	Márkák lekérdezése
/api/brands/{id}	GET	Nem	Az adott márka lekérdezése
/api/brands	POST	Igen	Új márka felvétele
/api/brands/{id}	PUT	Igen	Az adott márka adatainak módosítása
/api/brands/{id}	DELETE	Igen	Az adott márka törlése
/api/colors	GET	Nem	Színek lekérdezése
/api/colors/{id}	GET	Nem	Az adott szín lekérdezése
/api/colors	POST	Igen	Új szín felvétele
/api/colors/{id}	PUT	Igen	A szín adatainak módosítása
/api/r/{id}	DELETE	Igen	Az adott szín törlése
/api/gearboxes	GET	Nem	Váltó típusok lekérdezése
/api/gearboxes/{id}	GET	Nem	Az adott váltó típus lekérdezése
/api/propulsions	GET	Nem	Meghajtási típusok lekérdezése
/api/propulsions/{id}	GET	Nem	Az adott meghajtás típus

			lekérdezése

Indítás

Indításhoz először fútassa a composer install parancsot majd végezze el az adatbázis létrehozásának első módszerét.

Az API beüzemelése után az env.example fájl másolása és env-re való átnevezése után a DB_DATABASE mezőnek megkell adni egy tetszőleges nevet. Majd az XAMPP indítása után a böngészőbe a http://localhost/phpmyadmin/ felületen a megadott névvel létrehozni egy táblát majd az API-ban futtatni a php artisan migrate parancsot. Ezzel létrejönnek a megtervezett táblák és a laravel saját táblái, és a projekt működésének tesztelése érdekében kreált tesztadatok.

Osztályok

BaseController

Feladata a műveletek válaszainak küldése. Sikeres művelet esetén az adatot és üzenetet visszaküldi, sikertelen művelet esetén hibaüzenetet küld.

sendResponse()

- bejövő paraméterek: $result, $message (A felhasználónak szánt saját üzenet)
- kimenő adatok: Response (saját üzenet).

sendError()

-bejövő paraméterek: $error (A php által generált hibaüzenet), $errorMessages (Saját hibaüzenet), $code (A válaszban küldendő http kód).

-kimenő adatok: Response (hibaüzenet és a hiba kódja).

AuthController

Feladata egy új felhasználó felvétele, felhasználók autentikációja, felhasználók kijelentkeztetése..

register()

- bejövő paraméterek: $request (az regisztrációhoz szükséges adatok a kérésben, name, email, password, confirmed_password)

Az adatokat validálja, majd sikeres érvényesítés után bejegyzi az users adatbázis tábla megfelelő mezőibe (a jelszavakat titkosítva).

- kimenő adatok: üzenet

login()

Feladata a felhasználó azonosítása név és jelszó alapján. Sikeres autentikáció esetén generál egy tokent a felhasználó számára és bejegyzi a personal_access_tokens adatbázis tábla megfelelő mezőjébe, majd átadja a BaseController sendResponse() metódusának a saját üzenettel együtt.

- bejövő paraméterek: $request (a bejelentkezéshez szükséges adatok a kérésben, name, password).

- kimenő adatok: token + üzenet

logout()

Feladata a felhasználó kijelentkeztetése és a token törlése az adatbázis táblából.

- bejövő paraméterek: $request (a kijelentkezéshez szükségek adatok a kérésben, token)

- kimenő adatok: üzenet

BrandController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Brand model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Brand model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

Create()

-bejövő paraméter: $request.

Validálás után meghívja egy változóban a Brand modell create() metódusát, majd átadja neki az $regquest->all() kérést majd visszatér a általunk megadott adatokkal és az egyedi üzenettel

Update()

-bejövő paraméter: $request,$id.

Meghívja egy változóban a Brand model find() metódusát, majd átadja neki az $id-t. Miután megtalálta a keresett id-t lefuttatja rajta az update() metódust aminek átadja a request->all() kérést és lefrissíti a módosított adatokkal

Delete()

-bejövő paraméter: $id.

Meghívja egy változóban a Brand model destroy() metódusát, majd átadja neki az $id-t, majd visszatár a változóval ami törölte a megadott id-val rendelkező mezőt, és visszakapjuk az egyedi üzenetet

CarController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Car model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Car model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

Create()

-bejövő paraméter: $request.

Validálás után meghívja egy változóban a Car modell create() metódusát, majd átadja neki az $regquest->all() kérést majd visszatér a általunk megadott adatokkal és az egyedi üzenettel

Update()

-bejövő paraméter: $request,$id.

Meghívja egy változóban a Car model find() metódusát, majd átadja neki az $id-t. Miután megtalálta a keresett id-t lefuttatja rajta az update() metódust aminek átadja a request->all() kérést és lefrissíti a módosított adatokkal

Delete()

-bejövő paraméter: $id.

Meghívja egy változóban a Car model destroy() metódusát, majd átadja neki az $id-t, majd visszatár a változóval ami törölte a megadott id-val rendelkező mezőt, és visszakapjuk az egyedi üzenetet

ColorController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Color model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Color model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

Create()

-bejövő paraméter: $request.

Validálás után meghívja egy változóban a Color modell create() metódusát, majd átadja neki az $regquest->all() kérést majd visszatér a általunk megadott adatokkal és az egyedi üzenettel

Update()

-bejövő paraméter: $request,$id.

Meghívja egy változóban a Color model find() metódusát, majd átadja neki az $id-t. Miután megtalálta a keresett id-t lefuttatja rajta az update() metódust aminek átadja a request->all() kérést és lefrissíti a módosított adatokkal.

Delete()

-bejövő paraméter: $id.

Meghívja egy változóban a Color model destroy() metódusát, majd átadja neki az $id-t, majd visszatár a változóval ami törölte a megadott id-val rendelkező mezőt, és visszakapjuk az egyedi üzenetet.

PropulsionController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Propulsion model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Propulsion model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

GearboxController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Gearbox model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Gearbox model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

RentController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Rent model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Rent model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

Create()

-bejövő paraméter: $request.

Validálás után meghívja egy változóban a Rent modell create() metódusát, majd átadja neki az $regquest->all() kérést majd visszatér a általunk megadott adatokkal és az egyedi üzenettel

Update()

-bejövő paraméter: $request,$id.

Meghívja egy változóban a Rent model find() metódusát, majd átadja neki az $id-t. Miután megtalálta a keresett id-t lefuttatja rajta az update() metódust aminek átadja a request->all() kérést és lefrissíti a módosított adatokkal

Delete()

-bejövő paraméter: $id.

Meghívja egy változóban a Rent model destroy() metódusát, majd átadja neki az $id-t, majd visszatár a változóval ami törölte a megadott id-val rendelkező mezőt, és visszakapjuk az egyedi üzenetet.

RenterController

index()

-bejövő paraméter: nincs.

Meghívja egy változóban a Renter model all() metódusát, majd visszatér a változó adatával és az egyedi üzenettel

Show()

-bejövő paraméter: $id.

Meghívja egy változóban a Renter model find() metódusát, majd átadja neki az $id-t, utána visszatér a változó adatával és az egyedi üzenettel

Create()

-bejövő paraméter: $request.

Validálás után meghívja egy változóban a Renter modell create() metódusát, majd átadja neki az $regquest->all() kérést majd visszatér a általunk megadott adatokkal és az egyedi üzenettel

Update()

-bejövő paraméter: $request,$id.

Meghívja egy változóban a Renter model find() metódusát, majd átadja neki az $id-t. Miután megtalálta a keresett id-t lefuttatja rajta az update() metódust aminek átadja a request->all() kérést és lefrissíti a módosított adatokkal

Delete()

-bejövő paraméter: $id.

Meghívja egy változóban a Renter model destroy() metódusát, majd átadja neki az $id-t, majd visszatár a változóval ami törölte a megadott id-val rendelkező mezőt, és visszakapjuk az egyedi üzenetet.

Modellek

- Brand:

  Ez a model a brands táblával van kapcsolatban

- Car:

  Ez a model a cars táblával van kapcsolatban

- Color:

  Ez a model a colors táblával van kapcsolatban

- Gearbox:

  Ez a model a gearboxes táblával van kapcsolatban

- Propulsion:

  Ez a model a propulsions táblával van kapcsolatban

- Rent:

  Ez a model a rents táblával van kapcsolatban

- Renter:

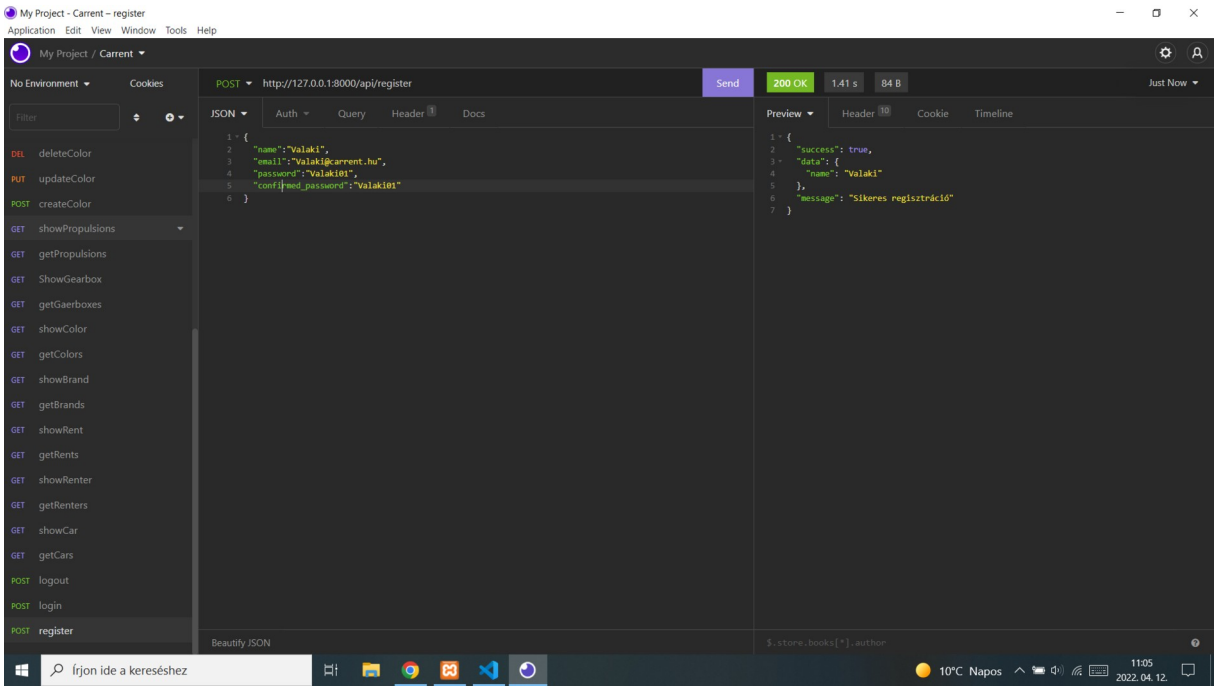
  Ez a model a renters táblával van kapcsolatban

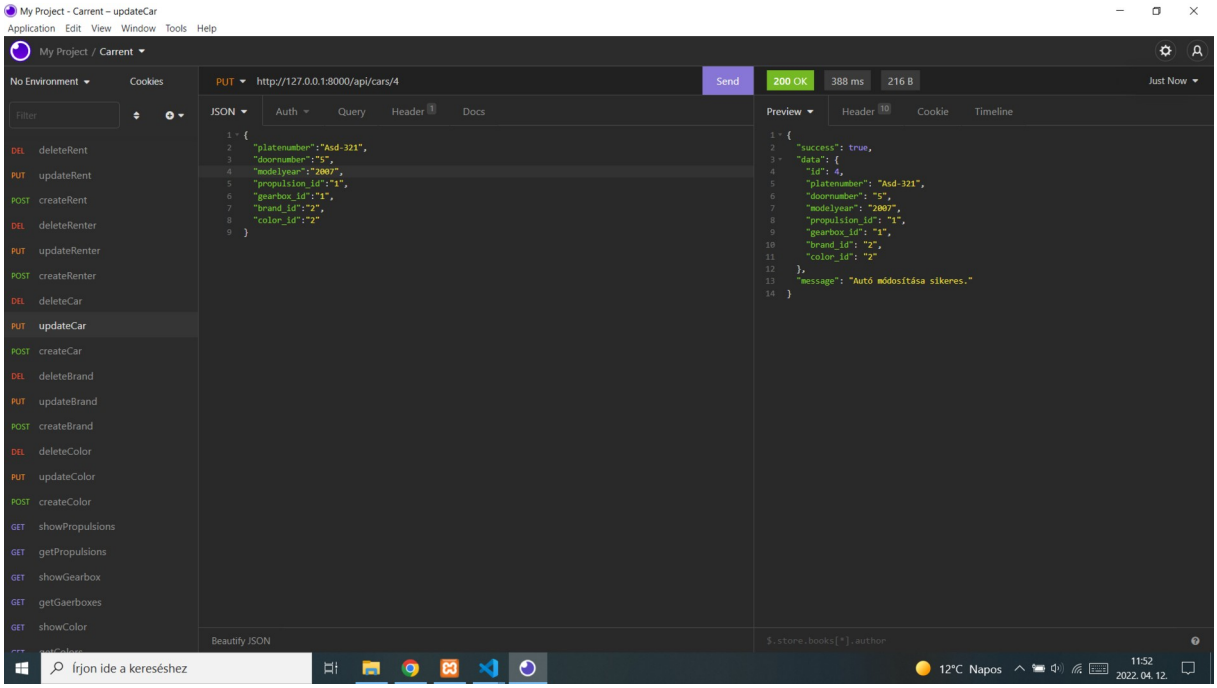
- User:

  Ez a model a users táblával van kapcsolatban

Api Tesztelése

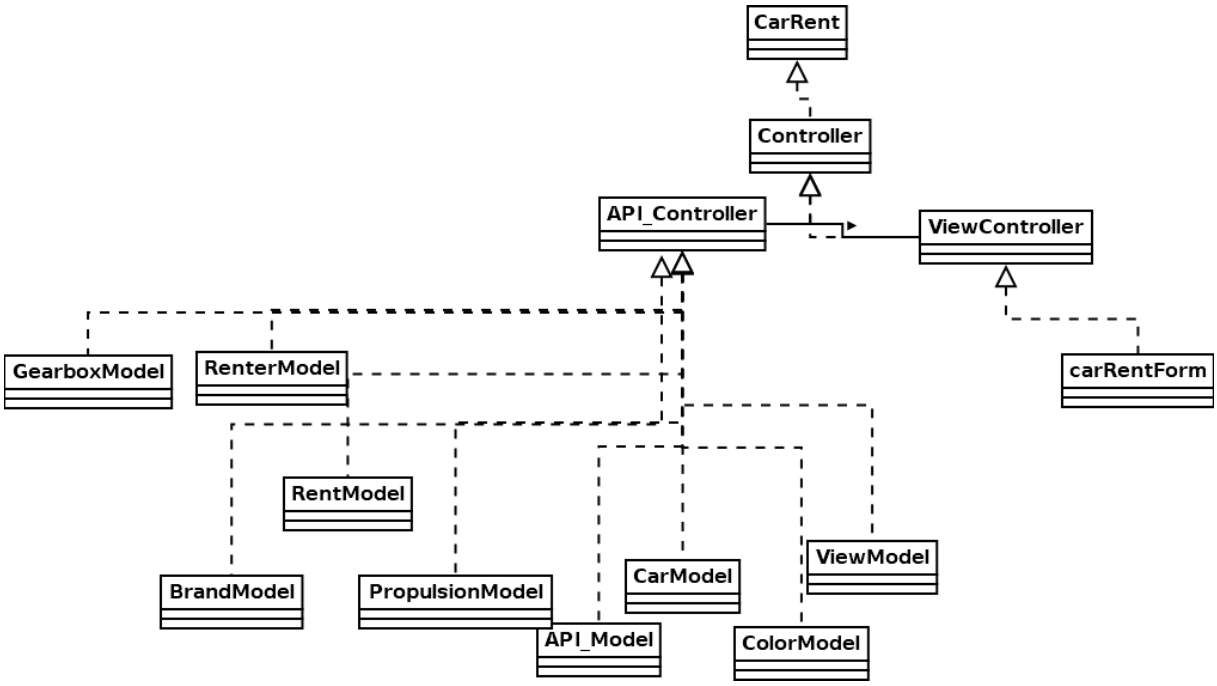
A REST API tesztelése Insomnia használatával történt és minden útvonalhoz egy teszt készült. Összesen 30 teszt készült amelyek az api mappában lévő tesztek mappában találhatoak





Asztali Alkalmazás

UML terv



Általános működés

Az asztali alkalmazás nem lesz publikus formában elérhető, ezért nincsen bejelentkező felület hanem az alkalmazás indítás után egyből belép egy API-tól kapott felhasználóval ami jelen esetben Roland. A program az adatbázis táblák adatait jeleníti meg, A programot egy asztali admin felület céljából kezdtem el csinálni ahol lehet adatokat törölni, módosítani és új adatokat felvenni. Azonban csak az adatok megjelenítéséig illetve a törlés funkció elkészítéséig sikerült eljutnom.

Az asztali alkalmazás használatához szükséges futó API mert az asztali alkalmazás az API-tól szerzi az adatokat, illetve a kéréseket

Indítás

Az alkalmazás indításához netbeans 12.6-os verzió szükséges.

Az alkalmazás java jdk 17-es verzióval működik, és a működéséhez szükséges még a gson-2.8.2 jar fájl beemelése netbeansban a libraries mappába, a gson-2.8.2 megtalálható a Library könyvtárban

Osztályok

Carrent

Ez az alkalmazás fő osztálya, itt példányosítja meg a Controller osztályt

Controller

Megpéldányosítja az API_Controllert és a ViewControllert, valamint az ViewControllernek átadjuk az API_Controller példányát

APIController

Funkciója: Az API_Model-töl átvett adatokat továbbítja a ViewController felé.

Osztályváltózói:

- APIModel APIMdl
- String token
- id

API_Controller()

Megpéldányosítja az API_Model osztályt

Elmenti a tokent sikeres belépés esetén

Létrehozza az id változót üresen

getToken()

Elmenti a API_Modeltől belépés után kapott tokent

Logout()

Meghívja a API_Model tryLogout függvényét és átadja neki a tokent

getRenters()

Meghívja a API_Model tryRenters nevű függvényét melynek adatait egy vectorban eltárolja

getRents()

Meghívja a API_Model tryRents nevű függvényét melynek adatait egy vectorban eltárolja

getCars()

Meghívja a API_Model tryCars nevű függvényét melynek adatait egy vectorban eltárolja

getBrands()

Meghívja a API_Model tryBrands nevű függvényét melynek adatait egy vectorban eltárolja

getColors()

Meghívja a API_Model tryColors nevű függvényét melynek adatait egy vectorban eltárolja

getGearboxes()

Meghívja a API_Model tryGearboxes nevű függvényét melynek adatait egy vectorban eltárolja

getPropulsions()

Meghívja a API_Model tryPropulsions nevű függvényét melynek adatait egy vectorban eltárolja

DeleteRenter()
Meghívja a API_Model tryDeleteRenters nevű függvényét és átadja neki a token és az
id változót

DeleteRent()
Meghívja a API_Model tryDeleteRents nevű függvényét és átadja neki a token és az id változót

DeleteCar()
Meghívja a API_Model tryDeleteCars nevű függvényét és átadja neki a token és az id változót

DeleteBrand()
Meghívja a API_Model tryDeleteBrands nevű függvényét és átadja neki a token és az
id változót

DeleteColor()
Meghívja a API_Model tryDeleteColors nevű függvényét és átadja neki a token és az id változót

setId()
Beállítja, hogy az osztályváltozó id egyenlő a paraméterként érkezett id-val

ViewController

Funkciója: Az ablakban minden funkció működjön és minden szükséges adat megjelenjen

Osztályváltozók:

- ViewModel viewMdl;
- API_Controller APICtr;
- carRentForm carFrm;
- Vector<Vector<Object>> tableData

ViewController()

- bejövő paraméterek: APICtr

Beállítja, hogy az osztályváltozó APICtr egyenlő a paraméterként érkezett APICtrvel

Megpéldányosítja a ViewModel osztályt

Elindítja a program elindulásához nélkülözhetetlen függvényeket

InitListeners()

Ráköti az ActionListener-eket a szükséges komponensekre

initTables()

Feltölti a táblázatokat adatokkal

exit()

Kitörli a tokent és bezárja a programot

delete()

Megvizsgálja, hogy melyik tábla van megnyitva és attól függően elindítja a

megfelelő törlési függvényt

deleteRenters()

Kitörli a kiválasztott kölcsönzőt, aztán utána újra

elindítja az adatok táblába töltését

deleteRents()

Kitörli a kiválasztott kölcsönzést, aztán utána újra

elindítja az adatok táblába töltését

deleteCars()

Kitörli a kiválasztott autót, aztán utána újra

elindítja az adatok táblába töltését

deleteBrands()

Kitörli a kiválasztott márkát, aztán utána újra

elindítja az adatok táblába töltését

deleteColors()

Kitörli a kiválasztott színt, aztán utána újra

elindítja az adatok táblába töltését

API_Model

Funkciója: Kommunikálás a API-val, adatok fogadása és küldése

tryLogin()
Megpróbálja lefuttatni a Login függvényt
- kimenő adatok: token

Login()
Bejelentkezik az alkalmazásba és kiolvassa a tokent

- kimenő adatok: token

tryLogout()

- bejövő paraméterek: token
Megpróbálja lefuttatni a Logout függvényt

Logout()
- bejövő paraméterek: token
Visszaadja a tokent a Backendnek

tryRenters

Megpróbálja lefuttatni a Renters függvényt

Renters

Lefuttatja a kérést a Backendnek és kiolvassa a kölcsönzök adatait

tryRents

Megpróbálja lefuttatni a Rents függvényt

Rents

Lefuttatja a kérést a Backendnek és kiolvassa a kölcsönzések adatait

tryCars

Megpróbálja lefuttatni a Cars függvényt

Cars

Lefuttatja a kérést a Backendnek és kiolvassa az autók adatait

tryBrands

Megpróbálja lefuttatni a Brands függvényt

Brands

Lefuttatja a kérést a Backendnek és kiolvassa a márkák adatait

tryColors

Megpróbálja lefuttatni a Colors függvényt

Colors

Lefuttatja a kérést a Backendnek és kiolvassa a Szinek adatait

tryGearbxes

Megpróbálja lefuttatni a Renters függvényt

Gearboxes

Lefuttatja a kérést a Backendnek és kiolvassa a váltó típusok adatait

tryPropulsions

Megpróbálja lefuttatni a Propulsions függvényt

Propulsions

Lefuttatja a kérést a Backendnek és kiolvassa a meghajtási típusok adatait

tryDeleteRenters()

- bejövő paraméterek: token, id

Megpróbálja lefuttatni a DeleteRenters függvényt

DeleteRenters()

- bejövő paraméterek: token, id

Elküldi az id-t a Backendnek és törli a kölcsönzőt és a hozzá tartozó adatokat

tryDeleteRents()

- bejövő paraméterek: token, id

Megpróbálja lefuttatni a DeleteRents függvényt

DeleteRents()

- bejövő paraméterek: token, id

Elküldi az id-t a Backendnek és törli a kölcsönzést és a hozzá tartozó adatokat

tryDeleteCars()

- bejövő paraméterek: token, id

Megpróbálja lefuttatni a DeleteCars függvényt

DeleteCars()

- bejövő paraméterek: token, id

Elküldi az id-t a Backendnek és törli a autót és a hozzá tartozó hirdetéseket

tryDeleteBrands()

- bejövő paraméterek: token, id

Megpróbálja lefuttatni a DeleteBrands függvényt

DeleteBrands()

- bejövő paraméterek: token, id

Elküldi az id-t a Backendnek és törli a felhasználót és a hozzá tartozó hirdetéseket

tryDeleteColors()

- bejövő paraméterek: token, id

Megpróbálja lefuttatni a DeleteColors függvényt

DeleteColors()

- bejövő paraméterek: token, id

Elküldi az id-t a Backendnek és törli a színt

ViewModel

Itt tárolom a táblák oszlopneveit:

GetRenterColumNames

A renter tábla oszlopnevei

GetCarColumNames

A car tábla oszlopnevei

GetRentColumNames

A rent tábla oszlopnevei

GetBrandColumNames

A Brand tábla oszlopnevei

GetColorColumNames

A color tábla oszlopnevei

GetPropulsionColumNames

A propulsion tábla oszlopnevei

GetGearboxColumNames

A gearbox tábla oszlopnevei

CarModel

Ez az osztály sablonként funkciónál az APIModel számára

RenterModel

Ez az osztály sablonként funkciónál az APIModel számára

RentModel

Ez az osztály sablonként funkciónál az APIModel számára

BrandModel

Ez az osztály sablonként funkciónál az APIModel számára

ColorModel

Ez az osztály sablonként funkciónál az APIModel számára

PropulsionModel

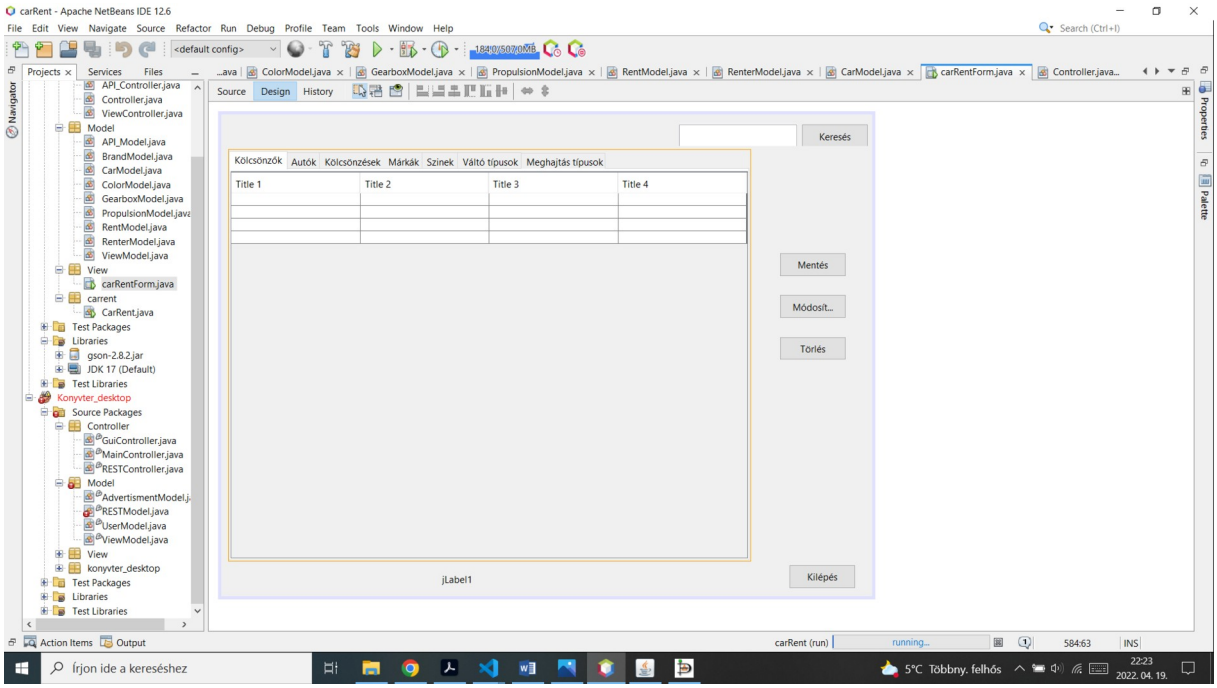
Ez az osztály sablonként funkciónál az APIModel számára

GearBoxModel

Ez az osztály sablonként funkciónál az APIModel számára

carRentForm

A carRentForm a program kinézetéért felelős



Frontend

Általános információ

A teljes Frontend Angular (TypeScript-alapú) keretrendszer használatával készült, ennek megfelelően Single-page applikáció, olyan webalkalmazás, amely interakcióba lép a felhasználóval azáltal, hogy dinamikusan váltogatja az aktuális weboldalt a REST API-ról származó új adatokkal kiegészítve. Emellett az oldal teljes mértékben reszponzív és használata során egy mobil applikáció érzetét kelti, ezt Bootstrap 5 használatával és CSS el sikerült elérni

Indítási parancsok

-npm install

-ng serve -o

Felépítése

| -node_modules/

` -src/

| -app/

```
|       |-components/
|       |       |-cards/
|       |       |       |-cards
|       |       |       |-oldcards
|       |       |       |-newcards
|       |       |-cars/
|       |       |-login/
|       |       |-main/
|       |       |-register/
|       |       |-vehicles/
|       |-models/
|       |-shared/
|       | -app-routing.modules.ts
|       | -app.components.css
|       | -app.component.html
|       | -app.component.spec.ts
|       | -app.components.ts
|       | -app.modules.ts
```

- ➔ cards - saját komponens
- ➔ main/ - saját komponens
- ➔ cars/ - saját komponens
- ➔ login/ - saját komponens
- ➔ register/ - saját komponens
- ➔ vehicles/ - saját komponens
- ➔ Models/- regisztráció model
- ➔ shared/ - auth.service és auth.guard
- ➔ app-routing.modules.ts - Az útválasztási bejegyzések
- ➔ app.components.css - Az alkalmazás CSS beállításai
- ➔ app.components.html - Az alkalmazás fő nézet oldala

- ➔ app.component.spec.ts - Script a teszteléshez
- ➔ app.components.ts - Fő script komponens
- ➔ app.modules.ts - Modulok betöltése
- ➔ index.html - Az alkalmazás induló HTML állománya

Komponensek:

app komponens

app.component.html:

- ➔ nav
- ➔ Router-Outlet

app.routing.module.ts

- ➔ itt találhatóak az útvonalak

app.component.css

- ➔ saját css az app.component.html re vonatkozóan

app.compnent.spec.ts

- ➔ Az app komponens tesztjei

app.component.ts

- ➔ isLogedIn() meghívja az auth.service isLoggedIn metódusát és vissza adja hogy be van lépve a felhasználó
- ➔ logout() meghívja az auth.service logout metódusát ami átnavigál a fő komponensre

app.module.ts

- ➔ modulok betöltése,componensek tárolása

cards komponens

cards.component.html:

- ➔ az összes jármű külön „card” okban megjelenítve

cards.component.css

➔ saját css a cards componensre vonatkozóan

cards.component.spec.ts

➔ tesztek a cards componensre vonatkozóan

cards.component.ts

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

oldcard komponens

oldcard.component.html:

➔ a régi járművek külön „card” okban megjelenítve

oldcard.component.css

➔ saját css a cards componensre vonatkozóan

oldcard.component.spec.ts

➔ tesztek a cards componensre vonatkozóan

oldcard.component.ts

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

newcard komponens

newcard.component.html:

➔ az új járművek külön „card” okban megjelenítve

newcard.component.css

➔ saját css a cards componensre vonatkozóan

newcard.component.spec.ts

➔ tesztek a cards componensre vonatkozóan

newcard.component.ts

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

register komponens

register.component.html:

➔ regisztrációs form validálással

register.component.css

➔ saját css a register componensre vonatkozóan

register.component.spec.ts

➔ tesztek a register komponensre vonatkozóan

register.component.ts

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

login komponens

login.component.html:

➔ bejelentkezés form validálással

login.component.css

➔ saját css a login komponensre vonatkozóan

login.componens.spec.ts

➔ tesztek a login komponensre vonatkozóan

login.component.ts

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

Main komponens

main.component.html:

➔ Carousel

➔ kereső

➔ footer

main.component.css:

➔ saját css a main komponensre vonatkozóan

main.component.spec.ts

➔ saját tesztek a main komponensre vonatkozóan

main.component.ts:

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

vehicle komponens

vehicle.component.html

➔ Új autók felvételéhez form és táblázat

vehicle.component.css

➔ saját css a vehicle komponensre vonatkozóan

vehicle.component.spec.ts

➔ tesztek a main kopmonensre vonatkozóan

vehicle.component.ts

➔ ngOnInit() amit azután hívnak meg amint az angular inicializálta egy direktiva összes tulajdonságát

shared könyvtár

auth.guard.ts

➔ canActivate() meghívja az auth.service isLoggedIn() metódusát, hogyha true akkor visszatér vele, ha false akkor átnavigál a főoldalra és visszaadja a false értéket

auth.service.ts:

Az autentikációval kapcsolatos metódusokat tartalmazza

➔ register() elküldi a regisztrációs adatokat a REST API-nak és visszatér a válasszal

➔ login() elküldi a bejelentkezési adatokat a REST API-nak és visszatér a válasszal

➔ logout() törli a tokent a lokális tárolóból és REST API-n keresztül is

➔ isLoggedIn() megnézi, hogy be van-e jelentkezve a felhasználó az által, hogy a lokális tárolóban van-e token, ha igen vissza tér vele, ha nem false értékkel tér vissza