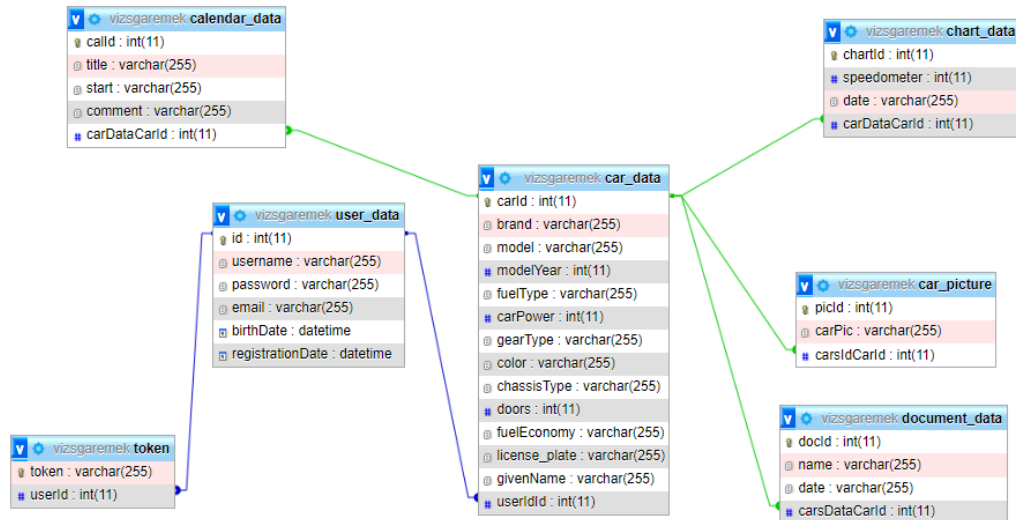


Fejlesztői dokumentáció

Frontend – Backend – Zuber Marcell

- **Adatbázis**



Az adatbázis legfontosabb adattáblája a “UserData”, mely a felhasználó regisztrációkor megadott adatait tárolja. Elsődleges kulcsa az “id” ami meghatározó szerepet játszik a program működésében.

A “token” adattábla (OneToMany) tárolja a backend által generált egyedi token- t, amely bejelentkezésnél kerül generálásra illetve kijelentkezésnél törlődik. Mellé minden esetben csatol egy felhasználói azonosítót, ami az éppen bejelentkezett felhasználót azonosítja. Az alkalmazások a bejelentkezéskor ezt az azonosítót kapják meg a szervertől, majd mentik el a programban, hogy ez alapján a POST típusú kéréseket megfelelő felhasználóhoz tudják csatolni.

A “carData” adattáblában(ManyToOne) tároljuk a felvett autók adatait. Itt a kulcs a “carId” és ez sokszor visszaköszön más táblák idegen kulcsaként is majd a későbbiekben. A tábla idegen kulcsa a “userId”, hogy össze lehessen csatolni az autó adatait a bejelentkezett felhasználóval.

A “carData” - hoz legközelebb álló tábla a “carPicture” (ManyToOne), amely az autóról feltöltött képek nevét és kiterjesztését tárolja, itt az idegen kulcs már a “carId” mivel a kép az autóhoz kapcsolódik.

A “chartData” tábla (ManyToOne) tárolja a diagrammra felvett értékeket, illetve a felvétel dátumát. Itt is szintén mivel az autóhoz kapcsolódik a diagramm, az idegen kulcs itt is a “carId”.

A “calendarData” adattábla (ManyToOne) tárolja a naptárban felvett eseményeket. Ez a tábla is szintén az autóhoz kapcsolódik ezért az idegen kulcsa ennek is a “carId”. Ebben a táblában van eltárolva az esemény neve, dátuma, illetve a további szöveges információ az eseményhez melynek megadása opcionális.

Az utolsó adattábla a “documentData” (ManyToOne) névre hallgat, itt tárolódnak a dokumentumok, amely szintén az autóhoz kapcsolódik, így itt sem meglepő idegen kulcsként a “carId”. Itt a dokumentum neve, illetve a lejárat dátuma van eltárolva, a program majd az utóbbiból fogja kiszámolni, hogy hány nap maradt még a lejáratig.

- **Navigációs sáv**



A navigációs sáv két formában jelenik meg, az egyik a képen is látható a másik formában viszont nem jelenik meg a Garázs és naptár menüpont. Ezek csak bejelentkezés után lesznek elérhetőek, hogy a regisztrálatlan felhasználó ne tudjon új autót hozzáadni fiók nélkül. A Naptár felület mivel az is szintén az autóhoz kapcsolódik ezért az is eltüntetésre kerül amíg nincs bejelentkezve a felhasználó.

- **Regisztráció**

Regisztráció

Felhasználónév

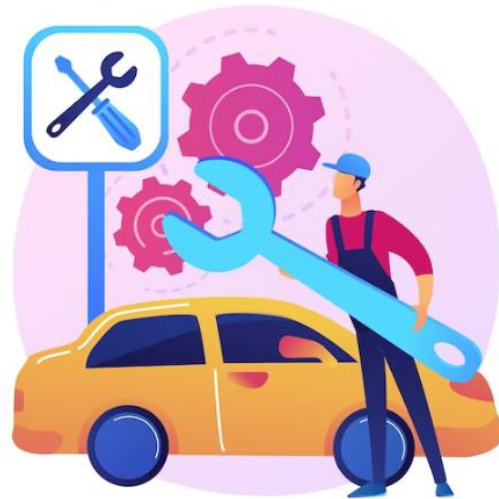
Email

Jelszó

Jelszó újra

Születési dátum

☐ Elfogadom a felhasználói feltételeket ([Terms of service](#))



A regisztráció úgy lett megírva, hogy jelezze a felhasználónak, hogy ha helyes adatot ad meg egy zöld pipa jelenik meg a mező alatt, ebben az esetben éri meg tovább haladni a következő mezőre.

A felhasználónév mezőre a következő "regular expression" érvényes:

```
const usernameReg = /^[a-zA-Z0-9]{3,20}$/;
```

Lényegében ez annyit takar, hogy 3-20 karakterig megadhatóak az angol ábécé bármely betűi, illetve természetes számok 0-9 ig.

Az email szintén csak érvényes formátumban adható meg.

```
const emailReg = /^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$/;
```

A jelszónál az egyetlen kritérium, hogy minimum 6 karakter hosszúságúnak kell lennie. A jelszó elküldéskor titkosításon esik át a szerveren, amely az adatbázisban a titkosított formában tárolja el a jelszót a "bcrypt" segítségével.

A regisztráció gombra kattintva lefut egy validáció mely ellenőrzi, hogy a megadott mezők helyesen vannak e megadva és ha mindent rendben talál végrehajtja az adatok feltöltését az adatbázisba, majd átdobja a felhasználót a bejelentkezés oldalra.

- **Login**



Belépés

Felhasználónév

Jelszó

Üdvözöljük!

Bejelentkezés

Nincs fiókja? [Regisztrálok](#)

Bejelentkezésnél a felhasználó megadja a felhasználónevét, illetve a hozzá kapcsolódó jelszót. A jelszó ugyan titkosítva van az adatbázisban, de a login során a "bcrypt.compare" segítségével a szerver össze tudja hasonlítani a 2 megadott jelszót.

A bejelentkezés gombra kattintva a regisztrációhoz hasonlóan, lefut egy validáció mely ellenőrzi az adatok egyezését az adatbázissal, majd, ha mindent rendben talál, generál egy token a "tokenData" táblában, amely tartalmazza majd a felhasználói azonosítót is. Ezt a 2 adatot elküldi a szerver és a böngésző helyi tárolójában eltároljuk.

- **Autó felvétele**

+ Autó hozzáadása

Autó neve*

Írja be az autója nevét

Autó márkája*

Írja be az autó márkáját

Autó típusa*

Írja be az autó típusát

Évjárat

0

Üzemanyag típusa*

Írja be az autó üzemanyagának a típusát

Lóerő*

0

Váltó típusa*

Írja be az autó váltójának típusát

Az autó felvétele gombra kattintva egy bootstrap modal ugik fel melyben egy form helyezkedik el, ahol az autót lehet feltölteni. Az adatok megadása teljesen a felhasználó kedvére van megírva, limitálások nélkül. Bootstrap –es “form-control” ellenőrzi a kötelező mezők állapotát, a számos mezőkbe csak szám kerülhet. A felvételre kattintva feltöltődik az autó a megadott adatokkal.

- **Kép feltöltése**

Töltsön fel autójáról egy képet!

Fájl kiválasztása

chaser.jpg

Feltöltés!



proba
Márka: *proba*
Modell: *proba*
Évjárat: 10
Üzemanyag típusa: *proba*
Lóerő: 10
Váltó típusa: *proba*
Rendszám: *proba*

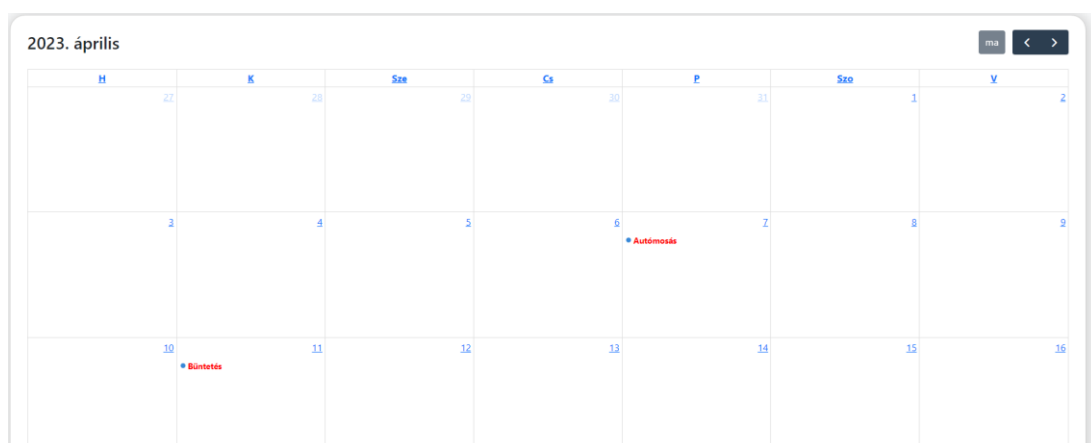
A kép feltöltésére 2 végpont is fel van véve az egyik a feltöltés utáni azonnali megjelenítésre szolgál, a másik adatbázisból tölti be a képet az oldal megnyitásakor. Itt is csatolódik egy "carId" a képhez az adatbázisban, hogy beazonosítható legyen melyik autóhoz kapcsolódik.

- **Diagramm**



A diagramm egy komponensként van megírva, a "Calendar" oldalon csak egy "data" és "labels" tömböt vesz át melyek tartalmazzák a kilométer óra jelenlegi állását, illetve az éppen akkori dátumot. A szerver dátum szerint növekvő sorrendben küldi vissza az adatokat, úgyhogy, ha esetleg visszamenőleg akarjuk felvenni a kilométer óra állását, az a megfelelő helyen fog megjelenni. A diagrammhoz chart.js-t használtunk, azon belül is egy szimpla egy vonalas "line chart" -ot.

- **Naptár**



Felvett események	
Autómosás: 3000Ft 2023. 04. 07.	törlés
Büntetés: Parkolás, 5000Ft 2023. 04. 11.	törlés
Pályamatrixa: 1 hónapos matrica megújítása, 10000Ft 2023. 04. 26.	törlés
Büntetés: Befizetési határidő vége, 80000Ft 2023. 05. 05.	törlés

A program legfontosabb oldala, erre alapul az autókezelő programunk lényegében. Egy “fullcalendar” naptár nézete tölti ki a képernyő nagy részét, celláiban az aznapon bekövetkező eseményeket írja ki jól észrevehető színnel. Az esemény felvétele igen magától értetődő, a cellába kattintva hozzá tudjuk adni az eseményt mely ott meg is jelenik majd. A naptár alatt “Felvett események” néven kapunk egy listát az eseményekről. Itt teljesen dátum szerint vannak rendezve, illetve törölni is lehet bármelyiket. Abban az esetben, ha a “mai napon” történik az esemény pirosan Mai napon felirat jelenik meg ehhez a program felhasználja a mai dátumot és összehasonlítja a megadottal.

- **Dokumentumok**

Dokumentumok

Kötelező biztosítás2023. 04. 19.

Lejár: Már lejárt.

Törlés

Forgalmi engedély2023. 04. 26.

Lejár: Már lejárt.

Törlés

Jogosítvány2025. 07. 26.

Lejár: 818 nap múlva

Törlés

+ Dokumentum hozzáadása

Dokumentum neve:

Adja meg a dokumentum nevét!

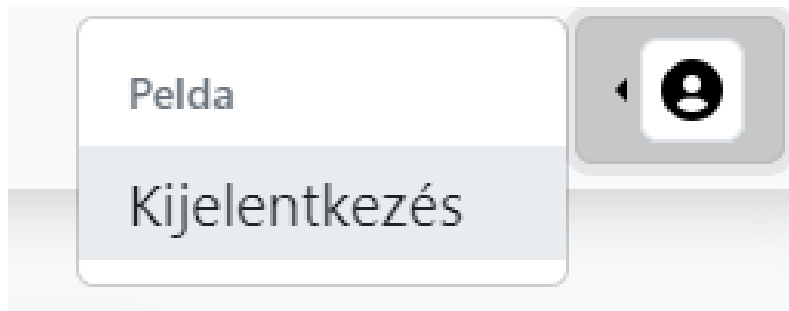
Lejárati dátum:

2023. 04. 30.

Hozzáadás

A dokumentumok megadása igen egyszerű, egy lejárat dátum, illetve egy név kerül megadásra. A frontend részen kerül kiszámolásra a hátralévő napszám, illetve a szöveg színek beállítása is if-else elágazásokkal megoldva.

- Kijelentkezés



A kijelentkezés gombra kattintva az adatbázisból törlődik a felhasználó tokenje a "tokenData" adattáblából, illetve a főoldalra irányítja a felhasználót és beállítja az "isLoggedIn" logikai típusú változó értékét hamisra, amely a navigációs sávból a garázs, illetve naptár opciók eltüntetését jelenti. A helyi tárolóból is kitörlődik minden felhasználóval kapcsolatos ott eddig ott tárolt adat.

Asztali alkalmazás - Bóta László Levente, Pfeffer Botond



- Bejelentkezési felület

A program indításakor a felhasználónév és jelszó TextField fogad minket, amiket kitöltve és a bejelentkezés gombot megnyomva küld egy bejelentkezési kérést a backend-nek a megadott adatokkal. Ha sikertelen a bejelentkezés azt a bejelentkezési felület egy labellel jelzi. Viszont ha sikeres akkor lekéri és eltárolja a felhasználó id-ját és Tokenjét. Bejelentkezés után a felhasználó id-ját felhasználva lekérdezi hogy van e felvéve autója, ha van egyből az AutóAdatok oldalra megy, ha pedig nincs akkor az AutoFelvétel oldalra megy. Ezen kívül ezen az oldalon még 2 másik gomb van, a Regisztráció amely megnyitja a frontend regisztrációs formját, illetve a Kilépés amely bezárja a programot.

```
public void validateLogin() throws IOException {
    try {
        LoginDTO loginDTO = new LoginDTO(usernameTextField.getText(), enterpasswordField.getText());
        HttpResponse<JsonNode> response = Unirest.post( url, "http://localhost:3001/auth/login")
            .header( "Content-Type",  is "application/json")
            .body(loginDTO).asJson();
        int status = response.getStatus();
        if (status == 201) {
            JSONObject responseBody = response.getBody().getObject();
            userId = responseBody.getString( key, "userId");
            token = responseBody.getString( key, "token");
            LoginResponse loginResponse = new LoginResponse(Integer.parseInt(userId), token);
            loginModell = new LoginModell(loginResponse);
            loginMessageLabel.setText("Succesful Login!");
            Stage stage = (Stage) loginButton.getScene().getWindow();
            stage.close();

            URL url = new URL( spec, "http://localhost:3001/userCar/" + userId);
            HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
            httpURLConnection.setRequestMethod("GET");
            if (httpURLConnection.getResponseCode() == 200) {
                InputStream inputStream = httpURLConnection.getInputStream();
                BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
                StringBuilder response2 = new StringBuilder();
                String line;
                while ((line = bufferedReader.readLine()) != null) {
                    response2.append(line);
                }
                bufferedReader.close();
                responseString = response2.toString();
            } else {
                responseString = "error";
            }

            //ha van auto akkor egybol az adatait mutatja, ha nincs felkell venni
            if (responseString.length() == 13) {
                //ez azért 13 mert a {"cars":null} 13 betű
                FXMLLoader fxmLoader = new FXMLLoader(App.class.getResource( name, "CarData.fxml"));
                Scene scene = new Scene(fxmLoader.load(),  w 1079,  ht 898);
                Stage stage2 = new Stage();
                stage2.initStyle(StageStyle.UNDECORATED);
                stage2.setTitle("TeAutód.hu");
                stage2.setScene(scene);
                ((CarDataController) fxmLoader.getController()).setLoginForCarData(loginModell);
            }
        }
    }
}
```

- Autó Felvétel


Autó feltöltése

Autónak adott név

Autó márkája

Model

Évjárat

Üzemanyag

Lóerő

Válto típusa

Szín

Kaszni

Ajtók száma

Fogyasztás

Rendszám

Bezárás

Feltöltés

Felvett autó nélküli felhasználó bejelentkezésekor ez a felület jön be, ahol meg kell adni az autó adott paramétereit a TextFielddek segítségével. Az a legtöbb adat VarChar(255), az évjáraton, lóerőn, és az ajtók számán kívül ami int. Feltöltés gombra kattintva feltölti az autót a bejelentkezett felhasználó id-jához kapcsolva és tovább dob az AutóAdatok oldalra. A bezárás gomb bezárja a programot.

```

1 package com.example.myapplication;
2
3 public void UploadButtonOnAction(ActionEvent event) throws IOException {
4     int modelYear = Integer.parseInt(ModelYearLabel.getText());
5     int carPower = Integer.parseInt(CarPowerLabel.getText());
6     int doors = Integer.parseInt(DoorsLabel.getText());
7
8     CarDataDTO carDataDTO = new CarDataDTO(CarNameLabel.getText(), CarBrandLabel.getText(), ModelLabel.getText(),
9         modelYear, FuelLabel.getText(), carPower, GearTypeLabel.getText(),
10        ColorLabel.getText(), ChassisTypeLabel.getText(), doors, FuelEconomyLabel.getText(), LicencePlateLabel.getText());
11
12     int status = Unirest.post("http://localhost:3001/car/" + loginModel.getLoginResponse().getId())
13         .header("Content-Type", "application/json")
14         .body(carDataDTO.toJson().getStatus());
15
16     successfulUploadLabel.setText("Autó feltöltése sikeres!");
17
18     Stage stage = (Stage) UploadBut.getScene().getWindow();
19     stage.close();
20
21     FXMLLoader fxmlLoader = new FXMLLoader(App.class.getResource("CarDataList.fxml"));
22     Scene scene = new Scene(fxmlLoader.load(), 612, 650);
23     Stage stage3 = new Stage();
24     stage3.initStyle(StageStyle.UNDECORATED);
25     stage3.setTitle("AutóAdatok");
26     ((CarDataListController) fxmlLoader.getController()).setLoginModelForCarDataList(loginModel);
27     stage3.setScene(scene);
28     stage3.show();
29 }

```

- Autó Adatok




Előzőleg bejelentkezett és eltárolt user_ID alapján lekérdezi a backendről az autó adatait és kiírja labeleken. A közelgő eseményeket ugyanígy kérdezzük csak ott egy ListViewba írjuk ki. Az eseményeket lehet törölni gomb-bal illetve hozzáadni. A Kilépés gomb bezárja a programot.

```

1 usage  ▸ ListView
2
3 public String carload() throws IOException {
4     URL url = new URL("http://localhost:3001/userCar/" + loginModel.getLoginResponse().getId());
5     HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
6     httpURLConnection.setRequestMethod("GET");
7     String responseString = "";
8
9     if (httpURLConnection.getResponseCode() == 200) {
10        InputStream inputStream = httpURLConnection.getInputStream();
11        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));
12        StringBuilder response = new StringBuilder();
13        String line;
14        while ((line = bufferedReader.readLine()) != null) {
15            response.append(line);
16        }
17        bufferedReader.close();
18        responseString = response.toString();
19    } else {
20        responseString = "error";
21    }
22    return responseString;
23 }

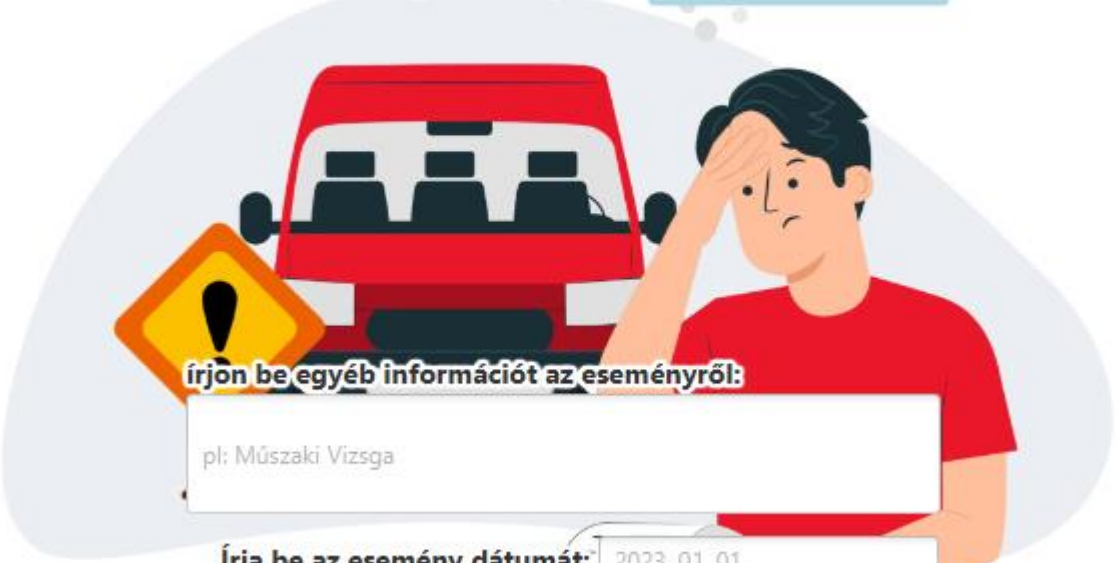
```


- Esemény Felvétel



Események

Válassza ki a közelgő eseményeket: Válasszon!



írjon be egyéb információt az eseményről:

pl: Műszaki Vizsga

Írja be az esemény dátumát: 2023.01.01.

Helyes formátumra figyeljen! (2023.-01. 01.)

Mégse Hozzáad

Lenyíló combobox-ból ki kell választani az esemény jellegét (pl. szervíz, büntetés) be lehet írni egyéb információt, illetve meg kell adni az esemény dátumát pont-al és szóközzel ellátva kell megadni, hogy helyes legyen a frontenden. a hozzád gombbal a megadott adatokat az eltárolt UserId-hoz rendeli. A mégse gomb pedig bezárja ezt a formot.