**Tick it easy**(Erdősi Péter, Hires Krisztián)  
/https://github.com/hireskrisz/full-stack/

**FE:** <https://tick-it-easy-fe.herokuapp.com/>

**BE:** <https://tick-it-easy-backend.herokuapp.com/>

A commitált kód CI környezetben tesztelve van, majd sikeres tesztelést követően közzé van téve a fent említett domain címeken.  
CI-tool: Travis CI  
CD-tool: Heroku

1. **fejlesztői környezet bemutatása, beállítása, használt technológiák:**A fejlesztői környezet Laravel 8 és Angular 10.  
   Backend futtatása: php artisan serve (lokálba a localhost:8000 porton)  
   Frontend futtatása: ng serve (lokálba localhost:4200) / npm run start (lokálba localhost:8080)  
   Használt technologiák:  
   - adatbázis: (travis környezetben MySQL, Produkciós környezetben PostgreSQL)  
   - laravelen belül: ORM megvalósítása Eloquent  
   - adatbázis szerkezet létrehozása illetve módisítása migrációval, kezdetleges adatfeltöltése seederekkel történik  
   - backend oldali tesztelés: PHPunit, tesztek feature tesztek a REST API tesztelésére  
   - későbbiekben az authentication-t a laravelben a passport package fogja végezni illetve a custom middleware-ek

|  |
| --- |
| Tickets |
| Id:number (primary key)  price:number  routeId:number  onDiscount:boolean  aviable:boolean |

1. **adatbázis-terv: táblák kapcsolati UML diagramja:**

|  |
| --- |
| Users |
| Id:number (primary key)  name:string  email:string  password:bcrypt  bucket:string  isAdmin:boolean  remember\_token:string |

N

N

N

N

|  |
| --- |
| Vehicles |
| Id:number (primary key)  type:string  licence:date  capacity:number |

1

|  |
| --- |
| Routes |
| Id:number (primary key)  from:string  to:string  startTime:date  endTime:date  activePassangers:number  vehicleId:number |
|  |

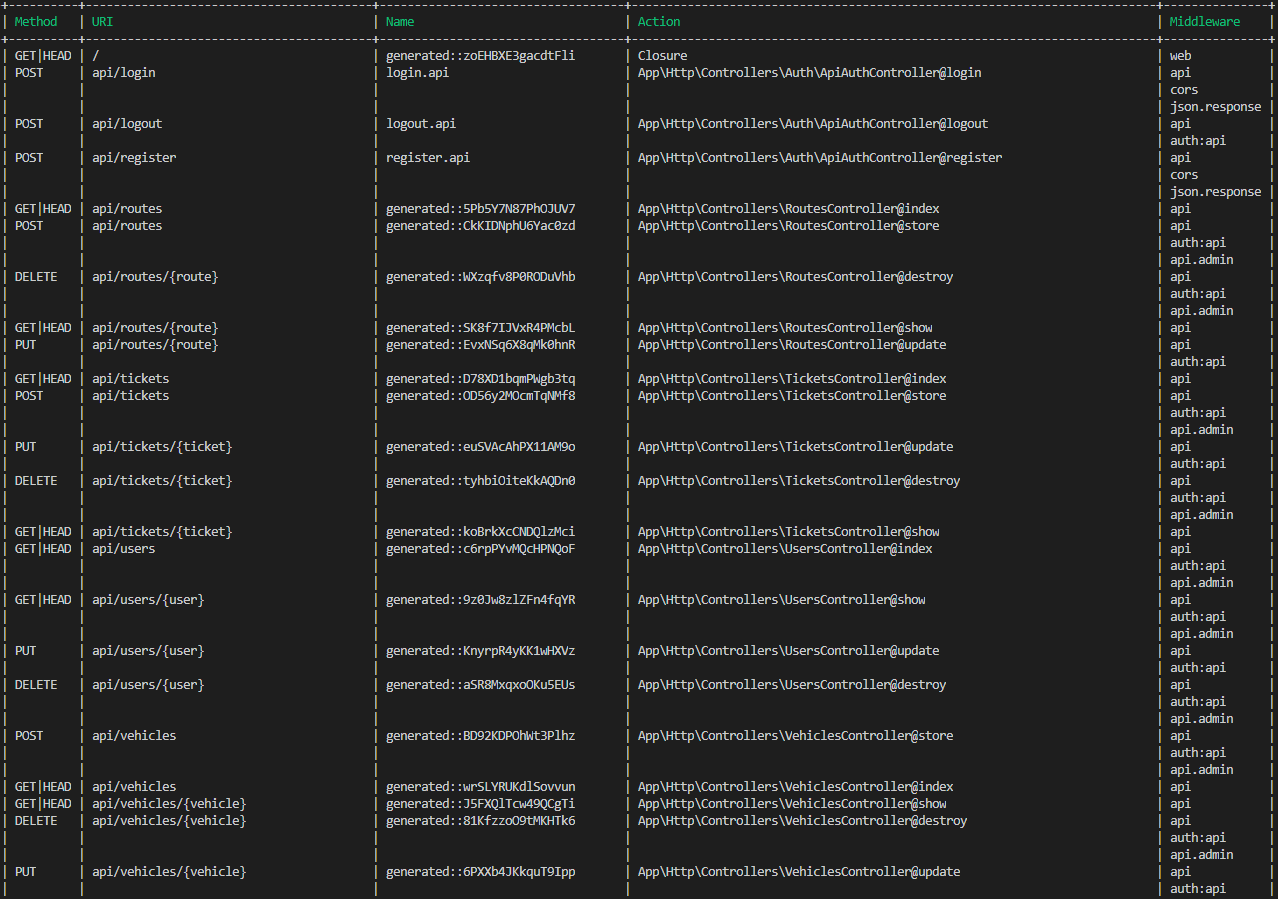
1

1

|  |
| --- |
| Users |
| id: number (primary key) name: string email: string bucket: string  isAdmin: boolean password: Hash(16bit integer)  remember\_token: string balance: number  timestamps |

N

N

1. **alkalmazott könyvtárstruktúra bemutatása:**A mappastrukturában így található meg: Laravel 8 – tick-it-easy-backend, Angular 10 – frontend. A strukturában jelen van még egy ReadMe.md ahol ismertetjük az webalkalmazást, illetve egy CI/CD folyamatot vezérlő / levezető yml fájl a Travis CI számára.
2. **Végpont-tervek és leírások:  
     
     
   api/login (POST) :** a bejelentkezéshez szükséges végpont, amely és cors middlewaret használ a cross origin requuestek kezelésére, illetve egy json response middlewaret, hogy a válasz mindenképpen json formátumban érkezzen meg. Input paraméterként vár egy az adatbázisban szereplő email címet, és a hozzátartozó jelszót. Válaszként egy Access tokent ad vissza, amely a további autentikációt igénylő végpontok eléréséhez szükséges.

**api/register (POST) :** a regisztrációhoz szükséges végpont, amely és cors middlewaret használ a cross origin requuestek kezelésére, illetve egy json response middlewaret, hogy a válasz mindenképpen json formátumban érkezzen meg. Input paraméterként vár egy az nevet, egy email címet, és a hozzátartozó jelszót. Válaszként egy Access tokent ad vissza, amely a további autentikációt igénylő végpontok eléréséhez szükséges.

**api/logout (POST) :** a kijelentkezéshez szükséges végpont, authentikáció szükséges hozzá, ezután kerül felszabadításra az adott felhasználóhoz tartozó acces token, válaszként unauthenticated hibaüzenetet, vagy egy Sikeresen kilejentkezett üzenetet kapunk.

**api/routes (GET) :** nyilvános végpont, authentikáció nem szükséges hozzá, az adatbázisban szereplő összes út lekérdezésére szolgál

**api/routes (POST) :** új útvonal felvételére szolgáló végpont, admin jogú felhasználótól szükséges authentikáció hozzá, válaszként hiba üzenetet ad vissza, ha egy adott mező helytelenül lett kitöltve ( validációk ), illetve egy success üzenetet, ha a felvétel sikerül.

**api/routes/{route} (DELETE):** egy útvonal törlésére szolgáló végpont, admin jogú felhasználótól szükséges authentikáció hozzá, válaszként hiba üzenetet ad vissza, ha a kért rekord nem szerepel az adatbázisban, vagy ha a felhasználónak nincs hozzá jogosultsága.

**api/routes/{route} (PUT) :** egy meglévő útvonal módosítására szolgáló végpont, felhasználói authentikáció szükséges az eléréséhez, amiatt, ha egy felhasználó bővíteni szeretné a kosarának a tartalmát, akkor az ehhez szükséges módosításokhoz ne kelljen admin jogú user authentikáció, válaszként hiba üzenetet ad vissza, ha egy adott mező helytelenül lett kitöltve ( validációk ), illetve egy success üzenetet, ha a módosítás sikerül.

**api/routes/{route} (GET) :** egy adott útvonal lekérdezésére szolgáló végpont, authentikáció nem szükséges hozzá, válaszul visszaadja az adott rekordot, ha szerepel az adatbázisban ( validáció ) .

**api/vehicles (GET) :** nyilvános végpont, authentikáció nem szükséges hozzá, az adatbázisban szereplő összes jármű lekérdezésére szolgál

**api/ vehicles (POST) :** új jármű felvételére szolgáló végpont, admin jogú felhasználótól szükséges authentikáció hozzá, válaszként hiba üzenetet ad vissza, ha egy adott mező helytelenül lett kitöltve ( validációk ), illetve egy success üzenetet, ha a felvétel sikerül.

**api/ vehicles /{ vehicles} (DELETE):** egy jármű törlésére szolgáló végpont, admin jogú felhasználótól szükséges authentikáció hozzá, válaszként hiba üzenetet ad vissza, ha a kért rekord nem szerepel az adatbázisban, vagy ha a felhasználónak nincs hozzá jogosultsága.

**api/ vehicles /{ vehicle } (PUT) :** egy meglévő jármű módosítására szolgáló végpont, felhasználói authentikáció szükséges az eléréséhez, amiatt, ha egy felhasználó bővíteni szeretné a kosarának a tartalmát, akkor az ehhez szükséges módosításokhoz ne kelljen admin jogú user authentikáció, válaszként hiba üzenetet ad vissza, ha egy adott mező helytelenül lett kitöltve ( validációk ), illetve egy success üzenetet, ha a módosítás sikerül.

**api/ vehicles /{ vehicle } (GET) :** egy adott jármű lekérdezésére szolgáló végpont, authentikáció nem szükséges hozzá, válaszul visszaadja az adott rekordot, ha szerepel az adatbázisban ( validáció ) .

**api/tickets (GET) :** nyilvános végpont, authentikáció nem szükséges hozzá, az adatbázisban szereplő összes jegy lekérdezésére szolgál

**api/ tickets (POST) :** új jegy felvételére szolgáló végpont, admin jogú felhasználótól szükséges authentikáció hozzá, válaszként hiba üzenetet ad vissza, ha egy adott mező helytelenül lett kitöltve ( validációk ), illetve egy success üzenetet, ha a felvétel sikerül.

**api/ tickets /{ ticket } (DELETE):** egy jegy törlésére szolgáló végpont, admin jogú felhasználótól szükséges authentikáció hozzá, válaszként hiba üzenetet ad vissza, ha a kért rekord nem szerepel az adatbázisban, vagy ha a felhasználónak nincs hozzá jogosultsága.

**api/ tickets /{ ticket } (PUT) :** egy meglévő jegy módosítására szolgáló végpont, felhasználói authentikáció szükséges az eléréséhez, amiatt, ha egy felhasználó bővíteni szeretné a kosarának a tartalmát, akkor az ehhez szükséges módosításokhoz ne kelljen admin jogú user authentikáció, válaszként hiba üzenetet ad vissza, ha egy adott mező helytelenül lett kitöltve ( validációk ), illetve egy success üzenetet, ha a módosítás sikerül.

**api/ tickets /{ ticket } (GET) :** egy adott jegy lekérdezésére szolgáló végpont, authentikáció nem szükséges hozzá, válaszul visszaadja az adott rekordot, ha szerepel az adatbázisban ( validáció ) .

**api/users ( GET ) :** az összes felhasználó lekérdezéséhez szükséges végpont, admin szintű authentikáció szükséges hozzá

**api/users/{user} ( GET ) :** egy adott felhasználó lekérdezéséhez szükséges végpont, admin szintű authentikáció szükséges hozzá

**api/users/{user} ( PUT ) :** a felhasználó saját adatainak módosítására szolgáló végpont, autentikáció szükséges hozzá az adott felhasználótól, ezen keresztül tudja megváltoztatni a felhasználó nevét, jelszavát, egyenlegét, illetve kosarának tartalmát.

**api/users/{user} ( DELETE ) :** egy felhasználó törlésére szolgáló végpont, admin jogú autentikáció szükséges , ezen keresztül tudunk törölni egy adott felhasználót az adatbázisból.

1. **1db funkció folyamatának a leírása:**

Belépés funkció:  
- a weboldalon egy form felületben beírt adatokat kinyerjük majd egy api hívással megszólítjuk a belépési végpontot, ami válaszként visszaadja nekünk a felhasználó számára biztosított tokent és a felhasználói adatokat. Némelyik részét az adatokat eltároljuk SessionStorage-ba, későbbi könnyed felhasználáshoz. Mindezt kilépéskor ürítjük, illetve egy service szolgálja ki nekünk ezek adatok könnyed elérését. Ábrák:



. ábra Belépési metódus



. ábra Auth service egy részlete



. ábra Kilépési metódus

1. **1 db végpont működésének leírása, mi történik, milyen lépések követik egymást:**

**<domain>/api/tickets működése:**

GET request

a servernek

Meghívódik az

index metóds

(controller)

A metódusban az

adatbázisban szereplő

jegyek reprezentálására

szolgáló szerkezet épül fel (model)

Valid JSON objektum(ok)

szolgáltatás a kliens alkalmazásnak

1. **Laravel könyvtár struktúra:**

app Alkalmazás logika

* 1. Commands ace parancsok
  2. Http HTTP specifikus alkalmazás logika
     1. Controllers HTTP kérések kezelése
     2. Middleware HTTP kérések kezelése előtt végrehajtandó
  3. Listeners Bejövő kérések figyelése
  4. Model Modell osztályok, logika, működés
     1. Services Komponensek
     2. Traits Modellek kiterjesztése

bootstrap Az app indulásához szükséges műveletek, konfigurációk

config Az app működését befolyásoló konfigurációs fájlok

database Adatbázissal kapcsolatos

* 1. migrations Adatbázis migrációk, séma meghatározása
  2. seeds Adatbázis seederek, példa/tesztadatok beszúrása

providers Providerek

public A webszerver által közvetlenül kiszolgált fájlok

resources Az app által felhasznált egyéb fájlok

views

storage Felhasználók által feltöltött képek

1. **Kliensoldali szolgáltatások listája:**

* belépés
* kilépés
* felhasználó törlése
* felhasználó adatainak a modositasa
* egyenleg feltöltés
* jelszó változtatás
* útvonal törlés
* jármű törlés
* új útvonal létrehozása
* kosárba rakás és fizetés (nem volt idő megvalósítani)

Tesztek

Backend tesztek:

1.: Route apik tesztelése ( 5db )

* + - összes route lekérdezése 200-as várt válaszüzenettel
    - egy route lekérdezése 200-as várt válaszüzenettel
    - -egy route módosítása, felhasználói szintű authentikációval apin keresztül, majd az adott elem lekérdezése és ezen két elem összehasonlítása
    - egy route létrehozása apin keresztül admin szintű authentikációval, majd ezen route lekérdezése az adatbázisból, és az eredmény összehasonlítása
    - egy adott route létrehozása apin keresztül, ezen route létezésének tesztelése az adatbázisban, majd ezen route törlése az adatbázisból apin keresztül, és ezen route hiányának tesztelése az adatbázisból

2.: Vehicle apik tesztelése ( 5db )

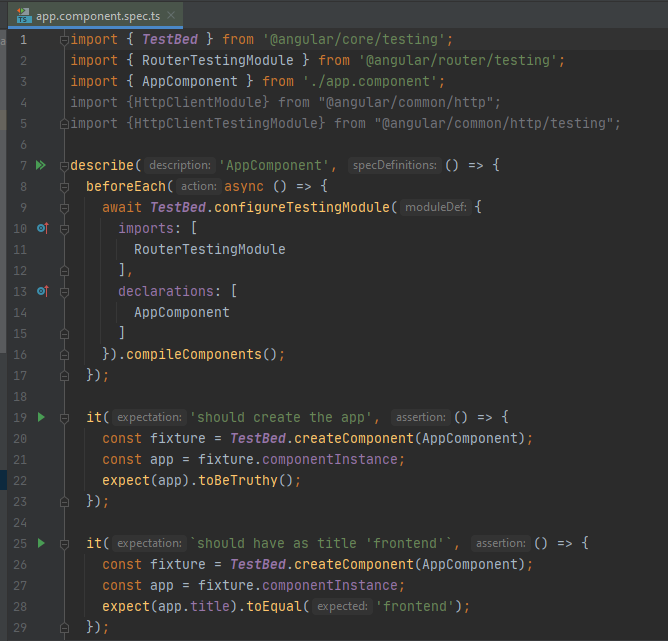
* + - összes vehicle lekérdezése 200-as várt válaszüzenettel
    - egy vehicle lekérdezése 200-as várt válaszüzenettel
    - egy vehicle módosítása, felhasználói szintű authentikációval apin keresztül, majd az adott elem lekérdezése és ezen két elem összehasonlítása
    - egy vehicle létrehozása apin keresztül admin szintű authentikációval, majd ezen vehicle lekérdezése az adatbázisból, és az eredmény összehasonlítása
    - egy adott vehicle létrehozása apin keresztül, ezen vehicle létezésének tesztelése az adatbázisban, majd ezen vehicle törlése az adatbázisból apin keresztül, és ezen vehicle hiányának tesztelése az adatbázisból

3.: Ticket apik tesztelése ( 5db )

* + - összes ticket lekérdezése 200-as várt válaszüzenettel
    - egy ticket lekérdezése 200-as várt válaszüzenettel
    - egy ticket módosítása, felhasználói szintű authentikációval apin keresztül, majd az adott elem lekérdezése és ezen két elem összehasonlítása
    - egy ticket létrehozása apin keresztül admin szintű authentikációval, majd ezen ticket lekérdezése az adatbázisból, és az eredmény összehasonlítása
    - egy adott ticket létrehozása apin keresztül, ezen ticket létezésének tesztelése az adatbázisban, majd ezen ticket törlése az adatbázisból apin keresztül, és ezen ticket hiányának tesztelése az adatbázisból

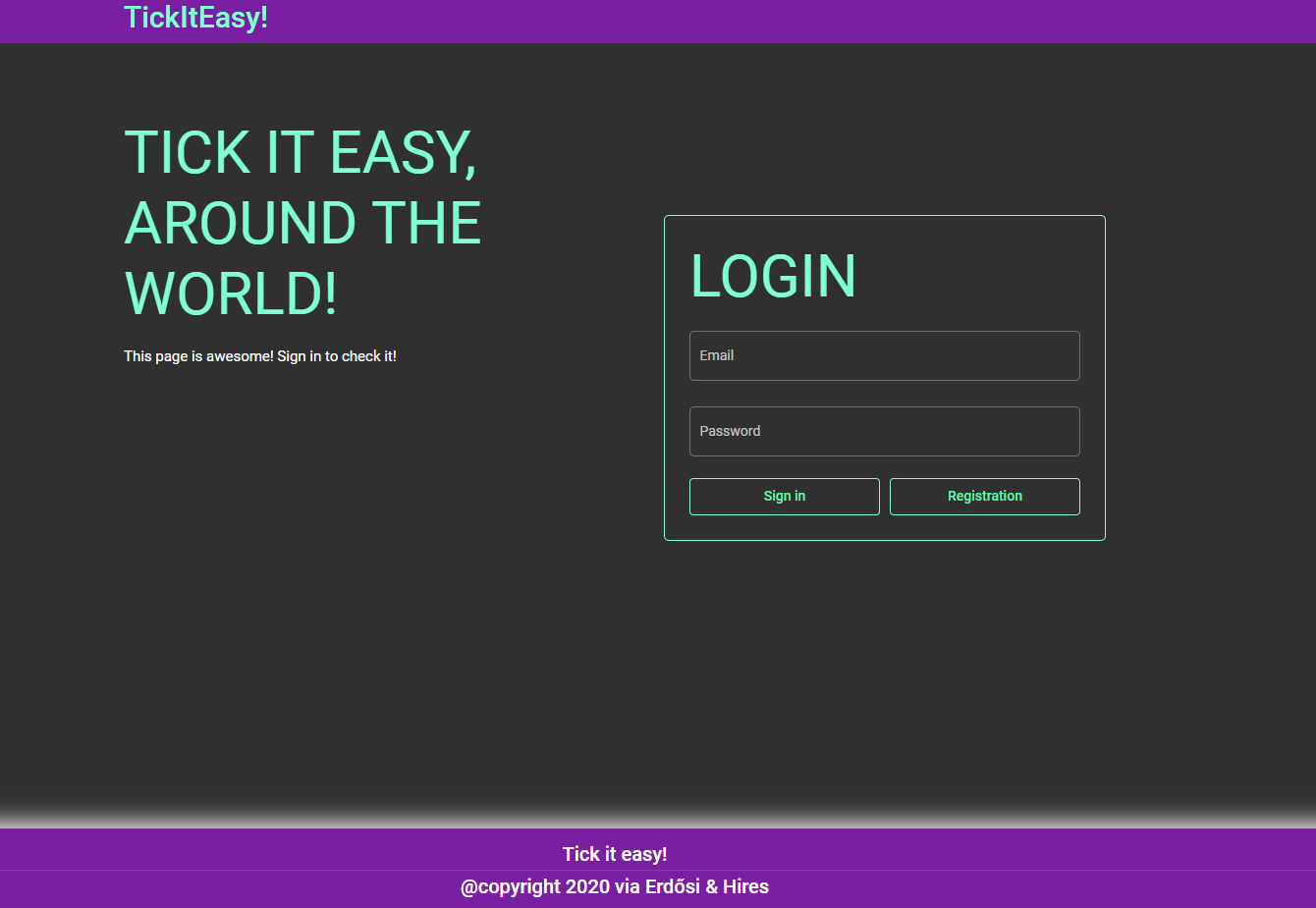
Frontend tesztek:

A frontend teszteket az Angular beépített tesztelőjével lettek tesztelve. (Karma-Jasmin)  
Ez le ellenőrzi, hogy a komponensek létrejöttek e megfelelően és hozzá vannak e rendelve a saját maga által használt modulok. Egy tipikus példa lehet:

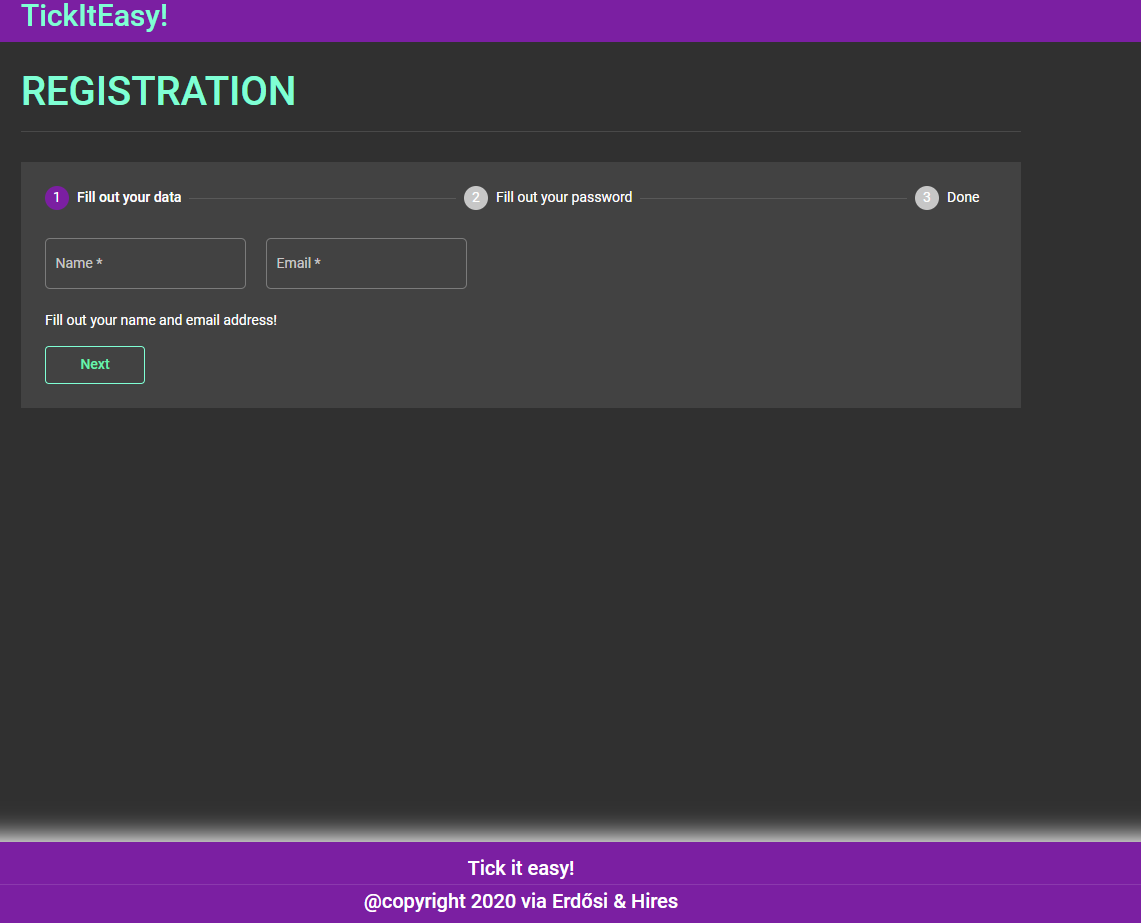


Felhasználói dokumentáció:

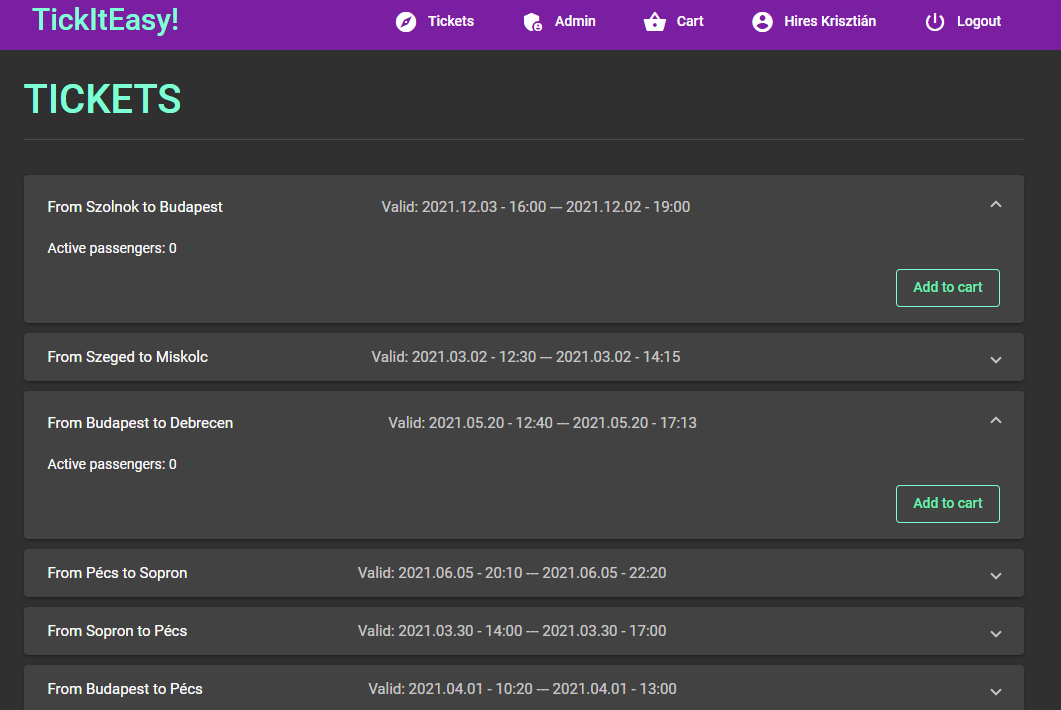
A felület interneten keresztül érhető el: <https://tick-it-easy-fe.herokuapp.com/>



Felhasználó adatainkat megadva érhető el az alkalmazás.  
Ha nincsen felhasználói adatunk könnyedén létrehozhatunk egyet a Regisztráció gombra kattintásával, ami átirányít minket erre az oldal-ra:

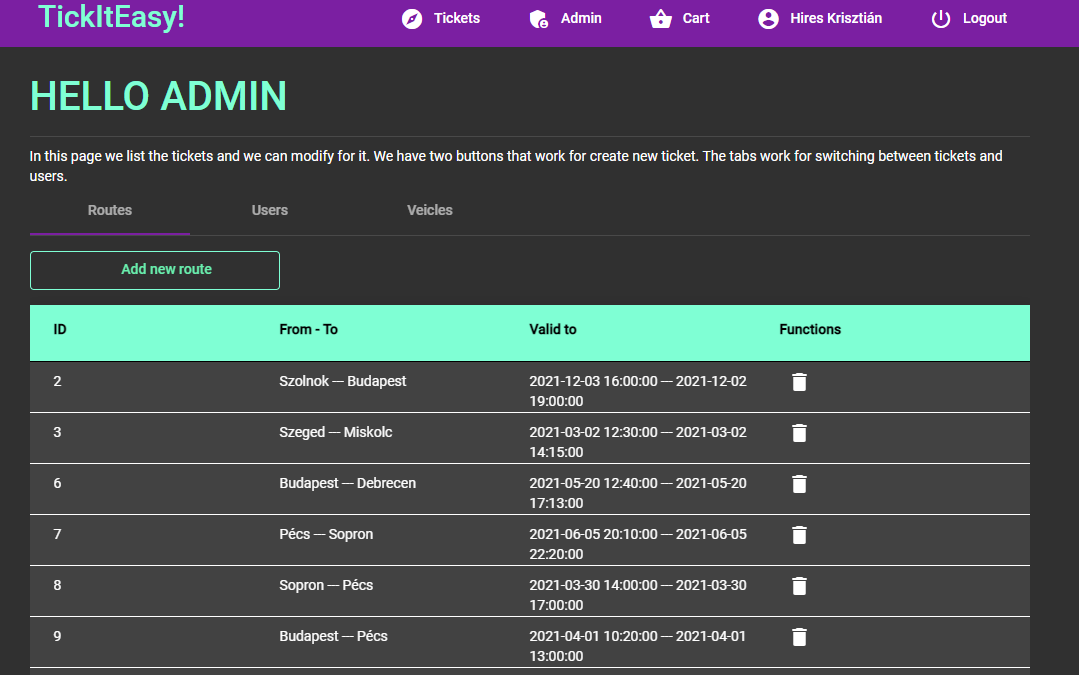


Miután sikeresen regisztráltunk a rendszer automatikus beléptet minket a fiókunkba. A kezdőlapon láthatóak az útvonalak ahová jegyeket tudunk vásárolni:

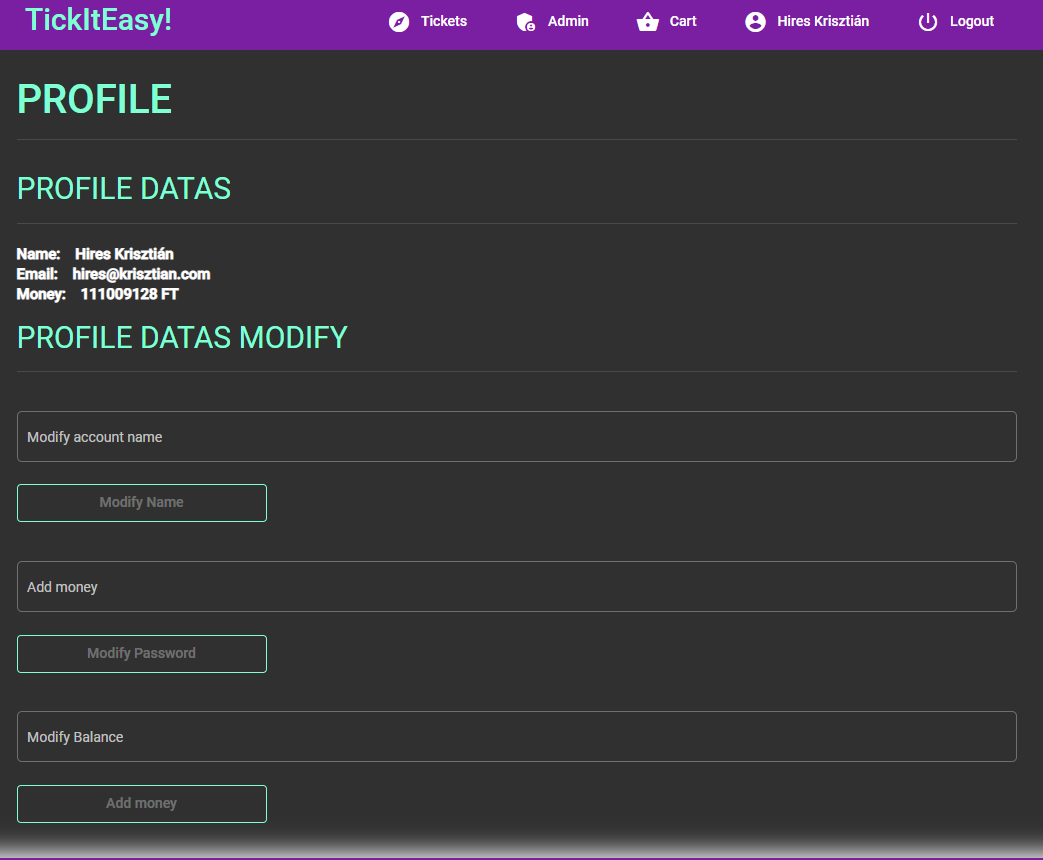


Kis kártyákon láthatóak az információk. A kosarunkhoz tudjuk adni ezeket a jegyeket.

Továbbá elérhető még Admin felhasználók számára egy úgynevezett Admin oldal is ahol a megjelenített tartalmakat illetve a háttérfolyamatokat tudjuk kezelni / manipulálni.



A profil oldalunkon láthatjuk az adatainkat, illetve képesek vagyunk azok módosítására, szigorúan csak a név és jelszó módosítható. Továbbá képesek vagyunk egyenleget is feltölteni, amit le tudunk vásárolni, amikor jegyeket szeretnénk venni



A kosárban láthatjuk azokat az elemeket amiket megszeretnénk vásárolni, a kosár oldalon tudjuk összegezve látni majd ott megvásárolni a jegyeket annak függvényében, hogy van e rá keretünk:

