**Vizsgaremek**

dokumentáció

Szoftverfejlesztő– és tesztelő technikus szakma

**Bevezetés**

A modern technológia fejlődésével egyre nagyobb igény mutatkozik az üzleti folyamatok digitalizálására és automatizálására. Ez nincs máshogy a sportlétesítmények területén sem, ahol a hatékonyság és a foglalások megfelelő kezelése kulcsfontosságú. Ezek megvalósításában egy jól megtervezett informatikai rendszer nagyban segíthet.

Így amikor szeptemberben nekiálltunk kigondolni az ötletet, hogy milyen típusú programot próbáljunk meg lefejleszteni, arra gondoltunk, hogy olyat szeretnénk készíteni, ami hasznos, és ráadásul élvezettel is csinálnánk.

Arra jutottunk, hogy mivel mind a hárman szeretjük a labdarúgást, ezért egy ehhez kapcsolatos projektnek kezdünk neki. Végül egy focipálya és tornaterem foglaló rendszer fejlesztése mellett döntöttünk, mivel olyan hasonló oldallal, ahol bárki regisztrálhat általa elérhető termet, és eközben bárki bérelhet az elérhető termek közül nem találtunk.

A pontos elképzelésünk egy közösségi médiára hasonlító oldal, ahol nem csak bérelni és bérbe adni lehet termeket, hanem például hozzászólás formájában meg lehet osztani saját élményeinket, véleményünket a teremmel kapcsolatban, vagy éppen feltehetjük kérdéseinket akár a terem tulajdonosának is.

Ezáltal a mostani piacot elnézve teljesen egyedi rendszer alakulna ki, ami magában egy teljesen más foglalási élményt nyújtana a felhasználóknak, hiszen jelenleg, ha például egy tornatermet szeretnénk lefoglalni, akkor ugyan lehet találni kisebb gyűjtőoldalakat, ahol megtalálható több terem is, viszont ezek maximum 2-3 termet jelenítenek meg összesen. Ezen kívül általában saját weboldalaik vannak a termeknek vagy sportlétesítményeknek, így megnehezítve a keresést és az összehasonlítást.

Fő célunk Magyarországon lévő iskolák meggyőzése, hogyha termet szeretnének kiadni, akkor a mi platformunkat válasszák, így akár a legvégén egy több száz iskolából, sportközpontból, vagy akár később egyéb szórakozási létesítményekből álló platform tud kialakulni, ami nekik is megéri, hiszen pszichológiai értelemben nagyobb valószínűséggel fog foglalni a felhasználó egy termet, hogy ha több opciót is lát egymás mellett (életben például a különböző gyorséttermek ezért vannak egymáshoz közel).

Másrészről pedig ezzel akár a későbbiekben az iskolák tornatermeinek minősége is javulhat, hiszen, ha egy jó, tiszta, terem kerül kiadásra, amit a felhasználók szívesen bérelnek, akkor az iskolákhoz már jelentős pénzösszeg kerülhet a foglalások által, amivel ösztönözve őket a tornatermeik fejlesztésére, igényesebbé tételére.

Ezáltal ráadásul az iskolában tartott testnevelés órák minősége, élménye is fejlődhet, amivel egy egészségesebb, sportolni szerető generáció nőhet fel.

Összegezve, a projekttel egy piaci űrt töltenénk ki, és ezenfelül magára a környezetre, és az emberekre is pozitív hatást fejtenénk ki.

**Fejlesztői dokumentáció**

**Felhasznált technológiák**

A backendet ExpressJS keretrendszer, az adatbázist MySQL szerver biztosítja. A frontend padig Angular keretrendszerben teljesen reszponzív nézettel rendelkezik, a karbantartás(admin) felület kivételével, ami csak a gépi nézetet támogatja.

**Backend**

*Készítéshez használt programok:*

* Visual Studio Code 1.98.2
* XAMPP v3.3.0
* phpmyadmin
* Postman 10.16.0
* Excel

**Frontend**

*Készítéshez használt programok:*

* Visual Studio Code 1.98.2
* Google Chrome

**Verziókezelés**

*Készítéshez használva:*

* Git, Github

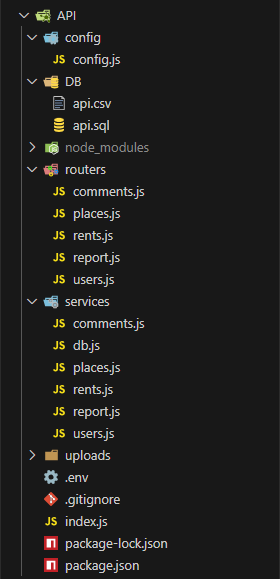
**Kódolási konvenciók**

A kódot git verziókezelővel használjuk.

**Alapkönyvtárak**

* Frontend
* API (Backend)
* Documents

**Backend**

A Backendet az API nevű mappa valósítja meg, annak felépítése:

**config/config.js** fájl fájl felel az adatbázis kapcsolat konfigurációjáért.

**DB/api.csv , DB/api.sql** fájlok felelnek az adatbázis esetleges importálására phpmyadmin segítségével.

**node\_modules** mappában találhatóak az npm által telepített csomagok.

**routers** mappában találhatóak a végpontokhoz kapcsolódó közvetlen metódusok, amikor a végpont meghívódik, először ezek a metódusok futnak le a backenden.

**services** mappában találhatóak azok a metódusok amik az adatbázisba írásban, vagy onnan olvasásban felelősek.

**uploads** mappába kerülnek a teremhez feltöltött képek , amik adatbázisban mint url elérhetőek.

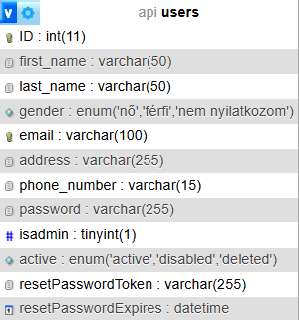
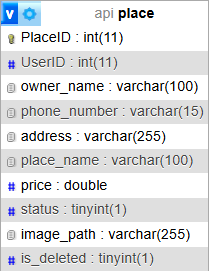
.**env** fájl tartalmazza a SECRETKEY-t, ami az athentication token előállításában játszik szerepet.

**package-lock.json / package.json** fájlok az npm csomagtelepítő fájljai.

**index.js** a törzs fájl, itt fut le a szerverindítás, kapcsolatlétrehozás, routerek meghívása, és a routerek végpontjainak az egyik beállítása: users/, places/, rents/, comments/, reported/.

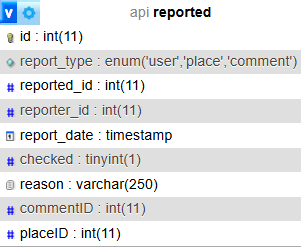
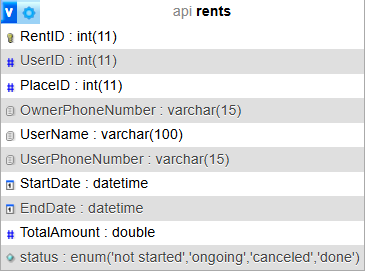
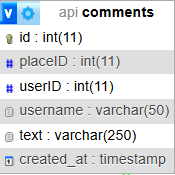
**.gitgnore** fájl tartalmazza azokat a fájlokat amiket nem szeretnénk feltölteni git verziókezelőbe.

Az adatbázis **api** néven lett létrehozva, ami az alábbi táblákra lett leosztva:



**Place**

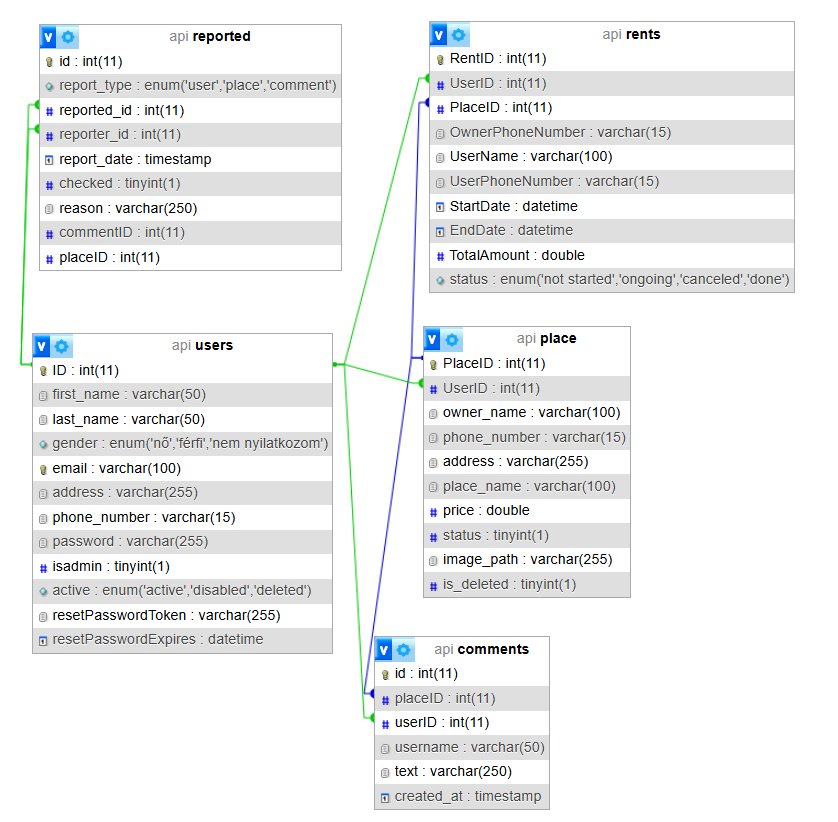
**Users**



**Reported**

**Comments**

**Rents**

**Adatbázis kapcsolatok**

Az adatbázishoz a fejlesztése közben az alábbi végpontok kerültek kialakításra:

**A users/ adattáblához tartozó végpontok:**

|  |  |  |  |
| --- | --- | --- | --- |
| Végpont | Metódus | Azonosítás | Leírás |
| / | get | nem | összes felhasználó lekérdezése |
| /profile | get | igen | felhasználó lekérdezése ID szerint a profil oldalhoz |
| /:id | get | nem | felhasználó lekérdezése megjelenítéshez (pl. Főoldali adatok, bérlésnél, terem adatoknál) |
| /register | post | nem | felhasználó regisztráció |
| /login | post | nem | felhasználó bejelentkezés |
| /:id | delete | igen | Felhasználó törlése |
| /:id | patch | igen | Felhasználó adatainak frissítése |
| /sadmin-update-profile/:id | patch | igen | admin által a felhasználó szerkesztése |
| /isAdmin/:id | get | nem | Megnézi, hogy a felhasználó admin-e, visszaad egy igaz / hamis értéket |
| /forgot-password | post | nem | felhasználó elfelejtett jelszó esetén újat kérhet, aminek beállításához a link e-mailben érkezik |
| /reset-password | put | igen | új jelszót hoz létre a felhasználónak, ha elfelejtette |

**A place/ adattáblához tartozó végpontok:**

|  |  |  |  |
| --- | --- | --- | --- |
| Végpont | Metódus | Azonosítás | Leírás |
| / | get | nem | Összes hely lekérdezése |
| /:id | get | igen | Helyek UserID szerinti lekérdezése |
| /places-rent/:PlaceID | get | igen | Helyek PlaceID szerinti lekérdezése |
| /create | post | igen | Felhasználó által hely létrehozása |
| /:id/image | delete | igen | A terem tulajdonosa tudja a terem képét törölni |
| /:id | delete | igen | Az terem tulajdonosa tudja a termet törölni |
| /:id | put | igen | A terem tulajdonosa tudja frissíteni a termet |

**A rents/ adattáblához tartozó végpontok:**

|  |  |  |  |
| --- | --- | --- | --- |
| Végpont | Metódus | Azonosítás | Leírás |
| / | get | nem | Összes foglalás lekérdezése |
| /rent/:PlaceID | get | nem | Foglalás lekérdezése termek PlaceID alapján |
| /:userID/:rentID | get | igen | Foglalás státuszának lekérdezése annak frissítéséhez |
| / | post | igen | Foglalás létrehozása felhasználó által |
| /:userID/:rentID | put | igen | Kezdődátum, befejezési dátum, és státusz frissítése |
| /:id | get | igen | Termek lekérdezése felhasználók szerint |
| /:userID/:rentID | delete | igen | Foglalás törlése |

**A comments/ adattáblához tartozó végpontok:**

|  |  |  |  |
| --- | --- | --- | --- |
| Végpont | Metódus | Azonosítás | Leírás |
| / | get | nem | Össze komment lekérése |
| /:placeId | get | nem | Kommentek termek szerinti lekérdezése |
| /user/:userID | get | igen | Kommentek felhasználó szerinti lekérdezése |
| /create | post | igen | Komment létrehozása felhasználó által |
| /:id | delete | igen | Komment törlése felhasználó által |

**A reported/ adattáblához tartozó végpontok:**

|  |  |  |  |
| --- | --- | --- | --- |
| Végpont | Metódus | Azonosítás | Leírás |
| / | get | nem | Összes bejelentés lekérdezése |
| /create | post | igen | Jelentés létrehozása a felhasználó által |
| /update | put | igen | Leellenőrzött státusz frissítése |
| /:id | get | igen | Jelentés lekérdezése ID alapján |

**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 fenti táblázatokban felsorolt, „Azonosítás” oszlopban „igen” szóval jelölt végpontok.

Ahol a fenti táblázatokban az „Azonosítás” oszlopban „nem” szó szerepel, azok a végpontok publikusak, nem szükséges regisztráció azok használatához.

Az adatok JSON formátumban kerülnek fogadásra / továbbadásra a Frontend oldal felé. A frontend oldalon pedig komponensek valósítják meg a vezérlést, amik közvetlenül a szervízekkel vannak kapcsolatban, ahol történik az adatbázisból való kiolvasás, és az adatok kiírása az adatbázisba.

Backend részen belül a routers mappában lévő metódusok felelősek a végpontok meghívására. A routers mappában lévő metódusok pedig általában a services mappában lévő metódusokat olvasva hajtják végre az adatbázisba való írást /olvasást.

**Frontend**

A frontend angular keretrendszerrel készült, azon belül ahogy említettük komponensek (innentől component) valósítják meg a vezérlést, amik közvetlenül szervízekkel (innentől services) vannak összekapcsolva, és ott történik az adatbázisba való írás / kiolvasás.

**Egyéb használt technológiák:**

* node package manager (npm)
  + NgbModule
  + FormsModule
  + NgxPaginationModule
* HtppClient

**„Frontend” mappa felépítése:**

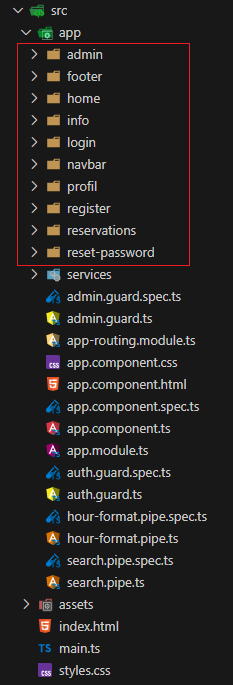
Az angular keretrendszernek, a GIT-nek, illetve az npm-nek köszönhetően az alábbi fájlok kerültek létrehozásra:

* .angular – angular működéséért felelős fájlok
* .vscode – angular működéséért felel, főleg a command parancsok kezelésére szolgál
* node\_modules – npm által telepített bővítmények tárolása
* src – fő programfájlok, részletesebben lentebb lehet olvasni
* .gitignore – azok a fájlok nevét tárolja, amiket nem szeretnénk GIT-re feltölteni
* angular.json – angular beállításáért felelős
* package-lock.json – npm, és az általa letöltött csomagok beállítása / működése
* package.json - npm, és az általa letöltött csomagok beállítása / működése
* README.md – Programismertető
* tsconfig.app.json – Konfigurációs fájl
* tsconfig.json – Konfigurációs fájl
* tsconfig.spec.json – Konfigurációs fájl

**„src” könyvtár felépítése:**

* app/ - app könyvtáron belül találhatóak a kontrollerek, modellek, pipe-ok, és guardok
* assets/ - mappába találhatóak a használt képek
* index.html – a fő html fájl, a header szerepéért felelős
* main.ts – fő typescript fájl, nem lett benne módosítva semmi.
* style.css – fő stílus fájl, egységes stílushoz használt beállítások találhatóak benne.

**„app” könyvtár felépítése:**

**Piros négyzetben lévő mappák a komponensek mappái:**

* **admin** – admin oldal
* **footer** – lábléc
* **home** – főoldal
* **info** – alapértelmezett oldal / információs oldal
* **login** – bejelentkezési oldal
* **navbar** – navigációs sáv oldal
* **profl** – profil oldal
* **registe**r – regisztrációs oldal
* **reservations** – foglalások megtekintésére szolgáló oldal
* **reset-pasword** – elfelejtett jelszó esetén elérhető oldal.

**services** mappában találhatóak a komponenshez található szervízek, amik az adatbázis kommunikációra

**admin.guard** fájlok játszanak szerepet abba, hogy csak admin felhasználó érhesse el a karbantartás (azaz admin) oldalt.

**app-routing-module** felelős a navigáció működéséért.

**app.component** fájlok az alapértelmezett html, css, ts fájlok a routingon belül.

**app.module** fájl felelős az importok, és a declarations útvonalának a beállításért.

**auth.guard** felelős arra, hogy csak azok érhessék el a főoldalt, a profil oldalt, és a foglalások oldalt, akik be vannak jelentkezve.

**Hour-format.pipe** felelős az órák kiírásáért, és megformázásáért foglaláskor.

**Search.pipe** felelős a kereső mezők működéséért.

**„Services”, és azon belüli eljárások**

**Auth service**

**isAdmin()** – megnézi, hogy a bejelentkezett felhasználó Admin-e., ha igen akkor pedig beállítja az isAdminSubject változót, hogy elérhető legyen az adminok számára a karbantartás oldal.

**getLoggedUser()** – lekéri a bejelentkezett felhasználót.

**ForgotPassword()** – egy emailt kér, és az emaillel együtt tovább küldi a users/forgot-password végpontra a kérést, így az email tulajdonosa kap egy elfelejtett jelszó emailt, amiben egy link található, ahol vissza lehet állítani a jelszót.

**isLoggedIn()** – igaz / hamis értéket küld vissza, lekéri a localstorage-ból a felhasználót, célja hogy megnézze a felhasználó be van-e jelentkezve.

**Register()** – users/register végpontra küldi a regisztrációkor megadott adatokat.

**Login()** – users/login végpontra küld bejelentkezési kérést, illetve, ha sikeres a bejelentkezés, akkor beállítja a localstorage-ba a loggedUser néven a bejelentkezett felhasználó adatait (jelszó, isAdmin, active, resetPasswordToken, resetPasswordExpire mezők kivételével), illetve beállítja a loggedUserSubjectet.

**Logout()** – users/logout végpontra küld kérést, majd ha sikeres, kijelentkezteti a felhasználót, azaz loggedUserSubject-et, isAdminSubject-et, és a userSub-ot nullára állítja, és eltávolítja a loggedUsert a localstorage-ból.

**Base service**

Itt található a httpOptions, illetve az apinak az url-je.

**Place service**

**getAllPlaces()** – lekéri a places/ végpontról az összes helyet.

**getAllComments()** – lekéri az összes kommentet.

**getPlaceById()** – lekéri a helyet a felhasználó azonosítója (userID) alapján a places/:userID végpontról.

**getPlaceByUserId()** – ugyanaz a metódus, mint a getPlaceById() viszont itt Observable-ként kapjuk meg az adatokat.

**getPlaceByPlaceID()** – terem azonosítója alapján kéri le a termet a places/places-rent/:placeID végpontról.

**deletePlaceImage()** – meghívja a places/:placeID/image végpontot, és kitörli a terem képének az url-jét.

**updatePlace()** – meghívja a places/:placeID végpontot, és a kapott adatokkal frissíti a terem adatait.

**deleteComment()** – meghívja a comments/:id végpontot, és az adott hozzászólást törli.

**updatePlaceFromForm()** – meghívja a places/:placeID végpontot és frissíti a terem adatait.

**deletePlace()** – meghívja a places/:placeID végpontot, és az adott azonosítójú termet törli.

**updateStatus()** – meghívja az updatePlace() metódust, a státusz értéket változtatva, felhasználóknak ez a fő törlés, ha azt szeretnék, hogy egyelőre ne legyen elérhető a terem.

**getComments()** – lekéri az összes hozzászólást a comments/ vlgpontról

**getCommentsById()** – felhasználó azonosítója alapján lekéri a comments/user/:userID végpontról a felhasználóhoz kapcsolódó kommenteket.

**addComment()** – comments/create végpontot meghívva létre tud hozni a felhasználó egy hozzászólást

**Rents service**

**getAllRents()** – lekéri az összes foglalást a rents/ végpontról.

**createRent()** – új foglalást hoz létre a rents/ végponton.

**cancelRent()** – véglegesen törli a foglalást a rents/ végponton.

**getRentsForUser()** – felhasználó azonosítója (userID) alapján kéri le a foglalásokat a rents/:userID végpontról

**getRentsByPlaceID()** – terem azonosítója (placeID) alapján kéri le a foglalásokat. A rents/rent/:placeID végpontról.

**updateRent()** – a rents/:userID/:rentID végpontot meghívva frissíti a foglalás adatait.

**getStatus()** – lekéri a foglalás státuszát a rents/:userID/:rentID végpontról.

Users service