Gurzó Dávid
MK9FBF

FRONT-END DOKUMENTÁCIÓ

1. Leírás

A MusicPlayr-re keresztelt alkalmazás egy webes SPA, mely a React segítségével jött létre. Az alkalmazás amellett, hogy műfajokat, előadókat, albumokat és dalokat jelenik meg, számos más funkcióval is rendelkezik.

2. A projekt telepítése és indítása

1. Navigáljunk a projekt „music-playr” nevű mappájába
2. Töröljük a „package-lock.json” fájlt
3. Adjuk ki az *„npm install”* parancsot
4. Adjuk ki az *„npm start”* parancsot
5. Az alkalmazás a http://localhost:3000 -en érhető el

Az alkalmazás forráskódja megtalálható a GitHub-on is. Innen klónozható a *„git clone https://github.com/dgurzo/music-playr.git”* parancs segítségével.

3. Technológia

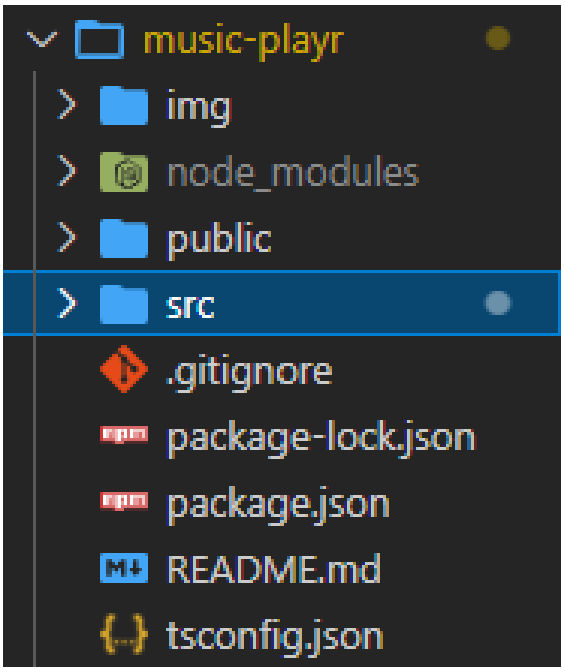
A front end megvalósításához használt eszközök:

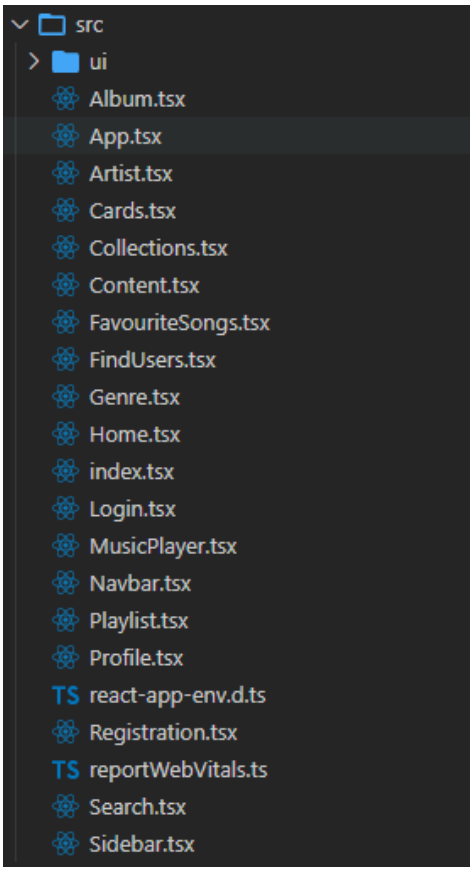
- React
- React Hooks
- Styled Components
- React router
- TypeScript

4. Projekt struktúra

Minden komponens, ami megjelenik a böngészőben az „src” mappában található meg.

A „ui” mappában styled components segítségével létrehozott stílus konstansok találhatók, amiket többször is fel vannak használva a komponensekben.





5. Megvalósított funkciók

1. *Regisztráció és bejelentkezés:*
    a. A „/registration” oldalra lépve regisztrálhatunk az oldalra. Regisztráció során meg kell adni egy nevet, email címet, falhasználónevet és jelszót is.
    b. Regisztrálás után a „/login” oldalra lépve van lehetőségünk a beregisztrált felhasználónévvel és jelszóval bejelentkezni.
    c. A kijelentkezés a jobb felső sarokban található, piros „Logout” gomb megnyomására történik meg.
2. *Navigáció:*
    a. A sidebar „Keresés” menüpontjára kattintva megjelennek az adatbázistól kapott műfajok.
    b. Egy adott műfajra kattintva a „/genre” route következik, ahol megjelennek az adott műfajhoz tartozó előadók.
    c. Az előadóra kattintva a „/artist” route következik, ahol a kiválasztott előadó albumai jelennek meg.
    d. Az album kártyájára kattintva a „/album” route jelenik meg, ahol az albumhoz tartozó dalok jelennek meg.

3. *Kedvelés:*
    a. A „/artist” oldalon az előadó neve alatt található zöld „Like” gomb segítségével kedvelhető az artist.
    b. Az „/album” oldalon található „Like” gomb használatával kedvelhető az adott album.
    c. Az „/album” oldalon található dalok címe mellett lévő szív megnyomására kedvelődik a dal.
    d. A „/collections” oldalon listázódnak a kedvel előadók és albumok. (Ez a sidebar „Gyűjteményeim” menüpontján keresztül is elérhető.)
    e. A „/favouritesongs” oldalon listázódnak a kedvelt dalok egy lejátszási lista formájában. (Ez a sidebar „Kedvencek” menüpontján keresztül is elérhető.)
4. *Felhasználók követése:*
    a. Bejelentkezés után a sidebar-on megjelenik a „Find users” menüpont.
    b. A „/findusers” oldalon listázódik az összes regisztrált felhsználó.
    c. Egy felhasználóra kattintva megjelenik a „/profile” oldal, ahol bekövethető egy másik user.
    d. A profil oldalon megjelenik a követett felhasználók listája.

6. Komponensek felépítése

A fejlesztés során React Function Component-eket használtam.

A komponensek állapotait a useState hook használatával hoztam létre és módosította. A useEffect hook tipikusan arra való, hogy amikor a komponens mount-olódik, lefut a benne lévő kód. Tipikusan én ilyenkor a back-end felé küldtem http kéréseket a fetch függvény segítségével.

Function Component-ek esetében nem kell a return-t a render metódusba rakni.

```
export const Search: FunctionComponent = () =>  {
    const [Genres, setGenres] = useState<Genre[]>([]);

    useEffect(() => {
        const getGenres = async () => {
            let response = await fetch("http://localhost:5000/api/genres/get/genres");
            let genres = await response.json();
            console.log(genres);
            setGenres(genres);
        }

        getGenres();
    }, []);

    return (
```

7. API fetch

Az alkalmazásban GET és POST kéréseket küldök a back end felé, ehhez a JavaScript fetch() metódusát használtam.

Ezeket általában egy arrow function belsejében valósítottam meg, amit végül a useEffetc hook belsejében hívtam meg.

GET:

```
React.useEffect(() => {
    GetUsers();
    console.log(users);
}, [])

const GetUsers = async () => {
    let response = await fetch("http://localhost:5000/users/get/all");
    let us = await response.json();
    console.log(us);
    setUsers(us);
}
```

POST:

```
useEffect(() => {
    console.log(genre);
    GetArtists();
}, []);

const GetArtists = async () => {
    let response = await fetch("http://localhost:5000/api/genres/get/artists", {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            genreid: genre._id
        })
    });
    let art = await response.json();
    console.log(art);
    setArtistis(art);
    console.log(artists);
}
```

8. React Router

Az „src” mappában található „Content.tsx” fájl tartalmazza az elérhető útvonalakat.

```
const Main = () => (
    <Margin>
        <BrowserRouter>
            <Switch>
                <Route exact path='/' component={Home} />
                <Route exact path='/login' component={Login} />
                <Route exact path='/registration' component={Registration} />
                <Route exact path='/search' component={Search} />
                <Route exact path='/genre' component={GenrePage} />
                <Route exact path='/artist' component={Artist} />
                <Route exact path='/album' component={Album} />
                <Route exact path='/profile' component={Profile} />
                <Route exact path='/playlist' component={Playlist} />
                <Route exact path='/favouritesongs' component={FavouriteSongs} />
                <Route exact path='/collections' component={Collections} />
                <Route exact path="/findusers" component={FindUsers} />
            </Switch>
        </BrowserRouter>
    </Margin>
)
```

9. Átirányítás

Az átirányítások tipikusan egy kártyára kattintva történnek meg. A kártyákat egy <Link> tag-be helyeztem el. Ebben a tag-ben megadható az átirányítási útvonal, ezenfelül egy állapot is elküldhető a következő oldalra.

```
<Link to={{
    pathname: '/album',
    state: album
}}  key={album._id} style={{textDecoration: 'none', color: 'white'}}>
    <Card key={album._id}>
        <CardImage></CardImage>
        <CardText>
            <h4>{album.name}</h4>
        </CardText>
    </Card>
</Link>
```

10. Adatok listázása

Az alkalmazásban listázódnak műfajok, előadók, albumok, dalok és felhasználók is. A kapcsolódó komponensekben megtalálható egy useState hook-ban lévő tömb. Ezen a tömbön a return jsx részében a map() függvény segítségével iterálok végig.

```
<CardContainer>
    {albums.map(album => (
        <Link to={{
            pathname: '/album',
            state: album
        }}  key={album._id} style={{textDecoration: 'none', color: 'white'}}>
            <Card key={album._id}>
                <CardImage></CardImage>
                <CardText>
                    <h4>{album.name}</h4>
                </CardText>
            </Card>
        </Link>
    ))}
</CardContainer>
```

11. UI design

Minden komponens kinézetét egyedileg készítettem el a styled components használatával.

A gyakorlatban ez úgy valósul meg, hogy egy const-nak adok css stílust, majd ezt a const-ot a komponenseim jsx részénél tag-ként tudom használni.

```
export const Card = styled("div")`
    display: grid;
    grid-template-columns: 200px;
    grid-template-rows: 200px 50px;
    grid-template-areas: "image" "text";
    border-radius: 18px;
    background: black;
    box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.9);
    text-align: center;
    transition: 0.5s;
    cursor: pointer;
    margin: 25px 35px;
    &:hover {
        transform: scale(1.2);
        box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.6);
    }
`;
```

Gurzó Dávid
MK9FBF

BACK-END DOKUMENTÁCIÓ

1. Leírás

Az alkalmazás back-end része REST API-ként működik, JSON formátumú információkkal szolgálja ki a kliens oldalt.

A front-end előre meghatározott endpoint-okon keresztül éri el, majd ezek hatására a megfelelő middleware-ek futnak le. A middleware-ek elsődleges feladata minden esetben valamilyen adatbázissal kapcsolatos művelet.

2. Telepítés és indítás

1. Navigáljunk a projekt „music-playr-backend” nevű mappájába
2. Töröljük a „package-lock.json” fájlt
3. Adjuk ki az *„npm install”* parancsot
4. Adjuk ki az *„npm run dev”* parancsot
5. Az alkalmazás a http://localhost:5000 -en érhető el

Az alkalmazás forráskódja megtalálható a GitHub-on is. Innen klónozható a *„git clone https://github.com/dgurzo/music-playr.git”* parancs segítségével.

3. Technológia

A back-end megvalósításához használt eszközök:

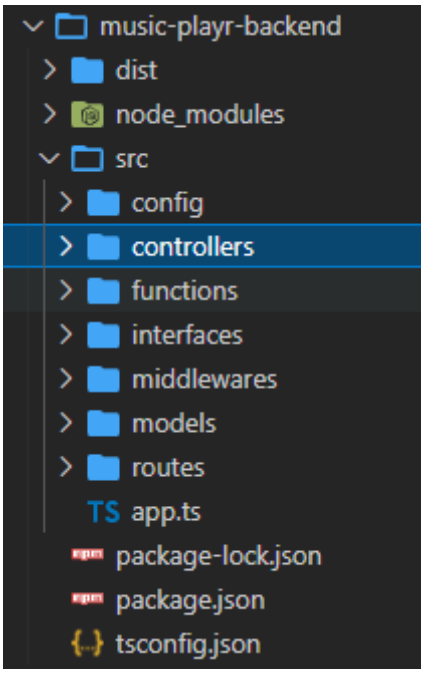
- NodeJS
- Express keretrendszer
- MongoDB
- TypeScript

4. Projekt struktúra

A gyökérmappában található app.ts fájl tartalmazza az alkalmazás alap beállításait, definícióját.

A MongoDB sémáihoz szükséges definíciók az interfaces és models mappákban találhatók. A controllers mappában találhatók az egyes táblákhoz

tartozó middleware-ek. A routes mappa tartalmazza az endpoint-ok middleware-ekkel való összerendelését.



5. Adatbázis diagram

Users tábla:

- Az adatbázisban szereplő felhasználókat tartalmazza.
- Regisztrációkor a user adatai ide kerülnek be.
- Bejelentkezéskor innen történik a jelszó ellenőrzése.

Genre, Artist, Album, Song táblák:

- Az adatbázisban kereshető, böngészhető adatokat tartalmazzák.
- Külső kulcsok használatával vannak összekapcsolva.
- A lent található képen ezek a kapcsolatok jelölve vannak.

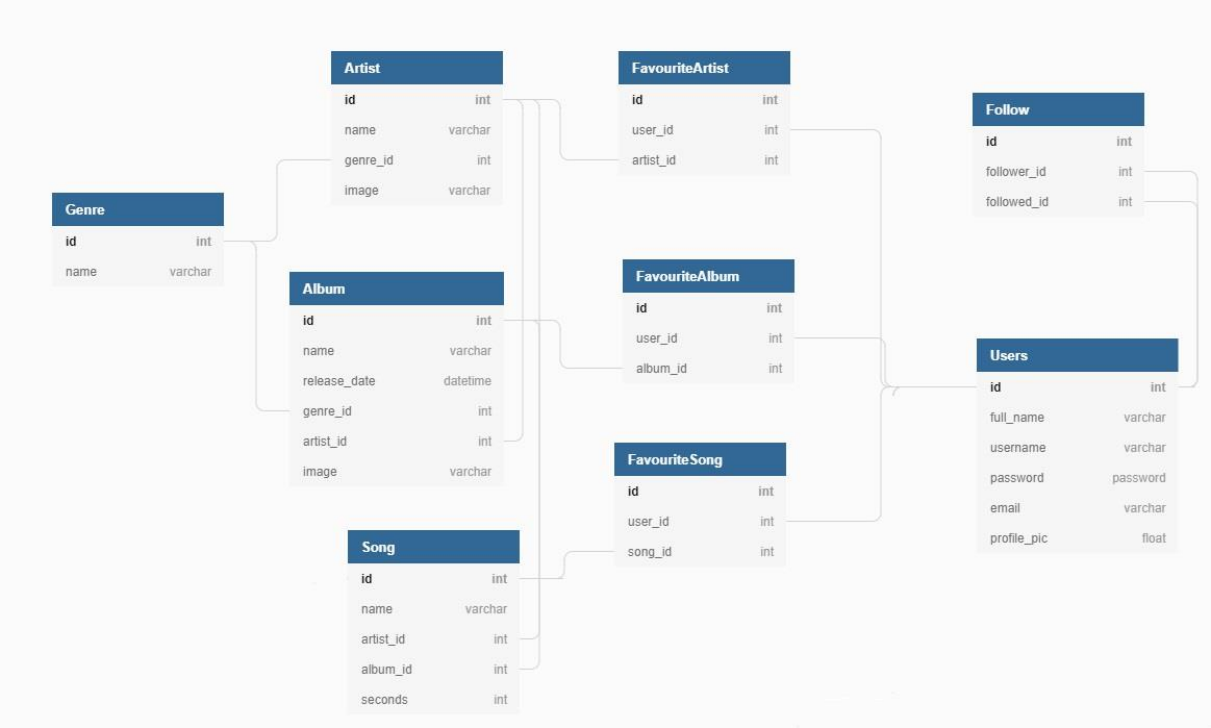
FavouriteArtist, FavouriteAlbum, FavouriteSong táblák:

- A kedvelés funkció megvalósításához szükséges táblák.
- Kapcsolótáblák, melyek összekötnek egy user-t egy előadóval, albummal vagy dallal.

Follow tábla:

- A felhasználóknak lehetőségük van követni egymást.
- A follow tábla két user-t kapcsol össze.

- A felhasználókra mutató foreign key-ek nem egyezhetnek meg.



6. API leírása (endpoint-ok)

USER

POST: /users/register

Leírás: Regisztráció.

Body: full_name, username, password, email

Válasz: User objektum.

POST: /users/login

Leírás: Bejelentkezés.

Body: username, password

Válasz: token, user objektum.

GET: /users/get/all

Leírás: Minden user lekérdezése.

Válasz: Felhasználók tömbje.

GENRE

GET: /api/genres/get/genres

Leírás: Minden műfaj lekérdezése.

Válasz: Műfajok tömbja.

POST: /api/genres/get/artists

Leírás: Adott műfajhoz tartozó előadók lekérdezése.

Body: genreid

Válasz: Műfajok tömbje.

POST: /api/genres/get/albums

Leírás: Adott műfajhoz tartozó albumok lekérdezése.

Body: genreid

Válasz: Albumok tömbje.

ARTIST

POST: /artist/get/albums

Leírás: Adott előadó albumainak lekérdezése.

Body: artistid

Válasz: Albumok tömbje.

POST: /artist/get

Leírás: Az előadó adatainak lekérdezése.

Body: artistname

Válasz: Artist objektum.

ALBUM

POST: /album/get

Leírás: Az album adatainak lekérdezése.

Body: albumname

Válasz: Album objektum.

POST: /album/get/songs

Leírás: Az album dalainak lekérdezése.

Body: albumid

Válasz: Dalok tömbje.

FOLLOW

POST: /follow/start

Leírás: Bejelentkezett user követni szeretne egy másikat.

Body: followerid, followedid

Válasz: A tábla új bejegyzése.

POST: /follow/getfollows

Leírás: Adott felhasználó által követett userek lekérdezése.

Body: userid (aki követ)

Válasz: Follow tábla bejegyzés összefésülve a User tábla bejegyzéssel.

Gurzó Dávid
MK9FBF

FAVOURITEARTIST

POST: /favouriteartist/like

Leírás: Előadó like-olása.

Body: userid, artistid

Válasz: A tábla új bejegyzése.

POST: /favouriteartist/getfavouriteartists

Leírás: A felhasználó által kedvelt előadók lekérdezése.

Body: userid

Válasz: FavouriteArtist tábla bejegyzés összefésülve az Artist táblával.

FAVOURITEALBUM

POST: /favouritealbum/like

Leírás: Album like-olása.

Body: userid, albumid

Válasz: A tábla új bejegyzése.

POST: /favouritealbum/getfavouritealbums

Leírás: A felhasználó által kedvelt albumok lekérdezése.

Body: userid

Válasz: FavouriteAlbum tábla bejegyzés összefésülve az Album táblával.

FAVOURITESONG

POST: /favouritesong/like

Leírás: Dal like-olása.

Body: userid, songid

Válasz: A tábla új bejegyzése.

POST: /favouritesong/getfavouritesongs

Leírás: A felhasználó által kedvelt dalok lekérdezése.

Body: userid

Válasz: FavouriteSong tábla bejegyzés összefésülve az Song táblával.

7. Middleware-ek

A middleware-ek feladata az adatázis manipulálása, bejegyzések létrehozása, törlése, továbbá adatok kinyerése. Ezeket tipikusan a MondoDB find, findOne, delete, deleteOne és aggregate metódusai segítségével teszik.

```
const getArtist = (req: Request, res: Response, next: NextFunction) => {
    Artist.findOne({name: req.body.artistname})
    .exec()
    .then(artist => {
        return res.status(200).json(artist);
    })
    .catch(error => {
        return res.status(500).json({
            message: error.message,
            error
            });
    });
}

const getAlbums = (req: Request, res: Response, next: NextFunction) => {
    Album.find({_artist_id: req.body.artistid})
    .exec()
    .then(album => {
        return res.status(200).json(album);
    })
    .catch(error => {
        return res.status(500).json({
            message: error.message,
            error
        });
    })
}
```