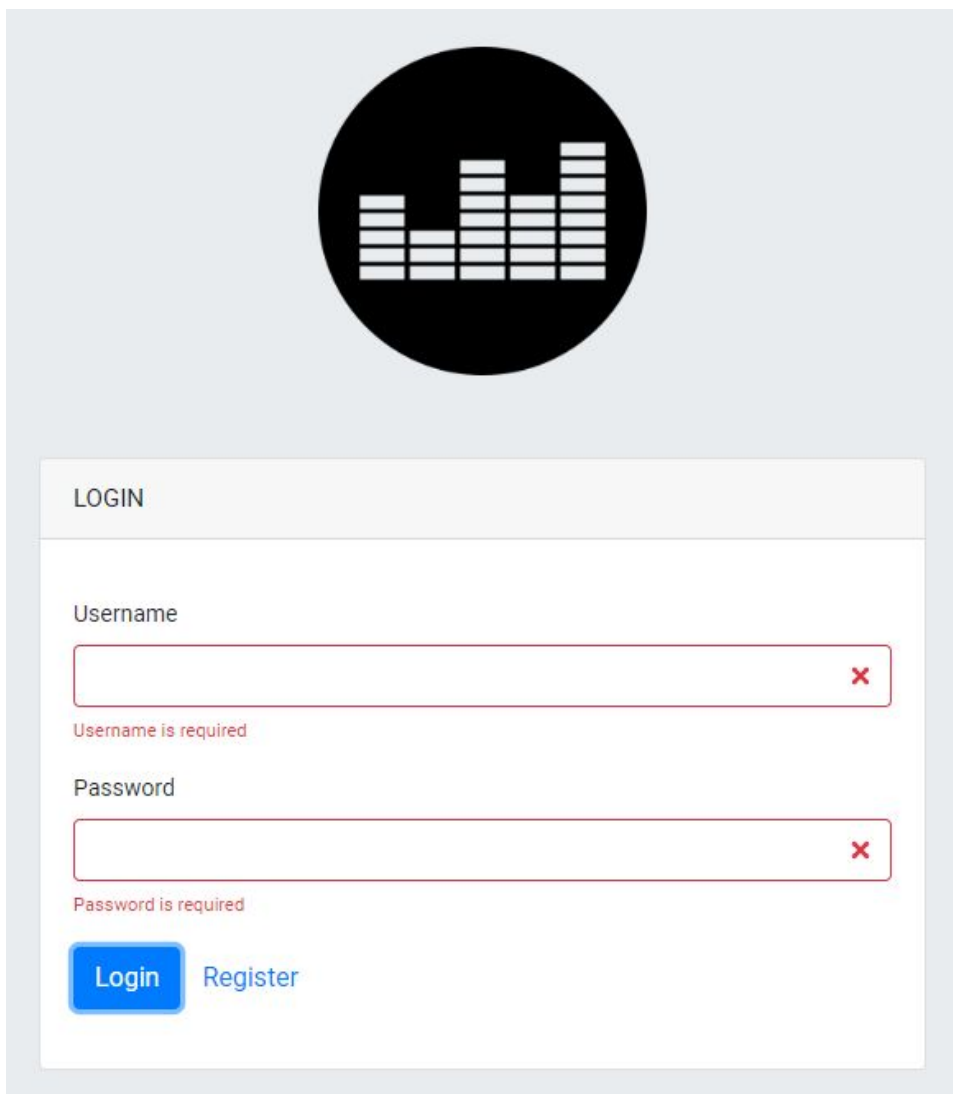


Date: 10.07.2020	Subject: Internet Engineering project report	Name: Damian Rodziewicz
---------------------	---	----------------------------

- **Subject:**

Main purpose of this SPA app is music and albums repository. App was created with Deezer streaming platform open API and Java spring backend made in previous term . User can create account, search for tracks and albums, add them to favourite lists and listen to samples.

- **Login components**



LOGIN

Username


Password

Login Register

*Login component with form control.*

Login component has been build with a single form with two fields. Both fields are secured by from control so they cannot be submitted blank. After submitting form and passing verification user is granted with token that is stored during application session.

- **Register component**



REGISTER

Username

✖

Username is required

Password

✖

Password is required

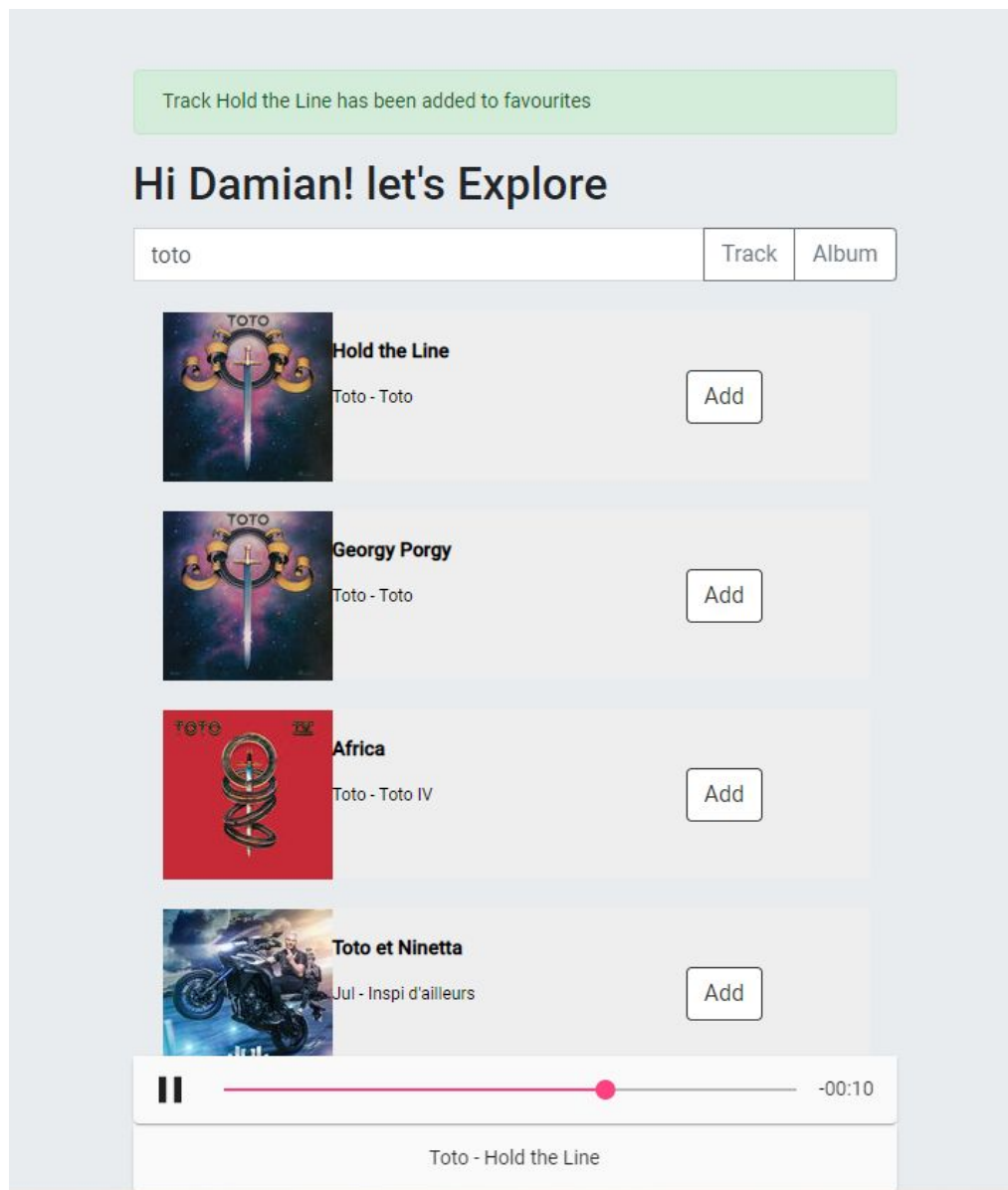
Register

Cancel

*Register component with form control.*

Register components purpose is to add new user to database. Form used in this component is similar to one used in login component. Submitting form makes application to push a POST request with users credentials.

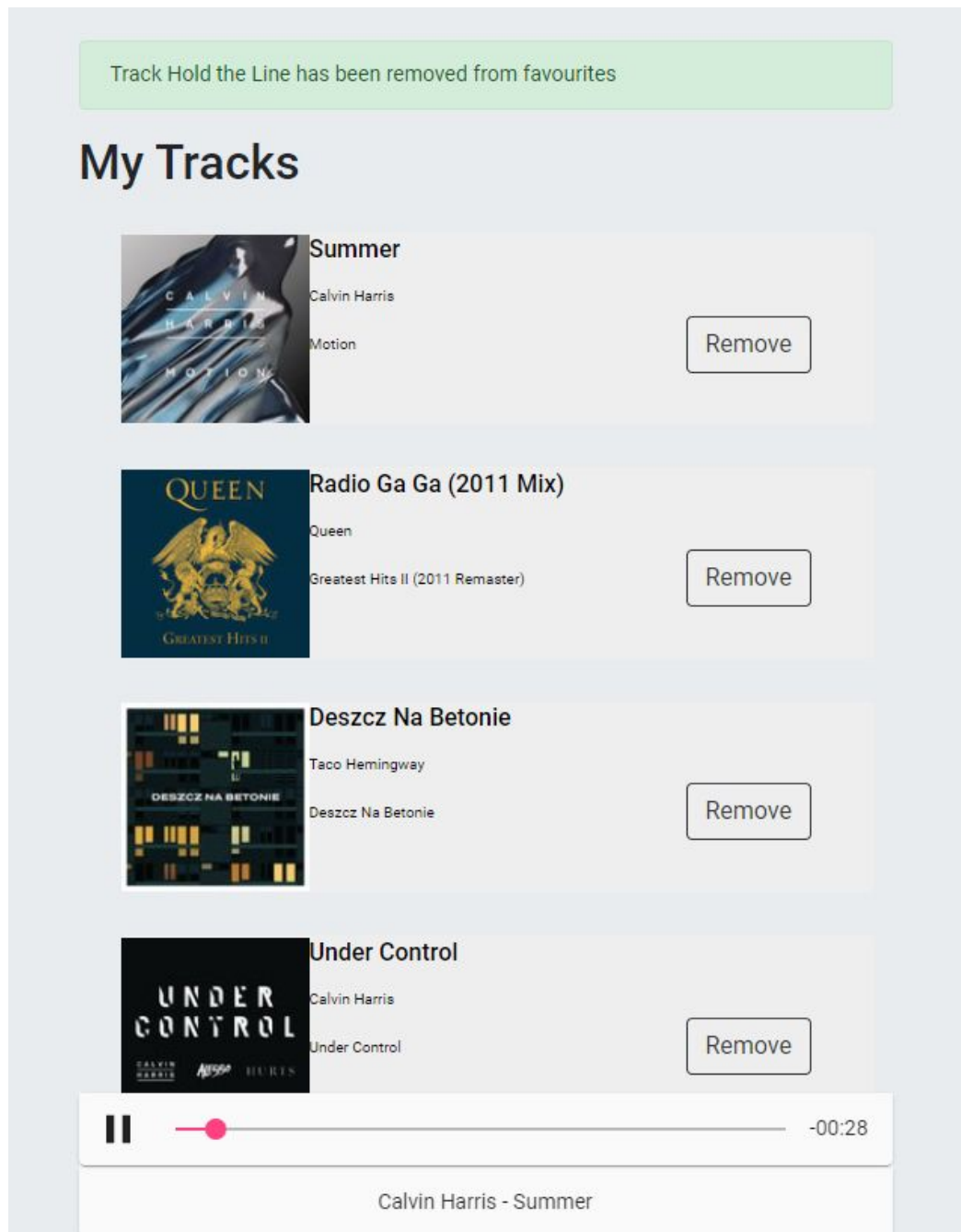
- **Explore page**



*Main page of application*

Explore page is the main page of application. Search bar is the main feature of this page. User is typing keywords referred to artists, album or track and results are printed below. From there user by clicking on one of the results can listen to 30 second sample of the track and add it to favorite list. Clicking add pushes POST method with informations about track. After successfully added track alert with confirmation is presented. Search bar was made with a single field form and both buttons purpose is to submit a form depending of what user want to find.

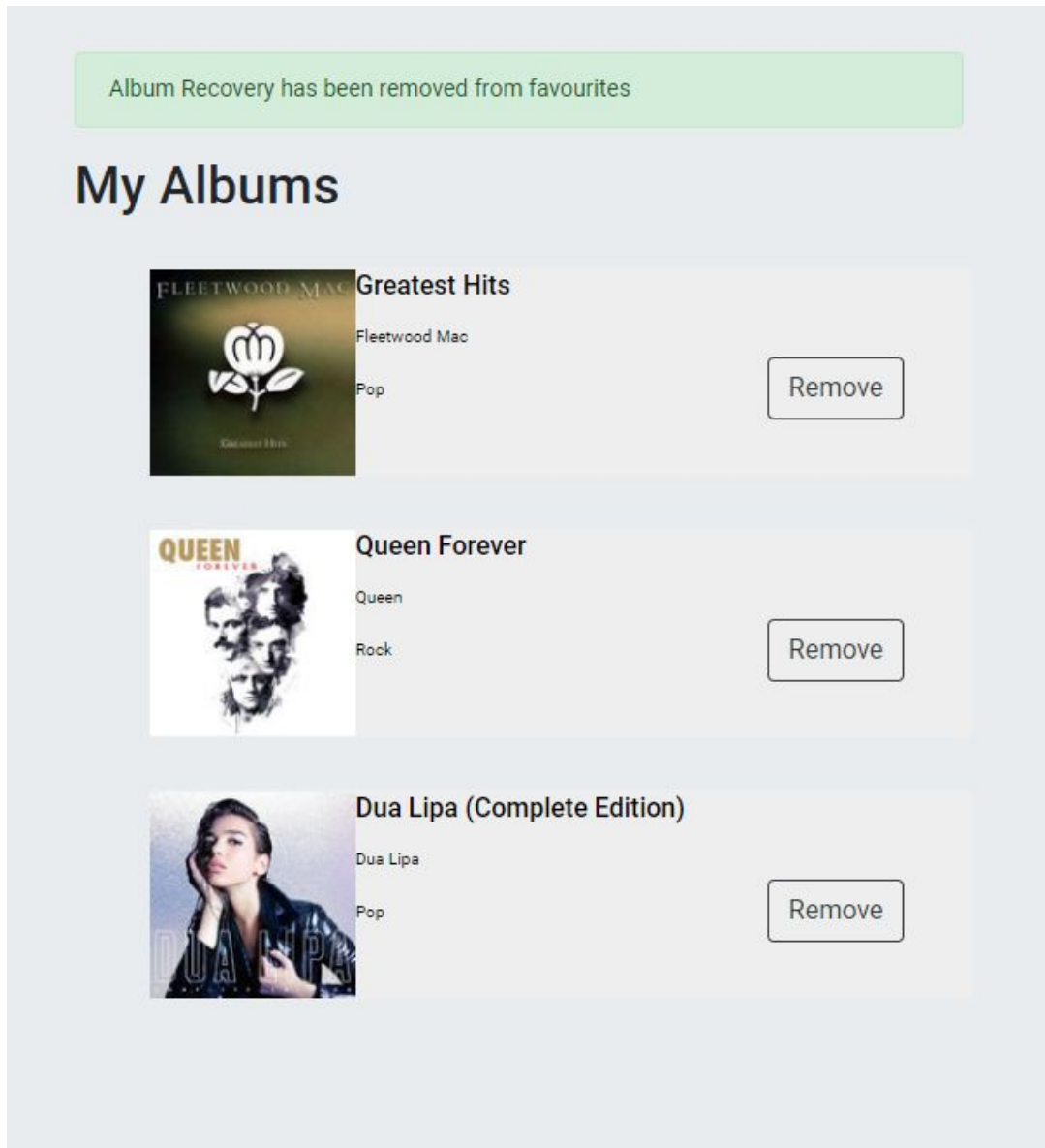
- **My Tracks page**



*My tracks page with favourite tracks*

My Tracks page stores all the tracks we added by explore component. Data is presented by making GET request from the backend API. On this page user can also listen to the samples and manage the list by removing positions with Remove buttons. Successfully deleted track will lead to alert with confirmation.

- **My Albums page**



*My Tracks page with data and alert confirming removal.*

My Albums purpose is same as My Tracks but without the player.

- **Passing token with every request.**

```
intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  
  
    let currentUser = this.authService.currentUserValue;  
    if(currentUser && currentUser.token) {  
        request = request.clone( update: {  
            setHeaders: {  
                Authorization: `Bearer ${currentUser.token}`  
            }  
        });  
    }  
}
```

*Body of intercept method responsible for setting headers.*

Intercept method in `_helpers/jwt.interceptor.ts` is responsible for setting header with currently logged user JWT Token. Method checks if user is currently logged and have token then Authorization token is setted.

- **Unauthorized error handling.**

```
intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  
    return next.handle(request).pipe(catchError( selector: err =>{  
        if(err.status === 401){  
            this.authService.logout();  
            location.reload( forcedReload: true);  
        }  
  
        const error = err.error.message || err.statusText;  
        return throwError(error);  
    })))  
}
```

*Body of method responsible for handling unauthorized errors.*

When user's token expires he cannot perform request any longer so `logout()` method from `AuthorizationService` is being called and user is send back to login page. Also alert with error information is being presented.

- Pages accessible after login in:

```
const routes: Routes = [  
  { path: '', component: HomeComponent, canActivate: [AuthGuard] },  
  { path: 'login', component: LoginComponent },  
  { path: 'register', component: RegisterComponent },  
  { path: 'tracks', component: TracksComponent, canActivate: [AuthGuard] },  
  { path: 'albums', component: AlbumsComponent, canActivate: [AuthGuard] },  
  // otherwise redirect to home  
  { path: '**', redirectTo: '' }  
];
```

*Defined routes in app-routing module*

```
canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot){  
  const currentUser = this.authService.currentUserValue;  
  if(currentUser){  
    return true;  
  }  
  
  this.router.navigate([ 'login' ], { extras: { queryParams: { returnUrl: state.url }}});  
  return false;  
}
```

*Body of method protecting user from unauthorized activities*

Only *login* and *register* pages are available without authorization. Rest of them is protected with mechanism that checks if user is currently logged in. If true value is returned user can navigate through app in other case user is automatically redirected to login page.

- **Storing user credentials**

```
login(username, password) {  
  return this.http.post<any>({ url: `${API_URL}authenticate`, body: { username, password}})  
    .pipe(map( project: user =>{  
      localStorage.setItem('currentUser', JSON.stringify(user));  
      this.currentUserSubject.next(user);  
      return user;  
    }));  
}
```

*Body of login method that stores the credentials of currently logged user*

After submitting form in login page method pushing request POST is called and response with user credentials along with token mapped to user model and placed in local storage.

- **Technology**

Application was entirely made with Angular 9 for frontend part with Java Spring framework for backendpart. For CSS was used Bootstrap Library.