Проектна задача по предметот

Напреден веб дизајн

Апликација за стримање на музика (MixTape)

Изработиле:

Иван Јанкоски 201105

Ксенија Алулоска 191048

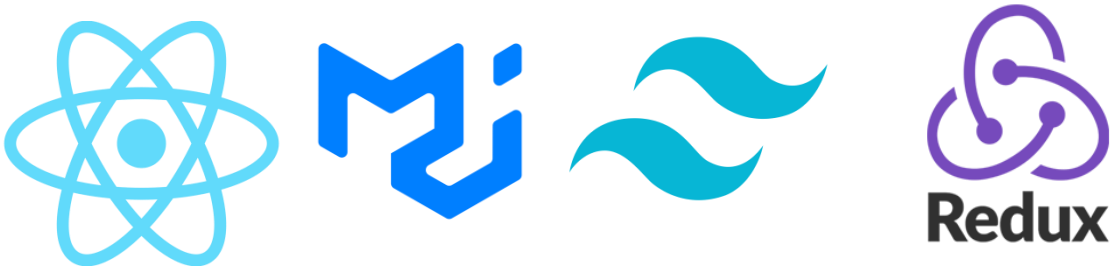
Содржина

За MixTape

MixTape претставува платформа за стримање на музика базирана на глобалната музичка стриминг платформа Spotify. Главната цел на овој проект кој го изработивме е да им се овозможи на сите корисници да слушаат музика потполно бесплатно и без рекалами.

Главни техологии со кои е изградена апликацијата:

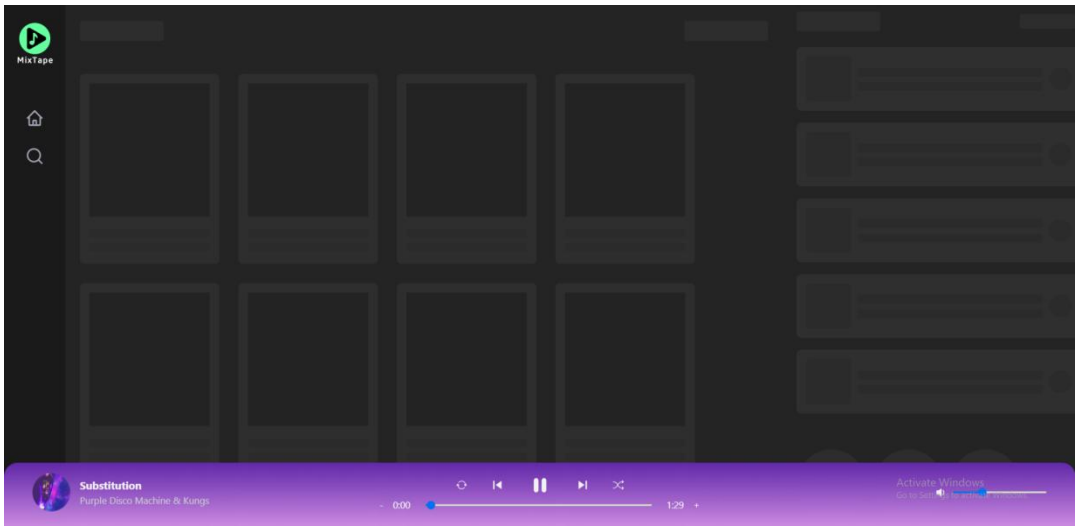
- React – Апликацијата која ја изградивме е базирана на страници и компоненти со помош на React.
- Redux – Податоците со кои што работиме во апликацијата ги менаџираме во еден голем контејнер со помош на Redux.
- Tailwind – CSS preprocessor кој нуди готови класи за стилизирање на компонентите.
- MaterialUI – React библиотека која нуди готови компоненти за реискористување.
- RapidAPI – Податоците со кои работиме во апликацијата ги преземаме од отворена API структура за податоци RapidAPI односно по конкретно го користиме сервисот Shazam API.



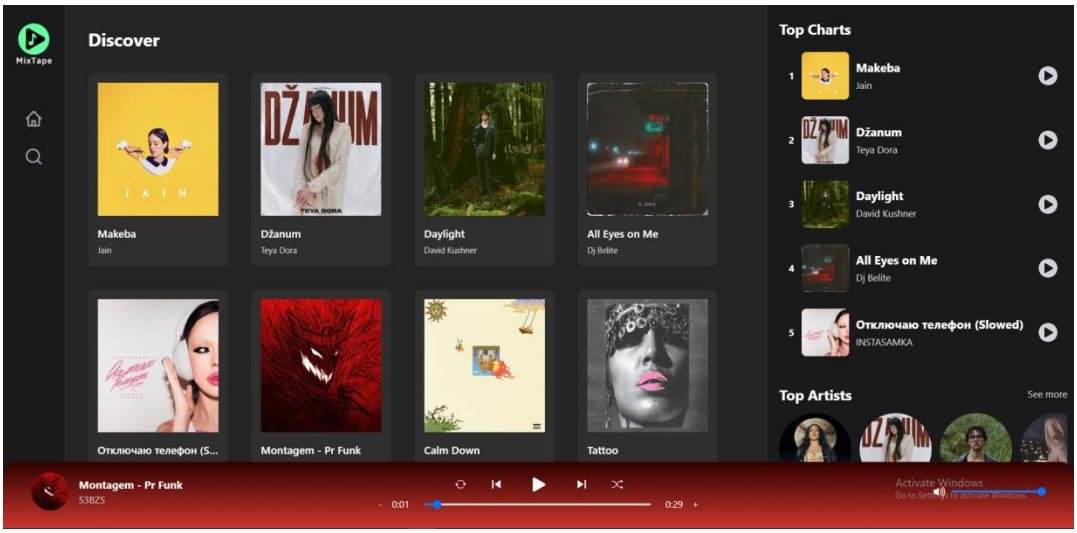
1. Страници

1.1 Discover page

Ова е почетниот екран на апликацијата. На средината се прикажуваат 20те моментално најпопуларни песни во светот според глобалната музичка листа за тековниот месец. Во десниот дел на страницата се прикажани 5 најслушани песни во светот како и 5те најпопуларни артисти за месецот. При повлекување на податоците од даденото API екранот прикажува т.н скелетон со цел да му се даде дознаење на корисникот дека податоците се вчитуваат. Песната се избира со клик на картичката на посакуваната песна.



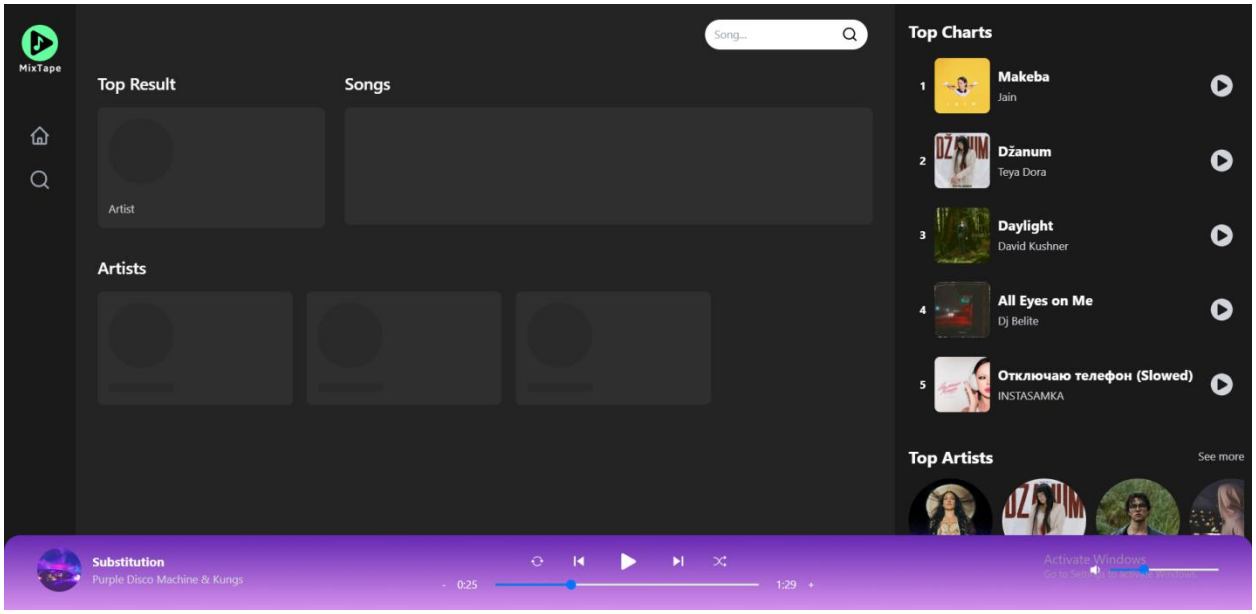
Вака изгледа почетната страна на апликацијата при вклучување додека се вчитуваат податоците



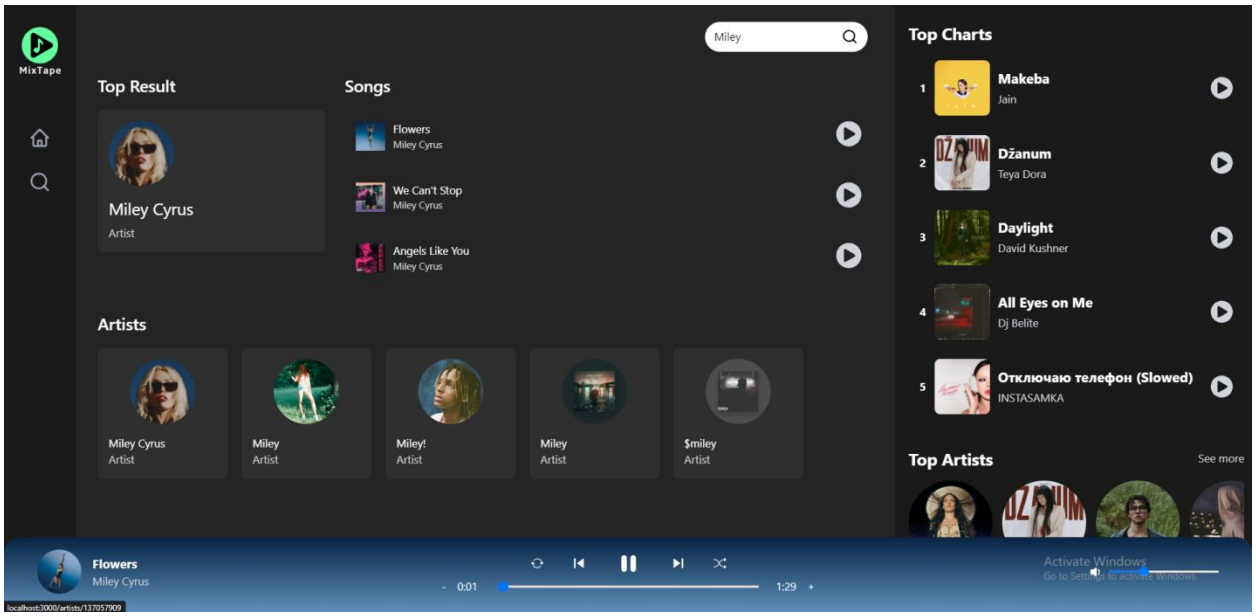
Вака изгледа почетната страна на апликацијата.

1.2 Search page

Оваа страница прикажува екран во кој може да се пребаруваат сите автори и песни при што резултатот кој се враќа веднаш ќе биде прикажан на екранот и може од тука директно да се пушти посакуваната песна. Пребарувањето е директно поврзано со API, односно на промена на вредноста во инпут полето се праќа барање до API-то и соодветно се враќа резултат.



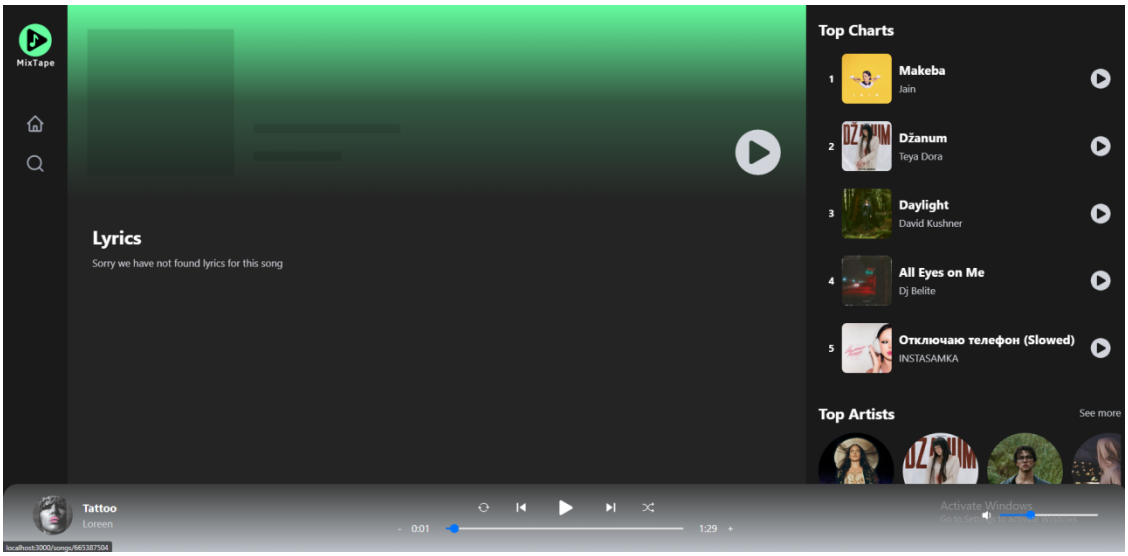
Изглед на страницата доколку не е внесена никаква вредност во инпут полето



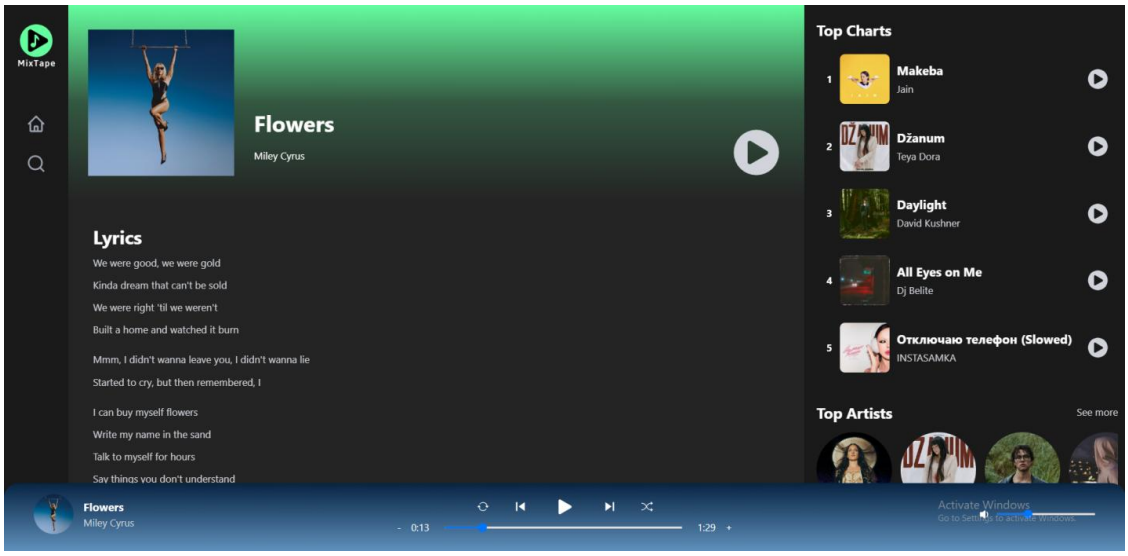
Изглед на екранот при внес на вредност во полето

1.3 Song details page

При клик на некоја песна апликацијата пренасочува кон детален поглед за таја песна на која е прикажана името на песната, насловот и артистот кој ја изведува. Дополнително доколку за песната е пронајден lyrics истиот ќе биде прикажан во продолжение. Со клик на копчето кое е прикажано десно од сликата на песната истата се предава на музичкиот плеер и се пушта.



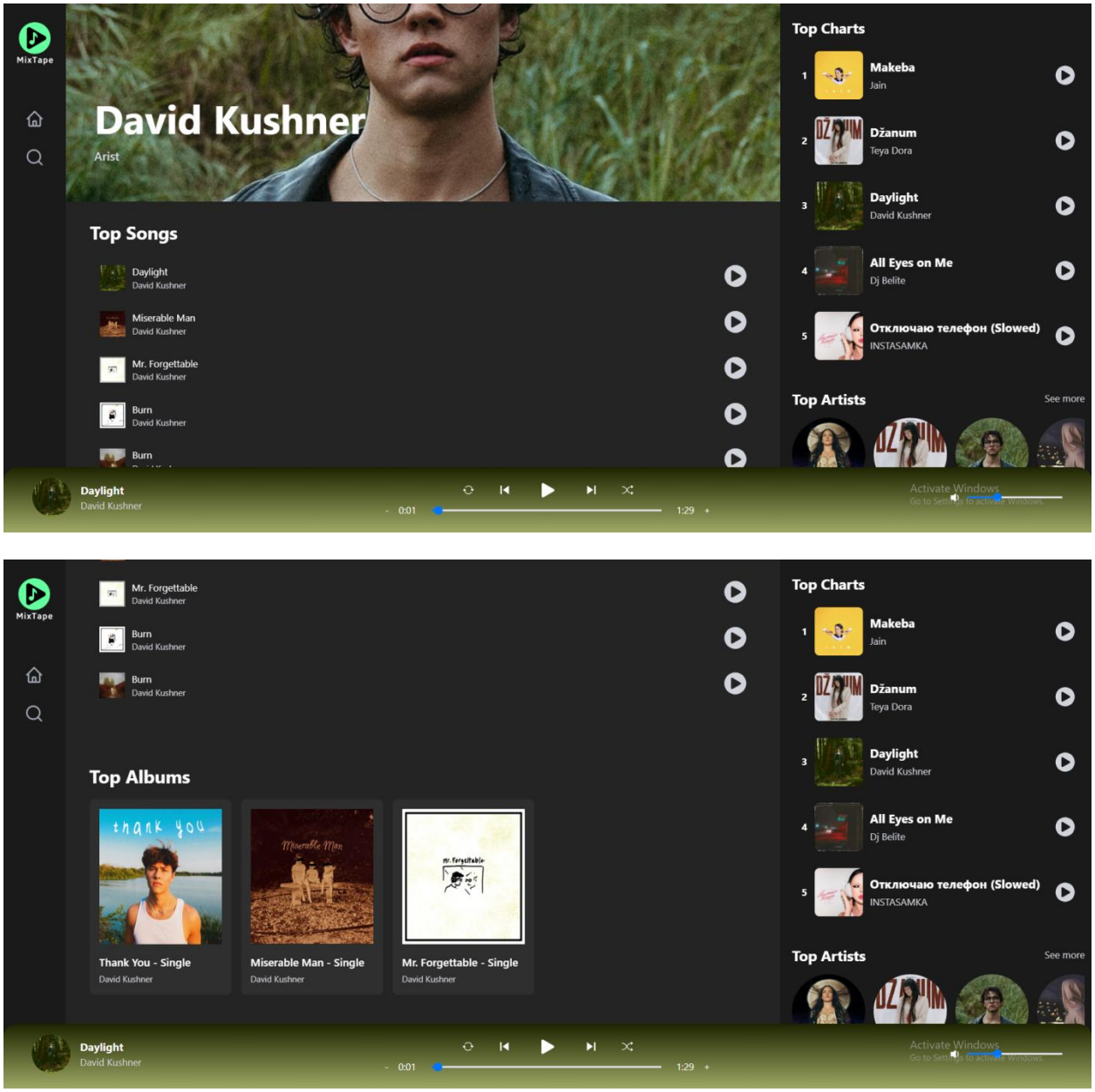
Изглед на екранот додека се вчитува избраната песна



Изглед на страната SongDetails кога е избрана песна

1.4 Artist details page

Слично како и за детали на песната апликацијата нуди и детален поглед на артистот. За артистот се прикажува неговата слика во форма на позадина и неговото име. Во продолжение се прикажуваат неговите 5 најслушани песни и неговите албуми.



Ова е изглед на екранот на детален поглед на артист

2. Компоненти

2.1 MusicPlayer

Музичкиот плеер во апликацијата ја презема избраната песна на корисникот и ја пушта. Составен е од повеќе делови и има неколку функционалности.

Play/Pause – со клик на копчето се пушта или паузира песната.

Prev/Next – за пуштање на следна или претходна песна која била избрана.

Repeat – доколку ова копче е активирано песната која е тековно пуштена ќе се повторува се додека не биде исклучено ова копче.

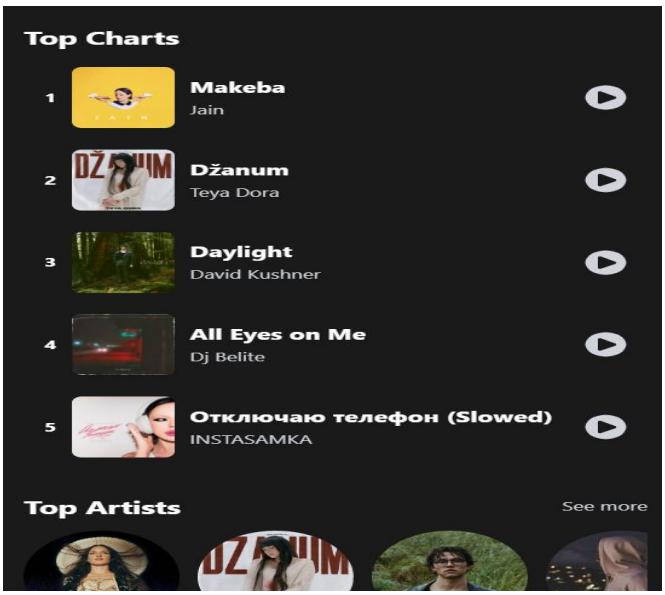
Shuffle – со активирање на ова копче музичкиот плеер песните ги бира рандом односно нема да одат по редослед.

Seekbar – лента на која се прикажува времетраењето на песната и истата може да се премотува.

Volume bar – лента за зголемување и намалување на звукот.

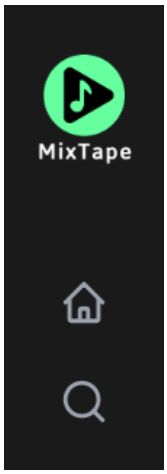
2.2 TopPlay

Оваа компонента ги прикажува најдобрите 5 песни и артисти за тековниот месец. Внатре во компонентата постојат други две различни компоненти SongCard и ArtistCard.



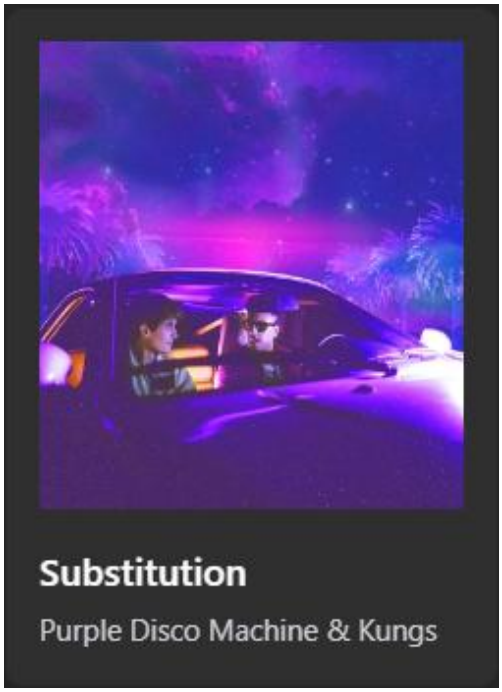
2.3 Sidebar

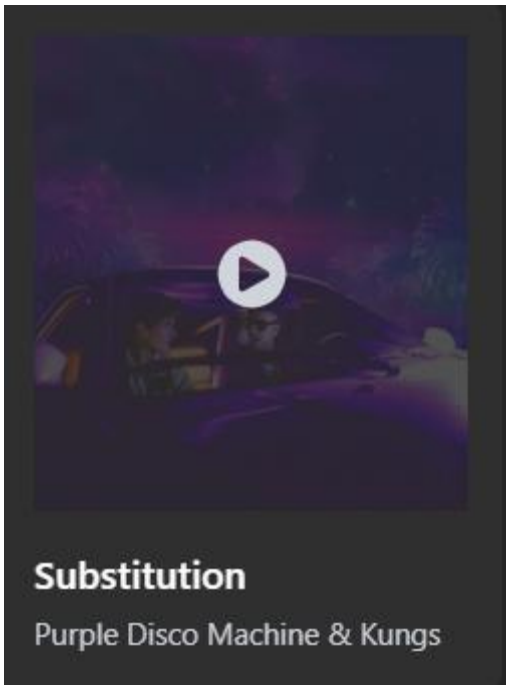
Оваа компонента претставува навигацијата на апликацијата прикажана на левата страна од екранот. Може да се навигира кон почетната страна и кон страната за пребарување на песни.



2.3 SongCard

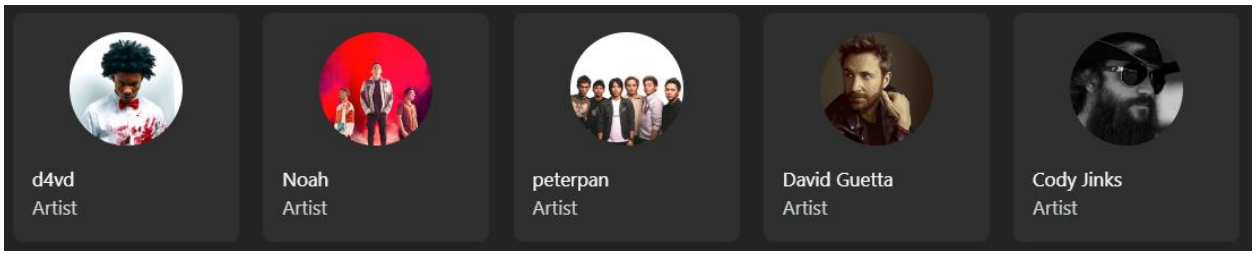
Оваа компонента се користи за приказ на песните на почетната страница на апликацијата. На левата слика е прикажана картичката во почетна состојба, а на десната слика картичката кога курсорот е поставен над неа.





2.4 ArtistCard

Оваа компонента ги претставува авторите кои се добиени од пребарувањето во страницата за пребарување.



Каричката содржи слика на авторот и неоговото име кое е во вид на линк и при клик на него се пренасочува кон детален поглед на авторот.

3. Redux Store

Како што спомнавме погоре во апликацијата за менаџирање на податоците користиме Redux. За поедноставно манипулирање со податоците оваа функционалност ја раздвовиме на два дела и тоа првиот дел е за менаџирање на API рутите и дел за манипулирање со податоците во апликацијата поточно со музичиот плеер.

3.1 shazamCore

Делот од Redux за полесно менаџирање на API рутите. Оваа функционалност ја имплементиравме така што со помош на redux toolkit направивме baseUrl и endpoints во кои по потреба додаваме нови рути во зависност од тоа кои податоци сакаме да ги повлечеме.

```
import {createApi ,fetchBaseQuery} from "@reduxjs/toolkit/query/react"


export const shazamCoreApi = createApi({
    reducerPath:'shazamCoreApi',
    baseQuery: fetchBaseQuery({
        baseUrl:'https://shazam.p.rapidapi.com/',
        prepareHeaders: (headers) => {
            headers.set('X-RapidAPI-Key','764b57a247mshecfbeff47cba933p1bbc3ajsnb0e8e1a57c61');
            return headers;
        }
    }),
    endpoints: (builder) => ({
        getTopCharts: builder.query({ query : () => '/charts/track'}),
        getSongDetails: builder.query({ query: (songid) => `/songs/get-details?key=${songid}`}),
        getArtistSongs: builder.query({ query: (artistid) => `/artists/get-top-songs?id=${artistid}` }),
        getSearch: builder.query({query: (term) => `/search?term=${term}`}),
        getArtistDetails: builder.query({query: (adamid) => `/artists/get-summary?id=${adamid}`}),
        getSongDetailsV2: builder.query({query: (songid) => `/songs/v2/get-details?id=${songid}`}),
        getArtistAlbums: builder.query({query: (artistid) => `/artists/get-summary?id=${artistid}`})
    })
})

export const { useGetTopChartsQuery,
useGetSongDetailsQuery,useGetArtistSongsQuery,
useGetSearchQuery, useGetArtistDetailsQuery, useGetSongDetailsV2Query, useGetArtistAlbumsQuery } = shazamCoreApi;
```

3.2 playerSlice

PlayerSlice како што самото име кажува е делче од целиот Redux контејнер на податоци и го користиме за манипулирање на податоците со музичкиот плеер пример за поставување на активна песна, пуштање на следна песна и сл.

На сликата долу е прикажан дел од кодот на оваа функционалност. Конкретно на сликата е прикажан кодот за поставување на тековно активна песна. Функцијата за поставување на активна песна прима два аргументи тоа се state (моменталната состоба на контејнерот) и action (акцијата која се извршува поставување на песната).

```
import { createSlice } from '@reduxjs/toolkit';

const initialState = {
  currentSongs: [],
  currentIndex: 0,
  isActive: false,
  isPlaying: false,
  activeSong: {},
  currTime:0,
  genreListId: '',
};

const playerSlice = createSlice({
  name: 'player',
  initialState,
  reducers: {
    setActiveSong: (state, action) => {
      state.activeSong = action.payload.song;

      if (action.payload?.data?.tracks?.hits) {
        state.currentSongs = action.payload.data.tracks.hits;
      } else if (action.payload?.data?.properties) {
        state.currentSongs = action.payload?.data?.tracks;
      } else {
        state.currentSongs = action.payload.data;
      }
      console.log(action.payload)
      state.currentIndex = action.payload.i;
      state.isActive = true;
    },

```