SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RACUNARSTVA

SEMINAR

Youtube Video Player

Daniel Ranogajec

Voditelj: Krešimir Križanović

Zagreb, svibanj 2022.

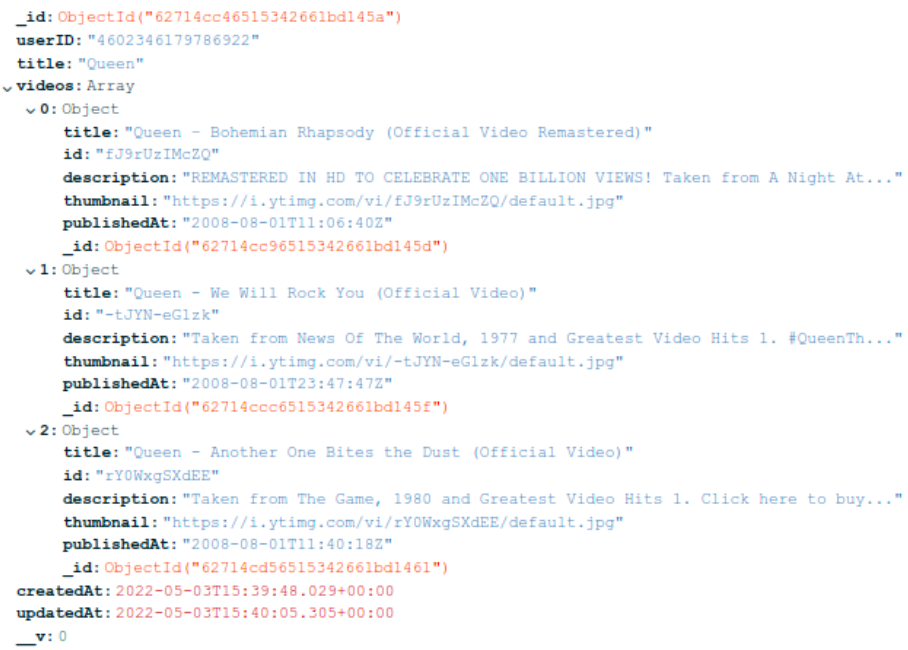
SADRŽAJ

1. Uvod

Cilj ovog seminarskog rada je dublje upoznavanje sa React Framework-om i izrada funkcionalne web-aplikacije. Funkcija aplikacije je izrada playlista te dohvaćanje videa pomoću Youtube-ovog API-ja. Kao što je već naznačeno, frontend aplikacije rađen je u Reactu, a backend u Node.js uz pomoć MongoDB baze podataka. Aplikacije radi tako da se korisnik prijavljuje pomoću Facebook računa, njegovi podaci se spremaju u bazu podataka, zatim korisnik može kreirati više različitih playlista na koje može dodavati videe koji se dohvaćaju pomoću Youtube API-ja. Na početnoj stranici, korisnik će nakon prijave moći odabrati neku od svojih postojećih playlista ili napraviti novu. Kada korisnik odabere playlistu, prikazani su mu svi videi koji su dodani u nju i može ih pokrenuti. Osim toga, korisnik ima opciju za pretraživanje i dodavanje novog videa.

2. Model podataka

Korisnici se na aplikaciju povezuju putem Facebooka, no u bazu podataka se ne spremaju podaci dostupni preko Facebook profila korisnika, već kada korisnik pravi novu playlistu uzima se i pohranjuje Facebook ID korisnika, kako bi tu playlistu mogao dohvatiti samo taj korisnik. Model podataka za playliste je slijedeći: za svaku playlistu spremaju se već objašnjeni korisnički ID, naziv playlist i lista videa. Schema videa sastoji se od naziva, ID-a, opisa, slike (“thumbnaila”) te datuma objavljivanja. Sve te podatke dohvaća se putem Youtubeovog API-ja upisivanjem naziva ili url-a videa u pretraživač na aplikaciji.



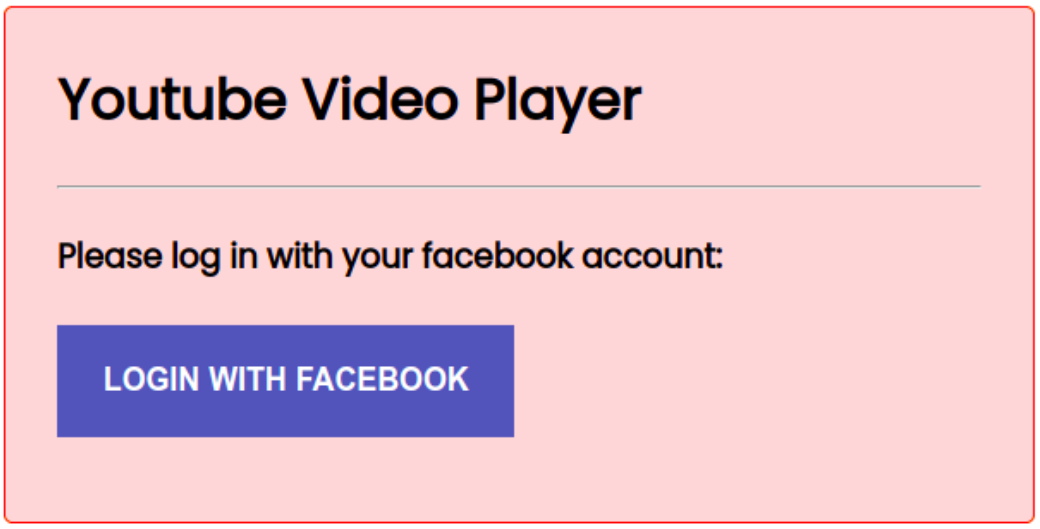
Slika 1. Prikaz podataka u bazi podataka

3. Arhitektura sustava

Arhitektura korištena za izradu web-stranice je MVC (model-view-controller) arhitektura. Ona je u suštini obrazac koji se sastoji od tri komponente, modela, prikaza i upravitelja. Model sadrži podatke, prikaz prikazuje podatke iz komponente model, a kontroler služi kao posrednik u komunikaciji modela i prikaza. Na ovoj web-stranici za rađenje komponente prikaz odnosno frontenda korišten je React. On se direktno spaja na kontroler. Kontroler je u ovom slučaju backend, odnosno server rađen u Node.js-u, kojem je funkcija protok podataka između modela i prikaza. Za model, odnosno bazu podataka korišten je MongoDB, te on služi za procesiranje podataka i spojen je samo na kontroler.

4. Youtube Video Player

Kada korisnik prvi puta otvara web stranicu ili u slučaju da još nije prijavljen ili su obrisani kolačići sesije, stranica ga preusmjerava na komponentu koja služi za prijavljivanje na Facebook.



Slika 2. Prijava na Facebook

```
<Route path='/' exact render={(props) => (<>
{user === null ? (<><FbLogin onLogin={onLogin}/></>):(
  <><AddPlaylist onAdd={createPlaylist}/>
  <Playlists playlists={playlists} onOpen={onOpen} onDeletePlaylist={deletePlaylist}/></>
)}
</>)}/>
<Route path='/video/:id' exact render={(props) => (<>
{user === null ? (<><FbLogin onLogin={onLogin}/></>):(
  <>
  <VideoPlayer videos={videos}/>
  <br />
  <Footer />
  </>
)}
</>)}/>
<Route path='/playlist/:id' exact render={(props) => (<>
  {user === null ? (<><FbLogin onLogin={onLogin}/></>):(
  <>
  <AddVideo onAdd={searchVideo}/>
  <Videos videos={videos} onDelete={deleteVideo}/>
  <br></br>
```

Prikazani kod ilustrira preusmjeravanje na na FbLogin komponentu u slučaju da korisnik nike prijavljen. Frontend zna da je korisnik prijavljen na Facebook račun tako što se nakon prijavljivanja id korisnika sprema u window.localStorage. Na taj način se korisnik neće morati svaki put prijavljivati, nego samo kada se obrišu kolačići sesije.

```
responseFacebook = response => {
  if (response.status === "unknown") {
    return
  }
  this.setState({
    userID: response.userID,
    name: response.name,
    email: response.email,
    picture: response.picture.data.url
  });
  this.props.onLogin(this.state)
};

render() {
  return <div>
  <h3>Please log in with your facebook account:</h3>
  <br />
  <FacebookLogin
    appId="394446275517679"
    autoLoad={true}
    fields="name,email,picture"
    callback={this.responseFacebook}
    />
    </div>;
}
```

Kod iznad prikazuje komponentu FbLogin koja nakon što se korisnik uspješno prijavi na facebook račun poziva onLogin funkciju od roditeljske komponente te joj predaje podatke o korisniku dobivene od Facebooka. U sljedećem kodu prikazana je funkcija onLogin koja sprema dobivene podatke u lokalno spremište preglednika, odnosno window.localStorage.

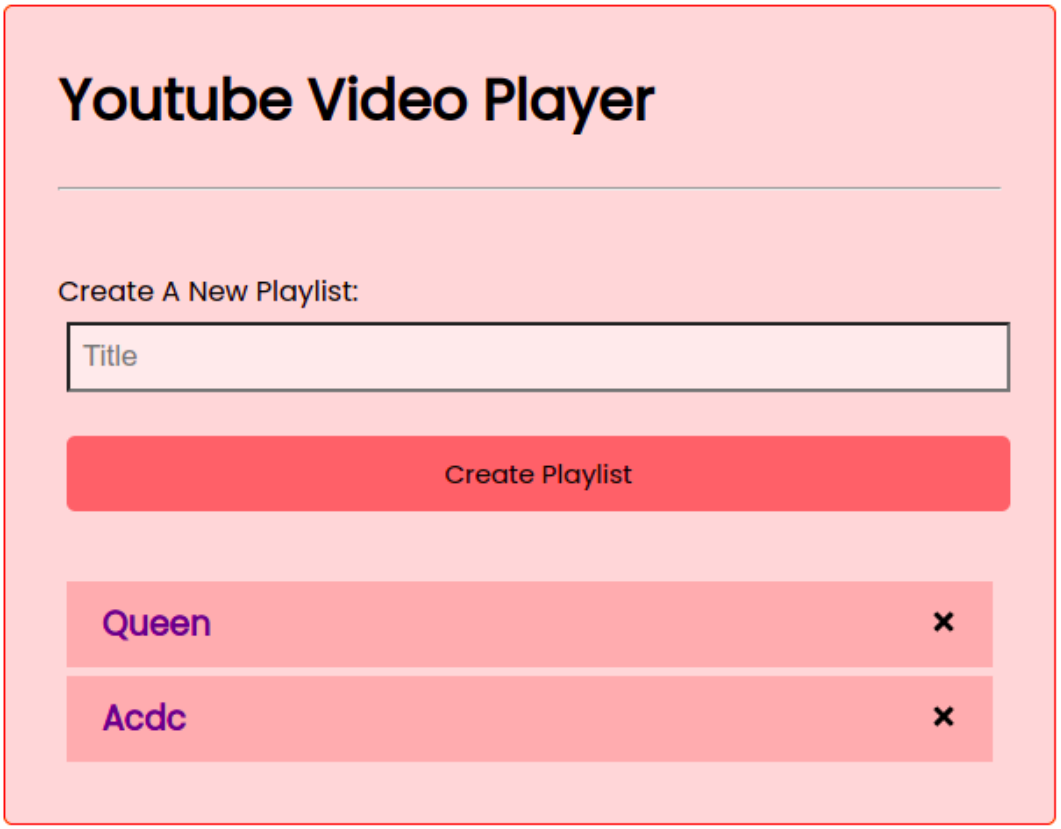
```
const onLogin = async (data) => {
  setUser(data.userID)
  window.localStorage.setItem("user", JSON.stringify(data.userID))
  const playlists = await fetchPlaylists(data.userID)
  setPlaylists(playlists)
}
```

```
useEffect(() => {
  var usr = window.localStorage.getItem("user") === null ?
    (null) : (window.localStorage.getItem("user").replace(/['"]+/g, ''))
  const getPlaylists = async () => {
    if (usr) {
      const playlists = await fetchPlaylists(usr)
      setPlaylists(playlists)
    }
  }
  getPlaylists()
}, [])

const fetchPlaylists = async (usr) => {
  const res = await fetch(`http://localhost:8000/user/${usr}`)
  const data = await res.json()
  return data
}
```

U navedenom kodu prikazan je useEffect hook koji služi da nakon što se web-stranica renderira, traži se “user” u window.localStorage te ako isti postoji, odnosno ako je korisnik prijavljen, onda se dohvaćaju njegove playliste pozivanjem poziva za dohvat playlista sa backenda.

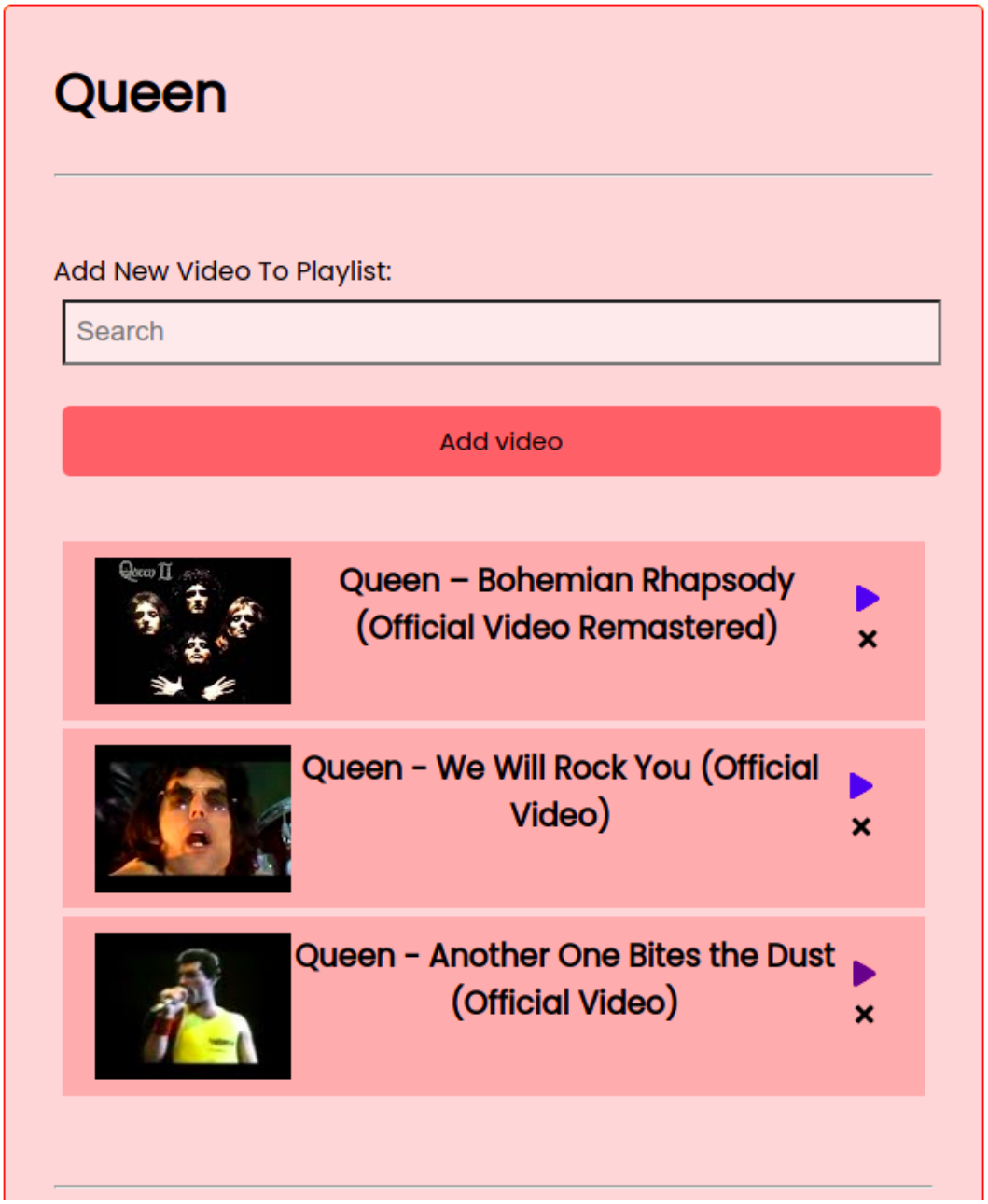
Nakon što se korisnik uspješno prijavi na svoj Facebook račun, na stranici mu se prikazuje komponenta koja služi za kreiranje novih playlista te prikaz već kreiranih, ako ih je korisnik prije izradio.



Slika 3. Komponenta za kreiranje novih playlista i popis izrađenih playlista

Na prikazanoj stranici korisnik u formu upisuje naziv playliste te se ista sprema u bazu podataka.

Jednom kada je playlista napravljena, pojavit će se u popisu playlista u komponenti ispod prethodne. Klikom na naziv playliste ista se otvara, a klikom na “x” se ista briše iz baze podataka.



Slika 4. Otvorena playlista

Kada se playlista otvori, na vrhu se pojavljuje nova forma putem koje se pretražuju i dodaju novi videi u istu. Dodavanje videa funkcionira na slijedeći način. Korisnik upisuje tekst u input kontejner te klikom na submit, odnosno “Add Video”, poziva se backend koji komunicira sa Youtube API-jem te mu šalje upisani naziv videa ili URL u slučaju da korisnik želi tako dodavati videe.

Youtube API kao odgovor šalje JSON objekt koji predstavlja video koji je najviše povezan sa upisanim naslovom i to je najčešće video sa najviše pregleda. Zatim se u backendu taj JSON objekt pretvori u Video objekt koji je opisan u drugom poglavlju te se isti video sprema u bazu podataka pod određenu Playlistu.

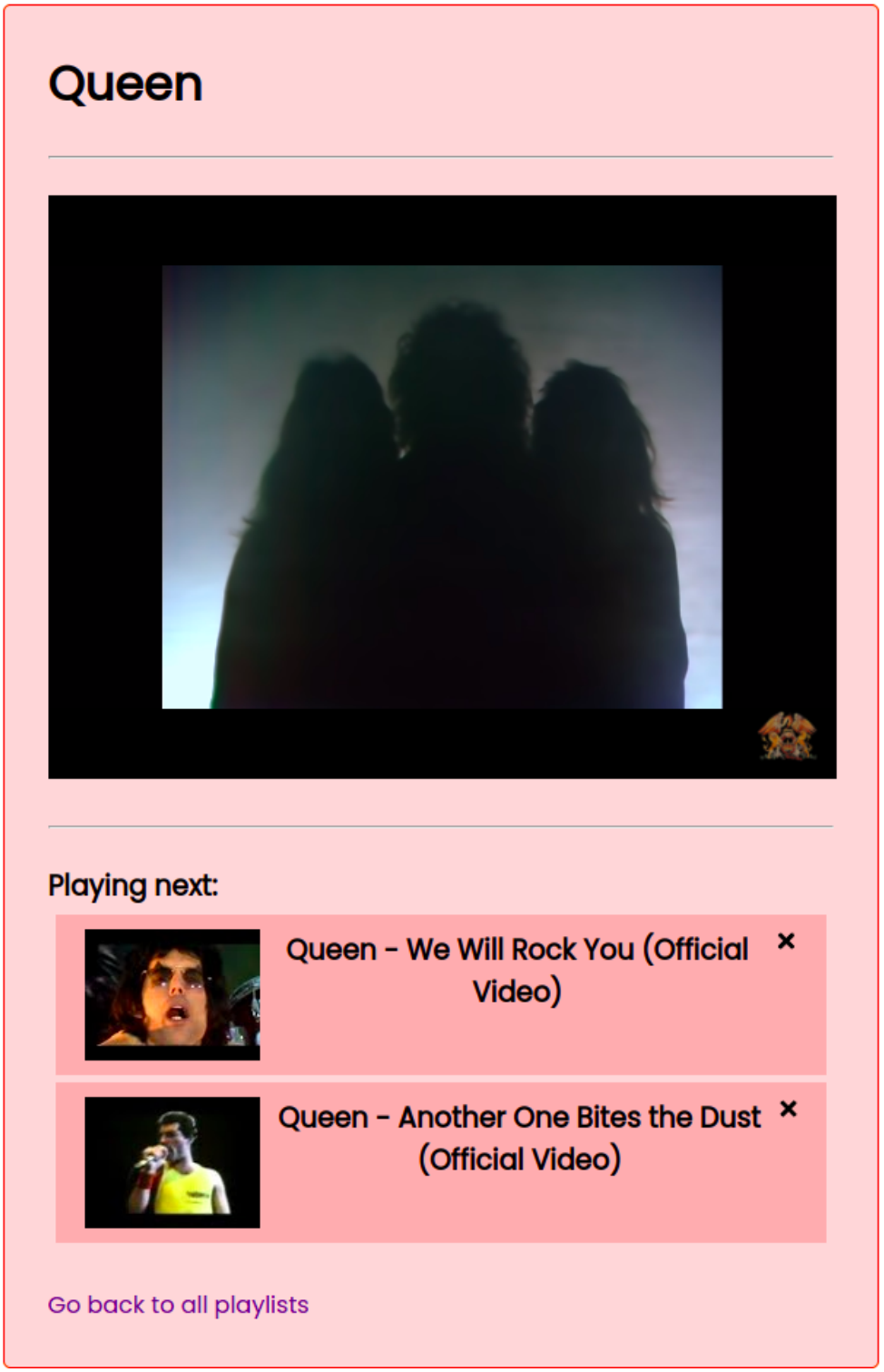
```
router.post('/search', function(req, res, next) {
  var video
  getVideo(req.body.Title).then(data => {
    video = new Video({
      title: data.items[0].snippet.title,
      id: data.items[0].id.videoId,
      description: data.items[0].snippet.description,
      thumbnail: data.items[0].snippet.thumbnails.default.url,
      publishedAt: data.items[0].snippet.publishTime
    })
    Playlist.updateOne({_id: req.body.Playlist}, {$addToSet: {videos: video}})
      .then(() => res.send(video))

  })
});

async function getVideo(title) {
  const response = await fetch('https://www.googleapis.com/youtube/v3/search?part=snippet&q='
  + encodeURI(title) + '&key=AIzaSyC0ku6PZlOYfyf1F6Xyad5x_D3TwNrywmg')
  return response.json()
}
```

Prikazani isječak koda prikazuje backend poziv za dodavanje novog videa. Prvo se poziva funkcija getVideo koja dohvaća JSON odgovor od Youtube API-ja, a zatim se taj video pretvara u Video objekt te se isti dodaje u odgovarajuću Playlistu te se ista ažurira u bazi podataka.

Kada korisnik postavi željene videe u playlistu, na bilo koji video može kliknuti gumb za pokretanje i web stranica se preusmjerava na stranicu koja služi kao video preglednik.



Slika 5. Prikaz video playera

Odabrani video se odmah po otvaranju pušta na video playeru koji je uvezen sa react-youtube paketa. Ostali videi su dodani u red čekanja, i pustiti će se po završetku trenutnog videa ili klikom na video iz reda.

Video se iz reda čekanja može obrisati klikom na “x” gumb, ali on tom radnjom neće biti izbrisan iz baze podataka, nego samo iz reda čekanja, kako bi korisnik ponovnim pokretanjem playliste opet mogao dobiti taj video. Sve navedene funkcionalnosti prikazane su u kodu ispod:

```
const onEnd = async (event) => {
    if (vids[0] !== undefined) {
        setVidId(vids[0].id)
        setVids(vids.filter(video => video.id !== vids[0].id))
    }
}

const onPlay = (videoId) => {
    setVidId(videoId)
    setVids(vids.filter(video => video.id !== videoId))
}

const onDelete = (videoId) => {
    setVids(vids.filter(video => video.id !== videoId))
}

return (
    <div>
        <YouTube videoId={vidId} opts={opts} onEnd={onEnd}/>
        <br />
        <hr />
        <br />
        {vids[0] !== undefined ? (<h3>Playing next:</h3>) : (<h3>The queue is empty!</h3>)}
        <>
        {vids.map((video) => (
            <VideoInPlaylist key={video.id} video={video} onPlay={onPlay} onDelete={onDelete}/>
        ))}
    </>
    </div>
)
```

Svaki korisnik može imati više playlista, sa više videa, ali samo će on moći vidjeti navedene playliste i videe. Ukoliko se na stranicu prijavi neki drugi korisnik on će vidjeti samo svoje playliste.

5. Korišteni alati

Kao razvojno okruženje za izradu web-stranice korišten je Visual Studio Code. Backend je izrađen pomoću Node.js-a i povezan je na NoSQL MongoDB bazu podataka. Zbog jednostavnosti podataka nisam odabrao relacijsku bazu podataka, već ne-relacijsku i svi se podaci nalaze u cloudu od MongoDB-a. Također, backend se služi Youtube API-jem od kojih dobiva rezultate pretraživanja za pojedini upit te se ti rezultati pohranjuju u bazu podataka. Frontend je rađen u React Frameworku. Od vanjskih komponenti koje se koriste na sučelju korišteni su react-youtube i react-facebook-login.

6. Zaključak

Cilj ovog seminara bio je dublje istražiti React Framework te napraviti funkcionalnu web-stranicu za izradu playlista i puštanje videa. Za frontend web-stranice korišten je već spomenuti React Framework, a za backend je korišten Node.js te NoSQL.

Prilikom izrade web-stranice prvo je isprogramiran backend koji se spaja na MongoDB cloud te kreira schema za playlistu. Tu je bilo potrebno i pozivati Youtube API te dohvaćati njegove rezultate kako bi se spremili u playlistu. Zatim je napravljen frontend dio koji komunicira sa backendom te dohvaća playliste ili određene videe za pojedine korisnike.

Web-stranica je funkcionalna i spremna za korištenje, ali može se još nadograditi. U budućnosti bi se mogle dodati još neke funkcionalnosti koje bi korisnicima bile od koristi. Primjerice lista prijatelja s kojima bi mogli dijeliti svoje playliste. Osim toga mogla bi se implementirati postavljanje playlista kao javnih, iako su zasad privatne. Kada bi se implementirala mogućnost javnih playlista, bilo bi potrebno napraviti neku listu ili pretraživač javnih playlista ili playlista prijatelja kako bi korisnik mogao iste vidjeti. Također, bilo bi korisno imati opciju pretraživanja playlista po određenom videu, točnije da korisnik može vidjeti u kojim se sve javnim playlistama ili playlistama prijatelja nalazi određeni video.

7. Literatura

1. P. J. Sadalage, M. Fowler. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. 2012. URL: https://bigdata-ir.com/wp-content/uploads/2017/04/NoSQL-Distilled.pdf

2. Youtube API službena dokumentacija. URL: https://developers.google.com/youtube/v3

3. React Framework službena dokumentacija. URL: https://reactjs.org/

Youtube Video Player

Sažetak

U sklopu seminarskog rada napravljena je web-stranica u kojoj korisnici izrađuju playliste, dodaju videe sa youtube-a i gledaju iste. Za pristup navedenim mogućnostima web-stranice, potreba je prijava korisnika putem Facebook računa. U MongoDB bazu podataka spremaju se playliste koje je korisnik kreirao, a dio podataka koji se pohranjuje dohvaća se pomoću Youtube API-ja. Na sučelju se korisniku kada prvi put otvori web-stranicu prikazuje komponenta za prijavu na Facebook račun, a zatim može kreirati playliste, pretraživati i dodavati videe u playliste te zatim pregledavati videe iz istih. Backend je rađen u Node.js-u, a frontend pomoću React Frameworka.

Ključne riječi: youtube, playlista, video, player, MongoDB, Node.js, React