VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

RAZVOJ SPLETNE STRANI Z OGRODJEM REACT

Kandidat: Blaž Petek

Vrsta študija: študent izrednega študija

Študijski program: Medijska produkcija – splet

Mentor – predavatelj: Matej Butala

Mentor v podjetju: Miha Jerot

Lektorica: Jasmina Vajda Vrhunec, prof. slov.

Maribor, 2021

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Blaž Petek sem avtor diplomskega dela z naslovom »Razvoj spletne strani z ogrodjem React«, ki sem ga napisal pod mentorstvom Mateja Butale.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli kot moje lastne – kaznivo po Zakonu o avtorski in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Pobrežje, junij 2021 Podpis študenta:

ZAHVALA

Zahvaljujem se mentorju in predavatelju Mateju Butali za pomoč in podporo pri pisanju diplomskega dela.

Zahvaljujem se tudi mentorju v podjetju Mihi Jerotu za njegovo pomoč pri razvoju spletne strani za to diplomsko delo in za njegovo konstantno podporo.

Prav tako se zahvaljujem vsem profesorjem in drugim delavcem na šoli, ki so mi pomagali pri izobrazbi in mi bili vedno na voljo za posvete in vprašanja.

Posebna zahvala velja družini, prijateljem in vsem, ki so mi kakorkoli pomagali in stali ob strani v času študija in pisanja diplomskega dela.

POVZETEK

V diplomskem delu bomo raziskali osnove razvoja prednjega oziroma začelnega (angl. frontend) in zalednega (angl. backend) dela spletnih strani. Osnova vsake spletne strani so tehnologije HTML, CSS in JavaScript, zato bomo opisali njihovo delovanje in jih podrobneje predstavili. Izvedli bomo tudi primerjavo med tremi trenutno najbolj priljubljenimi ogrodji JavaScript: Angular, Vue in ReactJS.

React.js je knjižnica JavaScript, ki se uporablja za upravljanje vidnega sloja spletnih aplikacij. Omogoča prilagodljivost glede orodij in arhitekture. Zaradi razvite skupnosti uporabnikov in podpore večjih podjetij je React zanesljiv. Razvijalci začelnega dela se uporabo ogrodja React naučijo hitreje in ga aktivno implementirajo.

Koda ogrodja React je napisana v jezikih JavaScript in JSX. Koda napisana deklarativno. To pomeni, da opišete rezultat in ne načina, kako ga doseči. Osnovni gradnik v ogrodju React je komponenta.

To razvijalcem omogoča ponovno uporabo kode, učinkovitejše testiranje, nadgrajevanje in vzdrževanje aplikacije. Deklarativna koda in komponente sta glavni značilnosti ogrodja Reacta.

Ker bomo ogrodje React uporabili tudi v diplomskem delu, ga bomo podrobneje opisali. Predstavili bomo vse korake razvoja spletnih strani, torej kako preiti od ideje do dizajna in na koncu do želenega izdelka. Opisali bomo, katere elemente je potrebno uporabiti za oblikovanje strani, primerjali najbolj priljubljena ogrodja za razvoj in zapisali kodo.

Danes se za ReactJS odloča večina spletnih razvijalcev. Razlog za to je, da ponuja zelo bogato knjižnico JavaScript. Knjižnica JavaScript omogoča razvijalcem spleta večjo prilagodljivost, da lahko izberejo želeni način.

Končni cilj diplomskega dela je podrobno spoznati ogrodje ReactJS in postaviti delujočo spletno stran, ki uporabnikom predlaga vsebine za ogled glede na filtre, ki so jih izbrali. Sama spletna stran bo na voljo za ogled na spletnem naslovu: https://imbored.tv/.

Ključne besede: razvoj spletne strani, React, prednji del, zaledni sistem, dizajn.

ABSTRACT

DEVELOPING A WEBSITE WITH REACT

As a part of this thesis, we will get to know the basics of frontend and backend web development. Basics of every website are HTML, CSS and Javascript, so we will look at how these technologies work in more detail. We'll also look at differences between three of the currently most popular Javascript frameworks; Angular, Vue and ReactJs. The latter is also the one that I've decided to use for this project, and so, it is also the one that is described in most detail.

ReactJs is a JavaScript package that is used to manage the view layer of web applications. It provides tools and architecture freedom for the user.

React is dependable because to its developed community and the support of larger companies. It is enables frontend developers to learn and use React faster than any other framework.

JavaScript and JSX are used to write React code. What is critical, though, is that the code is written declaratively. It means you define the end result rather than the commands on how to get there.

A component is a fundamental building part in React. It is what lets developers to reuse code and makes testing, scaling, and maintaining an application easier. Declarative code and components are the main characteristics of the React framework.

We will learn about all the steps of web development, from an idea, to design, and to a final product. We will learn about tools we can use for making website designs, compare most popular development frameworks, and write some code.

Today, ReactJS is chosen by most of the web developers. It is because it is offering a very rich JavaScript library. The JavaScript library provides more flexibility to the web developers to choose the way they want.

The ultimate goal of this thesis, is to make a fully functioning website, that suggests movies and TV shows to users, based on filters that they've chosen. The website itself is available at: https://imbored.tv/.

Keywords: Website development, React, frtontend, backend, design

KAZALO VSEBINE

1. UVOD 6

1.1 Opis področja in opredelitev problema 6

1.2 Namen, cilji in osnovne trditve 7

1.3 Predpostavke in omejitve 7

1.4 Uporabljene raziskovalne metode 7

2. RAZVOJ SPLETNIH STRANI 8

2.1 HTML 8

2.1.1 HEAD-značka *9*

2.1.2 BODY–značka *9*

2.1.3 CLASS in ID *10*

2.1.4 CSS 10

2.3 JavaScript in ogrodja 12

2.3.1 ReactJS *12*

2.3.2 Angular *14*

2.3.3 VueJS *14*

2.3.4 Python 15

2.3.5 Flask *15*

3. PRODUKCIJA 17

3.1 Dizajn 17

3.2 Zaledni sistem 18

3.2.1 Baza podatkov *19*

3.2.2 Razvoj *20*

3.3 Prednji oziroma začelni del 23

3.3.1 Komponente *24*

4. SKLEP 27

5. LITERATURA 29

KAZALO SLIK

Slika 1: Primer osnovne strukture dokumenta HTML 10

Slika 2: Primer uporabe glavnih HTML-značk 11

Slika 3: Primer CSS-izbiranja elementa 12

Slika 4: Primer CSS-izbiranja elementa prek ID-ja 12

Slika 5: Primer CSS-izbiranja elementa preko CLASS-a 12

Slika 6: Primer dodajanja barve in ozadja naslovu 12

Slika 7: Primer komponent v navigaciji spletne strani academia.si 14

Slika 8: Primer kode JSX 14

Slika 9: Primer definicije url-jev v ogrodju Flask 16

Slika 10: Logotip na podlagi osnovne barve spletne strani 18

Slika 11: Dizajn spletne strani v Figmi 19

Slika 12: Uvoz knjižnic v ogrodje Flask 21

Slika 13: Inicializacija ogrodja Flask 22

Slika 14: Funkcija za posodobitev vsebin v naši bazi 23

Slika 15: Funkcija za pridobivanje vsebin 24

Slika 16: Vizualni prikaz komponent na strani 25

Slika 17: Primer shrambe MobX s filtri 26

Slika 18: Klicanje zalednega sistema s podatki iz filtra 27

Slika 19: Funkcija za pridobivanje napovednikov vsebin 27

1. UVOD

Na leto izide več kot 1000 filmov in več kot 500 novih televizijskih serij, zato je težko najti vsebino, ki bi posameznika zanimala, brez neke časovne investicije. Tako sem prišel do ideje za to diplomsko delo. Cilj mojega dela je spletna stran, ki vam na podlagi vaših filtrov in nastavitev predlaga naključno vsebino za ogled. Ta vsebina je lahko film ali televizijska serija, ki vam je predlagana naključno, ob kliku na gumb. Vedno, ko uporabnik pritisne na ta gumb, se mu prikaže nova in naključna vsebina na podlagi nastavitev, ki jih je izbral. Nekatere izmed teh nastavitev so letnica izida vsebine (primer: od 2010 do 2019), povprečna ocena vsebine, minimalno število ocen vsebine itd. Tako uporabniku odstranimo breme raziskovanja in iskanja vsebin, ki bi ga zanimale, in to nadomestimo z enostavnim klikom na gumb.

Za to spletno stran bomo uporabili eno izmed modernejših ogrodij za gradnjo prednjega oziroma začelnega (angl. frontend) dela spletnih strani. To ogrodje se imenuje ReactJS in deluje na podlagi JavaScripta. Doda nam veliko dodatnih funkcionalnosti in izboljšav pri gradnji spletnih strani, vse od ponovne uporabljivosti kode ali tako imenovanih komponent (angl. Components) do vodenja stanja aplikacije (angl. State management). ReactJS ima tudi veliko skupnost uporabnikov, ki hitro priskoči na pomoč, če programer česa ne razume oziroma ne ve, kako naprej. Ta skupnost prav tako skrbi za veliko knjižnic, ki so v veliko pomoč pri gradnji spletnih strani v ogrodju ReactJS. Seveda pa to ni edino tako ogrodje. Imamo še nekaj drugih, ki so podobna, vendar ne povsem enaka. Vsako ima namreč svoje prednosti in slabosti. Tudi to bomo raziskali in predstavili razlike med njimi.

1.1 Opis področja in opredelitev problema

Pri oblikovanju kakovostnih spletnih strani nista dovolj samo znanje grafičnega oblikovanja ter programiranja, ampak je potrebno tudi poznavanje procesov dokumentiranja dela ter prednosti in slabosti izbranih orodij. Vsak takšen proces je treba začeti s projektiranjem arhitekture spletnega mesta. Najprej smo razmisliti o ciljih spletne strani. Odločili smo se, da ustvarimo informacijsko platformo, ki bo pospešila proces interakcije med uporabnikom in spletnimi mesti za recenziranje filmov ter serij.

1.2 Namen, cilji in osnovne trditve

Namen diplomskega dela je izdelati spletno stran z ogrodjem React. ReactJS je sestavljen iz več komponent, vsaka komponenta ima lastno logiko in kontrolnike. Te komponente so odgovorne za izpis majhnega, večkrat uporabnega dela kode HTML, ki ga lahko ponovno uporabimo. Ponovno uporabna koda pripomore k lažjemu razvoju in vzdrževanju aplikacij. Te komponente je mogoče ugnezditi z drugimi komponentami, kar omogoča, da se iz preprostih gradnikov zgradijo zapletene aplikacije. ReactJS uporablja virtualni mehanizem, ki temelji na spletnem standardu DOM, za izpolnjevanje podatkov v HTML DOM. Virtualni DOM deluje hitro, ker spreminja le posamezne elemente DOM, namesto da bi vsakič znova naložil celoten DOM. (React, 2020)

Pri izdelavi spletne strani z ogrodjem React smo si zastavili naslednje hipoteze – trditve:

- H1: Ogrodje ReactJS ima razvito napredno skupnost uporabnikov ter odprte repozitorije, ki so v pomoč pri gradnji spletnih strani.
- H2: Značilnosti ogrodja ReactJS omogočajo hitrejšo implementacijo naprednih funkcij v spletno stran, v primerjavi z drugimi ogrodji.
- H3: Ogrodje ReactJS omogoča zanesljivo posodabljanje vsebin iz drugih spletnih mest.

1.3 Predpostavke in omejitve

Hitrost razvoja ima tako prednosti kot slabosti. Ker se okolje nenehno tako hitro spreminja, nekateri razvijalci ne zmorejo, da bi se redno učili novih načinov dela.

To je omejitev, ki je značilna za vsa orodja, ki se nenehno posodabljajo. Tehnologije React se posodabljajo in pospešujejo tako, da časa za pripravo ustrezne dokumentacije ni na voljo. Da bi to premagali, razvijalci sami ustvarjajo navodila ob izdaji aktualnih projektov. (Angular vs. React vs. Vue: A performance comparison:, 2020)

1.4 Uporabljene raziskovalne metode

Pri pisanju diplomskega dela ter za ustvarjanje spletne strani bomo uporabil različne raziskovalne metode. S komparativno metodo bomo preučili programske jezike, ogrodja ter ostala orodja, ki so na voljo. Z opisno metodo bomo s pomočjo domače in tuje literature zagotovili tudi sekundarno raziskavo obravnavanih procesov ter orodij.

2. RAZVOJ SPLETNIH STRANI

Razvoj spletnih strani danes delimo na dva glavna dela: začelni (angl. frontend) in zaledni del (angl. backend). Prednji ali začelni del je del spletne strani, ki ga uporabnik lahko vidi in s katerim lahko interaktira. V ta del spadajo tri glavne tehnologije: HTML ali »Hypertext markup language«, CSS ali »Cascading Style Sheets« in JavaScript. Seveda obstajajo tudi drugi programski jeziki in ogrodja, ki se uporabljajo pri razvoju spletnih strani, vendar so ta tri ogrodja osnova skoraj vsake spletne strani. Izmed teh treh je HTML edina tehnologija, ki jo moramo uporabiti pri gradnji spletne strani, CSS in JavaScript pa sta zelo zaželena, vendar ne nujna. (CSS W3Schools, 2020)

2.1 HTML

HTML je kratica, ki pomeni »Hypertext markup language« (slov. jezik za označevanje nadbesedila) in je osnovna komponenta gradnje spletnih strani. Vsaka spletna stran, do katere lahko dostopamo, uporablja HTML. To je dokaj enostaven programski jezik, ki se ga lahko naučimo relativno hitro, hkrati pa je tudi zelo močan glede na njegove zmogljivosti.

HTML se v osnovi uporablja za dodajanje vsebine na spletno stran. Dodamo lahko različne vrste besedil, vse od naslovov do navadnega besedila, slik, povezav do drugih strani in še veliko več. To naredimo prek tako imenovanih značk. Vso vsebino, ki jo hočemo dodati na našo spletno stran, bomo dodali prek značk. (CSS W3Schools, 2020)

Osnovna struktura dokumenta HTML je tako zgrajena iz naslednjih značk:

- HTML
- HEAD (slov. glava)
- BODY (slov. telo)

```html
<!DOCTYPE html>

<head>
    <title>Naslov spletne strani</title>
</head>
<body>
    <p>Tukaj lahko dodajamo glavno vsebino naše spletne strani</p>
</body>
</html>
```

Slika 1: Primer osnovne strukture dokumenta HTML

2.1.1 HEAD-značka

HEAD-značka se uporablja za metapodatke (angl. metadata). Metapodatki so informacije o trenutni strani, ki niso vidne uporabnikom. To so na primer naslov strani, stil strani (CSS) in podobno. V sami glavi spletne strani lahko tudi uvozimo različne knjižnice, pisave in ostale metapodatke za spletno stran. (CSS W3Schools, 2020)

2.1.2 BODY–značka

V BODY-značko spada vsebina naše spletne strani, torej to, kar bo uporabnik videl, ko obišče našo spletno stran. To so lahko naslovi, besedila, povezave do spletnih strani, slike, videi in podobno. (CSS W3Schools, 2020)

Primer značk, ki jih lahko damo v BODY-značko:

- **p ali paragraph (slov. paragraf)** – ta značka se uporablja za vsebinsko besedilo spletne strani. Torej za besedilo, ki ni naslov;
- **h1 ali header1 (slov. naslov 1)** – te značke se uporabljajo za naslove. h1 je glavni naslov, ki bo prikazan z največjimi črkami. Imamo pa potem še h2, h3, h4, h5 in h6;
- **img ali image (slov. slika)** – uporablja se za dodajanje slik na spletno stran;
- **a** – uporablja se za dodajanje povezav do drugih strani na našo stran;
- **table (slov. tabela)** – uporablja se za dodajanje tabel na našo spletno stran.

```html
<!DOCTYPE html>

<head>
    <title>Naslov spletne strani</title>
</head>
<body>
    <h1>To je naslov</h1>
    <h2>To je podnaslov</h2>
    <p>To je navadno besedilo</p>
    <img src="http://prntscr.com/p4u99a" alt="Academia logotip">
    <a href="https://www.academia.si/">Obišči Academio</a>
</body>
</html>
```

Slika 2: Primer uporabe glavnih HTML-značk

2.1.3 CLASS in ID

Značke v dokumentu HTML lahko tudi označimo, da lahko kasneje do njih dostopamo in jih oblikujemo prek CSS-ja. Uporabljamo dve vrsti označevanja značk: ID in CLASS. Značko označimo z ID, če bo to edina značka, ki jo bomo oblikovali z enim stilom. Če torej hočemo oblikovati logotip spletne strani, se bo ta stil uporabil samo za logotip, če hočemo, da ima več različnih značk en stil, pa dodamo vsem tem značkam označbo ali atribut CLASS in ga nastavimo na enako vrednost. Tako lahko oblikujemo na primer naslove člankov na spletni strani, ki so načeloma vsi oblikovani enako. S tem preprečimo ponavljanje kode. (CSS W3Schools, 2020)

2.1.4 CSS

CSS ali »Cascading Style Sheets« (slov. kaskadne stilske podloge) se uporablja za oblikovanje spletnih strani. S HTML-jem dodamo spletni strani vsebino, s CSS-jem pa to vsebino oblikujemo. Vsako značko ali vsak element v dokumentu HTML lahko posebej oblikujemo. Tudi če imamo več kot eno značko h1, lahko vsako oblikujemo drugače. Značko, ki jo želimo oblikovati, najprej izberemo. To lahko storimo na tri načine:

1. Če hočemo oblikovati vse značke p, jih bomo izbrali tako, da napišemo:

```
p {

}
```

Slika 3: Primer CSS-izbiranja elementa

2. Če hočemo izbrati element z določenim atributom ID, bomo pred ta ID dodali znak za lojtro:

```
#imeIdja {

}
```

Slika 4: Primer CSS-izbiranja elementa prek atributa ID

3. Če hočemo izbrati vse elemente, ki imajo nek določen CLASS, pa bomo pred ta CLASS dodali piko:

```
.imeClassa {

}
```

Slika 5: Primer CSS-izbiranja elementa prek atributa CLASS

Ko smo element izbrali, ga lahko začnemo oblikovati. Vsak element ima vrednosti, ki jih lahko nastavljamo oziroma spreminjamo. Te vrednosti so lahko: barva ozadja, barva pisave, vrste pisave, robovi, pozicije elementov, prostor okoli elementov in še veliko več. Če bi torej hoteli spremeniti element h1, ki ima ID *»naslovEna«*, tako, da je rdeče barve in ima ozadje rumene barve, bi to storili tako:

Slika 6: Primer dodajanja barve in ozadja naslovu

2.3 JavaScript in ogrodja

JavaScript je programski jezik, ki se uporablja predvsem za izdelavo dinamičnih spletnih strani. Na začetku je ta programski jezik imel ime »LiveScript«, kasneje pa je bil preimenovan v JavaScript, predvsem zato, da se že po imenu vidi povezava med Javo in JavaScriptom. JavaScript je namreč programski jezik, ki je bil napisan v programskem jeziku Java. (Aston, 2015)

Danes je JavaScript najbolj uporabljan programski jezik, saj ga podpira vsak brskalnik, in to brez dodatnega dela s strani programerja ali uporabnika. JavaScript se lahko uporablja za več stvari:

- lahko manipulira kodo HTML spletne strani (ko na primer uporabnik klikne na nek gumb, se skrije en del spletne strani in se prikaže drugi del);
- lahko kontaktira druge spletne strani, pridobi neke podatke in jih prikaže na naši strani;
- uporablja se na strežnikih spletnih strani ali na tako imenovanem zalednem sistemu spletnih strani za obdelovanje podatkov.

2.3.1 ReactJS

ReactJS, drugače znan tudi kot React.js ali enostavno React, je odprtokodno ogrodje (angl. framework) JavaScript za gradnjo prednjega oziroma začelnega dela spletnih strani. Spletne strani, ki so narejene z ogrodjem React, so tako imenovane »enostranske spletne strani« (angl. Single page application – SPA), to pa zato, ker se celotna stran prikazuje na eni strani. Tudi če gre uporabnik iz domače strani (npr. mojastran.si) pod nek drug naslov, na primer *mojastran.si/novnaslov*, bo React zamenjal vsebino na domači strani z vsebino, ki je ustrezna za naslov *mojastran.si/novnaslov*. Torej se vse strani v bistvu prikazujejo na isti strani. To

omogoča Reactu, da je hiter, enostaven za uporabo in se lahko uporablja za majhne in prav tako za velike spletne strani. Posebnost Reacta je tudi, da uporablja virtualni model objekta dokumenta ali DOM (angl. Document Object Model), saj je bil HTML DOM narejen za statične strani. Razlika med virtualnim DOM-om in HTML DOM-om je v tem, da HTML DOM na novo prikaže vsebino celotne strani, ko se karkoli spremeni, medtem ko virtualni DOM deluje tako, da spremeni samo tisti del spletne strani, ki se je dejansko posodobil. Ker se vsebina in React koda spletne strani ves čas spreminjata, virtualni DOM zmanjša delo brskalnika in računalnika uporabnikov za kar velik odstotek. (Aston, 2015)

Velika prednost Reacta so tudi komponente, saj omogočajo ponovno uporabo določenih delov strani, brez podvajanja kode. Če imamo na naši spletni strani navigacijo, bi ta navigacija bila komponenta za sebe, potem pa bi v tej komponenti imeli še druge komponente, kot so logotip, navigacijski predmet, gumb za prijavo/registracijo ipd. (React, 2020)

Slika 7: Primer komponent v navigaciji spletne strani academia.si

Potem pa je tukaj še JSX (Javascript XML), ki je programski jezik za templatiranje React aplikacij. JSX združi HTML in React v eno in nam omogoča enostavno oblikovanje spletnih strani ter vstavljanje vsebine v prikaz podatkov na spletni strani. JSX se v glavnem uporablja v komponentah. (Aston, 2015)

```
<h1>{article.title}</h1>
<h2>{article.subtitle}</h2>
<img src={article.imgUrl} alt={article.imgAlt}>
<p>{article.body}</p>
```

Slika 8: Primer kode JSX

Na koncu sem se za ta projekt odločil uporabiti ReactJS, ker je izmed vseh ogrodij, ki so trenutno na voljo, najbolj razširjen v Sloveniji. Omogoča ti svobodo, da si strukturo projekta

sestaviš tako, kot si sam želiš, in, ker je React bolj knjižnica kot ogrodje, je tudi hitrejši in bolj učinkovit kot ostala ogrodja.

2.3.2 Angular

Angular se, tako kot ogrodje React, uporablja za izdelavo prednjega oziroma začelnega dela spletnih strani. Ogrodje je razvilo podjetje Google leta 2010 in je bilo prvotno poznano po imenu AngularJS, medtem ko se novejše različice imenujejo samo Angular, zraven pa je dodana še številka verzije (trenutna verzija je 4). Ogrodje je narejeno na programskem jeziku JavaScript, uporablja pa tudi TypeScript, ki je nadgradnja JavaScripta in nam omogoča veliko dodatnih funkcionalnosti, ki jih JavaScript v osnovi ne omogoča. Ena večjih prednosti so statični tipi, ki omogočajo, da definiramo vrsto spremenljivk, ki jih mora neka funkcija prejeti ali ki jih neka funkcija vrne. Tako lahko predhodno ujamemo veliko napak, za katere bi drugače porabili več časa. (Angular vs. React vs. Vue: A performance comparison:, 2020)

Angular je sestavljen iz več delov. V osnovi imamo komponente, katerih naloga je prikaz HTML-vsebine, potem pa imamo še tako imenovane storitve (angl. Services), v katerih se shranjujejo API-klici. Te komponente nato kličejo storitve, iz katerih potrebujejo podatke, in ko storitev podatek vrne, ga komponenta prikaže. (Angular vs. React vs. Vue: A performance comparison:, 2020)

2.3.3 VueJS

VueJS je ogrodje za razvoj prednjega oziroma začelnega dela spletnih strani, ki ga je leta 2014 izdal Evan You. Zasnovano je na programskem jeziku JavaScript in je trenutno eno najbolj priljubljenih ogrodij za gradnjo prednjega dela spletnih strani.

VueJS uporablja tako imenovani virtualni DOM (angl. Document object model), kar pa pomeni, da kakršnekoli spremembe, ki jih naredimo, niso takoj vidne v DOM-u, ampak samo v virtualnem DOM-u. Šele ko je potrebno, se DOM posodobi. To se uporablja predvsem za optimizacijo spletnih strani in za hitrejše prikaze sprememb uporabnikom. (Aston, 2015)

VueJS je eno izmed lažjih ogrodij za uporabo, vendar pa to ne pomeni, da je manj zmožno kot ReactJS ali Angular. Prav nasprotno. V veliko primerih naredimo več, hitreje in z manj kode

kot pri katerem drugem ogrodju. (Angular vs. React vs. Vue: A performance comparison:, 2020)

2.3.4 Python

Python je programski jezik, ki ga je proti koncu osemdesetih let izdal nizozemski programer Guido van Rossum. Guide je pred tem delal na programskem jeziku ABC in se je nato odločil, da bo na podlagi tega naredil svoj programski jezik, v katerem bo lahko odpravil probleme, ki jih je videl v programskem jeziku ABC. Tako smo dobili prvo verzijo Pythona.

Najbolj razširjeni verziji Pythona sta 2.* in 3.*, ki se uporabljata predvsem za razvoj zalednega sistema spletnih strani. Vendar pa se Python ne uporablja samo za spletne strani. Njegovo uporabo namreč srečujemo skoraj povsod, vse od razvoja programov za operacijske sisteme, avtomatizacije raznih nalog, obdelave podatkov do umetne inteligence (angl. Artificial Inteligence) in še veliko več. (Chinnathambi, 2018)

Python sem se odločil uporabiti, ker imam že nekaj predhodnega znanja v tem jeziku. Všeč sta mi predvsem njegova sintaksa in velika skupnost uporabnikov.

2.3.5 Flask

Flask je ogrodje, narejeno na podlagi Pythona, ki se uporablja za razvoj zalednega sistema spletnih strani. Torej prednji oziroma začelni del pošlje tako imenovani API ali *application programming interface* (slov. vmesnik za namensko programiranje) do zalednega sistema, ki pridobi neke podatke, bodisi iz baze podatkov ali katerega drugega API-ja, ali pa samo nekaj izračuna in te podatke vrne prednjemu oziroma začelnemu delu, ki te podatke prikaže uporabniku. Torej v Flasku lahko definiramo različne povezave, ki bodo na voljo, in kaj naj se zgodi, ko prednji oziroma začelni del obišče katero izmed teh povezav. (Chinnathambi, 2018)

```python
@app.route('/filmi')
def getMedia():
    return "Tukaj lahko vrnemo vse nase filme"


@app.route('/serije')
def updateDb():
    return "Tukaj lahko vrnemo vse nase serije"
```

Slika 9: Primer definicije povezav v ogrodju Flask

V primeru iz zgornje slike bi, če bi obiskali **www.mojaspletnastran.si/filmi**, na strani videli napis »*Tukaj lahko vrnemo vse nase filme*«. Seveda je to enostaven primer, kjer vrnemo samo besedilo. Na pravi strani bi tukaj iz baze podatkov ali katerega drugega API-ja pridobili filme, jih dali v pravilen format in jih vrnili namesto besedila, ki ga trenutno vračamo. (CSS W3Schools, 2020)

To je osnovni princip razvoja zalednih sistemov spletnih strani, ki se drugače imenuje REST ali *representational state transfer*.

Ker za mojo spletno stran potrebujem le osnovne funkcionalnosti v zalednem sistemu, je Flask kot nalašč za ta projekt. Ima vse, kar potrebujemo, torej: »routanje« ali definiranje različnih poti na spletni strani, povezovanje teh poti z določenimi funkcijami in osnovno avtentikacijo za zaščito teh poti. Flask prav tako ne potrebuje veliko pomnilnika in procesne moči na strežniku, zaradi česar lahko kasneje kupimo najcenejši server z 1 GB rama. (Aston, 2015)

3. PRODUKCIJA

Kot sem že prej omenil, se razvoj spletnih strani deli na dva dela: prednji oziroma začelni in zaledni del. Tako sem tudi začel z izdelavo moje spletne strani. Najprej dizajn, nato pa razvoj.

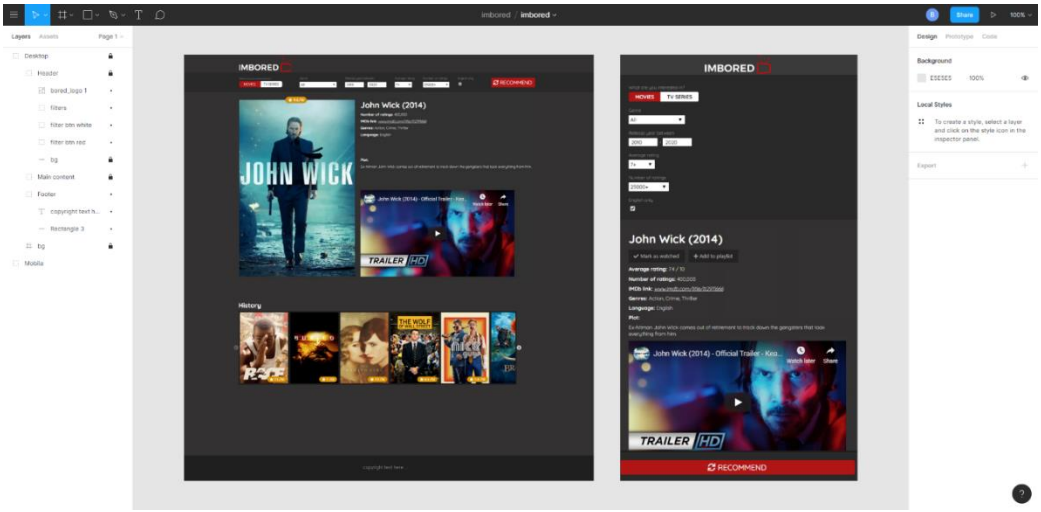
3.1 Dizajn

Ker nimam veliko izkušenj z dizajnom spletnih strani, sem najprej šel na spletno stran webdesign-inspiration.com in si ogledal nekaj spletnih strani. Na tej strani sicer nisem našel dizajna, ki bi se ujemal s spletno stranjo, ki sem si jo zamislil, sem pa dobil idejo za osnovno barvno paleto. Odločil sem se za temno podlago z rdečimi odtenki. Ideja za tem je bila, da veliko ljudi gleda filme in serije predvsem popoldan in zvečer, kar pa pomeni, da je pogosto ob tem času že tema, zato očem bolj ustrezajo spletne strani s temnejšo podlago. Zraven tega pa mi je tudi sam izgled te barvne palete všeč, saj se rdeči odtenki lepo skladajo s temno sivo podlago.

Za izdelavo dizajna sem se odločil, da bom uporabil program Figma, saj je eden izmed programov, ki smo se ga dokaj podrobno naučili na predavanjih, je zastonj in ima vse funkcionalnosti, ki sem jih potreboval. Najprej sem začel oblikovati stran za računalnike. Naredil sem si osnovni okvir strani in nastavil ozadje na temno sivo barvo, ki sem si jo izbral. Nato sem začel oblikovati glavo spletne strani, torej logotip in filtre za ogled serij. Za logotip sem se odločil uporabiti naslov spletne strani in zraven dodal ikono televizije. Besedilo logotipa je bele barve, ikona zraven logotipa pa rdeče, kar se bo zopet lepo ujema z barvno paleto spletne strani.

Slika 10: Logotip na podlagi osnovne barve spletne strani

Nato sem oblikoval glavni del spletne strani, kjer bo prikazan predlagani film ali serija. Tukaj sem se odločil uporabiti veliko sliko in oceno IMDB na levi strani ter informacije vsebine na desni strani. Na desni strani je prikazan tudi napovednik (angl. Trailer) vsebine. Pod prikazom vsebine je vidna zgodovina najdenih vsebin, ki vključuje sliko vsebine in oceno. Uporabnik ima nato možnost klikniti na okvirček, ki vsebuje vsebino in se vsi podatki zapolnijo v glavnem delu spletne strani. Na koncu sem še dodal enostavno nogo (angl. Footer), kjer sem vstavil avtorske pravice (angl. Copyright).



Slika 11: Dizajn spletne strani v ogrodju Figma

3.2 Zaledni sistem

Razvoj spletne strani sem začel z zalednim sistemom, saj bom kasneje veliko lažje začel z razvojem prednjega oziroma začelnega dela strani.

Prvi korak je bil najti način pridobivanja filmov in serij ter vseh njihovih podatkov, predvsem povprečja ocen in skupno število ocen, saj IMDB nima javno dostopnega API-ja. Prva rešitev, ki sem jo našel, je bila, da namesto ocen IMDB-ja uporabim spletno stran TMDB (angl. The movie database), ki ima javno dostopen API, prek katerega lahko dobimo vse podatke o vsebinah, vključno z ocenami. Vendar pa so to ocene, ki jih vrnejo iz njihove spletne strani, in ne ocene IMDB-ja. To mi ni bilo najbolj všeč, saj večina ljudi pozna IMDB in bi jih napačne oziroma drugačne ocene, kot jih ima IMDB, le zmedle. Na začetku nisem našel druge rešitve, zato sem se odločil narediti prvotno verzijo spletne strani prek TMDB API-ja. Tukaj sploh

nisem potreboval zadnjega dela spletne strani, saj sem lahko vse API-klice opravil kar prek Reacta.

Po nekaj dneh razvoja sem se odločil, da bom še enkrat poskusil poiskati rešitev pridobitve podatkov iz IMDB-ja, saj je to bila moja prvotna ideja in bo v veliko pomoč pri uporabnosti spletne strani za uporabnike. Ena izmed rešitev ki sem jo našel, je bila razvoj programa, ki bi obiskal strani vseh filmov in serij na spletni strani IMDB ter bi prebral vse podatke o vsebini (angl. Scraping) in jih shranil v bazo podatkov. Ko sem začel malo bolj raziskovati to idejo, sem naletel na nekaj večjih problemov, predvsem na to, da ima IMDB več kot 6,5 milijona vsebin in da bi program potreboval več dni ali celo tednov, da pridobi in shrani podatke o vseh vsebinah. Potem pa bi te vsebine tudi morali osveževati dokaj redno, da pridobimo nove ocene in število ocen. Tako sem to idejo kar hitro črtal in začel raziskovati naprej.

Nato pa sem našel nekaj zanimivega. IMDB namreč ponuja uporabnikom tako imenovane »datasete« vsebin. To so velike datoteke, ki vsebujejo podatke o vseh vsebinah IMDB-ja, ki imajo vsaj nekaj ocen. Sicer ne vsebujejo vseh podatkov, ki jih jaz potrebujem za spletno stran, imajo pa ocene in število ocen, in to je glavna stvar, ki jo potrebujem. Ostale informacije lahko pridobim iz katerega drugega API-ja, na primer TMDB-ja.

Sedaj sem torej vedel, kje bom dobil informacije o vseh vsebinah, moral sem še samo ugotoviti, kako. Ti »dataseti« se posodobijo vsakih nekaj dni z novimi informacijami. Odločil sem se, da naredim tako imenovano »cron opravilo« (angl. Cron job), ki bo vsak dan ob tretji uri zjutraj izvršilo funkcijo, ki bo avtomatsko prenesla zgoraj omenjene »datasete« in jih shranila v mojo bazo podatkov.

Kaj pa sploh je »cron opravilo«? To so tako imenovane naloge, ki se izvajajo v nekem časovnem intervalu. Lahko na primer naredimo »cron opravilo«, ki bo vsak ponedeljek ob 17. uri poslalo e-pošto vsem naročnikom.

3.2.1 Baza podatkov

Za bazo podatkov sem se odločil uporabiti SQLite, ki je enostavna baza, ki shranjuje podatke kar v datoteke, kjer se nahaja naša spletna stran. Za to spletno stran, ki bo vsebovala samo eno tabelo, je ta enostavna baza več kot dovolj.

Baza bo vsebovala vse podatke o posameznih vsebinah, ki jih vrne IMDB. To so: ID, seznam kategorij, jezik, povezava do slike, letnica izdaje, povzetek, naslov, povezava do napovednika, število ocen in povprečna ocena.

3.2.2 Razvoj

Za razvoj bom uporabil Flask, ki je ogrodje, narejeno na osnovi programskega jezika Python. Za začetek na računalnik naložimo Python, verzijo 2.7, in Flask, verzijo 1.1.1. Nato naredimo novo mapo in v tej mapi naredimo datoteko app.py. Ime te datoteke je lahko karkoli, vendar se mora končati s .py, saj le tako računalnik ve, da je to koda, napisana v Pythonu.

V tej datoteki lahko sedaj začnemo pisati kodo. Najprej uvozimo (angl. Import) vse knjižnice ki jih bomo potrebovali.

```
from flask import Flask, jsonify, request, render_template
from flask_basicauth import BasicAuth
from flask_cors import CORS
import pandas as pd
import sqlite3
```

Slika 12: Uvoz knjižnic v ogrodje Flask

Tukaj smo uvozili:

- ogrodje Flask, da lahko uporabljamo vse njegove funkcionalnosti,
- avtentikacijo Flask, da lahko naš zaledni sistem zavarujemo z uporabniškim imenom in geslom,
- CORS (angl. Cross origin resource sharing), da lahko do te kode dostopamo tudi iz drugih strežnikov v primeru, da bi imeli prednji del in zaledni sistem na različnih serverjih,
- Pandas, da lahko beremo »dataset« podatke, ki jih pridobimo iz IMDB-ja,
- Sqlite3 za shranjevanje in branje iz baze SQLite.

Nato inicializiramo Flask, vključimo CORS ter nastavimo osnovno uporabniško ime in geslo za naš zaledni sistem. Tukaj ne potrebujemo več kot eno uporabniško ime in geslo, saj bo le CRON uporabljal avtentikacijo.

```
app = Flask(__name__)
CORS(app)
basic_auth = BasicAuth(app)

app.config['BASIC_AUTH_USERNAME'] = 'admin'
app.config['BASIC_AUTH_PASSWORD'] = 'lkjqw21diuqfg9031kqsd'
```

Slika 13: Inicializacija ogrodja Flask

Ko uredimo uvoze in inicializacijo ogrodja Flask, lahko začnemo s pisanjem funkcij. Kot sem že omenil, potrebujemo le dve glavni funkciji in dva »endpointa« (slov. končna povezava). Eno funkcijo potrebujemo za pridobitev in shranjevanje podatkov o vsebinah iz IMDB-ja v našo bazo in drugo za vračanje naključne vsebine glede na definirane filtre.

Naš zaledni sistem spletne strani bo dostopen prek API-poddomene, torej ko obiščemo »api.imbored.tv«, bomo dostopali do API-ja. Do tega dela strani bosta dostopala samo zaledni sistem in React. Uporabniki tukaj nimajo neposrednega dostopa.

Najprej se bomo lotili pridobivanja in shranjevanja vsebin. Za to bomo definirali »route« ali URL »update-media«. Torej ko obiščemo »api.imbored.tv/update-media«, se bo izvedla funkcija, ki bo prenesla in shranila vsebine v našo bazo.

Slika 14: Funkcija za posodobitev vsebin v naši bazi

Iz zgornje slike je razvidno, kako delujeta pridobivanje in shranjevanje vsebin. IMDB ima dva »dataseta«, enega, ki vsebuje ID-vsebine in ocene, ter še enega, ki vsebuje osnovne podatke o vsebinah. Torej moramo ta »dataseta« najprej prenesti in združiti v enega. To naredimo s pomočjo deljenja prenosa na posamezne dele (angl. Chunking) in prenašanja vsakega izmed teh manjših delov naenkrat. S tem zmanjšamo uporabo delovnega pomnilnika na računalniku iz 14,5 GB na 700 MB, kar je skoraj dvajsetkrat manjša poraba. Ko enkrat združimo oba »dataseta« v enega, ta združeni »dataset« pretvorimo v SQL in ga shranimo v našo bazo. Če neka vsebina v bazi že obstaja, jo seveda prepišemo, drugače pa dodamo novo vsebino.

Tako smo napisali glavno funkcijo za pridobivanje in shranjevanje vsebin, sedaj pa moramo na strežniku samo nastaviti »cron opravilo«, ki bo vsak dan ob 3. uri zjutraj obiskalo »**api.imbored.tv/update-media**« in se bo vsebina tako vsak dan avtomatsko posodobila z najnovejšimi ocenami.

Naslednji korak je razvoj funkcije, ki bo na podlagi filtrov, ki jih prejmemo iz prednjega oziroma začelnega dela, vrnila naključno vsebino. Sedaj, ko že imamo bazo z vsemi podatki vsebin, vključno z ocenami in številom ocen, bo ta funkcija dokaj enostavna.

```python
@app.route('/get-media')
def getMedia():
    # Shranimo vrednosti ki jih prejmemo iz prednjega dela v spremenljivke, za lažje dostopanje
    titleType = request.args.get('titleType')
    numVotes = request.args.get('numVotes')
    averageRating = request.args.get('averageRating')
    startYear = request.args.get('startYear')
    endYear = request.args.get('endYear')
    genre = '%' + (request.args.get('genre') or '') + '%'

    # Vspostavimo povezavo z bazo
    conn = sqlite3.connect('database.db')
    conn.text_factory = str

    # Poiščemo naključno vsebino glede na filtre
    sql = """SELECT DISTINCT * FROM media WHERE numVotes ⩾ ? AND averageRating ⩾
            ? AND startYear ⩾ ? AND startYear ⩽ ?  AND titleType = ? AND genres LIKE ? ORDER BY RANDOM() LIMIT 1"""
    params = (numVotes, averageRating, startYear, endYear, titleType, genre)
    cursor = conn.cursor()
    cursor.execute(sql, params)
    res = cursor.fetchall()

    # Če je vsebina najdena, jo vrnemo, drugače vrnemo 0
    if (res):
        return jsonify(res[0])
    else:
        return "No media found for defined filters", 404
```

Slika 15: Funkcija za pridobivanje vsebin

Kot vidimo na zgornji sliki, je funkcija za pridobivanje vsebin dokaj enostavna. Prek prednjega oziroma začelnega dela pridobimo vse podatke za filtriranje vsebin in te podatke uporabimo za iskanje po bazi. Če najdemo vsebino glede na filtre, ki smo jih podali, to vsebino vrnemo prednjemu oziroma začelnemu delu, drugače pa enostavno vrnemo napako 404, za katero bo prednji oziroma začelni del preverjal in prikazal ustrezno sporočilo uporabniku.

3.3 Prednji oziroma začelni del

Za prednji oziroma začelni del spletne stran bomo uporabili JavaScript ogrodje ReactJS. Zraven Reacta pa bomo uporabili še eno zelo uporabno knjižnico, ki se imenuje MobX. MobX je tako imenovana knjižnica za vodenje stanja spletnih strani z orodjem React. V osnovi se uporablja kot shramba za vrednosti, ki jih prikazujemo na spletni strani, ali se uporabijo za kakšne druge funkcije. Ko se neka vrednost v shrambi MOBX spremeni, se avtomatsko tudi posodobi na vseh straneh in v vseh komponentah, kjer smo to vrednost uporabili. Tako si ne rabimo pošiljati vrednosti iz komponent v druge komponente, zato da jih lahko prikažemo, temveč samo dostopamo do shrambe MobX, ki jo potrebujemo, vzamemo ven podatke in jih prikažemo.

3.3.1 Komponente

Dizajn smo že naredili. Sedaj moramo ta dizajn razdreti na logične komponente. To lahko vsak naredi malo drugače. Jaz sem se odločil, da bom stran najprej razstavil na tri glavne komponente: filtre, vsebino in slajder, nato pa bom te komponente razstavil še naprej, v še manjše komponente.

Slika 16: Vizualni prikaz komponent na strani

Ko sem imel pripravljen načrt vseh komponent, ki jih potrebujem, sem začel z razvojem. Tukaj še nisem ničesar povezoval z zadnjim delom, temveč sem delal samo na obliki posameznik komponent. Vsa vsebina (naslovi, ocene, slike itd.) je trdo kodirana (angl. Hardcoded), kar pomeni, da je vedno ista in še ni dinamična.

Naslednji korak je povezava prednjega z zadnjim delom. Najprej sem naredil shrambo MobX (angl. Store), v kateri bodo shranjene vse glavne funkcije, trenutno izbrani filtri, vsebina ipd.

Slika 17: Primer shrambe MobX s filtri

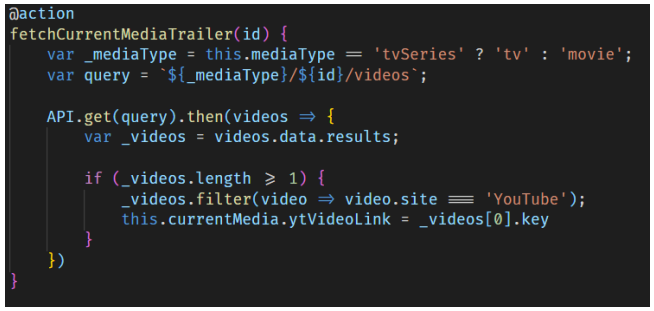
Kot je razvidno iz zgornje slike, imamo tudi filtre shranjene v spremenljivke kot JavaScript objekte. V teh filtrih imamo nastavljene začetne vrednosti, ki jih uporabnik vidi, ko obišče stran. Začetna letnica je torej nastavljena na 2010, končna pa na trenutno leto, jezik je nastavljen na angleškega in tako naprej. Te spremenljivke nato uporabimo v komponentah, ki smo jih že naredili. Če bi na primer hoteli na eni komponenti prikazati filter za začetno letnico izida, bi to naredili z naslednjo kodo: ***this.props.MediaStore.currentMedia.releaseYearFrom***. Ta vrednost bo vedno ista, ne glede na to, kje jo prikažemo oziroma uporabimo.

In kako lahko sedaj pridobimo vse podatke iz filtra, da jih lahko pošljemo zalednemu sistemu? Filtri so povezani z našo shrambo MobX, kar pomeni, da vedno, ko uporabnik spremeni katerega izmed filtrov, se spremeni vrednost tega filtra v naši shrambi. Ko torej uporabnik pritisne na gumb za nov predlog vsebine, vzamemo vse podatke filtra iz shrambe in jih pošljemo zraven v zahtevi (angl. Request) na zaledni system.

Slika 18: Klicanje zalednega sistema s podatki iz filtra

Ko dobimo iz zalednega sistema nazaj neko vsebino, jo shranimo v našo shrambo MobX kot objekt pod spremenljivko »currentMedia«, kar pa se nato avtomatsko prikaže v komponenti »media«, ki smo jo že naredili in pripravili. Torej vedno, ko se »currentMedia« posodobi, se posodobijo tudi vse komponente, ki uporabljajo to spremenljivko in njene vrednosti.

Sedaj uspešno pridobivamo priporočene vsebine glede na definirane filtre iz našega zalednega sistema. V naslednjem koraku je treba pridobiti napovednik izbrane vsebine in narediti funkcionalnost za zgodovino prikaza vsebin. Napovednike bomo pridobili prek TMDB API-ja. Na API pošljemo identifikacijsko številko, ali ID-vsebine, nazaj pa dobimo seznam napovednikov. Izbrali bomo prvega in ga shranili v našo shrambo MobX.



Slika 19: Funkcija za pridobivanje napovednikov vsebin

Zgodovina vsebin pa deluje tako: trenutno aktivna vsebina je shranjena v spremenljivki »**currentMedia**« in vedno, ko to vrednost spremenimo z novo vsebino, prej staro vsebino dodamo na seznam imenovan »**history**«, šele nato zamenjamo vsebino v **currentMedia**. Ta seznam »history« nato uporabimo za prikaz zgodovine vsebin. Če je seznam prazen, to pomeni, da uporabnik še nima zgodovine, zato ne bomo prikazali nič, če ima vsaj eno vsebino na seznamu, pa bomo vse te vsebine tudi prikazali na strani.

4. SKLEP

Namen tega projekta je bil razvoj spletne strani, ki uporabnikom na podlagi njihovih filtrov in nastavitev predlaga različne vsebine za ogled. Pri razvoju smo spoznali več programskih jezikov in ogrodij, vse od osnov, v obliki HTML, CSS, JavaScript in React do Python in Flask.

Glavno ogrodje, ki smo ga uporabljali, React, smo predstavili najpodrobneje, saj se ga je uporabilo za osrednji del spletne strani – del, ki ga uporabnik vidi in s katerim je v interakciji. Podrobno smo sledili tudi celotnemu procesu razvoja spletnih strani, od načrtovanja, oblikovanja ter do samega razvoja.

Med razvojem te spletne strani smo podrobneje razčlenili sistemska ogrodja, ki smo jih uporabili pri razvoju spletne strani. To je predvsem okrepilo pa pri logično strukturo gradnje in načrtovanja projekta, saj se velikokrat srečujemo z raznimi problemi. Ogrodje React Če bi to spletno stran hotel nadgraditi, bi bila prva stvar dodajanje možnosti registracija in prijave uporabnikov. Namen tega bi bil, da lahko natančneje predlagamo vsebine uporabnikom glede na njihovo zgodovino, in to vse od začetka njihovega računa.

Velika nadgradnja bi bila tudi možnost grajenja lastnih seznamov predvajanj, na katere lahko dodajaš vsebine, jih deliš z drugimi uporabniki in komentiraš. Komentiranje bi dodal tudi vsaki posamezni vsebini, da bi lahko uporabniki zapisali svoje mnenje in se o vsebini pogovarjali z drugimi uporabniki strani.

Ena nadgradnja bi bila tudi možnost izbire ponudnikov vsebine. Če torej hočeš, da ti stran predlaga samo vsebine, ki se predvajajo na spletni strani Netflix, bi to lahko izbral v filtru. To je le nekaj nadgradenj strani, ki sem si jih zamislil.

V uvodu diplomskega dela smo si zastavili tri hipoteze. Hipotezo H1: »Ogrodje ReactJS ima razvito napredno skupnost uporabnikov ter odprte repozitorije, ki so v pomoč pri gradnji spletnih strani« smo potrdili, ker smo ob izdelavi spletne strani odkrili različne odprtokodne podatkovne knjižnice, ki se osredotočajo na gradnjo spletnih strani.

Hipotezo H2: »Značilnosti ogrodja ReactJS omogočajo hitrejšo implementacijo naprednih funkcij v spletno stran, v primerjavi z drugimi ogrodji« lahko potrdimo, sicer ostala orodja

omogočajo podoben obseg del za zagotavljanje nadgradljivosti ter razširljivosti, ampak ReactJS omogoča hitrejšo implementacijo novih funkcij, zaradi fundamentalnih značilnosti ogrodja.

Hipotezo H3: »Ogrodje ReactJS omogoča zanesljivo posodabljanje vsebin iz drugih spletnih mest« lahko potrdimo, ker smo pri izdelavi spletne strani uporabili »cron opravilo«, ki vsak dan ob 3. uri zjutraj »api.imbored.tv/update-media« avtomatsko posodobi vsebino z najnovejšimi ocenami.

Sama stran je trenutno brez teh dodatnih funkcionalnosti, saj te niso bile načrtovane kot del diplomskega dela. S tem smo tudi omogočili odprte možnosti na nadgradljivost ter prilagodljivost spletne strani z ogrodjem React. Omogoča, da določimo, kako želimo obravnavati usmerjanje, testiranje in celo strukturo spletne strani.

Spoznali smo. da se React lahko uporabi za številne različne projekte: spletne aplikacije, statične strani, mobilne aplikacije in celo VR. Podpirajo ga vsi glavni brskalniki, vključno s starejšimi različicami IE, zaradi povezanosti s Facebookom pa bo še naprej deležen široke podpore.

V času, ki smo ga porabili za načrtovanje, razvoj in pisanje tega diplomskega dela, smo spoznali različne nove pristope do razvojnega postopka. Izdelana spletna stran je na voljo na spletnem naslovu: https://imbored.tv/.

5. LITERATURA

Angular vs. React vs. Vue: A performance comparison:. (2020). Pridobljeno iz Logrocket: https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/, 2021

Aston, B. (1. 4 2015). *A brief history of JavaScript.* Pridobljeno iz Medium: https://medium.com/@_benaston/lesson-1a-the-history-of-javascript-8c1ce3bffb17

Chinnathambi, K. (2018). *Learning React: A Hands-On Guide to Building Web Applications Using React and Redux.* Boston, ZDA: Addison-Wesley Professional; 2nd edition.

CSS W3Schools. (2020). Pridobljeno iz W3Schools: https://www.w3schools.com/css/, 2021

Flask vs Django. (2020). Pridobljeno iz Medium: https://medium.com/@SteelKiwiDev/flask-vs-django-how-to-understand-whether-you-need-a-hammer-or-a-toolbox-39b8b3a2e4a5, 2021

Mobx. (2020). Pridobljeno iz Mobx: https://mobx.js.org/getting-started, 2021

MobX vs Redux with React. (2020). Pridobljeno iz Codeburst: https://codeburst.io/mobx-vs-redux-with-react-a-noobs-comparison-and-questions-382ba340be09?gi=e01bd6336236, 2021

React. (2020). Pridobljeno iz ReactJS: https://reactjs.org/docs/getting-started.html, 2021

Sass Documentation. (2020). Pridobljeno iz Sass-lang: https://sass-lang.com/documentation, 2021