Usporedba radnih okvira Vue.js i React u izradi web aplikacija

Petrović, Ante

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1426

USPOREDBA RADNIH OKVIRA VUE.JS I REACT U IZRADI WEB APLIKACIJA

Ante Petrović

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1426

USPOREDBA RADNIH OKVIRA VUE.JS I REACT U IZRADI WEB APLIKACIJA

Ante Petrović

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1426

Pristupnik:	**Ante Petrović (0036535628)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Marko Horvat

Zadatak: **Usporedba radnih okvira Vue.js i React u izradi web aplikacija**

Opis zadatka:

Web aplikacije su danas najviše rasprostranjena vrsta računalnih aplikacija. Stoga je ispravan odabir programskog okvira za razvoj ovih aplikacija vrlo važan u programskom inženjerstvu. Cilj ovog rada je usporedba radnih okvira Vue.js i React u izradi reprezentativne web aplikacije istih funkcionalnosti. Upoznati se s programskim jezikom JavaScript i radnim okvirima temeljenih na ovom programskom jeziku. Odabrati domenu koja uključuje standardne funkcionalnosti web aplikacija. Modelirati i izraditi relacijsku bazu podataka za pohranu podataka o korisnicima. Implementirati autentifikaciju korisnika. Prikazati arhitekturu sustava i važne isječke programskog koda. Radu priložiti izvorni i izvršni kod razvijenog sustava uz potrebna dodatna objašnjenja i dokumentaciju. Citirati korištenu literaturu.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

1. Uvod

Otkrivanje, a potom standardizacija i široka uporaba interneta zauvijek su promijenili naše dosad analogno društvo. Internet je u vrlo kratkom roku postao ekosustav sam za sebe. Standardan pristup tom ekosustavu su upravo web stranice. Nekadašnje brošure i vizitke, koje su bile uobičajen način upoznavanja pojedinca s proizvodom, odnosno organizacijom, sada padaju u zaborav. Na njihovo mjesto dolaze web stranice kako za privatne, tako i za javne osobe. Uz to, internet pruža i direktne usluge preko web aplikacija, odnosno web stranica koje pružaju razne usluge poput trgoprodaje, izrade raznih materijala s pomoću online alata, međusobne komunikacije i mnoge druge stvari. To je veliki opus svakodnevnih i nužnih stvari koji je doveo do ubrzanog razvoja web tehnologija, odnosno načina za izradu, povezivanje i strukturiranje web stranica. Pogledamo li metode i tehnologije izrade web stranica sada i prije dvadeset godina, količina i složenost su neusporedive. Zbog velike količine tehnologija koje su danas prisutne pri izradi web aplikacija važno je odabrati pravu. U ovom radu napravit ćemo usporedbu dvaju od popularnijih razvojnih okruženja, React i Vue. Pokušat ćemo izraditi istu web aplikaciju, iste funkcionalnosti i istog izgleda, u Reactu i Vueu te usporediti razvojna okruženja i proces izrade.

2. Upoznavanje tehnologija

Prilikom izrade tehničkog dijela rada korištene su određene tehnologije koje ćemo pobliže upoznati u ovom poglavlju. Naglasak rada, kao što mu i tema nalaže, počiva na razvojnim okruženjima React i Vue. Neke osnovne funkcionalnosti i kratka povijest tih tehnologija dat će nam bolji uvid u tematiku ovog rada. Uz navedena razvojna okruženja, pri izradi web aplikacije koristimo još dvije tehnologije. To su Auth0, koji služi za autentifikaciju korisnika, XAMPP integrirani paket za razvoj web aplikacija koji služi za kreiranje jednostavnog lokalnog servera/poslužiteljski sloja , te OpenWeather API s kojim dobivamo podatke o vremenskoj prognozi pojedinog grada. Kratak opis tih tehnologija također je u ovom poglavlju.

2.1. Programski okvir React

Programski okvir React je popularna JavaScript knjižnica, razvojno okruženje za izradu korisničkih sučelja. Programerima omogućuje kreiranje interaktivnih i dinamičnih web aplikacija putem komponentnog pristupa. Ova knjižnica olakšava razvoj složenih aplikacija razbijanjem korisničkog sučelja na manje, ponovno upotrebljive komponente koje upravljaju vlastitim stanjem.

2.1.1. Povijest

U 2011. godini, programeri u Facebooku počeli su se mučiti s održavanjem koda. S obzirom na to da je Facebookova aplikacija za oglašavanje dobila povećan broj značajki, programerski tim trebao je zaposliti više članova. Ta promjena u kompleksnosti korisničkih sučelja značajno je usporavala napredak kompanije zbog neprestane potrebe za kaskadnim ažuriranjima. Kod je trebalo hitno unaprijediti za bolju efikasnost. Te iste godine, Jordan Walke stvara rani prototip Reacta pod nazivom FaxJS.[1]

2013. godine React postaje platforma otvorenog koda i dostupan je na integriranom razvojnom okruženju imena JSFiddle. Godine 2014. započeta je konferencija #reactjsworldtour. React se počinje širiti na tržištu i React Developer Tools postaju ekstenzija Chrome Developer Toolsa.

2.1.2. Karakteristike i značajke

Prilikom istraživanja karakteristika i značajki korišteni su sljedeći izvori.[2] [3] [4]

1. **JSX (JavaScript Syntax Extension)**: JSX je kombinacija HTML-a i JavaScripta. Omogućuje ugradnju JavaScript objekata unutar HTML elemenata. JSX nije podržan od strane preglednika, pa ga prevoditelj Babel transkompilira u JavaScript kod. JSX čini kodove lakima i razumljivima. Lako ga je naučiti ako poznajete prezentacijski jezik HTML i programski jezik JavaScript.

2. **Virtual DOM**: DOM je kratica za Document Object Model. React koristi virtualni DOM koji je kopija pravog DOM-a. Kada postoji izmjena u web aplikaciji, najprije se ažurira cijeli virtualni DOM i pronalazi razliku između stvarnog i virtualnog DOM-a.

3. **One-way Data Binding**: React koristi jednosmjerni protokol. Podaci teku samo u jednom smjeru, od nadređenih komponenti prema podređenim komponentama.

4. **Izvedba**: React koristi virtualni DOM i ažurira samo izmijenjene dijelove, što ubrzava rad DOM-a. DOM se izvršava u memoriji, pa možemo stvoriti zasebne komponente, čime se ubrzava rad DOM-a.

5. **Ekstenzije**: React ima mnogo ekstenzija koje se mogu koristiti za stvaranje UI aplikacija. Podržava razvoj mobilnih aplikacija i pruža prikazivanje na strani poslužitelja. Proširen je s Fluxom, Reduxom, React Nativeom itd.

6. **Uvjetne izjave**: JSX omogućuje pisanje uvjetnih izjava. Podaci u pregledniku prikazuju se prema uvjetima danim unutar JSX-a.

7. **Komponente**: React.js dijeli web stranicu na više komponenti. Svaka komponenta je dio dizajna korisničkog sučelja koji ima svoju logiku i dizajn. Komponente

koje su napisane u programskom jeziku JavaScript olakšavaju i ubrzavaju rad te se mogu ponovno koristiti.

8. **Jednostavnost**: React.js je temeljen na komponentama, što čini kod višekratnim. Koristi JSX koji je kombinacija HTML-a i JavaScripta, čime je kod lakši za razumijevanje i otklanjanje pogrešaka te ima manje koda.

9. **React Hooks**: Hookovi su dodani u React verziji 16.8. To su funkcije koje se mogu ponovno koristiti i omogućuju funkcijskim komponentama pristup stanju i drugim značajkama Reacta. Hookovi omogućuju da se "zakačimo" na značajke Reacta kao što su metode stanja i životnog ciklusa. Postoje tri pravila za hookove: mogu se pozivati samo unutar komponenti funkcije React, na najvišoj razini komponente i ne mogu biti uvjetni.

2.1.3. Zastupljenost na tržištu

Mnogi tehnološki divovi koriste React, uključujući: Uber, Airbnb, Facebook, Netflix, Pinterest, Instagram, Shopify, Amazon i mnogi drugi. React postoji već neko vrijeme i dostigao je stupanj zrelosti. Tijekom godina, React je stvorio čvrstu i lako dostupnu dokumentaciju i aktivnu zajednicu. Osjećaj stabilnosti koji proizlazi iz zrelosti ohrabruje mnoge tvrtke koje se oslanjaju na React kao ključnu komponentu u svom tehnološkom nizu. Na StackOverflow Developer Surveyu, React je izglasan kao jedna od najzastupljenijih web tehnologija 2023. godine. [5]

2.2. Programski okvir Vue

Vue je progresivni JavaScript okvir za izgradnju korisničkih sučelja. Omogućuje programerima stvaranje interaktivnih web aplikacija kroz komponentni pristup, s naglaskom na jednostavnost i fleksibilnost. Vue se može lako integrirati u projekte koji koriste druge JavaScript biblioteke, a također omogućuje razvoj složenih aplikacija

2.2.1. Povijest

Vue programski okvir je stvorio Evan You nakon što je radio za Google koristeći AngularJS u nekoliko projekata. Kasnije je sažeo svoj proces razmišljanja: "I figured, what if

I could just extract the part that I really liked about Angular and build something really lightweight." Prvi izvorni kod posvećen projektu datirao je u srpnju 2013., a Vue je prvi put javno objavljen sljedeće veljače, 2014. Nazivi verzija često su izvedeni iz manga i animea.[6]

2.2.2. Karakteristike i značajke

Prilikom istraživanja karakteristika i značajki korišteni su sljedeći izvori. [7][8]

1. **Virtual DOM**: VueJS koristi virtual DOM, slično kao React i Ember. Promjene se ne rade izravno na DOM-u. Umjesto toga, stvara se replika DOM-a u obliku JavaScript strukture podataka. Promjene se unose u ovu strukturu, koja se zatim uspoređuje s izvornom. Konačne promjene ažuriraju se na stvarni DOM, što je optimizirano, jeftinije i brže.

2. **Data Binding**: Povezivanje podataka pomaže u manipuliranju ili dodjeljivanju vrijednosti HTML atributima, mijenjanju stila i dodjeljivanju klasa uz pomoć direktive v-bind.

3. **Komponente**: Omogućuju stvaranje prilagođenih elemenata koji se mogu ponovno koristiti u HTML-u.

4. **Rukovanje događajima**: v-on je atribut koji se dodaje DOM elementima za slušanje događaja u VueJS-u.

5. **Animacija/prijelaz**: VueJS pruža različite načine primjene prijelaza na HTML elemente kada se dodaju, ažuriraju ili uklanjaju iz DOM-a. Postoji ugrađena prijelazna komponenta koja omotava elemente za učinak prijelaza. Mogu se dodati i biblioteke animacije treće strane.

6. **Izračunata svojstva**: Pomažu u slušanju promjena na elementima korisničkog sučelja i obavljanju potrebnih izračuna bez dodatnog kodiranja.

7. **Predlošci**: VueJS pruža predloške temeljene na HTML-u koji povezuju DOM s podacima instance Vue. Predlošci se kompiliraju u virtualne DOM Render funkcije.

8. **Direktive**: Ugrađene direktive poput v-if, v-else, v-show, v-on, v-bind i v-model

koriste se za izvođenje različitih radnji na sučelju.

9. **Watcher**: Watcheri se primjenjuju na podatke koji se mijenjaju, poput ulaznih elemenata obrasca. Oni automatski prate promjene podataka, pojednostavljujući kod.

10. **Usmjeravanje**: Navigacija između stranica obavlja se uz pomoć vue-routera.

11. **Lagan**: VueJS skripta je vrlo lagana, a performanse su vrlo brze.

12. **Vue-CLI**: Vue se može instalirati s pomoću sučelja naredbenog retka vue-cli, što olakšava izradu i kompajliranje projekta.

2.2.3. Zastupljenost na tržištu

VueJS programski okvir će zadržati, ako ne i povećati, svoju popularnost do 2025. godine. Njegova jednostavnost za početnike, u kombinaciji sa snažnim značajkama za napredni razvoj, čini ga privlačnim izborom za širok raspon programera.

2.3. Auth0

Auth0 je moderna platforma za upravljanje autentifikacijom i autorizacijom korisnika, koja pruža jednostavan način za dodavanje autentifikacije aplikacijama razvijenim u programskim okvirima React i Vue. Koristeći React Hooks, Auth0 olakšava upravljanje stanjem autentifikacije, omogućujući programerima da se fokusiraju na glavne funkcionalnosti svojih aplikacija bez brige o sigurnosnim protokolima. Suosnivači Eugenio Pace i Matias Woloski pokrenuli su Auth0 početkom 2013., a danas ga koriste velike tvrtke poput Atlassiana, Mazde i Siemensa. Među ključnim značajkama Auth0-a su univerzalna prijava, koja omogućuje centraliziranu stranicu za prijavu za sve aplikacije, te jedinstvena prijava (SSO), koja omogućuje prijavu u više aplikacija s jednim skupom vjerodajnica. Auth0 također podržava SAML za razmjenu autentifikacijskih i autorizacijskih podataka između različitih sustava, višefaktorsku provjeru autentičnosti (MFA) za dodatni sloj sigurnosti, te autentifikaciju bez lozinke putem metoda poput biometrije, SMS-a i e-pošte.[9][10]

2.4. XAMPP integrirani paket za razvoj web aplikacija

XAMPP integrirani paket za razvoj web aplikacija je besplatna platforma otvorenog koda za razvoj web aplikacija koja integrira Apache web server, MariaDB bazu podataka (ranije MySQL), PHP i Perl. Dizajniran od strane Apache Friends 2002. godine, XAMPP omogućuje jednostavno postavljanje lokalnog servera na različitim operativnim sustavima kao što su Windows, macOS i Linux, čime je postao popularan alat među web developerima zbog svoje jednostavnosti prilikom instalacije i upotrebe. Među značajkama XAMPP-a su: Apache HTTP poslužitelj, podrška za više platformi, MariaDB za upravljanje relacijskim bazama podataka, PHP za razvoj dinamičkih web stranica, Perl za administraciju sustava i web razvoj, te phpMyAdmin za grafičko upravljanje bazama podataka. Dodatno, XAMPP uključuje OpenSSL za implementaciju sigurnosnih protokola, kontrolnu ploču za jednostavno upravljanje komponentama, Mercury mail server za lokalnu e-mail komunikaciju i FileZilla FTP server za prijenos datoteka između lokalnog računala i udaljenih servera.[11]

2.5. OpenWeather API

OpenWeather API je online usluga u vlasništvu OpenWeather Ltd koja pruža globalne vremenske podatke putem API-ja, uključujući trenutne vremenske podatke, prognoze, trenutne prognoze i povijesne vremenske podatke. S ciljem olakšavanja pristupa preciznim i pouzdanim meteorološkim informacijama, OpenWeather API omogućava integraciju vremenskih podataka u različite aplikacije i usluge, što ga čini ključnim alatom za mnoge industrije poput poljoprivredne i putovanja. OpenWeather API podržava hiperlokalne prognoze oborina iz minute u minutu, te nudi razne endpointe, poput /weather za trenutne vremenske uvjete i /forecast za petodnevnu prognozu. Korisnici se trebaju registrirati kako bi dobili API ključ, koji se koristi za autentifikaciju svakog zahtjeva prema API-ju. Podaci se vraćaju u JSON formatu, što omogućava jednostavnu integraciju i obradu u različitim programskim jezicima i aplikacijama. Međutim, besplatna verzija API-ja ima ograničenja u broju zahtjeva, što možda biti ne dovoljno za aplikacije s velikim brojem zahtjeva.[12]

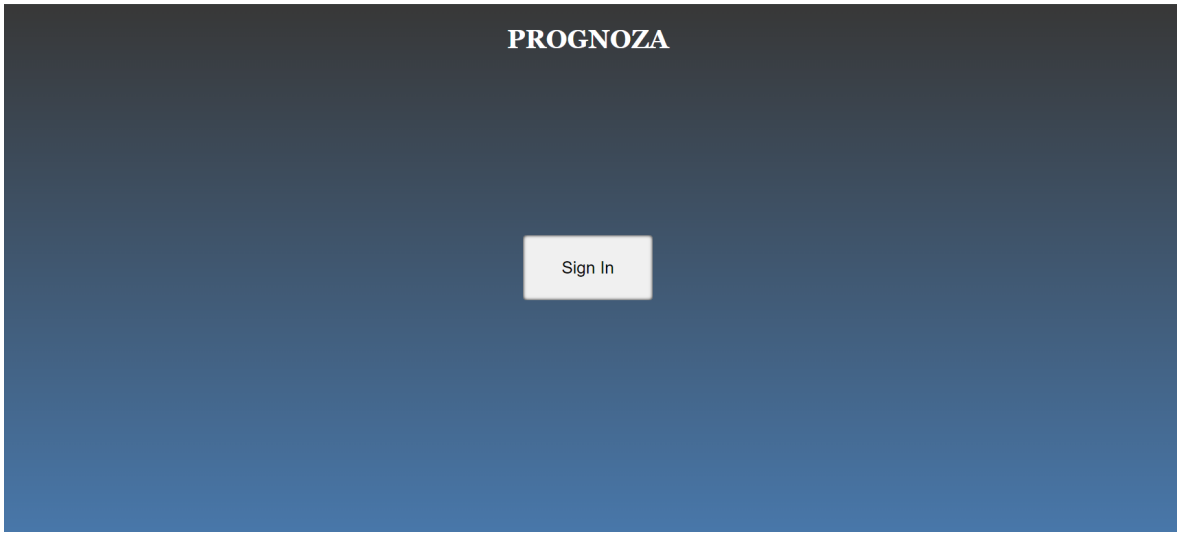
3. Implementacija programskih rješenja

Pri usporedbi programskih rješenja izrađene su dvije web aplikacije, jedna u React TypeScript-u, a druga u Vue JavaScript-u. Na početku ovog poglavlja pobliže ćemo upoznati samu strukturu web aplikacija, njihove zajedničke dijelove te pojedinačna rješenja, odnosno sličnosti i razlike u strukturi programskog koda tih dviju aplikacija.

3.1. Opis projekta

Projekt na temelju kojega uspoređujem dva razvojna okruženja, React i Vue, je web aplikacija za praćenje vremenske prognoze. Aplikacija radi tako da svaki korisnik ima vlastiti korisnički račun. Prilikom pokretanja stranice potrebno je prvo izvršiti registraciju, odnosno prijavu (Slika 3.1.). Stranica za prijavu jednaka je u objema aplikacijama. Po uspješnoj prijavi, stranica nas preusmjerava na HomePage (Slika 3.2., Slika 3.3.). Ta stranica je također izgledom jednaka u obje aplikacije. Na toj stranici nalazi se tražilica i lista gradova koje smo prethodno pretraživali. Upisivanjem nekog slova u tražilicu prikazuju se 5 najsličnijih gradova po imenu na svijetu. Svakom gradu pridružena je i oznaka države. Klikom na pojedini ponuđeni grad ili klikom na gumb s povećalom, koji odabire prvi ponuđeni grad, grad se u obliku kartice dodaje u listu gradova. Unutar liste gradova prvi grad je ujedno i trenutna lokacija. Ostali gradovi su gradovi koje smo prethodno s pomoću tražilice dodali. Svaka kartica grada sastoji se od: imena grada, ikone koja predstavlja trenutne padaline, temperature i kratkog opisa za trenutno vrijeme u pojedinom gradu. Ikona koja predstavlja padaline obojena je svijetlo plavom bojom, odnosno tamno plavom bojom, ovisno je li u pojedinom gradu u trenutku prikaza dan ili noć. Također, svaka dodana kartica u desnom gornjem kutu ima gumb za micanje grada iz prikaza. Klikom na bilo koju karticu grada, web aplikacija nas usmjerava na stranicu GrafPage.

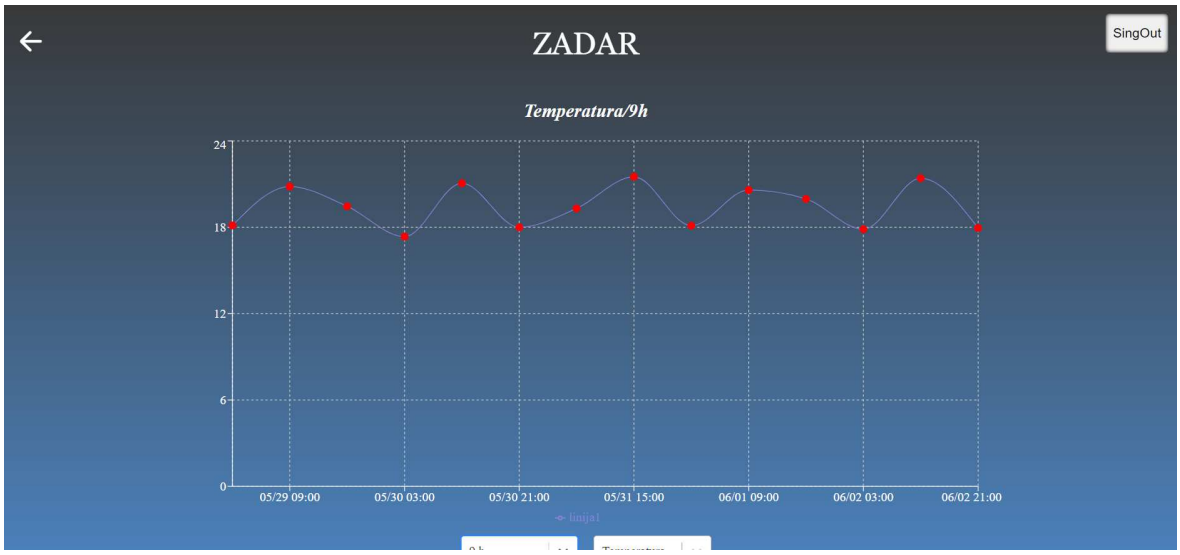
Prilikom izrade tih stranica nije bilo moguće napraviti identičan izgled, o čemu će biti više riječi u kasnijim poglavljima (Slika 3.4., Slika 3.5., Slika 3.6., Slika 3.7.). Za razliku od izgleda, struktura i funkcionalnost su identične. Na vrhu stranice nalazi se ime grada na koji smo prethodno kliknuli, odnosno želimo detaljnije pogledati vremensku prognozu. Ispod toga prikazan je tekst koji opisuje vrstu podataka i vremenski period unutar 5 dana kojeg želimo prikazati u grafu. Potom slijedi linijski graf koji prikazuje odabrani podatak kroz vrijeme. Ispod grafa nalaze se dva padajuća izbornika koja nam omogućuju odabir vrste podataka i vremenskih intervala prikaza na grafu. Vrste podataka su: temperatura, tlak, naoblaka, vlaga. Vremenski intervali su: 3 h, 6 h, 9 h, 12 h, 24 h. Na kraju stranice nalazi se tablica s podacima iz grafa. Tablica se sastoji od stupaca: datum, padaline (ikona padaline), opis padaline te podatak kojeg smo prethodno odabrali za prikaz u grafu. Tablica je dinamična, odnosno prilikom odabira različitih intervala ili vrste podataka za prikaz mijenja se sadržaj tablice. Stranice HomePage i GrafPage u svom desnom gornjem kutu imaju SignOut gumb. Klikom na njega korisnik se odjavljuje sa svog profila i preusmjerava se na početnu stranicu.



Slika 3.1. Prikaz LogInPage

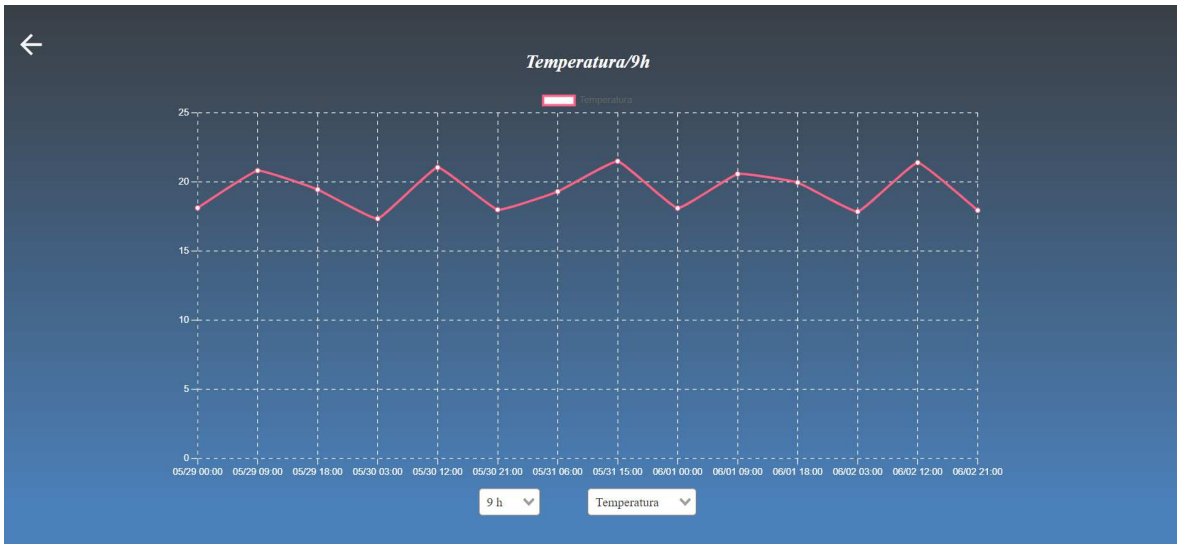
Slika 3.2. Prikaz HomePage

Slika 3.3. Prikaz HomePage prilikom unosa niza znakova u tražilicu



Slika 3.4. Prikaz React GrafPage, dio 1

Slika 3.5. Prikaz React GrafPage, dio 2



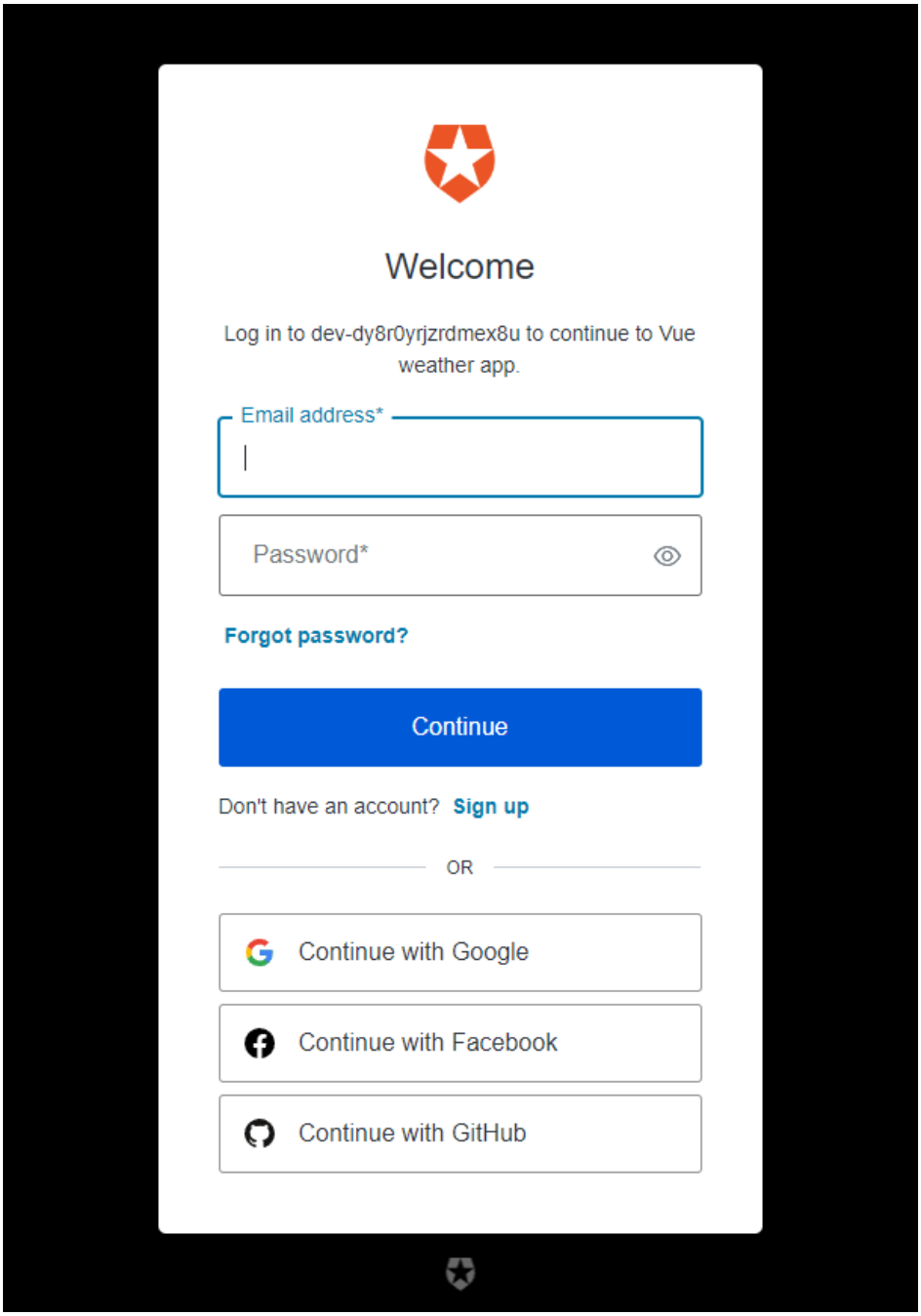
Slika 3.6. Prikaz Vue GrafPage, dio 1

Slika 3.7. Prikaz Vue GrafPage, dio 2

3.2. Autentifikacija korisnika

Autentifikacija korisnika odvija se preko tehnologije Auth0. Na stranicu nije moguće pristupiti bez prethodne prijave na korisnički račun. Prilikom prvog pokretanja, kako jedne tako i druge stranice, kod nas usmjerava na stranicu za prijavu (Slika 3.1.). U slučaju da smo prethodno prijavljeni, stranica nas automatski preusmjerava na stranicu koja služi za čekanje učitavanja pojedinih API-ja. U slučaju da nismo prijavljeni, kod nas vodi na stranicu Auth0 (Slika 3.8.), na kojoj se nakon uspješne registracije ili prijave preusmjeravamo na početnu stranicu koja nas dalje preusmjerava na stranicu za čeka-

nje učitavanja pojedinih API-ja. Prilikom uspješne prijave na stranici Auth0 dobivamo jedinstveni token pojedinog korisnika kojeg kasnije koristimo u poslužiteljskom sloju.



Slika 3.8. Auth0 login/register

3.3. Poslužiteljski sloj

Poslužiteljski sloj, koji je zajednički objema web aplikacijama, ostvaren je pomoću tehnologije XAMPP. S pomoću nje na lokalni server, odnosno samohostani server na localhostu, postavljena je jednostavna baza podataka te pripadne skripte napisane u programskom jeziku PHP. Skripte napisane u PHP-u koristimo za ostvarivanje komunikacije između baze podataka i samih web aplikacija. Baza podataka sastoji se od jedne tablice koja ima atribute id, lat i lon (Slika 3.9.). Atribut id je ključ koji dobivamo proslijeđen od stranice Auth0 prilikom uspješne prijave u sustav. Id nije jedinstven te nam služi isključivo za pamćenje lokacija pojedinog korisnika unutar baze podataka. Atributi lat i lon predstavljaju geografsku dužinu i širinu pojedinog grada kojeg je korisnik pretražio na stranici. Sve skripte nasljeđuju skriptu connect.php kojom dodajemo osnovna svojstva putanje pojedine PHP skripte. To uključuje korisnika, ime baze podataka i druge. Uz nju imamo još tri skripte: addCity, getCities i deleteCity. AddCity prima argumente id, lat i lon te potom te argumente sprema u bazu podataka hostanoj na lokalnom serveru uz pomoć XAMPP-a (Slika 3.10.). GetCities prima argument id te vraća listu objekata iz baze podataka s istovjetnim id-om. Objekti se sastoje od id, lat, lon (Slika 3.11.). Na posljetku imamo i skriptu deleteCity koja prima id te iz baze podataka briše n-torku s istovjetnim id-om (Slika 3.12.).

id	lat	lon
104014262584542425345	44.2318	14.8398
2628322150675627	44.1169	15.2353
104014262584542425345	42.69	18.0394
104014262584542425345	45.8132	15.9772
104014262584542425345	-33.8698	151.2083
2628322150675627	45.8336	17.3891
2628322150675627	45.8987	16.8422

Slika 3.9. Tablica baze podataka

```php
<?php
header('Access-Control-Allow-Origin:*');
include('./connection.php');

if(!isset($_POST['id'])&&!isset($_POST['lat'])&&!isset($_POST['lon'])){

    $response=['success'=>false,'message'=>'nisu dodani svi trazeni podaci','var dump'=>var_dump($_POST)];

}else{
    $id = $_POST['id'];
    $lat = $_POST['lat'];
    $lon = $_POST['lon'];


    $query="INSERT into listagradova (id, lat, lon) values('$id','$lat','$lon')";
    $result=mysqli_query($connect,$query);

    if($result){
        $response=['success'=>true,'message'=>'grad uspjesno dodan'];
    }else{
        $response=['success'=>false,'message'=>'grad nije dodan'];
    }

}
echo json_encode($response);
```

Slika 3.10. Dodavanje grada u bazu podataka

```php
<?php
header('Access-Control-Allow-Origin:*');
include('./connection.php');

if(!isset($_POST['id'])&&!isset($_POST['lat'])&&!isset($_POST['lon'])){

    $response=['success'=>false,'message'=>'nisu dodani svi trazeni podaci','var dump'=>var_dump($_POST)];

}else{
    $id = $_POST['id'];
    $lat = $_POST['lat'];
    $lon = $_POST['lon'];

    $query="delete from listagradova where id like '$id' and lat like '$lat' and lon like '$lon'";
    $result=mysqli_query($connect,$query);

    if($result){
        $response=['success'=>true,'message'=>'grad uspjesno izbrisan'];
    }else{
        $response=['success'=>false,'message'=>'grad nije izbrisan'];
    }

}
echo json_encode($response);
```

Slika 3.11. Brisanje grada iz baze podataka

```php
<?php
header('Access-Control-Allow-Origin:*');

include('./connection.php');

if(!isset($_POST['id'])){
    $response=['success'=>false,'message'=>'nisu dodani svi trazeni podaci'];
}else{
    $id = $_POST['id'];
    $query="Select * from listagradova where id like '$id'";
    $result=mysqli_query($connect,$query);

    $gradovi=array();

    while($row =mysqli_fetch_array($result)){
        $gradovi[]=array(
            'id'=>$row['id'],
            'lat'=>$row['lat'],
            'lon'=>$row['lon'],
        );

    }
    $response=['success'=>true,'data'=>$gradovi];
}
echo json_encode($response);
```

Slika 3.12. Dohvat svih gradova s traženim id iz baze podataka

3.4. Klijentski sloj

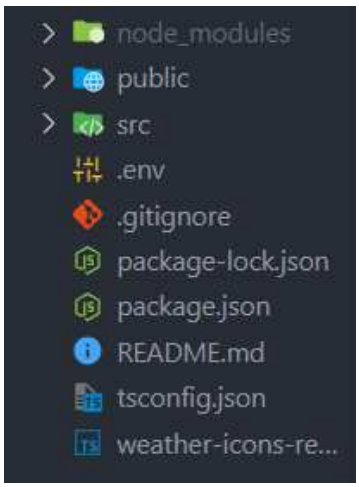
Prilikom sagledavanja klijentskog sloja, napravit ćemo razliku između koda implementiranog u React TS i Vue JS razvojnom okruženju. Pobliže ćemo opisati izvođenje funkcionalnosti stranice te programska rješenja koja su od interesa za usporedbu dvaju programskih okruženja.

3.4.1. Programski okvir React

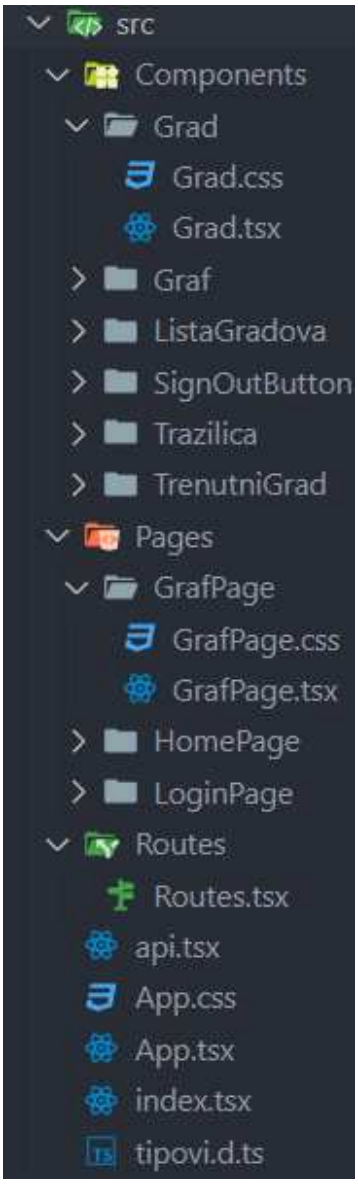
Web aplikacija za praćenje vremenske prognoze napisana u programskom okviru React koristi programski jezik TypeScript. Programski jezik TypeScript je proširenje JavaScripta koje uz sve funkcionalnosti JavaScripta dodaje tipiziranje programskog jezika. Odnosno, za razliku od programskog jezika JavaScripta, u TypeScriptu imamo objekte tipa string, int, Object i slično, a ne univerzalne tipove var i let.

Struktura aplikacije rađene u programskom okviru React (Slika 3.13.) sastoji se od datoteke .env, mape src i drugih datoteka i mapa automatski generiranih prilikom stvaranja

novog React projekta. Neke od tih generiranih mapa su i node-modules. Datoteka .env koristi se za pohranu ključeva za pristup OpenWeather API-ju i Auth0. Unutar mape src nalaze se tri podmape: Components, Pages i Routes te nekoliko dodatnih tsx datoteka poput api.tsx, index.tsx i drugih manje bitnih (Slika 3.14.).



Slika 3.13. Sadržaj React Projekta



Slika 3.14. Sadržaj src podmape

api.tsx

Datoteka api.tsx sadrži TypeScript funkcije koje se koriste za uspostavljanje veze s bazom podataka, odnosno za pozive PHP funkcija prethodno opisanih u poglavlju o poslužiteljskom sloju. Uz komunikaciju s bazom podataka, tu su prisutne i funkcije za dohvaćanje traženih podataka putem OpenWeather API-ja. Te funkcije su searchGrad, geocodeLocation, search5DaysWeather te searchWeatherHistory, koja se ne koristi jer se njena funkcionalnost otključava tek nakon plaćene pretplate na OpenWeather, ali je ostavljena radi potencijalnog budućeg rada. Funkcija searchGrad prima elemente lat i lon te se spaja na odgovarajući API preko kojeg pronalazi grad na tim koordinatama. Kao rezultat,

funkcija vraća osnovne podatke o gradu kao što su ime grada, država kojoj grad pripada, vremenska prognoza, vremenska zona i druge podatke. searchWeatherHistory s druge strane prima string koji sadrži slova te na temelju njega pozivamo jedan od OpenWeather API-ja kako bismo pretražili gradove koji su najsličniji poslanom stringu. Zatim vraća pet najpodudarnijih gradova na svijetu. Na kraju, funkcija search5DaysWeather prima argumente lat i lon te uz pomoć njih vraća, od trenutka kad je upit poslan, listu od četrdeset elemenata. Svaki element liste je zapravo vremenska prognoza s dodatnim podacima, za vremenske intervale u razmaku od 3 sata.

Karakteristika funkcije napisane programskim jezikom TypeScript je ta da joj je deklariran tip povratne informacije i tip argumenata koje ta funkcija prima (Slika 3.15.).

```
export const search5DaysWeather =
async (lat: number,lon: number): Promise<WeatherDataFor5Days | null> => {
```

Slika 3.15. Primjer deklaracije TypeScrip funkcije

index.tsx

Datoteka index.tsx (Slika 3.16.) pokreće web aplikaciju. Možemo je nazvati temeljnim elementom koji se prvi izvršava pri pokretanju stranice. Unutar index.tsx napravljena je referenca na router koji koristim za povezivanje stranica i inicijalizira se Auth0.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';
import { RouterProvider } from 'react-router-dom';
import { router } from './Routes/Routes';
import { Auth0Provider } from '@auth0/auth0-react';
const domain:string=process.env.REACT_APP_AUTH0_DOMAIN || 'development';
const clientId:string=process.env.REACT_APP_AUTH0_CLIENT_ID || 'development';
const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);
root.render(
  <React.StrictMode>
    <Auth0Provider
        domain={domain}
        clientId={clientId}
        authorizationParams={{
          redirect_uri: window.location.origin
        }}
      >
        <RouterProvider router={router}/>
    </Auth0Provider>

  </React.StrictMode>
);
```

Slika 3.16. Sadržaj datoteke index.tsx

Ostale podmape unutar src

Unutar src-a također imamo prethodno spomenuta tri mape: Components, Pages i Routes. mapa Routes sadrži Routes.tsx datoteku koju koristim za definiranje ruta stranice. Druga dvije mape, Components i Pages, sadrže niz datoteka s pomoću kojih je napravljena aplikacija. Imajući na umu da koristimo programski okvir React tako da stvaramo Single Page Application, odnosno jednu web aplikaciju koja po potrebi učitava određene komponente, razlika između komponenti u mapi Components i mapi Pages ne postoji. Takva podjela uvedena je radi semantičkih razloga, odnosno to je praksa koja se uvodi radi dodatne preglednosti koda. Svaka komponenta sastoji se od svoje pripadne tsx i css datoteke (Slika 3.14.).

Unutar mape Pages nalaze se tri komponente: GrafPage, LoginPage i HomePage. Komponente unutar mape Components su pojedini manji segmenti stranice koje ćemo obraditi unutar komponenata stranica.

LoginPage

LoginPage (Slika 3.1.) je inicijalna stranica. Prilikom prve prijave ili za korisnike koji trenutno nisu prijavljeni na stranicu, učitava se LoadingPage na kojem je jedina mogućnost pritiska gumba "Sign in". Po pritisku gumba odvija se algoritam opisan u potpoglavlju 3.2. Autentifikacija korisnika.

HomePage

Pri uspješnoj prijavi korisnika u sustav, korisniku je dodijeljen njegov pripadni token od strane Auth0 poslužitelja. Takav korisnik preusmjerava se na adresu HomePageWrapper.tsx (Slika 3.17.). Taj omot stranice služi za provjeru tokena dobivenog pri prijavi. Dohvat tokena je asinkrona funkcija te se prilikom čekanja na stranici ispisuje tekst "Učitavanje...". Pri uspješnom dohvatu ključa učitava se stranica HomePage (Slika 3.2.), te joj se pridodaje argument user. Argument user u sebi sadrži token.

```
const HomePageWraper = (props: Props) => {
    const{user,isAuthenticated,isLoading }=useAuth0();
    if (isLoading) {
        return <div>Učitavanje...</div>;
    }
    console.log(user);
  return (

    <HomePage userr={user} />
  )
}
```

Slika 3.17. Sadržaj datoteke HomePageWraper.tsx

Stranica "HomePage" pri svom učitavanju povlači podatke iz baze podataka s ID-om tokena koji je proslijeđen od strane "HomePageWrapper"-a. Dodatno, stranica prikazuje trenutnu lokaciju ako je za to dobila dozvolu, odnosno ako je korisnik potvrdio pop-up poruku koja je zaprimljena pri učitavanju stranice. Jedna od glavnih komponenti stranice je tražilica. Tražilica je implementirana unutar mape "Components". Ona funkcionira tako da pri svakom novom upisanom slovu vraća string do tada napisanih slova

svojoj korijenskoj komponenti, u našem slučaju roditeljskoj. Način povratka informacija odvija se preko reference funkcije "handleChange" koju komponenta "Tražilica" prima kao argument (Slika 3.18.). Klasa roditelj, to jest "HomePage", uz pomoć API-ja "geocodeLocation" vraća pet najsličnijih imena gradova na svijetu, potom ta dobivena imena gradova vraća komponenti "Tražilica" koja ih prikazuje ispod same tražilice (Slika 3.3.).

Klikom na jedan od tih gradova, podatak se vraća "HomePage"-u, pamte se njegova geografska širina i dužina te se s tokenom trenutno prijavljene osobe sprema u bazu podataka. Pri uspješnom spremanju novog grada u bazu podataka, stranica ponovno učitava cijelu bazu podataka. Takve podatke s pomoću API-ja "searchGrad" pretvara u trenutnu vremensku prognozu pojedinog grada te ih prosljeđuje karticama liste gradova. Lista gradova je zasebno definirana komponenta unutar mape "Components". Lista gradova se sastoji od komponenata "Grad" i komponente "TrenutniGrad". Kao parametar "ListaGradova" prima trenutnu lokaciju, listu gradova dobivenih iz baze podataka, referencu na funkciju "onClickDelete" i token trenutnog korisnika (Slika 3.19.). Komponenta "TrenutnaLokacija" prima argument "trenutnaLokacija" te na temelju njega stvara karticu za prikaz trenutne lokacije. Uz trenutnu lokaciju, "ListaGradova" ima niz komponenti "Grad" koji kao argument primaju niz podataka već unaprijed pripremljenih parsiranjem podataka dobivenih API-jem za pojedini grad (Slika 3.20.) te im se proslijeđuje referenca na funkciju "onClickDelete" koja je postavljena kao akcija na pritisak gumba za brisanje u gornjem desnom kutu kartice. Za naš rad važna je funkcija "onClickDelete" koja je originalno definirana u "HomePage"-u te se aktivira prilikom pritiska na gumb za brisanje unutar pojedine kartice grada. Funkcija prima parametre iz kartice "Grad" kao što su token, lat i lon te s pomoću njih briše grad iz baze podataka. Po završetku brisanja grada iz baze podataka, stranica povlači novo stanje baze podataka te ga prikazuje u listi gradova. Klikom na bilo koju karticu "Grad" ili klikom na "TrenutniGrad", stranica se preusmjerava na "GrafPage" s argumentima u putanji stranice. Primjer "return" funkcije unutar komponente "Grad" (Slika 3.21.) i primjer React hook komponente "Grad" (Slika 3.22.).

```
type Props={
  onClick:(e:SyntheticEvent,)=>void;
  search:string|undefined;
  handleChange:(e:ChangeEvent<HTMLInputElement>)=>void;
  podatak:ApiResponseGradovi[];
  onClickTrazilica:(e:SyntheticEvent,item_grad_prep:ApiResponseGradovi)=>void;
};
```

Slika 3.18. Ulazni argumenti komponente "Tražilica"

```
interface Props  {
  podatak:WeatherData[];
  onClickDelete:(e:SyntheticEvent,sid:string,lat:string,lon:string)=>void;
  trenutaLoakcija:WeatherData;
  sid:string
}
```

Slika 3.19. Ulazni argumenti komponente "ListaGradova"

```
interface Props{
  id:String;
  rezultat:WeatherData;
  iconVrijeme:IconType;
  onClickDelete:(e:SyntheticEvent,sid:string,lat:string,lon:string)=>void;
  timezone:number;
  sid:string;
}
```

Slika 3.20. Ulazni argumenti komponente "Grad"

```
return (
  <div className='grad'>
      <div className='button_container' >
        <IconButton className='minus_button' icon={<Minus />} type='submit' onClick={(e)=>onClickDelete(e,sid,str1,str2)}/>
      </div>
      <Link to={"/graf/"+rezultat.coord.lat+"/"+rezultat.coord.lon }  className='grad2' >
        <h2 className='ime_grada'>{rezultat.name}</h2>
        <IconComponent className="icon_vrijeme" style={{color}}/>
        <div className='podaci'>
          <p id='temperatura' className='txt_prikaz'>Temp: {rezultat.main.temp} °C</p>
          <p id='padaline' className='txt_prikaz'>Vrijeme: {rezultat.weather[0].description}</p>
        </div>
      </Link>
  </div>
);
```

Slika 3.21. Primjer return komponente "Grad"

```
const [color, setColor] = useState<string>('green');
```

Slika 3.22. Primjer React hook-a

GrafPage

"GrafPage" (Slika 3.4., Slika 3.5.) sastoji se od naslova, grafa koji je zasebna komponenta, gumba izbora i tabličnog prikaza podataka iz grafa. Pri učitavanju stranice kroz URL šalju se argumenti potrebni za rad stranice. Argumenti koje stranica prima su geografska širina i geografska dužina. Primjer takvog URL-a za grad Zadar je: "http://localhost:3000/graf/44.1329/15.2339". "GrafPage" parsira te argumente te s pomoću njih poziva API "search5DaysWeather" koji vraća listu od 40 podataka za pojedino vremensko razdoblje počevši od trenutka slanja pa sve do 5 dana s razmakom od 3 sata po podatku. Također iz tih podataka možemo dobiti i ime grada koje se potom dinamički dodjeljuje naslovu na vrhu stranice. Podnaslov koji se nalazi netom poslije naslova opisuje podatke koji će biti prikazani na grafu. Inicijalno je postavljen na "Temperatura/9h" što znači da ćemo grafom promatrati temperaturu za odabrani grad u intervalima svakih 9 sati. Odabir podatka kojeg želimo promatrati i intervala koji pratimo može se podesiti putem padajućeg izbornika. Sama komponenta "graf" prima argumente "listaPodataka" vremenske prognoze, interval prikaza i vrstu podataka za prikaz (Slika 3.23.). S pomoću tih podataka i strane knjižnice 'recharts' generira se graf čija y-os predstavlja podatak koji promatramo, a x-os predstavlja vrijeme tako da je svaka točka x-os tipa datum-sat. Iste podatke dostupne grafu prima i tablica. Tablica je napravljena uz pomoć knjižnice 'x-data-grid'. Podaci su sortirani na temelju datuma počevši od najmanjeg prema najvećem, odnosno prati raspored točaka u grafu. Uz datum u tablici su prikazani i podaci o opisu vremenske prognoze za dani datum kao što je na primjer oblačno, vedro i sumaglica. Pod stupcem "padaline" generirana je ikona koja odgovara pojedinoj opisu padaline. Na posljednjem stupcu tablice predstavljen je traženi podatak, odnosno podatak odabran za prikaz u grafu. Graf kao i tablica napravljeni su dinamički, odnosno prilikom mijenjanja podatka kojeg želimo pratiti ili intervala u padajućim izbornicima, mijenja se i sadržaj tablice odnosno grafa.

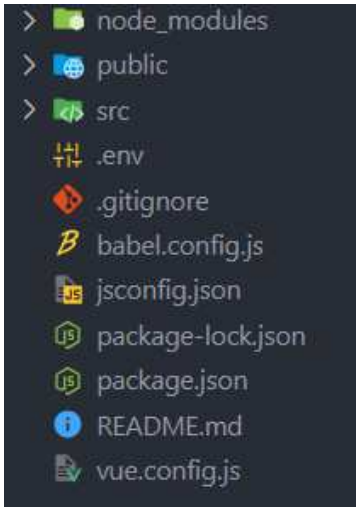
```
interface Props  {
  listaPodataka:WeatherEntry[],
  interval:number,
  tipGrafa:string,
}
```

Slika 3.23. Ulazni argumenti komponente "Graf"

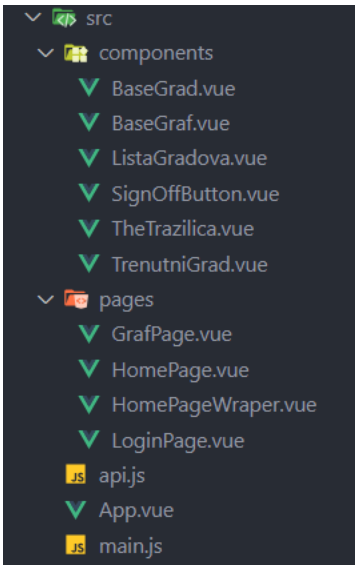
3.4.2. Programski okvir Vue

Kao što smo ranije učinili za web aplikaciju napravljenu u React razvojnom okruženju, u ovom potpoglavlju ćemo proći kroz neke ključne točke Vue web aplikacije. Aplikacija izrađena u Vue razvojnom okruženju, za razliku od Reacta, koristi programski jezik JavaScript, iako je također moguća uporaba programskog jezika TypeScript-a.

Struktura web aplikacije rađene u Vueu (Slika 3.24.) sastoji se od datoteka .env, mape src i drugih datoteka i mapa automatski generiranih prilikom stvaranja novog Vue projekta. Neke od tih generiranih mapa uključuju i node-modules. Datoteka .env koristi se za pohranu ključeva za pristup OpenWeather API-ju i Auth0. Unutar mape src nalaze se tri podmape: Components, Pages te neki dodatni JavaScript i Vue datoteke poput api.js, App.vue i main.js (Slika 3.25.).



Slika 3.24. Sadržaj React Projekta



Slika 3.25. Sadržaj src podmape

api.js

Datoteka api.js sadrži iste funkcije kao i api.tsx u React web aplikaciji. To su funkcije dohvata, spremanja i brisanja grada iz baze podataka, te funkcije dohvata podataka s OpenWeather API-ja: searchGrad, geocodeLocation i search5DaysWeather.

Razlika u funkcijama je ta što su ove funkcije napisane u JavaScriptu, odnosno nije deklariran tip podatka koji funkcija vraća niti tipovi argumenata koje funkcija prima (Slika 3.26.).

```
export const search5DaysWeather=async(lat,lon)=>{
```

Slika 3.26. Primjer deklaracije JavaScrip funkcije

main.js

Main.js (Slika 3.27.) obavlja istu svrhu kao i index.tsx unutar React web aplikacije. Razlika je u tome što kod Vue aplikacije nemamo zasebno definiranu komponentu "Routes", već njenu funkcionalnost definiramo u main.js. Osim toga, unutar main.js uvozimo razne biblioteke koje ćemo koristiti kroz aplikaciju te ih dodjeljujemo komponenti App.vue.

```
> import 'devextreme/dist/css/dx.light.css';···
> const vuetify = createVuetify({···
  });
> addIcons(···
  );
  const router = createRouter({
    history: createWebHistory(),
    routes: [
      { path: "", component: LoginPage },
      { path: "/homepage", component: HomePageWraper },
      { path: "/graf/:podatak1/:podatak2", component: GrafPage },
    ],
  });
  const domain = process.env.VUE_APP_AUTH0_DOMAIN;
  const clientId = process.env.VUE_APP_AUTH0_CLIENT_ID;

  const app = createApp(App);
  app.component("v-icon", OhVueIcon);
  app.component("v-select", vSelect);
  app.use(router);
  app.use(
    createAuth0({
      domain: domain,
      clientId: clientId,
      authorizationParams: {
        redirect_uri: window.location.origin,
      },
    })
  );
  app.use(vuetify);
  app.mount("#app");
```

Slika 3.27. Sadržaj datoteke main.js

Ostale mape unutar src

Unutar src-a također imamo dvije mape: Components i Pages. Obje mape sadrže .vue datoteke. Takve datoteke predstavljaju Vue komponente. Kao i kod Reacta, komponente se koriste radi semantičke strukture i veće preglednosti koda. U osnovi, nema razlike između .vue komponenti iz mapa "pages" i "components". Unutar mape "pages" nalaze se 4 datoteke: GrafPage, HomePage, HomePageWrapper i LoginPage. Te komponente,

radi jednostavnosti, nazivamo stranicama, iako su one samo prikazi istog DOM-a jedne stranice. Te stranice u sebi pozivaju i koriste komponente iz mape "Components". Komponente iz mape "Components" izvršavaju neke radnje odvojene po logičkoj pripadnosti, primjenjujući princip "Separation of concerns". Primjećujem da pojedine komponente nisu unutar mape, odnosno nemaju svoju CSS klasu. To je jedna od razlika strukture Vue i React razvojnih okruženja.

LoginPage

LoginPage (Slika 3.1.) je inicijalna stranica. Prilikom prvog posjeta ili za korisnike koji trenutno nisu prijavljeni na stranicu, učitava se LoadingPage na kojem je jedina mogućnost pritisnuti gumb "Sign in". Po pritisku gumba odvija se algoritam opisan u potpoglavlju 3.2. Autentifikacija korisnika.

HomePage

Pri uspješnoj prijavi korisnika u sustav, korisniku je dodijeljen njegov pripadni token od strane Auth0 poslužitelja. Takav korisnik preusmjerava se na adresu HomePageWrapper.vue (Slika 3.28.). Taj omot stranice služi za provjeru tokena dobivenog pri prijavi. Dohvat tokena je asinkrona funkcija te se prilikom čekanja na stranici ispisuje tekst "Učitavanje...". Pri uspješnom dohvatu ključa učitava se stranica HomePage (Slika 3.2.), te joj se pridodaje argument "user" te povratni argument "isReady". "IsReady" argument koji HomePage šalje svom roditelju, a njegova svrha je sprječavanje ponovnog učitavanja tokena jer je korisnik već prijavljen. Direktna veza u smjeru klase djeteta prema klasi roditelju ne postoji unutar React programskog okvira.

```vue
<template>
  <HomePage v-if="userr !== null && userr !== undefined" :userr="userr"  @isRedy="isReady" />
  <div v-else>Učitavanje...</div>
</template>

<script>
import HomePage from "./HomePage.vue";
export default {
  components: {
    HomePage,
  },
  data() {
    return {
      isReady: false,
      userr: null,
      isAuthenticated: this.$auth0.isAuthenticated,
      isLoading: this.$auth0.isLoading,
    };
  },
  watch: {···
  },
  mounted() {···
  },
};
</script>
```

Slika 3.28. Sadržaj datoteke HomePageWraper.vue

Stranica "HomePage" pri svom učitavanju povlači podatke iz baze podataka s ID-om tokena proslijeđenog od strane "HomePageWrapper"-a. Dodatno, stranica prikazuje trenutnu lokaciju ako je za to dobila dozvolu, odnosno ako je korisnik potvrdio pop-up poruku koja je zaprimljena pri učitavanju stranice. Jedna od glavnih komponenti stranice je tražilica. "Tražilica" je implementirana unutar mape Components. Ona funkcionira tako da pri svakom novom upisanom slovu vraća string do tada napisanih slova. Način vraćanja stringa do sada upisanih slova razlikuje se u odnosu na programski okvir React. Vue programski okvir ima ugrađen mehanizam komuniciranja s roditeljem, koji smo također vidjeli kod komunikacije "HomePageWrapper" i "HomePage". Tražilica uz argumente koje prima, emitira svojoj roditeljskoj klasi, odnosno HomePage, string koji je trenutno upisan u tražilicu. Emitirani string se obrađuje u "HomePage" te se uz pomoć API-ja searchGrad dobiva pet najsličnijih gradova po imenu na svijetu. Ti podaci se vraćaju "Tražilici" koja ih prikazuje ispod same tražilice na stranici (Slika 3.29.). (Opis ostatka klase HomePage jednak je opisu klase "HomePage" u React web aplikaciji, no važno je za spomenuti jer neki principi rada su zajednički Vue i React razvojnom okruženju. Jedan od njih je komunikacija od vrha prema dnu, o čemu će više biti rečeno u kasnijem poglavlju).

Klikom na jedan od tih gradova, podatak se vraća "HomePage"-u, pamte se njegove geografska širina i dužina te se s tokenom, trenutno prijavljene osobe, sprema u bazu podataka. Pri uspješnom spremanju novog grada u bazu podataka, stranica ponovno učitava cijelu bazu podataka. Takve podatke s pomoću API-ja searchGrad pretvara u trenutnu vremensku prognozu pojedinog grada te ih prosljeđuje karticama liste gradova. Lista gradova je zasebno definirana komponenta unutar mape Components. Lista gradova se sastoji od komponenti "BaseGrad" i komponente "TrenutniGrad". Kao parametar ListaGradova prima trenutnu lokaciju, listu gradova dobivenih iz baze podataka, referencu na funkciju "onClickDelete" i token trenutnog korisnika (Slika 3.30.). Komponenta "TrenutnaLokacija" prima argument trenutnaLokacija te na temelju njega stvara karticu za prikaz trenutne lokacije. Uz trenutnu lokaciju,"ListaGradova" ima niz komponenti "BaseGrad" koji kao argument primaju niz podataka već unaprijed pripremljenih parsiranjem podataka dobivenih API-jem za pojedini grad (slika 3.31.), te im se prosljeđuje referenca na funkciju "onClickDelete" koja je postavljena kao akcija na pritisak gumba za brisanje u gornjem desnom kutu kartice. Za naš rad važna je funkcija "onClickDelete" koja je originalno definirana u "HomePage"-u te se aktivira prilikom pritiska na gumb za brisanje unutar pojedine kartice grada. Funkcija prima parametre iz kartice "BaseGrad" kao što su token, lat i lon te s pomoću njih briše grad iz baze podataka. Po završetku brisanja grada iz baze podataka, stranica povlači novo stanje baze podataka te ga prikazuje u listi gradova. Klikom na bilo koju karticu "BaseGrada" ili klikom na "TrenutniGrad", stranica se preusmjerava na GrafPage s argumentima u putanji stranice. Primjer <template> dijela klase "BaseGrad" (Slika 3.32.) i primjer dodjele parametara klasi "BaseGrad" unutar klase Lista gradova (Slika 3.33.).

```
<script>
export default {
    data(){…
    },
    computed:{…
    },
    watch:{…
    },
    props:['onClick','search','podatak','onClickTrazilica'],
    emits:['myEmmitData'],
    methods:{…
    }
}
</script>
```

Slika 3.29. Ulazni argumenti komponente "Tražilica"

```
props: ["podatak", "onClickDelete", "trenutaLoakcija", "sid"],
```

Slika 3.30. Ulazni argumenti komponente "ListaGradova"

```
props: ["id", "rezultat", "iconVrijeme", "onClickDelete", "sid","theColor"],
```

Slika 3.31. Ulazni argumenti komponente "BaseGrad"

```
<template>
  <div class="grad">
    <div class="button_container">
      <button type="submit" @click="onClickDeliteWrap">
        <v-icon name= co-minus  />
      </button>
    </div>
    <div @click="redirectingOnClick()" class="grad2">
      <h2 class="ime_grada">{{ rezultat.name }}</h2>
      <v-icon
        :name="iconVrijeme"
        class="icon_vrijeme"
        :style="{ fill: theColor }"
      ></v-icon>
      <div class="podaci">
        <p id="temperatura" class="txt_prikaz">
          Temp: {{ rezultat.main.temp }} °C
        </p>
        <p id="padaline" class="txt_prikaz">
          Vrijeme: {{ rezultat.weather[0].description }}
        </p>
      </div>
    </div>
  </div>
</template>
```

Slika 3.32. Primjer <template> komponente "BaseGrad"

```
<BaseGrad
  v-for="rez in podatak"
  :id="rez.lon + rez.lat"
  :key="rez.lon + rez.lat"
  :rezultat="rez"
  :onClickDelete="onClickDelete"
  :theColor="changeStyles(rez.timezone)"
  :sid="sid"
  :iconVrijeme="
    WeatherIconGenerator(
      rez.weather[0].main,
      rez.timezone,
      rez.clouds.all
    )
  "
/>
```

Slika 3.33. Primjer dodjele parametara klasi "BaseGrad"

GrafPage

Funkcionalnost stranice GrafPage jednaka je funkcionalnosti stranice GrafPage u React web aplikaciji. Radi razlike u sintaksi koda i korištenim knjižnicama, nužno je ponoviti tekst radi kasnije usporedbe.

GrafPage (Slika 3.6., Slika 3.7.) sastoji se od naslova, grafa koji je zasebna komponenta, gumba izbora i tabličnog prikaza podataka iz grafa. Pri učitavanju stranice kroz URL šalju se argumenti potrebni za rad stranice. Argumenti koje stranica prima su geografska širina i geografska dužina. Primjer takvog URL-a za grad Zadar je: "http://localhost:8080/graf/
"GrafPage" parsira te argumente te s pomoću njih poziva API "search5DaysWeather" koji vraća listu od 40 podataka za pojedino vremensko razdoblje počevši od trenutka slanja pa sve do 5 dana s razmakom od 3 sata po podatku. Također, iz tih podataka možemo dobiti i ime grada koje se potom dinamički dodjeljuje naslovu na vrhu stranice. Podnaslov koji se nalazi netom poslije naslova opisuje podatke koji će biti prikazani na grafu. Inicijalno je postavljen na "Temperatura/9h" što znači da ćemo grafom promatrati temperaturu za odabrani grad u intervalima svakih 9 h. Odabir podatka koje želimo promatrati i intervala koje pratimo može se podesiti putem padajućeg izbornika. Sama komponenta

"BaseGraf" prima argumente listaPodataka vremenske prognoze, interval prikaza i vrstu podataka za prikaz (Slika 3.34.). S pomoću tih podataka i strane knjižnice 'chart.js/auto' generira se graf čija y-os predstavlja podatak koji promatramo, a x-os predstavlja komponentu vremena tako da je svaka točka x-os tipa datum-sat. Iste podatke dostupne grafu prima i tablica. Tablica je napravljena uz pomoć knjižnice 'vuetify'. Podaci su sortirani na temelju datuma počevši od najmanjeg prema najvećem, odnosno prati raspored točaka u grafu. Uz datume u tablici su prikazani i podaci o opisu vremenske prognoze za dani datum kao što je na primjer oblačno, vedro i sumaglica. Pod stupcem padaline generirana je ikona koja odgovara pojedinoj opisu padaline. Na posljetku, zadnji stupac tablice predstavlja traženi podatak, odnosno podatak odabran za prikaz u grafu. Graf kao i tablica napravljeni su dinamički, odnosno prilikom mijenjanja podatka koje želimo pratiti ili intervala u padajućim izbornicima, mijenja se i sadržaj tablice, odnosno grafa.

```
props: ["listaPodataka", "interval", "tipGrafa"],
```

Slika 3.34. Ulazni argumenti komponente "BaseGraf"

4. Usporedba programskih okvira React i Vue

Nakon ukratko opisanih značajki pojedine tehnologije te opisane njihove primjene na projektu, u ovom poglavlju opisat ćemo sličnosti i razlike dva programskih okruženja. Opisat ćemo iskustvo izrade projekta te napraviti tablicu prednosti i mana pojedine tehnologije. [13] [14]

4.1. Usporedba načina rada i strukture

Struktura projekta

Struktura pri izradi projekta, kako u Vue tako i u React programskom okviru, nije strogo definirana. Odnosno, ne postoji obrazac koji mora biti zadovoljen osim što mora imati ključne elemente kao što su main.js i index.tsx. Iako nije službeno pravilo razvojnih okruženja, postoje pravila dobre prakse koja se primjenjuju u oba razvojna okruženja. Neka od tih pravila su odvajanje ponavljajućeg koda u svoje komponente, stvaranje komponenti na principu "separation of concerns". Odvajanje natkomponenti i potkomponenti, što se vidi u projektu s komponentama Pages i ostalim komponentama koje Pages koriste. Također, dobra praksa je grupiranje komponenti koje su međusobno povezane u zasebne mape. Postoje i pravila imenovanja komponenti te mnoga druga pravila.

Struktura datoteke

Za razliku od strukture projekta struktura datoteka .jsx, odnosno u našem slučaju .tsx jer koristimo programski jezik TypeScript, i .vue datoteka je drugačija. U React programskom okviru koristimo .tsx datoteke (Slika 4.1.) čija se struktura sastoji od import deklaracija, funkcije istog imena kao i klase koja, uz razne funkcionalnosti, sadrži React hookove i return funkciju koja je u obliku sličnom HTML-u. Razlika u odnosu na

HTML je što u tsx funkciji imamo ugniježđene funkcije i varijable unutar HTML-a koje kasnije koristimo za dinamičku dodjelu vrijednosti. Bitno je primijetiti da .tsx datoteke u sebi ne sadrže CSS svojstva, ona se dodaju definiranjem zasebne .css datoteke. S druge strane, Vue programski okvir koristi .vue datoteke (Slika 4.2.). Te datoteke, odnosno komponente, sadrže tri glavne dijela: <template>, <script> i <style>. Možemo primijetiti da takva struktura podsjeća na "Vanilla" razvoj web aplikacija, odnosno na odvajanje HTML-a, CSS-a i JS-a. <template> sadrži HTML pomiješan sa nekim ugrađenim Vue direktivama kao što su v-on (skraćeno @), v-bind (skraćeno :), v-model i v-if i mnoge druge. Način na koji one funkcioniraju je da prilikom prijevoda <template> u JS kod, razvojno okruženje Vue prepoznaje ključne riječi kao što je na primjer v-if te uzima kod nakon nje i obrađuje ga po zasebnim pravilima. U suprotnom, Vue kompilira običan HTML. <script> je dio gdje se nalazi logika pojedine komponente te je podijeljena na više logičkih cjelina. Neki od važnijih su data, props, emits, watch i methods. Data sprema varijable koje koristimo unutar templatea za dinamičku dodjelu varijabli unutar <template>, odnosno obavlja istu ulogu kao i React hooks uz manju fleksibilnost. Props je struktura podataka koja nam govori o tome što pojedina klasa prilikom njenog poziva mora primiti kao argument. Emits je struktura koja nam govori što pojedina klasa daje nad klasom roditeljem. Unutar watch dijela definiramo pravila i funkcije koje se izvršavaju prilikom promjene pojedine varijable unutar segmenta data. Ovu funkcionalnost također obavlja React hooks. Potom imamo segment methods unutar kojeg definiramo JavaScript funkcije koje koristimo unutar komponente. Na kraju, imamo <style> u kojem pišemo CSS svojstva koja se najčešće primjenjuju unutar <template>.

Slika 4.1. Primjer strukture komponente "Grad.tsx"

Slika 4.2. Primjer strukture komponente "BaseGrad.vue"

Transformacia koda

React i Vue programski okviri dijele načine pretvorbe koda u web stranicu. U suštini, glavno načelo je pretvaranje .tsx i .vue koda u programski jezik JavaScript koji se zatim pretvara u web stranicu.

Reaktivnost

Problem reaktivnosti, odnosno automatskog renderiranja dijelova stranice prilikom promjene podataka za prikaz, jedan je od temeljnih zadataka zbog kojeg uopće postoji potreba za React i Vue programskim okvirima. React koristi jednosmjernu vezu podataka (one-way data binding), gdje komponente primaju podatke kao rekvizite i ažuriraju se samo kada se ti rekviziti promijene. U Vueu, ovo se postiže kroz dvosmjernu vezu podataka (two-way data binding), gdje promjene u modelu automatski ažuriraju prikaz i obrnuto. Primjer za usporedbu ova dva pristupa možemo primijetiti u komponenti Tražilica. U Vueu, uz pomoć ugrađene direktive v-model, u tražilici možemo čitati kod napisan od korisnika i upisivati kod s programom. Kod React programskog okvira, možemo slati podatke samo prema komponenti, odnosno ne možemo primati podatke nazad. Radi toga, u tražilici u web aplikaciji napisanoj u Reactu, imamo funkciju handleChange koja se aktivira pri promjeni stanja te komponente.

DOM

Kod oba razvojna okruženja, nakon zaprimanja novih vrijednosti za prikaz na web stranici, koristi se virtualni DOM. Virtualni DOM je kopija stvarnog DOM-a u kojoj se po potrebi vrše izmjene. Izmjene se događaju kada korisnik ili programski kod želi promijeniti postojeću varijablu unutar prikaza, odnosno kada se događa reaktivnost stranice. Virtualni DOM s izmjenama se uspoređuje sa stvarnim DOM-om, odnosno prikazom, te se unutar DOM-a mijenjaju samo dijelovi koji se razlikuju između njega i virtualnog DOM-a. Ovaj proces postoji jer je DOM velika datoteka koju nije praktično ni učinkovito svaki put cijelu učitavati.

Komunikacija komponenti

Način implementacije reaktivnosti određuje i način komuniciranja komponenti. U programskom okviru React pravilo je da se podaci spuštaju odozgo prema dolje. Takav pris-

tup zahtijeva da u korijenskoj klasi obrađujemo podatke, na primjer pozivamo API i obrađujemo podatke, te da obrađene podatke kasnije šaljemo komponenti djetetu. Primjer u kodu je Tražilica, njen algoritam opisan je u poglavlju 3.4.1 React Homepage. Za razliku od Reacta, u Vue programskom okviru moguće je slati podatke u smjeru odozdo prema gore, odnosno od djece prema roditeljima. Takav pristup također je opisan u Komponenti Tražilica u poglavlju 3.4.2 Vue Homepage.

4.2. Zapažanja tijekom izrade projekta

Izrada web aplikacija u React programskom okviru odvija se jednolikom brzinom. Svaki segment aplikacije, od dohvata API-ja do izrade pojedine komponente, bio je podjednako zahtjevan. Najveći problem predstavljala mi je uspostava komunikacije između komponenti od vrha prema dnu. Sintaksa React programskog okvira je intuitivna te ne uvodi veliki broj novih koncepata.Iako je sintaksa intuitivna, susreo sam se s novim konceptima u nepoznatom okruženju. Radi jednostavnosti sintakse mogao sam odmah započeti s pisanjem React projekta, počevši od najjednostavnijih komponenti. Problem sam primijetio prilikom pretraživanja interneta u svrhu učenja razvojnog okruženja. Postoji velika količina starih videa i materijala čija sintaksa više nije u uporabi ili barem dijelom ne odgovara sadašnjem obliku. Manjak konzistencije u sintaksi nadoknađuje velik broj kvalitetnih knjižnica kao što je 'recharts'. Programski okvir React, radi načina implementacije reaktivnosti, ima brži prijelaz i učitavanje komponenti.

Izrada web aplikacija u Vueu imala je dijelove sporog i brzog razvoja. Sam početak i savladavanje koncepata i sintakse išlo je dosta sporije u odnosu na programski okvir React. Velika količina direktiva kao što su v-on, v-model i druge usporavaju učenje same tehnologije. Uz velik broj direktiva, <script> dio .vue datoteke sadrži veliku količinu novih koncepata kao što su watch, data i druge. Efektivno pisanje koda ne može započeti dok se ne savladaju elementarni dijelovi aplikacije, što znači da se ne može odmah krenuti u izgradnju, već treba isprobavati i vježbati kod prije početka projekta. Kad se sintaksa jednom savlada, pisanje koda postaje dosta lako, lakše nego kod React programskog okvira jer postoje više specijaliziranih stvari nego React hooks, odnosno ima niz ugrađenih funkcionalnosti koje ubrzavaju i olakšavaju pisanje koda. Učenje sintakse olakšava jako dobro napisana dokumentacija za Vue. Gotovo suprotno mogu reći o knjižnicama.

Knjižnice za Vue su po broju, kvaliteti i održavanju dosta lošije od React knjižnica. Segment aplikacije u kojem koristim knjižnice za graf i tablicu znatno su usporile brzinu pisanja koda. Na primjer, knjižnica za tablicu, nakon implementacije, imala je bug koji je rezultirao nestankom gumba za listanje sadržaja tablice. Dok knjižnica za graf ima jako loše implementiran segment za tooltip i sve u svemu, graf izgleda dosta lošije od grafa u Reactu.

Također primjećujem da je kod napisan u React programskom okviru puno lakše prevesti u Vue kod, nego li obrnut slučaj.

4.3. Prednosti i mane

Razvojni okvir React

PREDNOSTI	MANE
Jednostavna sintaksa	Velik broj zastarjelih podataka
Velik broj knjižnica	Lošija dokumentacija
Popularnost	Veća konkurencija
Standard u industriji	Zahtjeva dobro poznavanje JavaSkripta (teži za početnike)
Razvijen od strane velike kompanije	
Provjeren na velikim projektima	

Razvojni okvir Vue

PREDNOSTI	MANE
Dobra dokumentacija	Lošije knjižnice
Jednostavnije za početnike (manje JavaSkripta)	Manji broj aktivnih programera
Porast zastupljenosti na tržištu	Nije standard u industriji
Manja konkurentnost	
Pregledniji kod	

5. Usporedba programskih jezika TypeScript i JavaScript

React i Vue programski okviri, kao standardne biblioteke za izradu korisničkih sučelja, koriste JavaScript kao svoj osnovni programski jezik. S obzirom na potrebe industrije i trend standardizacije, programski jezik TypeScript se počeo koristiti kao zamjena za JavaScript. React i Vue programski okviri podržavaju kreiranje projekata u TypeScriptu radi zadovoljenja tih potreba. Iako je TypeScript trenutno postao standardni jezik u industriji, postoji niz argumenata za i protiv zamjene programskog jezika JavaScripta TypeScriptom. Zbog toga sam odlučio napraviti jednu od ovih dviju aplikacija u TypeScriptu, a drugu u JavaScriptu, imajući na umu da je TypeScript nadskup JavaScripta, što znači da uključuje cijelu sintaksu JavaScripta uz dodatak mogućnosti deklariranja tipova.

Nakon završetka projekta došao sam do sljedećih zaključaka. TypeScript inicijalno usporava rad, ali na kraju sprječava greške. Deklaracija tipova varijabli pomaže manje iskusnim programerima da bolje razumiju kod. Programski jezik TypeScript povećava preglednost koda, ali u nekim situacijama, kao što su definiranje povratnih poziva API-a, može otežati preglednost dodatnom implementacijom interfejsa. Sam po sebi, TypeScript može koristiti i netipizirane varijable poput let i var. U konačnici, disciplina programera u korištenju tipova odlučuje hoće li korištenje TypeScripta biti efikasno.

Brzina pisanja koda u JavaScriptu je veća. Veći broj resursa za razvojna okruženja Vue i React napisan je u JavaScriptu. Programski jezik JavaScript je pogodniji za manje i samostalne projekte.

6. Zaključak

Izbor tehnologije jedan je od najvažnijih koraka pri izradi nekog projekta. Njegova važnost proizlazi iz činjenice da je to jedna od prvih odluka koju programer ili tim programera mora donijeti prije samog početka izrade projekta. U ovom radu uspoređujemo razvojna okruženja React i Vue. Funkcionalnost stranice, njezine karakteristike, uporaba i izgled igraju važnu ulogu pri odabiru razvojnog okruženja. Obje tehnologije imaju svoje prednosti i nedostatke. React je pogodan za brz razvoj. Dobar je izbor i za iskusne programere koji rade samostalno na manjim projektima i mogu iskoristiti sve pogodnosti alata i knjižnica koje pruža React zajednica. S druge strane, Vue ima jako dobru dokumentaciju osnova te mu je struktura komponenti dosta razgrađena, odnosno podijeljena u više dijelova. Takva struktura pogodna je za programere slabijeg iskustva jer mogu dublje razumjeti kako kod funkcionira. Također valja spomenuti da Vue ima bolju mogućnost parcijalne integracije u postojeće projekte. Oba se razvojna okruženja mogu koristiti pri izradi velikih projekata, s tim da moramo imati na umu da je većina projekata napravljena u Reactu te samim time na tržištu rada postoji više React developera. Same tehnologije u principu rade istu stvar te iskusan React developer ne bi trebao imati problema, uz brzu prilagodbu, raditi u Vue razvojnom okruženju i obrnuto.

Odabir tehnologije svodi se na trenutne okolnosti pojedinca ili kompanije. U slučaju programera početnika, bilo bi bolje koristiti Vue te iskoristiti pogodnosti koje on pruža. Dok za iskusnijeg programera, bolje je koristiti React jer otvara više prilika. Ako gledamo iz perspektive kompanije, kompanije koje rade na više manjih projekata, kao što su agencije, ili su nove na tržištu svakako profitiraju od odabira Reacta. Odabir Reacta im donosi veći opus potencijalnih programera te bolju kompatibilnost s raznim alatima za brži razvoj. S druge strane, firme koje posluju već duže vrijeme, na primjer one čiji je prethodni kod napisan u "Vanilli" (HTML, CSS, JS), imaju više smisla implementirati

pojedine segmente svoje aplikacije u Vue programskom okviru te postepeno mijenjati kod.

Literatura

[1] V. Komperla, P. Deenadhayalan, P. Ghuli, i R. Pattar, "React: A detailed survey", *Indonesian Journal of Electrical Engineering and Computer Science*, sv. 26, br. 3, str. 1710 1717, 2022. https://doi.org/10.11591/ijeecs.v26.i3.pp1710-1717

[2] legacy.reactjs.org, https://legacy.reactjs.org/, [Informacije korištene pri izradi projekta].

[3] A. Banks i E. Porcello, *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, Inc., 2020.

[4] devdocs, https://devdocs.io/react/, [Informacije korištene pri izradi projekta].

[5] a-team.global, https://a-team.global/blog/why-is-react-so-popular/, [Članak o zastupljenosti Reacta na tržištu].

[6] Wiki, https://en.wikipedia.org/wiki/Vue.js, [Članak o povijesti Vue-a].

[7] Vue.js, https://vuejs.org//, [Službena stranica Vue-a].

[8] C. Macrae, *Vue.js: Up and Running: Building Accessible and Performant Web Apps*. O'Reilly Media, Inc., 2018.

[9] auth0, https://auth0.com/blog/authors/eugenio-pace/, [Članak o Auth0 sa službene stranice].

[10] Frontegg, https://frontegg.com/guides/auth0, [Članak o Auth0].

[11] Javatpoint, https://www.javatpoint.com/xampp, [Opisana tehnologija XAMPP].

[12] OpenWeather, https://openweathermap.org/, [Službena stranica OpenWeather].

[13] A. S. Naik, “The front-end dilemma: How to choosethe perfect technology for your application.” *Journal of Computer Science and Technology Studies*, sv. 6, br. 1, str. 211–216, 2024. https://doi.org/10.32996/jcsts

[14] TechMagic, https://www.techmagic.co/blog/reactjs-vs-angular-vs-vuejs/, [Prednosti i mane pojedinog razvojnog okruženja].

Sažetak

Usporedba programskih okvira Vue.js i React u izradi web aplikacija

Ante Petrović

Cilj rada je usporediti razvojna okruženja Vue i React. Da bismo to efektivno napravili, u svrhu rada izrađene su dvije identične web aplikacije za praćenje vremenske prognoze. Jedna aplikacija koristi React, a druga Vue razvojno okruženje. Kroz rad prvo upoznajemo neke osnovne značajke tehnologija koje koristimo. Potom opisujemo izrađene aplikacije i naglašavamo razlike u kodu. Na posljetku uspoređujemo dvije tehnologije. Uspoređujemo način rada i iskustvo korištenja pojedinog razvojnog okruženja pri izradi web aplikacije. U zaključku navodimo skupine kojima bi određena tehnologija više koristila.

Ključne riječi: React; Vue; Način rada; Iskustvo; Aplikacija

Abstract

Comparison of Vue.js and React Frameworks in Web Application Development

Ante Petrović

The aim of this paper is to compare the development environments of Vue and React. To do this effectively, for the purpose of the paper, two identical web applications for weather forecasting were created. One application uses the React development environment, while the other uses Vue. Throughout the paper, we first introduce some basic features of the technologies we are using. Then, we describe the created applications and highlight the differences in the code. Finally, we compare the two technologies. We compare the working methods and user experience of each development environment in the creation of web applications. In conclusion, we identify the groups for which each technology would be more beneficial.

Keywords: React; Vue; Working methods; User experience; Application