Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/04BE36Subic

IMPLEMENTACIJA MOBILNE APLIKACIJE TRAVELIMG POMOĆU REACT NATIVE RADNOG OKVIRA

IMPLEMENTATION OF MOBILE APPLICATON TRAVELIMG USING REACT NATIVE FRAMEWORK

Nataša Subić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – RAČUNARSTVO I AUTOMATIKA

Kratak sadržaj – *U ovom radu prezentovan je alat za izgranju mobilnih aplikacija React Native. Kroz primer implementacije rešenja prikazani su osnovni koncepti ovog radnog okvira kao i biblioteke pomoću kojih je rešenje implementirano.*

Ključne reči: *JavaScript, React Native, REST*

Abstract – *This paper presents React Native tool for mobile application development. The paper contains basic concepts of this tool and descriptions of libraries used for implementation of the application.*

Keywords: *JavaScript, React Native, REST*

1. UVOD

Sa napretkom tehnologije došlo je do napretka mobilnih uređaja. Mobilni uređaji razvijaju se izuzetnom brzinom, povećava se pristupačnost istih i raste broj funkcionalnosti koje se pomoću njih mogu obaviti. Sve više i više veb aplikacija razvija se prvo za mobilne uređaje a kasnije im se prikaz prilagođava i za veće ekrane. Kako mobilni uređaji napreduju tako se razvijaju i mobilne aplikacije. Razvijaju se novi radni okviri za kreiranje mobilnih aplikacija i unapređuju postojeći. Jedan od radnih okvira za razvijanje mobilnih aplikacija jeste React Native. Ovaj radni okvir zasniva se na JavaScript programskom jeziku i o njemu će biti više reči u nastavku ovog rada. Tema ovog rada je mobilna aplikacija implementirana pomoću React Native radnog okvira.

2. REACT NATIVE

React Native je radni okvir pomoću kog se kreiraju mobilne aplikacije za Android i iOS mobilnu platformu [1]. Za razvijanje aplikacija koristi JavaScript programski jezik uz XML tagove i sve komponente prevodi u kod nativan odgovarajućoj platformi. Prednost ovog radnog okvira je što nema potrebe za razvojem aplikacija za pojedinačnu platformu. Mane su što se trenutno mogu razvijati aplikacije samo za Android i iOS mobilnu platformu. Bitno je istaći da postoji velika sličnost između ReactJS biblioteke i React Native radnog okvira.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.

ReactJS je JavaScript biblioteka pomoću koje se kreira korisnički prikaz za veb aplikacije [2]. Mnogi principi koji se koriste u ReactJS biblioteci koriste se i u React Native radnom okviru.

2.1. React Native komponenta

Komponenta predstavlja osnovnu jedincu za izgradnju prikaza aplikacije [3]. Ona omogućuje da čitav korisnički prikaz podelimo u više međusobno nezavisnih jedinica. Svaka od tih jedinica posmatra se kao izolovana celina koja se može više puta upotrebljavati u aplikaciji. Jedna izolovana celina čini blok čijim se spajanjem kreira potpun prikaz korisničkog interfejsa. React komponenta može biti JavaScript klasa ili funkcija. Funkcinalna komponenta koristi se za pisanje komponenti koje nemaju svoje stanje već samo prikazuju neke elemene korisničkog interfejsa. Klasne komponente služe za kreiranje kompleksnihih komponenti koje čuvaju i menjaju svoje stanje i koriste metode životnog ciklusa komponente.

2.2. JSX

JSX ili JavaScript eXtension predstvalja proširenje JavaScript-a i služi za opisivanje izgleda korisničkog interfejsa. JSX kombinuje XML sintaksu sa tagovima za označavanje i JavaScript kod [4]. JSX ne razdvaja logiku od prikaza nego ih sve spaja u jedinice odnosno komponente. Ona omogućuje ubacivanje prethodno definisanih promenljivih i poziv funkcija u HTML kod.

2.3. State i props

Razlikujemo dva tipa podataka koji kontrolišu komponentu. To su stanje komponente i svojstva komponente, odnosno props. Svojstva komponente su podaci koji se ne menjaju u toku života komponente. Za promenljive podatke koristi se stanje komponente.

Većina komponenti može se prilagoditi prilikom kreiranja pomoću različitih parametara. Ovi parametri nazivaju se props (skraćeno od properties, odnosno svojstva). Props-i služe za prosleđivanje podataka i svojstava od jedne komponente do druge. Obično roditelj komponenta prosleđuje svojstvo ili podatake dete komponenti. Oni omogućuju dinamiku i prilagodljivost svakog prikaza. Stanje komponente predstavlja objekat koji određuje kako će se neka komponenta ponašati ili šta će neka komponenta prikazivati. Samo klasne komponente mogu imati stanje. Ono uvodi dinamiku u komponente. Stanju jedne komponente može da pristupa samo komponenta koja je njen vlasnik i ona može da je menja.

Ukoliko je vrednost iz stanja potrebno proslediti drugoj komponenti to se može učiniti tako što se ta vrednost prosledi kao prop, komponenti koja je u hijerarhijskom poretku dete komponente iz koje se stanje prosleđuje. Svakom promenom stanja ili props-a komponenta ponovo osvežava svoj prikaz. Što je jedna od karakteristika React biblioteke i React Native radnog okvira.

2.4. Metode životnog ciklusa komponente

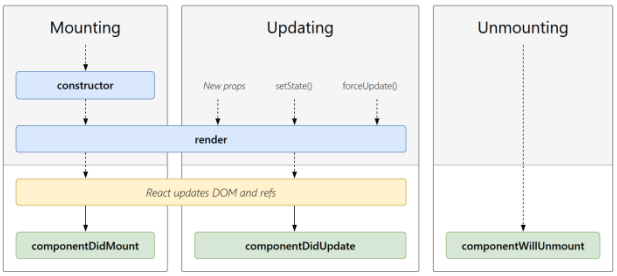
Jedna komponenta ima svoj životni ciklus od kreiranja pa do brisanja.

U različitim fazama životnog ciklusa u kojima se komponenta može naći pozivaju se različite metode.

Ove metode omogućuju da se na racionalniji način koriste resursi na mobilnom telefonu, da se kreira kvalitetnija aplikacija kao i da se isplanira šta i kako će se odraditi u određenoj fazi. Razlikujemo tri faze u kojima se komponenta može naći (Slika 1).

To su:

- Mounting,
- Ažuriranje,
- Unmounting.



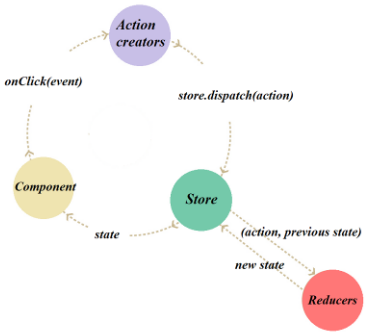
Slika 1. *Životni ciklus komponente i metode koje se pozivaju* [5].

U svakoj od ovih faza pozivaju se odgovarajuće funkcije životnog ciklusa. One omogućuju pravilnu manipulaciju podacima. Na slici 1 prikazan je životni ciklus React komponente za veb aplikacije. Iste faze životnog ciklusa primenjive su i na React Native komponentu. U prvoj fazi se inicijalizuje komponenta. U ovom periodu komponenta još uvek nije prikazana. Nakon inicijalizacije poziva se componentWillMount() metoda koja se poziva samo jednom pre metode render() i služi najčešće za dobavljanje podataka sa nekog eksternog servisa. render() je naredna metoda koja se pozove.

Ova metoda prikazuje elemente korisničkog interfejsa. Ona mora da vrati ili React Native komponentu, odnosno element ili ništa. Sledeća metoda koja se poziva jeste componentDidMount(), ona se isto kao i componentWillMount() poziva jednom ali razlika je u tome što se ona poziva tek nakon što je komponenta završila sa prikazivanjem elemenata korisničkog interfejsa. Druga faza je faza ažuriranja. Metode iz ove faze pozivaju se samo ukoliko je došlo do izmene stanja ili props-a.
Ukoliko dođe do ove izmene ponovo se poziva metoda render() koja sada prikazuje izmenjeno stanje i izmenjen props. componentDidUpdate() poziva se nakon što se elementi korisničkog intefejsa ponovo prikažu. Poslednja metoda koja se poziva jeste componentWillUnmount(). Ona se poziva pre nego što se komponenta uklanja iz prikaza i uništava.

2.5. Redux

Redux je JavaScript biblioteka koja manipuliše stanjem aplikacije i kontroliše tok podataka kroz aplikaciju [6]. Bitno je znati kada je treba koristiti. Ukoliko ima više podataka koji se menjaju tokom vremena i potreban je jedan izvor podataka za koji se uvek zna da je istinit onda treba koristiti Redux. Ako je aplikacija jednostavna nema potrebe za upotrebom Redux-a. Sve što je potrebno jeste da se određeni podaci čuvaju u stanju komponente na višem hijerarhijskom nivou. Na slici 2 prikazan je tok podataka Redux-a sa metodama koje se pozivaju.



Slika 2. *Tok podataka u Redux-u.*

Skladište (Store) predstavlja jezgro Redux-a koje povezuje akcije i reduktore. Postoji samo jedno skladište koje sadrži stanje aplikacije. Reduktor (Reducer) je čista funkcija koja prihvata prethodno stanje i akciju, obrađuje dobijene podatke i na osnovu te obrade vraća novo izmenjeno stanje ili trenutno stanje.

Promene stanja moraju biti immutable (nepromenljive) što znači da se objekat stanja ne sme menjati nakon što je kreiran, nego se uvek kreira novi objekat sa izmenjenim svojstvima. Akcije predstavljaju jedini izvor informacija za skladište. One su običan JavaScript objekat.

Akcije samo opisuju šta se dogodilo ne kako se stanje promenilo. Da bi se kreirala akcija koriste se funkcije za kreiranje akcija koje se nazivaju Action creators. Da bi se stanje izmenilo poziva se dispatch() metoda iz skladišta koja okida odgovarajuću akciju. Ta akcija prosleđuje se reduktoru koji je obrađuje i ažurira stanje.

3. REST

REST je arhitektonski stil koji obezbeđuje standardizaciju u komunikaciji između različitih sistema [7]. On uvodi ograničenja koja doprinose sklabilnosti i boljim performansama sistema u kom se isti koristi. REST arhitektonski stil definiše određena ograničenja koja se moraju poštovati. Adresabilnost je jedno od njih. Ovo ograničenje definiše da svaki resurs u sistemu mora imati jedinstveni identifikator, mora postojati uniformisani interfejs koji pojednostavljuje i odvaja arhitekturu.

Zatim, sve poruke moraju biti samoopisive poruke što znači da svaka poruka treba da sadrži dovoljno informacija o tome kako je obraditi.

Komunikacija treba da bude bez čuvanja stanja. Serverska strana ne čuva klijentsku sesiju, samo čuva stanje resursa koje nudi. Ovaj stil odvaja klijentsku od serverske strane.

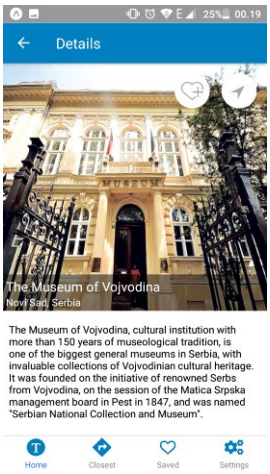
3. IMPLEMENTACIJA

Predmet ovog rada je softversko rešenje predstavljeno u vidu mobilne aplikacije. Mobilna aplikacija pod nazivom Travelimg nudi korisniku prikaz i pretragu znamenitosti u različitim gradovima uz oslonac na servise za lokaciju. Aplikacija zahteva pravljenje naloga i prijavu na isti. Korisniku takođe stoje na raspolaganju funkcinalnosti za prikaz, obeležavanje i pretragu znamenitosti. Pored toga korisnik ima mogućnost da vidi detalje konkretne znamenitosti i da izmeni podatke korisničkog naloga. Mobilna aplikacija implementirana je pomoću React Native radnog okvira, a server sa kojim ova aplikacija komunicira implementiran je pomoću Spring Boot radnog okvira.

Nakon što korisnik instalira aplikaciju prvi ekran koji mu se prikaže jeste ekran za prijavu na sistem. Da bi se korisnik uspešno prijavio na sistem mora se prethodno registrovati. Za prijavu na sistem potrebno je uneti korisničko ime i lozinku. Da bi se korisnik uspešno registrovao potrebno je da unesu korisničko ime, e-mail adresa i lozinka. Nakon što unese odgovarajuće podatke korisnik se preusmerava na početnu stranicu aplikacije. Sva polja za unos u okviru aplikacije se validiraju pre nego što se podaci pošalju eksternom servisu. Podaci koje korisnik unese šalju se serveru na proveru pomoću axios biblioteke [8]. Axios biblioteka koristi se za kreiranje HTTP zahteva i na taj način omogućuje komunikaciju sa REST API-jem. Pomoću ove biblioteke moguće je dobavljati i slati podatke koristeći GET, POST, PUT i DELETE zahteve iz aplikacije. Axios se zasniva na takozvanom Promise objektu. U Java Script-u Promise je objekat koji reprezentuje kompletno izvršavanje ili prekid asinhronih operacija.

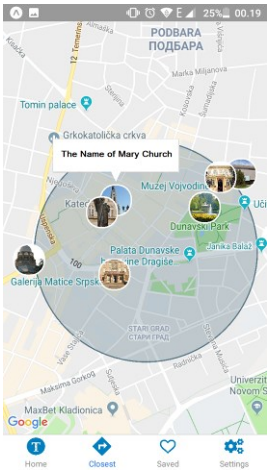
Slika 3. *Prikaz početnog ekrana mobilne aplikacije*.

Početni ekran, prikazan na slici 3, sadrži listu znamenitosti sa nazivom svake znamenitosti. U okviru ovog ekrana može se vršiti pretraga znamenitosti po nazivu ili lokaciji znamenitosti, odnosno po gradu i državi. Podaci koji se prikazuju u listi preuzimaju se iz skladišta podataka. To skladište kreira se upotrebom Redux biblioteke koja je opisana u prethodnom poglavlju. Redux je implementiran tako što je cela aplikacija izdeljena u domene. Ti domeni su na primer domen znamenitosti ili domen korisnika. Za svaki od ovih domena kreirani su posebni tipovi, akcije kao i reduktori. Za svaku akciju definisan je tip. Ovi tipovi koriste se u okviru akcije kao i reduktora. Akcije predstavljaju jedini izvor informacija koje se čuvaju u okviru stanja u skladištu. Reduktori definišu kako će se stanje aplikacije menjati u zavisnosti od akcije. Akcije samo opisuju šta se desilo, a reduktori opisuju promene stanja skladišta.
Pritiskom na konkretnu znamenitost na početnom ekranu, ekranu sa preporučenim ili obeleženim znamenitostima odlazi se na ekran sa prikazom detalja znamenitosti prikazan na slici 4.



Slika 4. *Prikaz detalja znamenitosti.*

Ekran sa mapom znamenitosti sadrži prikaz mape na kom se nalaze najbliže znamenitosti. Najbliže znamenitosti definišu se tako što se preuzme trenutna lokacija mobilnog telefona i prikažu se znamenitosti koje su u radijusu od distance koju je korisnik uneo u svojim korisničkim podešavanjeima. Inicijalna vrednost te distance je 500 metara. Za prikaz mape koristi se React Native Maps biblioteka [9]. Da bi se ova biblioteka koristila potrebna su dodatna podešavanja. . S obzirom na to da se prikaz radijusa koristi trenutna lokacija istoj se mora pristupiti putem telefona. Za to je potrebno da se definišu dozvole kao i prilikom implementacije nativnih aplikacija.



Slika 5. *Prikaz mape sa znamenitostima.*

Za prikaz mape koristi se Google mapa. Da bi se ista prikazala potrebno je u okviru aplikacije definisati API ključ koji se dobija od Google-a i koji omogućuje upotrebu mapa.

Bez istog se Google mapa ne bi mogla biti prikazana.
Pored prikaza znamenitosti na početnom ekranu korisnik ima mogućnost da vidi preporučene znamenitosti kao i one znamenitosti koje je sačuvao preko ekrana za prikaz detalja znamenitosti.
Preporučene znamenitosti su lista onih znamenitosti koje su korisnici najviše dodavali u svoju listu sačuvanih znamenitosti. Prikaz liste sačuvanih razlikuje se od prikaza liste preporučenih znamenitosti ne samo po podacima koji se prikazuju već i po tome što se sačuvane znamenitosti grupišu po gradovima u kojima se znamenitost nalazi. Još jedna od razlika jeste to što u stavci znamenitosti ne prikazuje samo naziv znamenitosti nego i trenutna udaljenost od te znamenitosti. Pored ovih funkcionalnosti korisnik može da izmeni podatke svog korisničkog naloga kao i distancu koja se koristi prilikom prikaza mape sa najbližim znamenitostima.

Serverska strana implementirana je uz pomoć Java programskog jezika u vidu REST API-ja. API omogućuje komunikaciju između dva softvera. U okviru implementirane aplikacije svaki zahtev koji se šalje ka serveru proizvodi odgovarajući odgovor. Kada se zahteva određeni resurs taj odgovor formatiran je u JSON formatu.

Pored toga u odgovoru se može naći i i informacija o uspešnosti izvršavanja određene akcije. REST je stateless što znači da ne čuva stanje aplikacije. U okviru rešenja ovaj stil se koristi preko HTTP protokola. Potrebno je navesti i način na koji se kreira šema baze podataka. Za to se koristi Hibernate biblioteka [10]. Pomoću ove biblioteke stavljaju se anotacije iznad polja određenog modela i na osnovu tih anotacija kreiraju se tabele u bazi sa definisanim kolonama.

4. ZAKLJUČAK

U ovom radu predstavljen je opis softverskog rešenja za prikaz i pretragu znamenitosti. Opisane su funkcionalnosti klijentske strane odnosno mobilne aplikacije. Za implementaciju klijentske strane korišćen je React Native radni okvir i JavaScript programski jezik. Serverska strana implementirana je pomoću Spring Boot radnog okvira i Java programskog jezika. Podaci su modelovani pomoću Java programskog jezika i Hibernate biblioteke na sistemu za upravljanje bazama podataka MySQL.

Softversko rešenje predstavljeno u ovom radu moglo bi se proširiti na više načina. Jedan od njih su obaveštenja koja bi stizala korisniku kada se određeno obeležje nalazi u njegovoj blizini. Pored toga potrebno je dodati mogućnost dodavanja komentara i prikaz istih u okviru stranice sa detaljima znamenitosti, kao i podršku za izmenu lozinke u slučaju zaboravljene lozinke. Da bi aplikaciju mogla više da se upotrebljava potrebno je odraditi lokalizaciju. Još jedna od mogućnosti jeste dodavanje audio zapisa za svako obeležje tako da korisnik ne mora da čita opis obeležja već može da sluša o istom putem tog zapisa.

5. LITERATURA

[1] React Native – A framework for building native apps using React, https://facebook.github.io/react-native/ (pristupljeno u aprilu 2019.)
[2] React - A JavaScript library for building user interfaces, https://reactjs.org/ (pristupljeno u aprilu 2019.)
[3] React Components and Props, https://reactjs.org/docs/components-and-props.html (pristupljeno u aprilu 2019.)
[4] JSX, https://reactjs.org/docs/introducing-jsx.html (pristupljeno u aprilu 2019.)
[5] Metode životnog ciklusa komponente, http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/ (pristupljeno u aprilu 2018.)
[6] Redux biblioteka, https://redux.js.org/ (pristupljeno u aprilu 2019.)
[7] RESTful Java with JAX-RS 2.0, Bill Burke
[8] Axios biblioteka, https://github.com/axios/axios (pristupljeno u aprilu 2019.)
[9] React Native Maps библиотека, https://github.com/react-native-community/react-native-maps
[10] Hibernate, Java persistence with Hibernate, Christian Bauer and Gavin King

Kratka biografija:

Nataša Subić rođena je 02.01.1994. godine u Novom Sadu. Završila je srednju ekonomsku školu „Svetozar Miletić“ u Novom Sadu 2013. godine. Fakultet tehničkih nauka u Novom Sadu upisala je 2013. godine, odsek Računarstvo i automatika. Osnovne studije završila je 2017. godine i iste se upisala na master studije, takođe na Fakultetu tehničkih nauka. Ispunila je sve obaveze i položila je sve ispite predviđene studijskim programom.