Primena tehnologije RiektJS u razvoju aplikacije iz domena elektronske trgovine

Nataša Vuksanović dipl. ing.*, prof. dr Dušan Vujošević, dipl. ing.

***Sadržaj* — Brzi napredak tehnologija podigao je razvoj rešenja za veb na novi nivo, koji sve više olakšava posao programera. Jedan od najčešće korišćenih stekova je MERN stek, koji danas igra vodeću ulogu u veb razvoju. Četiri komponente uključene u MERN stek su MongoDB baza podataka, Ekpres bekend veb frejmvork, RiektJS frontend biblioteka i Node.js JavaSkript okruženje. Ovaj rad se fokusira na razvoj aplikacije veb prodavnice poznatog kozmetičkog brenda Glosier koristeći JavaSkript tehnologije MERN steka. U radu su postepeno objašnjeni ključni koncepti pomenutih tehnologija, zajedno sa Ridaks, Riekt Rauter, JWT, Mongus i drugim bibliotekama. Proces kreiranja aplikacije je opisan od trenutka stvaranja ideje, do krajnje faze razvoja i implementacije. Kao rezultat, dobijena je potpuno funkcionalna aplikacija koje je u skladu sa prethodno postavljenim zahtevima. Ima jednostavan i intuitivan korisnički interfejs. Uključuje i admin panel koji služi administratoru za manipulisanje proizvodima, uključujući kreiranje, brisanje i ažuriranje. Ovaj rad se može koristiti kao kratak vodič za razvijanje ful stek MERN veb aplikacije.**

***Ključne reči* — elektronska trgovina, Ekspres, JavaSkript, MERN stek, MongoDB, Nod.js, Riekt, veb razvoj**

I. Uvod

NAPREDAK u komunikacionim tehnologijama u poslednjoj deceniji dvadesetog veka otvorio je put inovacijama, promovišući brzu globalizaciju. Internet je izrazito promenio način na koji se posluje, bilo da se radi o pronalaženju novih tokova prihoda, sticanju novih kupaca ili načinu upravljanja preduzećem. S obzirom na uticaj Interneta na gotovo svaki aspekt ekonomskog života, može se smatrati izvorom pojave nove, tkz. elektronske ekonomije, koja pruža nove mogućnosti za poslovne i industrijske aktivnosti i povećava mogućnost zapošljavanja. Sve ove prednosti Interneta dovele su do

*Nataša Vuksanović, Računarski fakultet, Srbija (e-mail: nvuksanovic1019m@raf.rs; nvuksanovic5@gmail.com).

pojave novog koncepta – *elektronska trgovina* ili *e-trgovina*. [1][2]

Elektronska trgovina je aktivnost elektronske kupovine ili prodaje proizvoda na mrežnim uslugama ili putem Interneta. [3] Svet biznisa zna da je Internet jedan od najboljih načina za poslovanje, na primer, proizvođač svoje proizvode može prodavati direktno javnosti, maloprodaje mogu proširivati svoje prodavnice na neograničene geografske lokacije, a preduzetnici imaju mogućnost osnivanja novog preduzeća veoma jeftino. Prevladavanje e-trgovine proširilo je poslovno okruženje tako da čak i mali start-ap može da se takmiči sa dobro utvrđenim poslovnim imenima i brendovima. Ipak, kada razmišljamo o pridruživanju zajednici e-trgovine, moramo imati na umu da prodaja proizvoda i usluga na Internetu predstavlja jedinstveni skup izazova.

Danas postoje različiti veb sajtovi, od jednostavnih statičnih sajtova, do onlajn gejming sajtova sa bogatim skupom interakcija. Svaki veb sajt je kreiran po određenom modelu, koji ima jedinstvene karakteristike. U ovom radu je predstavljen veb sajt model onlajn ili veb prodavnice. To je tip veb sajt modela koji se najčešće karakteriše kao sajt e-trgovine ili B2K (eng. Business to Consumer) sajt. Osnovna karakteristika onlajn prodavnice je da prikazuje proizvode ili usluge, kao i detaljne informacije (npr. specifikacije i cene), koje najčešće dobija iz baze podataka, sa svojstvima pretraživanja i metodom za onlajn kupovinu. Onlajn prodavnica takođe mora da pruža opsežne informacije o ponuđenim proizvodima ili uslugama koje ne samo da pomažu tako što privlače potrošače, već im i ulivaju poverenje u prodavca i proizvode ili usluge kako bi preduzeli sledeći korak – onlajn kupovinu.

Za uspešnu prodaju je bitna kvalitetna tehnologija. Prodajno okruženje, u ovom slučaju veb stranice, ono su što čini srž e-poslovanja. Možemo imati najbolji proizvod ili uslugu na svetu, ali ukoliko veb sajt ne pruža intuitivnu navigaciju, ne nudi jednostavan postupak naručivanja i ne održava visoke standarde kontrole kvaliteta, nikada nećemo postići dosledno zadovoljstvo kupaca. Veb sajt je, zapravo, konačan broj veb stranica povezanih zajedničkom temom i svrhom. Dobar dizajn je važan kako bi se kupcima omogućio lak pristup svim stranicama veb sajta. Bitno je pažljivo razmatrati brojne mogućnosti dizajna i tehnologija za izradu veb sajta. [4]

Osnovni cilj ovog rada je da se ilustruju i razumeju osnovni koncepti i upotreba svake od tehnologija MERN steka, sa akcentom na Riekt frontend tehnologiju, kao i njihove kompatibilnosti i prednosti u odnosu na druge veb tehnologije. Ideja je kreiranje veb prodavnice usled ogromnog porasta popularnosti e-trgovine i usled pružanja gomile prednosti i pogodnosti u odnosu na fizičke prodavnice.

II. MERN STEK

MERN stek je akronim koji označava skup od četiri tehnologije, čija je osnova JavaSkript, a koje se razlikuju po nameni. Svaka od ovih tehnologija se može koristiti odvojeno, ali zajedno kao celina čine osnovu novog pravca u pravljenju veb aplikacija. MongoDB služi kao baza podataka, Ekspres kao frejmvork za server, RiektJS kao klijentska biblioteka i Nod.js kao okruženje za pokretanje JavaSkripta na serveru.

A. Nod.js

Nod.js čini osnovu MERN steka, a predstavlja višeplatformsko rantajm JavaSkript okruženje otvorenog koda, koje izvršava JavaSkript kod na serverskoj strani. Omogućava da se JS skripte koriste i na serverskoj strani, što omogućava da se sadržaj dinamičnih veb stranica generiše na serveru pre nego što se pošalje do veb pregledača korisnika. Dakle, Nod.js je uveo paradigmu „JavaSkript svuda“ (eng. JavaScript everywhere). [5][6]

Kako svi pregledači raspoznaju samo HTML, CSS i JavaSkript, a ni jedna od tih tehnologija ne obezbeđuje pristup bazama podataka, kao ni serverski deo aplikacije, sve obrade na klijentskoj strani su se obavljale u JavaSkriptu, a na serverskoj strani najčešće u PHP-u, ASP-u i slično. To znači da je za isti deo programa bilo potrebno koristiti dve različite sintakse. Upravo je zbog ovoga nastao Nod.js.

B. Ekspres

Ekspres je mikro i fleksibilan frejmvork koji pruža brže i pametnije rešenje pri kreiranju serverske strane veb aplikacije. Čini programiranje prostijim i pruža programerima dodatna svojstva i alate kako bi nadogradili kod na serverskoj strani. Razvijen je od strane Holovajčuka (TJ Holowaychuk), u Maju 2010. godine. Uprkos činjenici da je sam Ekpres potpuno minimalistički, programeri su napravili mnoge kompatibilne midlver pakete i time rešili gotovo sve probleme u razvoju veba. Nudi brz i jednostavan način za kreiranje robusnog API-ja sa nizom korisnih HTTP metoda i midlvera. Pisanje glavnog serverskog skripta pomoću Ekspres frejmvorka sastoji se iz samo nekoliko koraka:

1. uvođenja potrebnih modula
2. instanciranja aplikacije

3. konfiguracije servera
4. konfiguracije posredničkih dodataka
5. definisanje ruta
6. obrade grešaka
7. pokretanja servera

C. MongoDB

MongoDB je NoSQL baza podataka. NoSQL je zapravo akronim nastao od izraza Not Only SQL i označava skup baza podataka koje nemaju principe relacionih baza podataka. NoSQL baze se mogu podeliti prema tipu, tačnije prema načinu na koji se čuvaju podaci u osnovi, gde MongoDB spada u baze koje se zasnivaju na modelu dokumenta. To bi značilo da se objekti podataka posmatraju i skladište kao dokumenti. Dokumenti imaju strukturu sličnu JSON formatu (BSON), a skup dokumenata, koji imaju istu namenu kao tabela kod relacionih baza, naziva se kolekcija. Svakom dokumentu dodeljuje se jedinstveni identifikator koji se smešta u polje _id. [7] [8]

Mongo baze podataka se koriste za opšte namene, ali pre svega za skalabilne aplikacije koje zahtevaju puno manipulacije sa ogromnom količnom podataka. Jedna od najznačajnih funkcionalnosti Mongo baza je mogućnost da skladišti dinamične podatke u fleksibilnom BSON formatu, što znači da dokumenti koji dele istu kolekciju mogu imati različita polja i ključ-vrednost parove, dok se struktura podataka može menjati bez ikakvih restrikcija i u bilo kom trenutku. Na ovaj način se uklanja potreba skladištenja strogo struktuiranih podataka, što je slučaj kod relacionih baza podataka, i značajno se povećava brzina operacija baza podatka.

D. Riekt.js

Riekt.JS je JavaSkript biblioteka otvorenog koda koja se koristi za izgradnju korisničkih interfejsa. Originalno je kreirana od strane Fejsbukovog softverskog inženjera Džordan Volke, konkretno za svojstvo zida novosti (eng. news feed), 2011. godine. Riekt je jednostavan za učenje, a zahvaljujući podršci Fejsbuka i snažne zajednice koja stoji iza njega, Riekt postaje sve popularniji i jedna od najčešće korišćenih JavaSkript biblioteka. [9] [10] [11]

Riekt se koristi za razvoj frontenda veb (RiektJS) i mobilnih (Riekt Nejtiv) aplikacija. RiektJS omogućava samo prikaz (renderovanje) podataka na

DOM (eng. Document Object Model), pa je zato potrebno koristiti i dodatne biblioteke za upravljanje stanjem i rutiranje, kao što su na prirmer Ridaks (Redux) i Riekt Rauter (React Router). Dve najbitnije činjenince o Riektu su:

1. pokreće se u pregledaču, a ne na serveru – ne moramo da čekamo da server odgovori kako bi se renderovala stranica
2. koristi komponente kako bi izgradio korisnički interfejs – Riekt kod je izgrađen od entiteta koji se zovu komponente. Komponente su nezavisni fragmenti koda koji se mogu koristiti iznova (eng. reusable).

Riekt koristi posebnu JavaSkript sintaksu JSX. Samim tim je potreban kompajler (BabelJS) koji prevodi JSX u kod koji će se normalno izvršavati u pregledaču.

Uz AJAX, virtualni DOM je jedno od najbitnijih otkrića kad je razvoj veba u pitanju. Virtualni DOM je ujedno i razlog što Riekt može da kreira tako brze i skalabilne veb aplikacije. Za statične veb sajtove, DOM radi savršeno. Međutim, kod dinamičnih veb sajtova, DOM može praviti velike probleme u performansama, posebno kod onih veb sajtova koji imaju veliku količinu interakcija i konstantna ažuriranja stranica. Do toga dolazi zato što se čitav DOM mora ažurarati kad se ažurira stranica. Drugi razlog je što DOM ima strukturu stabla, što dovodi do toga da je bilo kakva promena u stablu ekstremno zahtevna. Zbog toga je Riekt uveo virutalni DOM, koji predstavlja optimizaciju standardnog DOM-a. [20]

Kao što je već pomenuto, u Riektu komponente predstavljaju osnovne jedinice veb aplikacije. Pomoću komponenti Riekt omogućava programerima da podele korisnički interfejs na jednostavne i nezavisne komponente koje se mogu upotrebiti više puta. Primeri komponenti su recimo navigaciona traka, dugme, obrazac za unos, itd. Sve te komponente se mogu koristiti untar više različitih stranica i zato komponente predstavljaju savršen način da se skrati vreme razvoja aplikacije, pa samim tim se programeri mogu fokusirati na implementaciju važnijih svojstava i biznis logiku. Postoje dva tipa komponenti: komponente bazirane na funkcijama i komponente bazirane na klasama. [9] [10] [12] [13]

Komponente bazirane na funkcijama ili funkcionalne komponente su zapravo obične JavaSkript funkcije. Najčešće se nazivaju i glupim komponentama ili komponentama bez stanja, jer jednostavno prihvate podatke koje im se proslede i prikazuju ih u nekoj formi. Riekt metode životnog ciklusa se ne mogu koristiti u funkcionalnim komponentama. Takođe ne postoji metod za renderovanje, niti upravljaju. Odgovorne su za prikazivanje elemenata korisničkog interfejsa, pa su najčešće samo prezentacione komponente. Primaju svojstva (eng. props) kao argument i vraćaju Riekt element. Svojstva nam omogućavaju da prosledimo podatke sa roditeljske komponente (omotača) na potomka (ugnježdenoj komponenti).

Komponente bazirane na klasama ne samo da mogu vratiti JSX pomoću metode render, već takođe mogu otpremiti akciju, dohvatiti podatke, obraditi događaje i imati lokalno stanje. Ovakve komponente moraju naslediti Components klasu iz paketa Riekt. Time komponenta dobija i dodatna svojstva. Komponente na bazi klase se koriste u slučajevima kada je potrebno koristiti lokalno stanje ili metode životnog ciklusa. Unutar ugrađenog objekta stanje se čuvaju svojstva komponente. Kada se stanje promeni, komponenta se ponovo renderuje i dobija novo stanje. Pametne klase takođe imaju svojstva (props), samo što im se na drugačiji način pristupa nego kod svojstva funkcionalnih komponenti. Pomoću svojstva prosleđujemo podatke između komponenti, a pomoću stanja menjamo komponentu i to je osnovna razlika između objekta stanja i svojstva. [13]

Metode životnog ciklusa su niz događaja koji se izvršavaju od rođenja Riekt komponente do njene smrti. Zbog toga, tkz. život komponente možemo podeliti u tri ciklusa ili faze:

1. Faza montiranja ili kreiranja – rođenje komponente
2. Faza ažuriranja – rast komponente
3. Faza demontiranja – smrt komponente

Svaka od prethodno navedene tri faze životnog ciklusa ima određene metode koje se dešavaju u određenoj fazi.

III. IMPLEMENTACIJA APLIKACIJE

Nakon što su opisani najvažniji koncepti i svaka tehnologija MERN steka, u ovom poglavlju susrešćemo se sa praktičnom primenom MERN steka na aplikaciju e-trgovine. Aplikacija koja je napravljena predstavlja veb prodavnicu preduzeća Glosier (eng. Glossier). Glosier je poznati brend koji prodaje proizvode za negu lica. Korisnik može napraviti nalog i, ukoliko se prijavi, može kupiti proizvode. Postoje dve mogućnosti plaćanja, putem PejPel-a (PayPal) ili putem kreditne kartice. Cilj ove aplikacije je da korisnik može uspešno da kupi željene proizvode i da na jednostavan način može da koristi korisnički interfejs. Aplikacija takođe nudi opciju pretraživanja proizvoda po nazivu proizvoda.

A. Zahtevi aplikacije

Aplikacija je B2K (eng. Buisness to Customer) i sadrži dva tipa korisnika – administratora iliti admina, i korisnika. Admin je odgovoran za određene upravljačke zadatke, kao što su kreiranje, ažuriranje i uklanjanje proizvoda iz baze. Korisnik može da pregleda, pretražuje i čita detaljne opise proizvoda

koji se prikazuju na aplikaciji nakon što korisnik izabere proizvod i klikne na njega. Takođe, može dodati proizvod u korpu i izvršiti plaćanje za proizvod/e iz korpe. U bilo kom trenutku, korisnik ima potpuni pregled svoje korpe. Može uklanjati proizvode iz korpe i u svakom trenutku dodavati nove ili povećavati kvanitet određenog proizvoda koji želi da kupi. Neke rute su vidljive samo korisniku koji je prijavljen, a neke samo adminu.

B. Arhitektura aplikacije

Aplikacija je razvijena upotrebom MERN tehnologija - serverski deo je implementiran pomoću platforme Nod.js i Ekspres frejmvorka, a klijentski deo je implementiran za potrebe spoljašnjeg izgleda veb sajta pomoću RiektJS frejmvorka. Za bazu podataka je korišćena, naravno, MongoDB baza. Server obuhvata sve principe REST arihtekture. Kod SPA (Single Page Application), Riekt aplikacija i server komuniciraju tako što tipično razmenjuju JSON podatke. Za takav server kažemo da je REST ili RESTful (eng. Representational State Transfer) server. To znači da u osnovi koristi HTTP prokotol i model kako bi pružio usluge prenosa podataka. REST server treba da sledi klijent/server arhitekturu i da bude bez stanja (eng. stateless). Kažemo da je server bez stanja kada je implementacija klijentskog dela u potpunosti nezavnisna od serverskog.

Dakle, Riekt šalje zahtev serveru. Kada server prihvati zahtev, traži Ekspres rutu koja odgovara adresi sa kog je poslat zahtev i odgovarajući metod zahteva. Ruta, potom, obrađuje zahtev. Ukoliko ima potrebe, komunicira sa bazom podataka pomoću Mongus modula. Nakon toga, šalje odgovor nazad klijentu, odnosno Riekt aplikaciji. Riekt aplikacija prihvata odgovor i reaguje na određeni način na taj odgovor. Na primer, dobije potrebne podatke iz baze u odgovoru, koje će koristiti recimo za prikazavanje na nekoj stranici.

C. Razvoj aplikacije

Najpre ćemo videti kako izgleda baza podataka ove aplikacije. MongoDB baza je postavljena na platformi MongoDB Atlas. Aplikacija koristi jednu bazu podataka koja se nalazi u klasteru. Klaster se nalazi u Severnoj Virdžiniji (us-east-1) i sadrži tri skupa replika. Bazu podataka čine pet kolekcija: carts (korpe), products (proizvodi), users (korisnici), uploads.chunks i uploads.files. Kolekcije uploads.chunks i uploads.files se koriste za učitavanje slika od proizvoda.

Prva stvar koja treba da se uradi, kad je serverski deo aplikacije u pitanju, je postaviti Ekspres aplikaciju pomoću Ekspres generatora. Glavna datoteka

serverske aplikacije je server.js, gde se koriste neki od midlvera i Nod modula. Čitav kod datoteke server.js (eng. server.js) možemo raspodeliti na sedam delova: uvoz nezavisnih modula, uvoz ruta, kreiranje ekspres instance, konektovanje na bazu, kreiranje GridFS skladišnog pokretača, dodavanje midlvera na ekspres midlver stek i postavljanje ekspres instance da osluškuje na portu 5000. Ekpres modul se koristi za kreiranje ekspres instance. Moduli GridFS i multer su potrebni za učitavanje slika u MongoDB bazu.

Mongus je ODM (eng. Object Data Modeling) biblioteka koja se koristi za konektovanje MongoDB baze sa Nod.js. Upravlja relacijama između podataka, pruža validaciju shema i koristi se za prevođenje između objekata iz koda i reprezentacije tih objekata u Mongo bazi. [14] [15] Mongus pruža direktno rešenje zasnovano na shemi koja služi za kreiranje šablona podataka aplikacije. Modeli su konstruktori koji uzimaju shemu i kreiraju instance dokumenata ekvivalentne rekordima u relacionim bazama. Mongus shema je pokrivena unutar mongus modela, a određena shema kreira model. Primarna razlika između sheme i modela je sledeća: shema određuje formiranje dokumenta, podrazumevane vrednosti, validator itd., a model je odgovoran za operacije povezane sa dokumentima poput kreiranja, postavljanja upita, ažuriranja i brisanja. Ugrađeni validatori koje pruža Mongus olakšavaju rukovanje i validaciju podataka pre nego što se trajno uskladište podaci u bazi podataka. Međutim, pri kreiranju API-ja, podaci najjčešće dolaze iz HTTP zahteva do određenih endpointa i zbog toga postoji potreba da se podaci validiraju na nivou zahteva, a ne samo na nivou aplikacije. Zbog toga se često praktikuju oba tipa validacije. Za validaciju na nivou zahteva mogu se koristiti razni nezavnisni moduli, ali najpopularniji je Džoi (eng. Joi) modul. Pored toga što Džoi vrati validacionu grešku pre nego što se prijavljivanje ili registrovanje izvrši, Džoi pruža bolju kontrolu pri validaciji i omogućava kreiranje validatora sa jednostavnom i jasnom sintaksom. [16] [17]

Autorizacija je urađena pomoću JWT (eng. JSON Web Token) modula. Token je deo podataka koji sam po sebi nema nikakvo značenje ili upotrebu, ali u kombinaciji sa ispravnim sistemom tokenizacije postaje srž zaštite aplikacije. JSON Veb Token (JWT) je standard koji definiše kompaktan način za bezbedan prenos podataka, u JSON formatu, između klijenta i servera. Tokeni se mogu jednostavno prenositi putem URL-a ili zglavlja HTTP zahteva zahvaljujući svojoj kompaktnoj veličini. JWT token takođe sadrži sve neophodne informacije o korisniku. Podaci unutar tokena su verifikovani i pouzdani, jer se token potpisuje (eng. sign) sa skrivenim ključem ili javnim/privatnim parovima ključeva. Potpis takođe potvrđuje da ga je potpisao samo korisnik koja ima privatni ključ. Autentifikacijom proveravamo da li su korisničko ime i lozinka ispravni, dok se autorizacijom

proverava da li je korisnik koji je poslao zahtev serveru i korisnik koji je prijavljen tokom procesa autentifikacije isti korisnik. Autorizacijom kontrolišemo da li korisnik može imati pristup nekom resursu. Recimo, korisnik da bi kupio neki proizvod u veb prodavnici, mora biti prijavljen. Tradicijalno, autorizacija se vršila pomoću sesija. JWT umesto kolačića, koristi veb tokene. Nakon što se pošalju imejl i šifra serveru na autentifikaciju, umesto skladištenja tih informacija u memoriji sesije na serveru, server kreira veb token, enkodira ga i serijalizuje i dodeljuje mu tajni ključ. Dakle, osnovna razlika je u tome što se kod autorizacije zasnovane na tokenu podaci o korisniku čuvaju lokalno, odnosno unutar tokena, a ne na serveru kao što je to slučaj kod autorizacije zasnovane na sesijama. Što znači da možemo koristiti isti token na više servera. Tabela 1 pokazuje sve API rute sa odgovorajućom metodom i kratkim opisom.

TABELA 1: API DOKUMENTACIJA I RUTIRANJE

МЕТОД	ПУТАЊА РУТЕ	ОПИС
Аутентификација		
POST	/api/user/ register	Валидира податке о кориснику, уколико је све исправно, креира новог корисника у бази података и шаље генерисани JBT токен клијенту.
POST	/api/user/ login	Валидита имејл и шифру корисника. Уколико корисникова имејл адреса постоји у бази и прослеђена шифра је исправна за прослђену имејл адресу, шаље се генерисани JBT токен клијенту.
Производи		
POST	/api/products/	Верификује се да ли је пријављени корисник админ, уколико јесте, креира се нови производ са

		прослеђеним подацима о кориснику у бази података.
GET	/api/products/data	Узимају се сви подаци из колекције *производи* из базе података.
DELITE	/api/products/:title	Проверава се да ли је пријављени корисник админ, уколико јесте, брише се производ чији наслов одговара прослеђењеном наслову из базе података.
GET	/api/products/search	За прослеђену реч који је унео корисник у траку за претраживање, враћа производ или листу производа који садрже упитну реч у наслову.
Наруџбине		
POST	/api/orders/	Верификује се токен, креира се нова поруџбина у колекцији *корпа* у бази података.
GET	/api/orders/:token	Верификује се токен, на основу јединственог идентификатора корисника који је пријављен, претражује се одговорајућа поруџбина и шаље се клијенту.
Наплата		
POST	/api/payment/	Верификује токен и извршава наплату у америчким доларима на кредитној картици помоћу модула *страјп* (енг. Stripe).

Учитавање слика		
POST	/api/images /upload	Учитава се слика у колекцију *uploads.files* помоћу модула *GridFS*.
GET	/api/images /:filename	За дати назив слике, проналази у колекцији *uploads.files* одговарајућу слику и шаље је клијенту.

Za razliku od bekenda koji koristi više tehnologija, frontend se implementira kompletno pomoću frejmvorka Riekt.js. Veoma je kompleksno podesiti Riekt aplikaciju od nule, jer ovaj proces zahteva podešavanje Babel kompajlera za prevođenje JSX u JavaSkript, kao i konfigurisanje Vebpek-a (eng. Webpack) koji služi da zapakuje sve module. Zbog toga je Fejsbuk napravio modul Create React App koji služi za brzo postavljanje Riekt šablona, bez potrebe za komplikovim i dugim konfigurisanjem. Osim što generiše kostur Riekt aplikacije, Create React App pruža sve što je potrebno za SPA aplikaciju.

SPA aplikacija može imati više prikaza, odnosno stranica, a za razliku od konvencionalnih višestraničnih aplikacija (eng. multi-page applications), navigacija kroz više stranica ne rezultira ponovnim učitavanjem cele stranice. U jednostraničnim aplikacijama sav potreban kod - HTML, JavaScript i CSS - se preuzima u okviru učitavanja jedne stranice, ili se odgovarajući resursi učitavaju dinamički i po potrebi dodaju stranici, obično kao posledica neke akcije od strane korisnika. Rutiranje je proces koji sinhronizuje URL pregledača sa onim što treba da se rennderuje na stranici. Riekt Rauter omogućava deklerativno rutiranje. Deklarativan pristup omogućava kontrolu protoka podataka u aplikaciji, tako što definiše kako treba ruta da izgleda, pomoću modula react-router-dom. Postoje tri glavna tipa komponenti u Riekt Rauter Dom modulu:

- ruteri (<BrowserRouter> i <HashRouter>)
- rute podudatanja (<Route> i <Switch>)
- navigacija (<Link>, <NavLink> i <Redirect>)

Svaki put kada se komponente renderuje pomoću rutera, toj komponenti se prosleđuje objekat koji ima tri svojstva: location, match i history. Svojstvo history omogućava upravljanje istorijom sesija. Svojstvo match će sadržati informacije o ruti, da li je exact ili nije, koja je šablon putanje koji se koristi za podudaranje, deo URL-a koji se podudara, potencijalne parametre, itd.

Pomoću history svojstva možemo pratiti trenutnu lokacija (history.location) i prethodnu lokaciju.

Ridaks je nezavisna JavaSkript biblioteka koju možemo koristiti u Riekt aplikaciji kako bismo olakšali proces upravljanja stanjima. Upravljanje stanjima kao što smo do sada videli je poprilično jednostavno kad aplikacija nije velika. Međutim, čim postane velika i kompleksnija, postaje kompoleksnije i prosleđivanje stanja iz komponente A u komponentu B. Na primer, korisnik ako je prijavljen može da pristupi korpi. Međutim, početnoj stranici koja pokazuje listu proizvoda i stranicama sa detaljlima proizvoda, može pristupiti nevezano za to da li je prijvaljen ili nije. Da li je korisnik prijavljen ili ne čuvali bismo kao stanje, na primer u komponenti Authentification. Potom bismo morali da prosleđujemo to stanje do komponente ShoppingCart i do komponente ProductDetails i to bi bilo jako komplikovano i neelgantno rešenje. Zbog toga su Dan Abramov i Endru Klark došli na ideju da postoji neka globalna promenljiva u kojoj će se čuvati sva stanja aplikacije. U svakoj Riekt Ridaks aplikaciji imamo:

- Centralnu prodavnicu (eng. Central Store) u kojoj se čuvaju sva stanja aplikacije.
- Komponente koje žele da manipulišu stanjima aplikacije. S tim što se pristup globalnom objektu ne obavlja direktno, jer to ne bi radilo i centralna prodavnica bi bila neprevidiva.
- Akcije šalju komponente i predstavljaju samo informacioni paket. Ne sadrže nikakvu logiku i nemaju nikakav pristum centralnoj prodavnici.
- Redukatori su ono što zapravo pravi promene u centralnoj prodavnici. Akcije su povezane sa redukatorima, tako što Redukator proverava tip akcije koja je definisana u Akcijama. Redukatori sadrže logiku akcije čiji je tip definisan u Akcijama. Odnosno, redukator je funkcija koja dobija akciju kao ulaz i ažurira stanje i vraća ga kao rezultat i tako se menja u centralnoj prodavnici stanje. Važno je napomenuti da redukatori ne smeju biti asinhrone funkcije i da se ažuriranje stanja mora obavljati imutabilno.
- Model pretplate služi da možemo koristiti ažurirano stanje u komponentama. Kada se promeni stanje, centralna prodavnica automatski javi svim komponentama koje su se „predplatile“.

IV. ZAKLJUČAK

Cilj ovog rada bio je detaljno proučavanje različitih karakteristika svake od tehnologija MERN steka na osnovu kojih je razvijena aplikacija u domenu elektronske trgovine. Precizno su diskutovane tehnologije, od fundamentalnih do naprednih svojstava, kao i njihova upotreba u aplikaciji i prikazan svaki korak pri procesu implementiranja ovakve aplikacije. Aplikacija je u

potpunosti razvijena i funkcionalna. Istražujući i proučavajući alternative u razvoju aplikacije, stečeno je dosta korisnog znanja i pokazano je zašto je MERN stek popularan i danas igra vodeću ulogu u razvoju veba. Iako aplikacija i dalje ima nekih nedostataka i potrebna su joj dodatna poboljšanja, kako u pogledu stila, tako i u dodatnim funkcionalnostima, ona predstavlja kombinaciju jedne od najčešće korišćenih tehnologija veb steka i jedne od najnovijih poslovnih ideja u današnje vreme - e-trgovine.

MERN stek je veoma jednostavan i efikasan stek za pravljenje veb aplikacija. Međutim svaka od tehnologija MERN steka ima poneki nedostatak. MongoDB, recimo, koristi dosta memorije za skladištenje podataka, zahteva ograničenu veličinu dokumenta i ne podržava transakcije. Nod.js-ov API se veoma često menja, što ga čini poprilično nekonzistentnim. Takođe, Node.js–ov model asinhronog programiranja otežava održavanje koda. Pogodan je za komplikovane aplikacije, ali kada izvršava teška i dugotrajna računanja, znatno smanjuje performanse. Neupotreba izomorfnog pristupa za eksploataciju Riekt aplikacije dovodi do problema sa indeksiranjem pretraživača. Većina programera ne voli JSX dokumentaciju i smatraju je konfuznom. Riekt je usko fokusiran na korisnički interfejs, odnosno sadrži ogromnu kolekciju alata za kreiranje korisničkog interfejsa. Međutim, ukoliko koristimo MVC (eng. Model View Controller) arhitetkuru, Riekt će biti zadužen isključivo za prikaz aplikacije, dok se model i kontroler moraju kreirati uz pomoć dodatnih alata (npr. rutiranje). Budući da nije u potpunosti opremljen okvir, potrebno je dodatno naučiti besplatne biblioteke korisničkog interfejsa za integraciju u MVC okvir. Glavni konkurent MERN steku je MEAN stek (MongoDB Ekpres Angular Nod.js). MERN stek je idealan za manje i efikasne aplikacije, dok je MEAN stek bolja opcija za aplikacije gigantskih preduzeća. MERN stek ima iza sebe ogromnu podršku zajednice. Svaka komponenta steka je dobro dokumentovana i pruža skalabilna rešenja. Podržava MVC arhitekturu, što dosta olakšava programerima razvoj procesa rada. Sve komponente MERN steka su otvorenog koda, što znači da ne moramo da brinemo o problemima licenciranja. Takođe, postoji ogroman broj tutorijala i litertarure na engleskom jeziku za učenje MERN steka na internetu. Međutim, iako ceo stek koristi jedan programski jezik - JavaSkript, potrebno je dosta vremena za usavršavanje svake tehnologije steka. [18]

Iako aplikacija ispunjava sve zahteve i karakteristike koje su bile u planu, još uvek postoje neki aspekti potrebni za poboljšanje i nadogradnju. Neke od narednih funkcionalnosti bi se mogle razmotriti za dalji razvoj aplikacije:

- Upravljanje narudžbinama korisnika od strane admina
- Kreiranje kategorija i sortiranje proizvoda

- Notifikacija putem imejla nakon uspešne kupovine korisnika, dodavanje kontakt forme i dodavanje push notifikacija
- Dodavanje liste želja
- Mogućnost pisanja komentara, recenzije i ocenivanja proizvoda
- Kreiranje aplikacije za mobilne telefone
- Poboljšanje intuitivnosti korisničkog interfejsa

Kreiranje aplikcije za mobilne telefone je možda i najkorisnije poboljšanje. Opšte je poznato da korisnici ne vole spore veb sajtove. Ukoliko učitavanje stranica veb sajta traje više od tri sekunde, 42% korisnika će posetiti veb sajt koji je brži. Mobilne aplikacije omogućavaju prikazivanje sadržaja na brži i atraktivniji način. Čak većina mobilnih aplikacija e-trovine ni ne koristi internet konekciju za osnovne funkcionalnosti, za raliku od veb sajtova kojima je uvek potrebna internet konekcija za učitavanje i ažuriranje svakog dela sadržaja. Prema Flari Analitiks (eng. Flurry Analytics) istraživanju, 90% vremena korisnici pametnih telefona provode u interakciji sa aplikacijama. Takođe, 79% njih ima jednu ili više aplikacija e-trgovine na svojim uređajima, a 10% ima čak šest aplikacija onlajn prodavnica. Još jedna prednost mobilnih aplikacija u odnosu na veb sajtova su naprednije marketinške strategije. Na primer, Zara aplikacija nudi mušterijama drugačije iskustvo kupovine implementacijom takozvane proširene stvarnosti (eng. Augmented Reality). Preuzimanjem aplikacije i usmeravanjem kamere na manekena u Zarinoj prodavnici ili na ekranu kupovine e-trgovine, odeća će „oživeti“ na modelu za nekoliko sekundi. Na taj način se podstiču korsnici da fotografišu holograme i podele ih na društvenim mrežama. Takođe, Asos ima inovativnu marketinšku ideju dovodeći „shop the look“ na viši nivo. U njihovoj mobilnoj aplikaciji za Android i IOS postoji malo dugme za kameru. Kada se pritisne, možete dodati fotografiju odeće koja vam se sviđa, a Asos će se pobrinuti za pronalaženje sličnih na njihovom veb sajtu. Pored toga, primenom strategije svetionika (eng. beacons) moguće je čak i otkriti kada je uređaj u blizini određene prodavnice. Na osnovu lokacije korisnika, aplikacija može da šalje personalizovana push obaveštenja, koje sadrže posebne ponude koje su dostupne samo u prodavnici u blizini. [19] Već je pomenuto da postoji i Riekt Nejtiv, čitava platforma koja omogućava kreiranje nativnih i višeplatformskih mobilnih aplikacija. RiektJS je srž Riekt Nejtiv-a i oličava sve Riekt principe i sintaksu, tako da je učenje Riekt Nejtiv-a sa osnovom znanja RiektJS-a veoma lako. To je ujedno i jedan od razloga što sam za ovaj rad izbrala Riekt tehonologiju. Sledeći korak bi možda bio baš taj, kreiranje mobilne aplikacije za Glosier prodavnicu.

V. LITERATURA

[1] Elektronska trgovina: teorija i praksa. Margarita Išoraitė, Neringa Miniotienė. Jun 2018. URL:https://www.researchgate.net/publication/329704574_Electronic_Commerce_Theory_and_Practice.

[2] E-trgovina: kritički osvrt. Jonathan Reynolds. Novembar 2000. URL: https://www.researchgate.net/publication/247629187_E-commerce_A_critical_review.

[3] E-trgovina. Wikipedia. URL:https://en.wikipedia.org/wiki/E-commerce.

[4] Kompletna knjiga o e-trgovini: Dizajn, izgradnja i održavanje uspešnog poslovanja zasnovanog na Internetu, Drugo Izdanje. Janice Reynolds. CMP Books © 2004.

[5] Nod.js. Wikipedia. URL: https://en.wikipedia.org/wiki/Node.js.

[6] Nod.js - NPM. Tutiralspoint.URL:https://www.tutorialspoint.com/nodejs/nodejs_npm.htm.

[7] MongoDB zvanična dokumentacija. URL: https://docs.mongodb.com/manual/.

[8] Šta je MongoDB?. Decode Web. 28 Jun 2019. URL: https://medium.com/@decodeweb/what-is-mongodb-7693e2f2f4f6.

[9] Riekt – Kompletan vodič (uključuje Huks, Riekt Rauter, Ridaks). Academind by Maximilian Schwarzmüller, Udemy. URL: https://www.udemy.com/course/react-the-complete-guide-incl-redux/.

[10] Šta je i zašto Riekt.js?. Nitin Pandit. 5 Mart 2020. URL: https://www.c-sharpcorner.com/article/what-and-why-reactjs/.*Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.

[11] 7 разлога зашто би требало да користите Риект. Kutlu Sahin. 27 Јануар 2017. УРЈІ: https://stories.jotform.com/7-reasons-why-you-should-use-react-ad420c634247.

[12] Riekt zvanična dokumentacija. URL: https://reactjs.org/.

[13] Metode životnog ciklusa Riekt-a. Mosh Hamedani.19 Novembar 2018. URL: https://programmingwithmosh.com/javascript/react-lifecycle-methods/.

[14] Mongus zvanična dokumentacija. URL: https://mongoosejs.com/docs/guide.html.

[15] Uvod u Mongus za MongoDB. Nick Karnik. 11 Februar 2018. URL: https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/.

[16] Džoi zvanična dokumentacija. URL: https://joi.dev/api/?v=17.3.0.

[17] Kako koristiti Džoi za Nod API Validacionu Shemu. Glad Chinda. 23 Septembar 2020. URL:https://www.digitalocean.com/community/tutorials/how-to-use-joi-for-node-api-schema-validation.

[18] Zašto je MERN stek naša preferirana platforma za start-ap aplikacije?. Kanika Gupta. 1 Septembar 2020. URL: https://www.classicinformatics.com/blog/why-is-mern-stack-our-preferred-platform-for-startups-apps.

[19] Aplikacija e-trgovine: Zašto i na koji način kreirati je za vašu onlajn prodavnicu. Yeeply. URL: https://en.yeeply.com/blog/e-commerce-app-how-to-create-one/.

[20] Razumevanje virtualnog DOM-a. Ire Aderinokun. 24 Decembar 2018. URL: https://bitsofco.de/understanding-the-virtual-dom/.

ABSTRACT

The rapid development of web technologies has taken web development to a new level, which facilitates the work of programmers. One of the most commonly used stacks is the MERN stack, which today plays a leading role in web development. Four components included in the MERN stack are MongoDB database, Express backend web framework, ReactJS frontend library, and Node.js JavaScript environment.

This paper focuses on development of web store application of the well-known cosmetic brand Glossier using JavaScript MERN stack technology. The paper gradually explains the key concepts of the mentioned technologies, together with Redux, React Router, JWT, Mongoose and other libraries. The process of creating an application is described from the moment the idea is created, to the final stage of development and implementation.

As a result, a fully functional application was obtained that complied with the previously set requirements. It has a simple and intuitive user interface. It also includes an admin panel that serves admin to manipulate (create, delete and update) the products. This paper can be used as a tutorial for developing a full stack MERN web applications.

APPLICATION OF REACTJS TECHNOLOGY IN THE DEVELOPMENT OF E-COMMERCE APPLICATIONS

Nataša Vuksanović dipl. ing., prof. dr Dušan Vujošević, dipl. ing.