Razvoj jednostranične mrežne aplikacije s naglaskom na poboljšanje korisničkog iskustva

Ćulap, Katarina

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet Osijek

Dvopredmetni diplomski studij Informatologije i Informacijske tehnologije

Katarina Ćulap

Razvoj jednostranične mrežne aplikacije s naglaskom na poboljšanje korisničkog iskustva

Mentor: izv. prof. dr. sc. Tomislav Jakopec

Osijek, 2024.

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet Osijek

Odsjek za informacijske znanosti

Dvopredmetni diplomski studij Informatologije i Informacijske tehnologije

Katarina Ćulap

Razvoj jednostranične mrežne aplikacije s naglaskom na poboljšanje korisničkog iskustva

Diplomski rad

Društvene znanosti, Informacijske i komunikacijske znanosti, Informacijski sustavi i informatologija

Mentor: izv. prof. dr. sc. Tomislav Jakopec

Osijek, 2024.

Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnoga, odnosno diplomskog rada.

IZJAVA

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napisao/napisala te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s navođenjem izvora odakle su preneseni.

Svojim vlastoručnim potpisom potvrđujem da sam suglasan/suglasna da Filozofski fakultet u Osijeku trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta u Osijeku, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, 05.09.2024.

[redacted] 0122229017

Ime i prezime studenta, JMBAG

Sažetak

Jednostranična aplikacija (SPA) vrlo je popularan oblik aplikacija koji, kao što sam termin implicira, koristi samo jednu početnu stranicu koja dinamički mijenja podatke na zahtjev korisnika, umjesto učitavanja potpuno nove stranice kao što je to slučaj s višestraničnim aplikacijama (MPA). Takav pristup korisnicima omogućuje kretanje aplikacijom bez prekida, što znatno unapređuje korisničko iskustvo, na što se sve više stavlja naglasak u modernom mrežnom razvoju. Primjena jednostraničnih aplikacija je široko rasprostranjena, a koriste ih i neke popularne aplikacije kao što su Gmail, Airbnb, Netflix, PayPal i dr. Razvoj jednostraničnih aplikacija pojednostavljen je razvojem radnih okvira te su neki od najpoznatijih, kao što su Angular, React i Vue, opisani u radu. Također rad prikazuje temeljne razlike jednostraničnih i višestraničnih aplikacija te kroz opis prednosti i nedostataka opisuje funkcionalnosti ovakvog oblika aplikacije. Osim teorijskog dijela, cilj rada je prikazati na stvarnim primjerima osnovne koncepte koje jednostranična aplikacija koristi. Za razvoj aplikacije korišten je Vue radni okvir, a aplikacija je namijenjena s jedne strane korisnicima knjižnice kako bi rezervirali svoja mjesta te s druge strane zaposlenicima knjižnice odnosno administratorima, kako bi upravljali zahtjevima korisnika i imali potpuni uvid u zauzetost knjižnice.

Ključne riječi: jednostranična aplikacija, SPA, MPA, vue.js, korisničko iskustvo

Sadržaj

1. Uvod

Razvoj interneta i tehnologije svakodnevno napreduje te se samim time izgled i razvoj mrežnih aplikacija značajno promijenio u odnosu na prošlost. No, može se reći da među svim promjenama koje mrežne aplikacije doživljavaju već dugi niz godina, njezin osnovni cilj ostaje isti, a to je omogućiti jednostavnu i poboljšanu interakciju s korisnikom. U početku su prevladavale višestranične aplikacije (*eng. multi-page applications, MPA*) koje su se oslanjale na tradicionalni model učitavanja novih stranica i resursa koje ta stranica koristi prilikom svake interakcije s korisnikom. Međutim, pojava jednostraničnih aplikacija (*eng. single-page applications, SPA*) označila je značajan iskorak u evoluciji mrežnih tehnologija, postavljajući novi standard u dinamičnosti, brzini i interaktivnosti korisničkog iskustva. Jednostranične aplikacije su oblik mrežnih aplikacija koje, za razliku od višestraničnih aplikacija, učitavaju samo jednu HTML (*HyperText Markup Language)* stranicu, a zatim dinamički pomoću JavaScript programskog jezika mijenjaju sadržaj na istoj stranici bez potrebe za ponovnim osvježavanjem. Ovaj pristup omogućava korisnicima brže učitavanje sadržaja te poboljšano korisničko iskustvo, ali i smanjenje opterećenja na poslužitelju. U modernom vremenu, kada postoji jako velik broj aplikacija na internetu, korisnici imaju sve veća očekivanja od aplikacija i mogućnosti koje one pružaju. S obzirom na to da se u posljednjim desetljećima sve veći naglasak stavlja na korisničko iskustvo, ono postaje važan vodič programerima prilikom izrade i dizajniranja aplikacija. Iako je tradicionalni model višestraničnih aplikacija i dalje široko rasprostranjen te se ovaj oblik još uvijek koristi za razvoj kvalitetnih aplikacija, posebice za mrežne trgovine, sve je veći naglasak na razvoju jednostraničnih aplikacija koje, zbog svoje brzine i lakoće, postaju trend među programerima. Razvoj jednostraničnih aplikacija postao je izrazito značajan pojavom modernih JavaScript radnih okvira (*eng. framework*) kao što su Angular, React i Vue. Ovi okviri sadrže brojne funkcionalnosti koje programerima omogućuju jednostavniji i brži razvoj jer sadrže velik broj paketa i alata za izgradnju aplikacija s bogatim svojstvima. S obzirom na široku rasprostranjenost ovakvog tipa aplikacija, ovim radom nastoji se istražiti na koji način je strukturirana jednostranična aplikacija, koje funkcionalnosti pruža te je li to najbolji izbor prilikom razvoja mrežnih aplikacija s naglaskom na poboljšano korisničko iskustvo. U prvom poglavlju detaljno će se analizirati povijesni razvoj jednostraničnih aplikacija, od samih početaka i nastanka interneta do nastanka prvih radnih okvira i jednostraničnih aplikacija. Nadalje, proučit će se temeljne razlike između višestraničnih i jednostraničnih aplikacija te će se objasniti njihovi

osnovni standardi i arhitekture kako bi se na koncu mogao donijeti zaključak je li jednostranična aplikacija uvijek dobar izbor u mrežnom razvoju. Jednostranične aplikacije imaju brojne prednosti pa će se u daljnjim poglavljima detaljnije proučiti spomenute i dodatne prednosti, ali i eventualni nedostaci koje bi ovakav oblik aplikacije mogao imati. Također, opisat će se radni okviri: Angular, React i Vue te će se istražiti postoji li idealno rješenje pri njihovom korištenju u jednostraničnim aplikacijama. Na samom kraju rada, kako bi se prikazala primjena jednostraničnih aplikacija u stvarnom okruženju, izraditi će se jednostavna aplikacija s jednostraničnom arhitekturom koristeći Vue radni okvir. Praktični dio rada omogućit će bolje razumijevanje strukture jednostranične aplikacije, ali i potencijalnih izazova s kojima se susreću programeri tijekom razvoja takvih aplikacija. Rad istražuje koncept jednostraničnih aplikacija s ciljem razumijevanja njezinih značajki, prednosti i izazova te njihove uloge u kontekstu modernog mrežnog razvoja i korisničkog iskustva.

2. Povijesni pregled i nastanak jednostraničnih aplikacija

Prilikom izrade mrežne aplikacije prva faza koju je potrebno provesti zasigurno je odabrati kojoj kategoriji će pripadati sama aplikacija. Iako je jednostranična aplikacija već dugi niz godina dio mrežnog razvoja, posljednjih nekoliko godina sve se više stavlja naglasak na razvoj aplikacija ovakvog tipa. Može se reći da je ovaj trend nastao kao posljedica korisnika koji očekuju aplikacije koje će raditi brzo i učinkovito.[1] No, prije nego što se definira koncept jednostranične aplikacije potrebno je objasniti kratku povijest njenog nastanka. Povijest jednostranične aplikacije započela je 1990-ih godina kada je Tim Berners-Lee implementirao svjetsku mrežu poznatiju kao WWW (*eng. World Wide Web).*[2] U početku su postojale samo višestranične aplikacije koje su u osnovi bile izrađene u statičkom HTML-u te su kasnije, kako bi se omogućila dinamičnost mrežnih stranica i razvoj sofisticiranijih mrežnih aplikacija, razvijeni i skriptni jezici na strani poslužitelja kao što su PHP (*Hypertext Preprocessor)* ili ASP (*Active Server Pages*).[3] Iako je razvoj skriptnih jezika omogućio određeni napredak, svaka operacija rezultirala je ponovnim slanjem podataka na poslužitelj i osvježivanjem (*eng. refresh)* čitave stranice te je posljedično tome korisničko iskustvo bilo izuzetno loše.[4] Razvojem JavaScript programskog jezika 1995. godine, tada jedinog skriptnog jezika na strani klijenta, omogućene su brojne značajke te razvoj dinamičnijih mrežnih aplikacija.[5] No, iako je zbog svojih mogućnosti vrlo brzo postao popularan, stanje se nije znatno poboljšalo s obzirom da se ovaj programski jezik u tom obliku nije smatrao dovoljno programskim i sofisticiranim za izradu zahtjevnije mrežne stranice ili aplikacije.[6] Godinu kasnije, 1996. FutureWave razvija softversku platformu pod izvornim nazivom FutureSplash Animator, kasnije poznatiju kao Adobe Flash. Ovaj softver ponudio je brojna poboljšanja u smislu korisničkog iskustva jer je pružao velik broj svojstava za stvaranje interaktivnog digitalnog sadržaja kao što su: animacije, grafički efekti, video i audio elementi itd. Zbog svojih inovativnih značajki, Adobe Flash je niz godina bio standardna platforma za programere koji su koristili njegove značajke za

[1] Usp. Fink, Gil; Flatow, Ido. Pro Single Page Application Development: Using Backbone.js and ASP.NET. // Apress Berkeley, CA, 2014., str. 13 URL: https://link.springer.com/book/10.1007/978-1-4302-6674-7#bibliographic-information (2023-07-03)
[2] Usp. Isto, str. 3.
[3] Usp. Isto.
[4] Usp. Isto.
[5] Usp. History of JavaScript // GeeksforGeeks, 2024. URL: https://www.geeksforgeeks.org/history-of-javascript/ (2024-07-05)
[6] Usp. Fink, Gil; Flatow, Ido. Nav. dj.

stvaranje bogatih mrežnih aplikacija, ali i pozadinskih i mobilnih aplikacija, igara i sl.[7] Iako je Adobe Flash 2020-ih godina doživio kraj svog životnog vijeka, njegov razvoj obilježio je jednu eru u razvoju mrežnih aplikacija s naglaskom na poboljšanje korisničkog iskustva. Nekoliko godina kasnije osim Adobe Flash platforme ponudile su se i druge opcije kao što su primjerice kada Microsoft predstavlja dodatak Silverlight koji bi pregledniku omogućio prikaz tzv. bogatih internetskih aplikacija (*eng. Rich Internet Application,* RIA) ili na način da se koristi *iframe* HTML element gdje bi se tada ažurirala samo stranica na *iframe* površini. Navedeni prijedlozi i rješenja pružali su samo trenutna poboljšanja i mogućnosti, no s druge strane su se pojavljivali novi izazovi kao što su primjerice problemi s održavanjem stranice i sl. Prekretnica koja je svakako izazvala napredak i pokazala kako jednostranična aplikacija izgleda najbliže današnjoj verziji započela je oko 2005. godine koju je obilježila pojava i početak korištenja tzv. AJAX-a (*Asynchronous JavaScript And XML (Extensible Markup Language))* kao standardom za izradu mrežnih aplikacija.[8] Ovaj standard ne predstavlja nikakvu novu tehnologiju, nego kombinaciju nekoliko postojećih tehnologija koji uspostavljaju komunikaciju između klijenta i poslužitelja na znatno brži i jednostavniji način. Pomoću AJAX-a, mrežna stranica i svi podaci se učitavaju samo jednom prilikom početnog zahtjeva korisnika, dok svakim novim zahtjevom od strane korisnika poslužitelj asinkrono vraća sadržaj bez učitavanja nove stranice ili ometanja trenutnog prikaza mrežne aplikacije.[9] Na taj način ažuriraju se samo dijelovi koji se nalaze na stranici, ali ne i sama stranica što uvelike poboljšava korisničko iskustvo jer se promjene odvijaju puno kraće. Prije AJAX-a svaka izmjena na stranici rezultirala je ponovnim učitavanjem cijele mrežne stranice s poslužitelja što je osim čekanja od strane korisnika također rezultiralo i dupliciranjem podataka u svakoj instanci.[10] Ponudivši brojne mogućnosti, može se reći da je ovaj standard obilježio novu eru u razvoju mrežnih aplikacija, a među prvim tvrtkama koja je iskoristila ovakav pristup u razvoju aplikacija bila je tvrtka Google. Pomoću AJAX-a Google je razvio i danas korištene aplikacije: Google karte i Gmail te samim time tvrtka je ostvarila veliku popularnost i utjecala na sve veći

[7] Usp. Awati, Rahul. Adobe Flash // TechTarget, 2022. URL: https://www.techtarget.com/whatis/definition/Flash (2024-07-06)
[8] Usp. Isto, str. 4
[9] Usp. Kanade, Vijay. What Is AJAX (Asynchronous JavaScript and XML)? Meaning, Working and Applications: AJAX enhances a website's performance and overall user experience. // Spiceworks, 2023. URL: https://www.spiceworks.com/tech/devops/articles/what-is-ajax/ (2023-07-03)
[10] Usp. Isto.

razvoj aplikacija koristeći ovaj model.[11] Takve aplikacije koje su ubrzale, ali i pojednostavile korištenje mrežnih stranica i aplikacija najbliže opisuju kako izgleda model današnjih jednostraničnih aplikacija. Put do razvoja jednostraničnih aplikacija zabilježilo je i poboljšanje JavaScript jezika na koje je utjecalo stvaranje tzv. JavaScript biblioteka jQuery i DOM (*eng. skraćeno Document Object Model)* koje su ponudile nove mogućnosti i osigurale zapisivanje AJAX operacija u manje linija koda.[12] Osim biblioteka, na razvoj JavaScript-a utjecao je i HTML5, odnosno nova verzija HTML-a koja je ponudila brojne nove elemente te između ostaloga i mogućnost korištenja API-a (*eng. skraćeno Application Programming Interface)*.[13] Kao posljednja veća prekretnica koja se događa u povijesnom pregledu nastanka jednostraničnih aplikacija općenito je početak korištenja mobilnih uređaja u svakodnevnom životu koji zahtijevaju razvijenije vještine prilikom razvoja aplikacija, a poboljšanja i razvoj novih tehnologija omogućila su korištenje JavaScript-a kao platforme za izradu takvih aplikacija. Iako je evolucija JavaScript-a znatno utjecala na ideju stvaranja jednostranične aplikacije, tada je put do ideje za taj model još uvijek bio dalek, a prva naznaka jednostraničnih aplikacija dogodila se tek 2009. godine razvojem *Backbone.js* izdanja koji je ponudio okvir na strani klijenta uz pomoć kojega je moguće izgraditi jednostraničnu aplikaciju, no razvoj takve aplikacije još uvijek je bio kompliciran te je zahtijevao puno znanja i posla.[14] Spajanjem svih ideja, 2010. godine među prvima koji je pokrenuo trend i ponudio prvo pravo rješenje za razvoj jednostranične aplikacije bio je *Angular.js,* okvir koji se i danas koristi za razvoj aplikacija ovakvog tipa.[15] Danas su jednostranične aplikacije vrlo popularan model u razvoju aplikacija za različite svrhe, a uz navedene aplikacije Google tvrtke, još neke od popularnih aplikacija zasnovanih na ovakvom modelu su primjerice Facebook (Meta), Twitter, Netflix, Pinterest, Paypal i sl.[16] No, dolazi se do glavnog pitanja - koja je definicija i što zapravo označavaju jednostranične aplikacije? Ne postoji jedinstvena definicija koja opisuje što je jednostranična aplikacija, ali kao što i sam naziv termina govori, to je, ukratko objašnjeno, mrežna

[11] Usp. Fink, Gil; Flatow, Ido. Nav. dj. str. 4
[12] Usp. Isto
[13] Usp. Isto, str. 4-5
[14] Usp. Jaman, Shifat. Everything you need to know about "Single-page-application"// Medium, 2019. URL: https://shifat-jaman.medium.com/single-page-application-everything-you-need-to-know-6f00d87e5130 (2023-07-03)
[15] Usp. Isto.
[16] Usp. Acceptance of Single Page Application in Today's World // Great Learning, 2022. URL:https://www.mygreatlearning.com/blog/acceptance-of-single-page-application-in-todays-world/ (2023-07-03)

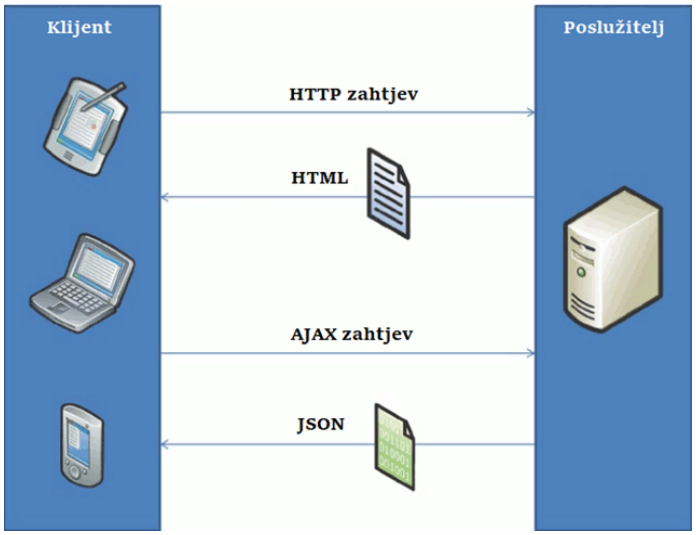
aplikacija ili mrežna stranica koja koristi odnosno učitava samo jednu stranicu te prilikom korisnikove interakcije sa stranicom, na njegov zahtjev se prepisuje novi sadržaj na istu stranicu umjesto da se učitava potpuno nova stranica kao što je slučaj na višestraničnim aplikacijama.[17] Iako su jednostranične aplikacije široko rasprostranjene, važno je napomenuti kako su isto tako i tradicionalne odnosno višestranične aplikacije također još uvijek u širokoj upotrebi.[18] Kako bi se u potpunosti razumio koncept jednostraničnih aplikacija u nastavku će se usporediti jednostranične i višestranične aplikacije i izdvojiti njihove temeljne razlike.

3. Usporedba višestraničnih i jednostraničnih mrežnih aplikacija

Proučavajući povijesni pregled mrežnih aplikacija najprije su postojale samo višestranične mrežne aplikacije, no razvojem JavaScript-a i AJAX-a došlo je do inovacije koja je ponudila novo rješenje te su danas jednostranične i višestranične aplikacije dva osnovna modela koja se mogu koristiti u suvremenom razvoju mrežnih aplikacija.[19] Oba modela imaju svoje prednosti i ograničenja te je prilikom odabira jednog od modela potrebno imati na umu osnovnu namjenu same aplikacije te pretpostaviti zahtjeve korisnika, a u ovom poglavlju usredotočit će se na glavne razlike između navedenih modela aplikacija. Prilikom usporedbe višestraničnih i jednostraničnih aplikacija, osnovna razlika se može razumjeti već iz samog naziva termina. Prema tome, u modelu višestraničnih aplikacija može se pronaći više stranica odnosno svaka stranica obično predstavlja posebnu HTML datoteku, dok jednostranična aplikacija sadrži samo jednu HTML datoteku.[20] S obzirom da višestranična aplikacija sadrži više stranica tj. HTML datoteka, to uključuje i više zahtjeva za novom stranicom prilikom interakcije od strane korisnika. Primjerice, kada korisnik klikom miša pokrene određenu akciju na aplikaciji, njegov zahtjev se šalje na poslužitelj te se preuzimanjem nove HTML datoteke od strane poslužitelja odgovor vraća i prikazuje korisniku na isti način sa svakim novim zahtjevom.[21] Slika 1 prikazuje životni ciklus tradicionalne odnosno

[17] Usp. BasuMallick, Chiradeep. What Is a Single-Page Application? Architecture, Benefits, and Challenges // Spiceworks, 2022. URL: https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/ (2023-07-03)
[18] Usp. Isto.
[19] Usp. Ćwik, Przemyslav. Single-page application: how SPA works and how it differs from MPA. // Kiss Digital, 2021. URL: https://kissdigital.com/blog/single-page-application-how-spa-works-and-how-it-differs-from-mpa#:~:text=The%20first%20single-page%20applications,used%20to%20create%20the%20frontend (2023-07-27)
[20] Usp. Isto.
[21] Usp. Fink, Gil; Flatow, Ido. Nav. dj. str. 6-7

višestranične aplikacije koji započinje nakon što se poslužitelju pošalje zahtjev za stranicom odnosno aplikacijom, a odgovor se vraća kao HTML. U nastavku životnog ciklusa višestranične aplikacije, korisnik može poslati novi zahtjev, a rezultati se ponovo šalju kao nova HTML datoteka. S obzirom da do klijenta dolazi nova HTML datoteka, to znači da se mrežna stranica u potpunosti osvježava.[22]



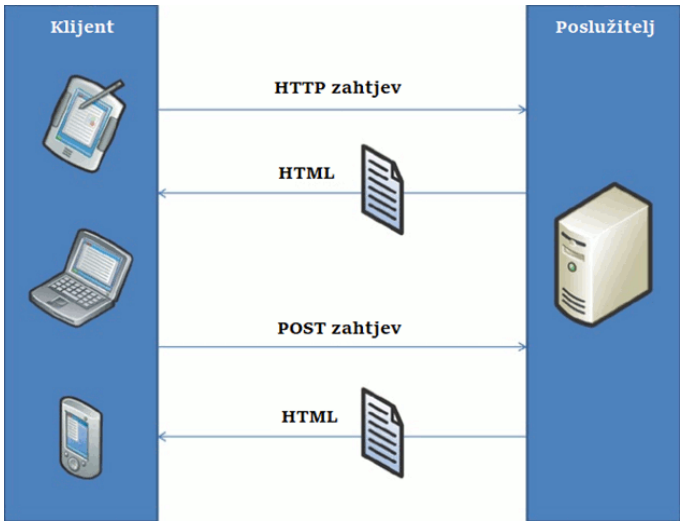
Slika 1. Prikaz životnog ciklusa višestranične aplikacije[23]

U tradicionalnom modelu većina logike odvija se na poslužitelju, dok se na strani klijenta prikazuju mrežne stranice odnosno HTML datoteke koje prima.[24] Iako se ovakav model čini logičan, u suvremenom razdoblju tehnologija je zaista napredovala te bilo kakvo kašnjenje sustava ili čekanje od strane korisnika može izazvati negativnu reakciju i loše korisničko iskustvo s aplikacijom. Dok korisnik komunicira s mrežnom aplikacijom, rukovanje događajima i akcijama na poslužitelju može biti dugotrajno te samim time dok korisnik čeka rezultat mrežna aplikacija neće reagirati, štoviše ukoliko internetska veza ne postoji ili nije stabilna aplikacija neće raditi.[25]

S druge strane, jednostranična aplikacija se učitava kao jedna HTML datoteka odnosno jedna URL (*Uniform Resource Locator)* veza gdje se na zahtjev korisnika mijenjaju samo komponente korisničkog sučelja te se revidira trenutna stranica bez ponovnog učitavanja cijele

[22] Usp. Isto.
[23] Usp. Isto.
[24] Usp. Isto.
[25] Usp. Isto.

stranice ili nekoliko stranica s poslužitelja.[26] Nakon objašnjenog životnog ciklusa tradicionalne aplikacije, u nastavku na Slici 2 prikazan je životni ciklus jednostranične aplikacije.



Slika 2. Prikaz životnog ciklusa jednostranične aplikacije[27]

Ukoliko se usporede Slika 1 i Slika 2 moguće je vidjeti temeljne razlike ove dvije vrste aplikacija. Razlika u životnom ciklusu jednostranične aplikacije u odnosu na tradicionalnu započinje nakon početnog HTTP zahtjeva i vraćanja HTML-a klijentu. Jednostranične aplikacije najčešće koriste AJAX za traženje podataka te se odgovor šalje klijentu u formatu JSON (*eng. skraćeno JavaScript Object Notation)*.[28] Nakon što klijent primi podatke, prikazati će sve promjene u HTML-u. Za razliku od višestraničnih aplikacija, sve izmjene u jednostraničnim aplikacijama događaju se na strani klijenta te u tom slučaju korisnik ne napušta, niti osvježava mrežnu stranicu učitanu početnim zahtjevom. Samim time korisnik će zatvaranjem preglednika ili gubitkom mreže, moći zadržati i vratiti zadnje stanje aplikacije što samu aplikaciju čini praktičnom za korisnike jer primljene podatke mogu koristiti i u izvanmrežnom radu ako je potrebno.[29] U teorijskom smislu riječi, jednostranične aplikacije imaju samo jednu stranicu koja predstavlja jezgru za sve ostale stranice aplikacije. S obzirom da nema potpunih slanja *postova* na poslužitelj i osvježavanja mrežne stranice, HTML elementi i sadržaj se iz jednog prikaza u drugi mijenjaju pomoću *frontend*

[26] Usp. BasuMallick, Chiradeep. Nav.dj.
[27] Usp. Fink, Gil; Flatow, Ido. Nav. dj.
[28] Usp. Isto. str. 11-12
[29] Usp. Isto.

usmjeravanja i mehanizama.[30] Stoga, gledajući iz kuta korisnika aplikacije, budući da se kod jednostraničnih aplikacija ažuriraju samo određeni dijelovi stranice, može se prihvatiti činjenica da korisnik doživljava manje prekida u radu pa je samim time korištenje aplikacije brže i jednostavnije. Ukoliko se karakteristike jednostranične aplikacije promotre bolje, može se primjetiti kako ona nalikuje izvanmrežnim aplikacijama ili bilo kojoj drugoj aplikaciji instaliranoj na računalu koje u trenutku odgovaraju na korisnikove naredbe.[31] U modernom razvoju mrežnih tehnologija sve se više razvijaju ovakvi tipovi aplikacija koje su poznatije kao progresivne mrežne aplikacije (*eng. Progressive web application, PWA*). Progresivne mrežne aplikacije imaju nekoliko prednosti, a kao osnovna značajka koja se može izdvojiti je da aplikacija radi izvanmrežno te je na taj način korisnicima uvijek dostupna pa su samim time i prekidi prilikom korištenja aplikacije znatno manji.[32] Jednostranične aplikacije i progresivne aplikacije u nekim aspektima imaju slične značajke te se najčešće nakon razvoja jednostraničnih aplikacija one distribuiraju u progresivne.[33] Unatoč tome i jednostranične aplikacije mogu imati nedostatke, prema tome nije dokazano da su takve aplikacije bolje rješenje od višestraničnih aplikacija.[34] Osim toga, u nekim slučajevima primjerice kod razvoja mrežnih trgovina i sličnih aplikacija koje nude širok raspon usluga, neizbježno je koristiti model višestraničnih aplikacija jer olakšavaju povezivanje s nekoliko korisničkih baza koje su potrebne za rad aplikacije pa su stoga ovakve aplikacije još uvijek široko rasprostranjene ili se koriste u kombinaciji s jednostraničnim modelom aplikacija kao hibridno rješenje.[35] Osim arhitekture i brzine, kao još jedna razlika koja se može istaknuti između ova dva modela je razvojni proces same aplikacije. S obzirom da se u višestraničnim aplikacijama koristi više HTML datoteka koje se kasnije učitavaju na zahtjev korisnika, nije potrebna stručnost u JavaScript-u, kao što je slučaj kod jednostranične aplikacije. Dakle, ukoliko se izrađuje jednostranična aplikacija, ona zahtijeva relativno više vremena i znanja

[30] Usp. Isto.
[31] Usp. Ćwik, Przemyslav. Nav. dj.
[32] Usp. Wieckowska, Sara; Bracisiewicz, Mateusz. PWA vs. MPA vs. SPA – What's the Best Choice for Your App? // Neoteric, 2022. URL: https://neoteric.eu/blog/pwa-vs-mpa-vs-spa-whats-the-best-choice-for-your-app/ (2023-07-27)
[33] Usp. Isto.
[34] Usp. Isto.
[35] Usp. Isto.

u JavaScript-u kako bi se omogućilo prikazivanje prednjeg i stražnjeg dijela aplikacije svakom akcijom korisnika.[36]

4. Arhitektura i radna okruženja jednostraničnih aplikacija

U modernom mrežnom razvoju postoje tri najčešća načina na koji se HTML, CSS i JavaScript kod obrađuje i prikazuje na mrežnoj aplikaciji *(eng. rendering, dalje u tekstu renderiranje)*. To su: renderiranje na strani poslužitelja (*eng. Server-side rendering, SSR),* renderiranje na strani klijenta *(eng. Client-side rendering, CSR)* te kao generator statičkih stranica (*eng. Static site generation).*[37] Odabir jednog od navedenih modela ovisi o specifičnim potrebama i zahtjevima projekta koji se želi izraditi te svaki od ovih modela ima određene prednosti i nedostatke. Prvi model, renderiranje na strani poslužitelja je proces izrade mrežnih stranica ili aplikacija gdje se HTML sadržaj generira na poslužitelju, a zatim se šalje klijentu, odnosno pregledniku korisnika na temelju traženog URL-a podataka. Generiranjem HTML-a na strani poslužitelja, smanjuje se količina JavaScript koda koji se mora učitati i izvršiti u pregledniku klijenta što rezultira znatno bržim početnim vremenom učitavanja, a samim time i boljom izvedbom na sporijim mrežama.[38] Osim navedenih prednosti, ovaj pristup omogućuje poboljšanu optimizaciju za tražilice (*eng. Search engine optimization, SEO)* što za neke aplikacije može biti bitan uvjet jer se na taj način osigurava bolja vidljivost i rangiranje u rezultatima tražilice. Međutim, ovaj oblik renderiranja sadrži i određena ograničenja pa u usporedbi s renderiranjem na strani klijenta, zahtijeva više obrade na strani poslužitelja što rezultira smanjenom dinamičnosti jer interakcije zahtijevaju dodatne zahtjeve s poslužitelja te su samim time i troškovi poslužitelja veći.[39] S druge strane, renderiranje na strani klijenta sve potrebne HTML, CSS i JavaScript datoteke učitava odjednom kada korisnik prvi put stupa u interakciju s aplikacijom što može produžiti početno vrijeme učitavanja, no sve daljnje interakcije su puno brže jer nema potrebe za ponovnim učitavanjem cijele stranice.[40] Ovakva tehnika zapravo opisuje arhitekturu jednostranične aplikacije te se često pojam renderiranje na strani klijenta i jednostranične aplikacije gledaju kao isti pojam, no zapravo je jednostranična aplikacija vrsta mrežne aplikacije koja uključuje ovu arhitekturu kao svoje ponašanje, odnosno renderiranje na

[36] Usp. BasuMallick, Chiradeep. Nav. dj.
[37] Usp. Cocca, Germán. Rendering Patterns for Web Apps – Server-Side, Client-Side, and SSG Explained // freeCodeCamp, 2023. URL: https://www.freecodecamp.org/news/rendering-patterns/ (2024-07-05)
[38] Usp. Isto.
[39] Usp. Isto.
[40] Usp. Isto.

strani klijenta je model prema kojemu jednostranične aplikacije djeluju.[41] Arhitekturom koje pruža renderiranje na strani klijenta, jednostranične aplikacije pružaju brzo i interaktivno korisničko iskustvo, ali zahtijevaju dodatna razmatranja za optimizaciju tražilice čime se dolazi do tzv. koncepta hidratacije (*eng. The Concept of Hydration)*. U mrežnom razvoju pojam hidratacije se odnosi se na proces u kojemu preglednik obrađuje HTML dokument koji je generirao poslužitelj te nadograđuje taj sadržaj izvršavanjem JavaScript koda kako bi se omogućilo dinamičko ponašanje stranice.[42] Ukratko, ovaj proces osigurava podudaranje početnog stanja stranice s HTML-om generiranim na poslužitelju te omogućava besprijekoran prijelaz na interaktivni prikaz na strani klijenta. Posljednji model je generiranje statičkih stranica koji nalikuje modelu renderiranje na strani poslužitelja, no u ovom slučaju sve potrebne datoteke se generiraju unaprijed tijekom procesa izgradnje stranice, a ne u stvarnom vremenu na svaki zahtjev korisnika. Ovakav model pruža brže vrijeme učitavanja i bolju sigurnost, no sadrži ograničenja u smislu sadržaja koje takve aplikacije nude te se najčešće koriste u slučajevima kada se izrađuju aplikacije koje ne zahtijevaju dinamičan i složen sadržaj kao što je prijava i druge slične radnje.[43]

Pored različitih arhitektura, programeri se prilikom izrade jednostraničnih aplikacija oslanjaju na različite radne okvire koji olakšavaju i ubrzavaju razvoj aplikacije. U nastavku će se opisati tri primjera radnih okvira za koja se mogu reći da su među najpopularnijim primjerima Javascript okvira za izradu jednostraničnih aplikacija, a to su: Angular, React i Vue. Angular, kao spomenuti okvir koji je među prvima ponudio pravo rješenje za izradu jednostraničnih aplikacija dobar je izbor za izgradnju složenih mrežnih aplikacija koje zahtjevaju optimizaciju tražilice. Ovaj okvir kojeg je razvila tvrtka Google proširuje HTML kod uvođenjem novih atributa i elemenata te na taj način olakšava izradu jednostraničnih aplikacija. Ovaj okvir ima velik broj dostupnih resursa te je podržan od strane zajednice zbog svoje modularnosti te lakoće održavanja i testiranja aplikacije. No, također ovaj okvir u usporedbi s nekim drugim okvirima zahtjeva više vremena i truda za svladavanje te može imati veću veličinu paketa, a u određenim situacijama može ograničiti

[41] Usp. Demystifying Rendering Modes - SPA/CSR, SSR, SSG, OMG // Wolfpack digital, 2022. URL: https://www.wolfpack-digital.com/blogposts/spa-csr-ssr-ssg-demystifying-rendering-modes (2024-07-05)
[42] Usp. Cocca, Germán. Nav. dj.
[43] Usp. Demystifying Rendering Modes - SPA/CSR, SSR, SSG, OMG. Nav. dj.

fleksibilnost jednostraničnih aplikacija.[44] Neki od poznatih primjera aplikacija koji su razvijeni uz pomoć ovog okvira su: Gmail, Paypal, Forbes itd. Sljedeći okvir koji je posljednjih godina zadobio izrazito veliku popularnost je React, a razvijen je od strane Facebook-a. Ovaj okvir slijedi arhitekturu koja se temelji na komponentama te su sučelja podijeljena na komponente za višekratnu upotrebu. Ono što je zanimljivo za React je da optimizira značajke korištenjem virtualnog DOM-a što rezultira bržim početnim vremenom učitavanja stranice, a što je važno za razvoj dinamičnih aplikacija. No, ovaj okvir također može predstavljati izazov za programere s manje iskustva te ponekad može stvoriti nepotrebnu složenost za razvoj manjih i jednostavnijih aplikacija.[45] Ovaj okvir također koriste neke od popularnih aplikacija kao što su Facebook, Instagram, Airnb itd. Za razliku od Angular-a i React-a, Vue ima manju popularnost i prihvaćenost, ali iako ima manju bazu korisnika, ovaj okvir zbog svoje jednostavnosti i lakoće korištenja, posljednjih godina ima rastuću zajednicu. Kreirao ga je Evan You i objavio 2014. godine, a s jednostavnim i fleksibilnim dizajnom olakšava integraciju u postojeće projekte ili izradu novih aplikacija.[46] Ovaj okvir također slijedi arhitekturu temeljenu na komponentama te koristi virtualni DOM za povezivanje podataka te učinkovito ažuriranje i renderiranje komponenti, a neke poznate aplikacije zasnovane na ovom okviru su Alibaba i Xiaomi. S obzirom na lakoću korištenja, Vue je idealan okvir za početnike, ali također i fleksibilan za programere s više iskustva pa će se s obzirom na to praktični dio rada odraditi uz pomoć ovog okvira. Naravno, osim navedenih Angular, React i Vue okvira, postoje i brojni drugi okviri koji razvojni programeri koriste za izradu jednostranične aplikacije kao što su: Ember, Backbone, Aurelija, Meteor, Nokaut itd. Odabir okvira jednostranične aplikacije je dugotrajan proces te kao što se može vidjeti na opisanim primjerima svaki od njih ima određene prednosti i nedostatke, stoga, iako je također moguće kombinirati različite okvire, zbog mogućeg sukoba i povećane složenosti preporučuje se odabrati jedan okvir koji najbolje odgovara potrebama same aplikacije ovisno o njezinim značajkama.[47]

[44] Usp. Patel, Bhaval. 8 Top Single-Page Application Frameworks for Web App Development // Space-O Technologies, 2024. URL: https://www.spaceotechnologies.com/blog/single-page-application-frameworks/ (2024-07-01)
[45] Usp. Isto.
[46] Usp. Isto.
[47] Usp. Isto.

5. Prednosti i nedostaci jednostraničnih aplikacija

Kroz usporedbu jednostraničnih i višestraničnih aplikacija te prikaz arhitekture jednostraničnih aplikacija u prethodnim poglavljima, već su se mogle uočiti prednosti razvoja jednostraničnih mrežnih aplikacija. No, kao i sve druge arhitekture, osim brojnih prednosti ovakav tip aplikacije može imati određene nedostatke. Kako bi se uvidjelo jesu li jednostranične aplikacije bolji izbor prilikom razvoja mrežnih aplikacija u nastavku će se najprije navesti pozitivne, a zatim i negativne strane takve aplikacije. Kao jedna od glavnih prednosti razvoja jednostraničnih aplikacija koja se može izdvojiti je brzina. U prethodnim poglavljima već je objašnjeno da jednostranična aplikacija funkcionira na način da se HTML datoteka zajedno sa CSS datotekom i skriptama učitava pri pokretanju aplikacije te da se tijekom korištenja aplikacije stranica ne osvježava, nego se mijenjaju podaci koji se prenose na i sa poslužitelja pa na ovaj način aplikacija znatno brže odgovara na korisničke upite odnosno zahtjeve.[48] Zbog brzine, sadržaj se dinamički mijenja zbog čega se korisniku pruža bolji doživljaj bez vidljivih prekida. Među brojnim provedenim studijama tvrtke Google, i ključnim nalazima tvrtke Amazon i Walmart donijeti su zaključci da brzina stranice može znatno ovisiti o uspješnosti posla pa tako stranica kojoj je potrebno više od 200 milisekundi za učitavanje može negativno utjecati na posao ili financijski koštati tvrtku.[49] Primjerice, Amazon navodi kako ih jedna sekunda kašnjenja stranice košta 1% prodaje što godišnje na ukupan iznos Amazon-ove prodaje iznosi 1,6 milijardi dolara.[50] S obzirom da je brzina jedna od najbitnijih stavki pozitivnog korisničkog iskustva, ovo je zasigurno dobar razlog za odabrati ovakav model pri razvoju mrežnih aplikacija. Osim na brzinu, izvršavanje samo jednog zahtjeva pri početnom preuzimanju omogućuje i predmemoriranje. Odnosno, korisnici jednom primljene podatke mogu koristiti i u izvanmrežnom radu što omogućuje manju potrošnju podatkovnog prometa, ali i sinkroniziranje podataka s poslužiteljem ukoliko mreža to dopušta.[51] Navedene prednosti dio su pozitivnih strana koje utječu na korisničko iskustvo koje je danas od iznimne važnosti, no osim što nudi mogućnost za boljim korisničkim iskustvom, jednostranične aplikacije imaju brojne prednosti u samom razvojnom procesu. Izrada jednostranične aplikacije sa stajališta programera je

[48] Usp. What is a Single Page Application? Pros and Cons the SPA Technology // Huspi, 2022. URL: https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications/ (2023-07-27)
[49] Usp. Isto.
[50] Usp. Isto.
[51] Usp. BasuMallick, Chiradeep. Nav. dj.

pojednostavljena i optimizirana te je puno lakše otkriti i otkloniti eventualne pogreške u kodu.[52] Primjerice, pomoću Google Chrome alata razvojni programeri mogu provjeravati kod pomoću preglednika te otkloniti pogreške bez pretraživanja većeg broja linija koda na strani poslužitelja. Osim toga, ovi alati pomoću konzole pružaju mogućnost praćenja i istraživanja svih postojećih elemenata na stranici, nadziranje mrežnih operacija te povezane podatke.[53] Također, proces razvoja aplikacije je brži jer je moguć paralelni rad dvaju ili više programera s obzirom da se prednji i stražnji dio jednostranične aplikacije može odvojiti, a sve promjene koje se odvijaju u jednom dijelu, ne utječu na drugi dio.[54] Kao još jedna prednost i razlog razvoja ovakvih aplikacija jest da se jednostranične aplikacije mogu lakše pretvoriti u druge vrste aplikacija s obzirom da se kod može jednostavno upotrijebiti za prijelaz s jednostranične mrežne aplikacije na primjerice mobilnu aplikaciju.[55] Pri razvoju jednostranične aplikacije moguće je koristiti jednu bazu kodova za izradu aplikacija koje se izvode na različitim uređajima, pregledniku ili operativnom sustavu.[56] Osim mobilnih aplikacija, već su spomenute i progresivne mežne aplikacije u koje je također moguće transformirati jednostraničnu aplikaciju. Na taj način moguće je osigurati lokalno predmemoriranje i ponuditi korisnicima izvanmrežno (*eng. offline)* iskustvo.[57]

Navedene prednosti samo su neki od razloga razvoja jednostranične aplikacije, no unatoč svim prednostima postoje izazovi kada jednostranična aplikacija nije idealno rješenje. Brojni klijenti i korisnici očekuju kvalitetne i bitne mrežne stranice ili aplikacije na visokim pozicijama u rezultatima tražilica odnosno u pogledu SEO-a, no većina tražilica kao što su Google ili Yahoo, nemaju mogućnost indeksiranja jednostranične mrežne aplikacije koje koriste AJAX za interakciju s poslužiteljem.[58] Zbog toga, većina jednostraničnih aplikacija nije indeksirana, a korištenje JavaScript umjesto HTML-a za indeksiranje šteti rangiranju što predstavlja ograničenje za određene aplikacije. No, arhitektura jednostranične aplikacije prikladna je za platforme SaaS (*eng. skraćeno Software-as-a-Service)*, odnosno zatvorene zajednice i društvene mreže kao što je

[52] Usp. Isto.
[53] Usp. Isto.
[54] Usp. Isto.
[55] Usp. Isto.
[56] Usp. Isto.
[57] Usp. Ćwik, Przemyslaw. Nav. dj.
[58] Usp. BasuMallick, Chiradeep. Nav. dj.

Facebook jer takvim aplikacijama nije potrebna optimizacija za pretraživanje u tražilicama.[59] Osim analitike u tražilicama, na jednostraničnim aplikacijama prisutan je izazov s praćenjem analitike same aplikacije. Iako je moguće dodati kod za praćenje prometa aplikacije, s obzirom da se radi o jednoj stranici teško je odrediti koji sadržaj je popularniji odnosno koja stranica je više posjećena. Međutim, moguće je osigurati dodatni kod za praćenje pseudo stranica dok se prikazuju, što zahtijeva više znanja i dodatni rad na razvoju.[60] Unatoč tome da se jednostranična aplikacija učitava samo jednom pri početnom zahtjevu, nedostatak je da su korisnici naviknuti u pregledniku koristiti gumb za povratak nakon kojega se očekuje brzo vraćanje na zadnje stanje. Iako višestranične aplikacije imaju mogućnost korištenja predmemoriranih kopija mrežnih stranica, to nije slučaj s jednostraničnim aplikacijama te će se klikom na gumb za povratak zahtjev poslati ponovo na poslužitelj, a stranica će se ponovo učitati odnosno osvježiti što će rezultirati dužim vremenom čekanja i promjenama sadržaja.[61] Nadalje, svaka jednostranična aplikacija za pokretanje i razvoj striktno koristi JavaScript. Postoji mogućnost da ga korisnici onemoguće na svojim uređajima, a ukoliko je JavaScript onemogućen, aplikacija se neće pokrenuti.[62] Također, osim o JavaScript-u, jednostranične aplikacije uvelike ovise o preglednicima. Najnoviji preglednici koji podržavaju potrebne alate omogućiti će nesmetan rad aplikacije, no stariji preglednici kao što su primjerice Internet Explorer mogli bi utjecati na učinkovitost i kvalitetu aplikacije pa i u potpunosti onemogućiti rad na aplikaciji.[63] Još jedan nedostatak kojeg je potrebno imati na umu pri razvoju jednostraničnih aplikacija je sigurnost. Pomoću XSS-a (*eng. skraćeno Cross-site-scripting)* hakeri mogu ostvariti pristup stranici i ubaciti nove skripte na strani klijenta.[64] Također, korisnici imaju mogućnost preuzimanja cijele aplikacije čime je povećana ranjivost i sigurnost aplikacije, a kako bi se to spriječilo programeri bi trebali ograničiti pristup te povezati aplikaciju sa sigurnošću mrežne aplikacije kao što je provjera autentičnosti i sl.[65] Nakon navedenih prednosti i nedostataka, može se reći kako jednostranična aplikacija također ima

[59] Usp. What is a Single Page Application? Pros and Cons the SPA Technology. Nav. dj.
[60] Usp. Isto.
[61] Usp. Isto.
[62] Usp. Munawar, Shahzaib. Single Page Applications (SPAs). Most Discussed Pros & Cons. // Novateus, 2022. URL: https://novateus.com/blog/single-page-applications-spas-most-discussed-pros-cons/ (2023-07-27)
[63] Usp. Isto.
[64] Usp. What is a Single Page Application? Pros and Cons the SPA Technology. Nav. dj.
[65] Usp. BasuMallick, Chiradeep. Nav. dj.

određene izazove koji bi mogli negativno utjecati na određene vrste aplikacija. Stoga, prilikom odluke o odabiru modela za razvoj aplikacije, potrebno je odrediti prioritete i odabrati aplikaciju u skladu s njima.

6. Razvoj jednostranične mrežne aplikacije

6.1. Opis aplikacije

Nakon što su se utvrdili temeljni teorijski koncepti jednostranične aplikacije u nastavku će se prikazati proces izrade aplikacije s takvom arhitekturom. Aplikacija koja se izrađuje poslužiti će isključivo za potrebe rada te neće javno objavljena, a svi scenariji zamišljeni su kako bi se pobliže pojasnile osnove ove vrste aplikacije. Prvi korak pri izradi aplikacije je istražiti problem i potrebu za aplikacijom, a potom definirati korisničku skupinu te osnovnu namjenu aplikacije. U modernim knjižnicama postoji potreba za efikasnim upravljanjem prostorom i omogućavanjem korisnicima da rezerviraju mjesta za različite aktivnosti. Korisnici knjižnica često žele rezervirati svoja mjesta prije dolaska u knjižnicu dok zaposlenici s druge strane trebaju kvalitetno rješenje za praćenje rezervacija i stanja u knjižnici. S obzirom na postojeći problem, zamišljeno je da se izradi aplikacija koja će riješiti navedene probleme pružajući korisničko sučelje za rezervacije i jednostavne administrativne alate za odobravanje i praćenje tih rezervacija. Dakle, primarna korisnička skupina aplikacije su članovi knjižnice koji žele rezervirati mjesta za različite prostorije (tihi rad, grupni rad, čitaonica), a sekundarna korisnička skupina su zaposlenici odnosno administratori knjižnice koji će koristiti aplikaciju za praćenje rezervacija i odobravanje istih. Aplikacija je dizajnirana da bude jednostavna za korištenje, ali također s glavnim ciljem da zadovolji standarde koje zahtijeva jednostranična mrežna aplikacija te standarde pozitivnog korisničkog iskustva. Za izradu aplikacije koristi se Vue kao *frontend* radni okvir, dok će *backend* biti podržan od strane *json-server* alata koji će simulirati REST API (*Representational State Transfer Application Programming Interface)* koristeći lokalnu JSON datoteku kao bazu podataka. Node.js će služiti kao platforma za pokretanje *json-server* alata i upravljanje razvojnim okruženjem. S obzirom da bi administrator trebao imati drugačije sučelje od korisnika knjižnice, za početak će se izraditi forma za prijavu i postaviti logika za autorizaciju. Nakon uspješne prijave, aplikacija će prikazivati sadržaj ovisno o ulozi korisnika. Na taj način, administratori će imati pristup administrativnim alatima, dok će korisnici imati pristup funkcionalnostima za rezervaciju. Izrada i detaljan opis sadržaja za svaku korisničku ulogu bit će objašnjeni u nastavku rada.

6.2. Hodogram razvoja aplikacije i korištene tehnologije

Prvi dio prikazuje hodogram izrade aplikacije, uključujući instalacije, sami kod i logiku aplikacije. Korištene tehnologije su HTML5, CSS3 te JavaScript programski jezik te osim osnovnih tehnologija, važno je odabrati razvojno okruženje koje će se koristiti pri izradi aplikacije. U svrhu ovog rada odabrani alat za izradu je Visual Studio Code, jedan od najpoznatijih alata za razvoj aplikacija. Osim programskog jezika, dobro je odabrati jedan od radnih okvira. Za izradu ove aplikacije odabran je Vue radni okvir zbog svoje jednostavnosti u razvoju jednostraničnih aplikacija te prikladnosti za početnike u programiranju. Nastojat će se opširno opisati kod koji se koristio za izradu ove aplikacije, no zbog nemogućnosti prikaza cjelokupnog koda, ostatak će biti dostupan na GitHub-u na sljedećoj poveznici: https://github.com/kculap/single-page-app

6.2.1. Instalacija potrebnih paketa i postavljanje radnog okruženja

Prvi korak je postavljanje razvojnog okruženja i instalacija potrebnih tehnologija. Za početak na računalu na kojemu se izrađuje aplikacija je potrebno imati instalirane alate koji će služiti pri razvoju aplikacije kao što je Visual Studio Code, Node.js te NPM (Node Package Manager). Node.js je tzv. JavaScript *runtime* koji omogućava izvršavanje JavaScript koda izvan preglednika te je zbog toga vrlo prikladan za izradu jednostraničnih aplikacija koje zahtijevaju interakciju u stvarnom vremenu između poslužitelja i klijenta pomoću API-a, dok NPM služi za upravljanje i omogućuje pristup velikom broju potrebnih paketa i modula koji će biti korišteni tijekom razvoja aplikacije s ciljem poboljšanja njezine funkcionalnosti.[66] S obzirom da se ova aplikacija razvija u Vue radnom okviru, za postavljanje radnog okruženja potrebno je imati instaliran i Vue CLI koji služi za brzo postavljanje Vue projekata te bez njega neće biti moguće kreirati projekt. U ovome radu koristi se Vue CLI v5.0.8 verzija, a najnoviju verziju moguće je instalirati u terminalu pomoću naredbe:

```
npm install -g @vue/cli
```

[66] Usp. Makinde, Felix. Node.js vs NPM: Understanding the Differences // HostAdvice, 2024. URL: https://hostadvice.com/blog/web-hosting/node-js/node-js-vs-npm/ (2024-07-01)

Nakon što je CLI uspješno instaliran, može se započeti s kreiranjem i razvojem aplikacije. Za početak potrebno je kreirati Vue projekt pomoću naredbe za kreiranje Vue projekta i dodavanjem naziva projekta, npr.:

```
vue create single-page-app
```

Kada je projekt uspješno kreiran, u korijenskom direktoriju vidljive su nove datoteke i direktoriji koje su potrebne za nastavak rada dok će neke datoteke biti dodane naknadno ovisno o potrebama aplikacije. Kako bi se aplikacija pokrenula na poslužitelju potrebno je ući u korijenski direktorij projekta, što se može učiniti pomoću naredbe:

```
cd single-page-app
```

Prilikom korištenja ove naredbe važno je imati na umu da je potrebno navesti točnu putanju gdje je projekt smješten prilikom kreiranja, npr. ukoliko se nalazi na pozadini računala to je potrebno također navesti u putanji. Nakon što se pozicionira na korijenski direktorij projekta, pomoću naredbe:

```
npm run serve
```

pokreće se lokalni razvojni server što omogućuje pregledavanje aplikacije u pregledniku te će također automatski osvježavati aplikaciju pri svakoj promjeni u kodu, što olakšava brzi razvoj. Nakon uspješnog kreiranja projekta i pokretanja razvojnog okruženja, sljedeći korak je integracija s *json-server* alatom kako bi se omogućila simulacija *backend* dijela aplikacije. S obzirom da je u ovom radu naglasak na *frontend* i korisničko iskustvo za ovakav tip jednostavnih aplikacija nije potrebno imati pravu bazu podataka stoga će za ovu aplikaciju poslužiti tzv. lažni (*mock)* API. Odnosno, *json-server* je alat koji omogućuje jednostavnu simulaciju REST API-a koristeći lokalnu JSON datoteku kao bazu podataka.[67] Kako bi se omogućio globalni pristup alatu iz bilo kojeg dijela aplikacije, najprije je potrebno u terminalu instalirati *json-sever* putem NPM-a čija se važnost i instalacija opisala ranije, a za to se koristi sljedeća naredba:

```
npm install -g json-server
```

[67] Usp. Sharma, Riya. Mock APIs Using JSON Server // Medium, 2023. URL: https://medium.com/@theriyasharma24/create-mock-apis-using-json-server-for-seamless-development-4a6ba03ef148 (2024-07-01)

Nakon što je json-server uspješno instaliran, kao izvor podataka koristi datoteku *db.json* u kojoj je potrebno definirati strukturu. U ovom slučaju struktura će se podijeliti na tri glavne komponente: postove, komentare i profil korisnika.[68] Unutar koda može se vidjeti podjela gdje su navedenim objektima dodijeljeni određeni atributi i vrijednosti. Odnosno, iz priloženog se može vidjeti struktura koja je definirana da postoji jedan post sa specifičnim naslovom i autorom što će poslužiti pri rezervaciji korisnika, jedan komentar koji se odnosi na taj post odnosno što će poslužiti kao određeni komentar ispod rezervacije te profil korisnika koji sadrži samo ime što će se iskoristiti za personaliziranje odnosno prikazivanje imena korisnika koji je rezervirao mjesto. Sljedeći korak je konfigurirati *axios.js* datoteku kako bi se omogućila komunikacija s *json-server* alatom putem HTTP zahtjeva. Najprije je potrebno pomoću NPM-a instalirati *axios* pomoću naredbe:

```
npm install axios
```

Zatim, unutar *axios.js* datoteke potrebno je dodati određene postavke poput API-a, vremenskog ograničenja, zaglavlja te podrške za kolačiće (*eng. cookies).* Također u ovoj datoteci postavit će i logika u Axios-u koja omogućava manipuliranje ili promatranje HTTP zahtjeva prije nego što se oni izvrše.[69] Nakon što je osigurana simulacija podataka za *backend* sljedeći korak je strukturiranje komponenti.

6.2.2. Prijava i registracija

Projekt je inicijalno kreiran kroz terminal, što znači da već sadržava osnovne datoteke i direktorije potrebne za daljnji razvoj. Međutim, kako bi se aplikacija uspješno razvila, bit će potrebno prilagoditi njezinu strukturu dodavanjem novih komponenti i resursa prema potrebama projekta dok će nepotrebni dokumenti biti izbrisani. U ovoj aplikaciji postoje dvije uloge unutar kojih će se odvijati sučelje: korisnik i administrator. Prema tome, nakon uspješne prijave izradit će se logika koja će provjeravati uloge i prikazati sadržaj na osnovu toga. No, najprije je potrebno izraditi formu za prijavu (*eng. login)* i registraciju (*eng.registration)*. Kod izrade sučelja prijave važno je imati na umu koji paketi iz NPM-a bi mogli koristiti i olakšati razvoj, a u ovom slučaju korišteni paketi su Pinia koja predstavlja moderniju zamjenu za Vuex te omogućuje upravljanje stanjem, Element

[68] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/db.json
[69] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/modules/axios.js

Plus koji će pomoći pri izradi korisničkog sučelja s gotovim komponentama te na kraju CryptoJS biblioteka koja se koristi za kriptografske algoritme te će poslužiti za hashiranje lozinki. Za Vue aplikacije, primarni alat za upravljanje stanjem bio je Vuex. Međutim, 2019. godine Vue je predstavio novi alat pod nazivom Pinia. S obzirom da je u 2023. godini objavljena njezina stabilna verzija te se za Vuex ne najavljuju nove značajke, programeri više preferiraju modernije pristupe pa će se samim time i u ovom radu koristiti takav pristup.[70] Kako bi se aplikacija povezala s Pinia podrškom, unutar glavne datoteke *main.js* koja će biti ulazna točka za tzv. *webpack,* uz konfiguraciju drugih alata i biblioteka potrebno je uvesti Pinia instancu za upravljanje stanjem.[71] Unutar *main.js* datoteke nalazi se i temeljna funkcija za implementaciju jednostraničnih aplikacija. Funkcija *createRouter* koristi se za inicijalizaciju Vue Router instance te prilikom razvoja jednostraničnih aplikacija ključna je jer osigurava navigaciju između različitih *ruta* odnosno stranica te ih organizira i osigurava mehanizme za upravljanje poviješću aplikacije. Ova svojstva standardi su jednostraničnih aplikacija koje ovakve aplikacije čine interaktivnijima i boljima za krajnje korisnike. Kada je postavljena struktura *main.js* aplikacije potrebno je vratiti se na postavljanje stanja aplikacije što će poslužiti za izradu *Login.vue i Registration.vue* komponenti. Stanja se mogu definirati unutar datoteke *store.js* te su dodana prema potrebi usporedno s definiranjem funkcija unutar komponenti. Nakon što su stvorene datoteke za prijavu i registraciju, zbog jednostavnosti najprije je izrađen HTML *template* te će se na osnovu njega dodavati potrebne funkcije i varijable te prilagođavati stilovi. Obično se svaka *.vue* datoteka sastoji se od tri vrste blokova: <*template*>, <*script*> i <*style*>. Nakon što se izradio *template* potrebno je dodati logiku za obje komponente te potrebne uvoze paketa i drugih modula. U *Login.vue* komponenti deklarirane su *ref* varijable *loginData* i *passwordModel* kako bi se omogućilo praćenje unosa korisničkog imena i lozinke. Nakon što korisnik klikne na *button* za prijavu potrebno je osigurati funkciju koja će pratiti ispravnost unesenih podataka, u ovom primjeru ta je funkcija nazvana *checkData()*. Dakle, nakon što korisnik klikne na *button* Prijava, pokreće se funkcija *login()* koja najprije provjerava jesu li svi navedeni uvjeti funkcije *checkData()* zadovoljeni kao što su duljina korisničkog imena i lozinke, format i sl. Ukoliko podaci nisu ispravni korisniku će se prikazati

[70] Usp. Gerchev, Ivaylo. Leveraging Pinia to simplify complex Vue state management //LogRocket: Frontend Analytics, 2024. URL: https://blog.logrocket.com/complex-vue-3-state-management-pinia/ (2024-07-02)
[71] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/main.js

obavijest, a ukoliko su ispravni pozvat će se funkcija *loginUser()*.[72] Funkcija *loginUser()* enkriptira unesenu lozinku pomoću MD5 *hash* funkcije iz CryptoJS paketa koji se prethodno preuzeo te potom poziva *backend* putem metode *store.userLogin()*. MD5 (eng. *skraćeno message-digest algorithm*) je kriptografski protokol koji pokreće datoteku kroz matematički algoritam kako bi provjerio autentičnost poruka, sadržaja ili digitalnih potpisa te se najčešće koristi za autentifikaciju.[73] Nakon što se provjera dovrši, ukoliko se korisnik uspješno prijavi podaci će se pohraniti lokalno preko *localStorage* te će se preusmjeriti na odgovarajuću stranicu ovisno o ulozi koja će se kasnije u radu definirati, u suprotnom, ako je prijava neuspješna prikazat će se obavijest o pogrešnoj prijavi.[74] Na sličan način implementirana je i komponenta za registraciju. U ovoj komponenti također su korišteni preuzeti paketi Element Plus te CryptoJS, a uvezeni su i ostali potrebni moduli poput *ref* funkcije za praćenje stanja, te *defineEmits* za definiranje emitiranih događaja i sl.. Varijable za praćenje stanja provjeravaju ispravnost unesenih podataka kao što su ime, prezime, email, korisničko ime, lozinka te ponovljena lozinka. Kako bi se provjerila dostupnost email-a i korisničkog imena potrebno je osigurati asinkronu provjeru sa serverom što izvršavaju funkcije *store.checkRegistrationEmail()* i *store.checkRegistrationUsername()* definirane u ranije spomenutoj datoteci *store.js*.[75] Na ovaj način želi se provjeriti postoji li već uneseni email i korisničko ime u sustavu. Nakon što se unesu svi ispravni podaci i zadovolje svi uvjeti, funkcija *save()* omogućava spremanje svih podataka na server pomoću metode *store.user.Registration()* te se odgovor pohranjuje u lokalnoj pohrani kao u prošlom primjeru s komponentom *Login.vue.*

6.2.3. Korisnik

Nakon što su se izradile komponente za prijavu i registraciju, slijedi prikaz sljedeće stranice ovisno o ulogama: korisnik ili administrator. Kako bi se prilagodilo koja bi se stranica trebala prikazati potrebno je dodati funkciju koja će provjeravati podatke i na osnovu njih prepoznati radi li se o

[72] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/Login.vue#L20
[73] Usp. Freda, Anthony. What Is the MD5 Hashing Algorithm and How Does It Work? // Avast, 2022. URL: https://www.avast.com/c-md5-hashing-algorithm (2024-07-02)
[74] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/Login.vue#L54
[75] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/stores/store.js#L9

korisniku ili administratoru. Ove uvjete provjeravati će funkcija *checkForPage()* koja je smještena u komponenti pod nazivom *MainPage.vue* te će ona biti prikazana kao glavna početna stranica i prikazivati će sadržaj ovisno o ulozi.[76] Dakle, unutar koda može se vidjeti kako funkcija najprije dohvaća podatke spremljene unutar lokalne pohrane (*eng. local storage)* pod ključem *userData*. Zatim, metoda *JSON.parse()* pretvara niz (*eng. string)* u JavaScript objekt te ga pohranjuje kao varijablu *userData.* U nastavku je definirana *else if* petlja koja ima zadatak provjeriti podatke o korisnicima, a ukoliko podataka nema ili određeni dio nedostaje i dalje će se prikazivati samo početna stranica za prijavu. Ukoliko varijabla vrati vrijednost *libraryAdmin* prikazat će se stranica administratora te ukoliko vrati vrijednost *user* stranica će prikazati sučelje za korisnika. Nakon definiranja *MainPage.vue* datoteke može se krenuti s izradom stranice za ulogu korisnika. Kako bi se organizirala preglednija struktura unutar direktorija *components* stvoreni su novi direktoriji *admin* i *user* te unutar njih nove komponente. U dijelu za korisnika dodane su komponente: *User.vue, UserNewAppointment.vue, UserAppointments.vue* te *UserDasboardRoomsStatus.vue.* Prva komponenta koja se nalazi zasebno u direktoriju predstavljat će glavnu komponentu korisnika te će ona sadržavati HTML *template* sa glavnim kontejnerima, navigacijskom trakom koja uključuje funkcije za odjavu korisnika i prikaz određenih elemenata iz drugih komponenti. Druga komponenta, *UserNewAppointment.vue* uključiti će potrebne funkcije za dodavanje novih rezervacija od strane korisnika te ostale značajke vezane za rezervacije. Za ovaj dio aplikacije važno je inicijalizirati varijable koje bi mogle biti korisne za prikaz sučelja za rezervaciju. S obzirom da je zamišljeno da korisnik može rezervirati mjesto za 3 različite prostorije potrebno je dodati *ref* varijablu koja će sadržavati niz odnosno sve opcije prostorija koje korisnik može odabrati te varijablu koja će pratiti trenutno stanje odabrane prostorije što će ograničiti daljnje korake u rezerviranju, ali i poslužiti za dodavanje trenutnog stanja u zadnjoj komponenti. U nastavku, s obzirom da korisnik bira datum i vrijeme svoje rezervacije, potrebno je ograničiti odabir pomoću funkcija za manipulaciju vremena kao što su: *disabledDatedisabledStartHours, disabledStartMinutes, disabledEndHours, disabledEndMinutes, disabledSeconds*.[77] Također, osim što korisnik mora označiti datum i vrijeme dolaska u prostoriju, od njega će se tražiti da

[76] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/MainPage.vue#L12
[77] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/user/components/UserNewAppointment.vue#L64

navede i vrijeme odlaska te će se podaci provjeravati unutar funkcije za postavljanje podataka *setAppointmentData,* a pomoću *computed()* svojstva u nastavku će se izračunati i prikazati trajanje termina, što će se moći vidjeti i u sljedećim poglavljima rada kod prikaza samog sučelja. Kada korisnik označi datum i vrijeme, u nastavku će mu se prikazivati zauzetost za taj termin što mu može koristiti pri odabiru termina. To će se omogućiti funkcijom *checkAppointmentState* koja filtrira unesene podatke i provjerava postoji li preklapanje s drugim terminima.[78] Nakon što se svi podaci ispravno ispune, klikom na *button* za nastavak poziva se funkcija za spremanje termina *saveAppointmentHandler* koja provjerava jesu li svi podaci uneseni te sprema podatke i prikazuje ih zajedno s ostalim terminima koje je korisnik rezervirao, a to se izvršava pomoću emitiranja događaja naredbom *emit('goToAppointments')* koja korisnika šalje na sljedeću komponentu, a to je *UserAppointments.vue.* Glavna svrha ove komponente je da ponudi prikaz svih rezervacija prijavljenog korisnika te da omogući korisniku pregledavanje detalja svake rezervacije kao što su datum, vrijeme te je li rezervacija odobrena ili odbijena od strane administratora. Za to će biti najprije potrebno dodati nekoliko HTML elemenata koji će prikazati listu u obliku vremenske trake. Zatim će se dodati funkcije za dohvaćanje termina, funkcija za promjenu odnosno ažuriranje stranice te funkcija za brisanje termina. S obzirom da je u temi diplomskog rada naglasak na korisničkom iskustvu, ukoliko korisnik ima velik broj rezervacija zbog količine podataka to bi moglo opteretiti aplikaciju te povećati njeno vrijeme učitavanja. Kako bi se poboljšala izvedba ovog dijela aplikacije za dohvaćanje termina koristiti će se paginacija što znači da će umjesto prikaza svih termina odjednom, dohvaćati samo određen broj termina odnosno stranica koje će se prikazati korisniku. Kada korisnik putem paginacije prijeđe na sljedeću stranicu, dohvaća se sljedeći skup termina. Kako bi napravili listu s paginacijom, najprije je potrebno definirati varijablu koja inicijalizira paginaciju sa početnom stranicom, brojem termina po stranici i ukupnim brojem termina.[79] Nakon toga, varijablu je potrebno unutar funkcija i HTML dijela uvesti na odgovarajući način. Posljednja komponenta u dijelu sučelja za korisnika je *UserDashboardRoomsStatus.vue* te je uključena s ciljem da korisniku prikaže stanje uživo (*eng.*

[78] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/user/components/UserNewAppointment.vue#L152

[79] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/user/components/UserAppointments.vue#L21

live) u svakoj od tri prostorije. Svakoj prostoriji definirano je početno stanje s nulom te se ono mijena sukladno rezerviranim terminima iz komponente za rezervaciju termina. Za svaku prostoriju potrebno je dodati zasebnu asinkronu funkciju koja će dohvaćati trenutno stanje zauzetosti za tu prostoriju. Ova funkcija može se prikazati pomoću tzv. *try catch* bloka. Blok *try* izvršava kod koji može izazvati određenu grešku te se pomoću naredbe *await* čeka završetak operacije za pohranjivanje rezervacija dok se asinkronom funkcijom *store.getAppointmentRequest()* šalje zahtjev za dohvaćanje podataka. Unutar funkcije definirani su parametri koji filtriraju termine koji su završeni i odobreni te ovisno o prostoriji specifično traže termine primjerice za prostoriju za tihi rad za trenutni datum.[80] Zatim se obrađuje odgovor kroz navedene podatke i filtere. S druge strane, blok *catch* „hvata" pogreške koje su se eventualno dogodile i vraća obavijest korisniku.

6.2.4. Administrator

Druga uloga koja ima nešto drugačije sučelje u odnosu na korisnika je administrator. Kao što se moglo vidjeti u opisu komponenti korisnika, korisnici zahtjeve za rezervaciju šalju administratorima na odobrenje stoga je za ovaj dio aplikacije potrebno omogućiti prikaz svih zahtjeva od strane korisnika. Ideja je da unutar navigacije administrator ima mogućnosti odabrati prikaz svih odobrenih i odbijenih zahtjeva te novih zahtjeva koji čekaju odgovor. Također, administrator bi trebao imati mogućnost pregledavanja zauzetosti po prostorijama za određeni datum i vrijeme te se planira uključiti i ta funkcionalnost. Najprije će se stvoriti sve potrebne komponente: *Admin.vue, AdminAllRequests.vue, AdminRequests.vue i AdminGroupOverview.* Poput komponente *User.vue* iz prethodnog potpoglavlja, komponenta *Admin.js* predstavljat će glavnu komponentu u kojoj će se najprije pokrenuti funkcija provjere stranice, a nakon toga će se prikazivati *template* koji će uključivati osnovne značajke, navigaciju, odjavu administratora i sl. Sljedeća komponenta koja se izrađuje je *AdminAllRequests.vue* koja će omogućiti prikaz svih zahtjeva za rezervaciju koje su korisnici zatražili. Za ovu komponentu bitne funkcije su za dohvaćanje zahtjeva za trenutnu stranicu specificiranu paginacijom te dohvaćanje ukupnog broja zahtjeva. U ovom primjeru te funkcije će se nazvati *getRequests i getTotalRequests.* Kao i u primjeru kada korisnik pokušava pristupiti svim poslanim zahtjevima administratoru, na sličan

[80] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/user/components/UserDashboardRoomsStatus.vue#L21

način će i ove funkcije putem *try catch* bloka i paginacijom dohvatiti zahtjeve od strane korisnika. Prilikom definiranja ovih funkcija važno je dodati i *onMounted* odnosno tzv. *hook* koji osigurava učitavanje podataka tek kada je komponenta spremna za prikaz.[81] Osim navedenih funkcija za dohvaćanje zahtjeva, kako bi se omogućio interaktivan prikaz dodati će se i funkcije za ažuriranje stranice te dodavanje dialoga koji će prikazati detalje zahtjeva i omogućiti njihovo brisanje. Za razliku od ove komponente koja omogućuje prikaz svih zahtjeva, komponenta *AdminRequests.vue* prikazivati će samo nove odnosno zahtjeve koji čekaju odgovor. Za to će biti također potrebna funkcija koja će dohvaćati pohranjene zahtjeve, ali u ovom slučaju uz paginaciju i sortiranje, morati će biti zadovoljen i parametar zahtjeva koji su u stanju čekanja.[82] Također, kako bi administrator mogao prihvatiti ili odbiti zahtjev potrebno je dodati funkcije koje će pratiti izvršavanje tih akcija.[83] Može se vidjeti kako postoje tri dodane funkcije, prva funkcija *handleCurrentChange* mijenja trenutnu stranicu paginacije te ponovno dohvaća zahtjeve za novu stranicu te ih prikazuje na vrhu stranice. Druga funkcija *rejectRequestHandler* postavlja status zahtjeva na „done“ (*hrv. dovršeno)* te poziva funkciju *saveRequest* koja sprema promjene zahtjeva u slučaju da je zahtjev odobren ili odbijen te ponovo dohvaća zahtjeve kako bi se osvježio prikaz stranice. Suprotno *rejectRequestHandler* funkciji, funkcija *approveRequestHandler* postavlja approved na true i status na "done", zatim poziva *saveRequest* čime je zahtjev odobren. Nadalje, osim u sučelju za korisnike, administrator također ima mogućnost prikaza zauzetosti prostorija. To će se prikazivati na način da će se unutar komponente *AdminGroupOverview* u navigaciji prikazati svaka prostorija te će svaka od njih prikazivati trenutnu zauzetost. Ovo će se postići dodavanjem funkcije za praćenje stanja na sličan način kao što se to učinilo kod praćenja zauzetosti u sučelju za korisnike. No, kako bi se dodala malo bolja funkcionalnost za administratora, osim trenutnog stanja u ovom dijelu biti će dodana funkcija i za praćenje budućih stanja za bilo koji rezervirani termin. Za to će biti potrebno uz funkciju *getTodayAppointments* koja će pratiti

[81] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/admin/components/AdminAllRequests.vue#L32
[82] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/admin/components/AdminRequests.vue#L22
[83] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/admin/components/AdminRequests.vue#L41

trenutno stanje, dodati i funkciju *getCalendarRequests* koja će učitati termine za odabrani datum u kalendaru.[84] Za prikaz kalendara i interakciju s njime koristiti će se komponenta *ElCalendar* koja dolazi iz Element Plus paketa koji je spomenut ranije u radu prilikom instalacije svih potrebnih paketa. Iz istog paketa koristiti će se i komponenta za integraciju s tablicom koja će prikazivati detalje o rezervaciji prema odabiru iz kalendara. Na samom kraju kako bi se dodale još neke funkcionalnosti koje utječu na korisničko iskustvo za obje uloge odnosno i kod administratora i korisnika, unutar navigacije dodan je profil korisnika na kojemu je moguće provjeriti podatke koji su se upisivali prilikom registracije te je moguće promijeniti lozinku. Ti podaci smješteni su unutar komponente *UserAndAdminDetails.js* te koriste funkcije kao i kod registracije koje provjeravaju duljinu, format i spremaju nove podatke.[85]

[84] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/admin/components/AdminGroupOverview.vue#L61
[85] https://github.com/kculap/single-page-app/blob/1863f661ada6fee6a79c3fc1dc98ce16299dc244/src/components/UserAndAdminDetails.vue

6.3. Prikaz, testiranje i mogućnosti poboljšanja aplikacije

Nakon što se prikazala i objasnila logika većeg dijela aplikacije, u nastavku će se prikazati sučelje same aplikacije te će se kroz testiranje određenih funkcionalnosti provjeriti njena ispravnost i očekivano ponašanje. Ukoliko se aplikacija pokreće na drugom računalu potrebno je imati instalirane sve pakete i tehnologije koje su spomenute u radu kao što su Node.js, *json-server,* Vue CLI, ali i ostale pakete iz *package.json* datoteke koje nisu navedene. Za instalaciju paketa potrebno je unutar terminala doći do korijenskog dijela aplikacije te izvršiti naredbu:

```
npm install
```

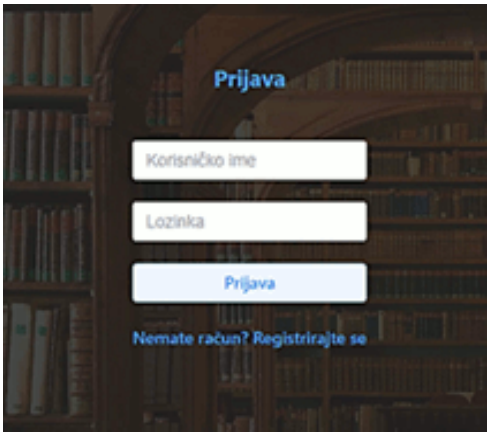
Nakon što su svi paketi preuzeti, u istom terminalu pokreće se Node.js aplikacija i skripte definirane u paketu, pomoću naredbe:

```
npm run dev
```

Zatim je potrebno pokrenuti još jedan terminal u kojemu se pokreće *json-server* odnosno lažna baza podataka i API koji će komunicirati s prednjim dijelom aplikacije, a to je moguće pokrenuti unutar korijenskog dijela aplikacije pomoću naredbe:

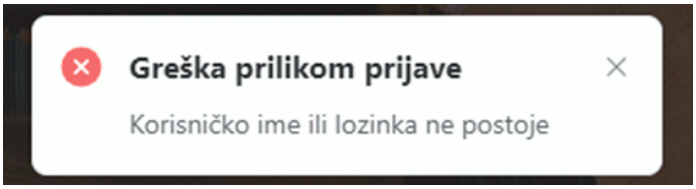
```
json-server --watch database.json
```

Nakon što su paketi preuzeti te su aplikacija i baza podataka uspješno pokrenuti, u pregledniku se učitava prva stranica aplikacije. Prilikom pokretanja sučelja za određenu ulogu, u aplikaciji je najprije potrebno izvršiti prijavu tijekom koje sustav provjerava podatke i učitava novu stranicu na osnovu upisanih podataka (Slika 3).



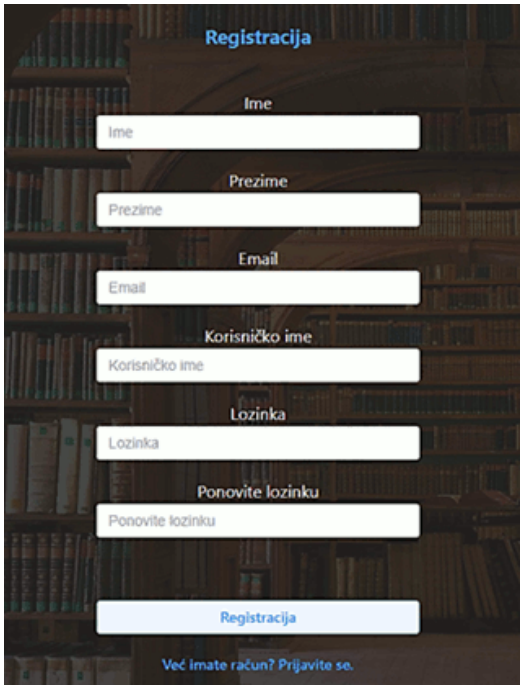
Slika 3. Prikaz sučelja aplikacije za prijavu korisnika/administratora

Ukoliko korisnik nije registriran, potrebno je najprije izvršiti registraciju jer u suprotnom prijava neće biti uspješna. Kako bi se ovaj dio testirao, unijeti su neispravni podaci te je sustav vratio obavijest o grešci čime je ovaj test uspješno prošao (Slika 4).



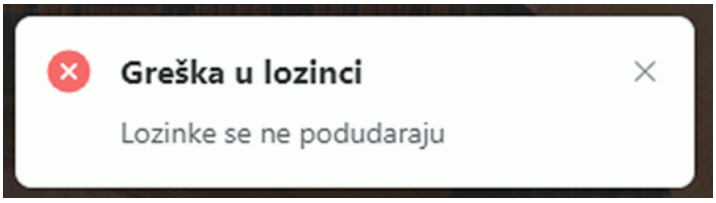
Slika 4. Prikaz obavijesti koja se javlja upisivanjem pogrešnog imena ili lozinke

Dakle, najprije je potrebno ući na stranicu za registriranje korisnika i registrirati se s ispravnim podacima. Ovdje je također postavljeno nekoliko pravila kao npr. da lozinka ne smije biti prekratka te da mora sadržavati barem jedan znak i sl.



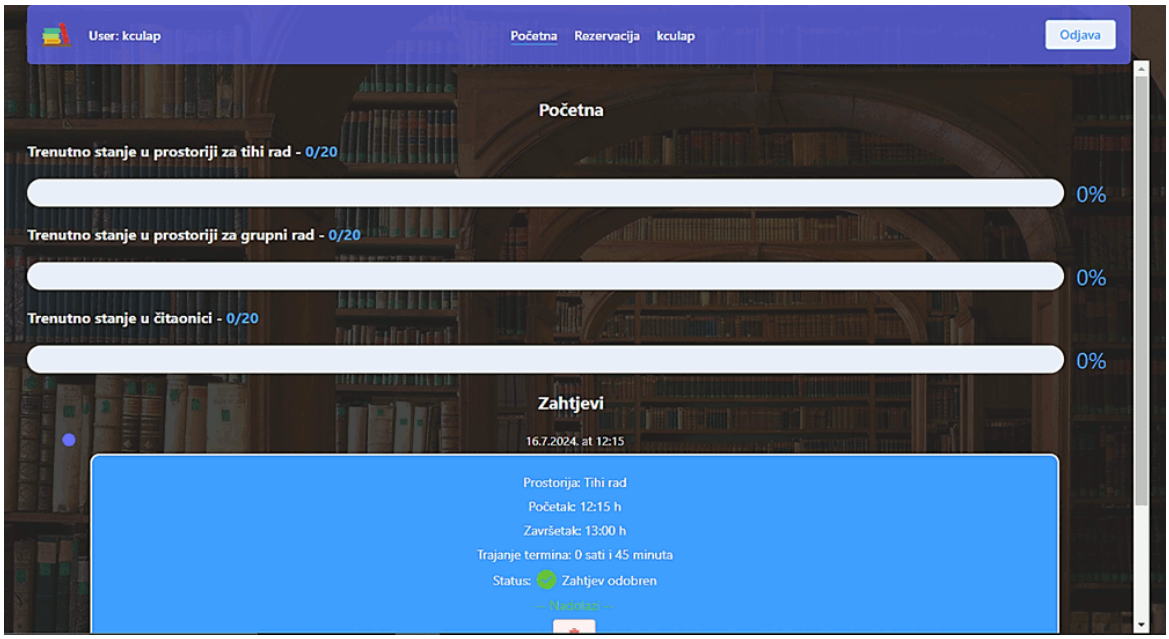
Slika 5. Prikaz registracije

Forma podržava osnovna pravila prilikom testiranja pa ukoliko se unese pogrešan format ili npr. ukoliko se unutar polja za ponovno upisivanje lozinke upiše drugačija lozinka, sustav će vratiti obavijest ovisno o situaciji.



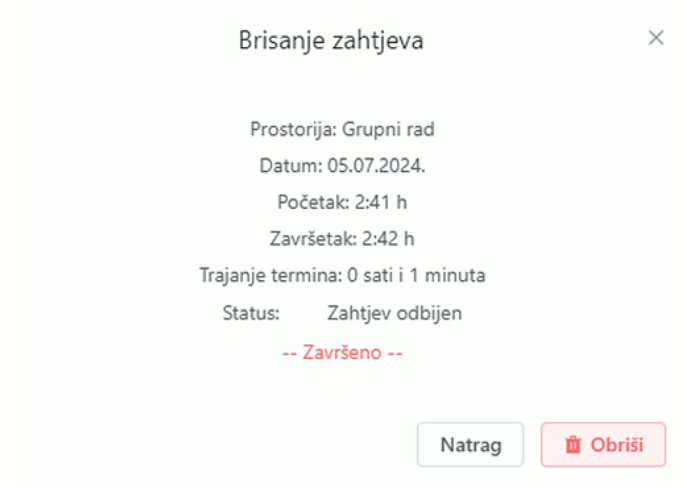
Slika 6. Prikaz greške u lozinci

Nakon što se korisnik uspješno prijavi, najprije mu se prikazuje početna stranica koja sadrži podatke o trenutnoj zauzetosti u prostorijama prikazane u progresnoj traci i postotcima (Slika 7). Ispod toga nalaze se svi poslani zahtjevi za rezervacijama od strane korisnika.



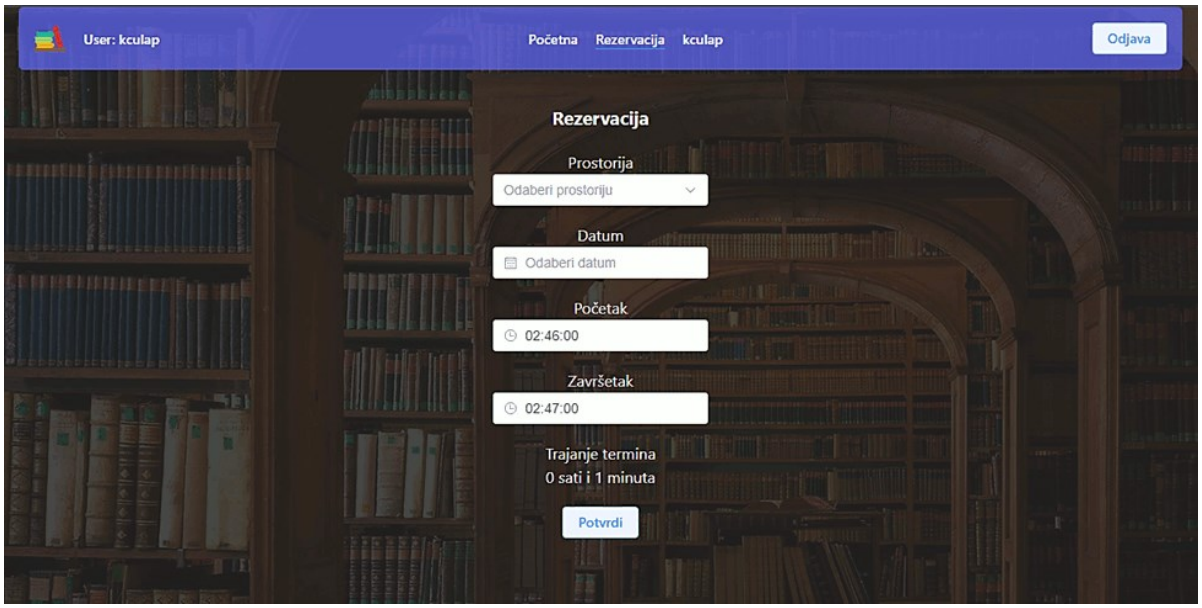
Slika 7. Prikaz početne stranice

Poslani zahtjevi prikazani su u karticama te prikazuju detalje kao što su odabrana prostorija, početak, kraj i ukupno trajanje termina te na kraju status koji prikazuje je li zahtjev još uvijek na čekanju, odbijen ili prihvaćen. Također nalazi se ikona za brisanje zahtjeva pa klikom miša na ikonu korisniku se otvara modul s ponovljenim podacima te opcijama da zatvori ili obriše zahtjev (Slika 8).



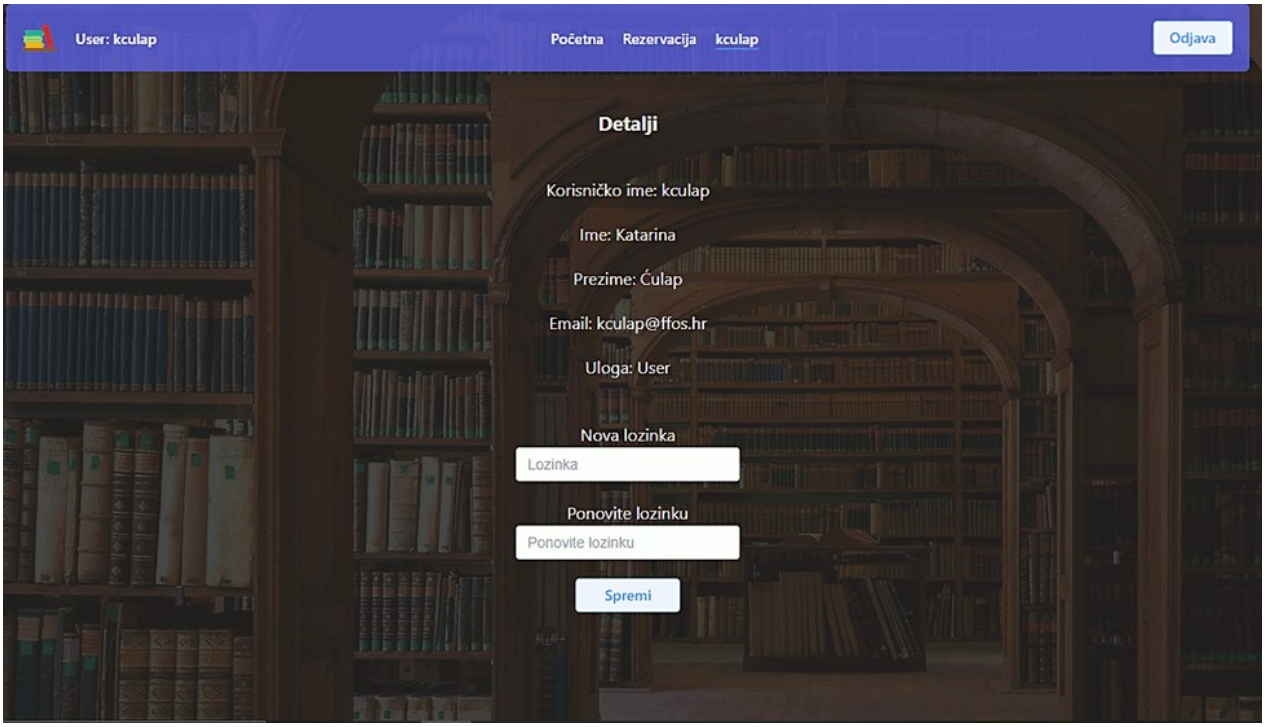
Slika 8. Prikaz modula unutar kojega je moguće obrisati zahtjev

U nastavku navigacijske trake nalazi se stranica Rezervacija koja prikazuje podatke za ispunjavanje koji su potrebni rezervaciju kao što su odabir prostorije, datum te dolazak i odlazak iz prostorije (Slika 9).



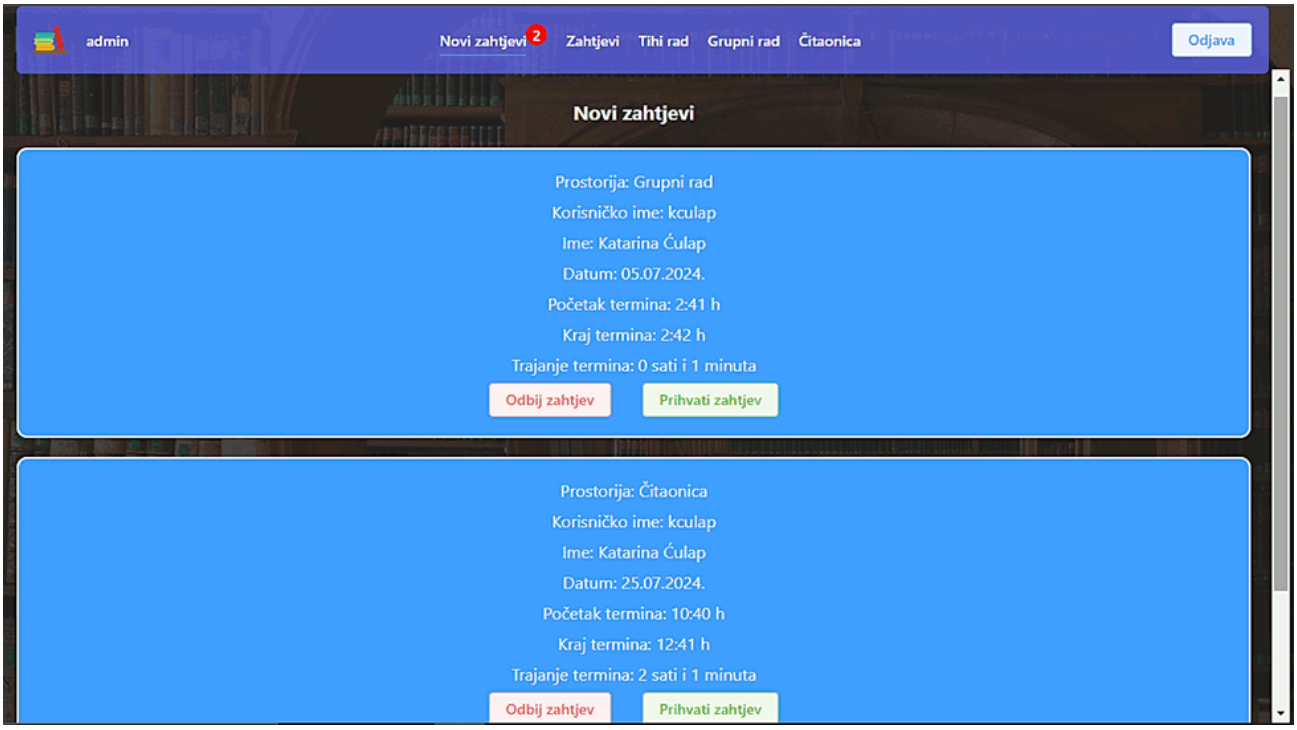
Slika 9. Prikaz forme Rezervacija

U nastavku navigacije nalazi se profil korisnika te klikom na njega prikazuju se osnovni podaci koje je korisnik upisao prilikom registracije (Slika 10). Također u ovom dijelu aplikacije korisniku je ponuđeno da promijeni lozinku. Ova funkcionalnost je testirana te se lozinka uspješno promijenila i koristila za sljedeću prijavu, a ukoliko se ponovi kriva lozinka korisniku se prikazuje obavijest o tome.



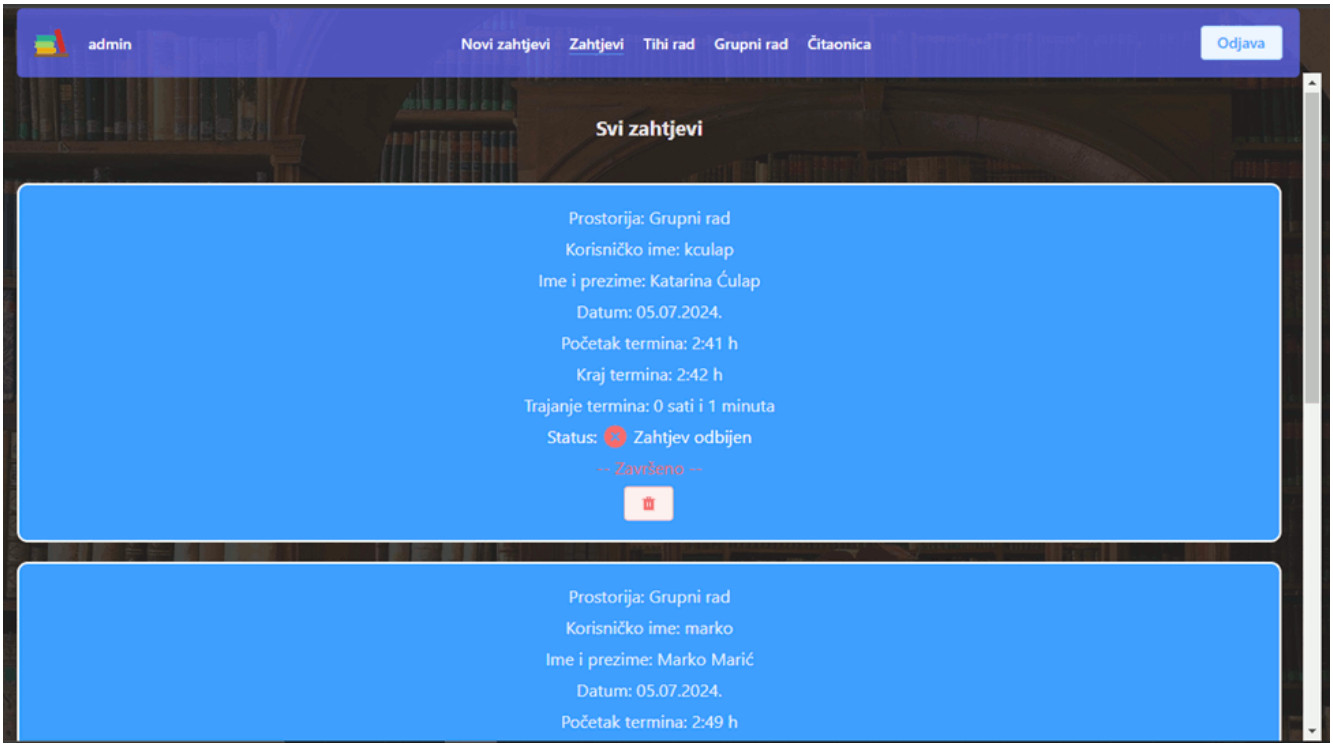
Slika 10. Prikaz profilne stranice s podatcima u korisniku

Ovim dijelom završava se dio s prikazom za korisnika te se prelazi na sučelje koje vidi administrator nakon prijave za svojim podacima. Kao prva stranica koju administrator vidi je stranica Novi zahtjevi gdje se prikazuju svi novi zahtjevi koji čekaju odobrenje (Slika 11).



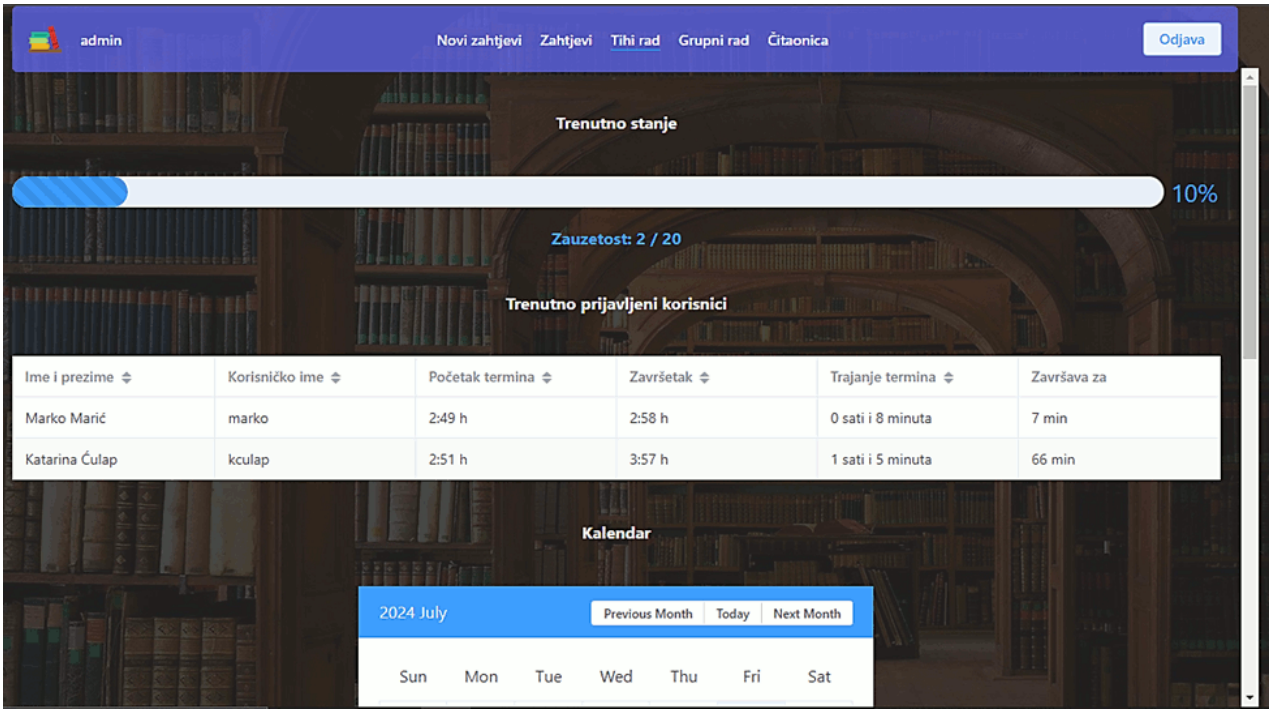
Slika 11. Prikaz početne stranice administratora – Novi zahtjevi

Na drugoj stranici u navigaciji odnosno na stranici Zahtjevi nalaze se svi odobreni ili odbijeni zahtjevi koje je također moguće trajno ukloniti sa stranice pomoću ikone za brisanje (Slika 12).



Slika 12. Prikaz stranice Zahtjevi unutar sučelja administratora

U navigacijskoj traci još se nalaze i sve tri prostorije za koje korisnici mogu rezervirati svoj dolazak. Svaka je dizajnirana na jednak način gdje se prikazuje trenutno stanje zauzetosti prostorije na kojoj je administrator pozicioniran te tablicu koja prikazuje korisnike koji su trenutno u knjižnici. Također osim trenutnog stanja na ovim stranicama nalazi se i kalendar u kojemu je moguće provjeriti zauzetost prostorije za određeni dan i provjeriti detalje o korisnicima koji imaju rezerviran termin za taj dan (Slika 13).



Slika 13. Prikaz stranice Tihi rad

Na kraju, administrator također ima isto sučelje za profil kao i korisnik te može na jednak način promijeniti lozinku. Osim toga i korisnik i administrator na istom mjestu u desnom kutu imaju gumb (*eng. button)* Odjava putem kojeg se izvodi odjava iz aplikacije. Pregledom kompletne aplikacije te testiranjem svih funkcionalnosti može se reći da aplikacija ispravno izvodi sve radnje. Iako se radi o jednostavnoj aplikaciji koja ne koristi veliki broj podataka, može se reći da ova jednostranična aplikacija zadovoljava uvjete za optimalno korisničko iskustvo. No, u svakom slučaju uvijek postoji mogućnost napretka i poboljšanja korisničkog iskustva pa samim time identificirane su i mogućnosti i dodaci koji bi mogli poboljšati ovu aplikaciju. Neke od tih mogućnosti uključuju optimizaciju postojećih značajki te dodavanje novih značajki, poboljšanje korisničkog sučelja, uvođenje sigurnosnih protokola za autentifikaciju i autorizaciju korisnika itd. što će se dodatno raspraviti u sljedećem poglavlju.

7. Rasprava

Prilikom proučavanja literature dobio se dublji uvid u koncepte i principe jednostranične aplikacije. Iako je ovaj model već duže vremena prisutan te velik broj aplikacija već koristi jednostraničnu arhitekturu, uključujući i neke od danas najpopularnijih aplikacija, ovakav model se neprekidno razvija, posebno u smjeru korisničkog iskustva. Može se reći da jednostranične aplikacije doživljavaju svoj stalni razvoj zahvaljujući pojavi novih tehnologija koje olakšavaju njihovu primjenu i otvaraju nove mogućnosti za poboljšano korisničko iskustvo. U drugom poglavlju ovog rada kroz povijesni pregled navedene su se neke od uočenih tehnologija koje su pridonijele razvoju jednostraničnih aplikacija i poboljšanju korisničkog iskustva te se većina njih i danas koristi u razvijenijem obliku. No, ono što je u najvećoj mjeri doprinijelo zadržavanju i povećanju broja jednostraničnih aplikacija je pojava radnih okvira koji su omogućili razvoj složenih jednostraničnih aplikacija uz relativno jednostavan pristup. U poslovnom području informacijske tehnologije uz dobro znanje određenog programskog jezika, najčešće je traženo i poznavanje nekog od radnih okvira. Stoga, za primjer aplikacije koja će potkrijepiti sve teorijske koncepte najprije je bilo potrebno odabrati pomoću kojeg radnog okvira će se izraditi aplikacija. Među najpopularnijim okvirima su Angular, React i Vue te je odabir bio između njih, a kako bi se odabrao odgovarajući okvir bilo je potrebno proučiti temeljne značajke i mogućnosti koje će poslužiti prilikom razvoja aplikacije. Kao najprikladniji okvir za izradu ove aplikacije odabran je Vue s obzirom na njegovu jednostavnost te niz mogućnosti koje nudi za razvoj i nadogradnju ove jednostavne jednostranične aplikacije. Vue se pokazao kao vrlo fleksibilan alat što je omogućilo jednostavniji razvoj i upravljanje stanjima unutar aplikacije. No, unatoč očitim prednostima te pretpostavci da ovaj okvir neće stvarati veće izazove pri razvoju aplikacije, s obzirom na to da je još uvijek relativno nov u usporedbi s Angularom i Reactom, manje je dostupnih izvora i podrške. Manji broj dostupnih primjera za specifične probleme zahtijevao je dodatni napor u istraživanju i testiranju. Također, kako bi se osiguralo upravljanje stanjima u aplikaciji bilo je potrebno istražiti i odabrati odgovarajući alat za tu svrhu. U ovoj aplikaciji, umjesto Vuex-a koji je poznati alat za upravljanje stanjima u Vue okruženju, uočeno je da postoji noviji alat kojeg je Vue razvio pod nazivom Pinia te se pokazao jednostavan za strukturiranje i upravljanje stanjem. Osim upravljanjem stanjem, bilo je potrebno omogućiti komunikaciju s poslužiteljem, a u ovoj aplikaciji poseban naglasak je stavljen na korištenje API-a i *json-server* alata. Pomoću *json-server* alata omogućena je uspostava tzv. lažnog API-a koji je simulirao pravi *backend* poslužitelj te se na taj

način izbjegla potreba za kreiranjem složene baze podataka za pohranu svih podataka. Na ovaj način omogućeno je testiranje i pokretanje aplikacije simuliranom bazom podataka prije nego što se potencijalno u budućnosti izradi prava baza podataka kao eventualna nadogradnja aplikacije. Postavljanje *json-server* API-a nije bilo zahtjevno te je omogućio brzi razvoj i testiranje. Međutim, iako je ovaj alat vrlo koristan za jednostavne aplikacije ili testiranje složenih aplikacija prije uključivanja pravih podataka, postojala su određena ograničenja koja su zahtijevala dodatne korake i zamjenske alate. Najveći izazov bio je u tome što podaci pohranjeni u JSON datoteci nisu trajni i mogu se izgubiti prilikom ponovnog pokretanja poslužitelja. Također, ovaj alat ne podržava autentifikaciju i autorizaciju stoga su svi podaci otvoreni i dostupni svakome tko ima pristup API-u što može predstavljati određena ograničenja i sigurnosne rizike u stvarnom okruženju. Unatoč tome, za potrebe demonstracije jednostranične arhitekture, ovaj pristup je bio prihvatljiv. U sljedećoj fazi bilo je potrebno razmisliti o potrebnim funkcionalnostima i sadržaju aplikacije kako bi se izradile odgovarajuće komponente. Cilj je bio implementirati nekoliko komponenti i funkcionalnosti kako bi se prikazalo ponašanje više sadržaja unutar jednostranične aplikacije te kako bi se demonstrirale različite metode kojima se može poboljšati korisničko iskustvo i interakcija s aplikacijom. U ovom dijelu bitno je organizirati navigaciju aplikacije kako bi se korisnik mogao jednostavno kretati kroz različite „stranice“ odnosno dijelove unutar same aplikacije. To se postiglo postavljanjem tzv. ruti koje jednostavno omogućuju prijelaze između različitih prikaza aplikacije bez ponovnog učitavanja cijele aplikacije što je osnovna svrha jednostranične arhitekture. Zbog jednostavnije organizacije, s obzirom da je sučelje za administratore i korisnike odvojeno, komponente su se podijelile unutar dva osnovna direktorija: administrator i korisnik te je ovaj pristup omogućio jasnu strukturu i bolju preglednost koda. Počevši s komponentama za prijavu i registraciju, unutar svake komponente najprije je dodan *template* u kojemu se oblikovala struktura i izgled korisničkog sučelja. Nakon što su dodani potrebni elementi, u sljedećoj fazi implementirane su metode i svojstva za svaku komponentu. Za privremenu pohranu podataka korištena je metoda lokalne pohrane (*eng. Local storage*) na strani klijenta, što se pokazalo kao jednostavno rješenje i dobra privremena zamjena za kompleksne baze podataka. Dodatno, korištenjem alata za kriptiranje određeni podaci su zaštićeni, čime je značajno poboljšana njihova sigurnost. No, važno je napomenuti da će u slučaju daljnjeg razvoja aplikacije biti potrebno implementirati pravu bazu podataka jer lokalna pohrana ne osigurava šifriranje svih podataka, pa neki osjetljivi podaci mogu biti vidljivi svima. Nadalje, osim alata za postavljanje

različitih funkcionalnosti, Vue nudi razne pakete za poboljšanje korisničkog sučelja aplikacija. Istraživanjem različitih primjera istaknuo se Element Plus paket koji je upotrijebljen za jednostavno dodavanje UI (*User Interface)* komponenti kao što su forme, tablice, dijalozi i druge komponente koje su se mogle vidjeti u prethodnim poglavljima u prikazu aplikacije. Korištenjem svih navedenih paketa i alata naglašava se važnost istraživanja novih trendova i upotrebe modernih alata kako bi se osiguralo vrhunsko korisničko iskustvo aplikacije. S obzirom da je naglasak na korisničkom iskustvu, pored odgovarajuće logike i funkcionosti, bilo je potrebno dodatno pojednostaviti i vizualno poboljšati izgled aplikacije kako bi se poboljšalo ukupno korisničko iskustvo. Sva dodatna poboljšanja imala su cilj olakšati korisniku kretanje aplikacijom te mu omogućiti pristupačno sučelje. Koristeći teorijska saznanja i praktična rješenja postigao se krajnji rezultat odnosno jednostranična aplikacija s brzim korisničkim sučeljem koje se prilagođava različitim širinama zaslona. Ovom aplikacijom potvrđuje se kako jednostranična arhitektura, zajedno s prednostima koje pružaju moderni radni okviri poput Vue-a, može značajno poboljšati razvoj aplikacija i podići ga na višu razinu funkcionalnosti i korisničkog iskustva.

8. Zaključak

Razvoj jednostraničnih aplikacija predstavlja značajan iskorak u modernom mrežnom razvoju, nudeći mnogobrojne prednosti koje značajno poboljšavaju korisničko iskustvo i optimiziraju svojstva aplikacija. Za razliku od tradicionalnih ili tzv. višestraničnih aplikacija, jednostranične aplikacije omogućuju izrazito bržu i bolju interakciju korisnika s aplikacijom. Razlog tome ponajviše leži u tome što višestranične aplikacije, a posebno kompleksne aplikacije koje imaju puno sadržaja, učitavaju svaku stranicu zasebno što može trajati duže od očekivanog. Jednostranične aplikacije umjesto ponovnog učitavanja stranica prilikom svakog korisničkog zahtjeva imaju arhitekturu koja omogućava dinamičko prikazivanje sadržaja jer su podaci već učitani s poslužitelja što smanjuje trajanje prikazivanja novog sadržaja. Proučavajući literaturu može se reći kako korisničko iskustvo ima sve veću ulogu u razvoju aplikacija s obzirom na sve zahtjevnije korisnike i moderne tehnologije koje se pojavljuju. Neočekivani prekidi, predugo učitavanje, ne prilagodba različitim širinama zaslona i neadekvatno korisničko sučelje najveći su primjeri koji izazivaju negativno korisničko iskustvo. Takvi problemi mogu dovesti do nezadovoljstva korisnika, smanjenja povjerenja u aplikaciju i, u krajnjem slučaju, gubitka korisnika. Zbog toga je važno usmjeriti pažnju na optimizaciju značajki, interaktivnosti, ali i jednostavnosti korištenja korisničkog sučelja kako bi se osiguralo pozitivno i zadovoljavajuće iskustvo za sve korisnike. S obzirom da jednostranične aplikacije pružaju osjećaj bržeg i jednostavnijeg rada aplikacije može se reći da zbog svojih karakteristika mogu značajno poboljšati korisničko iskustvo i ponuditi pouzdana rješenja. Ovaj rad pružio je uvid u prednosti koje jednostranična aplikacija pruža, ali kao što se već moglo zaključiti, razvoj aplikacije s jednostraničnom arhitekturom ne mora uvijek biti najbolje rješenje te se u određenim situacijama višestranične aplikacije mogu pokazati kao bolji izbor. Kroz ovaj rad, prikazano je kako jednostranična aplikacija može biti implementirana koristeći Vue, kao jedan od modernih JavaScript okvira koji omogućava jednostavno kreiranje komponenti te je lagan za održavanje i nadogradnju. Iako je Vue trenutno manje korišten unutar zajednice, za razliku od drugih radnih okvira spomenutih u radu kao što su Angular i React, pokazao se kao okvir koji pruža brojne mogućnosti za olakšan razvoj jednostraničnih aplikacija. Može se reći kako je Vue prikladan za početnike te nije bilo većih izazova prilikom izrade aplikacije.

Knjižnice kao zajednica koja se, kao i većina drugih, sve više okreće tehnologiji i modernim rješenjima koje ona pruža, imaju priliku unaprijediti svoje usluge, olakšati pristup informacijama i poboljšati korisničko iskustvo. Aplikacija koja je opisana u radu izrađena je s ciljem da se prouče temeljni koncepti jednostranične aplikacije, ali i da se proširi ideja o digitaliziranju usluga knjižnica. Zaključno, razvoj jednostranične aplikacije za upravljanje rezervacijama unutar knjižnice pokazao je kako moderne tehnologije mogu transformirati način na koji korisnici integriraju s digitalnim sadržajem. Ova aplikacija predstavlja samo jedan primjer kako se prednosti jednostranične aplikacije mogu iskoristiti za izgradnju učinkovitih i korisnički orijentiranih rješenja.

Na kraju svega napisanog može se zaključiti da jednostranične aplikacije omogućuju knjižnicama, i drugim institucijama, da efikasnije isporuče svoje usluge i bolje odgovore na potrebe svojih korisnika te tako osiguraju da ostanu relevantne i konkurentne u digitalnom dobu. Pojavom novih tehnologija posljednjih desetljeća, jednostranične aplikacije doživjele su još veću popularnost, stoga kontinuiranim razvojem i usavršavanjem, ne sumnja se da će jednostranične aplikacije nastaviti igrati važnu ulogu u evoluciji tehnologija i korisničkih iskustava.

9. Popis literature

1. Acceptance of Single Page Application in Today's World // Great Learning, 2022. URL:https://www.mygreatlearning.com/blog/acceptance-of-single-page-application-in-todays-world/ (2023-07-03)
2. Awati, Rahul. Adobe Flash // TechTarget, 2022. URL: https://www.techtarget.com/whatis/definition/Flash (2024-07-06)
3. BasuMallick, Chiradeep. What Is a Single-Page Application? Architecture, Benefits, and Challenges // Spiceworks, 2022. URL: https://www.spiceworks.com/tech/devops/articles/what-is-single-page-application/ (2023-07-03)
4. Cocca, Germán. Rendering Patterns for Web Apps – Server-Side, Client-Side, and SSG Explained // freeCodeCamp, 2023. URL: https://www.freecodecamp.org/news/rendering-patterns/ (2024-07-05)
5. Ćwik, Przemyslav. Single-page application: how SPA works and how it differs from MPA. // Kiss Digital, 2021. URL: https://kissdigital.com/blog/single-page-application-how-spa-works-and-how-it-differs-from-mpa#:~:text=The%20first%20single-page%20applications,used%20to%20create%20the%20frontend (2023-07-27)
6. Demystifying Rendering Modes - SPA/CSR, SSR, SSG, OMG // Wolfpack digital, 2022. URL: https://www.wolfpack-digital.com/blogposts/spa-csr-ssr-ssg-demystifying-rendering-modes (2024-07-05)
7. Fink, Gil; Flatow, Ido. Pro Single Page Application Development: Using Backbone.js and ASP.NET. // Apress Berkeley, CA, 2014., str. 13 URL: https://link.springer.com/book/10.1007/978-1-4302-6674-7#bibliographic-information (2023-07-03)
8. Freda, Anthony. What Is the MD5 Hashing Algorithm and How Does It Work? // Avast, 2022. URL: https://www.avast.com/c-md5-hashing-algorithm (2024-07-02)

9. Gerchev, Ivaylo. Leveraging Pinia to simplify complex Vue state management //LogRocket: Frontend Analytics, 2024. URL: https://blog.logrocket.com/complex-vue-3-state-management-pinia/ (2024-07-02)
10. History of JavaScript // GeeksforGeeks, 2024. URL: https://www.geeksforgeeks.org/history-of-javascript/ (2024-07-05)
11. Jaman, Shifat. Everything you need to know about "Single-page-application"// Medium, 2019. URL: https://shifat-jaman.medium.com/single-page-application-everything-you-need-to-know-6f00d87e5130 (2023-07-03)
12. Kanade, Vijay. What Is AJAX (Asynchronous JavaScript and XML)? Meaning, Working and Applications: AJAX enhances a website's performance and overall user experience. // Spiceworks, 2023. URL: https://www.spiceworks.com/tech/devops/articles/what-is-ajax/ (2023-07-03)
13. Makinde, Felix. Node.js vs NPM: Understanding the Differences // HostAdvice, 2024. URL: https://hostadvice.com/blog/web-hosting/node-js/node-js-vs-npm/ (2024-07-01)
14. Munawar, Shahzaib. Single Page Applications (SPAs). Most Discussed Pros & Cons. // Novateus, 2022. URL: https://novateus.com/blog/single-page-applications-spas-most-discussed-pros-cons/ (2023-07-27)
15. Patel, Bhaval. 8 Top Single-Page Application Frameworks for Web App Development // Space-O Technologies, 2024. URL: https://www.spaceotechnologies.com/blog/single-page-application-frameworks/ (2024-07-01)
16. Sharma, Riya. Mock APIs Using JSON Server // Medium, 2023. URL: https://medium.com/@theriyasharma24/create-mock-apis-using-json-server-for-seamless-development-4a6ba03ef148 (2024-07-01)
17. What is a Single Page Application? Pros and Cons the SPA Technology // Huspi, 2022. URL: https://huspi.com/blog-open/definitive-guide-to-spa-why-do-we-need-single-page-applications/ (2023-07-27)

18. Wieckowska, Sara; Bracisiewicz, Mateusz. PWA vs. MPA vs. SPA – What's the Best Choice for Your App? // Neoteric, 2022. URL: https://neoteric.eu/blog/pwa-vs-mpa-vs-spa-whats-the-best-choice-for-your-app/ (2023-07-27)