Hotwire pristup razvoju Ruby on Rails web aplikacija

Jović, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

HOTWIRE PRISTUP RAZVOJU RUBY ON RAILS WEB APLIKACIJA

Završni rad

Marko Jović

Osijek, 2021.

SADRŽAJ

1. UVOD

Ovim završnim radom prikazat će se prednosti Hotwire pristupa mrežnim (engl. *web*) aplikacijama. Primjer koji se koristi za prikazivanje istog je mrežna (engl. *web*) aplikacija s tematikom glazbenih bendova. Korisnici aplikacije pri prvom registriranju odnosno posjetu aplikacije definiraju svoj status. Odabiru pripadaju li skupini klijenata koji imaju želju unajmiti odnosno rezervirati određen glazbeni bend ili oni sami tvore jedan glazbeni bend koji stavlja svoje usluge na raspolaganje ostalim klijentima. Svi korisnici spremaju se u bazu podataka zajedno sa svojim podatcima kao što su e-pošta, lozinka te izbor skupine kojoj pripadaju. Korisnici grupe "klijenti" prilikom potrage za glazbenim bendom imaju mogućnost otvaranja padajućeg izbornika i pregleda svih bendova koji su na raspolaganju. Nakon odabira benda korisnik odabire jedan od dostupnih datuma prikazanih na kalendaru koji je realiziran pomoću jQuery izborom datuma (engl. *date picker*). Kada je korisnik ispunio sve što se od njega traži on potvrđuje rezervaciju tako da šalje zahtjev bendu koji ukoliko želi suradnju mora prihvatiti zahtjev te tako dati do znanja klijentu kako njihova suradnja može započeti. Uz sve navedeno tu se još nalazi i opcija pisanja napomene između korisnika. Glavna zadaća ovog rada je prikazivanje razlike između 'monolitnih' aplikacija i jednostranih aplikacija (engl. *Single Page Application, SPA*). Zapravo cilj je prikazivanje pogodnosti korištenja Hotwire-a zajedno s njegovim dodatcima Turbo i Stimulus koji čine osvježavanje aplikacije trenutnim odnosno prikazuju promjene korisniku u istom trenutku u kojemu ih i on sam čini. Programski jezik u kojemu je aplikacija realizirana je Ruby a radno okruženje kojim se koristi Rails. Uz to vidljiva je i upotreba HTML-a te CSS-a , jezika za izgradnju prednjeg kraja (engl. *front end*) aplikacije. Njihova upotreba nešto je slabija od Ruby-a jer se cilj završnog rada ne temelji na prednjem kraju (engl. *front end*) aplikacije. Bez obzira na količinu upotrebe stavki spomenutih u uvodu svaka od njih bit će obrađena u završnom radu.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je istražiti i opisati Hotwire pristup, istražiti i opisati razlike između 'monolit' mrežnih (engl. *web*) aplikacija i SPA (jednostrana aplikacija (engl. *single-page application)*) te utvrditi koje su prednosti i nedostatci ovakvog pristupa. Potrebno je istražiti i opisati sastavne dijelove Hotwire pristupa (Turbo i Stimulus), te ActionCable tehnologiju na kojoj se Turbo i Stimulus temelje. Praktični dio bazira se na mrežnoj (engl. *web*) aplikaciji pomoću koje će se demonstrirati prednosti Hotwire pristupa.

2. KONSTRUKCIJA I IZGLED APLIKACIJE

Kada je riječ o konstrukciji i izgledu aplikacije često korišteni jezici su HTMLi CSS. HTML zadužen je za prikazivanje sadržaja aplikacije korisniku te za samu konstrukciju aplikacije. Kao i u svemu tako i u mrežnim aplikacijama izgled tj. dizajn ima vrlo važnu ulogu. Većina korisnika prvo primjećuje izgled i prema tome određuju zanima li ih nešto ili ne. Zbog toga jezik poput CSS-a ima vrlo važnu ulogu. Često puta korisnik odluči dalje pretraživati određenu internet stranicu zbog znatiželje koja je izazvana dobrim dizajnom. U nastavku rada nalaze se kratka obilježja i segmenti dva prethodno navedena jezika!

2.1. HTML

Hyper text Markup Language puni je naziv jezika HTML. Većina dokumenata prikazanih na Internet preglednicima koriste HTML. Često, zapravo skoro uvijek dolazi u kombinaciji s jezicima koji ga nadopunjavaju, a neki od njih su CSS i JavaScript. Putem mrežnog (engl. *web*) servera ili putem lokalnih mjesta za pohranjivanje podataka preglednici primaju HTML dokumente te ih prikazuju kao multimedijske Internet stranice. HTML stranice se sastoje od blokova koji su zapravo elementi HTML-a. Postoji više vrsta elemenata koji se mogu prikazati na Internet stranici, kao što su tekst, slike ili interaktivni objekti. HTML-ova konstrukcija navedenih objekata omogućuje njihovo prikazivanje na stranici. Označavajući strukturnu semantiku za tekst kao što su odlomci, naslovi, citati i veze HTML nudi način stvaranja strukturiranih dokumenata . Svaki HTML element sastoji se od kutnih zagrada i sadržaja koji je namijenjen za prikazivanje. Postoje oznake koje direktno prenose sadržaj na stranicu . Neke od njih su <img> i <input>. S druge strane postoje oznake koje okružuju informacije o tekstu i imaju mogućnost pod sobom nositi još neke oznake. Podrazumijeva se da preglednici ne prikazuju kutne zagrade nego ih samo koriste kako bi prepoznavali radnju koju oznaka vrši .

Osnovna konstrukcija jedne Internet stranice pisane u HTML-u podijeljena je na tri dijela i naziva se još i “kostur stranice“ . Kostur stranice sastoji se od glave, tijela i podnožja. Dok se u glavi nalaze elementi kao što su navigacija ili naslov u ovom slučaju, u tijelu se većinom nalaze svi ostali elementi poput samog teksta stranice ili slika te ostalih interaktivnih objekata. Podnožje većinom sadrži informacije o kreatoru ili samoj mrežnoj (engl. *web*) stranici. Slika 2.1 prikazuje kostur jedne Internet stranice prema [1].

Linija ***Kod***

```
1:          <!DOCTYPE html>
2:              <head>
3:                    <title> My website </title>
4:              </head>
5:              <body>
6:                    <p>This is Peter</p>
7:                    <img src="user.img" alt="Simple user image">
8:              </body
9:              <footer> </footer>
10:         </html>
```

Slika 2.1 Primjer izgleda kostura Internet stranice.

2.1.1. Markup elementi

HTML označavanje sastoji se od nekoliko neizostavnih komponenti, uključujući one koje se zovu oznake, vrste podataka temeljene na znakovima i referencama identiteta. HTML oznake najčešće dolaze u parovima poput <h1> i </h1> dok neke oznake dolaze ne uparene poput oznake <img> koja u sebi sadrži oznake za otvaranje i zatvaranje elementa. Tekst koji se nalazi između <html> i </html> opisuje Internet stranicu dok tekst između oznaka <body> i </body> predstavlja sadržaj koji se vidi na stranici. Oznaka definira određenu sekciju elemenata koji se nalaze na Internet stranici. Također služe za jednostavnije upravljanje i uređivanje stranice. Pruža mogućnost definiranja sekcijskog ID-a (identifikacijska riječ) ili imenovanje klase sekcije kako bi se zasebno mogla urediti posebno ta sekcija elemenata . Slika 2.2 prikazuje grupiranje elemenata pod jednim nazivom klase prema [1].

Linija ***Kod***

```
1:          <div class = "Information">
2:              <p> This is an information about user! </p>
3:              <img src="user.img" alt="Simple user image">
4:          </div>
```

Slika 2.2 Primjer grupiranja elemenata pomoću klase.

2.1.2. Vrste podataka

HTML definira nekoliko vrsta podataka za sadržaj elemenata, kao što su podaci skripte i podaci u tablici stilova, te mnoštvo vrsta za vrijednosti atributa, uključujući ID-ove, imena, URI-eve, brojeve, jedinice duljine, jezike, deskriptore medija, boje, kodiranje znakova, datume i vremena itd. Svi ti tipovi podataka specijalizacije su znakovnih podataka prema [1].

2.2. CSS

Puni naziv jezika CSS je Kaskadne tablice stilova (engl. *Cascading Style Sheets)*. Riječ je o stilskom jeziku čija je uloga opis i sama prezentacija dokumenta odnosno Internet stranice napisane pomoću jezika HTML. Glavni razlog nastajanja ovog jezika je rasterećivanje HTML-a odnosno oslobađanje ga odgovornosti uređivanja samog dokumenta. S vremenom su se dodavale funkcije u sami HTML jezik ali nakon nekog vremena ustanovljeno je da to nema smisla te je zbog toga stvoren CSS. Njegov glavni zadatak je stil stranice , njen izgled i raspored prema [2].

2.2.1. Sintaksa

Postoji nekoliko pravila CSS-a i svako od njih se sastoji od selektora i deklaracijskog bloka. Kada je riječ o pisanju koda, stvara se poseban dokument sa inačicom ".css" . Unutar tog dokumenta se posebno uređuje svaki element HTML dokumenta na koji se odnosi CSS dokument tako da se piše prvo znak "." te nakon nje ime klase ili ID-a ovisno o tome koji element se želi uređivati. Kada se odabere element moguć je odabir posebnog segmenta kao što je odlomak ili slika. Slika 2.3 prikazuje nekoliko naredbi koji definiraju izgled naziva članka koji se nalazi pod klasom "naslovna".

Linija ***Kod***

```
1:          .naslovna h1{
2:              font-family: Lemon;
3:              color: rgba(255, 238,0);
4:              font-size: 20vh;
5:          }
```

Slika 2.3 Primjer koda u CSS-u.

2.2.2. Selektor

Služi za grupno uređivanje svih elemenata iste vrste podatka ili svih onih koji spadaju pod određenu klasu ili ID . Pomoću njega se utječe istovremeno na cijele grupe podataka odnosno elemenata.

- ID – jedinstveni element
- Klasa (engl. *class*) – može obuhvaćati više elemenata

Potrebno je spomenuti i pseudo-klase. To su klase koje pružaju mogućnosti opisivanja informacija koje nisu dostupne u DOM-u poput naredbe lebdjeti (engl. *hover*) koja identificira sadržaj samo ako korisnik drži pokazivač nad njim prema [2].

2.2.3. Deklaracijski blok

Predstavljaju ga vitičaste zagrade unutar koji se nalazi deklaracija. Svaka deklaracija sastoji se od dvije stvari a to su dvotočka i vrijednost. Deklaracije se odvajaju znakom " ; ". Riječi poput "sredina" (engl. *center*) ili "naslijedi" (engl. *inherit*) mogu predstavljati vrijednost. Postoje i brojčane vrijednosti poput 200 (veličina fonta), 300px (300 piksela) i slično. Prethodno su definirane vrijednosti boja kao na primjer crvena(engl. *red*) za crvenu boju prema [3].

2.3. Vue

Vue je jedan od JavaScript okvira (engl. *framework*). Njegova arhitektura model-view-view-model koristi se za razvoj SPA. Razvio ga je Evan You i koristeći svoju 'razdvojenost', dopušta Vue developerima da postupno izrađuju korisnička sučelja. Prema Github-u najzastupljeniji je JavaScript okvir. Vue također sadrži dvosmjerno vezivanje i komponente te koristi virtualni DOM. Međutim, Vue-ov glavni nedostatak je progresivni dizajn. Vue je osmišljen kako bi omogućio developerima da postupno migriraju postojeće projekte u okvir, premještajući značajke jednu po jednu, a ne sve odjednom prema [14].

2.4. React

React je JavaScript biblioteka koja se koristi za izradu korisničkog sučelja ili njegovih komponenti. React JavaScript nudi povećanu fleksibilnost korištenjem 'komponenti', kratkih, izoliranih dijelova koda koje programeri mogu koristiti za stvaranje složenijih logičkih i korisničkih sučelja. React stupa u interakciju s HTML dokumentima putem 'virtualnog DOM -a', koji je kopija stvarnog DOM -a, ali svi su elementi predstavljeni kao JavaScript objekti prema[14].

2.4.1. Razlika između Vue-a i React-a

Iako React.js i Vue.js imaju mnogo sličnosti, postoji nekoliko razlika između njih koje imaju značajan utjecaj na ono za što je svaki od njih najprikladniji. Glavna razlika leži u metodama koje koristi Vue i onima koje koristi React za iscrtavanje sadržaja na DOM -u. Vue koristi HTML predloške i JSX, dok React koristi samo JSX, proširenja koja omogućuju umetanje HTML-a izravno u JavaScript kod. JSX može ubrzati i pojednostaviti veće, složenije zadatke ali nedostatak mu je što također može zakomplicirati ono što bi trebao biti lak zadatak.

Osim toga, osnovni elementi Reacta su komponente, manipulacija DOM -om i upravljanje stanjem komponenti. Sve ostalo razvija i podržava zajednica prema [14].

2.5. Ruby on Rails

Ruby on Rails ili također poznat kao Rails radno okruženje za razvoj internet aplikacija na strani poslužitelja napisan je u programskom jeziku Ruby, a razvio ga je David Heinemeier Hansson pod MIT licencom. Podržava arhitekturu MVC (model-pogled-kontroler (engl. *model-controler-view*)) koja pruža zadanu strukturu za baze podataka, internet stranice i mrežne (engl. *web*) usluge, također koristi internet standarde poput JSON ili XML za prijenos podataka te HTML, CSS i JavaScript za korisničko sučelje. Naglašava uporabu drugih dobro poznatih obrazaca softverskog inženjeringa i paradigmi poput:

- DRY načelo - ne ponavljaj se (engl. *dont repeat yourself*)- načelo koje reducira ponavljanje istog koda ili informacija
- CoC načelo - konvencija umjesto konfiguracije (engl. *Convention Over Configurati*on) - Daje mnoga mišljenja o najboljem načinu rada u mrežnoj (engl. *web*) aplikaciji.

Od 2004. godine pa na dalje Ruby je brzo postao jedan od najmoćnijih i najpopularnijih alata za izradu dinamičnih mrežnih (engl. *web*) aplikacija . Za veliki dio uspjeha zadužen je jednostavan i kompaktan dizajn. Ruby on Rails prvi je put objavljen u srpnju 2004., ali do veljače 2005. nije dijelio prava. U kolovozu 2006. isporučen je Ruby on Rails s Mac OS X v10.5 “Leopard”. Najnovija verzija Ruby on Rails (Rails 5.0.1) objavljena 21. prosinca 2016. Akcijski kabel (engl. *Action cable*), Turbolinks 5 i API način predstavljeni su u ovoj verziji prema [4].

2.5.1. Obilježja

- Arhitektura modela – pogleda -kontrolera - Ruby on Rails koristi MVC arhitekturu i sadrži tri komponente, tj. model, pogled i kontroler. Model se koristi za održavanje odnosa između objekta i baze podataka, pogledi (engl. *views*) su predlošci koji se koriste za izgradnju podataka korisnika za mrežne (engl. *web*) aplikacije, a kontroler se koristi za spajanje modela i prikaza. MVC se općenito koristi za razvoj korisničkih sučelja koja dijele podatke na tri međusobno povezane komponente tako da se mogu odvojiti unutarnji prikaz informacija od načina na koji se prezentiraju i dobivaju od korisnika prema [5].
- Aktivni zapis (engl. *Active Record*) - okvir za aktivne zapise uključen je u Ruby on Rails. Vrlo korisna biblioteka koja razvojnom programeru omogućuje da u bazi podataka osmisli interaktivne upite.
- Ugrađeno testiranje - Ruby on Rails nudi vlastiti skup testova koji će se izvoditi na korisnikovom kodu. Uštedjet će vrijeme i trud.
- Programski jezik - sintaksa Ruby on Rails-a je jednostavna jer je sintaksa programskog jezika Ruby bliska engleskom, pa je uvijek lakše strukturirati svoje razmišljanje i zapisati ga u kod.
- Meta programiranje - Ruby on Rails koristi tehniku meta programiranja za pisanje programa.
- Skela (engl. *Scaffold*) - Ruby on Rails pruža značajku skele (engl. *Scaffold*) u kojoj je razvojnom programeru dopušteno definirati kako radi baza podataka. Nakon definiranja rada baze podataka, okvir automatski generira traženi kod prema zadanoj definiciji. Ova tehnika automatski stvara sučelja prema [4].

2.5.2. Prednosti i mane

Neke od prednosti Ruby on Rails-a su alati, biblioteke i automatizacija testiranja. Alati pomažu isporučiti više značajki u manje vremena. Pomoću biblioteka korisniku se nudi modul treće strane za gotovo sve što mu je potrebno (dragulj (engl. *gem*)). Ruby-eva velika zajednica testira automatizaciju i testiranje prema [4].

Kao i prednosti tako Ruby posjeduje i svoje mane, neke od njih su brzina izvođenja, nedostatak fleksibilnosti i dokumentacija. Brzina izvođenja Ruby on Rails-a je spora u usporedbi s Node.Js-om i Golang-om. Poznato je da je Ruby on Rails idealan za standardne mrežne (engl. *web*) aplikacije zbog svoje velike ovisnosti između komponenti i modela. No, što se tiče dodavanja

jedinstvene funkcionalnosti i prilagodbe u aplikacijama, to je izazov. Kada je riječ o dokumentaciji nerijetko ju je teško pronaći za manje poznate dragulje (engl. *gems*) prema [4].

2.5.3. Dragulji (engl. *gems*)

Gemfile je datoteka koja se stvara i koristi se za opisivanje ovisnosti o draguljima(engl. *gems*) za Ruby programe. Dragulj (engl. *gem*) je zbirka Ruby koda koja se može izdvojiti u "zbirku" koja se kasnije može pozvati. Omogućuje određivanje dragulja (engl. *gems*) koje korisnik želi koristiti uz odabir njihovih verzija. Primjer je vidljiv na slici 2.4

Linija ***Kod***

```
1:        source 'https://rubygems.org'gem 'nokogiri'
2:        gem 'rails', '3.0.0.beta3'
3:        gem 'rack', '>=1.0'
4:        gem 'thin', '~>1.1'
```

Slika 2.4 Primjer instalacije dragulja(engl. *gem*).

Datoteka Gemfile.lock mjesto je na kojem svežanj (engl. *bundler*) bilježi verzije korištenih dragulja (engl. *gems*). Ukoliko korisnik svoj rad odluči prebaciti drugom korisniku ili se iz nekog drugog razloga prebacuje na drugo računalo , korisnik pokreće paket “Bundle install“ koji provjerava zabilježene verzije dragulja (engl. *gems*) i instalira sve ono što nedostaje.

Svežanj (engl. *Bundler*) pruža dosljedno okruženje za Ruby projekte prateći i instalirajući potrebne dragulje i verzije. Svežanj (engl. *Bundler*) oslobađa od ovisnosti i osigurava da su dragulji koji su potrebni prisutni u razvoju, postavljanju i proizvodnji. Početak rada na projektu jednostavan je kao i instalacija paketa.

2.5.4. Devise

Zbog potrebe za autentifikacijom korisnika u samoj aplikaciji uveden je dragulj (engl. *gem*) Devise. Korisnik mora imati mogućnost prijave i registracije na mrežnu (engl. *web*) aplikaciju. Također potreba za definicijom svakog korisnika javlja se i zbog same logike rada aplikacije.

Korisnici se moraju moći međusobno vidjeti i raspoznavati te se njihove informacije moraju negdje zapisat prema [6].

2.5.5. Devise moduli

Autentifikacija baze podataka šifrira i pohranjuje lozinku u bazu podataka radi provjere autentičnosti korisnika kada se prijavljuje. Kako bi se korisnik aplikacije mogao prijavljivati pomoću računa kao što su Facebook, Twitter i Google Devise posjeduje modul Omniauthable. Također dragulj (engl. *gem*) Devise posjeduje mogućnosti oporavka korisnikove lozinke kao i njezino pamćenje pomoću kolačića kako je korisnik ne bi morao unositi svaki put prilikom prijave. Ukoliko korisnik zaboravi lozinku, Devise nudi modul i za to. Preostale mogućnosti samog dragulja (engl. *gem*) navedena su u nastavku prema [6].

- Potvrdivo (engl. *confirmable*) - omogućuje slanje e -pošte s uputama koje će pomoći u verifikaciji računa.
- Mogućnost registracije (engl. *registrable*) - obrađuje prijavu korisnika. Također omogućuje korisnicima uređivanje i brisanje računa.
- Praćenje (engl. *trackable*) - pomaže u praćenju broja prijavljivanja, vremenskih oznaka i IP adrese.
- Raspored sati (engl. *timeoutable*)- odgovoran za istek sesije koja nije bila aktivna neko vrijeme.
- Provjerljivo (engl. *validatable*) - e-pošta i lozinka moraju se provjeriti.
- Zaključavanje (engl. *lockable*)- pruža dodatni sloj sigurnosti kada je aktivirano, račun se može zaključati nakon zadanog broja neuspjelih pokušaja prijave prema [13].

2.6. Jquery odabir datuma (engl. date picker)

Prvobitna funkcionalnost mrežne (engl. *web*) aplikacije je rezervacija glazbenih bendova, stoga se javlja potreba za vizualnim kalendarom koji korisniku otvara mogućnost odabira datuma. U ovom slučaju korisnik može odabrati jedan dan i tako ga rezervirati za sebe. Drugim korisnicima je to vidljivo te oni mogu birati samo preostale slobodne datume.

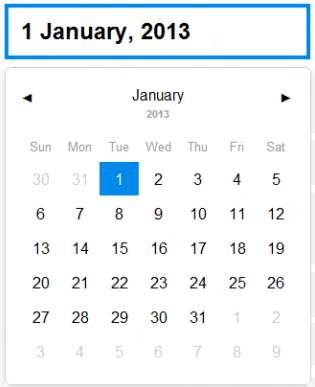
Dodatak za odabir datuma korisničkog sučelja olakšava korisnicima jednostavan i vizualni unos datuma. Korisnik može prilagoditi format i jezik datuma, ograničiti raspone datuma za odabir, lako dodati gumbe i druge navigacijske opcije. Primjer koda nalazi se na slici 2.5.

Linija ***Kod***

```
24:         <script>
25:           $('.js-datepicker').datepicker({
26:             prevText: '&#x3C;',
27:             nextText: '&#x3E;',
28:             dateFormat: 'dd.mm.yy.',
29:             firstDay: 1,
30:             isRTL: false,
31:             showMonthAfterYear: false,
32:             yearSuffix: '',
33:             closeText: 'Close',
34:             currentText: 'Today',
35:             monthNames: ['January', 'February', 'March', 'April', 'May',
            'June', 'July', 'August', 'September', 'October', 'November',
            'December'],
36:             dayNamesMin: ['SUN', 'MON', 'TUE', 'WED', 'THU', 'FRI',
            'SAT'],
37:             minDate: 1,
38:           });
39:         </script>
```

Slika 2.5 Ažuriranje rezervacije.

Metoda "jQuery UI datepicker ()" stvara alat za odabir datuma i mijenja izgled HTML elementa na stranicama dodavanjem novih CSS klasa. Na slici 2.6 prikazan je izbornik za datume sličan onome koji se koristi u samo aplikaciji prema [7].



Slika 2.6 Izbornik datuma.

3. MONOLITNE I JEDNOSTRANE APLIKACIJE

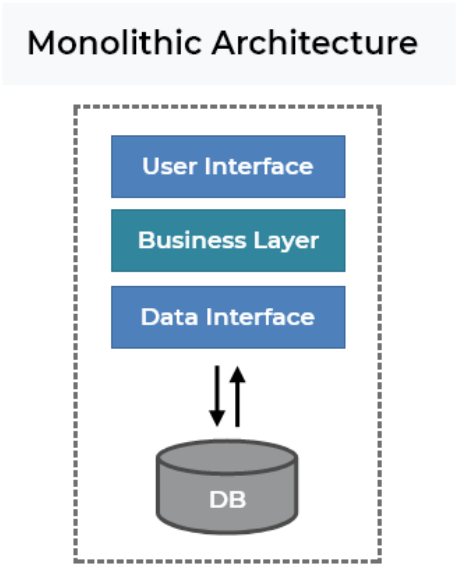
3.1. Monolit aplikacija

U softverskom inženjeringu monolitna aplikacija opisuje jednoslojnu softversku aplikaciju u kojoj se korisničko sučelje i pristupni kod podataka kombiniraju u jedan program s jedne platforme. Monolitna aplikacija je samostalna i neovisna o drugim računalnim aplikacijama. Filozofija dizajna je da je aplikacija odgovorna ne samo za određeni zadatak, već može izvesti svaki korak potreban za dovršavanje određene funkcije. Danas su neke aplikacije za osobne financije monolitne u smislu da pomažu korisniku u izvršavanju cjelovitog zadatka, s kraja na kraj, te su silosi za privatne podatke, a ne dijelovi većeg sustava aplikacija koje rade zajedno. Neki procesori teksta su monolitne aplikacije. Ove su aplikacije ponekad povezane s glavnim računalima prema [10].

U softverskom inženjeringu monolitna aplikacija opisuje softversku aplikaciju koja je dizajnirana bez modularnosti. Modularnost je općenito poželjna jer podržava ponovnu uporabu dijelova logike aplikacije, a također olakšava održavanje dopuštajući popravak ili zamjenu dijelova aplikacije bez potrebe za zamjenom na veliko prema [10].

Modularnost temeljena na kodu omogućuje programerima ponovnu uporabu i popravak dijelova aplikacije, ali za razvoj ovih funkcija održavanja potrebni su razvojni alati (npr. aplikaciju je možda potrebno ponovno sastaviti). Objektna modularnost pruža aplikaciju kao skup zasebnih izvršnih datoteka koje se mogu neovisno održavati i zamijeniti bez ponovnog raspoređivanja cijele aplikacije. Neki objekti pomoću mogućnosti razmjene poruka omogućuju distribuciju objektnih aplikacija na više računala. Arhitektura usmjerena na usluge koristi posebne komunikacijske standarde/protokole za komunikaciju između modula prema [10].

U svojoj izvornoj upotrebi, izraz "monolitni" opisivao je ogromne aplikacije na glavnom računalu bez upotrebljive modularnosti. To, u kombinaciji s naglim povećanjem računalne moći i stoga brzim povećanjem složenosti problema s kojima bi se mogao nositi softver , rezultiralo je neodrživim sustavima i "krizom softvera". Slika 3.1 prikazuje strukturu monolitnu strukturu prema [10].



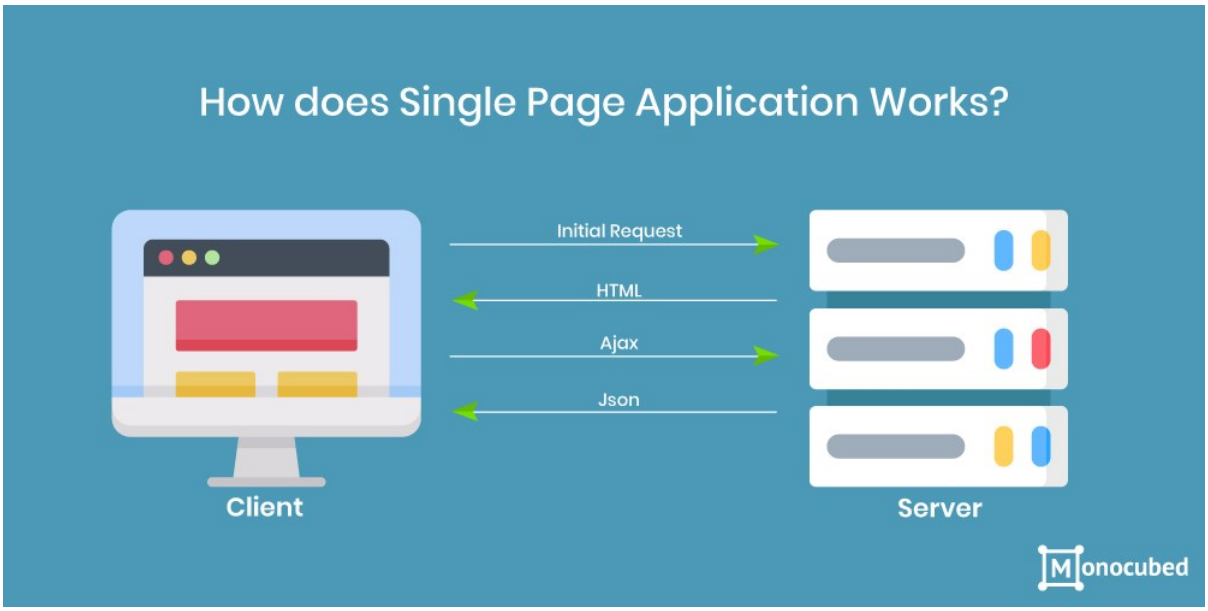
Slika 3.1 Monolitna struktura.

dinamičkim prepisivanjem trenutne internet stranice novim podacima s internet poslužitelja, umjesto zadane metode preglednika koji učitava čitave nove stranice. Cilj su brži prijelazi zbog kojih se Internet stranica više osjeća kao izvorna aplikacija.

U SPA-i nikad se ne događa osvježavanje stranice, umjesto toga, sve potrebne HTML, JavaScript i CSS kodove preglednik dohvaća s učitavanjem jedne stranice , ili se odgovarajući resursi dinamički učitavaju i dodaju na stranicu po potrebi, obično kao odgovor na radnje korisnika. Stranica se ne učitava ni u jednom trenutku procesa, niti prenosi kontrolu na drugu stranicu.

SPA implementacija je mrežna (engl. *web*) aplikacija koja učitava samo jedan Internet dokument, a zatim ažurira sadržaj tijela tog pojedinačnog dokumenta putem JavaScript API-ja, u ovom slučaju pomoću Hotwire-a, kada se želi prikazati drugačiji sadržaj.

To stoga korisnicima omogućuje korištenje Internet stranica bez učitavanja novih stranica s poslužitelja, što može rezultirati povećanjem performansi i dinamičnijim iskustvom, s nekim kompromisnim nedostacima. Slika 3.2 prikazuje način rada SPA prema [11].



Slika 3.2 Način rada SPA.

3.3. Razlika između monolit i SPA aplikacija

3.3.1. Vrijeme reakcije

Na tradicionalnoj internet stranici ili internet aplikaciji svaka će interakcija korisnika uzrokovati potpuno ponovno učitavanje stranice, što znači da korisnik mora čekati da se sve ponovo učita i to je zapravo mana monolitne internet stranice. Na SPA-u većina interakcija će samo učitati potrebne podatke i prikazati rezultat izravno na postojećoj stranici čime se poboljšava vrijeme reakcije i omogućuje implementacija lijepih animacija i efekata. To je osobito važno za mobilne aplikacije u kojima vrijeme prijenosa može biti vrlo veliko, u usporedbi s fiksnim kabelskim vezama.

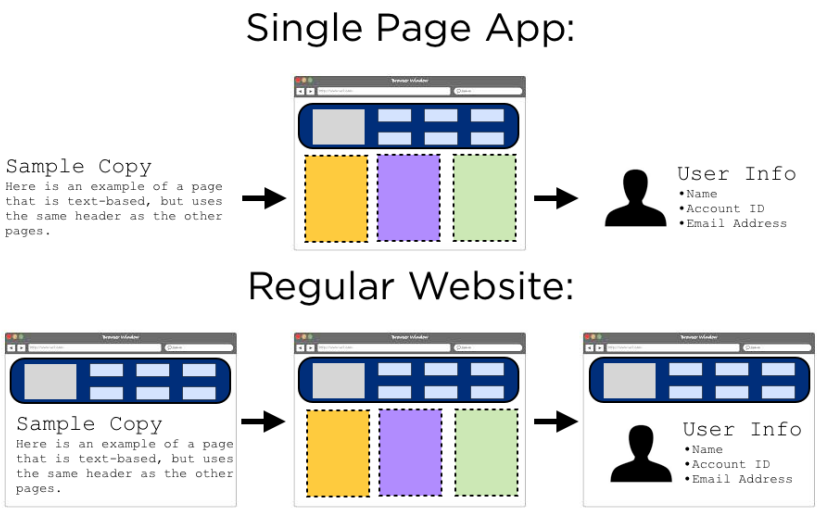
3.3.2. Vrijeme učitavanja

Osim početnog učitavanja aplikacije, SPA omogućuje primanje od poslužitelja samo podatke potrebne za generiranje razlika na sučelju ili, još bolje, samo podatke potrebne za reagiranje na interakciju korisnika. Ovo ne samo da skraćuje vrijeme čekanja za korisnike, već i uvelike smanjuje količinu zahtjeva koje poslužitelji moraju poslužiti kako bi pustili aplikaciju da radi. S druge strane monolit aplikacije ne pružaju takvu potporu i samim time gube prednost prema SPA aplikacijama.

3.3.3. Težina postavljanja

SPA puno je lakše implementirati, barem s prednjeg dijela, u usporedbi s tradicionalnim aplikacijama. Obično se sastoje od jedne .html datoteke (ili vrlo malo njih), CSS paketa i "hrpe" JavaScript datoteka . Takvi statički elementi lako se mogu postaviti na bilo koji mrežni (engl. *web*) poslužitelj bez potrebe za posebnom podrškom ili konfiguracijom prema [12].

Slika 3.3 prikazuje osnovnu razliku između monolitne i SPA aplikacije



Slika 3.3 Razlika između monolitne i SPA aplikacije.

4. HOTWIRE

Hotwire je alternativni pristup izgradnji modernih mrežnih (engl. *web*) aplikacija bez korištenja mnogo JavaScript-a slanjem HTML-a umjesto JSON-a putem žice. To omogućuje brzo učitavanje stranica pri prvom učitavanju, zadržava iscrtavanje predložaka na poslužitelju i omogućuje jednostavnije, produktivnije iskustvo razvoja u bilo kojem programskom jeziku, bez žrtvovanja brzine ili odziva povezanog s tradicionalnom aplikacijom na jednoj stranici.

Srce Hotwire -a je Turbo. Skup komplementarnih tehnika za ubrzavanje promjena stranica i slanja obrazaca, dijeljenje složenih stranica na komponente i strujanje djelomičnih ažuriranja stranica putem WebSocketa.

Dok Turbo obično brine o najmanje 80% interaktivnosti koja bi tradicionalno zahtijevala JavaScript-om , još uvijek postoje slučajevi u kojima je potrebna crtica prilagođenog koda. Stimulus to čini jednostavnim s pristupom stanju i ožičenjima usmjerenim na HTML. Kada je riječ o instalaciji, postavljanjem Hotwire-a istovremeno se postavljaju Turbo i Stimulus. Kako izgleda instalacija vidljivo je na slici 3.1. Prema [8].

Linija	***Kod***
1:	`gem 'hotwire-rails'`
2:	`.bin/bundle install`
3:	`.bin/rails hotwire:install`

Slika 4.1 Instalacija Hotwire-a.

Primjer koda prikazuje zamjenu turbo poveznica (engl. links) sa hotwire turbo poveznicama. Slika 4.2.

Linija	***Kod***
16:	`require("@rails/ujs").start()`
17:	`require("@rails/activestorage").start()`
18:	`require("channels")`
19:	`require("local-time").start()`
20:	`require("@hotwired/turbo-rails")`

Slika 4.2 Zamjena poveznica.

4.1. Turbo i Stimulus

Uz Turbo, poslužitelju je dopušteno izravno isporučivanje HTML-a, što znači svu logiku za provjeravanje dopuštenja i izravne interakcije s modelom domene, a sve ostalo što ide uz programiranje aplikacije može se dogoditi manje više isključivo unutar programskog jezika. Više se ne preslikava logika s obje strane JSON podjele. Sva logika živi na poslužitelju, a preglednik se bavi samo konačnim HTML -om.

Ključna atrakcija tradicionalnih aplikacija na jednoj stranici u usporedbi sa staromodnim pristupom zasebnih stranica je brzina navigacije. SPA-i dobivaju veliku brzinu od neprestanog rušenja procesa prijave samo da bi ga ponovno pokrenuli na sljedećoj stranici.

Turbo pruža istu brzinu korištenjem istog modela trajnog procesa, ali bez potrebe da cijela aplikacije bude izgrađena oko paradigme. Nema usmjerivača na strani klijenta za održavanje, nema stanja kojim se pažljivo upravlja.

To se događa presretanjem svih klikova na vezama do iste domene. Kada korisnik pritisne vezu koja ispunjava uvjete, Turbo sprječava preglednik da ga slijedi, mijenja URL preglednika pomoću API -ja za povijest, traži novu stranicu pomoću dohvaćanja, a zatim generira HTML odgovor.

Isto se odnosi i na obrasce. Njihovi podnesci (engl. *submissions*) pretvoreni su u zahtjeve za dohvaćanje iz kojih će Turbo slijediti preusmjeravanje i prikazati HTML odgovor.

Tijekom iscrtavanja, Turbo izravno zamjenjuje trenutni element i spaja sadržaj elementa prema [9]. Slika 4.3 prikazuje radnje koje aplikacija poduzima nakon obavljenih istih radnji, odnosno definirane su posljedične radnje.

Linija ***Kod***

```
4:          after_create_commit  { User.current_user ?
            broadcast_prepend_to("user_reservations", locals: { user:
            User.current_user, post: self } ) : nil}
5:              after_update_commit  { User.current_user ?
            broadcast_replace_to("user_reservations", locals: { user:
            User.current_user, post: self } ) : nil}
6:              after_destroy_commit  { User.current_user ?
            broadcast_remove_to("user_reservations", locals: { user:
            User.current_user, post: self } ) : nil}
```

Slika 4.3 Posljedične radnje.

4.1. Funkcionalnosti i rad aplikacije

4.1.1. Baza podataka

Prije prelaska na MVC model aplikacije važno je prikazati njenu bazu podataka, atribute korisnika i njihovih rezervacija. Primjer koda na slikama 4.4 i 4.5 prikazuje dokument 'schema.rb' aplikacije korištene za u kojem su jasno vidljivi svi atributi. Važno je napomenuti da su obije skupine (klijenti i bendovi) zapravo jedna grupa odnosno 'korisnici'. Razlikuju se po atributu 'is_band' tipa bool.

Linija ***Kod***

```
18:         create_table "user_reservations", force: :cascade do |t|
19:             t.integer "band_id"
20:             t.date "reserved_date"
21:             t.integer "user_id"
22:             t.datetime "created_at", precision: 6, null: false
23:             t.datetime "updated_at", precision: 6, null: false
24:             t.string "address"
25:             t.string "message"
26:             t.boolean "approved"
27:         end
```

Slika 4.4 Atributi rezervacija.

Linija ***Kod***

```
29:         create_table "users", force: :cascade do |t|
30:             t.string "username"
31:             t.string "email"
32:             t.boolean "is_band"
33:             t.datetime "created_at", precision: 6, null: false
34:             t.datetime "updated_at", precision: 6, null: false
35:             t.string "encrypted_password", default: "", null: false
36:             t.string "reset_password_token"
37:             t.datetime "reset_password_sent_at"
38:             t.datetime "remember_created_at"
39:             t.index ["email"], name: "index_users_on_email", unique: true
40:             t.index ["reset_password_token"], name:
           "index_users_on_reset_password_token", unique: true
41:           end
```

Slika 4.5 Atributi korisnika.

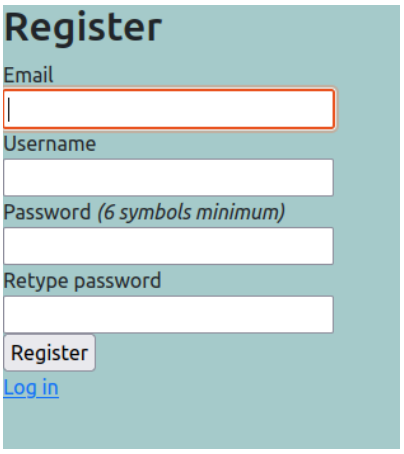
Nakon obrade svih segmenata važno je prikazati koncept i logiku rada same aplikacije. Od samog početka korištenja odnosno korisnikovog prvog pristupa pa sve do njegove prve rezervacije odnosno zahtjeva za rezervaciju.

Ukoliko korisnik prvi put pristupa aplikaciji tj. ne posjeduje račun aplikacija mu nudi opciju registracije koja je podržana od strane dragulja (engl. *gem*) Devise. Primjer koda prikazan je na slici 4.6 dok se na slici 4.7 može vidjeti ono što i sam korisnik vidi.

Linija ***Kod***

```
1:        <h2>Register</h2>
2:
3:        <%= form_for(resource, as: resource_name, url:
          registration_path(resource_name)) do |f| %>
4:          <%= render "users/shared/error_messages", resource: resource %>
5:
6:          <div class="field">
7:            <%= f.label :email %><br />
8:            <%= f.email_field :email, autofocus: true, autocomplete:
          "email" %>
9:          </div>
10:
11:         <div class="field">
12:           <%= f.label :username, "Username" %><br />
13:           <%= f.text_field :username %>
14:         </div>
15:
16:
17:         <div class="field">
18:           <%= f.label :password, "Password" %>
19:           <% if @minimum_password_length %>
20:           <em>(<%= @minimum_password_length %> symbols minimum)</em>
21:           <% end %><br />
22:           <%= f.password_field :password, autocomplete: "new-password"
          %>
23:         </div>
24:
25:         <div class="field">
26:           <%= f.label :password_confirmation, "Retype password" %><br
          />
27:           <%= f.password_field :password_confirmation, autocomplete:
          "new-password" %>
28:         </div>
29:
30:         <div class="actions">
31:           <%= f.submit "Register" %>
32:         </div>
33:       <% end %>
34:
35:       <%= render "users/shared/links" %>
```

Slika 4.6 Registracija.



Slika 4.7 Registracijska forma.

Kada korisnik ispuni formu registracije, ukoliko je to učinio pravilno na izbor mu se stavljaju dvije opcije. Prvo što korisnik vidi nakon registracije je odabir grupe kojoj pripada. Postoje dvije grupe kako je i navedeno u samom uvodu. Korisnik može biti predstavnik glazbenog benda koji sebe stavlja na raspolaganje ostalim korisnicima ili može biti tek običan korisnik koji želi rezervirati bend za svoje potrebe. Kada se odabir izvrši aplikacija postavlja korisniku vrijednost atributa 'is_band' tipa 'bool' u određenu vrijednost kako je i prikazano u kodu na slici 4.8. Slika 4.9 prikazuje ono što korisnik vidi prilikom izbora.

Linija ***Kod***

```
1:          <h1> Hi <%= @user.username %> </h1>
2:          <span>This is your first time at the website, choose your
            side!<span>
3:
4:          <%= link_to "Im a music band - looking for reservation",
            user_set_status_path(is_band: true) %>
5:          <%= link_to "Im a user - looking for bands",
            user_set_status_path(is_band: false) %>
```

Slika 4.8 Izbor skupine.

Slika 4.9 Izbor skupine.

Ukoliko korisnik odabere grupu glazbenih bendova njegov bend se automatski pojavljuje na padajućem izborniku ostalih korisnika koji su u potrazi za bendom. Prva stvar koju korisnik grupe glazbeni bendovi može vidjeti su sve rezervacije koje su imali. Uz to ima mogućnost vidjeti i sve informacije o svakoj rezervaciji kao što su datum, lokacija, napomena i e-pošta klijenta. Ispod sekcije 'Sve rezervacije' korisnik vidi i one nadolazeće. To su zapravo rezervacije koje čekaju potvrdu. Kada korisnik prihvati rezervaciju koju mu je ponudio klijent ona postaje važeća i njen datum postaje zauzet na kalendaru koji se također prikazuje na stranici benda kako bi imali lakši pregled svih rezervacija. Turbo znak okvira (engl. *frame tag*) definira mjesto gdje će podatci strujati (engl. *stream*) , u ovom slučaju su to 'Nadolazeće informacije'. Turbo tok znak (engl. *stream tag*) određuje točan tok s kojega se podatci preuzimaju. Putem modela za rezervacije podatci struje i prikazuju se na stranici.

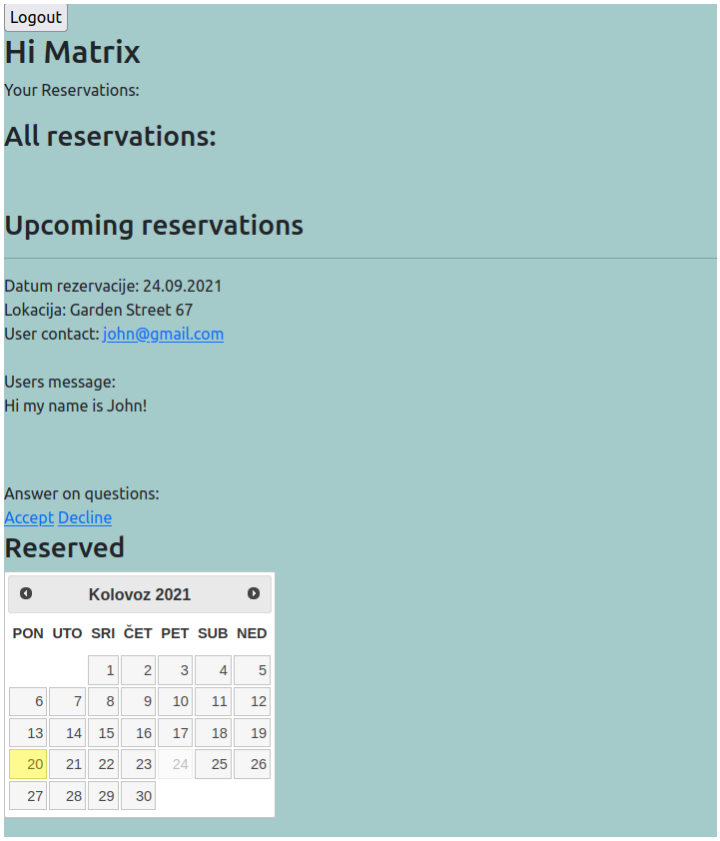
Kalendar koji se prikazuje realiziran je pomoću Jquery odabira datuma (engl. date picker). Važno je napomenuti kako se i sam kalendar mijenja u trenutku nakon prihvaćanja rezervacija. Kada bend prihvati rezervaciju kalendar 'zaključa' datum rezervacije i vidno ga razlikuje od onih slobodnih. Uz sve to drugom bojom označen je trenutni dan radi lakšeg snalaženja. Slika 4.10 prikazuje kod dok slika 4.11 prikazuje ono što korisnik vidi.

Linija ***Kod***

```
1:        <script src="https://code.jquery.com/jquery-1.12.4.js"></script>
2:        <script src="https://code.jquery.com/ui/1.12.1/jquery-
          ui.js"></script>
3:        <link rel="stylesheet"
          href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
4:        <p>Your Reservations: </p>
5:        <%# if @reservations.empty? %>
6:        <%#= "No reservation at the moment" %>
7:        <%# else %>
8:        <h3>All reservations: </h3>
9:        <div class="past-reservations">
10:         <%= render :partial => 'user/reservations/band_reservation',
          :collection => @reservations.where('reserved_date < ?',
          Date.today), as: :reservation %>
11:         <%#= render @reservations.where('reserved_date < ?',
          Date.today), user: current_user %>
12:       </div>
13:       <br>
14:       <br>
15:       <h3>Upcoming reservations</h3>
16:       <%= turbo_stream_from "band_reservations" %>
17:       <%= turbo_frame_tag 'band_reservations' do %>
18:         <div class="futre-reservations">
```

```
19:           <%= render :partial => 'user/reservations/band_reservation',
          :collection => @reservations.where('reserved_date >=?',
          Date.today), as: :reservation %>
20:           <%#= render @reservations.where('reserved_date >= ?',
          Date.today), user: current_user %>
21:         </div>
22:       <% end %>
23:
24:       <h3>Reserved</h3>
25:       <div id="from"></div>
26:       <%# end %>
27:
28:
29:       <style>
30:           .past-reservations {
31:               color: rgba(61, 59, 59, 0.8);
32:           }
33:       </style>
34:       <script>
35:         $('#from').datepicker({
36:           prevText: '&#x3C;',
37:           nextText: '&#x3E;',
38:           dateFormat: 'dd.mm.yy.',
39:           firstDay: 1,
40:           isRTL: false,
41:           showMonthAfterYear: false,
42:           yearSuffix: '',
43:           closeText: 'Close',
44:           currentText: 'Today',
45:           monthNames: ['January', 'February', 'March', 'April', 'May',
          'June', 'July', 'September', 'August', 'Oktober', 'November',
          'December'],
46:           dayNamesMin: ['SUN', 'MON', 'TUE', 'WES', 'THU', 'FRI',
          'SAT'],
47:           beforeShowDay: function (date) {
48:             var exclude =
49:             <%= raw @reserved_dates %>
50:             var day = jQuery.datepicker.formatDate('dd.mm.yy.', date);
51:             return [exclude.indexOf(day) == -1]
52:           }
53:         });
54:       </script>
```

Slika 4.10 Početna stranica benda.



Slika 4.11 Početna stranica benda.

Sadržaj jedne rezervacije prikazan je kodom na slici 4.12 a njen izgled moguće je primijetiti na slici 4.11 odmah iznad izbornika datuma. Uz pomoć turbo okvir oznake (engl. *frame tag*) osigurano je osvježavanje početne stranice benda ukoliko se rezervacija naknadno promijeni ili uništi. Klijent nema tu mogućnost ali ako se taj dio izvrši ručno od strane upravitelja (engl. *admin*) na njegov zahtjev važno je da za promjene sazna i glazbeni bend koji je rezerviran. Nakon navedenih radnji u istom trenutku se osvježava stranica benda i prikazuje im promjene na određenoj rezervaciji bez potrebe za konstantnim ručnim osvježavanjem. Ovim pristupom iznimno se povećava kvaliteta rada aplikacije. Sprječavaju se nesporazumi i protok krivih informacija kako to korisnik i očekuje.

Linija ***Kod***

```
1:          <%= turbo_frame_tag dom_id(reservation) do %>
2:            <% if reservation.reserved_date < Date.today %>
3:              <hr>
4:              <%= "Reservation date:
           #{reservation.reserved_date.strftime('%d.%m.%Y')}" %>
5:              <br>
6:              <%= "Location: #{reservation.address}" %>
7:              <br>
```

```
8:            User contact: <%= mail_to
          User.find(reservation.user_id).email %>
9:            <br>
10:           <br>
11:           <%= "Users message: " %>
12:           <%= raw reservation.message %>
13:         <% else %>
14:           <hr>
15:           <%= "Reservation date:
          #{reservation.reserved_date.strftime('%d.%m.%Y')}" %>
16:           <br>
17:           <%= "Lokacija: #{reservation.address}" %>
18:           <br>
19:           User contact: <%= mail_to
          User.find(reservation.user_id).email %>
20:           <br>
21:           <br>
22:           <%= "Users message: " %>
23:           <%= raw reservation.message %>
24:           <br>
25:           <br>
26:           <%= "Answer on questions: " %>
27:           <% if reservation.approved == nil %>
28:             <div class="reservation-confirmation">
29:               <%= link_to "Accept", user_approve_reservation_path(id:
          reservation.id, state: true) %>
30:               <%= link_to "Decline", user_approve_reservation_path(id:
          reservation.id, state: false) %>
31:             </div>
32:           <% else %>
33:             <b><p>Reservation <%= reservation.approved ? "accepted!" :
          "declined!" %></p></b>
34:           <% end %>
35:         <% end %>
36:       <% end %>
```

Slika 4.12 Forma rezervacije.

S druge strane korisnik koji je u potrazi za glazbenim bendom na svojoj početnoj stranici ima uvid u sve svoje dosadašnje rezervacije od kojih svaka prikazuje sve informacije o sebi. Moguć je uvid u datum, lokaciju, ime benda , kontakt odnosno e-pošta benda i status rezervacije. Status govori o tome da li je rezervacija prihvaćena od strane benda ili ne, po tome korisnik može znati da li je rezervacija važeća i hoće li se njihova suradnja ostvariti ili ne. Uz sve to korisnik na početnoj stranici vidi i poveznicu (engl. *link*) na novu rezervaciju. Kako i u prethodnim primjerima tako je i ovdje korišten turbo koji osigurava osvježavanje u trenutku. Kada bend prihvati ili odbije ponuđenu rezervaciju status iste se odmah mijenja i tako klijent u trenutku ima uvid u stanje svojih rezervacija. Slike 4.13 i 4.14 prikazuju kod dok slika 4.15 ono što korisnik vidi.

Linija ***Kod***

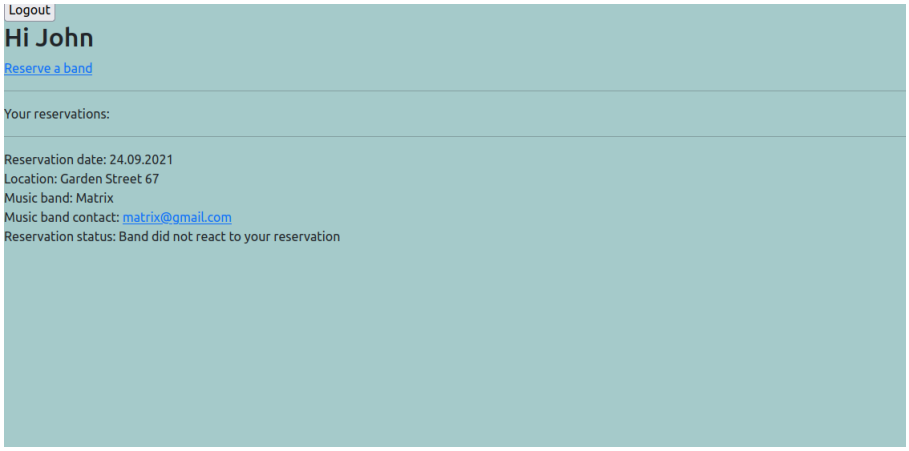
```
1:        <%= link_to "Reserve a band", new_user_reservation_path %>
2:        <hr>
3:        <p>Your reservations: </p>
4:        <%= turbo_stream_from "user_reservations" %>
5:        <%= turbo_frame_tag 'user_reservations' do %>
6:          <%= render @reservations, user: current_user   %>
7:        <% end %>
```

Slika 4.13 Nova rezervacija.

Linija ***Kod***

```
1:        <%= turbo_frame_tag dom_id(reservation) do %>
2:
3:
4:          <div>
5:            <hr>
6:            <%= "Reservation date:
          #{reservation.reserved_date.strftime('%d.%m.%Y')}" %>
7:            <br>
8:            <%= "Location: #{reservation.address}" %>
9:            <br>
10:           <%= "Music band: #{User.find(reservation.band_id).username}"
          %>
11:           <br>
12:           Music band contact: <%= mail_to
          User.find(reservation.band_id).email %>
13:           <br>
14:           Reservation status: <%= if reservation.approved == nil then
15:                                     "Band did not react to your
          reservation"
16:                                   else
17:                                     reservation.approved ? "Accept!" :
          "Declined!"
18:                                   end %>
19:         </div>
20:
21:       <% end %>
```

Slika 4.14 Prikaz prethodno kreiranih rezervacija.



Slika 4.15Prikaz prethodno kreiranih rezervacija.

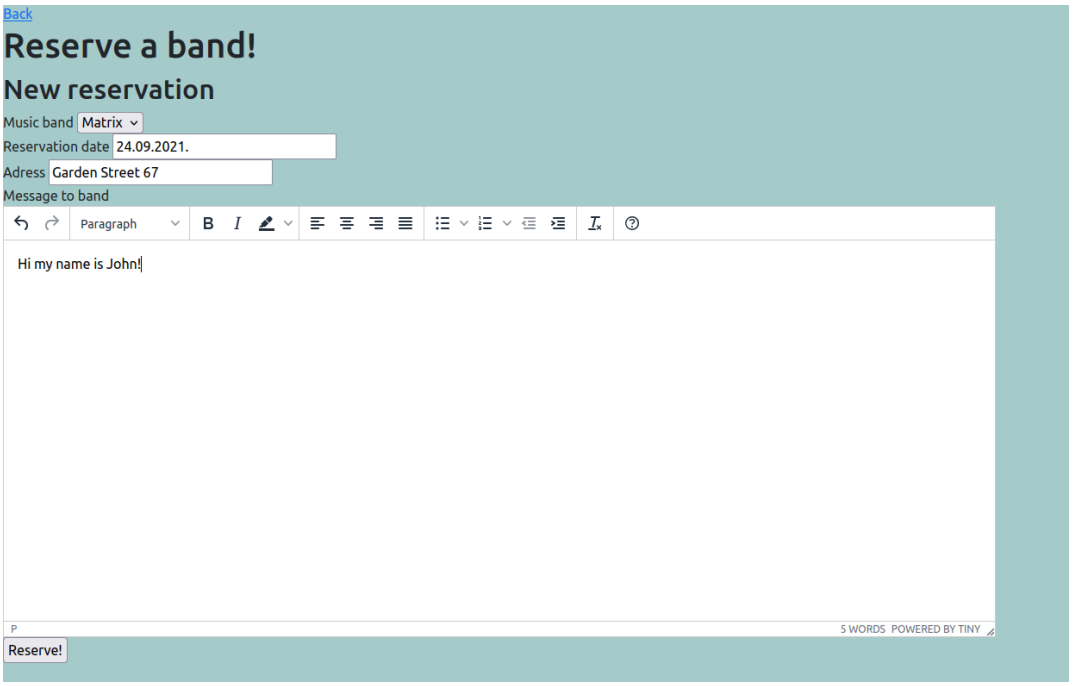
Kada je riječ o novoj rezervaciji korisniku se predstavlja forma koju je potrebno ispuniti. Ona se sastoji od padajućeg izbornika na kojemu su prikazani svi registrirani bendovi, izbornika datuma, adrese te napomene. Izbor datuma stvoren je pomoću Jquery izbornika datuma (engl. *date picker*) na kojem su onemogućeni zauzeti termini tj. dani. Slika 4.16 prikazuje kod a slika 4.17 ono što korisnik vidi.

Linija ***Kod***

```
1:          <%= form_for @reservation do |form| %>
2:            <div>
3:              <%= form.label :band_id, "Music band", class: "label" %>
4:              <%= form.select :band_id, band_select_form, {}, class:
          "input--select" %>
5:            </div>
6:
7:            <div>
8:              <%= form.label :reserved_date, "Reservation date", class:
          "label" %>
9:              <%= form.text_field :reserved_date, class: "js-datepicker",
          autocomplete: "off" %>
10:           </div>
11:           <div>
12:             <%= form.label :address, "Adress" %>
13:             <%= form.text_field :address %>
14:           </div>
15:           <div id="tiny">
16:             <%= form.label :message, "Message to band" %>
17:             <%= form.text_area :message, class: "tinymce" %>
18:           </div>
19:           <div>
20:             <%= form.submit "Reserve!", data: { disable_with: '...' } %>
21:           </div>
22:         <% end %>
```

```
23:
24:         <script>
25:           $('.js-datepicker').datepicker({
26:             prevText: '&#x3C;',
27:             nextText: '&#x3E;',
28:             dateFormat: 'dd.mm.yy.',
29:             firstDay: 1,
30:             isRTL: false,
31:             showMonthAfterYear: false,
32:             yearSuffix: '',
33:             closeText: 'Close',
34:             currentText: 'Today',
35:             monthNames: ['January', 'February', 'March', 'April', 'May',
          'June', 'July', 'August', 'September', 'October', 'November',
          'December'],
36:             dayNamesMin: ['SUN', 'MON', 'TUE', 'WED', 'THU', 'FRI',
          'SAT'],
37:             minDate: 1,
38:           });
39:         </script>
40:         <style>
41:             #tiny {
42:                 margin-right: 40%;
43:             }
44:         </style>
```

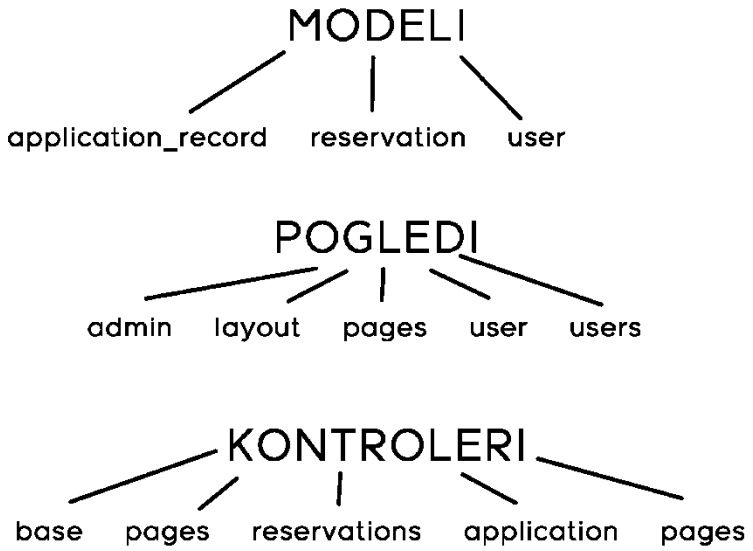
Slika 4.16 Forma nove rezervacije.



Slika 4.17 Kreiranje nove rezervacije.

4.1.2. MVC model aplikacije

Slika 4.18 prikazuje MVC strukturu aplikacije na kojoj se temelji završni rad.



Slika 4.18 MVC struktura aplikacije.

'Application recorder' model stvoren je kao predefinara klasa koju pruža Rails, nju nasljeđuju svi ostali modeli i ona pruža funkcije kao što su pristup bazi, pozivi itd.

Konkretan primjer jednog od kontrolera (eng. *controller*) mrežne (engl. *web*) aplikacije. Nalazi se na lokaciji 'app/controllers/user' .

Na slici 4.19 nalazi se kod provjere pripada li korisnik grupi glazbenih bendova ili grupi korisnika prema tome kreira njegove osobne rezervacije te njihove datume pretvara u oblik koji je prihvatljiv izborniku datuma.

Linija ***Kod***

```
4:          def index
5:                  if current_user.is_band
6:                      @reservations = User::Reservation.where(band_id:
            current_user.id)
```

```
7:                      @reserved_dates =
          @reservations.pluck(:reserved_date).map{ |d|
          d.strftime("%d.%m.%Y.")}
8:                  else
9:                      @reservations = User::Reservation.where(user_id:
          current_user.id)
10:                 end
11:
12:                 @bands = User.where(is_band: true)
13:                 @user = current_user
14:
15:          end
```

Slika 4.19 Provjera korisnika i prilagodba datuma.

Dio koda odgovoran za prihvaćanje odnosno odbijanje rezervacije od strane benda podržan je od strane ovog dijela kontrolera. Slika 4.20.

Linija ***Kod***

```
17:         def approve_reservation
18:                 reservation = User::Reservation.find(params[:id].to_i)
19:                 state =
          ActiveRecord::Type::Boolean.new.deserialize(params[:state])
20:
21:                 if(reservation.approved != nil)
22:                     redirect_to user_root_path, notice: "You have aleredy
          declined the reservation"
23:                     return
24:                 end
25:
26:                 reservation.update(approved: state)
27:                 redirect_to user_root_path, notice: "Successfuly done"
28:                 return
29:          end
```

Slika 4.20 Potvrda rezervacije

Kako korisnik ne bi mogao izbjeći biranje svog statusa odnosno skupine kojoj pripada postavljen je kod koji se nalazi na slici 4.21.

Linija ***Kod***

```
43:         def check_for_first_login
44:                 if current_user.is_band == nil
45:                     redirect_to user_first_login_path
```

```
46:                 end
47:         end
```

Slika 4.21 Prva prijava na stranicu.

Uz prethodno navedeni kontroler javlja se i neizostavan kontroler zadužen za kreiranje rezervacija. Ukoliko postoji mogućnost kreiranja svakako su potrebne i opcije koje nude prepravljanje i brisanje istih. Opcije ažuriraj (engl. *update*) i uništi (engl. *destroy*) dodane su također kako bi se HTML odnosno prikaz mogao osvježiti u trenutku. Kontroler se nalazi na lokaciji 'app/controllers/user' .Navedene tri opcije vidljive su na slikama 4.22 , 4.23 i 4.24 (tim redom).

Linija ***Kod***

```
5:          def create
6:
7:                  @reservation = User::Reservation.new(reservation_params)
8:                  @reservation.user_id = current_user.id
9:                  if @reservation.save
10:                     redirect_to user_root_path, notice: "Reservation done
          successfuly"
11:                 end
12:         end
```

Slika 4.22 Kreacija rezervacije.

Linija ***Kod***

```
29:         def update
30:                 @reservation = User::Reservation.new(params[:id])
31:                 respond_to do |format|
32:                     if @reservation.update(reservation_params)
33:                         format.html { redirect_to @reservation, notice:
          'Reservation was successfully updated.' }
34:                         format.json { render :show, status: :ok,
          location: @reservation }
35:                     else
36:                         format.html { render :edit }
37:                         format.json { render json: @reservation.errors,
          status: :unprocessable_entity }
38:                     end
39:                 end
40:         end
```

Slika 4.23 Ažuriranje rezervacije.

Linija ***Kod***

```
44:         def destroy
45:                 @reservation = User::Reservation.new(params[:id])
46:                 if @reservation.destroy
47:                     respond_to do |format|
48:                         format.html { redirect_to user_root_path, notice:
           'Reservation was successfully destroyed.' }
49:                         format.json { head :no_content }
50:                     end
51:                 end
52:          end
```

Slika 4.24 Uništenje rezervacije.

5. ZAKLJUČAK

Kada je riječ o mrežnim (engl. *web*) aplikacijama i korisnikovom iskustvu istih, vrlo važna karakteristika je brzina njihovog odgovora na korisnikove postupke. Ovaj završni rad tako proučava monolitne i SPA te njihove prednosti i mane kako bi se pronašla najbolja opcija za priloženu aplikaciju. Kako je i navedeno u radu glavni problem monolitnih aplikacija je njihovo 'cjelokupno' osvježavanje . Bez obzira na korisnikov postupak aplikacija će se osvježit u cjelovitosti kako bi mogla prikazati i najmanju promjenu . U ovom konkretnom slučaju to je možda neprimjetno zbog jednostavnosti aplikacije ali kada je riječ o nečemu puno opširnijem i složenijem vrijeme osvježavanja svakako postaje nešto što korisnik lako može primijetiti. SPA izbjegavaju taj problem tako što osvježavaju samo segment koji je korisnik promijenio i sve to čine u jednom okviru (engl. *frame*).

Zaključeno je da ta mogućnost potpuno poboljšava korisnikovo iskustvo korištenja aplikacije. Ne samo da se sve odvija vrlo brzo nego i sam korisnik ne primjećuje velike razlike prilikom korištenja aplikacije. Stoga se ovaj rad temelji na dodavanju Hotwire pristupa ovoj konkretnoj mrežnim (engl. *web*) aplikaciji i tako se stvara njeno osvježavanje u trenutku. Ako se za primjer uzmu dva korisnika gdje jedan čeka potvrdu drugog, ne stvara se potreba za konstantnim osvježavanjem aplikacije od strane korisnika kako bi uvidio da se dogodila promjena te da je njegov zahtjev prihvaćen od strane drugog korisnika. Dodavanjem Hotwire pristupa rješava se taj problem i aplikacija u trenutku mijenja samo status zahtjeva bez ostalog nepotrebnog osvježavanja ostalih segmenata na stranici. Pomoću Hotwire pristupa aplikacija postaje pristupačnija i jednostavnija za korisnike.

LITERATURA:

[1] HTML ,dostupno na https://en.wikipedia.org/wiki/HTML [pristupljeno: 5.9.2021.]

[2] CSS ,dostupno na https://hr.wikipedia.org/wiki/CSS [pristupljeno: 5.9.2021.]

[3] What is CSS ,dostupno na https://hr.wikipedia.org/wiki/CSS [pristupljeno: 6.9.2021.]

[4] D. Pareek ,Ruby on Rails Introduction ,dostupno na https://www.geeksforgeeks.org/ruby-on-rails-introduction/ [pristupljeno: 8.9.2021.]

[5] A. Singh ,Ruby On Rails M-V-C Architecture, dostupno na https://medium.com/stackavenue/m-v-c-architecture-of-ruby-on-rails-9712719a69ed [pristupljeno: 6.9.2021.]

[6] A. Ackerman, Beginner's guide to the Devise gem, dostupno na https://dev.to/ackers93/beginner-s-guide-to-the-devise-gem-35hm[pristupljeno: 6.9.2021.]

[7] Getting Started with jQuery UI ,dostupno na https://learn.jquery.com/jquery-ui/getting-started/ [pristupljeno: 7.9.2021.]

[8] Hotwire for Rails ,dostupno na https://github.com/hotwired/hotwire-rails [pristupljeno: 11.9.2021.]

[9] Hotwire-Turbo-Stimulus ,dostupno na https://turbo.hotwired.dev/handbook/introduction [pristupljeno: 12.9.2021.]

[10] Monolithic application ,dostupno na https://en.wikipedia.org/wiki/Monolithic_application [pristupljeno: 10.9.2021.]

[11] SPA (Single-page application) ,dostupno na https://developer.mozilla.org/en-US/docs/Glossary/SPA [pristupljeno: 12.9.2021.]

[12] A. Grosselle ,Single page application: from monolithic to modular ,dostupno na https://developer.mozilla.org/en-US/docs/Glossary/SPA [pristupljeno: 12.9.2021.]

[13] Learning Devise for Rails ,dostupno na https://subscription.packtpub.com/book/application_development/9781782167044/1/ch01lvl1sec08/devise-modules [pristupljeno: 20.9.2021.]

[14] Vue vs React: Comparison of Best JavaScript Frameworks,dostupno na https://www.monocubed.com/vue-vs-react/ [pristupljeno: 20.9.2021.]

SAŽETAK

Na samom početku završnog rada obrađeni su jezici za izradu Internet stranica kao uvod u cjelokupno područje razvoja Internet stranica kao i aplikacija. Spomenuti su ne programski jezici poput HTML-a i CSS-a te njihove osnove kako bi se jednostavno razumjela njihova sama struktura. Nakon toga obrađen je programski jezik Ruby i njegovo okruženje u kojem je odrađen praktični dio rada Rails kao i njegova glavna obilježja te struktura aplikacije odnosno MVC model. Predstavlja se prikaz konkretnog MVC modela aplikacije te dragulj (engl. gem) koje koristi kao što je Devise. Za lakše razumijevanje i shvaćanje potrebe dragulja (engl. gem) opisana su i njegove mogućnosti. Uz sve navedeno spomenut je i izbornik datuma koji se također nalazi u aplikaciji. Pristup Hotwire te njegovi dijelovi Turbo i Stimulus opisani su prije monolitnih i SPA. Kako i zašto se koriste i zašto stvaraju prednosti u priloženoj aplikaciji. Na samom kraju nalaze se obilježja i razlika između monolitnih i SPA.

Ključne riječi: monolit, jednostrana aplikacija, vrijeme reakcije, dragulji (engl. *gems*)

TITLE

Hotwire approach to developing Ruby on Rails web applications

ABSTRACT

At the very beginning of the final work, the languages for creating web pages were covered as an introduction to the entire field of web site development as well as applications. Non-programming languages are mentioned such as HTML and CSS and their basics in order to easily understand their structure. After that, the programming language Ruby and its environment Rails, in which the practical part of application was made, as well as its main features and the structure of the MVC model. A presentation of a specific MVC model of the application and a gem called Devise. To make it easier to understand and comprehend the need for a gem, its possibilities are also described. In addition to all the above, the date menu is also mentioned, which is also in the application. The Hotwire approach and its Turbo and Stimulus parts have been described before monolithic and single-page applications. How and why they are used and why they create advantages in the attached application. At the very end are the features and differences between monolithic and one-sided applications.

Keywords: monolith, one-sided application, reaction time, gems

ŽIVOTOPIS

Marko Jović rođen u Vinkovcima 11.8.1999. s prebivalištem u Vinkovcima i boravištem u Osijeku. Nakon završene osnovne škole Ivana Gorana Kovačića Vinkovci, upisuje Tehničku školu Ruđera Boškovića u Vinkovcima. Po završetku srednje škole upisuje na Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku na Preddiplomski sveučilišni studij Računarstvo 2018. godine na kojem i trenutno studira završavajući treću godinu preddiplomskog studija.

Marko Jović

_____

PRILOZI

Na CD-u:

1. Završni rad “ Hotwire pristup razvoju Ruby on Rails web aplikacija” u *.docx* formatu
2. Završni rad “ Hotwire pristup razvoju Ruby on Rails web aplikacija” u *.pdf* formatu
3. Izvorni kod eksperimentalnog dijela rada