**FAKULTET ZA INFORMACIONE TEHNOLOGIJE**

**PODGORICA**

**DOKUMENTACIJA PROJEKTA**

**ZA OSNOVNE STUDIJE**

**OPTIMIZACIJA REUSABLE KODA U VUE.JS FRAMEWORKU**

**Mentor:**

**Prof. dr Snežana Šćepanović**

**Student:**

**Borko Smolović 46/17**

**Komentor:**

**Prof. Ivan Knežević**

**Podgorica, JUL 2020**

**Sadržaj**

[1. UVOD 2](#_Toc46365643)

[2. Osnovne odlike Vue JS framework-a 3](#_Toc46365644)

[3. Vuetify 6](#_Toc46365645)

[4. Laravel 8](#_Toc46365646)

[5. Tehnički opis projekta i baze podataka 10](#_Toc46365647)

[5.1 Opis realnog sistema 10](#_Toc46365648)

[5.2 Opis funkcionalnosti sistema 10](#_Toc46365649)

[5.2.1 Identifikovanje aktera i slučajeva korištenja 10](#_Toc46365650)

[5.2.2 Specifikacije slučajeva korišćenja 15](#_Toc46365651)

[5.3 Klasni dijagram baze podataka 22](#_Toc46365652)

[6. Problemi i implementacija optimizacije reusable koda u vue.js frameworku 23](#_Toc46365653)

[6.1 Problemi ponavljanja koda u programiranju 23](#_Toc46365654)

[6.2 Implementacija optimizacije 24](#_Toc46365655)

[6.2.1 Implementacija univerzalnih metoda 24](#_Toc46365656)

[6.2.2 Implementacija univerzalnih komponenti 27](#_Toc46365657)

[6.2.2.1 Komponenta za tabelatni izbor objekata 29](#_Toc46365658)

[7. Zaključak 31](#_Toc46365659)

[8. Literatura 32](#_Toc46365660)

1. UVOD

Postoje brojni direktni i praktični razlozi za implementaciju optimizacije reusable koda u svim vrstama programiranja. Ona nosi brojne benefite kao sto su jednostavnost, preglednost, razumljivost i lakoća budućih izmjena koda.

Optimizacija se direktno odnosi na pravljenje univerzalnih komponenti kao i metoda koje mogu da se iskoriste na više mjesta, cime se olakšava rad sistema i ubrzavaju potencijalne izmjene, jer ukoliko se komponenta mijenja na jednom mjestu izmjene se primijene svudje gdje se komponenta koristi, umjesto da pišemo posebne komponente ili metode za svaki slučaj korištenja čime se dobija višak i ponavljanje koda, kao i povećanje kompleksnosti izmjena jer se izmjene moraju posebno implementirati na zasebnim komponentama.

Glavni benefiti se primjećuju kod velikih projekata koji se mogu sastojati od više stotina pa i hiljada komponenti, medju kojima se dosta stvari ponavlja.

Grupisanjem ovih komponenti u slične i pravljenjem univerzalnih dobija se ogromna ušteda kako na vremenu tako i na ostalim potrebnim resursima za realizaciju projekta, i upravo sa veličinom projekta skaliraju i benefiti korištenja ovog tipa programiranja.

NEKE OD GLAVNIH KORISTI I PREDNOSTI OPTIMIZACIJE KODA:

• Povećanje efektivnosti i efikasnosti web aplikacija

• Razvijanje preglednih, razumljivih i lakih za izmjenu web aplikacija

• Smanjenje viška koda

• Optimizacija resursa potrebnih za izradu web aplikacija

• Značajne uštede na vremenu izrade web aplikacija

• Poboljšanje performansi i održivosti sistema

2. Osnovne odlike Vue JS framework-a

U prethodnih 10 godina web stranice su postale mnogo naprednije i dinamičnije zahvaljujući javascript-u.

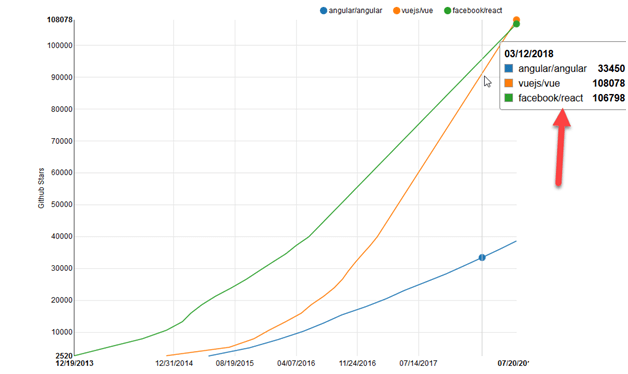
Ključne izmjene predstavljaju pomjeranje ogromne količine koda sa servera u naše pretraživače, ostavljajući nas sa javascript kodom koji je povezan sa mnoštvom html-a i css-a ali sa slabom organizacijom. Ovaj problem je direktno povezan sa nastankom javascript framework-a kao sto su vue i angular, react i slično.



Slika 2.1 React vs Vue vs Angular

Ovaj projekat je rađen u VueJS framework-u. Od gore navedenih framework-a VueJS je najmladji, nastao je 2014 godine, što je veoma mlado u svijetu programskih jezika (za referencu Java je nastala 1995 godine), ali je ujedno i najoptimizovaniji, i ima najbolju podršku od njih, redovne update, bug fix-ove, kao i najaktivniji community gdje svi mogu da učestvuju u razvoju kao i u predlozima za nadogradnje, izmjene i dodatke.

Upravo ovakav pristup i fokus na pristupačnost, raznovrsnost, performanse, jednostavnost i testabilnost su privukli ogroman broj kako senior developera tako i onih kojima je kodiranje samo hobi i postali jedan od najprepoznatljivijih frameworka za kratak period.



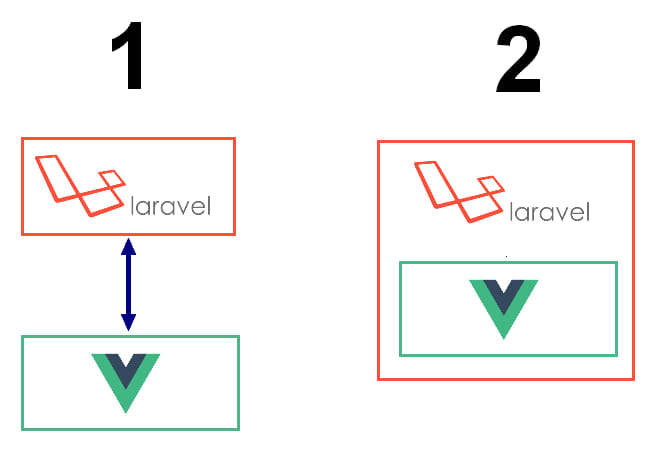
[Slika 2.2 Popularnost VueJS, Angular i React frameworka](https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu)

Vue je progresivan framework što znaci da može da se implementira u samo jedan dio web aplikacije, i da radi zajedno i paralelno sa drugim frameworkom. Česta kombinacija i ujedno jedna od najjačih su VueJS i Laravel.

Izdvajamo dva tipa korištenja VueJS + Laravel kombinacije.

* Prvi tip predstavlja posebne projekte koji su povezani ali mogu da funkcionišu nezavisno jedan od drugog.
* Dok drugi tip predstavlja implementaciju VueJS u Laravel projektu kao dio njega pri čemu vue zavisi od laravela.

Obično se kod većih projekata VueJS i Laravel razdvajaju i komuniciraju pomoću dodatnog Vue-ovog alata zvanog Core, Vuex i Vue-Router.



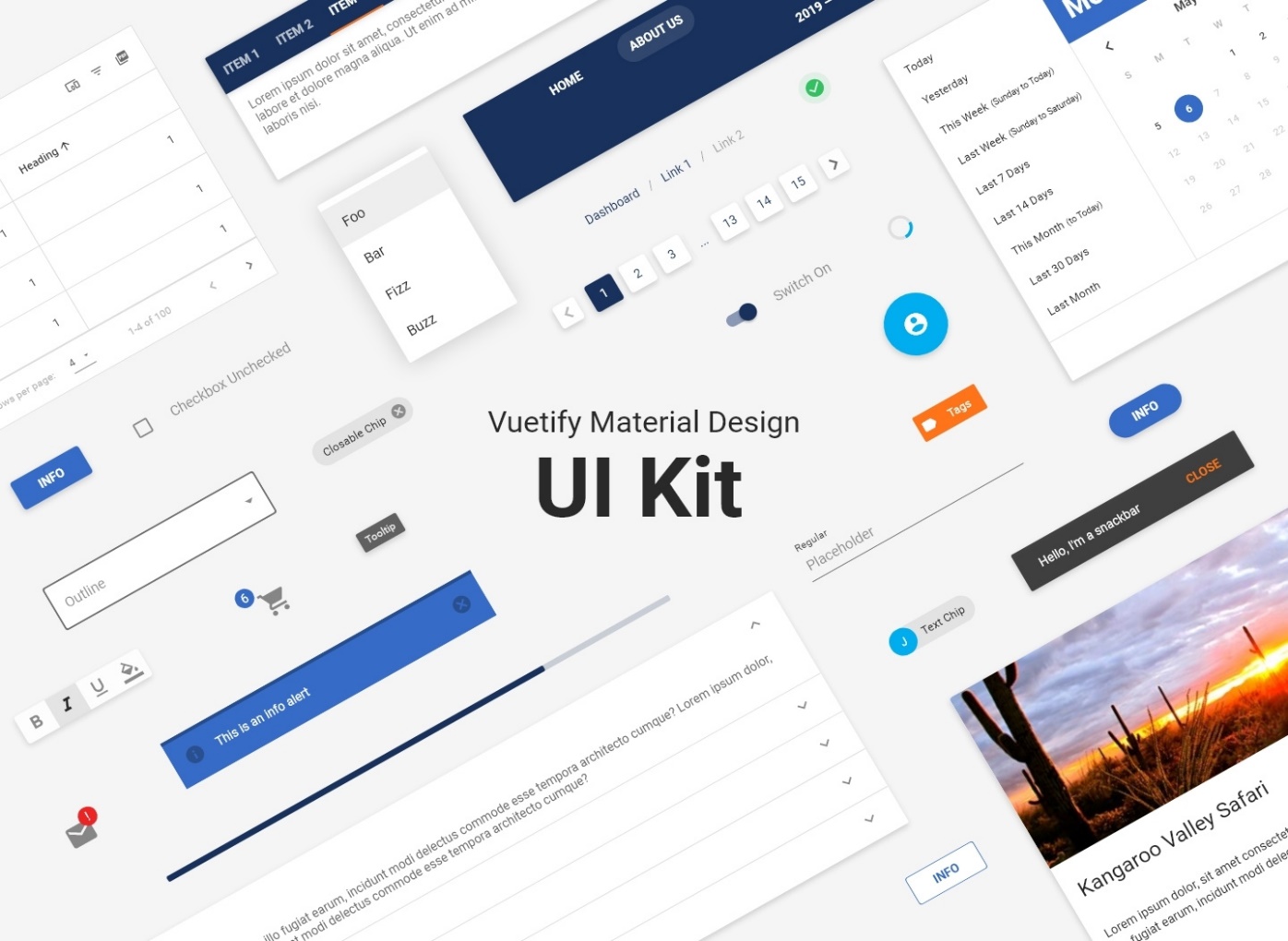
Kao i svaki framework Vue ima svoju predefinisanu biblioteku komponenti, koje se mogu pozivati unutar koda. Svaka komponenta ima svoj javascript, html i css, i moze se koristiti na više mjesta istovremeno.

Više informacija o ovome možete pronaći na VueJS oficijalnom websajtu [vuejs.org](file:///C:\Users\knezi\AppData\Local\Temp\Rar$DIa7120.43763\vuejs.org), gdje je čitav framework detaljno opisan kroz njihovu dokumentaciju.

3. Vuetify

Vuetify predstavlja najbolju biblioteku za VueJS koja se aktivno razvija jos od 2016 godine.

Glavni cilj ovog projekta je da obezbijedi korisnicima sve potrebne komponente, dodatke, alate i plugine za olakšanu izradu modernih, responsivnih i brzih web aplikacija.

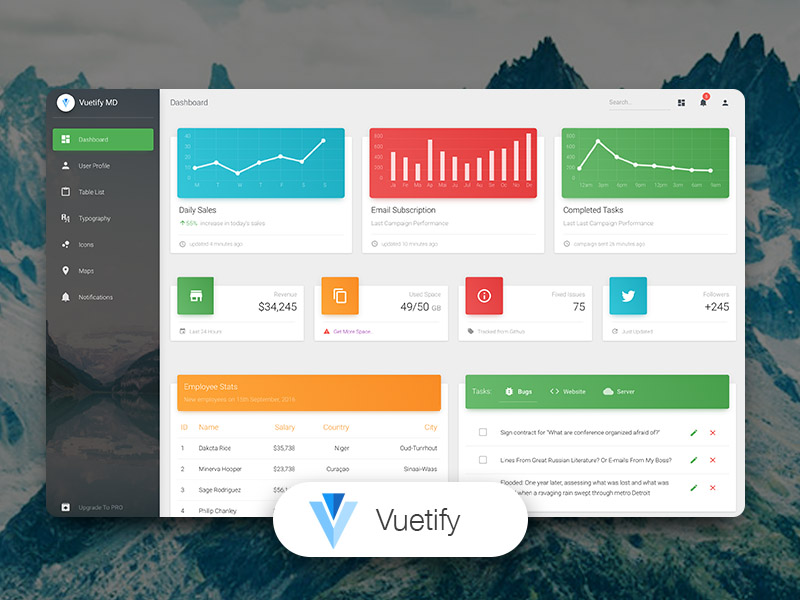


Slika 3.1 Vuetify Material Design UI Kit

U suštini Vuetify predstavlja obogaćenu kolekciju svih alata i komponenti koje koristi VueJS sa dodatnim funkcionalnostima.

Zbog svoje popularnosti Vuetify odlikuje bogat ekosistem alata i dodataka, nedeljna izdanja (update-i), masovnu zajednicu, profesionalne usluge i dugoročnu podršku. Više o Vuetify biblioteci možeTE pronaći na oficijalnom web sajtu <https://vuetifyjs.com/>

Pored pojačanih Vue komponenti, neke od glavnih prednosti Vuetify biblioteke je ogromna zajednica od koje možete besplatno preuzeti mnoštvo profesionalno odrađenih tema za razne vrste web aplikacija.



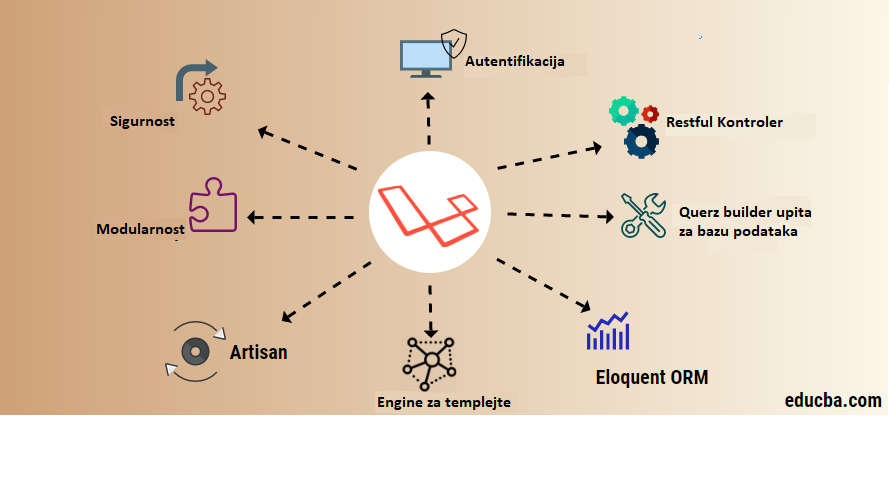
Slika 3.2 Vuetify Material Dashboard template

Tema korištena u ovom projektu je upravo preuzeta sa jednog takvog mjesta. Zove se Vuetify Material Dashboard i besplatna je.

Napravljena je od strane Creative Tim-a i možete je preuzeti na sledećem linku: <https://www.creative-tim.com/product/vuetify-material-dashboard>

4. Laravel

Laravel predstavlja besplatan, open-source web framework baziran na PHP platformi. Obično se koristi za realizaciju backend arhitekture web aplikacija, ali ima sve potrebne elemente i za izradu frontend-a.



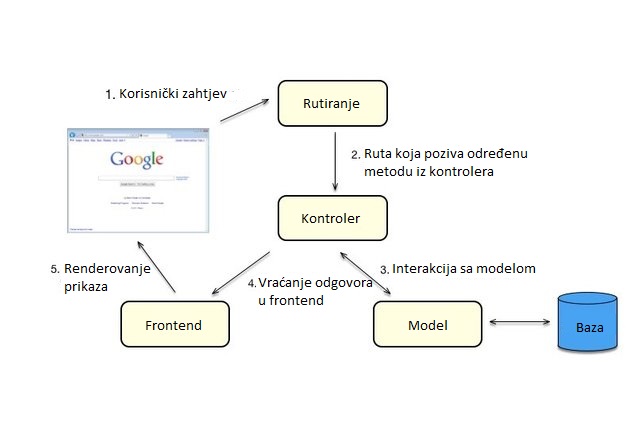
Slika 4.1 Funkcionalnosti Laravela

Ono što dijeli Laravel od konkurencije je to da u sebi ima već implementirane osnovne funkcije kao što su rutiranje, keširanje, autentifikacija i sesije.

On ima i dodatne univerzalne funkcije predviđene za rad sa bazom podataka, upravo zbog ovakvih olakšica, po mišljenju velikog broja developera, Laravel predstavlja najbolji PHP framework i najviše se koristi.

U laravel projektu možemo napraviti model tabele sa kojom planiramo da radimo, i koji će da sadrži sve promjenjive po nazivima polja iz tabele za koju pravimo model.

Dok za svaki model može postojati jedan ili više kontrolera u kojima možemo pisati funkcije. Pored jasno definisanih i intuitivnih laravelovih funkcija koje mogu da pokriju 90% potreba, laravel takođe podržava i raw (sirove) upite kakve bi kucali u normalnom radu sa bazom.



Slika 4.2 Komunikacija elemenata u Laravel-u

Pored modela i kontrolera u laravelu možemo i praviti bazu i tabele korišćenjem migracija.

Sve to objedinjuje i factory seeder, koji mnogo olakšava održavanje podataka u bazi, i svježe postavljanje nove baze, kao i testiranje web aplikacije sa većim količinama podataka.

Još neke značajne osobine Laravel-a:

* Pomaže u kreiranju bržih veb aplikacija.
* Sistem za autentifikaciju server-klijent je brži od ostalih
* Testiranje softvera može biti automatizovano
* Aktivnosti usmeravanja i keširanja su brže
* Pomaže u smanjenju nadredaja servera
* Kritične bezbednosne pretnje i ranjivosti mogu se lako popraviti

Upravo zbog ovoga Laravel predstavlja najrazvijeniji i najkorišteniji PHP framework trenutno. Više informacija možete pronaći na oficijalnom web sajtu <https://laravel.com/>.

5. Tehnički opis projekta i baze podataka

5.1 Opis realnog sistema

Informacioni sistem za INFO BIRO je aplikacija koja omogućava radnicima i korisnicima novinarske agencije da lakše vode evidenciju o svojim ugovorima, zakazanim snimanjima i slicno.

Neregistrovani korisnik moze da zakaze termin u kompaniji i potpise sa njima ugovor na odredjeni broj snimanja, ali nema nista od funkcionalnosti registrovanog korisnika.

Registrovan korisnik ima pregled svih zakazanih termina i ima mogućnost otkazivanja termina, pregleda i izmjene svojih postojecih podataka. Prilikom zakazivanja termina snimanja radnik info biro-a odluzuje da li ce da prihvati ili odbije navedeni zahtjev. Ako radnik odluci da prihvati, od mijenja status evidencije i dodjeljuje joj radnike i resurse.

Admin koristi svoje funkcionalnosti kako bi obnavljao i mijenjao spisak radnika koji koriste system, kao i obicnih korisnika. Aplikacija je jednostavna za upotrebu i predstavlja korak naprijed u svijetu vodjenja evidencija o snimanjima. I radnici i korisnici brzo uočavaju prednosti kada je u pitanju narucivanje evidencijam vodjenje zapisa i obavljanje posla.

Cilj aplikacije je da omogući korisnicima brz pristup i olaksan pristup narucivanju i zakazivanju evidencija za koje su ranije morali da idu u frimu, popunjavaju brdo formulara i pricaju direktno sa radnicima. Ova mogućnost ubrzava i modernizuje proces obavljanja snimanja na terenu. Sastanak sa radnicima je neophodan samo ako kupac prvi put kupuje ugovor.

5.2 Opis funkcionalnosti sistema

### 5.2.1 Identifikovanje aktera i slučajeva korištenja

Učesnici u ovom sistemu su:

• Admin - vlasnik sistema. Funkcije koje izvršava su većinom vezane za radnika i korisnika:

o Login

o Logout

o Pregled radnika i korisnika

o Izmjena radnika i korisnika

o Dodavanje radnika i korisnika

• Radnik - korisnik ovog sistema koji je zadužen za vodjenje evidencija. Ima sledeće funkcije na raspolaganju:

o Login

o Logout

o Pregled evidencija

o Izmjena evidencija

o Dodavanje evidencija

o Brisanje evidencija

o Pregled korisnika

o Pregled zakazanih termina

o Upravljanje partnerima, tipovima radnika, gradovima, drzavama, bankama, troskovima, korisnicima i izvjestajima

• Registrovani korisnik - najbitniji korisnik ovog sistema. Cilj ovog korisnika je da može da pregleda sve svoje evidencije i da zakaže termin novih evidencija. Ima funkcije:

o Login

o Logout

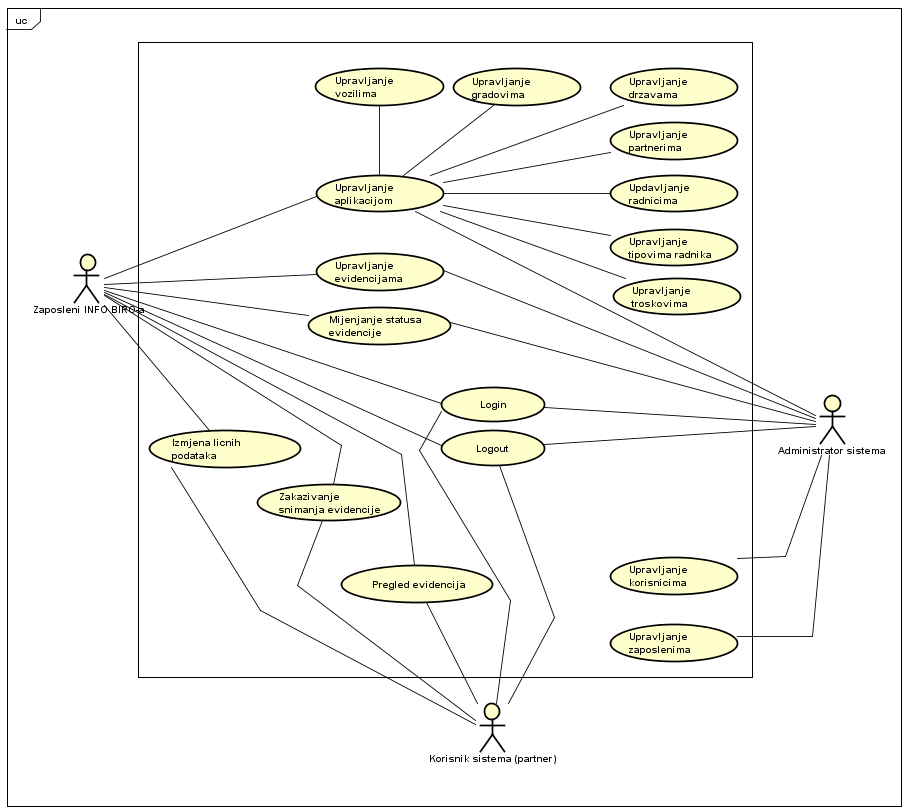
o Pregled evidencija

o Zakazani termini evidencija

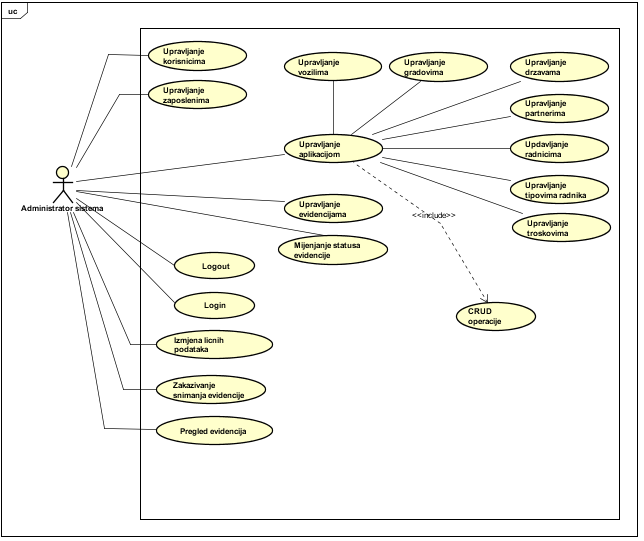
o Otkazivanja termina evidencija

o Pregled korisnikovih podataka

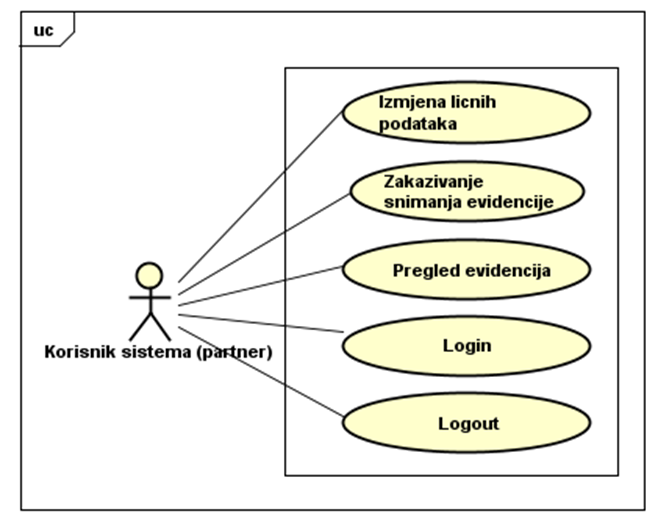
o Izmjena korisnikovih podataka



Slika 5.2.1.1 Dijagram slučajeva korištenja cijelog sistema



Slika 5.2.1.2 Dijagram slučajeva korištenja Admina



Slika 5.2.1.3 Dijagram slučajeva korišćenja partnera

### 5.2.2 Specifikacije slučajeva korišćenja

#### 5.2.2.1 Slučaj korišćenja: Login

Glavni tok:

1. Sistem traži od korisnika da unese svoj username i password.

2. Korisnik unosi svoj username i password.

3. Sistem provjerava da li su unijeti username i password validni i prijavljuje korisnika na sistem.

Alternativni tok:

• Ukoliko su je unesen pogrešan password ili username, sistem prikazuje odgovarajuću poruku. Sistem traži od korisnika da opet unese podatke koji nisu validni.

Preduslovi:

• Sistem je uključen.

• Sistem prikazuje formu za prijavu na sistem.

Postuslovi:

• Korisnik dodat na listu prijavljenih korisnika ili obavještenje zbog čega je nastao prekid izvršenja

#### 5.2.2.2 Slučaj korišćenja: Pregled evidencija

Glavni tok:

1. Korisnik ili agent poziva sistem da mu prikaže sve svoje evidencije.

2. Sistem prikazuje sve evidencije korisniku.

Alternativni tok:

• Ukoliko sistem nema evidencija u bazi, sistem prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen.

Postuslovi:

• Prikazane su evidencije.

#### 5.2.2.3 Slučaj korišćenja: Zakazane evidencije

Glavni tok:

1. Korisnik traži od sistema da mu prikaže sve evidencije koje je zakazao.

2. Sistem prikazuje korisniku sve evidencije koje je zakazao.

Alternativni tok:

• Ukoliko korisnik nema zakazanih evidencija, sistem prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

• Ukoliko dođe do greške, sistem prikazuje odgovarajuću poruku i prekida se scenario.

Preduslovi:

• Sistem je uključen.

Postuslovi:

• Prikazane su evidencije koje je korisnik zakazao.

#### 5.2.2.4 Slučaj korišćenja: Otkazivanje evidencije

Glavni tok:

1. Korisnik traži od sistema da obriše određenja evidencija.

2. Sistem briše traženi termin evidencije.

Alternativni tok:

• Ukoliko nije moguće obrisati termin evidencije, sistem prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

• Ukoliko dođe do greške, sistem prikazuje odgovarajuću poruku i prekida se scenario.

Preduslovi:

• Sistem je uključen. Sistem prikazuje termine evidencija.

Postuslovi:

• Otkazan je termin evidencija ili obavještenje zbog čega je nastao prekid izvršenja

#### 5.2.2.5 Slučaj korišćenja: Pregled korisnikovih podataka

Glavni tok:

1. Registrovani korisnik traži od sistema da mu prikaže podatke korisnika.

2. Sistem prikazuje regitrovanom korisniku podatke.

Alternativni tok:

• Ukoliko dođe do greške, sistem prikazuje odgovarajuću poruku i prekida se scenario.

Preduslovi:

• Sistem je uključen.

• Korisnik je ulogovan

Postuslovi:

• Sistem prikazuje podatke registrovanog korisnika ili obavještenje zbog čega je nastao prekid izvršenja.

#### 5.2.2.6 Slučaj korišćenja: Izmjena korisnikovih podataka

Glavni tok:

1. Registrovani korisnik traži od sistema da mu prikaže podatke korisnika.

2. Registrovani korisnik traži od sistema da mu prikaže formu za izmjenu korisnickih podataka.

3. Sistem mijenja podatke.

4. Sistem prikazuje poruku da je uspješno obavljena izmjena.

Alternativni tok:

• Ukoliko registrovani korisnik unese nevalidne podatke, sistem prikazuje odgovarajuću poruku i ne vrši izmjenu.

• Ukoliko registrovani korisnik unese već postojeći email, sistem prikazuje odgovarajuću poruku i ne vrši izmjenu.

• Ukoliko sistem ne može da izmijeni podatke, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen. Sistem prikazuje korisnikove podatke.

• Korisnik je ulogovan.

Postuslovi:

• Podaci su izmijenjeni.

• Ukoliko izmjena nije uspješno izvršena, prikazuje se odgovarajuća poruka.

#### 5.2.2.7 Slučaj korišćenja: Kontakt preko forme

Glavni tok:

1. Registrovan korisnik traži od sistema da mu prikaže formu za kontakt

2. Registrovan korisnik poziva sistem da pošalje poruku radniku.

3. Sistem šalje poruku radniku.

4. Sistem prikazuje poruku da je korisnikova poruka uspješno poslata.

Alternativni tok:

• Ukoliko registrovan korisnik ne popuni sva obavezna polja, sistem prikazuje odgovarajuću poruku i ne šalje poruku radniku.

• Ukoliko registrovan korisnik unese nepostojeći email, sistem prikazuje odgovarajuću poruku i ne šalje poruku radniku.

• Ukoliko sistem ne može da pošalje poruku agentu, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen.

• Korisnik je ulogovan.

Postuslovi:

• Poruka je poslata preko forme.

• Ukoliko poruka nije uspješno poslata, prikazuje se odgovarajuća poruka.

#### 5.2.2.8 Slučaj korišćenja: Izmjena evidencije

Glavni tok:

1. Radnik poziva sistem da izmijeni nekretnine.

2. Radnik traži od sistema da mu prikaže formu za izmjenu evidencije.

3. Sistem mijenja evidenciju.

4. Sistem prikazuje poruku da je uspješno obavljena izmjena.

Alternativni tok:

• Ukoliko agent ne popuni sva obavezna polja, sistem prikazuje odgovarajuću poruku i ne dodaje evidenciju.

• Ukoliko agent unese već postojeću evidenciju, sistem prikazuje odgovarajucu poruku i ne dodaje evidenciju.

• Ukoliko sistem ne može da izmijeni evidenciju, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen. Sistem prikazuje listu evidencija.

Postuslovi:

• Evidencija je izmijenjena.

• Ukoliko izmjena nije uspješno izvršena, prikazuje se odgovarajuća poruka.

#### 5.2.2.9 Slučaj korišćenja: Dodavanje evidencija

Glavni tok:

1. Radnik poziva sistem da izmijeni evidencije.

2. Radnik traži od sistema da mu prikaže formu za dodavanje evidencije.

3. Radnik poziva sistem da doda evidenciju.

4. Sistem dodaje evidenciju.

5. Sistem prikazuje poruku da je uspješno obavljeno dodavanje.

Alternativni tok:

• Ukoliko radnik ne popuni sva obavezna polja, sistem prikazuje odgovarajuću poruku i ne dodaje evidenciju.

• Ukoliko radnik unese već postojeću evidenciju, sistem prikazuje odgovarajucu poruku i ne dodaje evidenciju.

• Ukoliko sistem ne može da doda evidenciju, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen. Sistem prikazuje listu evidencija.

Postuslovi:

• Dodata je nova evidencija u bazu.

• Ukoliko dodavanje nije uspješno izvršeno, prikazuje se odgovarajuća poruka.

#### 5.2.2.10 Slučaj korišćenja: Brisanje evidencija

Glavni tok:

1. Radnik poziva sistem da izmijeni evidencije.

2. Radnik poziva sistem da obriše evidenciju.

3. Sistem briše aranžman.

4. Sistem prikazuje poruku da je uspješno obavljeno brisanje. Prekida se scenario.

Alternativni tok:

• Ukoliko sistem ne može da obriše evidenciju, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen. Sistem prikazuje listu evidencija.

Postuslovi:

• Evidencija je obrisana.

• Ukoliko brisanje nije uspješno izvršeno, prikazuje se odgovarajuća poruka.

#### 5.2.2.11 Slučaj korišćenja: Pregled zakazanih evidencija

Glavni tok:

1. Radnik poziva sistem da mu prikaže sve zakazane termine.

2. Sistem prikazuje sve zakazane termine.

Alternativni tok:

• Ukoliko sistem nema zakazanih termina, sistem prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen.

Postuslovi:

• Prikazani su zakazani termini korisnika.

#### 5.2.2.12 Slučaj korišćenja: Pregled radnika

Glavni tok:

1. Admin poziva sistem da mu prikaže sve radnike.

2. Sistem prikazuje sve radnike adminu.

Alternativni tok:

• Ukoliko sistem nema radnika u bazi, sistem prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen.

Postuslovi:

• Prikazani su svi radnici.

#### 5.2.2.13 Slučaj korišćenja: Izmjena podataka radnika

Glavni tok:

1. Admin traži od sistema da mu prikaže radnike.

2. Admin traži od sistema da mu prikaže formu za izmjenu radnikovih podataka.

3. Sistem mijenja podatke.

4. Sistem prikazuje poruku da je uspješno obavljena izmjena.

Alternativni tok:

• Ukoliko admin unese nevalidne podatke, sistem prikazuje odgovarajuću poruku i ne vrši izmjenu.

• Ukoliko admin unese već postojeći email, sistem prikazuje odgovarajuću poruku i ne vrši izmjenu.

• Ukoliko sistem ne može da izmijeni podatke, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

• Sistem je uključen. Sistem prikazuje radnike.

• Admin je ulogovan.

Postuslovi:

• Podaci su izmijenjeni.

• Ukoliko izmjena nije uspješno izvršena, prikazuje se odgovarajuća poruka.

#### 5.2.2.14 Slučaj korišćenja: Dodavanje radnika

Glavni tok:

1. Admin traži od sistema da mu prikaže formu za dodavanje radnika.

2. Admin poziva sistem da doda radnika.

3. Sistem dodaje radnika.

4. Sistem prikazuje poruku da je uspješno obavljeno dodavanje.

Alternativni tok:

• Ukoliko admin ne popuni sva obavezna polja, sistem prikazuje odgovarajuću poruku i ne dodaje radnika.

• Ukoliko admin unese već postojećeg radnika, sistem prikazuje odgovarajucu poruku i ne dodaje radnika.

• Ukoliko sistem ne može da doda radnika, prikazuje odgovarajuću poruku. Prekida se izvršenje scenarija.

Preduslovi:

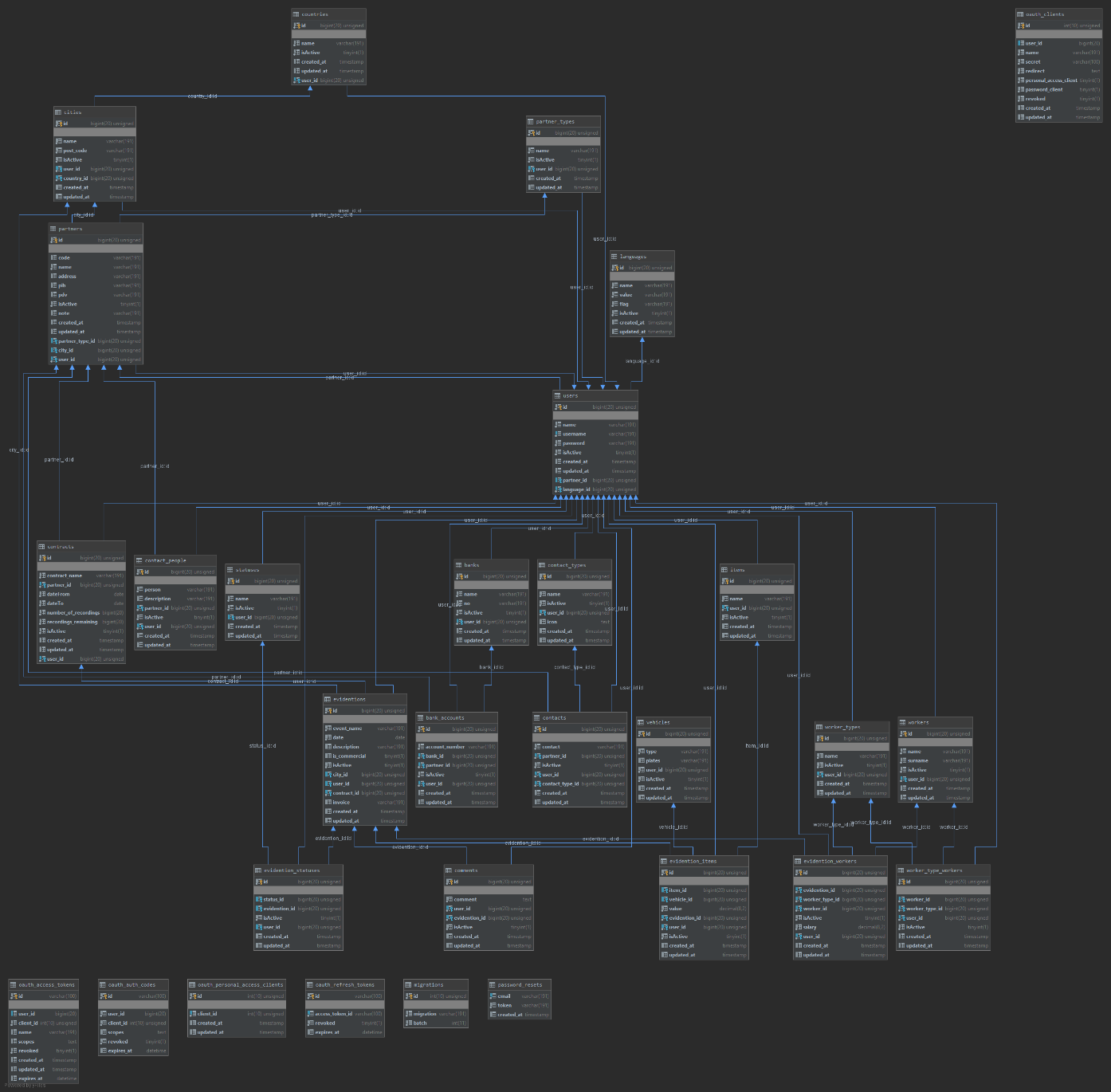
• Sistem je uključen. Sistem prikazuje listu radnika.

Postuslovi:

• Dodat je novi radnik u bazu.

• Ukoliko dodavanje nije uspješno izvršeno, prikazuje se odgovarajuća poruka.

5.3 Klasni dijagram baze podataka



**Slika 5.3.1 Klasni dijagram baze podataka**

S obzirom da se baza sastoji od 30 tabela kao prilog u folderu dokumentacije se nalazi i slika klasnog dijagrama potpune rezolucije kao i uml fajl baze.

6. Problemi i implementacija optimizacije reusable koda u vue.js frameworku

6.1 Problemi ponavljanja koda u programiranju

Problem viška koda se javlja već od nastanka samih programskih jezika i sa sobom nosi mnoge dodatne probleme kao što su funkcionalnost, efikasnost, razumljivost i lakoća izmjene podataka. Ovo se rješava na jednostavan način grupisanjem sličnih stvari u univerzalne komponente i metode.

Pravljenjem univerzalnih stvari eliminišemo potrebu za duplikatima i sebi olakšavamo:

* razumljivost koda, potrebno je dobro iskomentarisati metodu ili komponentu na posebnoj stranici i posle je samo pozivati (koristiti)
* efikasnost prostora, u prosjeku se dobija ušteda na linijama koda po sledećoj formuli – broj linija koda komponente/metode **x** broj ponavljanja/pozivanja iste
* lakoća izmjene, umjesto da mijenjamo komponentu/metodu na više mjesta u slučaju izmjena dovoljno je promijeniti je samo na jednom mjestu odakle se ona poziva

Beneficije se osjete već kod manjih projekata i samo skaliraju kako veličina projekta raste.

VueJS u ovom slučaju maksimalno izlazi u susret navedenim zahtjevima jer je predviđen za ovakvu vrstu optimizacije. Naime on dozvoljava developerima dizajniranje njihovih komponenti koje mogu koristiti unutar projekta. Kao i pravljenje univerzahlnih metoda koje se kasnije mogu samo pozivati na mjestima gdje će se koristiti.

Ovo omogućava veoma jednostavnu optimizaciju reusable koda uz malo planiranja.

6.2 Implementacija optimizacije

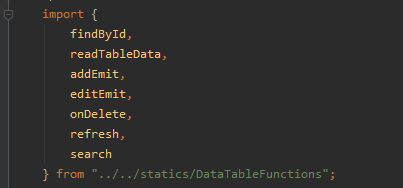
### 6.2.1 Implementacija univerzalnih metoda

Optimizaciju metoda vršimo tako što kreiramo poseban fajl koji će da nam sadrzi sve metode, ili folder sa više fajlova u kojima će metode biti podijeljenje po grupama tipa metode za tabelu, metode za formu itd.

Prilikom izrade ovog projekta sve metode su stavljene u poseban fajl koji se zove DataTableFunctions.js koji se nalazi u /front/src/statics

Prilikom pravljenja univerzalne metode treba sagledati sve slučajeve korištenja i isplanirati šta će metoda imati od ulaznih parametara, i kako od njih dobiti planirani izlaz.

Nakon ove faze planiranja, prilikom pisanja nove metode treba detaljno iskomentarisati svaki njen dio, jer što više slučajeva ta metoda pokriva to je njena kompleksnost veća.

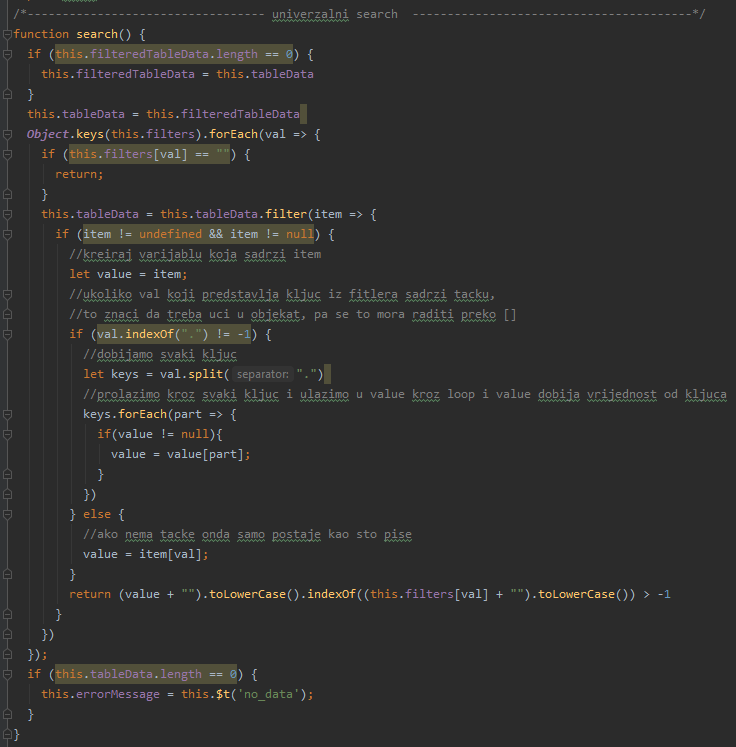


Slika 6.2.1.1 Pozivanje metoda na stranici

Kada napravimo univerzalnu metodu možemo je pozivati na stranicama gdje planiramo da je koristimo i ona će se ponašati isto kao da je na stranici na kojoj je pozvana, moze koristiti sve promjenjive sa stranice čak i ako nisu direktno definisane u metodi.

#### 6.2.1.1 Metoda za pretragu tabele po filterima

Za primjer možemo uzeti metodu pretrage, jer znamo da pretragu nećemo koristiti na samo jednoj strani, već bi svaka stranica sa tabelom trebala da ima svoju pretragu. Čim pretragu želimo da koristimo na više mjesta znači da će doći do ponavljanja koda što zadovoljava naš uslov za pravljenje univerzalne metode.



Slika 6.2.1.1.1 Izgled koda univerzalne metode za pretragu

Da bi metoda prikazana na prethodnoj slici radila moraju se poštovati određena pravila, parametri sa kojima radi moraju biti uniformni na svim stranicama na kojima će se ona koristiti.

U konkretnom slučaju ova metoda radi sa nizovima filters, tableData i filteredTableData koji nisu definisani nigdje u njoj tako da moraju biti definisani na stranici na kojoj se ona poziva pod istim imenima inače metoda neće funkcionisati i izbaciće grešku u konzoli.

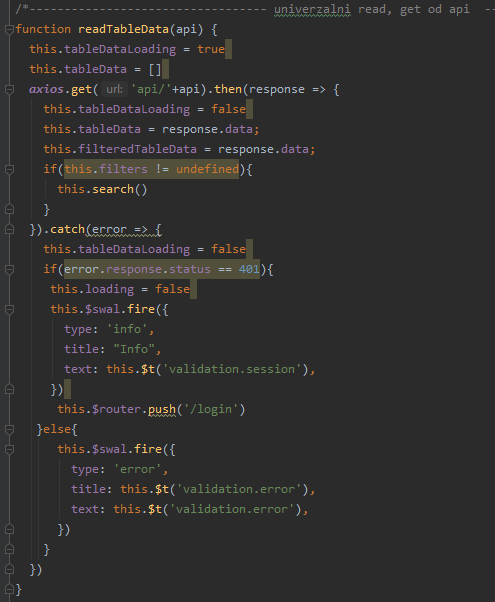
S obzirom da ce filteri na svakoj stranici predstavljati asocijativni niz vrijednosti koje se koriste za pretragu, tableData sve podatke iz tabele i filteredTableData nove filtrirane podatke ova metoda će da funkcioniše na sledeći način.

Ona sa bilo koje stranice na kojoj se pozove uzima gore navedena tri parametra, prolazi kroz glavni niz (niz tableData) u kojem se nalaze svi podaci i filtrira ih po proslijeđenim vrijednostima (niz filters) na kraju sve vrijednosti smješta u konačan niz (niz filteredTableData) i vraća odgovor.

#### 6.2.1.2 Metoda za učitavanje podataka u tabelu

Pored metoda za vezanih isključivo za frontend i javascript mogu se uniformisati i one koje komuniciraju sa backendom.

Za primjer ćemo uzeti univerzalnu metodu za učitavanje podataka u tabelu sa određene rute.



Slika 6.2.1.2.1 Izgled koda univerzalne metode za učitavanje podataka u tabelu

Uzimamo metodu za učitavanje podataka, jer znamo da tabelu nećemo popunjavati na samo jednoj strani, već bi svaka stranica sa tabelom trebala da učitava svoje podatke. Čim učitavanje želimo da koristimo na više mjesta znači da će doći do ponavljanja koda što zadovoljava naš uslov za pravljenje univerzalne metode.

Da bi metoda prikazana na prethodnoj slici radila moraju se poštovati određena pravila, parametri sa kojima radi moraju biti uniformni na svim stranicama na kojima će se ona koristiti.

U konkretnom slučaju ova metoda radi sa nizovima tableData i filteredTableData koji nisu definisani nigdje u njoj tako da moraju biti definisani na stranici na kojoj se ona poziva pod istim imenima inače metoda neće funkcionisati i izbaciće grešku u konzoli.

S obzirom da je backend predviđen da radi tako što GET metode uvijek vraćaju set podataka iz baze jedina promjenjiva koju naša metoda prima je string rute koju treba da aktivira.

Pozivanjem GET zahtjeva na zadatu rutu aktiviramo određenu metodu iz njenog kontrolera koji vrati potrebne podatke za tabelu koju popunjavamo.

Prilikom dobijanja odgovora iz kontrolera set podataka smjestamo u odgovarajuće nizove tableData i filteredTableData preko kojih se oni prikazuju u tabeli.

Pozivanjem ove rute na bilo kojoj strani i mijenjanjem rute koja joj se prosleđuje metoda će vratiti set podataka pogodan za odgovarajuću tabelu.

### 6.2.2 Implementacija univerzalnih komponenti

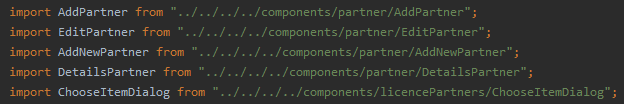
Slično kao i sa metodama možemo praviti univerzalne komponente za našu web aplikaciju. VueJS je i predviđen da radi na ovaj način i veoma je dobro osmišljen sistem koji implementira ove funkcionalnosti.

Optimizaciju univerzalnih komponenti vršimo tako što kreiramo poseban vue fajl koji će da nam sadrzi HTML, CSS i JavaScript potreban za funkcionisanje stranice, i u njoj takodje možemo pozivati i univerzalne metode.

Prilikom pravljenja univerzalne komponente treba sagledati sve slučajeve korištenja i isplanirati šta će komponenta imati od ulaznih parametara koji se nazivaju prop-ovi.

U VueJS frameworku prop-ovi služe za komunikaciju parent komponente sa child komponentom, i obično u slučajevima korištenja univerzalnih komponenti iste pozivamo u nekoj parent komponenti i prosleđujemo potrebne parametre za funkcionisanje stranice.

Nakon ove faze planiranja, prilikom pisanja nove komponente treba detaljno iskomentarisati svaki njen dio, jer što više slučajeva ta komponenta obuhvata to je njena kompleksnost veća.

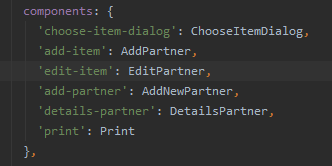


Slika 6.2.2.1 Import komponenti na stranici na kojoj se poziva

Kada napravimo univerzalnu komponentu možemo je pozivati na stranicama gdje planiramo da je koristimo putem import-a

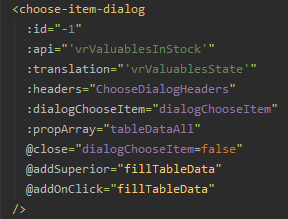
i ona će se ponašati isto kao da je na stranici na kojoj je pozvana, ali ne može koristiti promjenjive od parent komponente već samo one koje joj parent komponenta proslijedi pomoću prop-ova.

Nakon importa željene komponente istu moramo definisati na stranici tako sto je ubacujemo u posebnu Vue-ovu rezervisanu promjenjivu components



Slika 6.2.2.2 Definisanje pozvane komponente u parent komponenti

Nakon definisanja komponente možemo je pozivati u kodu i proslijediti joj potrebne podatke kroz prop-ove.



Slika 6.2.2.3 Pozivanje komponente u kodu

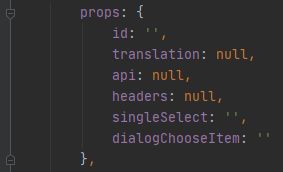
Kao što parent može da komunicira sa child komponentom pomoću prop-ova tako i child može da komunicira sa parentom pomocu emit-a, i upravo na slici 6.2.2.3 pored propova koji se prosleđuju imamo i emitove @close, @addSuperior i @addOnClick koji u slučaju da se dese pozivaju određene funkcije, a preko njih se mogu i prosljeđivati podaci. Emit uvijek možemo prepoznati po tome što sadrži @ ispred imena emit-a.

Za primjer ćemo uzeti komponentu za tabelarni izbor objekata

### 6.2.2.1 Komponenta za tabelatni izbor objekata

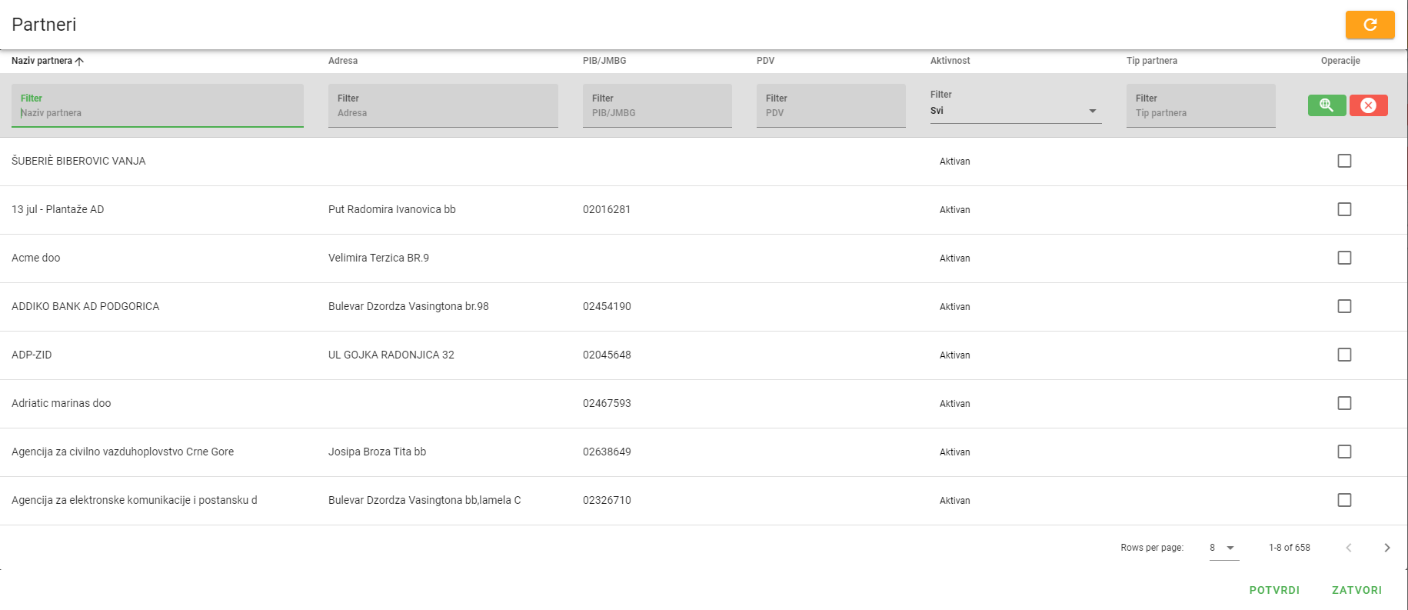
Komponenta za tabelarni izbor podataka prikazuje određene podatke sa neke rute tako da možemo selektovati jedan ili više redova u cilju korištenja istih na parent komponenti.

Od prop-ova koji joj se proslijede zavisi ponašanje komponente.



Slika 6.2.2.1.1 Prop-ovi komponente za tabelarni izbor objekata

* Prop api predstavlja rutu sa koje će komponenta da uzme podatke kojima popunjava tabelu
* Prop translation predstavlja prevode koje će komponenta da gleda pri prikazu istih
* Prop headers predstavlja zaglavlja tabele koja se prikazuje u komponenti
* Prop singleSelect predstavlja boolean vrijednost koje mijenja ponašanje komponente, u zavisnosti od njega se može selektovati jedan ili više redova u tabeli
* Prop id predstavlja dodatan parametar pri učitavanju podataka, ako je id neki broj veci od 0, komponenta će da ga koristi pri učitavanju podataka tako sto kombinuje api/id
* Prop dialogChooseItem predstavlja trenutnu vidljivost komponente



Slika 6.2.2.1.2 Izgled komponente za tabelarni izbor objekata

Komponenta nakon odrađene selekcije vrati u parent komponentu niz objekata ili jedan objekat u slučaju da je singleSelect true.

Ovo omogućava pozivanje i korištenje iste komponente na više mjesta i podaci kojima se popunjava komponenta kao i ponašanje komponente su određeni njenim propovima koje joj parent komponenta zadaje.

Ovim objezbjeđujemo značajne uštede na prostoru i vremenu, jer za svako pozivanje ove komponente uštedimo oko 240 linija koda.

7. Zaključak

Optimizacija reusable koda je od ogromnog značaja ne samo u VueJS-u i JavaScriptu već i u svim ostalim programskim jezicima. Razlike u efikasnosti koda rastu sa veličinom projekta.

Najveće benefite ostvaruje kod izmjena, nadogradnji i testiranja koda, jer pozivajući jednu komponentu na više mjesta navedene radnje vršimo samo jednom umjesto onoliko puta koliko se one pozivaju u web aplikaciji, čime se najviše štedi na vremenu ali i na ostalim resursima.

Implementacija zahtijeva temeljno poznavanje projekta, njegovog principa rada, komponenti i metoda koje koristi da bi se odradilo kvalitetno grupisanje po sličnošću i napravile univerzalne komponente ili metode. Ona zahtijeva više vremena u startu ali se višestruko isplati u kasnijim fazama projekta.

Razlike se još mogu primijetiti i u preglednosti i razumljivosti samog koda, gdje je znatno olakšano održavanje programerima zaduženim za to. Sami kod je dosta jasniji, dok logika funkcionisanja kako komponenti i metoda tako i same web aplikacije postaje mnogo očiglednija i lakša za snalaženje.

U principu bi svaki projekat trebao da ima ovakvu vrstu optimizacije jer se pored samog projekta uniformne komponente i metode mogu koristiti i za druge projekte.

Ideja koja je došla kao rezultat ovog projekta je kreiranje sopstvenog mini framework-a na osnovu univerzalnih komponenti koje često koriste kao na primjer tabelarni prikaz podataka, komponente dodavanja, izmjene brisanja i pretrage.

Ovo bi izradu svakog sledećeg projekta znatno olakšalo i ubrzalo, posebno ako se projekti rade za slične platforme tipa admin panel, upravljanje, prikaz i pregled podataka.

8. Literatura

1. Grafikon sa slike 2.2, “React vs. Angular vs. Vue.js: A Complete Comparison Guide” **-** <https://dzone.com/articles/react-vs-angular-vs-vuejs-a-complete-comparison-gu>
2. Dokumentacija VueJS **-** <https://vuejs.org/>
3. Dokumentacija Vuetify - <https://vuetifyjs.com/>
4. Dokumentacija Laravel - <https://laravel.com/>
5. Slika 3.2 Vuetify Material Dashboard Template - <https://www.creative-tim.com/product/vuetify-material-dashboard>