Univerzitet u Beogradu

Fakultet organizacionih nauka

Katedra za elektonsko poslovanje

BACKBONE.JS

Seminarski rad iz Elektronskog poslovanja

Mentor: Studenti:

Ljubica Ljubisavljević 3503/2020 Marko Bojanić 9/18

Jovana Burdžić 91/18

Danilo Guteša 1002/2020

Beograd, 2021.

**SADRŽAJ**

[1. UVOD 2](#_Toc11518)

[2. PREGLED LITERATURE 3](#_Toc16905)

[3. PREGLED POSTOJEĆIH REŠENJA 7](#_Toc2073)

[4. RAZVOJ APLIKACIJE 8](#_Toc10179)

[5. ZAKLJUČAK 12](#_Toc12004)

[6. LITERATURA 13](#_Toc29716)

# UVOD

Tema ovog seminarskog rada jeste Backbone.js, JavaScript biblioteka za efikasniju organizaciju koda. Razlog za izradu aplikacije pomoću Backbone.js jeste zanimljvi koncept na kom je baziran, pa je pogodan i ze pravljenje aplikacije Javni imenik koja će biti detaljnije predstavljena u jednom od poglavlja.

Na samom početku nalaze se informacije i teorijska znanja koja su potrebna da bi se razumeo koncept i funkcionisanje Backbone.js, detaljno su opisane karakteristike i način na koji se koristi ova biblioteka. Predstavljena su već realizovana rešenja, odnosno aplikacije implementirane pomoću ove biblioteke. U poglavlju Razvoj aplikacije opisana je aplikacija Javni imenik koja koristi funkcionalnosti Backbone.js kao i objašnjenje kodova.

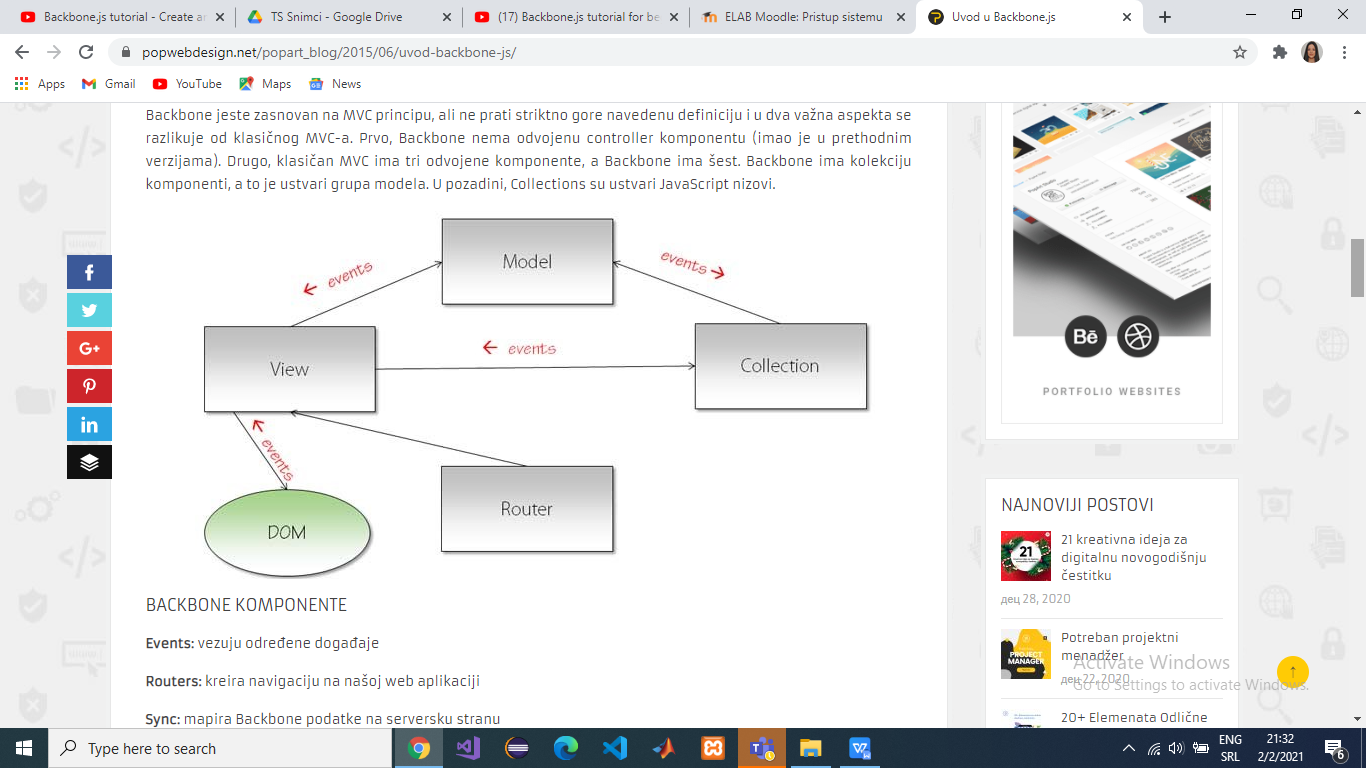
# PREGLED LITERATURE

Backbone.js je JavaScript biblioteka koja omogućava da se kod organizuje u urednom, čistom i efikasnom stilu. Backbone poseduje odličnu dokumentaciju, kod je jednostavan, postoji čak i [verzija sa beleškama](http://backbonejs.org/docs/backbone.html), koja detaljno objašnjava šta i kako radi. Ovaj framework je kreirao Jeremy Ashkenas koji je takođe poznat kao kreator CoffeeSript-a.

Backbone je više biblioteka nego framework. Framework-ci uglavnom zahtevaju da se stvari rade na njihov način, ako se želi njihova upotreba u web aplikacijama. To podrazumeva konfiguraciju koda na način na koji su kreatori framework-a zamislili, što ostavlja malo mesta za fleksibilnost. Kod Backbone-a je samo potrebno da se iskoriste komponente i njegova predefinisana svojstva i metode. Backbone je biblioteka alata koja pomaže da se prave bogatije web aplikacije.

Organizovan je u Model View Controller design pattern-u, poznatijem kao MVC. Sa MVC-em kod je podeljen na tri separatne komponente. M znači model koji je deo koda koji sadrži podatke aplikacije. V je view, zadužen je za prikazivanje podataka na korisnikovom ekranu. C je controller koji služi da transferuje podatke iz modela u view.

Na slici 1 prikazan je MVC.



***Slika 1 - MVC***

Kao što se vidi na slici MVC razbija kod u tri zasebna dela, od kojih svaki ima drugačiju odgovornost. Backbone jeste zasnovan na MVC principu, ali ne prati striktno gore navedenu definiciju i u dva važna aspekta se razlikuje od klasičnog MVC-a. Prvo, Backbone nema odvojenu controller komponentu (imao je u prethodnim verzijama). Drugo, klasičan MVC ima tri odvojene komponente, a Backbone ima šest. Backbone ima kolekciju komponenti, a to je ustvari grupa modela. U pozadini, Collections su ustvari JavaScript nizovi.

Backbone.Model opisuje formu u kojoj se nalazi neki podatak unutar aplikacije, ili model iz realnog života. Npr. model automobila bi sadržao glavne podatke o automobilu: model, godište, potrošnja...

Običan Javascript objekat je takođe jedan model, a Backbone.Model uz sve to nudi funkcionalnosti koje se skoro uvek koriste. Npr. često se čitaju vrednosti modela, ili menjaju neki atribute (za to su definisane funkcije get() i set() ), proveravaju validnosti vrednosti atributa modela ( validate() ), dalje, ukoliko je potrebno dovući podatke sa mreže, metode za učitavanje podataka u model ( fetch() ).

Moguće je i postaviti podrazumevane vrednosti za atribute modela, što se definiše u defaults objektu.

Backbone.Collection najčešće koristi kolekcije čiji su svi modeli istog tipa, ali takvo ograničenje ne postoji. Ukoliko se apstraktnije posmatra Backbone kolekcija, vidi se da je to niz objekata sa pomoćnim metodama koje omogućavaju manipulaciju, npr. sortiranje ( sort() ), filtriranje ( filter() ), izmene pojedinih objekata, dodavanje objekata u kolekciju ( add() ), brisanje ( remove() ), kao i operacije za popunjavanje kolekcije podacima sa mreže ( fetch() ), itd. Unutar Backbone.Collection objekta raspolaže se sa metodama: initialize() [prazan], toJSON(options) [mapira toJSON(options) iz Backbone.Model], sync() [Backbone.sync], add(models, options), remove(models, options), set(model, options), reset(models, options), push(models, options), pop(options), unshift(model, option) [dodaje model na početak], shift(options) [skida model sa početka kolekcije], slice(), get(obj), at(index), where(attrs, first), findWhere(attrs), sort(options), pluck(attrs), fetch(options), create(models, options), parse(resp, options) [return resp], clone(), modelId(attrs). itd.

Backbone.Model, Backbone.Collection i Backbone.Router su unutar Backbone biblioteke fizički odvojeni, ali čine jednu celinu - deo aplikacije koji se bavi podacima.

Backbone.View i Backbone.Event.

Osnovna svrha instanci Backbone.View objekta je da prikažu podatke sačuvane u modelima i kolekcijama, ili podatke nastale tokom izvršavanja aplikacije. Pored toga, Backbone.View ima zadatak da osluškuje događaje, i na njih adekvatno reaguje.Kao i drugi Backbone objekti, i Backbone.View ima više atributa koji se mogu predefinisati. Prvi od njih je el atribut, čija je vrednost DOM reprezentacija objekta kojim upravlja taj View (pored el često pristupamo $el atributu koji se interno pravi i to jQuery objekat napravljen od el - $(el)). Kada se kaže da View kontroliše (upravlja) neki DOM element, misli se da View ima mehanizme da menja taj element, da osluškuje njegove događaje, da ga sakrije, prikaže, obriše, itd.

Treba primetiti da se unutar nekog View objekta vide svi DOM elementi na stranici, ne samo onaj kojim upravlja taj View. To ne iznenađuje jer ne postoji poseban mehanizam enkapsulacije, a osim toga Backbone ima pristup jQuery objektu, jQuery celom DOM drvetu, tako da je jasno zašto je tako. Zbog toga je lako upasti u zamke prilikom pisanja aplikacije, koje će dovesti do upotrebe anti-paterna. Npr. situacija kada View pristupa DOM elementima na stranici za koje nije odgovoran je jedan anti-patern, jer suština postojanja View objekta je da se različitim delovima aplikacije dodele različiti objekti koji će da kotrolišu odgovarajuće delove.

Unutar Backbone.View objekta raspolaže se sa metodama: $(selector) [to je zapravo $.find() unutar instance View], initialize() [prazan], render() [samo return this;], remove() [$.remove i stopListening], setElement(element), delegateEvents(events), delegate(eventName, selector, listener), undelegateEvents(), undelegate(eventName, selector, listener).

Vidi se da unutar Veiw objekta ima dosta metoda koje osim što se bave DOM strukturom stranice, upravljaju i događajima. Događaji (event, eventListener) teško da se mogu logički razdvojiti od izgleda (prikaza i UX) aplikacije. Većina elemenata na stranici treba da ponudi interaktivnost sa korisnikom.

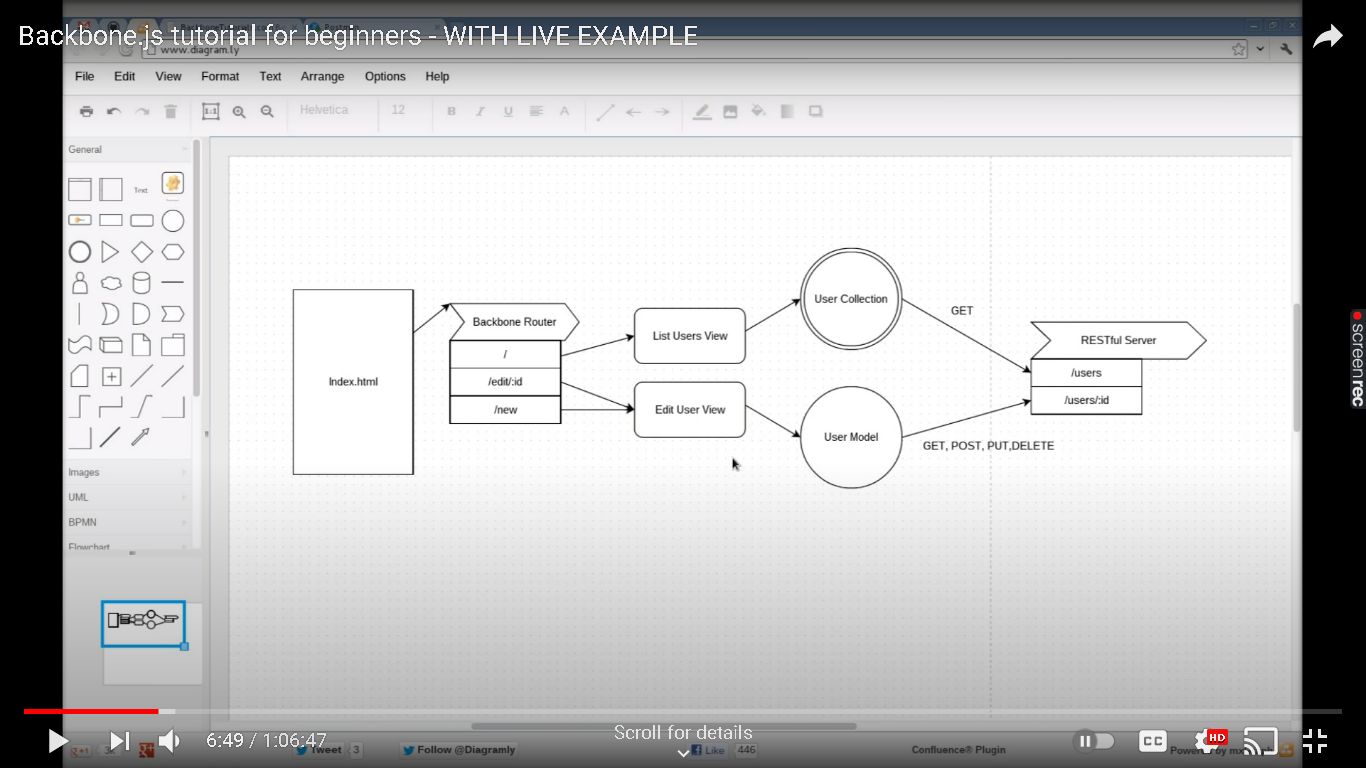
Backbone.Router povezuje određene URL adrese unutar aplikacije i oglašava događaje kada dođe do promene adrese. Pored očigledne mogućnosti da se različitim delovima aplikacije pristupa preko URL adrese i da se na osnovu parametara adrese dobijaju potrebne informacije, sa pozicije developera, Router pomaže da (potencijalno velika) aplikacija razbije na manje celine preko različitih adresa, a zatim delovi aplikacije razvijaju nezavisno od ostatka.

Unutar Backbone.Router objekta od koristi su funkcije navigate() [pomoću koje se menja trenutna adresa brauzera (logički gledano, to znači da se prelazi iz jednog dela aplikacije u drugi). Takav poziv se najčešće nalazi unutar nekog eventHandler-a koji treba da promeni trenutni pogled aplikacije.], dalje imamo execute(), kao i initialize().

[Backbone.History](http://backbonejs.org/docs/backbone.html#section-198) je objekat koji služi za upravljanje istorijom brauzera, praćenjem promena ruta (route), itd... Sadrži veći broj metoda i atributa. Značajan je metod start() koji pokreće praćenje hash promena u brauzeru.

Backbone.js forsira da se komunikacija sa serverom u potpunosti treba uraditi kroz RESTful API. Trenutno je trend na web-u da svi podaci/sadržaji budu izloženi putem API-ja. Sve ovo je potrebno zato što pretraživač nije jedini klijent, mi sada imamo mobilne i tablet uređaje, Google naočare, itd.

Na slici 2 prikazan je RESTful API.



***Slika 2 - RESTful API***

Sa leve strane slike nalazi se index.html strana, a sa desne strane RESTful Server. Oni međusobno komuniciraju pomoću komponenata koje se nalaze između njih i koje su prikazane na slici. Backbone Router kontroliše index.html stranicu i može se uočiti index,edit i new route. Index route učitaće prikaz List Users View i preko User Collection šalje zahtev RESTful Serveru za resursima. U pozadini aplikacije nalaze se edit i new route pomoću kojih možemo menjati podatke o user-u ili kreirati nove user-e. Edit User View komuniciraće sa User Model a on sa user-om i id user-a pomoću kohih koristi get, post, put i delete.

Backbone.js je neverovatno mala biblioteka za količinu funkcionalnosti i strukture koje pruža. Zasniva se na samo jednoj JavaScript biblioteci pod nazivom Underscore.js i omogućava da svi elementi aplikacije budu sinhronizovani (npr. više klijenata i server).

JSON, odnosno JavaScript Object Notation, je tekstualno baziran [otvoreni standard](https://sr.wikipedia.org/wiki/%D0%9E%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD%D0%B8_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D0%B4) dizajniran za [razumljivu ljudima](https://sr.wikipedia.org/w/index.php?title=Human-readable&action=edit&redlink=1) razmenu podataka. Ona je izvedena iz [Javaskript](https://sr.wikipedia.org/wiki/%D0%88%D0%B0%D0%B2%D0%B0%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82) jezika za predstavljanje jednostavnih [struktura podataka](https://sr.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D0%BE%D0%B4%D0%B0%D1%82%D0%B0%D0%BA%D0%B0) i [asocijativni niz](https://sr.wikipedia.org/wiki/%D0%90%D1%81%D0%BE%D1%86%D0%B8%D1%98%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B8_%D0%BD%D0%B8%D0%B7), nazvan objektima. Uprkos vezi sa JavaSkriptom, [jezički je nezavistan](https://sr.wikipedia.org/w/index.php?title=Language-independent_specification&action=edit&redlink=1), za mnoge jezike.

JSON format je prvobitno napravio [Daglas Krokford](https://sr.wikipedia.org/wiki/Douglas_Crockford), a opisan je u [RFC](https://sr.wikipedia.org/w/index.php?title=Request_for_Comments&action=edit&redlink=1) [4627](https://tools.ietf.org/html/rfc4627). Zvanična [vrsta internet medija](https://sr.wikipedia.org/w/index.php?title=Internet_media_type&action=edit&redlink=1) za JSON je application/json. Ekstenzija JSON fajlova je .json.

JSON format se često koristi za [serijalizaciju](https://sr.wikipedia.org/w/index.php?title=Serialization&action=edit&redlink=1) i prenos strukturiranih podataka preko mrežne veze. Pre svega se koristi za razmenu podataka između servera i veb aplikacije, kao zamena za [XML](https://sr.wikipedia.org/wiki/XML).

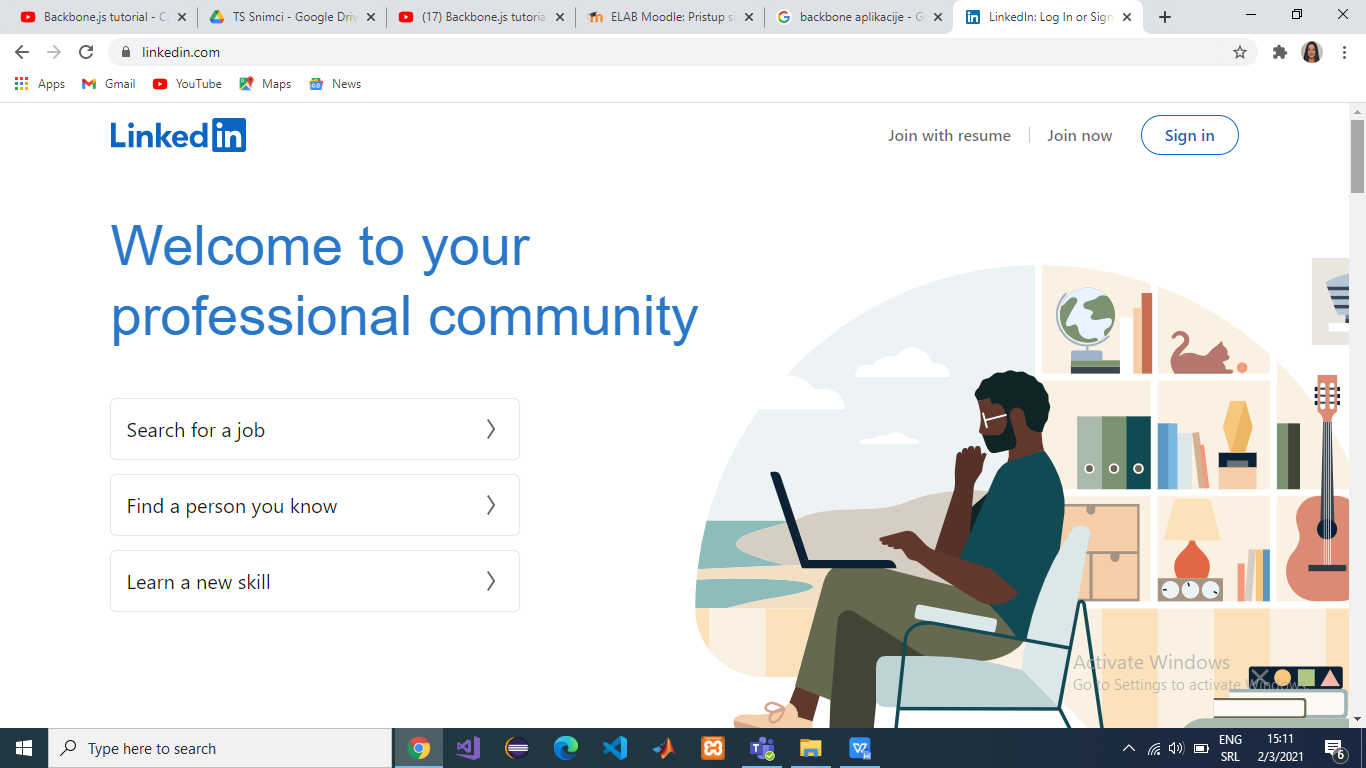
CSS ([engl.](https://sh.wikipedia.org/wiki/Engleski_jezik) Cascading Style Sheets) je jezik formatiranja pomoću kog se definiše izgled elemenata veb-stranice. CSS sintaksa se sastoji od opisa izgleda elemenata u dokumentu. Opis može da definiše izgled više elemenata, i više opisa može da definiše jedan element. Na taj način se opisi slažu jedan preko drugog da bi definisali konačni izgled određenog elementa (otuda naziv Cascading ([engl.](https://sh.wikipedia.org/wiki/Engleski_jezik) cascade - crep) da bi se dočaralo slaganje jednog stila preko drugog u definisanju konačnog izgleda elementa).

# PREGLED POSTOJEĆIH REŠENJA

Pošto je toliko lagan i bazičan, Backbone može da posluži kao osnova da se napravi sopstveni framework. Neki primeri 3rd party framework-a zasnovanih na Backbone-u su Aura, Backbone UI, Chaplin, Geppetto, Marionette, LayoutManager, Thorax, Vertebrae.

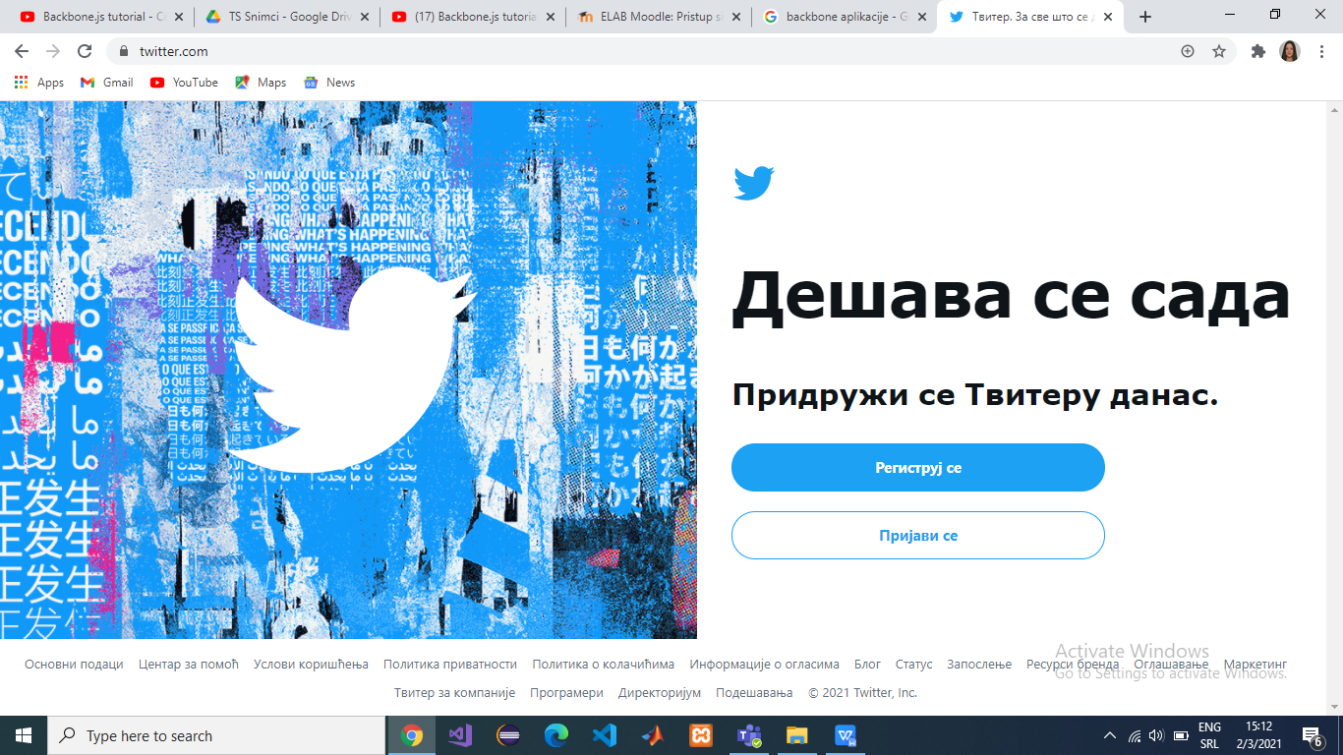
Mnoge popularne aplikacije koriste Backbone radni okvir, uključujući i Twitter, Foursquare i LinkedIn Mobile. Takođe treba spomenuti je da je veliki broj muzičkih aplikacija kreirano pomoću Backbonea, uključujući Soundcloud, Pitchfork i Pandora.

Na slici 3 je prikazan primer aplikacije koja koristi Backbone.js, LinkedIn



***Slika 3 - LinkedIn***

Na slici 4 je prikazan primer aplikacije koja koristi Backbone.js, Twitter



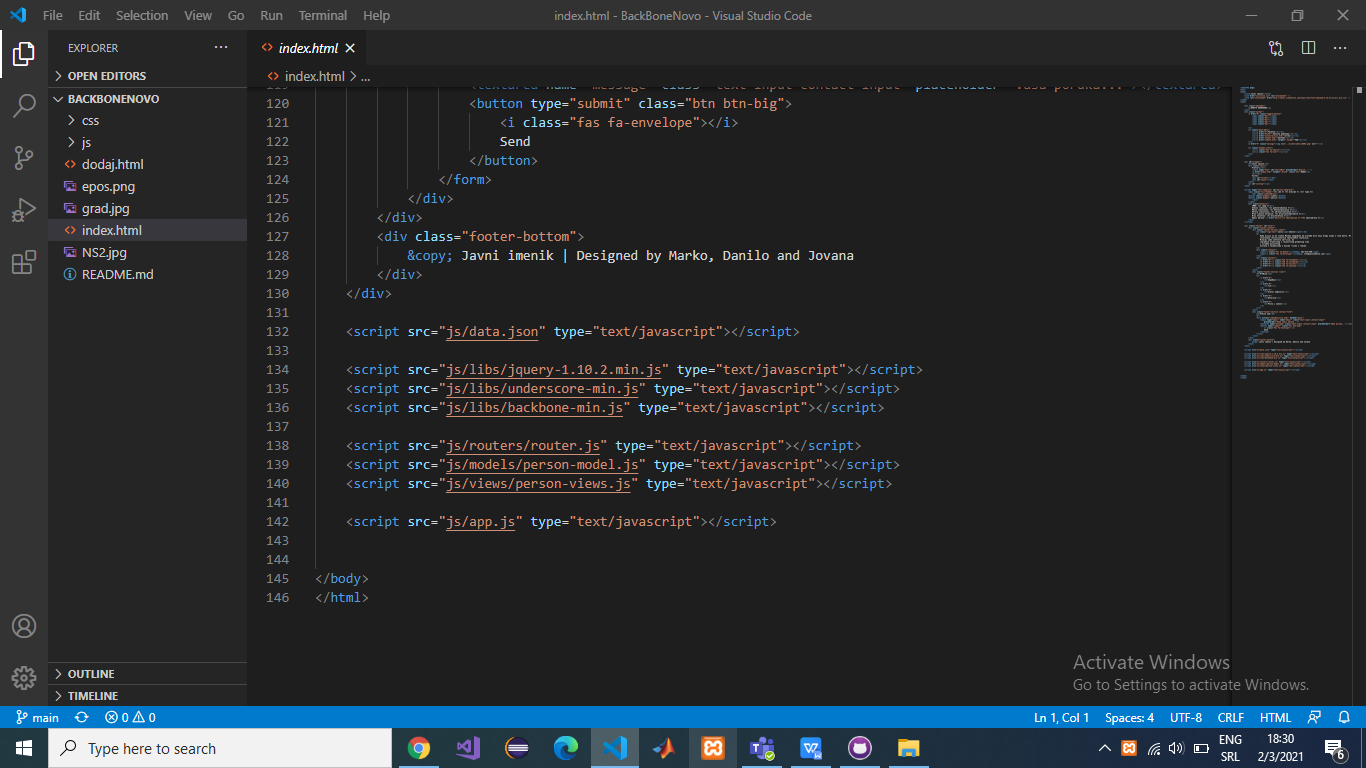
***Slika 4 - Twitter***

# RAZVOJ APLIKACIJE

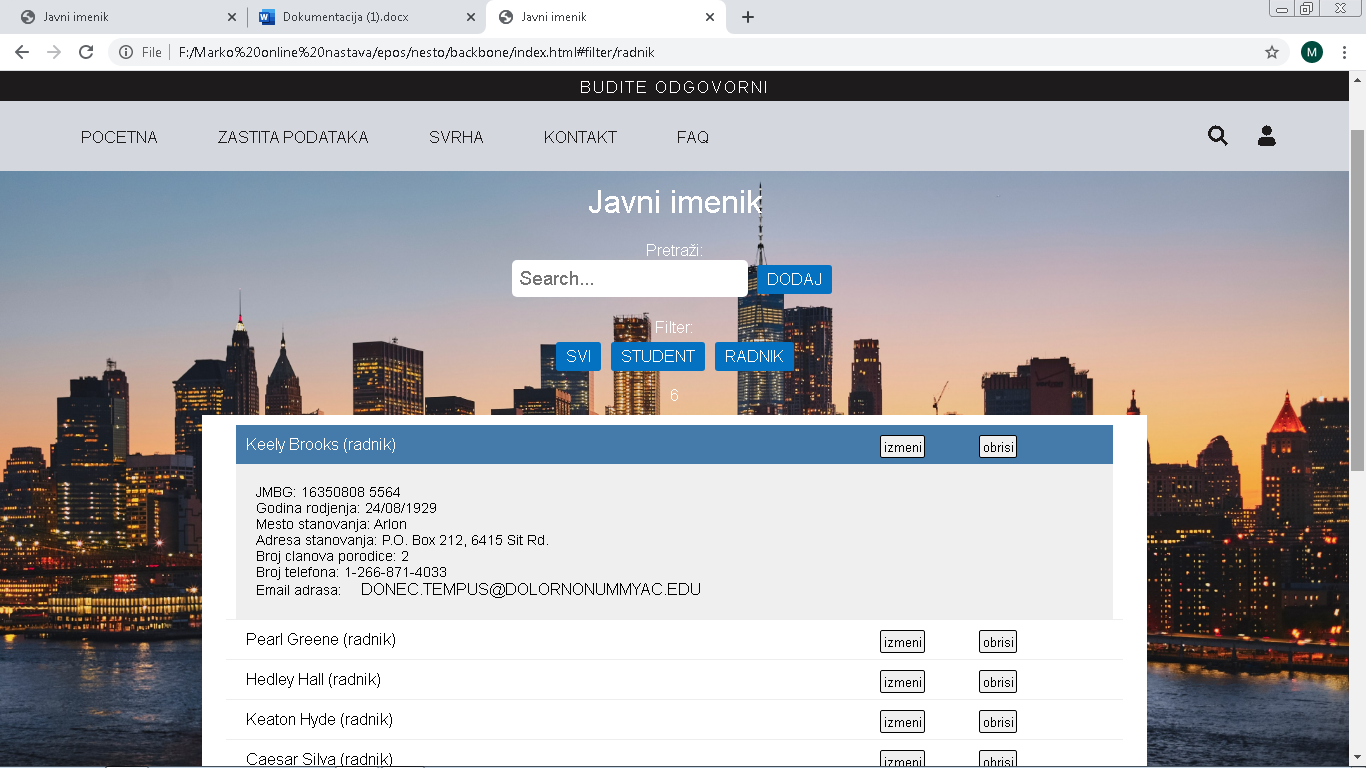
Aplikacija koja će biti detaljno predstavljena u ovom poglavlju zove se Javni imenik i treba da pruži korisniku bržu pretragu određene osobe ili dodavanje nove, izmenu i brisanje. Pruža pregledne informacije o svakoj osobi kao i detalje ukoliko se klikne na neku od ponuđenih. Moguće je filtrirati osobe prema tome da li pripadaju određenoj kategoriji ili ukucavanjem slova ili reči u polju pretraži. Pravljenje ove aplikacije u Backbone.js jeste upravo iz razloga što je njegov koncept veoma specifičan i pogodan za upotrebu, kao i lakše generisanje koda, a sam izgled aplikacije pregledan, jednostavan I razumljiv.

Pre početka kucanja pomoću Backbone.js neophodno je povezati određene biblioteke u index.html fajlu. Za to se koristi <script> tag.

Na slici 5 prikazan je kod sa korišćenim bibliotekama.

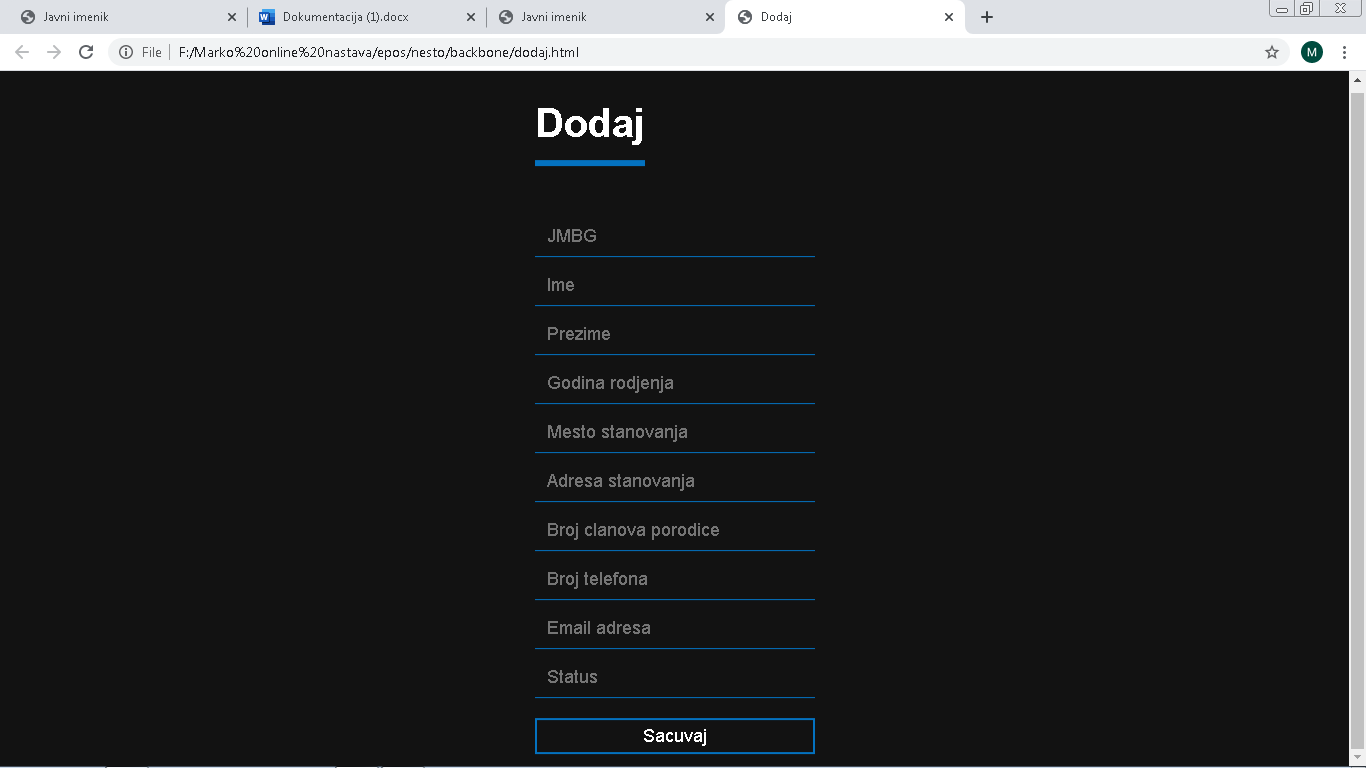
***Slika 5 - Slika programskog koda sa korišćenim bibliotekama***

Na slici 6 prikazan je izgled aplikacije.



***Slika 6 - Izgled aplikacije***

Na slici 7 prikazan je izgled aplikacije nakon što se pritisne dugme Dodaj.



***Slika 7 - Izgled aplikacije kada se pritisne dugme Dodaj***

U nastavku dat je pregled svakog fajla koji je napravljen:

*index.htm*

Kreira se jednostavan “wrapper” div tag i u okviru njega stajaće naziv same aplikacije, klasa ”tools” i u njoj Search, Filter i Count polje i na kraju neuređena lista.

*dodaj.htm*

Klikom na dugme dodaj otvara se stranica gde je potrebno popuniti datu formu podacima kako bi se dodala nova osoba. Klikom na sačuvaj osoba je dodata.

*data.json*

Generatedata.com je vebsajt za generisanje bilo koje vrste podataka koji su potrebni, tako da su izgenerisani podaci sa sajta prekopirani u novokreirani fajl.

Json podaci su stavljeni u globalnu varijablu directoryData kako bi mogli da se koriste u aplikaciji.

*router.js*

Ovde je u stvari definisana aplikacija.

(models, views, collection, roters, init: funtion)

*person-model.js*

Kreiran je model nazvan Person. Ima nekoliko default-nih vrednosti koje su kao polja napravljena u Json fajlu (JMBG, godina rođenja, mesto i adresa stanovanja, broj članova porodice, broj telefona, email adresa).

*person-view.js*

Napravljen je pregled svake osobe pojedinačno u okviru neuređene liste pomoću <li> taga. Napravljena je funkcija za prikaz detalja o osobi tako da kada se klikne na određenu osobu ispod se prikazuju podaci. Filter funkcija omogućava prikaz traženih modela tako što biramo da li želimo da se prikaže radnik ili student ili želimo da ukucamo ime odnosno prezime ili želimo da se prikažu svi modeli. Napravljena je funkcija koja broji koliko ima rezultata pretrage. Pored svakog imena nalazi se dugme izmeni I dugme obriši.

*app.js*

U ovom fajlu se pamti šta je defnisano u router.js. U init: function zapravo se pravi novi pregled ljudi i koristi se directoryData odnosno varijabla u kojoj su sačuvani Json podaci.

# ZAKLJUČAK

U ovom seminarskom radu detaljno je objašnjeno korišćenje JavaScript biblioteke Backbone.js, kako radi i na koji način se koristi, ilustrativno je predstavljen koncept i komunikacija sa serverom, osnovne karakteristike, za izradu kakvih aplikacija je pogodan i preporučljiv. Navedeni su primeri aplikacija gde je već korišćen kao i primer aplikacije Javni imenik koju smo implementirali i opisali kroz ovaj rad. Backbone.js veoma je specifičan i lak za korišćenje, kod je organizovan i upravo zbog svih navedenih karakteristika, veoma je preporučljivo koristiti ovu biblioteku kada god je to moguće.

# LITERATURA

1. Angular, Ember I Backbone, Preuzeto 3.2.2021. <https://saveti.kombib.rs/angular-ember-i-backbone-koje-javascript-resenje-je-pravo-za-vas.html>
2. BackboneWorkshop, Preuzeto 3.2.2021. <http://djordjeparovic.github.io/backboneWorkshop/>
3. Backbone.js, <https://backbonejs.org/>
4. <https://www.tutorialspoint.com/backbonejs/backbonejs_overview.htm>
5. <https://stackoverflow.com/questions/16544984/how-backbone-js-model-fetch-method-works>
6. <https://www.javatpoint.com/what-is-backbonejs>
7. <https://backbonejs.org/docs/backbone.html>
8. <https://www.w3schools.com/>