SVEUČILIŠTE U ZAGREBU
FILOZOFSKI FAKULTET
ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI
SMJER Nastavnički
Ak. god. 2018./2019.

Ian Christian Hanser

LMS Infostudent - web sučelje i mobilna aplikacija

Diplomski rad

Mentor: dr.sc.Krešimir Pavlina

Zagreb, 2019.

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Sadržaj

1. Uvod

Razvoj računalnih i web tehnologija nam danas dopušta izradu jednostavnih sustava kojima je moguće pristupiti putem različitih uređaja. Cilj ovoga rada je objasniti osnove web tehnologija koje se koriste pri izradi poslužiteljske arhitekture i web sučelja. Rad će također istražiti mogućnost izrade hibridnih mobilnih aplikacija temeljenih na istim tehnologijama.

U prvome poglavlju rada ću objasniti osnovna ograničenja, prednosti i nedostatke REST poslužiteljske arhitekture. Svako od ograničenja vezanih uz ovu vrstu arhitekture ću detaljnije opisati i predstaviti kako funkcioniraju osnovne interakcije s poslužiteljem. Također ću opisati načine na koji se odvija komunikacija između klijenta i poslužitelja.

U drugome poglavlju rada ću ukratko opisati osnovne web tehnologije koje sam koristio za izradu sustava InfoStudent, namijenjenog studentima i profesorima Filozofskog fakulteta. Ukratko ću opisati HTML jezik za označavanje, CSS jezik za izradu stilova i JavaScript programski jezik za programiranje klijentskog dijela sustava. Osim osnovnih web tehnologija ću opisati jQuery library koji olakšava programiranje sučelja i Bootstrap framework namijenjen za izradu responzivnog sučelja. Također ću objasniti karakteristike JSON web tokena koji služi za autentikaciju korisnika i Googleovu platformu Firebase.

Treće poglavlje rada sadrži opis Apache Cordova frameworka koji je namijenjen za izradu hibridnih mobilnih aplikacija temeljenih na HTML5 jeziku za označavanje. Ukratko ću opisati povijesni razvoj ovog frameworka i njegove karakteristike, tehnologije, priključke, mogućnosti i platforme koje pokriva.

U četvrtom poglavlju ću prikazati izgled web sučelja i mobilne aplikacije sustava InfoStudent. Ovo poglavlje sadrži slike web sučelja u funkciji i kratak opis ovlasti različitih korisnika (studenata i profesora.) Osim web sučelja, ovo poglavlje također sadrži slike mobilne aplikacije u funkciji.

2. REST api

Pozadinski servis je napravljen kao REST api, on podrazumijeva softwaresku arhitekturu koja se sastoji od skupa ograničenja koji se koriste za izradu web servisa. Ovakva vrsta web servisa dopušta interoperabilnost između različitih računalnih i mobilnih sustava (samom API-ju je moguće pristupiti neovisno o web pregledniku ili vrsti uređaja.) Oni također omogućuju prikaz i manipulaciju web resursa uz pomoć unaprijed definiranog skupa pravila. REST API prati strukturu orijentiranu na resurse (Resource Oriented Architecture) i nju možemo smatrati kao setom pravila za dizajniranje RESTful web servisa[1].

2.1. Jednostavnost i uniformnost sučelja

Uniformnost sučelja podrazumijeva četiri ograničenja vezana za hipermedije koji stvaraju jasnu razdiobu između koda koji se poziva na klijentskom dijelu i koda koji se nalazi na poslužitelju sustava. Ta ograničenja su sljedeća[2]:

1. Stanje aplikacije se održava na uređaju klijenta, zadatak klijentskog dijela je prikazivanje novih stanja
2. Klijent stanje može jedino mijenjati slanjem upita na poslužitelj i obradom primljenog odgovora
3. Klijent može vidjeti opcije kroz hipermedije koje je prihvatio do određenog trenutka (svaki gumb ili poveznica predstavljaju određeni zahtjev prema poslužitelju)
4. Hipermediji trebaju biti zaslužni za svaku promjenu u stanju aplikacije

REST arhitektura se oslanja na čiste, odnosno RESTful URLove. URL je skraćenica za Uniform Resource Locator, odnosno, web adresu određenog resursa na internetu, ona može biti HTML dokument, PHP skripta, slika ili bilo koja druga datoteka koja se nalazi na određenom web poslužitelju Za njih se propisuje pravilo da trebaju biti deskriptivni, odnosno sami URLovi trebaju biti intuitivni krajnjim korisnicima[3]. Oni su namijenjeni za poboljšanje korisnosti i pristupačnosti nekoj web stranici ili web servisu tako što su odmah shvatljivi čak i neiskusnim korisnicima. Uz korištenje ovakvih URL predložaka, moguće je razdvojiti korisničko sučelje za pozadinski servis i prikazati strukturu nekog resursa putem URLa. Ovakav način prikaza informacija je također u skladu sa SEO standardima. Search Engine

[1] Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007. str. 107 - 109
[2] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 348
[3] Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007.str. 83

Optimization je optimizacija pretraživača web stranice, ona uključuje niz aktivnosti i olakšava krajnjim korisnicima jednostavnije pretraživanje web stranice. Te aktivnosti su također često usmjerene prema podizanju broja posjećenosti stranice i lakšem indeksiranju od strane tražilica. olakšava pretraživanje web pretraživačima, te omogućava konzistentnost kod prikaza jednog resursa kroz uvijek isti URL.

Neočišćeni URL	Očišćeni URL
`http://adresa.com/vijesti/index.php?stranica=5`	`http://adresa.com/vijesti/5`
`http://adresa.com/karte/pretraga.php?zemlja=Hrvatska&grad=Zagreb`	`http://adresa.com/karte/pretraga/Hrvatska/Zagreb`
`http://adresa.com/korisnik/pretraga.php?ime=Ivo&prezime=Ivić&stranica=1`	`http://adresa.com/korisnik/pretraga/Ivo/Ivić/1`

Tablica 1. Razlike između očišćenog i neočišćenog URLa

Generiranje ovakvih pristupnih točaka se postiže uz proces koji se naziva URL mapping i on je najčešće zapisan unutar .htaccess konfiguracijske datoteke. To je datoteka koja se koristi na poslužiteljima za mijenjanje konfiguracije poslužitelja i dodavanje dodatnih funkcionalnosti. Neočišćeni URL se najčešće sastoji od putanje do skripte, imena skripte i varijabli upita. Parametri upita odlučuju koji resursi će biti prikazani korisnicima i često se sastoje od nečitljivih numeričkih ili tekstualnih vrijednosti koje reprezentiraju vrijednost resursa unutar baze podataka. Čisti URLovi se sastoje samo od putanje do određenog resursa u skladu s logičkom strukturom koju korisnici mogu lagano shvatiti i mijenjati.

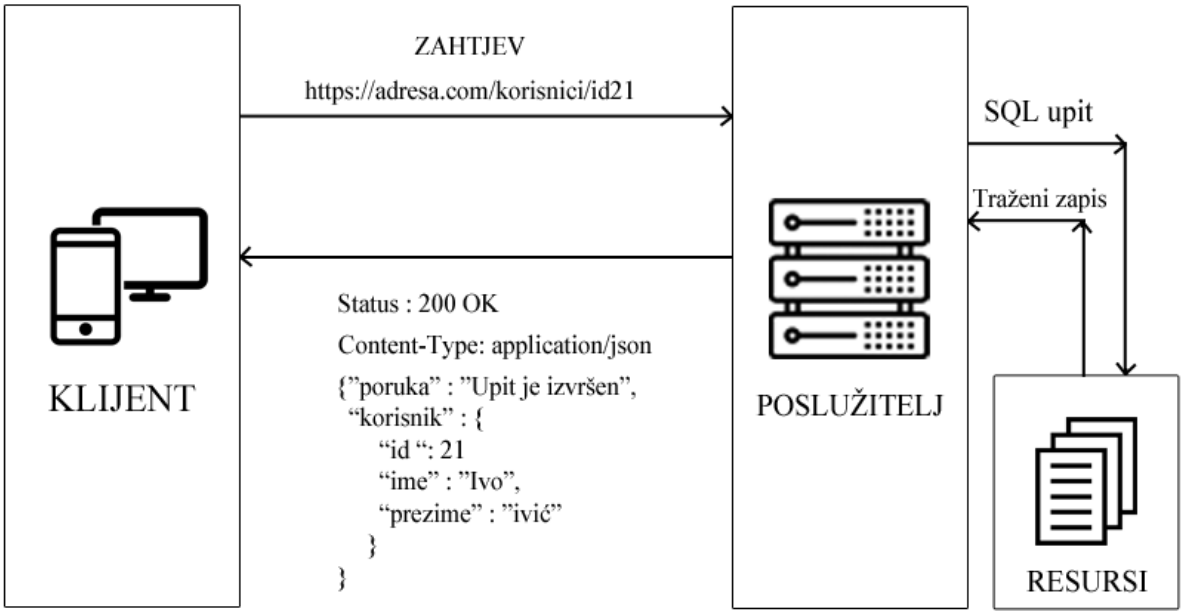
2.2. Klijentsko-poslužiteljska arhitektura

Klijentsko-poslužiteljska vrsta arhitekture podrazumijeva podjelu određenih zadataka između korisničkog sučelja i pozadinskog servisa. Pozadinski servis u ovom slučaju služi za skladištenje resursa, obradu korisnikovih zahtjeva i manipulaciju resursa na prema korisnikovim zahtjevima[4]. On se sastoji od krajnjih točaka koje su namijenjene za izvršavanje predodređenih akcija na poslužitelju. Takve akcije mogu uključivati: CRUD operacije ili složenije operacije kao pretraživanje prema jednom ili više parametara ili izvršavanje više upita na poslužitelju unutar jednog poziva. CRUD operacije (Create, Read, Update, Delete) su četiri osnovne operacije u računalstvu koje osiguravaju dosljednost podataka. One se

[4] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 349

koriste u okviru poslužitelja i nalažu da svaki resurs zapisan u bazi podataka može učitati, ažurirati i izbrisati, te da je moguće stvarati nove resurse.

Poslužitelj podatke obrađuje u njihovom najosnovnijem obliku i ne dodaje nikakve dodatne oznake (npr. HTML oznake) prije odgovora klijentu. Kod odgovora klijentu ovakav poslužitelj će odgovoriti u XML ili JSON formatu, što omogućuje slanje dodatnih informacija kao što su tokeni za osvježavanje, OAuth tokeni, JWT tokeni ili povratne poruke korisniku uz same zatražene resurse. Ovakav način baratanja podatcima nam omogućava prijenos što manje količine podataka između korisnika i poslužitelja i u konačnici dovodi do bržeg prijenosa istih.



Slika 1. Komunikacija između klijenta i poslužitelja

Klijenti mogu pristupati pozadinskom servisu neovisno o uređajima koje posjeduju (mobilni uređaji, tableti, računala.) Zahtjevi se na pozadinski servis šalju u Ajax pozivima[5], dok korisničko sučelje služi za prikaz dohvaćenih podataka na uređaju korisnika. Pri svakome zahtjevu, klijentski dio aplikacije šalje upit prema poslužitelju koji ga obrađuje. Upit se sastoji od podataka koji su nečitljivi ljudima, ali su razumljivi računalima. Poslužitelj na temelju primljenih podataka pokušava izvršiti određenu radnju ili korisniku prikazati neki resurs. Ako je radnja uspješna, poslužitelj odgovara klijentu šaljući mu izvještaj o radnji, prikaz trenutnog stanja jednog ili više resursa. Ako je radnja neuspješna ili je došlo do greške na poslužitelju, klijentu će biti poslan izvještaj o grešci. U REST arhitekturi, odgovor

[5] Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007. str. 315-317

poslužitelja je najčešće zapisan u XML ili JSON formatu,dok je zadatak klijentskog dijela prikaz informacija na vizualno intuitivan način[6].

2.3. Odsutnost stanja[7]

Odsutnost stanja (**statelessness**) označava da se svaki HTTP zahtjev odvija u izolaciji, odnosno, u svakom zahtjevu koji korisnik pošalje se nalaze sve informacije koje su potrebne za izvršenje tog upita. To označava odsutnost sesije i bilo kakve vrste informacija koje se bilježe na poslužitelju. Podatci koje se inače nalaze u sesiji, (id korisnika, ime, prezime, email adresa...) se u ovom slučaju šalju na poslužitelj od strane klijenta tako da je svaki dio upit prema poslužitelju u potpunosti izoliran, te da se ne referira niti na jedan od prethodno poslanih upita[8]. REST arhitektura ovaj problem zaobilazi pohranjivanjem unutar kolačića, ili lokalne pohrane (LocalStorage.)

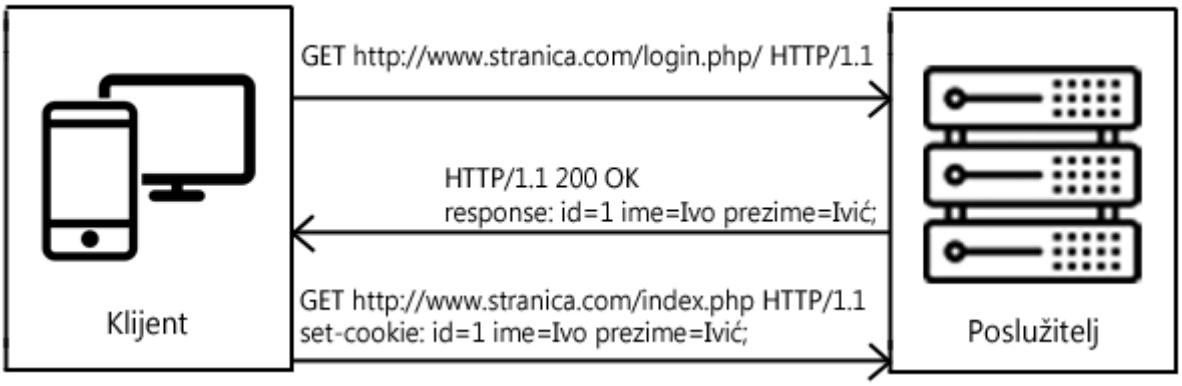
Kolačići[9] predstavljaju malu količinu podataka koju klijent automatski šalje poslužitelju pri svakom zahtjevu unutar set-cookie HTTP zaglavlja. U njima je moguće poraniti određene informacije o trenutnom stanju korisnika koji upućuju na to je li korisnik prijavljen u aplikaciju ili bilo koje drugo stanje (npr. stanje korisnikove košarice u slučaju online trgovine.) Kolačićima je također moguće odrediti vrijeme trajanja, nakon kojeg će se kolačić sam uništiti. Pri korištenju kolačića, maksimalna dopuštena veličina je otprilike 4 kilobajta (4093 bajta). Ako želimo dodatno osigurati kolačiće, možemo koristiti httponly opciju unutar set-header zaglavlja. Ta opcija osigurava da su kolačići poslani preko sigurnog kanala i onemogućavaju mijenjanje zaglavlja od strane korisnika i sprječavaju XSS napade. Neki autori nalaze primjedbu pri korištenju kolačića da oni krše pravilo odsutnosti stanja. U kasnijem dijelu rada će biti opisano korištenje Authorization zaglavlja koje predstavlja siguran i standardan način prilaganja identifikacijskih tokena.

[6] Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007. str. 86 - 89

[7]Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007. str. 86 - 89

[8] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 349

[9] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 287-289



Slika 2. Primjer pohrane kolačića

LocalStorage[10] je najjednostavniji, sinkroni način pohrane podataka koji funkcionira na principu vrijednosti i ključa. Ona se razlikuje od kolačića u tome što nije vremenski ograničena, odnosno, podatci se neće samostalno uništiti nakon određenog perioda. Ovi podatci se ne dodaju samostalno pri svakom HTTP zahtjevu i ako postoji potreba za njihovim slanjem na poslužitelj, to je potrebno učiniti putem koda. Prema veličini, on je veći od kolačića i maksimalna količina podataka koju može pohraniti je 5 megabajta. Upravo radi veće mogućnosti pohrane podataka, lokalnu pohranu možemo koristiti kako bi pohranili određene podatke za kasniju uporabu. Pri korištenju je potrebno posvetiti dodatnu pozornost upravljanjem samim podatcima upravo radi odsustva vremena isteka, također je preporučeno ne pohranjivati tokene ili bilo kakvu drugu vrstu osjetljivih informacija jer ih korisnik može lagano dohvatiti pokretanjem vlastite skripte na stranici.

Tokeni

Kako bi aplikacija bila sigurnija, preporučuje se korištenje tokena umjesto čistih vrijednosti koje predstavljaju korisnika. Token predstavlja podatke koje korisnik prima nakon prijave unutar aplikacije, sam token je nečitljiv ljudima, ali čitljiv računalima. Većina tokena se sastoji od podataka koji su vezani uz korisnika, podataka o izdavaču tokena, vremenu izdavanja i vremenu isteka. Token možemo zakriti uz pomoć algoritma za šifriranje kako bi korisniku token izgledao kao nasumičan niz znakova i kako bi samom korisniku otežao falsificiranje istog (korisnik ne može falsificirati token bez tajnog ključa koji je pohranjen na poslužitelju.)[11] Šifriranje je reverzibilna funkcija što znači da korisnik može saznati sadržaj

[10] Mark Lassoff, Tom Stachowitz. Mobile App Development with HTML5. 1st Edition. Conneticut : LearnToProgram.tv, incorporated. 2015. str. 154

[11] Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007. str. 253-254

tokena ako posjeduje ključ koji je pohranjen na poslužitelju i zna koji je algoritam šifriranja korišten[12].

Osim šifriranja, moguće je koristiti hash funkciju pomoću koje možemo potvrditi da je token zaista izdan od strane određenog poslužitelja. U ovome slučaju cijeli sadržaj tokena će biti dostupan klijentskom dijelu aplikacije (istovremeno i korisniku.) Kako bi se spriječilo falsificiranje, ovi tokeni su osigurani s digitalnim potpisom koji se generira na temelju sadržaja i tajnog ključa. Pri svakom zahtjevu koji se šalje na poslužitelj, korisnik prilaže i svoj identifikacijski token. Prije izvršavanja zahtjeva, poslužitelj provjerava je li token još uvijek valjan (je li token istekao) i vjerodostojnost tokena na temelju njegovog digitalnog potpisa. Vjerodostojnost se provjerava tako da će poslužitelj usporediti digitalni potpis tokena s potpisom koji će pokušati generirati na temelju poruke unutar tokena i tajnog ključa. Ako se ta dva potpisa ne podudaraju, poslužitelj može zaključiti da je token falsificiran i neće izvršiti zatraženi zahtjev. Primjer jedne vrste takvog tokena (JSON web token) će biti opisan u sljedećem poglavlju rada.

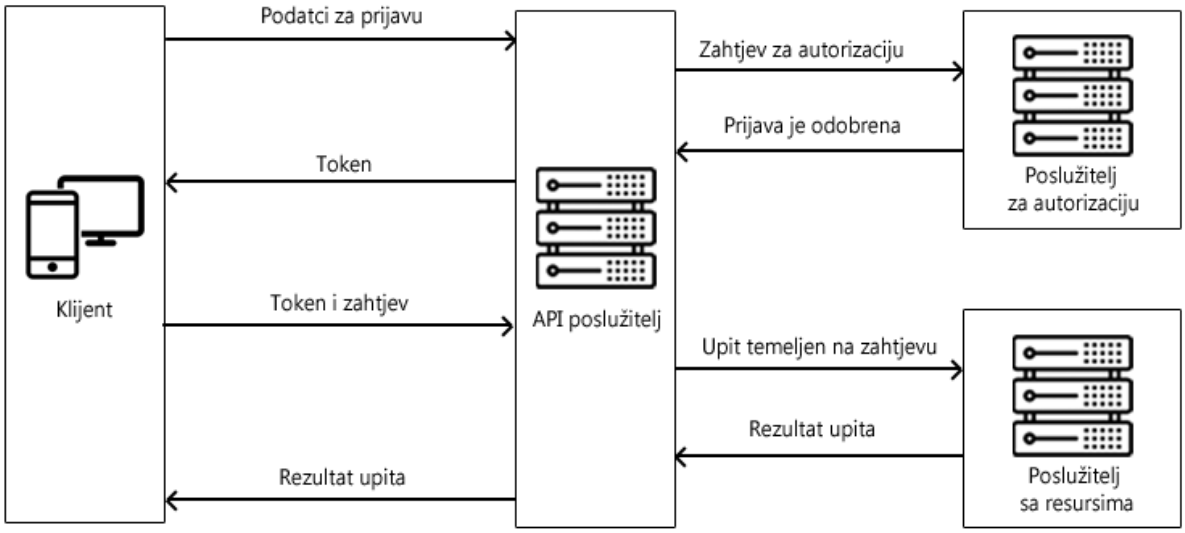
Također valja napomenuti kako se ovi tokeni mogu biti pohranjeni unutar set-cookie zaglavlja, ali se preporučuje pohranjivati ih unutar Authorization zaglavlja. To zaglavlje je rezervirano za pohranu identiteta i ovlasti korisnika koji šalju zahtjev. Ono služi poslužitelju za provjeru ima li korisnik ovlasti pristupati određenom resursu. Ako korisnik ne priloži akreditiv za autorizaciju ili je sam akreditiv nevažeći ili falsificiran, poslužitelj će odgovoriti sa statusom 401 Unauthorized[13]. Format tog zaglavlja je sljedeći: **Authorization: <tip> <akreditiv>**. Dio koji je označen sa <tip> označava vrstu autorizacijskog zaglavlja i on može poprimiti sljedeće vrijednosti: Basic, Bearer, Digest, Mutual, OAuth, SCRAM-SHA-1, SCRAM-SHA-2516 i Vapid. Zadana vrijednost je Basic i pri korištenju JSON Web tokena se preporučuje Bearer tip, dok se u slučaju OAuth tokena preporučuje OAuth tip. Na mjesto označeno sa <akreditiv> se zapisuje čista vrijednost akreditiva (najčešće tokena.)

2.4. Slojevitost sustava[14]

Slojevitost sustava označava da korisnik ne može spoznati je li priključen na krajnji poslužitelj ili na neki od posredničkih poslužitelja. To također označava da se krajnji

[12] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 249-252
[13] Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007. str. 249-250
[14] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 351-352

poslužitelj može spajati na druge poslužitelje kako bi se određeni zadatci izvršili. Na primjer, možemo zamisliti tri poslužitelja. Prvi poslužitelj na kojemu se nalazi pohranjen API i ostatak programske podrške. Drugi poslužitelj može posjedovati sve podatke o korisnicima i služiti kao server za autorizaciju koji korisniku izdaje token za potvrdu da je korisnik prijavljen u aplikaciju i provjerava valjanost korisnikovog tokena u svakom sljedećem pozivu. Treći poslužitelj može sadržavati podatke, odnosno, resurse kojima aplikacija može pristupati[15].



Slika 3. Prikaz komunikacije u slojevitom sustavu

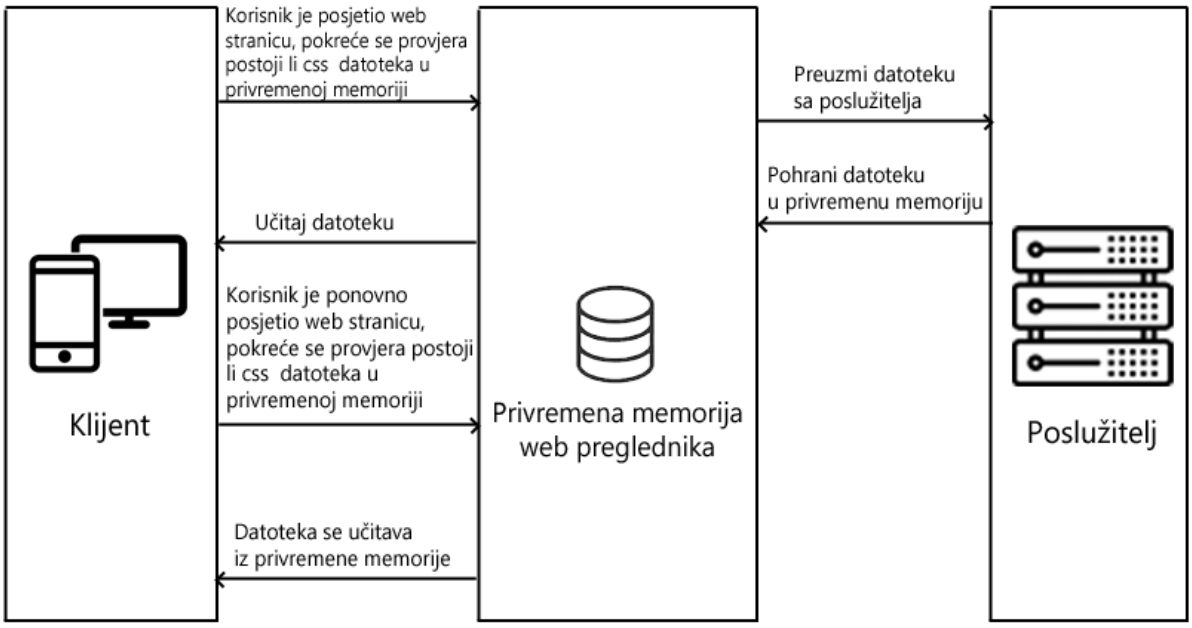
Kao što je prethodno spomenuto, prednost ovakvog sustava je što korisnik ne zna na koju je krajnju točku spojen. To pridonosi sigurnosti sustava, jer korisniku otežava falsificiranje tokena i SQL injekciju. Osim sigurnosti, ovaj pristup olakšava posao web razvojnim programerima jer je programska logika razdvojena na njene najosnovnije dijelove. Radi te razdvojenosti, web developeri mogu izvršavati nadogradnju određenih dijelova sustava bez da utječe na ostatak. Na primjer, moguće je dodati nove resurse na poslužitelj bez da se utječe na logiku autorizacije ili autentikacije.

2.5. Korištenje brze privremene memorije (cache)

Korištenje brze privremene memorije olakšava učitavanje web aplikacije krajnjim korisnicima, ono podrazumijeva pohranjivanje često korištenih datoteka i resursa na tvrdi disk korisnika[16]. Ono uključuje pohranjivanje datoteka vezanih uz korisničko sučelje koje uključuju: HTML, CSS i JavaScript datoteke i ostale resurse koji mogu uključivati: ikone,

[15] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 351
[16] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 351

slike, video i audio zapise, json datoteke i XML datoteke. Pohranjivanje navedenih resursa se izvršava automatski, te je moguće dodatno opisati samu logiku pohrane, dodatan opis ovakve logike olakšava svaku buduću nadogradnju sustava tako da korisnik pri učitavanju stranice preuzima ranije spomenute resurse samo ako je dostupna nova verzija istih.



Slika 4. Rad privremene memorije web preglednika

Osim datoteka korisničkog sučelja, na disk korisnika je moguće pohraniti i rezultate određenih zahtjeva koji su upućeni na poslužitelj. Oni se ne pohranjuju na tvrdi disk automatski, već je potrebno opisati logiku za takvo pohranjivanje. Najveći izazov u pisanju ovakve logike je ostvarivanje konzistentnosti podataka koji će biti prikazani korisnicima, prepreka je razaznati je li traženi resurs bio ikako izmijenjen ili pobrisan od strane korisnika koji ima ovlasti mijenjati spomenute resurse ili od strane administratora sustava.

2.6. Pozivanje koda na zahtjev (Code on demand)[17]

Jedna od karakteristika rest sustava uključuje pozivanje koda na korisnikov zahtjev. Ova komponenta označava da će sa na klijentu izvršiti određeni kod dok je aplikacija pokrenuta. Važno je napomenuti da se taj kod ne nalazi unutar web ili mobilne aplikacije, već se šalje od strane poslužitelja prema klijentu. Izvršavanje takvog koda je moguće pokrenuti unutar HTML-ove <script> oznake. Od dosad navedenih ograničenja REST sučelja, ovo ograničenje nije obavezno već je opcionalno.

[17] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 352-353

Ovoj proces razvojnim programerima omogućuje izradu sustava koji će zaobilaziti privremenu memoriju računala i sam će pokretati ažuriranje. Taj pristup olakšava nadogradnu sustava i omogućava brzi prijelaz na nove verzije bez bojazni da će korisnik imati pohranjenu krivu verziju određenog koda.

2.7. Sigurnost REST API-ja

REST API koristi nekoliko jednostavnih predložaka kako bi se zaštitio od SQL injekcije, XSS i CSFR napada. Osnovu ovakve zaštite predstavljaju HTTP metode i autorizacijski tokeni. Najosnovnije HTTP metode su sljedeće[18]:

- GET (učitavanje)
- POST (stvaranje)
- PUT (zamjena ili ažuriranje)
- DELETE (brisanje)

Pri svakom zahtjevu prema poslužitelju unutar zaglavlja označenog sa "General" klijent prilaže jednu od navedenih metoda. Poslužitelj pri primitku zahtjeva provjerava koja je metoda zapisana unutar zaglavlja i provjerava priloženi token. Na temelju tokena, poslužitelj može provjeriti ima li korisnik ovlasti za izvršavanje traženog zahtjeva. Na ovaj način možemo ograničiti kojim je korisnicima dopušteno brisati ili uređivati resurse koji se nalaze unutar baze podataka (te ovlasti su najčešće rezervirane za administratore sustava ili napredne korisnike kojima su spomenute radnje omogućene.) Na ovaj način je također moguće odrediti koje će metode biti prihvaćene od strane poslužitelja.

SQL injekcija označava metodu kojom je moguće dohvatiti informacije iz baze podataka ili brisanje istih. Ona funkcionira tako da korisnik u bilo koje unosno polje unese SQL kod koji potencijalno briše jedan ili više resursa unutar jedne tablice ili ispisuje podatke o jednom ili više korisnika. Sprječavanje SQL injekcije se vrši tako da se svaki korisnikov unos pročisti prije izvršavanja SQL upita. To možemo učiniti korištenjem unaprijed napisanih iskaza (eng. prepared statements) ili korištenjem pohranjenih procedura. Ovi procesi osiguravaju da se svaki korisnikov unos shvaća kao niz znakova, a ne kao dio koda koji potencijalno može naštetiti aplikaciji. Također je moguće zaštiti se od provale grubom silom (engl. brute force.) Poslužitelj može promatrati s koje IP adrese dolazi određeni zahtjev i ako se s iste adrese šalje veliki broj upita na jedan resurs, poslužitelj može blokirati istu IP adresu ili postaviti

[18] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 33

određeni period čekanja nakon nekoliko uzastopnih zahtjeva. Ako određeni korisnik pokušava pristupiti prijavi više puta unutar kratkog perioda s nevažećim podatcima, poslužitelj može prisiliti klijenta da pričeka određeni period. Isti princip se može koristiti na bilo kojem zahtjevu, te ako se posumnja da korisnik preko određenog uređaja šalje velik broj zahtjeva, moguće je blokirati IP adresu istog.

Kroz zaglavlja je također moguće odrediti koji format poslužitelj može prihvatiti (unutar Request zaglavlja) i u kojem formatu će biti isporučen rezultat (unutar Response zaglavlja.) Poslužitelj treba odbiti svaki zahtjev koji nije strukturiran prema pravilima postavljenim unutar tih zaglavlja, odnosno, poslužitelj ne smije prihvatiti unos za koji se tvrdi da je zapisan u JSON formatu bez da izvrši dodatnu provjeru istog. Poslužitelj također treba odbiti svaki zahtjev u kojemu se tokeni šalju unutar URLa zahtjeva (u slučaju GET metoda) jer postoji unaprijed određeno zaglavlje unutar kojeg se spomenuti tokeni smiju slati (najčešće unutar set-cookie i Authorization zaglavlja.) Poslužitelj također može priložiti X-Content-Type-Options:nosniff u zaglavlje kako bi onemogućio klijentskom pregledniku detekciju poslanih formata, što onemogućuje XSS napade.

Ako podatci na poslužitelju nisu dostupni samo za učitavanje, već je omogućeno brisanje, ažuriranje i slične akcije, potrebno je dodatno obratiti pozornost na sigurnost transportnog sloja. Ta sigurnost se može poboljšati korištenjem poslužiteljskih certifikata, odnosno, digitalnih potpisa. Ti certifikati se sastoje od imena poslužitelja, izdavača certifikata i javnog ključa poslužitelja. Na ovaj je način moguće sigurno slati informacije samo klijentima koji posjeduju digitalni certifikat i osigurati se od neželjene uporabe podataka s poslužitelja. Kako bi se sustav dodatno osigurao preporučuje se enkripcija podataka na samome poslužitelju, odnosno u bazi podataka i dekripcija podataka prije samog odgovora klijentu.

REST API se uvelike oslanja na korištenje statusnih kodova, odnosno, kodova koji opisuju rezultat određenog zahtjeva, u sljedećoj tablici su popisani često korišteni kodovi[19]:

Statusni kod	Značenje
200 OK	Poziv koji je korisnik poslao prema poslužitelju je uspješno izvršen u potpunosti, HTTP metoda može biti GET, POST, PUT, PATCH ili DELETE
400 Bad Request	Zahtjev nije pravilno formiran, poslužitelj je primio podatke u krivom formatu ili je zahtjev nepotpun.
401 Unauthorized	Korisnik nije priložio akreditiv (token ili identifikaciju korisnika) uz zahtjev.
403 Forbidden	Korisnik nema ovlasti da pristupi traženom resursu.
404 Not found	Korisnik je zatražio ispis ili pokušao obrisati ili ažurirati nepostojeći resurs
405 Method Not Allowed	Korisnik je u zahtjevu definirao krivu HTTP metodu, ova greška se javlja ako poslužitelj ne dopušta korištenje priložene metode nad određenim resursom.
429 Too Many Requests	Ova greška se javlja ako isti korisnik šalje previše zahtjeva na poslužitelj. Koristi se za sprječavanje potencijalnog DOS napada ili pri preopterećenosti poslužitelja.
500 Internal Server Error	Na poslužitelju je došlo do greške, ovaj kod se najčešće koristi kada je poslužitelj primjetio neočekivani uvjet koji onemogućuje potpuno izvršenje upita.

Tablica 2. Često korišteni statusni kodovi

[19] Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013. str. 305 - str. 3015

3. Tehnologije korištene u izradi web sučelja

3.1. HTML

HTML je skraćenica za HyperText Markup Language, odnosno, jezik za označavanje hiperteksta. Hipertekst označava strukturu međusobno povezanih točaka u web okruženju i omogućava prikaz određenih stranica ili resursa (hipermedija) putem hiperveza ili poveznica. Osnovni element ovog jezika je oznaka (engl. tag.) Sve oznake se zapisuju unutar izlomljenih zagrada, (< i >) svaku oznaku je potrebno otvoriti i zatvoriti. Sadržajem oznake smatramo podatke koji su zapisani nakon otvaranja i prije zatvaranja određene oznake. HTML dokument obavezno treba sadržavati sljedeće tri oznake:

- html - Otvaranje ove oznake označava početak html dokumenata i ostale oznake koje su ugniježđene unutar nje se tretiraju kao HTML elementi
- head - Označava zaglavlje web stranice i unutar nje se pohranjuju metapodatci o posjećenoj stranici, unutar njega se zapisuju podatci poput: naslova stranice, imenu autora web stranice i reference na css i JavaScript dokumente
- body - Označava tijelo dio web stranice, odnosno elemente in slijed kojim će biti prikazani na ekranu korisnika

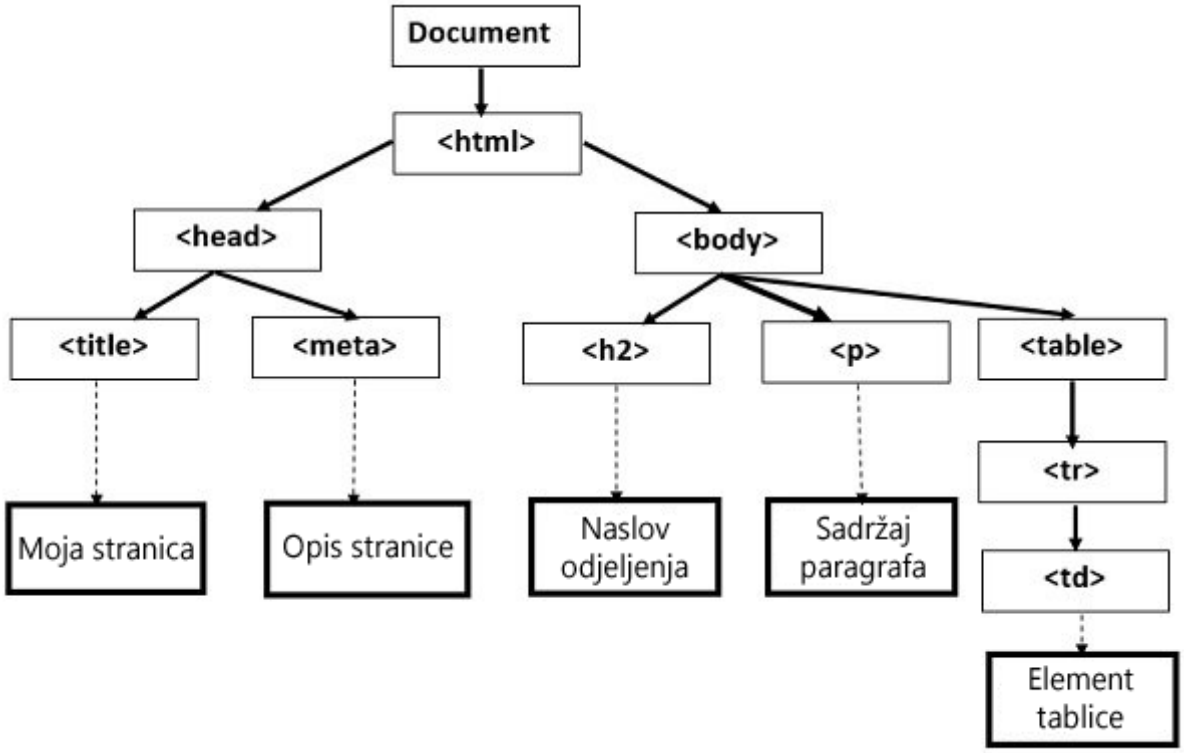
Često korištene oznake uključuju:

naslove	Naslov
podnaslove	Podnaslov
paragrafe	Paragraf
slike	
polja za unos	<input type="text"></input>
tablice	<table> </tr> <td> </td> </tr> </table>
Poveznice	 youtube poveznica
Sekcije	Naslov paragraf

Tablica 3. Često korištene HTML oznake

Ovakva struktura podržava ugnježđivanje više oznaka unutar jedne i omogućava nam hijerarhijski prikaz elemenata i tretiranje istih kao objekata. Prednost hijerarhijskog prikaza je što se omogućuje pretraživanje elemenata unutar određene oznake ili pristup podređenim

(child) i nadređenim (parent) elementima. Ovo prezentacijsko sučelje se naziva Document Object Model (DOM)[20].



Slika 5. Primjer DOM strukture

3.2. CSS

CSS je skraćenica za Cascading Style Sheet, to je jezik za definiranje stilova i koristi se za opis izgleda HTML elemenata. CSS kod predstavlja niz pravila koja se sastoje od jednog ili više selektora i deklaracijskog bloka. Uz pomoć njega možemo mijenjati boju, margine, poravnanja elemenata, možemo mijenjati veličinu, izgled i font slova, odrediti fiksne pozicije elemenata i mnoge druge stvari. Sami CSS kod je moguće zapisivati unutar HTML datoteke i u vanjskoj datoteci koju je potrebno referencirati unutar HTML datoteke. Preporučuje se drugi pristup kako napisani kod ne bi bio dostupan korisnicima koji pretražuju internet, te kako bi sami HTML i CSS kodovi bili puno pregledniji[21].

Moguće je pisati vlastiti kod i tako stvoriti vlastiti izgled aplikacije i osobni dizajn, no također je moguće koristiti online frameworkove za implementaciju dizajna koji liči na nativne aplikacije. Primjeri tih frameworkova su Ionic, framework 7, Onsen.io i jQuery

[20] David Sawyer McFarland. JavaScript and jQuery: The Missing Manual, Second edition. California : O'Riley Media Inc. 2012. str. 127-128
[21] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 413-415

mobile. Svaki od tih frameworkova dolazi s nekoliko praznih projekata koji pokazuju osnovni dizajn i navigaciju, tako se također razvojnim programerima daje veća sloboda kod unosa samog sadržaja. Neki od tih frameworkova su također dostupni unutar visual studia (potrebno ih je preuzeti s interneta.) Iako se sam CSS koristi u kombinaciji sa HTMLom i XHTMLom, ovaj jezik je moguće primijeniti na bilo koju XML datoteku, što uključuje i čisti XML i XUL (XML User Interface Language.)

Osim što se koristi za definiciju izgleda, kroz razdvajanje sadržaja i njegovog formata moguće je prikazati istu stranicu uz korištenje raznih metoda za iscrtavanje , kao na primjer: na zaslonu računala, u tisku, uz pomoć uređaja koji su zasnovani na Brailleovom pismu, te uz glasovne naredbe (uz pomoć preglednika koji se temelji na glasovnim naredbama ili uz pomoć čitača zaslona.)

3.3. JavaScript

JavaScript je klijentski skriptni programski jezik. To označava da ovaj jezik izvršava kod za vrijeme korištenja web stranice, dok se kod statičnih programskih jezika ti dijelovi izvršavaju pri kompilaciji programa[22]. Svojstvo netipičnosti označava da nije potrebno naglasiti kojeg su tipa određene varijable (cijeli broj/tekst, privatne/javne) i da korak inicijalizacije nije potreban već je varijablu moguće inicijalizirati pri samoj deklaraciji. Činjenica da je ovaj jezik interpretativni znači da se naredbe zapisane u JavaScript programskoj datoteci izvršavaju izravno, odnosno bez potrebe za kompajliranjem programa u naredbe strojnog jezika. U tom slučaju kod izvršava Interpreter tako da prevodi svaki segment u slijed jedne ili više subrutina koje su već kompajlirane u strojni kod.

Činjenica da je ovaj jezik skriptni znači da se procesi koje ovaj jezik pokušava automatizirati pohranjuju u manje datoteke gdje svaka od tih datoteka ima vlastitu funkciju, odnosno taj proces je moguće zamijeniti ljudskom interakcijom, ali je u ovom slučaju jednostavnije imati automatizirani sustav (na primjer: ispunjavanje formulara u kojemu postoje već prije dopuštena polja i očekuje se korisnikov unos, ovisno o korisnikovom unosu, dogodit će se određena akcija.)

Prema strukturi, JavaScript je baziran na prototipima i podržava First-Class funkcije**.** First-Class funkcija označava da će programski jezik davati prednost funkcijama i dopuštat će im prosljeđivanje funkcija u obliku argumenata drugim funkcijama i podržava vraćanje rezultata

[22] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 395

iz drugih funkcija koje je moguće pohraniti u varijable ili druge strukture podataka. To ga čini programskim jezikom s više paradigmi zato što podržava Objektno orijentirani, imperativni i funkcionalni stil programiranja. Sam jezik posjeduje API za rad s tekstom, listama, datumima i regularnim izrazima, ali ne podržava Input/Output pristup u smislu umreženja, pohrane podataka i pohrane slika, te se bazira na poslužiteljskom okruženju u koje je taj kod ugniježđen. JavaScript također nalazi široku primjenu u okružjima koja nisu bazirana na mreži, neki od tih primjera su PDF datoteke, desktop widgeti i Site-specific browseri. Desktop widgeti označavaju interaktivne virtualne alate koji su namijenjeni isključivo za jednu funkciju kao na primjer: prikaz novosti iz određenog izvora, prikaz trenutnog vremena, kalkulator, sat i ostali, dok pojam Site - specific browsera opisuje softwareski tip aplikacije koji ima pristup predodređenim stranicama iz jednog izvora (web stranice) u privatnoj mreži ili na internetu. Također, razvoj JavaScript virtualnih mašina je pokrenuo šire korištenje i njegovu popularnost pri izradi poslužiteljskih aplikacija.

3.4. PHP

PHP (hypertext preprocesor) je skriptni jezik opće namjene i koristi se za programiranje pozadinskih servisa na poslužitelju. Prednost ovog jezika je što se može jednostavno koristiti s ostatkom web tehnologija. U početku, PHP je podržavao funkcijsku i proceduralnu paradigmu programiranja[23].

Klijent komunicira s poslužiteljem putem Ajax poziva i u mogućnosti je slati podatke i zahtjeve za određenim resursima koji su pohranjeni na samom poslužitelju. Klijent zahtjeve šalje na URL određene PHP skripte koja je pohranjene na poslužitelju.Poslužiteljsku arhitekturu je moguće organizirati tako da jedna PHP skripta ovisno o zahtjevu klijenta pokreće neku akciju. Drugo rješenje je korištenje većeg broja pristupnih točaka pri čemu je svaka PHP skripta zaslužna za izvršavanje unaprijed definiranog zadatka.

Od verzije 2004. godine (PHP5) ovaj programski jezik podržava i objektno orijentiranu paradigmu programiranja što omogućuje pisanje vlastitih klasa i instanciranje objekata unutar skripte[24]. Radi svoje velike raširenosti, na internetu možemo pronaći frameworke koji razvojnim programerima olakšavaju izradu pozadinske logike, neki od poznatijih PHP frameworkova su: Laravel, CodeIgniter, Symfony i CakePHP. Spomenuti frameworkovi

[23] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 35-36

[24] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 106.

omogućuju jednostavnije i brže izvođenje primarnih zadataka na poslužitelju. Osim frameworka, na internetu je moguće pronaći i unaprijed napisane skripte koje sadrže klase za rješavanje određenog problema (manipulacija slika, prijenos datoteka, autentikacija...)

3.5. jQuery

JQuery je knjižnica koja olakšava korištenje JavaScripta u programiranju korisničkog sučelja. Podržan je od strane većih kompanija kao što su Microsoft i Google. Obje kompanije također drže najnovije verzije jQuery datoteka na vlastitim poslužiteljima i omogućuju referenciranje istih unutar vlastitih web stranica ili mobilnih aplikacija, nedostatak tog pristupa je što se te datoteke neće učitati ako uređaj nije spojen na internet. JQuery je koristan radi vlastite jednostavne sintakse koja omogućava jednostavnu manipulaciju DOM objekata i lančano pozivanje funkcija. JQuery podržava Ajax pozive što omogućuje izradu sustava koji omogućava spajanje na poslužitelj i manipulaciju zapisa unutar baze podataka ili asinkrono pokretanje određenih radnji na web stranici. Glavna prednost jQuerya je što se oslanja na na selektore. Pomoću njih moguće je označiti jedan ili više elemenata na temelju **klase, ida** ili vrsti HTML elementa. JQuery razvojnim programerima omogućuje[25]:

- jednostavnu HTML i DOM manipulaciju
- css manipulaciju
- event metode
- vizualne efekte i animacije
- korištenje Ajax poziva

HTML i DOM manipulacija omogućuju razvojnim programerima dinamično dodavanje, brisanje i uređivanje elemenata unutar jedne stranice. Funkcije koje se koriste mogu utjecati na jedan dohvaćeni element ili na veću skupinu elemenata. Sa JQueryem je također moguće postaviti određeni vremenski razmak između dodavanja ili micanja određene klase.

[25] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 500-501

Osnovne funkcije koje omogućuju tu manipulaciju HTML-a i njegovog sadržaja su sljedeće:

- **prepend()** - dodaje sadržaj na početak jednog ili više označenih elemenata
- **append()** - dodaje sadržaj na kraj jednog ili više označenih elemenata
- **before()** - dodaje sadržaj prije jednog ili više označenih elemenata
- **after()** - dodaje sadržaj nakon jednog ili više označenih elemenata
- **remove()** - briše sve elemente i sadržaj koji se nalazi unutar jednog ili više označenih elemenata
- **replaceAll()** - zamjenjuje jedan ili više označenih elemenata sa željenim elementom

JQuery omogućava dodavanje, brisanje ili mijenjanje CSS klasa putem koda. Uz pomoću jQuery selektora, moguće je označiti jedan ili više elemenata i mijenjati njegove css klase ili primjenjivati određena css svojstva. Postoje četiri funkcije koje utječu na css kod[26]:

1) **.addClass("ime klase")** - dodaje klasu na označeni element, ime klase treba biti tip podataka string.
2) **.removeClass("ime klase")** - briše klasu iz označenog elementa, ime klase treba biti tip podataka string.
3) **.toggleClass("ime klase")** - ako označeni element ne posjeduje zadanu klasu, ova funkcija će dodati zadanu klasu. Ako element posjeduje klasu, ta klasa će biti izbrisana , ime klase treba biti tip podataka string.
4) **.css("ime svojstva", vrijednost")** - dodaje određeno css svojstvo i vrijednost na označeni element.
5) **.css({ "ime svojstva" : "vrijednost", "ime svojstva" : "vrijednost" })** - dodaje više css svojstava odjednom, sve vrijednosti trebaju biti tip podataka string.

Rukovatelji događanja omogućuju pokretanje određenih, unaprijed napisanih funkcija pri određenom događaju. Najvažnija metoda je .ready() koju je moguće vezati uz klasu "document" putem koda. U tom slučaju, ta funkcija se pokreće tek nakon što se cijela stranica učitala zajedno s svim stilovima koji opisuju izgled web stranice, skriptama koje poboljšavaju funkcionalnost stranice i datotekama koje se prikazuju na njoj. Rukovatelji događanja se mogu vezati na specifične elemente (putem ida,) elemente iste vrste (putem HTML oznaka) i elemente iste funkcije (putem klasa.)

[26] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 535-536

Često korištene metode koje jQuery podržava su[27].

Događaji na mišu	Događaji na tipkovnici	Događaji obrazaca	Događaji dokumenta/prozora
.click() - poziva se kada korisnik klikne mišem na zadani element	**.keypress()** - poziva se pri pritisku gumba na tipkovnici (poziva se dok je gumb pritisnut)	**.submit()** - poziva se kada korisnik pokuša poslati sadržaj obrasca na poslužitelj	**.load()** - poziva se nakon što su zadani element i svi njegovi podelementi učitani
.dblclick() - poziva se kada korisnik dva puta klikne mišem na zadani element	**.keydown()** - poziva se nakon što je korisnik pritisnuo tipku na tipkovnici i prije nego što je otpusti	**.change()** - poziva se nakon što se sadržaj unosnog polja promjeni	**.resize()** - poziva se nakon što je zadani element promijenio veličinu
.mouseenter() - poziva se kada pokazivač miša uđe u okvir zadanog elementa	**.keyup()** - poziva se nakon što je korisnik pritisnuo i otpustio tipku	**.focus()** - poziva se kada zadani element uđe u fokus (korisnik je kliknuo na element)	**.scroll()** - poziva se dok korisnik koristi klizač u označenom prozoru
.mouseleave() - poziva se kada pokazivač miša izađe iz okvira zadanog elementa		**.blur()** - poziva se kada zadani element izađe it fokusa (korisnik je kliknuo na drugi element)	**.unload()** - nakon što korisnik napusti ili ponovno učita trenutnu stranicu

Tablica 4. Jednostavni događaji koje je moguće detektirati putem jQuerya

JQuery radi svoje jednostavne sintakse također omogućuje lančano povezivanje funkcija, odnosno, u jednoj liniji koda je moguće pozvati više funkcija nad elementom koje će se pozivati uzastopno. Ovaj način pisanja koda također omogućuje razvojnim programerima korištenje Ajaxa.

AJAX je skraćenica za Asinkroni JavaScript i XML. On predstavlja set tehnika kojima je moguće preuzeti podatke s poslužitelja na zahtjev klijenta. Pojam asinkronosti u ovom slučaju označava da se svi AJAX pozivi odvijaju u pozadini na klijentu i nisu sinkronizirani s ostatkom JavaScript koda. AJAX također omogućuje prikaz dinamičnog sadržaja, odnosno resursa koji seu zapisani u bazi podataka. Pri korištenju AJAX poziva, trenutna stranica neće

[27] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 508-521

biti učitana ispočetka, već će dohvaćeni sadržaj biti umetnut unutar željenog elementa na stranici. Samim time što se stranica ne učitava ispočetka, poslužitelj i klijent međusobno šalju manju količinu podataka (JavaScript, css, html i ostale datoteke se učitavaju samo pri prvom učitavanju stranice[28].)

Problem koji AJAX predstavlja je činjenica da se svi pozivi odvijaju neusklađeno s ostatkom JavaScript koda. To može dovesti do grešaka pri prikazu podataka korisniku. Najčešći primjeri tih grešaka se događaju pri slabijoj internetskoj vezi korisnika. Za primjer se može uzeti polje za pretraživanje koje treba korisniku ponuditi česte pojmove za pretragu na temelju njegovog unosa. Pri slabijoj mreži korisnik neće biti svjestan to može dovesti do situacije da se zatraženi podatci prekasno prikažu na korisnikovom uređaju i potencijalno dovedu do greške.

Unutar svakog AJAX poziva je potrebno definirati koji će se dio koda pokrenuti kada je poziv na poslužitelj uspješan ili kada se dogodila greška na poslužitelju. Ako AJAX poziv kao odgovor dobije informaciju o grešci, kod koji se pokreće je definiran unutar " **error:"** dijela, "**error:"** dio se sastoji od sljedeća tri dijela:

- xhr - opis neuspješnog poziva na poslužitelj
- status - HTTP statusni kod greške (500 označava grešku unutar poslužitelja)
- error - tekstualni opis greške koja je dovela do neuspješnog izvršavanja poziva ako je za grešku zaslužan sam poslužitelj, ovo polje će glasiti: "Internal server error"

3.6. Bootstrap

Bootstrap je besplatan framework otvorene licence napisan u CSS jeziku za označavanje. On razvojnim programerima olakšava i ubrzava proces dizajniranja web stranica koje su optimizirane za različite veličine računala, mobilnih uređaja i tableta[29]. Bootstrap framework se može koristiti samostalno ili u kombinaciji s vlastitom CSS datotekom koja služi kao dodatan opis izgleda stranice. Ovaj framework može poslužiti za izradu hibridnih mobilnih aplikacija koje su bazirane na web tehnologijama radi vlastite optimiziranosti za mobilne uređaje. Ovaj framework u osnovi nudi unaprijed predodređene klase za osnovne HTML elemente: naslove, podnaslove, paragrafe, tablice, slike, obrasce i ostale.

[28] Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015. str. 395
[29] Jake Spurlock. Bootstrap - Responsive Web Development. 1st edition. California : O'Riley Media Inc. 2013. str. 1

Glavne značajke Bootstrapa su:

- jednostavnost korištenja - svatko sa znanjem HTMLa i CSSa može koristiti Bootstrap
- responzivnost - izgled svih elemenata je prilagođen za različite rezolucije računala, mobitela i tableta
- kompatibilnost među web preglednicima: Chrome, Firefox, Internet Explorer, Edge, Safari i Opera
- osnovne stilove za većinu HTML elemenata
- velik broj unaprijed stiliziranih komponenata

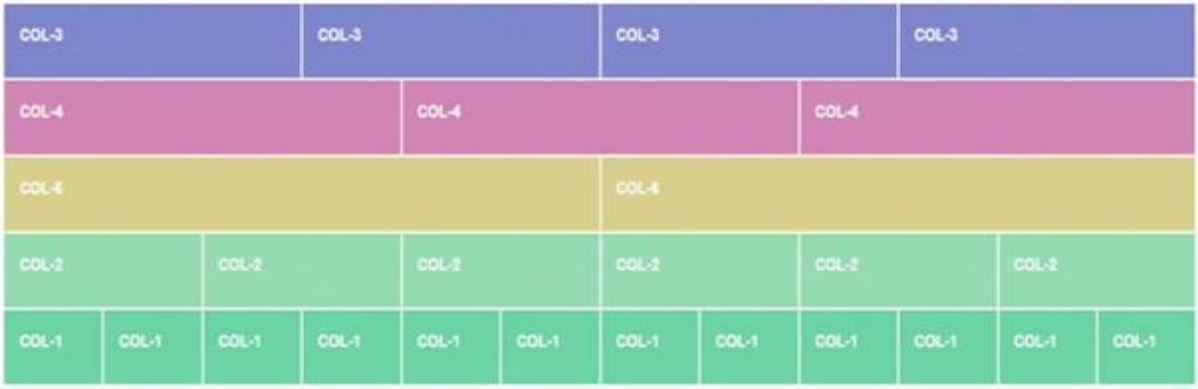
Bootstrap nudi unaprijed određeni osnovni izgled i funkcionalnost za često korištene elemente na web stranicama. Ti elementi uključuju: gumbe, padajuće izbornike, kartice, navigacijske trake, zaglavlja, podnožja, i unosna polja za različite formate (datum, datum i vrijeme, brojevi, kratki tekst, textbox i sl.) Svi navedeni elementi su konzistentni među većim brojem računalnih i mobilnih web preglednika (chrome, firefox, safari, opera.) Spomenuti elementi su također prilagođeni različitim rezolucijama računala i mobilnih uređaja.

Glavna prednost Bootstrapa je što podržava sustav mreže (grid system) temeljen na 12 jednako širokih polja[30]. Ovaj sustav olakšava izradu dinamičnih tablica, pozicioniranje elemenata kartica ili bilo kojih drugih elemenata kojima se želi definirati širina i pozicija. Pozicioniranje i optimizacija se odvijaju uz korištenje širina definiranih u pikselima.

[30] Jake Spurlock. Bootstrap - Responsive Web Development. 1st edition. California : O'Riley Media Inc. 2013. str. 3

Ovisno o širini uređaja, ta mreža elemenata će se razlomiti iz stupaca u retke i prilagoditi manjim ili većim uređajima. sufiksi koji se koriste u kombinaciji sa .col klasom su:

- xn - koristi se za prelamanje na mobilnim uređajima (0 - 767 piksela)
- sm - koristi se za prelamanje na tabletima (768 - 991 piksela)
- md - koristi se za prelamanje na prosječnoj rezoluciju računala (992 - 1199 piksela)
- lg - koristi se za veliku rezoluciju računala (1200 piksela i više)



Slika6 Bootstrapov sustav mreže

3.7. JSON web tokeni (JWT)

JSON web tokeni su bazirani prema otvorenom standardu (RFC 7519) za kreiranje pristupnih tokena. Osnove web tokena uključuju tvrdnje, odnosno, ono što je zapisano u tokenu predstavlja korisnika[31]. Tokeni često sadrže id korisnika, ime, prezime, adresu elektroničke pošte i razinu privilegija (administrator ili korisnik.) Klijent ga prilaže svakom zahtjevu koji je upućen na poslužitelj, koji će prvo provjeriti valjanost istog, i iščitati informacije koje su zapisane u njemu. Na temelju korisnikove razine privilegija, poslužitelj može odlučiti hoće li izvršiti određeni upit ili vratiti poruku o grešci i statusni kod greške prema pošiljatelju zahtjeva.

JSON web token se sastoji od tri dijela, svaki dio je JSON zapise šifriran u base64 formatu i odvojen točkom. Prvi dio označava zaglavlje u kojemu su pohranjene meta informacije o tokenu, drugi dio je sadržaj tokena koji uključuje informacije o izdavaču, vremenu izdavanja, vremenu isteka, namijenjenoj publici i slično. Treći dio je digitalni potpis koji je generiran od strane poslužitelja na temelju zaglavlja, sadržaja i tajnog ključa poslužitelja. JSON web token također sadrži unaprijed definirana, standardna polja za zaglavlje i sadržaj[32].

[31]Definicija json web tokena URL: https://tools.ietf.org/html/rfc7519.html#section-1
[32]Proces stvaranja Json Web Tokena: https://tools.ietf.org/html/rfc7519.html#section-7

Unaprijed definirana polja unutar sadržaja tokena[33]:

Kod	Naziv	Opis
iss	Izdavač	Naziv izdavača tokena (web adresa ili niz znakova.)
sub	Predmet	Predmet ili tema na koju se token referira.
aud	Publika	Publika kojoj je token namijenjen (može sadržavati više vrijednosti, svaka od kojih je web adresa ili niz znakova)
exp	Vrijeme isteka	Vrijeme nakon kojeg token prestaje biti važeći, svaki zahtjev s nevažećim tokenom će biti odbijen Ovo polje je obavezno
nbf	Ne prije	Vrijeme do kojeg će se token smatrati nevažećim.
iat	Izdano u	Vrijeme kada je token izdan.
jti	JWT ID	Unikatni identifikacijski niz tokena, ovaj niz treba biti unikatan među SVIM izdavačima.

Tablica 5. Unaprijed definirana polja unutar sadržaja tokena

Unaprijed definirana polja unutar zaglavlja tokena[34]:

Kod	Naziv	Opis
typ	Vrsta tokena	Označava vrstu tokena, vrijednost bi uvijek trebala biti: JWT
cty	Vrsta sadržaja	Označava vrstu sadržaja, koristi se ako je token dodatno šifriran ili ako se vrši više slojeva potpisivanja.
alg	Algoritam za autentikaciju tokena	Označava algoritam koji se koristi za digitalno potpisivanje tokena. Sam algoritam je najčešće hash funkcija. Verifikacija tokena na temelju ovog polja nije sigurna. Preporučuje se usporedba polja s korištenim algoritmom na poslužitelju. Algoritam za šifriranje je unaprijed definiran na poslužitelju i u slučaju krivo navedenog algoritma poslužitelj može pretpostaviti da korisnik pokušava falsificirati token. Važno je napomenuti da neki od preporučenih algoritama nisu sigurni.

Tablica 6. Unaprijed definirana polja unutar zaglavlja tokena

U sljedećem dijelu će biti prikazan primjer tokena koji predstavlja korisnika. Svaki dio tokena će biti zapisan u JSON i base64 formatu.

[33]Unaprijed definirana polja unutar sadržaja tokena URL:https://tools.ietf.org/html/rfc7519.html#section-4.1
[34]Unaprijed definirana polja unutar zaglavlja tokena URL: https://tools.ietf.org/html/rfc7519.html#section-5

Json zapis	Base64 zapis

Zaglavlje:
{
"typ": "JWT",
"cty": "JWT",
"alg": "HS256"
}
	Zaglavlje:
	eyJ0eXAiOiJKV1QiLCJjdHkiOiJKV1QiLCJhbGciOiJIUzI1NiJ9

Sadržaj:
{
"iss": "www.mojaStranica.com",
"sub": "www.mojaStranica.com",
"aud": "www.mojaStranica.com",
"iat": 1546300800,
"nbf": 1546300801,
"exp": 1546302600,
"jti": 4564261,
"ime": "Ivo",
"prezime": "Ivić",
"mail": "iivic@mail.hr",
"ulogiranKao": "administrator"
}
	Sadržaj:
	eyJpc3MiOiJ3d3cubW9qYVN0cmFuaWNhLmNvbSIsInN1YiI6Ind3dy5tb2phU3RyYW5pY2EuY29tIiwiYXVkIjoid3d3Lm1vamFTdHJhbmljYS5jb20iLCJpYXQiOjE1NDYzMDA4MDAsIm5iZiI6MTU0NjMwMDgwMSwiZXhwIjoxNTQ2MzAyNjAwLCJqdGkiOjQ1NjQyNjEsImltZSI6Ikl2byIsInByZXppbWUiOiJJdmnEhyIsIm1haWwiOiJpaXZpY0BtYWlsLmhyIiwidWxvZ2lyYW5LYW8iOiJhZG1pbmlzdHJhdG9yIn0

Digitalni potpis Ključ: 0e72ff88b1d486f164bd44331b96bfb1b9149c254e2b334b98d7fb64204b06b9 Digitalni potpis se generira na temelju zaglavlja i sadržaja u base64 formatu i tajnog ključa koji je najčešće pohranjen na poslužitelju i sakriven od korisnika. Za stvaranje tajnog potpisa se koristi SHA256 algoritam. SHA256 (ključ, zaglavlje.sadržaj)	**Digitalni potpis:** N3c21WsXKAgKmhu_M4VzAR3EeSAME4zjwHOSf4XyH-c

Tablica 7. Primjer JSON Web Tokena

Za stvaranje digitalnog potpisa u ovom slučaju se koristi SHA256 algoritam za raspršivanje. Ovaj algoritam nije fiksan i za stvaranje potpisa je moguće izabrati jednu od ponuđenih funkcija za raspršivanje (SHA256, SHA384, SHA512) ili jedan od ponuđenih kriptografskih algoritama (RS256, RS384, RS512, ES256, ES384, ES512, PS256, PS384, PS512)[35].

Funkcije raspršivanja kao rezultat uvijek vraćaju fiksni broj bitova što može dovesti do kolizija među konačnim vrijednostima. Važno je spomenuti kako su ove funkcije nepovratne, te je iz njihovog rezultata nemoguće iščitati unosne informacije, upravo radi te činjenice, informacije koje su pohranjene u rezultatu se ne mogu iščitati i taj rezulat reprezentira digitalni potpis. Provjera potpisa se vrši tako da poslužitelj na svakom tokenu koristi funkciju raspršivanja, tajni ključ, zaglavlje i sadržaj tokena, te nakon toga uspoređuje dobiveni rezultat s potpisom unutar primljenog tokena. Ako se potpisi podudaraju, poslužitelj zaključuje da je token valjan.

Valjanost tokena je moguće provjeriti na web stranici: https://jwt.io/. U sučelju je također moguće odabrati željeni algoritam za raspršivanje/šifriranje, te je moguće priložiti tajni ključ (u slučaju da koristimo funkciju raspršivanja) ili set privatnih i javnih ključeva (u slučaju korištenja kriptografskog algoritma.) S prethodno spomenute stranice je moguće preuzeti unaprijed napisane klase i skripte namijenjene za implementaciju u većem broju programskih jezika. Neke od implementacija uključuju .NET framework, JavaScript, PHP, ruby i ostali. Za neke od jezika je moguće preuzeti više različitih datoteka. Svaka od datoteka ima naznačene komponente koje su uključene u skripti/klasi i međusobno su podržane.

[35] Podržani algoritmi za potpisivanje i šifriranje URL: https://tools.ietf.org/html/rfc7518#section-3

3.7.1. Base64 format šifriranja[36]

Base64 format predstavlja shemu za šifriranje teksta koje se bazira na odabiru 64 često korištenih simbola (slova, brojevi i znakovi.) Pri šifriranju, unosni tekst se pretvara u binarni zapis i dijeli u blokove veličine 24 bita. Svaki blok se dodatno dijeli u 4 grupe od 6 bita i svaka grupa će biti zakodirana u jedan znak prema dolje prikazanoj tablici. Ako se posljednji blok ima manje od 24 bita, koristit će se znak jednakosti (=) da upozori na prazne blokove pri dekodiranju. Razlog korištenja 64 znaka je taj što blokovi od 6 bitovaa mogu predstavljati 64 različite vrijednosti (2^6=64.)

Indeks	Znak	Indeks	Znak	Indeks	Znak	Indeks	Znak
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+ ili -
15	P	31	f	47	v	63	/ ili _
(Dodatno punjenje) =							

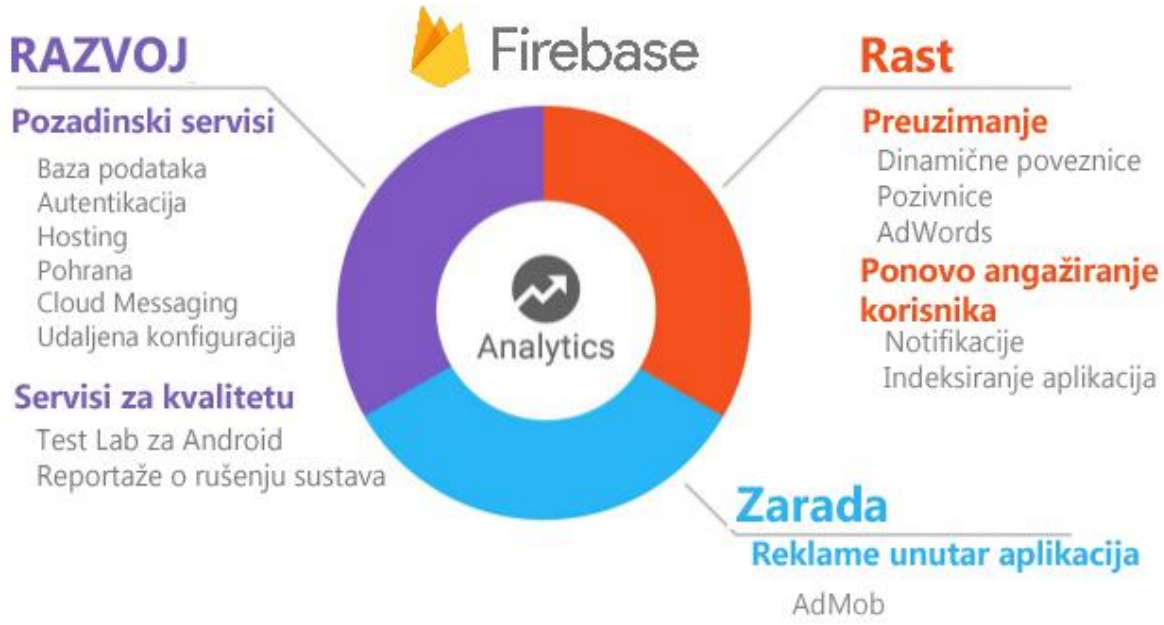
tablica8 indeksna tablica base64 formata šifriranja

[36]Base64 standard kodiranja URL: https://tools.ietf.org/html/rfc4648#section-4

Posljednja dva simbola unutar tablice variraju od implementacije do implementacije. Za slanje ovakvog formata putem interneta se preporučuje korištenje simbola minus (-) i donja crta (_) jer se smatraju sigurnima za slanje putem URLa[37]. Osim pri šifriranju teksta, ovim formatom je moguće prikazati slike, audio zapise i video zapise.

3.8. Firebase[38]

Firebase je Googleova platforma namijenjena za razvoj mobilnih i web aplikacija. Ona razvojnim programerima omogućuje pristup većem broju servisa koji olakšavaju razvoj istih. Firebase razvojnim programerima omogućuje stvaranje baze podataka, pohranjivanje datoteka, analizu prometa. On može služiti kao zasebni poslužitelj ili poslužitelj koji je zadužen za određenu komponentu web stranice ili mobilne aplikacije. Unutar ove platforme možemo pohraniti veći broj datoteka kao što su: slike, video zapisi, JSON, XML, HTML i CSS datoteke, PHP skripte i ostale. Platforma također posjeduje više međusobno povezanih servisa koji omogućuju statističku analizu web stranice, bilježenje grešaka, optimizaciju stranice za web tražilice, izdavanje pristupnih tokena i konačno, servis za slanje push notifikacija i sms poruka na uređaje korisnika.



Slika 7. Usluge Firebase platforme

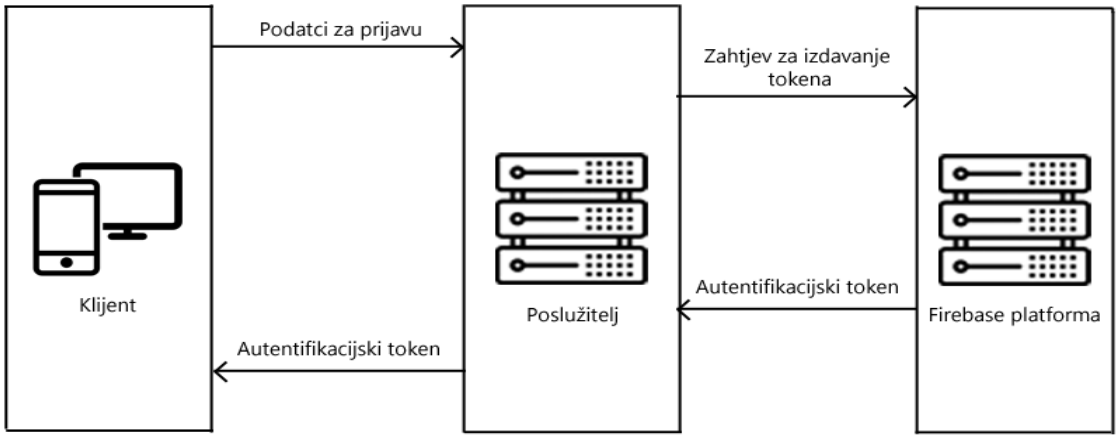
[37] Base64 kodiranje prilagođeno za slanje putem URLa URL: https://tools.ietf.org/html/rfc4648#section-5
[38] Chris Esplin. What is Firebase, the complete story abridged. URL:https://medium.com/Firebase-developers/what-is-Firebase-the-complete-story-abridged-bcc730c5f2c0

Neki od važnijih servisa vezanih uz Firebase platformu su:

- Korištenje baze podataka
- Autentikaciju korisnika
- Hosting stranice
- pohranu podataka
- Slanje push notifikacija korisnicima
- Slanje poruka unutar mobilne aplikacije
- Sustav za procjenu kvalitete i brzine aplikacija
- Obavještavanje o padu poslužitelja
- Implementaciju strojnog učenja unutar web stranice ili aplikacije (beta verzija)

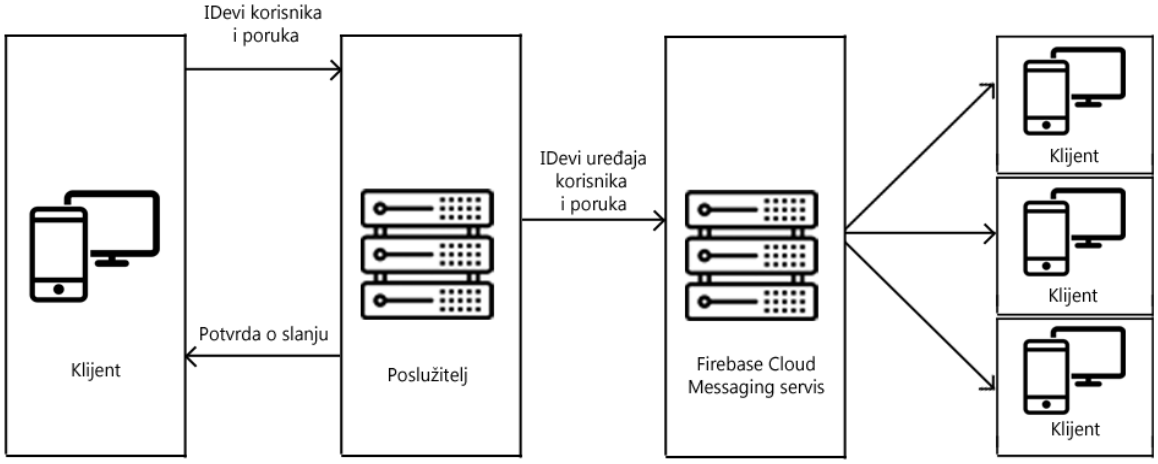
Firebase omogućuje razvojnim programerima izradu poslužitelja koji može poslužiti za jednu ili više svrha. Ova platforma dopušta kreiranje poslužitelja koji će se ponašati kao: poslužitelj za podatke, poslužitelj za autorizaciju ili poslužitelj za slanje push notifikacija i poruka unutar web aplikacija ili mobilnih notifikacija.

Prema standardu REST arhitekture koja podržava slojevite sustave, Firebase platformu je moguće koristiti kao zasebni server za autentikaciju koji će biti zadužen za generaciju OAuth tokena i prosljeđivanje istih. Osim tokena za autorizaciju, Firebase podržava generaciju tokena za osvježavanje. Oni su slični tokenima za autentifikaciju uz razliku da imaju puno duži period trajanja i služe za dohvaćanje autentikacijskog tokena. Njih je moguće primjenjivati unutar mobilnih aplikacija kako se korisnik ne bi trebao prijavljivati nakon isteka autentifikacijskog tokena. U ovom slučaju razvojni programeri mogu dizajnirati vlastiti poslužitelj i koristiti Firebase isključivo za autentikaciju korisnika.



Slika 8. Korištenje Firebasea kao autorizacijski poslužitelj

Firebase cloud messaging je Googleov servis namijenjen za slanje poruka namijenjenih krajnjim korisnicima. On nam dopušta slanje push notifikacija na računala i mobilne uređaje korisnika i slanje SMS poruka na mobilne uređaje korisnika. Push notifikacije su vrste poruka koji se šalju na uređaj korisnika neovisno o tome da li je korisnik prijavljen u aplikaciju. Korištenjem Firebase Cloud Messaging servisa je moguće slati poruke prema jednom korisniku, prema većem broju korisnika i unaprijed određenim grupama. Korisnicima je dopušteno pretplaćivati i odjavljivati se iz spomenutih grupa. Kao što smo prethodno spomenuli, Firebase se može tretirati kao zaseban poslužitelj koji je namijenjen isporuci push notifikacija



Slika 9. Proces slanja push notifikacija na uređaje korisnika

4. Apache Cordova

Apache Cordova (do 2011. godine poznata kao PhoneGap) svoj razvoj započinje 2009. godine u San Franciscu na iPhone development kampu gdje također dobiva People's choice nagradu. Originalni autor je bila kompanija Nitobi. Ovo sučelje je u samome početku htjelo omogućiti izradu mobilnih aplikacija razvojnim programerima uz korištenje web tehnologija **(**HTML, CSS i JavaScript.) Razvoj ovakvog sučelja je također omogućio Android razvojnim programerima da se više fokusiraju na unaprjeđivanje samog Androida, te su time smanjili potrebu za pojednostavljivanjem određenih dijelova izvornoga koda, dok je istovremeno izrada mobilnih aplikacija omogućena puno široj publici[39]. Također je važno napomenuti da Apache Cordova nije jedino sučelje koje koristi web tehnologije.

Slika 10. Tehnologije vezane uz Apache Cordova framework

Uspjehu i razvoju ovog frameworka je uvelike pomogla i tvrtka Apple.inc s time što je podržao PhoneGap framework i omogućio njihovo pokretanje na **iOS mobilnim uređajima** unatoč promjenama unutar developerske licence vezane uz verziju 4.0. U ovo vrijeme je izrada mobilnih aplikacija bila ograničena isključivo na iOS mobilne uređaje i aplikacije je bilo potrebno raditi isključivo na Appleovim računalima.

Veliki korak u razvoju se također dogodio 2. listopada 2011. godine kada je Adobe Systems otkupio Nitobi. Sukladno s time PhoneGap je postao dio **Apache software Foundationa**,

[39] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 7

neprofitne organizacije čiji je glavni cilj podržavanje softverskih rješenja koja se nalaze pod njihovim okriljem. Sama zadruga funkcionira kao decentralizirana skupina developera koji razvijaju softverska rješenja pod **open source licencom**. Sukladno s načinom rada Apache zadruge, gdje postoje tehnički eksperti koji vode sami projekt, pridonose njegovom razvoju i upućuju ostale volontere kako dalje pridonositi projektima, sam PhoneGap je integriran u takav način razvoja. Nakon tog spajanja je također došlo do promjene samog naziva PhoneGap u Apache callback, te kasnije u Apache Cordova[40].

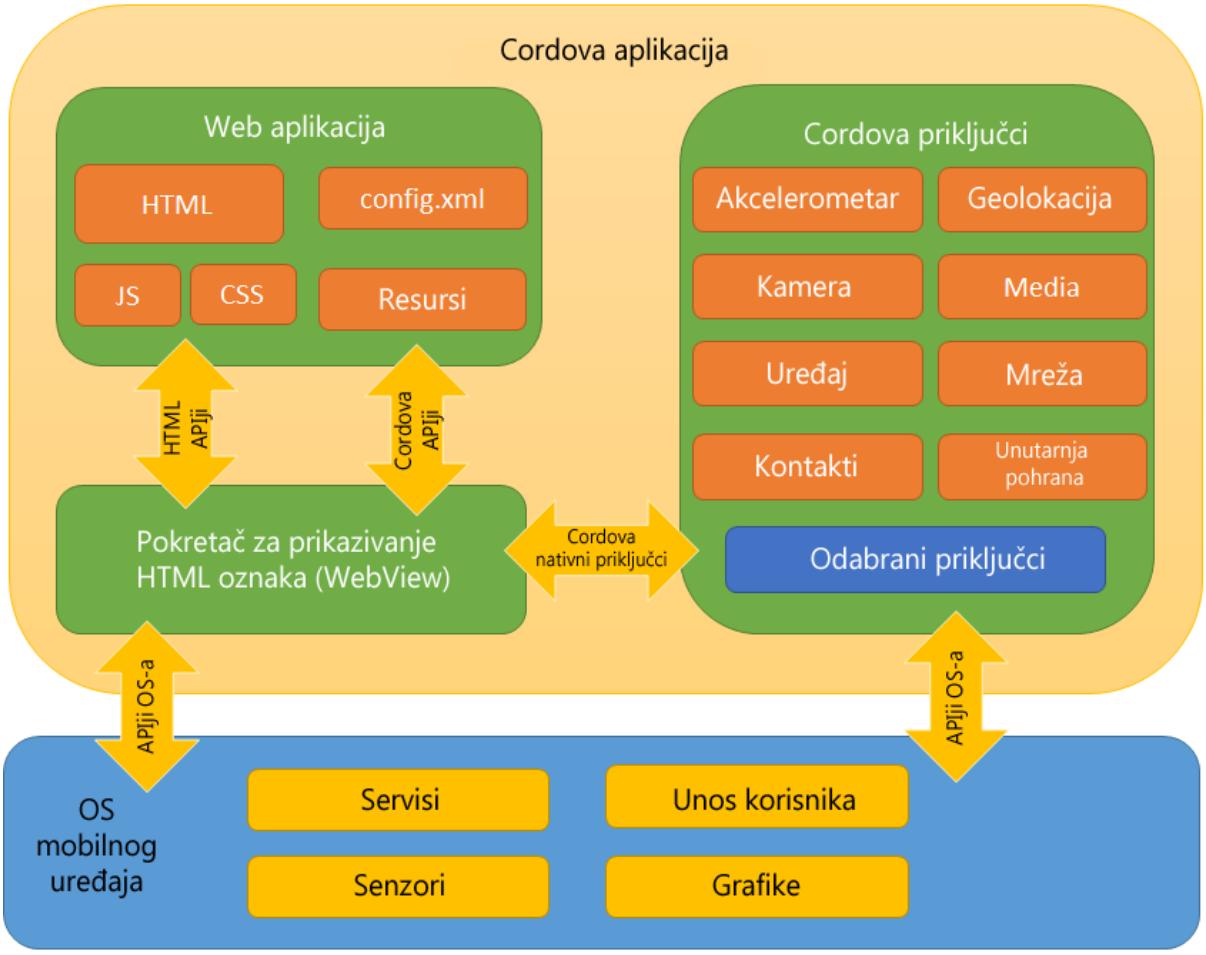
Phone aplikacije (njih je bilo moguće kreirati isključivo na Windows računalima.) U rujnu 2012. Godine je otkriven PhoneGap Build web servis koji dopušta programerima podizanje njihovog koda na njega i odabir ciljane platforme za koji je aplikacija rađena (windows Phone, Android), ciljanje verzije određene platforme i u konačnici kompajliranje same aplikacije unutar PhoneGapovog „cloud compilera“. Cloud compiler osim što kompajlira kod same aplikacije, također dopušta debuggiranje, odnosno, omogućava razvojnim programerima unošenje izmjena u aplikaciji koja je istovremeno otvorena u web pregledniku i provjeravati sadržaje podataka koji se izmjenjuju u mreži, što uvelike olakšava problem optimizacije u komunikaciji s poslužiteljem[41].

4.1. Karakteristike i mogućnosti[42]

Rezultat koji Cordova nudi je **hibridna** aplikacija. Taj pojam označava aplikacije koje za svoje funkcioniranje ne koriste isključivo nativne tehnologije mobilnih uređaja, već ih kombiniraju s Web tehnologijama koje je moguće koristiti na širem broju mobilnih uređaja (android, iOS, Windows phone.) Najčešći primjeri tih tehnologija su HTML CSS i JavaScript i njih ćemo dodatno objasniti u sljedećem dijelu seminarskog rada. Upravo radi tih široko korištenih tehnologija možemo zaključiti da će takva aplikacija biti dostupna širem broju ljudi. Glavni nedostatak ovakvog pristupa izradi aplikacija je što su hibridne aplikacije sporije od nativnih aplikacija upravo radi činjenice da ne koriste funkcionalnosti mobilnih uređaja izravno, već putem Cordova Libraryja prenose informacije komponentama kao što su kamera, pritisak po ekranu, klizanje i slično. To je jedan od razloga zašto su ovakve aplikacije jednostavnije i jeftinije te je vrijeme potrebno za njihovu izradu puno manje.

[40] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 8
[41] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 8
[42] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 1-7

U slučaju Cordova aplikacija, za prikaz grafičkih elemenata se koristi WebView klasa**.** To je komponenta mobilnih uređaja koja im omogućava brže i prilagođenije iscrtavanje elemenata web stranice na manjim zaslonima različitih dimenzija[43]. Upravo tako Cordova umjesto nativnog načina izrade grafičkog sučelja omogućava zaobilaženje korištenja većine nativnih tehnologija i razvojnim programerima omogućava razvoj mobilnih aplikacija koje su web stranice pohranjene na uređaju korisnika i komuniciraju s udaljenim poslužiteljem kako bi prikazane informacije bile ažurne. Iako se ovakve aplikacije dosta oslanjaju na korištenje web tehnologija, one nisu Web aplikacije čisto zato što ne koriste isključivo web tehnologije, već kombiniraju njihove mogućnosti s mogućnostima nekih od nativnih komponenata.



Slika 11. Povezanost Apache Cordova frameworka i mobilnog operativnog sustava

Unutar Cordova aplikacije je također moguće koristiti **Local Storage**, odnosno komponentu koja omogućava razvojnim programerima spremanje podataka o individualnome korisniku na

[43] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 3-4

samoj memoriji uređaja[44]. Tim postupkom možemo simulirati je li korisnik ulogiran u aplikaciju, te ako je, korisnika možemo preusmjeriti na zadnju stranicu aplikacije koja je bila uključena, a ako nije mu samo predočimo obrazac za registraciju ili prijavu.

Također, kako bi demonstrirali raširenost ovog Libraryja, spomenuti ćemo da je njega moguće koristiti u sklopu Visual Studio sučelja**,** Microsoftovog sučelja koje je namijenjeno za izradu desktop aplikacija, web projekata(uz pomoć ASP.Net frameworka) i mobilnih aplikacija uz pomoć većeg broja programskih jezika(c,c+, c++,c#, python, visual basic, JavaScript). Osim Visual Studio sučelja, sami Cordova library je moguće korisititi u kombinaciji s Monaca, appMobi, Viziapps, Convertigo Studiom i IBM Worklightom.

4.2. Tehnologije vezane uz Apache Cordova framework

U sljedećem dijelu rada će biti nabrojane i opisane karakteristike najčešće korištenih priključaka koje je moguće koristiti u kombinaciji s Apache Cordova libraryem. Osim priključaka ćemo objasniti kako je moguće ostvariti pohranu podataka na mobilnim uređajima (ovisno o količini podataka) i način na koji se odvija komunikacije s Web poslužiteljem i format u kojemu se ta komunikacija može odvijati.

4.2.1. Cordova i HTML

Apache Cordova za kostur aplikacije koristi HTML, CSS i JavaScript. Kao što smo ranije spomenuli da se radi o hibridnoj aplikaciji, te tehnologije koristimo na isti način kao i pri dizajniranju web stranica. HTML primarno koristimo za pozicioniranje određenih elemenata u aplikaciji. Čista HTML stranica, bez CSS-a prikazuje sve elemente istim redoslijedom kao i u dokumentu krećući od vrha prema dnu. U slučaju kada koristimo CSS kod koji upućuje da bi dva ili više elementa trebala stajati jedan pored drugog, slijed iscrtavanja elemenata je prvo slijeva na desno, pa od vrha prema dnu. Sam HTML je vrlo jednostavan i potrebna je jako mala količina vremena da bi se u potpunosti savladao. Njegov cilj je osim prikaza elemenata, pobrinuti se da se isti sadržaj prikazuje jednako na svim korištenim web preglednicima. Osnovni elementi HTML koda su tagovi, odnosno oznake koje se zapisuju u izlomljene zagrade(<>) i njihov popis i primjere korištenja je moguće pronaći na: www.w3c.org**.** W3C, odnosno World Wide Web Consortium je organizacija koja se bavi standardizacijom tehnologija koje se koriste na webu. Osnovana je u listopadu 1994. Godine u suradnji između MIT-a i CERN-a. Sam inicijator se ujedno i izumitelj Weba, Tim Berners-Lee Također je

[44] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 3247

važno napomenuti da u <meta> oznakama možemo referencirati i vanjske lbraryje kao što su jQuery library, angular JavaScript, jquery mobile i slični.

Trenutna verzija HTML jezika za označavanje je HTML5**,** koji je nastao 1999. godine u suradnji s World wide web Consortiumom i Web Hypertext Application Technology Working Group **(**WHATWG**)**. U 2008. je izdan radni dokument od strane W3C-a. U 2012. godini je došlo do podjele između W3C-a i WHATWG-a, te je uspostavljeno da će se W3C baviti razvojem HTML-a tako da ga pokušaju postaviti kao jedinstveni, konačni standard u izradi web stranica, dok je cilj WHATWG grupe baviti se HTML-om kao živim standardom. Živi standard predstavlja koncept u kojemu proizvod u razvoju nije nikada potpun, već se konstantno nadograđuje kako bi koristio što veći broj dostupnih tehnologija Kako bi sam HTML postao definitivni standard, u ovome slučaju je bilo potrebno proizvesti dvije različite, potpuno gotove implementacije koje mogu međusobno funkcionirati. Tako je u konačnici u 2014. godini HTML5 izdan kao stabilna verzija, čime je proces specifikacije završen. Također je važno napomenuti da ova verzija HTML-a podržava pristup određenim dijelovima nativnog mobilnog softwarea kao što su: GPS, kamera, brzinomjer i sl.

4.2.2. Podržani priključci

Priključci (ili ekstenzije) označavaju softwareske komponente koje dodaju specifične mogućnosti u postojeći računalni program, programski jezik ili prezentacijski jezik. Oni omogućuju veću razinu prilagodbi i u slučaju dizajn aplikacija smanjuje vrijeme koje je potrebno za izradu određenih funkcija ili izgleda[45]. Većina tih priključaka u sklopu Apache Cordove se zasniva na JavaScriptu koji ostvaruje komunikaciju između nativnog sloja i HTMLa.

Opća preporuka u korištenju priključaka je da se razvojni programeri ograniče na korištenje priključaka koji mijenjaju funkcionalnost ili izgled aplikacije na strani korisnika(odnosno priključci koji će biti pohranjeni ili referencirani na korisnikovom uređaju unutar HTML datoteka ili korištenje prethodno napravljenih tema ili sučelja za baratanje dizajnom.)

Neki od najčešće korištenih priključaka temeljeni su na libraryima JavaScript programskog jezika. Ti libraryji su: jQuery, AngularJS i jQuery mobile. Oni omogućavaju jednostavnije i elegantnije pisanje koda i svi dolaze s ugrađenom logikom Event listenera. Event listener

[45] John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012. str. 14

(slušač događanja) je subrutina koja se poziva asinkrono(ovisno o korisnikovom unosu) i sami poziv sinkrono utječe na tijek izvođenja programa. Naziv listener potječe iz zamisli da program iščekuje korisnikov unos i u trenutku unosa ili pritiska na određeni gumb pokreće određenu funkciju koja mijenja stanje aplikacije. Također, oni podržavaju strukturu programa u kojoj je moguće razdvojiti dio koda koji će se pokretati ako je određeni upit u bazi bio uspješan ili neuspješan(to razvojnim programerima olakšava konstrukciju logike koja će što jednostavnije naznačiti korisniku zašto određena radnja nije uspjela i koje je korake potrebno poduzeti da se ta ista radnja izvrši.)

Preuzimanje i pretraživanje priključaka je omogućeno unutar Visual studia ili preko Github sustava za verziranje programa. Ako se preuzimanje odvija preko Github sustava, preuzete priključke je moguće unjeti u projekt putem npm komandi ili referenciranjem unutar HTML datoteka.

Također, kao što smo prethodno spomenuli u sklopu CSS-a, unutar Apache Cordova Libraryja je moguće koristiti druga sučelja koja služe za mijenjanje samog dizajna aplikacije. Važno je napomenuti da se ti predlošci najčešće baziraju na Flat Dizajnu ili na Material dizajnu i da je svaki od tih predložaka automatski primijenjen ciljanom uređaju.(tako je moguće reproducirati nativni dizajn Android, iOS i Windows phone aplikacija, dok u realnosti developer izrađuje jednu aplikaciju.) Iako taj proces uvelike ubrzava proces izrade. Također treba istaknuti kako Apple.inc može odbiti gotove aplikacije koje želimo distribuirati putem appStorea ako njihov dizajn i vrijeme responsa ne zadovoljavaju Appleove standarde.

4.2.3. Načini pohrane podataka[46]

Unutar Cordova aplikacija je moguće spremati podatke na četiri različita načina. Ti načini su:

- LocalStorage
- WebSQL
- Indeksirane baze podataka
- Online Database

LocalStorage je najjednostavniji, sinkroni način pohrane podataka koji funkcionira na principu vrijednosti i ključa i podržan je od strane **WebView** implementacija na svim podržanim mobilnim uređajima. Veliku prednost ovoj komponenti daje jednostavan API, no neki od nedostataka kao što je pohrana isključivo tekstualnih podataka, te se kompleksnije strukture podataka trebaju serijalizirati. Također je važno napomenuti da local storage ima

[46]Načini pohrane podataka u Apache Cordova aplikacijama:https://Cordova.apache.org/docs/en/latest/Cordova/storage/storage.html

maksimalnu granicu od 5Mb što upućuje da se ova komponenta neće koristiti za pohranu većih količina podataka, također je važno upamtiti da iOS uređaji mogu automatski pobrisati njegov sadržaj ako je potrebno osloboditi prostor na uređaju korisnika.

WebSql je komponenta koja se sastoji od APIja za pohranu podataka unutar strukturirane baze podataka i moguće joj je pristupiti korištenjem standardne SQL sintakse.(točnije **SQLite**). Glavni nedostatak ovog pristupa je što je ova tehnologija dostupna isključivo na sljedećim mobilnim uređajima:

1. Android
2. BlackBerry
3. iOS

Indeksirane baze podataka označavaju pristup koji pokušava spojiti prednosti lokalne pohrane i korištenja WebSQLa, dok im istovremeno zaobilazi slabe strane. Ona se najčešće koristi za pohranu JavaScript objekata koje je moguće indeksirati pod određenim ključem. Također nam pruža neke od prednosti SQL tablica bez da im ograničava strukturu ili potrebu za definiranjem unaprijed. Prednost ovakvog pristupa je što nam uz jednostavnu funkcionalnost LocalStoragea omogućava kreaciju većeg broja baza podataka. Glavni nedostatak ovog pristupa je što nije u potpunosti podržana na windows mobilnim uređajima i ti mobiteli ne mogu pozvati određene komponente.

Baze podataka na poslužitelju – ova pristup označava da će potrebni podatci biti pohranjeni na web poslužitelju u bazi podataka i da će mobilni uređaj ovisno o željenoj akciji slati upit na sam poslužitelj koji će isti zahtjev obraditi i vratiti željeni rezultat. Ovakav pristup nam omogućava pohranu veće količine podataka i koristi se u slučajevima gdje se sadržaj konstantno osvježava i filtrira od informacija koje su zastarjele i više se ne koriste. Ovaj pristup je najkompliciraniji za implementaciju pošto je potrebno pisati programsku logiku za klijentske uređaje i programsku logiku za sami poslužitelj, uz dodatni posao izrade baze podataka i potrebnih tablica.

5. InfoStudent

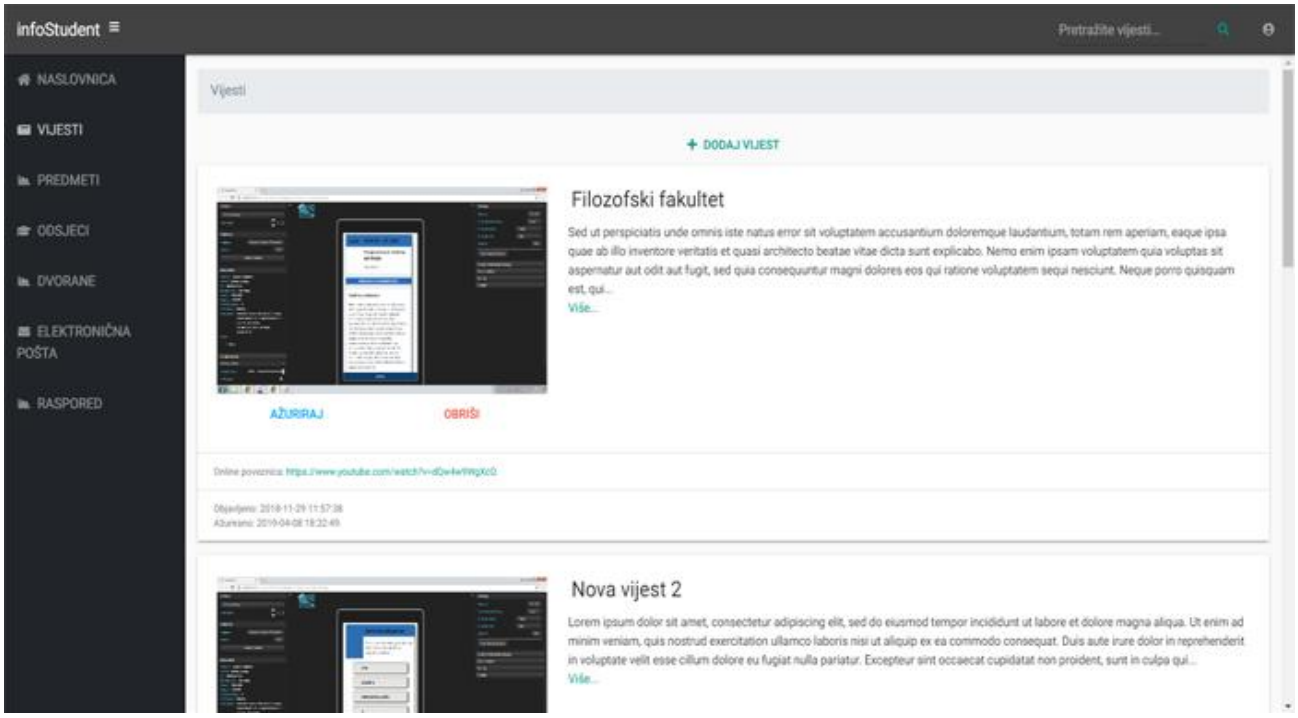
InfoStudent je zamišljen kao sustav namijenjen studentima i nastavnicima Filozofskog fakulteta Sveučilišta u Zagrebu. Sustav je osmišljen kao pomoć u organizaciji predmeta, predavanja, vježbi i seminara, te omogućuje profesorima objavljivanje vijesti, obavijesti i sličnih informacija koje su zanimljive studentima. Napravljen je uz pomoć web tehnologija i sastoji se od dva korisnička sučelja (sučelja namijenjenom internetskim preglednicima i sučelja namijenjeno mobilnim uređajima.) i pozadinskog servisa pomoću kojeg korisnici mogu preuzimati datoteke s poslužitelja i mijenjati sadržaj unutar baze podataka. Poslužitelju i svim podatcima i datotekama koji se nalaze na njemu je moguće pristupiti isključivo preko prethodno spomenutih sučelja. Internetskom sučelju je moguće pristupiti isključivo internetskim preglednicima preko osobnog računala, te u slučaju pristupa preko mobilnog internetskog preglednika, korisniku je omogućeno preuzimanje mobilne aplikacije.

InfoStudent je u suštini projektiran kao single-page aplikacija, taj pojam označava da se cijelo sučelje neće učitavati ispočetka pri otvaranju nove poveznice, već da će novi podatci dohvaćeni s poslužitelja biti prikazani unutar okvira namijenjenom prikazu podataka. Sustav slijedi smjernice REST arhitekture i za manipuliranje sadržajem koristi tehnologije navedene u prethodnim poglavljima rada.

5.1. Korisnici

Svi korisnici unutar ovoga sustava su studenti Filozofskog Fakulteta i za korištenje sustava je potrebno registrirati se putem mobilne aplikacije ili web sučelja preko vlastite **@ffzg.hr** adrese web pošte. Pri registraciji, sustav će na korisnikovu adresu e pošte poslati poveznicu za aktivaciju računa unutar sustava. Tako, sustav se ograničuje isključivo na studente fakulteta i njihove nastavnike. U sljedećem dijelu će biti objašnjene ovlasti koje posjeduju različite vrste korisnika i resursi kojima oni mogu pristupiti.

5.2. Vijesti



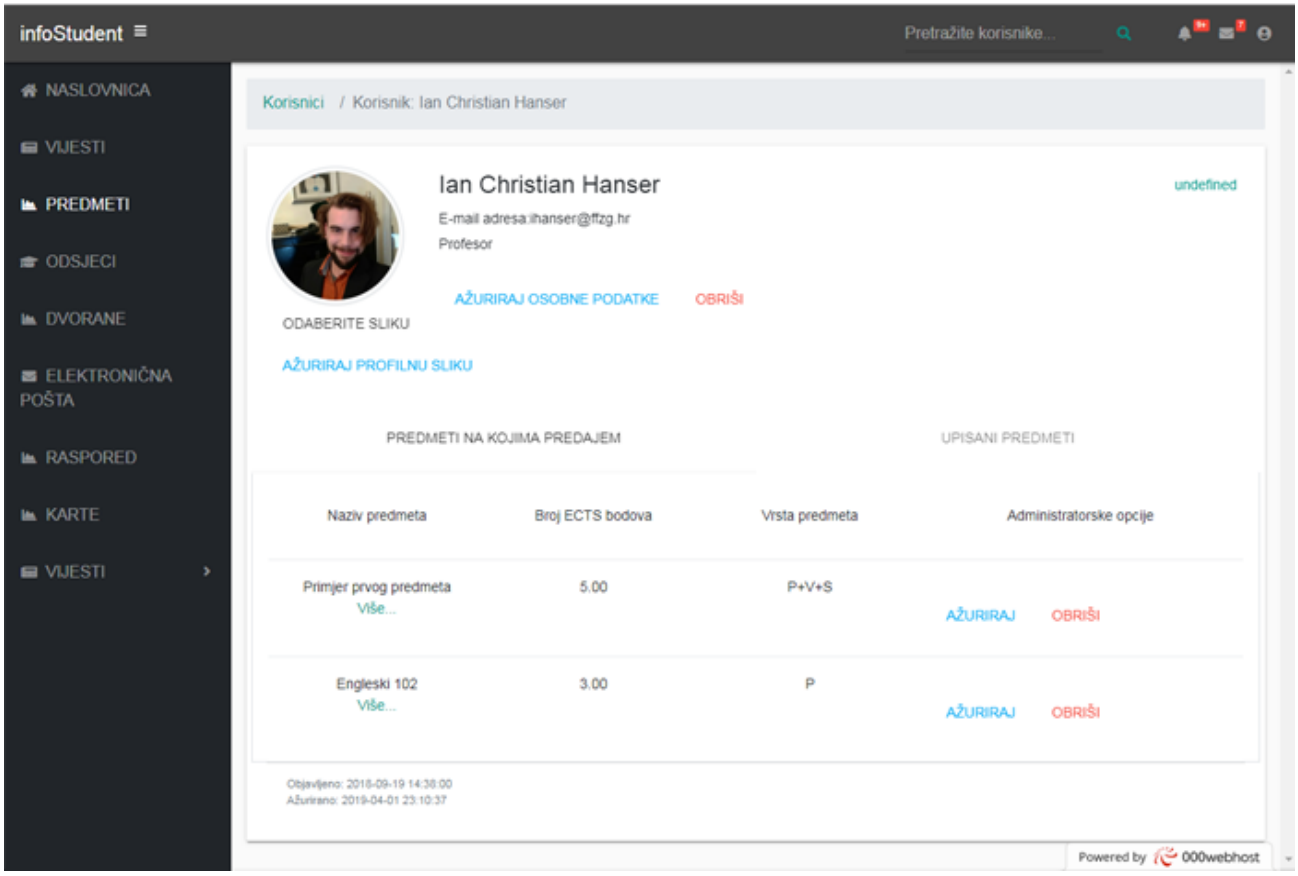
Slika 12. Primjer prikaza vijesti

Radnja	Ovlasti nastavnika	Ovlasti studenata
Unos novog vijesti	da	ne
Učitavanje vijesti	da	da
Ažuriranje vijesti	da	ne
Brisanje vijesti	da	ne

Tablica 14. - ovlasti vezane uz vijesti

Vijesti predstavljaju najjednostavniji resurs unutar cijelog sustava. One služe za prikazivanje novosti unutar fakulteta. Sastoje se od najjednostavnijih komponenata kao što su naslov, opis, slika i vanjska poveznica. Vijesti mogu pretraživati svi korisinici, dok je uređivanje, brisanje i stvaranje novih vijesti ograničeno isključivo za profesore. Vijesti je moguće pretraživati na temelju teksta (sustav će automatski pretraživati naslov i opis vijesti) ili na temelju datuma objave.

5.3. Osobni profil



Slika 13. Primjer osobnog profila korisnika

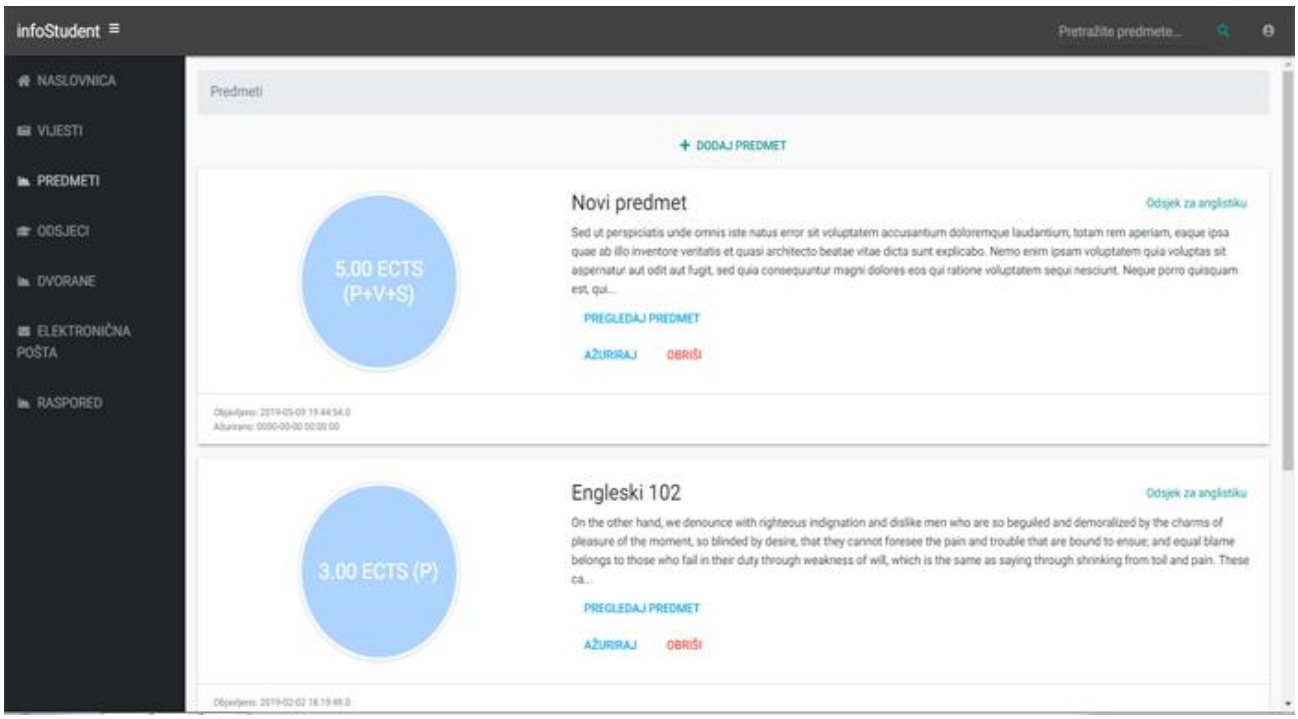
Radnja	Ovlasti nastavnika	Ovlasti studenata
Registracija korisnika	da (samo svojeg profila)	da (samo svojeg profila)
Učitavanje korisnika	da	da
Ažuriranje korisnika	da (samo svojeg profila)	da (samo svojeg profila)
Brisanje korisnika	ne (ova radnja je rezervirana za administratora)	ne (ova radnja je rezervirana za administratora)
Mijenjanje razine privilegija	ne (ova radnja je rezervirana za administratora)	ne (ova radnja je rezervirana za administratora)
Ažuriranje profilne slike	da (samo svojeg profila)	da (samo svojeg profila)
Povezivanje s mobilnim uređajem	da (samo svojeg profila)	da (samo svojeg profila)

Tablica 9. - Ovlasti vezane uz korisnike

Najvažniji resurs u InfoStudent sustavu su korisnici. Ovaj resurs može predstavljati profesora ili studenta u sustavu i preko njega je moguće prijaviti se u sustav InfoStudent. Bilo koji student ili nastavnik koji posjeduje email adresu filozofskog fakulteta (@ffzg.hr) se može registrirati u sustavu i kasnije se prijaviti. Pri registraciji, korisnik će na email adresu primiti poveznicu koja će aktivirati registrirani račun. Nakon registracije korisnik će se moći prijaviti u sustav kao student. Ako je navedeni korisnik zapravo profesor, trebat će kontaktirati administratora koji će promijeniti razinu privilegija koje su vezane uz novootvoreni račun.

Korisnike u sustavu je moguće pretraživati i pregledavati njihove profile, na svakome profilu je moguće vidjeti ime, prezime, email adresu, avatar korisnika (ako je postavljen,) i vrstu korisnika (student ili profesor.) Svim korisnicima je moguće poslati poruku uz napomenu da će poruka biti poslana preko sandučića sustava. Ako je profil koji se pregledava profil profesora, bit će prikazana dodatna informacija o mjestu i vremenu konzultacija. Na profilu profesora je također moguće vidjeti sve predmete na kojima taj profesor predaje.

5.4. Predmeti



Slika 14. Primjer prikaza većeg broja predmeta

Radnja	Ovlasti nastavnika	Ovlasti studenata
Učitavanje predmeta	da	da
Pretraživanje predmeta	da	da
Unos novog predmeta	da	ne
Ažuriranje predmeta	da	ne
Brisanje predmeta	da	ne
Upis studenata	da	Da (samo za sebe)
Ispis studenata	da	Da (samo za sebe)

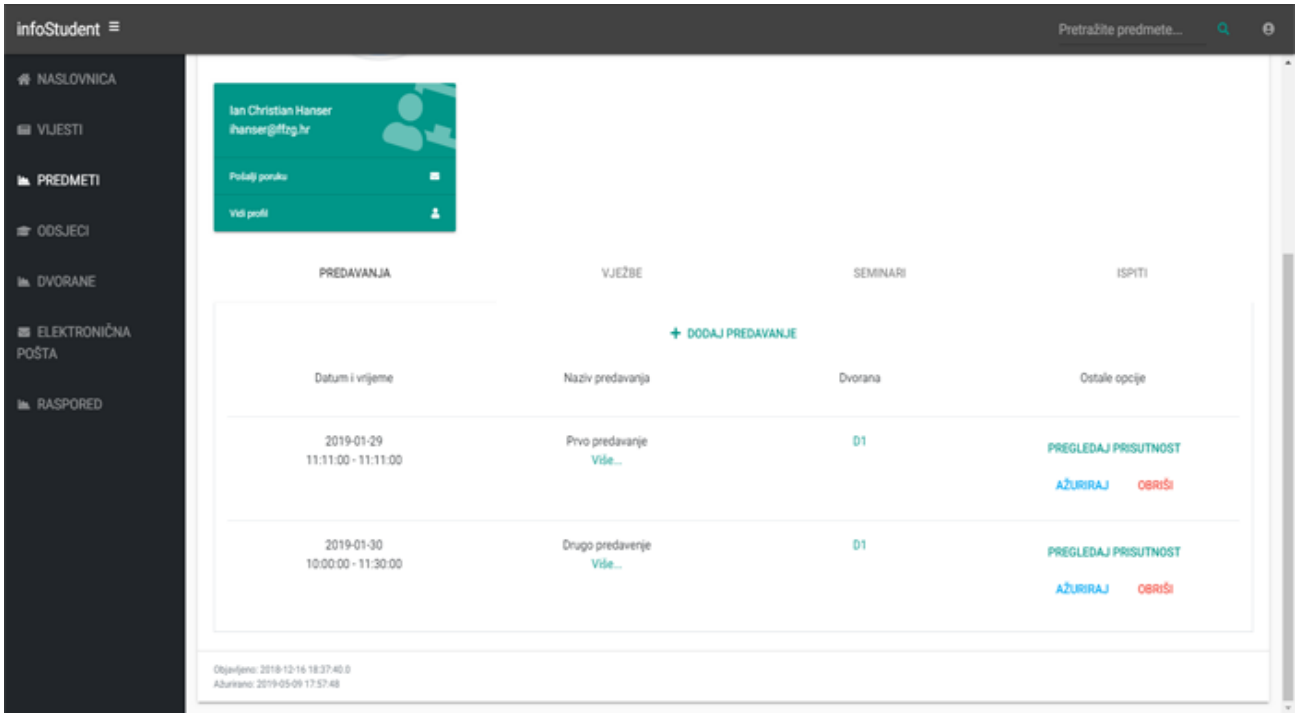
Tablica 10. - Ovlasti vezane uz predmete

Najvažniji resurs u InfoStudent sustavu je predmet. Pri pretraživanju je moguće vidjeti najosnovnije informacije vezane uz predmet. Za svaki predmet je moguće vidjeti njegov naslov, ime odsjeka ikoliko ECTS bodova taj predmet nosi. Broj bodova je opravdan satnicom i za svaki predmet je moguće vidjeti od koliko se sati predavanja, vježbi i seminara sastoji određeni predmet.

Ako student nije upisan u određeni predmet, student to može obaviti samostalno ili ga može upisati jedan od profesora na tome kolegiju. Profesorima je omogućeno postavljati dodatne uvijete za samostalan upis, odnosno, omogućeno im je postavljanje lozinke za upis kolegija i ograničenje o maksimalnom broju studenata na kolegiju. Profesorima je također omogućen upis studenata neovisno o maksimalno dopuštenom broju studenata (ako je definiran) ili postojanju lozinke sa kojom je kolegij zaštićen.

U detaljnom pregledu predmeta, moguće je vidjeti potpuni opis predmeta i popis profesora koji predaju na tome predmetu. Navedenim profesorima je moguće poslati poruku unutar aplikacije. Studentima se također omogućuje pregled predavanja, vježbi i seminara vezanih uz taj predmet, dok se profesorima omogućuje stvaranje, uređivanje i brisanje istih.

5.5. Predavanja / vježbe /seminari



Slika 15. Primjer popisa predavanja

Radnja	Ovlasti nastavnika	Ovlasti studenata
Pregled predavanja	da	da
Unos novog predavanja	da	ne
Ažuriranje predavanja	da	ne
Brisanje predavanja	da	ne
Potvrda prisutnosti	da	da (samo za sebe)
Brisanje prisutnosti	da	da (samo za sebe)

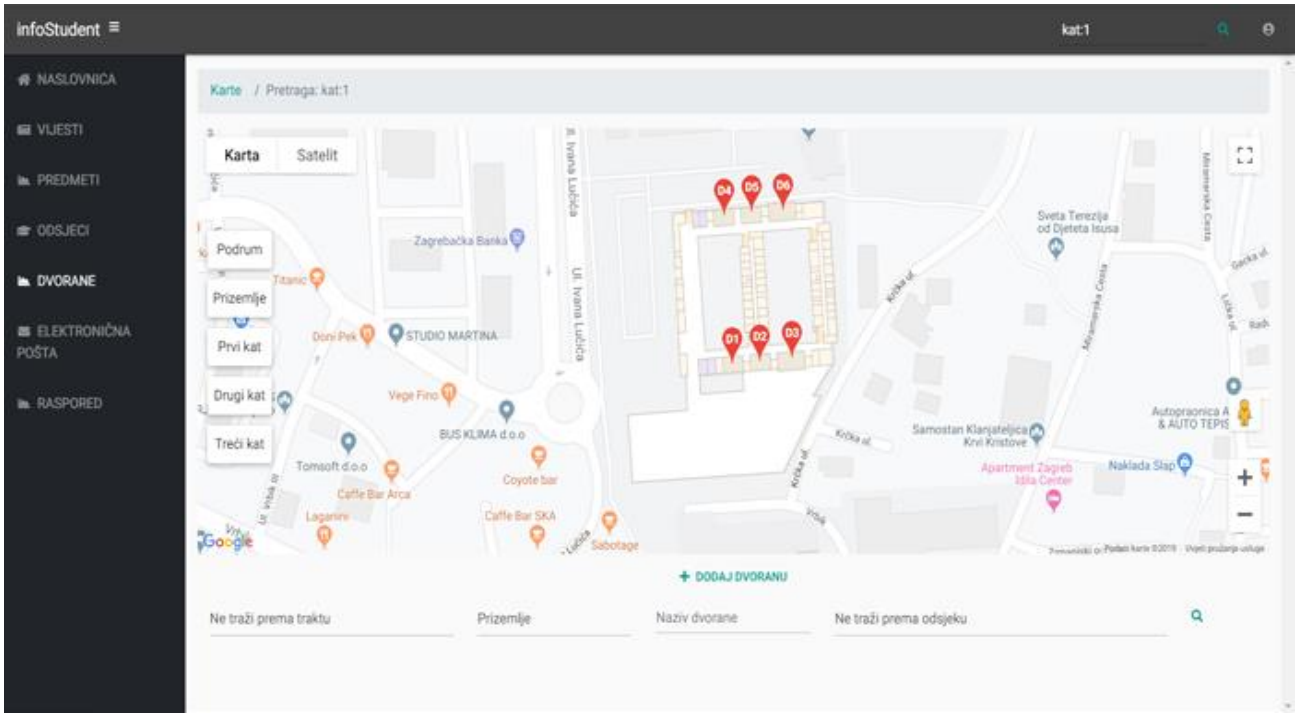
tablica11 - ovlasti vezane uz predavanje, vježbe i seminare

InfoStudent omogućava svojim korisnicima pregled predavanja, vježbi i seminara vezanih uz određeni kolegij, studentima se također omogućuje pregled prisutnosti na određenom predavanju vježbi ili seminaru. Ako student na određenom predavanju nema potvrđenu prisutnost, to može učiniti samostalno. Ako predavanje ima lozinku vezanu uz sebe studentu će upis biti omogućen jedino ako zna točnu lozinku. Profesorima se osim uređivanja, stvaranja i brisanja predavanja, vježbi i seminara omogućuje i pregled prisutnosti za sve studente koji pohađaju taj predmet. Osim pregleda statusa prisutnosti studenata, profesori mogu potvrditi prisutnost studenata pritiskom na jedan gumb. Ako profesori sumnjaju da student nije bio prisutan na predavanju, profesoru je omogućeno promijeniti studentov status prisutnosti.

Svim korisnicima su također odmah uočljive najosnovnije informacije vezane uz predavanje, vježbu ili seminar kao što su: naslov, opis, datum održavanja, početak, kraj i dvorana. Klikom na dvoranu, korisnicima se otvara novi prozor koji prikazuje tlocrt Filozofskog fakulteta, informaciju o katu koji se prikazuje i lokaciju označene dvorane. Pri pregledu pojedinačnog resursa, korisnicima je također omogućeno čitanje svih informacija i popis datoteka koje su namijenjene za predavanje, vježbu ili seminar.

Ako nastavnik odgodi određeno predavanje, vježbu ili seminar, novi datum će biti prikazan svim studentima. Ako su studenti koji pohađaju ova predavanja odobrili pohranjivanje identifikacijskog tokena na Google Firebase platformu i ako su odobrili primanje push notifikacija od strane InfoStudent sustava, njima će biti poslana obavijest o odgodi.

5.6. Karte - dvorane



Slika 16. Primjer prikaza fakultetskih dvorana

Radnja	Ovlasti nastavnika	Ovlasti studenata
Pregled dvorane	da	da
Unos nove dvorane	da	ne
Ažuriranje dvorane	da	ne
Brisanje dvorane	da	ne
Pretraživanje dvorane	da	da

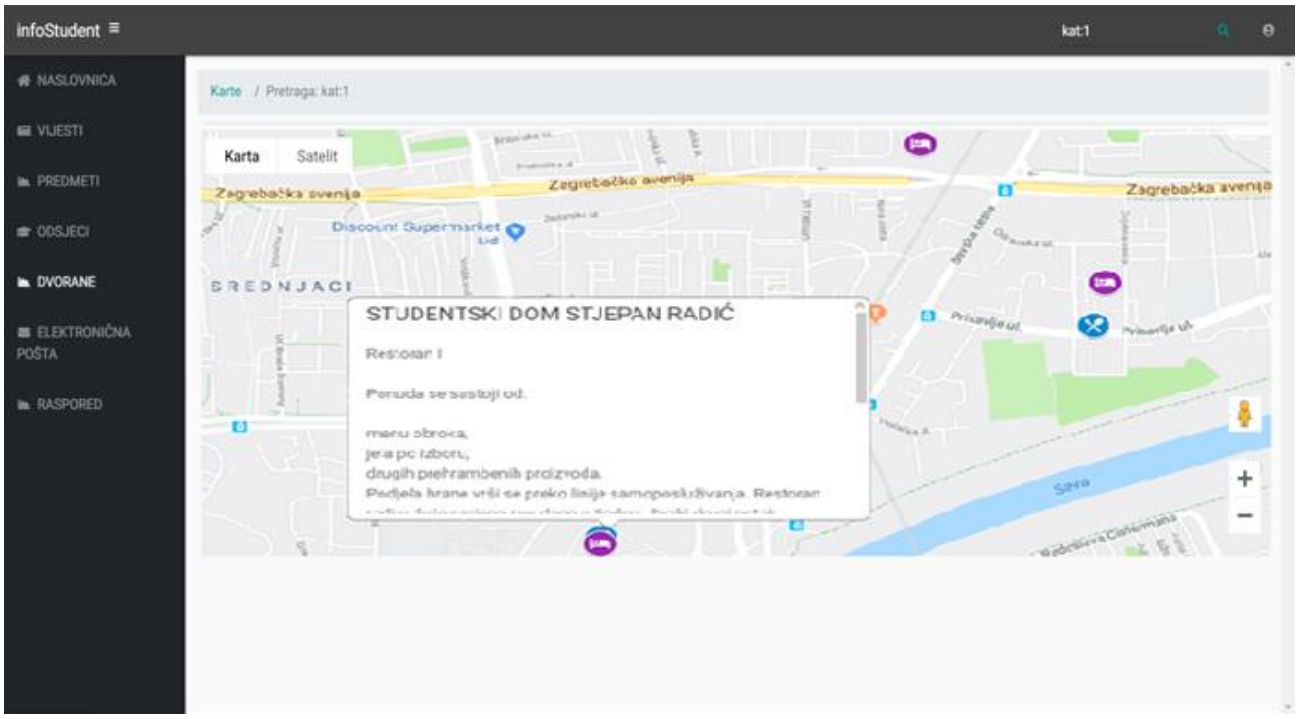
tablica12 - ovlasti vezane uz dvorane fakulteta i prikaz na karti

Korisnicima je omogućeno pregledavanje i pretraživanje dvorana unutar fakulteta. Razlog implementacije ove komponente je lakše snalaženje unutar prostorija fakulteta. Dvorane koje se prikazuju na kartama su često vezane uz ostale resurse i do određene dvorane je moguće doći putem poveznice ili pretraživanjem. Pretraživanje je moguće obaviti na više načina:

- na temelju kata
- na temelju trakta
- na temelju naziva dvorane
- na temelju odsjeka

Pri pretrazi će se prikazati tlocrt odgovarajućeg kata i sve dvorane koje odgovaraju upitu će biti prikazane na njihovoj točnoj lokaciji (kat treba uvijek biti definiran.) Dvorane kao resurs se često vežu uz ostale resurse kao što su predavanja, vježbe, seminari i kabinet u kojem se održavaju konzultacije određenog profesora.

Profesorima je omogućeno određivanje kabineta u kojemu održavaju konzultacije, vrijeme početka i vrijeme kraja konzultacija. Profesorima je također omogućeno definirati vrijeme početka,vrijeme kraja i dvoranu u kojoj će se održati ispit iz nekog predmeta.



Slika 17. Primjer prikaza karti koja sadrži popis korisnih lokacija

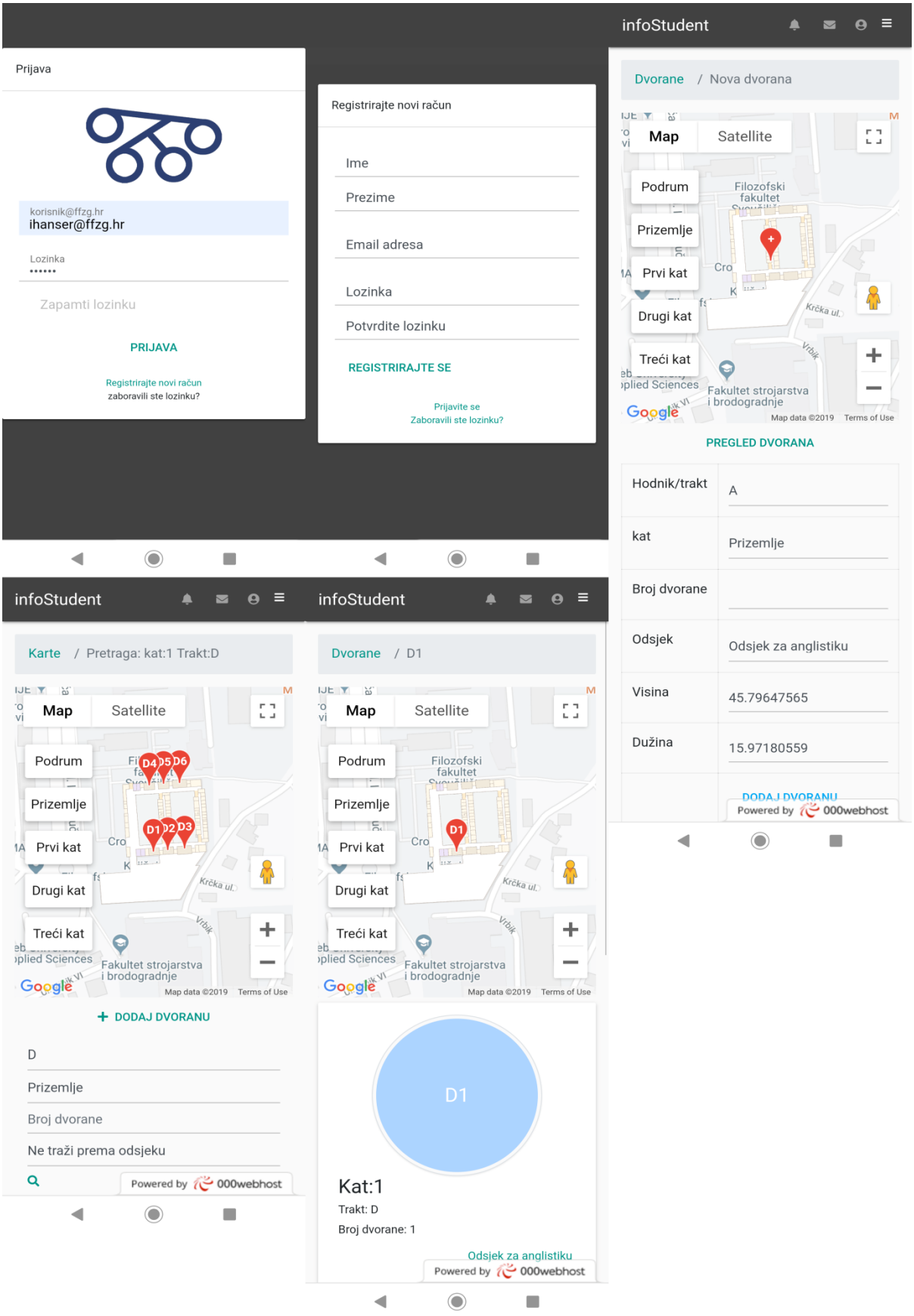
Radnja	Ovlasti nastavnika	Ovlasti studenata
Unos novih oznaka	ne	ne
Učitavanje oznaka	da	da
Ažuriranje oznaka	ne	ne
Brisanje oznaka	ne	ne

tablica13 - ovlasti vezane uz oznake na karti

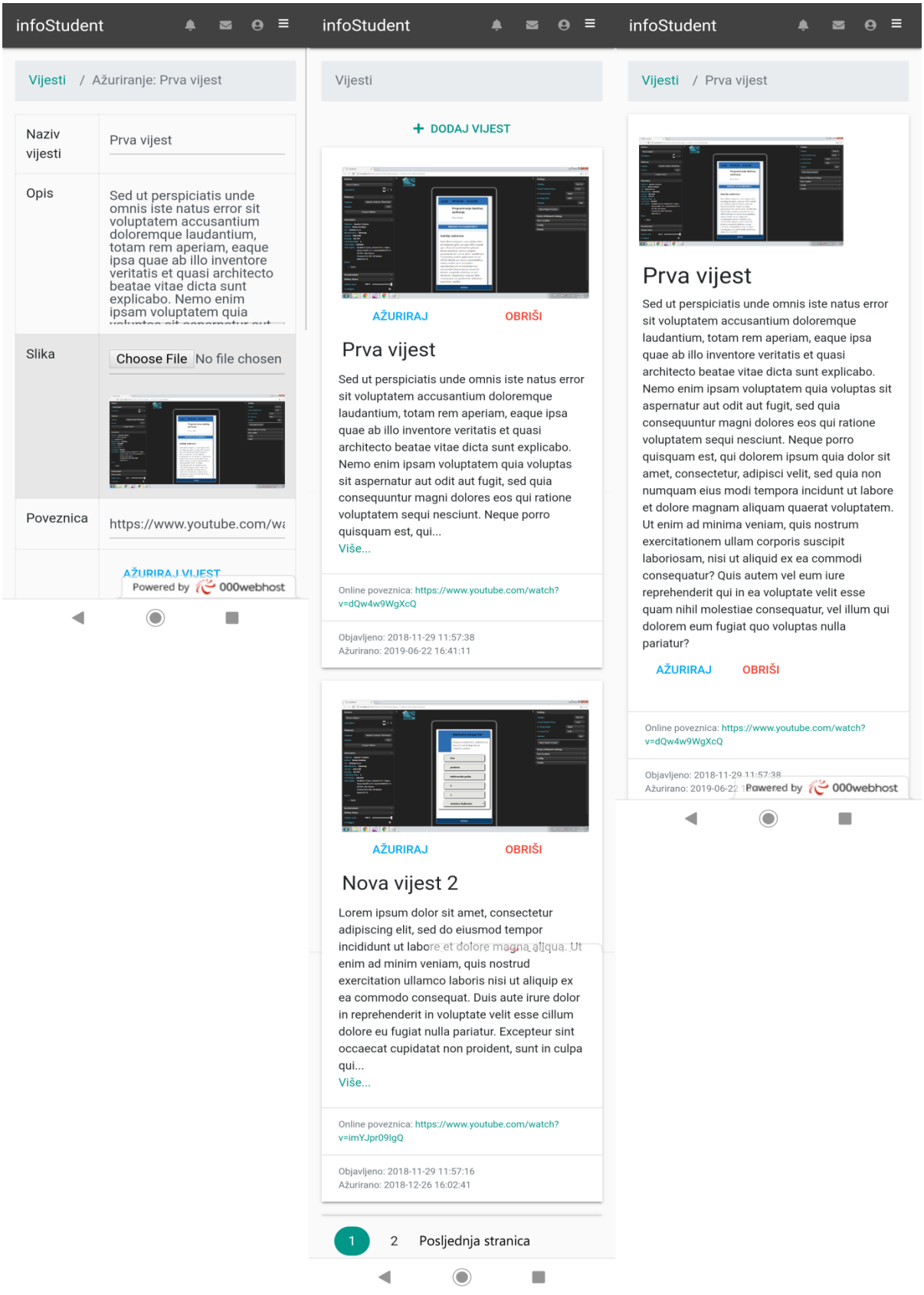
Unutar karata se također nalazi popis studentskih menzi, studentskih domova i studentskih centara. Ove oznake nije moguće pretraživati, već samo pregledavati. One su uvijek prisutne na bilo kojoj karti i na nju se vežu pri učitavanju same aplikacije.Ove oznake nisu pohranjene unutar baze podataka nego u zasebnoj datoteci

5.7. Mobilno sučelje

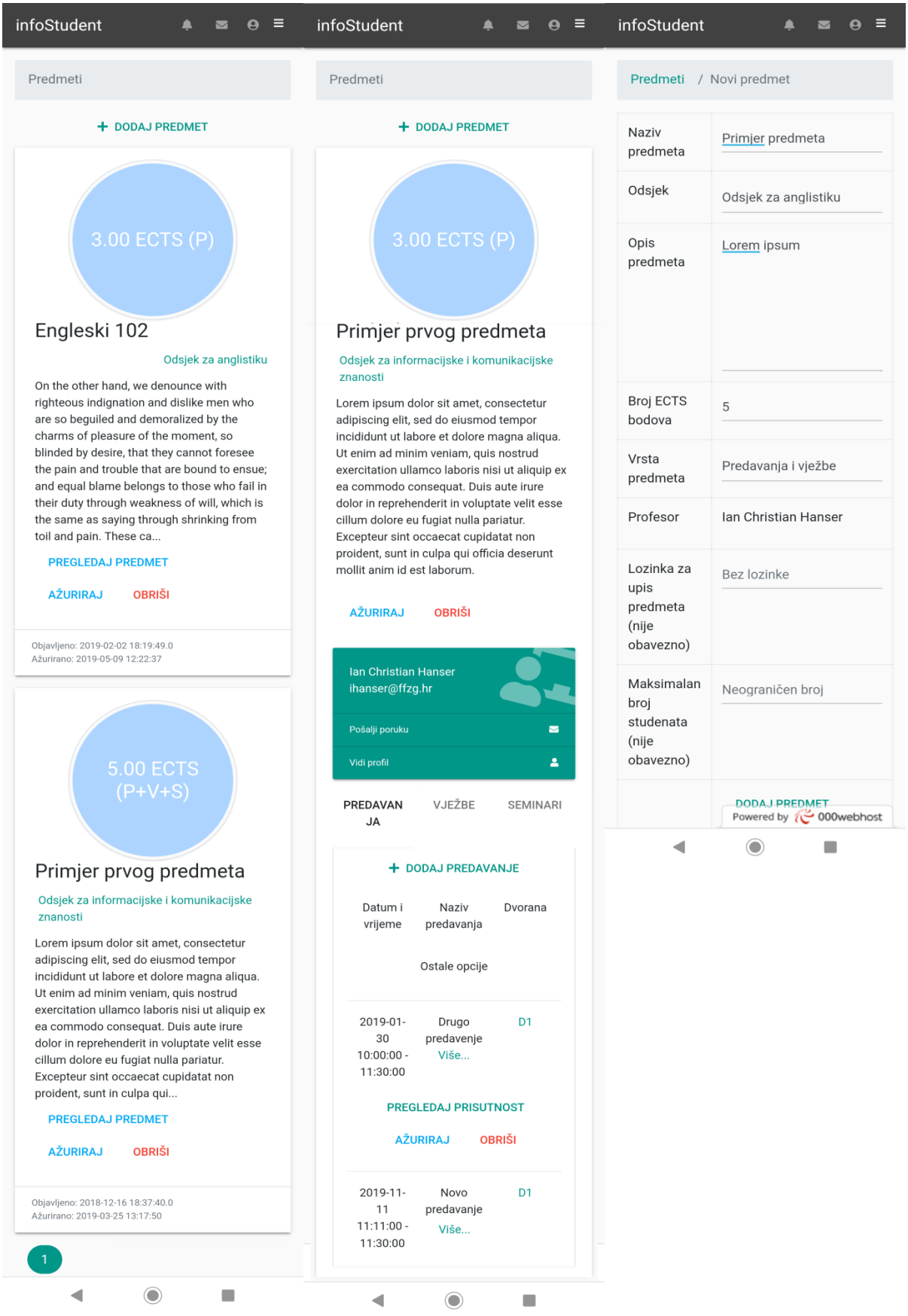
Mobilno sučelje je po funkcionalnosti isto kao i web sučelje. Osnovna razlika je što su sve datoteke potrebne za iscrtavanje sučelja i komunikaciju s poslužiteljem zapakirane u samu aplikaciju. Način pohrane tokena je također malo drugačiji i mobilna aplikacija poznaje token za osvježavanje. On se koristi kako bi se korisniku omogućila opcija da ostane prijavljen u vlastiti račun čak i kada ugasi aplikaciju. Unutar mobilnog sučelja je također moguće povezati vlastiti uređaj s profilom. Ta radnja poslužitelju omogućuje slanje push notifikacija na uređaj studenata.



Slika 18. Izgled mobilne aplikacije 1



Slika 19. Izgled mobilne aplikacije 2



Slika 20. Izgled mobilne aplikacije 3

6. Zaključak

Sa današnjom dostupnom tehnologijom izrada web stranica i mobilnih aplikacija nije nikada bila lakša. Web sučelja i frameworci su se razvili do te mjere da razvojni programeri mogu postići željene bez velike muke. Programski jezici su se razvili do nove razine gdje je detaljnu operaciju moguće izvršiti u svega nekoliko linija koda i prilagoditi za više uređaja. Međusobna podrška takvih frameworkova i libraryja osiguravaju konzistentnost izgleda i funkcionalnosti na različitim uređajima.

Smatram da su hibridne aplikacije, iako često sporije od nativnih, dobro riješenje za izradu jednostavnih sustava koji se oslanjaju na veliku količinu podataka. Sama činjenica da je moguće napisati jedan kod i isporučiti aplikaciju na veći broj mobilnih računalnih sustava upućuje na činjenicu da će se ove tehnologije nastaviti razvijati.

7. Literatura

7.1. Knjige

1.) Leonard Richardson, Sam Ruby. RESTful Web Services. 1st edition. California : O'Riley Media Inc. 2007.

2.) Leonard Richardson, Mike Amundsen. RESTful Web APIs. 1st edition. California : O'Riley Media Inc. 2013.

3.) David Sawyer McFarland. JavaScript and jQuery: The Missing Manual, Second edition. California : O'Riley Media Inc. 2012.

4.) Robin Nixon. Learning PHP, MySql & JavaScript. 4th Edition. Sebastopol : California O'Riley Media Inc. 2015.

5.) John M. Wargo. Apache Cordova 3 Programming. 1st Edition. Boston : Addison-Wesley Professional. 2012.

6.) Jake Spurlock. Bootstrap - Responsive Web Development. 1st edition. California : O'Riley Media Inc. 2013.

7.) Mark Lassoff, Tom Stachowitz. Mobile App Development with HTML5. 1st Edition. Conneticut : LearnToProgram.tv, incorporated. 2015.

7.2. Internet

1.) IETF - Internet Engieneering Task Force (svibanj 2015.) JSON Web Token (JWT) (RFC 7519). ISSN 2070 - 1721. URL: https://tools.ietf.org/html/rfc7519.html

2.) IETF - Internet Engieneering Task Force (svibanj 2015.) Json Web Algorithms (RFC 7518). ISSN 2070 - 1721. URL: https://tools.ietf.org/html/rfc7519.html

3.) IETF - Internet Engieneering Task Force (listopad 2006.) Base16, Base32 and Base64 Data encodings (RFC 4648). URL: https://tools.ietf.org/html/rfc4648

4.) Chris Esplin. What is Firebase, the complete story abridged.

URL:https://medium.com/Firebase-developers/what-is-Firebase-the-complete-story-abridged-bcc730c5f2c0

5.) Apache Corodova dokumentacija URL:https://Cordova.apache.org/docs/en/latest/

8. Slike

9. Tablice

Mobilna aplikacija za praćenje nastave i poboljšanje komunikacije između studenata i profesora

Sažetak

U ovome diplomskom radu su opisane osnovne karakteristike Apache Cordova Frameworka, njegove korištene tehnologije i karakteristike hibridnih aplikacija koje su produkt ovog sučelja. Radi činjenice da je Apache Cordova zasnovana na web tehnologijama HTML, CSS i JavaScript,) one će također biti opisane. Osim teorijskog dijela, ovaj rad također ima i praktični dio u kojemu je prikazana aplikacija namijenjena studentima i profesorima za olakšanje komunikacije i lakše praćenje nastave. U radu su prikazani odlike dizajna, programski kod i serverska arhitektura, te se ovaj rad također može koristiti kao primjer za daljnji razvoj mobilnih aplikacija unutar ovog sučelja.

Ključne riječi: REST api, hibridna aplikacija, web sučelje, single-page aplikacija,