Razvoj aplikacije e-zapisnik za praćenje košarkaške utakmice

Oblak, Domagoj

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:082232

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Image not found or type unknown

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

zir.nsk.hr

Image not found or type unknown

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

DOMAGOJ OBLAK

RAZVOJ APLIKACIJE E-ZAPISNIK ZA PRAĆENJE KOŠARKAŠKE UTAKMICE

Diplomski rad

Pula, rujan, 2020. godine

Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

DOMAGOJ OBLAK

RAZVOJ APLIKACIJE E-ZAPISNIK ZA PRAĆENJE KOŠARKAŠKE UTAKMICE

Diplomski rad

JMBAG: 0303061234, redoviti student

Studijski smjer: Diplomski sveučilišni studij Informatika

Predmet: Izrada informatičkih projekata

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Siniša Sovilj

Komentor: doc. dr. sc. Nikola Tanković

Pula, rujan, 2020. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Domagoj Oblak, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

U Puli, rujan, 2020. godine

Student

Domagoj Oblak

IZJAVA

o korištenju autorskog djela

Ja, Domagoj Oblak dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom „Razvoj aplikacije e-zapisnik za praćenje košarkaške utakmice“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, rujan, 2020. godine

Potpis

Domagoj Oblak

Sažetak

Tema ovog diplomskog rada je „Razvoj aplikacije e-zapisnik za praćenje košarkaške utakmice“. Detaljno je opisan razvoj i implementacija web aplikacije. Web aplikacije imaju široku primjenu u svim djelatnostima, većina aplikacija ima razvijenu i web verziju aplikacije jer je pristup takvim vrstama aplikacija brz i jednostavan i ne zahtijeva puno pripreme za korištenje. Opisane su sve tehnologije koje su korištene pri izradi web aplikacije, a to su *JavaScript*, *Python*, programski okvir *Vue.js*, programski okvir *Flask*, baza podataka *SQLite*, *HTML* za strukturu web aplikacije te *CSS* za dizajn pojedinih dijelova aplikacije.

Ključne riječi: *javascript, python, vue.js, flask, sqlite, html, css, web aplikacija, web servisi, sqlalchemy, front end, back end*

Abstract

In this graduation thesis "Development of an e-record application for monitoring a basketball game". The development and implementation of a web application is described in detail. Web applications have a wide application in all industries, most applications have developed and web version of the application because access to such types of applications is quick and easy and does not require much preparation for use. All technologies used in creating a web application are described, namely JavaScript, Python, Vue.js framework, Flask framework, SQLite database, HTML for web application structure and CSS for design of individual parts of the application.

Keywords: *javascript, python, vue.js, flask, sqlite, html, css, web application, web services, sqlalchemy, front end, back end*

Sadržaj

1. Uvod 1
2. REST (Representational State Transfer) 3
3. Korištene tehnologije 5
   3.1. Što je JavaScript 5
   3.2. Što je Python 6
   3.3. Programski okvir Vue.js 6
   3.4. Programski okvir Flask 7
   3.5. Baza podataka SQLite 8
   3.6. Hypertext Markup Language (HTML) 8
   3.7. Cascading Style Sheets (CSS) 9
4. Razrada funkcionalnosti 10
   4.1. Use Case dijagram 10
   4.2. Class dijagram 12
5. Motivacija 13
6. Prototip sučelja 14
7. Implementacija 19
   7.1. Front end 19
      7.1.1. Prijava i registracija 21
      7.1.2. Novosti 24
      7.1.3. Raspored utakmica 26
      7.1.4. Rezultati 28
      7.1.5. Timovi 29
      7.1.6. Zapisnik 30
   7.2. Back end 34
8. Korisničke upute 39
   8.1. Prijava i registracija 39
   8.2. Novosti, raspored i rezultati 41
   8.3. Timovi 44
   8.4. Službeni zapisnik 46
   8.5. Profil 48
   8.6. Pomoć 49
9. Zaključak 50

1. Uvod

Krajem prošlog stoljeća je došlo do pojave *World Wide Web-a* (*WWW*) i samim time veliki interes za razvijanje u početku statičke web stranice koje su bile pisane u samo u *HTML*-u (*Hypertext Markup Language*). Na početku se nije pridavala pozornost na izgled same web stranice nego je ona bila namijenjena kako bi korisnik što brže i jednostavnije dobio potrebne informacije, što dovodi do toga da se korisnik lakše snalazio i imao bolju preglednost informacija. Kako je popularnost weba rasla, web stranice su se počele sve više prilagođavati i upotrebljavati razni sadržaji. Razvijanjem raznih tehnologija s vremenom su se počele pojavljivati dinamične web stranice i aplikacije gdje se podaci mogu mijenjati dok se koristi određeni dio stranice ili aplikacije.

Web aplikacije imaju široku primjenu u svim djelatnostima, većina aplikacija ima razvijenu i web verziju aplikacije jer je pristup takvim vrstama aplikacija brz i jednostavan i ne zahtijeva puno pripreme za korištenje.

Glavna prednost web aplikacija je da ne zauzimaju prostor na korisnikovom računalu, već se cijela aplikacija nalazi na poslužitelju. Također nadogradnja se odvija na poslužitelju, tako korisnik ne mora izvršavati nikakve procedure kako bi nadogradio aplikaciju na naviju inačicu. Uz sve navedeno i štede vrijeme jer nema potrebe za instaliranjem aplikacije ili kasnije brisanjem jer je dovoljan samo web preglednik i internetska veza kako bi se pristupilo određenoj aplikaciji.

Nedostatci su također prisutni kod ovakve vrste aplikacija jer ovisi o brzini korisnikove internetske veze i dostupnosti poslužitelja kada je to potrebno. Možda i najveći problem kada se priča o web aplikacijama je sigurnost na internetu tj. opasnost od krađe podataka, neželjenih upada i virusa.

Košarkaško tržište je trenutno je na vrlo visokim granama, no prognoze su kako prava ekspanzija tog tržišta tek slijedi. Među nama kriju se mnogi koji su zaljubljenici u košarku iako se nekome sve čini kao velika zabava, brojke koje se kriju iza itekako to nisu. Ne čudi stoga ni sve veći interes onih koji žele osvojiti svoj udio na ovom brzo rastućem tržištu.

Cilj ovog rada je uspješno objasniti sve funkcije pojedinog dijela rada kako bi se navedena web aplikacija predstavila na što jednostavniji način.

Prvi dio rada bavi se *REST*-om, načinom komunikacije i stanjem *REST*-a, gdje je objašnjena svrha *REST*-a i načini komunikacije koji se odvijaju preko *HTTP* zahtjeva, kao i stanje *REST*-a koje može biti stanje resursa i stanje aplikacije.

U drugom poglavlju opisane su sve tehnologije koje su korištene pri izradi web aplikacije, a to su *JavaScript*, *Python*, programski okvir *Vue.js*, programski okvir *Flask*, baza podataka *SQLite*, *HTML* za strukturu web aplikacije te *CSS* za dizajn pojedinih dijelova aplikacije.

Treće poglavlje predstavlja razradu funkcionalnosti koje su prikazane preko dijagrama. Prvi dijagram odnosi se na *Use Case* dijagram koji prikazuje funkcionalnosti koje se korisniku se nude u aplikaciji. Drugi dijagram koji je potreban pri izradi aplikacije je *Class* dijagram, a smatra se glavnim dijagramom pri implementaciji aplikacijskih rješenja.

Četvrto poglavlje govori o motivaciji za izradu aplikacije i ciljanom tržištu. Također su predstavljena postojeća i konkurentska rješenja.

U petom poglavlju opisuje se prototip sučelja koji se koristi pri prezentaciji korisničkog sučelja dok još ne postoji implementacija sučelja. Tako se dobije uvid u dizajn aplikacije i njezine funkcionalnosti.

Šesti dio rada opisuje implementaciju web aplikacije kroz nekoliko potpoglavlja, a to su *front end* i *back end*. *Front end* predstavlja prikaz aplikacije koju korisnik vidi kada koristi, dok je *back end* zadužen za manipulaciju s podacima koje korisnik ne vidi.

U posljednjem dijelu rada prikazane su korisničke upute te na koji način se koristi aplikacija pomoću *screenshot-ova* sa stvarnog sučelja. Korisničke upute su namijenjene korisnicima aplikacije E-zapisnik kao pomoć pri korištenju aplikacije.

Poveznica na *Github* repozitorij je https://github.com/dooblak/e-zapisnik-application.

2. REST (Representational State Transfer)

Prije *REST*-a potrebno je znati što je *API*. *API* (*application programming interface*) je dio softvera koji priključuje jednu aplikaciju s podacima i servisima druge aplikacije tako što pruža pristup određenim dijelovima poslužitelja. *API*-ji omogućavaju komunikaciju dva dijela softvera što olakšava dijeljenje podataka između njih. *REST* (*Representational State Transfer*) se temelji na *URL*-u i *HTTP* protokolima. Za razmjenu podataka uglavnom se koristi *JSON* format koji je kompatibilan s web preglednicima odnosno *JavaScript*-om. *REST API*-ji mogu biti jednostavni za izgradnju i skaliranje, a mogu biti i veliki i komplicirani što ovisi o tome za što su napravljeni da rade. Razlozi zbog kojih bi se trebao koristi *API* koji će biti *RESTful* je ograničenje resursa, manji sigurnosni zahtjevi, kompatibilnost klijenta, lakši pristup podacima i skalabilnost (e-ucenje.unipu.hr, 2020).

2.1. Način komunikacije s REST-om

Informacija o vrsti zahtjeva ugrađuje se u *HTTP* zahtjev kao *method*. Pet najčešćih *HTTP* metoda su:

- *GET*: služi za dohvaćanje podataka

```
curl ^
  --location ^
  --include ^
  --request GET "https://postman-echo.com/get?foo1=bar1&foo2=bar2"
```

Slika 1. GET metoda (e-ucenje.unipu.hr)

- *POST*: spremanje podataka

```
curl ^
  --location ^
  --include ^
  --request POST "https://postman-echo.com/post" ^
  --data "This is expected to be sent back as part of response body."
```

Slika 2. POST metoda (e-ucenje.unipu.hr)

- *PATCH*: ažuriranje određenog podatka
- *PUT*: ažuriranje podataka

```
curl ^
  --location ^
  --include ^
  --request PUT "https://postman-echo.com/put" ^
  --data "This is expected to be sent back as part of response body."
```

Slika 3. PUT metoda (e-ucenje.unipu.hr)

- *DELETE*: brisanje podataka

```
curl ^
  --location ^
  --include ^
  --request DELETE "https://postman-echo.com/delete" ^
  --data "This is expected to be sent back as part of response body."
```

Slika 4. DELETE metoda (e-ucenje.unipu.hr)

Uz *HTTP* metode postoje i *HTTP* kodovi za status odgovora na zahtjev te označavaju je li određeni *HTTP* zahtjev uspješno završen. Oni se grupiraju u 5 klasa:

- Informativni (100 – 199)
- Uspješni (200 – 299)
- Preusmjereno (300 – 399)
- Greška klijenta (400 – 499)
- Greška poslužitelja (500 – 599)

2.2. Mogućnosti stanja (state) REST-a

U navedenom *RESTful* servisu moguća je pojava dva stanja *(state)*, a to su stanje resursa i stanje aplikacije. Stanje resursa su informacije o resursima koje su sadržane na poslužitelju, a stanje aplikacije su informacije koje su sadržane na strani klijenta sve dok se ne pojavi potreba za kreiranjem, ažuriranjem ili brisanjem određenog resursa. Stoga se u tom slučaju stanje šalje na poslužitelja i tada počinje biti stanje resursa. *RESTful* servis je „bez stanja“ ako poslužitelj nikada ne sprema stanje aplikacije. Tako u aplikaciji ako je „bez stanja“, poslužitelj obrađuje svaki zahtjev kao zaseban slučaj (e-ucenje.unipu.hr, 2020).

3. Korištene tehnologije

Pri implementacija E-zapisnik aplikacije korišteni su programski jezici *JavaScript* i *Python*. Za *JavaScript* korišten je programski okvir *Vue.js* gdje je još potrebno poznavati osnovne funkcionalnosti *HTML*-a i *CSS*-a kako bi se napravila cjelokupna aplikacija, programski okvir *BootstrapVue* je korišten je u zamjenu za *CSS*. Uz sve navedeno za *back end* dio aplikacije korišten je programski okvir *Flask* koji olakšava ponovnu upotrebu koda za uobičajene *HTTP* operacije kao što su *post*, *get*, *put* i *delete*. *SQLite* baza podatak je korištena zbog razlike od svih ostalih jezika *SQL*-a jer koristi sustav dinamičkog tipa, što znači da vrijednost pohranjena u stupcu određuje njegov tip podataka, a ne vrstu podataka stupca.

3.1. Što je JavaScript

Programski jezik *JavaScript* je u početku zamišljen kako bi pomogao da web stranice postanu dinamičnije. Programi koji se stvaraju programiranjem u navedenom jeziku nazivaju se skripte. Mogu se pisati u istom dokumentu gdje se piše i *HTML* web stranice i pokrenuti automatski kada se određena stranica učita. Skripte se izvode u obliku običnog teksta te nije potreban kompajler ili posebne pripreme kako bi se izvršila skripta. Kada je *JavaScript* kreiran imao je naziv „*LiveScript*“, ali u to vrijeme programski jezik *Java* je bila vrlo popularna, stoga je odlučeno kako naziv *JavaScript* bi mogao pomoći da bude što popularniji među već tada dosta velikim brojem programskih jezika. Tijekom razvitka postao je potpuno neovisan jezik s vlastitim specifikacijama i sada uopće nema nikakve veze s *Javom*.

Danas se *JavaScript* izvršava i na serveru, a ne samo u pregledniku. Također ima mogućnost da izvršava na bilo kojem uređaju koji ima *JavaScript engine*. Preglednik ima ugrađeni *engine* koji se ponekad može naći pod nazivom „*JavaScript virtual machine*“. Različiti *engine*-i imaju različite nazive, kao npr. V8 je naziv za *engine* koji radi u *Chrome*-u i *Operi*, dok je naziv *SpiderMonkey* za *Mozillu Firefox*. Rad *engine*-a je vrlo jednostavan i može se objasniti u tri koraka. Prvi korak je da *engine*, ako je riječ o pregledniku čita određenu skriptu, zatim pretvara (kompajlira) skriptu u strojni jezik, a onda strojni kod se pokreče vrlo brzo.

Postoje tri stvari koje čine *JavaScript* tako poželjnim pri kreiranju web stranica i aplikacija. Prva je potpuna integracija s *HTML*-om i *CSS*-om. Druga je da se

jednostavni problemi mogu riješiti na jednostavan način. Zadnja je to da je podržava od svih glavnih preglednika koji se koriste pri korištenju Interneta (JavaScript.info, 2020).

3.2. Što je Python

Programski jezik *Python* se na početku smatrao jezikom koji popunjava praznine za druge programske jezike. Služio je za pisanje skripti koje „automatiziraju dosadne stvari“ ili brzim prototipiranjem za aplikacije koje će se implementirati na drugim jezicima. Međutim, *Python* se razvio u prvorazredni programski jezik u modernom razvoju softvera, upravljanju infrastrukturom te u analizi podataka. Više nije tzv. jezik za popunjavanje praznina nego je glavna sila u kreiranju web aplikacija, upravljanju sustavima i strojnoj inteligenciji (InfoWorld, 2019).

Python je razvio *Guido van Rossum* krajem osamdesetih godina na Nacionalnom istraživačkom institutu za matematiku i računalne znanosti u Nizozemskoj. *Python* je izveden iz mnogih drugih jezika, uključujući *ABC*, *Modula-3*, *C, C ++, Algol-68, SmallTalk i Unix.* Neke glavne značajke *Pythona* su da je lagan za naučiti jer ima malo ključnih riječi, jednostavnu strukturu i jasnu sintaksu. Također jednostavan je i za čitanje jer je jasno definiran i razlike su lako vidljive očima. Izvorni kod *Pythona* je prilično lak za održavanje. Široka upotreba raznih biblioteka koje su kompatibilne s različitim platformama. Pruža sučelja za sve glavne komercijalne baze podatka. Vrlo je skalabilan i pruža podršku velikim skriptama (tutorialspoint, 2018).

3.3. Programski okvir Vue.js

Vue je *open-source JavaScript* progresivni programski okvir kojeg je razvio Evan You, bivši zaposlenik *Google*-a. Sličan je *Angularu* i *Reactu* jer kombinira značajke kao navedeni okviri. Iza *Vue*-a ne stoji velika tehnološka kompanija već tim koji održava i upravlja njime te je svakako doživio značajan pomak u popularnosti. Cjelovit je okvir, ali s manje značajki i paketa koji se mogu koristiti pri izradi određenih aplikacija.

Dokumentacija o *Vue*-u je opsežna te pruža pouzdan uvid u jednostavnost njegove sintakse. Svi programeri, nebitno o kojoj se razini iskustva *JavaScripta* i *HTML*-a radi, trebao bi im poslužiti kao jednostavna opcija. Jedan je od najpopularnijih *JavaScript* okvira na *GitHub-u* zahvaljujući njegovim značajkama koje pruža i sposobnosti

stvaranja učinkovitih, brzih i sofisticiranih aplikacija na jednoj stranici (Schwarzmüller, 2019).

Neke od prednosti su dobre performanse koje pokrivaju veliki dio onoga što *Vue* čini izvrsnim alatom, jednostavnost upotrebe tj. jednostavan je za učenje te isto privlači i početnike i profesionalne programere. Zatim, sjajna dokumentacija koju je tim *Vue*-a detaljno napisao, a korisnicima je korisno pri izradi određene aplikacije i laka integracija u druge projekte, što omogućuje da se odmah počne koristiti *Vue* u projektu. Nedostatci su relativno mala zajednica koja ipak raste. Budući da je novi okvir, trebat će još neko vrijeme da svoju zajednicu proširi na zavidnu razinu. Uz gore navedeni problem proizlazi i problem manjeg tržišta rada za koji će, također, trebati neko vrijeme prije nego što se na tržištu rada uoči velika potražnja za novim i kvalificiranim *Vue.js* programerima (Borrelli, 2019).

3.4. Programski okvir Flask

Programski okvir *Flask* se bazira na programskom jeziku *Python*. *Flask* olakšava ponovnu upotrebu koda za uobičajene *HTTP* operacije kao što su *post*, *get*, *put*, *delete*. U osnovi, navedeni okvir obuhvaća rad koji su programeri naučili u posljednjih dvadeset godina dok su programirali aplikacije i web stranice.

Flask je relativno mlad okvir, upotrebljava se tek od 2010. godine. Više se oslanja na programski jezik *Python* nego na programski okvir *Django* samo zato što je kod web aplikacije u *Flask*-u u većini slučajeva eksplicitniji. *Flask* je izbor većine početnika jer nema prepreka za postavljanje i pokretanje jednostavne web aplikacije.

Neke od prednosti su izuzetna fleksibilnost pri izradi raznih vrsta aplikacija. Lak je za održavanje i ne zahtijeva puno procesorske snage te je jednostavno za učenje i korištenje. Usmjeravanje *URL*-ova je jednostavno i brzo gdje se može lako manipulirati podacima koji se šalju na *front-end* ili bazu podataka.

Neke mane su mu da nema „prijateljski odnos“ s *async* funkcijom. Ograničena podrška i dokumentacija također stvaraju problem kada programer treba detaljan uvid u rad određene funkcije, kao i dosta slaba pokrivenost s mogućnostima koje nudi.

3.5. Baza podataka SQLite

SQLite se razlikuje od svih ostalih jezika *SQL*-a jer koristi sustav dinamičkog tipa, što znači da vrijednost pohranjena u stupcu određuje njegov tip podataka, a ne vrstu podataka stupca. Može komunicirati i sa *SQLite* bazom podataka koristeći *Java*, *Python*, *PHP* i *Node.js*. Najveća prednost *SQLite*-a je jednostavna upotreba, može se postaviti na bilo koji stroj pa čak i na mobitel. Ne zahtijeva mnogo konfiguracije.

Postavljanje je brzo i jednostavno, a korištenje jezika je, također, jednostavno. Ne treba brinuti o podatkovnom centru ili moćnoj mreži jer i bez toga radi vrlo brzo. Sljedeća je zanimljivost *SQL*-a da stupac u *SQLite*-u može pohraniti različite vrste podataka. Stoga, ako skup podataka nije očišćen i dalje se može otvoriti i zatražiti. Međutim, treba imati na umu sortiranje vrijednosti jer različite vrste podataka mogu utjecati na redoslijed rezultata. Najveći nedostatak su njegove primitivne sintakse i ograničenja oblikovanja.

Za razliku od *PostgreSQL*-a ili *MySQL*-a, *SQLite* ne podržava više funkcija. Moguće je da će doći do problema ako se *SQLite* koristi prilikom složenije manipulacije s podacima (poput višestrukih istovremenih operacija pisanja). Također, svaka datoteka *IMPORT* čita podatke samo kao *TEXT*, a nije ih jednostavno preoblikovati s osnovnom ograničenom sintaksom. Drugi problem je što *SQLite* ne podržava klase datuma i vremena. Kao što je rečeno, postoje ugrađene funkcije datuma i vremena koje mogu biti korisne.

3.6. Hypertext Markup Language (HTML)

HTML omogućuje korisniku strukturiranje i stvaranje dijelova, odjeljaka, odlomaka, naslova, veza i blokova za web stranice i aplikacije. *HTML* se ne smatra programskim jezikom što se odnosi na to da nema mogućnost da se kreiraju dinamičke funkcionalnosti. Odnosno ima mogućnost da se organiziraju i formatiraju dokumenti kao u tekstualnom editoru. Pri radu s *HTML*-om koriste se jednostavne strukture za označavanje web stranice ili aplikacije. Npr. odlomak se stvara pomoću taga * Ovo je odlomak .* Svi dokumenti imaju .html ekstenziju jer se tako može pogledati s bilo kojim web preglednikom (*Google Chrome*, *Safari*, *Mozilla Firefox*). Preglednik čita i strukturira *HTML* stranicu te ju prikazuje onakvu kakva je napisana u .html datoteci. Svaka web stranica prosječno ima nekoliko html dokumenta koje predstavljaju različite

stranice. Npr. početna stranica, kontakt stranica su odvojene jedna od druge. (Hostinger, 2019).

3.7. Cascading Style Sheets (CSS)

CSS je kratica za *Cascading Style Sheets* gdje je veliki naglasak na *Style*. Kao što je navedeno u prethodnom poglavlju *HTML* se koristi za strukturiranje web stranica tako što definira stvari poput naslova i odlomka, tako *CSS* određuje stil web stranice, a to su raspored stranica, boje, fontovi, razmak između elementa itd. Znači *HTML* je temelj, *CSS* je alat s kojim uređujemo stil web stranice. *CSS*-om se unosi jedinstveni stil web stranice ili aplikacije kako bi ona bila unikatna. Svaki element je pojedinačna *HTML* komponenta koja se može urediti na zaseban način kako bi se uklopio u stil web stranice (skillcrush, 2018).

4. Razrada funkcionalnosti

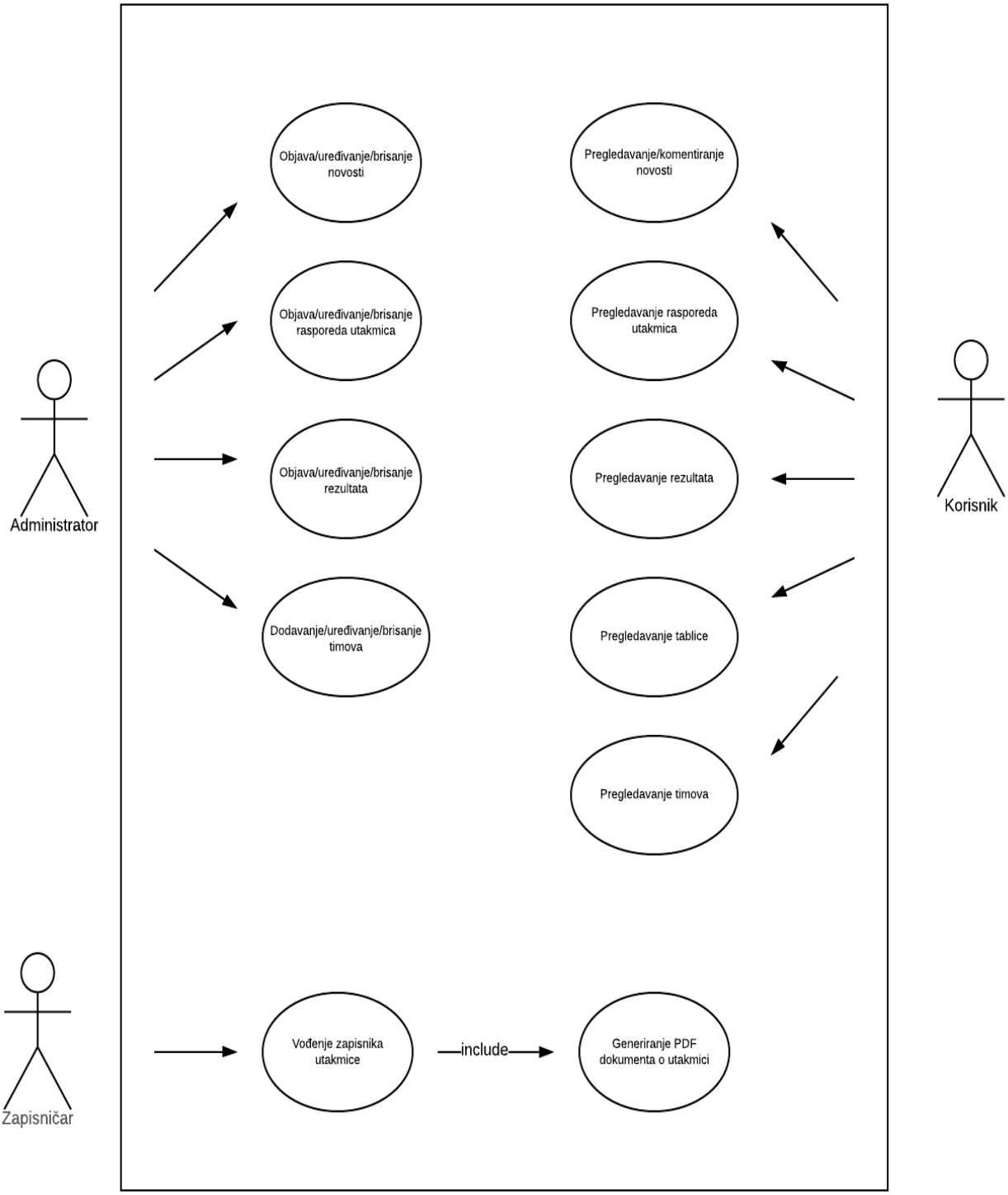
Korisnici su administrator, korisnik i zapisničar, a njihove funkcionalnosti bit će detaljno opisane u nastavku, kao i *Use Case* dijagram cijelog sustava, te *Class* dijagram gdje će se pojasniti ključni dijelovi aplikacije.

4.1. Use Case dijagram

Funkcionalnosti koje su prikazane na dijagramu korisniku se nude u aplikaciji. Prva funkcionalnost je pregledavanje i komentiranje objavljenih novosti gdje korisnik može vidjeti i komentirati sve objavljene novosti od strane administratora. Zatim ima mogućnost pregledavanja rasporeda utakmica gdje su objavljene utakmice koje će se igrati u narednom periodu. Za svaku utakmicu može se vidjeti gdje se igra, kada se igra, tj. datum i vrijeme te tko su domaćini, a tko gosti u određenoj utakmici. Treća funkcionalnosti odnosi se na pregledavanje rezultata gdje se točno mogu vidjeti rezultati odigrane utakmice. Sljedeća funkcionalnosti nudi pregledavanje tablice gdje se vide svi timovi te njihovi statistički atributi kao što su pobjede, porazi, odigrani susreti, bodovi, ukupan broj koševa itd. Zadnja funkcionalnosti koju ima korisnik na raspolaganju je pregledavanje timova određene košarkaške lige, gdje pišu detalje informacije o određenim igračima u određenim timovima.

Sljedeći korisnik je administrator koji ima uvid u velik broj podataka, kao i zadaću da dodaje nove podatke, uređuje postojeće podatke, te brisanje određenih podataka po potrebi. Tako administrator ima funkcionalnosti da objavljuje, uređuje i briše novosti, raspored, rezultate i timove.

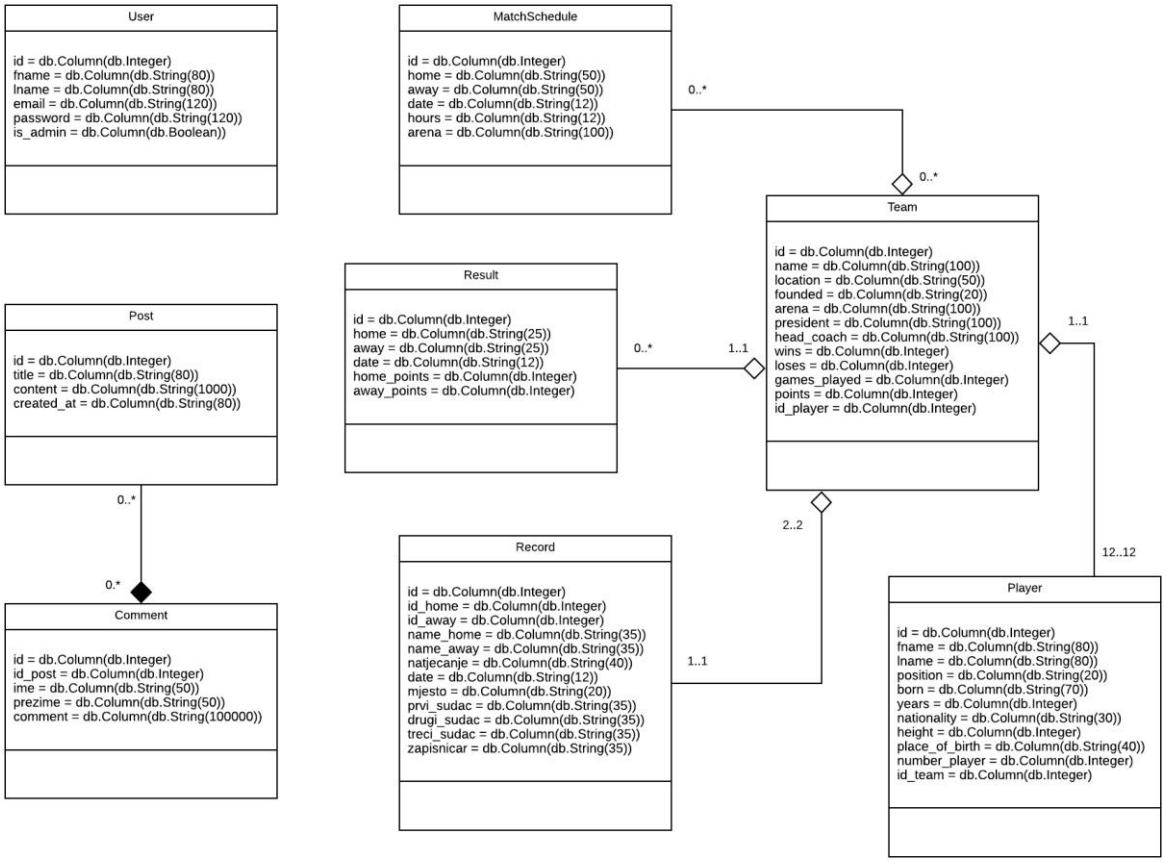
Glavni korisnik koji sudjeluje u vođenju zapisnika je zapisničar koji na kraju utakmice generira PDF dokument gdje pišu svi detalji o utakmici, kao što su igrači i njihove statistike, poeni, osobni prekršaji, tehnički prekršaji, skokovi, ukradene lopte i blokade.



Slika 5. Use Case dijagram (Izrada studenta)

4.2. Class dijagram

Drugi dijagram koji je potreban pri izradi aplikacije je klasni dijagram, a smatra se glavnim dijagramom pri implementaciji aplikacijskih rješenja. Bit će objašnjeni ključni dijelovi, a to su klase, atributi i veze među klasama. Prva klasa odnosi se na korisnika koji sadrži atribute *fname*, *lname*, *email*, *password*. Druga se odnosi na novosti koje su povezane s komentarima preko mehanizma kompozicije. Kompozicija omogućuje ako se obriše novosti da se briše i klasa komentari. Sljedeće klase su raspored, rezultati, zapisnik i igrač koje su povezane s klasom *team* preko agregacije, što znači ako se briše *team*, ostale klase se neće obrisati.



Slika 6. Class dijagram (Izrada studenta)

5. Motivacija

Košarkaško tržište je trenutno je na vrlo visokim granama, no prognoze su kako prava ekspanzija tog tržišta tek slijedi. Među nama kriju se mnogi koji su zaljubljenici u košarku iako se nekome sve čini kao velika zabava, brojke koje se kriju iza itekako to nisu. Ne čudi stoga ni sve veći interes onih koji žele osvojiti svoj udio na ovom brzo rastućem tržištu.

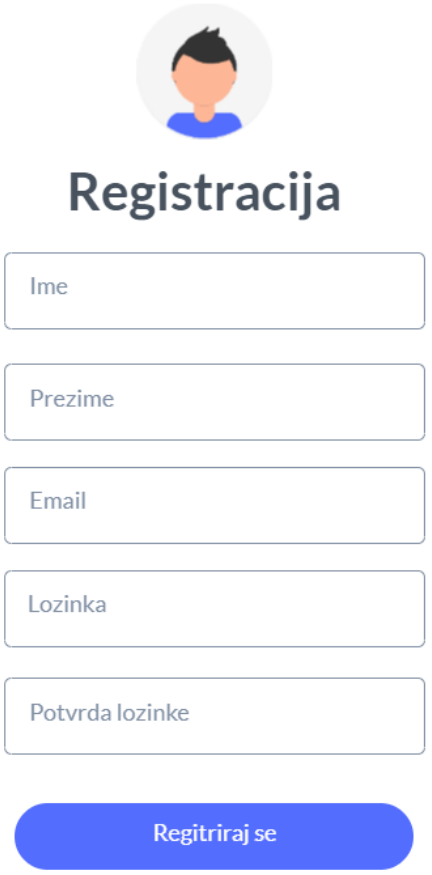
Ciljano tržište za aplikaciju bi bilo košarkaško tržište. Aplikaciju bi prvo koristio manja liga, a ako bi se pokazala uspješnom mogli bi je uvesti na sve lige u Republici Hrvatskoj. Postojećih i konkurentskih rješenja trenutno nema, već je riječ o inovaciji. Do sada se zapisnik vodio na papiru gdje bi zapisničar križao određene kvadratiće koji su imali određenu namjenu tijekom utakmice. Kroz *SWOT* analizu bit će istaknute najbitnije prednosti, nedostatci, mogućnosti i prijetnje.



Slika 7. SWOT analiza (Izrada studenta)

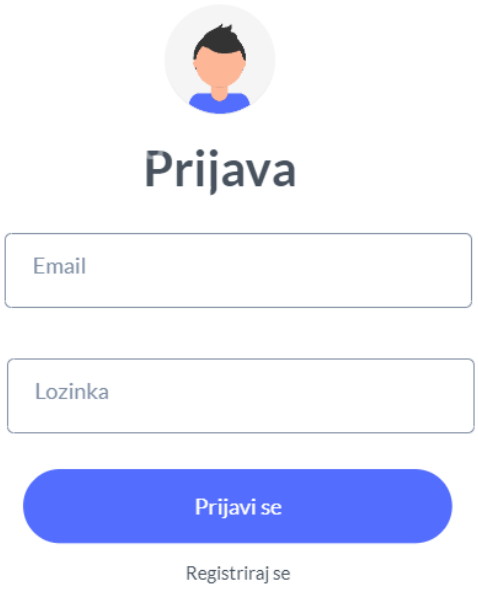
6. Prototip sučelja

Prototip sučelja se koristi pri prezentaciji korisničkog sučelja dok ne postoji implementacija samog sučelja. Tako se dobije uvid u dizajn aplikacije i njezine funkcionalnosti. Početni zaslon aplikacije kada je korisnik pokrene daje mogućnost da se registrira ako nema korisnički račun. Klikom na registraciju korisnik mora unesti ime, prezime, *email* te lozinku kako bi dobio pristup stranici za prijavu.



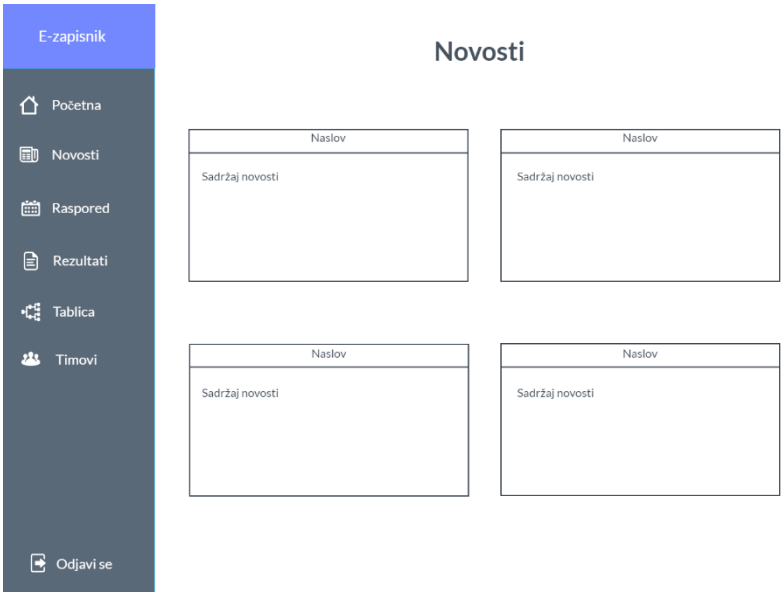
Slika 8. Registracija (Izrada studenta)

Nakon registracije sljedeći zaslon koji korisnik vidi je prijava, gdje mora unesti točno one podatke koje je unio pri registraciji. Ako kombinacija *e-maila* i lozinke ne odgovara dobit će odgovarajuću poruku da kombinacija ne odgovara te da pokuša ponovo.



Slika 9. Prijava (Izrada studenta)

Ako je kombinacija točna korisnik se prosljeđuje na sljedeći zaslon, a to su novosti. S lijeve strane je ponuđen izbornik sa svim funkcionalnostima koje nudi aplikacija te je kosnik slobodan birati koja ga funkcionalnost zanima. Na stranici s novostima nalaze se novosti s naslovom i sadržajem gdje kosnik može pročitati najnovije novosti o određenoj ligi.



Slika 10.Novosti (Izrada studenta)

Zaslon rasporeda utakmica pruža detaljan uvid u sve utakmice koje dolaze u narednom periodu. Za svaku utakmicu može se vidjeti gdje se igra, kada se igra, tj. datum i vrijeme te tko su domaćini, a tko gosti u određenoj utakmici.



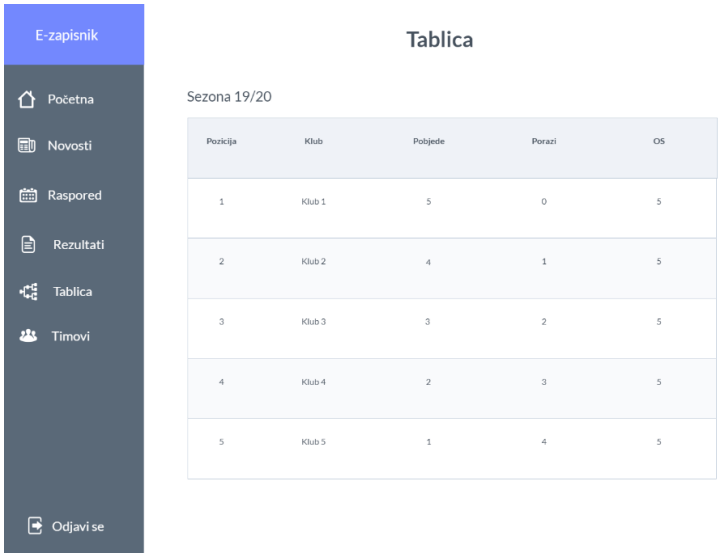
Slika 11. Raspored utakmica (Izrada studenta)

Naredni zaslon prikazuje detaljne rezultate odigranih utakmica gdje se vidi točan rezultat i pobjeda domaćina ili gostiju.



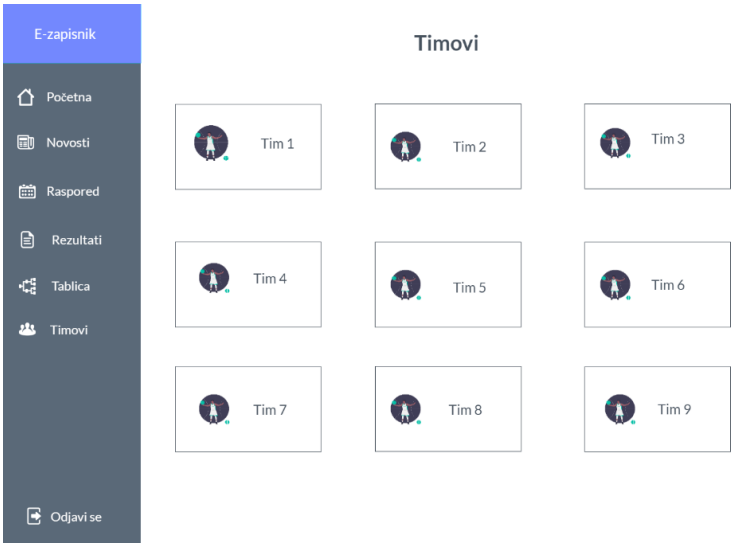
Slika 12. Rezultati (Izrada studenta)

Kod zaslona za pregledavanje tablice vide svi timovi te njihovi statistički atributi kao što su pobjede, porazi, odigrani susreti, bodovi, ukupan broj koševa, trenutna pozicija te status te pozicije (Sljedeća faza - Premijer liga (*Playoff*), Premijer liga (Relegacija - doigravanje) i Relegacija).



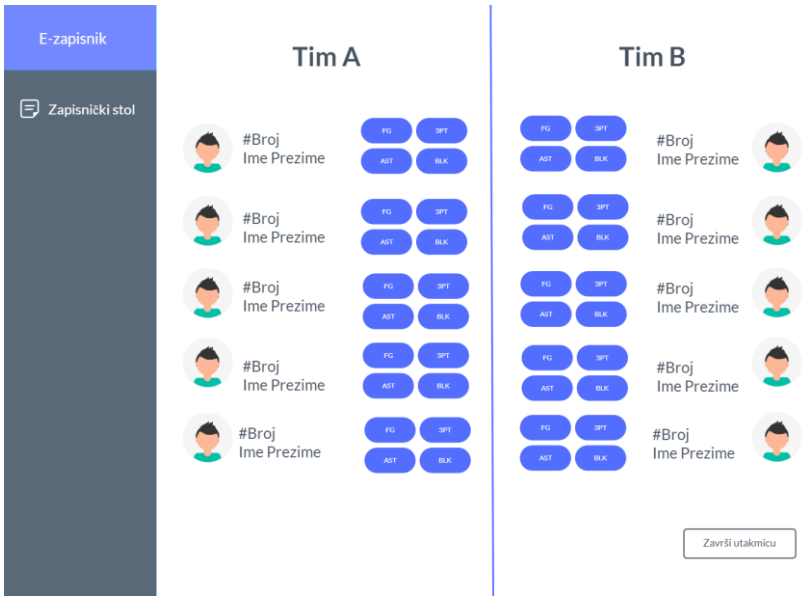
Slika 13. Tablica (Izrada studenta)

Zadnji zaslon koji korisnik ima na raspolaganju je pregledavanje timova određene košarkaške lige, gdje pišu detalje informacije o određenim igračima (ime, prezime, broj, starost, mjesto rođenja, visina, pozicija) u određenim timovima. Također za svaki tim se može vidjeti ime, lokacija, ime arene, ime trenera te ime predsjednika kluba.



Slika 14. Timovi (Izrada studenta)

Zaslon kojemu ima pristup samo zapisničar je elektronički zapisnički stol gdje zapisničar ima uvid u sve podatke kao i na papirnatom obrascu, a to su igrači i njihove statistike, poeni, osobni prekršaji, tehnički prekršaji, skokovi, ukradene lopte i blokade. S lijeve i desne strane zaslona prikažu se igrači određenog tima i njihovi brojevi na dresovima. Pokraj svakog igrača su gumbovi koji služe kako bi zapisničar dodavao svakom igraču statističke atribute kada je to potrebno. Zapisničar na zaslonu vidi domaću ekipu s lijeve strane i gostujuću ekipu s desne strane.



Slika 15. E-zapisnik (Izrada studenta)

7. Implementacija

U ovom dijelu rada detaljno će biti objašnjena i prikazana implementacija web aplikacije kroz nekoliko potpoglavlja, a to su *front end*, *back end* dijelovi aplikacije. Kroz funkcionalnosti (slike i kod) bit će prikazana svaka funkcionalnost koju aplikacija nudi. Korisnici su administrator, korisnik i zapisničar, a njihove funkcionalnosti bit će detaljno opisane u nastavku.

7.1. Front end

Kao što je navodno u prethodnom dijelu rada za *front end* korišten je *JavaScript* programski okvir otvorenog koda zvan *Vue.js* koji je pogodan za izgradnju korisničkih sučelja. Za *Vue.js* se smatra da ima neke od najboljih praksi koje nude konkurencija *React* i *Angular*. U odnosu na navedene okvire *Vue.js* je mnogo pristupačniji za početnike koji tek krenu stvarati moderne *front end* aplikacije. Instalacija je vrlo jednostavna preko *Vue CLI-a*. Potrebno je samo upisati nekoliko linija naredbi u terminal. Prvi korak je instalacija samog programskog okvira preko naredbe:

```
$ npm install -g @vue/cli@3.7.0
```

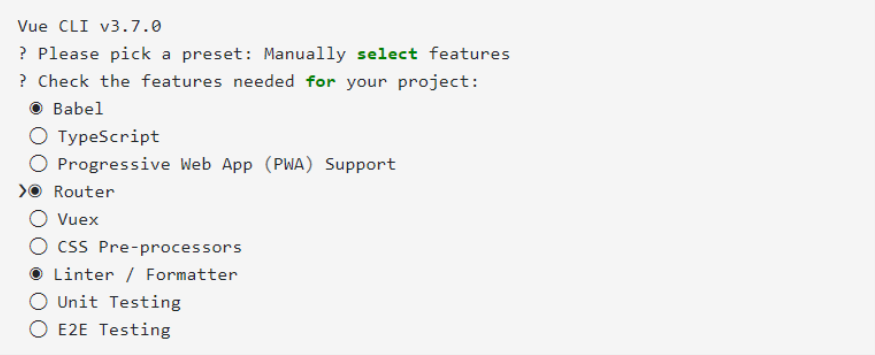
Slika 16. Instalacija Vue CLI-a (testdriven.io, 2019)

Zatim se kreira projekt:

```
$ vue create client
```

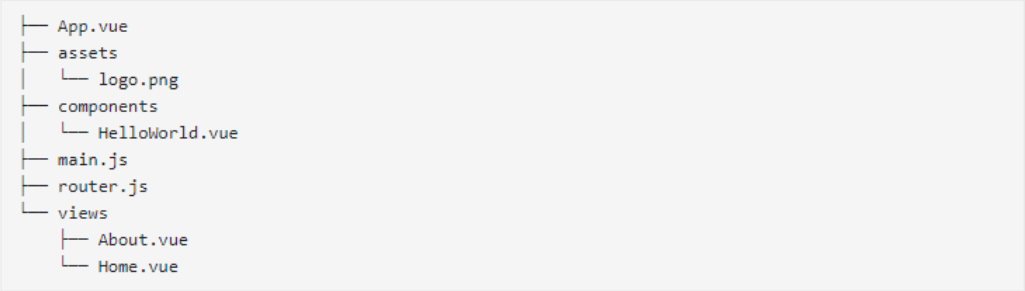
Slika 17. Kreiranje projekta (testdriven.io, 2019)

Nakon kreiranja projekta potrebno je odabrati strelicama na tipkovnici koje značajke želimo za naš projekt, a to su *Babel*, *Router* i *Linter*.



Slika 18. Instalacija Babela, Routera i Lintera (testdriven.io, 2019)

Na kraju bi kreirani projekt trebao imati strukturu kao na slici:



Slika 19. Struktura projekta (testdriven.io, 2019)

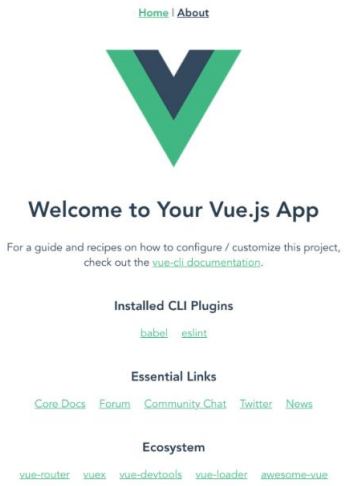
Main.js je ulazna točka aplikacije koja učitava i incijalizira *Vue* aplikaciju zajedno s *root* komponentom. *App.vue* je *root* komponenta iz koje se rade sve ostale komponente. Mapa *components* je mjesto gdje se pohranjuju sve komponente korisničkog sučelja. U *router.js* se definiraju svi *URL*-ovi komponenti. *Views* je mapa gdje se pohranjuju komponente koje su vezane za *router*. *Assets* je mjesto gdje se spremaju svi statički dokumenti kao što su slike i fontovi.

Prvi korak za pokretanje aplikacije je da se pozicioniramo u mapu gdje smo prvobitno kreirali aplikaciju, a to se radi preko terminala. Nakon pozicioniranja glavna naredba za pokretanje je *npm run serve.*

```
$ cd client
$ npm run serve
```

Slika 20. Pokretanje projekta (testdriven.io, 2019)

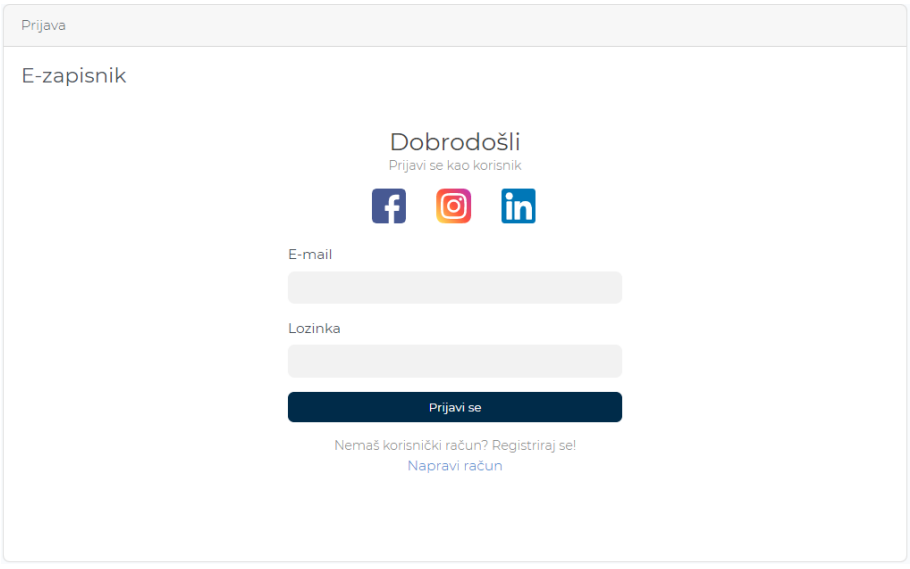
Nakon pokretanja aplikacije upisivanjem *http://localhost:8080* u web preglednik dobije se početna aplikacija kao na slici:



Slika 21. Početna aplikacija (Izrada studenta)

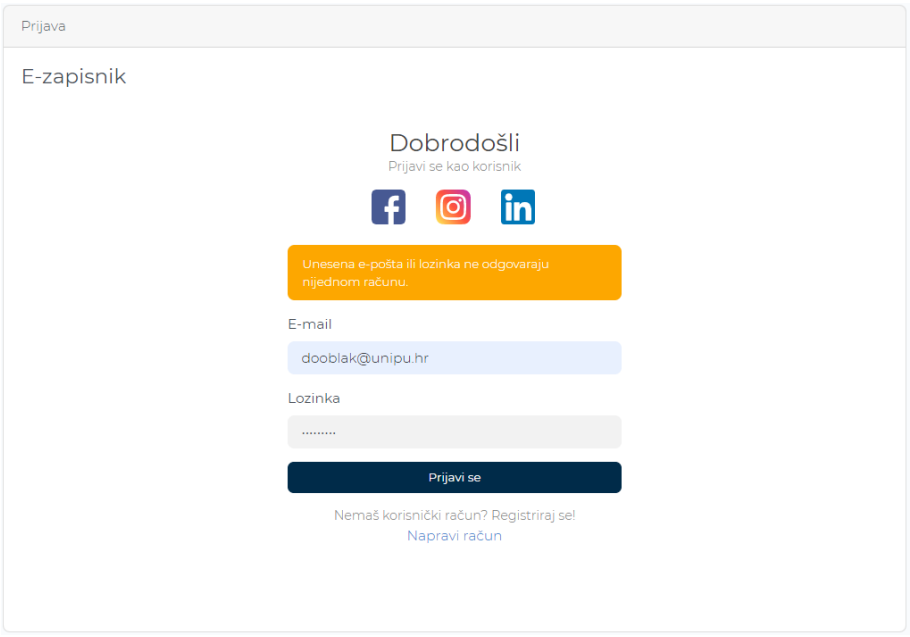
7.1.1. Prijava i registracija

Prva komponenta *front end* dijela aplikacije koja će biti prikazana je prijava. Prijava se sastoji od dva prazna polja gdje se upisuju *e-mail* i lozinka i gumb koji služi da provjeru točnosti upisanih podatka i prebacivanje u sljedeći dio aplikacije. Ispod gumba nalazi se link koji vodi do komponente za registraciju ako korisnik ne posjeduje korisnički račun potrebno ga je kreirati.



Slika 22. Prijava u sustav (Izrada studenta)

Ako korisnik upiše krivu kombinaciju *e-maila* i lozinke izbacuje mu se odgovarajuća poruka te je potrebno pokušati ponovno.



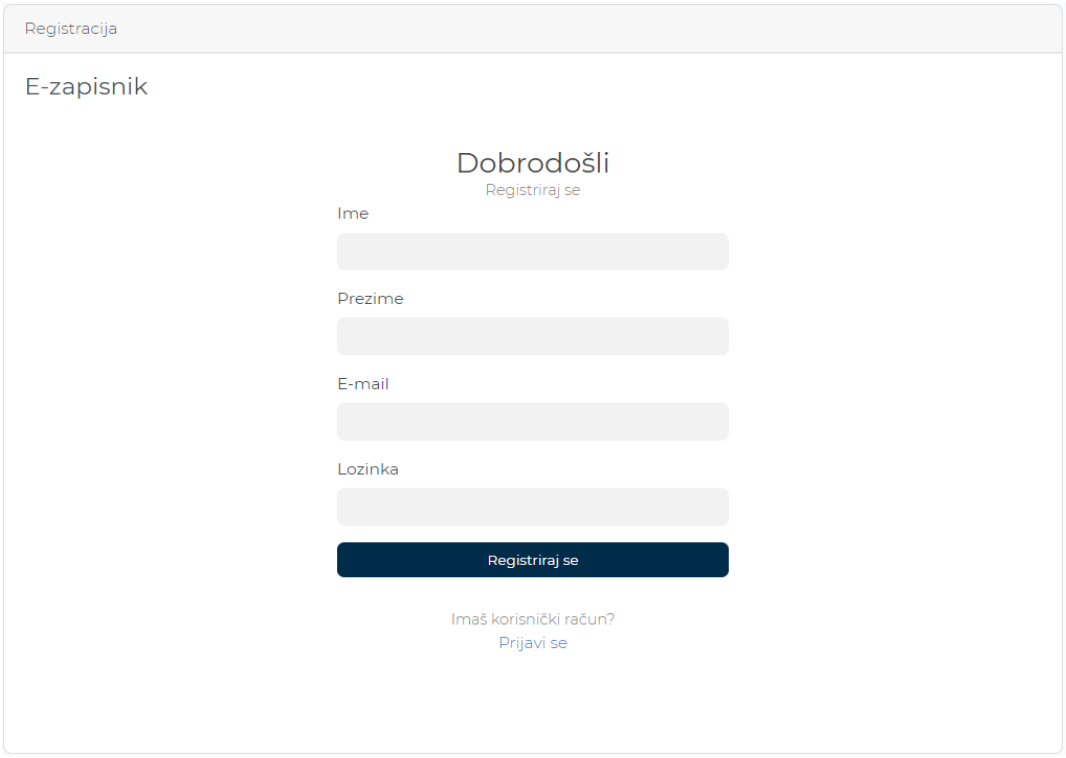
Slika 23. Prijava s neispravnim podacima (Izrada studenta)

Na slici 24 se vidi metoda *login* koja se poziva klikom na gumb prijavi se. Nakon klika poziva se *back end* ruta *http://localhost:8000/user/login* te se upisani podaci šalju na provjeru, ako podaci odgovaraju aplikacija prosljeđuje korisnika na komponentu *Posts* te postavlja korisnikove informacije u *localStorage,* a ako ne odgovaraju ispisuje se poruka „Unesena e-pošta ili lozinka ne odgovaraju nijednom računu“ kao što je vidljivo na slici 23.

```
login() {
  axios.post('http://localhost:8000/user/login', {
    email: this.email,
    password: this.password,
  }).then((res) => {
    if(res.status === 200){
    localStorage.setItem('usertoken', res.data);
    this.email = '';
    this.password = '';
    this.$router.push({ name: 'Posts' });
    }
  }).catch((err) => {
    this.errorMessage = 'Unesena e-pošta ili lozinka ne odgovaraju nijednom računu.';
    console.log(err);
  });
```

Slika 24. Funkcija za prijavu (Izrada studenta)

Komponenta za registraciju izgleda slično kao i komponenta za prijavu, sadrži više polja za unos podatka. Potrebno je u prazna polja upisati ime, prezime, *e-mail* i lozinku. Sva četiri polja su obavezna te nije moguće izvršiti registraciju ako je jedno od polja prazno. Također polje *e-mail* ima posebnu validaciju gdje pregledava da li je unesen ispravan format *e-maila* (npr. *example@mail.com*).



Slika 25. Registracija u sustav (Izrada studenta)

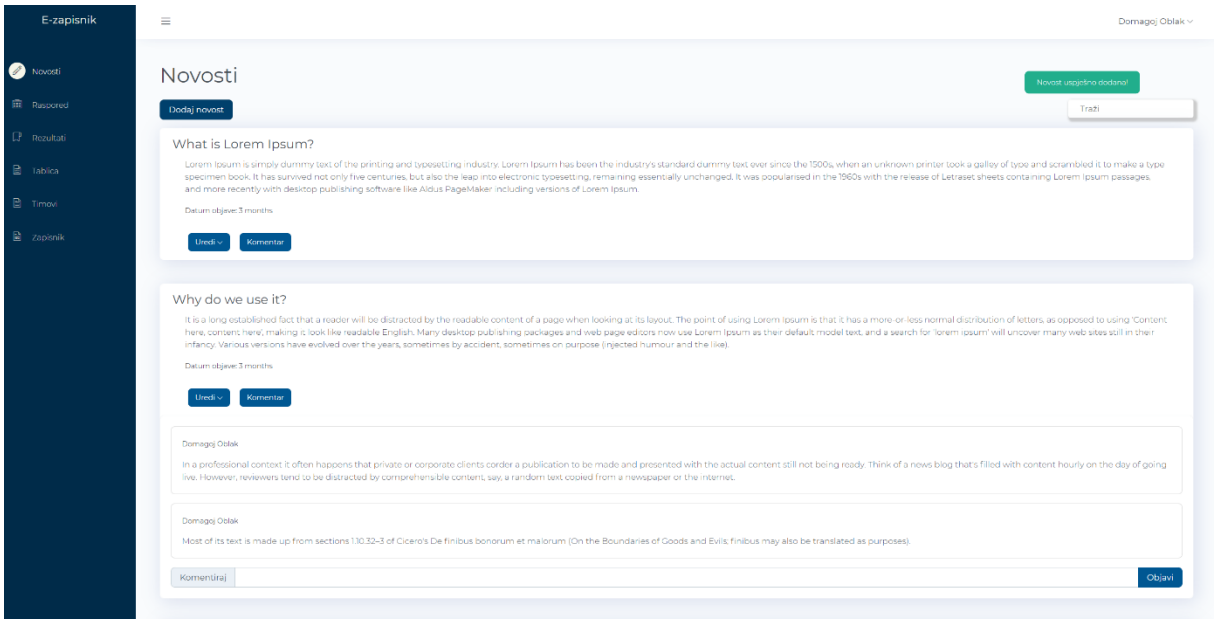
Metoda za registraciju poziva rutu *http://localhost:8000/user/register* u koju se šalju podaci upisani u prazna polja na zaslonu registracije. Varijable koje su inicijalizirane su označene s *this.ime varijable* gdje se spremaju vrijednosti iz inputa te se šalju na *back end*. Nakon toga se podaci spremaju u bazu podataka gdje se čuvaju podaci od svih registriranih korisnika. Ako je registracija uspješna korisnika se šalje na zaslon za prijavu u sustav.

```
register() {
  axios.post('http://localhost:8000/user/register', {
    fname: this.fname,
    lname: this.lname,
    email: this.email,
    password: this.password,
    is_admin: false,
  }).then((res) => {
    this.$router.push({ name: 'Login' });
    console.log(res);
  }).catch((err) => {
    console.log(err);
  });
},
```

Slika 26. Metoda za registraciju (Izrada studenta)

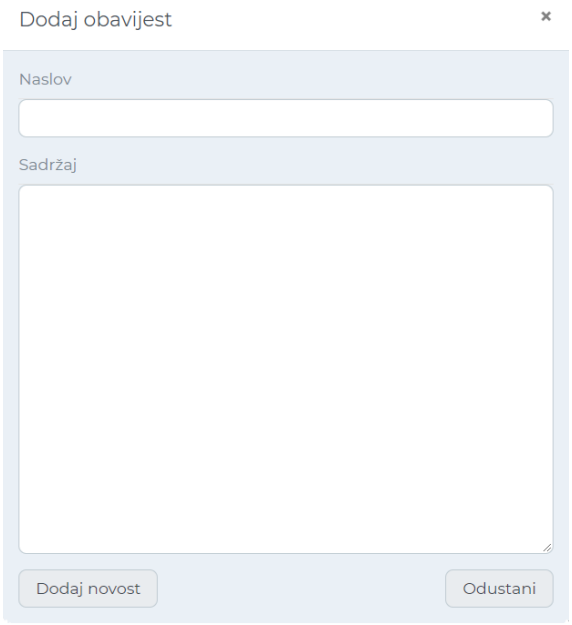
7.1.2. Novosti

Druga komponenta odnosi se na novosti koje objavljuje administrator, a korisnici imaju mogućnost čitanja i komentiranja. Navedena komponenta se sastoji od gumba za dodavanje novosti, tražilice koja filtrira novosti po naslovu te same novosti koja se sastoji od naslova i sadržaja. Svaku novost administrator može urediti tako da promijeni naslov i sadržaj ili da je obriše. Nakon dodavanja, brisanja ili uređivanja u gornjem desnom kutu korisnik dobije obavijest da je određena naredba izvršena.



Slika 27. Zaslon novosti (Izrada studenta)

Dodavanje novosti izvršava se preko <*b-modal*> elementa koji se klikom na gumb *dodaj novost* otvara kao zaseban element. Nudi mogućnost upisivanja naslova i sadržaja novosti preko *input-a* i *textfield-a*. Ako korisnik pokuša dodati novost, a prije toga nije ništa upisao u prazna polja za dodavanje novosti, korisnik dobiva obavijest da mora popuniti sva obavezna polja kod dodavanja novosti. To je omogućeno tako što je dodan parametar *required* u tag od *input*-a. Taj parametar osigurava da se u formi moraju ispunit sva navedena prije slanja na *back end* kako u bazi podataka nebi bilo praznih mjesta što bi kasnije stvaralo probleme pri izvlačenju podataka.



Slika 28. Prozor za dodavanje novosti (Izrada studenta)

Novosti se dohvaćaju u metodi *getPosts()* preko *back end* rute *http://localhost:8000/posts* te se spremaju u prazno polje *posts*. Zatim se <*template*> dijelu komponente kroz for petlju dohvaćaju sve novosti i ispisuju u željenom obliku. Istu stvar radi i metoda *getComments()* koja dohvaća sve komentare te ih sprema u polje nazvano *comments*, ali dohvaća one komentare koji su vezani za određeni *post* preko stranog ključa *post_id*. Objava komentara i kreiranje novosti radi na isti način tako da preko *back end* ruta dodaje upisane podatke iz *input-a* u bazu podataka. Brisanje je vrlo jednostavno i samo treba proslijediti *id* od novosti u *back end* rutu koju želimo obrisati. Uređivanje je malo zahtjevnija metoda gdje treba prvo u <*b-modal*> proslijediti postojeće podatke koje želimo urediti da se prikažu u *input-ima* kako bismo ih mogli urediti te ponovno poslati na bazu podataka gdje će biti novo uređeni podaci,

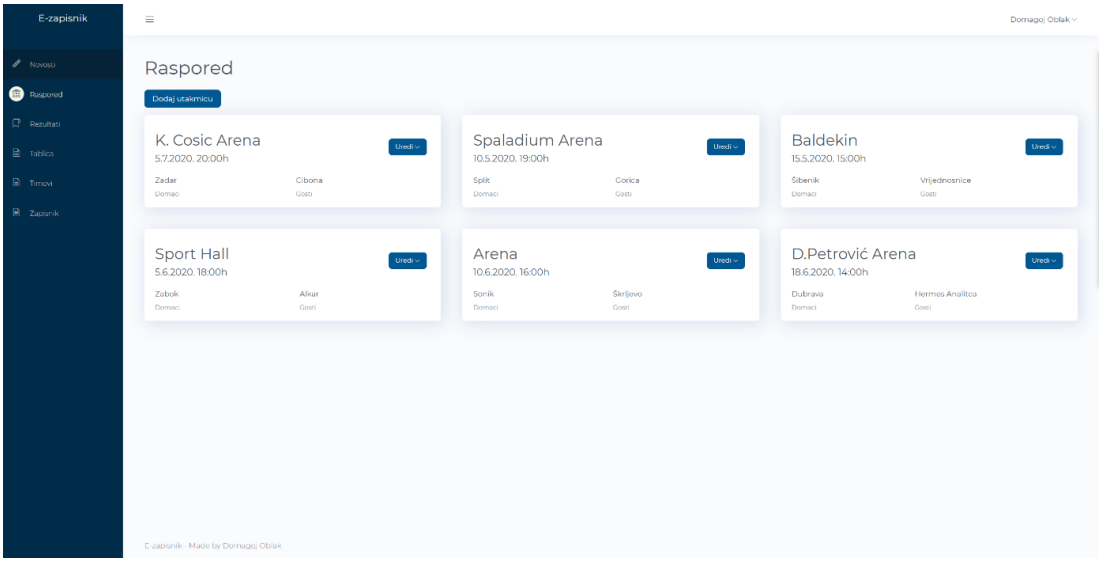
također je potreban *id* od posta kojeg želimo urediti kako bi uredili točno onu novost koju želimo. Metoda *created()* koja je predefinirana u programskom okviru *Vue.js* služi da se pri pokretanju aplikacije tj. ulaskom u komponentu *Novosti* automatski prikažu do sada objavljene novosti jer navedena metoda automatski poziva metodu *getPosts().*

```
getPosts() {
  axios.get('http://localhost:8000/posts').then((response) => {
    this.posts = response.data;
  })
},
```

Slika 29. Metoda za dohvaćanje novosti (Izrada studenta)

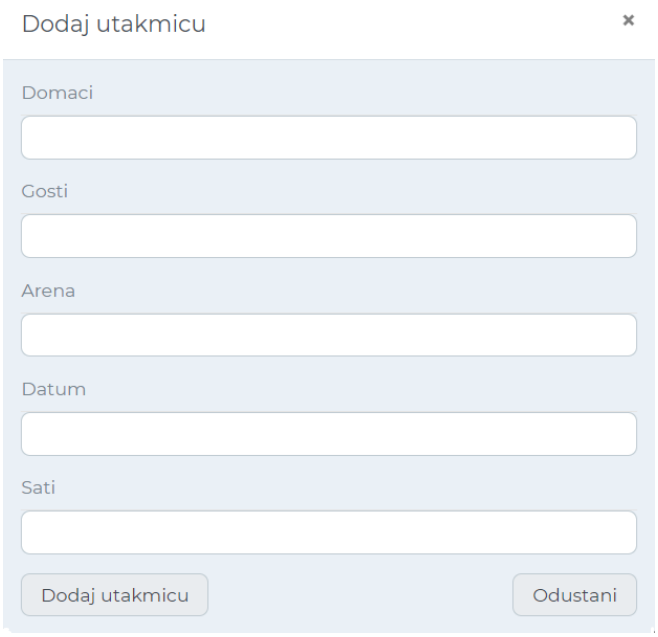
7.1.3. Raspored utakmica

Treća komponenta je raspored utakmica, a ona pruža detaljan uvid u sve utakmice koje dolaze u narednom periodu. Za svaku utakmicu može se vidjeti gdje se igra, kada se igra, tj. datum i vrijeme te tko su domaćini, a tko gosti u određenoj utakmici. Također se može vidjeti gumb za dodavanje nove utakmice koja poziva <*b-modal*> za upis novih podataka o utakmicama. Svaki element o određenoj utakmici ima gumb *Uredi* gdje se otvara padajući izbornik koju nudi dvije mogućnosti, a to su uređivanje i brisanje. Klikom na uređivanje poziva se <*b-modal*> koji se ispunjava podacima o odabranom dvoboju gdje se nudi mogućnost promjene određenih atributa. Klikom na gumb *Obriši* raspored će se jednostavno obrisati.



Slika 30. Zaslon rasporeda (Izrada studenta)

Dodavanje nove utakmice u raspored utakmica se vrši istom logikom kao i dodavanje novosti u prethodnom poglavlju. Klikom na gumb *Dodaj utakmicu* otvara se <*b-modal*> koji nudi prazna polja da se upišu domaći, gosti, datum, arena i vrijeme odigravanja utakmice. *Input-i* također koriste dodatni parametar *required* kako bi sva polja bila obavezna pri unosu novih podataka.



Slika 31. Prozor za dodavanje utakmica (Izrada studenta)

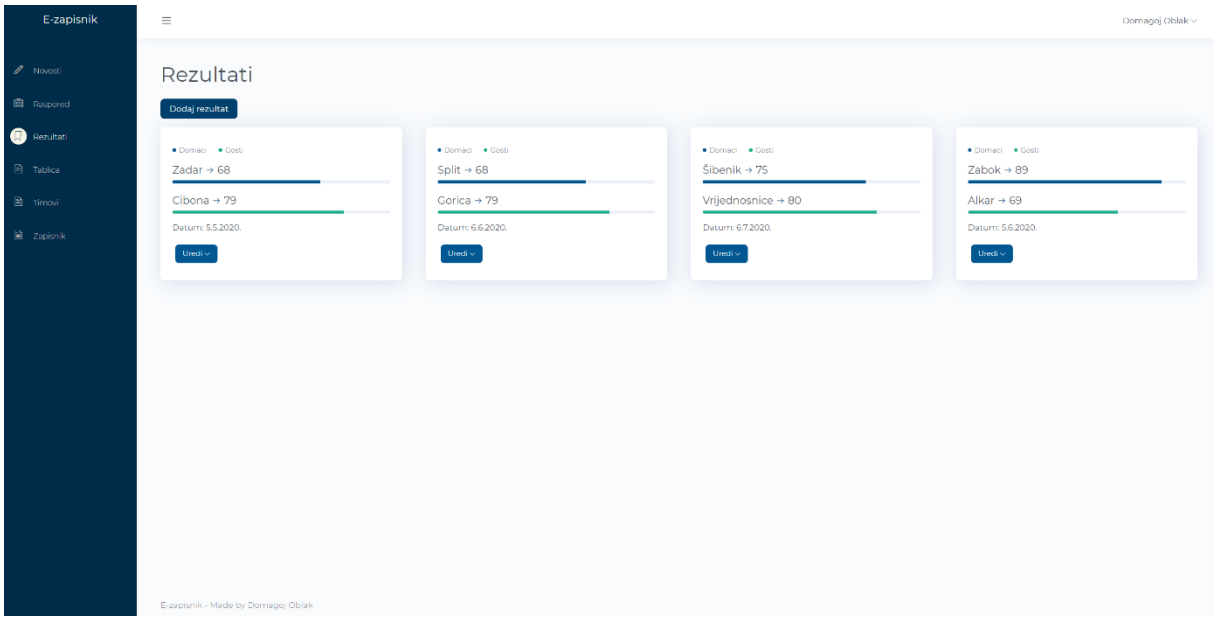
Raspored utakmica se ispisuju preko metode *getMatches()* koja radi na isti način kao metoda *getPosts()* u poglavlju 6.1.2. Metode za brisanje i uređivanje podataka također rade na isti način kao u prethodnom poglavlju te zahtijevaju *id* od utakmice u *back end* rutu koju želimo obrisati ili urediti. Znači metoda za uređivanje treba prvo u <*b-modal*> proslijediti postojeće podatke koje želimo urediti da se prikažu u *input-ima* kako bismo ih mogli urediti te ponovno poslati na bazu podataka gdje će biti novo uređeni podaci. Za brisanje podatka iz baze podataka potrebno je proslijediti *id* rezultata kojeg želimo obrisati te pozvati *back end* rutu u kojoj je definirano da se obriše taj podatak.

```
removeMatch(match_id) {
  const path = `http://localhost:8000/match/${match_id}`;
  axios.delete(path)
    .then(() => {
      this.getMatches();
    })
    .catch((error) => {
      console.error(error);
    });
},
```

Slika 32. Metoda za brisanje utakmice (Izrada studenta)

7.1.4. Rezultati

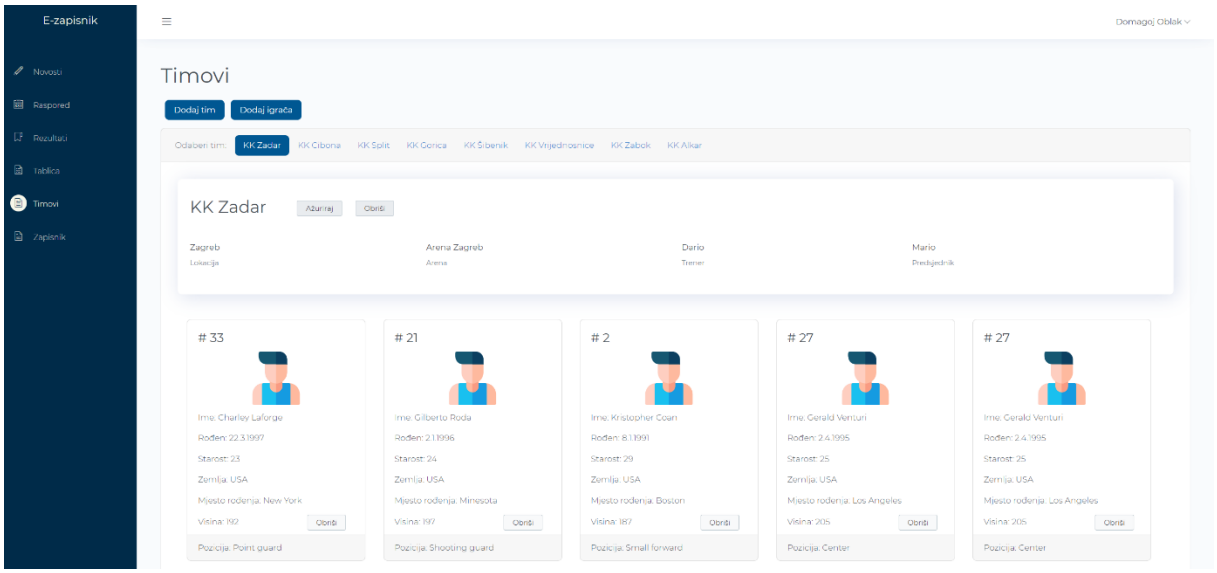
Četvrta komponenta su rezultati odigranih utakmica. Nakon svake odigrane utakmice administrator unosi podatke o rezultatima i pobjedniku gdje upisuje tko je igrao, kada, gdje i koliki je bio konačan rezultat. U slučaju korištena funkcionalnosti e-zapisnik nakon odigrane utakmice rezultat se automatski upisuje u komponentu *Rezultati* te je odmah vidljiv svima. Rezultat je slikovito prikazan preko *Progress bar-a* gdje se točno vidi koja je ekipa bila bolja i postigla više poena. Navedena komponenta se još sastoji i od gumba za dodavanje novih rezultata, legende koja opisuje tko je domaćin, a tko gostujuća ekipa te gumba za uređivanje postojećih podatka. Rezultati također imaju sve *CRUD* mogućnosti te rade na način kao što je objašnjeno u prethodna dva poglavlja te isti neće biti ovdje opisani. Kao i kod novosti i rasporeda za dodavanje novih rezultata potrebno je kliknuti na gumb *Dodaj rezultat* koji otvara *<b-modal>* s praznim poljima gdje je moguće unesti nove podatke o rezultatima. Navedeni element sadrži tag *<form>* unutar kojeg se stavljaju svi potrebni *input-i* za dodavanje novog rezultata. Svaki *input* sadrži svoje ime, id, v-model te parametar *required* što označava koje je polje obavezno pri unosu. Nakon svakog uspješnog unosa korisnik se obavještava notifikacijama da je unos bio uspješan.



Slika 33. Zaslon rezultata (Izrada studenta)

7.1.5. Timovi

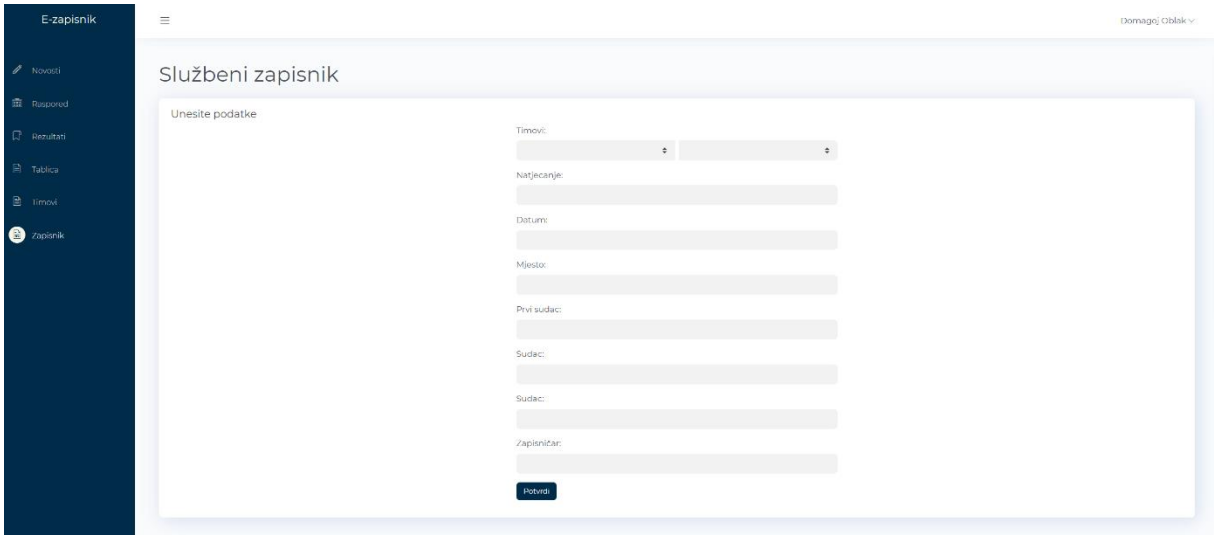
Predzadnja komponenta predstavlja klubove (timove) i njihove igrače. Svaki korisnik ima mogućnost pregledavanja timova određene košarkaške lige, gdje pišu detalje informacije o određenim igračima (ime, prezime, broj, starost, mjesto rođenja, visina, pozicija) u određenim timovima. Također za svaki tim se može vidjeti ime, lokacija, ime arene, ime trenera te ime predsjednika kluba. Komponenta sadrži nekoliko gumbova za dodavanje timova i za dodavanje igrača u određeni tim, kao i gumb za odabir tima kojeg se želi prikazati. Ispod tih gumbova nalazi se opis odabranog tima i gumbovi za ažuriranje i brisanje. Ispod svakog tima prikazani su igrači koji pripadaju timu te njihovi podaci koji su navedeni na početku ovo odlomka. Klikom na gumb *Dodaj tim* pruža se mogućnost da se unese novi tim kroz *back end* spremi u bazu. Istu mogućnost daje i gumb *Dodaj igrača* gdje se otvara <*b-modal*> s praznim poljima i jednim <*select*> u kojemu su ispisani svi mogući timovi koji se trenutno natječu u određenoj ligi. Svaki igrač se može detaljno pregledati ili obrisati ako više ne igra u određenom klubu. Klubovi su poredani u *tab-ove* i svakim klikom na ponuđeni klub poziva se preko *back end*-a baš taj klub s određenim *id-om.* Funkcionalnost koju pružaju *tab-ovi* su u ovom slučaju pogodni za ovakvu vrstu prikaza gdje ima velik broj timova, a svaki tim sadrži preko 10 igrača.



Slika 34. Zaslon timova (Izrada studenta)

7.1.6. Zapisnik

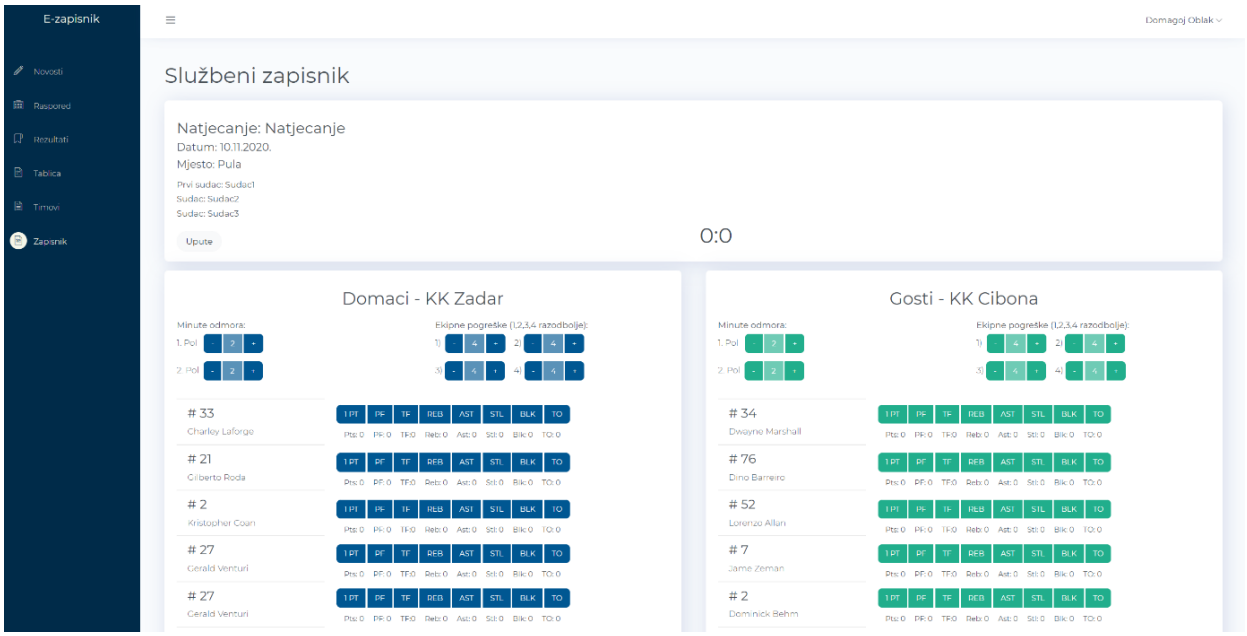
Zadnja i najbitnija komponenta je zapisnik u koji pristup samo zapisničar, a ne i obični korisnik aplikacije. Službeni zapisnik prvo zahtijeva da se u prazna polja unesu potrebni podaci, a to su timovi (domaći i gostujući), natjecanje tj. određena liga, datum, mjesto održavanja utakmice, prvi, drugi i treći sudac te ime i prezime samog zapisničara koji garantira da nema manipulacija i pogrešnog unosa podataka.



Slika 35. Zaslon službenog zapisnika (Izrada studenta)

Nakon unosa svih potrebnih podataka klikom na gumb *Potvrdi* zatvara se element za unos podataka i otvara se e-zapisnik s odabranim timovima i njihovim igračima.

Tako zapisničar dobiva na uvid sve igrače koji se nalaze na terenu za vrijeme utakmice te može brže upisati određene statističke atribute pojedenom igraču. Sa svake strane (domaće i gostujuće) nalaze se gumbovi za minute odmora i ekipne pogreške. Svaka ekipa ima mogućnost iskoristi 2 puta minute odmora po jednom poluvremenu, dok ekipne pogreške mogu iskoristi 4 puta po jednoj četvrtini. Minute odmora imaju gumbove + i - gdje se oduzimaju minute odmora ovisno o tome koliko ga je puta trener određene ekipe zvao u toku poluvremena s time da se broj za minute odmora ne može smanjiti manje od nule ili više od 2. Na isti način funkcioniraju gumbovi za smanjivanje i povećavanje ekipnih pogrešaka. Kada dođe do ekipne pogreške zapisničar će na gumb minus stisnuti da se smanji trenutno stanje s 4 na 3 ili ako je pogriješio ima mogućnost da klikom na plus vrati određenu vrijednost. Ispod gumbova za dodavanje i smanjivanje minuta odmora i ekipnih pogreški nalaze se ispisani igrači cijelog tima s lijeve i desne strane. Svaki igrač ima svoje gumbove na koje zapisničar stisne s obzirom na to što je navedeni igrač učinio (zabio koš, napravio prekršaj itd.). Klikom na određeni gumb dodaje se vrijednost za jedan, tako npr. ako igrač zabije koš za tri poena zapisničar mora tri puta kliknuti na gumb za dodavanje poena. Ako dođe do zabune pri upisivanju i zapisničar stisne na gumb za dodavanje poena, a htio je dodati osobni prekršaj duplim klikom na isti gumb koji je krivo stisnu vrijednost će se oduzeti za jedan i vratiti na staru ili početnu vrijednost. Mogućnosti koje zapisničar ima pri dodavanju vrijednosti su poeni, osobni prekršaji, tehnički prekršaji, skokovi, asistencije, ukradene lopte i blokade. Ako zapisničar slučajno klikne na gumb za osvježavanja dobit će obavijest od strane web preglednika da li je siguran da želi osvježiti navedenu web lokaciju ili da odustane kako nebi došlo do gubitka podatka.



Slika 36. Zaslon službenog zapisnika (Izrada studenta)

Kada je utakmica gotova zapisničar klikom na gumb završi utakmicu automatski dobiva *PDF* dokument o svim relevantnim podacima o utakmici koji se dalje mogu lako proslijediti ostalim distribuirati ostalim korisnicima u sustavu. U *PDF* dokumentu se nalaze timovi (domaći i gostujući), ime natjecanje tj. određene lige, datum, mjesto održavanja utakmice, prvi, drugi i treći sudac te ime i prezime samog zapisničara, kao i svi igrači i njihovi podaci o poenima, osobnim prekršajima, tehničkim prekršajima, skokovima, asistencijama, ukradenim loptama i blokadama što je vidljivo na slici 36.

Natjecanje: Natjecanje Datum: 10.11.2020. Mjesto: Pula

Ekipa A: KK Zadar

Broj	Prezime	Ime	Poeni	O.P.	T.P.	Skok.	Asist.	U.lopte	Blok.	Preok.
33	Laforge	Charley	0	0	0	0	0	0	0	0
21	Roda	Gilberto	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
27	Venturi	Gerald	0	0	0	0	0	0	0	0
27	Venturi	Gerald	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0
2	Coan	Kristopher	0	0	0	0	0	0	0	0

Ekipa B: KK Cibona

Broj	Prezime	Ime	Poeni	O.P.	T.P.	Skok.	Asist.	U.lopte	Blok.	Preok.
34	Marshall	Dwayne	0	0	0	0	0	0	0	0
76	Barreiro	Dino	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
7	Zeman	Jame	0	0	0	0	0	0	0	0
2	Behm	Dominick	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0
52	Allan	Lorenzo	0	0	0	0	0	0	0	0

Sudac: Sudac1 Sudac: Sudac2 Sudac: Sudac3

Zapisnicar: Zapisnicar Rezultat: 0:0

Slika 37. PDF dokument utakmice (Izrada studenta)

7.2. Back end

U ovom dijelu rada bit će objašnjen *back end* dio aplikacije gdje se obrađuju podaci. *HTTP* server je proces koji se odvija na računalu i radi dvije stvari:

- Osluškuje nadolazeće *HTTP* zahtjeve na određenoj *IP* adresi i *port-u*.
- Obrađuje primljene zahtjeve i šalje odgovor nazad klijentu.

Za navedeni dio aplikacije korišten je *Python* s programskim okvirom *Flask*. On nudi alate, knjižice i tehnologije koje omogućuju izradu web aplikacije. *Flask* spada u kategoriju mikro-okvira što označava okvir s malo ili nimalo zavisnosti o vanjskim knjižicama. Prednost je u tome što je lagan okvir, malo je zavisnih knjižnica koje je potrebno ažurirati i pratiti njihove sigurnosne greške. Nedostatak je da zahtjeva više programiranja ili je potrebno dodati neke druge dodatke. Ujedno i olakšava ponovnu upotrebu koda za uobičajene *HTTP* operacije kao što su *post*, *get*, *put*, *delete*. U osnovi, navedeni okvir obuhvaća rad koji su programeri naučili u posljednjih dvadeset godina dok su programirali aplikacije i web stranice. Prvi korak je instalacija programskog okvira s ekstenzijom *CORS* preko naredbe (e-ucenje.unipu.hr, 2020):

```
(env)$ pip install Flask==1.0.2 Flask-Cors==3.0.7
```

Slika 38. Instalacija Flaska (testdriven.io, 2019)

Da bi *cross-origin* zahtijevi koji potječu iz različitih protokola, *IP* adresa, domena ili porta potrebno je omogućiti *Cross Origin Resource Sharing (CORS),* a *Flask-Cors* je upravo za to. Drugi korak odnosi se na kreiranje *Flask* aplikacije. Kreira se tako da aplikacija mora stvoriti instancu objekta *Flask*, a web poslužitelj će proslijediti sve zahtjeve tom objektu. *__name__* je posebna *python* varijabla koja poprima vrijednost *"__main__"* kada se ta datoteka pokrene. Inače, poprima vrijednost imena modula.

```
from flask import Flask

app = Flask(__name__)
```

Slika 39. Kreiranje Flask aplikacije (testdriven.io, 2019)

Nakon instalacije potrebno je konfigurirati *app.py* kao na slici 39. Prvi korak je „*importanje*“ *flask-a, jsonfy-a i CORS-a,* nakon toga uključivanje *DEBUG-A.* Sljedeći

korak je instanciranje aplikacije naredbom *app = Flask(__Name__)* i uključivanje *CORS-a.* Zatim se upisuju određene rute koje su potrebne za aplikaciju i na kraju naredba za pokretanje same aplikacije.

```
from flask import Flask, jsonify
from flask_cors import CORS


# configuration
DEBUG = True

# instantiate the app
app = Flask(__name__)
app.config.from_object(__name__)

# enable CORS
CORS(app, resources={r'/*': {'origins': '*'}})


# sanity check route
@app.route('/ping', methods=['GET'])
def ping_pong():
    return jsonify('pong!')


if __name__ == '__main__':
    app.run()
```

Slika 40. Pokretanje Flask aplikacije (testdriven.io, 2019)

7.2.1. SQLAlchemy

Za spremanje podataka potrebna je baza podataka koja će čuvati podatke sigurnima i pružati mogućnost pristup podacima kada je to potrebno. Za rad s bazom podataka korišten je *Flask-SQLAlchemy* koji nudi mogućnost rada s *SQLAlchemy*. *SQLAlchemy* je *Python SQL* alat i *Object Relational Mapper* koji programerima daje punu snagu i fleksibilnost *SQL*-a. Pruža mogućnost za učinkovit i uspješan pristup bazi podataka te je prilagođen jednostavnom *Pythonic* načinu pisanja koda. *SQLAlchemy* uključuje podršku za *SQLite*, *Postgresql*, *MySQL*, *Oracle*, *MS-SQL*, *Firebird*, *Sybase* itd. Pri definiranju baze podataka potrebno je postaviti njezin *URI* u *app.config['SQLALCHEMY_DATABASE_URI']* (e-ucenje.unipu.hr, 2020).

Definiranje modela se odvija preko klase čije instance objekta želimo spremati moraju nasljeđivati *db.Model* klasu. Atribut *__tablename__* je definirani naziv tablice, ako nije određen ime tablice će dobit naziv po imenu klase pretvoren u mala slova. Nakon toga se definiraju atributi, odnosno kolone u bazi podataka. Uz naziv atributa definira se i tip koji može biti *db.String*, *db.Text*, *db.DateTime*, *db.Integer* s dodatnim parametrima (*unique*, *nullable*, *primary_key*, *default*) (e-ucenje.unipu.hr, 2020).

```
class MyUser(db.Model):

    __tablename__ = "my_user"

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)
    created_at = db.Column(db.DateTime(), default=datetime.now(),
nullable=False)
```

Slika 41. Definiranje modela (e-ucenje.unipu.hr, 2020)

Definiranje veza prema drugim tablicama se izvodi na poznat način definiranjem dodatne kolone koja će biti vanjski ključ, npr. *db.Column(db.Integer, db.ForeignKey('naziv_tablice.naziv_kolone')*. Ako je u pitanju *master-detail* veza u modelu je potrebno definirati dodatne atribute u koje će se u slučaju *master-a* spremati lista *detail-a*, a u slučaju *detail-a* referenca na objekt *master-a* (e-ucenje.unipu.hr, 2020).

```
class MyUser(db.Model):

    #...

    tasks = db.relationship('MyTask', back_populates='user', lazy="dynamic")

    #...

class MyTask(db.Model):

    #...

    user_id = db.Column(db.Integer, db.ForeignKey('my_user.id'), nullable=False)

    user = db.relationship('MyUser', back_populates='tasks', lazy=True)

    #...
```

Slika 42. Definiranje veza (e-ucenje.unipu.hr, 2020)

Uz sve navedeno mora se još obratiti pažnja i na parametre koji su *back_populates* i *lazy.* Parametar *back_populates* govori jednoj vezi o drugoj i obrnuto. Ukazuje vezama da međusobno trebaju ostvariti dvosmjernu komunikaciju. Što znači da će se prilikom dodavanja novog *detail-a* u mastera tom istom *detail-u* definirati i referenca prema tom *master* objektu. *Lazy* parametar koji određuje kako se povezani objekti učitavaju kada se podaci čitaju iz baze. Obično se, prilikom čitanja sa baze, zajedno učitaju i povezani podaci, ali se sa *lazy* parametrom može kontrolirati kako se oni učitavaju (e-ucenje.unipu.hr, 2020).

7.2.2. Rute komponente posts

Nakon instalacije i pripreme okružena za rad s *Flask-om* na red dolazi izrada metoda i ruta za rad s podacima i za obrađivanje zahtjeva . U cijeloj aplikaciji ima relativno velik broj metoda tj. grupa metoda za neki dio aplikacije, stoga će biti prikazana samo metode za manipuliranje s postovima u aplikaciji jer su sve ostale metode izrađene na sličan način. Metoda koja će obrađivati zahtjeve za dohvaćanje svih podataka unutar grupe za postove su trivijalne. Potrebno je samo kreirati rutu s metodom da dohvaća sve postove te da ih vrati u *JSON* formatu na određenom *URL*-u.

```
@app.route('/posts', methods=['GET'])
def get_posts():
    posts = Post.query.all()
    return jsonify(posts)
```

Druga ruta sadrži metodu za dohvaćanje postova prema parametru *id*.

```
@app.route('/post/<int:post_id>', methods=['GET'])
def get_post(post_id):
    post = Post.query.filter_by(id=post_id).first()
    return jsonify(post)
```

Spremanje novog posta odvija se preko metode *POST* gdje se podaci nalaze u *body-u*. Stoga je potrebno napraviti *commit* na bazu podataka kako bi se transakcija spremila u bazu. Postupak izrade rute je sličan kao u prethodnim slučajevima te se mora definirati parametar *id* kako bi se prema njemu moglo kasnije dohvatit, uredit ili obrisat određeni podatak.

```
@app.route('/post', methods=['POST'])
def add_post():
    data = request.get_json()
    new_post = Post(title=data["title"],
                    content=data["content"],
                    created_at=data["created_at"])
    db.session.add(new_post)
    db.session.commit()
    return jsonify(new_post)
```

Kod uređivanja posta potrebno je u metodi proslijediti *id* posta kojeg se želi urediti, te nakon toga ponovno spremiti uređene podatke u bazu podataka pomoću *commit-a.*

```
@app.route('/post/<int:post_id>', methods=['PUT'])
def update_post(post_id):
    data = request.get_json()
    post = Post.query.filter_by(id=post_id).first()

    post.title = data["title"]
    post.content = data["content"]
    post.created_at = data["created_at"]

    db.session.add(post)
    db.session.commit()
    return jsonify(post)
```

Zadnja *CRUD* operacija odnosi se na brisanje podataka. Pri brisanje nekog podatka potrebno je također proslijediti *id* podataka kojeg se briše kako bi se taj podatak pronašao u bazi te iz nje uklonio. Na kraju se kao odgovor vraća prazan rječnik što će označavati da je podatak uspješno izbrisan.

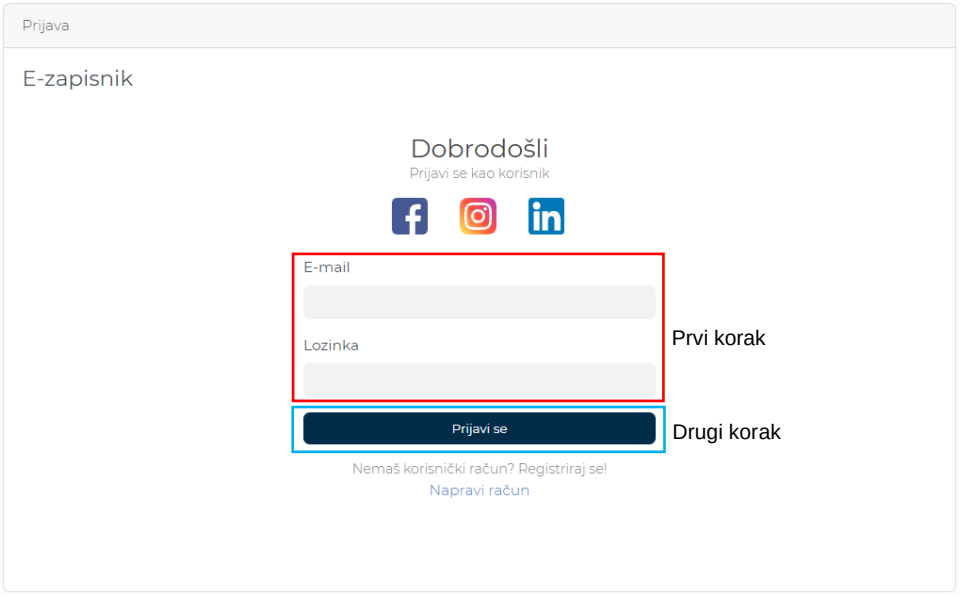
```
@app.route('/post/<int:post_id>', methods=['DELETE'])
def delete_post(post_id):
    post = Post.query.filter_by(id=post_id).first()
    db.session.delete(post)
    db.session.commit()
    return jsonify(post)
```

8. Korisničke upute

U ovom dijelu rada detaljno će biti objašnjene korisničke upute te na koji način se koristi aplikacija pomoću *screenshot-ova* sa stvarnog sučelja. Korisničke upute su namijenjene korisnicima aplikacije E-zapisnik kao pomoć pri korištenju aplikacije. Unutar korisničkih uputa bit će pojašnjeno na koji način administrator, korisnik i zapisničar mogu koristiti funkcionalnosti aplikacije.

8.1. Prijava i registracija

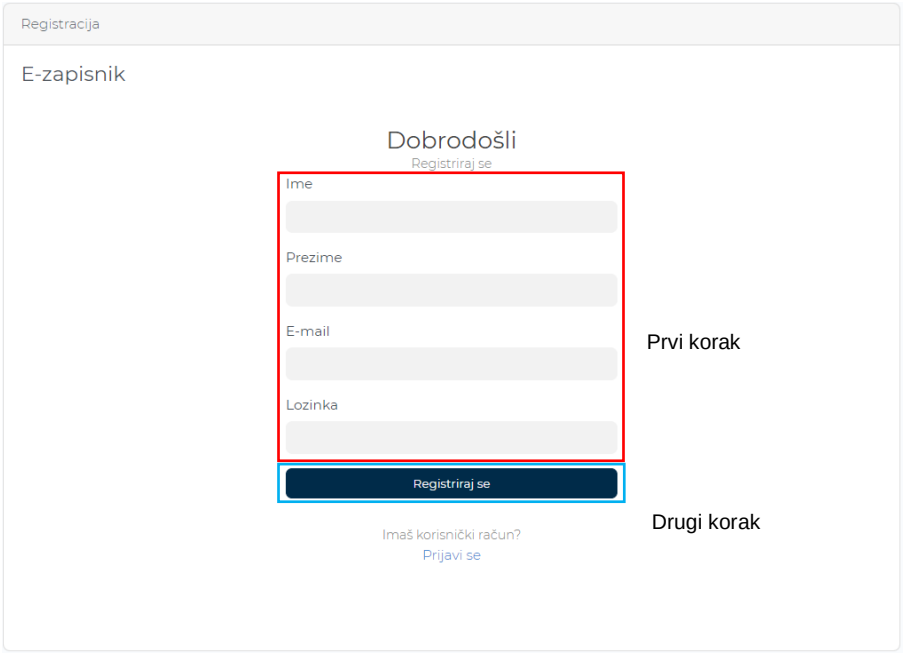
Ulaskom u aplikaciju otvara se zaslon za prijavu korisnika, gdje korisnik ima mogućnost upisati svoje podatke kako bi se prijavio u sustav. Prijava se izvršava tako da upiše svoju e-mail i lozinku u za to predviđeno mjesto koje je označeno na slici 43. Nakon upisivanja podataka klikom na gumb „Prijavi se“ aplikacija će preusmjeriti korisnika u sustav.



Slika 43. Prijava u sustav (Izrada studenta)

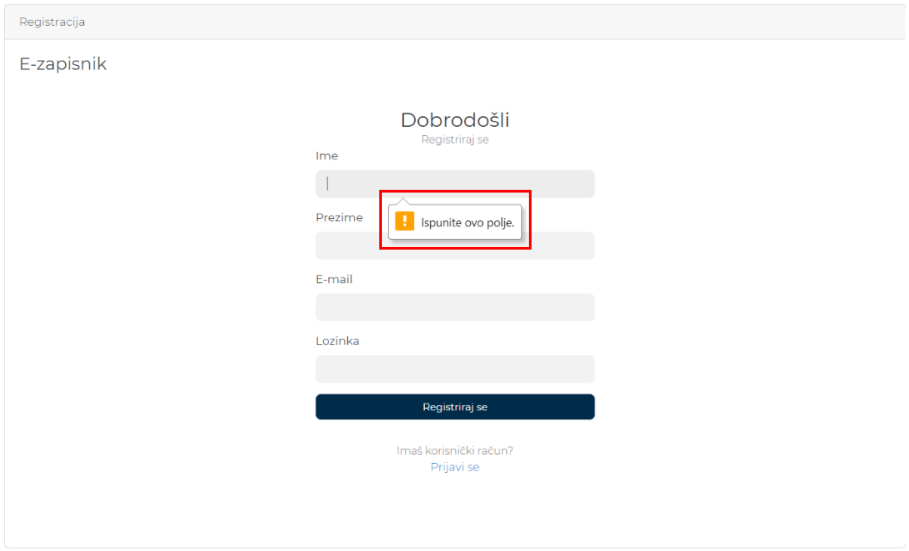
Na početku korisnik neće posjedovati korisnički račun, stoga ga prije prijave mora napraviti, a to može učiniti klikom na poveznicu „Napravi račun“ koja se nalazi na početnom zaslonu ispod gumba za prijavu. Nakon toga otvara se zaslon s praznim poljima ime, prezime, e-mail i lozinka, što se može vidjeti na slici 44. Poslije otvaranja zaslona korisnik ispunjava navedena polja svojim podacima koje će kasnije koristiti za prijavu u sustav. Pri unosu podataka korisnik mora ispuniti sva polja koja su tražena

kao i točan format e-mail (npr. *name@example.com*). Ako su sva polja ispunjena i format e-mail ispravan korisnik klikom na gumb „Registriraj se“ završava registraciju te ga aplikacija preusmjerava na prijavu u sustav.



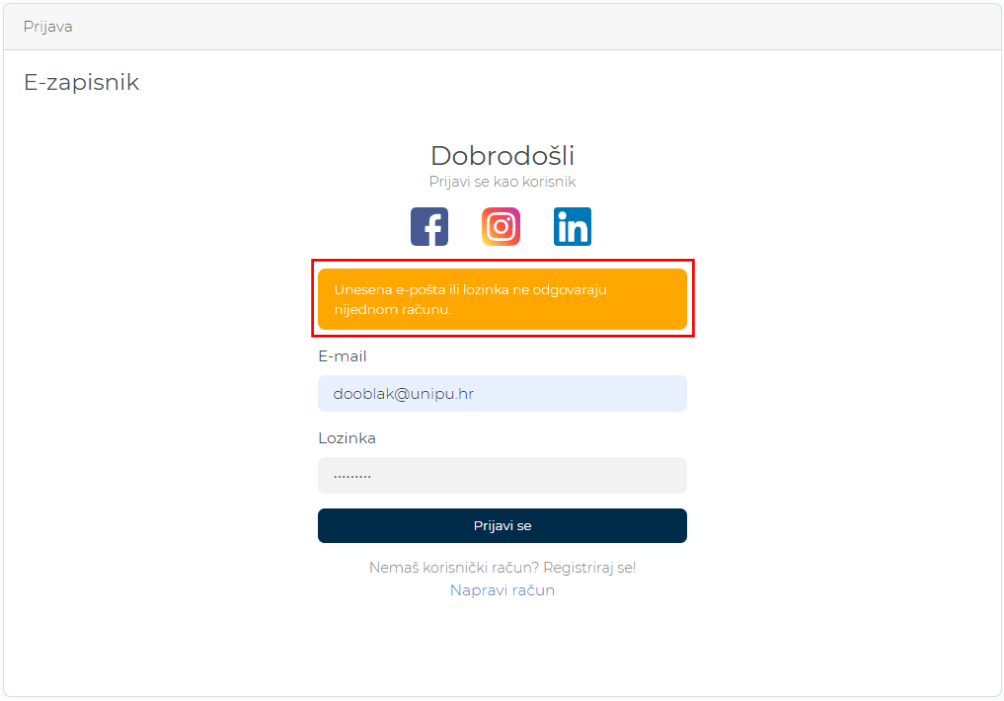
Slika 44. Registracija u sustav (Izrada studenta)

Ako se određeni podatak ne unese, a klikne se na gumb „Registriraj se“, izbacit će se upozorenje „Ispunite ovo polje“ jer su sva polja kod registracija i prijave obavezna. Kod prijave također klikom na gumb „Prijavi se“ postoji ista validacija koja provjerava jesu li uneseni podaci ili su polja prazna.



Slika 45. Obavezno ispunjavanje polja (Izrada studenta)

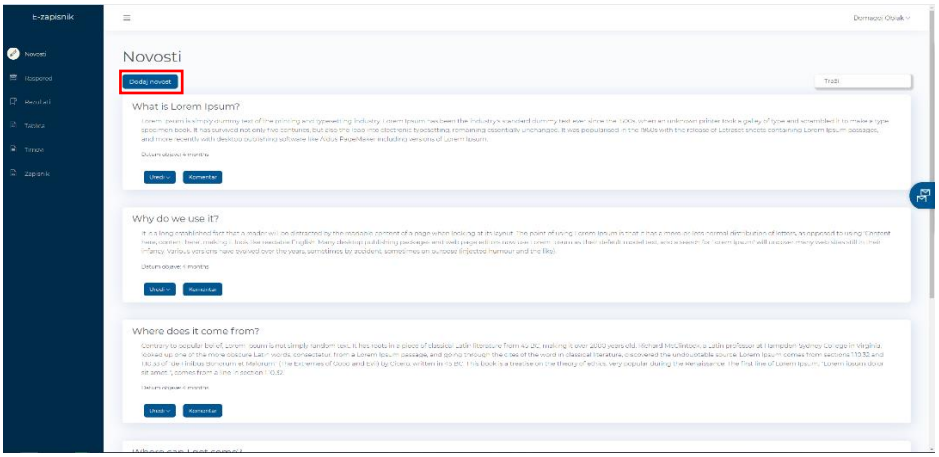
Upisom neispravne e-mail adrese ili lozinke prikazat će se obavijest s porukom „Unesena e-pošta ili lozinka ne odgovaraju nijednom računu“ i sve dok korisnik ne upiše ispravnu kombinaciju podataka za prijavu neće moći pristupiti aplikaciji.



Slika 46. Pograšna prijava (Izrada studenta)

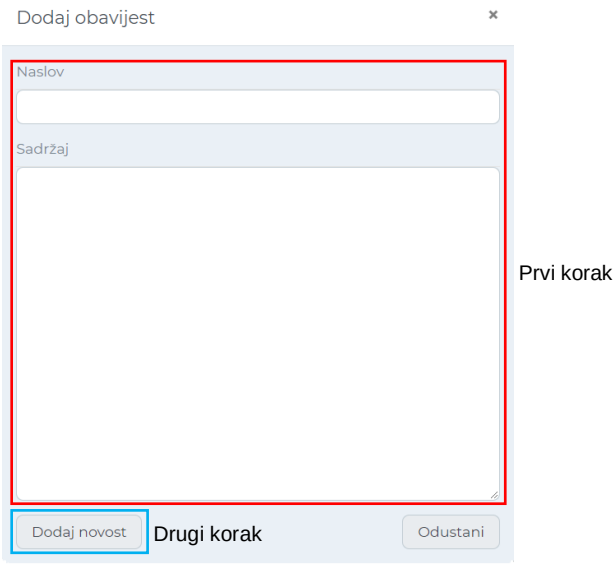
8.2. Novosti, raspored i rezultati

Nakon uspješne registracije i prijave u sustav otvara se zaslon s novostima gdje su vidljive sve do sada objavljene novosti. Administrator ima mogućnost dodavanja novosti klikom na gumb „Dodaj novost“ koji se nalazi u gornjem lijevom kutu kraj izbornika što je vidljivo na slici 47.



Slika 47. Zaslon s novostima (Izrada studenta)

Nakon klika na navedeni gumb otvara se prozor s praznim poljima koja nude upis novih podatka tj. unos novosti. Polja naslov i sadržaj su obavezna te nije moguće klikom na gumb „Dodaj novost“ dodati novosti ako je polje prazno jer izbacit će se upozorenje „Ispunite ovo polje“ . Klikom na navedeni gumb prozor za dodavanje novosti se zatvara i novost se pojavljuje na zaslonu gdje se nalaze sve novosti.



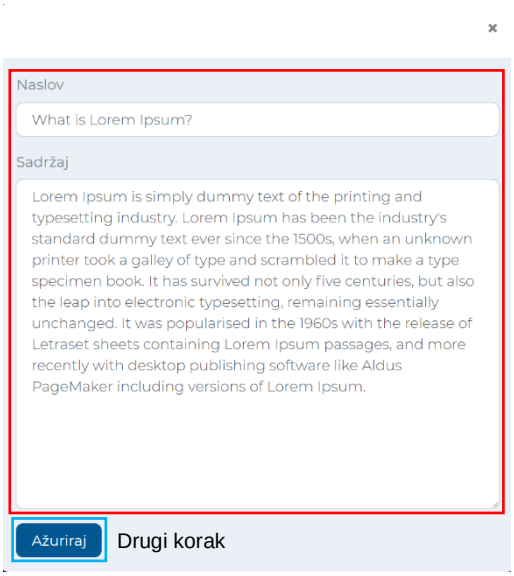
Slika 48. Dodavanje novosti (Izrada studenta)

Klikom na gumb „Uredi“ koji se nalazi ispod svake novosti, nudi se mogućnost uređivanja određene novosti ili brisanje iste preko padajućeg izbornika. Ako se administrator odluči na uređivanje već unesenih podataka potrebno je kliknuti na gumb „Ažuriraj“ ili ako želi obrisati određenu novost potrebno je kliknuti na gumb „Obriši“. Pri odabiru za brisanje novosti, novost će se obrisati iz baze podataka trajno te će također nestati s popisa novosti na aplikaciji.



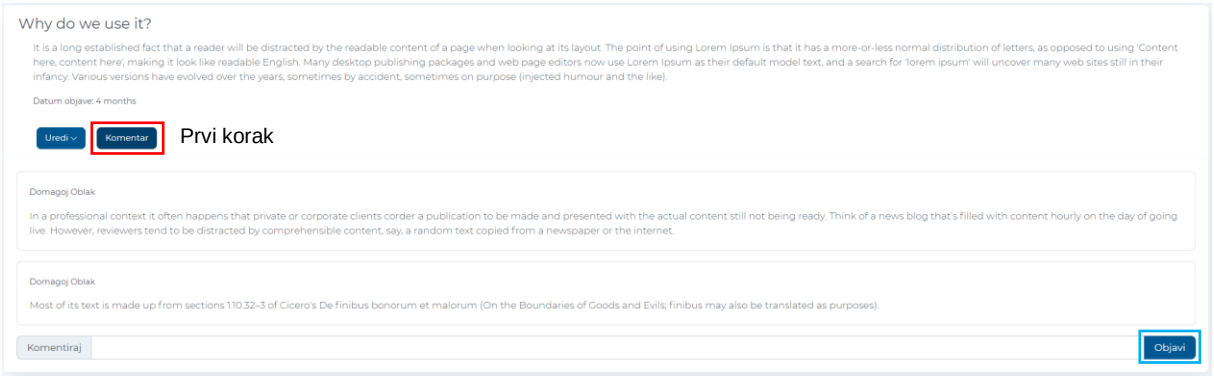
Slika 49. Prikaz novosti (Izrada studenta)

Ako je odabir uređivanje novosti otvara se prozor s poljima gdje su upisani podaci za odabranu novost kako bi se oni mogli izmijeniti. Nakon uređivanja klikom na gumb „Ažuriraj“ zatvara se prozor za uređivanje podataka te se novi podaci spremaju u bazu podataka i ispisuju na zaslonu za novosti. Ako administrator ipak ne želi uređivati podatke, a već je kliknuo na gumb „Ažuriraj“ može zatvoriti prozor klikom na gumb „X“ koji se nalazi u gornjem desnom kutu ili klikom na bilo koji dio zaslona će također zatvoriti navedeni prozor za uređivanje podataka, a podaci će ostati isti i neće se slati na bazu.



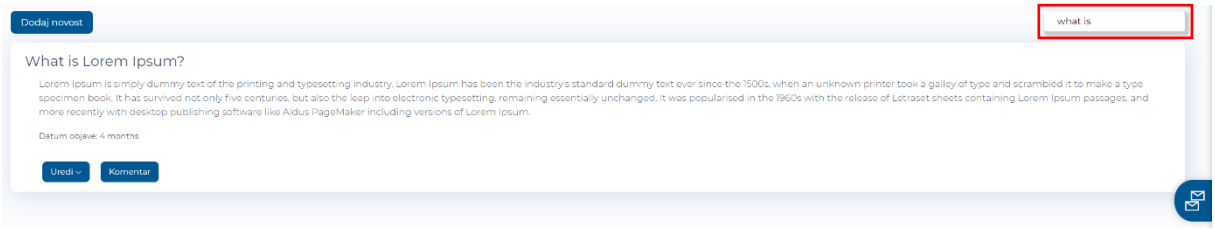
Slika 50. Uređivanje novosti (Izrada studenta)

Uz sve prethodno navedeno korisnik ima mogućnost komentiranja pojedine novosti. Klikom na gumb „Komentar“ otvara se odlomak s komentarima na kojem su vidljivi svi komentari vezani za određenu novost, kao i forma za dodavanje novog komentara. Ako je polje za dodavanje komentara prazno, a korisnik klikne da objavi komentar izbacit će se upozorenje kao i kod ostalih polja u aplikaciji koja moraju biti ispunjena. Ako se korisnik odluči objaviti komentar, komentar će biti vidljiv svima te će kraj njega stajati njegovo ime.



Slika 51. Komentiranje novosti (Izrada studenta)

Zadnja funkcionalnost koja se nudi korisniku na zaslonu za novosti je pretraživanje novosti po naslovu. U gornjem desnom kutu zaslona nalazi se tražilica za upis tj. pretraživanje novosti po naslovu, korisnik ima mogućnost pretražiti novost upisom naslova po želi te nakon upisa će biti prikazana samo novost koja je tražena. Ako određeni pojam ne postoji u bazi podataka, neće biti prikazana ni jedna novost.

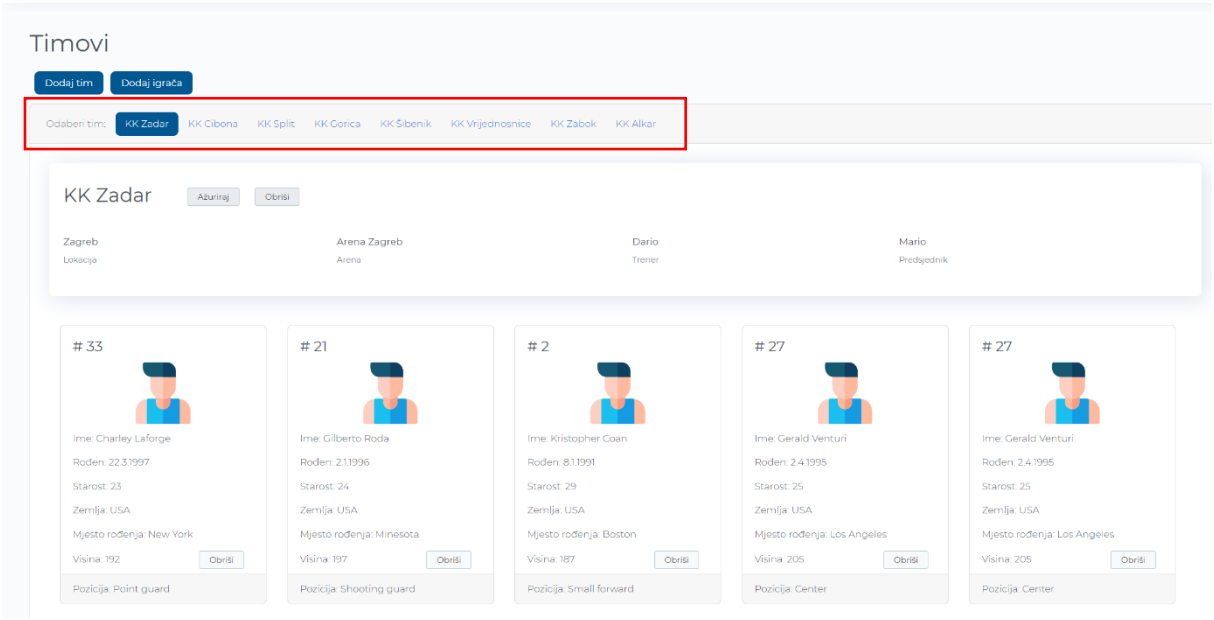


Slika 52. Tražilica (Izrada studenta)

Zasloni raspored i rezultati su napravljeni po istoj logici što znači da je korištenje tih zaslona identično navedenom zaslonu za novosti, stoga za korištenje ta dva zaslona treba pratiti upute koje su opisane u ovom poglavlju.

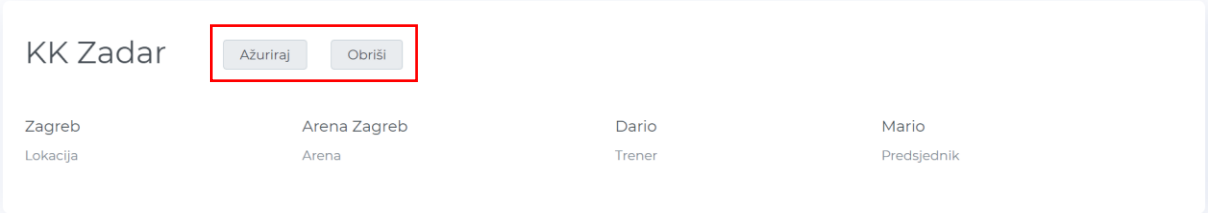
8.3. Timovi

Uz mogućnosti kao što su pregled novosti, rasporeda i rezultata korisnik može pregledavati i timove. Ova komponenta predstavlja klubove (timove) i njihove igrače. Svaki korisnik ima mogućnost pregledavanja timova određene košarkaške lige, gdje pišu detalje informacije o određenim igračima (ime, prezime, broj, starost, mjesto rođenja, visina, pozicija) u određenim timovima. Također za svaki tim se može vidjeti ime, lokacija, ime arene, ime trenera te ime predsjednika kluba. Pregled određenog tima počinje tako da korisnik odabere neki od ponuđenih timova na predloženom izborniku, što se može vidjeti na slici 53.



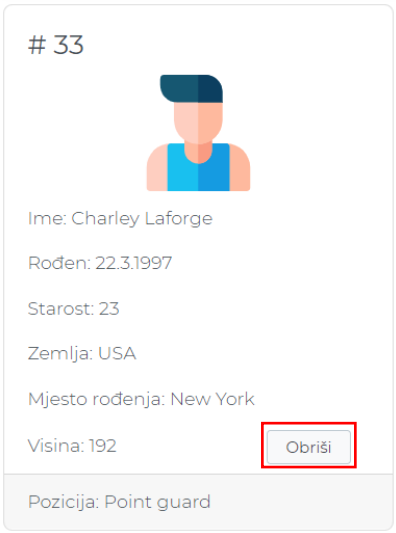
Slika 53. Pregled timova (Izrada studenta)

Klikom na gumb „Ažuriraj“ koji se nalazi pokraj svakog naziva tima, nudi se mogućnost uređivanja određenog tima. Nakon klika na navedeni gumb otvara se prozor s podacima i koracima kao u prethodnom poglavlju, koje korisnik treba pratiti kako bi izvršio naredbu, što je vidljivo na slici 50. Ako se administrator odluči na uređivanje već unesenih podataka potrebno je kliknuti na gumb „Ažuriraj“ ili ako želi obrisati određenu novost potrebno je kliknuti na gumb „Obriši“. Pri odabiru za brisanje novosti, novost će se obrisati iz baze podataka trajno te će također nestati s popisa novosti na aplikaciji. Ako se želi obrisati tim koji sadrži igrača/će to neće biti moguće te će sustav izbaciti upozorenje u gornjem desnom kutu aplikacije „Ne možete obrisati tim ako sadrži igraće“.



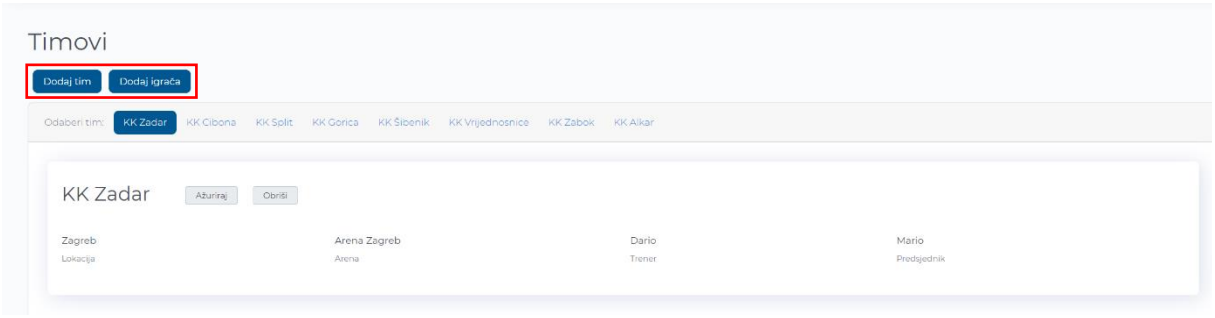
Slika 54. Ažuriranje i brisanje timova (Izrada studenta)

Administrator također ima mogućnost brisanja određenog igraća iz tima klikom na gumb „Obriši“ koji se nalazi ispod atributa o pojedinom igraču.



Slika 55. Brisanje igrača (Izrada studenta)

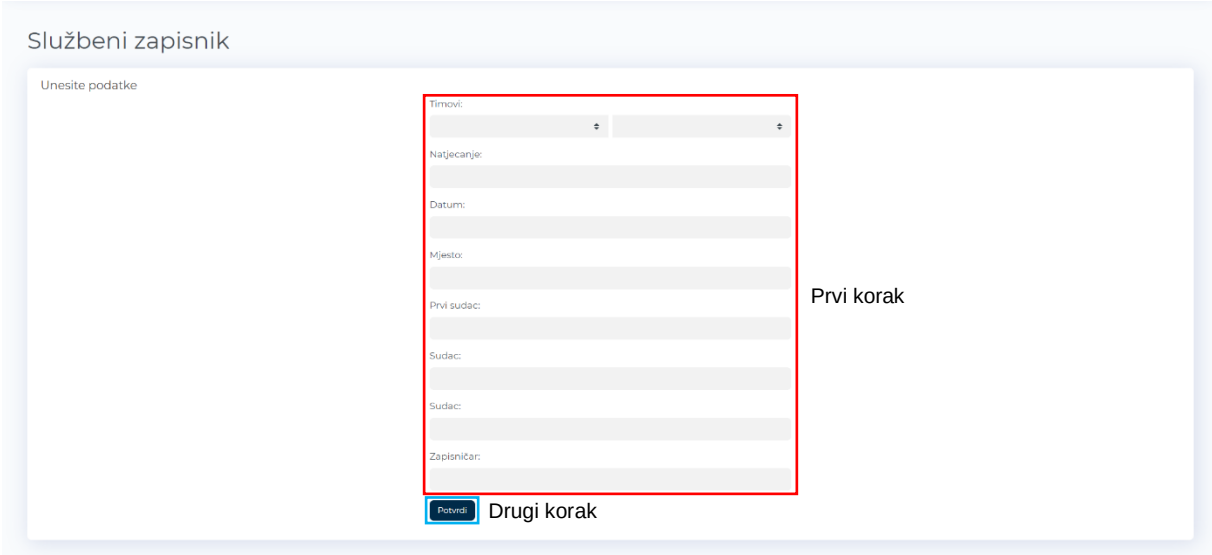
Zadnje dvije funkcionalnosti koje su na izboru administratoru su dodavanje timova i igrača. Nakon klika na jedan od dva gumba otvara se prozor s praznim poljima koji nudi unos novih podataka u vezi tima ili pojedinog igrača. Prozor izgleda kao na slici 48.



Slika 56. Dodavanje timova i igrača (Izrada studenta)

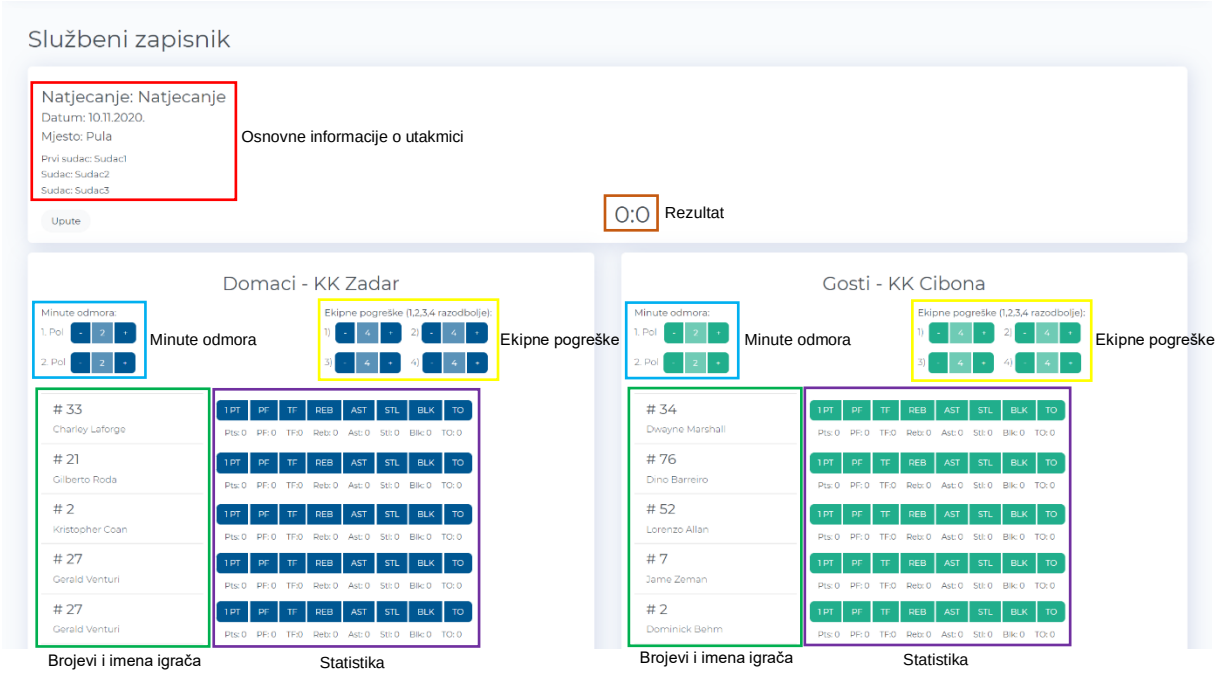
8.4. Službeni zapisnik

Jedna od zadnjih komponenti koja će biti objašnjena u korisničkim uputama je službeni zapisnik. Navedenu komponentu može koristiti samo zapisničar koji ima korisnički račun za vođenje zapisnika. Nakon unosa svih potrebnih podataka klikom na gumb „Potvrdi“ zatvara se element za unos podataka i otvara se e-zapisnik s odabranim timovima i njihovim igračima.



Slika 57. Unos podataka za zapisnik (Izrada studenta)

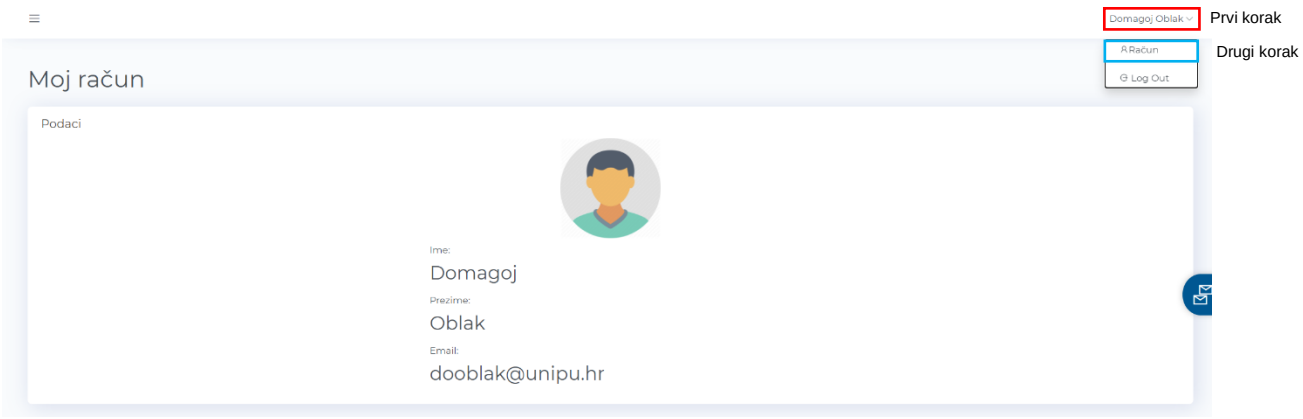
Tako zapisničar dobiva na uvid sve igrače koji se nalaze na terenu za vrijeme utakmice te može brže upisati određene statističke atribute pojedenom igraču. Sa svake strane (domaće i gostujuće) nalaze se gumbovi za minute odmora i ekipne pogreške. Svaka ekipa ima mogućnost iskoristi 2 puta minute odmora po jednom poluvremenu, dok ekipne pogreške mogu iskoristi 4 puta po jednoj četvrtini. Minute odmora imaju gumbove + i - gdje se oduzimaju minute odmora ovisno o tome koliko ga je puta trener određene ekipe zvao u toku poluvremena s time da se broj za minute odmora ne može smanjiti manje od nule ili više od 2. Na isti način funkcioniraju gumbovi za smanjivanje i povećavanje ekipnih pogrešaka. Kada dođe do ekipne pogreške zapisničar će na gumb minus stisnuti da se smanji trenutno stanje s 4 na 3 ili ako je pogriješio ima mogućnost da klikom na plus vrati određenu vrijednost. Ispod gumbova za dodavanje i smanjivanje minuta odmora i ekipnih pogreški nalaze se ispisani igrači cijelog tima s lijeve i desne strane. Svaki igrač ima svoje gumbove na koje zapisničar stisne s obzirom na to što je navedeni igrač učinio (zabio koš, napravio prekršaj itd.). Klikom na određeni gumb dodaje se vrijednost za jedan, tako npr. ako igrač zabije koš za tri poena zapisničar mora tri puta kliknuti na gumb za dodavanje poena. Ako dođe do zabune pri upisivanju i zapisničar stisne na gumb za dodavanje poena, a htio je dodati osobni prekršaj duplim klikom na isti gumb koji je krivo stisnu vrijednost će se oduzeti za jedan i vratiti na staru ili početnu vrijednost. Mogućnosti koje zapisničar ima pri dodavanju vrijednosti su poeni, osobni prekršaji, tehnički prekršaji, skokovi, asistencije, ukradene lopte i blokade.



Slika 58. Vođenje zapisnika (Izrada studenta)

8.5. Profil

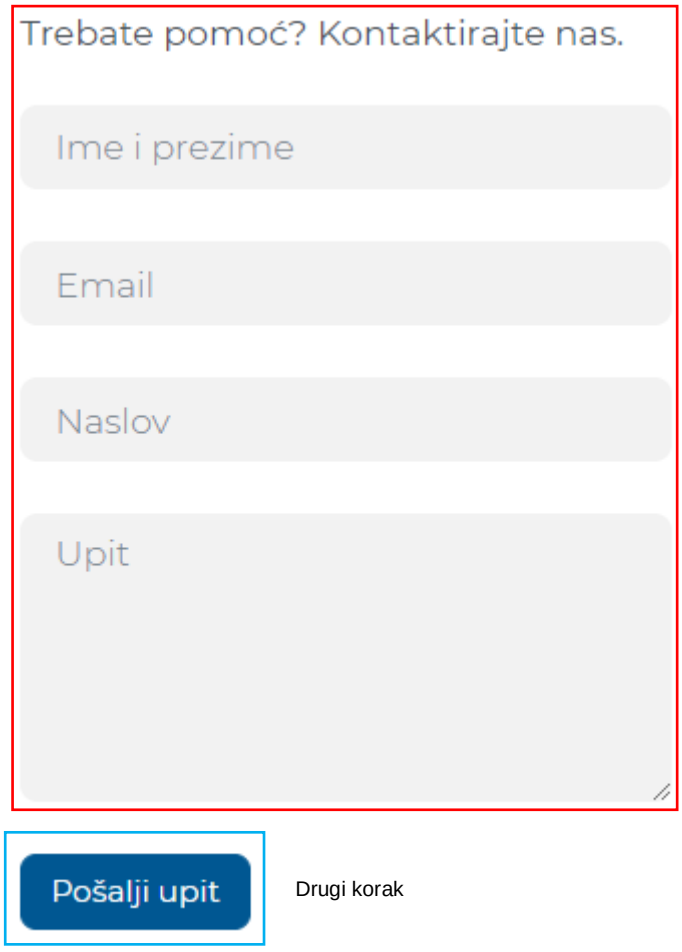
Svaki korisnik klikom na svoje ime i prezime u gornjem desnom kutu ima mogućnost da pregleda svoje podatka koje je koristio pri izradi korisničkog računa. Nakon klika na ime otvara se padajući izbornik koji nudi dvije mogućnosti, a to su „Račun“ i „Odjava“. Klikom na račun otvara se korisnički račun s podacima kao što je navedeno gore.



Slika 59. Profil korisnika (Izrada studenta)

8.6. Pomoć

Zadnja komponenta u korisničkim uputama odnosi se na pomoć korisniku tj. forma za kontaktiranje. Ako korisnik ima nekih problema u korištenju aplikacije treba ispuniti formu koja se otvara klikom na dvije plave ikone na desnoj strani aplikacije. Klikom na navedene ikone otvara se forma s praznim poljima koje korisnik treba ispuniti kako bi točno opisao problem koji treba riješiti. Nakon što je korisnik ispunio formu klikom na „Pošalji upit“ šalje se upit na e-mail s navedenim problemom te će on biti riješen u što kraćem roku.



Slika 60. Slanje upita (Izrada studenta)

9. Zaključak

Pri implementaciji web aplikacije nudi se mnoštvo programskih okvira i jezika koji nude puno mogućnosti koje uvelike pomažu u izgradnji određenog sustava. Iz analize se zaključuje da je neophodno analizirati pojedine programske okvire i jezike kako bi se odabrao najpogodniji koji odgovara određenom problemu. Zato su neki programski jezici i okviri pogodni samo za manje *SPA (single page application).*

Cilj ovog rada je opisati izradu navedene web aplikacije kroz implementaciju *front end-a* i *back end-a*, kao i principe rada pojedinog programskog jezika ili okvira te ukazati na prednosti i nedostatke takve vrste implementacije.

Glavna prednost web aplikacija je da ne zauzimaju prostor na korisnikovom računalu, već se cijela aplikacija nalazi na poslužitelju. Također nadogradnja se odvija na poslužitelju, tako korisnik ne mora izvršavati nikakve procedure kako bi nadogradio aplikaciju na naviju inačicu. Uz sve navedeno i štede vrijeme jer nema potrebe za instaliranjem aplikacije ili kasnije brisanjem jer je dovoljan samo web preglednik i internetska veza kako bi se pristupilo određenoj aplikaciji.

Nedostatci su također prisutni kod ovakve vrste aplikacija jer ovisi o brzini korisnikove internetske veze i dostupnosti poslužitelja kada je to potrebno. Možda i najveći problem kada se priča o web aplikacijama je sigurnost na internetu tj. opasnost od krađe podataka, neželjenih upada i virusa.

API (application programming interface) je dio softvera koji priključuje jednu aplikaciju s podacima i servisima druge aplikacije tako što pruža pristup određenim dijelovima poslužitelja. *API*-ji omogućavaju komunikaciju dva dijela softvera što olakšava dijeljenje podataka između njih. *REST* (*Representational State Transfer)* se temelji na *URL*-u i *HTTP* protokolima. Za razmjenu podataka uglavnom se koristi *JSON* format koji je kompatibilan s web preglednicima odnosno *JavaScript*-om.

Pri implementacija E-zapisnik aplikacije korišteni su programski jezici *JavaScript*, *Python*. Za *JavaScript* korišten je programski okvir *Vue.js* gdje je još potrebno poznavati osnovne funkcionalnosti *HTML*-a i *CSS*-a kako bi se napravila cjelokupna aplikacija, programski okvir *BootstrapVue* je korišten je u zamjenu za *CSS*. Uz sve navedeno za *back end* dio aplikacije korišten je programski okvir *Flask* koji olakšava ponovnu upotrebu koda za uobičajene *HTTP* operacije kao što su *post*, *get*, *put* i

delete. *SQLite* baza podatak je korištena zbog razlike od svih ostalih jezika *SQL*-a jer koristi sustav dinamičkog tipa, što znači da vrijednost pohranjena u stupcu određuje njegov tip podataka, a ne vrstu podataka stupca.

Korisničke upute su namijenjene korisnicima aplikacije E-zapisnik kao pomoć pri korištenju aplikacije. Unutar korisničkih uputa pojašnjeno je na koji način administrator, korisnik i zapisničar mogu koristiti funkcionalnosti aplikacije.

Poveznica na *Github* repozitorij je https://github.com/dooblak/e-zapisnik-application.

Literatura

1. Bootstrap-vue (2020), dostupno na: https://bootstrap-vue.org
2. E-ucenje UNIPU (2020), dostupno na:https://eucenje.unipu.hr/course/view.php?id=5000
3. Filipova, O.: „Learning Vue.js 2“, Packt Publishing Ltd., Birmingham, 2016.
4. Flask (2020), dostupno na: https://flask.palletsprojects.com/
5. Hostinger (2019), dostupno na: https://www.hostinger.com/tutorials/what-is-html
6. InfoWorld (2018), dostupno na: https://www.infoworld.com/article/3204016/what-is-python-powerful-intuitive-programming.html
7. JavaScript.info (2020), dostupno na: https://javascript.info/intro
8. Skillcrush (2019), dostupno na: https://skillcrush.com/blog/css/
9. Tutorialspoint (2017), dostupno na: https://www.tutorialspoint.com/python/python_overview.htm
10. Testdriven.io (2019), dostupno na: https://testdriven.io/blog/developing-a-single-page-app-with-flask-and-vuejs/
11. Vuejs (2020), dostupno na: https://vuejs.org/

Popis slika

Slika 1. GET metoda (e-ucenje.unipu.hr) 3
Slika 2. POST metoda (e-ucenje.unipu.hr) 3
Slika 3. PUT metoda (e-ucenje.unipu.hr) 4
Slika 4. DELETE metoda (e-ucenje.unipu.hr) 4
Slika 5. Use Case dijagram (Izrada studenta) 11
Slika 6. Class dijagram (Izrada studenta) 12
Slika 7. SWOT analiza (Izrada studenta) 13
Slika 8. Registracija (Izrada studenta) 14
Slika 9. Prijava (Izrada studenta) 15
Slika 10.Novosti (Izrada studenta) 15
Slika 11. Raspored utakmica (Izrada studenta) 16
Slika 12. Rezultati (Izrada studenta) 16
Slika 13. Tablica (Izrada studenta) 17
Slika 14. Timovi (Izrada studenta) 17
Slika 15. E-zapisnik (Izrada studenta) 18
Slika 16. Instalacija Vue CLI-a (testdriven.io, 2019) 19
Slika 17. Kreiranje projekta (testdriven.io, 2019) 19
Slika 18. Instalacija Babela, Routera i Lintera (testdriven.io, 2019) 20
Slika 19. Struktura projekta (testdriven.io, 2019) 20
Slika 20. Pokretanje projekta (testdriven.io, 2019) 20
Slika 21. Početna aplikacija (Izrada studenta) 21
Slika 22. Prijava u sustav (Izrada studenta) 21
Slika 23. Prijava s neispravnim podacima (Izrada studenta) 22
Slika 24. Funkcija za prijavu (Izrada studenta) 22
Slika 25. Registracija u sustav (Izrada studenta) 23
Slika 26. Metoda za registraciju (Izrada studenta) 24
Slika 27. Zaslon novosti (Izrada studenta) 24
Slika 28. Prozor za dodavanje novosti (Izrada studenta) 25
Slika 29. Metoda za dohvaćanje novosti (Izrada studenta) 26
Slika 30. Zaslon rasporeda (Izrada studenta) 26

Slika 31. Prozor za dodavanje utakmica (Izrada studenta) 27
Slika 32. Metoda za brisanje utakmice (Izrada studenta) 28
Slika 33. Zaslon rezultata (Izrada studenta) 29
Slika 34. Zaslon timova (Izrada studenta) 30
Slika 35. Zaslon službenog zapisnika (Izrada studenta) 30
Slika 36. Zaslon službenog zapisnika (Izrada studenta) 32
Slika 37. PDF dokument utakmice (Izrada studenta) 33
Slika 38. Instalacija Flaska (testdriven.io, 2019) 34
Slika 39. Kreiranje Flask aplikacije (testdriven.io, 2019) 34
Slika 40. Pokretanje Flask aplikacije (testdriven.io, 2019) 35
Slika 41. Definiranje modela (e-ucenje.unipu.hr, 2020) 36
Slika 42. Definiranje veza (e-ucenje.unipu.hr, 2020) 36
Slika 43. Prijava u sustav (Izrada studenta) 39
Slika 44. Registracija u sustav (Izrada studenta) 40
Slika 45. Obavezno ispunjavanje polja (Izrada studenta) 40
Slika 46. Pograšna prijava (Izrada studenta) 41
Slika 47. Zaslon s novostima (Izrada studenta) 41
Slika 48. Dodavanje novosti (Izrada studenta) 42
Slika 49. Prikaz novosti (Izrada studenta) 42
Slika 50. Uređivanje novosti (Izrada studenta) 43
Slika 51. Komentiranje novosti (Izrada studenta) 44
Slika 52. Tražilica (Izrada studenta) 44
Slika 53. Pregled timova (Izrada studenta) 45
Slika 54. Ažuriranje i brisanje timova (Izrada studenta) 45
Slika 55. Brisanje igrača (Izrada studenta) 46
Slika 56. Dodavanje timova i igrača (Izrada studenta) 46
Slika 57. Unos podataka za zapisnik (Izrada studenta) 47
Slika 58. Vođenje zapisnika (Izrada studenta) 48
Slika 59. Profil korisnika (Izrada studenta) 48
Slika 60. Slanje upita (Izrada studenta) 49