Web aplikacija za organiziranje turnira u košarci korištenjem višekriterijskog raspoređivanja

Barbić, Roko

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:701096

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-07**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij

WEB APLIKACIJA ZA ORGANIZIRANJE TURNIRA U KOŠARCI KORIŠTENJEM VIŠEKRITERIJSKOG RASPOREĐIVANJA

Završni rad

Roko Barbić

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 13.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Roko Barbić
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4316, 22.07.2019.
OIB Pristupnika:	59006687539
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za organiziranje turnira u košarci korištenjem višekriterijskog raspoređivanja
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	U teorijskom dijelu završnog rada potrebno je proučiti i opisati probleme, izazove i postojeća rješenja za organiziranje sportskih turnira s naglaskom na košarku. Nadalje, treba definirati vremenske, prostorne, financijske i uvjete raspoloživosti ljudskih resursa, odnosno parametre turnira, ekipa i dvorana za održavanje natjecanja, te prikladne kriterije raspoređivanja. Iz tih uvjeta, parametara i kriterija treba definirati funkcionalne i nefunkcionalne zahtjeve, model, arhitekturu i dizajn web aplikacije. Uz to, treba definirati odgovarajuće ovlasti korisnika web aplikacije pri unosu i upravljanju navedenim ulaznim podacima, te prikazu rasporeda utakmica prema ekipama i dvoranama i prikazu ukupnih rezultata turnira. Također, treba ukratko analizirati i izabrati programske tehnologije i razvojnu okolinu za razvoj web aplikacije, programski ostvariti web aplikaciju na strani korisnika i poslužitelja, te je ispitati na prikladnim ulaznim podacima i slučajevima korištenja.
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina

Datum prijedloga ocjene od strane mentora:	13.09.2022.
Datum potvrde ocjene od strane Odbora:	21.09.2022.
Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 21.09.2022.

Ime i prezime studenta:	Roko Barbić
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R 4316, 22.07.2019.
Turnitin podudaranje [%]:	13

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za organiziranje turnira u košarci korištenjem višekriterijskog raspoređivanja**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. PREGLED TRENUTNOG STANJA I POSTOJEĆA RJEŠENJA 2
2.1. Organiziranje sportskih događanja 2
2.2. Izazovi pri organiziranju turnira u košarci 2
2.3. Pregled stanja u području 2
2.4. Slične aplikacije na tržištu 2
2.4.1. Winner 2
2.4.2. Tournify 3
2.4.3. League Lobster 4
2.4.4. Doodle 5
2.4.5. Calendly 5
2.5. Idejno rješenje aplikacije 6
3. MODEL I GRAĐA WEB APLIKACIJE 7
3.1. Funkcionalni zahtjevi na web aplikaciju 7
3.2. Nefunkcionalni zahtjevi na web aplikaciju 7
3.3. Dijagram rada i komponente aplikacije 7
3.4. Višekriterijsko raspoređivanje 9
4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE 10
4.1. Korištene programske tehnologije 10
4.1.1. MySQL 10
4.1.2. HTML 10
4.1.3. CSS 10
4.1.4. JavaScript 10
4.1.5. Express JS 10
4.1.6. JSON 11
4.1.7. Visual Studio Code 11
4.2. Prikaz programskog rješenja 11
4.2.1. Baza podataka 11
4.2.2. Prijava i registriranje korisnika 12
4.2.3. Stvaranje turnira 16
4.2.4. Dodavanje dvorane 21
5. PRIKAZ I ISPITIVANJE RADA WEB APLIKACIJE S ANALIZOM REZULTATA 24

5.1. Način korištenja web aplikacije 24
5.2. Ispitivanje rada web aplikacije 28
5.2.1. Stvaranje dvorana 28
5.2.2. Stvaranje turnira 29
5.3. Analiza rezultata 31
6. ZAKLJUČAK 32
LITERATURA 33
ŽIVOTOPIS 34
SAŽETAK 35
ABSTRACT 36
PRILOZI 37

1. UVOD

Zbog užurbanog tempa života uzrokovanog sve dužim radnim vremenom i drugim obavezama građani se iz zdravstvenih razloga ili iz hobija sve više bave sportom i tako utječu na kvalitetu života. Pronalazak sportskih terena za rekreativno bavljenje sportom često predstavlja veliki izazov za građane. Pored sportskih terena građani se susreću s izazovom pronalaska ljudi i usklađivanje termina za bavljenje sportom. Najveći izazov je stvoriti uvjete za provođenje natjecateljskih događaja. Na tržištu ne postoji puno aplikacija koje pomažu korisnicima pri organiziranju i usklađivanju termina i mjesta održavanja natjecateljskih događaja.

Cilj ovog završnog rada je istražiti postojeća rješenja i usporediti ih te sukladno tome napraviti vlastito rješenje koje će korisnicima ponuditi aplikaciju za organiziranje turnira uz pomoć s naglaskom na košarku. Aplikacija će omogućiti registraciju i prijavu korisnika, dodavanje novih sportskih dvorana, pregled prethodnih turnira i stvaranje novih turnira uz višekriterijsko raspoređivanje.

U drugom poglavlju opisani su izazovi raspoređivanja sportskih događaja i slična postojeća rješenja. Treće poglavlje prikazuje funkcijske i nefunkcijske zahtjeve na aplikaciju i prikazuje dijagram tijeka korištenja aplikacije. U četvrtom poglavlju opisane su korištene programske tehnologije i najbitniji dijelovi programskog koda, dok je u petom poglavlju opisan način korištenja aplikacije te je ispitana i analizirana ispravnost aplikacije.

1.1. Zadatak završnog rada

U teorijskom dijelu završnog rada potrebno je proučiti i opisati probleme, izazove i postojeća rješenja za organiziranje sportskih turnira s naglaskom na košarku. Nadalje, treba definirati vremenske, prostorne i uvjete raspoloživosti ljudskih resursa, odnosno parametre turnira, ekipa i dvorana za održavanje natjecanja, te prikladne kriterije raspoređivanja. Iz tih uvjeta, parametara i kriterija treba definirati funkcionalne i nefunkcionalne zahtjeve, model, arhitekturu i dizajn web aplikacije. Uz to, treba definirati odgovarajuće ovlasti korisnika web aplikacije pri unosu i upravljanju navedenim ulaznim podacima, te prikazu rasporeda utakmica prema ekipama i dvoranama i prikazu ukupnog rasporeda turnira. Također, treba ukratko analizirati i izabrati programske tehnologije i razvojnu okolinu za razvoj web aplikacije, programski ostvariti web aplikaciju na strani korisnika i poslužitelja, te je ispitati na prikladnim ulaznim podacima i slučajevima korištenja. Također treba predložiti koncept napretka aplikacije, te koje su moguće daljnje nadogradnje i razvoji web aplikacije.

2. PREGLED TRENUTNOG STANJA I POSTOJEĆA RJEŠENJA

2.1. Organiziranje sportskih događanja

Organizacija sportskih događaja nosi sa sobom niz izazova, a prije svega potrebno je pronaći zadovoljavajući broj natjecatelja, ali ništa manje nije bitan i pronalazak odgovarajućih prostornih resursa za odvijanje natjecanja. Stoga organizatorova uloga u određivanju mjesta i vremena odvijanja nije lagana jer dostupnost dvorana i vremenskih mogućnosti natjecatelja treba uskladiti na što bolji način.

2.2. Izazovi pri organiziranju turnira u košarci

Izazovi kod organizacije turnira u košarci slični su kao i kod organiziranja većine sportskih događaja. Košarkaška utakmica na turniru sastojala bi se od četiri četvrtine i svaka u trajanju od 10 minuta i vrijeme teče bez prekida tijekom cijele četvrtine, dok bi pauze bile u trajanju od 5 minuta između četvrtina, te velikom pauzom od 10 minuta između 2. i 3. četvrtine, što bi značilo da utakmica traje 60 minuta. Također ekipe bi prije utakmice imale 30 minuta za zagrijavanje. Ukupno vrijeme trajanja utakmice sa zagrijavanjem bilo bi 90 minuta. Često termini za rezervaciju dvorana i jesu 90 minuta što i odgovara izazovu pri organiziranju turnira u košarci. Format turnira kojim se koristi ova web aplikacija je pojedinačna eliminacija, što znači da samo pobjednik utakmice prolazi dalje u sljedeće kolo.

2.3. Pregled stanja u području

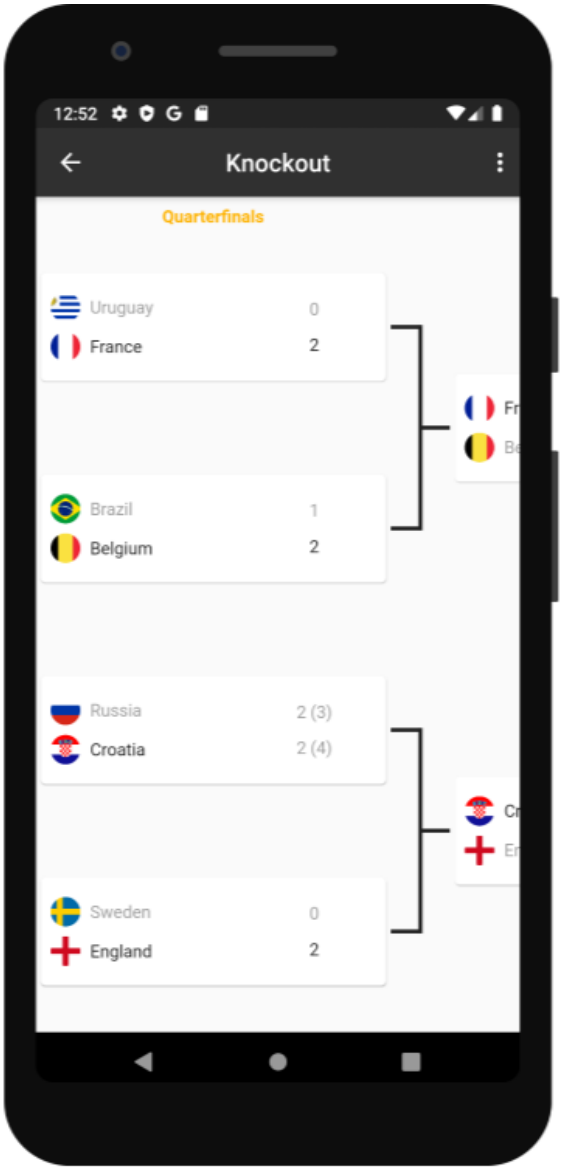
Trenutno stanje na tržištu za aplikacije koje se bave problematikom organiziranja i vremenskog usklađivanja turnira, kakve se opisuju u ovome radu i ne postoje. Postoje aplikacije koje se bave isključivo stvaranjem rasporeda utakmica na pojedinom turniru, ali ni takve aplikacije ne uzimaju u obzir više kriterija, poput vremenske raspoloživosti ekipa i igrališta. Također postoje aplikacije koje se bave isključivo organiziranjem događaja ili sastanaka u ovisnosti o vremenskoj raspoloživosti ljudskih resursa. Iz tog razloga odlučeno je napraviti aplikaciju koja će korisnicima pomoći u rješavanju problema vremenske raspoloživosti u organiziranju turnira, te također odabrati najbolje dostupne dvorane i stvoriti raspored utakmica koji će odgovarati svakoj ekipi.

2.4. Slične aplikacije na tržištu

2.4.1. Winner

Prema [1], Winner je besplatna aplikacija za mobilne uređaje koja se bavi stvaranjem turnira i ima preko milijun preuzimanja na Google Play-u. Aplikacija korisniku dopušta stvaranje turnira

u nogometu, futsalu, PES-u, košarci, odbojci i još mnogim drugim sportovima te dopušta odabir formata turnira od kupa pa do playoff-a. Korisnik također može sam dodavati igrače i ekipe, i također igrališta na kojima će se održavati utakmice. Ono što razlikuje aplikaciju Winner od ovog rada je to što ona ne pruža korisniku raspoređivanje utakmica s obzirom na kriterije. Na slici 2.1 je prikazan zaslon aplikacije Winner.

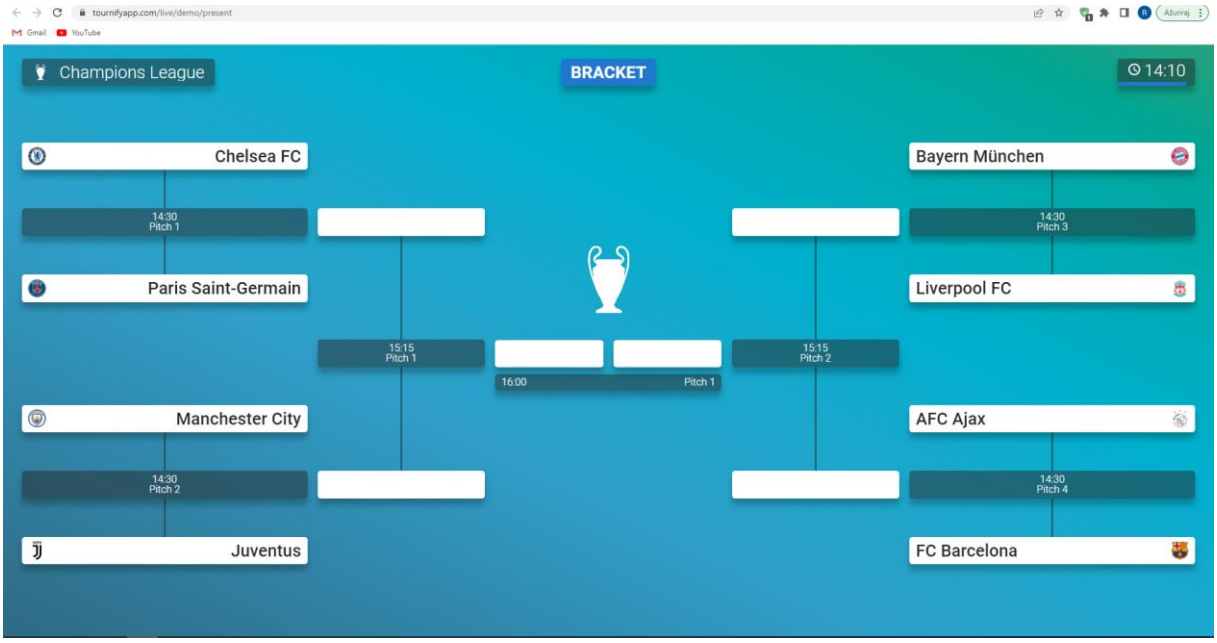


Slika 2.1. Zaslon aplikacije Winner

2.4.2. Tournify

Prema [2], Tournify je aplikacija za mobilne uređaje i web korisnike, koja se također bavi stvaranjem i upravljanjem turnira. Aplikacija korisniku dopušta stvaranje turnira u raznim sportovima te dopušta odabir formata turnira Također aplikacija ima više od 30 000 korisnika i omogućava im komuniciranje putem poruka kako bi dogovorili termin utakmice. Aplikacija

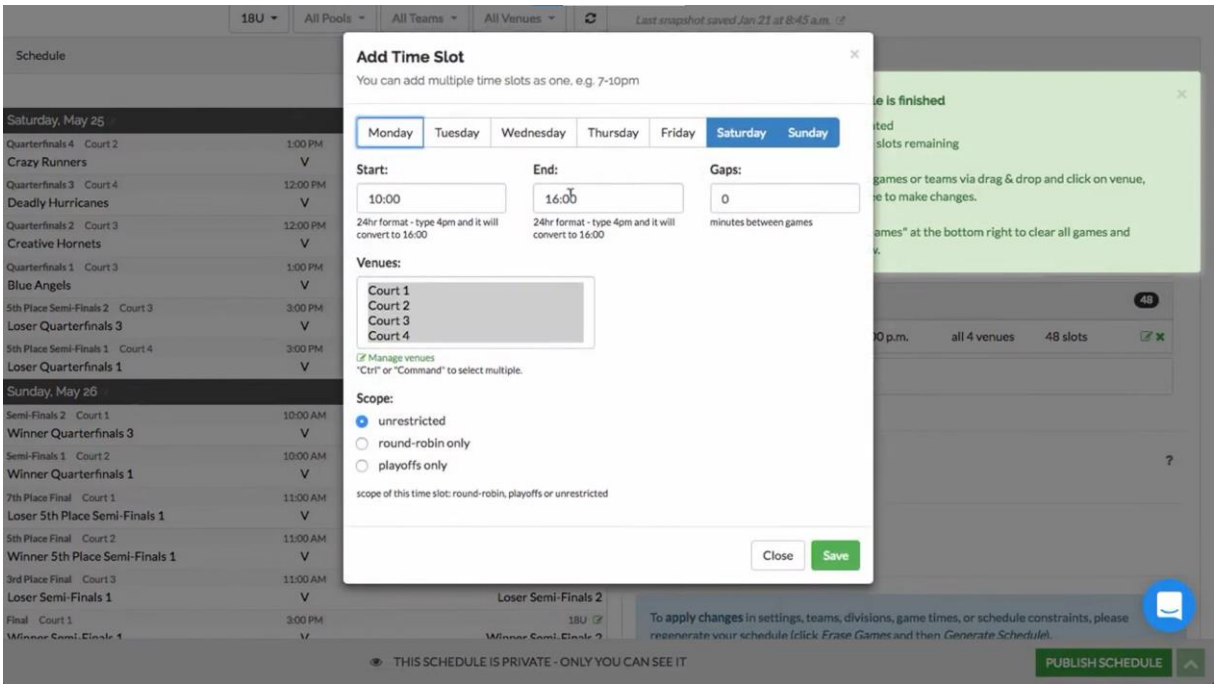
pruža jasan i jednostavan pregled rasporeda i rezultata. Aplikacija Tornify zahtijeva od korisnika ručno unošenje rasporeda utakmica što se može vidjeti na slici 2.2, dok aplikacija u ovome radu ima mogućnost sama napraviti odgovarajući raspored utakmica.



Slika 2.2. Prikaz zaslona aplikacije Tournify

2.4.3. League Lobster

Prema [3], League Lobster je web aplikacija koja se također bavi stvaranjem i upravljanjem turnira i liga. Aplikacija korisniku dopušta stvaranje turnira u raznim sportovima te omogućava odabir formata turnira. Korisniku daje mogućnosti za odabiranje svih detalja potrebnih u kvalitetnoj organizaciji turnira, ali s nedostatkom vrlo kompliciranog sučelja za korisnike. Ono što razlikuju ovu aplikaciju od mog rada je to što korisnik ručno mora popunjavati raspored utakmica. Slika 2.3 prikazuje zaslon aplikacije League Lobster.



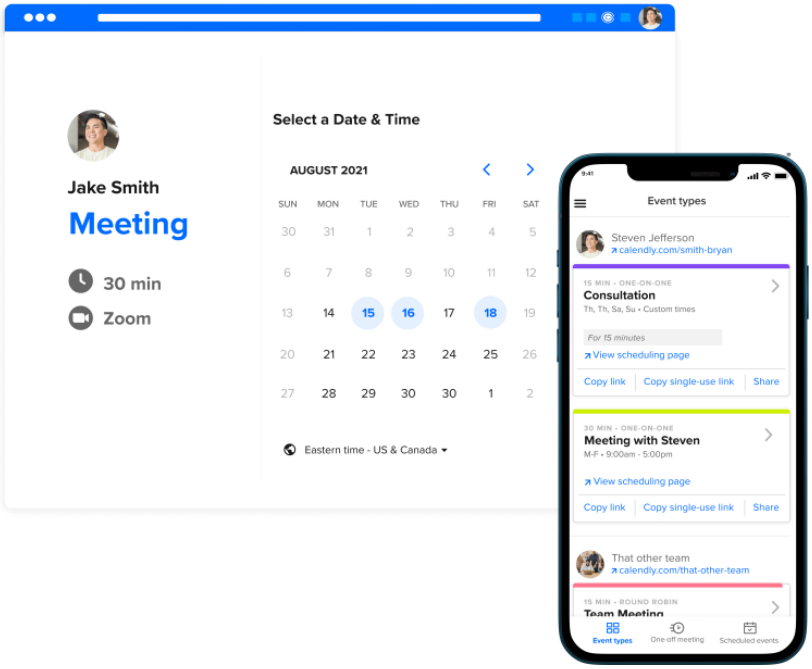
Slika 2.3 Prikaz zaslona aplikacije League Lobster

2.4.4. Doodle

Prema [4], Doodle je web i mobilna aplikacija koja se bavi planiranjem i raspoređivanjem događaja u ovisnosti o vremenskoj raspoloživosti korisnika. Njezin cilj je pronaći termin koji bi odgovarao većini sudionika određenog događaja, u ovom slučaju sastanaka. Aplikacija Doodle funkcionira tako da korisnici postavljaju svoju raspoloživost termina te na taj način drugi korisnici mogu vidjeti njihove termine i pozvati na sastanak u vremenu kad je to moguće. Također ima opciju grupnih anketa s ciljem odabira termina održavanja nekog događaja. Postoji besplatna inačica Doodle aplikacije, ali sve mogućnosti koje ona nudi su moguće jedino uz mjesečnu pretplatu. Ova aplikacija se bavi isključivo izazovima vremenskog raspoređivanja te je to ključna razlika između ovoga rada i aplikacije Doodle.

2.4.5. Calendly

Prema [5], Calendly je također web i mobilna aplikacija koja se bavi rezerviranjem i raspoređivanjem događaja u ovisnosti o vremenskoj raspoloživosti njezinih korisnika. Cilj aplikacije je korisniku omogućiti brže i lakše rezerviranje sastanaka, stoga su njezini izazovi isključivo vezani uz vremensko raspoređivanje te se stoga razlikuje od ovoga rada. Aplikacija ima besplatnu inačicu u kojoj korisnicima daje pristup jednostavnim alatima, dok bi potpuna inačica aplikacije korisnika koštala mjesečnu ili godišnju pretplatu. Izgled zaslona aplikacije Calendly se može vidjeti na slici 2.5.



Slika 2.5. Prikaz izgleda zaslona aplikacije Calendly

2.5. Idejno rješenje aplikacije

Predložena aplikacija trebala bi korisniku, tj. organizatoru turnira omogućiti lakše i brže stvaranje rasporeda utakmica, na način da korisnik unosi vremensku raspoloživost ekipa i za uzvrat dobiva odgovarajući raspored utakmica. Aplikacija bi trebala moći stvoriti turnir s osam ekipa, gdje bi se utakmice održavale isključivo u terminima koji odgovaraju ekipama. Također, utakmice bi se trebale odviti u dvoranama koje korisnik postavi da su većeg prioriteta. Turnir bi se odvijao kroz tri tjedna, gdje bi 1. kolo bilo u 1. tjednu i sadržavalo bi ukupno četiri utakmice, 2. kolo u 2. tjednu s dvije utakmice te finale u 3. tjednu. Korisnik bi trebao imati mogućnost pregleda prethodnih turnira jer bi oni trebali biti spremljeni u bazu podataka. Također, korisnik bi imao mogućnost dodavanja novih dvorana u bazu podataka.

3. MODEL I GRAĐA WEB APLIKACIJE

U ovom poglavlju opisana je analiza zahtjeva na web aplikaciju i dijagram rada aplikacije.

3.1. Funkcionalni zahtjevi na web aplikaciju

Funkcionalni zahtjevi su zahtjevi koje korisnik zahtjeva kao osnovne mogućnosti koje bi sustav trebao moći izvršiti.

Web aplikacija ima sljedeće funkionalne zahtjeve:

- Registriranje i prijava korisnika
- Odabir i pregled prethodnih turnira
- Stvaranje novog turnira te odabir mjesta i tjedna u kojemu se turnir kreće održavati
- Odabir dvorana po prioritetu
- Odabir termina koji ekipama odgovara
- Dodavanja dvorana
- Odjava korisnika iz aplikacije.

Na slici 3.1 mogu se lakše vidjeti funkcionalni zahtjevi.

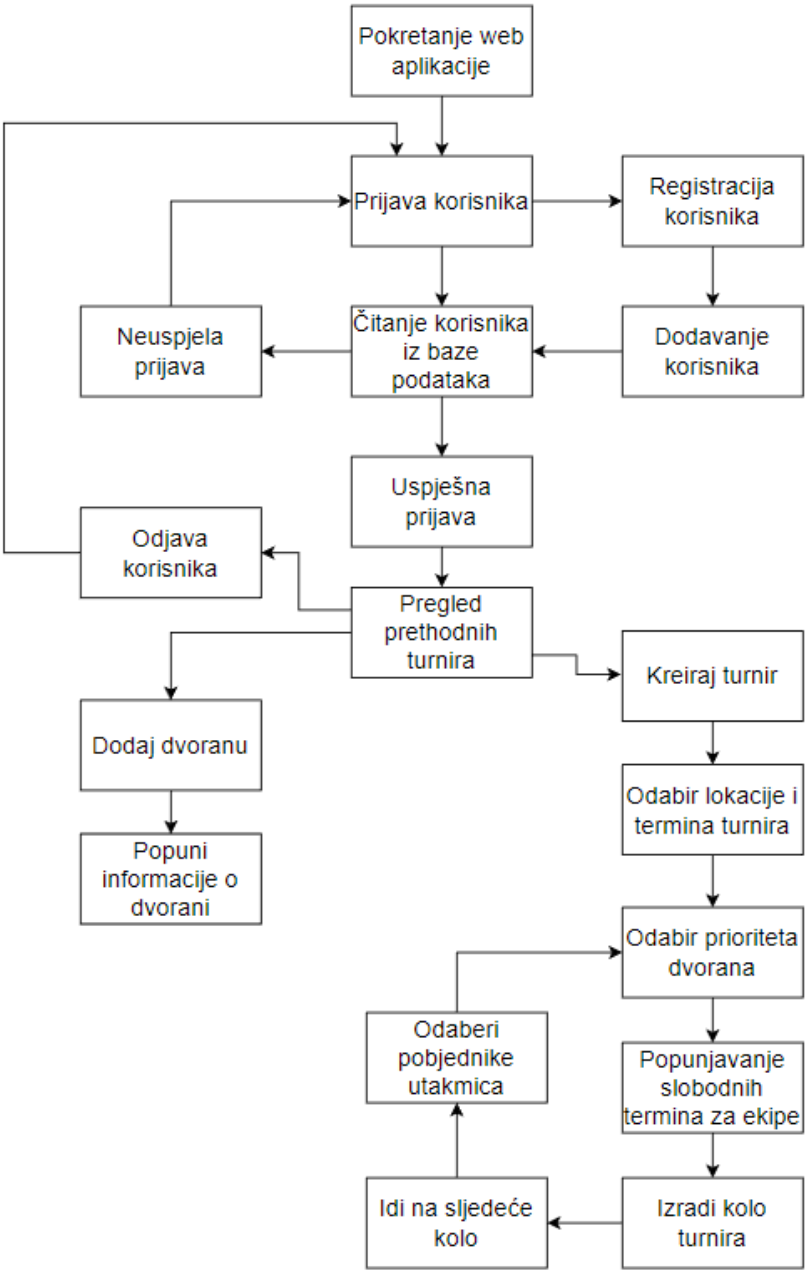
3.2. Nefunkcionalni zahtjevi na web aplikaciju

Nefunkcionalni zahtjevi su zahtjevi koji određuju kvalitetu sustava koji će krajnji korisnici iskusiti i često su vezani uz sigurnost, pouzdanost i održavanje web aplikacije. Neki od nefunkcionalnih zahtjeva su sposobnost korištenja aplikacije i učinkovitost aplikacije. Prema [6], sposobnost korištenja aplikacije je mogućnost lakog razumijevanja toka korištenja aplikacije, tj. korisnici bi trebali moći koristiti aplikaciju bez ikakvih smjernica ili pomoći stručnjaka. Učinkovitost aplikacije se može odrediti prema vremenu odziva, odnosno vremenu potrebnom za dovršavanje danog zadatka.

3.3. Dijagram rada i komponente aplikacije

Dijagram rada aplikacije prikazuje tehničku strukturu za lakše shvaćanje cijelog procesa kroz koji korisnik prolazi tijekom korištenja aplikacije te je vrlo korisno pri stvaranju jer prikazuje strukturu i plan izrade same aplikacije. Pokretanjem web aplikacije, korisniku se prikazuje sučelje za prijavu u aplikaciju, te ima mogućnost odabira registracije ukoliko nema već postojeći račun. Nakon prijave, korisnik ima mogućnost odabira nekog od njegovih prethodnih turnira, a detalji o odabranom turniru će se ispisati na ekranu. Također korisnik ima mogućnost stvaranja novog turnira, gdje se prikazuje sučelje za odabir lokacije i vremena turnira, s obzirom na odabranu lokaciju i vrijeme održavanja turnira korisniku se nude opcije za odabir tri dvorane

po prioritetu. Nakon unosa dvorana korisnika dočekuje sučelje za unos imena ekipa i slobodnih termina u kojima ekipe mogu igrati, te nakon toga korisnik ima mogućnost stvaranja 1. kola turnira, ako je turnir moguće napraviti korisniku se nudi opcija za stvaranje sljedećeg kola turnira. Također, korisnik ima opciju dodavanja dvorane, gdje ga dočekuje sučelje za unos imena dvorane, lokacije, tjedna i vremena u kojemu je dvorana slobodna za odabrani tjedan.



Slika 3.1. Dijagram rada aplikacije

3.4. Višekriterijsko raspoređivanje

Prema [7], rješenje problema raspoređivanja uvijek mora zadovoljiti određeni broj ograničenja i kriterija. Web aplikacija opisana u ovome radu bavi se stvaranjem rasporeda utakmica koji moraju zadovoljiti veći broj kriterija, među kojima su vremenska raspoloživost ekipa i dvorana te razina prioriteta dvorana. Vremenska raspoloživost ekipe odnosi se na vremena u kojima je ekipa sposobna igrati utakmice. Vremenska raspoloživost dvorane opisuje zauzetost dvorane u određenom vremenskom rasponu. Razina prioriteta dvorana se odnosi na listu dvorana koju korisnik sastavljaju kako bi izdvojio dvorane u kojima bi htio igrati utakmice. Kako bi utakmica mogla biti stvorena moraju se ispuniti svi kriteriji, barem dvije ekipe moći igrati u istom terminu i barem jedna dvorana većeg prioriteta mora biti slobodna u tom terminu.

Prema [8], pravilnik o rasporedu utakmica Nacionalnog košarkaškog saveza (NBA) definira složeni problem putujućeg turnira i naziva se problem raspoređivanja (engl. *scheduling problem*) NBA (NBASP). Glavni izazov rješavanja problema je smanjenje duljine putovanja. Također, veliki broj uzastopnih utakmica, četiri utakmice u pet dana i utakmice vikendom su još neki od problema raspoređivanja. NBA rasporeda utakmica za sezonu 2014.-2015. korišten je u testiranju i rezultati su pokazali smanjenje duljine putovanja do 14%.

Prema [9], primjer korištenja višekriterijskog raspoređivanja u profesionalnom sportu je organiziranje nogometnog turnira u Brazilu. Jedan od kriterija je bio povećanje broja utakmica koje se mogu emitirati putem otvorenih TV kanala, a drugi kriteriji je pronalaženje uravnoteženoga rasporeda utakmica s minimalnim brojem pauza kod kuće i u gostima.

4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE

4.1. Korištene programske tehnologije

4.1.1. MySQL

Prema [10], MySQL je sustav za upravljanje bazom podataka. Baza podataka može biti bilo što, od jednostavnog popisa brojeva u imeniku, do galerije slika ili ogromne količine informacija u korporativnoj mreži. Za dodavanje, pristup i obradu podataka pohranjenih u računalnoj bazi podataka potreban vam je sustav za upravljanje bazom podataka kao što je MySQL Server. SQL dio MySQL označava engl. *Structured Query Language*, prevedeno na hrvatski *strukturirani jezik upita*. Također SQL je najčešći standardizirani jezik koji se koristi za pristup bazama podataka.

4.1.2. HTML

Prema [11], HTML (engl. *Hyper* Text *Markup Language*) je prezentacijski jezik koji služi za izradu web stranica. Njegova jednostavnost i laka uporaba su jedni od razloga njegove opće prihvaćenosti i popularnosti. Besplatan je i dostupan svima, a prikaz hiperteksta omogućuje web preglednik na računalu. HTML nije programski jezik i s njim ne možemo izvršavati ni najjednostavnije zadaće. Osnovni gradivni elementi svake stranice su tagovi (engl. *tags*) koji opisuju način prikazivanja u web pregledniku.

4.1.3. CSS

Prema [12], CSS (engl. *Cacade Style Sheet*) je stilski jezik za dodatno uređivanje stranica koje su napisane u HTML prezentacijskom jeziku. Pomoću CSS-a stranice je moguće dodatno urediti te ih prilagoditi svim vrstama ekrana. CSS je nezavisan od HTML-a, ali se najčešće koriste zajedno. Razdvajanjem HTML-a i CSS-a se dobiva lakše održavanje stranica, dijeljenje stilova na više stranica i prilagođavanje stranica drugim okolinama.

4.1.4. JavaScript

Prema [13], JavaScript je javno raspoloživ skriptni jezik. Cilj njegovog kreiranja bio je dodati interaktivnost HTML stranicama. JavaScript je interpreter odnosno njegova se skripta izvršava odmah naredbu po naredbu bez prethodnog prevođenja cijelog programa i kreiranja izvršne datoteke. Omogućava nam programiranje u okviru HTML stranica, a u HTML se umeće pomoću skripte.

4.1.5. Express JS

Prema [14], Express JS je sloj programiran iznad Node.js koji pomaže u upravljanju poslužiteljima i rutama. Ukratko Node.js je okruženje za izvršavanje JavaScripta koje koristimo

za izradu aplikacija na strani poslužitelja, ali ono ne zna kako izvršiti posluživanje datoteka, rukovati zahtjevima, te također ne zna rukovati HTTP metodama, tako da zbog toga koristimo Express JS.

4.1.6. JSON

Prema [15], JSON je format podataka temeljen na tekstu koji slijedi sintaksu JavaScript objekta, koju je popularizirao Douglas Crockford. Iako vrlo nalikuje sintaksi JavaScript objekta, može se koristiti neovisno o JavaScriptu, a brojna programska okruženja imaju mogućnost čitanja i generiranja JSON-a. JSON postoji kao tekstualni niz što je korisno kada želite prenijeti podatke preko mreže, ali treba ga pretvoriti u izvorni JavaScript objekt kada želite pristupiti podacima.

4.1.7. Visual Studio Code

Prema [16], Visual Studio Code je besplatan, lagan, ali moćan uređivač izvornog koda. Dostupan Je za Windows, macOS, Linux i Raspberry Pi OS. Dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js te ima bogat ekosustav proširenja za druge programske jezike.

4.2. Prikaz programskog rješenja

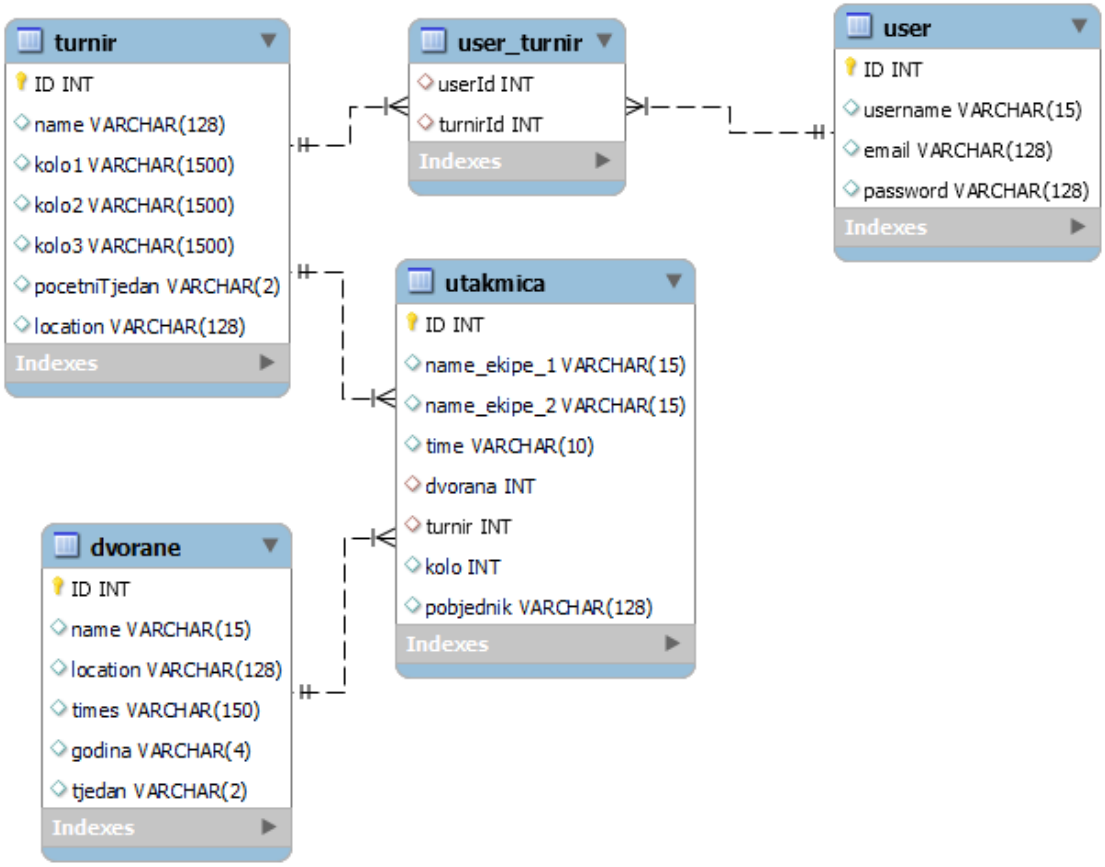
U nastavku su opisani ključni dijelovi programskog koda ove web aplikacije.

4.2.1. Baza podataka

Web aplikacija koristi MySQL kao sustav za upravljanje bazom podataka. Baza podataka se sastoji od pet tablica, koje su aplikaciji potrebne za prijavu korisnika, stvaranje turnira i vođenje računa o slobodnim terminima dvorana. Na slici 4.1 prikazan je izgled dijagrama baze podataka. Tablica *turnir* posjeduje informacije o pojedinom turniru, u njoj se nalaze podatci kao što su ID (engl. *identity*), tj. identitet turnira koji je zaseban za svaki turnir, ime turnira, početni tjedan i lokacija na kojoj se turnir održava, te također u tekstualnom obliku se nalazi raspored utakmica za svako kolo.

Tablica *user* posjeduje informacije o korisniku kao što su njegov jedinstveni ID, korisničko ime, lozinka i email adresa, te one služe pri prijavi korisnika u aplikaciju. Tablica *user_turnir* drži informacije o ID-u korisnika i ID-u turnira, uz pomoć kojih aplikacija dohvaća korisnikove turnire. Tablica *utakmica* posjeduje informacije o jedinstvenom ID utakmice, imenima ekipa koje igraju utakmicu, ID turnira na kojem se utakmica održava i kolu u kojem se nalazi, dvorani i vremenu u kojem se održava, te informaciji tko je pobjednik utakmice. Tablica *dvorane* posjeduje informacije o dostupnosti pojedine dvorane u određenom tjednu u godini, dostupnost dvorane je zapisana u tekstualnom obliku, ali aplikacija tu informaciju prebacuje u polje nula i

jedinica pošto ono predstavlja vremensku raspoloživost dvorane.



Slika 4.1. Prikaz dijagrama baze podataka

4.2.2. Prijava i registriranje korisnika

Za korištenje ove web aplikacije nužno je imati korisnički račun, pa se stoga pri otvaranju aplikacije korisnik prvo mora prijaviti ili registrirati ukoliko nema već postojeći račun. Slika 4.2 prikazuje HTML kod koji se odnosi na prijavu korisnika, vrlo je jednostavan i sastoji se od polja u kojima korisnik unosi ime i lozinku.

```html
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <link rel="stylesheet" href="style.css">
        <script src="script.js"></script>

    </head>
    <body>
        <div class="Login-container">
            <label for="ime"><b>Korisničko ime:</b></label>
            <input type="text" placeholder="Unesi ime" name="ime" id="ime" required>

            <label for="lozinka"><b>Lozinka:</b></label>
            <input type="password" placeholder="Unesi lozinku" name="lozinka" id="lozinka" required>

            <button class="button" type="submit" onclick="GoRegistriraj()" >Registriraj se</button>
            <button class="button" type="submit" onclick="Prijava()" >Prijava</button>
            <p id="error"></p>
            <label>
        </div>
    </body>
</html>
```

Slika 4.2. Prikaz HTML koda za prijavu korisnika

Na slikama 4.3 i 4.4 su prikazani dijelovi programskog koda za prijavu korisnika u web aplikaciju.

```javascript
function Prijava(){
    console.log("prijava")
    const ime = document.getElementById('ime').value;
    const lozinka = document.getElementById('lozinka').value;

    fetch('http://localhost:3000/login', {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify({
            "username": ime,
            "password": lozinka
        })
    }).then( (response) =>{
        if(response.status == 200){
            window.location = "pages/HomePage.html";
        } else if(response.status == 400) {
            const error = document.getElementById('error')
            error.innerText = "Nisu uneseni svi podaci."
        }
    })
}
```

Slika 4.3. Prikaz funkcije za prijavu korisnika

```
app.post('/login', (req, res) => {
    const username = req.body.username
    const password = req.body.password

    if(!username || !password){
        res.sendStatus(400)
        return
    }

    connection.query('SELECT ID, username, password FROM user WHERE username = ?', [username], function (error, results, fields) {
        if (error){
            res.sendStatus(500)
            return
        }

        if(results.length == 0 ||  results[0].password !== password){
            res.sendStatus(400)
            return
        }
        const user = {"id": results[0].ID,
                      "username": results[0].username}
        req.session.user = user
        res.sendStatus(200)
      });
})
```

Slika 4.4. Prikaz metode rute za provjeru korisnika

Slika 4.3 prikazuje kod funkcije *Prijava()* koja se poziva kada korisnik pritisne gumb za prijavu. Ova funkcija dohvaća ime i lozinku koju je korisnik unio te ih šalje web rutom kako bi provjerio u bazi podataka ispravnost. Slika 4.4 prikazuje metodu rute, koristi se funkcija *post* koja služi za slanje podataka i definiran je */login* koji služi kao adresa za dohvaćanje i slanje podataka. Kod provjerava postoji li uneseno ime korisnika u bazi podataka i ispravnost upisane lozinke, ako su svi uvjeti zadovoljeni šalje statusni HTTP kod (200) koji ukazuje da su određeni HTTP zahtjevi uspješno dovršeni, te funkcija *Prijavi()* korisnika upućuje na početnu stranicu aplikacije. su uneseni krivi podatci ispisat će se tekst koji ukazuje na vrstu pogreške pri prijavi. Prikaz programskog koda za registraciju korisnika se nalazi na slikama 4.5, 4.6 i 4.7.

```
<body>
    <div class="Register-container">
        <label for="ime"><b>Korisničko ime:</b></label>
        <input type="text" placeholder="Unesi ime" name="ime" id="ime" required>

        <label for="email"><b>E-mail adresa:</b></label>
        <input type="text" placeholder="Unesi E-mail adresu" name="email" id="email" required>

        <label for="lozinka"><b>Lozinka:</b></label>
        <input type="password" placeholder="Unesi lozinku" name="lozinka" id="lozinka" required>

        <label for="lozinka"><b>Potvrdi lozinka:</b></label>
        <input type="password" placeholder="Unesi lozinku" name="lozinka" id="potvrda_lozinke" required>

        <p id="error"></p>
        <button class="button" type="submit" onclick="Registracija()">Registriraj se</button>
        <label>
    </div>
</body>
```

Slika 4.5. Prikaz HTML koda za registraciju

```
function Registracija(){
    const ime = document.getElementById('ime').value;
    const email = document.getElementById('email').value;
    const lozinka = document.getElementById('lozinka').value;
    const potvrda_lozinke = document.getElementById('potvrda_lozinke').value;
    if(lozinka != potvrda_lozinke){
        const error = document.getElementById('error')
        error.innerText = "Potvrda lozinke nije ista kao i lozinka."
    }
    else{
        fetch('http://localhost:3000/register', {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({
                "username": ime,
                "email": email,
                "password": lozinka
            })
        }).then( (response) =>{
            if(response.status == 200){
                window.location = "HomePage.html";
            } else if(response.status == 500){
                const error = document.getElementById('error')
                error.innerText = "Korisničko ime ili email već postoje."
            } else if(response.status == 400) {
                const error = document.getElementById('error')
                error.innerText = "Nisu uneseni svi podaci."
            }
        })
    }

}
```

Slika 4.6. Prikaz funkcije za registraciju korisnika

```
app.post('/register', (req, res) => {
    console.log("POST register")
    console.log(req.body)
    const username = req.body.username
    const email = req.body.email
    const password = req.body.password

    if(!username || !email || !password){
        console.log("login info not complete")
        res.sendStatus(400)
        return
    }

    connection.query('INSERT INTO user(username, email, password) VALUES (?,?,?)', [username, email, password], function (error, res
        if (error){
            console.log(error)
            res.sendStatus(500)
            return
        }
        // connected!
        res.sendStatus(200)
    });
})
```

Slika 4.7. Prikaz metode rute za registraciju korisnika

Registriranje korisnika funkcionira na način da korisnik popuni informacije o svom imenu, emailu i lozinki. Funkcija sa slike 4.6 te podatke šalje metodom rute na adresu sa nastavkom */registration* te upravitelj uz pomoć tih podataka sprema korisnika u bazu podataka i ukoliko nema nikakve greške šalje potvrdni statusni HTTP kod, s tom informacijom funkcija *Registracija()* korisnika usmjerava na početni zaslon aplikacije.

4.2.3. Stvaranje turnira

Turnir se kreira tako da korisnik prvo mora popuniti osnovne informacije o turniru, kao što su ime turnira, lokacija i tjedan u kojemu se turnir počinje održavati. Potom mu se nude opcije za odabir tri dvorane koje su mu od najvećeg prioriteta. Na slikama 4.8 i 4.9 mogu se vidjeti dijelovi programskog za stvaranje turnira.

```
<div class="Fill_turnir">
    <p>Ime Turnira:</p>
    <input type="text" id="Ime_turnira" class="Input_ime" maxlength="50">
    <label for="dvorane">Odaberi lokaciju na kojoj se dvorana nalazi:</label>
        <select id="lokacija" name="dvorane" onchange="fetchDvorane()">
            <option value="Osijek">Osijek</option>
            <option value="Zagreb">Zagreb</option>
            <option value="Split">Split</option>
        </select>
    <label for="dvorane">Odaberi tjedan:</label>
        <select id="tjedan" name="tjedan" onchange="fetchDvorane()">
        </select>
    <label for="dvorane">Odaberi dvorane ovisno o prioritetu, 1. dvorana: </label>
    <select id="dvorana1"  ></select>
    <label for="dvorane">2. dvorana: </label>
    <select id="dvorana2" ></select>
    <label for="dvorane">3. dvorana: </label>
    <select id="dvorana3" ></select>
    <p></p>
    <button class="button"   id="buttonDvorana" onclick="AddPDvorana(1)">Unesi dvorane</button>
    <p> </p>
    <p id="UneseneDvorane" class="UnesenoP"></p>
</div>
```

Slika 4.8. Prikaz jednog dijela HTML koda za stvaranje turnira

Slika 4.8 prikazuje HTML kod u kojemu se može vidjeti da korisnik ima tri opcije za odabir grada i također treba odabrati tjedan u godini koji će biti početni tjedan turira.

Funkcija *fetchDvorane()* sa slike 4.9 ima ulogu dohvaćanja dvorana iz baze podataka koje se nalaze u odabranom gradu i u tjednu koji je korisnik odabrao kao početni tjedan turnira, te dvorane koje je dohvatio nudi ih korisniku kao opcije za odabir, klikom na gumb *Unesi dvorane,* odabrane dvorane se spremaju kao dvorane većeg prioriteta. Nakon unosa dvorana potrebno je unijeti ekipe te je funkcija za unos ekipa prikazana na slici 4.10.

Prilikom stvaranja turnira aplikacija od korisnika zahtjeva unos imena ekipa i vremena kada su pojedine ekipe u mogućnosti igrati utakmice u tjednu koji je odabran, to je izvedeno tako da postoji polje za unos imena, te tablica s vremenima u periodu od tjedan dana u kojoj su smješteni potvrdni okviri (engl. *checkbox*) koji predstavljaju termine, svaki termin je duzine devedeset minuta. Postoji 10 mogućih termina u danu kroz tjedan dana, što bi bilo 70 termina za koje se ekipe moraju odlučiti u kojima mogu igrati.

```
function fetchDvorane(){
    const lokacija = document.getElementById('lokacija').value;
    const tjedan = document.getElementById('tjedan').value;

    if(!lokacija || !tjedan){
        return
    }

    fetch(`http://localhost:3000/dvorana/${tjedan}/${lokacija}`).then( (response) =>{
        if(response.status == 200){
            response.json().then(data =>{
                select1.innerHTML = ""
                select2.innerHTML = ""
                select3.innerHTML = ""

                data.forEach(dvorana =>{
                    console.log(dvorana)
                    const opt1 = document.createElement('option');
                    opt1.setAttribute("value", dvorana.ID)
                    opt1.innerHTML = dvorana.name;
                    select1.appendChild(opt1);

                    const opt2 = document.createElement('option');
                    opt2.setAttribute("value", dvorana.ID)
                    opt2.innerHTML = dvorana.name;
                    select2.appendChild(opt2);

                    const opt3 = document.createElement('option');
                    opt3.setAttribute("value", dvorana.ID)
                    opt3.innerHTML = dvorana.name;
                    select3.appendChild(opt3);
            }))
        } else if(response.status == 400) {
        }
    })
}
```

Slika 4.9. Prikaz funkcije za dohvaćanje dvorana

```
function funkcijaa1(p1){
    const ime = document.getElementById('Ime'+p1);
    var checkboxes = document.getElementsByClassName("ck"+p1);
    var ukupno = [];
    for(let i=0; i<checkboxes.length; i++){
        if(checkboxes[i].checked){
            ukupno.push(1);
        }
        else{
            ukupno.push(0);
        }
    }
    document.getElementById("rjesenje"+p1).innerHTML = "POPUNJENO";
    if(p1==1){
        const ekipaP = new Ekipa(ukupno, ime.value, 0);
        ekipa1.CopyEkipa(ekipaP);
        if(ekipa1counter == 0){
            ekipe.push(ekipa1);
            ekipa1counter++;
        }

    }
```

Slika 4.10. Prikaz funkcije za unos ekipe

Funkcija sa slike 4.10 se poziva kada korisnik pritisne gumb za unos ekipe, te nakon toga učitava ime ekipe i potvrdne okvire, te od potvrdnih okvira stvara polje nula i jedinica koje predstavljaju vremensku raspoloživost ekipe. Ukoliko ekipa može igrati potvrdni okvir bi bio označen te bi funkcija dodala u polje 1, ako potvrdni okvir nije označen funkcija u polje dodaje 0. Svako mjesto u polju predstavlja točno jedan termin u tjednu. Prilikom stvaranja turnira, korisnik mora popuniti informacije za svih 8 ekipa, na isti način popunjava i za ekipe u kasnijim kolima turnira. Nakon što su sve informacije za ekipe popunjene korisnik može stvoriti kolo turnira te se funkcija za stvaranje kola turnira može vidjeti na slikama 4.11 i 4.12.

```
function algoritamRada(kolo){
    let numberOfUtakmica = 0;
    let text2 = "";
    var ekipaa1 = new Ekipa(a, "lol", 0);
    var ekipaa2 = new Ekipa(a, "ne radi", 0);
    var Pdvorana = new Dvorana([0,0], " ");

    var counter=0;
    const algoritamRada_utakmice = []
    for (let i = 0; i < 70; i++) {
        if(IsEnoughEkipa(i)){
            for (const dvorana of dvorane) {
                if(dvorana.times[i]==1)
                {
                    dvorana.times[i]=0;
                    for (const ekipa of ekipe) {
                        if (ekipa.IsScedualed == 0 && ekipa.times[i] == 1 && counter==0)
                        {
                            counter++;
                            ekipaa1.CopyEkipa(ekipa);
                            ekipa.IsScedualed = 1;
                            ekipa.times[i]=0;
                            text2+=ekipa.name + " ";
                        }
                        else if (ekipa.IsScedualed == 0 && ekipa.times[i] == 1 && ekipa.CanPlay(i) && counter == 1)
                        {
                            numberOfUtakmica++;
                            counter=0;
                            ekipaa2.CopyEkipa(ekipa);
                            let utakmica1 = new Utakmica(ekipa1, ekipa2, Pdvorana, i);
                            utakmice.push(utakmica1);
                            ekipa.IsScedualed = 1;
                            ekipa.times[i]=0;
                            text2+=ekipa.name + " " + dvorana.name + " " + ConvertVrijemeIntToH(i) + ", \n";
                            let textVrijeme = ConvertVrijemeIntToH(i);
                            i--;
                            algoritamRada_utakmice.push({
                                "name_ekipa1": ekipaa1.name,
```

Slika 4.11. Prikaz funkcije za stvaranje kola turnira (1. dio)

```
                                "name_ekipa2": ekipaa2.name,
                                "time": textVrijeme,
                                "dvorana": dvorana.name,
                                "kolo": kolo
                            })

                            break;
                        }
                    }
                    break;
                }
            }
        }
        ekipaa1.ClearEkipa();
        ekipaa2.ClearEkipa();
        Pdvorana.ClearDvorana();
        counter = 0;
        if(i == 69 && numberOfUtakmica<ekipe.length/2){
            text2 = "Nije moguce napraviti turnir, pokusajte ponovo!"
            return
        }
    }
    document.getElementById("prvaUtakmica").innerHTML = text2;
    if(kolo === 1){
        //stvori turnir
        const imeTurnira = document.getElementById('Ime_turnira').value;
        const tjedan = document.getElementById('tjedan').value;
        const lokacija = document.getElementById('lokacija').value;
        fetch('http://localhost:3000/turnir', {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({
                "turnir": {
                    "ime":imeTurnira,
                    "kolo1": text2
```

```
                    "lokacija": lokacija
                },
                "utakmice": algoritamRada_utakmice,
                "tjedan": tjedan
            })
        }).then( (response) =>{
            if(response.status == 200){
                dvorane.forEach(dvorana => {
                    UpdateDvorana(dvorana)
                });
                response.json().then(data => noviTurnirId = data.turnirId)
            } else if(response.status == 400) {
            }
        })
    } else {
        //update turnir
        const queryString = window.location.search;
        const urlParams = new URLSearchParams(queryString)
        const turnirId = urlParams.get("turnir")
        fetch(`http://localhost:3000/turnir/${turnirId}`, {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({
                "turnir": {
                    "koloText": text2,
                    "kolo": kolo
                },
                "utakmice": algoritamRada_utakmice
            })
        }).then( (response) =>{
            if(response.status == 200){
                noviTurnirId = turnirId
            } else if(response.status == 400) {
            }
        })
    }
}
```

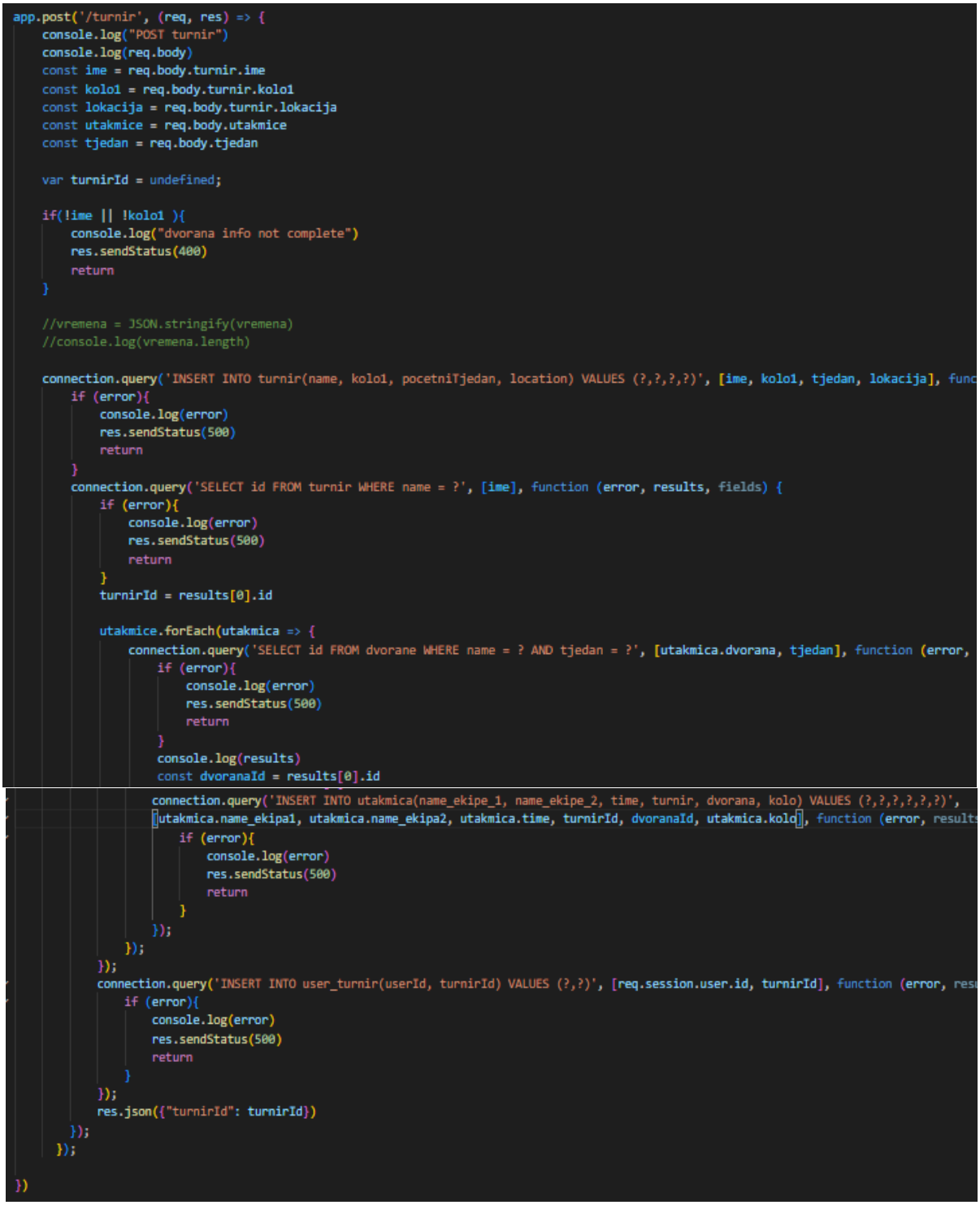
Slika 4.12. Pikaz funkcije za stvaranje kola turnira (2. dio)

Funkcija *algoritamRada()* služi za stvaranje kola turnira te je prikazana na slikama 4.11 i 4.12. Ona se poziva kada korisnik popuni ranije navedene podatke koji su potrebni kako bi se turnir mogao stvoriti. Funkcija kao parametar prima koje kolo stvara pošto nisu sva kola turnira ista. Prvo kolo turnira je ono koje će započeti turnir, ima osam ekipa što znači da ima četiri utakmice i odvija se u tjednu koji je korisnik odabrao, drugo kolo ima četiri ekipe i dvije utakamice te je vrijeme održavanja tog kola tjedan poslije početnog tjedna, zatim zadnje kolo je finale turnira i odvija se u tjednu poslije 2. kola.

Algoritam unutar funkcije *algoritamRada()* ima zadatak pronalaska ekipa koje su voljne igrati u istom terminu i za taj zadani termin traži dvoranu koja je slobodna, počevši s onom od najvećeg prioriteta. Kako bi to postigao prvo se izvršava petlja koja prolazi kroz svih 70 termina u tjednu i poziva funkciju *IsEnoughEkipa()* koja provjerava za svaki termin postoje li barem dvije ekipe koje mogu igrati, ako se taj uvjet ispuni onda provjerava postoji li dvorana koja je slobodna u tom terminu. Potom ako su oba uvjeta ispunjena znači da su osnovni uvjeti za stvaranje utakmice ispunjeni, onda će program odabrati slobodnu dvoranu koja je najveća po prioritetu i njenu vrijednost za taj termin u polju vremenske raspoloživosti promijeniti u nulu, također će proći još jednom kroz polje svih ekipa i pronaći koje su to ekipe u mogućnosti igrati u tom terminu i njihove vrijednosti spremiti u privremene varijable koje su namijenjene za stvaranje utakmica. Kada su svi podatci potrebni za stvaranje utakmice pronađeni, algoritam stvara utakmicu i sprema je u bazu podataka. Svaka utakmica je također zapisana u tekstualnom obliku tako da pišu imena ekipa, termin održavanja utakmice i dvorana u kojoj se utakmica igra. Kada je utakmica napravljena petlja će se vratiti na isti termin, kako bi bilo moguće napraviti više utakmica u istom terminu, ali u različitim dvoranama i s različitim ekipama. Petlja će proći kroz sve termine. Kada je poznat raspored svih utakmica u kolu, tekstualni zapis utakmica se sprema u tablicu turnir u bazi podataka pod *kolo1*, *kolo2* ili *kolo3*, ovisno o kojem kolu se radi. Programski kod za spremanje turnira je prikazan na slici 4.13.

Na slikama 4.11 i 4.12 prikazano je da se na adresu */turnir* šalju podaci web rutom o stvorenom turniru i utakmicama koji su uz pomoć JSON-a pretvoreni u tekstualni oblik. Metoda rute je opisana na slikama 4.13 i 4.14 te prikazuje na koji način se spremaju podatci u bazu podataka. Ukoliko su svi zahtjevi nad bazom izvedeni na ispravan način, bez pogreške, šalje se statusni HTTP kod (200) koji opet ukazuje da su određeni HTTP zahtjevi uspješno dovršeni, nakon čega

korisnik ima mogućnost pravljenja sljedećeg kola turnira.



```
app.post('/turnir', (req, res) => {
    console.log("POST turnir")
    console.log(req.body)
    const ime = req.body.turnir.ime
    const kolo1 = req.body.turnir.kolo1
    const lokacija = req.body.turnir.lokacija
    const utakmice = req.body.utakmice
    const tjedan = req.body.tjedan

    var turnirId = undefined;

    if(!ime || !kolo1 ){
        console.log("dvorana info not complete")
        res.sendStatus(400)
        return
    }

    //vremena = JSON.stringify(vremena)
    //console.log(vremena.length)

    connection.query('INSERT INTO turnir(name, kolo1, pocetniTjedan, location) VALUES (?,?,?,?)', [ime, kolo1, tjedan, lokacija], func
        if (error){
            console.log(error)
            res.sendStatus(500)
            return
        }
        connection.query('SELECT id FROM turnir WHERE name = ?', [ime], function (error, results, fields) {
            if (error){
                console.log(error)
                res.sendStatus(500)
                return
            }
            turnirId = results[0].id

            utakmice.forEach(utakmica => {
                connection.query('SELECT id FROM dvorane WHERE name = ? AND tjedan = ?', [utakmica.dvorana, tjedan], function (error,
                    if (error){
                        console.log(error)
                        res.sendStatus(500)
                        return
                    }
                    console.log(results)
                    const dvoranaId = results[0].id
                    connection.query('INSERT INTO utakmica(name_ekipe_1, name_ekipe_2, time, turnir, dvorana, kolo) VALUES (?,?,?,?,?,?)',
                    [utakmica.name_ekipa1, utakmica.name_ekipa2, utakmica.time, turnirId, dvoranaId, utakmica.kolo], function (error, results
                        if (error){
                            console.log(error)
                            res.sendStatus(500)
                            return
                        }
                    });
                });
            });
            connection.query('INSERT INTO user_turnir(userId, turnirId) VALUES (?,?)', [req.session.user.id, turnirId], function (error, resu
                if (error){
                    console.log(error)
                    res.sendStatus(500)
                    return
                }
            });
            res.json({"turnirId": turnirId})
        });
    });

})
```

Slika 4.14. Prikaz metode rute za adresu */turnir*

4.2.4. Dodavanje dvorane

Korisnik aplikacije ima mogućnost dodavanja dvorana u kojima se utakmice mogu odigrati. Ovo je nužna stavka aplikacije pošto se na neki način vremenska raspoloživost dvorana treba unijeti u bazu podataka. Programski kod za unos dvorana je prikazan na slikama 4.14 i 4.15.

Dvorane se unose pozivom funkcije *funkcija()* sa slike 4.15, ali prilikom stvaranja turnira njihova vremenska raspoloživost se također mijenja i ažurira po potrebi. Početna vremenska raspoloživost se postavlja sa funckija *funkcija()* na isti način kao i kod ekipa, gdje se učitava stanje potvrdnih okvira i iz toga se pravi polje jedinica i nula koje predstavljaju vremensku raspoloživost dvorane za svaki termin u tjednu. Funkcija sve informacije o dvorani uz pomoć JSON-a pretvara u tekstualni oblik te ih šalje web rutom na adresu */dvorana* kako bi ih upravitelj spremio u bazu podataka.

```
<div class="Fill_turnir">
    <p>Ime Dvorane:</p>
    <input type="text" id="Ime1" class="Input_ime" maxlength="15">
    <label for="dvorane">Odaberi tjedan:</label>
        <select id="tjedan" name="tjedan" >
        </select>

    <label for="dvorane">Odaberi lokaciju na kojoj se dvorana nalazi:</label>
    <select id="lokacija" name="dvorane">
        <option value="Osijek">Osijek</option>
        <option value="Zagreb">Zagreb</option>
        <option value="Split">Split</option>
    </select>
</div>
<p class="TerminTxt">Unesi termine kada je dvorana slobodna:</p>
<div class="row">
    <div class="column">
        <h3>Ponedjeljak</h3>
        <div>
            <input type="checkbox" value="1" id="0" class="ck1">
            <label for="0">8:00</label>
        </div>
        <div>
            <input type="checkbox" value="1" id="1" class="ck1">
            <label for="1">9:30</label>
        </div>
        <div>
            <input type="checkbox" value="1" id="2" class="ck1">
            <label for="2">11:00</label>
        </div>
        <div>
            <input type="checkbox" value="1" id="3" class="ck1">
            <label for="3">12:30</label>
        </div>
```

Slika 4.14. Prikaz jednog dijela HTML koda za dodavanje dvorane

```
function funkcija(p1){
    const lokacija =document.getElementById('lokacija');
    const ime = document.getElementById('Ime'+p1);
    var checkboxes = document.getElementsByClassName("ck"+p1);
    var ukupno = [];
    for(let i=0; i<checkboxes.length; i++){
        if(checkboxes[i].checked){
            ukupno.push(1);
        }
        else{
            ukupno.push(0);
        }
    }
    const dvoranaP = new Dvorana(ukupno, ime.value, lokacija.value);
    dvorana.CopyDvorana(dvoranaP);
    console.log("Dvorana POST")
    console.log(ukupno)
    const tjedan =document.getElementById('tjedan');
    fetch('http://localhost:3000/dvorana', {
        method: "POST",
        headers: {
            "Content-Type": "application/json"
        },
        body: JSON.stringify({
            "name": dvorana.name,
            "week": tjedan.value,
            "location": dvorana.location,
            "times": ukupno,
        })
    }).then( (response) =>{
        if(response.status == 200){
            //
        } else if(response.status == 400) {
            //ERORR
        }
    })
}
```

Slika 4.15. Prikaz funkcije za unos dvorane

5. PRIKAZ I ISPITIVANJE RADA WEB APLIKACIJE S ANALIZOM REZULTATA

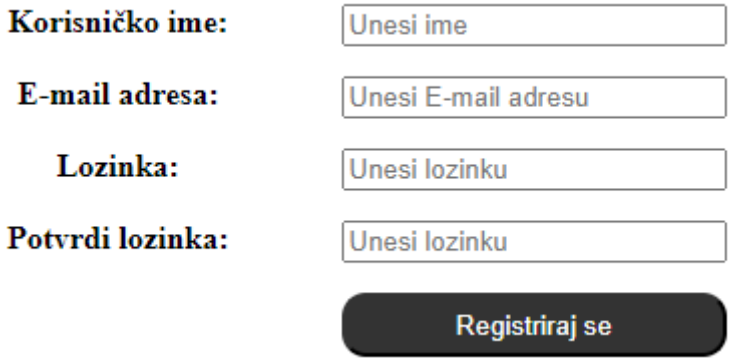
U ovom poglavlju bit će opisan način korištenja aplikacije od strane korisnika, njezin izgled i ispitana točnost rada aplikacije.

5.1. Način korištenja web aplikacije

Otvaranjem web aplikacije korisnik je usmjeren na prijavu u aplikaciju, koja je nužna za ispuniti, te ako korisnik nema račun onda se mora registrirati kako bi mogao nastaviti s korištenjem aplikacije. Izgledi zaslona za registraciju i prijavu u aplikaciju su prikazani na slikama 5.1 i 5.2.

Slika 5.1. Prikaz zaslona za prijavljivanje u aplikaciju

S ispravno unesenim imenom i lozinkom, korisnik je upućen na početni zaslon aplikacije.



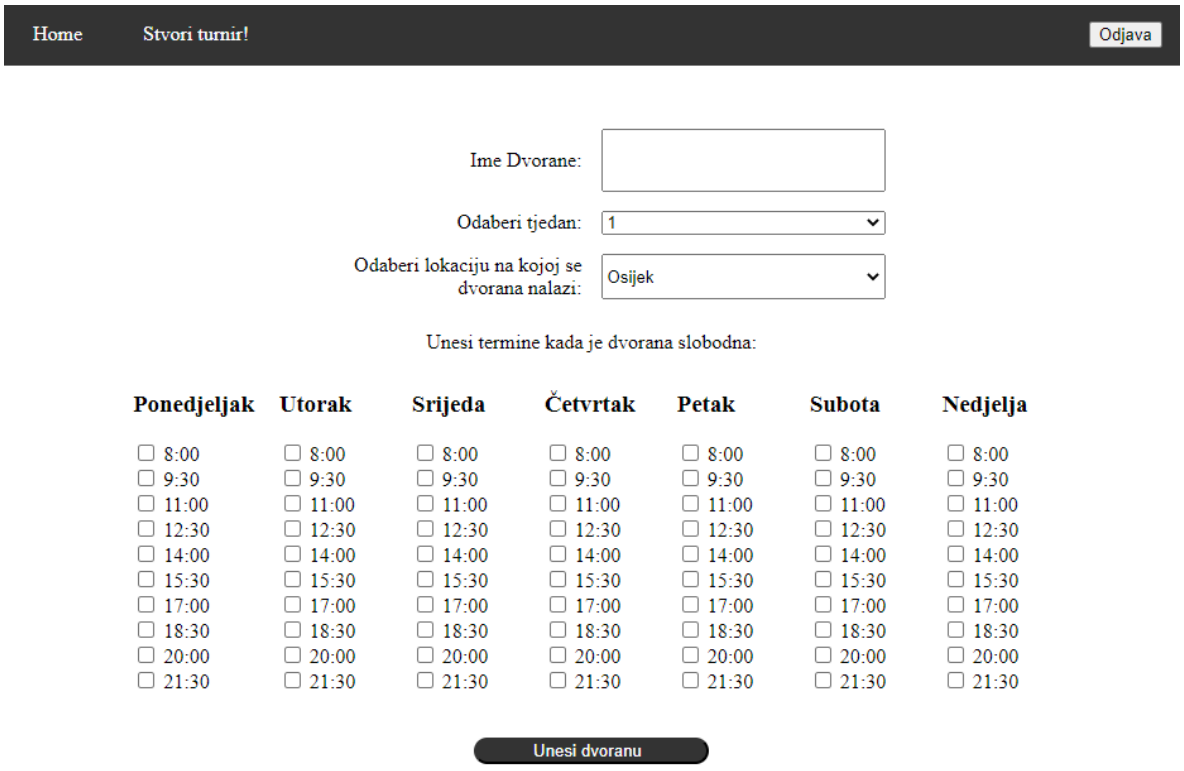
Slika 5.2. Prikaz zaslona za registraciju korisnika

Za registriranje korisnika potrebno je unijeti podatke o korisničkom imenu, adresi elektroničke pošte (engl. *e-mail adress*) i lozinki, te također lozinku treba potvrditi kako ne bi došlo do slučajne greške pri upisu. Korisničko ime i email moraju biti jedinstveni, u protivnom će korisnik dobiti poruku da pokuša s novim podatcima jer već postoji račun s istim.

Nakon prijave korisnik je upućen na početni zaslon aplikacije, na kojemu ima mogućnost pregleda prethodnih turnira što se može i vidjeti iz slike 5.3.

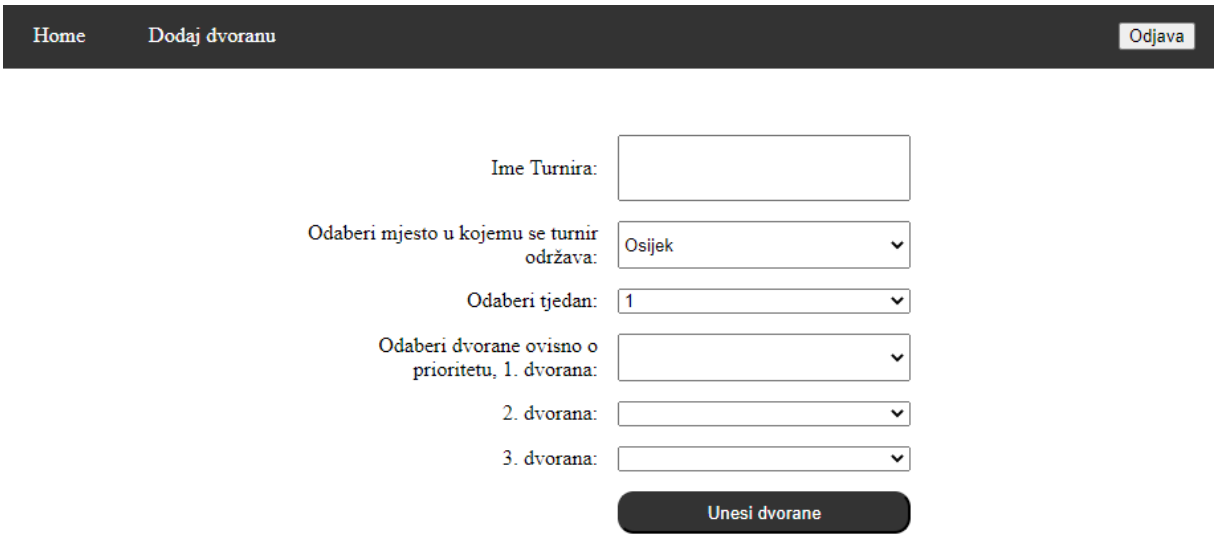
Slika 5.3. Djelomični prikaz početnog zaslona web aplikacije

Slika 5.3 prikazuje na koji način korisnik odabire prethodni turnir koji želi vidjeti, te u gornjem izborniku ima mogućnosti za odabir stvaranja novog turnira ili dodavanje dvorane. Odabirom *Dodaj dvoranu* korisnik je usmjeren na zaslon za dodavanje dvorane, što je prikazano na slici 5.4.



Slika 5.4. Prikaz zaslona za dodavanje dvorane

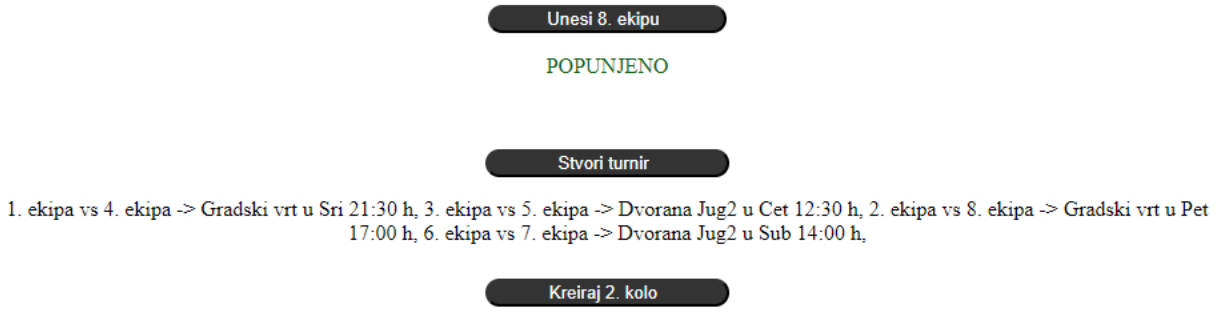
Ime dvorane i tjedan u kojem se nalazi moraju biti jedinstveni, jer ako već postoji dvorana s istim imenom i istim tjednom u bazi podataka, nova dvorana se neće moći stvoriti. Korisnik pri dodavanju dvorane treba unijeti ime dvorane, grad u kojemu se dvorana nalazi, te odabrati jedan tjedan u godini za koji će predstaviti termine u kojima je dvorana slobodna.



Slika 5.5. Prikaz dijela zaslona za stvaranje turnira

Slika 5.5 prikazuje da stvaranje turnira od korisnika zahtjeva unos imena turnira, mjesta održavanja turnira i također tjedna u kojemu će turnir započeti s održavanjem. S obzirom na ranije unesene podatke korisniku će biti ponuđene sve dvorane iz baze podataka koje odgovaraju unesenim podacima, te korisnik treba odabrati tri dvorane u kojima bi htio da utakmice budu održane.

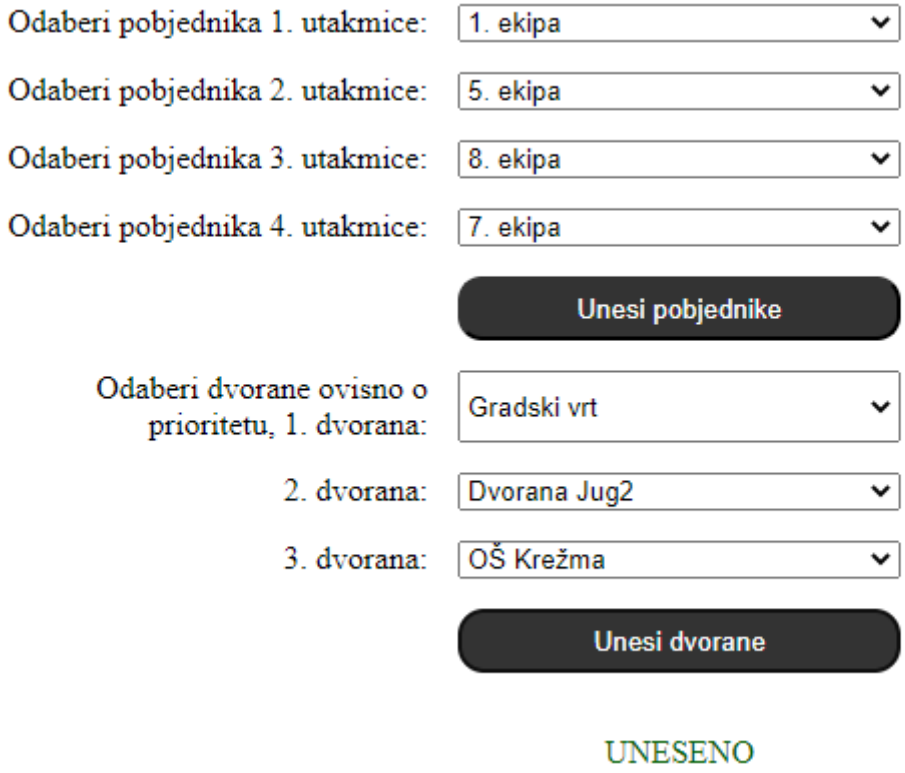
Nakon što su uneseni podatci za sve ekipe, moguće je stvoriti prvo kolo turnira. Ono se stvara pritiskom gumba *Stvori turnir* te će se raspored utakmica ispisati na zaslonu što se može vidjeti na slici 5.6, ako nije moguće stvoriti turnir, ispisat će se poruka koja ukazuje na problem pri stvaranju.



Slika 5.6. Prikaz izgleda stranice pri stvaranju prvog kola turnira

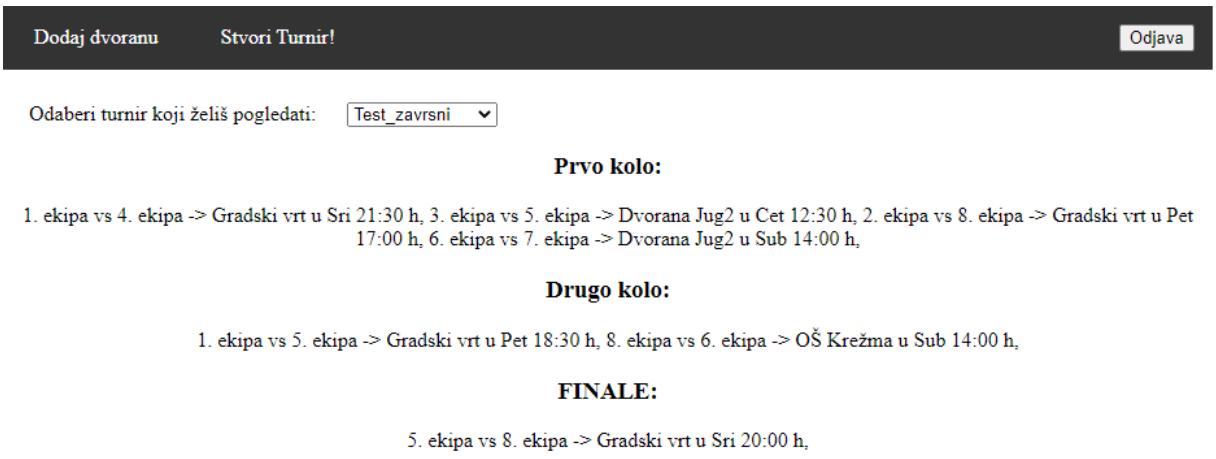
Korisnik nakon stvaranja 1. kola ima mogućnost stvaranja 2. kola turnira, a kada odabere tu opciju bit će usmjeren na stranicu za stvaranje 2. kola prikazano na slici 5.7. Korisnik će morati odabrati pobjednika svake utakmice, te zatim opet odabrati dvorane po prioritetu pošto se turnir odvija u drugom tjednu. Potom korisnik treba ispuniti informacije o tome kada su pobjedničke ekipe slobodne i spremne igrati utakmice 2. kola, nakon toga može stvoriti raspored utakmica

za 2. kolo pritiskom na gumb *Stvori turnir*.



Slika 5.7. Prikaz unosa podataka za stvaranje 2. kola turnira

Na isti način kao i do sada, korisniku će biti ponuđena opcija za stvaranje sljedećeg kola turnira što bi u ovom slučaju bilo finale. Pritom će korisnik opet morati unijeti pobjednike i vremensku raspoloživost ekipa. Nakon toga ima opciju stvaranja finala i samim time završava se proces stvaranja turnira, te korisnik na početnom zaslonu web aplikacije može stvoreni turnir pogledati kao što se vidi na slici 5.8.



Slika 5.8. Prikaz početnog zaslona aplikacije

5.2. Ispitivanje rada web aplikacije

Ispitivanje rada aplikacije će utvrditi ispunjavanje glavnih značajki, odnosno funkcionalnosti aplikacije. Ispitivanje će biti provedeno tako da će se stvoriti dvorane i turnir koji će se odigrati u tim dvoranama.

5.2.1. Stvaranje dvorana

Za početak, kako bi se moglo stvoriti turnir prvo se mora napraviti dvorane u kojima se turnir može održati.

Ispitivanje stvaranja dvorane je izvršeno tako da se stvori četiri različite dvorane i svaku dvoranu potrebno je definirati za 10., 11. i 12. tjedan, lokacija dvorana je Osijek, a vremenske raspoloživosti dvorana su različite za svaku dvoranu.

Imena stvorenih dvoranu su Gradski vrt, Zrinjevac, Dvorana Jug2 i OŠ Krežma.

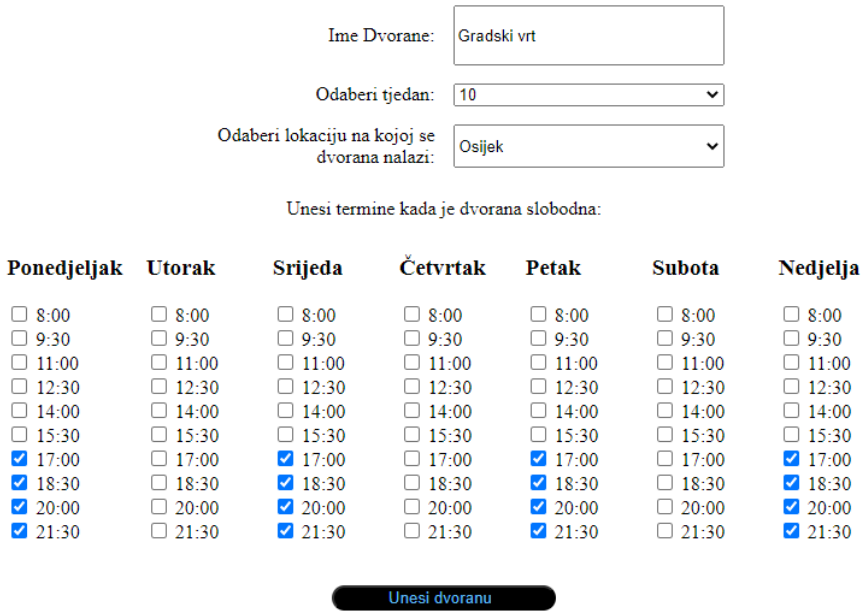
Vremenska raspoloživost svake dvorane ostaje ista kroz sva 3 tjedna radi jednostavnosti provođenja ispitivanja.

Dvorana Gradski vrt slobodna je ponedjeljkom, srijedom, petkom i nedjeljom u terminima od 17:00 h do kraja dana, što se vidi i na slici 5.9.

Dvorana Zrinjevac slobodna je isključivo nedjeljom i to kroz cijeli dan.

Dvorana Jug2 slobodna je četvrtkom i subotom od 12:30h do 15:30h.

Dvorana OŠ Krežma slobodna je isključivo subotom cijeli dan.



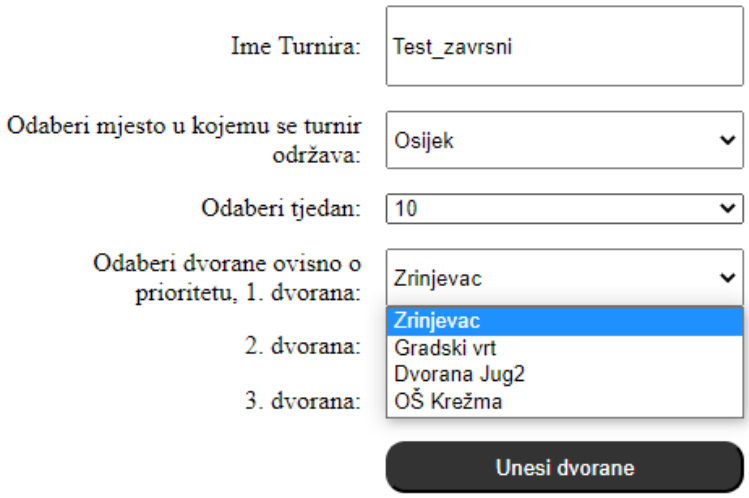
Slika 5.9. Prikaz stvaranja dvorane Gradski vrt u 10. tjednu

Kako bi se provjerilo da su dvorane uistinu stvorene, one bi trebale biti ponuđene kod stvaranja turnira kada se popune informacije o lokaciji i tjednu početka turnira. To je potvrđeno na slici

5.10.

5.2.2. Stvaranje turnira

Dvorane iz prijašnjeg poglavlja služit će pri stvaranju tunira, što znači da će se turnir održati u Osijeku s početkom u 10. tjednu, ime turnira će biti *Test_zavrsni*.



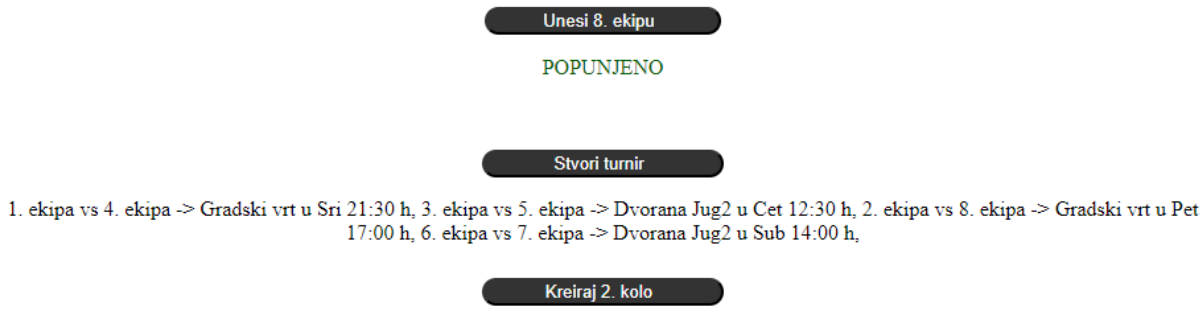
Slika 5.10. Prikaz osnovnih podataka za stvaranje turnira

Dvorane su postavljene po prioritetu tako da je prva Gradski vrt, druga je dvorana Jug2 i treća je OŠ Krežma.

Ekipe su imenovane tako da je ime prve ekipe *1. ekipa* , druge *2. ekipa* i tako dalje. Vremenska raspoloživost ekipa je postavljena da:

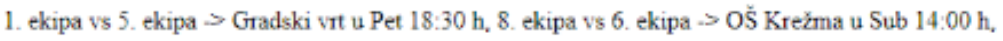
- *1. ekipa* može igrati u utorak cijeli dan, srijedu u 21:30h te nedjelju cijeli dan
- *2. ekipa* može igrati u utorak cijeli dan i petak također
- *3. ekipa* može igrati jedino u četvrtak 12:30h
- *4. ekipa* može igrati tijekom cijele srijede
- *5. ekipa* može igrati u četvrtak od 8:00h do 14:00h
- *6. ekipa* može igrati tijekom cijele subote
- *7. ekipa* može igrati u četvrtak u 12:30h i subotu u 14:00h
- *8. ekipa* može igrati u petak od 17:00h pa do kraja dana

Nakon što su sve ekipe definirane može se stvoriti 1. kolo turnira te se na slici 5.11. vidi raspored utakmica.

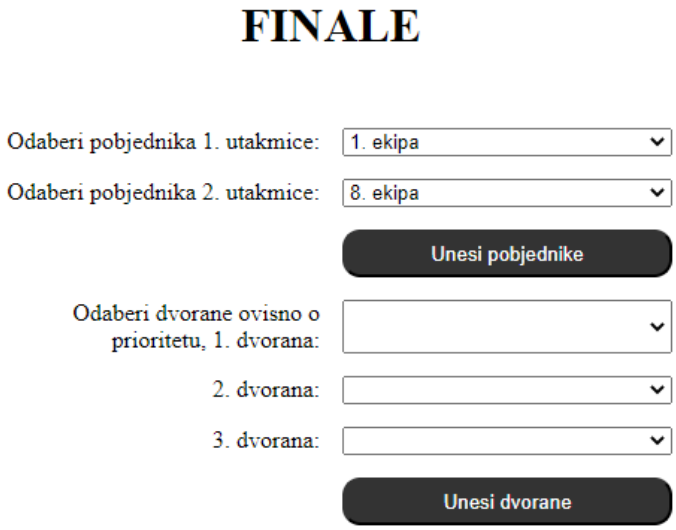


Slika 5.11. Prikaz rasporeda utakmica za 1. kolo turnira

Slika 5.11 prikazuje da je uspješno napravljen raspored utakmica za 1. kolo. te slika 5.7. prikazuje uspješno prikazivanje stvaranja drugog kola turnira. Dvorane po prioritetima su posložene kao i u 1. kolu, te za pobjednike utakmica 1. kola odlučeno je da su *1. ekipa*, *5. ekipa, 6. ekipa* i *8. ekipa*. Vremenska raspoloživost pobjedničkih ekipa je nasumično određena, te su rezultati rasporeda utakmica za 2. kola prikazani na slici 5.12.

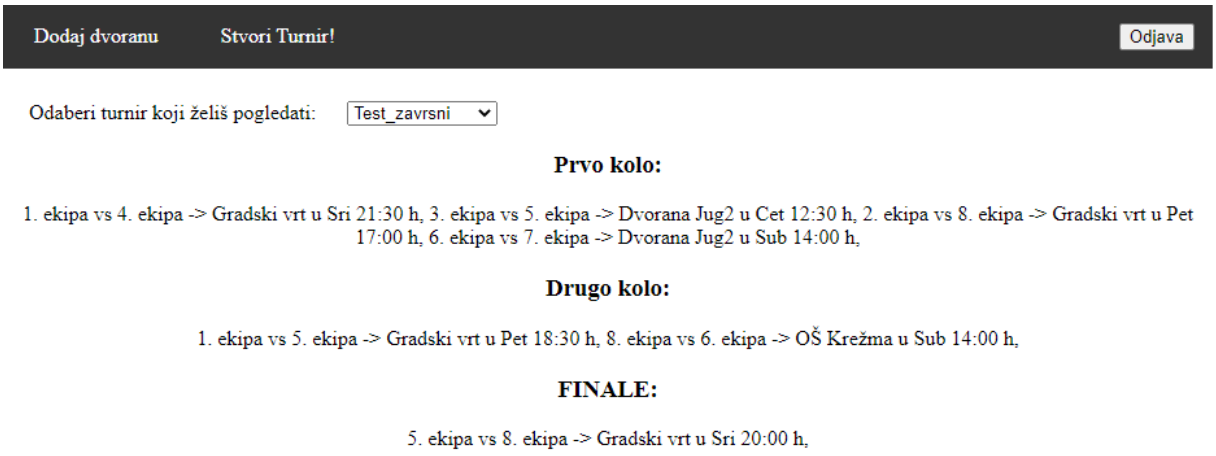


Slika 5.12. Prikaz rasporeda utakmica za 2. kolo turnira.



Slika 5.13. Prikaz unosa podataka o finalu turnira

Za pobjednike drugog kola turnira odabrani su *5. ekipa* i *8. ekipa.* Vremenska raspoloživost tih ekipa je opet nasumično određena te finale turnira je zakazano u terminu i dvorani koji su prikazani na slici 5.14.



Slika 5.14. Prikaz ispisa cijelog turnira

5.3. Analiza rezultata

U potpoglavlju 5.2 prikazano je uspješno stvaranje dvorana i turnira. U testnom primjeru iz prošlog potpoglavlja može se vidjeti da su dvorane stvorene i da postoje kroz sva tri tjedna održavanja turnira te sa time potvrđuju točnost rada aplikacije. U testnom primjeru za stvaranje turnira prikazano je da aplikacija uzima u obzir vremensku raspoloživost ekipa i dvorana, što se može vidjeti zato što nije stvorena niti jedna utakmica koja ne bi odgovarala ekipi ili dvorani. Također, može se prepoznati da se odabir dvorana provodio uzimajući u obzir prioritete korisnika.

6. ZAKLJUČAK

U današnje vrijeme došlo je do povećanja obaveza u svakodnevnom životu ljudi, te manjka slobodnog vremena koji nije lagano uskladiti s obavezama drugih ljudi, što čini organiziranje sportskih događaja teškim. Stoga je cilj ovoga rada olakšati organizatorima turnira složeni posao sastavljanja rasporeda utakmica koji bi svima odgovarao. Korisniku, koji bi u ovom slučaju bio organizator turnira u košarci bilo je omogućeno stvaranje turnira i dodavanje novih dvorana u bazu podataka.

Za izradu korisničkog sučelja korišten je HTML i CSS, a za izradu web aplikacije na poslužiteljskoj strani korišteni su JavaScript i Express JS. Kako bi korisnici mogli pregledati svoje prethodne turnire i voditi računa o dvoranama koristila se baza podataka MySQL. Web aplikacija je stvorena i ispitana i za granične slučajeve korištenja koji pokazuju njenu točnost u radu.

Moguća unaprjeđenja web aplikacije tiču se stvaranja korisničkog računa za ekipe, kako bi se komunikacija između ekipa i organizatora događala izravno preko aplikacije. Također, dodatna nadogradnja na bazu podataka bi bila usmjerena praćenju rezultata utakmica i statistika igrača.

LITERATURA

[1] Aplikacija Winner, https://winner-9bee4.firebaseapp.com/ [25.6.2022.]

[2] Aplikacija Tournify, https://www.tournifyapp.com/ [30.6.2022.]

[3] Aplikacija League Lobster, https://scheduler.leaguelobster.com/home/ [30.6.2022.]

[4] Aplikacija Doodle, https://doodle.com/en/about-us/ [6.9.2022.]

[5] Aplikacija Calendly, https://calendly.com/features [7.9.2022.]

[6] Nefunkcionalni zahtjevi, https://medium.com/@vishwasng/non-functional-requirement-of-the-mobile-development-system-e0ed98f2a872 [13.9.2022.]

[7] Vincent T'kindt i Jean-Charles Billaut, Multicriteria Scheduling Theory, Models and Algorithms, Springer-Verlag Media Berlin, Heidelberg, 2006

[8] Feng-Cheng Yang, NBA Sports Game Scheduling Problem and GA-based Solver, Institute of Industrial Engineering National Taiwan University, Taipei, Taiwan

[9] Celso C. Riberio, Sports scheduling: problems and applications

[10] MySQL, https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html [30.6.2022.]

[11] HTML, https://hr.wikipedia.org/wiki/HTML [30.6.2022.]

[12] J. Niederst Robbins, Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics, O'Reilly Media, Newton, 2012.

[13] JavaScript, http://www.mathos.unios.hr/wp/wp2009-10/P8_Java.pdf [30.8.2022.]

[14] Express JS, https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js [7.9.2022.]

[15] JSON, https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON [7.9.2022.]

[16] Visual Studio Code, https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html [30.6.2022.]

ŽIVOTOPIS

Roko Barbić rođen je 24. studenog 2000. godine u Osijeku. Pohađao je osnovnu školu Antuna Mihanovića u Osijeku, a nakon toga upisuje Prirodoslovnu matematičku gimnaziju Osijek. Nakon završetka srednjoškolskog obrazovanja upisuje Fakultet Elektrotehnike, Računarstva i Informacijskih tehnologija u Osijeku, preddiplomski studij računarstva.

SAŽETAK

Web aplikacija za organiziranje turnira u košarci korištenjem višekriterijskog raspoređivanja razvijena u ovom radu temelji se na potrebama poboljšanja i olakšavanja organiziranja sportskih događaja, a u prvom redu u košarci. Razvijena web aplikacija korisniku omogućuje registriranje i prijavu u sustav, dodavanje novih dvorana u sustav, stvaranje turnira i pregled prijašnjih turnira. Glavni cilj aplikacije je stvaranje turnira gdje korisnik unosi vremenske raspoloživosti ekipa koje sudjeluju u turniru te za uzvrat dobiva raspored utakmica koji odgovara svakoj ekipi. Korišten je programski jezik JavaScript, te Express JS koji pomaže u upravljanju poslužiteljima i rutama, a za bazu podataka korišten je MySQL. Rezultati ispitivanja aplikacije pokazuju njenu ispravnost u radu.

Ključne riječi: JavaScript, košarka, organiziranje turnira, višekriterijsko raspoređivanje, web aplikacija.

ABSTRACT

The web application for organizing basketball tournaments using multi-criteria scheduling developed in this paper is based on the need to improve and facilitate the organization of sports events, primarily basketball. The developed web application allows the user to register and log in to the system, add new sports halls to the system, create tournaments, and view previous tournaments. The main goal of the application is to create a tournament where the user enters the time availability of the teams participating in the tournament and in return receives a match schedule that suits each team. The programming language JavaScript was used, as well as Express JS, which helps in managing servers and routes, and MySQL was used for the database. The test results of the application show its accuracy in work.

Keywords: JavaScript, basketball, tournament organizing, multi-criteria scheduling, web application.

PRILOZI

Prilog 1. Završni rad u datoteci docx

Prilog 2. Završni radu u datoteci pdf

Prilog 3. Programski kod projekta web aplikacije