Igra križić kružić za dva igrača na Android platformi

Turić, Danijel

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Danijel Turić

IGRA KRIŽIĆ KRUŽIĆ ZA DVA IGRAČA NA ANDROID PLATFORMI

ZAVRŠNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Danijel Turić

Matični broj: 4450/15–R

Studij: Informacijski sustavi

IGRA KRIŽIĆ KRUŽIĆ ZA DVA IGRAČA NA ANDROID PLATFORMI

ZAVRŠNI RAD

Mentor:

Dr. sc. Konecki Mladen

Varaždin, kolovoz 2018.

Danijel Turić

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema ovog rada je izrada sustava za igru za više igrača (eng. *Multiplayer game*) bazirana na mobilnoj tehnologiji, prvenstveno na Android platformu. Sustav se sastoji od baze podataka, servera te aplikacije. Baza je MySQL baza podataka, server je napisan u GoLang programskome jeziku, te sama aplikacija je napisana i izvođena na Android operativnom sustavu.

Primjer igre koja je izabrana za realizaciju multiplayer sustava je igra križić kružić. U igri sudjeluju dva igraća. Igra se igra tako da svaki od igraća naizmjenice postavlja križić ili kružić na podlogu koja je podijeljena na 9 polja. Pobjednik igre je igrač koji uspije spojiti 3 polja uzastopno. Ako prilikom 9 poteza na polju nisu povezana tri polja istim znakom tada se igra proglašava neriješenom.

Računalne igre su se u zadnjih dva do tri desetljeća drastično napredovale, tako da u sada više nije zabavno igrati protiv računala, već se stvaraju sustavi koji igraču omogućuju da igra protiv drugih igrača.

Glavna karakteristika sustava igre za više igrača je ta da svaki igrač upravlja svojom igrom(u ovom primjeru je to igra na Android mobilnom uređaju), svaki korak igrača se šalje na poslužitelja, koji obrađuje podatke te ih bilježi i prosljeđuje drugom igraču, te se na njegovom uređaju prikazuju potezi njega i njegovog protivnika.

Ključne riječi: Android operativni sustav, GoLang, MySQL, Android studio, GoLand, Multiplayer, Križić kružić

Sadržaj

1. Uvod

Tema ovog završnog rada koja je opisana u naslovu je, Igra za više igrača na Android platformi. Kao što sam naslov govori potrebno je napraviti sustav koji će omogućiti da dva igrača igraju jedan protiv drugoga, svatko na svome uređaju. Glavna karakteristika ovakvog sustava igranja igara je ta da se omogućuje uspoređivanje vještina igrača. Prije je način uspoređivanja bilo bodovanje. Kod igre u kojoj igraju više igrača nema „čekanja“, odnosno ne događa se da jedan igrač čeka da drugi igrač završi igru te da ju on može zaigrati, te da kad igrač završi sa igrom da se uspoređuju rezultati. Već se vještine uspoređuju istovremeno kada se igra igra.

Sama tema pokriva nekolicinu područja, među kojima su: baze podataka, programiranje, algoritmi, mreže, programiranje na strani poslužitelja, mobilne tehnologije, te samo projektiranje sustava. Sva gore navedena područja su započeta prilikom studiranja na fakultetu, te je ovo prilika da se sva ta područja zaokruže u jednu cjelinu te da se proširi njihovo znanje.

Motivacija za igru je ta da je autor ovog rada igrao igre i još ih igra, te se stvorila radoznalost što stoji iza igre, odnosno samog sustava koji omogućuje igru za više igrača. Prilikom početaka igranja igara glavna misao na pameti bila je igra, ali s vremenom se glavna misao mijenjala u to kako sustav radi i što se događa „iza kulisa“. Iz tog razloga je odabrana baš ova tema rada. Rad je načinjen na način da igraju dva igrača istodobno, igra se malim preinakama može izmijeniti tako da se u igru mogu pridružiti i više igrača.

2. Metode i tehnike rada

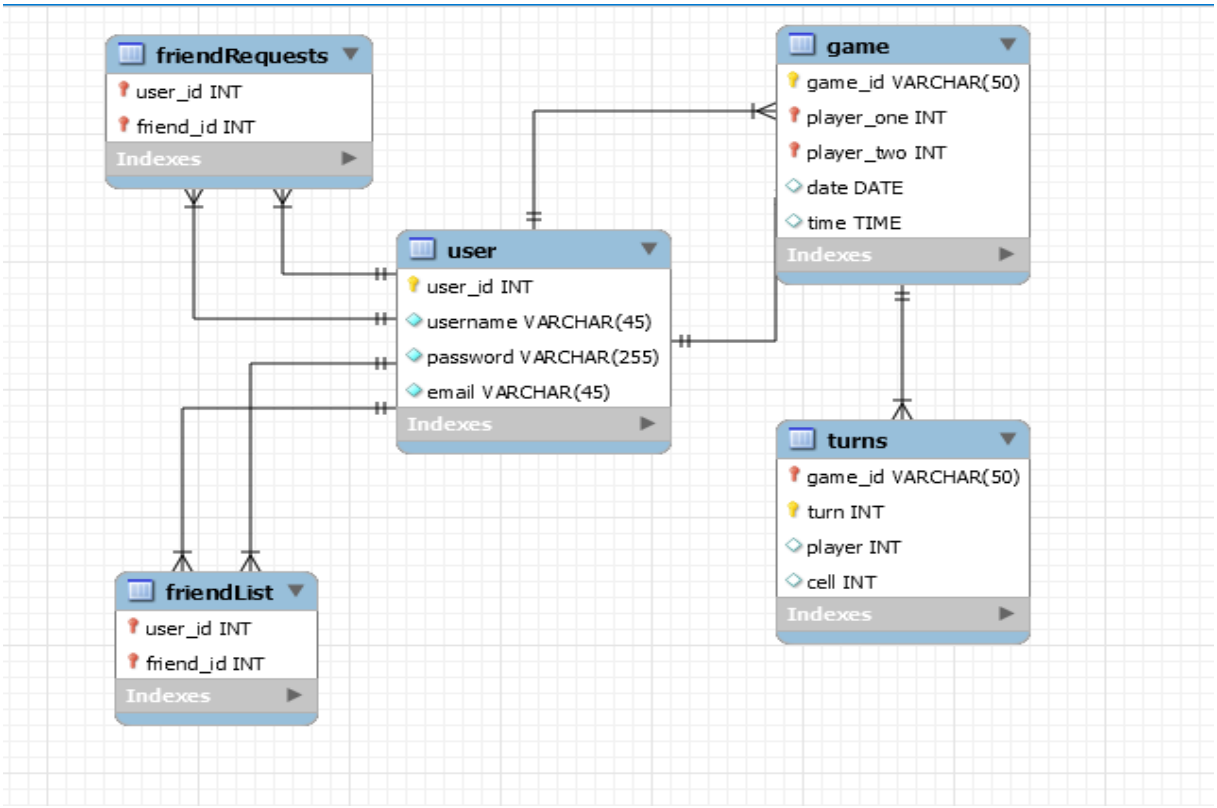
U ovom poglavlju biti će objašnjene tehnologije korištene za izradu sustava igre križić kružić za dva igrača. Tehnologije će biti objašnjene od dna prema vrhu (eng. *Bottom to Top*).

2.1. Baza podataka - MySQL

Temelj svake aplikacije tako i ove, je baza podataka (eng. *Database*). U bazu podataka pohranjuju se svi podaci potrebni za praćenje statistike igrača. Tako se na primjer u bazu pohranjuju korisnički računi igrača, igre, te detalji pojedine igre, kao što su potezi, pobjednik, vrijeme itd.

Za sustav je korištena baza podataka MySQL. Prema podacima sa en.wikipedia [1] program je nastao 1995. godine u Švedskoj pod firmom MySQL AB, napisan je u C/C++ programskom jeziku i osmišljen je kao sustav za upravljanjem nad relacijskom bazom podataka. MySQL je pod GPL licencom, odnosno moguće ju je besplatno koristiti.

U projektu je korišten MySQL sustav za upravljanjem nad bazom podataka. Slika 1. prikazuje ERA-model baze podataka.



Slika 1. ERA model baze podataka

U modelu se nalazi pet tablica. Glavna tablica je tablica 'user', u njoj se pohranjuju podaci igrača. Podaci su ID[1] korisnika, korisničko ime, email adresa te zaporka. 'friendRequests' i 'friendList' tablice su tablice koje se odnose na korisnikove prijatelje unutar igre. Unutar tablice 'friendRequest' se spremaju svi zahtjevi za prijateljstvom pojedinog igrača. Ako korisnik koji je zadobio zahtjev potvrdi prijateljstvo, igrač se dodaje u tablicu 'friendList' i briše iz tablice 'friendRequests'. Kada igrač potvrdi zahtijeva za prijateljstvom, tek tada igrači mogu međusobno igrati igru, a da ne moraju prolaziti kroz red čekanja.

Tablica 'game' se odnosi na jednu odigranu igru. Svaka igra koja se odigra se sprema u tablicu. Atributi su: ID igre, ID igrača jedan, ID igrača dva, vrijeme igre, ID pobjednika igre, zastavica koja označava dali je igra završila neriješeno te koliko je igra imala poteza. Zadnja tablica se odnosi na poteze unutar pojedine igre. Atributi su: ID igre, redni broj poteza, igrač koji je odigrao potez, te polje koje je odabrao igrač.

2.2. Poslužitelj - GoLang

Poslužitelj je odgovoran za upravljanjem tijeka igre te za prijenos podataka poteza jednog igrača drugom igraču. Svaki potez kojeg igrač napravi bilježi se te se prosljeđuje drugom igraču. Poslužitelj je također odgovoran za dohvaćanje potrebnih podataka, kao što su statistika, igračevi prijatelji, zahtjevi za prijateljstvom igrača te ono glavno prijava i registracija u sustav.

Za realizaciju poslužitelja korišten je GoLang programski jezik. Prema [2] početci izrade jezika su 2008. godina, dok je 2012. godine jezik službeno izbačen u javnost. Programski jezik se u početcima koristio pretežito u Google-ovim tehnologijama dok danas ima raširenu upotrebu.

Razlozi pisanja novog programskog jezika su ti da je bio potreban jezik koji bi bio:

- statički i lako proširiv na velike sustave (kao što su Java i C++)
- produktivan te lako čitljiv (Ruby i Python)
- jezik koji nema svoje integrirano razvojno okruženje[2] (eng *Integrated development environment - IDE*) (u nastavku IDE), ali se lako integrira u postojeće IDE-ove
- te da je namijenjen za mrežno programiranje te jednostavna višedretvenost

[1] ID – skračenica koja dolazi od Engleske riječi *Identifier*, Indetifikator
[2] IDE – tekstualni uređivač teksta koji pomaže pri pisanju programskog koda

Prema članku [3] sintaksa GoLang-a vrlo je slična sintaksi C/C++ programskog jezika, iz razloga što su zaposlenici Google-a morali što prije naučiti GoLang. To se može pročitati iz sljedećeg citata (talks.golang.org, 2012)

> It must be familiar, roughly C-like. Programmers working at Google are early in their careers and are most familiar with procedural languages, particularly from the C family. [2, paragraf 6]

„Prijevod bi bio: Jezik mora biti poznat, ponajviše kao C. Programeri koji rade u Google su u početcima svojih karijera i najviše su upoznati sa proceduralnim jezicima, ponajviše iz obitelji C programskih jezika, a to si C/C++.

Drugi citat iz istog poglavlja nastavlja razloge razvoja programskog jezika:

> „ It must be modern. C, C++, and to some extent Java are quite old, digned before the advet of multicore machines, networking, and web application development.“ [2, paragraf 6]

Prijevod citata nam daje do znanja da je programski jezik nastao zbog nedostataka C/C++, Java programskih jezika koji su relativno stari te nisu napisani dovoljno dobro da bi mogli u punom obujmu obavljati zadatke koji su potrebni u današnje vrijeme. Problemi koje jezik olakšava su izrada višedretvenih sustava, sustava na internetu te web aplikacija.

Kao što se može vidjeti, GoLang je nastao na principu tri popularna jezika, a to su Java od koje je preuzeto skaliranje na većim sustavima, C/C++ od kojih je preuzet proceduralni način izvođenja te Python od kojeg je preuzeta brzina izvođenja i sama sintaksa.

Na slici 2 može se vidjeti primjer ispisa „Hello world!“ koji se koristi kao primjer prvog programa napisanog u nekom programskom jeziku.

```
package main

import "fmt"

func main() {
    fmt.Println( a: "Hello world!")
}
```

Slika 2. Primjer ispisa 'Hello world!'

Package se odnosi isto kao i kod Jave, dijelovi projekta koji mogu biti korišteni unutar više projekata. U ovome primjeru to je package main, odnosno paket koji označava da je to glavni dio projekta. Zatim slijedi 'import „fmt“' linija koda. U toj liniji se u projekt uključuje paket pod imenom „fmt“. Unutar paketa se nalaze metode koje omogućuju ispis u konzolu. U primjeru se koristi metoda „Println“. Sličnu metodu možemo sresti u Javi gdje se koristi prilikom ispisa u konzolu. 'ln' na kraju naziva metode 'Println' stoji za to da se dodaje ASCII[3] znak 13 odnosno znaka '\n' koji označava novi red.

Funkcija se označava ključnom riječi func te naziv funkcije. Slika 3 prikazuje jednostavnu funkciju te tip podatka struct koja se nalazi unutar paketa 'primjer'.

```
package primjer

type Podatak struct {
    x int
}

func getPodatak() Podatak {
    return Podatak{}
}

func kvadriranje(x int) (int, int) {
    return x * x, x
}
```

Slika 3. Primjer funkcije i strukture

Na slici 3 vidi se da se datoteka nalazi u paketu 'primjer'. U primjeru je napisana jednostavna struktura[4] imena 'Podatak', koja sadrži atribut 'x' tipa integer. Funkcija 'getPodatak' vraća tip podatka 'Podatak' odnosno instancu[5] strukture. To se može vidjeti tako što je navedeno nakon argumenata funkcije te što je kod return-a jednostavan anoniman konstruktor strukture 'Podatak'. Druga funkcija je funkcija koja kvadrira proslijeđeni broj, te vraća kvadrat broja te sam broj koji smo proslijedili funkciji. Funkcije unutar GoLang-a imaju mogućnost vraćanja više vrijednosti.

[3] ASCII tablica – tablica koja služi za pretvaranje binarnog koda u čitljive znakove ljudske abecede.
[4] Struktura – složeni tip podataka, koji služi za spremanje više jednostavniji ili složenih tipova podataka
[5] Instanca – kopija strukture u koju se zapravo spremaju podaci

2.3. Klijent - Android

Igra kao glavni dio sustava se nalazi na Android platformi. Android je trenutačno najpopularnija mobilna platforma, iza koje slijedi iOS. Prema podacima [4] postotak Android uređaja u razdoblju od druge polovice 2017. godine do kraja prve polovice 2018. godine na tržištu iznosi 77.32%, dok iOS uređaja iznosi 19.4%.

Android operativni sustav je mobilna platforma čiji su početci 2004. godina. Prema riječima [5] izumitelja Andy Rubina glavna zamisao je bila da Android bude poboljšani operativni sustav za digitalne kamere. 2005. godine Google pokazuje interes za platformom te ju kupuje. Od tada pa do 2007. godine traje unaprijeđenije operativnog sustava. Operativni sustav izlazi u studenom 2007. godine pod verzijom Android 1.0. Verzija je bila beta[6] te je bila namijenjena developerima.

Zanimljiva činjenica je ta da su kodna imena za verzije Androida imenovane po slasticama. Prve dvije verzije nisu imale javna kodna imena. Kodna imena se počinju koristiti kod druge verzije Androida, točnije kod verzije 1.1 gdje se koristilo interno kodno ime „*Petit four*“ što je ime za francuski desert. Od toga trenutka pa na dalje sve nove verzije su dobile naziv po nekom desertu poredane abecednim redom. Od verzije 4.4. ili poznatije kao KitKat Google je potvrdio imenovanje verzija citatom:

> “Since these devices make our lives so sweet, each Android version is named after a dessert.” Ryan Gibson, 2013. [5, paragraf 3.]

„Pošto ti uređaji naš život čine slađim, svaka verzija Android sustava je nazvana po desertu.“ Od tuda se očituje da su sva kodna imena verzija imenovana s namjerom.

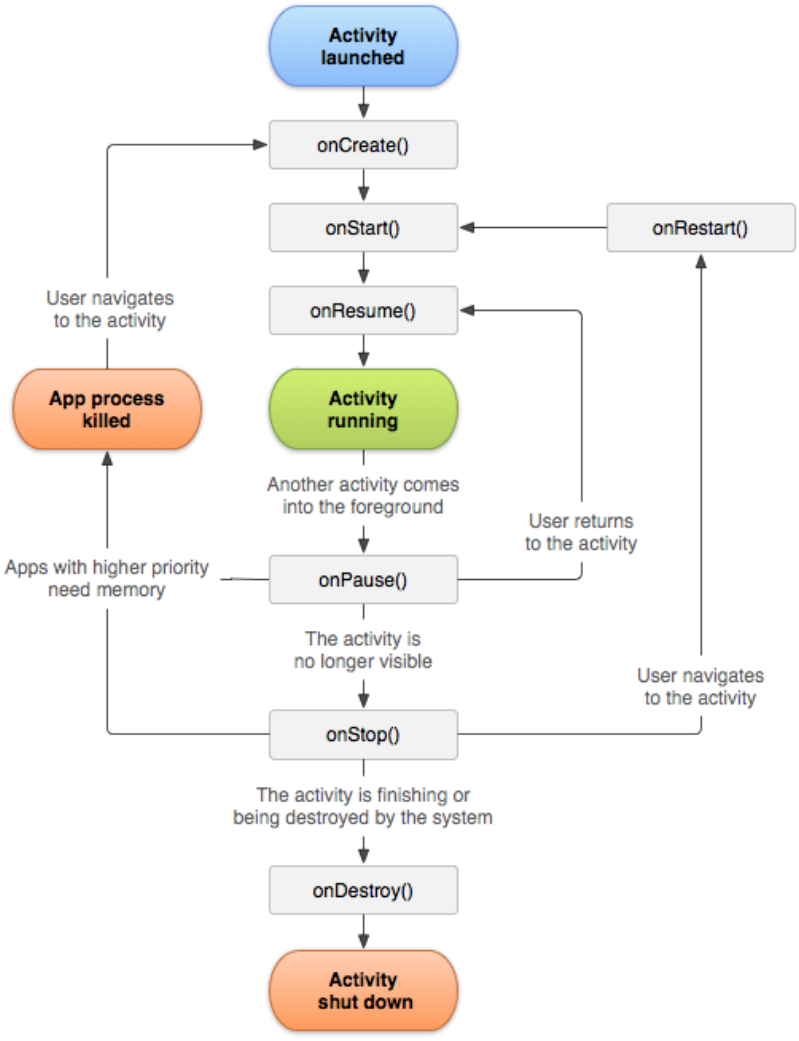
Android aplikacije su pisane u Java programskom jeziku, a od 2016. godine, izlaskom Kotlin programskog jezika na tržište moguće je raditi aplikacije i u Kotlin programskom jeziku. Kotlin je nastao od strane Google kompanije. Integrirano razvojno okruženje za izradu Android aplikacija je Android studio. IDE je napravljen po uzoru na IntelliJ IDEA.

Android studio od verzije 3.0.0 podržava i Kotlin programski jezik. Za izradu projekta potrebno je pokrenuti vodič koji korisniku olakšava pripremu projekta. Prilikom pokretanja novog projekta korisnik odabire package u kojemu će biti projekt, odabire ime projekta, po novome odabire dali želi podršku za Kotlin ili želi nastaviti raditi u Javi. Također mu se omogućuje odabir za koji uređaj želi razvijati. Na odabir ima phone, TV, wear, auto. Odabire

[6] Beta verzija – označava da je produkt u nestabilnoj fazi i da ima mane, ali se može koristiti

se i minimalna verzija koja je potrebna za pokretanje aplikacije. Preostale postavke olakšavaju početak izrade aplikacije. Kao što je na primjer početna aktivnost.

Svaka Android aplikacija se sastoji od više aktivnosti. Svaka aktivnost unutar aplikacije ima svoj životni vijek [6] Slika 4. Svaka aktivnost započinje pozivanjem metode onCreate(), koja se pokreće prilikom prvog pokretanja aktivnosti, nakon toga slijede metode onStart() i onResume(). OnStart() metoda se poziva kada uređaj ponovo uđe u aplikaciju, a da se prilikom izlaska iz aktivnosti aplikacija nije ugasila, već da je otišla u pozadinu. OnResume() se pokreće prilikom mijenjanja aktivnosti unutar aplikacije, a da pri tome nije pozivana metoda finish(), koja pokreće metodu onDestroy() te se time ta aktivnost u potpunosti zatvara. Prije metode onDestroy(), u životnom ciklusu postoje još dvije metode, a to su onPause() i onStop(). Metoda onPause() priprema aktivnost za metodu onResume(), dok metoda onStop() priprema za onStart(). Odnosno, te metode se pozivaju ako se aktivnost skriva od korisnika i kada korisnik ponovo pokreće aktivnost, pozivaju se jedna on metoda, onStart() ili onResume().



Slika 4 životni ciklus aktivnosti (Izvor: developer.android.com, 2018)

2.4. Igra za više igrača i algoritam križić kružić igre

Igra za više igrača ili popularno zvano po engleskoj riječi multiplayer igre, su igre u kojima igrač igra na svojoj kopiji igre, dok se njegovi potezi prosljeđuju drugim igračima pomoću poslužitelja, koji provjerava poteze i tako kontrolira tijek igre odnosno pravila igre. Iz tog razloga igrači imaju osjećaj da igraju protiv drugih igrača izravno na njihovim uređajima, dok su zapravo na udaljenim lokacijama.

U današnje vrijeme malo računalnih igara ima samo opciju samostalnog igranja, odnosno da igra protiv računala, dapače veći dio igara je orijentirano na komponenti igre više igrača te se igra ne može igrati sve dok se igrač ne spoji na poslužitelja.

2.4.1. Komunikacija između igrača

Gore je već spomenut način komunikacije prilikom odigravanja igre, a u nastavku će se detaljnije pojasniti ta komunikacija.

Prilikom mrežnog prometa [7] slajd 66 kaže da postoji 5 slojeva, koji su redom:

- aplikacijski sloj,
- transportni sloj,
- mrežni sloj,
- poveznički sloj,
- te fizički sloj.

Igra se smatra aplikacijskim slojem mreže iz razloga što aplikacija šalje i prima podatke potrebne za igru. Prilikom tog procesa aplikacija potrebne podatke priprema za slanje odnosno čitanje i zbog tih operacija aplikacija spada pod aplikacijski sloj. Nakon što se podaci pripreme za slanje šalju se na transportni sloj, odnosno prilikom primanja poruka aplikacijski sloj preuzima poruke od transportnog sloja.

Unutar transportnog sloja postoje dva protokola koja se koriste u velikoj većini mrežnog prometa. A to su TCP i UDP protokoli. Glavne razlike [8] između ta dva protokola su TCP protokol koji stvara „tunel“ između klijenta i poslužitelja te se jedna poruka šalje za drugom, dok kod UDP-a klijent i poslužitelj nemaju stalnu vezu, odnosno šalju se paketi koji su adresirani na adresu poslužitelja. Prilikom slanja TCP-om podaci će sigurno stići na odredište, što kod UDP-a nije slučaj. Pošto UDP samo šalje podatke, ne postoji garancija da je paket stigao na odredište, već da se izgubio u procesu slanja.

Uz prethodno opisanu razliku dvaju protokola, za komunikaciju unutar igara za više igrača odabire se UDP protokol. Razlog tome je što prilikom uspostave veze TCP protokolom potrebno je poslati 4 poruke, dok kod UDP-a potrebna je samo jedna, i to sa podacima. Samim time se smanjuje kašnjenje između podataka te ako je igra složenija potrebno je slanje više podataka i tada je UDP bolji izbor, iz razloga što se paketi šalju na adresu, dok bi se kod TCP-a svaki novi potez igrača morala ponovo raditi provjera je li paket stigao.

2.4.2. Algoritam igre križić kružić

Igra križić kružić [9] je veoma jednostavna. Algoritam se svodi na to da dva igrača međusobno na polje koje je veličine tri puta tri postavljaju križić odnosno kružić, ovisno koji je znak koji igrač odabrao na početku igre. Cilj igre je da se spoje tri ista znaka u niz, točnije ako se tri znaka mogu povezati jednom ravnom linijom igrač je pobijedio. Ako se popuni cijela ploča, odnosno svih 9 polja i nema pobjednika, igra se proglašava neriješenom.

Prema pravilima možemo napisati algoritam. Pošto na ploči dimenzija 3 puta 3 možemo uočiti da postoji samo 8 mogućih kombinacija koje donose pobjedu. Glavni cilj igre je da znak bude 3 puta uzastopno jedan za drugim. Da igrač pobijedi potrebne su jedne od sljedećih kombinacija: popunjen prvi, drugi, treći red, odnosno prvi, drugi ili treći stupac, ili popunjene dijagonale ploče. To su svih 8 mogućih ishoda pobijede. Stoga možemo početi provjeru od gornjeg lijevog kuta ploče. I krećemo se desno po ploči, ako na polju nije isti znak ili je prazno, možemo stati i ispitati drugu mogućnost, ako je polje popunjeno istim znakom prelazimo na treće polje. Ako je to polje isto popunjeno istim znakom možemo proglasiti pobjednika, a ako nije vraćamo se dalje na provjeru.

Sljedeća provjera može biti dijagonala od gornjeg lijevog polja prema donjem desnom polju, ako ni to nije pobjednička kombinacija provjeravamo prvi stupac, sljedeći korak u slučaju da nema pobijede provjera se seli na drugi stupac, te na treći, ako ni tada nema provjerava se dijagonala gornje desno polje, na donje lijevo polje. Ako i dalje nema pobijede, provjeravaju se preostala dva retka na ploči. Provjera se nema potrebe provjeravati sve do petog poteza jer prije toga nije moguće povezati tri ista znaka. I ako je na ploči popunjeno 9 polja i nema pobjednika igra se zatvara te se proglašava neriješena igra.

3. Razrada teme

Tema rada je bila izrada sustava igre za više igrača. Kao što su u prethodnom poglavlju na grubo pojašnjene tehnologije i način izrade sustava, u ovom poglavlju ćemo pobliže pojasniti način izvedbe igre, od komunikacije pa do mogućnosti same Android aplikacije.

Komunikacija između poslužitelja i klijenta se odvija putem dva kanala, a to su HTTP[7] i UDP protokoli. HTTP je zaslužan za statičke podatke kao što su podaci o igraču, statistika, lista prijatelja i lista zahtijeva za prijateljstvom, dok se preko UDP protokola odvija sama igra, te pripreme za igru, kao što su spajanje dvaju igrača (eng. Matchmaking) u slučaju slobodne igre ili ako se želi igrati igra protiv prijatelja, te kada dva igrača žele ponovo igrati jedan protiv drugoga, kada igra završi.

Prilikom HTTP komunikacije sa poslužiteljem potrebno je u zaglavlju svakog zahtijeva imati autentifikacijski ključ, u protivnom poslužitelj vraća poruku o nedozvoljenom pristupu.

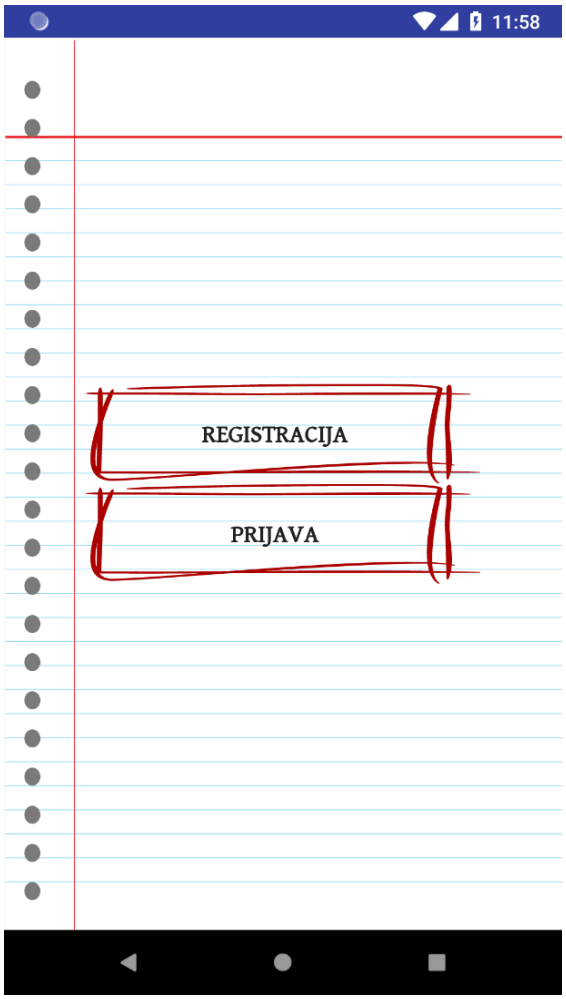
Svaka krajnja točka (eng. Endpoint) sadrži provjeru autentifikacijskog ključa, sadrži provjeru dali su u zahtijeva poslani podaci te provjera primljenih podataka. Te sve provjere su potrebne prije pristupanja bazi podataka. Prilikom svake greške klijentu se šalje odgovarajuća povratna informacija u obliku JSON[8]-a.

U slučaju da su sve provjere provjerene, slijedi spajanje na bazu podataka, vrši se upit te se provjerava uspješnost upita. Ako se prilikom toga dogodi pogreška, klijentu se također vraća JSON sa povratnom informacijom. Sljedeći korak je obrada podataka te spremanje u oblik za slanje, točnije u JSON. Ako se okine koja pogreška prilikom obrade podataka klijentu se vraća informacija o pogrešci. Ako se do ovoga trenutka nije okinula ni jedna od provjera, korisniku se šalju zahtjevni podaci. Popis svih krajnjih točaka nalazi se u prilogu ovog dokumenta.

[7] HTTP engleska skračenica od HyperTextTransferProtokol, protokol koji služi za slanje podataka putem interneta

[8] JSON engleski JavaScriptObjectNotation podatkovni format koji služi za jednostavan prijnos podataka putem interneta, JSON objekt je objekt prikazan u obliku String-a

3.1. Glavni izbornik - neprijavljeni korisnik

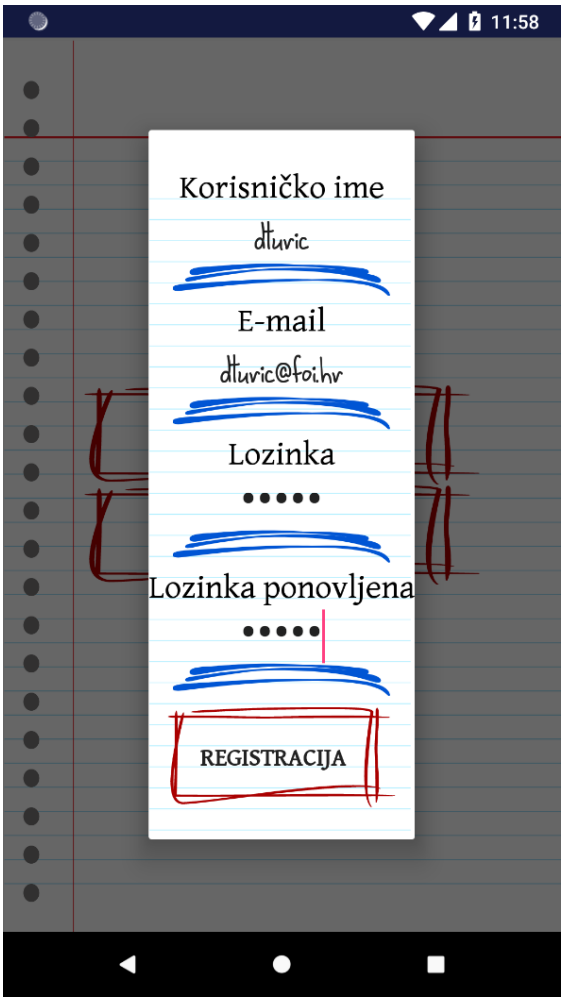


Slika 5 Prikaz početnog zaslona-neprijavljeni korisnik

Na slici 5 vidi se početni zaslon aplikacije koji se otvara tek dok je provedena provjera je li uređaj spojen na internet i je li se uspio povezati sa poslužiteljem. Taj zaslon se prikazuje na uređaju na kojemu prije toga nije bilo prijavljenog korisnika, odnosno ako se korisnik odjavio iz aplikacije.

Zaslon je vrlo jednostavan. Postoje dva gumba koji su imenovani „Registracija“ i „Prijava“. U jednom i drugom slučaju otvaraju se dijalog prozori te omogućuju korisniku unos potrebnih podataka. Tako na primjer ako korisnik pritisne „Registraciju“ otvara se dijalog prozor sa formom za prijavu, a kod „Prijave“ dijalog sa formom za prijavu korisnika.

3.1.1. Registracija



Slika 6 Prikaz dijalog prozora registracije

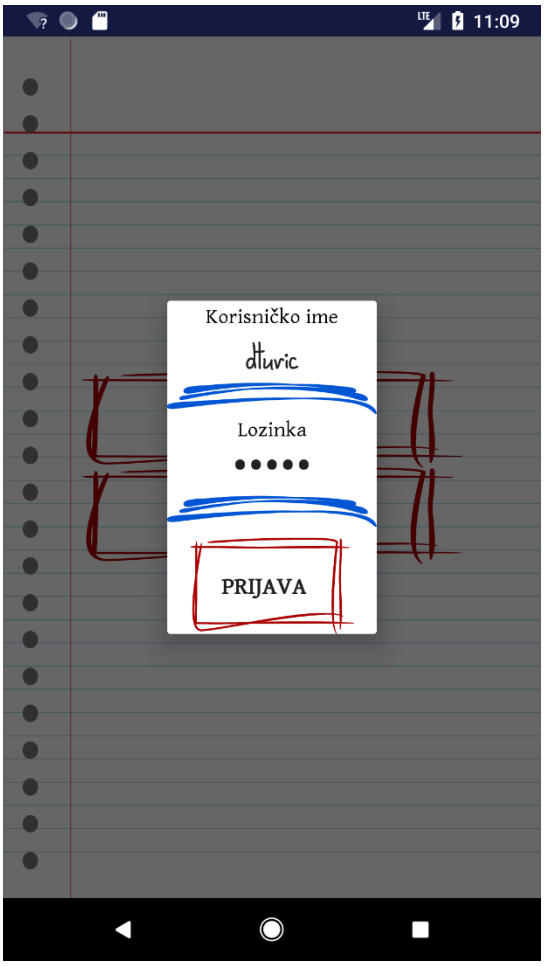
Slika 6 prikazuje dijalog prozor koji se otvara pritiskom na gumb prijava. Trenutno je na slici prikazana forma kada je popunjena podacima. U formu se unose standardni podaci koji su potrebni, a to su:

- Korisničko ime igrača,
- E-mail adresa,
- Lozinka kojim će se prijavljivati u sustav,
- te ponovljena lozinka, da se smanji mogućnost prijave “botova”.

Uneseni podaci se šalju na poslužitelja POST metodom. Endpoint nalazi se na putu „/register“. U slučaju pozitivne povratne poruke od strane poslužitelja korisnik se preusmjerava na glavni izbornik za prijavljene korisnike te se tretira kao prijavljeni korisnik. Ukoliko dođe do greške, korisnik ostaje na dijalogu registracije te mu se ispisuje greška.

U aplikaciji je zamišljeno da tekstualni dijelovi koji su statički budu fonta kao u radnim bilježnicama, dok dio teksta kojeg unosi korisnik bude fonta sličnog rukopisu. Na dnu imaju plavu šaru koja inducira da se tu može unijeti tekst, dok su gumbovi i svi dijelovi koji se trebaju pritisnuti okruženi crvenom šarom.

3.1.2 Prijava



Slika 7 Prikaz dijalog prozora prijave

Slika 7 prikazuje dijalog prozor prijave, odnosno forme za prijavu u aplikaciju. Prozor je vrlo jednostavan. Podaci koji se unose prilikom prijava su:

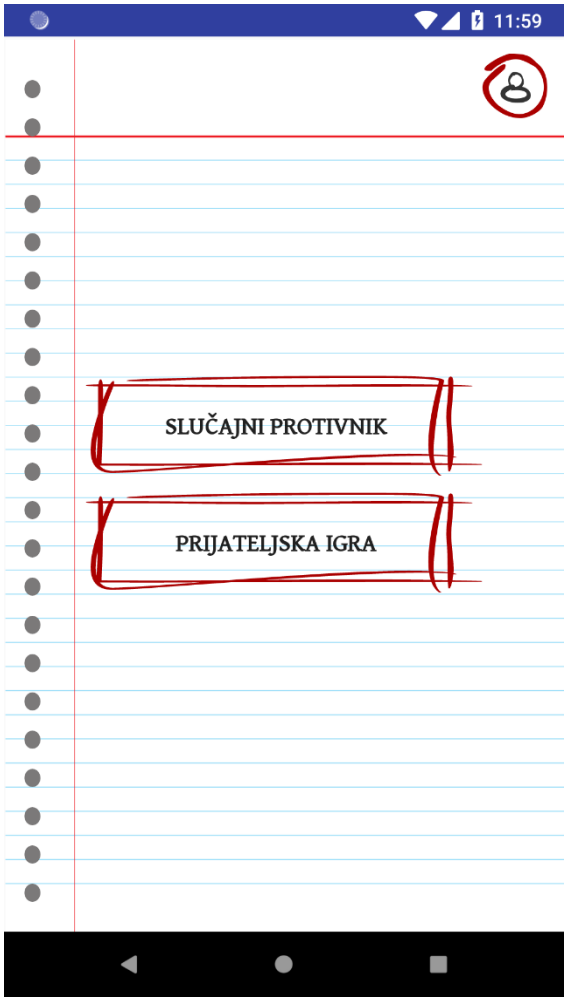
- Korisničko ime,
- Te lozinka koja je registrirana prilikom registracije na korisnika s unesenim korisničkim imenom

Prijava se odvija na način da nakon što korisnik unese potrebne podatke i pritisne gumb „Prijava“, podaci se šalju na endpoint „/login“, na kojem se provjeravaju uneseni podaci te u

slučaju ispravnih podataka korisnik se prijavljuje i preusmjerava na glavni izbornik prijavljenih korisnika. U slučaju netočnih podataka odnosno pogreške ispisuje se pogreška.

Dizajn je isti kao i prilikom registracije. Statički tekst ima font sličan radnim bilježnicama, uneseni tekst sliči rukopisu i gumb ima crveni obrub.

3.2. Glavni izbornik, prijavljeni korisnik



Slika 8 Prikaz glavnog izbornika-prijavljeni korisnik

Slika 8 prikazuje zaslon prijavljenog korisnika. Glavna razlika između zaslona prijavljenog korisnika i ne prijavljenog je ta da na zaslonu prijavljenog korisnika u gornjem desnom kutu postoji ikona koja prilikom klika na nju vodi na profil korisnika. Ostatak zaslona se razlikuje po tipkama, odnosno njihovim funkcijama.

Tako postoje dva gumba, prvi se zove „Slučajni protivnik“ koji igrača šalje u red čekanja za igru te „Prijateljska igra“ u kojoj korisnik odabire prijatelja sa svoje liste prijatelja te šalje zahtijeva za igrom.

Prilikom prikaza zaslona glavnog izbornika-prijavljeni korisnik, u pozadini se pokreće servis koji sluša zahtijeva za prijateljskom igrom. Servis radi tako dugo dok korisnik ne dobije zahtijeva za igrom od drugog korisnika ili dok korisnik ne ukloni zaslon s ekrana uređaja. Drugim riječima, ako korisnik uđe u red čekanja za slučajnom igrom, servis se gasi. Ako korisnik izađe iz aplikacije ili ju ugasi, servis se gasi.

Prilikom pozivanja servisa, prva poruka koja se šalje poslužitelju je poruka sa statusom „Register“. Tom porukom klijent daje do znanja da je korisnik spreman prihvatiti prijateljsku igru, odnosno da je u glavnom izborniku. U slučaju kada se gasi servis, tada servis šalje poruku sa statusom „Unregister“. Tada klijent više nije u mogućnosti dobiti zahtijeva za prijateljskom igrom, dok poslužitelj igrača briše iz reda igrača koji čekaju zahtijeva za prijateljskom igrom. Servis komunicira na portu 1500.

3.2.1. Slučajna igra

Kao što samo ime govori, slučajna igra je način igre gdje korisnik nasumično traži igrača koji također traži igru. To je što korisnik vidi, dok se na poslužitelju odvija algoritam spajanja dva igrača koji su kompatibilni za igru.

```
func matchMakingAlgorithm(con *net.UDPConn) {
    for {
        if queueNumberOfPlayers >= 2 {

            fmt.Println("Connected: " + queue[0].address.String() + " AND " + queue[1].address.String())
            var gameData sendGameData
            gameData.GameId = gameIdGenerator( n: 20)
            gameData.PlayerTwo = string(queue[0].userId)
            gameData.PlayerOne = string(queue[1].userId)
            decoded, err := json.Marshal(gameData)

            if err != nil {
                panic(err)
                return
            }

            con.WriteTo([]byte(decoded), queue[0].address)
            con.WriteTo([]byte(decoded), queue[1].address)

            queue[0].inGame = true
            queue[1].inGame = true
            playerOne, queue := queue[0], queue[1:]
            playerTwo, queue := queue[0], queue[1:]

            activeGames = append(activeGames, game{gameId: gameData.GameId, playerOne: playerOne, playerTwo: playerTwo})
            playerOne = player{}
            playerTwo = player{}
            queueNumberOfPlayers = queueNumberOfPlayers - 2

            fmt.Println(queue)
        } else {
            time.Sleep(time.Second)
        }
```

Slika 9. Algoritam spajanja dva igrača

Slika 9 prikazuje algoritam. Algoritam se izvodi u zasebnoj dretvi, koja se pokreće s pokretanjem poslužitelja. Algoritam se nalazi unutar beskonačne for petlje iz razloga što nakon što spoji dva igrača treba biti u mogućnosti spojiti druga dva igrača i tako sve dok poslužitelj radi. Algoritam ima dva stanja, u slučaju da u redu čekanja ima dva ili više igrača ili da je u redu čekanja manje od dva igrača. U slučaju da nema dovoljno igrača za igru odnosno da je u redu čekanja manje od dva igrača, tada algoritam „spava“ sekundu, do sljedeće provjere broja igrača u redu čekanja. U slučaju da u redu čekanja postoje dva ili više igrača, tada algoritam ispisuje poruku koja dva igrača je spojio, točnije njihove adrese. Sljedeći korak je stvaranje instance strukture sendGameData u koju se upisuje id nove igre, pseudo slučajni String od 20 znakova te ID-evi od igrača. Ti podaci se pretvaraju u JSON te se šalju igračima. Nakon što su se igrači obavijestili o igri, postavljaju se zastavice „inGame“ na true, koje služe da poslužitelj zna da su maknuti igrači iz reda čekanja, prilikom provjere aktivnosti igrača. Korisnici se brišu iz reda čekanja te se prebacuju u red aktivnih igri.

U slučaju ovoga rada, algoritam spajanja igrača je veoma jednostavan. Napisan je na način da se spajaju prva dva igrača unutar reda čekanja. Algoritam se jednostavno proširi na način da se u igru uvede način bodovanja, kao što je u šahu ELO[9]. Te se preko sustava bodovanja traže dva igrača koji imaju sličnu razinu vještina.

Slika 10 prikazuje komunikaciju klijenta i poslužitelja. Unutar aplikacije pritiskom gumba „Slučajna igra“ preko HTTP-a i end pointa „/getInTheQuery“ korisnik se prijavljuje u red čekanja za igru. Ako je provjera podataka izvršena i ako je sve uredu, korisnik se prebacuje na UDP konekciju na portu 1000.

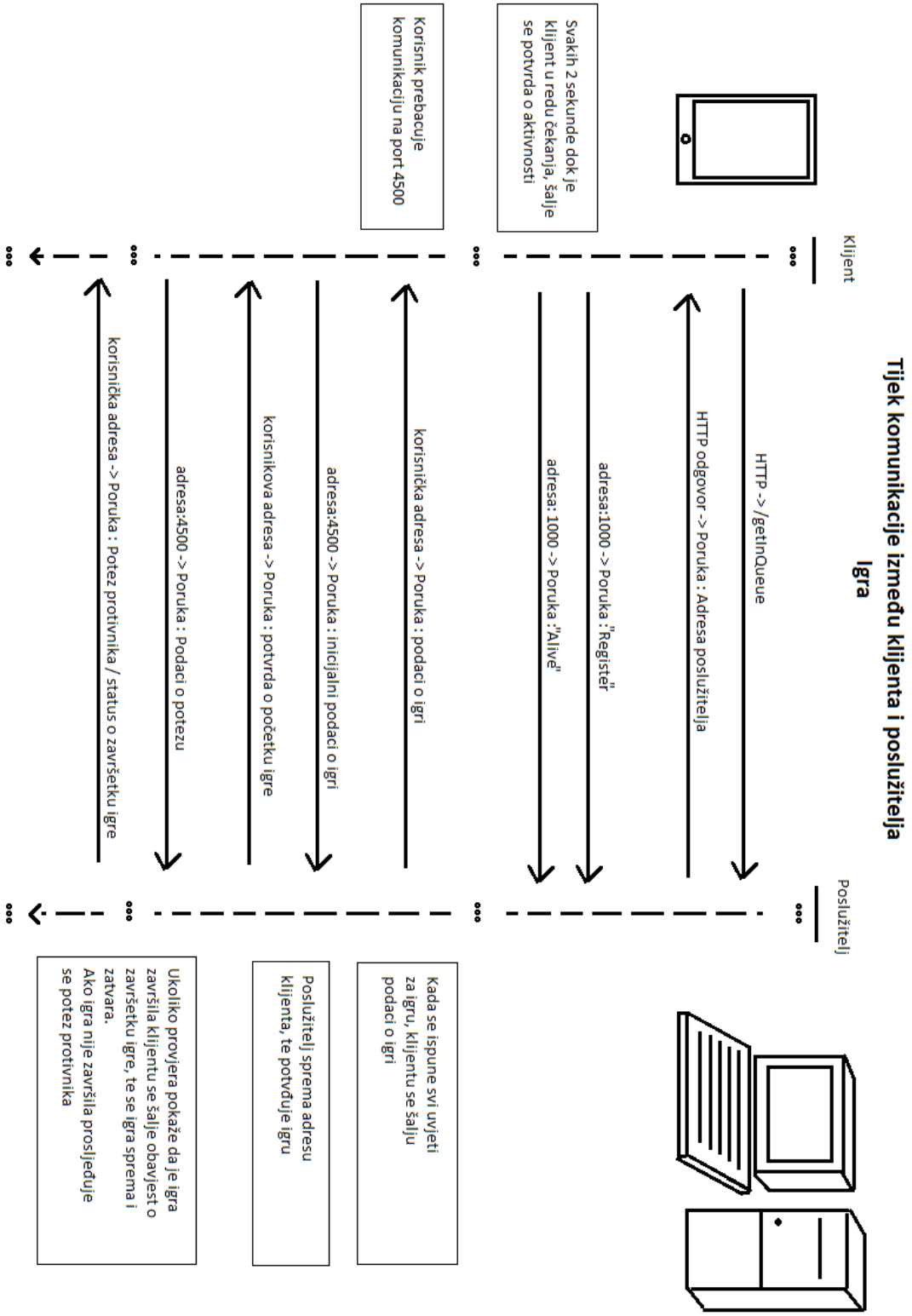
Tijekom boravka u redu čekanja, korisnik svakih 2 sekunde šalje informaciju serveru da je i dalje aktivan, dok na poslužitelju postoji dretva koja provjerava je li se korisnik prijavio da je i dalje aktivan ili nije. Ta provjera služi da se u slučaju neaktivnosti korisnika, korisnik makne iz reda čekanja te da drugi igrači dobiju priliku zaigrati.

Algoritam na klijentu napisan je na način da je konekcija s poslužiteljem podijeljena na dvije dretve. Jedna dretva šalje podatke, kao što su registracija i potvrda o aktivnosti, dok druga dretva sluša novosti o novoj igri. Obje dretve su povezane na jednu konekciju, samo što je zadatak podijeljen na slanje i slušanje.

Komunikacija unutar aplikacije se odvija u sljedeće tri datoteke.

- UDPConnection.java,
- GameListener.java,
- Te ConnectionUpdater.java.

[9] ELO – metoda bodovanja igračevih vještina, metoda se najčešće koristi u šahu

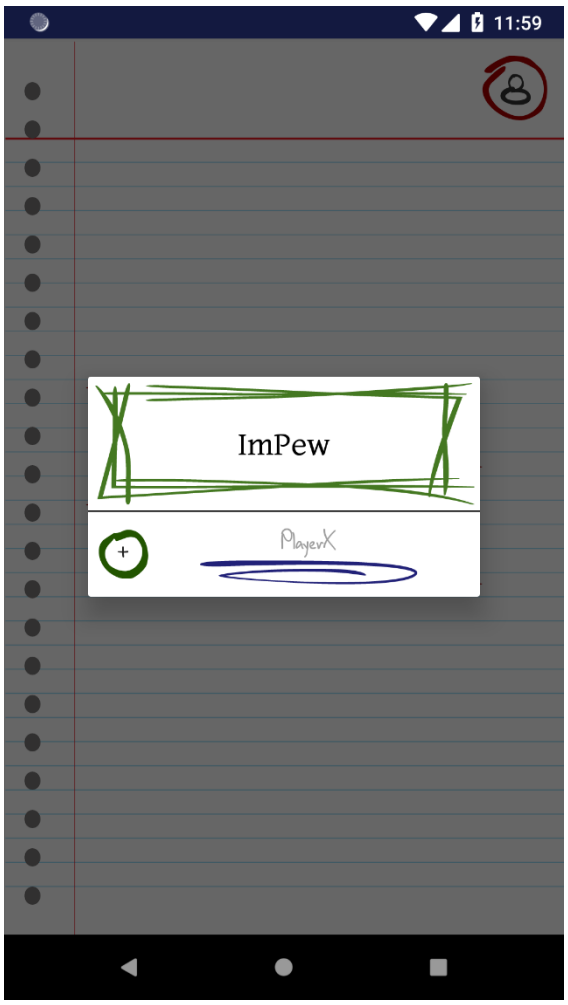


Slika 10 Dijagram komunikacije

3.2.2. Prijateljska igra

Kao što je gore već pojašnjen način spajanja igrača u prijateljsku igru, u ovom poglavlju će cijeli proces biti detaljnije pojašnjen.

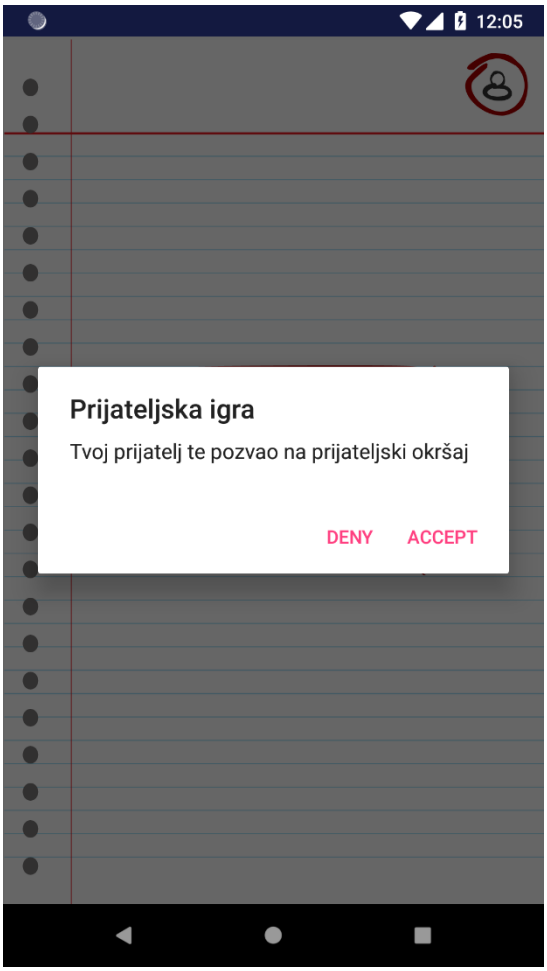
Pritiskom na gumb prijateljska igra, otvara se dijalog prozor na kojem se nalazi lista prijatelja te na dnu postoji mogućnost dodavanja novog prijatelja preko korisničkog imena.



Slika 11 Prikaz liste prijatelja

Na slici 11, prikazan je dijalog prozor nakon što je pritisnu gumb „Prijateljska igra“. Zeleni obrub imaju stavke liste, odnosno jedan od prijatelja igrača. Ispod liste nalazi se dio koji omogućuje dodavanja prijatelja na listu, točnije slanja zahtjeva navedenom igraču.

Nakon što korisnik odabere prijatelja, prijatelju se pojavi prozor kao što je na slici 12. Ukoliko korisnik nije aktivan ili odbije zahtjev servis se ponovo pokreće te igrači ponovo slušaju zahtjeve za prijateljskim okršajem.

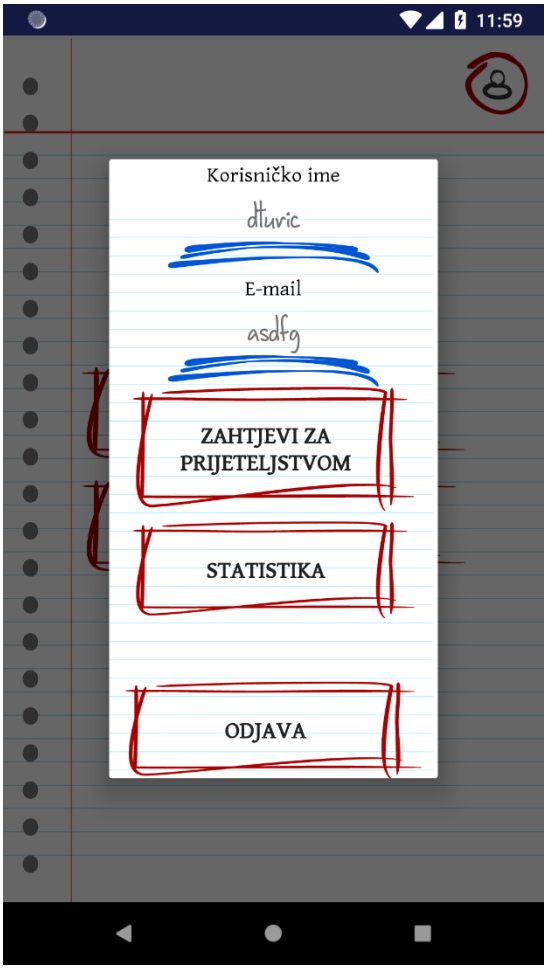


Slika 12 Prikaz skočnog prozora sa zahtjevom za prijateljskim okršajem

Prozor izgleda veoma jednostavno. Na njemu su dva gumba koja potvrđuju zahtjev ili ga odbijaju. Ako se odbije, prozor se jednostavno zatvara te se na poslužitelja šalje poruka sa statusom „Deny“, a u slušaju prihvaćanja šalje se status „Accept“ te se oba igrača preusmjeravaju na zaslon na kojemu je ispisana ploča, odosno zaslon same igre.

3.2.3. Korisnički Profil

Do korisničkog profila se dolazi na jednostavan način, pritiskom na gumb na kojem je zaokruženo poprsje ikone čovjeka. Taj gumb se nalazi u gornjem desnom kutu na zaslonu prijavljenog korisnika. Otvaranjem profila prikazuje se zaslon sa slike 13.

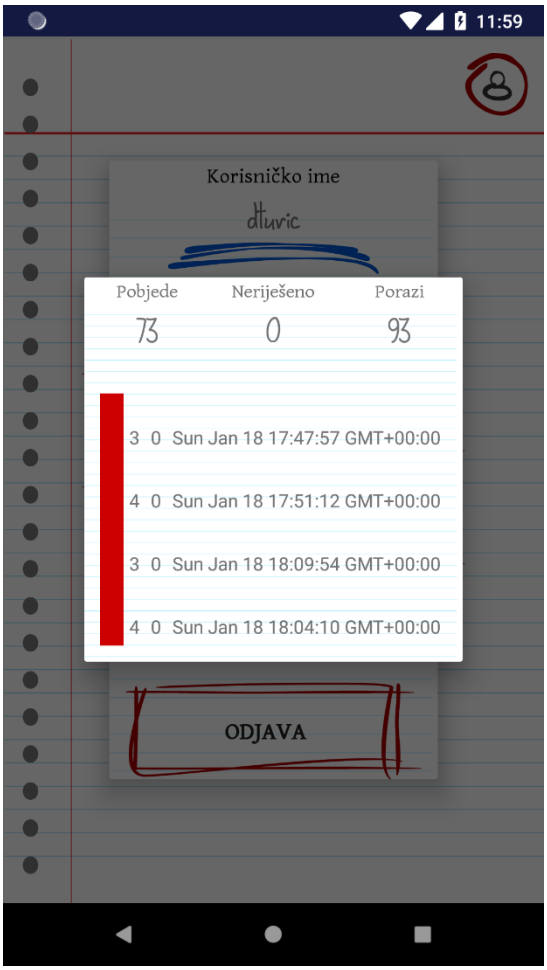


Slika 13 Prikaz profila korisnika

Slika 13 prikazuje izgled profila igrača. Profil je vrlo jednostavan. Korisniku se ispisuju korisničko ime i e-mail adresa s kojom se registrirao te ima gumbe koji mu omogućuju prikaz zahtijeva za prijateljstvom koje su mu slali drugi igrači. Prilikom odabira statistike korisniku se otvara dijalog prozor kao na slici 12 te mu se prikazuje broj pobijeđenih, neriješenih te izgubljenih dvoboja i lista posljednjih pet igara.

Lista zahtijeva za prijateljstvom se dohvaća preko end pointa koji se nalazi na „/getFriendRequests“. Zahtijev se šalje HTTP POST metodom. Parametar koji se prosljeđuje poslužitelju je ID igrača. Ukoliko je zahtjev uspješan odgovor poslužitelja je lista zahtijeva za prijateljstvom unutar igre.

Statistika se također dohvaća HTTP POST metodom. Parametar koji se šalje poslužitelju je ID korisnika za koji se želi prikazati statistika. Prilikom uspješne obrade podataka, povratna poruka je broj pobjeda, poraza i neriješenih igara te lista sa posljednjih pet igara.

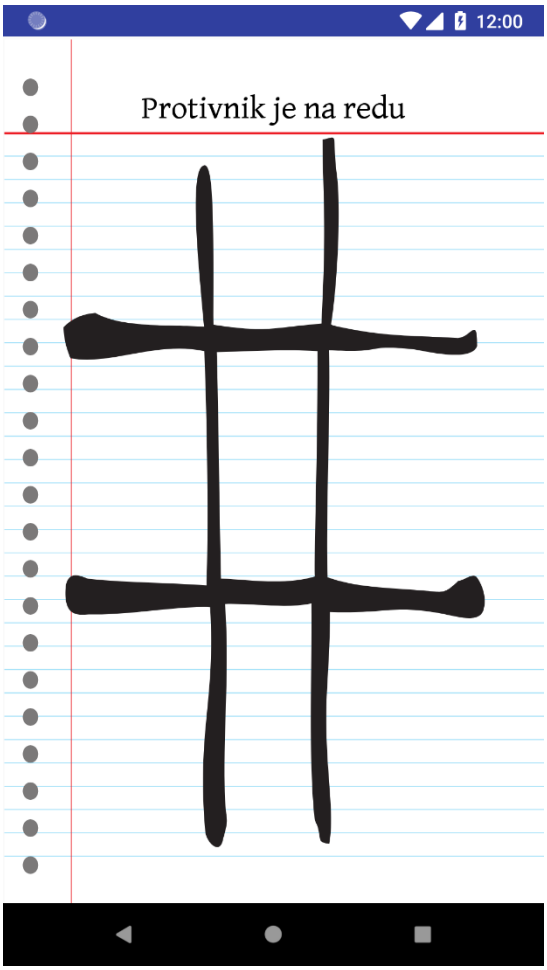


Slika 14 Prikaz statistike

Slika 14 prikazuje profil igrača. Na vrhu su ispisane brojčano pobijede, porazi i neriješeni dvoboji. Ispod toga se nalazi lista s podacima posljednjih pet igara. Ukoliko je igrač izgubio igru, indikator je crvene boje, u slučaju pobjede je zelene boje te sive ako je dvoboj odigran neriješeno.

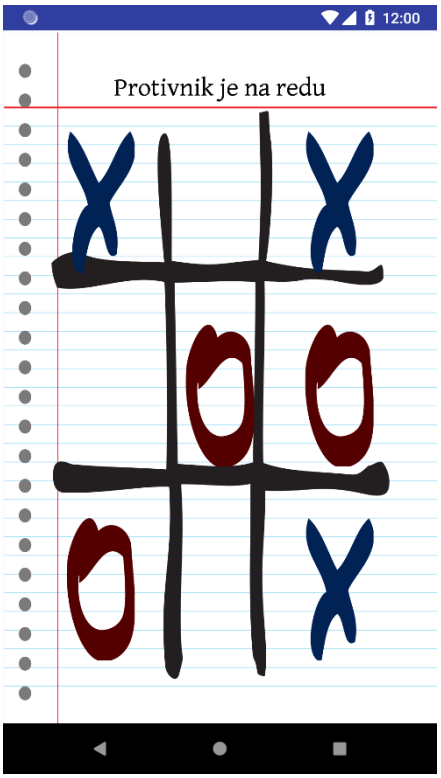
3.3. Igra

Igra, odnosno zaslon koji se prikazuje prilikom igre je prikazan na slici 15. Kao što je već opisano u poglavlju 3.2.1. slika 10, igrač se preko HTTP zahtjeva prijavljuje u red čekanja te se komunikacija između klijenta i poslužitelja odvija preko UDP konekcije. Konekcija se odvija preko porta 1000.

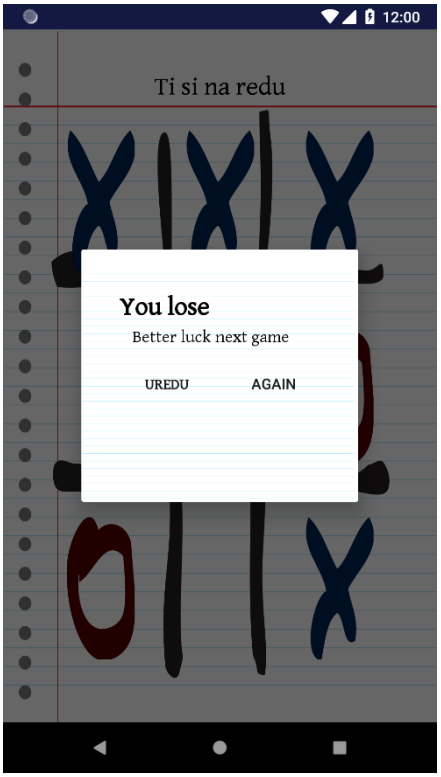


Slika 15 Prikaz ploče za igru

Igra se igra na način da se pričeka igračev red. Kada je red na igraču jednostavno pritisne polje na koje želi postaviti svoj znak. Prilikom pritiska na polje postavlja se znak, križić ili kružić, slika 16, te se poslužitelju šalju podaci o potezu. Poslužitelj te podatke sprema u bazu te ih prosljeđuje protivniku. Također sama provjera igre se vrši na poslužitelju. Ukoliko prilikom provjere dođe do pobijede odnosno poraza drugog igrača, igračima se šalje informacija o pobijedi, odnosno porazu. I tada se na korisnikovom ekranu prikazuje dijalog na kojemu je moguće odabrati ponovnu igru ili povratak na glavni izborni, slika 17.



Slika 17 Prikaz dijaloga o završetku igre



Slika 16 Prikaz aktivne igre

4. Zaključak

Zaključak ovog rada je funkcionalan poslužitelj igre za više igrača te prateća aplikacija na Android operativnom sustavu. Korištene tehnologije su opcionalne, cijeli sustav se mogao napisati u bilo kojoj drugoj tehnologiji. Glavna je sama logika i raspored komunikacije između klijenta i korisnika.

Algoritam i komunikacija korištena prilikom izrade ovog sustava može se jednostavno upotrijebiti za izradu poslužitelja za igre koje imaju više od dva igrača u jednoj igri. Sve što je potrebno je izmijeniti poslužitelja te napraviti objekt koji sadrži podatke o svim igračima unutar aktivne igre. I najvažniji dio, proširiti logiku sustava, jer što je više igrača u igri to je komunikacija kompleksnija, odnosno svi igrači moraju slati svoje poteze te primati poteze ostalih igrača unutar aktivne igre.

Prilikom izrade sustava igre bilo je potrebno osnovno znanje baza podataka, mrežnih tehnologija te programiranja na strani klijenta i poslužitelja. Potrebno je bilo proučiti moguće načine komunikacije te odabrati odgovarajuće. Također potrebno je i optimizirati kod koliko je moguće i prilagoditi ga na veliki broj mogućih korisnika, koji bi se istodobno služili sustavom.

Sljedeći korak u izgradnji sustava, odnosno nadogradnji, bilo bi testiranje performansi sustava, maksimalnom broju korisnika koje sustav može posluživati u jednom trenutku. Ali za to je potrebno mnogo veće testiranje sustava nego li je učinjeno prilikom trenutne izrade.

Kao što je u samome uvodu rečeno, autor je želio vidjeti što se događa „iza kulisa“ u sustavima igara više igrača, što je postignuto ovim radom.

5. Popis literature

[1] Wikipedia En, MySQL [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/MySQL [Pristupljeno: 10-kol-2018]

[2] Wikipedia En, Go (Programing Language) [Na internetu]. Dostupno na: https://en.wikipedia.org/wiki/Go_(programming_language) [Pristupljeno: 10-kol-2018]

[3] Talks GoLang.org, Go at Google: Language Design in the Service of Software Engineering, 6. Enter GO [Na internetu]. Dostupno na: https://talks.golang.org/2012/splash.article [Pristupljeno 10-kol-2018]

[4] StatCounter, OS Market Share , Mobile Operating System Market Share Worldwide, [Na internetu]. Dostupno na: http://gs.statcounter.com/os-market-share/mobile/worldwide [Pristupljeo 10-kol-2018]

[5] Android Authority, The history of Android OS: its name, origin and more, Preparing for the launch of Android 1.0 [Na internetu]. Dostupno na: https://www.androidauthority.com/history-android-os-name-789433/ [Pristupljeno na: 10-kol-2018]

[6] Developer Android, Activity [Na iternetu] Dostupno na: https://developer.android.com/reference/android/app/Activity [Pristupljeno 10 - kol - 2018]

[7] Moodle arhive 2016./17., Uvod u mreže-2016., Slojevi protokola, slajd 66. [Na moodle-e sustavu]. Dostupno na: https://elfarchive1617.foi.hr/pluginfile.php/56521/mod_resource/content/9/1.%20Uvod_2016.pdf [Pristupljeno 10-kol-2018]

[8] Diffen, TCP vs. UDP, Comparison chart [Na internetu]. Dostupno na: https://www.diffen.com/difference/TCP_vs_UDP [Pristupljeno 10-kol-2018]

[9] a.En, TicTacToe [Na internetu] Dostupno na: HYPERLINK "https://en.wikipedia.org/wiki/Tic-tac-toe"https://en.wikipedia.org/wiki/Tic-tac-toe [Pristupljeno10-kol-2018.][Pristupljeno10-kol-2018.]

6. Popis slika

7. Prilozi

Prilog 1. Game API documentation\Dokumentacija Web servisa

This is documentation of game HTTP API-s
All end point returns JSON formated data

U nastavku se nalaze svi dostupni webservisi te njihovi opisi
Sve krajnje točke vraćaju poruke u JSON obliku

Request - header data

Header name\Naziv zaglavlja: auth
Header value\Vrijednost zaglavlja : K7DT8M18PLOM

Register

End point path\Putanja do krajnje točke: /register
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- username : String
- password : String
- email : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, message: Empty post form
- Code 4, message: Error with Form data
- Code 5, message: Error with Database
- Code 6, message: Username already exists
- Code 7, message: Internal error - parsing JSON

Data response\Struktura primljenih podataka:

- user_id : String
- username : String
- password : String
- email : String

Login

End point path\Putanja do krajnje točke: /login
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- username : String
- password : String

Possible responses\Moguči odgovori

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, message: Empty post form
- Code 4, message: Error with Form data
- Code 5, message: Error with Database
- Code 7, message: Internal error - parsing JSON
- Code 8. message: Wrong username/password

Data response\Struktura primljenih podataka:

- user_id : String
- username : String
- password : String
- email : String

Forgotten Password

End point path\Putanja do krajnje točke: /forgottenPassword
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- username : String
- email : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, message: Empty post form
- Code 4, message: Error with Form data
- Code 5, message: Error with Database
- Code 7, message: Internal error - parsing JSON
- Code 9. message: Username and Email doesn't match any user

Data response\Struktura primljenih podataka:

- user_id : String
- username : String
- password : String
- email : String

Reset password

End point path\Putanja do krajnje točke: /resetPassword
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- email : String
- password : String
- repeated_password : String
- new_password : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, message: Empty post form
- Code 4, message: Error with Form data
- Code 5, message: Error with Database
- Code 7, message: Internal error - parsing JSON
- Code 9. message: Username and Email doesn't match any user
- Code 10, message: Password updated successfully
- Code 11, message: Password are not equal

Data response\Struktura primljenih podataka:

- user_id : String
- username : String
- password : String
- email : String

Get Server Address

End point path\Putanja do krajnje točke: /getServerAddress
Method\Metoda: POST

Needed atributs\Potrebni atributi: null

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 7. message: Internal error – Parsing JSON

Data response\Struktura primljenih podataka:

- address : String

Get Friend list

End point path\Putanja do krajnje točke: /getFriendList
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- user_id : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 5, Error with Database
- Code 7. message: Internal error – Parsing JSON

Data response\Struktura primljenih podataka:

- Data : list<Friend>

Friend element data

- user_id : String
- username : String

Get Friend requests

End point path\Putanja do krajnje točke: /getFriendRequests
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- user_id : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 5, Error with Database
- Code 7. message: Internal error – Parsing JSON

Data response\Struktura primljenih podataka:

- Data : list<Friend>

Friend element data: Struktura objekta Friend

- user_id : String
- username : String

Get User by ID

End point path\Putanja do krajnje točke: /getUserById
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- user_id : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 7. message: Internal error – Parsing JSON
- Code 9. message: Username and Email doesn't match any user

Data response\Struktura primljenih podataka:

- user_id : String
- username : String
- password : String
- email : String

Accept friend request

End point path\Putanja do krajnje točke: /acceptFriendRequest
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- user_id : String
- friend_id : String

Possible responses\Moguči odgovori:

- Code 1, message: Success
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 7. message: Internal error – Parsing JSON
- Code 9. message: Username and Email doesn't match any user

Data response\Struktura primljenih podataka:

- null

Decline friend request

End point path\Putanja do krajnje točke: /declineFriendRequest
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- user_id : String
- friend_id : String

Possible responses\Moguči odgovori:

- Code 1, message: Success
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 7. message: Internal error – Parsing JSON
- Code 9. message: Username and Email doesn't match any user

Data response\Struktura primljenih podataka:

- null

Send friend request

End point path\Putanja do krajnje točke: /sendFriendRequest
Method: POST

Needed atributs\Potrebni atributi:

- user_id : String
- friend_username : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 5, message: Error with Database
- Code 7. message: Internal error – Parsing JSON
- Code 9. message: Username and Email doesn't match any user

Data response\Struktura primljenih podataka:

- user_id : String
- username : String
- password : String
- email : String

Get Statistics

End point path\Putanja do krajnje točke: /getStatistics
Method\Metoda: POST

Needed atributs\Potrebni atributi:

- user_id : String

Possible responses\Moguči odgovori:

- Code 1, message: null
- Code 2, message: Authentication needed!
- Code 3, Empty post form
- Code 4, message: Error with Form data
- Code 5, Error with Database
- Code 7. message: Internal error – Parsing JSON

Data response\Struktura primljenih podataka:

- Data : list<StatisticsResonse>

StatisticsResonse element data\Struktura objekta StatisticsResponse:

- wins : int
- draws : int
- loses : int
- games : list<Game>

Game element data\ Struktura objekta Game

- game_id : String
- player_one : String
- player_two : String
- time : Long
- winner : String
- draw : boolean
- turns : int

Prilog 2. Izvorni kodovi

Izvorni kodovi

- Poslužitelj: https://gitlab.com/Im_Pew/Zavrsni_Server
- Klijent: https://gitlab.com/Im_Pew/KrizicKruzic