**PROJEKTNA DOKUMENTACIJA**

Nacrtaj&Pogodi

**Predmet**:ElektroničkoPoslovanje

**Grupa 3**

Marko Rezić 590/RM

Hrvoje Zovko 571/RM

Toni Jelonjić 585/RM

Sadržaj

[Uvod 3](#_Toc144991064)

[1. Tehnologije 3](#_Toc144991065)

[1.1. React 3](#_Toc144991066)

[1.2. Node 4](#_Toc144991067)

[1.3. MySQL 4](#_Toc144991068)

[1.4. Figma 5](#_Toc144991069)

[2. Arhitektura Aplikacije 5](#_Toc144991070)

[2.1. Frontend 5](#_Toc144991071)

[2.2. Backend 5](#_Toc144991072)

[2.2.1. REST API 6](#_Toc144991073)

[2.2.2. WebSocket 6](#_Toc144991074)

[2.3. Baza 8](#_Toc144991075)

[3. Struktura Projekta 9](#_Toc144991076)

[3.1. Frontend 10](#_Toc144991077)

[3.2. Backend 10](#_Toc144991078)

[4. Aplikacija 10](#_Toc144991079)

[4.1. Prijava 11](#_Toc144991080)

[4.2. Lobby 11](#_Toc144991081)

[4.3. Game Setup 11](#_Toc144991082)

[4.4. Game 12](#_Toc144991083)

[4.5. Rezultati 13](#_Toc144991084)

[4.6. Postavke 13](#_Toc144991085)

[5. Development 14](#_Toc144991086)

[5.1. Github 14](#_Toc144991087)

[5.2. Lokalno okruženje 14](#_Toc144991088)

[5.3. Pokretanje projekta 15](#_Toc144991089)

[5.4. Postman 15](#_Toc144991090)

[5.5. Praćenje razvoja projekta 16](#_Toc144991091)

[5.6. Buduća proširenja 16](#_Toc144991092)

[6. Zaključak 17](#_Toc144991093)

# Uvod

Ovaj dokument služi kao tehnička dokumentacija i uputa za razumijevanje razvoja, funkcionalnosti i pokretanja projekta "Nacrtaj&Pogodi".

Ideja projekta je bila stvoriti platformu za igricu crtanja i pogađanja u stvarnom vremenu (real-time).

Kroz iduća poglavlja biti će detaljno opisani koraci kako je projekt nastao, te kako ga se može klonirati i pokrenuti lokalno za dalji razvoj.

Projekt je stvoren kao projektni zadatak za kolegij "Elektroničko poslovanje" u svrhu pokazivanja kako se može koristiti Websocket za komunikaciju višestrukih klijenata i servera u stvarnom vremenu.

Ovakav pristup omogućava funkcionalnosti kao što su slanje poruka, provjeravanje statusa, praćenje događaja i slično.

Razvoj projekta podijeljen je u sljedeće cjeline i osobe:

* Frontend: Toni Jelonjić, Marko Rezić, Hrvoje Zovko
* Backend: Marko Rezić
* Dizajn: Hrvoje Zovko
* Baza: Toni Jelonjić
* Code Review: Marko Rezić

# 1. Tehnologije

## 1.1. React

React je besplatna open-source biblioteka za izgradnju korisničkog sučelja od ponovno iskoristivih komponenti.

Razvio ga je Facebook i prvi put objavio 29.5.2013.

React spada među popularnije JavaScript frameworke, ima širok spektar funkcionalnosti te veliku bazu korisnika za referencu prilikom traženja nekog specifičnog koda.

React-ova dinamična JSX sintaksa omogućava lagano kombiniranje HTML elemenata sa JS logikom, što je vrlo korisno u razvoju nečega poput igrice.

Premda React nije nužno najbolja opcija kada je u pitanju game development, ovaj projekt se pokazao da nije previše zahtijevan sa strane performansi, te stoga je ova tehnologija odabrana za Frontend zbog svoje praktičnosti u brzom razvoju, te zbog većinske upoznatosti tima sa tom tehnologijom.

React također daje odličnu podršku za praćenje stanja aplikacije (state), što je neophodno za bilo kakav vid igre, s obzirom da se u svakom trenutku prelazi iz jednog stanja u iduće kako bi korisnicima neprestano tekao životni ciklus aplikacije.

U ovom projektu je pretežito korištena paradigma funkcionalnih komponenti (za razliku od klasnih), te tzv. React Hooks koji su vrlo korisni za brzu implementaciju raznih funkcionalnosti koje su pretežito imale klasne komponente.

Primjeri korištenih Hooks su:

* useState - za stanje,
* useEffect - za detektiranje promjena u stanju ili drugima varijablama,
* useLocation - za praćenje promjena lokacije preglednika,
* useNavigate - za kontroliranu navigaciju bez strogog učitavanja stranice nanovo,
* useContext - za praćenje globalnog stanja, koje je dostupno svakoj komponenti - ovaj Hook se ispostavio i najkorisniji za svrhu igre, budući da se sastoji od mnogih komponenti koje nisu uvijek nužno povezane, ali ovise o istom stanju,
* useRef - za zadržavanje referenci bitnih objekata u aplikaciji (poput ploče za crtanje - canvas), te
* useParams - za praćenje dinamičkih parametara u ruti preglednika.

Ručno pravljen Hook za ovaj projekt je useDebounce, koji za proslijeđenu funkciju dodijeli određeni delay, te ako se funkcija ponovo pozove dok taj delay nije istekao, prethodni poziv za funkciju se odbacuje.

Ovaj Hook se značajno koristio kod pravljenja zahtjeva na server jer omogućava da korisnik nesmetano pritišće gumbove, a da se pri tome samo konačna vrijednost šalje na server.

## 1.2. Node

*Node* je open-source JavaScript *runtime* okruženje za različite platforme (*cross-platform*).

Poznat je po svojoj brzini i asinkronosti, te lakoći pisanja. Kreiranje Node servera jedan je od najlakših i najbržih načina za uspostavljanje lokalnog HTTP servera, te uz naročito velik broj dostupnih *npm* paketa se pokazao jako širokih sposobnosti.

Asinkronost mu naročito poboljšava iskoristivost, jer jedan zahtjev na server ne mora nužno zablokirati ostatak rada dok se ne dovrši.

*Node* *backend* je čest izbor uz *React frontend*, a s obzirom da su izgrađeni na istim tehnologijama, također uklanja potrebu učenja drugog programskog jezika.

## 1.3. MySQL

*MySQL* je open-source sustav za upravljanje relacijskim bazama podatak, jedan od popularnijih izbora pored *PostgreSQL*, SQL Server i drugih.

Pruža sve potrebne funkcionalnosti za kreiranje koliko jednostavnih, toliko i kompleksnih baza podataka za bilo kakav regularan scenarij. Pod regularan se smatra neki set podataka koji ima jasne ili relativno jasne veze, s toga i relacije.

Za ovaj projekt nisu bili kompleksni zahtjevi za bazu, korištene su dvije tablice od kojih je jedna vezana stranim ključem za drugu.

Baza podataka je bila potrebna za čuvanje trajnih podataka o korisnicima.

## 1.4. Figma

Premda nije tehnologija za razvoj softvera, već alat za razvoj korisničkog sučelja i prototipa, *Figma* se ispostavila kao vrlo koristan alat za brzo slaganje dizajna aplikacije, te korisna zajednička referenca na određene dijelove aplikacije kako bi se što lakše uskladili razvojni programeri.

# 2. Arhitektura Aplikacije

## 2.1. Frontend

*Frontend* aplikacije je zamišljen kao polazna točka korisnicima za pristup sustavu; prijavom ili registracijom im je omogućen pristup ostalim dijelovima platforme.

Korisnik kroz određene dijelove *Frontend*-a aplikacije (Prijava, Lobby, Igra) komunicira sa *Backend* servisima kako bi postigao određenu radnju.

Prije svega za korisnika je nužno da je prijavljen odnosno da je lokalno spremio podatke koje je dobio od servera kako bi radio bilo kakvu drugu radnju.

Nakon autentifikacije korisniku se u *sessionStorage* (lokalnu sesiju) spremaju njegovi podaci, te to omogućava da za tu specifičnu karticu preglednika ostane prijavljen dok ju ne ugasi

Zbog niskih zahtjeva za sigurnost radi same prirode aplikacije donesena je odluka da sesije na serveru putem kolačića nisu nužne.

Samim time odjava korisnika ne zahtjeva slanje zahtjeva prema serveru, već je dovoljno pobrisati lokalnu pohranu.

## 2.2. Backend

*Backend* je podijeljen u dva dijela, *REST API* koji je zadužen za trajne korisničke podatke, te *WebSocket* koji upravlja procesima u stvarnom vremenu.

Zbog lakšeg razlikovanja ova dva dijela servera, dodijeljeni su zasebni portovi, 4000 se koristio za *REST*, a 5000 za *WebSocket*.

## 2.2.1. REST API

*REST API* upravlja korisničkim računima i rezultatima, te je neophodan ukoliko želimo da naši korisnici budu zapamćeni idući put kada se vrate na sustav

API ima dostupne sljedeće rute:

* POST /users/login - prijava, potrebni parametri u POST tijelu su { username: string, password: string },
* POST /users/register - registracija, potrebni parametri u POST tijelu su { username: string, password: string },
* PUT /users/username/:user\_id - promjena korisničkog imena korisnika sa ID brojem user\_id, potrebni parametri u PUT tijelu su { username: string },
* PUT /users/avatar/:user\_id - promjena slike profila korisnika sa ID brojem user\_id, potrebni parametri u PUT tijelu su { avatar: string },
* POST /scores - spremanje rezultata nakon igre, potrebni parametri su { total\_points\_user\_id\_map: { user\_id: string: points: number } }, odnosno mapiranje korisničkih ID brojeva i rezultata, ova ruta se isključivo poziva kod websocketa nakon igre, client ne smije pozivati ovu rutu, te
* GET /scores?user\_count=param1&round\_count=param2 - dohvaćanje rezultata svih igara sa param1 brojem igrača i param2 brojem rundi.

Za *REST* je korišten *Express* kao *framework*, sa *json middleware parser*-om, koji omogućava čitanje *body* objekta za svaki *request*.

## 2.2.2. WebSocket

*WebSocket* obrađuje sve zahtjeve koji trebaju biti praćeni u stvarnom vremenu.

Korištena je biblioteka *SocketIO*, koja omogućava kreiranje višestrukih soba uz razne opcije emitiranja u sobe ili specifične *socket-*e.

Nakon spajanja, odnosno uspostave "*connection*" eventa sa serverom, *client* sada ima dvosmjernu real-time komunikaciju na tom *socket*-u sa serverom.

Dalje se slanjem poruka specifičnog imena server i klijent su postavljeni da odgovaraju na međusobne promjene stanje, te da te promjene odašilju ostalim *socket*-ima u istoj sobi.

Postoje 2 tipa odašiljanja poruka:

* *emit*: *socket* šalje poruku svima u sobi,
* *broadcast*: *socket* šalje poruku svima u sobi osim sebi.

Specifične poruke koje klijent šalje (a server sluša sa *socket.on*(...)) su:

* "*connection*": primarno uspostavljanje veze između klijenta i servera
* "*disconnect*": ne šalje se ručno već je automatski na serveru detektirano ukoliko *socket* izgubi postojeću vezu, emitira obavijest "*user-leave*" ostalim korisnicima u sobi
* "*join-room*": slanje zahtjeva za pridruživanje sobi, potrebno je proslijediti *user\_id* i *room\_id*.

Kako bi se korisnik mogao pridružiti sobi potrebno je ispuniti sljedeće zahtjeve:

* ne smije već biti u sobi
* soba ne smije biti puna (maksimalan broj igrača dostignut)
* ukoliko je korisnik u nekoj drugoj sobi, sve se sobe osim zatražene napuštaju za taj *socket*

Nakon uspješnog pridruživanja šalje se obavijest "*user-joined*" svima u sobi o tome tko se pridružio.

Ukoliko je korisnik jedini u toj sobi, smatra se da ju je on stvorio.

Nakon neuspješnog pridruživanja tom *socket*-u se šalje obavijest "*join-failed*" uz odgovarajući razlog:

* "*leave-room*": napuštanje sobe, šalje se u svrhu obavještavanja drugih *socket*-a porukom "*user-leave*" u toj sobi o njezinom napuštanju, te da bi se znalo koji korisnik je u pitanju,
* "*change-settings*": slanje obavijesti "*update-game-settings*" o promjenama za postavke igre koje je korisnik uveo u toj sobi ,
* "*start-game*": slanje zahtjeva za započinjanje igre, ne šalje se nikakva obavijest kao odgovor već se promijeni stanje igre te backend automatski obavještava sa "*game-chat-message*" sve korisnike u sobi o njenom početku,
* "*send-message*": slanje poruke u prostor za čavrljanje, obavještava sa "*game-chat-message*" tekst poruke, te
* "*canvas-data*": slanje podataka o trenutnom stanju ploče za crtanje, slika se šalje u obliku *base64* enkodiranog *string*-a, te se ponovno emitira ostalim *socket*-ima u toj sobi sa istom porukom "*canvas-data*".

Specifične poruke koje server šalje klijentima (a klijent sluša sa *socket.on*(...)) su:

* "*user-join*": obavijest o novom korisniku koji se pridružio sobi,
* "*user-leave*": obavijest o korisniku koji je napustio sobu,
* "*update-game-state*": glavna poruka za promjenu i sinkroniziranje stanja igre na server i u klijentima, šalje *server*-ovo vjerodostojno stanje igre (*gameState*) te u određenim slučajevima *gameSettings* kao postavke igre.

Svaki ovaj zahtjev ima i informaciju o tome kakav točno event je to u kontekstu igre odnosno *event\_name*:

* + next-tick: označava promjenu stanja u idućoj sekundi igre, šalje se svaku sekundu
  + next-user-turn: označava početak crtanja idućeg korisnika
  + next-round: označava početak iduće runde
  + correct-guess: označava pogodak tražene riječi
  + game-over: označava kraj igre
  + admin-change: označava promjenu admina sobe
  + user-list-change: označava promjenu broja korisnika, korisno za detektiranje novih ili odlazećih korisnika u slučaju da poruke "user-leave" ili "disconnect" nisu detektirane
* "update-game-settings": obavijest o specifičnim (korisnički uzrokovanim) promjenama postavki sobe
* "game-chat-message": slanje poruke u chat neke sobe, poruke imaju specifično formatiranje i pristup na temelju tipa poruke specificirano u MessageType i pristupu specificiranom u MessageAccess:
  + public: javni pristup, svima vidljivo
  + private: privatni pristup, vidljivo specificiranom socketu
  + info: informacije o igri
  + success: uspješna radnja, poput točnog pogotka
  + winner: naglašavanje pobjednika igre
  + generic: obične poruke

Server je jedini svjestan popisa soba, korisnici nemaju pristup tome, već samo znaju koji je *ID* broj njihove sobe pomoću rute na kojoj se nalaze.

Samim time, za sva stanja soba i njihovo periodično ažuriranje zaslužan je server.

Za ovaj projekt postavljena je asinkrona funkcija koja poziva samu sebe da ažurira sve sobe, radnje koje može poduzet su sljedeće:

* + brisanje sobe: ovo se može dogoditi u slučaju kada je soba prazna(osim za *public* sobu),
  + uklanjanje identičnih korisnika u sobi (indicira na grešku prilikom pristupanja),
  + mijenjanje *admin*-a sobe, ovo se može dogoditi ukoliko *admin* sobe napusti sobu. Po pravilu svaka soba osim *public* sobe mora imati *admin*-a, jer jedini on može pokrenuti igru,
  + ažuriranje stanja sobe na temelju iznad navedenih "*game-state-update*" poruka, te
  + ažuriranje postavki sobe na temelju iznad navedenih "*game-settings-updated*" poruka.

## 2.3. Baza

Baza i *Backend* međusobno komuniciraju korištenjem *mysql2 npm* paketa, koji omogućava sve funkcionalnosti za slanje upita na bazu.

Radi preglednosti i lakšeg korištenja u drugim dijelovima projekta, konfiguracija i inicijalizacija baze je premještena u zaseban file "*database.js*".

Za bazu su kreirane dvije pristupne točke "*db*" i "*db\_parallel*". Obe su u suštini isti objekt, *Pool*, odnosno skup dostupnih konekcija, ali je *db\_parallel* konfiguriran da dopušta izvršavanje višestrukih upita u jednom zahtjevu.

Premda bi se to moglo implicirati iz naziva, *db\_parallel* zapravo ne može odraditi više upita "paralelno", već može slati upit koji sadrži više naredbi u sebi, te ih obrađuje sekvencijalno.

Zbog same prirode unosa podataka u bazu nije ni moguće ostvariti paralelni unos.

Dvije glavne tablice su:

* + *users*: tablica svih korisnika, sa sljedećim stupcima:
    - *user\_id*: ID broj, primarni ključ,
    - *username*: Korisničko ime, mora biti unikatno (UNIQUE KEY),
    - *password*: Šifra korisnika, nije implementirana nikakva posebna enkripcija zbog niskih sigurnosnih zahtjeva projekta,
    - avatar: Profilna slika, zapisano kao cijeli broj koji označava indeks dostupnih statičkih slika avatara na *Frontend*-u,
    - *created\_at*: Generirani stupac koji označava vrijeme trenutka stvaranja korisnika, te
    - *updated\_at*: Generirani stupac koji označava vrijeme zadnjeg ažuriranja korisnika.
  + *scores*: tablica svih rezultata, sa stranim ključem koji se veže na tablicu *users*. Ima sljedećim stupcima:
    - *score\_id*: ID broj,
    - *user\_id*: Korisnički ID broj, strani ključ (FOREIGN KEY),
    - *points*: broj bodova, cijeli broj,
    - *round\_count*: broj rundi u igri na koju se rezultat odnosi (služi za kategorizaciju),
    - *user\_count*: broj korisnika u igri na koju se rezultat odnosi (služi za kategorizaciju),
    - *created\_at*: Generirani stupac koji označava vrijeme trenutka stvaranja rezultata, te
    - *updated\_at:* Generirani stupac koji označava vrijeme zadnjeg ažuriranja rezultata.

*REST API* jedini komunicira sa bazom direktno, indirektno *Frontend* i *Websocket* prave zahtjeve na *API* → Baza.

# 3. Struktura Projekta

U ovom poglavlju ćemo dati generalni pregled struktura datoteka, foldera, modula i njihovog međusobnog povezivanja.

Struktura projekta sama po sebi se možda čini kao trivijalan izbor, no postaje eksponencijalno bitnije imati čistu i logičnu strukturu s porastom broja članova na projektu.

## 3.1. Frontend

*Frontend* projektna struktura najprije započinje kao standardni nanovo-stvoreni React projekt, sa svim globalnim komponentama u glavnom (*root*) folderu projekta, a komponentama smještenim u *src* folderu.

U glavnom folderu također imamo:

* foldere za statičke datoteke medija "/*img*", "/*music*", "*sounds*",
* globalno korištene elemente poput kontekst *API*-a (*Context.js*),
* *axios* za pravljenje HTTP *request*-ova na *REST API* sa postavljenom konfiguracijom i portovima za zajedničku instancu,
* generalno korisne funkcije "/*utils*".

Dalje u "/*src*" folderu se nalaze iskoristive *React* komponente, te je svaka veća cjelina odvojena u zaseban folder zajedno sa svojim "*style.css*" *file*-om.

Folderi podrazumijevaju sljedeće cjeline (ekrane):

* + "/*common*": zajedničke komponente koje se mogu pojaviti na svakom ekranu, poput modala i *loader*-a,
  + "/*auth*": prijava i registracija,
  + "/*lobby*": glavni izbornik nakon autentifikacije,
  + "/*game-setup*": postavke igre prije početka, te
  + "/*game*": *real-time* igrica (kad je već pokrenuta).

## 3.2. Backend

*Backend* struktura je značajno jednostavnija, ovo je rađeno zbog malih zahtjeva što se tiče rutera za http zahtjeve.

*REST API* i *WebSocket* su smješteni u isti "*index.js*" file, te pokrenuti jedno za drugim.

Konfiguracija baze se dobiva iz "*database.js*" datoteke

Podaci za spajanje na bazu se nalaze u ".*env*" datoteci

Datoteka za varijable okruženja ".*env*" se ne prati u *git* repozitoriju iz sigurnosnih razloga, te je potrebno tu datoteku kreirati lokalno i ispuniti odgovarajuće podatke za spajanje.

Primjer datoteke sa nazivima varijabli okruženja je u .*env.example.*

U glavnom folderu se nalaze i dvije tekstualne datoteke "*words\_hr.txt*" i "*words\_en.txt*", koje sadrže popis riječi za pogađanje na hrvatskom i engleskom jeziku, moguće je lokalni backend onda podesiti da koristi prilagođeni set riječi.

# 4. Aplikacija

Ovo poglavlje će proći kroz korištenje i upute za sve ekrane aplikacije, te dati primjer ispravnog rada aplikacije za lokalnu referencu.

## 4.1. Prijava

Ruta: "/*login*".

Prijava je prvi ekran na koji korisnik ima pristup, potrebno je unijeti korisničko ime (nadimak, ili *username*) i šifru.

Korisnik ima opciju prijave ili registracije, ukoliko do sada nije imao račun mora se registrirati, te mora se pobrinuti da nema korisničko ime koje se već koristi.

Uspješnom autentifikacijom korisnik je spremljen u sesiju od kartice preglednika (*sessionStorage*) i neće se morati ponovno prijaviti dok ne ugasi karticu.

Ruta za prijavu je blokirana i preusmjeri korisnika na rutu "/" odnosno *Lobby* dok god je korisnik prijavljen.

## 4.2. Lobby

Ruta: "/"

Ovaj ekran služi kao središnja točka iz koje korisnik može pristupiti svim ostalim autentificiranim ekranima.

Korisniku je omogućen izbor raznih "*avatara*" odnosno statičkih slika od kojih jednu može izabrati da se prikazuje tijekom igre. Svakim odabirom avatara se odradi *useDebounce* zahtjev za promjenu.

Unos za *username* je tu prisutan kako bi korisnik mogao promijeniti korisničko ime. Svakom promjenom polja se odradi *useDebounce* zahtjev za promjenu.

Opcija "Igraj" korisnika smjesti u javnu sobu, odnosno sobu sa ID brojem *room\_id = "public".*

Opcija "Kreiraj privatnu sobu" smjesti korisnika u novu kreiranu sobu, generira se unikatni *UUID* broja verzije 4 (*UUIDv4*) kao *room\_id*, što statistički osigurava da tu sobu nitko neće moći nasumično pogoditi, te da do sada nije stvorena soba sa tim *ID*-om.

Obe prethodne opcije prebacuju korisnika u "*GameSetup*" ekran.

Opcija "Odjavi se" briše podatke iz sesije te vraća korisnika na ekran "Prijava".

## 4.3. Game Setup

Ruta: *"/room/:room\_id".*

Ovaj ekran je jedino dostupan ukoliko soba sa *ID* brojem *room\_id* NIJE ZAPOČELA igru (stanje igre "*started*" je *false*).

Ovaj ekran se drugačije ponaša ovisno o vrsti sobe:

* + *public*: Ovo je jedina soba koja je vječno aktivna i njome samo server upravlja, nema *admin*-a, ima beskonačno rundi, neograničen broj korisnika, runde traju 60 sekundi i ima 2 pomoćna slova po rundi.
  + *private*: Soba kreirana od strane korisnika, sa svojim unikatnim *UUIDv4* kao *room\_id*.

Svojstva *UUIDv4* omogućavaju da se čak i sobe od više spojenih servera međusobno nikad ne preklapaju.

Korisnik u privatnoj sobi koju kreira postaje automatski *admin*, te može upravljati svim postavkama sobe:

* + Maksimalan broj igrača (2-8),
  + Vrijeme po rundi (20-120),
  + Broj rundi (2-10),
  + Broj pomoćnih slova (0-10),
  + Nasumične riječi (dodatne riječi razdvojene zarezom koje će se najprije koristiti za pogađanje umjesto nasumičnih riječi sa servera), te
  + Početak igre.

Chat je ovdje dostupan kako bi korisnici mogli komunicirati i prije početka igre.

## 4.4. Game

Ruta: *"/room/:room\_id".*

Ovaj ekran je jedino dostupan ukoliko soba sa *ID* brojem *room\_id* je ZAPOČELA igru (stanje igre "*started*" je *true*).

Na ovom ekranu igrači u svakoj rundi crtaju jednom, dok ostali pogađaju.

Iduća runda počinje kada istekne vrijeme crtanja, ili kada svi igrači (osim crtača) pogode riječ.

Igrač koji crta ima alate na donjem dijelu ekrana da podesi boju i veličinu olovke, također ima ispisanu riječ koju treba nacrtati, te ne može pisati u chat dok crta.

Igrači pogađaju riječ slanjem poruke u chat, ukoliko je poruka točan pogodak neće se prikazati riječ u chatu, i korisnik neće moći pisati poruke do kraja runde kako ostali igrači nebi saznali riječ od tog korisnika.

Ukoliko su postavljena pomoćna slova u postavkama, postepeno će se otkrivati slova tražene riječi kako se vrijeme runde smanjuje.

Svaki točan odgovor pridodaje bodove korisniku koji pogodi.

Igrači dobivaju bodove na temelju sljedećih faktora:

* + Ako je pogodio prvi: +50,
  + Ako je pogodio unutar 10 sekundi od početka runde: +100 bodova,
  + Ako je pogodio riječ dužu od 6 slova: +30 bodova za svako slovo preko šestog,
  + Ako je pogodio riječ dok su prikazana pomoćna slova: -10 bodova za svako otkriveno slovo, te
  + Bodovi na temelju preostalog vremena runde: 30 + 200 \* (1 - (ukupno\_vrijeme - preostalo\_vrijeme) / ukupno\_vrijeme) (maksimalno 230, minimalno 30).

Ukupni bodovi za tu rundu su zbroj individualnih.

Igrači se rangiraju na temelju bodova, te se za prva tri mjesta prikazuju "medalje".

Igra završava kada je maksimalan broj rundi dostignut, ili u sobi ostane manje od 2 igrača.

Nakon završetka igre ekran se vraća na "*GameSetup*", rezultati se spremaju u bazu, te se u chat proglašava pobjednik.

Soba se briše kada nema više igrača u njoj (osim *public* sobe).

## 4.5. Rezultati

Ovaj element služi kao modal za prikazivanje svih rezultata po kategorijama na temelju broja igrača i broja rundi u igri.

U modalu imaju opcije za mijenjanje kategorija, te se prikazuje popis korisnika i njihovih rezultata, rangiranih od najvećeg broja bodova do najnižeg.

## 4.6. Postavke

Ovaj element služi kao modal za prikazivanje i podešavanje generalnih postavki aplikacije.

Trenutno podržava samo dvije postavke:

* + Glasnoća glazbe
  + Glasnoća zvukova

Obe postavke imaju input tipa *range* (*slider*) čijim mijenjanjem se podešava glasnoća u rasponu od 0 do 100%.

Pozadinskom glazbom i zvukovima isključivo upravljaju *MusicPlayer* i *AudioManager* instance.

Ove klase sadrže predefiniran set zvukova i glazbe koji su popisani u objektima *MusicType* i *SoundType*.

Zvučni efekti podrazumijevaju kratke zvukove koji se obično čuju prilikom nekih kratkih interaktivnih događaja poput klika na tipku, slanje poruke, naglašavanje runde i slično.

Pozadinska glazba se automatski pokreće i bira na temelju trenutnog ekrana.

Zbog sigurnosnih pravila u preglednicima nužno je ostvariti bar jednu korisničku interakciju sa stranicom kako bi HTML Audio mogao reproducirati zvukove.

# 5. Development

Premda je cilj ovog projekta postignut, postoji još raznih mogućnosti za razvoj ovakvih sustava. "Nacrtaj&Pogodi" predstavlja primjer osnovne primjene *WebSocketa*, te je kod javno dostupan za preuzimanje.

Sljedeća poglavlja će preći postavljanje projekta na lokalnom računalu.

## 5.1. Github

Projektni repozitorij je dostupan i može se klonirati sa *github* linka:

*https://github.com/MarkoRezic/React-DrawIO-with-Websocket*

Potrebno je imati *git* instaliran i prepoznat kao komandu da biste klonirali repozitorij putem konzole.

Da biste ga klonirali lokalno možete otvoriti bilo koji terminal i upisati sljedeću naredbu:

*git clone https://github.com/MarkoRezic/React-DrawIO-with-Websocket*

Nakon toga biste trebali imati folder "*React-DrawIO-with-Websocket*" u kojem su 2 foldera, jedan za *frontend* i jedan za *backend*.

## 5.2. Lokalno okruženje

Da bi backend ispravno radio potrebno je postaviti varijable okruženja u ".*env*" datoteku.

Ova datoteka se ne prati u promjenama repozitorija, te ju je potrebno kreirati ukoliko ne postoji.

Primjer dostupnih varijabli je u ".*env.example*" datoteci, te treba popuniti sa odgovarajućim vrijednostima za spajanje na bazu

Ukoliko želite koristiti *defaultne* postavke *MySQL*-a potrebno ga je imati pokrenutog i dostupnog na portu 3306.

Također je potrebno kreirati shemu, koju možete nazvati po želji, ali mora imati strukturu kao "*drawio\_dump.sql*" datoteka.

Ako imate dostupan neki od alata za upravljanje *MySQL* bazom poput *MySQL Workbench,* možete preko njih importati file direktno u program na *Server > Data Import*.

Ukoliko koristite *MySQL* preko terminala, potrebno je se logirati u *MySQL* terminal i odraditi ručno import sa specificiranom putanjom do *dump* datoteke.

Dalje je bitno se smjestiti u foldere "/*frontend*" i "/*backend*" te zasebno u oba terminala pokrenuti *npm install* kako bi se dobavili *node\_modules* specificirani u *package.json*.

## 5.3. Pokretanje projekta

Nakon što je sve postavljeno najprije pokrenite *backend* naredbom (provjerite da se nalazite u *backend* folderu u terminalu):

* + - * *node ./index.js*

Frontend pokrenite sa:

* + - * *npm run start*

Stranica bi po defaultu trebala biti dostupna na *http://localhost:3000*

*REST API* bi trebao biti dostupan na *http://localhost:4000*

*WebSocket* bi trebao biti dostupan na *http://localhost:5000*

Ukoliko želite promijeniti portove, potrebno je u oba projekta podesiti *rest\_api\_port*, *websocket\_port, api\_origin, websocket\_origin, frontend\_origin, apiBaseUrl, websocketBaseUrl.*

Ukoliko želite da ima dodatno *origin* dopuštenje, možete promijeniti *alt\_origin*.

Za testiranje na više računala preporučeno je na privatnoj lokalnoj mreži dopustiti da se vaša računala međusobno mogu otkriti, te umjesto *localhost* koristiti IP adresu od računala na kojem je pokrenut dev server.

## 5.4. Postman

Za razvoj i testiranje *backend*-a preporučujemo *Postman*, desktop verziju.

Ovaj besplatni alat će vam omogućiti da neovisno o drugim dijelovima aplikacije možete kontrolirano testirati backend.

Štoviše, *Postman* je nedavno omogućio korištenje *WebSocket* tehnologije sa njihovim *API*-em, te ima vrlo intuitivno sučelje za testiranje slanja poruka, čitanje odgovora i slično.

Napomena: desktop verzija je potrebna da biste mogli se spojiti preko *WebSocket*-a.

## 5.5. Praćenje razvoja projekta

Za praćenje razvoja ovog projekta znatno je koristan neki alat za kreiranje i upravljanje zadacima, za ovaj projekt smo koristili *Trello*, međutim dostupne su i druge opcije kao *Azure DevOps* ili *Jira*.

Opseg ovog projekta nije nužno zahtijevao potpuno pridržavanje zadacima, ali je dobra praksa u općem slučaju, naročito ako se planira projekt kasnije proširiti.

*Trello* link od projekta dostupan je na: [*https://trello.com/b/4GuTITVH/grupa-3*](https://trello.com/b/4GuTITVH/grupa-3)

## 5.6. Buduća proširenja

Ovaj projekt sam po sebi jedva je iskoristio potencijal ovakvih tehnologija, te su u budućnosti planirane mnoge druge ideje poput:

* + Dodavanje i dopisivanje sa prijateljima,
  + Preuzimanje i spremanje crteža na profil,
  + Dodavanje raznih drugih alata za crtanje poput oblika, linija,
  + Dodavanje drugih funkcionalnosti za crtanje poput vraćanje na zadnji korak i slično,
  + Dodavanje boljeg prepoznavanja ako je korisnik blizu pogođene riječi,
  + Drugačiji *mode*-ovi igranja, te
  + Druge igre na istoj platformi.

*WebSocket* tehnologija kako bi što bolje radila treba se optimizirati backend logika.

To podrazumijeva smanjenje resursa potrebnih za obavljanje razmjene podataka kako bi se omogućio što veći broj istovremenih konekcija sa što boljim prijenosom.

Ovaj projekt se nije previše bazirao na optimizaciju samih metoda ažuriranja stanja igre, s obzirom da je lokalan i nema puno konekcija u bilo kojem trenutku, ali u slučaju deploymenta bi se svakako trebalo dovesti u razmatranje.

Glavno unaprjeđenje aplikacije bi se moglo smatrati da se promjene emitiraju sa servera samo kad su apsolutno nužne (a ne za svaku sekundu igre).

Također bi se trebalo samo slati promjena stanja, a ne čitavo stanje igre na svaki zahtjev kako bi se uštedilo na podatkovnom prijenosu.

# 6. Zaključak

Kroz ovaj projekt vidjeli smo i naučili kako se može koristiti *WebSocket* za prikazivanje i mijenjanje stanja višestruko povezanih klijenata u stvarnom vremenu.

Jedna implementacija tog principa je upravo igra crtanja i pogađanja ali ova tehnologija je daleko od toga da bude ograničena samo na taj slučaj.

Ovakav oblik razmjene podataka omogućava puno bolju interakciju korisnika sa serverom i sa drugim korisnicima, jer bolje emulira stvarni svijet te čini iskustvo uzbudljivijim.

U konačnici imamo platformu koja se dalje može širiti po potrebi te omogućiti interesantno iskustvo svojim korisnicima.