

*Prolećni semestar, 2024/25*

*SE201 Uvod u softversko inžinjerstvo*

“Ludo Game”

Asistent:Bogdan Janković Student: Anđela Nikodijević 5529

Profesor: Jovana Jović

Datum 09.10.2025.

Sadržaj

[Agilna metodologija 3](#_Toc208877679)

[Murov obrazac 5](#_Toc208877680)

[Funkcionalnih zahteva 6](#_Toc208877681)

[Nefunkcionalni zahtevi 8](#_Toc208877682)

[Korisničke priče 9](#_Toc208877683)

[Persona 10](#_Toc208877684)

[Scenariji za Persone 12](#_Toc208877685)

[Stilovi softverske arhitekture 15](#_Toc208877686)

[Klijent server 15](#_Toc208877687)

[Mikroservisi 16](#_Toc208877688)

[Funkcionalnosti aplikacije 17](#_Toc208877689)

[Tabla i pozicije piuna 17](#_Toc208877690)

[Klasa BoardPosition 17](#_Toc208877691)

[Klasa BoardDrawer 17](#_Toc208877692)

[Bacanje Kocke 18](#_Toc208877693)

[Klasa Dice 18](#_Toc208877694)

[Pomeranje piuna 19](#_Toc208877695)

[Klasa Pawn 19](#_Toc208877696)

[Upravljanje igrom 19](#_Toc208877697)

[Klasa PawnMenager 19](#_Toc208877698)

[Mrežna komunikacija 20](#_Toc208877699)

[Klasa Server 20](#_Toc208877700)

[Klasa Client 20](#_Toc208877701)

[Baza podataka 20](#_Toc208877702)

[Klasa Database 20](#_Toc208877703)

[Grafički interfejs 20](#_Toc208877704)

[Klasa Menu 20](#_Toc208877705)

[Klasa Chart 21](#_Toc208877706)

[Klasa Game 21](#_Toc208877707)

[Prikaz poruka i obaveštenja 21](#_Toc208877708)

[Klasa AlertMessage 21](#_Toc208877709)

[Tehnički detalji 22](#_Toc208877710)

[Arhitektura Aplikacije 24](#_Toc208877711)

[Ključne klase I njihove odgovornosti 26](#_Toc208877712)

[Database 26](#_Toc208877713)

[Server 28](#_Toc208877714)

[Client klasa 29](#_Toc208877715)

[PawnManager 30](#_Toc208877716)

[Pawn 32](#_Toc208877717)

[Ponovno upotrebljive komponente 35](#_Toc208877718)

[Računarstvo u oblaku 37](#_Toc208877719)

[Multiplayer server logika 37](#_Toc208877720)

[Čuvanje rezultata i statistike (Highscore, statistika partija) 37](#_Toc208877721)

[Hosting klijentske aplikacije 37](#_Toc208877722)

[Mikroservisi 38](#_Toc208877723)

[Servis za korisnike 38](#_Toc208877724)

[Servis za igre 38](#_Toc208877725)

[Servis za rezultate i statistiku 38](#_Toc208877726)

[Atributi kvaliteta sistema 39](#_Toc208877727)

[Scenariji testiranja 40](#_Toc208877728)

[Testovi 42](#_Toc208877729)

[Trello alat 43](#_Toc208877730)

[Git verzionisanje 44](#_Toc208877731)

[Zaključak 45](#_Toc208877732)

[Reference 46](#_Toc208877733)

# **Agilna metodologija**

Scrum je jedan od najčešće korišćenih **agilnih metodologija** i zasniva se na iterativnom i inkrementalnom razvoju. Projekat se deli na **sprintove** (najčešće 2–4 nedelje), a tim redovno pregleda napredak i prilagođava se.

Product Backlog (zaostatak proizvoda) je lista zadataka koji treba da se urade da bi se završio razvoj nekog proizvoda.

Lista funkcionalnosti koje treba razviti:

* P1: Osnovnu tabla sa poljima
* P2: Kretanje figure po tabli
* P3: Pravila bacanja kocke
* P4: Sistem za više igrača (multiplayer)
* P5: Čuvanje rezultata u bazi

Sprint Backlog

Prvi sprint, 2 nedelja:

* Osnovna tabla
* Kretanje figure
* Pravila bacanja kocke

Cilj sprinta: „Funkcionalna offline verziju igre za jednog igrača“

Sprint ciklus

1. **Planiranje sprinta** – tim dogovara koje stavke backlog-a ulaze u sprint
2. **Dnevni Scrum(tismki sastanak koji se održava svakog dana)** – ko je šta juče radio , ko je šta radio danas i koje su prepreke imali
3. **Izvođenje sprinta** – svako izvršava svoj zadatak
4. **Sprint Review** – na kraju sprinta demonstrira se da igra funkcioniše za jednog igrača, što je bio cilj

Drugi sprint, 4 nedelja:

* Sistem za više igrača
* Čuvanje rezultata u bazi

Cilj: „Dodati multiplayer podršku“

Prednosti Scrum-a:

* **Demo verzija posle prvog sprinta(offline)**
* **Fokus na prioritete**, prvo kretanje i pravila, pa onda multiplaywe
* **Lakše ispravljanje grešaka** jer se problemi otkrivaju na kraju svakog sprinta, a ne na kraju projekta

# ****Murov** **obrazac****

Murov obrazac je vizija i ukratko opisuje šta treba da se razvije, ko su kupci proizvoda i zašto bi trebalo da kupe proizvod koji razvijate.

Murov obrazac:

**ZA** ljubitelje društvenih igara i ljude koji žele da igraju zabavnu, poznatu igru online sa prijateljima  
**KOJI** žele da se povežu i zabave kroz digitalnu verziju tradicionalne igre  
**„Ne ljuti se čoveče“ Multiplayer** je **igrica**  
**KOJA** omogućava igranje klasične igre u modernom, digitalnom obliku sa više igrača u realnom vremenu  
**ZA RAZLIKU OD** sličnih aplikacija koje nude samo igranje protiv robota ili pojednostavljene verzije igre  
**NAŠ PROIZVOD** pruža autentično iskustvo društvene igre sa funkcionalnostima za online multiplayer, čuvanje rezultata u bazi i interaktivno korisničko okruženje

**„Ne ljuti se čoveče”** definiše svrhu, vrednost i pravac razvoja. Cilj proizvoda je da prenese iskustvo tradicionalne društvene igre u digitalni svet i omogući igračima da se povežu i igraju zajedno online, čak i kada nisu fizički blizu. Igra je zasnovana na jednostavnosti korišćenja, originalnih pravila i mehanike, kao i na društvenog aspekta kroz multiplayer opciju. Prvo izrada osnovne offline verzije, zatim dodavanje multiplayer sistema i čuvanja rezultata u bazi, a dugoročno obogaćivanje igre dodatnim funkcionalnostima poput statistike, koja boja najviše pobeđuje. Vizija određuje šta i zašto pravimo, a Scrum definiše kako i kada ćemo to isporučiti kroz sprintove.

# ****Funkcionalnih zahteva****

Funkcionalni zahtevi su **konkretne funkcionalnosti i ponašanja koje softver ili sistem mora da izvršava da bi zadovoljio potrebe korisnika**. Definišu **šta sistem treba da radi**, a ne kako će to tehnički biti implementirano.

1. Prikaz tebele – Tabela sa poljima na ekranu po kojima se kreću piuni, gde su kuće svkog piuna označene njihovim bojama kao I početne pozicije, prioritet je visok jer na osnovu tabele treba kasnije da se napravi za koliko će da se pomeraju piuni.
2. Kretanje piuna – piuni se kreću po tabeli, ali samo po određenim poljima I za onoliko polja Koliko su dobili broj na kockici, visok je prioritet zbog kasnijeg dodavanja pravila poput izbacivanje drugih piuna
3. Bacanje kocke – igrač može da baci kocku I dobiće broj od 1 do 6, ako dobije 6 imaće izbor da izbaci piuna na table ili da se kreće sa drugim piunom koji se već nalazi na table, prioritet je visok zbog kretanje piuna
4. Izlaženje iz baze – igrač mora dobiti 6 na kocki da bi mogao izaći iz baze, visok prioritet, jer bez toga igrač ne bi mogao da igra igru
5. Izbacivanje protivnika – igrač koji stane na već zauzeto polje vraća tog igrača u njegovu bazu, taj igrač mora opet dobiti 6 na kocki da bi mogao izaći iz baze, visok prioritet
6. Multiplayer – mogućnost da igraju više igrača zajedno, visok prioritet
7. Sinhronizacija poteza – obezbeđivanje da će svaki igrač doći na svoj red I moći da odradi svoj potez, visok prioritet
8. GUI za igrača – prikazivanje tabele sa poljima, piuna, kockice, …, visok prioritet bez toga igrač ne bi mogao da vidi šta se dešava u igri
9. Testiranje – testiranje je bitno da vidimo da li naša igra funkcioniše onako kako je naš poslodovac očekivao I na željeni način, visok prioritet
10. Čuvanje rezultata – na kraju svake partije čuva se ko je pobednik igre, prioritet je srednjeg nivoa
11. Prikaz pobednika – na kraju partije prikazuje se ko je pobedio, nizak nivo prioriteta
12. Animacija kockice – kockica koja se vrti kada se baci I pada na neki broj od 1 do 6, nizak nivo prioriteta

Kako su svi zahtevi povezani:

Zahtev 1. Je povezan sa 2, 3 I 4 zahtevom

Zahtev 2. Je povezan sa 1 I 3 zahtevom

Zahtev 3. Sa drugim zahtevom

Zahtev 4. Sa 2 I 3 zahtevom

Zahtev 5. Sa drugim zahtevom

Zahtev 6. Sa 7,8 I 10 zahtevom

Zahtev 7. Sa šestim zahtevom

Zahtev 8. Sa 1, 2, 3 I 6 zahtevima

Zahtev 9. Sa svim zahtevima, sve je potrebno testirati da bi smo znali da funkcioniše na željeni način

Zahtev 10. Sa 11-tim zahtevom

Zahtev 11. Sa 10 zahtevom

Zahtev 12. Sa 3 I 8 zahtevom

# ****Nefunkcionalni zahtevi****

Nefunkcionalni zahtevi su **zahtevi koji opisuju kako sistem treba da funkcioniše**, a ne šta konkretno radi. Definišu **kvalitet, performanse, pouzdanost, bezbednost i ograničenja sistema**, umesto konkretnih funkcionalnosti.

1. **Vreme odziva -** Sistem mora reagovati na potez igrača i bacanje kocke u **maksimalno 1 sekundu**, kako bi korisničko iskustvo bilo fluidno i bez zastoja
2. **Dostupnost -** Multiplayer komponenta igre mora biti dostupna najmanje **99% vremena** tokom dana, kako bi igrači mogli nesmetano da se povežu i igraju
3. **Skalabilnost -** Sistem mora da podrži istovremeno **do 4 igrača** bez degradacije performansi, sa mogućnošću proširenja servera za veći broj igrača u budućnosti
4. **Bezbednost -** Svi podaci igrača kao rezultat igre, moraju bti **šifrovani pri čuvanju u bazi i prenosu preko mreže**
5. **Oporavak od grešaka -** Ako igrač izgubi vezu sa serverom, sistem mora omogućiti **ponovno povezivanje u roku od 30 sekundi** bez gubitka dosadašnjeg napretka u igri
6. **Pristupačnost -** Vizuelni interfejs mora koristiti jasne **boje** i fontove kako bi bio čitljiv svim korisnicima, kao I one sa slabim vidom

# Korisničke priče

Korisničke priče su detaljnije naracije koje na detaljniji način predstavljaju šta korisnik želi da system radi.

1. Kao igrač(korisnik), želim da vidim tablu sa svim poljima kao I kućicama I početnim pozicijama piuna. Kada igrač pokrene igru vidi traženu tablu sa svim poljima. Tabla se prikazuje pravilno sa svim poljima, početnim pozicijama I piunima.
2. Kao igrač, želim da bacim kocku kako bih odredio broj polja za pomeranje. Igrač klikne na sliku kocke i rezultat se prikazuje na kocki. Rezultat kocke je između 1 i 6, kretanje piuna se ažurira u skladu rezultatom
3. Kao igrač, želim da pomeram svoje piune po tabli prema bacanju kocke. Igrač baca kocku i pomera piune prema prikazanom broju. Piun se pomera pravilno I izlazi iz baze ili se kreće po tabli
4. Kao igrač, želim da mogu izbaciti protivnikovog piuna kada stanem na isto polje gde se on nalazi. Piun igrača staje na polje gde se nalazi protivnički piun. Protivnikov piun se vraća u bazu, a novi piun zauzima mesto starog koji se vratio u bazu
5. Kao igrač, želim da igram sa prijateljima online. Igrač se povezuje na server i čeka druge igrače dok se ne povežu. Igrač se uspešno povezuje, stanje igre je sinhronizovano između igrača
6. Kao igrač, želim da moj potez bude vidljiv svim igračima u realnom vremenu. Igrač pomeri piuna, drugi igrači vide promenu odmah. Potezi se prikazuju kod svih igrača u realnom vremenu, bez kašnjenja većeg od 1 sekunde
7. Kao igrač, želim da sistem prati čiji je red za potez. Sistem označava aktivnog igrača. Redosled poteza se automatski menja, svi igrači vide aktivnog igrača
8. Kao igrač, želim da se rezultati igre čuvaju u bazi. Kada se igra završi rezultati se upisuju u bazu. Rezultati se pravilno upisuju u bazu i mogu se videte na meniju
9. Kao igrač, želim da vidim pobednika na kraju igre. Kada igrač dovede sve pionire u cilj, sistem prikazuje pobednika. Pobeda se pravilno prikazuje, igra se završava i rezultati se čuvaju
10. Kao igrač, želim vizuelno da vidim kako se baca kocka. Kocka se okreće I dobija se broj na kocki. Animacija je kako se kocka vrti I na kraju posle dobijenog broja piun se pomera.
11. Kao igrač, želim da sistem automatski proverava validnost poteza. Igrač pokušava potez koji nije dozvoljen. Sistem blokira nevalidan potez i prikazuje obaveštenje
12. Kao igrač, želim da mogu videti prethodne statistiku pobede. Igrač vidi na meni sa statistiku prethodnih igara. Prikazuju se krug sa na kojem vidimo koja boja najviše pobeđuje

# ****Persona****

Ličnost (persona) oslikava vrstu korisnika koja bi koristila naš proizvod. O njima znamo demografske podatke, ciljeve, probleme, ponašanje I motivaciju.

**Persona 1**   
Marko, 22 godine, studira na Univerzitetu u Beogradu i provodi većinu vremena u učionici i biblioteci, što često čini njegov dan monotonim. Da bi se opustio između predavanja, igra igre pogotovo one koje može igrati online sa prijateljima. Njegov cilj je da mu vreme između pauza brže prođe i da se zabavi. Problem je što nema dovoljno vremena za fizičke društvene igre zbog gustog rasporeda, pa mu je bitno da igra bude jednostavna za pokretanje na laptopu.

Marko igra povremeno, ali često u kratkim intervalima i preferira multiplayer igre koje omogućavaju interakciju sa prijateljima. Svaka partija mu donosi drugačiji rezultat, što ga motiviše da ponovo igra i da se trudi da pobedi. Njegove bolne tačke uključuju komplikovan interfejs, sporo učitavanje i kašnjenje poteza, jer to narušava tok igre i smanjuje zabavu.

**Persona 2**  
Ana, 28 godina, je zaposlena majka koja radi u firmi za šivenje u Knjaževcu od 7h do 15h, pored posla završava svoje obaveze kod kuće. Većinu svog vremena provodi čuvajući svoju decu I na poslu. Cilj joj je kada se vrati kući da se opusti, ali sa decom je to teže, zbog toga potrebna joj je jednostavna igra u koju bi mogla uključiti I svoju decu u igri I ona da se zabave sa njom dok se igraju. Problem je kod dece što ne mogu dugo da sede na jednom mestu. Ana tokom igre voli da “pojede” piune svoje dece, jer ih tako zadražava zainteresovanim da ostanu I da igraju igru

Ana ne igra igru samo sa svojom decom, već I na poslu tokom svoje pauze za ručak sa drugim kolegenicama. SvNjena motivacija da igra je to što može da iskoristi svoje vreme sa decom na zabavan način, tako da nikome nije dosano.

**Persona 3**   
Nikola, 16 godina, srednjoškolac koji pohađa tehničku školu u Zaječaru želi da igra neku igru sa svojim drugarima tokom časa, iako to ne bi trebao da radi. Svoje vreme provodi u školi sa svojim drugarima na odmoru I ponekad ne zanju kako da se zabave. Cilj mu je da se na pauzi zabavi, zbog toga mu je potrebna igra gde može da se zabavi sa svojim vršnjacima, koja nije vremenski ograničena I može da se igra u bilo kom trenutku.

Nikola igra igre I kada je kod kuće, ne samo tokom pauza u školi. Dopada mu se to što je igra online I može da igra sa svojim prijateljima od kuće ili na bilo kojem mestu da se nalazi, što ga motiviše da igru iključi I bilo kom trenutku I zabavi se.

**Persona 4**   
Petar, 35 godina, je softverski inžinjer koji radi od kuće u Zrenjaninu, a firma u kojoj radi nalazi se u Kladovu. Već duže vreme se bavi softverskim inžinjerstvom, kada završi sa svojim poslom traži način da se upusti I zabavi. Cilj mu je da svoje vreme tokom posla provede sa opuštanjem, ali je problem što nema sa kim da igra igru, pošto svoj posao završava online, a drugovi igraju druge vrste igara. Zbgp toga mu je potrebna igra gde može da se samo uključi I igra sa drugima.  
  
Petar svoje vreme provodi kod kuće uz igranje igara ili gledanje sci-fi filmova. Nešto što ga motiviše da igra igre, jeste zabava koju donose i to što se opusti uz zabavu.

**Persona 5**   
Jelena, 12 godina, pohađa osnovnu školu u Nišu, želi da igra sa prijateljima i uči pravila društvenih igara. Posle škole voli da igra igra sa svojim starijim bratom I starijom sestrom. Problem je što teško razume komplikovane interfejse i pravila nekih igara, cilj je da igra jednostavnu i vizuelno razumljivu igru koja. Potrebna joj je igra gde je vizuelno jasno šta se očekuje od nje kao igrača.

Jelena igra kratko, često ostavi brau ili sestri da nastave umesto nje, ali uvek posle bržeg vremena nastavi sama da igra, jer vizuleno joj igra izgleda veoma zanimljivo što je motiviše da igra igru idalje.

# ****Scenariji za Persone****

Scenario je narativ koji opisuje situaciju u kojoj korisnik koristi proizvod da uradi nešto što želi. Scenario treba ukratko da objasni problem korisnika i predstavi zamišljeni način na koji bi problem mogao biti rešen.

Persona 1

1. Marko je student na Univerzitetu u Beogradu I ima 22 godine, ima naporan raspored pun predavanja i učenja u biblioteci. Između dva časa, obično ima oko 20–30 minuta slobodnog vremena i želi da se opusti. Umesto da sedi i gubi vreme, odlučuje da pokrene igru „Ne ljuti se čoveče“ na laptopu. Zbog jednostavnog interfejsa i brzom pokretanju, Marko može odmah da započne partiju. Problem sa kojim se susreo je da mora da čeka dok se drugi igrači ne povežu, ali I to vreme brzo prođe.
2. Nakon više sati učenja u biblioteci, Marko i njegovi kolege žele da se opuste. Umesto da traže fizičku tablu i figure, što često oduzima vreme i zahteva da svi budu fizički prisutni, oni koriste multiplayer opciju igre. Marko kreira privatnu sobu I drugi se samo priključe ako igraju preko iste mreže. Na taj način svi se povezuju online, čak i oni koji su otišli kući. Sistem rešava problem organizacije i omogućava im da u kratkom vremenu započnu zajedničku partiju, bez dodatnog stresa.
3. Marko voli osećaj takmičenja. Iako mu je primarni cilj zabava, svaki put kada igra, on želi da pobedi. Sistem prati njegove rezultate i prikazuje koliko puta je pobedio u odnosu na prijatelje. To mu daje dodatnu motivaciju da se vraća igri, jer svaki put može da doživi drugačiji ishod i pokuša da ostvari bolji rezultat. Problem gubljenja interesa rešen je kroz rezultate i takmičenja, što ga tera da igru koristi redovno.

Persona 2

1. Ana ima 28 godina i radi u firmi za šivenje u Knjaževcu i vraća se kući posle posla umorna. Iako želi da se odmori, njena deca žele pažnju i zabavu. Ana pokreće „Ne ljuti se čoveče“ na računaru i uključuje svoju decu u igru. Pošto deca ne mogu dugo da sede mirno, ona ih dodatno motiviše time što tokom partije često „pojede“ njihove pijune, što kod dece izaziva smeh i uzbuđenje. Na ovaj način deca ostaju zainteresovana da nastave da igraju, a Ana uspeva da se opusti dok se i dalje druži sa njima.
2. Tokom pauze za ručak na poslu Ana i njene koleginice često traže način da se opuste i nasmeju. Umesto da razgovaraju samo o svakodnevnim obavezama odlučuju da odigraju brzu partiju „Ne ljuti se čoveče“ na svojim telefonima. Jednostavan interfejs igre omogućava im da se brzo uključe i odigraju nekoliko poteza u vremenu koje imaju. Igra postaje izvor zabave i neformalne povezanosti među koleginicama, čineći radni dan manje stresnim.
3. Ana često ima situacije da paralelno završava kućne poslove i pazi na decu. Kada deca postanu nemirna, ona im predlaže da pokrenu igru „Ne ljuti se čoveče“ i uključe je zajedno na laptopu. Dok igra traje, Ana uspeva da im posveti pažnju kroz poteze i reakcije u igri, a istovremeno može da napravi pauzu za obavljanje kraćih kućnih zadataka između poteza. Tako igra postaje način balansiranja između obaveza i porodične zabave.

Persona 3

1. Nikola želi da se zabavi tokom pauze između časova. On i njegovi prijatelji žele da igraju igru zajedno, ali njihovi uređaji nisu svi povezani na istu mrežu. Problem je što postojeći školski uređaji i mreža nisu optimizovani za multiplayer online igre. Rešenje je omogućavanje igre koja koristi lokalnu mrežu, tako da se igra može igrati bez potrebe za stalnim internetom i omogućava brzo povezivanje među vršnjacima.

Persona 4

1. Petar je softverski inženjer iz Zrenjanina koji ima 35 godina I radi od kuće i veći deo dana provodi za računarom. Tokom dužih radnih sati često mu je potrebna kratka pauza kako bi se opustio. Problem je što nema društvo za klasične društvene igre, jer njegovi prijatelji preferiraju druge žanrove igara. Petar želi brzu i jednostavnu igru kojoj može da se priključi za nekoliko minuta i zatim da se vrati poslu odmoren. Zbog toga se uključuje u multiplayer verziju „Ne ljuti se čoveče“. Igra mu omogućava da odmah pronađe protivnike bez dodatnog dogovaranja.
2. Posle završetka posla Petar često ostaje kod kuće i opušta se uz filmove ili igre. Međutim, filmovi mu nekad deluju dosadno, a želi nešto što uključuje interakciju. Pošto nema prijatelje koji su raspoloženi za igranje „Ne ljuti se čoveče“, povezuje sa online i igra protiv nepoznatih protivnika. Na ovaj način dobija osećaj društvene interakcije i zabave, iako fizički igra sam kod kuće.
3. Petar ima rodbinu i prijatelje koji žive u drugim gradovima i retko se viđaju. Pošto im nedostaje druženje, on im predlaže da zajedno igraju „Ne ljuti se čoveče“ preko online verzije. Tako može da se ali kako u igri nemaju mogućnost da se čuju glasovno(voice chat), moraju da koriste spoljašnji program.

Persona 5

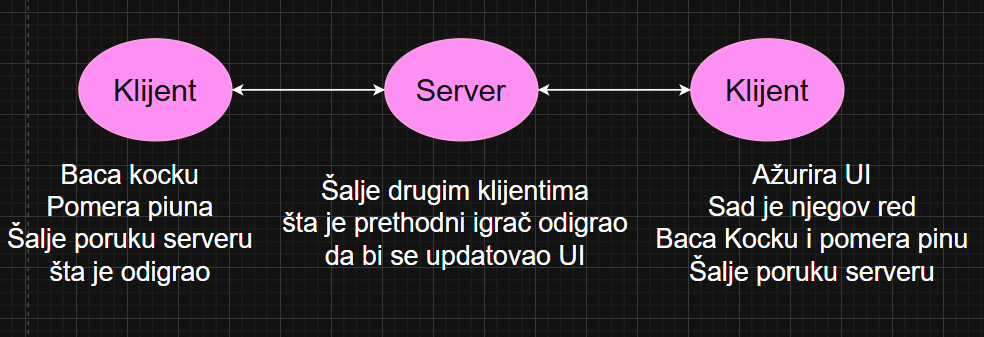
1. Jelena ima 12 godina I pohađa osnovnu školu, želi da se igra sa svojim školskim prijateljima kako bi učila pravila društvenih igara i uživala u zabavi. Problem je što složeni interfejsi i komplikovana pravila otežavaju razumevanje i frustriraju je. Rešenje je jednostavna i vizuelno razumljiva igra koja jasno prikazuje šta se očekuje od igrača, omogućavajući Jeleni da se brzo uključi i zabavi sa prijateljima bez konfuzije.

# ****Stilovi softverske arhitekture****

## Klijent server

Klijent-server(Slika 0.1.) arhitektura spada u najpoznatije i najčešće korišćene stilove u distribuiranim aplikacijama. Osnovna ideja je jasna podela na dve uloge: **server**, koji je centralna tačka sistema, i **klijent**, koji se povezuje na server i koristi njegove usluge. Klijent šalje zahteve, server ih obrađuje i vraća odgovore.

“Ne ljuti se, čoveče” ima server koji vodi celu partiju, dodeljuje igračima boje, kontroliše redosled poteza, proverava pravila i obaveštava klijente o svim promenama. Klijenti predstavljaju korisnike igre: oni bacaju kocku, pomeraju figure i šalju svoje poteze serveru, a zatim od njega dobijaju informacije o potezima ostalih igrača. Ovakav pristup omogućava da svi igrači budu sinhronizovani i da pravila igre budu striktno poštovana, jer se sva logika igre izvršava na serveru.



Slika 0.1. Dijagram Klijent-Server

Prednosti klijent-server arhitekture su pre svega u centralizovanoj kontroli i jednostavnosti sinhronizacije. Pravila igre su implementirana samo na jednom mestu, čime se smanjuje mogućnost grešaka i nekonzistentnosti. Takođe klijent-server model omogućava jednostavnije održavanje i nadogradnju jer je dovoljno promeniti kod na serveru, dok klijenti mogu ostati nepromenjeni.

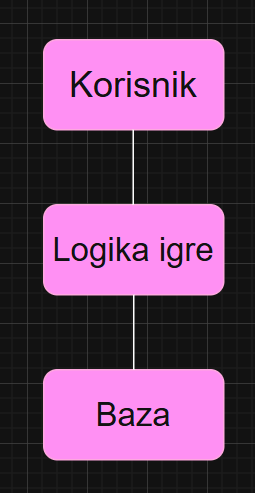
Mane su ako server prestane da radi, cela igra automatski staje jer klijenti zavise od njega. Svako kašnjenje u komunikaciji ili prekid veze direktno utiče na iskustvo igrača i može dovesti do problema sa sinhronizacijom. Skalabilnost takođe predstavlja problem, jer proširenje kapaciteta servera zahteva složena rešenja poput dodavanja dodatnih servera i balansiranja opterećenja. Fleksibilnost je ograničena, pošto se logika igre nalazi na serveru, pa svaka izmena pravila zahteva njegovo ažuriranje i ponovno pokretanje.

## Mikroservisi

Servisno-orijentisana(Slika 0.2.) arhitektura zasniva se da aplikacija ne mora biti jedan monolitni sistem već se može sastojati od više manjih, nezavisnih servisa. Svaki servis pruža određenu funkcionalnost i komunicira sa ostalima preko dobro definisanih interfejsa. Mikroservisi su moderna implementacija ovog stila i omogućavaju visoku skalabilnost i fleksibilnost.

“Ne ljuti se, čoveče” mikroservisi trenutno nisu primenjeni, ali bi se mogli koristiti kao ideja za buduće proširenje. Na primer, sistem bi mogao biti podeljen na sledeće servise:

* **Servis za korisnike** – upravlja igračima, autentifikacijom i njihovim profilima.
* **Servis za igre** – vodi pojedinačne partije i primenjuje pravila igre.
* **Servis za rezultate i statistiku** – čuva i obrađuje podatke o pobednicima i koracima.



Slika 0.2. Mikroservis

Ovo bi omogućilo da se igra ne ograničava samo na lokalnu mrežu, već da podrži hiljade igrača preko interneta. Svaki servis mogao bi se razvijati i skalirati nezavisno od ostalih, čime bi se povećala otpornost i fleksibilnost sistema.

Prednosti ovog pristupa su kod velikih sistema: skalabilnost, nezavisno razvijanje servisa, jednostavnije održavanje i bolja otpornost na greške. Umesto jedne aplikacije sistem se sastoji od niza manjih servisa koji međusobno komuniciraju, što zahteva posebnu koordinaciju. Ovakav pristup povećava i troškove infrastrukture, jer je neophodno održavati više servera ili kontejnera, a to podrazumeva korišćenje naprednih DevOps alata kao što su Docker ili Kubernetes.

# ****Funkcionalnosti aplikacije****

Aplikacija predstavlja digitalnu verziju društvene igre "**Ne ljuti se čoveče**" (Ludo), implementiranu u JavaFX okruženju. Omogućava igru za do četiri igrača putem mrežne komunikacije, sa vizuelnim prikazom table, bacanjem kocke, pomeranjem figura i evidentiranjem pobednika.

## Tabla i pozicije piuna

### Klasa BoardPosition

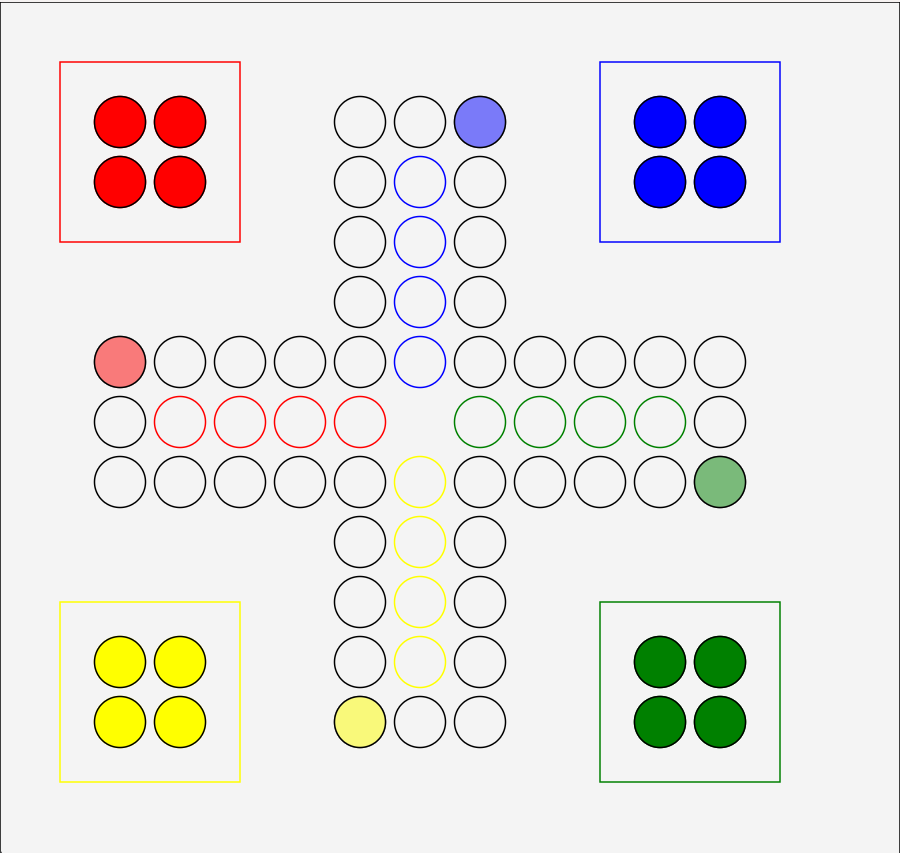
Definiše ključne koordinate na tabli, uključujući:

* **Putanju piona (pathCoordinates)** - Matrica sa svim pozicijama kroz koje se figure kreću.
* **Startne pozicije (redStartPosition, blueStartPosition, greenStartPosition, yellowStartPosition)** - Lokacije na tabli odakle svaka boja započinje igru.
* **Domaće pozicije (redHome, blueHome, greenHome, yellowHome)** - Mesta gde se figure nalaze pre nego što krenu u igru.
* **Ciljne pozicije (redGaol, blueGaol, greenGaol, yellowGaol)** - Zona gde figure dolaze kada završe svoj put.

### Klasa BoardDrawer

Implementira vizuelni prikaz table(slika 1.):

* Crta kvadrate za domove I ciljeve igrača.
* Crta kružni put kroz koji se figure kreću.
* Prikazuje sve pozicije I pomaže u identifikaciji lokacija.
* Omogućava dodavanje figura na tablu.



Slika 1. Vizuelni prikaz table

## Bacanje Kocke

Klasa Dice (slika 2.)

* Kreira vizuelni prikaz kocke koristeći ImageView.
* Klikom na kocku, ako je igračev red, generiše nasumičan broj od 1 do 6.
* Implementira animaciju bacanja kocke gde se prikazuju različiti brojevi pre finalnog rezultata.
* Ako igrač pokuša da baci kocku van svog poteza, prikazuje se poruka upozorenja.

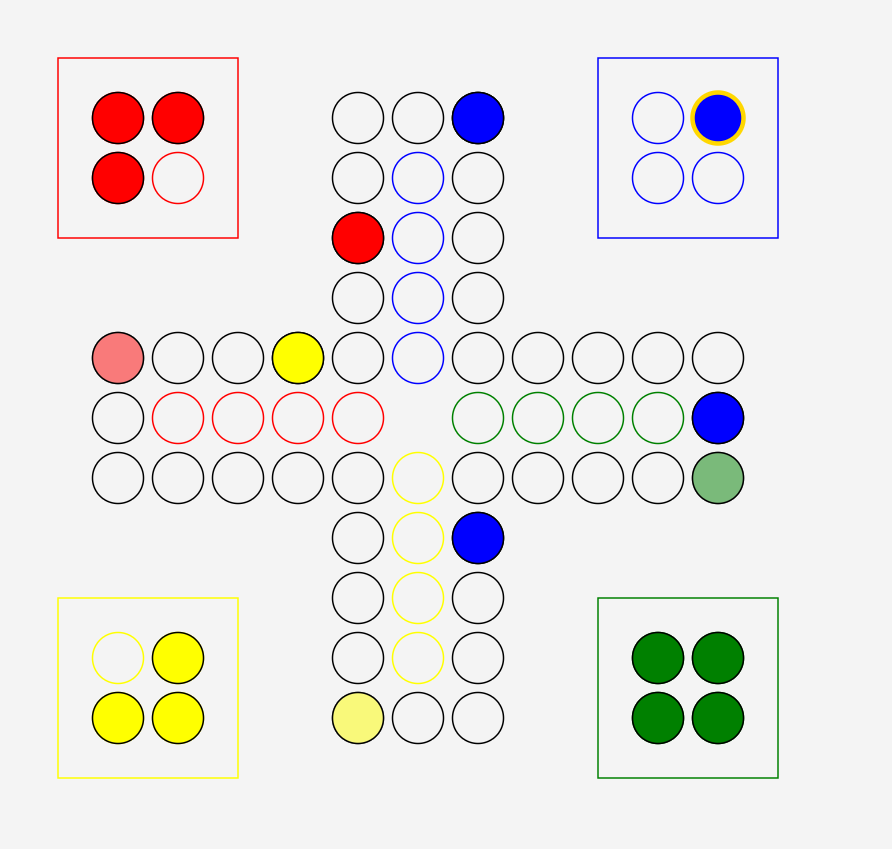


Slika 2. Prikaz kocke

## Pomeranje piuna

Klasa Pawn (slika 3.)

* Svaki pion ima boju, jedinstveni ID i trenutno stanje (da li je kod kuće, na tabli ili u cilju).
* Klikom na pion, proverava se da li igrač može da pomeri tu figuru.
* Ako je pion u kući, može se izbaciti samo bacanjem broja 6.
* Pomeranjem, pion menja svoju poziciju prema tabeli BoardPositions.
* Ako pion dođe na polje zauzeto od strane protivničkog piona, vraća ga u kuću.
* Ako pion dođe do cilja, završava igru.



Slika 3. Pomeranje piuna

## Upravljanje igrom

### Klasa PawnMenager

* Prati listu svih piona i njihovih trenutnih pozicija.
* Proverava da li se može baciti kocka.
* Obezbeđuje pravilno menjanje poteza između igrača.
* Određuje redosled kretanja i osigurava da svaki igrač igra po pravilima.
* Proverava da li je neko pobedio i proglašava pobednika.
* Prati koje boje su zauzete i menja prikaz tabele u skladu s tim.

## Mrežna komunikacija

### Klasa Server

* Pokreće server na lokalnom računaru.
* Prati broj povezanih igrača i dodeljuje im boje.
* Prima poruke od klijenata i prosleđuje ih svima.
* Menja potez nakon što igrač završi kretanje.
* Prati pobednika i obaveštava sve igrače kada neko pobedi.

### Klasa Client

* Povezuje se na server preko lokalne mreže.
* Šalje poteze i rezultate bacanja kocke serveru.
* Primalac poruka koje mu omogućavaju da prati poteze drugih igrača.
* Proverava kada je red na njega i omogućava mu da igra.

## Baza podataka

### Klasa Database

* Omogućava povezivanje sa MySQL bazom podataka.
* Čuva ime pobednika u tabeli winners.
* Omogućava kasniji pregled rezultata.

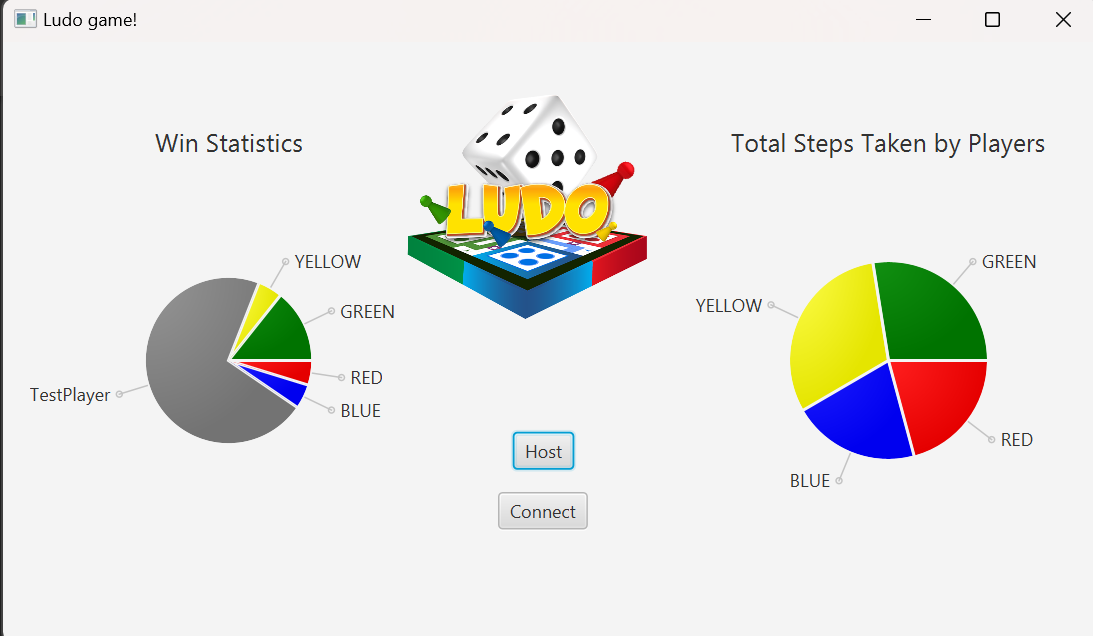
## Grafički interfejs

### Klasa Menu

* Omogućava igračima da započnu igru kao server ili klijent.
* Sadrži dugmad za pokretanje igre.
* Prikazuje logo igre.

### Klasa Chart

* Vizuelni prikaz pie charts.
* Prikazuje krugove i odnos koliko je koji igrač pobedio i totalSteps napravio.
* Uvek se ažurira na kraju igre.



Slika 4. Grafički prikaz

### Klasa Game

* Kreira vizuelni prikaz igre.
* Prikazuje tablu, kocku i poruke o potezima.
* Prati redosled igrača i osigurava da svi prate pravila igre.

## Prikaz poruka i obaveštenja

### Klasa AlertMessage

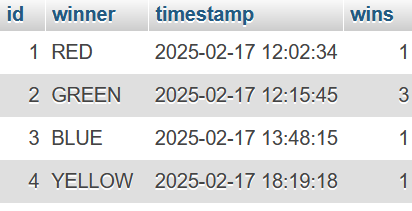
* Prikazuje poruke greške ako igrač pokuša neispravan potez.
* Obaveštava igrače kada pion ne može ući u cilj.
* Prikazuje pobednika igre kada neko završi sa svim figurama.

# Tehnički detalji

Struktura Baze podataka

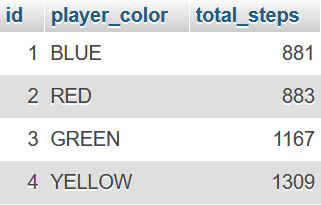
Aplikacija koristi **MySQL** bazu podataka sa sledećim tabelama:

1. winners – čuva pobednike i broj pobeda(slika 5.).
   * winner (VARCHAR) – ime pobednika.
   * wins (INT) – ukupan broj pobeda.

****

Slika 5.

1. player\_steps – beleži ukupan broj koraka svakog igrača(slika 6.).
   * player\_color (VARCHAR) – boja igrača.
   * total\_steps (INT) – ukupan broj pređenih koraka.



Slika 6.

1. game\_dice\_rolls – skladišti podatke o bacanjima kockice(slika 7.).
   * game\_id (INT) – ID igre.
   * player (VARCHAR) – boja igrača koji baca kocku.
   * dice\_roll (INT) – broj koji je pao na kocki.



Slika 7.

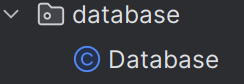
**Relacije:**

* winners tabela čuva istoriju pobednika.
* player\_steps prati ukupan broj koraka po igraču tokom igre.
* game\_dice\_rolls čuva podatke o bacanjima kocke svakog igrača.

# Arhitektura Aplikacije

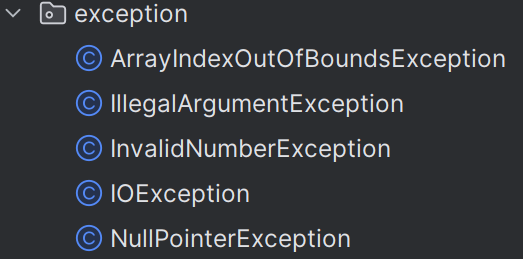
Aplikacija je podeljena u 5 paketa:

* database – interakcija sa bazom podataka(slika 8.).



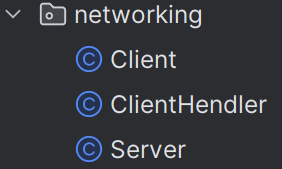
Slika 8.

* exception– klasa sa izuzeci(slika 9.).



Slika 9.

* networking – klijent-server komunikacija(slika 10.).



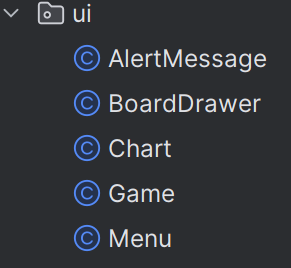
Slika 10.

* game – logika igre, upravljanje figurama, bacanje kocke(slika 11.).



Slika 11.

* UI – JavaFX korisnički interfejs(slika 12.).



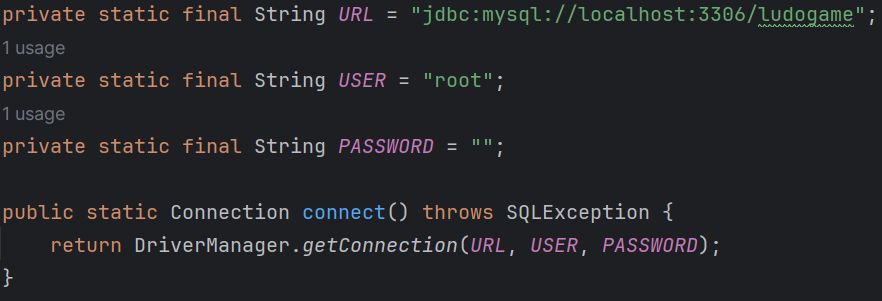
Slika 12.

# Ključne klase I njihove odgovornosti

## Database

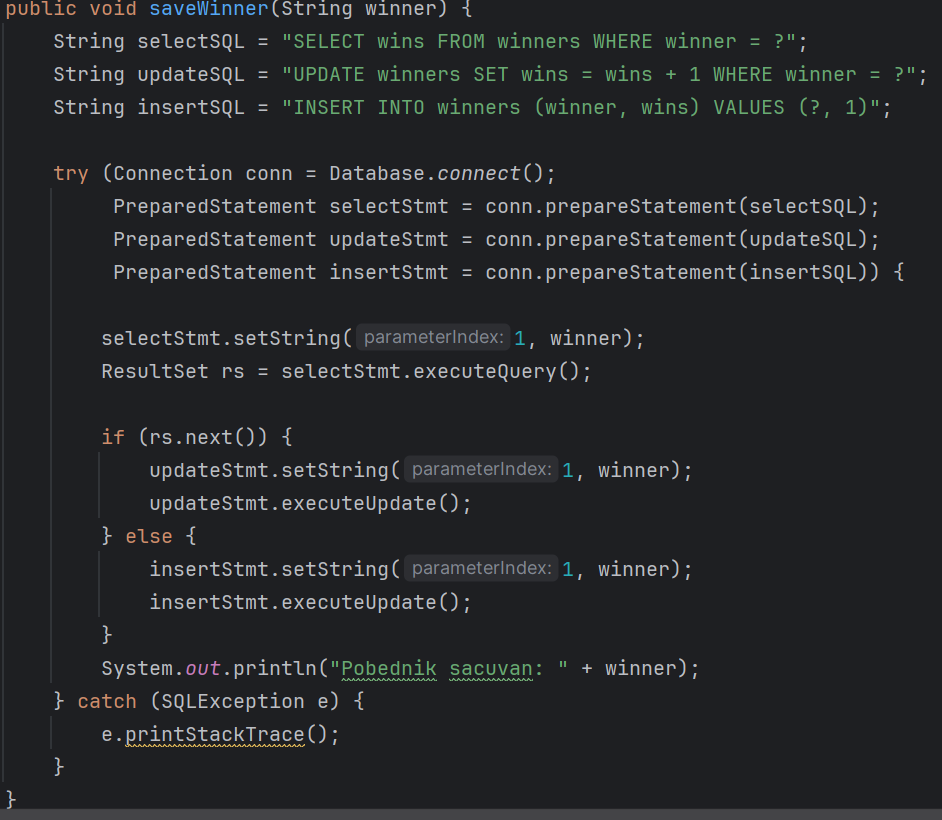
Klasa služi za upravljanje bazom podataka u igri Ludo. Omogućava povezivanje sa MySQL bazom, čuvanje i dohvaćanje podataka o pobednicima, broju koraka igrača i bacanjima kocke.

Connect(slika 13.) metoda otvara konekciju sa bazom podataka koristeći DriverManager.getConnection() I vraća objekat Connection koji omogućava rad sa bazom, ako dođe do greške, baca SQLException.



Slika 13. connect()

SaveWinner(Slika 14.) služi za čuvanje ili ažuriranje broja pobeda igrača. Prvo proverava da li igrač već postoji u tabeli winners, ako postoji, povećava broj pobeda za 1, uostalom ako ne postoji, ubacuje novog igrača sa wins = 1.



Slika 14. saveWinner()

getStepCount(slika 15.) vraća mapu sa bojama igrača i ukupnim brojem pređenih koraka. SQL upit SELECT player\_color, SUM(total\_steps) as steps FROM player\_steps GROUP BY player\_color sabira korake svih igrača iste boje. Rezultati se dodaju u mapu (<boja, broj koraka>) i vraćaju pozivaocu.



Slika 15. getStepCounts()

## Server

Klasa Server implementira serversku stranu igre Ludo. Omogućava povezivanje do 3 klijenta (igrača) uz fiksiranog servera kao četvrtog igrača (crveni igrač). Ova klasa komunicira sa klijentima pomoću klase ClientHendler i upravlja dodeljivanjem boja igračima, kao i slanjem poruka svim povezanim klijentima

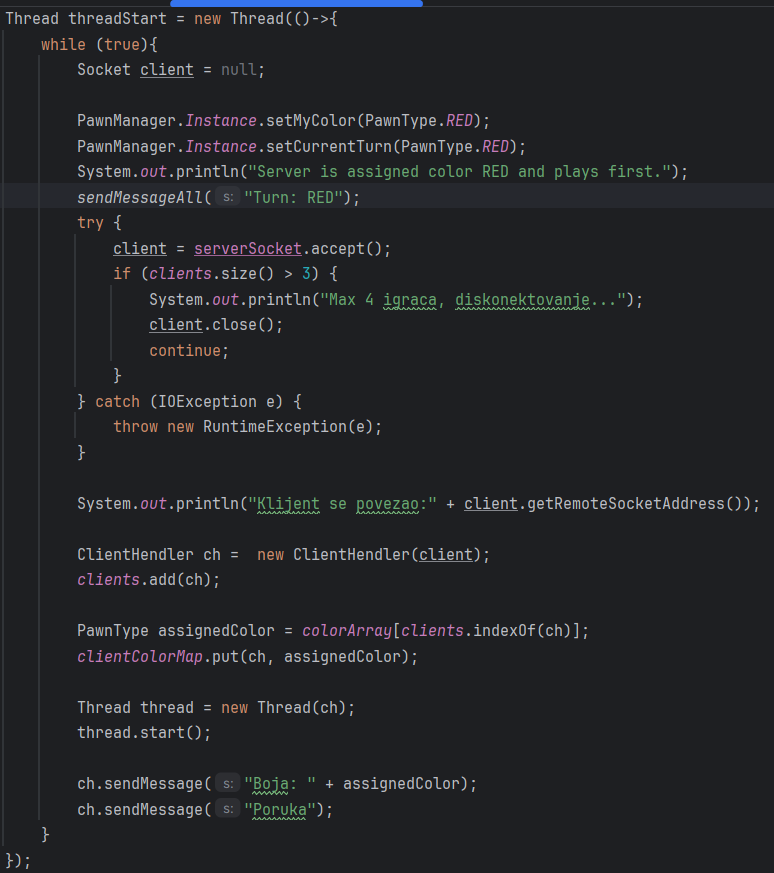
Prvo kreira server socket na zadatom portu (5529) i ispisuje poruku da je server pokrenut(slika 16.).



Slika 16. Pokretanje servera

Zatim pokreće beskonačnu petlju u posebnoj niti koja(slika 17.):

* Dodeljuje serveru boju RED i postavlja njegov prvi potez.
* Osluškuje dolazne klijentske konekcije.
* Odbija klijenta ako je već povezan maksimalan broj igrača (4).
* Kreira novog ClientHendler objekta za svakog klijenta.
* Dodeljuje igraču odgovarajuću boju iz colorArray.
* Pokreće nit za klijenta i šalje mu početne poruke.

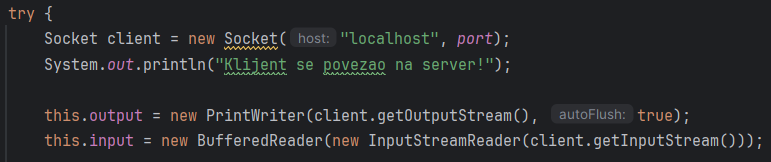


Slika 17.

## Client klasa

Klasa predstavlja klijentsku stranu mrežne komunikacije. Klijent se povezuje na server, sluša dolazne poruke i obrađuje ih kako bi ažurirao stanje igre.

Povezuje se na server koji radi na localhost 5529 i postavlja output i input tokove za slanje i primanje podataka(slika 18.).



Slika 18.

Pokreće nit koja **sluša dolazne poruke** i obrađuje ih. Zatim klijent obrađuje poruke koje dobija od servera i menja stanje igre na osnovu njih.

Ako server pošalje poruke u vezi sa **pomeranjem piuna**, poziva odgovarajuće metode u PawnManager(slika 19.).



Slika 19.

## PawnManager

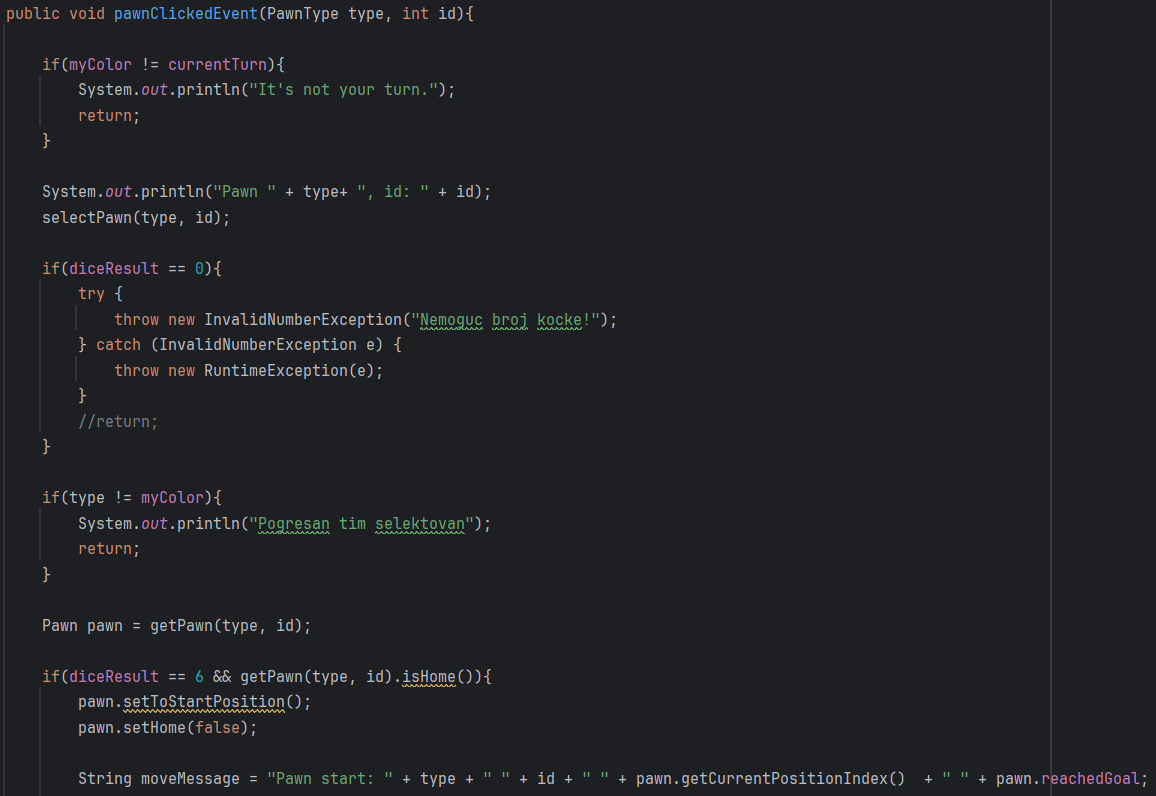
Klasa PawnManager je zadužena za upravljanje piunima u igri i koordinaciju njihovih poteza u skladu sa pravilima igre. Takođe, komunicira sa serverom i klijentom kako bi se sinhronizovali potezi u mrežnom modu igre.

pawnClicledEvent(slike 20. i 21.) proverava da li je igračev potez, ako jeste igrač može da baci kocku i zatim proverava da li je kocka bačena (diceResult mora biti veći od 0).

Ako je piun kod kuće i igrač dobije 6, piun se postavlja na startnu poziciju, uastalom ako figura nije kod kuće, pomera piuna na osnovu rezultata kocke.

Metoda takođe provera da li je neki piun stao preko drugog piuna(checkPawnOverlap). Zatim ažurira mrežnu igru slanjem poruka Server.sendMessageAll() ili Client.sendMessage().

Ako igrač nije dobio 6, prelazi na sledeći potez (nextTurn()).



Slika 20.

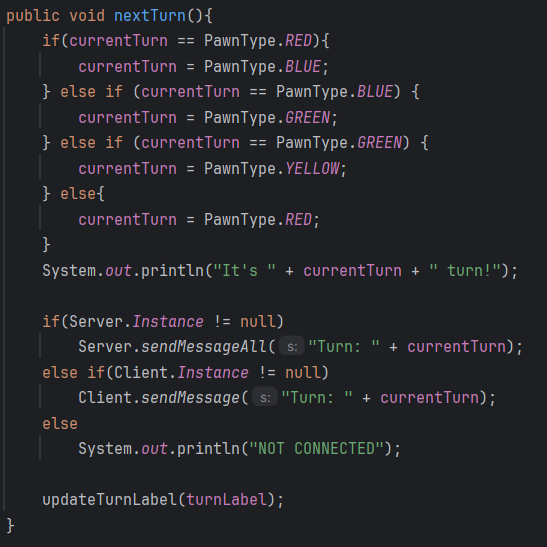


Slika 21.

nextTurn(slika 22.) rotira potez među igračima u redosledu:

RED -> BLUE -> GREEN -> YELLOW -> RED

Ažurira labelu sa informacijom o trenutnom igraču i šalje poruku serveru ili klijentu o tome ko je na potezu.



Slika 22.

## Pawn

Klasa Pawn predstavlja piuna u JavaFX igri na tabli. Svaki piun ima svoju boju, poziciju na tabli i može se kretati, vraćati kući i dostići cilj. Takođe, podržava mrežnu komunikaciju kroz **server i klijent** sisteme.

Pawn(slika 23.) kreira piuna određene boje (color) i pozicije kod kuće (homePosition). Inicijalizuje pawnUI u zavisnosti od boje i postavlja event listener za klik na pijuna (pawnUI.setOnMouseClicked) i postavlja statičku instancu Instance.

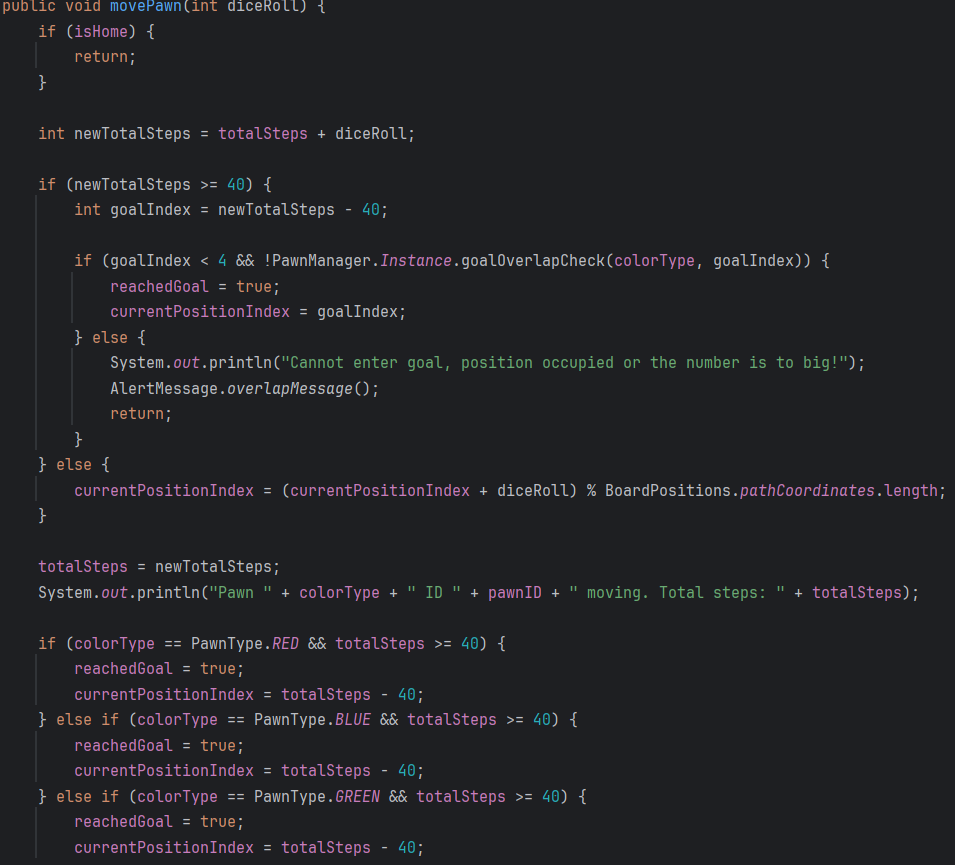


Slika 23.

movePawn(slika 24.) pomera piuna unapred na osnovu rezultata bacanja kocke (diceRoll). Ako dostigne 40 koraka, ulazi u ciljnu zonu i proverava da li je ciljna pozicija zauzeta (PawnManager.Instance.goalOverlapCheck).

Ako nije dostigao cilj, ažurira pawnUI pomoću Platform.runLater(), koji osigurava da se UI ažurira u glavnoj niti JavaFX-a.

Ako piun stigne do cilja, šalje poruku svim igračima preko Servera/Klienta i proverava pobednika (PawnManager.Instance.checkWinner())(slika 24.).



Slika 23.



Slika 24.

# Ponovno upotrebljive komponente

Jedan od važnih ciljeva u softverskom inženjerstvu jeste razvijanje komponenti koje se mogu ponovo koristiti u različitim projektima. Takav pristup smanjuje vreme razvoja, povećava pouzdanost sistema i doprinosi održivosti projekta.

* 1. **Klasa za bacanje kocke (Dice)**  
     Komponenta Dice omogućava generisanje nasumičnih brojeva od 1 do 6 i vizuelni prikaz kocke. Ova klasa je univerzalna jer svaka igra koja koristi element slučajnosti (igre na tabli, igre karata i simulacije) može upotrebiti ovaj generator. Ponovna upotreba smanjuje potrebu da se iznova implementira funkcija za slučajne brojeve i animacije, čime se štedi vreme i povećava konzistentnost.
  2. **Klasa za upravljanje bazom podataka (Database)**  
     Komponenta Database obezbeđuje konekciju sa MySQL bazom, upis i čitanje podataka. Ova klasa može se koristiti u različitim aplikacijama koje zahtevaju čuvanje rezultata, evidenciju korisnika ili bilo koju CRUD operaciju. Njena ponovna upotreba obezbeđuje standardizovan način rada sa bazom i smanjuje rizik od grešaka.
  3. **Komponenta za mrežnu komunikaciju (Server i Client)**  
     Server i klijent implementirani su pomoću socket programiranja, što omogućava slanje i primanje poruka između više aplikacija. Komunikaciona logika može se koristiti u bilo kojoj multiplayer igri, ali i u širem spektru aplikacija koje zahtevaju razmenu podataka preko mreže. Ponovna upotreba smanjuje kompleksnost razvoja novih projekata koji zahtevaju umrežavanje.
  4. **Klasa za upravljanje igračima i figurama (PawnManager)**  
     PawnManager koordinira poteze igrača, proverava pravila i menja redosled poteza. Iako je razvijen specifično za „Ne ljuti se, čoveče“, osnovna logika turn-based igre može se iskoristiti i u drugim igrama koje imaju sličan princip. Omogućava brža implementacija novih igara sa sličnim pravilima.
  5. **Grafički interfejs za tablu i figure (BoardDrawer, Pawn)**  
     BoardDrawer i Pawn zadužene su za vizuelni prikaz table i figura. Iako su prilagođene ovoj igri, koncept vizuelizacije objekata na tabli može se koristiti i u drugim igrama na mreži (šah, dame, monopol). Njihova ponovna upotreba smanjuje vreme razvoja i olakšava implementaciju novih projekata sa sličnom logikom prikaza.
  6. **Klasa za statistiku i vizuelizaciju (Chart)**  
     Chart obezbeđuje vizuelni prikaz rezultata putem pie chart-a. Ovaj element je generički i može se koristiti u raznim aplikacijama koje prikazuju statistiku ili poređenja (poslovne aplikacije, edukativni sistemi). Njegova ponovna upotreba smanjuje potrebu za dodatnim bibliotekama i obezbeđuje konzistentnost u prikazu podataka.

# ****Računarstvo u oblaku****

**Računarstvo u oblaku** je virtuelni server koji se nalazi na udaljenim računarima koje poseduju kompanije poput Amazona ili Google-a. Umesto da koristiš svoj fizički hardver, možeš iznajmiti onoliko servera koliko ti treba, instalirati softver ili koristiti gotove aplikacije u oblaku. Virtuelni serveri rade na moćnom hardveru uz podršku za virtuelizaciju, što omogućava više servera da rade istovremeno bez velikog uticaja na performanse. Softver za upravljanje oblakom olakšava skaliranje, automatske nadogradnje i otpornost na kvarove, a plaća se po potrebi ili na ugovoreni period.

## Multiplayer server logika

Server koji održava stanje igre, uključujući pozicije piona, redosled poteza, bacanje kocke i pravila igre, najbolje funkcioniše na **Platform as a Service (PaaS)** ili **Infrastructure as a Service (IaaS)** oblaku. PaaS je idealan jer omogućava brzo postavljanje i skaliranje server aplikacije bez brige o fizičkom hardveru, dok IaaS nudi potpunu kontrolu nad serverom i mrežnom konfiguracijom, što je korisno za specifične zahteve igre, ali zahteva više tehničkog rada. Računarstvo u oblaku omogućava automatsko skaliranje broja instanci servera kada veći broj igrača igra simultano, čime se osigurava glatko i pouzdano iskustvo igranja.

## Čuvanje rezultata i statistike (Highscore, statistika partija)

Bazu podataka koja čuva podatke o igračima, njihovim rezultatima i istoriji partija najpogodniji je **Database as a Service (DBaaS)**, što je deo PaaS-a. DBaaS omogućava lako kreiranje i skaliranje baze podataka (MySQL ili MongoDB u oblaku), gde se podaci čuvaju centralizovano i dostupni su svim klijentima igre. Automatski upravlja kopijama i dostupnošću baze, što povećava pouzdanost i smanjuje potrebu za ručnim održavanjem.

## Hosting klijentske aplikacije

Web ili mobilna aplikacija koja se pokreće kod igrača najbolje se hostuje preko **Software as a Service (SaaS)** ili **Content Delivery Network (CDN)**. Ako je igra dostupna kao web-aplikacija, CDN omogućava distribuciju statičkog sadržaja (HTML, CSS, JS, slike) što je bliže korisniku i smanjuje kašnjenje. SaaS pruža direktan pristup igri preko browsera ili mobilne aplikacije, bez potrebe za instaliranjem ili održavanjem servera od strane krajnjeg korisnika, čime se olakšava korišćenje i povećava dostupnost aplikacije.

# ****Mikroservisi****

## Servis za korisnike

Servis za korisnike upravlja igračima, autentifikacijom i njihovim profilima. Njegova prednost je što omogućava nezavisno skaliranje i povećava sigurnost, jer se podaci o korisnicima i pristup sistemu odvaja od logike igre i statistike. Izazovi uključuju potrebu za visokom dostupnošću i pouzdano upravljanje sesijama, jer svaki igrač mora biti autentifikovan pre nego što učestvuje u partiji. U arhitektonskom dijagramu, ovaj servis se povezuje sa servisom za igre radi autentifikacije igrača pre početka partije, a takođe komunicira sa servisom rezultata kako bi povezivao rezultate sa korisničkim profilima.

## Servis za igre

Servis za igre vodi pojedinačne partije, primenjuje pravila igre i prati pozicije piona i redosled poteza. Prednosti uključuju nezavisno skaliranje za svaku partiju i mogućnost lakše nadogradnje pravila ili dodatnih funkcionalnosti bez uticaja na ostale servise. Izazovi su koordinacija i sinhronizacija sa servisom korisnika i servisom rezultata, posebno kada više igrača igra istovremeno i njihovi potezi moraju biti pravilno raspoređeni. U arhitektonskom dijagramu, servis za igre prima podatke od servisa za korisnike i šalje informacije servisu rezultata radi beleženja toka partije i pobednika.

## Servis za rezultate i statistiku

Servis za rezultate i statistiku čuva i obrađuje podatke o pobednicima i koracima igrača. Prednosti uključuju nezavisno skaliranje baze i mogućnost obrade analitike i generisanja izveštaja bez opterećenja primarne logike igre. Izazovi su obezbeđivanje konzistentnosti podataka, jer više servisa može istovremeno upisivati rezultate, i pouzdana integracija sa servisom za igre i servisom korisnika. U arhitektonskom dijagramu, ovaj servis prima informacije od servisa za igre i koristi podatke iz servisa za korisnike kako bi statistika bila tačno povezana sa profilima igrača.

# ****Atributi kvaliteta sistema****

Atributi kvaliteta su **osobine sistema** koje opisuju koliko je sistem „dobar“ iz perspektive korisnika i programera, a ne samo šta sistem radi. Za razliku od funkcionalnosti koja definiše šta sistem može da uradi, atributi kvaliteta se odnose na **kako sistem to radi.**

1. **Sigurnost** je ključni atribut jer igra ima rezultate. Ako sistem nije siguran, može doći do curenja ličnih podataka ili manipulacije rezultatima, što direktno narušava poverenje korisnika. Implementacija enkripcije i autentifikacije smanjuje rizik napada i povećava pouzdanost celog sistema.
2. **Performanse** su bitne, jer sistem mora brzo reagovati na akcije igrača, posebno u multiplayer okruženju. Kašnjenja u ažuriranju stanja igre ili u prikazivanju rezultata mogu dovesti do lošeg korisničkog iskustva. Optimizacija mrežnih protokola i baze podataka omogućava bržu obradu zahteva i fluidniju igru.
3. **Skalabilnost** obezbeđuje da sistem može podržati rast broja igrača. Kako broj partija raste, serveri i baza podataka moraju biti sposobni da automatski povećaju resurse bez prekida u radu. Korišćenje mikroservisa i cloud infrastrukture omogućava horizontalno skaliranje i bolju dostupnost igre širom sveta.
4. **Pouzdanost** garantuje da će sistem raditi konzistentno i bez prekida, čak i u slučaju grešaka ili opterećenja. Ako server padne usred partije, igrači gube poverenje u igru. Implementacija mehanizama za toleranciju na greške, rezervne kopije i failover servere povećavaju stabilnost sistema.
5. **Upotrebljivost** utiče na to koliko je igra laka i intuitivna za korišćenje. Ako korisnici ne mogu lako pokrenu igru ili provere rezultate, napustiće igru. Dobro dizajnirani interfejs i jasna navigacija obezbeđuju dobro iskustvo i povećavaju angažovanost korisnika.
6. **Održavanje** je važno za dugoročni razvoj i unapređenje sistema. Ako je kod komplikovan i slabo dokumentovan, svaka izmena je rizična. Modularni dizajn i mikroservisi olakšavaju dodavanje novih funkcionalnosti i smanjuju troškove održavanja.

# ****Scenariji testiranja****

Scenariji testiranja sadrže opis funkcionalnosti, ulazne podatke, očekivane rezultate i varijacije ulaza kada je to potrebno. Omogućava proveru da li sistem ispunjava definisana pravila igre i zahteve performansi.

1. Kreiranje novog hosta  
   Kreiranja nove multiplayer partije, prvi igrač bira opciju Host. Sistem pravi praznu partiju sa 4 mesta i automatski ubacuje prvog igrača kao učesnika(što je zapravo host). Očekivan rezultat je da je host napravljen I da mogu I drugi igrači da se priključe.
2. Bacanje kocke  
   Tokom igre svaki igrač na svom potezu bira opciju „Baci kocku“. Sistem tada generiše slučajan broj između 1 i 6 i na osnovu dobijenog rezultata pomera piuna tog igrača za odgovarajući broj polja unapred. Na ovaj način se simulira realno bacanje kocke i omogućava nastavak igre prema pravilima. Očekivani rezultat je uvek isti: broj mora biti u granicama 1–6, a piun se pomera tačno onoliko polja koliko je prikazano.
3. Pridružavanje hostu  
   Kada drugi igrač želi da se pridruži postojećoj partiji, bira opciju “Connect”. Sistem proverava da li u toj partiji još ima slobodnih mesta i ako ima dodaje je ga kao igrač. Ako je partija već popunjena, sistem prikazuje poruku o grešci i sprečava pridruživanje. Očekujemo da u partiji uvek učestvuje tačno 4 igrača.
4. Pomeranje piuna  
   Kada igrač dobije broj na kocki, njegov piun se pomera tačno za onoliko polja koliko je prikazano. Sistem zatim proverava da li je polje na koje je piun sleteo zauzeto od strane protivnika. Ako jeste, primenjuje se pravilo „izbacivanja“, što znači da se protivnički piun vraća na početnu poziciju. Očekujemo da pomeranje piuna uvek odgovara rezultatu bacanja kocke, uz poštovanje pravila igre o zauzetim poljima i izbacivanju protivnika.
5. Kraj igre  
   Kad jedan od igrača pomeri svog piuna u delu koji predstvalja “kuću” poslednjeg polja na tabli, sistem prepoznaje da je igrač stigao do cilja i proglašava ga pobednikom. Očekivani rezultat je da se na ekranu prikaže poruka koji je igrač pobedio, nakon čega se partija završava.
6. Skalabilnost servera  
   Ponašanje servera pri velikom opterećenju. Kada između 50 i 100 igrača simultano kreira i igra partije, očekuje se da server uspešno obradi sve zahteve, bez kašnjenja u radu i bez pada sistema. Ovako se proverava stabilnost i kapacitet sistema u realnim uslovima.
7. **Oporavka od greške servera**Ako dođe do pada ili restartovanja servera tokom aktivne partije, sistem treba da obezbedi nastavak igre iz poslednjeg poznatog stanja. Očekujemo da igrači ne bi smeli da izgube napredak, a partija bi trebalo da se automatski obnovi i nastavi čim server ponovo proradi.
8. **Izbacivanje protivničkog piuna**Kad igrač pomeri svog piuna na polje na kojem se već nalazi piun protivnika, sistem primenjuje pravilo „izbacivanja“. Očekivani rezultat je da se piun protivnika vrati na startnu poziciju, a piun aktivnog igrača zauzme to polje. Sistem dodatno može da prikaže animaciju ili poruku o izbacivanju. Nakon toga, partija se nastavlja normalno, bez gubitka stanja drugih piuna. Testiranje obuhvata više ulaza: polje zauzeto jednim piunom, polje zauzeto više piuna (ako pravila to dozvoljavaju) i polje koje je prazno.
9. **Specijalna šestica**Ukoliko igrač baci broj 6, sistem mora da primeni specijalno pravilo igre. Očekivani rezultat je da igrač dobije dodatni potez. Ako mu se piun nalazi na startu, on se izmešta na početno polje, a ako je piun već u igri, igrač dobija priliku za još jedno bacanje. Sistem mora pravilno da obradi sve interakcije sa drugim piunovima i da vodi računa o ograničenjima, kako bi se sprečilo beskonačno izvođenje poteza. Testiranje uključuje različite situacije: bacanje šestice dok je piun na startu, bacanje šestice kada je piun već u igri i bacanje šestice dok je piun blizu cilja, uz proveru da li dodatni potez pravilno funkcioniše.
10. **Nepravilna kocka**Ako sistem dobije ulaz koji nije broj između 1 i 6 (negativan broj ili string), on ignoriše takav unos i prikazuje poruku o grešci. Sprečava narušavanje toka igre i osigurava da kocka uvek generiše validne rezultate.

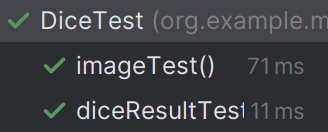
# Testovi

Testiranje aplikacije uključuje dve glavne klase testova:

1. **DiceTest(slika 25.)**: Testira funkcionalnost kockice.
   * diceResultTest(): Proverava da li rezultat bacanja kockice spada u opseg od 1 do 6.
   * imageTest(): Proverava da li putanja do slike završava ekstenzijom ".png".
2. **DatabaseTest(slika 26.):** Testira funkcionalnost baze podataka.
   * testSaveWinner(): Proverava da li se pobednik pravilno čuva u bazi podataka.
   * testSaveWinnerIncrementsWins(): Proverava da li se broj pobeda pravilno inkrementira.
   * testGetWinCounts(): Proverava da li metoda getWinCounts() vraća validne podatke.
   * testGetStepCounts(): Proverava da li metoda getStepCounts() vraća validne podatke.
   * testGetLastGameId(): Proverava da li metoda getLastGameId() vraća validan ID poslednje igre.

Rezulatati testova:

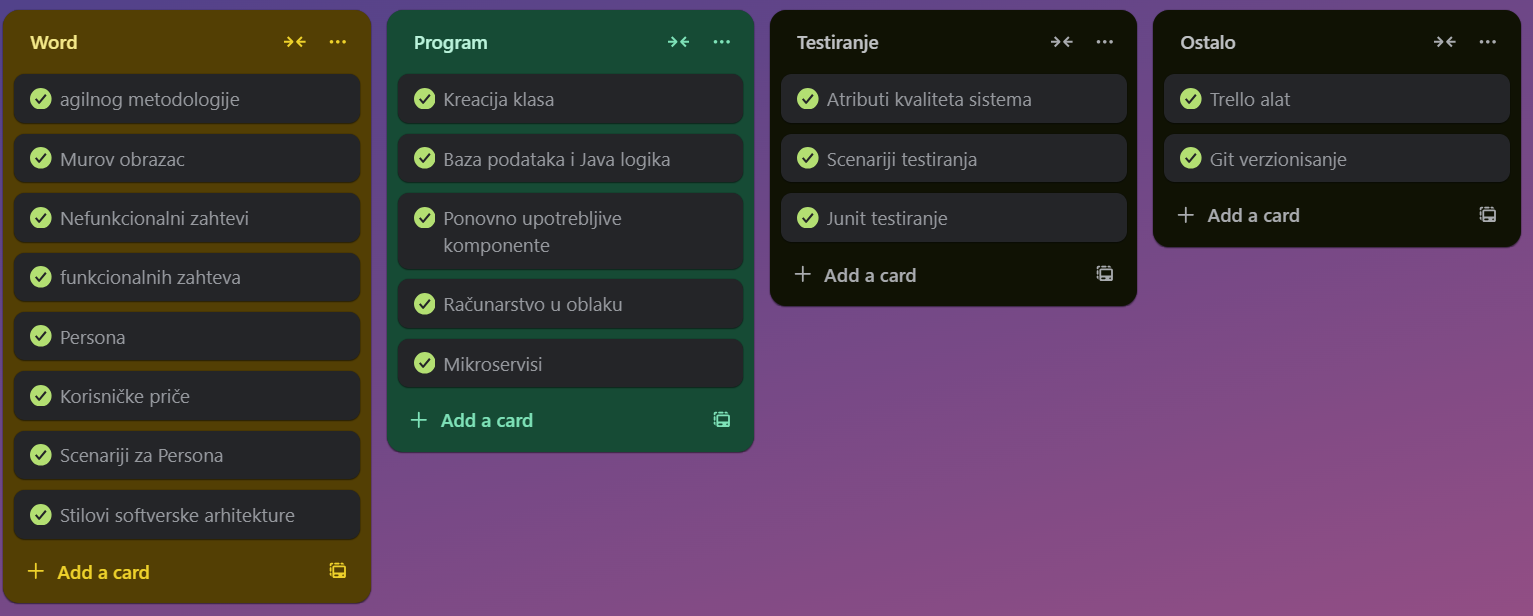
* Svi testovi su uspešno prošli bez grešaka.
* Pokrivenost testovima uključuje osnovne funkcionalnosti kockice i baze podataka.

Slika 25. I Slika 26.

# ****Trello alat****

Trello(Slika 27.) je vizuelni alat za organizaciju zadataka i projekata. Osnovna ideja je da omogući jednostavno i pregledno praćenje toka rada kroz liste i kartice. Svaka lista predstavlja određenu fazu ili kategoriju posla, dok se kartice koriste za pojedinačne zadatke, koji se mogu premestiti iz jedne liste u drugu, u zavisnosti od napretka.



Slika 27. Trello

Lista **Word** sadrži zadatke vezane za dokumentaciju i teorijski deo. Tu spadaju teme poput agilnih metodologija, Murovog obrasca, nefunkcionalnih I funkcionalnih zahteva, persona i korisničkih priča. Takođe su obuhvaćeni scenariji za personu i stilovi softverske arhitekture. Lista služi za pripremu i pisanje sadržaja u Word dokumentima.

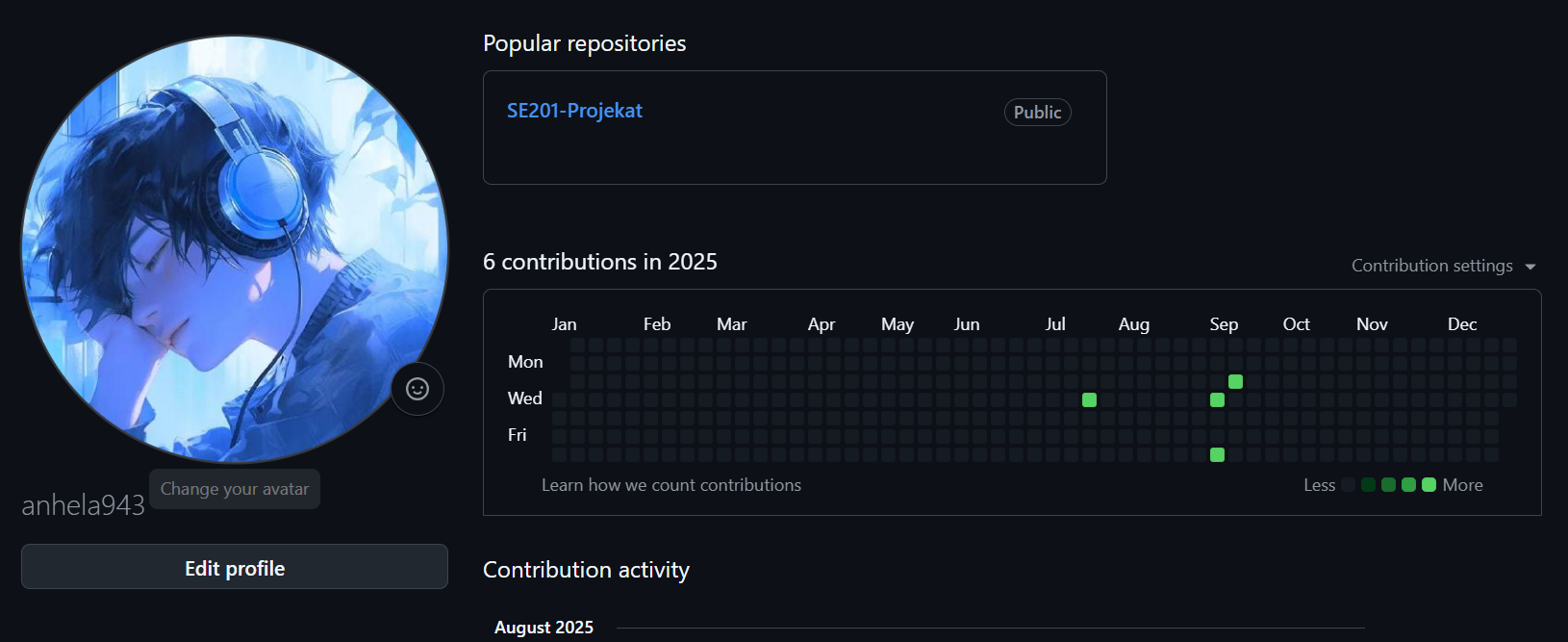
Lista **Program** gde se nalaze zadaci vezani za tehničku implementaciju i programerski rad. Tu je kreacija klasa, povezivanje baze podataka i Java logike, pravljenje ponovo upotrebljivih komponenti, kao i teme kao što su računarstvo u oblaku i mikroservisi. Pored toga, obrađeni su i sigurnosni aspekti. Pokriva praktičan deo razvoja.

Lista **Testiranje** je provera kvaliteta sistema i pouzdano programiranje. U njoj se nalaze zadaci kao što su pisanje scenarija testiranja i korišćenje JUnit-a za testiranje. Cilj liste je da se osigura ispravnost i kvalitet koda nakon implementacije.

Lista **Ostalo**, u njoj su zadaci koji nisu direktno povezani sa razvojem ili dokumentacijom, ali su potrebni za projekat. Tu su dva važna alata: Trello za organizaciju i Git za verzionisanje koda. Oni predstavljaju podršku za efikasan rad tima i praćenje verzija projekta.

# Git ****verzionisanje****

Git(Slika 28.) je distribuirani sistem za kontrolu verzija, koji omogućava efikasno praćenje i upravljanje promenama u kodu tokom razvoja softverskih projekata. Osnovna ideja je da svaki član tima ima lokalnu kopiju celog repozitorijuma, uključujući i njegovu istoriju. Tako programeri mogu raditi nezavisno jedni od drugih, a promene se kasnije objedinjuju kroz proces commit-ovanja i spajanja (merge).



Slika 28. Git

# Zaključak

Scrum metodologija je korišćena kako bi se razvoj projekta podelio na sprintove i tako obezbedilo da se svaka funkcionalnost postepeno implementira i testira. Persone su pomogle da se bolje razumeju potrebe različitih tipova korisnika i da se igra prilagodi njihovim očekivanjima i načinu korišćenja. Kroz implementaciju koda u JavaFX-u ostvarene su ključne funkcionalnosti igre – bacanje kocke, pomeranje figura, pravila za izbacivanje i povezivanje više igrača preko mreže.

# Reference

1. <http://lams.metropolitan.ac.rs:8080/lams>, LAMS