

**ZAVRŠNI RAD**

**Python server kao podrška Unity klijentu**

**Predmet: Web programiranje**

**Mentor:** **Student:**

**dr Slavimir Stošović** **Filip Krčmarević**

**REr 22/19**

**Jun 2022.**

**ZAVRŠNI RAD**

**Python server kao podrška Unity klijentu**

**Predmet: Web programiranje**

**Mentor:** **Student:**

**dr Slavimir Stošović** **Filip Krčmarević**

**REr 22/19**

**Članovi komisije:**

1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# ZAHVALNICA

Zahvaljujem se svim profesorima i asistentima Akademije tehničko vaspitačkih strukovnih studija – Odsek Niš koji su mi predavali, i koje sam imao prilike da upoznam tokom trajanja studija.

Posebnu zahvalnost dugujem svom mentoru, profesoru Slavimiru Stošoviću za sve dragocene savete, izdvojeno vreme i pomoć prilikom izrade završnog rada.

**SADRŽAJ**

[ZAHVALNICA - 3 -](#_Toc104905904)

[1. UVOD - 5 -](#_Toc104905905)

[1.1 Klijent-server arhitektura - 3 -](#_Toc104905906)

[1.1.1 Dvoslojna arhitektura - 3 -](#_Toc104905907)

[1.1.2 Troslojna arhitektura - 5 -](#_Toc104905908)

[1.1.3 Višeslojna arhitektura - 5 -](#_Toc104905909)

[1.2 Gejming industrija - 3 -](#_Toc104905910)

[1.2.1 Slojevi u lancu vrednost - 3 -](#_Toc104905911)

[1.2.2 Gejming industrija u 2020-tim - 3 -](#_Toc104905912)

[1.2.3 Multiplayer igre - 3 -](#_Toc104905913)

[1.2.4 Multiplatform igre - 3 -](#_Toc104905914)

[1.2.5 Serveri - 4 -](#_Toc104905915)

[2. Komuniciranje preko mreže - 5 -](#_Toc104905916)

[2.1 Web tehnologije - 5 -](#_Toc104905917)

[2.2 Soketi - 5 -](#_Toc104905918)

[3. Tehnologije korišćene na serverskoj strani - 6 -](#_Toc104905919)

[3.1 Python programski jezik - 6 -](#_Toc104905920)

[3.2 MySQL - 6 -](#_Toc104905921)

[3.3 XAMPP - 6 -](#_Toc104905922)

[4. Tehnologije korišćene na klijentskoj strani - 7 -](#_Toc104905923)

[4.1 C# - 7 -](#_Toc104905924)

[4.2 Unity - 7 -](#_Toc104905925)

[5. Realizacija projekta - 8 -](#_Toc104905926)

[5.1 Ostvarivanje komunikacije izmedju servera i klijenata - 8 -](#_Toc104905927)

[5.1.1 Kreiranje TCP servera - 8 -](#_Toc104905928)

[5.1.2 Kreiranje TCP klijenta - 9 -](#_Toc104905929)

[5.1.3 Komade - 10 -](#_Toc104905930)

[5.1.4 Korisnički profil - 12 -](#_Toc104905931)

[5.1.5 Kreiranje soba - 16 -](#_Toc104905932)

[5.1.6 Sistem prijateljstva - 18 -](#_Toc104905933)

[5.1.7 Slanje emajlova - 21 -](#_Toc104905934)

[5.1.7 GameLoop - 22 -](#_Toc104905935)

[6. Zaključak - 29 -](#_Toc104905936)

[7. Literatura - 30 -](#_Toc104905937)

# UVOD

Projekat je rađen u python programskom jeziku na serverskoj strani koja je ujedno i glavni fokus ovog rada, korisnička strana je rađena u Unity game engine korišćenjem C# programskog jezika, server je lokalni uz podršku XAMPP-a, za čuvanje podataka koristi se lokalna MySql baza podataka, server i klijent komuniciraju preko soketa. Cilj projekta jeste da se pokaže da iako python nije jedan od popularniji server side jezika može da posluži i lako konfiguriše za neke nezahtevnije projekte. Unity klijent će služiti samo za vizualno prikazivanje i kako bi dao namenu serveru, rad neće zalaziti u dubinu objašnjavanja klijentske strane.

## 1.1 Klijent-server arhitektura

Za distribuirane aplikacije koriste se različite arhitekture hardvera i softvera, od kojih je najrasprostranjenija klijent-server arhitektura. Ona podrazumeva deljenje zadataka između onih koji obezbeđuju resurse ili usluge tj. servere i onih koji traže usluge tj. klijente.

U klasičnim klijent-server modelima mogu se uočiti tri klase komponenti:

* **Server**
* **Klijent**
* **Mreža**

Računarska mreža i komunikacioni softver omogućavaju prenos podataka između klijenta i servera. U ovom modelu klijent je aktivan korisnik koji ne deli svoje resurse ali zahteva serverov sadržaj ili uslugu i on započinje sesiju komunikacije sa serverom koji je pasivni učesnik u komunikaciji, čeka na zahteve, izvršava ih i odgovara. Serveri su obično veoma jaki računari s dobrim karakteristikama zato što moraju istovremeno obraditi mnogo klijentskih zahteva.

Klijent i server mogu biti smešteni u istom ili u različitim računarima.

Osnovna podela klijent-server arhitekture

* **Dvoslojna**
* **Troslojna**
* **Višeslojna**

### 1.1.1 Dvoslojna arhitektura

Dvoslojna arhitektura sastoji se od 3 komponente distribuirane u dva sloja, u klijentskom i serverskom.

* Korisnički interfejs (korisnički sloj) - sesije, unos teksta, prikaz na ekranu
* Upravljanje procesima (delom korisnički a delom serverski sloj) - generisanje, izvođenje i nadgledanje procesa i neophodnih resursa
* Upravljanje podacima (serverski skoj) - servisi vezani za deljenje podataka i datoteka

Logika aplikacije je podeljena između klijenta i servera tako da obezbedi optimalno korišćenje resursa.

Prezentacija podataka i provera ulaznih podataka su sastavni deo klijent-aplikacije dok rukovanje podacima, u smislu njihovog fizičkog smeštaja i kontrole pristupa vrši se na serveru.

Neke od prednosti ovakvog modela obrade podataka su:

* Centralizovano upravljanje resursima sistema
* Jednostavnije obrađivanje sigurnosti podataka
* Osnovni problem je nedostatak skalabilnosti

### 1.1.2 Troslojna arhitektura

Troslojna arhitektura predstavlja sisteme sa tri, skoro u potpunosti nezavisna podsistema:

* Podsistem za interakciju sa korisnikom
* Podsistem za implementaciju osnovnih funkcija sistema
* Podsistem za rukovanje podacima

Neke od prednosti ovakvog modela su:

* jednostavan za održavanje
* skalabilnost
* pouzdanost i fleksibilnost

### 1.1.3 Višeslojna arhitektura

Višeslojna arhitektura se postiže daljim proširivanjem koncepta troslojnih sistema, gde se vrši dalja podela na komponente u okviru srednjeg sloja sa ciljem još većeg povećanja skalabilnosti, odnosno performansi.

Prednost ovakve arhitekture je ta da se obrada podataka širi na razne servere čime se izjednačava opterećenje među pojedinim serverima i povećava skalabilnost sistema

Nedostaci: veće opterećenje mreže i znatno je teža za razvoj i testiranje aplikacija koje rade u ovoj arhitekturi.

## 1.2 Gejming industrija

Industrija video igara je industrija uključena u razvoj, marketing i monetizaciju video igara.

Moderni lični računari duguju mnoge napretke i inovacije industriji igara: zvučne kartice, grafičke kartice i 3D grafički akceleratori, brži CPU.

### 1.2.1 Slojevi u lancu vrednost

* **Razvoj igrara** - predstavljaju programeri, dizajneri, umetnici I njihovo rukovodstvo, uz podršku razvojnih alata.
* **Izdavaštvo** - Ili izdavač, ubično uključuje izvor finansiranja razvoja video igre kao I pružanje marketinga I reklamiranja za proizvod.
* **Distribucija** - Obično uključuje proizvodnju I umnožavanje medija za igre I pakovanje za maloprodaje igre. A može biti izvršena preko: maloprodaja ili digitalnih
* **Prodavac**
* **Kupci**
* **Platforma** - Predstavlja platformu/hardver za koji je igrica razvijana

### 1.2.2 Gejming industrija u 2020-tim

COVID-19 pandemija je doprinela tome da je gejming industrija imala porast od 20% u prihodima u odnosu na 2019. godinu dostignuvši $179 milijarde globalnih prihoda. Ljudi su nedostatak socijalnog kontakta rešavali raznim multiplayer igricama od kojih su najpopularnije bile: Animal Crossing: New Horizons, Fall Guys i Among Us.

Pandemija je trajala duže nego što su stručnjaci mislili pa je industrija u 2021 bila pod uticajem globalnog nedostatka poluprovodničkih čipova koji su se koristili za proizvodnju hardvera. Što je uticalo i na najveće proizvođače konzola, što je doprinelo do toga da su neke novije verzije hardvera ovih kompanija nedostupne većini ljudi zbog njihove cene. Ovaj period je pogodio i korisnike personalnih računara, kripto majneri su uvideli da im je isplativije da koriste grafičke kartice što je istima jako podiglo cenu.

### 1.2.3 Multiplayer igre

**Multiplayer** igrice su igrice u kojima više od jednog igrača može da bude u istom okruženju u istom vremenu, bilo da se radi o:

* lokalnom sistemu (na jednom, tj. na istom računaru),
* lokalnom sistemu na različitim računarima (koji su umreženi),
* globalnom mrežnom sistemu (obično je to internet)

Ciljevi multiplayer igrica mogu biti različiti od sarađivanja sa drugim igračima kako bi se igrica pobedila do masovnih borbi izmedju samih igrača bez ikakvog prisustva NPC-jeva. Razvoj multiplayer igrica je doveo do raznih kompetitivnih igrica u kojima se igrači takmiče međusobno kako bi osvojili pozamašnu sumu novca, neke od najpoznatijih ovakvih igrica su: LoL,Dota,Fifa.

### 1.2.4 Multiplatform igre

**Multiplatformske** igrice su igrice čije je izvršavanje podržano na više od jednog hardverskog sistema i često se vezuju za multiplayer igrice. Multiplatformske igrice su zahtevnije za programere i zahtevaju veće znanje kako bi se realizovale, programeri često žrtvuju performance igrice zarad njene brže izrade tako što koriste game engine koji su prilagođeni za pravljenje igrica za različite platforme kao što su:

* **Unity**: Windows, Mac, Universal Windows platform (UWP),Linux Standalone, iOS,Android,PS5,PS4,Xbox One, Nitendo Switch, Google Stadia,WebGL…
* **Unreal**: Windows, macOS, and Linux. PlayStation 5, PlayStation 4, Xbox Series X, Xbox Series S, Xbox One, Nintendo Switch, i Google Stadia
* **Godot**:  Linux, macOS, Windows, HTML5, WebAssembly, Android,iOS…

Jedna od popularnijih multiplatforskih igrica jeste Fortnite koja podržava: Nitendo Switch, PlayStation 4, Xbox One, Xbox S,iOs,Android,macOs,Windows I PlayStation 5

### 1.2.5 Serveri

Serveri su autoritivni (poverljivi) izvori događaja u multiplayer igricama. Server šalje dovoljno podataka svojim klijentima kako bi oni mogli da održavaju svoju tačnu verziju sveta igre. Serveri se mogu podeliti na:

* **Namenski serveri** - Namenski serveri simuliraju svetove igara bez podrške direktnog unosa ili izlaza, osim onoga što je potrebno za njihovu administraciju. Igrači moraju da se povežu na server sa zasebnim klijentskim programima da bi videli igru i komunicirali sa njom
* **Listen server** - Serveri za slušanje rade u istom procesu kao klijent igre. Oni inače funkcionišu kao namenski serveri, ali obično imaju nedostatak što moraju da komuniciraju sa udaljenim igračima preko rezidencijalne internet konekcije igrača za hostovanje.
* **Host migration**– Isti kao listen server samo sa dodatnom funkcionalnošću koja omogućava prebacivanje hosta sa jednog igrača na drugog ako prvobitni host izubi vezu a sa ciljem da ne dođe do prekida igre.
* **Peer-to-Peer** – Arhitektura u kojoj nemamo glavni server već igrači međusobno jedni drugima šalju podatke i sami određuju njihovu validnost.

# 2. Komuniciranje preko mreže

## 2.1 Web tehnologije

Web tehnologije se odnose na različite alate i tehnike koje se koriste u procesu komunikacije između različitih tipova uređaja preko mreža (pr. internet).

Web tehnologije se mogu podeliti na sledeće sekcije:

* **World Wide Web(WWW)** – bazirana je na više različitih tehnologija(Web pretraživači, ,Hypertext Markup Language(HTML) I HTTP)
* **Web Browser** – softver koji je koristi za pretraživanje www-a, pruža interfejs izmedju servera i klijenta i zahteva
* **Web Server** – je program koji obrađuje korisničke zahteve i šalje im fajlove, fajlovi grade web stranicu. Ova razmena se realizuje upotrebom HTTP.
* **Web Pages** – digitalni document koji je povezan na WWW i može mu pristupiti svako ko ima konekciju sa mrežom i neki web pretraživač
* **Web Development** – predstavlja kreiranje,razvoj i održavanje web sajtova. Može biti podeljen na dva dela:
  + **Frontend Development** – Deo web sajta ili aplikacije sa kojim korisnik ima direktnu interakciju, takodje je poznat i kao klijentska strana aplikacije. Neki od frontend jezika: CSS, JavaScript
  + **Backend Development** – Serverski deo web sajta ili aplikacije. Deo koji korisnik ne može da vidi i sa kojim on ne može da ima interakciju a uglavnom se koristi za skladištenje i uređivanje podataka. Neki od backend jezika: PHP,Python,C#

## 2.2 Soketi

Soketi omogučavaju komunikaciju između dva različita procesa na istoj ili na nekoj drugoj mašini. Soketi su se prvi put pojavili u 2.1BSD a zatim prerađeni u svoj trenutni oblik sa 4.2BSD. Soketi se koriste u klijent-server aplikacionim framework-ovima. Soketi se mogu podeliti na:

* **Stream** – Dostava u mrežnom sistemu je zagarantovana i redosled primljenih podataka će biti isti kao redosled poslath podataka. Ovakvi soketi koriste TCP (Transmission Control Protocol) za razmenu podataka. Ako razmena podataka nije moguća, pošiljalac dobija poruku o grešci.
* **Datagram** - Dostava podataka nije zagarantovana, koriste UDP (User Dataram Protocol)
* **Raw** – Omogućavaju korisnicima pristup osnovnim komunikacionim protokolima sa mogućnošću soketske abstrakcije.
* **Sequenced Packet Sockets –** Slični su steam soketima, sa izuzetkom da su sačuvane granice zapisa.

# 3. Tehnologije korišćene na serverskoj strani

## 3.1 Python programski jezik

Python je programski jezik visokog nivoa opšte namene podržava objektno-orjentisani,imperativni i funkcionalni still programiranja. Sintaksa jezika omogućava pisanje veoma preglednih programa. Jezik se brzo i lako uči što je i jedan od glavnih razloga što je u poslednjim godinama dobio znatno na popularnosti. Programi pisani u Pythonu se najčešće interpretiraju. Python se najčešće koristi za razvijanje web sajtova, automatizaciju, analiziranje I vizualizaciju podataka, a autor ovog jezika je Guido van Rossum.

Neki od najpopularnijih python modula su: TensorFlow, Scikit-Learn, Numpy, Keras, PyTorch, LightGBM, Eli5, SciPy, Theano, Pandas

## 3.2 MySQL

MySQL je višenitni višekorisnički sistem za upravljanje bazama podataka, radi kao server i obezbedjuje višekorisnički interfejs za pristup bazi podataka. MySQL je jezik visokog nivoa I koristi se za upravljanje relacionim bazama podataka. Relaciona baza podataka organizuje podatke u jednu ili više tabela u kojima tipovi podataka mogu biti međusobno povezani, ovi odnosi pomažu u strukturiranju podataka. SQL je jezik koji se koristi za kontrolu pristupa bazi podataka kao i za kreiranje, modifikovanje i izdvajanje podataka. MySQL je besplatan softver ali je dostupan i pod raznim vlasničkim licencama.

## 3.3 XAMPP

XAMPP je besplatno rešenje za višeplatformi sa javnim kodom koji se uglavnom sastoji od Apache HTTP servera, MariaDB baze podataka i interpretatora za skripte napisane u programskim jezicima PHP i Perl. XAMPP se uglavnom koristi za ostvarivanje lokalnog servera tj. za test servere, s obzirom na to da večina realnih solucija za web servere koriste iste komponente kao XAMPP prelaz sa lokalnog test server na globalni server je moguća i u većini slučajeva ne toliko komplikovana.

# 4. Tehnologije korišćene na klijentskoj strani

## 4.1 C#

C# je jednostavan, moderan, objektno orjentisan programski jezik opšte namene kojeg je razvio Microsoft u okviru svoje .NET tehnologije. Pored objektno orijentisane paradigme podržava i druge paradigme poput: funkcionalne, komponentne, generičke…

Jedna od osnovnih prednosti C# jeste njegova portabilnost a podržava pisanje programa za ugrađene sisteme. Iako je C# namenjen da bude lak za memoriju i procesor nikad nije bio namenjen da se takmiči sa C i asembler programskim jezicima.

## 4.2 Unity

Unity je multi-platformsko okruženje za razvijanje video igrara. Iako je u početku bio namenjen za Mac operativne sisteme, danas unity podržava različite platforme za desktop, mobilne, konzole i virtualnu realnost, a posebno je popularan za razvoj mobilnih igara za iOS I Android a neki od najpopularnijh naslova rađeni u ovom okruženju za mobilne platforme jesu: Pokémon Go, Monument Valley, Call of Duty: Mobile, Beat Saber i Cuphead.

Smatra se da je jednostavan za korišćenje a može se koristiti za razvijanje 2D i 3D igara, a svoj rad bazira na sistemu komponenti. Iako je namenjen za razvoj video igrara, razne industrije kao što su filmska, industrijska, arhitektura... Unity je pisan u C++ programskom jeziku a podržava skripte pisane u C# i JavaScript programskim jezicima. Unity je besplatan softver za sve, sve dok godišnja primanja korisnika ne prođu granicu od $100 000, nakon čega Unity više nije besplatan.

# 5. Realizacija projekta

## 5.1 Ostvarivanje komunikacije izmedju servera i klijenata

Komunikacija između servera i klijenata ostvaruje se pomoću TCP protokola, na portu 5000.

Komunikacija se vrši tako što server i klijenti razmenjuju poruke (komande) a na osnovu sadžaja poruke znaju šta treba da se desi. Sva komunikacija ide preko servera, nema komunikacije međusobno između klijenata.

### 5.1.1 Kreiranje TCP servera

Serverski deo projekta kreira TCP server sledećim blokom koda (u python programskom jeziku):



### 5.1.2 Kreiranje TCP klijenta

Klijentski deo projekta kreira TCP klijenta sledećim blokom koda (u C# programskom jeziku):



### 5.1.3 Komade

#### 5.1.3.1 Komande koje klijent poziva na serveru



#### 5.1.3.2 Komande koje server poziva na klijentu



#### 5.1.3.3 Komande koje server poziva na serveru



### 5.1.4 Korisnički profil

Nakon uspesnog pokretanja servera, server pokreće par novih niti od kojih je jedan zadužen za slušanje zahteva za prijavu klijenta:



Funkcija newConnection se poziva startovanjem servera i ostaje uključena sve dok je server uključen



#### 5.1.4.1 Prijava korisnika



#### 5.1.4.2 Registracija korisnika



### 5.1.5 Kreiranje soba

Implementirane su dve vrste soba/mečeva: prijateljski i regularni. Kako bi se počeo prijateljski meč potrebno je da barem 2 igrača budu u sobi, a sobi se pristupa preko njenog imena ili preko poziva prijatelja, na kraju meča nema nagrada za učesnike kako bi se sprečila zloupotreba prijateljskog sistema meča ali se on računa za leaderboard i meč može da pokrene bilo ko iz sobe. Što se tiče regularnih mečeva, da bi se pokrenuli neophodno je 5 igrača i samo server može da pokrene igru kada se soba napuni, na kraju igre svi preživeli igrači dobijaju odgovarajući broj bodova i novčića dok svi igrači koji su izgubili gube bodove ukoliko imaju više od 10 bodova i doijaju malu sumu novčića. Kreiranje soba postiže se sledećim kodom



### 5.1.6 Sistem prijateljstva

#### 5.1.6.1 Pretraživanje



#### 5.1.6.2 Dodavanje prijatelja



#### 5.1.6.3 Provera da li je prijatelj trenutno aktivan



#### 5.1.6.4 Pozivanje prijatelja u sobu



### 5.1.7 Slanje emajlova



### 5.1.7 GameLoop

#### 5.1.7.1 Započinjanje igre



#### 5.1.7.2 Odgovaranje na pitanje



#### 5.1.7.3 Tok igre

##### 5.1.7.3.1 Slanje poruke svima u sobi



##### 5.1.7.3.2 Dodavanje kutija



##### 5.1.7.3.3 Trajanje runde



##### 5.1.7.3.4 Provera odgovora



##### 5.1.7.3.5 Uklanjanje kutija na kraju runde



5.1.7.6 Provera da li je neki igrač izgubio



##### 5.1.7.7 Nagrađivanje igrača



##### 5.1.7.8 Početak igre



##### 5.1.7.9 Igra



##### 5.1.7.10 Završnica igre



# 6. Zaključak

# 7. Literatura

[1] Model klijent—server, Wikipedia: <https://sr.wikipedia.org/wiki/Model_klijent—server>

[2] Klijent server tehnologija, prezentacija profesora Mirka Kosanovića: <https://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/1521843285KS%20Predavanje%203%202016.pdf>

[3] ,,Video game industry”, Wikipedia: <https://en.wikipedia.org/wiki/Video_game_industry>

[4] ,,Game server”, Wikipedia: <https://en.wikipedia.org/wiki/Game_server>

[5] ,, What is a Socket?” : <https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm>

[6] Python (programming language): <https://en.wikipedia.org/wiki/Python_(programming_language)>

[7] ,, MySQL”, Wikipedia: <https://en.wikipedia.org/wiki/MySQL>

[8] ,, XAMPP”, Wikipedia: <https://en.wikipedia.org/wiki/XAMPP>

[9] ,, Unity (game engine)”, Wikipedia: <https://en.wikipedia.org/wiki/Unity_(game_engine)>