

**ZAVRŠNI RAD**

***Python* web servis kao podrška *Unity* klijentskoj aplikaciji**

**Predmet: Web programiranje**

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

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

**profesor strukovnih studija REr 22/19**

**Avgust 2022.**

**ZAVRŠNI RAD**

**Python web servis kao podrška Unity klijentskoj aplikaciji**

**Predmet: Web programiranje**

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

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

**profesor strukovnih studija REr 22/19**

**Članovi komisije:**

1. dr Mirko Kosanović, profesor strukovnih studija

2. dr Dušan Stefanović, profesor strukovnih studija

# ZAHVALNICA

**SADRŽAJ**

[ZAHVALNICA - 3 -](#_Toc109844401)

[1. UVOD - 5 -](#_Toc109844402)

[1.1 Klijent-server arhitektura - 5 -](#_Toc109844403)

[1.1.1 Dvoslojna arhitektura - 6 -](#_Toc109844404)

[1.1.2 Troslojna arhitektura - 7 -](#_Toc109844405)

[1.1.3 Višeslojna arhitektura - 7 -](#_Toc109844406)

[1.2 Gejming industrija - 9 -](#_Toc109844407)

[1.2.1 Slojevi u lancu kreiranja video igara - 9 -](#_Toc109844408)

[1.2.2 Industrija igrica u 2020-tim - 9 -](#_Toc109844409)

[1.2.3 *Multiplayer* igre - 10 -](#_Toc109844410)

[1.2.4 Multiplatform igre - 11 -](#_Toc109844411)

[1.2.5 Serveri - 12 -](#_Toc109844412)

[2. Komuniciranje preko mreže - 13 -](#_Toc109844413)

[2.1 Web tehnologije - 13 -](#_Toc109844414)

[2.2 Soketi - 13 -](#_Toc109844415)

[3. Tehnologije korišćene na serverskoj strani - 14 -](#_Toc109844416)

[3.1 Python programski jezik - 14 -](#_Toc109844417)

[3.2 *MySQL* - 15 -](#_Toc109844418)

[3.3 *XAMPP* - 16 -](#_Toc109844419)

[4. Tehnologije korišćene na klijentskoj strani - 17 -](#_Toc109844420)

[4.1 *C#* - 17 -](#_Toc109844421)

[4.2 *Unity* - 18 -](#_Toc109844422)

[5. Realizacija projekta - 21 -](#_Toc109844423)

[5.1 Projektni zadatak - 21 -](#_Toc109844424)

[5.2 Ostvarivanje komunikacije između servera i klijenata - 21 -](#_Toc109844425)

[5.2.1 Kreiranje TCP servera - 21 -](#_Toc109844426)

[5.2.2 Kreiranje TCP klijenta - 22 -](#_Toc109844427)

[5.2.3 Komade - 23 -](#_Toc109844428)

[5.2.4 Korisnički profil - 25 -](#_Toc109844429)

[5.2.5 Kreiranje soba - 29 -](#_Toc109844430)

[5.2.6 Sistem prijateljstva - 31 -](#_Toc109844431)

[5.2.7 Slanje emajlova - 34 -](#_Toc109844432)

[5.2.7 GameLoop - 35 -](#_Toc109844433)

[6. Zaključak - 42 -](#_Toc109844434)

[7. Literatura - 43 -](#_Toc109844435)

# UVOD

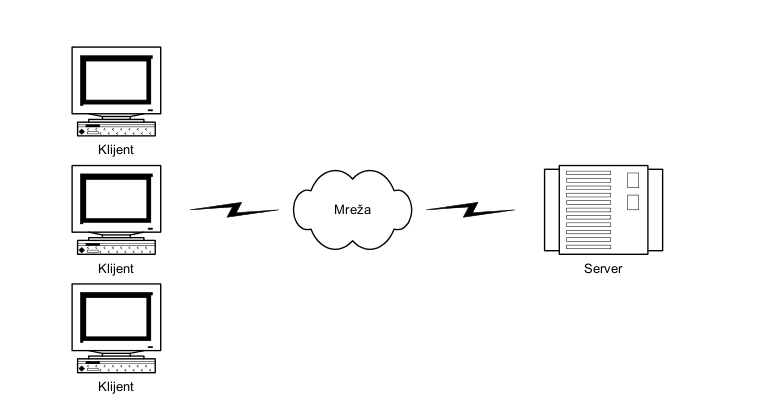
Projekat je rađen u *python* programskom jeziku na serverskoj strani koja je ujedno i glavni fokus ovog rada, korisnička strana je rađena u *Unity Game Engine*-u korišćenjem C# programskog jezika, server je lokalni uz podršku XAMPP-a, za čuvanje podataka koristi se lokalna *MySql* baza podataka, server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



Sl. 3.1. Primer klijent server sistema

Na slici 3.1 prikazana je osnovna klijent-server struktura u kojoj više klijenata komunicira preko mreže sa udaljenim serverom. Klijenti mogu da budu različiti uređaji poput: telefona, računara i drugih električnih uređaja koji mogu da prime, skladište i šalju informacije, najpopularniji izbor mreže jeste internet, dok se za server obično bira jak računar.

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.

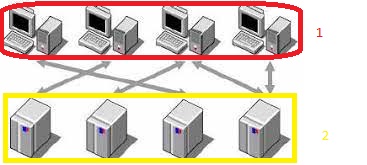
Osnovna podela klijent-server arhitekture

* **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 sloj) - servisi vezani za deljenje podataka i datoteka



Sl. 1.1.1.1 Dvoslojna arhitektura

Na slici 1.1.1.1 je prikazana dvoslojna arhitektura u kojoj su računari u crvenoj grupi (grupi 1) zaduženi za korisnički interfejs kao i za deo upravljanjima procesima dok su računari u žutoj grupi (grupi 2) zaduženi za upravljanje podacima kao i za deo upravljanjem procesima.

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.

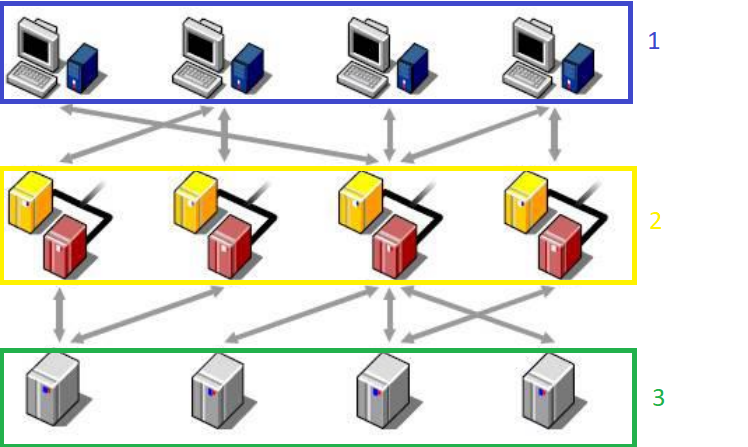
Neke od prednosti ovakvog modela obrade podataka su:

* 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



Sl. 1.1.2.1 Troslojna arhitektura

Na slici 1.1.2.1 prikazana je šema troslojne arhitekture u kojoj plava grupa predstavlja korisnički sloj, tj. sloj zadužen za korisnički interfejs, žuta grupa predstavlja srednji sloj, tj. sloj zadužen za upravljanje procesima, i poslednja zelena grupa predstavlja sloj podataka tj. sloj zadužen za upravljanje podacima.

Neke od prednosti ovakvog modela su:

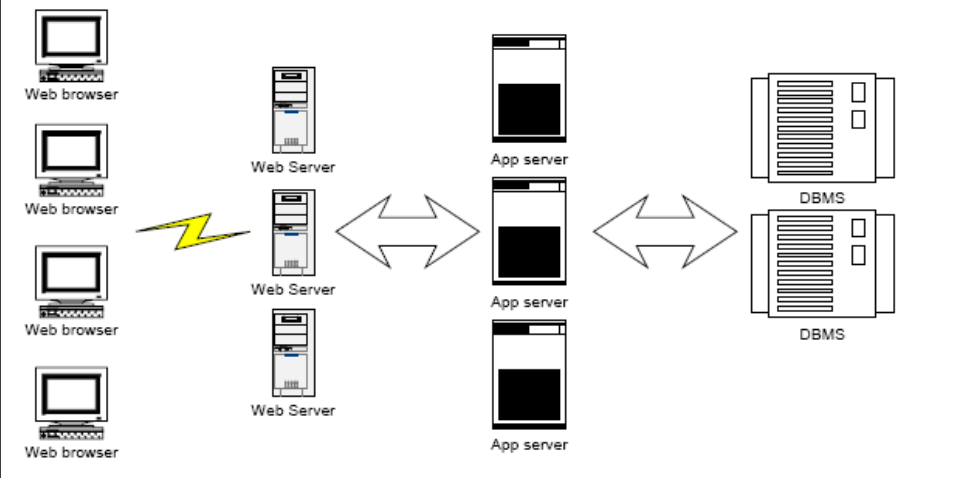
* jednostavan za održavanje
* skalabilnost
* pouzdanost i fleksibilnost

### 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.



Ilustracija 1Sl. 1.1.3.1 Višeslojna arhitektura

Na slici 1.1.3.1 prikazana je višeslojna arhitektura tj. arhitektura jednog ovakvog sistema.

## 1.2 Gejming industrija

Industrija video igara je industrija uključena u razvoj, marketing i monetizacije 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 kreiranja video igara

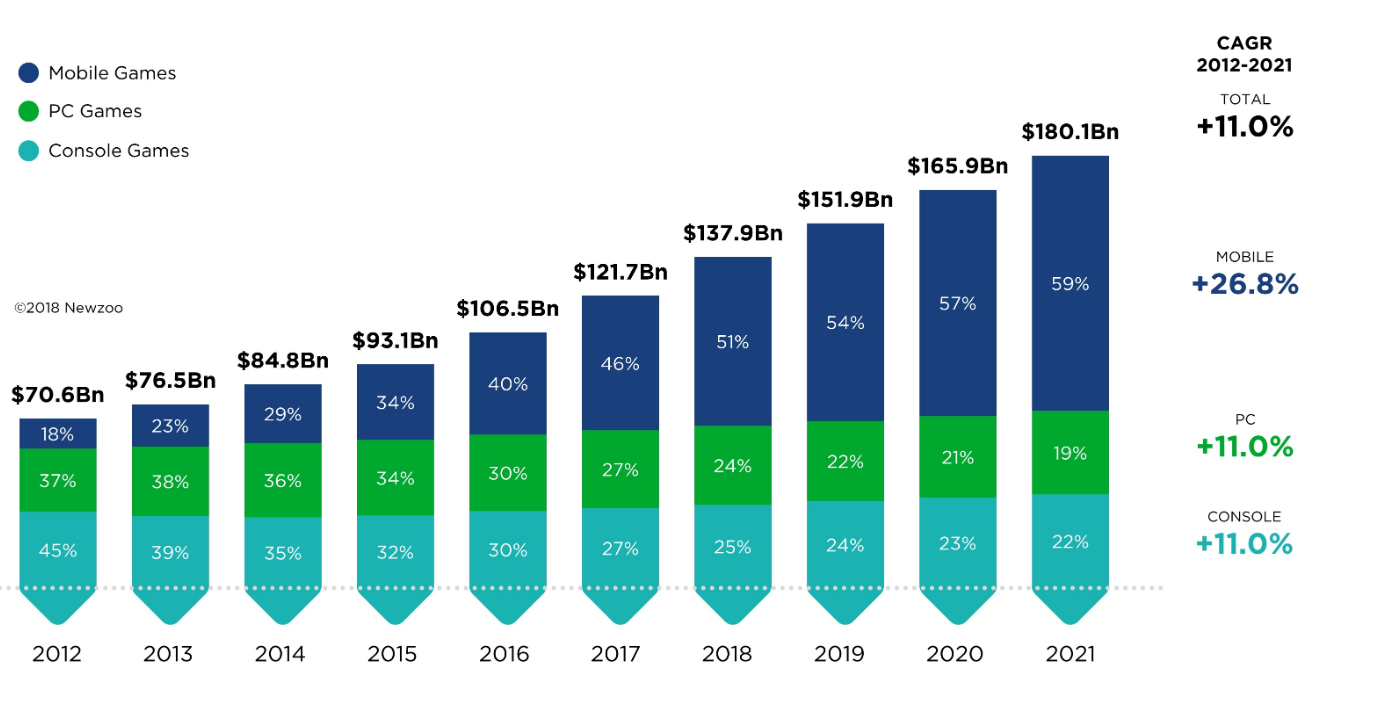
Razvoj igara je proces koji se sastoji od 6 slojeva, svaki od slojeva ima svoje slojeve i podjednako je bitan, osnovni slojevi su:

* **Razvoj igara** - predstavljaju programeri, dizajneri, umetnici I njihovo rukovodstvo, uz podršku razvojnih alata.
* **Izdavaštvo** - Ili izdavač, obično 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 Industrija igrica u 2020-tim

COVID-19 pandemija je doprinela tome da je industrija igrica 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 kopači su uvideli da im je isplativije da koriste grafičke kartice što je istima jako podiglo cenu.



Sl. 1.2.2.1 Prihodi industrije igrica

Na slici 1.2.2.1 prikazan je prihod industrije igara u periodu između 2012 i 2021 godine, kao i procenat doprinosa 3 najzastupljenija sistema za igranje video igrica.

### 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 između 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 date na sledećim slikama:



Sl 1.2.3.1 LoL



Sl 1.2.3.2 Dota



Sl 1.2.3.3 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 engin-*e 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 multiplatformskih 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 autoritativni (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 između 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 dokument 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, takođe 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 omogućavaju 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 poslatih 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 apstrakcije.
* ***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 stil programiranja, a autor ovog jezika je *Guido van Rossum*, holandski programer 80-tih godina 20. veka.

Sintaksa jezika omogućava pisanje veoma preglednih programa. Ono što ga razlikuje od većine drugih programskih jezika jeste to što se kod njega blokovi koda ne ograničavaju tj. ne obeležavaju početak i kraj koda bloka vitičastim zagradama („{“, „}“) već uvlačenjem koda, tj. ukoliko uvučemo liniju koda za jedan tab unutra to znači da je deo koda do izvlačenja linije nazad jedan blok kod.



Sl. 3.1.1 primer koda u python-u

Na slici 3.1.1 može se videti struktura jednog *python* koda.

Ono što ga još izdvaja od većine drugih programskih jezika jeste da nema potrebe za deklaracijom tipova promenjivih već je interpretator dovoljno pametan da sam zaključi o kom tipu podatka se radi, ova funkcionalnost mu je omogućila da bude jedan od osnovnih i prvih programskih jezika koji se uče pri počinjanju sa programiranjem.

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 *Python*-u se najčešće interpretiraju. *Python* se najčešće koristi za razvijanje *web* sajtova, automatizaciju, analiziranje i vizualizaciju podataka,.

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

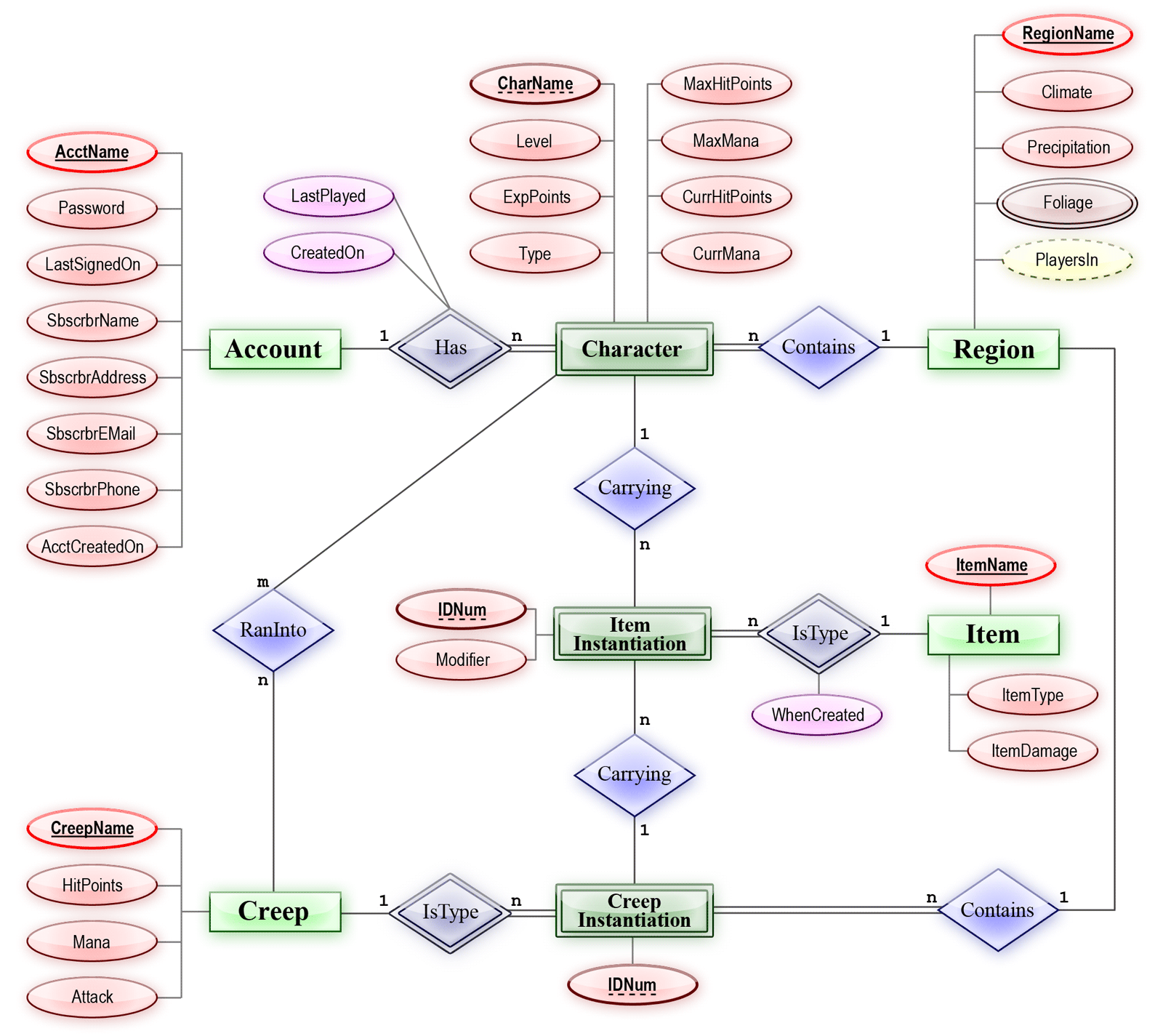
Iako je usvojen u većini škola za početni jezik on je jezik opšte namene i u njemu se mogu razvijati sve vrste aplikacija, ima primene u inženjerskim, naučnim, poslovnim i web aplikacijama, takođe mogu se razvijati prototipovi ali i produkcione aplikacije a u poslednje vreme je jako popularno rešenje za razvijanje AI programa.

## 3.2 *MySQL*

*MySQL* je više nitni višekorisnički sistem za upravljanje bazama podataka, radi kao server i obezbeđuje višekorisnički interfejs za pristup bazi podataka. MySQL je jezik visokog nivoa i koristi se za upravljanje relacionim bazama podataka. Relaciona baza podataka skladišti podatke u tabelama a njihovo upravljanje realizovano je korišćenjem SQL jezika, neke od najčešće korišćenih komandi jesu:

* *INSERT* – koristi se za dodavanje informacija u tabeli,
* *UPDATE* – koristi se za izmenu informacija u tabeli,
* *DELETE* – koristi se za brisanje informacija iz tabela,
* *SELECT* – koristi se za uzimanje informacija iz tabela.

Svaka tabela mora da ima *primary key,* primarni ključ se koristi zauređivanje i sortiranje podataka unutar tabele, takođe služi za identifikovanje svakog reda tabele samim tim primarni ključ unutar jedne tabele ne sme imati ponavljanja. Baza podataka je modelovana korišćenjem odnosa između entiteta što se jasno može videti na slici 3.2.1.



Sl. 3.2.1 Relacioni model

Na slici 3.2.1 se takođe može videti da postoje 2 vrste veza između entiteta:

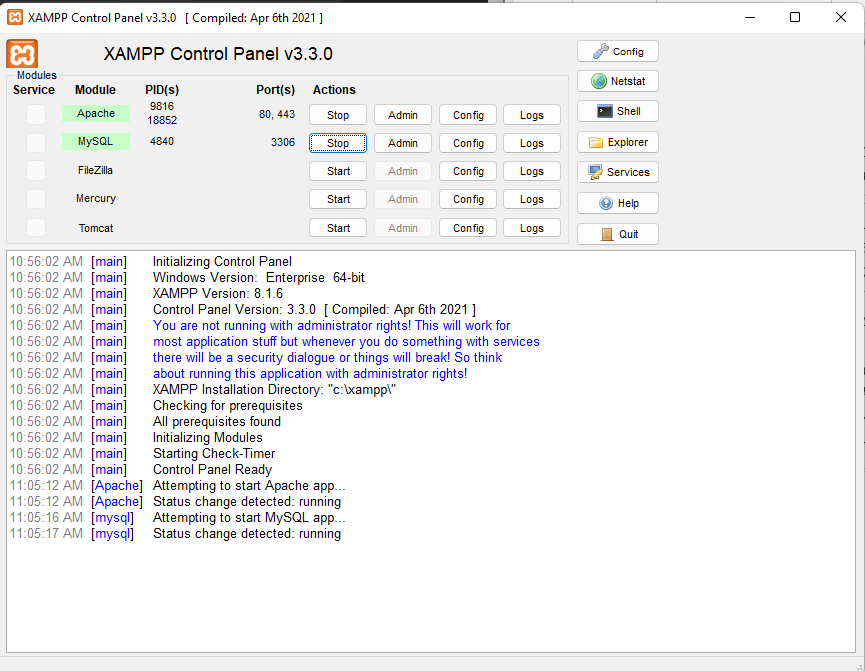
* 1-N – jedan prema više je veza između entiteta u kojoj se strani ključ stavlja u tabeli na kojoj je strani N,
* N-N – više prema više je veza u kojoj se radi povezivanja dve tabele pravi nova tabela kako bi se smanjila redundantnost.

MySQL je besplatan softver ali je dostupan i pod raznim vlasničkim licencama.

## 3.3 *XAMPP*

*XAMPP* je besplatno rešenje za više platformi 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*. Uglavnom koristi za ostvarivanje lokalnog servera tj. za test servere, s obzirom na to da većina 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. Jako je jednostavan za korišćenje uz samo 1 klik moguće je podiči jedan od servisa koje on nudi, a nudi sledeće servise:

* Apache – omogućava *hostovanje* više različitih web sajtova ili drugog kontenta na jednom računaru,
* MySQL – najpopularnije rešenje za relacione baze podataka,
* FileZilla – omogućava *FTP*,
* Mercury – lokalni mail server,
* Tomcat – omogućava upravljanje i kreiranje serverskih aplikacija, i daje informacije o statusu servera.



Sl. 3.3.1 XAMPP Control Panel

Na slici 3.3.1 prikazan je izgled *XAMPP Control Panel*-a na kome su pokrenuti Apache servis na portu 80 i 443 kao i MySQL na portu 3306, pored aktivnih servisa imamo i statusne poruke u izlaznom prozoru aplikacije koje nas obaveštavaju o uspešnosti aktiviranja servisa kao i o nesuglasicama i greškama pri pokretanju ili radu nekog od servisa.

# 4. Tehnologije korišćene na klijentskoj strani

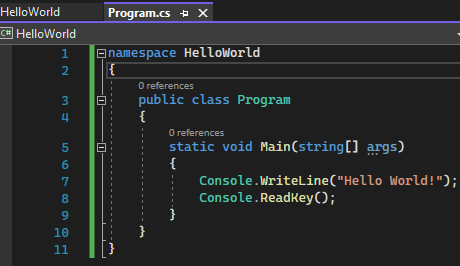
## 4.1 *C#*

C# je jednostavan, moderan, objektno orjentisan programski jezik opšte namene kojeg je razvio *Microsoft* u okviru svoje .*NET* tehnologije, kompailuje se isto kao i ostali .*NET* jezici u *MSIL* što omogućava lakšu integraciju u ostale programske jezike. Razvija ga *Anders Hejlsberg*, on takođe razvija i *Turbo Pascal, Delphi, Visual J++* a pušten je u produkciju 2001-2002 godine. Kada je izašao C# popularno mišljenje je bilo da je to samo klon ili zamena za Javu ali je C# uzeo mnogo veći zamah, iako je na početku imao probleme kao što su preveliko opterećenje memorije, ilegalne reference na pokazivače, previše složeno višestruko nasleđivanje i sl. svi ovi problemi su rešeni u sledećih godinu dana implementiranjem skupljača đubreta, izbacivanjem pointera… Jedan od osnovnih razloga zašto je *C#* prevazišao *Javu* jeste što je kreiran iznova sa implementacijom u .net *framework*-u na umu, takođe kombinuje snagu *C* i *C++* jezika kao i produktivnost *Visual Basic*-a, takođe je zadrža poznatu sintaksu kako bi olakšao prelazak *C++* i *Java* programera.

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.

Na slici 4.1.1 prikazan je „Hello World“ program napisan u *C#* jeziku.



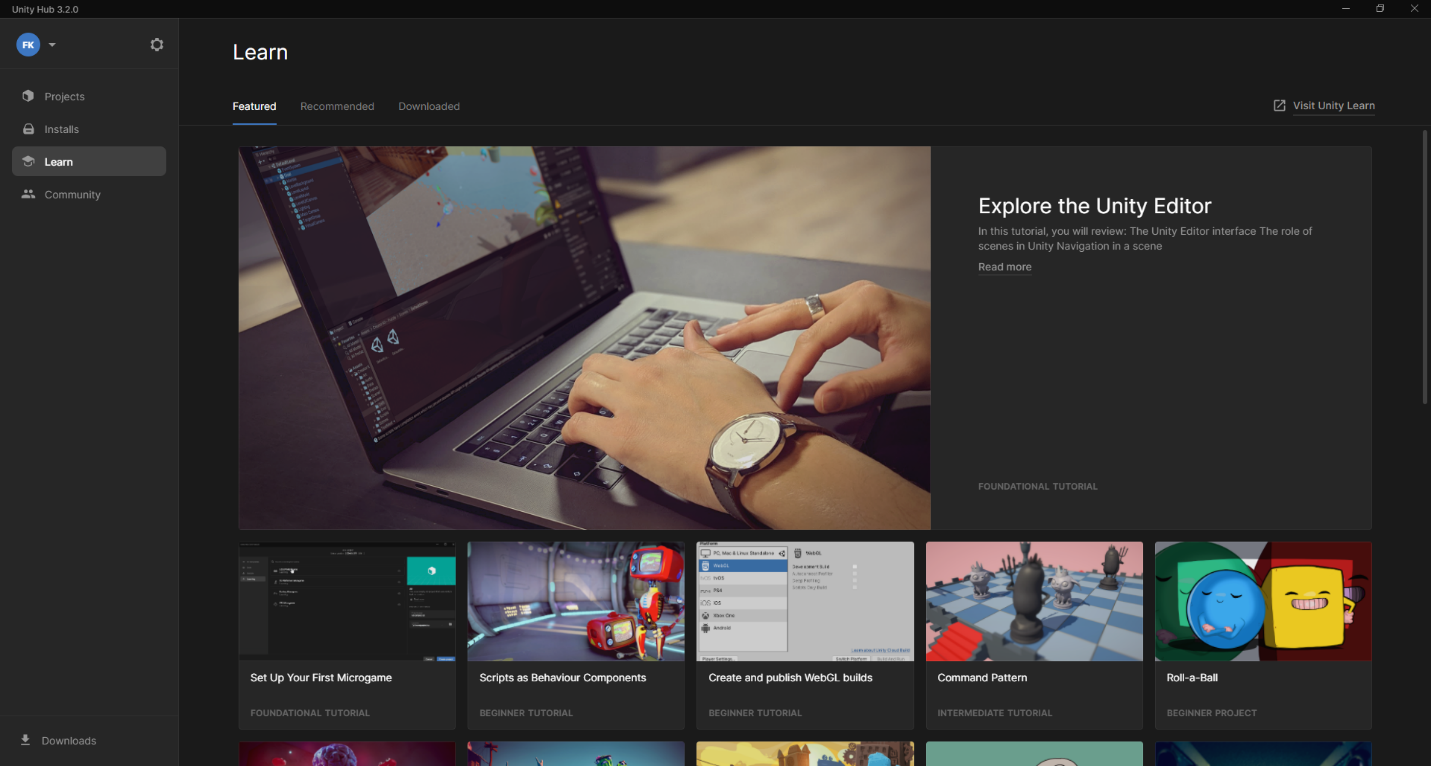
Sl. 4.1.1 ,,Hello world"

## 4.2 *Unity*

Unity je multi-platformsko okruženje za razvijanje video igara. Iako je u početku bio namenjen za Mac operativne sisteme, danas 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 najpopularnijih 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 igara, razne industrije kao što su filmska, industrijska, arhitektura i sl. koriste *Unity* za svoje potrebe. *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.

Ono što je dodatno približilo *Unity* programerima video igara (kako iskusnim tako i početnicima) jeste *Unity Learn* projekat koji je pun besplatnih lekcija (pisane ili u video formatu) koje pomažu u boljem razumevanju *Unity Game Engina*, i odlična su stanica za početak rada sa ovim programom. Takođe tu je i velika baza video snimaka na raznim društvenim mrežama od već iskusnih programera koji se trude da svoje znanje podele sa ostalima. Na slici 4.2.1 pokazana je početna strana *Unity Learn* dela aplikacije.



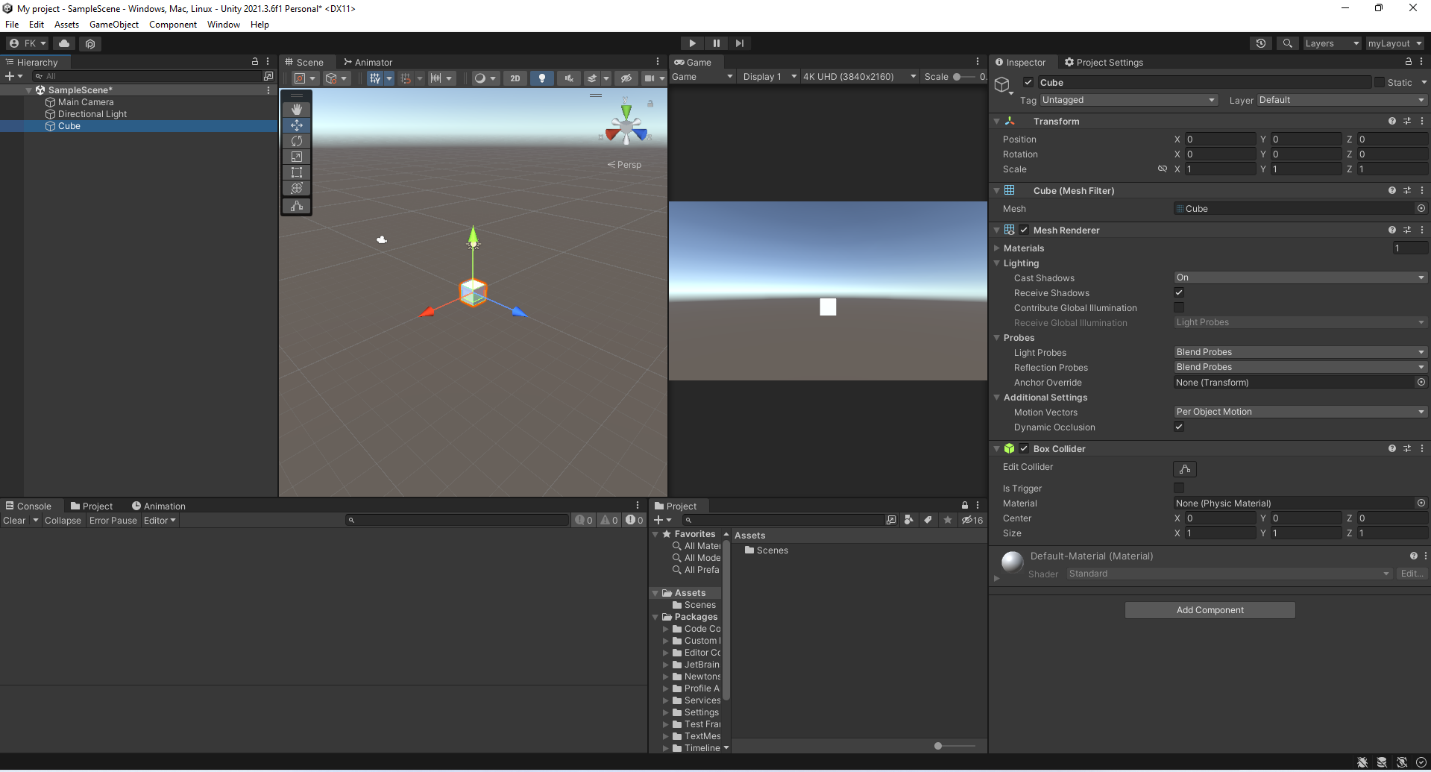
Sl. 4.2.1 Unity Learn

*A screenshot of a computer

Description automatically generated with medium confidence*

Sl. 4.2.2 Unity Hub 3.2.0

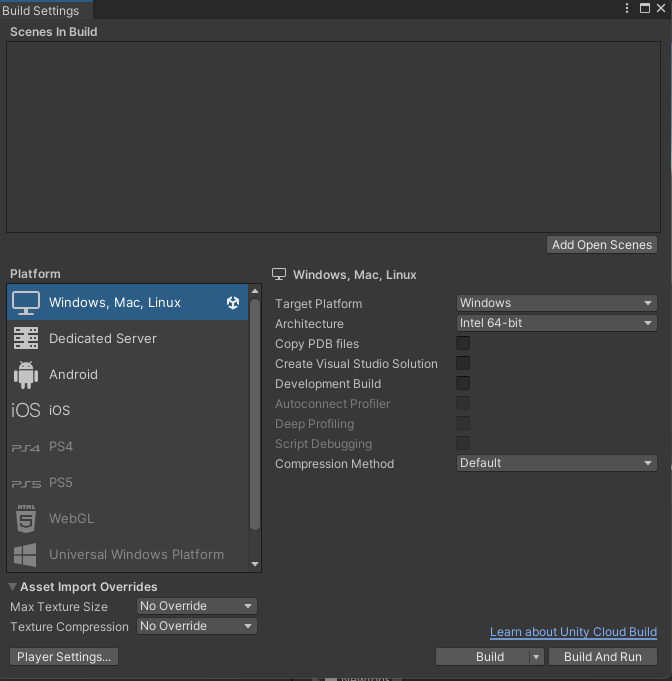
Na slici 4.2.2 prikazan je izgled Unity Hub-a, tačnije prikazan je proces kreiranja projekta. Pored osnovnih templejta *Unity* nudi i početničke *templejte* kao što su: FPS Microgame, 2d Platformer, Karting Microgame…



Sl. 4.2.3 Izgled Unity projekta

Na slici 4.2.3 prikazan je novo kreirani projekat sa odabranim templejtom ,,3D core“ sa dodatom kockom i malo preuređenim prozorčićima. Osnovni prozori su:

* *Hierarchy* – daje listu svih objekta u trenutno aktuelnoj sceni, takođe određuje redosled renderovanja objekata ukoliko su oni u nekom konfliktu (na istom su sloju i na istoj ili sličnoj poziciji pa je došlo do preklapanja objekata) u tom slučaju će se objekti prikazati na ekranu u zavisnosti od njihovog redosleda (odozgo na dole)
* *Scene* – je prozor koji omogućava dodavanje objekata privlačenjem na taj prozor kao i preuređivanje izgleda scene
* *Game View* – prikazuje kako će igrica izgledati na krajnjem uređaju
* *Inspector* – prikazuje zakačene komponente za trenutno odabrani objekat
* *Project Window* – prikazuje sve fajlove koje koristimo u projektu
* *Console* – prikazuje poruke o greškama



Sl. 4.2.4 Build settings

Na slici 4.2.4 prikazan je prozor za odabir platforme za koju želimo da izvezemo naš projekat kao i mogućnost biranja nekih dodatnih opcija kao što su arhitektura, verzija za testiranje, metoda kompresije…

# 5. Realizacija projekta

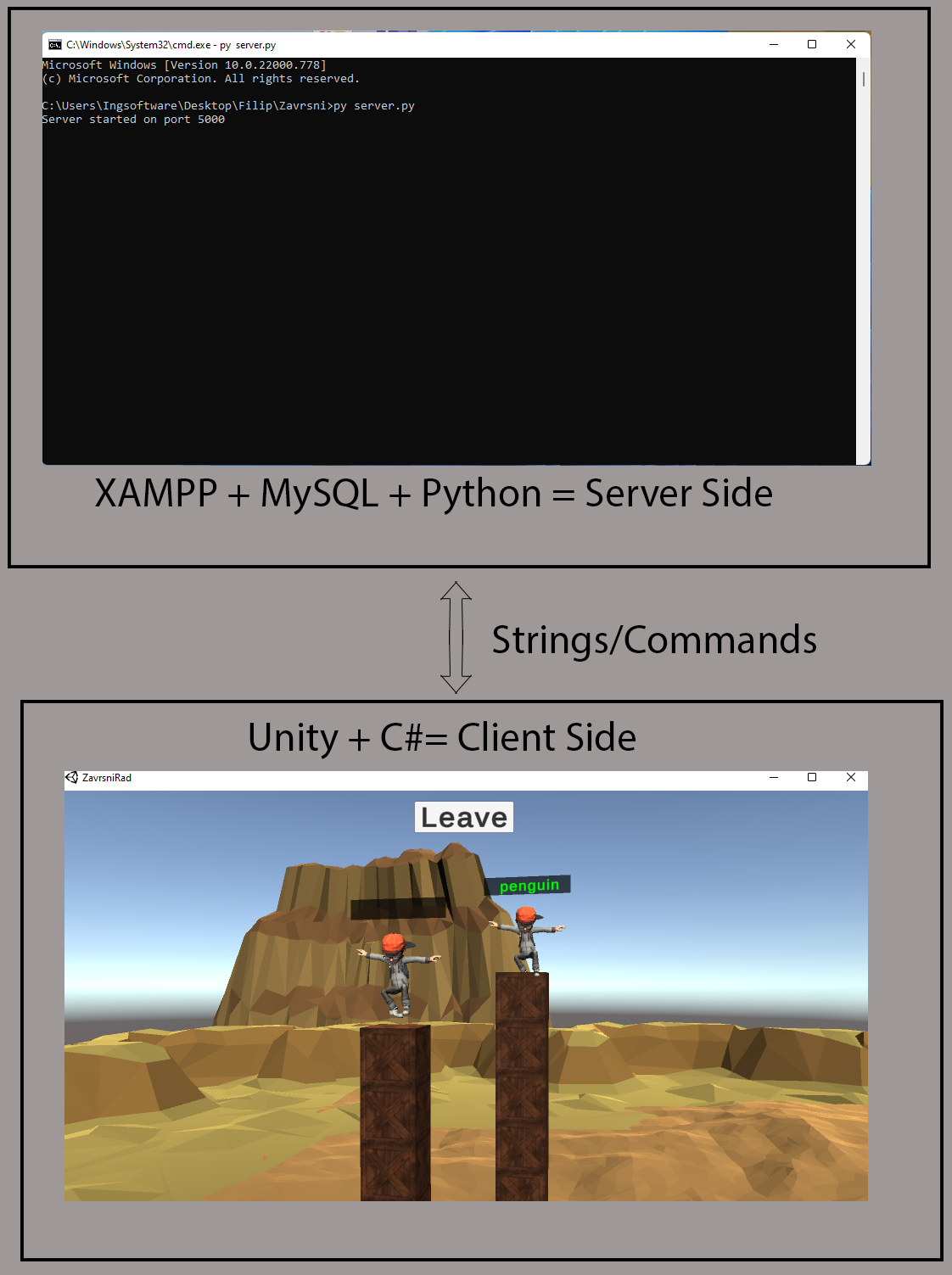
Opis projektnog zadatka i blok šema na kojoj se vidi čime je šta rešeno

## 5.1 Projektni zadatak

Projektni zadatak jeste kreiranje *python web servis*-a koji će služiti kao podrška *Unity* klijentu, tačnije *web servis* će omogućiti igranje multiplayer igrice na klijentskoj strani tako što će omogućiti klijentima:

* Registraciju, prijavu i obnavljanje lozinke kako bi čuvali svoje napretke,
* Dodavanje prijatelja kako bi igrica bila zanimljivija,
* Uvid u *leaderboard* radi kometitivnosti igrača,
* Mogućnost kreiranja prijateljskih i kompetitivnih mečeva,
* Mogućnost da do 5 igrača može da igra istu igru.

Korišćene tehnologije se mogu videti na slici 5.1.1.



Sl. 5.1.1 Blok šema strukture arhitekture

U nastavku rada dat je detaljan uvid u kod samog projekat na serverskoj strani uz dodatan opis koda.

## 5.2 Ostvarivanje komunikacije između 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 sadržaja poruke znaju šta treba da se desi. Sva komunikacija ide preko servera, nema komunikacije međusobno između klijenata.

### 5.2.1 Kreiranje TCP servera

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



### 5.2.2 Kreiranje TCP klijenta

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



### 5.2.3 Komade

#### 5.2.3.1 Komande koje klijent poziva na serveru

Klijent ne može sam pozvati komande, tj. ne postoji prozor u kome on može da ukuca komandu već se one automatski pozivaju kada za to dođe vreme.



#### 5.2.3.2 Komande koje server poziva na klijentu

Serverske komande se obično pozivaju kao odgovor na neku klijentsku komandu ali mogu i bez toga, primera radi pozivanje prijatelja u zajednički meč.



#### 5.2.3.3 Komande koje server poziva na serveru

Set komandi koje se pozivaju na serveru su uglavnom služile za *debugging.*



### 5.2.4 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



#### 5.2.4.1 Prijava korisnika

Prijava korisnika je bitna kako bi korisnici mogli da prelaze sa računara na računar bez gubitka svog napretka, forma za prijavu može videti na slici 5.2.4.1.1.



Sl. 5.2.4.1.1. login forma

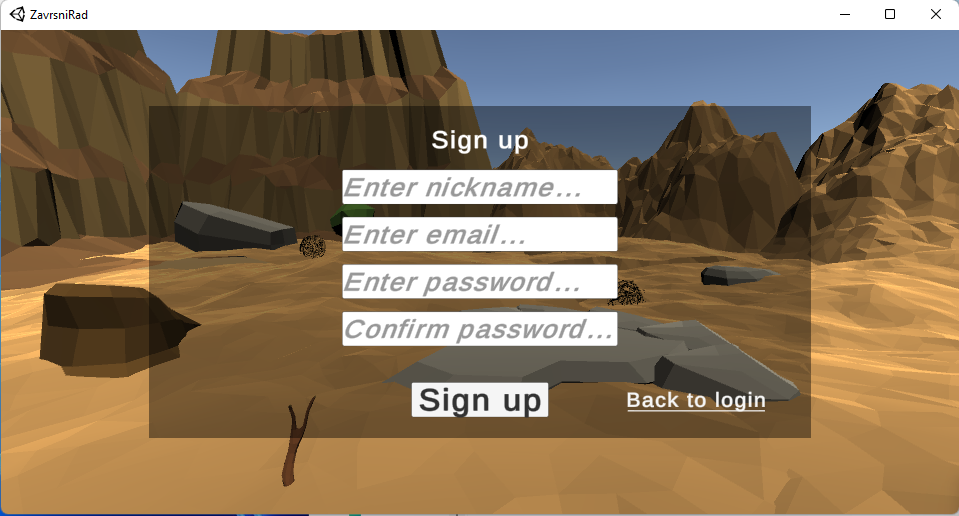


#### 5.2.4.2 Registracija korisnika



### 5.2.5 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. Na slici 5.2.5.1. prikazana je forma za kreiranje soba.



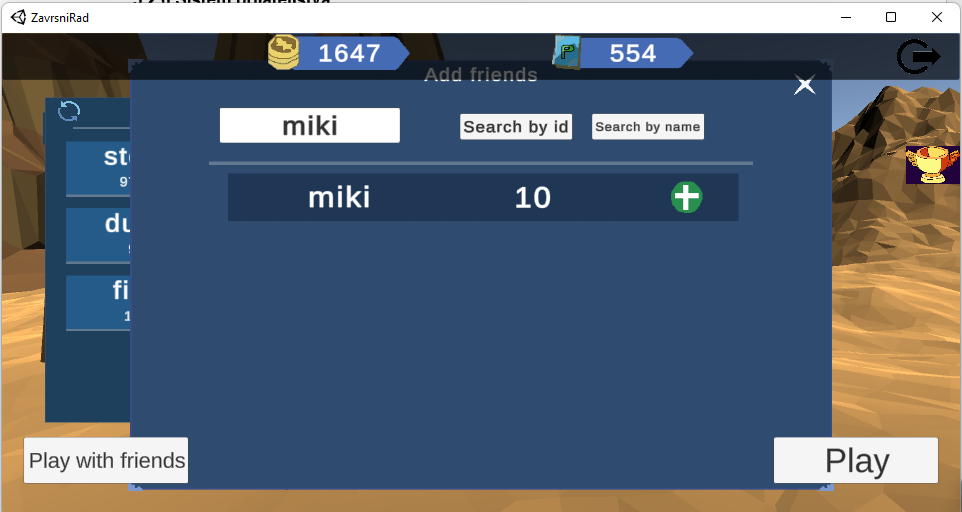
Kreiranje soba postiže se sledećim kodom



### 5.2.6 Sistem prijateljstva

#### 5.2.6.1 Pretraživanje

Mogućnost dodavanja prijatelja je jedna od funkcionalnosti koja čini igre zanimljivijima jer je uvek zanimljivije igrati sa drugarima, u početku je uredu imati samo opciju za pretraživanje po imenu ali ukoliko igrica dozvoljava unošenje duplikatnih imena onda može postati teško naći svog prijatelja stoga je bitno imati pretragu po nečim unikatnim. Na slici 5.2.6.1. prikazana je slika pretrage prijatelja.



Sl. 5.2.6.1 Pretraga prijatelja



#### 5.2.6.2 Dodavanje prijatelja

Sistem dodavanja prijatelja je napravljen tako da kako bi 2 igrača postala prijatelji neophodno je da jedan drugog pretraže po imenu ili ID-u i da se međusobno dodaju, nakon toga će ih sistem prepoznati kao prijatelje.



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

Indikator o tome da li je prijatelj trenutno aktivan ili ne je bitan deo multiplayer igrice, u mom projektu je to realizovano tako što ukoliko je prijatelj aktivan pojaviće se zeleno dugme pored njegovog imena što označava da je prijatelj trenutno aktivan i ujedno služi za pozivanje prijatelja u zajednički meč. Vizualni prikaz je dat na slici 5.2.6.3.1.



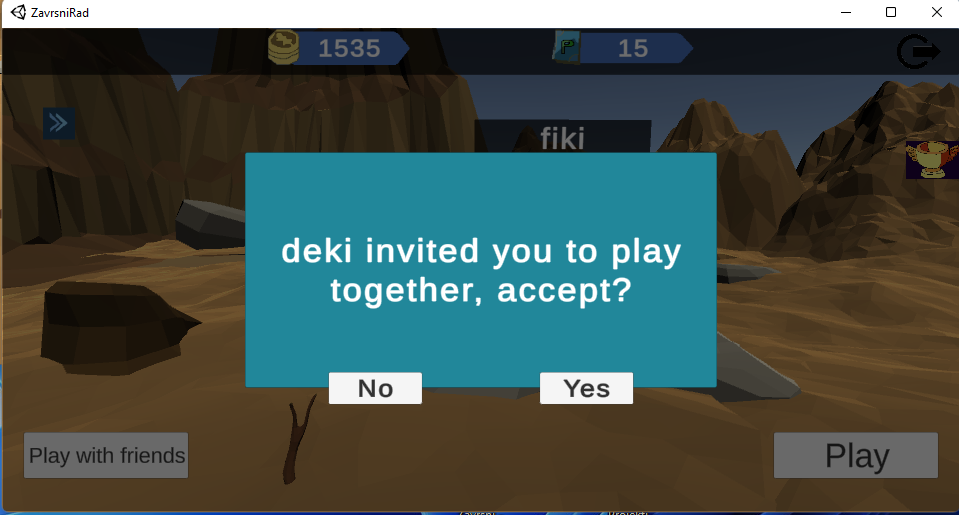
Sl. 5.2.6.3.1. prikaz sistema aktivnosti

Deo koda na serverskoj strani koji je zadužen za ovu funkcionalnost:



#### 5.2.6.4 Pozivanje prijatelja u sobu

Bez sistema za pozivanje u zajedničku igru sistem prijateljstva nema neku ulogu. Da bi jedan igrač pozvao drugog u zajedničku igru potrebno je da prvo on bude u nekoj sobi i da soba nije puna a zatim da prijatelj koga želi da pozove već nije u nekom meču i da je aktivan. Izgled zahteva za zajedničkom igrom dat je na slici 5.2.6.4.1.

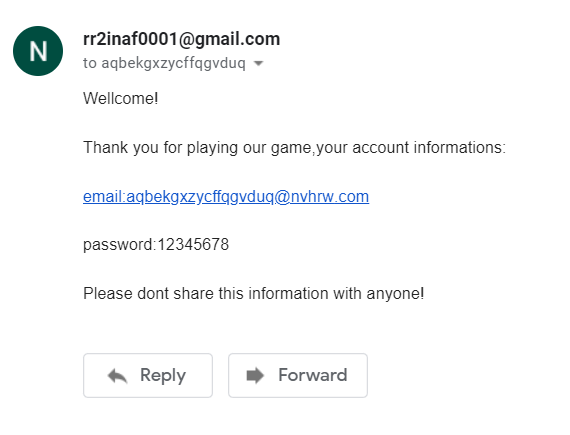


Sl. 5.2.6.4.1 zahtev za zajedničkom igrom



### 5.2.7 Slanje emajlova

Slanje emajlova ili nekog vida poruke je bitan za svaku savremenu igricu kako zbog slanja promotivnih poruka ili obaveštenja tako i zbog upravljanja korisničkim profilom. Na slici 5.2.7.1. prikazan je izgled poruke dobrodošlice klijentu.



Sl. 5.2.7.1. Poruka dobrodošlice



### 5.2.7 *GameLoop*

*GameLoop* je najbitniji segment svake igre, u ovom radu *GameLoop* se sastoji iz 5 rundi u kojima igrači dobijaju pitanje i trebaju da daju što duži tačni odgovor, nakon unetog odgovora ili nakon isteka 15sec od pojavljivanja pitanja server proverava tačnost odgovora svakog igrača i dodeljuje bodove na osnovu dužine i tačnosti odgovora, nakon čega server smanjuje broj bodova igračima kako bi igrači mogli da izgube, proces se ponavlja 5 rundi ili dok ne ostane samo 1 igrač. U nastavku je dat detaljan pregled koda *GameLoop*-a uz dodatno objašnjenje koda.

#### 5.2.7.1 Započinjanje igre



#### 5.1.7.2 Odgovaranje na pitanje



#### 5.2.7.3 Tok igre

##### 5.2.7.3.1 Slanje poruke svima u sobi



##### 5.2.7.3.2 Dodavanje kutija



##### 5.2.7.3.3 Trajanje runde



##### 5.2.7.3.4 Provera odgovora



##### 5.2.7.3.5 Uklanjanje kutija na kraju runde



5.2.7.6 Provera da li je neki igrač izgubio



##### 5.2.7.7 Nagrađivanje igrača



##### 5.2.7.8 Početak igre



##### 5.2.7.9 Igra



##### 5.2.7.10 Završnica igre



# 6. Zaključak

U ovom radu realizovan je web servis u *python* programskom jeziku koji služi kao podrška *Unity* klijentskoj aplikaciji koja je rađena pomoću *C#* programskog jezika. Pored ova dva programska jezika korišćen je i *XAMPP* koji jeomogućio lokalno podizanje relacione baze podataka MySQL. Web servis je omogućio: kreiranje korisničkog naloga, obnavljanje lozinke korisničkog naloga, prijavljivanje na korisnički nalog, čuvanje podataka o napredovanju i dostignućima korisnika, dodavanja prijatelja, igranje sa prijateljima, igranje kompetetivnih mečeva, održavanje lestvice najboljih igrača po odnosu pobeda i poraza, i po broju osvojenih bodova.

Izazov rada je bio uspostavljanje veze između klijenta i servera što je rešeno upotrebom *Socket-*a, kao i implementirati mogućnost igranja više nezavisnih mečeva istovremeno. Kroz ovaj rad poboljšao sam svoje veštine pasajući *python* serverski kod, naučio sam nešto više o *Unity Game Engin-u,* i sigurniji sam u pisanju SQL iskaza.

# 7. Literatura

[1] Klijent server tehnologija, prezentacija profesora Mirka Kosanovića: <https://vtsnis.edu.rs/wp-content/plugins/vts-predmeti/uploads/1521843285KS%20Predavanje%203%202016.pdf> - jun 2022.

[3] ,, Video game industry - Statistics & Facts ”, <https://www.statista.com/topics/868/video-games/#topicHeader__wrapper> - jun 2022.

[4] ,, Introduction to Gaming Server Hosting ”, <https://digital.com/best-web-hosting/gaming-server-hosting/#:~:text=A%20gaming%20server%20host%20is,virtual%20world%20and%20play%20together> - jun 2022.

[5] ,, What is a Socket?” : <https://www.tutorialspoint.com/unix_sockets/what_is_socket.htm> - jun 2022.

[6] ,, Introduction To PYTHON“ <https://www.geeksforgeeks.org/python-language-introduction/#:~:text=Python%20is%20a%20widely%20used,in%20fewer%20lines%20of%20code>. - jun 2022.

[7] ,, MySQL”, <https://dev.mysql.com/doc/> - jun 2022.

[8] ,, XAMPP”, <https://www.apachefriends.org/docs> - jun 2022.

[9] ,, Unity (game engine)”, <https://docs.unity3d.com/Manual/index.html> - jun 2022.

[10] ,,Python”, predavanja profesora Slavimira Stošovića iz predmeta Web Programiranje

[11] ,,MySQL”, predavanja profesora Dušana Stefanovića iz predmeta Baze Podataka i Administriranje Baza Podatak.