

# **Faza I - Arhitekturni projekat AmongUs - OurWay**

Luka Mladenovic 16766

Nemanja Momcilovic 16768

Sadrzaj

1. Kontekst i cilj projekta…………………………………………………………………..2
2. Arhitekturni zahtevi……………………………………………………………………..2
   1. Glavni funkcionalni zahtevi…………………………………………………….2
   2. Nefunkcionalni zahtevi…………………………………………………………6
   3. Tehnicka i poslovna ogranicenja………………………………………………6
3. Arhitekturni dizajn……………………………………………………………………….6
   1. Arhitekturni obrasci…….………………………………………………………6
   2. Strukturni pogled……….………………………………………………………7
   3. Bihevioralni pogled……..………………………………………………………8
   4. Implementacija………….………………………………………………………9

Kontekst i cilj projekta

Projekat predstavlja realizaciju popularne drustvene igre dedukcije kao web aplikacije AmongUs - OurWay. Igra se sastoji od tima clanova posade (Crewmates) koji plove svemirom, medjutim u njihove redove postoje uljez/i (Imposter(s)) koji glume clanove posade. Cilj posade je zavrsavanje zadataka ili izbacivanje svih uljeza, antagonisticki tome cilj uljeza je uklanjanje clanova posade ubijanjem ili izbacivanjem. Glavno oruzje posade je njihova saradnja, dok je prednost uljeza sposobnost sabotiranja kljucnih delova broda kao i koriscenje posebnih prolaza za brze kretanje po mapi. Cilj projekta je eliminacija upotrebe 3rd party aplikacije (discord/teamspeak itd.) za komunikaciju u igri kao i prebacivanje ove igre na web pretrazivac.

# **Arhitekturni zahtevi**

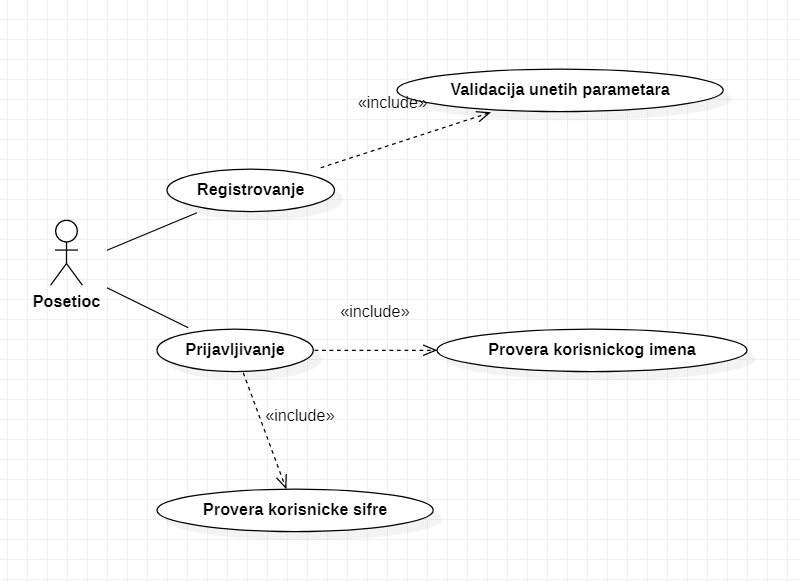
## Glavni funkcionalni zahtevi:

* **Prijavljivanje korisnika** - Korisnici ce imati mogucnosti prijavljivanja. Korisnicki nalozi ce sadrzati istoriju odigranih partija kao i mogucnost pregleda istorije jedne partije.
* **Vecinu funkcionalnosti igre AmongUs:**
* **Formiranje predvorja igre** - Korisnici mogu da formiraju predvorje igre koje ima jedinstven kod za prikljucivanje.
* **Pocetak igre** - Nakon sakupljanja igraca u predvorju igra moze otpoceti. Pocetak inicira kreator predvorja.(Igra ne moze poceti ako je u predvorju manje od 4 igraca)
  + - **Uloge igraca** - Postoje 2 vrste uloga:  
       - **Crewmate(clan posade)** - Cilj ove uloge je da zavrsi sve  
       zadatke pre nego sto uljez izvrsi dovoljno ubistava ili da  
       Izbaci uljeza  
       - **Impostor(Uljez)** - Cilj ove uloge je ubijanje clanova   
       Posade i prikrivanje svojih namera.
    - **Akcije igraca tokom igre:** - **Zapocni zadatak** - Clanovi posade mogu da zapocnu   
       zadatak kada se nalaze u njegovoj blizini.  
       - **Ubi** - Uljezi mogu da ubiju clana posade kada se nalaze  
       U njegovoj blizini. Nakon sto su ubijeni, igraci postaju  
       duhovi i mogu da nastave da zavrsavaju zadatke.  
       - **Koristi tunel** - Uljezi mogu na posebnim mestim na  
       koriste pomocne tunele za brze kretanje po mapi.  
       - **Pritisni dugme** - Na mapi postoji dugme cijim pritiskom   
       se saziva sastanak. Tokom sastanka otvaraju se svi   
       komunikacioni kanali i igraci ulaze u debatu kako bi  
       otkrili uljeza.   
       - **Prijavi telo** - U slucaju da igrac pronadje “ubijenog”   
       igraca, telo se moze prijaviti nakon cega se odmah   
       saziva sastanak.  
       - **Glasanje** - Tokom sastanka igraci mogu da glasaju za  
       druge igrace. Igrac sa najvecim brojem glasa se  
       “Izbacuje” iz igre. Izbaceni igraci postaju duhovi.
    - **Kraj igre** - Uslovi pobede clanova posade:  
       - Izbaceni su uljezi.  
       (ili)  
       - Zavrseni zadaci.  
        
        
       - Uslovi pobede uljeza:

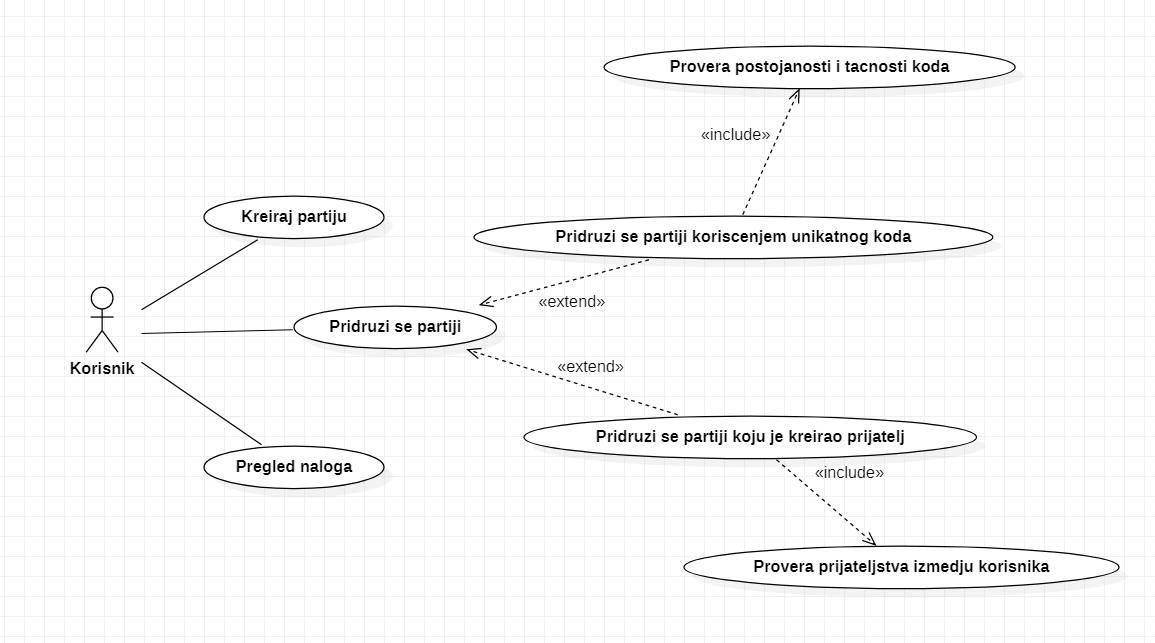
- Preostali broj zivih clanova posade je jednak broju   
 preostalih uljeza.

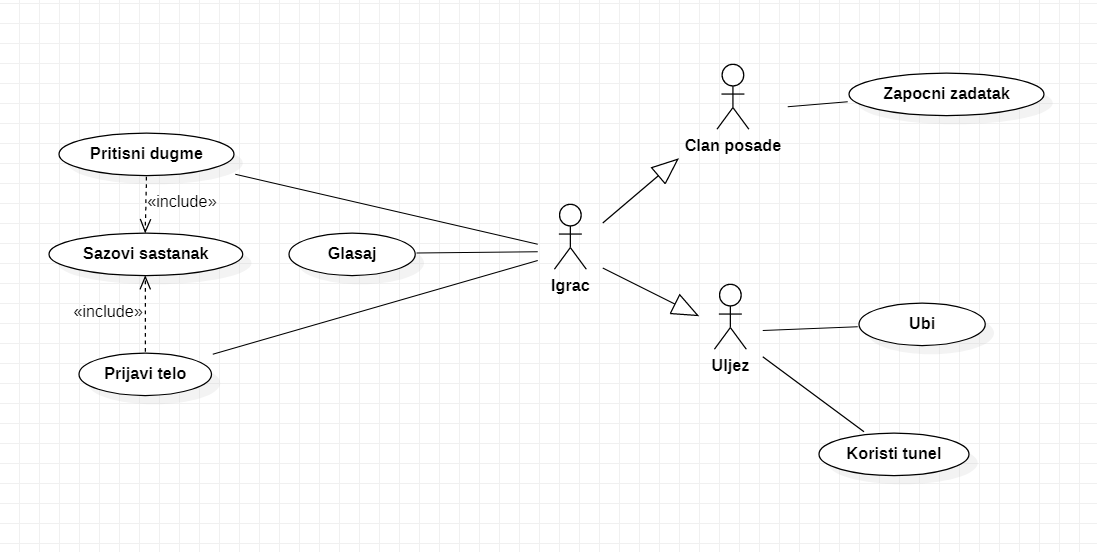
* **Dodavanje prijatelja** - Korisnici mogu dodavati prijatelje, sto dalje omogucava   
  pridruzivanje partiji bez poznavanja unikatnog koda predvorja.

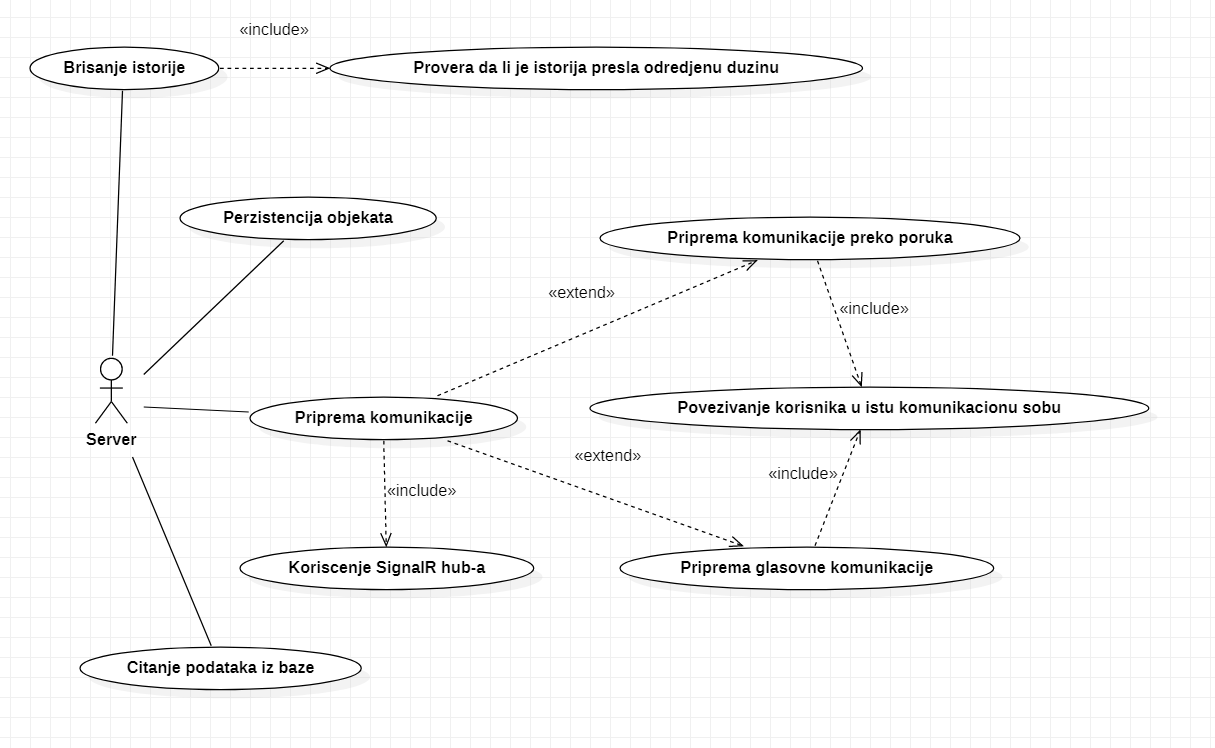
## Arhitekturno znacajni slucajevi koriscenja



#### 







## 

## Ne-funkcionalni zahtevi:

* Kasnjenje prilikom razmene poruka i glasovne komunikacije mora biti zanemarljivo.
* Kretanje igraca mora biti “glatko” i bez naglih skokova.
* Raspodela uloga mora biti slucajna i nepredvidljiva.
* Ubacivanje “bad luck” sistema koji sprecava da odredjen igrac dobije istu ulogu nakon odrejenog broja partija. Broj mora varirati kako bi prethodni zahtev bio ispunjen.
* Jednostavan i intuitivan korisnicki interfejs.
* Maksimalni broj igraca u jednoj partiji je 10.
* Obezbediti skaliranje broja uljeza u slucaju napustanja partije. (Funkcionise samo u predvorju)

## Tehnicka i poslovna ogranicenja:

* Velicina sobe ce biti ogranicena na 10 igraca zbog performansi i integriteta partije.
* Kod svake sobe mora biti jedinstven u sistemu.
* U toku jedne partije svaki igrac moze imati samo jedan zivot i moze pricati samo dok je sastanak posade u toku.
* Aplikacija ce biti optimizovana za sledece Web pretrazivace: Internet Explorer 8.0 i noviji, Opera 8.0 i noviji, Google Chrome i Firefox (Mozilla).

# **Arhitekturni dizajn**

## Arhitekturni obrasci

**Publish-Subscribe obrazac** – Sloj poslovne logike poseduje publish-subscribe mogucnosti kako bi omogucio distribuirano izvrsavanje aplikacije. Bice iskoriscen za prosledjivanje dogadjaja tokom igre.

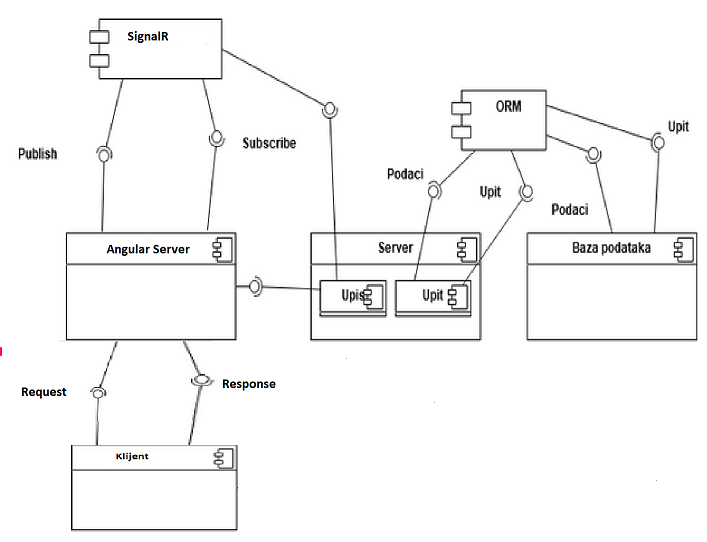
**MVC** – Angular framework ce predstavljati **V**iew komponentu, same Angular komponente koje ce biti renderovane na stranicama. REST API na back end-u ce predstavljati **M**odel i **C**ontroller i bice zaduzen za obezbedjivanje podataka za View komponente.

**Repository –** Sistem će sadržati centralizovanu bazu podataka.

**Višeslojna arhitektura –** Troslojna arhitektura aplikacije.



## Strukturni pogled



## Bihevioralni pogled

## 

## 

## 

# **Implementacija**

* Na front end-u ce biti koriscen Angular.
* Na back end-u ce biti koriscen ASP .NET Core Framework i programski jezik C#.
* Kao ORM ce biti koriscen Microsoft EntityFramework Core.
* Baza ce biti Azure database.

**Angular i MVC obrazac**

Angular koristi komponente koje predstavljaju templejtske direktive. Sadrzi 2 vrste direktiva:

1) Strukturne direktive: menjaju raspored DOM-a izmenom njegovih elemenata.

2) Direktive atributa: menjaju kako se DOM ponasa i izgled elemenata.

Komponente definisu preglede (**views**), koju su skupovi elemenata na ekranu koje Angular moze da bira i modifikuje prema logici i podacima naseg programa. Komponente koriste **servise (services),** koji pruzaju specificne funkcionalnosti koje nisu direktno povezane sa pregledom/izgledom. Servisi se mogu ubaciti (eng. noun **Injection**) u komponente kao zavisnosti ( eng. **Dependencies**), sto doprinosi modularnosti, povecava upotrebljivost i efikasnost koda.

U angular **aplikacijama**, **View** je predstavljem .**html** templejtima u kombinaciji sa .**css** delovima komponenti. Ako pratimo MVC sablon, View bi definisao mesta u koja mozemo da ubacimo nase podatke. Takodje on pruza metode za registrovanje **callback** kad god je dugme kliknuto. U ovom smislu, View ne bi znao za postojanje **Controller**-a. Kontroler za za postojanje View-a i Modela i povezuje ih. Angular ima drugaciji pristup, nazvan **MVVM pattern (Model-View-ViewModel)**.

**ViewModel** se nalazi u **.ts** fajlu komponenata.

U @Component dekotatoru klase, mozete videti referencu na **View .html** i **.css** fajlove. U ostatku klase, s druge strane, nema reference na View. Umesto nje, ViewModel koji se nalazi u klasi Componente otvara atribute i metode kojima View moze da pristupi. U prevodu, u odnosu na MVC arhitekturu, zavisnost je obrnuta.

**ViewModel** ne zna da View postoji, ali pruza API slican Modelu koji moze da se koristi od strane View-a. Posledica toga je da View mora da zna za postojanje ViewModela i njegovih metoda. Razlog za obrtanje zavisnosti je smanjenje ponavljanja koda. Podaci su dvosmerno spregnuti izmedju View-a i ViewModel-a. Podaci u View-u su uvek sinhronizovani sa podacima u ViewModelu.