FAZA I

* ARHITEKTURA SOFTVERA -

*SecretLink*

Tim:

Ivana Avramović, 18067

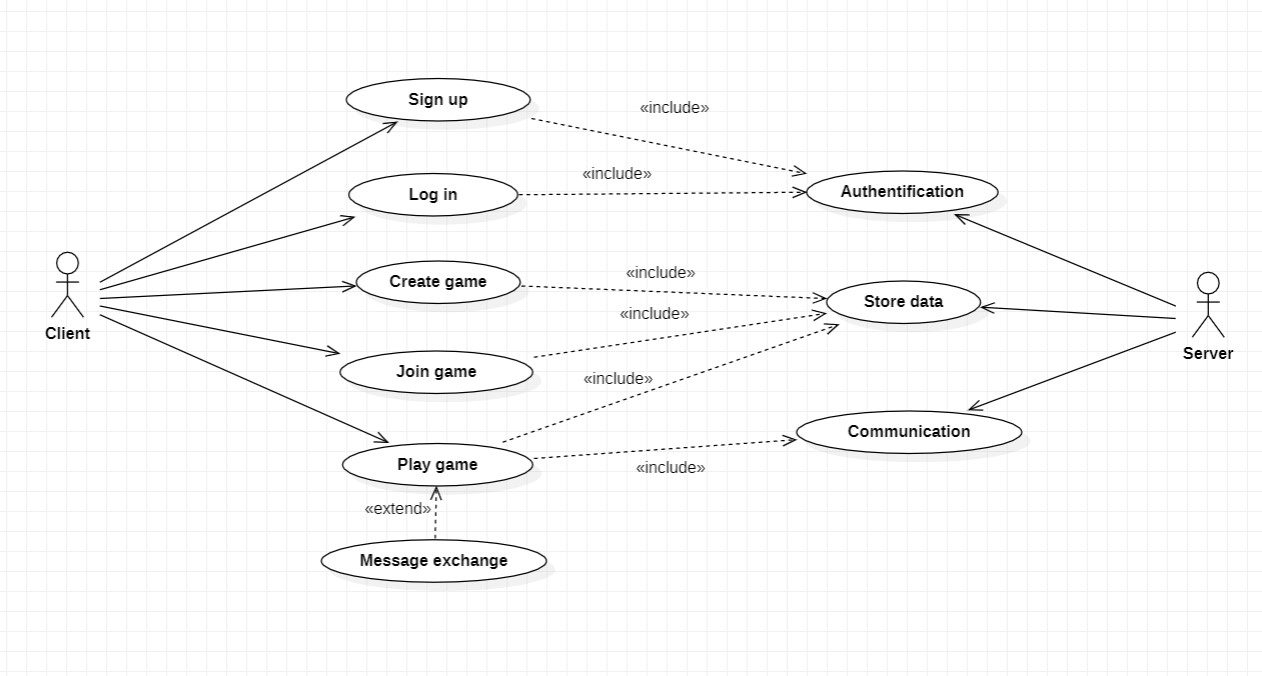
Kristina Dimitrijević, 18131

1. **KONTEKST I CILJ SOFTVERSKOG PROJEKTA**

SecretLink je online multiplayer igra zasnovana na popularnoj društvenoj igri Connect4 i igri pogađanja reči. U igri mogu učestvovati 2 igrača gde je cilj svakog od njih pogađanje nepoznate reči. U početku je otkriveno samo jedno slovo, kao i broj slova koja čine reč. Igrači se prvo takmiče u igri povezivanja 4 kruga zaredom (horizontalno, vertikalno ili po dijagonali), ‘pobedniku’ ovog dela igre se otkriva još jedno slovo reči koju treba pogoditi, takođe dobija šansu da prvi pogađa reč. Drugi igrač ima samo mogućnost pogađanja reči (nakon prvog igrača). Igra se odvija sve dok neko od igrača ne pogodi reč (povezivanje 4 kruga zaredom pa pogađanje reči). Na početku jedan igrač kreira partiju kojoj se priključuje drugi igrač.

1. **ARHITEKTURNI ZAHTEVI**
   1. **Funkcionalni zahtevi**

* registracija korisnika
* prijavljivanje korisnika
* kreiranje igre
* pridruživanje igri
* odigravanje poteza
* kraj igre
* razmena poruka



Slika 1. Use-case dijagram sistema

* 1. **Nefunkcionalni zahtevi**
* **Performanse** – treba da se obezbedi što manje vreme odziva (< 7s) jer je reč o sistemu koji radi u realnom vremenu i kako bi se obezbedilo što bolje korisničko iskustvo.
* **Pouzdanost** – igra treba da bude dostupna u 99% slučajeva, ukoliko dođe do prekida konekcije između servera i klijenta pri ponovnom povezivanju klijenta treba obezbediti mogućnost da nastavi tamo gde je stao.
* **Skalabilost** – potrebno je da podrži veći broj korisnika i više zahteva istovremeno kao i rast broja korisnika u budućnosti.
* **Modifikabilnost** – realizovati sistem tako da podržava lako dodavanje novih i izmenu vec postojećih funkcionalnosti.
* **Lakoća korišćenja** – igra treba biti intuitivna i laka za navigaciju novim korisnicima, interfejs treba razviti u skladu sa tim.
* **Kompatibilnost** – osigurati da igra funkcioniše na svim popularnim web pretraživačima.
  1. **Tehnicka i poslovna ogranicenja**
* **Skrivenost baze podataka** – način reprezentacije podataka u bazi je skriven od korisnika. Korisnicima su dostupni samo podaci predviđeni za prikaz.
* **Komunikacija** – potrebno je obezbediti dva tipa komunikacije: sinhronu komunikaciju (komunikacija između klijenta i servera) i asinhronu komunikaciju (razmena poruka između klijenata, kao i prosleđivanje izmene stanja svim igračima).
* **Pristup preko web-a** – neophodno je obezbediti korisncima da pristupe igri putem web-a (koristeći odgovarajuće tehnologije koje pružaju mogućnost interakcije i komunikacije korisnika i sistema).

1. **ARHITEKTURNI DIZAJN**

U nastavku su dati arhitekturni obrasci koji će biti iskorišćeni za realizaciju sistema.

* + 1. **LAYERED OBRAZAC**

Aplikacija će imati 3 sloja: prezentacioni sloj, serverski sloj i sloj perzistencije (baza podataka).

**Prezentacioni sloj** se izvršava na klijentu i on je odgovoran za prikazivanje informacija korisnicima i za interakciju korisnika sa sistemom. On komunicira sa slojem ispod sebe, a to je serverski sloj. **Serverski sloj** se izvršava na serveru i implementira funkcionalnosti koje aplikacija treba da pruži korisniku. Poslovna logika se nalazi ovde, a ovaj sloj koristi podatke iz sloja ispod sebe (sloj baze podataka). Komunicira i sa prezentacionim slojem. Komunikacija sa klijentima može biti sinhrona preko RESTful API-ja, a može biti i asinhrona preko message broker-a. **Sloj perzistencije** predstavlja samu bazu podataka, koja će skladištiti podatke relevantne za aplikaciju.

* + 1. **MVC OBRAZAC**

MVC (Model-View-Controller) obrazac je popularan pristup razvoju softvera koji se često koristi u web aplikacijama. Obrazac će biti iskorišćen u sklopu .NET Web API-ja.

Model predstavlja logiku podataka i operacija nad podacima. Ovaj sloj je odgovoran za pristup podacima, njihovo ažuriranje i slanje obaveštenja View i Controller slojevima o promenama. View predstavlja korisnički interfejs (UI) i odgovoran je za prikazivanje podataka korisnicima. Controller je odgovoran za obradu zahteva korisnika, komuniciranje sa Modelom i prosleđivanje podataka View-u.

* + 1. **REPOSITORY OBRAZAC**

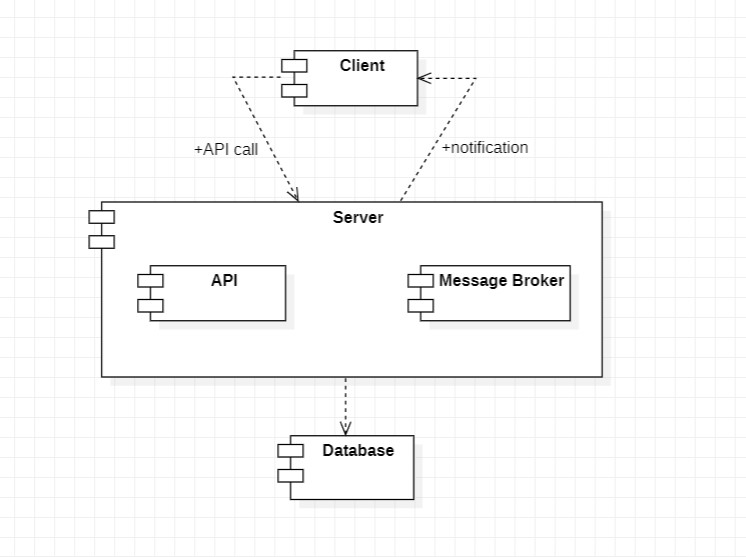
Za implementaciju perzistencije koristiće se centralizovano skladište, odnosno, centralizovana baza podataka. Serverska aplikacija pristupa i radi sa centralnim skladištem preko Entity Framework-a.

* + 1. **PUBLISH-SUBSCRIBE OBRAZAC**

Publish-Subscribe obrazac (ili Pub-Sub obrazac) je softverski obrazac dizajna koji omogućava komunikaciju između različitih delova sistema putem asinhronog događajnog modela. U aplikaciji *SecretLink* ovaj obrazac će biti implementiran tako da koristi message-broker. Ovaj obrazac će omogućiti svim igračima jedne “sobe” da u realnom vremenu primaju informacije o izmenama stanja igre, kao i prijem poruka od ostalih igrača.

* 1. **GENERALNA ARHITEKTURA**

Arhitektura sistema podrazumeva postojanje klijenta, servera i baze podataka u kojoj će se čuvati informacije o korisnicima i njihovim igrama.

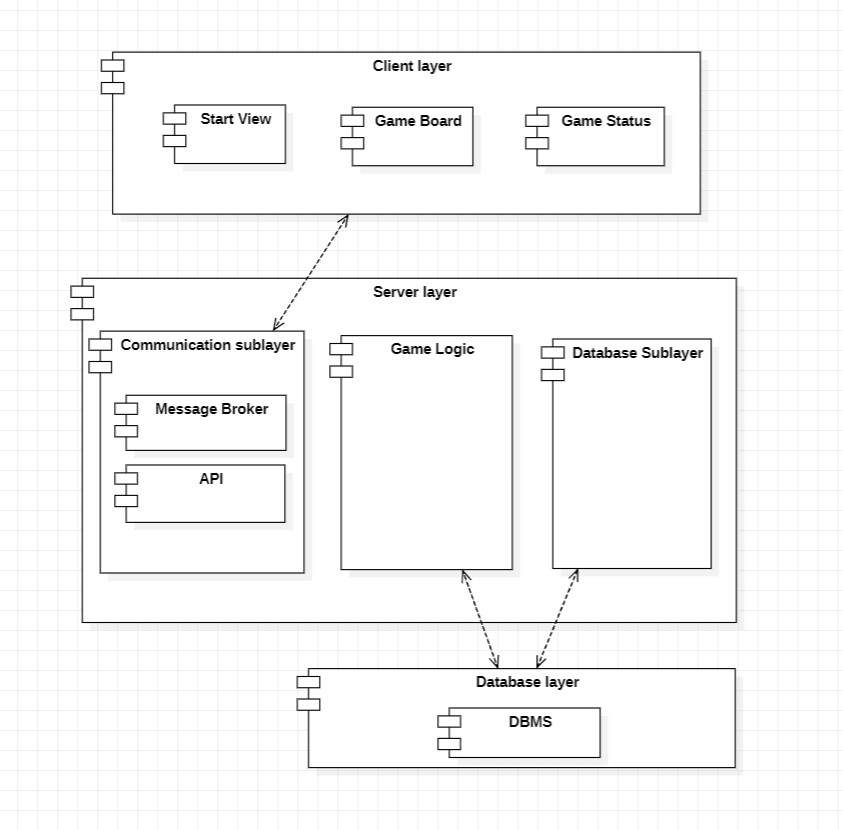
****

* 1. **STRUKTURNI POGLED**

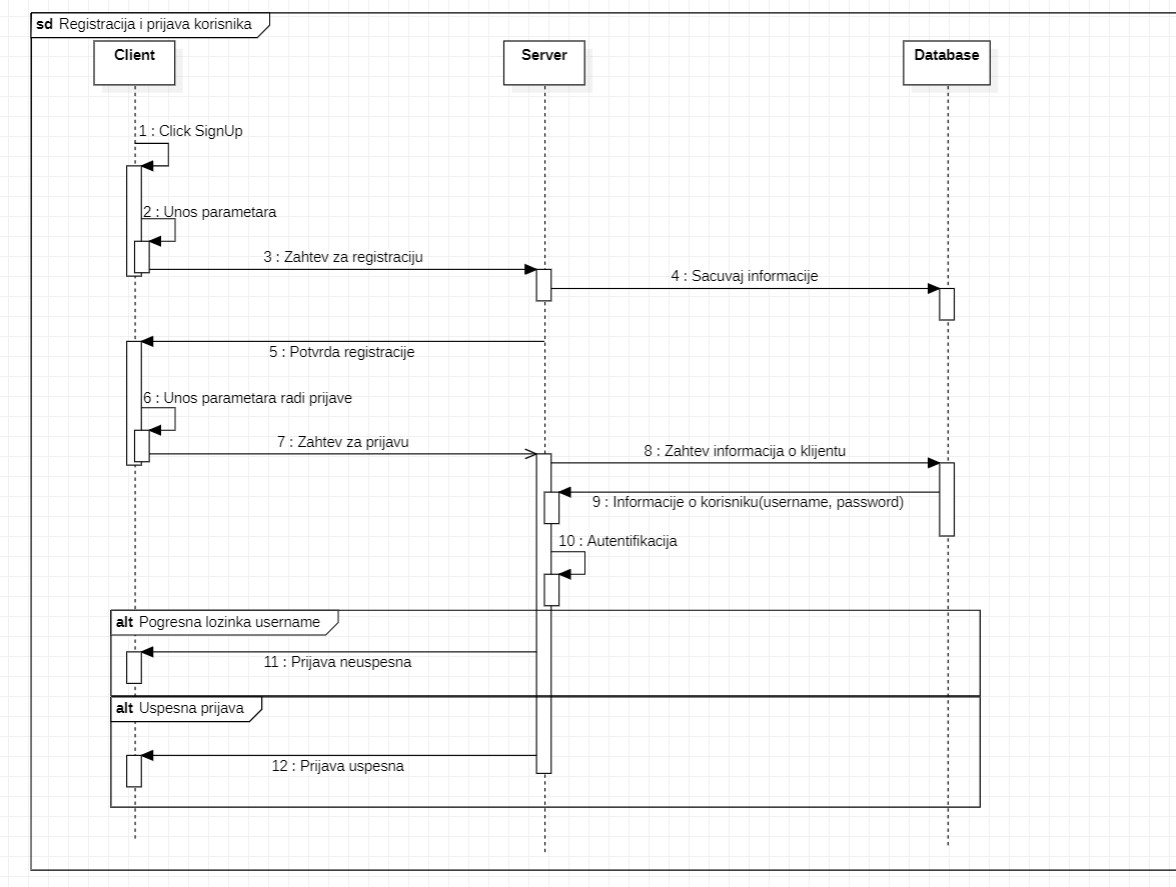
Sledeći dijagram ilustruje strukturu sistema navodeći komponente sistema kao i njihovu međusobnu povezanost.

Serverska aplikacija je zadužena za komunikaciju sa bazom podataka. Komunikacioni podsloj obuhvata RESTful Web API za sinhronu komunikaciju i Message Broker za asinhronu komunikaciju sa klijentom. **Game Logic**: Poslovna logika koja upravlja samom igrom, provjeravajući ispravnost poteza, pratiti redosled igrača, i određivanje pobednika. Database **Sublayer**: Koristi se za čuvanje informacija o igrama (npr. trenutno stanje ploče, potezi igrača), ova komponenta omogućava pristup i ažuriranje podataka.

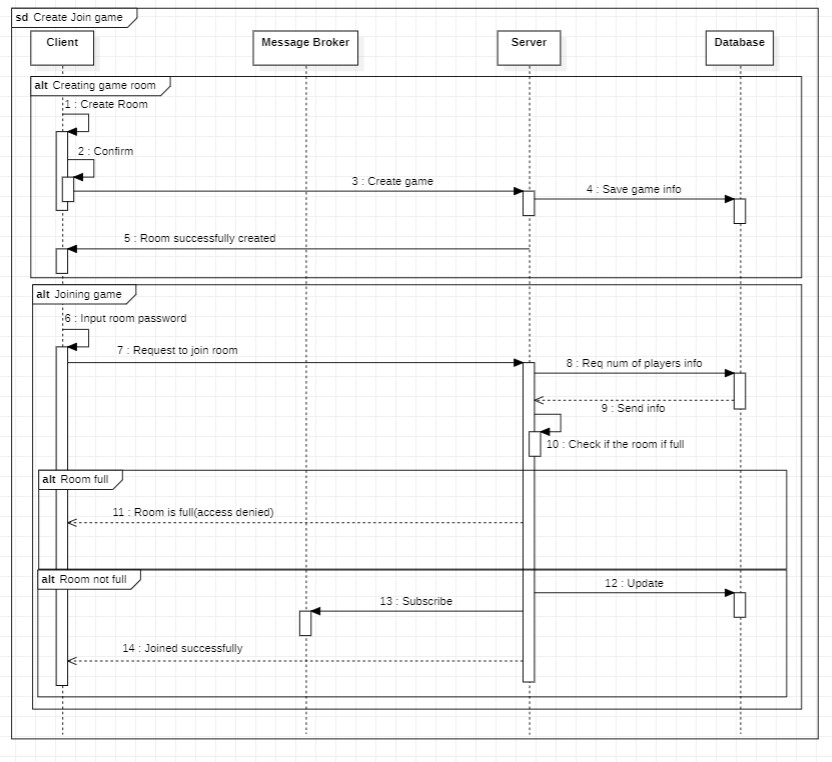
Na sloju perzistencije se nalazi DBMS kao konekcija sa bazom podataka. Baza podataka koja se koristi za skladištenje podataka o igrama, potezima igrača i rezultatima.



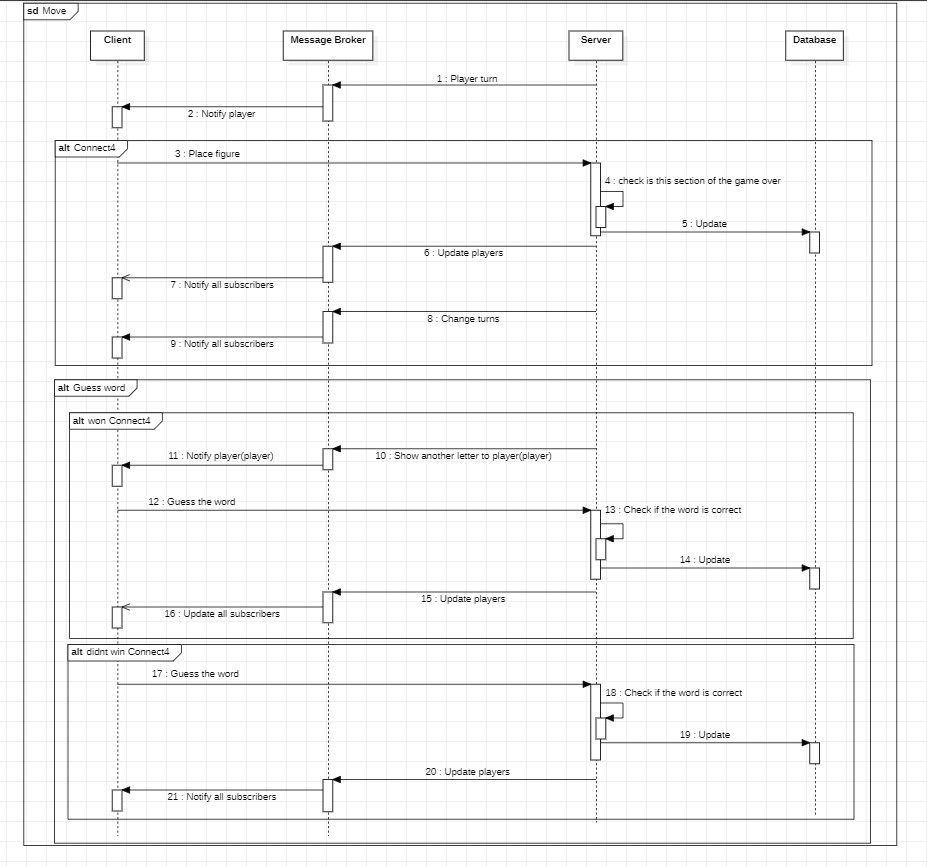
* 1. **BIHEVIORALNI POGLED**
* prijavljivanje i registracija korisnika

****

* kreiranje igre i prudruživanje igri

****

* odigravanje poteza



1. **IMPLEMENTACIJA**

* Frontend: ReactJS
* Backend: .NET Core
* Database: SQL DBMS
* SignalR - Message broker, omogućava API za real-time klijent-server komunikaciju
* ORM – Entity Framework