

UNIVERZITET U SARAJEVU  
ELEKTROTEHNIČKI FAKULTET  
ODSJEK ZA TELEKOMUNIKACIJE

Software Design Descriptions

# Protokol za geografsko lociranje korisnika

Ćutahija Zerina, 1685/17085  
Hasanbegović Selma, 1574/17753  
Mahovac Nerman, 1575/17919  
Repeša Almin, 1684/17550  
Velić Nejra, 1634/17313

Sarajevo, 2021. godina

---

# Sadržaj

|   |           |
|---|-----------|
| <b>Sadržaj</b>  | <b>ii</b> |
| <b>Popis slika</b>  | <b>1</b>  |
| <b>1 Uvod</b>   | <b>2</b>  |
| 1.1 Svrha SDD dokumenta . . . . .   | 2         |
| 1.2 Opis projektnog zadatka . . . . .   | 2         |
| 1.3 Definicije, akronimi i skraćenice . . . . .   | 4         |
| 1.3.1 Definicije . . . . .  | 4         |
| 1.3.2 Akronimi i skraćenice . . . . .   | 4         |
| 1.4 Reference . . . . .   | 5         |
| 1.5 Pregled dokumenta . . . . .   | 5         |
| <b>2 Opis rada protokola</b>  | <b>6</b>  |
| 2.1 FSM dijagrami stanja . . . . .  | 6         |
| 2.1.1 FSM dijagram stanja za obje strane protokola . . . . .  | 6         |
| 2.1.2 FSM model dizajniran u UPPAAL-u . . . . .   | 7         |
| 2.2 MSC dijagrami za sve slučajeve upotrebe . . . . .   | 11        |
| 2.2.1 MSC dijagram - registracija korisnika . . . . .   | 11        |
| 2.2.2 MSC dijagram - ažuriranje lokacijskog registra prilikom promjene<br>IP adrese i/ili TCP porta . . . . . | 12        |
| 2.2.3 MSC dijagram - izlistavanje koordinata drugih dostupnih korisnika .                                     | 12        |
| 2.2.4 MSC dijagram - izlistavanje koordinata posljednje ažuriranih korisnika                                  | 13        |
| 2.2.5 MSC dijagram - slanje/prijem poruka svim korisnicima na određenoj<br>geografskoj lokaciji . . . . .     | 14        |
| 2.3 SDL dijagrami za svaku stranu protokola . . . . .   | 14        |
| 2.3.1 SDL dijagram na serverskoj strani protokola . . . . .   | 15        |
| 2.3.2 SDL dijagram na klijentskoj strani protokola . . . . .  | 16        |
| <b>3 Dijagrami klasa</b>  | <b>17</b> |
| 3.1 Klasa server . . . . .  | 17        |
| 3.1.1 Stanja klase server . . . . .   | 17        |
| 3.1.2 Atributi klase server . . . . .   | 18        |
| 3.1.3 Metode klase server . . . . .   | 19        |

|       |                                  |    |
|-------|----------------------------------|----|
| 3.2   | Klasa klijent . . . . .          | 19 |
| 3.2.1 | Stanja klase klijent . . . . .   | 19 |
| 3.2.2 | Atributi klase klijent . . . . . | 20 |
| 3.2.3 | Metode klase klijent . . . . .   | 21 |

---

# Popis slika

|      |   |    |
|------|---|----|
| 2.1  | FSM dijagram stanja na strani klijenta . . . . .  | 7  |
| 2.2  | FSM dijagram stanja na stani servera . . . . .  | 7  |
| 2.3  | FSM model klijenta . . . . .  | 8  |
| 2.4  | FSM model servera . . . . .   | 9  |
| 2.5  | Simulacija FSM modlela . . . . .  | 9  |
| 2.6  | Verifikacija FSM modela . . . . .   | 10 |
| 2.7  | MSC dijagram - registracija korisnika . . . . .   | 11 |
| 2.8  | MSC dijagram - ažuriranje lokacijskog registra . . . . .  | 12 |
| 2.9  | MSC dijagram - izlistavanje koordinata drugih dostupnih korisnika u slučaju da: postoje drugi korisnici u sistemu (lijevo); ne postoje drugi korisnici u sistemu (desno) . . . . .                                | 13 |
| 2.10 | MSC dijagram - izlistavanje koordinata dostupnih ažuriranih korisnika u slučaju da: postoje ažurirani korisnici u sistemu (lijevo); ne postoje ažurirani korisnici u sistemu (desno) . . . . .                    | 13 |
| 2.11 | MSC dijagram - slanje/prijem poruka svim korisnicima na određenoj geografskoj lokaciji u slučaju da: postoje korisnici na traženoj lokaciji (lijevo); ne postoje korisnici na traženoj lokaciji (desno) . . . . . | 14 |
| 2.12 | SDL dijagram - server . . . . .   | 15 |
| 2.13 | SDL dijagram - klijent . . . . .  | 16 |
| 3.1  | Detaljni dijagram klasa . . . . .   | 17 |

---

# 1. Uvod

## 1.1. Svrha SDD dokumenta

SDD (engl. *Software Design Description*) predstavlja pisani izvještaj o *software-u* koji opisuje njegovu sveopštu arhitekturu. Ovaj dokument opisuje dizajn protokola za geografsko lociranje korisnika, koji je prethodno predstavljen SRS dokumentom. Opisan je način implementacije protokola, a relacije između entiteta su prikazane odgovarajućim SDL, FSM i MSC dijagramima.

## 1.2. Opis projektnog zadatka

Cilj projektnog zadatka je dizajn i implementacija protokola za geografsko lociranje korisnika. Protokol se sastoji iz dva entiteta, korisničkog terminala i servera. Server treba podržati proizvoljan broj korisničkih terminala, a sama signalizacija se treba odvijati isključivo preko servera. Dakle, *peer-to-peer* konekcija između korisničkih terminala nije dozvoljena. Protokol treba omogućiti signalizaciju za:

- **registraciju korisničkog terminala na server uz pomoć URI-a (jedinstvenog alfanumeričkog identifikatora)** - Korisnik se putem terminala na server registruje pomoću svog jedinstvenog identifikatora - URI-a. Ukoliko je registracija uspješno izvršena, korisnik će biti u mogućnosti dobiti informaciju o svojoj lokaciji, o lokaciji/statusu drugih korisnika od servera. U slučaju neuspješne registracije, korisnik se neće moći povezati na server.

- **vođenje lokacijskog registra na serveru:**
  - **ažuriranje lokacijskog registra prilikom promjene IP adrese i/ili porta TCP servera korisničkog terminala** - Vršiti se ažuriranje lokacijskog registra svaki put kada se promijeni IP adresa i/ili TCP port servera korisničkog terminala.
  - **prevođenje URI-a u IP adresu i port TCP servera na strani korisničkog terminala** - URI svakog korisnika se nalazi u lokacijskom registru, zajedno sa IP adresom i portom.
- **automatsko ažuriranje statusa korisnika prema obrascu:**
  - **ako korisnik nije prijavljen na server, tada je u statusu Nedostupan**  
- Ukoliko korisnik nije registrovan na server, odn. ukoliko korisnikov URI se ne nalazi u lokacijskom registru, korisnik je u statusu Nedostupan.
  - **ako je korisnik prijavljen na server i nema uspostavljen poziv, tada je u statusu Dostupan**
  - **kako bi održao stanje dostupnosti, korisnik se periodički prijavljuje na server sa ažuriranim informacijama o geografskoj lokaciji (GPS koordinate)** - Korisnik se periodički prijavljuje na server da bi održao status Dostupan i to slanjem GPS koordinata. Ukoliko server ne primi ažuriranu informaciju o geografskoj lokaciji korisnika nakon nekog vremena, korisnik prelazi u status Nedostupan.
- **izlistavanje koordinata drugih dostupnih korisnika sistema na korisničkom terminalu** - Server, nakon provjere statusnog registra, šalje korisniku koordinate svih drugih korisnika sa statusom Dostupan u tom trenutku.
- **izlistavanje posljednjih ažuriranih dostupnih korisnika sistema na korisničkom terminalu** - Server šalje korisniku listu svih korisnika sistema na korisničkom terminalu koji imaju status Dostupan (nakon provjere statusnog registra) u tom trenutku.
- **slanje/prijem poruka svim korisnicima na određenoj geografskoj koordinati za sastanak putem korisničkog terminala** - Server provjerava da li na određenoj geografskoj koordinati postoje dostupni korisnici. Ukoliko da, server dostupnim korisnicima šalje poruku za sastanak. Ukoliko ne postoje dostupni korisnici na određenoj lokaciji, na korisničkom terminalu se ispisuje prikladna poruka.

Realizacija i analiza svakog od navedenih zadataka će biti urađena u nastavku ovog dokumenta.

## 1.3. Definicije, akronimi i skraćenice

### 1.3.1. Definicije

| Pojam               | Definicija  |
|---------------------|---|
| Korisnični terminal | Entitet koji se sastoji iz dva dijela: klijent i aplikacija   |
| Server              | Entitet koji pruža različite funkcionalnosti klijentima   |
| Aplikacija          | Program koji klijentu omogućava interfejs pomoću kojeg pristupa serveru, a instaliran je na klijentskom uređaju                               |
| Klijent             | Korisnik usluge, aplikacije koja mu omogućava geografsko lociranje  |
| URI                 | Niz znakova koji se koristi za identifikaciju jednog korisnika, a čine ga IP adresa, port i drugi identifikatori.                             |
| Lokacijski registar | Entitet na strani servera u kojem se čuvaju podaci o korisnicima (URI, IP adresa, port)   |
| GPS koordinate      | Jedinstveni identifikator tačnog geografskog položaja na zemlji izražen alfanumeričkim znakovima, a sastoji se od geografske širine i dužine. |

### 1.3.2. Akronimi i skraćenice

| Akronim    | Značenje                               |
|------------|--|
| <b>SRS</b> | Software Requirements Specifications   |
| <b>SDD</b> | Software Design Descriptions           |
| <b>URI</b> | Uniform Resource Identifier            |
| <b>UML</b> | Universal Modeling Language            |
| <b>IP</b>  | Internet Protocol                      |
| <b>TCP</b> | Transport Control Protocol             |
| <b>GPS</b> | Global Positioning System              |
| <b>FSM</b> | Finite State Machine                   |
| <b>SDL</b> | Specification and Description Language |
| <b>MSC</b> | Message Sequence Chart                 |

## 1.4. Reference

- [1] IEEE. *IEEE Std 1016-2009 - IEEE Standard for Information Technology–Systems Design–Software Design Descriptions* IEEE Computer Society, 2009.
- [2] M. Mehić. Predavanja iz predmeta Softverski dizajn protokola, Elektrotehnički fakultet Sarajevo, 2020.
- [3] UML Sequence Diagram Online Tool «SequenceDiagram» Dostupno na: <https://sequencediagram.org/>
- [4] Diagram Software «Visual Paradigm Online» Dostupno na: <https://online.visual-paradigm.com/>
- [5] Softverski alat «SmartDraw» Dostupno na: <https://www.smartdraw.com/>
- [6] Behrmann, G., David, A., Larsen, K. G. (2006). A tutorial on Uppaal 4.0. Department of computer science, Aalborg University. Dostupno na: <https://www.it.uu.se/research/group/darts/papers/texts/new-tutorial.pdf>

## 1.5. Pregled dokumenta

U ovom dokumentu je prikazan opis dizajna i implementacija protokola za geografsko lociranje korisnika, pri čemu su korišteni MSC dijagrami za sve slučajeve upotrebe, SDL dijagrami, kao i dijagrami stanja, odnosno FSM dijagrami za svaku stranu protokola. Također, dat je i prikaz i opis FSM modela koji je dizajniran korištenjem UPPAAL-a.

Za izradu MSC dijagrama korišten je alat «SequenceDiagram» [3], alat «Visual Paradigm Online» [4] je korišten za izradu FSM dijagrama, dok je alat «SmartDraw» [5] korišten pri izradi SDL dijagrama.



---

## 2. Opis rada protokola

Poglavlje objašnjava rad protokola za geografsko lociranje korisnika. Protokol se izvršava na klijentskoj i serverskoj strani pri čemu se može naći u različitim stanjima. Poruke između servera i klijenta se razmjenjuju pomoću TCP transportnog protokola. Iako protokol podržava više korisnika aplikacije, komunikacija se odvija isključivo u pravcu korisnik-server i obratno – dakle, *peer-to-peer* komunikacija među korisnicima nije podržana.

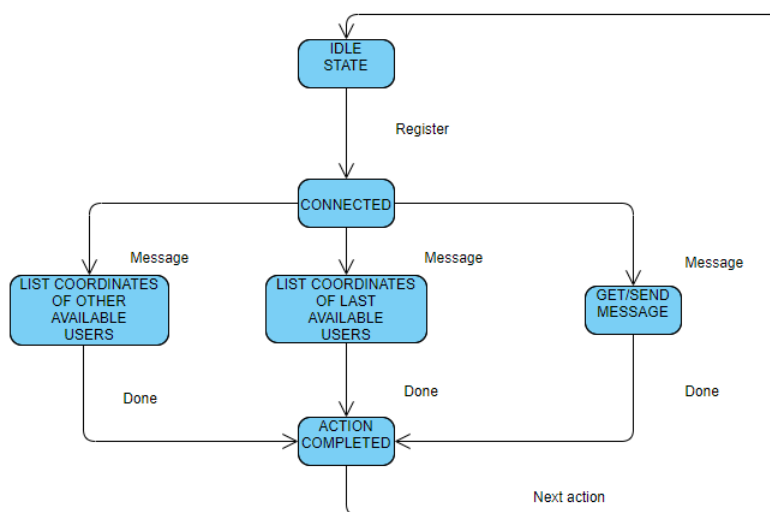
### 2.1. FSM dijagrami stanja

FSM (engl. *Final State Machine*) dijagram detaljno prikazuje stanja i tranzicije između pojedinih stanja te poruke koje se šalju i primaju. Definisan je listom inicijalnih stanja, kao i događaja koji trigeruju tranzicije u druga stanja. U našem slučaju potrebno je realizirati FSM dijagrame stanja za svaku stranu komunikacije prilikom geografskog lociranja korisnika. Server pri tome vrši registraciju, ažuriranje, lociranje i drugo, dok se klijent prijavljuje na server da bi saznao svoju lokaciju. Za obje strane je potrebno napraviti FSM dijagram stanja.

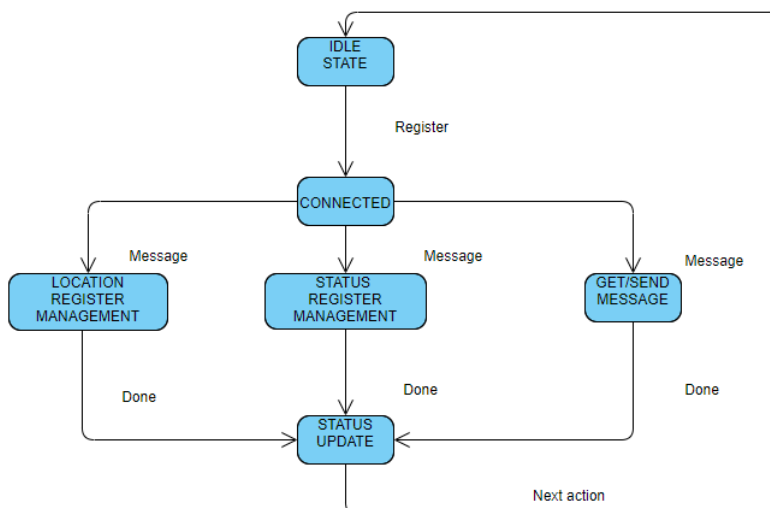
#### 2.1.1. FSM dijagram stanja za obje strane protokola

FSM dijagram server i klijenta se sastoji od 6 stanja. Kada se govori o klijentu on iz inicijalnog stanja prelazi u stanje CONNECTED. Ova tranzicija se dešava kada se on registruje na server sa ispravnim URI identifikatorom. Nakon toga on može preći u neko od tri prikazana stanja u zavisnosti od toga da li traži listu svih aktivnih korisnika, listu aktivnih korisnika na specifičnoj lokaciji ili želi da pošalje poruku drugom korisniku preko servera. Završetkom neke od ovih akcija klijent se vraća u stanje IDLE. Navedeno je prikazano na slici 2.1.

Server također počinje iz IDLE stanja. Nakon što se korisnik konektuje na server, server može imati jedno od tri prikazana stanja. Može ažurirati lokacijski registar, ažurirati statusni registar ili slati poruke klijentu (ovdje se podrazumijeva slanje poruka, geografskih lokacija dostupnih korisnika ili posljednjih ažuriranih korisnika). Nakon odrađene akcije server se vraća u stanje IDLE. Navedeno je prikazano na slici 2.2.



Slika 2.1: FSM dijagram stanja na strani klijenta

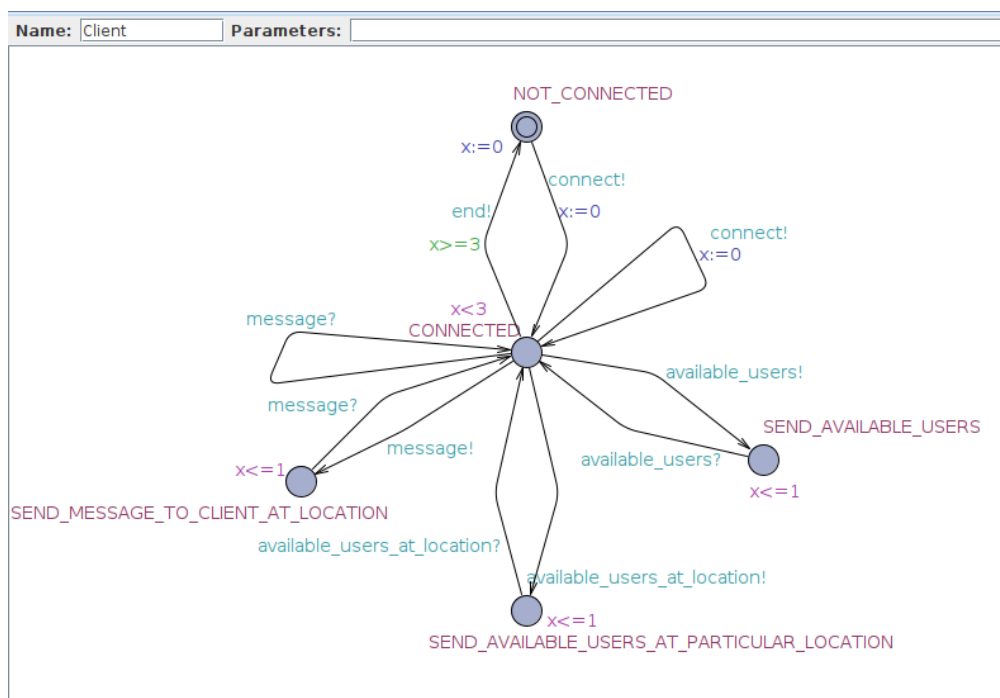


Slika 2.2: FSM dijagram stanja na stani servera

### 2.1.2. FSM model dizajniran u UPPAAL-u

Za dizajniranje FSM modela protokola za geografsko lociranje koirisnika korišten je UPPAAL. UPPAAL je alat koji služi za modeliranje, verifikaciju i validaciju sistema u realnom vremenu, a baziran je na teoriji vremenskih automata. Za potrebe izrade FSM modela i razumijevanje UPPAAL alata korišten je tutorijal dostupan [6].

Na slici 2.3 prikazan je FSM model klijenta u UPPAAL-u. Klijent pri tome može imati pet stanja, i to: NOT\_CONNECTED, CONNECTED, SEND\_AVAILABLE\_USERS, SEND\_AVAILABLE\_USERS\_AT\_PARTICULAR\_LOCATION i SEND\_MESSAGE\_TO\_CLIENT\_AT\_LOCATION.

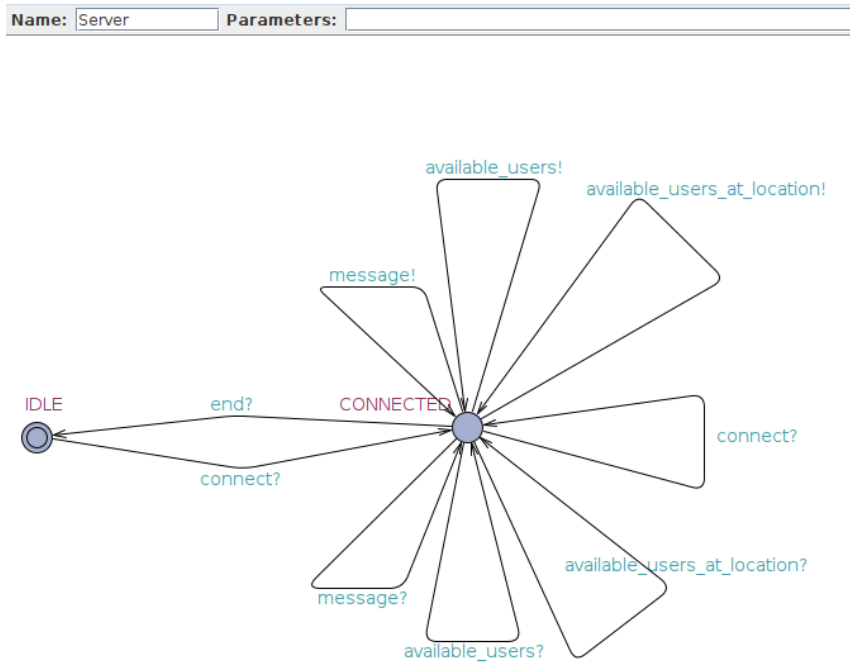


Slika 2.3: FSM model klijenta

Ukoliko se klijent nalazi u stanju `NOT_CONNECTED`, jedino stanje u koje on može preći je `CONNECTED`. Pri tome se pokreće tajmer koji je na početku jednak nuli. Prelaskom u stanje `CONNECTED`, klijent ima mogućnost prelaska u sva ostala stanja ili može ostati u navedenom stanju. Ukoliko dođe do isteka tajmera, to vrijeme je u UPPAAL-u definisano kao  $x \geq 3$ , klijent se vraća u stanje `NOT_CONNECTED`. U stanju `CONNECTED` klijent će ostati ako ponovo pošalje zahtjev serveru za registracijom, pri čemu se njegov tajmer sada vraća na vrijednost 0. Navedeno predstavlja slučaj upotrebe periodičnog slanja lokacije serveru u svrhu zadržavanja statusa "Dostupan". Ukoliko klijent u navedenom stanju primi poruku unutar 3 minute, također ostaje u stanju `CONNECTED`.

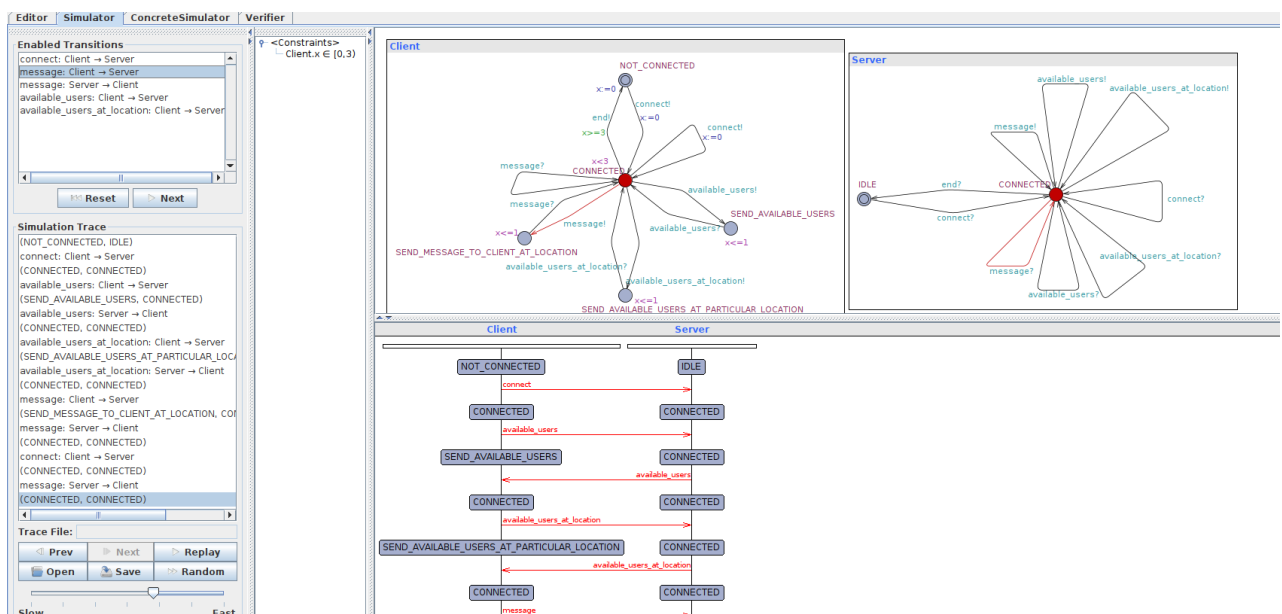
Kada klijent od servera zahtijeva slanje poruke drugim korisnicima, listu dostupnih korisnika ili listu dostupnih korisnika na određenoj lokaciji, tada prelazi u jedno od preostala 3 stanja, i to `SEND_AVAILABLE_USERS`, `SEND_AVAILABLE_USERS_AT_PARTICULAR_LOCATION` i `SEND_MESSAGE_TO_CLIENT_AT_LOCATION`, respektivno. U tom stanju ostaje dok je vrijednost tajmera  $x \leq 1$ . Kada tajmer istekne on se vraća u stanje `CONNECTED`.

Na slici 2.4 prikazan je FSM model servera u UPPAAL-u. Server može biti samo u dva stanja. To su `IDLE` stanje i `CONNECTED` stanje. Ukoliko se server nalazi u stanju `IDLE`, on prilikom povezivanja klijenta prelazi u stanje `CONNECTED`. Iz ovog stanja se može vratiti u `IDLE` stanje ukoliko dođe do kraja komunikacije, koju prekida klijent. U suprotnom ostaje u stanju `CONNECTED` radi ispunjenja klijentovih zahtijeva koji su navedeni na samom modelu.

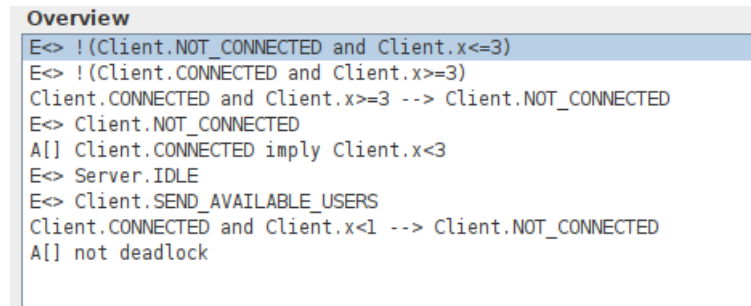


Slika 2.4: FSM model servera

Slika 2.5 daje prikaz simuliranog modela u UPPAAL-u. Nakon početnog pokretanja simulacije, u gornjem lijevom dijelu prozora prikazane su moguće tranzicije iz inicijalnog stanja. Odabirom željene tranzicije ili klikom na *Next*, koji vrši slučajni odabir naredne tranzicije, i klijent i server prelaze u odgovarajuće stanje; stanje definisano odabranom tranzicijom. Tranzicije u različita stanja klijenta i servera se mogu uporedno pratiti u prozoru *Simulation Trace* koji prikazuje prethodna stanja i tranzicije, kao i na dijagramu sekvenci. Također, kompletni modeli klijenta i servera, kao i njihova trenutna stanja su vidljivi tokom simulacije.



Slika 2.5: Simulacija FSM modela



```

Overview
E<> !(Client.NOT_CONNECTED and Client.x<=3)
E<> !(Client.CONNECTED and Client.x>=3)
Client.CONNECTED and Client.x>=3 --> Client.NOT_CONNECTED
E<> Client.NOT_CONNECTED
A[] Client.CONNECTED imply Client.x<3
E<> Server.IDLE
E<> Client.SEND_AVAILABLE_USERS
Client.CONNECTED and Client.x<1 --> Client.NOT_CONNECTED
A[] not deadlock

```

Slika 2.6: Verifikacija FSM modela

Na slici 2.6 je dat prikaz verifikatora koji je korišten za verifikaciju prethodnih FSM modela. Kao što se da uočiti, fokus pri verifikaciji je bio na modelu klijenta zbog kompleksnosti njegovog modela u odnosu na model servera. U nastavku je objašnjeno značenje svih upita unutar verifikatora.

$E<> !(Client.NOT\_CONNECTED \text{ and } Client.x \leq 3)$  upit provjerava da li je nemoguća tranzicija u stanje NOT\_CONNECTED pri vrijednosti tajmera  $x \leq 3$ . U našem slučaju, na navedeni upit treba dobiti potvrđan odgovor.

$E<> !(Client.CONNECTED \text{ and } Client.x \leq 3)$  upit provjerava da li je nemoguća tranzicija, odnosno dalje zadržavanje u stanju CONNECTED u slučaju da je vrijednost tajmera  $x \leq 3$ . Na ovaj upit treba dobiti potvrđan odgovor također.

$Client.CONNECTED \text{ and } Client.x \geq 3 \rightarrow Client.NOT\_CONNECTED$  upit provjerava da li se dešava tranzicija iz stanja CONNECTED u stanje NOT\_CONNECTED ukoliko je vrijednost tajmera  $x \leq 3$ . Kako je tranzicija moguća, a nužna je, odgovor na upit treba biti potvrđan.

Upiti  $E<> Client.NOT\_CONNECTED$ ,  $E<> Client.SEND\_AVAILABLE\_USERS$  i  $E<> Server.IDLE$  provjeravaju da li će se klijent naći u stanju NOT\_CONNECTED i SEND\_AVAILABLE\_USERS, a server u stanju IDLE, respektivno. Na sva tri upita odgovor treba biti potvrđan, što je provjereno i simulacijom.

$A[] Client.CONNECTED \text{ imply } Client.x < 3$  upit ima za cilj provjeru da li vrijedi sljedeće: ukoliko je klijent u stanju CONNECTED, globalno, za sva stanja, vrijedi da tajmer ima vrijednost tajmera  $x < 3$ . Odgovor treba biti potvrđan ili 'da je zadovoljena tvrdnja'.

$Client.CONNECTED \text{ and } Client.x < 1 \rightarrow Client.NOT\_CONNECTED$  upit provjerava da li je moguća tranzicija iz stanja CONNECTED u stanje NOT\_CONNECTED ukoliko za tajmer vrijedi  $x < 1$ . Odgovor na ovaj upit treba biti 'ne zadovoljava' odnosno treba biti negativan jer navedena tranzicija je moguća samo u slučaju da tajmer ima vrijednost  $x \geq 3$ , kao što je prethodno provjereno (trećim upitom).

Konačno, upit  $A[] \text{ not deadlock}$  provjerava da li će doći do prelaska u stanje *deadlock*, odnosno u stanje iz kojeg nije moguće obaviti bilo kakvu tranziciju nakon prelaska u njega. Očekivani odgovor na ovaj upit je potvrđan, jer sam upit provjerava nedolazak u navedeno stanje.

## 2.2. MSC dijagrami za sve slučajeve upotrebe

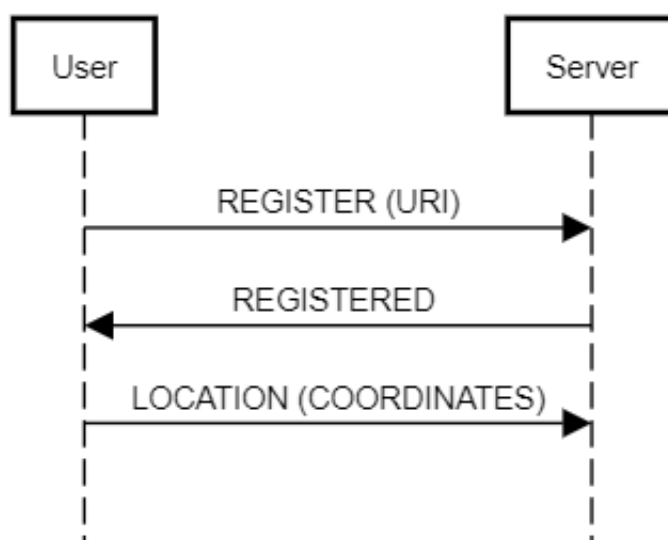
MSC (engl. *Message Sequence Chart*) je dijagram koji pruža specifikaciju i opis komunikacijskog ponašanja sistemskih komponenti i njihovog okruženja u smislu razmjene poruka. MSC dijagramom je ponašanje u komunikaciji prikazano na veoma intuitivan način, jer je u pitanju grafički prikaz. Kombinovan sa drugim dijagramima, kao što je to slučaj u ovom dokumentu, koristi se za podršku metodologiji, dizajnu, simulaciji, testiranju i dokumentaciji.

U nastavku će biti prikazani MSC dijagrami za sve slučajeve upotrebe, i to:

- Registracija na server pomoću URI-a
- Ažuriranje lokacijskog registra prilikom promjene IP adrese i/ili porta TCP servera korisničkog terminala
- Izlistavanje koordinata drugih dostupnih korisnika
- Izlistavanje koordinata posljednje ažuriranih dostupnih korisnika
- Slanje/prijem poruka svim korisnicima na određenoj geografskoj lokaciji za sastanak

### 2.2.1. MSC dijagram - registracija korisnika

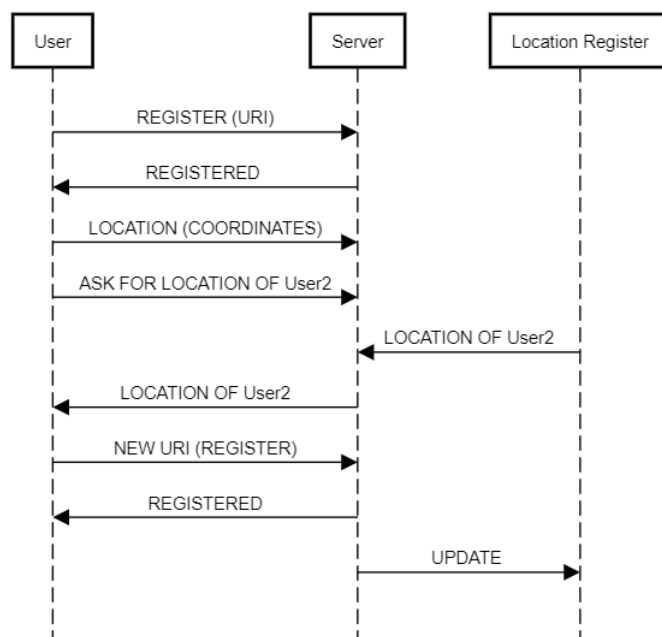
Korisnik se prijavljuje na server na način da šalje zahtjev za registraciju. Na serverskoj strani se iz korisničkog URI-a ekstraktuju IP adresa i TCP port, nakon čega se navedene vrijednosti povezuju sa jedinstvenim identifikatorom korisnika. Nakon uspješne registracije, korisnik se obavještava o istoj. Osim toga, korisnik nakon uspješno izvršene registracije periodično šalje svoje GPS koordinate kako bi održao stanje dostupnosti.



Slika 2.7: MSC dijagram - registracija korisnika

### 2.2.2. MSC dijagram - ažuriranje lokacijskog registra prilikom promjene IP adrese i/ili TCP porta

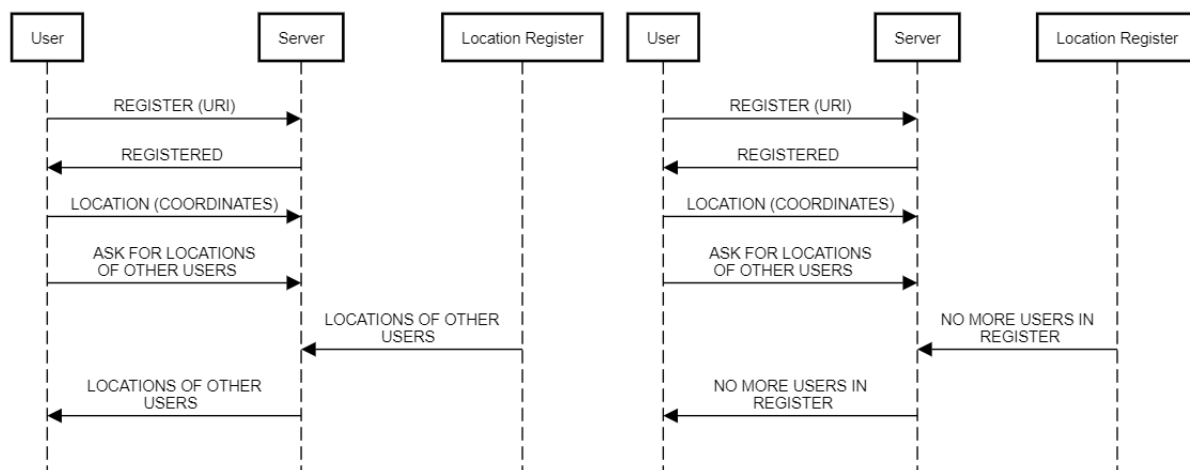
Kada korisnik u toku korištenja usluge, promijeni IP adresu i/ili TCP port, o tome obavještava server koji prevodi korisnikov novi URI u adresu i port te povezuje nove vrijednosti sa jedinstvenim identifikatorom korisnika.



Slika 2.8: MSC dijagram - ažuriranje lokacijskog registra

### 2.2.3. MSC dijagram - izlistavanje koordinata drugih dostupnih korisnika

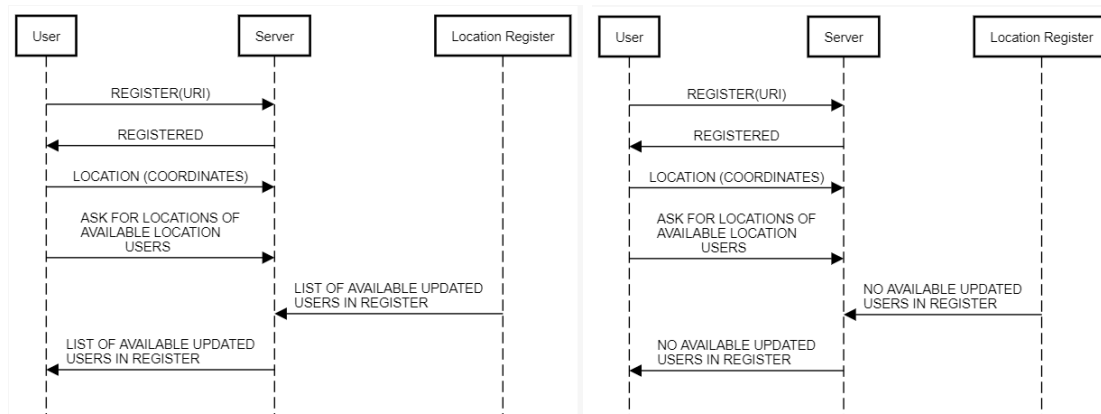
Korisnik putem upotrebe TCP-a šalje zahtjev serveru za izlistavanje koordinata drugih dostupnih korisnika. Server u tom slučaju, uzimajući podatke iz lokacijskog registra upotrebom TCP-a, vraća korisniku listu koordinata dostupnih korisnika, ukoliko takvih ima u datom trenutku. Proces počinje registracijom korisnika na server, nakon čega se traži lista koordinata ostalih dostupnih korisnika. Ukoliko nema drugih korisnika u sistemu, poruka se ispisuje na ekranu korisnika, u protivnom se ispisuje lista koordinata.



Slika 2.9: MSC dijagram - izlistavanje koordinata drugih dostupnih korisnika u slučaju da: postoje drugi korisnici u sistemu (lijevo); ne postoje drugi korisnici u sistemu (desno)

#### 2.2.4. MSC dijagram - izlistavanje koordinata posljednje ažuriranih korisnika

Proces počinje registracijom korisnika na server. Korisnik putem TCP-a šalje zahtjev serveru za izlistavanje koordinata posljednje ažuriranih korisnika koji su u statusu "Dostupan". Poruke o ažuriranim korisnicima u stanju "Nedostupan" se ne izlistavaju u terminalu. Ukoliko dostupni korisnici ne postoje poruka se ispisuje na ekranu, dok u suprotnom se ispisuje lista dostupnih posljednje ažuriranih korisnika.

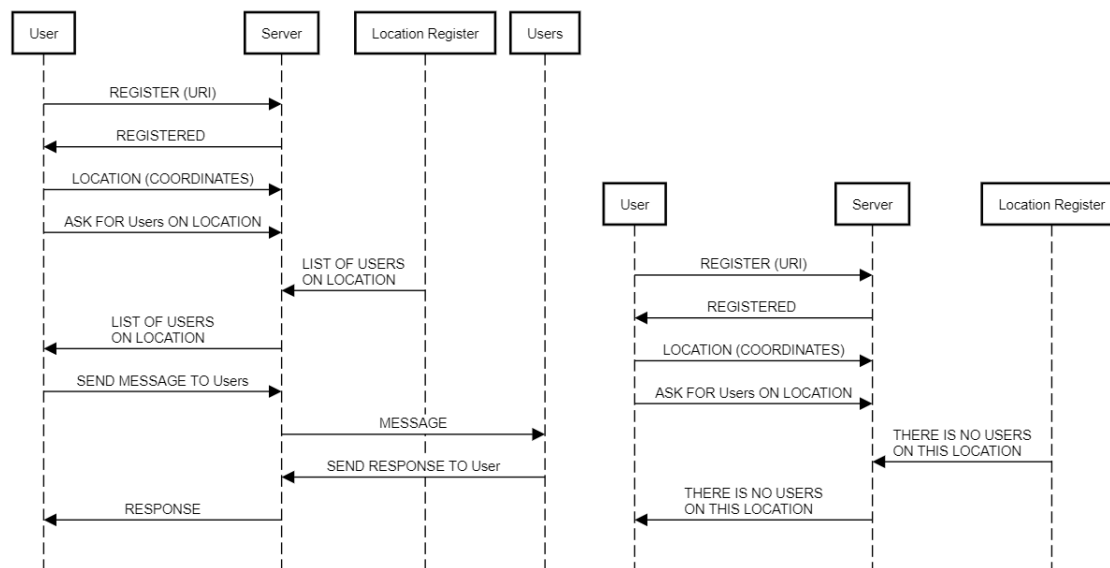


Slika 2.10: MSC dijagram - izlistavanje koordinata dostupnih ažuriranih korisnika u slučaju da: postoje ažurirani korisnici u sistemu (lijevo); ne postoje ažurirani korisnici u sistemu (desno)



### 2.2.5. MSC dijagram - slanje/prijem poruka svim korisnicima na određenoj geografskoj lokaciji

Proces započinje registracijom korisnika na server, nakon čega korisnik serveru šalje zahtjev za slanje poruke za sastanak svim korisnicima na specifičnoj geografskoj lokaciji. Server nakon toga provjerava listu dostupnih korisnika na navedenoj lokaciji. U slučaju da ne postoje drugi dostupni korisnici na traženoj lokaciji, na korisničkom terminalu se ispisuje prikladna poruka. U suprotnom, server šalje svima poruku/obavijest o sastanku.



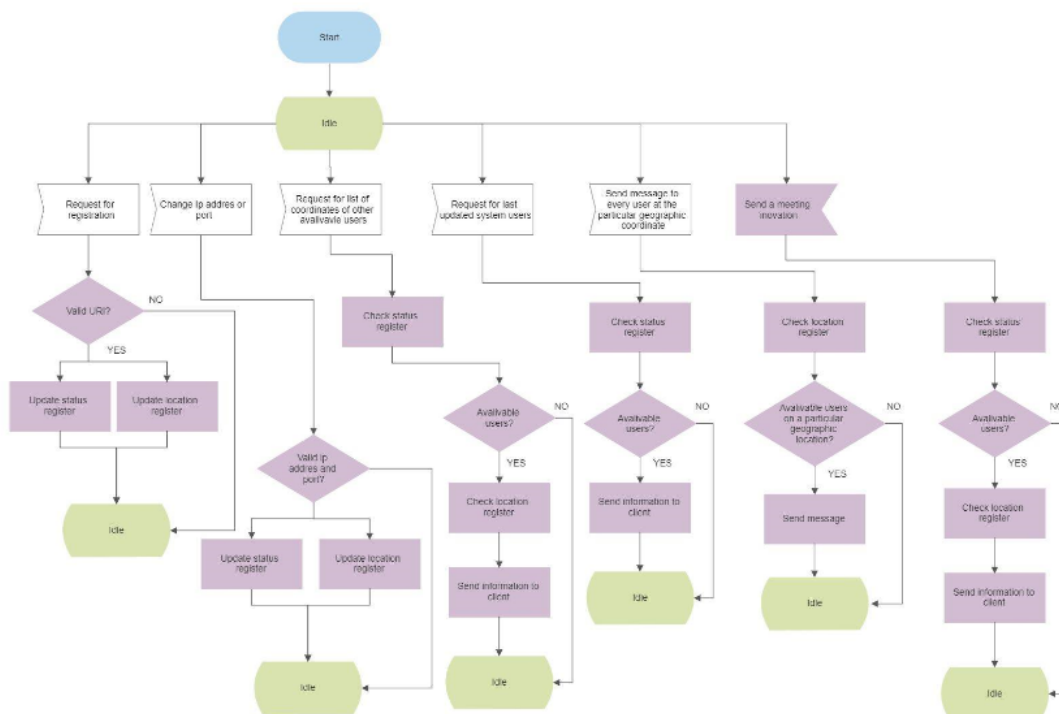
Slika 2.11: MSC dijagram - slanje/prijem poruka svim korisnicima na određenoj geografskoj lokaciji u slučaju da: postoje korisnici na traženoj lokaciji (lijevo); ne postoje korisnici na traženoj lokaciji (desno)

## 2.3. SDL dijagrami za svaku stranu protokola

SDL (engl. *Specification and Description Language*) dijagram ilustrira proces specifikacije i opisa automata sa listom stanja i tranzicija. Sastoji se od tri dijela: definicija sistema, blokovi i procesi.

Kako je potrebno realizirati SDL dijagrame za svaku stranu protokola, u nastavku će biti prikazan SDL dijagrami za klijentsku i serversku stranu protokola.

### 2.3.1. SDL diagram na serverskoj strani protokola



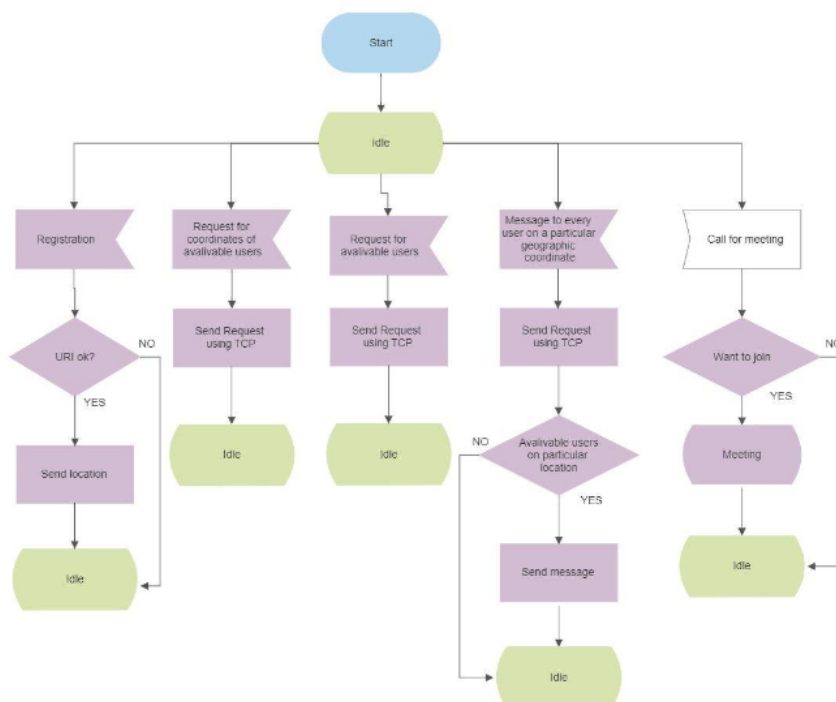
Slika 2.12: SDL dijagram - server

Na slici 2.12 prikazan je SDL dijagram na serverskoj strani protokola. Server je prvobitno u stanju IDLE, sve dok od korisnika ne dobije zahtjev za izvršavanje jedne od navedenih funkcija. Ukoliko korisnik traži zahtjev za registraciju, provjerava se validnost njegovog URI-a. Za validan URI, server registruje korisnika i prelazi u stanje IDLE. U suprotnom odmah prelazi u IDLE stanje. Ukoliko dođe do promjene IP adrese korisnika i/ili njegovog TCP porta vrši se provjeravanje validnosti ovih podataka. Ako su podaci validni, ažuriraju se lokacijski i statusni registar, te server prelazi u stanje IDLE, u suprotnom odmah prelazi u navedeno stanje.

Ako korisnik od servera zahtijeva listu dostupnih korisnika ili želi da pošalje zahtjev za poruku, server provjerava statusni registar. Ukoliko ne postoje dostupni korisnici server prelazi u IDLE stanje. Ipak, ukoliko postoje provjerava se lokacijski registar, informacija se šalje korisniku i prelazi u IDLE stanje. Ukoliko korisnik od servera zahtijeva listu posljednje ažuriranih korisnika provjerava se statusni registar. Ako su ti korisnici dostupni, server šalje informaciju korisniku i prelazi u stanje IDLE. U suprotnom odmah prelazi u IDLE stanje. Konačno, ukoliko korisnik želi da pošalje poruku ostalim korisnicima na određenoj lokaciji server vrši provjeru lokacijskog registra. Ukoliko na tim lokacijama postoje dostupni korisnici, šalje im se poruka i server prelazi u stanje IDLE. Ukoliko ih nema odmah prelazi u IDLE stanje.

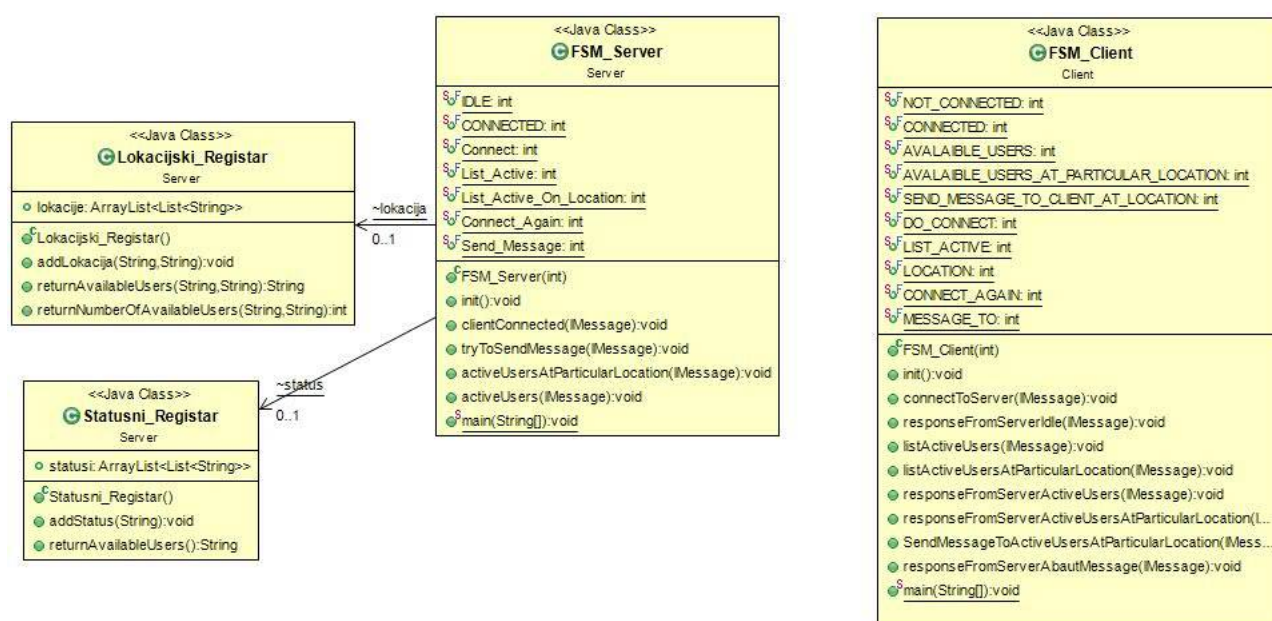
### 2.3.2. SDL dijagram na klijentskoj strani protokola

Na slici 2.13 prikazan je SDL dijagram na klijentskoj strani protokla. Klijent je na početku u IDLE stanju. Ukoliko želi da se registruje na server, šalje svoj URI. Ako URI nije validan korisnik se vraća u stanje IDLE. U suprotnom, šalje svoju lokaciju i prelazi u IDLE stanje. Ako želi poslati zahtjev za listu dostupnih korisnika šalje zahtjev koristeći TCP i prelazi u stanje IDLE. Ukoliko želi poslati poruku svakom korisniku na određenoj geografskoj lokaciji opet šalje zahtjev putem TCP protokola. Ukoliko ne postoje korisnici na toj lokaciji prelazi u IDLE stanje, u suprotnom im šalje poruku i prelazi u navedeno stanje.



Slika 2.13: SDL dijagram - klijent

## 3. Dijagrami klasa



Slika 3.1: Detaljni dijagram klasa

### 3.1. Klasa server

#### 3.1.1. Stanja klase server

Server je realizovan tako da može imati samo dva stanja: IDLE i CONNECTED. Prije nego što se aktivira server je u IDLE stanju. Kada primi poruku *Connect* poziva metodu *clientConnected* i tada prelazi u stanje CONNECTED. U ovom stanju ostaje prilikom svih akcija koje od njega zahtijeva korisnik, osim kada se pozove metoda *pressAnykey* koja znači kraj komunikacije i prelazak u stanje IDLE.

| Početno stanje | Primljena poruka        | Metoda koja se poziva           | Stanje u koje prelazi |
|----------------|-------------------------|---------------------------------|-----------------------|
| IDLE           | Connect                 | clientConnected                 | CONNECTED             |
| CONNECTED      | List_Active             | activeUsers                     | CONNECTED             |
| CONNECTED      | List_Active_On_Location | activeUsersAtParticularLocation | CONNECTED             |
| CONNECTED      | Connect_Again           | clientConnected                 | CONNECTED             |
| CONNECTED      | Send_Message            | tryToSendMessage                | CONNECTED             |
| CONNECTED      |                         | pressAnykey                     | IDLE                  |

Tablica 3.1: Stanja klase server

### 3.1.2. Atributi klase server

U programskom okruženju *Java* server je realizovan tako da može imati vrijednosti atributa 0,1,2 ili 3, kao što je prikazano u tabeli 3.2. Vrijednost atributa 0 je za početno stanje prije konektovanja bilo kojeg korisnika na server, kao i interne poruke pomoću kojih se klijent pokušava povezati na server ili traži listu aktivnih korisnika. Vrijednost atributa 1 je za stanja u kojem se server nalazi kada je barem jedan klijent povezan na njega ili internu poruku pomoću koje klijent traži aktivne korisnike na preciziranoj lokaciji. Vrijednost atributa 2 je za ponovno povezivanje klijenta na server, te vrijednost 3 za slanje poruke preko servera ostalim korisnicima na određenoj geografskoj lokaciji.

| Modifikator pristupa, naziv i vrijednost atributa          | Opis   |
|--|--|
| public static final int <b>IDLE</b> = 0                    | Početno stanje servera, prije nego se bilo koji klijent konektovao na server                                   |
| public static final int <b>CONNECTED</b> = 1               | Stanje u kojem se server nalazi kada je bar jedan klijent konektovan na njega                                  |
| public static final int <b>Connect</b> = 0                 | Interna poruka pomoću koje se klijent pokušava povezati na server  |
| public static final int <b>List_Active</b> = 0             | Interna poruka pomoću koje klijent traži listu aktivnih korisnika na serveru                                   |
| public static final int <b>List_Active_On_Location</b> = 1 | Interna poruka pomoću koje klijent traži listu aktivnih korisnika na preciziranoj lokaciji                     |
| public static final int <b>CONNECT_AGAIN</b> = 2           | Interna poruka pomoću koje se klijent ponovo konektuje na server   |
| public static final int <b>Send_Message</b> = 3            | Interna poruka pomoću koje klijent pokušava da pošalje poruku korisnicima na preciziranoj geografskoj lokaciji |

Tablica 3.2: Atributi klase server

### 3.1.3. Metode klase server

Svaka od metoda klase server je realizovana kao *public void*, te svaka od njih prima vrijednost *IMessage msg* kao što je prikazano u tabeli 3.3. Metoda *clientConnected* se poziva kada se klijent želi spojiti na server. Kada server primi zahtjev od klijenta za listu aktivnih korisnika poziva se metoda *activeUsers*, za listu aktivnih korisnika na određenoj geografskoj lokaciji poziva se metoda *activeUsersAtParticularLocation*, dok se za slanje poruka određenim korisnicima preko servera poziva metoda *tryToSendMessage*.

| Modifikator pristupa, povratni tip metode | Signature metode  | Opis metode   |
|---|---|---|
| <b>public void</b>                        | <i>clientConnected</i><br>( <i>IMessage msg</i> )                 | Metoda koja se poziva kada se klijent spoji na server   |
| <b>public void</b>                        | <i>activeUsers</i><br>( <i>IMessage msg</i> )                     | Metoda koja se poziva kada server primi zahtjev od klijenta za listu aktivnih korisnika   |
| <b>public void</b>                        | <i>activeUsersAtParticularLocation</i><br>( <i>IMessage msg</i> ) | Metoda koja se poziva kada server primi zahtjev od klijenta za listu aktivnih korisnika na preciziranoj geografskoj lokaciji      |
| <b>public void</b>                        | <i>tryToSendMessage</i><br>( <i>IMessage msg</i> )                | Metoda koja se poziva kada server primi zahtjev od klijenta za slanje poruke klijentima na tačno definisanoj geografskoj lokaciji |

Tablica 3.3: Metode klase server

## 3.2. Klasa klijent

### 3.2.1. Stanja klase klijent

Klijent je realizovan tako da može imati 5 stanja, ito: *NOT\_CONNECTED*, *CONNECTED*, *AVAILABLE\_USERS*, *AVAILABLE\_USERS\_AT\_PARTICULAR\_LOCATION*, *SEND\_MESSAGE\_TO\_CLIENT\_AT\_LOCATION*.

Ukoliko primi poruku za konektovanje klijent iz stanja *NOT\_CONNECTED* prelazi u stanje *CONNECTED*. U ovom stanju ostaje, svaki put kada šalje određenu poruku drugom korisniku ili zahtjev za lokacijom. Ukoliko je u jednom od sljedećih stanja: *AVAILABLE\_USERS*, *AVAILABLE\_USERS\_AT\_PARTICULAR\_LOCATION* ili *SEND\_MESSAGE\_TO\_CLIENT\_AT\_LOCATION* tada nakon obavljene akcije prelazi u stanje *CONNECTED*.

| Početno stanje                         | Primljena poruka | Metoda koja se poziva                             | Stanje u koje prelazi                  |
|--|------------------|---|--|
| NOT_CONNECTED                          | DO_CONNECT       | connectToServer                                   | CONNECTED                              |
| CONNECTED                              | Message 77       | responseFromServerIdle                            | CONNECTED                              |
| CONNECTED                              | CONNECT_AGAIN    | connectToServer                                   | CONNECTED                              |
| CONNECTED                              | LIST_ACTIVE      | listActiveUsers                                   | AVAILABLE_USERS                        |
| AVAILABLE_USERS                        | Message 78       | responseFromServerActiveUsers                     | CONNECTED                              |
| CONNECTED                              | LOCATION         | listActiveUsersAtParticularLocation               | AVAILABLE_USERS_AT_PARTICULAR_LOCATION |
| AVAILABLE_USERS_AT_PARTICULAR_LOCATION | Message 79       | responseFromServerActiveUsersAtParticularLocation | CONNECTED                              |
| CONNECTED                              | MESSAGE_TO       | SendMessageToActiveUsersAtParticularLocation      | SEND_MESSAGE_TO_CLIENT_AT_LOCATION     |
| SEND_MESSAGE_TO_CLIENT_AT_LOCATION     | Message 79       | responseFromServerAboutMessage                    | CONNECTED                              |

Tablica 3.4: Stanja klase klijent

### 3.2.2. Atributi klase klijent

Atributi klase klijent mogu imati vrijednosti 0,1,2,3 ili 4, kao što je prikazano u tabeli 3.5. Vrijednost atributa 0 opisuje početno stanje klijenta prije nego se konektuje na server, te interne poruke kojima se klijent konektuje na server ili traži listu aktivnih korisnika sa servera. Vrijednost atributa 1 opisuje stanje korisnika koji je povezan na server ili internu poruku pomoću koje klijent od servera traži listu korisnika na određenoj geografskoj lokaciji. Vrijednost atributa 2 opisuje stanje klijenta kada traži listu aktivnih korisnika od servera ili internu poruku kojom se ponovno povezuje na server. Za listu aktivnih korisnika na određenoj geografskoj lokaciji ili internu poruku pomoću koje se pokušava poslati poruka korisnicima na određenoj lokaciji koristi se vrijednost atributa 3. Konačno vrijednost atributa 4 opisuje stanje korisnika koji želi poslati poruku svim korisnicima na preciziranoj geografskoj lokaciji.

| Modifikator pristupa, naziv i vrijednost atributa                         | Opis   |
|---|--|
| public static final int <b>NOT_CONNECTED</b> = 0                          | Početno stanje klijenta, prije nego se konektuje na server   |
| public static final int <b>CONNECTED</b> = 1                              | Stanje klijenta kada je konektovan na server   |
| public static final int <b>AVAILABLE_USERS</b> = 2                        | Stanje klijenta kada traži od servera listu aktivnih korisnika   |
| public static final int <b>AVAILABLE_USERS_AT_PARTICULAR_LOCATION</b> = 3 | Stanje klijenta kada traži listu aktivnih korisnika na određenoj lokaciji                                      |
| public static final int <b>SEND_MESSAGE_TO_CLIENT_AT_LOCATION</b> = 4     | Stanje klijenta kada preko servera želi poslati poruku korisnicima na određenoj geografskoj lokaciji           |
| public static final int <b>DO_CONNECT</b> = 0                             | Interna poruka kojom se klijent konektuje na server  |
| public static final int <b>LIST_ACTIVE</b> = 0                            | Interna poruka pomoću koje klijent traži listu aktivnih korisnika na serveru                                   |
| public static final int <b>LOCATION</b> = 1                               | Interna poruka pomoću koje klijent traži listu aktivnih korisnika na preciziranoj lokaciji                     |
| public static final int <b>CONNECT_AGAIN</b> = 2                          | Interna poruka pomoću koje se klijent konektuje na server  |
| public static final int <b>MESSAGE_TO</b> = 3                             | Interna poruka pomoću koje klijent pokušava da pošalje poruku korisnicima na preciziranoj geografskoj lokaciji |

Tablica 3.5: Atributi klase klijent

### 3.2.3. Metode klase klijent

Svaka od metoda klase klijent je realizovana kao *public void*, te svaka od njih prima vrijednost *IMessage msg*. Metoda *connectToServer* se poziva za spajanje klijenta na server. Ukoliko klijent dobija odgovor od servera o uspješnosti komunikacije poziva se metoda *ResponsFromServerIdle*. Kada klijent od servera traži listu aktivnih korisnika ili korisnika na preciziranoj geografskoj lokaciji pozivaju se metode *listActiveUsers* i *listActiveUsersAtParticularLocation*, respektivno. Za slanje liste aktivnih korisnika koja uključuje i samog klijenta koji zahtijeva ovu listu, te slanje liste aktivnih korisnika na određenoj geografskoj lokaciji poziva se metoda *responseFromServerActiveUsers* i *responseFromServerActiveUsersAtParticularLocation*, respektivno. Metoda koja se poziva kada klijent šalje poruke ostalim korisnicima preko servera je *SendMessageToActiveUsersAtParticularLocation*. Konačno, kada server obavještava klijenta o porukama drugih klijenata na



određenoj lokaciji poziva se metoda *responseFromServerAboutMessage*.

| Modifikator pristupa, povratni tip metode | Signature metode  | Opis metode   |
|---|---|---|
| <b>public void</b>                        | connectToServer (IMessage msg)                                    | Metoda koja se poziva za spajanje klijenta na server  |
| <b>public void</b>                        | ResponsFromServerIdle (IMessage msg)                              | Metoda koja se poziva kada klijent dobija odgovor od servera o uspješnosti komunikacije                                     |
| <b>public void</b>                        | listActiveUsers (IMessage msg)                                    | Metoda koja se poziva kada klijent želi od servera listu aktivnih korisnika na serveru                                      |
| <b>public void</b>                        | listActiveUsersAtParticular Location(IMessage msg)                | Metoda koja se poziva kada klijent želi od servera listu aktivnih korisnika na preciziranoj geografskoj lokaciji            |
| <b>public void</b>                        | responseFromServerActive Users (IMessage msg)                     | Metoda koja se poziva kada server šalje klijentu listu aktivnih korisnika koja uključuje i samog klijenta                   |
| <b>public void</b>                        | responseFromServerActive UsersAtParticularLocation (IMessage msg) | Metoda koja se poziva kada server šalje korisniku listu aktivnih korisnika na preciziranoj geografskoj lokaciji             |
| <b>public void</b>                        | SendMessageToActive UsersAtParticularLocation (IMessage msg)      | Metoda koja se poziva kada klijent preko servera želi poslati poruku drugim klijentima na preciziranoj geografskoj lokaciji |
| <b>public void</b>                        | responseFromServerAbout Message(IMessage msg)                     | Metoda koja se poziva kada server obavještava klijenta o poruci drugim klijentima na preciziranoj geografskoj lokaciji      |

Tablica 3.6: Metode klase klijent