AGENTSKO OKRUŽENJE – DRUGA FAZA

Realizovati agentsko okruženje upotrebom JEE platforme, gde se sistem sastoji od sledećih elemenata:

• **Agentski centar** – čvor mreže koja predstavlja agentsko okruženje. Ovi čvorovi su zaduženi za upravljanje životnim ciklusom agenta, kao i za razmenu poruka među agentima;

• **Agent** – softverski entitet koji izvršava zadatak;

• **Klijent** – aplikacija koja se spaja na agentske centre i od njih traži da se izvrši određen zadatak upotrebom odgovarajućih agenata. Treba biti implementirana pomoću Angular, React ili Vue.js okruženja.

MODEL PODATAKA

Agentski centar sadrži sledeća polja: alias i address.

Agent klasa sadrži sledeća polja: Id – AID. Svaki agent je **stateful session bean**, koji kroz interfejs nudi **handleMessage(ACL poruka)** metodu, kojom će agent adekvatno reagovati na poruku. Pored navedenog, određen agent (naslednik Agent klase) će imati polja koja su relevantna za izvršavanje njegovog zadatka.

AID (Agent id) sadrži sledeća polja: name, host (**Agentski centar**) i type (**Agent type**).

Agent type klasa predstavlja tip agenta i sadrži sledeća polja: name i module.

ACL poruka

Poruka koju agenti razmenjuje između sebe i u komunikaciji sa agentskim centrom ima sledeća polja:

- Performative – Enum;
- Sender – AID;
- Receivers – AID[];
- ReplyTo – AID;
- Content – String;
- ContentObj – Object;
- UserArgs – HashMap<String, Object>;
- Language – String;
- Encoding – String;
- Ontology – String;
- Protocol – String;
- ConversationId – String;
- ReplyWith – String;
- InReplyTo – String;

• ReplyBy – Long;

KOMUNIKACIJA

Svaki REST zahtev ima prefiks „<host address>:<host port>”.

KLIJENT – AGENTSKI CENTAR

Klijent je front end web aplikacija koja vrši interakciju sa agentskim centrom po sledećem REST protokolu:

• **GET /agents/classes –** dobavi listu svih tipova agenata na sistemu;

• **GET /agents/running –** dobavi sve pokrenute agente sa sistema;

• **PUT /agents/running/{type}/{name} –** pokreni agenta određenog tipa sa zadatim imenom;

• **DELETE /agents/running/{aid} –** zaustavi određenog agenta;

• **POST /messages –** pošalji ACL poruku;

• **GET /messages –** dobavi listu performativa.

Klijentska aplikacija treba da podrži i automatsko osvežavanje liste tipova agenata, liste pokrenutih agenata i ispis razmenjenih poruka (log) putem WebSocket-a.

AGENTSKI CENTAR – AGENTSKI CENTAR

Agentski centar čini jedan čvor u mreži. Kada se podigne prvi agentski centar on postaje master čvor mreže. Svaki naredni agentski centar koji se podigne treba da kontaktira master čvor kako bi postao deo mreže. Prilikom aktiviranja novog ne-master čvora potrebno je da se izvrši handshake, baziran na REST pozivima, u kom će:

• **POST /node** – Nov ne-master čvor kontaktira master čvor koji ga registruje;

• **GET /agents/classes** – Master čvor traži spisak tipova agenata koje podržava nov ne-master čvor;

• **POST /node** – Master čvor javlja ostalim ne-master čvorovima da je nov ne-master čvor ušao u mrežu;

• **POST /agents/classes** – Master čvor dostavlja spisak novih tipova agenata (ukoliko ih ima) ostalim ne- master čvorovima;

• **POST /node** – Master čvor dostavlja spisak ostalih ne-master čvorova novom ne-master čvoru;

• **POST /agents/classes** – Master čvor dostavlja spisak tipova agenata novom ne-master čvoru koje podržava on ili neki od ostalih ne-master čvorova;

• **POST /agents/running** – Master čvor dostavlja spisak pokrenutih agenata novom ne-master čvoru koji se nalaze kod njega ili nekog od preostalih ne-master čvorova; Nije neophodno da naveden handshake bude u potpunosti ispoštovan što se tiče navedenih metoda (moguće je smanjiti broj zahteva upotrebom povratnih vrednosti funkcija). Ako se u bilo kom trenutku desi otkaz (tako što se, npr. desi timeout) vrši se ponovni pokušaj datog zahteva. Ako se otkaz ponovo desi potrebno je izvršiti rollback, što podrazumeva „čišćenje“ svakog čvora sistema od informacija koje je nov ne-master čvor ubacio u sistem sa sledećim REST pozivom:

• **DELETE /node/{alias}** – Master čvor javlja ostalim ne-master čvorovima da obrišu čvor koji nije uspeo da izvrši handshake, kao i sve tipove agenata koji su potencijalno dostavljeni ostalim čvorovima.

Ova operacija se takođe treba eksplicitno pokrenuti kada se neki čvor priprema za gašenje i želi da se odjavi iz klastera. Prilikom gašenja čvora treba pogasiti i sve agente koji trče na datom čvoru;

Svaki put kada se desi pokretanje agenta potrebno je izvršiti **POST /agents/running** zahtev na sve preostale čvorove u mreži, kako bi svi imali informaciju o najnovijem agentu. Kada se ažurira lista tipova agenata ili lista pokrenutih agenata potrebno je obavestiti WebSocket klijetne da ažuriraju svoje liste. Kada neki čvor prihvati poruku namenjenu za određenog agenta on prosleđuje tu poruku čvoru na kom se agent nalazi.

Svaki čvor u klasteru treba da implementira heartbeat protokol kojim periodično proverava da li su ostali članovi klastera i dalje živi. Protokol se svodi na sledeći REST poziv:

• **GET /node** – Ukoliko se desi da čvor ne odgovori zahtev se izvršava još jednom i ukoliko čvor ni tada ne odgovori smatra se da je ugašen i javlja se ostalim čvorovima da izbace zapis o ugašenom čvoru.

AGENTSKI CENTAR – AGENT

Kada agentski centar prihvati poruku od klijenta on treba da uposli određenog agenta da izvrši zadatak. Upotrebom JMSa centar ispaljuje poruku koju prihvata MDBConsumer. Consumer vrši lookup za agenta za kog je poruka namenjena i delegira mu poruku.

AGENTI

Za potrebe testiranja agentskog okruženja potrebno je implementirati nekoliko grupa agenata, po uzoru na Siebog sistem:

• Ping i pong;

•Contractnet protokol

ZA 9 i 10

Za višu ocenu neophodno je iskoristiti implementirano agentsko okruženje za rešavanje nekog distribuiranog AI zadatka. Studenti sami biraju zadatak koji će raditi i prijavljuju ga kao issue ovde**, nakon čega čekaju odobrenje asistenta.** Nakon odobrenja studenti mogu početi sa rešavanjem zadatka.