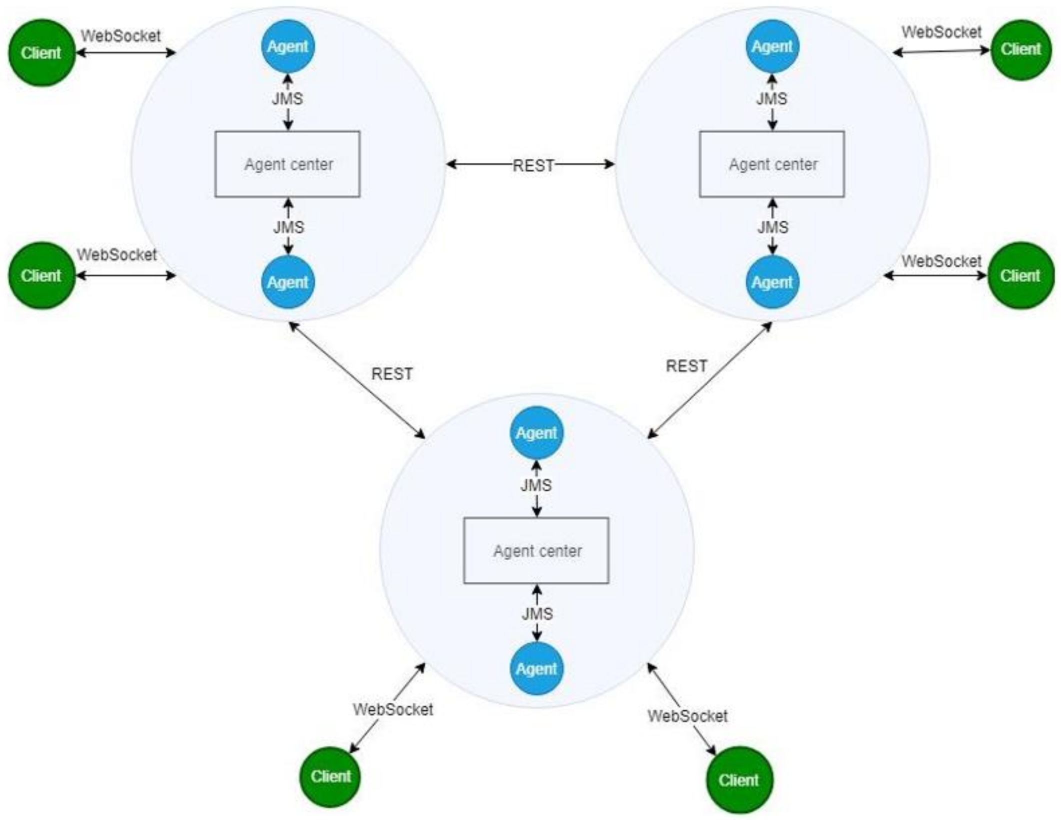
Chat Assignment

Implement Client-Server web application for chatting as an agent environment using JEE platform.

The system is composed of:

- **Host** – network node that represents agent environment. These nodes are responsible for managing agent lifecycle and providing message exchange between agents.
- **Agent –** software entity responsible for executing some tasks. In this assignment, agent represents user.
- **Client** – UI application that provides user with login, registration and chatting functionalities. It should be implemented using one of the following frameworks: Angular, React or Vue.js. You can opt to use standard HTML, CSS with Vanilla JavaScript.



Data Model

User contains fields: **username**, **password,** and **host** where user is logged in.
Host contains fields: **alias** (node's name) and **address**.
Message contains fields: **receiver** (User), **sender** (User), **timestamp** (Date of creation), **header** (message subject) and **content**.

Communication

Each REST request has the prefix **<host address>:<host port>**.

Client-Server communication

Client is a frontend application that provides user with registration, login and chatting functionalities. Implement the following REST protocols:

- POST /users/register – register user (username and password required)
- POST /users/login – log in user
- GET /users/loggedIn – list all logged-in users
- GET /users/registered – list all registered users
- POST /messages/all – message all logged-in users
- POST /messages/user – message only one user
- GET /messages/{user} – list all user's messages
- DELETE /users/loggedIn/{user} – log out user

Client application should automatically refresh list of logged-in users each time new user logs in (and logs out). It should also refresh list of received messages when a new message arrives. Implement these features using WebSocket protocol.

Server-Server communication

Server represents one network node. When the first host is run, it becomes the master node in the network. When another non-master host is run, it should first contact the master node. They should communicate following the handshake protocol, which is based on the REST requests:

- POST /register – new non-master node contacts the master node and requires registration
- POST /node – the master node notifies all other non-master nodes that a new node has been added to the network
- POST /nodes – the master node sends a list of all other system's non-master nodes to the new non-master node
- POST /users/loggedIn – master node sends a list of all users logged-in on it or any other system's non-master node to the new non-master node.

If a request can't be competed for any reason (e.g., timeout), the same request is being resent. If it fails again, then rollback action should be executed. Rollback implies that all network nodes should be cleansed of information introduced by a new non-master node. This should be performed using the following REST protocol:

- DELETE /node/{alias} – master node informs the other non-master nodes to remove node that has not managed to complete the handshake. This operation should be explicitly run when a node is ready to shut down and be removed from the cluster. When shutting a node down, all agents running on that node should be shut down as well.

Every time a new user logs in on some node, the POST /users/loggedIn request should be sent to all other network nodes, so every node has information about new user. Similar should be done when a user logs out. Furthermore, every client should update their list of logged-in users via WebSocket protocol. When node receives a message directed to a certain user, it forwards the message to a node where the user is logged in. The message list should also be updated using

WebSocket protocol upon receival.
Each node in the cluster should implement heartbeat protocol which periodically checks if all the other nodes in the cluster are still alive. Protocol consists of the following request:

- GET /node – If a node does not respond to a request, the request is being resent. If it still does not respond, then the node is probably shut down and DELETE /node/{alias} request is sent (all node's information are deleted).

GRADING

For 20 points you need to implement agent framework that does not work in cluster setting (only Client-Server communication).
For 30 points you need to implement agent framework that does work in cluster setting (Client-Server and Server-Server communication).

ADDITIONAL INFORMATION

- Chat assignment is not mandatory. If you decide to engage, you must do it individually.
- Assignment deadline is May 20th. Before the deadline you will be required to fill out the form (yet to be published) with a link to your GitLab/GitHub repository.
- For Server-Server communication, your application should run on (at least two) different machines (Hosts). You can use physically different machines or host and virtual machine.
- You are not required to implement data persistence. RAM data structures can be filled with data each time the application is launched.