Murlan Multiplayer

Technical Design Document

Angular – Spring Boot communication API

Angular StompClient -> Connect

* /game

Angular StompClient -> Subscribe

* /topic/lobby
  + LobbyEvent
    - LobbyEventType
      * ROOM\_OPENED
      * ROOM\_UPDATED
      * ROOM\_CLOSED
      * USER\_JOINED
      * USER\_LEFT
* /user/queue/lobby
  + LobbyMessage
    - LobbyMessageType
      * LOBBY\_STATE
      * CREATE\_ROOM\_SUCCESS
      * CREATE\_ROOM\_FAILURE
      * JOIN\_ROOM\_SUCCESS
      * JOIN\_ROOM\_FAILURE
* /topic/room/{roomId}
  + RoomEvent
    - RoomEventType
      * PLAYER\_JOINED
      * PLAYER\_DISCONNECTED
      * PLAYER\_LEFT
      * PLAYER\_PLAYED
      * ROUND\_STARTED
      * ROUND\_ENDED
      * GAME\_ENDED
* /user/queue/room/{roomId}
  + RoomMessage
    - RoomMessageType
      * ROOM\_STATE
      * INVALID\_PLAY
      * YOUR\_TURN\_TO\_PLAY
      * YOUR\_TURN\_WAS\_SKIPPED
      * YOU\_WON\_ROUND
      * YOU\_LOST\_ROUND

Angular StompClient -> Publish

* /app/lobby/command/createRoom – CreateRoomController.java
* /app/lobby/command/joinRoom – JoinRoomController.java
* /app/room/command/leaveRoom – LeaveRoomController.java
* /app/room/command/makePlay – MakePlayController.java

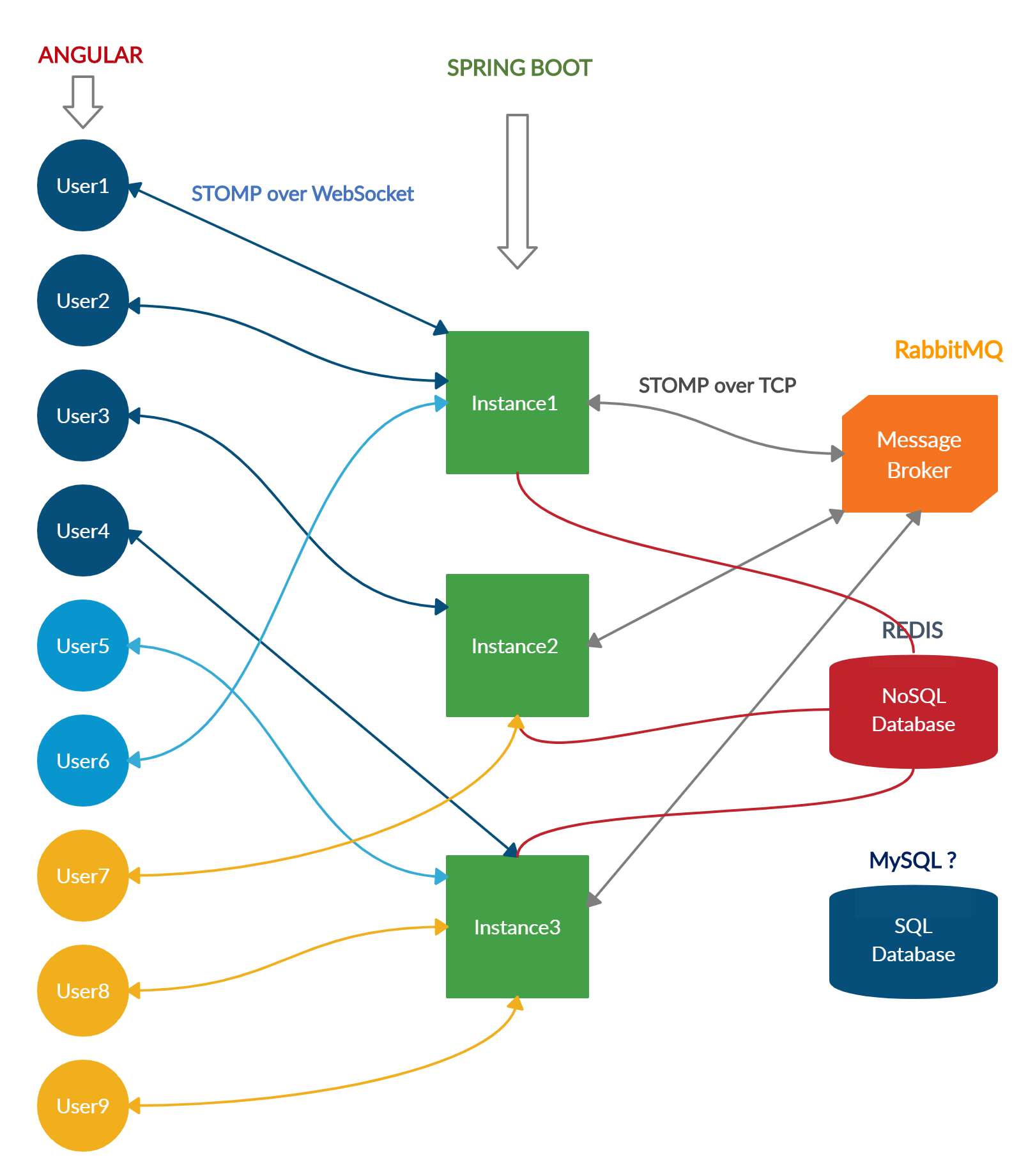
Functional requirements

* User is given the choice of playing as a guest.
* User visits the Lobby page and sees real time updates happening without his interaction.
* User is shown the option to create a new game room.
* User can select and join an available game room.
* After successfully joining, User is redirected to the Room page.
* Game automatically starts after the max number of players have joined.
* User's screen is updated in real time according to the game in play.
* On user's turn, user is asked to make his play.
* In case of connectivity issues, User should be informed accordingly.
* In case of connectivity issues, User should be able to re-join and play again, although the game may have continued in his absence.

Non-functional requirements

* Messages are exchanged with low latency.
* The order and time of messages sent to client are validated.
* Network error handling is consistent.
* Servers communicate with clients in real time and in an efficient way
* Software is designed to support horizontal scaling.
* Software is designed to support various types of game implementations in the future.
* System components can be easily added/removed based on the goals and requirements of the moment.

High-level overview



Simple use case – creating a new room

Precondition: Users have already connected and subscribed to /topic/lobby successfully.

