WhatsApp Web Simulator

This is a messaging simulator for web clients (education purpose). By now it is working on pure html, but it can run on different platforms because it is based on a **decoupled architecture**, using one layer based on **RESTful Api services** and other layer based on **WebSockets subscribers** for real time events. Both have been implemented using Javascript from the client side and Spring Boot from the server side.

The basics to understand the simulator:

* **RESTful Api services:** All the information from **client -> server** goes this way. The response from the server is processed too.

**REST Api**

* **WebSockets:** Events from **server -> client** use this communication channel. If the server needs to send a communication event it redirects the message to a WebSocket endpoint (topic), in this manner subscriptors can receive messages from these endpoinsts (topics).

**Web**

**Socket**

There are only **two kind of topics** (subscriptions) activated at once for each user connection, one associated to the **menu area** and other one for the **messages area**, ready to update the area they are associated to in case of any event.

**/topic/messages/{communicationId}/{telephone}/{telephoneTo}**

**/topic/menu/{telephone}**

WebSockets can work as transmitters or subscribers but this example only uses subscribers, because of REST Api microservices which are used at transmitter phase.

* **MongoDB:** NoSQL database chosen because of its JSON format and its transfer capacities. Only two collections, **‘telephones’** and **‘messages’** associated to each area to store information, without ACID transactions configured (all atomic).

There are two important fields regarding notifications, one for each collection; **advices** in ‘telephones’ which stores the number of unread messages, and **dates.receive** at ‘messages’ which indicates if a message have been read by the recipient of the message.

This documentation is oriented to understand the relation between communications channels for each case of use. After the installation (read **installation.txt**) you can execute and look at logs on your navigator and execution environment to understand the workflow.

**Access application** **Re/Connect to telephoneTo** **Disconnect application**

User

Topic

**3:unsubscribe message topic**

**4:subscribe message topic**

User

Topic

**5:unsubscribe all topics**

User

Topic

**1:add user**

**2:subscribe menu topic**

**1: (WS) SEND: /app/add/user**

**{**

**'telephone': '{telephone}'**

**}**

**2: (WS) SUBSCRIBE: /topic/menu/{telephone}**

**3: (WS) UNSUBSCRIBE: /topic/messages/{communicationId}/{telephone}/{telephoneTo}**

**4: (WS) SUBSCRIBE: /topic/messages/{communicationId }/{telephone}/{telephoneTo}**

**Receiver connected to the channel**

User 1

Topic

Mongodb

User 2

**1:send message**

**2:search connections**

**5:response message (received)**

**3:insert message (received)**

**4:send message (ws)**

Service

**1: (REST) POST: /messages/send/{communicationId}/{telephone}/{telephoneTo}**

**{**

**'type** **':** **'message** **',**

**'text': '{text}',**

**'images': {...},**

**'videos': {...},**

**'docs': {...},**

**'dates': {**

**'user': {time\_milliseconds}**

**}**

**}**

**4: (WS) SEND: /topic/messages/{communicationId}/{telephone}/{telephoneTo}**

**{**

**'type** **':** **'message** **',**

**'id': '5d8fd54a8ca1e10258088fb0',**

**'communicationId': '{communicationId}',**

**'text': '{text}',**

**'images': {...},**

**'videos': {...},**

**'docs': {...},**

**'dates': {**

**'user': {time\_milliseconds},**

**'send': {time\_milliseconds},**

**'receive': {time\_milliseconds}**

**}**

**}**

**5: (REST) response from 1**

**{**

**'type** **':** **'message** **',**

**'id': '5d8fd54a8ca1e10258088fb0',**

**'communicationId': '{communicationId}',**

**'text': '{text}',**

**'images': {...},**

**'videos': {...},**

**'docs': {...},**

**'dates': {**

**'user': {time\_milliseconds},**

**'send': {time\_milliseconds},**

**'receive': {time\_milliseconds}**

**}**

**}**

**Receiver not connected or connected but not to the same channel**

Service

User 2

Mongodb

Topic

User 1

**1:send message**

**2:search connections**

**3:insert message (no received)**

**4:update telephones (advice +1)**

**(if connected) 5:send telephones notification (ws)**

**6:response message (no received)**

**1: (REST) POST: /messages/send/{communicationId}/{telephone}/{telephoneTo}**

**{**

**'type** **':** **'message** **',**

**'text': '{text}',**

**'images': {...},**

**'videos': {...},**

**'docs': {...},**

**'dates': {**

**'user': {time\_milliseconds}**

**}**

**}**

**5: (WEBSOCKET) SEND: /topic/menu/{telephone}**

**{**

**'type': 'notification',**

**'advices':{**

**'{communicationId}': '{communicationId}'**

**}**

**}**

**6: (REST) response from 1**

**{**

**'type** **':** **'message** **',**

**'id': '5d8fd54a8ca1e10258088fb0',**

**'communicationId': '{communicationId}',**

**'text': 'text',**

**'images': {...},**

**'videos': {...},**

**'docs': {...},**

**'dates': {**

**'user': {time\_milliseconds},**

**'send': {time\_milliseconds}**

**}**

**}**

Receiver reconnects with lecture advices, sender connected or disconnected

User 1

Service

Mongodb

Topic

User 2

**1:send notification**

**2:update message (received)**

**3: update telephones (advice +1)**

**4:search connections**

**(if connected) 5:response notification (received)**

**6:response advice**

**1: (REST) POST: /telephones/notification/{communicationId}/{telephone}/{telephoneTo}**

**5: (WEBSOCKET) SEND: /topic/messages/{communicationId}/{telephone}/{telephoneTo}**

**{**

**'type': 'messageDelayed'**

**}**

**6: (REST) response from 1**

Add/Update user telephone

User 1

Service

Mongodb

User 2

**1:send telephone**

**2:add telephones**

**3: update telephone description**

**(if added) 4:send telephone (communication)**

**5:response telephone**

**1: (REST) POST: /telephones/telephone**

**{**

**'telephone': '{telephone}',**

**'telephoneTo': ' {telephoneTo}',**

**'description': ' {description}'**

**}**

**4: (WEBSOCKET) SEND: /topic/messages/{communicationId}/{telephone}/{telephoneTo}**

**{**

**'communicationId': ' {communicationId}',**

**'telephone': '{telephone}',**

**'telephoneTo': ' {telephoneTo}',**

**'description': ' {description}'**

**}**

**5: (REST) response from 1**

**Files Transfer**

During the files transfer, a copy of the **real file is saved into your local computer from the server (/whatsapp-simulator/upload/{fileType})** and only a **thumbnail** codified on **Base64** is sent to the receptor. The Base64 thumbnail and the path to the real file are stored for each message, so the file can be previewed and recovered to show it on the navigator.

**RabbitMQ**

The application is prepared to use RabitMQ to improve messaging, it can be enabled uncommenting the dependency **spring-boot-starter-amqp** at pom.xml and configuring your WebSocket broker. Some configuration files like Dockerfile or k8s-deployment.yaml have been included just in case you want to go further.