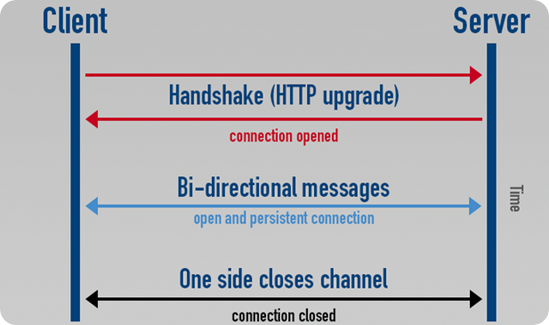
**WEBSOCKET Test of concept**

The behavior of a websockets-type connection allows a full-duplex bidirectional communication (client-server), with capacity to send data at any time. Websockets communicates through the TCP layer, but to open the connection is established by http.



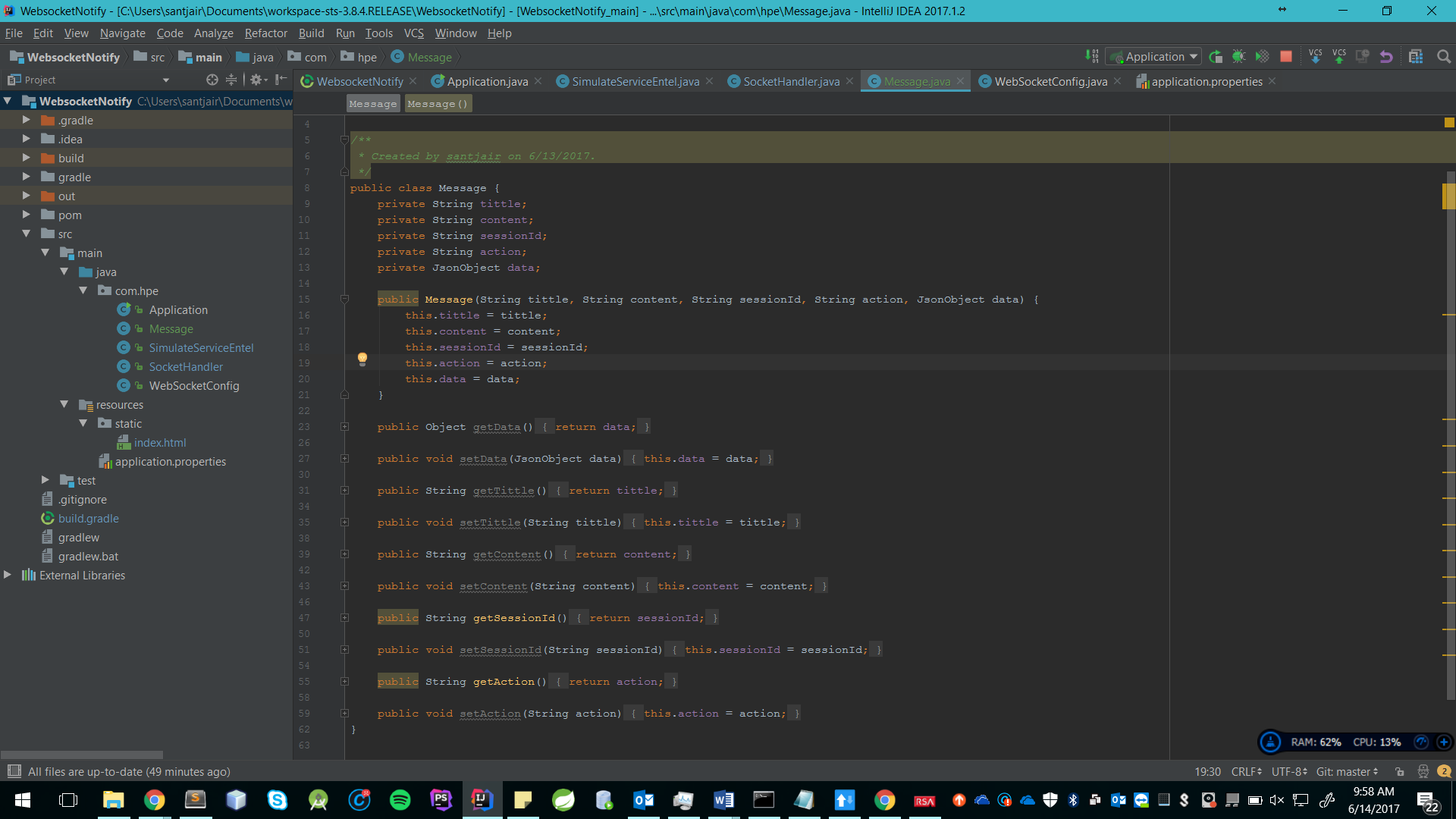
Unfortunately the Jboss version 7.1.3 of the HPSA 7 doesn´t supports this communication. As from Wildfly 8+ Websocket is supported, and standardized by the IETF as RFC 6455. There are two ways to resolve the incompatibility of the current version of Jboss. First, configure another Wildfly 8+ application server and execute an application (WAR). The second way, make a microservice that allows executing a tomcat embedded with the spring boot framework.

**Microservice:**

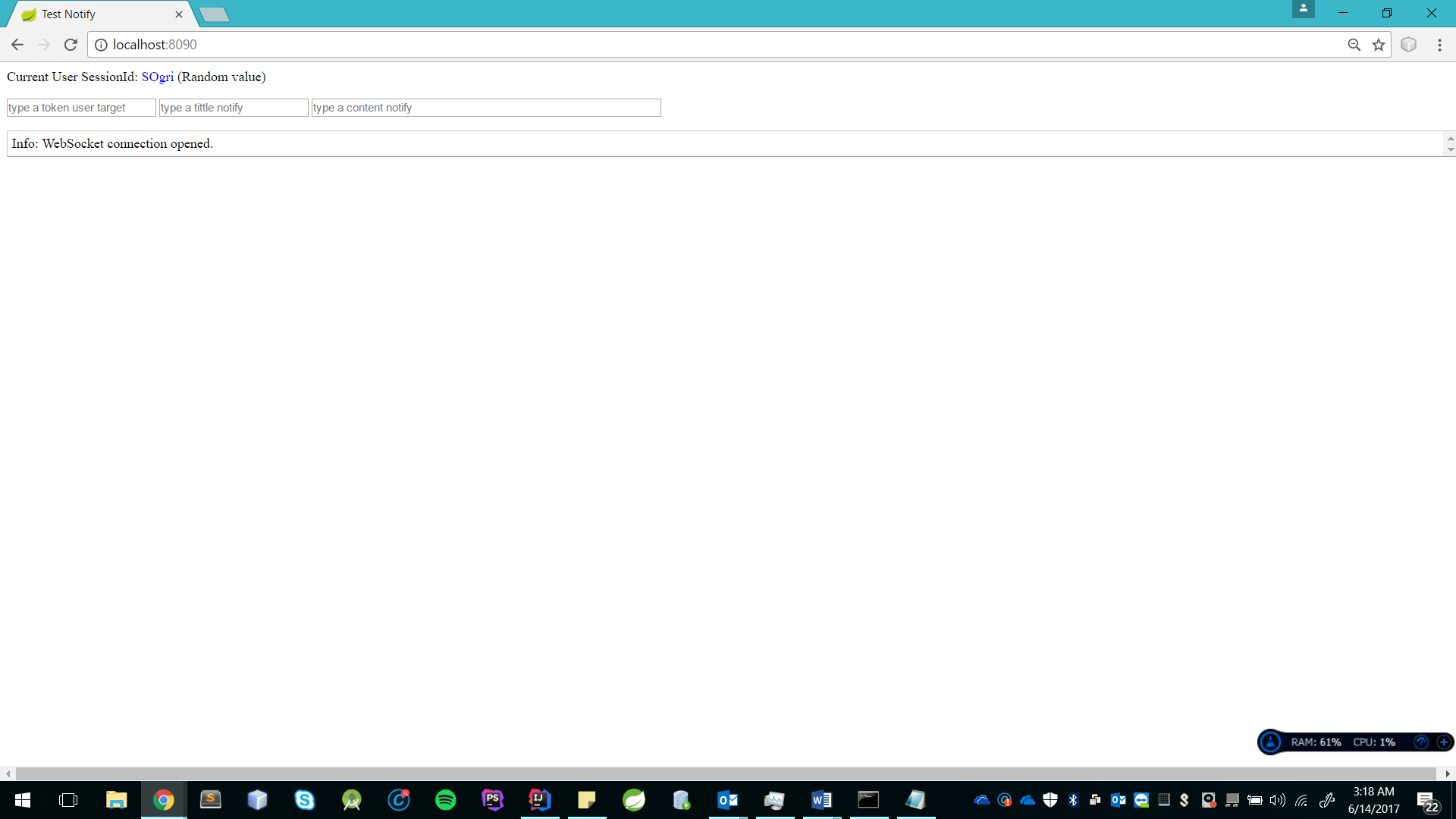
Spring libraries (org.springframework.web.socket) are used by websockets, for the SocketHandler class. This class contains the 4 basic events for websockets (afterConnectionEstablished, handleTextMessage, afterConnectionClosed, handleTransportError).

It contains a function to perform unicasting to send messages from the server to the client with a token attribute; in this case it has been called "sessionId", by business logic.

It also contains a Bean "Message" associated to the structure of the message by webockets. Moreover, the "WebSocketConfig" class as called "notify", configures the endpoint for the websocket.

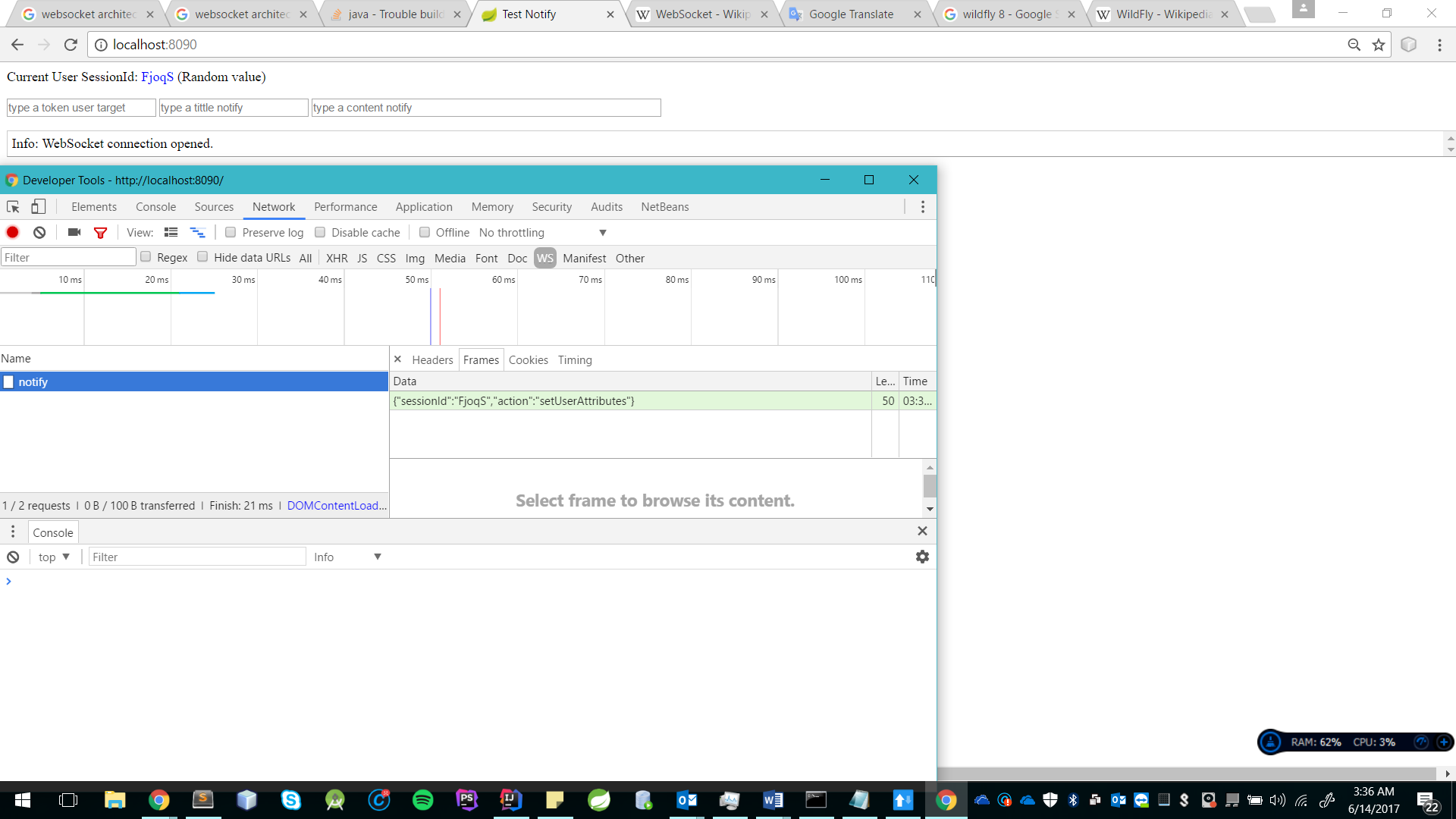


**Stage**

• First, the project has a demo to simulate the Android client side (index.html).

Generate a sessionId randomly for each session, each time the page loads. In this example, I get "SOgri".

At the start, a request is made to open the websocket connection, if the connection is successful a message is sent in json format with the sessionId and the message action with value "setUserAttributes" websocket. All this behavior is executed by just entering the "index.html"

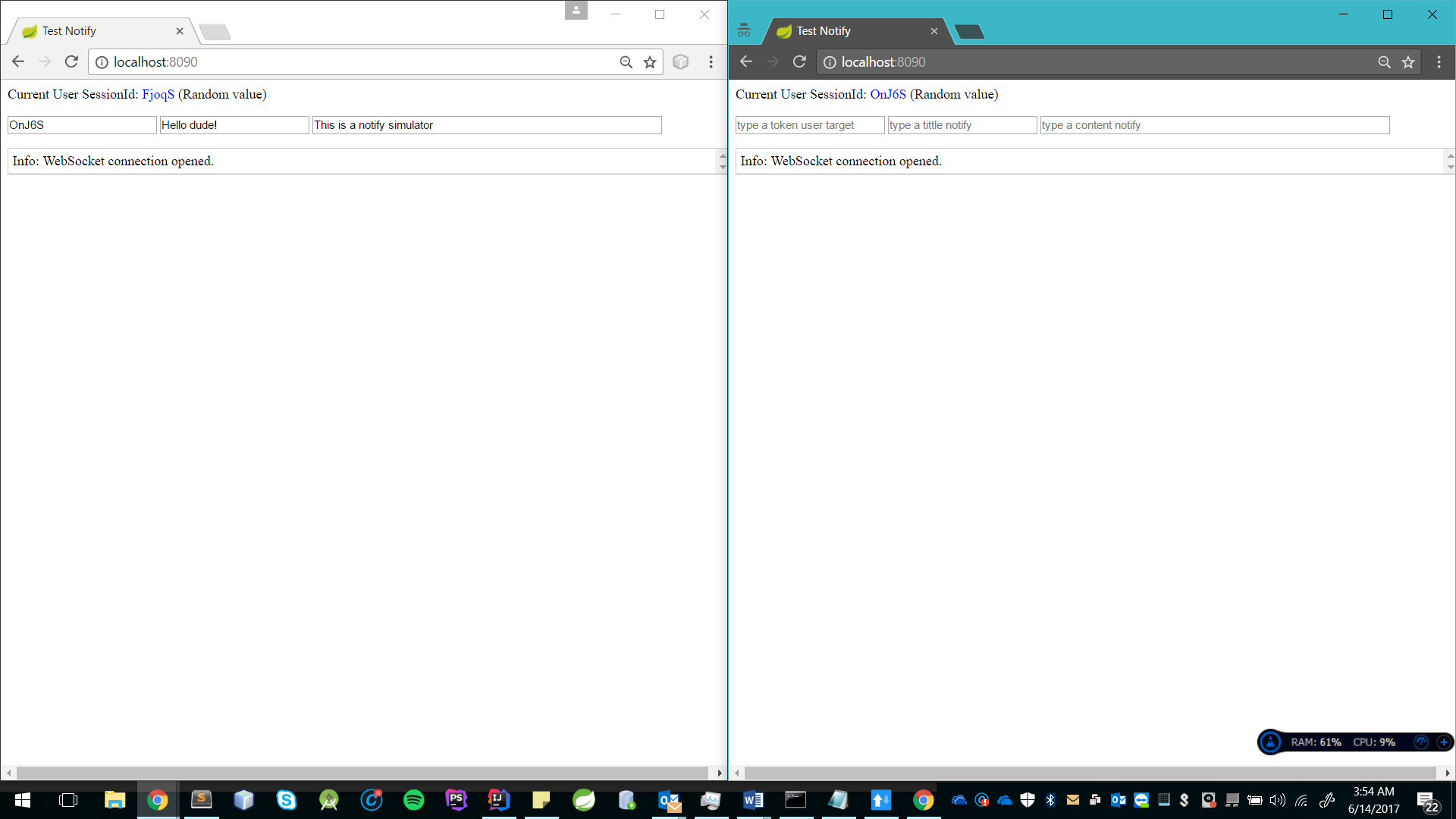


The sessionId is sent to identify which user is going to be sent the notifications.

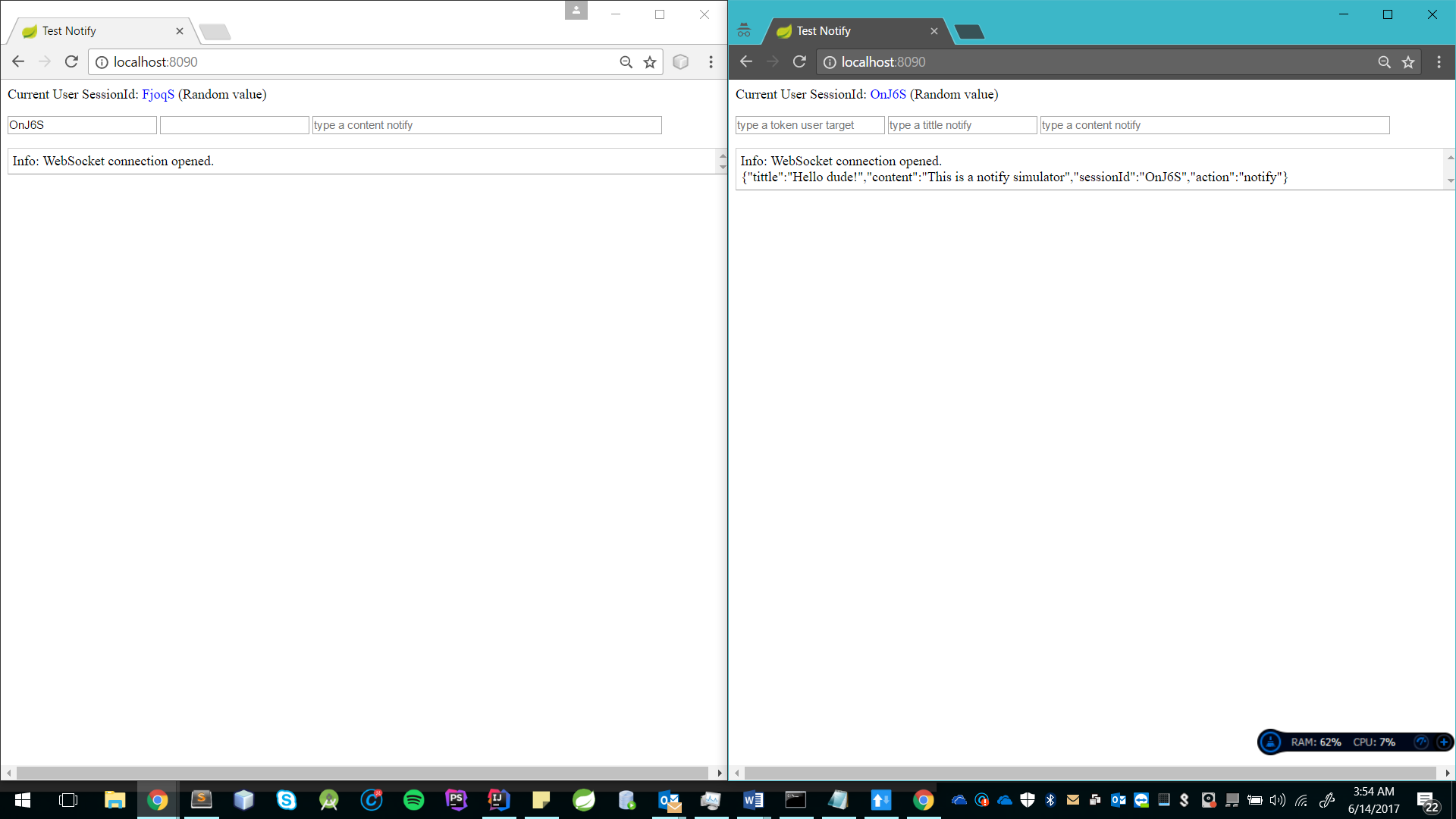
On the server side all the attributes of the connection are stored in Map-type Collections with "key and value", to add additional data to the connection. This is done because "WebsocketSession" doesn't allow adding more attributes.

static Set<Map<String, Object>> connections = Collections.synchronizedSet(new HashSet< Map<String, Object>>()) ;

To simulate bidirectional communication has to open in incognito mode or also open it in another browser.

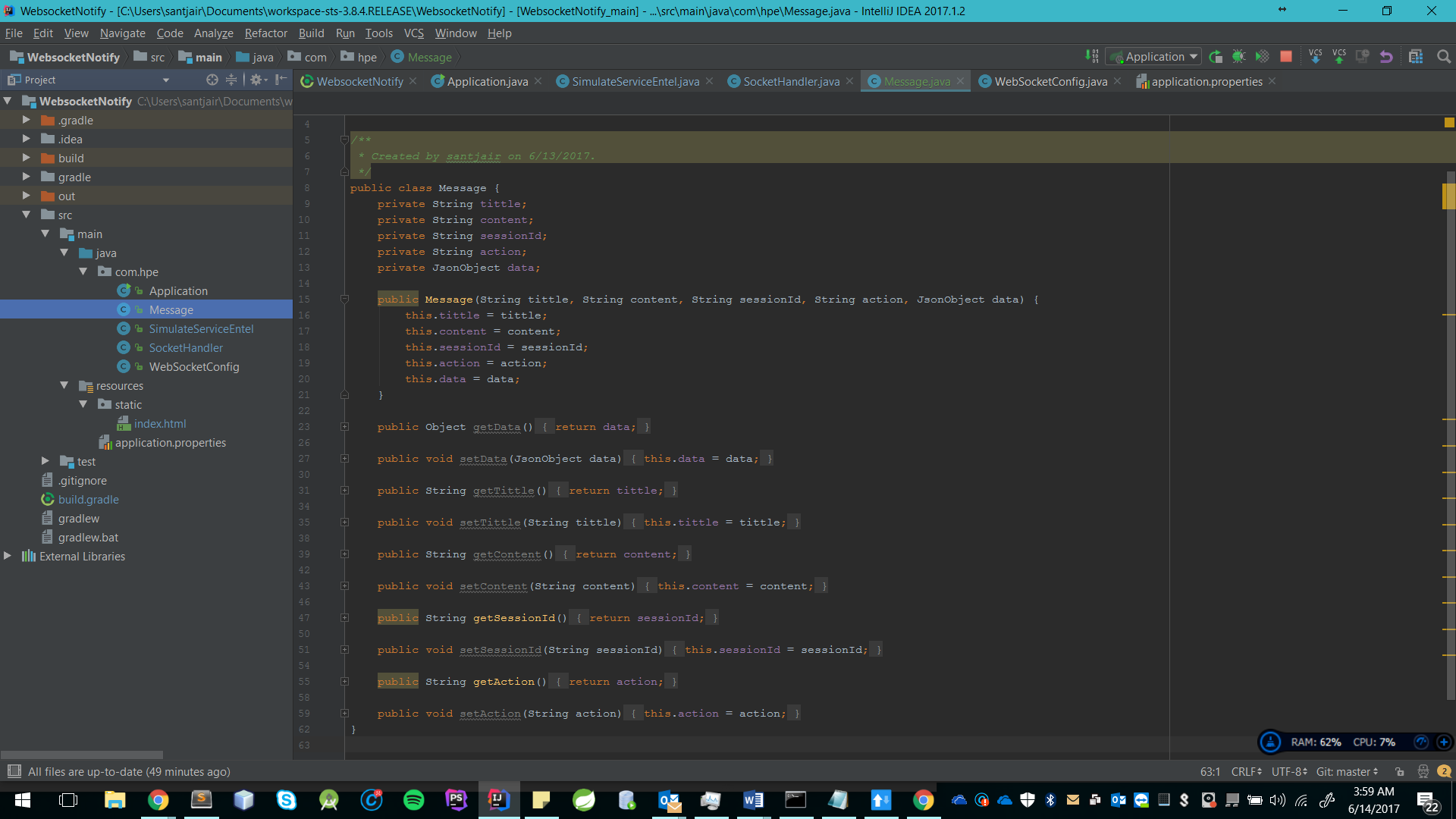


Pressing enter sends a message, the server receives it, searches in the collection the sessionId, the sessionId is associated with a WebSocketSession, and identifies to which client browser to send.



The second client browser receives a text sequence of type json and prints it. This demo simulates the data that the Android application needs to be able to notify.

The Bean "Message” facilitates the structure to send the data.



Contains additional attributes (tittle, content, sessionID, action) by default for the notification action that the Android application must perform and an attribute called "data" of type "JsonData" to send data in a dynamic way and json type without any problem.

{

"tittle":"Entel Mayoristas",

"content":"Nueva respuesta",

"sessionId":"004of",

"action":"notify",

"data":{

"txnId":"POR10001",

"docValue":"430154100",

"portedMsisdn":"999999999",

"requestStartTime":"24/04 16:15",

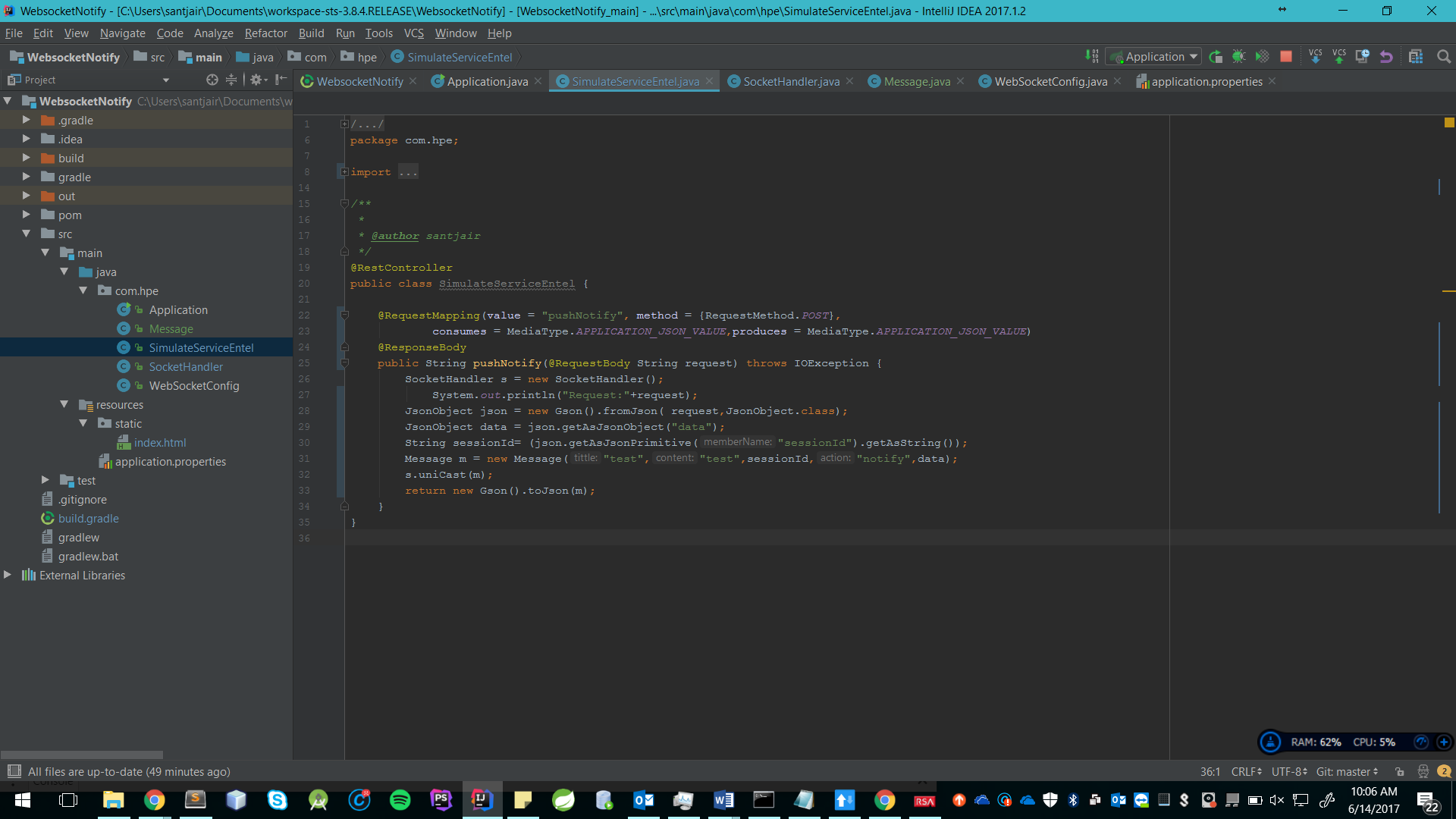
"statusDescription":" Rechazada",

"reasonDescription":"El cliente mantiene una deuda con su operador actual "

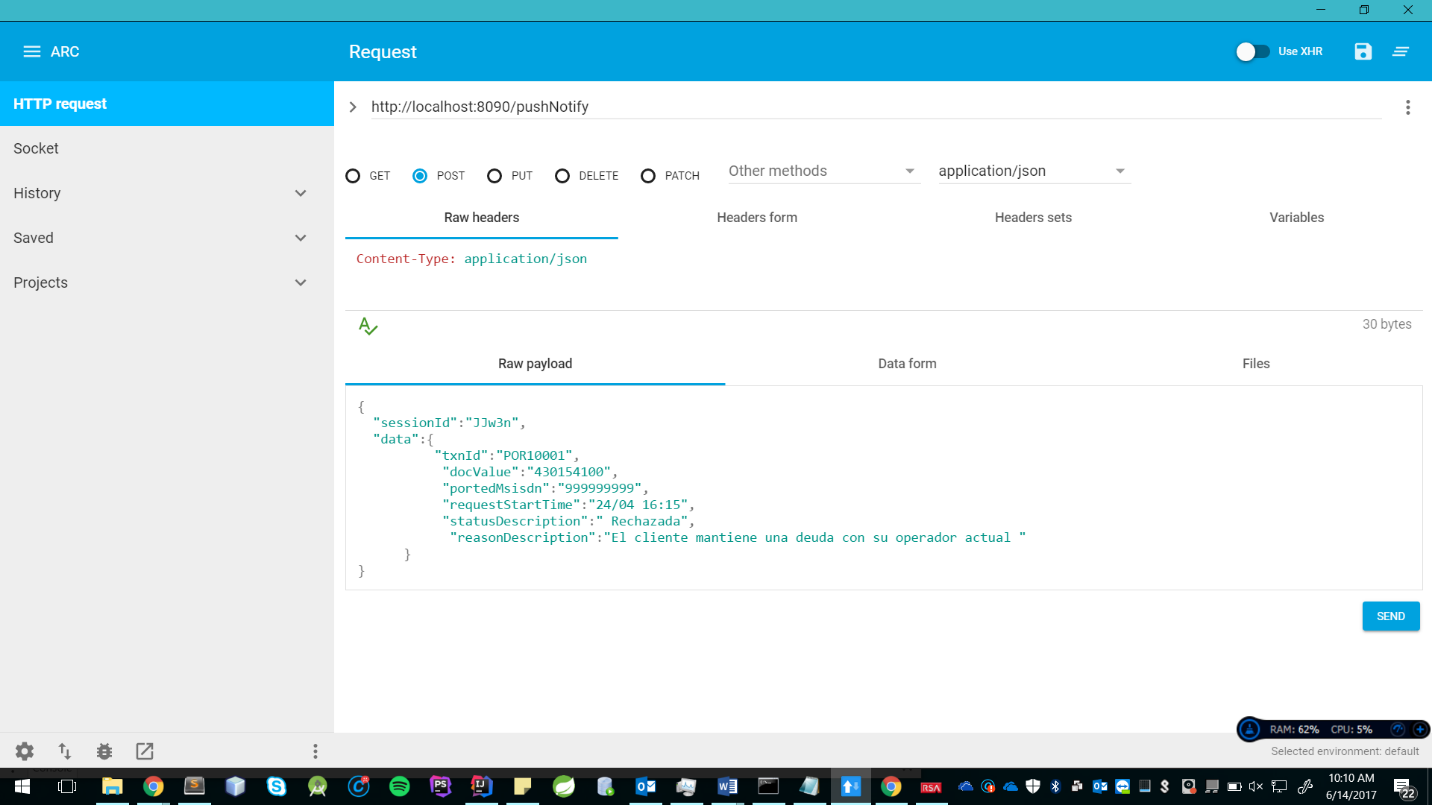
}

}

Exist another "SimulateServiceEntel" class, this class contains an endpoint that when executing the request, simulates the Push behavior for the client (Android Application) and makes notification possible without having a websockets connection. Basically, it enters the Connection Collection and searches for the associated sessionID for notify.



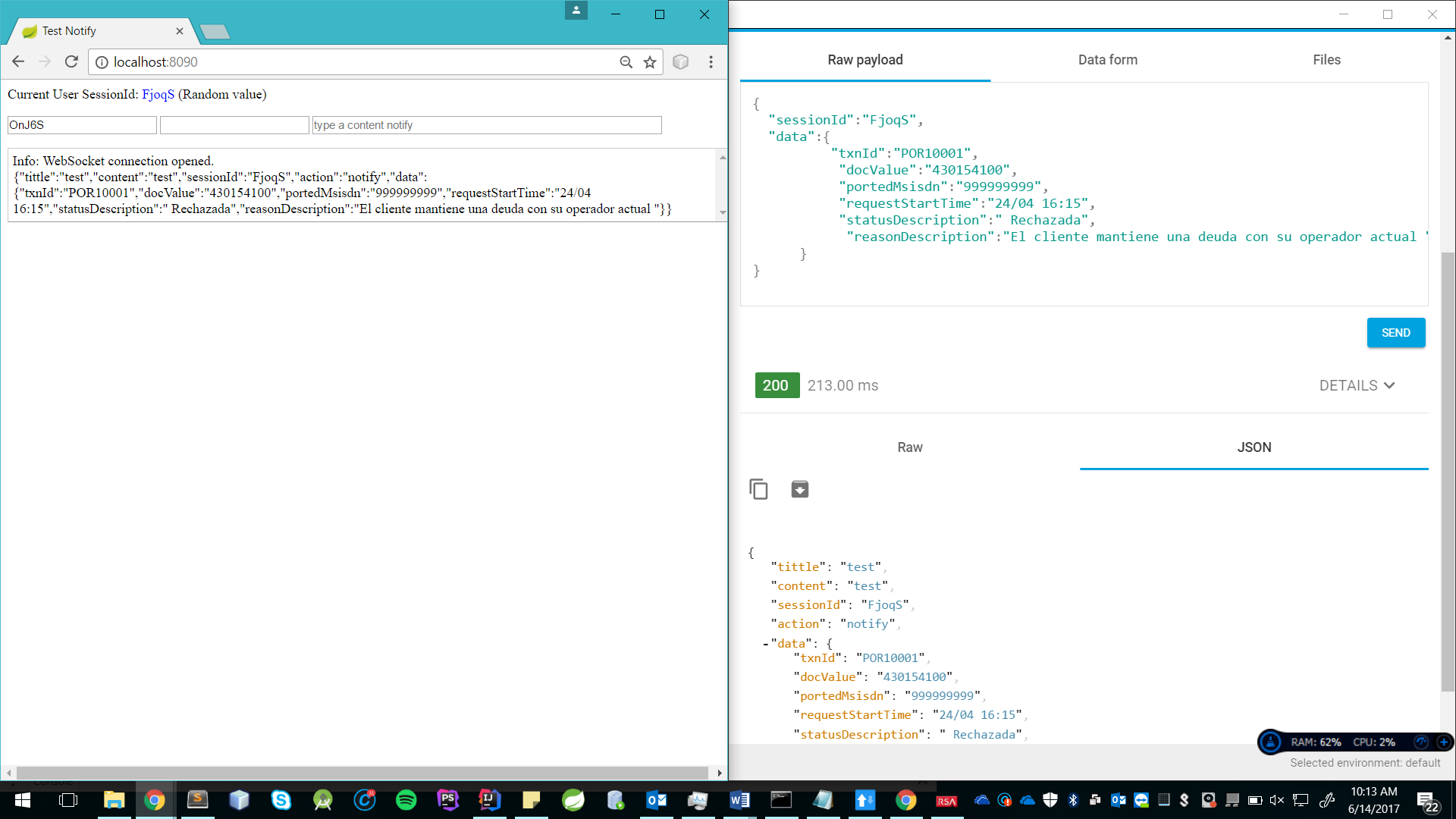
The endpoint is invoked with the post method and the request content is of type "json / application".



The test was performed using the "Advanced Rest Client" tool, sent the sessionId to identify the web client and the data that need to be sent.

In this example I sent a sessionId of value "FaqS", and I get in response the values needed in the notification, this answer is fully personalized.

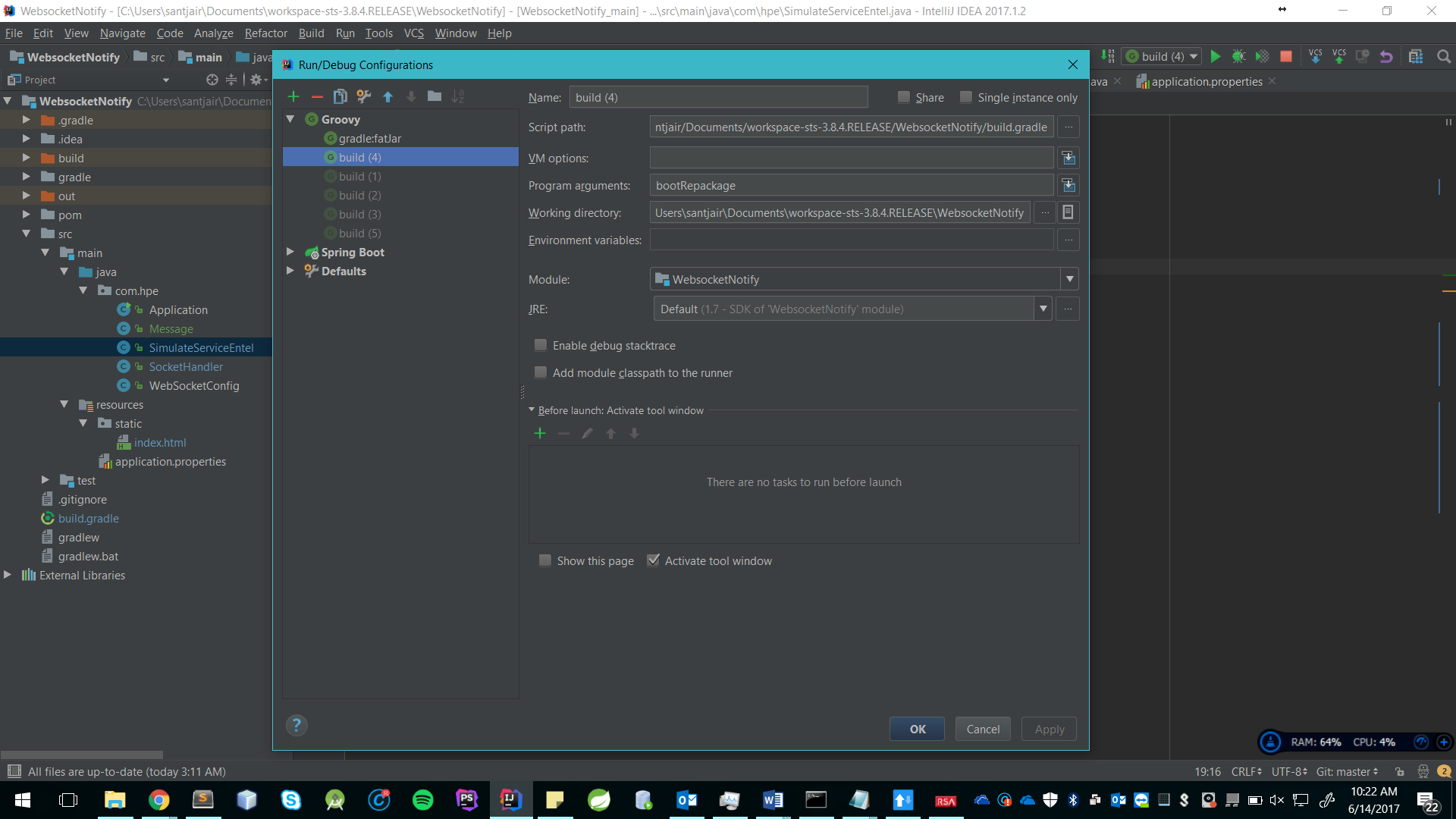
Finally get the notification from the web client side successfully.



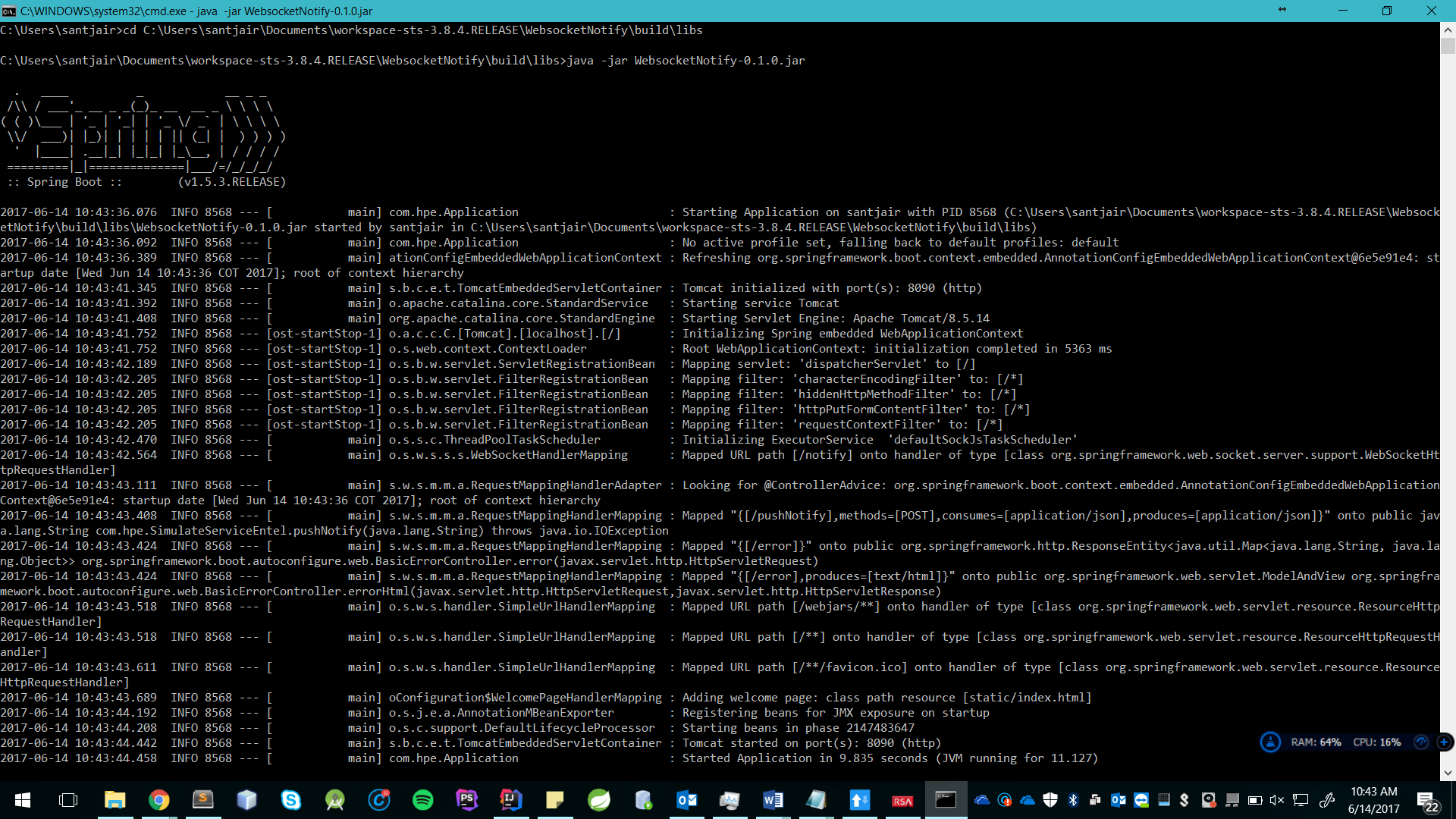
**Deployment:**

In case you have problems with "Gradle", there is also a "Pom.xml" file inside the project of the "Pom" folder, for settings directly as a Maven project.

Preferably use the IDE Intellij IDEA or directly with Gradle from the command line execute the task "bootRepackage", to construct an executable of type "JAR"



It is located inside the "/ build / libs" folder with the file "WebsocketNotify-0.1.0.jar". To start the application execute "java -jar WebsocketNotify-0.1.0.jar".



For more information on the deployment visit: <https://spring.io/guides/gs/messaging-stomp-websocket/>