Giulio Bosco, I4AA

Diario di lavoro

Luogo	Canobbio
Data	02.03.2020

Lavori svolti

Oggi ho iniziato cercando gli errori che trovavo nel codice, il primo che ho trovato, sono le richieste relative hai dati, come **http://localhost:8080/data/user/1** questa richiesta ritorna un errore not found, siccome nella classe **FreqlineServer** le classi vengono aggiunte per la path indicate, senza cio che segue, quini ho modificato il metodo **addServlet** come segue:

```
private void addServlet(ServletContextHandler context, BaseServlet baseServlet) {
    context.addServlet(new ServletHolder(baseServlet), "/" + baseServlet.getPath() + "/*");
}
```

Dopo di che ho sistemato la classe **LoginServlet** con le sue risposte, in caso di login o di errore:

```
 if (user != null) {
    sm.initSession(user);
    ok(response, LOGGED_IN);
} else {
    sm.destroySession();

    unauthorized(response, WRONG_USERNAME_PASSWORD);
}
```

Dopo di che ho aggiunto l'azione Logout che mancava.

La quale principalmente elimina la sessione attiva con il metodo **checkOldSession()** della classe **LoginServlet**.

```
/**
 * Do log out servlet get request.
 *
 * @param req  Http request.
 * @param resp Http response.
 * @throws ServletException Error in servlet.
 * @throws IOException      Input Output Error.
 */
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    LoginServlet.checkOldSession(req);

    ok(resp, LOGGED_OUT);
}
```

Dal momento che non ho trovato altri errori nel codice del back-end quindi ho deciso di iniziare a sistemare il codice del front-end, cosi nel caso in cui trovassi li degli errori del back-end potrei risistemarli avendo ancora tempo.

Dopo di che eseguendo le richieste tramite Angular JS sono ri-incappato nel problema di CORS, che ho risolto assieme al compagno di classe Filippo Finke, aggiungendo agli header **0** ed **1** nel parametro **Access-Control-Allow-Headers**.

Dopo di che ho deciso di aggiungere una variabile globale, per cambiare l'url di base a cui fare le richieste, quindi nel file di inizializzazione della webapp, ho deciso di aggiungere una variabile:

```
app.value('baseUrl', 'http://localhost:8080');
```

Dopo di che ho modificato tutti i service, inizializzandoli come segue:

```
app.factory('GeneratorDecibelService', ['$http', 'baseUrl', function($http, baseUrl) {
    var serviceUrl = baseUrl + "/action/generatorDecibel";
```

Per utilizzare l'url di base istanziato precedententemente.

Dopo di che mi sono accorto che dopo un tot di richieste il server si blocca siccome ha ricevuto troppe richieste.
Per risolvere il problema ho semplicmente chiuso il **JdbcConnector** tutte le volte che lo aprivo, come nell possibile esempio che segue:

```
JdbcConnector connector = null;
try {
    connector.openConnector();

    // execute queries
} catch (SQLException sqle) {
    // execute catch
} finally {
    if (connector != null) {
        connector.close();
    }
}
```

Il codice soprastante permette di chiudere la connessione ogni volta la quale viene utilizzata.

Problemi riscontrati e soluzioni adottate

CORS:
Avendo su due porte diverse il server del front-end e del back-end avevo dei problemi relativi al cors che ho risolto inserendo la provenienza dell'host.

Too many connections:
L'errore era dovuto alla connessione al database mysql, rileggendo attentamente il codice ho notato che non veniva chiusa mai la connessione a mysql.

Punto della situazione rispetto alla pianificazione

Sono circa in linea con la pianificazione, ho deciso di fare assieme 2 attività. (13-14)

Programma di massima per la prossima giornata di lavoro

Risolvere i problemi rimanenti relativi al back-end ed al front-end.