```xml
<?xml version="1.0" encoding="utf-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version
="2.4"
         xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <servlet>
        <servlet-name>eventsServlet</servlet-name>
        <servlet-class>it.polimi.iol.duque.EventsServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>eventsServlet</servlet-name>
        <url-pattern>/eventsServlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>statsServlet</servlet-name>
        <servlet-class>it.polimi.iol.duque.StatsServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>statsServlet</servlet-name>
        <url-pattern>/statsServlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>loginServlet</servlet-name>
        <servlet-class>it.polimi.iol.duque.LoginServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>loginServlet</servlet-name>
        <url-pattern>/loginServlet</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>usersServlet</servlet-name>
        <servlet-class>it.polimi.iol.duque.UsersServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>usersServlet</servlet-name>
        <url-pattern>/usersServlet</url-pattern>
    </servlet-mapping>


    <!-- DB Parameters -->
    <context-param>
        <param-name>username</param-name>
        <param-value>Duque</param-value>
    </context-param>
    <context-param>
        <param-name>password</param-name>
        <param-value>postgres</param-value>
    </context-param>
    <context-param>
        <param-name>dbDriver</param-name>
        <param-value>org.postgresql.Driver</param-value>
    </context-param>
    <context-param>
        <param-name>dbURL</param-name>
        <param-value>jdbc:postgresql://localhost:5432/tudex</param-value>
    </context-param>

</web-app>
```

```
<?xml version="1.0"?>
<project name="TUDEX" default="main" basedir=".">

    <property file="im.properties" />

    <path id="classpath">
        <fileset dir="${lib.dir}" includes="**/*.jar"/>
    </path>

    <!-- ================================================ -->
    <!-- Init all java files                              -->
    <!-- ================================================ -->
    <target name="init" />

    <!-- ================================================ -->
    <!-- Clean project from class files                   -->
    <!-- ================================================ -->
    <target name="clean">
        <delete dir="${build.dir}"/>
        <delete dir="${dist.dir}"/>
    </target>

    <!-- ================================================ -->
    <!-- Compile all java files                           -->
    <!-- ================================================ -->
   <target name="compile" depends="init">
        <mkdir dir="${classes.dir}"/>
        <javac srcdir="${src.dir}" destdir="${classes.dir}" classpathref="classpath" debug="on"/>
         <copy todir="${classes.dir}">
            <fileset dir="${src.dir}" excludes="**/*.java"/>
        </copy>
    </target>

    <!-- ================================================ -->
    <!-- Create a jarfile                                 -->
    <!-- ================================================ -->
    <target name="jar" depends="compile">
        <mkdir dir="${jar.dir}"/>

        <jar destfile="${jar.dir}/${ant.project.name}.jar" basedir="${classes.dir}">
            <manifest>
                <attribute name="Built-By" value="${user.name}" />
                <attribute name="Main-Class" value="${main-class}"/>
                <attribute name="Class-Path" value="${libs.project}" />
            </manifest>
        </jar>
    </target>

    <!-- ================================================ -->
    <!-- Create a distributuin file (jar file with all jar's
)                                                                    -->
    <!-- ================================================ -->
    <target name="dist" depends="jar" description="Create binary distribution">
      <delete dir="${dist.dir}" />

      <!-- contains all library dependencies -->
      <copy todir="${dist.dir}" file="${jar.dir}/${ant.project.name}.jar" />

    </target>

    <!-- ================================================ -->
    <!-- First delete all classes than rebuild jar
                                                                    -->
    <!-- ================================================ -->
    <target name="clean-build" depends="clean,jar"/>

    <!-- ================================================ -->
    <!-- Make a fresh rebuild and deploy everything jar
                                                                    -->
    <!-- ================================================ -->
    <target name="main" depends="copyclasses"/>


    <!-- ================================================ -->
    <!-- Deploy jar to WebApplication
-->
    <!-- ================================================ -->
    <target name="copyclasses" depends="dist">
            <mkdir dir="${deploy.path}/TUDEX"/>

            <copy todir="${deploy.path}/TUDEX" overwrite="true">
                        <!--<fileset dir="${html.dir}">
                            <include name="**/*" />
                        </fileset>
                        <fileset dir="${jsp.dir}">
                            <include name="**/*" />
                        </fileset>-->
                <fileset dir="${webapp.dir}">
                    <include name="**/*" />
                </fileset>
            </copy>

            <copy todir="${deploy.path}/TUDEX/WEB-INF/lib" overwrite="true">
                <fileset dir="${dist.dir}">
                    <include name="**/*" />
                </fileset>
                <fileset dir="${lib.dir}">
                    <include name="**/*" />
                </fileset>
            </copy>
```

```
        <copy todir="${deploy.path}/TUDEX/WEB-INF" overwrite="true">
            <fileset dir=".">
                <include name="**/web.xml" />
            </fileset>
        </copy>

    </target>
</project>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<module type="JAVA_MODULE" version="4">
  <component name="FacetManager">
    <facet type="web" name="Web">
      <configuration>
        <descriptors>
          <deploymentDescriptor name="web.xml" url="file://$MODULE_DIR$/web.xml" />
        </descriptors>
        <webroots>
          <root url="file://$MODULE_DIR$" relative="/WEB-INF" />
        </webroots>
      </configuration>
    </facet>
  </component>
  <component name="NewModuleRootManager" inherit-compiler-output="true">
    <exclude-output />
    <content url="file://$MODULE_DIR$">
      <sourceFolder url="file://$MODULE_DIR$/src" isTestSource="false" />
    </content>
    <orderEntry type="inheritedJdk" />
    <orderEntry type="sourceFolder" forTests="false" />
    <orderEntry type="library" name="lib" level="project" />
    <orderEntry type="library" name="JPA 2.0-2.0" level="project" />
    <orderEntry type="library" name="Hibernate 5.2.9-5.2.9" level="project" />
  </component>
</module>
```

```
deploy.path=/usr/local/Cellar/tomee-plus/1.7.4/libexec/webapps
src.dir=src
webapp.dir=webapp
lib.dir=lib
build.dir=bin
classes.dir=${build.dir}/classes
jar.dir=${build.dir}/jar
dist.dir=${build.dir}/dist
main-class=
libs.project=
```

File - /Users/duque/MyStuff/PoliMi/WebTecnologies/PrimaProva/campus-explorer/TUDEX/src/it/polimi/iol/duque/DBManager.java

```java
package it.polimi.iol.duque;

import it.polimi.iol.duque.model.Events;
import it.polimi.iol.duque.model.Users;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * Created by Duque on 01/05/2017.
 */
public class DBManager {

    private static Connection connection = null;  // Connection to DataBase

    public static void initConnection(ServletConfig config) throws ServletException {
        if (connection == null) {

            ServletContext context = config.getServletContext();
            String loginUser = context.getInitParameter("username");
            String loginPasswd = context.getInitParameter("password");
            String loginUrl = context.getInitParameter("dbURL");

            // Load PostgreSQL driver and sets the connection up
            try {
                Class.forName("org.postgresql.Driver");
                connection = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
                context.setAttribute("db-connection", connection);
            } catch (ClassNotFoundException ex) {
                System.err.println("ClassNotFoundException: " + ex.getMessage());
                throw new ServletException("Class not found Error");
            } catch (SQLException ex) {
                System.err.println("SQLException: " + ex.getMessage());
            }
        }
    }


    public static List<Events> getAllEvents() {
        List<Events> events = new ArrayList();

        try {
            String query = "SELECT * FROM events";
            PreparedStatement statement = connection.prepareStatement(query);
            try (ResultSet rs = statement.executeQuery()) {
                while (rs.next()) {

                    Events event = new Events();
                    event.setType(rs.getString("type"));
                    event.setId(rs.getInt("id"));
                    event.setTitle(rs.getString("title"));
                    event.setDescription(rs.getString("description"));
                    event.setFlames(rs.getInt("flames"));
                    event.setLatitude(rs.getDouble("latitude"));
                    event.setLongitude(rs.getDouble("longitude"));
                    event.setStartTime(rs.getLong("starttime"));
                    event.setEndTime(rs.getLong("endtime"));

                    events.add(event);
                }
                rs.close();
            }
            statement.close();

        } catch (Exception ex) {
            ex.printStackTrace();
            String errMsg = "An error occurred while getting events: " + ex.getMessage();
            System.err.println(errMsg);
        }
        return events;
    }

    public static List<Users> getAllUsers() {
        List users = new ArrayList<Users>();

        try {
            String query = "SELECT * FROM users";
            PreparedStatement statement = connection.prepareStatement(query);
            try (ResultSet rs = statement.executeQuery()) {
                while (rs.next()) {

                    Users user = new Users();
                    user.setId(rs.getInt("id"));
                    user.setUsername(rs.getString("username"));
                    user.setSurname(rs.getString("surname"));
                    user.setPoints(rs.getInt("points"));
                    user.setFaculty(rs.getString("faculty"));

                    users.add(user);
                }
                rs.close();
            }
            statement.close();

        } catch (Exception ex) {
            ex.printStackTrace();
            String errMsg = "An error occurred while getting users: " + ex.getMessage();
```

```
            System.err.println(errMsg);
        }

        return users;
    }

    public static Users getUser(String username, String password) {
        Users user = new Users();

        try {

            String query = "SELECT * FROM users WHERE username=? AND password=?";
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setString(1, username);
            statement.setString(2, password);

            ResultSet rs = statement.executeQuery();
            while (rs.next()) {

                user.setId(rs.getInt("id"));
                user.setUsername(rs.getString("username"));
                user.setSurname(rs.getString("surname"));
                user.setPoints(rs.getInt("points"));
                user.setFaculty(rs.getString("faculty"));

            }
            rs.close();

            statement.close();

        } catch (Exception ex) {
            ex.printStackTrace();
            String errMsg = "An error occurred while getting users: " + ex.getMessage();
            System.err.println(errMsg);
        }

        return user;
    }

    public static void updateUserPoints(Users user) {
        try {

            String query = "UPDATE users SET points=? WHERE id=?";
            PreparedStatement statement = connection.prepareStatement(query);
            statement.setInt(1, user.getPoints());
            statement.setInt(2, user.getId());

            ResultSet rs = statement.executeQuery();
            rs.close();

            statement.close();

        } catch (Exception ex) {
            ex.printStackTrace();
            String errMsg = "An error occurred while updating user: " + ex.getMessage();
            System.err.println(errMsg);
        }
    }
}
```

File - /Users/duque/MyStuff/PoliMi/WebTecnologies/PrimaProva/campus-explorer/TUDEX/src/it/polimi/iol/duque/LoginServlet.java

```java
package it.polimi.iol.duque;

import com.google.gson.Gson;
import it.polimi.iol.duque.model.Users;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Created by Duque on 01/05/2017.
 */
public class LoginServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("application/json");

        String username = request.getParameter("username");
        String password = request.getParameter("password");

        Users user = DBManager.getUser(username, password);
        String userJson = new Gson().toJson(user); // Here I'm using Gson, one of the most popular JSON
serializing libraries

        System.out.println(user);
        response.getWriter().print(userJson);
    }

    @Override
    public void init(ServletConfig config) throws ServletException {
        DBManager.initConnection(config);//Init database connection
        super.init(config);
    }
}
```

```java
package it.polimi.iol.duque;

import com.google.gson.Gson;
import it.polimi.iol.duque.model.Users;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

/**
 * Created by Duque on 01/05/2017.
 */
public class StatsServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("application/json");

        List<Users> usersList = DBManager.getAllUsers();
        String usersJson = new Gson().toJson(usersList); // Here I'm using Gson, one of the most popular JSON
serializing libraries

        System.out.println(usersJson);

        response.getWriter().print(usersJson);
    }

    @Override
    public void init(ServletConfig config) throws ServletException {
        DBManager.initConnection(config);//Init database connection
        super.init(config);
    }
}
```

```java
package it.polimi.iol.duque;

import com.google.gson.Gson;
import it.polimi.iol.duque.model.Users;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * Created by Duque on 01/05/2017.
 */
public class UsersServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        response.setContentType("application/json");

        String user = request.getParameter("currentUser");
        System.out.println(user);

        Users currentUser = new Gson().fromJson(user, Users.class);
        System.out.println(currentUser);

        DBManager.updateUserPoints(currentUser);
    }

    @Override
    public void init(ServletConfig config) throws ServletException {
        DBManager.initConnection(config);//Init database connection
        super.init(config);
    }
}
```

```java
package it.polimi.iol.duque;


import com.google.gson.Gson;
import it.polimi.iol.duque.model.Events;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;

/**
 * Created by Duque on 29/04/2017.
 */
public class EventsServlet extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("application/json");
        PrintWriter out = response.getWriter();//It writes the response

        List<Events> events = DBManager.getAllEvents();
        String eventsJson = new Gson().toJson(events); // Here I'm using Gson, one of the most popular JSON
serializing libraries

        System.out.println(eventsJson);

        out.print(eventsJson);
    }

    public String getServletInfo() {
        return "Servlet that connects to PostgreSQL database and returns result of a SELECT (in this case a
list of events)";
    }

    @Override
    public void init(ServletConfig config) throws ServletException {
        DBManager.initConnection(config);//Init database connection
        super.init(config);
    }

}
```

File - /Users/duque/MyStuff/PoliMi/WebTecnologies/PrimaProva/campus-explorer/TUDEX/src/it/polimi/iol/duque/model/Users.java

```java
package it.polimi.iol.duque.model;

import java.io.Serializable;

/**
 * Created by Duque on 29/04/2017.
 */
public class Users implements Serializable{
    private int id;
    private String username;
    private String surname;
    private String password;
    private String faculty;
    private Integer points;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getFaculty() {
        return faculty;
    }

    public void setFaculty(String faculty) {
        this.faculty = faculty;
    }

    public Integer getPoints() {
        return points;
    }

    public void setPoints(Integer points) {
        this.points = points;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Users users = (Users) o;

        if (id != users.id) return false;
        if (username != null ? !username.equals(users.username) : users.username != null) return false;
        if (password != null ? !password.equals(users.password) : users.password != null) return false;
        if (faculty != null ? !faculty.equals(users.faculty) : users.faculty != null) return false;
        if (points != null ? !points.equals(users.points) : users.points != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = id;
        result = 31 * result + (username != null ? username.hashCode() : 0);
        result = 31 * result + (password != null ? password.hashCode() : 0);
        result = 31 * result + (faculty != null ? faculty.hashCode() : 0);
        result = 31 * result + (points != null ? points.hashCode() : 0);
        return result;
    }
}
```

```java
package it.polimi.iol.duque.model;

import java.io.Serializable;

/**
 * Created by Duque on 30/04/2017.
 */
public class Events implements Serializable {
    private int id;
    private String type;
    private String title;
    private String description;
    private Integer flames;
    private Double latitude;
    private long startTime;
    private long endTime;
    private Double longitude;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public Integer getFlames() {
        return flames;
    }

    public void setFlames(Integer flames) {
        this.flames = flames;
    }

    public Double getLatitude() {
        return latitude;
    }

    public void setLatitude(Double latitude) {
        this.latitude = latitude;
    }

    public long getStartTime() {
        return startTime;
    }

    public void setStartTime(long startTime) {
        this.startTime = startTime;
    }

    public long getEndTime() {
        return endTime;
    }

    public void setEndTime(long endtime) {
        this.endTime = endtime;
    }

    public Double getLongitude() {
        return longitude;
    }

    public void setLongitude(Double longitude) {
        this.longitude = longitude;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Events events = (Events) o;

        if (id != events.id) return false;
        if (startTime != events.startTime) return false;
        if (endTime != events.endTime) return false;
        if (type != null ? !type.equals(events.type) : events.type != null) return false;
```

```
        if (title != null ? !title.equals(events.title) : events.title != null) return false;
        if (description != null ? !description.equals(events.description) : events.description != null) return
 false;
        if (flames != null ? !flames.equals(events.flames) : events.flames != null) return false;
        if (latitude != null ? !latitude.equals(events.latitude) : events.latitude != null) return false;
        if (longitude != null ? !longitude.equals(events.longitude) : events.longitude != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = id;
        result = 31 * result + (type != null ? type.hashCode() : 0);
        result = 31 * result + (title != null ? title.hashCode() : 0);
        result = 31 * result + (description != null ? description.hashCode() : 0);
        result = 31 * result + (flames != null ? flames.hashCode() : 0);
        result = 31 * result + (latitude != null ? latitude.hashCode() : 0);
        result = 31 * result + (int) (startTime ^ (startTime >>> 32));
        result = 31 * result + (int) (endTime ^ (endTime >>> 32));
        result = 31 * result + (longitude != null ? longitude.hashCode() : 0);
        return result;
    }
}
```

```java
package it.polimi.iol.duque.model;

import java.io.Serializable;

/**
 * Created by Duque on 30/04/2017.
 */
public class EventsPK implements Serializable {
    private String title;
    private long starttime;
    private long endtime;

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public long getStarttime() {
        return starttime;
    }

    public void setStarttime(long starttime) {
        this.starttime = starttime;
    }

    public long getEndTime() {
        return endtime;
    }

    public void setEndTime(long endtime) {
        this.endtime = endtime;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        EventsPK eventsPK = (EventsPK) o;

        if (starttime != eventsPK.starttime) return false;
        if (endtime != eventsPK.endtime) return false;
        if (title != null ? !title.equals(eventsPK.title) : eventsPK.title != null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = title != null ? title.hashCode() : 0;
        result = 31 * result + (int) (starttime ^ (starttime >>> 32));
        result = 31 * result + (int) (endtime ^ (endtime >>> 32));
        return result;
    }
}
```

```
//GLOBAL VARIABLES
let events; // Global variable for events
let stats; // Global variable for stats
let currentUser;// Global variable for currentUser stored in local storage to keep it in other pages

/*This would be my start function, it gets browser's location and starts the application*/
function start() {
    //prepareInsertStatements(events); //This would be the function to update events on DB
    callEventsServlet();

    //In this case I'm using my current position to see the events, i have configured in my local database
    let position = {
        coords: {
            latitude: 45.615887099999995,
            longitude: 9.4587554
        }
    };
    getPosition(position);
    //useGeolocation();//Enable if you want to use browser's geolocation
}

function useGeolocation() {
    console.log("Getting geolocation...");
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(getPosition);
    } else {
        alert("Please enable Geolocation.");
    }
}

function callEventsServlet() {
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", "http://localhost:8080/TUDEX/eventsServlet", false);
    xhttp.setRequestHeader("Content-type", "application/json");
    xhttp.send();

    var response = xhttp.responseText;
    events = JSON.parse(response);
}

function callStatsServlet() {
    var xhttp = new XMLHttpRequest();
    xhttp.open("GET", "http://localhost:8080/TUDEX/statsServlet", false);
    xhttp.setRequestHeader("Content-type", "application/json");
    xhttp.send();

    var response = xhttp.responseText;
    stats = JSON.parse(response);

    currentUser = localStorage.currentUser ? JSON.parse(localStorage.currentUser) : null;
    manageStats(currentUser);
}

function callLoginServlet(username, password) {
    var xhttp = new XMLHttpRequest();
    var params = "?username=" + username + "&password=" + password;

    xhttp.open("POST", "http://localhost:8080/TUDEX/loginServlet" + params, false);
    xhttp.setRequestHeader("Content-type", "application/json");
    xhttp.send();

    var response = xhttp.responseText;
    console.log(response);
    var currentUserJson = JSON.parse(response);

    localStorage.currentUser = JSON.stringify(currentUserJson);

    window.location.href = 'index.html';
}

function updateUserPoints(flamesCount) {
    currentUser.points = flamesCount;

    var userToPost = JSON.stringify(currentUser);

    var xhttp = new XMLHttpRequest();
    var params = "?currentUser=" + userToPost;

    xhttp.open("POST", "http://localhost:8080/TUDEX/usersServlet" + params, false);
    xhttp.setRequestHeader("Content-type", "application/json");
    xhttp.send();

    var response = xhttp.responseText;
    console.log(response);
}

function getPosition(position) {
    console.log("Starting application...");
    currentUser = localStorage.currentUser ? JSON.parse(localStorage.currentUser) : null;

    manageEvents(currentUser, events, position);
}

function getEventTimeSet(startTime, endTime) {
    let startDate = new Date(startTime);
    let endDate = new Date(endTime);

    return normalizeToTwoDigits(startDate.getHours()) + ":" + normalizeToTwoDigits(startDate.getMinutes()) +
" - " + normalizeToTwoDigits(endDate.getHours()) + ":" + normalizeToTwoDigits(endDate.getMinutes());
}
```

```
/* This function normalizes the numbers to two digits adding a zero on the left of the number if < 10, example
: 1 => 01 */
function normalizeToTwoDigits(digit) {
    return digit > 9 ? digit : '0' + digit;
}

/*The haversine formula determines the great-circle distance between two points on a sphere
 given their longitudes and latitudes.*/
function getDistanceFromLatLonInMeters(lat1, lon1, lat2, lon2) {
    const R = 6371 * 1000; // Radius of the earth in meters
    let dLat = deg2rad(lat2 - lat1); // deg2rad below
    let dLon = deg2rad(lon2 - lon1);
    let haversine =
        Math.sin(dLat / 2) * Math.sin(dLat / 2) +
        Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
        Math.sin(dLon / 2) * Math.sin(dLon / 2);
    let c = 2 * Math.atan2(Math.sqrt(haversine), Math.sqrt(1 - haversine));
    let distance = R * c; // Distance in meters
    return distance;
}

function deg2rad(deg) {
    return deg * (Math.PI / 180)
}

function prepareInsertStatements(events) { //Updates DB given an Event list (see events.js for example)
    let insertStm = "";
    events.forEach(function (event) {
        insertStm += "\nINSERT INTO public.events(type, title, description, flames, coordinates, starttime,
endtime) VALUES ( " + "'" + event.type + "'" + ", " + "'" + event.title + "'" + ", " + "'" + event.description
 + "'" + ", " + event.points + "," + "'" + "(" + event.coords.latitude + ", " + event.coords.longitude + ")" +
 "', " + event.startTime + ", " + event.endTime + ");";
    });
    console.log(insertStm);
}
```

```
let events_model = [{
    type: "NOTIFICATION",
    title: "Levon Vincent",
    description: "Deep scurissima da New York City via Berlino. Serata a cura di Le Cannibale.",
    points: 1,
    coords: {
        latitude: 45.61589190000001,
        longitude: 9.4587276
    },
    startTime: 1492856548276,
    endTime: 1492858623573
}, {
    type: "NOTIFICATION",
    title: "Nul: Donna Leake",
    description: "Dj set a base di jazz, afrobeat & other danceable rarities. ",
    points: 1,
    coords: {
        latitude: 45.615900599999996,
        longitude: 9.458768
    },
    startTime: 1492857713573,
    endTime: 1492859723573
}, {
    type: "NOTIFICATION",
    title: "Deftones",
    description: "Chitarre oceaniche con la band di Chino Moreno . ",
    points: 1,
    coords: {
        latitude: 45.61675180000101,
        longitude: 9.4591376
    },
    startTime: 1492857858276,
    endTime: 1492858853573
}, {
    type: "NOTIFICATION",
    title: "La Femme",
    description: "La band francese con il suo sound tra french pop & elettronica anni 80. ",
    points: 1,
    coords: {
        latitude: 69.983269,
        longitude: -49.836534
    },
    startTime: 1492787012,
    endTime: 14923170182
}, {
    type: "NOTIFICATION",
    title: "Incontro con Ira Schneider",
    description: "All'interno della rassegna cinematografica gratuita 'The New American Cinema. Torino 1967',
il Cinema della Fondazione. ",
    points: 1,
    coords: {
        latitude: -35.875294,
        longitude: 92.61492
    },
    startTime: 149278705,
    endTime: 1495317075
}, {
    type: "NOTIFICATION",
    title: "Record Store day",
    description: "Per il decennale del Record Store Day, il negozio apre le porte a mezzanotte +. ",
    points: 1,
    coords: {
        latitude: 45.61589190000001,
        longitude: 9.4587376
    },
    startTime: 1492822713573,
    endTime: 1492827723573
}, {
    type: "NOTIFICATION",
    title: "Ben Rivers - Phantoms",
    description: "la prima personale italiana dell’artista e filmmaker inglese Ben Rivers. A cura di Lucia
Aspesi,. ",
    points: 1,
    coords: {
        latitude: 64.805297,
        longitude: 115.396502
    },
    startTime: 149278702,
    endTime: 1492317072
}, {
    type: "NOTIFICATION",
    title: "Tempo di libri #1",
    description: "La prima edizione della nuova fiera dell'editoria italiana a Milano, in concorrenza con il
Salone. ",
    points: 1,
    coords: {
        latitude: 45.615898,
        longitude: 9.4587439
    },
    startTime: 1492787011,
    endTime: 14913170181
}, {
    type: "NOTIFICATION",
    title: "Nobody's Business Dancing Nobody's Indiscipline",
    description: "Nobodys business è una piattaforma di ricerca aperta ad artisti, teorici, curatori e
organizzatori locali. ",
    points: 1,
    coords: {
        latitude: 30.389528,
        longitude: 2.667512
    },
```

```
    startTime: 149278708,
    endTime: 1498317078
}, {
    type: "NOTIFICATION",
    title: "Paolo Benvegnù + Angela Baraldi",
    description: "Il cantautore presenta il nuovo album, oggi in doppio set con Angela Baraldi. ",
    points: 1,
    coords: {
        latitude: 86.281254,
        longitude: -113.506823
    },
    startTime: 1492821613573,
    endTime: 1492821623573
}, {
    type: "NOTIFICATION",
    title: "q|LAB - Uploading Eden: Sem&Stènn",
    description: "Sem&Stènn, giovani esponenti del synth-pop queer italiano, incontrano q|LAB, one night gay-
friendly tra le più. ",
    points: 1,
    coords: {
        latitude: -56.010318,
        longitude: -78.074596
    },
    startTime: 149278703,
    endTime: 1493317073
}, {
    type: "NOTIFICATION",
    title: "Olympia + More Festival",
    description: "Deep house con Simone De Kunovich e Marco Sala. ",
    points: 1,
    coords: {
        latitude: 7.995643,
        longitude: 68.164961
    },
    startTime: 149278705,
    endTime: 1495317075
}, {
    type: "NOTIFICATION",
    title: "Touch and Go Closing Party : Ralf & Carola Pisaturo",
    description: "Lo zio della house italiana insieme aCarola Pisaturo, che si destreggia in consolle a colpi
. ",
    points: 1,
    coords: {
        latitude: 45.61675181000101,
        longitude: 9.4591375
    },
    startTime: 1492857849276,
    endTime: 1492858854573
}, {
    type: "NOTIFICATION",
    title: "4cento Garden Party",
    description: "Deep house & afro house, in consolle Fede Beige e G.dots. ",
    points: 1,
    coords: {
        latitude: 45.615898,
        longitude: 9.4787439
    },
    startTime: 1492787012,
    endTime: 14923170182
}, {
    type: "NOTIFICATION",
    title: "Hood 129",
    description: "Una notte che porta a Milano la UK garage con sfumature house e night bass,. ",
    points: 1,
    coords: {
        latitude: 45.61689290010001,
        longitude: 9.4587276
    },
    startTime: 1492787021,
    endTime: 14913170281
}, {
    type: "NOTIFICATION",
    title: "Virgo",
    description: "Una nuova serata dedicata al mondo femminile a base di disco, funky, house, pop oriented. ",
    points: 1,
    coords: {
        latitude: 45.61675190010001,
        longitude: 9.4587276
    },
    startTime: 1492856848276,
    endTime: 1492858633573
}, {
    type: "NOTIFICATION",
    title: "La Valse / Symphony In C / Shéhérazade",
    description: "Con il Corpo di Ballo e Orchestra del Teatro alla Scala, musiche di Ravel, Bizet,. ",
    points: 1,
    coords: {
        latitude: 45.61589190000001,
        longitude: 9.268304
    },
    startTime: 149278702,
    endTime: 1492317072
}, {
    type: "NOTIFICATION",
    title: "Narciso Contreras - Libya - A Human Marketplace",
    description: "In mostra le opere del vincitore della settima edizione del Prix Carmignac di
fotogiornalismo. Inaugurazione. ",
    points: 1,
    coords: {
        latitude: 45.61675280000101,
        longitude: 9.4591386
```

```
    },
    startTime: 1492857758276,
    endTime: 1492858873573
}, {
    type: "BONUS",
    title: "Hidden Treasure Found: ",
    description: "",
    points: 20,
    coords: {
        latitude: 45.61589190000001,
        longitude: 9.4587276
    },
    startTime: 1492858623573,
    endTime: 1492859623573
}, {
    type: "NOTIFICATION",
    title: "Brian Calvin e Wendy White",
    description: "Mostra a cura di Maria Chiara Valacchi. Inaugurazione 21 aprile h. 18:30. Aperto dal
mercoledì. ",
    points: 1,
    coords: {
        latitude: 45.61675180000101,
        longitude: 9.4591376
    },
    startTime: 1492857858276,
    endTime: 1492858853573
}, {
    type: "NOTIFICATION",
    title: "Rollower: Bell Towers",
    description: "Tra italo-disco e techno dj australiano di base a Londra ora accolto anche da Public. ",
    points: 1,
    coords: {
        latitude: 45.61675190110001,
        longitude: 9.4591376
    },
    startTime: 1492856858276,
    endTime: 1492858653573
}, {
    type: "BONUS",
    title: "Hidden Treasure found: ",
    description: "",
    points: 51,
    coords: {
        latitude: 45.61589190000001,
        longitude: 9.4587276
    },
    startTime: 1492858623573,
    endTime: 1492868427573
}, {
    type: "NOTIFICATION",
    title: "Magnolia in da house: The Magician",
    description: "Il Re Mida dell'elettronica. In apertura Valentina Sartorio. ",
    points: 1,
    coords: {
        latitude: 45.61675181000101,
        longitude: 9.4591276
    },
    startTime: 1492857958276,
    endTime: 1492858953573
}, {
    type: "BONUS",
    title: "Hidden Treasure Found: ",
    description: "",
    points: 61,
    coords: {
        latitude: 45.615896899999996,
        longitude: 9.55873
    },
    startTime: 1492857623573,
    endTime: 1492878723573
}, {
    type: "BONUS",
    title: "Hidden Treasure Found: ",
    description: "",
    points: 11,
    coords: {
        latitude: 45.61675180000101,
        longitude: 9.4591376
    },
    startTime: 1492868729573,
    endTime: 1492959622573
}];
```

```css
html {
    height: 100%;
}

header {
    display: flex;
    flex-direction: row;
    background-color: #00A6D6;
}

body {
    font-family: Arial, sans-serif;
    background-color: #f7f7f7;
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
}

.content {
    flex: 1;
    background-color: #f7f7f7;
    overflow: auto;
}


/****************** ************** ******************/
/********************** Badge Box **********************/
/****************** ************** ******************/

.badge_box {
    position: relative;
    box-shadow: 4px 4px 8px 4px #888888;
    background: #fff;
    color: #00A6D6;
    margin: 1.5em 1em;
    padding: 1em;
    font-size: x-large;
}

.badge_head {
    border-bottom: 1px solid gray;
    font-weight: bold;
    padding-bottom: 0.2em;
    margin-bottom: 0.2em;
}

.badge_body {
    margin: auto;
    color: #004b61;
    padding-bottom: 0.1em;
}

.badge_footer {
    margin: auto;
    display: flex;
    flex-direction: row;
    background: rgb(0, 40, 51);
    font-weight: bold;
    color: white;
}

.bonus-div {
    display: flex;
    flex-direction: row;
    background: rgb(0, 40, 51);
    box-shadow: 4px 4px 8px 4px #888888;
    color: #fed952;
    margin: 1.5em 3em;
    padding: 1em;
    font-size: xx-large;
    font-weight: bold;

}


.badge_footer_left {
    text-align: center;
}

.distance_image {}

.clock_image {}

.badge_footer_right {
    text-align: center;
}


/****************** ************** ******************/
/****************** Image Text Div ******************/
/****************** ************** ******************/

.img_txt_container {
    margin: auto;
    display: flex;
    flex-direction: row;
    font-size: xx-large;
}
```

```
.img_txt_container_image {
    padding: 0.2em;
}

.img_txt_container_text {
    padding: 0.2em;
    flex: 1;
    text-align: center;
}

.logo {
    padding: 2em;
    width: 20%;
}

.centered_text {
    padding-top: 1em;
    padding-bottom: 1em;
    font-size: 3em;
    color: white;
    text-align: center;
    flex: 1;
}


/******************** *************** ********************/
/*********************** Footer ***********************/
/******************** *************** ********************/

footer {
    display: flex;
    flex-direction: row;
    border-top: 8px solid #f7f7f7;
}

.footer-text {
    color: #fed952;
    background: rgb(0, 40, 51);
    display: flex;
    font-size: xx-large;
    font-weight: bold;
    flex: 1;
}

.arrow-button-right {
    padding: 1em;
    border-left: 0.4em solid white;
    background-color: #00A6D6;
}

.arrow-button-right:active {
    background-color: rgb(40, 119, 142);
    box-shadow: 2px 2px 2px 2px #666;
    transform: translateX(4px);
}

.arrow-button-left {
    padding: 1em;
    border-right: 0.4em solid white;
    background-color: #00A6D6;
}

.arrow-button-left:active {
    background-color: rgb(40, 119, 142);
    box-shadow: 2px 2px 2px 2px #666;
    transform: translateX(4px);
}

.arrow-right {
    border-top: 30px solid transparent;
    border-bottom: 30px solid transparent;
    border-left: 40px solid white;
    margin-right: 2em;
    margin-left: 2em;
}

.arrow-left {
    border-top: 30px solid transparent;
    border-bottom: 30px solid transparent;
    border-right: 40px solid white;
    margin-right: 2em;
    margin-left: 2em;
}

/******************** *************** ********************/
/*********************** Stats Box ***********************/
/******************** *************** ********************/

.stats_box {
    position: relative;
    box-shadow: 4px 4px 8px 4px #888888;
    background: #002833;
    color: #00A6D6;
    margin: 1.5em 1em;
    padding: 1em;
    font-size: xx-large;
}

.stats_head {
    border-bottom: 4px solid #fed952;
```

```
    font-weight: bold;
    padding-bottom: 0.2em;
    margin-bottom: 0.2em;
}

.stats_container {
    position: relative;
    background: #002833;
    padding: 2em;
    font-size: x-large;
    height: 100%;
    display: grid;
}

.double_txt_img_container {
    width: 100%;
    margin: auto;
    display: flex;
    flex-direction: row;
    font-size: xx-large;
}

.double_txt2{
    padding: 0.2em;
    text-align: center;
}

.double_txt1{
    padding: 0.2em;
    flex: 1;
}

/******************* *************** *******************/
/********************** Login Box **********************/
/******************* *************** *******************/
.login_head {
    border-bottom: 4px solid rgb(0, 40, 51);
    color: rgb(0, 40, 51);
    font-weight: bold;
    font-size: xx-large;
    padding: 1em;
}

.login_box {
    position: relative;
    box-shadow: 4px 4px 8px 4px #888888;
    background: #fff;
    color: #00A6D6;
    margin: 50% 1em;
    padding: 1em;
    font-size: x-large;
}

.login_input_container {
    margin: auto;
    display: flex;
    flex-direction: row;
    font-size: xx-large;
    padding: 1.5em;
}

.login_input_txt {
    padding: 0.2em;
}

.login_input {
    flex: 1;
    text-align: center;
    height: 100%;
    font-size: xx-large;
    color: #00a6d6;
    border-width: 1px;
    border-style: dashed;
    border-color: initial;
    border-image: initial;
}

.login_button {
    box-shadow: 0px 4px 4px 2px rgb(0, 166, 214);
    background: rgb(0, 40, 51);
    font-weight: bold;
    color: white;
    text-align: center;
    padding: 10px;
}

.login_button:active {
    background-color: rgb(0, 166, 214);
    transform: translateY(4px);
}

.login_footer{
    color: #fed952;
    background: rgb(0, 40, 51);
    font-weight: bold;
    width: 100%;
    padding-bottom: 10%;
}
```

```
<!DOCTYPE html>
<html>

<head>
    <link rel="stylesheet" href="style.css"/>
    <title>TUDEX (TUDelft Campus Explorer)</title>
    <script type="text/javascript" src="events.js"></script>
    <script type="text/javascript" src="utils.js"></script>
    <script type="text/javascript" src="eventManager.js"></script>

</head>

<body>
<header>
    <img src="https://www.tudelft.nl/typo3conf/ext/tud_styling/Resources/Public/img/logo.svg" alt="logo" class
="logo"/>
    <div class="centered_text">Campus Explorer</div>
    <!-- I'm putting the following empty_div to balance the flex display distribution, in order to keep the
header responsive -->
    <div id="emty_div" class="logo"/>
</header>

<div class="content"></div>

<footer>
    <div class="footer-text"></div>
    <div class="arrow-button-right">
        <div class="arrow-right" onclick="window.location.href='stats.html';"></div>
    </div>
</footer>
</body>

<!--I'm calling this script at the end to ensure the qhole document has already been loaded-->
<script type="text/javascript">
    start();
</script>

</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="style.css"/>
    <script type="text/javascript" src="events.js"></script>
    <script type="text/javascript" src="utils.js"></script>
    <script type="text/javascript" src="statsManager.js"></script>
    <script type="text/javascript" src="eventManager.js"></script>
    <script type="text/javascript" src="loginManager.js"></script>
</head>

<body>
<header>
    <img src="https://www.tudelft.nl/typo3conf/ext/tud_styling/Resources/Public/img/logo.svg" alt="logo" class
="logo"/>
    <div class="centered_text">Campus Explorer</div>
    <!-- I'm putting the following empty_div to balance the flex display distribution, in order to keep the
header responsive -->
    <div id="emty_div" class="logo"/>
</header>

<div class="content"></div>

<footer>
    <div class="login_footer"></div>
</footer>
</body>

<!--I'm calling this script at the end to ensure the qhole document has already been loaded-->
<script type="text/javascript">
    initLogin();
</script>

</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Stats</title>
    <link rel="stylesheet" href="style.css"/>
    <script type="text/javascript" src="events.js"></script>
    <script type="text/javascript" src="utils.js"></script>
    <script type="text/javascript" src="statsManager.js"></script>
    <script type="text/javascript" src="eventManager.js"></script>
</head>

<body>
<header>
    <img src="https://www.tudelft.nl/typo3conf/ext/tud_styling/Resources/Public/img/logo.svg" alt="logo" class
="logo"/>
    <div class="centered_text">Campus Explorer</div>
    <!-- I'm putting the following empty_div to balance the flex display distribution, in order to keep the
header responsive -->
    <div id="emty_div" class="logo"/>
</header>

<div class="content"></div>

<footer>
    <div class="arrow-button-left">
        <div class="arrow-left" onclick="window.location.href='index.html';"></div>
    </div>
    <div class="footer-text"></div>
</footer>
</body>

<!--I'm calling this script at the end to ensure the qhole document has already been loaded-->
<script type="text/javascript">
    callStatsServlet();
</script>
</html>
```

```
/**
 * Created by Duque on 28/04/2017.
 */
function manageEvents(currentUser, events, position) {
    let flamesCount = currentUser ? currentUser.points : 0;

    events.forEach(function(event) {
        const METERS_100 = 100;

        let distance = getDistanceFromLatLonInMeters(event.latitude, event.longitude, position.coords.latitude
, position.coords.longitude);

        /*//Debugging stuff
        console.log(event);
        console.log(position);
        console.log(event.title + " -> " + getEventTimeSet(event.startTime, event.endTime) + " # " + distance
);*/

        if(event.type ==="BONUS"){
            console.error(event.title + " -> " + getEventTimeSet(event.startTime, event.endTime) + " # " +
distance);
            if (distance <= METERS_100 && isEventInADaySlot(event)) {
                appenDivToContent(createBonusDiv(event));
                flamesCount = flamesCount + event.flames;
            }
        }

        if(event.type ==="NOTIFICATION") {
            if (distance <= METERS_100 && isEventInAnHourSlot(event)) {
                let mtsDistance = parseInt(distance) + ' meters'; //gets the integer value and concats meters
                appenDivToContent(createBagdeDiv(event, mtsDistance));
                flamesCount = flamesCount + event.flames;
            }
        }
    });

    let flamesFooterDiv = createTxtImgDiv('images/fire_1.png', 'Your Flame count: ' + flamesCount);

    updateUserPoints(flamesCount);
    document.body.getElementsByClassName('footer-text')[0].appendChild(flamesFooterDiv);

}

function isEventInAnHourSlot(event) {
    const HOUR_IN_MILIS = 1000 * 60 * 60; //I'm using a constant here since an Hour will be always an hour :)

    let nowTimestamp = 1492858623573; //TODO go back to Date.now(); if you wanna work with dinamic time
    let fromNowToStart = event.startTime - nowTimestamp;

    return (nowTimestamp >= event.startTime && nowTimestamp <= event.endTime) || //Event has already started
but hasn't finished yet OR
        (nowTimestamp < event.startTime && fromNowToStart >= 0 && fromNowToStart <= HOUR_IN_MILIS); //Event
will start in less than an hour
}

function isEventInADaySlot(event) {
    const DAY_IN_MILIS = 1000 * 60 * 60 *24;

    let nowTimestamp = 1492858623573; //TODO go back to Date.now(); if you want to work with dinamic date
    let fromNowToStart = event.startTime - nowTimestamp;

    return (nowTimestamp >= event.startTime && nowTimestamp <= event.endTime) || //Event has already started
but hasn't finished yet OR
        (nowTimestamp < event.startTime && fromNowToStart >= 0 && fromNowToStart <= DAY_IN_MILIS); //Event
will start in less than an hour
}

function createBonusDiv(event) {
    let bonusDiv = document.createElement('div');
    bonusDiv.className = 'badge_box';

    let bonusImgDiv = createImgTxtDiv('images/fire_1.png', event.title + " "+event.flames+'+ Flames!');
    bonusImgDiv.appendChild(createImgDiv('images/fire_1.png'));

    bonusImgDiv.className = 'bonus-div';
    bonusDiv.appendChild(bonusImgDiv);

    return bonusImgDiv;
}


function createBagdeDiv(event, distance) {
    let badgeDiv = document.createElement('div');
    badgeDiv.className = 'badge_box';

    //Badge Header
    badgeDiv.appendChild(createRowDiv('badge_head', event.title));

    //Badge Body
    badgeDiv.appendChild(createRowDiv('badge_body', event.description));

    //Badge Footer
    let badgeFooter = createRowDiv('badge_footer');
    let eventTimeSet = getEventTimeSet(event.startTime, event.endTime);

    let footerLeft = createImgTxtDiv('images/clock.png', eventTimeSet);
    badgeFooter.appendChild(footerLeft);

    let footerRight = createImgTxtDiv('images/distance.png', distance);
    badgeFooter.appendChild(footerRight);
```

```
    badgeDiv.appendChild(badgeFooter);
    return badgeDiv;
}


function createRowDiv(className, innerHtml) {
    let badgeRowDiv = document.createElement('div');
    badgeRowDiv.className = className ? className : 'badge_body';
    badgeRowDiv.innerHTML = innerHtml ? innerHtml : '';

    return badgeRowDiv;
}

function createTxtImgDiv(img, txt) {
    let containerDiv = document.createElement('div');
    containerDiv.className = 'img_txt_container';
    containerDiv.appendChild(createTxtDiv(txt));
    containerDiv.appendChild(createImgDiv(img));

    return containerDiv;
}

function createImgTxtDiv(img, txt) {
    let containerDiv = document.createElement('div');
    containerDiv.className = 'img_txt_container';
    containerDiv.appendChild(createImgDiv(img));
    containerDiv.appendChild(createTxtDiv(txt));

    return containerDiv;
}

function createImgDiv(img) {
    let imgDiv = document.createElement('img');
    imgDiv.className = 'img_txt_container_image';
    imgDiv.src = img;

    return imgDiv;
}

function createTxtDiv(txt, className) {
    let txtDiv = document.createElement('div');
    txtDiv.className = className ? className : 'img_txt_container_text';
    txtDiv.innerHTML = txt;

    return txtDiv;
}


function appenDivToContent(newDiv) {
    document.body.getElementsByClassName('content')[0].appendChild(newDiv);
}
```

```
/**
 * Created by Duque on 01/05/2017.
 */
function initLogin() {
    appenDivToContent(createLoginBox());
}

function createLoginBox() {
    let badgeDiv = document.createElement('div');
    badgeDiv.className = 'login_box';

    //Badge Header
    badgeDiv.appendChild(createRowDiv('login_head', "Login"));

    //Badge Body
    let usernameInput = createInputRowDiv("usernameInput", 'text', "Username:");
    badgeDiv.appendChild(usernameInput);

    let passInput = createInputRowDiv('passInput', 'password', "Password:");
    badgeDiv.appendChild(passInput);

    let submitButton = createRowDiv('login_button', "Login");
    submitButton.onclick = function () {
        let usernameVal = document.getElementById('usernameInput').value;
        let passVal = document.getElementById('passInput').value;

        console.log(usernameVal);
        console.log(passVal);

        callLoginServlet(usernameVal, passVal);
    };

    badgeDiv.appendChild(submitButton);

    return badgeDiv;
}

function createInputRowDiv(id, type, name) {
    let containerDiv = document.createElement('div');
    containerDiv.className = 'login_input_container';
    containerDiv.appendChild(createTxtDiv(name, 'login_input_txt'));
    containerDiv.appendChild(createInput(id, type, name, 'login_input'));

    return containerDiv;
}

function createInput(id, type, name, className) {
    let input = document.createElement('input');
    input.className = className ? className : 'login_input';
    input.type = type;
    input.id = id;
    input.name = name;

    return input;
}
```

```
/**
 * Created by Duque on 01/05/2017.
 */
function manageStats(currentUser) {
    sortStatsByUserPoints();

    let tudexBox = createStatsBox(stats, "Tudelf");
    appenDivToContent(tudexBox);

    let currentUserFaculty = currentUser.faculty;

    let facultyStats = stats.filter(function(stat) {
        return stat.faculty === currentUserFaculty;
    });

    let facultyBox = createStatsBox(facultyStats, currentUserFaculty);
    appenDivToContent(facultyBox);

    let flamesFooterDiv = createTxtImgDiv('images/fire_1.png', 'Top Campus Explorers');
    document.body.getElementsByClassName('footer-text')[0].appendChild(flamesFooterDiv);
}

let createStatsBox = function (stats, boxTitle) {
    let i = 1;

    let badgeBox = document.createElement('div');
    badgeBox.className = 'stats_box';

    //Badge Header
    badgeBox.appendChild(createRowDiv('stats_head', boxTitle));

    //Stats Container
    let statsContainer = document.createElement('div');
    statsContainer.className = 'stats_container';

    stats.forEach(function (stat) {
        let statsText = i + '.  ' + stat.username + ' ' + stat.surname;
        let statDiv = createDoubleTxtImgDiv('images/fire_1.png', statsText, stat.points);

        //In this way i highlight the current user
        if (stat.username === currentUser.username) {
            statDiv.style.color = "#fed952";
            statDiv.style.fontWeight = "bold";
        }

        statsContainer.appendChild(statDiv);
        i++;
    });

    badgeBox.appendChild(statsContainer);

    return badgeBox;
};

function createDoubleTxtImgDiv(img, txt1, txt2) {
    let containerDiv = document.createElement('div');
    containerDiv.className = 'double_txt_img_container';
    containerDiv.appendChild(createRowDiv('double_txt1',txt1));
    containerDiv.appendChild(createRowDiv('double_txt2',txt2));
    containerDiv.appendChild(createImgDiv(img));

    return containerDiv;
}

function sortStatsByUserPoints() {
    stats.sort(compareUserPoints);
}

function compareUserPoints(a, b) {
    if (a.points > b.points)
        return -1;
    if (a.points < b.points)
        return 1;
    return 0;
}
```