Instalación de PostgreSQL en nuestra máquina

Antes de usar PostgreSQL explicaremos como se instala. Si esta instalación la vamos a realizar en las máquinas virtuales facilitadas en la asignatura primeramente deberemos instalar "Windows Installer4.5" el cual podremos encontrar en el siguiente enlace: https://support.microsoft.com/en-us/kb/893803, concretamente deberemos elegir "WindowsServer2003-KB942288-v4-x64.exe".

Una vez esté instalado deberemos descargar e instalar PostgreSQL el cual incluye pgAdmin, una interfaz gráfica que nos ha resultado muy útil para entender mejor la base de datos pero que no será necesaria en ningún momento ya que todos los comandos se ejecutaran mediante la terminal. Para descargarlo deberemos acceder al siguiente enlace http://www.enterprisedb.com/products-services-training/pgdownload#windows y descargar la versión más actualizada del 9.4 de 32 bits ya que ambas limitaciones nos las marca la librería que debemos usar.

La ejecución del instalador debemos realizarla desde el usuario administrador (Boss) y, durante la misma, nos solicitará diferentes datos como el usuario del administrador de la base de datos (postgres), la contraseña (en nuestro caso la misma que para el root de MYSQL: V3rY=$tR0nG=P@$$w0rd$), el puerto a usar para conectarse a la base de datos (5432) y los locales (Spanish-Spain). Tras finalizar la instalación nos preguntará si deseamos lanzar el "Stack Builder" a lo que diremos que no ya que tenemos instalado todo lo necesario. Con esto estaría instalada la base de datos PostgreSQL. Es recomendable reiniciar tras la instalación.

Configuración PostgreSQL en nuestra máquina

Al reiniciar la máquina lanzaremos la terminal de PostgreSQL para crear la base de datos del proyecto. La terminal se encuentra situada en Inicio > Todos los programas > PostgreSQL > SQL Shell (psql). Al ejecutarla nos solicitará determinados datos. En caso de querer introducir los que están entre corchetes solo es necesario pulsar enter. Los datos con los que debemos acceder son:

- Server: localhost
- Database: postgres
- Port: 5432
- Username: postgres
- Password: V3rY=$tR0nG=P@$$w0rd$

Una vez logueados proederemos a ejecutar los siguientes comando para la creación de los usuarios y de la base de datos:

```
create user "acme-user" password 'ACME-Us3r-P@ssw0rd';

create user "acme-manager" password 'ACME-M@n@ger-6874';

drop database if exists "Acme-Orienteering";

create database "Acme-Orienteering" owner "acme-manager";

grant all on database  "Acme-Orienteering" to "acme-user";

grant all on database  "Acme-Orienteering" to postgres;
```

Tras esto deberemos conectarnos a la base de datos con los siguientes datos:

- Server: localhost
- Database: Acme-Orienteering
- Port: 5432
- Username: postgres
- Password: V3rY=$tR0nG=P@$$w0rd$

Y ejecutar los siguientes comandos:

```
alter schema public owner to "acme-manager";
```

Una vez realizado esto ya tendremos totalmente configurada la base de datos en nuestro sistema para lanzar el proyecto o los test.

Cambios realizados a nivel de aplicación

Para hacer posible que la base de datos funcionará se tuvieron que hacer los siguientes cambios en nuestro o proyecto (o cualquier otro).

Primeramente se tuvo que modificar el archivo "pom.xml" alojado en la raíz del proyecto. En este se debe añadir la dependencia a la librería de postgres necesaria para que funcione correctamente la aplicación. Las líneas a añadir son las siguientes (preferiblemente como la última dependencia):

```
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.26</version>
</dependency>

<!-- Hibernate -->
```

```
                <artifactId>mysql-connector-java</artifactId>
                <version>5.1.26</version>
        </dependency>
+
+       <!-- PostgreSql -->
+       <dependency>
+               <groupId>org.postgresql</groupId>
+               <artifactId>postgresql</artifactId>
+               <version>9.4.1208.jre7</version>
+       </dependency>

        <!-- Hibernate -->
```

Otra configuración de carácter general que se tuvo que editar fue el archivo persistence.xml alojado en /src/main/resources/META-INF y en target/classes/META-INF para indicar como debe conectarse e interactuar con la base de datos. La configuración es la siguiente:

```
<!--
<provider>org.hibernate.ejb.HibernatePersistence</provider> -->

        <properties>
-               <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />
-               <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/Acme-Orienteering" />
                <property name="javax.persistence.jdbc.user"
value="acme-manager" />
                <property name="javax.persistence.jdbc.password"
value="ACME-M@n@ger-6874" />

-               <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQLDialect" />
                <!-- <property name="hibernate.hbm2ddl.auto"
value="none" /> -->
        </properties>
```

```
<!--
<provider>org.hibernate.ejb.HibernatePersistence</provider> -->

        <properties>
+               <property name="javax.persistence.jdbc.driver"
value="org.postgresql.Driver" />
+               <property name="javax.persistence.jdbc.url"
value="jdbc:postgresql://localhost:5432/Acme-Orienteering" />
                <property name="javax.persistence.jdbc.user"
value="acme-manager" />
                <property name="javax.persistence.jdbc.password"
value="ACME-M@n@ger-6874" />

+               <property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect" />
                <!-- <property name="hibernate.hbm2ddl.auto"
value="none" /> -->
        </properties>
```

Al igual que el paso anterior se debe configurar la manera en la que spring se conecta a la base de datos. El archivo a modificar es data.xml alojado en src/main/resources/spring/config y en target/classes/META-INF debiendose dejar de la siguiente manera:

```
-		<property name="driverClass" value="com.mysql.jdbc.Driver" />
-		<property name="jdbcUrl" value="jdbc:mysql://localhost:3306/Acme-Orienteering" />
+		<property name="driverClass" value="org.postgresql.Driver" />
+		<property name="jdbcUrl" value="jdbc:postgresql://localhost:5432/Acme-Orienteering" />
-		<constructor-arg value="org.hibernate.dialect.MySQLDialect" />
+		<constructor-arg value="org.hibernate.dialect.PostgreSQLDialect" />
```

Para poder lanzar la herramienta PopulateDatabase.java también se tuvo que editar el archivo DatabaseUtil.java localizado en src/main/java/utilities/internal dejándose de la siguiente manera:

```
-		databaseScript.add(String.format("drop database `%s`", databaseName));
+		databaseScript.add(String.format("drop schema public cascade"));
-		databaseScript.add(String.format("create database `%s`", databaseName));
+		databaseScript.add(String.format("create schema public"));
-
+
-		schemaScript.add(String.format("use `%s`", databaseName));
+		schemaScript.add(String.format("grant select, insert, update, delete on all tables in schema public to \"acme-user\""));
+		schemaScript.add(String.format("grant all on schema public to \"acme-user\""));
```

Esta modificación se debe al funcionamiento de PostgreSQL. Primeramente no se puede borrar una base de datos mientras se está conectado por lo que se deberá borrar el schema que es donde se alojan todas las tablas de la base de datos lo que para nosotros significa lo mismo. Otro problema es que en PostgreSQL cada usuario tiene distintos permisos para cada tabla por lo que los mismos deben asignarse tras la creación de las mismas.

Por último, para que la aplicación funcionará perfectamente se tuvo que modificar la anotación @Lob utilizadas en las clases de tipo domain con la cual indicábamos que era un texto largo por las anotaciones @Column(length=10485760) y @Size(max=10485760) para que siga permitiendo un texto bastante grande.