Ejercicio SWT

Diana Angelica Gamez Diaz

October 28, 2016

1 Base de Datos

1.1 Esquema

De acuerdo a la base de datos proporcionada, el modelo entidad-relacion, que servira como referencia para realizar las consultas establecidas, se muestra en la Figura 1.

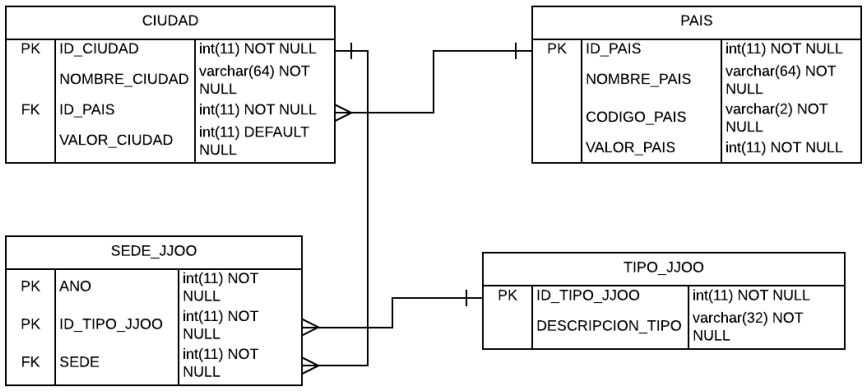

Figure 1: Modelo entidad-relacion

Se identifican relaciones 1:m entre las tablas de $PAIS$ y $CIUDAD$ asi como tambien $CIUDAD$ y $SEDE\_JJOO$ al igual que entre $TIPO\_JJOO$ y $SEDE\_JJOO$.

1.2 Consulta

Una vez que la estructura y las relaciones de la base de datos fueron establecidas, de acuerdo al script proporcionado se ingresan registros e informacion a las tablas para poder realizar consultas.

```
SELECT
P.ID_PAIS AS ID_PAIS, P.NOMBRE_PAIS AS PAIS,
C.ID_CIUDAD AS ID_CIUDAD, C.NOMBRE_CIUDAD AS CIUDAD,
    CASE WHEN C.VALOR_CIUDAD IS NULL THEN P.VALOR_PAIS ELSE C.VALOR_CIUDAD END AS VALOR,
T.DESCRIPCION_TIPO AS DESCRIPCION,
COUNT(S.SEDE) AS NUMERO_SEDE
FROM CIUDAD AS C
JOIN PAIS AS P ON C.ID_PAIS = P.ID_PAIS
LEFT JOIN SEDE_JJOO AS S ON C.ID_CIUDAD = S.SEDE
LEFT JOIN TIPO_JJOO AS T ON S.ID_TIPO_JJOO = T.ID_TIPO_JJOO
```

```
GROUP BY C.ID_CIUDAD;
```

El resultado de la consulta se muestra en la Figura 2.

ID_PAIS	PAIS	ID_CIUDAD	CIUDAD	VALOR	DESCRIPCION	NUMERO_SEDE
1	ESPAÑA	1	LA CORUÑA	93	*(NULL)*	0
1	ESPAÑA	2	MADRID	100	*(NULL)*	0
1	ESPAÑA	3	BARCELONA	124	VERANO	1
2	PORTUGAL	4	LISBOA	134	*(NULL)*	0
2	PORTUGAL	5	OPORTO	200	*(NULL)*	0
2	PORTUGAL	6	COIMBRA	200	*(NULL)*	0
3	FRANCIA	7	CHAMONIX	123	INVIERNO	1
3	FRANCIA	8	PARÍS	5	VERANO	2
3	FRANCIA	9	NIZA	50	*(NULL)*	0
4	ITALIA	10	MILÁN	135	*(NULL)*	0
4	ITALIA	11	ROMA	125	VERANO	1
4	ITALIA	12	TURÍN	190	INVIERNO	1

Figure 2: Consulta

2 Estructura y desarrollo de la aplicación

Al conectarse con la base de datos, es necesario implementar un modelo el cual represente el objeto a obtener. Una vez que este modelo ha sido desarrollado, es entonces cuando las consultas basicas como $Create, Read, Update, Delete$ pueden ser declaradas en el DAO para acceder a la informacion y obtener los registros a manera de objetos.

Para el desarrollo del proyecto, se utilizo la estructura que se muestra en la Figura 3.

La clase principal es *App.java* donde se llevan a cabo operaciones para mostrar la pantalla principal del programa y la conexión con las pantallas para mostrar la consulta *CiudadPantalla.java*, mostrar la lista de sedes *SedePantalla.java* y las encargadas de crear y editar los registros *Crear.java* y *Editar.java*.

Estas clases utilizan SWT para dibujar los elementos que mostrarán la información obtenida en las consultas definidas en las clases *DAO*.

Dentro del paquete *com.diana.dao* se encuentran las clases encargadas de realizar operaciones de consulta e interacción con la base de datos. Cada una de las tablas contiene su propia clase DAO en donde se obtienen los objetos que seran utilizados para las operaciones CRUD.Para identificar cada una de las funciones, se ha comentado el código.

Seguido a esto, el paquete *com.diana.model* contiene los modelos *JPA* que representan la estructura de la base de datos. En estas clases también se encuentran las relaciones entre cada una de las tablas de acuerdo al modelo mostrado en la Figura 1.

Al utilizar *Hibernate*, es necesario desarrollar un archivo que pueda realizar el mapeo entre las variables que se encuentran en los modelos y los nombres de las columnas en cada una de las tablas de la base de datos. Estos archivos, a manera de configuración de Hibernate y en formato xml, se encuentran bajo el paquete *com.diana.util*. De igual manera, el archivo con las configuraciones necesarias para realizar la conexión con la base de datos se encuentra contenido en este paquete.

Las pruebas realizadas con JUnit para comprobar que la consulta en DAO es correcta, se encuntran en la clase contenida en el paquete de *test*. Estas pruebas son las siguientes:

- Comprobar la funcion *getCiudades* con los campos definidos en la consulta
- Comprobar que la ciudad *Paris* ha sido dos veces sede

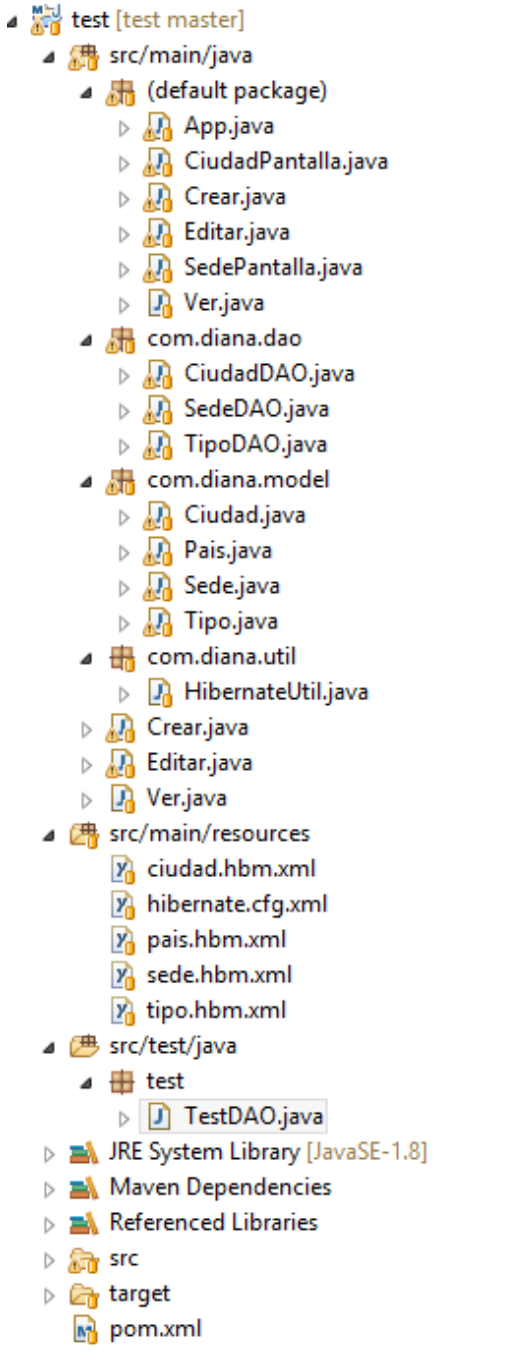

Figure 3: Estructura del proyecto

Finalmente, el archivo *pom.xml* contiene las dependencias administradas por *Maven* para el desarrollo del proyecto.

3 Despliegue y configuraciones de la aplicación

Inicialmente, se debe agregar la libreria de *SWT* proporcionada por Eclipse. Para lograr esto, es necesario importarla al proyecto como una variable. En el explorador del proyecto, sobre la carpeta que contiene los archivos se seleccionan las propiedades. Seguido a esto, se abre una pantalla con las configuraciones del proyecto, dentro de esta, en la pestaña de librerias se agrega una variable que hará referencia a el jar que contiene las funciones de SWT requeridas para el proyecto. El nombre de la variable deberá ser SWT_SRC y el archivo se encuentra dentro de la carpeta *plugins* en el directorio de instalación de Eclipse. El archivo se llama $org.eclipse.swt.win32.win32[version]$. Una vez que esta variable sea agregada al proyecto, las pantallas que involucran la libreria SWT podrán ser desplegadas.

Para el despliegue de la aplicación, es necesario configurar los parámetros de Hibernate para así, realizar la conexión a la base de datos. Estas opciones se encuentran dentro del archivo $hibernate.cfg.xml$. En este archivo, las siguientes líneas deben ser modificadas con las credenciales de autentificación y nombre de la base de datos.

```
<!-- Information about the database and mysql credentials, this should be modified -->
        <property name="connection.url">jdbc:mysql://[servidor]:[puerto]/[nombreBD]</property>
        <property name="connection.username">[nombreUsuario]</property>
        <property name="connection.password">[password]</property>
```

La clase principal es *App.java* ubicada en *src/main/java*. Al compilar la aplicación, se deberá elegir esta clase.

4 Trabajo a futuro

Con la finalidad de realizar la aplicación como Web Service, se recomienda utilizar el framework Spring, acompañado de Hibernate. Spring cuenta con una documentación extensa así como también módulos para creación de aplicaciones que siguen el modelo MVC, además de módulos de seguridad y pruebas.

Al utilizar Maven, se recomienda crear un proyecto con los modelos y clases DAO para, posteriormente añadirlo como dependencia y trabajar en un proyecto independiente para el desarrollo de la aplicación web.