

SPEED PROJECT

Realizado por:
Antonio Calzado
Javier Gomez
Alberto Adame

Curso: 2º DAW

INDICE

- Script creación de tablas
- Campos que se reservan para los datos
- Clases y métodos Java(Comentados)
- Enlace de github con video y todos los demás campos

CREACION DE TABLAS

```
CREATE TABLE ZAPATILLA(  
  ID NUMERIC(5),  
  NAME VARCHAR(100),  
  PRICE FLOAT(5,2),  
  SIZES NUMERIC(2),  
  RELEASEDATE DATE,  
  STOCK BOOLEAN,  
  CONSTRAINT PK_ZAPATILLA PRIMARY KEY(ID)  
);  
  
CREATE TABLE USUARIO(  
  USERNAME VARCHAR(50),  
  PASSWORDD VARCHAR(50),  
  CONSTRAINT PK_USUARIO PRIMARY KEY(USERNAME)  
);
```

Este sería nuestro script para la creación de las tablas en nuestra base de datos, dicho script con todos los datos ya listos para insertarlos en la tabla y comenzar a trabajar sobre estas.

CAMPOS DE LA TABLA

```
CREATE TABLE ZAPATILLA(  
  ID NUMERIC(5),  
  NAME VARCHAR(100),  
  PRICE FLOAT(5,2),  
  SIZES NUMERIC(2),  
  RELEASEDATE DATE,  
  STOCK BOOLEAN,  
  CONSTRAINT PK_ZAPATILLA PRIMARY KEY(ID)  
);  
  
CREATE TABLE USUARIO(  
  USERNAME VARCHAR(50),  
  PASSWORDD VARCHAR(50),  
  CONSTRAINT PK_USUARIO PRIMARY KEY(USERNAME)  
);
```

Como podemos ver la tabla Zapatilla, tenemos un campo numérico entero recogido por el Id de las zapatillas, el nombre es un campo de tipo String, el precio es un campo numérico decimal que recoge 2 decimales nada más, la talla recoge un campo numérico entero, la fecha recoge un campo de tipo date, y el stock recoge el campo boolean para comprobar si hay stock o no.

CLASES Y METODOS JAVA

```
public Connection conectar() {  
    Connection conexion = null;  
  
    try {  
        Class.forName(CONTROLADOR);  
        conexion = DriverManager.getConnection(URL, USUARIO,  
CLAVE);  
        System.out.println("Conexión OK");  
    } catch (ClassNotFoundException e) {  
        System.out.println("Error al cargar el controlador");  
        e.printStackTrace();  
    } catch (SQLException e) {  
        System.out.println("Error en la conexión");  
        e.printStackTrace();  
    }  
  
    return conexion;  
}
```

Utilizamos este método para conectarnos con la base de datos.

```
private static final String CONTROLADOR = "com.mysql.cj.jdbc.Driver";
    private static final String URL =
"jdbc:mysql://localhost:3306/mysqlpdb?useSSL=false";
    private static final String USUARIO = "root";
    private static final String CLAVE = "rootpass";
```

Como podemos ver declaramos variables de tipo final y static para guardar los datos utilizados sobre la base de datos.

```
public void addShoes(int idShoes,String name,double price,int sizes, int
year, int month,int day, boolean stock){
    Connection conexion=this.conectar();
    try {
        Statement instruccion=conexion.createStatement();
        String query = "INSERT INTO ZAPATILLA (ID, NAME,
PRICE, SIZES, RELEASEDATE, STOCK) VALUES (" + idShoes+", "+"\""+
name+"\""+", "+price+", "+sizes+", "+"\""+year+"-"+month+"-"+day+"\""+",
"+stock+")";
        instruccion.executeUpdate(query);
    } catch (SQLException e) {
        System.out.println("Error");
        e.printStackTrace();
    }
}
```

El siguiente método es para añadir un zapato a nuestra base de datos directamente, y no sobre la colección de java.

```
public void deleteShoes(int idShoes) {
    Connection conexion=this.conectar();

    try {
        Statement instruccion=conexion.createStatement();
        String query = "DELETE FROM ZAPATILLA WHERE
ID="+idShoes;
        instruccion.executeUpdate(query);
    } catch (SQLException e) {
        System.out.println("Error");
        e.printStackTrace();
    }
}
```

El siguiente método lo utilizaremos para borrar una zapatilla de la base de datos directamente.

```
public void updateShoes(int idShoes,String name,double price,int sizes,
boolean stock){
    Connection conexion=this.conectar();
    HashSet<Shoes> lista = this.loadList();

    Shoes s=null;
    for(Shoes e:lista) {
        if (e.getIdShoes()==idShoes){
            s=e;
        }
    }

    if (name.isEmpty() || name==null) {
        name=s.getName();
    }
    if (price<=0) {
        price=s.getPrice();
    }
    if (sizes<=35 || sizes>=52) {
        sizes=s.getSizes();
    }
    if (stock!=true || stock!=false) {
```

```
        stock=s.isStock();
    }
    try {
        Statement instruccion=conexion.createStatement();
        String query = "UPDATE ZAPATILLA SET
NAME="+""+name+""+", PRICE="+price+", SIZES="+sizes+",
STOCK="+stock+" WHERE ID="+idShoes;
        System.out.println(query);
        instruccion.executeUpdate(query);

    } catch (SQLException e) {
        System.out.println("Error");
        e.printStackTrace();
    }

}
```

Como ya podemos intuir este método es para actualizar las zapatillas que hay dentro de la base de datos, pero aquí recogeremos los campos con if para que controlar las excepciones y los null pointers.

```
public Shoes findShoe(int idShoes) {

    Shoes newShoes=null;
    HashSet<Shoes> lista = this.loadList();
    for(Shoes e:lista) {
        System.out.println(e.toString());
        if (e.getIdShoes()==idShoes) {
            newShoes=e;
        }

    }
    return newShoes;
}
```

Este método lo utilizaremos dentro de otro método más tarde, pero como podemos ver le pasamos un id. Con el método loadlist() que será el siguiente que veamos, este nos sirve para encontrar la zapatilla que buscamos.


```
public HashSet<Shoes> loadList() {
    Connection conexion= this.conectar();
    HashSet<Shoes> shoesList= new HashSet<>();
    try {
        Statement instruccion=conexion.createStatement();
        String query = "SELECT * FROM ZAPATILLA";
        ResultSet resultado=instruccion.executeQuery(query);
        while (resultado.next()) {

            Shoes shoesLoad=new Shoes(resultado.getInt("ID"),
resultado.getString("NAME"), resultado.getDouble("PRICE"),
resultado.getInt("SIZES"), resultado.getDate("RELEASEDATE"),
resultado.getBoolean("STOCK"));
            shoesList.add(shoesLoad);
        }
    } catch (SQLException e) {
        System.out.println("Error");
        e.printStackTrace();
    }
    return shoesList;
}
```

Este método dentro de la clase Dao es el mas importante ya que es el que nos ordena los campos de base de datos en un HashSet para que no haya 2 zapatillas iguales, este método nos devuelve una lista de zapatos.

Las clase Java no las mostraremos ya que lo único que contienen son sus atributos y los métodos getters, setters, toString y equals.

En la clase **main.jsp** nos encontramos con esto:

```
<%Dao d=new Dao();
for(Shoes e:d.loadList()) {%>
    <tr>
        <td>
            <%=e.getIdShoes()%>
        </td>
        <td>
            <%=e.getName()%>
```

```

</td>
<td>
        <%=e.getPrice()%>
</td>
<td>
        <%=e.getSizes()%>
</td>
<td>
        <%=e.getReleaseDate()%>
</td>
<td>
        <%=e.isStock()%>
</td>
<td>
        <a

```

```

href="editar.jsp?id=<%=e.getIdShoes()%>"></a>

```

```

        <a
href="confirmacionBorrar.jsp?id=<%=e.getIdShoes()%>"></a>

```

```

</td>

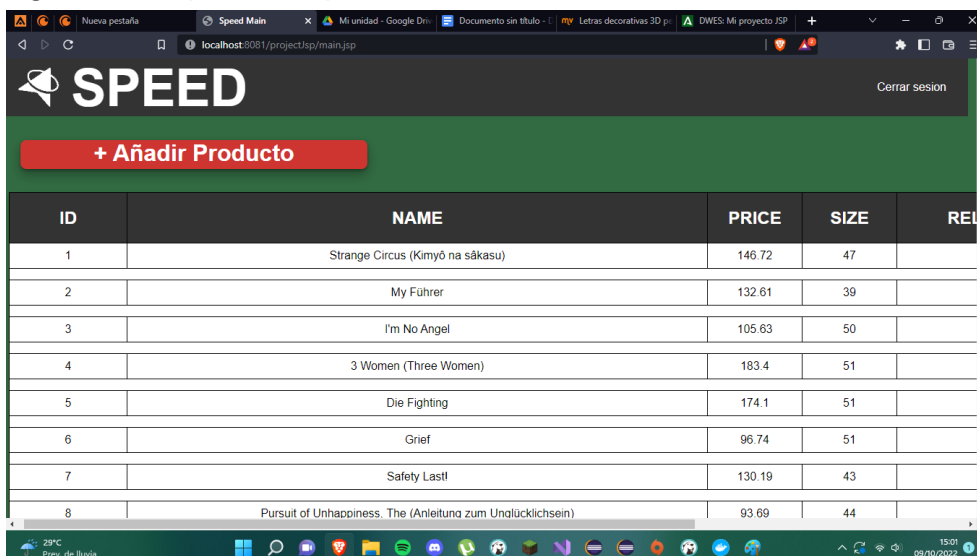
```

```

<tr>

```

Lo que conseguiremos con esto es que por cada fila que tenemos sacar lo que contiene ese campo e imprimirlo por pantalla, como mostraremos en la siguiente captura de pantalla.



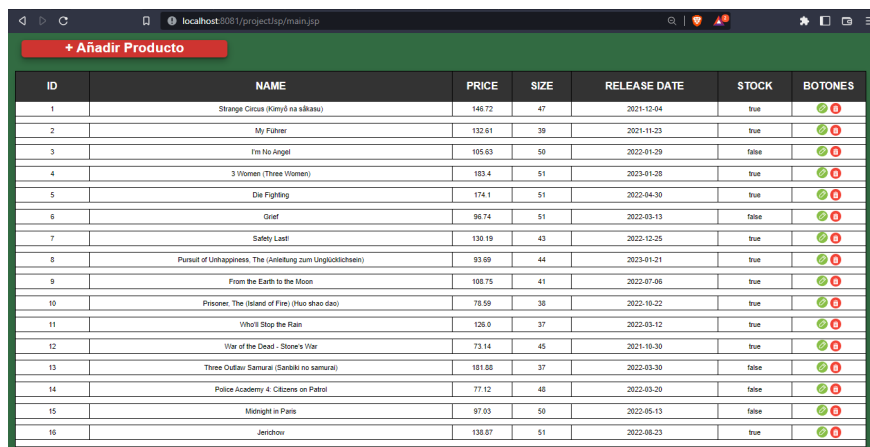
ID	NAME	PRICE	SIZE	REL
1	Strange Circus (Kimyô na sâkasu)	146.72	47	
2	My Führer	132.61	39	
3	I'm No Angel	105.63	50	
4	3 Women (Three Women)	183.4	51	
5	Die Fighting	174.1	51	
6	Grief	96.74	51	
7	Safety Last!	130.19	43	
8	Pursuit of Unhappiness. The (Anleitung zum Unglücklichsein)	93.69	44	































En el **index.jsp** nos encontraremos con el siguiente código que lo utilizaremos para crear los campos del login que los recogeremos más tarde en el **loginExec.jsp**

```
<div id="login">
    <br>
    <div id="introducir">
        <input type="text" class="usuario" id="usuario" name="usuario"
placeholder="User" >
    </div>
    <div id="introducir">
        <input type="password" class="password" id="password"
name="password" placeholder="Password" required>
    </div>
    <div id="introducir">
        <button type="submit" class="join"><h2>Log In</h2></button>
    </div>
    <br>
    <div id="barra">
        <a href="#" class="link">Forgot password?</a>
    </div>

</div>
```

[Enlace a github.](#)



ID	NAME	PRICE	SIZE	RELEASE DATE	STOCK	BOTONES
1	Strange Circus (Kimyô na sirkusu)	146.72	47	2021-12-04	true	 
2	My Father	132.61	39	2021-11-23	true	 
3	I'm No Angel	105.63	50	2022-01-29	false	 
4	3 Women (Three Women)	103.4	51	2023-01-28	true	 
5	Die Fighting	174.1	51	2022-04-30	true	 
6	Grief	96.74	51	2022-03-13	false	 
7	Safety Last!	130.19	43	2022-12-25	true	 
8	Pursuit of Unhappiness, The (Anleitung zum Unglücklichsein)	93.69	44	2023-01-21	true	 
9	From the Earth to the Moon	108.75	41	2022-07-06	true	 
10	Prisoner: The Island of Fire (Huo shao dao)	78.59	36	2022-10-22	true	 
11	Whirl! Stop the Rain	126.0	37	2022-03-12	true	 
12	War of the Dead - Stone's War	73.14	45	2021-10-30	true	 
13	Three Outlaw Samurai (Sanbiki no samurai)	101.88	37	2022-03-30	false	 
14	Police Academy 4: Citizens on Patrol	77.12	48	2023-03-20	false	 
15	Midnight in Paris	97.03	50	2022-05-13	false	 
16	Jerdchow	130.87	51	2022-08-23	true	