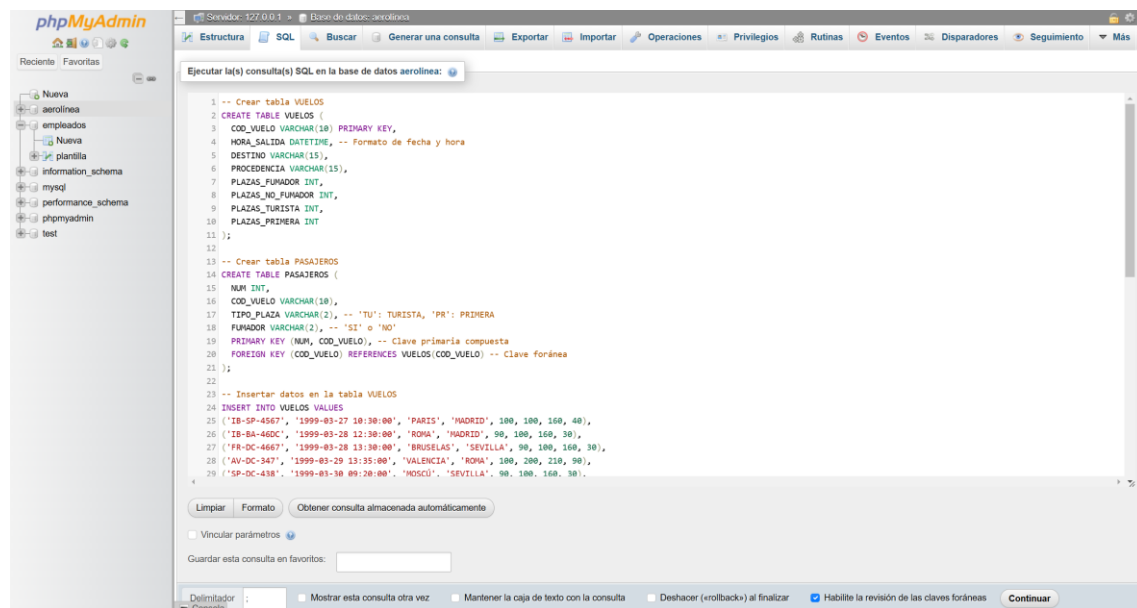


Se trata de hacer una aplicación en Java que acceda a una base de datos Oracle de una aerolínea. Consiste en una conexión a las tablas VUELOS y PASAJEROS de una BD Oracle que se implementara con el fichero que se adjunta. (Se puede hacer con otras bases de datos como MySQL, SQLite o PostgreSQL)

### [Base de datos de datos para la tarea \(0.01 MB\)](#)

En primer lugar, para crear la base de datos se ha usado XAMPP. Para ello ha habido que modificar algunas consultas SQL:

Se ha creado la base de datos Aerolínea en PHPmyAdmin y se han insertado las tablas con sus relaciones:



```
1 -- Crear tabla VUELOS
2 CREATE TABLE VUELOS (
3   COD_VUELO VARCHAR(10) PRIMARY KEY,
4   HORA_SALIDA DATETIME, -- Formato de fecha y hora
5   DESTINO VARCHAR(15),
6   PROCEDENCIA VARCHAR(15),
7   PLAZAS_FUMADOR INT,
8   PLAZAS_NO_FUMADOR INT,
9   PLAZAS_TURISTA INT,
10  PLAZAS_PRIMERA INT
11 );
12
13 -- Crear tabla PASAJEROS
14 CREATE TABLE PASAJEROS (
15   NUM INT,
16   COD_VUELO VARCHAR(10),
17   TIPO_PLAZA VARCHAR(2), -- 'TU': TURISTA, 'PR': PRIMERA
18   FUMADOR VARCHAR(2), -- 'SI' o 'NO'
19   PRIMARY KEY (NUM, COD_VUELO), -- Clave primaria compuesta
20   FOREIGN KEY (COD_VUELO) REFERENCES VUELOS(COD_VUELO) -- Clave foránea
21 );
22
23 -- Insertar datos en la tabla VUELOS
24 INSERT INTO VUELOS VALUES
25 ('IB-SP-4567', '1999-03-27 10:30:00', 'PARIS', 'MADRID', 100, 100, 160, 40),
26 ('IB-BA-460C', '1999-03-28 12:30:00', 'ROMA', 'MADRID', 80, 100, 160, 30),
27 ('FR-DC-4667', '1999-03-28 13:30:00', 'BRUSELAS', 'SEVILLA', 90, 100, 160, 30),
28 ('AV-DC-347', '1999-03-29 13:35:00', 'VALENCIA', 'ROMA', 100, 200, 210, 50),
29 ('SP-DC-438', '1999-03-30 09:20:00', 'MOSCÚ', 'SEVILLA', 90, 100, 160, 30);
```

Consta de las siguientes clases MAIN:

1. Mostrar y pedir información de la base de datos en general.
2. Mostrar la información de la tabla pasajeros.
3. Ver la información de los pasajeros de un vuelo, pasando el código de vuelo como parámetro.
4. Insertar un vuelo cuyos valores se pasan como parámetros.
5. Borrar el vuelo que se metió anteriormente en el que se pasa por parámetro su número de vuelo.
6. Modificar los vuelos de fumadores a no fumadores.

## 2. Configuración del Proyecto en Java

En primer lugar, como estamos trabajando con Maven, tendremos que añadir a las dependencias el conector de mysql:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.myccompany</groupId>
  <artifactId>Aerolinea</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <dependencies>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.27</version>
    </dependency>
  </dependencies>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.release>21</maven.compiler.release>
    <exec.mainClass>com.myccompany.aerolinea.Aerolinea</exec.mainClass>
  </properties>
</project>
```

En el caso de trabajar con un proyecto Ant habría que:

- Conector JDBC para MySQL: Descarga el conector MySQL JDBC desde: [MySQL Connector/J](#).
- Agrega el archivo .jar al proyecto:
  1. Haz clic derecho en el proyecto y selecciona Add Library o Add External JAR.
  2. Selecciona el archivo mysql-connector-java-x.x.x.jar.

### 1. Clase Principal: Aerolinea

La clase principal del programa tiene las siguientes funcionalidades:

1. Conexión a la Base de Datos:

Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/aerolínea", "root", "");

- localhost: La dirección del servidor MySQL.
- 3306: Puerto por defecto de MySQL.
- aerolínea: Nombre de la base de datos.
- root y "": Usuario y contraseña del servidor.

```
public static void main(String[] args) {
    try {
        boolean salir = false; // Variable para controlar la salida del programa.
        Scanner scan = new Scanner(System.in); //Entrada de datos por consola
        int seleccion;

        //En caso de querer hacer con OracleDatabase usar:
        // Cargar el driver de Oracle JDBC y establecer conexión con la base de datos.
        //Class.forName("oracle.jdbc.driver.OracleDriver");
        //Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "vuelo", "vuelo");

        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/aerolínea", "root", "");

        Statement statement; //Interfaz para el envío de sentencias SQL a la base de datos con parámetros estáticos.
        ResultSet resultSet; //Almacena los resultados devueltos por la ejecución de una consulta SQL.
        PreparedStatement preparedStatement; //Interfaz para el envío de sentencias SQL a la base de datos con parámetros dinámicos.
```

- Menú de Opciones: El programa presenta un menú para que el usuario seleccione una opción:

```
System.out.println("1. Mostrar información general de la base de datos.");  
System.out.println("2. Mostrar la información de la tabla pasajeros.");  
System.out.println("3. Ver información de pasajeros de un vuelo.");  
System.out.println("4. Insertar un nuevo vuelo.");  
System.out.println("5. Borrar un vuelo insertado previamente.");  
System.out.println("6. Modificar los vuelos de fumadores a no fumadores.");  
System.out.println("7. Salir del programa.");
```

```
while (!salir) {  
    // Mostrar el menú de opciones disponibles al usuario.  
    System.out.println("Indica un numero para seleccionar una opcion del programa:");  
    System.out.println("1. Mostrar y pedir informacion de la base de datos en general.");  
    System.out.println("2. Mostrar la informacion de la tabla pasajeros.");  
    System.out.println("3. Ver la informacion de los pasajeros de un vuelo, pasando el codigo de vuelo como parametro.");  
    System.out.println("4. Insertar un vuelo cuyos valores se pasan como parametros.");  
    System.out.println("5. Borrar el vuelo que se metio anteriormente en el que se pasa por parametro su numero de vuelo.");  
    System.out.println("6. Modificar los vuelos de fumadores a no fumadores.");  
    System.out.println("7. Salir del programa");  
    seleccion = scan.nextInt();  
  
    //Creación de las variables.  
    String codigo, horaSalida, destino, procedencia;  
    int plazasFumador, plazasNoFumador, plazasTurista, plazasPrimera;  
  
    //Iniciación de la variable codVueloAnterior para la opcion 5.  
    String codVueloAnterior = null;
```

## 2. Funcionalidades Implementadas

### 1. Mostrar información de las tablas: Consulta los nombres y tipos de las columnas de las tablas PASAJEROS y VUELOS.

```
//Estructura de control condicional para ejecutar las distintas opciones del programa  
switch (seleccion) {  
    case 1:  
        /**  
         * Caso 1: Muestra la información de las tablas "pasajeros" y "vuelos".  
         * Muestra los nombres de las columnas y sus tipos de datos.  
         */  
  
        statement = con.createStatement(); //Crea un objeto Statement a partir de la conexión establecida a la base de datos  
        resultSet = statement.executeQuery("select * from pasajeros"); // Selecciona todos los registros de la tabla pasajeros  
  
        ResultSetMetaData result = resultSet.getMetaData(); //Obtiene todos los datos de la consulta.  
        int columnas = result.getColumnCount(); // obtiene el número de columnas de la tabla pasajeros.  
        System.out.println("Informacion de la tabla PASAJEROS:");  
        System.out.println("-----");  
        //Iterador que recoge la información sobre el nombre y el tipo de dato de cada columna de la tabla pasajeros.  
        for (int i = 1; i <= columnas; i++) {  
            System.out.println("Informacion acerca de la columna " + result.getColumnName(i));  
            System.out.println("Tipo de dato: " + result.getColumnTypeName(i) + ".");  
        }  
  
        resultSet = statement.executeQuery("select * from vuelos"); // Selecciona todos los registros de la tabla vuelos  
        result = resultSet.getMetaData(); //Obtiene todos los datos de la consulta.  
        columnas = result.getColumnCount(); // obtiene el número de columnas de la tabla vuelos.  
        System.out.println("\nInformacion de la tabla VUELOS:");  
        System.out.println("-----");  
        //Iterador que recoge la información sobre el nombre y el tipo de dato de cada columna de la tabla vuelos.  
        for (int i = 1; i <= columnas; i++) {  
            System.out.println("Informacion acerca de la columna " + result.getColumnName(i));  
            System.out.println("Tipo de dato: " + result.getColumnTypeName(i) + ".");  
        }  
  
        //cierre de los objetos statement y resultSet para liberar recursos.  
        statement.close();  
        resultSet.close();  
        break;
```

## 2. Mostrar la información de pasajeros: Consulta todo el contenido de la tabla PASAJEROS:

```
case 2:
/**
 * Caso 2: Muestra el contenido de la tabla "pasajeros".
 * Incluye las columnas NUM, COD_VUELO, TIPO_PLAZA, y FUMADOR.
 */
statement = con.createStatement();//Crea un objeto Statement a partir de la conexión establecida a la base de datos
resultSet = statement.executeQuery("select * from pasajeros");// Selecciona todos los registros de la tabla pasajeros

System.out.println("NUM\tCOD_VUELO\tTIPO_PLAZA\tFUMADOR\t");
//Obtiene cada
while (resultSet.next()) { //Recorre los datos contenidos en el objeto resultSet y los imprime por pantalla
    System.out.print(resultSet.getInt("NUM") + "\t");
    System.out.print(resultSet.getString("COD_VUELO") + "\t");
    System.out.print(resultSet.getString("TIPO_PLAZA") + "\t" + "\t");
    System.out.print(resultSet.getString("FUMADOR") + "\t");
    System.out.println("");
}
//cierre de los objetos statement y resultSet para liberar recursos.
statement.close();
resultSet.close();
break;
```

## 3. Consultar pasajeros de un vuelo: Solicita al usuario el código de vuelo y filtra por COD\_VUELO.

```
case 3:
/**
 * Caso 3: Solicita un código de vuelo y muestra los pasajeros asociados a ese vuelo.
 */
String codVuelo;
System.out.println("Indica el código del vuelo:");
codVuelo = scan.next();
//Crea un objeto PreparedStatement a partir de la conexión establecida a la base de datos que prepara
//una consulta SQL que recupera datos de la tabla pasajeros en función del valor que se proporcione para el parámetro ?
preparedStatement = con.prepareStatement("select * from pasajeros where cod_vuelo = ?");
//Se utiliza el método setString para asignar el valor de codVuelo al marcador de posición ? (1)
preparedStatement.setString(1, codVuelo);
resultSet = preparedStatement.executeQuery();//Ejecuta la consulta

System.out.println("NUM\tCOD_VUELO\tTIPO_PLAZA\tFUMADOR\t");
while (resultSet.next()) { //Recorre los datos contenidos en el objeto resultSet y los imprime por pantalla
    System.out.print(resultSet.getInt("NUM") + "\t");
    System.out.print(resultSet.getString("COD_VUELO") + "\t");
    System.out.print(resultSet.getString("TIPO_PLAZA") + "\t" + "\t");
    System.out.print(resultSet.getString("FUMADOR") + "\t");
    System.out.println("");
}
//cierre de los objetos statement y resultSet para liberar recursos.
preparedStatement.close();
resultSet.close();
break;
```

#### 4. Insertar un nuevo vuelo: Recoge los datos del usuario y ejecuta una consulta INSERT.

```
case 4:
/**
 * Caso 4: Solicita los datos para un nuevo vuelo e inserta el registro en la tabla "vuelos".
 */
System.out.println("Inserte los datos para crear un nuevo vuelo:");
System.out.println("Inserte el codigo");
codigo = scan.next();
System.out.println("Inserta hora de salida (formato: DD/MM/AA 00:00 )");
horaSalida = scan.next();
System.out.println("Inserta el destino:");
destino = scan.next();
System.out.println("Inserta la procedencia:");
procedencia = scan.next();
System.out.println("Indica el numero de plazas para fumadores:");
plazasFumador = scan.nextInt();
System.out.println("Indica el numero de plazas para no fumadores:");
plazasNoFumador = scan.nextInt();
System.out.println("Indica el numero de plazas en turista:");
plazasTurista = scan.nextInt();
System.out.println("Indica el número de plazas en primera:");
plazasPrimera = scan.nextInt();

//Crea un objeto PreparedStatement a partir de la conexión establecida a la base de datos que prepara
//una consulta SQL que inserta datos de la tabla vuelos en función del valor que se proporcione para los parámetros ?
PreparedStatement = con.prepareStatement("insert into vuelos values (?, ?, ?, ?, ?, ?, ?, ?)");

//Se utilizan los métodos setString y setInt para asignar el valor de las variables solicitadas al marcador de posición ? (1)
preparedStatement.setString(1, codigo);
preparedStatement.setString(2, horaSalida);
preparedStatement.setString(3, destino);
preparedStatement.setString(4, procedencia);
preparedStatement.setInt(5, plazasFumador);
preparedStatement.setInt(6, plazasNoFumador);
preparedStatement.setInt(7, plazasTurista);
preparedStatement.setInt(8, plazasPrimera);

//Condional que comprueba que se haya insertado datos a la tabla
if (preparedStatement.executeUpdate() > 0) {
    System.out.println("Vuelo insertado");
    codVueloAnterior = codigo; //Igualamos el valor codVueloAnterior a código para usarlo en el siguiente apartado.
} else {
    System.out.println("Error al insertar el vuelo");
}

//cierre del objeto PreparedStatement para liberar recursos.
preparedStatement.close();
break;
```

#### 5. Borrar un vuelo: Usa el código del vuelo insertado para ejecutar un DELETE.

```
case 5:
/**
 * Caso 5: Elimina el vuelo previamente insertado (identificado por su código).
 */

//Condional que comprueba que se haya insertado un vuelo anteriormente. Si se cumple la condicion borra el vuelo.
if (codVueloAnterior != null) {
    //Crea un objeto PreparedStatement a partir de la conexión establecida a la base de datos que prepara
    //una consulta SQL que borra datos de la tabla vuelos en función del valor que se proporcione para los parámetros ?
    PreparedStatement = con.prepareStatement("delete from vuelos where cod_vuelo = ?");
    //Se utiliza el método setString para asignar el valor de codVueloAnterior al marcador de posición ? (1)
    preparedStatement.setString(1, codVueloAnterior);
    //Condional que comprueba que se haya realizado al menos una modificación en la tabla.
    if (preparedStatement.executeUpdate() > 0) {
        System.out.println("Se ha borrado el vuelo anterior");
    } else {
        System.out.println("Error al borrar el vuelo. No se ha insertado nuevo vuelo antes.");
    }
    preparedStatement.close();
}
break;
```

## 6. Modificar vuelos a no fumadores: Actualiza las columnas FUMADOR y las plazas de la tabla VUELOS.

```
case 6:
    /**
     * Caso 6: Actualiza los registros para convertir plazas de fumadores en no fumadores.
     */

    //Se crea un objeto Statement utilizando la conexión a la base de datos con
    statement = con.createStatement();
    //Esta consulta actualizará todos los registros de la tabla pasajeros, cambiando el valor de la columna fumador a 'NO'
    //El método executeUpdate() devuelve el número de filas afectadas por la operación.
    //El valor devuelto se almacena en la variable filas.
    int filas = statement.executeUpdate("update pasajeros set fumador = 'NO'");

    //Condional que comprueba que al menos un pasajero fue actualizado.
    if (filas > 0) {
        System.out.println("Se han actualizado los pasajeros de Fumadores a No fumadores");
    }

    //Esta consulta actualizará todos los registros de la tabla vuelos, convierte las plazas para fumadores en plazas para no fumadores.
    filas = statement.executeUpdate("update vuelos set plazas_no_fumador = plazas_no_fumador + plazas_fumador, plazas_fumador = 0");
    //Condional que comprueba que al menos un vuelo fue actualizado.
    if (filas > 0) {
        System.out.println("Se han actualizado los vuelos incluyendo solo pasajeros no fumadores");
    }
    break;
```

## 7. Salir de la aplicación: Salimos del bucle while:

```
        case 7:
            /**
             * Caso 7: Sale del programa.
             */
            salir = true;
            break;

        default:
            System.out.println("La opción indicada no existe");
    }
}
} catch (SQLException ex) {
    Logger.getLogger(Aerolinea.class.getName()).log(Level.SEVERE, null, ex);
}
}
```