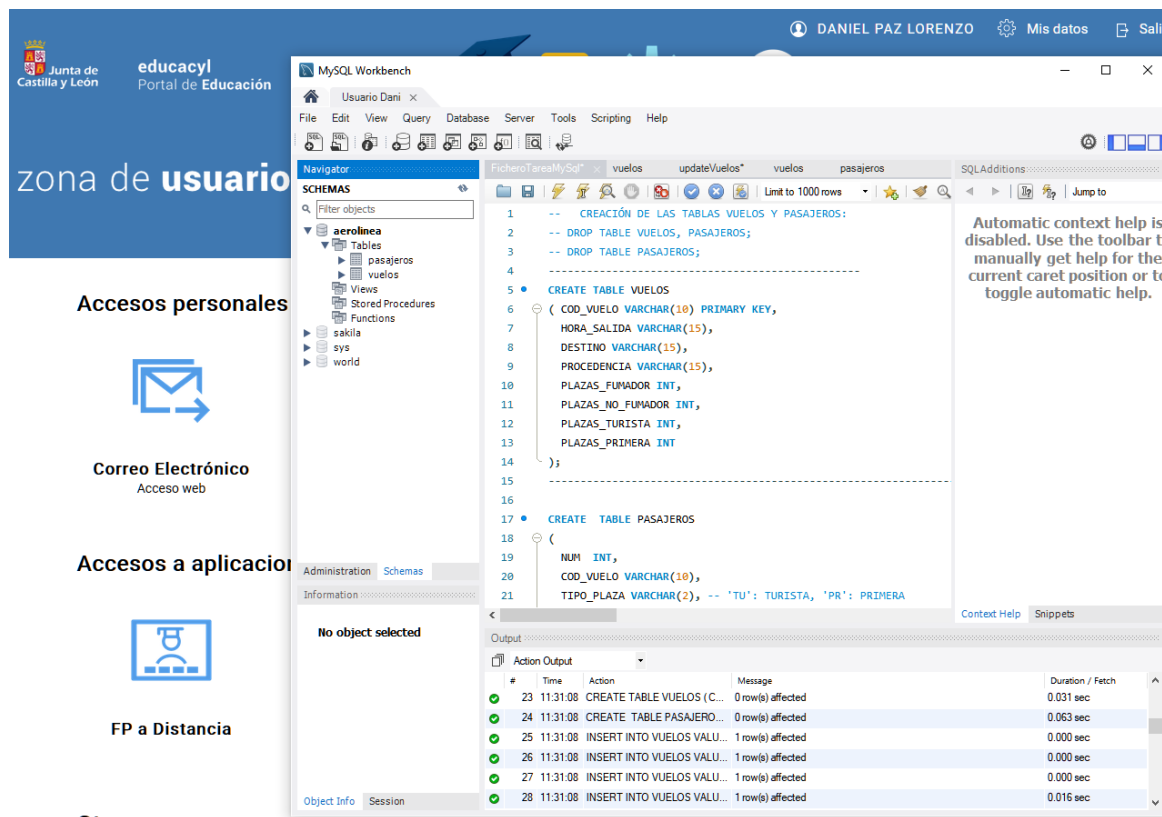


En esta tarea se nos pide que hagamos una aplicación que acceda a la base de datos de una aerolínea y muestre o modifique algunos datos, según la elección del usuario.

Lo primero que haremos será crear la base de datos de la aerolínea con el fichero .sql que nos proporciona la práctica:

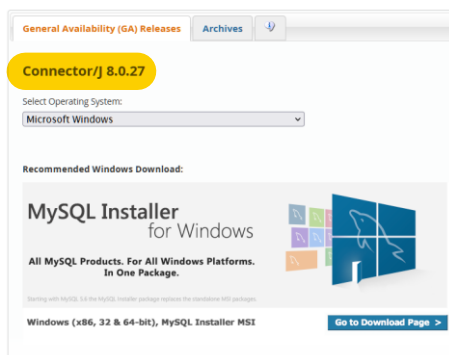


He utilizado la base de datos de MySQL, he creado un esquema llamado "aerolínea" y dentro del mismo ejecuto el script "FicheroTarea.sql" con algunas modificaciones ya que viene preparado para ejecutarse en Oracle (este fichero modificado lo he guardado con la tarea)

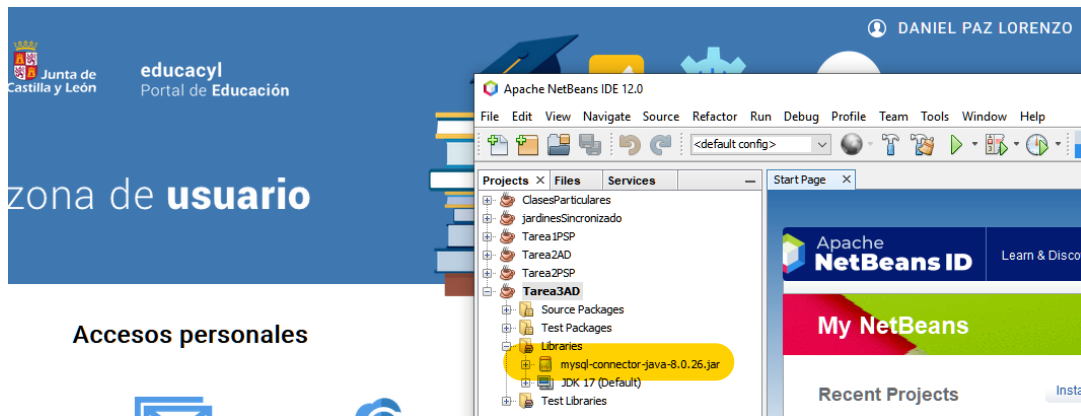
El siguiente paso será descargarnos el driver de conexión JDBC para MySQL que es de tipo 4 (protocolo nativo), es decir que las sentencias que nosotros creamos en nuestra aplicación las transfiere directamente a la base de datos.

[MySQL Community Downloads](#)

Connector/J

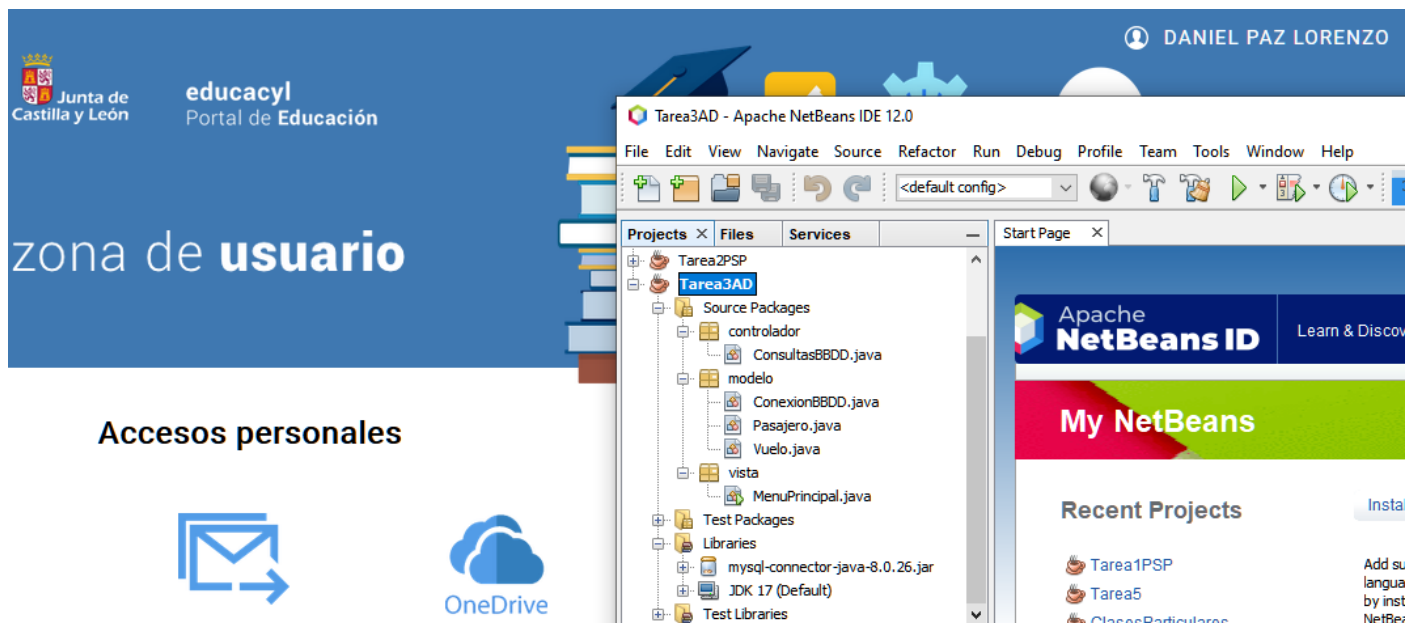


Lo vamos a descargar desde la propia página de MySQL



*Una vez descargado lo añadimos como librería a nuestro proyecto en NetBeans*

En este punto ya estamos preparados para empezar a diseñar nuestra aplicación y me voy a basar en el patrón **modelo-vista-controlador** en el que cada uno será un paquete.



*Dentro de cada paquete crearé las clases que vamos a necesitar como muestro en la imagen*

- **Paquete modelo:** Contiene 3 clases:

- *ConexionBBDD*: En ella vamos a gestionar la conexión con la base de datos mediante un objeto de la clase *Connection*.

```
public class ConexionBBDD {

    //Datos para la conexión con mi base de datos de MySql
    static final String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost:3306/aerolinea";
    static final String USER = "Dani";
    static final String PASS = "5678";

    private Connection Conexion;

    public ConexionBBDD() {

        try {
            Class.forName(JDBC_DRIVER);
            this.Conexion = DriverManager.getConnection(DB_URL, USER, PASS);
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(ConexionBBDD.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public Connection getConexion() {
        return Conexion;
    }
}
```

- *Pasajero*: La vamos a diseñar como si fuera la tabla "pasajero" de la bbdd, referenciamos sus campos como atributos de esta clase. Esto lo haremos para poder trabajar con los datos de la tabla dentro de nuestra aplicación ya que Java trabaja con objetos y la base de datos es relacional.

```
public class Pasajero {

    //Declaración de atributos de clase
    int num;
    String cod_vuelo;
    String tipo_plaza;
    String fumador;

    /**
     * Constructor de clase pasajero
     */
    public Pasajero() {
    }
}
```

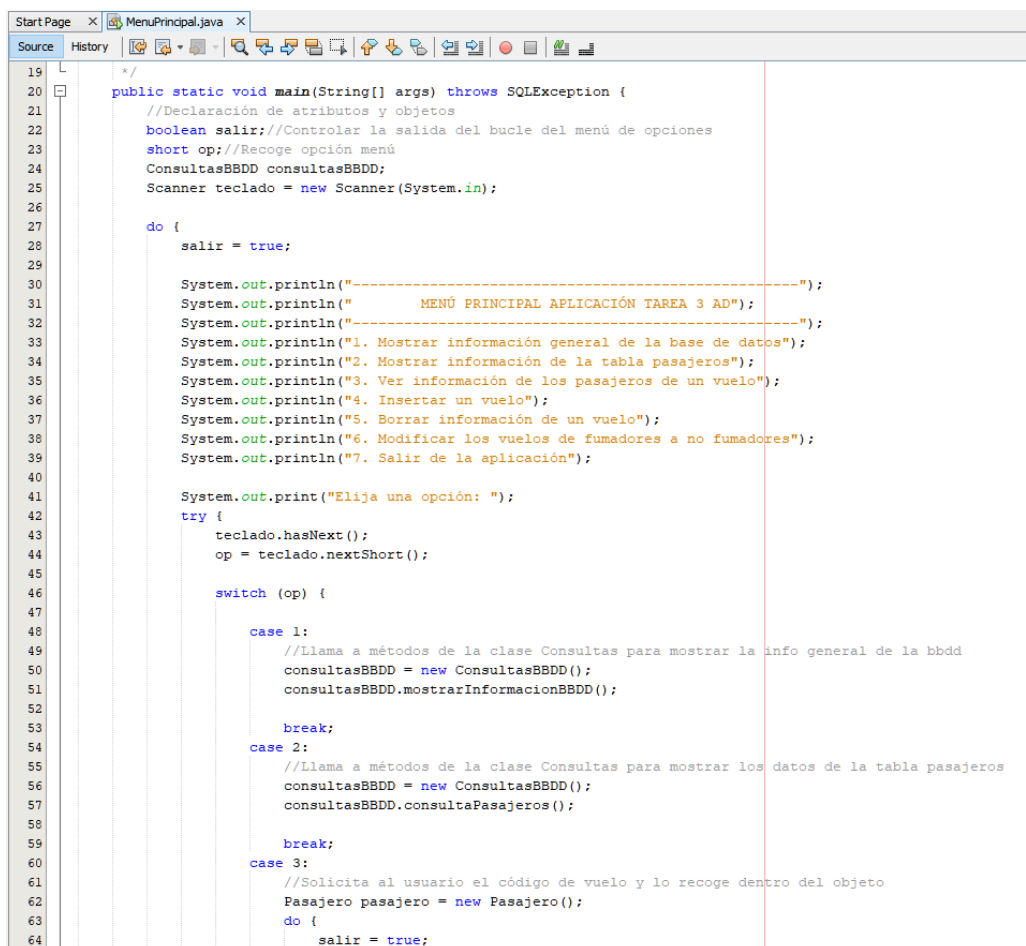
- *Vuelo*: Al igual que *Pasajero* la vamos a diseñar como si fuera la tabla "vuelo" de la bbdd, referenciamos sus campos como atributos de esta clase. Esto lo haremos para poder trabajar con los datos de la tabla dentro de nuestra aplicación, ya que Java trabaja con objetos y la base de datos es relacional.

```
public class Vuelo {

    //Declaración de atributos de la clase
    String cod_vuelo = null;
    String hora_salida;
    String destino;
    String procedencia;
    int plazas_fumador;
    int plazas_nofumador;
    int plazas_turista;
    int plazas_primera;

    /**
     * Constructor de la clase Vuelo
     */
    public Vuelo() {
    }
}
```

- **Paquete vista:** En esta parte vamos a crear lo que verá el usuario final y que va a interaccionar con el mismo mostrando y recogiendo la información que nos proporciona. Contiene la clase principal:
  - *MenúPrincipal:* En esta clase vamos a mostrar los menús de la aplicación según las opciones del usuario. Desde aquí llamaremos a los distintos métodos de clases según sean necesarios.



```

19  */
20  public static void main(String[] args) throws SQLException {
21      //Declaración de atributos y objetos
22      boolean salir; //Controlar la salida del bucle del menú de opciones
23      short op; //Recoge opción menú
24      ConsultasBBDD consultasBBDD;
25      Scanner teclado = new Scanner(System.in);
26
27      do {
28          salir = true;
29
30          System.out.println("-----");
31          System.out.println("      MENÚ PRINCIPAL APLICACIÓN TAREA 3 AD");
32          System.out.println("-----");
33          System.out.println("1. Mostrar información general de la base de datos");
34          System.out.println("2. Mostrar información de la tabla pasajeros");
35          System.out.println("3. Ver información de los pasajeros de un vuelo");
36          System.out.println("4. Insertar un vuelo");
37          System.out.println("5. Borrar información de un vuelo");
38          System.out.println("6. Modificar los vuelos de fumadores a no fumadores");
39          System.out.println("7. Salir de la aplicación");
40
41          System.out.print("Elija una opción: ");
42          try {
43              teclado.hasNext();
44              op = teclado.nextShort();
45
46              switch (op) {
47
48                  case 1:
49                      //Llama a métodos de la clase Consultas para mostrar la info general de la bbdd
50                      consultasBBDD = new ConsultasBBDD();
51                      consultasBBDD.mostrarInformacionBBDD();
52
53                      break;
54                  case 2:
55                      //Llama a métodos de la clase Consultas para mostrar los datos de la tabla pasajeros
56                      consultasBBDD = new ConsultasBBDD();
57                      consultasBBDD.consultaPasajeros();
58
59                      break;
60                  case 3:
61                      //Solicita al usuario el código de vuelo y lo recoge dentro del objeto
62                      Pasajero pasajero = new Pasajero();
63                      do {
64                          salir = true;

```

- **Paquete controlador:** Podríamos decir que este paquete junto con sus clases forman el motor de nuestra aplicación y una de las partes más importantes de la misma ya que se va a encargar de enlazar la base de datos con la interfaz de usuario.
  - *ConsultasBBDD:* Esta clase va a gestionar nuestras consultas a la base de datos devolviendo a su vez los resultados requeridos. Muestra de como quedaría definida la clase sin desarrollar los métodos:

```

public class ConsultasBBDD {

    private final ConexionBBDD conexionBBDD;

    /**
     * Constructor de la clase que genera la conexión con la bbdd
     */
    public ConsultasBBDD() {
        this.conexionBBDD = new ConexionBBDD();
    }
}

```

```
/**
 * Método para mostrar y pedir información de la base de datos en general.
 *
 * @throws java.sql.SQLException
 */
public void mostrarInformacionBBDD() throws SQLException {

/**
 * Método para mostrar la información de la tabla pasajeros.
 *
 * @throws SQLException
 */
public void consultaPasajeros() throws SQLException {
}

/**
 * Método para ver la información de los pasajeros de un vuelo, pasando el
 * código de vuelo como parámetro.
 *
 * @param codigoVuelo String de máx. 10 caracteres
 * @throws SQLException
 */
public void consultaPasajerosVuelo(String codigoVuelo) throws SQLException {
}

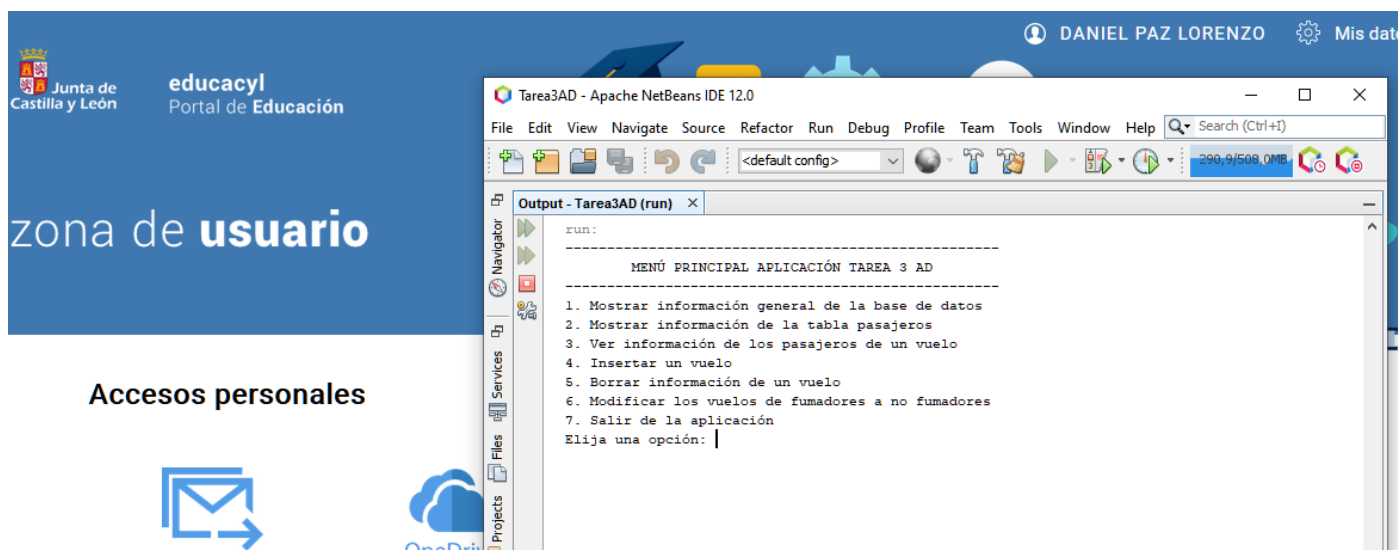
/**
 * Método para insertar un vuelo cuyos valores se pasan como parámetros,
 * dentro de la tabla vuelos
 *
 * @param vuelo objeto de la clase del tipo Vuelo
 * @throws SQLException
 */
public void insertarVuelo(Vuelo vuelo) throws SQLException {
}

/**
 * Método para borrar un vuelo, de la tabla vuelos, que se mete como
 * parámetro
 *
 * @param codigoVuelo String de máx. 10 caracteres
 * @throws SQLException
 */
public void borrarVuelo(String codigoVuelo) throws SQLException {
}

/**
 * Método para modificar los vuelos de fumadores a no fumadores
 *
 * @throws SQLException
 */
public void modificarVuelosFumadoresNoFumadores() throws SQLException {
}
}
```

## PRUEBAS Y FUNCIONAMIENTO DE LA APLICACIÓN

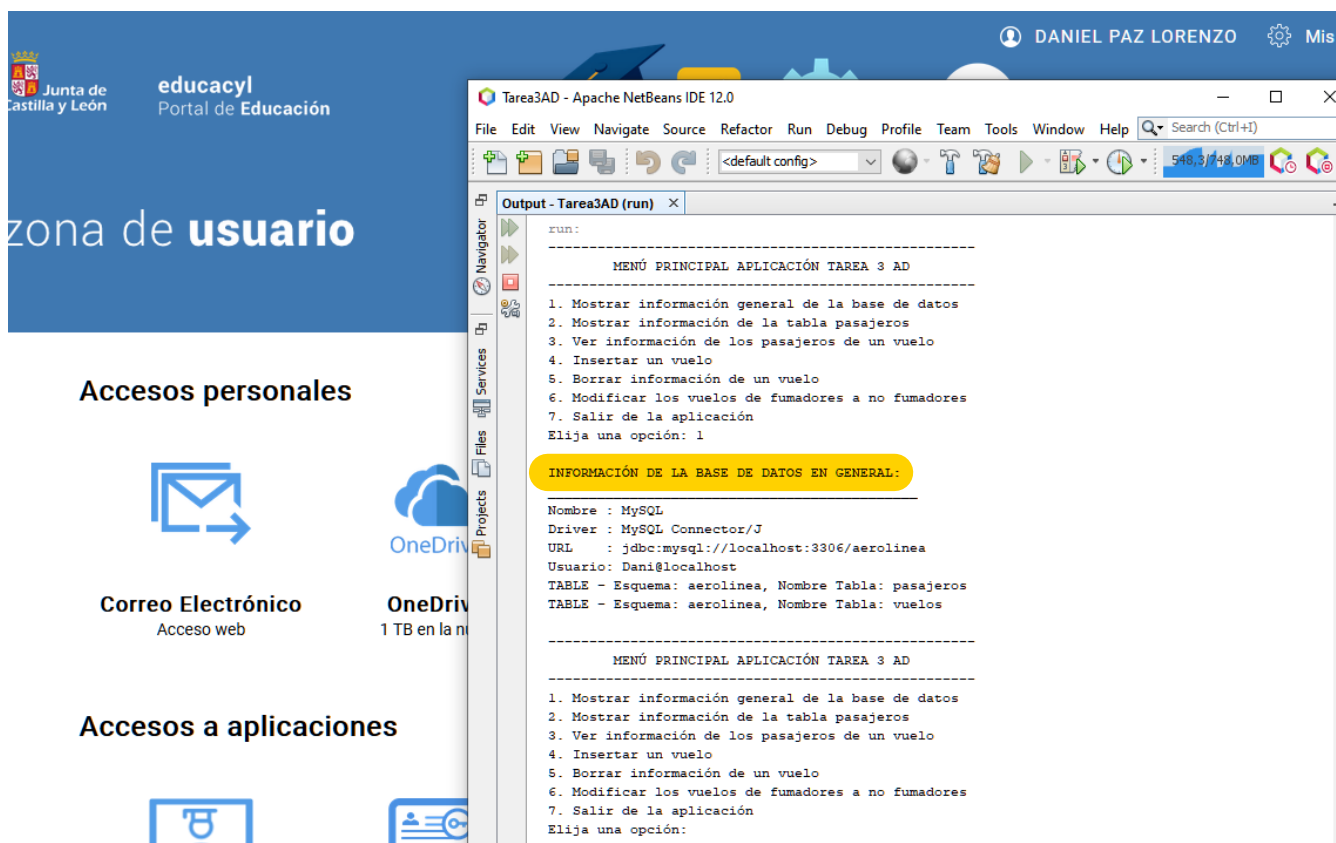
Lo primero que vamos a hacer es una vez que hemos depurado y comprobado que no tiene errores, ejecutar la aplicación.



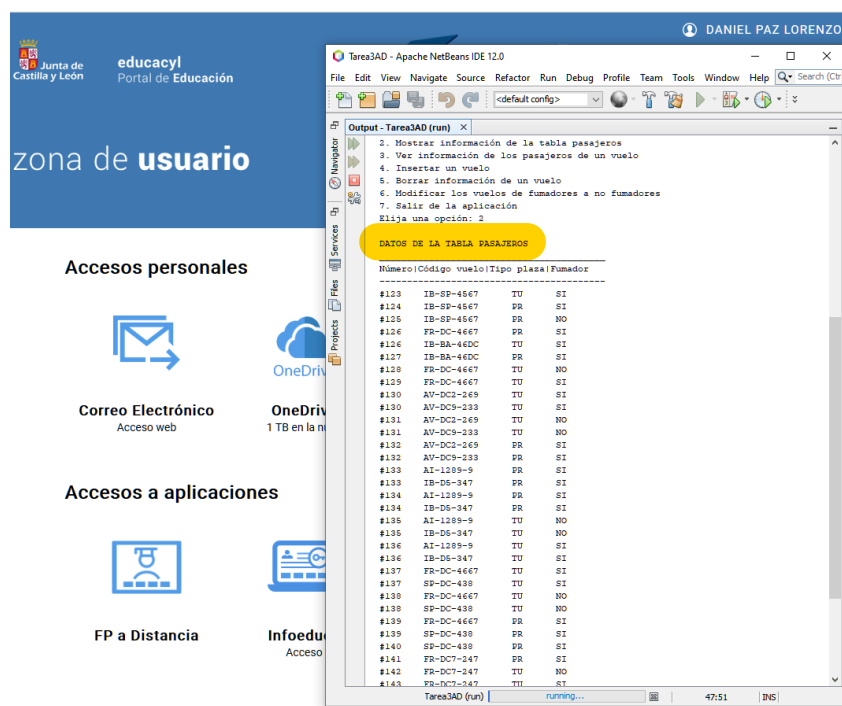
Nos mostrará el menú principal con las distintas opciones para que el usuario seleccione una

Ahora vamos en orden de la primera a última para ver que todas funcionan:

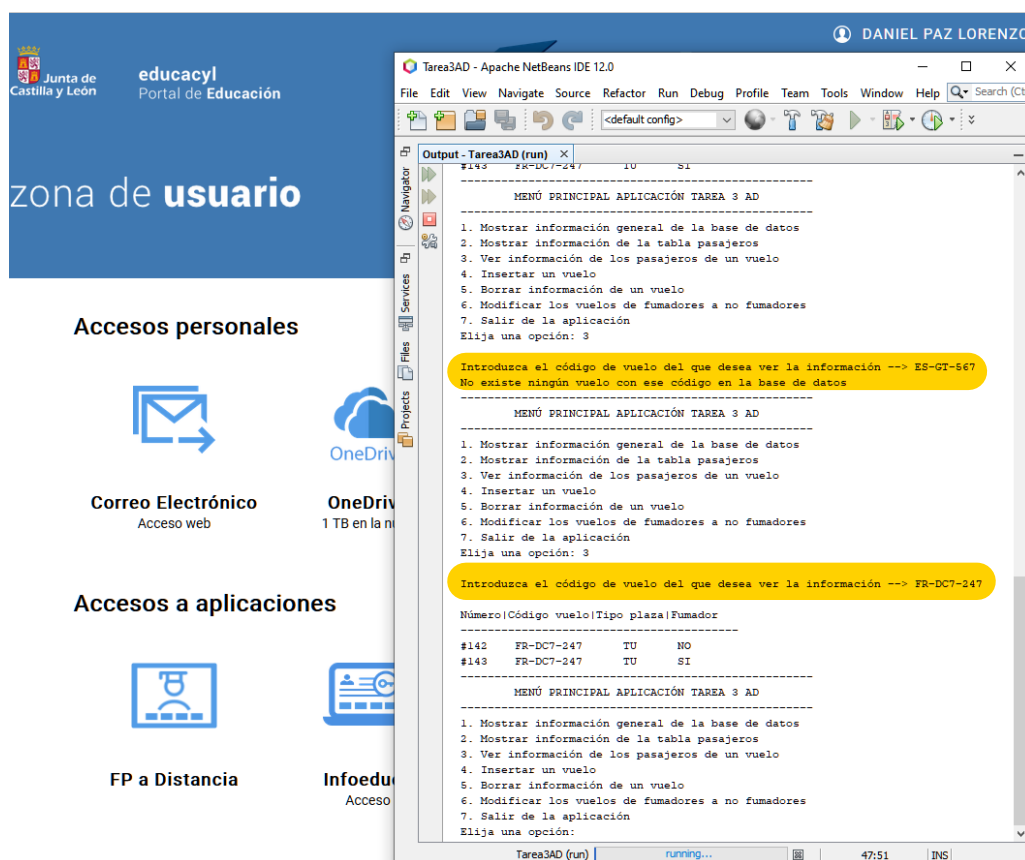
1. Opción 1: Muestra información relacionada con la base de datos en general como son el nombre de la misma, driver, url, usuario, esquema, nombre de las tablas y lo muestra por pantalla. Cuando termina vuelve al menú principal para seleccionar otra opción.



2. Opción 2: Muestra por pantalla toda la información contenida dentro de la tabla "pasajeros".

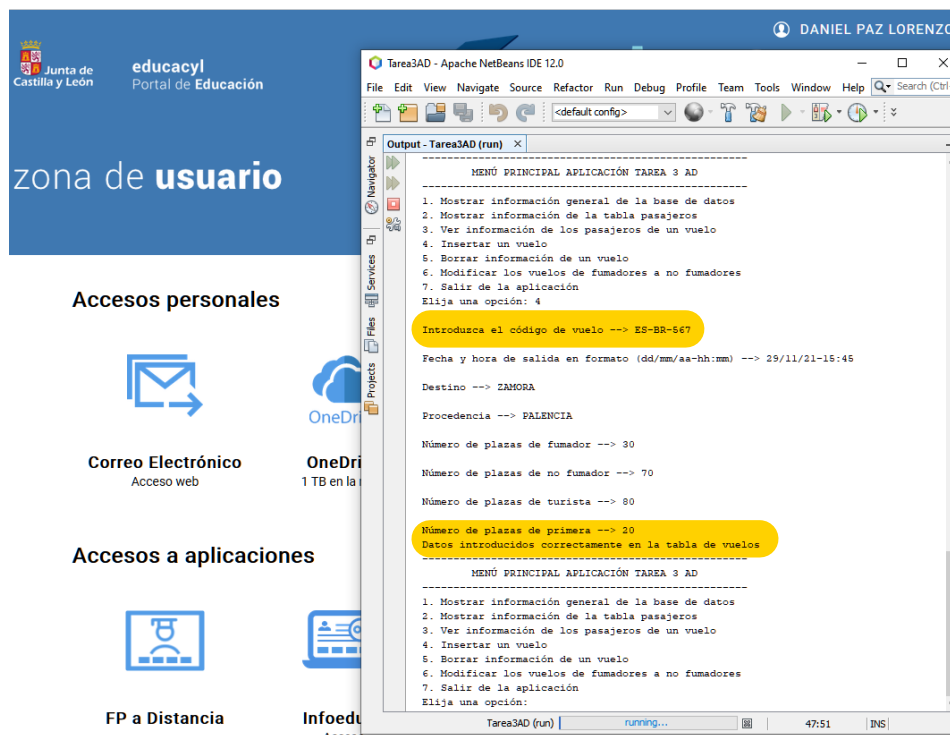


3. Opción 3: Solicita número o código de vuelo y muestra por pantalla la información del mismo. Si este no existiera dentro de la base de datos la aplicación nos muestra un mensaje advirtiéndonos:

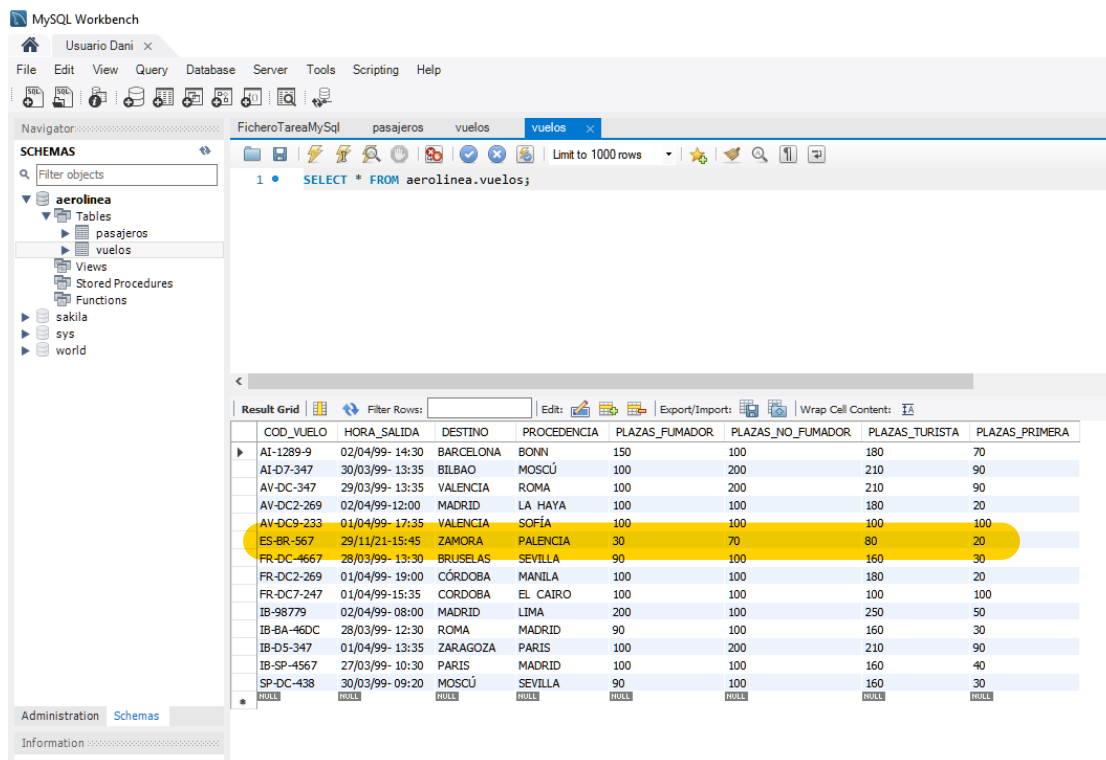


Comprobamos con un vuelo que no existe y lanza el mensaje y con uno que existe y muestra la información

4. Opción 4: Aquí vamos a introducir todos los datos de un vuelo dentro de la tabla “vuelos” de la base de datos.

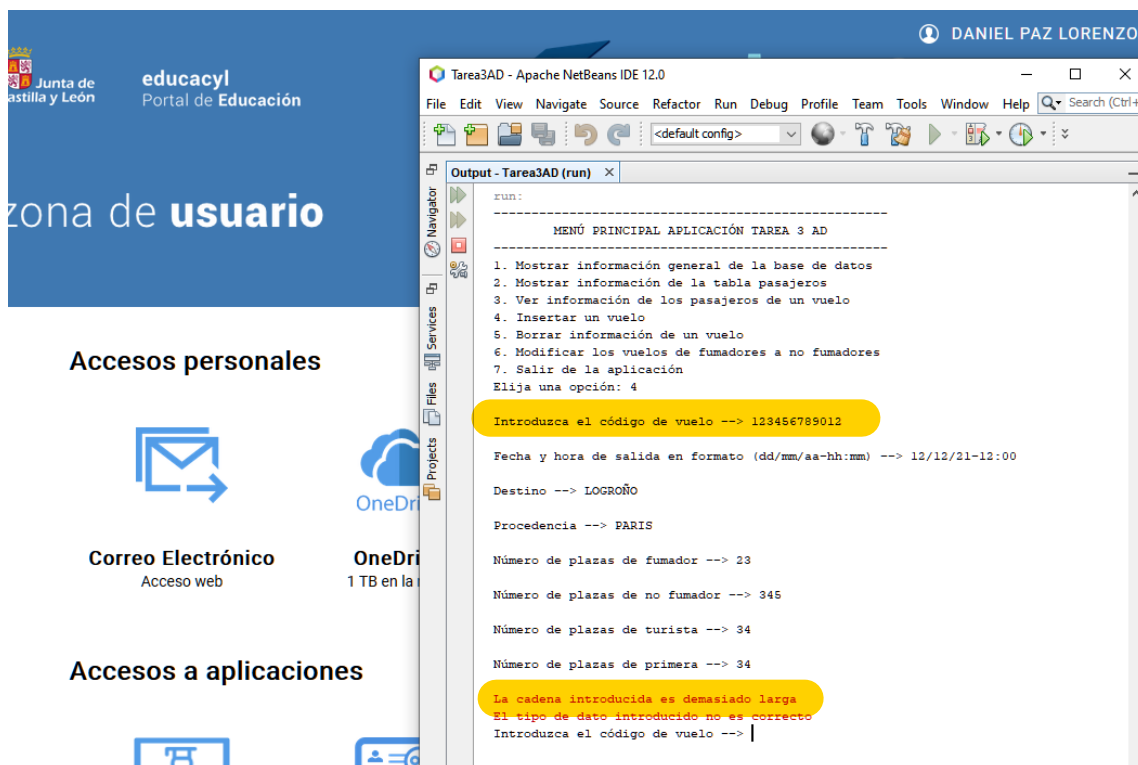


Introducimos los datos que nos solicita la aplicación para posteriormente mandarlos a la bbdd mostrando un mensaje de que se ha realizado correctamente la tarea para que el usuario lo sepa



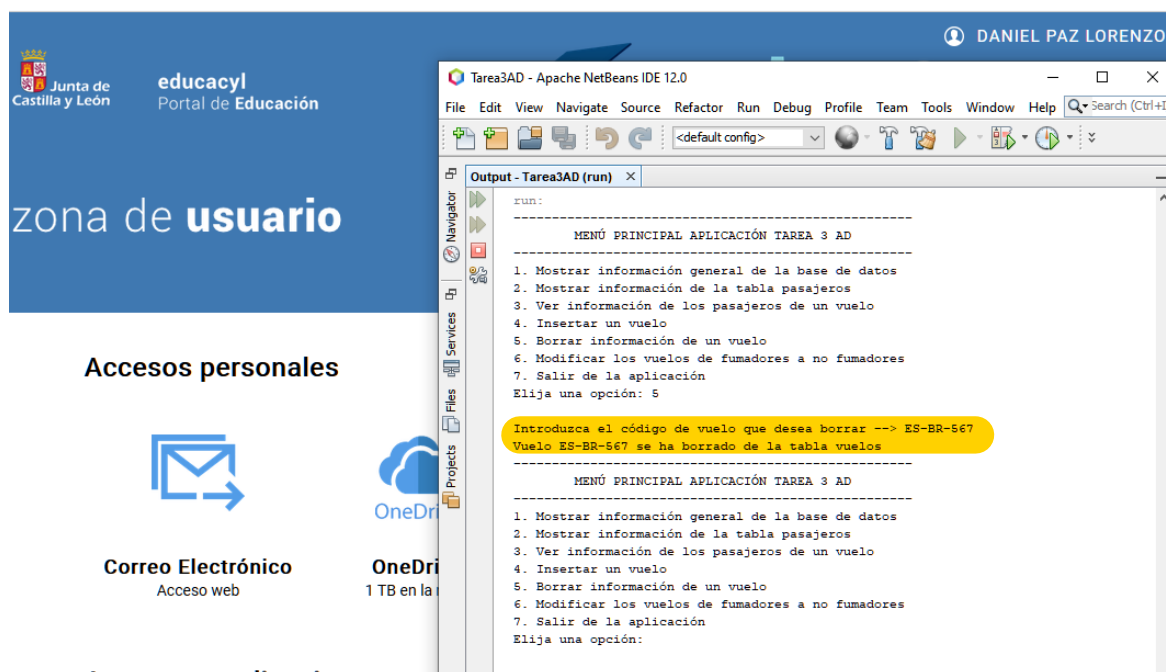
Abrimos nuestra base de datos para comprobar que los datos realmente se han introducido de manera correcta





*Si alguno de los datos introducidos no cumpliera con el tipo de dato o el tamaño máximo establecido en la base de datos lanza una excepción sin introducir estos en la base de datos y vuelve a solicitar los mismos. Con esto evitamos errores en la conexión con la base de datos.*

5. Opción 5: Ahora vamos a borrar la información del vuelo que seleccionemos por código:



*Vamos a seleccionar para este ejemplo el mismo vuelo que hemos introducido anteriormente "ES-BR-567"*

COD_VUELO	HORA_SALIDA	DESTINO	PROCEDENCIA	PLAZAS_FUMADOR	PLAZAS_NO_FUMADOR	PLAZAS_TURISTA	PLAZAS_PRIMERA
AI-1289-9	02/04/99- 14:30	BARCELONA	BONN	150	100	180	70
AI-D7-347	30/03/99- 13:35	BILBAO	MOSCÚ	100	200	210	90
AV-DC-347	29/03/99- 13:35	VALENCIA	ROMA	100	200	210	90
AV-DC2-269	02/04/99-12:00	MADRID	LA HAYA	100	100	180	20
AV-DC9-233	01/04/99- 17:35	VALENCIA	SOFIA	100	100	100	100
FR-DC-4667	28/03/99- 13:30	BRUSELAS	SEVILLA	90	100	160	30
FR-DC2-269	01/04/99- 19:00	CÓRDOBA	MANILA	100	100	180	20
FR-DC7-247	01/04/99-15:35	CÓRDOBA	EL CAIRO	100	100	100	100
IB-98779	02/04/99-08:00	MADRID	LIMA	200	100	250	50
IB-BA-46DC	28/03/99- 12:30	ROMA	MADRID	90	100	160	30
IB-D5-347	01/04/99- 13:35	ZARAGOZA	PARIS	100	200	210	90
IB-SP-4567	27/03/99- 10:30	PARIS	MADRID	100	100	160	40
SP-DC-438	30/03/99- 09:20	MOSCÚ	SEVILLA	90	100	160	30
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

*Comprobamos en nuestra base de datos que efectivamente ese vuelo ha sido eliminado*

6. Opción 6: En este punto se nos pide que todos los vuelos que eran de fumadores pasen a ser no fumadores. Para ello vamos a eliminar las plazas de fumadores y se las vamos a sumar a las de no fumadores dentro de la tabla “vuelos”.

**Accesos personales**

**Accesos a aplicaciones**

**Menú Principal Aplicación Tarea 3 AD**

- Mostrar información general de la base de datos
- Mostrar información de la tabla pasajeros
- Ver información de los pasajeros de un vuelo
- Insertar un vuelo
- Borrar información de un vuelo
- Modificar los vuelos de fumadores a no fumadores**
- Salir de la aplicación

Elija una opción: 6

**Modificados todos los vuelos de fumadores a no fumadores**

*Modifica los vuelos y nos muestra un mensaje de confirmación*

The screenshot shows the MySQL Workbench interface. The 'Schemas' pane on the left displays the database structure, including the 'aerolinea' database with tables 'pasajeros' and 'vuelos'. The 'Query' pane shows the SQL query: `SELECT * FROM aerolinea.vuelos;`. The 'Result Grid' pane displays the query results, which are 20 rows of flight data. The columns are: COD\_VUELO, HORA\_SALIDA, DESTINO, PROCEDENCIA, PLAZAS\_FUMADOR, PLAZAS\_NO\_FUMADOR, PLAZAS\_TURISTA, and PLAZAS\_PRIMERA. The 'PLAZAS\_FUMADOR' column is highlighted in yellow, and the 'PLAZAS\_NO\_FUMADOR' column is highlighted in orange.

COD_VUELO	HORA_SALIDA	DESTINO	PROCEDENCIA	PLAZAS_FUMADOR	PLAZAS_NO_FUMADOR	PLAZAS_TURISTA	PLAZAS_PRIMERA
AI-1289-9	02/04/99- 14:30	BARCELONA	BONN	0	250	180	70
AI-D7-347	30/03/99- 13:35	BILBAO	MOSCÚ	0	300	210	90
AV-DC-347	29/03/99- 13:35	VALENCIA	ROMA	0	300	210	90
AV-DC2-269	02/04/99-12:00	MADRID	LA HAYA	0	200	180	20
AV-DC9-233	01/04/99- 17:35	VALENCIA	SOFÍA	0	200	100	100
FR-DC-4667	28/03/99- 13:30	BRUSELAS	SEVILLA	0	190	160	30
FR-DC2-269	01/04/99- 19:00	CÓRDOBA	MANILA	0	200	180	20
FR-DC7-247	01/04/99-15:35	CÓRDOBA	EL CAIRO	0	200	100	100
IB-98779	02/04/99- 08:00	MADRID	LIMA	0	300	250	50
IB-BA-46DC	28/03/99- 12:30	ROMA	MADRID	0	190	160	30
IB-D5-347	01/04/99- 13:35	ZARAGOZA	PARIS	0	300	210	90
IB-SP-4567	27/03/99- 10:30	PARIS	MADRID	0	200	160	40
SP-DC-438	30/03/99- 09:20	MOSCÚ	SEVILLA	0	190	160	30
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

*Podemos comprobar que se han eliminado las de fumador y se han sumado a las de no fumador*

#### 7. Opción 7: Salimos de la aplicación