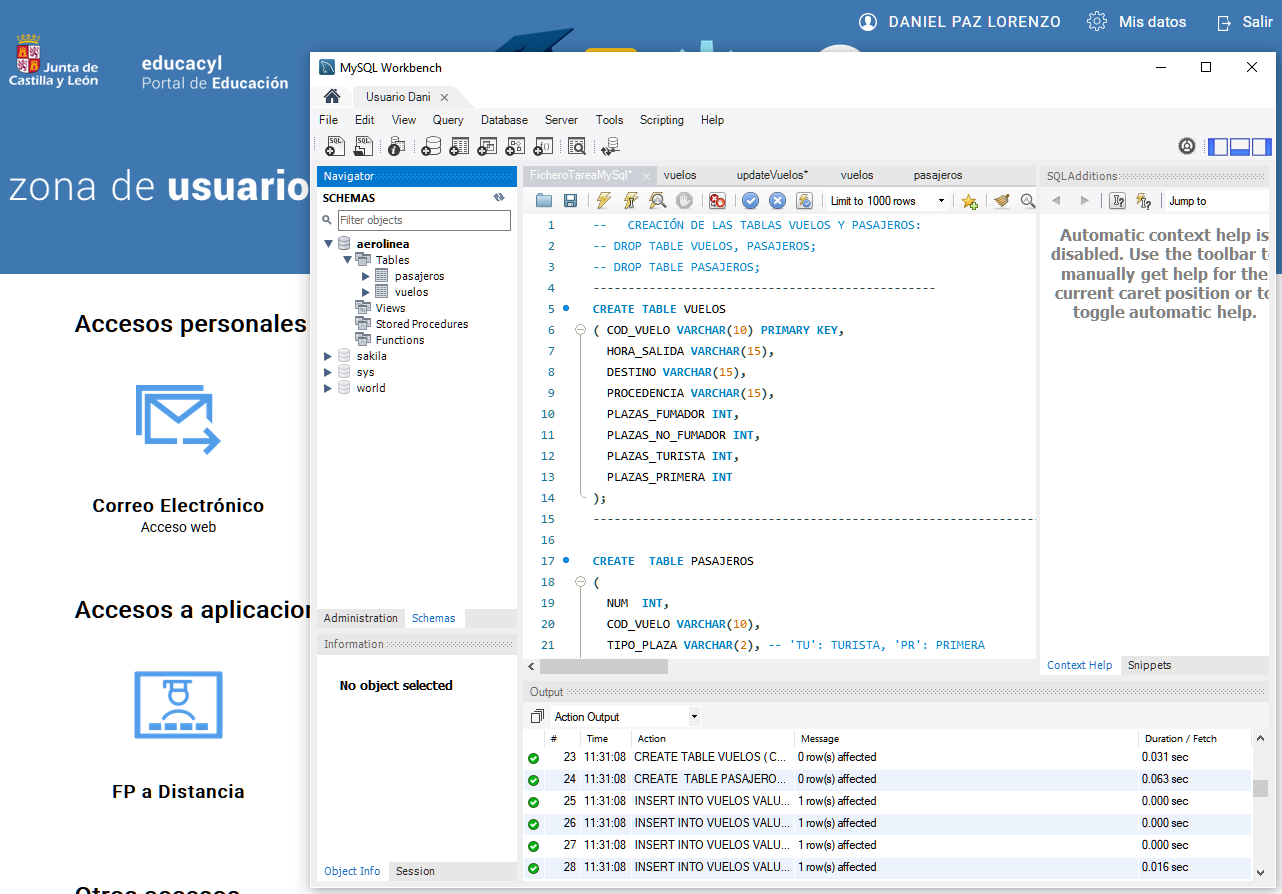
En esta tarea se nos pide que hagamos una aplicación que acceda a la base de datos de una aerolínea y muestre o modifique algunos datos, según la elección del usuario.

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



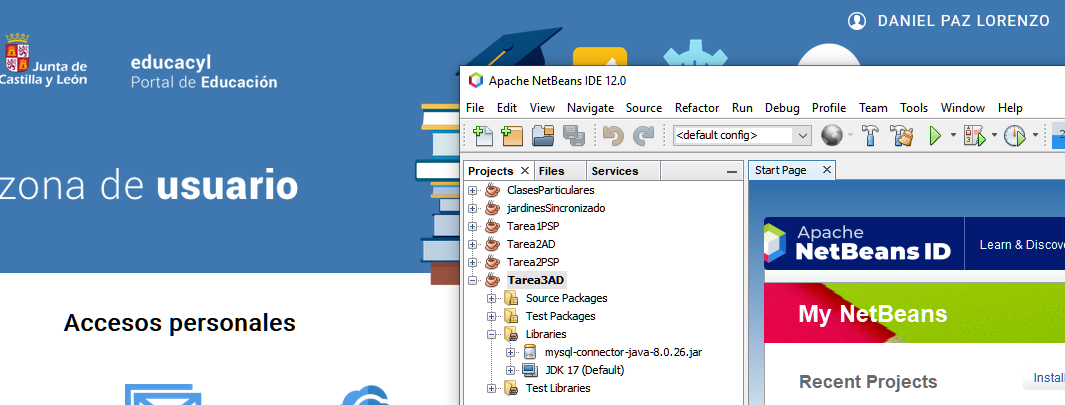
*He utilizado la base de datos de MySQL, he creado un esquema llamado “aerolínea” y dentro del mismo ejecuto el script “FicheroTarea.sql” con algunas modificaciones ya que viene preparado para ejecutarse en Oracle (este fichero modificado lo he guardado con la tarea)*

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.





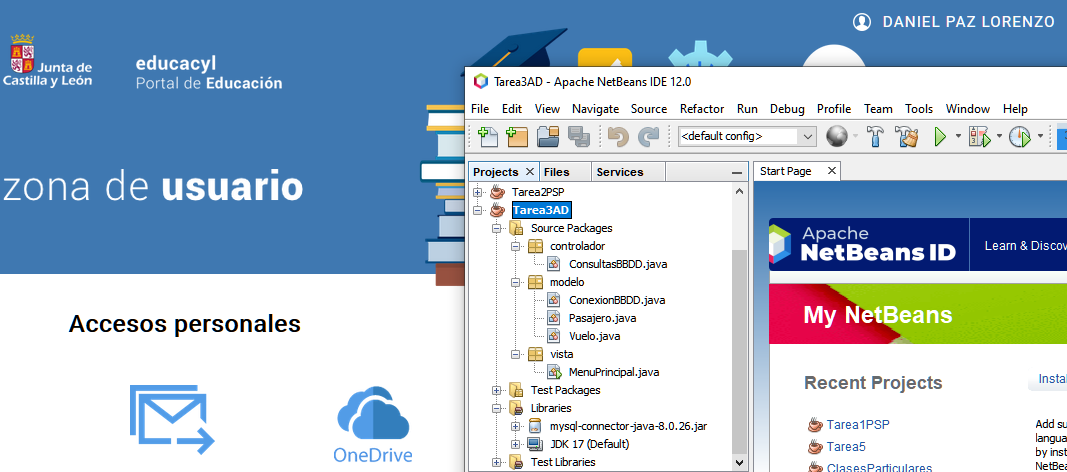
*Lo vamos a descargar desde la propia página de MySQL*





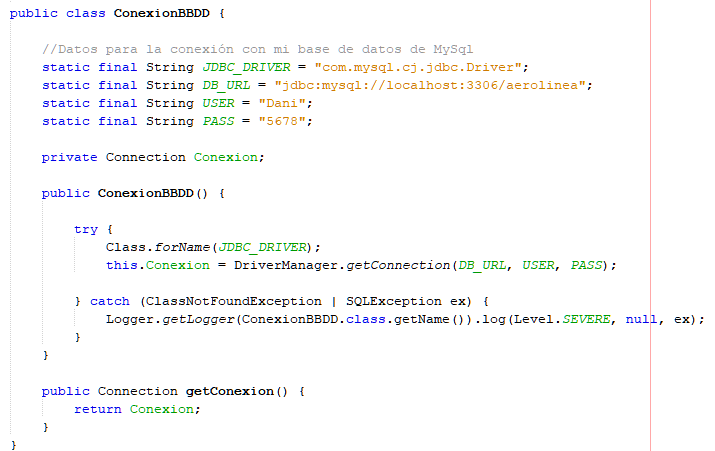
*Una vez descargado lo añadimos como librería a nuestro proyecto en NetBeans*

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.

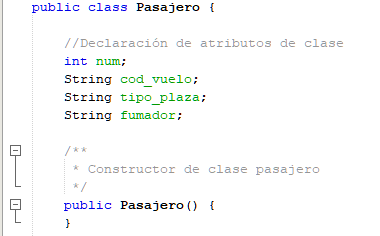


*Dentro de cada paquete crearé las clases que vamos a necesitar como muestro en la imagen*

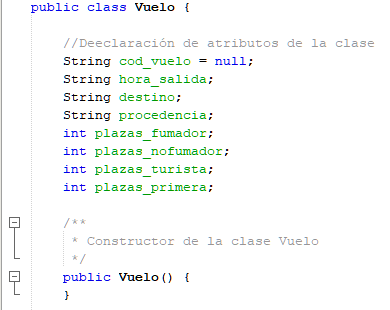
* **Paquete modelo:** Contiene 3 clases:
  + *ConexionBBDD:* En ella vamos a gestionar la conexión con la base de datos mediante un objeto de la clase Conection.



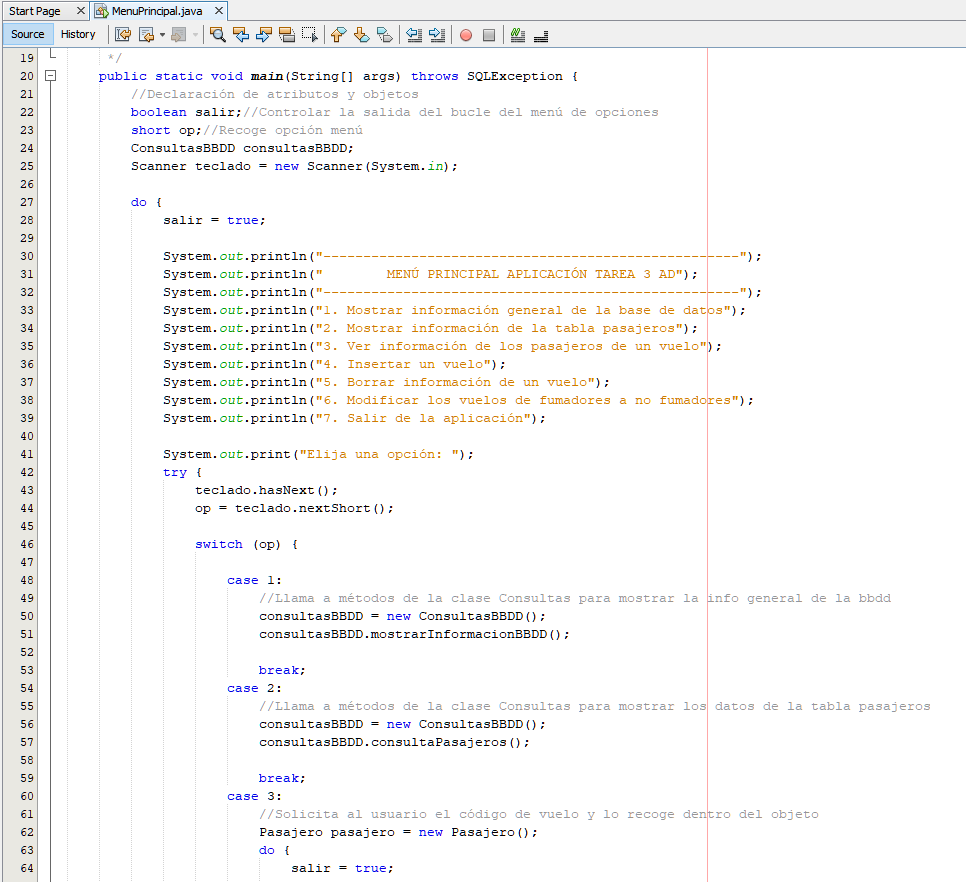
* + *Pasajero:* La vamos a diseñar como si fuera la tabla “pasajero” de la bbdd, referenciamos sus campos como atributos de esta clase. Esto lo haremos para poder trabajar con los datos de la tabla dentro de nuestra aplicación ya que Java trabaja con objetos y la base de datos es relacional.



* + *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.



* **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.



* **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 caractéres

\* **@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 caractéres

\* **@throws** SQLException

\*/

public void borrarVuelo**(**String codigoVuelo**)** **throws** SQLException **{**

**}**

/\*\*

\* Método para modificar los vuelos de fumadores a no fumadores

\*

\* **@throws** SQLException

\*/

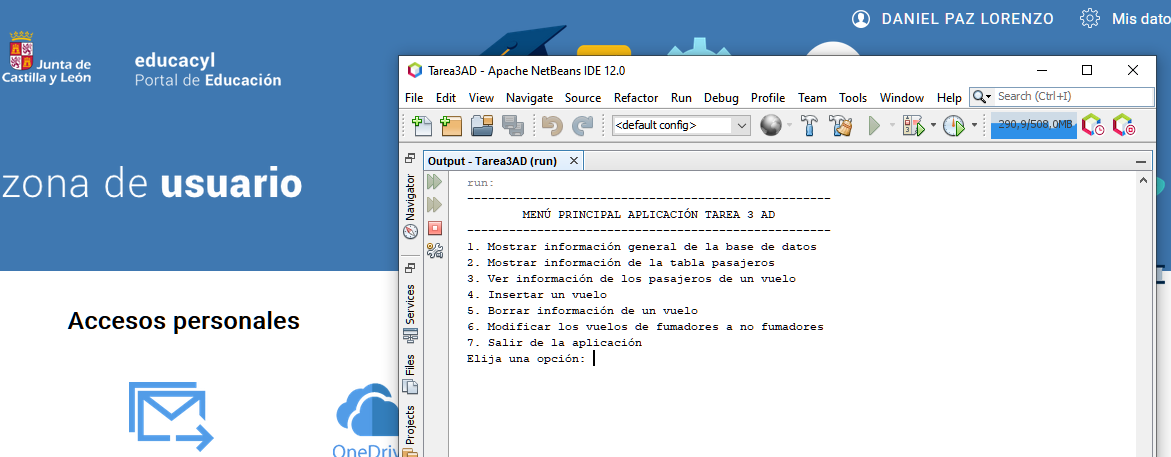
public void modificarVuelosFumadoresNoFumadores**()** **throws** SQLException **{**

**}**

**}**

**PRUEBAS Y FUNCIONAMIENTO DE LA APLICACIÓN**

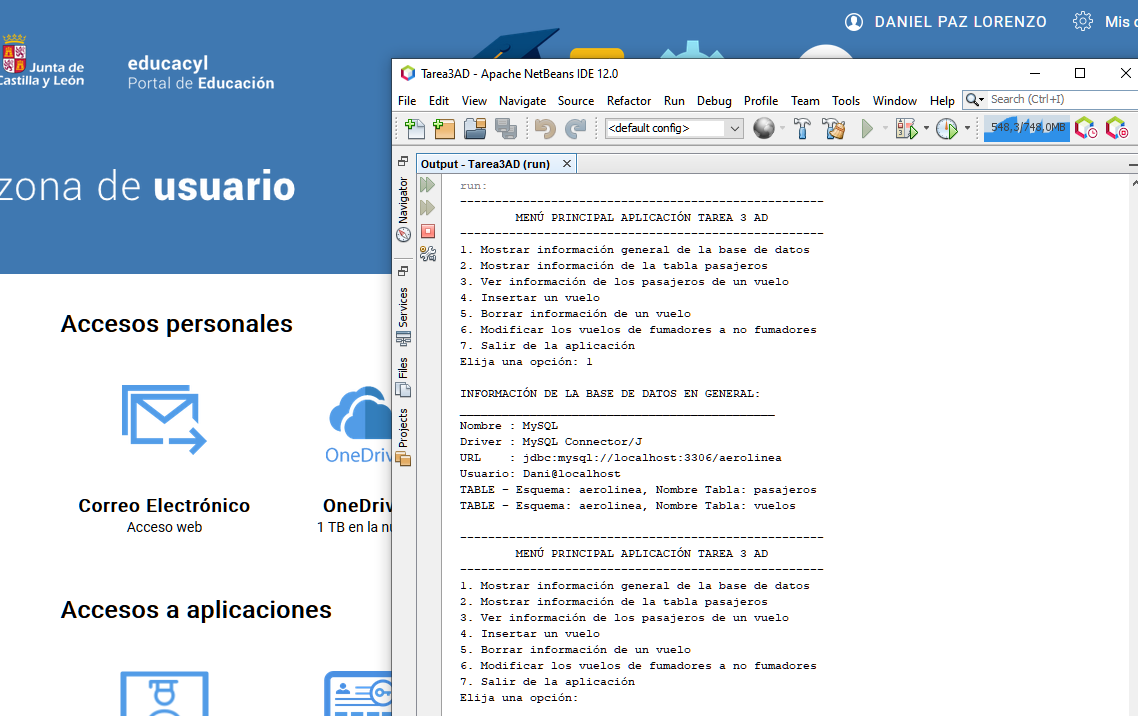
Lo primero que vamos a hacer es una vez que hemos depurado y comprobado que no tiene errores, ejecutar la aplicación.



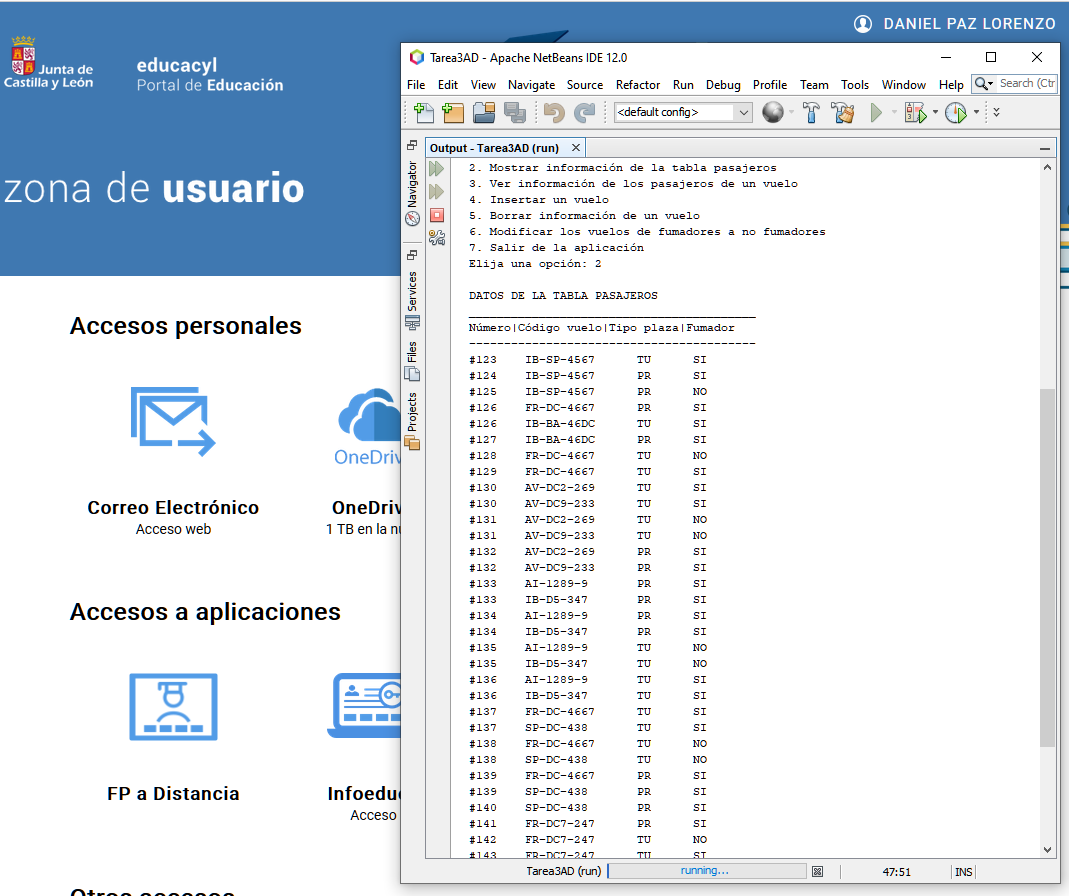
*Nos mostrará el menú principal con las distintas opciones para que el usuario seleccione una*

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

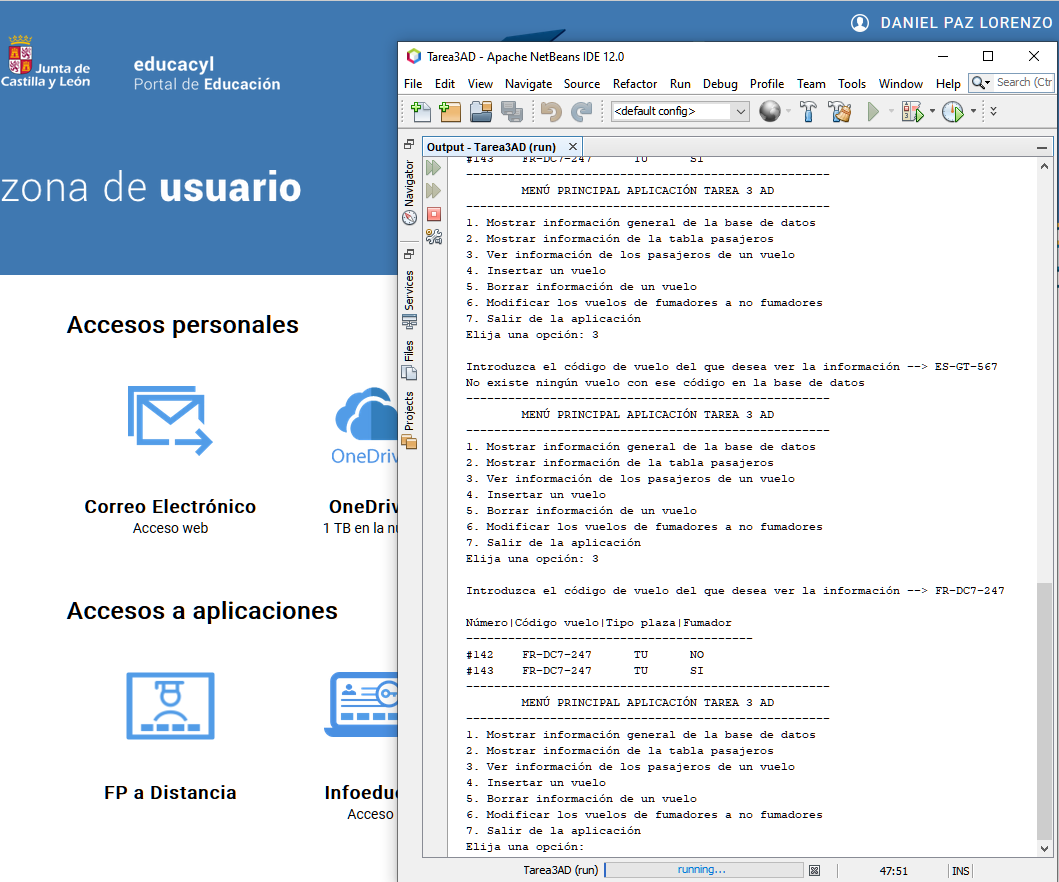
1. Opción 1: Muestra información relacionada con la base de datos en general como son el nombre de la misma, driver, url, usuario, esquema, nombre de las tablas y lo muestra por pantalla. Cuando termina vuelve al menú principal para seleccionar otra opción.



1. Opción 2: Muestra por pantalla toda la información contenida dentro de la tabla “pasajeros”.

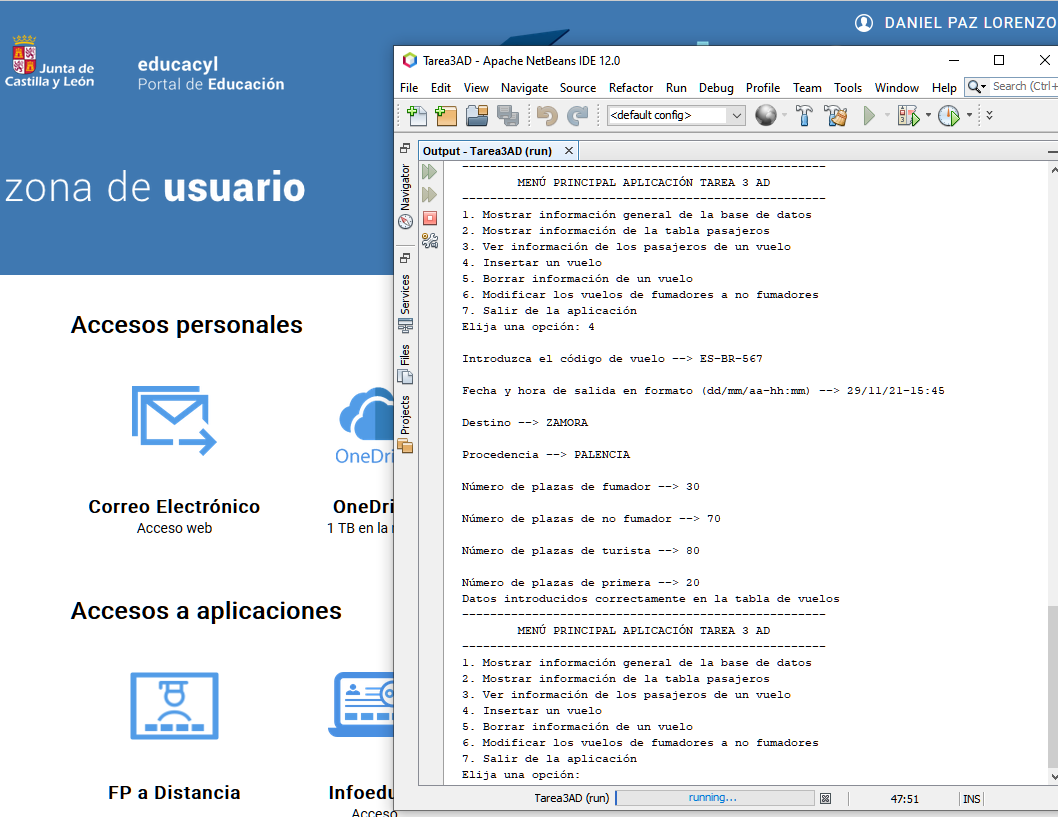


1. Opción 3: Solicita número o código de vuelo y muestra por pantalla la información del mismo. Si este no existiera dentro de la base de datos la aplicación nos muestra un mensaje advirtiéndonos:

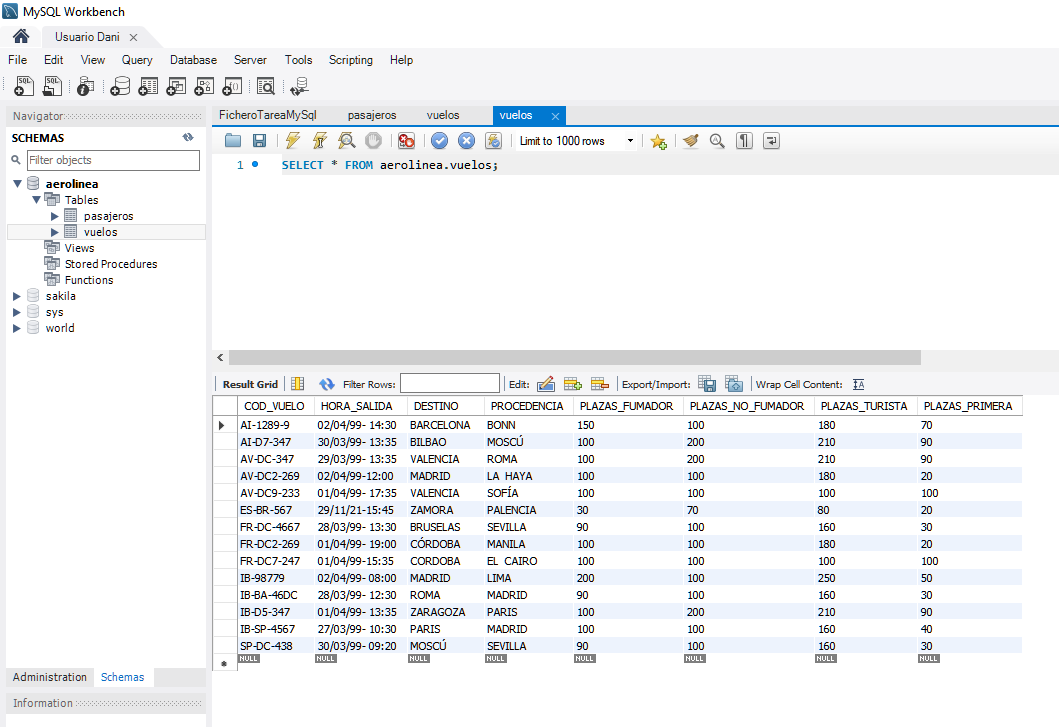


*Comprobamos con un vuelo que no existe y lanza el mensaje y con uno que existe y muestra la información*

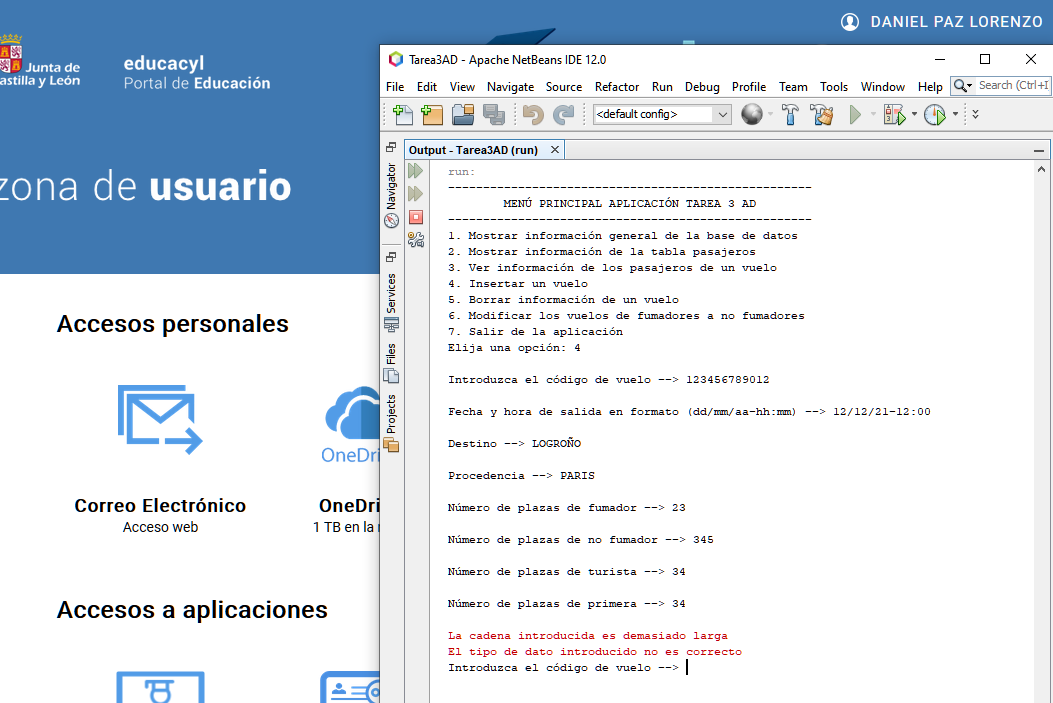
1. Opción 4: Aquí vamos a introducir todos los datos de un vuelo dentro de la tabla “vuelos” de la base de datos.



*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*

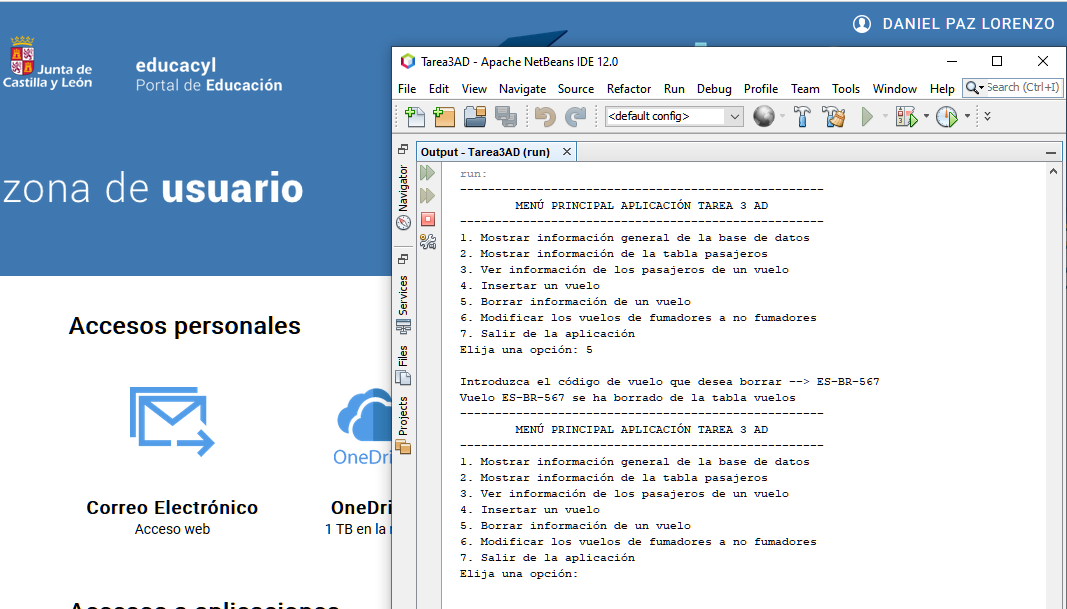


*Abrimos nuestra base de datos para comprobar que los datos realmente se han introducido de manera correcta*

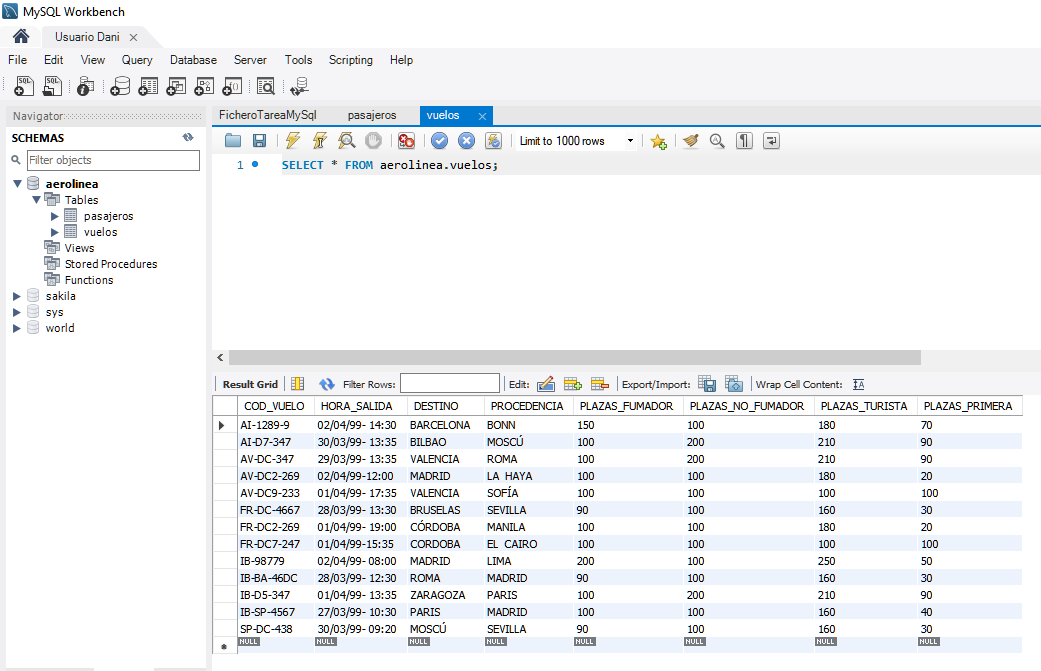


*Si alguno de los datos introducidos no cumpliera con el tipo de dato o el tamaño máximo establecido en la base de datos lanza una excepción sin introducir estos en la base de datos y vuelve a solicitar los mismos. Con esto evitamos errores en la conexión con la base de datos.*

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

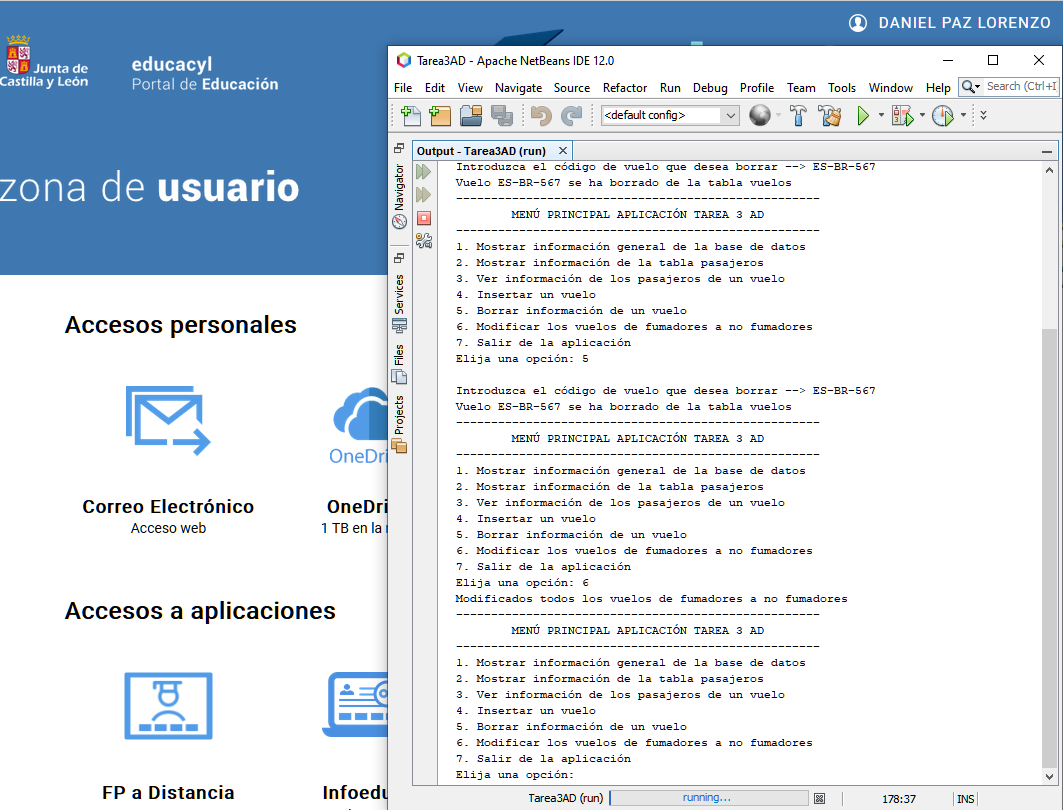


*Vamos a seleccionar para este ejemplo el mismo vuelo que hemos introducido anteriormente “ES-BR-567”*

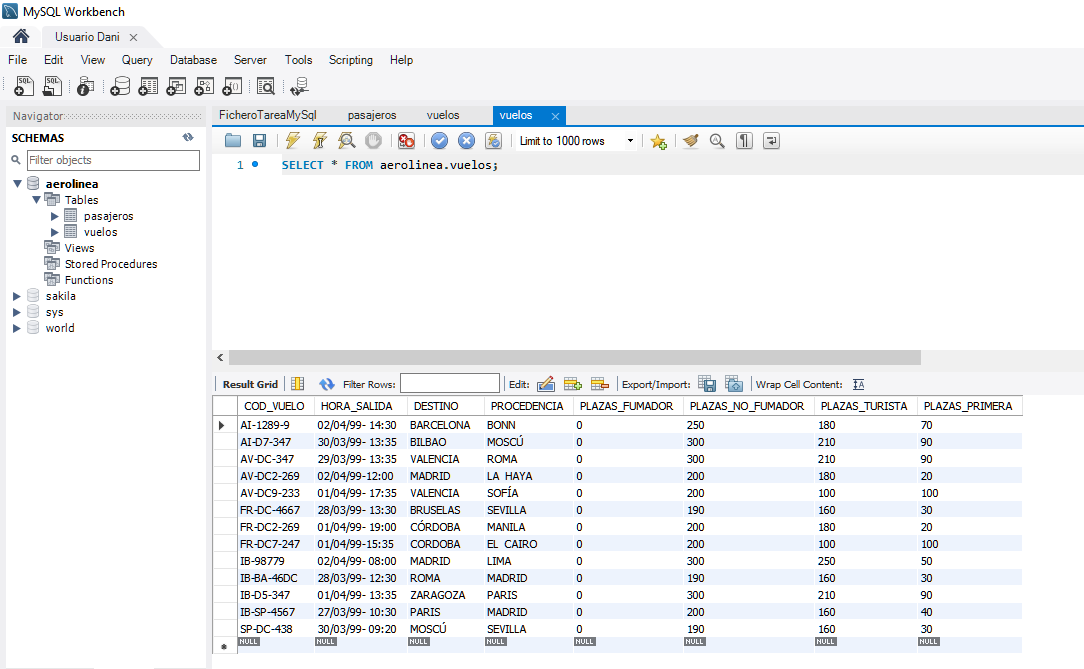


*Comprobamos en nuestra base de datos que efectivamente ese vuelo ha sido eliminado*

1. 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”.



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



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

1. Opción 7: Salimos de la aplicación