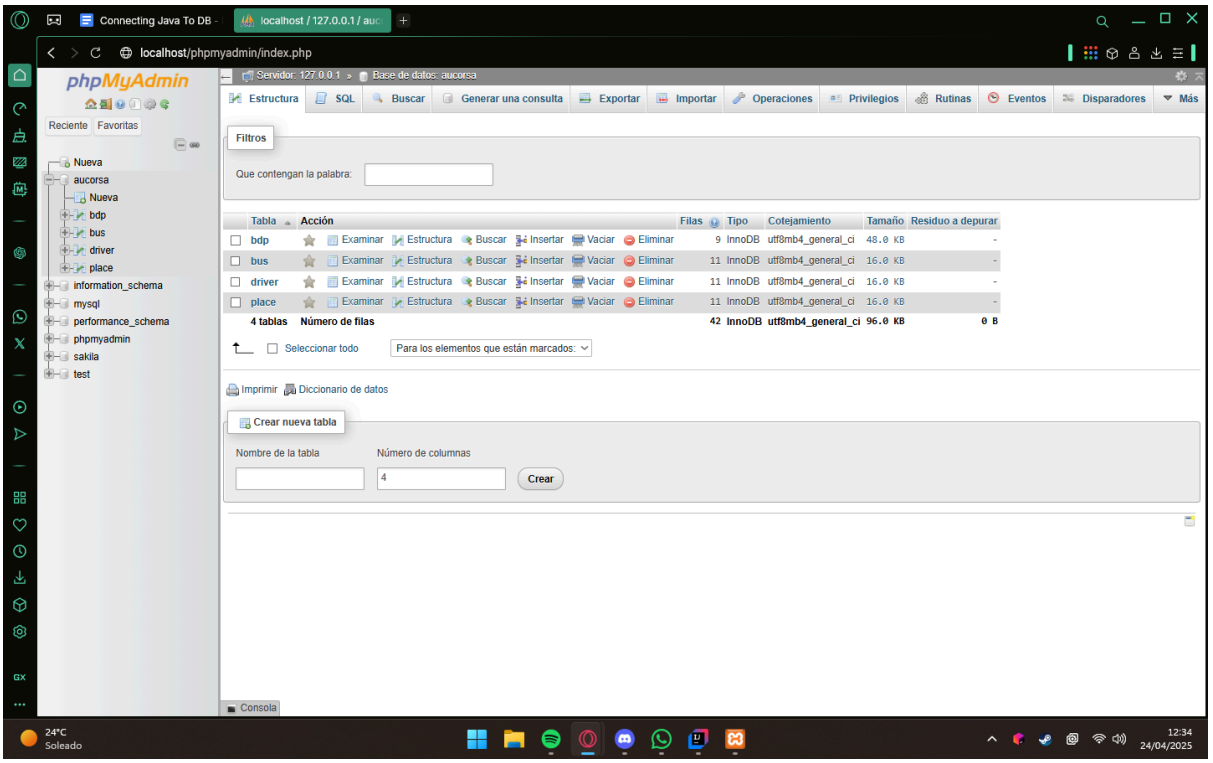
Connecting Java To DataBase

First of all, I installed XAMPP, “a free and open-source web server solution stack package that provides a local development environment for web developers”. It includes the Apache HTTP Server and MariaDB database.

Once MariaDB was operational, I used the script posted in moodle. First, I intended to build the database by hand, however I was running out of time so I used the premade one. So, by using phpMyAdmin, I created the database “aucorsa” and then run the sql file.



Then, after installing the jdbc driver, I was finally able to start with the code. What I did was a class for handling the connection and a second one to function as Main.

The Controller Class

It consists of several actions to execute later in the Main method.

```
public class Controller {  7 usages

    Connection conn = null;  4 usages
    // Custom Exception
    public class NotSelectQuery extends Exception{...}

    // Starting and closing the connection
    public Controller (String user,String pass){...}

    public void closeConnection () {...}

    // Generic update for later use
    private void executeUpdate (String query) throws SQLException {...}

    // Methods to insert into the 4 different tables
    public void insertBus(String register, String type, String license) {...}

    public void insertDriver(int numDriver, String name, String surname) {...}

    public void insertPlace(int idPlace, String cp, String city, String site) {...}

    public void insertRoute(String register, int numDriver, int idPlace, String dayOfTheWeek) {...}

    // Specific actions that the exercise requires
    public void updateDayOfWeek(int numDriver, String dayOfWeek){...}

    public void deleteRoute(String register, int numDriver, int idPlace){...}

    // A method that only accepts Select queries
    public void executeSelectQuery (String query) throws NotSelectQuery {...}

    private void printQueryResultSet (ResultSet rs) throws SQLException {...}
```

I forgot that *preparedStatement* existed so, what I did is use *String.format()* for making the sentences. For example, here we have the method *insertDriver* where I've used *String.format()* to introduce the *numDriver*, *name* and *surname*.

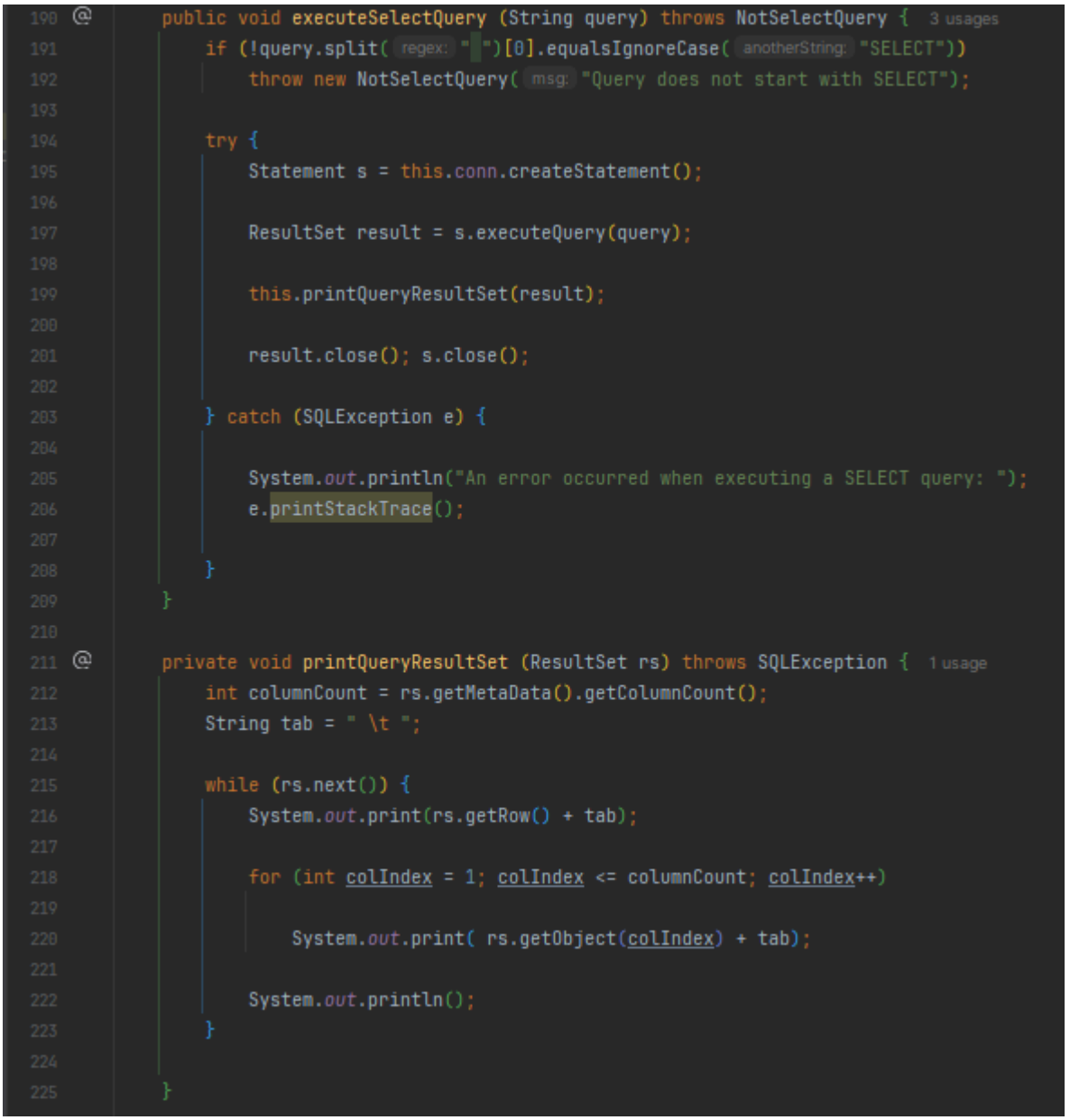
```
85      public void insertDriver(int numDriver, String name, String surname) {  1 usage
86
87          try {
88
89              this.executeUpdate(
90                      String.format(
91                              "INSERT INTO driver VALUES (%d, '%s', '%s')",
92                              numDriver, name, surname
93                      )
94              );
95
96          } catch (SQLException e) {
97
98              System.out.println("An error occurred when inserting into table DRIVER: ");
99              e.printStackTrace();
100
101         }
102
103     }
```

The rest of the methods use the same gimmick and are pretty straightforward. The ones that are different are the last two. Let's take a look at them.

The first method is for making a Select query. For making sure that is a select query, it checks that the first word of the query is “SELECT” (ignoring lower and upper case). If it is not “SELECT” it throws a custom exception defined earlier in the class. Other than that, this method does not have much more.

So, going a bit further down we have *printQueryResultSet*. This uses a while loop using the *iterator* from the *ResultSet*. What I want to point out is that I had issues with getting the data. This is because from the resultset you are obligated to specify the type of the value that you want to get.

However, I wanted to get the data without specifying the type, because this method is for printing any query from any table. So, what I found out is that the *resultSet* has a *getObject()* method. This allows you to get any type of data, perfect for printing.



```
190 @   public void executeSelectQuery (String query) throws NotSelectQuery {  3 usages
191         if (!query.split( regex: " ")[0].equalsIgnoreCase( anotherString: "SELECT"))
192             throw new NotSelectQuery( msg: "Query does not start with SELECT");
193
194         try {
195             Statement s = this.conn.createStatement();
196
197             ResultSet result = s.executeQuery(query);
198
199             this.printQueryResultSet(result);
200
201             result.close(); s.close();
202
203         } catch (SQLException e) {
204
205             System.out.println("An error occurred when executing a SELECT query: ");
206             e.printStackTrace();
207
208         }
209     }
210
211 @   private void printQueryResultSet (ResultSet rs) throws SQLException {  1 usage
212         int columnCount = rs.getMetaData().getColumnCount();
213         String tab = " \t ";
214
215         while (rs.next()) {
216             System.out.print(rs.getRow() + tab);
217
218             for (int colIndex = 1; colIndex <= columnCount; colIndex++)
219
220                 System.out.print( rs.getObject(colIndex) + tab);
221
222             System.out.println();
223         }
224
225     }
```

Then, it is just a matter of calling the controller and using its methods.

```
public static void main(String[] args) {

    Controller controller = new Controller(getProperty("user"),getProperty("pass"));


    insertValues(controller);

    update(controller);

    deleteRoute(controller);

    querys(controller);

    controller.closeConnection();

}
```

Like getting the data from a busdriver by the identifier.

```
private static void querys(Controller controller) {  1 usage
    try {

        int numdriver = 111;
        System.out.println("Driver data from numdriver : " + numdriver);

        controller.executeSelectQuery(
            "SELECT numdriver, name, surname " +
                "FROM driver " +
                "WHERE numdriver = " + numdriver
        );

        System.out.println();
```

Or insert some values into the tables.

```
private static void insertValues(Controller controller) {  1 usage

    controller.insertBus( register: "B012", type: "Escolar",  license: "LIC011");

    controller.insertDriver( numDriver: 111,  name: "Arnold", surname: "Schwarzenegger");

    controller.insertPlace( idPlace: 11, cp: "29680",  city: "Estepona",  site: "Reloj");

    controller.insertRoute( register: "B012",  numDriver: 111,  idPlace: 11,  dayOfTheWeek: "Sunday");

}
```

I also did a properties file, where I store the user and password. With this simple method, I can get the property by name.

```
25      private static String getProperty(String propertyName) {  2 usages
26
27          String propertyValue = "";
28
29          try {
30
31              Properties properties = new Properties();
32
33              properties.load(new FileInputStream( name: "src/data.properties"));
34
35              propertyValue = properties.getProperty(propertyName);
36
37
38          } catch (IOException e) {e.printStackTrace();}
39
40          return propertyValue;
41      }
```

Final thoughts

I know this could be 100 times fancier, for example, doing classes to store the data from tables. I will develop a project (with GUI) with a better structure.