DOCUMENTATION

ASSIGNMENT *3*

Orders Management System

STUDENT: VĂLCĂUAN LAURA-MARIA
GROUP: 30422

TABLE OF CONTENTS

1.THE PURPOSE OF THE PAPER

The purpose of the paper is represented by the implementing and design of a management system of a warehouse with a graphical interface which has the aim to make life easier to the user and to improve the adding data process (the clients, the products, the orders) by the user and the possibility of the user to edit or delete items/inputs which already exists in the database. The management system must be easy to use and easy to understand by any user besides the level of knowledge in technology.

Side objectives considered in case of designing the main objective are:

1. Clearly state the main objective and sub-objectives required to reach it: realize the classes and packages which will be part of the composition of the computer system.
2. Analyze the problem and define the functional and non-functional requirements. Creating the graphical interfaces: making interfaces which will be used by the user to add, visualize and data processing.
3. Design the solution. The implementation of a parsing and storage method of the input data introduced by the user.
4. The database connection to the application in which the information about the clients, products and orders will be stored.
5. Implement the solution. The validation of the data introduced by the user.
6. Test the solution.

2. PROBLEM ANALYSIS, MODELING, SCENARIOS, USE CASE

SQL is a programming language used by almost all relational databases, for querying, managing and defining data, as well as for controlling the provision of access.
SQL was first developed by IBM in the 1970's with the help of Oracle which led to the implementation of the ANSI SQL standard, later SQL extending from companies
such as IBM, Oracle and Microsoft. Although SQL is still widely used, it appears in
continue new programming languages.

Database software is used to create, edit, and maintain files and database records, making it easier to create files and records as well as data entry, editing, updating and reporting. The software also manages data storage, backup and reporting, as well as multi-access control and security. At the moment,

Strong database security is especially important because data theft a has become more and more common. Database software is sometimes referred to as a “system database management ”(DBMS).

Database software simplifies data management, allowing users to store the data in a structured form and then access it. It usually has an interface graphics to help create and manage data and, in some cases, to enable users to create their own databases using software.

A database usually requires comprehensive software for this program known as the database management system (DBMS). A DBMS serves as the interface between the database and end users or programs, allowing users to download, update and manage how information is organized and optimized. Also a DBMS facilitates the supervision and control of the database, allowing administrative operations such as performance monitoring, adjustment, backup and recovery.

MySQL is an open source relational database management system SQL. It has been designed and optimized for web applications and can run on any platform. With the advent of new requirements on the Internet, MySQL has become the preferred platform for web developers and for the development of web-based applications. Because it is created to process millions of queries and thousands of transactions, MySQL is a popular option for electronic trading companies, which have to handle numerous money transfers. The main feature of MySQL is on-demand flexibility.

Reflection is the ability of a program to perform processing on the program itself, and in particular the structure of its source code. A Java program to run, for example, can examine the classes from which it is made, the names and signatures of their methods and so on.

Reflection in general can also allow a program to dynamically change its structure.

For example, it is technically possible to create applications capable of applying packages update while running, dynamically replacing parts of their code.

The solving of the purpose problem can be divided in the next steps:

- The access to the database.
- Take the needed data from the respective database.
- The storage of the data obtained in specific objects.
- Display the results processed by the user through the graphical interface.

We can deduce that the application which will be designed will have to respect a well predefined structure to ensure the correctness and security of the processed data, added or updated through the database.

There are more others scenarios and use cases of the application, as shown above:

- Adding to a new line in the table (as well for the client and product).
- Updating a new line in the table.
- Deleting a line in the table.
- The view of the data from the database.

For each case the necessary elements for making this action are well defined and specified in the graphical user interface and in the implemented code.

3. DESIGN

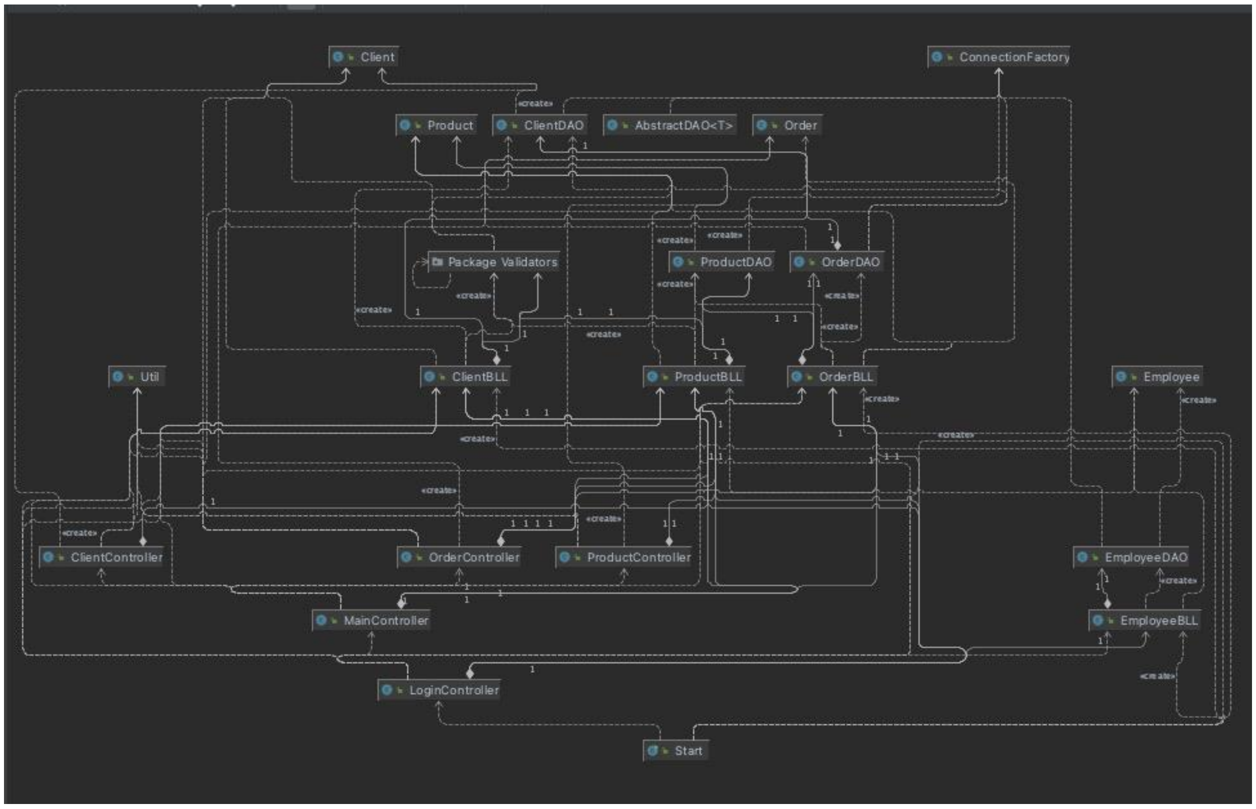

Figure 1 - Class diagram

The application contains more main classes which have the scope to define the application and the graphical user interface and the tables in which the information will be displayed in the database.

For the design of this application the classes were organized by the model layered architecture which has at the base the information transmission from a layer to another (packages and classes). This model makes the maintenance of the program easier and ensures the security of the data and their integrity by the fact that the package BussinesLogic, the data added by the user is verified before its update or to be added in the database.

For the data access is used the reflection technique, which is implemented in an abstract class which is extended in any other classes which uses a table type. So to add or to delete some type of data there is no need for major modifications of the program. This method is used also for the table creation from the graphical user interface with the user. The reflection is used from the fact of generalization, because using this method we don’t have to make modification to the data access logic of the data because the methods from the abstract class automatic extract the set and get methods populating them with current data.

The data structures used are grouped in the Model layer of the project being implemented as elements of the database. This classes contain the exact same type of primitives variables which are accessed through reflection with the help of the methods get and set. In this package are stored the necessary methods to create the table structures in which will be displayed the data.

The graphical user interface is divided in many windows each with a different role.The main Interface (GUI) presents 3 buttons (Clients, Products, Place order) which when pressed will open a new window.

The ClientGUI window will be generated by pressing the Clients button and consists of object where the data itself will be stored as well as 3 buttons (Add Client, Update,Client, Delete Client) which will be used to process the data in the table. Additionally there is a button (Reload table) that will update the table in the GUI after modification of the table data. We have chosen this implementation for an easier observation of changes made within the table.

The ProductGUI window will be similar to the ClientGUI window, the only difference being that the table will display the information from the Product table.

The OrderGUI window shows 2 JTable objects where both stored clients will be displayed in the database as well as the available products. The Add to order button will add a selected product to an order, this can be seen by clicking the View order button which will display all products added to the current order and give the user the option to remove products from the order.

For all update and new data entry options, when the button responsible for performing the action is pressed, a new Form window will be generated through which the user can easily enter the desired information.

4.IMPLEMENTATION

In order to implement the model chosen for the application, we have divided the classes into several packages, depending on their functionality and role:

•connection: this package is used to establish a connection with the database implemented in MySQL Workbench using the ConnectionFactory class. In this class methods for creating the connection and closing the connection are implemented.

•dataAccess: in this package an abstract class AbstractDAO is defined in which are implemented the basic methods for selecting, deleting, updating and inserting new data into the database tables. These operations are performed by establishing the with the database using the package described above, and then using methods methods that return a

•businessLogic: Inside the methods in the classes of this package is the implementation of the application logic and how data is passed to the user through the GUI. Also, also within this package is implemented the validator_new interface which will have the role of checking the data entered by the user and in case of a wrong entry will return a specific exception.

•view: this part of the application deals with retrieving data from the layer and processing it so that it takes a user-friendly form. In the case of this application, the data passed as a list of objects is entered in a table, and then displayed in a separate panel. The user can use a variety of buttons to select the type of operation they wish to perform. After selected an operation, the user can enter any values in the fields specific to each operation. table. For example, if the user wants to enter a new product into the database, he button first to open the table responsible for storing the product. product data, then click on the Add product button and then he will be able to enter each field in the table into a series of JTextField generated in a new window. The data is validated only after pressing the OK button, when the SQL statement is generated and passed to MySQLWorkbench for execution. Table creation is based on reflection, the constructor of the class responsible for this activity accepting a class from the "model" package from which it will extract the write methods and read methods ("getters" and "setters") and a list of objects of type model class from the which will extract the actual data.

5.RESULTS

The resulting application is an interface capable of connecting to a SQL database and providing the ability to extract, modify and delete any type of data stored within the database.

For the user, the application is intuitive and easy to use, with most of the interaction between the user and the interface being performed through buttons and basic operations of selecting an object from the table with a simple click. The table can also be customised by allowing the user to change the order and size of the table columns by dragging with the mouse. Thanks to the chosen implementation model, the application is easy to maintain and modify by the programmer.

6.CONCLUSIONS

In this project, the knowledge of Java was deepened and the connection to a database was made. It also highlighted the importance of a well-organised layered data model which brings with it a lot of benefits. Further developments include: offering the possibility to log into the account as a client, introducing roles and offering different access permissions depending on the role of each user.

7.BIBLIOGRAPHY

1.https://www.baeldung.com/java-jdbc
2.Java Reflection Tutorial (jenkov.com) Java Reflection Tutorial (jenkov.com)
3.MySQL :: MySQL Workbench Manual :: 6.5.2 SQL Data Export and Import Wizard
4.Connect to MySQL with JDBC driver - Mkyong.com
5. https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf