SDJ3 Project Documentation

Group members:
Claudia Nou (237425)
Mindaugas Budrys (240228)
Oskaras Goborovas (240279)

Warehouse System

The main goal of this system was to implement a system for a warehouse using a three-tier architecture.
A class diagram representing this architecture can been see below and can show in a more clear way how the structure is implemented and how the tiers connect between each other.

First tier

Our first tier represents the Java RMI Client and the C# Web Service Client, which are what we use to send the information of the products that should be retrieved. In real life, this would be the UI which the workers at the warehouse or outside would use.

Second tier

The second tier represent the main server, which is the one that connects to the third tier (when some data must be retrieved from the database) and to the first tier (when some data must be retrieved from the user). This tier is the one that makes all the calculations, and knows “where to go” and “what to do” when an operation has been sent from the user.
It contains a Java RMI Client that connects to the Java RMI Database Server which retrieves the data from the database.
So, basically, this tier is connected using RMI and Web Service to the first tier - To interact with the user.
And is connected using RMI to the third tier - To interact with the database.

Third tier

The third tier represents the communication with the database. No connection with the database can be found if it is not done through this tier.
This does not mean that this tier contains directly the database, but the classes that allow the system to connect with it.

RMI System

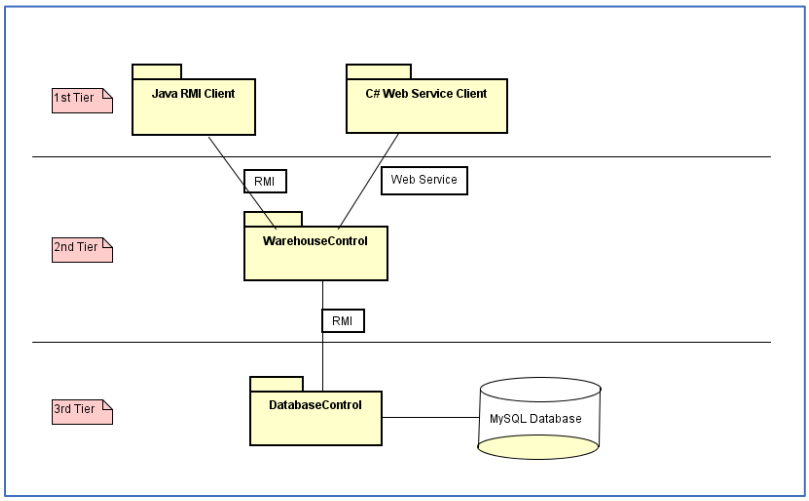
The implementation of the RMI for our project is the connection between the Java console (The UI of the workers) and the server of the second tier. As well as the connection between the tier

two (business logic) server and tier three (database) server which is used to retrieve/update information in the database and do something with the data.
In real life, the tier one to tier two RMI clients could be every computer available in the warehouse and outside, containing different features depending on which kind of user is meant to use it. For example, the manager of the warehouse would not have the same options than the client located at his small supermarket willing to order some products from the warehouse.

Web Service

The implementation of the Web Service in our project is the connection between the C# Client, which is the UI for the workers and allows them to interact, as they would do it using the Java RMI Client, with the system.
We created a class named WebServiceBridge in Java, which implements the methods that are defined as the web service. This “bridge” allows the Web Service to send and receive the strings and the Array of strings that need to send from the user using the C# Client to the system and vice versa.

Architecture Diagram and Class Diagram



The Class Diagram can be found together with the source code of the project in the folder where this document is located.

Warehouse Database

To keep track of the goods available at the warehouse, a database must be retrievable from the system.
A database for a real warehouse could be huge and complex, but in our case, in order to focus on the most complex features of the project, we kept the database as simple as possible.
Two basic tables were created, ‘Products’ and ‘Pallets’.

The table 'Products' contains the data of the different products that the warehouse stores or has stored previously. Its primary key is the 'product_id' which the system uses to identify which product should retrieve from the pallet when an order containing that product is made.
It also contains a column 'name' which will describe the specific model of the product. For example, Jysk offers pillows, but there are thousand different models of pillows. This name tag will differentiate each model.

The column 'type' contains the kind of product, in other words, its main category.[2] (i.e. chair, table, pillow, towel…)
The table 'Pallets' contains the data of the different pallets that the warehouse has stored. Its primary key is the 'pallet_id' which the system uses to identify which pallet to retrieve the products from.
It also contains a column 'product_id', this is the foreign key that connects it with the table 'Products'. This foreign key contains the data to know which model of products is stored in the pallet.
The column 'shelf_id' contains the number of shelf which the pallet is located at.[1]
The column 'quantity' contains the number of products which the pallet is containing. This number will be used when an order will be done and a specific number of products should be retrieved. Different algorithms could be implemented to obtain the products of the pallets as fast and as optimized as possible, but this is an out-of-scope parameter for our project.

[1] The attribute 'shelf_id' in real life would be a foreign key which would connect to a table containing the data of each shelf, its location in the warehouse, the quantity of products, etc…

[2] The attribute 'type', in real life would be an id which would connect to a table containing the id and the type of product as text data.

Setting up the database

To set up and have the database ready to be connected to, it is needed to:

- Start the Apache and MySQL modules in XAMPP
- Connect to the MySQL database
- Execute the sdj3_warehouse.sql file inside the database

Testing the project

To test the project after having the database set up, it is needed to:

- Start the database server in eclipse (DatabaseServer class)
- Start the tier two server in eclipse (MyServer class)
    - To run the Java Client: Run the MyClient class in Eclipse.
    - To run the C# Client: Unzip the WebServiceClient.zip file and start the WebServiceClient in Visual Studio.