Managementul unei Farmacii

Student:Bratean Larisa

**Group:30235**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 4

3. System Architectural Design 6

4. UML Sequence Diagrams 6

5. Class Design 7

6. Data Model 7

7. System Testing 8

8. Bibliography 9

1. Requirements Analysis

# Assignment Specification

We have an application for the employees of a pharmacy. The application have two types of users (a regular user represented by the employee and an administrator user) which provide a username and a password in order to use the application. The regular user can do operations based on medicine, and the administrator can do operations based on medicine and also employee’s informations.

# Functional Requirements

The regular user can do:

* can search medication by name, ingredients or manufacturer;
* sell medication to a certain client.

The administrator can:

* create, read, delete or update the quantity of a medicine;
* create, read, delete or update the address of an employee;
* can generate two types of files, one in pdf format and one in csv format, with the medicines that are out of stock.

# Non-functional Requirements

* + 1. **Availability**

The system is available 100% for the users of the application, 24 hours a day, while the Java application is running.

* + 1. **Performance**

The application it s develop to work really fast, answer fast when the regular user or administrator tries to make an operation, no matter if they try to add, delete, view or update something.

* + 1. **Security**

The application it’s safe, because both the administrators and the regular users have to log in before entering it. They have to enter an username and a password, which is encrypted with the special character \*.

* + 1. **Usability**

The system allows users to access the application using Java, so it’s a desktop application. Nobody needs training to use it, because we run it and the interface it will open, and you just need to log in. The system it’s friendly and simple.

2. Use-Case Model

2.1 **Employee**

The employee has the purpose to log in successfully in the application, and make operations based on medications dates. He can also sell medicine to a client.

He has an important role for selling medicine, and making sure the client has enough money, and the product is on stock. He also can search and see all the products that have a specific name, manufacturer or ingredients.

A successful scenario would be: the employee is trying to log in the application, and succeeds. Before he tries to sell something, he is looking for the products with some specific information, and he finds them. After, he is selling a product to a client, that has enough money to buy the product.

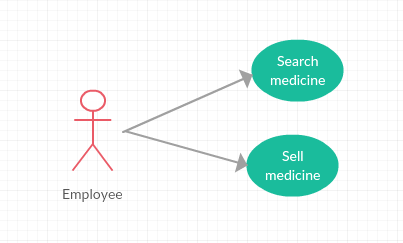
A success-failure scenario would be:

* Preconditions: the employee tries to log in to see some product
* Postconditions:

Success : the employee logs in the application, and see the products

Failure: the employee can’t log in the application, because he made a mistake on his password

Use case:



2.2 **Administrator**

The administrator has the purpose to log in the application, and make different operations based on the information about employees and medications. He can also generate reports about the medications that are out of stock.

He has an important role for taking care of the farmacy.

A successful scenario would be: the administrator logs in the application, and wants to add a medication. He writes the name, manufacturer, ingredients, available quantity, and the price of the medication, and a new medication is saved in the XML file.

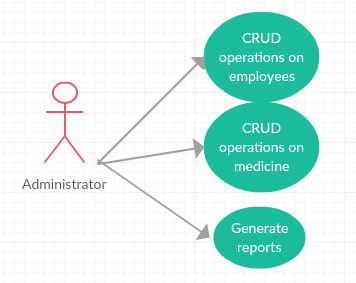
A success-failure scenario would be:

* Preconditions: the administrator tries to log to delete an employee.
* Postconditions:

Success: the administrator will log in the application, and will delete an employee from the farmacy

Failure: the administrator has an error while he tries to log in, because he made a mistake with the password.

Use case:



3. System Architectural Design

**3.1 Architectural Pattern Description**

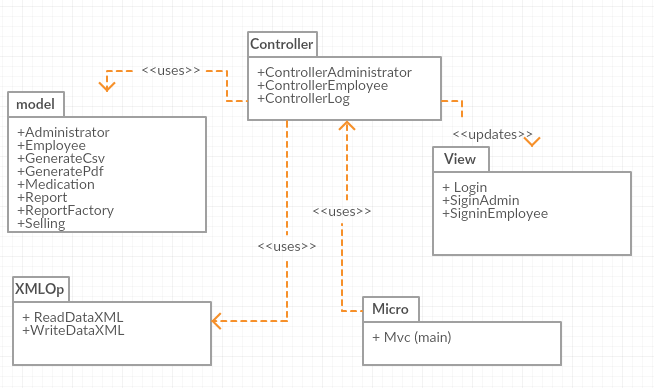
**3.1.1 MVC Design Pattern**

Stand for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

* **Model** - Model represents an object carrying data. It can also have logic to update controller if its data changes.
* **View** - View represents the visualization of the data that model contains.
* **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

I put in Model all the classes, like: Administrator, Employee, Medication, Selling. In View I have the interfaces for the log in, for the administrator, and for the regular user. And the most important part I have 3 controllers, for each of the interface, where I’m making all the operations.

**3.2 Diagrams**



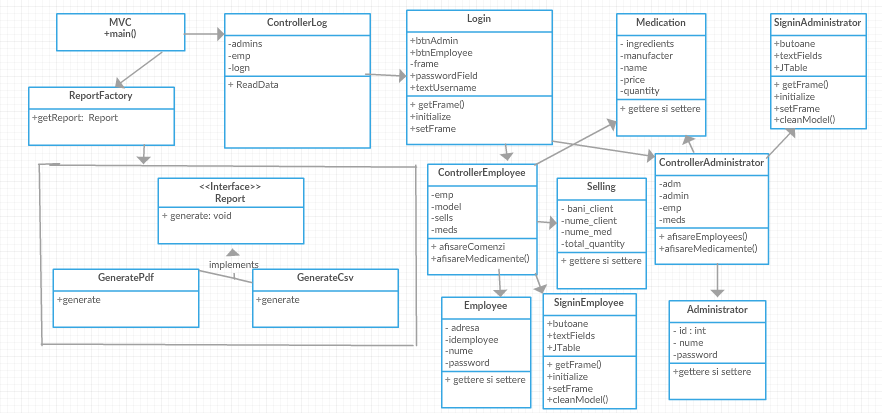
5. Class Design

**5.1 Design Patterns Description**

The design pattern that I used is Factory Pattern. Factory pattern is one of the most used design patterns in Java. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface. We have an interface Report, and two classes: GenerateCsv and GeneratePdf that will implement Report’s method: generate. We have another class ReportFactory, that will choose what class will create, depending on a name.

**5.2 UML Class Diagram**



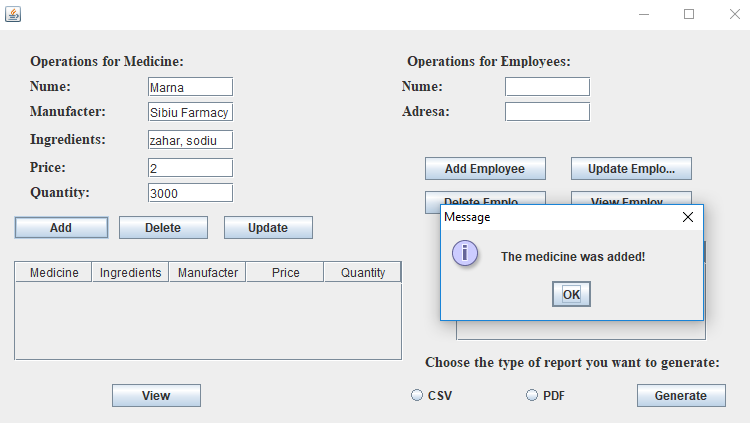
6. Data Model

I used XML files for keeping the data about Sells, Medications, and also Employees and Administrators. When we need to read or write from them, I created two separated classed, that will do that for us.

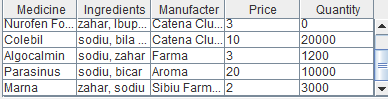
7. System Testing

Some scenarios are:

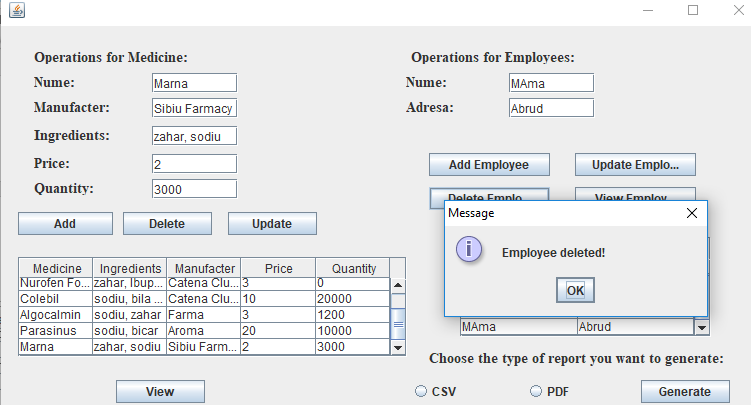
**7.1. Adding a medicine by the administrator:**



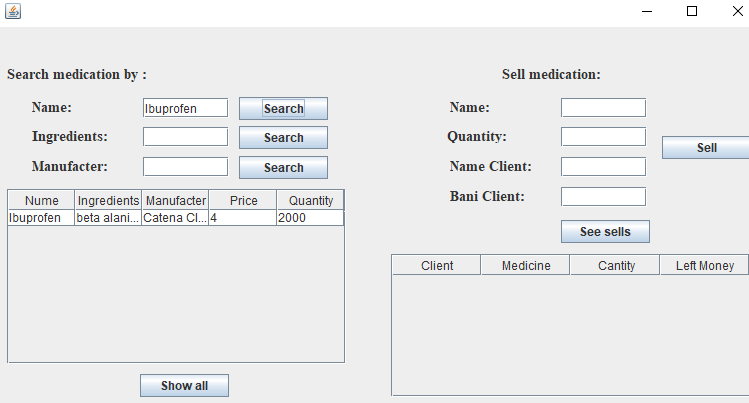
List with medicines after:



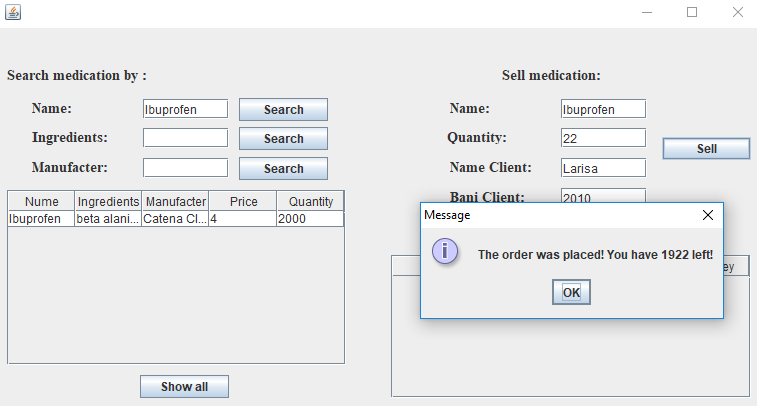
**7.2. Delete an employee:**



**7.3. Search medicine by name**



**7.4. Selling medicine to a client**



**7.5**. We have the reports saved in the project’s folder, for generating the medicine that is out of stock.

8. Bibliography

<https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm>

<https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>

<https://stackoverflow.com/questions/13063815/save-xml-file-with-xstream>

<http://x-stream.github.io/tutorial.html>

<https://examples.javacodegeeks.com/core-java/writeread-csv-files-in-java-example/>

<http://www.baeldung.com/java-pdf-creation>