Objective of the thesis is to design a chemical inventory management system for pharmaceutical industry and implement its data access layer (back end).

Perform analysis of the user requirement specifications as well as analysis of the regulatory authority (European commission) requirements for computerized systems, defined in "The Rules Governing Medicinal Products in the European Union, Annex 11" of EDURALEX Vol. 4. Completely design the system. Implement data access layer of the system. Perform testing of the implemented part of the system focused on storing and loading data. The system will be developed in Java together with MySQL database.

Czech Technical University in Prague
Faculty of Information Technology
Department of Software Engineering

Bachelor's thesis

Chemical inventory management system for pharmaceutical industry

Mgr. Juraj Škvarla

Supervisor: Mgr. Monika Součková

5th May 2016

Acknowledgements

I take this opportunity to express gratitude to my supervisor Mgr. Monika Součková for precious advices and time spent helping me elaborate the thesis. I also wish to express my sincere thanks to all of the faculty members for their help and support during my study, especially to Bc. Eva Dudíková. I would like to also acknowledge my schoolmate Bc. Jan Anděl and everyone who directly or indirectly, have contributed to the thesis. Finally, I am most grateful to my family for the continuous encouragement, support and attention.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 5th May 2016 .....

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Škvarla, Juraj. *Chemical inventory management system for pharmaceutical industry.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2016.

Abstrakt

Bakalářska práce se zabývá analýzou, návrhem a vývojem aplikace použitelné pro řízení skladového hospodářství chemikálií v malé až středně velké farmaceutické firmě v regulovaném prostředí správné výrobní praxe (SVP). Je kladen důraz jak na analýzu uživatelských požadavků, tak na splnění legislativních požadavků Evropské Unie na počítačové systémy v SVP. Navržený software poskytne pohodlné řesení problematiky řízení skladu jak vstupních surovin, tak meziproduktů a produktů.

Klíčová slova Správná výrobní praxe, Validace počítačových systémů, Regulované prostředí, Java, Systém řízení skladu chemikli, Kontrola kvality chemikli

Abstract

Bachelor thesis deals with analysis, design and development of the software usable as a stock management system of chemicals for small or mid-sized pharmaceutical companies, operating in regulated good manufacturing practice (GMP) environment. The emphasis is given to analysis of user requirements as well as to compliance with European Union legislative requirements on computerized systems used within GMP environment. Designed software

shall provide confortable solution for stock management of raw materials, intermediates as well as products.

Keywords Good Manufacturing Practice, Computerized System Validation, Regulated Environment, Java, Chemical inventory management system, Quality control of chemicals

Contents

List of Figures

Introduction

With the starting usage of computers in the industry in the 1980s, demands were also initiated to document the quality of such systems with regulatory requirements. The Blue Book of the FDA from 1983 is a notable example. As a response to the increased use of computerized systems and the increased complexity of these systems, European Union in 2010 issued a revised guideline on use of computerized systems in regulated environment of Good Manufacturing Practice for manufacturing of Medical Products for Human and Veterinary Use [1]. According to Annex 11 of this guideline, EU pharmaceutical companies are by law required to have all computerized systems validated.

For pharmaceutical companies this means:

- Limited selection of the software/developer
- Additional cost for software validation

This makes use of computerized systems by pharmaceutical companies somewhat complicated. Therefore it is to advantage that the developer is familiar with the regulatory requirements. Not only because the validation of computer systems is an interdisciplinary task, but because its tightly interconnected with the process of development of the computer software.

Aim

Obsolete chemical inventory management systems have a tendency to create needless tasks for the whole organization, from spending time searching for chemical containers to reordering stock that cant be found. After years of research, what may seem to be inconsequential losses of time and funds can add up to a serious amount of wasted resources. You may also be rightly concerned about staying in compliance if you arent sure where and in what state your regulated chemicals are at all times. This affects not only the qualitative environment of the lab but also your organizations long-term operational costs.

Aim of the bachelor thesis is to analyse both user requirements of a small pharmaceutical company and legal requirements of the European Union, according to that design simple Java stock-management desktop application that would be easy to integrate within already established GMP quality management system, implement and test its back-end.

Chapter 1

State-of-the-art

Most of the GMP-oriented applications that i have searched through are usually complex and very expensive software solutions e.g. instantGMP^{tm} or GMPPro. They are modular, however more or less designed to replace whole "paper based" quality management system of a the company. This is often very risky and financially demanding. Only very big pharmaceutical companies have enough resources for such big and risky investments. It is cheaper, safer and more convenient for small to mid-sized pharmaceutical company to have simple computerized system avaliable that solves only one problem and can be integrated into the working "paper-based" QMS and therefore replace only QMS tasks related to it.

1.1 Known existing applications and solutions

There are a lot of stand-alone off-the-shelf or open source stock management systems available. None of them that I have found satisfy all of the user and legal requirements discussed. Because they cannot be configured so that they would fulfill industry specific user and legal requirements, they would inevitably fail during the validation process.

1.2 Relationship of Computer System Validation to the Software Development Life Cycle

In order for software product to be usable within GMP environment, it must be validated. Computer System Validation is carried out through activities that occur throughout the entire Software Development Life Cycle (SDLC). The well known V Diagram Fig. 1.1, is widely used in the IT literature to emphasize the importance of testing and testing related activities at every step in the SDLC. It is also development strategy that is recommended by the GAMP 5 [2], well know industry guidance to achieve compliant computerized

systems. In the V-model, each stage of verification phase has a corresponding stage in the validation phase. I will start testing from the bottom of the V-diagram and test only corresponding, implemented part of the software by integration tests. Together with the analysis and design documentation I will generate enough data for the beginning of the project test and integration validation process phase.

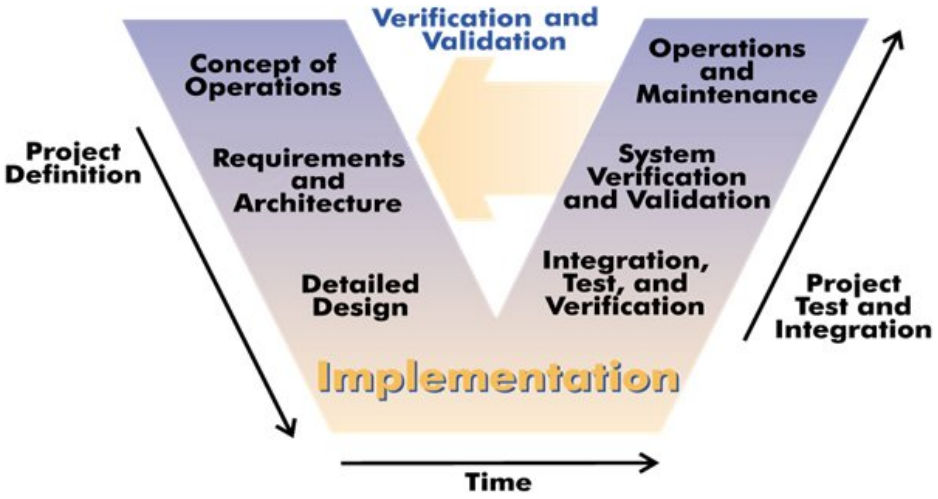

Figure 1.1: V-model of the SDLC, may be considered an extension of the waterfall model

Chapter 2

Analysis

2.1 Analysis of user requirements

Purpose for creating such system is to improve current solution for storing and managing chemicals within GMP QMS. It should replace currently used non-validated MS Access-based database. New system should be more robust, safer, easier to use, maintain and backup. There are three types of materials to be stored:

- Input materials
- Intermediates
- Final products

Each material has its unique material number, supplier, location to be stored in, units of measure and expiration interval (reanalysis interval). Each material can have several batches present in a warehouse, each with different batch number (also called LOT number). Each batch in a warehouse is in a particular state. There are three most important GMP-relevant states of the material:

- Approved for use - batch state which is approved for use.
- Quarantine- batch state of the material, while waiting for QC/QA to be analyzed.
- To be disposed - batch that has been disapproved for use or validity of the approval expired.

and three other states in which a material can be:

- Sampled - batch that has been sampled for analysis bat not yet analyzed.
- Disposed - batch that is no longer in warehouse.

- Expired - batch with expired approval.

2.1.1 Functional requirements

Summary of the basic functions that the software should perform:

1. Add, delete and edit records, i.e. user , material number, batch and transactions such as withdraw of the material.

2. Keep track of an amount of the chemicals in stock, i.e. it must be possible to see how much of the material for each batch there is left in a warehouse. It should be obvious which batch (if there are more then one batch) should be used first (according to a rule first-in first-out), to avoid expiration of the material.

3. Keep track of a location of the chemicals, i.e. it should be possible to locate each batch within the warehouse.

4. Keep track of expiry dates, i.e. it must be possible to see how much time is there left before expiration of the material

5. Classify chemicals according to their QC/QA state (quarantine, approved for use, disapproved for use, etc.), i.e. it must be possible to identify in which state a particular batch is present in a warehouse.

6. Staff shell be informed (by email), if there was a critical batch state change, i.e. if there is a new batch in quarantine, batch has expired, batch to be disposed and batch to be supplied

   Other requirements

7. Amount of chemicals shall be stored in different containers and states. Therefore different units of measure and unified precision shall be used (one decimal place). At least following units of measure shall be available: Litres, Millilitres, Kilograms, Grams, Milligrams, Pieces.

8. Every batch number has format of PPP-MMMMM-YY where PP is identification number of the batch (1-999), MMMMM is material number and YY are two last digits of the current year.

It is estimated that there will be approximately 100 to 300 different materials stored into the database. For each usually 1-5 different batches.

2.1.2 Non-functional Requirements

1. Security - Access to the database should be restricted for unauthorized persons.
2. Maintainability - Database maintenance should be cheap and easy.
3. Accessibility - System shall be accessible from local computers within private LAN network for several (1-10) users at the same time.
4. Operating systems - System must be able to run on multiple operating systems.
5. Interface - System should have graphical user interface.
6. Language - Available at least in English and Czech

2.1.3 User Characteristics

There will be four groups of users:

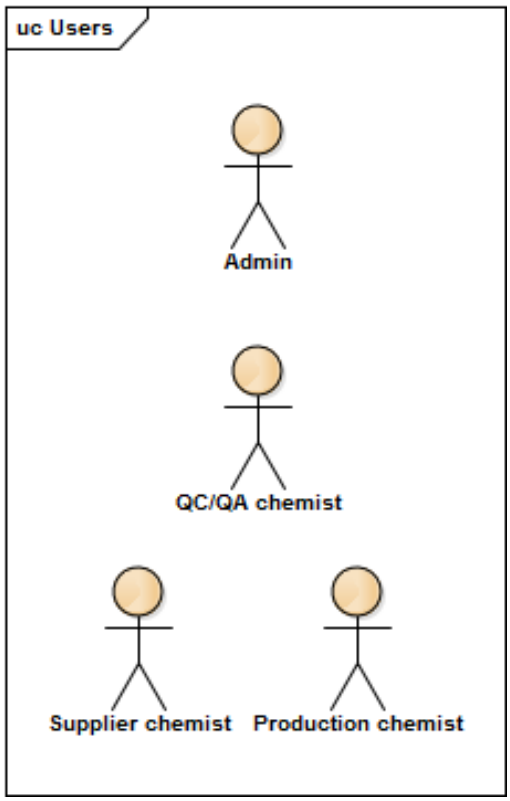

Figure 2.1: Different types of users working with the database

There are two different actions users can perform. They can create transactions or they can configure database. Configuration of the database is for Admin group only. Other three groups of users are allowed to perform particular type of transactions.

- Production chemists – chemists who can search through the database and withdraw certain amount of chemicals, user group 1.
- Supplier chemists - group of users who are responsible for supply and disposal of the chemicals. They can only add chemicals into the quarantine and dispose the chemicals, user group 2.
- QC/QA department - chemists who sample, approve and disapprove chemicals for use, user group 3.
- Admins – admins can configure the database, user group 4.

There are no special user specific requirements for the software. GUI should be as simple as possible.

2.1.4 Use Cases

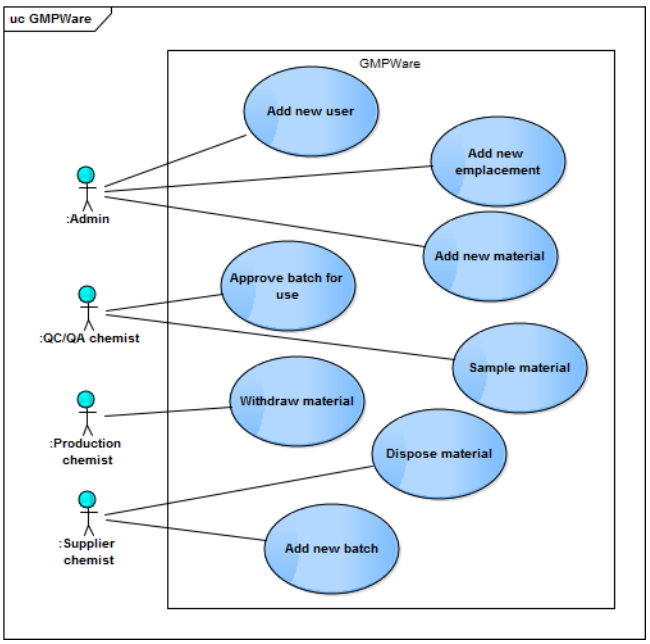

Figure 2.2: Top-level use cases

Sampling of the material Scenario - Batch has been received in storage area and put into the quarantine. It is necessary to sample and analyze the material so it can be approved for use, see Fig. 2.3.

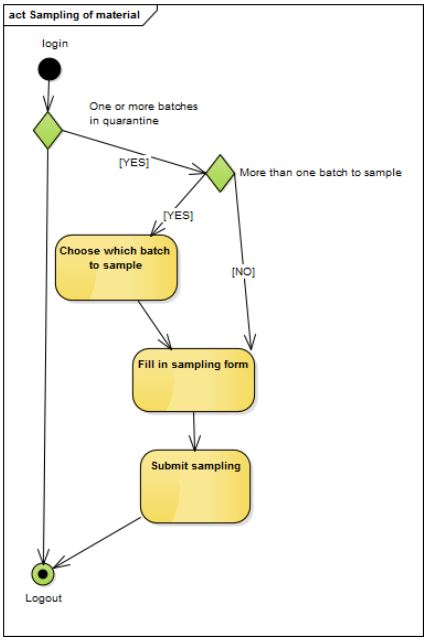

Figure 2.3: Sampling of the material use case

1. Once one or more batches are in quarantine, a person responsible for sampling and/or analysis is notified by email about the fact.
2. If there are more batches to sample, person will choose which batch to sample and fill in sampling form.
3. After control of the sampling record can be submitted.

Withdraw material Scenario - It is necessary to withdraw material that will be used in production:

1. Production chemist will fill in withdraw requirement form (Material number and amount required)
2. If there is approved batch of material available with sufficient quantity, its location and withdraw form is shown.
3. If there is not enough material or there is no batch. Email is send to responsible users.

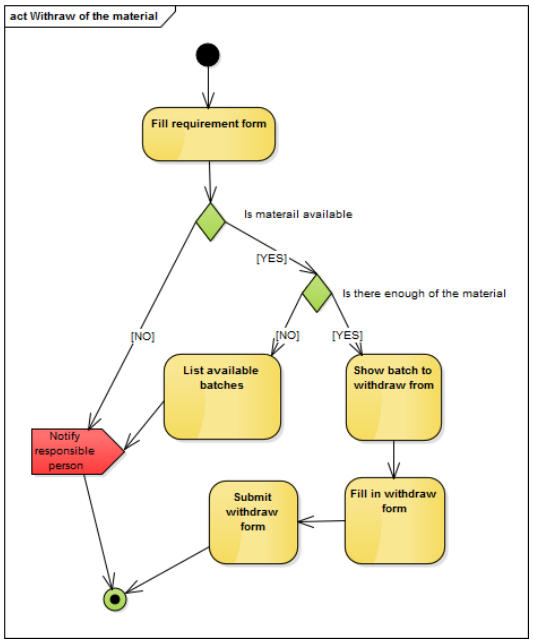

Figure 2.4: Withdraw of the material use case

2.2 State diagrams

Transition between states of the material shall follow schema in Fig. 2.5. Each transition between batch states must be documented and traceable.

2.3 Access of legal requirements influencing design of the system

Some of the legal requirements are already reflected into user requirements. This chapter focuses on those that are not. Each legal requirement is assessed and design aspect proposed.

Annex 11 is divided into 3 parts:

1. General Requirements - these requirements apply purely to the approach of the customer to the computerized system management, supplier selection, etc.

2. Project Phase - these requirements apply to the project phase of the developing process. It focuses on precise URS definition and other quality-relevant practices for project phase.

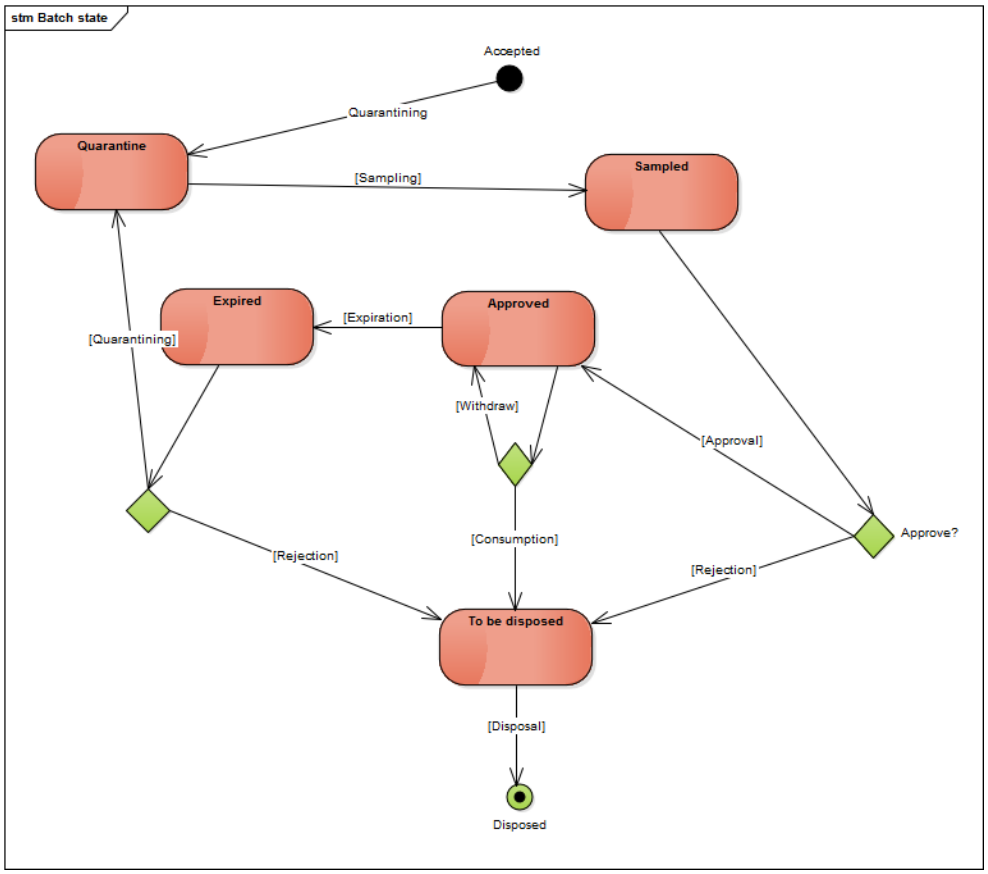

Figure 2.5: States in which a batch can be. Each transition between states corresponds to actions that different users can perform.

3. Operational phase - these are general requirements on operation of the computerized system. From these I have selected requirements which are relevant from database-type computerized system and could be fulfilled by correct design of the system. Each desing feature and its function is briefly described.

According to chapter 5: *"Computerised systems exchanging data electronically with other systems should include appropriate built-in checks for the correct and secure entry and processing of data, in order to minimize the risks."* Manual entry of the data in specific format is required for amount of material (correct units and precision), etc. see section 2.1. Therefore for entry of these data there shall be special "input data format validation" methods.

According to chapter 6: *"For critical data entered manually, there should be an additional check on the accuracy of the data. This check may be done*

by a second operator or by validated electronic means. The criticality and the potential consequences of erroneous or incorrectly entered data to a system should be covered by risk management." Similarly to the previous risk of the format missentry, where critical data will be enterd manually, confirmation notice should be available when user is asked to confirm correctness of the entered data. This can be implemented for example on presentation layer (accuracy checks of the data).

According to chapter 8: *"It should be possible to obtain clear printed copies of electronically stored data."* Therefore there shall be used a library for pdf export e.g. iText PDF Api that would allow to provide such functionality.

According to chapter 9: *"Consideration should be given, based on a risk assessment, to building into the system the creation of a record of all GMP-relevant changes and deletions (a system generated "audit trail"). For change or deletion of GMP-relevant data the reason should be documented. Audit trails need to be available and convertible to a generally intelligible form and regularly reviewed."* In order to comply there will be created a class "Trail-Logger" responsible for logging all GMP-relevant changes, e.g. communication with the database. Following changes are considered to be "GMP-relevant":

1. Deletion of the records.
2. Upload of the records.
3. Creation of new records.
4. Sing-in and sign-out of the user.

According to chapter 12: *"Physical and/or logical controls should be in place to restrict access to computerized system to authorized persons. Suitable methods of preventing unauthorized entry to the system may include the use of keys, pass cards, personal codes with passwords, biometrics, restricted access to computer equipment and data storage areas."* User access control will be designed so that only authorized operators can perform critical operations. Only hash of the passwords shall be stored in database.

According to chapter 13: *"All incidents, not only system failures and data errors, should be reported and assessed. The root cause of a critical incident should be identified and should form the basis of corrective and preventive actions."* Exception thrown will be logged into the database from which user can assess root cause and perform corrective and preventive actions.

Chapters 7, 10, 11, 14 and 15 does not influence design of the database ether because the software will not provide mentioned functionality or because it is not possible to fulfill such requirement purely by correct design of the application.

As can be seen there are at least 6 legal requirements that shall be taken into the account during the design stage.

Chapter 3

Design

3.1 Selecting the right tools

The following section lists some tools and libraries used during the development process.

- MySQL database - For purposes of the small to mid-sized company and the fact that there will be only a few users logged in at the time, MySQL performance will be sufficient. Also MySQL as an open-source solution will comply with the requirement on low purchase and maintenance cost of the database [3].
- JUnit - Testing framework JUnit [4] is widely accepted as a standard for unit testing in Java. Many of the available testing products on the market are either based on or extend JUnit. Also, many IDEs currently have built-in support for JUnit. The majority of unit tests written in this book are JUnit tests.
- JDBC - It's an established standard API for database access[3].
- Swing - Library provides a flexible graphical user interface tools from which to develop user interface. It is part of Oracle's Java Foundation Classes[5].
- Hibernate - For application with complex interrelationships, Hibernate can take away a huge amount of coding effort and will result in an application that performs better than the alternative handcrafted JDBC[6].
- iText - This library offers powerful and flexible tools for PDF creation, editing and inspection[7].
- javamail - The JavaMail API provides a protocol-independent framework to build mail and messaging applications[8].

3.2 Deployment and design pattern

Based on the analysis of the user and legal requirements. Application deployment as shown in Fig. 3.1 will meet user requirement on accessibility and will not restrain fulfillment of any legal requirements. It shall be based on 2-tier architecture style and will use Model-View-Controller (MVC) design pattern.

Basic principle of the application with the centralized database would be that user would sign in into the application and application would then sign in into the database within local area network (LAN). Information necessary for application to log into the database (IP address of the database server and port) will be provided by the config.properties file.

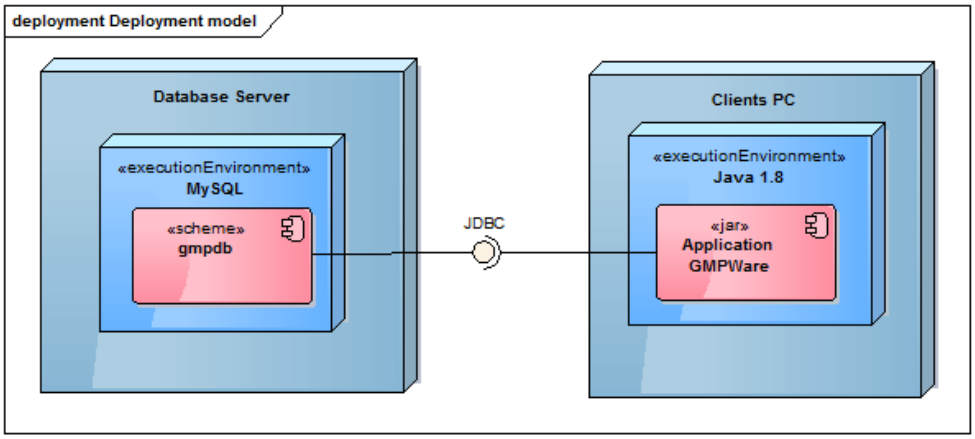

Figure 3.1: Two-tier deployment of the application

3.3 Components

Source code is structured into 3 main packages (layers).

1. Presentation package
2. Business package
3. Data package

Each package with different responsibility, as depicted in Fig. 3.2.

3.4 Presentation (PL) layer

Presentation layer contains view classes package and controller classes package.

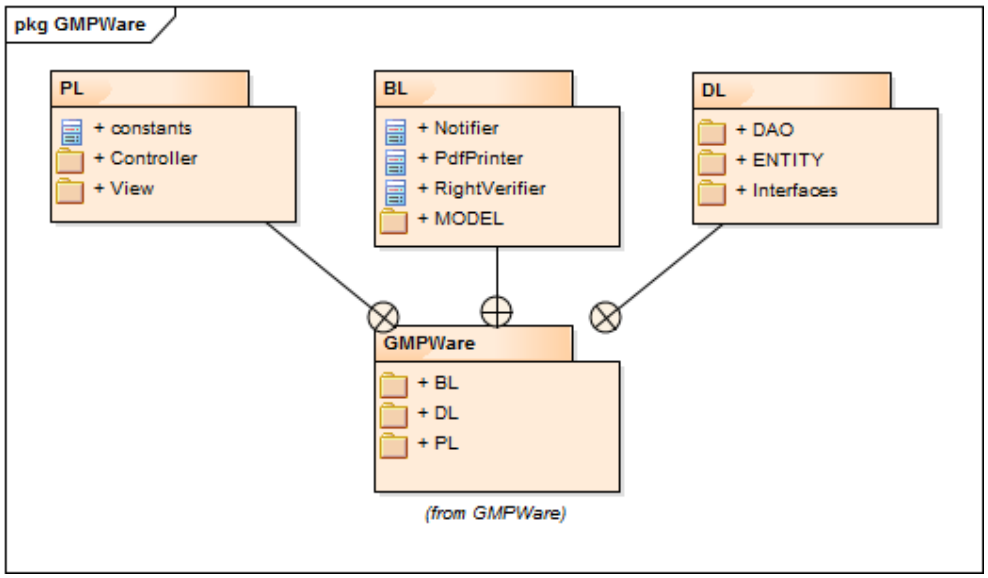

Figure 3.2: Structure of the project source files and packages

Because it is required, that the application is in two different languages, Non-functional requirement No. 6, in order to allow for easy GUI translation, there will be a class of constants shared among all view and controller classes that contains GUI labels and messages.

3.4.1 Views

Package of views will contain classes that are necessary in order to interact with the user.

There will be three main view classes:

1. LoginView - will be resposible for getting login information from the user such as username and password

2. HomeView - will be main view from which users can initiate all batch transactions such as putting new or expired batch into the quarantine, sample a material, etc. Home view will also show actions that are necessary to perform, in order to keep database updated and all batches approved for use. Example of how homeView could look like, see. Fig. 3.5

3. ConfigureView - will be responsible for getting information from the user required in order to add new, update and delete configuration data into the database

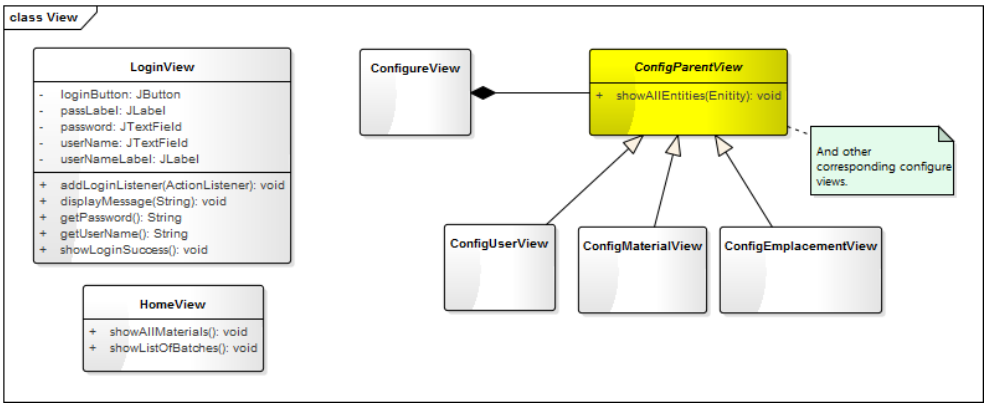

Figure 3.3: Classes of the view package with examples of some of the required methods and member variables.

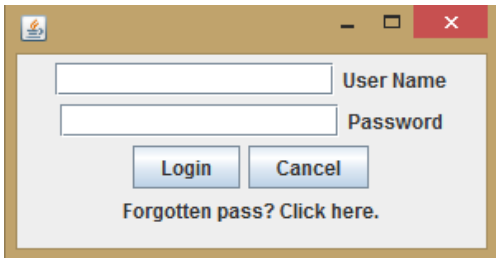

Figure 3.4: Example of the proposed LoginView.

3.4.2 Controllers

There will be following controller classes, see. Fig. 3.6.

1. LoginController
2. HomeController
3. ConfigureController

LoginController is responsible for all actions necessary in order to log user in. It will pick user based on username from the database and verify its password hash. There could be for example java.security package and its Secure Hash Algorithm (SHA-256) used for that. If the hashes matches it will create HomeController class and inject the user object into it. [1]. LoginController will also be responsible for reset of the forgotten user password.

[1] Because application will have the same identity during the communication with database for each user, therefore information about the user that is logged in must be available

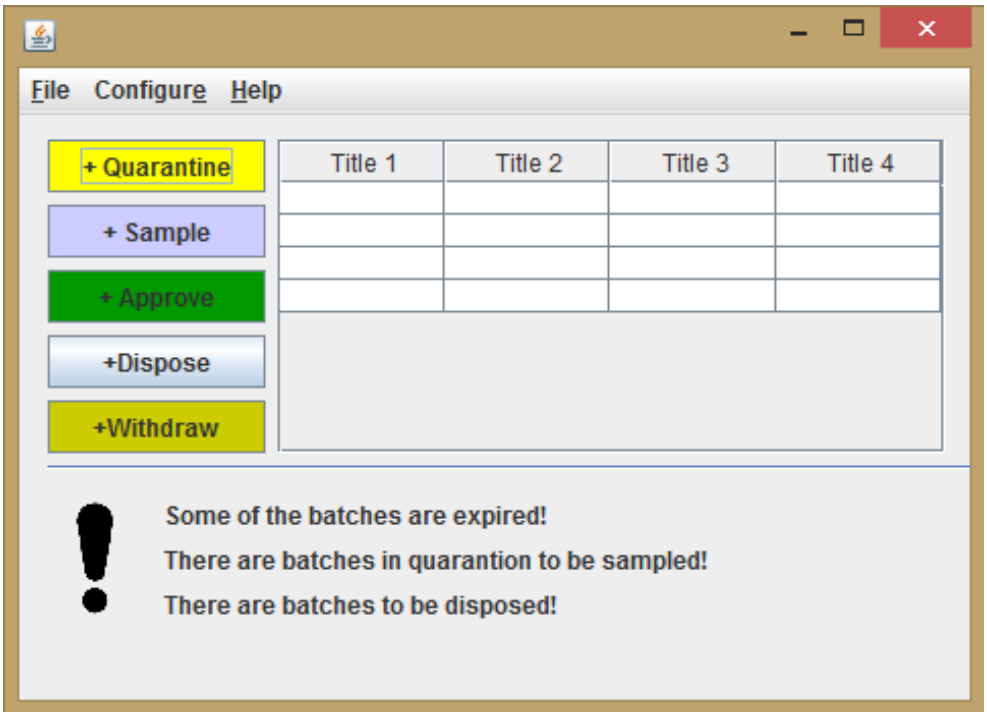

Figure 3.5: Example of the proposed HomeView.

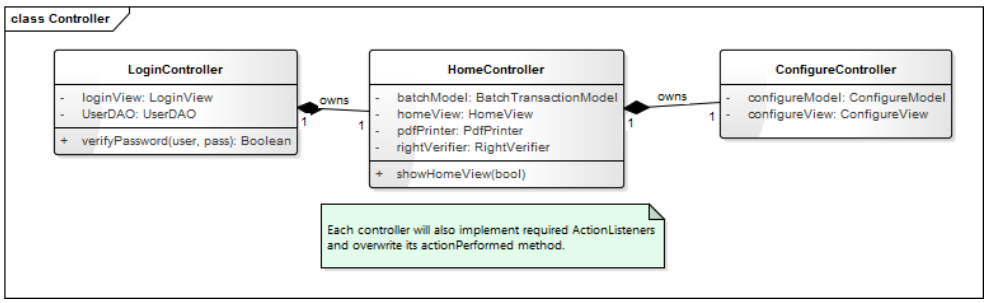

Figure 3.6: Class diagram of the controller classes with examples of main member variables and some required methods.

HomeController now has information about user who is logged in and takes responsibility and controls actions performed by the user.

If the user wants to configure the database, then the HomeController creates ConfigureController and passes responsibility for configuration actions to it. Configure controller then controls creating and displaying views for adding, deleting and updating data for configuration, see Sec. 3.6.

3.5 Business (BL) layer

Classes in BL package are responsible for required (business related) actions.

3.5.1 Model package

There will be two main model classes, see Fig. 3.7.

1. BatchTransactionsModel - will communicate with DL through DAO interfaces. Will be responsible for all actions related to batch transactions (Use cases of the QC/QA, Production and Supplier user groups). It will have methods like verifyRetestInterval to check if there is batch that should expire or velidateBatchNumber that would validate that batch number format is correct, to comply with the functional requirement 8 and legal requirement from chapter 5 of the Annex 11 2.3

2. ConfigureModel - will also use DAO interfaces and will be responsible for actions related to database configuration (Use cases of Admin user group).

3.5.2 Other BL classes

In order to comply with the legal requirements from chapter 2.3,. for generating portable document format files from lists of data, for this there is class PdfPrinter class which utilizes iText library.

Emailing notifications, function req. 6, when a critical change to the batch state occurs are handled to EmailNotifier class, that utilizes javamail library and will send email to correct user who should react to it, etc.

RightVverifier class would be used by the HomeController and would be responsible for verifying if the user has enough rights to perform particular action (based on the groups he is member of).

within the code (could also be persisted after login and loaded every time info about the user that is logged in will be required, but this is in my opinion not a better solution that pass user if required)

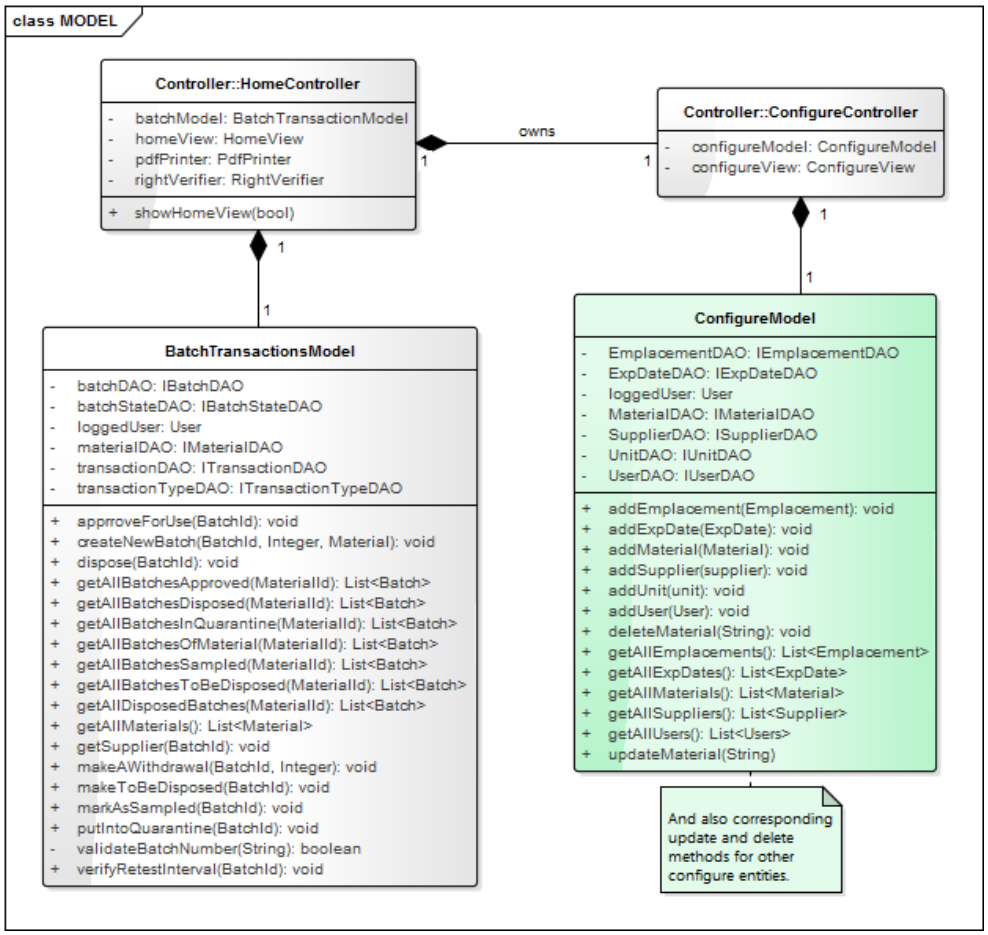

Figure 3.7: Class diagram of the model classes with examples of main member variables and methods.

3.6 Persistent (DL) layer

For loading, storing, updating and deleting persistent objects will be responsible persistence layer, which uses data access objects (DAOs) in order to comply with the single responsibility design principle [9]. Each DAO object implements public interface with required methods. Since User group, batch states and transaction types cannot be changed or modified, there are only getters to the data, Fig. 3.10.

Classes that are to be persisted are in ENTITY package. These objects are mapped to the relational database by the Hibernate ORM framework. HibernateUtil class is responsible for creation of SessionFactory for DAO objects. Singleton design pattern will be applied to the HibernateUtil class, so that there is only one SessionFactory for entire application.

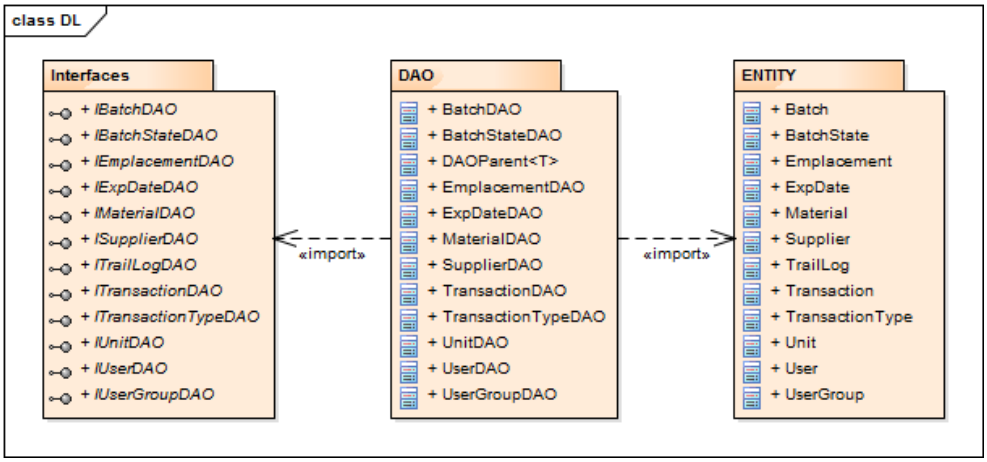

Figure 3.8: Package diagram of the DL layer

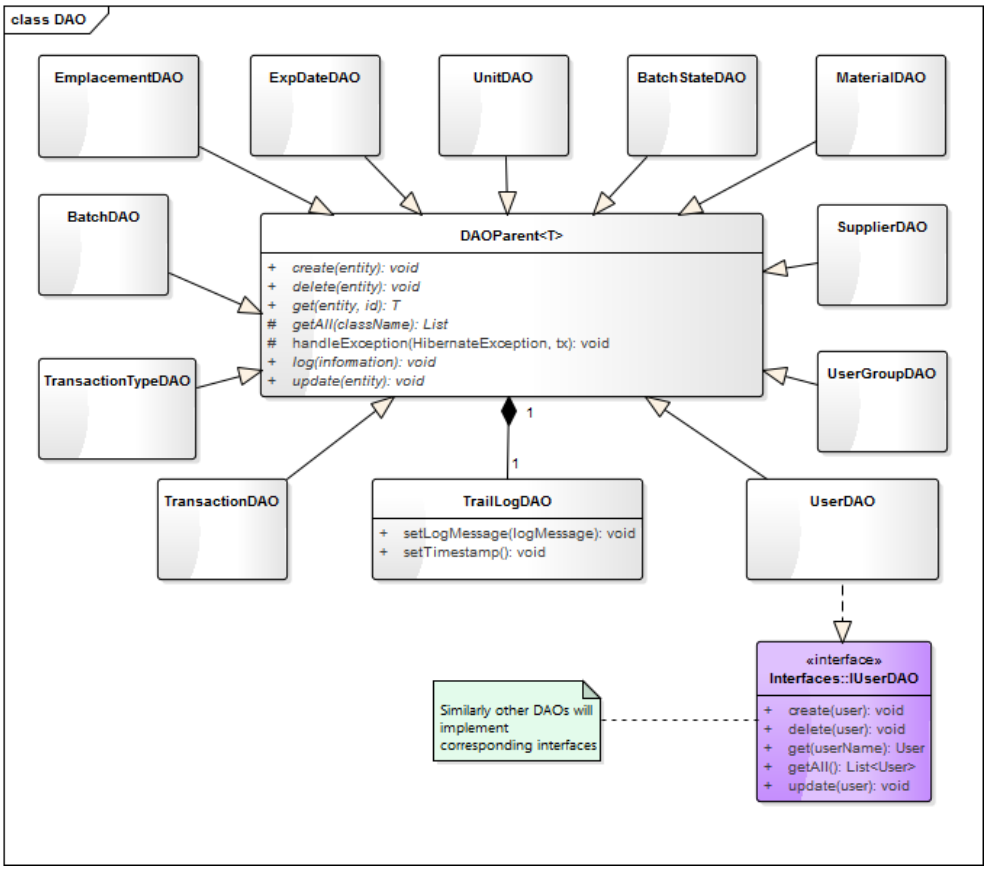

Figure 3.9: Class diagram of the persistent entity classes

3.6.1 Types of data

Persistent layer will store three types of data:

1. Data used for application
2. Data used for configuration
3. Runtime data records

The main difference between the three types is, who and when creates and/or modifies the data.

3.6.1.1 Data used for application

Without these data application cannot run and also should not be changed by anyone.[2] Information about the batch states, transaction types and user groups was generated during analysis and is inserted into the database before application connects to it, see Fig. 2.5 and Fig. 2.1. They are stored in tables:

a. transaction_type

b. user_group

c. batch_state

3.6.1.2 Data used for configuration

This information is a configuration of the database. Users from admin group can edit, add or remove records from the tables. Database should be configured in this way so that is reflects users, materials and its suppliers, etc. They are stored in tables:

a. emplacement

b. user

c. unit

d. exp_date

e. supplier

f. material

g. supplier_has_materials

h. user_has_usergroups

[2]These data could be constants within the application code and attribute to batch, transaction and user entities. I have decided to put them into the relational database in order to avoid data redundancy.

3.6.1.3 Runtime data records

Are generated by the application to document changes in batch states, amount of material of each batch available, etc. These data can be created, edited or deleted by other three user groups at any time. They are stored in tables:

a. transaction

b. batch

c. traillog

d. exceptionlog

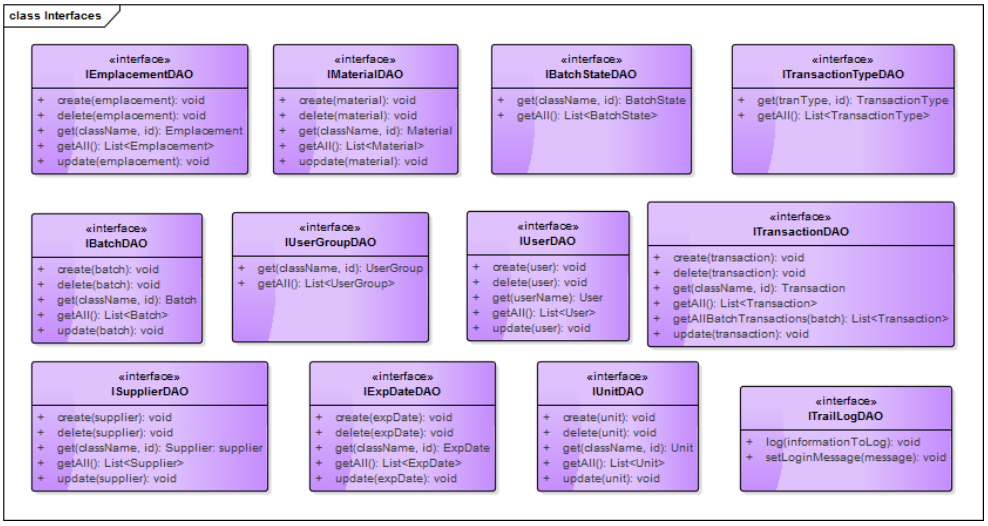

Figure 3.10: Interfaces of the DL layer

3.6.2 Database EER model

One main database schema will persist most of the entity objects, Fig.3.11, but there are two more standalone tables (databases) that are created.

1. Audit Trail database

2. Exception database

Audit Trail database is a table for TrailLog class which manages logging creation, deletion and update of persisted objects, logging in, logging out of

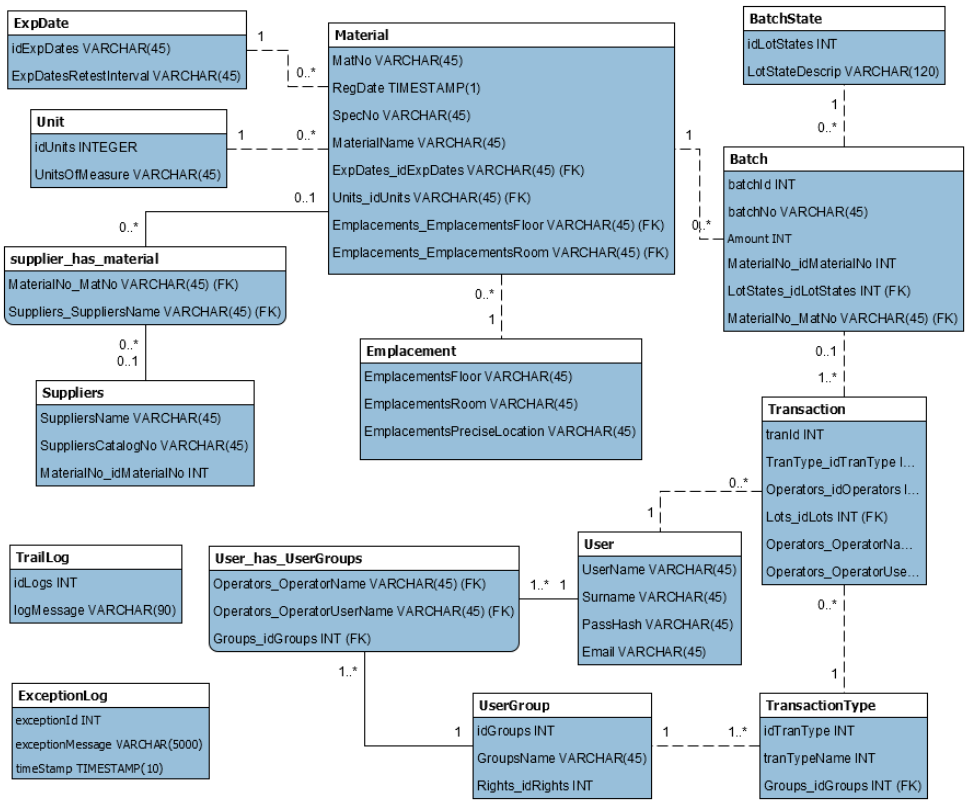

Figure 3.11: Relational models of the database: Main relational model, TraiLog and ExceptionLog tables

the user and also unsuccessful login attempts[3]. It is to implement legal requirement from chapter 9, see 2.3. For logging creation, deletion and update of persisted objects would be responsible DL layer. In order for DAO objects to be able to log such data, information about the user that is logged in will be passed from the controller to model objects which will inject this information into the DAOs through constructor, so that DAOs have all information necessary to log theirs transactions.

Exceptions database table is for logging exceptions that were thrown. It is to cope with the legal requirement from chapter 13, see 2.3. It shall be implemented by the class "ExceptionLog" and "ExceptionLogDAO". Hibernate exceptions that are thrown by the Hibernate will be logged automatically by

[3]There is also another possibility how to implement audit trail functionality and its to store information about the database transactions as a column to each entity record table. This would be easy to implement, however not easy to evaluate and view for the user, since order in which the transaction took place would be lost.

the HandleException method of the DAO objects. For logging other exceptions are responsible methods that caught the exception.

Chapter 4

Realisation

4.1 Naming Conventions

- Methods should be verbs, in mixed case with the first lowercase letter. The first letter of each internal word capitalized.
- The names of variables declared class constants should be all uppercase with words separated by underscores.
- Class names should be nouns. In mixed case with the first letter of each internal word capitalized.
- Variables are in mixed case with a lowercase first letter.

4.2 Implementation of Back-end

I have implemented:

- DL package and all of of the JPA Entity classes
- Package with DAO classes
- DAO interfaces

I have also created import.sql script that imports all data necessary for application and some of the data necessary for DL integration tests, see. 4.3.

In order to configure Hibernate, there is the hibernate.cfg.xml file. For mapping entities I have been using Hibernate Annotations. Entity classes were written according to the requirements in Java Persistence API specification [10]. Apart from the getters and setters of the entity member variables, for each entity there is a custom constructor and overrided equals and hashCode methods.

4.3 Testing

When trying to deliver a good software product, especially when there is a need for validation of the product for pharmaceutical industry, testing of the code is very important part not only during development stage[11]. Without sufficient amount of well designed and documented tests (unit test, integration tests, system tests and user acceptance tests) software should not be accepted by the customer, because he cannot prove its correct functioning.

In order to test correct mapping of the entities to the MySQL relational database I have written basic integration tests for all entities and verified that persistent objects can be stored, deleted and edited from the testing database as it is expected. The tests were written so that after the test execution, the data initially stored into the database are the same as before the test. See list of the tests and their results after last implementation iteration, Fig. 4.1.

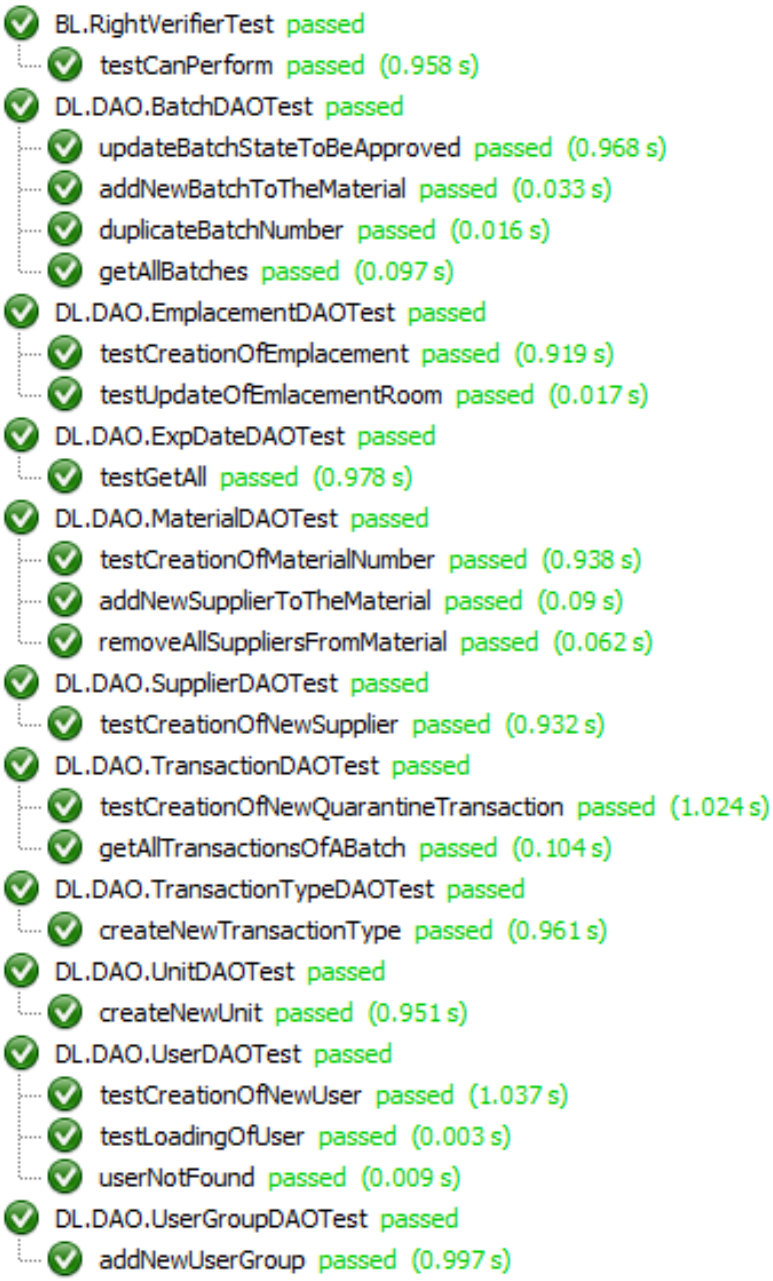

Figure 4.1: List of the integration tests and their results

Conclusion and discussion

Proposed industry specific chemical management solution was designed to meet the specific requirements and responsibilities of users, that will result in up-to-date and ready-to-use stock of chemicals for manufacturers working within regulated GMP environment. Further programming is necessary in order to create fully functional application. However, implemented DL package can already provide basic persistence functionality of the designed application.

I have analyzed both user requirements and legal requirements for computerized systems in regulated environments. From legal requirements i have chosen those that apply to the particular type of the computerized system (stock management system) and can be implemented by the correct design. By using a few libraries I have designed very simple Java application with minimal required functionality. I have implemented and also tested persistent layer of the application.

Assessment of compliance with the user and legal requirements

From 15 functional and non-functional user requirements I believe that application more or less comply with all of them. There should be performed software validation in order to verify that its true.

From 6 important legal requirements I also think that application comply with all of them, but I think there is a lot more room for improvements. For example legal requirement No. 9, see. 2.3, states: "Audit trails need to be available and convertible to a generally intelligible form and regularly reviewed." Audit trail as is designed would mix logging of all DAO transactions and configuration actions would be mixed with batch transaction records, which may be hard for reviewing. Therefore this table could be separated or there could be method to convert it to more "generally intelligible form".

Known bottlenecks

Because the application shares database and multiple clients can connect from multiple computers to it and manipulate the data simultaneously, there are also limitations on how can application be used. There should be only one admin user configuring the database at the same time, since if for example one admin would load user object and changed its username, but meanwhile another admin would try to change group of that particular user, update of the user-usergroup relation by the second admin would be unsuccessfull. There would be an exception thrown, since user with that username would no longer exist. This cannot happen for other user groups during their tasks, since they are allowed only to create new batch or add batch transactions. In a GMP environment, this should not be a problem anyway, since each change to the application configuration should be only made by the controlled manner.

Expiry date as an interval between reanalysis (expiry interval) will be calculated by the application. This could be a problem if someone want to track expiry based not on reanalysis interval but rather based on fixed date provided by the supplier (e.g. when material cannot be analyzed and responsibility for expiry lies within supplier). And could also make problems during leap years.

Expiry of approved batches, as designed, is verified by the verifyRetestInterval method of the BatchTransactionsModel class. Method can be called automatically, for example when user logs in, but cannot be called if there would be no user logged in regularly (i.e. application would not run at all).

Future functional improvements

In the future program could be extended, so that it would calculate cost of keeping materials approved (based on batch analysis cost, reanalysis frequency, etc.) therefore optimize amount of chemicals supplied (avoid oversupply and minimize expiry of the batches).

System could be also extended to provide Bill-of-materials functionality. If there was an information about the product for which the batch has been withdrawn, it would be very easy to produce BOM (list and amount of materials required in order to produce a particular amount of product). In practice this information is usually laborously piked up from the manufacturing instructions an is only estimated.

Also because each batch must be labeled and the database will contain information about the expiry dates, batch numbers, approval dates, etc. It would be possible to use it to produce labels of chemicals (quarantine, approved and also to be disposed).

Bibliography

[1] European Commusion Health and Consumers Directore-General. *EudraLex, The Rules Governing Medicinal Products in the European Union Volume 4, Good Manufacturing Practice, Medicinal Products for Human and Veterinary Use, Annex 11 - Computerised Systems*. 2010. Available from: `http://ec.europa.eu/health/files/eudralex/vol-4/annex11_01-2011_en.pdf`

[2] International Society for Pharmaceutical Engineering. *GAMP®5: A Risk-Based Approach to Compliant GxP Computerized Systems*. 2007. Available from: `http://www.ispe.org/gamp-5`

[3] MySQL [online]. May 2016. Available from: `https://www.mysql.com/`

[4] JUnit [online]. November 2015. Available from: `http://junit.org/`

[5] The Free Encyclopedia. Wikimedia Foundation Inc. Updated 14:02 10:55 CET. [online]. April 2016. Available from: `https://en.wikipedia.org/wiki/Swing_(Java)`

[6] Hibernate [online]. May 2016. Available from: `http://hibernate.org/`

[7] ITextPDF [online]. May 2016. Available from: `http://itextpdf.com/`

[8] JavaMail [online]. March 2016. Available from: `https://java.net/projects/javamail/pages/Home`

[9] Martin, R. C. *Agile Software Development, Principles, Patterns, and Practices*. Upper Saddle River, NJ, USA: Prentice Hall, 2003, ISBN 978-0135974445.

[10] Sun Microsystems. *JSR 220: Enterprise JavaBeans, Java Persistence API Specification*. 2006. Available from: `https://jcp.org/aboutJava/communityprocess/final/jsr220/index.html`

[11] Freeman, S. *Growing Object-Oriented Software, Guided by Tests.* Addison-Wesley Professional, 2009.

Appendix A

Acronyms and terminolgy

API Application Programming Interface

BOM Bill Of Materials

CSV Computerized System Validation

DAO Data Access Object

EU European Union

GUI Graphical User Interface

GMP Good Manufacturing Practice

IDE Integrated Development Environment

IP Internet Protocol

JDBC Java Database Connectivity

JPA Java Persistance API

QC/QA Quality Control/Quality Assurance

QMS Quality Management System

LAN Local Area Network

ORM Object Relational Mapping

PDF Portable Document Format

SDLC Software Development Life Cycle

SQL Structured Query Language

V&V Validation and Verification

XML Extensible Markup Language

Validating computerized system means showing in documented form that with great probability, the system will function in a reproducible manner as the specification states it should function.

Good manufacturing practices are the practices required in order to conform to the guidelines recommended by agencies that control authorization and licensing for manufacture and sale of food, drug products, and active pharmaceutical products.

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `GMPWare` ..... the main directory of source codes
  - `src` ..... implementation source files
    - `PL` ..... examples of PL package classes
    - `BL` ..... examples of BL package classes
    - `DL` ..... implemented DL package
  - `test` ..... implementation sources integration tests
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format
  - `thesis.ps` ..... the thesis text in PS format

Appendix C

GMPWare User Guide

User Guide was written for users of the application so that they understands how application works and how to use it. For information on how to set up and configure the system see Admin Guide.

C.1 Basic functions description

GMPWare is very simple Java based software that was designed to provide support for activities related to management of materials (only chemicals) within Good Manufacturing Practice. GMPWare is designed to follow and document actions necessary to analyze starting materials and (dis)approve them for use as well as actions necessary to avoid expiry of materials. See "quality controlled" material flow, C.1.

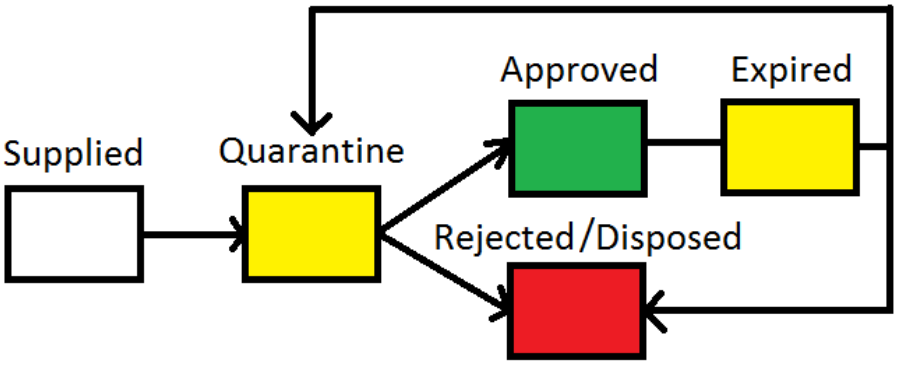

Figure C.1: Basic material quality control principle. Until it was proven that material has sufficient quality, it cannot be used for production. Rejected - not sufficient quality, Approved - sufficient quality

C.2 User groups, rights and responsibilities

For each GMP-relevant group of actions there is an user group. There are four user groups, each user group with different responsibility:

- Production – chemists who can search through the database and withdraw certain amount of chemicals, user group 1.
- Supplier - group of users who are responsible for supply and disposal of the chemicals. They can only add chemicals into the quarantine and dispose the chemicals, user group 2.
- QC/QA - chemists who sample, approve and disapprove chemicals for use, user group 3.
- Admins – admins can configure the database, user group 4.

After sign in each user group can do different actions, see. C.2. Admins can configure database by clicking on Configure drop-down menu. Users from supplier group can put new material into the quarantine by clicking +Quarantine button or dispose batches that are in "to be disposed" state by clicking +Dispose button. Users from QC/QA group can sample batches that are in quarantine by clicking on +Sample button and they also can approve already sampled batches for use by clicking on +Approve button. Users from production group can withraw material by clicking on +Withdraw button.

If there is an action necessary, see C.3

C.3 Batch states, expiry and material compliance control

Every time any user is signed in into the application, application shall check if there is any batch in "quarantine", "expired" or "to be disposed" state (non-compliant state). If there is such a batch, application will send email to every user from user group that is responsible for required consecutive action. This way users are informed if there is an action required in order to keep every batch in "approved for use", i.e. they are ready to be used for production, "sampled", i.e. ready to be analyzed or "disposed" state, i.e. they are already disposed.

C.4 Home window of the GUI

After login Graphic User Intefrace consists of one main "Home Window", see C.2, from which all basic actions with the batches can be done using transaction buttons (Quarantine, Sample, Approve, Dispose, Withdraw). There

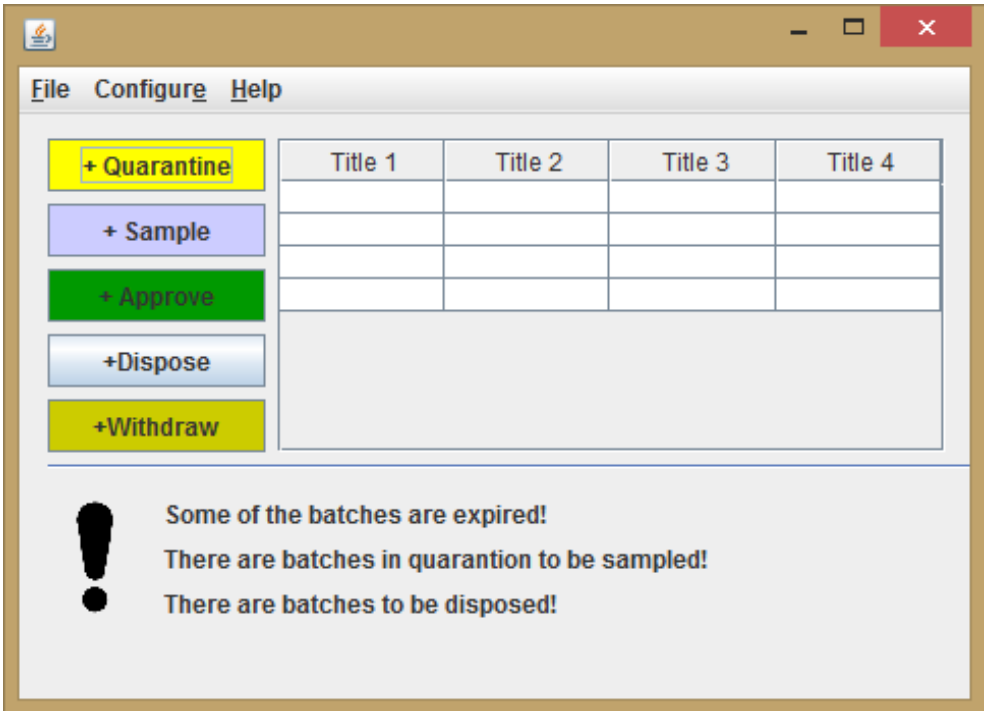

Figure C.2: Home Window

is also table of the materials that are available you can choose with. Home window also constains Information section, so that users are informed about all necessary actions to be done.

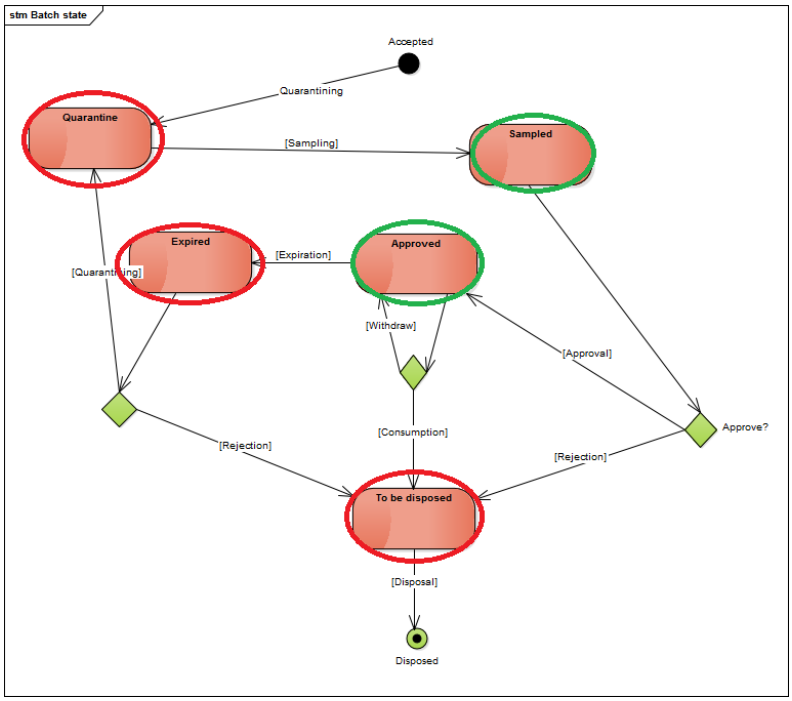

Figure C.3: States at which a batch can be. Red circle - batch states at which an action is necessary, Green circle - batch states that are ready to be used or no action is necessary

Appendix D

GMPWare Administrator guide

Administrator Guide was written for users that will be installing and setting up the GMPWare. It contains only basic information that is necessary to make the system work and maintain. For further information see development documentation. It does not contain information on how to perform acceptance tests, which are required to test correct setup of the application.

D.1 Design principle

GMPWare consists of a centralized MySQL database and "client" applications configured within LAN network. Client applications connect to the database and communicate with it through embedded JDBC interface.

D.2 Setup of the database

There must be MySQL (recommentded version 5.7.12) installed (available from http://dev.mysql.com/downloads/installer/) in server computer within LAN network. In order to create and configure the database, before any client app can connect to it, you should run following scripts:

- create.sql (will create tables and insert foreign keys)
- config.sql (will insert basic data used by the client app)

Also you must make sure that you have created secret root user and set password that will be used by the client application to connect to the database. Username of the root user must be "gmpwareroot". Its password must be "gmpware123".

D.3 Setup of the client apps

In order to run clinet app, it is necessary to copy executable file GMPWare.java and configuration file config.properties files into the computer with Java Runtime Environment (recommended version 1.7) installed. Configuration file contains information that is necessary, so that the clinet app can connect to the database. You must set:

- "IPdatabase" parameter (IP of the server that has MySQL database installed and configured)
- "port" parameter (port at which MySQL database listens)

Example of the config file content(for database at localhost):

#Client connects to database
IPdatabase=127.0.0.1
port=3306

D.4 Configuration of the system

Database should be filled with the essential information about materials (materials that are to be managed) and user data (who will manage the materials) of the company before it can be used. In order to configure the system fully, there should be list of users, list of materials with its suppliers, units of measure, emplacements and expiry intervals available. Configuration of the system can be done through client application, see User Guide or manually by inserting required data into the database by using SQL (not recommended).

Other configuration requirements:
There must be at least one person configured within each user group (see User Guide) so that application can run properly.

D.5 Maintenance

In order to maintain the system you should regularly:

- Backup database data manually by storing an import script (if there is no other way)
- Edit configuration data such, as users, material managed etc., so that it reflects current state
- Review and verify exceptions that were thrown and/or report other known issues to developer