Find My Patient

Software documentation



|  |  |  |
| --- | --- | --- |
|  | By: | Shazil Khan  Gerard Klomphaar |
|  | Date: | 24-01-2016 |

Contents

[Subject description 3](#_Toc442453663)

[Subject analysis 4](#_Toc442453664)

[User Requirements 4](#_Toc442453665)

[Application Feasibility 5](#_Toc442453666)

[Business case 5](#_Toc442453667)

[Data description 6](#_Toc442453668)

[Expected results 6](#_Toc442453669)

[Algorithms study 6](#_Toc442453670)

[Scope of the application (limits, evolutions) 6](#_Toc442453671)

[Conception 7](#_Toc442453672)

[Chosen algorithm 7](#_Toc442453673)

[Architecture 7](#_Toc442453674)

[Overview 7](#_Toc442453675)

[Architectural MVC pattern 7](#_Toc442453676)

[DAO & template pattern 7](#_Toc442453677)

[Search strategy pattern 7](#_Toc442453678)

[Global application flow 8](#_Toc442453679)

[Global schema and major features schema 8](#_Toc442453680)

[Console operations description 9](#_Toc442453681)

[<One section by operation> 9](#_Toc442453682)

[Configuration instructions 9](#_Toc442453683)

[Commented Screenshots 9](#_Toc442453684)

[Bibliography 9](#_Toc442453685)

# Subject description

This project is part of the Java course given at EPITA. To test and show a sufficient amount of knowledge a software application will be designed and developed.

As basis of the application the content created during the lectures will be used. The functionality will be extended where necessary to create the application.

In the subject analysis the requirements, features and results will be set.

The Conception chapter displays the design of the application.

# Subject analysis

The requirements for the project are subtracted from:

<http://www.thomas-broussard.fr/work/java/courses/project/fundamental.xhtml>

## User Requirements

The required features of the application are:

1. Do operations on identity information
   1. Access
   2. Create
   3. Modify
2. Persist Identity data in a database
   1. E.g. Database
3. Work process
   1. Be robust
   2. Capable of good performance
4. Authentication
   1. E.g. Login + password.

Additional features:

1. Propose a simple but efficient user interface

## Application Feasibility

To determine the feasibility of the application we look at the requirements.

|  |  |  |
| --- | --- | --- |
| Requirement | Analysis | Risk |
| 1.1, 1.2, 1.3 | These operations are handled during the class and are well known, can be applied in multiple ways. | Low risk |
| 2.1 | Create a database connection and doing operations are handled in class, still there minimal knowledge about connections, drivers and SQL statements in the team. | Medium risk |
| 3.1 | To enforce a robust application unit testing critical parts should be performed.  Code review or pair programming will be done on request by the implementer. | Low risk |
| 3.2 | ‘Good performance’ requirement is a broad and vague specification, especially since the target audience is also unknown. We will re-define good performance for the chosen target as following;   * Error handling and reporting to the end user. * Responsible GUI, (no GUI freezes > 1s) | Medium risk |
| 4.1 | Authentication can be done in multiple ways since there is no specification we will chose the method which is most suitable to the use case. | Low risk |

Table - Requirement analysis

### Business case

Patient management -> Explain

## Data description

From the requirements we can extract the necessary data models. There is one obvious model which is the ***Patient*** model, this model contains all the details of a single patient. Also a ***User*** model will be created which will contain the details and login information about the users of the application.

|  |  |
| --- | --- |
| Patient | User |
| Front name | Name |
| Last name | Password |
| Room | Application rights |
| … |  |

Table – Data models

## Expected results

## Algorithms study

|  |  |  |
| --- | --- | --- |
| Subject | Description | Solution(s) |
| Saving an POJO |  | JDBC, XML, File |
| Patient & User DAO |  | Template pattern |
| Searching patients |  | Strategy pattern |
| Business logic and View separation |  | MVC pattern |
| JDBC Connection management |  | Connection pool |
| User password security |  | Encryption |
| (optional)View update |  | Observer pattern |

Table - Algorithm study

## Scope of the application (limits, evolutions)

# Conception

## Chosen algorithm

## Architecture

### Overview

The general architecture of the application is depicted in Figure 1.

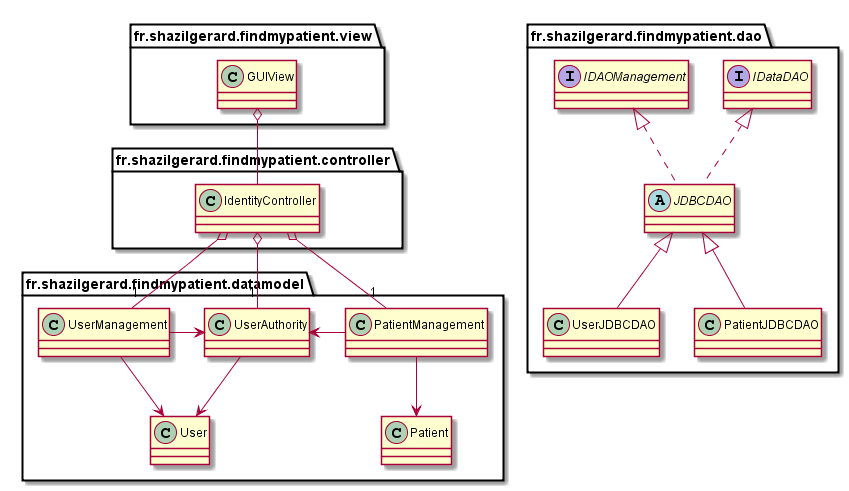


Figure - General application architecture

### Architectural MVC pattern

The application base design is around the model view controller structure. This should ensure that the business logic is separated from a view. The controller will instantiate all the necessary components to run the business logic.

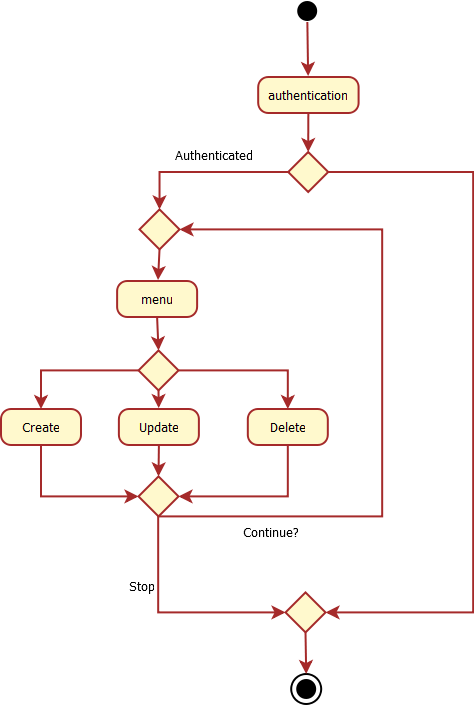
### DAO & template pattern

Since there are two data models which both require the same CRUD operations, it is chosen to create an abstract JDBCDAO which will implement the general CRUD implementation to a database. The specifics for each model will be handled in the specialization classes UserJDBCDAO and PatientJDBCDAO.

### Search strategy pattern

Searching for specific fields of both models will be implemented using the strategy pattern. This ensures that new search methods can be easily added.

## Global application flow



## Global schema and major features schema

# Console operations description

## <One section by operation>

# Configuration instructions

# Commented Screenshots

# Bibliography