Insert here your thesis' task.

Czech Technical University in Prague
Faculty of Information Technology
Department of software engineering

Bachelor's thesis

Specialized Information System Maintaining Patients Participating in Epileptosurgical Programme – Reporting Module

Martin Dvořáček

Supervisor: Ing. Petr Ježdík Ph.D.

21st April 2014

Acknowledgements

This thesis has benefited greatly from the support of many people, some of whom I would sincerely like to thank here.

To begin with, I am deeply grateful for the support and guidance of Mr. Ježdík with whom I have a possibility to cooperate for past two years. Thanks to him, I could work on such an interesting topic as implementation of an information system for a the biggest hospital in the Czech Republic.

Also, I owe special thanks to all my classmates that have somehow participated in the process of development of the Genepi IS.

Finally, but first in my heart, my parents and my brother are due my deep gratitude for their continued moral and financial support throughout my studies, the former being of much greater importance. The broad education that I was able to enjoy while growing up has proven invaluable.

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 a school work under the provisions of Article 60(1) of the Act.

In Prague on 21st April 2014 .....

Citation of this thesis

Dvořáček, Martin. *Specialized Information System Maintaining Patients Participating in Epileptosurgical Programme – Reporting Module.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2014.

Abstract

The goal of this thesis was to implement an export module to Genepi IS. Thanks to this module it is possible to export data stored in the system to varied formats according to the user's choice. User may also adjust the range of exported data, according to his needs. Key requirements were robustness, reliability and security.

Keywords bla

Abstrakt

Cílem této práce byla implementace exportovacího modulu do Genepi IS. Díky tomuto modulu je možné exportovat data uložená v systému do rozličných formátů na základě výběru uživatele. Uživatel také může upravit rozsah exportovaných dat na základě svých potřeb. Mezi klíčové požadavky patřila robustnost, spolehlivost a bezpečnost.

Klíčová slova export dat

Contents

List of Figures

Chapter 1

Introduction

The reporting module, part of the GENEPI - the information system, adds an important functionality to this software. Thanks to this module the user will be able to export data saved in the system to sundry formats. This is useful for the doctors that take care of patients with epilepsy, as well as for the researchers that make analysis above the data from this system. In this thesis I'll describe the design and implementation of the reporting module, as well the process of final testing of this product.

Chapter 2

Analysis and design

The repoting module, as well as the whole information system, was designed given the requested robustness, accessibility, reliability and the cleanness of the source code. GENEPI has a three-tier architecture, uses access according to the roles of the users via Spring Security and thanks to the optimalized data layer it saves the computing resources.

Due to the fact, that users, who should work with the exported data, have different levels of access and different requirements on the format of the exported data, there is a need to make the module to be able to anonymize sensitive data as well as to export data to different formats. The contracting authority also requested that the user could choose which data does he want to export. For research purposes is also important to determine, if data that should be exported, contain only patients, whose data were verified by the intrusted user.

2.1 GENEPI - the information system

That the reader could fully understand the functioning of the reporting module, first of all it is needed to introduce the information system that it extends. GENEPI - the information system was being created within the subjects BI-SP1 and BI-SP2 on the Czech Technical University, Faculty of Information Technologies in the school year 2012/2013 as a student's project. It should replace the original information system that was used in the Faculty Hospital Motol in Prague for maintaining patients in epileptosurgical programme. The main reasons for replacing the original system was a fact, that it didn't comply with the current requirements of the medics, contained bugs and its design wasn't optimal. GENEPI IS is being developed in the java programming language, using frameworks Spring 3 and

Hibernate 4.2. Front-end part is realized via JSP, using HTLM5, Javascript and CSS. Libraries that were used to ease the programming of the front-end were Twitter Bootstrap and jQuery. As a database has been chosen MySQL 5.5 and as an application server Apache Tomcat 7. All of the used libraries are distributed under some kind of free license. The GENEPI is a bilingual software due to the expected deployment in the Miami Children's hospital in Florida, the United States of America. Languages that are currently supported, are czech and english, nevertheless thanks to the suitably implemented localization it is easy to extend the application and add any other language. During the whole period of the programming of the system, the team consulted the approach often and regularly with contracting authority, to fully meet the needs of the medics.

2.2 Design of the reporting module

Architecture of the reporting module doesn't differ from the architecture of the rest of the information system. Thanks to this, I could guarantee the robustness and the security of this module. It has also provided me an easy and elegant way to access the other components in the system.

In this tree diagram you can see the parts of the IS, that concern the reporting module:

```
Backend
└── src
    └── main
        ├── java ..... java classes
        │   └── cz
        │       └── cvut
        │           └── fit
        │               └── genepi
        │                   ├── businessLayer
        │                   │   ├── service ..... interface of the services
        │                   │   ├── serviceImpl .. implementation of services
        │                   │   └── VO ..... view objects
        │                   ├── dataLayer
        │                   │   ├── DAO ..... interface of the DAO
        │                   │   ├── DAOImpl ..... implementation of DAO
        │                   │   └── entity ..... entities
        │                   └── presentationLayer ..... controllers
        ├── resources ..... property files
        └── webapp ..... files for front-end
```

```
├── resources ... front-end libraries, pictures and other media
└── WEB-INF
    ├── tags ..... templates for jsp
    └── views ..... jsp files
```

2.2.1 Design of the back-end part

Back-end part of the reporting module was designed to follow the three-tier architecture of the rest of the system. Thus the classes that this modul uses, are devided into three different packages. Every package contains classes that belong to the same tier. These packages are called Presentation layer, Data layer and Business Layer.

2.2.1.1 Presentation layer

In the presentation layer there is a controller - a Spring MVC component that calls function according to the HTML request and its URL that is mapped to a particular function. This function executes simple operations such as verification of passed parameters and calls to the other layers. The results of the functions from other layers may be saved to a org.springframework.ui.Model object and passed to front-end. This layer never executes any more complicated actions, as those should be done within the business layer. Functions of a controller usually return the name of the view that should be displayed to a user.

2.2.1.2 Business layer

There are three packages in in this tier - concretly those are service, serviceImpl and VO. Service contains interfaces whereas in serviceImpl are the classes that implement those interfaces from service package. It is analogical as in the data tier which I will describe further in the next paragraph. In ServiceImpl classes there is usally implemented some more complicated logic, that shouldn't be implemented within presentation layer, like adjusting of VOs, sorting of lists of objects under some conditions and many other actions. The third package - VO - contains classes, that are used to transform classes from a form in which the data come from the database - in enitity classes - to a form, that is more propriate for work within business tier.

2.2.1.3 Data layer

This layer contains three packages - DAO, DAOImpl and entity. In DAO package, there are interfaces of the data access objects, in DAOImpl are the classes that implement the interfaces from DAO package and finally in entity package, we can find the entity classes. In DAOImpl classes you can find the functions, that are used to access the database and to get some data from it. There is a direct use of hibernate functions or queries in HQL. In entity classes is described, how the structures of the tables in the database look like. Every entity contains a property for every single column of the table. The type of this variable must match the type of the column of the table. Every property has to have a getter and a setter as well. Thanks to anotations from javax.* and org.hibernate.validator.constrains.* it is possible to check the requirements for attributes of the input that should be saved to the database, ie. minimal or maximal size, regexp pattern or if the input may be blank. Very important is hibernate anotation @Column, as it describes the mapping of the property to the certain table in the database.

I would like to point out the generic GenericDAO and GenericDAOImpl classes that have been implemented. These are abstract classes that were implemented to ease the process of programming of the other DAO and DAOImpl classes. In the GenericDAOImpl class there are implemented the most used methods, that may be used by any entity such as save, delete, getCount, getCountOfUnhidden, findByID and many other. The other DAO and DAOImpl methods extend these generic classes so then it wasn't needed to implement these basic functions to every single class and the programmer had to programme only those functions, that were unique for that certain class.

2.2.2 Design of the frontend part

Chapter 3

Realisation

All that will be described in this chapter is located in the business layer. The standard approach during performance of an export is following:

1. The user chooses in the export view the properties that wants to include to the exported file, the format, anonymization option and hits the export button

2. PatientExportPOST method in PatientController entity is called and according to the given parameters is calls method to export the data to the right format. It also checks if the data should be anonymized and if the user isn't authorized to see sensitive data, then it sets the anonymize flag to true automatically. All given parameters with all those informations are passed to called export function in the Bussines tier.

3. In this step we've moved from Presentation tier to Business tier, where one of the methods for export to certain format was called. This is the part, where the data is being written to the file itself. The program iterates through all the patient that user wanted to export and gradually prints out their data.

   At first the patient's header is printed out. This header contains information about patient such as his age, gender, age at the beginning of the epilepsy and other important data. It may also contain sensitive data such as his name, address or other contact

popsat obecny postupy pri realizaci a na co se zvlast kladl duraz

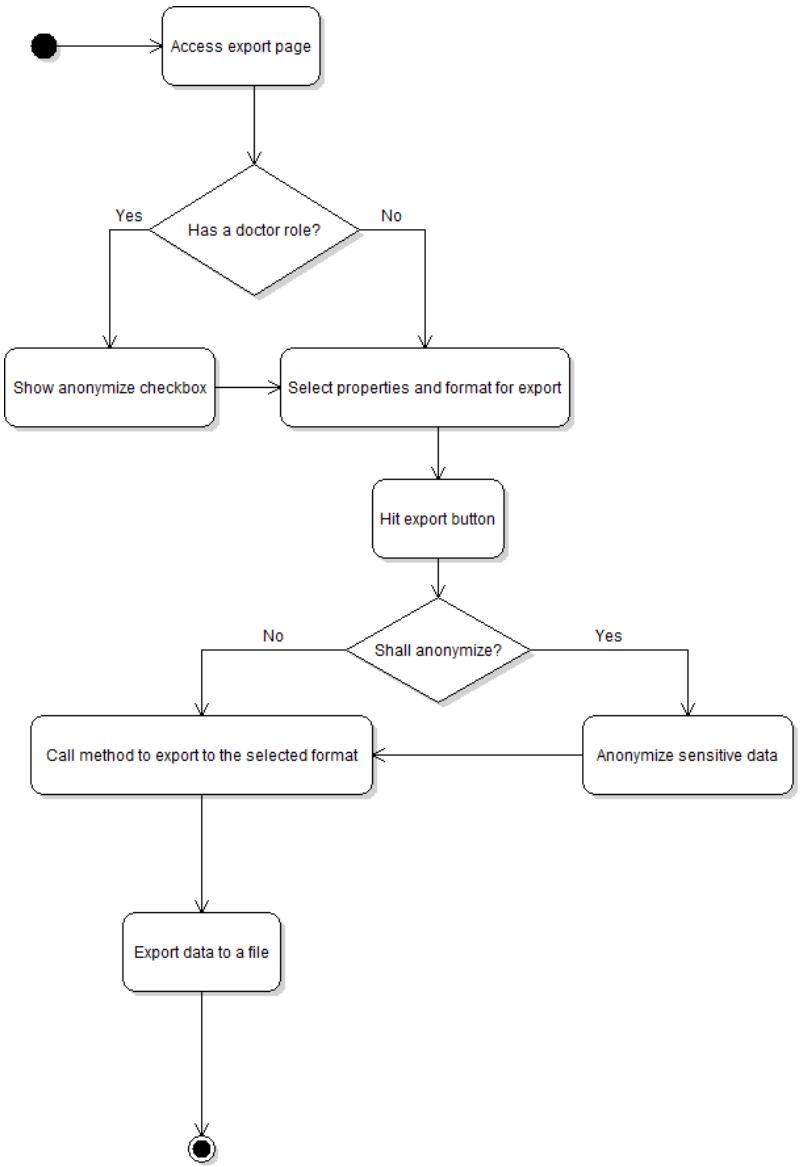

Figure 3.1: export activity diagram

3.1 Customization of the export

There was a strong demand from the contracting authority to create a module that would let the user to customize the report. This is because every patient can have filled in fifteen cards. This means circa 340 properties that can be stored for every single patient. Nevertheless not all of them are needed to be exported in certain situations. Therefore was needed to implement some solution, that would allow user to select only those cards or properties of those cards, that should be exported. Due to a high amount of those properties, it was quite challenging to create solution, that would meet the requirements and will be user-friendly at the same time. I've chosen treeview on the fron-end and special entity on the back-end.

On the front-end there was implemented a customized treeview, where every item has a checkbox. Patient's cards are represented by the nodes, whereas porperties of these cards can be represented by the leaves of this tree or nodes as well. They may be nodes, if they denote some category that can hold some properties that couldn't exist withou their parent cathegory. By checking and unchecking these items, the user can choose, what all should be involved in the exported data. When unchecking the node, also all leaves of this node are deselected. On the other hand, by selecting even a single leaf from an unselected branch, the parent node is selected as well. Here I attach a simple example of the treeview I was writting about:

```
Anamnesis.....anamnesis card
    Anamnesis property
    Another anamnesis property
Seizures.....seizure card
    Seizure property
    Another seizure property
```

After confirming the export form, the state of the treeview is saved to an instance of an entity called ExportParams. This entity has a boolean property for every single value from the treeview in the export form. Thanks to this, I am able to determine, which properties the user wanted to export and which didn't. It gives me another advantage - I can save the current setting of the treeview for a later usage, but I will write yet about this feature later.

3.2 Export to the particular formats

During the programming of the classes that procure the logic of the reporting, I was trying to use the fact, that there already exist java libraries

that can export data to docx, pdf, xls and csv formats. I also avoided the duplication of the code by transforming data from one format to another. Thanks to these measures, it is much easier to do changes in the code of the classes that handle export itself and it also eases the understanding of the code for anybody, who would read it.

3.2.1 txt

Export to the text format is realized by components form java.io.* package. Specifically java.io.FileOutputStream, java.io.OutputStreamWriter and java.io.BufferedWriter. Output is encoded to UTF-8 format. Every property is printed out to a new line, sections are delimited by dash lines, star lines or empty lines.

3.2.2 docx

Export to the docx format was not as easy as to the txt, so I decided to look up suitable libraries, compare them and use that one, that would best suit my needs. After researching the possible solutions I ended up with two libraries - apache POI and docx4j. o POI o docx4j

I've chosen to use docx4j because it provided me more functionalities and clearer API. The main problem I had with Apachi POI was the formating of the cells, when I wanted to to format data to the tables. I could hardly adjust the size of the cells and redefine my own styles of headers and text.

3.2.3 xls

As well as for the docx format, I used that there already were programmed libraries for export to xls. I've chosen to use Apache POI library, as this library provides me anything I could need to export data to a xls file.

3.2.4 pdf

There are of course java libraries that provide you some way to export data to pdf as well, nevertheless I've chosen different approach. While I already had implemented the export to docx, I've chosen not to implement export to pdf, but to create file with data exported to docx and using the classes form apache.poi.xwpf.converter.* package convert this file to pdf. Thanks to this approach I didn't duplicate the code because the only logic needed to format and print out the data for export was already implemented in the export to docx.

3.2.5 csv

Before I started to implement the export to csv format, I was also thinking, if I should implement whole logic of export to csv, but then I decided to proceed similarly as in the case of implementation of the export to pdf. Nevertheless in this case I don't use file with data exported to docx, but file created by export to xls, as this format can be easily transformed to csv and vice versa. During the export to csv I create at first xls file with exported data and then I walk through this file and print out the values to the file via the classes from java.io.* package. These are the same classes, that have been used in the export to txt. Based on the requirements of the contracting authority, the comma, as a standard delimiter in csv format, has been replaced with semicolon. The main reason of this change was a fact, that every card, that patient may have, contains comment, which is a text, which may also contain commas. As the semicolon is a much less used character in the sentences, it was decided to use it as a delimiter. Otherwise the report could look confusingly for the user.

3.3 Security and anonymization

In GENEPI - the information system there are 3 main levels of an access, according to the visibility of the patient's data.

1. Administrators

   Administrators don't have any access to the patient's data. They are not even able to access the URL to view or export these data.

2. Researchers

   Researchers have limited access to patient's data - they are not allowed to see sensitive data.

3. Doctors Doctors have full access to patient's data

Because of this fact, it was needed to implement some way, how to anonymize the exported data. I have implemented two types of anonymization - volitelnou a povinnou.

Optional anonymization is accessable only for the doctors. Before they commit the export, they can decide if they want to anonymize the exported data. This could be usefull for example in situations, when the doctor needs to provide the data to somebody, who normally doesn't have an access to the system and can't see the sensitive informations at the same time. On the

other hand, the doctor still has a possibility to export unanonymized data, which might be usefull for the medical purposes. Mandatory anonymization is done automatically, when somebody, who doesn't have sufficient access level wants to export data. This feature will prevent anybody unauthorized from seeing the sensitive data of the patients.

3.4 Custom configurations

3.5 Graphical user interface of the export module

On the following pictures I'll explain all components, that are used to controll the export view. During the phase of design and implementation of the view, I had to take into account the fact, that users of this module need a user-friendly interface that could provide them all tools that need to customize exports.

On the first picture, you can see the interface for loading and deleting of custom configurations. In the first row, there is a combobox with generic custom configurations which the user may load or even delete, if he's a creator of this configuration.

On the next picture there is displayed a section for export itself. Firstly there is a list of IDs of patients that are prepared for an export. The user may access the patient's overview by clicking to his ID. Next it's needed to choose the format for export via several radiobuttons. In the case, that user chooses as a format docx or pdf, the NEJAKA KOMPONENTA appears and lets him to choose if exported data should be formated into tables or as a text. If the user has a doctor role assigned, then he's able to see a combobox, that allows him to anonymize exported data.

Chapter 4

Testing

Chapter 5

Conclusion

Appendix A

Acronyms

DAO Data access object

GUI Graphical user interface

HQL Hibernate query language

XML Extensible markup language

VO View object

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `exe` ..... the directory with executables
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `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