

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.

2nd May 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 2nd May 2014

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2014 Martin Dvořáček. All rights reserved.

This thesis is a school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

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

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 2 | Analysis and design | 3 |
| 2.1 | Genepi - the information system | 3 |
| 2.2 | Design of the reporting module | 6 |
| 3 | Realisation | 9 |
| 3.1 | Customization of the export | 10 |
| 3.2 | Export to the particular formats | 12 |
| 3.3 | Security and anonymization | 13 |
| 3.4 | Custom configurations | 14 |
| 3.5 | Graphical user interface of the export module | 15 |
| 4 | Testing | 19 |
| | Conclusion | 21 |
| A | Acronyms | 23 |
| B | Contents of enclosed CD | 25 |

List of Figures

| | | |
|-----|---|----|
| 3.1 | export activity diagram | 10 |
| 3.2 | create configuration activity diagram | 16 |
| 3.3 | delete configuration activity diagram | 17 |

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. The goal was to extend the IS with another functionality without letting the user to notice he's even using any extension. Therefore the reporting module was CITLIVE ZAPRACOVAN DO STAVAJICIHO source code of the Genepi IS, using the same NAVRHOVYCH VZORU A POSTUPU as the resto of the IS.

In this thesis I'll describe the design and implementation of the reporting module, as well the process of final testing of this product.

Analysis and design

The reporting 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 driven by Spring MVC module, uses access according to the roles of the users via Spring Security module and thanks to the optimized data layer that uses Hibernate framework, it saves the computing resources and eases the developers to understand and extend the source code.

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 was 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 information 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 HTML5, 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 software and libraries are distributed under some kind of free license. Genepi IS 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.

The frameworks that were used to implement the back end are:

1. Spring MVC

BLABLABLA PAR SLOV OBEČNE O HTML + k čemu se konkrétne
používa v Genepi BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA

2. Spring Security

BLABLABLA PAR SLOV OBEČNE O HTML + k čemu se konkrétne
používa v Genepi BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA BLAB-
LABLA BLABLABLA BLABLABLA BLABLABLA BLABLABLA
BLABLABLA BLABLABLA BLABLABLA BLABLABLA

3. Spring Hibernate

4. Twitter Bootstrap

Twitter Bootstrap is the most popular front-end framework for developing responsive, mobile first projects on the web. CITO VAT <http://getbootstrap.com/>. It is build with HTML, CSS and JavaScript. Thanks to this framework it was much easier to create responsive front end with PRIJEMNY and ergonomic look and feel.

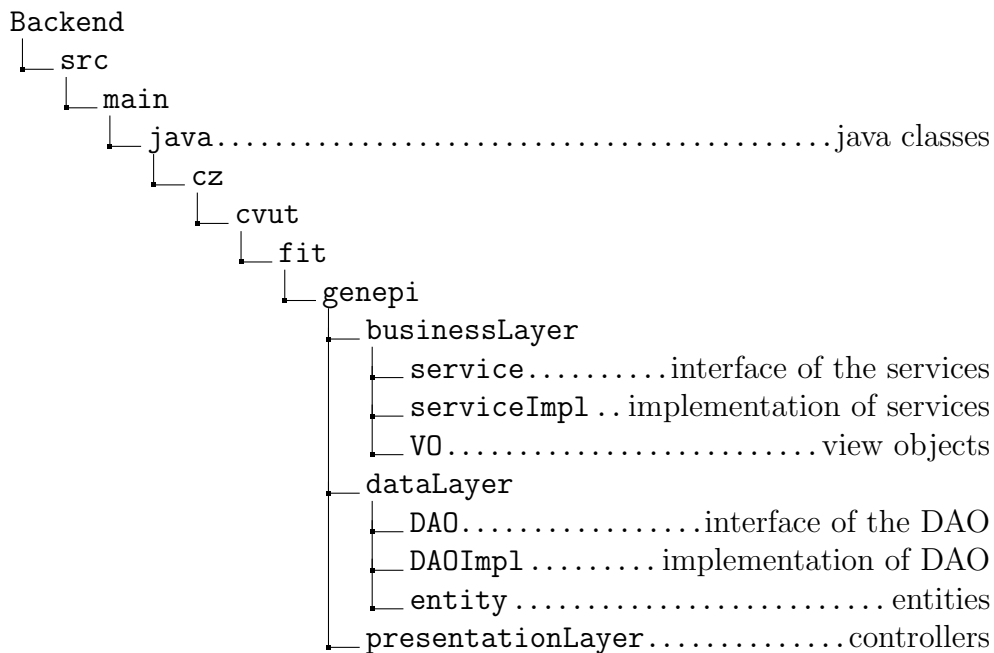
5. jQuery

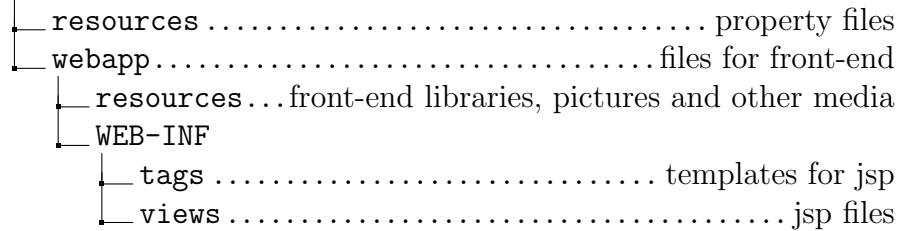
jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write [?] JavaScript.

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 may see the parts of the IS, that concern the reporting module:





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, Business layer and Data 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 nor doesn't access the database as this should be done within the Data layer. Functions of a controller usually return the name of a view - a .jsp file - 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 serviceImpl contains classes that implement those interfaces from service package. It is analogical as in the data tier which I will describe in great details 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 the

entity classes - to a form, that is more appropriate for work within Business tier.

2.2.1.3 Data layer

This layer contains three packages - DAO, DAOImpl and entity. In the 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. If these conditions wouldn't be satisfied, the DOTAZ NAD DATABAZI would fail and the AS would print out NEJAKA HIBERNATI EXCEPTION.

Every property has to have a getter and a setter as well. Thanks to annotations from javax.* and org.hibernate.validator.constraints.* 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 or NULL.

Mandatory is also a hibernate annotation @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 GenericDAO class is interface of GenericDAOImpl class, as in the usual in the rest of the project.

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 front end part

As well as the back end part, the front end had to follow the style and design of the rest of the system. If I wouldn't do so, the work with reporting module could be confusing for users. On the basis of these facts, it was needed to implement frontend based on the same technologies as the rest of the Genepi IS. Thanks to this approach the UX from work with this module is positive and user accepts reporting module as a casual part of the IS.

Realisation

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

1. The user chooses in the export view the properties that he wants to include to the exported file, the format, anonymization option and hits the export button
2. PatientExportPOST method in PatientController class 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 information.
4. In the last step is the user redirected to an URL of the newly created file with exported data to download. This file persists on the server and can be repeatedly downloaded. The time that is this file

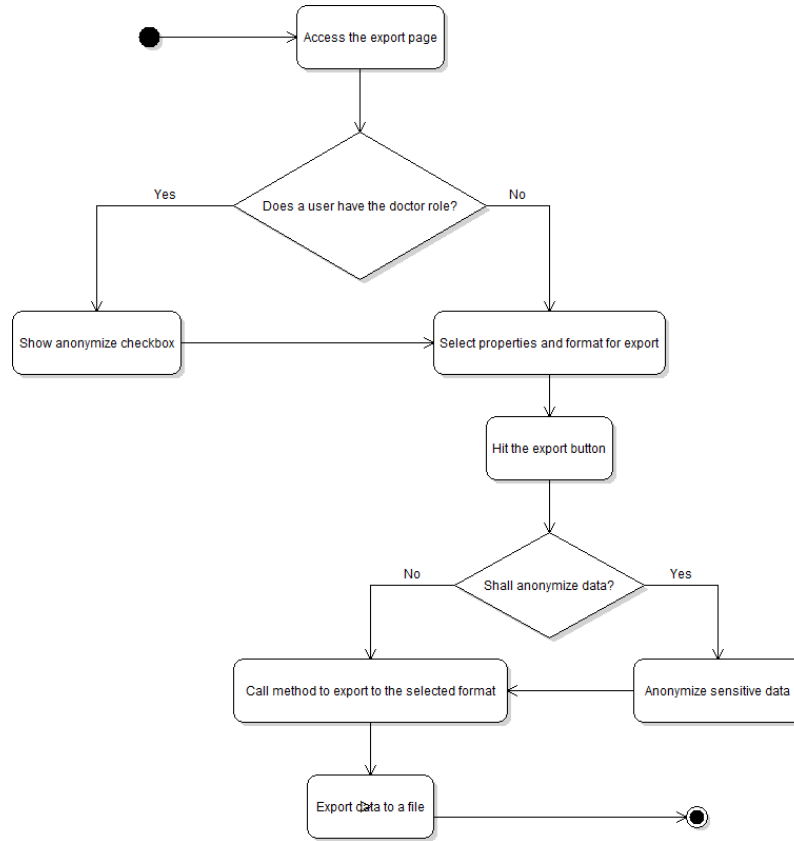


Figure 3.1: export activity diagram

accessible depends on the setting of the server CRON, as this should regularly delete old files in the directory dedicated for export files.

The goal was to make the whole process secure, reliable and easy to understand for programmer who would read the code, as well as for the user that would use the export module. Thanks to chosen design and implementation approach were these requirements satisfied.

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 front 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 properties 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 without their parent category. 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 writing about:

```
├─ Anamnesis.....anamnesis card
│   └─ Anamnesis property
│   └─ Another anamnesis property
├─ Seizures.....seizure card
│   └─ Seizure property
│   └─ Another seizure property
├─ ...
├─ ...
├─ Neuropsychology.....neuropsychology card
│   └─ Neuropsychology inner node
│       └─ Neuropsychology property
│       └─ Another Neuropsychology property
│   └─ Another Neuropsychology inner node
│       └─ Neuropsychology property
│   └─ Neuropsychology property
├─ ...
├─ ...
└─ Outcome ..... outcome card
```

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 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 from `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.

I've chosen to use docx4j because it has provided me more functionalities and clearer API than Apache POI could. The main problem I had with Apache POI was the formatting of the cells, when I wanted to format data to the tables. I could hardly adjust the size of the cells and redefine my own styles of headers and text. In docx4j provide really easy and elegant way to solve these issues.

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, such as merged cells, creating of new lists, .

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 from `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 information system there are 5 main levels of an access, according to the visibility of the patient's data.

1. Users User role is a basic role that every new user gets from the system by default. This role allows you to access the homepage and your user profile. Without any further role, the user is not allowed to perform any further actions.
2. 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.
3. Researchers

3. REALISATION

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

4. Doctors

Doctors have full access to patient's data, they can add new patients and modify their data. Nevertheless after every change, the patient is set as unverified. This means, that those changes weren't checked by an authorized person, can't be with certainty trusted and shouldn't be included to any research. Doctors by default doesn't have authority to verify patients whose data were changed and are set as unverified.

5. Superdoctors

Users with role Superdoctor have the same privileges as users with role Doctor. Moreover they have the right to verify unverified patients.

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

Optional anonymization is accessible for doctors and superdoctors only. Before they commit the export, they can decide if they want to anonymize the exported data. This could be useful 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 useful 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. Anonymized data will be mostly used by researchers, who make analysis upon the data and don't need to know the names and addresses of patients, because they can distinguish them according to their IDs, which is a mandatory property that is printed out for every patient in every export.

3.4 Custom configurations

Customization of the configuration of exported data, was one of the key features of this module. Due to the fact, that there are several types of users who work with this IS, it's needed to count with that, that the ways how they need to use it differs. Researchers usually do some analysis above some part of the data from the IS, whereas doctors usually need some more complex report about patient's health state and the process of the treatment.

It is obvious, that settings of these configurations will be needed regularly, as it is not expected that users would need to create unique configuration for every single export he'd like to perform. Because of this, it was decided that users should have an option to save their adjusted configuration for a later use. Export module for Genepi IS distinguishes two types of saved configurations.

First of them is configuration saved by a user with the Superdoctor role. This configuration may be set as generic. That means that the user chose to let his configuration to propagate through the system, so other users may see and use it.

The other type are user's private configurations. Every user with sufficient access level to access the export view is allowed to create and use them. These are configurations that were created by a user himself and are available for him only. The other users are not able to use these configurations, nor to see them.

Deleting of configurations is possible as well. Private configurations can be deleted by a user that can see them in his list of private configurations list and not by anybody else.

On the other hand generic configurations are visible to all users in the system, therefore it is needed to determine, if a user who wants to delete some generic configuration, has even the right to perform this action. To delete a generic configuration, the user has to own a Superdoctor role. Then he's allowed to delete any generic configuration created by any other user. If user doesn't have the Superdoctor role, then the delete button for generic configurations is not even displayed to this user in the export view. On the following pictures you can see the activity diagram of saving and then deleting of a configuration from the point of view of the user.

3.5 Graphical user interface of the export module

On the following pictures I'll explain all components, that are used to control 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

3. REALISATION

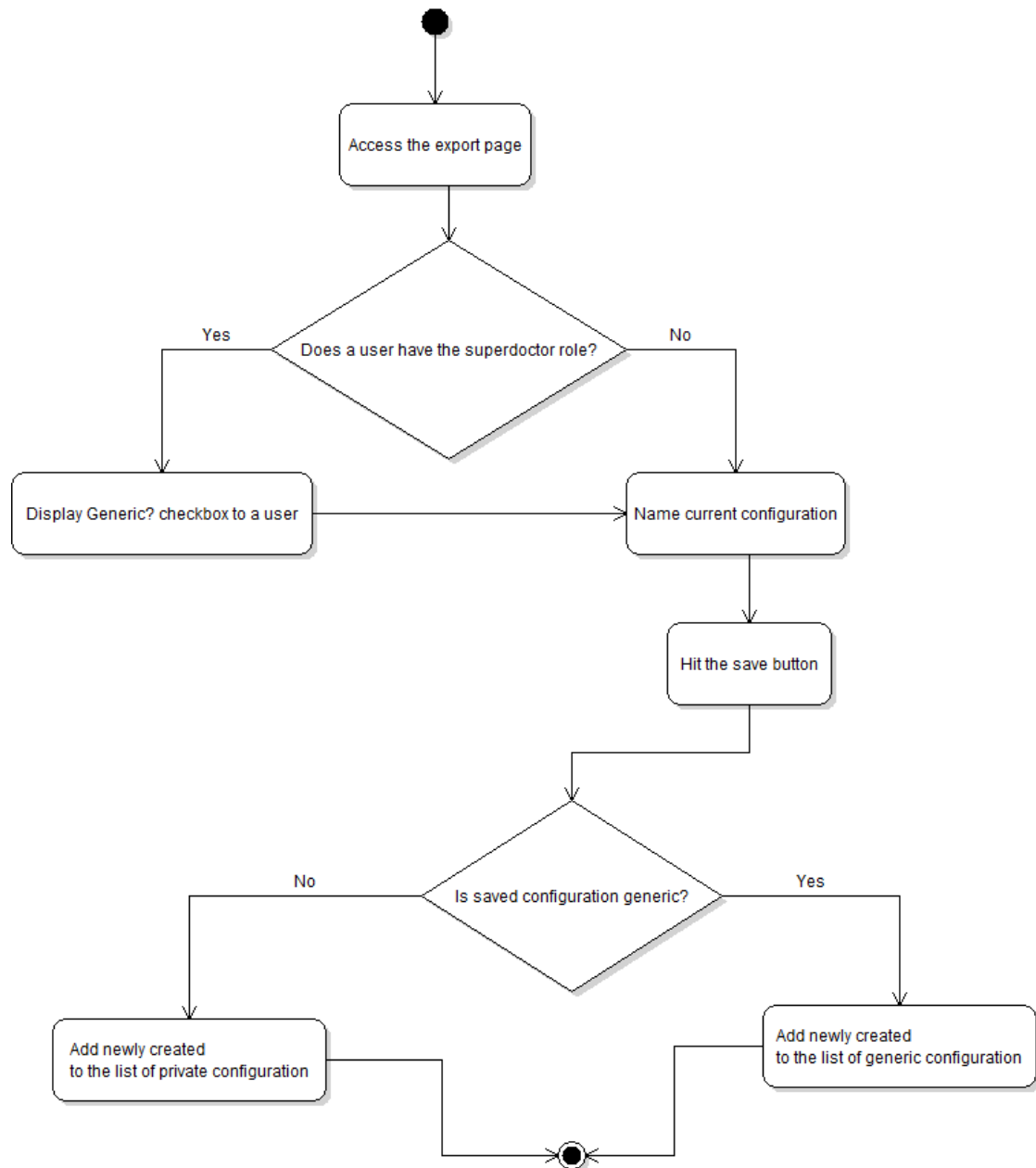


Figure 3.2: create configuration activity diagram

3.5. Graphical user interface of the export module

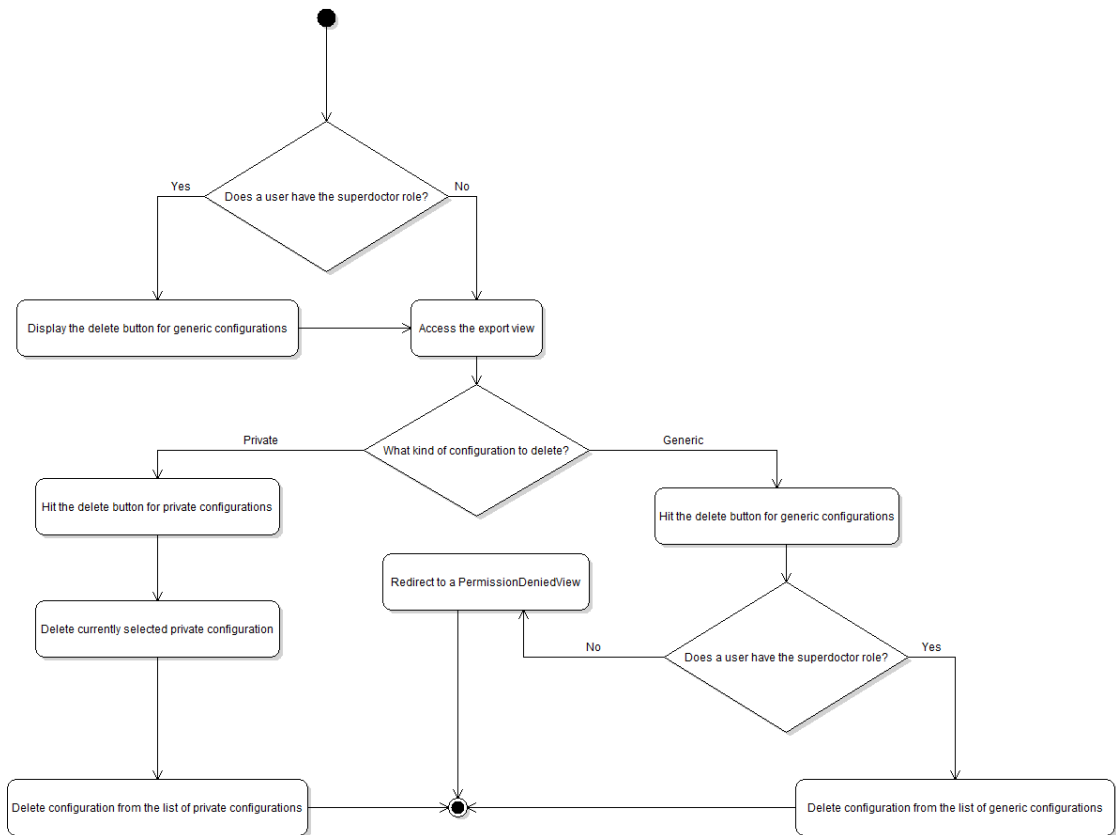


Figure 3.3: delete configuration activity diagram

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 formatted 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.

Testing

Testing of the export module as well as the Genepi IS itself was being performed during the development by the developers themselves and by a member of the team who has held the position of a tester. After key functionalities were implemented, Ing. Jezdik Ph.D got the access to the system and tested it on usual use cases. He of course had many change requests that could enbetter the application or found bugs that should be fixed. Nowadays the Genepi IS with export module should be deployed on produciton servers in the hospital and fully NAHRADIT STAVAJICI applicaiton. As in most of the cases from commercial area, it is awaited, that users will also have some change requests. But those are expected to be minor change requests, mostly on a front end and should be implemented within a few mandays.

Conclusion

The goal of my bachelors thesis was to design and implement an exporting module for Genepi IS. This extension should be robust, reliable, secure with user-friendly GUI.

Acronyms

AS Application server

DAO Data access object

GUI Graphical user interface

HQL Hibernate query language

IS Information system

JSP Java server page

MVC Model view controller

XML Extensible markup language

VO View object

Contents of enclosed CD

To my bachelors thesis I enclose a CD, that contains all important data, that are needed to start your own Apache Tomcat AS, MySQL database and depoloy Genepi IS. It also contains source codes of Genepi IS and libraries that it uses. In the Database directory is also enclosed model for MySQL Workbench, where the reader may see the structure of the database. In the same folder you can also find migration scripts. These are important when migrating from the software that was previously used in FN Motol to Genepi IS. Migration scripts allow you to copy data from previously used database to the databse that is being used by the Genepi IS. Thanks to this, the reader should be able to administrate or even modify Genepi IS without any larger problems. If the reader would like to use enclosed Apache Tomcat AS and/or MySQL database, he has to agree and follow the licence agreements that are enclosed in the same directories as the software, that is being distributed under them

Next important directory is the directory with samples of exported data. This folder contains 14 files with all possible formates of exported data - pdf, docx, xls, csv, txt - all of these formates in both anonymized and unanonymized varieties and both printed to as a plain text and formatted to a table for pdf and docx.

The reader may also find useful directory that contains deployment packages of the results of the previously mentions projects from the subjects BI-SP1 and BI-SP2. The documentation to these projects is included in that directory.

Another content of enclosed CD is text directory, that contains this this theis in pdf and ps format and readme.txt file, where is in great deatails described the content of the CD.

B. CONTENTS OF ENCLOSED CD

| | |
|------------------------|--|
| readme.txt | the file with CD contents description |
| Apache Tomcat | the directory that contains Apache Tomcat |
| Database ... | the directory that contains MySQL database and database scripts |
| MySQL | the directory that contains MySQL database |
| data .. | the directory that holds model of the database and migration scripts |
| war | the directory with war file |
| Backend | the directory of a maven project |
| src | the directory of Genepi IS source codes |
| main | |
| java | |
| cz | |
| cvut | |
| fit | |
| genepi | the directory of source code of an implementation |
| resources | the directory configuration files |
| webapp | |
| resources | the directory of librarieres for front end |
| WEB-IF | |
| spring | the directory for spring context file |
| tags | the directory of templates for jsp pages |
| views | the directory of jsp pages |
| test | the directory of source codes of unit tests |
| Previous projects | the direcotry of results of BI-SP1 and BI-SP2 |
| BI-SP1 | the directory of the BI-SP1 project |
| BI-SP2 | the direcotry of the BI-SP2 project |
| Sample Data | the directory that contains samples of exported data |
| Text | the thesis text directory |
| thesis.pdf | the thesis text in PDF format |
| thesis.ps | the thesis text in PS format |