CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER THESIS

2009 **Jan PADĚRA**

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

STATEMENT OF A MASTER THESIS

Student: **Bc. Jan Paděra**

Master Study Program: Electrical Engineering and Information Technology N2612/2259
Specialization: Computer Science and Engineering

Thesis Title: **Web Medical Interface Development**

Guidelines:

1) Perform feasibility study of the web medical interface
   a. First, focus on elderly as a target group
   b. Second, extend the study on other stakeholders as GP, nurses, dietician, social workers and elderly family.
2) Design the web portal framework using Java technology.
3) Implement the web portal as desktop and mobile application.

References: Will be provided by the supervizor

Supervisor: Ing. Daniel Novák, Ph.D.

Validity until: January 2010

Doc. Ing. Miroslav Šnorek, CSc.
Head of Department

L.S.

Doc. Ing. Boris Šimák, CSc.
Dean

Prague April 22, 2008

CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Department of Computer Science and Engineering

Web Medical Interface Development

Master Thesis

Author: Paděra Jan
paderj1@fel.cvut.cz

Advisors: Ing. Novák Daniel, Ph.D.
Dept. of Cybernetics, CTU FEE, Prague, Czech Republic
xnovakd1@labe.felk.cvut.cz

Cuesta Frau, David
Dept. of Systems Data Processing and Computers, UPV EPSA, Alcoy, Spain
dcuesta@disca.upv.es

Reviewer: Ing. Koutník Jan, Ph.D.
Dept. of Computer Science and Engineering, CTU FEE, Prague, Czech Republic
koutnij@fel.cvut.cz

Study programme: Electronics and Computer Science & Engineering, Structured, Connected Master's

Study of qualification: Computer Science and Engineering

Submitted: May 2009

Proclamation of the Author

I hereby declare that I have completed this master thesis independently and that I have listed all the literature and publications used.

I have no objection to usage of this work in compliance with the act §60 Zákona č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

Prague ……. …….

signature

Aknowledgments

I would like to thank Daniel Novák and David Cuesta, my supervisors, for their many suggestions and constant support during this research.

Jan Paděra

Abstract

The thesis implements two medical web portals for better management of diabetes and bipolar disorder under the framework of OLDES and Bipometrics projects. The first portal is aimed at patients suffering from diabetes. The portal provides visualization of physiological signals as blood pressure, level of glycemia and information about daily food consumption. The target group of the second web application are patients suffering from bipolar disorder. Information about their movement activity and results of questionnaires are accessible. In both cases alarms informing about potential health risks have been implemented. The study with focus on design an application for elderly and doctors was performed before the design and implementation of the both applications. Moreover, advantages of the improvement, related to developed projects, with respect to existed projects are discussed. The web portals are implemented on the basis of Java EE technologies such as Apache Struts, Hibernate or Apache Axis.

Abstrakt

V této diplomové práci byly implementovány dva webové portály, které umožňují snadnější léčbu a přístup k informacím pacientů trpících diabetem a bipolární poruchou. První portál je součástí projektu OLDES, který je zaměřený na pacienty s cukrovkou. Webová aplikace umožňuje náhled na záznamy fyziologických signálů jako je krevní tlak, hladina glykemie a informace o denním příjmu potravy. Cílovou skupinou druhé aplikace, která je součástí projektu Bipometrics, jsou pacienti trpící bipolární poruchou. Zde jsou přístupné informace o jejich pohybové aktivitě a výsledky vyplněných dotazníků. Oba portály byly navrženy tak, aby byli lékaři i pacienti okamžitě informováni v případě ohrožení zdraví pacienta. Před samotným designem a implementací obou aplikací byla provedena studie, která byla zaměřena na podobu aplikací pro starší obyvatelstvo a pro lékaře. Dále jsou diskutovány výhody, které s sebou oba projekty přinášejí s ohledem na již existující řešení. Webové portály jsou implementovány s použitím Java EE technologií jako např. Apache Struts, Hibernate nebo Apache Axis.

Contents

List of Figures

List of Examples

Chapter 1

Introduction

In these days we, as human beings, are surrounded with a wide range of the information and communication technologies. The improvement of the computer science, IT and any other similar technical branches is so fast that one can not follow all of the new products or devices. Nobody from the consume society can imagine living without computers, cell phones, the Internet, GPS navigation systems etc. What is really important is to try integrate the modern technologies to an environment where it can help people, especially older or chronic ill persons.

During my work, I have had the chance to join two e-Care projects where the main goal is to develop systems which enable vulnerable people to remain in their habitual environment and, thus, improving their quality of life, increasing their confidence and their safety. The first one – OLDES (Old people's e-services at home) – is focused on elderly people who suffer from diabetes. The second – Bipometrics – has dealt with patient with mental illness as its focus group. More detailed description about the both projects can be found below.

The main aim of this work is to develop medical interface which should serve to doctors/social workers or users who are involved in the health care process. The interface must be accessible through web browser and fulfil security requirements. The biggest technological demand is using only low-cost technology which are open-sources at best. Server side logic should be implemented in Java programming language. Due to these requirements the whole application is built on several open-source frameworks based on Java. As I mentioned above I have cooperated on two e-Care projects for different groups of patients but thanks to a lot of similarities in the users requirements I could share some of the source code or thoughts in both of applications. The short feasibility study was written before the implementation in order to estimate if the new system is worth making.

1.1 Overview of the Document

This thesis is organized into 7 chapters. After the *Introduction* chapter follows *e-Care Projects Description* where both developed projects are introduced. Furthermore, this part describes existing e-Care projects and compares them to OLDES/Bipometrics improvement. *Technology and Feasibility Study* analyses projects requirements. The suitable technology for implementation is also chosen within this study. *Design of the Web Medical Portals* discusses architecture which was designed for easy and corresponding implementation. The fifth chapter *Implementation* is focused on specific and interesting solutions with source code examples. The next *Testing* chapter describes results from OLDES pilot project testing. The last chapter *Conclusion* summarizes the whole work and offers possible future extensions.

1.1.1 Attached CD

The CD attached to this document contains:

- This document in PDF format
- war archive of the applications
- XML files for Bipometrics testing
- Complete DB schema
- Test scenarios

For more information see *index.html* file in the root directory of CD.

Chapter 2

e-Care Projects Description

In this chapter general information about e-Care projects is described. In general e-Care or e-Health systems are health care practices which are supported by ICT technologies. Common application for these types of systems is telemedicine [36] - *"Telemedicine is a rapidly developing application of clinical medicine where medical information is transferred via telephone, the Internet or other networks for the purpose of consulting, and sometimes remote medical procedures or examinations"*.

2.1 OLDES (Old people's e-services at home)

OLDES is EU co-funded project under the IST Programme[1)] that will offer new technological solutions to improve the quality of life of older people, through the development of a very low cost and easy to use entertainment and health care platform, designed to ease the life of the elderly in their homes [20].

The consequence of increasing life expectancy and decreasing birth rates is that population is becoming increasingly older [15]. This trend creates substantial problems in terms of resources needed for assisting elderly. 11 partners from European universities or companies participate in OLDES project. In the Czech Republic two partners are joint to the project - Charles University and Czech Technical University in Prague.

Czech partners are involved in diabetes pilot project which is focused on the management of older persons suffering from diabetes disease. This part can be divided in three different products which must cooperate together finally.

1. Physical values measurement
2. User interface for patients accessible via INK PC (see Appendix B)
3. Web medical interface for professionals

1) Information Society Technologies is one of the thematic priorities in the European Union Sixth Framework Programme for research and technological development.

Figure 1 shows medical devices (glucose continuous monitor, blood pressure monitor, heart rate monitor, digital scale and digital food scale). Data are collected by *Datalogger* which is directly connected to INK PC where data storing to database in *Central Node* should be handled. Beside this, the INK PC must serve patients who interact with the system via web interface and can check their measured physical data or a diet. Daily consumption is also entered by patients. The next endpoint which is not visible on the figure, provides access to patients clinical data via web browser – web medical interface for professionals/doctors.

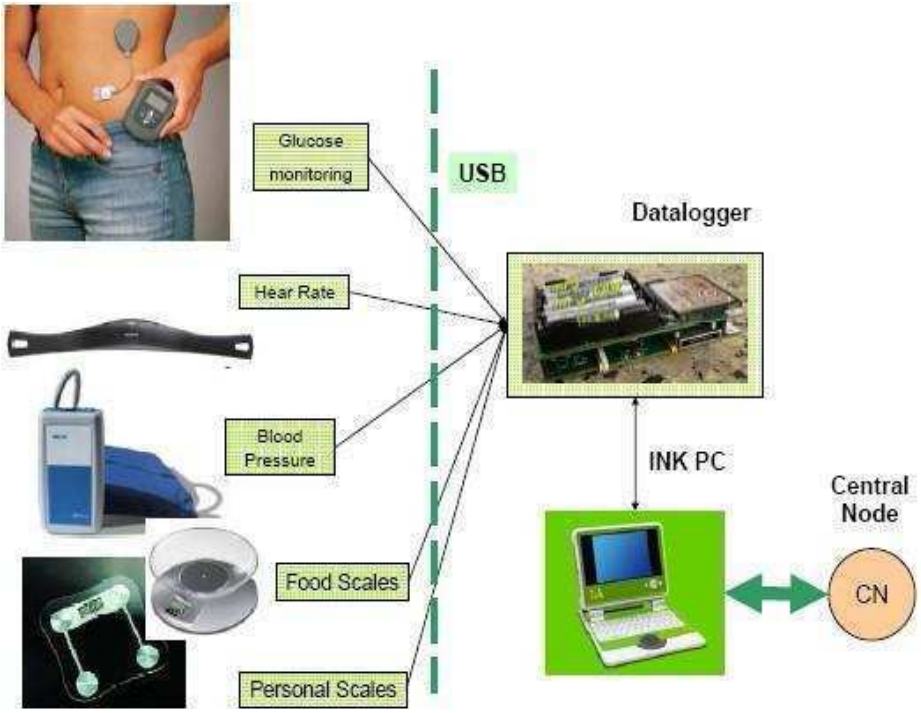


Figure 1: INK PC Devices

2.2 Bipometrics

The second project – Bipometrics [21] [17] - is devoted to predicting relapses of bipolar patients. Bipolar disorder, or manic depression in other words, belongs to category of mood disorders and refers to the cycling between high (mania) and low (depression) episodes/poles. The author of Bipometrics, MUDr. Filip Španiel, Ph.D. from Prague Psychiatric Center, have extended idea of his another similar project - Information Technology Aided Relaps Prevention in Schizophrenia (ITAREPS) [32]. The main aim of ITAREPS is quite the same because a doctor tries to estimate whether some relaps could occur for patient. Difference from Bipometrics is that the patients within ITAREPS suffer from schizophrenia. Prediction is made according to current patient's feelings and mood. The question is how to monitor the mood? The author prepared list of 10 items which must be matched to score by patients and theirs family member each week when the patient sends his/her answers via SMS. The score is from specific range (0 – 4):

- 0: No change or improvement
- 1: Mild worsening
- 2: Moderate worsening
- 3: Severe worsening
- 4: Extreme worsening

The items varied for each patient depending on his/her diagnose. According to total score the doctor can change medication, recommend hospitalization and so on.

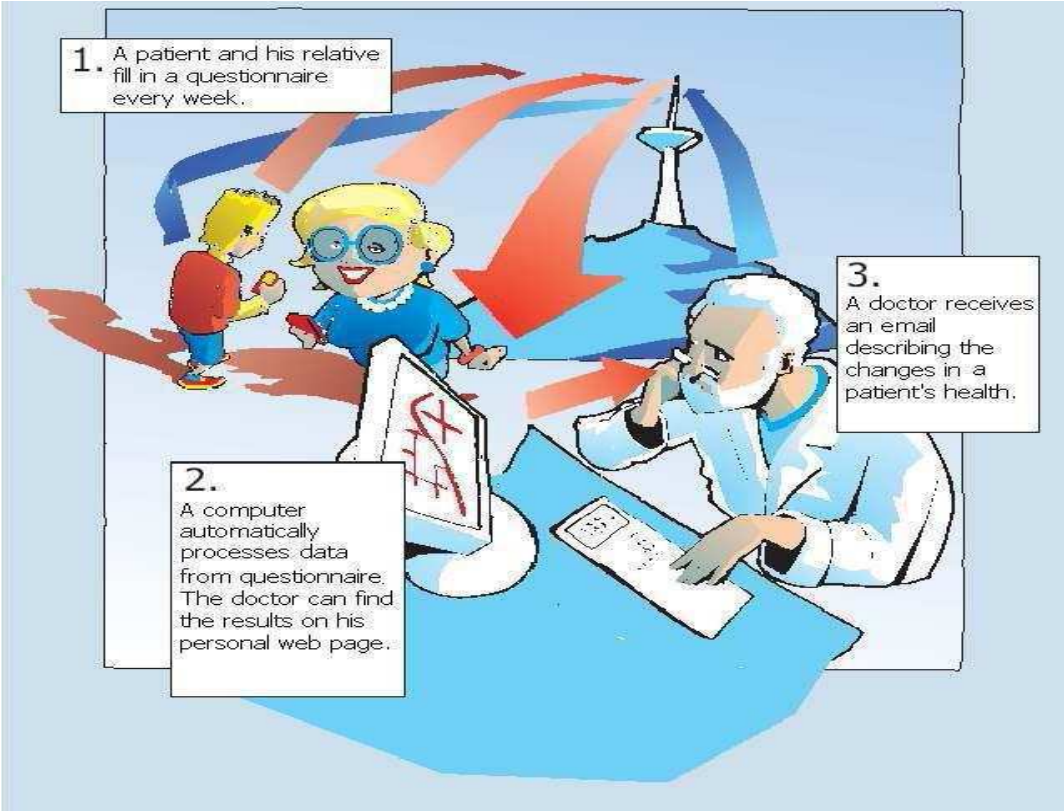


Figure 2: Schema of ITAREPS [32]

2.2.1 Bipometrics specifics

Bipolar disorder differs from schizophrenia and that is why new questionnaire have been developed. Patient, and the family member as well, is ought to match score for 18 items and the score must be from 0 to 9.

Questions for the subjective questionnaire:

1. I feel like I am able to do anything
2. I really feel well inside
3. It seems to me that I will not succeed at anything
4. I am depressed
5. I feel full of energy
6. I feel sped up
7. I have racing thoughts
8. I am overly active

9. I am restless
10. I am impulsive
11. My moods change a lot
12. I feel like people are out to get me
13. I feel like the world is against me
14. I feel irritated
15. I feel argumentative
16. I get distracted easily
17. I cannot concentrate well
18. I sleep well

Both poles of the mood are so dissimilar that it could be helpful to have record of the patient's activity. For the purpose of the Bipometrics study, actigrahy method was incorporated into the platform. The method is called *Actigraphy* and also can help with sleep phases detection. It is important because sleep contains different information from any other part of a day and changes in a sleep pattern are typical symptoms of depression or mania.

2.3 State of Art: e-Care projects

In this part the similar types of projects will be summarized with contribution of [20].

In the last years, all over the world different projects have been developed about integrated elderly care system, e-Care system, e-Disease Management and, generally speaking, telemedicine devices for promoting integrated care and improving the quality of life of elderly or help worth treatment. Telemedicine, tele-aid and tele-assistance use technological infrastructures within the e-Care projects with the main aim to help ensuring that vulnerable people can remain in their habitual environment and, thus, improving their quality of life, increasing their confidence and their safety. The objective is to provide a group of services which allow elderly, the disabled or people with health disorders to be connected to one or more specialized care centre through a focused contact centre that can provide an appropriate response to arising critical situations by mobilizing other human material sources.

The goals of the e-Care networks developed in the last years are the followings:

- Connecting the stakeholders in the socio-sanitary processes: citizens and their families, GPs, specialists, nurses, social assistants, and Health Trusts (public, private and non-profit)
- Sharing the information needed during the treatment, to provide a more complete and integrated care using ICT solutions

- To obtain a real integration of health and social services through an effective and integrated network of services
- Collecting citizen's health information through the network
- Connecting citizens with information on their own health by giving them an easy way to access the network

2.3.1 OLDES and Bipometrics challenges

The OLDES project tries to contribute to improve actual approach in elderly home care. The Bipometrics, on the other hand, wants to help to bipolar patients to decrease the number of bipolar disorder relapses. To summarize, the aim is the same for all of the e-Care projects-improving quality of vulnerable people's life.

In the scope of this work, the web interface for professionals/doctors is one of the most important part in the whole e-Care projects and provides significant information about elderly or patient. Because of 24h/7weeks availability, doctors can immediately react (change medicine or diet, inform family, order to hospitalize the patient etc.). Apart from displaying people conditions, the applications will also evaluate possible alarms and inform involved doctor or patient. In OLDES project this functionality means that web application and patient's application on INK PC are operating with the same set of data (physiologic data, food consumption) – measured data are processed by business logic in the web application.

Bipometrics tries to implement new solution based on ITEREPS project (see above). Currently the psychiatrists, involved into the Bipometrics, used Excel tables to store patient's mood. For actigraphy evaluation the Matlab application is used. Matlab is a high-level language and interactive environment that enables performing computationally intensive tasks. The biggest disadvantage of this approach is availability and complexity. Each doctor does not have to be familiar with Excel tables or even with Matlab. In addition these applications are not available on each computer but web browsers and the Internet connection almost are.

Doctors from 3rd Clinic of Medicine, 1st Faculty of Medicine, Charles University and General Teaching Hospital in Prague which participated in the OLDES Prague pilot project and treats diabetics have clinical application. This application is available only from the computers in the hospital, contains information about patient's condition collected within hospitalizations or visits. The OLDES Web Medical Interface will offer actual data in charts representation and it can be accessed from their office, home and also from cell phone.

In conclusion the both medical interfaces will provide easy-to-access application with actual data and patient's condition, allow immediately reaction to current state and also make possible to brows patient's history development.

Chapter 3

Technology and System Feasibility Study

One part of the assignment is to perform feasibility study (FS) which is described in this chapter. Generally speaking FS means to assess whether a project is advisable, with main focus on economic point of view. In this project we will leave aside economic analysis and concentrate more on technology feasibility. The study partly overrides analysis of the system. It covers project requirements, considers contribution to users against current solution, notices of the similar projects which have already existed and analyses advantages of selected technology and design. Each part is described separately for the both projects (ODLES, Bipometrics).

3.1 Universal and Inclusive Design

The both methodologies are used generally for designing buildings, facilities and other products used by users and its ideas can be easily implemented in computer applications to help design user interfaces.

Universal design and inclusive design are closely linked, although have few differences. The common characteristic is to increase a product accessibility to ensure the needs of the widest possible audience, irrespective of age or ability. Two major trends have driven the growth of these approaches - population ageing and the growing movement to integrate disabled people into mainstream society.

The main differencing factor exists in understanding of higher accessibility [20]. *Universal design* does this by focusing on the usability of the technology, making it more accessible for all potential users. On the other hand, *Inclusive design* does it by focusing on the capabilities of the intended users and building technologies that are accessible to them but not necessarily accessible or usable by other groups of users.

[10] says that *Universal design strives to create designs which are usable by the entire population, regardless of age, race, disability, gender, ability level, perceptions, values, and so on. However, this utopia ideal can never be fully realized and certain user groups will always require technology to be developed specifically for them. It means that these groups require their technology to be inclusively designed*. The products of *Inclusive design* which help bridge the gap between user capabilities and the demands of mainstream technology are called assistive technologies (although technologies designed to be assistive need not be inclusive in the social sense).

3.1.1 Considerations for Elderly

According to inclusive design exists a number of practical and social considerations for a target group – elderly in our case. List of practical considerations in design of user interface (UI) follows:

1. *Utility* – is about improvement of existing methods of performing current tasks and development of new functionalities which provide practical value to specific members of the users.
2. *Usability* – according to definition in the ISO 9241 standard [18] means: *"The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use"* and is related to, so-called, five 'E's of usability: effective, efficient, engaging, error tolerant, and easy to learn.
3. *Accessibility* – to have a lot of functions which achieve solution of concrete tasks is not the only thing the users needs. These functionalities have to be accessible for intended group. Thank to this, the design has to be also focused on the remove of access barriers. Its important to have in mind that elderly often suffer from wide range of disorders (motional, visual or cognitive).

No less significant considerations are the social ones:

1. *Aesthetics* – some people can object to importance of aesthetic consideration. It is true that UI with nice design and colours is useless without good functions, but the same is valid on conversely as well. Furthermore it is not easy to find compromise between two different group of users or even two persons.
2. *Trust* – everybody knows that lot of older people is sceptic to new technologies, because they have not had any experiences with them. Several focus group studies and experience shows that elderly are not averse to learning to use new technology when they can see benefit of it.

3.2 Requirements

3.2.1 OLDES

The goal is to create a web interface for two different groups – elderly (patients) and professionals (general practitioner (GP), physician, social worker, e-Care operator). It is quite clear that the first group has more requirements to the interface design. The most of the elderly have marginal experience with usage of ICT devices. Moreover they can suffer from some motoric or visual disorders which make usage of 'new-days' technologies harder for such people.

3.2.1.1 OLDES further details

The Prague pilot project of the OLDES project is focused on elderly with diabetes disease. This group of users has to interact with developed system. Interaction can be direct and indirect. Measurement of physiologic data is called indirect interaction because the measurement of glycemia, blood pressure and heart rate is performed periodically without any special user (patient) interaction.

On the other hand patient's diet and consumption writing or communication with GP or family demands usage of particular computer and system. Patients are responsible for filling their daily food consumption as accurate as possible. Also they are able to observe their current physical state, medications and so on.

One of the considerable functionality is to monitor alarms. An alarm can occur when the glycemia or blood pressure value step over the concrete threshold. Three different types of alarms should be distinguished – hypoglycemia, hyperglycemia and hypertension. The thresholds are different for each patient. Whenever an alarm will occur, the system must inform the patient and doctor/GP about it.

3.2.1.2 OLDES GUI design

Since measurement of physiologic data is out of the scope of this work, this part describes design of UI for elderly within OLDES project. Based on state of art in section 3.1, we conclude that Inclusive design is more appropriate method for designing user interface for elderly. The group of users is so specific that the design has to be oriented to theirs needs. The problem how to gain information about the needs and requirements can be solved in different ways. The focus group can answer a questionnaire which tries to scan theirs experiences with similar devices and notions about final product and its usage. Next approach is in form of an interview when the interviewer wants to get the same information but the pros is direct contact with the interviewee (questioner can observe his/her behaviour, feelings etc.). By contrast, this method is quite time-consuming mostly for the both sides.

Usability Lab

For our purpose, paper and software prototyping was chosen in order the users can directly interact with developed system and evaluators can monitor how the participants use the product. This type of observing and testing takes place in special laboratory – *Usability Lab* (see Appendix A). In the first session our participants used *paper prototyping* where they build their first design of each screen by papers, with names of the items etc., according to theirs feelings, experience and thoughts. According to [33]: *Paper prototypes or other mockups (i) clarify requirements and (ii) enable draft interaction designs and screen designs to be very rapidly simulated and tested*. In Figure 3 is shown part of prototype developed according to user requirements.

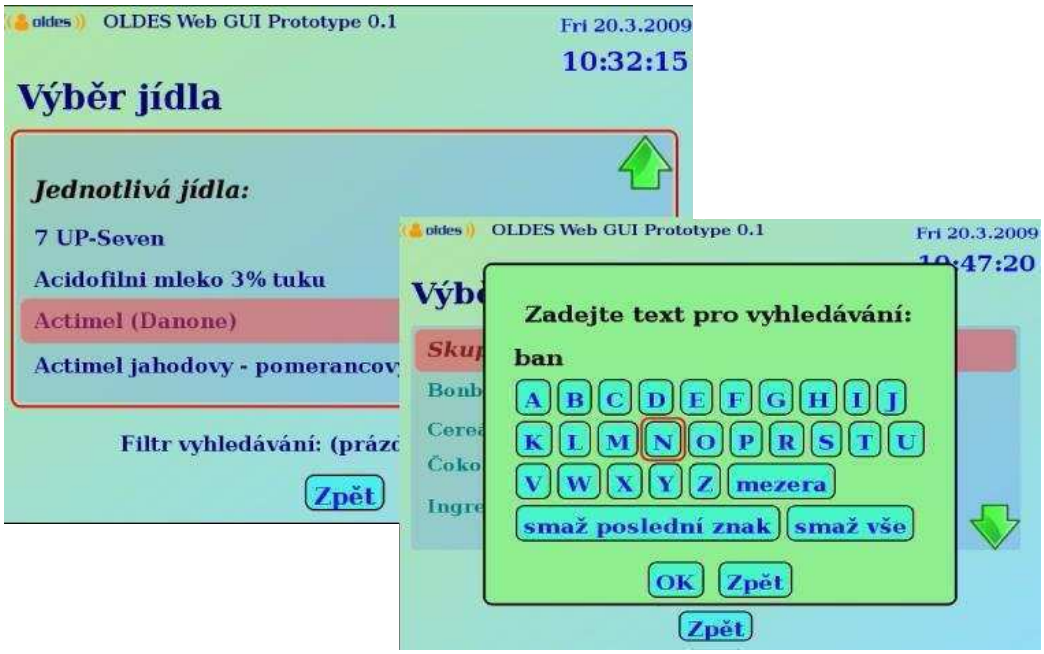


Figure 3: Prototype of GUI for elderly

Next advantage of the lab is possible usability testing because usability can be defined as the measure of the quality of a user's experience when interacting with a software product or a system (web site, a software application, mobile technology or any user operated device).

INK PC

After the conception of UI which has originated from paper prototyping, the UI input and output and used components (devices) must be figured out. One of OLDES goals is to grant a quality of assistance services reducing the total cost of ownership of the system. OLDES system must also reduce the cost of components. The node of the system which is assigned to patients is located at the old person's home and performs several functions such as:

- hub for medical devices (measurement of physiologic data)
- interface for accessing OLDES services such as information or entertainment channels and VOIP.

Several projects in telemedicine or tele-care does not care much about the node at old person's home, using a commercial PC, maybe buying just needed components. OLDES, on the contrary, intends to reduce costs but also to grant a limited impact into old person's homes. It is developing together with INK's existing low cost PC project, an appropriate PC with small dimensions, free software and not-movable mechanical parts such as hard-disk. See Appendix B for more information.

Basically INK PC is classical laptop with only necessary equipment to decrease cost of the device. On this account, implementation and used technology for UI has to count with some limitations.

The implementation of this part of OLDES is not included in this paper because it was developed by other participants and used different technologies. The application is described by a set of XML files: pages, skins and page handlers. The GUI executable serves as a viewer for such application. When run from the command line, it takes the path of the default page as an argument. The pages (roughly similar to ASP.NET .aspx pages or HTML pages) may contain various controls which can be linked to predefined actions or used as Hyperlinks to other pages.

3.2.1.3 OLDES web medical interface

After the study of the part where the elderly is the main focus group, the web medical interface for professionals will be discussed below. Professionals, in the OLDES sense, are people who can be involved into patient treatment or elderly assistance - GP, nurses, dietician, social workers and elderly family. From this wide range of users (with different age and qualification) follows that design of the web medical interface should apply the *Universal paradigm*. It must be easy-to-use and friendly as much as possible. In the step of the project the UI for GP should be prepared because they will be the majority of the users. The best way how to design a look of such web application comes from an experience. All of potential users are more or less familiar with web browsing and so the UI should look in common way – all needed features are easy-to-find and easy-to-understand.

Functional requirements

The concrete functional requirements were gathered directly from doctors (from 3rd Clinic of Medicine, 1st Faculty of Medicine, Charles University and General Teaching Hospital in Prague) who are involved in the project. We had arranged few meetings and discussed doctors requirements and possible solutions. The emphasis was put on visualization of physiologic data which must be displayed via the web application in tables and/or charts.

In concrete:

- glycemia (1 per day)
- pulse (continuously)
- blood pressure (3 per day)
- weight (1 per day)
- food consumption per day

The application must inform GPs about alarm, in the same manner as GUI on INK PC and in addition via e-mail.

Non-functional requirements

Professional operators (carers) access the system using a web portal providing an authentication and authorization system. Once a user logs into the system (authentication), he can access a set of services linked to his personal or group role (authorization).

The application must fulfil security issues, since the portal will be available for all users with Internet connection and the carers have to send confidential data (e.g. password). However, the Internet is an insecure network and that is why we must solve security of our web application.

One of the biggest technological demand is to use only free projects which are open-source at best. Selected technology (framework) should be easy to extend new functionalities.

3.2.2 Bipometrics

The Bipometrics project has a lot of similarities with OLDES Web Medical Interface. The goal is the same – implementing an application which will be accessible from a web browser and enables doctors to monitor condition of theirs patients. Several conclusions and solutions could be gained from the previous discussed project. Nowadays the potential users are “only“ doctors but nobody can guarantee that the group of people which interacts with the application will not extend in the future. Therefore the *Universal design* methodology rules are suitable for UI designing.

It is obvious that similar type of application has similar non-functional requirements which have been already described in the previous part. The only difference is in details of the

functionality because the focus group is quite different – patients which suffer from mental illness (manic depression).

3.2.2.1 Functional requirements

Bipometrics project description says that the main aim is to predict a relapse of a patient who suffers from bipolar disorder. This prediction can be done as a consequence of patient's mood and activity monitoring. Information about patient's condition is obtained in two ways. First – *the mood* – each patient (and his/her family member) has to answer, or match the score, 18 questions via SMS. The doctors requires easy reading of the answers and have decided to show that score in charts. Second – *activity* – should also provide output in charts of actigraphy. The application must solve gained data inserting or uploading.

The next common part with OLDES is informing doctors about alarms which can occur after the changing the patient's mood. The alarm must be displayed in the application and sent to doctors via email.

3.3 Suitable technology

The aim of this part is to find and discuss suitable technology for the web interface implementation. It is necessary to consider assignment, functional and non-functional requirements when choosing the technology. Concrete technology should be easy-to-use, easy-to-learn, provide solution of security tasks, low-cost or even to be provided on free basis.

There exist several types of software frameworks in these days that are designed to support and simplify web applications development – also called web application frameworks. Thanks to these frameworks which are based on a wide range of programming languages, development of such application is easier and transparent. Developers do not need to take care of common development tasks (architectural issues) such as session handling, input validation etc.

The assignment of this thesis implementation demands was carried out in Java programming language. Java is a good choice of several reasons. Java is free software product under GNU General Public License [6] and this is one of the non-functional requirements of the developed interfaces. Java with its Enterprise Edition (Java EE) [26] is a powerful tool for implementing web based information systems based on Model View Controller pattern (MVC). Many web application frameworks is implemented in Java and based on MVC. The MVC will be discussed later in the section 4.2. Java EE is superstructure of the Standard Edition (Java SE) [27] which provides many useful technologies and application programming interfaces (APIs) which helps and supports implementation of demanded application like:

- *security* – includes implementation of common-used security algorithms and helps with authentication and authorization
- *XML* – provides tools for XML processing

- *JDBC* – allows DB connecting and data manipulating
- *Logging* – produces log reports
- *Exception handling* – supports handling of special conditions that change the normal flow of execution

Moreover, a lot of free third-parties libraries in Java can be helpful within discussed web applications. Very common method to simplify persistent data manipulating is an object-relation mapping and the most favourite framework is also written in Java programming language – Hibernate. The next technology which will be used in the system are web services and here we can find worthy frameworks based on Java again.

3.4 Summary

In this study the similar projects, OLDES and Bipometrics details, in connection with UI design were discussed. The end of the chapter was engaged in suitable technology for implementation. Thanks to these circumstances the conclusion has to be proclaimed, whether the projects are advisable, worth doing and why.

New web interface will add practical and easy-to-use tool which support and assist with patients treatment and care for vulnerable people. GPs will get better overview about actual patient's condition and can react to any changes in time. From the study results added value of this type of interface clearly in comparison to Excel tables or Matlab application which are used currently.

Chosen technology - Java EE - is a guarantee of as easy design and implementation as conceivable. Also it makes a future extension simply possible. Important point is that the developer is experienced in analogous web applications developing in Java. Since many free third-parties libraries, also written in Java, are available on the SW market, the very powerful system can be implemented and fulfilled the big requirement in using only free products.

To summarize, designed projects are advisable, feasible and worth to create. Furthermore, various new features will be able to be added and so the systems can be used for a long time in production.

Chapter 4

Design of Web Medical Portal

This chapter includes design of the implementation – architecture of the system. From the design and also implementation point of view, both developed applications are very similar. This fact allows description of the common problems at one place because the outcome will be same for both web portals. Of course that some differences have occurred. For better transparency, in case of any dissimilarity, the problem is discussed separately.

4.1 Physical architecture

Physical architecture means physical separation between different types of devices or hosts. The type of application is given in the assignment that is why we will discuss web application architecture in the next few lines. Every web application is comprised from tiers. The most common, well-known and well-tried is three-tiered architecture. In Figure 4 this type of application is illustrated.

The first tier is every client's web browser which serves to interaction between client/user and the application. Every client's request is processed on application server from second tier. A Java container is typical application server where server-side implementation runs. In our concrete case the container is Apache Tomcat.

This web server is one of the most favourite servlet container. Tomcat implements the Java Servlet the JavaServer Pages (JSP) specification from Sun Microsystems. It is suitable for ours web portal because it is free and developed in an open environment and released under the Apache Software License [1]. Tomcat allows solving security issues like digest access authentication and Hypertext Transfer Protocol Secure (HTTPS). It can run on varied number of operation systems (Windows, Linux, BSD, Solaris etc.) and provides easy configuration by administration web console. What is important on Java EE applications like ours is that it is independent on a container. It can run on other containers as JBoss or GlassFish.

The third tier contains database server where all data are stored. In the application is used PostgreSQL [16] database server.

4.1.1 DB Server

Selection of the right database server stands before the same problems as choosing of the programming language or the suitable framework for the development – software market offers many types of implementations. The reasons for choosing particular PostgreSQL solution are enumerated below.

Perhaps the most robust and stable solution is Oracle [14]. The drawbacks of Oracle are big complexity which is not necessary for this project, hardness of the configuration and commercial license. The next very spread server is MS-SQL [9]. Again it can be considered as very powerful tool compared to the previous one. Unfortunately it has the same drawbacks – commercial license, huge complexity and in addition the dependency on the Microsoft platform. To fulfil the requirement on usage of only free components we should focused on two main representatives in this field. The first is MySQL [29] which is the most common in the web applications based on MySQL – PHP – Apache as web server. The main advantage is its performance – speed especially. On the other hand, this aspect was in contrary to a long lack of foreign keys or transactions. Since the version 5.0 is most of these absences adjusted.

PostgresSQL has sufficient performance, amount of functionality and it is stable. It is suitable for complex as well as for more simple applications. It has good maintenance and user administration. Matter of fact is graphical administrative tool as PGAdmin and free availability. In comparison to MySQL, PostgreSQL has higher limits (like max. DB/table/row size etc.) and bigger or longer support of some functionalities – transactions, views, foreign keys. Due to these reasons, the OLDES and Bipometrics teams have decided for PostgreSQL DB server.

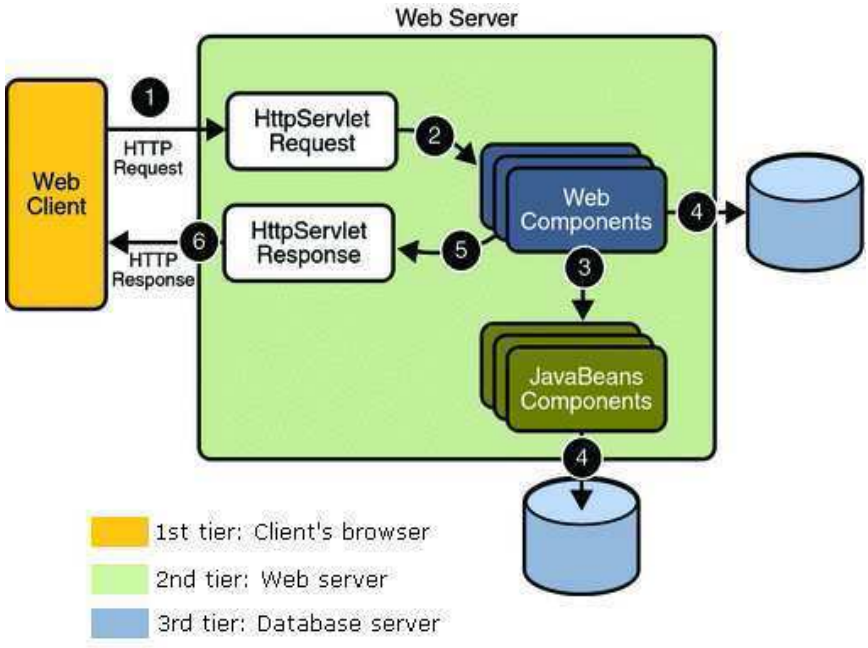


Figure 4: Java Web Application Request Handling [30]

4.2 Logical architecture

Now when we introduced physical part of application architecture we can focus on architecture inside the server-side implementation which is called logical architecture (in this document). Each web application must solve request handling. Common Java web application request handling is also described in Figure 4. According to Java EE Tutorial [30]: *"Web components are either Java servlets, JSP pages, or web service endpoints. The client sends an HTTP request to the web server. A web server that implements Java Servlet and JavaServer Pages technology converts the request into an HTTPServletRequest object. This object is delivered to a web component, which can interact with JavaBeans components or a database to generate dynamic content. The web component can then generate an HTTPServletResponse or it can pass the request to another web component. Eventually a web component generates a HTTPServletResponse object. The web server converts this object to an HTTP response and returns it to the client.*"

Applications tend to mix data access code, business logic code and presentation code. In software architecture several "templates" exist - known as design patterns - which solve common and often problems in software developing. One of widespread pattern within software developers is Model-View-Controller (MVC) which solved the problem of "code-mixing".

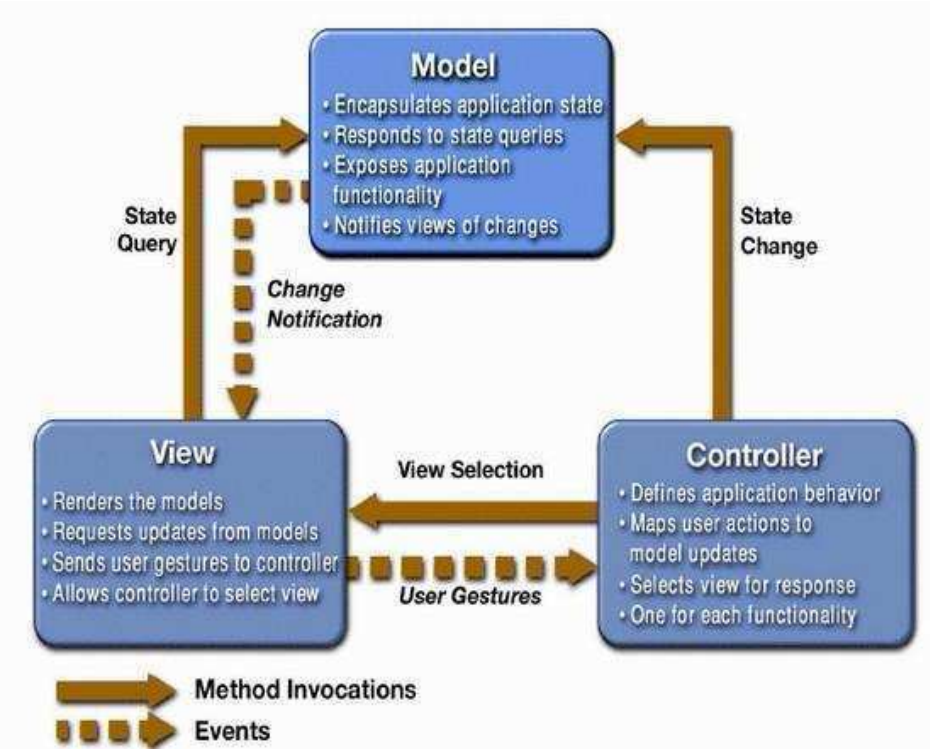


Figure 5: Model-View-Controller pattern [25]

Figure 5 shows simple idea of MVC and flow of events and method invocations. The figure also describes responsibilities of each part. The template decoupling leads to model re-use and easier support of new client types, because it is not restricted to web applications (or web browsers). In Java EE typically Controller = Java Servlet, View = JavaServer Pages (JSP) page, Model = JavaBean component.

4.2.1 Apache Struts

The applications are not implemented by "standard" Java technologies but use special framework which is based on these technologies. Reason for using the framework is to simplify developing MVC application. There exist a lot of similar types of frameworks which offer much the same things. In the OLDES web medical interface is used Apache Struts 1 and in the Bipometrics web medical interface Apache Struts 2 in contrary.

The following definition is taken from [37]: *"Apache Struts is an open-source web application framework for developing Java EE web applications. It uses and extends the Java Servlet API to encourage developers to adopt a model-view-controller (MVC) architecture."* In comparison with other similar frameworks it is easier to use and less complicated. Again the complexity of the developed application with complexity of a framework has to be considered. In my opinion the Stuts is good choice for discussed web portals. We do not need any of complicated business logic where Enterprise JavaBeans (EJB) can be more useful. The same viewpoint can be applicated in development of user interface. JavaServer Faces (JSF) which intend to simplify this development but the approach is quite different from MVC frameworks because this technology is component-based. Furthermore, Struts can be easy extended by any other technologies which would be demanded.

As we said earlier, Struts provides API to easy implementation of MVC design pattern. The Struts Controller acts as a bridge between the application's Model and the web View. When a request is received, the Controller invokes an Action class. The Action class consults with the Model to examine or update the application's state. Furthermore Struts supports easy internationalization (i18n) or Tiles to help with common GUI building.

4.2.1.1 Struts 1 vs. Struts 2

This part describes differences between the both versions of the framework. Since the OLDES application is implemented on the bases of the Struts 1 against this, Bipometrics portal uses Struts 2. The reason for "changing" the framework was to even more simplify the development and learn this easy usage of the framework. Some new features such as file uploading are required and Struts 2 provides this functionalities.

The Struts 2 has evolved from the first version of Struts and WebWork [13] and as [5] says: *"This new version of Struts is simpler to use and closer to how Struts was always meant to be."* The main advantages of the second version are: support of powerful expression language – OGNL [12], simple request filtering by interceptors, common functionalities like file uploading, workflow etc. are included and so on. To more information about comparison of the both versions see Appendix C.

For better understanding of request processing within Struts framework, the below lines compare life-cycle of a request in Struts 1 (see Figure 6) and Struts 2 (see Figure 7).

A request comes to the Servlet container where is processed in filter chain and reach `ActionServlet` which is default Struts controller. This servlet populates value to form-bean (`ActionForm`) and validate it. The form-bean holds data from user's inputs. Then the `ActionServlet` calls action class according to actions mapping in *struts-config.xml* file where the business processing begins (Model). As a next step action class provides the key to the `ActionServlet` to determine the result (from *struts-config.xml* again) and the servlet displays the result (View).

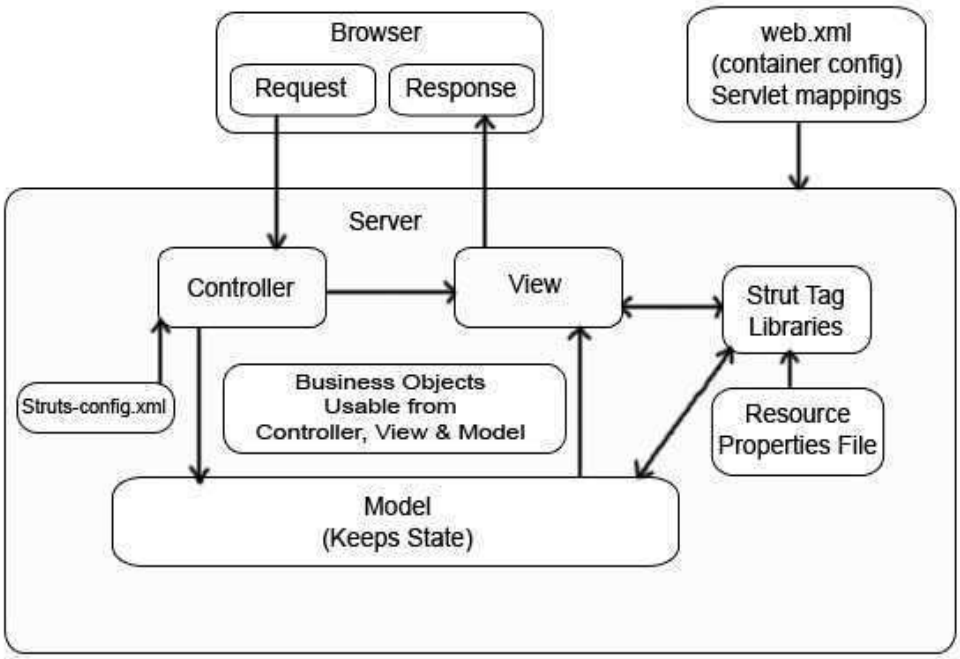


Figure 6: Struts 1 Request handling

In Figure 7, an initial request goes to the Servlet container which is passed again through a standard filter chain (same as in Struts 1 above). Next, the `FilterDispatcher` delegates control to the `ActionProxy`. The `ActionProxy` consults configuration file – *struts.xml*. After that the `ActionProxy` creates an `ActionInvocation` which is responsible for the command pattern implementation. This includes invoking any Interceptors (the before clause) in advance of invoking the `Action` itself. Once the `Action` returns, the `ActionInvocation` is responsible for looking up the proper result associated with the Action result code mapped in struts.xml. The result is then executed which often (but not always, as is the case for Action Chaining) involves a template written in JSP or FreeMarker to be rendered. While rendering, the templates can use the Struts Tags provided by the framework. Interceptors are executed again (in reverse order, calling the after clause). Finally, the response returns through the filters configured in the *web.xml*.

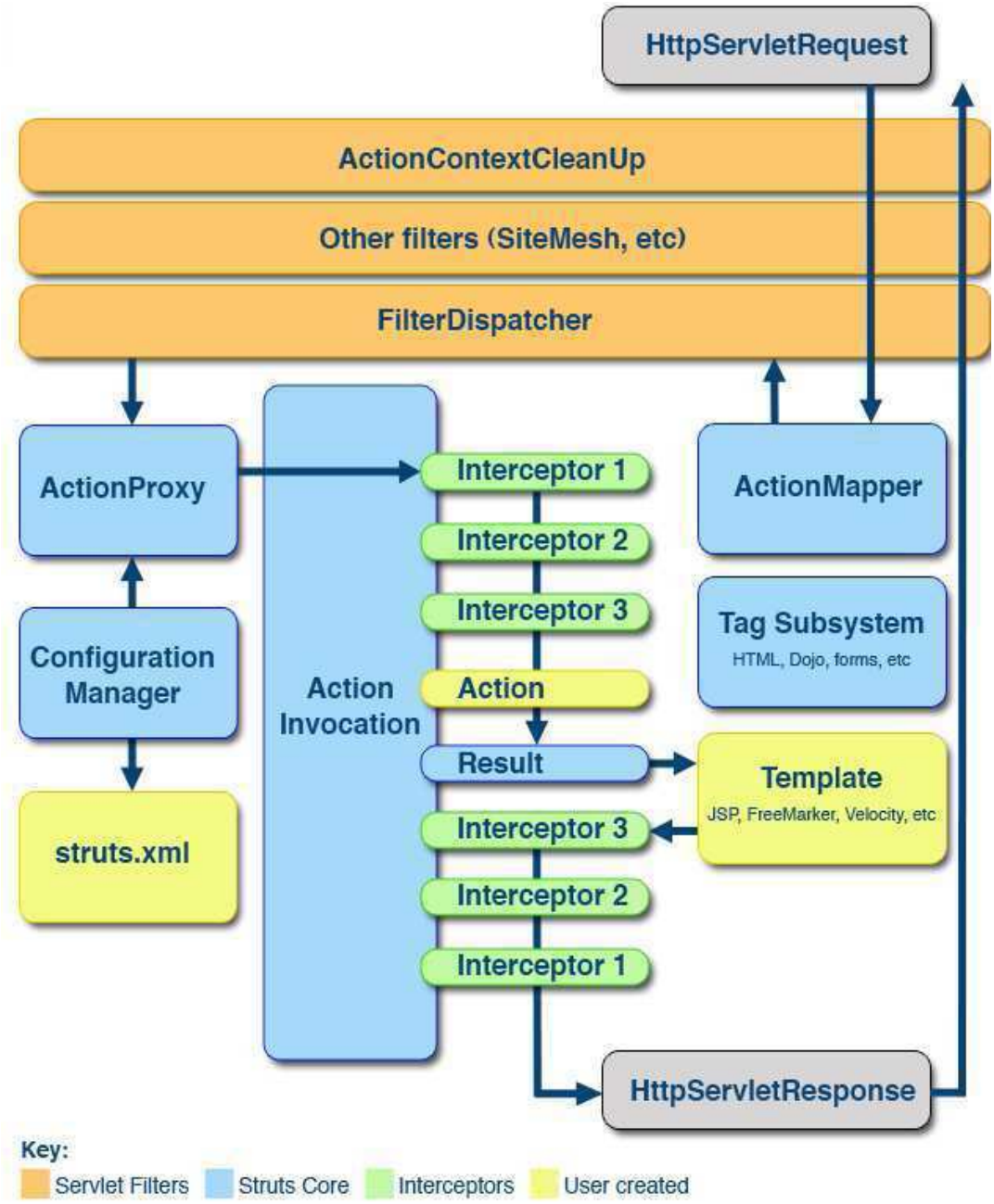


Figure 7: Struts 2 Architecture [2]

4.3 Common tasks

We have already discussed the high-level architecture of web applications. This section will describe lower level with focus on common tasks design.

4.3.1 Data source access

One of the most common tasks during a request processing is to obtain/manipulate data from DB. Implementation should be independent on the View or Controller components and on data source (database typically) from the other side. Persistent data are stored in PostgreSQL database but the data source can be changed in the future. From this reason special interface is implemented there - *Data Access Object (DAO)* - which encapsulates all access to the data source.

4.3.1.1 DAO

The DAO is design pattern which manages the connection with the data source to obtain and store data.

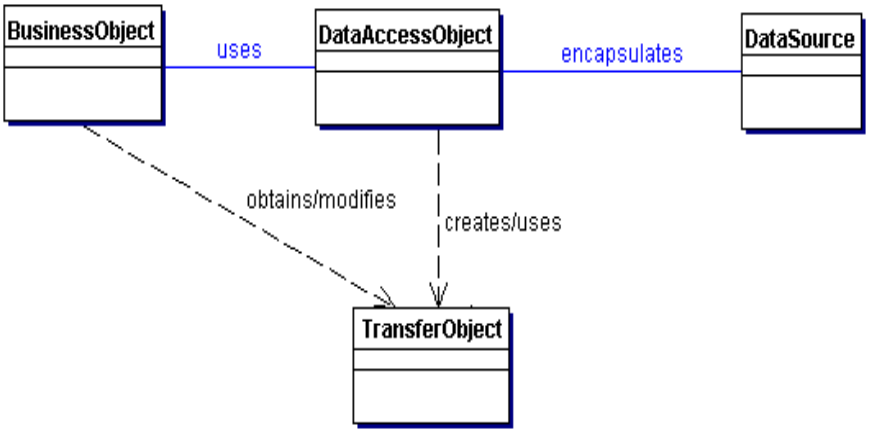


Figure 8: Relationships for the DAO pattern [23]

Figure 8 shows class diagram with relationships for the DAO pattern. Business object contains implementation of some logic which requires access to data source – Action class in our case. Underlying data access implementation which enables transparent access to the data source is included in `DataAccesObject`. `DataSource` represents a data source implementation – PostgreSQL in our case. The last object – `TransferObject` – carries obtained data.

In the both implemented applications, there will be concrete DAO instance gained from factory object. This factory decides which DAO implementation in a relation to used data source (Oracle, DB2, XML etc.) should be used. It can seem useless but we should consider possible future changes which can lead to provide more or other data sources. See sequence diagram of interactions between `BusinesObject`, DAO and data source (Figure 9).

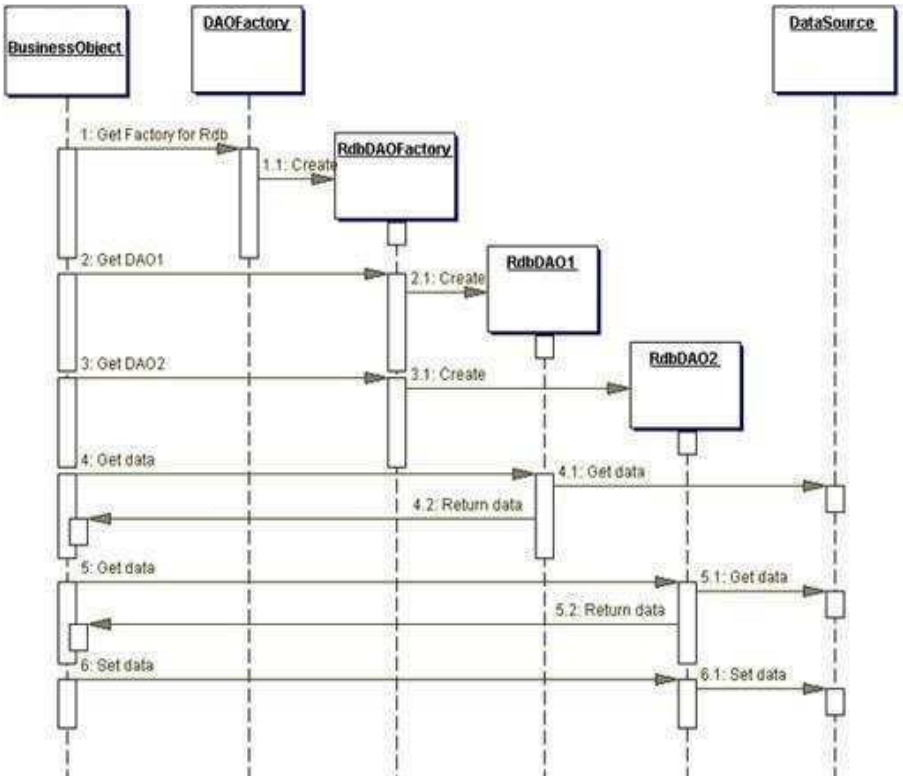


Figure 9: Factory for Data Access Objects using Abstract Factory sequence diagram [23]

4.3.1.2 Data Transfer Object

In Figure 8 a class called `TransferObject` is drawn which represents *Data Transfer Object (DTO)* and is used as a data carrier. DAO may use a DTO to return data to the client. The DAO may also receive the data from the client in a DTO to update the data in the data source. Implementation of some DTO is according to JavaBean conventions. Each bean has private property fields which are accessible through public getters and setters.

4.3.1.3 DB connection pooling

Database connection pooling is technique to reuse connection to DB. It is a type of connection cache which enhance the performance of executing commands on a database.

Each DB connection is placed back to pool and can be use over again. It means that we do not have to establish new connection whenever we need it. Connection pooling cuts down on the amount of time a user must wait to establish a connection to the database and reduces the cost of connection creation.

This method is common for web or enterprise applications and is supported by majority of application servers. Apache Tomcat Servlet container used in discussed applications is not an exception. Pooling configuration on Tomcat is quite easy - just to add few lines to configuration file of Tomcat.

4.3.2 JDBC vs. Hibernate

The problem with data source access was solved by DAO. The next task is how to query and update database from DAO. Java SE provides the *Java Database Connectivity (JDBC)* API which allows user to connect to DB and manipulate its data via SQL statement. After the statement execution a result set is returned and represented by `ResultSet` class. `ResulSet` is special collection of selected rows from DB. The selected data are commonly carried by DTOs. The same approach is used in the OLDES application.

Working with JDBC is easy but also brings some drawbacks. The main problem is that there is a mismatch between how data is represented in objects versus relational database. In OLDES is used simple Java bean as transfer object and common task is to map relational model to object model (Java bean). Other disadvantage can be SQL which is used in JDBC. Each DB server implements the SQL standard in different way and so it is possible that in new DAO for new data source will require new SQL statements. Other disadvantage is responsibility to handle JDBC result set and convert it to Java objects which is annoying for developers and can be place for pointless errors.

One of the common and powerful solutions of listed problems is an *Object-Relational Mapping (ORM)*. The most spread solution of ORM is well-known *Hibernate* [19] framework. To facilitate communication with DB objects, Hibernate was included in the Bipometrics application architecture. Hibernate provides table-to-object/object-to-table mapping and it is described in XML files. The result from DB select is returned directly in form of Java objects, thanks to ORM. To suppress differences between each DB implementation of SQL, Hibernate offers query language *Hibernate Query Language (HQL)* which makes a business logic independent on DB server. Using DAO within business logic has no changes and DAO is obtained in the same way as in Figure 9. The only change is that DAOs use Hibernate implementation to query and update DB data.

4.3.3 Authorization and Authentication

The portals support different types of users (roles) which are then used in authorization. To distinguish users, each user has to be authenticated – logged in the application. The portals offer traditional way of user log in – web form which is available on the main page. Person who wants to use the application is prompt to write his/her user name and password.

Each user is casted at least in one role but also can be casted in more than one role. Three types of roles are now supported – *anonymous, user, admin*.

Anonymous is each user who is not logged in the application and has rights only for displaying page with log in form and performing log in process. In case that a user wants to perform some action for which does not have rights, the error page with message about authorization failing is displayed. The *user* role is commonly used for doctor's access to the application. Authenticated doctor can monitor current patient's condition. The last role – *admin* – handles administration of the application. Administrator is responsible for patients management (in OLDES) and/or data uploading (in Bipometrics).

4.3.3.1 JAAS

The idea of authentication and authorization described above is quite easy. The question is how to implement these processes? The answer is – use pure *Java security solution; Java Authentication and Authorization Service (JAAS)* [28] is a security framework for user-centric security. This API is part of Java SE and so any other libraries are not required. Big advantage is that the JAAS can be used for both purposes [24]:

- for *authentication* of users to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet; and
- for *authorization* of users to ensure they have the access control rights (permissions) required to do the actions performed

JAAS grants a user permission for an action. To distinguish whether the user is permitted to perform concrete action will be made by URL permission. Management of performing concrete action by user is handled by URL permission method. The permissions are granted to roles in *jaas.policy* configuration file; e.g. an user cast in role *admin* can perform actions with *anonym* and *admin* permissions, role *user* can perform actions with *anonym* and basic permissions in contrary.

In Appendix D takes place sequence diagram which shows interaction among JAAS API within login action. After the successful login, the user roles are stored in the session object. After that when user invokes new action, JAAS implementation performs permission checking – action name with permission prefix against collection of roles from the session.

4.3.4 Web services

As we already have known OLDES requires storing physiological data measured by a device connected to INK PC. The values should be stored into DB common for the web portal. The value to store must be process on the server to handle potential alarm. One of the easiest approaches which were designed within OLDES application is to use web services. The services are deployed on Tomcat server and are available for application on INK PC to perform data storing and checking the alarms. See Figure 10 where in the (1) step the *SOAP Request* is sent from INK to the server where concrete service processes data and (2) inserts new row into DB or checks current data (alarms) in DB. The last step (3) is *SOAP Response* from service send back to INK. This approach allows same flow from any other device or application not only from INK.

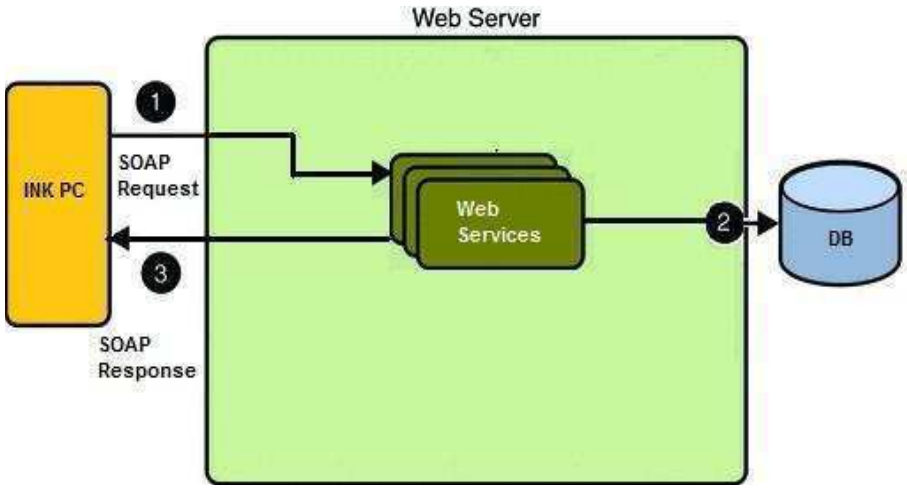


Figure 10: INK interaction with medical web server

4.3.5 Alarm info sending

The both applications have to be able to inform doctors about an alarm via e-mail message. Java provides a mailing API which makes the solution of e-mail sending easier.

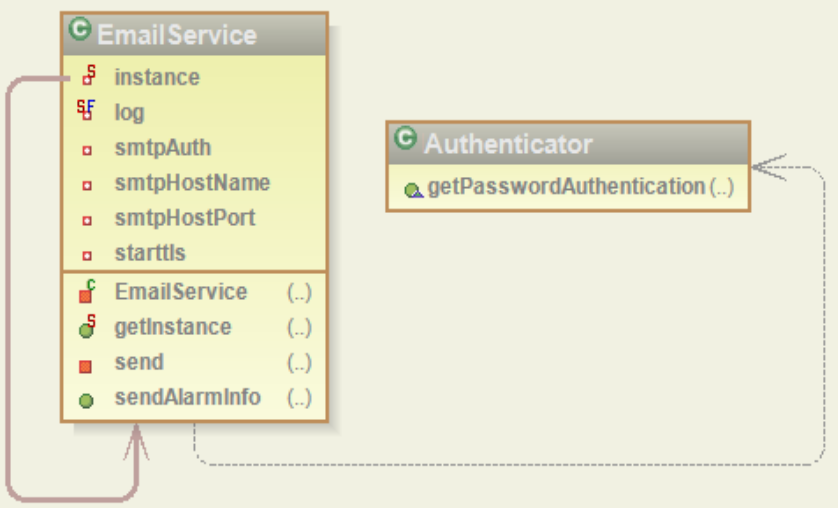


Figure 11: Class diagram of the `EmailService`

Design of this service is shown in Figure 11. The `EmailService` uses an `Authenticator` class which reads values (like host, account name) from property file.

4.3.6 Exception handling

One of the most neglected part of each software application (does not depend whether web, desktop or any other type of application is meant) is exception handling.

The Java forces developers to handle one type of exceptions (checked exceptions – see below). Lots of programmers (me as well) time after time handle exceptions only with error printing on console or even source code comment! Obviously it is fast and easy but only until first debugging attempt or bug occurrence when the application is deployed on real machine and it is available to use for clients.

How we can find with this approach a location of an error? How can we let clients know about the reason of the error? How can any client post the error to our help-desk centre and how can be the application recovered? These are only few questions and possible problems which can occur as soon as the exception handling in our application is marginalized. From my own and quite number of developers experience I know that efficacious exception handling is not time consuming in reality - if we imagine all problems which can come around in case of approach which we have discussed above.

4.3.6.1 Java exceptions

Java programming language offers quite wide API for exception handling. According to [31] exist three kinds of exceptions (*checked exception*, *error*, *runtime exception*) which are divided in two groups (*checked exceptions*, *unchecked exceptions*).

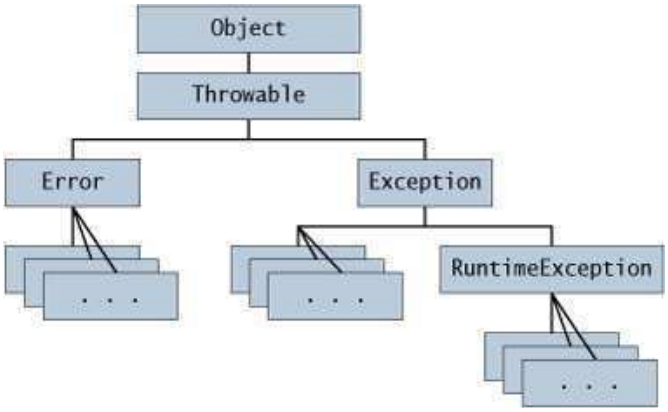


Figure 12: Java exception API [31]

Checked exceptions

This group of exceptions forces developers to handle and recover whether an error has occurred. This exception should not be cause of the application termination. Developer should inform the users about the problem. Typical checked exception example is when user type wrong input file path. In this case developer must give notice to user and he/she can correct the input string. Application is still running and can continue in normal flow after the correction by the user.

This group contains the first kind of exception – checked exception. In concrete implementation checked exception is each exception which is inherited from `java.lang.Exception` or itself.

Unchecked exceptions

Unchecked exceptions mean error or runtime exception. From an error exception the application usually can not be recovered. The exceptional conditions for error type are external to the application (e.g. is unable to read a file because of a hardware or system malfunction).

The last kind - runtime exception - is usually unrecoverable too. The exceptional conditions for error type are internal to the application (e.g. logic errors or improper use of an API). Well-known runtime exception is `NullPointerException`.

Unchecked exceptions are `java.lang.Error`, `java.lang.RuntimeException` and their subclasses.

4.3.6.2 Struts exception mapping

Two types of errors can occur in principle:

- an error which is caused by user's wrong input
- an internal error within the application (e.g. DB is not running, logic errors etc.)

Apache Struts web framework provides configurable and easy to use error handling – exception mappings (similar in both versions). Exception mappings are a powerful feature for dealing with an `Action` class that throws an `Exception`. The core idea is that an Exception thrown during the Action method can be automatically caught and mapped to a predefined result. This declarative strategy is especially useful for frameworks like Hibernate that throw `RuntimeExceptions`.

Configuration of each exception mapping is managed in *struts.xml/struts-config.xml* configuration file.

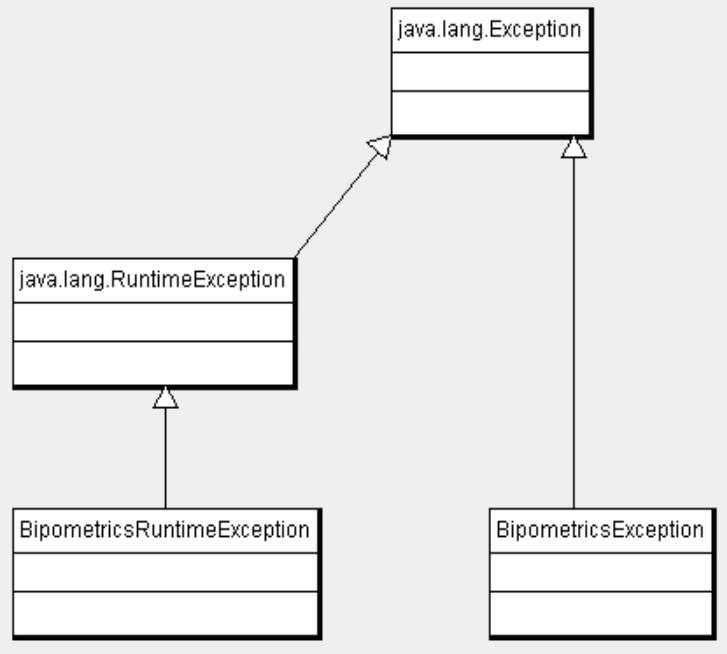


Figure 13: Bipometrics exception division

Chapter 5

Implementation

The goal of this chapter is to describe implementation of the web medical portals with focus on specific and interesting solutions. Many required tasks are implemented on the similar basis for OLDES and Bipometrics projects.

5.1 DB Scheme description

5.1.1 Bipometrics

The following section describes schema of the database with emphasis on the most important parts.

5.1.1.1 Users

Figure 14 shows part of DB scheme where relation of user, role, doctor and patient is displayed.

- *person* – carries common data for any person stored in DB (*doctor, friend, patient*)
- *patient* – is related to concrete person and is extended by *anamnesis* and *ident* (personal identification number) column. The next relation is to *friend* table, where typically a family member is stored.
- *friend* – does not have any special columns (all attributes are extended from *person*)
- *doctor* – is next extension of person. Contains person *id* and *specialization*.
- *doctor_patient* - supports possibility that one doctor can have more patients and one patient can be treated by more doctors.
- *web_user* – the last extension of person. Users from the table can be logged in the application. Two important columns are *username* and *password* (serves to user identification within authentication process). This table has composite primary key from *id* and *username* columns (an *username* can be used only once). *role* – contains names of roles which are available. Roles serve to the authorization process.
- *user_role* – provides possibility of casting an user to 1 or more roles. The table is related to a *web_user* and *role* table.

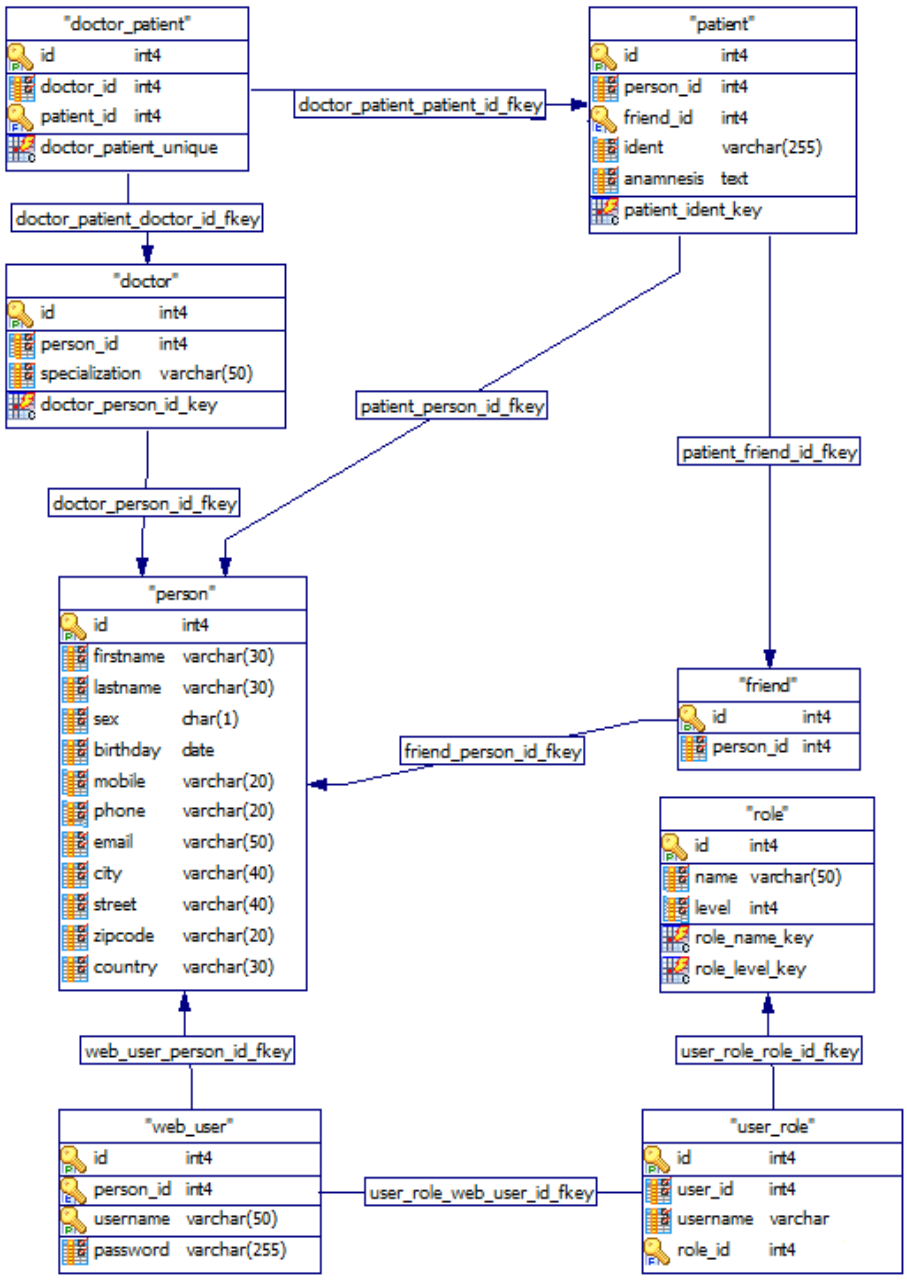


Figure 14: DB Schema - user, role, patients

5.1.1.2 Questionnaires

The core of Bipometrics web application is to display score of patient's and family member's questionnaires. See Figure 15 to find out how the questionnaires are stored in DB.

- *question* – stores set of questions which can be used in the questionnaire. Column *order* serves to ordering questions in an output for user.
- *questionnaire* - is related to *question* and *patient* to identify which question and for who the score was matched. Score is stored in the *value* column and date of the answer is in the column *date*.
- *friend_questionnaire* – has the same structure as *questionnaire* but is related to a *friend* in contrary. Stores score of family member's answers.

Dividing questionnaires in two tables has a reason – reduce size of one table and the schema is also more readable.

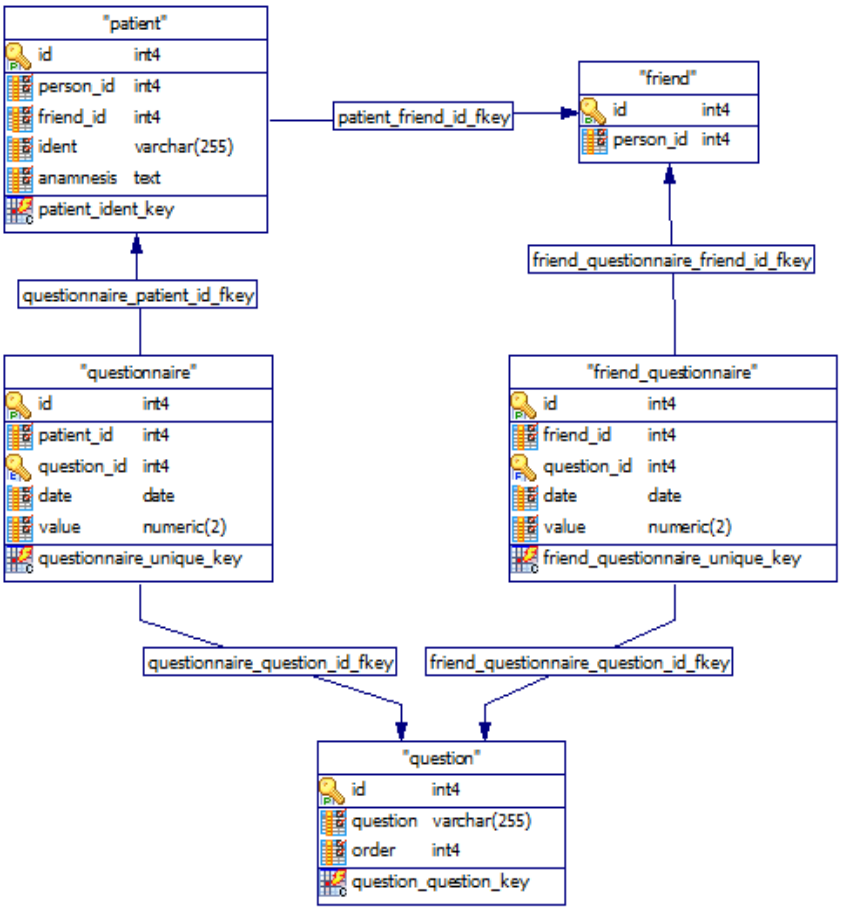


Figure 15: DB Schema - questionnaires

5.1.1.3 Patient's activity

Patient's activity is described by actigraphy (movement activity of patient), sleep phases and NAP (short period of sleep during the day) phases. In Figure 16 is shown how data is stored for these objects.

- *actigraphy* – stores data of *start* and *end* time of the activity measurement.
- *actigraphy_data* – is related to one *actigrahy* (activity measurement) and contains measured value of activity.
- *nap* – stores time range of a NAP phase.
- *sleep* – stores time range of s sleep phase.

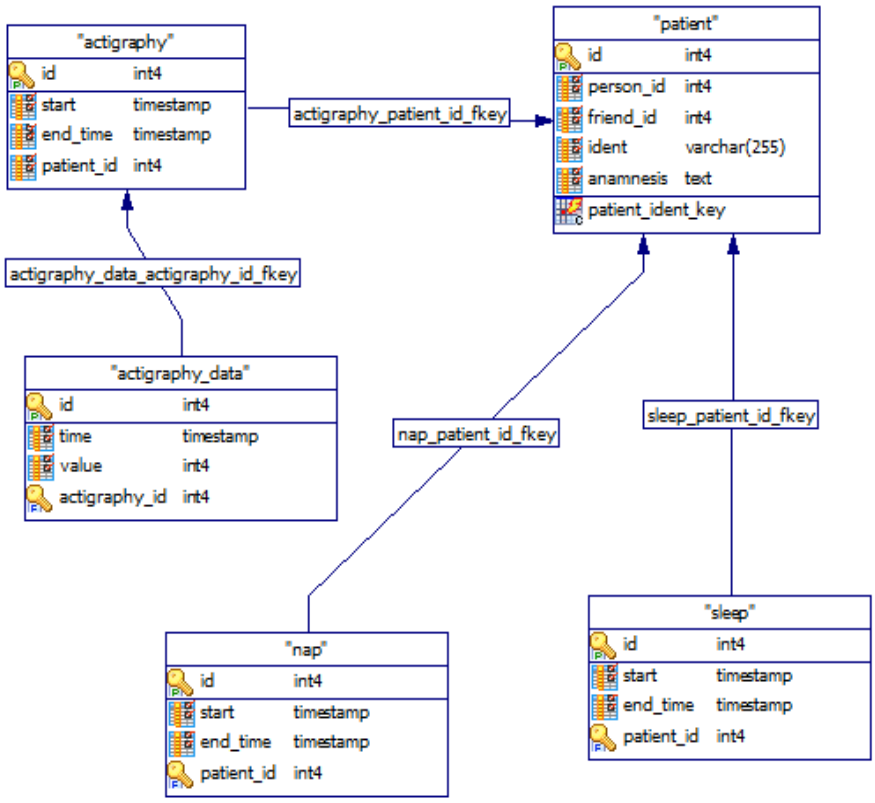


Figure 16: DB Scheme - actigraphy

Other tables which are not included in previous schema parts can be found on the attached CD. Structure of these tables is easy and does not need closer explanation. For instance table *hospitalization* is related to *patient* and stored time range of patient's hospitalization (*depression, mania* in the same way).

5.1.2 OLDES

In this part the important part from DB Schema will be described. The whole database which is shared for ODLES web interface and INK PC application contains 25 tables, 7 views and 4 trigger functions.

5.1.2.1 Users

This part is very similar to Figure 14 where the structure of *user, patient, doctor* and *role* is placed. The only difference is that in OLDES is no *friend* or person *table*.

5.1.2.2 Patient physiological values

Each physiological value (Figure 17) is stored in one table (*pulse, pressure, weight, glycemia*). These tables are quite simple – contains *date* of measurement and measured *value* and of course link to *patient*. *Diet* is not physiological value – it holds patient's food consumption – but the organisation of the table is similar.

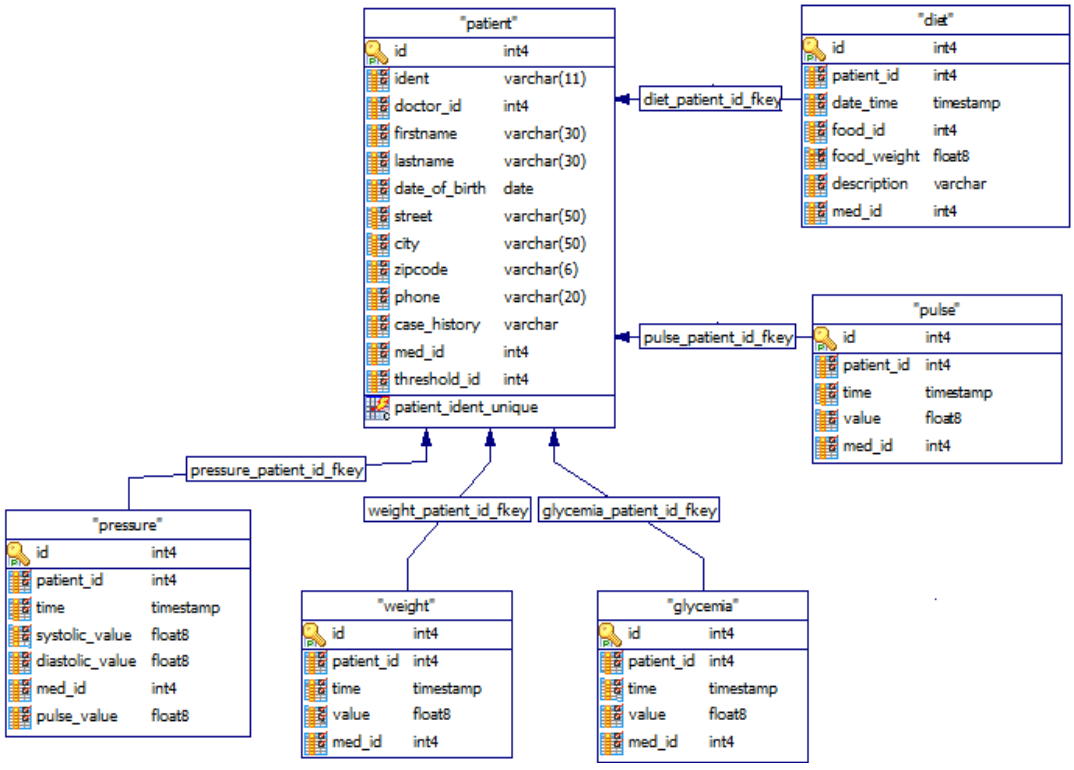


Figure 17: DB Schema - Physiological Data

5.1.2.3 Alarms

When an alarm is detected it must be added/stored in the DB – table *alarm* (Figure 18). This table is related to a concrete patient to distinguish person with the alarm. The table contains short info about alarm and two boolean columns. One indicates whether the alarm was acknowledge by patient and second indicates the doctor's acknowledgement. Detection of an alarm takes *threshold* values into consideration. Each patient has different threshold for hyperglycemia, hypoglycemia and hypertension – *threshold* table.

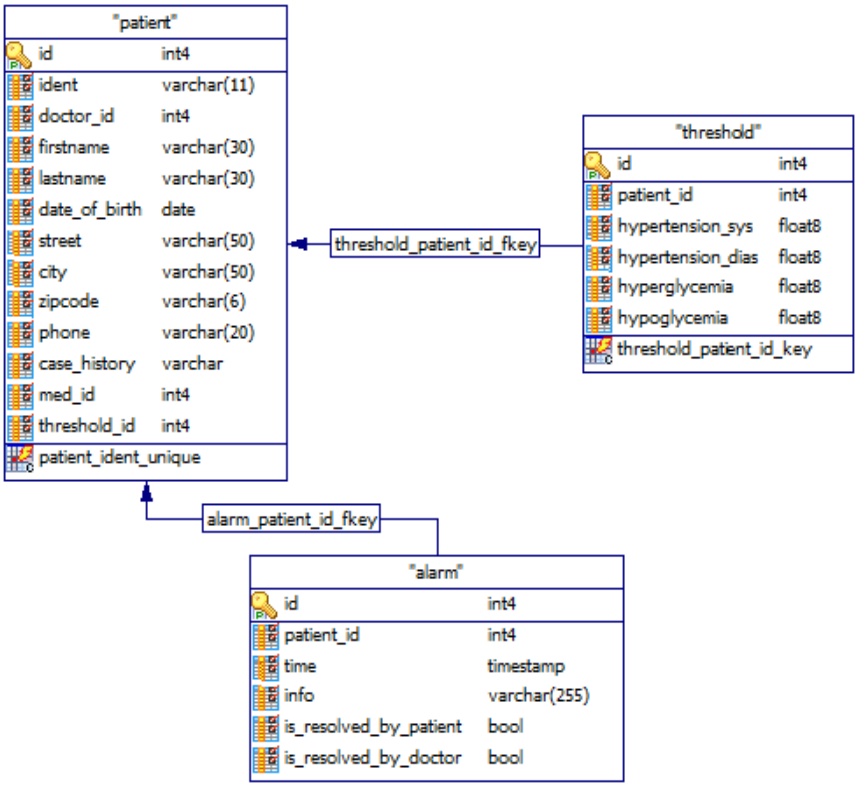


Figure 18: DB Scheme - alarm

The whole DB schema is included on the attached CD and contains rest of the tables. These tables are not so important for web application because they are related to a food menu mainly used on INK application.

5.1.2.4 Triggers

DB implements some triggers as well. The triggers are used to insert current time into tables with physiological values. See Example 1 below where after insertion of new row into *weight* table, the new record is updated – current server time is set to the column for *time*.

```
CREATE OR REPLACE FUNCTION update_weight_time() RETURNS trigger AS '
  DECLARE
    current timestamp;
  BEGIN
    IF tg_op = ''INSERT'' THEN
       current := CURRENT_TIMESTAMP;
       UPDATE weight SET "time" = current WHERE id = NEW.id;
    END IF;
    RETURN new;
  END
  ' LANGUAGE plpgsql;

  CREATE TRIGGER update_weight AFTER INSERT ON weight
    FOR EACH ROW EXECUTE PROCEDURE update_weight_time();
```

Example 1: Trigger to update time of weight insertion

5.2 View

All view components are based on JavaServer Pages (JSP). This Java technology uses simple HTML tags together with special 'user-defined' tags which allow to built HTML content dynamically. Struts in its both versions offers set of useful tags which include cycles, conditions etc. Furthermore, with *Tiles* [3] technology is easier to develop common look and feel web pages.

5.2.1 Tiles

The most of web applications have common layout for all use cases. The same feature can be noticed in the discussed application. Tiles is a templating system which is a part of Struts framework and provides common look and feel for a web application and reusable view components. Thanks to Tiles we can use one JSP file which defines whole web page layout and concrete content is included in this JSP.

In our Example 2 is shown template which defines structure of HTML page. Tag `<tiles:insert>` in the template file determines location where content should be placed. Tag `<tiles:put>` in the second code example (Example 3) defines source of concrete JSP files which will be displayed in client browser. Now we can use only one layout file (*basicLayout.jsp*) and vary its content.

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html"%>
<%@ taglib uri="http://struts.apache.org/tags-bean" prefix="bean"%>
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

...
    <div id="id-wrap">

        <!-- Header page information -->
        <tiles:insert attribute="header"/>

        <!-- Links bar -->
        <tiles:insert attribute="links"/>

        <!-- Main body information -->
        <div id="id-main">
          <div id="id-main-body">
            <div id="id-main-body-content">
              <tiles:insert attribute="body-content"/>
            </div>
          </div>
        </div>

        <!-- Footer information -->
        <tiles:insert attribute="footer"/>

    </div>

...
```

Example 2: Tiles template - basicLayout.jsp

```
<%@ taglib uri="http://struts.apache.org/tags-tiles" prefix="tiles"%>

<tiles:insert page="/jsp/eu/oldes/doctors/web/layout/basicLayout.jsp"
flush="true">

<tiles:put name="header"
value="/jsp/eu/oldes/doctors/web/layout/header.jsp" />
<tiles:put name="links"
value="/jsp/eu/oldes/doctors/web/layout/links.jsp" />
<tiles:put name="body-content"
value="/jsp/eu/oldes/doctors/web/patients/patientsListContent.jsp"/>
<tiles:put name="footer"
value="/jsp/eu/oldes/doctors/web/layout/footer.jsp" />

</tiles:insert>
```

Example 3: Concrete JSP file where content is included

Examples above are from OLDES application. In Bipometrics or Struts 2 is the only difference in existence of the central tiles definition for all pages but again the approach is the same.

5.2.2 GUI for mobile devices

Application for OLDES project requires access through web browser and mobile devices (cell phone, PDA, etc.) as well. The doctors require displaying of patient's personal data, anamnesis and some physiologic data which are relevant for medical care. In contrast to Web Interface this application should not display charts but only the last few values which are necessary for doctors to obtain view of patient's actual condition. The main task is to modify presentation layer because thanks to layered approach (MVC) the business logic is varied minimally.

Easy way how solve this is to use *XHTML MP* [39]. The technology uses subset of classic XHTML and the developer does not have to learn new approach to develop an interface for mobile devices. For instance WML is another possibility to implement the interface which is accessible through WAP but it is not pure HTML or XHTML. The greatest advantage brought by XHTML MP is that developers can now use the same technologies for the development of web sites and WAP sites.

```
...
  <!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtmlmobile10.
  dtd">
  ...
  <body>
    <div id="id-wrap">
      <!-- Header page information -->
      <tiles:insert attribute="header"/>
      <!-- Main body information -->
      <div id="id-main">
        <div id="id-main-body">
           <div id="id-main-body-content">
             <tiles:insert attribute="body-content"/>
           </div>
        </div>
      </div>
    <!-- Footer information -->
    <tiles:insert attribute="footer"/>
  </div>
```

Example 4: Basic layout of views for mobile devices

5.2.3 Charts creation

JFreeChart [11] library is used for chart creation. The library supports wide variety of chart types.

JFreeChart supports many output types including Swing components, image files (including PNG and JPEG) and vector graphics file formats (including PDF). It means that we can not included the result in JSP (or HTML) page before previous image saving. Problem with embedding graphical charts into a web page is solved thanks to next third-party library – *Cewolf* [22]. This library is based on JFreeChart and uses its rendering engine to render the final chart image into the client´s response stream. No files are created on server side. Everything is based on lightweight session objects and dynamic data analysis. CewolfServlet handles the chart rendering. The tag-library translates the chart definition included in the JSP, into an HTML `img` tag which consults the rendering servlet for retrieval of the appropriate chart. Cewolf is distributed under the same licence term (LGPL) as JFreeChart.

5.2.3.1 Code example

The box below (Example 5) contains sample implementation of `DatasetProducer` interface which requires method `produceDataset(Map)`. The method produces an `org.jfree.data.Dataset` object which will be rendered as a chart afterwards. In the example `TimeSeriesCollection` is used (implementation of `DataSet`) which is composite from `TimeSeries` objects. `TimeSeries` represents a sequence of zero or more data items in the form - period/value.

```
  public class PatientPhysiologicDataset implements DatasetProducer,
CategoryToolTipGenerator, Serializable {

   ...

   public Object produceDataset(Map arg0) throws DatasetProduceException {

        TimeSeries s = null;
        TimeSeriesCollection dataset = new TimeSeriesCollection();

         for (PatientPhysiologicData physiologicData : getData()) {
                s = new TimeSeries(physiologicData.getName(),
Minute.class);
                for (PhysiologicDataCollection physiologicDataCollection :
physiologicData.getData()) {
                    s.addOrUpdate(new
Minute(physiologicDataCollection.getTime().getTime()),
physiologicDataCollection.getValue());
                }
                dataset.addSeries(s);
           }

           return dataset;
   }

   ...
}
```

Example 5: DatasetProducer implementation

Example 6 shows sample of embedding chart in JSP via Cewolf tags.

```
  <%@ taglib uri='/WEB-INF/cewolf.tld' prefix='cewolf' %>

<%@page import="eu.oldes.doctors.web.charts.PatientPhysiologicData"%>

<jsp:useBean id="glycemiaData"
class="eu.oldes.doctors.web.charts.PatientPhysiologicData"/>
<jsp:useBean id="glycemiaDataset"
class="eu.oldes.doctors.web.charts.PatientPhysiologicDataset"/>

<%
    Object reqData =
request.getAttribute(Constants.PATIENT_GLYCEMIA_DATA_KEY);

     glycemiaData.setData((Collection<PhysiologicDataCollection>)reqData);
     glycemiaData.setName("Glycemia");

     List<PatientPhysiologicData> dataList = new
ArrayList<PatientPhysiologicData>();
     dataList.add(glycemiaData);
     glycemiaDataset.setData(dataList);

%>
...
<cewolf:chart
    id="glycemia-chart"
    title="Patient's Glycemia"
    type="timeseries"
    xaxislabel="Time"
    yaxislabel="Value">
    <cewolf:data>
        <cewolf:producer id="glycemiaDataset"/>
    </cewolf:data>
</cewolf:chart>

<cewolf:img chartid="glycemia-chart" renderer="cewolf" width="880"
height="600" />

...
```

Example 6: Chart embedded in JSP

5.2.3.2 PDF and CSV export

Each chart created by JFreeChart and displayed in JSP view can be exported to PDF (example in Figure 19) and CSV format. A chart in PDF may serve to easier printing of the result. CSV allows further processing in a table processor like Excel.

Below is introduced implementation of the PDF export from Bipometrics project. Core business logic is placed in a Struts action implementation – class `PrintAsPDFAction.java`. The action uses *iText* [8] library which serves to manipulating with PDF document. The chart definition is gained from session and then `ByteArrayOutputStream` of the document is written to `ServletOutputStream` which sends binary data to the client. Example of source code follows (Example 7) with the result in Figure 19.

```
   // get img definition
   ChartImageDefinition imgDef = (ChartImageDefinition)
sessionMap.get(MapParamConstants.PARAM_SESSION_CHART_TO_PRINT);
   JFreeChart chart = (JFreeChart) imgDef.getChart();

   // create byte array of the document
   Document document = new Document(PageSize.A4.rotate(), 0, 0, 0, 0);
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    PdfWriter writer = PdfWriter.getInstance(document, baos);
    document.open();
   // chart drawing in PDF
    PdfContentByte cb = writer.getDirectContent();
    PdfTemplate tp = cb.createTemplate(PageSize.A4.getHeight(),
PageSize.A4.getWidth());
    Graphics2D g2 = tp.createGraphics(PageSize.A4.getHeight(),
PageSize.A4.getWidth(), null);
    Rectangle2D r2D = new Rectangle2D.Double(20, 22,
PageSize.A4.getHeight() - 40, PageSize.A4.getWidth() - 44);
    chart.draw(g2, r2D, null);
    g2.dispose();
    cb.addTemplate(tp, 0, 0);
   document.close();

   // sends data to the client
   ServletOutputStream out = response.getOutputStream();
    baos.writeTo(out);
    out.flush();
```

Example 7: PDF document creation

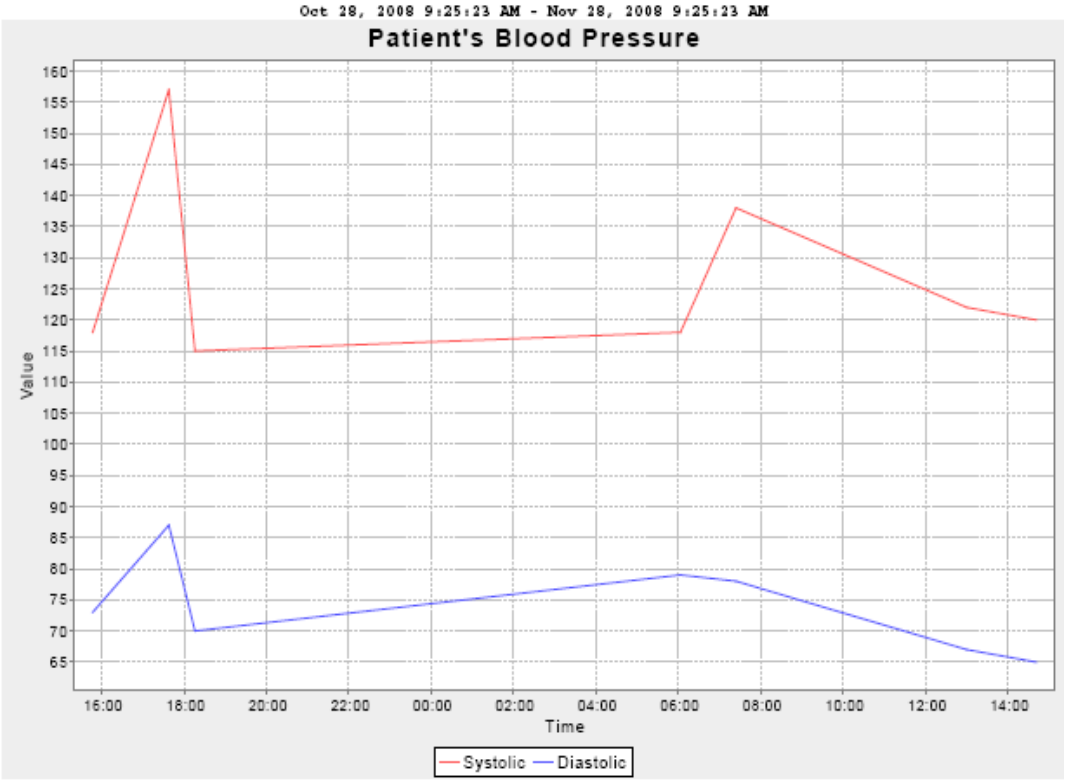


Figure 19: Example of chart after export to PDF

Exporting in CSV uses the same approach with writing a byte array to servlet output stream.

5.3 Data uploading

One of the requirements for Bipometrics was to allow uploading data (questionnaire score or actigraphy) through the web interface. The data is stored in a file in XML format. This file is uploaded firstly and then processed by parser. Acquired data is inserted into DB tables.

5.3.1 XML format

The application supports uploading of two different XML files - questionnaire and actigraphy. Each of this document is defined by XML Schema (XSD) - see attached CD.

5.3.1.1 Actigraphy XML file

Actigraphy is a method of monitoring human activity cycles. To get valuable results for the measurement, it has to last at least the whole day with suitable sampling rate. The rate is usually between 30 seconds and 2 minutes depending on accuracy requirements. If we take into consideration than an usual recording is long over one year it is clear that we must cope with big amount of data. Such huge XML files could naturally cause performance problems within XML uploading and processing.

Fortunately, the sampling rate of a signal can be reduced by method from signal processing theory. The preprocessing of raw data is out of the scope of this work. For persons who are interested, the *downsampling* [38] and *mean/average filter* [7] were used to reduce data size.

5.3.2 File uploading

One of the reasons why Struts 2 was chosen for Bipometrics implementation is that this version supports an easy way for file uploading. Struts 2 utilizes the service of File Upload Interceptor. The Interceptor is part of default Struts interceptor stack and is applied whenever a file input is used.

At first, JSP body with file input must be prepared. The file will be stored to the request under upload name (attribute name in `s:file` tag). To enable the interceptor the `enctype` attribute in form tag has to contain `multipart/form-data` value.

```
   ...
   <s:form action="admin_QuestionnairesInsert" method="post"
enctype="multipart/form-data">
            <s:file name="upload" label="%
{getText('itareps.patient.questions.insert.file')}" />
            <s:submit cssClass="button" value="%
{getText('itareps.patient.questions.insert.submit')}" />
   </s:form>
   ...
```

Example 8: JSP form for file uploading

The uploaded file will be used in *admin_QuestionnairesInsert* action which is mapped to `QuestionnairesInsertAction.java` class.

```
   <action name="admin_QuestionnairesInsert"
class="cz.itareps.web.action.QuestionnairesInsertAction">
    ...
    <result name="success">jsp/cz/itareps/web/admin/upload.jsp</result>
    <result name="input">jsp/cz/itareps/web/admin/upload.jsp</result>
   </action>
```

Example 9: Struts 2 action mapping

The only thing which has to be presented at action implementation are getters and setters of uploaded file, its name and content type. Name of field (upload in our case) for uploaded file must match with the name `s:file` tag on JSP.

```
   public class QuestionnairesInsertAction extends ActionSupport {
   /** Uploaded file */
   private File upload;
   /** Uploaded file content type */
   private String uploadContentType;
   /** Uploaded file name */
   private String uploadFileName;
   ...
   public String execute() throws Exception {
    ....
     // uploaded file processing by parser
    parser = SAXParser.getSAXParser(SAXParser.SAX_TYPE_QUESTINNAIRE,
upload);
    handler = new QuestionnairesHandler();
    parser.parse(handler);
   ...
   // data inserting into DB
   DAOFactory factory = DAOFactory.instance(DAOFactory.HIBERNATE);
    QuestionnaireDAO questionnaireDAO = factory.getQuestionnaireDAO();
    listToInsert =
questionnaireDAO.checkConstraint(handler.getPatientData());
    questionnaireDAO.insert(listToInsert, true);
   ...
   }

   public File getUpload() {
    return upload;
   }
   public void setUpload(File upload) {
    this.upload = upload;
   }
   public String getUploadContentType() {
    return uploadContentType;
   }
   public void setUploadContentType(String uploadContentType) {
    this.uploadContentType = uploadContentType;
   }
   public String getUploadFileName() {
    return uploadFileName;
   }
   public void setUploadFileName(String uploadFileName) {
    this.uploadFileName = uploadFileName;
   }
   ....
   }
```

Example 10: Action which uploads XML file

5.3.3 XML parsing

The uploaded file in XML format is parsed by `SAXParser` which checks validity of the document against XSD. Core of the parsing is performed in handler where parsed data are transformed to objects which are carried by container. The container is then used in action to inserting data into DB. See Example 10 above where is source code of `SAXParser` calling from `QuestionnairesInsertAction`.

The diagram below (Figure 20) shows dependency of classes within questionnaires insertion. The action calls `SAXParser` with `QuestionnaireHandler` instance. The handler contains filed with `QuestionnaireContainer` where gained data is stored and then carried back to `QuestionnaireInsertAction`.

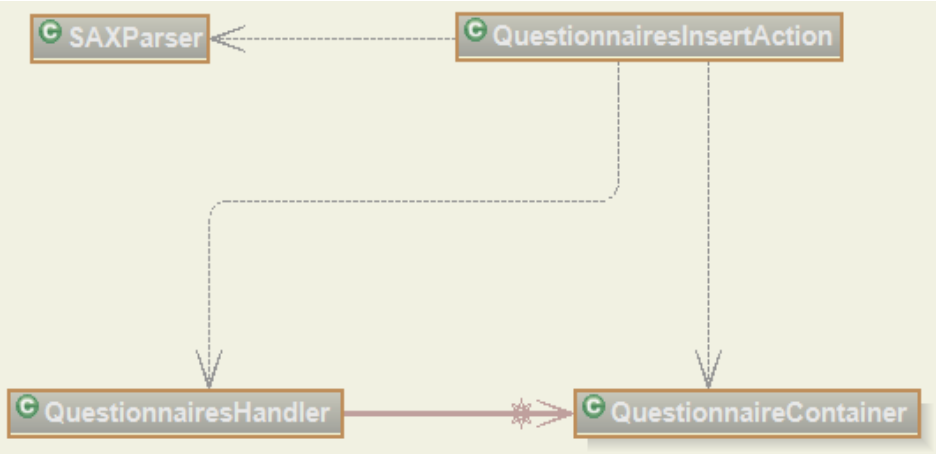


Figure 20: Dependency of the classes within XML parsing

5.4 DB access by Hibernate

Design of DB access has been already discussed in Chapter 4. Here we describe concrete implementation from Bipometrics where Hibernate as ORM framework is used. The class diagram (Figure 21) shows interactions between DAO factories, DAO objects and transfer object. To simplify the diagram, only `Questionnaire.java` class as transfer object is displayed as well as DAO for questionnaire (`QuestionnaireDAO.java`) only. The transfer object which carries data from relational table questionnaire is mapped automatically by Hibernate and the mapping is written in XML file (see Example 11).

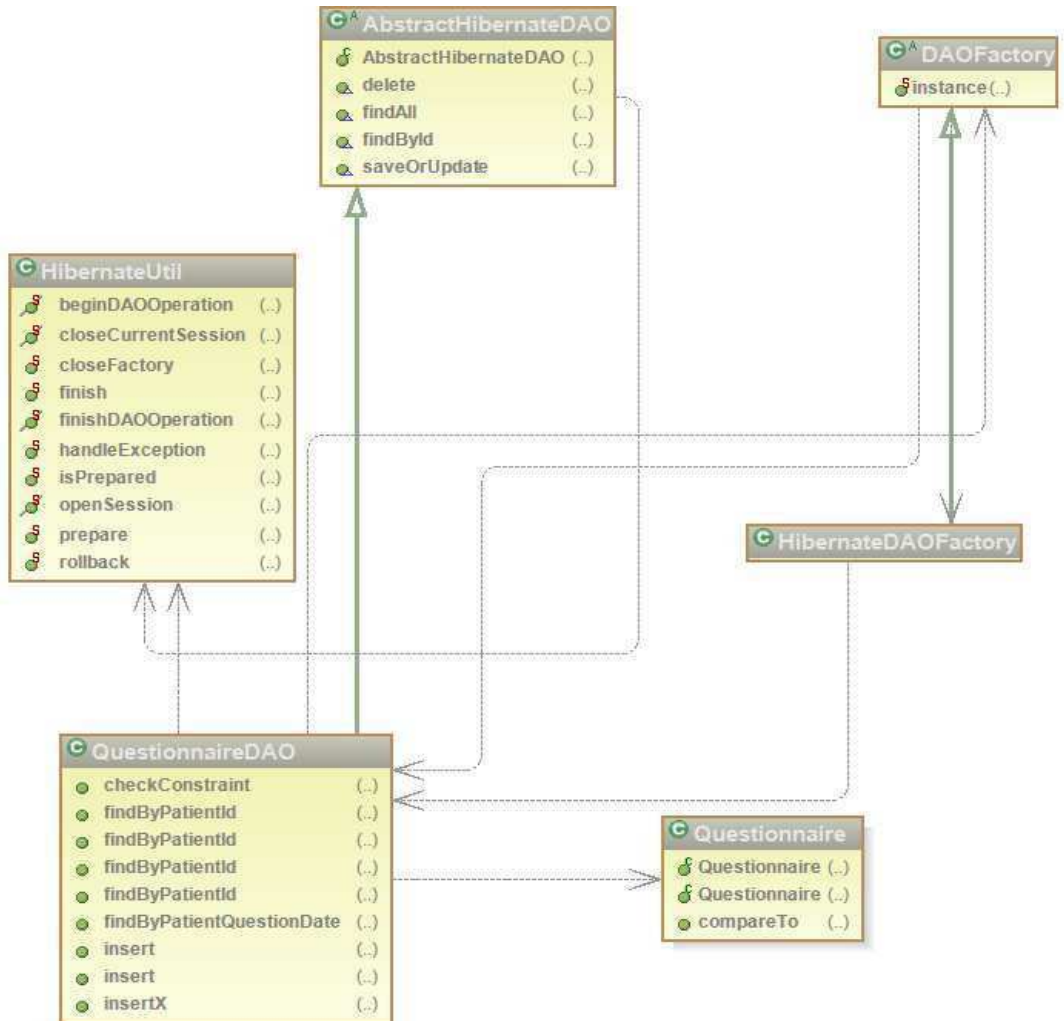


Figure 21: Classes related to DB access by Hibernate

5.4.1 Detailed description of each class

- `DAOFactory` - The abstract factory (*Abstract Factory design pattern*) encapsulates group of individual factories. Nowadays the application supports only DAOs implemented via Hibernate framework. The Hibernate supports many DB types and changing concrete DB machine is easy and does not have an impact on DAO implementations. This factory exists if in the future any other implementation of persistence is needed (e.g. JPA). It means that now is always return `DAOFactory` which creates DAOs implemented via Hibernate framework.

- `HibernateDAOFactory` - The factory creates instances of concrete DAOs implemented via Hibernate framework. Contains methods like `getQuestionnaireDAO()` an so on.
- `AbstractHibernateDAO` - This abstract class implements common methods (`delete`, `findAll`, `findById`, `saveOrUpdate`) which are needed for interaction with data source.
- `QuestionnaireDAO` - Data Access Object which provides methods to manipulate `Questionnaire` data object (value object) persistence. The objects are accessed and managed by Hibernate framework. The class extends `AbstracHibernateDAO`.
- `HibernateUtil` - The helper class which takes care of start-up and makes accessing a `SessionFactory` convenient. A `SessionFactory` can open up new Session's.
- `Questionnaire` - Value object which holds questionnaire properties. Implemented on basis of JavaBean conventions.

```
  <hibernate-mapping>
      <class name="cz.itareps.web.vo.Questionnaire" table="questionnaire"
schema="public">
          <id name="id" type="int">
              <column name="id" />
              <generator class="seqhilo">
               <param name="sequence">questionnaire_id_seq</param>
              </generator>
          </id>
          <many-to-one name="question" class="cz.itareps.web.vo.Question"
fetch="select">
              <column name="question_id" not-null="true" />
          </many-to-one>
          <many-to-one name="patient" class="cz.itareps.web.vo.Patient"
fetch="select">
              <column name="patient_id" not-null="true" />
          </many-to-one>
          <property name="date" type="date">
              <column name="date" length="13" not-null="true" />
          </property>
          <property name="value" type="byte">
           <column name="value" precision="2" scale="0" not-null="true" />
          </property>
          <property name="patientId" type="int" insert="false"
update="false" >
              <column name="patient_id" not-null="true" />
          </property>
          <property name="questionId" type="int" insert="false"
update="false" >
              <column name="question_id" not-null="true" />
          </property>
      </class>
  </hibernate-mapping>
```

Example 11: Hibernate mapping file for questionnaire table

5.4.2 Comparison with basic JDBC

As we have already known from section 4.3.2, Bipometrics application uses Hibernate to simplify and make data manipulating and querying more error-resistant in comparison to OLDES JDBC usage. Example 12 or Example 13 is the implementation of a simple query by JDBC or Hibernate.

```
  ....
  conn = PostgresHelper.createConnection();
pstmt = conn.prepareStatement(SQL_SELECT_SYSTOLIC_PRESSURE);
pstmt.setInt(1, aId);
pstmt.setTimestamp(2, new Timestamp(aTimeFrom.getTimeInMillis()));
pstmt.setTimestamp(3, new Timestamp(aTimeTo.getTimeInMillis()));
rs = pstmt.executeQuery();
while (rs.next()) {
Calendar cal = Calendar.getInstance();
cal.setTime(rs.getTimestamp(1));
systolicPressure.add(new PhysiologicDataCollection(cal, rs.getFloat(2)));
  ...
```

Example 12: Simple DB query by JDBC

```
  List<Criterion> criterias = new ArrayList<Criterion>();
   if (id != null) {
    criterias.add(Restrictions.eq(DB_PROPERTY_NAME_PATIENT_ID, id));
   }
   if (timeFrom != null) {
    criterias.add(Restrictions.ge(DB_PROPERTY_NAME_START, timeFrom));
   }
   if (timeTo != null) {
    criterias.add(Restrictions.le(DB_PROPERTY_NAME_END, timeTo));
   }
   Criterion[] cr = (Criterion[]) criterias.toArray(new Criterion[0]);
   List<Mania> list = findByCriteria(cr);
   return list;
```

Example 13: Simple DB query by Hibernate

The First example with basic JDBC uses a SQL statement which depends on DB implementation. The Hibernate example does not use any SQL but restriction by criterias only. Hibernate returns a value object (Mania) directly but with JDBC new instance has to be created by hand and result must be iterated through the `ResulSet`.

5.5 Authentication

The sequence diagram in Attachment D shows how the interaction within login action with JAAS API usage was designed. Here is discussed concrete implementation view on class diagram (Figure 22) of JAAS authentication from Bipometrics.

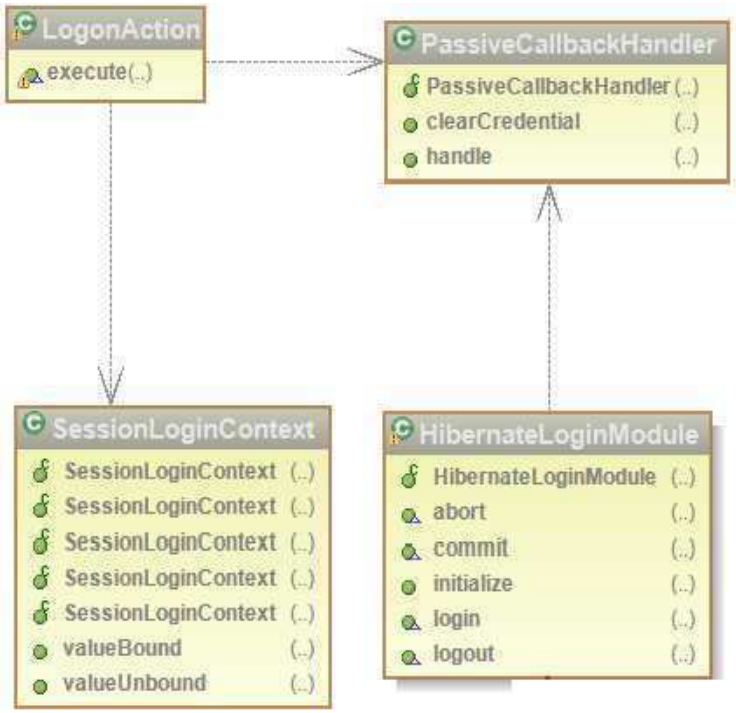


Figure 22: Authentication class interaction

- `LogonAction` - Performs log on the application. Stores logged on `User` object to the session. If authentication process failed, client is returned to log on page and error message is displayed. The `SessionLoginContext` is created with `PassiveCallbackHandler` and stored in the session.

```
   PassiveCallbackHandler cbh = new PassiveCallbackHandler(getUsername(),
getPassword());
   SessionLoginContext lc = new
SessionLoginContext(SecurityUtil.KEY_PARAM_LOGIN_CTX, cbh);

   // Setting the SessionLoginContext object in the Session
   // will trigger the valueBound() method in the object which
   // calls login() on the SessionLoginContext.
   getSessionMap().put("loginContext", lc);
   // returns subject if login was successful
   Subject subj = lc.getSubject();
```

Example 14: Login action example

- `PassiveCallbackHandler` - Holds username and password (take by constructor) so its `handle()` method does not have to prompt the user for input. The `handle()` method handles name and password callbacks.

- `SessionLoginContext` - Extends the JAAS `LoginContext` so that when the instance is bounded to a `HttpSession`, it will execute `login()` and when the session times out, it will execute `logout()`. The `login()` and `logout()` methods are implemented in `LoginModule` which is selected by `LoginContext` according to *jaas.config* file module configurations.

```
   OLDES {
      eu.oldes.doctors.web.security.RDBMSLoginModule required debug="true"
      ;
   };

   BIPOMETRICS {
      cz.itareps.web.security.authent.HibernateLoginModule required
debug="true"
      message.digest.algorithm=MD5;
   };
```

Example 15: jaas.config file

- `HibernateLoginModule` - The login module that authenticates a given username/password credential against a DB data source using `Hibernate DAO` objects and return subject object (user).

5.6 Authorization

Before an action is invoked, the JAAS authorization process must decide whether the user has permission to perform the action. The decision is made according to action URL where before each action stands a prefix. The prefix says which type of permission is needed. The Bipometrics checks the permissions in the `AuthorizationInterceptor` which is invoked before each action performing. Since Struts 1 does not support interceptors, the `ActionServlet` (`OldesActionServlet`) is extended to preform the permissions check. Class diagram of JAAS authorization in Bipometrics follows.

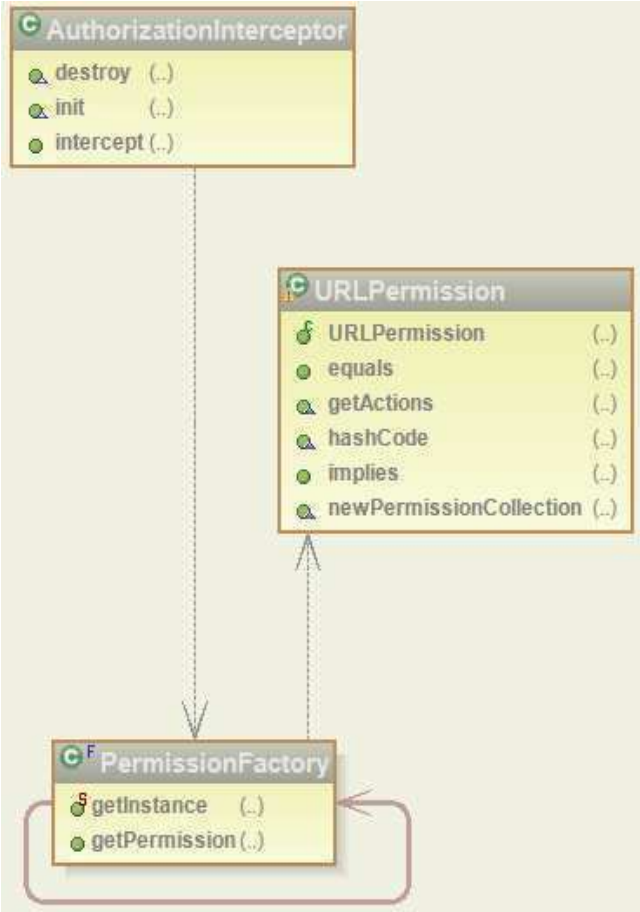


Figure 23: Authorization classes interaction

- `AuthorizationInterceptor` - Interceptor which protects secure actions from unauthorized access. Checks whether the `Subject` (user) stored in session scope has permission to perform requested action.

```
  ...
  String permissionName = actionName.substring(0,
actionName.indexOf("_"));
   Permission permission =
PermissionFactory.getInstance().getPermission(permissionName);
   Subject subj = (Subject)
sessionMap.get(MapParamConstants.PARAM_SESSION_USER_SUBJECT);
  if (!SecurityUtil.permitted(subj, permission)) {
   // subject is not permitted
   return SecurityUtil.ACTION_RETURN_CODE_UNAUTHORIZED;
   }
  ..
```

Example 16: Interceptor of the authorization

- `PermissionFactory` - The factory class creates instance of `Permission` - now only one is supported – `URLPermission`.

- `URLPermission` - The class represents access to a system resource. The name of the permission is a role name which is permitted to invoke the action which URL contains role name before action name. In JAAS policy file is specified permission - permission `cz.itareps.web.security.author.URLPermission` "*general*"; - it means that the role with this permission is permitted to invoke actions like - */general_MyAction.action*.

```
  grant  Principal cz.itareps.web.security.RolePrincipal "user" {
    permission cz.itareps.web.security.author.URLPermission "anonym";
    permission cz.itareps.web.security.author.URLPermission "general";
    permission cz.itareps.web.security.author.URLPermission "basic";
  };

  grant  Principal cz.itareps.web.security.RolePrincipal "admin" {
    permission cz.itareps.web.security.author.URLPermission "anonym";
    permission cz.itareps.web.security.author.URLPermission "general";
    permission cz.itareps.web.security.author.URLPermission "admin";
  };
```

Example 17: jaas.policy file

5.7 Web services

OLDES application offers set of services which are available for calling from INK application:

- insert new weight value into DB
- insert new glycemia value into DB
- insert new blood pressure value into DB
- check existence of an unsolved alarm
- solve an alarm

The client (INK) communicates with server (Tomcat) where the services are deployed, through *SOAP* [34]. SOAP is an XML-based communication protocol and encoding format for inter-application communication. Implementation of Web Service server side is elementary thanks to *Apache Axis* [4]. Axis is a framework for constructing SOAP processors such as clients, servers etc.

Developer is free of deploying the services because all steps are performed by mentioned framework. Axis implements the `ActionServlet` – `AxisServlet` – which publishes all requested services and prepares their description via *WSDL* [35] files. When the `AxisServlet` is configured correctly within the web server, the deploying process will be performed automatically with the server launching.

The developer must provide only two things – service implementation and deployment descriptor to tell Axis which service should be published. The implementation is *Plain Old Java Object (POJO)* which queries DB through the concrete DAO and returns suitable value.

Deployment descriptor is placed in an AAR file (Axis archive – simple zip file) in *services.xml*. In Example 18 all public methods from `OLDEService` will be available for remote invoking as web services.

```
  <service name="OLDES_WS">
      <description>
          OLDES web services
      </description>
      <parameter
name="ServiceClass">eu.oldes.doctors.ws.OLDESService</parameter>
      <messageReceivers>
    <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"
     class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
      </messageReceivers>
  </service>
```

Example 18: Web service deployment descriptor

Axis tries to deploy all AARs from *WEB-INF/services* folder which are listed in *services.list* file from the same folder.

5.8 Security

5.8.1 SSL

Each client must perform login into the application which means to send delicate data (e.g. password, patient's personal information) over the Internet. However the Internet is an insecure network, that is why we must solve security of our web application. Fortunately, cryptographic protocols exist which provide secure communication on the Internet. One well-tried protocol from this family is *Secure Sockets Layer (SSL)*. Configuration of web server (Tomcat in our case) to support this communication through SSL is described in Appendix E.

5.8.2 SHA

The next security bottleneck occurs in a database where data are stored. Problem is with user's password again. It is not secure to store password in its pure form. Solution is in a hash algorithm which computes a fixed-length digital representation (known as a message digest) of an input data sequence (the message) of any length. The algorithm is called *Secure Hash Algorithm (SHA)* and is also implemented in Java language. Static method `TextHash.execute(String)` (see Example 19) is invoked whenever we need return message digest of an input string (e.g. password storing or modification) in the applications.

```
public class TextHash {

  public static String execute(String aText) {
      MessageDigest md = null;
      try {
          md = MessageDigest.getInstance("SHA");
      } catch (NoSuchAlgorithmException e) {
          // exception handling
      }
      try {
          md.update(aText.getBytes("UTF-8"));
      } catch (UnsupportedEncodingException e) {
          // exception handling
      }
      byte raw[] = md.digest();
      String hash = (new BASE64Encoder()).encode(raw);
      return hash;
  }

}
```

Example 19: Text hashing function

Chapter 6

Testing

It is necessary to test implemented web portals before their release to production use. The applications are available for testing on NIT server in Department of Cybernetics. OLDES Web Medical Interface can be found at *https://nit.felk.cvut.cz/oldes/* (or *https://nit.felk.cvut.cz/wap/oldes/* for access from mobile devices) and Bipometrics Psychiatrist Web Portal at *https://nit.felk.cvut.cz/tomcat/bipometrics/*. At the time of submitting this work (May 2009), OLDES project has already passed the first phase of pilot testing. The testing of Bipometrics web portal is planned for the second half of 2009. On the other hand several bugs had been fixed thanks to testing by other participants on this project.

The test scenarios which describes in detail the test plans to test the functionalities of the implemented portals are available on the attached CD.

6.1 Results of OLDES Pilot Testing

The main goal of diabetes's pilot is to achieve better compensation of diabetes in hard-to-compensate patients by flexible individualized approach to insulin dose adjustment. To evaluate the OLDES feasibility, in total 5 patients were involved in the hospital pilot (3 men and 2 women) for a period of 1 week. The inclusion criteria were: age over sixty and type 2 diabetes mellitus diagnosis. Patients measured their weight once a day, blood pressure and glucose level three times per day and provided input concerning their food. Moreover, patients filled up the evaluation questionnaires about ergonomics of GUI, wearable components and user friendliness of the system. Furthermore, the following technical parameters were tested:

1. performance, reliability and integration of the sensors,
2. data transfer and storage,
3. GUI and Web portal interface.

Figure 24: Example of food intake. Patient 2 selected food item before food scaling using the remote controller.

Patients used interactive scale connected to a computer database with data on frequent foodstuff and its sacharids content - see Figure 24. Automatic computation of total daily consumption simplified patient's control of sacharids intake. The resulting information were exported to a dietitian who could suggest recommendations for modification of patient's diet. The daily food consumption, blood pressure and weight development of patient 2 is depicted in Figure 25. The slight weight reduction is demonstrated in Figure 25 after few days of interaction with OLDES system. The increase of systolic and diastolic blood pressure was not severe in this case. All patients agreed that the OLDES system was user friendly. They mainly appreciated smart control of the GUI using remote controller and easy-to-use measurement of vital physiological parameters.

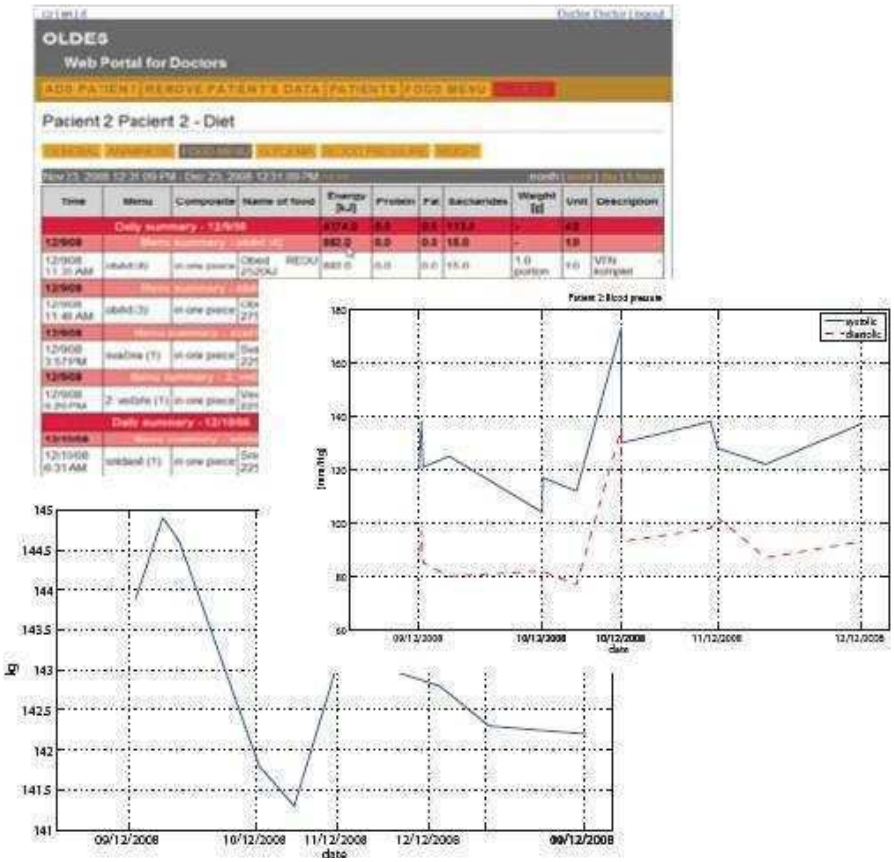


Figure 25: Example of food daily consumption, blood pressure and weight measurement of patient 2.

The expected results of the project and of the diabetes pilots can be resumed as follows:

- an open service oriented low-cost platform is provided allowing a granular management of users and of their authorisations and an easy connection of new services by public, private or non-profit service providers;
- the project proves feasibility of daily use of a low cost home access point allowing to access the services and to efficiently manage chronic diseases as diabetes applying vital physiological signals monitoring.

Regarding future direction, the diabetes pilot will be carried out in 10 households of senior persons during the next following months. The hospital pilot showed that OLDES approach can help to compensate patients suffering from diabetes. The expected advantage of OLDES monitoring at patient's homes will be improvement in their diabetes management without any need to increase frequency of patient's visits in a hospital.

6.2 Bipometrics Outputs

Next, we will demonstrate the main functions of Bipometrics web portal. The whole actigraph recording is shown in Figure 26. The detail is shown in Figure 27 where detection of two events is depicted: i) sleep start and ii) sleep end. User can change time scale within one day, one week and one year (see Figure 28).

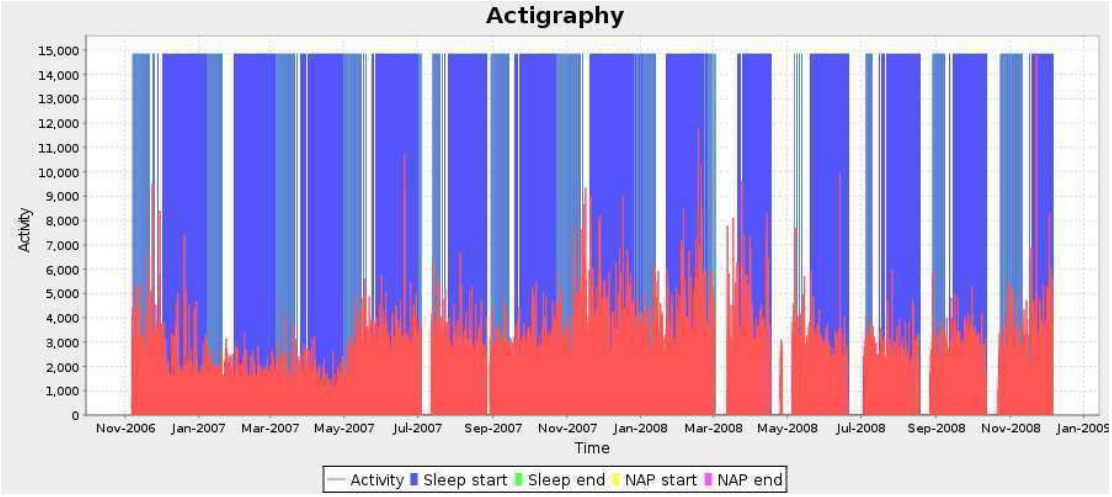


Figure 26: Whole actigraphy record

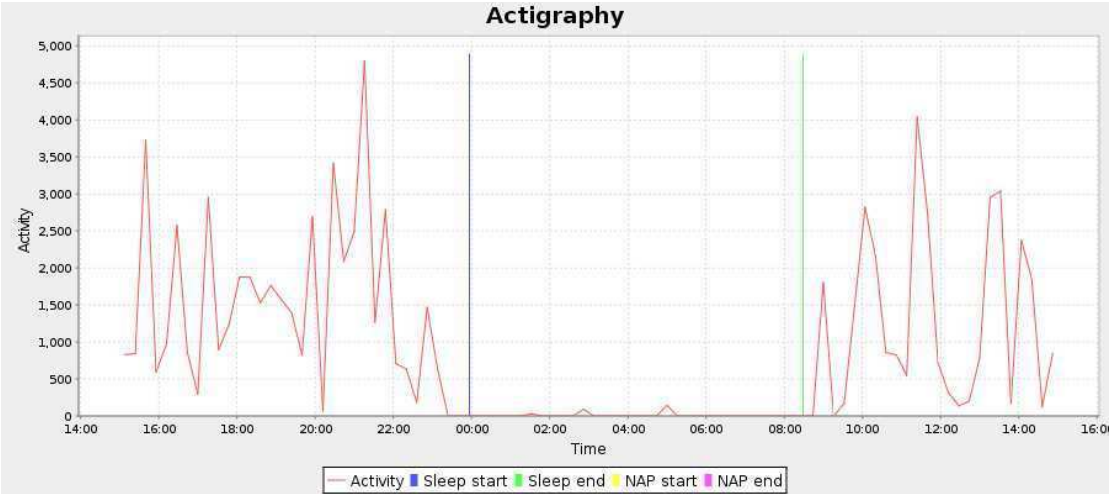


Figure 27: Actigraphy wirh sleep phase

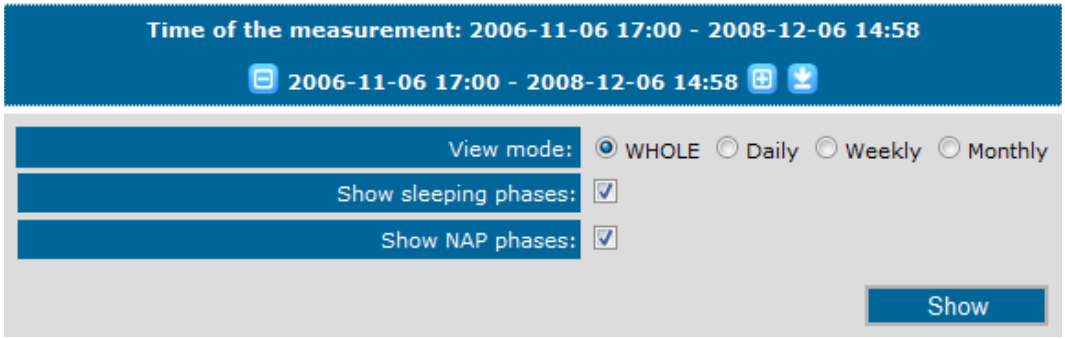


Figure 28: Display properties of actigraphy

Visualization of questionnaires is also available as can be seen in Figure 29. The threshold for alarm generation was set up to 7 in this case.

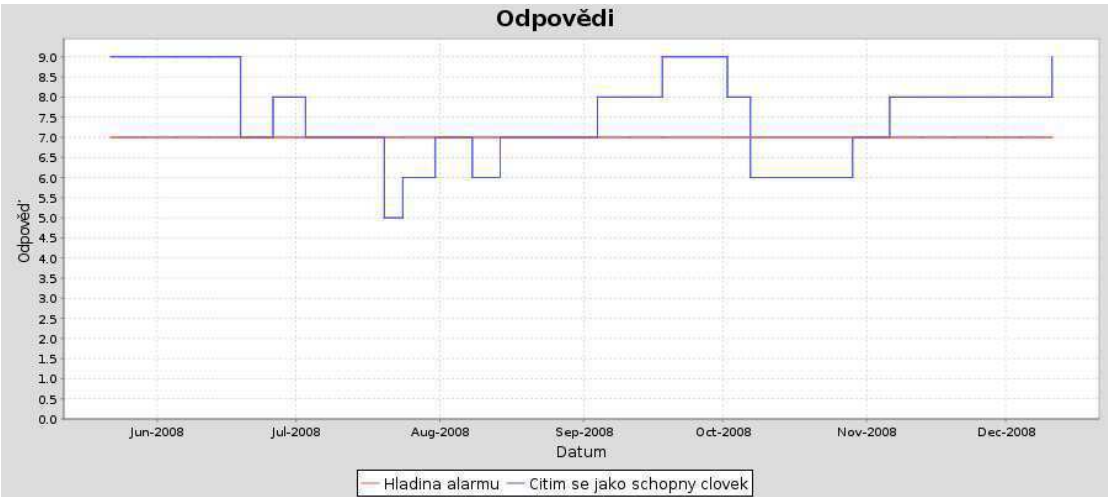


Figure 29: Example of a question score

User can change time scale (within one month, three month, six month and one year) and select concrete question (see Figure 30). Moreover, the events as a depression and mania onset and date of hospitalization due to relapse can be shown as well.

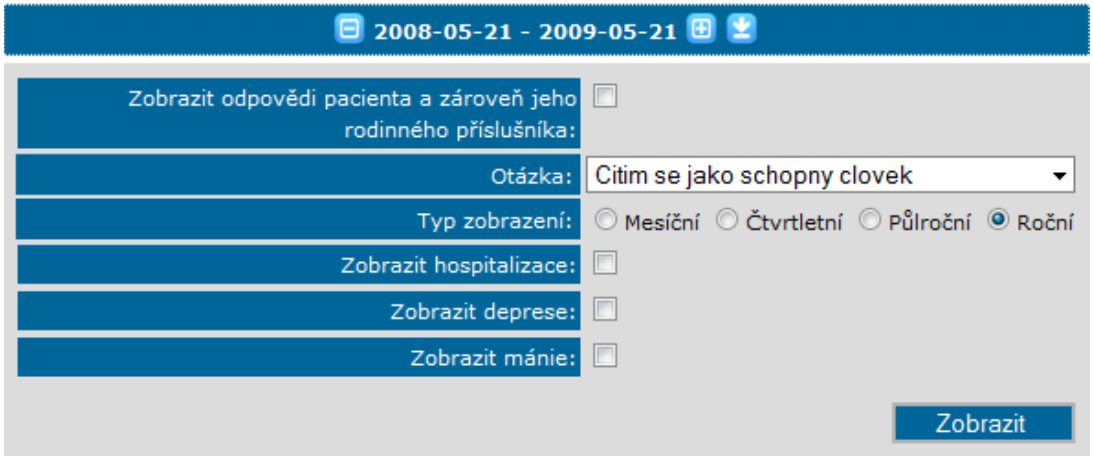


Figure 30: Display properties of questionnaire score

Chapter 7

Conclusion

This chapter includes summary of the whole thesis. The second part outlines additional future plans for the web portals extensions.

Two web portals for medical usage have been created. Fist one – OLDES Web Medical Interface – supports treatment of elderly people which suffers from diabetes. Each doctor with access to the portal can monitor actual patient's condition or brows history to see development of patient's treatment. The advantage of the web interface, in comparison to current applications which doctors use, is accessibility. The doctor can check current state from any computer which is connected to the Internet. Furthermore, basic information are also available via mobile devices (cell phone, PDA etc.). Physiological data – blood pressure, glycemia, weight – is displayed in more transparent way. All these values can be shown via charts and export to PDF document. CSV format, which is suitable for further processing in some table processor, is also available.

This system tightly interacts with local hub - INK PC - at patient's home. All measured values are inserted to the database by web services. These services are part of the web application but are invoked remotely by the INK. The application is able to find out an alarm during a physiologic value insertion. Patient and his/her doctor are immediately informed about the alarm occurrence. The notification is sent via e-mail message to a doctor's mail box. Furthermore, INK and web applications display message note about new alarm.

The second web application – Bipometrics Psychiatrists Web Portal – serves to psychiatrists who treat peoples suffering from bipolar disorder. Patient's mood is bounded to his/her activity. As it is discussed above, with a web access to the application is easier and faster for doctors to react to patient's condition changes. The Bipometrics main aim is to predicted relapses where the fast and concrete reaction is important.

The system again provides easy browsing of patient's history. Actigraphy and questionnaire scores are displayed in charts representation. Whenever an alarm (high score) is generated the doctor is immediately notified by e-mail message. An administrator is able to upload new data (score, activity) to the system from the XML file.

The both applications are implemented on the bases of MVC pattern in Java programming language. This fact allows future more simple extension according to user's requests.

7.1 Future Directions

It is planned to implement or upgrade further functionalities like:

- Detailed management of patients (better manipulating with patient's data and its maintaining) (OLDES, Bipometrics)
- Distinguish several types of actigraphy in context of diferrent sampling methods (Bipometrics)
- Asynchrounous alarms notification to INK applications (OLDES)
- More sophisticated method for alarm determination (Bipometrics)

Bibliography

[1] Apache Software Foundation. *Licenses*. <*http://www.apache.org/licenses/*>.

[2] Apache Software Foundation. *Struts 2 Big Picture*. <*http://struts.apache.org/2.0.14/docs/big-picture.html*>.

[3] Apache Software Foundation. *Tiles 2*. <*http://tiles.apache.org/*>. 2009.

[4] Apache Software Foundation. *Web Services – Axis*. <*http://ws.apache.org/axis/*>.

[5] Brown, D., Davis, C., Stanlick, S. *Struts 2 in Action.* Manning Publications. May 2008.

[6] Free Software Foundation, Inc. *GNU General Public License*. <*http://www.gnu.org/licenses/gpl-3.0.txt*>. June 2007.

[7] Librow. *Mean filter or avarage filter.* <*http://www.librow.com/articles/article-5*>.

[8] Lowagie, B., *iText – a Free Java-PDF Library.* <*http://www.lowagie.com/iText/index.html*>.

[9] Microsoft. *Microsoft SQL Server.* <*http://www.microsoft.com/sqlserver/2008/en/us/default.aspx*>.

[10] Newell, A., Gregor, P. *User sensitive inclusive design*. JIM 2001 Interaction Homme/Machine & Assistance. 2001.

[11] Object Refinery Limited. *JFreeChart*. <*http://www.jfree.org/jfreechart/*>.

[12] OpenSymphony. *Object-Graph Navigation Language*. <*http://www.opensymphony.com/ognl/*>

[13] OpenSypmhony. *WebWork Web Application Framework*. <*http://www.opensymphony.com/webwork/*>

[14] Oracle. *Oracle database*. <*http://www.oracle.com/global/cz/database/index.html*>.

[15] P. D. of the Department of Economic and W. P. P. Social Affairs of theUnited Nations Secretariat. *The 2008 revision and world urbanization prospects.* 2008.

[16] PosgreSQL Global Development Group. *PostgreSQL*. <*http://www.postgresql.org/*>.

[17] Poupě, J. *Activity analysis of bipolar disorder.* Master Thesis, Department of Cybernetics FEE CTU. 2008.

[18] Quesenbery, W. *User-centered design works*. <*http://www.wqusability.com/*>.

[19] Red Hat Middleware. *Relational Persistence for Java and .NET.* <*https://www.hibernate.org/*>.

[20] Rinaldi, G., Tamburriello, F. *OLDES basic technology infrastructure*. <*www.oldes.eu*>. 2007.

[21] Sedláčková, K. *Analýza bipolárních pacientů*. Master Thesis, Department of Cybernetics FEE CTU. 2008.

[22] Sourceforge.net. *Cewolf*. <*http://www.jfree.org/jfreechart/*>.

[23] Sun Microsystems. *Core J2EE Patterns – Data Access Object*. <*http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html*>.

[24] Sun Microsystems. *JAAS Authentication Tutorial*. <*http://java.sun.com/j2se/1.5.0/docs/guide/security/jaas/tutorials/GeneralAcnOnly.html*>.

[25] Sun Microsystems. *Java BluePrints, Model-View-Controller.* <*http://java.sun.com/blueprints/patterns/MVC-detailed.html*>.

[26] Sun Microsystems. *Java Platform, Enterprise Edition (Java EE)*. <*http://java.sun.com/javaee/*>.

[27] Sun Microsystems. *Java Platform, Standard Edition (Java SE).* <*http://java.sun.com/javase/*>.

[28] Sun Microsystems. *Java SE Security.* <*http://java.sun.com/javase/technologies/security/*>.

[29] Sun Microsystems. *MySQL*. <*http://www.mysql.com/*>.

[30] Sun Microsystems. *The Java EE 5 Tutorial*. <*http://java.sun.com/javaee/5/docs/tutorial/doc/*>.

[31] Sun Microsystems. *The Java Tutorials – Eceptions*. <*http://java.sun.com/docs/books/tutorial/essential/exceptions/*>.

[32] Španiel, F. *Information Technology Aided Relapse Prevention in Schizophrenia.* <*www.itareps.com*>. 2007.

[33] Usability Net. *Paper prototyping*. <*http://www.usabilitynet.org/tools/prototyping.htm*>. 2006.

[34] W3C Consortium. *SOAP Version 1.2*. <*http://www.w3.org/TR/soap/*>

[35] W3C Consortium. *Web Services Description Language (WSDL) 1.1*. <*http://www.w3.org/TR/wsdl*>.

[36] Wikipedia. *Telemedicine*. <*http://en.wikipedia.org/wiki/Telemedicine*>.

[37] Wikipedia. *Apache Struts*. <*http://en.wikipedia.org/wiki/Struts*>.

[38] Wikipedia. *Downsampling*. <*http://en.wikipedia.org/wiki/Downsampling*>.

[39] Wikipedia. *XHTML Mobile Profile*.
<*http://en.wikipedia.org/wiki/XHTML_Mobile_Profile*>.

Appendix A

Usability Lab

The usability lab consists of two rooms (see Figure A.1). In one, the participant room, the user will interact with the technology (software and hardware environment) under evaluation, as if working alone or with a co-All these data are fed into the observer room (next to the testing room) where evaluators monitor all sources on independent screens. Evaluators can communicate with users from the control room via microphone connected to a speaker in the testing room.

As the name of the lab suggest - the main issue is usability testing. Usability can be defined as the measure of the quality of a user's experience when interacting with a software product or a system (web site, a software application, mobile technology or any user operated device). The goal of usability testing is to ascertain what will help users accomplish their tasks and what may impede them. Usability testers build a set of scenario tasks they will ask users to attempt. As detailed information about user successes (or failures) is gathered and reported, the prototype of the system (or software product) can be modified. With wall-mounted cameras recording user's facial expressions and physicals movements and system responses directly captured by digital recording, the complete process of the user's interaction can be reviewed and analysed. Microphone in the room can record any comments the user makes.

The lab is used for joint projects with Sun Microsystems Czech Republic and for testing software developed in the framework of running projects in the department (e.g. mobile computing applications developed in the framework of the EC funded project Mummy, I2HOME).

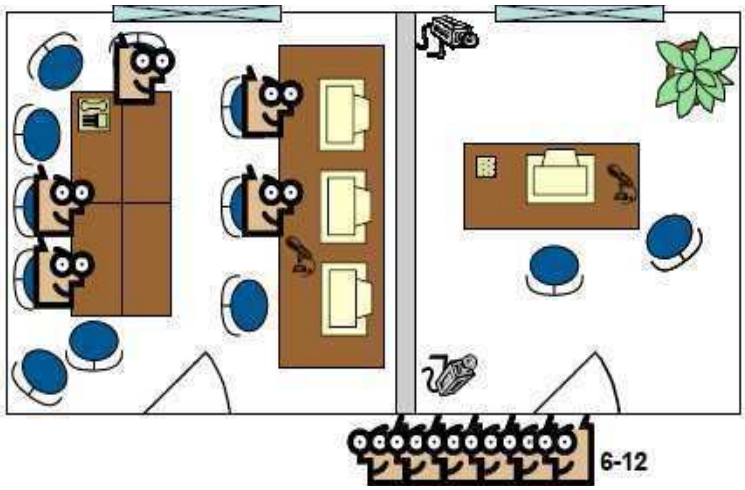


Figure A.1: Usability Lab

Appendix B

INK PC

Several projects in telemedicine or tele-care do not care much about the node at old person's home, using a commercial PC, maybe buying just needed components. OLDES, on the contrary, intending to reduce costs but, also, to grant a limited impact into old person's homes, is developing together with INK's existing low cost PC project, an appropriate PC with small dimensions, free software and just needed component.

Main characteristics required to the INK PC are:

- low cost;
- no software license fees;
- possibility of connecting directly a home TV by an input connector such as SCART;
- output connectors facilities for connecting external devices such as remote control and medical devices;
- input/output for audio;
- possibility of hosting applications for managing OLDES services;
- possibility of remote assistance.

Main declared characteristics of INK are:

- 4 x USB 2.0
- 1 x USB 1.1 (custom connector – intended for keyboard)
- 2 x SD card slot (4-bit transfers, with SDIO capability)
- 1 x VGA monitor connector
- 1 x Mic in
- 1 x Audio out (stereo)
- 1 x 10/100 Base T Ethernet
- 1 x 802.11 b/g WLAN
- 256MB DDR RAM
- 512MB/1GB NAND Flash (holds OS and apps – initial units have 1GB, may be reduced)

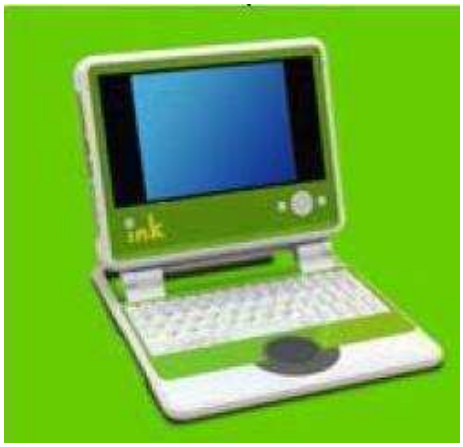


Figure B.1: INK PC

Appendix C

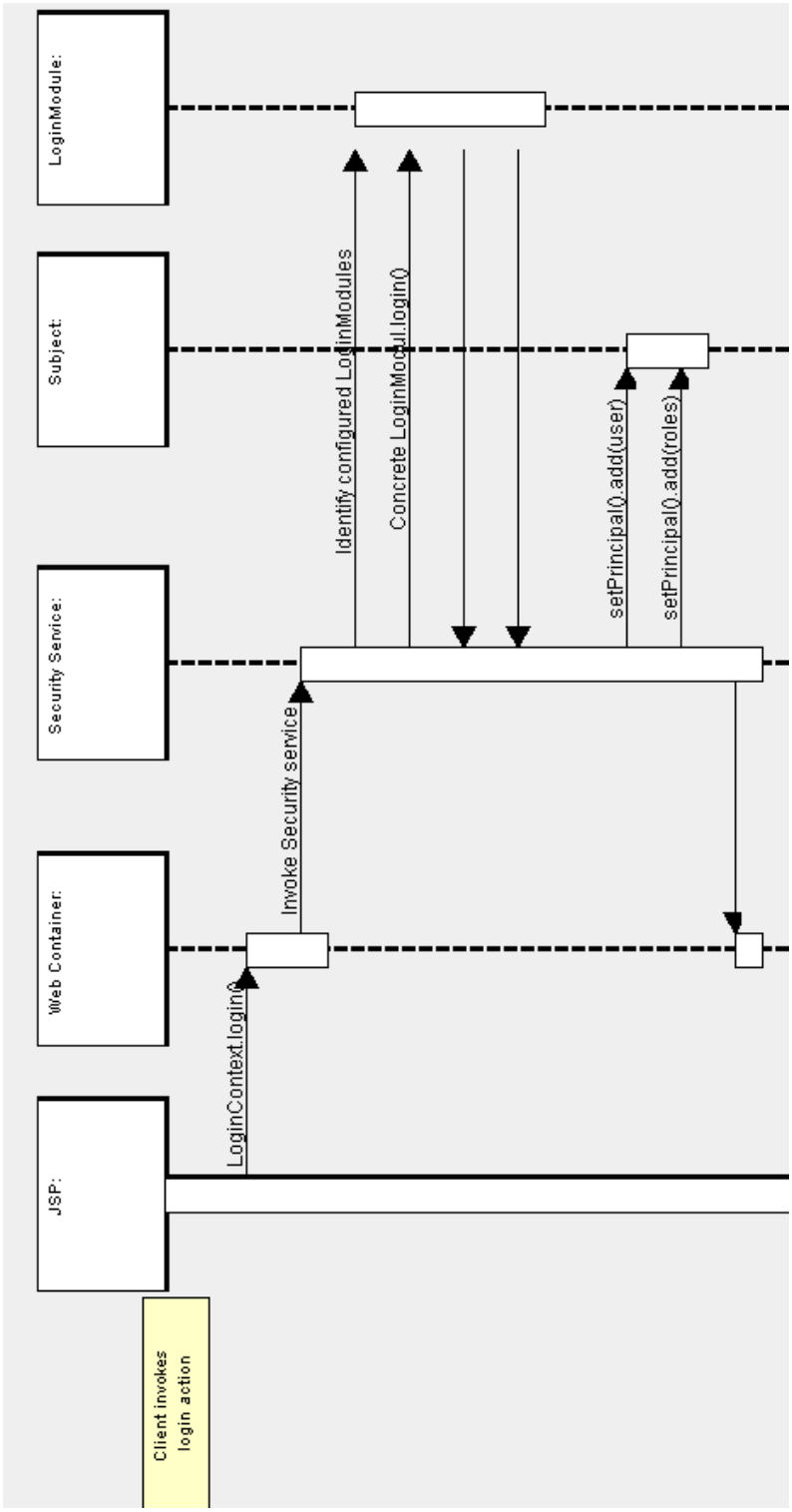
Struts 1 and Struts 2 comparison

Feature	Struts 1	Struts 2
Action classes	Struts 1 requires Action classes to extend an abstract base class. A common problem in Struts 1 is programming to abstract classes instead of interfaces.	An Struts 2 Action *may* implement an `Action` interface, along with other interfaces to enable optional and custom services. Struts 2 provides a base ActionSupport class to implement commonly used interfaces. Albeit, the Action interface is **not** required. Any POJO object with a `execute` signature can be used as an Struts 2 Action object.
Threading Model	Struts 1 Actions are singletons and must be thread-safe since there will only be one instance of a class to handle all requests for that Action. The singleton strategy places restrictions on what can be done with Struts 1 Actions and requires extra care to develop. Action resources must be thread-safe or synchronized.	Struts 2 Action objects are instantiated for each request, so there are no thread-safety issues. (In practice, servlet containers generate many throw-away objects per request, and one more object does not impose a performance penalty or impact garbage collection.)

Servlet Dependency	Struts 1 Actions have dependencies on the servlet API since the HttpServletRequest and HttpServletResponse is passed to the `execute` method when an Action is invoked.	Struts 2 Actions are not coupled to a container. Most often the servlet contexts are represented as simple Maps, allowing Actions to be tested in isolation. Struts 2 Actions can still access the original request and response, if required. However, other architectural elements reduce or eliminate the need to access the HttpServetRequest or HttpServletResponse directly.
Testability	A major hurdle to testing Struts 1 Actions is that the `execute` method exposes the Servlet API. A third-party extension, Struts TestCase, offers a set of mock object for Struts 1.	Struts 2 Actions can be tested by instantiating the Action, setting properties, and invoking methods. Dependency Injection support also makes testing simpler.
Harvesting Input	Struts 1 uses an ActionForm object to capture input. Like Actions, all ActionForms must extend a base class. Since other JavaBeans cannot be used as ActionForms, developers often create redundant classes to capture input. DynaBeans can used as an alternative to creating conventional ActionForm classes, but, here too, developers may be redescribing existing JavaBeans.	Struts 2 uses Action properties as input properties, eliminating the need for a second input object. Input properties may be rich object types which may have their own properties. The Action properties can be accessed from the web page via the taglibs. Struts 2 also supports the ActionForm pattern, as well as POJO form objects and POJO Actions. Rich object types, including business or domain objects, can be used as input/output objects. The ModelDriven feature simplifies tag references to POJO input

		objects.
Expression Language	Struts 1 integrates with JSTL, so it uses the JSTL EL. The EL has basic object graph traversal, but relatively weak collection and indexed property support.	Struts 2 can use JSTL, but the framework also supports a more powerful and flexible expression language called "Object Graph Notation Language" (OGNL).
Binding values into views	Struts 1 uses the standard JSP mechanism for binding objects into the page context for access.	Struts 2 uses a "ValueStack" technology so that the taglibs can access values without coupling your view to the object type it is rendering. The ValueStack strategy allows reuse of views across a range of types which may have the same property name but different property types.
Type Conversion	Struts 1 ActionForm properties are usually all Strings. Struts 1 uses Commons-Beanutils for type conversion. Converters are per-class, and not configurable per instance.	Struts 2 uses OGNL for type conversion. The framework includes converters for basic and common object types and primitives.
Validation	Struts 1 supports manual validation via a `validate` method on the ActionForm, or through an extension to the Commons Validator. Classes can have different validation contexts for the same class, but cannot chain to validations on sub-objects.	Struts 2 supports manual validation via the `validate` method and the XWork Validation framework. The Xwork Validation Framework supports chaining validation into sub-properties using the validations defined for the properties class type and the validation context.
Control Of Action Execution	Struts 1 supports separate Request Processors (lifecycles) for each module, but all the Actions in the module must share the same lifecycle.	Struts 2 supports creating different lifecycles on a per Action basis via Interceptor Stacks. Custom stacks can be created and used with different Actions, as needed.

Appendix D
JAAS Sequence Diagram



Appendix E

SSL Configuration

```
   echo off
   set KEY_TOOL_HOME=<path_to_java_keytool>
   set KEY_FOLDER=<path_to_folder_for_certificate>
   set ALIAS=oldes
   set PASS=nitpass
   rem creates a key pair (public/private key) for the server
   %KEY_TOOL_HOME%\keytool -genkey -keyalg RSA -dname "cn=nit,o=CTU,
l=Prague, c=CZ" -alias %ALIAS% -storepass %PASS% -keystore %KEY_FOLDER
%\oldes.ks
   rem exports the certificate (DER format) - n order to make the client
trust the server
   %KEY_TOOL_HOME%\keytool -export -keystore %KEY_FOLDER%\oldes.ks -alias
%ALIAS% -storepass %PASS% -file %KEY_FOLDER%\oldes.cer
   rem we don't want to share the server keystore file itself with
arbitrary clients, because it holds the private key.
   rem instead we create a separate keystore, we import the certificate
*.cer like this
   %KEY_TOOL_HOME%\keytool -import -file %KEY_FOLDER%\oldes.cer -alias
%ALIAS% -storepass %PASS% -keystore %KEY_FOLDER%\oldes_trusted.ks
```

Example E.1: Certificate creation - script

- `<path_to_java_keytool>` is typically *%JAVA_HOME%/bin/*
- `<path_to_folder_for_certificate>` is source of folder where the certificate will be placed

Tomcat SSL support - uncomment the *SSL HTTP/1.1 Connector* entry in *<apache_tomcat_home_folder>/conf/server.xml*.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    keystoreFile="<path_to_folder_for_certificate>\oldes.ks"
    keystorePass="nitpass"
    clientAuth="false" sslProtocol="TLS" />
```

Example E.2 - Tomcat SSL support. Connector from `server.xml`

Appendix F

User Guides of Web Portals

F.1 OLDES Web Medical Portal

This document describes in detail the functionalities of OLDES Web Medical Interface and serves as user guide.

F.1.1 Login Page - Authentication

This page (see *Figure F.1*) is the unique entry point to the web portal for doctors. According to the login/password information provided by the user (doctor) a credential is generated which allows or disallows the user to access the functionalities of the web portal. Without being identified using this page, the user is not able to access any of the functionalities of the web portal.

Figure F.1: Login page

This page contains the following functional elements:

- Username: The user enters in this text box his valid user name
- Password: The user enters in this text box his valid password
- Login button: This button executes the authentication process. If the login information provided by the user are correct the user is redirected to next page. It these information are not correct the login page is displayed again with an error message and the user is not able to access the other pages and functionalities of the portal.

F.1.1.1 Authorization

Each user is casted at least in one role, but also can be casted in more than one role. Three types of roles are now supported – **anonymous, user, admin**.

- *anonymous* is each user which is not logged in the application and has rights only for displaying page with login form and performing authorization process. In case that a user wants to perform some action for which does not have rights, the error page with message about authorization failing is displayed.
- *user* - this role is the most common within the portal. Each doctor which is allowed to view data about patients, must be cast in the *user* role.
- *admin* - the administrator role has access to the OLDES admin portal where the user can managed patients data. This functionalities were demanded for pilot testing to make patients managing more simple.

F.1.2 Common Functionalities

F.1.2.1 Page Header

After successful authentication process the user can invokes any of the available actions. Each page contains common header where several links to user settings takes place – see Figure F.2.

Figure F.2: Common Page Header

In the concrete:

- Top-left part contains three links (cz, en, it) to switch language of the page for the whole session. User can choose between Czech, English and Italian.
- Top-right part contains one link with user name (e.g. Jan Novak) which redirects to Persona; Setting Page. Next to this, you can find link to logout current user from the session. The link navigates to Login Page.

Below these links, page heading is placed (OLDES Web Portal for Doctors). The last part contains main navigation bar with links to actions which varied according to current user's role.

F.1.2.2 Personal Settings Page

In this page each user/doctor can change his/her username and password for login to web portal (see *Figure F.3*).

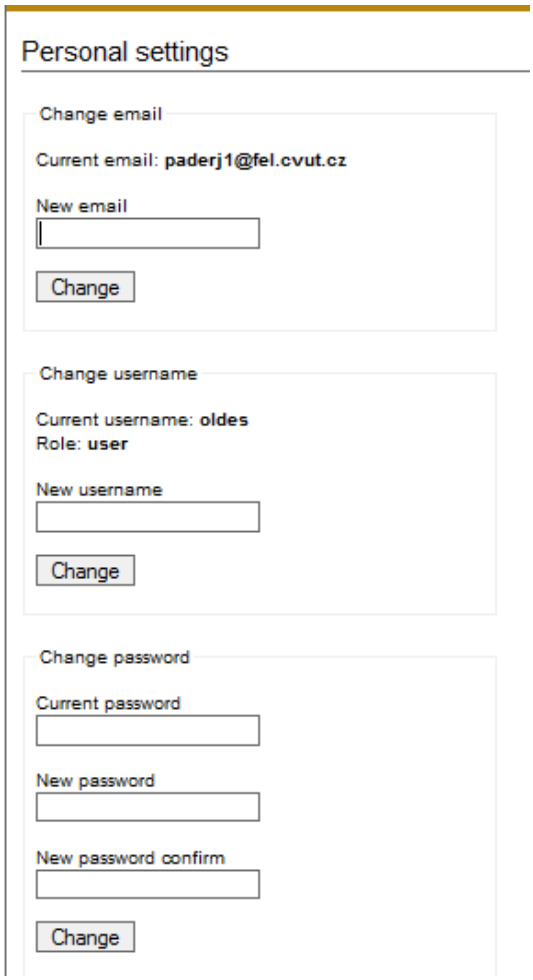


Figure F.3: Personal Settings Page

This page contains the following functional elements:

- Current username.
- New username: The user enters his new username in this text box.
- Change button (in *Change username group*): Performs username change.
- Current password: The user enters in this text box his current password.
- New password: The user enters his new password in this text box.
- New password confirm: The user enters his new password again in this text box.
- Change button (in *Change password group*): Performs password change.

F.1.3 *user* Role Functionalities

F.1.3.1 List of Patients

This action is performed directly after the correct login (as user role) and contains list of patients (see Figure F.4) for current user/doctor. This action can also invoked by clicking on link PATIENTS on main navigation bar.

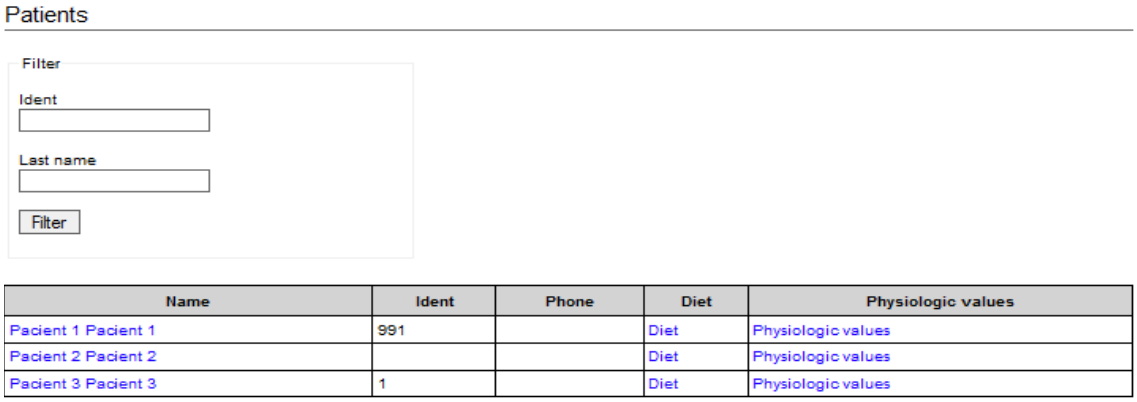


Figure F.4: List of Patients

Body of this page can be divided into two parts. The first part is used for filtering the list of patients. Each doctor can treat a lot of patients and thanks to this use case he is able to find concrete person more quickly. The user can filter the list according to patient's Identification Number or his/her Surname. The filtering is processed after the Filter button is clicked. The user can switch the filtering off by using link (*Switch Filtering Off*) - the link is occurred above the table with patients when the filtering is on.

The most important part of the page body is table with list of patients. The table contains six columns:

- Name of the patient. This name is link which navigates to patient's detailed information
- Identification number of the patient
- Phone number
- Link to food consumption history
- Link to physiologic values browsing

F.1.3.2 Patient's detail

When patient's detail is displayed (by clicking on any of link from the table with list of patients) – see Figure F.5 - another navigation bar is available. This navigation provides actions which are related to currently selected patient.

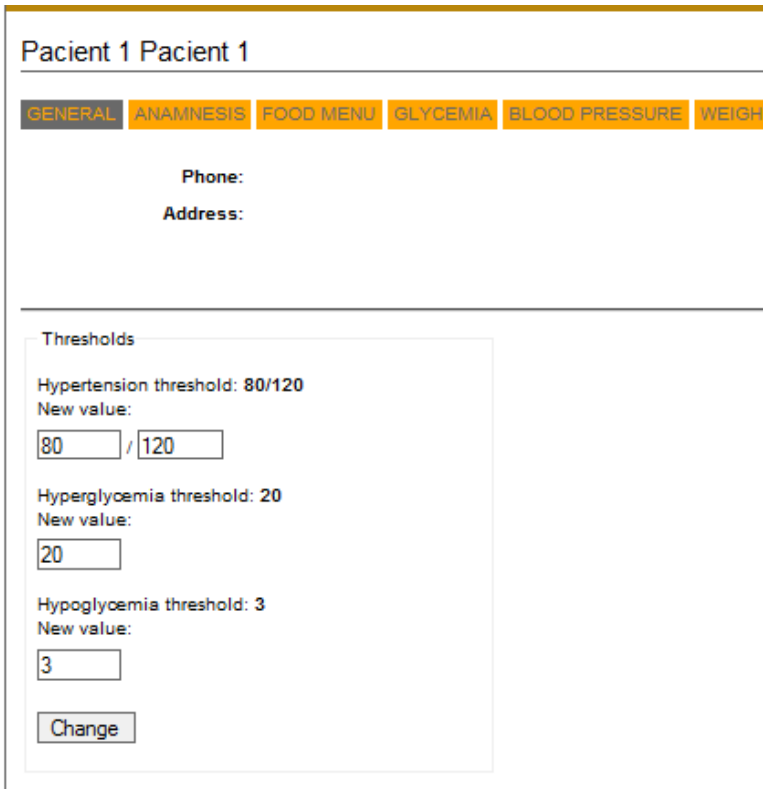


Figure F.5: General Information about the Patient

List of the links:

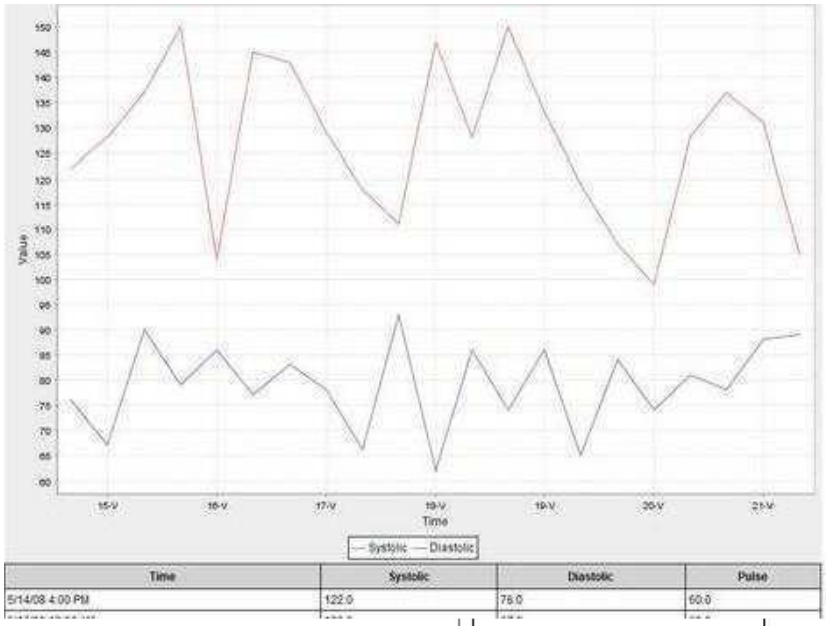
- GENERAL – shows patient's phone and address. In the bottom of the page are listed values of the alarm thresholds. New values of the thresholds can be set at the same place
- ANAMNESIS – shows patient's anamnesis
- FOOD MENU – displays history of patient's food consumption - see Figure F.6 - with daily and menu summary

Nov 19, 2008 8:21:33 PM - Dec 19, 2008 8:21:33 PM << >> month | week | day | 8-hours

Time	Menu	Composite	Name of food	Energy [kJ]	Protein	Fat	Sacharides	Weight [g]	Unit	Description
Daily summary - 11/25/08				26633.0	12.0	4.3	771.30005	-	23.58	
11/25/08	Menu summary - přesnídávka (3)			504.0	0.0	0.0	30.0	-	1.0	
11/25/08 AM 12:00	přesnídávka (3)	in one piece	Presnidavka DIA 275g S	504.0	0.0	0.0	30.0	1.0 portion	1.0	VFN - komplet
11/25/08	Menu summary - přesnídávka (2)			210.0	0.0	0.0	15.0	-	1.0	
11/25/08 AM 12:00	přesnídávka (2)	in one piece	Presnidavka DIA 175g S	210.0	0.0	0.0	15.0	1.0 portion	1.0	VFN - komplet
11/25/08	Menu summary - oběd (1)			2897.0	0.3	0.3	87.0	-	2.2	
11/25/08 AM 12:00	oběd (1)	in one piece	Obed DIA 225g S	2499.0	0.0	0.0	64.0	0.75 portion	1.0	VFN - komplet
11/25/08 AM 12:00	oběd (1)	weighted	Banan	398.0	0.3	0.3	23.0	120.0	1.2	

Figure F.6: Food Consumption

- GLYCEMIA - contains values of glycemia which were measured in the specific time period. Values for glycemia view are displayed via chart and in table.
- BLOOD PRESSURE – displays values of systolic and diastolic blood pressure in chart and table representation. In the table is added column with pulse - see Figure F.7.
- WEIGHT – displays chart and table with measured weight for specific time range.



Time	Systolic	Diastolic	Pulse
5/14/08 4:00 PM	122.0	76.0	60.0

Figure F.7: Blood Pressure

F.1.3.3 Time Range Bar

Patient's detail is divided in few parts as we discussed above. Each of listed part contains special navigation bar for time range specification: food menu, glycemia, blood pressure and weight. The navigation bar provides changing of displayed time period.

The navigation bar is composed from three elements (see Figure F.8):

1. Current time period which is displayed
2. Links (<<, >>) which enables shifting of time period
3. Different view modes. Users can choose how long period should be displayed. Selected view mode is highlighted.

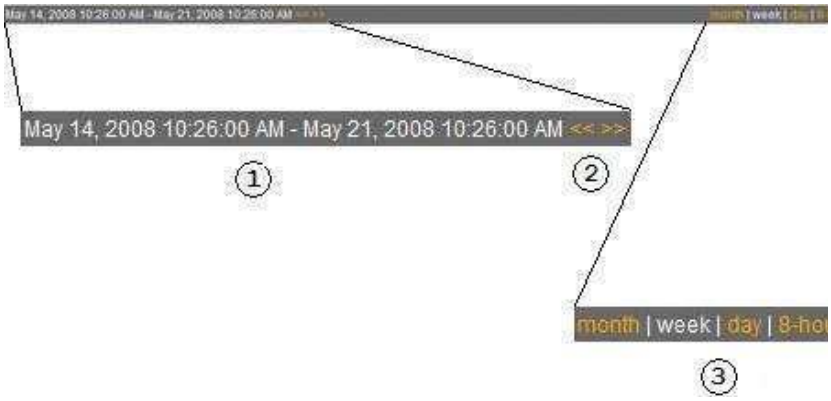


Figure F.8: Time Range Bar

F.1.3.4 Export Bar

Under the Time Range Bar is placed another bar with two links which serves to exporting displayed chart in CSV or PDF format.

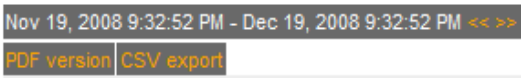


Figure F.9: Time range with export bar

F.1.3.5 Food Menu

Second link in the main navigation bar redirects to list of food items with their category, energy, weight, proteins, fat and sacharides.

F.1.3.6 Alarms

This link is present in the main navigation bar in case that an unsolved alarm exist for doctor's (user's) patient. The view contains table with all unsolved alarms and provides possibility to mark an alarm as solved.

F.1.4 admin Role Functionalities

F.1.4.1 Add Patient

The link Add Patient is presented for *admin* role in the main navigation bar and serves to inserting new patient to the system. The view contains form to input information about new patient:

- List of doctors to select one which will be responsible for treatment of new patient.
- Field for personal identification number of the patient.
- Fields for First name and Last name
- Field for date of birth input
- Fields for address and phone number
- Text area for his/her anamnesis input

After clicking the submit button, new patient is inserted and his/her id from DB is written in the top of the page.

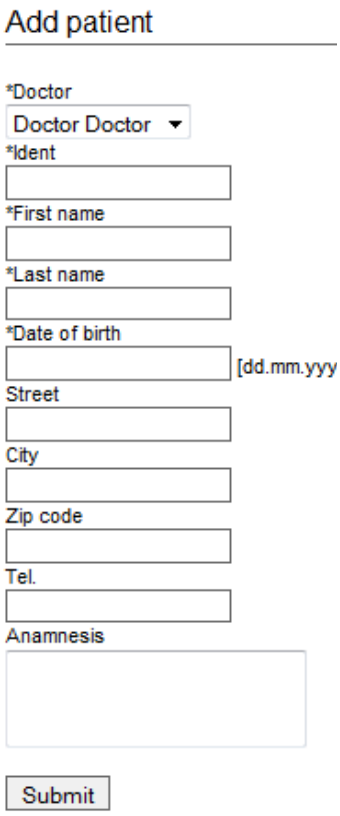


Figure F.10: Add Patient Form

F.1.4.2 Remove Patient

Next link from main navigation bar is Remove Patient's Data. This use-case provides to the administrator list box with all patients. The selected patient can be removed completely from the system (when check box Completely remove patient is ticked) or his/her physiologic and consumption values only.

F.2 Bipometrics Psychiatrist Web Portal

This document serves as user reference guide for Bipometrics Psychiatrist Portal.

F.2.1 Login Page - Authentication

This page (see *Figure F.11*) is the unique entry point to the web portal for doctors. According to the login/password information provided by the user (doctor) a credential is generated which allows or disallows the user to access the functionalities of the web portal. Without being identified using this page, the user is not able to access any of the functionalities of the web portal.

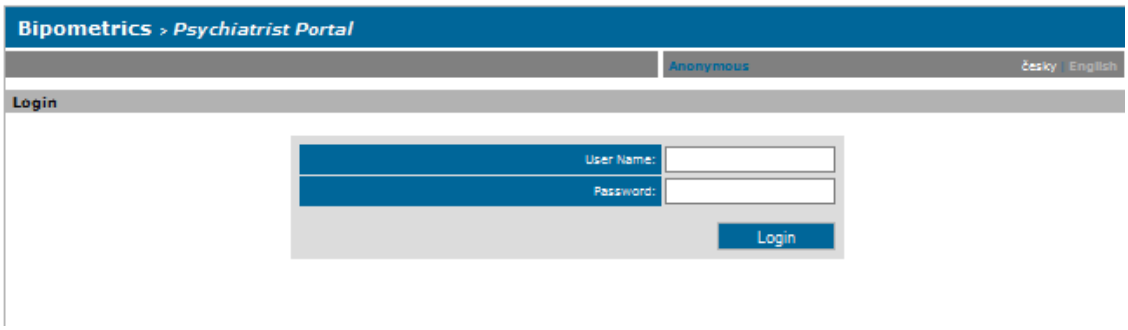


Figure F.11: Login Page

This page contains the following functional elements:

- Username: The user enters in this text box his valid user name
- Password: The user enters in this text box his valid password
- Login button: This button executes the authentication process. If the login information provided by the user are correct the user is redirected to next page. It these information are not correct the login page is displayed again with an error message and the user is not able to access the other pages and functionalities of the portal.

F.2.1.1 Authorization

Each user is casted at least in one role, but also can be casted in more than one role. Three types of roles are now supported – **anonymous, user, admin**.

- *anonymous* is each user which is not logged in the application and has rights only for displaying page with login form and performing authorization process. In case that a user wants to perform some action for which does not have rights, the error page with message about authorization failing is displayed.
- *user* - this role is the most common within the portal. Each doctor which is allowed to view data about patient's condition, must be cast in the *user* role.
- *admin* - the administrator role can perform actions which are not allowed to doctors, but manages the system – e.g. data uploading.

F.2.2 Common Functionalities

The portal several functionalities which are available for all types of users. Commonly are these action accessible from page header which is almost the same for all roles (*user* and *admin*). The Figure F.12 shows left part of the page header. The grey bar is main navigation (for user role in the example) where links varied according to current role. Heading (Bipometrics) is a link to the login page.

Figure F.12: Left Part of the Common Page Header

The right part of the page header – see Figure F.13 - offers language switching (Czech, English) for current session, logout action and link (with username) to user detail.

Figure F.13: Right Part of the Common Page Header

F.2.1.1 User detail

This action displays basic information (name, e-mail etc.) about the current user.

F.2.1.2 Intro

The default action which is performed after successful login and displays short information text about the project. The action is also available from main navigation bar (link Intro).

F.2.2 *user* Role Functionalities

Main actions are provided by the main navigation bar (see Figure F.12).

F.2.2.1 Add hospitalization/depression/mania

These actions serves to doctor to insert new record about concrete patient's hospitalization/mania/depression.

All these three use-cases are very similar and has same view – see Figure F.14 with Add hospitalization action.

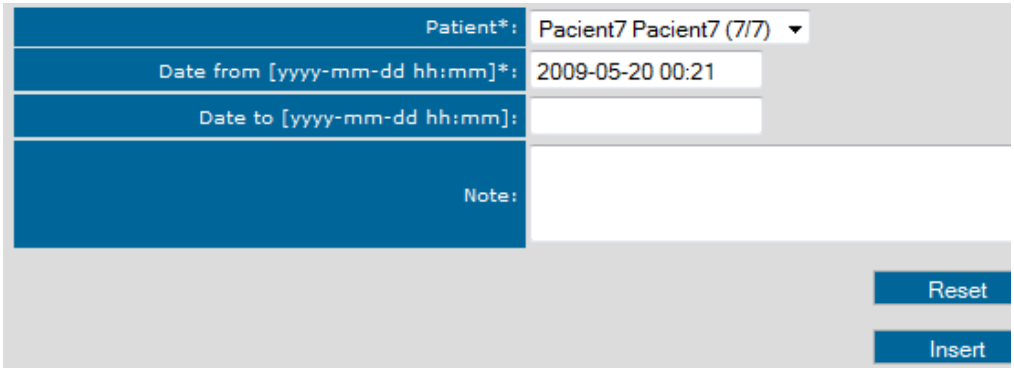


Figure F.14: Form for inserting hospitalization/mania/depression

F.2.2.2 Alarms

The link to list of alarms is presented only if an unsolved alarm record exists for some patient which is related to logged doctor.

The action view contains list of alarms and provides links to mark alarm item as solved or remove the item. The alarm can be removed only if was marked as solved.

F.2.2.3 List of patients

Use-case List of patients is invoked by clicking Patients link from the main navigation bar. The list of patients offers several functions which are always related to selected patient:

- General information – by clicking on the patient name in the list, general information (name, email, family member name) is displayed.
- Hospitalization/Depression/Mania – each of this action is used to display basic table with hospitalization/depression/mania records. Above the table is link to inserting new items.

- Questionnaire - this link redirects to overview of patient's answers which are displayed in chart and table representation. Data from table can be export to CSV format and chart to PDF document or PNG image. Above the chart take place special panel (Figure F.15) where user defines displaying of data.

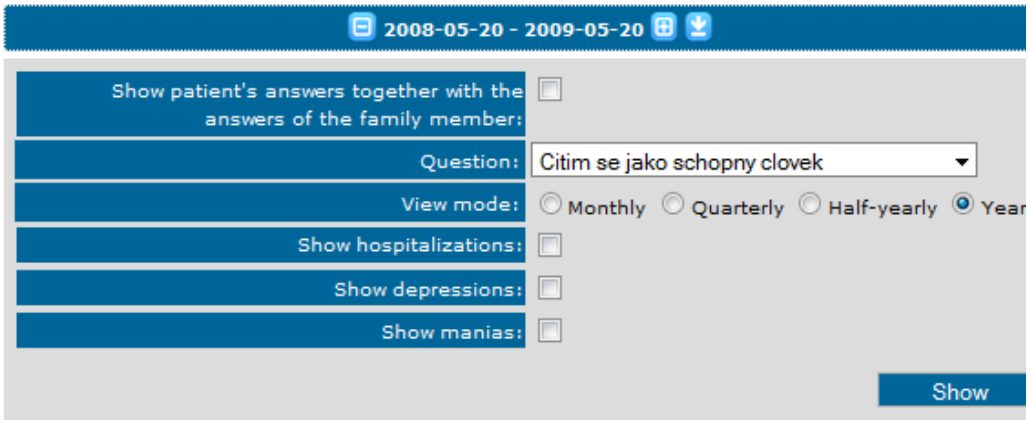


Figure F.15: Panel to Specify View Mode

The top contains time range which can be shift to the left (-) or right (+). The button with arrow shifts time range to the current date. The first check box sets whether user wants to observe friend's answers together with patient's ones. User can choose concrete question from the list box where one option is to select all questions. With specification of the time range is related view mode.The chart can also contain time of hospitalizations/depressions/manias.

- Actigraph – the action lists measurements of the activity. Each measurement can be displayed via chart. The panel (Figure F.16) serves to definition of the chart display in the same sense like the previous panel for questionnaire.

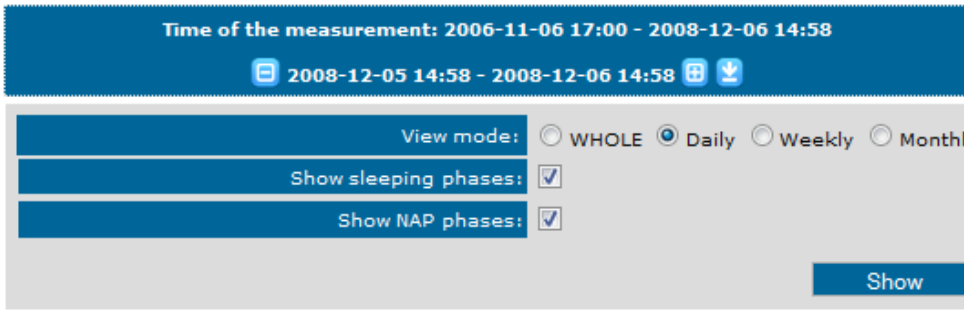


Figure F.16: Panel to Specify Actigraphy View

F.2.3 *admin* role Functionalities

F.2.3.1 Answer insert

Administrator is allowed to add new patient's and friend's answers to a question. The form for the answer inserting is shown below – see Figure F.17

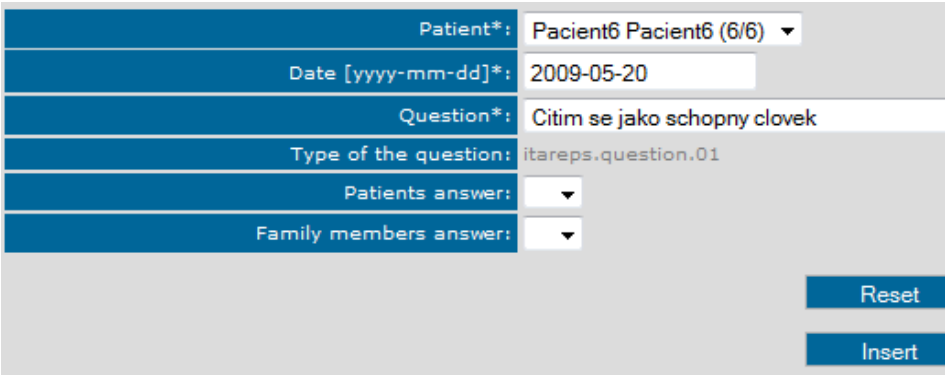


Figure F.17: Insert Answer Form

Administrator has to choose patient and question from the list boxes, fill in the right date and select at least one answer. If the answer for the question and date have already existed in DB administrator is prompt to edit the inputs.

F.2.3.2 Questionnaire insert

This use-case uploads selected XML file (Example F.1) which contains answers from a questionnaire. The file can be used to inserting 1 or more questionnaires for patients. Each questionnaire is identified by patient's personal identification number. After the file processing user is informed whether the uploading was successful, any duplicates occurred or an error arose. In case of any duplicate records, the user can replace values in DB with new ones or skip the replacing.

```
  <questionnaires>
    <questionnaire patient="1/1">
      <answer question='itareps.question.01' date='2006-10-19'
patientValue='7' />
      <answer question='itareps.question.02' date='2006-10-19'
patientValue='7' friendValue=6 />
      <answer question='itareps.question.03' date='2006-10-19'
patientValue='2' />
  ....
   <questionnaire>
  ...
  <questionnaires>
```

Example F.1: XML File to Insert Answers

F.2.3.3 Actigraphy insert

The last functionality for administrator is actigraphy values uploading. This process is made on the same basis as previous questionnaire inserting - special XML file (Example F.2) is uploaded to the system.

```
  <actigraphy patient="1/1">
    <measurement startDate="2006-11-06" startTime="17:00:00"
endDate="2008-12-06" endTime="14:58:22" sampling_rate="16">
      <sleep startDate="2006-11-06" startTime="23:24:00"
endDate="2006-11-07" endTime="08:12:00" />
      <sleep startDate="2006-11-07" startTime="23:24:00"
endDate="2006-11-08" endTime="07:24:00" />
  <nap startDate="2006-11-08" startTime="23:24:00" endDate="2006-11-09"
endTime="07:24:00" />
  <sample value="69" />
      <sample value="302" />
      <sample value="257" />
      <sample value="259" />
      <sample value="373" />
  ...
  <actigraphy>
```

Example F.2: XML File to Insert Data from Activity Measurement

Each file contains data for one patient who is identified by his/her personal identification number. The measurement can contain NAP ans sleep phases together with activity values in sample tags.