



JAGIELLONIAN UNIVERSITY
IN KRAKÓW

FACULTY OF PHYSICS, ASTRONOMY
AND APPLIED COMPUTER SCIENCE

Mateusz Haber

Album number: 1063770

**Development and implementation
of Electronic Logbook for J-PET research group
using Symphony2 framework**

Diploma thesis
in the field of Applied Computer Science

Diploma thesis supervised by:
Dr inż. Marcin Zieliński
Nuclear Physics Department

KRAKÓW 2015

Oświadczenie autora pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplmowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami. Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Kraków, dnia

Podpis autora pracy

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Kraków, dnia

Podpis kierującego pracą

Abstract

The main aim of this thesis was to develop and implement an interactive Electronic Logbook for Jagiellonian Positron Emission Tomography research group. Using the modern and innovative Symfony2 framework, Bootstrap, jQuery and Angular.js technologies we have created a fully working client-server web application based on the Model-View-Controller architecture. The data storage was organized using two database engines: MySQL, and PostgreSQL, served in back-end by the Doctrine package. Application was designed in the Responsive Web Design (RWD) approach, making it possible to use it on the any portable devices (e.g. like tablets and smartphones).

Application was prepared according to the functional requirements provided by the J-PET group. The main functionality of the application is a multifunctional and intuitive activity logging system, which allows to register and further monitor the situation of the experimental work. Secondly the application allows to associate the laboratory work information with the experimental parameters available from external database. Furthermore, we have implemented the error and warning handling system, allowing to monitor and report all the custom situations encountered during the laboratory work. Application was supplemented with the Shift Management module, allowing to organize weekly laboratory work of the research group.

Streszczenie

Głównym celem pracy magisterskiej było zaprojektowanie i stworzenie interaktywnego dziennika badawczego dla grupy Pozytonowej Tomografii Emisyjnej J-PET. Korzystając z najnowszych i innowacyjnych technologii internetowych takich jak Symfony 2, Bootstrap, jQuery i Angular.js, stworzono w pełni działającą aplikację o architekturze Klient-Serwer, opartą na wzorcu projektowych Model-Widok-Kontroler (MVC). Dane aplikacji są przechowywane w dwóch niezależnych bazach danych: MySQL oraz PostgreSQL, obsługiwanych przez pakiet Doctrine. W trakcie projektowania i implementacji aplikacji zadbano aby strony były dostępne na dowolnych urządzeniach, w tym mobilnych takich jak tablet lub smartphone.

Aplikacja została przygotowana zgodnie z wymaganiami funkcjonalnymi grupy J-PET. Główną funkcjonalnością aplikacji jest możliwość tworzenia indywidualnych wpisów dziennika badawczego. Dodatkowo każdy wpis dziennika można uzupełnić o dane eksperymentalne pobierane z zewnętrznej usługi bazodanowej. Ważną zaimplementowaną funkcjonalnością jest system zgłaszania błędów i ostrzeżeń, pozwalający na bieżące monitorowanie i raportowanie wszystkich niestandardowych sytuacji zaistniałych w trakcie pracy laboratoryjnej. Aplikacja została również wyposażona w funkcjonalność pozwalającą na zarządzanie godzinami pracy w laboratorium, co przyczynia się do jej wydajnej organizacji.

Contents

1	Introduction	9
2	Motivation and main functionality of the application	11
3	Architecture of the application	13
3.1	MVC and Symfony2 framework	13
3.2	Front-end technologies	16
3.3	Database management systems	19
3.3.1	MySQL	19
3.3.2	PostgreSQL	19
3.4	GIT version control system	20
3.5	Other Open Source libraries	21
4	Implementation of main functionalities of developed Logbook system	23
4.1	Logbook database	23
4.2	User authentication	27
4.3	Main Views of the Logbook	30
4.4	Logbook functionality	32
4.4.1	Adding logbook entry	32
4.4.2	Single logbook view	34
4.4.3	Logbook search module	37
4.5	Experimental error and warning handling system	38
4.6	Shifts managements system	41
4.6.1	Shift planner	41
4.6.2	Shift calendar	42
4.7	Administration panel	43
5	Summary	45
A	Installation of the application	47
	Bibliography	49
	Podziękowania	51

1. Introduction

Nowadays organization and storage of information in electronic form is commonly used by the large corporations and small companies in most industry sector. There is a strong tendency to replace the paper information flow with the electronic version only. Also this trend from many years is observed in field of the research. Currently researchers communicate between each other via electronic mail, dedicated websites or on-line chats. Also, the experimental data is stored on large farms of servers and in the cloud services. Thus, the good organization of the information flow and methods of keeping them are crucial for efficient scientific work.

One of the common thing done by researchers is to record results of the experimental work in the accessible form for other researchers. Up to few years ago very popular way of recording the laboratory activity was to write a short note in the paper notebook. However, as the development of computers and internet technologies has grow also the researchers started to use electronic tools to store laboratory information. One of the reasons to use the electronic form of logbook, was the number of people working at the same time in one group but, not necessary in the same geographical place. However, as the work of each experimental group has it own specific it is hard to use the commercially available software. Therefore, the main aim of this thesis was to design, develop and implement an Electronic Logbook dedicated for Jagiellonian Positron Emission Tomography research group, in order to support and improve the information storage and data flow. Developed application will be used to report, control and store data from laboratory work in fully electronic form. The projected features allow to prepare a single electronic entry with the description of the laboratory work at given time. Also, the developed solution has a feature of managing the time and task in the laboratory.

The system was design to be intuitive, clear and user friendly to working in the Client-Server architecture in the form of the web based application. One of important features was to design it to work on every modern device like PC, tablet and smartphone. The application was created utilizing Symfony2 framework as the back-end¹ technology and Bootstrap, jQuery and Angular.js, as the front-end² technologies. For the data storage MySQL and PostgreSQL databases were used, with the Doctrine tool to support the object-relational mapping (ORM).

In the second chapter of this thesis we will describe motivation and main functionality of implemented application. The third chapter is a detailed introduction to the architecture of

¹Back-end is commonly used in web applications as a name for the technologies used on server side.

²Front-end is commonly used in web applications as a name for the technologies used on client side in the web browser.

the application, supplemented with the description of the utilized back-end, and front-end technologies, as well as the database management systems. In the fourth chapter we will describe the implementation of main functionalities of developed Logbook system. The fifth chapter contains a short summary. The work has one appendix which describes the method of the installation of the application on server side.

2. Motivation and main functionality of the application

One of the most popular diagnostic techniques is Positron Emission Tomography (PET), which relies on a mapping of the spatial distribution of the selected substance in the volume of the body and allows the measurement of the substance concentration changes over time, which will determine the rate of metabolism of the individual tissue cells.

In general, this method allows for non-invasive imaging of physiological processes in the body, but now is primarily the most technologically advanced diagnostic method of early stages of cancer and to determine the degree of cancer malignancy, using the fact that cancer cells show an increase in metabolic activity compared to normal tissue. Accordingly, in the highly-developed countries PET plays an important and unique role in medical diagnostics and monitoring the effects of cancer therapies.

Typical PET detectors are built out of inorganic scintillating detection modules, which enable the registration of the annihilation γ quanta in coincidence. In the newer of PET detectors, the resolution of the tomographic image is improved by the determination of the annihilation point along the line of response (LOR). It is based on measurements of the time difference between the arrivals of the γ quanta to the detectors. This technique is known as time flight (TOF) and improves the reconstruction of PET images by increasing signal-to-noise ratio due to reduction of noise propagation along the LOR during the reconstruction.

Jagiellonian Positron Emission Tomography [1–4] (J-PET) is a research group from the Jagiellonian University in Krakow, working on a novel solution for the construction of Positron Emission Tomography, for the whole human body imaging. J-PET team is developing a novel device which will allow to determine the impact position as well as time of the annihilation of gamma quanta. The device is comprised of scintillation chamber consisting of organic scintillation detectors surrounding the body of the patient (Fig. 2.1). The application of the fast scintillation detectors will enable to take advantage of the difference between time of the registration of the annihilation quanta. The invented method will permit to use a thick layers of detector allowing to determine the Time of Flight for the annihilation gamma quanta. This will allow for increasing the size of the diagnostic chamber without a significant increase of costs.

The J-PET project gathers a large group of scientists working together on construction of the novel diagnostic device. One of the main drawbacks in such projects is the information flow and the logging of all experimental activities. Therefore, in the framework of this master thesis we have develop fully functional Electronic Logbook for J-PET research group.

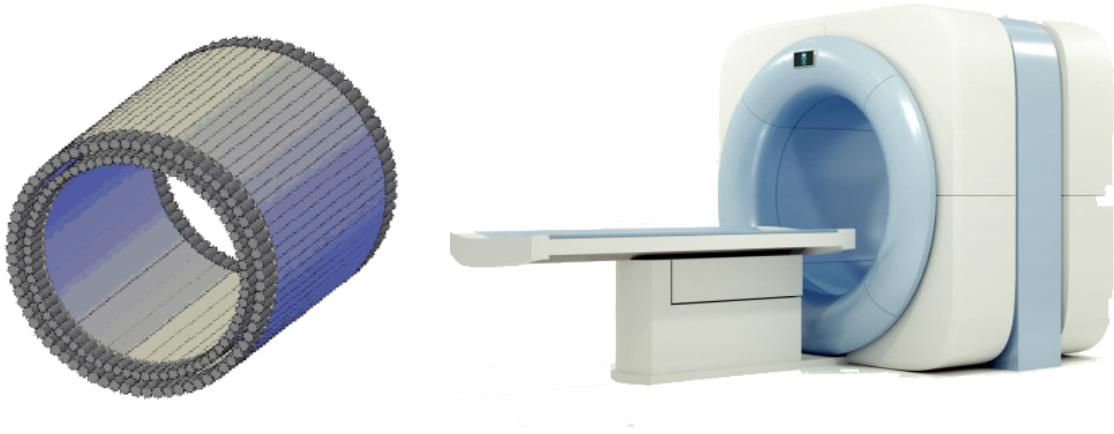


Figure 2.1: J-PET scintillation detectors system: **(left)** - view of the diagnostic chamber consisting of scintillating individual modules with photomultipliers attached at the end of each module [4], **(right)** - visualisation of the J-PET tomography scanner.

Developed solution is a fully WEB2.0 application driven by the users. Proposed solution will replace the currently used system based on open source MediaWiki [5] software.

Commercially, there are many “ready-made” Content Management System (CMS) [6] solutions like e.g. MediaWiki, Wordpress [7], Drupal [8] or Joomla! [9], which could be adopted to be a Logbook system. However, these constitutes a general software which do not posses the dedicated functionality related with the research conducted in the J-PET group. Therefore, we have prepared completely new software system with the functionalities dedicated for the J-PET group.

The main aim of developed application is to facilitate the information flow with in the experimental group and allow to preserve individual information from the experiment in the electronic form. Additionally, system allows to: add, view, delete, and search existing logbook content. All the information are stored in the form of chronological posts with the possibility of attaching media content like e.g. graphics, plots, equations, video. Furthermore, each individual post can be supplemented, if needed, with the experimental data parameters downloaded on-line from external J-PET database. In such way the experimentalist will automatically be able to comment on the current situation in the laboratory providing the actual experimental data. Furthermore the developed system provides standard authentication features enabling access to the content of the logbook to limited authorized users only.

The second important functionality of the developed application is the shift management system which is strictly related to the logbook ability. Shift planner allow users to mark their availability for the laboratory work and help them to maintain efficiently the work. This feature allows to mange duties of the experimental group members in the laboratory.

Finally, the third functionality of developed application is the experimental error and warning handling system. This function allow users to report warning or error during the experiment.

3. Architecture of the application

The aim of the this work was to create an intuitive, efficient and secure web application utilizing the most modern and innovative solutions available in the domain of web development. The main requirement for the design Electronic Logbook, was to make it work in the web browser, independent of the hardware and software platform. Therefore we have utilized the Client-Server application model, with the communication between two parts using the HTTP (Hypertext Transfer Protocol) protocol. The simple scheme of the application model is presented on Fig. 3.1. As a core of developed application on the server side (back-end), we have used the fully scalable and object oriented Symfony2 framework [10] with the architecture based on the most reliable and efficient Model-View-Controller (MVC) design pattern [11]. The Symfony2 framework was supported by the Doctrine package for the database management, and with the TWIG templates to facilitate the view preparation. On the Client side (front-end) for the presentation and data processing, we have used the Twitter Bootstrap, Angular.js and jQuery libraries.

In this chapter we will introduce all the technologies utilized during the implementation of the application in the context of the used design pattern.

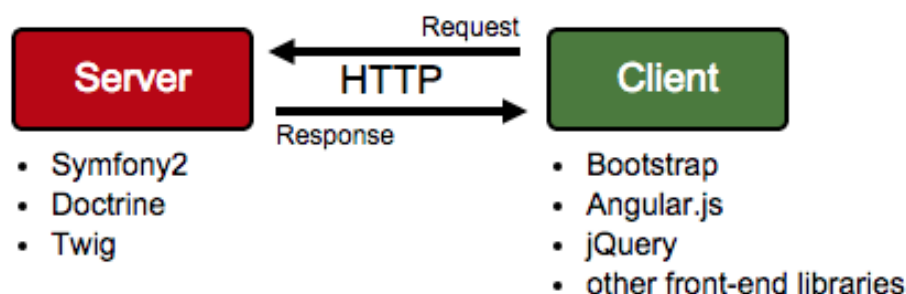


Figure 3.1: Scheme of the architecture of the Logbook application in view of the communication and Client-Server technologies.

3.1 MVC and Symfony2 framework

Developed web application was prepared using the Symfony2, which is a free, fully object oriented PHP5 [12], released under the MIT [13] licence, developers framework. Symfony2 was created by the SensioLabs [14] company in 2005 year. This framework supports fully the MVC design patten making the development of web application much more flexible and efficient. Additionally, this programming environment is supported by numerous developers community which is a very important feature.

Symfony2 speeds up the process of building objected oriented application, assuring at the same time full security of software and data. It automatizes many of the operations like entity creation, and object-relational-mapping (ORM) [15]. It uses many of the PHP Open Source [16] projects as a part of itself. It utilizes, framework Propel [17] and Doctrine [18] as object relational mapping layers and to build database abstraction level. To build fast and flexible views Symfony2 uses TWIG [19] as a template engine. Software also uses PHPUnit [20] for the unit tests.

Symfony2 is based on the Model-View-Controller (MVC), one of the most popular and widely used design patterns in most of the web and native applications. In this approach, MVC pattern forces partition of software into three interconnected parts:

- **Model** represents the application logic, which means that it stores and specifies the methods of operation on data. The data from the Model can be retrieved by the Controller and then displayed by the View layer. Model does not know about the existence of Controller and View.
- **View** receives data sent by the Controller, requested from the Model, and generates user output with the data. One can say that the View is the graphical representation of Model. However it is not associated with the way of presentation, and therefore it not entails any changes to the Model.
- **Controller** responds to user actions, often triggered by links in the View. The main role of the Controller is to be responsible for the changes of the state of the Model and refreshing the View.

In the MVC approach firstly User is taking an action, which then triggers the method in the Controller. Then the controller communicates with the Model to acquire data, which is

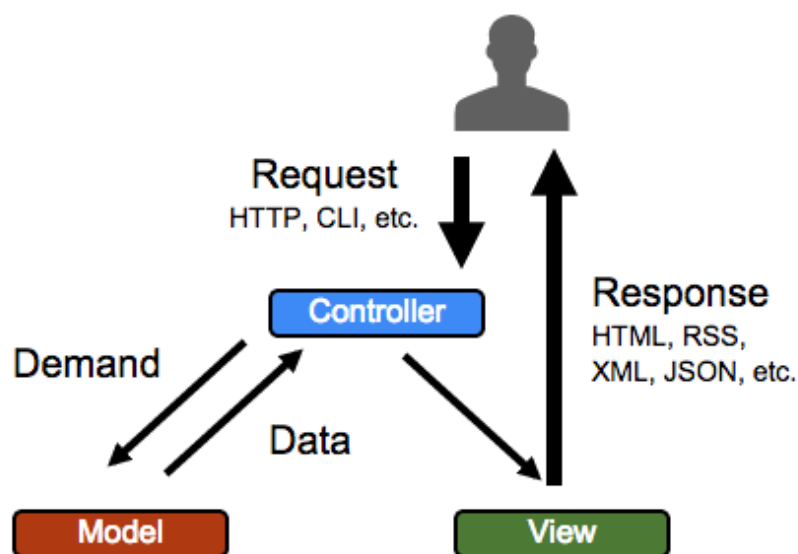


Figure 3.2: The basic scheme of the three layer MVC based application.

sent to the View to be presented to the User. The basic scheme of the MVC pattern with the information flow between layers is shown in Fig. 3.2.

In Symfony2, Doctrine entities and Twig templates match MVC Model and View partition. Every user request to the server is processed by the HTTP kernel, and according to the defined route¹ determines the active controller. Than controller on demand downloads the data from Doctrine entities (Model) and displays them by the Twig template engine (View). The developed application was prepared taking into account all the rules of the MVC design pattern. Figure 3.3 shows the partition of the Logbook application files according to the function held in the MVC pattern.

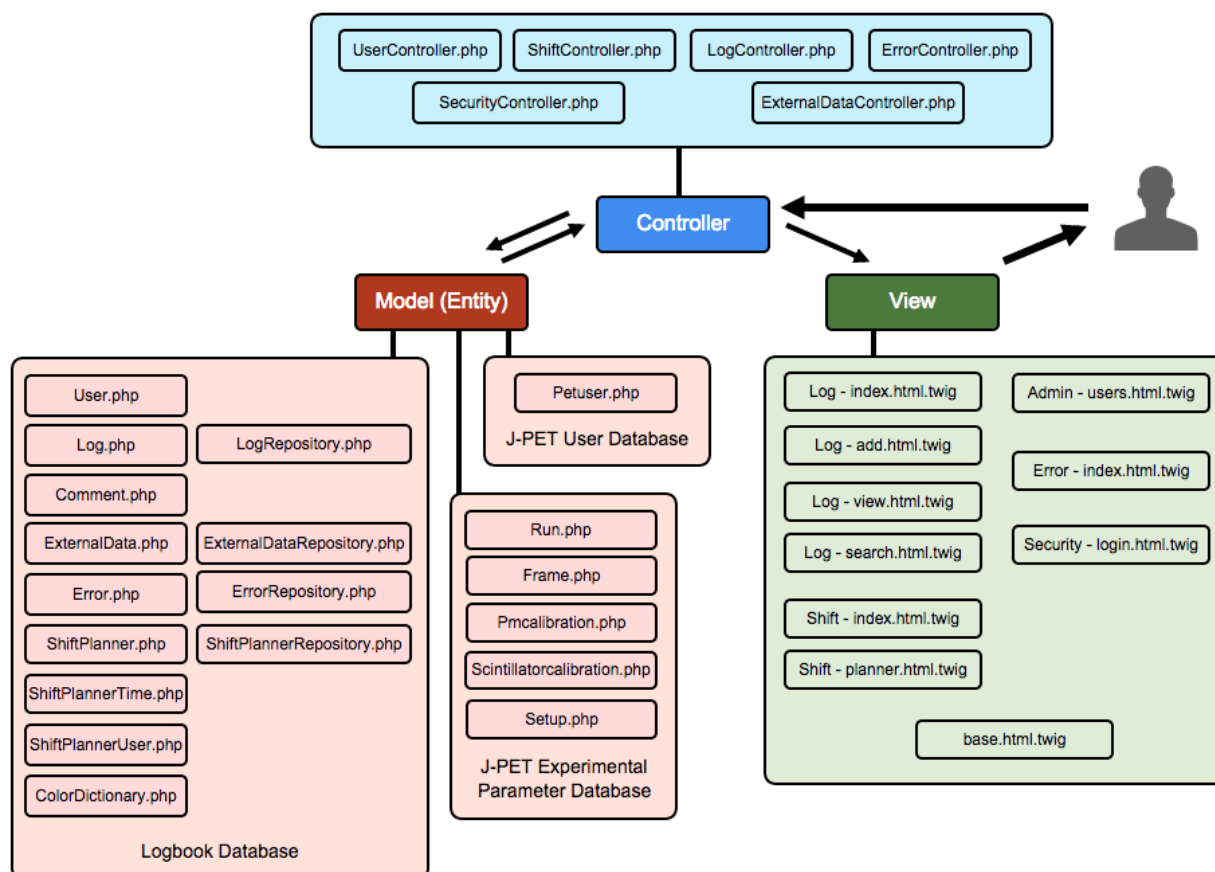


Figure 3.3: Representation and partition of the Logbook application files according to the MVC design pattern. Each small rectangle denotes the name of the file in particular layer.

In the case of developed application for comprehensive support of the communication with database we have chosen to use Doctrine package. It is featured with Doctrine Query Language to write database queries in an object oriented SQL language². One of the useful features of Doctrine, is that it is capable of performing database operations on data like: joins and fetches with related objects automatically, so in most cases writing queries is not required. In the Doctrine approach database structure is described in the application logic by annotation mechanism or by XML [21] and YAML [22] formats. One of the very useful features of Doctrine is the ability to create entity classes based on the existing database

¹All routes are defined in file: `app/conf/routing.yml` available in the in project directory.

²Alternative to this is QueryBuilder class to build queries in an intuitive way.

structure, and conversely, to generate database table structure based on existing entity classes in the application. This property of Symfony2 automatizes many operation related with the database operations.

In the Symfony2 framework the presentation layer is organized using the TWIG template engine. This special meta language is more concise than the pure PHP expressions and the syntax is more human readable and intuitive, extended with shortcuts for common patterns. TWIG was developed with PHP5, therefore it is fully flexible and extensible with user defined tags, filters and domain-specific language (DSL) [23]. Also, it is a very fast, because templates are compiling down to pure PHP code each time the action of rendering a view is requested. Furthermore, TWIG originally was developed by Symfony2 authors, therefore both libraries are fully compatible.

For the interaction of the Client and Server sides we have implemented two different ways of communication: synchronous, and asynchronous. Utilization of these two different solutions allows to improve usability and introduce more intuitive GUI of the application. Synchronously requests are performed each time while user requests action in the MVC controller. In contrast, asynchronously requests are executed on demand by the Angular.js library, without refreshing the web page. The model of used Client-Server communication is shown on Fig. 3.4.

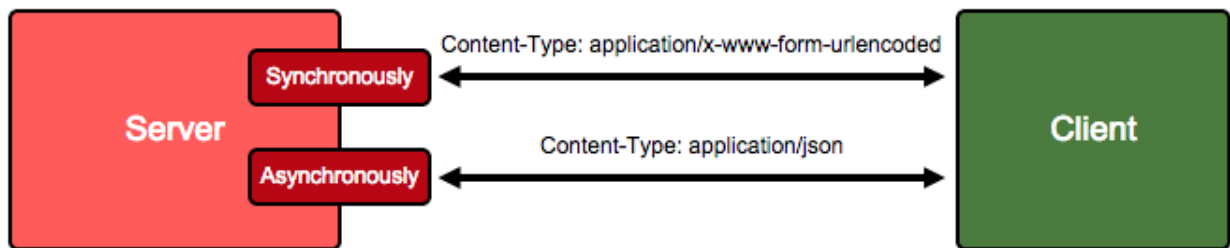


Figure 3.4: Client-server communication in Logbook application.

3.2 Front-end technologies

One of the most important aspects of the application development is to prepare user friendly Graphical User Interface (GUI) [24]. Therefore, developed Electronic Logbook system interface should be characterized by usability, intuitive and easy to handle features. In order to achieve these objectives, we have utilized the newest, Open Source, front-end technology.

In most of modern web application one can not use directly only basic internet technologies such as: HTML5 [25], CSS3 [26] and JavaScript [27] to design the fully dynamic GUI. Therefore, nowadays programmers and developers uses complex libraries based on these technologies to prepare presentation layers. In the developed Logbook application to implement GUI, we have used the Twitter Bootstrap [28], FontAwesome [29], jQuery [30] and Angular.js [31] libraries.

Twitter Bootstrap [28] is very popular HTML, CSS and JavaScript framework created by Twitter [32] company. It is commonly used to prepare responsive³ and modern websites [33]. It contains prepared CSS and JavaScript files which includes predefined: fonts, blocks, panels, forms, and many others graphical features. Bootstrap library enables to prepare easily and efficiently scaled websites for all existing devices, from computer to smartphone (Fig. 3.5). Each element of library can be customize using on-line creator tool available on the Bootstrap web page [34].

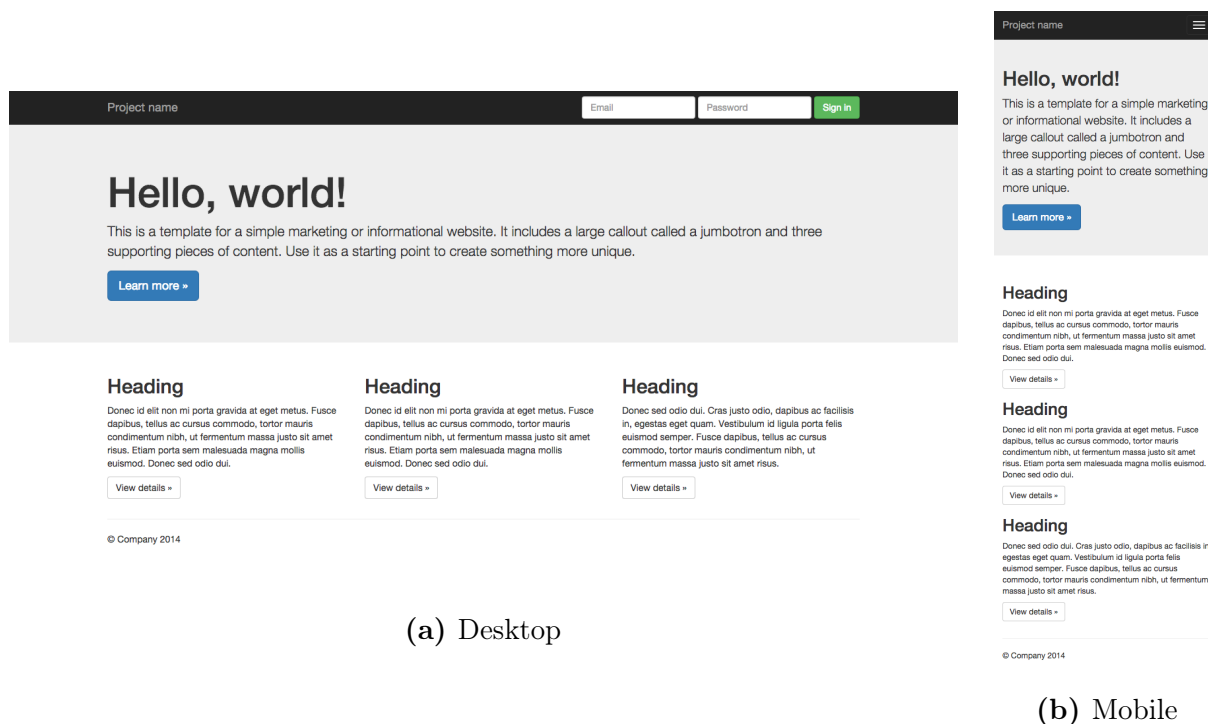


Figure 3.5: Simple "Hello, world!" website, developed with Twitter Bootstrap where (a) shows the layout of the site displayed on standard PC computer with high resolution wide screen, and (b) the same site displayed on the portable device like e.g. smartphone.

Moreover, to achieve a very modern design of the developed application we have used the FontAwesome [29] package which is the iconic font and CSS library specially design to work together with the Bootstrap. With FontAwesome one has access to 519 vector icons that can be freely customize. To use the FontAwesome icons within the website one needs to add a special class attribute in the `<i>` tag starting with the `fa` prefix. Few examples of popular icons together with the code are shown in Fig. 3.6.

To design fully dynamic front-end of the application we have used the jQuery library [30] which is based on the JavaScript language. It was release in 2006 by the Open Source community, under the MIT license. Nowadays, jQuery library is the one of the most popular tool supporting web design used by the largest internet companies like: Wordpress, IBM, Mozilla, Adobe, Intel and MaxCDN [35]. In principle jQuery allows to manipulate HTML DOM⁴ structure and supports event handling. jQuery is browser independent, flexible and

³The term "responsive" (or Responsive Web Design - RWD) is commonly used in the sense of adaptation of the application layout to the screen size of the device on which the application is displayed.

⁴DOM - Document Object Model



Figure 3.6: Examples of FontAwesome usage.

characterized by brevity and clarity, because a large part of functions can be written in a form of chain of objects. Figure 3.7 shows an example of jQuery usage, where $\$$ is a factory method for the jQuery object with chain of functions. As a result shown in this example, each *div* element will be extended by *p* element, with *quote* class, and class attribute “red” will be added to each matched element, together with slowly slide down animation. Nowadays 65% of websites with highest traffic in the Internet uses jQuery. This makes jQuery the most popular JavaScript library in whole Internet.

```
$("div").add("p.quote").addClass("red").slideDown("slow");
```

Figure 3.7: Example of jQuery chain usage.

In order to facilitate some front-end operations we have used also the Angular.js [31] package which is an open source web development framework original designed by Google. The main idea of library is to support creation and development of single-page web applications. In order do this it uses the Model-View-Controller (MVC) and Model-View-View-Model (MVVM) architectures on clients-side.

In the classical website approach, the template systems works with one-way data binding mode as it is shown in Fig. 3.8. This means that after one-time merge, changes to the model are not automatically refreshed in the view. Whereas, Angular.js adopts and extends traditional HTML pages by presenting dynamic content in real-time through two-way data binding mode (Fig. 3.9). In this case any changes in the model are immediately projected to the view and vice versa, any changes in the view are sent to the model.

Angular.js contain $\$scope$ object which refers to the application model. The $\$scope$ service detects changes in both sides of view and model, and promotes changes via controller. $\$scope$ is like observer responding to any changes in the application.

Another Angular.js feature are directives which are HTML-like elements and attributes that define behaviour of DOM elements and responds for data binding. The most popular used directives are e.g.: ng-app, ng-bind, ng-model, ng-model-options, ng-class, ng-controller.

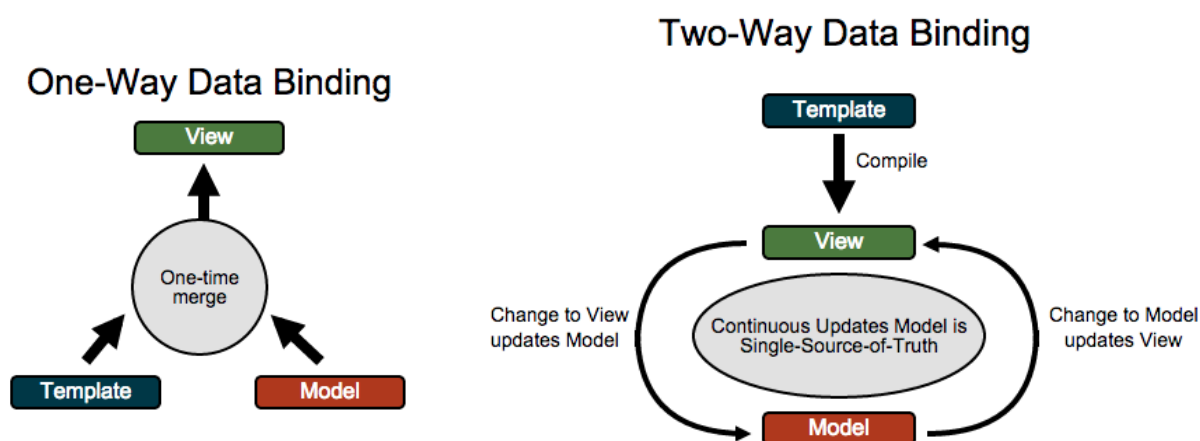


Figure 3.8: Data binding in classical website template systems.

Figure 3.9: Data binding in Angular.js website templates.

3.3 Database management systems

The projected Logbook application could not work without the support of the database management system (DBMS) [36]. These systems enables to store data in the ordered digital form, with the capabilities of adding, modifying and removing existing data. Most famous database engines are MySQL [37], PostgreSQL [38], Microsoft SQL Server [39], Oracle [40], Sybase [41] and IBM DB2 [42]. Since 1980 all of the DBMS supports Structured Query Language (SQL) [43] language. In case of the Logbook application we will work only with MySQL and PostgreSQL systems.

3.3.1 MySQL

MySQL is a relational database management system (RDBMS) [44], which constitutes the most popular database solution for web applications. Because of a very good integration of this database engine with Linux, Apache, and PHP the biggest internet portals like e.g. Facebook, Google Twitter or YouTube are using it to serve data. MySQL is written in C and C++, but works on many system platforms like e.g. AIX, FreeBSD, Linux, OS X, Microsoft Windows, Oracle Solaris, and SunOS. MySQL project is more focused on performance than on compatibility with SQL standard. However, over the years project started to be in agreement with ISO/IEC 9075-1:2003 standard, which supports more primary SQL functions like transactions, stored procedures, triggers, views, cursors, partitioning of tables, schedule of tasks. MySQL technology is now using the newest SQL:2003 [45] standard.

3.3.2 PostgreSQL

PostgreSQL is the second most popular DBMS, specifically object-relational database management system (ORDBMS) [46]. In contrast to MySQL, PostgreSQL is much more consistent with SQL standards, concretely SQL:2011 [47]. It is available for the most of the operating systems, but especially for Linux distributions. The most significant function of

PostgreSQL engine is the multi version concurrency control (MVCC) to manage transactions. In PostgreSQL one can also write predefined rules, which can rewrite incoming query. It also supports much more index types, data types and large sizes of data queries than the MySQL engine. Technology is full of add-ons, scalability and customization to suit developer needs and the application requirements.

3.4 GIT version control system

The whole process of the Logbook application development and implementation was managed and supported by using GIT system. GIT [48] is a application design to safely store and manage the IT project with the ability of controlling the programmers code version. Originally it was developed for Linux to replace Concurrent Versions System (CVS) [49], improving operations atomicity, optimization of the storage space and support for distributed version control system. GIT provides security for single project and comfort for shared projects. In our case, we use GitHub [50] - web-based GIT repository hosting service. GitHub is supported by intuitive and simple GUI to manage whole project commits as it is shown in Fig. 3.10. On the left side of Fig. 3.10 one sees the list of commits, and on the right side the developers changes to the application code are visible.

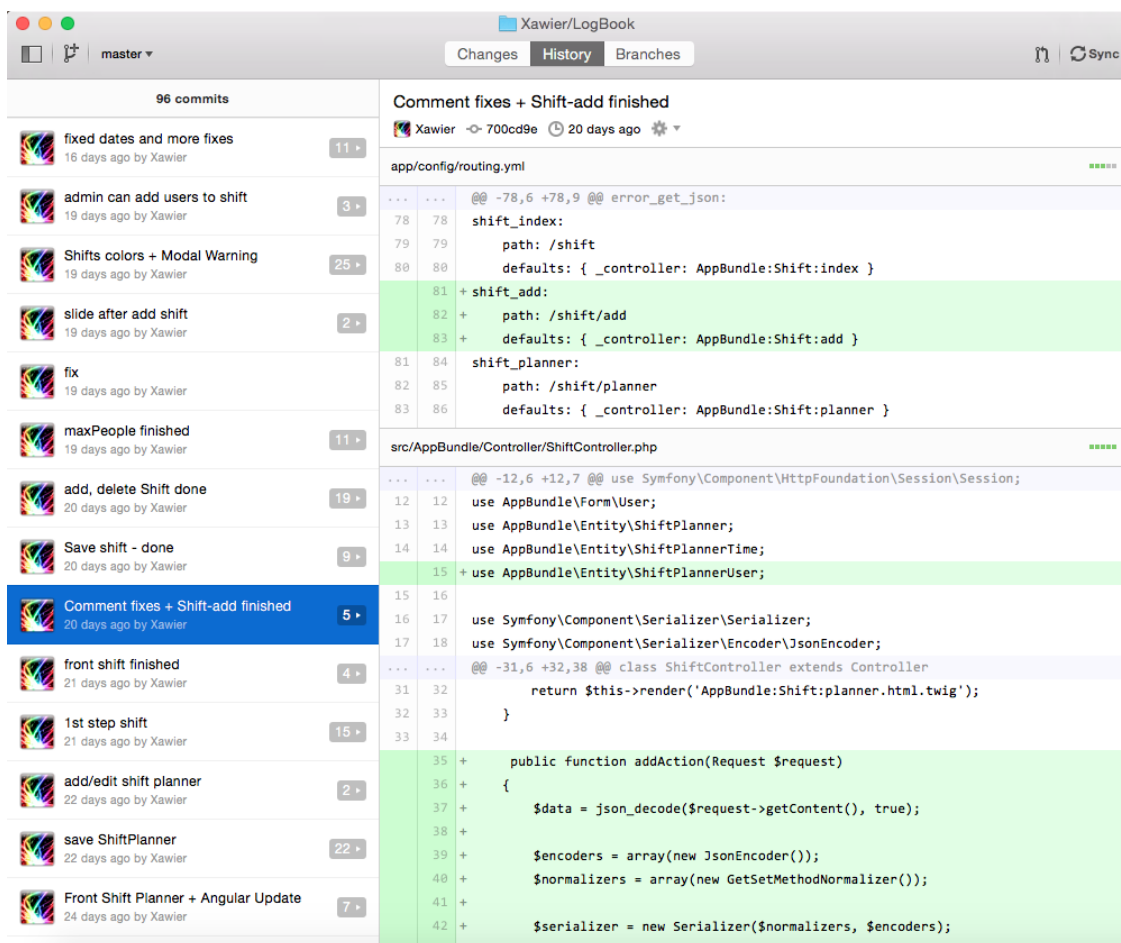


Figure 3.10: GitHub GUI with Logbook project commits.

3.5 Other Open Source libraries

Despite the usage of large front-end frameworks, they do not support many of specific features which were required to developed the Electronic Logbook application. Therefore, we have used many additional open source libraries available for the developers. We list them with a very brief description, with references giving additional and rich information.

List of front-end libraries:

1. CKEditor [51] - WYSIWYG, rich in features HTML editor.
 2. MathJax [52] - JavaScript display engine for LaTeX.
 3. Moment.js [53] - JavaScript date-time library for manipulating, formatting and parsing dates.
 4. Angular ngInfiniteScroll [54] - Angular.js module for infinity scroll effect to dynamically load data, without pagination in the asynchronously mode.
 5. Angular.js ngAnimate [55] - Angular.js module for animating HTML DOM elements.
 6. Angular.js Gravatar [56] - Angular.js module for generate avatar images from email in real time.
 7. Angular.js SmartTable [57] - Angular.js module to display data table with filtering and sorting features.
 8. Angular.js ResponsiveCalendar [58] - Angular.js module with calendar templates.
 9. Symfony2 ornicar/GravatarBundle [59] - Gravatar bundle for Symfony2.
 10. Symfony2 friendsofsymfony/rest-bundle [60] - Symfony2 bundle to develop RESTful API.
 11. Symfony2 jms/serializer-bundle [61] - Symfony2 bundle to serialize and unserialize data.
 12. Bootstrap DateTimePicker [62] - Functional date time picker for Bootstrap.
 13. Bootstrap Dialog [63] - Modal dialog boxes for Bootstrap.
-

4. Implementation of main functionalities of developed Logbook system

Implementation of an Electronic Logbook for the J-PET group, required to utilize and combine many existing technologies and frameworks, and to design entirely new dedicated solutions. In this chapter we will show the most interesting functional solutions used in the application together with the short user description.

4.1 Logbook database

Logbook database structure and relationships was designed based on the application functional requirements. In order to store various logbook data we have used a MySQL relational database. To design and implement database structure together with the relations we have used MySQL Workbench [64], which is a visual tool for data modelling and SQL database development. To design the relation between entities we have used Crow's foot notation, where each relation is marked with different symbol. The explanation of used Crow's notation symbols is given in Fig 4.1.

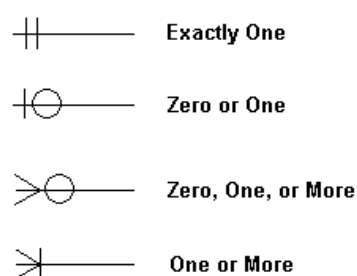


Figure 4.1: Used database relation symbols according to the Crow's foot notation.

Designed database consist of nine entity interconnected with different types of relations. Entities reflects three main functionality of the design application:

1. saving the logbook entries,
2. shift planner management,
3. error and warning handling system.

Moreover, two entities are related with the user management and user settings of the application. The entity relational diagram (ERD) is presented in Fig. 4.2. The detailed description

of each entity with the explanation of each data field is given in tables: 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9.

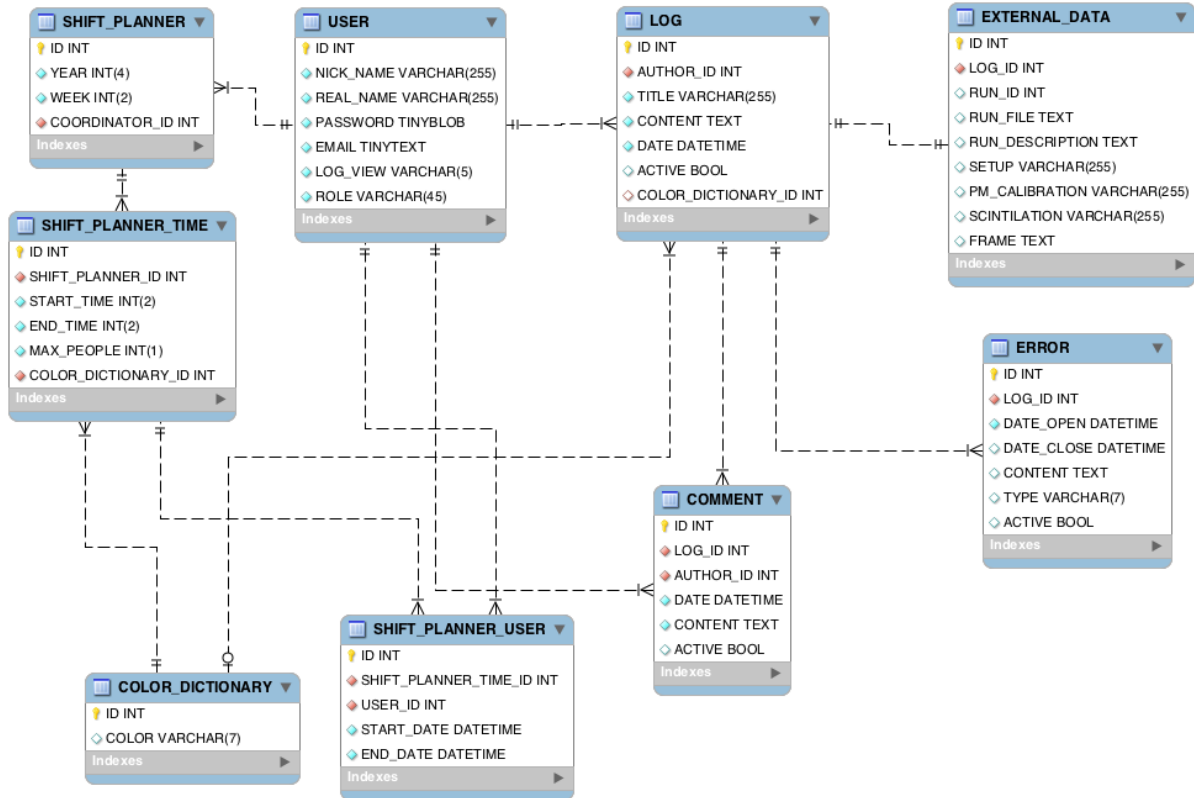


Figure 4.2: The designed Entity Relation Diagram (ERD) for the developed Electronic Logbook application.

USER	
ID	User id (from J-PET User Database)
NICK_NAME	Login name (from J-PET DB)
REAL_NAME	First name and surname (from J-PET DB)
PASSWORD	Encrypted user password (from J-PET DB)
EMAIL	User email (from J-PET DB)
LOG_VIEW	Preferable user main view
ROLE	User role in application (ADMIN, USER or INACTIVE)

Table 4.1: Entity USER, containing information about the users.

LOG	
ID	Log Id
AUTHOR_ID	User id (Reference to USER table)
TITLE	Log title
CONTENT	Log content
DATE	Log creation date
ACTIVE	Log is active(1) or deleted(0)
COLOR_DICTIONARY	Log shift color based on create time (Reference to COLOR_DICTIONARY table - can be null)

Table 4.2: Entity LOG, containing single post information.

EXTERNAL_DATA	
ID	Unique Id
LOG_ID	Log id (Reference to LOG table)
RUN_ID	Current test run id (from J-PET Parameter DB)
RUN_FILE	Measurement Run number (from J-PET Parameter DB)
RUN_DESCRIPTION	Run description (from J-PET Parameter DB)
SETUP	Setup identifier (from J-PET Parameter DB)
PM_CALIBRATION	PM calibration identifier(from J-PET Parameter DB)
SCINTILATION	Scintillation detector calibration identifier(from J-PET Parameter DB)
FRAME	Frame id with description (from J-PET Parameter DB)

Table 4.3: Entity EXTERNAL_DATA, containing information from the external experimental database.

COMMENT	
ID	Comment id
LOG_ID	Log id (Reference to LOG table)
AUTHOR_ID	User id (Reference to USER table)
DATE	Comment date-time of creation
CONTENT	Content of the comment field
ACTIVE	Comment is active(1) or deleted(0)

Table 4.4: Entity COMMENT, containing information about the comments to the logbook posts.

SHIFT_PLANNER	
ID	Shift planner id
YEAR	Current year
WEEK	Current week
COORDINATOR_ID	User id (Reference to USER table)

Table 4.5: Entity SHIFT_PLANNER, containing information about the shift plan for each week.

SHIFT_PLANNER_TIME	
ID	Shift planner time id
SHIFT_PLANNER_ID	Shift planner id (Reference to SHIFT_PLANNER table)
START_TIME	Shift start hour (0-23)
END_TIME	Shift end hour (0-23)
MAX_PEOPLE	Max people for shift (1-5)
COLOR_DICTIONARY	Shift color based on start hour (Reference to COLOR_DICTIONARY table)

Table 4.6: Entity SHIFT_PLANNER_TIME, containing information about the single shifts time intervals allocated in the weekly shift plan.

SHIFT_PLANNER_USER	
ID	Shift planner user id
SHIFT_PLANNER_TIME_ID	Shift planner time id (Reference to SHIFT_PLANNER_TIME table)
USER_ID	User id (Reference to USER table)
START_DATE	Shift start date
END_DATE	Shift end date

Table 4.7: Entity SHIFT_PLANNER_USER, containing information about the users assigned for each shift time interval in the weekly shift plan.

COLOR_DICTIONARY	
ID	Color id and shift start hour (0-23)
COLOR	Color in hex code

Table 4.8: Entity COLOR_DICTIONARY, containing information about the colors for each shift hour shown in the weekly shift plan.

ERROR	
ID	Error id
LOG_ID	Log id (Reference to LOG table)
DATE_OPEN	Error create date
DATE_CLOSE	Error close date
CONTENT	Error content
TYPE	Error type (ERROR or WARNING)
ACTIVE	Error is active(1) or deleted(0)

Table 4.9: Entity ERROR, containing information about reported errors and warnings.

4.2 User authentication

The developed application it is a part of a larger project and dedicated to specific group of users, therefore the access to the information should be secured and limited. Symfony2 provides the security system which can authenticate users using three methods: (i.) database, (ii.) Active Directory, and (iii.) OAuth server. In case of developed system we will use authentication via users database with the support of the Doctrine entity manager. In this manner the security module will identify the user trying to gain access to the application, and than it will check if the user has appropriate access right to view the application content. If the authorization module will not find the given authentication data in the database the user will not be granted access to the application.

The main requirement of the designed authentication module was to integrate it with existing J-PET user account database. However to prevent numerous connections to the external database, user authentication data is synchronized with the local Logbook database (Fig. 4.3) on demand. In this solution system maps only the most important informations such as: login, password, email, and the name of the user.

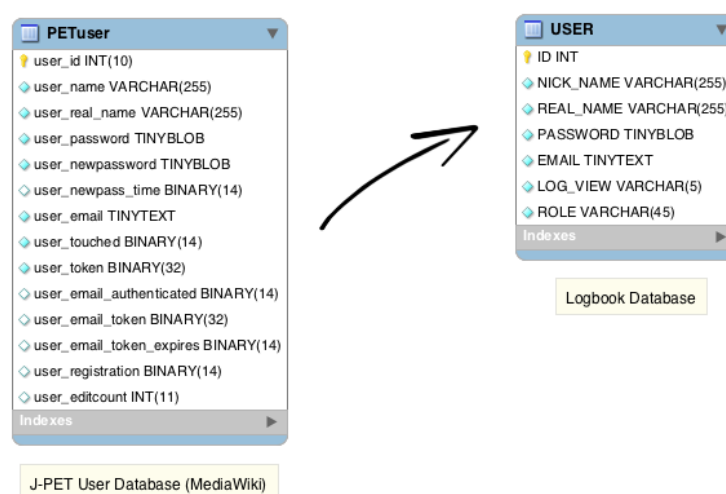


Figure 4.3: Structure of the User Entities in the (left) J-PET-User database and (right) locally synchronized Logbook database.

In the developed application, the users authentication data comes from external database, therefore, a dedicated user login service had to be written. In the external database the user authentication data is kept in the form of encrypted password and the security phrase called “salt”. These two special words are saved in the database in the form of a string kept in a TinyBlob [65] format. Example of such encrypted string is shown in Fig. 4.4. First element

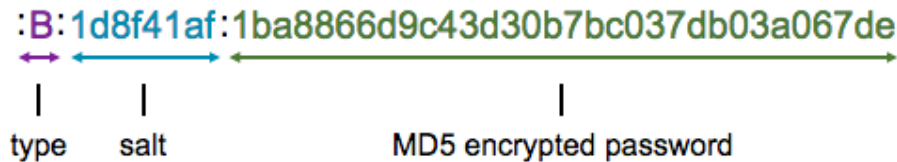


Figure 4.4: The password encrypted with the MD5 algorithm consisting from three parts: encryption type, salt and MD5 hash, kept in the TinyBLOB format.

of the string is a prefix which determines whether the password is encrypted with salt or no: *A* - means that the password has not salt encryption, *B* - denotes that the password is salt protected. In the case of J-PET users database all user accounts were created with the salt, therefore all the encrypted strings in the database are preceded by `:B:` prefix. The second part of the string is the salt (in type *A* this field is empty). Accordingly, the third part of the string is the encrypted password with the MD5 [66] algorithm.

The Symfony2 framework and Doctrine tool, by default does not handle properly TinyBlob format. Therefore, we had to directly use the *User* entity, together with the dedicated written “getter function”. This method uses special PHP5 stream function called `stream_get_contents`, which can convert the password stream into a plain string [67]. The function code is shown in Fig. 4.5.

```
public function getPassword()
{
    $resource = $this->password;
    $string = @stream_get_contents($resource);
    return $string;
}
```

Figure 4.5: Function providing the service for the proper reading of the TinyBLOB format.

After obtaining the authentication data in a form of a plain string, the next step was to implement Symfony2 password encoder service, which can validate the correctness of the password provided by the user during the authentication process. The corresponding code of validating function is shown in Fig. 4.6. To proceed with the user authentication we have prepared special function called `isPasswordValid` which accepts following three parameters:

1. `$encoded` - encrypted string from database,
2. `$raw` - password given by user during authentication,
3. `$salt` - special salt phrase.

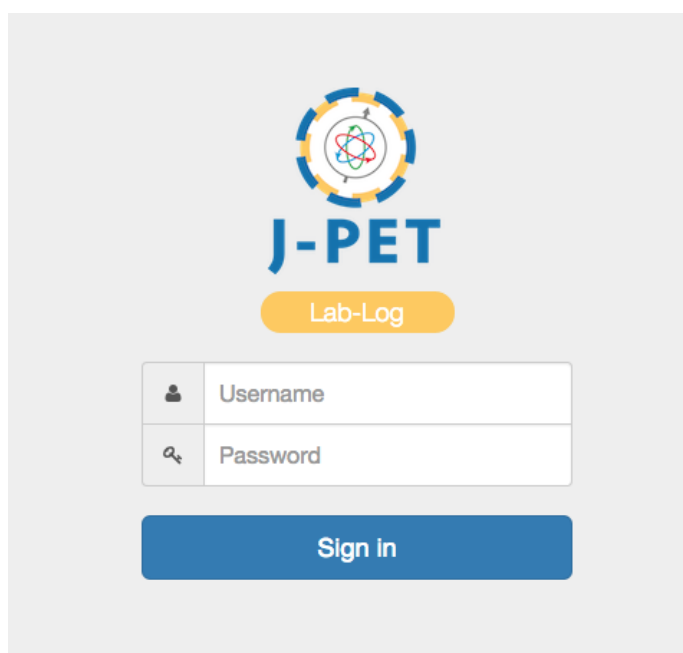
```
public function isPasswordValid($encoded, $raw, $salt)
{
    list($dummy, $type, $salt, $hash) = explode(':', $encoded);

    $pwd_hash = md5($raw);
    if ($type == 'B' && md5("$salt-$pwd_hash") === $hash) {
        return true;
    } elseif ($type == 'A' && $salt === $pwd_hash) {
        return true;
    } else {
        return false;
    }
}
```

Figure 4.6: Dedicated password encoder service function providing the password validation and user authentication.

During this process the *\$encoded* string is divided to a list of three strings corresponding to: *\$type*, *\$salt* and *\$passwordMD5*, as it was described previously. Next the parameter *\$raw* is encrypted using the MD5 algorithm and saved to the new *\$pwd_hash* variable. In the case when the *\$type* is equal to *B* value, than *\$passwordMD5* is compared with the MD5 encrypted result of the *\$salt-\$pwd_hash* operation. In case when *\$type* is equal to *A* value, than the function is comparing only with the *\$pwd_hash*. In both cases when the password encryption function return TRUE value, then password validation is correct, and the requesting user is authenticated to the application.

For the user side, for the authentication process we have prepared an user friendly authorization panel which is constructed from two input fields and an action button pointing at the authentication service in the corresponding controller (Fig. 4.7). User in order to authenticate, has to provide username and password which is then compared to the one stored in the J-PET User Database, according to the procedure described above. When



The image shows a login interface for the J-PET application. At the top center is the J-PET logo, which consists of a stylized atom symbol with blue and yellow orbits, and the text 'J-PET' in blue. Below the logo is a yellow rounded rectangle containing the text 'Lab-Log'. Underneath this are two input fields: the first is labeled 'Username' with a person icon to its left, and the second is labeled 'Password' with a magnifying glass icon to its left. At the bottom of the form is a blue rounded rectangle with the text 'Sign in' in white.

Figure 4.7: The view of login window to the Electronic Logbook application.

the credentials given by the user in the input fields are correct, the authentication module redirects the user to the main view of the application.

4.3 Main Views of the Logbook

Home page of Logbook is the most important view of the designed application. In the main application window users see all the logbook entries displayed in the chronological order from the newest one to the oldest. In this way users can easily browse and follow the latest research activities. The default view is the grid tile mode (Fig. 4.8) which is organized in such a way that on the PC three tiles are shown in one row. Every single tile is composed of:

- Author name,
- Title of the logbook entry,
- Date and time of creation,
- Color of the shift on which entry was crated,
- Warning and Error alerts (optional - when the information is available).

To read the detail information from single lab log entry one needs to click on the single tile.

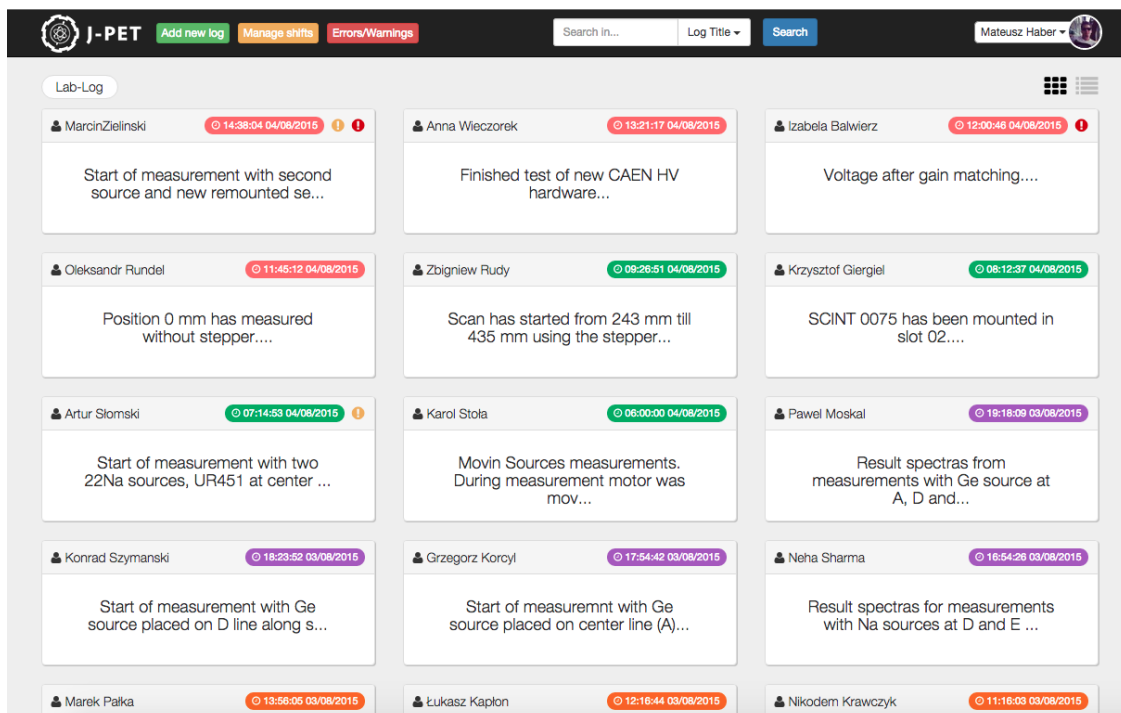


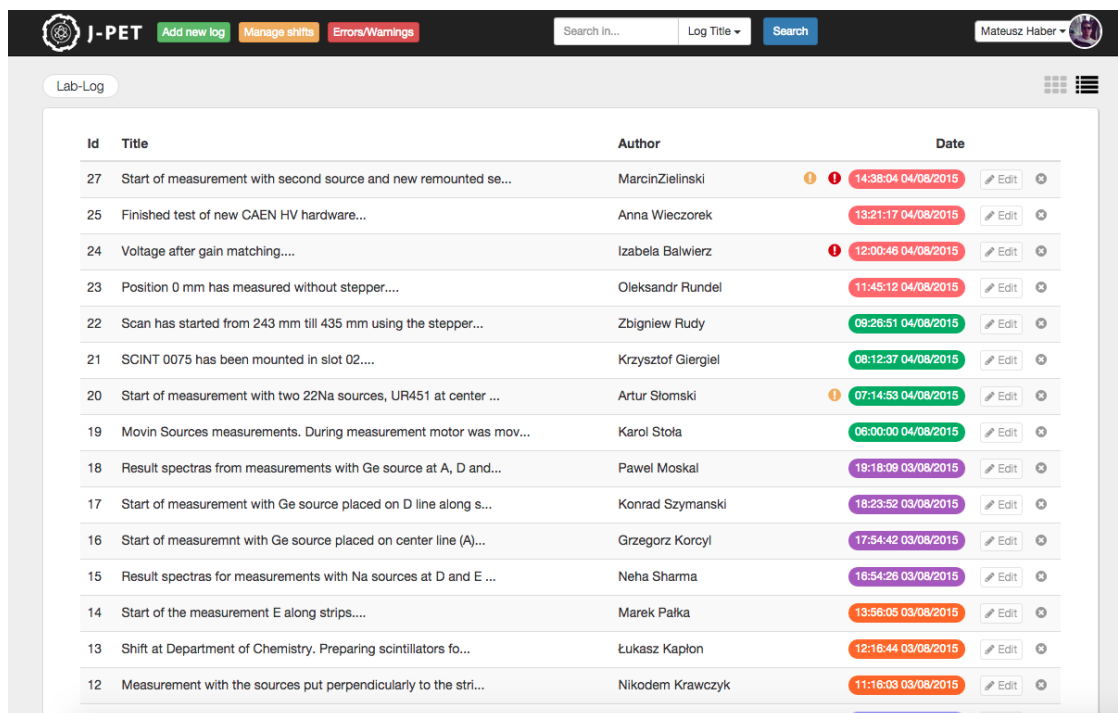
Figure 4.8: View of the Logbook home page in the tiles mode on a standard PC.

The second view of the main window of the application is a list of logs ordered chronologically (Fig. 4.9), where in a single row we have access to information like:

- Id of the logbook entry,
- Title of the logbook entry,
- Author name,
- Warning and Error alerts (optional - when the information is available)
- Date and time of creation,
- Color of the shift on which entry was created,
- Edit and delete buttons (only for administrator).

Similarly as before, to access detail information from the single logbook entry one needs to click on the desired row.

In case of both views the colored date and time correspond to the shift time intervals. (shift manager is described in chapter 4.6).



Id	Title	Author	Date
27	Start of measurement with second source and new remounted se...	MarcinZielinski	14:38:04 04/08/2015
25	Finished test of new CAEN HV hardware...	Anna Wieczorek	13:21:17 04/08/2015
24	Voltage after gain matching....	Izabela Balwierz	12:00:46 04/08/2015
23	Position 0 mm has measured without stepper....	Oleksandr Rundel	11:45:12 04/08/2015
22	Scan has started from 243 mm till 435 mm using the stepper...	Zbigniew Rudy	09:26:51 04/08/2015
21	SCINT 0075 has been mounted in slot 02....	Krzysztof Giergiel	08:12:37 04/08/2015
20	Start of measurement with two 22Na sources, UR451 at center ...	Artur Słomski	07:14:53 04/08/2015
19	Movin Sources measurements. During measurement motor was mov...	Karol Stola	06:00:00 04/08/2015
18	Result spectras from measurements with Ge source at A, D and...	Pawel Moskal	19:18:09 03/08/2015
17	Start of measurement with Ge source placed on D line along s...	Konrad Szymanski	18:23:52 03/08/2015
16	Start of measurmnt with Ge source placed on center line (A)...	Grzegorz Korcyl	17:54:42 03/08/2015
15	Result spectras for measurements with Na sources at D and E ...	Neha Sharma	16:54:26 03/08/2015
14	Start of the measurement E along strips....	Marek Pałka	13:56:05 03/08/2015
13	Shift at Department of Chemistry. Preparing scintillators fo...	Łukasz Kaplon	12:16:44 03/08/2015
12	Measurement with the sources put perpendicularly to the stri...	Nikodem Krawczyk	11:16:03 03/08/2015

Figure 4.9: View of the Logbook home page in the list mode on the standard PC.

Switching between two views is simple and intuitive, possible without refreshing the web page. In order to switch to different view one needs to use the two button icons in the right corner of the main window, just above the tails or the list of post. After changing to preferred view, setting are immediately saved to the user session and database. Both views have been enriched by ngInfiniteScroll directive available from Angular.js library. In this way, it becomes possible to dynamically download additional content without pagination of the web page

By utilizing the Bootstrap schemes the all the views in the application are responsive and can be displayed on most portable devices. Application, automatically based on the

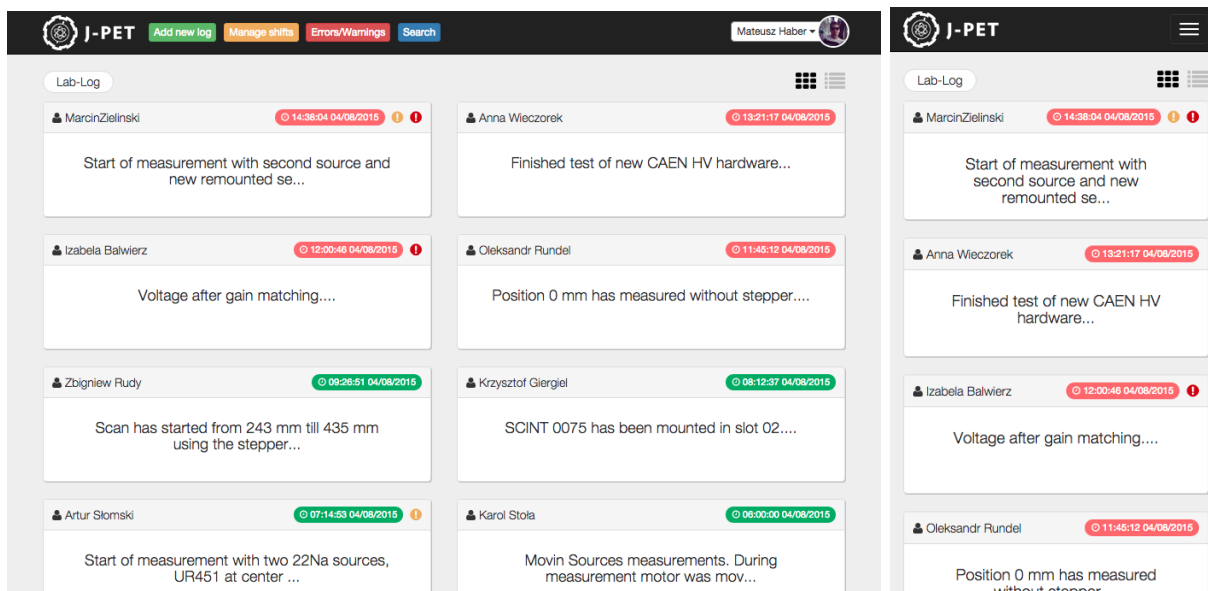


Figure 4.10: View of the Logbook home page in the tiles mode on: **(left)** tablet device, **(right)** on smartphone.

information about the size of the screen, adjusts the layout to the device requirements. On the tablet (Fig. 4.10 **(left)**) tiles are reduced to two column grid. Also, in the main navigation bar the search input field is replaced with the button "Search". On the smartphone (Fig. 4.10 **(right)**) tiles are reduced to one column, with collapsed menu to the single icon.

4.4 Logbook functionality

The main functionality of the developed application is the possibility to register information about the laboratory and research related work within a single post in the chronologically sorted form. This will enable researchers to collect research results in a one place.

4.4.1 Adding logbook entry

The most crucial function of the developed application is an ability of adding single post which presents the results of the researcher work or status of the conducted measurements. To place a new logbook entry one needs to use "Add new log" button which is always visible for authenticated users in the navigation bar on the top of the main application window (Fig. 4.11). Every log: consists of title, content and extra data imported from the J-PET Experimental Database. Content text area was build based on the CKEditor, which offers a very rich text formatting options. The editor offers the possibility of attaching images to each post, which are stored on the server.

As it was mentioned before each single post can be supplemented with the experimental data imported from external experimental parameter database. This is realized in the asynchronously mode without refreshing the view using the Angular.js library. To download experimental parameters, Logbook application connects to the PostgreSQL J-PET Experimental Experimental database [68], which contains all current and historical information

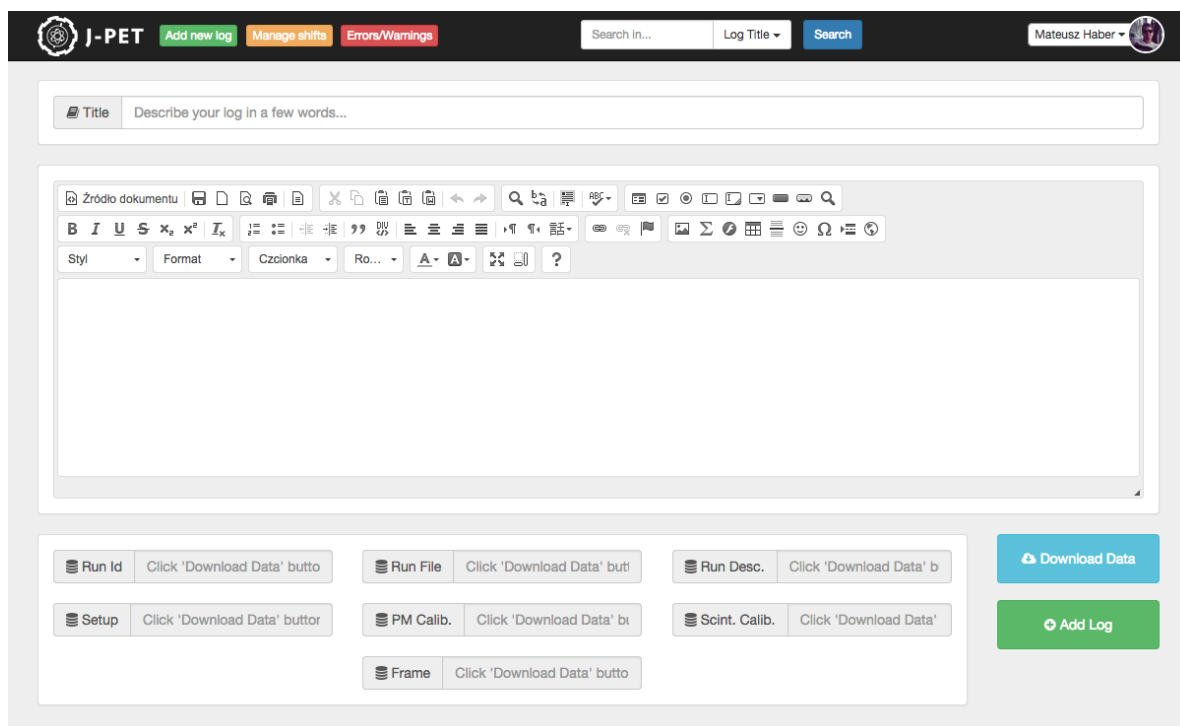


Figure 4.11: View presenting the feature of adding new entry to the logbook.

about parameters and conditions of measurement conducted with the J-PET apparatus. To connect with the database system we have used the PHP *pdo_pgsql* driver [69]. As the Logbook is an independent application, the experimental parameters are downloaded and stored into local database entity "EXPERIMENTAL_DATA". The data items which is downloaded and stored locally are:

1. current measurement id,
2. name of the file with the experimental data,
3. measurement description,
4. detector setup configuration used in the measurement,
5. calibration for the photomultipliers,
6. calibration for the scintillating modules,
7. the identifier of the hardware frame used during the measurement with the description.

The schematic view of the interesting fragment of the J-PET parameter database and the local Logbook entity EXTERNAL_DATA is shown in Fig. 4.12.

Controller action responsible for downloading data from J-PET Experimental Parameter Database was named: *getExternalDataAction*. The code of this action is shown in Fig. 4.13. This method, after execution connects via Doctrine to the PostgreSQL database. Next by using the *Criteria()* method it filters retrieved data by *id* property, and only the latest experimental data are fetched. Finally, retrieved data is serialized to JSON format and returned from the function to be displayed by the Angular.js, in the asynchronously mode.

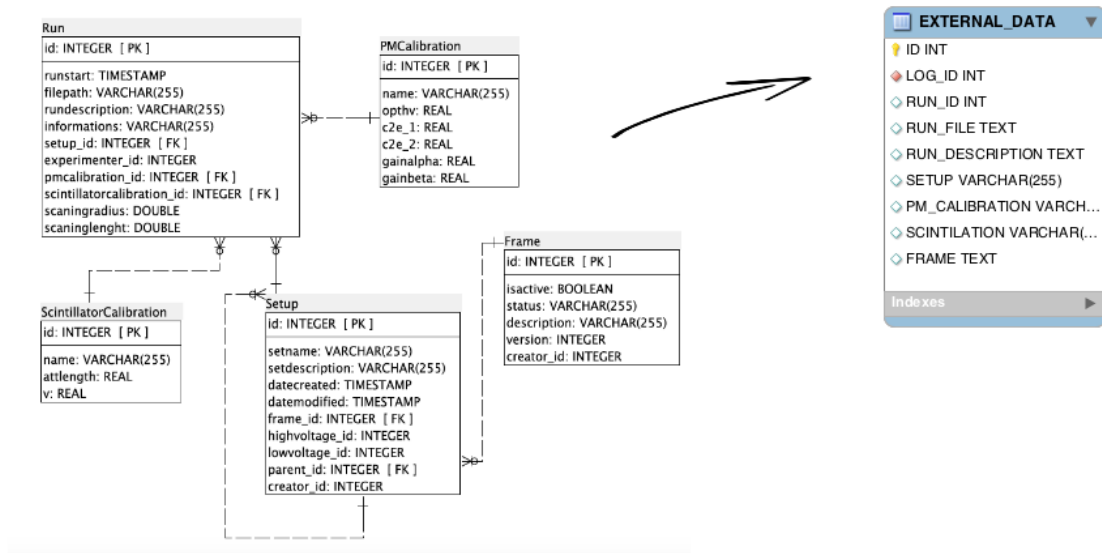


Figure 4.12: Part of the Entity Relational Diagram of the J-PET Experimental Parameter (PostgreSQL), and local Logbook database (MySQL) showing EXTERNAL_DATA entity.

```

public function getExternalDataAction()
{
    $encoders = array(new JsonEncoder());
    $normalizers = array(new GetSetMethodNormalizer());

    $serializer = new Serializer($normalizers, $encoders);

    $criteria = new Criteria();

    $criteria->where($criteria->expr()->gt('id', 1))
        ->orderBy(array('runstart' => Criteria::DESC))
        ->setMaxResults(1);

    $externalData = $this->getDoctrine()
        ->getRepository('AppBundle:Run', 'postgresql')->matching($criteria);

    $jsonContent = $serializer->serialize($externalData->getValues()[0], 'json');

    return new Response($jsonContent, 200, array('Content-type' => 'application/json'));
}

```

Figure 4.13: Function *getExternalDataAction* responsible for downloading external data from J-PET Experimental Parameter Database.

4.4.2 Single logbook view

After creating a logbook entry (exactly like it was shown in section 4.4.1) user will be redirect to the single logbook view. Every single logbook post is a collection of text provided by the user during the post creation and optionally the experimental parameters downloaded from the external database (Fig. 4.14). It includes the title, detailed description, experimental parameters and author section. Additionally, text in each post can be enriched with LaTeX [70] expression (emulated by MathJax), photos, plots and variety of multimedia content. On the right side of the single post view the user data box is displayed. This

J-PET

[Add new log](#)
[Manage shifts](#)
[Errors/Warnings](#)

Log Title ▾
Search

Mateusz Haber

WARNING was reported!
Open Time: 14:40 04/08/2015

Stepper has crashed again, so all position need to measured manually. Still to measure with 15k events: 360, 381, 393, 405, 417, 429, 435 and with 5k events 141, 144, 147.

Close WARNING

ERROR was reported!
Open Time: 14:41 04/08/2015

Zip file for position 360 copied to petpc is empty, so this position needs to be measured again.

Close ERROR

WARNING was reported!
Open Time: 14:41 04/08/2015

Positions 360 - 435 are not measured!

- RESOLVED -

Close TIME

Start of measurement with second source and new remounted setup.

17,7cm
15,0cm

17,7cm
15,0cm

Near SDA | Near window

a 21,96 | 21,92
b 17,98 | 18,05
c 18,02 | 18,01
d 18,01 | 18,04

We mounted setup for A1 measurements. The spectrum after first mounting:

A1Test_area1
Entries: 7202
C1_CM9491: 86.16
C2_CM9497: 23.78
C3_CM9503
C4_CM9475

A1Test2_area1
Entries: 4925
C1_CM9491: 84.75
C2_CM9497: 22.69
C3_CM9503
C4_CM9475

Spectrum after second remount:

A1Test4_area1
Entries: 5025
C1_CM9491: 87.53
C2_CM9497: 25.85
C3_CM9503
C4_CM9475

A1Test4_amplitude1
Entries: 5025
C1_CM9491: 1049
C2_CM9497: 309.7
C3_CM9503
C4_CM9475

Mateusz Haber

14:38:04 04/08/2015

Run Id 29

Run File will be completed lat

Run Desc. KB7 time calibratic

Setup Calibration of time offs

Frame 5 : Generator inputted

Report Warning

Report Error

Edit Log

Delete Log

Marcin Zieliński

Data for position 168 is measured, copied, checked and linked. Info in Szymon's spreadsheet.

14:42 04/08/2015

Marcin Zieliński

Before running stepper I've restarted SDA and deleted data in tmpSM. New directory: /home/pet/Desktop/PomiaryPET/2014/August/BC-420-500-Supplementary/

14:43 04/08/2015

Do you have something to say about that?

Add Comment

Figure 4.14: View of the single logbook entry. Upper part shows the information from the Error and Warning Handling System, middle part is the logbook entry content, right sidebar shows the information about the: user, date and time of post creation, and the experimental parameters downloaded from the J-PET database, lower part contains the users comments.

right sidebar contains the basic information about the user who created the entry and the information about the date and time of the post creation. If the user possesses the avatar in his account, it will be also displayed. This functionality is based on the Gravatar service [71], which allows connecting user email with an avatar picture. Every lab log entry is fixed and can not be modified. However, to have a possibility of adding or discussing the content of the post the functionality of commenting was introduced. Each user can add unrestricted number of comment. However, for special cases the user with the administrative right can edit and remove post from the log. Additionally, each single lab log entry can be supplemented with the "Warning" or "Error" message handled by the separate module of the application described in chapter 4.5.

The view of the single post was built using the TWIG engine. The template responsible for displaying this view has many lines of code, therefore as an example we will show and discuss the section responsible for displaying the users comments. Figure 4.15. shows a comments part of log view TWIG template. Presented template is a compilation of the

```

{% for logComment in logComments %}
  <div class="row">
    <div class="col-md-12">
      <div class="panel panel-default">
        <div class="panel-body">
          <div class="col-md-1 col-sm-2 col-xs-3">
            <div class="row comment-author">
              <br>
              {{ logComment.author.realName }}
            </div>
          </div>
          <div class="col-md-11 col-sm-10 col-xs-9">
            <div class="row">
              <div class="well comment">
                {{ logComment.content }}
                <span class="comment-date pull-right">{{ logComment.date|date("H:i d/m/Y") }}</span>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
{% endfor %}

```

Figure 4.15: Part of log view TWIG template, responsible for displaying comments, in the single logbook entry.

Bootstrap and TWIG code. First and last line of the code is a TWIG loop command: `{% for logComment in logComments %} {% endfor %}`, responsible for displaying each single comment in a separate `<div>` box object. The `<div>` elements are attributed with Bootstrap tags, for building the responsive grid schema. Finally, middle part of the code shows the `` element associated with the Gravatar directive to generate image avatar based on the user email, followed by the TWIG variables: `{{ logComment.author.realName }}`, `{{ logComment.content}}` and `{{ logComment.date }}` containing the author, content and date of the comment, respectively. Additionally, the date variable is supplemented with the option `date(H: i /d/n/Y)` to print the date in a desirable format.

4.4.3 Logbook search module

As the Logbook will store many data one needs efficient way of searching for a particular information. Therefore, the Logbook was equipped with a dedicated search engine. It was build based on the Angular.js library, which enabled to prepare efficient and user friendly search functionality working in the asynchronous mode without need to refresh the view. The search engine enables to search logbook entries restricting results to five parameters:

- title of logbook singe entry,
- content included in the logbook entry,
- author name of the logbook entry,
- date of creation of the logbook entry,
- identifier of logbook entry.

In principle, it is possible to search only by one parameter or add them in a conjunction. Search results can be sorted by desired parameter also without need to refresh the page.

Id	Title	Author	Date
17	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	19:00:57 15/07/2015
16	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	18:00:31 15/07/2015
15	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aen...	Mateusz Haber	03:02:03 14/07/2015
14	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	17:37:01 13/07/2015
13	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	17:35:28 13/07/2015
12	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:33:08 13/07/2015
11	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:11:43 13/07/2015
10	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:08:59 13/07/2015
9	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:07:49 13/07/2015
8	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:06:02 13/07/2015
7	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:04:25 13/07/2015
6	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:03:29 13/07/2015
5	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:03:19 13/07/2015
4	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Don...	Mateusz Haber	01:01:14 13/07/2015

Figure 4.16: View presenting the search engine functionality. The search parameters can be entered in the upper part of the window, and the results are presented in the table below in the form of list.

On the server side the search functionality was build based on the Symfony2 MySQL query builder. It is a part of a *getLogs* function (shown in Fig. 4.17) in the *LogRepository* module. Function accepts four parameters: *\$pageStart*, *\$pageItems*, *\$search*, and *\$sort*.

```

public function getLogs($pageStart,$pageItems,$search = null, $sort = null)
{
    if($search){
        $query = $this
            ->createQueryBuilder('l')
            ->where('l.active = 1');
        $i = 0;
        foreach ($search as $key => $value) {
            if($key = 'author'){
                $authorId = $this->getEntityManager()->createQueryBuilder()
                    ->select('u')
                    ->from("AppBundle:User", "u")
                    ->where('u.realName LIKE :userRealName')
                    ->setParameter('userRealName', '%'. $value .'');
                ->getQuery()
                ->getResult();
                $a = 0;
                foreach ($authorId as $keyA => $valueA) {
                    if($a){
                        $search[$key] .= ',';
                    }
                    $search[$key] = $authorId[$keyA]->getId();
                }
                $query->andWhere('l.'. $key .' IN (:'. $key .'Value)');
            }else{
                $query->andWhere('l.'. $key .' LIKE :'. $key .'Value');
            }
            $i++;
        }

        foreach ($search as $key => $value) {
            if($key = 'author'){
                $query->setParameter($key.'Value', $value);
            }else{
                $query->setParameter($key.'Value', '%'. $value .'');
            }
        }
    }

    if($sort){
        if($sort['reverse']){
            $order = 'DESC';
        }else{
            $order = 'ASC';
        }
        $query->orderBy('l.'. $sort['predicate'], $order);
    }else{
        $query->orderBy('l.date', 'DESC');
    }

    $logs = $query
        ->setMaxResults($pageItems)
        ->setFirstResult($pageStart)
        ->getQuery()
        ->getResult();

    foreach ($logs as $log) {
        $isLogHaveErrors = $this->getEntityManager()
            ->getRepository('AppBundle:Log')
            ->isLogHaveErrorsIndex($log->getId());

        $log->setIsLogHaveErrors($isLogHaveErrors);
    }

    return $logs;
}

```

Figure 4.17: Function *getLogs* responsible for displaying the logbook list of entries, and search functionality.

When the value of the *\$search* variable is not NULL, then function returns search results. This variable is given in a form of associative array containing search parameters and their values (e.g. *title => 'Night J-PET measurement'*). Next function iterates over all given parameters and builds the database query with Doctrine QueryBuilder. In the body part of the function, also we have implemented code responsible for data sorting and pagination, which is necessary for infinity scroll view. This function is also used for display logbook entries in the main logbook view in the form of list or tiles.

4.5 Experimental error and warning handling system

Logbook functionality has been supplemented with Error and Warning Handling System. This allows each user to report warnings or errors during the experiment and laboratory work. After reporting warning or an error with a short description it is in "open" status, which means that it needs to be handled. In this way each user of the Logbook is aware of problem event. Also, from this moment everyone, who resolves the problem, can close the report.

All warnings and error posts are displayed in one place on a special Error/Warning view page (see Fig. 4.18). The display page was prepared by introducing the infinity scroll functionality. This solution allows to avoid the pagination, and enables to load additional content without refreshing the web page. Warnings are marked with yellow color while the errors, in contrast, are marked with red color. Resolved error and warning reports independent on the type are marked with the blue color. Also, each error and warning report is supplemented with the start and end date/time signature. Furthermore, from the

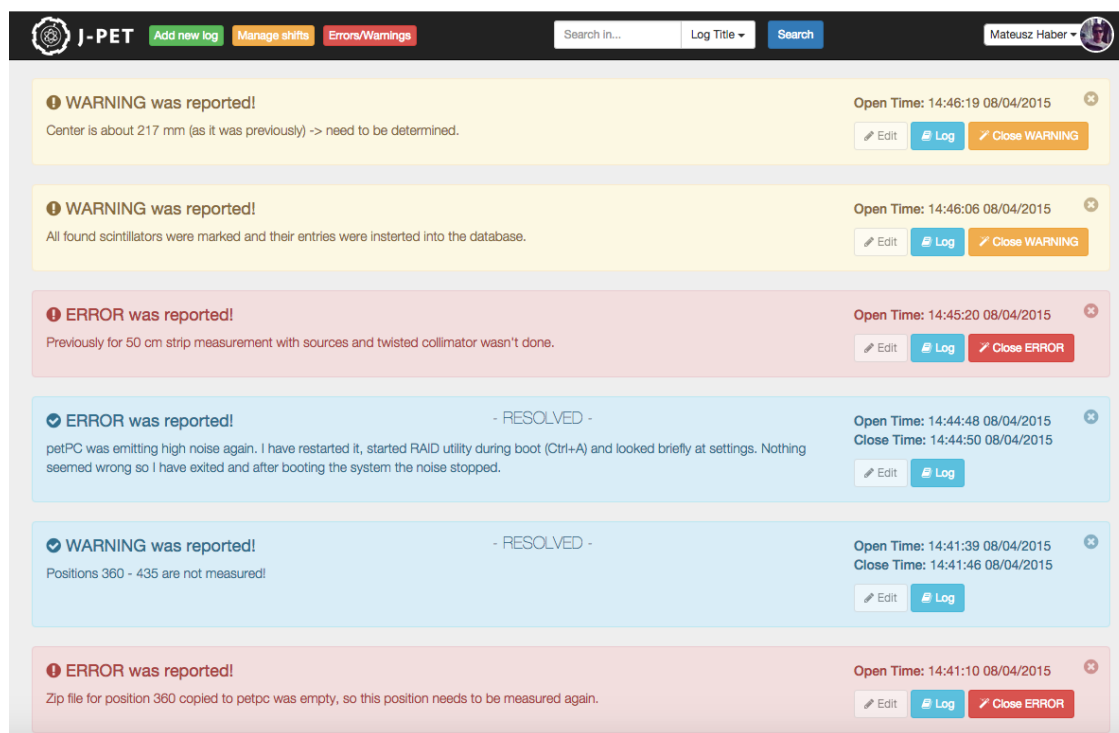


Figure 4.18: View of the experimental Error and Warning Handling System.

list of errors and warnings one can access to the single lab log entry in which the event was reported. In this way one can easily find out the more detail context of the occurred event. The administrator user in contrast to the normal user can delete and modify the reported errors and warnings.

To create and implement the Error and Warning Handling System we have used features offered by the Angular.js library. The whole functionality is build based on two application controllers: *Resource* (shown in Fig. 4.19,) and *errorCtrl* (shown in Fig. 4.20).

The factory module *Resource* is responsible for downloading data each time when it is called by the user action. It contains function *getPage* which accepts three parameters:

1. *start* - position of the first element to downloaded,
2. *number* - number of elements to be downloaded,
3. *params* - parameters to sort content.

The *\$http.post()* method sends post request to the server in JSON format, and waits for the data to be returned also in JSON form. This process is supported by *\$q* service which helps in run the asynchronously mode.

In case of the controller module *getPage*, it uses the *Resource* factory controller when executing the *callServer* function “on load”. This function is defined in SmartTable [57] directive in TWIG template. That module contains *\$scope.loadMore* function which is a part of ngInfiniteScroll module [54] of Angular.js. With the help of this function new data are downloaded every time when user scrolls down the web page. Moreover the controller function contains the *\$scope.isLoading*, and *\$scope.noMore* directives to animate a loading icon during the process of data downloading.

```

app.factory('Resource', ['$q', '$timeout', '$http', function($q, $timeout, $http) {

    function getPage(start, number, params) {

        var deferred = $q.defer();

        $http.post('er/get/json', {
            pageStart: start,
            pageItems: number
        }).
        success(function(data, status, headers, config) {
            dataJson = data;
            deferred.resolve({
                data: dataJson
            });
        }).
        error(function(data, status, headers, config) {
            console.log(data);
        });

        return deferred.promise;
    }

    return {
        getPage: getPage
    };

}]);

```

Figure 4.19: Developed factory “Resource” function of an error/warning handling module responsible for the infinity scroll written in the Angular.js.

```

app.controller('errorCtrl', ['Resource', '$scope', '$http', '$timeout', function(service, $scope, $http, $timeout) {

    var ctrl = this;

    ctrl.callServer = function callServer(tableState) {

        ctrl.isLoading = true;

        var pagination = tableState.pagination;

        var start = pagination.start || 0;
        var number = pagination.number || 10;

        service.getPage(start, number, tableState).then(function(result) {
            ctrl.displayed = result.data;
            ctrl.isLoading = false;
        });

        $scope.loadMore = function() {
            ctrl.isLoading = true;
            start = start + 10;
            service.getPage(start, number, tableState).then(function(result) {
                if (result.data.length > 0) {
                    ctrl.displayed = ctrl.displayed.concat(result.data);
                    ctrl.isLoading = false;
                } else {
                    ctrl.isLoading = false;
                    ctrl.noMore = true;
                    $timeout(function() {
                        ctrl.noMore = false;
                    }, 1000);
                }
            });
        };

        //...//

    });
}]);

```

Figure 4.20: Developed controller “ErrorCtrl” function of an error/warning handling module responsible for the infinity scroll written in the Angular.js.

4.6 Shifts managements system

One of the important things in the laboratory work is a good organization of the tasks. Therefore, together with the Logbook functionality we have developed the shift management system which can help in better organization of research. Developed feature is based on Angular.js library and JSON format, supported by Moment.js. The shift management module is divided in two parts: (i.) user, and (ii.) administrator sections. We will describe them in following subsections.

4.6.1 Shift planner

Shift planner (Fig. 4.21) is administration section, allowing authorized users to plan the shift intervals for each week in the current year. In order to configure the week for shifts one needs to:

1. Select a week to configure.
2. Select coordinator (person responsible for the laboratory work).
3. Assign shift intervals, and the number of people needed for one shift.

Administrator can choose the length of a single shift interval from 1 up to 24 hours. Also, for each shift slot one has to define the maximal number of people to work on one shift. The upper limit for the number of people assigned to one shift was set to five. Each new shift interval has its individual color visible in the calendar and also reflected in the lab log entries as the color dates. After planing the whole week administrator has to save the settings by pressing the “*Save Shift Plan*” button. This operation is done asynchronously using Angular.js, without refreshing the web page.

Shift	From	To	Max People
1. Shift	0:00	6:00	2
2. Shift	6:00	12:00	3
3. Shift	12:00	18:00	2
4. Shift	18:00	24:00	3

Figure 4.21: View presenting the administration part of the shift plan module where one can plan shift intervals.

4.6.2 Shift calendar

The shift calendar is the module available for each active user of the application eligible to take shifts (Fig. 4.22). If the shift intervals were assigned by the administrator this is visible by colored hours on the left sidebar time scale of the calendar. Each color denotes one single shift interval. Users can assign for available shifts in current and upcoming weeks if the plan is available. To sign for the particular shift, user uses the input field available in the upper part of the web page where he picks up the date and the time of desired shift. If the shift time slot is available it will be allocated for the user and his name together with the avatar will appear in the calendar view. Color of the tile in the calendar is generated for each user individual based on his email. In case when the slot is not available or the

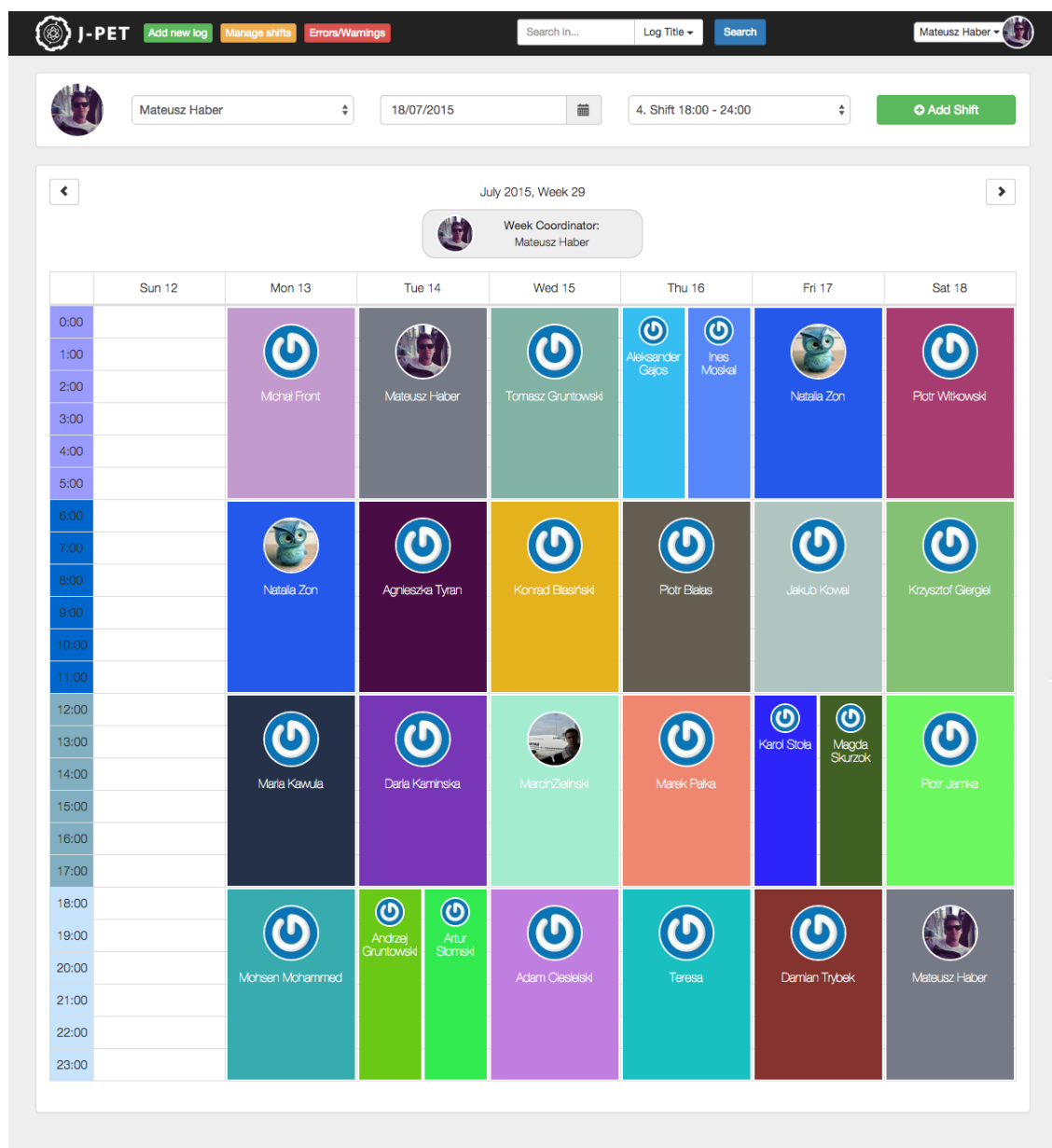
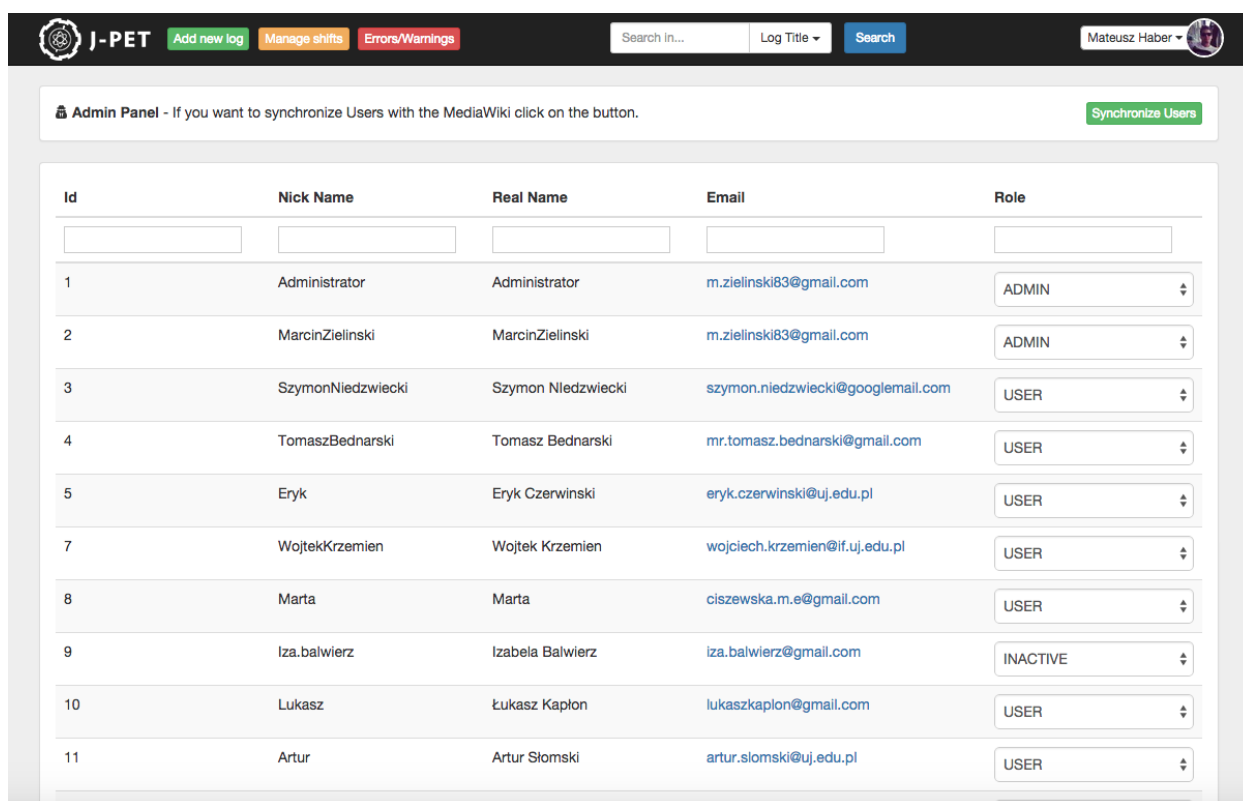


Figure 4.22: View of the Shift Calendar available for the users eligible to take shifts. Module visible in the upper part of the window can be used to sign for the shifts, and the lower part is the week calendar view in which user can see the filling of the shift plan.

maximal number of people for the shift is exceeded the system will inform the user in a form of a modal window with short information.

4.7 Administration panel

One of the requirements for the application was to implement the different access level for the users, depending on their role. Therefore, we have prepared the administration panel enabling to manage user right to different part of the application. However implemented administration panel has a special properties, because it uses the external J-PET User Database. In this approach administrator of the system has an ability to synchronize the application local user list with the central user database. This is realized by the button "*Synchronize Users*" shown in Fig. 4.23. The method in the *User* controller responsible for



The screenshot shows the J-PET Administration Panel. At the top, there is a navigation bar with the J-PET logo, buttons for "Add new log", "Manage shifts", and "Errors/Warnings", a search bar, and a user profile for "Mateusz Haber". Below the navigation bar, there is a message: "Admin Panel - If you want to synchronize Users with the MediaWiki click on the button." and a "Synchronize Users" button. The main content is a table with the following columns: Id, Nick Name, Real Name, Email, and Role. The table contains 11 rows of user data.

Id	Nick Name	Real Name	Email	Role
1	Administrator	Administrator	m.zielinski83@gmail.com	ADMIN
2	MarcinZielinski	MarcinZielinski	m.zielinski83@gmail.com	ADMIN
3	SzymonNiedzwiecki	Szymon Niedzwiecki	szymon.niedzwiecki@googlemail.com	USER
4	TomaszBednarski	Tomasz Bednarski	mr.tomasz.bednarski@gmail.com	USER
5	Eryk	Eryk Czerwinski	eryk.czerwinski@uj.edu.pl	USER
7	WojtekKrzemien	Wojtek Krzemien	wojciech.krzemien@if.uj.edu.pl	USER
8	Marta	Marta	ciszewska.m.e@gmail.com	USER
9	Iza.balwierz	Izabela Balwierz	iza.balwierz@gmail.com	INACTIVE
10	Lukasz	Łukasz Kaplon	lukaszkaplon@gmail.com	USER
11	Artur	Artur Słomski	artur.slomski@uj.edu.pl	USER

Figure 4.23: Administration panel with users list.

this operation is named: *synchronizeDatabaseAction*. Implementation code for this functionality is shown in Fig. 4.24. In the first step Doctrine manager download user data by *Petuser* entity, from J-PET User Database. Next function iterates over the J-PET users data, and for each user object data is compared with the existing in the local database. In case when the data does not exists in the local database then automatically information is copied to the Logbook database. However in the case when the user already exists in the Logbook database only data which has been changed is updated. The synchronization takes place only in one way from the external database to the local. Any changes made on the local database will not affect the external J-PET user database.

```

public function synchronizeDatabaseAction(){
    $usersMediaWiki = $this->getDoctrine()
        ->getRepository('AppBundle:Petuser', 'mediawiki')
        ->findAll();

    $em = $this->getDoctrine()->getManager();

    foreach ($usersMediaWiki as $userMediaWiki) {

        $user = $em
            ->getRepository('AppBundle:User')
            ->find($userMediaWiki->getUserid());

        $userExist = $user;

        if(!$userExist){
            $user = new User;
        }

        $user->setNickName($userMediaWiki->getUserName());
        $user->setRealName($userMediaWiki->getUserRealName());
        $user->setEmail($userMediaWiki->getUserEmail());
        $user->setPassword($userMediaWiki->getUserPassword());

        $em->persist($user);

        if(!$userExist){
            $user->setId($userMediaWiki->getUserid());
            $user->setRole('USER');
            $metadata = $em->getClassMetadata(get_class($user));
            $metadata->setIdGeneratorType(\Doctrine\ORM\Mapping\ClassMetadata::GENERATOR_TYPE_NONE);
        }
    }

    $em->flush();

    return $this->redirect($this->generateUrl('user_admin_index'));
}

```

Figure 4.24: Function *synchronizeDatabaseAction* to synchronize users from external J-PER User Database to local database.

The view which is displayed based on the local Logbook database shows the table including the typical information about each user like: user id, nickname, real name, email and user role. The user table was prepared using the Angular.js and SmartTable module which allows sorting the table by any column. Furthermore, it allows to asynchronous data loading which prevents each time to refresh the page. Additionally, the table was equipped with the search fields, enabling to find user by any of the displayed parameters.

Supplementary, the local Logbook database was equipped with the functionality of assigning roles to each user independently of the central database. The user roles are organized in three local access levels:

1. Administrator - can administrate users, logs and shifts.
2. User - can log in and use application.
3. Inactive - can not log in and use application.

In such an approach one can control access to the application on the local level making much easier to comprehend users administration.

5. Summary

This thesis aimed at the designing, developing and implementing the fully functional Electronic Logbook for Jagiellonian Positron Emission Tomography group. Created Electronic Logbook helps in the information flow and logging every laboratory activities in the experiment. System allows to add, view, delete and search logbook posts. Every single post is supplemented with text, media and the experimental data. Moreover we prepared the Error and Warning Handling system which allows to report all unexpected events occurred during the laboratory work. Design system has been also enhanced with the Shift Management functionality to help in better planing of laboratory work. Application was equipped with the administration panel enabling user management.

The Logbook application was created using the modern and innovative solutions available for developers in the domain of implementation of web applications. The application was prepared in the Client-Server approach, where as a core back-end technology we have utilized the Symfony2 framework and Model-View-Controller (MVC) design pattern, together with the Doctrine and TWIG template engine. Additionally to enrich the presentation layer of the application we have used the Twitter Bootstrap web framework, jQuery and Angular.js technology. We also used additional Open Source libraries like CKEditor, MathJax, Moment.js and some Angular.js and Symfony2 modules. The store data we have utilized features offered by the MySQL and PostgreSQL database management systems.

According to the methodology of creation of modern web applications, Logbook system was made in Responsive Web Design pattern (RWD). This results, in proper appearance of the application in regardless, of used device like PC, tablet or smarthphone. The designed front-end layer of the application is minimalistic and intuitive which is a very important feature for the users. In some aspects of data presentation we have used the asynchronously data loading to overcome the necessity of reloading the view.

A. Installation of the application

In order to install the Electronic Logbook application on a web server one need to configure it in a proper way. Bellow we present a short instruction how to install developed application on a server side:

1. Execute query from file database.sql in MySQL database
2. Install Symfony2 [10] version 2.6.3
3. Install ornicar/gravatar-bundle [59], friendsofsymfony/rest-bundle [60] and jms/serializer-bundle [61]

(a) Add these bundles to your project composer.json

```
"require": {
    "ornicar/gravatar-bundle" : "dev-master",
    "friendsofsymfony/rest-bundle": "~1.5",
    "jms/serializer-bundle": "0.13.*"
}
```

(b) Run composer update to install the bundles and regenerate the autoloader

```
$ composer.phar update
```

(c) Add these bundles to your application's kernel:

```
// application/ApplicationKernel.php
public function registerBundles()
{
    return array(
        // ...
        new Ornicar\GravatarBundle\OrnicarGravatarBundle(),
        new FOS\RestBundle\FOSRestBundle(),
        new JMS\SerializerBundle\JMSSerializerBundle(),
        // ...
    );
}
```

4. Copy src/AppBundle to src/AppBundle
5. Replace app/config in app/config
6. Copy web/ to web/ (ckeditor, js, css, images, fonts)
7. Create web/uploads/images/ path
8. Setup app/config/parameters.yml or app/config/parameters_dev.yml file with database parameters

9. Login on yourdomain.com by login: Administrator and password: Administrator
 10. Go to yourdomain.com/user/admin and click "Synchronize Users" to synchronize user database with J-PET User Database
 11. From this time, administrator password is changed
-

Podziękowania

Składam serdeczne podziękowania niezastąpionemu promotorowi - doktorowi Marcinowi Zielińskiemu za pomoc w wyborze tematu pracy magisterskiej oraz za mobilizację do napisania jej w języku angielskim. Za miłą atmosferę współpracy. Za każdą godzinę spędzoną na dyskusji. Za każdego wymienionego maila. Za każdą sugestię dotyczącą pracy. A przede wszystkim, za wsparcie oraz nieocenioną pomoc podczas jej redakcji.

Dziękuję grupie J-PET i Zakładowi Fizyki Jądrowej za umożliwienie wykonania tej pracy w ramach prowadzonych przez nich badań.

Dziękuję również kierownikowi studiów - doktorowi habilitowanemu Pawłowi Górze za wszelką pomoc oraz dobrą radę podczas całego toku nauki.

Pragnę również podziękować wspaniałemu sekretariatowi Informatyki Stosowanej - magister Annie Czeluśniak oraz Ewie Łanoszce za wszelką pomoc, szczerą uśmiech oraz zawsze miłe słowo podczas pokonywania każdego kolejnego etapu studiów.

Szczególnie dziękuję rodzinie i przyjaciołom za wiarę oraz nieustanne wsparcie w dążeniu do celu.

Bibliography

- [1] P. Moskal, P. Salabura, M. Silarski, J. Smyrski, J. Zdebik, M. Zieliński, Bio-Algorithms and Med-Systems, Vol. **7**, No. 2, 2011, pp. 73-78.
- [2] P. Moskal, et al., Nucl. Instr. and Meth. A **775** 54-62 (2015).
- [3] P. Moskal, et al., Nucl. Instr. and Meth. A **764** 317-321 (2014).
- [4] P. Moskal, et al., Nuclear Medicine Review **15** C68 (2012).
- [5] MediaWiki: <https://www.mediawiki.org>
- [6] Andreas Mauthe, Peter Thomas, *"Professional Content Management Systems: Handling Digital Media Assets"*, John Wiley & Sons (2004).
- [7] Wordpress: <https://wordpress.org>
- [8] Drupal: <https://www.drupal.org>
- [9] Joomla!: <http://www.joomla.org>
- [10] Włodzimierz Gajda, *"Symfony 2 od podstaw"*, Helion (2012).
Symfony2: <http://symfony.com>
- [11] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *"Design Patterns: Elements of Reusable Object-Oriented Software"*, Addison-Wesley (1994).
- [12] Hasin Hayder, *"Programowanie obiektowe w PHP 5"*, Helion (2009).
PHP5: <http://www.php.net>
- [13] MIT license: <http://opensource.org/licenses/MIT>
- [14] SensioLabs: <https://sensiolabs.com>
- [15] Lausen Georg, Vossen Gottfried, *"Obiektowe bazy danych"*, Helion (2000).
- [16] OpenSorce: <http://opensource.org>
- [17] Propel: <http://propelorm.org>
- [18] Doctrine: <http://www.doctrine-project.org>

-
- [19] Twig: <http://twig.sensiolabs.org>
- [20] PHPUnit: <https://phpunit.de>
- [21] XML: <http://www.w3.org/XML>
- [22] YAML: <http://yaml.org>
- [23] Martin Fowler, *"Domain Specific Languages"*, Addison-Wesley (2010).
- [24] Jason Beard, James George, *"Niezawodne zasady web designu. Projektowanie spektakularnych witryn internetowych. Wydanie III"*, Helion (2015).
Graphical user interface (GUI): <http://www.linfo.org/gui.html>
- [25] Chuck Hudson, Tom Leadbetter, *"HTML5. Podręcznik programisty"*, Helion (2013).
HTML5: <http://www.w3.org/html>
- [26] Witold Wrotek, *"CSS3. Zaawansowane projekty"*, Helion (2015).
CSS3: <http://www.w3.org/Style/CSS>
- [27] Shelley Powers, *"JavaScript. Wprowadzenie"*, Helion (2012).
JavaScript: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [28] Syed Fazle Rahman, *"Bootstrap. Tworzenie interfejsów stron WWW. Technologia na start!"*, Helion (2015).
Bootstrap: <http://getbootstrap.com>
- [29] FontAwesome: <http://fontawesome.github.io/Font-Awesome>
- [30] Paweł Mikołajewski, *"jQuery. Kod doskonały"*, Helion (2012).
jQuery: <https://jquery.com>
- [31] Dariusz Kalbarczyk, Arkadiusz Kalbarczyk, *"AngularJS. Pierwsze kroki"*, Helion (2015).
Angular.js: <https://angularjs.org>
- [32] Twitter Community: <https://engineering.twitter.com/opensource>
- [33] Thoriq Firdaus, *"Responsive Web Design. Nowoczesne strony WWW na przykładach"*, Helion (2014).
Responsive Web Design (RWD): <http://alistapart.com/article/responsive-web-design>
- [34] Bootstrap customization: <http://getbootstrap.com/customize/>
- [35] jQuery Foundation Members: <https://jquery.org/members>
- [36] Ling Liu, Tamer M. Ozsü (Eds.), *"Encyclopedia of Database Systems"*, Springer (2009).
-

-
- [37] Michele Davis, Jon Phillips, *"PHP i MySQL. Wprowadzenie. Wydanie II"*, Helion (2012).
MySQL: <https://www.mysql.com>
- [38] Richard Stones, Neil Matthew, *"Bazy danych i PostgreSQL. Od podstaw"*, Helion (2002).
PostgreSQL: <http://www.postgresql.org.pl>
- [39] Microsoft SQL Server: <http://www.microsoft.com/pl-pl/server-cloud/products/sql-server>
- [40] Oracle Database: <http://www.oracle.com/pl/database/overview/index.html>
- [41] Sybase Database: <http://www.sybase.com.pl/>
- [42] IBM DB2: <http://www-01.ibm.com/software/data/db2/>
- [43] C. J. Date with Hugh Darwen, *"A Guide to the SQL standard : a users guide to the standard database language SQL"*, Addison Wesley (1997).
- [44] S. Sumathi, S. Esakkirajan, *"Fundamentals of Relational Database Management Systems"*, Springer (2008).
- [45] ISO/IEC 9075-1:2003: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=34132
- [46] Stonebraker, Michael with Moore Dorothy, *"Object-Relational DBMSs: The Next Great Wave"*. Morgan Kaufmann Publishers (1996).
- [47] ISO/IEC 9075-1:2011: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53681
- [48] GIT: <https://git-scm.com/>
- [49] CVS: <http://www.nongnu.org/cvs/>
- [50] GitHub: <https://github.com/>
- [51] CKEditor: <http://ckeditor.com>
- [52] MathJax: <https://www.mathjax.org>
- [53] Moment.js: <http://momentjs.com>
- [54] ngInfiniteScroll module for AngularJS: <https://github.com/sroze/ngInfiniteScroll>
- [55] ngAnimate module for AngularJS: <https://docs.angularjs.org/api/ngAnimate>
- [56] Gravatar module for AngularJS: <https://github.com/wallin/angular-gravatar>
-

-
- [57] Smart Table module for AngularJS: <https://github.com/lorenzofox3/Smart-Table>
- [58] Responsive Calendar module for AngularJS: <https://github.com/twinssbc/AngularJS-ResponsiveCalendar>
- [59] ornicar/GravatarBundle: <https://github.com/ornicar/GravatarBundle>
- [60] friendsofsymfony/rest-bundle: <https://github.com/FriendsOfSymfony/FOSRestBundle>
- [61] jms/serializer-bundle: <https://github.com/schmittjoh/JMSSerializerBundle>
- [62] Bootstrap DateTimePicker: <https://github.com/Eonasdan/bootstrap-datetimepicker>
- [63] Bootstrap Dialog: <https://github.com/nakupanda/bootstrap3-dialog>
- [64] Workbench: <https://www.mysql.com/products/workbench>
- [65] TinyBlob: <https://dev.mysql.com/doc/refman/5.0/en/blob.html>
- [66] R. Rivest, "*The MD5 Message-Digest Algorithm*", RFC: 1321 (1992).
- [67] PHP Streams function: `stream_get_contents` <http://php.net/manual/en/function.stream-get-contents.php>
- [68] E. Czerwiński, M. Zieliński, et. al., *Bio-Algorithms and Med-Systems* Vol. **10**, Issue 2, 79 (2014).
- [69] PHP PostgreSQL Functions (PDO_PGSQL): <http://php.net/manual/en/ref.pdo-pgsql.php>
- [70] LaTeX: <http://www.latex-project.org>
- [71] Gravatar: <https://gravatar.com>
-