PROYECTO FIN DE GRADO

TÍTULO: ADVANCED SYSTEM OF PLANNING AND ORGANIZATION OF CLASSES AT THE UNIVERSITY

AUTOR: ALVARO GUADAMILLAS HERRANZ

TUTOR (o Director en su caso): DR RADOSLAW CZARNECKI

CENTRO DE LECTURA: CRACOW UNIVERSITY OF TECHNOLOGY

DEPARTAMENTO: FACULTY OF ELECTRICAL AND COMPUTER ENGINEERING

TITULACIÓN: Grado en Ingeniería Telemática

VºBº

Miembros del Tribunal Calificador:

PRESIDENTE: DR ZBIGNIEW MROZEK

TUTOR: DR RADOSLAW CZARNECKI

SECRETARIO: DAMIAN GRELA

Fecha de lectura: 17 **de** febrero **de** 2014

Calificación:

El Secretario,

Abstract

This document is the result of a process of web development to create a tool that will allow to Cracow University of Technology consult, create and manage timetables. The technologies chosen for this purpose are Apache Tomcat Server, My SQL Community Server, JDBC driver, Java Servlets and JSPs for the server side. The client part counts on Javascript, jQuery, AJAX and CSS technologies to perform the dynamism.

The document will justify the choice of these technologies and will explain some development tools that help in the integration and development of all this elements: specifically, NetBeans IDE and MySQL workbench have been used as helpful tools.

After explaining all the elements involved in the development of the web application, the architecture and the code developed are explained through UML diagrams. Some implementation details related to security are also deeper explained through sequence diagrams.

As the source code of the application is provided, an installation manual has been developed to run the project. In addition, as the platform is intended to be a beta that will be grown, some unimplemented ideas for future development are also exposed.

Finally, some annexes with important files and scripts related to the initiation of the platform are attached.

Summary

This project started through an existing tool that needed to be expanded. The main purpose of the project along its development has focused on setting the roots for a whole new platform that will replace the existing one.

For this goal, it has been needed to make a deep inspection on the existing web technologies: a web server and a SQL database had to be chosen. Although the alternatives were a lot, Java technology for the server was finally selected because of the big community backwards, the easiness of modelling the language through UML diagrams and the fact of being free license software. Apache Tomcat is the open source server that can use Java Servlet and JSP technology. Related to the SQL database, MySQL Community Server is the most popular open-source SQL Server, with a big community after and quite a lot of tools to manage the server. JDBC is the driver needed to put in contact Java and MySQL.

Once we chose the technologies that would be part of the platform, the development process started. After a detailed explanation of the development environment installation, we used UML use case diagrams to set the main tasks of the platform; UML class diagrams served to establish

the existing relations between the classes generated; the architecture of the platform was represented through UML deployment diagrams; and Enhanced entity–relationship (EER) model were used to define the tables of the database and their relationships.

Apart from the previous diagrams, some implementation issues were explained to make a better understanding of the developed code - UML sequence diagrams helped to explain this.

Once the whole platform was properly defined and developed, the performance of the application has been shown: it has been proved that with the current state of the code, the platform covers the use cases that were set as the main target. Nevertheless, some requisites needed for the proper working of the platform have been specified.

As the project is aimed to be grown, some ideas that could not be added to this beta have been explained in order not to be missed for future development.

Finally, some annexes containing important configuration issues for the platform have been added after proper explanation, as well as an installation guide that will let a new developer get the project ready.

In addition to this document some other files related to the project are provided:

- **Javadoc**. The Javadoc containing the information of every Java class created is necessary for a better understanding of the source code.
- **database_model.mwb**. This file contains the model of the database for MySQL Workbench. This model allows, among other things, generate the MySQL script for the creation of the tables.
- **ScheduleManager.war**. The WAR file that will allow loading the developed application into Tomcat Server without using NetBeans.
- **ScheduleManager.zip**. The source code exported from NetBeans project containing all Java packages, JSPs, Javascript files and CSS files that are part of the platform.
- **config.properties**. The configuration file to properly get the names and credentials to use the database, also explained in **Annex II. Example of *config.properties* file**.
- **db_init_script.sql**. The SQL query to initiate the database explained in **Annex III. SQL statements for MySQL initialization**.

Resumen

Este proyecto tiene como punto de partida la necesidad de evolución de una herramienta web existente. El propósito principal del proyecto durante su desarrollo se ha centrado en establecer las bases de una completamente nueva plataforma que reemplazará a la existente.

Para lograr esto, ha sido necesario realizar una profunda inspección en las tecnologías web existentes: un servidor web y una base de datos SQL debían ser elegidos. Aunque existen muchas

alternativas, la tecnología Java ha resultado ser elegida debido a la gran comunidad de desarrolladores que tiene detrás, además de la facilidad que proporciona este lenguaje a la hora de modelarlo usando diagramas UML. Tampoco hay que olvidar que es una tecnología de uso libre de licencia. Apache Tomcat es el servidor de código libre que permite emplear Java Servlets y JSPs para hacer uso de la tecnología de Java. Respecto a la base de datos SQL, el servidor más popular de código libre es MySQL, y cuenta también con una gran comunidad detrás y buenas herramientas de modelado, creación y gestión de la bases de datos. JDBC es el driver que va a permitir comunicar las aplicaciones Java con MySQL.

Tras elegir las tecnologías que formarían parte de esta nueva plataforma, el proceso de desarrollo tiene comienzo. Tras una extensa explicación de la instalación del entorno de desarrollo, se han usado diagramas de caso de UML para establecer cuáles son los objetivos principales de la plataforma; los diagramas de clases nos permiten realizar una organización del código java desarrollado de modo que sean fácilmente entendibles las relaciones entre las diferentes clases. La arquitectura de la plataforma queda definida a través de diagramas de despliegue. Por último, diagramas EER van a definir las relaciones entre las tablas creadas en la base de datos.

Aparte de estos diagramas, algunos detalles de implementación se van a justificar para tener una mejor comprensión del código desarrollado. Diagramas de secuencia ayudarán en estas explicaciones.

Una vez que toda la plataforma haya quedad debidamente definida y desarrollada, se va a realizar una demostración de la misma: se demostrará cómo los objetivos generales han sido alcanzados con el desarrollo actual del proyecto. No obstante, algunos requisitos han sido aclarados para que la plataforma trabaje adecuadamente.

Como la intención del proyecto es crecer (no es una versión final), algunas ideas que se han podido llevar acabo han quedado descritas de manera que no se pierdan.

Por último, algunos anexos que contienen información importante acerca de la plataforma se han añadido tras la correspondiente explicación de su utilidad, así como una guía de instalación que va a permitir a un nuevo desarrollador tener el proyecto preparado.

Junto a este documento, ficheros conteniendo el proyecto desarrollado quedan adjuntos. Estos ficheros son:

- **Documentación Javadoc**. Contiene la información de las clases Java que han sido creadas.
- **database_model.mwb**. Este fichero contiene el modelo de la base de datos para MySQL Workbench. Esto permite, entre otras cosas, generar el script de iniciación de la base de datos para la creación de las tablas.
- **ScheduleManager.war**. El fichero WAR que permite desplegar la plataforma en un servidor Apache Tomcat.

- **ScheduleManager.zip.** El código fuente exportado directamente del proyecto de Netbeans. Contiene todos los paquetes de Java generados, ficheros JSPs, Javascript y CSS que forman parte de la plataforma.
- **config.properties**. Ejemplo del fichero de configuración que permite obtener los nombres de la base de datos
- **db_init_script.sql**. Las consultas SQL necesarias para la creación de la base de datos

Table of contents

Index of figures

Index of tables

1. Introduction

In the last years, the availability of the information has been largely increased thanks mainly to the Technologies of Information and Communications (TIC). Internet and wireless technologies have made possible through a simple click that two people from different parts of the world can easily communicate and exchange information.

This exchange of information has naturally moved to the services of our society: nowadays it is easy to buy a ticket for the bus, do shopping, pay a rental or learn to play guitar if a few hours without getting out of home.

When talking of web development, it is difficult to choose among the amounts of different technologies we have to make possible the creation of the services mentioned above. It is easy to develop a project with a technology that in a few years will be obsolete.

This project is intended to create a web platform that will offer different services for the workers of a University. After explaining clearly which the target of this platform is, a wide inspection will be made among the technologies that can make possible the purpose of the project. Pros and cons will be carefully analyzed in order to make the best choice to avoid the obsolescence of the technologies selected.

After having these technologies chosen, an exercise of software engineering will be done in order to clarify the use and the development of these technologies, and a better understanding of the developed code. As a proof of concept, this platform needs to keep growing, so software engineering is also important for this purpose.

Finally, after showing how it works, some guidelines on what should be the next steps will be discussed. In addition, some Annexes are attached to the project with details about the code; this way, a developer will be able to continue the work with easiness.

For a better understanding of the whole document, some knowledge of TICs, Web Development and Internet protocols should be acquired. Among the technologies we are going to talk about, the main knowledge of the reader should be related to Hypertext Markup Language (HTML), Servers, SQL Databases, Object Oriented Programming and Universal Modeling Language (UML). Nevertheless, every technology mentioned is previously defined, and bibliographical documentation is provided to extend the given explanations.

2. Purpose of the project

The tool that the Cracow University of Technology provides to their users to book a room is currently the webpage http://sale.wieik.pk.edu.pl/. This web page allows to the user to observe a timetable through the following interface:

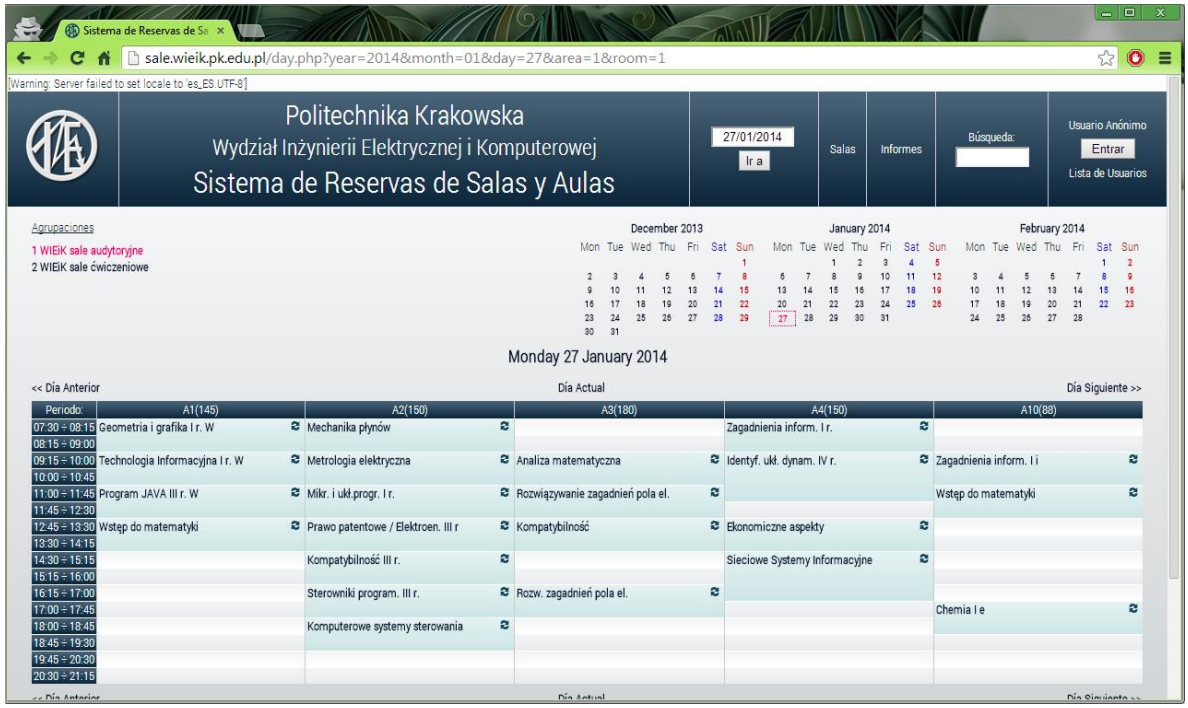

Figure 1. Day information on the web http://sale.wieik.pk.edu.pl/

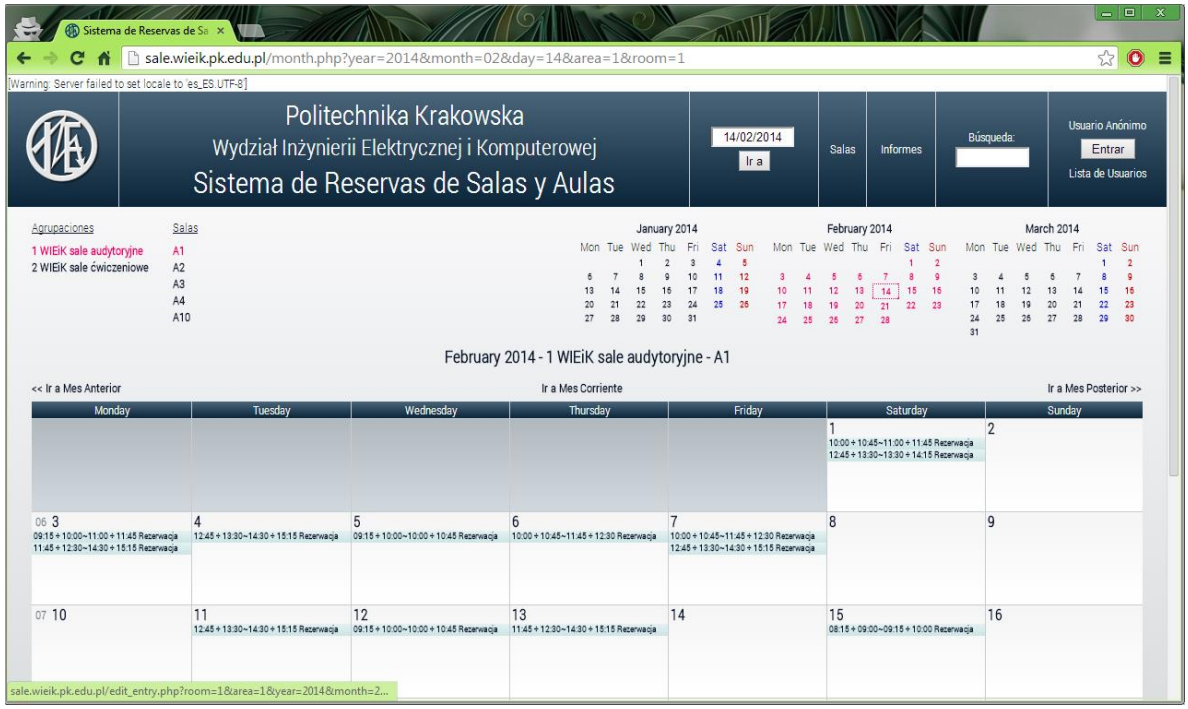

Figure 2. Month information on the web http://sale.wieik.pk.edu.pl/

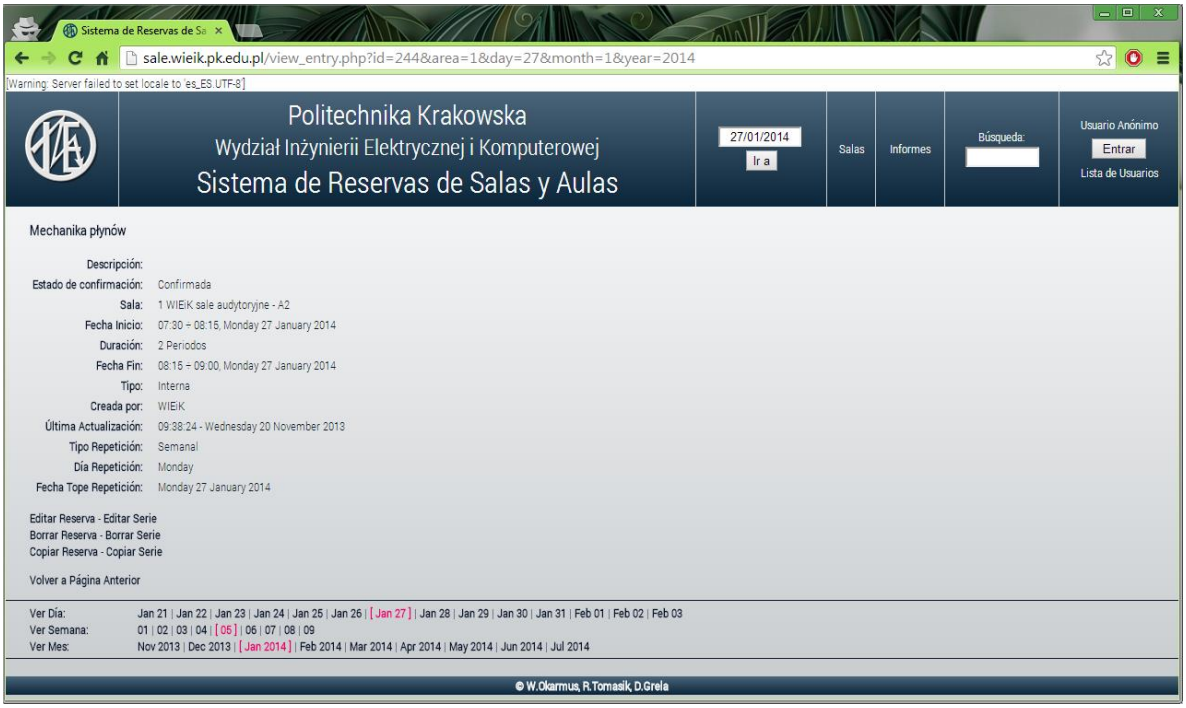

Figure 3. Booking information on the web http://sale.wieik.pk.edu.pl/

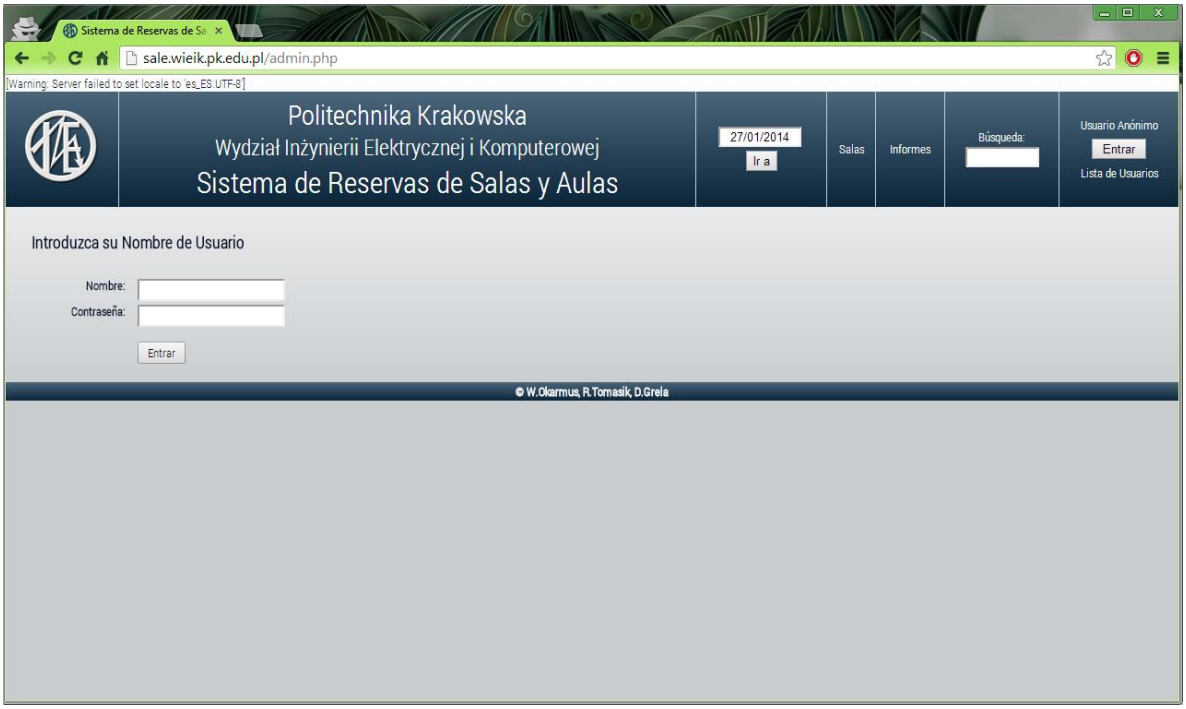

Figure 4. Log in on the web http://sale.wieik.pk.edu.pl/

The features that we can observe in the previous images are:

- Pick a date on the calendar to see what rooms are booked that day [Figure 1. Day information on the web http://sale.wieik.pk.edu.pl/Figure 1].
- Pick a month on the calendar to see the rooms that are booked that month [Figure 2].

- Click on an item of the calendar to see the information related to the booking [Figure 3].
- Log in to access to private features on the platform (make a new reservation).

This tool is a web adaptation of an existing system for booking rooms that Cracow University of Technology internally uses.

The purpose of this project is creating an entirely new system that will expand this model so that the bookings can be shown as timetables. On this new project the booking will be reduced to a part and new options will allow creating, modifying or deleting timetables. The same utile will be used by students and intern personal for a better organization of the distribution of the rooms in the University. This new tool will feature the following operations:

- Creation and management of Subjects, Lecturers, Rooms and Groups elements.
- Creation of timetables along the calendar.
- Access to the platform and the different tools through distinct roles (public user, timetable manager and administrator).
- User management under administrator role.

The project will cover these functionalities mentioned, but it is conceived as a prototype. During the development of the project is not possible the access to the system and the technologies used in the system of the University this product has to be understood as a first step, a proof of concept, to achieve the evolution of the current system.

3. Theoretical part

In this section we will perform a study of the new functionalities to be implemented. This study will reveal which technologies from current fit best our requirements. In first place, we will figure out what requirements are needed for the new version of the platform. After that, we are going to make an inspection of the technologies that can suit these requirements and make a choice of one of them to develop the new platform.

3.1. Requirements

To begin with, we will focus on the requirements needed to make the application work once it has been developed:

Web based tool. As the application to develop is based on the web, the basic tool to develop the platform is a server. Nowadays, in a society based on Technologies of Information and Communication, is easy to find different distributions of servers that can hold our application.

Need of data storage. The storage of information is a common need that has plenty of solutions, being the most accepted the use of Databases. Databases provide an organized collection of data typically organized to model relevant aspects of reality in a way that supports processes requiring this information.

Security. The security is a need in every tool used on a broad scale. This application will have a control of users. This control will imply the use of passwords and sessions. Security is important in order to avoid a user to perform unauthorized operations on the platform. In addition, the exchange of passwords between the platform and the client has to be secure to avoid spoofing attacks.

The database also needs to be properly secured so that the information keeps protected and unmodifiable by unauthorized users.

3.2. Used technologies

As we talked in the introduction, there are a lot of technologies that serve to the same purpose. Each technology has its advantages and disadvantages, and a good election of the proper technology will highly depend on the knowledge of these differences. For this reason, in this section will be made a selection of the technologies based on a previous study of the most popular ones.

3.2.1. Server

For the choice of a server it is necessary to make a previous revision of the programming or scripting languages that allow developing server applications. The most popular programming languages to develop server-side applications according to the site http://w3techs.com/ are shown as follows (1):

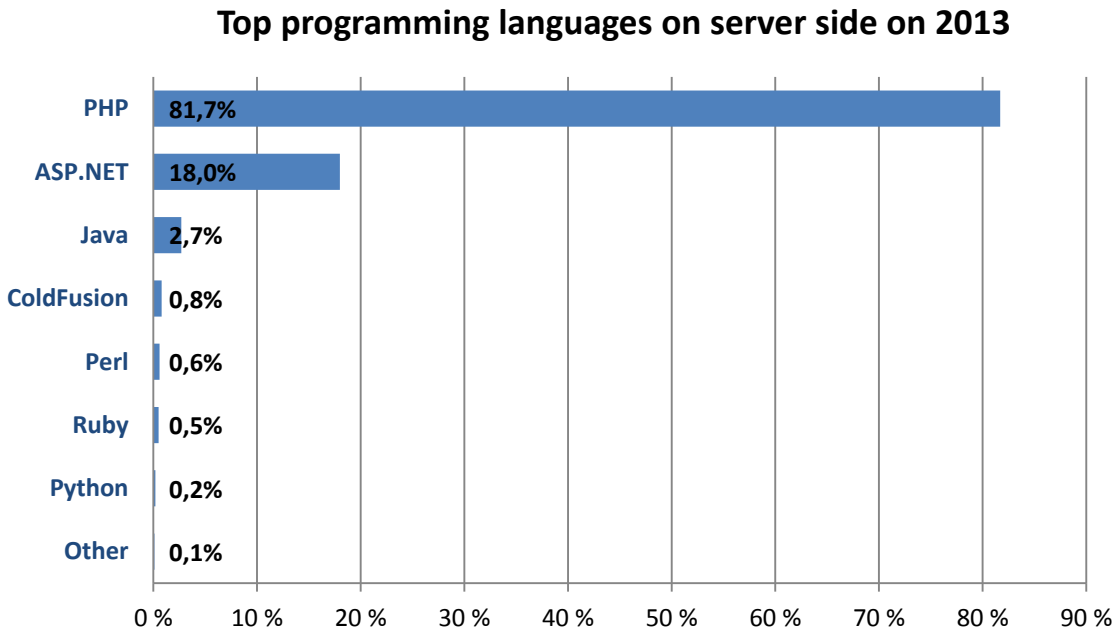

Figure 5. Top programming languages on server side[1].

[1] . **Note**: a website may use more than one server-side programming language.

The choice will be based on one of these programming languages under the premise that if it is popular, it is for a good reason (of course, there are always exceptions). We are first going to have a quick view of these most popular languages to take out their most important features:

- **PHP** (2). *PHP Hypertext Preprocessor* is a widely-used, open source scripting language for the server side. It is an open-source project that can contain text, HTML, CSS, JavaScript, and PHP code. The result of the processed PHP script is returned to the browser as plain HTML. This technology covers the basic functionalities for a server:
  - Generate dynamic page content.
  - Create, open, read, write, and close files on the server.
  - Send and receive cookies.
  - Add, delete, and modify data in a database.
  - Restrict users to access some pages on a website.
  - Encrypt data.
  - Output images, PDF files, Flash movies or any text, such as XHTML and XML.

  Besides, PHP runs on various platforms (Windows, Linux, Unix, Mac OS X…) and is compatible with almost all servers used today, as well as a wide range of databases. PHP has become a multi-paradigm programming language since version 5, supporting both procedural and object-oriented programming.

- **ASP.NET** (3). Classic ASP introduced in 1998 as Microsoft's first server side scripting engine. ASP.NET is a new ASP generation not compatible with Classic ASP, but ASP.NET may include Classic ASP. ASP.NET counts on compiled pages, which makes them faster than Classic. It supports three different development models: Web Pages, MVC (Model–View-Controller) and Web Forms. It includes ASP.NET Razor, a simple markup syntax for embedding server code into ASP.NET web pages (much like Classical ASP). The programming languages involving ASP.NET are Visual Basic and C#. ASP.NET is a framework used in Windows OS.

- **Java** (4)**.** Java is a high-level object-oriented programming language originally developed by Sun Microsystems and released in 1995. This language has become popular thanks to be able to work in any system once is written. This interoperability is achieved thanks to the virtual machine that has to be installed in the different systems, which adapts the high-level code written to the low-level code of the system. Although this is not a language created in its origin to be used in server side programming, his interoperability has derived to many Java-based technologies, being one of the the server-side programming, such as Java Servlets or JavaServlet Pages (JSP). These technologies enable the creation of dynamic, platform-independent method for building Web-based applications and, as are Java, they have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

- **ColdFusion** (5). This technology by Adobe consist of scripting language, ColdFusion Markup Language that compares to the scripting components of ASP, JSP and PHP in purpose and features, but its tag syntax more closely resembles HTML, while its script

syntax resembles JavaScript. ColdFusion is most often used for data-driven websites or intranets, but can also be used to generate remote services such as SOAP web services or Flash remoting. It gathers the basic server functionalities mentioned in the previous technologies with some added particularities connected to Adobe tools, like conversion from HTML to PDF. With the release of ColdFusion MX 6.0, the engine had been re-written in Java and supported its own runtime environment, which was easily replaced through its configuration options with the runtime environment from Sun. Version 6.1 included the ability to code and debug Shockwave Flash.

- **Perl** (6). Perl is a programming language developed by Larry Wall, especially designed for text processing. It stands for "*Practical Extraction and Report Language*". Perl is an Open-Source software, licensed under its Artistic License, or the GNU General Public License (GPL). One of the distinct features of Perl is that it is an interpreted language, so the code can be run as is, without a compilation stage that creates a non-portable executable program. Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others. Among its features, we can underline:
  - Perl's database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.
  - It works with HTML, XML, and other mark-up languages.
  - Perl supports both procedural and object-oriented programming.
  - Perl interfaces with external C/C++ libraries through XS or SWIG.
  - Perl is extensible. There are over 500 third party modules available from the Comprehensive Perl Archive Network (CPAN).
  - It runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

- **Ruby** (7). Ruby is a scripting language designed by Yukihiro Matsumoto. It is a pure object-oriented programming language. Although it is open-source , it is subject to a license. It has a server-side scripting language is similar to Python and Perl, but Ruby can be used to write Common Gateway Interface (CGI) script and can be embedded into HTML. It runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

- **Python** (8). Python is a general-purpose interpreted, interactive (it is possible to interact with the interpreter directly to write your programs), object-oriented and high-level programming language. Python was created by Guido van Rossum in the late eighties and early nineties. Like Perl, Python source code is also now available under the GNU General Public License (GPL). This language has become popular because of its ease to be learnt: it has simple structure, clearly defined syntax and easily maintainable. The Interactive Mode that we mentioned above allows interactive testing and debugging of snippets of code. It is also portable because it has the same interface on all platforms.

Once we have made a small review of all of the most popular languages for server-side integration, it is necessary to decide one to implement the project. All of the products described

above can widely cover the necessities of the project to develop, so we have make a decision based on practical factors. The first factor that will determine this decision is the need of using a GNU GPL license or any Open-Source License.

This project is a prototype, a proof of concept, so there is no need to use licensed software which can achieve the same results. With this first requisite, it is not possible to count on Adobe ColdFusion. ASP.NET has not free license, but it has free version of their development products (Visual Studio Express and Windows Server Express), so that it can be used for this aim.

The second factor is the interoperability of the language. If the project developed will be improved in a future, it would be better not to have to depend on the machine where it will be running. For this reason, ASP.NET, which is highly dependent on Microsoft Systems (although a big amount of Universities and Companies base their systems on Microsoft products with .NET framework), will be discarded.

Another important feature for this purpose is to choose a technology with a big community support. This way, there are more chances that the code can be understood and improved in new revisions, as well as it will be easier to find help on the Internet in case of need. If we discard the minor technologies from Figure 5, there are only PHP and Java technologies left.

To choose one of these “winner” technologies, we should have in mind that we are going to develop a project that can be migrated to other system or language, or can be improved or expanded in a next future. In order to clear the explanation of the code, a fully object-oriented programming language can make use of Universal Modelling Language (UML) to perform any necessary explanation. Although PHP can be coded by using OOP paradigm, Java technology allows only this kind of programming, what helps to avoid bad practices during the development of the application. In addition, in PHP is easy to lose modularity considering that the different views are written often in the same file. Scalability -that is, the property that a system doesn’t get worse efficiency if the number of users increases- is also better handled by Java.

Therefore, although PHP is more used for web development on server side according to surveys, we are going to use Java Servlets and JSPs to develop the tool for timetable managing due to their possibilities to be easily improved and explained.

Once we know the programming language we are using for our server-side application, we have to make a review of the servers supporting this language. Previously we will use again the site w3techs (1) to check the top servers used in the web.

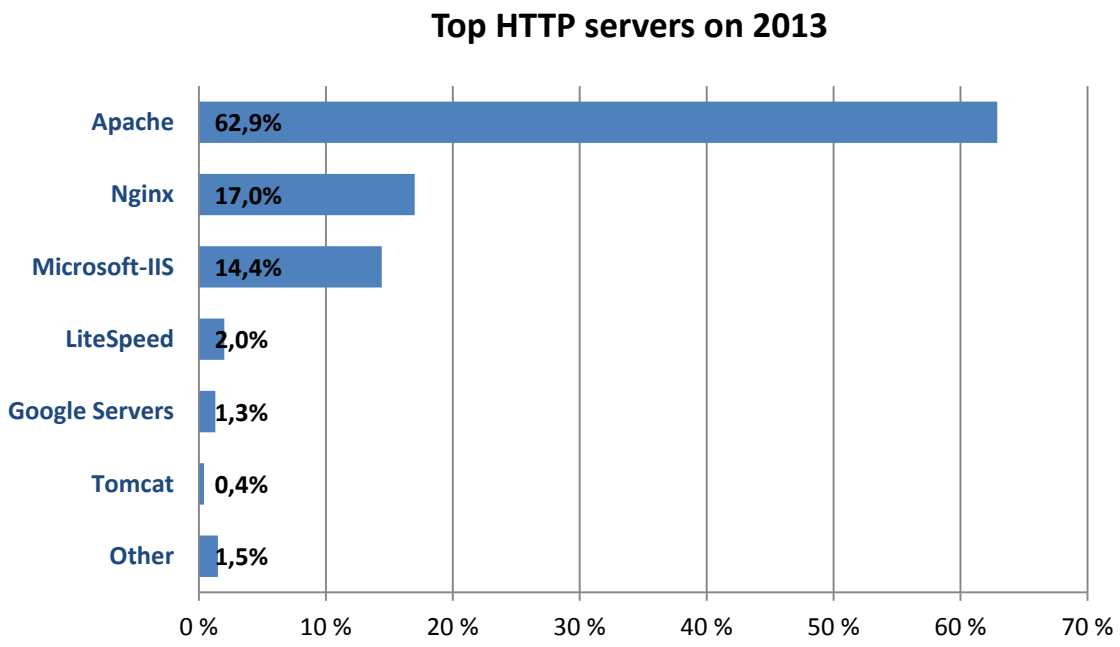

Figure 6. Top HTTP servers used for hosting websites on 2013.

Once more, we take a look on the server listed in this graph to find out whether they are multi-system and whether they support the programming language chosen before, Java:

- **Apache** (9). The Apache HTTP Server is an open-source collaborative project part of the Apache Software Foundation. It has been developed for UNIX, Windows and Macintosh systems and others that implement HTTP/1.1 protocol. Among the programming languages that are implemented on this software, the most popular – Perl, Python, Tcl and PHP – but it is no compatible with Java Servlets directly. Those features are available as add-ons from other Apache projects such as Geronimo and Tomcat.

- **Nginx** (10). Nginx is an open source revers proxy server for HTTP, HTTPS, SMTP, POP3 and IMAP protocols. This server is well known for his high performance with low memory footprint (it uses around 2.5 MB per 10,000 inactive HTTP keep-alive connections thanks to event-driven architecture). Among the languages supported by this server we can find CGI, PHP, Python, Ruby and Perl.

- **Microsoft-IIS** (11). Internet Information Services (IIS) is web server and a set of services for Windows Operative Systems. The services offered by this server include FTP, SMTP, NNTP and HTTP/HTTPS. It is based on different modules which enable the server to process different types of pages as PHR or Perl, apart from ASP and ASP.NET included by default in the system. Microsoft IIS doesn't support Java technologies. IIS is included in latest versions of Windows OS since Windows Server 2003 and Windows XP Professional x64 Edition.

- **LiteSpeed** (12). This is a lightweight proprietary web server which is able to read Apache configurations directly. LiteSpeed implements HTTP 1.0/1.1 and HTTPS protocols and popular scripting languages like PHP, Perl, Ruby or Python. It can also

forward requests to external applications to process and generate dynamic content, applications like CGI, FastCGI, Java Servlets or JSPs. LiteSpeed runs on Linux, FreeBSD, MacOSX and Solaris.

- **Google Servers .** Google Servers are included in this ranking not as accessible servers to develop on, but as servers that host the 1.3 % of web pages used in the survey. The survey made for this ranking is based on the top 10 million websites according to Alexa (13).

- **Tomcat** (14)**.** Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the Java Community Process. It can also feature PHP (not supports PHP 5 and over yet) through external installation. Apache Tomcat is developed in an open and participatory environment and released under the Apache License version 2. This server is often combined with Apache server. As tomcat is written on Java, it can work in any system with JVM installed.

In this occasion, it is necessary to focus on the servers that can run Java Servlets and JSP as it is the chosen technology to develop the project. It is also required a multi-platform server that can work in different operative systems. The only server that easily fits these two requisites is Apache Tomcat, **so the open-source Java-based web server Apache Tomcat will be the distribution used to develop the project**.

3.2.2. Database

The first step to choose a database is to know what kind of data we are going to store on it. For this project, we will first analyze the data that will be necessary to store:

- The main functionality of the application is focused in the generation and modification of timetables based on Subjects, Lecturers, Rooms and Groups of students. These elements have their own attributes that can be diverse.
- The application will also consider the use by different users, so there is a need of storing these users with their different roles for a good user-authentication performance. Although this task is not the main task for a database – there are different authentication services such as LDAP –, in this project we are going to perform this authentication by using the database.

There are two types of database: relational and non-relational. Basically, the relational databases describe the data they store through tables, where each column describes an attribute. It is possible to set relationships between tables. This allows structuring the data and offering simplicity, robustness, flexibility, performance, scalability, and compatibility in managing generic data. They use SQL (Structure Query Language) to retrieve the data. Non-relational are databases that allow bigger scalability at the expense of losing integrity. These kind of database used different methods for storing the data (key/value, document-oriented or graphs are some methods) that don't force the content of the data to follow a structure.

While the relational databases were the first ones to be born, in the last years non-relational databases have lately become popular (we can read some reasons and a deeper explanation of both types in the article "Is the relational database doomed?" in (15)). Here we will give a brief comparison given in (15) between relational databases and key/value databases (non-relational):

Database Definition	
Relational Database	**K/V Database**
• Database contains tables, tables contain columns and rows (tuples), and rows are made up of column values. Rows within a table all have the same schema. • The data model is well defined in advances. A schema is strongly typed and it has constraints and relationships that enforce data integrity. • The data model is based on a "natural" representation of the data it contains, not on an application's functionality. • The data model is normalized to remove data duplication. Normalization establishes table relationships. Relationships associate data between tables.	• Domains can initially be thought of like a table, but unlike a table you don't define any schema for a domain. A domain is basically a bucket that you put items into. Items within a single domain can have differing schemas. • Items are identified by keys, and given item can have a dynamic set of attributes attached to it. • In some implementations, attributes are all of a string type. In other implementations, attributes have simple types that reflect code types, such as ints, string arrays, and lists. • No relationships are explicitly defined between domains or within a given domain.

Table 1. Relational databases compared to key-value databases.

As we are going to manage data with a relational model of data that will not have problems of scalability, **we are going to opt for a relational database** (SQL database). In first place, we will give a brief description of the top 5 DBMS (Database management systems) according to the document (16):

- **Microsoft SQL Server** (17). This a DBMS based on relational model which uses T-SQL and ANSI SQL as query language. MS SQL Server is only free in his express edition and works only in Microsoft OS.

- **Oracle Database** (18). Considered as one of the most complete DBMS is supported by most of the OS, including Windows and Linux. It is distributed in 5 different versions, being the Express Edition the one to be used for free for the purpose of developing, prototyping and running applications.

- **IBM DB2** (19). This DBMS, available for Linux, UNIX and Windows OS, includes products that support the relational model, but in recent years some of these products have been extended to support object-relational features and non-relational structures, in particular XML. Today, there are three main products in the DB2 family: DB2 for Linux, UNIX and Windows (informally known as DB2 LUW), DB2 for z/OS (mainframe), and

DB2 for iSeries (formerly OS/400). A fourth product, DB2 for VM / VSE is also available. There is a DB2 Express-C version, which is a no-cost community edition of DB2; available for download, deployment, and redistribution.

- **MySQL** (20). MySQL is a relational multithread and multiuser DBMS. It is offered under the GNU GPL for any use compatible with this license, but there is also an Enterprise Edition for companies that decide to use special products. It works on multiple platforms, including FreeBSD, Linux, Mac OS, or Windows.

- **Microsoft Access** (21). Access is a DMB included in Office Package by Microsoft which combines the relational Microsoft Jet Database Engine with a graphical user interface and software development tools. g, it provides a special syntax that allows creating queries with parameters, in a way that looks like creating stored procedures, but these procedures are limited to one statement per procedure. As part of Microsoft Office package, it is necessary to buy a license to use it and it only works on Microsoft OS.

Once described these DBMS, we are going to pick one by following the same process as used before: a multi-system, free to use DBMS compatible with Java. With these restrictions, Microsoft SQL Server and Microsoft Access are discarded. IBM DB2 and Oracle Database are powerful DBMS but simplified in their Express Versions comparing with MySQL. Also **MySQL is widely used with a big community backwards**, which fits the philosophy followed in previous choices and makes it the best option of these five technologies.

After this study, the technologies used for the server-side development of the project will be **Apache Tomcat + Java Servlets and JSP + MySQL**. The technology that will allow Java to communicate with MySQL is a java API called **JDBC**.

3.2.3. Integrated Development Environment (IDE)

The Integrated Development Environment will help to develop the project, but it will not have influence in the application itself. There is currently a big variety of IDEs that can handle the technologies chosen for the application, as these are mature technologies with a big background based on years of development and establishment. The two best IDEs that support the technologies chosen are Eclipse (22) and NetBeans (23).

The choice of the IDE is totally up to the developer. It should be based on the previous experience of the programmer and its familiarization with the tool. In this case, NetBeans will be the IDE to develop the project. Some of the features that best fit the project are:

- **Editing and refactoring**. The language-aware NetBeans editor detects errors while typing and assists with documentation popups and smart code completion.

- **Debugger**. The NetBeans Debugger allows placing breakpoints in the source code, adding field watches, stepping through the code, running into methods, taking snapshots and monitoring execution as it occurs.

- **Databases**. NetBeans IDE provides drivers for the Java DB, MySQL, Oracle, and PostgreSQL database servers. It also allows registering any other JDBC driver with the IDE, so that any database that provides a JDBC driver can be explored. It also provides a SQL Editor to open, edit, and run any SQL scripts.

- **Deployment on Servers**. The IDE works with any standard Java Enterprise Edition (Java EE) container, and provides support for GlassFish Server Open Source Edition 3.1.2.2, WebLogic 12c and 11g, Apache Tomcat 7.0 and 6.0, JBoss 6.1, and others.

- **HTTP Monitoring**. In addition to the debugging and profiling support, the IDE provides the HTTP Server-Side Monitor to help diagnose problems with data flow from JSP page and servlet execution on the web server. The HTTP Server-Side Monitor gathers data about HTTP requests that are processed by the servlet engine. For each HTTP request that is processed, the monitor records data about the incoming request and the data states maintained on the server. You can view data, store data for future sessions, and replay and edit previous requests.

3.2.4. Security on the platform

Although the project will not be a highly attacked product because of the information contained, the basic security on this project has to cover three aspects:

- **Protection of data.** The stored data will contain user information such as passwords, identity numbers or mail addresses. This information has to be properly secured to avoid unauthorized propagation of this personal information.

- **User authentication**. The platform will support special tools as timetable creation for determined users. To properly recognize the users, security on authentication will have to be strong enough to avoid possible unauthorized access to these special tools. The authentication process itself has to be also protected to avoid filtering of user credentials.

- **User privileges**. The application will show different content for the different roles of the users registered. It has to make sure that a user can only access his part and avoid possible unauthorized operations.

The implementation of security will stay into the basics because the project is not final software. The project will focus on the functionality and the new possibilities of the timetable managing web tool.

4. Practical part

Now that we know how to begin with the project, it is time to star to develop. On this section it will be explained how the different elements have been deployed, with implementation details. Guidelines on which development tools have been used to ease the process will be also provided.

4.1. Installation of the Environment

As we concluded in the previous section, different tools will be needed to deploy our application: Apache Tomcat, MySQL and JDBC. The installation of these tools can be made separately, but as we are going to use NetBeans IDE, we are going to install more of these tools with this program: it well ease the project and will let a quick deployment when making changes on the code. Nevertheless, information about installation without using NetBeans will be also provided.

4.1.1. NetBeans IDE 7.4

The version of NetBeans used for the development of this project is NetBeans IDE 7.4. From the official website (24), we have access to this version and previous ones. There are different configurations of the NetBeans package depending on the OS or what technologies the user wants to develop. Our distribution of NetBeans will contain tools for developing with Java SE (basic Java applications), Java EE (JSPs and Servlets) and Apache Tomcat 7.0.41 for Windows.

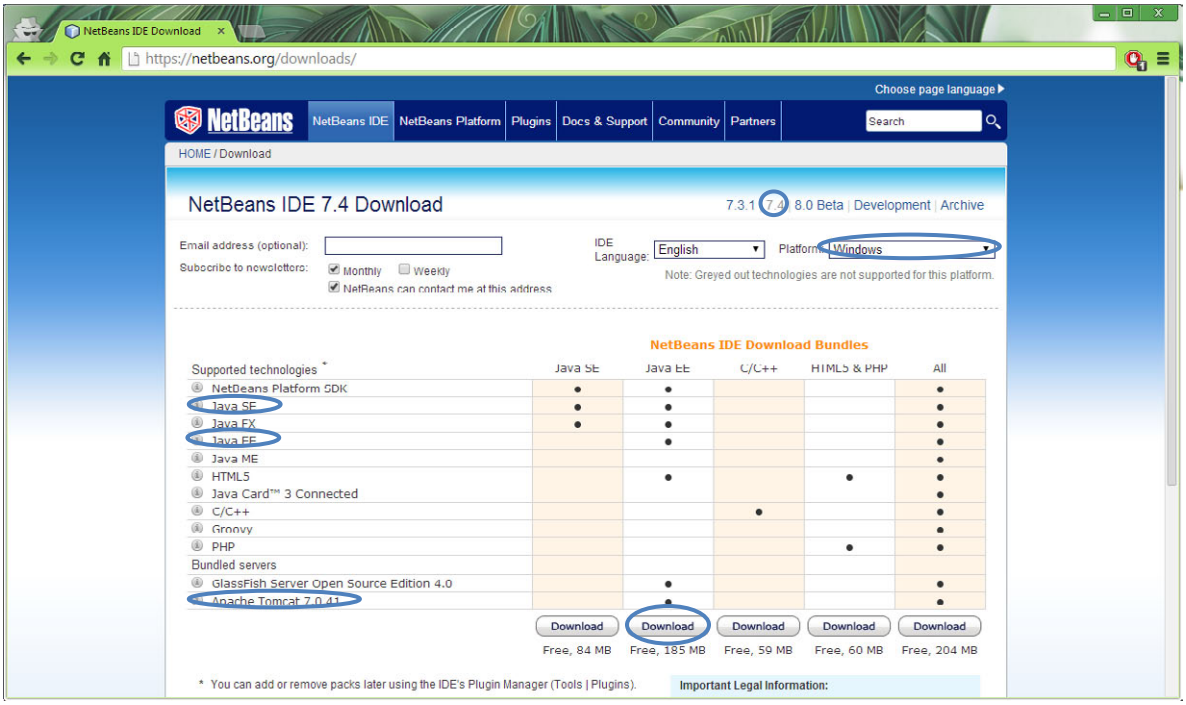

Figure 7. NetBeans IDE 7.4 download configuration

After download, the installation process doesn't require any special configuration, but for further details about installation, the website widely explains this process (25). This package will install both NetBeans and Apache Tomcat 7.0.41.

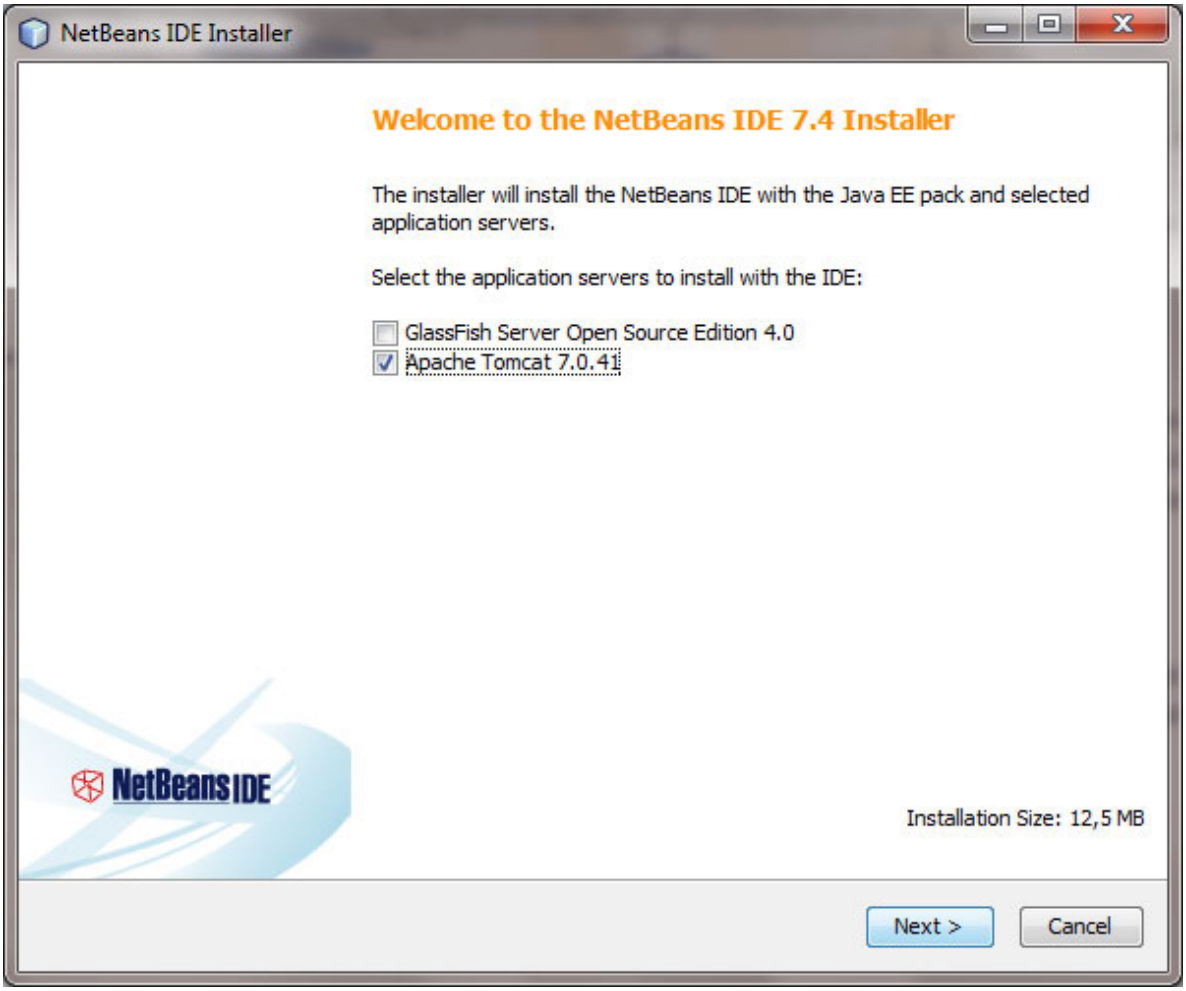

Figure 8. Optional installation of Tomcat during the installation of NetBeans IDE 7.4

4.1.2. Apache Tomcat 7.X

Tomcat can also be downloaded and installed through the official website (26). The easiest way to download the server for windows is downloading the binary distribution. It is a ZIP file with all the needed files to make Tomcat work.

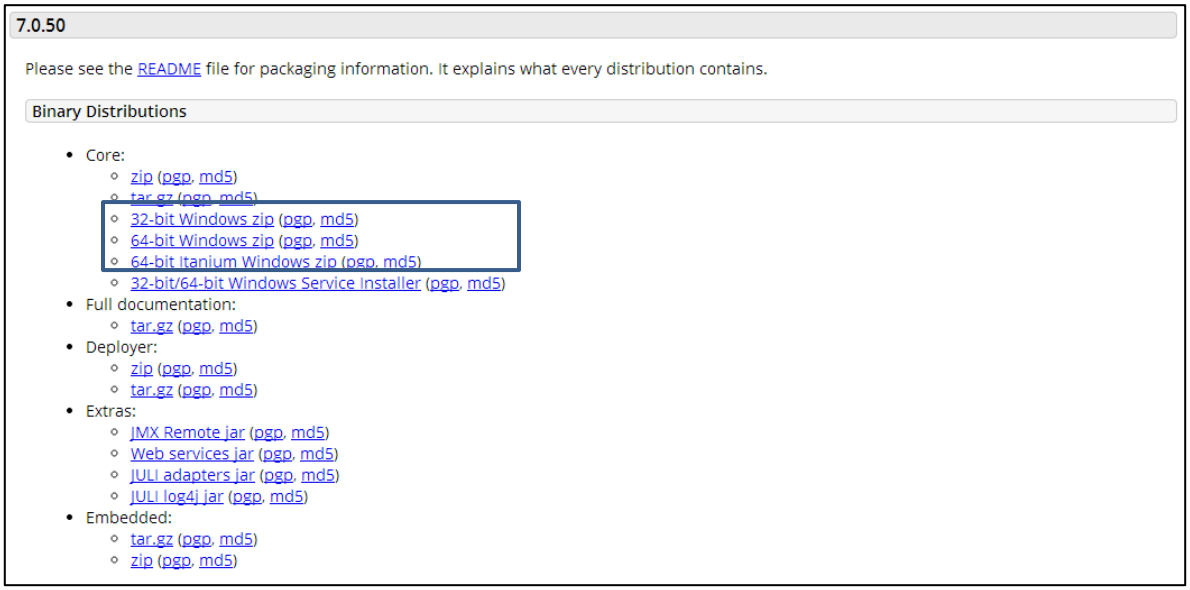

Figure 9. Binary Distributions of Apache Tomcat 7.0.50

Once downloaded the distribution, it is necessary to extract in the path where we want it to run. This path will be known as CATARINA home. As we are not going to start and run Tomcat manually, this will be enough. For manual managing of the server or installation on other systems, it is necessary to consult the documentation provided on the official website (27).

After having installed both Tomcat and NetBeans, we can make NetBeans manage the server. The main advantage of managing Tomcat through NetBeans is the chance of the automatic rebuilt of the project when a piece of code is edited. There is no need of manually stop, start and restart the server or re-build the project.

The configuration of the server is easily done through NetBeans. On the "*Service*" section (top left view of standard distribution of views) there is an option called "*Servers"*. Right click on it and "*Add Servers…*" option will begin the configuration process.

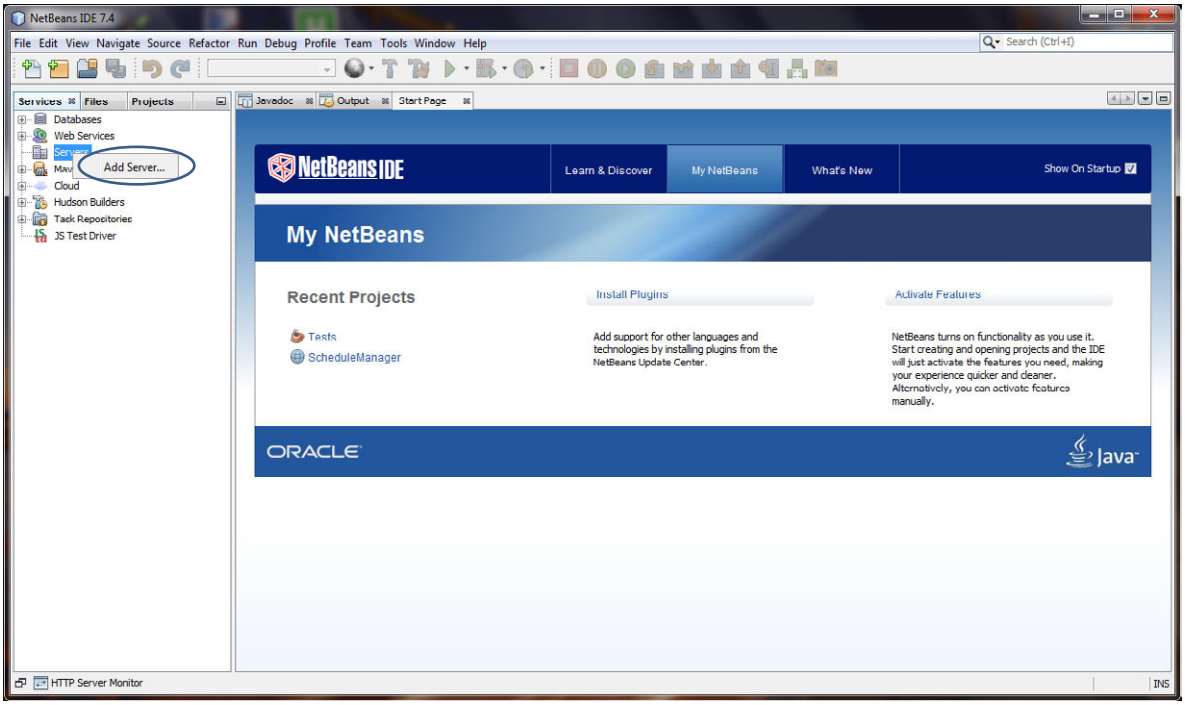

Figure 10. Tomcat installation through NetBeans

Next two steps are picking up a server (Apache Tomcat in this case) and the path where the server will is located (Catalina Home). This path is the directory where the Apache Tomcat ZIP file was extracted or the installation directory selected when installing NetBeans + Tomcat. The user and the password are created to give credentials to the manager of the server.

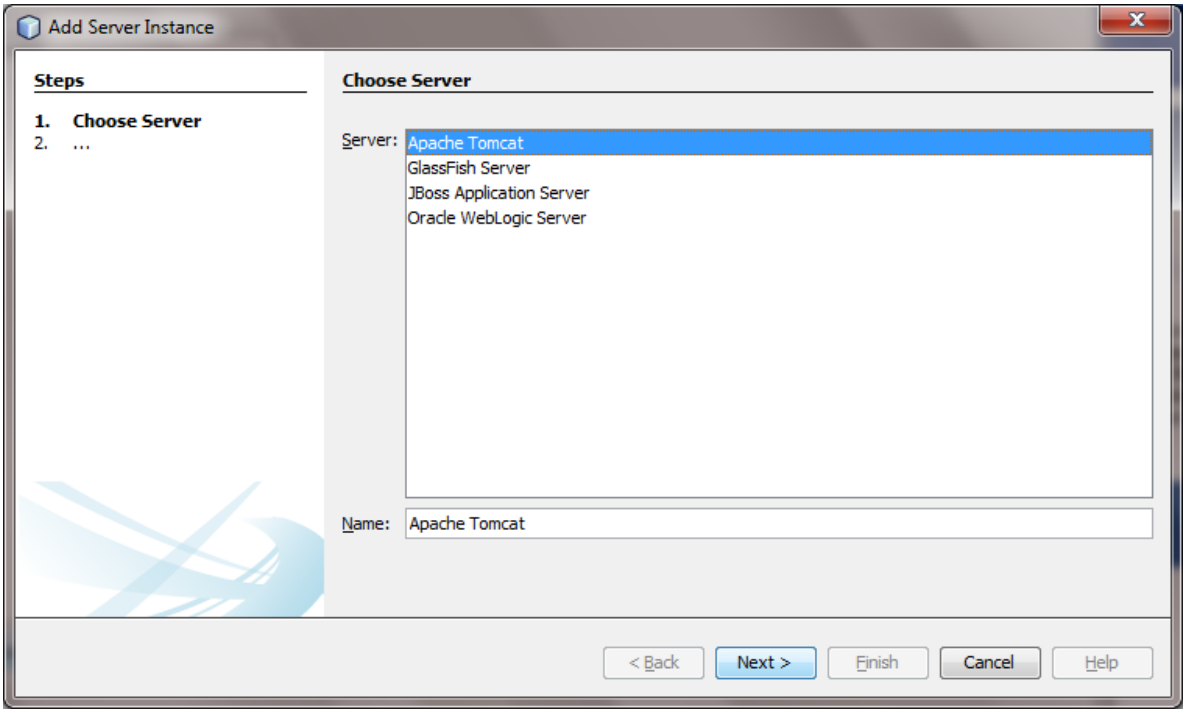

Figure 11. Step 1 of Tomcat installation on NetBeans.

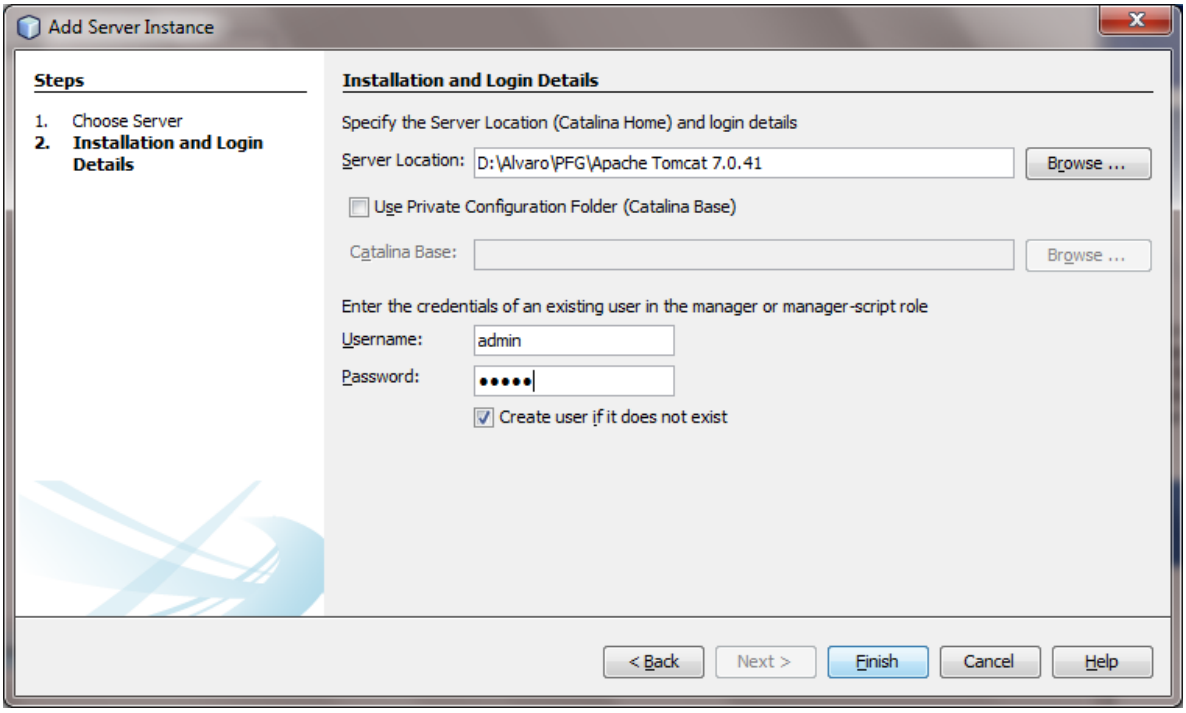

Figure 12. Step 2 of Tomcat installation on NetBeans.

4.1.3. MySQL Community Server 5.6

The installation of MySQL Community Server is a driven process through auto-installer software for Windows (MySQL Installer 5.6 for Windows (28)), or ZIP distribution (29). We are going to use the web installer process - for other operative systems or different download methods, a detailed installation guide can be found in the official website (30).

Figure 13. MySQL Installer 5.6.X

From web installer, the needed products are selected, downloaded and installed. The products that we are going to install are MySQL Server 5.6.X and its documentation.

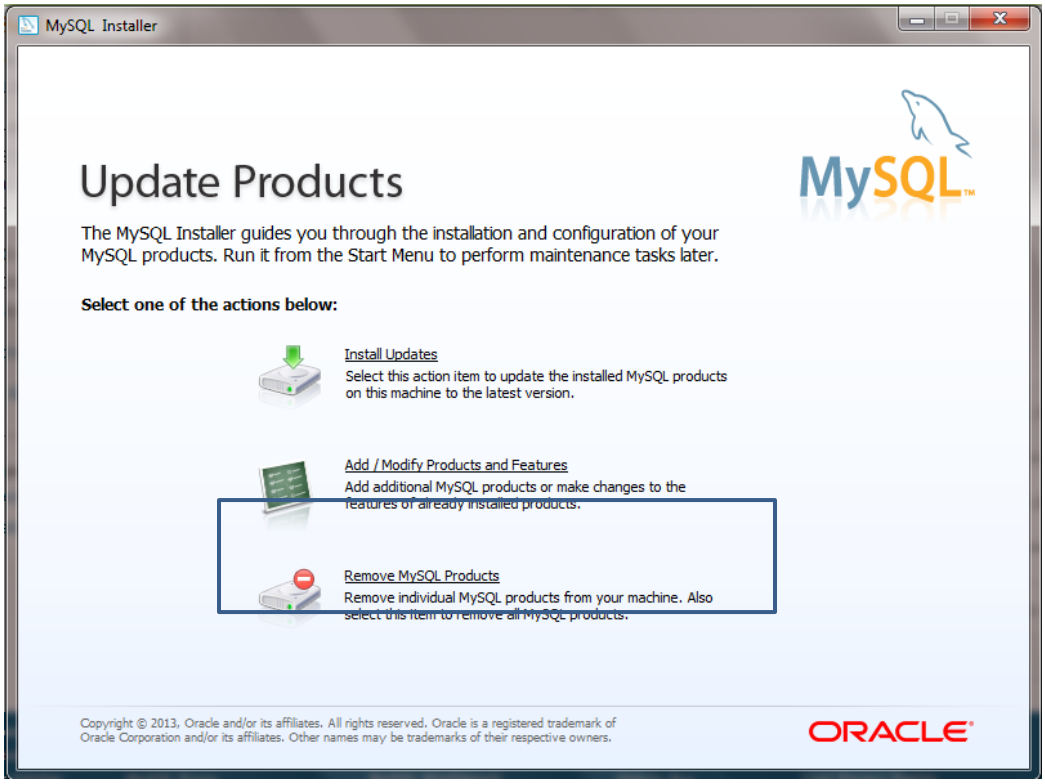

Figure 14. First step of MySQL Server installation.

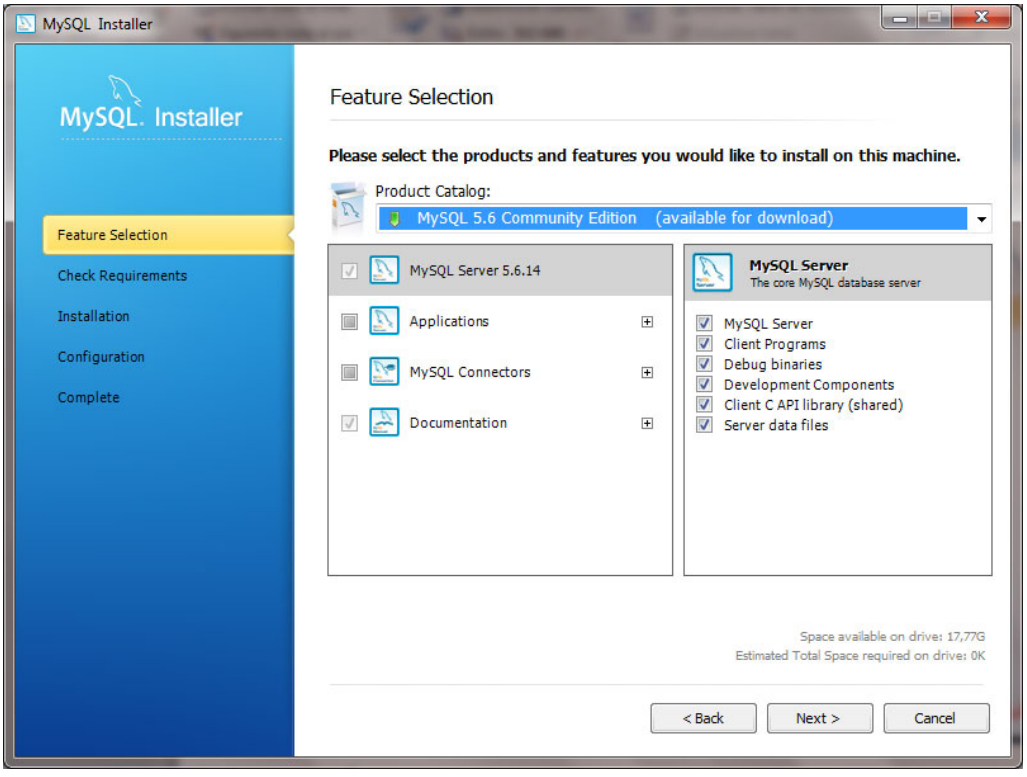

Figure 15. Second step of MySQL Server installation.

Once installed, the server configuration begins. In the configuration, we set up details like users of the database, the name of the database or the type of database that will be created – such us

multifunctional, transactional or non-transactional. No special comments will be given in this part of the configuration.

To run MySQL Server after installation process, a MySQL Command Line Client is installed (31). Nevertheless, we are going to use MySQL Workbench to connect the server, a powerful tool that allows managing, developing, design and administration of databases.

4.1.4. MySQL Workbench 6.0

With a similar process that MySQL Community Server, this software is downloaded from MySQL site (32).

Figure 16. MySQL Workbech 6.0 download for Windows.

The installation process does not require any special configuration apart from the installation directory.

4.1.5. JDBC

The JDBC is a standard to connect Databases, but every Database makes the implementation of this JDBC. MySQL has his JDBC driver called MySQL Connector/J (33), but the package of NetBeans downloaded also includes this connector (Right click on the project > Properties > Libraries > Add Library > MYSQL JDBC Driver).

Figure 17. Download of the JDBC Driver for MySQL

The ZIP downloaded contains a JAR file that we must add to our project CLASSPATH in order to use JDBC.

4.2. Implementation of the project

In this section we will first define some previous concepts that will be used along the rest of the document. With these concepts clarified, we are then going set the design of the project.

For the explanation of the Java part of the server-side we are going to use UML (34) diagrams. This language allows having an easy overview of the project and his architecture and it is possible to deeply explain the implementation details.

For the modelling of the database we are going to use the tools provided by MySQL Workbench (35), a tool by MySQL that allows to model, create and manage the MySQL DBMS.

4.2.1. Previous concepts

Although the project doesn't have deep concepts, a quick view to the main elements that we are going to refer to is good to avoid misunderstandings or ambiguity.

The target of the application is set into a student context. The basics elements we are going to deal with are:

- **Groups**. A group is a set of students that will attend to classes. It can also be considered a set of lecturers or people working in the University that needs to create a timetable. The group usually has a key name that represents why those people belong to the same group – for instances ERASMUS to refer the incoming students from other countries, or COMP_ENG to represent those students belonging to Computer Engineering speciality. In a timetable, the user belongs to one or more groups, and will look for his timetable according to the groups that he belongs to.

- **Subjects**. The subject is any branch of learning considered as a course of study. In the timetable, it will mean what activity will be taken at a specific time, so it can also be

used to simply book a room – The "subject" here would have the name of "Reservation", for instance.

- **Lecturers**. The lecturer is the person that is going to lecture or teach in a field of learning at a university or college. In the schedule, this person will be the one in charge of the group for a specific time.

- **Room**. The room is the physical place where the lecture is going to take place. It will serve to academic reasons or simply to meeting reasons.

- **Schedule**. The basic meaning of the schedule is the time when a lecturer teaches a subject to a group in a room. In addition, with the little ambiguity that can have the elements composing a schedule, it can be taken as the time when a room has been booked by a person (lecturer) for a group.

- **Timetable.** A timetable is a set of schedules that belong to an element of the previous mentioned. For instance, a timetable for a lecturer will show all the schedules that the lecturer has. The timetables are shown from Monday to Friday and they depend on the date (they can be change for different weeks).

4.2.2. Use Case diagrams

The use case diagrams will establish the basic operations that the platform will cover and the different users that will make use of the platform. This will help to clarify the goals of the project to start developing the software.

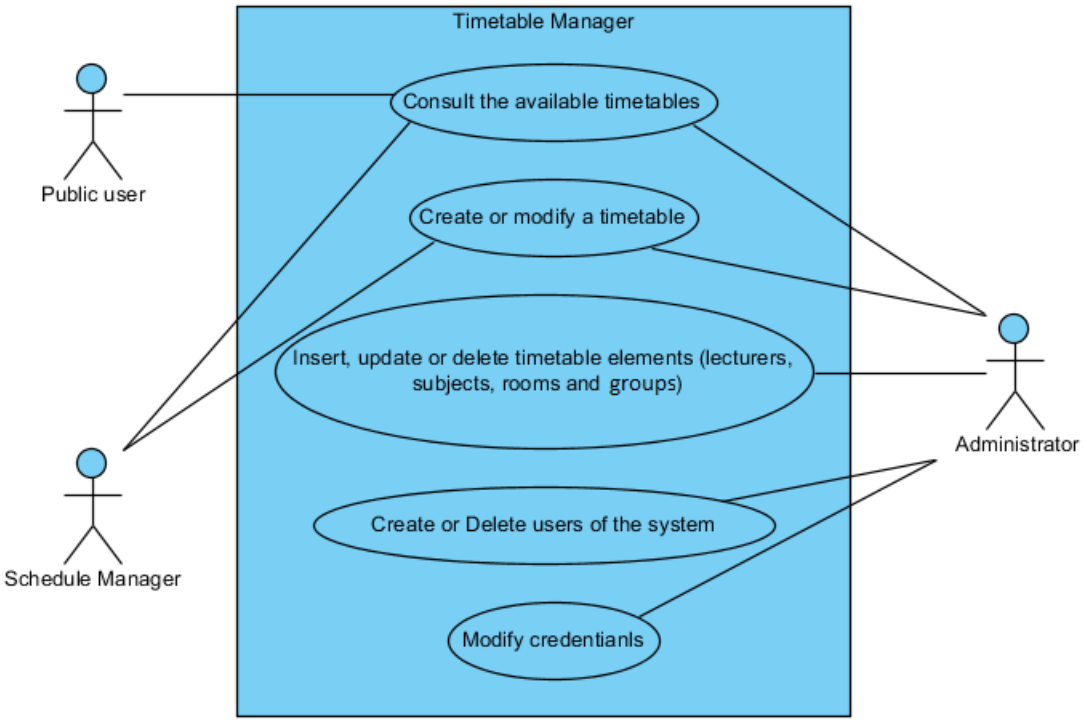

Figure 18. Use case diagram

For the explanation of the use cases we will focus on these elements:

- **Description**. What the use case is about.
- **Main actor**. The main actor that will be involved in the use case.
- **Secondary actor**. The rest of the actors that can be involved in the use case.
- **Pre-conditions**. The state of the application before the use case takes place.
- **Trigger**. The action that makes the use case to begin.
- **Post-conditions**. The state of the application after the use case takes place.

Use case: Consult the available timetables
Description: The user can consult a timetable that has been previously created. The timetables can be searched by lecturer, by group or by room.
Main actor: The public user.
Secondary actor: Schedule Managers and Administrators can also consult the timetables, but if they enter with that role, the purpose will be another usage.
Pre-conditions: The web site is accessed.
Trigger: A group, lecturer or room element of existing is clicked.
Post conditions: A timetable shows information of the time when it takes place and which elements are involved.

Table 2. Description of use case "Consult the available timetables".

Create or modify a timetable
Description: The user can create a timetable or modify an existing one.
Main actor: The schedule manager.
Secondary actor: Administrators can also modify or create timetables.
Pre-conditions: The user has been successfully logged in.
Trigger: An element of the existing ones on the database is clicked.
Post conditions: The new timetable is stored in the database.

Table 3. Description of use case "Create or modify a timetable".

Insert, update or delete timetable elements (lecturers, subjects, rooms and subjects)
Description: The user can create, modify or delete an element of the database.
Main actor:

The administrator.
Secondary actor: None.
Pre-conditions: The user has been successfully logged in.
Trigger: An option in the menu for this operation is clicked.
Post conditions: The element is inserted, updated or deleted in the database.

Table 4. Description of use case "Insert, update or delete timetable elements".

Create or Delete users of the system
Description: The actor can manage the users of the system.
Main actor: The administrator.
Secondary actor: None.
Pre-conditions: The user has been successfully logged in.
Trigger: An option in the menu for this operation is clicked.
Post conditions: The user is created or deleted in the database.

Table 5. Description of use case "Create or Delete users of the system".

Modify credentials
Description: The actor can modify his credentials (password).
Main actor: The administrator.
Secondary actor: None.
Pre-conditions: The user has been successfully logged in.
Trigger: An option in the menu for this operation is clicked.
Post conditions: The credentials are modified in the database.

Table 6. Description of use case "Modify credentials".

For the achievement of these use cases, we are next going to design the class diagram that will establish the classes to code.

4.2.3. Class diagram

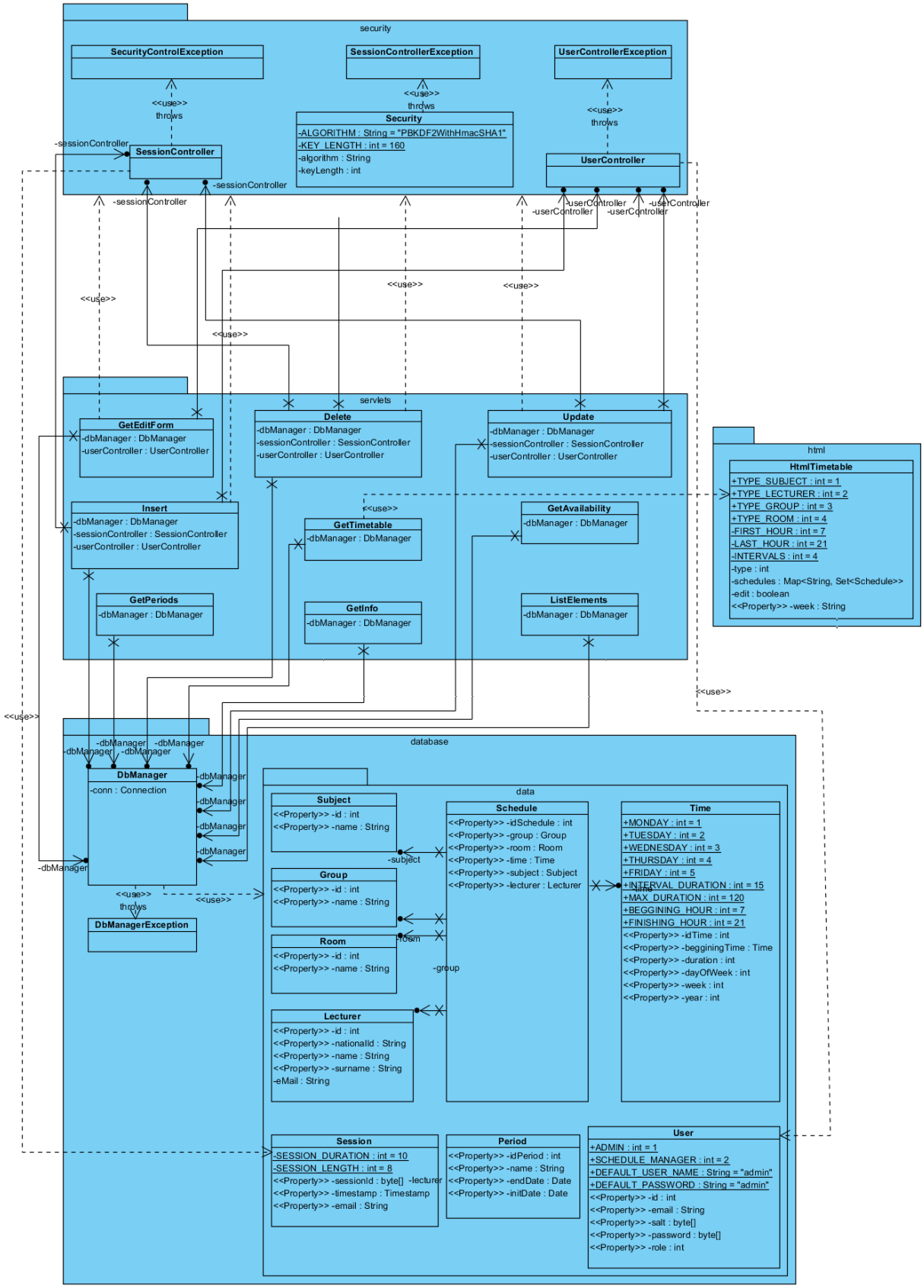

Figure 19.General overview of Class diagram.

The class diagram is composed by five packages. Every package will be explained in detail.

- **Package database**. It contains the classes that will interact with the database.

Figure 20. Package database in detail

The classes contained in this package have the following tasks:

- **DbManager**. This class uses JDBC API (36) to make queries on the database. For this project, this is the class that will transparently contact the database for updating, modifying, deleting or listing operations. The *DbManager* class will use the classes contained in package *database.data* that represent the data stored.
- **DbManagerException**. This Exception will be raised when a database error occurs or when any operation wasn't successful.

▪ **Package database.data**. It contains the classes that represent the data stored in the database. It contains the basic information that the system will deal with, so it is important to clarify what these classes mean.

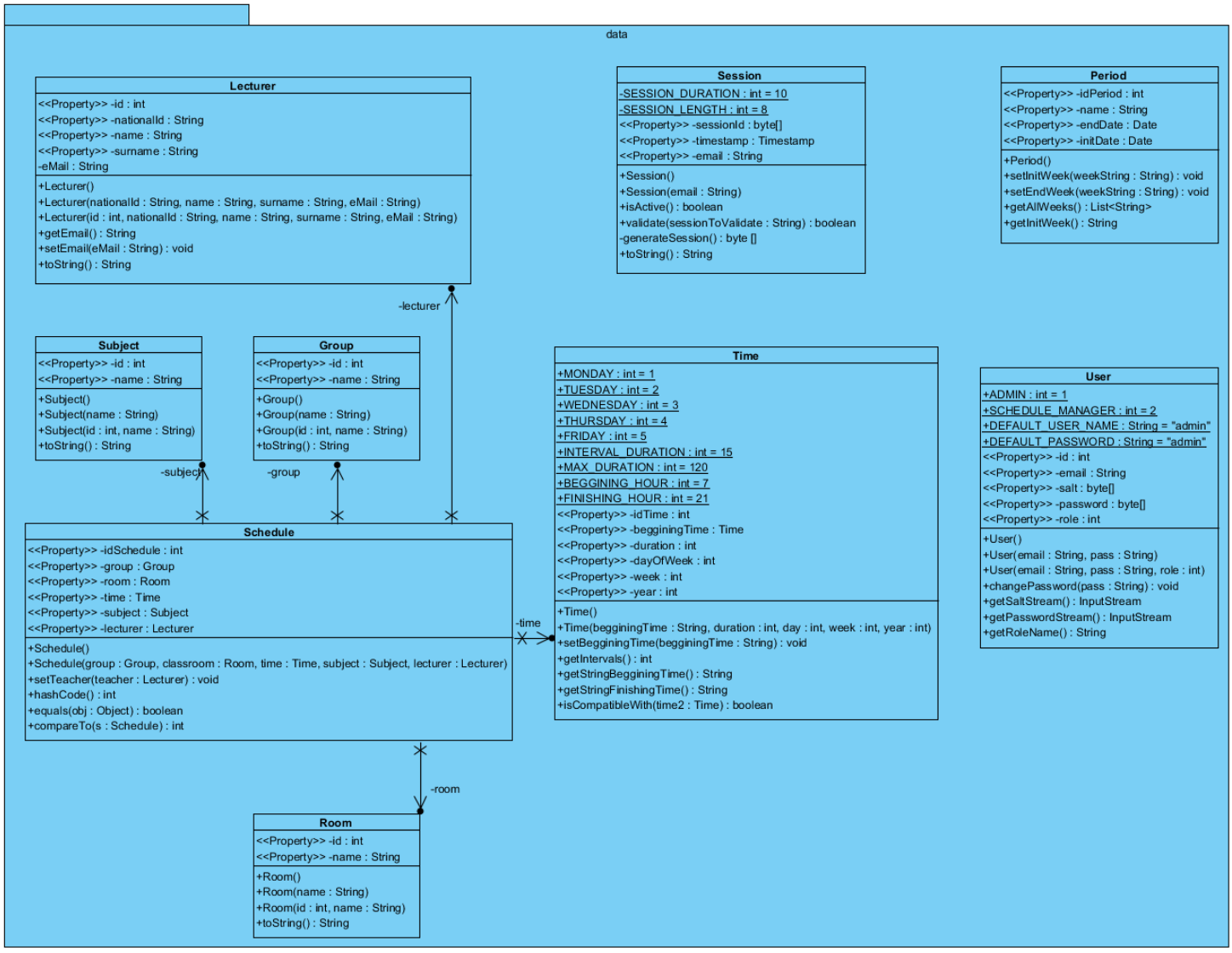

Figure 21. Package database.data in detail

The classes contained in this package have the following tasks:

- **Lecturer.** Represents a lecturer.

- **Group.** Represents a group.
- **Subject.** Represents a subject.
- **Room.** Represents a room.
- **Time.** Represents a time when a schedule takes place. The time indicates the hour, the day of the week, the week of the year and the year.
- **Schedule.** A schedule is the combination of a Lecturer, a Group, a Subject, a Room and a Time. It indicates at what Time the Group has the Subject given by the Lecturer in a Room. A timetable is a set of one or more schedules.
- **Period.** Represents a period of time. The period will be used to let the application insert schedules for a period of time.
- **User.** Represents a user of the application.
- **Session**. Represents a session hold.

- **Package security**. It contains the classes that perform the security issues in the system: password generation, user authentication and session control.

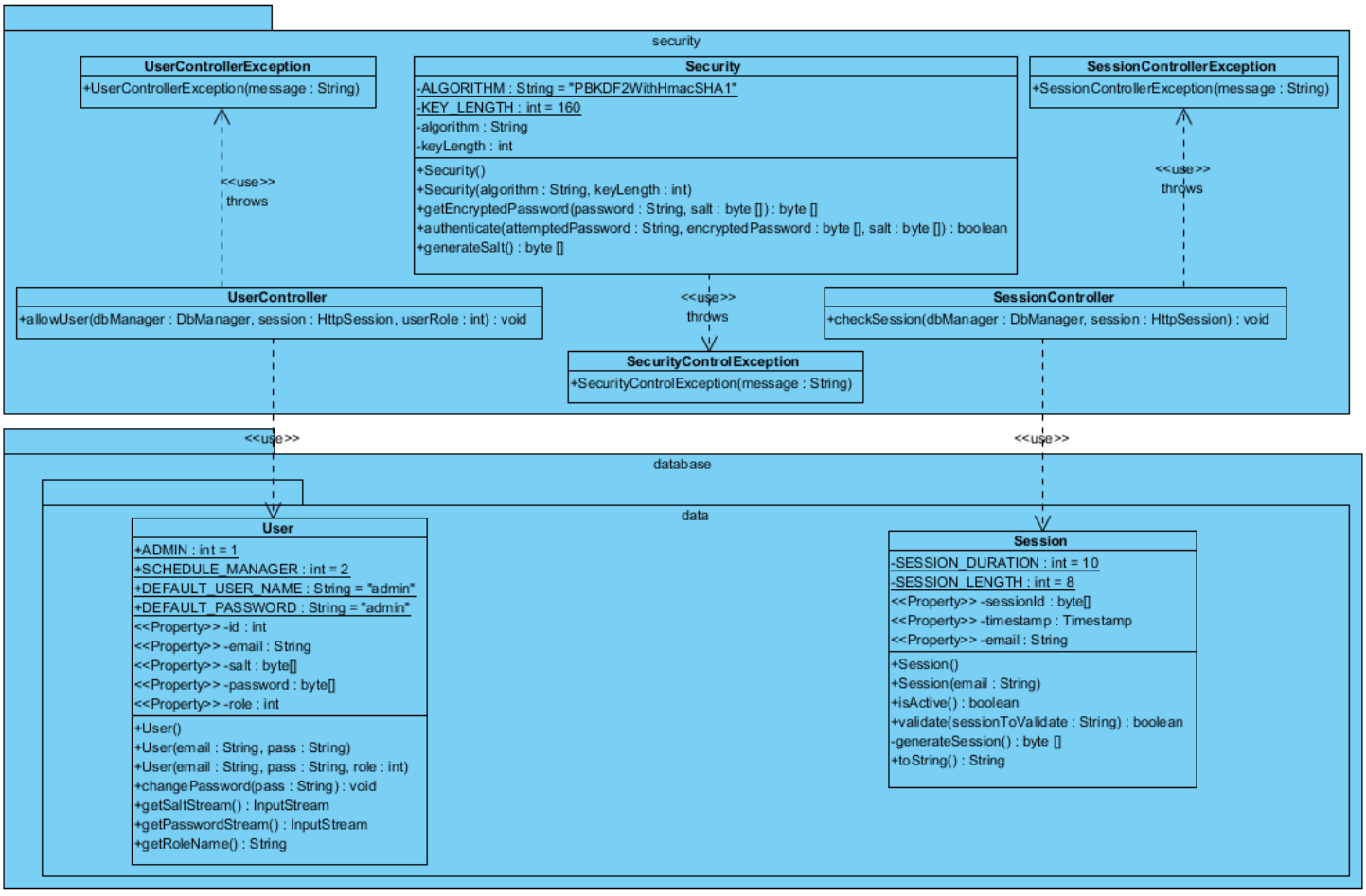

Figure 22. Package security in detail.

The classes contained in this package have the following tasks:

- **Security**. This class performs security operations as the generation of Secure Random Salts or the hashing of a password. See ***¡Error! No se encuentra el rigen de la referencia.*** section for a deeper explanation of these elements. This class throws *SecurityControlException*.
- **SecurityControlException.** This exception is raised when an error related to a Security operation occurs.
- **SessionController.** This class is in charge of checking the authenticity and state of a session. See ***¡Error! No se encuentra el origen de la referencia.*** section for ore details. This class throws *SessionControllerException*.
- **SessionControllerException.** Exception raised when a session is not valid or is not active.
- **UserController.** This class checks if a user has the right role to access to a part of the servlet. To see more, go to the *Requirements* section. This class throws *UserController*.
- **UserControllerException.** This Exception is thrown when the user is not in the role needed to access a part of the application.

- **Package html**. This package contains the classes that represent a HTML targets.

- **Package servlets**. Package containing the servlets that the application will use to generate dynamic webpages. The servlets of this package extend from the class javax.servlet.http.*HttpServlet* (37) which will let them receive HTTP requests and create HTTP responses.

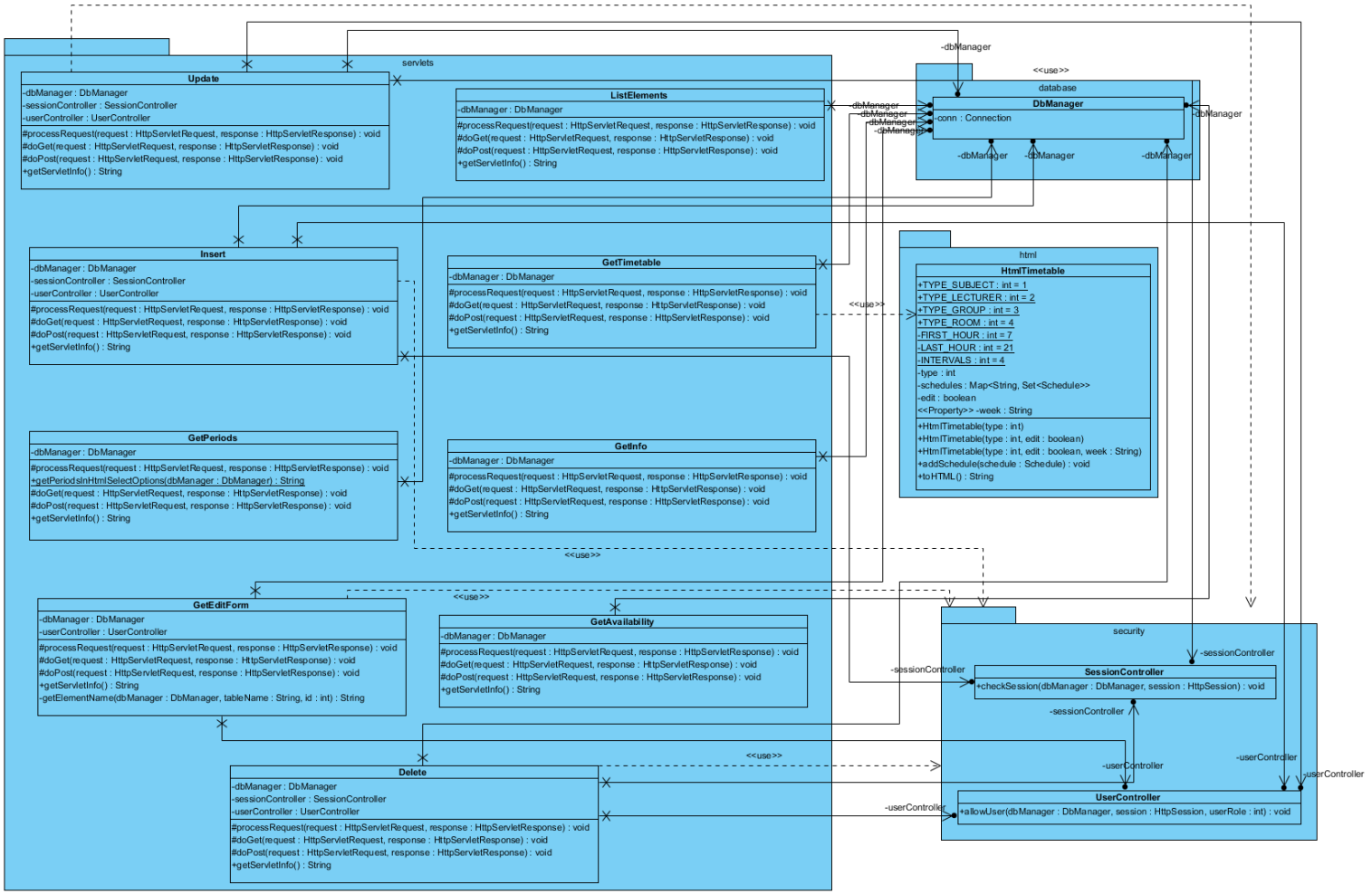

Figure 23. Packages servlets and html in detail

The classes contained in both *html* and *servlet* packages have the following tasks:

- **html.HtmlTimetable**. This class transforms a set of *database.data.Schedule* objects into a HTML table element.
- **servlets.Update**. Updates an element of the database (through a database.*DbManager* object) and gives as answer the result of the operation. This class uses security.*SessionController* and security.*UserController* objects to make sure that the user invoking the servlet is authenticated and authorized to use the servlet.
- **servlets**.**ListElements**. Servlet that generates a HTML list (<ul>) with the elements existing in the database through a database.*DbManager* object. This is a public servlet that doesn't need authentication or authorization.
- **servlets.Insert**. This servlet is involved in the creation a new element and its insertion in the database through a database.*DbManager* object. security.*SessionController* and security.*UserController* objects are used for authentication and authorization.
- **servlets**.**GetTimetable**. Servlet invoked when a timetable is visualized. The schedules composing the timetable are retrieved from the database by using a

database.*DbManager* object. It is a public servlet so there is no need of authentication and authorization process.

- **servlets**.**GetPeriods**. This servlet obtains the existing periods from the database and returns them in a list of HTML <option> elements. No authentication and authorization processes are required.
- **servlets**.**GetInfo**. Servlet used to obtain the information related to an element of the database. The information is retrieved by using a database.*DbManager* object and it is presented in a HTML <table> element. As public servlet, no authorization or authentication is required.
- **servlets**.**GetEditForm**. This Servlet sends as response a HTML form to create, edit or delete an element of the database. It uses the database.*DbManager* object to obtain the current element information when it will be modified or deleted. The object security.*UserController* is used to check user authorization.
- **servlets**.**GetAvailability**. Servlet that checks the time availability of a schedule. This servlet is used before a new schedule is created: for a concrete time, it returns what elements are not already scheduled. Through the database.*DbManager* object this servlet will retrieve all the schedules stored to check the availability of the involving elements.
- **servlets**.**Delete**. This servlet deletes elements of the database using the database.*DbManager* object. The result of the operation is sent to the client as HTTP Response, and it takes control of the authenticity and the authorization of the user invoking the servlet.

4.2.4. Deployment diagram

We will finish the specification of the project through a component diagram. On this diagram we can see where the different developed software has to be located and how it works with the rest of technologies that are involved in the system.

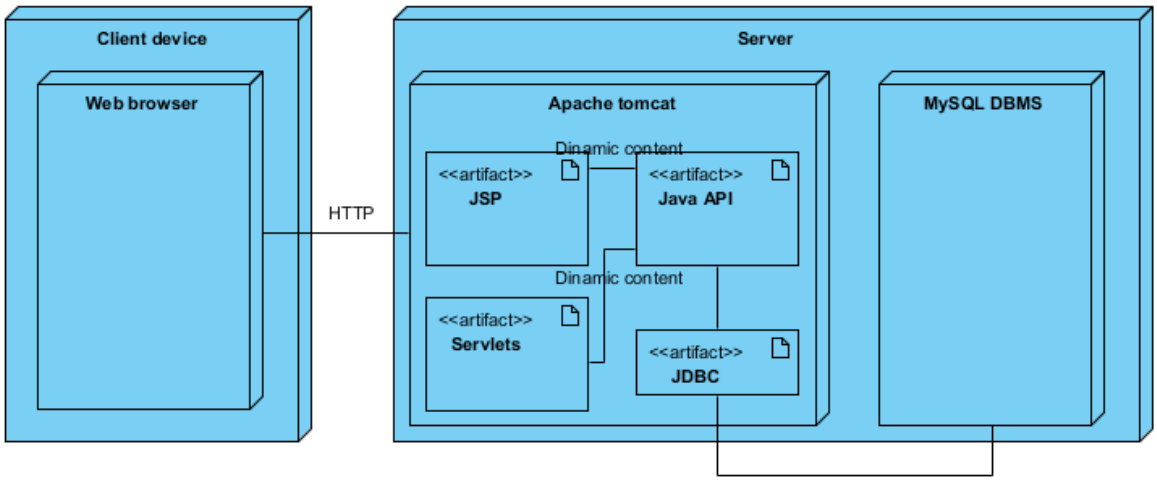

Figure 24. Deployment diagram

The elements involving the diagram are explained as follows:

- **Client device.** The device where the user is accessing to the platform from.
    - **Web browser.** The software that the user employs to connect the web server in the platform. It exchanges requests and responses through HTTP protocol.

- **Server.** The device hosting the servers. It can be the same machine for both servers (web and SQL server) or one machine per server. For this development, a single device hosts the two servers.
    - **Apache tomcat.** The Apache tomcat distribution that is hosting the JSP and the Servlets. It receives the HTTP requests, executes the JSP/Servlet requested and sends back the generated response.
        - **JSP.** The Java Servlet Pages code. They include both HTML and Java code and they use the API developed to access the database and generate the dynamic content.
        - **Servlets.** The Servlets developed. They use the Java API to generate the dynamic responses.
        - **Java API.** The API (Application Programming Interface) developed to interact with the database and perform the operations related to the application. It uses JDBC API to use the database.
        - **JDBC.** The API used to connect the DBMS hosted in the same machine or a different one

    - **MySQL DBMS.** The MySQL server hosting the DBMS that contains the data of the system.

4.2.5. Enhanced entity–relationship (EER) model

Once we have defined the necessary software for the system, we are going to focus on the data to store. The modelling of the data has to be according to the classes developed in the *database.data* package.

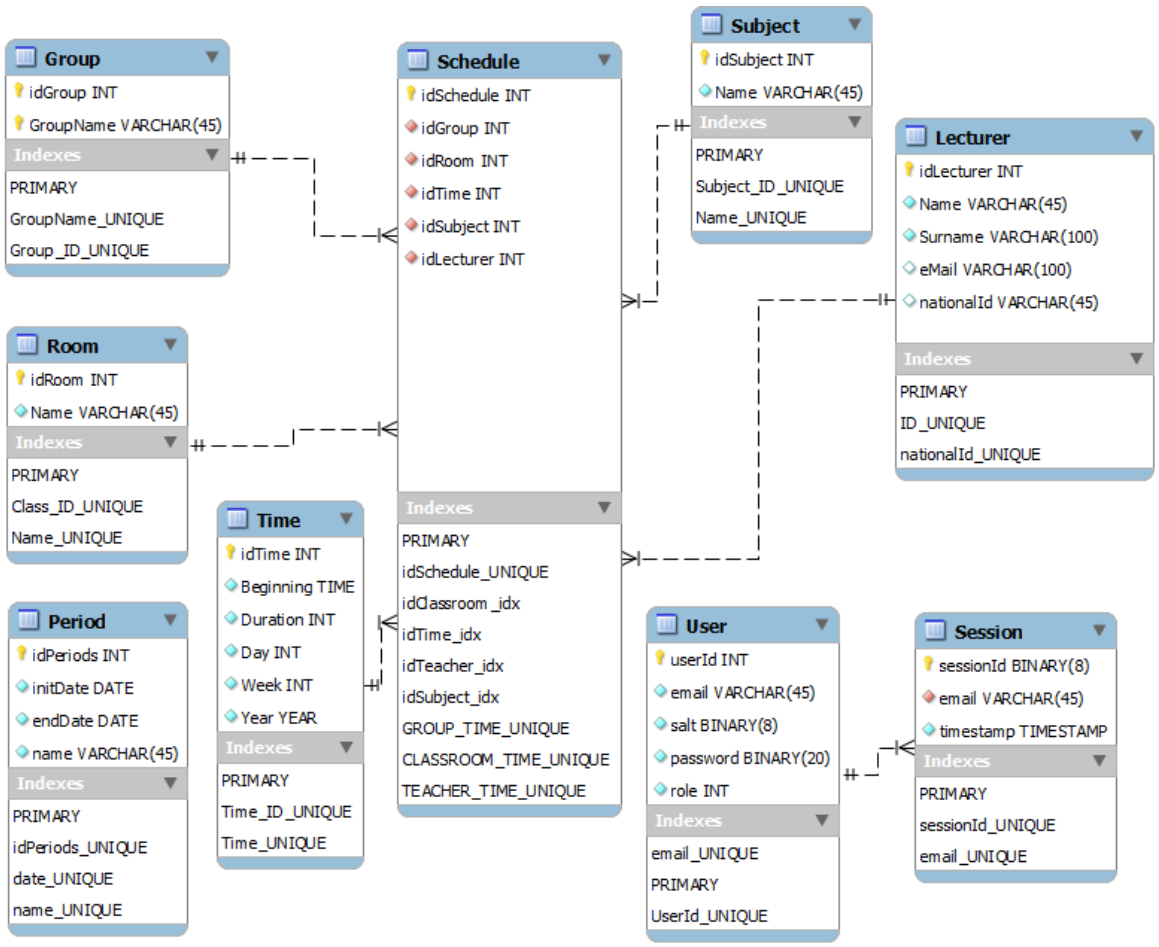

Figure 25. Database model.

This model represents the data with the following specifications:

Table Group
Columns: ▪ **idGroup:** INT.Integer that uniquely identifies the Group. ▪ **GroupName:** VARCHAR(45). String (maximum of 45 characters) containing the name of the Group.
Restrictions: ▪ idGroup: there cannot be groups with the same id number (Group_ID_UNIQUE). ▪ Name: there cannot be groups with the same name (GroupName_UNIQUE).
Primary Key: idGroup

Table 7. Specification of table Group.

Table Lecturer
Columns: ▪ **idLecturer**: INT. Integer that uniquely identifies the Lecturer. ▪ **Name**: VARCHAR(45). String (maximum of 45 characters) containing the name of the Lecturer. ▪ **Surname**: VARCHAR(100). String (maximum of 100 characters) containing the surname of the Lecturer. ▪ **eMail**: VARCHAR(100). String (maximum of 100 characters) containing the e-mail

address of the Lecturer. ▪ **nationalId**: VARCHAR(45). String (maximum of 45 characters) containing the National ID of the Lecturer.
Restrictions: ▪ idLecturer: there cannot be lecturers with the same id number (ID_UNIQUE). ▪ nationalId: there cannot be lecturers with the same national ID (nationalId_UNIQUE).
Primary Key: idLecturer

Table 8. Specification of table Lecturer

Table Subject
Columns: ▪ **idSubject**: INT. Integer that uniquely identifies the Subject. ▪ **Name**: VARCHAR(45). String (maximum of 45 characters) containing the name of the Subject**.**
Restrictions: ▪ idSubject: there cannot be subjects with the same id number (Subject_ID_UNIQUE). ▪ Name: there cannot be subjects with the same name (Name_UNIQUE).
Primary Key: idSubject

Table 9. Specification of table Subject

Table Room
Columns: ▪ **idRoom**: INT. Integer that uniquely identifies the Room. ▪ **Name**: VARCHAR(45). String (maximum of 45 characters) containing the name of the Room**.**
Restrictions: ▪ idRoom: there cannot be rooms with the same id number (Class_ID_UNIQUE). ▪ Name: there cannot be subjects with the same name (Name_UNIQUE).
Primary Key: idRoom

Table 10. Specification of table Room

Table Time
Columns: ▪ **idTime**: INT. Integer that uniquely identifies the Time. ▪ **Beginning**: TIME. String (with the format “hh:mm:ss”) containing the starting time of a schedule**.** ▪ **Duration:** INT. Integer that represents the duration (in minutes) of the time. ▪ **Day:** INT. Integer that represents the day of the week when the schedule takes place (from Monday 1 to Friday 5). ▪ **Week:** INT. Integer that represents the number of the week in the year according to ISO 8601 **(38)**. ▪ **Year:** INT. Integer that represents the year.
Restrictions: ▪ idTime: there cannot be times with the same id number (Time_ID_UNIQUE). ▪ Beginning, Duration, Day, Week, Year: there cannot be times with that happen at the same time (that have the same Beginning, Duration, Day, Week and year values). (Time_UNIQUE).
Primary Key: idTime

Table 11. Specification of table Time

Table Schedule
Columns: ▪ **idSchedule**: INT. Integer that uniquely identifies the Schedule. ▪ **idGroup**: INT. The idGroup of an existing group. ▪ **idLecturer**: INT. The idLecturer of an existing lecturer. ▪ **dRoom**: INT. The idRoom of an existing room. ▪ **idTime**: INT. The idTime of an existing time.
Restrictions: ▪ idSchedule: there cannot be schedules with the same id number (idSchedule_UNIQUE). ▪ idGroup, idTime: there cannot be more than one schedule with the same group at the same time (GROUP_TIME_UNIQUE). ▪ idRoom, idTime: there cannot be more than one schedule with the same room at the same time (CLASSROOM_TIME_UNIQUE). ▪ idLecturer, idTime: there cannot be more than one schedule with the same lecturer at the same time (TEACHER_TIME_UNIQUE).
Primary Key: idSchedule
Foreign keys: ▪ idGroup ▪ idRoom ▪ idTime ▪ idSubject ▪ idLecturer

Table 12. Specification of table Schedule

Table Period
Columns: ▪ **idPeriods**: INT. Integer that uniquely identifies the Period. ▪ **initDate**: Date. String (with the format “yyyy-mm-dd”) containing the initial date of the period. ▪ **endDate**: Date. String (with the format “yyyy-mm-dd”) containing the ending date of the period. ▪ **name**: VARCHAR(45). String (maximum of 45 characters) containing the name given to the period.
Restrictions: ▪ idPeriods: there cannot be periods with the same id number (idPeriods_UNIQUE). ▪ initDate, endDate: there cannot be periods with the same initDate and endDate (date_UNIQUE). ▪ name: there cannot be periods with the same name (name_UNIQUE).
Primary Key: idPeriods

Table 13. Specification of table Period

Table User
Columns: ▪ **userId**: INT. Integer that uniquely identifies the User. ▪ **email**: VARCHAR(45). String (maximum of 45 characters) containing the email of the user. ▪ **salt**: BINARY(8). Byte Array (of 8 bytes) with the random salt generated for the user (view Implementation issues for more information). ▪ **password**: BINARY(20). Byte Array (of 20 bytes) with the password of the user (view

Implementation issues for more information).
Restrictions: ▪ userId: there cannot be users with the same id number (UserId_UNIQUE). ▪ email: there cannot be users with the same email (email_UNIQUE).
Primary Key: userId

Table 14. Specification of table User

Table Session
Columns: ▪ **sessionId:** BINARY(8). Byte Array (of 8 bytes) with random bytes that uniquely identifies the session (view Implementation issues for more information). ▪ **email**: VARCHAR(45). String (maximum of 45 characters) containing the email of the user that started the session. ▪ **timestamp**: TIMESTAMP. String (with the format "yyyy-mm-dd hh:mm:ss") containing the time of last renovation of the session (view Implementation issues for more information).
Restrictions: ▪ sessionId: there cannot be sessions with the same id byte array (sessionId_UNIQUE). ▪ email: there cannot be sessions with the same email (email_UNIQUE).
Primary Key: sessionId
Foreign Key: email

Table 15. Specification of table Session

To see the SQL statements that will create this model, it is possible to consult the Annex III. SQL statements for MySQL initialization.

4.3. Implementation issues

With the model explained in the previous section, and the Javadoc attached in the **¡Error! No se encuentra el origen de la referencia.**, it is expected to have gathered enough information to easily understand how the application works, as well as the code developed. Nevertheless, some aspects have not been covered at all yet, so this section will treat these issues related to the implementation of the design above.

4.3.1. Security issues

This project, as mentioned along the document, has been born to serve as a prototype for a future implementation and with the intention of be grown and modify. For that reason, the security involving the current project is not a priority because it depends on many factors. Many of these factors are related to the devices where the software will run (for example, a firewall to avoid unappropriated connections), but other factors depend directly on the platform developed.

For the factors related to the platform, we are going to explain the solutions taken and some guidance on how to improve this solution in case that the system will be exposed to the public.

Security on Authentication

First step on security for a project that requires the authentication of a user is to find a secure authentication method. Nowadays we can find a huge range of services that are completely focused on authentication: for companies, Universities or Institutions is common to find an intern user authentication service as LDAP or Kerberos for this goal; for a web application it is easy to find OAuth (39) authentication services from big companies as Google or Facebook.

The authentication from external reliable software is recommendable for two reasons:

1. You don't have to store user's credentials, which is sensitive information that cannot be compromised if there is an attack on the database.
2. The authentication algorithms developed by a specialized software will probably be more secure that the one being implemented by yourself.

Additionally, using a third party authentication software will allow update this services without making substantial changes on the application.

For all of this a good practice would be the use of an external tool for authentication. However, for the approach being developed here we are going to use our own authentication system. This method will be based on user-password credentials, but **the password has not to be stored as plain text**. If we do this, we are compromising one of the most sensitive informations of a person. The user will probably use the same password for our system that had previously used in other systems, so if this information is compromised from our database during an attack, we are giving the key (or powerful clues) for many other systems. Here is where salt and encryption algorithm are needed.

The **salt** is a randomly generated sequence of bits that is unique to each user and is added to the user's password as part of the encryption process. This prevents rainbow table attacks (40). In addition, with the salt, two passwords that are equal will have different encryption.

Once we have the salt, a **one-way encryption** has to be made in order not to store the password as text plan. This same algorithm will be used when a user is doing the log in process, and the result of the algorithm is what has to be compared. It is important not to choose a broken algorithm; otherwise this measure would not have much sense. The algorithm chosen for our approach will be PBKDF2. This algorithm, apart from being recommended by the NIST (41) (Section 5.3 of Special Publication 800-132), contains adjustable key stretching to defeat brute force attacks. The basic idea of key stretching is applying the hash algorithm to the result of the hash as many times as it is desired. This greatly increases the time it takes to try to hack the hash the billions of possible passwords. The more times it is applied, the more secure it is, but more processing time is required.

The operations of encryption are hold in the class *security.Security*. The process for password generation and authentication are summarized in the following sequence diagrams:

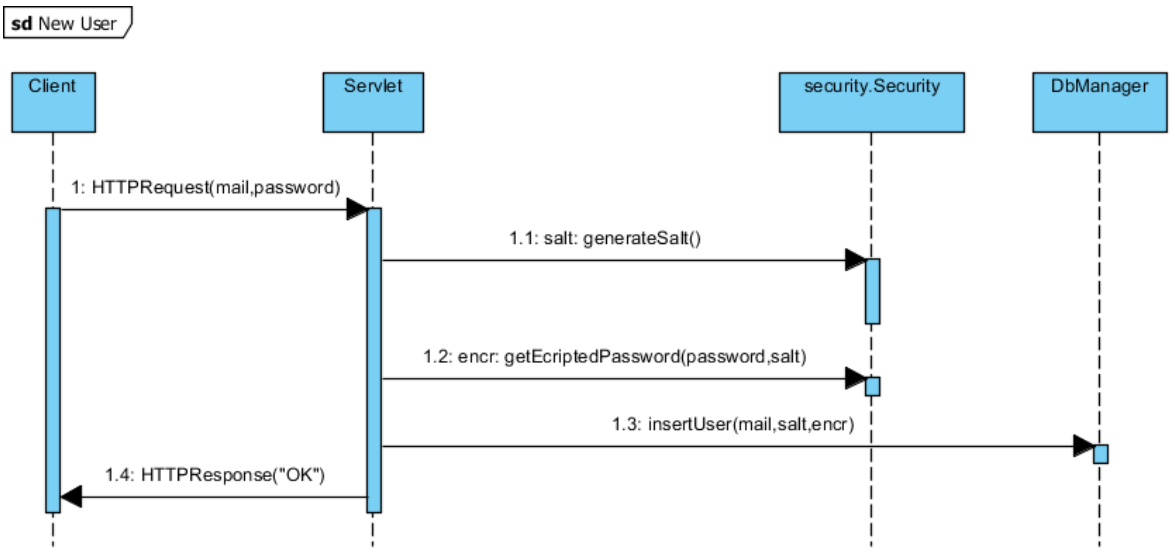

Figure 26. New password generation.

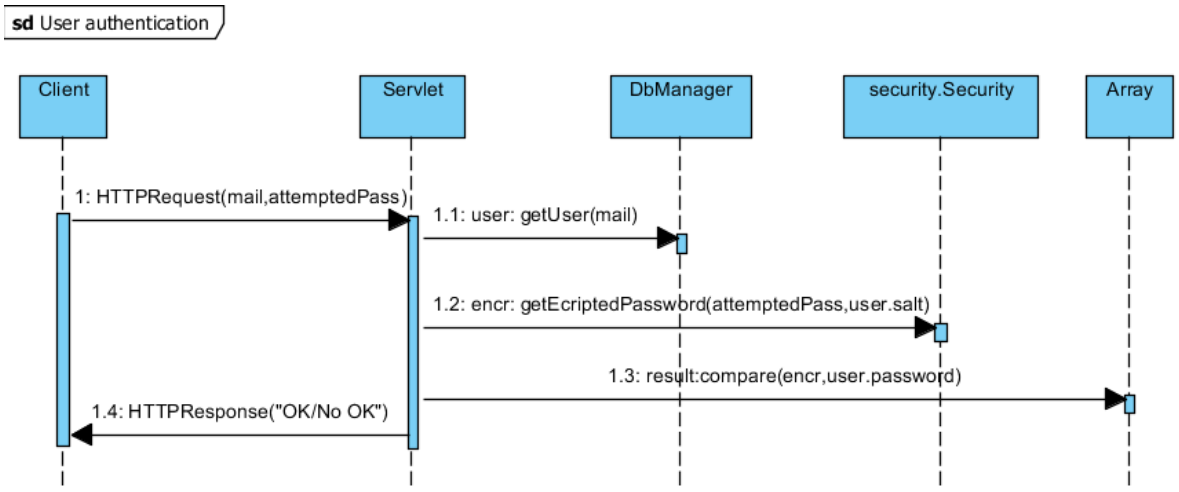

Figure 27. User authentication.

Security on session

During the usage of the platform, a user must login only once. After this first authentication, it will access to the servlets that need authentication through a session ID. This session ID is sent to the client after authentication has succeeded, and it is automatically sent back to the server every time the client sends an HTTPRequest.

As this session ID is modifiable by the user, the session control must not to be predictable. If the session ID is predictable, an attacker can predict future sessions after achieving a previously generated session ID. The algorithm that generates the salt can be used to generate session IDs.

In addition, the session needs to have a timestamp. We must think of the session as the "temporary password" of a user: after authentication, only session ID will be checked. This session should not be valid forever, as it will be stored in the user browser and another person could use the same browser. The timestamp will determine when a session should be invalidated. In our

implementation, the timestamp will be consulted every time that session control is needed. If the session is still active, the timestamp will be updated. If the session is not active, it will be deleted and the user won't be able to make use of the servlet.

Apart from session control, multisession for the same user won't be allowed. A session will be uniquely attached to a user, so if there is a new log in and the session was already started, the first session will be replaced by the new one. For this implementation, the session will always be checked with the user of that session.

The session control implemented on this project is summarized in the following sequence diagram:

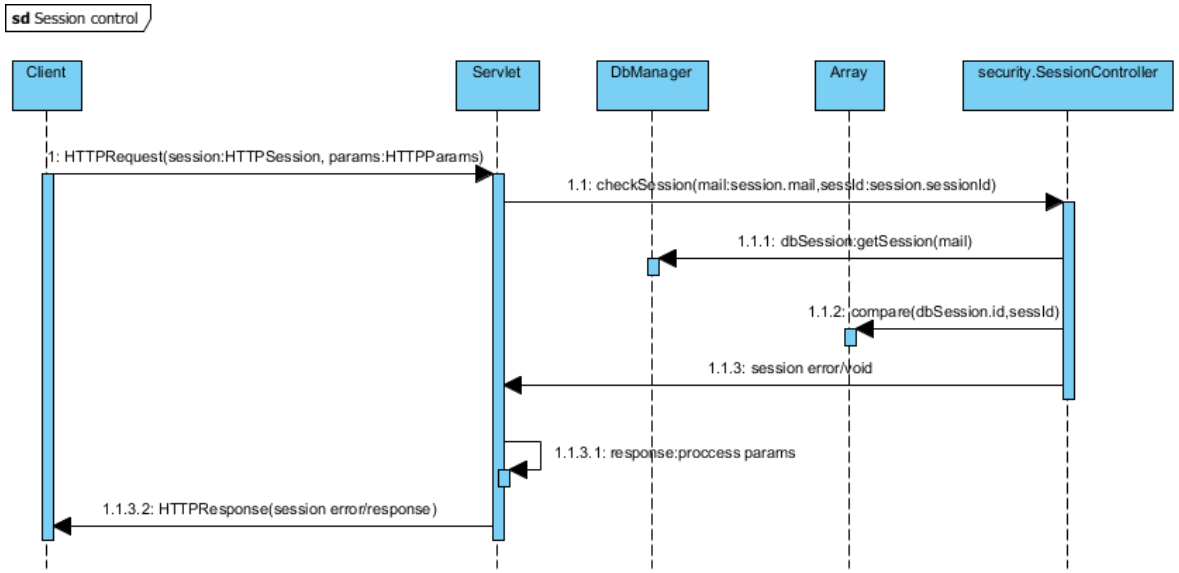

Figure 28. Session control

Security on user control

Apart from authentication and user control, he developed tool makes use of different roles for the users. Each role enables certain tools for the user, so in the platform it is needed to have a control on the roles of these users.

User control on this project is aimed to be used after session control: that way, we make sure that the mail attached to the session is valid. With this mail, we simply check through the database which is the role assigned to the user and adequate the content of the page to the user.

Security on communications

When a client sends information to a server, the HTTP protocol mainly uses two different methods: POST and GET. After a brief comparison that is detailed on (42), we can see that both methods send parameters (including sensitive information as passwords) in plain-text. That is, anyone with a packet sniffer could see what parameters we are exchanging.

The solution to this problem is use a secure layer over HTTP protocol: Transport Layer Security (TLS) (43) and its predecessor, Secure Sockets Layer (SSL) (44). They are cryptographic protocols which are designed to provide communication security over the Internet. The use of these protocols with HTTP is commonly known as HTTPS, and both client and server must implement this protocol to be able to establish a secure connection.

In the Apache Tomcat distribution used in this project, the HTTPS protocol has not been installed due to not needed secure connections (to include HTTPS on Tomcat, consult (45)), but the final server will need to implement this protocol to strengthen the security on the platform.

Security on database

Our last look to the security is focused on the database. MySQL uses security based on Access Control Lists (ACLs) for all connections, queries, and other operations that users can attempt to perform. There is also support for SSL-encrypted connections between MySQL clients and servers.

Security guidance can be found on MySQL website (46) with tips on what is necessary to protect and with MySQL configuration details. In our implementation we have worked with standard configuration - which is totally avoidable for public use, a minimum of security configuration has to be done.

4.3.2. Client side

On the previous section we have seen how the server works to process the request to servlets and JSP pages. However, a web application is entirely dedicated to the client, and the presentation of the tools that the client (user) will use is very important. We have right now the tools to reach the purpose of the project, but in this section we will see what technologies are going to be used to present the user an optimal presentation of these tools.

HTML

The HTML is the language that a web browser will interpret. It is the skeleton of the web page that is being showed to the user, and the main element of a HTTP Response.

The HTML code that our server is going to generate depends in first place on the user: it will be different for public user, schedule manager and administrator. The main purpose of the servlets generated is the achievement of this dynamic HTML. JSP pages are the elements that will serve the three types of views to a client browser; they will make use of the Java API developed to adapt the content of the HTML to the different users.

In our project, three different JSP pages have been created:

- **index.jsp**. This page is designed to serve to public users. It will offer the tools for the visualization of the timetables that have been previously created. On this page, it is possible to navigate through the different timetables organized by Group, Lecturers or Rooms. It also includes a quick-login tool to let users to enter the platform.

Figure 29. Example of index.jsp

- **login.jsp**: This page will allow the login to the users so that they can enter to the admin.jsp page. It will also show login errors.

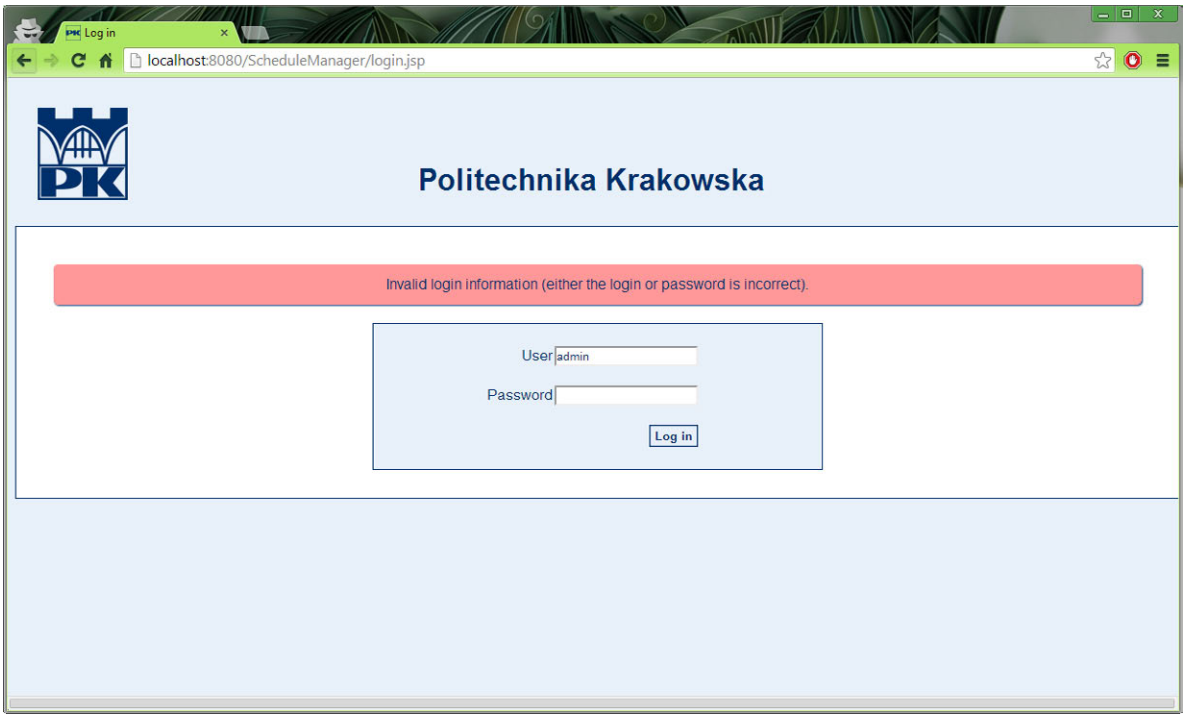

Figure 30. Example of login.jsp with wrong credentials.

- **admin.jsp**. The admin.jsp will firstly perform the authentication of the user and create the session in case of success. Then, depending on the role of the user, will show or

tools for schedule manager or tools for administrator. If authentication fails, it will redirect to login.jsp.

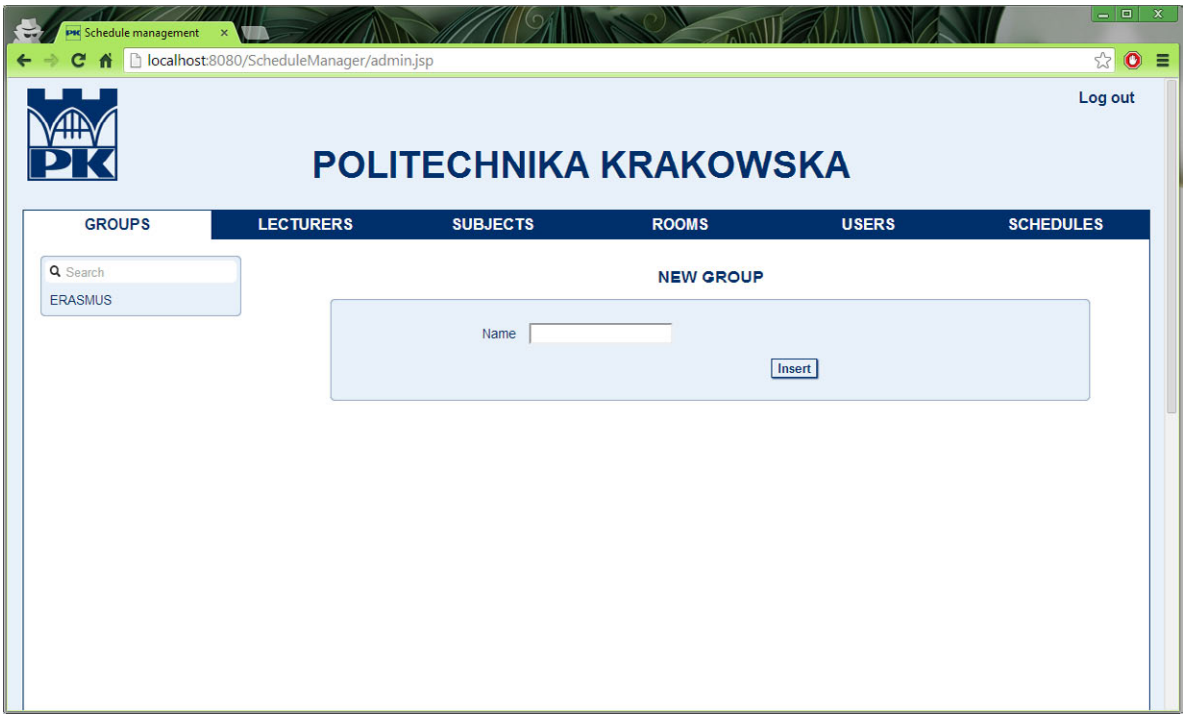

Figure 31. Example of admin.jsp with Administrator role.

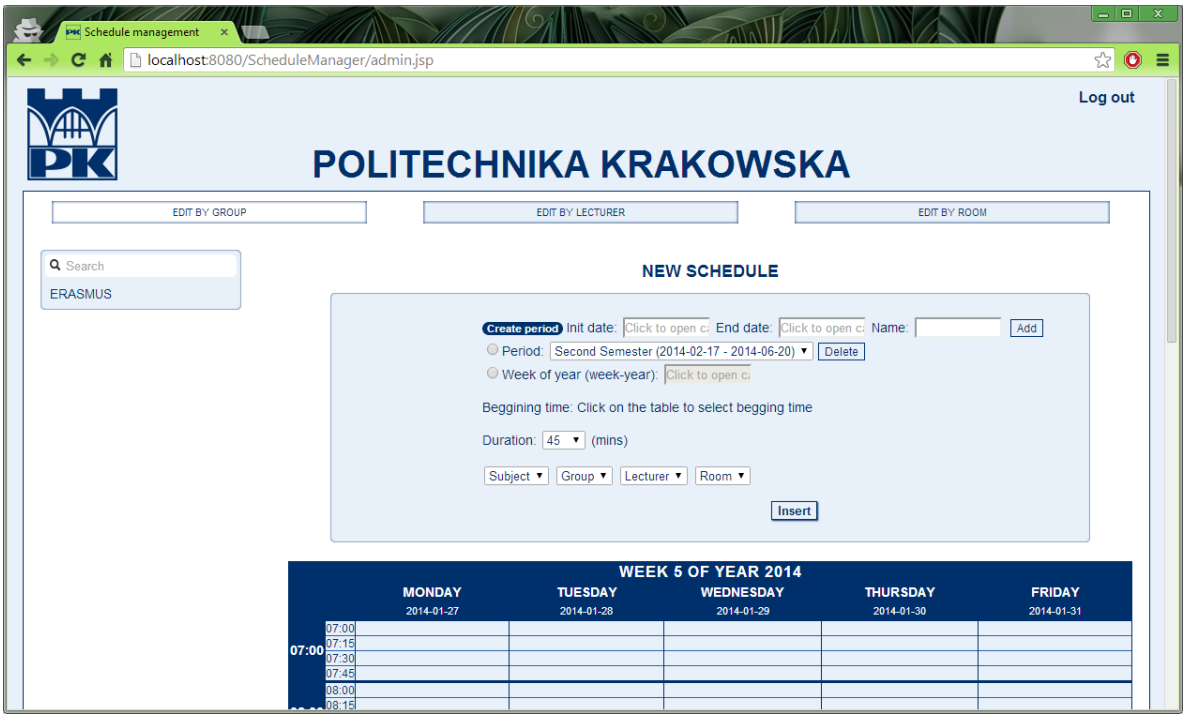

Figure 32. Example of admin.jsp with Schedule Manager role.

CSS

Cascading Style Sheets are used to set the presentation of the HTML models. Shapes, colors, positions… If HTML defines what elements are shown in the webpage, CSS defines how these elements are represented.

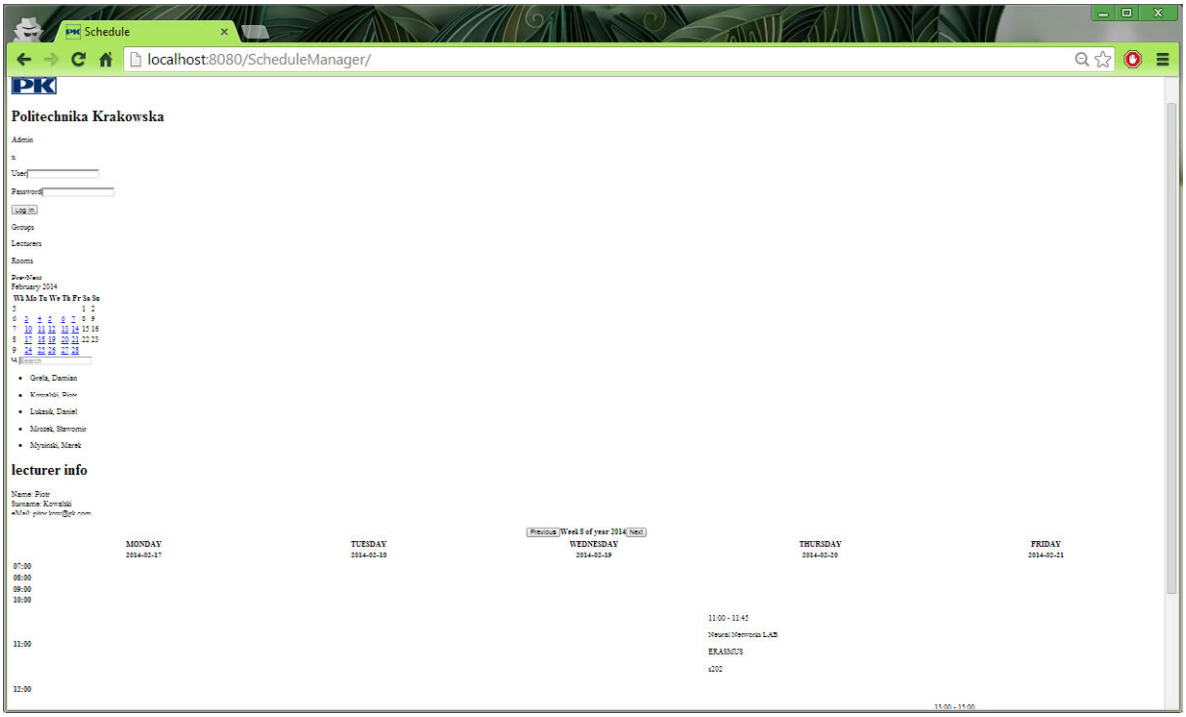

Figure 33. Example of index.jsp without index.css

Figure 34. Same example as in Figure before with index.css.

For the project, three CSS have been defined, one per JSP: **index.css**, **login.css** and **admin.css**. Blue tones have been chosen according to the logotype of Cracow University of Technology.

JavaScript

If HTML indicates which elements are on the page and CSS indicates how to place and view these elements, JavaScript allows making these elements dynamic. This will achieve, for example, the navigation among different options of the menu.

JavaScript is a scripting language that allows introducing code in the webpage. As in most scripting languages, it associates types with values, not with variables. It is almost entirely object-based, with objects as associative arrays augmented with prototypes. JavaScript typically relies on a run-time environment as web browsers.

In this project, all JavaScript functions have been written in the same page, **functions.js**.

jQuery, AJAX and jQuery UI

HTML, CSS and JavaScript can show a series of static elements with a certain grade of dynamism, but we have developed Servlet to get real information from our server without creating many JSPs. How is it possible to retrieve this new information from the existing page? jQuery and AJAX will give the answer.

jQuery (47) is a fast, small and feature-rich JavaScript library. It eases document traversal and manipulation, event handling, animation with an API that works across a multitude of browsers. It also provides easy access to AJAX techniques.

AJAX, an acronym for Asynchronous JavaScript and XML, is a group of interrelated web development techniques used on the client-side to create asynchronous web applications. With Ajax, web applications can send data to, and retrieve data from, a server asynchronously (in the background) without interfering with the display and behavior of the existing page.

Then, with jQuery we can easily manage AJAX, and with AJAX we can invoke the servlets without changing the whole existing page. The answer received from the servlet will easily be located in the webpage using jQuery functions.

jQuery UI (48) is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. It will be used in our project to show the calendars and the messages sent to the user.

jQuery and jQuery UI are imported as external JavaScript pages in the same way our own JavaScript page is imported to the HTML. For this project, the versions used are jQuery v2.0.3 (49) and JQuery UI 1.10.3 (50).

Server properties

As the project depends on a database and this database can have different URL, user and password, a .properties file will contain the different data related to the database, so the server can load it dynamically with no need of recompiling the whole project.

The Java class in charge of loading the elements of this file is called *DatabaseElements.java* and it is contained in the package *database*.

It will use the class *java.util.Properties* to read de file, and the format of the file will be a list of "property.name=property.value". **"Annex II. Example of *config.properties* file"** section gives an example of the conifg.properties file used in this project. This file must be located in the *ScheduleManager/build/web/WEB-INF/* directory. This is the directory containing the loaded classes and libraries used in the project.

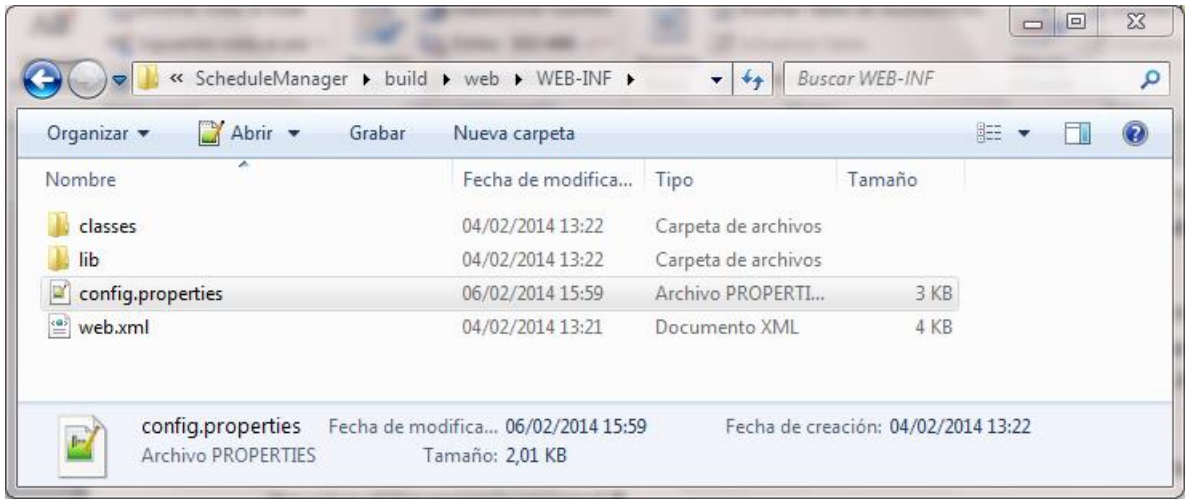

Figure 35. Directory of config.properties file

This *DatabaseElements* class will contain a private static attribute containing the properties read. It will avoid that the file is read every time a property is needed. This will help not to waste too much time reading a file that will probably not change in months. If this file changes, the whole class needs to be reloaded (the project must be rebooted).

Once instanced an element of the class DatabaseElements, the method getElement will return the value of the property required.

4.4. Performance of developed application

Now that we have explained the development of the application, it is time to check if it covers the proposed goals. We are going to see the Use Cases that we set as goals and use them in the platform to see the performance reached.

4.4.1. Use case "Insert, update or delete timetable elements (lecturers, subjects, rooms and groups)"

In this use case, the user, as an administrator, can insert, update or delete elements from the database. The necessary steps to achieve this use case are summarized in the following steps.

1. **Log in as an administrator**. The first user with administrator role registered in the database is the user "admin" with the password "admin".

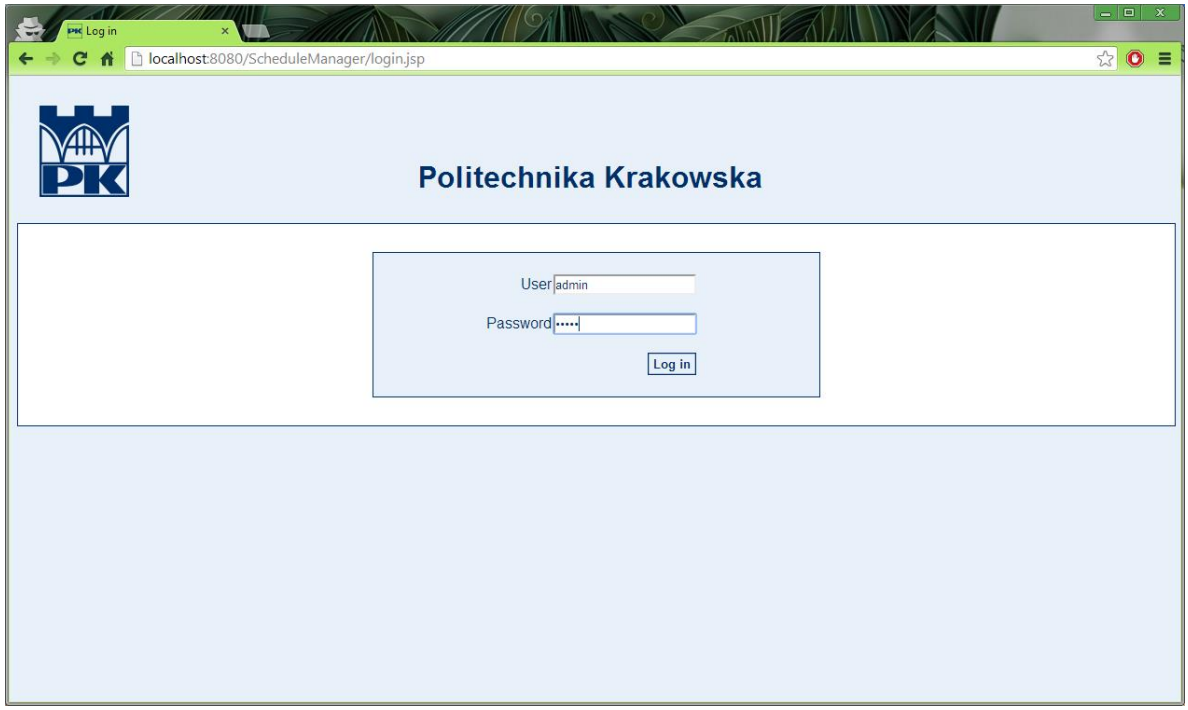

Figure 36. Log in as an adiministrator.

2. **Select a menu to insert a new element**.

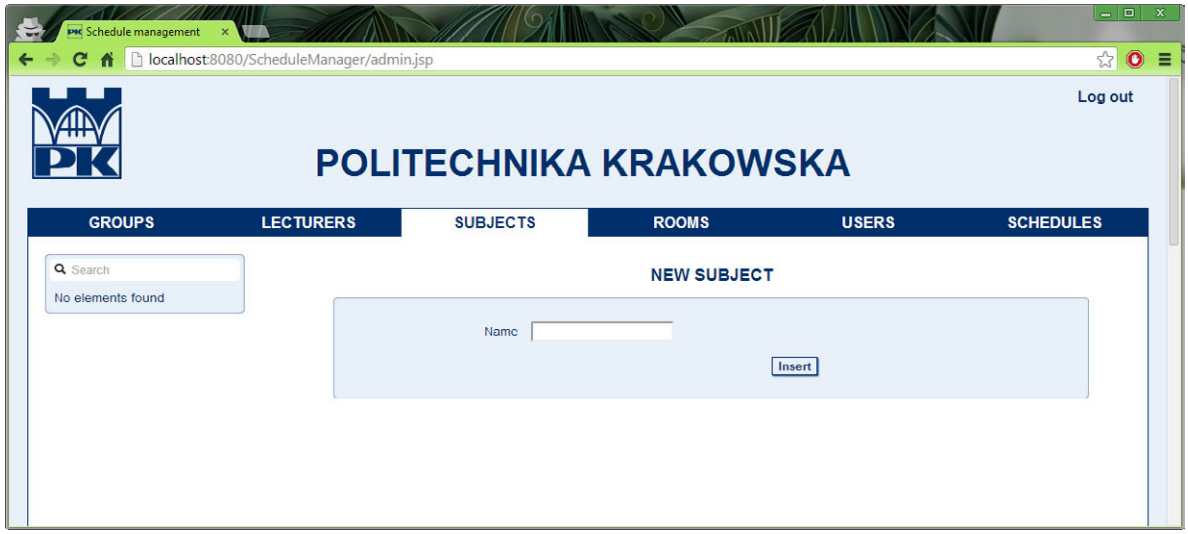

Figure 37. Menu subjects selected.

3. **Insert a new element in the database**.

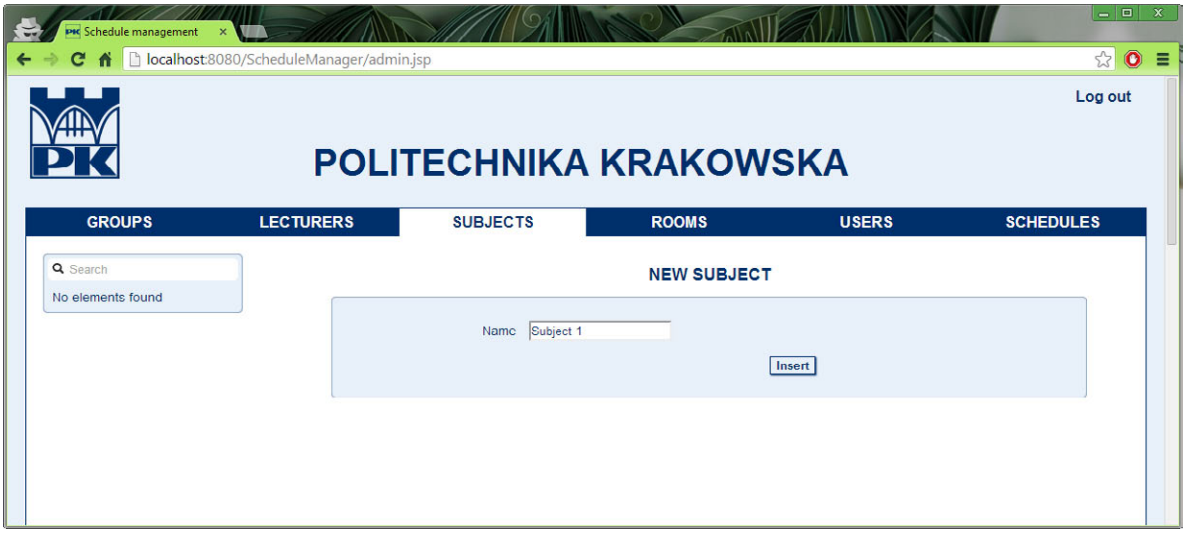

Figure 38. Inserting a subject.

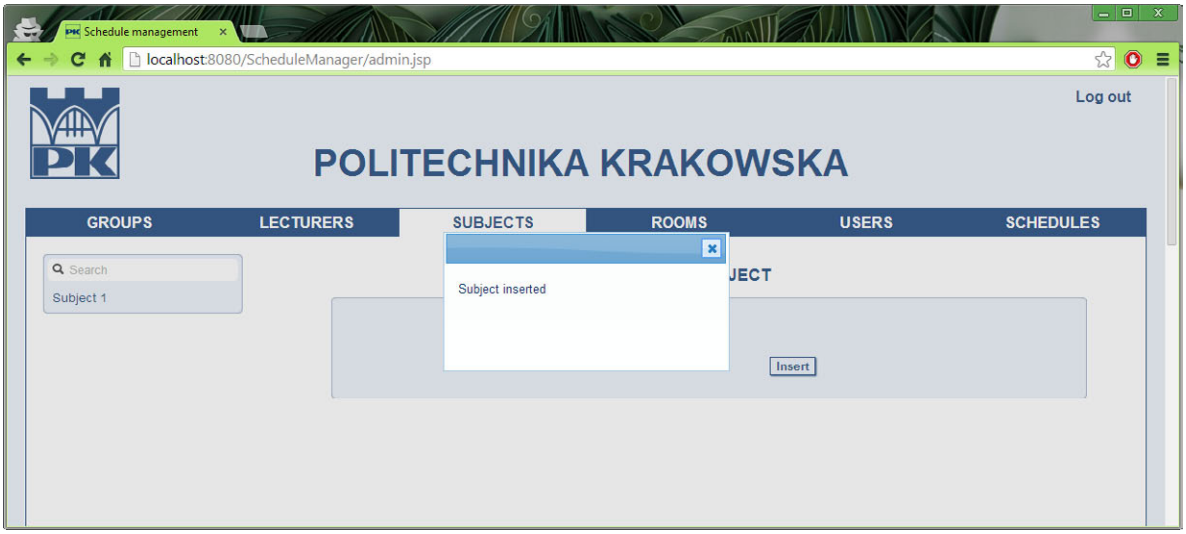

Figure 39. Subject successfully inserted.

4. **Edit the created element**.

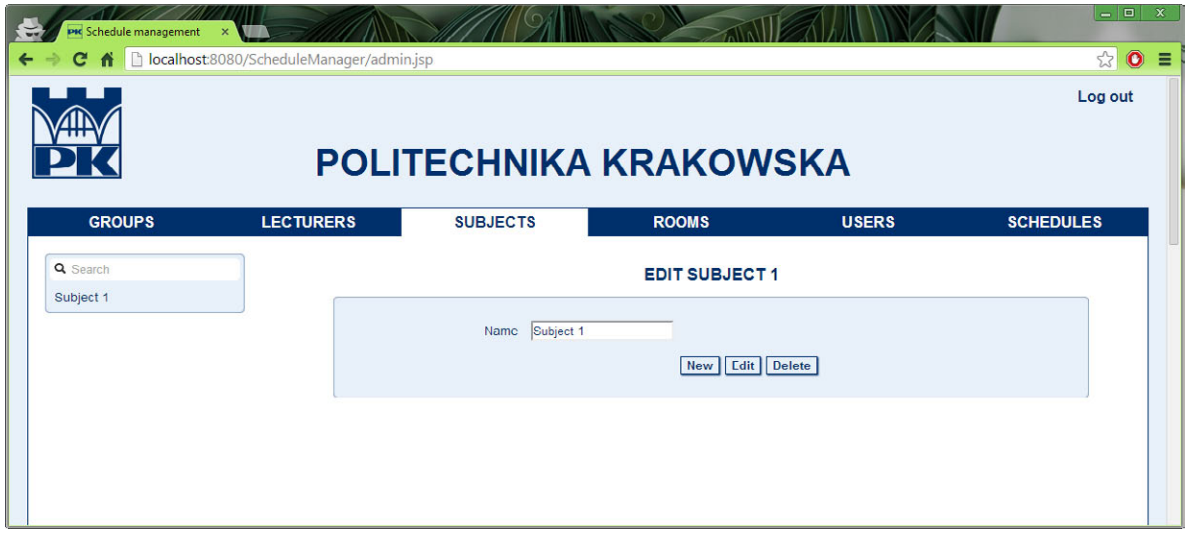

Figure 40. Selected subject to be edited or removed.

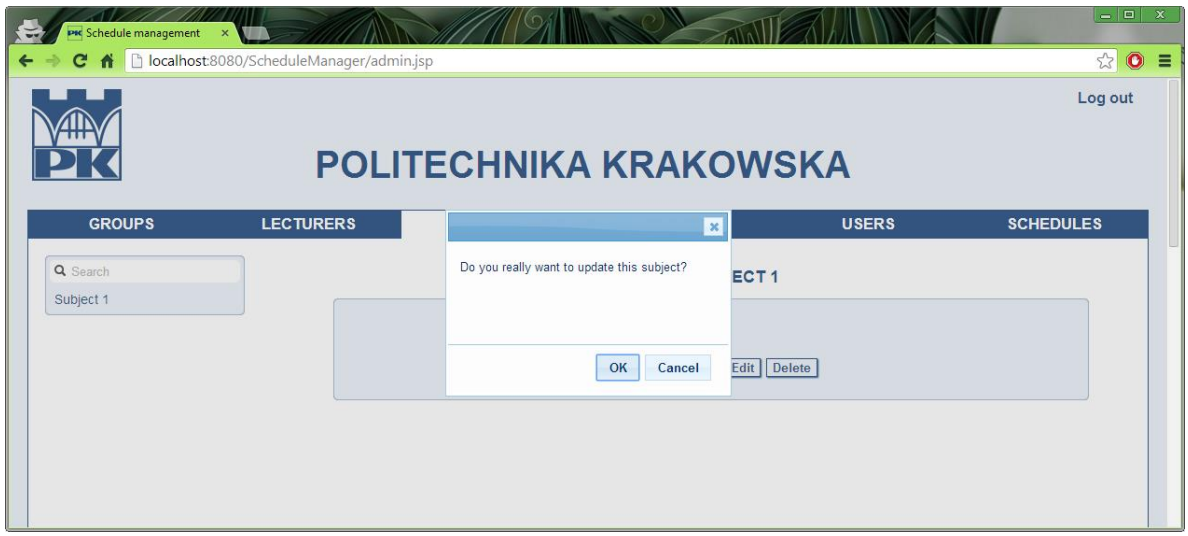

Figure 41. Confirmation message to edit the subject.

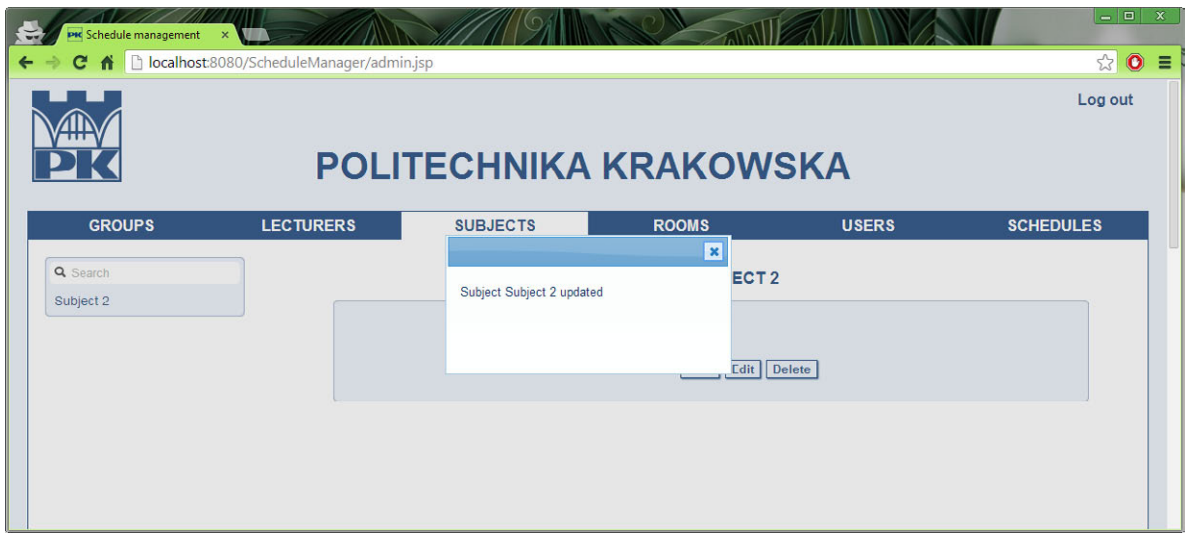

Figure 42. Message showing the result of the operation after inserting subject.

5. **Delete the created element**.

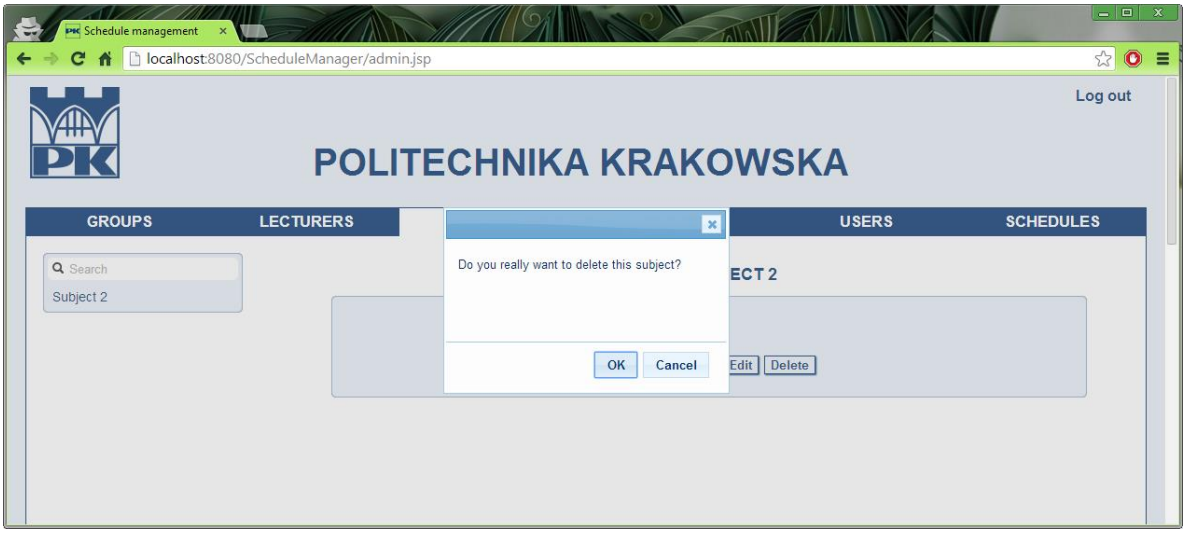

Figure 43. Confirmation message to delete the element.

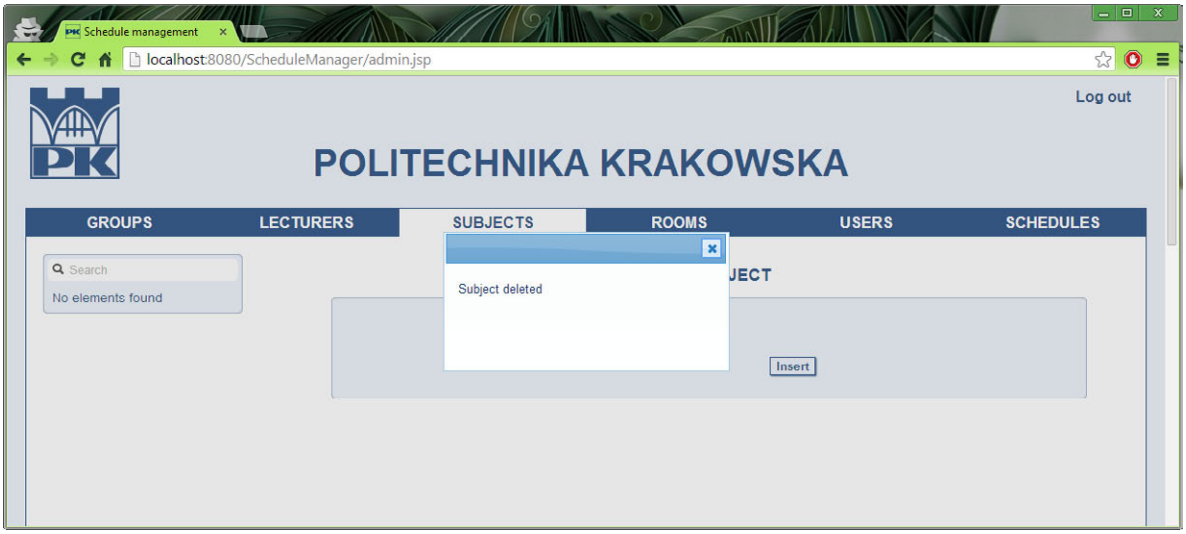

Figure 44. Message with the result of the operation.

This process is identical for groups, lecturers and rooms and same results are obtained.

4.4.2. Use case "Create or Delete users of the system"

In this use case the user, as administrator, can manage the different users and roles that can use the system. The steps to perform this use case are the following ones:

1. **Select USERS menu**. Once the administrator has logged in as administrator – as shown in **Use case "Insert, update or delete timetable elements (lecturers, subjects, rooms and groups)"** - he can access the USERS menu to manage the users.

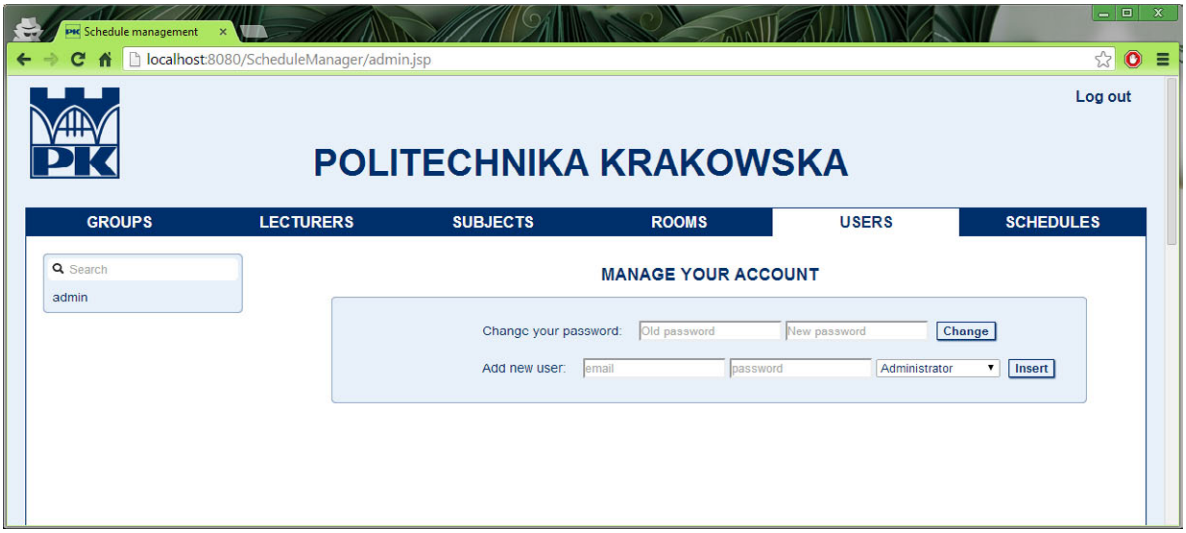

Figure 45. Users menu selected.

2. **Create a new user with the role Administrator.**

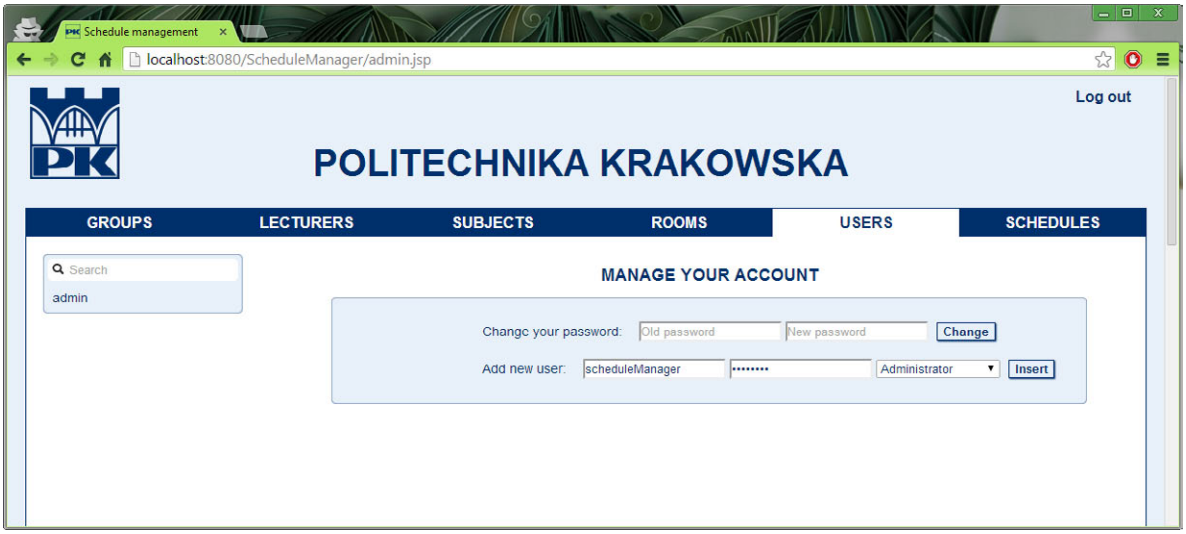

Figure 46. New user data inserted in the inputs.

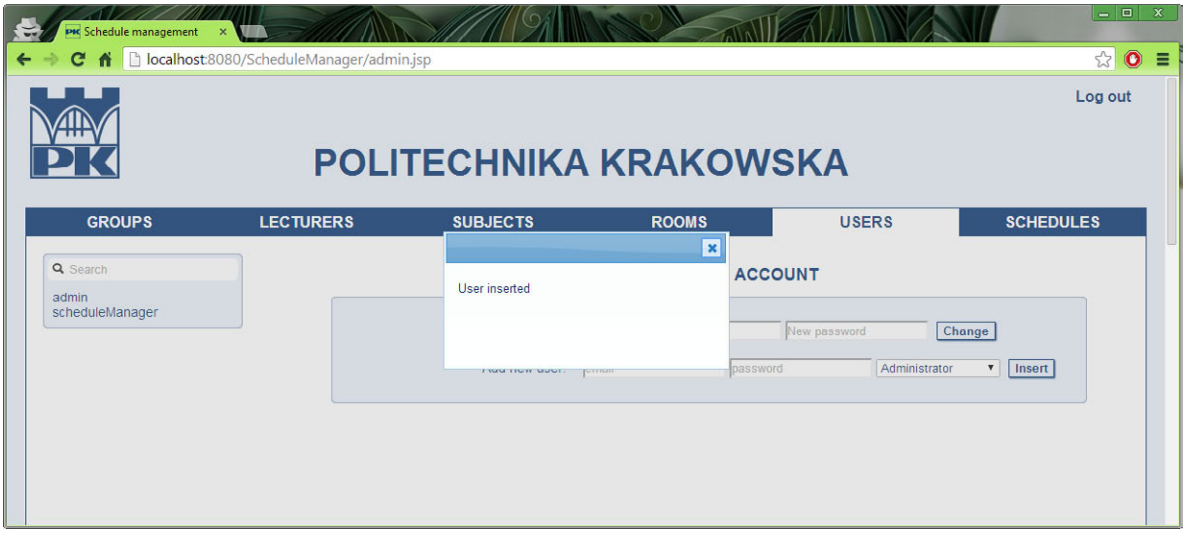

Figure 47. Result of the operation after clicking Insert button.

3. **Change the role of the user.**

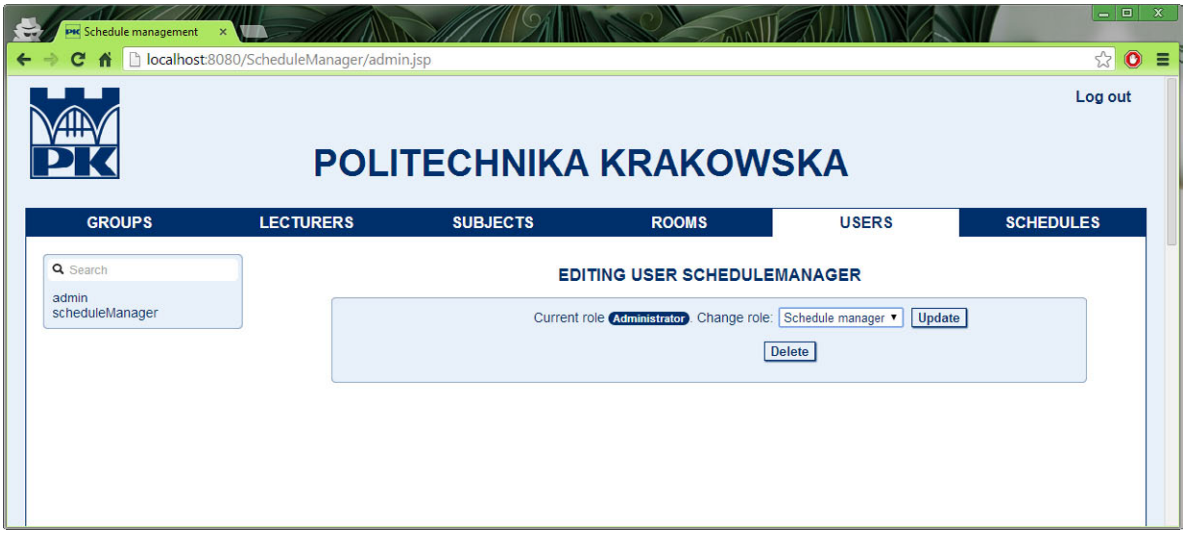

Figure 48. New role for the user selected.

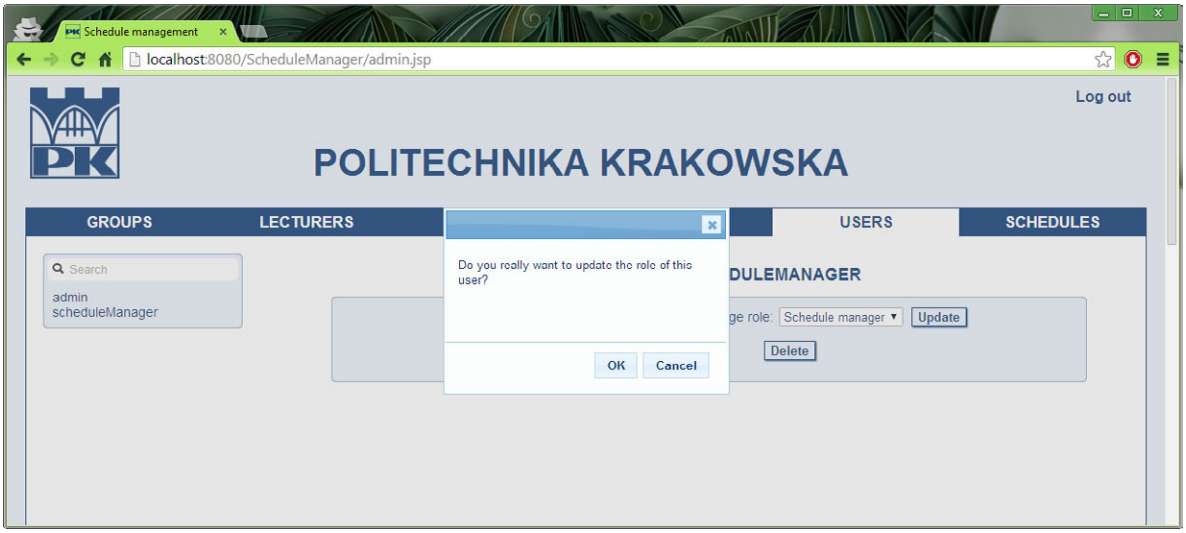

Figure 49. Confirmation message to change the role of the user.

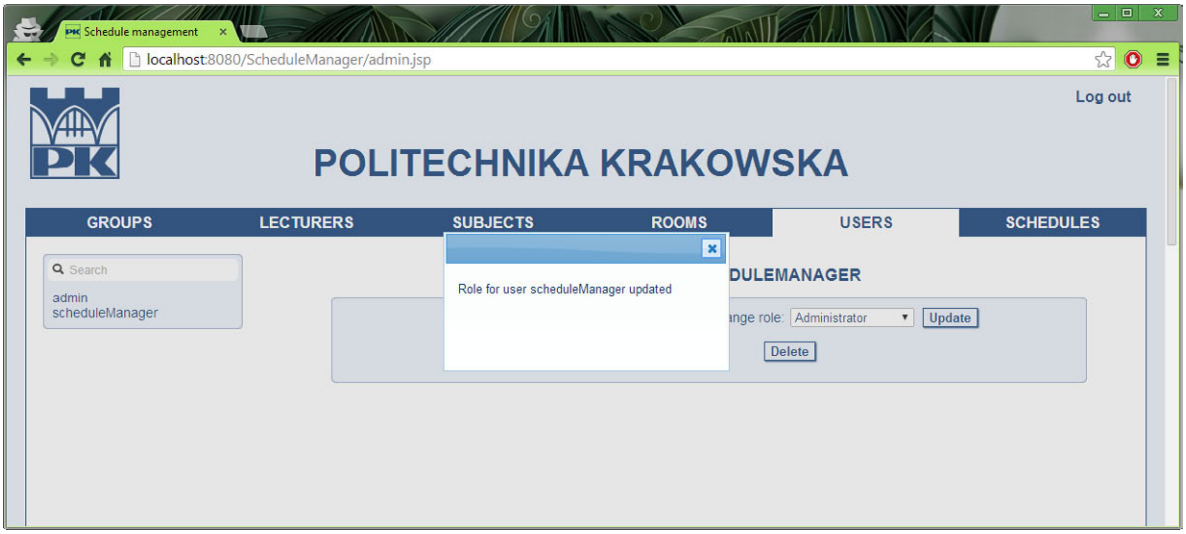

Figure 50. Message with the result of the operation aftoer clicking Update button.

4. **Log in with the new user credentials.**

Figure 51. New login with new user credentials.

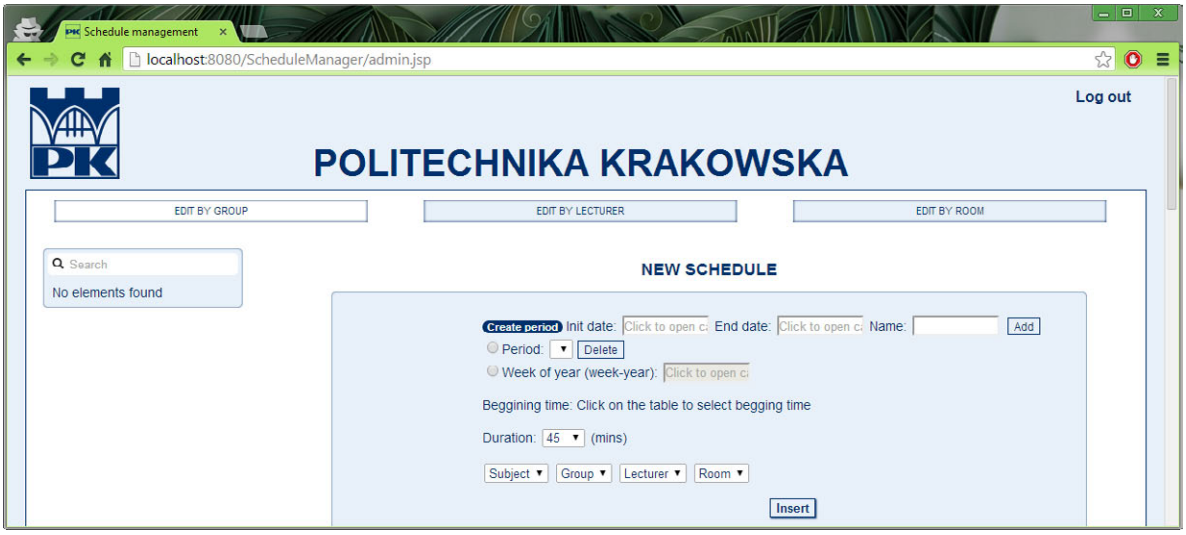

Figure 52. New view adapted to the new role of the user.

5. **Delete the user created.** If we log in again as user, we can delete existing users from the database.

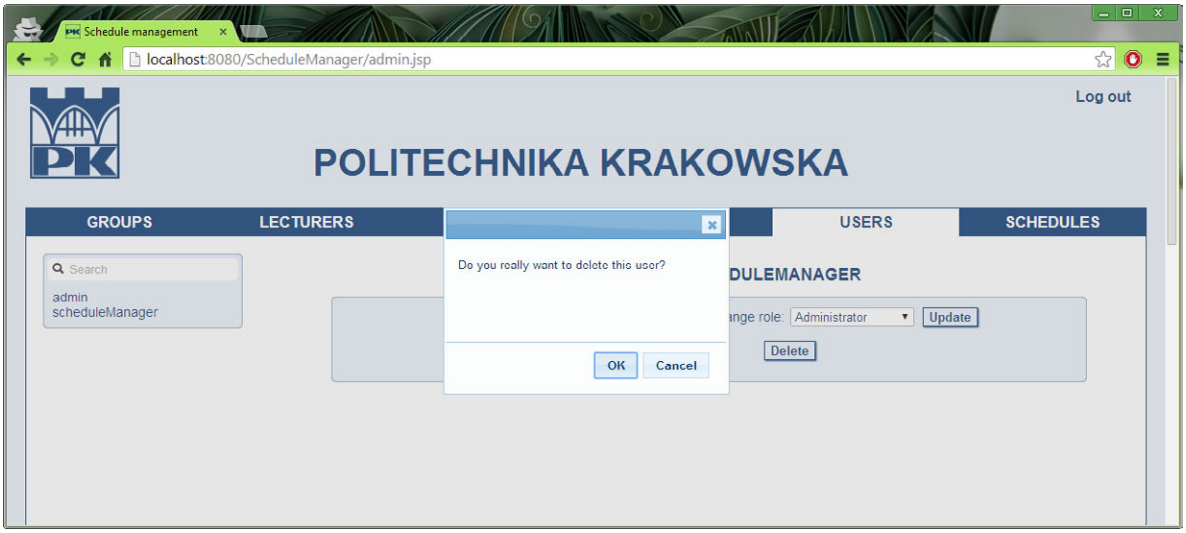

Figure 53. Confirmation message for deleting the user.

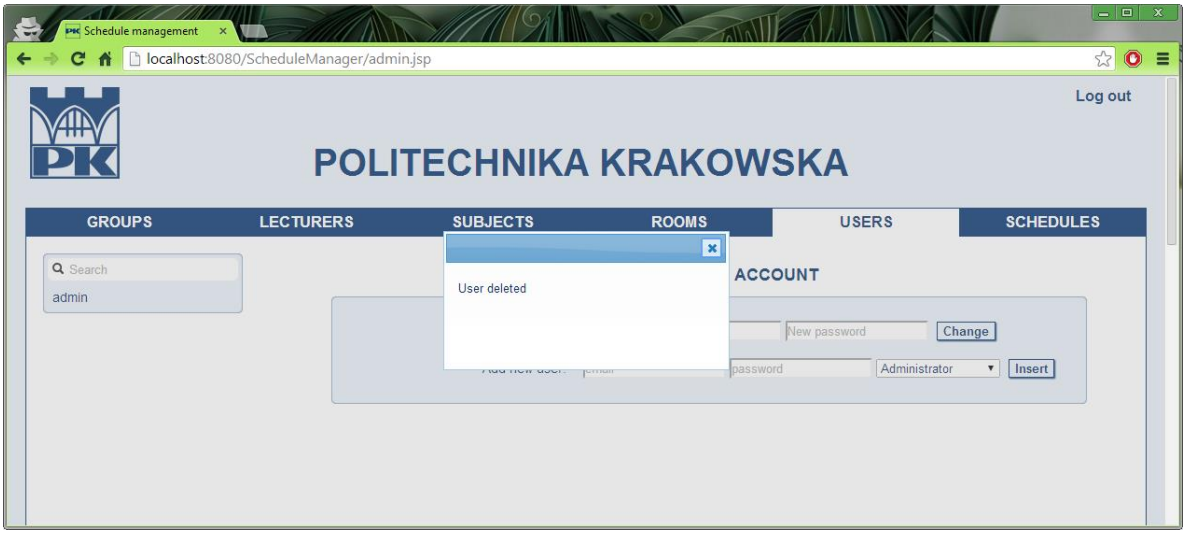

Figure 54. Result of the delete operation after clicking Delete button.

4.4.3. Use case "Modify credentials"

In this use case, an administrator can modify his credentials: that is, modify his password. This option is available also in the USERS menu:

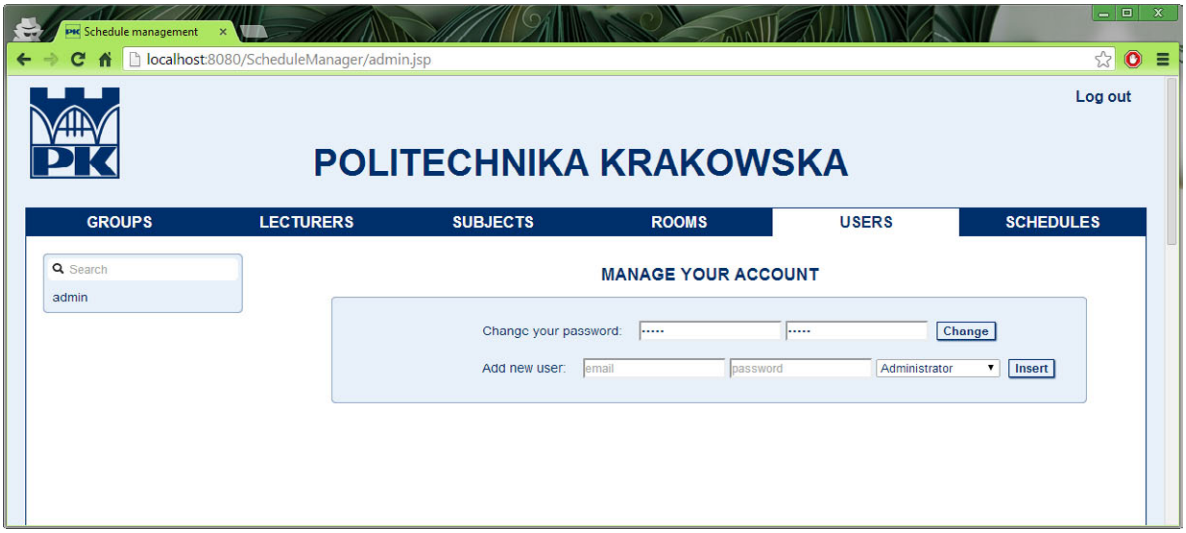

Figure 55. New credentials inserted in the inputs.

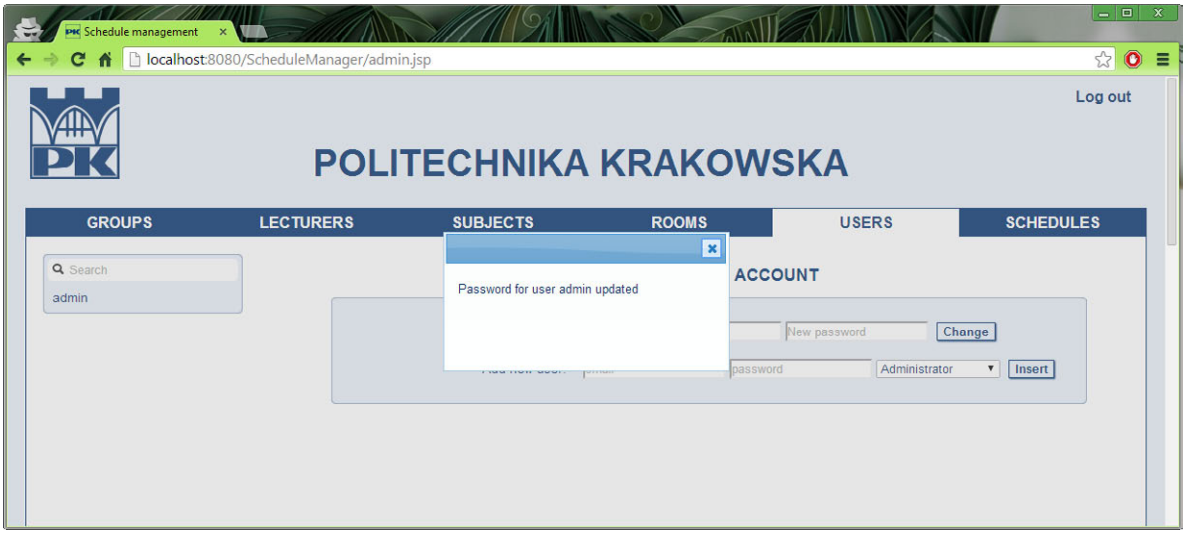

Figure 56. Message with the result of the operation is shown after clicking Change button.

4.4.4. Use case "Create or modify a timetable"

In this use case, the user, both as administrator or schedule manager, will be able to create, modify or delete timetables. A timetable is created through schedules, and the schedules can be created or modified in the following way:

1. **Create a new schedule in a concrete week**. If only is necessary to create a schedule for a day (for example, in order to book a room for a day), It is possible to save the schedule for a concrete date.

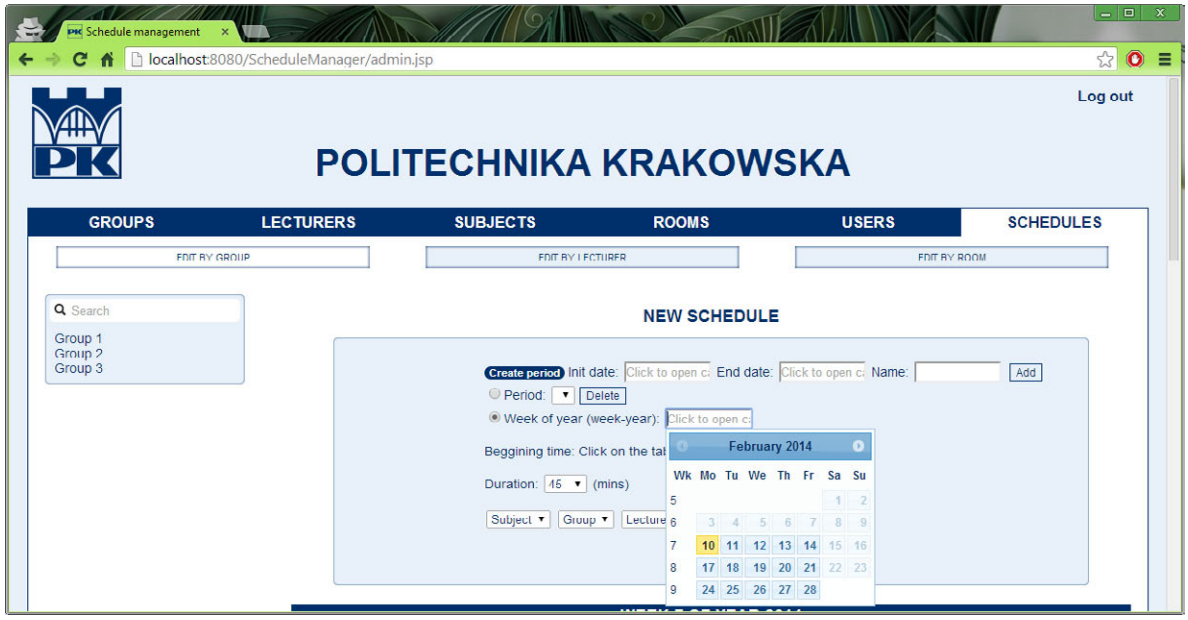

Figure 57. The week option is selected and a day is picked up from the calendar.

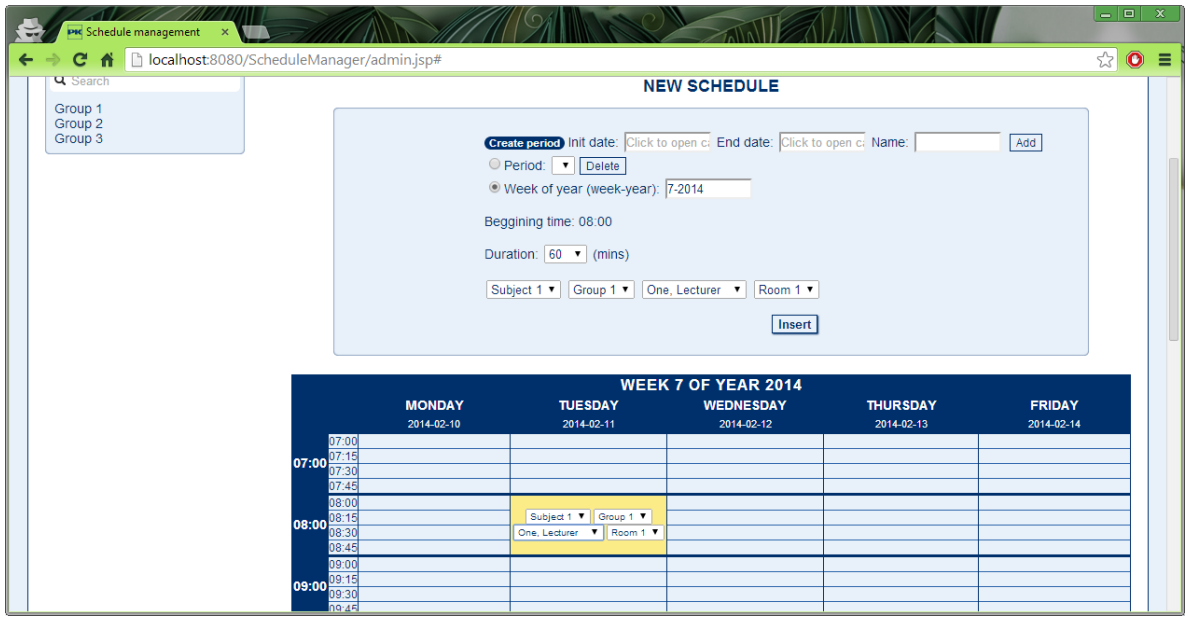

Figure 58. The duration is set to 60 minutes and a time is clicked (8.00 AM). The available elements at that time are shown.

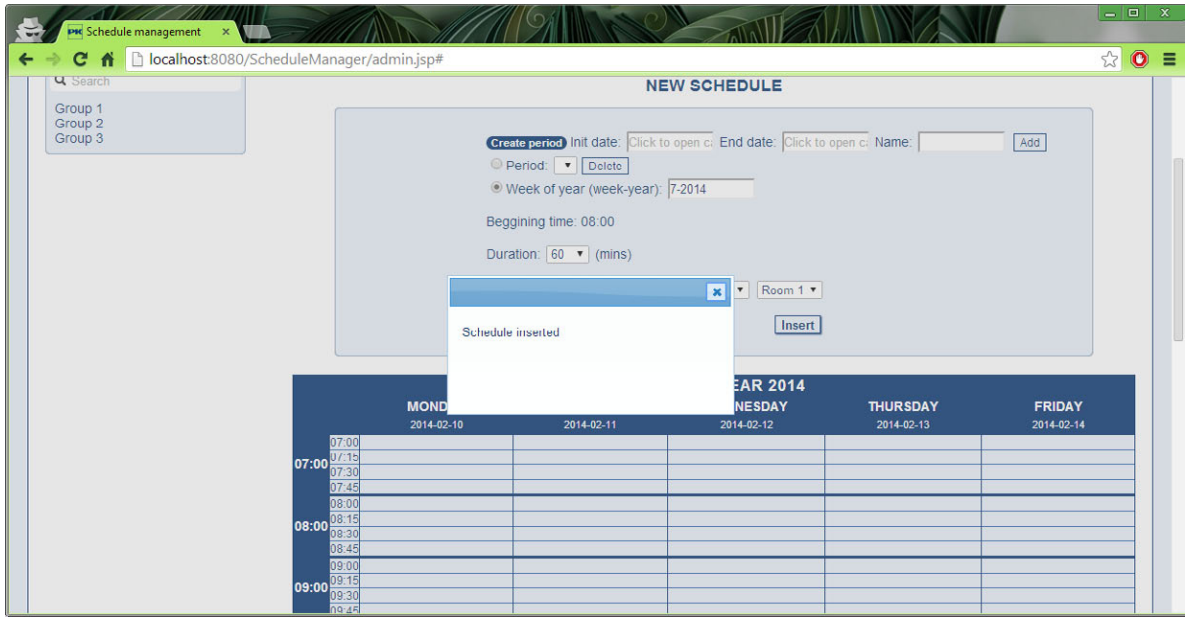

Figure 59. After picking up between the available elements and after clicking Insert, a message informs about the result of the operation.

2. **Create a new schedule in a period**. This process is similar to the previous one, but it is necessary to previously create a period.

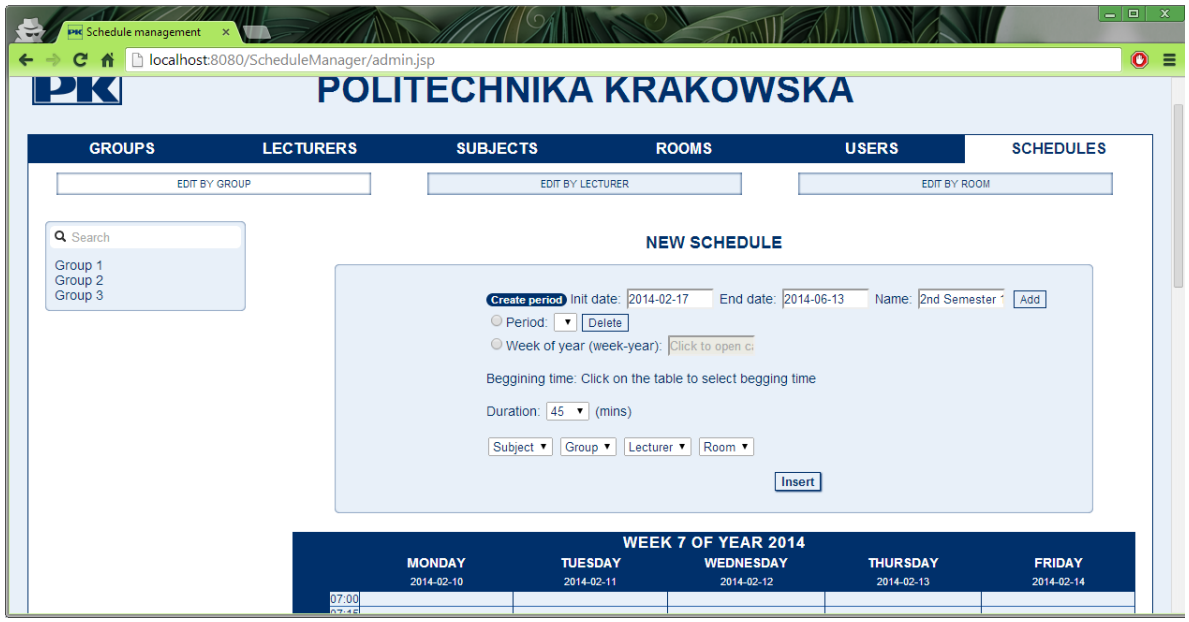

Figure 60. As no periods are created, new dates and a name have been written to create a new one.

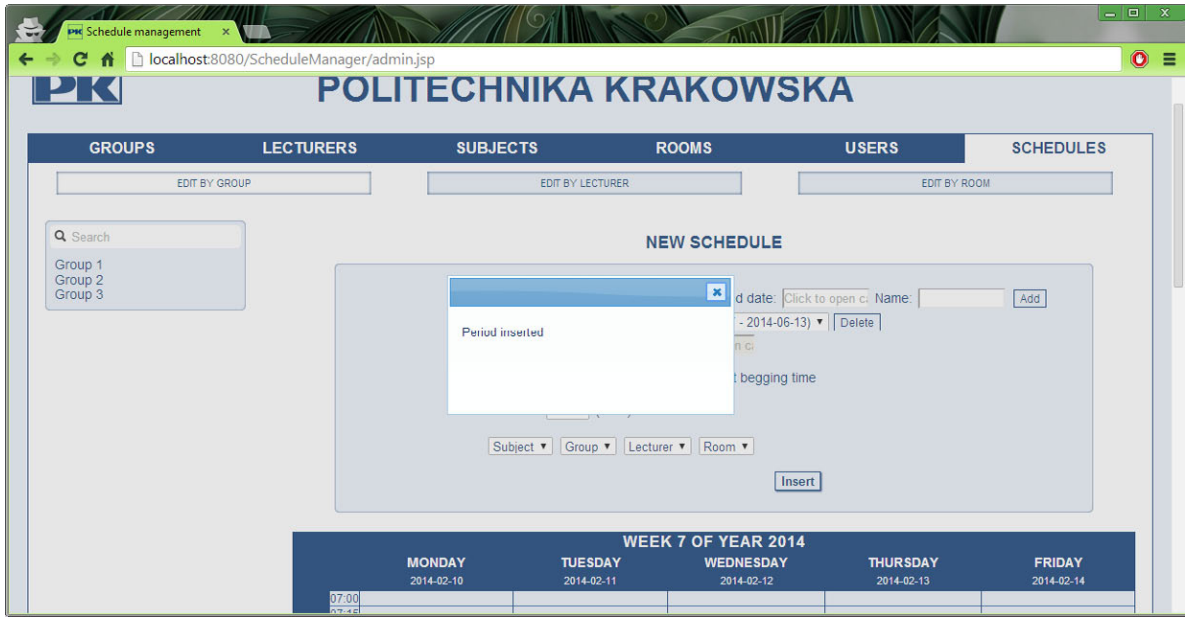

Figure 61. After Adding the period, a message with the result of the operation is shown.

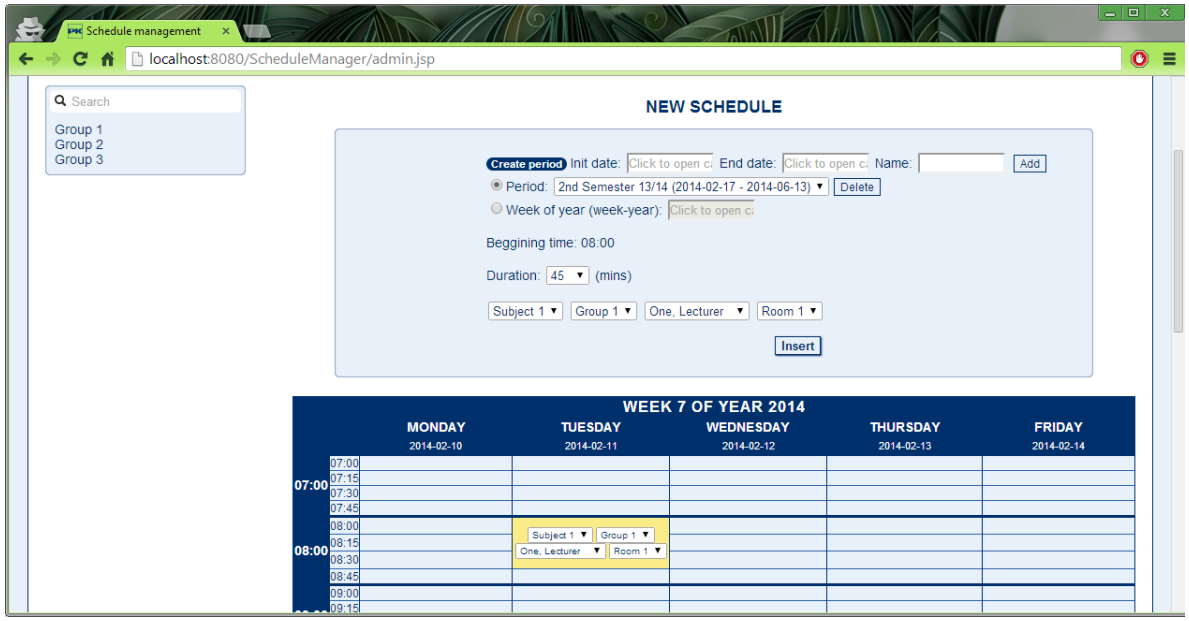

Figure 62. With that period selected, it is possible to set a new schedule. Availability is checked for the whole period this time.

3. **Delete an existing schedule**. Now that we can insert schedules, it is important to delete the schedules in case that there is no more need of them.

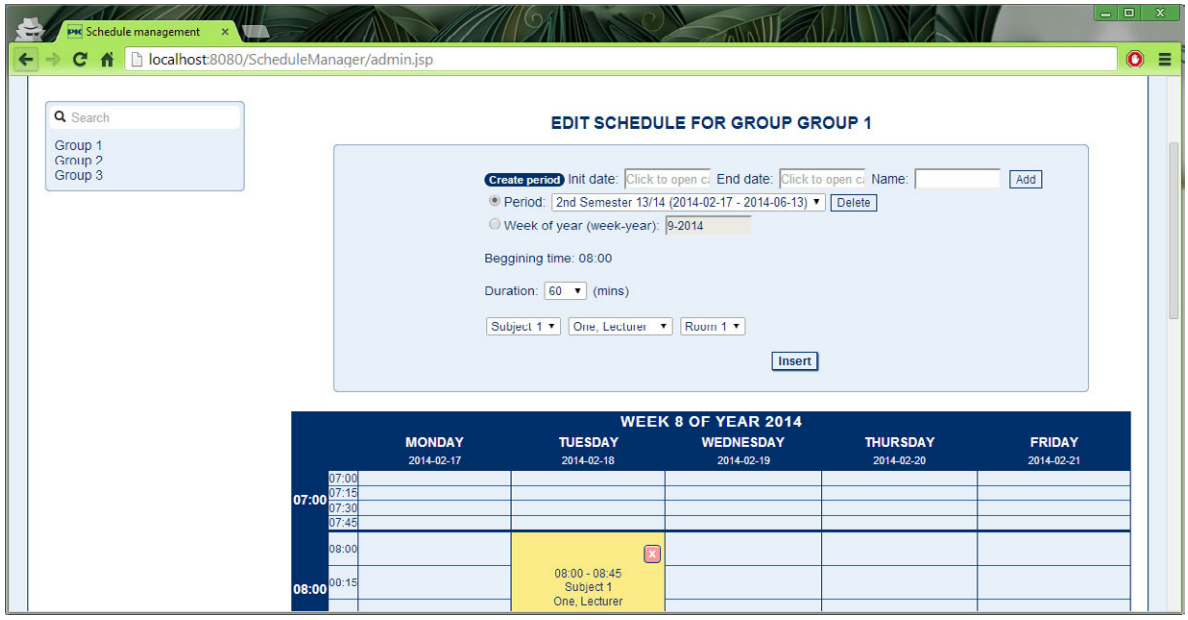

Figure 63. The period and week are used to navigate between the days of the calendar.

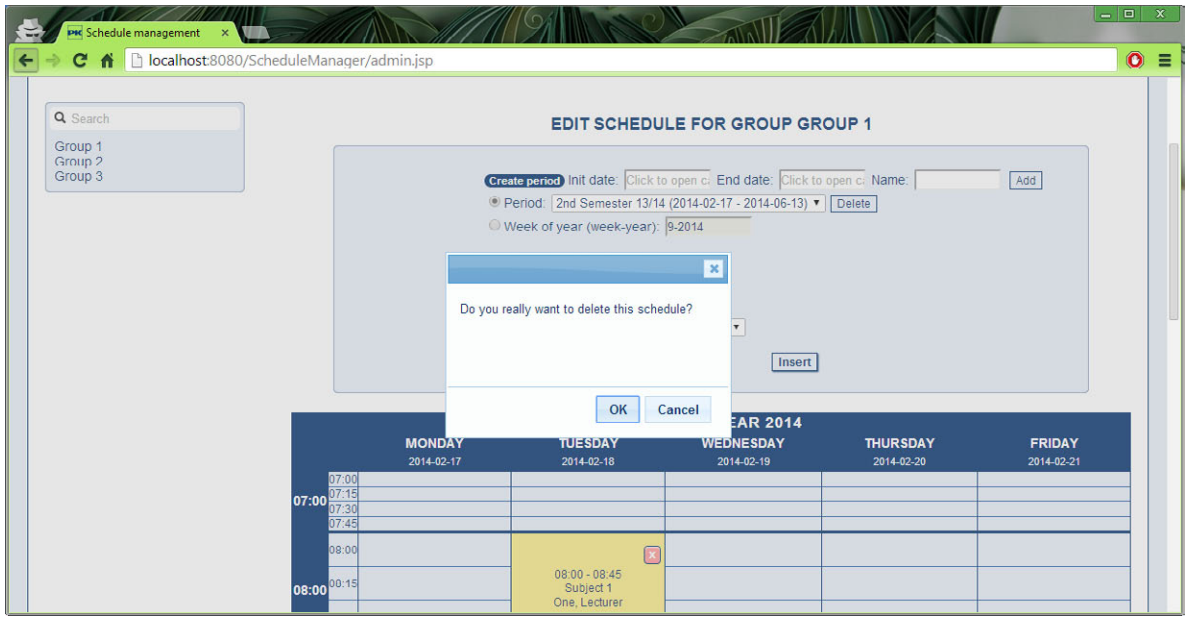

Figure 64. Once the schedule to delete is found, it is necessary to clickk on the top right cross. A confirmation message will appear.

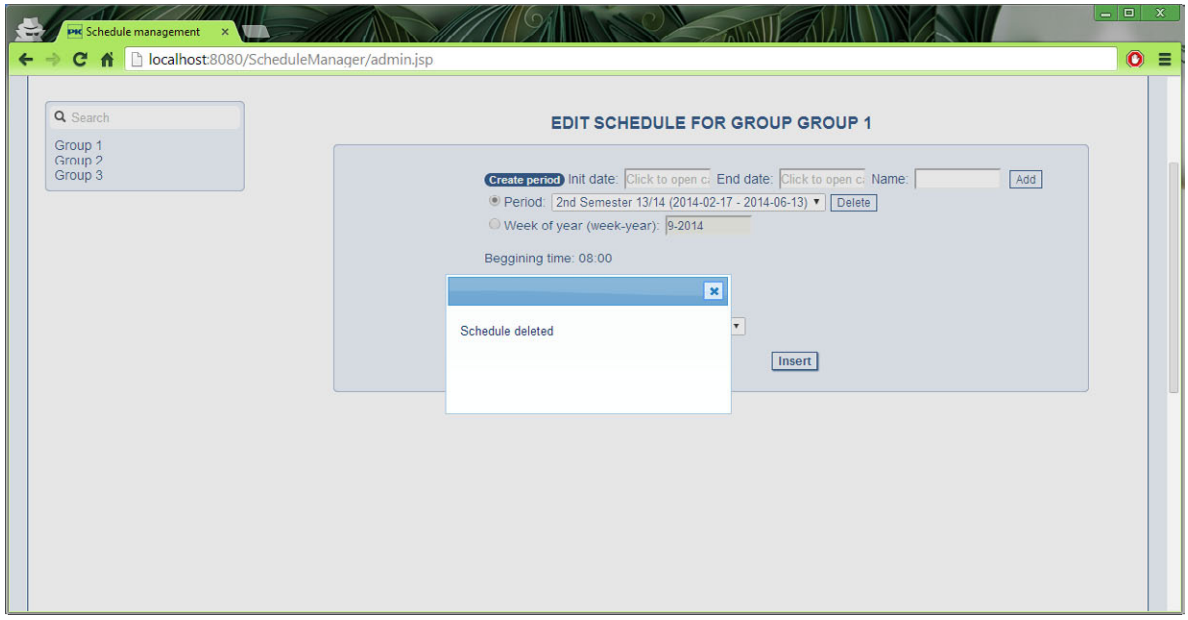

Figure 65. When it is deleted, a message with the result of the operation is shown.

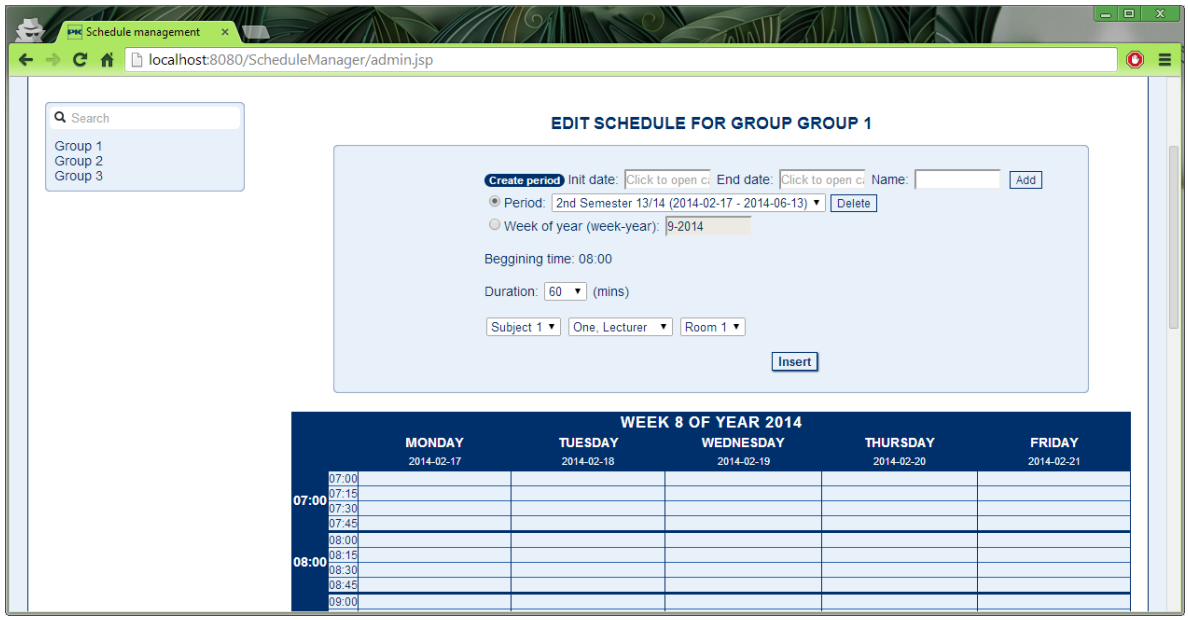

Figure 66. If we go back to that week, the schedule is not there anymore.

4.4.5. Use case "Consult the available timetables"

The last use case we have to check is the one though for the users who don't need to log in. Just entering to the index.jsp page is possible to select an element of any of the existing and see the schedule for that element.

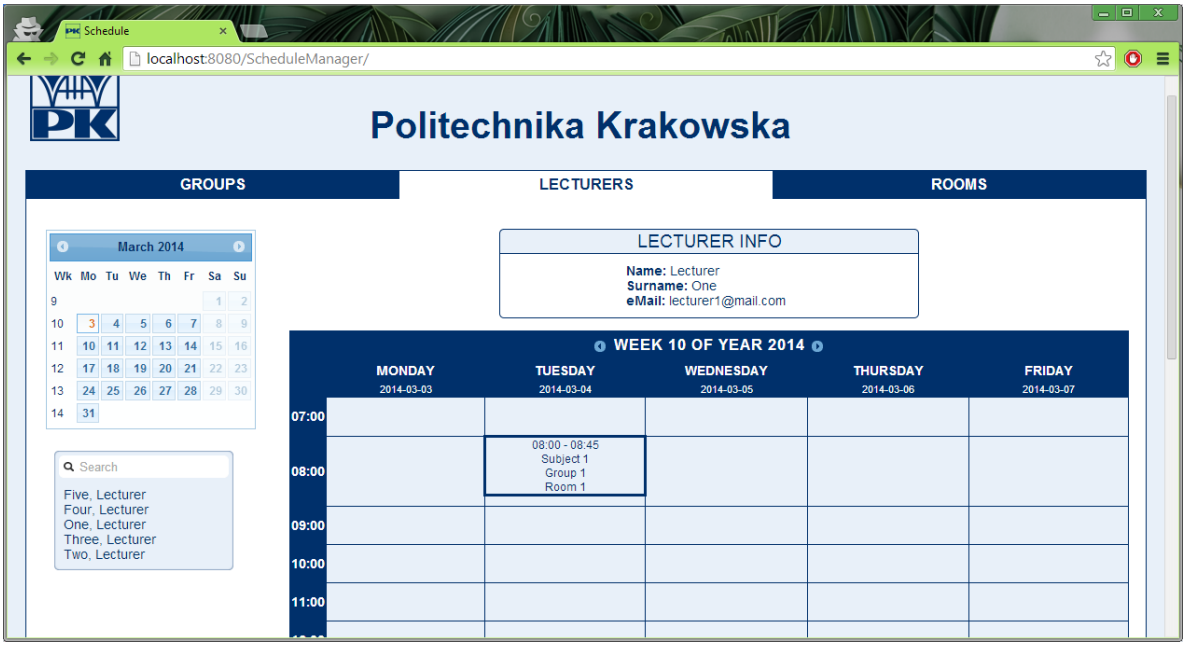

Figure 67. Timetable view from an unregistered user.

4.5. Requisites

The code developed has some limits that are important to be explained in case that limits will suppose a problem for future implementations.

4.5.1. Collision between schedules

Relating to the database and the elements involved, we can find in the model UNIQUE constraints than involve more than one column. This is important to know because it will reduce the number of combinations of that data that can be an entry in the table. The most important UNIQUE defined is the one related to the schedule. If we take a look to the SQL script:

```
-- -----------------------------------------------------
-- Table `mydb`.`Schedule`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `mydb`.`Schedule` (
  `idSchedule` INT NOT NULL AUTO_INCREMENT,
  `idGroup` INT NOT NULL,
  `idRoom` INT NOT NULL,
  `idTime` INT NOT NULL,
  `idSubject` INT NOT NULL,
  `idLecturer` INT NOT NULL,
  PRIMARY KEY (`idSchedule`),
  UNIQUE INDEX `idSchedule_UNIQUE` (`idSchedule` ASC),
  INDEX `idClassroom_idx` (`idRoom` ASC),
  INDEX `idTime_idx` (`idTime` ASC),
  INDEX `idTeacher_idx` (`idLecturer` ASC),
  INDEX `idSubject_idx` (`idSubject` ASC),
  UNIQUE INDEX `GROUP_TIME_UNIQUE` (`idGroup` ASC, `idTime` ASC),
  UNIQUE INDEX `CLASSROOM_TIME_UNIQUE` (`idRoom` ASC, `idTime` ASC),
  UNIQUE INDEX `TEACHER_TIME_UNIQUE` (`idLecturer` ASC, `idTime` ASC),
  CONSTRAINT `idGroup`
    FOREIGN KEY (`idGroup`)
```

```
    REFERENCES `mydb`.`Group` (`idGroup`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idRoom`
    FOREIGN KEY (`idRoom`)
    REFERENCES `mydb`.`Room` (`idRoom`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idTime`
    FOREIGN KEY (`idTime`)
    REFERENCES `mydb`.`Time` (`idTime`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idSubject`
    FOREIGN KEY (`idSubject`)
    REFERENCES `mydb`.`Subject` (`idSubject`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idLecturer`
    FOREIGN KEY (`idLecturer`)
    REFERENCES `mydb`.`Lecturer` (`idLecturer`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

We can see that on this table there cannot be two entries with the same group at the same time or two entries with the same room at the same time or two entries with the same lecturer at the same time.

This has been configured this way because it is logic to think that a lecturer will not be teaching at two different groups at the same time or a room will not have two different groups being taught at the same time.

So if this requisite needs to be solved in a final version, these SQL statements must be revised, as well as the HTLM representation of the timetables (they don't consider collisions with schedules in time neither).

4.5.2. First Administrator

As the credentials stored in the database are Secure Random bytes, it is not easy to move this information through SQL statements. Nevertheless after setting up the platform for the first time, there has to be at least one administrator registered in the database so that he can create more users.

The solution for this not to happen is in the login.jsp page: when checking the credentials of any user, it will first check if there is any user in the database. If the database is empty, it will insert a new user with the administrator role with the credentials user = admin and password = admin.

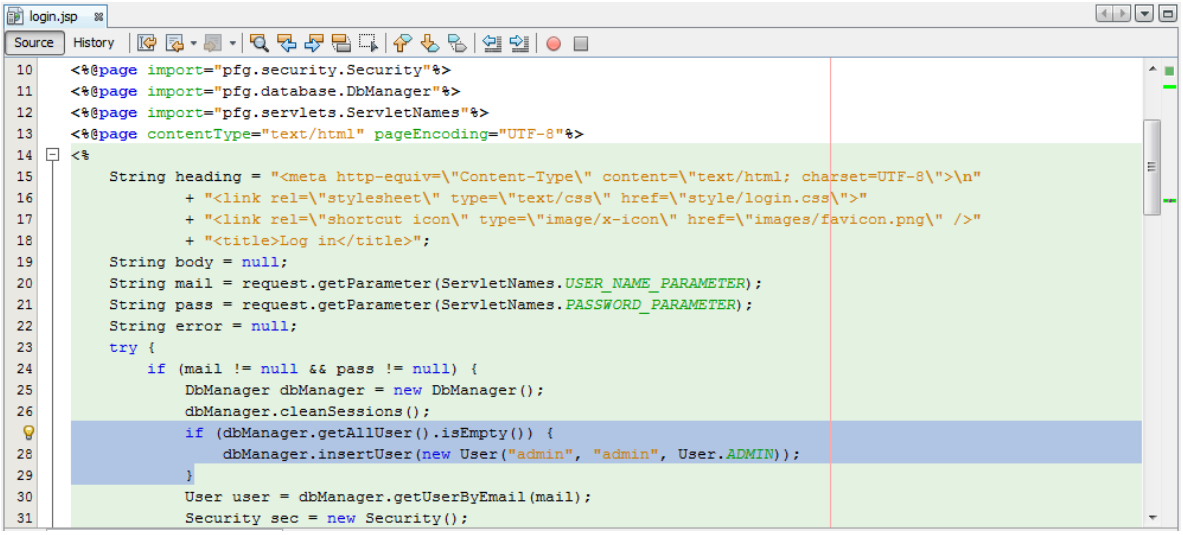

Figure 68. First administrator initialization.

There can be other alternatives to avoid this solution. One of them is creating an external jar that uses DbManager class to insert this first user (or any other inserted by command line). Other solution is the migration of the database through tools provided by MySQL. Some examples are described on the Reference Manual (51).

5. Future work

This project was born with the aim of establishing a first step to create a big platform. During the development of the project, some ideas had to be left in order to set a good base, but in this section they are going to be described so that some of them can ever be developed.

5.1. New options to manage schedules

The current state of the project allows creating schedules for an entire period, but it is not possible to delete all schedules in a period. This problem can be solved by adding a new button on the schedule edit form with this option, and a new table in the database that will join a period with several schedules.

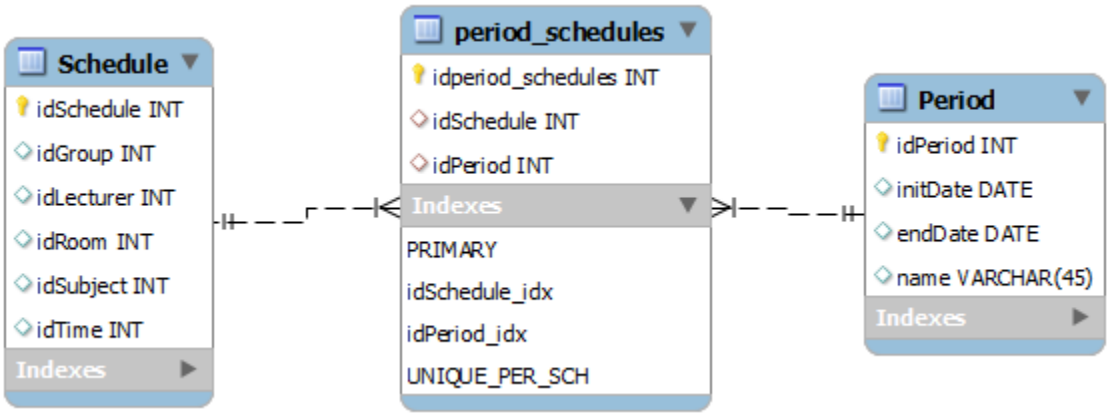

Figure 69. New SQL table to relate Schedules with Periods.

With this table above, it would be easy to delete all schedules belonging to a Period, or all the schedules of the group with groupId X belonging to a Period. A more powerful tool would be provided with small steps.

5.2. More personalization for the users

Currently a user of the system is just a name, a password and a role. Nevertheless, when you register into a webpage, it is common to find more information about you in a corner – for instance a profile picture. In that corner is also common to find options to manage your account (recover password via email, change profile picture…).

These options would make the site more familiar to the user and it would become more accessible.

5.3. Print button on timetables

The timetables shown are useful for the students, but it is likely that they will print the timetable to have it with them. An extra for the platform would be a button in the timetable with the option of transforming the HTML timetable to a PDF timetable.

5.4. Adapt language to locale

The project has been totally developed in English because it is a majoritarian language, but more language options should be provided to the users. The most common solution is to let the user picking up the language he wants to use, but it is also possible to use the locale sent by the client browser to retrieve this information and automatically adapt the language to the locale.

6. Acknowledgments

In first place, I wish to acknowledge the help provided by, D Damian Grela from the Cracow University of Technology for the supervising and help in the whole development of this project. I

would like to offer my special thanks to Dr Radoslaw Czarnecki and Dr Zbigniew Mrozek for the coordination provided during this process.

Apart of these four months devoted to the development of this project, I have to thank the previous years of studies to the Universidad Politécnica de Madrid, in Spain. It not only supposed knowledge acquired, but a whole process of maturity, learning and evolution.

I want to thank also those who participated in my learning of software developing skills. The first time I worked in a software development process on November of 2011, D. Vicente Hernandez Díaz from the Universidad Politécnica de Madrid successfully leaded the project by conceding advices and guidelines to succeed in that task.

The second time I participated in a software process it was during an internship with Ericssion and the Universidad Politéctica de Madrid, and I would like to express my very great appreciation to D. Alejandro Bascuñana Muñoz and D. Alberto Mozo Velasco to make the development team I was involved in learn how properly work together in an I+D project.

Finally I need to thank the support and the guidance that my family has had to me; without their help and encouragement it would not have been easy to reach my achievements.

Bibliography

1. **Q-Success.** Web techhnology Surveys. [Online] [Cited: January 23, 2013.] http://w3techs.com/.

2. **PHP Group.** PHP: Hypertext Preprocessor. [Online] [Cited: October 23, 2013.] http://php.net/.

3. **Microsoft.** ASP.NET: The Official Microsoft ASP.NET Site. [Online] [Cited: October 23, 2013.] http://www.asp.net/.

4. **Oracle Corporation.** Java Servlet Technology. [Online] [Cited: October 23, 2013.] http://www.oracle.com/technetwork/java/index-jsp-135475.html.

5. **Adobe.** Adobe ColdFusion 10 Family: Build, develop, & manage rich internet applications. [Online] [Cited: October 23, 2013.] http://www.adobe.com/es/products/coldfusion-family.html.

6. The Perl Programming Language. [Online] [Cited: October 24, 2013.] http://www.perl.org/.

7. Ruby Programming Language. [Online] [Cited: October 23, 2013.] https://www.ruby-lang.org/.

8. **Python Software Foundation.** Python Programming Language – Official Website. [Online] [Cited: October 23, 2013.] http://www.python.org/.

9. **Apache Software Foundation.** The Apache HTTP Server Project. [Online] [Cited: October 23, 2013.] http://httpd.apache.org/.

10. **Community, Nginx.** [Online] [Cited: October 23, 2013.] http://wiki.nginx.org/Main.

11. **Microsoft.** The Official Microsoft IIS Site. [Online] [Cited: October 23, 2013.] http://www.iis.net/.

12. **LiteSpeed.** LiteSpeed Technologies Web Server. [Online] [Cited: October 23, 2013.] http://www.litespeedtech.com/products/litespeed-web-server/overview.

13. **Alexa.** Alexa - Analytics for any Website. [Online] [Cited: January 24, 2014.] http://www.alexa.com/.

14. **Apache Foundation.** Apache Tomcat. [Online] [Cited: October 23, 2013.] http://tomcat.apache.org/.

15. **Bain, Tony.** ReadWrite - Web Apps, Web Technology Trends, Social Networking and Social Media. [Online] February 12, 2009. [Cited: October 23, 2013.] http://readwrite.com/2009/02/12/is-the-relational-database-doomed.

16. **Bassil, Youssef.** A Comparative Study on the Performance. [Online] February 2012. [Cited: October 23, 2013.] http://arxiv.org/ftp/arxiv/papers/1205/1205.2889.pdf. ISSN 2227-328X.

17. **Microsoft.** Microsoft SQL Server. [Online] [Cited: October 23, 2013.] Microsoft SQL Server. https://www.microsoft.com/en-us/sqlserver/default.aspx.

18. **Oracle.** Orable Database. [Online] [Cited: October 23, 2013.] http://www.oracle.com/us/products/database/overview/index.html.

19. **IBM.** IBM - DB2 database software. [Online] [Cited: October 23, 2013.] http://www-01.ibm.com/software/data/db2/.

20. **Sun Microsystems.** MySQL - The world's most popular open source database. [Online] [Cited: October 23, 2013.] http://www.mysql.com/.

21. **Microsoft.** Microsoft Access - database software and applications. [Online] [Cited: October 23, 2013.] http://office.microsoft.com/en-us/access/.

22. **Eclipse Foundation.** Eclipse - The Eclipse Foundation. [Online] [Cited: October 23, 2013.] http://www.eclipse.org/.

23. **Oracle Corporation.** NetBeans IDE. [Online] [Cited: October 23, 2013.] https://netbeans.org/.

24. **NetBeans.** NetBeans IDE Download. [Online] [Cited: January 28, 2014.] https://netbeans.org/downloads/.

25. NetBeans IDE 7.4 Installation Instructions. [Online] [Cited: January 28, 2014.] https://netbeans.org/community/releases/74/install.html.

26. **Apache Foundation.** Apache Tomcat. *Tomcat 7 Downloads.* [Online] [Cited: January 28, 2014.] http://tomcat.apache.org/download-70.cgi.

27. **Apache Foundation**. Apache Tomcat 7. *Tomcat Setup.* [Online] [Cited: January 28, 2014.] http://tomcat.apache.org/tomcat-7.0-doc/setup.html.

28. **Sun Microsystems.** MySQL Installer 5.6. [Online] [Cited: January 28, 2014.] http://dev.mysql.com/downloads/installer/5.6.html.

29. **Sun Microsystems**. Download MySQL Community Server. [Online] [Cited: January 28, 2014.] http://dev.mysql.com/downloads/mysql/.

30. **Sun Microsystems**. MySQL 5.6 Reference Manual :: 2 Installing and Upgrading MySQL. [Online] [Cited: January 28, 2014.] http://dev.mysql.com/doc/refman/5.6/en/installing.html.

31. **Sun Microsystems**. MySQL 5.6 Reference Manual. *Tutorial.* [Online] [Cited: January 30, 2014.] http://dev.mysql.com/doc/refman/5.6/en/tutorial.html.

32. **Sun Microsystems**. MySQL Workbench 6.0.9. [Online] [Cited: January 29, 2014.] http://dev.mysql.com/downloads/tools/workbench/.

33. **Sun Microsystems**. MySQL. *Download Connector/J.* [Online] [Cited: January 28, 2014.] http://dev.mysql.com/downloads/connector/j/.

34. Unified Modeling Language. [Online] [Cited: November 13, 2013.] http://www.uml.org/.

35. **Oracle.** MySQL Workbench. [Online] [Cited: November 13, 2013.] http://www.mysql.com/products/workbench/.

36. **Oracle**. JDBC Overview. [Online] [Cited: November 13, 2013.] http://www.oracle.com/technetwork/java/overview-141217.html.

37. **Oracle**. HttpServlet (Java EE 6). [Online] [Cited: November 13, 2013.] http://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpServlet.html.

38. **International Organization of Standarization (ISO).** ISO 8601:2004 - Data elements and interchange formats -- Information interchange -- Representation of dates and times. [Online] [Cited: November 13, 2013.] http://www.iso.org/iso/catalogue_detail?csnumber=40874.

39. OAth Community Site. [Online] [Cited: January 28, 2014.] http://oauth.net/.

40. **O'Donnell, Andy.** About.com. *Rainbow Tables: Your Password's Worst Nightmare.* [Online] [Cited: January 28, 2014.] http://netsecurity.about.com/od/hackertools/a/Rainbow-Tables.htm.

41. **NIST.** Recommendation for Password-Based Key. [Online] [Cited: January 28, 2013.] http://csrc.nist.gov/publications/nistpubs/800-132/nist-sp800-132.pdf.

42. **w3schools.** HTTP Methods GET vs POST. [Online] [Cited: January 28, 2014.] http://www.w3schools.com/tags/ref_httpmethods.asp.

43. **Internet Engineering Task Force (IETF).** RFC 5246 - The Transport Layer Security (TLS) Protocol. [Online] [Cited: January 28, 2014.] http://tools.ietf.org/html/rfc5246.

44. **Internet Engineering Task Force (IETF)**. RFC 6101 -. *The Secure Sockets Layer (SSL) Protocol Version 3.0.* [Online] [Cited: January 28, 2014.] http://tools.ietf.org/html/rfc6101.

45. **Oracle.** Apache Tomcat 6.0. *SSL Configuration HOW-TO.* [Online] [Cited: January 28, 2014.] http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html.

46. **Oracle**. MySQL. *Security in MySQL.* [Online] [Cited: January 28, 2014.] http://dev.mysql.com/doc/mysql-security-excerpt/5.1/en/index.html.

47. jQuery UI Blog. [Online] [Cited: November 13, 2013.] http://blog.jqueryui.com/.

48. jQuery UI. [Online] [Cited: November 13, 2013.] http://jqueryui.com/.

49. jQuery. *jQuery 1.10.2 and 2.0.3 Released.* [Online] [Cited: November 13, 2013.] http://blog.jquery.com/2013/07/03/jquery-1-10-2-and-2-0-3-released/.

50. jQuery UI 1.10.3. [Online] [Cited: November 13, 2013.]

51. **Sun Microsystems.** MySQL 5.6 Reference Manual. *14.2.6.2 Moving or Copying InnoDB Tables to Another Machine.* [Online] [Cited: February 10, 2014.] http://dev.mysql.com/doc/refman/5.6/en/innodb-migration.html.

52. Alexa. *Analytics for any Website.* [Online] [Cited: January 24, 2014.] http://www.alexa.com/.

53. **Microsystems, Sun.** MySQL. *Download MySQL Community Server.* [Online] [Cited: January 28, 2014.] http://dev.mysql.com/downloads/mysql/.

Annex I. Installation manual.

In this manual it will be explained how to set up the developed project in order to start to work. For this tutorial, it is necessary to have installed the development tools as defined on section **Installation of the Environment**.

Setup of the database on MySQL Community Server

In first place, we are going to create the database from the model explained in section **Enhanced entity–relationship (EER) model**. The MySQL script for that model is attached both in the code attached to this project and in **Annex III. SQL statements for MySQL initialization**.

We are going to use that script to initialize the database through MySQL Workbench 6.0. The results of those scripts are shown as follows:

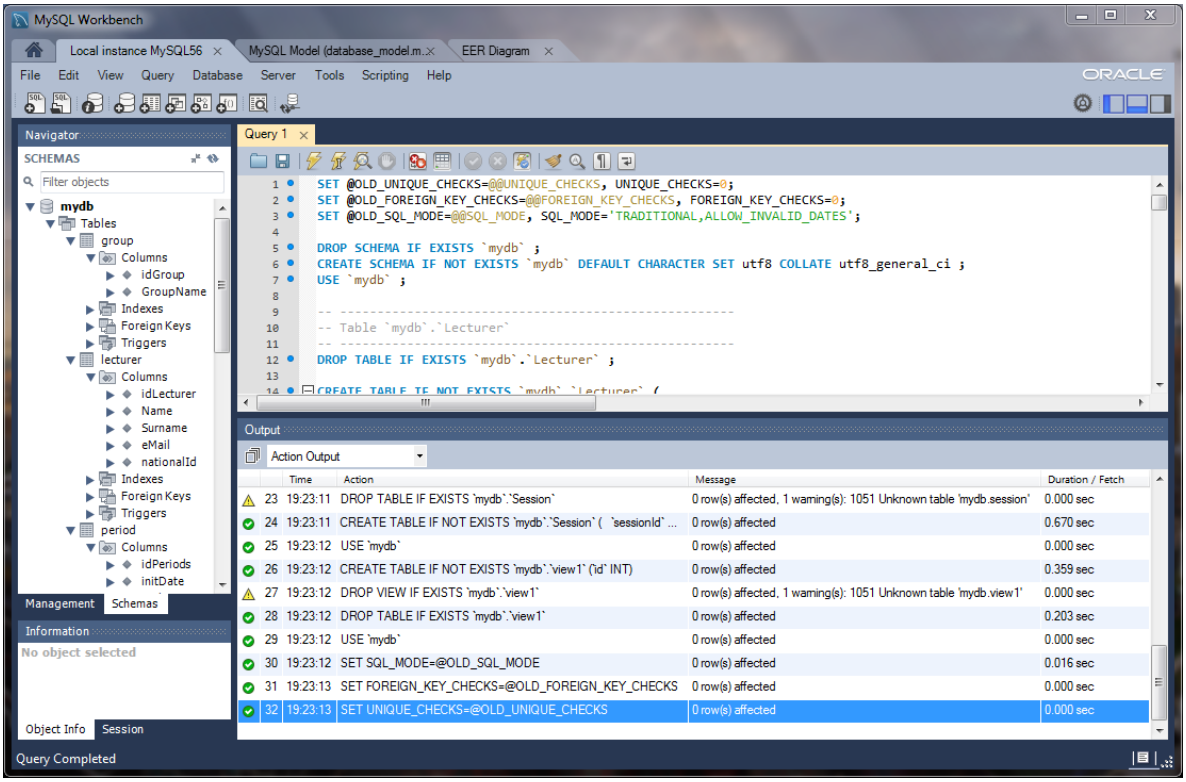

Figure 70. Result of execution of database init script

As we can see on **Figure 70**, the CREATE statements have been successfully queried and the DROP statements warms that there wasn't any table like that created. So the result of all those statements is the database **mydb** as we can see on the left view with the tables created according to the model EER. This empty database will be filled through the web page developed.

Configuration of the project in NetBeans

As the project developed has been exported from NetBean, it is easier to import this proyect to another NetBean. We will see this process and will see that the JDBC library will be also included.

Importing NetBeans project

The NetBeans project to import will be the file ***ScheduleManager.zip***. To import a project in NetBeans we need to click on *File > Import Project > From ZIP…*

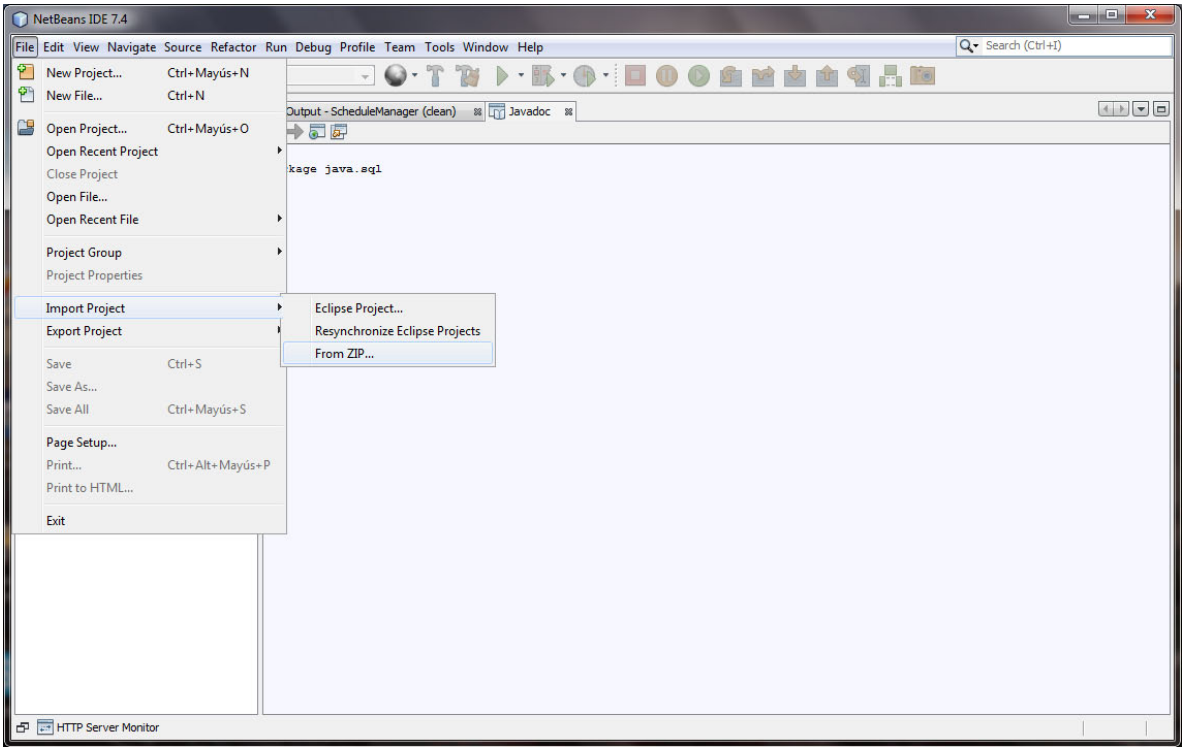

Figure 71. Step 1 to import NetBeans project.

After this step, we are going to be asked about the path where the project to import is located (ZIP File) and where the project must be imported (Folder). When we fill both paths, we click Import.

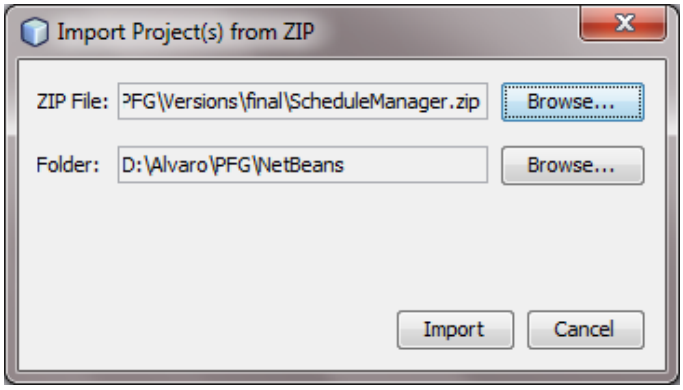

Figure 72. Setp 2 to import NetBeans project.

Finally, we can see the project completely imported without errors (all needed libraries were also imported).

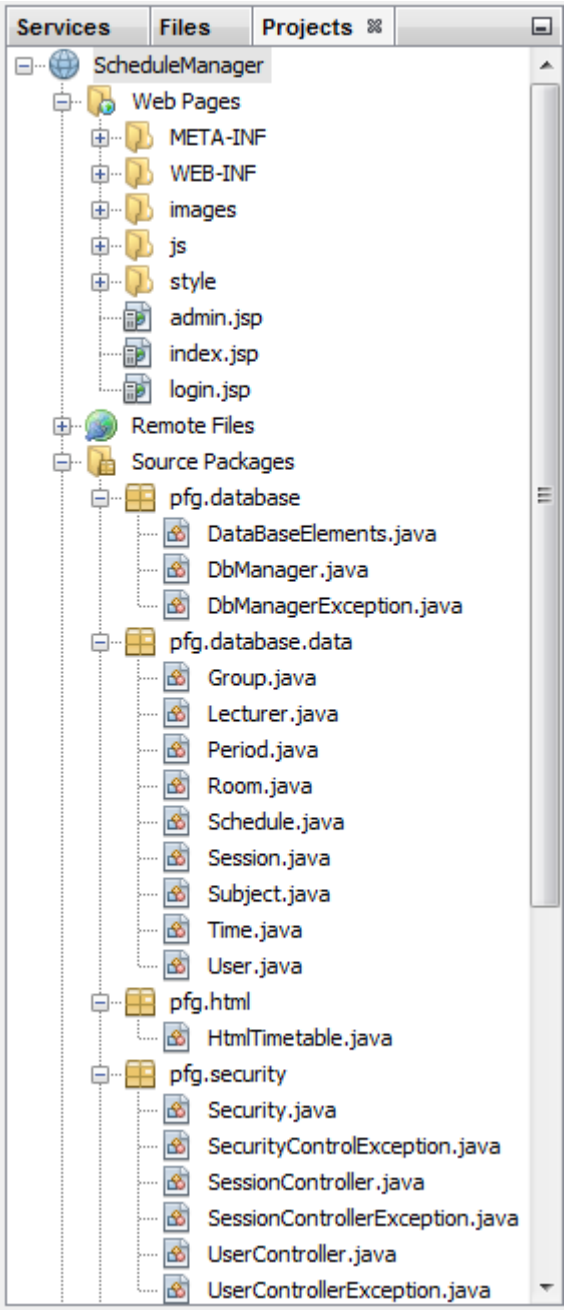

Figure 73. NetBeans project totally imported.

Adding JDBC library

Although JDBC should have been properly added, we are going to explain how it should be imported to the project.

If we downloaded the JDBC from MySQL site as explained in section **JDBC**, we have to follow the steps: *right click on ScheduleManager > Properties > Libraries > Add JAR/Folder*. After that, we look the path of the JAR file and it will finally be added to the project.

Other option is using NetBeans standard libraries downloaded in the version mentioned on section **NetBeans IDE 7.4**. In that case, the steps to follow would be: *right click on ScheduleManager > Properties > Libraries > Add Library…* There, it is possible to select the library *MySQL JDBC Driver*.

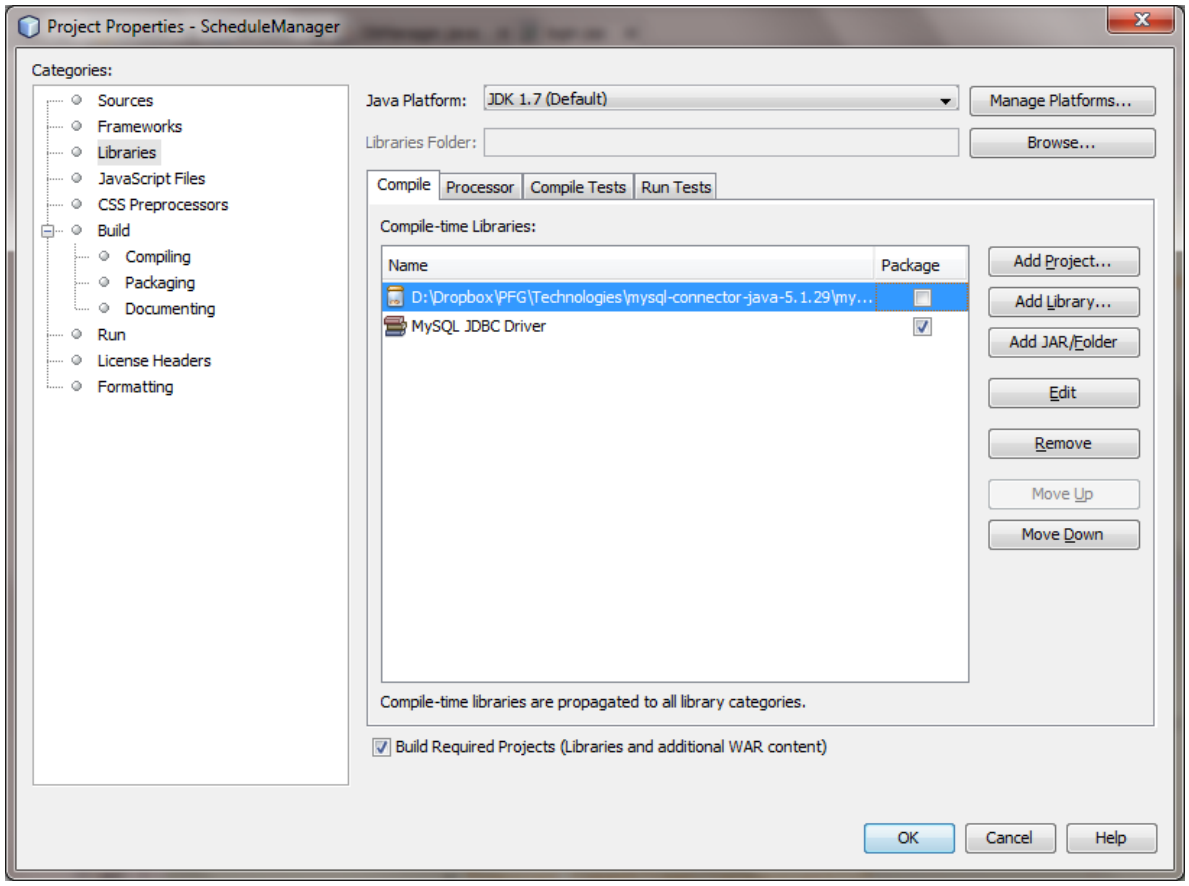

Figure 74. Both JDBC Libraries added to the NetBeans project. Only one should be ticked to avoid collisions.

Adding config.properties file

To finish configuring NetBeans environment, it will be necessary to properly locate the config.properties file (**Annex II. Example of *config.properties* file**). This file has to be located where the libraries for the web project are built. In NetBeans case, it will be in the path where the project is located ("Folder" path from import project from ZIP options). In this path, every time the code is modified, the folder "build" that contains all the generated elements is updated, as well as the libraries used for the deployment of tomcat.

Then the path where locating the config.properties file would be:

Project Folder/build/web/WEB-INF

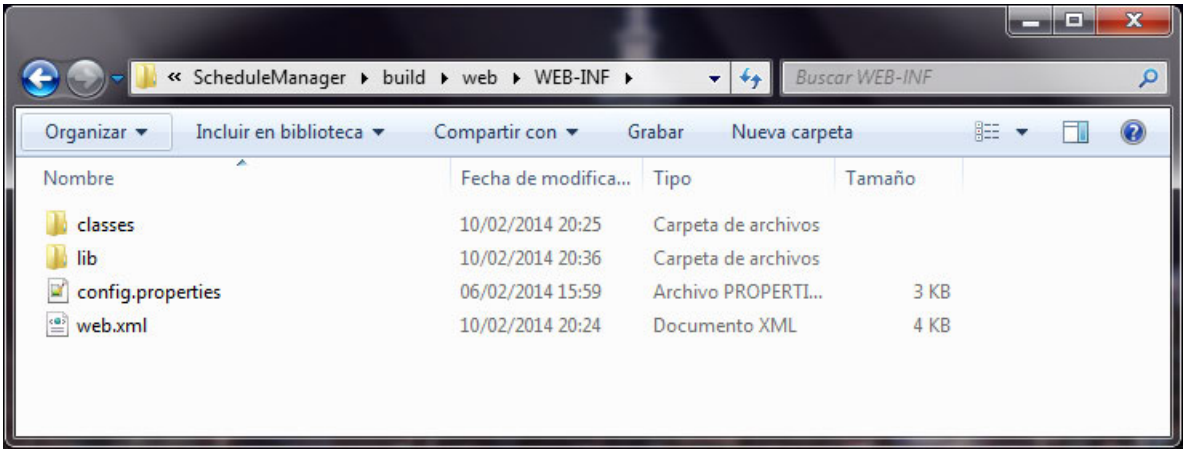

Figure 75. Path to config.properties file

Setup of Apache Tomcat

Once we have the database ready, the project compiled with all his dependencies (JDBC) and the config.properties file properly located, we can run the server. Once more, we are going to configure it from NetBeans to ease the process. If we didn't do it previously, the steps for this configuration are in section **Apache Tomcat 7.X**.

To run the server, we simply have to press the button Run Project of press F6 when the project is selected. A new tab in the browser will automatically be opened with the content of page *index.jsp*.

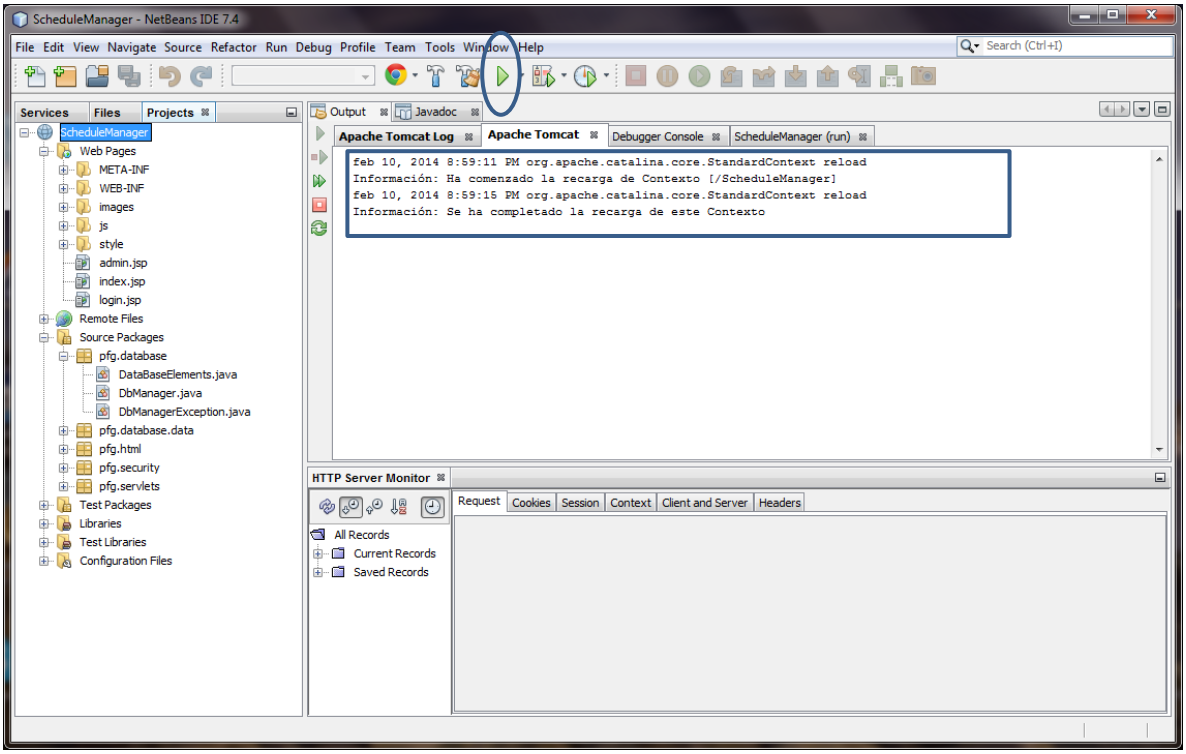

Figure 76. Apache Tomcat running from NetBeans.

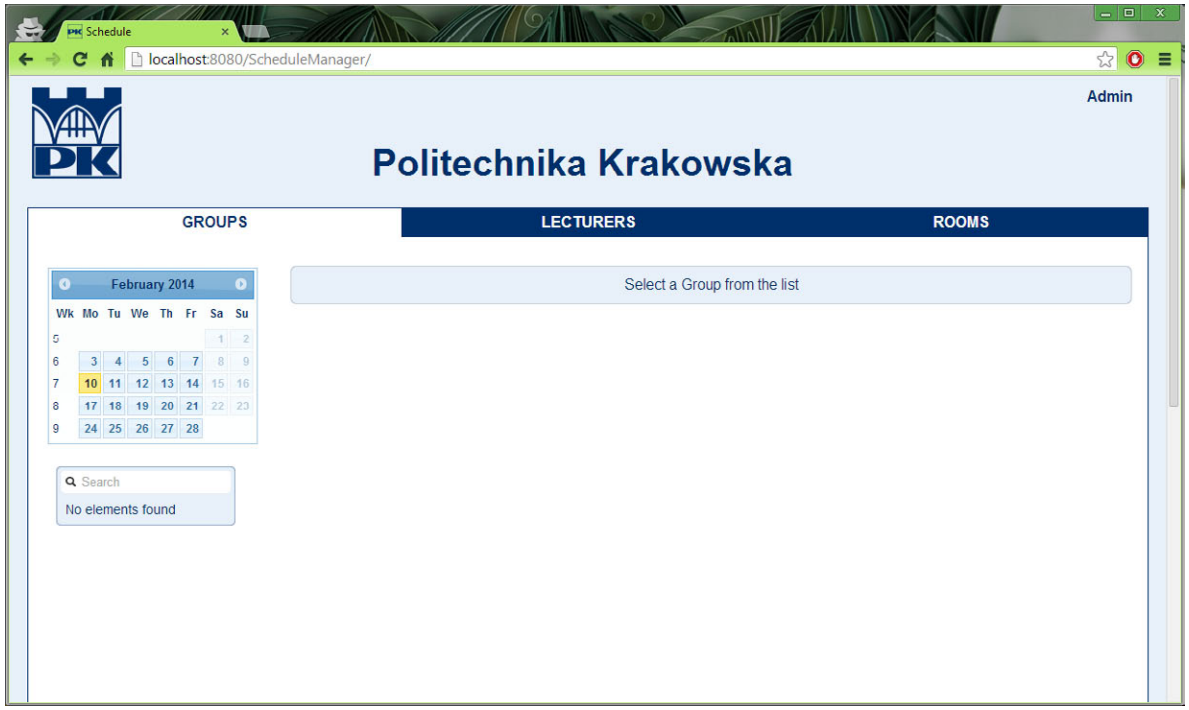

Figure 77. New tab in the browser opened after running Tomcat in NetBeans.

Annex II. Example of *config.properties* file

```
#The data to connect the database
database.url=localhost
database.user=schedule
database.pass=password
database.name=mydb
#The data related to lecturer table
database.table.name.lecturer=lecturer
database.lecturer.field.id=idLecturer
database.lecturer.field.national.id=nationalId
database.lecturer.field.name=name
database.lecturer.field.surname=surname
database.lecturer.field.email=eMail
#The data related to subject table
database.table.name.subject=subject
database.subject.field.id=idSubject
database.subject.field.name=name
#The data related to group table
database.table.name.group=group
database.group.field.id=idGroup
database.group.field.name=GroupName
#The data related to room table
database.table.name.room=room
database.room.field.id=idRoom
database.room.field.name=Name
#The data related to time table
database.table.name.time=time
database.time.field.id=idTime
database.time.field.begginig=Beginning
database.time.field.duration=Duration
database.time.field.day=Day
database.time.field.week=Week
database.time.field.year=Year
#The data related to the period table
database.table.name.period=period
database.period.field.id=idPeriods
database.period.field.init.date=initDate
database.period.field.end.date=endDate
database.period.field.name=name
#The data related to the user table
database.table.name.user=user
database.user.field.id=userId
database.user.field.email=email
database.user.field.salt=salt
database.user.field.password=password
```

```
database.user.field.role=role
#The data related to the session table
database.table.name.session=session
database.session.field.id=sessionId
database.session.field.email=email
database.session.field.timestamp=timestamp
#The data related to the schedule table
database.table.name.schedule=schedule
database.schedule.field.id=idSchedule
database.schedule.field.group.id=idGroup
database.schedule.field.room.id=idRoom
database.schedule.field.time.id=idTime
database.schedule.field.subject.id=idSubject
database.schedule.field.lecturer.id=idLecturer
```

Annex III. SQL statements for MySQL initialization

The following SQL statements are set for a database with name 'mydb'. For other database name, it is necessary to replace it.

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

DROP SCHEMA IF EXISTS `mydb` ;
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8
COLLATE utf8_general_ci ;
USE `mydb` ;

-- -----------------------------------------------------
-- Table `mydb`.`Lecturer`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Lecturer` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Lecturer` (
  `idLecturer` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(45) NOT NULL,
  `Surname` VARCHAR(100) NOT NULL,
  `eMail` VARCHAR(100) NULL,
  `nationalId` VARCHAR(45) NULL,
  PRIMARY KEY (`idLecturer`),
  UNIQUE INDEX `ID_UNIQUE` (`idLecturer` ASC),
  UNIQUE INDEX `nationalId_UNIQUE` (`nationalId` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Room`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Room` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Room` (
  `idRoom` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idRoom`),
  UNIQUE INDEX `Class_ID_UNIQUE` (`idRoom` ASC),
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Group`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Group` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Group` (
  `idGroup` INT NOT NULL AUTO_INCREMENT,
  `GroupName` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idGroup`, `GroupName`),
  UNIQUE INDEX `GroupName_UNIQUE` (`GroupName` ASC),
  UNIQUE INDEX `Group_ID_UNIQUE` (`idGroup` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Time`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Time` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Time` (
  `idTime` INT NOT NULL AUTO_INCREMENT,
  `Beginning` TIME NOT NULL,
  `Duration` INT UNSIGNED NOT NULL DEFAULT 45,
  `Day` INT UNSIGNED NOT NULL,
  `Week` INT UNSIGNED NOT NULL,
  `Year` YEAR NOT NULL,
  PRIMARY KEY (`idTime`),
  UNIQUE INDEX `Time_ID_UNIQUE` (`idTime` ASC),
  UNIQUE INDEX `Time_UNIQUE` (`Beginning` ASC, `Duration` ASC,
`Day` ASC, `Week` ASC, `Year` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Subject`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Subject` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Subject` (
  `idSubject` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idSubject`),
  UNIQUE INDEX `Subject_ID_UNIQUE` (`idSubject` ASC),
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `mydb`.`Schedule`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Schedule` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Schedule` (
  `idSchedule` INT NOT NULL AUTO_INCREMENT,
  `idGroup` INT NOT NULL,
  `idRoom` INT NOT NULL,
  `idTime` INT NOT NULL,
  `idSubject` INT NOT NULL,
  `idLecturer` INT NOT NULL,
  PRIMARY KEY (`idSchedule`),
  UNIQUE INDEX `idSchedule_UNIQUE` (`idSchedule` ASC),
  INDEX `idClassroom_idx` (`idRoom` ASC),
  INDEX `idTime_idx` (`idTime` ASC),
  INDEX `idTeacher_idx` (`idLecturer` ASC),
  INDEX `idSubject_idx` (`idSubject` ASC),
  UNIQUE INDEX `GROUP_TIME_UNIQUE` (`idGroup` ASC, `idTime` ASC),
  UNIQUE INDEX `CLASSROOM_TIME_UNIQUE` (`idRoom` ASC, `idTime`
ASC),
  UNIQUE INDEX `TEACHER_TIME_UNIQUE` (`idLecturer` ASC, `idTime`
ASC),
  CONSTRAINT `idGroup`
    FOREIGN KEY (`idGroup`)
    REFERENCES `mydb`.`Group` (`idGroup`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idRoom`
    FOREIGN KEY (`idRoom`)
    REFERENCES `mydb`.`Room` (`idRoom`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idTime`
    FOREIGN KEY (`idTime`)
    REFERENCES `mydb`.`Time` (`idTime`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idSubject`
    FOREIGN KEY (`idSubject`)
    REFERENCES `mydb`.`Subject` (`idSubject`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `idLecturer`
```

```
    FOREIGN KEY (`idLecturer`)
    REFERENCES `mydb`.`Lecturer` (`idLecturer`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`User`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`User` ;

CREATE TABLE IF NOT EXISTS `mydb`.`User` (
  `userId` INT NOT NULL AUTO_INCREMENT,
  `email` VARCHAR(45) NOT NULL,
  `salt` BINARY(8) NOT NULL,
  `password` BINARY(20) NOT NULL,
  `role` INT NOT NULL,
  UNIQUE INDEX `email_UNIQUE` (`email` ASC),
  PRIMARY KEY (`userId`),
  UNIQUE INDEX `UserId_UNIQUE` (`userId` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Period`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Period` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Period` (
  `idPeriods` INT NOT NULL AUTO_INCREMENT,
  `initDate` DATE NOT NULL,
  `endDate` DATE NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idPeriods`),
  UNIQUE INDEX `idPeriods_UNIQUE` (`idPeriods` ASC),
  UNIQUE INDEX `date_UNIQUE` (`initDate` ASC, `endDate` ASC),
  UNIQUE INDEX `name_UNIQUE` (`name` ASC))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Session`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Session` ;
```

```
CREATE TABLE IF NOT EXISTS `mydb`.`Session` (
  `sessionId` BINARY(8) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `timestamp` TIMESTAMP NOT NULL,
  PRIMARY KEY (`sessionId`),
  UNIQUE INDEX `sessionId_UNIQUE` (`sessionId` ASC),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC),
  CONSTRAINT `email`
    FOREIGN KEY (`email`)
    REFERENCES `mydb`.`User` (`email`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

USE `mydb` ;
```