
# Developing a RESTful Web Application for GAL System

**Elene Gulordava**

Dissertation submitted to the School of Technology and Management of Bragança to obtain the Master's Degree in Information Systems.

Work supervised by:

Albano Alves and Mikheil Rukhaia

This work does not include the comments and suggestions made by the Jury.

Bragança

2016


# Developing a RESTful Web Application for GAL System

**Elene Gulordava**

Dissertation submitted to the School of Technology and Management of Bragança to obtain the Master's Degree in Information Systems.

Work supervised by:

Albano Alves and Mikheil Rukhaia

This work does not include the comments and suggestions made by the Jury.

Bragança

2016

# Dedication

I dedicate to my Parents.

# Abstract

In today's reality of Information Technology, we are able to automate many manual work. This fact behaves as a motivator to make job easier for the stuff and appropriate for nowadays standards, rapidly growing around us.
Implemented work serves to exploit manual setup part used in Schedule Generator System of Polytechnic Institute of Braganca (GAL). Around this system work many applications and our new teaching duties project will be one of them. In more detail the problem concerns to assign curricular units to the teachers and departments. In order to solve this issue we will demonstrate Spring Boot Framework as a flexible technology for building RESTful Web Application. It will be guideline of technologies for future related works. Teaching duties project suggests starting point for complete integration with one of the private system in Polytechnic Institute of Bragança (IPB). Eventually commonly used technologies will give us way for proper arrangement of the problem with security issues using Java security framework, which is an Apache Shiro in our case. All the other instruments for the implementation of RESTful Web Application will be matched to Spring Boot Framework and exactly this capability will help us to solve problem without enforcing huge strength.

# Resumo

Na realidade atual das Tecnologias de Informação, somos capazes de automatizar muitas tarefas que no passado eram feitas manualmente. Este facto funciona como motivação para tornar o trabalho mais fácil para as pessoas e adequado às normas de hoje em dia, que rapidamente se impõe à nossa volta. Esta influência positiva da tecnologia fácil trouxe-nos uma ideia para resolver um problema existente, com recurso a uma solução aplicacional. A questão principal é entender o problema na sua essência e determinar como podemos contribuir para a sua solução. Este trabalho serviu para contornar o problema da configuração manual necessária para utilizar o sistema gerador de horários do Instituto Politécnico de Bragança (IPB), denominado Gestão de Atividades Letivas (GAL). Em torno deste sistema, são já usadas muitas plataformas e este projeto será um novo serviço que ficará disponível para informatizar o Serviço Docente. O problema principal consiste na atribuição de unidades curriculares a departamentos e posteriormente a docentes, de forma simples e sem recurso ao papel. Para resolver esta questão, usou-se a Spring Boot Framework, como uma tecnologia flexível para a construção de uma aplicação web RESTful. A metodologia usada servirá também como orientação para futuros trabalhos relacionados com o SI do IPB. O projeto de informatização do Serviço Docente tem como requisito fundamental a integração com o sistema de informação do IPB. As tecnologias usadas nas várias plataformas informáticas do IPB permitiram solucionar questões de segurança, com a utilização da framework Java, recorrendo ao Apache Shiro. Todas as restantes ferramentas utilizadas na implementação da aplicação web RESTful foram integradas na plataforma Spring Boot Framework, o que permitiu resolver os problemas de codificação sem um grande esforço.

# Acknowledgements

I'm very thankful to Professor Albano Alves and Mikheil Rukhaia, my supervisors from Portugal and Georgia for all the supports during this project. Without their inspiration, motivation and huge effort this work wouldn't be complete.

A special thanks to developers in IPB Luis Lobo and Filipe Sousa for their technical support, giving me hand in every critical situation and guaranteeing the best working environment.

To my family, being always with me and giving a lot of human positives especially throughout my master academic year.

I'm grateful to my Mum and Dad, two important persons of my life. They were always supporters of my ideas and enthusiasts for me to successfully overcome all the obstacles.

Warm thanks to all my friends becoming me stronger and more concentrated to my successfully done work.

Thanks to Polytechnic Institute of Braganca for giving me chance to have working experience with all these amazing people.

Yes we did it...

To all my thanks.

# Contents

# List of Figures

# Acronyms

**API** Application Programming Interface.

**CRUD** Create, Read, Update, Delete operations.

**CSS** Cascading Style Sheets.

**DAO** Database Access Object.

**DOM** Document Object Model.

**GAL** Schedule Generator System of Polytechnic Institute of Braganca.

**HTML** Hyper Text Markup Language.

**HTTP** Hypertext Transfer Protocol.

**IDE** Integrated Development Environment.

**IPB** Polytechnic Institute of Bragança.

**IS** Information System.

**JSON** JavaScript Object Notation.

**JVM** Java Virtual Machine.

**LDAP** Lightweight Directory Access Protocol.

**MVC** Model View Controller architecture.

**POJO** Plain Old Java Object.

**RDBMS** Relational Database Management System.

**REST** Representational State Transfer.

**RPC** Remote Procedure Call.

**SDK** Software development kit.

**SOAP** Simple Object Access Protocol.

**SQL** Structured Query Language.

**URI** Uniform Resource Identifier.

**URL** Uniform Resource Locator.

**WAR** Web Application Archive.

**XML** Extensible Markup Language.

# Chapter 1

# Introduction

Nowadays, in technological revolution century most of problem that appears in variety of organizations has been solved by information systems. In general Information System (IS) serves to collect technical and human resources in order to provide the storage, computing, distribution and communication for the information required by all or some part of organization members. In educational organization such as IPB problem appeared during the assignment process of teaching duties. In architectural viewpoint Polytechnic Institute of Braganca is divided by five separate schools according their study areas. Each school has various departments corresponding their directions. For particular departments there is one responsible coordinator. Each department includes many teachers and curricular units. The work through the department goes these following steps. First of all each direction provides list of curricular units. On the department side there is made paper distribution of the services. It goes ahead for scientific consult to approve and finishes with manual setup from the direction of school for the GAL system. All the assignment of curricular units to the department or teacher are done manually, using paper. In relationship view we have "many to many" connection between them. There are a lot of curricular units, teachers and departments which are needed to be arranged to each other. Respectively, human resources job is associated with hard and confusing work. The paper distribution of huge services and its manual setup requires right planning of processes and lot of time as well.

## 1.1 Context

The working system in IPB stands on collection of different tools. There exists database called academic service, which contains information about students, courses, curricular units and their classification. Around this data storage work different web or desktop applications, some of them are virtual apps also. There is opportunity to be connected to each other all the separate applications around the academic service. Eventually the GAL system is a desktop application which generates schedule for the students and relates to the other application called Sumarios, which serves the complete form of schedule of the year. Before getting start with the application for authentication process, there exists Lightweight Directory Access Protocol (LDAP). IPB uses it to develop a central place for storing username and passwords, that allows many different applications with their services to connect to the LDAP server for validate users in order to check if users identity and credential are correct. Each new service provided for the university makes integration with LDAP server, inclusively for this time developed teaching duties web application, which is the new working solution on GAL and academic service combination field.

## 1.2 Objectives

In order to prevent and eradicate manual setup part in assignment process of teaching duties we will develop a Single Page Web Application. Responsible user from each department will have access to the proper module according their Rules and Permissions. There will be two levels of access: Assignments and Curricular Units. In particular application responsible coordinator will be able to assign curricular units to the teachers or to the departments. The global picture of distribution curricular units ,departments and teachers, integrated with new application implementation on the database design view looks like this: Figure 1.1 shows standard structure of distribution teachers, departments and curricular units.

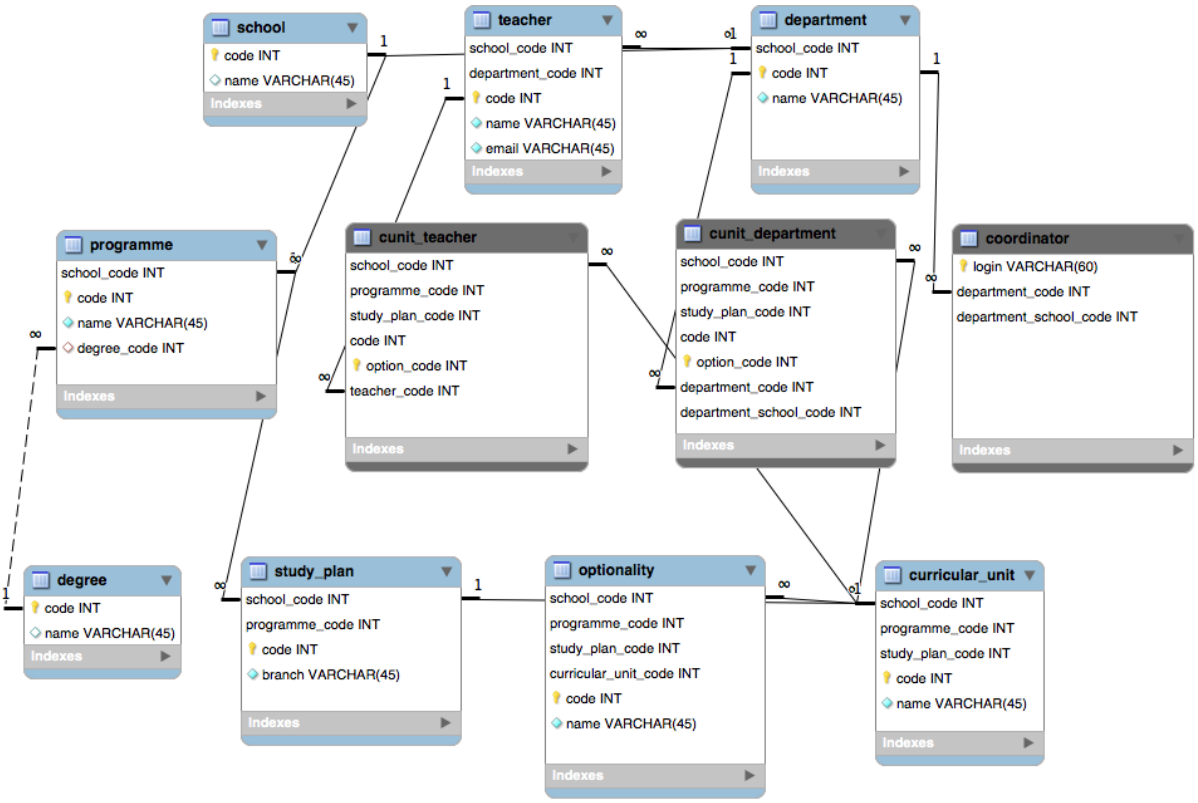


Figure 1.1: Distribution of teachers, departments and curricular units.

# Chapter 2

# Solution

## 2.1 Concepts and Technologies

In order to solve existing problem the main idea is to think about appropriate algorithm, technology, that will give a hand to do work easily and comfortably. The first issue stands on database side. Its design view helps to gain a global picture of the system to decide in which way it could be match with new solutions. After getting point of the problem, analyzing details, discussing and thinking as a client, comes in order information about the data model, tables and relationship between them. In assignment process of curricular units to the teachers we need to connect already existing two tables: curricular units and teachers. A curricular unit can be assigned to a variety of teachers. As a reverse case a teacher can be assigned to a different curricular units. We got that identification of relation is "many to many", which means that a new table will appear between curricular units and teachers. The same approach is used for the departments. Figure 2.1 shows relationship between curricular units and teachers. Figure 2.2 shows relationship between curricular units and department. On the application view side there are two profiles like a Role: Pivot and Coordinator. Each Role is assigned to the specific user interface such as Curricular Units and Assignments modules. The Curricular Units interface provides to assign curricular units to the teachers, second module serves assignment of curricular units

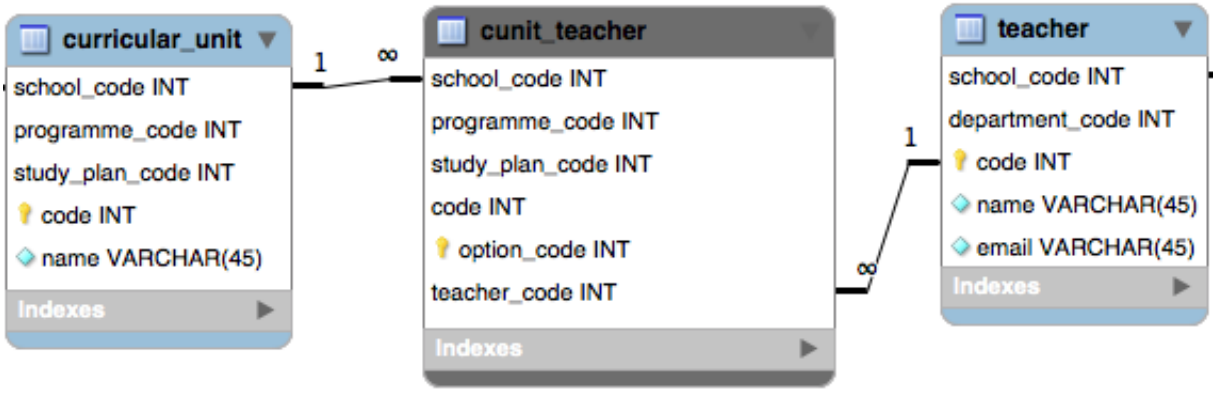


Figure 2.1: Relationship between curricular units and teachers.

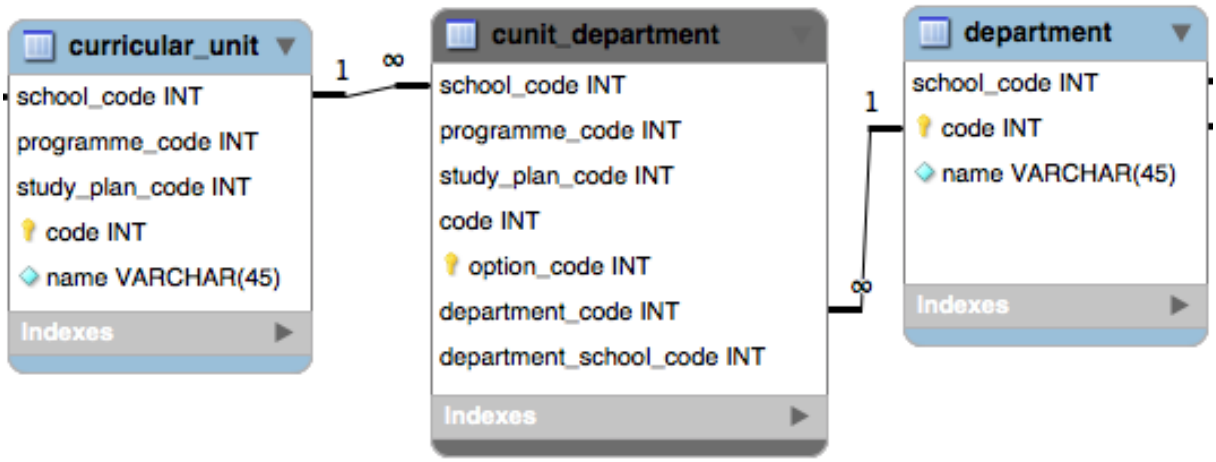


Figure 2.2: Relationship between curricular units and departments.

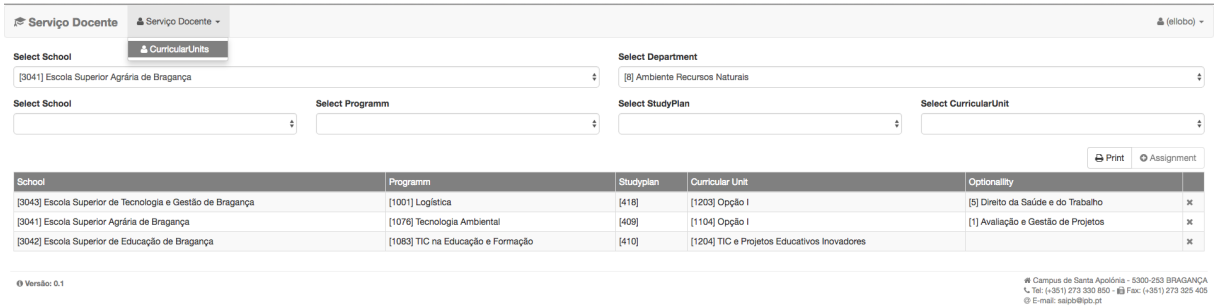


Figure 2.3: Curricular Units module view.

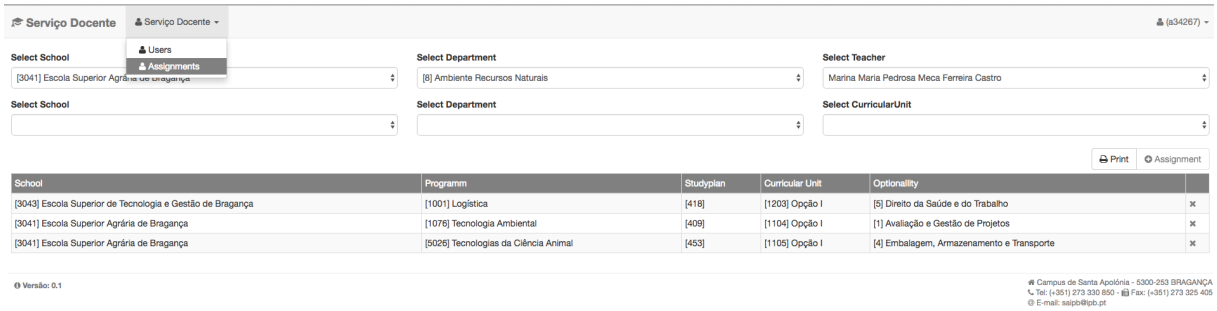


Figure 2.4: Assignments module view.

to the departments. Opportunities for authorized users on Curricular Units module are to select school and department, see the view if there exist some assignments according for already chosen values or add new curricular units to the department by identifying school, program, study plan and curricular unit. There is also opportunity to delete unnecessary record. On Assignments module side is required to choose school, department, teacher and check if there are already assigned curricular units to the selected teacher or add new assignment or delete. Figure 2.3 shows assignment view of curricular units to the department. Figure 2.4 shows assignment view of curricular units to the teacher.

## 2.2 Restful Web Application description

More academically and abstract way as a solution we have gotten RESTful web application, which combines services that are made to work best on the web. It's more a collection

of principles than it is a set of standards. Representational State Transfer (REST) is a hybrid style coming from several of the network-based architectural styles such as data flow, replication, hierarchical or other styles. REST is a frequently used approach in the development of web services. The advantage of REST is commonly preferred over the more heavyweight Simple Object Access Protocol (SOAP) style. The view point of SOAP insists writing or using a server program in order to serve data and client program as well to request this already mentioned data, which actually doesn't make a better fit for using over the internet. REST has a decoupled architecture style and light weight relationship between client and server. In its architectural style, there are several constraints such as decouple consumers from producers, stateless existence, able to leverage a cache, layered system, uniform interface. In the comparison process with SOAP there appears several advantages of REST. Its Web services are simple to use a lever by most tools, inclusive those that are free and not costing a lot of money. SOAP represents harder services than REST. Representational State Transfer architecture is commonly used some kind of services which are defenseless via the Internet such as Twitter or Facebook. Communication within REST application is faster then within SOAP based application. This property of the REST saves time which saves money as well. It applies a short message format than SOAP. For all the messages SOAP utilizes just Extensible Markup Language (XML) that increases the message size which gives less quality. Particularly REST is created for using over the Internet or Web which means that it's a better choice to serve network applications. The widely distributed application of REST is the World Wide Web itself, which utilized it as a basics for Hypertext Transfer Protocol (HTTP) 1.1 development. It has various of benefits, but doesn't represent a pattern, which means that we aren't enforced to follow it in the process of building our web services. It only provides some kind of guide with its recommendations. REST architectural style identifies constraints, such as the uniform interface, that with Web service provides desirable properties, such as performance, scalability and modifiability, which are giving possibility services to work well on the Web. For the REST architectural style, data and its functionality are discussed as resources, that are links on the web, which can be accessed using Uniform

Resource Identifiers. The resources come in action by using a set of easy, well-defined operations. The REST architectural style is a client-server approach which is designed to use a stateless communication protocol, commonly HTTP. In its architecture clients and servers exchange representations of resources by the standardized interface and protocol. REST Server simply develops access to resources and REST client accesses them. It uses different type of representations for the resources such as text, JavaScript Object Notation (JSON) and XML. For the last years JSON is the most popular format to be used in Web services. All these principles make Restful Web Applications simple and lightweight. More precisely the REST architectural style contains six constraints:

- **Uniform Interface:** This constraint specifies a significant interface between clients and servers. It's some kind of contract for communication through clients and server. They represent short rules to return components in the generic way. It makes easier presentation of architecture in separation way of the whole system in order to evolve them independently. There are four guiding principles of the Uniform Interface: Resource-Based, Manipulation of Resources Through Representations, Self-descriptive Messages, Hypermedia as the Engine of Application State. **Resource Based-** Individual resources are defined in requests using Uniform Resource Identifier (URI)s which are resource identifiers. The resources are decoupled from the representations that are given to the client. The idea is that server doesn't send as a response its database, just some JSON or XML that are one of the way for database record expressing. **Manipulation of Resources Through Representations-** The client or receiver of resource representation has enough information to modify or delete the resources on the server with proper permitted permission. **Self-descriptive Messages-** Any message includes good enough information to find out how to process them and each responses explicitly indicate their cache-ability. **Hypermedia as the Engine of Application State:** Clients deliver state from the server through the body contents, query-string parameters, request headers and the requested URIs which contain resource name. This is technically recognized as hyperlinks within hypertext. In general a resource is a data, identified in database

storage. Resource state, on the other hand is constant across every client who makes a request about this particular resource.

- **Stateless:** Idea of statelessness represents that the necessary state to capture the request is bind within the request itself, even if as part of the URI, query-string parameters, body or headers. When the server finishes it's processing the proper state, or the pieces of state are getting into contact back to the client trough headers, status and response body. Statelessness enables greater scalability while the server doesn't provide processes to maintain, update or communicate that session state. Basically state or application state is that point where server cares about to fulfill a request data necessary for the current session or request. Here client has the opportunity to send multiple requests to the server, which must be independent from each other and in every request can be included the necessary information. After that server starts properly defined processes with them.

- **Cacheable:** As on the World Wide Web, clients has opportunity to cache responses which must implicitly or explicitly provide themselves as cacheable or not in order to prevent clients reusing stale or inexpediency data in response for further requests. Well-defined caching partially or completely spin-offs some client–server interactions, further improving scalability and performance. More specifically many clients get access to the same server and request the same resources. That is way are useful these responses to be cached, in order to avoid unnecessary processing and increase quality of performance.

- **Client-Server:** The uniform interface decouples clients from servers. This separation of concerns means that servers are not interested with the user interface or its state, so these servers can be ordinary and more scalable. Servers and clients may also be developed independently, as long as the interface is not changed. This is the basic restriction of a REST application, implementing separation of architecture and responsibilities on client and server side. In the end this approach gives two independent evolution of architecture. Figure 2.5 shows that client requests and

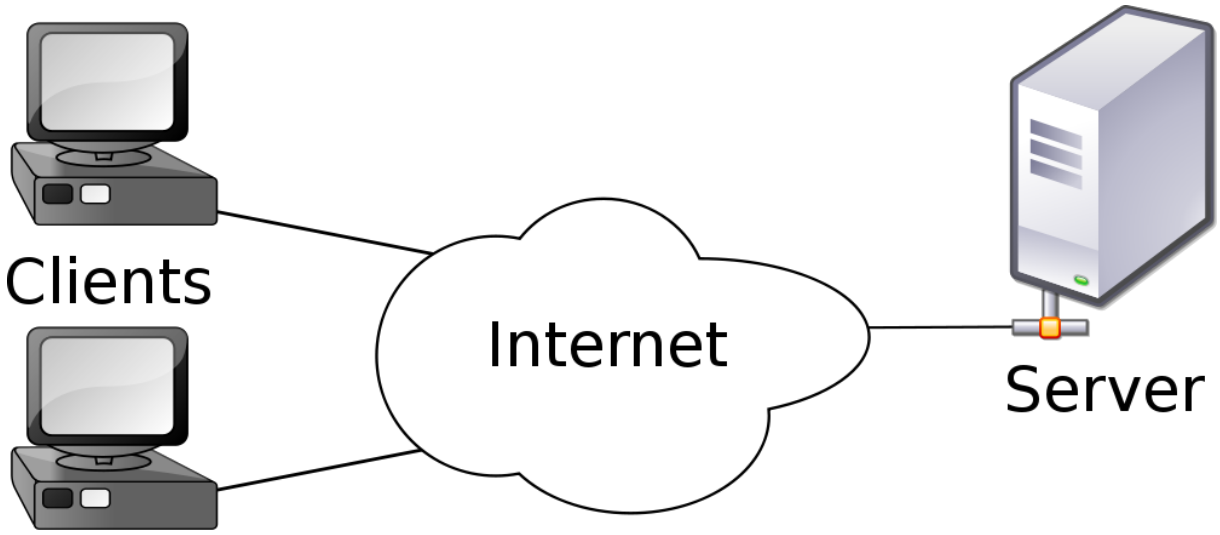


Figure 2.5: Client-Server constraint.
http://2016karma.jimdo.com/2016/02/28/counter-strike-source-gcf-download/

server responses trough the internet.

- **Layered System:** A client can't usually tell whether it's connected directly to the end server or trough the intermediate opportunity. Intermediary servers are improving system scalability by providing shared caches. Layers can also enforce security policies. It means that our application must contain layers which should be changeable when we need to add or delete other layers. The principle of this approach is that the customer couldn't directly call the application server. Customer needs to take care of communication trough the intermediary which is responsible for distributing the requests on servers. In the case of some changes, it's more flexible to change intermediary.

- **Code on Demand:** Servers can temporarily extend or customize the functionality of a client. Submission with constraints and thus conforming to the REST architectural style, will make an opportunity about different kind of distributed hypermedia system to have desirable properties, such as performance, scalability, visibility, portability and reliability.

### 2.2.1 HTTP Methods

Related to the recourse on the web server there are only a few basic things which can be done with them. The HTTP functions represent a major portion of uniform interface constraint. The primary and most commonly used HTTP functions are POST, GET, PUT, and DELETE. Their combination is known as Create, Read, Update, Delete operations (CRUD). The HTTP GET method is used to read a representation of a resource on the web server. In its path GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it mostly returns a 404 (NOT FOUND) or 400 (BAD REQUEST). According to the design structure of the HTTP specification, GET requests are used only to read data and don't make changes in it. Furthermore, this way of retrieving data is recognized as a safe. GET method can be called without under of data modification risk. After its calling process as a result there is always same effect. Beside that, GET method is idempotent, which means that making multiple similar requests will have the same result as a single request. It's important not to expose unsafe operations trough GET function. There is anything in its responsibility to modify any resources on the server. PUT function is frequently used for update capabilities. It's a process of putting to a known resource URI with the request body including the newly updated representation of the original resource. PUT function also can be used to create a resource where its ID will be chosen by the client and not by the server. On successful update, PUT function returns 200 or 204. If it uses return HTTP status 201 is a successful creation. PUT is not a safe operation. It modifies or creates state on the server, which means that in process of making the same call again, the resource is still there and still has the same state as it was with the first call. The POST method is more usable for creation of new resources. In particular cases, it's used to create subordinate resources which are considered to create a new resource. POST does non-idempotent resource requests. DELETE is pretty easy to understand. This functionality is used to delete a resource identified by an URI. On successful deletion, return HTTP status is 200 along with a response body. DELETE operations are known as an idempotent. To

DELETE a resource is associated to remove it. Constantly calling DELETE on that resource gives the same result, it just disappears. After second time Calling it will often return a 404 (NOT FOUND) since it was already removed, which makes DELETE operations no longer idempotent, but is a proper compromise if resources are removed from the database instead of being just simply marked as deleted.

### 2.2.2 Resource Naming

To initialize the HTTP functions and make resource naming properly is one of the most important concept to mention when creating an understandable Web Application Programming Interface (API). The resources need to be named clearly. In essence the RESTful API represents simple collection of URIs. HTTP calls to those URIs, where as a response come JSON or XML representations of resources, that will contain relational links. Each resource on the web server has its own address and every resource in a service will have even one identifying URI. It's simple to find out which are looking for, when URI makes sense and appropriately describes the resource. URIs should follow a predictable structure to increase understandability and usability as well. In additionally RESTful APIs are written for consumers. The name and structure of URIs should be meaningful to those consumers. It's often difficult to know what the data boundaries should be, but with understanding of data there is guaranty of representation from the clients. Design needs to be for the clients not for only data.

### 2.2.3 Representation

RESTful web services have opportunity to support multiple representations of resources, including JSON and XML, as well as wrapped JSON and XML. As the default representation there is JSON, but services should allow clients to specify alternative representations as well. For a client to request a representation format, there is a question about header file extension style format specifier, query string parameter and so on. In best cases,

services would support all of those methods. According recommendation the use of representation file extensions should be such as '.json', '.xml' and wrapped options, '.wjson' and '.wxml'. When the talk goes about REST services, XML is largely misplaced. Simply anyone uses XML with REST and also it's a recommendation which standards and conventions are not competitive action.

### 2.2.4 Unique of Rest

In general Representational State Transfer is a new architecture for web services which nowadays has an important influence on the industry. Many of new services from large companies such as Google depends on REST as the useful technology to share information from the multiple sources. REST isn't a standard, just architectural style of network based systems which main concepts are clients and servers. Unique of this kind of web service is usability. Practically process with RESTful web services starts when clients initialize requests to servers. After that server processes request and returns proper responses to clients. Requests and responses are concentrated around resources. A resource can be actually any clear and understandable concept which is addressed. Representation of a resource is a normally document which represents the current or requested state of a resource. The main motivation to use REST approach is to do things easier. It demands on hypertext, which by itself has good enough opportunities to make communication over the internet. HTTP is a power of network because of its valuable properties such as verbs, URIs, request and response headers. What makes characteristic and preferred technology REST is that it stands on protocols and standards which power the Internet and is simpler than other traditional Web services. REST server is not demand on some other fixed APIs to change resources. It makes huge difference with Remote Procedure Call (RPC) schemes where client and server are enforced to be agreed upon a detailed protocol which normally needs compilation process into both ends. Preferring REST means that will be in use HTTP calls that will be message based and depend on the HTTP standard to describe all the messages. Information produced by REST are existing independently

from the technologies that make easier production. All the opportunity that it gives such as performance, flexibility, easy of use, clean, secure, extensible, maintainable are aimed to make rapid implementation of our APIs in the organizations, in order to keep safe our employers and clients from the online swindling. [11] [15] [1] [17]

## 2.3 Apache Shiro Characteristic

To manage the secure authentication and authorization process integrated with LDAP server we have used Apache Shiro framework, suggested by Apache Software Foundation, which is an American non-profit corporation to produce Apache software projects. History starts in 2004 year when these two person Les Hazlewood and Jeremy Haile could not find a corresponding Java security framework that could have an affected influence at the application level. Between 2004 and 2008 this new technology was hosted on SourceForge, which represents a web based service to control free and open-source software projects. Apache Shiro, as an easy to use product was produced by the Apache Software Foundation. Its useful solution in security issues related with session API which is unique in the world of security framework. It's usable in every Java Virtual Machine (JVM) based applications and in each architectural tier. Shiro's architecture allows for pluggable session data stores, which means just only once configuration and it's done for the other uses also. There is no necessary reconfiguration according different deployment of the application. Essentially Apache Shiro represents a powerful Java security framework, built as a solution for developers to easily integrate security features such as authentication, authorization, cryptography and session management. Before realizing final version of Apache Shiro, first was originally developed as JSecurity in 2004. JSecurity was submitted to the Apache Software Foundation in 2008 and renamed Ki (pronounced key) due to trademark concerns. After this comes history of the versions: Apache Shiro 1.0 was released in July 2010, Version 1.1 in November 2010, Version 1.2 in January 2012 and the current version is 1.2.1 which was released in July 2012. As a main objective of this framework is to reduce the complexity regarding to the management of applications. It can be used to secure of

any JVM based application from the command line applications to the largest web and enterprise APIs. In architectural view Shiro has three main concepts: Subject, Security Manager and Realms. As a Subject is considered the currently existing user, the thing which at the present time interacts with the software. Once when we will get the Subject, we have access to almost everything we had want to do with Shiro for the current user. Manage the operations such as login, logout, access their sessions, execute authorization checks in order to see if the current user is allowed to perform the requested action or not. Security Manager of Apache Shiro manages security operations for all users. It's heart of Shiro's architecture, known as an Umbrella object, that references lot of internally nested security components which are forming an object graph. The default implementations of Security Manager are Plain Old Java Object (POJO)s and they are configurable with any POJO-appropriate configuration technique. In general Plain Old Java Object doesn't bound any special restriction and not required any class path. Final core concept Realm behaves as bridge or connector between Shiro and application security data. It represents synonym of Database Access Object (DAO)s which encapsulate connection details for secure data sources and make them available to Shiro as needed. In abstract way to show how authentication process is done with Shiro, there are eventually three steps:

- Collect the user's identifying information, called principals and supporting proof of identity, called credentials.
- Submit the principals and credentials to the system.
- If the submitted credentials match what the system expects for that user identity (principal), then will be considered successful authentication. If they don't match, the user is not considered to authenticate.

Commonly Authentication is the process of identity verification. In the application is necessary to prove a user is who says she or he is. The user needs to provide some kind of proof of identity that system understands and trusts. In Authentication part the main concepts are the following:

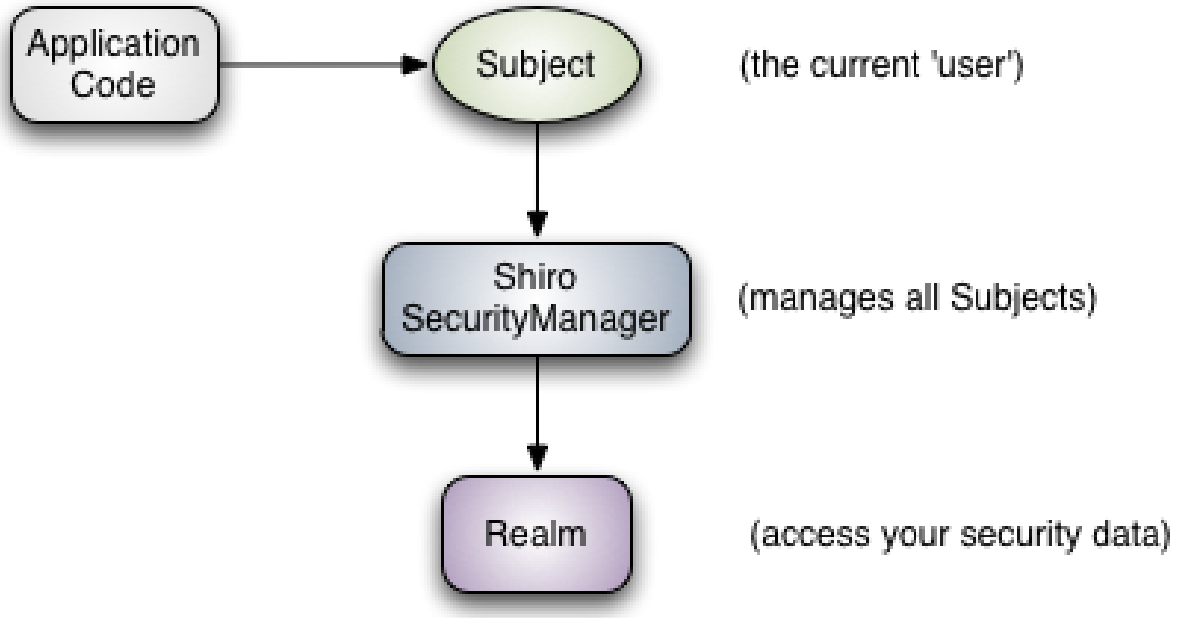


Figure 2.6: Apache Shiro components.
`http://shiro.apache.org/architecture.html`

- **Subject** - Security specific user 'view' of an application user.
- **Principals** - A subjects identifying attributes.
- **Credentials** - secret data that are used to verify identities.
- **Realms** - Security specific DAO, software component that talks to a backend data source.

Figure 2.6 shows Apache Shiro main components structure. Authorization process with Apache Shiro is controlling access process, what can be done in the application by users. Its function is to specify access rights to resources, identify who can have access and on what functionality. Users are available to perform actions according defined concepts for them, such as Roles and Permissions. User is allowed to do something or not based on what Roles and/or Permissions are assigned to them. After that, application can make a control what functionality is exposed which is established on checks for these Roles and Permissions. The Subject API gives opportunity to perform these built rules.

### 2.3.1 Permission

In particularly Permissions are the security policies and statements of functionality. They define what can be done in the application. A well formed Permission describes resource types and what actions can be performed and are possible when will have an interaction with those resources. Most commonly used actions for data related resources are CRUD. It's necessary to understand that Permissions don't hold information about a current interacting user, who can perform the actions, they are only statements of what actions can be performed. In Apache Shiro there are Permission levels in order of granularity:

- **Resource Leve**l - This is very wide and easiest to build. A user can edit records. The resource is specified but not a specific instance of that resource.
- **Instance Level** - Initializes the instance of a resource.
- **Attribute Level** - The permission now identifies an attribute of an instance or resource.

### 2.3.2 Roles

In the Authorization process, Roles represent collection of Permissions, that are successfully used to simplify the management of users which can be assigned Roles instead of being assigned Permissions directly, that can get complicated in relationship of huge user databases and more complex applications. There are two types of Roles: implicit and explicit. **Implicit Roles-** are what will be defined implicitly where application implies a set of permissions. Particularly, user has a private Role as opposed to the Role explicitly being assigned Permissions or application checking for them. Role checks if there is the reflection of an implicit Role. **Explicit Roles-** An explicit role has Permissions explicitly assigned. It develops an explicit collection of Permissions which check in the code reflections of explicit Roles. User is able to perform these actions, not because of some implicit Role name based on a string, but according corresponding Permission which was explicitly assigned to user Role. The big benefits of explicit roles are simple manageability and

lower maintenance of the application. In Apache Shiro technology is able to dynamically add, remove, or change Roles at runtime and authorization checks will always have up to date values. There is no necessity to force users to logout and log back in order to get their new Permissions. [13] [12] [14]

## 2.4 Oracle Distribution

The main database engine of IPB is an Oracle. In generally an Oracle database represents collection of data. The objective of database is to store and retrieve connected information. A database server is the key in order to solve problems related an information management. Any server regularly manages a large number of data in a multiuser environment which means that huge amount of users can access the same data. All this can be improved while delivering high performance. A database server also detects unauthorized access and represents responsible solutions for failure renewal. Oracle as a world's leading supplier of software for information management is known for its sophisticated relational database products, which are used in many corporations or largest Web sites. Oracle relational database was the first in the world to support the Structured Query Language (SQL). Oracle targets high end workstations and minicomputers as the server platforms in order to run database systems. It has been a champion of network computers for a long time. It's the world's first software company which is developing and providing Internet enabled enterprise software such as enterprise business applications, decision support tools, application development, database, server. Oracle database as a Relational Database Management System (RDBMS) from the Oracle Corporation originally was developed in 1977 and till nowadays its reputation is high quality and represents one of the most trustful and broad in scope used relational database engine. Oracle is an entirely scalable relational database architecture and is mainly used by global enterprises, which deal and process data over wide and local networks. The Oracle database has its own network section to allow communications over networks. It runs on most commonly used operation system platforms such as Windows, UNIX, Linux and Mac OS. A major feature

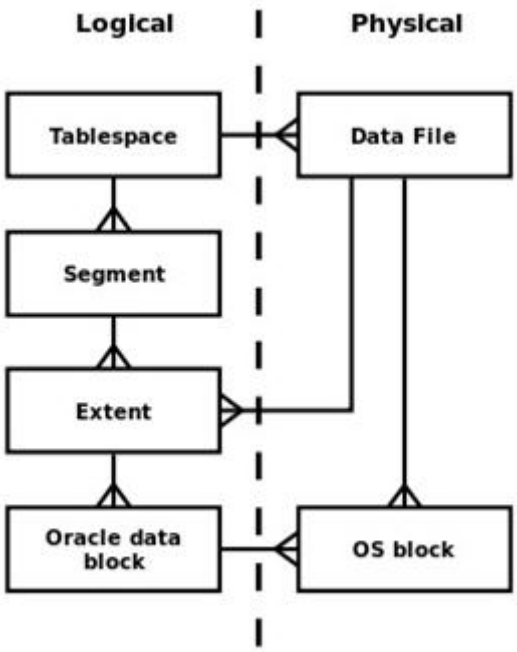


Figure 2.7: Logical and Physical Storage.
https://docs.oracle.com/cd/E11882_01/server.112/e40540/logical.htm#CNCPT301

of Oracle database is that its architecture is divided into logical and physical parts, which means that for huge distributed computing the data location is inappropriate to the user, give opportunities for a modular physical structure without affecting the activity of the database. These separated parts gives chance to manage physical storage of data without affecting to logical structures. In general the logical part of database are data blocks, extents, segments and table spaces. At a physical level, the data are located in data files on disk which is allocated in operating system blocks. Figure 2.7 shows Logical and Physical Storage entities. [16]

## 2.5 AngularJS and Bootstrap interpretation

All the RESTful Web Applications working around the academic service database use AngularJS and Bootstrap User Interface. AngularJS extends Hyper Text Markup Language (HTML) with its new attributes. HTML is useful for declaring static documents,

but it loses its strength when we are declaring dynamic views in web applications. In order to solve this problem appears Single Page Web Application which is an AngularJS. It's a structural framework for dynamic web apps. It allows to use HTML as a template language and lets extend HTML's syntax to express our application's components in such a way that is easy to understand. Its data binding and dependency injection opportunities takes away some of the code that we have to write at the given time. It can be an ideal partner with any server technology. The characteristic features of AngularJs are the following:

- Robust JavaScript based development framework which provides rich Internet Application.
- Good enough possibility to write client side application using JavaScript in a clean MVC way.
- Applications implemented in AngularJS are cross browser compliant. It automatically handles JavaScript code appropriate for each browser.
- AngularJS is open source, entirely free framework which is used by thousands of developers around the world. It has official permission under the Apache License version 2.0.

Core concepts of AngularJS which are commonly used are the following: **Data Binding** -The automatic synchronization of data in the middle of model and view components. **Scope** -Objects that indicate to the model. They behave as a glue between controller and view. **Controller** -JavaScript functions which are captured in a particular scope. **Services** -Single objects that are instantiated only once in application. **Filters** -Select a subset of units from an array and returns a new array. **Directives** -Markers on Document Object Model (DOM) elements such as elements or attributes. They can be used to define custom HTML tags that serve as new, custom widgets. **Templates** -The view with information from the controller and model which can be a single file or multiple views. **Routing** -The concept of switching views. **Model View Whatever** -Design pattern for

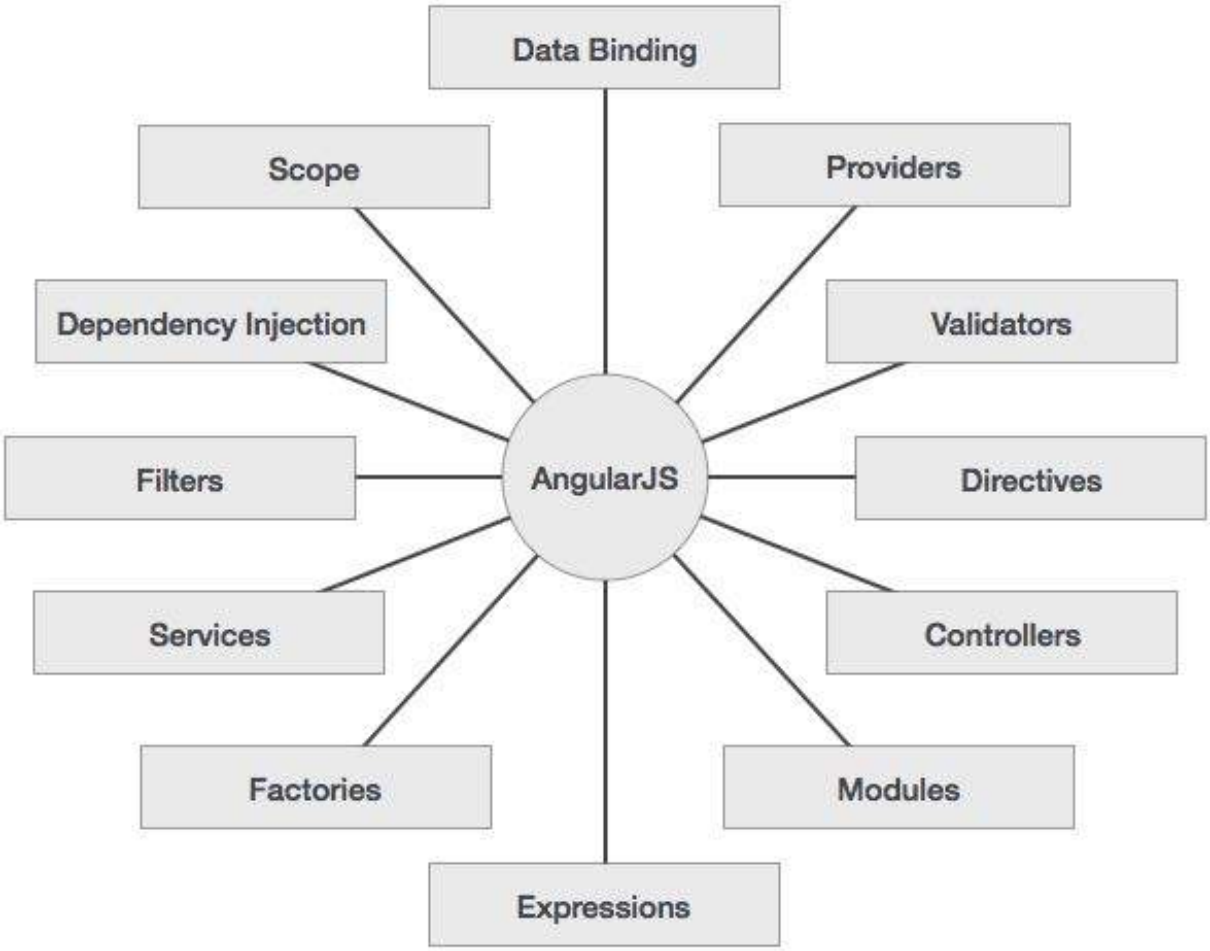


Figure 2.8: AngularJS components.
http://www.tutorialspoint.com/angularjs/angularjs_overview.htm

splitting an application into different pieces such as Model, View and Controller. **Deep Linking** -Encoding process the state of application in the Uniform Resource Locator (URL). **Dependency Injection** -Easy way to develop, understand and test application. Figure 2.8 shows AngularJS components. AngularJs main architecture is based on Model,View,Controller concept which is software design pattern for developing Web Applications. **Model** -Lowest level of the pattern responsible for maintaining data. The model is responsible for managing application data. It makes instructions from controller to update itself. **View** -It is responsible for displaying all or a portion of the data to the user. **Controller** -It's a software code that controls the interactions between the Model

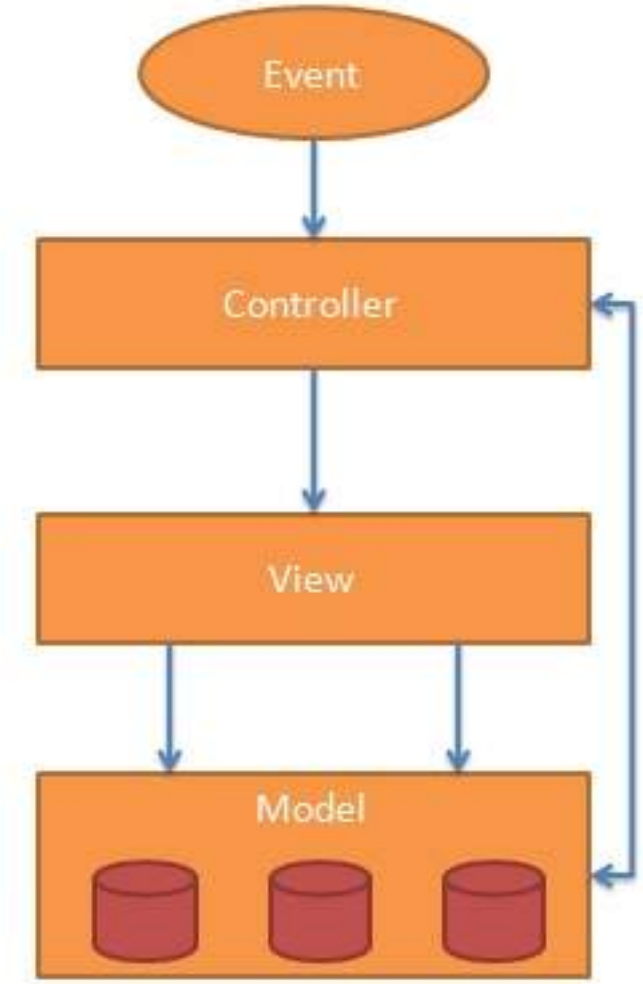


Figure 2.9: MVC abstraction.
http://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm

and View. It responds user input and fulfills interactions on the data model objects. The controller receives input, validates it and then makes in action business operations which modify the data model. In general Model View Controller architecture (MVC) separates the application logic from the user interface layer and supports isolation of concerns. The controller retrieves requests and then works with the model to prepare data which is necessary for the view. Then the view uses the data arranged by the controller to produce a final response representation. Figure 2.9 shows MVC abstraction. **Bootstrap** is the

commonly used and popular HTML, Cascading Style Sheets (CSS) and JavaScript framework for easy and fast creating responsive, mobile-first Web Applications. Bootstrap was developed by Mark Otto and Jacob Thornton and it was released as an open source in August 2011 on GitHub. Bootstrap produces mobile first styles throughout the whole library. It has support from all the commonly used web browser. Only basic knowledge of CSS and HTML makes easy to get started with Bootstrap. It provides an unique solution for building user interface. It's rich with nice and useful built in components that are flexible for customizing. Its basic structure is based on Grid System. In graphical design a Grid System is mostly related with two dimensional structure, crisscrossing vertical and horizontal lines in order to structure content. It's one of the best solution to build layout and content concepts in print design. Its easy to use methods with CSS and HTML clear consistent layout. In general, grids in web design field arrange structure of content which helps to make easy scan and reduce the conscious mental processes load on users. Bootstrap uses the main settings of CSS and also basic HTML elements. It obtains a great amount of reusable components which are created to produce navigation, alerts and so on. Related with plugins, Bootstrap contains a lot of JQuery custom plugins. In order to customize its components just needed LESS variables and JQuery plugins. It isn't necessary to set up our own environment to start working with Bootstrap. It has its own online environment where we are free to execute and test tasks. It really needs to mention that using Bootstrap we are designing beautiful web application where we don't have to write many CSS codes. [2] [3] [4]

## 2.6 Spring Boot Framework

As a powerful solution for building RESTful web application appeared Spring Boot technology. It enables to create Java based applications that are Stand Alone, production grade which just need to be run. Commonly it intends easy deployment of services with minimum effort. Spring Boot takes a strongly view expression of its platform for new and existing users quickly get to the bits that they need. It enables to create Java applications

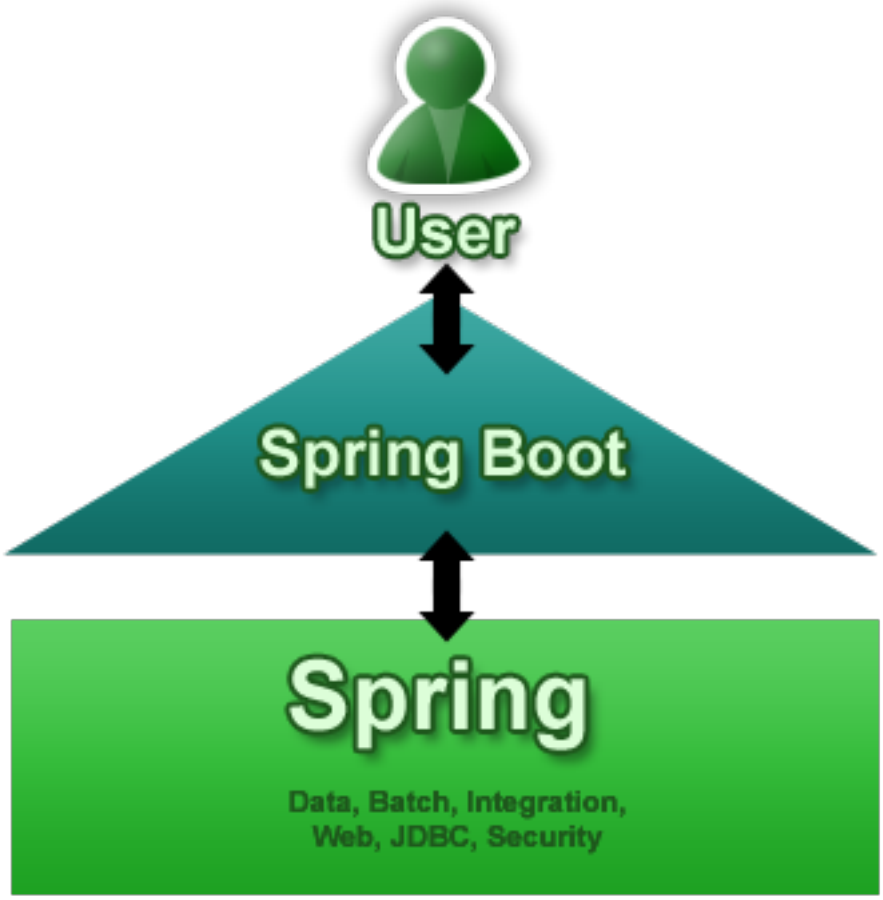


Figure 2.10: Surface field for users to extract value from the rest of Spring. `https://spring.io/blog/2013/08/06/spring-boot-simplifying-spring-for-everyone`

that can be started using java jar or more traditional Web Application Archive (WAR) deployments. Figure 2.10 shows Spring Boot as a point of focus on the larger Spring system. Spring Boot Framework mainly serves the following:

- Develop an effectively faster and widely accessible "getting started" experience for all Spring development.
- To be biased out of the box, but get out of the way quickly as requirements start to sever from the defaults.

- Make available a range of non functional features that are common to large classes of projects
- No code generation and no must have to requirement for XML configuration.

Spring Boot accomplish tasks with a small command line application that can be used to run Spring scripts which are written in Groovy. It's familiar Java syntax, without so huge code. Its'n necessary to use the command line tool or write Groovy code to get the benefits of Spring Boot. In the advantages list of this technology comes the first class Java support. It also includes convenient features which are needed in the process of pushing an application into production. It is also possible to provide web endpoints automatically that can be used to monitor application state, develop basic metrics or use to analyze production issues. By default, Spring Boot 1.4.0.BUILD-SNAPSHOT needs Java 7 support. Explicit build support is developed for Maven (3.2+) and Gradle (1.12+). List of embedded servlet containers which are supported out of the box: Spring Boot is familiar

Table 2.1: Table of Servlet Containers

| Servlets with Java version | | |
|---|---|---|
| Name | Servlet Version | Java Version |
| Tomcat 8 | 3.1 | Java 7+ |
| Tomcat 7 | 3.0 | Java 6+ |
| Jetty 9 | 3.1 | Java 7+ |
| Jetty 8 | 3.0 | Java 6+ |
| Undertow 1.1 | 3.1 | Java 7+ |

to be used with classic Java development tools or installed as a command line tool. There is just requirement of Java Software development kit (SDK) v1.6 or higher. It can be used in the same way as any standard Java library. Just simply include the appropriate spring-boot-*.jar files on proper class path. It does not require any special tools integration, so there is free choice about Integrated Development Environment (IDE) or text editor. Its applications are possible to run and debug as any other Java program. In order to copy Spring Boot jars it is recommended to use the build tool supporting dependency management such as Maven or Gradle. [18] For building RESTful web application using

Spring Boot there are other technologies that are supported and were used for teaching duties application: NPM, RequireJS, Bower, Grunt, Mybatis.

### 2.6.1 NPM

This technology in teaching duties project has been used as a package manager for JavaScript which aims to find, share or reuse packages of code from thousands of developers and bound them in powerful new ways. So NPM makes easy for javascript programmers to share the code that was created to solve particular problem and reuse them around other applications. When there is dependency to this shared code there is also possible to check about updates and download them which approves its high level usability. It's one of best reusable package module. NPM provides huge database of registers that are packages shared by the people. When there is desire to share some own code, NPM client enables to publish code up to the register and when there is an entry point trough register, other people can use also their NPM client to install packages from the register. The entry in register for this package is also reflected to the website. As a result NPM is one of the best opportunity to use codes from others, share your code with others also and make easy management of their different versions. [5] Eventually NPM can be represented as:

- Package manager for Node.js
- Public collection of packages of open-source code for front-end web, mobile, server-side applications.
- Helpful for millions of developers around the world to share/reuse JavaScript code.

### 2.6.2 RequireJS

For the implemented Web Application, RequireJS represents JavaScript file and module loader which is optimized for in-browser use, but it can be used in other JavaScript environments as well, like Rhino and Node. Using this modular script loader improves

the speed and quality of the code. RequireJS does a different approach to script loading than traditional script tags. While it can also run fast and optimize well, the main goal goes on encouragement of modular code. As part of that, it encourages to use module IDs instead of URLs for script tags. It loads all the codes related to a base URL which is set to the same directory as the script used in a data-main attribute for the top level script to load for a page. The require.js file will check this value to start script loading. Base URL also can be set manually trough the RequireJS config. It also assumes by default that all dependencies are scripts, which means that it does not expect to see a trailing ".js" suffix on module IDs. RequireJS will automatically add it in the process of translating the module ID to a path. With the paths config it is possible to set up locations of a group of scripts. All of these capabilities allows to use smaller strings for scripts as compared to traditional script tags. [6]

### 2.6.3 Bower

To build Web sites there is lot of things to manage — frameworks, libraries, assets, utilities. Bower undertakes all these things to make work easier and it's the part where was used this technology in our new project. In its possibility are considered components that contain HTML, CSS, JavaScript, fonts or even image files. Bower doesn't connect or minimize codes, it just installs the right versions of the packages and their dependencies that are needed. To get started with Bower it's necessary to mention its way of working. This technology does the work by fetching and installing packages from all over, taking care of hunting, finding, downloading and saving all the stuff that are looking for. Bower keeps all of these packages in a manifest file, bower.json. How the packages will be used it's up to developer. Bower develops hooks to make easier and less difficult using packages in our tools or workflows. It's optimized for the front-end. In case of using multiple packages Bower will download it just once. This is known as a flat dependency graph which helps to reduce page load. [7]

### 2.6.4 Grunt

Grunt represents Javascript task runner which serves automation of the processes. The less work is required in process of performing repetitive tasks like minification, compilation, unit testing, linting. In order to achieve all these results we used this system in our project. It really made job easier for us. After finishing its configuration part through a Gruntfile, a task runner can do most of work with none of an effort from our side. The Grunt environment is increasing and growing every day. With literally hundreds of plugins to choose from, we can use Grunt to automate just about anything with a less of effort. Grunt and its plugins are installed trough NPM. It requires stable Node.js file which is considered unstable development version. Before setting up Grunt ensure that your NPM is up-to-date by running: npm update -g npm command. [8]

### 2.6.5 Maven

In general Maven can appear to be many things, but in a nutshell for the new developed application Maven performs a struggle to apply patterns to a project's build infrastructure to encourage comprehension and productivity by providing a well defined path in the use of best practices. Maven develops a project management and completely understandable tool. It helps to manage the following: Builds, Documentation, Reporting, Dependencies, Releases Distribution. Maven has the ability to provide benefits for our build process by employing standard practices to hasten our development cycle while at the same time giving us hand to achieve a higher rate of success. Maven configuration happens at three levels:

- Project - Most static configuration occurs in pom.xml
- Installation - This is the configuration which added once for a Maven installation.
- User - It's the specific configuration to a particular user.

The separation is quite understandable. Project defines information that applies to it, no matter who is building. We should have our projects inherit from a company-wide

parent pom.xml. We can specify our user configuration in user.home/.m2/settings.xml. It is possible also to do a full reference to the configuration file. [9]

### 2.6.6 Mybatis

MyBatis represents first class persistence framework with support for custom SQL, stored procedures and advanced level mappings. It gets rid of almost all of the JDBC code and manual setting of parameters and retrieval of results. For us this technique was used to have integration with database. MyBatis can also use simple XML or annotations for configuration, map interfaces and Java POJOs to database records. To use MyBatis we just need to include the mybatis-x.x.x.jar file in the classpath. If we are using Maven it is necessary to add proper dependency to pom.xml. Each MyBatis application centers around an instance of SqlSessionFactory. A SqlSessionFactory instance can be captured by using the SqlSessionFactoryBuilder. SqlSessionFactoryBuilder can build a SqlSessionFactory instance from a XML configuration file or a custom arranged instance of the Configuration class. Building a SqlSessionFactory instance from a XML file is very simple. It is recommended to use a class path resource for this configuration, but we could use any Input Stream instance, including one created from a literal file path or a file:// URL. MyBatis contains an utility class, named as Resources, that includes numerous methods that make easier to load resources from the class path and the other locations. [10]

# Chapter 3

# Implementation

## 3.1 Creating Tables and Relationship Between them

Our deal with Oracle database started in the process of creating tables. There are three tables which are new in already existing list: COORDINATOR, CUNIT_DEPARTMENTS, CUNIT_TEACHERS. First table gives us all the user who is able to work with the application. Second is used to save records about assignments of curricular units to the teachers and third table was created to push all the curricular units which are assigned to the departments. Properly performed adding or removing operations make influence on the last two tables. As for relationship, each department has an only one responsible coordinator, so it tells about one to one relationship between Coordinator and Department tables. CUNIT_TEACHERS table was created because of many to many relation between curricular units and teachers tables. CUNIT_DEPARTMENTS is also the result of many to many connection between curricular units and departments tables. We are assigning different curricular units to the variety of teachers or departments that is why we have these many to many relations. Flexibility of using data from database is achieved by Mybatis integration with Spring Boot Framework. All the queries are written in an appropriate xmls. In GeneratorConfig.xml file we just made connection with Oracle database, specified tables and got easy interaction with the data. Each queries from database were

bounded in mapper.xml files properly constructed in Spring Boot Framework.

## 3.2 User Interface Configuration with AngularJS and Bootstrap

For Assignments and Curricular Units user interface modules we have gotten three main parts such as Controller, Model and View implemented with AngularJS and Bootsrtap components. These three parts are defined for both of modules: Assignments and Curricular Units. All the fields on Assignments module are represented by bootstrap comboboxes. The records are described as a table. We are handling each input event using a controller which interacts with the module and changes the view as users require it from the beginning. For the assignment module we implemented two controllers: AssignmentsCtrl.js and PrintCtrl.js. In the first controller we are catching all the input data by the watching statements such as "school" or "department". Scope for getting assignments according teacher code value uses this path: /academic/assignment/:teacherCode and returns all the appropriate records using curricularUnits scope. The second Print controller is used to display curricular units assigned to the teacher and print them. On the view side there are appropriate htmls for the controller. With bootstrap comboboxes we used div and label classes. Information about the assignments is expressed using a table. For the Print.tpl.html file we have used table layout of the contents. The final module part identifies module called Assignments with name, url, controller, templateUrl and data parameters. The same approach is done for the second implemented module named as a Curricular Units.

### 3.2.1 Graphical User Interface of Assignments Module

After the authentication and authorization process of users, there are proper modules according their rules. For the user "a34267" we defined Pivot rule, which is responsible on Assignments module with read/add/delete Permissions. The resembling method is

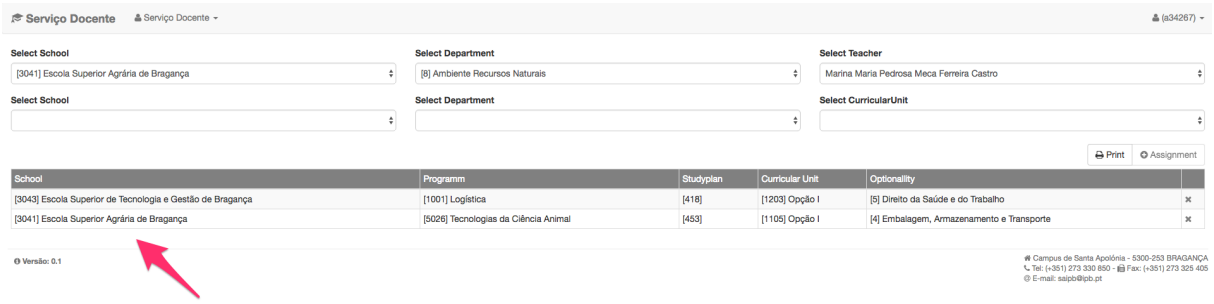


Figure 3.1: Reading data.

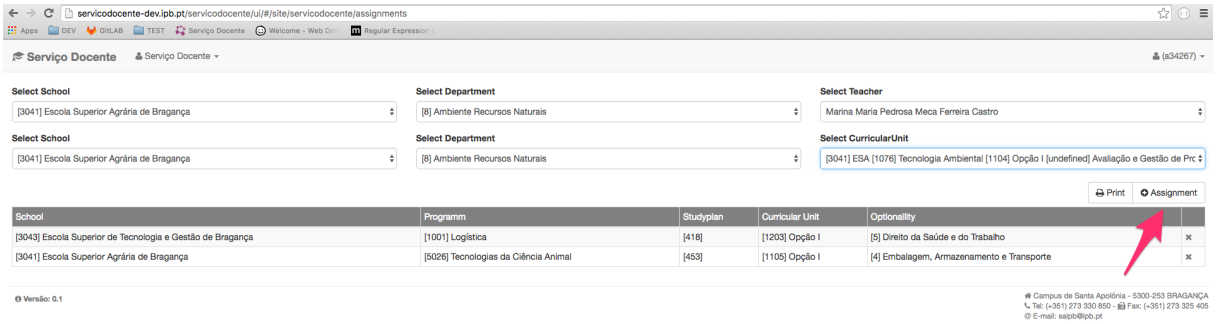


Figure 3.2: Add functionality.

used for the second Curricular Units module with the appropriate Rule and Permissions. Detail interpretation of Assignments module includes the following:

Figure 3.1 shows existing assignments in Assignments module. Figure 3.2 shows adding of new assignment in Assignments module. Figure 3.3 shows records with new assignment in Assignments module. Figure 3.4 shows an error message in response of adding same record in Assignments module. Figure 3.5 shows a dialog message box to approve or cancel a delete operation in Assignments module. Figure 3.6 shows the result of a delete operation in Assignments module.

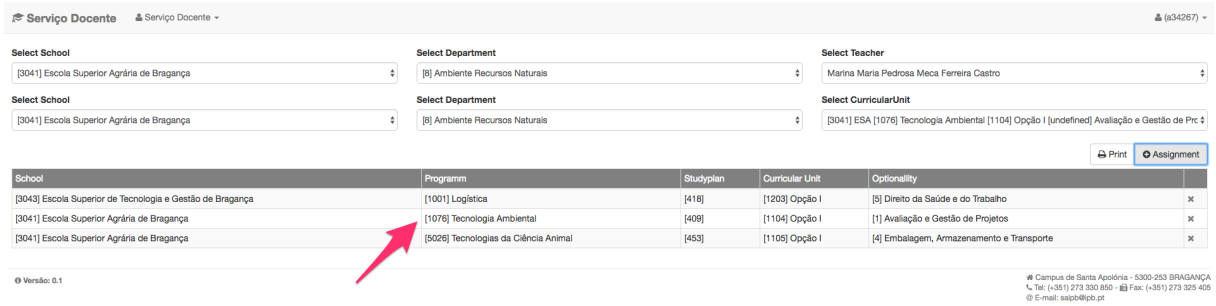


Figure 3.3: New record.

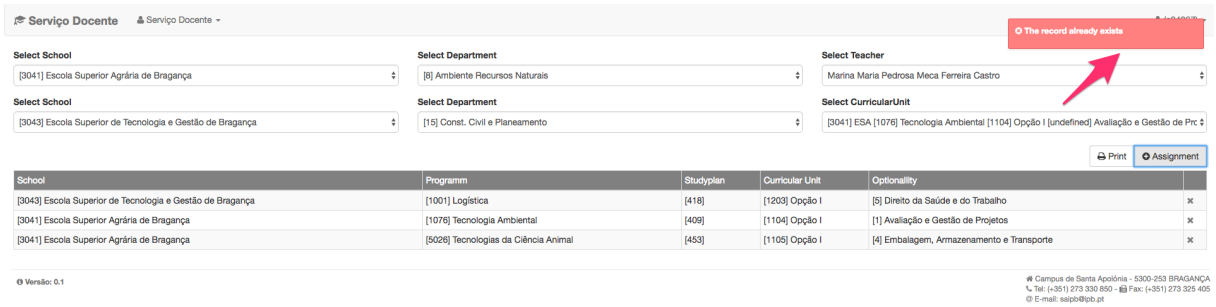


Figure 3.4: Adding same curricular unit from different school.

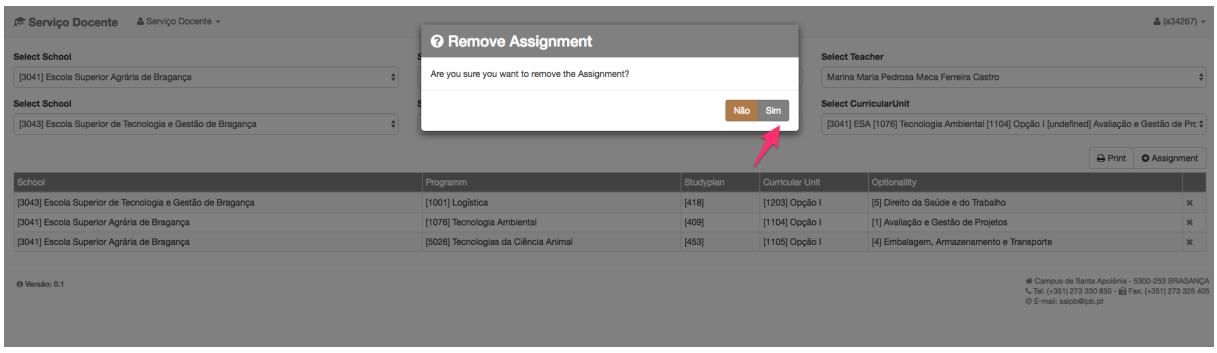


Figure 3.5: Warning before delete operation.

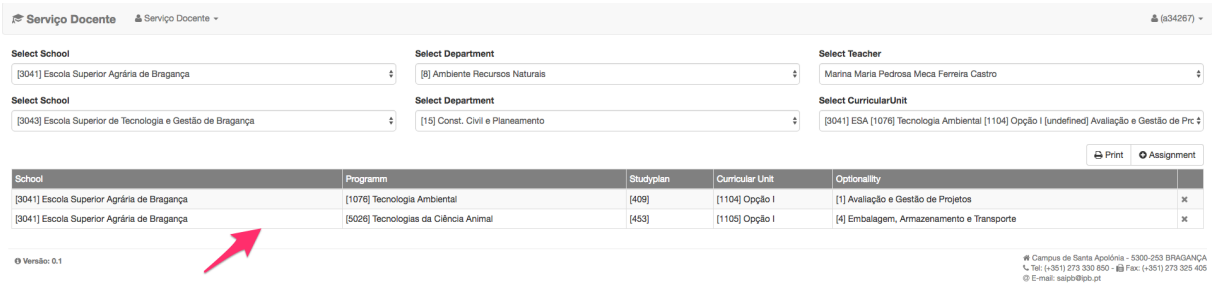


Figure 3.6: Delete operation.

## 3.3 Rest Implementation with Spring Boot Framework

One of the power of completed project is using flexible, comfortable, modern and widespread technology which is Spring Boot Framework with variety of possibilities. It's able to do entire integration with other technologies which underlines its unique character that really makes sense. All various parts of implementation help to do work easier, increases the speed of building process with quality and secure results. With these technologies, configurations are simple and suitable for the just in time made work. Of course isn't any doubt about difficulties during the implementation, which were referred to the functionality and view building process. The main issue was how to give access all the teachers from different departments to the various curricular units, by other hand how curricular units can be assigned to departments from different school. These functionalities were core realization of teaching duties project. Their implementation are similar to each other and that is why we just describe one of them. Global picture of existing situation in IPB showed that one teacher can be assigned to different curricular units from various department and one curricular unit can be given to diverse department. The building process of the algorithm for Curricular Units module was divided in several steps. First level was initialization of school and department. Then check if the some curricular unit was already assigned to the selected department according the school or not. For the second level if the result from previous condition is "yes" we can select and see this

curricular unit with additional parameters from the database. Third level is add more curricular units from different or same school and department or delete the record. For the forth level if the first condition will give "no" as a result we will make initialization of these fields: school, program, study plan, curricular unit and then will add the record. All these actions are fit to the capability of Spring Boot Framework. Selection of school in Assignments user interface module makes "Get" request from the server trough the URL:/school and returns a list of schools, which is mapped in SchoolMapper.xml file. Selecting department field also makes "Get" request from the server trough this URL:/department/schoolCode and maps result in DepartmentMapper.xml file. The third field in this module "Gets" teachers for chosen school and department according teacherCode via URL:/department/schoolCode/departmentCode/teacher and mapping process goes on CunitTeacherMapper.xml file. If the teacher is already assigned to the curricular unit we will get a view in these parameters: School, Programm, StudyPlan, Curricular Unit and Optionallity. Otherwise the result of the request will be an empty view. In order to add assignments to teachers first we need to make initialization process of fields: school, department and curricular unit, which gives opportunity to choose a curricular unit from where we wish. For adding or removing assignments to teachers we are using all these parameters: schoolCode, programmeCode, studyplanCode, Code, optionCode, teacherCode where Code is Curricular unit code. If we will try to add the same assignment it will give us notification about this using exception handling. Other technologies integrated with Spring Framework worked automatically after their installation process. Maven helped in project building and managing part. It gave us opportunity to understand completely the present situation of a development activity in the short time period. Mybatis was mediator between Spring Boot and Oracle database. Grunt runs javascript tasks and automates many processes. Bower managed components that were containing HTML and Javascript. RequireJs as a module loader made improvement of the code speed and quality. NPM traditionally appeared as a package manager for Javascript code. All together with Spring Boot Framework they made application building process easier and convinient.

## 3.4 Authentication and Authorization with Apache Shiro Framework

In the process of authentication and authorization with Apache Shiro technology, Subject for the application is human User. In authentication controller class as we get it then we collect Subject's principals and credentials using UsernamePasswordToken Class, where is represented POJOs and function overloading. UsernamePasswordToken implements HostAuthenticationToken and RememberMeAuthenticationToken classes. Both of them extend AuthenticationToken class, that gives information about principals and credentials. RememberMeAuthenticationToken class does the job for the application in order to remember users when they return. The next step in the Authentication process is to submit the token to an authentication system, that is supported by security-specific DAOs, which are idea of Realms. If the login() method call is successful, it means that the user is logged in with an associated user account. In other cases works exception handling process. The logout part is done using subject.logout(); which will close the user session. To implement part of Authorization we have created interface Permissions, where are listed all the permissions. Every interaction with the server are secure via permission checking. In user interface modules which are developed to assign curricular units to the departments and teachers we have different Roles for each modules. Every Role has an access to proper Module. According Roles users have an access of Permissions. From the database view there are several tables for authorization process, for instance such as Roles.

Table 3.1: Piece of Roles table.

| Roles description | |
|---|---|
| RoleId | Description |
| coordinator | Department Coordinator |

As a whole using the Apache Shiro technology it was achieved secure Authentication and Authorization process. The scenario of the process implementation was easy, without

complexity. Secure part of the application was gained and difficulties got rid of.

# Chapter 4

# Discussion

From the beginning we were supposed to achieve desirable assignments of curricular units for the teachers and departments. Despite of many difficulties we have gotten defined goal. In front of us was problem how to manage different assignments of curricular units for the various teachers and departments. Functionality which we have implemented was needed to match our design. We tried to avoid additional windows on the web pages in order to do work simply. First Customer just need to perform authentication and authorization process. According the rules user will be able to see the appropriate modules and with properly defined permissions will accomplish actions in the application. On the user interface customer just needs to make a selection from the comboboxes and click on one button to assign the selected assignment. For removing and printing assignments there is also only print and remove buttons. As it seems design is pretty customized. Permitted users for the application need little effort instead of manual work related with papers. This new project exploited an existing problem and appeared as a foundation for future improvements which can be other tasks with their solutions related to the GAL system. As an advantage for implemented work could be that it's the first project which is integrated to the system existing in IPB. We have used same technologies, programming languages and all other instruments to make the sense of usability. Each used technologies gave chance to build stand alone projects with high level security and integration to other systems.

# Bibliography


[1] http://www.tutorialspoint.com/soap/.

[2] https://angularjs.org/.

[3] https://en.wikipedia.org/wiki/AngularJS.

[4] http://getbootstrap.com/.

[5] https://www.npmjs.com/.

[6] http://requirejs.org/.

[7] https://bower.io/.

[8] http://gruntjs.com/.

[9] https://maven.apache.org/.

[10] http://www.mybatis.org/mybatis-3/.

[11] Todd Fredrich. Restful service best practices. Technical report, May 29, 2012.

[12] Nathan A. Good. Introducing apache shiro. September 14, 2010.

[13] Les Hazlewood. Application security with apache shiro. March 14, 2011.

[14] Christopher Lynch. Apache shiro - executive summary.

[15] John Mueller. Understanding soap and rest basics and differences. January 8, 2013.

[16] 10g Release 2 (10.2) Oracle Database Online Documentation. Introduction to the oracle database.

[17] Cesare Pautasso. Ws-* vs. restful services.

[18] Phillip Webb. Spring boot reference guide. Technical report, 2013-2016.

# Appendix A

# IntelliJ IDEA and WebStorm

JetBrains IntelliJ IDEA is a cross-platform technology which works on Windows, OS X and Linux that brings the whole range of precise developers tools, all collected together to create the convenient development environment. In order to support the Spring Framework, IntelliJ IDEA develops a set of plugins and a devoted facet type. All Spring plugins are bundled with the IDE and are enabled by default as well. The Spring facet is commonly used in combination with Hibernate. The Spring Boot support in IntelliJ IDEA contains:

- Integration of Spring Initialization which lets to create a custom Spring Boot starter project using a pre-configured project template directly from the New Project wizard.
- Advanced coding assistance for editing our application configuration which contains code completion, error highlighting, navigation and quick fixes.
- A dedicated Run/Debug configuration that allows to quickly override the Spring Boot settings.

The smartest JavaScript IDE WebStorm represents lightweight yet powerful IDE, efficiently appointed for complex client-side development and server-side development with Node.js. WebStorm helps to write code better via smart code completion, on-the-fly error

detection, powerful navigation and refactoring. The IDE develops first-class support for JavaScript, Node.js, HTML and CSS, as well as their modern successors.

# Appendix B

# Pieces of code implemented in the project

```
//bundling the username and password with Apache Shiro
public UsernamePasswordToken(String username, char[] password)
//Setting  remembered principals
public boolean isRememberMe()

//Getting currently existing user
Subject subject = SecurityUtils.getSubject();
//Authenticating currently interacting subjects with the system
subject.login(credentials);

//Removing curricular units according departments
String DEPARTMENT_CURRICULAR_UNIT_DELETE =
"department:curricular_unit:remove";

//checks ACADEMIC_SCHOOL_READ permission for the subject
@RequestMapping(method = RequestMethod.GET, value = "/school")
@RequiresPermissions
(Permissions.ACADEMIC_SCHOOL_READ)
```

```
public List<School> getSchools() {
    return schoolMapper.getSchools();
}
```

//getting teachers for the curricular units, rest implementation

```
@RequestMapping(method = RequestMethod.GET, value =
"/assignment/{teacherCode}")
@RequiresPermissions(Permissions.ACADEMIC_ASSIGNMENT_TEACHER_READ)
public List<CurricularUnitInfo>
getCurricularUnitTeachers(@PathVariable("teacherCode")
Integer teacherCode) {
return cunitTeacherMapper.getTeachersForCurricularUnit(teacherCode);
    }
```

//getting assignments for the teacher using controller

```
$scope.getAssignment = function () {
            api.get('/academic/assignment/:teacherCode', {
                teacherCode: $scope.teacher.code
            }).then(function (curricularUnits) {
                $scope.curricularUnits = curricularUnits;
            });
        };
```

// query for getting curricular units for the teacher

```
<select id="getTeachersForCurricularUnit"
resultType="pt.ipb.servicodocente.model.CurricularUnitInfo">
        SELECT
            sc.code                 "school.code",
            sc.name                 "school.name",
            sc.abbreviation         "school.abbreviation",
```

```
        u.school_code              "curricularUnit.schoolCode",
        u.programme_code           "curricularUnit.programmeCode",
        u.study_plan_code          "curricularUnit.studyPlanCode",
        u.code                     "curricularUnit.code",
        u.name                     "curricularUnit.name",

        o.school_code              "optionality.schoolCode",
        o.programme_code           "optionality.programmeCode",
        o.study_plan_code          "optionality.studyPlanCode",
        o.curricular_unit_code     "optionality.curricularUnitCode",
        o.code                     "optionality.code",
        o.name                     "optionality.name",

        p.school_code              "programm.schoolCode",
        p.code                     "programm.code",
        p.name                     "programm.name",

        s.school_code              "studyPlan.schoolCode",
        s.programme_code           "studyPlan.programme_code",
        s.code                     "studyPlan.code",
        s.branch                   "studyPlan.branch"

    FROM cunit_teachers a -- assignment

        JOIN school sc
            ON sc.code = a.school_code

        JOIN curricular_unit u
```

```
            ON a.school_code = u.school_code
                AND a.programme_code = u.programme_code
                AND a.study_plan_code = u.study_plan_code
                AND a.code = u.code

        LEFT JOIN optionality o
            ON a.school_code = o.school_code
                AND a.programme_code = o.programme_code
                AND a.study_plan_code = o.study_plan_code
                AND a.code = o.curricular_unit_code
                AND NVL(a.option_code, 0) = NVL(o.code, 0)

        JOIN study_plan s
            ON a.school_code = s.school_code
                AND a.programme_code = s.programme_code
                AND a.study_plan_code = s.code

        JOIN programm p
            ON a.school_code = p.school_code
                AND a.programme_code = p.code

    WHERE a.teacher_code = #{teacherCode}
</select>
```