Yabi - Yet Another Business Inteligence

Vitório Miguel Prieto Cilia - 40920

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Prof. Albano Alves

Prof. Lúcio Valentin

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2017-2018

Yabi - Yet Another Business Inteligence

Vitório Miguel Prieto Cilia - 40920

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação.

Trabalho orientado por:

Prof. Albano Alves

Prof. Lúcio Valentin

Esta dissertação não inclui as críticas e sugestões feitas pelo Júri.

Bragança

2017-2018

Dedicatória

(Facultativo) Dedico este trabalho a ...

Agradecimentos

(Facultativo) Agradeço a ...

Resumo

O resumo (no máximo com 250 palavras), permite a avaliação do interesse de um documento e facilita a sua identificação na pesquisa bibliográfica em bases de dados onde o documento se encontre referenciado.

É recomendável que o resumo aborde, de forma sumária:

- Objetivos principais e tema ou motivações para o trabalho;
- Metodologia usada (quando necessário para a compreensão do relatório);
- Resultados, analisados de um ponto de vista global;
- Conclusões e consequências dos resultados, e ligação aos objetivos do trabalho.

Como este modelo de relatório se dirige a trabalhos cujo foco incide, maioritariamente, no desenvolvimento de software, algumas destas componentes podem ser menos enfatizadas, e acrescentada informação sobre análise, projeto e implementação do trabalho.

O resumo não deve conter referências bibliográficas.

Palavras-chave: termos (no máximo 4), que descrevem o trabalho.

Abstract

Direct translation (maximum of 250 words) to English of the section “Resumo”.

Keywords: direct translation of “Palavras-chave”

Contents

List of Tables

List of Figures

Siglas

AJAX Asynchronous JavaScript and XML. 48

API Application Programming Interface. 7, 9, 15, 27, 28, 48, 56

CLI Command Line Interface. 26, 27

CORS Cross-Origin Resource Sharing. 48

CRUD Create Retrieve Update Delete. 17, 18, 29

CSS Cascading Style Sheet. 8, 27

DB Database. 27

GPL GNU General Public License. 20

GUI Graphical User Interface. 28

HATEOAS Hypermedia As The Engine Of Application State. 18, 61

HTML Hypertext Markup Language. 8, 9, 27

HTTP Hypertext Transfer Protocol. 28, 47, 48, 50, 52, 56, 61

IDE Integrated Development Environment. 21, 23

IETF Internet Engineering Task Force. 20

IoC Inversion of Control. 15, 16

IPB Instituto Politécnico de Bragança. 4, 5, 29, 55

IT Information Technology. 55

JAR Java ARchive. 22

JDBC Java Database Connectivity. 17, 19, 20, 49, 54

JPA Java Persistence API. 18, 41, 43

JSON Javascript Object Notation. 50

JVM Java Virtual Machine. 14

LDAP Lightweight Directory Access Protocol. 18–21, 46, 49, 55

LDIF LDAP Data Interchange Format. 21

LoC Line of Code. 23

LSP Language Server Protocol. 23, 24

LTS Long Term Support. 9

MVC Model View Controller. 15

MVVM Model View ViewlModel. 15

ORM Object-Relational Mapping. 15, 43–45

OS Operating System. 24

POJO Plain Old Java Object. 23

RCP Rich Client Platform. 21

RDBMS Relational Database Management System. 3, 16, 19–21, 52, 54

REST Represnetational State Transfer. 18

RFC Request For Comments. 20

SASS Syntactically Awesome Style Sheets. 8, 13

SQL Structured Query Language. 4–6, 20, 22, 25, 30

UI User Interface. 10

URL Universal Resource Locator. 9, 48

VM Virtual Machine. 24

XML eXtensive Markup Language. 20

Yabi Yet Another Business Inteligence. 35, 46–50, 53–55, 57

Chapter 1

Introduction

1.1 Textual Convetions

`Typewritter` text is used to reference pieces of code, parts of a system, class names and methods invocation with the class name e.g. `Authentication`, `SqlQueryController`, `DatabaseReader.runQuery`, `varchar`.

Italic `TODO:` mau escrito references message passing or class methods without a corresponding parent object, e.g. *authenticate*, *runQuery*.

[Slanted] - Italic

Emphasize

Bold idicates resource paths, e.g. **/user**, **/PermissionTrees**, **/runQuery/{id}**.

SMALL CAPS are used to denote HTTP verbs, e.g. DELETE, GET.

Sans Serif

Roman

Chapter 2

Context

TODO: Isso parece mais como uma introducao As companies and institutions develop, they tend to implement and depend on digital systems [1]. Generally speaking, solutions involve the deployment of a information center such as a relational database or a directory service, Relational Database Management System (RDBMS) being the most common [2]. Once the information center is made available, it is frequently updated with new information that, if not properly processed and made available, does not generate any meaningful insight.

It is not a hard task to give access to the raw information contained within a RDBMS but because of how it is split in logical relations to avoid unnecessary data repetition [3, Part V], technical knowledge is required to harness its potential into a tangible understanding that can help in the decision-making process.

A good system would enable anyone, expert in computational systems or not, to co-relate and extract the information held in their institution, however this is too broad of a scope. Therefore, an approximation of such system that is able to give meaningful insights and handle the most frequent cases is considered good enough even if it requires some manual work by an administrator.

TODO: Acho que esse eh um contexto melhor

With more than 8,500 students and professors, Instituto Politécnico de Bragança (IPB) is composed of 5 schools that span diverse fields of study including but not limited to Education, Administration, Chemistry, Health, Tourism, Biology and Engineering.

Towards the beginning and the end of the semester and during student registration time, professors often need some insights on their educational affairs. To do so, they get reach out to the Academic System department and request in broad terms what they need, the technician then stop his current task, write a Structured Query Language (SQL) script, run it and email back the results as an spreadsheet file.

With time, the technician built a list of about 40 common requests and their respective scripts so that this process takes less of his time, which in turn enables him to further develop IPB's in-house software. However, interruptions still happen often enough that an automated system is still needed.

`TODO:` acho que isso nao precisa vvv Talvez vira intrtoducao

On a daily basis, a lot new information is generated from its many In-house and third-party software. Their custom administrative portal, `On-Line`[1], alone is able to handle many aspects of an student's interaction with the institution, for example, it can manage student balance, internship application, document request, submission of final thesis and reports, reserving meals and managing an electric bicycle rental subsystem called `IPBike`[2]. More specific features are available in function of user roles so that professors, researchers, employees, and course coordinators spend less time and resources into administrative tasks.

[1] https://apps2.ipb.pt/online
[2] http://ipbike.ipb.pt

Chapter 3

Objective

To develop a web platform that exposes SQL scripts to IPB's employees in a convenient way that does not require expert knowledge of the underlying system architecture and eases the technician workload during the institution's critical moments.

Such platform will be maintained by an administrator that is responsible for registering the desired SQL scripts.

Based on their role in the institution, professors and other employees are presented to a list of queries coupled with meaningful title and description in which they can run, see the resulting table and download an spreadsheet file.

As a non-functional objective, the system should be easily maintained by the current Academic Systems department team and therefore should comply to their current technologies.

Following is a translation of the original proposal found in Appendix A:

> Proposal nº 2
>
> To architect and construct from scratch a "business intelligence" system with emphasis on education management. Now a days we know how valuable and important information is for those who manage institutions and the impact that data analysis tools have in the decision-making process. At the time writing, IPB is already provided of a centralized database through which a

large number of SQL of many diverse purposes queries are run. The intention is that, based on certain criteria, provide access to information without having to manually write queries that are often more than 30 lines long.

The proposed system would be fed with "clusters" of queries and provide a way to easily insert and validate individual or group of queries depending on the currently logged-in user's profile.

Keywords are reuse and automatic parameterization of queries that are supported by an automatic web search interface based on the current query.

All in all, its about deploying a intelligent search system that is able to adapt to the necessities and profile of each user. The end result will always be tables of data that can be exported to many different formats

Chapter 4

Concepts and Technologies

Throughout the development of this project quite a few tools and technologies were employed. To address them all, this chapter was broken in three sections; Front-end in Section 4.1, that is focused mainly in user-facing web technologies; Back-end in Section 4.2, which is concerned with the developed Web Application Programming Interface (API) and lastly; Development in Section 4.3, with the tools used to write, build and test this project.

TODO: Onde eu defino a aplicacao stateless?

4.1 Front-end

This section is concerned with the description of technologies used to assemble the human-interactive part of this project.

4.1.1 Typescript

“ A super-set of JavaScript that compiles to plain JavaScript ” [4], Typescript is a language maintained by Microsoft and developed by *Anders Hejlsberg* in 2012 with the goal of improving the quality and manageability of JavaScript code bases with features such as static typing and object-orientated qualities [5]. Ultimately, Typescript must be compiled

to JavaScript before being executed, for compatibility reasons, the default JavaScript target is version ES3 but newer back-ends are also available.

4.1.2 HTML

The Hypertext Markup Language (HTML), the " World Wide Web's core markup language " [6] is a declarative language through which the vast majority of online content is structured, shared and accessed. It is a specification of elements that can be used to structure the content of web pages, such as headings, images, link to other documents, buttons and many others [7].

4.1.3 CSS

Cascading Style Sheet (CSS) is another declarative language that pairs with HTML. Its purpose is to describe how the elements present in a web page are presented. Some of definitions handle colors, fonts, element arranging, visibility, interaction and many others aspects [7].

4.1.4 SASS

Syntactically Awesome Style Sheets (SASS) is a augmentation of CSS with features that are similar to a object-oriented languages, with loops, variables, functions and rule nesting [8]. SASS files need to be compiled into plain CSS before deployment, there are many of such compilers, some re-generate CSS files upon file changes.

4.1.5 Angular

Front-end web framework developed as a side project at Google that proved itself as a valuable tool for modern application development. The core idea is that HTML faults when it comes to declare dynamic content [9], therefore a new middle-ware is introduced between the rendered page and the underling code so that all the elements and events in

the HTML document are captured and made available to its components. Such binding goes both ways, so if the state of the underling code changes, the document is re-rendered to reflect the new state.

The first version of Angular is now called AngularJs and can be included in a HTML document just like any other JavaScript library. This version proved it's value but was considered confusing and some times, slow. Since then it entered Long Term Support (LTS) stage and no features are added. Angular version 2 and up is a Typescript re-write that includes some new features that aid in the architecture and development of scalable and reusable code, namely, the introduction of Components, Router, Ahead-of-Time compilation and Observables [10].

An overview of key Angular elements follows:

Module Internally referenced as a `NgModule`, these are the basic elements through which an Angular application is structured[11]. They declare the elements that will be provided to its child Modules, Services and Components.

Component Binds a Template to behavior and data. Components are the elements that directly interact with the information perceived by an user. They typically rely on Services to acquire information and on Modules to fulfill their dependencies.

Router A special kind of service that is responsible for managing the navigation through an application, mapping Universal Resource Locator (URL)s to Components.

Service Akin to a library, a Service have methods that can be used by multiple Components and other Services to provide some functionality. They are commonly implemented to act as the means of interaction to a remote API.

Template An augmented HTML file that is bound to a Component. Effectively, they are medium through which information is displayed and interacted with. Among other things, Templates can have elements that are dependent of some expression, values that are provided by a Component and events that notify the underlying Component.

The relationship between such elements is that a Component may be dependent on Services, Components and Templates form what is called a View; Views and Routers are exposed to the application under a Module; which in turn forms a tree with a root Module. Other important features include the dependency injection mechanism, template directives and directional data binding. Through these features Angular aims to be a highly modular framework capable of fast development.

4.1.6 Angular Material

Material Design is a set of guidelines and principles made by Google for designing User Interface (UI) that aims to bring natural and consistent interactions between users and computers. The guiding principle is based on paper and ink but it is not limited to what they can do in the physical world [12].

Angular Material [13] is the implementation made by Google of components like buttons, text input and separators that follow the Material Design guidelines to be used by Angular applications, providing a consistent look across devices.

4.1.7 Sb-Admin-Material

To accelerate the development speed and have faster working prototypes, many web-based projects begin form a ready-made template. This saves time by keeping developers from re-writing common pieces of code commonly referred as “boilerplate”.

SB Angular Material is a re-write of the famous SB Admin template [14], a free and open source template developed by Start Bootstrap [15] in Angular using components developed in the previously discussed Angular Material project.

As the name implies this template tries to assess the need for an administrator panel, and in doing so it provides a few ready-made components, to name a few, a login component as seen in figure 4.1; the main screen with a top and a collapsible side navigation components, seen in figure 4.2 and 4.3. This template already encompass some amount

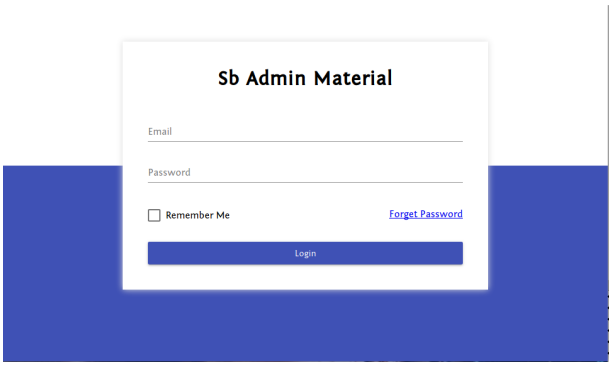

Figure 4.1: Login Screen

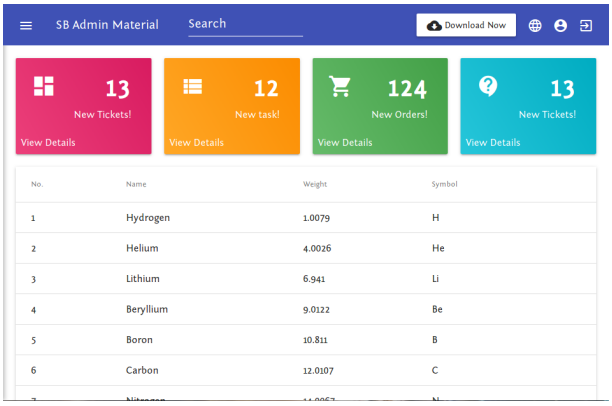

Figure 4.2: Dashboard with collapsed side menu

of responsive design by toggling the ability of said side navigational panel to be collapsed depending on the user's screen width.

In the following subsection this template's folder structure will be explained so that one can understand where what are the main parts in which it can be extended to fit any particular project.

Project Structure

SB Admin Angular was written with the intention of being modified and extended by other developers. Because the team did not express any guidelines towards how it should be further developed, it is important to give an overview of the current project structure so that the changes made to accommodate the topic of this project are better understood.

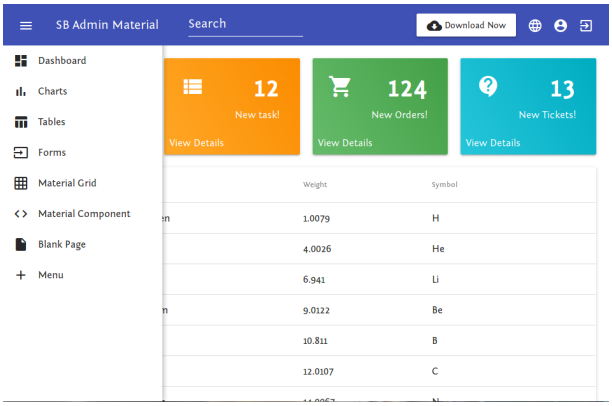

Figure 4.3: Dashboard with visible side menu

- **root** This item is not a folder but the root of the project. In here there are configurations for code linter, JavaScript dependency descriptor and the license statement.
- **dist** Once the project is built for deployment, this directory will hold all the assets and optimized code ready for production, including the main `index.html` file that bootstraps the whole project.
- **e2e** This holds the source code for End-to-End test cases, hence the name.
- **src** This is the heart of this template, a directory that holds all the structure, content and behavior needed per application.
    - **app** The Angular entry-point and application wide router module.
        - ➛ **layout** All the components used to compose the navigational elements and menus and their subsequent pages plus some example pages.
            - **black-page** A inaccessible component that does nothing, probably unfinished.
            - **blank-page** An example component that is white.
            - **charts** A component that display chart capabilities of the integrated JavaScript module chartjs[1].

[1] Available at https://www.chartjs.org/

        - **components** Omnipresent page elements such as the Topbar and the collapsible Sidebar.
            - **topnav** The blue navigation top bar as seen on Figure 4.2.
            - **sidebar** The Menu on the left side of the screen as seen on Figure 4.3.
        - **dashboard** The page in which the user is redirected after logging in.
        - **forms** Demonstration of the many different input methods such as Auto Complete text input, Date picker, Text Area and others.
        - **grid** A demo of the available page subdivisions.
        - **material-components** An example page displaying the main components of Angular Material such as buttons, Dialog and Notifications.
        - **nav** Unused component, deprecated by the side bar component.
        - **tables** A example component displaying Angular Material's table mechanisms.
    - **login** This is the Login component as see on Figure 4.1
    - **shared** Code that can be used in a application wide manner so that higher abstractions and code reuse can be achieved.
- **assets** Static content directory. Images, fonts, and i18n translations.
- **environments** Depending on how the project is run, either in development or in production mode, a the respective configuration file that holds environment constants is used, allowing developers to use the same reference name throughout the code base no matter the environment.
- **styles** SASS files that define the look and feel.

After this overview, it is interesting to note the following:

- The Layout folder hosts, for the most part, Components and Modules that are listed in the sidebar.

- There are some unused components that were probably left over from design changes and were not deleted, which is the case of black-page and the nav component.
- There are many examples that proved as a handy reference during development, namely the forms and material-components.

4.2 Back-end

The tools and concepts described in this section refer to the server-side of this project. It manages the authenticated and authorized data access as well as business-specific functionalities.

4.2.1 Stateless Web Application

TODO: Escrever sobre mim!

4.2.2 HTTP

TODO: Escrever sobre mim ! e nao esquecer de mencionar os *verbos*, principalmente o delete pq eh usado na implementacao

4.2.3 Java

Given how ubiquitous it is, there is not much to be said. Java is a Object-Oriented programming language firstly developed by James Gosling at Sun Microsystems, it is statically and explicitly typed and gets compiled to a machine-independent byte code that is then interpreted by the Java Virtual Machine (JVM) [16].

Because of its high adoption, many concepts were developed to accommodate its deficiencies and improve the development cycle. In fact, because of the recurring solutions for recurring situations in software design, a group of skilled professionals got together to discuss and write a book entitled "Design Patterns: Elements of Reusable Object-Oriented Software" [17], bringing the concept of repeatable ways to some problem.

This leads to the next topic, View Model, which was used throughout the development of this project to filter the information that is sent to a non-administrator user, removing from the role of filtering sensitive information to the insecure front-end application.

View Model

The ViewModel is a piece of a bigger pattern called Model View ViewlModel (MVVM) created by John Gossman to address the scenario in which a model as described in the Model View Controller (MVC) pattern can't be completely mapped to a View [18], therefore it is sensible to specify another model that partially reflects the original model but is able to be completely bound to user interface elements.

During the implementation of this project, the ViewModel pattern was used out of the MVVM pattern, however the goal of decoupling code the same. More precisely, some models can not have all of its attributes serialized back to a View or they expose different attributes depending on the user who is requesting it; ViewModel classes were implemented in such cases.

4.2.4 Spring

Developed by Pivotal Software in 2002, the Spring Framework provides most of the "plumbing" necessary for fast development and deployment of enterprise Java applications [19], some of the main features include: Dependency Injection, a Inversion of Control (IoC) mechanism; Spring Data, a set of tools for information access such as Object-Relational Mapping (ORM) configurations; Spring Security, a application-wide security mechanisms and configurations.

One of the strongest design philosophies of the Spring Framework is the following:

> "Provide choice at every level. Spring lets you defer design decisions as late as possible. For example, you can switch persistence providers through configuration without changing your code. The same is true for many other infrastructure concerns and integration with third-party APIs." [19]

This leads to a feature-centered development model that is able to quickly deliver prototypes and changes on-demand.

Dependency Injection

Conceptually, Dependency Injection is a implementation of the IoC principle, abdicating any given class class from managing its own dependencies; leaving them to a overseer object that knows how to create and inject dependencies to each class[20].

In practical terms, when writing a new class the programmer declare its dependencies through some mechanism in which the framework is able to reason about. Later in the run-time when a such dependent class is about to be instantiated, its dependencies are made available and injected into the new instance.

The key idea is that for the most part an application does not need to know which specific class is provided as long as it implements some given interface. It is the Framework's job to choose which class is injected. The programmer, however, is able to tailor the Framework's behavior to their liking.

When using Spring, one can express dependencies by declaring them in the class's constructor or by annotating an attribute using the Autowired annotation[21].

Data

`TODO:` essa seccao precisa discorrer sobre o que é um repositorio do spring, o que ele faz, porque é mais rapido usar ele e talvez um exemplo já que a implementacao eh simples
When developing a enterprise-level application, often times there is a need for some sort of persistence storage, in pactice this usually translates to a RDBMS. To reduce the amount of code needed to manage such interactions, Spring Data module was developed. Its main interface is called `Repository` and it decouples entities that are being persisted from the underlying storage system [22].

Although it acts as the root interface through which other, more feature packed, models are developed, `Repository` is hardly used directly [23], instead `CrudRepository` and `PagingAngSortingRepository` are interfaces that provides useful features.

Listing 4.1: Repository for YabiUser

```
1 public interface YabiUserRepository extends PagingAndSortingRepository<
      YabiUser, Long> {
2     public YabiUser findByName(String name);
3 }
```

As the name suggests `CrudRepository` offers Create Retrieve Update Delete (CRUD) capabilities for a specified entity and `PagingAngSortingRepository` extends it by including paging and sorting support.

The `CrudRepository` on its own provide the following default methods:

save(Entity) Persists a entity in the data store and returns the saved instance with updated generated values.

findOne(ID) Retrieve one entity from the data store. If not found, returns null.

findAll() Retrieve all entities currently stored as a list.

count() Returns a number that represents the amount of entities currently stored.

delete(Entity) Deletes the given entity. Returns nothing.

exists(ID) Checks the data store for the existence of the given primary key value.

As shown in Listing 4.1, a entity-specific repository can be made by declaring a new interface that extends a base `Repository`, specify the generic types for the entity and the primary key and lastly declare custom methods following a naming convention. In this case a `PagingAndSortingRepository` for `YabiUser` with a `Long` as the primary key was defined as a `YabiUserRepository` and it exposes a new method that retrieve an instance of `YabiUser` given it's username.

These functionalities are made available through the following modules that come with Spring Data:

- Spring Data Java Database Connectivity (JDBC).

- Spring Data Java Persistence API (JPA).
- Spring Data Lightweight Directory Access Protocol (LDAP).
- Spring Data Represnetational State Transfer (REST)

Hypermedia As The Engine Of Application State (HATEOAS)

REST, according to its creator:

> "REST is a coordinated set of architectural constraints that attempts to minimize latency and network communication, while at the same time maximizing the independence and scalability of component implementations. This is achieved by placing constraints on connector semantics, where other styles have focused on component semantics. REST enables the caching and reuse of interactions, dynamic substitutability of components, and processing of actions by intermediaries, in order to meet the needs of an Internet-scale distributed hypermedia system." [24]

In other words, this "new" architecture expresses requests and responses as the application state itself being transmitted in the client and server approach. There are a few ways in which the state can be communicated and structured, one of which is the subject of this section.

HATEOAS is a response structure that enables a client to discover and navigate related information for that resource[24]. Mainly it is able to specify what are the related information, where it is located and how to interact with it. This is accomplished by including some meta-data in the response in which the client can parse, present to the user and issue proper requests.

In the context of Spring Framework, `CrudRepository` is a interface which can be extended for each class with Entity annotation that needs integration with some kind of persistence mechanism. In its default implementation there is a REST service that is coherent with HATEOAS, providing CRUD capabilities; indexing as in listing all entries

and internal resource linking, which give the ability to retrieve a resource's related resource (think of if like a RDBMS table relation).

Security

This spring project aims to provide both authentication and authorization mechanisms throughout the application's components, exposing implementable interfaces that enable developers to override necessary parts for their specific needs.

It is important to clarify the distinction between Authentication and Authorization because they have their respective software counterpart that play important roles in Spring Framework.

Authentication, in by definition means "To prove real or genuine" [25]. In Spring this translates to a custom extension of the `WebSecurityConfigurerAdapter` abstract class that defines how to verify that a given user exists and is allowed access the resources. There are a few ways to achieve this, two of the most common are: JDBC authentication, though which credentials are queried and matched from a RDBMS and LDAP authentication, that binds to some remote directory for the given user and password pair. Note that it does not define *what* may be accessed by such user, only if the used has access to the system as a whole.

Authorization, "the act of endorsing, or permitting by some recognized authority" [25]. Similarly to Authentication can also be specified via a custom extension of `WebSecurityConfigurerAdapte` and they can co-exist in the same extension. Authorization mechanisms are usually related to some attribute of the current authenticated user, in Spring, the `GrantedAuthority` interface is the central piece that unifies *what* the user has access to. Authorization points can be defined at the global level by the `WebSecurityConfigurerAdapter`, at the controller level or at the method level though proper annotations.

Boot

Although Spring Framework is a marvelous piece of software for its malleability and wide range of available features, for a while it was considered a *Configuration Hell* because

of its eXtensive Markup Language (XML) configuration that would require a lot of expertise into writing [26] [27] [28] [29] [30]. In face of this, the Spring Team came up with Spring Boot, a dependency that can be inserted into a project and provides sane, already-configured Spring packages to accelerate development and keep code organized.

4.2.5 MariaDB

Due to legal concerns Michael Monty Widenius founded Monty Program AB, whose main product, MariaDB, started as a fork of his previous work, the MySQL RDBMS [31]. Such relational databases allow the user to define data structures and perform operations such as inserts, retrievals, updates and removals through a language known as SQL

MariaDB is an open source project licensed under the GNU General Public License (GPL) and its current stable version is 5.2. Its SQL dialect is and configuration files are either identical or very similar to those of MySQL. One of the main goals of MariaDB is to keep enhancing its performance [32]

MariaDB server is said to properly execute in many operating systems, namely Microsoft Windows, Solaris, Linux, MacOS and Free BSD. There are many packages that handle connections to MariaDB, graphically like DBeaver or phpMyAdmin, textually like mycli and not further than that, programming language connectors such as Java's JDBC [33].

4.2.6 Apache Directory Studio

LDAP in its core is a protocol defined by Internet Engineering Task Force (IETF)'s Request For Comments (RFC) number 4511 [34] that defines access to X.500 compliant directory services. A Directory is an agglomerations of cooperative systems that serve structured information about the real world [35]. Different from a traditional RDBMS, directory services are expected to be automatically accessed by other interconnected systems, therefore they are better optimized for frequent queries and fewer updates.

Alex Karasulu, founder of the Apache Directory Project, was right when he stated that the need for interconnected systems grew alongside the expansion of the Internet but unlike his expectation, Directory services were replaced with RDBMS systems that don't exactly address the same goals and further complicate interconnected systems[2].

Given this situation, his project have the goal to modernize the tooling and functionalities of Directory systems and in doing so, two main sub-projects were created: Apache Directory Service [36], a modern, LDAPv3 compatible, Java based implementation of a Directory Service that introduces triggers, stored procedures, view and queues; Apache Directory Studio [37], a complete LDAP tool developed as an Eclipse Rich Client Platform (RCP) extension that offers a more friendly user experience with visual elements for LDAP Data Interchange Format (LDIF) editor, tree explorer and permission management.

4.3 Development

This section cover the tools used during the development phase of this project. The tools in question do not only satisfy the coding needs but also mimics the production environment through which the application interacts with so that no sensitive information was touched by a potentially insecure, unfinished application.

In general, there was a need to comfortably edit Java and Angular projects, with code completion and refactoring support; a project manager that automatically downloads dependencies; a container tool to quickly deploy an environment with databases and directory services, without the need of editing non-functional configurations and lastly, a browser to access the system as the end-user would.

4.3.1 Apache NetBeans

One of the Duke's Choice Award winner [38], NetBeans is a general-purpose, cross-platform Integrated Development Environment (IDE) mainly focused for Java development with the goal to maximize productivity. In 2016 NetBeans was added to the Apache

Incubator so that it could be further developed by the community [39], as of 2019 it became one of Apache's Top Level project [40] and is expected to attract a even bigger community.

Because Java was the main focus of NetBeans during its first few years, support for the language is very broad in features. Developers can easily operate code with context-sensitive refactoring; mark line, method, expression and class breakpoints; step through paused code; automatic JavaDoc generation; semantic code completion and more [41].

Through its module system, support for other languages and resources were introduced, namely source code management with Git, Mercurial and Subversion, database management with support for viewing data and running SQL queries, unit testing, PHP, HTML, JavaScript and CSS [42].

4.3.2 Maven

Maven is a Build System for mainly used for Java applications, that is: Through a the `pom.xml` file, developers declare their project's attributes and dependencies file and if needed, tweak the building process; from there on, Maven is capable of downloading dependencies from a remote repository, compiling them if necessary and generate an executable Java ARchive (JAR) file or loadable library [43].

The project goal is to unify the project structure so that there is less time spent by the developer to understand how the a given application code is arranged and to centralize common project actions such as previouly mentioned dependency resolution, run unit and integration tests and generate packages that are able to be distributed [44].

4.3.3 Lombok

This plugin offers an annotation-based code-scaffolding tool for class definitions. Give the right annotations, common methods like getters, setters and no attribute constructor are automatically generated in build or compile-time [45]. This tool consists of a two-part system that includes the integration with the compiler/build-system and another one that

interacts with the developer's IDE so that the completion system is able to recognize the implicitly generated methods.

Some of the notable annotations include:

- `@Data`, useful for Plain Old Java Object (POJO) classes, this annotation generates getters, setters, a string converter and equality methods.

- `@NoArgsConstructor` and `@AllArgsConstructor`, as the name implies, one generates a constructor that takes no arguments and the other, a constructor that generates all arguments.

At first glance this might not be a necessary tool given that most IDEs often have support for a similar form of code refactoring but the key difference is that lombok does not clutter the classes with generated implementation therefore it reduces the project's Line of Code (LoC) count.

4.3.4 Visual Studio Code

Microsoft's take on open-source, this code editor gained traction among developers as one of the most used code editors [46]. Like any other modern code editor, it offers syntax highlight; auto-completion, though Microsoft's *IntelliSense* integration and a plugin system that enables users to add custom behavior and further develop the editor's support for programming languages [47].

One of the acclaimed features that arose with Visual Studio Code was the open source specification of Language Server Protocol (LSP) [48]. This specification aims to define a communication protocol that is used between a code editor, referred as a LSP client and a editor-independent program referred as a LSP server, that takes care of features such as code completion, highlight, error detection, contextual variable renaming and jumping to definition [49]. This decoupling reduces the amount of code needed to develop a highly capable editor because language-dependent support is now transferred to said LSP server.

4.3.5 Docker

Akin to a Virtual Machine (VM), containers provides a way to have a different computing environment than the active running in the hardware. The key difference is that instead of emulating the whole computing stack, from processor to applicaion, a container system shares the core host resources with its guests and thus is generally less resource-hungry. One downside of a container is that the guest operating system must share the same kernel with the host.

In the other hand, Docker is more than just the sandboxing of processes, handling image building through a `Dockerfile` specification, containers that can be shared among different machines, a online registry of extendable containers, a command line interface that downloads, builds and manages containers [50].

Core Docker definitions are brought up [51]:

Dockerfile A file that declares the steps taken to build an Image [52].

Image A blueprint of a container generated once a `Dockerfile` is built.

Container If an image is an compiled binary, a container is the running process.

Volume Is a shared folder between the host Operating System (OS) and a running container.

4.3.6 Docker-compose

The flexibility offered by the Docker ecosystem is of great use for moderate applications. Define a `Dockerfile`, build it into an image and create a running container. However, not all solutions require a single image, in fact, most of them are composed of independently working pieces that are tailored together and bundling all pieces in a single image is not considered good practice [50].

Distributed with Docker, this tool provides a way to define a multi-container applications, their virtual connections and manage all containers as one single entity. Such

separation of concerns enable the tool to take smarter choices when starting a solution that includes a modified image, preserving volume data between solution runs and on the development side, it makes a local instance of a solution to be quickly deployed to testing purposes [53].

4.3.7 Chinook Database

This is a single-script data-set. Structured in many SQL dialects, it represents a digital media store and in total it contains 11 tables, 11 constraints, 66 columns and 15607 rows [54]. The data used for this project was generated from a real iTunes library, sales information was randomly generated and customer and employee were manually inserted [55].

List of Available dialects:

- MySQL
- SQL Server
- SQL Server Compact
- SQLite
- PostgreSQL
- Oracle
- DB2

Such diversity in dialects, structural complexity and amount of information makes the Chinook Database a good sample for database-generic applications.

4.3.8 Angular CLI

Angular applications have a basic directory for its components. Often times a directory contains most of the code for a specific piece of an application, such as a Component

definition, a Service, a Template, a Style definition, a Class definition and lastly, these related pieces are then declared in a Module definition.

Managing this volume of files and relations can sometimes lead to confusion. Angular Command Line Interface (CLI) was developed to make this task more manageable. With it a developer can quickly initialize a new application skeleton, generate Components and Modules, build a deployment-ready applicaion and run a testing server that re-compile with code changes [56].

4.3.9 Firefox

From the downfall of Netscape browser and the release of its source code, the Mozilla project started with the mission to ensure the Internet to be a global public place, open and accessible to all [57]. Its main product is the open source web browser, Firefox.

Firefox comes bundled with plenty of tools that facilitate web development, considered as the most valuable are the following:

Source Mapping Is the ability to map some generated code back to its source. Some web frameworks like Angular4.1.5 generate applications that are not developed in JavaScript itself but compiled to JavaScript in order to be executed [58]. In the beginning this led to great confusion because the browser's built-in debugger would display not the original source but the generated code.

Debugger Support for breakpoints, conditional breakpoints, expression stepping and variable lookup. Even better with the previous element [59].

Network Monitor Often times it is needed to inspect the outgoing requests and their responses. The integrated monitor is able to expose all elements of the communication exchange and measure the different attributes of a network operations [60].

Storage Inspector Provides access to the information storage that is managed by the browser for each page [61], namely:

- Cache Storage

- Cookies
- Indexed Database (DB)
- Local Storage
- Session Storage

Console Enables the input of expressions in the page context and output information associated to the current page, including explicit calls for the `console.log` function.

Page Inspector Examine the page's HTML structure and CSS rules [62]. Developers can quickly experiment new possibilities by temporarily altering CSS rules and the page's structure.

4.3.10 Webpack

With the growing complexity of web applications, websites got slower and development, trickier. Webpack was developed to generate a optimized, ready to run, package that can be deployed in production [63]. Such packages are not meant for code only, they may contain images, CSS rules and anything else. It offers an API that can transform the contents of a package before it is bundled, for example, Typescript sources and its compiled counterpart or extracting inline CSS from a HTML document into a separate file.

In the context of Firefox's debugger and an angular 2+ applications, when serving the application using Angular CLI, discussed in Section 4.3.8, it automatically bundles Typescript source code so that it can be instrumented and debugged inside the browser by accessing the Webpack element under the debugging tab.

4.3.11 Postman

As programs grew larger and were split into smaller pieces, it is now a common practice to have a API that concentrate on the business logic and a Graphical User Interface (GUI)s that consume them; Such separation of concerns got even more pronounced when Web

systems popularized, web browsers acting as GUI and interacting with remote Hypertext Transfer Protocol (HTTP) servers as their source of information.

As with any software, APIs need to be tested and validated in order to provide a good quality product. In this context Postman was developed to be a Web API suite, offering a nice user interface through which developers can not only send, receive and analyze HTTP requests but also generate and manage documentation so that front-end and back-end teams have a single source of truth, manage test cases for remote HTTP APIs, mock API that are still under development and more [64].

Chapter 5

Project

This section will discuss the entities and requirements evaluated from the proposal. TODO: e o design geral do project, overview das seccoes

5.1 Requirements

Given the written proposal and eventual consultations with stakeholders, the general behavior and requirements of this application were evaluated. In regards to use cases, two actors were identified, a User that represents professors and employees that belongs to IPB and a Administrator whose use-cases include classic CRUD operations.

As an authenticated User, there are two goals that are covered by the use-case diagram shown in Figure 5.1, Run a Query and Export a Spreadsheet. The former models the action of temporarily accessing some of the institution's information and the latter, even though taking the same steps, provides the user with a file that can be processed by other tools or used at a later time.

An authenticated Administrator, because he is expected to manage the system, is provided of more use-cases that follow a similar pattern, that is, managing the entities that compose the system by the means of CRUD operations. Added to this, the use-case diagram found in Figure 5.2 brings to light even another action, to "Associate User and Permission" that indicates the process of enabling an User to run a Query.

The proposal expresses the following requirements:

1. The system should be able to run queries in the database currently employed at the institution.

2. Users can only run queries in which they have Permission to.

3. Query commands may be longer than 30 lines long.

4. Running a Query yields a table that can be downloaded.

5. No SQL knowledge is needed to execute a Query.

6. The system enables the insertion of new Queries.

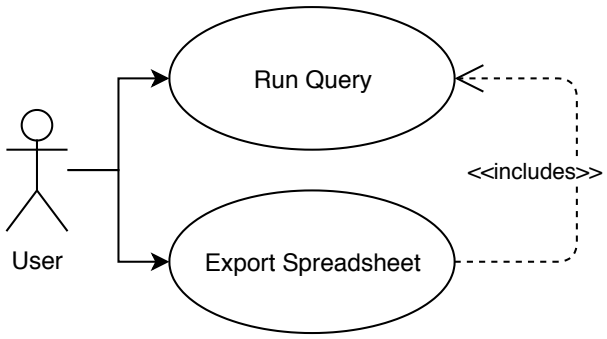

Figure 5.1: User use-case

Database Where a Query is run.

It is responsible to hold the information that enables the access to a given database, effectively meeting requirement 1. In the most basic form, a connection requires the network address and authentication credentials.

Query A script that is run in a Database and gather information into a single table.

A SQL script must be issued during a session with a Database. To fulfill requirement 5, some meta information such as a title and a description so that the target audience is able to find the Query that fulfill their needs.

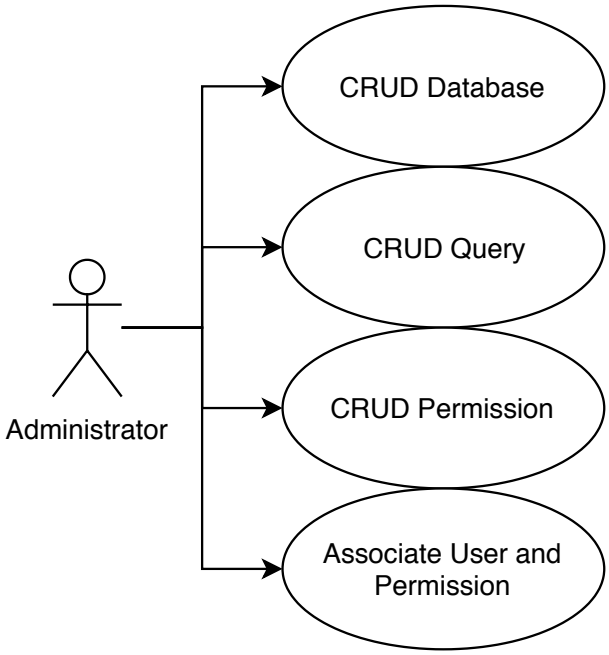

Figure 5.2: Administrator use-cases

Permission The binding between Users and Queries.

In order to fulfill requirement 2, this entity is responsible to handle the relation between a User and all the Queries that they may access.

User Represents the person currently logged-in.

It has two purposes, first is to differentiate users according to their roles, either "Administrators" or "User" so that certain actions are disabled, for example the Administrative task stated in requirement 6 The second is to be used when filtering Queries so that requirement 2

TODO: Comentar a imaem

5.2 Project Details

TODO: importante mas nao sei onde por ainda Queries must be associated to one Permission. Users may have more than one permission. The system must authenticate using

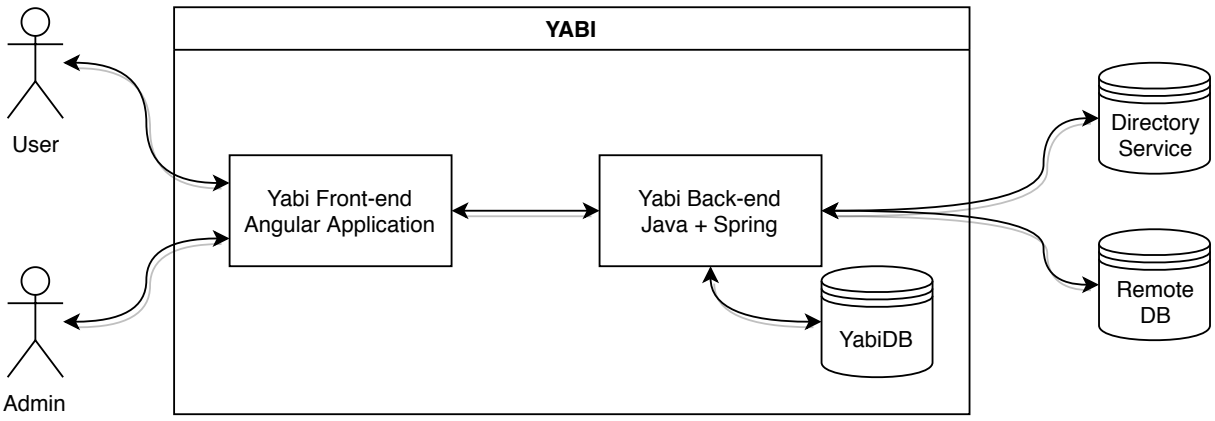

Figure 5.3: Yabi Overview

the institute's centralized directory server. An Administrator can manage the system by altering everything that is in the scope of this system.

Permissions is a central piece of this system. It's what associates users with information. Permissions follow a hierarchy. There is a root Permission whose path is simply "/" and it's parent is itself. There are two roles that any given user may be assigned to, either Administrator or User.

5.3 Class Diagram

TODO: citar a batida de nomes TODO: de alguma forma, chegou nesse diagrama

5.3.1 Query

...

5.3.2 Database

...

5.3.3 Permission

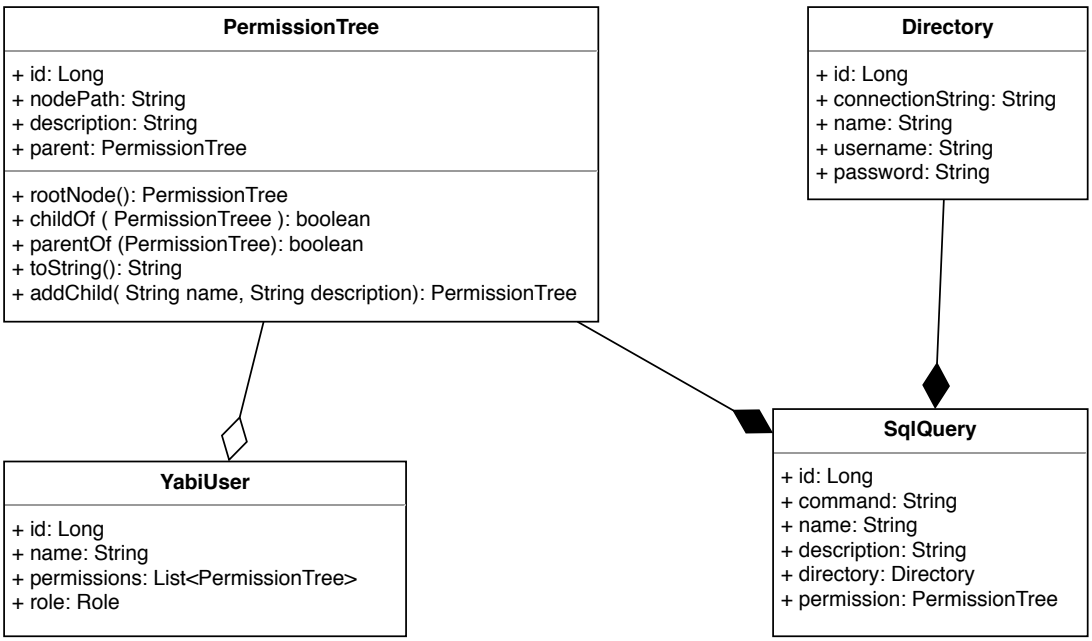

Figure 5.4: Class diagram

TODO: O no raiz deve sempre existir, as permissoes sao pendiradas na arvore e a raiz precise ser boostrapped no deploy. se a raiz for deletada, nao tem como adicionar mais permissoes. ...

5.3.4 User

...

5.4 Authentication and Authorization

...

5.5 Template Sb-Admin-Material

...

5.6 Multi-Database Support

TODO: Deve citar os bancos mais usados, e o banco mais usado pelo IPB (oracle)

Even though the context in which the application described in this document will be primarily accessing Oracle databases, support for other databases was added to broaden its usefulness.

As far as the project for implementing this feature goes, Figure?? hopefully describes the process. When a request to run a given Query arrives,

Chapter 6

Implementation and Results

6.1 Front-end

6.1.1 Component Structure

When developing the Angular front-end over the `Sb-Admin-Material` template, it was noted that the example pages that could be accessed by links listed in the left sidebar, seen on Figure 4.3, were found inside the `/app/layout` folder. Therefore it made sense to follow this approach and implement Yet Another Business Inteligence (Yabi)'s custom pages in the same place.

In general, each entity have got a folder that contains:

- A "model" file with:
    - A class that represents the entity, to be used when retrieving entities depending on the current user.
    - A class that extends `PagingAndSortingRepository`, to map Spring Repository responses.
    - A class that extends `Entity`, representing the elements contained within the repository response.

    - A repository "accessor" that indicates key used to access the list of `Entity` contained within the repository response.

- A Service file that extends `PagingAndSortingRepository`, specifying it for the given entity.
- The Module file declaring its dialog Components to also be loaded.
- The Template file that renders a listing with the available entities.
- The Component file that interacts with the Template.
- The Style file with rules to correctly render the "add" button.
- A folder with a dialog Component for creating more entries.
- A folder with a dialog Component for editing an entry.

There are some variations to this rule. `User` Component only has only one dialog that is used to show more information about the current user. `Query` Component was one of the first components to be developed and it has three dialogs, one for showing more information and the results of running it, one for creating new queries that is not used and a "form" dialog that maps a `Query` to input elements that is used for editing and creating.

Components & Dialogs

db

query

permission

user

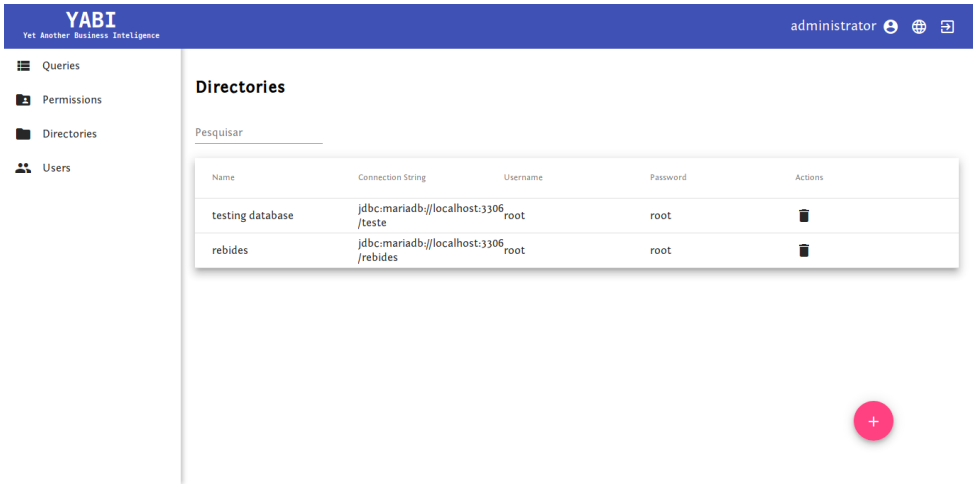

Figure 6.1: Listing of all registered `Directory`

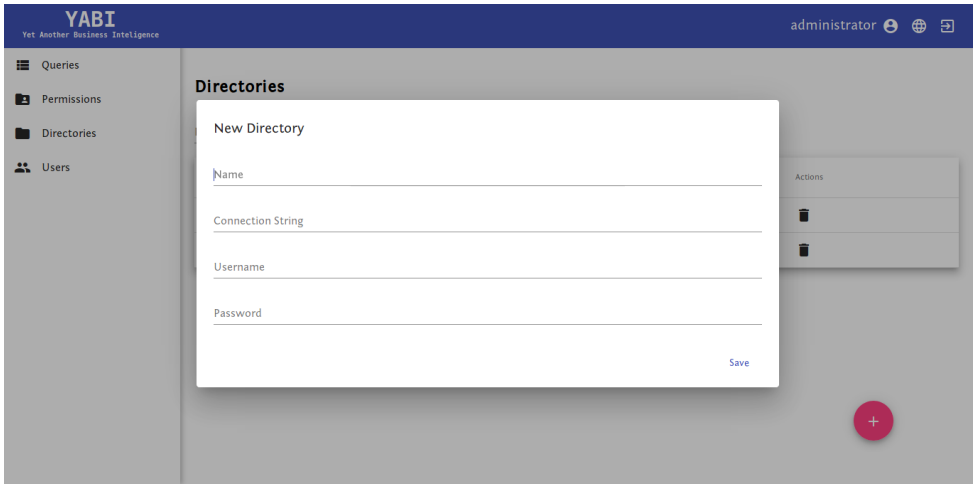

Figure 6.2: Dialog for creating a new `Directory`

Services

Modules

...

6.1.2 Generic Form Control Builder

6.1.3 Spring HATEOAS Classes

...

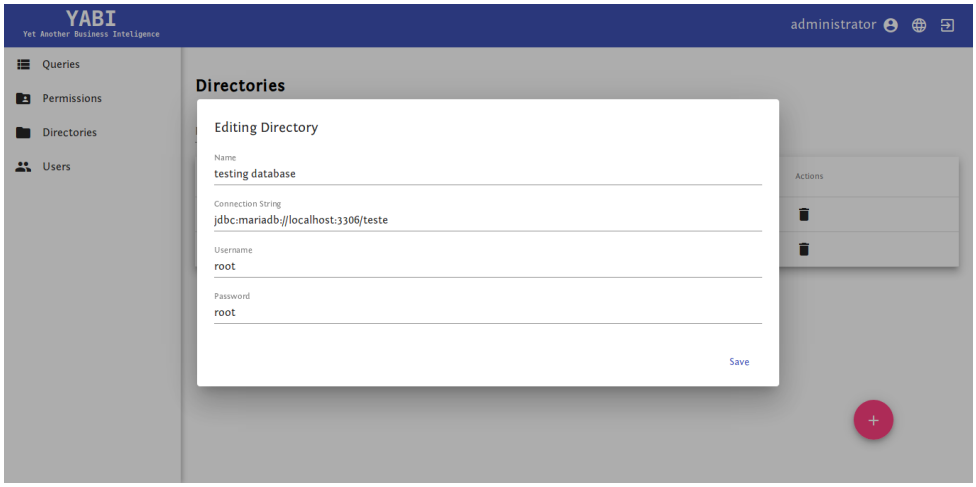

Figure 6.3: Dialog for editing an existing `Directory`

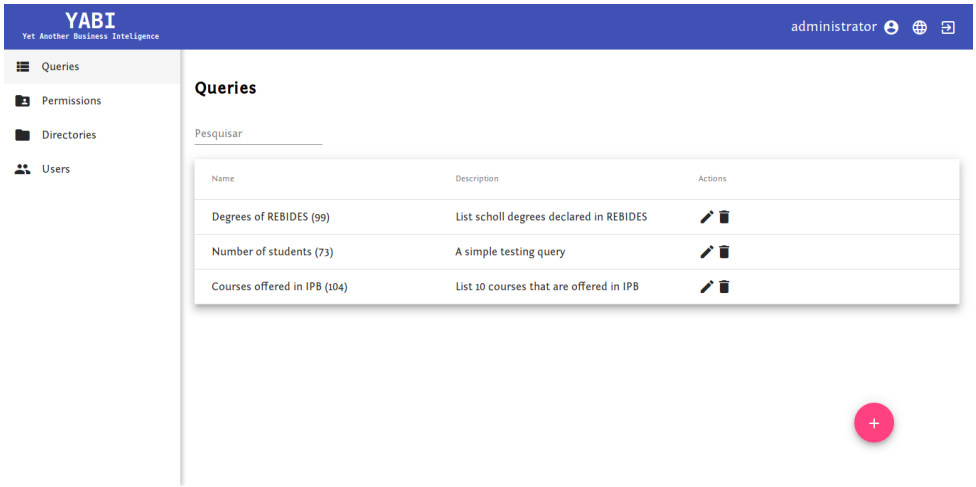

Figure 6.4: Listing of all registered `Query`

Entity Class

...

Acessor Class

...

Repository Class

...

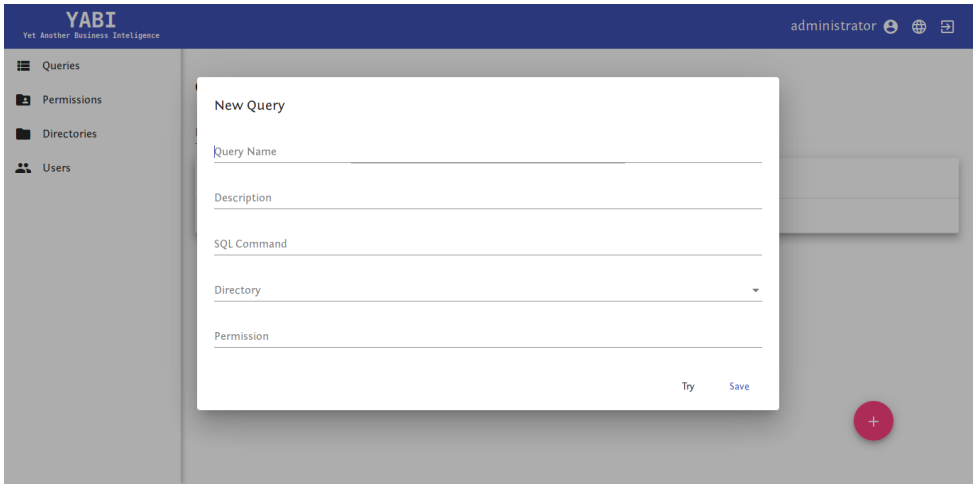

Figure 6.5: Dialog for creating a new `Query`

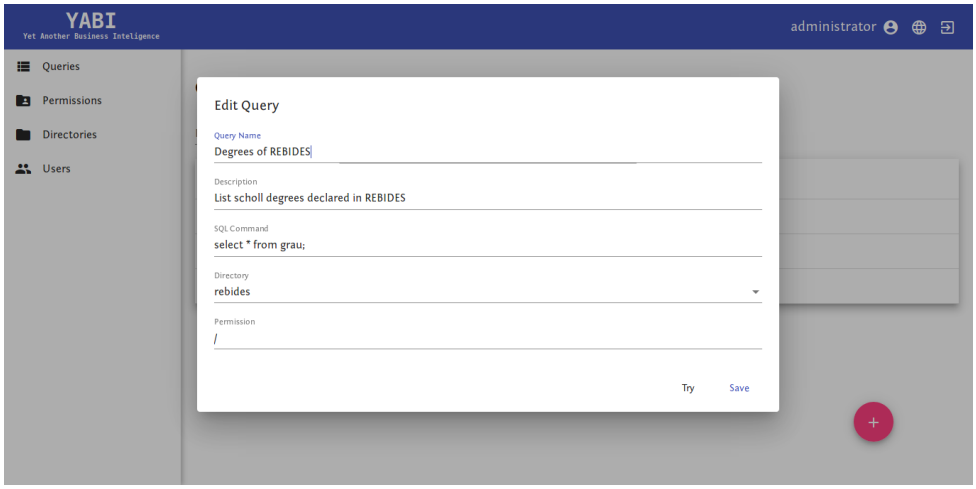

Figure 6.6: Dialog for editing an existing `Query`

Repository Service Class

...

6.1.4 Temporal Caching Repository

...

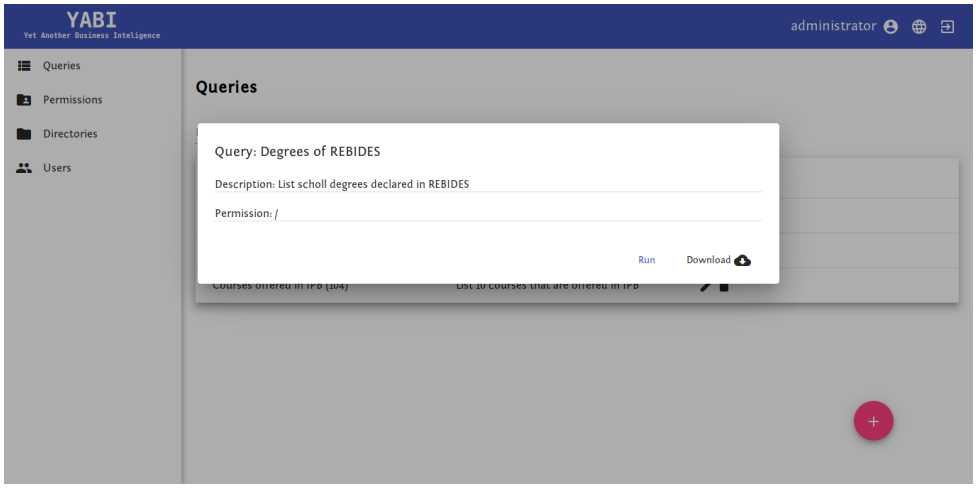

Figure 6.7: Dialog for running a `Query`

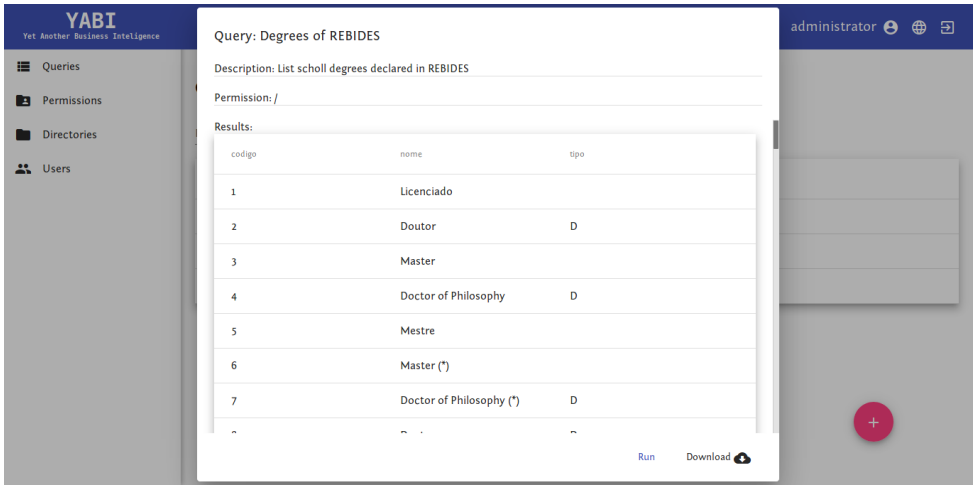

Figure 6.8: Dialog for running a `Query` after it was executed

6.1.5 Error Handler

...

6.1.6 Database Reader

...

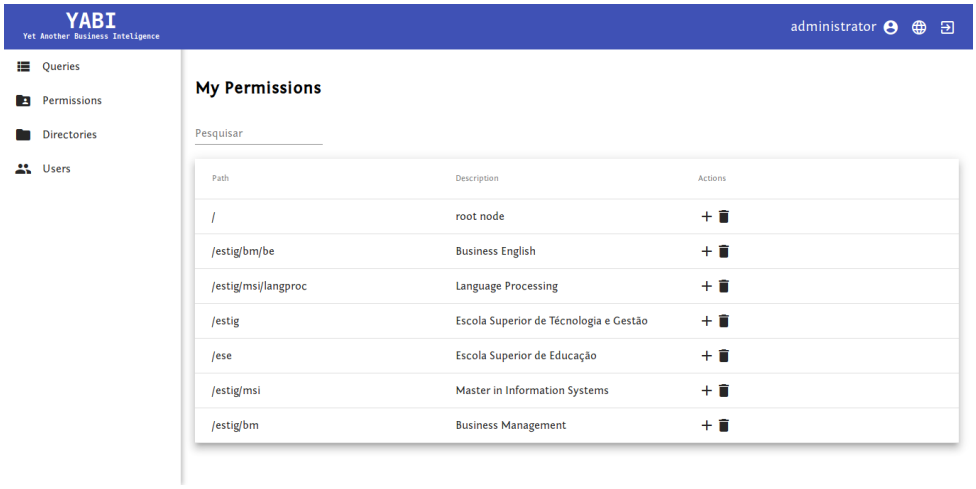

Figure 6.9: Listing of all registered `Permission`

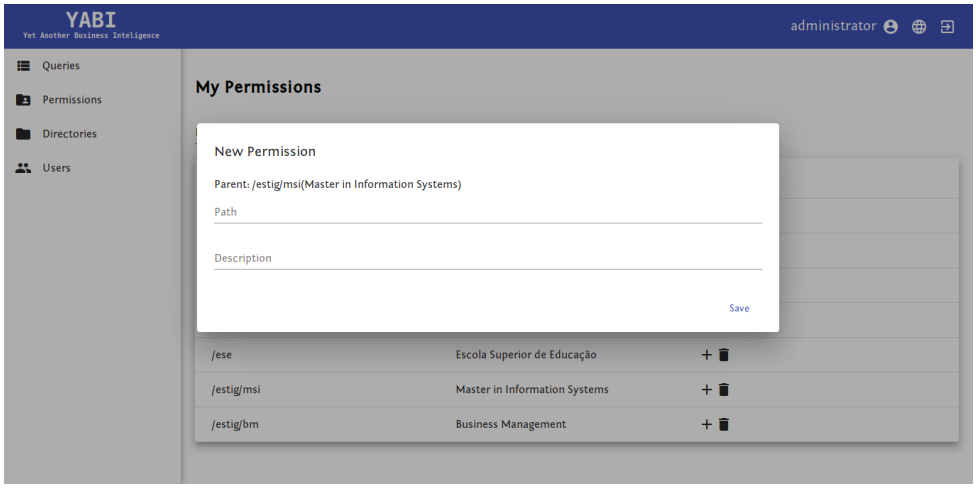

Figure 6.10: Dialog for creating a new `Permission`

6.2 Back-end

...

6.2.1 Entities

Following the class diagram in Figure 5.4 and their relations, classes were created and properly annotated with so that JPA is able to properly generate a relational model. Hence the `Entity` annotation is present in all classes.

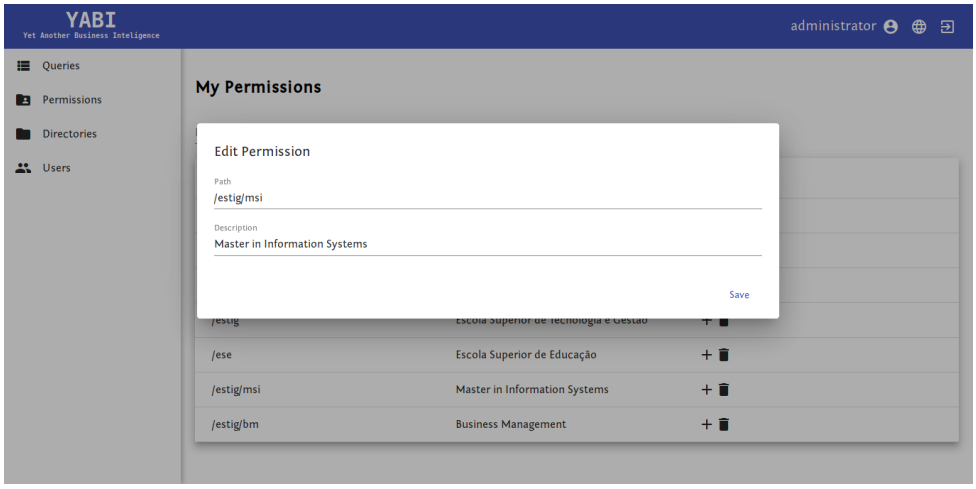

Figure 6.11: Dialog for editing an existing `Permission`

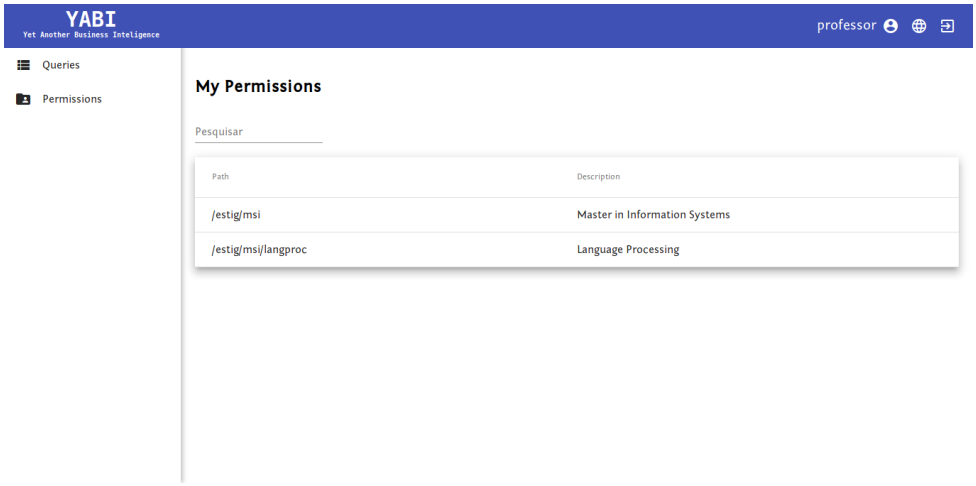

Figure 6.12: Listing of all registered `Permission` for a given `User`

Listing 6.1 presents the implementation of the Query model defined in Section 5.3.1, it servers as an overview into other model implementations as it shares much of the common features but also adds some of its own.

Lines 1, 2 and 5 are provided by the Lombok package as discussed in Section 4.3.3, instructing the creation of constructor and other common methods during compilation. All models make use of `@Data` and `@NoArgsConstructor` annotations and all but `PermissionTree` uses `@AllArgsConstructor`.

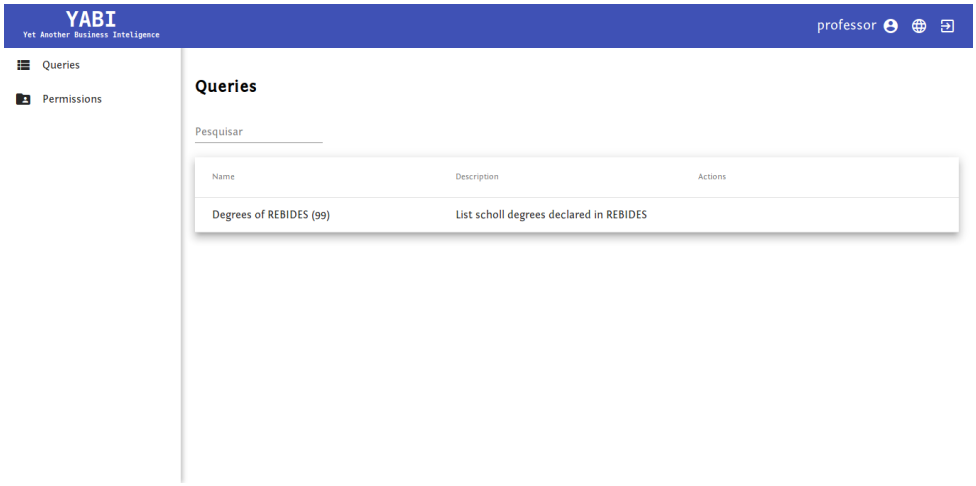

Figure 6.13: Listing of all registered `Query` for a given `User`

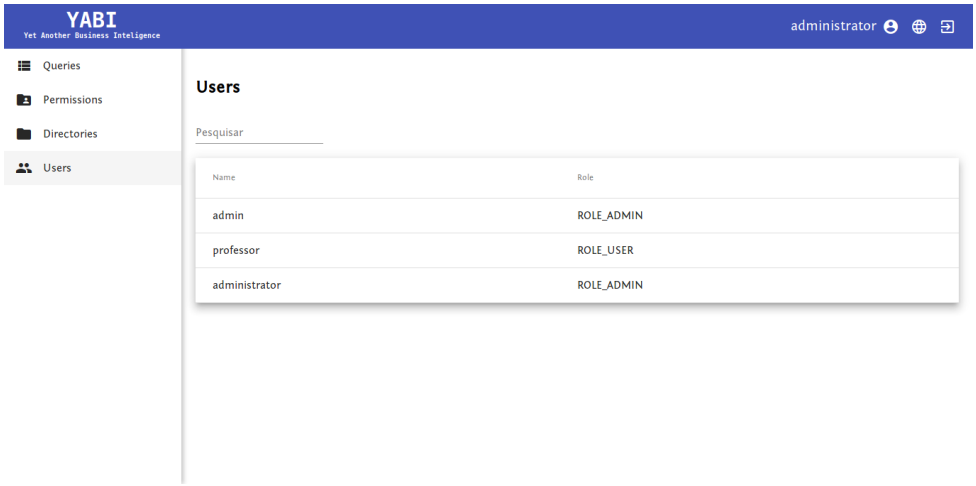

Figure 6.14: Listing of all registered `User`

In regards to ORM, `@Entity` annotation in line 3 is the entry-point through which JPA evaluates what classes are meant to be taken into account when building the relational model. In general, some attributes need don't need to be declared as they are correctly inferred but in some cases it is desired to configure the generated database, `@Column` annotation on line 11 changes the default behavior so that the length of the corresponding `varchar` field in the table is able to hold larger amounts of characters; The other models, `YabiUser` `PermissionTree` and `Directory` make extensive use of `@Column` to specify columns that shouldn't have repeated values.

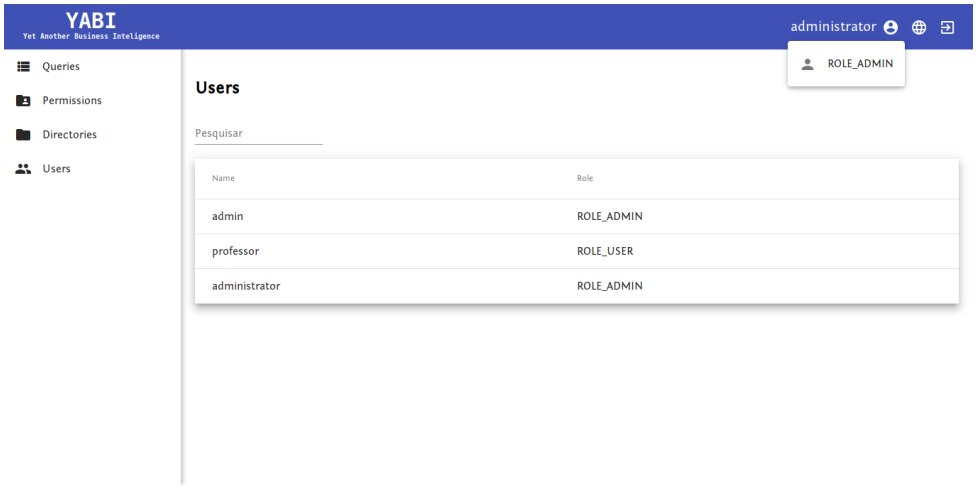

Figure 6.15: Small pop-up showing the current user's role

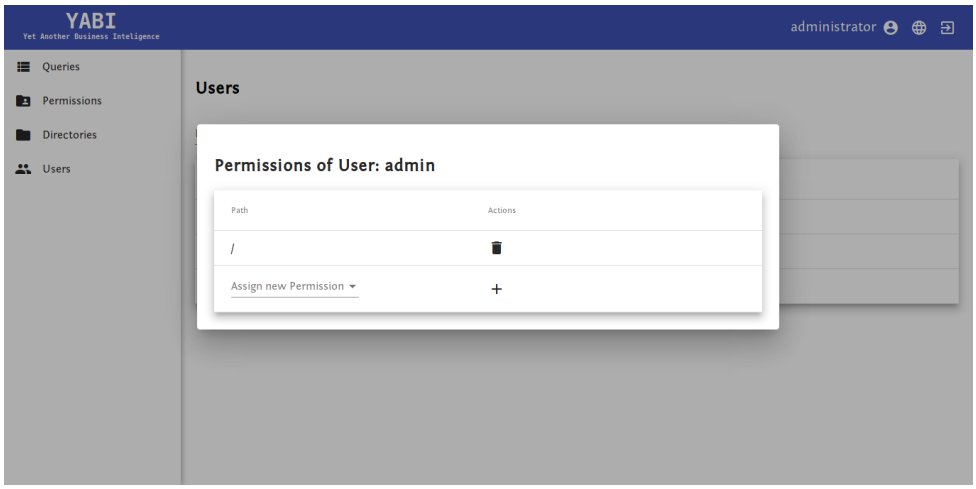

Figure 6.16: Dialog for assigning a new `Permission` to a `User`

Relation between entities are made though `@OneToOne`, `@ManyToOne` and `@ManyToMany` annotations. The first two represents single value association between entities, however, they represent different semantics and where the foreign key will be created. `@ManyToOne` indicates that the foreign key will stay in the table in which this model is mapped to, `@OneToOne` makes no distinction, leaving for the ORM back-end implementation to decide. Line 17 is declaring that more that one instance of `SqlQuery` may reference a single `Directory` and that `SqlQuery` will hold the foreign key to `Directory`. `@ManyToMany` represents a collection of associations, here used to associate `YabiUser` to `PermissionTree`

Listing 6.1: Implementation of the Query model

```
1  @NoArgsConstructor
2  @AllArgsConstructor
3  @Entity
4  @Table(uniqueConstraints = @UniqueConstraint(columnNames = {"permission"
       , "name"}))
5  public @Data class SqlQuery {
6
7      @Id
8      @GeneratedValue
9      private Long id;
10
11     @Column(length = 2048)
12     private String command;
13
14     private String name;
15     private String description;
16
17     @ManyToOne
18     @JoinColumn(name = "directory_id", nullable = false)
19     private Directory directory;
20
21     @OneToOne
22     @JoinColumn(name = "permission", nullable = false)
23     private PermissionTree permission;
24
25 }
```

so that many users can have many, overlapping, permissions. One possible parameter to `@ManyToMany` is the `FetchType`, instructing the ORM engine to retrieve the associated entity only when it is accessed or together with its parent is retrieved.

`@JoinColumn` annotation is a general purpose configuration for relational fields, in line 18 and 22 it's used to configure the name in which the column will be called and whether it can have no specified value.

The remaining entities follow a similar pattern in it's implementation.

6.2.2 Spring Configuration

In order for Spring Framework to stay out of the way as much as it can and allow developers to focus on the implementation of business functionalities, it makes many assumptions

about how its components interact, however, at some point the application being developed grow some needs that conflict with Spring defaults. When this eventually happens, which was the case with Yabi, developers can override some of Spring's default behavior by implementing specific interfaces. More on Spring can be found in Section 4.2.4.

This section presents Spring configurations that took place so that Yabi is able to work as expected.

Security

Yabi's security model uses a directory server to authenticate and a relational database to load user roles and execute authorization checks. Because this is a stateless application, every request follows the steps shown in Figure 6.17 before being executed by the controllers.

Authentication is done through a anonymous bind to a LDAP server, Section 6.2.2 goes through the Spring configuration necessary to make it work and Section 6.2.2 explains how the roles are loaded into the Sprig Security's `UserDetails` object.

In regards to authorization, there are two possible roles in which a use must be assigned to, either `ADMIN` or `USER`. With this, non-administrative resources are simply not dependent on the user's role therefore accessible to all users, meanwhile, administrative resources are explicitly marked to be accessible by users whose role is `ADMIN`, more on this in Section 6.2.2.

Figure 6.17, presents in a general view the steps taken to authenticate and load an user's details. It is infeasible to model the whole of Spring Web and Spring Security as it is quite a extensive feature and it is out of the scope of this report, therefore it was abstracted into fewer elements, namely `WebSecurity` entity representing the objects that gets built by the configuration shown in Listing 6.2, the *authenticate* message is abstracting Spring's authentication provider voting system, `LDAP AuthenticationManager` entity representing LDAP's `AuthenticationProvider` and *anonymous bind*, as the name implies is the authentication that takes place in the directory server.

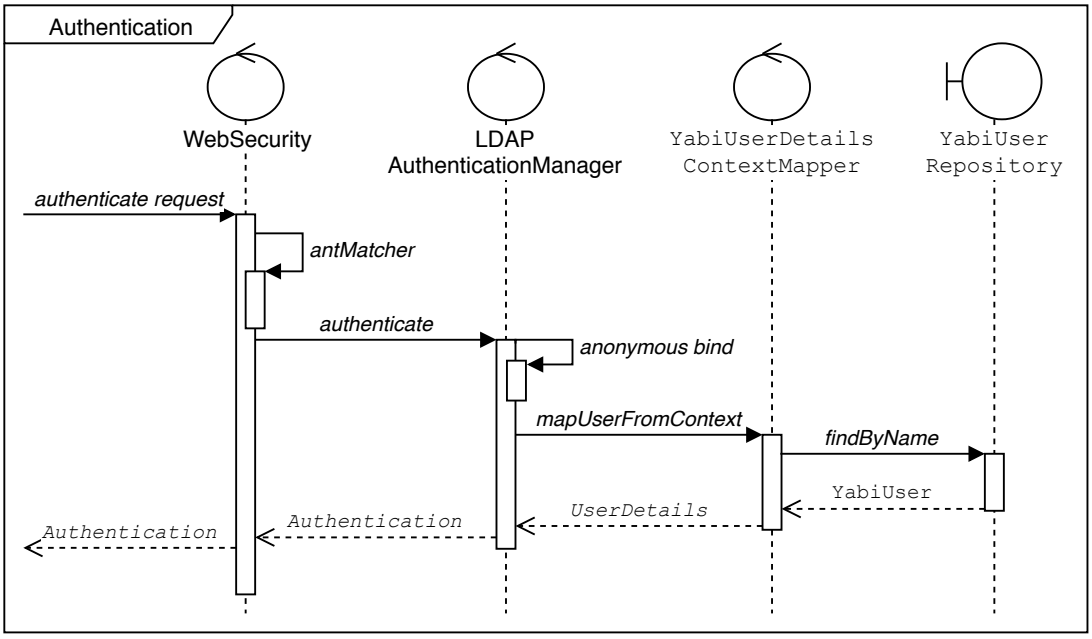

Figure 6.17: Authentication Sequence Diagram

The main point of Figure 6.17 is to show that once the bind takes place in the `LDAP AuthenticationManager`, the library requests an object that implements the `UserDetailsContextMapper` to populate it's newly created `Authentication` object with business-specific information in the form of a `UserDetails`. In this case `YabiUserDetailsManager` is chosen by Spring's dependency injection mechanism and it provides an instance of `YabiUser`, which implements `UserDetails` interface, that is retrieved from Yabi's database.

Admin Resources

Not all endpoints are freely accessed to all users because they involve possibly destructive interactions with the information contained in the system. In this implementation, non-administrative users get their information through custom `RestController` that expose fewer functionalities and administrative users may directly request the repositories. Repositories are further discussed in Section 6.2.4.

Suffice to say that certain endpoints require the user to be authenticated and to have an administrator role. Listing 6.2 is the configuration that enforces this statement.

Listing 6.2: HttpSecurity configuration

```
1       protected void configure(HttpSecurity http) throws Exception {
2           http
3                   .cors()
4                   .and()
5                   .csrf().disable()
6                   .httpBasic()
7                   .and()
8                   .authorizeRequests()
9                   .antMatchers("/user").permitAll()
10                  // Spring Repositories
11                  .antMatchers("/directories/**", "/yabiUsers/**", "/
                        permissionTrees/**", "/sqlQueries/**").hasRole("ADMIN
                        ")
12                  // Custom Controllers
13                  .antMatchers(HttpMethod.DELETE, "/permission/**").
                        hasRole("ADMIN")
14                  .anyRequest().authenticated()
15                  .and()
16                  .logout().permitAll();
17      }
```

The call to `antMatchers` in lines 47, 49 and 51 works by specifying a list of HTTP paths and applying some definitions or restrictions whenever an incoming request is a member of it. TODO: membro da lista de caminhos http . In this specific case, all requests to repositories are eligible to continue only if the `hasRole` definition evaluates to true.

Line 51 is protecting the **/permission** endpoint from being requested with a HTTP DELETE verb. Lines 46 and 52 declare that all HTTP requests are to be authenticated.

TODO: as linhas estao certas ?

Cross-Origin Resource Sharing (CORS) Mapping

Because Yabi is a web API and a front-end application, it is necessary that both parties can interact but due to security reasons, most browsers implementations block Asynchronous JavaScript and XML (AJAX) calls if the remote server does not explicitly add the current domain to it's response header.

Listing 6.3: LDAP Authentication Configuration

```
1       @Autowired
2       public void configureGlobal(AuthenticationManagerBuilder auth,
            YabiUserDetailsContextMapper ap) throws Exception {
3           auth
4                   .ldapAuthentication()
5                   .userDetailsContextMapper(ap)
6                   .userDnPatterns(env.getProperty("yabi.ldap.
                        userDnPatterns"))
7                   .groupSearchBase(env.getProperty("yabi.ldap.
                        groupSearchBase"))
8                   .contextSource()
9                   .url(env.getProperty("yabi.ldap.url"));
10      }
```

In Spring, specifying allowing domains to access it's resources is done by implementing the `WebMvcConfigurer`, overriding the method *addCorsMapping(CorsRegistry)* and calling *allowedOrigins* on it's parameter. The method *allowedOrigins* takes a list of stings that contain valid URL addresses.

Yabi configures this using the `application.properties` file under the key `yabi.web.allowedOrigins`, allowing for a centralized configuration.

LDAP

Following the authentication specification in Section 5.4, Listing 6.3 presents the configuration that implements the desired behavior.

TODO: checar linhas no listing gerado In this configuration, `AuthenticationManagerBuilder` is a class used by Spring in it's security pipeline. It comes with built-in support for LDAP, JDBC and in-memory authentication mechanisms; line TODO: 4 is declaring LDAP authentication to be used.

Line TODO: 5 is considered important because it is mapping a custom details context mapper to the authentication pipeline. What this does is to provide a hook in the authentication pipeline to allow explicit customization of the user object after it is authenticated. The given mapper, `YabiUserDetailsContextMapper` retrieves the authenticated user's instance of `YabiUser`.

Lines `TODO:` 6 to 9 configure the connection to the remote directory service, including it's address and what to bind with.

User Details Context Mapper

Often times a directory service is used as an authentication mechanism. Applications issue an anonymous bind request to the server passing their user's credentials and if properly found and matched, a boolean value is returned, however, applications often has information about the user that must be sent accessible to other parts of the framework. To do so, Spring Security utilizes this interface to retrieve a `UserDetails` instance that gets injected into the commonly accessible `Authentication` interface.

For this application, a new implementation of the `UserDetailsContextMapper` interface is provided, `YabiUserDetailsContextMapper` returns an instance of `YabiUser`, which implements said `UserDetails` interface and adds Yabi-specific attributes, enabling other parts of the system to query the current user's related information such as their role, `PermissionTree` and name. More information about the `YabiUser` class is found in Section 5.3.4.

6.2.3 Custom Controllers & View Models

`RestController` is a Spring Web annotation that enables a given class or method to handle HTTP requests. In essence it is a combination of two other annotations, the `Controller`, which is what trigger the framework into considering the class as a possible resolver of HTTP requests and `ReponseBody`, that wraps the method call into a response body. In simple cases the returned Object is mapped to a Javascript Object Notation (JSON) string.

Because the `PermissionTree` class contains a reference to it's parent, and the root references itself, there was a need to circumvent a infinite loop during it's JSON serialization. To do so, rather simple and serializable classes whose role were to convey information

Listing 6.4: **/queries** Endpoint

```
1       @CrossOrigin
2       @GetMapping("/queries")
3       public List<SqlQueryViewModel> getQueries(Authentication auth){
4           YabiUser user = (YabiUser) auth.getPrincipal();
5           List<SqlQueryViewModel> queries = new ArrayList<>();
6
7           for ( PermissionTree permission : user.getPermissions() ){
8               for ( SqlQuery q : queryRepo.
                    findByPermissionNodePathStartingWith(permission.
                    getNodePath()) ){
9                   queries.add( new SqlQueryViewModel(q) );
10              }
11          }
12          return queries;
13      }
```

to the front-end were written, namely, `SqlQueryViewModel`, `PermissionTreeViewModel` and `YabiUserViewModel`.

To accommodate the special handling of `PermissionTree` model and provide some custom functionalities some custom `RestController` were implemented, `SqlQueryController`, `PermissionTreeController` `YabiUserController` and `DatabaseReaderController`.

`TODO:` isso esta errado, sql query é filtrado pelo metodo do controller findByPermissionNodePathStartingWith `TODO:` Filtering in relation to `Permission` was required when providing `SqlQuery` and `PermissionTree` objects to non-administrative users, it was implemented by iterating over the current user's permissions and appending all the elements under that permission to a list. Listing 6.4 presents the implementation of such filtering applied to `SqlQuery` model.

DatabaseReaderController

`TODO:` esta ruim `TODO:` qual eh o path que ele escuta, qual eh o argumento que ele recebe ?, como ele faz a checagem de permissao ? The retrieval of information contained in a remote database is exposed through the `DatabaseReaderController`. It's a wrapper to `DatabaseReader.runQuery` method that does a permission check before executing.

PermissionTreeController

Because of it's tree-like nature, once a `PremissionTree` is deleted, all of it's child nodes need also to be removed. However, because it reference its parent but not its children, leaving the RDBMS to execute a cascading delete would delete every parent permission untill the root node. Therefore `PermissionTreeController` implements a custom a delete that cascades through its children. It is bound to a DELETE **/pemrission/{id}**, `id` being the identifier of the permission to be deleted.

It is implemented by first retrieving the permission whose id was specifies in the HTTP request, retrieve all of its children with the custom repository method *findAllByNodePathStartingWith* and sequentially deleting all permissions found and lastly delete the permission itself.

One restriction is that the root node can never be deleted. Therefore, before executing the before-mentioned steps, the permission to be deleted is matched with the root node, if it does, the operation is aborted with an error.

YabiUserController

To provide the front-end with information about the current user, `YabiUserController` replies to GET requests on the path **/user** with information contained inside the `Authentication` that was generated during the user's authentication procedure and thus avoid reaching out to the database a second time.

SqlQueryController

`SqlQuery` is one of those entities that are related to the current logged-in user. In other words, administrators may see all registered `SqlQuery` and users see only those which they can run, Therefore `SqlQueryController` was created.

It has only one method that replies to GET requests to **/queries** endpoint with a list of `SqlQueryViewModel`. Implementation-wise it returns a list containing every `SqlQuery`

found in the database by calling `SqlQueryRepository.findAllByNodePathStartingWith` for every permission associated to the current `YabiUser`.

6.2.4 Spring Repositories

`PagingAndSortingRepository` were created for all entities evaluated during the project evaluation phase. The `Directory` entity whose did not require any changes in regards to what Spring already provides and won't be discussed. The remaining entities had their repositories augmented with functionalities presented below.

YabiUserRepository

After binding in the directory service, `UserDetailsContextMapper.mapUserFromContext` is used to fill an instance of `Authentication` class with business data. In Yabi's custom implementation, this data comes from a relational database that reflect `YabiUser` objects.

Because the method *mapUserFromContext* uses the username as a key to retrieve information, `YabiUserRepository` had to be augmented with a new method to do just that. Therefore the following declaration was added:

```
YabiUser findByName(String username);
```

PermissionTreeRepository

When the system needs to validate an action or filter information depending on a permission, it retrieves all of its child permissions. Because this action is frequently used, this repository had also to be augmented.

In this case, the method signature used was as follows:

```
List<PermissionTree> findAllByNodePathStartingWith(String nodePath);
```

SqlQueryRepository

When an user request a list of queries that they can execute, the system must retrieve from the relational database all queries in which the permission is a child of the user's permission. Again, this repository had to be augmented.

This was accomplished by using the following method interface:

```
List<SqlQuery> findByPermissionNodePathStartingWith( String nodePath );
```

6.2.5 Multi-Database Support

Paraphrasing Requirement 1, the application must be able to retrieve information from the institution's database. However, because it has many in-house applications, they might use different RDBMS and Yabi should then be able to connect to then.

The core part of this feature is JDBC 4.0's `DriverManager` class and it's *getConnection* method that upon being called, attempts to make a connection using drivers that were loaded on initialization-time, therefore, as long as the driver is loaded and the connection string is properly formed, *getConnection* will select the correct database driver.

Because of this, implementing this feature was as simple as declaring dependencies for database drivers in the `pom.xml` configuration file. Notably, Oracle[1] requires the creatin of an Oracle account and configuring a custom maven repository in order to download their drivers.

6.3 Development Environment

This section will focus on the tools that were configured to accommodate the development and testing of this application.

[1] https://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html

Listing 6.5: Local server LDAP configuration

```
1 yabi.ldap.url=ldap://localhost:10389/dc=ipb,dc=pt
2 yabi.ldap.groupSearchBase=ou=groups
3 yabi.ldap.userDnPatterns=uid={0},ou=users
```

6.3.1 Directory Service

When developing an application, it is good practice to avoid reaching out and interacting to remote services and instead provide a local instance that is able to mimic the real-world one.

In this case because a directory service is used as a part of it's authentication mechanism, a local LDAP server was created in Apache Directory Studio that mimics IPB's directory service enough so that the configuration provided by their Information Technology (IT) team is able to be used locally with minimal changes, in fact, the only difference is the service's address.

Listing 6.5 exposes the configurations Yabi uses to access the server, note that the directory address is declared in line 1 and line 3 declare the entry whose elements are anonymously bound.

Figure 6.18 show how the directory is structured in the server-side. Users are of class `inetOrgPerson`, they are held under the `users` organizational unit which in turn is under the `ipb`, `pt` domain component. At the moment this figure was taken, passwords were stored in plain text for local testing purposes.

6.3.2 Database Initializer

When starting Yabi for the first time, it needs to generate its database, create the root permission and an administrator account. To do so, the class `DbInitializer` was written. It implements the `CommandLineRunner` interface so that Spring instantiate and runs it upon initialization.

There are two properties that interact with `DbInitializer`, `yabi.db.init` and `yabi.db.init.admin.u`
The former regulates when yabi should create a new database, the root permission and

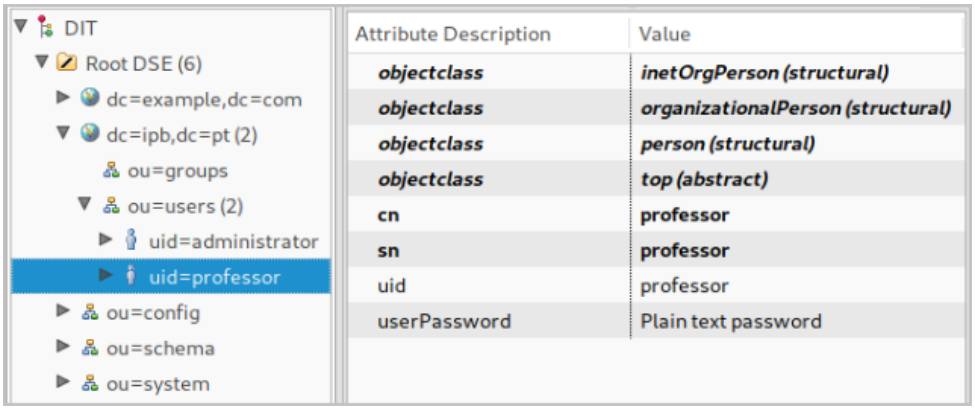

Figure 6.18: Directory structure and the properties of user `professor`

the administrator account, the latter declares the administrator's username that should be used. If the initialization is desired, `yabi.db.init` should be set to `create`, otherwise, it should be set to anything else. It is necessary that the administrator username is able to be bound in the directory service otherwise the authentication will fail.

6.3.3 Postman Tests

When developing the API, some test cases were created in Postman to assess the prevention of data duplication. All four entities were tested. In essence, every test consists of two requests, one that creates a new entity and expects a HTTP status 201 response and the other that tries to re-create the same entity and expects a HTTP status 409.

It is important to note that these tests are validating the following restrictions imposed in each entity:

- There must be only one `PermissionTree` per `nodePath`.
- One `Directory` per `connectionString` so that each database is referenced once.
- One `Directory` per `name` so that each name maps to only one database.
- One `SqlQuery` under a `PermissionTree` with a given `name`, avoiding ambiguous `SqlQuery` entries.
- No more than one `YabiUser` with a given `name`.

6.3.4 Conclusion

This section discussed the implementation details all the parts that compose the Yabi application and its development. **TODO:** Bruno! o que colocar na coclusao ?

Chapter 7

Conclusion

TODO: a associacao com o pai em PermissionTree nao vale o problemas que trouxe. Ela nao eh usada em nada e por causa disso tive de escrever view-models e controllers cutomizados. As manipulacoes tiveram de usar o nodePath de qualquer forma.

Chapter 8

Future Work

8.1 Code Re-structure

8.1.1 Administrative Resources

Currently, resource filtering is implemented by having a small set of controllers that are eligible to handle non-administrative users however this led to some confusion because its response structure does not follow the HATEOAS convention.

Some re-structuring could be done at the repository level by configuring the authorization mechanism to filter HTTP methods based on the current user's requesting role.

Ultimately, all resources should follow the HATEOAS pattern so that the complexity of the front-end application is lower.

8.1.2 Bullk information manager

TODO:

Bibliography

[1] S. J. Berman, "Digital transformation: Opportunities to create new business models", *Strategy & Leadership*, vol. 40, no. 2, pp. 16–24, 2012. DOI: 10.1108/10878571211209314. eprint: https://doi.org/10.1108/10878571211209314. [Online]. Available: https://doi.org/10.1108/10878571211209314.

[2] A. D. Project, *Architecting the modern ldap renaissance: The apache directory vision*, http://directory.apache.org/vision.html, May 2019.

[3] R. Ramakrishnan and J. Gehrke, *Database Management Systems*, 2nd. New York, NY, USA: McGraw-Hill, Inc., 2000, ISBN: 0072440422.

[4] typescriptlang team, *Typescript website*, http://www.typescriptlang.org, May 2019.

[5] D. Maharry and T. Meister, *Typescript revealed*, 1st. Apress, lda, 2013, ISBN: 978-1-4302-5725-7.

[6] w3c, *Html 5.2, w3c recommendation*, https://www.w3.org/TR/html52/introduction.html, Dec. 2017.

[7] w3c, *Html & css*, https://www.w3.org/standards/webdesign/htmlcss, Dec. 2017.

[8] J. Anne and N. Weizenbaum, *Sass documentation*, https://sass-lang.com/documentation, May 2019.

[9] A. Team, *Angularjs homepage*, https://angularjs.org/, May 2019.

[10] S. K. Kasagoni, *Building Modern Web Applications Using Angular*, 1st. Packt Publishing Ltd., 2017, ISBN: 978-1-78588-072-8.

[11] A. Team, *Introduction to modules*, `https://angular.io/guide/architecture-modules`, May 2019.

[12] I. G. Clifton, *Android User Interface Design*, 2nd. Pearson Education Inc., 2016, ISBN: 978-0-134-19140-9.

[13] A. M. Team, *Angular material homepage*, `https://material.angular.io/`, May 2019.

[14] F. Martino and N. K. Mishra, *Sb admin material*, `https://github.com/start-javascript/sb-admin-material`, May 2019.

[15] D. Miller, Kouceyla, A. Kumar, and B. Clees, *Sb material*, `https://github.com/BlackrockDigital/startbootstrap-sb-admin`, May 2019.

[16] J. Gosling, B. Joy, G. Steele, and G. Bracha, *The Java language specification*. Addison-Wesley Professional, 2000.

[17] G. Zoric, K. Smid, and I. S. P, "Design patterns: Elements of reusable object-oriented", *Software', Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, Addison-Wesley*, 1995.

[18] J. Gossman, *Introduction to model/view/viewmodel patter for building wpf apps*, `https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/`, May 2019.

[19] S. F. Team, *Spring docs*, `https://docs.spring.io/spring/docs/5.1.7.RELEASE/spring-framework-reference/overview.html`, May 2019.

[20] M. Fowler, *Inversion of control containers and the dependency injection pattern*, `https://martinfowler.com/articles/injection.html`, May 2019.

[21] S. F. Team, *Spring docs dependecies*, `https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/core.html#beans-dependencies`, May 2019.

[22] S. F. Team, *Spring data*, https://spring.io/projects/spring-data, May 2019.

[23] S. F. Team, *Working with spring data repositories*, https://docs.spring.io/spring-data/data-commons/docs/1.6.1.RELEASE/reference/html/repositories.html, May 2019.

[24] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture", *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002, ISSN: 1533-5399. DOI: 10.1145/514183.514185. [Online]. Available: http://doi.acm.org/10.1145/514183.514185.

[25] M. Webster, *Merriam webster online dictionary*, https://www.merriam-webster.com, May 2019.

[26] S. Atkinson, *Why i hate spring*, http://samatkinson.com/why-i-hate-spring/, May 2019.

[27] L. Gupta, *13 spring best practices for writing configuration files*, https://howtodoinjava.com/best-practices/13-best-practices-for-writing-spring-configuration-files/, May 2019.

[28] U. Peter, *Configuration-hell remedy with singleton injection*, https://xebia.com/blog/configuration-hell-remedy-with-singleton-injection/, May 2019.

[29] B. Java, *Programming over convention over configuration?*, https://www.beyondjava.net/programming-over-convention-over-configuration, May 2019.

[30] R. Hightower, *Spring xml hell? escape xml hell*, https://dzone.com/articles/draft-spring-xml-hell, May 2019.

[31] P. MAVRO, *MariaDB High Performance*. Packt Publishing, Sep. 2014, ISBN: 978-1-78398-160-1.

[32] D. Bartholomew, "Mariadb vs. mysql", *MariaDB White Paper*, Sep. 2012.

[33] MariaDB, *Mariadb: Knowledge base*, https://mariadb.com/kb/en/, May 2019.

[34] E. J. Sermersheim, *Lightweight directory access protocol (ldap): The protocol*, https://tools.ietf.org/html/rfc4511, Jun. 2006.

[35] T. S. S. O. I. T. Union, *Information technology – Open Systems Interconnection –The Directory: Overview of concepts, models and services*. International Telecomunication Union, Oct. 2016.

[36] A. D. Foundation, *Apache directory service homepage*, `https://directory.apache.org/apacheds/`, Aug. 2018.

[37] A. D. Foundation, *Apache directory studio homepage*, `https://directory.apache.org/studio/`, Aug. 2018.

[38] Y. Poirier, *Announcing 2018 duke's choice award winners*, `https://blogs.oracle.com/java/announcing-2018-dukes-choice-award-winners`, Nov. 2018.

[39] A. S. Foundation, *Netbeans project incubation status*, `http://incubator.apache.org/projects/netbeans.html`, May 2019.

[40] Sally, *The apache software foundation announces apache netbeans as a top-level project*, `https://blogs.apache.org/foundation/entry/the-apache-software-foundation-announces51`, Apr. 2019.

[41] geertjanw, vieiro, JeremyCavanagh, and cunka, *Code assistance in the netbeans ide java editor: A reference guide*, `http://netbeans.apache.org/kb/docs/java/editor-codereference.html`, May 2019.

[42] geertjanw, vieiro, omerhakanbilici, and emilianbold, *Getting help*, `http://netbeans.apache.org/help/index.html`, May 2019.

[43] M. Team, *Apache maven project*, `http://maven.apache.org/what-is-maven.html`, May 2019.

[44] M. Team, *Maven in 5 minutes*, `https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html`, May 2019.

[45] T. P. L. Authors, *Project lombok*, `https://projectlombok.org/`, May 2019.

[46] S. Team, *2019 stackoverflow survey*, `https://insights.stackoverflow.com/survey/2019#technology-_-most-popular-development-environments`, May 2019.

[47] M. V. S. C. Team, *Visual studio code docs*, https://code.visualstudio.com/Docs, May 2019.

[48] V. C. Team, *A commom protocol for languages*, https://code.visualstudio.com/blogs/2016/06/27/common-language-protocol, Jun. 2016.

[49] C.-D. Document, *Language server protol specification*, https://github.com/microsoft/language-server-protocol/blob/gh-pages/specification.md, May 2019.

[50] D. Inc, *Docker faq*, https://docs.docker.com/engine/faq/, May 2019.

[51] D. Inc, *Docker overiew*, https://docs.docker.com/engine/docker-overview/, May 2019.

[52] D. Inc, *Dockerfile reference*, https://docs.docker.com/engine/reference/builder/, May 2019.

[53] D. Inc, *Docker compose overview*, https://docs.docker.com/compose/overview/, May 2019.

[54] S. Org, *Chinook database analysis*, schemaspy.org/sample/, May 2019.

[55] Lerocha, *Chinook database*, https://github.com/lerocha/chinook-database, May 2019.

[56] A. Team, *Angular cli*, https://angular.io/cli, May 2019.

[57] M. Foundation, *We're building a better internet*, https://www.mozilla.org/en-US/mission/, May 2019.

[58] M. Community, *Use a source map*, https://developer.mozilla.org/en-US/docs/Tools/Debugger/How_to/Use_a_source_map, Mar. 2019.

[59] jlaster and H. Kirschner, *Debugging modern web applications*, https://hacks.mozilla.org/2018/05/debugging-modern-web-applications/, May 2018.

[60] M. Community, *Network monitor*, https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor, Mar. 2019.

[61] M. Community, *Storage inspector*, https://developer.mozilla.org/en-US/docs/Tools/Storage_Inspector, Mar. 2019.

[62] M. Community, *Page inspector*, https://developer.mozilla.org/en-US/docs/Tools/Page_Inspector, Mar. 2019.

[63] C. Written, *Concepts*, https://webpack.js.org/concepts, May 2019.

[64] P. Teamn, *Sending the first request*, https://learning.getpostman.com/docs/postman/launching_postman/sending_the_first_request, May 2019.

Appendix A

Proposta Original do Projeto

A.1 Proposta nº 2

Pretende-se desenhar e construir de raíz um sistema de "business inteligence" aplicado à gestão letiva. Sabemos bem o valor e a importância que a informação tem hoje em dia para quem gere instituições e as mais valias que as ferramentas de análise de dados trazem para a tomada de medidas e decisões. O IPB tem já uma base de dados centralizada à qual é aplicado diariamente um grande número de queries em SQL para os mais diversos fins. Pretende-se abrir o acesso a esta informação de forma criteriosa mas sem implicar a escrita manual de queries muitas delas com mais de 30 linhas de código.

O sistema seria pré-alimentado com "clusters" de queries mas teria uma característica evolutiva que daria a possibilidade de introduzir de forma fácil, suportada e validada novas queries ou novos grupos de queries, dependendo do perfil do utilizador.

As palavras chave serão a reutilização e a parametrização automática desses grupos de queries suportadas pela geração automática de interfaces web de pesquisa de informação, tendo por base o tipo de query a implementar.

Trata-se da disponibilização de um sistema de consulta inteligente no sentido em que se adapta às necessidades e ao perfil de cada utilizador. O resultado final serão sempre tabelas exportáveis para os mais variados formatos.