



My Career

B.Sc.(Hons) in Software Development

Alexander Souza

G00317835@GMIT.IE

Supervisor

Damien Costello

Submission Date 16th April 2018

GitHub

https://github.com/alexpt2000gmit/4Year_MainProject_MyCareer



Applicant Tracking System

<http://54.210.4.37/mycareer/>

MyCareer Applicant Tracking Simplified

Welcome to MyCareer
Better recruiting comes from better processes.

Positions Available

Title	Position	Location	Publication	Apply Now
Senior Software Engineer Developer - Samples	Developer	Galway	23/01/2018	+ Apply Now
Mid-Level Front End Developer - Samples	Developer	Dublin	10/01/2018	+ Apply Now
Graduate Software engineer - Developer Role	Developer	Galway	23/01/2017	+ Apply Now
Senior JavaScript Developer	Developer	Galway	20/10/2017	+ Apply Now
Sales Development Representative	Sales	Galway	20/01/2018	+ Apply Now
SQL Developer	Developer	Galway	01/02/2018	+ Apply Now
Software Engineer III	Developer	Dublin	23/01/2018	+ Apply Now
C/C++ Software Engineer	Developer	Dublin	01/02/2018	+ Apply Now
C# Software Engineer	Developer	Dublin	01/02/2018	+ Apply Now
Design & Development Engineer	Developer	Dublin	10/02/2018	+ Apply Now
Senior C# Developer	Developer	Galway	19/02/2018	+ Apply Now
Senior Front End Developer Javascript	Developer	Galway	19/02/2018	+ Apply Now
Senior Front End Developer	Developer	Galway	19/02/2018	+ Apply Now
Test Development Engineer	Developer	Galway	19/02/2018	+ Apply Now

Sign Up for Alerts
Join our talent community and get MyCareer news and job alerts delivered to your inbox.
Email: [Sign Up](#)

Alexander Souza

Title: Senior Software Engineer Developer - Samples Position: Developer Location: Galway

Phone: 0877487528 Status: Did not pass Date: 29/01/2018
email: alexsp2009@gmail.com Profile URL: https://www.w3schools.com/

Questions

Subject	Question	Rating	Answer
Java	What is Bytecode?	★★★★★	<input checked="" type="checkbox"/>
Java	What is the difference between JVM and JRE?	★★★★★	<input checked="" type="checkbox"/>
HR Interview	What are your strengths?	★★★★★	<input checked="" type="checkbox"/>
HR Interview	Tell me about yourself	★★★★★	<input checked="" type="checkbox"/>
HR Interview	Why should we hire you?	★★★★★	<input checked="" type="checkbox"/>
HR Interview	Where do you see yourself in five years time?	★★★★★	<input checked="" type="checkbox"/>

Table of Contents

1	Introduction.....	7
1.1	The Idea.....	7
1.2	The Application	7
1.3	Project Scope	7
1.4	Summary	7
2	Methodology	8
2.1	Planning.....	8
2.2	Methodology.....	9
2.3	Project Management	9
2.3.1	GitHub	9
3	Technology Review.....	10
3.1	Java	10
3.2	Hibernate	10
3.3	JWT.....	10
3.4	Angular 4.....	11
3.5	Joomla	11
3.6	MySQL.....	11
3.7	Heroku.....	12
3.8	Amazon AWS.....	12
3.9	Spring Framework	12
3.10	Flyway.....	13
3.11	Postman	13
3.12	Swagger UI.....	14
3.13	GitHub.....	14
3.14	PrimeNG	14
3.15	Node.js	15
4	System Design.....	16
4.1	System Architecture.....	16
4.1.1	Website.....	16
4.1.2	Frontend (Applicants)	18
4.1.3	Backend (Applicants).....	18
4.1.4	Security	18
4.1.4.1	JSON Web Tokens	18
4.1.5	API	19
4.1.6	Database	19
4.2	System Design	20
4.3	REST.....	20
4.3.1	Architectural constraints	20
4.3.2	HTTP methods.....	21

4.3.3	HTTP status codes	21
4.3.4	Design Patterns	22
4.4	System Deployment	22
4.4.1	Deploy Joomla (AWS)	23
4.4.2	Deploy Angular (Heroku)	24
4.4.3	Deploy API (Heroku)	24
5	System Evaluation	29
5.1	Robustness & Efficiency	29
5.1.1	Tomcat	29
5.1.2	MongoDB	29
5.2	Space / Time Complexity	29
5.3	Security and Validation	29
5.4	Deliverable Software Analysis	31
5.4.1	Limits of the system.	31
6	Conclusion	32
6.1	Summary	32
6.2	Learning Outcomes	32
6.3	Reflection	32
7	References	33
	References	33
8	Appendices	34

Table of Illustrations

Fig. 1. Initial page of Project	7
Fig. 2. Slides for presentation of project.....	8
Fig. 3. Microsoft Access, planning for database	9
Fig. 4. System Architecture	16
Fig. 5. Website – frontend	17
Fig. 6. Website – backend.....	17
Fig. 7. JWT requesting JSON web token	18
Fig. 8. UML API	19
Fig. 9. Design database.....	19
Fig. 10. Basic Rest.....	20
Fig. 11. Deploy Joomla website - Configuration	23
Fig. 12. Deploy Joomla website - Database.....	23
Fig. 13. Deploy Joomla website - Overview	24
Fig. 14. Deploy Joomla website - Congratulations	24
Fig. 15. Login Heroku	25
Fig. 16. Create a new app on Heroku.....	25
Fig. 17. Add MySQL to the API.....	25
Fig. 18. JawsDB – Settings.....	26
Fig. 19. Database profile on API.....	26
Fig. 20. Deploy – Commit API and Push to master	27
Fig. 21. Deploy process	27
Fig. 22. Heroku print the log.....	28
Fig. 23. Add Authorization on Headers to request token	29
Fig. 24. Decode Authorization value.....	29
Fig. 25. Add body, key and value to request token.....	30
Fig. 26. Get JSON Web Token with Postman	30
Fig. 27. Decode JSON web Token with JWT.....	31
Fig. 28. Request applicants	31

Abstract.

1 Introduction

1.1 The Idea



Fig. 1. Initial page of Project

1.2 The Application

1.3 Project Scope

As this is a 4th and final year software development project, it was felt that this project would demonstrate many learning outcomes.

1.4 Summary

Throughout this report, several different topics are going to be discussed, ranging from how the project was started and how requirements were gathered, to the actual final product that is being delivered. For clarity, here's a synopsis of what each chapter is about.

2 Methodology

2.1 Planning

Before any project commences, it's quintessential that the proper planning and organization is done to ensure as smooth a development process as possible.

The first port of call before starting was to write down the various high-level components that are involved in a software development environment.

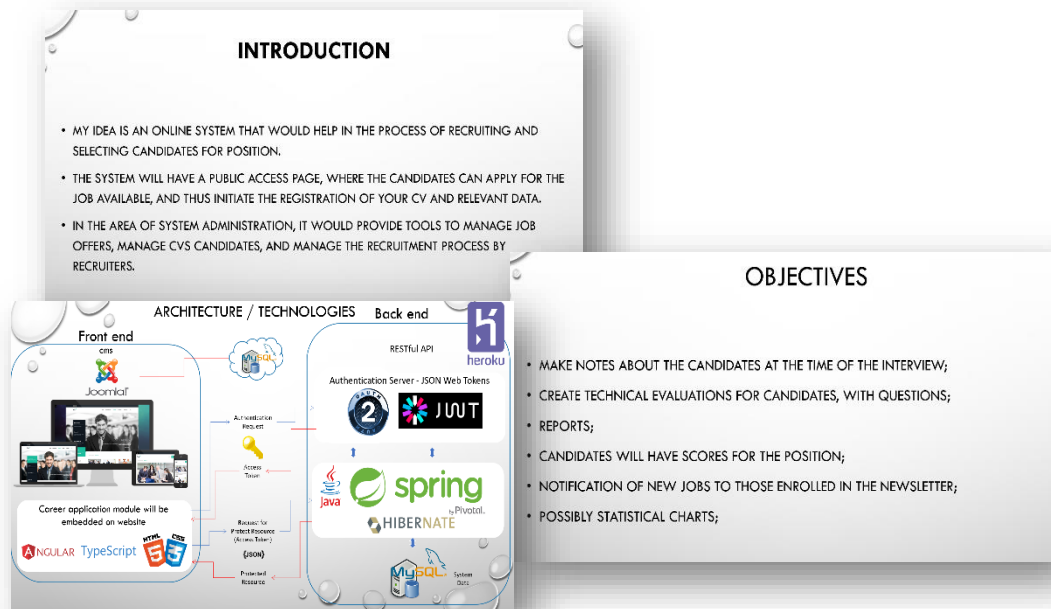


Fig. 2. Slides for presentation of project.

These are the key parts of the Software Development process, that were focused on.

- **Project Management** – Pivotal in any project and allowed the team to manage time and efforts in the most efficient manner.
- **Architecture** – Relates to the overall abstract view of the system.
- **Technology** – Looked at the different technologies that were used as part of the development process, for example, programming languages and IDE's (Integrated Development Environments).

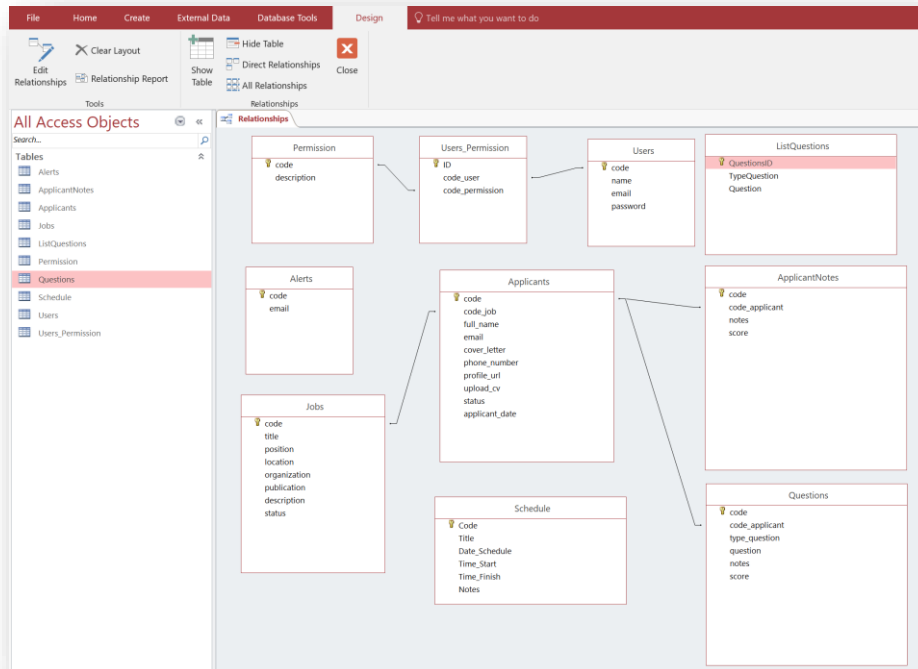


Fig. 3. Microsoft Access, planning for database

2.2 Methodology

As students, the feeling was that the best way to do this was by using flavours of the Agile methodology and its associated subsets such as RAD (Rapid Application Development), Kanban and XP Programming.

This route was chosen as fundamentally, Agile offers a prototype based, iterative approach which not only allowed for rapid development but introduced a loose structure that suited the team's development skillsets.

2.3 Project Management

2.3.1 GitHub.

3 Technology Review

MyCareer is an enterprise web application and several technologies were used to build the applications;

3.1 Java

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture.



3.2 Hibernate

Hibernate ORM (Hibernate in short) is an object-relational mapping tool for the Java programming language. It provides a framework for mapping an object-oriented domain model to a relational database. Hibernate handles object-relational impedance mismatch problems by replacing direct, persistent database accesses with high-level object handling functions.



<http://hibernate.org/orm/>

3.3 JWT

JSON Web Token (JWT) is a compact token format intended for space constrained environments such as HTTP Authorization headers and URI query parameters. JWTs encode claims to be transmitted as a JSON object that is base64url encoded and digitally signed and/or encrypted. Signing is accomplished using JSON Web Signature (JWS). Encryption is accomplished using JSON Web Encryption (JWE).



<https://jwt.io/>

3.4 Angular 4

Angular (commonly referred to as "Angular 4" or "Angular 2") is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite of the same team that built AngularJS.



<https://angular.io/>

3.5 Joomla

Joomla! is a free and open-source content management system (CMS) for publishing web content. Over the years Joomla! has won several awards. It is built on a model–view–controller web application framework that can be used independently of the CMS that allows you to build powerful online applications.

Joomla! is one of the most popular website software, thanks to its global community of developers and volunteers, who make sure the platform is user-friendly, extendable, multilingual, accessible, responsive, search engine optimized and so much more.

Joomla! can be used for:

- Corporate websites or portals, intranets and extranets
- Small business websites
- Online magazines, newspapers, and publications
- E-commerce and online reservations
- Government, non-profit and organisational websites
- Community-based, school and church websites or portals
- Personal or family homepages ...



<https://www.joomla.org/>

3.6 MySQL

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned

and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available and offer additional functionality.

MySQL is a central component of the LAMP open-source web application software stack (and other "AMP" stacks). LAMP is an acronym for "Linux, Apache, MySQL, Perl/PHP/Python". Applications that use the MySQL database include TYPO3, MODx, Joomla, WordPress, Simple Machines Forum, phpBB, MyBB, and Drupal. MySQL is also used in many high-profile, large-scale websites, including Google, Facebook, Twitter, Flickr, and YouTube.



3.7 Heroku

Heroku is a cloud platform as a service (PaaS) supporting several programming languages that is used as a web application deployment model. Heroku, one of the first cloud platforms has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages.



3.8 Amazon AWS

Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers, available all the time, through the Internet.



3.9 Spring Framework

The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE (Enterprise Edition) platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an addition to, or even replacement for the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.



<https://spring.io/>

3.10 Flyway

Flyway is an open source database migration tool. It strongly favors simplicity and convention over configuration.

It is based around 7 basic commands: Migrate, Clean, Info, Validate, Undo, Baseline and Repair.

Migrations can be written in SQL (database-specific syntax (such as PL/SQL, T-SQL, ...) is supported) or Java (for advanced data transformations or dealing with LOBs).

It has a Command-line client, a Java API (also works on Android) for migrating the database on application startup, a Maven plugin and a Gradle plugin.

Plugins are available for Spring Boot, Dropwizard, Grails, Play, SBT, Ant, Griffon, Grunt, Ninja and more.

Supported databases are Oracle, SQL Server, DB2, MySQL (including Amazon RDS), MariaDB, PostgreSQL (including Amazon RDS and Heroku), CockroachDB, Redshift, H2, Hsql, Derby, SQLite, SAP HANA, Sybase ASE and Phoenix.



3.11 Postman

An API Development Environment - or ADE - is a platform that supports and enhances API development. A good ADE will streamline the development process, create a single source of truth for an organization's APIs, and enhance collaboration on APIs across the organization.



<https://www.getpostman.com/>

3.12 Swagger UI

Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from your Swagger specification, with the visual documentation making it easy for back end implementation and client side consumption.



<https://swagger.io/>

3.13 GitHub

GitHub (originally known as Logical Awesome LLC) is a web-based hosting service for version control using git. It is mostly used for computer code. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

GitHub offers plans for both private repositories and free accounts which are commonly used to host open-source software projects. As of April 2017, GitHub reports having almost 20 million users and 57 million repositories, making it the largest host of source code in the world.



<https://github.com/>

3.14 PrimeNG

PrimeNG is a collection of rich UI components for Angular. All widgets are open source and free to use under MIT License. PrimeNG is developed by PrimeTek Informatics, a vendor with years of expertise in developing open source UI solutions.



<https://www.primefaces.org/primeng/>

3.15 Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side. Historically, JavaScript was used primarily for client-side scripting, in which scripts written in JavaScript are embedded in a webpage's HTML, to be run client-side by a JavaScript engine in the user's web browser. Node.js enables JavaScript to be used for server-side scripting and runs scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.



<https://nodejs.org/>

4 System Design

4.1 System Architecture

The MyCareer application has been completely created from the ground up using a methodical and structured approach.

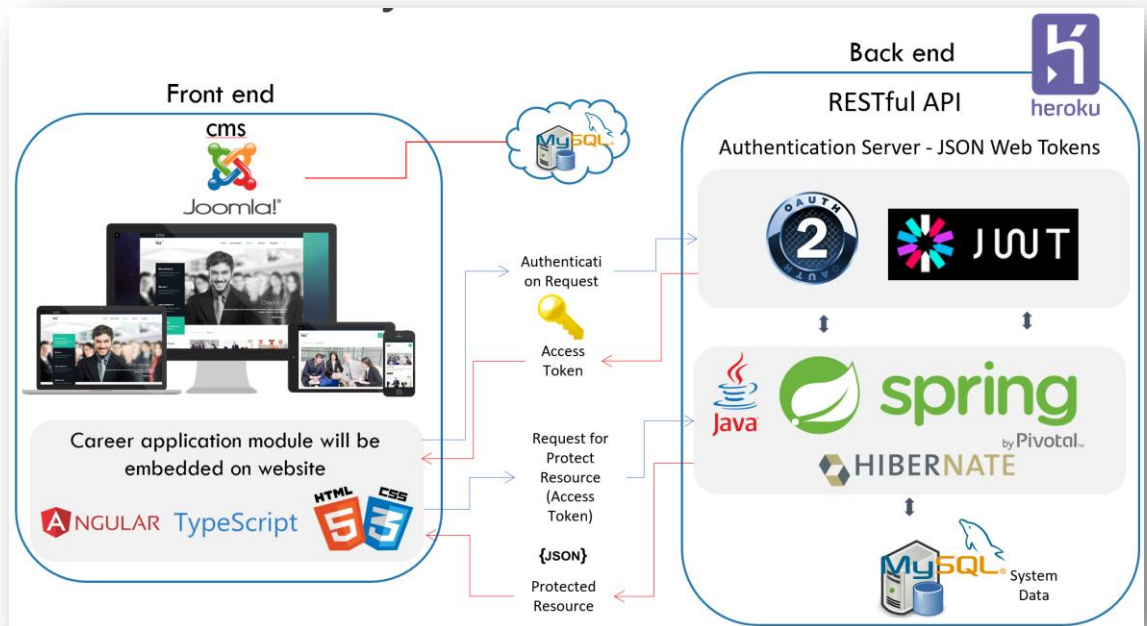


Fig. 4. System Architecture

4.1.1 Website

To offer a better user experience, a website has been implemented to better represent the use of the MyCareer system.

The system also demonstrates how the system can also be used on the company's current website, in which case it has been used by CMS Joomla, a powerful web site development tool that allows users to manage website content from a backend.



Fig. 5. Website – frontend

The backend or administration area of the website allows users to be able to maintain content, menus, images, videos or change the template.

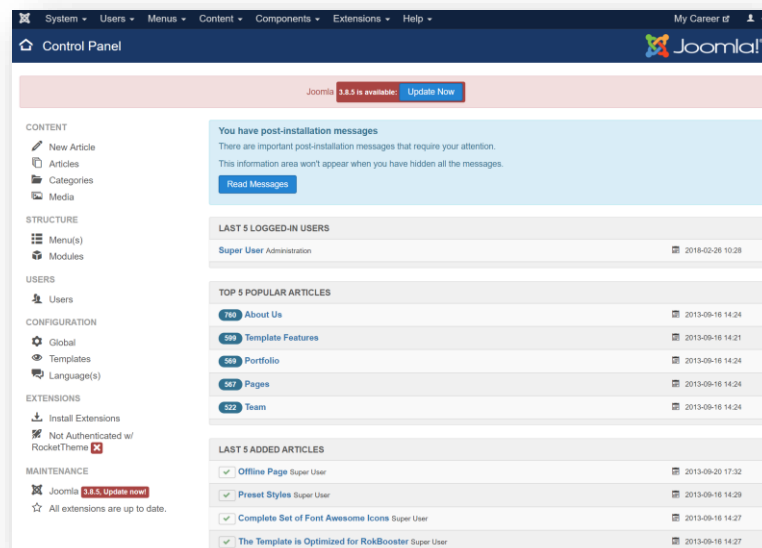


Fig. 6. Website – backend

4.1.2 Frontend (Applicants)

4.1.3 Backend (Applicants)

4.1.4 Security

One of the goals of this project is to implement security features in a RESTful API. The security is the way of protecting the system from an anonymous user access. In computer security, the data should be protected so that only authorized users can only access the data. This is typically known as Information security. Authentication and authorization are the two basic concepts in security. Authentication refers to the login access to the system. It means that the user who is registered to the system all with some role will have some login credentials to access the system.

4.1.4.1 JSON Web Tokens

JSON Web Tokens, commonly known as JWTs, are tokens that are used to authenticate users on applications. This technology has gained popularity over the past few years because it enables backends to accept requests simply by validating the contents of these JWTs. That is, applications that use JWTs no longer have to hold cookies or other session data about their users. This characteristic facilitates scalability while keeping applications secure.

During the authentication process, when a user successfully logs in using their credentials, a JSON Web Token is returned and must be saved locally (typically in local storage). Whenever the user wants to access a protected route or resource (an endpoint), the user agent must send the JWT, usually in the Authorization header using the Bearer schema, along with the request.

When a backend server receives a request with a JWT, the first thing to do is to validate the token. This consists of a series of steps, and if any of these fails then, the request must be rejected.

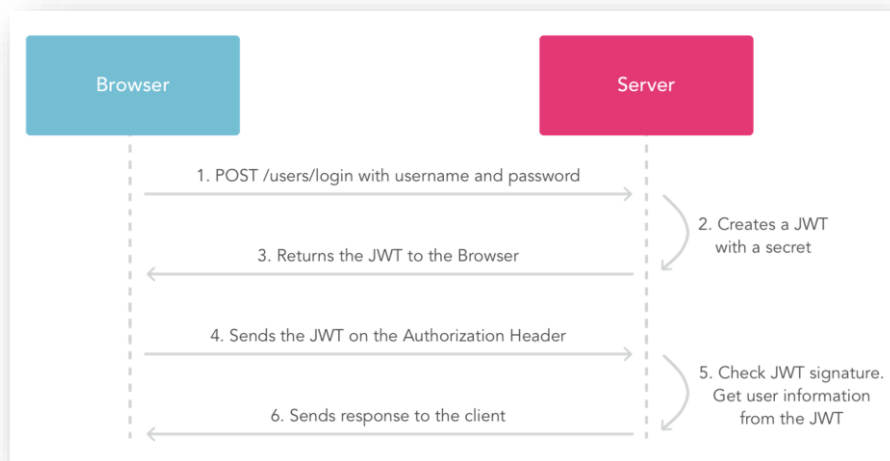


Fig. 7. JWT requesting JSON web token

4.1.5 API

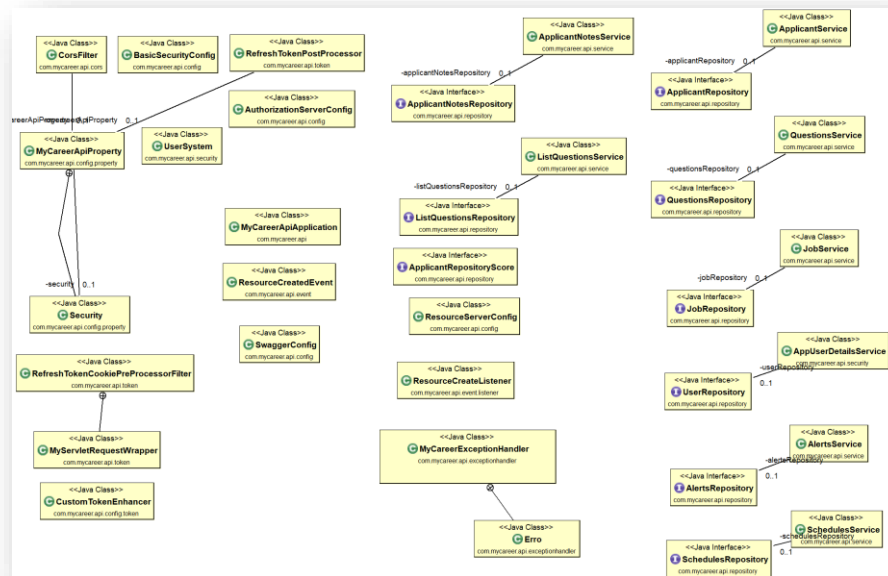


Fig. 8. UML API

4.1.6 Database

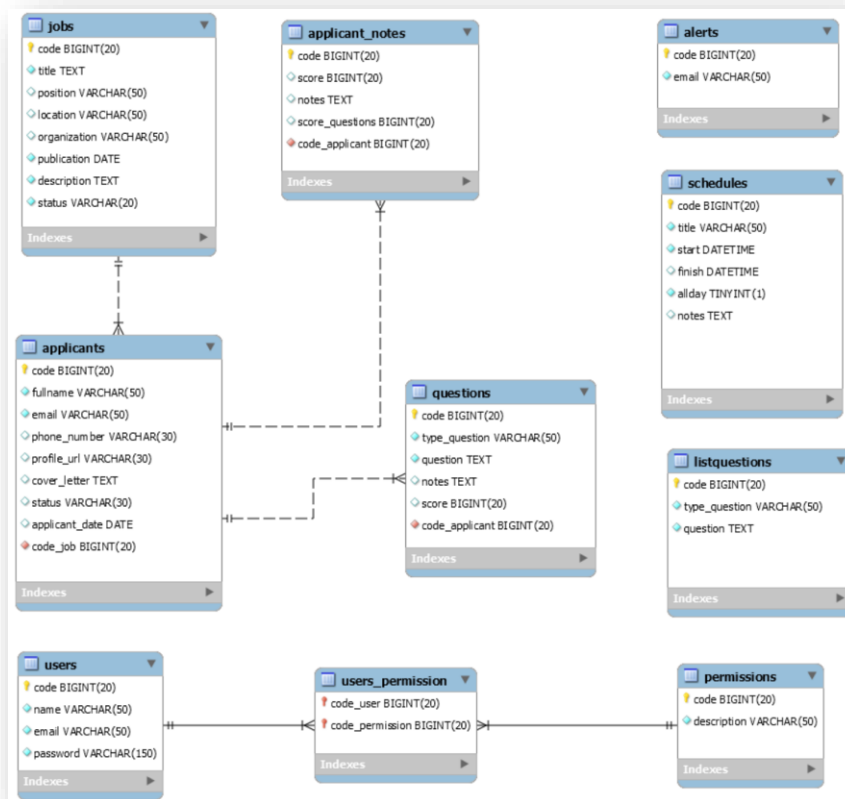


Fig. 9. Design database

4.2 System Design

Languages, tools, and technologies used during development are key factors in making applications different from one another. Once decisions are made on which tools and technologies to use, the overall system architecture, entity-relationship model, and REST API and Front-end.

4.3 REST

REST stands for Representational State Transfer. REST architecture style is mainly based on the stateless, client-server and HTTP protocol. This architectural style is a key aspect in designing network applications and distributed systems. REST does not completely rely on HTTP but mostly linked with it. The properties of REST play the vital role in the REST architecture style and make the REST architecture simpler. REST architecture is a lightweight alternative to other mechanisms like RPC4, SOAP5, and WSDL6. Moreover; REST is a platform-independent, and language-independent service. The following sub-sections explain in detail about REST.

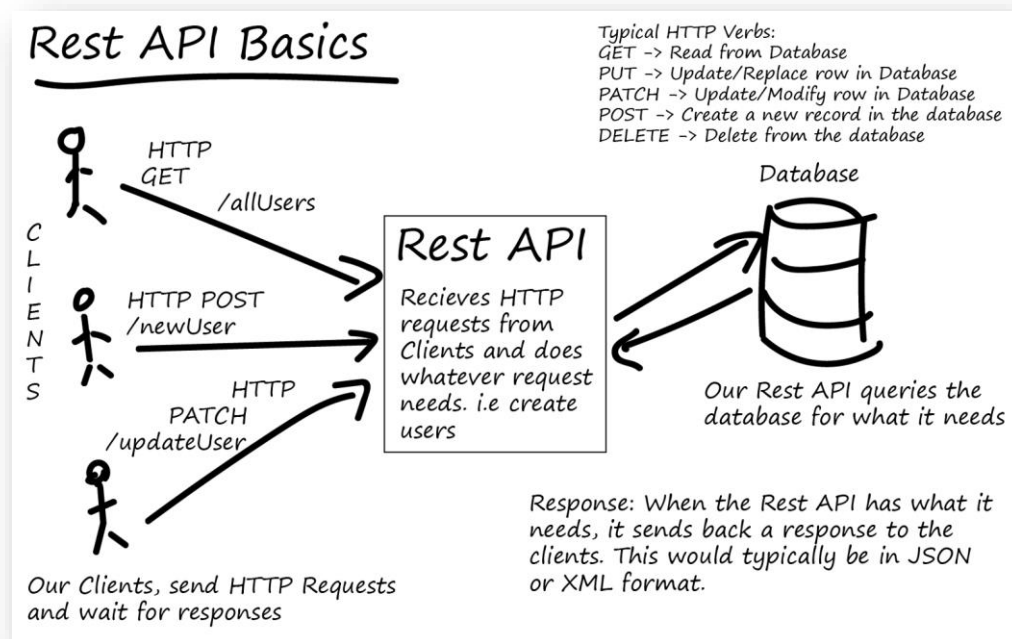


Fig. 10. Basic Rest

4.3.1 Architectural constraints

REST has a set of constraints to components, data elements, and connectors. The main constraints include client-server, stateless, layered system and uniform interface.

1. Client-server:

The client-server constraint is the most common constraint where the user-interface separates the clients from servers. Some properties or features are not mandatory for the clients, but some of those properties are important of servers. For example, some database related codes are not important to be present in client

side, but it is more important for the server. Hence, portability of the code can be improved on the client side.

2. Stateless:

The client-server communication must be stateless in nature. The stateless nature is because that server must have all the information that are needed to respond to the request made by the client. The session state entirely depends on the client. In this constraint, the properties such as visibility, scalability and reliability are improved based on different aspects. The main drawback is that is the decrease in the performance of the network by sending the same data in the cluster of requests.

4.3.2 HTTP methods

HTTP methods are used to map CRUD operations to HTTP requests. HTTP methods are used with REST to form as a RESTful service. GET, POST, PUT, and DELETE are the four main HTTP methods.

1. GET

‘GET’ is used to retrieve the data from the database. The GET requests can be partial or conditional. The partial request retrieves all the information from the particular table. The conditional request retrieves only the specific data from the database based on the condition.

Example:

GET /applicants – gets all the plants of the respective table

GET / applicants /1 – gets the plant with an ID of 1 of the respective table

2. POST

This HTTP method is mainly used to create a new entity in the table. However, it can also be used for update an existing entity.

Example:

POST / applicants – creates a new plant

3. PUT

Like POST, ‘PUT’ can be used to create a new entity and also used to update an existing entity in the table. PUT is idempotent.

Example:

PUT / applicants /1 – update the plant with an ID of 1

4. DELETE

If a resource has to be removed, then DELETE method can be used.

Example:

DELETE / applicants /1 – deletes the plant with an ID of 1

4.3.3 HTTP status codes

HTTP response status codes are the codes that are results of the HTTP requests. When an HTTP request is made from the client, the server will send an appropriate status code along with the data, if any. The browser translates these status codes. The types of HTTP status codes are:

- 1XX - informational
- 2XX - success
- 3XX - redirection
- 4XX – client error

- 5XX – server error

4.3.4 Design Patterns

Design Patterns are general reusable solutions to commonly occurring problems. Patterns are not complete code, but it can be used as a template which can be applied to a problem. Patterns are re-usable and they can be applied to similar kinds of design problems regardless of the domain. A pattern used in one practical context can be re-usable in other contexts also. Here are some of the reasons to use design patterns;

1. **Flexibility**: Using design patterns makes code flexible. It helps to provide the correct level of abstraction due to which objects become loosely coupled to each other which makes the code easy to change.
2. **Reusability**: Loosely coupled and cohesive objects and classes can make the code more reusable. This kind of code becomes easy to be tested as compared to the highly-coupled code.
3. **Shared Vocabulary**: Shared vocabulary makes it easy to share the code with other team members. It creates more of an understanding between the team members in relation to the code.
4. **Capture best practices**: Design patterns capture solutions which have been successfully applied to problems. By learning these patterns and the related problems, an inexperienced developer learns a lot about software design.

4.4 System Deployment

The best practice to deploy a spring boot and angular application is to separate the user interface code with the business logic. This provides decoupling of the client code with server code and hence the application becomes highly scalable and manageable. The frontend developer can continue with the frontend development in parallel with the backend developer. The backend code becomes free to use and integrate different technology stacks and it becomes available for multiple clients such as the same APIs can be re-used for building android application and same can be integrated with third party clients too. It also reduces the downtime of your application. Whenever, your APIs are not available or down, your client application is still up.

But sometimes it becomes an overhead to manage two servers for a small team and a small application. If a single full stack developer is handling all the UI and server related configurations, packaging frontend and backend application into a single web application is sometimes more helpful. Still, you can expose REST APIs and integrate angular frontend code within the same application and deploy to a tomcat and other mobile client can reuse the same APIs.

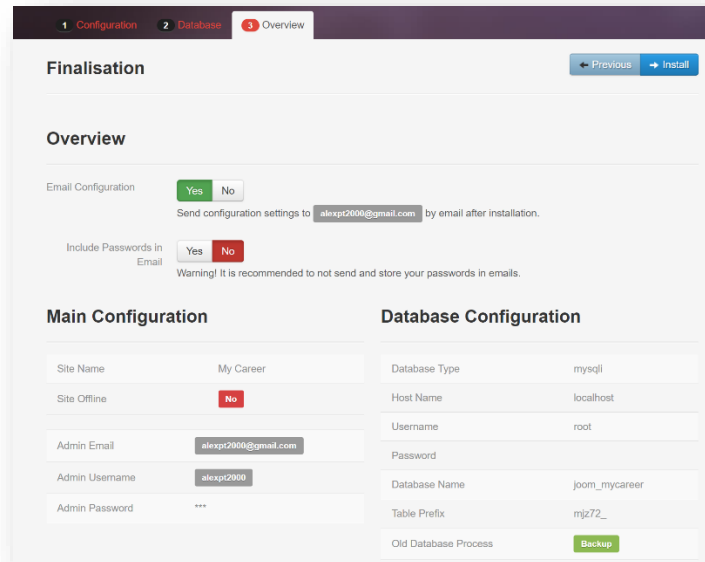
4.4.1 Deploy Joomla (AWS)

The screenshot shows the 'Main Configuration' step of the Joomla! installation process. At the top, there's a progress bar with three steps: '1 Configuration' (active), '2 Database', and '3 Overview'. Below the progress bar, there's a 'Select Language' dropdown set to 'English (United Kingdom)' and a 'Next' button. The main configuration area includes fields for 'Site Name' (My Career), 'Description', 'Admin Email' (alexpt2000@gmail.com), 'Admin Username' (alexpt2000), 'Admin Password' (masked with dots), and 'Confirm Admin Password'. There are also instructions for each field. At the bottom, there's a 'Site Offline' section with 'Yes' and 'No' radio buttons, and a note about setting the site online later.

Fig. 11. Deploy Joomla website - Configuration

The screenshot shows the 'Database Configuration' step of the Joomla! installation process. At the top, there's a progress bar with three steps: '1 Configuration', '2 Database' (active), and '3 Overview'. Below the progress bar, there's a 'Previous' button and a 'Next' button. The database configuration area includes fields for 'Database Type' (MySQL), 'Host Name' (localhost), 'Username' (root), 'Password', 'Database Name' (joom_mycareer), and 'Table Prefix' (mjz72_). There are also instructions for each field. At the bottom, there's an 'Old Database Process' section with 'Backup' and 'Remove' buttons, and a note about replacing existing backup tables.

Fig. 12. Deploy Joomla website - Database



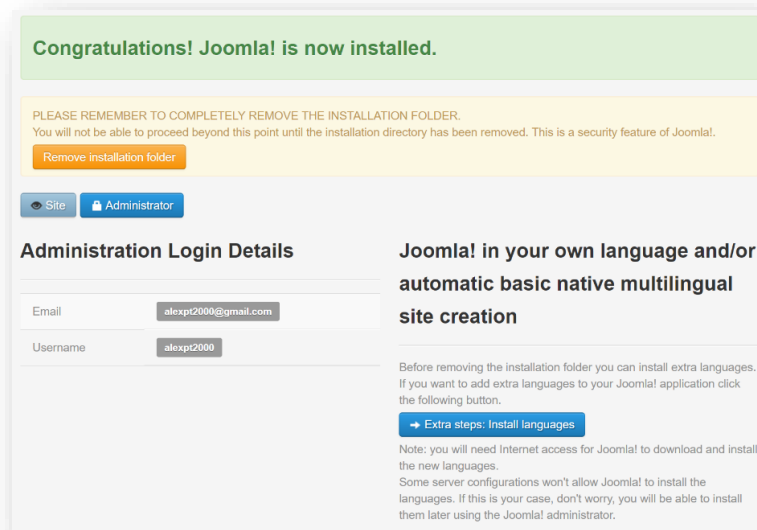
The screenshot shows the Joomla! installation 'Overview' screen. At the top, there are tabs for '1 Configuration', '2 Database', and '3 Overview', with '3 Overview' being the active tab. Below the tabs is a 'Finalisation' header with 'Previous' and 'Install' buttons. The main content area is titled 'Overview' and contains several configuration sections:

- Email Configuration:** Includes a 'Yes' button for 'Email Configuration' and a 'No' button for 'Include Passwords in Email'. A message states: 'Send configuration settings to alexpi2000@gmail.com by email after installation. Warning! It is recommended to not send and store your passwords in emails.'
- Main Configuration:** A table with the following data:

Site Name	My Career
Site Offline	No
Admin Email	alexpi2000@gmail.com
Admin Username	alexpi2000
Admin Password	***
- Database Configuration:** A table with the following data:

Database Type	mysql
Host Name	localhost
Username	root
Password	
Database Name	joom_mycareer
Table Prefix	mjz72_
Old Database Process	Backup

Fig. 13. Deploy Joomla website - Overview



The screenshot shows the Joomla! installation 'Congratulations' screen. At the top, a green banner reads 'Congratulations! Joomla! is now installed.' Below this is a yellow warning box: 'PLEASE REMEMBER TO COMPLETELY REMOVE THE INSTALLATION FOLDER. You will not be able to proceed beyond this point until the installation directory has been removed. This is a security feature of Joomla!.' A 'Remove installation folder' button is provided. Below the warning box are two buttons: 'Site' and 'Administrator'. The main content area is titled 'Administration Login Details' and contains a login form with the following data:

Email	alexpi2000@gmail.com
Username	alexpi2000

To the right of the login form, there is a section titled 'Joomla! in your own language and/or automatic basic native multilingual site creation'. It includes a note: 'Before removing the installation folder you can install extra languages. If you want to add extra languages to your Joomla! application click the following button.' A button labeled 'Extra steps: Install languages' is provided. Below this is a note: 'Note: you will need Internet access for Joomla! to download and install the new languages. Some server configurations won't allow Joomla! to install the languages. If this is your case, don't worry, you will be able to install them later using the Joomla! administrator.'

Fig. 14. Deploy Joomla website - Congratulations

4.4.2 Deploy Angular (Heroku)

<https://medium.com/@ervib/deploy-angular-4-app-with-express-to-heroku-6113146915ca>
<https://codeforgeek.com/2017/03/deploy-awesome-angular-app-heroku/>

4.4.3 Deploy API (Heroku)

<https://devcenter.heroku.com/articles/deploying-spring-boot-apps-to-heroku>
<https://dzone.com/articles/spring-boot-heroku-and-cicd>

The Spring Boot model of deploying standalone applications is a great fit for Heroku. You can use either Maven or Gradle to deploy a Spring application on Heroku.

Heroku supports Java applications packaged as WAR and JAR out of the box. The prerequisites are

- Create a Heroku account.
- Install the Heroku CLI.
- Set up Heroku locally using your Heroku login.

Once installed, you can use the Heroku command from the terminal to log in using the email address and password you used when creating your Heroku account:

```
C:\Users\Alexander Souza\Desktop\GIT\4Year_MainProject_MyCareer\API-Spring>heroku login
Enter your Heroku credentials:
Email: alexpt2000@gmail.com
Password: *****
```

Fig. 15. Login Heroku

```
μfcfb2:\\wλc9λeεελ-9b7·μελοκn9bb·com\\ | μfcfb2:\\ε7fc·μελοκn·com\\wλc9λeεελ-9b7·ε7fc
cλe9λ7uε wλc9λeεελ-9b7·... qoue
C:/n2εελ2/γ7εx9uqελ 2on29/De2κfob/ε7I1/φλe9λ_μ97uελo]εcf~wλc9λeεελ/φbI-2bλ7uε>μελοκn cλe9λe wλc9λeεελ-9b7
```

Fig. 16. Create a new app on Heroku

You can attach a MySQL database to your app by Add-ons Marketplace to see what is available.

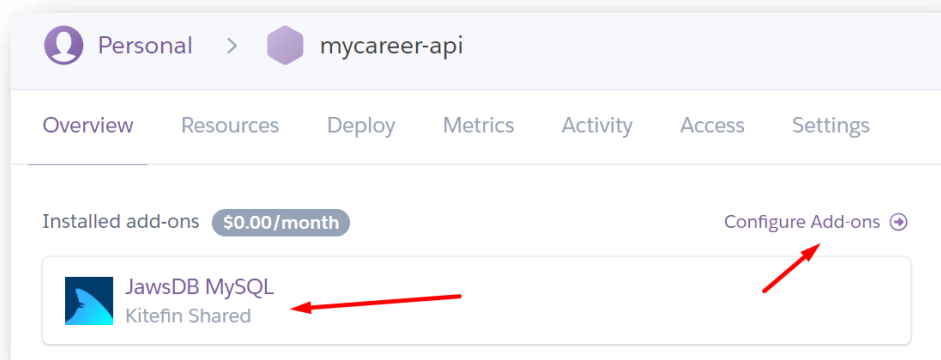


Fig. 17. Add MySQL to the API

Now you can list the configuration variables for your app to display the URL needed to connect to the database, DATABASE_URL:

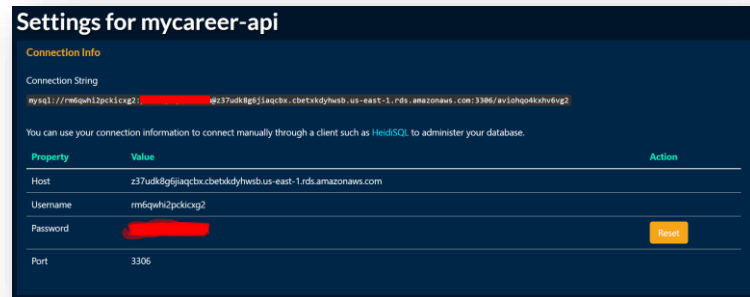


Fig. 18. JawsDB – Settings

```

1 mycareer.security.enable-https=true
2
3 spring.datasource.url={JDBC_DATABASE_URL}
4 spring.datasource.username={JDBC_DATABASE_USERNAME}
5 spring.datasource.password={JDBC_DATABASE_PASSWORD}
6
7 mycareer.origin-allowed=https://mycareer-webadmin.herokuapp.com
8
9 spring.profiles.active=oauth-security
10

```

Fig. 19. Database profile on API

Once the database add-on has been created, Heroku will automatically populate the environment variables `JDBC_DATASOURCE_URL`, `JDBC_DATASOURCE_USERNAME`, and `JDBC_DATASOURCE_PASSWORD`.

```

heroku config:set JDBC_DATABASE_URL=jdbc:mysql://z37udk8g6jiaqcbx.cbetxkdyhwsb.us-east-1.rds.amazonaws.com:3306/wkgt01bjg7r2hj7o JDBC_DATABASE_USERNAME=kj5hcza3lok330ed JDBC_DATABASE_PASSWORD=gfo2yz94v3mhh5r8 -a mycareer-api

```

Heroku supports AWS-style Procfile. Create one in the application root directory (with capital P) and enter.

The `heroku deploy:jar` command can pass additional parameters to the jar file, so it should be possible to execute.

```

Procfile
1 web: java -Dserver.port=$PORT -Dspring.profiles.active=oauth-security,prod $JAVA_OPTS -jar target/mycareer*.jar
2

```

Before you can deploy the app to Heroku, you'll need to create a Git repository for the application and add all of the code to it by running these commands:

- Git init
- Git add .

then

```
C:\Users\Alexander Souza\Desktop\GIT\4Year_MainProject_MyCareer\API-Spring>git commit -m "READ to deploy"
[master warning: LF will be replaced by CRLF in Procfile.
The file will have its original line endings in your working directory.
14922ee] READ to deploy
warning: LF will be replaced by CRLF in Procfile.
The file will have its original line endings in your working directory.
1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Alexander Souza\Desktop\GIT\4Year_MainProject_MyCareer\API-Spring>git status
On branch master
nothing to commit, working directory clean

C:\Users\Alexander Souza\Desktop\GIT\4Year_MainProject_MyCareer\API-Spring>git push heroku master
```

Fig. 20. Deploy – Commit API and Push to master

Now the API been Deploy.

```
1.0.0-SNAPSHOT.jar
remote:      [INFO] Installing /tmp/build_dd1552d9a1403262da0cc22c7e01c5ae/pom.xml to
remote:      [INFO] -----
remote:      [INFO] BUILD SUCCESS
remote:      [INFO] -----
remote:      [INFO] Total time: 16.191 s
remote:      [INFO] Finished at: 2018-02-19T14:36:04Z
remote:      [INFO] Final Memory: 43M/353M
remote:      [INFO] -----
remote: -----> Discovering process types
remote: Procfile declares types -> web
remote:
remote: -----> Compressing...
remote: Done: 81.4M
remote: -----> Launching...
remote: Released v5
remote: https://mycareer-api.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/mycareer-api.git
 * [new branch]      master -> master

C:\Users\Alexander Souza\Desktop\GIT\4Year_MainProject_MyCareer\API-Spring>_
```

Fig. 21. Deploy process

Heroku automatically detects the application as a Maven/Java app due to the presence of a pom.xml file. It installed Java 8 by default, but you can easily configure this with a system.properties file as described in the Specifying a Java version Dev Center article. It will run your app using the default command.

You can view the logs for the application by running this command.

```
heroku logs --tail
```

[illegible]

Fig. 22. Heroku print the log

Reload your application in the browser, and you'll see another log message generated for that request. Press Control+C to stop streaming the logs.

Using Spring Boot, Heroku and GitHub, a complete application can be created and deployed using free tools (or at least the free versions of paid tools). Heroku itself is very well documented and easy to use.

Now the API can access in:

<https://mycareer-api.herokuapp.com/>

5 System Evaluation

5.1 Robustness & Efficiency

5.1.1 Tomcat

5.1.2 MongoDB

5.2 Space / Time Complexity

5.3 Security and Validation

The API is now secured and supports authentication with JWTs. When the server is up and running, can test this out by querying <https://mycareer-api.herokuapp.com/oauth/token>, and should get a message saying "Access Denied." To authenticate ourselves, send a POST request to <https://mycareer-api.herokuapp.com/oauth/token/> with our user's credentials in the body: {"username":"admin","password":"password"}.

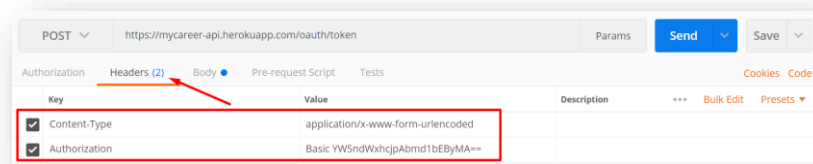


Fig. 23. Add Authorization on Headers to request token

Client credentials are being sent to the API in BASE64.

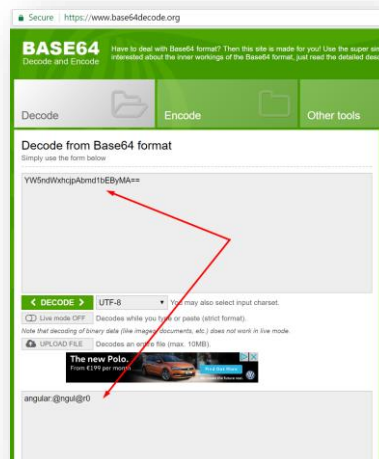


Fig. 24. Decode Authorization value

Then the authentication information of the client will be informed in the Body.

Request Token

POST https://mycareer-api.herokuapp.com/oauth/token Params Send Save

Authorization Headers (2) **Body** Pre-request Script Tests Cookies Code

☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/> client_id	angular			
<input checked="" type="checkbox"/> username	admin@mycareer.com			
<input checked="" type="checkbox"/> password	admin			
<input checked="" type="checkbox"/> grant_type	password			

Fig. 25. Add body, key and value to request token

In the response to this request, will get a token as part of the Authorization header, prefixed by "Bearer". Can copy this token, with the Bearer prefix, to issue GET requests to /applicants path.

Request Token X List Applicants OAuth List Jobs Web - No Security List Jobs Web - No Se + No Environment

Request Token

POST https://mycareer-api.herokuapp.com/oauth/token Params Send Save

Authorization Headers (2) **Body** Pre-request Script Tests Cookies Code

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Body Cookies (1) Headers (15) Test Results Status: 200 OK Time: 2743 ms Size: 1.88 KB

Pretty Raw Preview JSON Save Response

```

1 {
2   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9",
3   "token_type": "bearer",
4   "expires_in": 1799,
5   "scope": "read write",
6   "name": "Administrator",
7   "jti": "b771f7fb-2dab-44c8-b346-cb41533190a"
8 }

```

Fig. 26. Get JSON Web Token with Postman

The copied token can be decoded on JWT.io web site. This can provides information's like authentication roles.



Fig. 28. Request applicants

5.4.1 Limits of the system.

6 Conclusion

6.1 Summary

6.2 Learning Outcomes

6.3 Reflection

7 References

References

- [1] "Spring Security," [Online]. Available: <https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/>.

8 Appendices