13/09/2018

Internship Tutor

Valentini Stefano	254825	stefano.valentini2@student.univaq.it
Cecchini Valentina	255596	valentina.cecchini@student.univaq.it
Battista Federico	254376	federico.battista@student.univaq.it
Berardini Daniele	255683	daniele.berardini1@student.univaq.it

Software Dependencies

Client side:

- HTML5
- CSS3
- Javascript
- Bootstrap 3
- jQuery
    - EasIng
    - UltoTop
- Toastr

Server side:

- JavaEE
- Spring Boot
- Spring Security
- Spring Data Jpa
- MySql Connector
- Thymeleaf
- Apache PdfBox
- Apache Maven
- Apache Commons Text

Features

All the required and optional features have been implemented.

Navigability and flow of events

The navigation model can be found in the same directory as this document.

The homepage contains the public list of some of the currently published internships, also available to anonymous users and a search bar that allows to perform queries to find internship offers.

In the right top corner, if the user is not logged in, a “Register” link will appear; in the case of a student, after the registration form is submitted it will be immediately activated.

Companies, instead, will require a manual confirmation by the administrator.

From the top menu, logged users can reach their respective Dashboards:

- the *student dashboard* allows the student to see its personal information, with the lists of the completed internships, ongoing internships and internships waiting for approval. From the entries in the completed internships is possible to download the training project and the final report, while from the ongoing internships only the training project is available for download.

- the *company dashboard* allows the company to see some of its personal information, with a link to the form used to insert a new internship and a download link to get the agreement that the company has signed with the university (obviously the company must be activated before accessing its dashboard). There are also several tables in which the company can retrieve information and documents, just like the student dashboard. From its dashboard the company can also update its internship offers and accept/reject students (this will be later explained).

- the *administrator dashboard* follows the scheme of the *company* and *student dashboards*, allowing the administrator to see several statistics about the internships. A banner is also available at the top of the page to notify the admin that some companies are waiting for approval. The administrator can activate a company by downloading the precompiled agreement file and uploading it back after having filled and signed it.

A logged student can apply to an internship by clicking on a button in the page that contains all the details of the offer, this page can be reached by clicking on the summary of the internship itself in the home page.

After a student applies to an offer, the respective company will see a new entry in its dashboard, from the update page the company can immediately reject the student or proceed to accept him.

To accept a student the company has to download the precompiled training project file and reupload it after it has been filled and signed.

After the internship has ended, the company, using the same procedure, can download a precompiled final report so that it can be filled, signed and uploaded to the system.

From its dashboard the student, after the internship has been marked as completed, can give a review (with a mark from 1 to 5) to the company. The average of the reviews will be displayed on the published offers of the company.

Finally, from the top menu, the administrator can access the report tables of all the entities on the database; by clicking on the respective entry, the administrator can update its information or delete it. He can also insert a new entry by using the respective button.

ER Model of the database

The ER model can be found in the same directory as this document.

Layout

The layout is totally responsive and it relies on **Bootsrap 3**.

It is composed by a header in which there are the links to the social accounts of the university.

Just below it there is a sub-header that contains a horizontal drop-down menu that links to all the principal parts of the application. The first level of the drop-down menu Is activated by a click, while the second level is revealed on hover.

The horizontal menu also contains the login box.

In the middle of the page there is the content that occupies the full horizontal length.

At the bottom there is the footer which contains the address of the university and an iframe that embeds a map of the university's buildings.

In every page there is also a pop-up box in the top right corner that can be triggered by the backend of the application and it automatically disappears after some seconds or on click.

Adopted technologies

Spring Boot: the whole application relies on spring boot and more in general on the Spring framework. It allows us to use an advanced implementation of the MVC pattern that is also enriched by several extra sub-frameworks such as **Spring Security** that secures the application and implements the login functionality out-of-the-box.

In fact, it is possible to just provide the query that is used to retrieve the roles of the users and the module does the rest; given a series of filters it also checks that a certain resource is only accessed by entities that do not clash with the aforementioned filters.

The whole server-db communication is implemented using the **Spring Data Jpa** sub-framework that is embedded in Spring. This allows us to abstract the communication by just declaring the names of the methods that will be implemented by the framework itself to retrieve information from the database. It is only necessary to actually write queries if there are complex operations to perform, like multiple joins, group by, etc.

The pdfs are created and filled with information using **Apache PDFBox**, it is one of the most used libraries for pdf manipulation in java. After having created the pdf-form version of the file we are interested in, it is enough to open them through the PDFBox apis and fill them through the *.setValue()* methods; when we are finished we can just save as copy.

We used **Thymeleaf** as template engine because it allows strong integration with Spring such as:

- all the CRUD forms in the backoffice are implemented through the binding offered by Thymeleaf and Spring, that is, if we define that in a form we are listing the fields of a particular entity, it will enough to tell Spring that the object that is coming from that POST request is actually of that type.
  In the same way Thymeleaf understands what object is being displayed in the forms and so it is able to provide useful ui enhancements.

- all the static resources and url routes are automatically completed when the page is built, e.g. *@{/css/bootstrap.css}* becomes */internshiptutor/static/css/bootstrap.css*

Screenshots

The screenshots can be found in the same directory as this document.

Contributions and Roles

The work has been partitioned in features, meaning that every member has experienced every aspect of the stack (backend, database, frontend).

Member	**Feature**
Valentini Stefano	Dashboards, CRUD Forms, PDF Files generation and download.
Cecchini Valentina	Dashboards, CRUD Forms, PDF Files generation and download.
Battista Federico	CRUD Forms, Authentication, Index, Internship Page, Search Page.
Berardini Daniele	CRUD Forms, Index, Internship Page, Affiliated Companies Page.