



**POLITECNICO**  
**MILANO 1863**

**Implementation**  
**&**  
**Testing Document**

**Software Engineering 2**

Emanuele Cimino - Gabriele Lorenzetti

Prof. Di Nitto Elisabetta



Contents

**Table of Contents** . . . . . 2

**List of Figures** . . . . . 4

**List of Tables** . . . . . 4

**1 Introduction** . . . . . 5

1.1 Purpose . . . . . 5

1.2 Definition, Acronyms, Abbreviations . . . . . 5

1.2.1 Definitions . . . . . 5

1.2.2 Acronyms . . . . . 5

1.3 Revision History . . . . . 6

1.4 References . . . . . 6

1.5 Document Structure . . . . . 6

**2 Development** . . . . . 7

2.1 Implemented Functionalities . . . . . 7

2.2 Adopted development frameworks . . . . . 7

2.2.1 Backend . . . . . 7

2.2.2 Frontend . . . . . 8

**3 Source Code** . . . . . 11

3.1 Backend . . . . . 11

3.2 Frontend . . . . . 12

**4 Testing** . . . . . 13

4.1 Test Cases . . . . . 13

4.1.1 Code coverage . . . . . 13

**5 Installation and Usage** . . . . . 14

5.1 Installation . . . . . 14

5.1.1	Prerequisites . . . . .	14
5.1.2	Installation Steps . . . . .	14
<b>6</b>	<b>Effort Spent . . . . .</b>	<b>15</b>

List of Figures

1	Spring MVC . . . . .	7
2	Spring Functionalities . . . . .	8
3	Packages . . . . .	11
4	Enter Caption . . . . .	13
5	Enter Caption . . . . .	13

List of Tables

1	Effort Recap . . . . .	15
---	------------------------	----

# 1 Introduction

## 1.1 Purpose

The aim of this document is to provide an exhaustive overview of the development process of the S&C platform.

Highlighting the advantages and disadvantages of methodologies and technologies taken into consideration for the chosen framework and the structure of developed source code.

In the last section we will show a step by step guide for installing the prototype of the S&C platform to test its functionalities.

## 1.2 Definition, Acronyms, Abbreviations

### 1.2.1 Definitions

- **Student:** is a user who try to find an internship.
- **Company:** is a user who wants to offer an internship.
- **Internship:** is a temporary work experience offered by a company to students, recent graduates, or young professionals. It aims to provide practical training and professional skills in a specific field.
- **Keyword:** is word or series of words to perform a student search for internships.
- **Request:** Student requests to apply and be interviewed for internship.
- **Form:** It is a form with questions chosen by the company at the creation of the internship to interview the student and understand if he/she is a good fit for the internship.
- **Complaint:** during the internship both the company and the student can leave a complaint about the other.
- **Feedback:** after the middle and at the end of the internship, for one week; the company and the student can provide feedback on the service offered by the platform by answering the questions with a score of 0 to 5.
- **Review:** at the end of the internship, for a week the company and the student can leave a review and a score from 0 to 5 on how the internship went.

### 1.2.2 Acronyms

- **S&C :** Student and Companies platform
- **MVC :** Model View Controller
- **DTO :** Data Transfer Object, architectural pattern used for transferring data between software application subsystems. It is an object that defines the structure of data to be sent or received by/from servers.
- **JWT:** Json Web Token
- **JS:** JavaScript
- **HTML:** Hyper Textual Modeling Language

- **DOM:** Document Object Model, a programming interface for web documents. It represents the web page and programs can access it to change the document structure, style, and content.
- **VDOM:** Virtual DOM.
- **JSON:** JavaScript Object Notation, universal standard for the exchange of object-like information.
- **GUI:** Graphic User Interface
- **RASD:** Requirement Analysis and Specification Document
- **DD:** Design Document

### 1.3 Revision History

- v1.0 – First version of the document. Revised on 02/02/2025

### 1.4 References

- "Assignment IT AY 2024/2025"
- "RASD by Emanuele Cimino - Gabriele Lorenzetti"
- "DD by Emanuele Cimino - Gabriele Lorenzetti"
- [Spring boot documentation](#)
- [Vue.js documentation](#)
- [Postgresql documentation](#)

### 1.5 Document Structure

1. **Introduction:** This section offers a brief introduction to the document that is here presented, including all the definitions, acronyms, and abbreviations that will be found reading it.
2. **Development:** This section offers a detailed view on all the aspects concerning the development process. In particular, the implemented functionalities are described, together with all the adopted frameworks, which are described taking into consideration advantages and disadvantages.
3. **Source Code:** This section offers a concise but comprehensive overview of the structure and organization of the source code for the S&C platform, describing the roles of the main modules of the application.
4. **Testing:** This section provides an exploration of the testing methodologies applied during the development of the S&C platform.
5. **Installation and Usage :** This section is addressed to those involved in installing, configuring, and using the prototype of the S&C platform. It includes a comprehensive installation guide with step-by-step instructions, ensuring a smooth setup process.

## 2 Development

### 2.1 Implemented Functionalities

All the functionalities presented in the RASD and DD have been implemented except notifications and rank list, even though some of the functionalities can results a bit simplified, according to a prototype implementation.

- Register Student/Company
- Login Student/Company
- Modify Info
- Insert Internship (*Company related*)
- Search Internship with filters(*Student related*)
- Apply for an internship (*Student related*)
- Select a trainee (*Company related*)
- Interview Compiling (*Student related*)
- Trainee can renounce before the start of internship
- Sending complaints, feedback and reviews
- Public pages with their information and reviews

As requested to 2-member group, the collection of statistics and the implementation of the recommendation system are not included.

### 2.2 Adopted development frameworks

#### 2.2.1 Backend

- Spring MVC

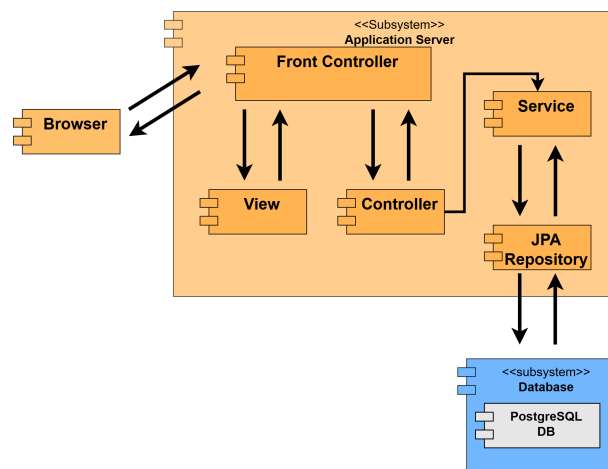


Figure 1: Spring MVC

The Spring MVC framework follows the Model-View-Controller architectural design pattern. Works around the Front Controller, which is the representation of the dispatcher servlet in the

next image. The servlet dispatcher handle and, as said, dispatch the request to the appropriate `@RestController` and follow the appropriate `@RequestMapping`. Then the requests are handled by a service class that handles the true logic of the method and interface with the database through in our case the JPA repository

- **@RestController** : Signal that a specific class is a controller
- **@RequestMapping** : Maps a specific request to a specific method in the controller class

- **Spring Model View Controller flow diagram**

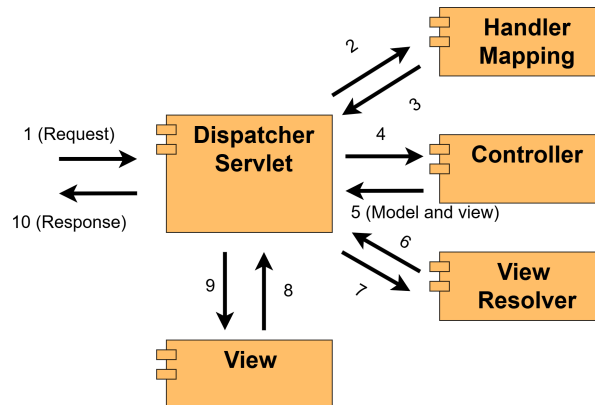


Figure 2: Spring Functionalities

1. **Dispatcher servlet:** The primary controller that takes the request and sends it to the appropriate controller through the handler Mapping, as it intercepts all requests, it serves as the front controller.
2. **Handler mapping:** It is an XML configuration file that help to find the appropriate handler mapping for every request. Redirect every request to the designated controller.
3. **Controller:** The chosen controller that process the request, after the processing it returns a model-view object that represent the data and how they should be seen.
4. **View:** The dispatcher goes now back to the XML file to identify the correct view resolver and call him to render appropriately the response

### 2.2.2 Frontend

- **Vue.js:**

It is a JavaScript framework designed for building user interfaces. It extends standard HTML, CSS, and JavaScript, offering a declarative and component-based programming model that simplifies the development of both simple and complex UIs.

This framework is built on Node.js, which manages the creation and initialization of a Vue project. Node.js provides all the necessary modules for setting up the project structure, including essential files such as `index.html`, `main.js`, and various configuration files. Additionally, Vue leverages npm (Node Package Manager) to facilitate development.

By running `npm run serve`, a local development server is launched, allowing real-time visualization of code changes. While the server is running, Vue components are continuously recompiled and reorganized by Node.js into HTML and JavaScript modules whenever an update is detected. Consequently, the DOM (Document Object Model) is dynamically updated to reflect these changes.

- **Vue.js Rendering Mechanism:**



Vue.js employs the Virtual DOM (VDOM) paradigm as its rendering mechanism. This means that an ideal, virtual representation of the UI is maintained in memory and kept in sync with the actual DOM.

In Vue.js, the Virtual DOM tree is composed of plain JavaScript objects known as virtual nodes (vnodes). These vnodes represent UI elements and contain all the necessary data to generate the actual DOM elements. The runtime renderer processes the Virtual DOM tree and constructs the real DOM tree in a process called mounting. This approach enhances performance by updating only the necessary parts of the DOM instead of re-rendering the entire interface.

- **Vue.js code structure:**

Vue's source code is organized into Vue components (.vue files), which interact with each other to exchange data and collaborate. Each component consists of three main sections:

<template>

This section defines the HTML structure of the component, specifying the layout of the corresponding page or frame. In addition to standard HTML tags, Vue directives (such as v-model, v-bind, etc.) can be used to establish reactive data binding between the template and the script.

Thanks to Virtual DOM (VDOM) compliance, dynamic behaviors can be defined for HTML elements, allowing their properties to be updated automatically in response to changes in the script.

<style>

This section is responsible for defining the styling of the component. It includes CSS code to customize the visual appearance of the template. Styles can be scoped to the component to prevent conflicts with styles from other components.

<script>

This section contains the logic of the Vue component. It defines the component's data, methods, and properties. Vue also provides pre-defined lifecycle methods, which can be overridden to control the component's behavior during different stages of its existence—for example, when the page is created, when data is updated, and so on.

- **Advantages of Vue.js:**

- **Flexibility and reactiveness:** Vue.js provides a high level of flexibility, enabling developers to integrate it gradually into existing projects or utilize specific parts of the framework as needed. This adaptability makes it suitable for a variety of applications. Moreover, Vue.js is inherently reactive, meaning that any changes to the underlying data are automatically reflected in the user interface. This reactive nature streamlines the development process, eliminating the need for manual DOM updates when the data changes.
- **Two-way data binding:** Vue.js features a two-way data binding system that synchronizes the model and the view. When the data in the model changes, the corresponding view updates automatically, and changes in the view reflect back in the model. This bidirectional data flow minimizes the code required to manage application state, resulting in cleaner and more maintainable code.
- **Component-based architecture:** Vue.js adopts a component-based architecture, fostering modularity and reusability. Each component encapsulates its own logic, template, and styles, simplifying the management and scaling of complex applications. This approach enables developers to work on isolated components and seamlessly integrate them to construct the overall application.

- **Ease of learning and Intuitiveness:** The framework's syntax is clear and concise, making it accessible to developers new to Vue.js or frontend development in general. Its intuitive nature simplifies the integration process, allowing developers to quickly understand key concepts and start building applications.

Overall, Vue.js is a smart and versatile framework that adapts to various UI projects. However, due to its features, it is particularly well-suited for Single Page Applications (SPAs)- applications that load a single HTML page and dynamically update content as users interact.

## 3 Source Code

### 3.1 Backend

The backend is developed merging the maven framework with the spring boot one

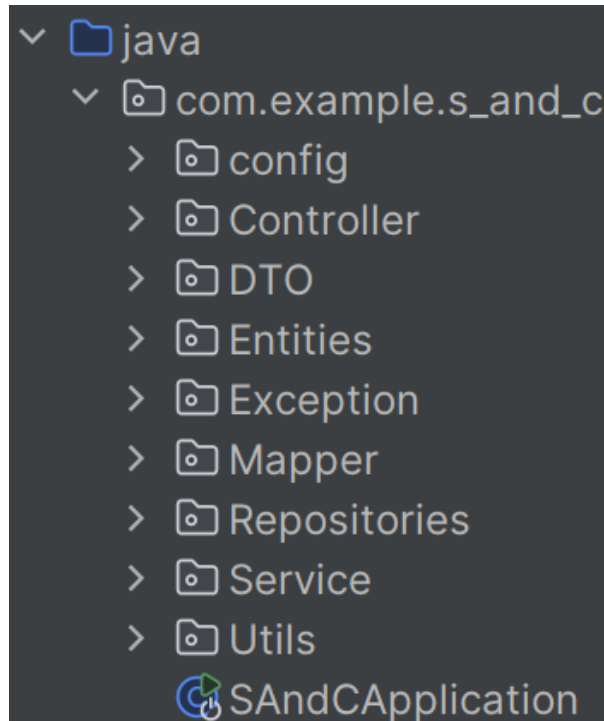


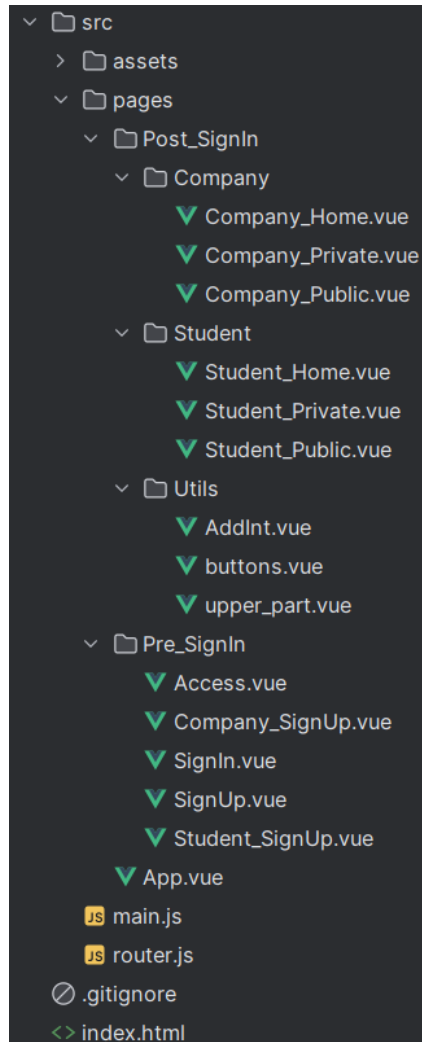
Figure 3: Packages

- **Config:** Contain all the configuration classes of the application, such as the one used to manage security and the general application configuration.
- **Controller:** This package contains the controllers of the application according to the MVC paradigm.
- **DTO:** According to the Spring framework, this package contains all DTOs used by the client to send information to the server and vice versa, it's the model in the MVC pattern.
- **Entities:** This package contains classes that represent database entities, such as student companies and internships.
- **Exception:** Contains the ad hoc created exceptions useful for the development of the application.
- **Mapper:** This package contains all the classes that help to create an entity from the model information retrieved with the DTOs.
- **Repositories:** This package contains all the JPA queries for every entities.
- **Service:** This package contains all the interfaces and implementations of the methods used by the controllers classes.
- **Utils:** This package contains classes useful for the functioning of the application.

## 3.2 Frontend

The frontend source code pushed in the GitHub repository consists only of the VUE and JS files which are behind the build of the node project.

The code is structured as follows:



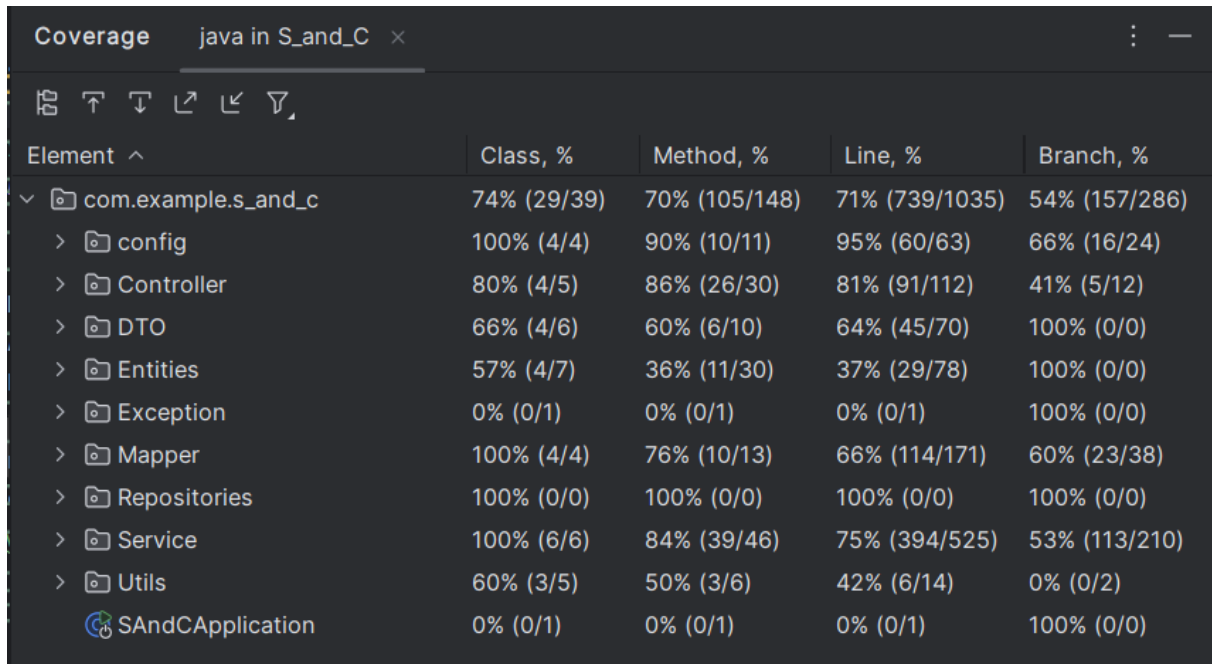
- **index.html:** this is the core HTML file of the whole frontend application.
- **main.js and App.vue:** these two files are responsible for the boot of the VUE application. App.vue displays the view dispatched by the Router and main.js creates and mounts the application.
- **router.js:** this is the configuration file for the Vue Router. Here all the main sub-routes of the application are defined and configured.
- **assets:** this directory contains all the icons, logos and images loaded by the components.
- **Utils:** this directory includes the common views of the student and company; upper-part is a sort of menù with the icons to navigate in the website.
- **Pre-Signin:** this directory includes the views before the signup.
- **Company:** this directory contains the main views of the company.
- **Student:** this directory contains the main views of the student.

## 4 Testing

### 4.1 Test Cases

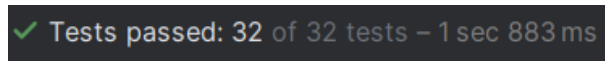
#### 4.1.1 Code coverage

With 32 out of 33 test passed at 2/2/2025, some test depends on the actual local date, so if want to test, date in setup method may need to be changed This code coverage has been reached



| Element ^             | Class, %    | Method, %     | Line, %        | Branch, %     |
|-----------------------|-------------|---------------|----------------|---------------|
| ✓ com.example.s_and_c | 74% (29/39) | 70% (105/148) | 71% (739/1035) | 54% (157/286) |
| > config              | 100% (4/4)  | 90% (10/11)   | 95% (60/63)    | 66% (16/24)   |
| > Controller          | 80% (4/5)   | 86% (26/30)   | 81% (91/112)   | 41% (5/12)    |
| > DTO                 | 66% (4/6)   | 60% (6/10)    | 64% (45/70)    | 100% (0/0)    |
| > Entities            | 57% (4/7)   | 36% (11/30)   | 37% (29/78)    | 100% (0/0)    |
| > Exception           | 0% (0/1)    | 0% (0/1)      | 0% (0/1)       | 100% (0/0)    |
| > Mapper              | 100% (4/4)  | 76% (10/13)   | 66% (114/171)  | 60% (23/38)   |
| > Repositories        | 100% (0/0)  | 100% (0/0)    | 100% (0/0)     | 100% (0/0)    |
| > Service             | 100% (6/6)  | 84% (39/46)   | 75% (394/525)  | 53% (113/210) |
| > Utils               | 60% (3/5)   | 50% (3/6)     | 42% (6/14)     | 0% (0/2)      |
| SAandCApplication     | 0% (0/1)    | 0% (0/1)      | 0% (0/1)       | 100% (0/0)    |

Figure 4: Enter Caption



✓ Tests passed: 32 of 32 tests – 1 sec 883 ms

Figure 5: Enter Caption

- **OngoingInternshipTest:** Control functions from the ongoing internships.
- **StudentAndCompanyDataTest:** Control functions from the selectionphase.
- **OuthControllerTest:** Control function from the authorization controller.

## 5 Installation and Usage

### 5.1 Installation

#### 5.1.1 Prerequisites

- IDE
- Java JDK 21 corretto
- [DataBase PostGres](#)
- [Node.js and npm](#)

#### 5.1.2 Installation Steps

- Clone the repository [github](#).
- Create the table public in the PostGres database.
- Start the main, SandCApplication and the frontend in package.json.
- Use <http://localhost:5173>; for each user use a different browser.

For any doubts or issues you can contact us:

- [emanuele.cimino@mail.polimi.it](mailto:emanuele.cimino@mail.polimi.it)
- [gabriele.lorenzetti@mail.polimi.it](mailto:gabriele.lorenzetti@mail.polimi.it)

## 6 Effort Spent

| Sections               | Cimino | Lorenzetti | Together | Total |
|------------------------|--------|------------|----------|-------|
| Introduction           | 0      | 0          | 1        | 1     |
| Development            | 1      | 1          | 1        | 3     |
| Source Code            | 1      | 1          | 0        | 2     |
| Testing                | 0      | 0          | 1        | 1     |
| Installation and Usage | 0      | 0          | 2        | 2     |
| Total                  | 2      | 2          | 5        | 9     |

Table 1: Effort Recap