Traineeship Application

Sprint Report

NESTOR MOULIAS A.E.

Παναγιώτης Τύπος, 5131
Νέστωρ Μούλιας, 4737
Ευάγγελος Βαΐτσης, 5096

VERSIONS HISTORY

Date	Version	Description	Author
15/03/2025	V1.0	First meeting and role assignment to each one	Panagiotis Trypos
16/03/2025	V1.1	Creating a plan for the project and a design for the structure	Panagiotis Trypos
18/03/2025	V1.2	Completion of general user story about registration and login actions	Panagiotis Trypos
23/03/2025	V1.3	Completion of user stories with ID US4, US7 and US13	Panagiotis Trypos
04/04/2025	V1.4	Completion of user stories with ID US5, US8, US10, US11 and US16	Panagiotis Trypos
16/04/2025	V1.5	Completion of user stories with ID US17, US18, US19, US20	Panagiotis Trypos
22/04/2025	V1.6	Completion of user stories with ID US6, US9, US14	Panagiotis Trypos
29/04/2025	V1.7	Completion of user stories with ID US12, US15, US21	Panagiotis Trypos
07/05/2025	V1.8	Implementation of the tests	Panagiotis Trypos
10/05/2025	V1.9	Completion of creating the UML Diagrams (Class and package)	Panagiotis Trypos
17/05/2025	V1.10	Creation of the CRC cards	Panagiotis Trypos
20/05/2025	V1.11	Completion of the Use Cases	Panagiotis Trypos

1 Introduction

1.1 Purpose

The purpose of the project is to create a Traineeship Application where students can apply for traineeship, companies can add traineeship positions for the students and professors can be supervisors in traineeship positions.

1.2 Document Structure

The rest of this document is structured as follows. Section 2 describes out Scrum team and specifies the Sprint's backlog. Section 3 specifies the main design concepts for this release of the project.

2 Scrum team and Sprint Backlog

2.1 Scrum team

Product Owner	Panagiotis Trypos
Scrum Master	Nestor Moulias
Development Team	Panagiotis Trypos, Nestor Moulias, Eyaggelos Vaitsis

2.2 Sprints

Sprint No	Begin Date	End Date	Number of weeks	User stories
1	**14/03/2025**	**18/03/2025**	**0.5**	**US1, US2, US3**
2	**19/03/2025**	**23/03/2025**	**0.5**	**US4, US7, US13**
3	**28/03/2025**	**4/04/2025**	**1**	**US5, US8, US10, US11, US16**
4	**9/04/2025**	**16/04/2025**	**1**	**US17, US18, US19, US20**
5	**17/04/2025**	**22/04/2025**	**0.7**	**US6, US9, US14**

6	**23/04/2025**	**29/04/2025**	**0.9**	**US12, US15, US21**

3 Use Cases

<Specify the concrete Use Cases that describe the interaction of the user with the applications, as derived from the abstract user stories. Give a **UML Use Case diagram** and the **detailed use case descriptions**.>

3.1 Register User

Use case ID	UC_1
Actors	Student Company Professor Traineeship Committee
Pre conditions	The user must have selected the option to register a new account
Main flow of events	1. The use case begins when the user chooses the register option 2. The system displays the registration form 3. The user inserts the required information 4. The user clicks on the “Sign Up” button 5. The system creates a new account
Post conditions	The user’s account is saved to the database

3.2 Login User

Use case ID	UC_2
Actors	Student Company Professor Traineeship Committee
Pre conditions	The user should already have an account and must have selected the option to log in to the account.
Main flow of events	1. The use case begins when the user chooses the login option 2. The system displays the login form

	3. The user enters their credentials 4. The user clicks on the “Sign in” button 5. If valid, the system redirects the user to the dashboard
Post conditions	The user is authenticated and has access to the application’s functionalities

3.3 Logout User

Use case ID	UC_3
Actors	Student Company Professor Traineeship Committee
Pre conditions	The user is logged in to the traineeship app
Main flow of events	1. The use case begins when the user has ended their session 2. The user selects the log out option 3. Once selected, the system redirects the user back to the login/register page
Post conditions	The user no longer has access to the application

3.4 Create Student Profile

Use case ID	UC_4
Actors	Student
Pre conditions	The student must be logged into their account.
Main flow of events	1. The use case begins when the student selects the “Create Profile” button 2. The system displays the profile form 3. The student fills in full name, university ID number, interests, skills, and preferred location 4. The student submits the form by clicking the button “Save Profile” 5. The system validates the input and creates the student profile

Alternative Flow	1. The use case begins when the company member clicks on the “Profile” button
Post conditions	The student profile is saved to the database and linked to the student's account

3.5 Apply For Traineeship

Use case ID	UC_5
Actors	Student
Pre conditions	The student must be logged into their account The student must have created a profile
Main flow of events	1. The use case begins when the student clicks on the button “Traineeship Application” from the dashboard 2. The system displays the option to apply for the traineeship 3. The user clicks on “Submit Application” button 4. The system accepts the student’s request 5. The system displays a message to notify the student
Alternative Flow 1	1. The use case begins when the student clicks on the button “Traineeship Application” from the dashboard but does not have created a profile 2. The system displays the option to apply for the traineeship 3. The user clicks on “Submit Application” button 4. The system denies the student’s request and displays a message to notify the user
Alternative Flow 2	1. clicks on the button “Traineeship Application” from the dashboard but already assigned 2. The system displays the option to apply for the traineeship 3. The user clicks on “Submit Application” button 4. The system denies the student’s request and displays a message to notify the user
Alternative Flow 2	1. clicks on the button “Traineeship Application” from the dashboard but already applied for traineeship 2. The system displays the option to apply for the traineeship

	3. The user clicks on “Submit Application” button 4. The system denies the student’s request and displays a message to notify the user
Post conditions	The application status changes in the database and the trainee committees can now see the student on the pending applications page.

3.6 Fill Traineeship Logbook Report (TODO:)

Use case ID	UC_6
Actors	Student
Pre conditions	The student already assigned to a position
Main flow of events	1. The user clicks on the button “Logbook” from the dashboard 2. The system displays a form with a text area and a save button
Alternative Flow	1. The user clicks on the button “Logbook” from the dashboard without be assigned to a traineeship position 2. The system displays an error message to the student
Post conditions	The text that the user wrote is now saved in the database

3.7 Create Company Profile

Use case ID	UC_7
Actors	Company
Pre conditions	The company member must me logged in the app
Main flow of events	1. The use case begins when the company member clicks on the “Profile” button 2. The system displays the profile form 3. The company member enters its name and location 4. The company member submits the form 5. The system validates and creates the company profile 6. The company profile is saved to the database

Alternative Flow	2. The use case begins when the company member clicks on the “Profile” button 3. The system displays a form with the required fields 4. The company member fills in the form 5. The company member clicks on the “Cancel” button
Post conditions	The system redirects the company member back to the dashboard

3.8 Access List of Advertised Positions

Use case ID	UC_8
Actors	Company
Pre conditions	The company must me logged in to the app
Main flow of events	1. The use case begins when the company member clicks on the button “Available Traineeship Positions” to access the list of advertised traineeship positions 2. The system displays the list from the database with the “Delete” action
Post conditions	

3.9 Access List of Assigned Positions

Use case ID	UC_9
Actors	Company
Pre conditions	The company member must me logged in to the app
Main flow of events	1. The use case begins when the company member clicks on the button “Traineeship Positions in Progress” to access the list with the assigned traineeship positions 2. The system displays the list from the database with the action “Evaluate Student”
Post conditions	

3.10 Create New Traineeship Position

Use case ID	UC_10
Actors	Company
Pre conditions	The company member must have click on the "Traineeship Positions in Progress" button
Main flow of events	1. The use case begins when the company member clicks on the button "Create New Position" 2. The system displays a form with the required fields 3. The company member fills in the entries 4. The company member clicks on "Create" button 5. The system processes the action and saves the position into the database
Alternative Flow	1. The use case begins when the company member clicks on the button "Create New Position" 2. The system displays a form with the required fields 3. The company member fills in the entries 4. The company member clicks on "Cancel" button 5. The system returns the company member back to the advertised positions page
Post conditions	A new traineeship position has been added to the database

3.11 Delete a Traineeship Position

Use case ID	UC_11
Actors	Company
Pre conditions	The company member must have click on the "Available Traineeship Positions" button
Main flow of events	1. The use case begins when the company member clicks on the button "Delete" in the "Actions" column of the table with the advertised traineeship positions that is displayed 2. The system processes the request and deletes the traineeship position from the database

Post conditions	The traineeship position no longer exists in the database

3.12 Evaluate Student

Use case ID	UC_12
Actors	Company
Pre conditions	The company has at least one assigned student and the company member has clicked on the button “Traineeship Positions in Progress”
Main flow of events	1. The company member clicks on the button “Evaluate student” which is in the cells of the column “Action” 2. The system displays a table for the evaluation and for each category, it has got a scale from 1 to 5 3. The company member fills the grade for each category and clicks on the “Submit Evaluation” button 4. The system saves the evaluation into the database
Post conditions	The system redirects the company member back to the “Traineeship Positions in Progress” page

3.13 Create Professor Profile

Use case ID	UC_6
Actors	Professor
Pre conditions	The professor must be logged in to his account
Main flow of events	1. The use case begins when the professor clicks on the “Profile” button 2. The system displays a form with the required fields 3. The professor fills in the form 4. The professor clicks on the “Save” button 5. The system saves the information of the professor to the database
Alternative Flow	1. The use case begins when the professor clicks on the “Profile” button 2. The system displays a form with the required fields 3. The professor fills in the form 4. The professor clicks on the “Cancel” button

Post conditions	The system redirects the professor back to the dashboard

3.14 Access Supervised Positions

Use case ID	UC_14
Actors	Professor
Pre conditions	The professor must be logged in
Main flow of events	1. The use case begins when the professor selects “My Supervised Traineeships” 2. The system retrieves the list of traineeship positions supervised by the professor 3. The system displays the list
Post conditions	

3.15 Evaluate Supervising Position

Use case ID	UC_15
Actors	Professor
Pre conditions	The professor must supervise at least one assigned student.
Main flow of events	1. The use case begins when the professor selects a supervised traineeship for evaluation 2. The system displays the evaluation form 3. The professor rates the student on motivation, effectiveness, and efficiency (1 to 5) 4. The professor evaluates the company on facilities and trainee guidance (1 to 5) 5. The professor submits the evaluation

	6. The system saves the evaluation linked to the traineeship and student
Post conditions	The evaluation is stored in the system

3.16 Evaluate Supervising Position

Use case ID	UC_16
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee member must be logged in
Main flow of events	1. The use case begins when the traineeship committee member clicks on the “Students Applications” button 2. The system displays a list with the students that have applied for traineeship
Post conditions	

3.17 Display The Students Who Applied For a Traineeship Position

Use case ID	UC_16
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee member must be logged in
Main flow of events	1. The use case begins when the traineeship committee member clicks on the “Students Applications” button 2. The system displays a list with the students that have applied for traineeship
Post conditions	

3.18 Select a Student to Assign

Use case ID	UC_17
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee member must be in “Student Applications” page
Main flow of events	1. The use case begins when the traineeship committee member clicks on the “Select” button in the column “Actions” on the student they want 2. The system redirects the traineeship committee member to the search of the positions page
Post conditions	The system displays the search page of the positions

3.19 Search for Traineeship Positions

Use case ID	UC_18
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee member must have selected a student and have been redirected automatically to the search of position page by the system
Main flow of events	1. The use case begins when the traineeship committee member selects an option of the three that are available 2. The traineeship committee member clicks on the “Search” button 3. The system displays the positions that match the requirements based on the option that has been selected before
Post conditions	A list of the positions is displayed with a “Select” button

3.20 Select Traineeship Position

Use case ID	UC_19
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee member must have searched for positions
Main flow of events	1. The use case begins when the traineeship committee member has chosen which position want and clicks on the “Select” button 2. The system redirects the traineeship committee member to search for supervisor
Post conditions	The position has been selected temporally, till the process of assignment has been finished or cancelled

3.21 Search for Supervisor

Use case ID	UC_20
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee member must have selected a position and have been redirected automatically to the search of position page by the system
Main flow of events	1. The use case begins when the traineeship committee member an option of the two that are available 2. The traineeship committee member clicks on the “Search” button 3. The system displays the professors that match the requirements based on the option that has been selected before
Post conditions	A list of the professors is displayed with a “Select” button

3.22 Select a Professor for Supervisor

Use case ID	UC_21
Actors	Traineeship Committee Member

Pre conditions	The traineeship committee members must search for professors
Main flow of events	1. The use case begins when the traineeship committee member has chosen which professor they want 2. The traineeship committee member clicks on the “Select” button near of the chosen professor in the “Actions” column 3. The system redirects the traineeship committee member to the page where details of the chosen student, position and supervisor
Post conditions	The system display 3 cards with information about the selected student, position and supervisor for verification purposes

3.23 Assign Student and Supervisor to the Selected Position

Use case ID	UC_22
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee members must select the supervisor
Main flow of events	1. The use case begins when the traineeship committee member clicks on “Assign Student” button 2. The system assigns professor and student to the selected position
Post conditions	The system connects the professor and the student with the position

3.24 Display Traineeship Positions in Progress

Use case ID	UC_23
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee members must be logged in to the app
Main flow of events	1. The use case begins when the traineeship committee member clicks on “Traineeships in Progress” button

	2. The system displays a list with all the traineeship positions that are in progress
Post conditions	The system displays a table with all the traineeship positions that are in progress and have a “Review Traineeship” button

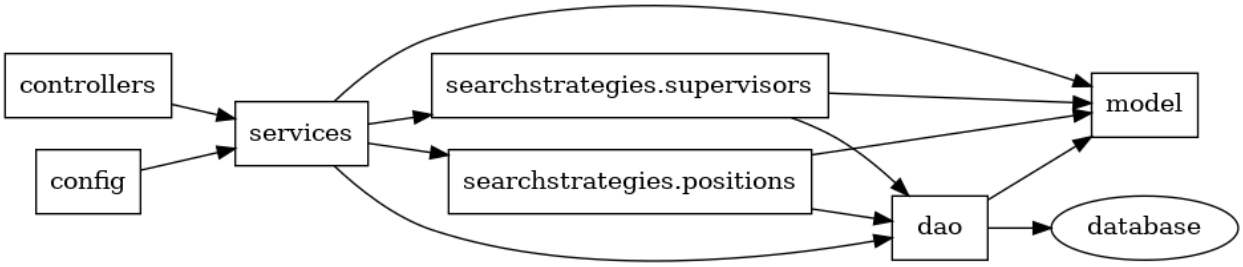
3.25 Select Traineeship Positions in Progress

Use case ID	UC_24
Actors	Traineeship Committee Member
Pre conditions	The traineeship committee members must be in “Traineeship Positions in Progress” page
Main flow of events	1. The use case begins when the traineeship committee member clicks on “Review Evaluation” button of a position that is in progress 2. The system displays the company and supervisor’s evaluations and two options “Pass” and “Fail” grade 3. The traineeship committee member clicks on “Pass” grade or the “Fail” grade based on the evaluations 4. The system saves this option
Post conditions	The system saves the option to the database and redirects the traineeship committee member back to the “Traineeship Positions in Progress” page

4 Design

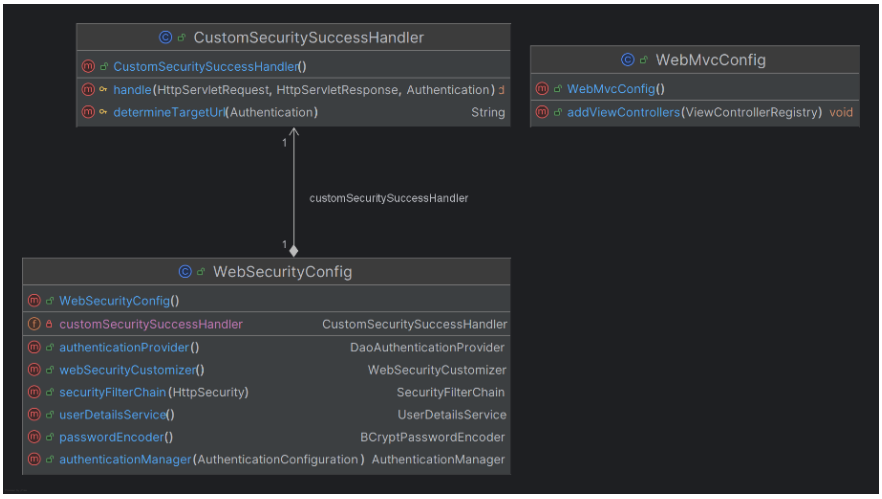
4.1 Architecture

Package UML Diagram

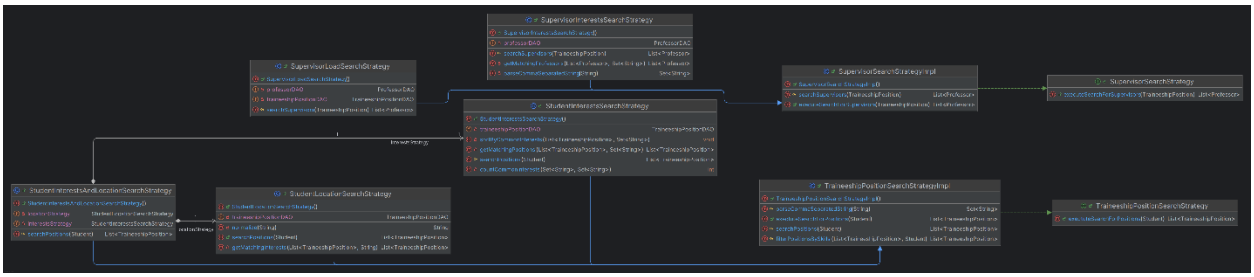


4.2 Design

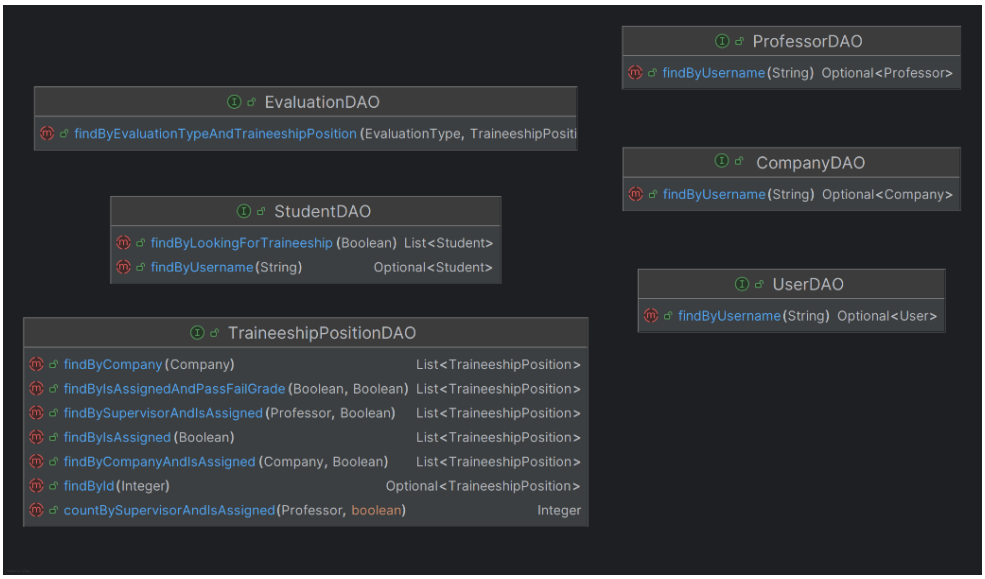
Config Package



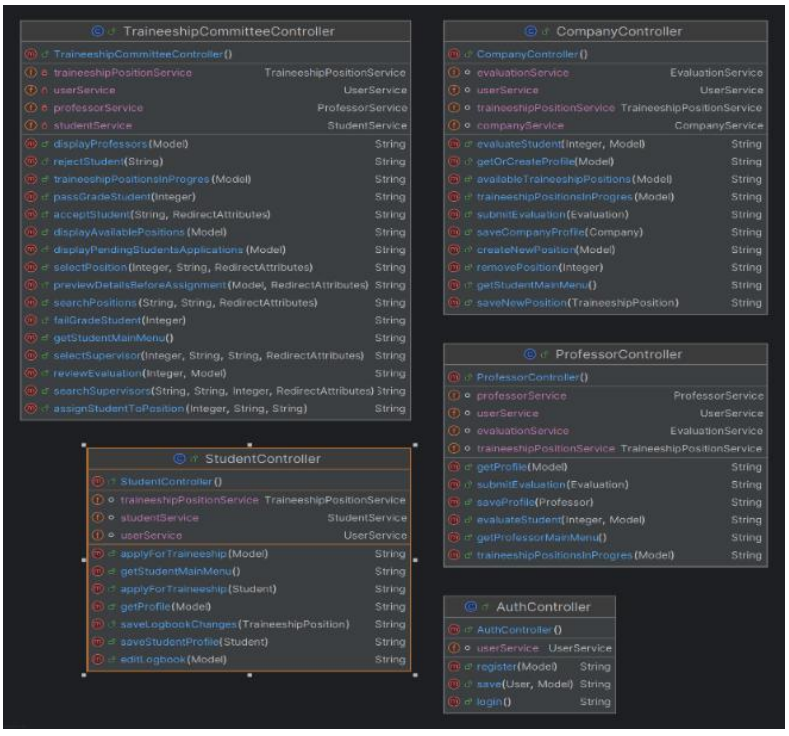
Searchstrategies Package



Dao Packages

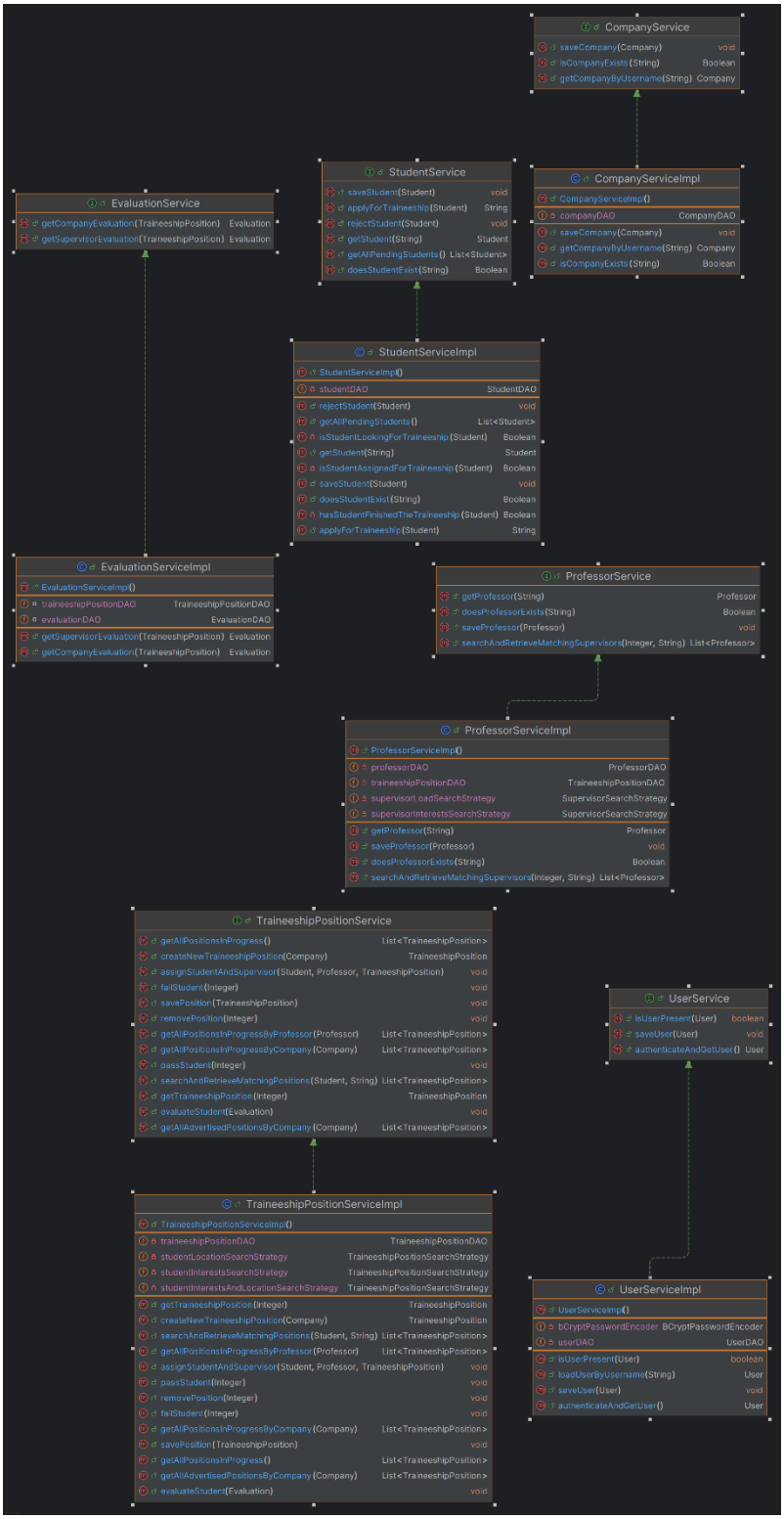


Controllers Package



Model Package

Services Package



Class Name: CustomSecuritySuccessHandler	
Responsibilities: ▪ Decide which dashboard URL matches the authenticated user's role and send an HTTP redirect	**Collaborations:** ▪ WebSecurityConfig

Class Name: WebMvcConfig	
Responsibilities: ▪ Add a view controller that maps the root path "/" straight to the *homepage* view (no explicit controller method needed).	**Collaborations:**

Class Name: WebSecurityConfig	
Responsibilities: ▪ Declares SecurityFilterChain ▪ Authentication provider ▪ Password encoder ▪ Path/role rules	**Collaborations:** ▪ UserServiceImpl ▪ CustomSecuritySuccessHandler

Class Name: AuthController	
Responsibilities: ▪ Serve *login* page ▪ *Register* page ▪ Persist a new User on "/save"	**Collaborations:** ▪ UserService

Class Name: ProfessorController	
Responsibilities: ▪ Professor dashboard ▪ Profile CRUD ▪ List of supervising positions	**Collaborations:** ▪ UserService ▪ ProfessorService ▪ TraineeshipPositionService

▪ Supervisor's evaluation of a student	▪ EvaluationService

Class Name: StudentController	
Responsibilities: ▪ Student dashboard ▪ Profile CRUD ▪ Traineeship application workflow ▪ Log-book editing	**Collaborations:** ▪ UserService ▪ StudentService ▪ TraineeshipPositionService

Class Name: CompanyController	
Responsibilities: ▪ Company dashboard ▪ Profile management ▪ Advertise / delete traineeship positions ▪ Submit company evaluations.	**Collaborations:** ▪ UserService ▪ StudentService ▪ TraineeshipPositionService

Class Name: TraineeshipCommitteeController	
Responsibilities: ▪ Review student applications ▪ Search positions ▪ Search supervisors ▪ Assign students ▪ Review evaluations ▪ Set pass/fail.	**Collaborations:** ▪ UserService ▪ StudentService ▪ ProfessorService ▪ TraineeshipPositionService

Class Name: TraineeshipPositionDAO	
Responsibilities: ▪ Supplies extra derived-query methods such as *findByCompanyAndIsAssigned*, *countBySupervisorAndIsAssigned*, etc.	**Collaborations:**

Class Name: EvaluationDAO	
Responsibilities: ▪ Adds a finder that pairs **type** and **position**.	**Collaborations:**

Class Name: StudentDAO	
Responsibilities: ▪ Custom methods to get by username ▪ Returns a list of students searching for traineeships.	**Collaborations:**

Class Name: ProfessorDAO	
Responsibilities: ▪ Exposes *findByUsername*.	**Collaborations:**

Class Name: CompanyDAO	
Responsibilities: ▪ Exposes *findByUsername*.	**Collaborations:**

Class Name: Role	
Responsibilities: ▪ Hold the four possible application roles (Student, Professor, Company, Traineeship-committee member) ▪ Expose their display value.	**Collaborations:** ▪ User

Class Name: User	
Responsibilities: ▪ Persist credentials & role ▪ Implement Spring-Security's UserDetails API (authorities, username, password).	**Collaborations:**

Class Name: Student	
Responsibilities: ▪ Student profile & application state: personal data, interests, skills, location ▪ Current traineeship position ▪ Whether the student is still looking.	**Collaborations:** ▪ TraineeshipPosition

Class Name: Professor	
Responsibilities: ▪ Professor's profile ▪ The list of positions they supervise.	**Collaborations:** ▪ TraineeshipPosition

Class Name: Company	
Responsibilities: ▪ Company profile (name, location) ▪ The traineeship positions it advertises.	**Collaborations:** ▪ TraineeshipPosition

Class Name: TraineeshipPosition	
Responsibilities: ▪ Full description of an internship slot: metadata (title, dates, topics, required skills) ▪ Assignment state ▪ Log-book ▪ Pass/fail grade ▪ Links to student, supervisor, company ▪ Up to two evaluations.	**Collaborations:** ▪ Student ▪ Professor ▪ Company ▪ Evaluation

Class Name: EvaluationType	
Responsibilities: ▪ Distinguish **COMPANY** vs **SUPERVISOR** evaluation ▪ Offer helper predicates	**Collaborations:** ▪ TraineeshipPosition

Class Name: Evaluation	
Responsibilities: ▪ Record three numeric scores (motivation/efficiency/effectiveness) for a given position and type	**Collaborations:** ▪ EvaluationType ▪ TraineeshipPosition

Class Name: TraineeshipPositionSearchStrategy (Interface)	
Responsibilities: ▪ declare a single method to return positions that suit a given student based on the search method	**Collaborations:** ▪ EvaluationType ▪ TraineeshipPosition

Class Name: TraineeshipPositionSearchStrategyImpl (abstract template)	
Responsibilities: ▪ Concrete subclasses supply *searchPositions*, the template then filters those results so the student's skills cover all required skills, with a utility to parse comma-separated strings.	**Collaborations:** ▪ Student ▪ TraineeshipPosition

Class Name: StudentLocationSearchStrategy	
Responsibilities: ▪ Fetch unassigned positions, keep those whose company location contains the student's preferred location	**Collaborations:** ▪ TraineeshipPositionDAO ▪ Student ▪ TraineeshipPosition

Class Name: StudentInterestsSearchStrategy	
Responsibilities: ▪ Fetch unassigned positions, keep those whose company location contains the student's preferred location	**Collaborations:** ▪ TraineeshipPositionDAO ▪ Student ▪ TraineeshipPosition

Class Name: StudentInterestsAndLocationSearchStrategy	
Responsibilities: ▪ Combine the previous two strategies: intersect the location-matched and interests-matched lists	**Collaborations:** ▪ StudentLocationSearchStrategy ▪ StudentInterestsSearchStrategy ▪ TraineeshipPosition ▪ Student

Class Name: SupervisorSearchStrategy (interface)	
Responsibilities: ▪ Contract for algorithms that turn a position into a ranked list of potential supervisors	**Collaborations:** ▪ Professor ▪ TraineeshipPosition

Class Name: SupervisorSearchStrategyImpl *(abstract template)*	
Responsibilities: ▪ Delegate the real search to *searchSupervisors* implemented by subclasses and simply returns the list	**Collaborations:** ▪ Professor ▪ TraineeshipPosition

Class Name: SupervisorSearchStrategyImpl (abstract template)	
Responsibilities: ▪ Delegate the real search to *searchSupervisors* implemented by subclasses and simply returns the list	**Collaborations:** ▪ Professor ▪ TraineeshipPosition

Class Name: SupervisorInterestsSearchStrategy	
Responsibilities: ▪ Select professors whose research interests fully cover the position's topics	**Collaborations:** ▪ Professor ▪ TraineeshipPosition ▪ ProfessorDAO

Class Name: SupervisorLoadSearchStrategy	
Responsibilities: ▪ Rank professors by ascending current supervising load (count of assigned positions), using a DAO-level aggregate	**Collaborations:** ▪ ProfessorDAO ▪ TraineeshipPositionDAO ▪ TraineeshipPosition ▪ Professor

Class Name: SupervisorLoadSearchStrategy	
Responsibilities: ▪ Rank professors by ascending current supervising load (count of assigned positions), using a DAO-level aggregate	**Collaborations:** ▪ ProfessorDAO ▪ TraineeshipPositionDAO ▪ TraineeshipPosition ▪ Professor

Class Name: UserService (interface)	
Responsibilities: ▪ Persisting a new user ▪ Check if a username already exists and return the currently authenticated User	**Collaborations:** ▪ User

Class Name: UserServiceImpl	
Responsibilities: ▪ Encrypt passwords with BCrypt and save users ▪ Expose authenticateAndGetUser() by reading the Spring-Security context ▪ Supply loadUserByUsername() to the authentication manager	**Collaborations:** ▪ UserDAO

Class Name: CompanyService (interface)	
Responsibilities: ▪ Save company data ▪ Look professors up ▪ Fetch by username.	**Collaborations:** ▪ Company

Class Name: CompanyService	
Responsibilities: ▪ implement the methods of the interface	**Collaborations:** ▪ CompanyDAO ▪ Company

Class Name: ProfessorService (interface)	
Responsibilities: ▪ Save professor data ▪ Look professors up ▪ Test existence ▪ Produce a ranked list of potential supervisors for a traineeship position based on a chosen search option.	**Collaborations:** ▪ Professor ▪ TraineeshipPosition

Class Name: ProfessorServiceImpl	
Responsibilities: ▪ Implements the methods of the interface	**Collaborations:** ▪ ProfessorDAO ▪ TraineeshipPositionDAO ▪ SupervisorSearchStrategy ▪ Professor ▪ TraineeshipPosition

Class Name: StudentService (interface)	
Responsibilities: ▪ Save student data ▪ Apply for traineeship ▪ Reject student ▪ List of pending requests	**Collaborations:** ▪ Student

Class Name: StudentServiceImpl	
Responsibilities: ▪ Implement the methods of the interface	**Collaborations:** ▪ StudentDAO ▪ Student ▪ TraineeshipPosition

Class Name: EvaluationService (interface)	
Responsibilities: ▪ Fetch (or initialise) the *company* and *supervisor* evaluations attached to a given traineeship position	**Collaborations:** ▪ Evaluation ▪ TraineeshipPosition

Class Name: EvaluationServiceImpl	
Responsibilities: ▪ Implement the method of the interface	**Collaborations:** ▪ EvaluationDAO ▪ TraineeshipPositionDAO ▪ Evaluation ▪ EvaluationType ▪ TraineeshipPosition

Class Name: TraineeshipPositionService (interface)	
Responsibilities: ▪ Create new positions ▪ Remove a traineeship position ▪ Show all positions to a professor that he is supervising ▪ Show all positions to a company that are in progress ▪ Perform search matching for students ▪ Assign student and supervisor ▪ Record the evaluations ▪ Mark pass / fail	**Collaborations:** ▪ Company ▪ Student ▪ Professor ▪ Evaluation ▪ TraineeshipPosition

Class Name: TraineeshipPositionService	
Responsibilities: ▪ Implement the methods of the interface	**Collaborations:** ▪ TraineeshipPositionDAO ▪ TraineeshipPositionSearchStrategy ▪ Company ▪ Student ▪ Professor ▪ Evaluation ▪ TraineeshipPosition