SOFTWARE DEVELOPMENT II

PROJECT REPORT:

TEAM

Vasileios Papakyriakou AM: 5324
Anna Tarasidou AM: 5361

TABLE OF CONTENTS

Εισαγωγή

Σκοπός της παρούσας εργασίας ήταν η αναδιοργάνωση (reengineering) και η επέκταση μιας υπάρχουσας εφαρμογής Java, η οποία διαχειρίζεται ένα σύστημα πρακτικής άσκησης.
Η **πρώτη φάση** του έργου εστιάστηκε στην ανάλυση της αρχιτεκτονικής της υπάρχουσας εφαρμογής (legacy code). Μελετήθηκε ο υπάρχων κώδικας, τα user stories και οι λειτουργίες που ήταν ήδη υλοποιημένες, εντοπίζοντας σημεία που έχρηζαν βελτίωσης ως προς τη δομή και τη συντηρησιμότητα.
Στη **δεύτερη φάση**, προχωρήσαμε σε εκτεταμένο refactoring του κώδικα για τη βελτίωση της αποδοτικότητας και της ποιότητάς του, καθώς και στην υλοποίηση των user stories που εκκρεμούσαν. Παράλληλα, αναδιαμορφώσαμε το frontend ώστε να προσφέρει μια πιο φιλική και εύχρηστη εμπειρία πλοήγησης στον χρήστη.
Για την επίτευξη των παραπάνω, εφαρμόστηκαν συγκεκριμένες τεχνικές Refactoring και Design Patterns:

- **Extract Class:** Για τη διάσπαση της κλάσης Controller σε μικρότερες κλάσεις με διακριτές αρμοδιότητες.
- **Extract Service / Move Method:** Για τη μεταφορά της επιχειρησιακής λογικής (business logic) από τους Controllers σε ένα διακριτό επίπεδο υπηρεσιών (Service Layer).
- **Template Method:** Για την εξάλειψη διπλότυπου κώδικα και την εφαρμογή πολυμορφισμού στις στρατηγικές αναζήτησης και ανάθεσης.

Επιπλέον, υιοθετήθηκε το **Strategy Pattern** για την ευέλικτη διαχείριση των αλγορίθμων αναζήτησης και ανάθεσης.
Μέσω αυτών των τεχνικών επιλύθηκαν προβλήματα όπως ο επαναλαμβανόμενος κώδικας και η παραβίαση της αρχής της μοναδικής ευθύνης, οδηγώντας σε μια πιο καθαρή και επεκτάσιμη αρχιτεκτονική (Clean Code). Τέλος, η εφαρμογή εμπλουτίστηκε με επιπλέον λειτουργίες που καλύπτουν τις ανάγκες όλων των ρόλων χρηστών.

THE TRAINEESHIP APP PROJECT

Traineeship App
Η εφαρμογή αποτελεί ένα ολοκληρωμένο σύστημα διαχείρισης πρακτικής άσκησης που συντονίζει τη συνεργασία μεταξύ τεσσάρων ρόλων χρηστών: Φοιτητών, Εταιρειών, Καθηγητών και της Επιτροπής Πρακτικής Άσκησης. Μέσω της πλατφόρμας, οι Εταιρείες έχουν τη δυνατότητα να δημοσιεύουν και να διαχειρίζονται θέσεις πρακτικής, ενώ η Επιτροπή αναλαμβάνει τον κεντρικό ρόλο της αντιστοίχισης των Φοιτητών σε αυτές, αξιοποιώντας δυναμικές στρατηγικές αναζήτησης βάσει τοποθεσίας ή ακαδημαϊκών ενδιαφερόντων. Επιπρόσθετα, το σύστημα αυτοματοποιεί την ανάθεση εποπτών Καθηγητών στις θέσεις, λαμβάνοντας υπόψη κριτήρια όπως ο τρέχων φόρτος εργασίας ή η συνάφεια του γνωστικού αντικειμένου, και ολοκληρώνει τον κύκλο της πρακτικής με την υποβολή αξιολογήσεων από τους εμπλεκόμενους φορείς.

ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Application architecture after reengineering

ΛΕΠΤΟΜΕΡΕΙΕΣ ΣΧΕΔΙΑΣΜΟΥ – REFACTORING TASKS

Σε αυτό το κομμάτι της αναφοράς εξηγούμε ακριβώς τις αλλαγές και το refactoring που έχει γίνει σε κάθε κλάση και ακολουθούν τα UML διαγράμματα των κλάσεων αυτών και όλων των κλάσεων του κάθε πακέτου.

TraineeshipAppController Class

Το βασικό πρόβλημα με την κλάση TraineeshipAppController ήταν ότι έκανε τα πάντα. Λειτουργούσε ουσιαστικά ως μια **"God Class"**, αφού διαχειριζόταν ταυτόχρονα όλους τους ρόλους χρηστών (Student, Professor, Company, Committee) και όλη την επιχειρησιακή λογική. Αυτό δημιουργούσε μεγάλο μπέρδεμα και παραβίαζε την αρχή **Single Responsibility Principle (SRP)**, κάνοντας τον κώδικα δύσκολο στη συντήρηση.

Για να λύσουμε αυτό το πρόβλημα, "σπάσαμε" τον μεγάλο Controller σε μικρότερα κομμάτια χρησιμοποιώντας την τεχνική **Extract Class**. Πλέον, έχουμε ξεχωριστούς Controllers, όπου ο καθένας αναλαμβάνει αποκλειστικά έναν ρόλο:

1. StudentController
2. ProfessorController
3. CompanyController
4. CommitteeController

Search Strategy Classes

Στο κομμάτι της αναζήτησης θέσεων πρακτικής, παρατηρήσαμε ότι οι κλάσεις SearchBasedOnInterests και SearchBasedOnLocation είχαν αρκετό διπλότυπο κώδικα (code duplication). Και οι δύο επαναλάμβαναν ακριβώς την ίδια διαδικασία για να βρουν τα στοιχεία του φοιτητή από τη βάση και να αρχικοποιήσουν τη λίστα των αποτελεσμάτων, πριν εκτελέσουν την πραγματική αναζήτηση.

Για να το διορθώσουμε αυτό, εφαρμόσαμε το **Template Method Pattern**. Συγκεκριμένα:

1. Δημιουργήσαμε μια νέα αφηρημένη κλάση, την AbstractPositionsSearchStrategy.
2. Μεταφέραμε εκεί τον κοινό κώδικα (εύρεση φοιτητή, διαχείριση λίστας).
3. Ορίσαμε μια "μέθοδο-σκελετό" (search()) που εκτελεί τα βασικά βήματα και καλεί την αφηρημένη μέθοδο findMatchingPositions() για τις λεπτομέρειες.

Έτσι, οι επιμέρους κλάσεις έγιναν πολύ πιο απλές και ξεκάθαρες, εστιάζοντας μόνο στη δική τους λογική:

- Η **SearchBasedOnLocation** πλέον ψάχνει μόνο ποιες εταιρείες βρίσκονται στην περιοχή προτίμησης του φοιτητή και επιστρέφει τις θέσεις τους.
- Η **SearchBasedOnInterests** συγκρίνει απλώς τα keywords των ενδιαφερόντων του φοιτητή με τα αντικείμενα (topics) των διαθέσιμων θέσεων.

Supervisor Assignment Strategies

Ακριβώς το ίδιο μοτίβο ακολουθήσαμε και για την ανάθεση εποπτών καθηγητών. Οι κλάσεις AssignmentBasedOnLoad και AssignmentBasedOnInterests είχαν επαναλαμβανόμενο κώδικα: και οι δύο έπρεπε να βρουν τη θέση πρακτικής και τη λίστα των καθηγητών από τη βάση, και στο τέλος να αποθηκεύσουν την ανάθεση.

Εφαρμόσαμε και εδώ το **Template Method Pattern** για να καθαρίσουμε τον κώδικα. Φτιάξαμε την αφηρημένη κλάση AbstractSupervisorAssignmentStrategy, η οποία αναλαμβάνει την ανάκτηση δεδομένων και αποθήκευση. Οι επιμέρους κλάσεις πλέον έχουν μόνο μία δουλειά: να αποφασίσουν ποιος καθηγητής είναι ο κατάλληλος (μέθοδος selectSupervisor), είτε με βάση τον φόρτο εργασίας είτε με βάση τα ενδιαφέροντα, χωρίς να μπλέκονται με τη βάση δεδομένων.

Service Classes

Το επόμενο πρόβλημα που εντοπίσαμε ήταν ότι έλειπε εντελώς το επίπεδο της επιχειρησιακής λογικής (Business Logic). Οι Controllers καλούσαν απευθείας τους Mappers (δηλαδή τη βάση δεδομένων) για να υλοποιήσουν τα use cases. Αυτό δεν είναι σωστή πρακτική γιατί μπλέκει τη διαχείριση του αιτήματος με την ουσία της λειτουργίας.

Για να το διορθώσουμε, εισάγαμε το **Service Layer**. Δημιουργήσαμε ξεχωριστά Service interfaces και τις υλοποιήσεις τους (classes) για κάθε τύπο χρήστη:

- StudentService / StudentServiceImpl
- CompanyService / CompanyServiceImpl
- ProfessorService / ProfessorServiceImpl
- CommitteeService / CommitteeServiceImpl

Πλέον, οι Controllers αναθέτουν τη δουλειά σε αυτά τα Services, και τα Services με τη σειρά τους επικοινωνούν με τους Mappers, κρατώντας τον κώδικα οργανωμένο και καθαρό.

TraineeshipPosition Class

Στην αρχική έκδοση, η κλάση αυτή ήταν απλώς μια αποθήκη δεδομένων και δεν υποστήριζε ολόκληρο τον κύκλο ζωής μιας πρακτικής.

Στο πλαίσιο του refactoring και των νέων user stories, εμπλουτίσαμε την κλάση TraineeshipPosition προσθέτοντας νέα πεδία και λογική, όπως το αν έχει ολοκληρωθεί η πρακτική (isCompleted), τον βαθμό (passFailGrade) και το ημερολόγιο του φοιτητή (studentLogbook). Πλέον, η κλάση δεν κρατάει απλώς δεδομένα, αλλά διαχειρίζεται και την κατάστασή της (π.χ. πότε θεωρείται ολοκληρωμένη).

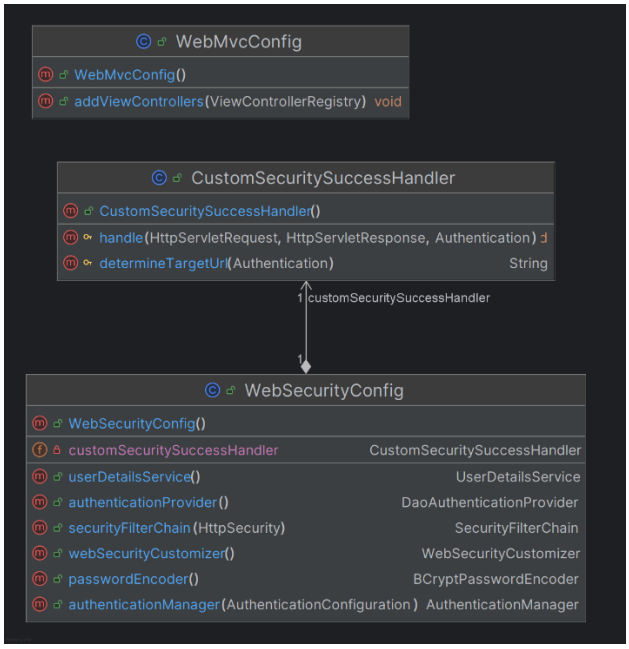
Other refactors and fixes

Πέρα από τις μεγάλες αλλαγές στην αρχιτεκτονική, κάναμε και μια σειρά από μικρότερες αλλά ουσιαστικές βελτιώσεις για να κάνουμε τον κώδικα πιο ασφαλή και λειτουργικό:

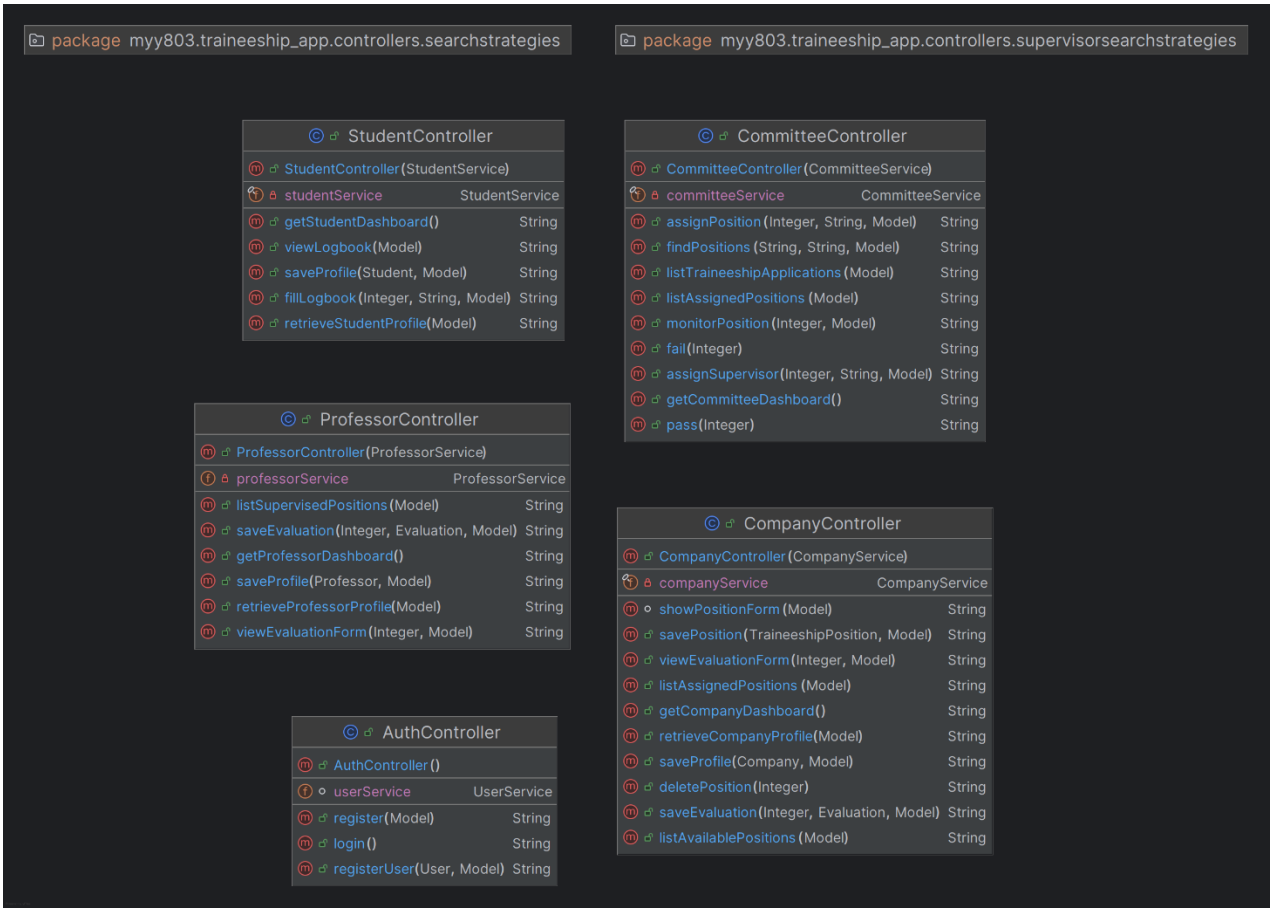
- Για την αποφυγή σφαλμάτων, αντί για απλό .get(), χρησιμοποιούμε πλέον .orElseThrow(), ώστε αν δεν βρεθεί κάτι (π.χ. μια θέση πρακτικής), να εμφανίζεται σωστό μήνυμα λάθους και να μην κρασάρει η εφαρμογή.
- Προσθέσαμε δικλείδες ασφαλείας, όπως την απαγόρευση αξιολόγησης σε θέσεις που δεν έχουν ανατεθεί, προστατεύοντας τη βάση από λάθη.
- Ρυθμίσαμε την εφαρμογή να ανακατευθύνει αυτόματα τον χρήστη στο επόμενο λογικό βήμα μετά από κάθε ενέργεια.
- Προσθέσαμε νέα πεδία όπως το isCompleted στην κλάση TraineeshipPosition, ώστε να παρακολουθούμε σωστά αν μια πρακτική έχει ολοκληρωθεί και βαθμολογηθεί.

UML CLASS & PACKAGE DIAGRAMS

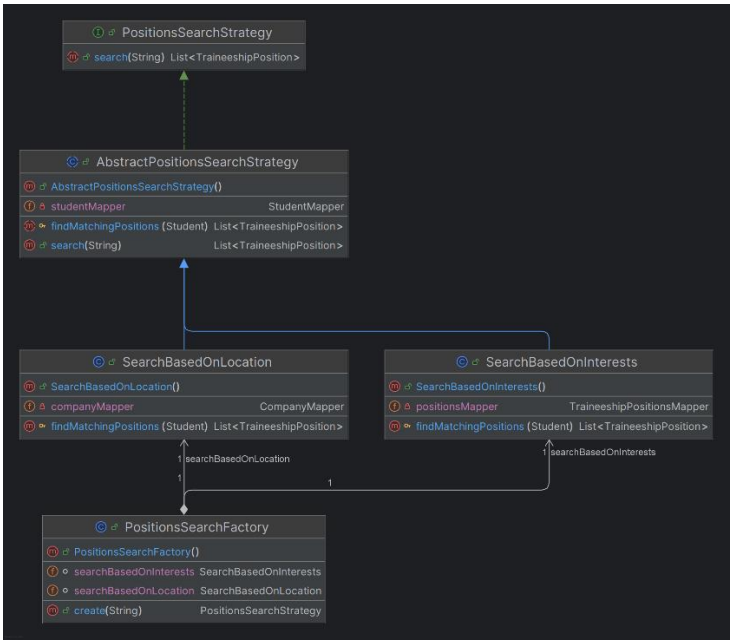
UML package diagram which outlines the general functionality of the application's architecture



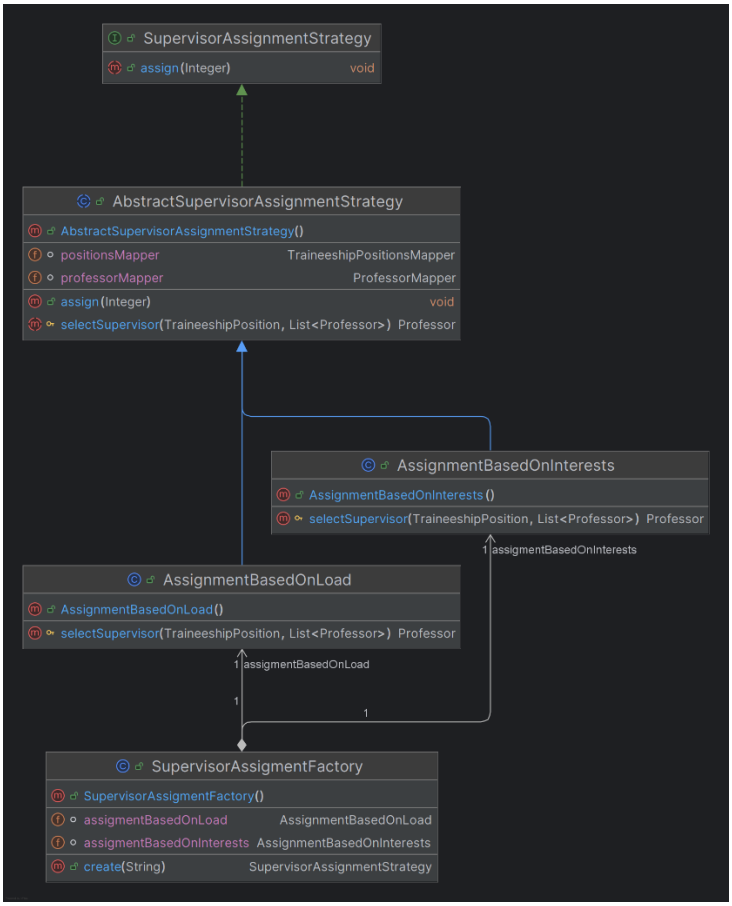
UML class diagram for the classes of the config package



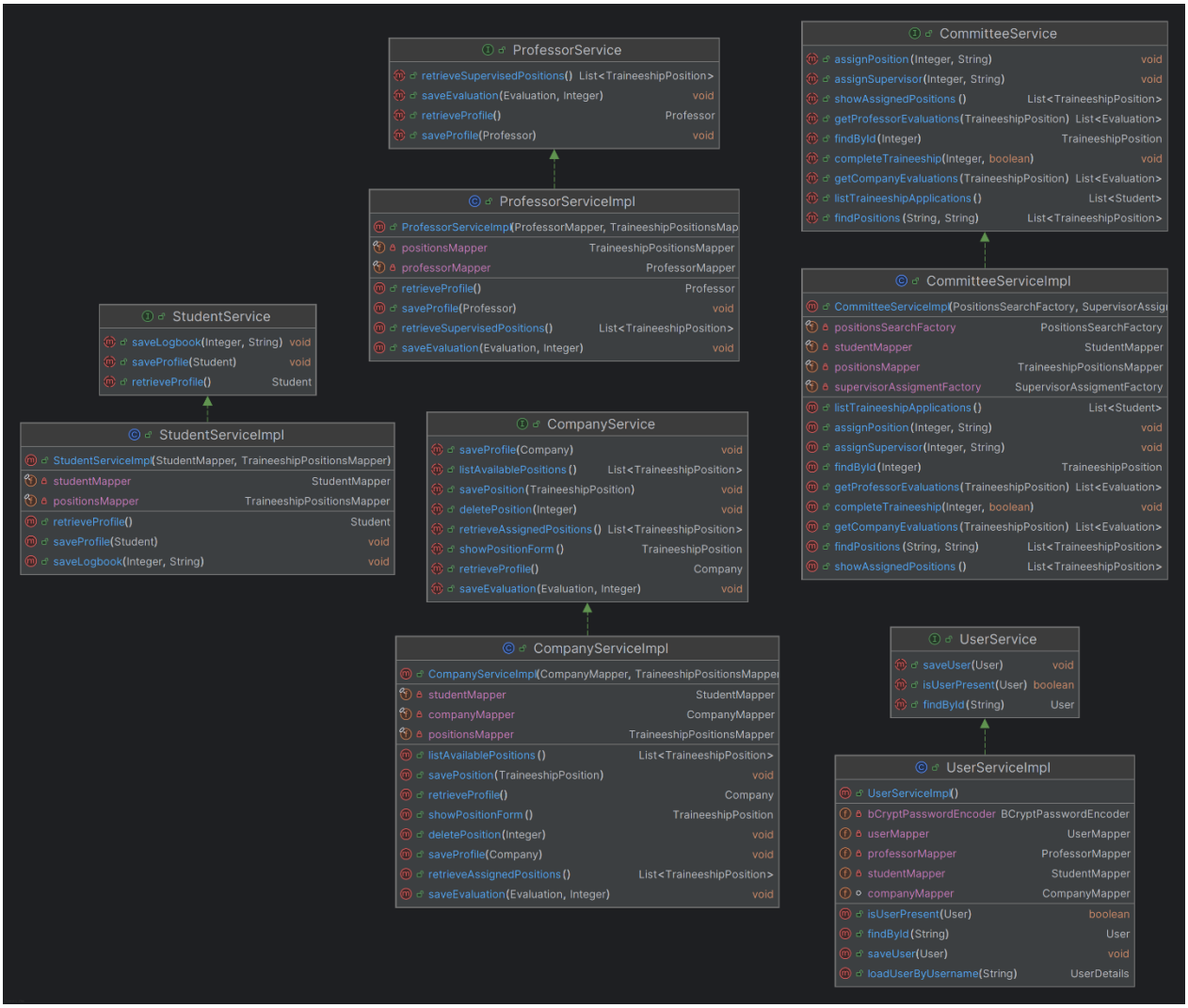
UML class diagram for the classes of the controller package



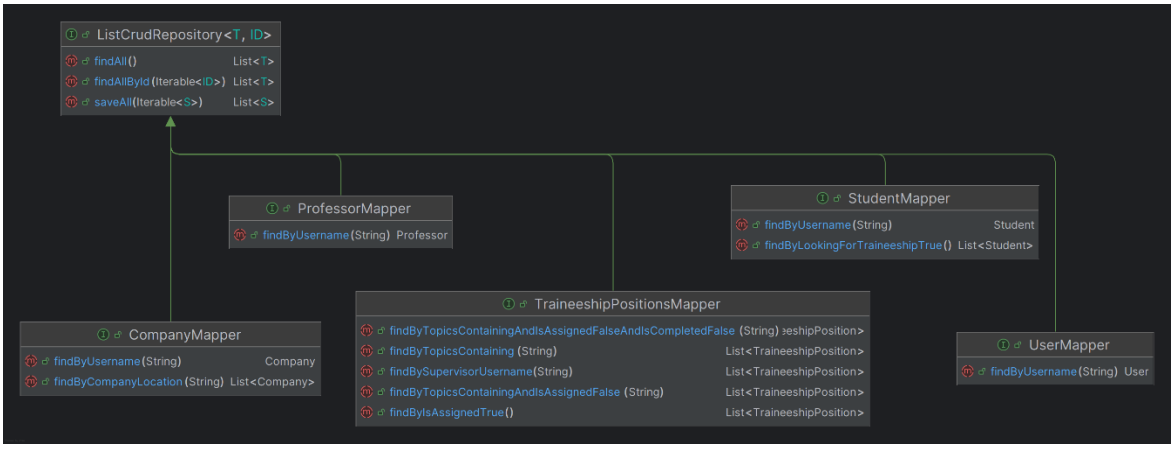
UML class diagram for the classes of the controller.searchstrategies package



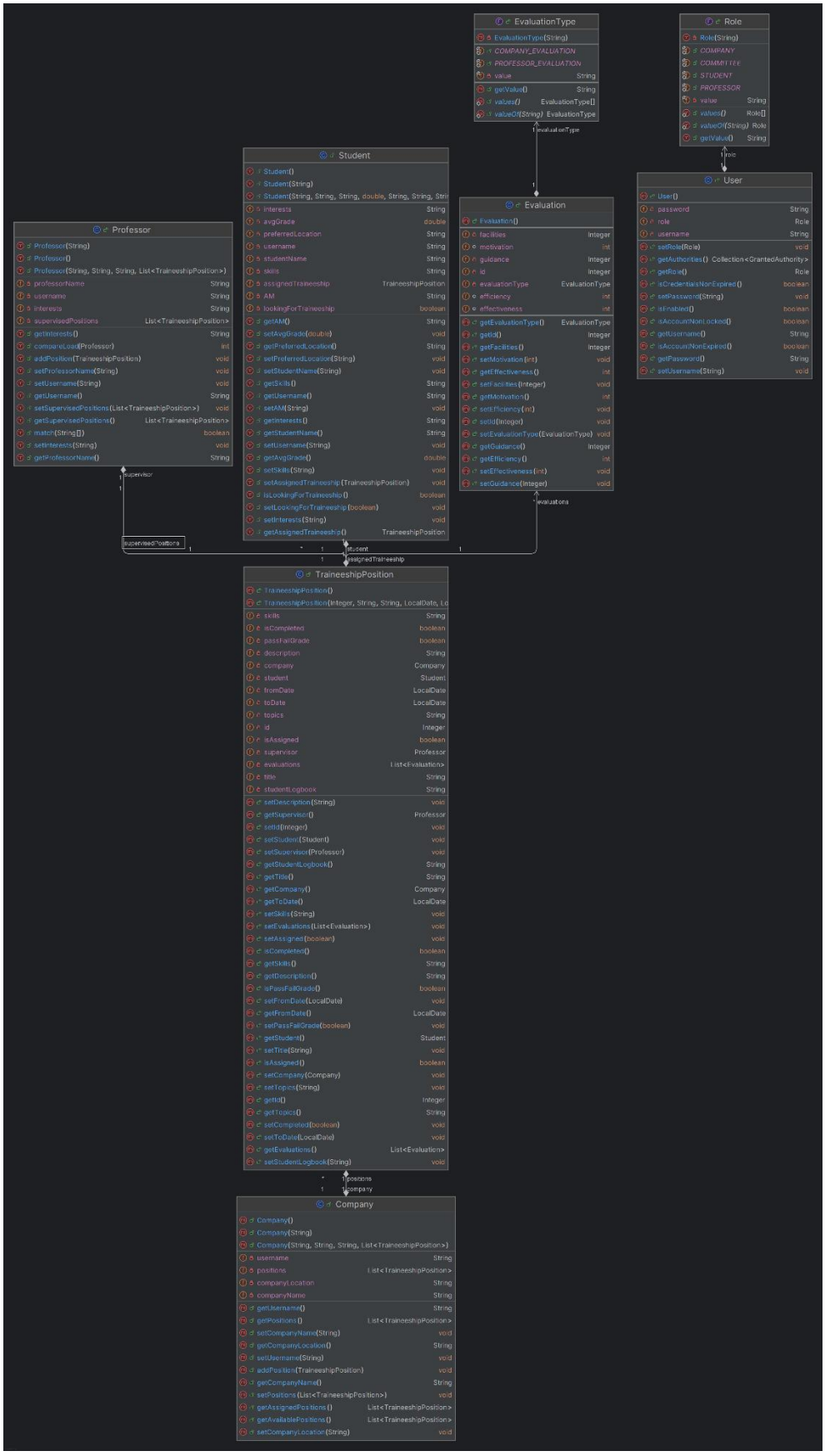
UML class diagram for the classes of the supervisorsearchstrategies package



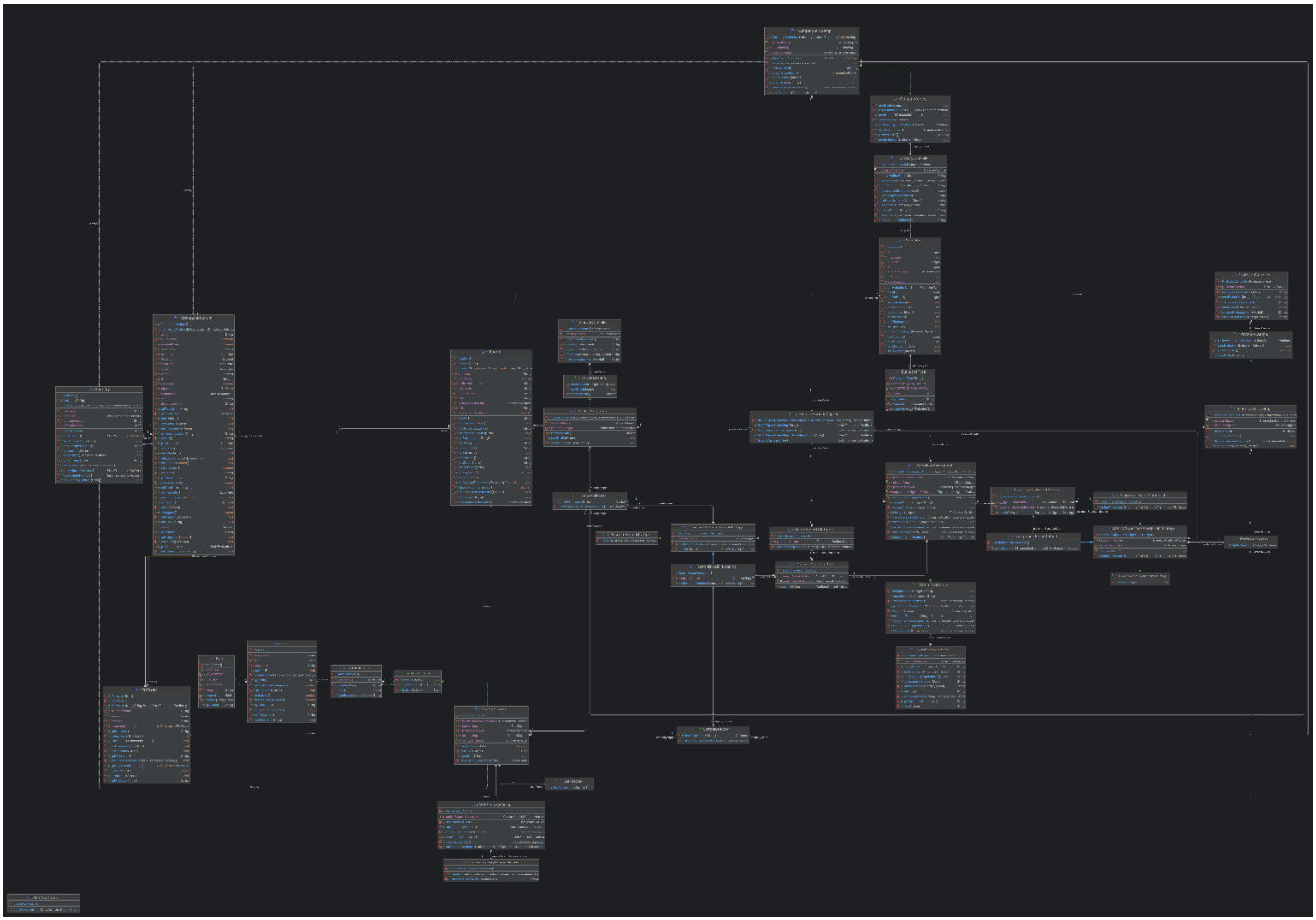
UML class diagram for the classes of the service package



UML class diagram for the classes of the mappers package



UML class diagram for the classes of the domain package



UML class diagram of the whole project together

USE CASES

Use Case 1: Εγγραφή Χρήστη

Use case ID	UC-01
Actors	Μη εγγεγραμμένος Χρήστης (Student, Professor, Company)
Pre conditions	Ο χρήστης δεν πρέπει να είναι συνδεδεμένος.
Main flow of events	1. Ο χρήστης επιλέγει "Εγγραφή" (Register) στην αρχική σελίδα. 2. Το σύστημα εμφανίζει τη φόρμα εγγραφής. 3. Ο χρήστης συμπληρώνει τα στοιχεία του (Username, Password, Role) και υποβάλει τη φόρμα. 4. Το σύστημα κρυπτογραφεί τον κωδικό πρόσβασης. 5. Το σύστημα ελέγχει αν υπάρχει ήδη χρήστης με το ίδιο Username. 6. Το σύστημα αποθηκεύει τον χρήστη και δημιουργεί το αντίστοιχο προφίλ (Student, Company, ή Professor) ανάλογα με τον ρόλο που επιλέχθηκε. 7. Το σύστημα ανακατευθύνει στη σελίδα εισόδου.
Alternative flow 1	Ο Χρήστης υπάρχει ήδη: 5.1. Το σύστημα εμφανίζει μήνυμα "User already registered!". 5.2. Η διαδικασία τερματίζεται και ο χρήστης παραμένει στη φόρμα εγγραφής/εισόδου.
Post conditions	Έχει δημιουργηθεί νέος λογαριασμός στη βάση δεδομένων και η αντίστοιχη οντότητα (π.χ. Student) έχει αρχικοποιηθεί.

Use Case 2: Είσοδος Χρήστη (Login)

Use case ID	UC-02
Actors	Εγγεγραμμένος Χρήστης (Φοιτητής, Καθηγητής, Εταιρεία, Επιτροπή)
Pre conditions	Ο χρήστης πρέπει να έχει εγγραφεί και να μην είναι ήδη συνδεδεμένος.
Main flow of events	1. Ο χρήστης βρίσκεται στην σελίδα σύνδεσης στην αρχική σελίδα. 2. Ο χρήστης συμπληρώνει τα στοιχεία του (Username, Password) και υποβάλει τη φόρμα. 3. Ο χρήστης πατάει το κουμπί Login. 4. Εφόσον τα στοιχεία είναι σωστά, το σύστημα ανακτά τον ρόλο του χρήστη. 6. Το σύστημα συνδέει τον χρήστη και τον ανακατευθύνει στο αντίστοιχο dashboard.
Alternative flow 1	Λανθασμένα Στοιχεία: 4.1. Το σύστημα απορρίπτει την είσοδο. 4.2. Εμφανίζεται μήνυμα λάθους στην οθόνη.
Post conditions	Ο χρήστης είναι αυθεντικοποιημένος (Authenticated) και βρίσκεται στην αρχική σελίδα του ρόλου του (Dashboard).

Use Case 3: Διαχείριση Προφίλ (Manage Profile)

Use case ID	UC-03
Actors	Εγγεγραμμένος Χρήστης (Student, Professor, Company)
Pre conditions	Ο χρήστης πρέπει να είναι συνδεδεμένος (Authenticated).
Main flow of events	1. Ο χρήστης επιλέγει "Προφίλ" (Profile) από το μενού. 2. Το σύστημα ανακτά τα τρέχοντα στοιχεία του χρήστη από τη βάση δεδομένων. 3. Το σύστημα εμφανίζει τη φόρμα επεξεργασίας προφίλ με συμπληρωμένα τα υπάρχοντα δεδομένα. 4. Ο χρήστης τροποποιεί τα πεδία που επιθυμεί. 5. Ο χρήστης πατάει το κουμπί "Αποθήκευση" (Save). 7. Το σύστημα ανακατευθύνει τον χρήστη στο Dashboard του.
Alternative flow 1	Μη έγκυρα δεδομένα: 5.1. Το σύστημα εμφανίζει μήνυμα λάθους. 5.2. Ο χρήστης παραμένει στη φόρμα για διόρθωση.
Post conditions	Τα στοιχεία του χρήστη έχουν ενημερωθεί στη βάση δεδομένων και οι αλλαγές είναι ορατές στην επόμενη προβολή του προφίλ.

Use Case 4: Δημιουργία Θέσης Πρακτικής

Use case ID	UC-04
Actors	Εταιρεία (Company)
Pre conditions	Ο χρήστης πρέπει να είναι συνδεδεμένος με ρόλο **COMPANY**.
Main flow of events	1. Η Εταιρεία επιλέγει "Προσθήκη Νέας Θέσης" από το Dashboard. 2. Το σύστημα εμφανίζει τη φόρμα δημιουργίας θέσης. 3. Η Εταιρεία συμπληρώνει τα στοιχεία (Τίτλος, Περιγραφή, Ημερομηνίες, κλπ). 4. Η Εταιρεία υποβάλει τη φόρμα. 5. Το σύστημα ανακτά το προφίλ της συνδεδεμένης εταιρείας. 6. Το σύστημα συνδέει τη νέα θέση με την εταιρεία και την αποθηκεύει στη βάση δεδομένων. 7. Το σύστημα ανακατευθύνει στο Dashboard της εταιρείας.
Post conditions	Η νέα θέση πρακτικής είναι διαθέσιμη στη λίστα των διαθέσιμων θέσεων της εταιρείας.

Use Case 5: Αντιστοίχιση Φοιτητή σε Θέση

Use case ID	UC-05
Actors	Επιτροπή Πρακτικής (Committee)
Pre conditions	Ο χρήστης είναι συνδεδεμένος ως **COMMITTEE**. Υπάρχουν φοιτητές που αναζητούν πρακτική.
Main flow of events	1. Η Επιτροπή βλέπει τη λίστα αιτήσεων φοιτητών. 2. Η Επιτροπή επιλέγει έναν φοιτητή και μια Στρατηγική Αναζήτησης (π.χ. "Βάσει Ενδιαφερόντων" ή "Βάσει Τοποθεσίας"). 3. Το σύστημα εκτελεί την αναζήτηση μέσω του PositionsSearchFactory και επιστρέφει τις κατάλληλες θέσεις. 4. Η Επιτροπή επιλέγει μία από τις προτεινόμενες θέσεις και πατάει "Ανάθεση". 5. Το σύστημα συνδέει τον φοιτητή με τη θέση. 6. Το σύστημα ενημερώνει το προφίλ του φοιτητή ότι δεν αναζητά πλέον πρακτική.
Alternative flow 1	Στο βήμα 3, αν η στρατηγική δεν επιστρέψει θέσεις, η λίστα είναι κενή και η Επιτροπή πρέπει να επιλέξει άλλη στρατηγική ή να ενημερώσει τον φοιτητή.
Post conditions	Η νέα θέση πρακτικής είναι διαθέσιμη στη λίστα των διαθέσιμων θέσεων της εταιρείας.

Use Case 6: Ανάθεση Επόπτη Καθηγητή

Use case ID	UC-06
Actors	Επιτροπή Πρακτικής (Committee)
Pre conditions	Υπάρχει θέση πρακτικής που έχει ανατεθεί σε φοιτητή αλλά δεν έχει επόπτη.
Main flow of events	1. Η Επιτροπή επιλέγει "Ανάθεση Επόπτη" για μια συγκεκριμένη θέση. 2. Η Επιτροπή επιλέγει τη στρατηγική ανάθεσης (π.χ. "Βάσει Φόρτου" ή "Βάσει Ενδιαφερόντων"). 3. Το σύστημα καλεί το SupervisorAssigmentFactory για να βρει τον κατάλληλο καθηγητή. 4. Η στρατηγική επιλέγει αυτόματα τον καθηγητή (π.χ. αυτόν με τις λιγότερες θέσεις). 5. Το σύστημα αποθηκεύει την ανάθεση και συνδέει τον καθηγητή με τη θέση.
Alternative flow 1	Αδυναμία εύρεσης Επόπτη: Στο βήμα 3, αν δεν βρεθεί κατάλληλος καθηγητής (π.χ. κανείς δεν ταιριάζει στα ενδιαφέροντα): 3.1. Το assignSupervisor πετάει Exception. 3.2. Ο Controller πιάνει το Exception (catch Exception e). 3.3. Το σύστημα εμφανίζει μήνυμα λάθους "No professor matches this strategy.".
Post conditions	Η θέση πρακτικής έχει πλέον επόπτη καθηγητή.

Use Case 7: Συμπλήρωση Ημερολογίου (Logbook)

Use case ID	UC-07
Actors	Φοιτητής (Student)
Pre conditions	Ο φοιτητής έχει αναλάβει θέση πρακτικής.
Main flow of events	1. Ο Φοιτητής πλοηγείται στην καρτέλα "Logbook". 2. Το σύστημα ελέγχει αν υπάρχει ανατεθειμένη θέση. 3. Το σύστημα εμφανίζει τη φόρμα του ημερολογίου. 4. Ο Φοιτητής γράφει το κείμενο και πατάει "Αποθήκευση". 5. Το σύστημα αποθηκεύει το νέο κείμενο στο πεδίο studentLogbook της θέσης. 6. Εμφανίζεται μήνυμα επιτυχίας "Logbook saved".
Post conditions	Το ημερολόγιο της θέσης έχει ενημερωθεί.

Use Case 8: Υποβολή Αξιολόγησης (Εταιρεία/Καθηγητής)

Use case ID	UC-08
Actors	Εταιρεία (Company) ή Καθηγητής (Professor)
Pre conditions	Η θέση πρακτικής είναι σε εξέλιξη.
Main flow of events	1. Ο χρήστης επιλέγει μια θέση από τη λίστα των ανατεθειμένων θέσεων. 2. Επιλέγει "Αξιολόγηση". 3. Το σύστημα εμφανίζει τη φόρμα αξιολόγησης. 4. Ο χρήστης συμπληρώνει βαθμολογίες (motivation, efficiency, etc.) και υποβάλλει τη φόρμα. 5. Το σύστημα ελέγχει αν η θέση είναι ενεργή. 6. Το σύστημα αποθηκεύει την αξιολόγηση και τη συνδέει με τη θέση. 7. Εμφανίζεται μήνυμα επιτυχίας.
Post conditions	Μια νέα εγγραφή **Evaluation** έχει προστεθεί στη λίστα αξιολογήσεων της θέσης.

Use Case 9: Ολοκλήρωση Πρακτικής (Pass/Fail)

Use case ID	UC-09
Actors	Επιτροπή Πρακτικής (Committee)
Pre conditions	Η θέση έχει αξιολογηθεί από τους επόπτες.
Main flow of events	1. Η Επιτροπή επιλέγει "Monitor" για μια θέση. 2. Το σύστημα εμφανίζει τις αξιολογήσεις της Εταιρείας και του Καθηγητή. 3. Η Επιτροπή κρίνει το αποτέλεσμα και πατάει το κουμπί "Pass" ή "Fail". 4. Το σύστημα καλεί την completeTraineeship. 5. Το πεδίο isCompleted γίνεται true. 6. Το πεδίο isAssigned γίνεται false. 7. Ο βαθμός αποθηκεύεται (passFailGrade). 8. Ο φοιτητής αποσυνδέεται από τη θέση.
Post conditions	Η πρακτική θεωρείται ολοκληρωμένη και ο φοιτητής δεν έχει πλέον ενεργή ανάθεση.

TEST CASES

Για κάθε ιστορία χρήστη (User Story) της εφαρμογής έχουν αναπτυχθεί αντίστοιχες περιπτώσεις ελέγχου (Test Cases), οι οποίες καλύπτουν τα επίπεδα του Controller, του Service και του Mapper.

Οι περιπτώσεις ελέγχου βρίσκονται οργανωμένες στις αντίστοιχες κλάσεις ελέγχου (*Test.java) εντός της δομής του έργου και παρέχουν υψηλή κάλυψη του πηγαίου κώδικα.»

Για τη διασφάλιση της ποιότητας του λογισμικού υλοποιήθηκαν αυτοματοποιημένοι έλεγχοι (Unit & Integration Tests) καλύπτοντας τα τρία βασικά επίπεδα της αρχιτεκτονικής της εφαρμογής (Mappers, Services, Controllers). Χρησιμοποιήθηκαν τα εργαλεία **JUnit 5**, **Mockito** και **Spring Boot Test**.

1. Πακέτο: myy803.traineeship_app.mappers (Data Layer Tests)

Στο επίπεδο πρόσβασης δεδομένων, πραγματοποιήθηκαν Integration Tests με χρήση του @DataJpaTest και in-memory βάσης δεδομένων (H2) για την επαλήθευση των custom ερωτημάτων (queries) προς τη βάση.

- **TraineeshipPositionsMapper:** Επαληθεύτηκε η ορθή ανάκτηση θέσεων βάσει κριτηρίων (αναζήτηση με βάση θέμα/topic, φιλτράρισμα μη ανατεθειμένων θέσεων, εύρεση θέσεων ανά επιβλέποντα καθηγητή).

- **CompanyMapper:** Ελέγχθηκε η εύρεση εταιρείας βάσει username και η αναζήτηση εταιρειών με βάση την τοποθεσία (Location Strategy).

- **StudentMapper:** Πιστοποιήθηκε η λειτουργία εύρεσης φοιτητή και η ανάκτηση λίστας φοιτητών που αναζητούν ενεργά πρακτική άσκηση.

- **ProfessorMapper:** Ελέγχθηκε η ορθή ανάκτηση και αποθήκευση των προφίλ των καθηγητών.

2. Πακέτο: myy803.traineeship_app.service (Business Logic Tests)

Στο επίπεδο της επιχειρησιακής λογικής, υλοποιήθηκαν Unit Tests με χρήση του **Mockito** (@ExtendWith(MockitoExtension.class)) για την απομόνωση των εξαρτήσεων και τον έλεγχο των κανόνων της εφαρμογής.

- **CommitteeService:** Ελέγχθηκαν οι διαδικασίες ανάθεσης θέσεων σε φοιτητές, η αυτόματη επιλογή επιβλέποντα (Design Patterns: Strategy & Factory), η προβολή αξιολογήσεων και η τελική βαθμολόγηση (Pass/Fail) της πρακτικής.

- **StudentService:** Επιβεβαιώθηκε η λογική αποθήκευσης του Logbook, η ενημέρωση προφίλ και η διαχείριση του status αναζήτησης πρακτικής ("LookingForTraineeship").

- **CompanyService:** Ελέγχθηκε η δημιουργία και διαγραφή θέσεων πρακτικής, η διασφάλιση ακεραιότητας κατά τη διαγραφή (αποδέσμευση φοιτητή) και η υποβολή αξιολογήσεων από την πλευρά της εταιρείας.

- **ProfessorService:** Επαληθεύτηκε η ανάκτηση εποπτευόμενων θέσεων και η υποβολή αξιολογήσεων, διασφαλίζοντας ότι αξιολογούνται μόνο ανατεθειμένες θέσεις.

3. Πακέτο: myy803.traineeship_app.controllers (Web Layer Tests)

Στο επίπεδο παρουσίασης, πραγματοποιήθηκαν έλεγχοι με χρήση του @WebMvcTest και του **MockMvc** για την προσομοίωση HTTP αιτημάτων και την επαλήθευση της αλληλεπίδρασης χρήστη-συστήματος.

- **CommitteeController:** Ελέγχθηκε η ορθή δρομολόγηση στα views (Dashboard, List Applications), η μεταφορά δεδομένων στο Model (θέσεις, φοιτητές) και η διαχείριση των φορμών ανάθεσης.

- **StudentController:** Πιστοποιήθηκε η εμφάνιση του προφίλ, η διαχείριση σφαλμάτων κατά την πρόσβαση στο Logbook (αν δεν υπάρχει θέση) και η επιτυχής υποβολή φόρμας.

- **CompanyController:** Ελέγχθηκε η ροή δημοσίευσης νέας θέσης, η λίστα διαθέσιμων/ανατεθειμένων θέσεων και η ασφάλεια ρόλων μέσω του @WithMockUser.

- **ProfessorController:** Επιβεβαιώθηκε η εμφάνιση των εποπτευόμενων θέσεων και η λειτουργία της φόρμας αξιολόγησης.