



ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

**Ανάπτυξη Εφαρμογής Ιστού για τη Διαχείριση Μαθημάτων,
Εκπαιδευτών και Σπουδαστών Φροντιστηρίου Μέσης
Εκπαίδευσης**

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Βογιατζή Ευστάθιου

Επιβλέπων: Κυριάκος Κριτικός

Μέλη εξεταστικής επιτροπής: Μαρία Καρύδα, Παναγιώτης Συμεωνίδης

Σάμος, Ιούλιος, 2026

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πρόλογος και ευχαριστίες

Η εκπόνηση της παρούσας διπλωματικής εργασίας αποτέλεσε μια δημιουργική γέφυρα ανάμεσα στη θεωρητική ακαδημαϊκή γνώση και τις σύγχρονες, πραγματικές απαιτήσεις της βιομηχανίας παραγωγής λογισμικού. Η επιθυμία να σχεδιαστεί ένα πλήρως λειτουργικό και ρεαλιστικό σύστημα διαχείρισης, ικανό να υποστηρίξει τις περίπλοκες ανάγκες ενός εκπαιδευτικού οργανισμού, αποτέλεσε το βασικό κίνητρο αυτής της προσπάθειας. Η πρόκληση της υλοποίησης μιας καθαρής και επεκτάσιμης υποδομής, με αυστηρή προσήλωση στα πρότυπα της Εξαγωνικής Αρχιτεκτονικής, δεν ήταν για εμένα απλώς μια ακαδημαϊκή άσκηση, αλλά ένα ουσιαστικό βήμα επαγγελματικής και τεχνικής ωρίμανσης.

Η ολοκλήρωση αυτού του ταξιδιού δεν θα ήταν εφικτή χωρίς την πολύτιμη καθοδήγηση και υποστήριξη των ανθρώπων που στάθηκαν αρωγοί στην προσπάθειά μου. Αρχικά, θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου στον επιβλέποντα καθηγητή μου, κ. Κυριάκο Κριτικό, για την εμπιστοσύνη, τον χρόνο που διέθεσε και τις καίριες υποδείξεις του σε κάθε στάδιο της ανάπτυξης του έργου. Επιπλέον, ευχαριστώ θερμά το Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων του Πανεπιστημίου Αιγαίου για τα πολύτιμα εφόδια και τις ακαδημαϊκές βάσεις που μου παρείχε. Τέλος, το πιο μεγάλο ευχαριστώ το οφείλω στην οικογένειά μου, τους φίλους μου, και φυσικά τη γυναίκα μου, για την αμέριστη συμπαράσταση, την υπομονή και την ενθάρρυνση που μου προσέφεραν καθ' όλη τη διάρκεια των σπουδών μου.

© 2026

του

Βογιατζή Ευστάθιου

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Η σελίδα αυτή είναι σκόπιμα λευκή.

Πίνακας Περιεχομένων

1	Εισαγωγή	1
1.1	Η Ψηφιακή Μετάβαση των Φροντιστηρίων Μέσης Εκπαίδευσης	1
1.2	Αντικείμενο Διπλωματικής	2
1.2.1	Τεχνικές Προκλήσεις και Αρχιτεκτονική Προσέγγιση	2
1.3	Μεθοδολογία Ανάπτυξης Συστήματος	2
1.4	Δομή της Διπλωματικής	3
2	Θεωρητικό Υπόβαθρο	4
2.1	Φροντιστήρια Μέσης Εκπαίδευσης	4
2.1.1	Ο Ρόλος και οι Παρεχόμενες Υπηρεσίες	4
2.1.2	Βασικές Λειτουργίες και Προκλήσεις Εσωτερικής Διαχείρισης	5
2.1.3	Τρέχουσα Υπολογιστική Υποστήριξη	6
2.2	Λογισμικό Υποστήριξης και Διαχείρισης Φροντιστηρίων	6
2.2.1	Οφέλη και Πλεονεκτήματα της Ψηφιοποίησης	6
2.2.2	Καλυπτόμενες Λειτουργίες Εσωτερικής Διαχείρισης	7
2.2.3	Επισκόπηση Υφιστάμενων Λύσεων (Γνωστά Λογισμικά)	8
2.2.4	Κριτήρια Σύγκρισης και Κοινή Αρχιτεκτονική	8
2.2.5	Τρέχοντα Κενά και η Συνεισφορά της Παρούσας Εργασίας	10
3	Ανάπτυξη Συστήματος	12
3.1	Εισαγωγή στη Μεθοδολογία Ανάπτυξης	12
3.2	Ανάλυση Απαιτήσεων	13
3.2.1	Είδη Χρηστών (Οντότητες)	13
3.2.2	Λειτουργικές Απαιτήσεις Διαχείρισης Χρηστών	14
3.2.3	Λειτουργικές Απαιτήσεις Διαχείρισης Μαθημάτων & Προγράμματος	15
3.2.4	Μη Λειτουργικές Απαιτήσεις	17
3.3	Αρχιτεκτονικός και Εννοιολογικός Σχεδιασμός	19
3.3.1	Διάγραμμα Περιβάλλοντος (Context Diagram)	19
3.3.2	Αρχιτεκτονική Συστατικών (Component Diagram – Hexagonal Architecture)	20
3.3.3	Διαγράμματα Περιπτώσεων Χρήσης (Use-Case Diagrams)	22
3.3.4	Διάγραμμα Δραστηριοτήτων / Ακολουθίας (Activity / Sequence Diagrams)	27
3.3.5	Διάγραμμα Κλάσεων / Βάσης Δεδομένων (Class / Database Diagram)	38
3.4	Υλοποίηση	41
3.4.1	Τεχνολογίες και Γλώσσες Προγραμματισμού	41

3.4.2	Πλαίσια Ανάπτυξης και Βιβλιοθήκες.....	42
3.4.3	Δομή Πηγαίου Κώδικα (Packages, DTOs, Mappers)	43
3.4.4	Ανάλυση Βασικών Κλάσεων.....	45
3.5	Βαθμός Ικανοποίησης Απαιτήσεων.....	47
3.5.1	Πίνακας Αξιολόγησης και Αιτιολόγηση Αποκλίσεων.....	48
4	Εγκατάσταση και Επίδειξη Συστήματος	51
4.1	Εγκατάσταση Συστήματος.....	51
4.1.1	Προαπαιτούμενα Εγκατάστασης.....	51
4.1.2	Οδηγίες Εγκατάστασης και Εκτέλεσης.....	52
4.2	Επίδειξη Συστήματος (Σενάρια Χρήσης & Στιγμιότυπα Οθόνης).....	53
4.2.1	Σενάριο 1: Διαχείριση Λογαριασμών και Αυθεντικοποίηση	53
4.2.2	Σενάριο 2: Δημιουργία και Προγραμματισμός Μαθημάτων (Admin)	56
4.2.3	Σενάριο 3: Προβολή Προγράμματος (Μαθητής / Καθηγητής).....	58
5	Συμπεράσματα και Μελλοντική Εργασία	63
5.1	Συμπεράσματα	63
5.2	Μελλοντική Εργασία	64
Βιβλιογραφία.....		66
Παράρτημα I: Επιλεγμένος Πηγαίος Κώδικας Συστήματος (Java / Spring Boot)		70

Λίστα Σχημάτων

Σχήμα 1: Διάγραμμα Περιβάλλοντος (Context Diagram) του Πληροφοριακού Συστήματος	20
Σχήμα 2: Διάγραμμα Συστατικών & Ροή Μετασχηματισμού Δεδομένων (Hexagonal Architecture)	22
Σχήμα 3: Διάγραμμα Περιπτώσεων Χρήσης - Αυθεντικοποίηση και Διαχείριση Λογαριασμών	23
Σχήμα 4: Διάγραμμα Περιπτώσεων Χρήσης - Λειτουργίες Διαχειριστή	24
Σχήμα 5: Διάγραμμα Περιπτώσεων Χρήσης - Λειτουργίες Εκπαιδευτικού	25
Σχήμα 6: Διάγραμμα Περιπτώσεων Χρήσης - Λειτουργίες Σπουδαστή	26
Σχήμα 7: Διάγραμμα Ακολουθίας - Αυθεντικοποίηση Χρήστη (Login)	28
Σχήμα 8: Διάγραμμα Δραστηριοτήτων - Διαχείριση Αιτημάτων Εγγραφής	29
Σχήμα 9: Διάγραμμα Ακολουθίας - Κεντρικός Έλεγχος Απουσιών (Master Override)	30
Σχήμα 10: Διάγραμμα Ακολουθίας - Δημιουργία και Ανάθεση Εξέτασης (Create Test)	31
Σχήμα 11: Διάγραμμα Ακολουθίας - Βαθμολόγηση Σπουδαστών (Grade Student)	32
Σχήμα 12: Διάγραμμα Δραστηριοτήτων - Διαχείριση Απουσιών (Teacher Absences)	34
Σχήμα 13: Διάγραμμα Ακολουθίας - Αίτημα Εγγραφής σε Τμήμα (Enroll Request)	35
Σχήμα 14: Διάγραμμα Ακολουθίας - Προβολή Προσωπικού Προγράμματος (Weekly Schedule)	36
Σχήμα 15: Διάγραμμα Δραστηριοτήτων - Αίτημα Αποχώρησης (Request Drop)	37
Σχήμα 16: Διάγραμμα Κλάσεων (Domain Model Class Diagram)	39
Σχήμα 17: Διάγραμμα Βάσης (Database Diagram)	40
Σχήμα 18: Συμπληρωμένη δυναμική φόρμα εγγραφής νέου εκπαιδευτικού με δήλωση διδακτικής ικανότητας	54
Σχήμα 19: Ενημερωτικό μήνυμα συστήματος σχετικά με την ανενεργή κατάσταση του νέου λογαριασμού	55
Σχήμα 20: Περιβάλλον Διαχειριστή - Καρτέλα λεπτομερειών και ενεργοποίηση εκκρεμούς λογαριασμού χρήστη	55
Σχήμα 21: Επιτυχής αυθεντικοποίηση και ανακατεύθυνση στο αρχικό ταμπλό (Dashboard) του Εκπαιδευτικού	56
Σχήμα 22: Κεντρική φόρμα αυθεντικοποίησης (Login) για την είσοδο του Διαχειριστή	56
Σχήμα 23: Ο κεντρικός πίνακας ελέγχου (Dashboard) για τη διαχείριση των μαθημάτων του φροντιστηρίου	57
Σχήμα 24: Φόρμα εισαγωγής μεταδεδομένων για τη δημιουργία νέου μαθήματος	57
Σχήμα 25: Δυναμική λίστα επιλογής εκπαιδευτικού, φιλτραρισμένη βάσει των προτιμήσεων διδασκαλίας	58
Σχήμα 26: Επιτυχής ανάθεση (assignment) του μαθήματος στον επιλεγμένο εκπαιδευτικό	58
Σχήμα 27: Το εβδομαδιαίο ημερολόγιο του εκπαιδευτικού, εμφανίζοντας ενεργά και ακυρωμένα (λόγω απουσίας) μαθήματα	59

Σχήμα 28: Αναλυτική προβολή και διαχείριση των δραστηριοτήτων μαθήματος (Lesson Log & Planner) από τον εκπαιδευτικό	59
Σχήμα 29: Δυναμικό κλείδωμα της φόρμας δραστηριοτήτων με σχετικό προειδοποιητικό μήνυμα, λόγω δηλωμένης απουσίας	60
Σχήμα 30: Προβολή του ημερολογίου από την πλευρά του σπουδαστή, σε πλήρη συγχρονισμό με τα δεδομένα του εκπαιδευτικού	61
Σχήμα 31: Προβολή των δραστηριοτήτων και των κατ' οίκον εργασιών (Homework) από τον σπουδαστή	61
Σχήμα 32: Ενημέρωση του σπουδαστή για την αιτία ακύρωσης της προγραμματισμένης παράδοσης.....	62

Λίστα Πινάκων

Πίνακας 1: Συγκριτική αξιολόγηση εκπαιδευτικών λογισμικών.....	9
Πίνακας 2: Απόσπασμα κώδικα της μεθόδου εγγραφής (StudentController).....	45
Πίνακας 3: Απόσπασμα κώδικα ρυθμίσεων ασφαλείας (SecurityConfig).....	46
Πίνακας 4: Συγκεντρωτική Αξιολόγηση Ικανοποίησης Απαιτήσεων Συστήματος	48

Ακρωνύμια

Ακρωνύμιο	Αγγλικός Όρος	Ελληνική Απόδοση
API	Application Programming Interface	Διεπαφή Προγραμματισμού Εφαρμογών
CRM	Customer Relationship Management	Διαχείριση Πελατειακών Σχέσεων
CRUD	Create, Read, Update, Delete	Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή (Βασικές λειτουργίες Βάσης Δεδομένων)
CSRF	Cross-Site Request Forgery	Πλαστογράφηση Αιτήματος Διασταυρούμενου Ιστότοπου (Επίθεση στον Παγκόσμιο Ιστό)
CSS	Cascading Style Sheets	Διαδοχικά Φύλλα Στυλ
DI	Dependency Injection	Εισαγωγή Εξαρτήσεων / Έγχυση Εξαρτήσεων
DTO	Data Transfer Object	Αντικείμενο Μεταφοράς Δεδομένων
ERP	Enterprise Resource Planning	Σύστημα Ενδοεπιχειρησιακού Σχεδιασμού / Σχεδιασμός Πόρων Επιχείρησης
HCI	Human-Computer Interaction	Αλληλεπίδραση Ανθρώπου-Υπολογιστή
HTML	HyperText Markup Language	Γλώσσα Σήμανσης Υπερκειμένου
HTTP	HyperText Transfer Protocol	Πρωτόκολλο Μεταφοράς Υπερκειμένου
IoC	Inversion of Control	Αντιστροφή Ελέγχου
ISA	“Is-A” Relationship	Σχέση Γενίκευσης / Σχέση «Είναι-Ένα»
ISO/IEC	International Organization for Standardization / International Electrotechnical Commission	Διεθνής Οργανισμός Τυποποίησης / Διεθνής Ηλεκτροτεχνική Επιτροπή
JPA	Java Persistence API	Διεπαφή Προγραμματισμού Εφαρμογών Διατήρησης (Java)
JS	JavaScript	-
JSON	JavaScript Object Notation	Μορφότυπος Ανταλλαγής Δεδομένων JavaScript
JWT	JSON Web Token	Διακριτικό Ιστού JSON (Διακριτικό Ασφαλείας)
LMS	Learning Management System	Σύστημα Διαχείρισης Μάθησης
MFA	Multi-Factor Authentication	Πολυπαραγοντική Αυθεντικοποίηση
MIS	Management Information System	Πληροφοριακό Σύστημα Διοίκησης
MVC	Model-View-Controller	Μοντέλο-Προβολή-Ελεγκτής (Σχεδιαστικό Πρότυπο)
OOP	Object-Oriented Programming	Αντικειμενοστρεφής Προγραμματισμός
ORM	Object-Relational Mapping	Αντικείμενο-Σχεσιακή Αντιστοίχιση
OTP	One-Time Password	Κωδικός Μίας Χρήσης

POM	Project Object Model	Μοντέλο Αντικειμένου Έργου (Σύστημα Maven)
RBAC	Role-Based Access Control	Έλεγχος Πρόσβασης Βάσει Ρόλων
REST	Representational State Transfer	Μεταφορά Αντιπροσωπευτικής Κατάστασης (Πρότυπο Αρχιτεκτονικής Ιστού για τη Μεταφορά Δεδομένων μεταξύ Εφαρμογών)
SDLC	Software Development Life Cycle	Κύκλος Ζωής Ανάπτυξης Λογισμικού
SIS	Student Information System	Πληροφοριακό Σύστημα Σπουδαστών
SMTP	Simple Mail Transfer Protocol	Απλό Πρωτόκολλο Μεταφοράς Αλληλογραφίας
SQL	Structured Query Language	Δομημένη Γλώσσα Ερωτημάτων
SSR	Server-Side Rendering	Απόδοση από την Πλευρά του Διακομιστή
UI	User Interface	Διεπαφή Χρήστη
UML	Unified Modeling Language	Ενοποιημένη Γλώσσα Μοντελοποίησης
URL	Uniform Resource Locator	Ενιαίος Εντοπιστής Πόρων
UX	User Experience	Εμπειρία Χρήστη
XML	eXtensible Markup Language	Επεκτάσιμη Γλώσσα Σήμανσης
XSS	Cross-Site Scripting	Διασταυρούμενη Δέσμη Εντολών (Επίθεση στον Παγκόσμιο Ιστό)
YAML	YAML Ain't Markup Language	Η YAML δεν είναι Γλώσσα Σήμανσης (Γλώσσα Σειριοποίησης Δεδομένων)

Περίληψη

Η παρούσα διπλωματική εργασία πραγματεύεται τον σχεδιασμό, την ανάλυση και την ανάπτυξη μιας ολοκληρωμένης Εφαρμογής Ιστού (Web Application) για τη διαχείριση Φροντιστηρίων Μέσης Εκπαίδευσης. Σκοπός του συστήματος είναι η επίλυση των προβλημάτων οργάνωσης και κατακερματισμού της πληροφορίας που αντιμετωπίζουν οι σύγχρονοι εκπαιδευτικοί οργανισμοί, παρέχοντας μια κεντριοποιημένη ψηφιακή πλατφόρμα που διευκολύνει το καθημερινό διοικητικό και εκπαιδευτικό έργο.

Για την υλοποίηση του έργου ακολουθήθηκε ο Κύκλος Ζωής Ανάπτυξης Λογισμικού βάσει του μοντέλου / μεθοδολογίας του Καταρράκτη (Waterfall), περιλαμβάνοντας εκτενή ανάλυση απαιτήσεων και μοντελοποίηση μέσω διαγραμμάτων UML. Σε τεχνικό επίπεδο, η εφαρμογή αναπτύχθηκε στο οικοσύστημα της Java με χρήση του πλαισίου Spring Boot. Ιδιαίτερη έμφαση δόθηκε στη συγγραφή διατηρήσιμου κώδικα (Clean Code) υιοθετώντας το αυστηρό πρότυπο της Εξαγωνικής Αρχιτεκτονικής (Hexagonal Architecture / Ports and Adapters). Η αρχιτεκτονική αυτή εγγυάται τον απόλυτο διαχωρισμό της επιχειρησιακής λογικής από τους μηχανισμούς αποθήκευσης των δεδομένων, διατηρώντας, για παράδειγμα, τις οντότητες του συστήματος σε ένα πλήρως ανεξάρτητο επίπεδο αποθετηρίου (repository module). Η διεπαφή χρήστη κατασκευάστηκε με τεχνολογίες HTML5, CSS3, JavaScript και Thymeleaf, ενώ η διαχείριση της σχεσιακής βάσης δεδομένων υλοποιήθηκε μέσω MySQL. Παράλληλα, το σύστημα θωρακίστηκε με σύγχρονους μηχανισμούς κυβερνοασφάλειας, αξιοποιώντας διακριτικά JSON Web Tokens (JWT) για την αυθεντικοποίηση χωρίς κατάσταση (stateless authentication).

Το τελικό προϊόν ενσωματώνει αυστηρό Έλεγχο Πρόσβασης Βάσει Ρόλων (Role-Based Access Control), προσφέροντας εξατομικευμένα περιβάλλοντα (dashboards) για Διαχειριστές, Εκπαιδευτικούς και Σπουδαστές. Μέσω της εφαρμογής υποστηρίζεται, μεταξύ άλλων, η πλήρης διαχείριση γνωστικών αντικειμένων, η ανάθεση μαθημάτων βάσει ικανότητας διδασκαλίας, η καταχώρηση διδαχθείσας ύλης, καθώς και η δυναμική αποτροπή διενέξεων (conflicts) στο ωρολόγιο πρόγραμμα. Εν κατακλείδι, το σύστημα βελτιώνει δραματικά τον συντονισμό των εμπλεκόμενων μερών και προάγει την ψηφιακή μετάβαση των εκπαιδευτηρίων, αποτελώντας ένα ασφαλές και επεκτάσιμο λογισμικό.

Λέξεις Κλειδιά: Πληροφοριακό Σύστημα Φροντιστηρίου, Εφαρμογή Ιστού, Εξαγωνική Αρχιτεκτονική, Μεθοδολογία Καταρράκτη, Spring Boot, Διαχείριση Εκπαίδευσης.

Abstract

This thesis addresses the design, analysis, and development of a comprehensive Web Application for the management of Secondary Education Tutoring Centers. The primary objective of the system is to resolve the organizational and information fragmentation problems faced by modern educational institutions, providing a centralized digital platform that facilitates daily administrative and educational operations.

The project was implemented according to the Software Development Life Cycle by following the Waterfall methodology, encompassing extensive requirement analysis and modeling via UML diagrams. On a technical level, the application was developed within the Java ecosystem utilizing the Spring Boot framework. Special emphasis was placed on writing maintainable code (Clean Code) by adopting the strict pattern of Hexagonal Architecture (Ports and Adapters). This architecture guarantees the absolute separation of business logic from the data storage mechanisms, maintaining, for instance, the system's entities in a fully independent repository module. The user interface was built using HTML5, CSS3, JavaScript, and Thymeleaf technologies, while the relational database management was implemented via MySQL. Simultaneously, the system was fortified with modern cybersecurity mechanisms, utilizing JSON Web Tokens (JWT) for stateless authentication.

The final product incorporates strict Role-Based Access Control (RBAC), offering personalized dashboards for Administrators, Teachers, and Students. The application supports, among other features, full curriculum management, course assignments based on teaching proficiency, logging of taught material, as well as the dynamic prevention of time conflicts in the weekly schedule. To conclude, our system dramatically improves the coordination of the involved parties and promotes the digital transition of educational centers, constituting a secure, reliable, and scalable software solution.

Keywords: *Tutoring Management System, Web Application, Hexagonal Architecture, Waterfall Methodology, Spring Boot, Education Management.*

1

Εισαγωγή

1.1 Η Ψηφιακή Μετάβαση των Φροντιστηρίων Μέσης Εκπαίδευσης

Η ιδιωτική συμπληρωματική εκπαίδευση, και ειδικότερα ο θεσμός των Φροντιστηρίων Μέσης Εκπαίδευσης, αποτελεί έναν από τους πιο νευραλγικούς τομείς του εκπαιδευτικού συστήματος. Σε ένα άκρως ανταγωνιστικό περιβάλλον, τα φροντιστήρια καλούνται να διαχειριστούν καθημερινά έναν τεράστιο όγκο δεδομένων που αφορούν μαθητές, εκπαιδευτικούς, γνωστικά αντικείμενα, βαθμολογίες και ωρολόγια προγράμματα. Η ομαλή διεξαγωγή των μαθημάτων εξαρτάται άμεσα από τον άρτιο συντονισμό αυτών των πόρων, γεγονός που καθιστά την εσωτερική διοικητική λειτουργία τους εξαιρετικά περίπλοκη.

Παρά τη ραγδαία εξέλιξη της τεχνολογίας, ένα σημαντικό ποσοστό φροντιστηριακών οργανισμών εξακολουθεί να βασίζεται σε παραδοσιακές, κατακερματισμένες μεθόδους οργάνωσης. Η χρήση χειρόγραφων απουσιολογίων, μεμονωμένων λογιστικών φύλλων (spreadsheets) και μη διασυνδεδεμένων εφαρμογών επικοινωνίας δημιουργεί ένα περιβάλλον επιρρεπές σε ανθρώπινα λάθη. Τα συστήματα αυτά στερούνται κεντρικής διαχείρισης, δυσκολεύοντας την άμεση άντληση πληροφοριών και τη σφαιρική αξιολόγηση της εκπαιδευτικής διαδικασίας.

Ως αποτέλεσμα, η διοίκηση ενός φροντιστηρίου έρχεται αντιμέτωπη με καθημερινά προβλήματα, όπως η ύπαρξη διενέξεων στο ωρολόγιο πρόγραμμα (time conflicts), η δυσκολία παρακολούθησης των υποχρεώσεων από τους μαθητές και η ελλιπής εποπτεία της διδακτικής ύλης. Ταυτόχρονα, τίθενται σοβαρά ζητήματα ασφάλειας και ιδιωτικότητας, καθώς η απουσία ψηφιακών ρόλων και δικαιωμάτων πρόσβασης καθιστά ευάλωτα τα ευαίσθητα προσωπικά δεδομένα μαθητών και εκπαιδευτικών. Αναδεικνύεται, συνεπώς, η επιτακτική ανάγκη για μετάβαση σε ολοκληρωμένες, ψηφιακές λύσεις που να αυτοματοποιούν τις διαδικασίες, να διασφαλίζουν την ακεραιότητα των πληροφοριών και να βελτιώνουν την εμπειρία όλων των εμπλεκόμενων μερών.

1.2 Αντικείμενο Διπλωματικής

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι ο σχεδιασμός, η ανάλυση απαιτήσεων και η πλήρης ανάπτυξη ενός σύγχρονου Πληροφοριακού Συστήματος Διαχείρισης Φροντιστηρίου (Tutoring Management System). Η εργασία στοχεύει στην επίλυση των προβλημάτων κατακερματισμού της πληροφορίας και οργανωτικής δυσλειτουργίας, προσφέροντας μια ενιαία, κεντριοποιημένη διαδικτυακή εφαρμογή / πλατφόρμα (Web Application) που εξυπηρετεί ταυτόχρονα τις ανάγκες της Διοίκησης, των Εκπαιδευτικών και των Σπουδαστών.

Για την επίλυση των προαναφερθέντων προβλημάτων, η διπλωματική κατασκευάζει ένα σύστημα που εφαρμόζει αυστηρό Έλεγχο Πρόσβασης Βάσει Ρόλων (Role-Based Access Control). Μέσω διακριτών πινάκων ελέγχου (dashboards), ο Διαχειριστής μπορεί να οργανώνει το πρόγραμμα εξαλείφοντας τις διενέξεις, ο Εκπαιδευτικός να καταχωρεί ψηφιακά τη διδαχθείσα ύλη και τις απουσίες, και ο Μαθητής να παρακολουθεί σε πραγματικό χρόνο τις ακαδημαϊκές του υποχρεώσεις. Με αυτόν τον τρόπο, καταργείται η ανάγκη για πολλαπλά ανεξάρτητα προγράμματα / εφαρμογές και διασφαλίζεται η “Μία και Μοναδική Πηγή Αλήθειας” (Single Source of Truth) για όλο τον οργανισμό.

1.2.1 Τεχνικές Προκλήσεις και Αρχιτεκτονική Προσέγγιση

Από τεχνικής σκοπιάς, η εργασία καλείται να αντιμετωπίσει την πρόκληση της δημιουργίας ενός εταιρικού επιπέδου (enterprise-grade) λογισμικού που να είναι επεκτάσιμο και συντηρήσιμο. Για τον σκοπό αυτό, δεν υιοθετείται μια συμβατική μονολιθική προσέγγιση, αλλά εφαρμόζονται αυστηρά πρότυπα Μηχανικής Λογισμικού. Η επιχειρησιακή λογική υλοποιείται με βάση το πρότυπο της Εξαγωνικής Αρχιτεκτονικής (Hexagonal Architecture / Ports and Adapters). Η συγκεκριμένη προσέγγιση επιλύει το πρόβλημα της στενής σύζευξης (tight coupling), διασφαλίζοντας, για παράδειγμα, ότι οι οντότητες του τομέα (domain entities) και η διαχείριση της βάσης δεδομένων παραμένουν πλήρως απομονωμένες στο δικό τους επίπεδο (repository module), μακριά από τον πυρήνα της εφαρμογής (application core). Ταυτόχρονα, τα ζητήματα κυβερνοασφάλειας αντιμετωπίζονται με την εφαρμογή σύγχρονων μηχανισμών παραγωγής και επικύρωσης διακριτικών ασφαλείας (JSON Web Tokens), εξασφαλίζοντας την προστασία των δεδομένων του φροντιστηρίου, καθώς και του Ελέγχου Πρόσβασης Βάσει Ρόλων (RBAC). Παράλληλα, η αρχιτεκτονική της εφαρμογής ενσωματώνει μηχανισμούς θωράκισης (όπως η χρήση HttpOnly cookies) για την αποτελεσματική αντιμετώπιση κοινών διαδικτυακών απειλών, όπως επιθέσεις Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF) και έκχυσης SQL (SQL injection).

1.3 Μεθοδολογία Ανάπτυξης Συστήματος

Για τη συστηματική και οργανωμένη κατασκευή του συστήματος, ακολουθήθηκε ο Κύκλος Ζωής Ανάπτυξης Λογισμικού (SDLC) υπό το πρίσμα της μεθοδολογίας Καταρράκτη (Waterfall) (Pressman, 2005). Η προσέγγιση αυτή επέτρεψε τη γραμμική και εις βάθος ανάλυση των απαιτήσεων του συστήματος, τη σχεδίαση όλων των απαραίτητων διαγραμμάτων UML πριν τη συγγραφή του κώδικα, και την οργανωμένη παραγωγή του λογισμικού, εξασφαλίζοντας ότι το τελικό σύστημα ανταποκρίνεται πλήρως στις αρχικές προδιαγραφές (που τέθηκαν για αυτό).

1.4 Δομή της Διπλωματικής

Η παρούσα διπλωματική εργασία αναπτύσσεται σε πέντε διακριτά κεφάλαια. Το Κεφάλαιο 2 παρουσιάζει το θεωρητικό υπόβαθρο του χώρου των φροντιστηρίων, τα οφέλη της ψηφιοποίησης και την επισκόπηση υφιστάμενων λογισμικών υποστήριξης. Στο Κεφάλαιο 3 παρατίθεται αναλυτικά η διαδικασία ανάπτυξης του συστήματος μέσω της μεθοδολογίας Καταρράκτη, περιλαμβάνοντας την ανάλυση απαιτήσεων, τη μοντελοποίηση (UML), τον αρχιτεκτονικό σχεδιασμό και την υλοποίηση. Το Κεφάλαιο 4 επικεντρώνεται στην εγκατάσταση του συστήματος και την επίδειξη της λειτουργικότητάς του μέσα από ολοκληρωμένα σενάρια χρήσης. Τέλος, στο Κεφάλαιο 5 συνοψίζονται τα συμπεράσματα και η αποκομισθείσα εμπειρία, ενώ καταγράφονται και προτάσεις για τη μελλοντική επέκταση του συστήματος.

2

Θεωρητικό Υπόβαθρο

Στο παρόν κεφάλαιο αναλύεται το ευρύτερο περιβάλλον μέσα στο οποίο εντάσσεται η προτεινόμενη εφαρμογή. Αρχικά, εξετάζεται η φύση και η λειτουργία των Φροντιστηρίων Μέσης Εκπαίδευσης στην Ελλάδα, ενώ στη συνέχεια γίνεται μια επισκόπηση των υφιστάμενων τεχνολογικών λύσεων που χρησιμοποιούνται για την εσωτερική τους οργάνωση.

2.1 Φροντιστήρια Μέσης Εκπαίδευσης

Η ιδιωτική, συμπληρωματική εκπαίδευση – ευρέως γνωστή στη διεθνή βιβλιογραφία ως «σκιάδης εκπαίδευση» (shadow education) – αποτελεί έναν από τους κρισιμότερους πυλώνες του ελληνικού εκπαιδευτικού συστήματος. Τα Φροντιστήρια Μέσης Εκπαίδευσης λειτουργούν παράλληλα με το επίσημο σχολικό δίκτυο, αναλαμβάνοντας ένα βαρύ φορτίο τόσο στην προετοιμασία των μαθητών όσο και στη διαμόρφωση της ακαδημαϊκής τους πορείας.

2.1.1 Ο Ρόλος και οι Παρεχόμενες Υπηρεσίες

Στο σύγχρονο, ταχέως μεταβαλλόμενο κοινωνικό τοπίο, ο ρόλος των Φροντιστηρίων Μέσης Εκπαίδευσης υπερβαίνει τη μηχανιστική μετάδοση εγκυκλοπαιδικών γνώσεων. Οι εκπαιδευτικοί που στελεχώνουν αυτούς τους οργανισμούς καλούνται να λειτουργήσουν σε ένα περιβάλλον όπου η απλή απομνημόνευση έχει δώσει τη θέση της στην κριτική διήθηση της πληροφορίας. Όπως χαρακτηριστικά αναφέρουν οι Μπορουτζή κ.ά. (2016), η εκπαιδευτική διαδικασία στον εκπαιδευτικό χώρο απέχει σημαντικά από μία «στείρα τραπεζική αντίληψη» της γνώσης. Αντιθέτως, αναπτύσσεται μία ουσιαστική παιδαγωγική σχέση, μέσα από την οποία ο καθηγητής αναλαμβάνει ρόλο εμπνευστή και συνεργάτη, εστιάζοντας στην εξατομικευμένη διδασκαλία και την καλλιέργεια της κριτικής σκέψης.

Είναι ιδιαίτερα σημαντικό να τονιστεί ότι η φύση της οικονομικής συναλλαγής που χαρακτηρίζει την ιδιωτική εκπαίδευση δεν ακυρώνει την παιδαγωγική και ηθική αποστολή των λειτουργιών της. Οι καθηγητές των φροντιστηρίων, πέρα από τον προφανή στόχο της ενίσχυσης της εμπέδωσης της σχολικής ύλης και της προετοιμασίας για τις απαιτητικές πανελλαδικές εξετάσεις, λειτουργούν ως αγωγοί μετάδοσης αξιών, προσαρμόζοντας σύγχρονες συνεργατικές και βιωματικές μεθόδους στις ανάγκες των μαθητών τους (Μπορουτζή κ.ά., 2016).

Αυτός ο ενισχυμένος παιδαγωγικός ρόλος, ωστόσο, δεν παραμένει σε θεωρητικό επίπεδο, αλλά μετουσιώνεται πρακτικά μέσα από ένα συγκεκριμένο πλέγμα παρεχόμενων υπηρεσιών. Εξετάζοντας αυτές τις υπηρεσίες, διαπιστώνεται πως ο κύριος λόγος που οι μαθητές στρέφονται μαζικά στη σκιάδη εκπαίδευση είναι η ανάγκη κάλυψης των μαθησιακών κενών που συχνά αφήνει το δημόσιο σχολείο, καθώς και η εντατική προετοιμασία για την εισαγωγή τους στην τριτοβάθμια εκπαίδευσης (Kassotakis & Verdis, 2013). Για να ανταποκριθούν σε αυτές τις αυξημένες απαιτήσεις, τα οργανωμένα φροντιστήρια έχουν αναπτύξει ένα εξαιρετικά δομημένο εκπαιδευτικό πλαίσιο. Σε αντίθεση με τις πολυπληθείς σχολικές τάξεις, επενδύουν στη δημιουργία ολιγομελών και ομοιογενών τμημάτων, γεγονός που επιτρέπει την άμεση αλληλεπίδραση και την καλύτερη παρακολούθηση της προόδου του κάθε μαθητή (Bray, 2024).

Επιπρόσθετα, η υποστήριξη αυτή δεν περιορίζεται μόνο στην προφορική παράδοση του μαθήματος. Τα φροντιστήρια παρέχουν εξειδικευμένες, στοχευμένες σημειώσεις που οργανώνουν την ύλη με τρόπο πιο πρακτικό από τα συμβατικά σχολικά εγχειρίδια. Παράλληλα, καθοριστικό ρόλο στην προετοιμασία παίζει η διεξαγωγή συχνών διαγωνισμάτων. Η διαδικασία αυτή εξοικειώνει τους υποψήφιους με τις αυστηρές συνθήκες και τους χρόνους των πραγματικών εξετάσεων, βοηθώντας τους να διαχειριστούν το άγχος της τελικής αξιολόγησης.

2.1.2 Βασικές Λειτουργίες και Προκλήσεις Εσωτερικής Διαχείρισης

Παρά την έκδηλη εκπαιδευτική τους αποστολή, τα Φροντιστήρια Μέσης Εκπαίδευσης αποτελούν ταυτόχρονα και επιχειρησιακούς οργανισμούς. Η εύρυθμη λειτουργία τους δεν εξαρτάται αποκλειστικά από την ποιότητα της διδασκαλίας, αλλά και από την αποτελεσματική διαχείριση ενός σύνθετου πλέγματος διοικητικών διαδικασιών. Οι βασικές λειτουργίες εσωτερικής διαχείρισης περιλαμβάνουν την εγγραφή και την τήρηση μητρώου σπουδαστών, την ανάθεση των διδασκόντων στα αντίστοιχα μαθήματα, τον έλεγχο των παρουσιών των μαθητών (απουσιολόγιο), καθώς και τη συνεχή καταγραφή των ακαδημαϊκών επιδόσεων (βαθμολογίες) για την άμεση ενημέρωση των κηδεμόνων.

Ωστόσο, η υλοποίηση αυτών των καθημερινών εργασιών συνοδεύεται από σημαντικές προκλήσεις. Η πιο κρίσιμη, ίσως, διοικητική δυσκολία εντοπίζεται στον προγραμματισμό και τον συντονισμό των μαθημάτων (Subbey, 2023). Οι διαχειριστές των φροντιστηρίων καλούνται να δημιουργήσουν εβδομαδιαία προγράμματα που εξισορροπούν τη διαθεσιμότητα των καθηγητών, τη χωρητικότητα των αιθουσών και τον εξαιρετικά περιορισμένο ελεύθερο χρόνο των μαθητών, αποφεύγοντας παράλληλα τις αλληλεπικαλύψεις. Όταν οι διαδικασίες αυτές εκτελούνται χειροκίνητα, ο κίνδυνος οργανωτικών σφαλμάτων αυξάνεται κατακόρυφα.

Το αποτέλεσμα αυτού του αυξημένου διοικητικού φόρτου είναι ο αποπροσανατολισμός του ανθρώπινου δυναμικού. Όπως επισημαίνεται χαρακτηριστικά στη βιβλιογραφία, όταν το προσωπικό και οι εκπαιδευτικοί ενός οργανισμού αναλώνουν δυσανάλογα μεγάλο μέρος του χρόνου τους στη διεκπεραίωση χειρόγραφων ή κατακερματισμένων γραφειοκρατικών εργασιών, η συνολική ποιότητα των παρεχόμενων υπηρεσιών πλήττεται. Αντιθέτως, η μετάβαση σε αυτοματοποιημένα Πληροφοριακά Συστήματα Διοίκησης

(Management Information Systems – MIS) μειώνει δραστικά τον φόρτο εργασίας και τον χρόνο αναζήτησης δεδομένων, επιτρέποντας στο εκπαιδευτικό προσωπικό να επικεντρωθεί απρόσκοπτα στο κύριο, παιδαγωγικό του έργο (Shah, 2014). Καθίσταται, επομένως, επιτακτική η ανάγκη μετάβασης από τις παραδοσιακές μεθόδους οργάνωσης στην υιοθέτηση σύγχρονων συστημάτων υπολογιστικής υποστήριξης.

2.1.3 Τρέχουσα Υπολογιστική Υποστήριξη

Τα σημερινά Φροντιστήρια Μέσης Εκπαίδευσης, στην προσπάθειά τους να ανταποκριθούν στον αυξημένο διοικητικό φόρτο που περιγράφηκε παραπάνω, καταφεύγουν σε διάφορες λύσεις υπολογιστικής υποστήριξης. Η πιο συνηθισμένη προσέγγιση, ειδικά σε μικρομεσαίους οργανισμούς, είναι η χρήση γενικών λογισμικών γραφείου, όπως τα λογιστικά φύλλα (π.χ. MS Excel), ως πρωτεύοντα συστήματα αρχειοθέτησης. Αν και τα εργαλεία αυτά είναι προσιτά και εύχρηστα, η απουσία μιας κεντρικής, σχεσιακής βάσης δεδομένων αποτελεί μια ευάλωτη πρακτική που οδηγεί συχνά σε ασυνέπειες, απώλεια πληροφοριών και κενά ασφαλείας (Yahaya & Abd Warif, 2024).

Από την άλλη πλευρά, μεγαλύτερα εκπαιδευτικά κέντρα συχνά υιοθετούν έτοιμα, εμπορικά συστήματα Διαχείρισης Πελατειακών Σχέσεων (CRM) ή προγράμματα οργάνωσης πόρων (ERP) (Bamufleh κ.ά., 2021). Ωστόσο, όπως επισημαίνεται στη διεθνή βιβλιογραφία, τα γενικά συστήματα CRM είναι αρχιτεκτονικά σχεδιασμένα για επιχειρήσεις με επίκεντρο τις πωλήσεις και τα οικονομικά μεγέθη (Seeman & O'Hara, 2006). Συνεπώς, στερούνται εξειδικευμένων εκπαιδευτικών λειτουργιών, όπως η δυναμική διαχείριση του ωρολογίου προγράμματος, η καταγραφή απουσιών ανά διδακτική ώρα και η παρακολούθηση της ακαδημαϊκής εξέλιξης των σπουδαστών.

Η αδυναμία των παραπάνω προσεγγίσεων να καλύψουν τις ιδιαιτερότητες της ιδιωτικής εκπαίδευσης αναδεικνύει την ανάγκη μετάβασης σε εξειδικευμένα διαδικτυακά Πληροφοριακά Συστήματα Διαχείρισης Σπουδαστών (Student Information Management Systems). Η ανάπτυξη τέτοιων εφαρμογών επιτρέπει την κεντροποιημένη διαχείριση της πληροφορίας, την αυστηρή αυθεντικοποίηση διαφορετικών ρόλων χρηστών και την παροχή στοχευμένων λειτουργιών που ανταποκρίνεται στις πραγματικές ανάγκες ενός εκπαιδευτικού οργανισμού (Liu κ.ά., 2010). Αυτό ακριβώς το τεχνολογικό και οργανωτικό κενό έρχεται να καλύψει το σύστημα που αναπτύσσεται στο πλαίσιο της παρούσας διπλωματικής εργασίας.

2.2 Λογισμικό Υποστήριξης και Διαχείρισης Φροντιστηρίων

Η μετάβαση από τις παραδοσιακές, χειρόγραφες μεθόδους οργάνωσης στη χρήση εξειδικευμένων πληροφοριακών συστημάτων αποτελεί πλέον μονόδρομο για τη βιωσιμότητα και την ανάπτυξη των σύγχρονων εκπαιδευτικών οργανισμών. Στη παρούσα ενότητα εξετάζεται αναλυτικά η συμβολή του λογισμικού στη διαχείριση των φροντιστηρίων. Αρχικά, αναλύονται τα γενικότερα οφέλη της ψηφιοποίησης, και στη συνέχεια γίνεται επισκόπηση των βασικών καλυπτόμενων λειτουργιών και των υφιστάμενων εμπορικών λύσεων, προκειμένου να αναδειχθούν τα κενά που στοχεύει να καλύψει η προτεινόμενη εφαρμογή.

2.2.1 Οφέλη και Πλεονεκτήματα της Ψηφιοποίησης

Η ενσωμάτωση πληροφοριακών συστημάτων στην καθημερινή λειτουργία ενός εκπαιδευτικού κέντρου επιφέρει πολυδιάστατα οφέλη, τα οποία επεκτείνονται από την εξοικονόμηση χρόνου έως την ενίσχυση της ασφάλειας των δεδομένων. Το σημαντικότερο πλεονέκτημα της ψηφιακής μετάβασης (digital transformation) είναι η δραστική μείωση του διοικητικού φόρτου. Με την αυτοματοποίηση

επαναλαμβανόμενων διαδικασιών – όπως η εγγραφή μαθητών, η μαζική καταχώρηση απουσιών και η παρακολούθηση των διδάκτρων – το εκπαιδευτικό κέντρο αποδεσμεύεται από την γραφειοκρατία. Αυτή η απελευθέρωση πόρων επιτρέπει στη διεύθυνση να επικεντρωθεί στη στρατηγική ανάπτυξη του οργανισμού και στους εκπαιδευτικούς να εστιάσουν αποκλειστικά στο παιδαγωγικό τους έργο (Benavides κ.ά., 2020).

Παράλληλα, η ψηφιοποίηση αναβαθμίζει κατακόρυφα την ταχύτητα και την ακρίβεια της πληροφορίας. Σε αντίθεση με τα κατακερματισμένα λογιστικά φύλλα, η εγκαθίδρυση μιας κεντρικής, σχεσιακής βάσης δεδομένων εγγυάται ότι οι πληροφορίες είναι επικαιροποιημένες, διατηρήσιμες και άμεσα προσβάσιμες σε πραγματικό χρόνο. Όπως προκύπτει από την ακαδημαϊκή έρευνα, η ανάπτυξη συστημάτων διαχείρισης εκπαιδευτηρίων με τη χρήση σύγχρονων εργαλείων – όπως το πλαίσιο (framework) Spring Boot – οδηγεί στην άμεση βελτίωση της συνολικής εργασιακής αποδοτικότητας. Συγκεκριμένα, μέσα από στοχευμένα συστήματα ελέγχου δικαιωμάτων (permission management), οι καθηγητές αποκτούν άμεση και αποτελεσματική εποπτεία των ακαδημαϊκών επιδόσεων των μαθητών, ενώ παράλληλα οι ίδιοι οι σπουδαστές έχουν γρήγορα πρόσβαση αποκλειστικά στα δικά τους δεδομένα, αποφεύγοντας οργανωτικές καθυστερήσεις (Yang, 2022).

2.2.2 Καλυπτόμενες Λειτουργίες Εσωτερικής Διαχείρισης

Για να θεωρηθεί ένα πληροφοριακό σύστημα επαρκές και ικανό να υποστηρίξει εξολοκλήρου τη λειτουργία ενός Φροντιστηρίου, οφείλει να ενσωματώνει ένα συγκεκριμένο σύνολο υποσυστημάτων (modules). Σύμφωνα με τη βιβλιογραφία γύρω από την αρχιτεκτονική των Συστημάτων Διαχείρισης Σπουδαστών (Student Information Management System), οι βασικές λειτουργίες επικεντρώνονται στην ψηφιακή οργάνωση των μητρώων, την κατανομή των μαθημάτων και την παρακολούθηση της ακαδημαϊκής πορείας (Liu κ.ά., 2010).

Ειδικότερα, προκειμένου να καλυφθούν οι πραγματικές ανάγκες ενός εκπαιδευτικού οργανισμού, οι λειτουργίες εσωτερικής διαχείρισης διακρίνονται στους εξής βασικούς άξονες:

- **Διαχείρισης Χρηστών (Μητρώο):** Περιλαμβάνει τη δημιουργία και διαχείριση ξεχωριστών προφίλ με βάση το ρόλο του κάθε χρήστη στον οργανισμό (π.χ. Διαχειριστές, Εκπαιδευτικοί, Σπουδαστές). Η λειτουργία αυτή επιτρέπει την ασφαλή καταχώρηση και επεξεργασία των προσωπικών στοιχείων και των στοιχείων επικοινωνίας των χρηστών (Liu κ.ά., 2010).
- **Διαχείριση Μαθημάτων και Προγράμματος:** Αφορά τη δομική οργάνωση του εκπαιδευτηρίου. Το σύστημα πρέπει να υποστηρίζει τη δημιουργία διακριτών μαθημάτων ή τμημάτων, την κατανομή των διδασκόντων σε αυτά, καθώς και τον ακριβή χρονικό προγραμματισμό (δημιουργία ωρολογίου προγράμματος).
- **Παρακολούθηση Παρουσιών (Απουσιολόγιο):** Αποτελεί βασική λειτουργία για την καθημερινή ροή του φροντιστηρίου. Παρέχεται η δυνατότητα στους εκπαιδευτικούς ή στη γραμματεία να καταγράφουν ψηφιακά τις απουσίες ανά διδακτική ώρα, έχοντας άμεση στατιστική εικόνα της συμμετοχής του κάθε μαθητή.
- **Διαχείριση Βαθμολογιών (Αξιολόγηση):** Συνιστά τον πυρήνα της ακαδημαϊκής παρακολούθησης. Όπως έχει αναδειχθεί σε σύγχρονες υλοποιήσεις αντίστοιχων συστημάτων, η παροχή ενός υποσυστήματος όπου οι καθηγητές μπορούν να καταχωρούν, να επεξεργάζονται και να διαχειρίζονται τις επιδόσεις (achievements) των μαθητών, είναι ζωτικής σημασίας για την άμεση εξαγωγή συμπερασμάτων γύρω από την πρόοδο τους (Yang, 2022).

2.2.3 Επισκόπηση Υφιστάμενων Λύσεων (Γνωστά Λογισμικά)

Στη σύγχρονη αγορά λογισμικού, υφίσταται μια ποικιλία συστημάτων που αποσκοπούν στη διευκόλυνση της εκπαιδευτικής διαδικασίας και της διοικητικής μέριμνας. Οι λύσεις αυτές διακρίνονται κυρίως σε δύο κατηγορίες: στα συστήματα ανοιχτού κώδικα, που εστιάζουν στη Διαχείριση Μάθησης (Learning Management Systems – LMS), και στις εμπορικές πλατφόρμες Διαχείρισης Εκπαιδευτηρίων (Student Information Systems – SIS).

Στην πρώτη κατηγορία δεσπόζει το Moodle, ένα από τα πιο διαδεδομένα συστήματα παγκοσμίως. Το Moodle προσφέρει ισχυρά εργαλεία για την ψηφιακή εκπαίδευση, όπως η ανάρτηση σημειώσεων, η διεξαγωγή διαγωνισμάτων και η βαθμολόγηση. Ωστόσο, η αρχιτεκτονική του εστιάζει σχεδόν αποκλειστικά στο ακαδημαϊκό και μαθησιακό κομμάτι. Ως αποτέλεσμα, αφήνει ακάλυπτες κρίσιμες διοικητικές ανάγκες ενός τυπικού φροντιστηρίου, όπως είναι η δημιουργία πολύπλοκων ωρολογίων προγραμμάτων, η παρακολούθηση παρουσιών σε επίπεδο ανθρώπινου δυναμικού και η γενικότερη εσωτερική οργάνωση (Moodle, χ.χ.).

Στη δεύτερη κατηγορία συναντώνται ολοκληρωμένες εμπορικές πλατφόρμες, όπως το Classter, το οποίο αποτελεί ένα εξαιρετικά ισχυρό σύστημα SIS βασισμένο στο νέφος (cloud-based). Πλατφόρμες αυτού του βεληνεκού ενσωματώνουν πρακτικά κάθε πιθανή λειτουργία (από εγγραφές και παρουσιολόγια μέχρι οικονομικά στοιχεία και εργαλεία CRM). Το μειονέκτημα, ωστόσο, τέτοιων συστημάτων είναι ότι έχουν σχεδιαστεί με γνώμονα τεράστιους εκπαιδευτικούς οργανισμούς (όπως κολέγια, ιδιωτικά σχολεία ή πανεπιστήμια). Ως εκ τούτου, η ενσωμάτωσή τους σε μικρομεσαία Φροντιστήρια Μέσης Εκπαίδευσης κρίνεται συχνά δυσκίνητη, το περιβάλλον χρήσης τους ιδιαίτερα πολύπλοκο για τον μέσο χρήστη, ενώ το κόστος συντήρησης και ετήσιων αδειών παραμένει απαγορευτικό για μικρότερες επιχειρήσεις (Classter, 2025).

Η επισκόπηση αυτών των λύσεων καταδεικνύει ότι, ενώ υπάρχουν διαθέσιμα εργαλεία στην αγορά, συχνά κινούνται στα δύο άκρα: είτε είναι αμιγώς μαθησιακά (LMS) αγνοώντας τις ανάγκες της γραμματείας, είτε είναι υπερβολικά πολύπλοκα και κοστοβόρα (εμπορικά SIS).

2.2.4 Κριτήρια Σύγκρισης και Κοινή Αρχιτεκτονική

Κατά την αξιολόγηση και σύγκριση των υφιστάμενων λογισμικών διαχείρισης εκπαιδευτηρίων, τα πρότυπα της Μηχανικής Λογισμικού (Software Engineering) επιβάλλουν τη χρήση συγκεκριμένων, μετρήσιμων κριτηρίων. Για τον σκοπό αυτό, υιοθετείται το διεθνές πρότυπο αξιολόγησης ποιότητας λογισμικού ISO/IEC 25010, εστιάζοντας σε τέσσερις θεμελιώδεις άξονες: τη λειτουργική καταλληλότητα (functional suitability), τη χρηστικότητα (usability) του περιβάλλοντος διεπαφής, την ασφάλεια (security) των δεδομένων, και τη συντηρησιμότητα (maintainability) του κώδικα στο βάθος του χρόνου (International Organization for Standardization, 2011).

Προκειμένου, ωστόσο, η ανάλυση να ανταποκρίνεται πλήρως στις ρεαλιστικές απαιτήσεις και τους περιορισμούς ενός σύγχρονου εκπαιδευτικού οργανισμού, το θεωρητικό αυτό πλαίσιο εμπλουτίζεται με κρίσιμα πρακτικά κριτήρια, όπως το συνολικό κόστος (αγοράς, συντήρησης και αδειοδότησης), την επεκτασιμότητα (extensibility) για μελλοντική προσθήκη λειτουργιών, καθώς και το καθεστώς ανοικτότητας της άδειας χρήσης (licensing). Η συγκριτική αποτίμηση των επικρατέστερων λύσεων της αγοράς (Moodle, Classter) σε αντιπαραβολή με το προτεινόμενο σύστημα αποτυπώνεται συγκεντρωτικά στον Πίνακα 1.

Πίνακας 1: Συγκριτική αξιολόγηση εκπαιδευτικών λογισμικών

Κριτήριο Αξιολόγησης	Moodle (LMS)	Classter (SIS / Εμπορικό)	Προτεινόμενο Σύστημα
Λειτουργική Καταλληλότητα	Χαμηλή (Μόνο μαθησιακό σκέλος, απουσία διοικητικών εργαλείων)	Πολύ Υψηλή (Πλήρης κάλυψη, αλλά με περιττές λειτουργίες)	Υψηλή (Στοχευμένη ακριβώς στις διοικητικές ανάγκες φροντιστηρίου)
Χρηστικότητα (Usability)	Μέτρια (Απαιτεί εξοικείωση από το προσωπικό)	Χαμηλή (Υπερβολικά πολύπλοκο περιβάλλον για τον μέσο χρήστη)	Υψηλή (Απλό, καθαρό και εστιασμένο περιβάλλον διεπαφής)
Κόστος	Χαμηλό (Δωρεάν λογισμικό, αλλά με κόστος παραμετροποίησης)	Πολύ Υψηλό (Ακριβές ετήσιες άδειες χρήσης και συντήρησης)	Μηδενικό / Χαμηλό (Ανάπτυξη εκ του μηδενός, χωρίς ετήσιες άδειες)
Συντηρησιμότητα & Επεκτασιμότητα	Μέτρια (Πολύπλοκη δομή legacy κώδικα)	Χαμηλή (Κλειστό σύστημα, απόλυτη εξάρτηση από την εταιρεία)	Υψηλή (Σύγχρονη αρχιτεκτονική προσαρμοσμένη για επεκτάσεις)
Άδεια Χρήσης / Ανοικτότητα	Ανοικτού Κώδικα (Open Source)	Εμπορική / Ιδιόκτητη (Proprietary)	Ανοικτή / Προσαρμοσμένη

Από τη συγκριτική ανάλυση προκύπτει ότι το προτεινόμενο σύστημα γεφυρώνει το χάσμα μεταξύ των υφιστάμενων λύσεων, επιτυγχάνοντας τη βέλτιστη ισορροπία. Ενώ οι εμπορικές πλατφόρμες (SIS) προσφέρουν υψηλή λειτουργικότητα με τίμημα το μεγάλο κόστος και την υψηλή πολυπλοκότητα, και τα συστήματα LMS υστερούν στις δομικές διοικητικές ανάγκες, η αναπτυσσόμενη εφαρμογή προσφέρει μια στοχευμένη, εύχρηστη και οικονομικά βιώσιμη λύση, προσαρμοσμένη ακριβώς στις ιδιαιτερότητες ενός μεσαίου φροντιστηριακού οργανισμού.

Όσον αφορά τη δομή των σύγχρονων αυτών πληροφοριακών συστημάτων, η συντριπτική πλειονότητα των διαδικτυακών λύσεων υιοθετεί την αρχιτεκτονική Πελάτη – Εξυπηρετητή (Client-Server), η οποία βασίζεται στον φυσικό και λογικό διαχωρισμό των διεργασιών παρουσίασης και επεξεργασίας (Guynes & Windsor, 2011). Εμβαθύνοντας σε ένα μεσαίο επίπεδο ανάλυσης (mid-level architecture), τα συστήματα αυτά οργανώνονται τυπικά σε μια πολυεπίπεδη δομή τριών επιπέδων (Three-Tier Architecture), όπου αναδεικνύονται συγκεκριμένα κοινά δομικά συστατικά (components) σε κάθε επίπεδο:

- **Επίπεδο Παρουσίασης (Presentation Tier / Client):** Περιλαμβάνει τα κοινά στοιχεία διεπαφής (Web UI), όπως πίνακες ελέγχου (dashboards) για τη γραμματεία, τους διδάσκοντες και τους μαθητές, εξασφαλίζοντας τη διαδραστική επικοινωνία με τον χρήστη.
- **Επίπεδο Επιχειρησιακής Λογικής (Application/Logic Tier / Server):** Αποτελεί τον πυρήνα του συστήματος, όπου υλοποιούνται οι κοινές εκπαιδευτικές και διοικητικές λειτουργίες, όπως η διαχείριση χρηστών και δικαιωμάτων (RBAC), η ανάθεση διδασκόντων στα μαθήματα, η καταγραφή απουσιών και η διαχείριση του ωρολογίου προγράμματος.
- **Επίπεδο Δεδομένων (Data Tier):** Αναλαμβάνει την ασφαλή αποθήκευση, ευρετηρίαση και ανάκτηση των οντοτήτων του συστήματος μέσω ενός σχεσιακού συστήματος διαχείρισης βάσεων δεδομένων (RDBMS).

Αυτός ο πολυεπίπεδος αρχιτεκτονικός διαχωρισμός εξασφαλίζει ότι το backend αναλαμβάνει την απομονωμένη και ασφαλή επικοινωνία με τη βάση δεδομένων, καθώς και την αυστηρή επιβολή των κανόνων ασφαλείας, της επιχειρησιακής λογικής και της αυθεντικοποίησης. Η υλοποίηση τέτοιων δομών επιτυγχάνεται σήμερα με τη χρήση ισχυρών πλαισίων ανάπτυξης (frameworks), όπως το Spring Boot, το οποίο προσφέρει ενσωματωμένους μηχανισμούς για την ταχεία και ασφαλή ανάπτυξη αυτόνομων εφαρμογών επιπέδου παραγωγής (Spring, 2026). Πάνω σε αυτή ακριβώς τη σχεδιαστική, πολυεπίπεδη και τεχνολογική φιλοσοφία βασίζεται η εφαρμογή που αναπτύσσεται στην παρούσα εργασία.

2.2.5 Τρέχοντα Κενά και η Συνεισφορά της Παρούσας Εργασίας

Από την εκτενή ανασκόπηση της βιβλιογραφίας και την επισκόπηση των υφιστάμενων εφαρμογών, καταδεικνύεται μια σαφής διχοτόμηση στην αγορά των συστημάτων υποστήριξης της εκπαιδευτικής διαδικασίας. Από τη μία πλευρά, διατίθενται πλατφόρμες ανοιχτού κώδικα με αμιγώς μαθησιακό προσανατολισμό, οι οποίες εστιάζουν στην οργάνωση του εκπαιδευτικού υλικού, αλλά αφήνουν ακάλυπτες τις εξειδικευμένες διοικητικές και οργανωτικές ανάγκες της εσωτερικής διαχείρισης ενός φροντιστηρίου (Moodle, χ.χ.). Από την άλλη πλευρά, προσφέρονται ολοκληρωμένα εμπορικά πληροφοριακά συστήματα διαχείρισης σπουδαστών, τα οποία όμως έχουν σχεδιαστεί για μεγάλους εκπαιδευτικούς οργανισμούς, με αποτέλεσμα η δομή τους να είναι ιδιαίτερα σύνθετη, δυσκίνητη στην παραμετροποίηση και οικονομικά απρόσιτη για μικρότερες δομές (Classter, 2025).

Επομένως, το ουσιαστικό κενό εντοπίζεται στην έλλειψη μιας ενδιάμεσης, ευέλικτης και στοχευμένης λύσης, η οποία να προσφέρει τις απαραίτητες λειτουργίες εσωτερικής διοίκησης (όπως το ψηφιακό μητρώο, η παρακολούθηση παρουσιών και η διαχείριση βαθμολογιών) χωρίς την περιττή πολυπλοκότητα των μεγάλων συστημάτων της αγοράς. Όταν οι υφιστάμενες λύσεις αδυνατούν να ευθυγραμμιστούν οικονομικά και λειτουργικά με τις ακριβείς προδιαγραφές, η ανάπτυξη λογισμικού ειδικού σκοπού (domain-specific software) αποτελεί την ενδεδειγμένη επιλογή. Όμως, αντί της δημιουργίας ενός αποκλειστικά κατά παραγγελία (custom) συστήματος για έναν μεμονωμένο πελάτη, προτείνεται μια προσέγγιση που εστιάζει στην ανάπτυξη μιας λύσης που καλύπτει το συνολικό προφίλ και τις απαιτήσεις μιας ολόκληρης κλάσης οργανισμών, εν προκειμένω των μικρομεσαίων εκπαιδευτηρίων, εξασφαλίζοντας έτσι ευρύτερη εφαρμοσιμότητα και υψηλή καταλληλότητα (Iso, 2011).

Η παρούσα διπλωματική εργασία έρχεται να καλύψει αυτό ακριβώς το τεχνολογικό και οργανωτικό κενό. Η συνεισφορά της συνίσταται στον σχεδιασμό και την υλοποίηση μιας εξειδικευμένης διαδικτυακής εφαρμογής ειδικού σκοπού, βασισμένης στην καθιερωμένη αρχιτεκτονική Πελάτη – Εξυπηρετητή (Guynes & Windsor, 2011). Το σύστημα διαχωρίζει πλήρως το τμήμα της διεπαφής χρήστη από την επιχειρησιακή λογική, αξιοποιώντας τις δυνατότητες του σύγχρονου πλαισίου ανάπτυξης Spring Boot για τη δημιουργία ενός ασφαλούς, γρήγορου και επεκτάσιμου περιβάλλοντος διαχείρισης (Yang, 2022; Spring, 2026). Με αυτόν τον τρόπο, η εργασία προσφέρει μια ρεαλιστική και άμεσα εφαρμόσιμη πρόταση για τον ψηφιακό μετασχηματισμό και την αποδοτική λειτουργία των μικρομεσαίων Φροντιστηρίων Μέσης Εκπαίδευσης. Η διασφάλιση ότι η εφαρμογή ανταποκρίνεται στις πραγματικές ανάγκες αυτών των οργανισμών πηγάζει από την άμεση παρατήρηση και γνώση του τομέα της ιδιωτικής εκπαίδευσης (domain knowledge). Ειδικότερα, κατά τη φάση της ανάλυσης απαιτήσεων, χαρτογραφήθηκαν οι καθημερινές διοικητικές και εκπαιδευτικές προκλήσεις (π.χ. διαχείριση προγραμμάτων διδασκόντων, καταχώρηση δραστηριοτήτων ανά μάθημα, παρακολούθηση βαθμολογιών, καθώς και η οργάνωση των αναθέσεων καθηγητών στα αντίστοιχα τμήματα από τη διοίκηση) από την

οπτική γωνία των πραγματικών ροών εργασίας, εξασφαλίζοντας ότι το προτεινόμενο σύστημα εστιάζει στην επίλυση ρεαλιστικών και πρακτικών προβλημάτων.

3

Ανάπτυξη Συστήματος

Στο παρόν κεφάλαιο περιγράφεται αναλυτικά η μετάβαση από το θεωρητικό πλαίσιο της εργασίας στην πρακτική ανάπτυξη του (αντίστοιχου προτεινόμενου) λογισμικού. Ουσιαστικά, παρουσιάζεται η αυστηρή μηχανική διαδικασία που ακολουθήθηκε για τη μετατροπή των επιχειρησιακών αναγκών ενός φροντιστηρίου σε ένα λειτουργικό, ασφαλές και αποδοτικό Πληροφοριακό Σύστημα. Η δομή του κεφαλαίου καθοδηγείται από τα στάδια του Κύκλου Ζωής Ανάπτυξης Συστήματος (SDLC), ξεκινώντας από τη συγκέντρωση των απαιτήσεων, προχωρώντας στον αρχιτεκτονικό σχεδιασμό και καταλήγοντας στην παροχή σημαντικών λεπτομερειών όσον αφορά την υλοποίησή του. Το κεφάλαιο ολοκληρώνεται με μια τελευταία ενότητα, η οποία αξιολογεί τον βαθμό ικανοποίησης των απαιτήσεων που τέθηκαν από το υλοποιημένο σύστημα.

3.1 Εισαγωγή στη Μεθοδολογία Ανάπτυξης

Για την ανάπτυξη της συγκεκριμένης διαδικτυακής εφαρμογής υιοθετήθηκε το Μοντέλο του Καταρράκτη (Waterfall Model). Πρόκειται για την πλέον κλασική και καθιερωμένη μεθοδολογία στη Μηχανική Λογισμικού, η οποία προτείνει μια αυστηρά γραμμική και ακολουθιακή προσέγγιση (Pressman, 2005). Σύμφωνα με το μοντέλο αυτό, η διαδικασία ανάπτυξης διαχωρίζεται σε διακριτές, μη επικαλυπτόμενες φάσεις: τη μελέτη απαιτήσεων, τον σχεδιασμό, την υλοποίηση (συγγραφή κώδικα), τον έλεγχο και τη συντήρηση.

Όπως επισημαίνεται στη βιβλιογραφία, το βασικό χαρακτηριστικό του μοντέλου του Καταρράκτη είναι ότι η κάθε φάση πρέπει να έχει ολοκληρωθεί, ελεγχθεί και τεκμηριωθεί πλήρως προτού ξεκινήσει η επόμενη. Η ροή της εργασίας κινείται σταθερά προς τα κάτω, εξασφαλίζοντας υψηλό επίπεδο ελέγχου και προβλεψιμότητας σε κάθε βήμα της ανάπτυξης (Balaji & Murugaiyan, 2012).

Η επιλογή του συγκεκριμένου μοντέλου για την παρούσα εργασία δεν έγινε τυχαία, αλλά βασίστηκε στα εξής τρία κρίσιμα πλεονεκτήματα που ταυτίζονται με τη φύση του έργου:

1. **Σταθερότητα Απαιτήσεων:** Οι λειτουργίες που πρέπει να επιτελεί ένα φροντιστήριο (εγγραφές, απουσίες, μαθήματα) είναι γνωστές εξ αρχής, ξεκάθαρες και δεν μεταβάλλονται δυναμικά κατά τη διάρκεια του έργου, γεγονός που καθιστά ιδανική τη σειριακή προσέγγιση.
2. **Αυστηρή Τεκμηρίωση:** Το μοντέλο επιβάλλει τη δημιουργία αναλυτικών διαγραμμάτων πριν από τη φάση της προγραμματιστικής υλοποίησης, εξασφαλίζοντας την αρτιότητα, τον ακριβή σχεδιασμό της συνολικής αρχιτεκτονικής του συστήματος (συμπεριλαμβανομένης και της βάσης δεδομένων) και την πλήρη ιχνηλασιμότητα (traceability) των λειτουργιών του.
3. **Σαφής Δομή:** Παρέχει ένα ξεκάθαρο, γραμμικό χρονοδιάγραμμα, διευκολύνοντας τον προγραμματισμό των παραδοτέων, την αξιολόγηση των ενδιάμεσων ορόσημων (milestones) και τη συνολική διαχείριση του έργου (project management), το οποίο είναι σημαντικό στο πλαίσιο εκπόνησης διπλωματικής εργασίας.

3.2 Ανάλυση Απαιτήσεων

Το πρώτο και καθοριστικότερο στάδιο στο μοντέλο του Καταρράκτη, όπως εφαρμόστηκε στην παρούσα εργασία, είναι η πλήρης ανάλυση και ορισμός των απαιτήσεων (Requirements Analysis & Definition). Σε αυτή τη φάση καταγράφονται με ακρίβεια οι ανάγκες που καλείται να καλύψει το πληροφοριακό σύστημα, ώστε να διασφαλιστεί ότι το τελικό προϊόν θα ανταποκρίνεται πλήρως στις λειτουργικές προδιαγραφές ενός σύγχρονου φροντιστηρίου. Οι απαιτήσεις αυτές προέκυψαν από τη μελέτη του θεωρητικού υποβάθρου (Κεφάλαιο 2) και κατηγοριοποιούνται σε λειτουργικές (functional), οι οποίες περιγράφουν τις ενέργειες που πρέπει να εκτελεί το σύστημα, και μη λειτουργικές (non-functional), οι οποίες καθορίζουν τα ποιοτικά του χαρακτηριστικά. Οι λειτουργικές απαιτήσεις χωρίζονται επιπλέον σε 2 κατηγορίες με βάση την κύρια επιχειρησιακή οντότητα που αφορούν (π.χ. μάθημα, χρήστης, κλπ.).

Προτού, ωστόσο, καθοριστούν οι επιμέρους λειτουργίες, είναι απαραίτητος ο προσδιορισμός των διακριτών κατηγοριών χρηστών (actors) που θα αλληλοεπιδρούν με το σύστημα, καθώς κάθε κατηγορία χρηστών αντιστοιχεί σε διαφορετικά δικαιώματα πρόσβασης στο σύστημα και τις λειτουργίες του.

3.2.1 Είδη Χρηστών (Οντότητες)

Το πληροφοριακό σύστημα διαχείρισης φροντιστηρίου έχει σχεδιαστεί ώστε να υποστηρίζει τέσσερις (4) διακριτές βαθμίδες χρηστών, καθεμιά από τις οποίες απαιτεί διαφορετικά δικαιώματα πρόσβασης και προσαρμοσμένο περιβάλλον διεπαφής (χρήστη):

1. **Διαχειριστής (Administrator):** Αποτελεί τον ανώτατο χρήστη του συστήματος (συνήθως τη διεύθυνση ή τη γραμματεία του φροντιστηρίου). Έχει πλήρη και ανεμπόδιστη πρόσβαση σε όλες τις λειτουργίες. Είναι αποκλειστικά υπεύθυνος για την εγγραφή νέων χρηστών, τη διαχείριση του ανθρώπινου δυναμικού, τη δημιουργία των μαθημάτων, τον καθορισμό των τμημάτων και την ανάθεση των εκπαιδευτικών στα αντίστοιχα τμήματα. Επιπλέον, αναλαμβάνει τον αποκλειστικό έλεγχο (έγκριση ή απόρριψη) των αιτημάτων ένταξης και αποχώρησης που υποβάλλουν οι σπουδαστές, καθώς και τη συνολική δόμηση του ωρολογίου προγράμματος.
2. **Εκπαιδευτικός (Teacher):** Είναι ο χρήστης που αναλαμβάνει το αμιγώς διδακτικό και παιδαγωγικό έργο. Μέσω του προσωπικού του πίνακα ελέγχου (dashboard), έχει τη δυνατότητα να

προβάλλει το εβδομαδιαίο του πρόγραμμα και τα τμήματα που του έχουν ανατεθεί. Στις βασικές του αρμοδιότητες περιλαμβάνονται η καταχώρηση δραστηριοτήτων μαθήματος (lesson activities), η βαθμολόγηση των μαθητών των τμημάτων του, καθώς και η δήλωση των δικών του απουσιών. Συγκεκριμένα, ο εκπαιδευτικός έχει τη δυνατότητα να δηλώσει απουσία είτε για ένα συγκεκριμένο προγραμματισμένο μάθημα (μερική απουσία), είτε για ολόκληρη την ημέρα (ολοήμερη απουσία), ώστε να ενημερώνεται δυναμικά το πρόγραμμα του φροντιστηρίου. Επιπλέον, του παρέχεται η δυνατότητα πλήρους παραμετροποίησης του λογαριασμού του, όπως η αλλαγή κωδικού πρόσβασης, η μεταφόρτωση προσωπικής εικόνας (avatar) και η επιλογή προσαρμοσμένου χρωματικού θέματος (template) για τη διεπαφή του.

3. **Σπουδαστής / Μαθητής (Student):** Αποτελεί τον τελικό αποδέκτη των εκπαιδευτικών υπηρεσιών. Μέσα από ένα αυστηρά προσωποποιημένο περιβάλλον, έχει τη δυνατότητα να παρακολουθεί το προσωπικό του εβδομαδιαίο πρόγραμμα, ενημερωμένος παράλληλα για τυχόν ακυρώσεις λόγω δηλωμένης απουσίας κάποιου εκπαιδευτικού. Για κάθε ενεργό μάθημα, μπορεί να προβάλλει τις αναρτημένες δραστηριότητες (lesson activities) και να ενημερώνεται για τις βαθμολογίες του στα αντίστοιχα διαγωνίσματα. Μία από τις πλέον διαδραστικές λειτουργίες του ρόλου αυτού είναι η ικανότητα υποβολής αιτημάτων ένταξης (enroll request) ή αποχώρησης (drop request) από συγκεκριμένα τμήματα, τα οποία στη συνέχεια προωθούνται προς αξιολόγηση και έγκριση από τον Διαχειριστή. Τέλος, όπως και οι εκπαιδευτικοί, ο σπουδαστής έχει πλήρη έλεγχο της διαχείρισης του προφίλ του (ενημέρωση κωδικού, avatar, επιλογή χρωματικού θέματος).
4. **Μη Εγγεγραμμένος Χρήστης / Επισκέπτης (Guest):** Πρόκειται για οποιονδήποτε χρήστη πλοηγείται στην εφαρμογή χωρίς να έχει πραγματοποιήσει ταυτοποίηση (login). Η πρόσβασή του περιλαμβάνει την προβολή της αρχικής, δημόσιας σελίδας του φροντιστηρίου, καθώς και των φορμών εισόδου (login) και δημιουργίας νέου λογαριασμού (register). Το σημαντικότερο, ωστόσο, χαρακτηριστικό αυτής της βαθμίδας είναι η δυνατότητα ελεύθερης περιήγησης στο προσφερόμενο πρόγραμμα σπουδών. Ο επισκέπτης μπορεί να δει τη συνολική λίστα των μαθημάτων του φροντιστηρίου και, επιλέγοντας οποιοδήποτε από αυτά, να ενημερωθεί αναλυτικά για τα ενεργά του τμήματα, τις ημέρες και ώρες διεξαγωγής τους, καθώς και για τον εκάστοτε εκπαιδευτικό που έχει αναλάβει τη διδασκαλία. Με αυτόν τον τρόπο, το σύστημα παρέχει πλήρη διαφάνεια και άμεση ενημέρωση στους ενδιαφερόμενους, πριν καν απαιτηθεί η εγγραφή τους.

3.2.2 Λειτουργικές Απαιτήσεις Διαχείρισης Χρηστών

Οι λειτουργικές απαιτήσεις διαχείρισης χρηστών περιγράφουν τις απαραίτητες διεργασίες για την ασφαλή είσοδο στο σύστημα, την εγγραφή νέων μελών και τη συντήρηση των προφίλ. Με βάση την ανάλυση των αναγκών του φροντιστηρίου, το σύστημα οφείλει να υποστηρίξει τις ακόλουθες λειτουργίες:

- **Ταυτοποίηση και Πρόσβαση (Authentication):** Το σύστημα πρέπει να παρέχει μηχανισμό ασφαλούς σύνδεσης (login) και αποσύνδεσης (logout) για όλους τους εγγεγραμμένους χρήστες, αναγνωρίζοντας αυτόματα το ρόλο τους (Διαχειριστής, Καθηγητής, Μαθητής) προκειμένου να τους κατευθύνει στο αντίστοιχο περιβάλλον εργασίας. Η αυθεντικοποίηση θα πραγματοποιείται με κλασικά διαπιστευτήρια (όνομα χρήστη και κωδικό πρόσβασης), με τους κωδικούς να αποθηκεύονται αυστηρά κρυπτογραφημένοι στη βάση δεδομένων. Για τη διαχείριση της συνεδρίας (session) υιοθετείται το πρότυπο JSON Web Token (JWT) μέσω ασφαλών HttpOnly cookies.

- **Εγγραφή Νέων Χρηστών (Registration):** Το σύστημα πρέπει να επιτρέπει τη δημιουργία νέων λογαριασμών μέσα από διακριτές φόρμες εγγραφής, τόσο για τους μαθητές (student registration) όσο και για τους εκπαιδευτικούς (teacher registration). Η διαδικασία εγγραφής υπακούει στους παρακάτω κανόνες:
 - **Διαφοροποίηση Πληροφοριών:** Αν και υφίσταται ένας κοινός κορμός δεδομένων για όλους (όνομα χρήστη, email, ονοματεπώνυμο, διεύθυνση), συλλέγονται επιπλέον εξειδικευμένες πληροφορίες ανά ρόλο. Για παράδειγμα, στους μαθητές καταχωρούνται στοιχεία, όπως η ηλικία, σχολική τάξη, ονοματεπώνυμο και ΑΦΜ κηδεμόνα, ενώ στους εκπαιδευτικούς στοιχεία, όπως η ειδικότητα και το βιογραφικό.
 - **Έλεγχοι Εγκυρότητας:** Πραγματοποιείται αυστηρός έλεγχος μοναδικότητας (uniqueness validation) στο όνομα χρήστη και στο email. Παράλληλα, κατά την πληκτρολόγηση του κωδικού πρόσβασης, γίνεται δυναμικός έλεγχος της ισχύος του (password strength) μέσω προειδοποιήσεων στη διεπαφή του χρήστη (client-side).
 - **Αρχική Κατάσταση:** Όλοι οι λογαριασμοί κατά την εγγραφή τους δημιουργούνται προεπιλεγμένα σε ανενεργή κατάσταση (disabled) και η είσοδός τους στο σύστημα μπλοκάρεται μέχρι να εγκριθούν / ενεργοποιηθούν από τον Διαχειριστή.
 - **Δημιουργία Διαχειριστή:** Ο λογαριασμός του Διαχειριστή δεν εγγράφεται μέσω κάποιας δημόσιας φόρμας, αλλά εισάγεται και αρχικοποιείται αυτόματα κατά την έναρξη του συστήματος (system startup) μέσω ειδικού σεναρίου βάσης δεδομένων (SQL script), σύμφωνα με την παραμετροποίηση της εφαρμογής.
- **Διαχείριση Προσωπικού Προφίλ:** Κάθε ταυτοποιημένος χρήστης πρέπει να έχει τη δυνατότητα προβολής και επεξεργασίας των προσωπικών του στοιχείων. Αναφορικά με τη διαχείριση των διαπιστευτηρίων, διευκρινίζεται ότι το σύστημα δεν υποστηρίζει λειτουργία ανάκτησης (password recovery) μέσω εξωτερικής επιβεβαίωσης σε περίπτωση απώλειας. Αντιθέτως, η λειτουργία περιορίζεται στην ασφαλή αλλαγή κωδικού (password change), η οποία είναι εφικτή αποκλειστικά για τους ήδη συνδεδεμένους χρήστες. Η ενέργεια αυτή πραγματοποιείται μέσω από το αντίστοιχο μενού της διεπαφής τους, όπου απαιτείται η εισαγωγή του τρέχοντα κωδικού και η διπλή πληκτρολόγηση του νέου προς επιβεβαίωση.
- **Κεντρική Διαχείριση Λογαριασμών (Από τον Διαχειριστή):** Ο διαχειριστής του συστήματος πρέπει να διαθέτει ένα κεντρικό περιβάλλον (Master View) από όπου θα μπορεί να προβάλει το σύνολο των εγγεγραμμένων χρηστών (ή να αναζητά συγκεκριμένους χρήστες), να ενεργοποιεί (activate) τους εκκρεμείς λογαριασμούς που δημιουργήθηκαν από τη διαδικασία εγγραφής και να διαγράφει οριστικά χρήστες όταν αυτό κρίνεται απαραίτητο.

3.2.3 Λειτουργικές Απαιτήσεις Διαχείρισης Μαθημάτων & Προγράμματος

Αυτή η κατηγορία απαιτήσεων αποτελεί τον πυρήνα της επιχειρησιακής λογικής του συστήματος, καθώς ψηφιοποιεί πλήρως την καθημερινή ροή του εκπαιδευτηρίου. Οι λειτουργίες διαχωρίζονται ανάλογα με τον ρόλο του εκκινήτη (primary actor) της εκάστοτε περίπτωσης χρήσης, ως εξής:

Α. Δόμηση Εκπαιδευτικού Προγράμματος (Διαχειριστής):

- **Διαχείριση Μαθημάτων (Courses):** Ο διαχειριστής πρέπει να έχει τη δυνατότητα δημιουργίας, επεξεργασίας, προβολής και διαγραφής (CRUD) των προσφερόμενων μαθημάτων. (Σημειώνεται ότι η προβολή των μαθημάτων είναι ανοιχτή και στο ευρύ, μη εγγεγραμμένο κοινό).
- **Ανάθεση Εκπαιδευτικών στα Τμήματα:** Το σύστημα πρέπει να επιτρέπει στον διαχειριστή να αναθέτει (assign) ή να αφαιρεί (unassign) συγκεκριμένους εκπαιδευτικούς ανά τμήμα (scheduled slot) του εκάστοτε μαθήματος, διασφαλίζοντας την ακριβή κατανομή του διδακτικού προσωπικού.
- **Δημιουργία Τμημάτων (Scheduled Slots) και Ωρολογίου Προγράμματος:** Στο πλαίσιο του συστήματος, το εκάστοτε τμήμα ταυτίζεται με μία προγραμματισμένη ώρα διδασκαλίας (scheduled slot) που περιλαμβάνει συγκεκριμένη μέρα, ώρα, εκπαιδευτικό και αίθουσα. Ο διαχειριστής διαθέτει πλήρη δικαιώματα διαχείρισης (CRUD) επί αυτών των τμημάτων, έχοντας τη δυνατότητα όχι μόνο να τα δημιουργεί και να τα διαγράφει, αλλά και να επεξεργάζεται τα στοιχεία τους.

B. Ροή Εγγραφών Σπουδαστών (Enrollment Workflow):

- **Αιτήματα Σπουδαστών:** Οι μαθητές πρέπει να έχουν τη δυνατότητα να υποβάλλουν ηλεκτρονικό αίτημα ένταξης (enroll) σε ένα προγραμματισμένο τμήμα, αλλά και αίτημα αποχώρησης (drop) από αυτό. Η τρέχουσα κατάσταση κάθε αιτήματος (π.χ. Pending, Active) είναι άμεσα ορατή στο προσωπικό περιβάλλον του σπουδαστή.
- **Έγκριση / Απόρριψη Αιτημάτων:** Ο διαχειριστής λαμβάνει τα εκκρεμή αιτήματα (enroll/drop requests) και διαθέτει το δικαίωμα να τα εγκρίνει (approve) ή να τα απορρίπτει (reject), διατηρώντας τον απόλυτο έλεγχο της χωρητικότητας των τμημάτων. Η απόφασή του ενημερώνει δυναμικά το status του αιτήματος στο dashboard του μαθητή, χωρίς να απαιτείται προς το παρόν επιπλέον γραπτή αιτιολόγηση απόρριψης.

Γ. Εκπαιδευτική Διαδικασία (Εκπαιδευτικοί):

- **Προβολή Προγράμματος και Μαθητών:** Οι εκπαιδευτικοί πρέπει να μπορούν να βλέπουν το ατομικό τους ωρολόγιο πρόγραμμα και να προβάλλουν τη λίστα των ενεργών μαθητών (roster) για κάθε τμήμα τους.
- **Δραστηριότητες Μαθήματος (Lesson Activities):** Το σύστημα πρέπει να επιτρέπει στους καθηγητές να δημιουργούν, να επεξεργάζονται και να διαγράφουν δραστηριότητες / καταγραφές για τις παραδόσεις τους (όπως το ημερήσιο Classwork της τάξης και η ανάθεση του επόμενου homework), τις οποίες στη συνέχεια μπορούν να βλέπουν οι μαθητές.
- **Διαχείριση Απουσιών:** Οι εκπαιδευτικοί πρέπει να μπορούν να δηλώνουν μελλοντικές απουσίες, καθώς και να τις επεξεργάζονται ή να τις διαγράφουν, ενημερώνοντας έμμεσα το πρόγραμμα. (Σημειώνεται πως όλες οι δηλωθείσες απουσίες είναι άμεσα ορατές και πλήρως διαχειρίσιμες από τον διαχειριστή, όπως περιγράφεται στην ενότητα του Κεντρικού Ελέγχου).
- **Αξιολόγηση (Tests & Grades):** Το σύστημα οφείλει να παρέχει στους καθηγητές τη δυνατότητα δημιουργίας διαγωνισμάτων, ανάθεσής τους στα τμήματα, και καταχώρησης της τελικής βαθμολογίας για τον κάθε μαθητή. Λόγω της βαρύτητας και της πολύπλοκης φύσης της διαδικασίας αξιολόγησης, απαιτείται η αντιμετώπιση των διαγωνισμάτων (Tests) ως ξεχωριστών οντοτήτων από τις καθημερινές Δραστηριότητες (Lesson Activities). Ο διαχωρισμός αυτός καθίσταται απαραίτητος διότι τα διαγωνίσματα πρέπει να διατηρούν το δικό τους ξεχωριστό ιστορικό, να συνδέονται άμεσα με εξατομικευμένες “κόλλες” επιδόσεων (Test Results) των μαθητών και να παρέχουν εξειδικευμένες προβολές παρακολούθησης (π.χ. ποσοστιαία πρόοδος

ολοκλήρωσης της βαθμολόγησης ενός τμήματος), απαιτώντας έτσι ειδική μεταχείριση και ανεξάρτητη διαχείριση από το σύστημα.

Δ. Κεντρικός Έλεγχος & Εποπτεία (Master Overrides):

- Για λόγους ασφαλείας και κεντρικής διαχείρισης δεδομένων, ο διαχειριστής πρέπει να διατηρεί υπερεξουσίες (master override) επί όλων των οντοτήτων του συστήματος. Ουσιαστικά, ο ρόλος του διαχειριστή απαιτείται να διαθέτει πλήρη δικαιώματα λειτουργιών διαχείρισης (CRUD – Create, Read, Update, Delete) για το σύνολο των δεδομένων. Μέσω μιας κεντρικής διεπαφής ελέγχου (Audit Panel), ο διαχειριστής οφείλει να έχει τη δυνατότητα να εποπτεύει, να τροποποιεί ή να διαγράφει εγγραφές που αφορούν δραστηριότητες μαθημάτων, απουσίες εκπαιδευτικών, διαγωνίσματα και βαθμολογίες. Η παροχή αυτής της λειτουργικότητας κρίνεται απαραίτητη ώστε να μπορεί να διορθώνει τυχόν λάθη ή παραλείψεις των χρηστών, διασφαλίζοντας την εύρυθμη λειτουργία του εκπαιδευτηρίου.

3.2.4 Μη Λειτουργικές Απαιτήσεις

Ενώ οι λειτουργικές απαιτήσεις περιγράφουν τις απαραίτητες λειτουργίες και υπηρεσίες του συστήματος, οι μη λειτουργικές απαιτήσεις (Non-Functional Requirements) καθορίζουν τα ποιοτικά χαρακτηριστικά, τους τεχνικούς περιορισμούς και τα πρότυπα λειτουργίας του. Βασίζόμενοι στο πρότυπο αξιολόγησης ISO/IEC 25010 που αναλύθηκε στο θεωρητικό υπόβαθρο της εργασίας, καθώς και σε σύγχρονες αρχές μηχανικής λογισμικού που επιτάσσουν αυξημένη φορητότητα (portability) και κλιμακωσιμότητα (scalability), καθορίστηκαν οι ακόλουθες απαιτήσεις για το Πληροφοριακό Σύστημα του Φροντιστηρίου.

Α. Ασφάλεια (Security) και Ιδιωτικότητα

Δεδομένου ότι το σύστημα διαχειρίζεται ευαίσθητα προσωπικά δεδομένα (μητρώα μαθητών, βαθμολογίες, απουσίες), η ασφάλεια αποτελεί την κρίσιμότερη με λειτουργική απαίτηση.

- **Αυθεντικοποίηση και Εξουσιοδότηση:** Το σύστημα πρέπει να διασφαλίζει την πρόσβαση στους πόρους του αποκλειστικά σε αυθεντικοποιημένους χρήστες, εφαρμόζοντας σύγχρονους μηχανισμούς διακριτικών (token-based authentication). Επιπλέον, πρέπει να εφαρμόζεται αυστηρός Έλεγχος Πρόσβασης Βάσει Ρόλων (RBAC), αποτρέποντας την κλιμάκωση προνομίων (π.χ. ένας μαθητής δεν πρέπει να έχει πρόσβαση σε δεδομένα άλλου μαθητή ή σε εργαλεία διαχειριστή).
- **Προστασία Δεδομένων:** Η διακίνηση των δεδομένων αυθεντικοποίησης (όπως τα tokens) καθώς και όλων των ευαίσθητων επιχειρησιακών δεδομένων πρέπει να γίνεται μέσω ασφαλών καναλιών επικοινωνίας, επιβάλλοντας τη χρήση του πρωτοκόλλου HTTPS στο περιβάλλον παραγωγής. Παράλληλα, το σύστημα οφείλει να ενσωματώνει αυστηρούς μηχανισμούς άμυνας: χρήση HTTP-only και SameSite=Strict cookies για τα JWT tokens προς αποτροπή επιθέσεων XSS και CSRF αντίστοιχα, καθώς και συστηματική επικύρωση δεδομένων (data validation) σε συνδυασμό με παραμετροποιημένες επρωτήσεις (parameterized queries) κατά την αλληλεπίδραση με τη βάση δεδομένων (μέσω του ORM), για την απόλυτη προστασία (των δεδομένων) έναντι επιθέσεων SQL Injection.
- **Κατακερματισμός (Hashing):** Οι κωδικοί πρόσβασης των χρηστών απαγορεύεται να αποθηκεύονται σε μορφή απλού κειμένου (plaintext) στη βάση δεδομένων και πρέπει να κατακερματίζονται (hashing) με ισχυρούς αλγορίθμους. Συγκεκριμένα, το σύστημα απαιτείται να αξιοποιεί τον αλγόριθμο BCrypt (μέσω της αντίστοιχης υλοποίησης BCryptPasswordEncoder), ο

οποίος δεν εφαρμόζει απλώς ισχυρό κατακερματισμό, αλλά παράγει και ενσωματώνει αυτόματα μια μοναδική, τυχαία κρυπτογραφική ακολουθία (cryptographic salt) για τον κάθε κωδικό. Η τεχνική αυτή διασφαλίζει τη μέγιστη δυνατή προστασία έναντι επιθέσεων προσχεδιασμένων πινάκων (όπως τα Rainbow Tables) και εξαντλητικής αναζήτησης.

B. Επεκτασιμότητα (Scalability) και Συντηρησιμότητα (Maintainability)

- **Αρχιτεκτονική Αποσύζευξης (Decoupling) και Απομόνωση Πυρήνα:** Η επιχειρησιακή λογική (domain logic) του συστήματος αποτελεί τον κρίσιμότερο τομέα της εφαρμογής. Συνεπώς, τίθεται ως αυστηρή μη λειτουργική απαίτηση η πλήρης απομόνωσή της από εξωτερικές τεχνολογίες, πλαίσια ανάπτυξης (frameworks) και μηχανισμούς βάσεων δεδομένων. Για την ικανοποίηση αυτής της απαίτησης, επιβάλλεται η υιοθέτηση του προτύπου της Εξαγωνικής Αρχιτεκτονικής (Hexagonal Architecture ή Ports and Adapters). Το σύστημα πρέπει να σχεδιαστεί με τρόπο που να επιτρέπει την αλλαγή ενός εξωτερικού στοιχείου (π.χ. της διεπαφής χρήστη ή της βάσης δεδομένων) χωρίς να επηρεάζεται ή να αναδιαγράφεται ο πυρήνας της εφαρμογής.
- **Διαλειτουργικότητα και Τυποποίηση Δεδομένων:** Η επικοινωνία των εξωτερικών συστημάτων με τον πυρήνα της εφαρμογής πρέπει να γίνεται μέσω αυστηρά καθορισμένων διεπαφών RESTful (REST APIs), τα οποία να ικανοποιούν τουλάχιστον το Επίπεδο 2 του μοντέλου ωριμότητας Richardson (Vassiliou-Gioles, 2020), διαχωρίζοντας ορθά την ανάκτηση από τον χειρισμό δεδομένων μέσω των αντίστοιχων HTTP ρημάτων. Η ανταλλαγή δεδομένων απαιτείται να γίνεται μέσω ελαφριών προτύπων (JSON), διασφαλίζοντας την εύκολη ενσωμάτωση μελλοντικών πελατών. Επιπρόσθετα, καθίσταται υποχρεωτική η παροχή ζωντανής, διαδικτυακής τεκμηρίωσης για το σύνολο των APIs, αξιοποιώντας εργαλεία όπως το Swagger / OpenAPI 3.

Γ. Απόδοση (Performance)

- **Ταχύτητα Απόκρισης και Φόρτος Εργασίας:** Το σύστημα πρέπει να είναι σε θέση να εξυπηρετεί ταυτόχρονα τα αιτήματα όλων των χρηστών (σπουδαστών, εκπαιδευτικών και διαχειριστών) χωρίς αισθητές καθυστερήσεις, διατηρώντας υψηλά επίπεδα απόδοσης υπό συνθήκες τυπικού φόρτου λειτουργίας ενός φροντιστηρίου. Η αρχιτεκτονική της εφαρμογής οφείλει να στοχεύει στην ελαχιστοποίηση του χρόνου απόκρισης του διακομιστή μέσω της βέλτιστης διαχείρισης των πόρων (π.χ. αποδοτικά ερωτήματα στη βάση δεδομένων). Βασική απαίτηση αποτελεί η διατήρηση της ομαλής ροής αλληλεπίδρασης, παρέχοντας άμεση ανατροφοδότηση στον χρήστη και αποτρέποντας φαινόμενα καθυστερήσεων που θα μπορούσαν να υποβαθμίσουν τη συνολική εμπειρία χρήσης του συστήματος.

Δ. Χρηστικότητα (Usability)

- **Διεπαφή Χρήστη (UI):** Το γραφικό περιβάλλον πρέπει να είναι διαισθητικό και απόλυτα φιλικό προς τον χρήστη (user-friendly), διατηρώντας ένα ενιαίο και συνεπές οπτικό αποτέλεσμα (unified look-and-feel) σε όλες τις επιμέρους οθόνες. Η πλοήγηση (ειδικά στους ρόλους του σπουδαστή και του εκπαιδευτικού) οφείλει να καθοδηγεί τον χρήστη με φυσικό τρόπο, χωρίς να απαιτείται προηγούμενη εκπαίδευση για την αξιοποίηση του συστήματος.
- **Προσαρμοστικότητα (Responsiveness):** Η εφαρμογή πρέπει να προσαρμόζεται δυναμικά στο μέγεθος της οθόνης της εκάστοτε συσκευής (desktop, tablet, smartphone), εξασφαλίζοντας απρόσκοπτη εμπειρία πλοήγησης από οποιοδήποτε μέσο.

Ε. Ελεγχιμότητα (Testability) και Τεκμηρίωση (Documentation)

- **Αυτοματοποιημένος Έλεγχος:** Το πληροφοριακό σύστημα πρέπει να είναι σχεδιασμένο με γνώμονα την ελεγκσιμότητα. Αυτό σημαίνει ότι τα βασικά δομικά του στοιχεία πρέπει να επιδέχονται μεμονωμένες δοκιμές ελέγχου (unit tests) για την επιβεβαίωση της ορθής λειτουργίας της επιχειρησιακής λογικής. Επιπρόσθετα, η ίδια η αρχιτεκτονική του συστήματος (οι εξαρτήσεις μεταξύ των κλάσεων και των πακέτων) οφείλει να ελέγχεται προγραμματιστικά μέσω εξειδικευμένων αρχιτεκτονικών δοκιμών (architectural tests, π.χ. με εργαλεία όπως το ArchUnit). Αυτό διασφαλίζει ότι οι περιορισμοί της Εξαγωνικής Αρχιτεκτονικής τηρούνται αυστηρά και αποτρέπεται η σταδιακή υποβάθμιση του σχεδιασμού (architectural decay).
- **Εσωτερική και Εξωτερική Τεκμηρίωση:** Η τεκμηρίωση του συστήματος οφείλει να καλύπτει δύο διακριτά επίπεδα. Αφενός, ο πηγαίος κώδικας πρέπει να περιλαμβάνει εκτενή και τυποποιημένα σχόλια (όπως τα πρότυπα Javadoc) για όλες τις δημόσιες διεπαφές και τις κλάσεις επιχειρησιακής λογικής, διευκολύνοντας τη συντήρηση και τη μελλοντική μεταβίβαση (handover) του έργου σε άλλους προγραμματιστές (π.χ. υπό το πλαίσιο περαιτέρω επαγγελματικής ανάπτυξης του). Αφετέρου, όπως αναφέρθηκε και στη διαλειτουργικότητα, είναι απολύτως αναγκαία η αυτοματοποιημένη τεκμηρίωση του εκτεθειμένου API (π.χ. μέσω Swagger / OpenAPI). Η τεκμηρίωση αυτή διασφαλίζει την κατανόηση και την ομαλή χρήση του πυρήνα του συστήματος από οποιοδήποτε εξωτερικό σύστημα ή πελάτη επιθυμεί να διαλειτουργήσει με αυτό.

3.3 Αρχιτεκτονικός και Εννοιολογικός Σχεδιασμός

Έχοντας καθορίσει τις λειτουργικές και μη λειτουργικές απαιτήσεις, το επόμενο στάδιο στη μεθοδολογία του Καταρράκτη αφορά τον εννοιολογικό και αρχιτεκτονικό σχεδιασμό του συστήματος. Στην παρούσα ενότητα, οι θεωρητικές απαιτήσεις μετασχηματίζονται σε οπτικά μοντέλα, αξιοποιώντας τα καθιερωμένα πρότυπα της Ενοποιημένης Γλώσσας Μοντελοποίησης (UML – Unified Modeling Language). Σύμφωνα με τη βιβλιογραφία, η UML προσφέρει ένα τυποποιημένο σύνολο διαγραμμάτων που επιτρέπουν την οπτικοποίηση, τον σχεδιασμό και την τεκμηρίωση τόσο της στατικής δομής όσο και της δυναμικής συμπεριφοράς ενός συστήματος λογισμικού (Bhatt & Nandu, 2021; Chonoles, 2017).

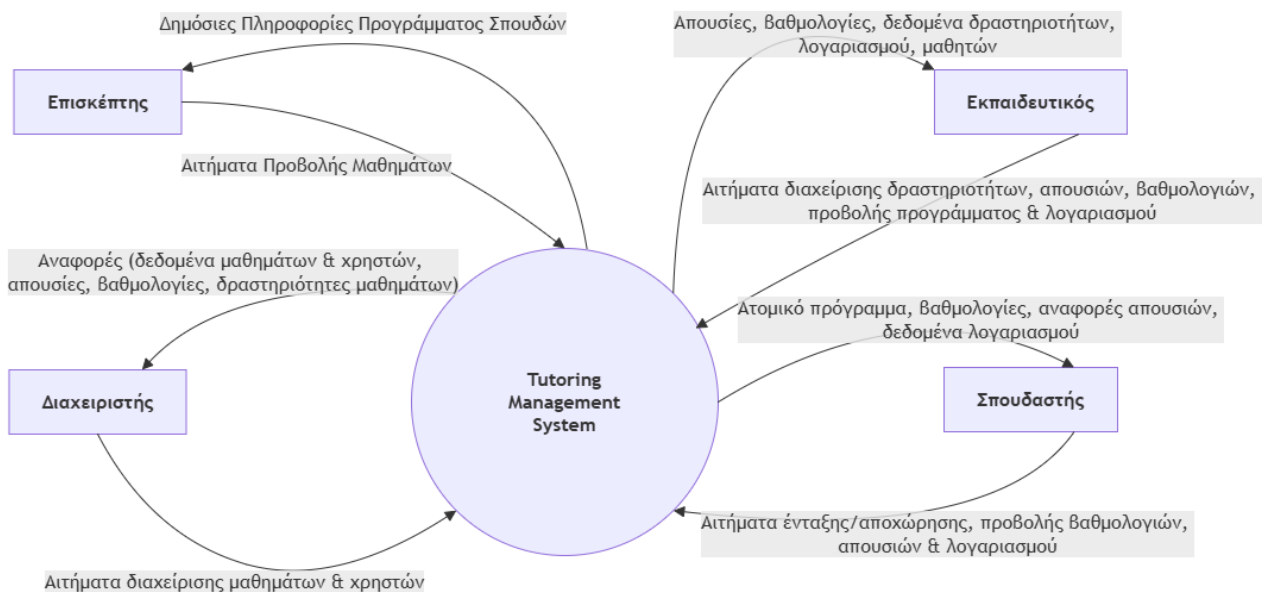
Ο σχεδιασμός αυτός αποτελεί το αυστηρό τεχνικό προσχέδιο (blueprint) επί του οποίου βασίστηκε η προγραμματιστική υλοποίηση της παρούσας εφαρμογής. Μέσα από μία ιεραρχική σειρά διαγραμμάτων – ξεκινώντας από την υψηλού επιπέδου αλληλεπίδραση (Διάγραμμα Πλαισίου) και προχωρώντας στις Περιπτώσεις Χρήσης και την εσωτερική δομή του κώδικα (Διάγραμμα Κλάσεων) – διασφαλίζεται η πλήρης ιχνηλασιμότητα των απαιτήσεων σε αρχιτεκτονικά συστατικά.

3.3.1 Διάγραμμα Περιβάλλοντος (Context Diagram)

Το Διάγραμμα Περιβάλλοντος (ή Μοντέλο Περιβάλλοντος) αποτελεί το υψηλότερο επίπεδο αφαίρεσης κατά τον σχεδιασμό ενός συστήματος. Στόχος του είναι να επιδείξει το λειτουργικό περιβάλλον της εφαρμογής, ορίζοντας με σαφήνεια τα όρια του συστήματος και αναδεικνύοντας τους εξωτερικούς δρώντες (actors) με τους οποίους αυτό αλληλοεπιδρά. Στο συγκεκριμένο μοντέλο, το πληροφοριακό σύστημα αντιμετωπίζεται ως μια ενιαία, αδιαίρετη οντότητα (απεικονιζόμενη ως κεντρικός κόμβος), χωρίς να αποκαλύπτεται η εσωτερική του δομή.

Όπως αποτυπώνεται στο παρακάτω σχήμα, το Πληροφοριακό Σύστημα Διαχείρισης Φροντιστηρίου αποτελεί ένα αυτόνομο, κλειστό οικοσύστημα, το οποίο δεν αντλεί ούτε αποστέλλει δεδομένα σε τρίτα, εξωτερικά συστήματα (π.χ. εξωτερικά APIs ή άλλες πλατφόρμες). Οι μοναδικοί

εξωτερικοί παράγοντες που αλληλοεπιδρούν με το λογισμικό είναι οι τέσσερις ανθρώπινοι ρόλοι: Διαχειριστής, Εκπαιδευτικός, Σπουδαστής και Επισκέπτης. Το παρόν διάγραμμα συνοψίζει την αμφίδρομη ροή πληροφορίας μεταξύ του συστήματος και των χρηστών, περιγράφοντας τα βασικά αιτήματα (εισόδους) και τις αντίστοιχες αποκρίσεις (εξόδους) ανά ρόλο. Συγκεκριμένα, ο Διαχειριστής αποστέλλει αιτήματα διαχείρισης μαθημάτων και χρηστών, λαμβάνοντας συγκεντρωτικές αναφορές, ο Εκπαιδευτικός διαχειρίζεται δραστηριότητες και αξιολογήσεις μαθητών, ο Σπουδαστής εκτελεί αιτήματα ένταξης και προβολής του ατομικού του προγράμματος και των επιδόσεών του, ενώ ο Επισκέπτης περιορίζεται στη λήψη δημόσιων πληροφοριών του προγράμματος σπουδών.



Σχήμα 1: Διάγραμμα Περιβάλλοντος (Context Diagram) του Πληροφοριακού Συστήματος

Η συγκεκριμένη απεικόνιση οριοθετεί την εμβέλεια του έργου και επιβεβαιώνει ότι όλες οι επιχειρησιακές λειτουργίες και η αποθήκευση δεδομένων συντελούνται αποκλειστικά εντός των συνόρων της εφαρμογής.

3.3.2 Αρχιτεκτονική Συστατικών (Component Diagram – Hexagonal Architecture)

Ενώ το διάγραμμα περιβάλλοντος καλύπτει τις αλληλεπιδράσεις ενός συστήματος ως μαύρο κουτί με εξωτερικούς δρώντες και συστήματα, το Διάγραμμα Συστατικών (Component Diagram) επιτρέπει την ιεραρχική αποσύνθεση ενός συστήματος, αναδεικνύοντας τα δομικά του μέρη υπό ένα υψηλότερο επίπεδο αφαίρεσης. Βασικός σκοπός του είναι η αποτύπωση των διασυνδέσεων και των εξαρτήσεων μεταξύ των ανεξάρτητων μονάδων (components) του λογισμικού, χρησιμοποιώντας σημασιολογικά τις έννοιες των προσφερόμενων (provided) και απαιτούμενων (required) διεπαφών (interfaces) (Ermel κ.ά., 2018).

Στην παρούσα διπλωματική εργασία, το Διάγραμμα Συστατικών αξιοποιείται για να οπτικοποιηθεί την εφαρμογή της Εξαγωνικής Αρχιτεκτονικής (Hexagonal Architecture). Η αρχιτεκτονική του συστήματος έχει υλοποιηθεί αυστηρά μέσω τεσσάρων (4) διακριτών ενοτήτων (Maven Modules), διασφαλίζοντας την αρχή της αντιστροφής εξαρτήσεων (Dependency Inversion). Όπως ορίζει η βιβλιογραφία, ο πυρήνας της επιχειρησιακής λογικής πρέπει να παραμένει απολύτως απομονωμένος από τις τεχνολογικές λεπτομέρειες των εξωτερικών συστημάτων (Vieira, 2022).

Στο Σχήμα 2 που ακολουθεί, η δομή του συστήματος απεικονίζεται ακριβώς όπως έχει κατασκευαστεί στον πηγαίο κώδικα:

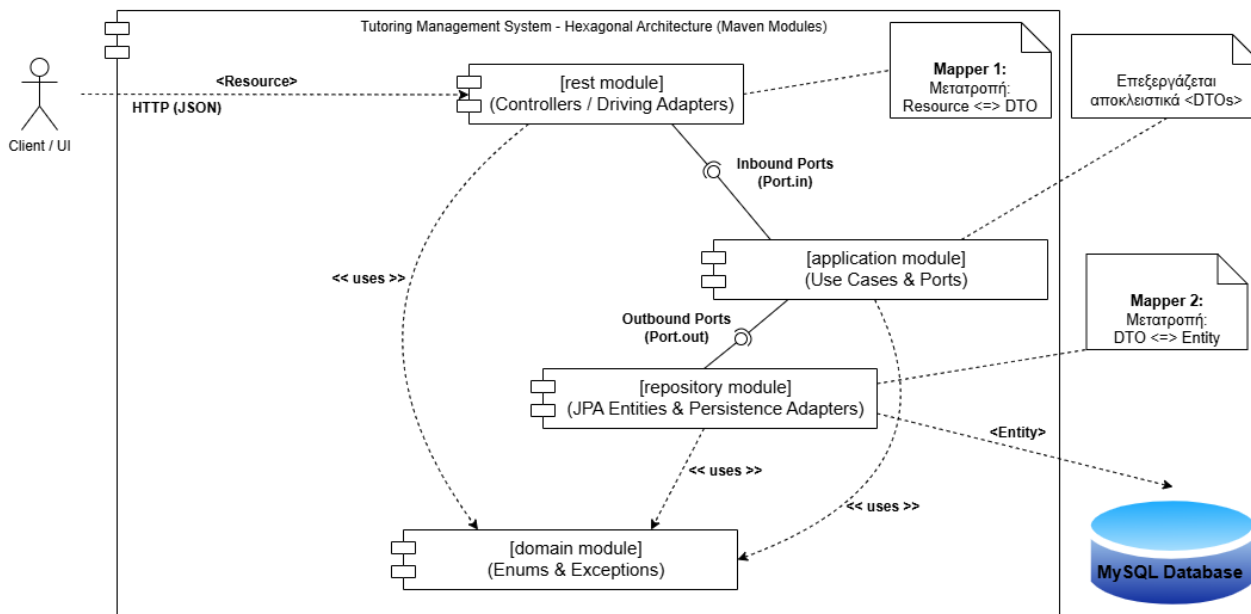
1. **REST Module (Driving Adapters):** Περιέχει τους ελεγκτές (Controllers) που υποδέχονται τα HTTP αιτήματα. Το συστατικό αυτό απαιτεί την εκτέλεση λειτουργιών από τον πυρήνα μέσω των Inbound Ports. Παράλληλα, φιλοξενεί τον κεντρικό μηχανισμό διαχείρισης σφαλμάτων (GlobalExceptionHandler), ο οποίος αναλαμβάνει να “μεταφράσει” τα εσωτερικά σφάλματα του πυρήνα στα κατάλληλα HTTP Status Codes, προστατεύοντας τον κώδικα από διαρροές (information leaks) προς τον τελικό χρήστη.
2. **Application Module (Use Cases & Ports):** Ο ενορχηστρωτής του συστήματος. Προσφέρει τις λειτουργίες (application.service) προς το REST module μέσω των *port.in* και ορίζει τις απαιτήσεις μέσω των *port.out*.
3. **Domain Module:** Η καρδιά του επιχειρησιακού τομέα. Περιλαμβάνει τους τύπους δεδομένων (όπως Enumerations), καθώς και την ιεραρχία των προσαρμοσμένων εξαιρέσεων της εφαρμογής (Custom Exceptions). Λόγω της ανεξαρτησίας του από εξωτερικά πλαίσια (frameworks), καταναλώνεται ελεύθερα ως εξάρτηση (dependency) από όλα τα υπόλοιπα υποσυστήματα.
4. **Repository Module (Driven Adapters):** Το επίπεδο υποδομής που περιέχει τα JPA Entities. Παρέχει την υλοποίηση (repository.jpa.service) των *port.out* που απαιτεί το Application Module.

Στρατηγική Μετασχηματισμού Δεδομένων (Double Mapping Strategy)

Ένα από τα σημαντικότερα αρχιτεκτονικά χαρακτηριστικά που αποτυπώνεται στο διάγραμμα είναι η αυστηρή απομόνωση των δεδομένων μέσω “διπλού μετασχηματισμού” (double mapping) κατά τη διέλευσή τους από τα όρια των συστατικών:

- **Όριο Εισόδου (Web Boundary):** Στο rest module, τα δεδομένα που ανταλλάσσονται με τον χρήστη ορίζονται ως Resources (JSON payloads). Οι ελεγκτές (Controllers) αναλαμβάνουν τη μετατροπή τους σε DTOs (Data Transfer Objects) πριν τα προωθήσουν στον πυρήνα.
- **Όριο Υποδομής (Persistence Boundary):** Ο πυρήνας (application module) επεξεργάζεται αποκλειστικά DTOs. Όταν το repository module παραλαμβάνει αυτά τα DTOs μέσω των Outbound Ports, αναλαμβάνει μια δεύτερη μετατροπή (mapping) από DTO σε JPA Entity, ώστε να αποθηκευτούν με ασφάλεια στη βάση δεδομένων.

Η αντίστροφη πορεία ακολουθείται κατά την ανάκτηση πληροφορίας. Αυτός ο μηχανισμός εγγυάται ότι η εσωτερική δομή της βάσης δεδομένων (Entities) δεν εκτίθεται ποτέ στο διαδίκτυο, προστατεύοντας το σύστημα από ευπάθειες τύπου “Mass Assignment” και επιτρέποντας την ανεξάρτητη εξέλιξη του API και της βάσης δεδομένων.



Σχήμα 2: Διάγραμμα Συστατικών & Ροή Μετασχηματισμού Δεδομένων (Hexagonal Architecture)

3.3.3 Διαγράμματα Περιπτώσεων Χρήσης (Use-Case Diagrams)

Το Διάγραμμα Περιπτώσεων Χρήσης της UML αποτελεί το βασικότερο εργαλείο αποτύπωσης των λειτουργικών απαιτήσεων του συστήματος, καθώς περιγράφει τη συμπεριφορά του λογισμικού από την οπτική γωνιά των χρηστών του (Actors) (Mule κ.ά., 2015). Σκοπός της συγκεκριμένης μοντελοποίησης δεν είναι η αναπαράσταση εσωτερικών διαδικασιών ή ροών εργασίας (workflows), αλλά η οπτικοποίηση της αξίας (value) που παρέχει το σύστημα σε κάθε δρώντα που αλληλοεπιδρά με αυτό.

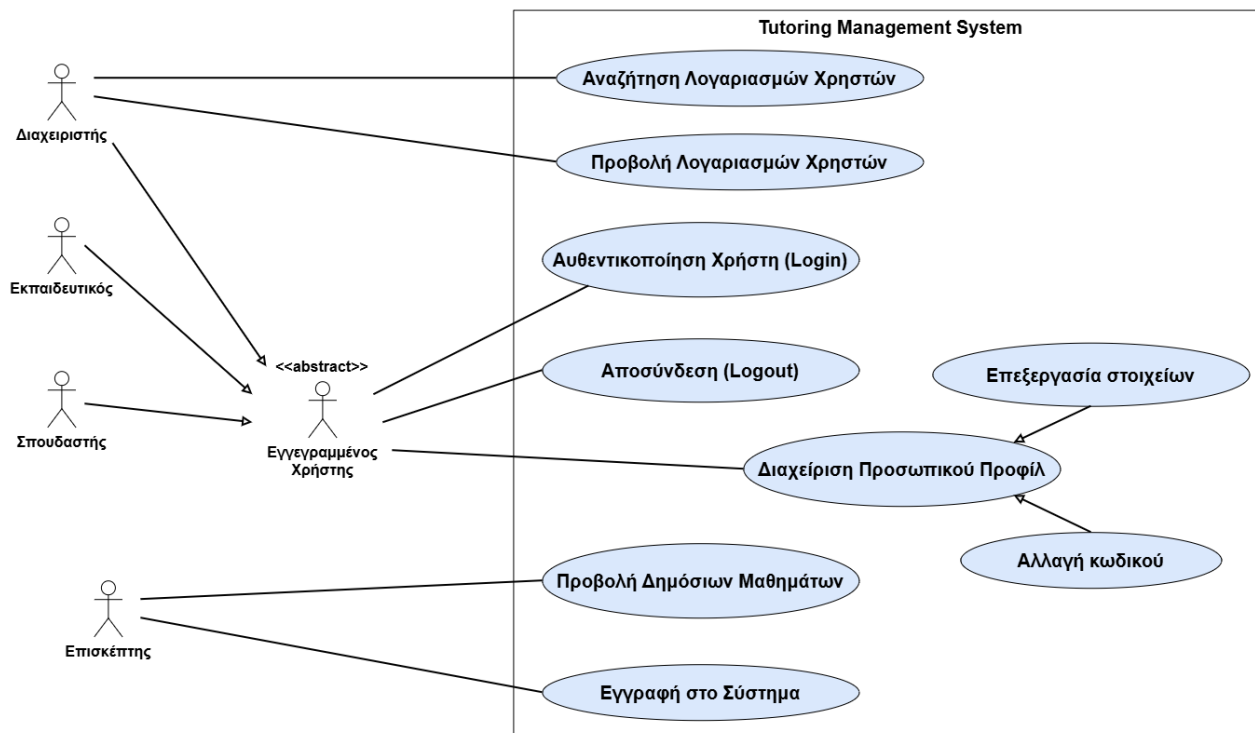
Λόγω του μεγάλου πλήθους των λειτουργικών απαιτήσεων που προδιαγράφηκαν στην Ενότητα 3.2, η μοντελοποίηση των περιπτώσεων χρήσης διαχωρίζεται σε τέσσερα (4) επιμέρους διαγράμματα, οργανωμένα κυρίως με γνώμονα τους διακριτούς ρόλους των χρηστών εκτός από το πρώτο, το οποίο καλύπτει τις κοινές λειτουργίες διαχείρισης λογαριασμού. Η συγκεκριμένη σχεδιαστική επιλογή αποτρέπει την οπτική συμφόρηση (visual clutter) ενός ενιαίου, υπερφορτωμένου διαγράμματος και διασφαλίζει την αυστηρή τήρηση των βέλτιστων πρακτικών της UML.

Διάγραμμα 1: Ταυτοποίηση και Κοινές Λειτουργίες Προφίλ

Το πρώτο διάγραμμα εστιάζει στις βασικές λειτουργίες ελέγχου πρόσβασης, καθώς και στη διαχείριση των λογαριασμών του συστήματος. Για την αποφυγή πλεονασματικών συσχετίσεων (redundant associations), εφαρμόστηκε η τεχνική της γενίκευσης δρώντων (actor generalization). Ειδικότερα, δημιουργήθηκε ένας αφηρημένος υπερ-δρών (abstract super-actor) με την ονομασία «Εγγεγραμμένος Χρήστης», τον οποίο κληρονομούν ο Διαχειριστής, ο Εκπαιδευτικός και ο Σπουδαστής. Έτσι, θεμελιώδεις λειτουργίες όπως η αυθεντικοποίηση (Login), η αποσύνδεση (Logout) και η διαχείριση του προσωπικού προφίλ συνδέονται απευθείας με τον υπερ-δρώντα.

Η λειτουργία της διαχείρισης προφίλ αναλύεται περαιτέρω, μέσω σχέσεων γενίκευσης (ISA), σε πιο εξειδικευμένες περιπτώσεις χρήσης, αποτυπώνοντας διακριτά την επεξεργασία των προσωπικών στοιχείων και την αλλαγή του κωδικού πρόσβασης. Επιπρόσθετα, στο παρόν διάγραμμα ομαδοποιούνται και οι λειτουργίες αναζήτησης και προβολής λογαριασμών, οι οποίες εκτελούνται αποκλειστικά από τον

Διαχειριστή. Τέλος, αποτυπώνεται ο ρόλος του Επισκέπτη (Guest), ο οποίος έχει τη δυνατότητα εγγραφής και προβολής του δημόσιου προγράμματος μαθημάτων.



Σχήμα 3: Διάγραμμα Περιπτώσεων Χρήσης - Αυθεντικοποίηση και Διαχείριση Λογαριασμών

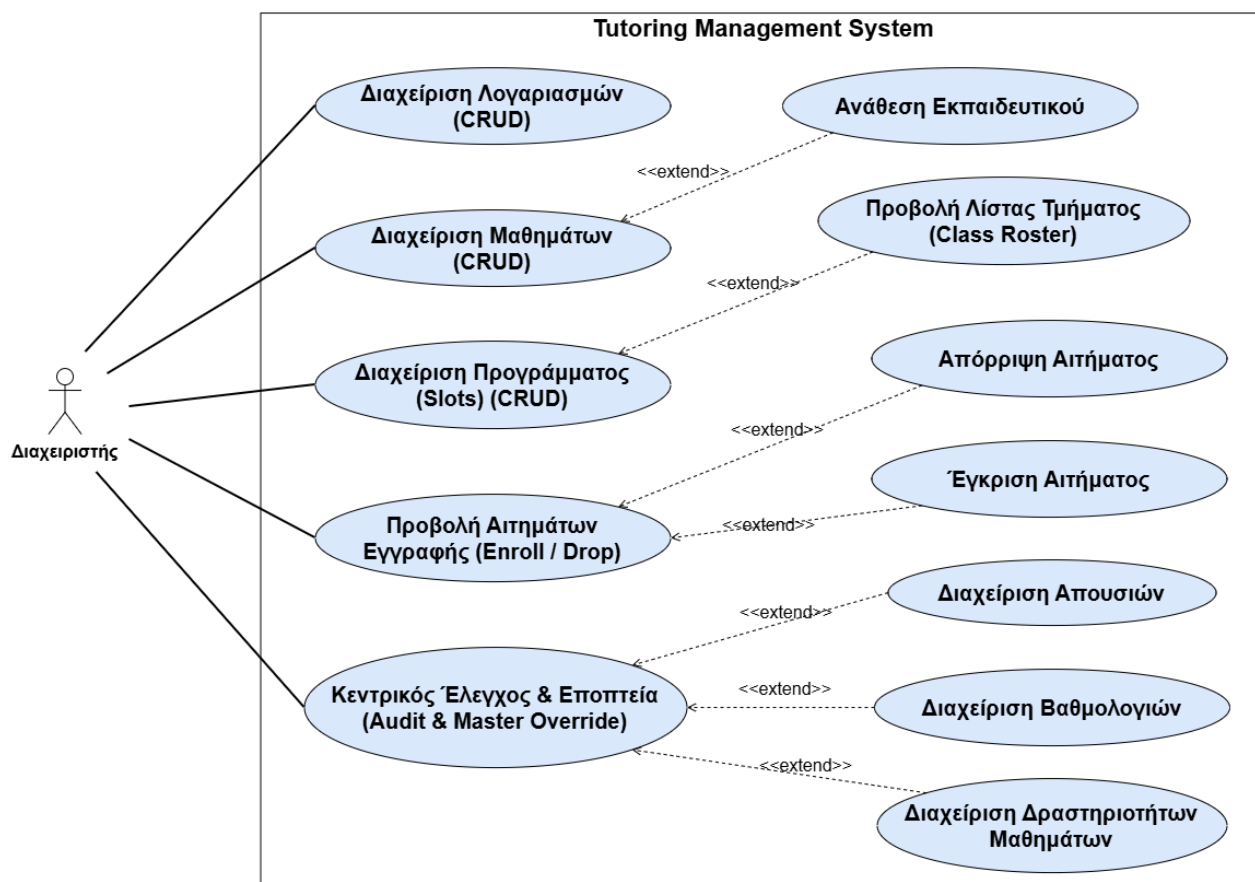
Η παραπάνω μοντελοποίηση αποτυπώνει τον θεμελιώδη μηχανισμό εισόδου και διαχείρισης χρηστών της εφαρμογής. Με την αυθεντικοποίηση να αποτελεί προαπαιτούμενο για τις περισσότερες ενέργειες, τα επόμενα διαγράμματα επικεντρώνονται στην εξειδικευμένη επιχειρησιακή λογική (business logic) που αναπτύσσεται στο εσωτερικό του συστήματος, ξεκινώντας από τα ανώτατα προνόμια του Διαχειριστή.

Διάγραμμα 2: Λειτουργίες Διαχειριστή (Admin Workflow)

Το δεύτερο διάγραμμα αποτυπώνει τις περιπτώσεις χρήσης του Διαχειριστή, ο οποίος διαθέτει τα ανώτατα προνόμια συστήματος (superuser). Ο ρόλος αυτός είναι επιφορτισμένος με τη δόμηση του ακαδημαϊκού πλαισίου και τον έλεγχο, εκτελώντας λειτουργίες πλήρους διαχείρισης (CRUD) σε Λογαριασμούς, Μαθήματα και στο Ωρολόγιο Πρόγραμμα (Slots). Ως ρυθμιστικός παράγοντας για την ομαλή διεξαγωγή των μαθημάτων, προβάλλει και αξιολογεί τα αιτήματα των σπουδαστών, εγκρίνοντας ή απορρίπτοντας τις εγγραφές (Enroll) και τις διαγραφές (Drop) από τα αντίστοιχα Τμήματα.

Για την πληρέστερη μοντελοποίηση της επιχειρησιακής λογικής, αξιοποιήθηκαν οι σχέσεις επέκτασης («extend»), οι οποίες αντικατοπτρίζουν άμεσα τη ροή στο περιβάλλον διεπαφής (UI) της εφαρμογής. Για παράδειγμα, η «Προβολή Λίστας Τμήματος» (Class Roster) αποτυπώνεται ως άμεση επέκταση της Διαχείρισης Προγράμματος, καθώς παρέχεται ως πρακτική συντόμευση: όταν ο Διαχειριστής επιλέγει ένα συγκεκριμένο τμήμα (slot) μέσα από το συνολικό ωρολόγιο πρόγραμμα, αποκτά άμεση πρόσβαση (εφόσον το επιθυμεί) στους εγγεγραμμένους σπουδαστές. Με παρόμοια λογική, η «Ανάθεση Εκπαιδευτικού» προκύπτει ως εξειδικευμένη ενέργεια μέσα από το περιβάλλον της Διαχείρισης Μαθημάτων. Παράλληλα, μέσα από την κεντρική διεπαφή Προβολής Αιτημάτων, ο Διαχειριστής έχει τη δυνατότητα να προχωρήσει στην τελική Έγκριση ή Απόρριψη ενός αιτήματος, ρυθμίζοντας ομαλά τη σύνθεση των τμημάτων.

Επιπρόσθετα, το σύστημα παρέχει στον Διαχειριστή κεντρικό έλεγχο και δικαιώματα εποπτείας (Audit). Μέσω των υπερεξουσιών (Master Overrides), δύναται να επεμβαίνει διορθωτικά, παρακάμπτοντας τους αυστηρούς χρονικούς περιορισμούς που επιβάλλονται στους υπόλοιπους χρήστες από την επιχειρησιακή λογική του συστήματος. Χαρακτηριστικό παράδειγμα αποτελεί η καταγραφή δραστηριοτήτων μαθήματος (Lesson Activities): ενώ ο Εκπαιδευτικός περιορίζεται στο να καταχωρεί ή να τροποποιεί δραστηριότητες μόνο σε ένα χρονικό παράθυρο από 14 ημέρες στο παρελθόν έως 30 ημέρες στο μέλλον (ώστε να διασφαλίζεται η τήρηση του τρέχοντος ωρολογίου προγράμματος), ο Διαχειριστής διατηρεί το δικαίωμα της κεντρικής διαχείρισης και τροποποίησης αυτών των εγγγραφών ανά πάσα στιγμή. Ο ίδιος μηχανισμός κεντρικού ελέγχου και παράκαμψης κανόνων εφαρμόζεται αντίστοιχα στη Διαχείριση Απουσιών και στη Διαχείριση Βαθμολογιών (Tests).



Σχήμα 4: Διάγραμμα Περιπτώσεων Χρήσης - Λειτουργίες Διαχειριστή

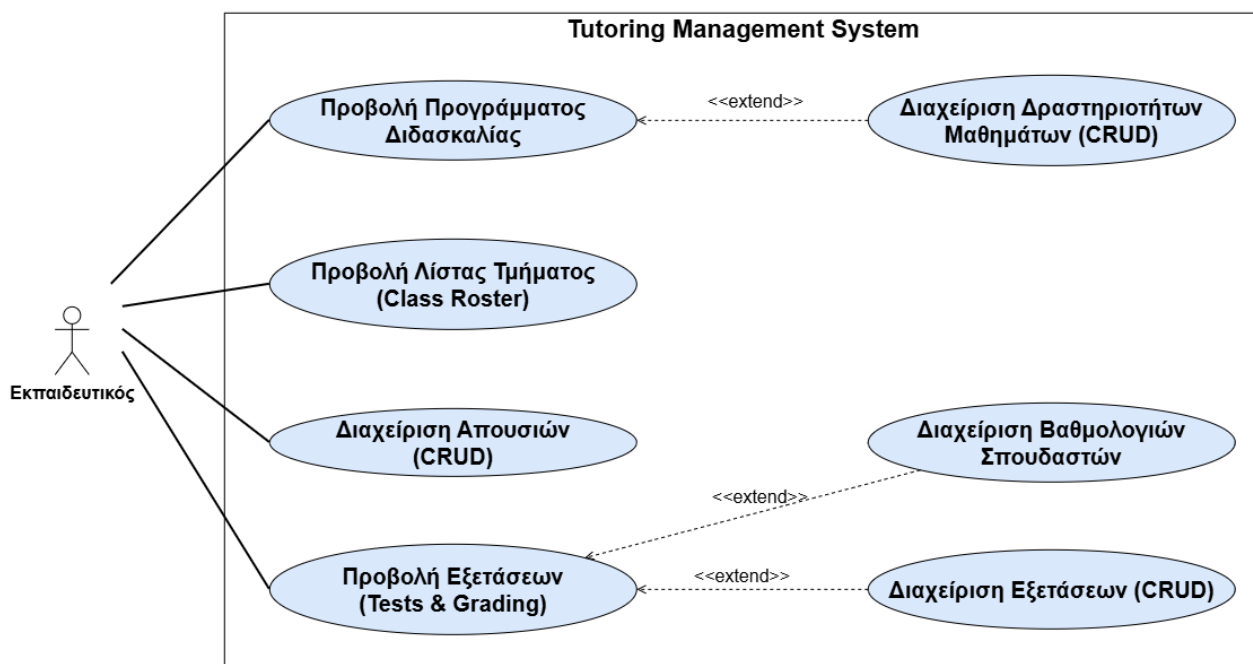
Όπως καθίσταται σαφές από την παραπάνω μοντελοποίηση, ο Διαχειριστής λειτουργεί κυρίως ως ο θεματοφύλακας της πλατφόρμας, διαμορφώνοντας την απαραίτητη εκπαιδευτική και οργανωτική υποδομή. Ωστόσο, το σύστημα αποκτά την πραγματική του αξία κατά την καθημερινή χρήση του στην εκπαιδευτική διαδικασία. Με το δομικό υπόβαθρο (Μαθήματα, Τμήματα, Χρήστες) να έχει ήδη εξασφαλιστεί, η παραγωγή του εκπαιδευτικού περιεχομένου και η άμεση αλληλεπίδραση με τους μαθητές περνάει στη δικαιοδοσία του Εκπαιδευτικού, οι λειτουργίες του οποίου αναλύονται στο επόμενο διάγραμμα.

Διάγραμμα 3: Λειτουργίες Εκπαιδευτικού (Teacher Workflow)

Το τρίτο διάγραμμα επικεντρώνεται στον Εκπαιδευτικό, τον δρώντα που τροφοδοτεί το σύστημα με τα πρωτογενή ακαδημαϊκά δεδομένα της καθημερινότητας. Η λειτουργικότητα του Εκπαιδευτικού

οργανώνεται σε τρεις κεντρικούς πυλώνες: τη διαχείριση της διδασκαλίας (προβολή προγράμματος και λίστας μαθητών), τη διαχείριση της διαθεσιμότητάς του (διαχείριση απουσιών) και την καταγραφή της ακαδημαϊκής πορείας (δραστηριότητες μαθημάτων, διαχείριση εξετάσεων και βαθμολόγηση σπουδαστών).

Όπως αποτυπώνεται στο μοντέλο, ο σχεδιασμός του περιβάλλοντος εργασίας (UI) του Εκπαιδευτικού ενσωματώνει σχέσεις επέκτασης «extend». Ειδικότερα, η κεντρική ενέργεια της προβολής των εξετάσεων (Tests & Grading) λειτουργεί ως σημείο επέκτασης. Μέσα από αυτή τη βασική ροή, ο Εκπαιδευτικός έχει την επιλογή είτε να διαχειριστεί πλήρως (δημιουργία, τροποποίηση, διαγραφή) τις εξετάσεις για ένα επιλεγμένο τμήμα, είτε να μεταβεί στο περιβάλλον αξιολόγησης προκειμένου να καταχωρήσει βαθμολογίες και σχόλια (feedback) στις επιδόσεις των σπουδαστών. Αντίστοιχη διαχείριση (τύπου CRUD) παρέχεται και στις δραστηριότητες μαθημάτων (Lesson Activities), διασφαλίζοντας την πλήρη ιχνηλασιμότητα της διδακτικής ύλης.



Σχήμα 5: Διάγραμμα Περιπτώσεων Χρήσης - Λειτουργίες Εκπαιδευτικού

Όπως καθίσταται σαφές από την παραπάνω μοντελοποίηση, ο Εκπαιδευτικός αποτελεί τον κεντρικό πυλώνα της εκπαιδευτικής διαδικασίας, έχοντας τον έλεγχο της διδακτέας ύλης, της αξιολόγησης των μαθητών και της διαθεσιμότητάς του. Με τις λειτουργίες του Διαχειριστή να έχουν διαμορφώσει το πλαίσιο και αυτές του Εκπαιδευτικού να το έχουν εμπλουτίσει με περιεχόμενο, το σύστημα ολοκληρώνεται με τον τελικό αποδέκτη: τον Σπουδαστή. Οι περιπτώσεις χρήσης που αφορούν τον Σπουδαστή, οι οποίες επικεντρώνονται στην πρόσβαση στην πληροφορία και τη διαχείριση της ακαδημαϊκής του πορείας, αναλύονται στο επόμενο διάγραμμα.

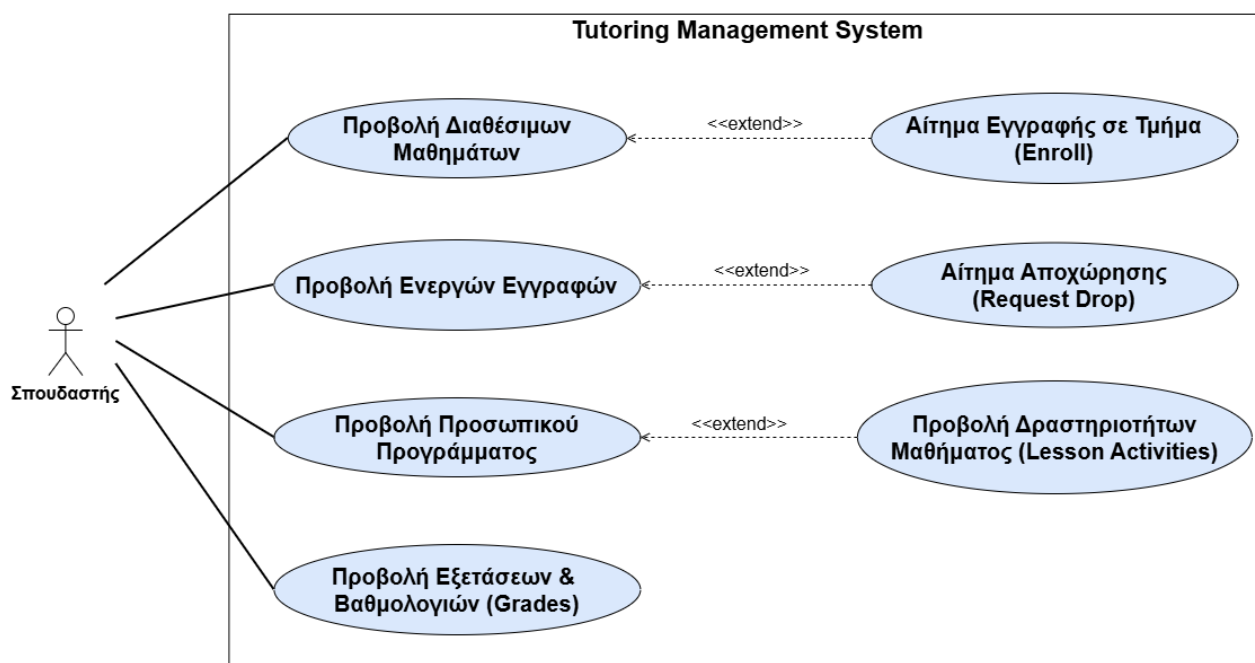
Διάγραμμα 4: Λειτουργίες Σπουδαστή (Student Workflow)

Το τέταρτο και τελευταίο διάγραμμα αποτυπώνει τις λειτουργίες του Σπουδαστή, ο οποίος αποτελεί τον τελικό αποδέκτη (end-user) της εκπαιδευτικής υπηρεσίας. Η μοντελοποίηση των περιπτώσεων χρήσης του σπουδαστή έχει σχεδιαστεί με γνώμονα την εύκολη πρόσβαση στην πληροφορία και την αυτοματοποίηση των ακαδημαϊκών του αιτημάτων.

Όπως παρατηρείται στο διάγραμμα, οι κύριες αλληλεπιδράσεις του Σπουδαστή αφορούν την προβολή των δεδομένων του (Προβολή Διαθέσιμων Μαθημάτων, Ενεργών Εγγραφών, Προσωπικού

Προγράμματος και Εξετάσεων). Η δυναμική φύση του συστήματος αναδεικνύεται μέσα από τη χρήση των σημείων επέκτασης («extend»). Συγκεκριμένα, η υποβολή ενός Αιτήματος Εγγραφής σε Τμήμα (Enroll) προκύπτει ως προαιρετική επέκταση της διαδικασίας περιήγησης και προβολής των διαθέσιμων μαθημάτων. Αντίστοιχα, μέσα από το πάνελ των ενεργών εγγραφών του, ο σπουδαστής δύναται να εκκινήσει τη διαδικασία διαγραφής του, υποβάλλοντας αίτημα αποχώρησης (Request Drop), το οποίο προωθείται στον Διαχειριστή για τελική έγκριση.

Επιπλέον, η περιήγηση στο εβδομαδιαίο πρόγραμμα ενσωματώνει σύνθετη επιχειρησιακή λογική, καθώς το σύστημα συγχωνεύει τα σταθερά προγραμματισμένα τμήματα με τις δηλωμένες απουσίες των εκπαιδευτικών, προβάλλοντας σε πραγματικό χρόνο την ανάλογη κατάσταση της κάθε διδακτικής ώρας («Scheduled» ή «Cancelled»). Παράλληλα, η συγκεκριμένη περίπτωση χρήσης παρέχει την προαιρετική δυνατότητα προβολής των δραστηριοτήτων (Lesson Activities) για κάθε επιλεγμένο μάθημα. Τέλος, η προβολή των εξετάσεων και των βαθμολογιών αποτελεί μια διακριτή, ανεξάρτητη περίπτωση χρήσης, διαχωρίζοντας την επίσημη αξιολόγηση από τις καθημερινές δραστηριότητες.



Σχήμα 6: Διάγραμμα Περιπτώσεων Χρήσης - Λειτουργίες Σπουδαστή

Με την ολοκλήρωση και του τέταρτου διαγράμματος, αποτυπώνεται πλήρως ο κύκλος λειτουργίας του πληροφοριακού συστήματος από την πλευρά των τελικών χρηστών. Η κατάτμηση των περιπτώσεων χρήσης σε διακριτά μοντέλα ανά ρόλο επέτρεψε την αναλυτική χαρτογράφηση των δικαιωμάτων και των επιχειρησιακών ορίων κάθε δρώντα, εξασφαλίζοντας παράλληλα τη συμμόρφωση με τις βέλτιστες πρακτικές της UML και αποφεύγοντας την οπτική συμφόρηση. Καθώς ο εννοιολογικός σχεδιασμός των λειτουργικών απαιτήσεων έχει πλέον εδραιωθεί, το επόμενο κρίσιμο βήμα για τη γεφύρωση της ανάλυσης με την αρχιτεκτονική υλοποίηση είναι η εξέταση της δυναμικής συμπεριφοράς του συστήματος. Αυτή η διάσταση πραγματώνεται στην επόμενη ενότητα, αρχικά μέσω των Διαγραμμάτων Δραστηριοτήτων που αποτυπώνουν τη ροή των επιχειρησιακών διαδικασιών (στο πλαίσιο μιας περίπτωσης χρήσης/λειτουργίας του συστήματος), και εν συνεχεία μέσω των Διαγραμμάτων Ακολουθίας, τα οποία εξειδικεύουν τη χρονική αλληλεπίδραση μεταξύ των συστατικών του λογισμικού (πάλι υπό το ίδιο πλαίσιο).

3.3.4 Διάγραμμα Δραστηριοτήτων / Ακολουθίας (Activity / Sequence Diagrams)

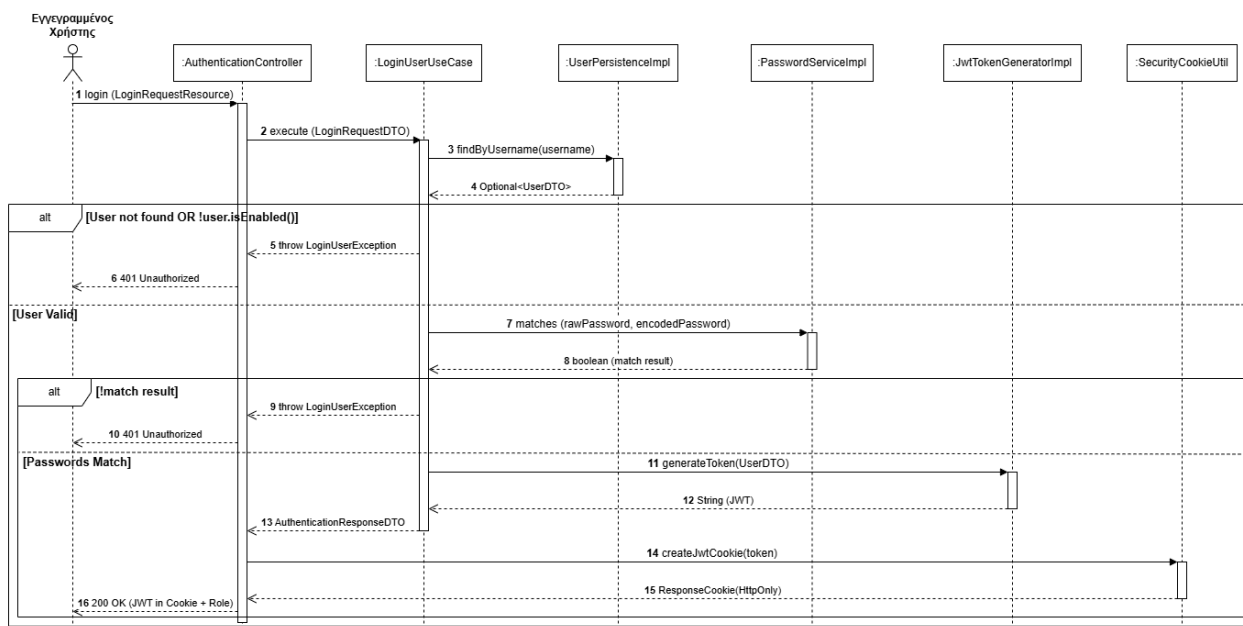
Η στατική και εννοιολογική αποτύπωση των απαιτήσεων μέσω των διαγραμμάτων περιπτώσεων χρήσης καθορίζει τα όρια του συστήματος, ωστόσο δεν επαρκεί για την πλήρη κατανόηση της εσωτερικής λειτουργίας του. Για την ολιστική σχεδίαση του λογισμικού, κρίνεται επιβεβλημένη η μετάβαση στη δυναμική μοντελοποίηση (dynamic modeling), η οποία περιγράφει τη συμπεριφορά του συστήματος κατά τον χρόνο εκτέλεσης (runtime). Στο παρόν υποκεφάλαιο, η δυναμική συμπεριφορά αναλύεται μέσα από δύο συμπληρωματικά πρίσματα.

Αρχικά, αξιοποιούνται τα Διαγράμματα Δραστηριοτήτων (Activity Diagrams) για την οπτικοποίηση των αλγοριθμικών βημάτων και της επιχειρησιακής ροής, παρέχοντας μια ξεκάθαρη αποτύπωση των συνθηκών ελέγχου και του παραλληλισμού των διεργασιών (Siewe & Ngounou, 2025). Εν συνεχεία, η αρχιτεκτονική αλληλεπίδραση των συστατικών μερών του συστήματος μοντελοποιείται μέσω των Διαγραμμάτων Ακολουθίας (Sequence Diagrams). Η συγκεκριμένη προσέγγιση επιτρέπει την αυστηρή, χρονολογική καταγραφή της ανταλλαγής μηνυμάτων μεταξύ των αντικειμένων, διασφαλίζοντας την ορθότητα της επικοινωνίας στα διαφορετικά επίπεδα της εφαρμογής (Han κ.ά., 2016).

Διάγραμμα 1: Αυθεντικοποίηση Χρήστη (Sequence Diagram)

Το πρώτο διάγραμμα ακολουθίας (Sequence Diagram) αποτυπώνει την κρίσιμη λειτουργία της αυθεντικοποίησης χρήστη (Login). Επιλέχθηκε η συγκεκριμένη μοντελοποίηση διότι αναδεικνύει την αλληλεπίδραση πολλαπλών αρχιτεκτονικών επιπέδων (Hexagonal Architecture) (Vieira, 2022) για την ολοκλήρωση μιας συναλλαγής.

Όπως φαίνεται στο γράφημα, η ροή εκκινεί από το UI, το οποίο αποστέλλει τα διαπιστευτήρια στον `AuthenticationController` (Inbound Adapter). Ο Controller αναθέτει την επιχειρησιακή λογική στο `LoginUserUseCase`. Εκεί πραγματοποιούνται οι διαδοχικοί έλεγχοι: ανάκτηση του χρήστη μέσω του `UserPersistence` (Outbound Port), επαλήθευση κωδικού μέσω του `PasswordService`, και παραγωγή του JWT μέσω του `JwtTokenGenerator`. Η χρήση πλαισίων ελέγχου (alt fragments) καταδεικνύει τη συμπεριφορά του συστήματος σε περίπτωση αποτυχίας (π.χ. ανενεργός λογαριασμός ή λανθασμένος κωδικός), όπου εγείρεται η αντίστοιχη εξαίρεση. Τέλος, ο Controller αξιοποιεί το `SecurityCookieUtil` για την ασφαλή τοποθέτηση του JWT σε HTTP-Only cookie, ολοκληρώνοντας τον κύκλο της αυθεντικοποίησης.



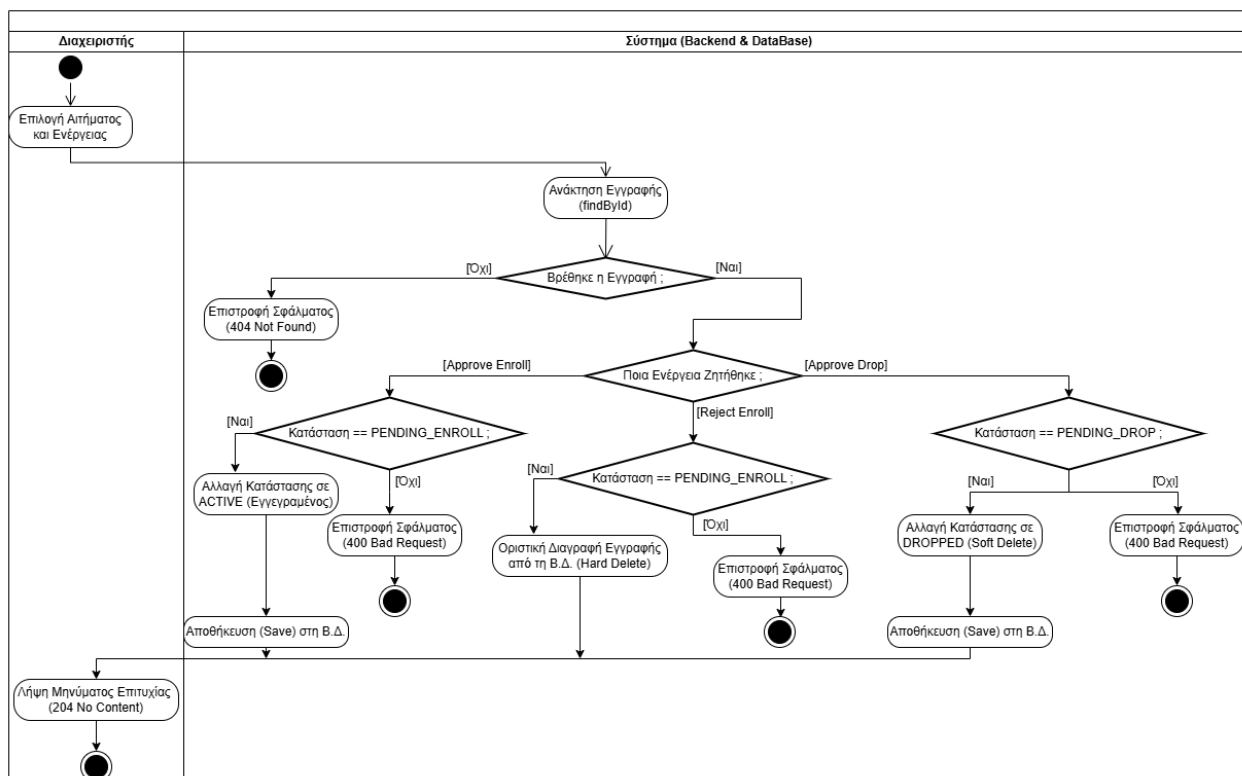
Σχήμα 7: Διάγραμμα Ακολουθίας - Αυθεντικοποίηση Χρήστη (Login)

Η επιτυχής αυθεντικοποίηση του χρήστη διασφαλίζει την πρόσβαση στο σύστημα, ανοίγοντας τον δρόμο για την εκτέλεση των εξειδικευμένων περιπτώσεων χρήσης ανά ρόλο. Μία από τις πλέον σύνθετες επιχειρησιακές διαδικασίες που ακολουθούν, αφορά τον Διαχειριστή (Admin) και τη διαχείριση των αιτημάτων εγγραφής (Enrollment Requests) των σπουδαστών. Για τη βέλτιστη αποτύπωση αυτής της ροής, η οποία περιλαμβάνει λογικές διακλαδώσεις, ελέγχους εγκυρότητας και τελικές αποφάσεις έγκρισης ή απόρριψης, η δυναμική ανάλυση μεταβαίνει από την αρχιτεκτονική ακολουθία (Sequence Diagram) στη μοντελοποίηση ροής εργασιών μέσω ενός Διαγράμματος Δραστηριοτήτων (Activity Diagram).

Διάγραμμα 2: Διαχείριση Αιτημάτων Εγγραφής (Activity Diagram)

Το δεύτερο διάγραμμα αποτελεί ένα Διάγραμμα Δραστηριοτήτων (Activity Diagram), το οποίο επιλέχθηκε ειδικά για να οπτικοποιήσει τη ροή εργασιών (workflow) του Διαχειριστή κατά την επεξεργασία των αιτημάτων εγγραφής (Approve Enroll, Reject Enroll, Approve Drop, Reject Drop). Σε αντίθεση με τη γραμμική ακολουθία, η συγκεκριμένη περίπτωση χρήσης εμπεριέχει πολύπλοκη επιχειρησιακή λογική που βασίζεται σε μηχανή καταστάσεων (State Machine) και πολλαπλές συνθήκες ελέγχου.

Το διάγραμμα διαχωρίζεται σε δύο περιοχές ευθύνης (Swimlanes): τις ενέργειες του Διαχειριστή και τις διεργασίες του Συστήματος. Αποτυπώνεται ξεκάθαρα πώς το σύστημα, για κάθε ξεχωριστό αίτημα (API Endpoint), ανακτά την εγγραφή από τη βάση δεδομένων και εκτελεί αυστηρούς ελέγχους κατάστασης (Status Checks). Για παράδειγμα, η έγκριση μιας εγγραφής απαιτεί η τρέχουσα κατάσταση να είναι PENDING_ENROLL, ενώ η έγκριση μιας αποχώρησης προϋποθέτει κατάσταση PENDING_DROP. Η χρήση των ρόμβων απόφασης (Decision Nodes) καταδεικνύει την ικανότητα του συστήματος να διακλαδώνει τη ροή, είτε ολοκληρώνοντας επιτυχώς τη συναλλαγή (αλλαγή κατάστασης ή διαγραφή), είτε διακόπτοντάς την με την επιστροφή των κατάλληλων σφαλμάτων (404 Not Found ή 400 Bad Request) σε περίπτωση παραβίασης των επιχειρησιακών κανόνων.



Σχήμα 8: Διάγραμμα Δραστηριοτήτων - Διαχείριση Αιτημάτων Εγγραφής

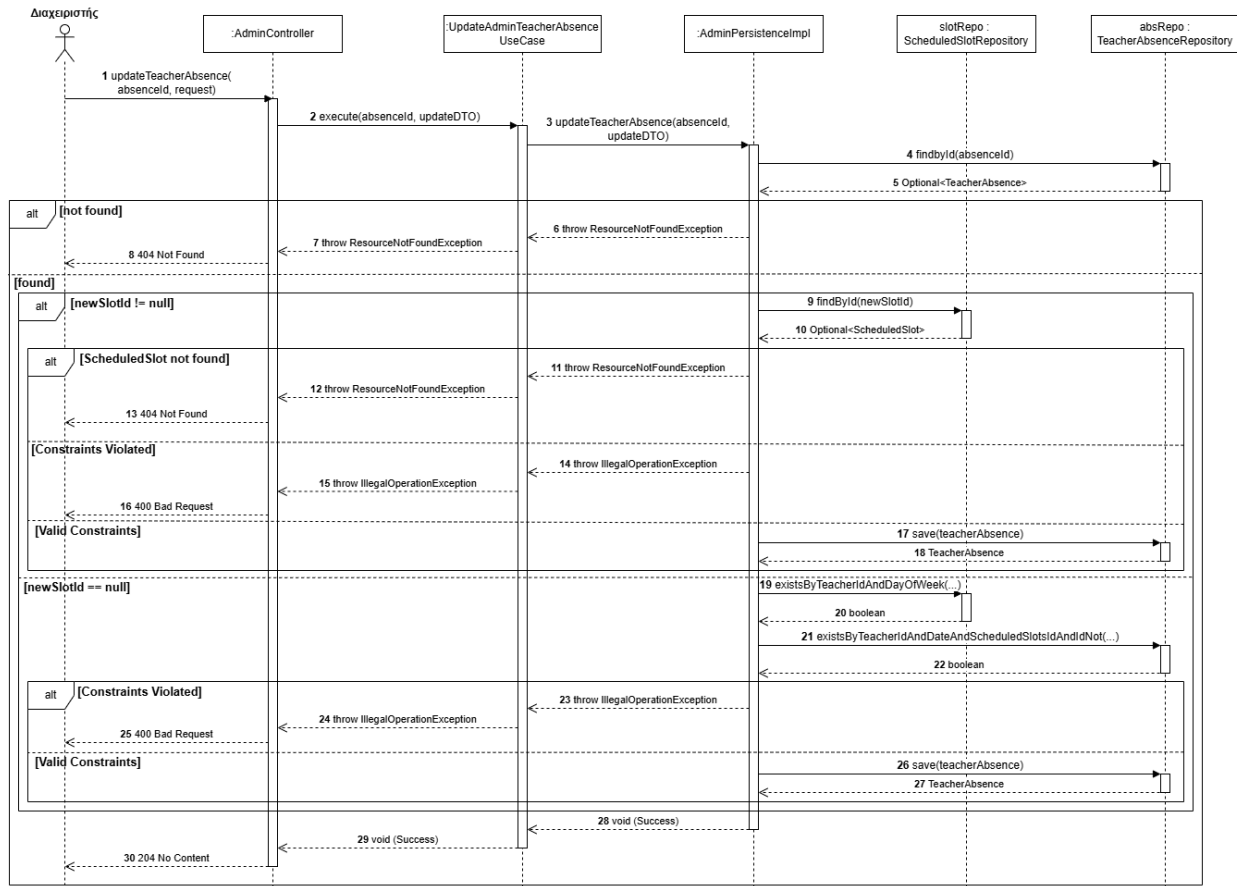
Η αποτύπωση της διαχείρισης των αιτημάτων μέσω του Διαγράμματος Δραστηριοτήτων επιβεβαιώνει ότι το σύστημα προστατεύει την ακεραιότητα των δεδομένων του, μην επιτρέποντας αυθαίρετες μεταβάσεις καταστάσεων (state transitions) χωρίς την εκπλήρωση των απαραίτητων προϋποθέσεων (Guard Conditions). Εντούτοις, σε ένα ρεαλιστικό περιβάλλον διαχείρισης, προκύπτουν έκτακτες περιπτώσεις (edge cases) όπου ο Διαχειριστής οφείλει να παρακάμψει την τυπική ροή για τη διόρθωση σφαλμάτων ή την άμεση παρέμβαση σε δεδομένα. Αυτές οι “διορθωτικές” κινήσεις ανήκουν στην κατηγορία του Κεντρικού Ελέγχου. Για την αποτύπωση αυτών των κρίσιμων και συχνά τεχνικά περίπλοκων συναλλαγών (Master Overrides), η σχεδίαση επιστρέφει στο μοντέλο των Διαγραμμάτων Ακολουθίας, προκειμένου να χαρτογραφηθεί με ακρίβεια η βαθιά αλληλεπίδραση μεταξύ των επιπέδων της εφαρμογής και της βάσης δεδομένων, όπως θα αναλυθεί στο αμέσως επόμενο διάγραμμα.

Διάγραμμα 3: Κεντρικός Έλεγχος / Παρέμβαση – Master Overrides (Sequence Diagram)

Το τρίτο διάγραμμα αποτυπώνει την αρχιτεκτονική λειτουργία του Κεντρικού Ελέγχου (Master Overrides). Ως αντιπροσωπευτικότερο παράδειγμα επιλέχθηκε η ενημέρωση / παρέμβαση σε μια καταχωρημένη Απουσία Εκπαιδευτικού (Teacher Absence) από τον Διαχειριστή. Σε αντίθεση με την τυπική ροή ενός εκπαιδευτικού, ο Διαχειριστής έχει την υπερεξουσία να παρακάμπτει χρονικούς περιορισμούς, ωστόσο το σύστημα οφείλει να διασφαλίσει τη δομική ακεραιότητα των δεδομένων στη βάση.

Για την αποτύπωση αυτής της πολύπλοκης συναλλαγής, επιλέχθηκε εκ νέου το Διάγραμμα Ακολουθίας (Sequence Diagram). Η σχεδίαση αναδεικνύει την ισχυρή αποσύνδεση της Εξαγωνικής Αρχιτεκτονικής, όπου ο `AdminController` και το `UpdateAdminTeacherAbsenceUseCase` λειτουργούν ως αγωγοί (pass-through), αναθέτοντας το βάρος των ελέγχων στο επίπεδο υποδομής (`AdminPersistenceImpl`). Όπως απεικονίζεται, το σύστημα εφαρμόζει υβριδική λογική (Hybrid Logic):

ελέγχει δυναμικά αν η παρέμβαση αφορά συγκεκριμένο μάθημα ή ολόημερη απουσία, και εκτελεί τα αντίστοιχα ερωτήματα στα Repositories (ScheduledSlotRepository και TeacherAbsenceRepository). Μέσω των πολλαπλών διακλαδώσεων ελέγχου (alt fragments), αποδεικνύεται η ικανότητα του συστήματος να αναχαιτίζει ασύμβατες ενέργειες (όπως η καταχώρηση ολόημερης απουσίας σε ημέρα που ο εκπαιδευτικός δεν διδάσκει) επιστρέφοντας δομημένα σφάλματα (400 Bad Request), εξασφαλίζοντας έτσι την απόλυτη συνέπεια της βάσης δεδομένων.



Σχήμα 9: Διάγραμμα Ακολουθίας - Κεντρικός Έλεγχος Απουσιών (Master Override)

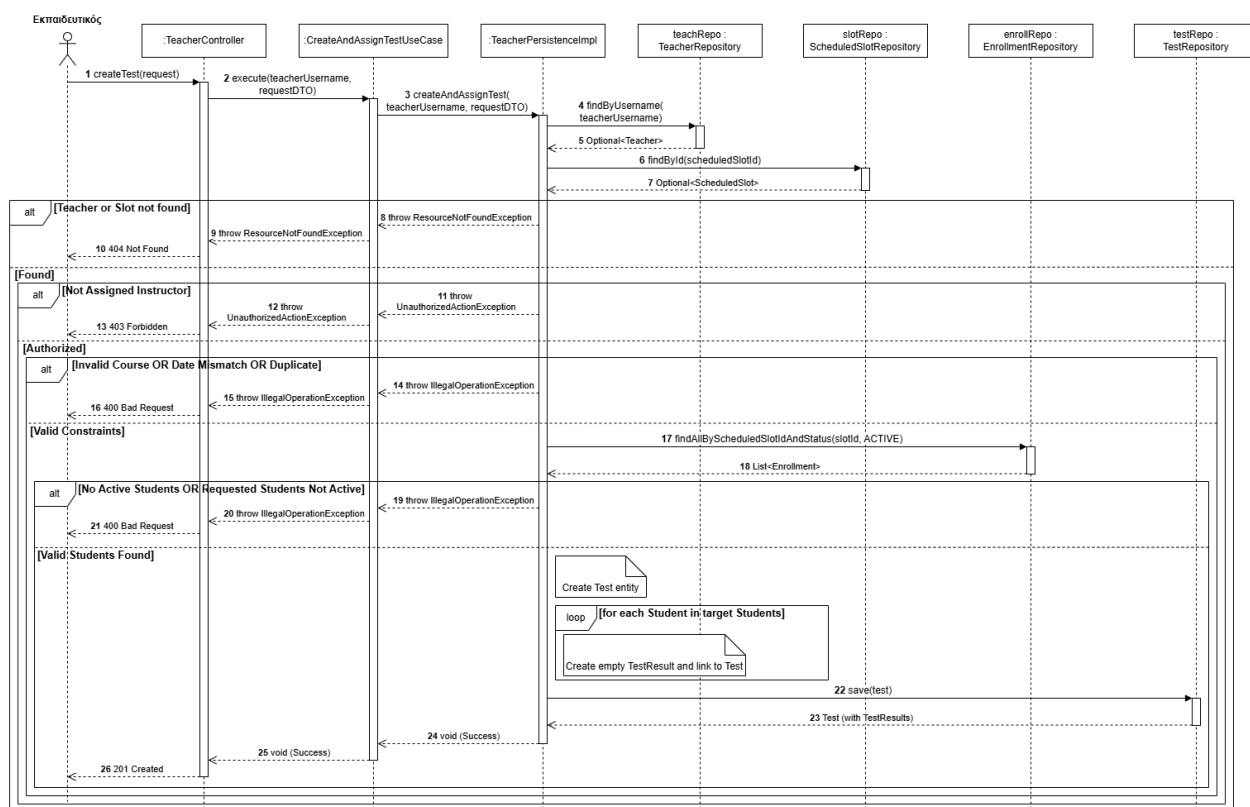
Η χαρτογράφηση του Κεντρικού Ελέγχου επιβεβαιώνει την ευρωστία του συστήματος, αποδεικνύοντας ότι η υπερεξουσία του Διαχειριστή δεν ταυτίζεται με την παράκαμψη των δόκιμων κανόνων του συστήματος, αλλά με την ελεγχόμενη ευελιξία του. Με την ολοκλήρωση της ανάλυσης των λειτουργιών του Διαχειριστή, το κέντρο βάρους της μελέτης μετατοπίζεται στον πυρήνα της εκπαιδευτικής διαδικασίας: τον Εκπαιδευτικό (Teacher). Ο Εκπαιδευτικός αποτελεί τον δημιουργό του ακαδημαϊκού περιεχομένου και φέρει την ευθύνη για την οργάνωση των αξιολογήσεων. Συνεπώς, η επόμενη επιχειρησιακή ροή που αναλύεται είναι η “Δημιουργία Εξέτασης” (Create Test), η οποία αποτελεί μια κρίσιμη συναλλαγή που συνδέει άμεσα τον Εκπαιδευτικό με τους εγγεγραμμένους Σπουδαστές του τμήματος.

Διάγραμμα 4: Δημιουργία και Ανάθεση Εξέτασης – Create Test (Sequence Diagram)

Μεταβαίνοντας στον ρόλο του Εκπαιδευτικού (Teacher), αναλύεται μία από τις σημαντικότερες ακαδημαϊκές ροές του συστήματος: η δημιουργία μιας νέας εξέτασης και η δυναμική ανάθεσή της στους σπουδαστές του τμήματος (Create and Assign Test). Αυτή η περίπτωση χρήσης αποτελεί εξαιρετικό παράδειγμα μιας συναλλαγής “1-προς-N” (από έναν εκπαιδευτικό προς πολλούς σπουδαστές) και

αναδεικνύει την ικανότητα του συστήματος να διασφαλίζει αυστηρά κριτήρια εξουσιοδότησης (Authorization) και ακεραιότητας δεδομένων.

Επιλέχθηκε το Διάγραμμα Ακολουθίας (Sequence Diagram) προκειμένου να αποτυπωθεί η εκτενής αλληλεπίδραση του επιπέδου διαχείρισης δεδομένων (TeacherPersistenceImpl) με πολλαπλά Repositories. Όπως διακρίνεται στο γράφημα, η ροή εκτέλεσης διέρχεται από διαδοχικά στάδια ελέγχου (alt fragments). Αρχικά, επιβεβαιώνεται η πρόσβαση (UnauthorizedActionException εάν ο εκπαιδευτικός δεν διδάσκει στο τμήμα) και έπειτα ελέγχεται η δομική ορθότητα των δεδομένων, όπως η ταύτιση της ημέρας διεξαγωγής του μαθήματος με την ημερομηνία της εξέτασης. Στη συνέχεια, το σύστημα ανακτά τους εγγεγραμμένους και ενεργούς σπουδαστές του τμήματος (Enrollments). Ανάλογα με τις παραμέτρους εισόδου, η εξέταση είτε ανατίθεται μαζικά σε όλο το τμήμα, είτε σε συγκεκριμένο υποσύνολο σπουδαστών. Τέλος, χρησιμοποιώντας έναν επαναληπτικό βρόχο (loop fragment), δημιουργούνται οι συσχετιζόμενες οντότητες βαθμολόγησης (TestResult), οι οποίες αποθηκεύονται ταυτόχρονα (cascade save) με την εξέταση, ολοκληρώνοντας τον κύκλο της λειτουργίας με την επιστροφή κωδικού 201 Created.



Σχήμα 10: Διάγραμμα Ακολουθίας - Δημιουργία και Ανάθεση Εξέτασης (Create Test)

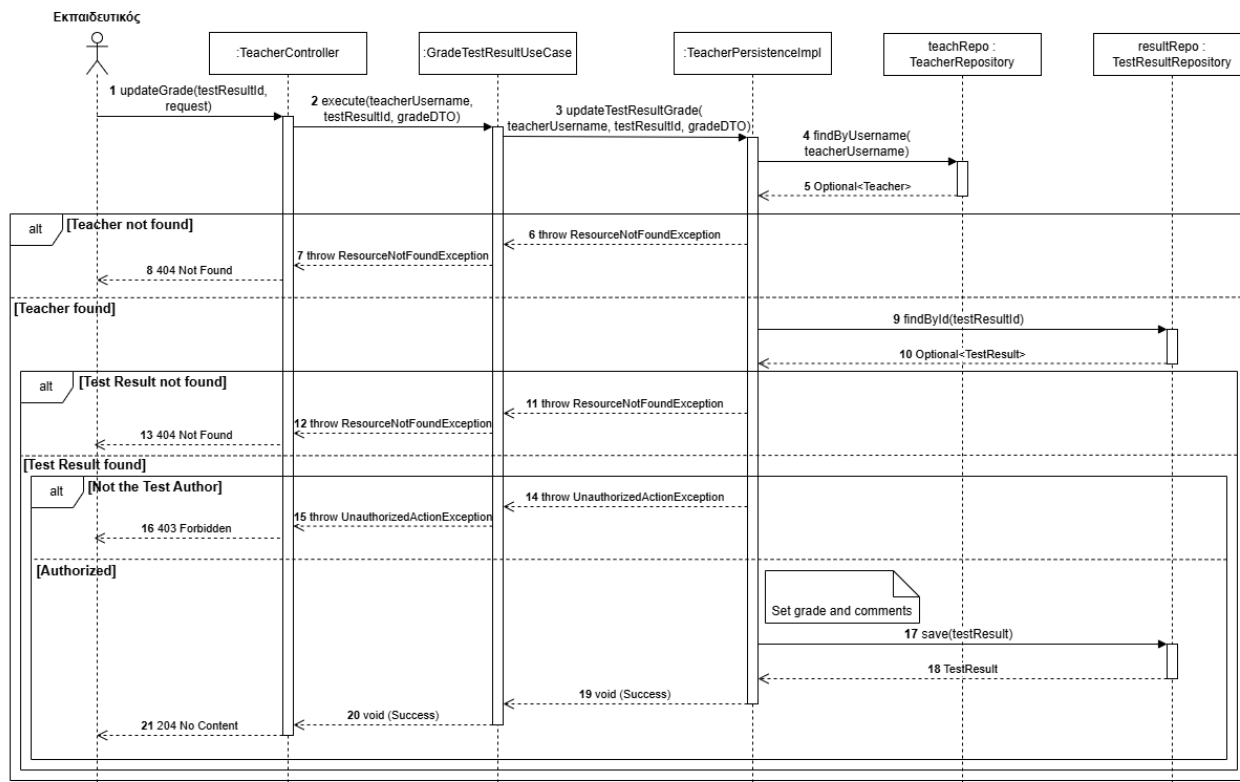
Η σχεδίαση της διαδικασίας δημιουργίας εξέτασης καταδεικνύει τη σταθερότητα του συστήματος και τη δυνατότά του να διαχειρίζεται πολύπλοκους κανόνες επιχειρησιακής λογικής, ενώ ταυτόχρονα προστατεύει τα δεδομένα των εγγεγραμμένων σπουδαστών. Έχοντας εξασφαλίσει την ορθή δημιουργία και ανάθεση της εξέτασης, το φυσιολογικό επόμενο βήμα στον ακαδημαϊκό κύκλο είναι η αξιολόγησή της. Συνεπώς, η ροή της μελέτης παραμένει στη σφαίρα ευθύνης του Εκπαιδευτικού (Teacher), περνώντας στην ανάλυση της “Βαθμολόγησης Σπουδαστών” (Grade Students). Μέσω της επόμενης μοντελοποίησης, θα αποτυπωθεί ο τρόπος με τον οποίο το σύστημα διαχειρίζεται την ασφαλή καταχώρηση και ενημέρωση των

αποτελεσμάτων (Test Results), διασφαλίζοντας την ακεραιότητα της βαθμολογικής κλίμακας και τα δικαιώματα πρόσβασης του εκπαιδευτικού.

Διάγραμμα 5: Βαθμολόγηση Σπουδαστών (Sequence Diagram)

Μετά την επιτυχή διεξαγωγή μιας εξέτασης, ο ακαδημαϊκός κύκλος ολοκληρώνεται με τη βαθμολόγηση των συμμετεχόντων. Η συγκεκριμένη λειτουργία (Grade Students) αναδεικνύει τους αυστηρούς μηχανισμούς εξουσιοδότησης (Authorization) που διαθέτει το σύστημα, διασφαλίζοντας το αδιάβλητο της εκπαιδευτικής διαδικασίας.

Το Διάγραμμα Ακολουθίας που ακολουθεί, αποτυπώνει την αλληλεπίδραση του Εκπαιδευτικού με το σύστημα κατά την ενημέρωση ενός αποτελέσματος εξέτασης (Test Result). Η αρχιτεκτονική ροή παραμένει συνεπής με το εξαγωνικό πρότυπο, μεταφέροντας το βάρος της επιχειρησιακής λογικής στο επίπεδο υποδομής (TeacherPersistenceImpl). Πριν την τελική αποθήκευση της βαθμολογίας, το σύστημα εκτελεί διαδοχικές ανακτήσεις από τη βάση δεδομένων και, το σημαντικότερο, επαληθεύει (Security Validation) ότι ο εκπαιδευτικός που επιχειρεί τη βαθμολόγηση είναι ο νόμιμος δημιουργός (Author) της εκάστοτε εξέτασης. Οποιαδήποτε απόπειρα παρέμβασης σε εξέταση άλλου εκπαιδευτικού, αναχαιτίζεται άμεσα με την επιστροφή σφάλματος άρνησης πρόσβασης (403 Forbidden).



Σχήμα 11: Διάγραμμα Ακολουθίας - Βαθμολόγηση Σπουδαστών (Grade Student)

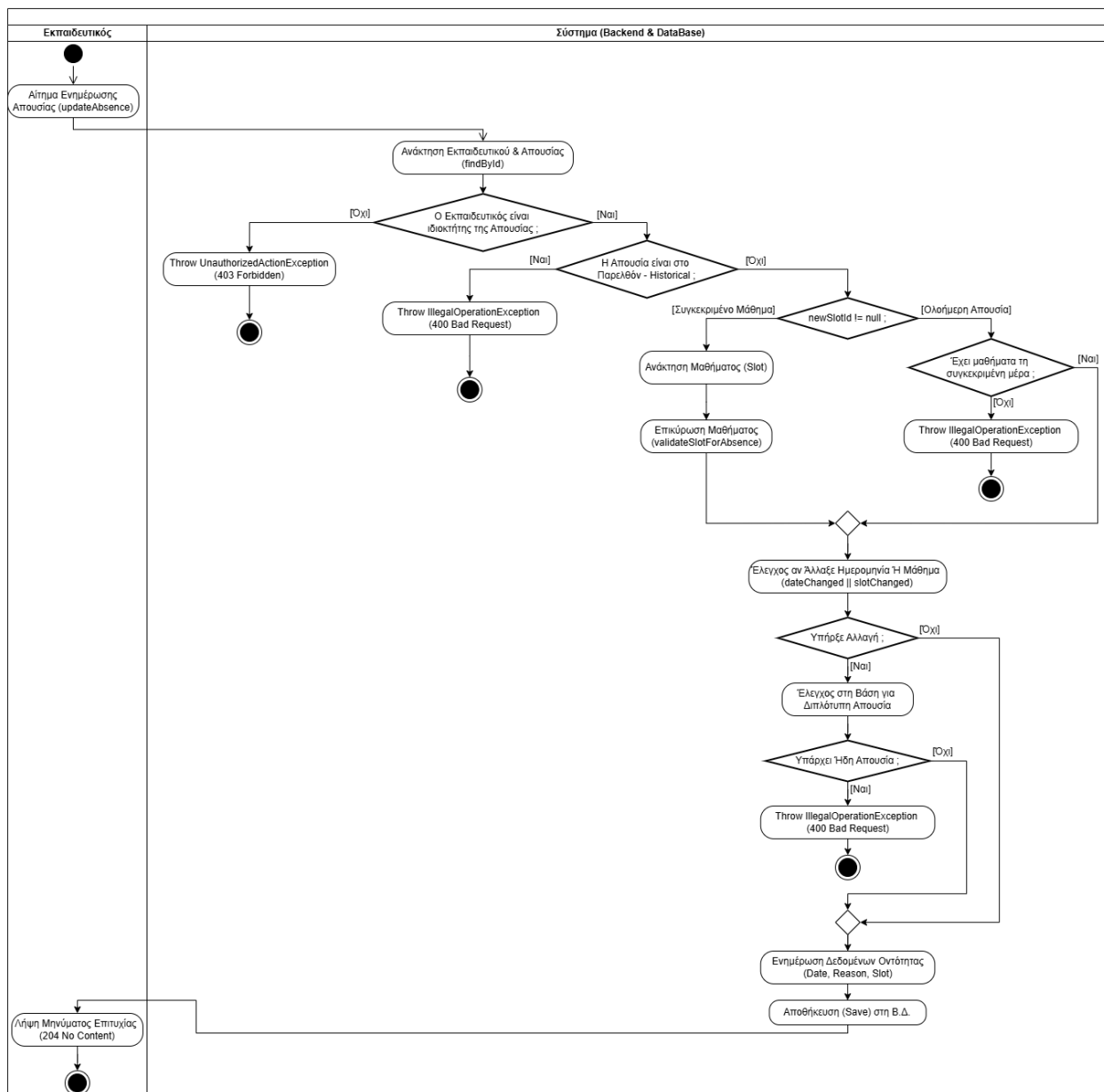
Η παραπάνω ακολουθία επιβεβαιώνει την ικανότητα του συστήματος να προστατεύει ευαίσθητα ακαδημαϊκά δεδομένα, περιορίζοντας τα δικαιώματα εγγραφής (write access) αποκλειστικά στους υπεύθυνους εκπαιδευτικούς. Η τελευταία περίπτωση χρήσης που αφορά τη λειτουργικότητα του Εκπαιδευτικού, επικεντρώνεται στη διαχείριση της διαθεσιμότητάς του. Στο επόμενο διάγραμμα αναλύεται η διαδικασία “Διαχείρισης Απουσιών” (Teacher Absences), μέσω της οποίας ο εκπαιδευτικός μπορεί να δηλώσει κώλυμα, είτε για ολόκληρη την ημέρα, είτε για συγκεκριμένο προγραμματισμένο μάθημα. Η

διαδικασία αυτή εμπεριέχει κρίσιμους δομικούς ελέγχους, προκειμένου να αποφευχθούν επικαλύψεις και να διασφαλιστεί η συνέπεια του ωρολογίου προγράμματος.

Διάγραμμα 6: Διαχείριση Απουσιών (Activity Diagram)

Η διαχείριση της διαθεσιμότητας του εκπαιδευτικού προσωπικού (Teacher Absences) αποτελεί μια από τις πλέον απαιτητικές περιπτώσεις χρήσης του συστήματος από πλευράς επιχειρησιακής λογικής (Business Logic). Ο εκπαιδευτικός διαθέτει την ευελιξία να δηλώσει, αλλά και να τροποποιήσει δυναμικά, είτε μια ολόκληρη απουσία, είτε την ακύρωση της παράδοσης ενός συγκεκριμένου μαθήματος (Hybrid Logic).

Για την αποτύπωση της πολυπλοκότητας αυτής της “υβριδικής” συμπεριφοράς, επιλέχθηκε το Διάγραμμα Δραστηριοτήτων (Activity Diagram). Σε αντίθεση με τις απεριόριστες εξουσίες του Διαχειριστή (Master Overrides), το σύστημα υποβάλλει τα αιτήματα του Εκπαιδευτικού σε αυστηρότερους ελέγχους (Guard Conditions). Αρχικά, διασφαλίζεται η ιδιοκτησία της εγγραφής (Authorization) και ακολούθως επιβάλλεται ο χρονικός περιορισμός που απαγορεύει την τροποποίηση ιστορικών / παρελθοντικών απουσιών. Στη συνέχεια, η ροή διακλαδώνεται: εφόσον ζητηθεί συγκεκριμένο μάθημα (slot), το σύστημα ανακτά το μάθημα και εκτελεί προσαρμοσμένη επαλήθευση (validate slot), ενώ στην περίπτωση ολόκληρης απουσίας, ελέγχει αν ο εκπαιδευτικός διδάσκει τη συγκεκριμένη ημέρα. Τέλος, η εφαρμογή αξιολογεί έξυπνα εάν τροποποιήθηκε η ημερομηνία ή η ώρα, εκτελώντας ερωτήματα στη βάση δεδομένων για αποτροπή διπλότυπων (Duplicate Check) μόνο εφόσον είναι απολύτως απαραίτητο, πριν ολοκληρώσει την αποθήκευση της νέας κατάστασης.



Σχήμα 12: Διάγραμμα Δραστηριοτήτων - Διαχείριση Απουσιών (Teacher Absences)

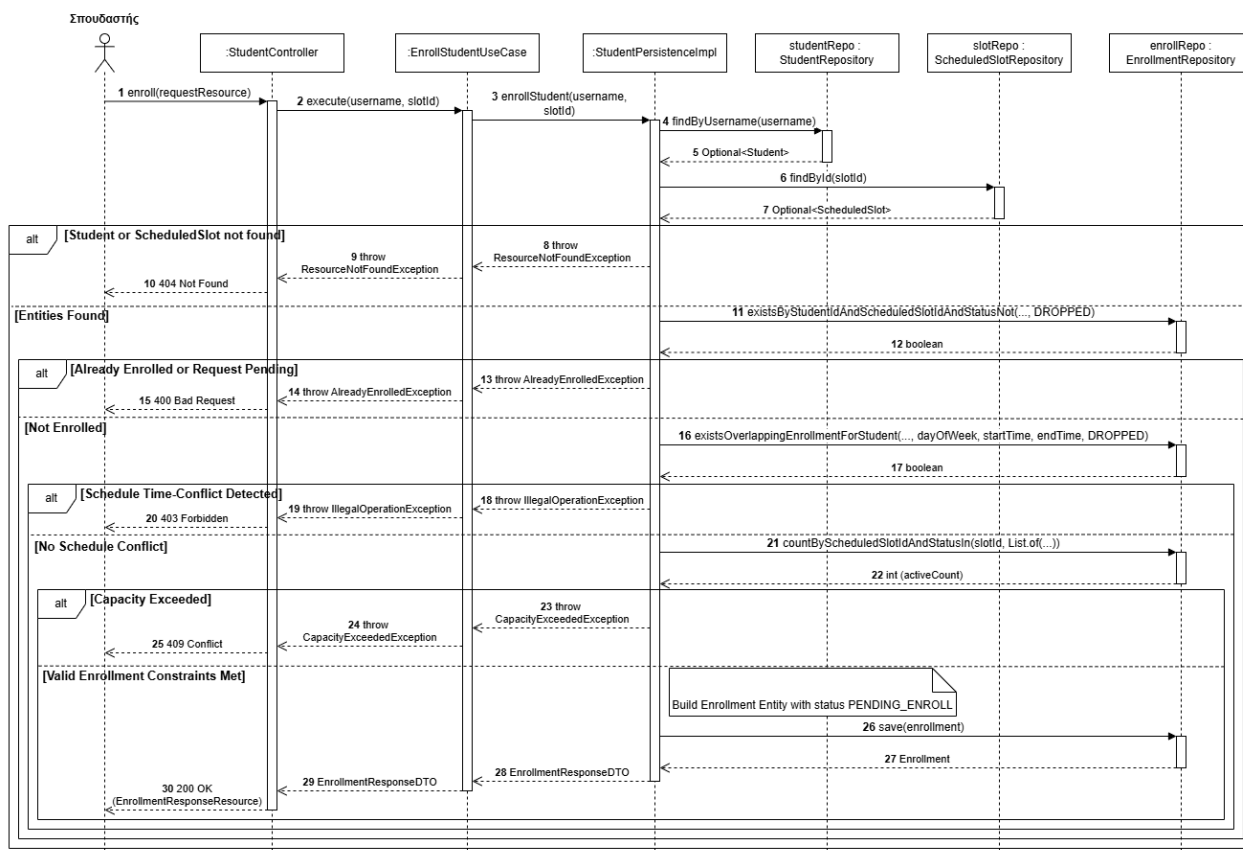
Η ανάλυση της λειτουργίας ελέγχου απουσιών ολοκληρώνει τον κύκλο των περιπτώσεων χρήσης που σχετίζονται με το εκπαιδευτικό προσωπικό. Η υλοποίηση της υβριδικής λογικής αναδεικνύει την ικανότητα του συστήματος να προσαρμόζεται σε ρεαλιστικά σενάρια λειτουργίας του φροντιστηρίου, αποτρέποντας ταυτόχρονα φαινόμενα απώλειας ιστορικών δεδομένων ή δημιουργίας επικαλύψεων. Με την ολοκλήρωση της μελέτης των ρόλων του Διαχειριστή και του Εκπαιδευτικού, η προσοχή στρέφεται πλέον στον πυρήνα της εκπαιδευτικής υπηρεσίας: τον Σπουδαστή (Student). Το επόμενο διάγραμμα σηματοδοτεί την έναρξη ανάλυσης των λειτουργιών του Σπουδαστή, εστιάζοντας στην πιο θεμελιώδη διαδικασία συμμετοχής του: το “Αίτημα Εγγραφής σε Τμήμα” (Enroll Request), μέσω του οποίου ξεκινά η παρακολούθηση των ακαδημαϊκών προγραμμάτων.

Διάγραμμα 7: Αίτημα Εγγραφής σε Τμήμα – Enroll Request (Sequence Diagram)

Περνώντας στον ρόλο του Σπουδαστή (Student), η πρώτη και πιο κρίσιμη λειτουργία που εξετάζεται είναι η υποβολή αιτήματος εγγραφής σε κάποιο προγραμματισμένο τμήμα (Enroll Request). Η συγκεκριμένη

περίπτωση χρήσης παρουσιάζει έντονο αλγοριθμικό ενδιαφέρον, καθώς το σύστημα καλείται να επικυρώσει το αίτημα σε πραγματικό χρόνο, εκτελώντας σύνθετους ελέγχους ακεραιότητας πριν επιτρέψει την καταχώρηση στη βάση δεδομένων.

Για την οπτικοποίηση αυτής της δυναμικής διαδικασίας, επιλέχθηκε το Διάγραμμα Ακολουθίας (Sequence Diagram). Όπως αναδεικνύεται στο γράφημα, η ροή ξεκινά από τον StudentController και μέσω του EnrollStudentUseCase καταλήγει στην υλοποίηση του επιπέδου υποδομής. Εκεί, το σύστημα αλληλοεπιδρά με το EnrollmentRepository για την επιβολή τριών διαδοχικών επιχειρησιακών κανόνων (Guard Clauses). Πρώτον, ελέγχεται η αποτροπή διπλότυπης εγγραφής (AlreadyEnrolledException). Δεύτερον, εκτελείται ο αλγόριθμος ελέγχου χρονικής επικάλυψης ωραρίου, ο οποίος διασφαλίζει ότι ο σπουδαστής δεν παρακολουθεί άλλο μάθημα την ίδια ημέρα και ώρα (IllegalOperationException). Τρίτον, επαληθεύεται ότι το τμήμα δεν έχει υπερβεί τη μέγιστη χωρητικότητά του (CapacityExceededException). Η χρήση φωλιασμένων δομών ελέγχου (nested alt fragments) αποτυπώνει την ικανότητα της εφαρμογής να εφαρμόζει τη στρατηγική άμεσης διακοπής σε περίπτωση αποτυχίας (Fail-Fast), επιστρέφοντας τους κατάλληλους κωδικούς σφαλμάτων (400, 403, 409), ενώ στο επιτυχές σενάριο (Happy Path) δημιουργείται η οντότητα με κατάσταση PENDING_ENROLL, επιστρέφοντας κωδικό 200 OK με τα πλήρη στοιχεία της εγγραφής.



Σχήμα 13: Διάγραμμα Ακολουθίας - Αίτημα Εγγραφής σε Τμήμα (Enroll Request)

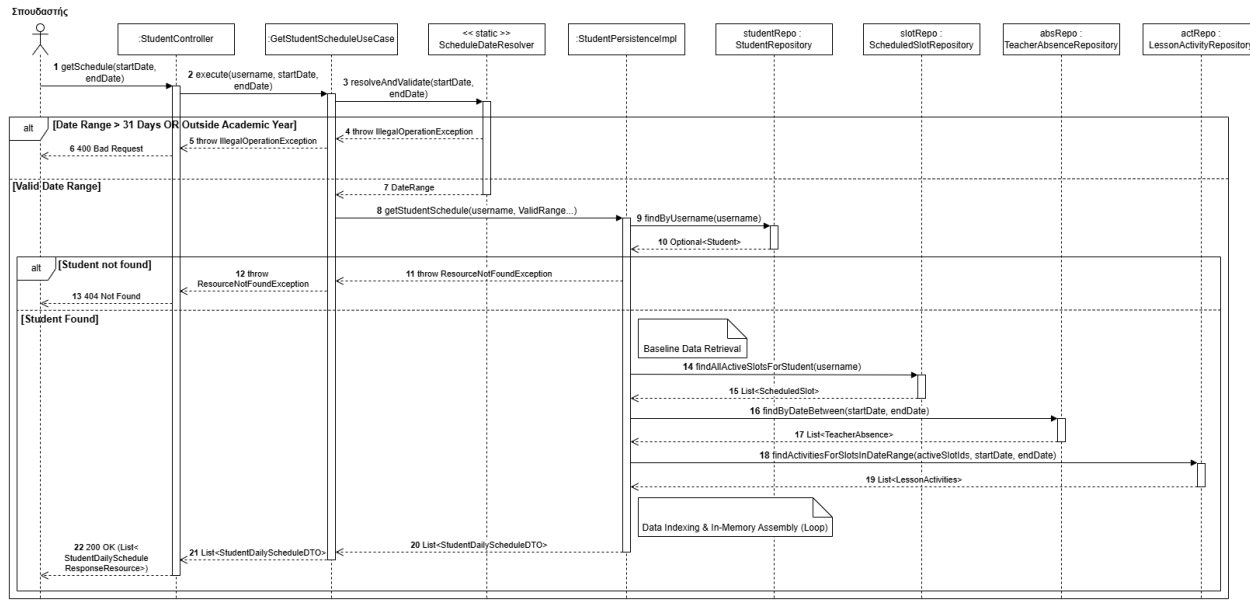
Η επιτυχής μοντελοποίηση του αιτήματος εγγραφής επιβεβαιώνει τη δομική ευστάθεια της εφαρμογής, η οποία αποτρέπει σε πρώτο επίπεδο οποιοδήποτε σφάλμα διπλοκρατήσεων ή υπερπλήρωσης των τμημάτων. Μόλις ο σπουδαστής ενταχθεί με επιτυχία στα επιθυμητά μαθήματα, η επόμενη βασική ανάγκη του είναι η παρακολούθηση της εκπαιδευτικής του καθημερινότητας. Ως εκ τούτου, η ανάλυση προχωρά στο 8^ο διάγραμμα, το οποίο αφορά την “Προβολή Προσωπικού Προγράμματος” (Weekly

Schedule). Η συγκεκριμένη ροή θα αναδειξεί τον τρόπο με τον οποίο το σύστημα συγκεντρώνει, φιλτράρει και ταξινομεί δεδομένα από διαφορετικές οντότητες, προκειμένου να κατασκευάσει μία ενιαία, εύχρηστη και χρονικά δομημένη εβδομαδιαία προβολή για τον σπουδαστή.

Διάγραμμα 8: Προβολή Προσωπικού Προγράμματος – Weekly Schedule (Sequence Diagram)

Μετά την εγγραφή του στα μαθήματα, η βασικότερη καθημερινή ανάγκη του σπουδαστή είναι η παρακολούθηση του ωρολογίου προγράμματός του. Η διαδικασία ανάκτησης του προγράμματος (Weekly Schedule) παρουσιάζει ιδιαίτερο αρχιτεκτονικό ενδιαφέρον, καθώς το σύστημα δεν ανακτά απλώς στατικά δεδομένα από μία πηγή, αλλά λειτουργεί ως μηχανισμός σύνθεσης (Data Aggregator).

Το Διάγραμμα Ακολουθίας (Sequence Diagram) που σχεδιάστηκε, αποτυπώνει την ικανότητα του συστήματος να αναλαμβάνει τη συγκέντρωση και επεξεργασία δεδομένων. Αρχικά, η ροή διέρχεται από την κλάση-εργαλείο `ScheduleDateResolver`, η οποία επικυρώνει το χρονικό εύρος του συστήματος, αποτρέποντας υπερφορτώσεις της βάσης δεδομένων (π.χ. αναζητήσεις άνω των 31 ημερών). Εφόσον τα όρια είναι έγκυρα, το επίπεδο υποδομής (`StudentPersistenceImpl`) αναλαμβάνει δράση, εκτελώντας μαζικές ανακτήσεις (bulk-fetch) από τρία διαφορετικά Repositories: τα ενεργά τμήματα του σπουδαστή (`ScheduledSlotRepository`), τις δηλωμένες απουσίες των καθηγητών (`TeacherAbsenceRepository`) και το καταχωρημένο εκπαιδευτικό υλικό (`LessonActivityRepository`). Το σύστημα, αξιοποιώντας δομές ευρετηρίασης (indexing), ταυτίζει στη μνήμη τα στατικά μαθήματα με τις δυναμικές αλλαγές (π.χ. μετατρέποντας την κατάσταση ενός μαθήματος σε “CANCELLED” εάν εντοπιστεί απουσία εκπαιδευτικού). Το τελικό αποτέλεσμα δομείται χρονολογικά, μετασχηματίζεται (mapping) στο επίπεδο ελέγχου, και επιστρέφεται στον χρήστη (200 OK) προσφέροντας μια ενιαία, πραγματικού χρόνου εικόνα της εκπαιδευτικής του εβδομάδας.



Σχήμα 14: Διάγραμμα Ακολουθίας - Προβολή Προσωπικού Προγράμματος (Weekly Schedule)

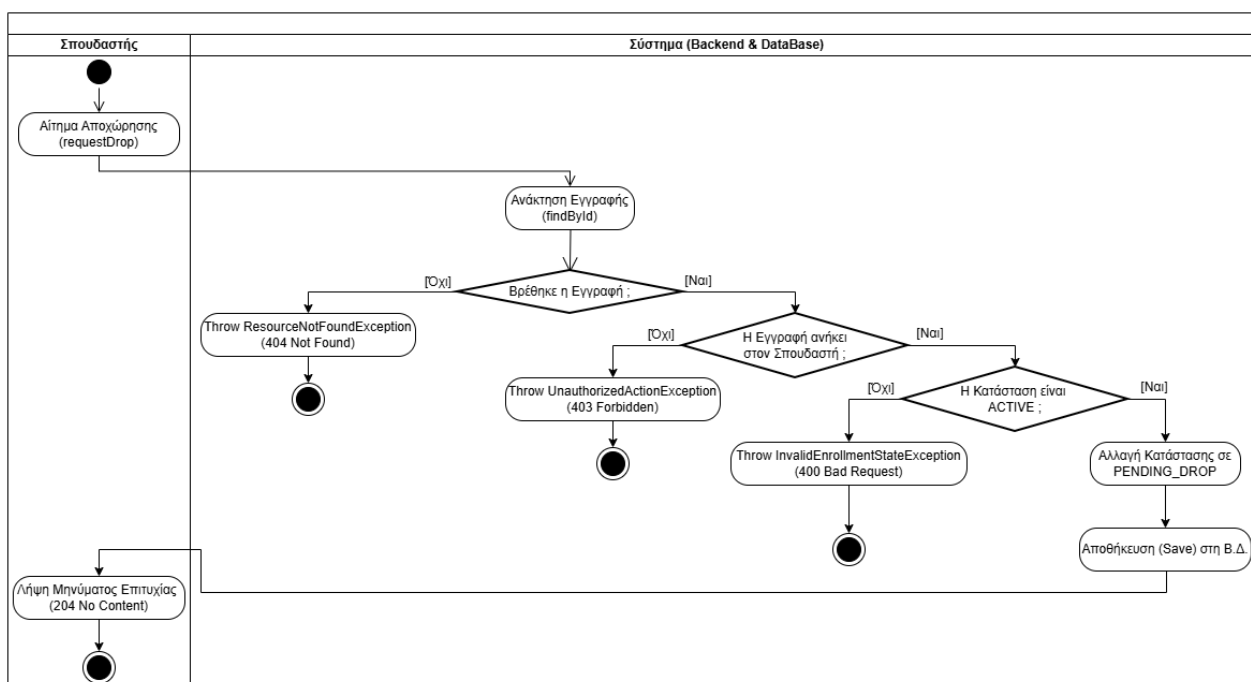
Το διάγραμμα συλλογής του εβδομαδιαίου προγράμματος καταδεικνύει την αποδοτικότητα της υποδομής, η οποία συγκεντρώνει διασκορπισμένα δεδομένα με ελάχιστα αιτήματα δικτύου (queries) δημιουργώντας μια πολύτιμη επισκόπηση για τον σπουδαστή. Η παρουσίαση των βασικών διαδικασιών ολοκληρώνεται με το 9^ο και τελευταίο διάγραμμα του συστήματος, το οποίο πραγματεύεται τη διαδικασία αποχώρησης από ένα τμήμα (“Αίτημα Αποχώρησης” – Request Drop). Η συγκεκριμένη ροή κλείνει τον

κύκλο ζωής μιας εγγραφής (Enrollment Lifecycle), απαιτώντας την ορθή μετάβαση καταστάσεων και την αποδέσμευση της κλειδωμένης χωρητικότητας του τμήματος.

Διάγραμμα 9: Αίτημα Αποχώρησης – Request Drop (Activity Diagram)

Ο κύκλος ζωής μιας εκπαιδευτικής εγγραφής (Enrollment Lifecycle) ολοκληρώνεται με την επιθυμία του σπουδαστή να αποχωρήσει από ένα τμήμα. Η αρχιτεκτονική του συστήματος δεν επιτρέπει την άμεση και οριστική διαγραφή της εγγραφής από τον ίδιο τον χρήστη, αλλά απαιτεί την υποβολή ενός αιτήματος αποχώρησης (Request Drop), το οποίο θέτει την εγγραφή σε κατάσταση αναμονής (PENDING_DROP) μέχρι την τελική έγκρισή της από τον Διαχειριστή.

Για την οπτικοποίηση αυτής της αλλαγής κατάστασης (State Transition), επιλέχθηκε το Διάγραμμα Δραστηριοτήτων (Activity Diagram), το οποίο χαρτογραφεί τη λογική ελέγχου πρόσβασης και εγκυρότητας. Κατά την εκτέλεση του αιτήματος στο επίπεδο της υποδομής (StudentPersistenceImpl), το σύστημα εφαρμόζει δύο κρίσιμους περιορισμούς: Πρώτον, εκτελείται Έλεγχος Ιδιοκτησίας (Property Control), διασφαλίζοντας ότι ο σπουδαστής μπορεί να ακυρώσει μόνο τις δικές του εγγραφές (επιστρέφοντας 403 Forbidden σε αντίθετη περίπτωση). Δεύτερον, εξετάζεται η τρέχουσα κατάσταση της οντότητας, καθώς το σύστημα επιτρέπει την αποχώρηση αυστηρά και μόνο από ενεργά τμήματα (ACTIVE), απορρίπτοντας λανθασμένα αιτήματα (π.χ. προσπάθεια αποχώρησης από ήδη ακυρωμένο μάθημα) με κωδικό 400 Bad Request. Εφόσον οι έλεγχοι ικανοποιηθούν, η κατάσταση μεταβάλλεται και αποθηκεύεται με ασφάλεια στη βάση δεδομένων.



Σχήμα 15: Διάγραμμα Δραστηριοτήτων - Αίτημα Αποχώρησης (Request Drop)

Η μοντελοποίηση του αιτήματος αποχώρησης ολοκληρώνει τον κύκλο ανάλυσης των διαγραμμάτων συμπεριφοράς του συστήματος. Μέσα από την παραπάνω ροή, εξασφαλίζεται η ομαλή μετάβαση καταστάσεων, κλείνοντας τον κύκλο που άνοιξε με τη αυθεντικοποίηση χρήστη (βλ. Σχήμα 7).

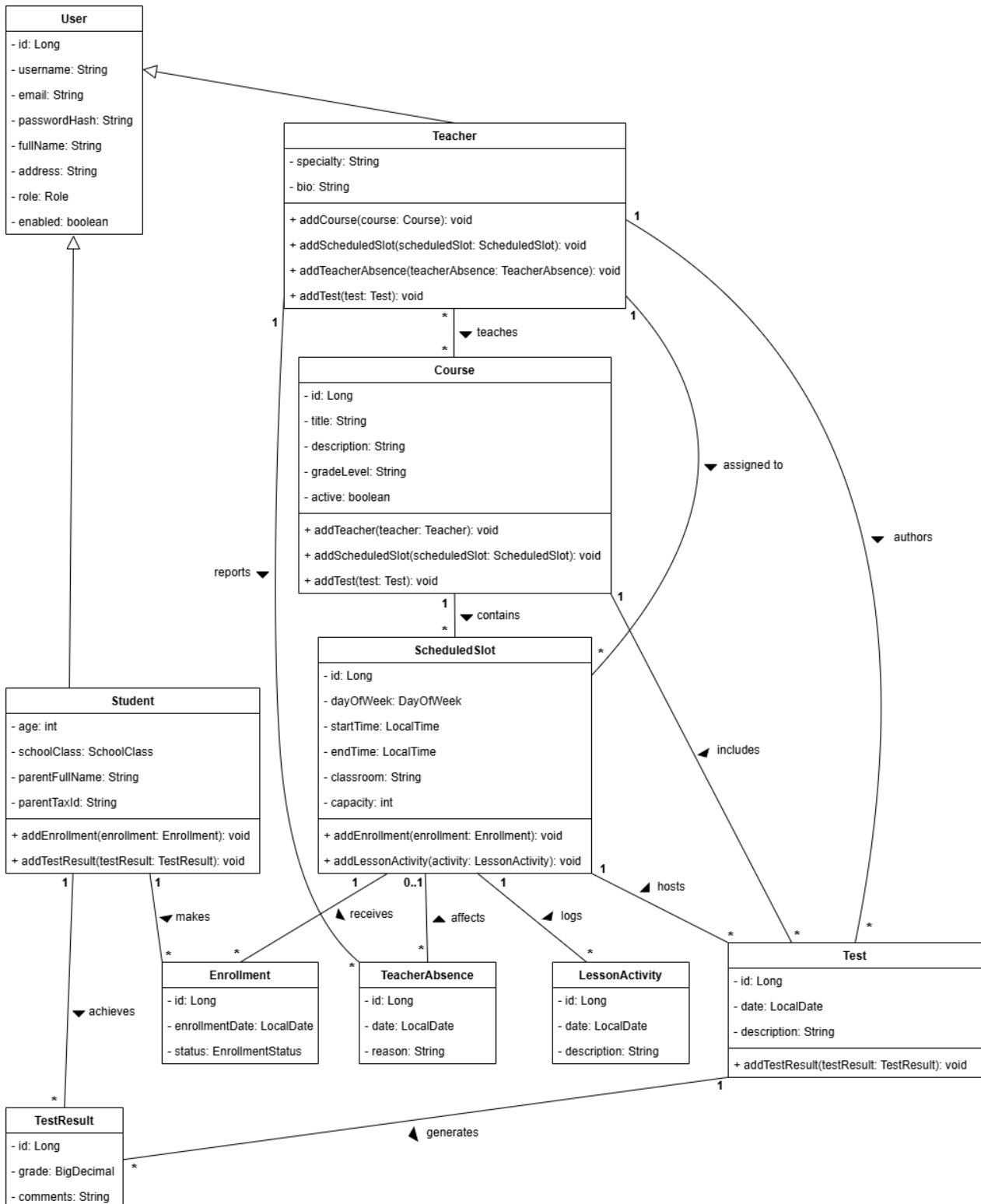
Συνοψίζοντας το παρόν υποκεφάλαιο, η ενδελεχής σχεδίαση των εννέα (9) κρίσιμων σεναρίων χρήσης – αξιοποιώντας τόσο Διαγράμματα Ακολουθίας όσο και Δραστηριοτήτων – ανέδειξε την αρχιτεκτονική συνέπεια της εφαρμογής. Αποδείχθηκε έμπρακτα η ευρωστία της Εξαγωνικής Αρχιτεκτονικής, η αυστηρή τήρηση των κανόνων ασφαλείας (Authorization), καθώς και η ικανότητα του

συστήματος να επιλύει περίπλοκους επιχειρησιακούς κανόνες για όλους τους ρόλους (Διαχειριστή, Εκπαιδευτικό και Σπουδαστή), διατηρώντας ακέραια την αξιοπιστία των δεδομένων (Data Integrity).

3.3.5 Διάγραμμα Κλάσεων / Βάσης Δεδομένων (Class / Database Diagram)

Το Διάγραμμα Κλάσεων (Class Diagram) αποτελεί ίσως το σημαντικότερο στατικό διάγραμμα της Ενοποιημένης Γλώσσας Μοντελοποίησης (UML), καθώς παρουσιάζει δομημένα την οργάνωση του συστήματος, αποτυπώνοντας τα συστατικά του μέρη (κλάσεις) και τις μεταξύ τους συσχετίσεις (Planas & Cabot, 2020). Στο πλαίσιο της Εξαγωνικής Αρχιτεκτονικής που υιοθετείται στην παρούσα εφαρμογή αλλά και του γεγονότος πως βρισκόμαστε στο στάδιο της σχεδίασης, η μοντελοποίηση του διαγράμματος επικεντρώνεται στον πυρήνα του συστήματος, αναπαριστώντας τις θεμελιώδεις επιχειρησιακές οντότητες (Domain Entities).

Όπως φαίνεται στο Σχήμα 16, αξιοποιούνται αυστηροί κανόνες ορατότητας (visibility), καθορίζοντας όλα τα πεδία ως ιδιωτικά (- private) για τη διασφάλιση της ενθυλάκωσης (encapsulation), ενώ οι λειτουργίες συγχρονισμού (helper methods) είναι δημόσιες (+ public). Ιδιαίτερο ενδιαφέρον παρουσιάζει η δομή κληρονομικότητας (Generalization), όπου οι κεντρικοί ρόλοι του συστήματος (Student και Teacher) επεκτείνουν τη βασική κλάση User, κληρονομώντας κοινά χαρακτηριστικά ταυτοποίησης και ασφάλειας. Επιπλέον, αποτυπώνονται με σαφήνεια οι πληθυκότητες (multiplicities) των συσχετίσεων, όπως η σχέση 1-προς-Πολλά (1..*) μεταξύ του Μαθήματος (Course) και των Προγραμματισμένων Τμημάτων (ScheduledSlot).

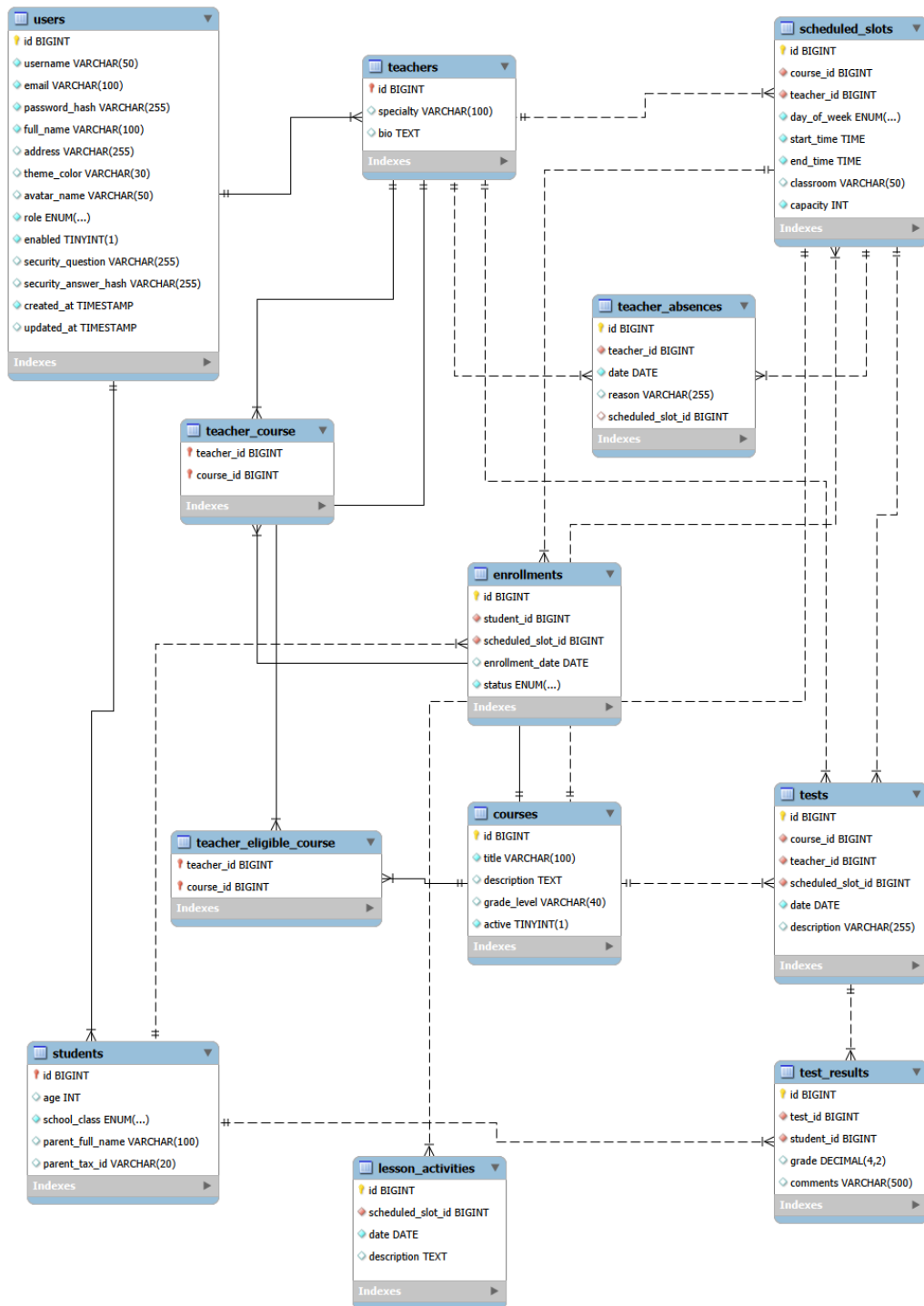


Σχήμα 16: Διάγραμμα Κλάσεων (Domain Model Class Diagram)

Η δομική αρχιτεκτονική που αναλύθηκε στο Διάγραμμα Κλάσεων μεταφράζεται άμεσα στο φυσικό σχήμα της σχεσιακής βάσης δεδομένων (Database Schema). Για την πιστή απεικόνιση της δομής αποθήκευσης των δεδομένων, χρησιμοποιείται το Διάγραμμα Βάσης Δεδομένων (Database Diagram). Όπως επισημαίνεται στη σύγχρονη βιβλιογραφία, η χρήση διαγραμμάτων Database είναι κρίσιμη για την

κατανόηση της ροής και της δομής δεδομένων, διευκολύνοντας την κλιμάκωση (scalability) και τη διαχείριση της πληροφορίας (Ordonez κ.ά., 2020).

Στο Σχήμα 17 παρουσιάζεται το διάγραμμα βάσης, το οποίο παρήχθη μέσω μεθόδων αντίστροφης μηχανικής (Reverse Engineering) απευθείας από τη βάση δεδομένων της εφαρμογής, εξασφαλίζοντας την απόλυτη ταύτιση της θεωρητικής μοντελοποίησης με την τελική τεχνική υλοποίηση. Το διάγραμμα αποτυπώνει τα πρωτεύοντα (Primary Keys) και τα ξένα κλειδιά (Foreign Keys), καθώς και τους ενδιάμεσους πίνακες (π.χ. teacher_course) που δημιουργούνται αυτόματα για την επίλυση των σχέσεων Πολλά-πορος-Πολλά (Many-to-Many).



Σχήμα 17: Διάγραμμα Βάσης (Database Diagram)

Παρατηρώντας το φυσικό σχήμα της βάσης δεδομένων (Σχήμα 17) σε σχέση με το εννοιολογικό Διάγραμμα Κλάσεων (Σχήμα 16), διαπιστώνεται μια αυξημένη δομική πολυπλοκότητα, η οποία είναι απόλυτα αναμενόμενη. Αυτή η διαφοροποίηση οφείλεται στην εγγενή αναντιστοιχία Αντικειμενοστρεφούς-Σχεσιακού μοντέλου (Object-Relational Impedance Mismatch). Για παράδειγμα, ενώ στον κώδικα (Java) μια σχέση Πολλά-προς-Πολλά αναπαρίσταται απλώς ως μια συλλογή δεδομένων (Collection), στο Σχεσιακό Μοντέλο απαιτείται η δημιουργία ενδιάμεσων πινάκων (join tables) με σύνθετα πρωτεύοντα κλειδιά. Επιπλέον, κεντρικές οντότητες, όπως το Προγραμματισμένο Τμήμα (scheduled_slots), λειτουργούν ως κόμβοι, συγκεντρώνοντας πολλαπλά ξένα κλειδιά για τη διατήρηση της ακεραιότητας των συσχετίσεων.

Με την ολοκλήρωση του Διαγράμματος Βάσης Δεδομένων, κλείνει ο κύκλος του Αρχιτεκτονικού και Εννοιολογικού Σχεδιασμού. Το Υποκεφάλαιο 3.4 (Υλοποίηση) που ακολουθεί, θα αναλύσει τα εργαλεία και τις τεχνολογίες (όπως το Spring Boot και το MapStruct που διεκπεραιώνει τα προαναφερθέντα mappings) που αξιοποιήθηκαν στο εσωτερικό αυτών των modules.

3.4 Υλοποίηση

Η φάση της υλοποίησης αποτελεί το κρίσιμότερο στάδιο του κύκλου ζωής της ανάπτυξης λογισμικού, καθώς μετουσιώνει τον θεωρητικό, αρχιτεκτονικό και εννοιολογικό σχεδιασμό σε ένα πλήρως λειτουργικό σύστημα. Στο παρόν κεφάλαιο αναλύεται η διαδικασία μετάβασης από τα διαγράμματα της Ενοποιημένης Γλώσσας Μοντελοποίησης (UML) και την Εξαγωνική Αρχιτεκτονική στη συγγραφή του πηγαίου κώδικα. Παρουσιάζονται αναλυτικά οι τεχνολογίες, οι γλώσσες προγραμματισμού και τα πλαίσια ανάπτυξης (frameworks) που επιλέχθηκαν, ενώ παράλληλα επεξηγείται η οργάνωση του κώδικα σε πακέτα (packages) και αναλύονται τα δομικά στοιχεία που διασφαλίζουν την ασφάλεια, την επεκτασιμότητα και τη συντηρησιμότητα της εφαρμογής. Σημειώνεται ότι εκτενή, αυτούσια αποσπάσματα του παραγόμενου πηγαίου κώδικα παρατίθεται συγκεντρωτικά στο Παράρτημα Ι της παρούσας εργασίας, προσφέροντας άμεση πρόσβαση στις αναλυτικές τεχνικές λεπτομέρειες της υλοποίησης.

3.4.1 Τεχνολογίες και Γλώσσες Προγραμματισμού

Για την ανάπτυξη του Συστήματος Διαχείρισης Φροντιστηρίου, υιοθετήθηκε μία σύγχρονη τεχνολογική στοίβα (tech stack) που διαχωρίζει διακριτά τη λογική του διακομιστή (Back-End) από τη διεπαφή του χρήστη (Front-End).

Τεχνολογίες Διακομιστή (Back-End)

Η επιχειρησιακή λογική, η διαχείριση των δεδομένων και η υλοποίηση του πυρήνα του συστήματος αναπτύχθηκαν αποκλειστικά στη γλώσσα προγραμματισμού **Java**. Η επιλογή της Java βασίστηκε στα ισχυρά χαρακτηριστικά αντικειμενοστρεφούς προγραμματισμού (Object-Oriented Programming – OOP) που διαθέτει, τα οποία είναι θεμελιώδη για την ανάπτυξη πολύπλοκων εταιρικών συστημάτων (Enterprise Applications).

Όπως τεκμηριώνεται στη σύγχρονη βιβλιογραφία, οι αρχές του OOP, όπως η ενθυλάκωση (encapsulation) και η κληρονομικότητα (inheritance), επιτρέπουν τη δημιουργία επεκτάσιμου και επαναχρησιμοποιήσιμου κώδικα, ελαχιστοποιώντας τα σφάλματα κατά τη φάση της συντήρησης (Singh κ.ά., 2021). Στο πλαίσιο της εφαρμογής, οι αρχές αυτές αξιοποιήθηκαν πλήρως για τη μοντελοποίηση των οντοτήτων. Χαρακτηριστικό παράδειγμα αποτελεί η ιεραρχική δομή των χρηστών, όπου μια κεντρική

κλάση (π.χ. User) ενθυλακώνει τα κοινά δεδομένα ταυτοποίησης (όπως όνομα χρήστη και κωδικό πρόσβασης), επιτρέποντας σε εξειδικευμένες κλάσεις (όπως οι Student και Teacher) να κληρονομούν αυτά τα χαρακτηριστικά και να επεκτείνουν τη λειτουργικότητα με εξειδικευμένες ιδιότητες (όπως η ηλικία ή η ειδικότητα αντίστοιχα). Επιπλέον, το αυστηρό σύστημα τύπων (strong typing) της Java διασφαλίζει την ακεραιότητα των δεδομένων κατά τη μεταφορά των διαφορετικών υποσυστημάτων της Εξαγωνικής Αρχιτεκτονικής.

Τεχνολογίες Διεπαφής Χρήστη (Front-End)

Για τον σχεδιασμό της αλληλεπίδρασης με τον τελικό χρήστη, αξιοποιήθηκε ο συνδυασμός των θεμελιωδών τεχνολογιών του Παγκοσμίου Ιστού: **HTML5**, **CCS3** και **JavaScript**, υποστηριζόμενες από τη μηχανή προτύπων (template engine) **Thymeleaf**.

- **HTML5 & CSS3:** Η γλώσσα σήμανσης HTML5 χρησιμοποιήθηκε για τη σημασιολογική δόμηση (semantic structuring) του περιεχομένου της ιστοσελίδας. Αντίστοιχα, η γλώσσα μορφοποίησης CSS3 ανέλαβε την οπτική παρουσίαση (styling) και την υλοποίηση του προσαρμοστικού σχεδιασμού (responsive design). Όπως επισημαίνουν οι Rao κ.ά. (2023), οι σύγχρονες τεχνικές ιστού (modern web design) επιτρέπουν τη δημιουργία ελκυστικών και προσβάσιμων διεπαφών. Αξιοποιώντας αυτές τις δυνατότητες, η εφαρμογή υιοθετεί τη σχεδιαστική φιλοσοφία των “καρτών” (Card-based UI). Αντί της χρήσης παραδοσιακών, δυσανάγνωστων πινάκων δεδομένων, η πληροφορία (όπως τα στοιχεία των χρηστών ή τα διαθέσιμα μαθήματα) παρουσιάζεται σε δυναμικές κάρτες, οι οποίες βελτιστοποιούν την εμπειρία πλοήγησης (UX) και είναι πλήρως προσαρμόσιμες σε κινητές συσκευές.
- **JavaScript:** Ως γλώσσα προγραμματισμού από την πλευρά του πελάτη (client-side), η JavaScript ενσωματώθηκε για να εμπλουτίσει τη δυναμική συμπεριφορά της εφαρμογής. Αναλαμβάνει κρίσιμες λειτουργίες στο επίπεδο της διεπαφής, όπως τη διαχείριση συμβάντων (event handling) για την εμφάνιση πλαισίων διαλόγου (Modals) με αναλυτικές πληροφορίες όταν ο χρήστης επιλέγει μια κάρτα, την επικύρωση δεδομένων φορμών σε πραγματικό χρόνο, καθώς και τη διαχείριση της κατάστασης της διεπαφής (state management), όπως η αποθήκευση και ανάκτηση της προτίμησης θέματος (Forest Green/Royal Purple mode) μέσω του *localStorage* του περιηγητή (browser).
- **Thymeleaf (Εισαγωγή):** Η σύνδεση της HTML/CSS με τα δεδομένα του Java Back-End επιτεύχθηκε μέσω της μηχανής Thymeleaf, η οποία επιτρέπει τη δυναμική δημιουργία των σελίδων και την επαναχρησιμοποίηση δομικών στοιχείων (όπως η μπάρα πλοήγησης), λειτουργία που θα αναλυθεί εκτενέστερα στην επόμενη ενότητα των πλαισίων ανάπτυξης.

3.4.2 Πλαίσια Ανάπτυξης και Βιβλιοθήκες

Για την ταχεία, ασφαλή και αξιόπιστη ανάπτυξη του συστήματος, η γλώσσα Java συνοδεύτηκε από ένα σύνολο σύγχρονων πλαισίων, εργαλείων και βιβλιοθηκών, με κυρίαρχο το οικοσύστημα του Spring.

Το Πλαίσιο Spring Boot

Το κεντρικό πλαίσιο (framework) πάνω στο οποίο βασίστηκε η αρχιτεκτονική της εφαρμογής είναι το Spring Boot. Παραδοσιακά, η ανάπτυξη εταιρικών εφαρμογών Java απαιτούσε εκτενείς και πολύπλοκες ρυθμίσεις (boilerplate configuration). Το Spring Boot επιλύει αυτό το πρόβλημα παρέχοντας μηχανισμούς “Αυτόματης Ρύθμισης” (Autoconfiguration) και ενσωματωμένους διακομιστές (όπως ο Apache Tomcat), επιτρέποντας την άμεση εκτέλεση της εφαρμογής (ιστού).

Μέσω του αρχείου διαχείρισης εξαρτήσεων `pom.xml` (Maven), το Spring Boot απλοποίησε δραματικά την ενσωμάτωση εξωτερικών βιβλιοθηκών, διασφαλίζοντας τη συμβατότητα των εκδόσεων (version management) χωρίς χειροκίνητες παρεμβάσεις. Παράλληλα, ο πυρήνας του Spring προσφέρει εγγενώς τους μηχανισμούς Αντιστροφής Ελέγχου (Inversion of Control – IoC) και Εισαγωγής Εξαρτήσεων (Dependency Injection – DI). Οι μηχανισμοί αυτοί αποδέσμευσαν τον κώδικα από τη χειροκίνητη δημιουργία αντικειμένων (instantiation), επιτρέποντας στο framework να διαχειριστεί τον κύκλο ζωής των συστατικών (Beans), μειώνοντας κατακόρυφα τον χρόνο υλοποίησης και αυξάνοντας τη δοκιμασιμότητα (testability) του κώδικα.

Spring Security: Αυθεντικοποίηση και Εξουσιοδότηση

Η προστασία των δεδομένων και ο έλεγχος πρόσβασης (Access Control) υλοποιήθηκαν μέσω της βιβλιοθήκης Spring Security. Όπως αναλύεται στη βιβλιογραφία, η ασφάλεια σε συστήματα βασισμένα σε Java απαιτεί ισχυρούς μηχανισμούς τόσο για την αυθεντικοποίηση (authentication – επιβεβαίωση ταυτότητας χρήστη) όσο και για την εξουσιοδότηση (authorization – έλεγχος δικαιωμάτων) (Jangala, 2024). Το Spring Security παραμετροποιήθηκε ώστε να εφαρμόζει Έλεγχο Πρόσβασης Βάσει Ρόλων (Role-Based Access Control – RBAC). Διαχειρίζεται τον κατακερματισμό των κωδικών πρόσβασης (hashing) και επιτρέπει ή απορρίπτει την πρόσβαση σε συγκεκριμένα τελικά σημεία (endpoints) ανάλογα το ρόλο του συνδεδεμένου χρήστη (Διαχειριστής, Εκπαιδευτικός, Σπουδαστής), διασφαλίζοντας την ακεραιότητα της πλατφόρμας.

MapStruct: Βελτιστοποίηση Μετασχηματισμού Δεδομένων

Για την υλοποίηση της στρατηγικής του “διπλού μετασχηματισμού” (Double Mapping) που παρουσιάστηκε στο Διάγραμμα Συστατικών, επιλέχθηκε η βιβλιοθήκη MapStruct. Η μετατροπή δεδομένων μεταξύ των Αντικειμένων Μεταφοράς (DTOs) και των Οντοτήτων (JPA Entities) είναι μια συνήθης πρακτική που, ωστόσο, μπορεί να κοστίζει σε απόδοση. Σε αντίθεση με παλαιότερα εργαλεία που χρησιμοποιούν τη δυναμική αλλά αργή μέθοδο του Reflection κατά τον χρόνο εκτέλεσης (runtime), το MapStruct λειτουργεί κατά το χρόνο μεταγλώττισης (compile-time). Παράγει αυτόματα καθαρό κώδικα Java για τη χαρτογράφηση των πεδίων, εντοπίζοντας πιθανά σφάλματα αντιστοίχισης πριν καν εκτελεστεί η εφαρμογή, προσφέροντας έτσι ασφάλεια τύπων (type safety) και μέγιστη ταχύτητα εκτέλεσης.

Thymeleaf: Server-Side Rendering (SSR)

Όπως αναφέρθηκε στην προηγούμενη ενότητα, η διεπαφή της εφαρμογής τροφοδοτείται από τη μηχανή προτύπων Thymeleaf. Η αρχιτεκτονική του Thymeleaf βασίζεται στο μοντέλο Απόδοσης από την Πλευρά του Διακομιστή (Server-Side Rendering – SSR). Σε αντίθεση με τις προσεγγίσεις Client-Side Rendering, όπου το πρόγραμμα περιήγησης αναλαμβάνει την κατασκευή του HTML μέσω της JavaScript, το SSR κατασκευάζει την πλήρη σελίδα στον διακομιστή πριν την αποστείλει στον χρήστη (Eriksson, 2022). Η επιλογή αυτή εναρμονίστηκε πλήρως με το οικοσύστημα του Spring MVC, επιτρέποντας την άμεση και ασφαλή ενσωμάτωση δεδομένων από το Back-End (όπως τα στοιχεία του τρέχοντος χρήστη) απευθείας στα αρχεία HTML, ενισχύοντας τόσο την ταχύτητα φόρτωσης όσο και την ασφάλεια έναντι επιθέσεων XSS (Cross-Site Scripting).

3.4.3 Δομή Πηγαίου Κώδικα (Packages, DTOs, Mappers)

Η αρχιτεκτονική του πηγαίου κώδικα οργανώθηκε με γνώμονα τον αυστηρό διαχωρισμό των αρμοδιοτήτων (Separation of Concerns), υιοθετώντας μια δομή πολλαπλών ενοτήτων (multi-module) μέσω του εργαλείου αυτοματοποίησης Maven. Η προσέγγιση αυτή επιβάλλει φυσικά όρια μεταξύ των επιπέδων

της Εξαγωνικής Αρχιτεκτονικής, αποτρέποντας την κυκλική εξάρτηση (circular dependency) και προστατεύοντας τον πυρήνα (Domain/Application) από τις τεχνολογικές λεπτομέρειες των εξωτερικών προσαρμογών.

Οργάνωση των Maven Modules

Το έργο χωρίζεται στις εξής διακριτές ενότητες, καθεμία με αποκλειστικό ρόλο:

- **domain:** Αποτελεί τον ανεξάρτητο πυρήνα που περιέχει τους θεμελιώδεις τύπους (π.χ. Enumerations όπως το `SchoolClass`) και την ιεραρχία των προσαρμοσμένων εξαιρέσεων της εφαρμογής (Custom Exceptions). Δεν εξαρτάται από εξωτερικά πλαίσια (frameworks).
- **application:** Φιλοξενεί την επιχειρησιακή λογική (Use Cases) και καθορίζει τις συμβάσεις επικοινωνίας (interfaces) μέσω των θυρών `port.in` και `port.out`. Εδώ ορίζονται και τα Αντικείμενα Μεταφοράς Δεδομένων (DTOs) που χρησιμοποιεί ο πυρήνας.
- **infrastructure:** Λειτουργεί ως υπερ-ενότητα (parent module) για τους τεχνολογικούς προσαρμογείς. Περιλαμβάνει υποενότητες όπως το **repository** (όπου βρίσκονται τα JPA Entities και η επικοινωνία με τη βάση δεδομένων) και το **rest** (που περιέχει τους ελεγκτές API, τα Resources του UI και τον μηχανισμό ασφαλείας).
- **webapplication:** Το τελικό εκτελέσιμο τμήμα που συγκεντρώνει όλα τα υποσυστήματα. Περιέχει τα στατικά αρχεία, τα πρότυπα της διεπαφής (Thymeleaf, CSS, JS) και τις αυτοματοποιημένες δοκιμές ελέγχου αρχιτεκτονικής (ArchUnit Tests). Οι δοκιμές αυτές αποτελούν μια προηγμένη τεχνική διασφάλισης ποιότητας, καθώς “επιβάλλουν” μέσω κώδικα ότι καμία κλάση δεν θα παραβιάσει τους κανόνες της αρχιτεκτονικής (π.χ. απαγορεύοντας ρητά σε ένα αρχείο του **application** να εισάγει κώδικα από το **repository**).

Αντικείμενα Μεταφοράς Δεδομένων (DTOs) και Ασφάλεια

Για την προστασία του μοντέλου δεδομένων, η ανταλλαγή πληροφοριών μεταξύ των επιπέδων γίνεται αποκλειστικά μέσω DTOs (Data Transfer Objects). Η πρακτική αυτή αποτρέπει την άμεση έκθεση των JPA Entities στο διαδίκτυο, επιλύει ζητήματα “Lazy Loading” και προστατεύει από επιθέσεις μαζικής εκχώρησης (Mass Assignment).

Χαρακτηριστικό παράδειγμα αποτελεί η οντότητα καταγραφής απουσιών (TeacherAbsence) και το αντίστοιχο μεταφερόμενο αντικείμενο (TeacherAbsenceDTO). Ενώ στη βάση δεδομένων η απουσία ενός εκπαιδευτικού συνδέεται δομικά με μια ολόκληρη οντότητα ScheduledSlot (μέσω της σχέσης @ManyToOne), το DTO μεταφέρει προς το περιβάλλον χρήστη μόνο το αυστηρά απαραίτητο αναγνωριστικό (Long slotId). Αυτή η “επιπεδοποίηση” (flattening) των δεδομένων ελαχιστοποιεί το μέγεθος του ωφέλιμου φορτίου (payload) και κρύβει την εσωτερική σχεσιακή δομή της βάσης δεδομένων από τον τελικό χρήστη.

Αυτοματοποιημένος Μετασχηματισμός (MapStruct)

Η γεφύρωση μεταξύ των DTOs του επιπέδου Application και των Entities του επιπέδου Repository επιτυγχάνεται μέσω της βιβλιοθήκης MapStruct, εντός κεντρικών κλάσεων όπως ο `EntityMapper`. Ο κώδικας αυτού του μετασχηματιστή αποτελεί τον ακρογωνιαίο λίθο για την ασφαλή διαχείριση της πληροφορίας, υλοποιώντας κρίσιμες λειτουργίες κατά τον χρόνο μεταγλώττισης (compile-time):

- **Παράβλεψη Πεδίων (Field Ignoring):** Μέσω της επισήμανσης `@Mapping(target = “...”, ignore = true)`, όπως στην περίπτωση μετατροπής ενός `CourseDTO` σε `Course`, το σύστημα παραβλέπει ηθελημένα εσωτερικές λίστες (π.χ. `scheduledSlots`, `tests`). Αυτό προστατεύει το σύστημα από

απρόβλεπτες διαγραφές ή επικαλύψεις συλλογών κατά τις λειτουργίες ενημέρωσης (updates) στη βάση δεδομένων.

- **Πολυμορφική Χαρτογράφηση (Polymorphic Mapping):** Επειδή το σύστημα αξιοποιεί κληρονομικότητα (User, Student, Teacher), η διαχείριση των χρηστών υλοποιείται μέσω μιας κεντρικής ρουτίνας `toUserDTO`. Η μέθοδος αυτή αξιολογεί δυναμικά τον τύπο της οντότητας που επιστρέφει η βάση δεδομένων (μέσω `instanceof`) και ανακατευθύνει τη ροή στον κατάλληλο εξειδικευμένο μετασχηματιστή.
- **Επίλυση Σύνθετων Σχέσεων:** Σε δεδομένα που προορίζονται για πολύπλοκους πίνακες ελέγχου διαχειριστών (π.χ. `AdminLessonActivityDTO`), ο μετασχηματιστής εξάγει στοχευμένα πληροφορίες από βαθύτερες συγγενικές οντότητες (όπως `scheduledSlot.course.title`). Με αυτόν τον τρόπο, προετοιμάζει τα δεδομένα ακριβώς στη μορφή που τα απαιτεί το Front-End, μειώνοντας δραστικά την ανάγκη για επεξεργασία από την πλευρά του πελάτη (client).

3.4.4 Ανάλυση Βασικών Κλάσεων

Για την πληρέστερη κατανόηση της τεχνικής υλοποίησης, επιλέχθηκαν κομβικές κλάσεις του συστήματος, οι οποίες αναλύονται προκειμένου να αναδειχθεί η πρακτική εφαρμογή των αρχιτεκτονικών προτύπων.

Ο Ρόλος του Ελεγκτή (REST Controller) – StudentController

Στο επίπεδο διεπαφής (Web Boundary), οι ελεγκτές (Controller) λειτουργούν ως Προσαρμογείς Οδήγησης (Driving Adapters). Αποκλειστική τους ευθύνη είναι η υποδοχή των HTTP αιτημάτων, η επικύρωση των δεδομένων και η δρομολόγησή τους προς τον πυρήνα της εφαρμογής. Χαρακτηριστικό παράδειγμα αποτελεί η κλάση `StudentController`, η οποία διαχειρίζεται τις λειτουργίες των σπουδαστών.

Ενδεικτικά της αρχιτεκτονικής προσέγγισης, στον Πίνακα 2 παρατίθεται ο πυρήνας της μεθόδου εγγραφής σε προγραμματισμένο μάθημα (`enroll`). Ο πλήρης πηγαίος κώδικας του ελεγκτή, συμπεριλαμβανομένων των δηλώσεων τεκμηρίωσης OpenAPI, παρατίθεται στο Παράρτημα Ι).

Πίνακας 2: Απόσπασμα κώδικα της μεθόδου εγγραφής (`StudentController`)

```
@PostMapping("/me/enrollments")
public ResponseEntity<EnrollmentResponseResource> enroll(
    @Valid @RequestBody EnrollmentRequestResource enrollmentRequestResource,
    Principal principal)
    throws ResourceNotFoundException, IllegalArgumentException {

    // 1. Ανάθεση της επιχειρησιακής λογικής στο Use Case (Πυρήνας)
    EnrollmentResponseDTO enrollmentResponseDTO = enrollStudent.execute(
        principal.getName(),
        enrollmentRequestResource.getSlotId()
    );

    // 2. Μετασχηματισμός του DTO σε Resource για την απάντηση στο UI
    EnrollmentResponseResource enrollmentResponseResource =
```

```
EnrollmentResourceMapper.MAPPER.toEnrollmentResponseResource(  
    enrollmentResponseDTO);
```

```
// 3. Επιστροφή επιτυχούς HTTP απάντησης (200 OK)
```

```
return ResponseEntity.ok(enrollmentResponseResource);
```

```
}
```

Όπως διακρίνεται από τον παραπάνω κώδικα, η αρχιτεκτονική ροή εκτελείται σε τρία αυστηρά καθορισμένα βήματα:

1. **Ασφάλεια και Επικύρωση:** Μέσω του annotation `@Valid`, το Spring Boot επικυρώνει αυτόματα το ωφέλιμο φορτίο (payload) της αίτησης (`EnrollmentRequestResource`) πριν καν εκτελεστεί η μέθοδος. Παράλληλα, αντλείται με ασφάλεια η ταυτότητα του συνδεδεμένου χρήστη μέσω του αντικειμένου `Principal`, αποτρέποντας φαινόμενα πλαστογράφησης αιτημάτων (π.χ. προσπάθεια εγγραφής εκ μέρους άλλου μαθητή).
2. **Απομόνωση Πυρήνα:** Ο ελεγκτής δεν εκτελεί καμία επιχειρησιακή λογική (π.χ. δεν ελέγχει αν υπάρχει κενή θέση στην τάξη). Αντίθετα, “ενίεται” (injected) με τη διεπαφή `EnrollStudent` (ένα `Inbound Port` του `Application Module`) και απλώς καλεί τη μέθοδο `execute`, περνώντας της τα απαιτούμενα ορίσματα. Τυχόν παραβιάσεις κανόνων (όπως υπέρβαση χωρητικότητας) περνούν ως εξαιρέσεις (`Exceptions`) τις οποίες διαχειρίζεται κεντρικά ο `GlobalExceptionHandler`.
3. **Μετασχηματισμός Εξόδου:** Το αποτέλεσμα της λειτουργίας επιστρέφεται από τον πυρήνα ως `EnrollmentResponseDTO`. Πριν αποσταλεί στον πελάτη (`Front-End`), ο ελεγκτής αξιοποιεί τον `EnrollmentResourceMapper` για να το μετατρέψει σε `EnrollmentResponseResource`, ολοκληρώνοντας τον κύκλο της στρατηγικής “Διπλού Μετασχηματισμού” (`Double Mapping`) που αναλύθηκε προηγουμένως.

Ασφάλεια και Εξουσιοδότηση – `SecurityConfig`

Η δεύτερη κομβική κλάση που αναλύεται αφορά την καρδιά της ασφάλειας του συστήματος. Σε μια αρχιτεκτονική διαχωρισμένων επιπέδων, η προστασία των τελικών σημείων (endpoints) πρέπει να γίνεται κεντρικά, πριν το αίτημα φτάσει στους ελεγκτές. Αυτό επιτυγχάνεται μέσω της κλάσης `SecurityConfig`, η οποία υλοποιεί το πρότυπο ελέγχου πρόσβασης βάσει ρόλων (RBAC).

Στον Πίνακα 3 παρατίθεται το κρίσιμο απόσπασμα της αλυσίδας φίλτρων (`Security Filter Chain`), το οποίο καθορίζει τους κανόνες πρόσβασης. Ο πλήρης κώδικας παραμετροποίησης φιλοξενείται στο Παράρτημα I.

Πίνακας 3: Απόσπασμα κώδικα ρυθμίσεων ασφαλείας (`SecurityConfig`)

```
http.csrf(AbstractHttpConfigurer::disable)
```

```
.authorizeHttpRequest(auth -> auth
```

```
// 1. Ελεύθερη πρόσβαση σε δημόσιους πόρους (Π.χ. Σύνδεση/Εγγραφή)
```

```
.requestMatchers("/api/auth/**", "/login", "/register").permitAll()
```

```
// 2. Αυστηρός έλεγχος πρόσβασης βάσει ρόλου (RBAC)
```

```
.requestMatchers("/api/admin/**").hasAuthority("ROLE_ADMIN")
```

```
.requestMatchers("/api/teachers/**").hasAuthority("ROLE_TEACHER")
.requestMatchers("/api/students/me/**").hasAuthority("ROLE_STUDENT")

// 3. Δίκτυ Ασφαλείας (Catch-All): Όλα τα υπόλοιπα απαιτούν ταυτοποίηση
.anyRequest().authenticated()
)

// 4. Απενεργοποίηση Sessions (Αρχιτεκτονική Stateless)
.sessionManagement(session -> session.sessionCreationPolicy(
    SessionCreationPolicy.STATELESS)
)

// 5. Ένθεση του προσαρμοσμένου φίλτρου JWT
.addFilterBefore(jwtAuthenticationFilter,
    UsernamePasswordAuthenticationFilter.class
);
```

Από το παραπάνω απόσπασμα αναδεικνύονται τρεις κρίσιμες αρχιτεκτονικές αποφάσεις:

1. **Κεντρική Εξουσιοδότηση (Authorization):** Αντί να ελέγχεται ο ρόλος του χρήστη εσωτερικά σε κάθε μέθοδο (κάτι που θα προκαλούσε επανάληψη κώδικα και πιθανά κενά ασφαλείας), η πρόσβαση “κλειδώνει” στο επίπεδο του URL. Για παράδειγμα, οποιοδήποτε αίτημα προς το `/api/admin/` απορρίπτεται αυτόματα (403 Forbidden) αν το διακριτικό ασφαλείας δεν περιέχει την εξουσία `ROLE_ADMIN`.
2. **Αρχιτεκτονική Χωρίς Κατάσταση (Stateless):** Με την εντολή `SessionCreationPolicy.STATELESS`, η εφαρμογή εγκαταλείπει την παραδοσιακή διαχείριση συνεδρίων (sessions) μέσω μνήμης διακομιστή. Ο διακομιστής δεν “θυμάται” τον χρήστη μεταξύ των αιτημάτων, βελτιστοποιώντας έτσι την κλιμακωσιμότητα (scalability) του συστήματος.
3. **Αυθεντικοποίηση μέσω Διακριτικού (Token-Based Authentication):** Για να λειτουργήσει η “χωρίς κατάσταση” αρχιτεκτονική, προστίθεται ένα προσαρμοσμένο φίλτρο (`jwtAuthenticationFilter`) πριν από τα προεπιλεγμένα φίλτρα του Spring. Το φίλτρο αυτό αναλαμβάνει να αποκωδικοποιήσει το JSON Web Token (JWT) από την κεφαλίδα του HTTP αιτήματος (Authorization Header) και να εγκαθιδρύσει την ταυτότητα του χρήστη για τη διάρκεια του συγκεκριμένου και μόνο αιτήματος.

3.5 Βαθμός Ικανοποίησης Απαιτήσεων

Η αξιολόγηση ενός λογισμικού συστήματος οφείλει να συνοδεύεται από την αντικειμενική αποτίμηση του βαθμού ικανοποίησης των αρχικών απαιτήσεων. Στην παρούσα ενότητα παρουσιάζεται η συγκεντρωτική καταγραφή τόσο των βασικών λειτουργικών απαιτήσεων (Διαχείριση Χρηστών, Μαθημάτων και Προγράμματος) όσο και των επιπρόσθετων λειτουργιών εμπλουτισμού που τέθηκαν κατά τη φάση της ανάλυσης, καθώς και ο βαθμός υλοποίησής τους στο τελικό σύστημα.

3.5.1 Πίνακας Αξιολόγησης και Αιτιολόγηση Αποκλίσεων

Στον Πίνακα 4 που ακολουθεί, αποτυπώνεται η κατάσταση υλοποίησης των απαιτήσεων. Ο βαθμός ικανοποίησης διακρίνεται σε “Πλήρη” (100% υλοποίηση), “Μερική” (υλοποίηση βασικού κορμού με στοχευμένες αρχιτεκτονικές αποκλίσεις) και “Μη Υλοποιημένη” (συνειδητή εξαίρεση από το τρέχον πεδίο εφαρμογής).

Πίνακας 4: Συγκεντρωτική Αξιολόγηση Ικανοποίησης Απαιτήσεων Συστήματος

Κατηγορία Λειτουργίας	Περιγραφή Απαίτησης	Βαθμός Ικανοποίησης
A. ΔΙΑΧΕΙΡΙΣΗ ΧΡΗΣΤΩΝ		
Υποστήριξη Ρόλων	Υπαρξη 4 ρόλων (Επισκέπτης, Μαθητής, Καθηγητής, Διαχειριστής) με διακριτά δικαιώματα, μενού και προσβάσεις στο σύστημα.	Πλήρης
Εγγραφή Χρήστη	Αίτηση εγγραφής με συγκεκριμένα πεδία ανά ρόλο (π.χ. ΑΦΜ γονέα για μαθητές). Αποτροπή διπλοεγγραφών (Unique Email / Username). Απενεργοποιημένος λογαριασμός αρχικά.	Πλήρης
Δημιουργία Admin	Αποτροπή εγγραφής υπερ-χρήστη (Διαχειριστή) από το περιβάλλον διεπαφής (UI). Δημιουργία μέσω ασφαλούς τρόπου (backend / database scripts).	Πλήρης
Αυθεντικοποίηση (Login)	Σύνδεση με έλεγχο κατακερματισμένου κωδικού. Παραγωγή ασφαλούς διακριτικού πρόσβασης (JWT Token) με αυστηρό χρονικό όριο εγκυρότητας.	Πλήρης
Ενημέρωση Προφίλ	Δυνατότητα ανανέωσης προσωπικών πληροφοριών. Ενημέρωση κωδικού με αυστηρούς κανόνες πολυπλοκότητας και επιβεβαίωση μέσω του παλαιού κωδικού.	Πλήρης
Αναζήτηση & Προβολή (Χρηστών)	Αναζήτηση χρηστών (από τον διαχειριστή) και προβολή λεπτομερειών προφίλ. Δυνατότητα ενεργοποίησης (Activate) λογαριασμών από τον Admin.	Πλήρης
Διαγραφή Χρήστη	Δυνατότητα οριστικής διαγραφής. Ο Admin διαγράφει οποιονδήποτε, ενώ οι απλοί χρήστες μπορούν να αιτηθούν την διαγραφή του εαυτού τους.	Μερική
Επαναφορά Κωδικού	Μηχανισμός ερωτήσεων ασφαλείας, δημιουργία προσωρινού κωδικού μέσω email και αυτόματη απενεργοποίηση λογαριασμού σε περίπτωση μη έγκαιρης αλλαγής.	Μη Υλοποιημένη
Αποσύνδεση (Logout)	Χειροκίνητη αποσύνδεση από το σύστημα και αυτόματη αποσύνδεση μετά από αδράνεια.	Μερική

B. ΜΑΘΗΜΑΤΑ ΚΑΙ ΠΡΟΓΡΑΜΜΑ		
Δημιουργία Μαθημάτων	Ορισμός εκπαιδευτικών προγραμμάτων και δημιουργία εβδομαδιαίων τμημάτων (Scheduled Slots) αποκλειστικά από τον Διαχειριστή.	Πλήρης
Ανάθεση Διδασκαλίας	Ορισμός των κατάλληλων εκπαιδευτικών στα αντίστοιχα μαθήματα και τμήματα (Assignments) από τον Διαχειριστή.	Πλήρης
Διαχείριση Αιτημάτων Εγγραφής / Αποχώρησης	Υποβολή αιτημάτων εγγραφής (Enroll) ή αποχώρησης (Drop) από τους Σπουδαστές και έγκριση ή απόρριψή τους από τον Διαχειριστή του συστήματος.	Πλήρης
Προβολή Προγράμματος	Εξατομικευμένη προβολή του ημερήσιου και εβδομαδιαίου προγράμματος φροντιστηρίου για Μαθητές και Καθηγητές.	Πλήρης
Δηλώσεις Προτιμήσεων	Δυνατότητα των καθηγητών να δηλώνουν ψηφιακά τις προτιμήσεις τους για τις ώρες και ημέρες των μαθημάτων τους.	Μερική
Γ. ΕΜΠΛΟΥΤΙΣΜΕΝΕΣ ΛΕΙΤΟΥΡΓΙΕΣ		
Διαχείριση Απουσιών	Δυνατότητα των καθηγητών να δηλώνουν δικαιολογημένες απουσίες (Absence Requests) και αυτόματη ενημέρωση / ακύρωση του προγράμματος μαθημάτων.	Πλήρης
Δραστηριότητες (Lesson Activities)	Καταχώρηση δραστηριοτήτων, σημειώσεων και κατ'οίκον εργασιών (Homework) από τον καθηγητή ανά ημερομηνία διεξαγωγής του μαθήματος.	Πλήρης
Διαγωνίσματα (Tests)	Δημιουργία, προγραμματισμός και εκτέλεση διαγωνισμάτων από τους εκπαιδευτικούς. Καταχώρηση και προβολή των βαθμολογιών (Test Results).	Πλήρης
Οικονομική Διαχείριση	Καταχώρηση μηνιαίων πληρωμών σπουδαστών και παρακολούθηση οικονομικών υποχρεώσεων (χρέη / οφειλές).	Μη Υλοποιημένη

Αιτιολόγηση Αρχιτεκτονικών και Λειτουργικών Αποκλίσεων

Η ανάπτυξη συστημάτων επιβάλλει συχνά την αναπροσαρμογή των αρχικών απαιτήσεων προκειμένου να διασφαλιστεί η ακεραιότητα, η ασφάλεια και η βιωσιμότητα της εφαρμογής. Στο πλαίσιο αυτό, οι αποκλίσεις που καταγράφονται στον πίνακα υπήρξαν συνειδητές σχεδιαστικές επιλογές:

- **Διαγραφή Χρηστών & Ακεραιότητα Δεδομένων:** Η δυνατότητα αυτό-διαγραφής (self-deletion) από μαθητές και εκπαιδευτικούς δεν υλοποιήθηκε προκειμένου να προστατευθεί η σχεσιακή ακεραιότητα (Referential Integrity) της βάσης δεδομένων. Σε ένα εκπαιδευτικό σύστημα, η άμεση

διαγραφή ενός χρήστη δημιουργεί “ορφανά” (orphan) δεδομένα σε κρίσιμους πίνακες, όπως το ιστορικό απουσιών, οι βαθμολογίες διαγωνισμάτων και οι εγγραφές σε προγράμματα σπουδών. Αντί της διαγραφής, υιοθετείται το πρότυπο της “ήπιας διαγραφής” (Soft Delete) ή της απενεργοποίησης (Deactivation) αποκλειστικά από τον Διαχειριστή, διασφαλίζοντας την ιστορικότητα των δεδομένων του φροντιστηρίου.

- **Επαναφορά Κωδικού & Σύγχρονη Ασφάλεια:** Η υλοποίηση μηχανισμού “ερωτήσεων ασφαλείας” (security questions) και αποστολής προσωρινών κωδικών μέσω email παραλείφθηκε. Στη σύγχρονη βιβλιογραφία κυβερνοασφάλειας, η χρήση προσωπικών ερωτήσεων θεωρείται πλέον εξαιρετικά ευπαθής μέθοδος. Όπως έχει αποδειχθεί ερευνητικά (Schechter κ.ά., 2009), οι απαντήσεις είναι συχνά εύκολα προβλέψιμες από γνωστούς του χρήστη ή μέσω επιθέσεων κοινωνικής μηχανικής, ενώ η αξιοπιστία απομνημόνευσής τους φθίνει δραματικά σε βάθος χρόνου. Αντ’ αυτού, το σύστημα εστιάζει στην ασφαλή ενημέρωση κωδικού (Update Password) με χρήση του τρέχοντος, ενώ η ενσωμάτωση εξωτερικού διακομιστή SMTP για ανάκτηση μέσω email τέθηκε εκτός του τρέχοντος πεδίου εφαρμογής (scope).
- **Διαχείριση Αδράνειας (Idle Timeout):** Παρέχεται πλήρως η δυνατότητα χειροκίνητης αποσύνδεσης (Logout). Ωστόσο, η παρακολούθηση της αδράνειας του χρήστη στο επίπεδο της διεπαφής (Client-Side Idle Tracking) δεν εφαρμόστηκε. Το κενό αυτό καλύπτεται εγγενώς από την επιλεγμένη αρχιτεκτονική: λόγω της χρήσης JSON Web Tokens (JWT), κάθε διακριτικό ασφαλείας έχει αυστηρή ημερομηνία λήξης (Expiration Time). Ακόμα και αν ο χρήστης παραμείνει αδρανής και δεν αποσυνδεθεί χειροκίνητα, το σύστημα θα απορρίψει αυτόματα οποιοδήποτε επόμενο αίτημά του μόλις παρέλθει ο χρόνος ζωής του Token, εξασφαλίζοντας την απαραίτητη προστασία.
- **Δηλώσεις Προτιμήσεων (Μαθήματα vs. Ωράριο):** Η απαίτηση ικανοποιήθηκε μερικώς, καθώς δόθηκε προτεραιότητα στον καθορισμό της διδακτικής ικανότητας (Teaching Capacity). Κατά την εγγραφή τους, οι εκπαιδευτικοί δηλώνουν ψηφιακά τα μαθήματα που δύνανται και επιθυμούν να διδάξουν, παρέχοντας στον Διαχειριστή τη βάση δεδομένων για τις τελικές αναθέσεις (Assignments). Ωστόσο, η δήλωση προτιμήσεων για συγκεκριμένες ημέρες και ώρες παραλείφθηκε. Ο καθορισμός του ωρολογίου προγράμματος αποφασίστηκε να παραμείνει αυστηρά συγκεντρωτικός (centralized) από τον Διαχειριστή, προκειμένου να αποφευχθούν περίπλοκες συγκρούσεις (time conflicts) και να διασφαλιστεί η βέλτιστη κάλυψη των αναγκών του φροντιστηρίου.
- **Οικονομική Διαχείριση (Πληρωμές):** Η παρακολούθηση οικονομικών συναλλαγών αποτελεί ένα αυτόνομο, περίπλοκο υποσύστημα (ERP/Billing). Η ενσωμάτωσή του θα απαιτούσε εξειδικευμένη μοντελοποίηση (αποδείξεις, συμμόρφωση με φορολογικούς κανόνες). Αποφασίστηκε το σύστημα να παραμείνει αμιγώς εκπαιδευτικό, καταγράφοντας την οικονομική διαχείριση ως την κύρια μελλοντική επέκταση της πλατφόρμας.

4

Εγκατάσταση και Επίδειξη Συστήματος

Στο παρόν κεφάλαιο, τεκμηριώνεται η διαδικασία εκκίνησης και ελέγχου της πλατφόρμας. Αρχικά, παρατίθενται οι οδηγίες και τα τεχνικά προαπαιτούμενα για την εγκατάσταση του συστήματος σε τοπικό περιβάλλον ανάπτυξης (Local Environment). Στη συνέχεια, παρουσιάζεται η πραγματική λειτουργία του συστήματος μέσα από συγκεκριμένα σενάρια χρήσης (Use Cases), συνοδευόμενα από αντίστοιχα στιγμιότυπα οθόνης (screenshots) που αποδεικνύουν την επιτυχή διεπαφή ανθρώπου-μηχανής (HCI).

4.1 Εγκατάσταση Συστήματος

Για τη διασφάλιση της ορθής εκτέλεσης του συστήματος, κρίνεται απαραίτητη η παραμετροποίηση του περιβάλλοντος φιλοξενίας σύμφωνα με τις προδιαγραφές του πηγαίου κώδικα (όπως αυτές ορίζονται και στο αρχείο τεκμηρίωσης README.md του αποθετηρίου της εφαρμογής: <https://github.com/StathisVog/tutoring-management-system>). Επισημαίνεται ότι ο πηγαίος κώδικας είναι πλήρως διαθέσιμος στο εν λόγω αποθετήριο.

4.1.1 Προαπαιτούμενα Εγκατάστασης

Η εφαρμογή αποτελεί ένα πολύ-αρθρωτό (multi-module) έργο βασισμένο στο οικοσύστημα της Java και του Spring Boot. Για την επιτυχή μεταγλώττιση (compilation) και εκτέλεσή της, απαιτείται η προ-εγκατάσταση των παρακάτω λογισμικών:

- **Java Development Kit (JDK):** Έκδοση 17 ή νεότερη (απαραίτητη για την υποστήριξη των νεότερων χαρακτηριστικών της γλώσσας και του Spring Boot 3.x).
 - Οδηγίες εγκατάστασης: <https://adoptium.net/installation/>

- **Διαχειριστής Πακέτων (Build Tool):** Apache Maven έκδοσης 3.6 ή νεότερης, για τη διαχείριση των εξαρτήσεων (dependencies) και την κατασκευή των modules (application, domain, repository, rest, webapplication).
 - ο Οδηγίες εγκατάστασης: <https://maven.apache.org/install.html>
- **Σύστημα Ελέγχου Εκδόσεων (Version Control):** Git, προκειμένου να καταστεί δυνατή η εκφόρτωση / κλωνοποίηση (cloning) του αποθετηρίου της εφαρμογής στο τοπικό περιβάλλον του χρήστη.
 - ο Οδηγίες εγκατάστασης: <https://git-scm.com/downloads>
- **Σύστημα Διαχείρισης Βάσης Δεδομένων:** Ενεργός διακομιστής MySQL (MySQL Server). Ως προκαθορισμένη ρύθμιση, η εφαρμογή αναμένει τη βάση στην τυπική θύρα 3306. Η επιλογή αυτή υφίσταται ως προεπιλογή (default configuration) για την άμεση εκτέλεση του συστήματος, ωστόσο ο χρήστης δύναται να χρησιμοποιήσει οποιαδήποτε άλλη θύρα επιθυμεί, τροποποιώντας απλώς την αντίστοιχη μεταβλητή περιβάλλοντος (DB_PORT) κατά την εκκίνηση, χωρίς να απαιτείται καμία επέμβαση στον πηγαίο κώδικα.
 - ο Οδηγίες εγκατάστασης: <https://dev.mysql.com/doc/mysql-installation-excerpt/8.0/en/>

4.1.2 Οδηγίες Εγκατάστασης και Εκτέλεσης

Η διαδικασία εγκατάστασης έχει σχεδιαστεί ώστε να είναι πλήρως αυτοματοποιημένη, μειώνοντας τις απαιτούμενες ενέργειες από την πλευρά του διαχειριστή συστήματος. Τα βήματα εκκίνησης έχουν ως εξής:

Βήμα 1^ο: Κλωνοποίηση Αποθετηρίου (Git Clone)

Το πρώτο βήμα είναι η κλωνοποίηση του αποθετηρίου πηγαίου κώδικα της εφαρμογής στο τοπικό σύστημα μέσω Git. Ανοίγοντας το τερματικό (Command Line / Terminal), ο χρήστης εκτελεί:

```
git clone https://github.com/StathisVog/tutoring-management-system.git
```

Βήμα 2^ο: Διαμόρφωση Εφαρμογής

Το σύστημα κάνει χρήση του μηχανισμού Hibernate (ORM) για την αυτόματη διαχείριση της βάσης. Κατά την εκκίνηση, το σύστημα επικοινωνεί με τον διακομιστή MySQL χρησιμοποιώντας τα προεπιλεγμένα διαπιστευτήρια (χρήστης: root, χωρίς κωδικό πρόσβασης)¹. Εφόσον η σύνδεση επιτευχθεί, το σύστημα:

1. Δημιουργεί αυτόματα το σχήμα (schema) με την ονομασία *tutoring_management* (μέσω της ρύθμισης `createDatabaseIfNotExist=true`).
2. Δημιουργεί όλους τους απαραίτητους πίνακες σύμφωνα με τις οντότητες της εφαρμογής.
3. Εκτελεί το αρχείο `data.sql`, το οποίο τροφοδοτεί τη βάση με τα αρχικά δοκιμαστικά δεδομένα και δημιουργεί τον προεπιλεγμένο λογαριασμό του Διαχειριστή (Admin) του συστήματος / εφαρμογής, (όχι της βάσης). Προς διασφάλιση των βέλτιστων πρακτικών ασφαλείας (security best

¹ Τα διαπιστευτήρια της βάσης δεδομένων, καθώς και άλλες παραμέτρους, ο χρήστης έχει την ευκαιρία να τα παραμετροποιήσει όπως επιθυμεί ώστε να ταιριάζουν με το σύστημά του. Η αλλαγή γίνεται με ασφάλεια στο αρχείο ρυθμίσεων, το ολοκληρωμένο τοπικό μονοπάτι του οποίου είναι:

`webapplication/src/main/resources/application.yaml`

(διαθέσιμο και διαδικτυακά στο αποθετήριο: <https://github.com/StathisVog/tutoring-management-system/blob/main/webapplication/src/main/resources/application.yaml>).

practices), παρότι τα δεδομένα αυτά προορίζονται για τοπική ανάπτυξη, ο προεπιλεγμένος κωδικός του διαχειριστή καταχωρείται στη βάση δεδομένων ήδη κρυπτογραφημένος (BCrypt Hashed) και όχι σε μορφή απλού κειμένου (plain text).

Βήμα 3ο: Μεταγλώττιση του Πηγαίου Κώδικα (Build)

Μέσω του τερματικού, ο χρήστης πλοηγείται στον ριζικό (root) φάκελο του έργου και εκτελεί την εντολή κατασκευής του Maven:

```
cd tutoring-management-system  
mvn clean install
```

Η εντολή αυτή κατεβάζει όλες τις απαραίτητες βιβλιοθήκες, περνάει επιτυχώς τα αρχιτεκτονικά τεστ (ArchUnit Tests) και δημιουργεί τα εκτελέσιμα αρχεία της εφαρμογής.

Βήμα 4ο: Εκκίνηση της Εφαρμογής

Αφού ολοκληρωθεί επιτυχώς η μεταγλώττιση, ο χρήστης μεταβαίνει στον φάκελο του κυρίου module (webapplication) και εκκινεί τον ενσωματωμένο διακομιστή (embedded Tomcat) και άρα και την εφαρμογή μας που αυτός εξυπηρετεί:

```
cd webapplication  
mvn spring-boot:run
```

Βήμα 5ο: Πρόσβαση στο Σύστημα

Μόλις η εφαρμογή εκκινήσει, οι προσφερόμενες υπηρεσίες είναι άμεσα προσβάσιμες μέσω οποιουδήποτε περιηγητή ιστού (Web Browser):

- **Περιβάλλον Διεπαφής Χρήστη (Web UI):** Πρόσβαση στο γραφικό περιβάλλον της εφαρμογής μέσω της διεύθυνσης <http://localhost:8080>
- **Τεκμηρίωση API (Swagger UI):** Πρόσβαση στην τεκμηρίωση και το περιβάλλον δοκιμών των REST Endpoints μέσω της διεύθυνσης <http://localhost:8080/docs>

4.2 Επίδειξη Συστήματος (Σενάρια Χρήσης & Στιγμιότυπα Οθόνης)

Στην παρούσα ενότητα παρουσιάζεται η διεπαφή της εφαρμογής μέσα από τρία βασικά σενάρια χρήσης (Use Cases). Κάθε σενάριο αποτυπώνει την αλληλεπίδραση των διαφορετικών ρόλων του συστήματος (Διαχειριστής, Εκπαιδευτικός, Σπουδαστής) με τις βασικές λειτουργίες του φροντιστηρίου, αποδεικνύοντας την ορθή εφαρμογή των επιχειρησιακών κανόνων (business rules) και των μηχανισμών ασφαλείας.

4.2.1 Σενάριο 1: Διαχείριση Λογαριασμών και Αυθεντικοποίηση

Το πρώτο σενάριο εστιάζει στον κύκλο ζωής ενός λογαριασμού χρήστη, από την αρχική αίτηση εγγραφής έως την επιτυχή αυθεντικοποίησή του. Στόχος του σεναρίου είναι να αναδείξει τη δυναμική συμπεριφορά των φορμών εγγραφής και την αυστηρή επιβολή του κανόνα ασφαλείας που απαιτεί την έγκριση των νέων χρηστών από τον Διαχειριστή.

Βήμα 1ο: Αίτηση Εγγραφής Νέου Εκπαιδευτικού

Ο επισκέπτης μεταβαίνει στη σελίδα εγγραφής και επιλέγει τον ρόλο του Καθηγητή (Teacher). Το σύστημα ανταποκρίνεται δυναμικά, εμφανίζοντας τα εξειδικευμένα πεδία που αφορούν τον συγκεκριμένο

ρόλο. Ο χρήστης συμπληρώνει τα προσωπικά του στοιχεία και δηλώνει την ειδικότητά του, επιλέγοντας παράλληλα από τη διαθέσιμη λίστα τα γνωστικά αντικείμενα (μαθήματα) που επιθυμεί να διδάξει. Το πεδίο του κωδικού πρόσβασης (Password) ελέγχεται σε πραγματικό χρόνο για την πολυπλοκότητά του.

Create an Account

Join the Tutoring Management System

I am registering as a: *

Teacher

Username *

ktsigklifisis

Email Address *

ktsigklifisis@example.com

Password *

Your password strength Strong

Full Name *

Kostas Tsigklifisis

Home Address *

48 Kifissias Ave, Athens

Specialty (e.g., Mathematician, Physicist) *

Mathematician

Courses you can teach *

☒ Algebra - High School 1st Year
☒ Mathematics - High School 3rd Year (Advanced)
☐ Physics - High School 2nd Year (Advanced)
☐ Chemistry - High School 3rd Year (Advanced)
☐ Modern Greek Language & Literature - High School 3rd Year
☐ Ancient Greek - High School 2nd Year (Advanced)
☐ History - High School 3rd Year (Advanced)
☐ Biology - High School 3rd Year (Health Sciences)

Short Biography

A seasoned educator with 10+ years of experience in Panhellenic exam preparation.

Register

Σχήμα 18: Συμπληρωμένη δυναμική φόρμα εγγραφής νέου εκπαιδευτικού με δήλωση διδακτικής ικανότητας

Βήμα 2^ο: Ενημέρωση Κατάστασης Λογαριασμού

Μετά την επιτυχή υποβολή της φόρμας, το σύστημα αποθηκεύει με ασφάλεια τα δεδομένα (κρυπτογραφώντας τον κωδικό πρόσβασης), αλλά θέτει τον λογαριασμό σε κατάσταση αδράνειας (INACTIVE). Μέσω ενός ενημερωτικού μηνύματος, ο χρήστης πληροφορείται ότι η εγγραφή του ολοκληρώθηκε, αλλά απαιτείται η τελική έγκριση από τη διαχείριση προτού του επιτραπεί η είσοδος στο σύστημα (δείτε Σχήμα 19).

Create an Account

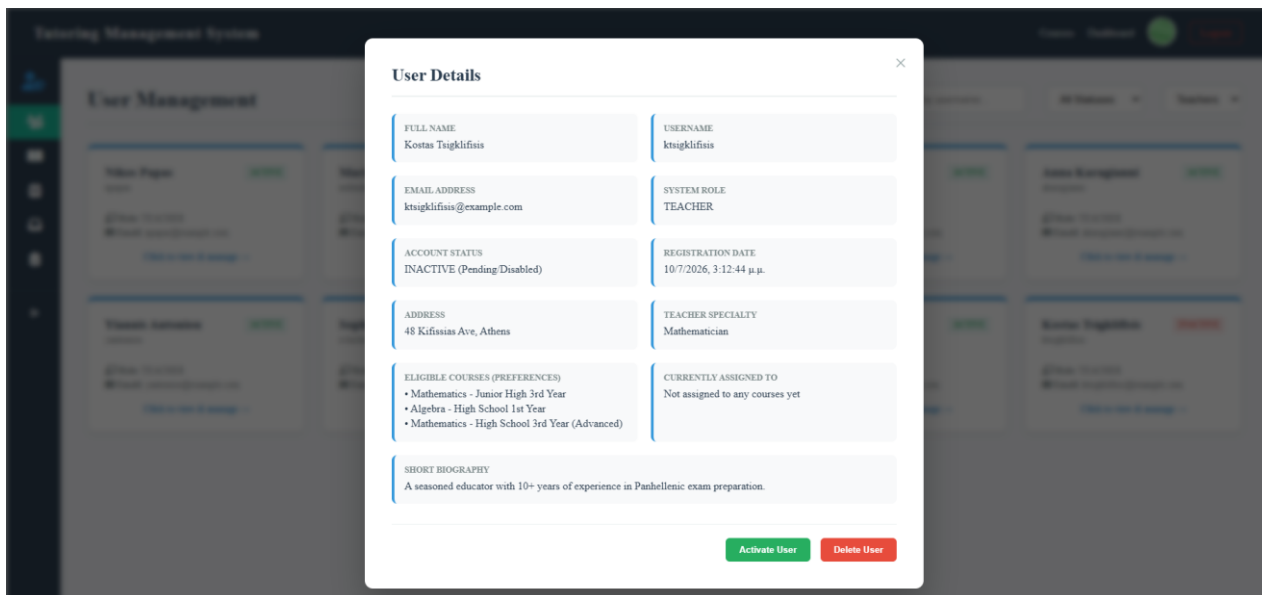
Join the Tutoring Management System

Registration successful! Your account has been created but is currently INACTIVE. Please wait for an Administrator to approve it before logging in.

Σχήμα 19: Ενημερωτικό μήνυμα συστήματος σχετικά με την ανενεργή κατάσταση του νέου λογαριασμού

Βήμα 3^ο: Έγκριση και Ενεργοποίηση από τον Διαχειριστή

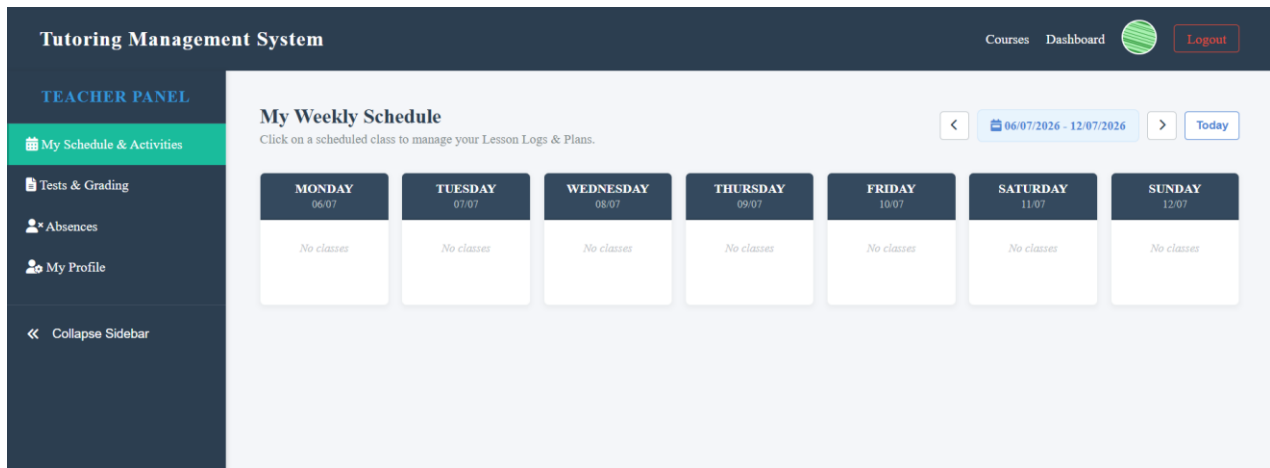
Ο Διαχειριστής του συστήματος (Admin) αυθεντικοποιείται και μεταβαίνει στον πίνακα Διαχείρισης Χρηστών (User Management). Εντοπίζει τη νέα εγγραφή και ανοίγει την αναλυτική καρτέλα λεπτομερειών (User Details). Από αυτό το σημείο, έχοντας πλήρη εικόνα των στοιχείων και των προτιμήσεων του εκπαιδευτικού, προχωρά στην ενεργοποίηση του λογαριασμού πατώντας το κουμπί “Activate User” (δείτε Σχήμα 20).



Σχήμα 20: Περιβάλλον Διαχειριστή - Καρτέλα λεπτομερειών και ενεργοποίηση εκκρεμούς λογαριασμού χρήστη

Βήμα 4^ο: Αυθεντικοποίηση και Πρόσβαση στο Ταμπλό

Ο νέος εκπαιδευτικός, διαθέτοντας πλέον έναν πλήρως ενεργό λογαριασμό, μεταβαίνει στη σελίδα σύνδεσης. Με την εισαγωγή των διαπιστευτηρίων του, το σύστημα παράγει και επικυρώνει το διακριτικό ασφαλείας (JWT). Ο χρήστης ανακατευθύνεται αυτόματα στο εξατομικευμένο ταμπλό του (Teacher Panel), όπου έχει άμεση πρόσβαση στο εβδομαδιαίο του πρόγραμμα και στο διακριτό μενού των λειτουργιών που αντιστοιχούν στον ρόλο του (δείτε Σχήμα 21). Επειδή ο εκπαιδευτικός είναι νέος, δεν του έχει ανατεθεί ακόμα τμήμα μαθήματος, οπότε το πρόγραμμά του φαίνεται κενό.



Σχήμα 21: Επιτυχής αυθεντικοποίηση και ανακατεύθυνση στο αρχικό ταμπλό (Dashboard) του Εκπαιδευτικού

4.2.2 Σενάριο 2: Δημιουργία και Προγραμματισμός Μαθημάτων (Admin)

Το δεύτερο σενάριο εστιάζει στις κεντρικές λειτουργίες διαχείρισης του εκπαιδευτικού προγράμματος (Curriculum Management). Παρουσιάζεται η ικανότητα του Διαχειριστή (Admin) να δημιουργεί νέα γνωστικά αντικείμενα και να ολοκληρώνει τον κύκλο της διδακτικής ενότητας, αναθέτοντας το μάθημα σε εκπαιδευτικούς που το έχουν ήδη δηλώσει στις προτιμήσεις τους.

Βήμα 1^ο: Ταυτοποίηση Διαχειριστή (Admin Login)

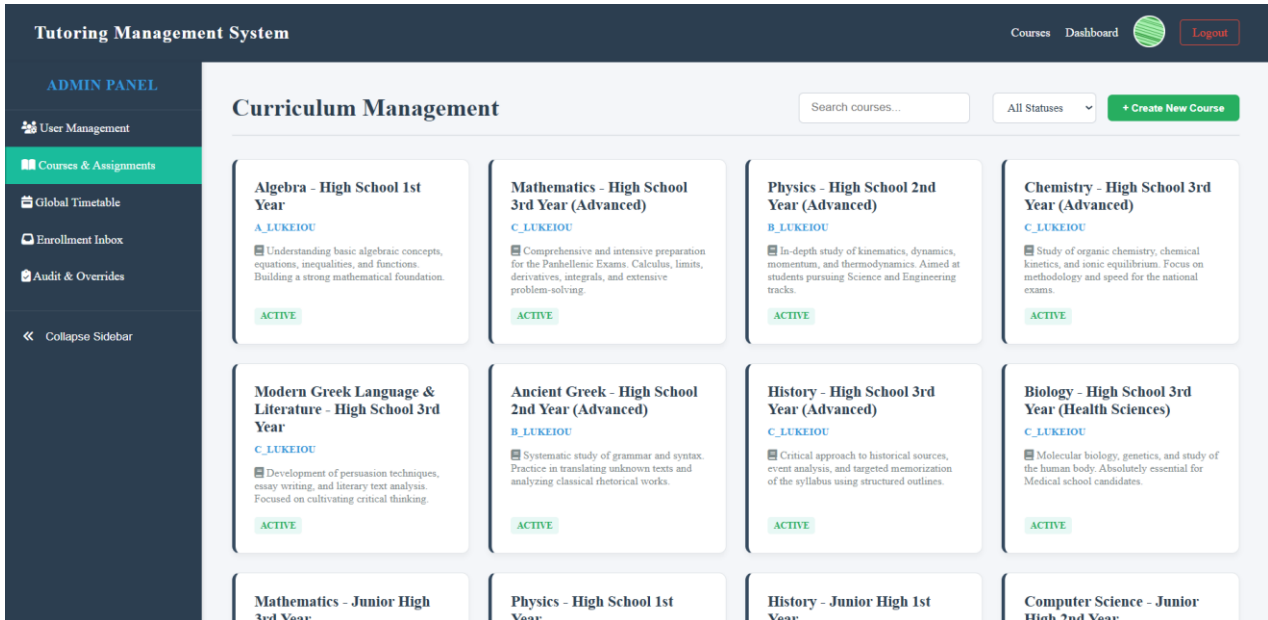
Για την πρόσβαση στις κρίσιμες λειτουργίες διαχείρισης, απαιτείται η είσοδος στο σύστημα με λογαριασμό που διαθέτει τα μέγιστα δικαιώματα (Role: ADMIN). Ο Διαχειριστής εισάγει τα διαπιστευτήριά του στην κεντρική πύλη αυθεντικοποίησης του συστήματος.

The image shows a login form titled 'Login to the System'. Below the title is the instruction 'Enter your credentials to access your account'. There are two input fields: 'Username' and 'Password'. Below these fields is a large blue button labeled 'Login'. At the bottom of the form, there is a link that says 'Don't have an account? Register here'.

Σχήμα 22: Κεντρική φόρμα αυθεντικοποίησης (Login) για την είσοδο του Διαχειριστή

Βήμα 2º: Διαχείριση Γνωστικών Αντικειμένων (Courses)

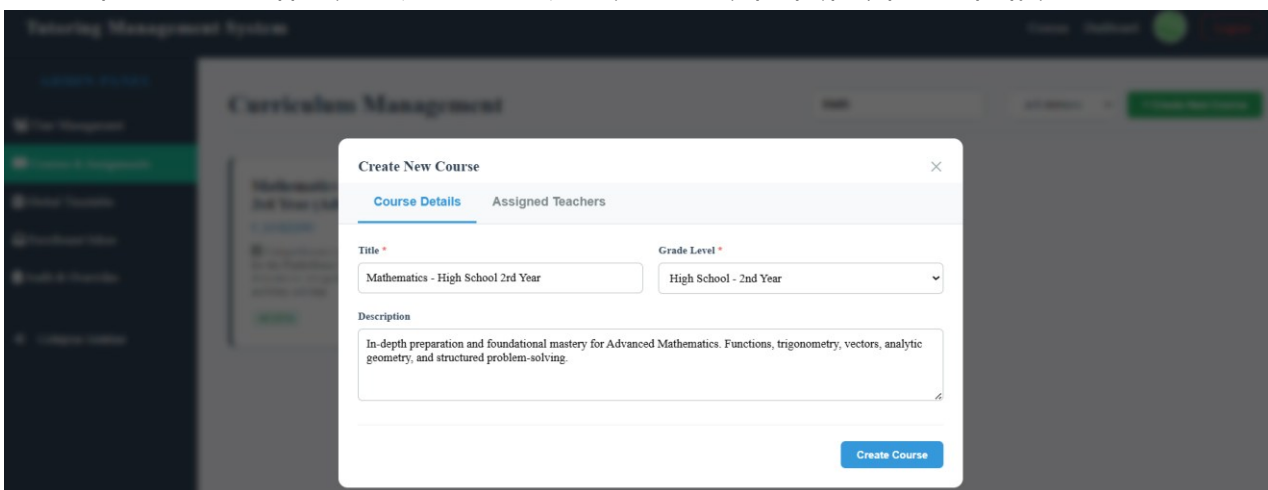
Μετά την επιτυχή είσοδο, ο Διαχειριστής πλοηγείται μέσω του πλευρικού μενού (Sidebar) στην ενότητα “Courses & Assignments” (δείτε Σχήμα 23). Εκεί προβάλλεται το σύνολο των διαθέσιμων μαθημάτων του φροντιστηρίου, οργανωμένα σε διακριτές κάρτες (cards) για βέλτιστη οπτική αναγνώριση. Από αυτήν την οθόνη, επιλέγεται η ενέργεια “+ Create New Course” (πάνω δεξιά) για την εισαγωγή ενός νέου μαθήματος.



Σχήμα 23: Ο κεντρικός πίνακας ελέγχου (Dashboard) για τη διαχείριση των μαθημάτων του φροντιστηρίου

Βήμα 3º: Δημιουργία Νέου Μαθήματος

Το σύστημα εμφανίζει ένα αναδυόμενο παράθυρο (Modal) πολλαπλών καρτελών (δείτε Σχήμα 24). Στην πρώτη καρτέλα (Course Details), ο Διαχειριστής καταχωρεί τα μεταδεδομένα του μαθήματος, όπως τον τίτλο, την εκπαιδευτική βαθμίδα (Grade Level) και μια σύντομη περιγραφή του περιεχομένου του.

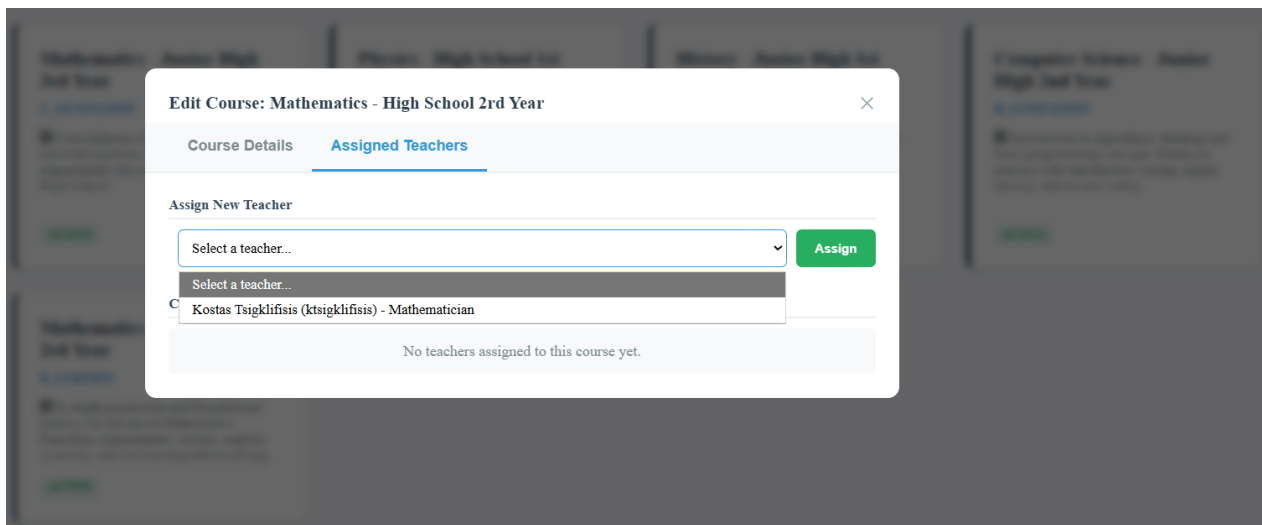


Σχήμα 24: Φόρμα εισαγωγής μεταδεδομένων για τη δημιουργία νέου μαθήματος

Βήμα 4º: Δυναμικό Φιλτράρισμα και Ανάθεση (Assignment)

Μεταβαίνοντας στη δεύτερη καρτέλα (Assigned Teachers), ο Διαχειριστής καλείται να ορίσει τον εκπαιδευτικό που θα αναλάβει τη διδασκαλία. Στο σημείο αυτό αναδεικνύεται η επιχειρησιακή λογική (business logic) του συστήματος: η αναπτυσσόμενη λίστα (dropdown) δεν προβάλλει όλους τους χρήστες ανεξαιρέτως, αλλά φιλτράρει δυναμικά και εμφανίζει αποκλειστικά τους εκπαιδευτικούς που έχουν

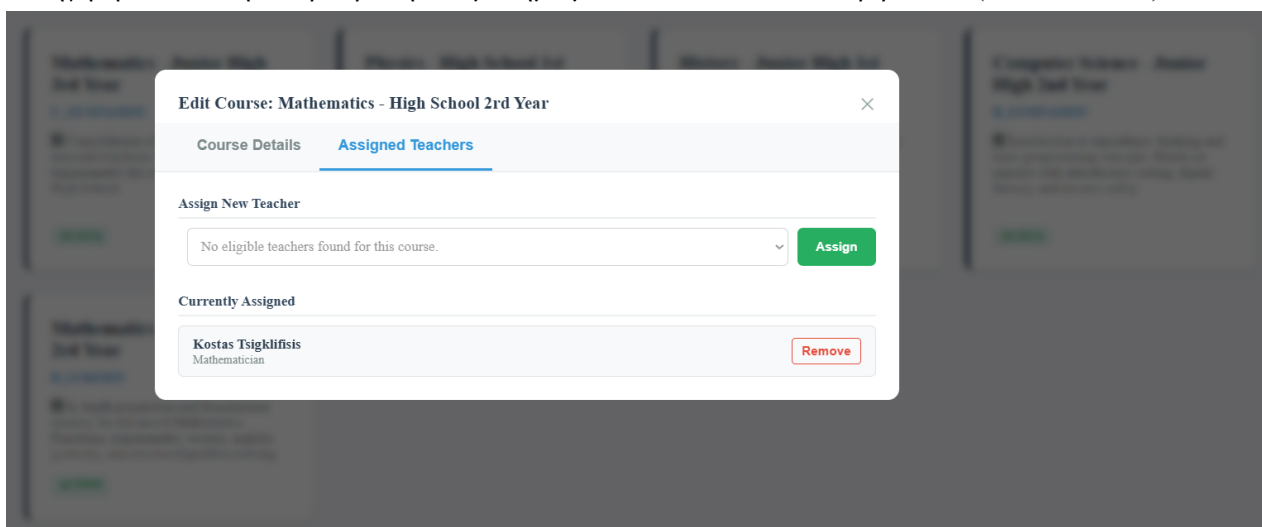
δηλώσει το συγκεκριμένο μάθημα στην ικανότητα διδασκαλίας τους κατά την εγγραφή τους (όπως φάνηκε στο Σενάριο 1).



Σχήμα 25: Δυναμική λίστα επιλογής εκπαιδευτικού, φιλτραρισμένη βάσει των προτιμήσεων διδασκαλίας

Βήμα 5^ο: Ολοκλήρωση Ανάθεσης

Ο Διαχειριστής επιλέγει τον επιθυμητό εκπαιδευτικό και ολοκληρώνει τη διαδικασία (Assign) (δείτε Σχήμα 26). Το σύστημα ενημερώνει ακαριαία τη σχεσιακή βάση δεδομένων, συνδέοντας την οντότητα του Μαθήματος με αυτή του Εκπαιδευτικού. Η ανάθεση είναι πλέον ορατή, επιτρέποντας στον συγκεκριμένο καθηγητή να αποκτήσει πρόσβαση στο μάθημα μέσα από το δικό του περιβάλλον (Teacher Panel).



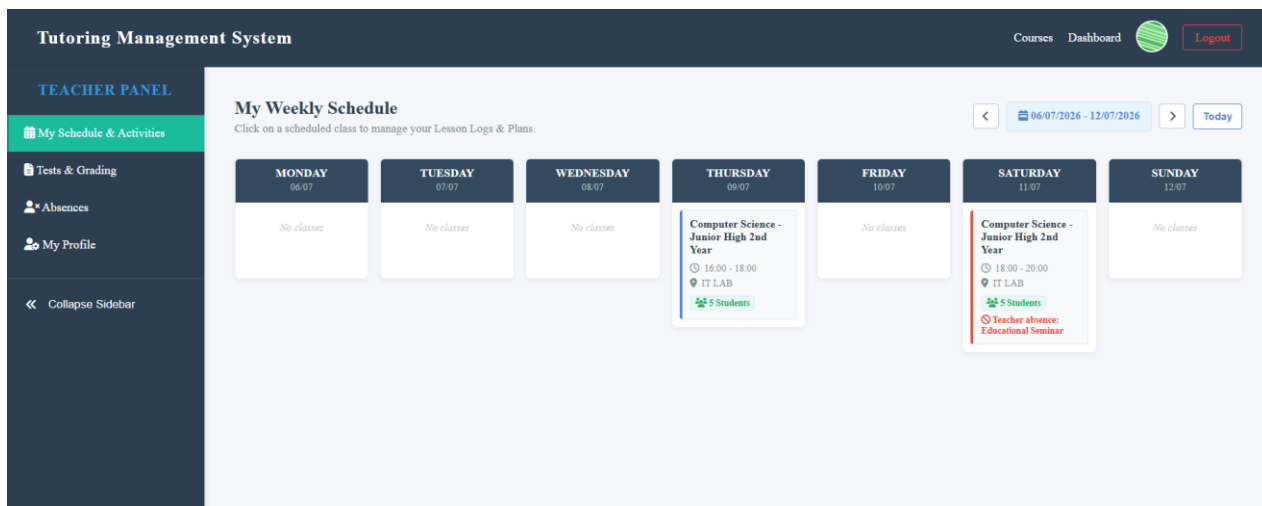
Σχήμα 26: Επιτυχής ανάθεση (assignment) του μαθήματος στον επιλεγμένο εκπαιδευτικό

4.2.3 Σενάριο 3: Προβολή Προγράμματος (Μαθητής / Καθηγητής)

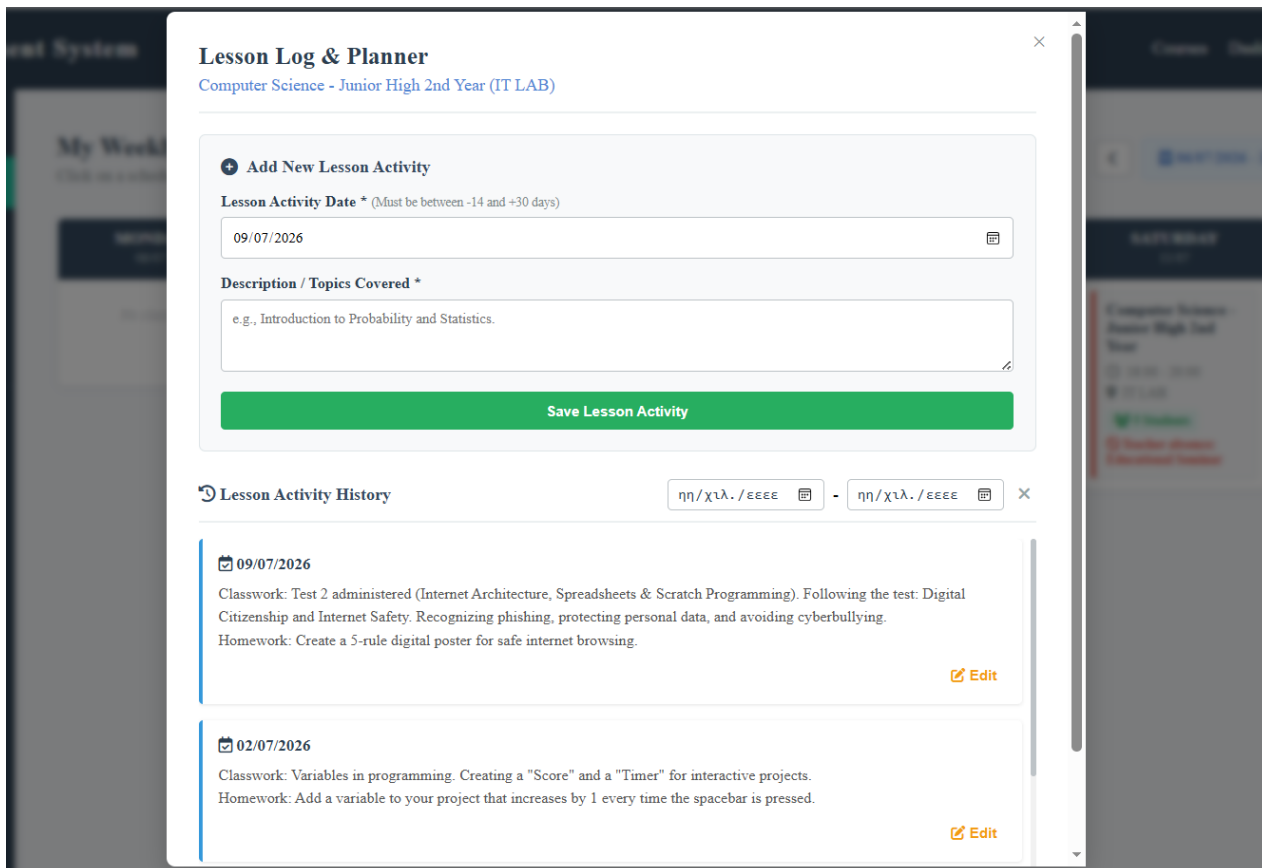
Το τρίτο και τελευταίο σενάριο εστιάζει στην εμπειρία του τελικού χρήστη (End-User Experience) και αποδεικνύει τον συγχρονισμό πραγματικού χρόνου (real-time synchronization) των δεδομένων. Παρουσιάζεται η προβολή του εβδομαδιαίου προγράμματος από την οπτική γωνία τόσο του Εκπαιδευτικού όσο και του Σπουδαστή, αναδεικνύοντας παράλληλα τους μηχανισμούς ελέγχου πρόσβασης (Role-Based Access Control) και την επιχειρησιακή λογική (business logic) αποτροπής σφαλμάτων.

Βήμα 1^ο: Προβολή Προγράμματος και Δραστηριοτήτων (Οπτική Εκπαιδευτικού)

Ο εκπαιδευτικός, μεταβαίνοντας στο ταμπλό του, έχει πλήρη εικόνα των εβδομαδιαίων υποχρεώσεών του. Στο πρόγραμμα αποτυπώνονται με σαφήνεια οι προγραμματισμένες παραδόσεις, καθώς και οι τυχόν δηλωμένες απουσίες του (δείτε Σχήμα 27). Επιλέγοντας ένα ενεργό μάθημα (π.χ. της 09/07), του παρέχεται η δυνατότητα να προσθέσει ή να επεξεργαστεί τη ροή του μαθήματος (Lesson Activities), καταχωρώντας τη διδαχθείσα ύλη και τις κατ' οίκον εργασίες (Homework) (δείτε Σχήμα 28).



Σχήμα 27: Το εβδομαδιαίο ημερολόγιο του εκπαιδευτικού, εμφανίζοντας ενεργά και ακυρωμένα (λόγω απουσίας) μαθήματα



Σχήμα 28: Αναλυτική προβολή και διαχείριση των δραστηριοτήτων μαθήματος (Lesson Log & Planner) από τον εκπαιδευτικό

Βήμα 2º: Επιχειρησιακή Λογική και Αποτροπή Διενέξεων (Conflict Prevention)

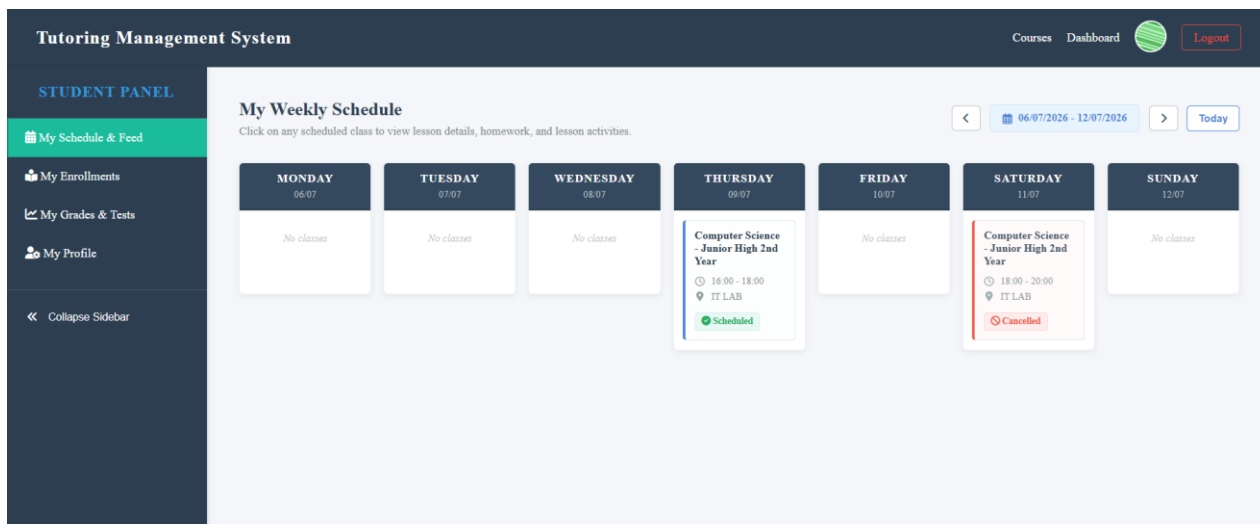
Μια κρίσιμη λειτουργία προστασίας της ακεραιότητας των δεδομένων αποκαλύπτεται όταν ο εκπαιδευτικός επιχειρεί να διαχειριστεί ένα μάθημα για το οποίο έχει δηλώσει απουσία (π.χ. της 11/07). Το σύστημα αναγνωρίζει την ακύρωση του μαθήματος και απενεργοποιεί δυναμικά (disable) τη φόρμα καταχώρησης νέων δραστηριοτήτων, αποτρέποντας τη δημιουργία λογικών σφαλμάτων στη βάση δεδομένων (π.χ. καταχώρηση ύλης σε ημέρα απουσίας).

The screenshot displays the 'Lesson Log & Planner' interface for 'Computer Science - Junior High 2nd Year (IT LAB)'. A yellow warning banner at the top states: '⚠ This lesson is cancelled (Teacher absence: Educational Seminar). Adding new activities is disabled.' Below this, the 'Add New Lesson Activity' form is shown with the date '11/07/2026' and a description 'e.g., Introduction to Probability and Statistics.' The 'Save Lesson Activity' button is present but disabled. The 'Lesson Activity History' section below shows two previous activities: one from 04/07/2026 and another from 27/06/2026, each with details on classwork and homework.

Σχήμα 29: Δυναμικό κλείδωμα της φόρμας δραστηριοτήτων με σχετικό προειδοποιητικό μήνυμα, λόγω δηλωμένης απουσίας

Βήμα 3º: Προβολή Προγράμματος (Οπτική Σπουδαστή)

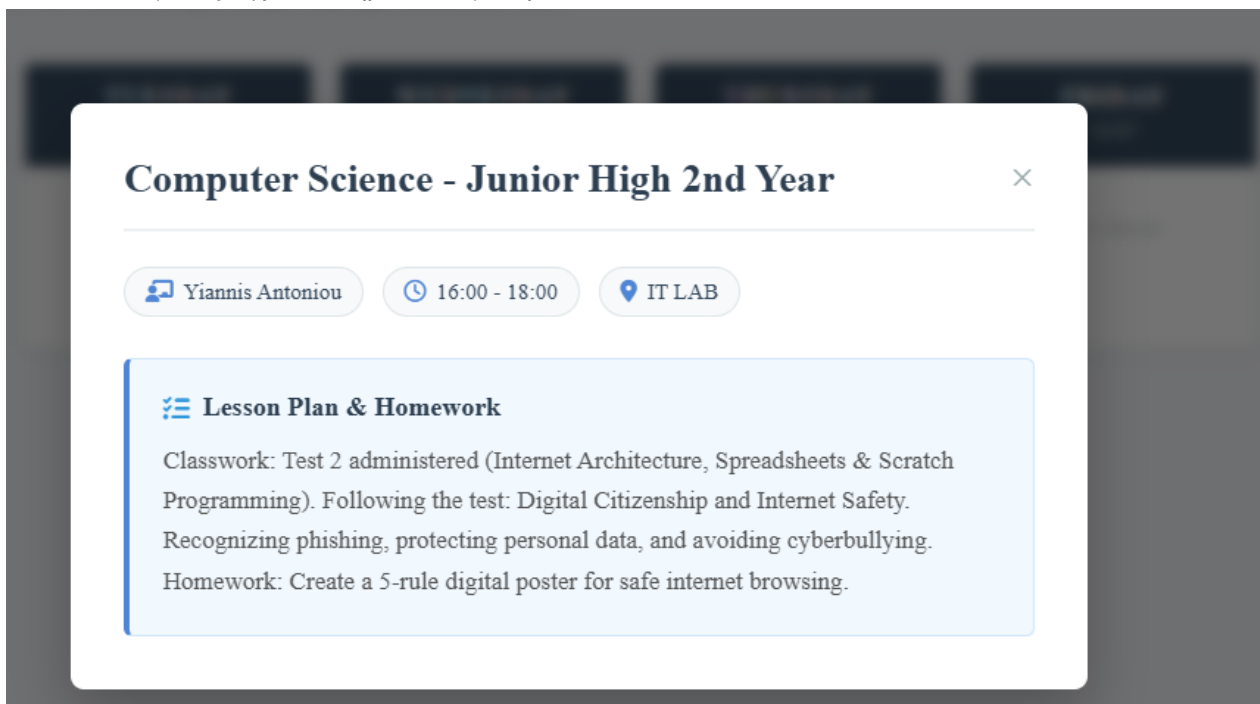
Στον αντίποδα, ένας σπουδαστής που είναι εγγεγραμμένος στο συγκεκριμένο τμήμα συνδέεται στο δικό του ταμπλό. Το σύστημα αντλεί τα δεδομένα της βάσης και του προβάλλει ακριβώς την ίδια, ενημερωμένη πραγματικότητα. Όπως φαίνεται από το Σχήμα 30, το μάθημα της 09/07 εμφανίζεται κανονικά ως “Scheduled”, ενώ το μάθημα της 11/07 φέρει αυτόματα την ένδειξη “Cancelled”, ενημερώνοντας άμεσα τον μαθητή.



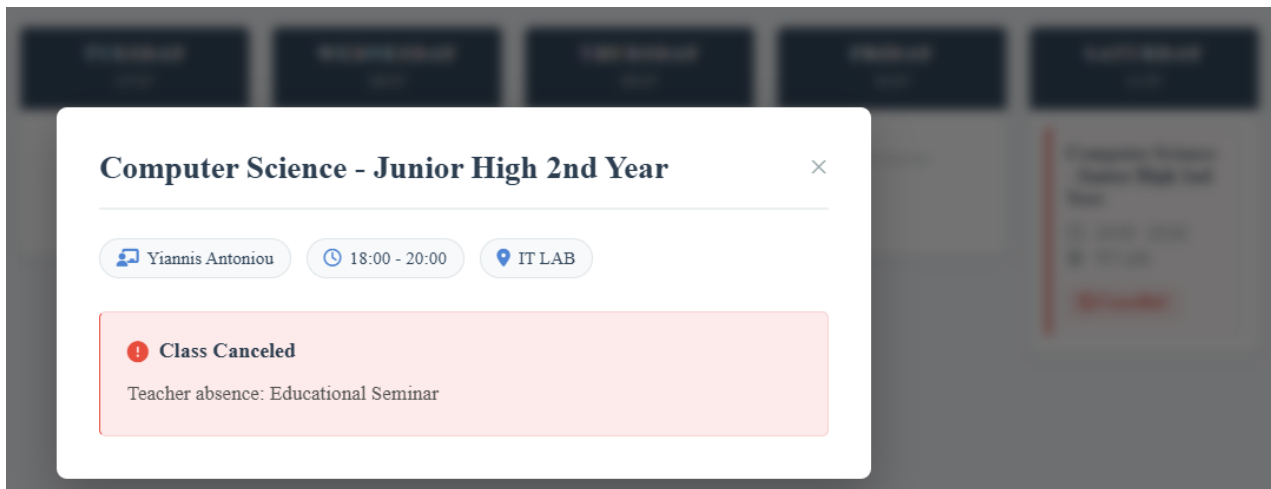
Σχήμα 30: Προβολή του ημερολογίου από την πλευρά του σπουδαστή, σε πλήρη συγχρονισμό με τα δεδομένα του εκπαιδευτικού

Βήμα 4ο: Εξατομικευμένη Ενημέρωση (Read-Only Access)

Ο σπουδαστής, διαθέτοντας αυστηρά δικαιώματα ανάγνωσης (Read-Only), μπορεί να αλληλοεπιδράσει με το πρόγραμμά του. Πατώντας στο ενεργό μάθημα, ενημερώνεται αναλυτικά για το Homework που του ανατέθηκε από τον εκπαιδευτικό. Αντίστοιχα, επιλέγοντας το ακυρωμένο μάθημα, ενημερώνεται αυτοματοποιημένα από το σύστημα για την ακριβή αιτία της ακύρωσης, εξασφαλίζοντας την απρόσκοπτη επικοινωνία μεταξύ φροντιστηρίου και μαθητών.



Σχήμα 31: Προβολή των δραστηριοτήτων και των κατ' οίκον εργασιών (Homework) από τον σπουδαστή



Σχήμα 32: Ενημέρωση του σπουδαστή για την αιτία ακύρωσης της προγραμματισμένης παράδοσης

5

Συμπεράσματα και Μελλοντική Εργασία

Στο παρόν και τελευταίο κεφάλαιο συνοψίζονται τα βασικά ευρήματα και η συνεισφορά της διπλωματικής εργασίας. Επιπλέον, καταγράφεται η προσωπική αποκομισθείσα εμπειρία από τον πλήρη κύκλο ανάπτυξης του λογισμικού, ενώ προτείνονται συγκεκριμένες και στοχευμένες κατευθύνσεις για τη μελλοντική επέκταση της πλατφόρμας.

5.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία πέτυχε τον αρχικό της στόχο: την ανάπτυξη ενός ολοκληρωμένου, ασφαλούς και σύγχρονου πληροφοριακού συστήματος ικανού να υποστηρίξει τις περίπλοκες εσωτερικές λειτουργίες ενός Φροντιστηρίου Μέσης Εκπαίδευσης. Μέσω της δημιουργίας διακριτών ρόλων, αυτοματοποιημένης διαχείρισης προγράμματος και πλήρους ιχνηλασιμότητας δεδομένων (δραστηριότητες, απουσίες, βαθμολογίες), το σύστημα παρέχει μια ψηφιακή λύση που καταργεί τον κατακερματισμό των παραδοσιακών μεθόδων.

Αποκομισθείσα Εμπειρία

Πέρα από την προφανή συνεισφορά σε επίπεδο λογισμικού, η υλοποίηση της παρούσας εργασίας αποτέλεσε μια πολύτιμη διαδρομή επαγγελματικής και τεχνικής ωρίμανσης.

Σε **επίπεδο αρχιτεκτονικής**, εμπεδώθηκε πλήρως η αξία της συγγραφής καθαρού και διατηρήσιμου κώδικα (Clean Code). Η αυστηρή υιοθέτηση της Εξαγωνικής Αρχιτεκτονικής (Hexagonal Architecture / Ports and Adapters) σε συνδυασμό με τη χρήση αρχιτεκτονικών ελέγχων (ArchUnit tests) δίδαξε πρακτικά πώς διασφαλίζεται το “Separation of Concerns” (Διαχωρισμός Αρμοδιοτήτων). Η γνώση ότι το σύστημα προστατεύεται από αρχιτεκτονικές αλλοιώσεις (architectural decay), ανεξάρτητα από μελλοντικές προσθήκες, αποτελεί θεμέλιο λίθο για κάθε επαγγελματία Μηχανικό Λογισμικού.

Σε **τεχνικό επίπεδο**, η υλοποίηση αποτέλεσε την πρώτη σε βάθος τριβή με μηχανισμούς σύγχρονης κυβερνοασφάλειας, εμβαθύνοντας στη διαχείριση ταυτότητας (Identity Management), τον αυστηρό διαχωρισμό ρόλων και την πρακτική εφαρμογή αυθεντικοποίησης χωρίς κατάσταση (stateless) μέσω JSON Web Tokens (JWT) και Cookies.

Τέλος, σε **επιχειρησιακό επίπεδο (Domain Knowledge)**, η έρευνα και η ανάλυση οδήγησαν σε βαθιά κατανόηση του πώς λειτουργεί εσωτερικά ένας εκπαιδευτικός οργανισμός. Η νοοτροπία μετατοπίστηκε από την απλή ικανοποίηση μιας τεχνικής απαίτησης στον εξονυχιστικό σχεδιασμό χαρακτηριστικών (feature planning) που λύνουν πραγματικά προβλήματα (αντιμετωπίζοντας λογικές ασυνέπειες και αποφεύγοντας συγκρούσεις δεδομένων, όπως τα time conflicts).

5.2 Μελλοντική Εργασία

Παρά την πλήρη λειτουργικότητα της πλατφόρμας και την κάλυψη των απαιτήσεων, ένα λογισμικό (application) βρίσκεται σε διαρκή εξέλιξη. Με γνώμονα την περαιτέρω αναβάθμιση του συστήματος, προτείνονται οι ακόλουθες λειτουργικές επεκτάσεις:

1. Υποσύστημα Οικονομικής Διαχείρισης (ERP Integration)

Η πρωταρχική μελλοντική επέκταση αφορά την ανάπτυξη μιας υπομονάδας για την παρακολούθηση των μηνιαίων συνδρομών και πληρωμών των σπουδαστών. Θα μπορούσε σε πρώτη φάση να υλοποιηθεί ένα λειτουργικό πρωτότυπο ελέγχου χρεών και οφειλών, προκειμένου να δοκιμαστεί η αντοχή και η ευχρηστία του πριν την πλήρη ενσωμάτωση του στο σύστημα.

2. Μηχανισμός Αιτημάτων Αυτό-Διαγραφής Λογαριασμών

Για την ενίσχυση του ελέγχου των χρηστών στα προσωπικά τους δεδομένα, σχεδιάζεται η προσθήκη λειτουργικότητας υποβολής “Αιτήματος Διαγραφής” (Account Deletion Request) από το προφίλ των εκπαιδευτικών και των σπουδαστών. Το αίτημα θα προωθείται στον Διαχειριστή για έγκριση, ώστε να εκτελείται συντεταγμένα η “ήπια διαγραφή” (Soft Delete) χωρίς να πλήττεται η ακεραιότητα της βάσης (orphan records).

3. Δυναμική Διαχείριση Συνεδρίας (Dynamic Session Management)

Στο πλαίσιο ενίσχυσης της ασφάλειας, η τρέχουσα λήξη των JWT tokens (server-side expiration) μπορεί να συμπληρωθεί με έναν μηχανισμό ελέγχου αδράνειας στη διεπαφή (client-side timeout). Εφόσον η πλατφόρμα δεν ανιχνεύσει αλληλεπίδραση (π.χ. κίνηση ποντικιού, πληκτρολόγηση) για 15 λεπτά, το σύστημα θα αποσυνδέει (Logout) τον χρήστη αυτόματα και θα διαγράφει το Token, αποτρέποντας την έκθεση δεδομένων σε κοινόχρηστους υπολογιστές.

4. Σύγχρονη και Ασύγχρονη Επικοινωνία (Live Chatrooms)

Προτείνεται η δημιουργία ενός εσωτερικού υποσυστήματος ανταλλαγής μηνυμάτων (messaging). Κάθε εβδομαδιαίο τμήμα (Class) θα διαθέτει ένα αποκλειστικό, ασφαλές ψηφιακό δωμάτιο (Chatroom) στο οποίο μαθητές και εκπαιδευτικοί θα μπορούν να αλληλοεπιδρούν σε πραγματικό χρόνο, επιλύοντας απορίες εκτός της προγραμματισμένης ώρας του μαθήματος.

5. Δυναμικός Δημιουργός Διαγωνισμάτων (Drag & Drop Test Builder)

Για τη βελτιστοποίηση της Εμπειρίας Χρήστη (UX) των εκπαιδευτικών, η τρέχουσα φόρμα δημιουργίας διαγωνισμάτων προτείνεται να αναβαθμιστεί σε έναν διαδραστικό κατασκευαστή (Interactive Test Builder). Μέσω λειτουργικότητας Drag & Drop, οι καθηγητές θα μπορούν να

συνθέτουν εξατομικευμένες κόλλες εξετάσεων προσθέτοντας μπλοκ (ερωτήσεις πολλαπλής επιλογής, ανάπτυξης, κενά), βελτιώνοντας δραματικά την παραγωγικότητά τους.

6. Πολυπαραγοντική Ταυτοποίηση (Multi-Factor Authentication – MFA)

Για την περαιτέρω θωράκιση της πρόσβασης, ειδικά όσον αφορά τους λογαριασμούς των Διαχειριστών και των Εκπαιδευτικών, σχεδιάζεται η ενσωμάτωση πρωτοκόλλου MFA. Κατά τη σύνδεση, ο χρήστης θα καλείται να εισάγει έναν επιπρόσθετο Κωδικό Μίας Χρήσης (OTP), ο οποίος θα παράγεται μέσω εφαρμογών ελέγχου ταυτότητας σε κινητές συσκευές (π.χ. Google Authenticator).

Βιβλιογραφία

Ελληνόγλωσση Βιβλιογραφία

Μπορουτζή, Δ., Βαλκάνος, Ε., Σιπητάνου, Α. Α., & Σταμπουλής, Μ. (2016). Οι Απόψεις των Εκπαιδευτικών Φροντιστηρίων Μέσης Εκπαίδευσης για την Επιμόρφωσή τους. *Education Sciences*, 2016(2), 95-116.

Ξενόγλωσση Βιβλιογραφία

- Balaji, S., & Murugaiyan, M. S. (2012). Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, 2(1), 26-30.
- Bamufleh, D., Almalki, M. A., Almohammadi, R., & Alharbi, E. (2021). User acceptance of Enterprise Resource Planning (ERP) systems in higher education institutions: A conceptual model. *International Journal of Enterprise Information Systems (IJEIS)*, 17(1), 144-163.
- Benavides, L. M. C., Tamayo Arias, J. A., Arango Serna, M. D., Branch Bedoya, J. W., & Burgos, D. (2020). Digital transformation in higher education institutions: A systematic literature review. *Sensors*, 20(11), 3291.
- Bhatt, B., & Nandu, M. (2021). An overview of structural uml diagrams. *International Research Journal of Engineering and Technology*, 8(8), 1577-1583.
- Bray, M. (2024). Missing data: What we know and what we need to know about private supplementary tutoring. *Revue internationale d'éducation de Sèvres*, (96).
- Chonoles, M. J. (2017). *Ocup 2 certification guide: Preparing for the omg certified uml 2.5 professional 2 foundation exam*. Morgan Kaufmann.
- Classter. (2025). *Classter - All-in-one Student & Learning Management System*.
- Eriksson, J. (2022). Migration of the User Interface of a Web Application: from Thymeleaf to Angular.
- Ermel, G., Farias, K., Gonçalves, L. J., & Bischoff, V. (2018, June). Supporting the composition of UML component diagrams. In *Proceedings of the XIV Brazilian Symposium on Information Systems* (pp. 1-9).
- Guynes, C. S., & Windsor, J. (2011). Revisiting client/server computing. *Journal of Business & Economics Research (JBER)*, 9(1).
- Han, D., Xing, J., Yang, Q., Wang, H., & Zhang, X. (2016, June). Formal sequence: Extending UML sequence diagram for behavior description and formal verification. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)* (Vol. 2, pp. 474-481). IEEE.

- Iso, I. S. O. (2011). Iec25010: 2011 systems and software engineering—systems and software quality requirements and evaluation (square)—system and software quality models. International Organization for Standardization, 34(2910), 208.
- Jangala, V. K. (2024). Authentication and authorization mechanisms in Java-based systems. International Journal of Contemporary Research in Multidisciplinary, 3(1), 277-284.
- Kassotakis, M., & Verdis, A. (2013). Shadow education in Greece: Characteristics, consequences and eradication efforts. In Private tutoring across the Mediterranean: Power dynamics and implications for learning and equity (pp. 93-113). Rotterdam: SensePublishers.
- Liu, Z., Wang, H., & Zan, H. (2010, October). Design and implementation of student information management system. In 2010 International symposium on intelligence information processing and trusted computing (pp. 607-610). IEEE.
- Moodle. (χ.χ.). Moodle - Open-source learning platform.
- Mule, S. S., Waykar, Y., & Mahavidyalaya, S. V. (2015). Role of USE CASE diagram in s/w development. International Journal of Management and Economics.
- Ordonez, C., Al-Amin, S. T., & Bellatreche, L. (2020, December). An er-flow diagram for big data. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 5795-5797). IEEE.
- Planas, E., & Cabot, J. (2020). How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus. Computer Standards & Interfaces, 67, 103363.
- Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.
- Rao, P. R., Chaurasia, A. K., & Singh, S. P. (2023). Modern web design: Utilizing HTML5, CSS3, and responsive techniques. The International Journal of Research and Innovation in Dynamics of Engineering, 1(8), a1-a18.
- Schechter, S., Brush, A. B., & Egelman, S. (2009, May). It's no secret. measuring the security and reliability of authentication via “secret” questions. In 2009 30th IEEE symposium on security and privacy (pp. 375-390). IEEE.
- Seeman, E. D., & O'Hara, M. (2006). Customer relationship management in higher education: using information systems to improve the student-school relationship. Campus-wide information systems, 23(1), 24-34.
- Shah, M. (2014). Impact of management information systems (MIS) on school administration: What the literature says. Procedia-Social and Behavioral Sciences, 116, 2799-2804.
- Siewe, F., & Ngounou, G. M. (2025). On the Execution and Runtime Verification of UML Activity Diagrams. Software, 4(1), 4.
- Singh, N., Chouhan, S. S., & Verma, K. (2021, October). Object oriented programming: Concepts, limitations and application trends. In 2021 5th International Conference on Information Systems and Computer Networks (ISCON) (pp. 1-4). IEEE.
- Spring. (2026). Spring Boot Reference Documentation.
- Subbey, M. (2023). Administrative challenges of centre workers of University of Education, Winneba. Journal of e-learning Research, 2(1), 70-81.

- Vassiliou-Gioles, T. (2020, December). A simple, lightweight framework for testing RESTful services with TTCN-3. In 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C) (pp. 498-505). IEEE.
- Vieira, D. (2022). Designing Hexagonal Architecture with Java: An architect's guide to building maintainable and change-tolerant applications with Java and Quarkus. Packt Publishing Ltd.
- Yahaya, N. F., & Abd Warif, N. B. (2024). A Kindergarten Management System with Dual Authentication for the City Kindergarten. Applied Information Technology And Computer Science, 5(1), 143-162.
- Yang, Y. (2022). Design and implementation of student information management system based on SpringBoot. Advances in Computer, Signals and Systems, 6(6), 21-25.

Παράρτημα I: Επιλεγμένος Πηγαίος Κώδικας

Συστήματος (Java / Spring Boot)

Στο παρόν παράρτημα παρατίθεται εκτενέστερος ο πηγαίος κώδικας από επιλεγμένες κλάσεις του συστήματος. Σκοπός είναι η ανάδειξη της αρχιτεκτονικής προσέγγισης (π.χ. διαχείριση εξαρτήσεων, annotations) που δεν είναι εμφανής στα μικρά αποσπάσματα του κυρίως κειμένου.

1. Κλάση StudentController (REST Controller / Driving Adapter)

Η παρακάτω κλάση αποτελεί το βασικό σημείο εισόδου για τις λειτουργίες των Σπουδαστών. Φαίνεται καθαρά η χρήση της Εξαγωνικής Αρχιτεκτονικής, καθώς ο Controller κάνει “inject” αποκλειστικά τις διεπαφές (Inbound Ports) για την εκτέλεση της επιχειρησιακής λογικής. (Σημείωση: Έχουν παραλειφθεί οι υπόλοιπες μέθοδοι του ελεγκτή για λόγους συντομίας).

```
package com.education.tutoring.management.rest.controller;

import com.education.tutoring.management.application.dto.student.StudentDailyScheduleDTO;
import com.education.tutoring.management.application.dto.EnrollmentResponseDTO;
import com.education.tutoring.management.application.dto.student.StudentLessonActivityDTO;
import com.education.tutoring.management.application.dto.student.StudentTestResponseDTO;
import com.education.tutoring.management.application.port.in.student.*;
import com.education.tutoring.management.domain.exception.IllegalOperationException;
import com.education.tutoring.management.domain.exception.ResourceNotFoundException;
import com.education.tutoring.management.domain.exception.UnauthorizedActionException;
import com.education.tutoring.management.domain.util.EnrollmentStatus;
import com.education.tutoring.management.rest.adapter.EnrollmentResourceManager;
import com.education.tutoring.management.rest.adapter.StudentResourceManager;
import com.education.tutoring.management.rest.model.*;
import com.education.tutoring.management.rest.model.student.EnrollmentRequestResource;
import com.education.tutoring.management.rest.model.student.StudentDailyScheduleResponseResource;
import com.education.tutoring.management.rest.model.student.StudentLessonActivityResponseResource;
import com.education.tutoring.management.rest.model.student.StudentTestResponseResource;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.responses.ApiResponse;
import io.swagger.v3.oas.annotations.tags.Tag;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.security.Principal;
import java.time.LocalDate;
import java.util.List;

/**
 * REST controller acting as the primary driving adapter for student-centric operations.
 * Exposes endpoints tailored to the student persona, orchestrating use cases for class
 * enrollment lifecycles, schedule retrieval, and the continuous monitoring of academic
 * progress, including test results and daily lesson activities.
 */
```

```
@Slf4j
@RestController
@RequiredArgsConstructor
@RequestMapping("/api/students")
@Tag(name = "Student API", description = "Endpoints for student operations")
public class StudentController {

    private final EnrollStudent enrollStudent;

    private final GetStudentEnrollments getStudentEnrollments;

    private final RequestEnrollmentDrop requestEnrollmentDrop;

    private final GetStudentSchedule getStudentSchedule;

    private final GetStudentTestResults getStudentTestResults;

    private final GetStudentLessonActivities getStudentLessonActivities;

    /**
     * Enrolls the currently authenticated student into a specific scheduled slot.
     * @param enrollmentRequestResource the request payload containing the ID of the
     * scheduled slot
     * @param principal the authenticated user principal representing the current student
     * @return a {@link ResponseEntity} containing the {@link EnrollmentResponseResource}
     * with the enrollment details
     * @throws ResourceNotFoundException if the specified scheduled slot does not exist
     * @throws IllegalArgumentException if the requested scheduled slot overlaps in time
     * with another class the student is already enrolled in
     */
    @Operation(summary = "Enroll in a scheduled class",
        description = "Allows an authenticated student to enroll in a specific scheduled slot. "
            + "Validates capacity, duplicate enrollments, and time-schedule conflicts.")
    @ApiResponse(responseCode = "200", description = "Successfully enrolled")
    @ApiResponse(responseCode = "400", description = "Already enrolled")
    @ApiResponse(responseCode = "403", description = "Schedule time-conflict detected")
    @ApiResponse(responseCode = "404", description = "Slot or student not found")
    @ApiResponse(responseCode = "409", description = "Capacity exceeded")
    @PostMapping("/me/enrollments")
    public ResponseEntity<EnrollmentResponseResource> enroll(
        @Valid @RequestBody EnrollmentRequestResource enrollmentRequestResource, Principal
principal)
        throws ResourceNotFoundException, IllegalArgumentException {

        EnrollmentResponseDTO enrollmentResponseDTO = enrollStudent.execute(principal.getName(),
            enrollmentRequestResource.getSlotId());

        EnrollmentResponseResource enrollmentResponseResource = EnrollmentResourceMapper.MAPPER
            .toEnrollmentResponseResource(enrollmentResponseDTO);

        return ResponseEntity.ok(enrollmentResponseResource);
    }

    // Παράλειψη των υπολοίπων μεθόδων: getMyEnrollments, getSchedule κτλ.
}
```

2. Κλάση SecurityConfig (Ρυθμίσεις Ασφαλείας)

Η παρακάτω κλάση εμπεριέχει τον πυρήνα της πολιτικής ασφαλείας της εφαρμογής, η οποία έχει αναπτυχθεί με βάση το πλαίσιο Spring Security. Ο κώδικας παρατίθεται αυτούσιος, καθώς αναδεικνύει την υλοποίηση αυθεντικοποίησης χωρίς κατάσταση (stateless authentication) μέσω φίλτρου JSON Web Tokens (JWT) και περιγράφει αναλυτικά τη λεπτομερή διαχείριση εξουσιοδότησης βάσει ρόλων (Role-Based Access Control) για τα διακριτά υποσυστήματα του φροντιστηρίου (Διαχειριστές, Εκπαιδευτικοί, Σπουδαστές).

```
package com.education.tutoring.management.rest.configuration;

import com.education.tutoring.management.rest.security.JwtAuthenticationFilter;
import lombok.RequiredArgsConstructor;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configurers.AbstractHttpConfigurer;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

/**
 * Configuration class for security-related beans.
 */
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class SecurityConfig {

    private final JwtAuthenticationFilter jwtAuthenticationFilter;

    /**
     * Provides a PasswordEncoder bean for hashing passwords.
     * @return Bcrypt password encoder
     */
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

        http.csrf(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests(auth -> auth
                // --- SWAGGER & API DOCS ---
                .requestMatchers("/docs/**", "/v3/api-docs/**", "/swagger-ui/**"))
    }
}
```

```
.permitAll()

.requestMatchers("/css/**", "/js/**", "/images/**", "/favicon.ico")
.permitAll()

// --- REST API AUTHENTICATION ---
.requestMatchers("/api/auth/**")
.permitAll()

// --- UI PAGES (PUBLIC) ---
.requestMatchers(HttpMethod.GET, "/", "/login", "/register", "/courses", "/error")
.permitAll()

// --- UI PAGES (SECURED BY ROLE) ---
.requestMatchers(HttpMethod.GET, "/admin/**")
.hasAuthority("ROLE_ADMIN")
.requestMatchers(HttpMethod.GET, "/teacher/**")
.hasAuthority("ROLE_TEACHER")
.requestMatchers(HttpMethod.GET, "/student/**")
.hasAuthority("ROLE_STUDENT")

// --- REST API STUDENTS ---
.requestMatchers("/api/students/me/**")
.hasAuthority("ROLE_STUDENT")

// --- REST API TEACHERS ---
.requestMatchers("/api/teachers/**")
.hasAuthority("ROLE_TEACHER")

// --- REST API ADMIN ---
.requestMatchers("/api/admin/**")
.hasAuthority("ROLE_ADMIN")

// --- REST API SCHEDULED SLOTS (GLOBAL TIMETABLE) ---
.requestMatchers(HttpMethod.GET, "/api/slots")
.authenticated()

// --- REST API COURSES ---
.requestMatchers(HttpMethod.GET, "/api/courses")
.permitAll()
.requestMatchers(HttpMethod.GET, "/api/courses/*/slots")
.permitAll()

// --- REST API COURSES (ADMIN) ---
```

```
.requestMatchers("/api/courses/**")
    .hasAuthority("ROLE_ADMIN")

// --- REST API USERS ---
.requestMatchers("/api/users/**")
    .authenticated()

// --- CATCH-ALL (Safety Net) ---
.anyRequest()
    .authenticated()

.sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
    .addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class);

return http.build();
}
}
```