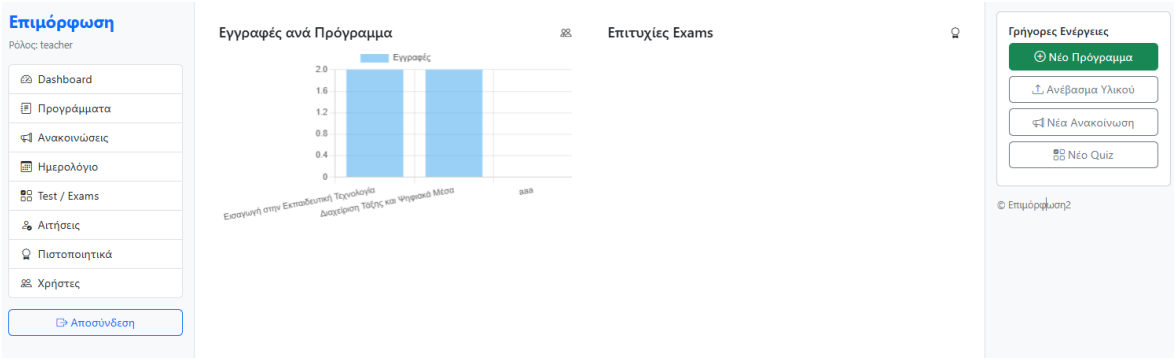
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Πλατφόρμα διαχείρισης επιμορφωτικών προγραμμάτων για εκπαιδευτικούς»



Της Φοιτήτριας

ΕΛΕΝΗ ΤΣΑΧΟΥΡΙΔΟΥ

ΑΜ **144382**

Επιβλέπων

Κυριάκος Τσιακμάκης

Επίκουρος Καθηγητής

Σεπτέμβριος 2025

Πλατφόρμα διαχείρισης επιμορφωτικών προγραμμάτων για εκπαιδευτικούς

Κωδικός: 25225

Φοιτητής: Τσαχουρίδου Ελένη

Εισηγητής: Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 30-03-2025

Ημερομηνία περάτωσης Π.Ε. 30-08-2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η παρούσα εργασία υλοποιεί μια πλατφόρμα επιμόρφωσης με δύο ρόλους: επιμορφωτή και επιμορφούμενο. Ο επιμορφωτής δημιουργεί προγράμματα, οργανώνει ενότητες και υλικό, δημοσιεύει ανακοινώσεις, ορίζει αξιολογήσεις (quizzes/exams) και εκδίδει πιστοποιητικά. Ο επιμορφούμενος βλέπει ενεργά/διαθέσιμα προγράμματα, υποβάλλει αίτηση, μελετά περιεχόμενο, συμμετέχει σε αξιολογήσεις και παραλαμβάνει βεβαιώσεις. Η εφαρμογή βασίζεται σε Python/Flask (backend) και MySQL (επίμονη αποθήκευση), με REST-like endpoints που επιστρέφουν JSON και διεπαφή HTML/Bootstrap με δυναμικούς πίνακες (DataTables). Η πρόσβαση ελέγχεται μέσω session και ρόλων, ενώ η ροή εγγραφής είναι σε δύο φάσεις (αίτηση–έγκριση→εγγραφή). Οι αξιολογήσεις διαθέτουν χρονικό όριο, απόπειρες και λεπτομερή καταγραφή απαντήσεων, ώστε να υπολογίζονται αποτελέσματα και πρόοδος. Τα πιστοποιητικά εκδίδονται βάσει κανόνων ολοκλήρωσης. Υποστηρίζονται επίσης γεγονότα/ημερολόγιο για χρονικό προγραμματισμό και ενιαία μορφοποίηση δεδομένων για σταθερό UI/UX.

«Training program management platform»

Abstract

This thesis implements a training platform with two roles: instructor and learner. The instructor creates programs, organizes modules and materials, publishes announcements, defines assessments (quizzes/exams), and issues certificates. The learner views active/available programs, submits an application, studies content, takes assessments, and receives certificates. The application is built on Python/Flask (backend) and MySQL (persistent storage), with REST-like endpoints returning JSON and an HTML/Bootstrap interface using dynamic tables (DataTables). Access is enforced via sessions and roles, while enrollment follows a two-step flow (application → approval → enrollment). Assessments support time limits, multiple attempts, and detailed answer logging to compute results and progress. Certificates are issued based on completion rules. The platform also supports events/calendar for scheduling and consistent data formatting for stable UI/UX.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Η εισαγωγή της εργασίας

1.1 Λίγα πράγματα για το θέμα που πραγματεύεται

Η εργασία αφορά τον σχεδιασμό και την υλοποίηση μιας διαδικτυακής πλατφόρμας επιμόρφωσης. Η πλατφόρμα υποστηρίζει δύο βασικούς ρόλους, επιμορφωτή και επιμορφούμενο, και οργανώνει τη ροή: πρόγραμμα → ενότητες → περιεχόμενο → αξιολόγηση → πιστοποίηση.

Στόχος είναι η συγκέντρωση των βασικών λειτουργιών σε ένα ενιαίο, ελαφρύ σύστημα. Ο επιμορφωτής δημιουργεί προγράμματα, αναρτά υλικό, δημοσιεύει ανακοινώσεις, ορίζει quizzes/exams και εκδίδει πιστοποιητικά. Ο επιμορφούμενος βλέπει ενεργά και διαθέσιμα προγράμματα, υποβάλει αίτηση εγγραφής, μελετά το υλικό, συμμετέχει σε αξιολογήσεις και λαμβάνει βεβαιώσεις.

Η υλοποίηση βασίζεται σε Flask (Python) για το backend, MySQL για την αποθήκευση δεδομένων και HTML/Bootstrap/JavaScript για το frontend. Η επικοινωνία γίνεται μέσω REST-like endpoints που επιστρέφουν JSON ώστε οι πίνακες και τα γραφήματα να ενημερώνονται δυναμικά.

Η λογική πρόσβασης υλοποιείται με διακριτούς blueprints ανά ρόλο και έλεγχο session. Τα δεδομένα χρήστη, προγραμμάτων, εγγραφών, περιεχομένου, αξιολογήσεων και πιστοποιήσεων αποθηκεύονται σε κανονικοποιημένους πίνακες με ευθεία αντιστοίχιση στις λειτουργικές οντότητες της πλατφόρμας.

Η διαδικασία εγγραφής σε πρόγραμμα γίνεται σε δύο στάδια: αίτηση (registration) και οριστικοποίηση (enrollment) μετά από έγκριση επιμορφωτή. Οι ανακοινώσεις συνδέονται με συγκεκριμένα προγράμματα, τα υλικά αποθηκεύονται ανά ενότητα και τα events ορίζουν χρονοθυρίδες δραστηριοτήτων.

Το υποσύστημα αξιολόγησης υποστηρίζει Test/Exam με χρονικό όριο, όριο επιτυχίας, ερωτήσεις και επιλογές απαντήσεων. Οι προσπάθειες (attempts) καταγράφονται αναλυτικά ανά χρήστη και ερώτηση ώστε να υπολογίζεται αποτέλεσμα και πρόοδος.

Η έκδοση πιστοποιητικών βασίζεται σε κανόνες ολοκλήρωσης (π.χ. απαιτούμενος αριθμός επιτυχών εξετάσεων). Η κατάσταση του χρήστη σε κάθε πρόγραμμα προκύπτει από συνδυασμό εγγραφών, αποτελεσμάτων και χρονικών ορίων.

Ο σχεδιασμός στοχεύει σε απλή ανάπτυξη και εύκολη συντήρηση. Τα endpoints είναι σαφή και επαναχρησιμοποιήσιμα, οι σελίδες είναι λειτουργικές χωρίς περίπλοκες εξαρτήσεις και η αρχιτεκτονική επιτρέπει μελλοντικές επεκτάσεις (ειδοποιήσεις, στατιστικά, υποβολές εργασιών).

Η επιλογή τεχνολογιών έγινε με κριτήρια διαθεσιμότητας, τεκμηρίωσης και χαμηλού κόστους. Η πλατφόρμα μπορεί να αναπτυχθεί σε τυπικό xampp περιβάλλον ή σε container, με ελάχιστες αλλαγές ρυθμίσεων.

Σε λειτουργικό επίπεδο, το σύστημα καλύπτει τον βασικό κύκλο ζωής μιας επιμόρφωσης: δημιουργία προγράμματος, διάθεση υλικού, ενημέρωση συμμετεχόντων, αξιολόγηση γνώσεων και τελική πιστοποίηση. Αυτή είναι η περιοχή που πραγματεύεται η εργασία.

Η πλατφόρμα διακρίνεται σε δύο όψεις. Στην πλευρά του επιμορφωτή περιλαμβάνει δημιουργία και διαχείριση προγραμμάτων, οργάνωση ενοτήτων, ανάρτηση υλικού, δημοσίευση ανακοινώσεων, ορισμό quizzes ή exams, έλεγχο αιτήσεων εγγραφής και έκδοση πιστοποιητικών. Στην πλευρά του επιμορφούμενου περιλαμβάνει προβολή ενεργών και διαθέσιμων προγραμμάτων, υποβολή αίτησης, πρόσβαση σε ανακοινώσεις και υλικό ανά ενότητα, συμμετοχή σε αξιολογήσεις με χρονικό περιορισμό και προβολή βεβαιώσεων. Κοινή υποδομή υποστηρίζει γεγονότα (events) με ημερομηνίες έναρξης και λήξης, ώστε να σηματοδοτούνται περίοδοι δραστηριοτήτων.

Η υλοποίηση έγινε με Flask για το backend, MySQL για την επίμονη αποθήκευση και HTML/Bootstrap/JavaScript για το frontend. Τα endpoints ακολουθούν REST-like λογική και επιστρέφουν JSON. Η διεπαφή κάνει χρήση δυναμικών πινάκων και ασύγχρονων κλήσεων, ώστε τα δεδομένα να ανανεώνονται χωρίς πλήρη επαναφόρτωση. Η αρχιτεκτονική οργανώνεται με Blueprints ανά ρόλο (teacher, student) και διακριτή ενότητα αυθεντικοποίησης. Η πρόσβαση ελέγχεται μέσω session και απλών ελέγχων ρόλου στην αρχή κάθε route. Η βάση δεδομένων ακολουθεί κανονικοποιημένη δομή με ισομορφία προς τις λειτουργικές οντότητες, ώστε τα ερωτήματα να παραμένουν σαφή και προβλέψιμα.

Η εγγραφή σε πρόγραμμα είναι διφασική: αρχικά αίτηση από τον επιμορφούμενο και στη συνέχεια έγκριση ή απόρριψη από τον επιμορφωτή· αν εγκριθεί δημιουργείται εγγραφή συμμετοχής. Οι ανακοινώσεις συνδέονται με συγκεκριμένο πρόγραμμα και φαίνονται μόνο στους εγγεγραμμένους. Το περιεχόμενο οργανώνεται σε ιεραρχία προγράμματος–ενότητας–τεκμηρίου, με σαφείς μεταδεδομένες ιδιότητες. Οι αξιολογήσεις υποστηρίζουν ορισμό τύπου (Test/Exam), χρονικού ορίου και ορίου επιτυχίας. Κάθε προσπάθεια αποθηκεύεται με ανάλυση απαντήσεων ανά ερώτηση, επιτρέποντας υπολογισμό βαθμολογίας και ελέγχους πληρότητας. Η έκδοση πιστοποιητικού προϋποθέτει επίτευξη κανόνων ολοκλήρωσης, όπως ελάχιστος αριθμός επιτυχών αξιολογήσεων.

Η πλατφόρμα τοποθετείται ως ελαφριά, πλήρως παραμετροποιήσιμη λύση για μικρές και μεσαίες δομές επιμόρφωσης. Σε αντίθεση με σύνθετες πλατφόρμες γενικού σκοπού, εδώ προτιμήθηκε σαφής ελάχιστος πυρήνας λειτουργιών που καλύπτει συχνά σενάρια χωρίς περίσσεια ρυθμίσεων. Η απλότητα της στοίβας και η καθαρή αντιστοίχιση των οντοτήτων σε πίνακες και routes διευκολύνουν την υιοθέτηση, τη συντήρηση και την εκπαίδευση νέων μελών ομάδας πάνω στον κώδικα.

Στα επόμενα κεφάλαια παρουσιάζονται η ανάλυση απαιτήσεων και τα διαγράμματα ροών, η λεπτομερής περιγραφή της βάσης δεδομένων και των οντοτήτων, η αρχιτεκτονική της εφαρμογής με έμφαση στα Blueprints και στα API, παραδείγματα κώδικα αντιπροσωπευτικών ροών (δημιουργία

προγράμματος, αίτηση–εγγραφή, εκτέλεση αξιολόγησης, έκδοση πιστοποιητικού), τα σενάρια δοκιμών και μετρικές, και τέλος τα συμπεράσματα με προτάσεις βελτίωσης.

Στο πλαίσιο της συγγραφής της εργασίας αξιοποιήθηκε εργαλείο τεχνητής νοημοσύνης για υποβοήθηση στη γλωσσική επιμέλεια, στη βελτίωση της διατύπωσης και στην παραγωγή εναλλακτικών εκφράσεων.

Κεφάλαιο 2ο: Υπάρχουσες πλατφόρμες και εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση

2.1 Moodle

Το Moodle είναι ένα από τα πιο διαδεδομένα συστήματα διαχείρισης μάθησης (LMS) ανοιχτού κώδικα και λειτουργεί ως πλήρες περιβάλλον οργάνωσης μαθημάτων, δραστηριοτήτων και αξιολογήσεων. Αξιοποιείται ευρέως σε σχολεία, ΑΕΙ και οργανισμούς, κυρίως λόγω της ευελιξίας του, της μεγάλης κοινότητας και της επεκτασιμότητάς του μέσω πρόσθετων. Στη δική μας υλοποίηση, που στοχεύει σε ελαφριά, στοχευμένη πλατφόρμα επιμόρφωσης με συγκεκριμένες ροές (πρόγραμμα → ενότητες → περιεχόμενο → αξιολόγηση → πιστοποίηση), το Moodle αποτελεί σημείο αναφοράς: μας δείχνει ροές και δυνατότητες που μπορούμε να υιοθετήσουμε τμηματικά χωρίς το συνολικό βάρος ενός «μεγάλου» LMS. Στη συνέχεια παρουσιάζονται οι βασικές περιοχές του Moodle και ο χάρτης αντιστοίχισης προς τις λειτουργίες του δικού μας συστήματος. [1-2]

Στον πυρήνα του Moodle βρίσκεται η έννοια του μαθήματος και των δραστηριοτήτων/πόρων. Η δομή μαθήματος οργανώνεται σε ενότητες, μέσα στις οποίες ο εκπαιδευτής προσθέτει υλικό, εργασίες, συζητήσεις και αξιολογήσεις. Για την παρακολούθηση προόδου προσφέρονται μηχανισμοί activity completion και course completion, οι οποίοι επιτρέπουν να οριστούν κριτήρια ολοκλήρωσης ανά δραστηριότητα και συνολικά ανά μάθημα. Το Moodle εμφανίζει την πρόοδο τόσο στον εκπαιδευόμενο όσο και στον εκπαιδευτή, με αναφορές και ενδείξεις ολοκλήρωσης. Η σχετική τεκμηρίωση περιγράφει ρητά την παρακολούθηση και τη σήμανση ολοκλήρωσης δραστηριοτήτων/μαθημάτων.

Η αντιστοίχιση στη δική μας πλατφόρμα είναι άμεση: το «μάθημα» του Moodle αντιστοιχεί στο «πρόγραμμα», οι «ενότητες» είναι οι δικές μας ενότητες, ενώ το «activity completion» προσεγγίζεται από τους κανόνες επιτυχίας στις αξιολογήσεις και η «course completion» από τη λογική έκδοσης πιστοποιητικού με κανόνες ολοκλήρωσης. Στην πράξη, το Moodle διαθέτει βαθμολόγιο και συγκεντρωτικές προβολές προόδου, κάτι που καλύπτουμε σήμερα πιο ελαφρά με αποθήκευση προσπαθειών και αποτελεσμάτων και μπορούμε να εξελίξουμε προς κατευθύνσεις παρόμοιες με το gradebook.

Ένα υποσύστημα του Moodle είναι το Quiz. Ο εκπαιδευτής δημιουργεί τράπεζα ερωτήσεων και συνθέτει αξιολογήσεις με χρονικό περιορισμό, πολλαπλές απόπειρες, τυχαιοποίηση σειράς ερωτήσεων/επιλογών και πλούσιες ρυθμίσεις ανατροφοδότησης. Η τεκμηρίωση αναφέρει ποικιλία τύπων ερωτήσεων, χρήση question bank, έλεγχο attempts και ρυθμίσεις εμφάνισης καθώς και δυνατότητα τυχαίας εισαγωγής ερωτήσεων από κατηγορίες. Αυτές οι πρακτικές αντιστοιχούν άμεσα

στο δικό μας υποσύστημα quizzes/exams και επιβεβαιώνουν ότι η αποθήκευση προσπάθειας ανά ερώτηση, ο χρονικός περιορισμός και η τυχαιοποίηση αποτελούν συνηθισμένες απαιτήσεις.

Η επικοινωνία με τους εκπαιδευόμενους στο Moodle καλύπτεται θεσμικά από τον Announcements forum: κάθε μάθημα έχει ειδικό χώρο ανακοινώσεων όπου ο εκπαιδευτής δημοσιεύει ενημερώσεις που εμφανίζονται στους συμμετέχοντες και μπορούν να αποσταλούν και ως ειδοποιήσεις. Η δική μας υλοποίηση προσφέρει σελίδα ανακοινώσεων συνδεδεμένη με πρόγραμμα· λειτουργικά, εξυπηρετεί την ίδια ανάγκη, αν και το Moodle διαθέτει πιο ενσωματωμένο οικοσύστημα μηνυμάτων/ειδοποιήσεων.

Η διαχείριση εγγραφών στο Moodle υλοποιείται με «enrolment plugins». Υποστηρίζονται πολλαπλές μέθοδοι, όπως manual enrolment, self-enrolment με κλειδί, εγγραφή μέσω εξωτερικής βάσης ή επί πληρωμή με PayPal. Το μοντέλο μας με «αίτηση» και «έγκριση» αντιστοιχεί εν μέρει σε σενάρια που στο Moodle υλοποιούνται με κατάλληλη μέθοδο εγγραφής ή με συμπληρωματικά πρόσθετα/ροές (π.χ. enrolment key ή εξωτερική πηγή που ορίζει ποιος μπαίνει πού). Σε κάθε περίπτωση το Moodle προσφέρει ευρύ φάσμα μεθόδων που επιτρέπει να προσεγγίσουμε τις δικές μας ροές χωρίς να «πειράξουμε» το μάθημα.

Σημαντική ομοιότητα με τη δική μας λογική «events» είναι το ημερολόγιο του Moodle. Το σύστημα διαθέτει ημερολόγιο με συμβάντα επιπέδου ιστότοπου, μαθήματος, ομάδας και χρήστη, συνδέει προθεσμίες εργασιών/quiz και υποστηρίζει υπενθυμίσεις μέσω πρόσθετου. Αυτή η λειτουργικότητα προσομοιάζει τα δικά μας «events» ανά πρόγραμμα με ημερομηνίες έναρξης/λήξης και ενισχύει τη θέση ότι ο χρονικός προγραμματισμός πρέπει να είναι ορατός και συγκεντρωμένος.

Για την έκδοση πιστοποιητικών, το Moodle δεν περιορίζεται σε στατικό μηχανισμό αλλά προσφέρει το πρόσθετο Custom certificate, με το οποίο δημιουργούνται δυναμικά PDF με πλήρη παραμετροποίηση μέσα από τον φυλλομετρητή. Αυτό βρίσκεται πολύ κοντά στη δική μας έννοια «πιστοποιητικού μετά από κανόνες ολοκλήρωσης» και υποδεικνύει πρακτικές για διάταξη, πεδία, ασφάλεια και έκδοση.

Το Moodle εκθέτει web services και υποστηρίζει REST, δίνοντας τη δυνατότητα να αυτοματοποιηθούν ροές, να ενσωματωθεί με τρίτα συστήματα ή να δημιουργηθούν εξωτερικοί πελάτες που καταχωρούν/ανακτούν δεδομένα με tokens. Στη δική μας αρχιτεκτονική, όπου ήδη χρησιμοποιούμε REST-like endpoints, αυτό αποτελεί ξεκάθαρο μονοπάτι για μελλοντική διαλειτουργικότητα και αυτοματισμούς.

Σε επίπεδο «φιλοσοφίας» σχεδίασης, το Moodle επιλέγει γενικότητα και πληρότητα: καλύπτει μεγάλο εύρος χρήσεων με πλούσια παραμετροποίηση, ρόλους και υποσυστήματα. Το δικό μας σύστημα, με στοχευμένη στοίβα Flask/MySQL και λιτές ροές, δίνει προτεραιότητα στην απλότητα ανάπτυξης, στην άμεση χαρτογράφηση οντοτήτων και στην ευκολία συντήρησης. Αυτό σημαίνει ότι για μικρές/μεσαίες δομές επιμόρφωσης η δική μας λύση μπορεί να εγκατασταθεί και να προσαρμοστεί ταχύτερα, χωρίς να απαιτείται εξειδικευμένη γνώση ενός μεγάλου LMS, ενώ εξακολουθεί να ευθυγραμμίζεται με τις καθιερωμένες πρακτικές που αναδεικνύει το Moodle (πρόοδος/ολοκλήρωση, αξιολογήσεις,

ανακοινώσεις, ημερολόγιο, πιστοποιήσεις). Όταν προκύπτει ανάγκη για πιο σύνθετα σενάρια (π.χ. δεκάδες μέθοδοι εγγραφής, εκτεταμένο gradebook, ολοκληρωμένη μηνυματοδοσία), το Moodle προσφέρει πιο πολλά.

2.2 Canvas LMS (Instructure)

Το Canvas LMS είναι ένα ευρέως διαδεδομένο σύστημα διαχείρισης μάθησης που οργανώνει τα μαθήματα σε «Modules» (ενότητες) και επιτρέπει στον διδάσκοντα να ρυθμίζει τη ροή του μαθήματος, το υλικό και τις δραστηριότητες. Τα Modules λειτουργούν ως γραμμικός κορμός περιεχομένου (εβδομάδες, θεματικές ενότητες κ.λπ.), ενοποιώντας αρχεία, σελίδες, κουίζ και εργασίες κάτω από ενιαία ροή πλοήγησης. Αυτή η φιλοσοφία αντιστοιχίζεται άμεσα στο δικό μας μοντέλο «πρόγραμμα → ενότητες → περιεχόμενο», όπου οι ενότητες οργανώνουν το υλικό και τις ενέργειες του επιμορφωτή σε σαφή σειρά. [3-4]

Για την επίσημη επικοινωνία προς τους εκπαιδευόμενους, το Canvas παρέχει Announcements σε επίπεδο μαθήματος· ο διδάσκων αναρτά ενημερώσεις που εμφανίζονται κεντρικά και μπορούν να ενεργοποιήσουν ειδοποιήσεις. Η λειτουργία αυτή καλύπτει τον ίδιο σκοπό με τις «Ανακοινώσεις» ανά πρόγραμμα στη δική μας πλατφόρμα: άμεση, ορατή ενημέρωση σε όλους τους συμμετέχοντες του εκάστοτε προγράμματος.

Στον τομέα της αξιολόγησης, το Canvas διαθέτει το υποσύστημα New Quizzes: ο εκπαιδευτής δημιουργεί κουίζ με διάφορους τύπους ερωτήσεων, χρονικά όρια, τυχαιοποίηση και διαχείριση προσπαθειών. Το «New Quizzes» έχει σταδιακά αντικαταστήσει το «Classic Quizzes» και υποστηρίζεται με οδηγούς/τεκμηρίωση για δημιουργία, ρυθμίσεις και αναφορές. Η δική μας υλοποίηση υιοθετεί αντίστοιχες αρχές (ερωτήσεις, επιλογές, χρονικά όρια, καταγραφή προσπαθειών), άρα η αρχιτεκτονική μας ευθυγραμμίζεται με πρακτικές που θεωρούνται καθιερωμένες στον χώρο.

Το Gradebook του Canvas συγκεντρώνει τις βαθμολογίες ανά δραστηριότητα και φοιτητή, με προβολές, εισαγωγή/εξαγωγή (π.χ. CSV) και αυτοματισμούς για ελλιπείς/εκπρόθεσμες υποβολές. Στη δική μας πλατφόρμα η αποθήκευση προσπαθειών/βαθμολογιών υπάρχει ήδη σε επίπεδο quiz· ένα επόμενο βήμα είναι η δημιουργία ενοποιημένης προβολής προόδου τύπου gradebook, στα πρότυπα του Canvas.

Ο χρονοπρογραμματισμός στο Canvas υλοποιείται με Calendar (μαθήματος/λογαριασμού) και Events, που προβάλλονται σε ημέρα/εβδομάδα/μήνα και συνδέονται με τις δραστηριότητες του μαθήματος. Η λογική αυτή αντιστοιχεί στα δικά μας «events» ανά πρόγραμμα (έναρξη/λήξη), όπου επιδιώκουμε η χρονική πληροφορία να είναι συγκεντρωμένη και συνεπής στη διεπαφή χρήστη.

Σε επίπεδο διασύνδεσης, το Canvas εκθέτει εκτεταμένο REST API, επιτρέποντας σε εξωτερικά συστήματα να δημιουργούν/ανακτούν δεδομένα (OAuth2), και υποστηρίζει LTI (External Tools) για ενσωμάτωση τρίτων εργαλείων απευθείας στο περιβάλλον του μαθήματος. Η δική μας πλατφόρμα ήδη χρησιμοποιεί REST-like endpoints· η αρχιτεκτονική μας μπορεί να επεκταθεί με εξωτερικές ολοκληρώσεις κατά το πρότυπο του Canvas (π.χ. σύνδεση εργαλείων ή υπηρεσιών αξιολόγησης).

Επίσης είναι ότι το Canvas διατίθεται και ως ανοικτού κώδικα (AGPLv3) από την Instructure, γεγονός που έχει ενισχύσει την κοινότητα, τα πρόσθετα και τα παραδείγματα ολοκληρώσεων. Η δική μας στοίβα (Flask/MySQL) είναι ελαφρύτερη, αλλά το άνοιγμα διεπαφών/τεκμηρίωσης και η καθαρή μοντελοποίηση που ακολουθεί το Canvas αποτελούν πρακτικούς οδηγούς για την περαιτέρω ωρίμανση του συστήματός μας.

2.3 Open edX

Το Open edX είναι μια ανοικτού κώδικα πλατφόρμα μάθησης που χρησιμοποιείται διεθνώς για τη δημιουργία και λειτουργία online μαθημάτων μεγάλης κλίμακας. Αρχιτεκτονικά αποτελείται από πολλαπλές υπηρεσίες που συνεργάζονται: το περιβάλλον συγγραφής μαθημάτων (Studio) για τους διδάσκοντες και το περιβάλλον μάθησης (LMS) για τους εκπαιδευόμενους, πλαισιωμένα από APIs, πρόσθετα και βοηθητικά συστήματα. Η επίσημη τεκμηρίωση περιγράφει την πλατφόρμα ως σύνολο web υπηρεσιών που παρέχει όλα τα απαραίτητα εργαλεία για να χτίσεις και να «τρέξεις» μαθήματα στο διαδίκτυο.[5-6]

Στο επίπεδο μοντελοποίησης περιεχομένου, το Studio λειτουργεί ως «runtime» συγγραφής. Οι διδάσκοντες δημιουργούν και οργανώνουν το courseware (μαθήματα, ενότητες, μονάδες) και ρυθμίζουν τα συστατικά μάθησης που εμφανίζονται στο LMS. Η τεχνολογία-κλειδί εδώ είναι τα XBlocks, δηλαδή επεκτάσιμα δομικά στοιχεία που υλοποιούν δραστηριότητες, πόρους και αλληλεπιδράσεις μέσα στο μάθημα· η τεκμηρίωση εξηγεί πώς το LMS και το Studio επεκτείνουν τον πυρήνα των XBlocks, πώς γίνεται η ανάπτυξη/δοκιμή/ανάπτυξη παραγωγής και πώς αποδίδονται τα XBlocks στο περιβάλλον του χρήστη. Για εμάς αυτό αντιστοιχεί άμεσα στη λογική «ενότητα → περιεχόμενο» της πλατφόρμας μας, με τη διαφορά ότι στο Open edX η επεκτασιμότητα είναι ενσωματωμένη στον ίδιο τον τρόπο δόμησης περιεχομένου.

Η παρακολούθηση προόδου και η ολοκλήρωση μαθήματος είναι βασικές δυνατότητες: οι δραστηριότητες μπορούν να έχουν κριτήρια ολοκλήρωσης και το μάθημα συνολικά να θέτει προϋποθέσεις «course completion». Στο δικό μας σύστημα αυτό τοποθετείται ως κανόνες ολοκλήρωσης προγράμματος (π.χ. ελάχιστος αριθμός επιτυχών αξιολογήσεων), ενώ στο Open edX η έννοια είναι περισσότερο γενικευμένη και ενοποιημένη στο επίπεδο courseware.

Το υποσύστημα αξιολόγησης στο Open edX βασίζεται σε κουίζ και ερωτήσεις που εντάσσονται ως XBlocks και υποστηρίζουν ποικιλία ρυθμίσεων: τύποι ερωτήσεων, χρονικά όρια, απόπειρες, τυχαιοποίηση, βαθμολόγηση και ανατροφοδότηση. Η τεκμηρίωση και οι οδηγοί χρήσης για τα REST APIs αναδεικνύουν ότι η πλατφόρμα διαθέτει σαφή σημεία ενοποίησης για διαχείριση αποτελεσμάτων και ροών· στη δική μας υλοποίηση έχουμε ήδη υιοθετήσει αντίστοιχες έννοιες (quizzes/exams, attempts, per-question answers) και ετοιμάζουμε ενοποιημένες προβολές τύπου gradebook με βάση το ίδιο σκεπτικό.

Η απονομή πιστοποιητικών είναι πρώτη λειτουργία-πολίτης στο Open edX: μπορεί να παραχθεί αυτόματα ψηφιακό πιστοποιητικό με βάση την ολοκλήρωση του μαθήματος και να γίνει διαθέσιμο στο dashboard, στο profile του εκπαιδευόμενου και στη σελίδα προόδου. Η τεκμηρίωση εξηγεί πότε εμφανίζονται, πώς ενεργοποιούνται σε self-paced ή instructor-paced ροές και ποιες διαμορφώσεις απαιτούνται στον κόμβο. Αυτό αντιστοιχίζεται ευθέως με το δικό μας μηχανισμό έκδοσης βεβαιώσεων/πιστοποιητικών μετά από κανόνες ολοκλήρωσης, δίνοντάς μας πρακτικές για το πότε και πού πρέπει να εμφανίζονται τα σχετικά κουμπιά/σύνδεσμοι στον εκπαιδευόμενο.

Στο κομμάτι επικοινωνίας, οι συζητήσεις και οι ανακοινώσεις είναι κεντρικές. Το Open edX υποστηρίζει Course Discussions τόσο σε επίπεδο μαθήματος όσο και σε επίπεδο συγκεκριμένων ενοτήτων/μονάδων, ενώ από την έκδοση Olive και μετά επιτρέπει παραμετροποίηση τρίτου παρόχου συζητήσεων (π.χ. εναλλακτικό discussion provider) πριν την έναρξη του μαθήματος. Αυτό τοποθετεί την επικοινωνία ως αποσυνδεδεμένο αλλά ενσωματώσιμο υποσύστημα, κάτι που ευθυγραμμίζεται με τη δική μας επιλογή να κρατάμε τις ανακοινώσεις/συζητήσεις κοντά στο πρόγραμμα, αλλά με περιθώριο ενοποίησης αν απαιτηθεί.

Όσον αφορά τις διεπαφές ολοκλήρωσης, το Open edX διαθέτει ανοιχτή τεκμηρίωση REST APIs και καθοδήγηση για αυθεντικοποίηση με JWT. Αυτό επιτρέπει σε εξωτερικά συστήματα να ανακτούν/δημιουργούν πόρους (π.χ. χρήστες, εγγραφές, αποτελέσματα) και να ενοποιούν υπηρεσίες. Η δική μας πλατφόρμα ήδη εκθέτει REST-like endpoints και αξιοποιεί JSON· η στοχοθεσία για μελλοντική διαλειτουργικότητα (π.χ. σύνδεση με συστήματα αναφορών ή badges) συμβαδίζει με τις πρακτικές του Open edX.

2.4 Τι χρησιμοποιήσαμε για αυτό το έργο

2.4.1 Python

Η Python αποτελεί σήμερα μία από τις πιο διαδεδομένες γλώσσες προγραμματισμού, κυρίως λόγω της καθαρής σύνταξης και της χαμηλής καμπύλης εκμάθησης που προσφέρει. Ο σχεδιασμός της προωθεί την αναγνωσιμότητα και τη σαφήνεια, επιτρέποντας στον προγραμματιστή να εκφράζει την πρόθεση

του κώδικα με λίγες, κατανοητές γραμμές. Η γλώσσα είναι δυναμικά τυποποιημένη και ερμηνευόμενη, κάτι που ευνοεί την ταχεία ανάπτυξη και τον πειραματισμό, χωρίς να απαιτείται πολύπλοκη διαδικασία μεταγλώττισης. Παράλληλα, η τυπική βιβλιοθήκη της είναι εκτεταμένη και καλύπτει μεγάλο εύρος καθημερινών αναγκών, από χειρισμό αρχείων και ημερομηνιών έως δικτυακό προγραμματισμό και επεξεργασία δεδομένων. Γύρω από αυτόν τον πυρήνα έχει αναπτυχθεί ένα πλούσιο οικοσύστημα πακέτων και πλαισίων εργασίας, το οποίο καθιστά την Python κατάλληλη τόσο για διδασκαλία όσο και για παραγωγικές, επαγγελματικές εφαρμογές. [7-10]

Ένα χαρακτηριστικό που ξεχωρίζει είναι η έμφαση στη φιλοσοφία «ένα προφανές, σαφές μονοπάτι για να γίνει κάτι». Η σύνταξη αποθαρρύνει την περιττή πολυπλοκότητα και ευνοεί την ομοιομορφία ύφους, γεγονός που μειώνει τον χρόνο που απαιτείται για να κατανοήσει κάποιος κώδικα που δεν έγραψε ο ίδιος. Αυτό συνδέεται άμεσα με τη συντηρησιμότητα. Και ομάδες που εργάζονται σε Python συχνά αναφέρουν ότι ο κώδικας παραμένει προσβάσιμος σε περισσότερα μέλη. Σε επίπεδο εργαλείων, η Python προσφέρει τυποποιημένη διαχείριση εξαρτήσεων και απομονωμένων περιβαλλόντων, επιτρέποντας να διατηρείται σταθερό το λογισμικό ανεξάρτητα από τις βιβλιοθήκες που εξελίσσονται. Η ύπαρξη αξιόπιστων εργαλείων για δοκιμές, καταγραφή και αποσφαλμάτωση διευκολύνει τη σταδιακή προσθήκη λειτουργιών χωρίς να υποβαθμίζεται η ποιότητα.

Η Python είναι επίσης γλώσσα ικανή να διασυνδέει διαφορετικά υποσυστήματα με χαμηλό κόστος. Στον χώρο του ιστού υπάρχουν ελαφριές και βαριές προσεγγίσεις, από τα μικρά πλαίσια μέχρι τα πληρέστερα, ενώ για την αποθήκευση δεδομένων διατίθενται οδηγοί προς όλες τις δημοφιλείς σχεσιακές και μη σχεσιακές βάσεις. Αυτό επιτρέπει στον σχεδιαστή να ξεκινήσει με μια καθαρή ελάχιστη λύση και να εξελίξει την εφαρμογή σταδιακά, μόνο όπου υπάρχει πραγματική ανάγκη. Παράλληλα, η κοινότητα της Python είναι ιδιαίτερα ενεργή και προσφέρει συνεχή υποστήριξη μέσω τεκμηρίωσης, παραδειγμάτων και διαλόγου. Ο συνδυασμός αυτών των παραγόντων κάνει την Python κατάλληλη επιλογή για υλοποιήσεις που πρέπει να φτάσουν γρήγορα σε λειτουργικό αποτέλεσμα και να παραμείνουν ευέλικτες για μελλοντικές προσαρμογές.

Σε σχέση με το συγκεκριμένο έργο, η Python αποτέλεσε το φυσικό υπόβαθρο πάνω στο οποίο χτίστηκε όλη η επιχειρησιακή λογική της πλατφόρμας επιμόρφωσης. Η εφαρμογή υιοθετεί ξεκάθαρο διαχωρισμό ευθυνών: οι ροές του επιμορφωτή, οι ροές του επιμορφούμενου και η αυθεντικοποίηση οργανώνονται σε διακριτές ενότητες λογισμικού, ώστε να παραμένει σαφές ποιο κομμάτι του κώδικα εξυπηρετεί ποια διεργασία. Η χρήση της Python διευκολύνει αυτόν τον διαχωρισμό, καθώς επιτρέπει ορισμό μικρών, στοχευμένων συναρτήσεων και καθαρών διεπαφών μεταξύ τους. Για κάθε ενότητα αποφεύγονται οι περιττές εξαρτήσεις και η δομή επιλέγεται ώστε η ροή «αίτημα προς τον server, έλεγχος πρόσβασης, πρόσβαση στη βάση, επιχειρησιακός κανόνας, απόκριση» να είναι διαφανής και επαναλαμβανόμενη.

Ειδική μέριμνα δόθηκε στην αναγνωσιμότητα και τη σταθερότητα του κώδικα. Η μορφή των αποκρίσεων προς το frontend παραμένει ομοιόμορφη, πράγμα που μειώνει την πολυπλοκότητα στην πλευρά της διεπαφής. Οι έλεγχοι ρόλων τοποθετούνται στην αρχή κάθε ροής, ώστε να αποτρέπονται γρήγορα μη εξουσιοδοτημένες ενέργειες και να περιορίζεται η επιφάνεια σφαλμάτων. Για επαναλαμβανόμενες εργασίες, όπως η μορφοποίηση ημερομηνιών ή η αξιολόγηση κανόνων ολοκλήρωσης, χρησιμοποιούνται βοηθητικές ρουτίνες, με στόχο οι ίδιες συμβάσεις να εφαρμόζονται σε όλη την εφαρμογή. Αυτή η επιμέλεια είναι χαρακτηριστική ενός Python έργου που στοχεύει στη συντηρησιμότητα και όχι μόνο στην επίδειξη λειτουργικότητας.

Στον άξονα της αποθήκευσης, η Python λειτουργεί ως σταθερός ενδιάμεσος προς τη MySQL. Τα ερωτήματα είναι σαφή, κοντά στο σημείο χρήσης, και μετατρέπονται άμεσα σε δομές που εξυπηρετούν την ανάγκη του ιστού. Η επιλογή να παραμείνουν τα ερωτήματα απλά και εκφραστικά ταιριάζει με τη φιλοσοφία της γλώσσας για ευανάγνωστο, προβλέψιμο κώδικα. Η μετατροπή των αποτελεσμάτων σε ουδέτερες αναπαραστάσεις, όπως οι συμβολοσειρές ημερομηνιών σε καθορισμένη μορφή, ελαχιστοποιεί τη γνώση που απαιτείται από την πλευρά του περιηγητή και βελτιώνει την ανθεκτικότητα της εφαρμογής σε αλλαγές διεπαφής.

Η Python αξιοποιείται επίσης για την κωδικοποίηση των κανόνων που διέπουν τον κύκλο ζωής ενός επιμορφωτικού προγράμματος. Η διφασική εγγραφή, η σύνδεση ανακοινώσεων με συγκεκριμένα προγράμματα, η οργάνωση του περιεχομένου σε ιεραρχία και η λογική αξιολογήσεων με χρονικά όρια και αποθήκευση προσπαθειών εκφράζονται με καθαρές δομές ελέγχου και σύντομες, επαναχρησιμοποιήσιμες συναρτήσεις. Η γλώσσα βοηθά να διατυπωθούν αυτοί οι κανόνες με τρόπο άμεσο, χωρίς πρόβλημα από τεχνικές λεπτομέρειες, και να δοκιμαστούν εύκολα μέσω μικρών σεναρίων. Το αποτέλεσμα είναι ότι ο κώδικας δεν λειτουργεί μόνο ως μηχανισμός αλλά και ως ζωντανή τεκμηρίωση των διαδικασιών του συστήματος.

Σημαντική είναι και η συμβολή της Python στην ποιότητα του έργου. Η δυνατότητα γρήγορης καταγραφής σφαλμάτων και η ύπαρξη ολοκληρωμένων βιβλιοθηκών για logging και δοκιμές επιτρέπουν να εντοπίζονται και να διορθώνονται ζητήματα στην περίμετρο των διεπαφών, πριν αυτά διαχυθούν σε ολόκληρη την εφαρμογή. Σε κάθε σημείο όπου μπορεί να προκύψει ασυνέπεια, προτιμάται να ενσωματώνεται ένας μικρός, ξεκάθαρος έλεγχος που διακόπτει την ροή με προβλέψιμο μήνυμα, αντί να επιτρέπεται στο σφάλμα να εμφανιστεί αργότερα στο περιβάλλον χρήστη. Αυτή η πειθαρχία είναι ευκολότερο να εφαρμοστεί όταν η γλώσσα ενθαρρύνει τη λιτότητα και την αναγνωσιμότητα, όπως συμβαίνει με την Python.

Από πλευράς επεκτασιμότητας, η Python επιτρέπει να χαραχτεί μια ρεαλιστική διαδρομή εξέλιξης. Η πλατφόρμα μπορεί να υποδεχθεί επιπλέον λειτουργίες χωρίς να αλλάξει ο πυρήνας της αρχιτεκτονικής. Η προσθήκη ειδοποιήσεων, η ενοποίηση με μηχανισμούς διαπιστευτηρίων ή η εισαγωγή ενοποιημένων αναφορών προόδου μπορεί να γίνει με νέες ενότητες και στοχευμένες διεπαφές, με ελάχιστη επίδραση

στις υπάρχουσες ροές. Η Python προσφέρει επαρκή ωριμότητα εργαλείων και βιβλιοθηκών ώστε τέτοιες επεκτάσεις να παραμένουν εφικτές και διαχειρίσιμες, διατηρώντας το έργο απλό για τον προγραμματιστή και προβλέψιμο για τον τελικό χρήστη.

Με βάση όλα τα παραπάνω, καθίσταται σαφές γιατί η Python επιλέχθηκε ως θεμέλιο για το συγκεκριμένο σύστημα. Συνδυάζει ταχύτητα ανάπτυξης με καθαρή δομή, διευκολύνει τον σαφή διαχωρισμό ευθυνών, υποστηρίζει την ομοιομορφία στις διεπαφές και καθιστά απλή τη σύνδεση με τη βάση και το περιβάλλον του ιστού. Η ίδια η γλώσσα ενθαρρύνει τις αρετές που χρειάζεται μια πλατφόρμα επιμόρφωσης: λιτότητα, προβλεψιμότητα, συντηρησιμότητα και ευκολία εξέλιξης. Στο συγκεκριμένο έργο, αυτές οι ιδιότητες μεταφράστηκαν σε μία υλοποίηση που όχι μόνο λειτουργεί αξιόπιστα, αλλά παραμένει έτοιμη να υποστηρίξει τα επόμενα βήματα χωρίς να θυσιάζει την απλότητα που την χαρακτηρίζει.

2.4.2 Flask

Το Flask είναι ένα ελαφρύ web framework για Python που δίνει τα απολύτως απαραίτητα για να υλοποιηθεί μια καθαρή, προβλέψιμη web εφαρμογή: routing (αντιστοίχιση URL σε συναρτήσεις), request/response αντικείμενα, templates με Jinja2, μηχανισμό session και έναν απλό κύκλο ζωής αιτήματος. Η φιλοσοφία του δεν επιβάλλει ORM, scheduler ή σύνθετα patterns και αφήνει την ομάδα να επιλέξει βιβλιοθήκες και να κρατήσει τον κώδικα όσο λιτό ή όσο πλούσιο απαιτείται. Αυτό ταιριάζει σε έργα που θέλουν γρήγορο ρυθμό ανάπτυξης, καθαρό διαχωρισμό ευθυνών και ελεγχόμενη πολυπλοκότητα.[11-14]

Στον πυρήνα του Flask βρίσκεται η έννοια του route: μια διαδρομή URL που δένει ένα HTTP μέθοδο (GET/POST/…) με μια συνάρτηση Python. Η συνάρτηση διαβάζει εισόδους (query params, form, JSON), εφαρμόζει επιχειρησιακή λογική, μιλά με τη βάση και επιστρέφει είτε HTML (μέσω template) είτε JSON. Το Jinja2 επιτρέπει να αποδώσουμε δυναμικές σελίδες με μεταβλητές και απλά blocks, ενώ ο μηχανισμός session μάς δίνει τρόπο να θυμόμαστε ελαφριά στοιχεία κατάστασης ανά χρήστη (π.χ. ρόλο/ταυτότητα). Για μεγαλύτερα έργα, τα Blueprints σπάνε την εφαρμογή σε υπο-«μικρο-εφαρμογές» με δικά τους routes και templates, ώστε να διατηρείται καθαρή δομή φακέλων και να αποφεύγεται ένα τεράστιο, δυσανάγνωστο αρχείο.

Στο δικό μας σύστημα, το Flask χρησιμοποιείται ακριβώς με αυτόν τον τρόπο: έχουμε ξεκάθαρα Blueprints για την αυθεντικοποίηση, τον επιμορφωτή και τον επιμορφούμενο. Κάθε ρόλος έχει το δικό του namespace (π.χ. /teacher, /student), τα δικά του templates και τα δικά του REST-like endpoints που επιστρέφουν JSON. Η διεπαφή χρήστη (HTML/Bootstrap/JavaScript) καταναλώνει αυτά τα endpoints με ασύγχρονα αιτήματα και ενημερώνει δυναμικούς πίνακες. Ο έλεγχος πρόσβασης είναι συνειδητά «νωρίς» σε κάθε route (έλεγχος session/ρόλου πριν από οποιαδήποτε βάση ή business rule), ώστε τα

σφάλματα να είναι προβλέψιμα και η επιφάνεια κινδύνου μικρή. Η συνεργασία με τη MySQL υλοποιείται μέσω μικρών helper συναρτήσεων: οι routes παραμένουν ευανάγνωστες, ενώ το I/O προς τη βάση είναι συγκεντρωμένο και επαναχρησιμοποιήσιμο.

Μια ακόμα επιλογή που ευνοεί το Flask είναι η σαφήνεια στη ροή: το αίτημα διαβάζεται, επικυρώνεται, εφαρμόζεται ο κανόνας, γράφεται/διαβάζεται η βάση, επιστρέφεται απάντηση. Για τα templates προτιμήσαμε απλά Jinja views για σελίδες πλοήγησης (dashboard, λίστες, φόρμες) και JSON για δεδομένα πινάκων/ενεργειών. Έτσι, ο browser φορτώνει γρήγορα το «σκελετό» της σελίδας και γεμίζει τα περιεχόμενα με AJAX, χωρίς περιττές επαναφορτώσεις.

Από πλευράς οργάνωσης, τα Blueprints έδωσαν ξεκάθαρη δομή: teacher.py για διαχείριση προγραμμάτων, ενοτήτων, περιεχομένου, αξιολογήσεων, αιτήσεων και πιστοποιητικών· student.py για προβολή/αίτηση προγραμμάτων, υλικών, αξιολογήσεων, βεβαιώσεων και events· auth.py για login/logout και γέμισμα του session. Η διαμόρφωση (config) παραμένει απλή, με βασικά μυστικά και παραμέτρους βάσης, ώστε το έργο να σηκώνεται εύκολα σε τοπικό ή production περιβάλλον. Η καταγραφή σφαλμάτων (logging) τοποθετείται στην περίμετρο των endpoints: συλλαμβάνουμε το exception, επιστρέφουμε δομημένη JSON απόκριση με κατάλληλο status και καταγράφουμε trace για διάγνωση.

Η κλιμάκωση σε μεγαλύτερες ανάγκες δεν απαιτεί αλλαγή φιλοσοφίας: το Flask τρέχει πίσω από WSGI/application servers (π.χ. gunicorn/uwsgi) με reverse proxy (π.χ. Nginx), και μπορεί να συνδυαστεί με cache ή task queues αν χρειαστεί. Στο παρόν έργο δεν επιλέξαμε βαρύ ORM· κρατήσαμε απλά, εκφραστικά SQL ερωτήματα, ώστε τα δεδομένα να αντιστοιχούν ευθέως στις οντότητες του συστήματος και τα joins να είναι ορατά στον προγραμματιστή. Με αυτή τη «λιτή» στοίβα, το Flask παραμένει ευχάριστο στην ανάπτυξη και σταθερό στην πράξη.

Παρακάτω ένα μικρό, αντιπροσωπευτικό παράδειγμα από τη ροή «ο επιμορφούμενος υποβάλλει αίτηση εγγραφής σε πρόγραμμα». Φαίνεται καθαρά το pattern του Flask: route με μέθοδο POST, έλεγχος ρόλου, επικύρωση εισόδου, απλές κλήσεις προς τη βάση, δομημένη JSON απόκριση.

```
from flask import Blueprint, request, session, jsonify, abort
from db import fetch_one, execute

student_bp = Blueprint('student', __name__, url_prefix='/student')

def _require_student():
    if session.get('role') != 'student':
        abort(403)  # απαγόρευση πρόσβασης για μη επιμορφούμενο

@student_bp.route('/api/registrations/create', methods=['POST'])
def create_registration():
    _require_student()
    sid = session.get('user_id')
    pid = request.form.get('program_id', type=int)
```

```
# ελάχιστη επικύρωση εισόδου
if not pid:
    return jsonify(ok=False, error="Λείπει το πρόγραμμα"), 400

# έλεγχος ότι το πρόγραμμα είναι δημοσιευμένο/διαθέσιμο
prog = fetch_one("SELECT id FROM programs WHERE id=%s AND is_published=1", (pid,))
if not prog:
    return jsonify(ok=False, error="Μη διαθέσιμο πρόγραμμα"), 404

# αποφυγή διπλής αίτησης ή ήδη εγγεγραμμένου
dup = fetch_one("""
    SELECT 1 FROM registrations
    WHERE program_id=%s AND student_id=%s AND status IN ('pending','approved')
""", (pid, sid))
if dup:
    return jsonify(ok=False, error="Υπάρχει ήδη αίτηση ή εγγραφή"), 409

# καταχώριση αίτησης ως pending
execute("INSERT INTO registrations (program_id, student_id, status) VALUES (%s,%s,'pending')", (pid, sid))
return jsonify(ok=True, msg="Η αίτηση υποβλήθηκε")
```

Το παράδειγμα έχει τις βασικές αρχές που ακολουθούμε σε όλο το έργο. Ο έλεγχος πρόσβασης τερματίζει νωρίς μη επιτρεπτές κλήσεις. Η επικύρωση εισόδου είναι ρητή και επιστρέφει ξεκάθαρα HTTP status με φιλικό μήνυμα. Οι κλήσεις προς τη βάση είναι απλές και διάφανες, χωρίς «κρυφή» λογική. Η απάντηση είναι πάντα JSON στην ίδια μορφή (ok, msg/error) ώστε το frontend να την καταναλώνει ομοιόμορφα. Η ίδια δομή μπορεί να επαναληφθεί για ανακοινώσεις, υλικό, αξιολογήσεις και έκδοση πιστοποιητικών, κρατώντας κοινό ρυθμό σε ολόκληρη την εφαρμογή.

Το Flask μας επέτρεψε να υλοποιήσουμε μια πλατφόρμα επιμόρφωσης με καθαρή αρχιτεκτονική και χαμηλή πολυπλοκότητα. Τα Blueprints οριοθετούν ευθύνες, τα templates δίνουν γρήγορη απόδοση σελίδων, τα JSON endpoints τροφοδοτούν τη διεπαφή με ζωντανά δεδομένα και ο κύκλος ζωής αιτήματος παραμένει διαυγής. Αυτή η ισορροπία απλότητας και ευελιξίας είναι ο λόγος που το Flask «κουμπώνει» ιδανικά με τις απαιτήσεις του έργου: κάνει τα βασικά σωστά, δεν εμποδίζει επεκτάσεις και κρατά τον κώδικα προσβάσιμο στον επόμενο που θα τον αναλάβει.

2.4.3 MySQL

Η MySQL είναι η βάση δεδομένων που κρατά όλα τα στοιχεία της πλατφόρμας μας. Είναι γρήγορη, σταθερή και δωρεάν. Αποθηκεύει δεδομένα σε «πίνακες» με γραμμές και στήλες, όπως ένα υπολογιστικό φύλλο. Επιλέξαμε MySQL γιατί είναι εύκολη στη ρύθμιση, έχει καλή τεκμηρίωση και δουλεύει πολύ καλά με εφαρμογές σε Python/Flask. Στο έργο μας κρατά: χρήστες, προγράμματα, αιτήσεις και εγγραφές, ανακοινώσεις, υλικό μαθημάτων, κουίζ, προσπάθειες και πιστοποιητικά.

Χρησιμοποιούμε κωδικοποίηση utf8mb4 για σωστά ελληνικά και σύμβολα, και απλά DATE/DATETIME πεδία για χρονικές πληροφορίες. [15-16]

Η λογική είναι καθαρή: το Flask λαμβάνει ένα αίτημα, ελέγχει τον ρόλο του χρήστη και εκτελεί ένα έτοιμο SQL ερώτημα προς τη MySQL με παραμέτρους. Δεν χτίζουμε strings «στο χέρι», για να μην υπάρχουν θέματα ασφάλειας (SQL injection). Τα ερωτήματα επιστρέφουν «λεξικά» (key/value), τα μετατρέπουμε σε JSON και τα στέλνουμε στο frontend. Κρατάμε τα ερωτήματα κοντά στα routes για να είναι ευανάγνωστα: βλέπεις τι ζητάς από τη βάση και τι στέλνεις πίσω στον χρήστη.

Η δομή των πινάκων είναι κανονικοποιημένη και ακολουθεί τις οντότητες της εφαρμογής. Ο πίνακας users έχει βασικά στοιχεία και ρόλο (student/teacher/admin). Ο programs συνδέει κάθε πρόγραμμα με τον επιμορφωτή του και τις ημερομηνίες. Οι registrations κρατούν αιτήσεις (pending/approved/rejected). Όταν εγκριθεί μια αίτηση, δημιουργείται εγγραφή στον πίνακα enrollments. Τα announcements ανήκουν σε πρόγραμμα. Το περιεχόμενο οργανώνεται ανά ενότητα/τεκμήριο. Τα quizzes έχουν ερωτήσεις και επιλογές, και οι προσπάθειες/απαντήσεις καταγράφονται ξεχωριστά για να βγαίνει σωστά το αποτέλεσμα. Τα certificates εκδίδονται όταν ικανοποιούνται οι κανόνες ολοκλήρωσης. Τα κλειδιά (π.χ. program_id, student_id) έχουν ευρετήρια (indexes) για γρήγορα joins.

Κρατάμε απλές και καθαρές σχέσεις με ξένα κλειδιά (foreign keys), ώστε αν διαγραφεί ένα πρόγραμμα να μην μένουν «ορφανά» δεδομένα. Όπου χρειάζεται, χρησιμοποιούμε ON DELETE κανόνες (π.χ. RESTRICT ή CASCADE) ανάλογα με το τι είναι ασφαλές για το σενάριο. Στις εγγραφές ενημερώνουμε μόνο ό,τι πρέπει (INSERT/UPDATE), και κάθε τέτοια πράξη γίνεται μέσα σε συναλλαγή της MySQL για σιγουριά. Για αναφορές και λίστες, αφήνουμε τη MySQL να κάνει το φιλτράρισμα και την ταξινόμηση, για να φορτώνουμε λιγότερα δεδομένα στην εφαρμογή.

Για αντίγραφα ασφαλείας, η MySQL παρέχει απλά εργαλεία (π.χ. mysqldump) και μπορεί να στηθεί εύκολα σε περιβάλλον ανάπτυξης και παραγωγής. Το σχήμα της βάσης παραμένει διαυγές και ευανάγνωστο: κάθε πίνακας αντιστοιχεί σε κάτι που ο χρήστης βλέπει στην εφαρμογή (πρόγραμμα, ανακοίνωση, κουίζ, πιστοποιητικό). Αυτό μειώνει τον χρόνο διάγνωσης σφαλμάτων: από μια σελίδα ή μια λειτουργία πηγαίνεις κατευθείαν στον αντίστοιχο πίνακα και στο σχετικό ερώτημα.

Χρησιμοποιούμε τη MySQL με απλό και πειθαρχημένο τρόπο: καθαρές σχέσεις, παραμετρικά ερωτήματα, σταθερές μορφές δεδομένων και indexes στα σωστά σημεία. Αυτή η προσέγγιση κρατά την πλατφόρμα γρήγορη, ασφαλή και εύκολη στη συντήρηση. Αν χρειαστεί αργότερα να προστεθεί αναζήτηση πλήρους κειμένου, στατιστικές αναφορές ή replication, η MySQL δίνει ξεκάθαρο δρόμο χωρίς να αλλάξει ο πυρήνας της εφαρμογής.

Κεφάλαιο 3ο: Το έργο – η πλατφόρμα

3.1 Περιγραφή

Η παρούσα πτυχιακή εργασία ασχολείται με την ανάπτυξη μιας εκπαιδευτικής πλατφόρμας επιμόρφωσης, η οποία έχει στόχο να φέρει πιο κοντά τον εκπαιδευτή με τον εκπαιδευόμενο μέσα από ένα εύχρηστο περιβάλλον. Η ιδέα ξεκίνησε από την ανάγκη που βλέπουμε συχνά στη διδασκαλία: πολλοί καθηγητές θέλουν να οργανώνουν υλικό, τεστ και ανακοινώσεις για τους μαθητές τους, αλλά οι λύσεις που υπάρχουν είναι είτε πολύπλοκες είτε δεν ταιριάζουν στις πραγματικές ανάγκες. Έτσι δημιουργήσαμε κάτι πιο απλό αλλά και πρακτικό.

Η πλατφόρμα που υλοποιήσαμε επιτρέπει σε δύο βασικούς ρόλους, τον Teacher και τον Student, να συνεργάζονται με σαφή και οργανωμένο τρόπο. Ο Teacher είναι υπεύθυνος να δημιουργεί τα προγράμματα επιμόρφωσης, να ορίζει τίτλο, περιγραφή και χρονικό διάστημα διεξαγωγής. Μέσα σε κάθε πρόγραμμα μπορεί να προσθέτει διάφορα στοιχεία, όπως υλικό σε μορφή αρχείων (PDF, Word, PowerPoint), πολυμέσα (π.χ. βίντεο YouTube), ανακοινώσεις αλλά και quiz. Από την άλλη, ο Student έχει τη δυνατότητα να βλέπει ποια προγράμματα είναι ενεργά και να κάνει αίτηση για να εγγραφεί. Αν εγκριθεί, τότε έχει πλήρη πρόσβαση στο περιεχόμενο που ανέβασε ο καθηγητής.

Σημαντικό κομμάτι της πλατφόρμας είναι τα quiz, τα οποία χωρίζονται σε δύο είδη: test quiz για εξάσκηση και exam quiz για την τελική αξιολόγηση. Ο εκπαιδευόμενος πρέπει να περάσει έναν ελάχιστο αριθμό από exams με βαθμολογία πάνω από 50%, ώστε να θεωρηθεί ότι ολοκλήρωσε με επιτυχία το πρόγραμμα. Στην περίπτωση αυτή, η πλατφόρμα εκδίδει αυτόματα ένα πιστοποιητικό, το οποίο επιβεβαιώνει την επιτυχία του χρήστη. Έτσι, το σύστημα δεν περιορίζεται μόνο στην παράδοση υλικού αλλά συνδυάζει και αξιολόγηση.

Τεχνολογικά, το έργο υλοποιήθηκε σε Python με το framework Flask και χρησιμοποιεί MySQL για τη βάση δεδομένων. Για το περιβάλλον χρήστη αξιοποιήθηκαν βιβλιοθήκες όπως Bootstrap για το σχεδιασμό, DataTables για την εμφάνιση των δεδομένων σε πίνακες και Chart.js για την οπτικοποίηση στατιστικών με γραφήματα.

Μέσα από την εργασία μάθαμε στην πράξη πώς να οργανώνουμε ένα έργο με πολλούς ρόλους, πώς να φτιάχνουμε backend που συνδέεται με βάση δεδομένων και πώς να συνδυάζουμε διάφορες βιβλιοθήκες στο frontend για να κάνουμε το περιβάλλον πιο εύχρηστο. Η όλη διαδικασία είχε αρκετές δυσκολίες, από απλά λάθη στον κώδικα μέχρι τον σχεδιασμό της βάσης, αλλά τελικά τα ξεπεράσαμε βήμα βήμα. Το αποτέλεσμα είναι μια ολοκληρωμένη πλατφόρμα που μπορεί να χρησιμοποιηθεί για πραγματικά προγράμματα επιμόρφωσης και να βοηθήσει τόσο τους καθηγητές όσο και τους φοιτητές.

Η πλατφόρμα δεν είναι απλά μια “σελίδα με αρχεία”. Προσπαθήσαμε να φτιάξουμε ένα ολοκληρωμένο περιβάλλον, όπου ο κάθε ρόλος (teacher ή student) να νιώθει ότι έχει τον δικό του χώρο. Για τον λόγο αυτό, φτιάξαμε δύο βασικά dashboards: το Teacher Dashboard και το Student Dashboard.

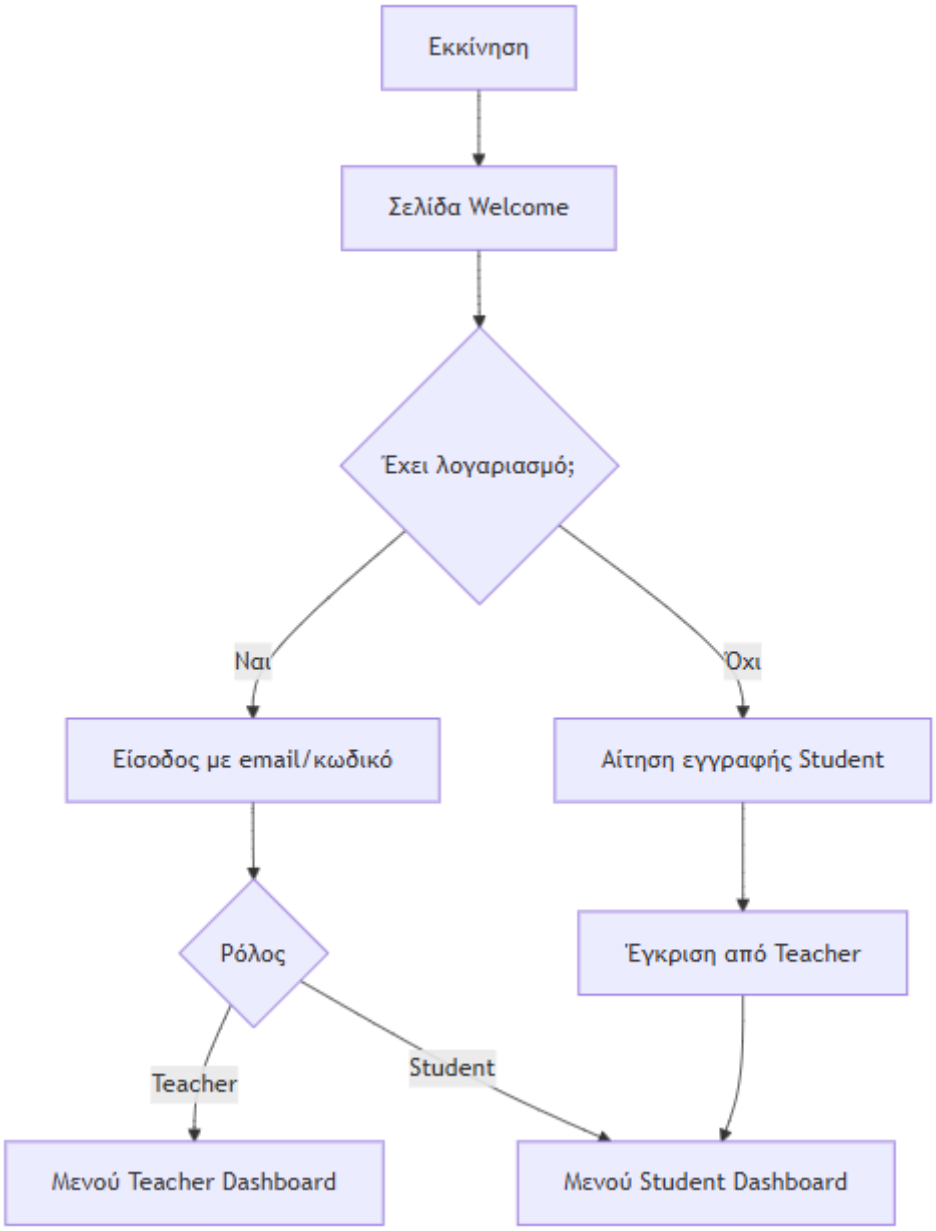
Στο Teacher Dashboard ο καθηγητής βλέπει μαζεμένα όλα τα προγράμματά του, με γραφήματα για εγγραφές, για τα quiz που έχουν ολοκληρωθεί, καθώς και για την επιτυχία των εκπαιδευόμενων στα exams. Έτσι μπορεί να παρακολουθεί εύκολα πώς πηγαίνει το μάθημα συνολικά. Από εκεί μπορεί με απλό τρόπο να δημιουργήσει νέο πρόγραμμα, να προσθέσει υλικό, να ανεβάσει αρχεία, να δημοσιεύσει ανακοίνωση ή να φτιάξει quiz. Όλα αυτά γίνονται μέσα από φόρμες που είναι σχετικά απλές και δεν μπερδεύουν. Ένα σημαντικό σημείο είναι ότι ο Teacher έχει και τον έλεγχο των αιτήσεων εγγραφής: βλέπει ποιοι φοιτητές έκαναν αίτηση και μπορεί να τους εγκρίνει ή να τους απορρίψει.

Το Student Dashboard, από την άλλη, είναι πιο απλό και προσαρμοσμένο στον εκπαιδευόμενο. Ο χρήστης βλέπει ποια προγράμματα είναι ενεργά και σε ποια έχει κάνει εγγραφή. Μόλις εγκριθεί, έχει πρόσβαση στη σελίδα του προγράμματος, όπου βρίσκει οργανωμένα όλα: υλικό, πολυμέσα, ανακοινώσεις, ημερολόγιο με events, καθώς και τα quiz. Έτσι δεν χρειάζεται να ψάχνει δεξιά και αριστερά – όλα βρίσκονται συγκεντρωμένα σε ένα σημείο. Ένα επιπλέον στοιχείο είναι ότι το σύστημα εμφανίζει πόσα exams έχει περάσει και αν πλησιάζει στο όριο για να πάρει το πιστοποιητικό. Αυτό βοηθάει τον εκπαιδευόμενο να έχει κίνητρο και να ξέρει πού βρίσκεται.

Στην καθημερινή χρήση, τα σενάρια είναι αρκετά απλά: ο Teacher δημιουργεί το πρόγραμμα, ο Student κάνει αίτηση, και όταν εγκριθεί ξεκινάει να βλέπει το υλικό και να συμμετέχει σε quiz. Αν κάτι αλλάξει (π.χ. μια νέα ανακοίνωση ή μια αλλαγή στην ώρα ενός event), ο Student το βλέπει αμέσως στη σελίδα του. Αντίστοιχα, όταν ο Student ολοκληρώσει ένα exam quiz, η βαθμολογία του καταγράφεται και επηρεάζει την τελική του πρόοδο. Στο τέλος, αν πληρούνται τα κριτήρια, το σύστημα εκδίδει πιστοποιητικό αυτόματα.

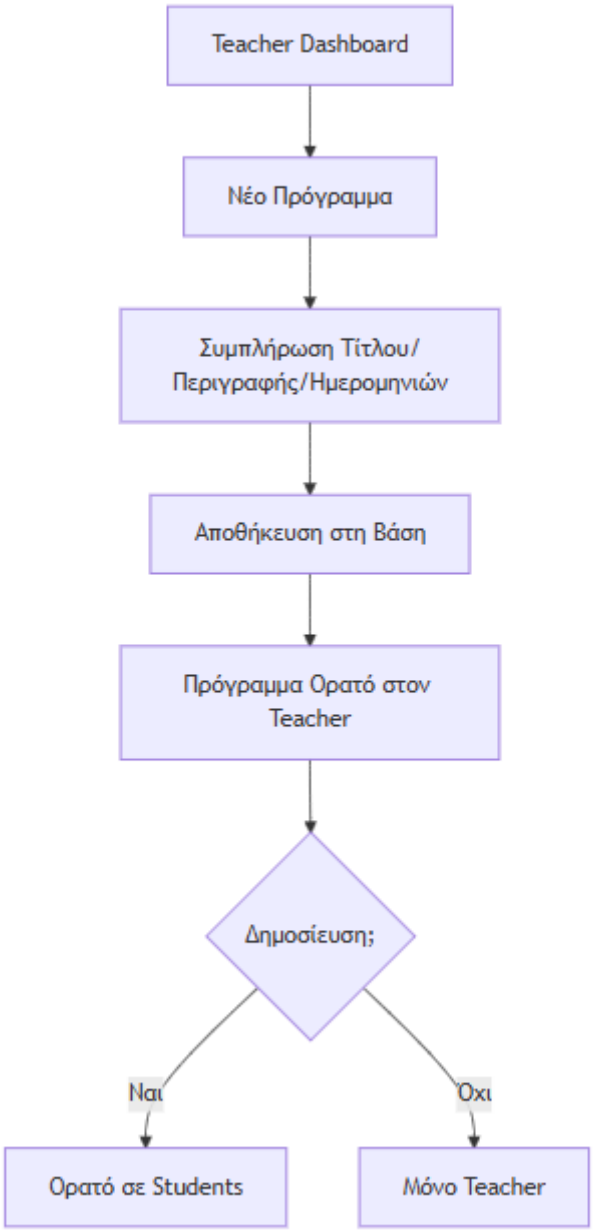
Τεχνικά, όλο αυτό έγινε με Flask στο backend, MySQL για την αποθήκευση δεδομένων, και στο frontend φροντίσαμε να είναι όσο πιο “ζωντανό” γίνεται. Για παράδειγμα, οι πίνακες των δεδομένων (προγράμματα, ανακοινώσεις, events κλπ.) χρησιμοποιούν DataTables, ώστε να έχουν αναζήτηση και ταξινόμηση. Επίσης, χρησιμοποιήσαμε Chart.js για να δείχνουμε όμορφα γραφήματα στο dashboard, ώστε ο καθηγητής να έχει μια πιο ξεκάθαρη εικόνα με μια ματιά.

Η πλατφόρμα που δημιουργήσαμε δεν είναι απλά μια πρόχειρη υλοποίηση αλλά μπορεί να χρησιμοποιηθεί σε πραγματικές συνθήκες. Μπορεί ένας καθηγητής να οργανώσει μαθήματα, οι φοιτητές να συμμετέχουν, να αξιολογούνται και να παίρνουν πιστοποιητικά. Αυτό κάνει το έργο μας να έχει πρακτική αξία και όχι μόνο ακαδημαϊκή.



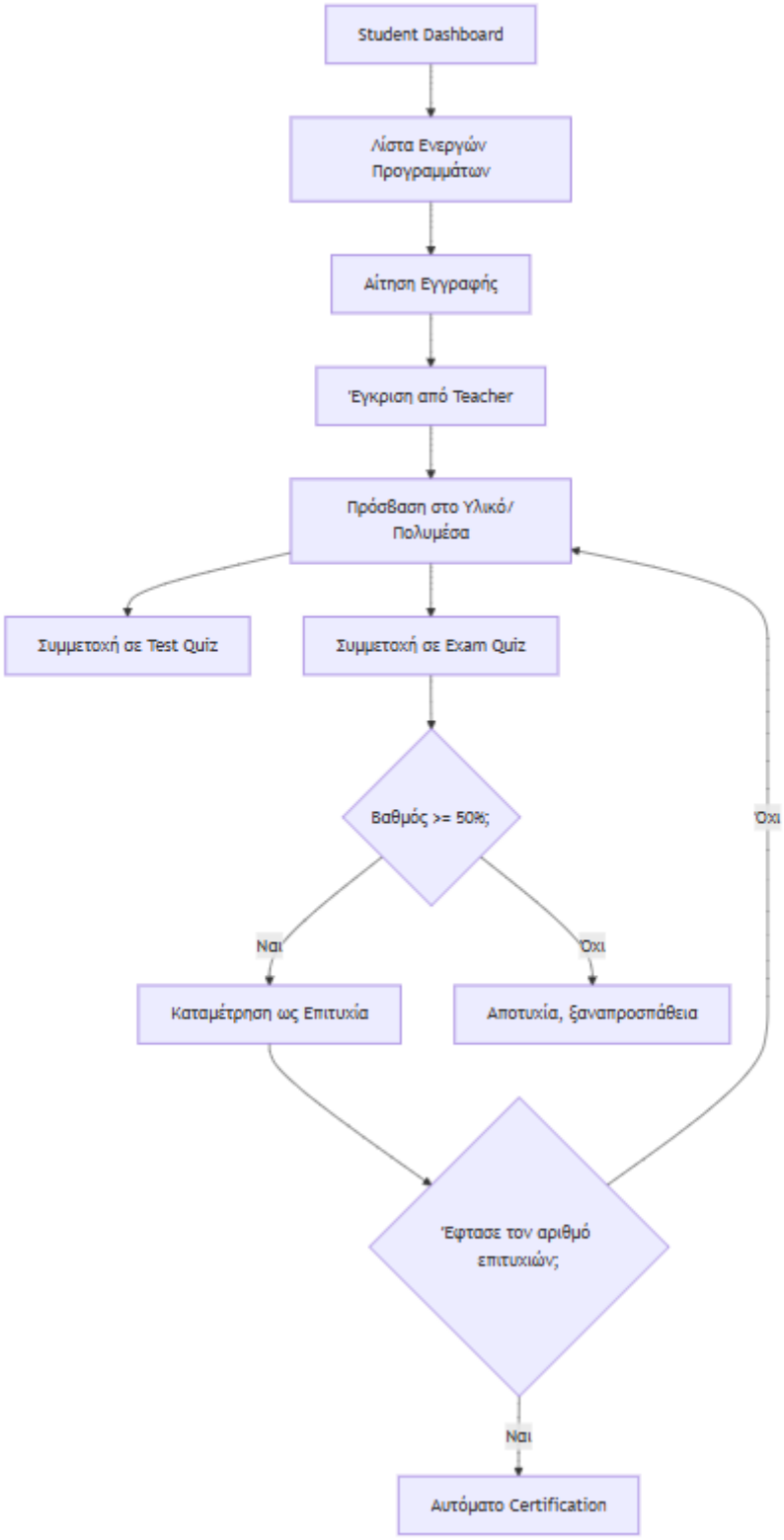
Εικόνα 3.1: Ροή σύνδεσης χρήστη

Η εικόνα 3.1. παρουσιάζει τη βασική διαδικασία σύνδεσης στην πλατφόρμα. Ο χρήστης ξεκινά από τη σελίδα υποδοχής (welcome) και αν έχει λογαριασμό εισάγει τα στοιχεία του για να προχωρήσει. Αν δεν έχει, τότε μπορεί να κάνει αίτηση εγγραφής ως Student, η οποία πρέπει να εγκριθεί από τον Teacher. Ανάλογα με τον ρόλο του (Teacher ή Student), το σύστημα τον κατευθύνει στο αντίστοιχο dashboard με τις διαθέσιμες επιλογές.



Εικόνα 3.2: Ροή δημιουργίας προγράμματος από Teacher

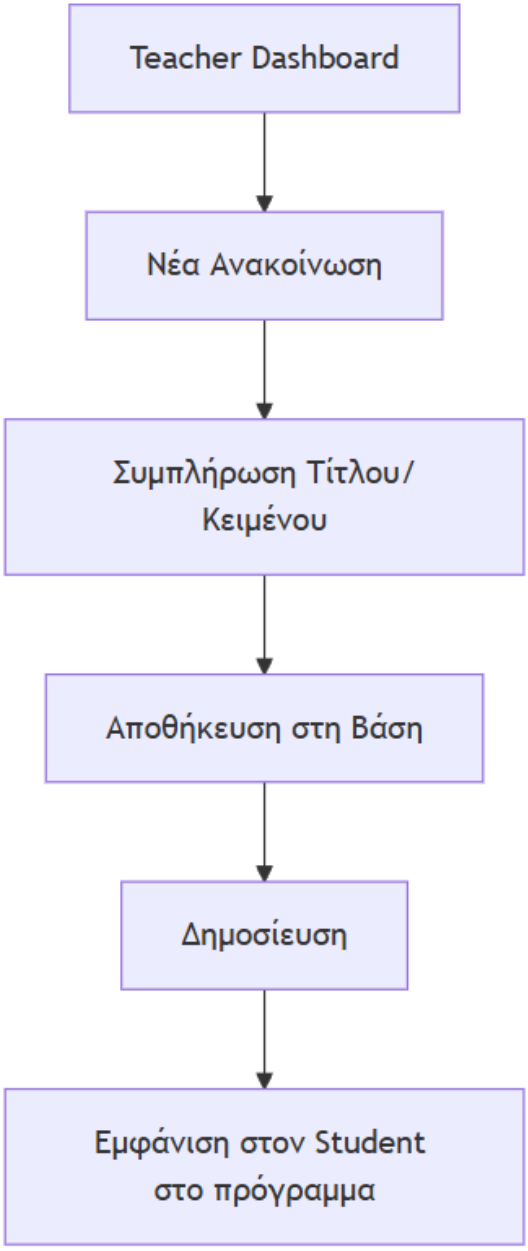
Η εικόνα 3.2 δείχνει πώς ο Teacher δημιουργεί ένα νέο πρόγραμμα επιμόρφωσης. Από το dashboard επιλέγει «Νέο Πρόγραμμα», εισάγει τίτλο, περιγραφή και ημερομηνίες έναρξης/λήξης και αποθηκεύει. Το πρόγραμμα αποθηκεύεται στη βάση δεδομένων και μπορεί να παραμείνει προσωρινά μόνο ορατό στον Teacher ή να δημοσιευθεί ώστε να το βλέπουν οι Students και να κάνουν αίτηση εγγραφής.



Εικόνα 3.3: Ροή συμμετοχής Student σε πρόγραμμα

Η εικόνα 3.3 αποτυπώνει τα βήματα που ακολουθεί ένας Student για να συμμετάσχει σε ένα πρόγραμμα. Ο φοιτητής βλέπει τη λίστα ενεργών προγραμμάτων, κάνει αίτηση και μετά την έγκριση

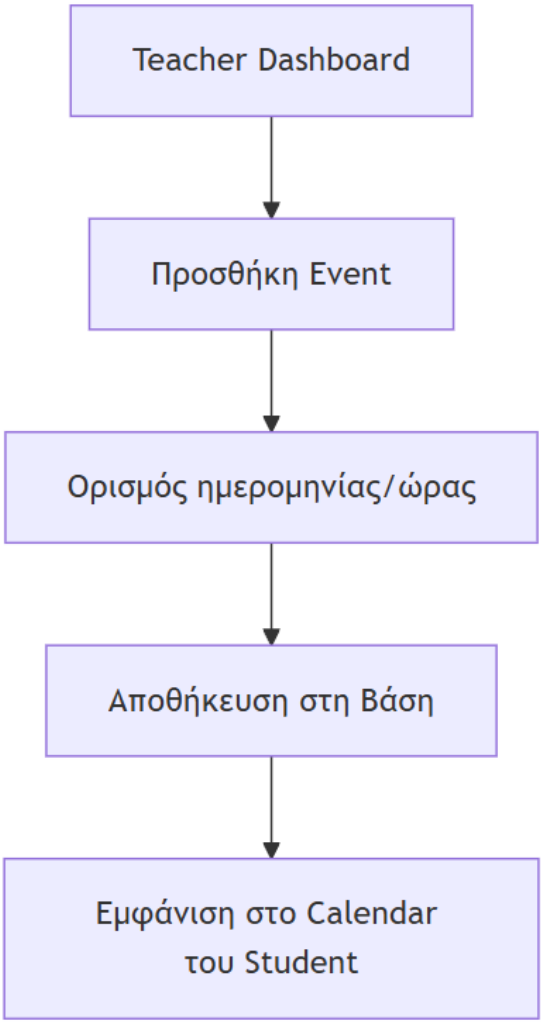
από τον Teacher αποκτά πρόσβαση στο υλικό, τα πολυμέσα και τα quiz. Κατά τη διάρκεια του προγράμματος μπορεί να συμμετέχει σε test quiz για εξάσκηση ή σε exam quiz για αξιολόγηση. Αν περάσει επιτυχώς τα απαιτούμενα exam quiz με βαθμολογία άνω του 50%, το σύστημα του εκδίδει αυτόματα πιστοποιητικό.



Εικόνα 3.4: Ροή Διαχείρισης Ανακοινώσεων

Η εικόνα 3.4 περιγράφει πώς ένας Teacher δημιουργεί και δημοσιεύει μια ανακοίνωση. Ο καθηγητής εισάγει τίτλο και κείμενο, τα αποθηκεύει στη βάση και στη συνέχεια η ανακοίνωση εμφανίζεται στους

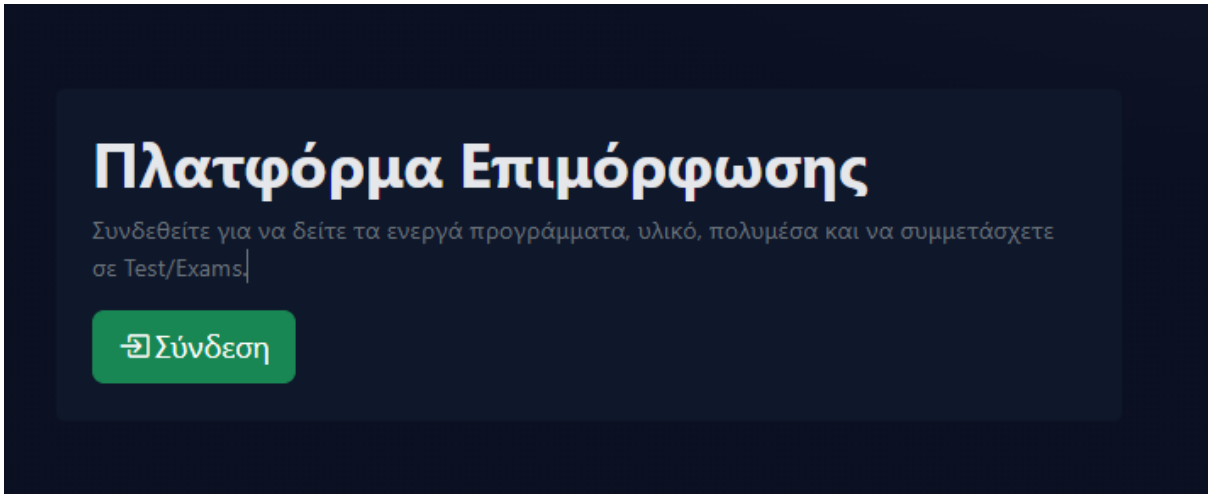
Students που είναι εγγεγραμμένοι στο συγκεκριμένο πρόγραμμα. Με τον τρόπο αυτό οι ανακοινώσεις λειτουργούν ως κεντρικό κανάλι ενημέρωσης για αλλαγές, υπενθυμίσεις ή οδηγίες.



Εικόνα 3.5: Ροή Events (Ημερολόγιο)

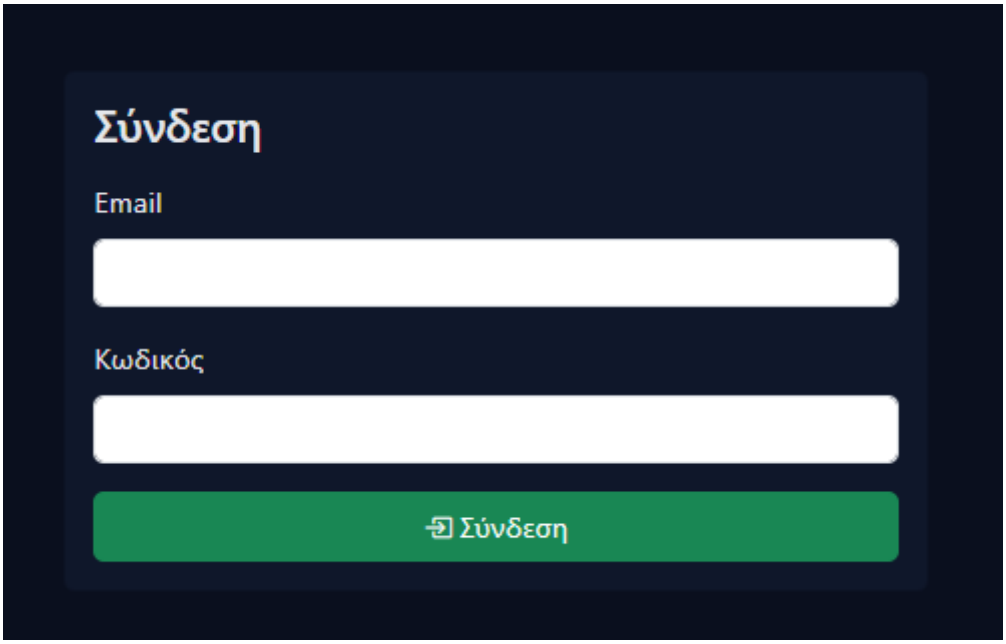
Η εικόνα 3.5 δείχνει τη διαδικασία διαχείρισης events από τον Teacher. Ο εκπαιδευτής μπορεί να προσθέσει μια νέα δραστηριότητα ή εκδήλωση, να ορίσει ημερομηνία και ώρα και να την αποθηκεύσει. Στη συνέχεια το event εμφανίζεται στο ημερολόγιο των Students, ώστε να είναι όλοι ενημερωμένοι για προγραμματισμένα μαθήματα, εξετάσεις ή άλλες σημαντικές δράσεις.

3.2 Η πλατφόρμα



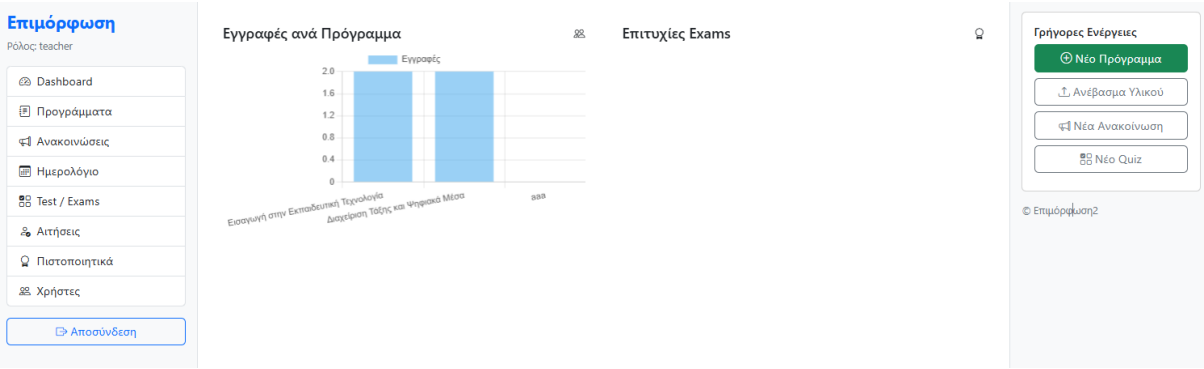
Εικόνα 3.6: Η σελίδα εισαγωγής - welcome

Στην Εικόνα 3.6 απεικονίζεται η αρχική σελίδα καλωσορίσματος της πλατφόρμας. Ο χρήστης που εισέρχεται στον ιστότοπο βλέπει μια γενική παρουσίαση του συστήματος και καλείται να προχωρήσει είτε σε σύνδεση είτε σε εγγραφή, ανάλογα με τον ρόλο του.



Εικόνα 3.7: Η σελίδα για τη σύνδεση - login

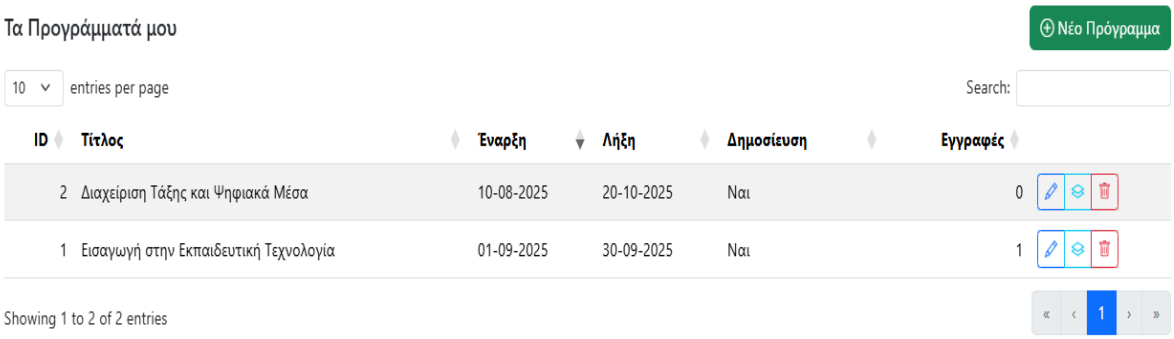
3.2.1 Από την πλευρά του Teacher - Επιμορφωτή



Εικόνα 3.8: Σελίδα Dashboard – Επιμορφωτής

Όπως φαίνεται στην Εικόνα 3.7, πρόκειται για τη σελίδα σύνδεσης των χρηστών στην πλατφόρμα. Παρέχεται η δυνατότητα εισαγωγής email και κωδικού πρόσβασης, ενώ υπάρχει και ένδειξη για επαναφορά κωδικού σε περίπτωση που ο χρήστης τον έχει ξεχάσει.

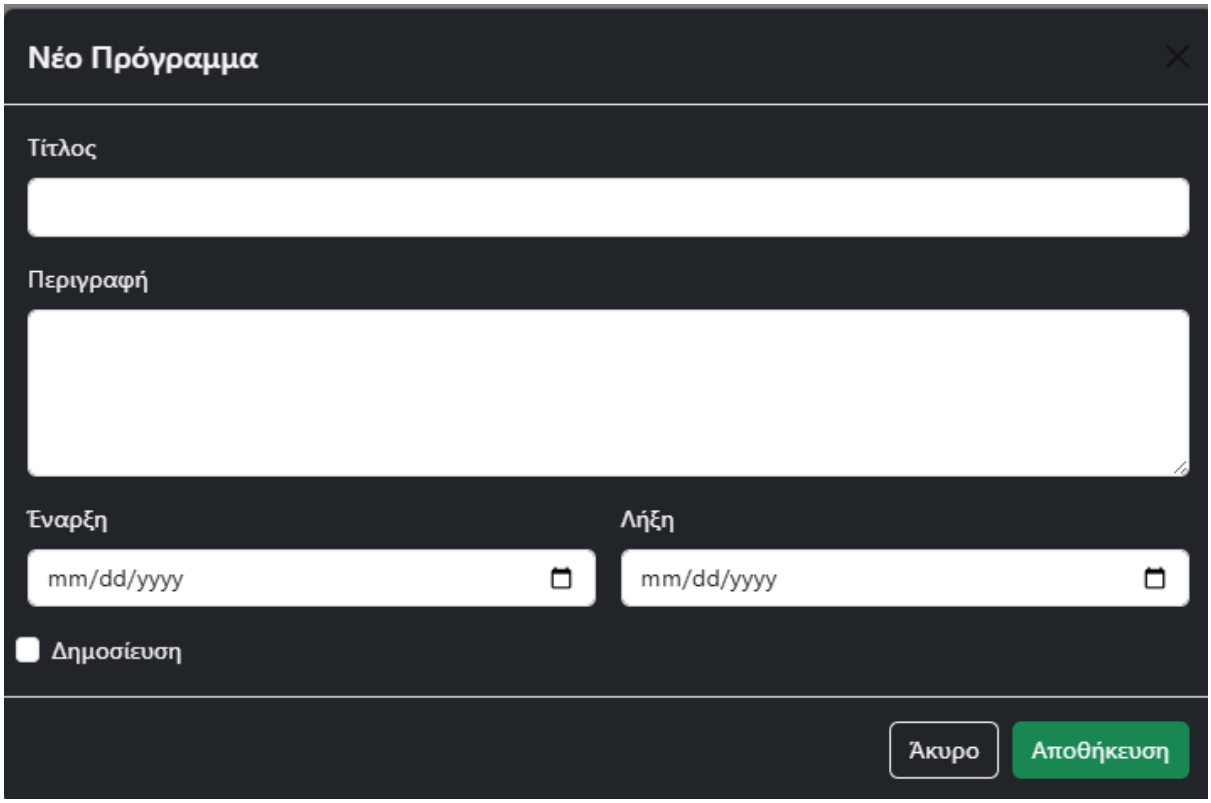
Παρουσιάζεται στην Εικόνα 3.8 η βασική σελίδα πίνακα ελέγχου (Dashboard) του επιμορφωτή. Από το σημείο αυτό, ο επιμορφωτής έχει πρόσβαση σε όλες τις βασικές λειτουργίες που του παρέχει η πλατφόρμα, όπως τη δημιουργία και επεξεργασία προγραμμάτων, την ανάρτηση περιεχομένου και την παρακολούθηση της δραστηριότητας των επιμορφούμενων.



Εικόνα 3.9: Σελίδα με τα προγράμματα του επιμορφωτή – Επιμορφωτής

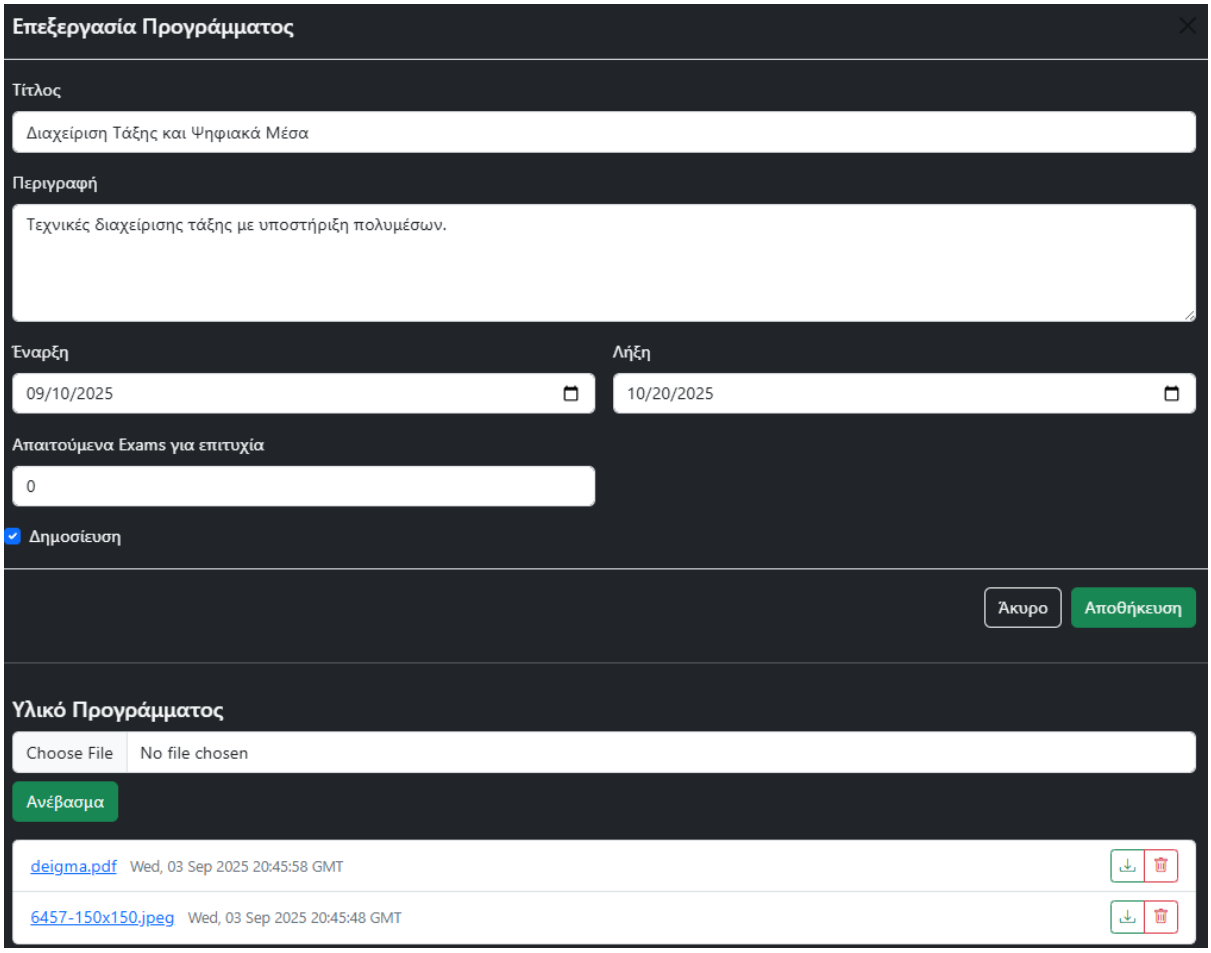
Η Εικόνα 3.9 δείχνει τη σελίδα με τα προγράμματα που έχει δημιουργήσει ή διαχειρίζεται ο επιμορφωτής. Παρουσιάζονται βασικά στοιχεία κάθε προγράμματος, όπως ο τίτλος, οι ημερομηνίες

και η κατάστασή του. Από εδώ ο επιμορφωτής μπορεί να επιλέξει πρόγραμμα για επεξεργασία ή να προχωρήσει σε δημιουργία νέου.



Εικόνα 3.10: Φόρμα για νέο Πρόγραμμα – Επιμορφωτής

Στην Εικόνα 3.10 παρουσιάζεται η φόρμα δημιουργίας νέου επιμορφωτικού προγράμματος. Ο επιμορφωτής εισάγει πληροφορίες όπως τίτλο, περιγραφή, ημερομηνίες και ρυθμίσεις δημοσίευσης. Η φόρμα είναι απλή και κατευθυνόμενη, ώστε να διευκολύνεται η διαδικασία εισαγωγής.



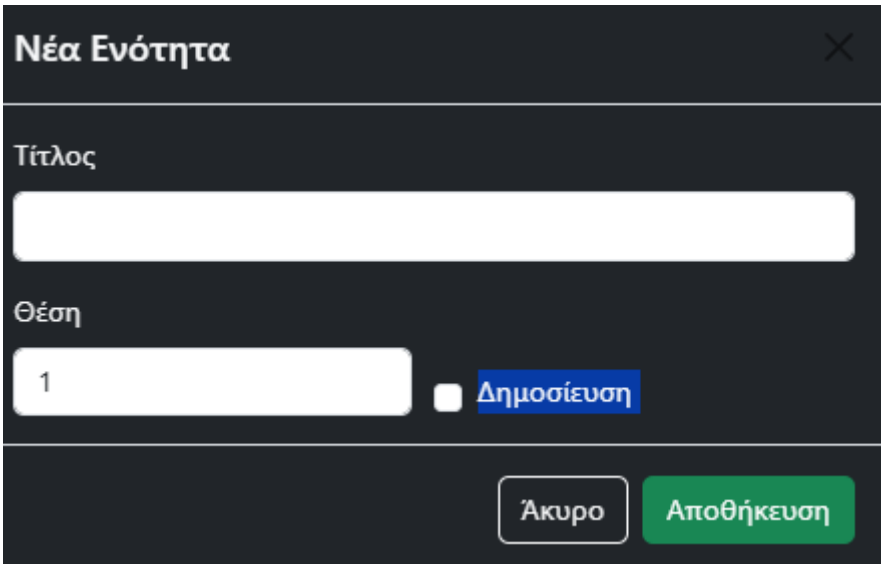
Εικόνα 3.11: Φόρμα – Σελίδα για επεξεργασία ενός Προγράμματος – Επιμορφωτής

Όπως φαίνεται στην Εικόνα 3.11, πρόκειται για τη σελίδα επεξεργασίας υπάρχοντος προγράμματος. Ο επιμορφωτής μπορεί να τροποποιήσει τα βασικά στοιχεία του προγράμματος και να διαχειριστεί τις αντίστοιχες ενότητες και ρυθμίσεις. Το περιβάλλον παραμένει ίδιο με εκείνο της δημιουργίας, με εστίαση στην ευκολία χρήσης. Εδώ μπορεί να ανεβάσει και αρχεία.

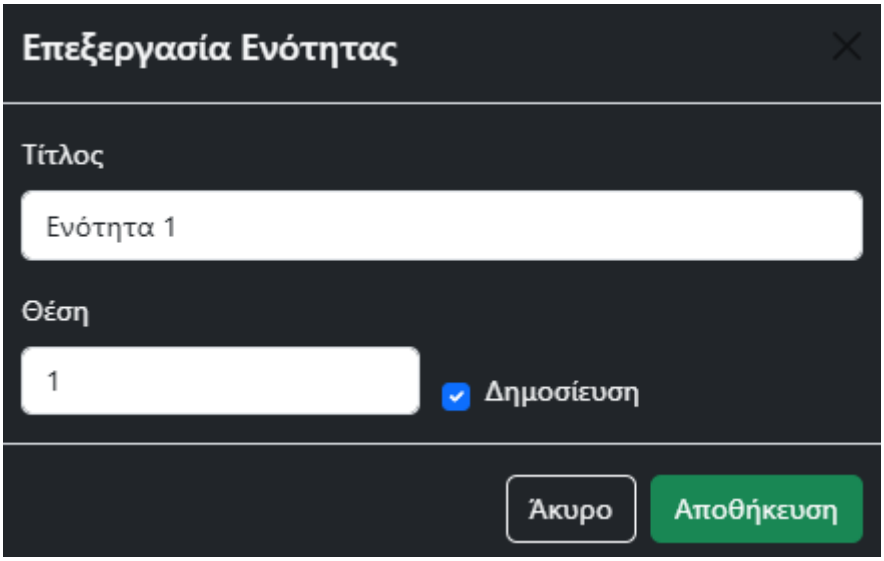
Όπως φαίνεται στην Εικόνα 3.12, εμφανίζεται λίστα με όλες τις ενότητες που έχουν προστεθεί σε ένα πρόγραμμα. Από εδώ ο επιμορφωτής μπορεί να επεξεργαστεί, να αφαιρέσει ή να εμπλουτίσει κάθε ενότητα με επιπλέον υλικό.

Εικόνα 3.12: Σελίδα για την προβολή ενοτήτων ενός προγράμματος – Επιμορφωτής

Η Εικόνα 3.13 απεικονίζει τη διαδικασία προσθήκης νέας ενότητας σε υπάρχον πρόγραμμα. Ο επιμορφωτής ορίζει τον τίτλο και την περιγραφή της ενότητας, ώστε να οργανωθεί καλύτερα το εκπαιδευτικό περιεχόμενο.

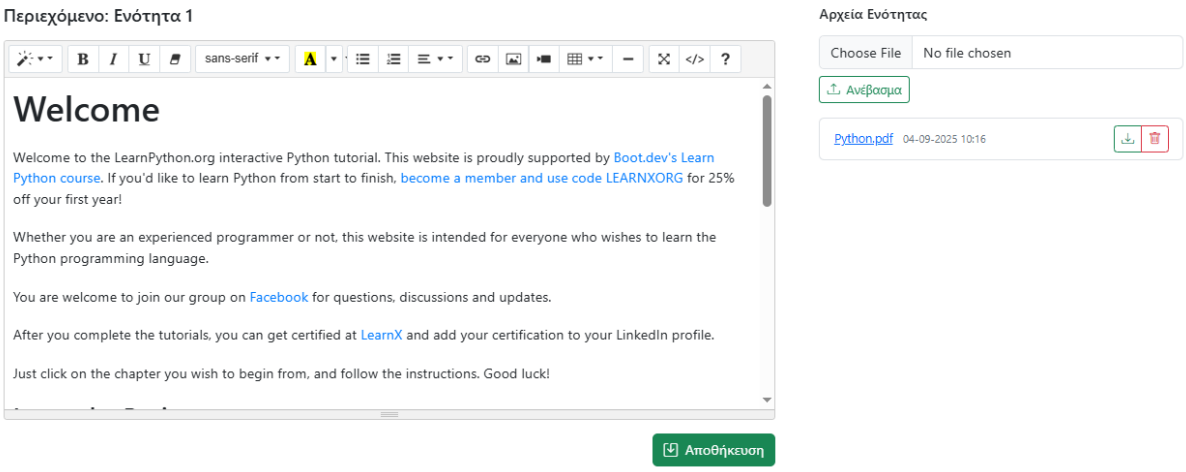


Εικόνα 3.13: Φόρμα – Σελίδα για τη δημιουργία μιας νέας ενότητας – Επιμορφωτής



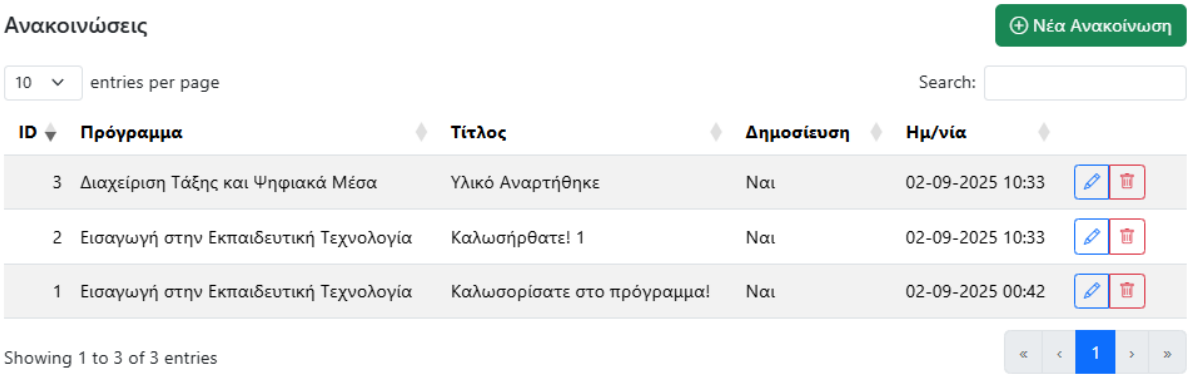
Εικόνα 3.14: Φόρμα – Σελίδα για την επεξεργασία μιας νέας ενότητας – Επιμορφωτής

Στην Εικόνα 3.14 παρουσιάζεται η φόρμα επεξεργασίας μιας ενότητας εντός προγράμματος. Ο επιμορφωτής μπορεί να δώσει τίτλο, περιγραφή και να διαχειριστεί βασικές παραμέτρους της ενότητας πριν την οριστική αποθήκευση.



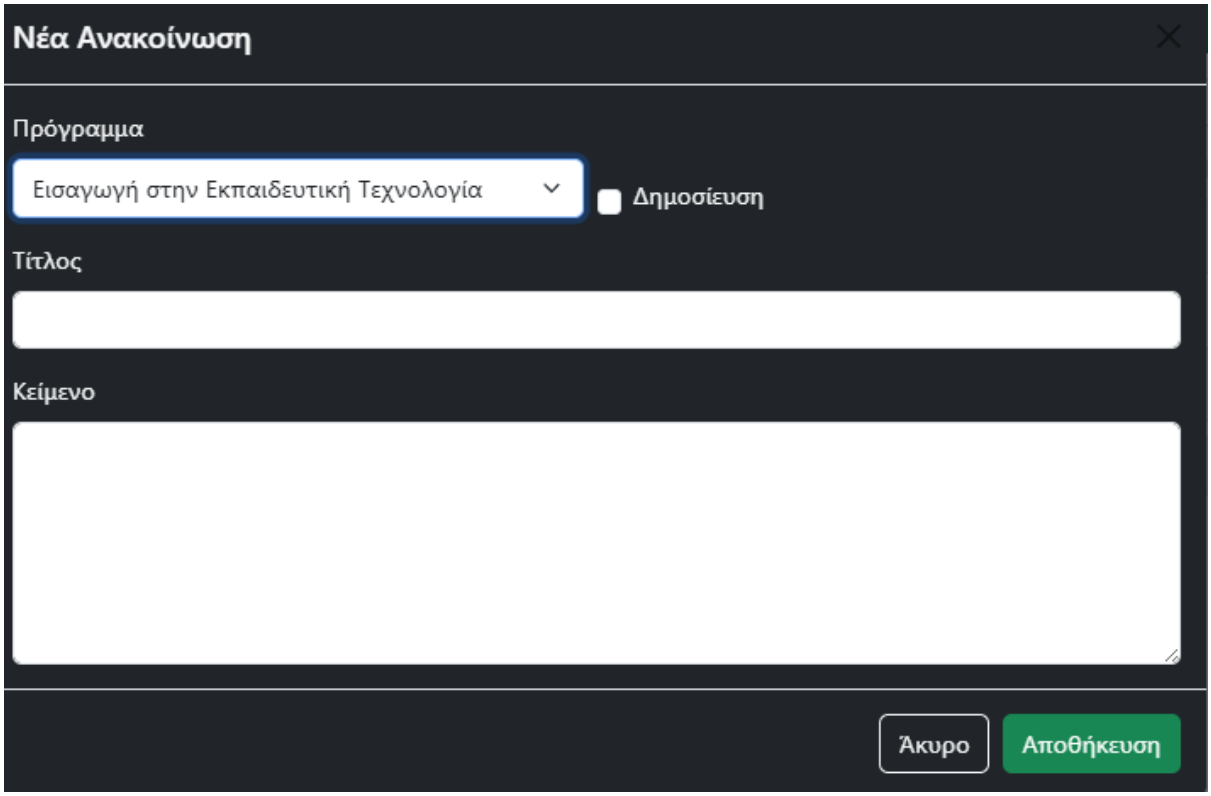
Εικόνα 3.15: Σελίδα για τη δημιουργία εμπλουτισμένου περιεχομένου μιας νέας ενότητας – Επιμορφωτής

Όπως φαίνεται στην Εικόνα 3.15, ο επιμορφωτής έχει τη δυνατότητα να προσθέσει εκπαιδευτικό υλικό σε μία ενότητα. Στη φόρμα περιλαμβάνονται πεδία για τίτλο, περιγραφή και επισύναψη αρχείων προς χρήση από τους επιμορφούμενους.



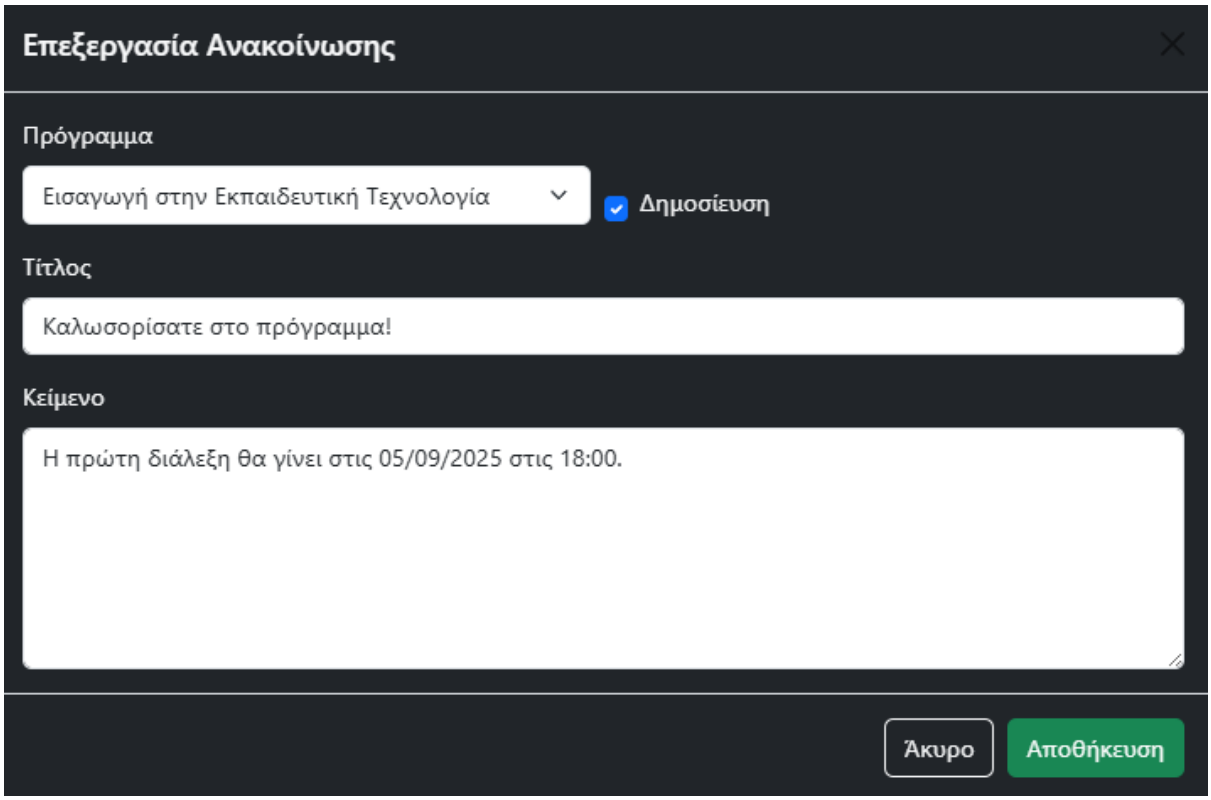
Εικόνα 3.16: Σελίδα για την προβολή των ανακοινώσεων – Επιμορφωτής

Στην εικόνα 3.16 παρουσιάζεται η σελίδα όπου ο επιμορφωτής μπορεί να δει συγκεντρωτικά όλες τις ανακοινώσεις που έχουν αναρτηθεί. Εμφανίζονται πληροφορίες όπως ο τίτλος, η περιγραφή, η ημερομηνία και η συσχέτιση με συγκεκριμένα προγράμματα ή ενότητες.



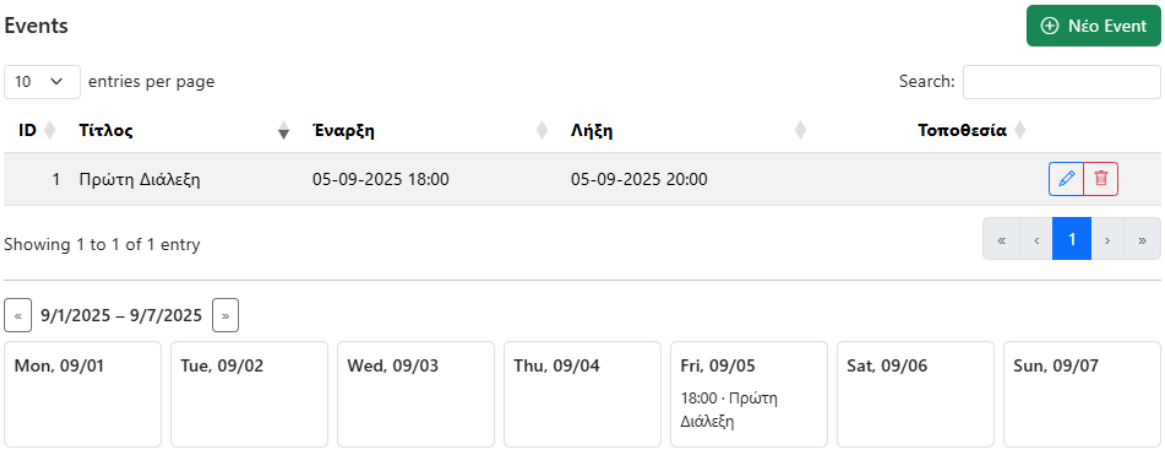
Εικόνα 3.17: Φόρμα – Σελίδα για τη δημιουργία μια νέας ανακοίνωσης για ένα πρόγραμμα – Επιμορφωτής

Η εικόνα 3.17 δείχνει τη φόρμα με την οποία ο επιμορφωτής μπορεί να δημιουργήσει νέα ανακοίνωση. Υπάρχουν πεδία για τίτλο, περιγραφή, προσθήκη αρχείων και επιλογή των προγραμμάτων στα οποία θα εμφανιστεί η ανακοίνωση.



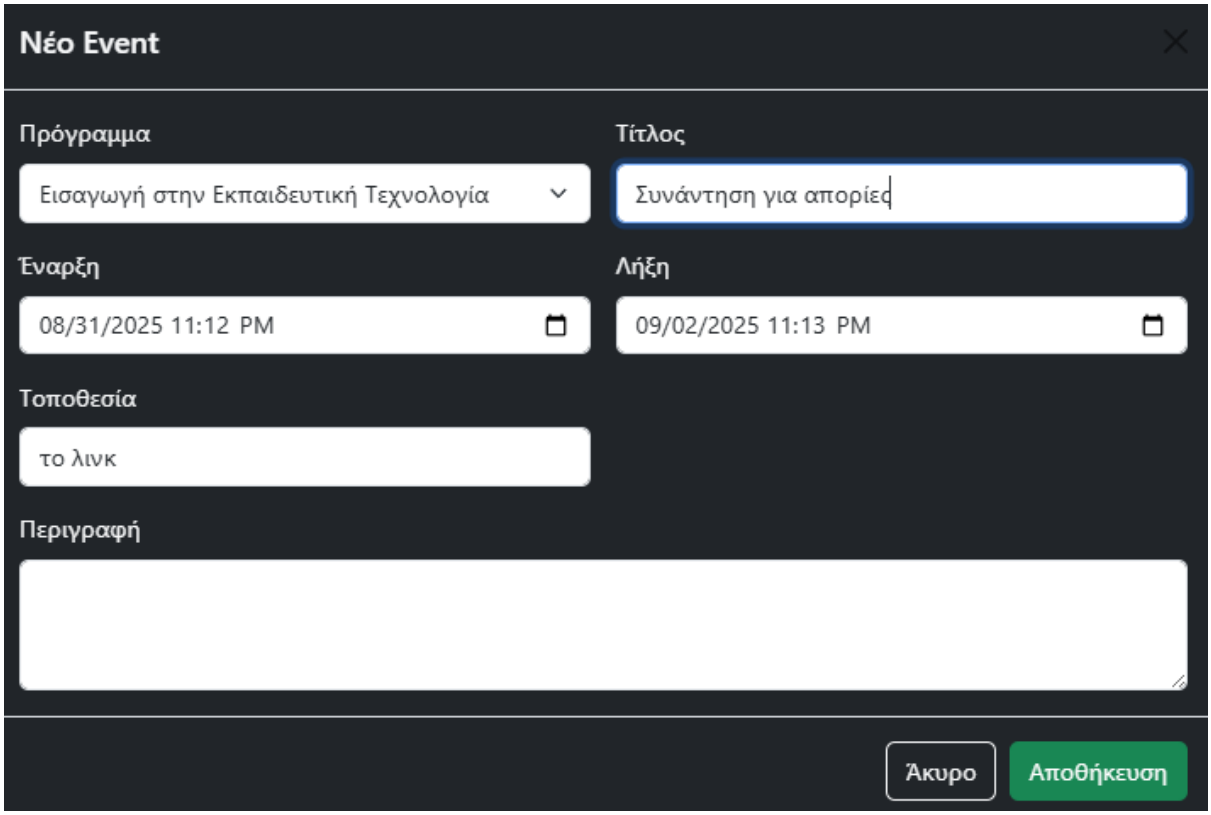
Εικόνα 3.18: Φόρμα – Σελίδα για την επεξεργασία μιας ανακοίνωσης για ένα πρόγραμμα – Επιμορφωτής

Η εικόνα 3.18 παρουσιάζει τη φόρμα με την οποία ο επιμορφωτής μπορεί να επεξεργαστεί μία υπάρχουσα ανακοίνωση. Παρέχεται δυνατότητα αλλαγής του τίτλου, της περιγραφής και των συνημμένων αρχείων, καθώς και διαχείρισης της συσχέτισής της με συγκεκριμένο πρόγραμμα ή ενότητα.



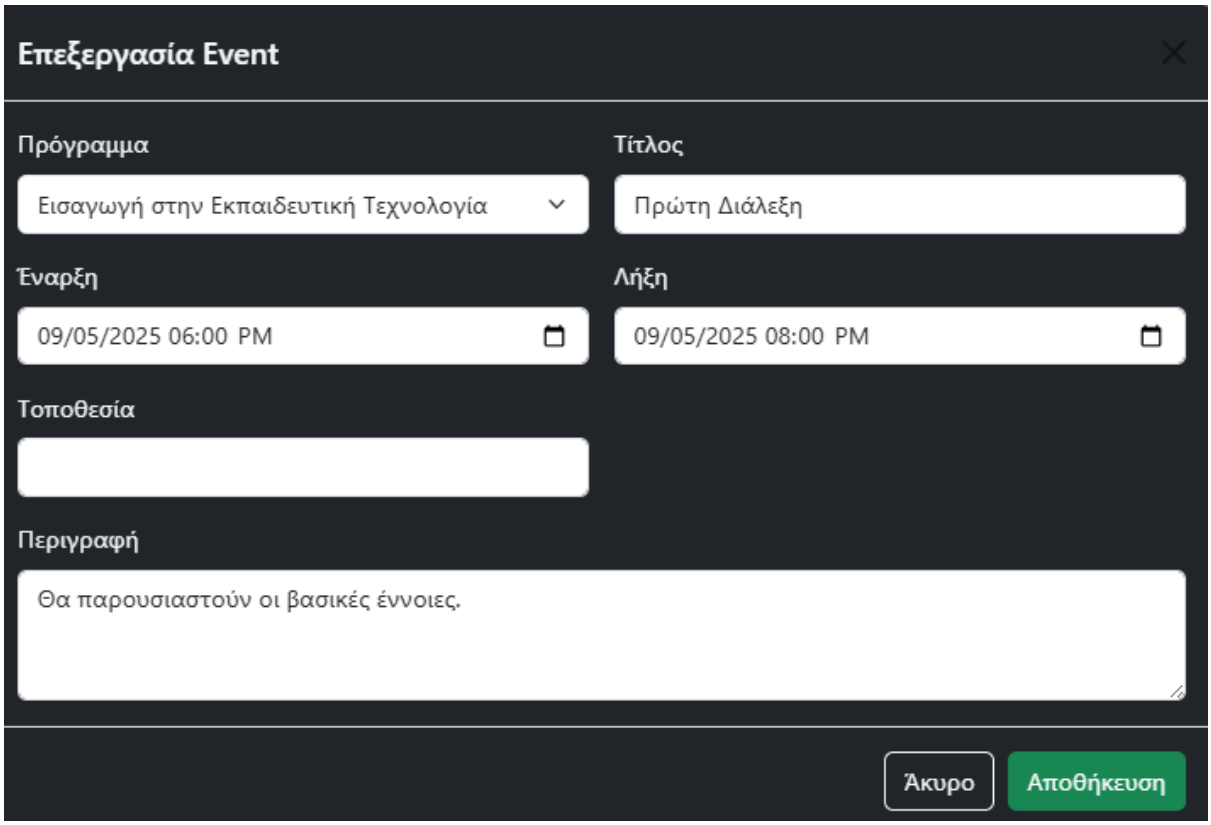
Εικόνα 3.19: Σελίδα με όλα τα Events για τα ανήκοντα επιμορφωτικά πρόγραμμα – Επιμορφωτής

Η εικόνα 3.19 εμφανίζει τη σελίδα όπου συγκεντρώνονται όλα τα γεγονότα (events) που αφορούν τα προγράμματα του επιμορφωτή. Περιλαμβάνονται πληροφορίες για αναρτήσεις, εγγραφές, απαντήσεις αξιολογήσεων και άλλες ενέργειες, οργανωμένες με χρονολογική σειρά για εύκολη παρακολούθηση.



Εικόνα 3.20: Φόρμα – Σελίδα για τη δημιουργία ενός νέου Event για ένα συγκεκριμένο πρόγραμμα με ορισμό ημερομηνιών έναρξης και λήξης – Επιμορφωτής

Στην εικόνα 3.20 φαίνεται η φόρμα μέσω της οποίας ο επιμορφωτής μπορεί να δημιουργήσει ένα νέο event για κάποιο από τα προγράμματά του. Περιλαμβάνει πεδία για τίτλο, περιγραφή και προσδιορισμό των ημερομηνιών έναρξης και λήξης, ώστε να καταγραφεί με ακρίβεια η χρονική διάρκεια του γεγονότος.

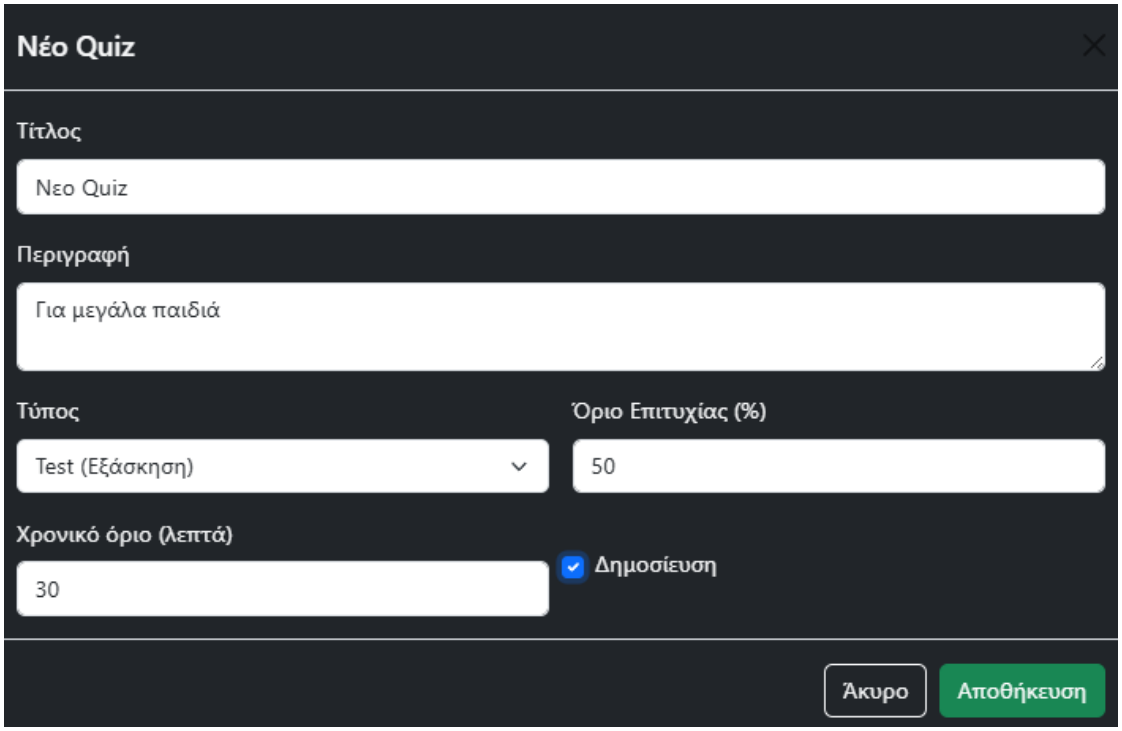


Εικόνα 3.21: Φόρμα – Σελίδα για την επεξεργασία ενός Event για ένα συγκεκριμένο πρόγραμμα με ορισμό ημερομηνιών έναρξης και λήξης – Επιμορφωτής

Η εικόνα 3.21 απεικονίζει τη φόρμα επεξεργασίας ενός ήδη υπάρχοντος event. Ο επιμορφωτής έχει τη δυνατότητα να τροποποιήσει τον τίτλο, την περιγραφή, καθώς και τις ημερομηνίες έναρξης και λήξης του γεγονότος, προσαρμόζοντας το event στις ανάγκες του προγράμματος.

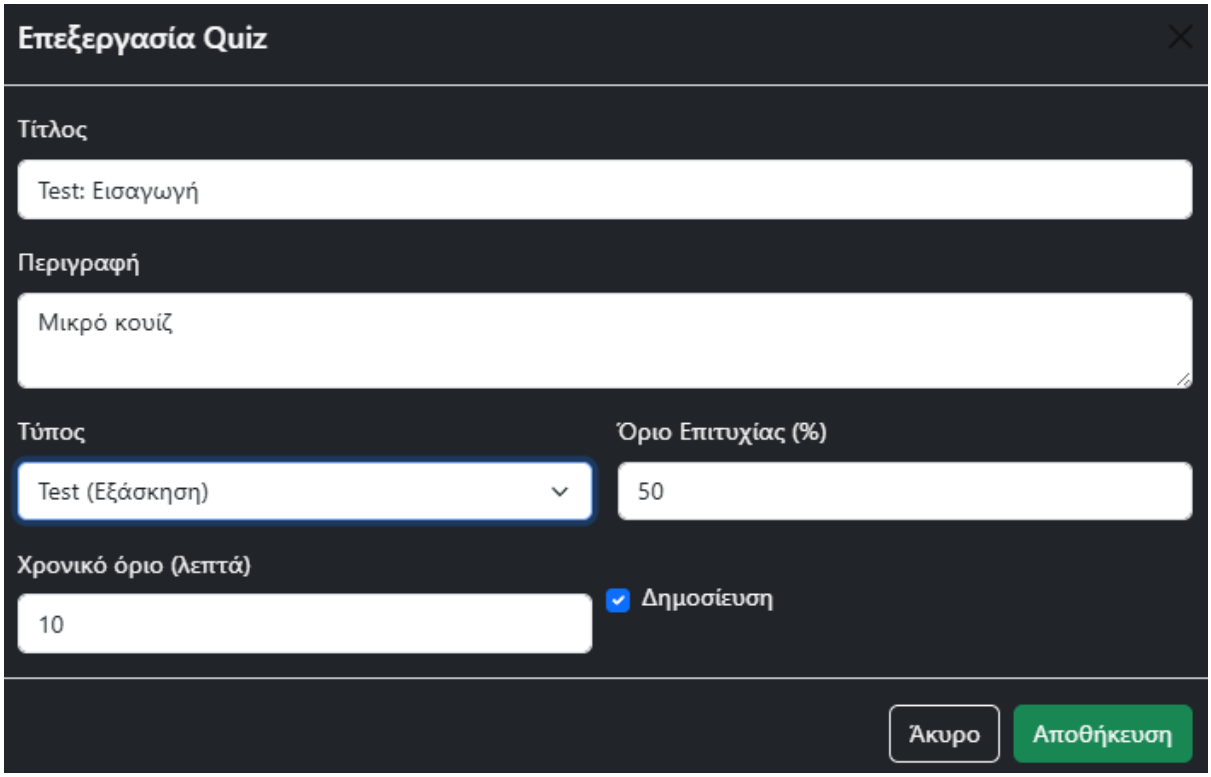
Εικόνα 3.22: Σελίδα για την προβολή των Quizzes που ανήκουν στον επιμορφωτή – Επιμορφωτής

Η εικόνα 3.22 παρουσιάζει τη σελίδα όπου εμφανίζονται όλα τα quizzes που έχει δημιουργήσει ο επιμορφωτής. Για κάθε quiz εμφανίζονται πληροφορίες όπως ο τίτλος, το είδος (Test ή Exam), ο χρόνος διάρκειας και το όριο επιτυχίας, με δυνατότητα διαχείρισης ή επεξεργασίας.



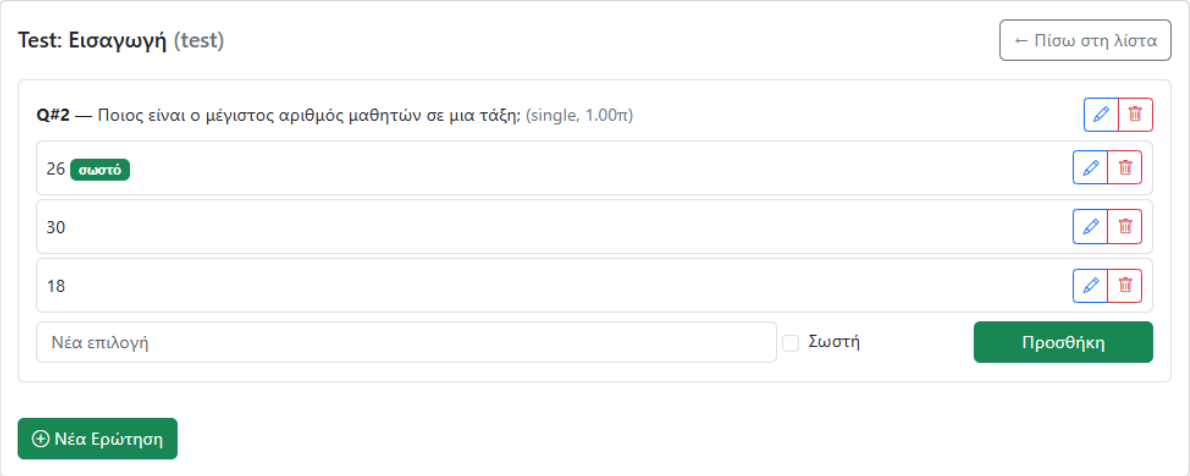
Εικόνα 3.23: Φόρμα – Σελίδα για τη δημιουργία ενός νέου Quiz με επιλογή αν είναι Test ή Exam και όριο επιτυχίας και χρονικό όριο. – Επιμορφωτής

Στην εικόνα 3.23 αποτυπώνεται η φόρμα δημιουργίας ενός νέου quiz. Ο επιμορφωτής μπορεί να ορίσει τίτλο, είδος αξιολόγησης (Test ή Exam), χρονικό όριο και ποσοστό επιτυχίας. Η δομή είναι απλή και καθοδηγεί τον χρήστη ώστε να συμπληρώσει σωστά τα απαραίτητα πεδία.



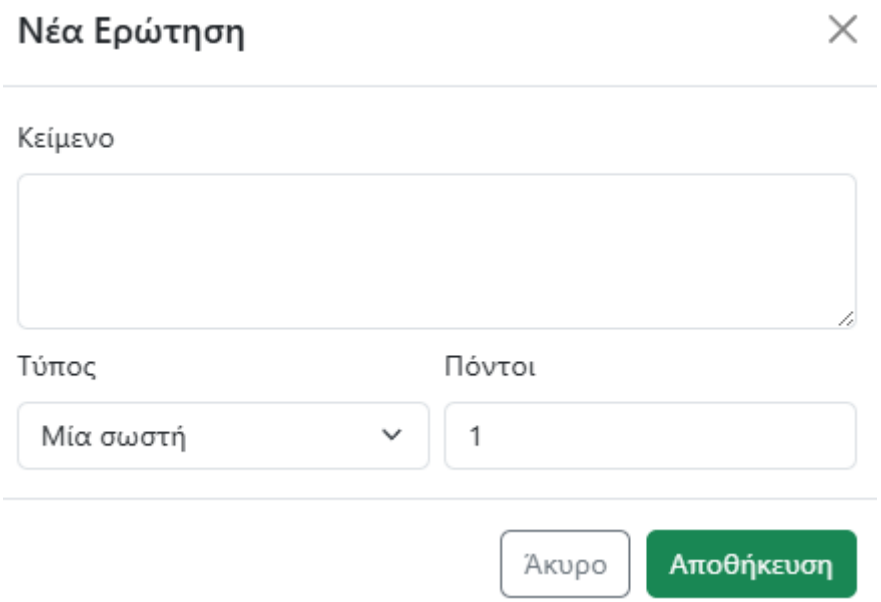
Εικόνα 3.24: Φόρμα – Σελίδα για την επεξεργασία ενός Quiz – Επιμορφωτής

Η εικόνα 3.24 δείχνει τη σελίδα επεξεργασίας ενός ήδη δημιουργημένου quiz. Ο επιμορφωτής έχει τη δυνατότητα να αλλάξει βασικές παραμέτρους, όπως τίτλο, διάρκεια και όριο επιτυχίας, καθώς και να προσθέσει ή να αφαιρέσει ερωτήσεις από το quiz.



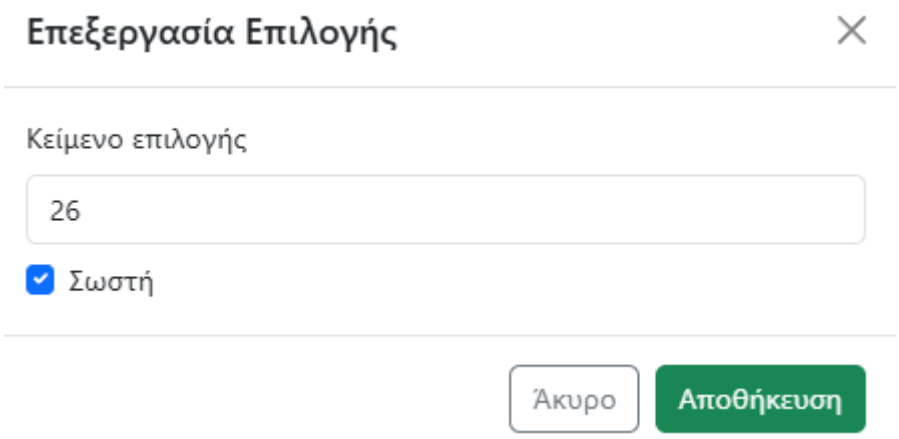
Εικόνα 3.25: Σελίδα για την Εισαγωγή μια νέας ερώτησης ή επεξεργασίας και επεξεργασία των απαντήσεων – Επιμορφωτής

Η εικόνα 3.25 απεικονίζει τη σελίδα όπου ο επιμορφωτής μπορεί να εισάγει νέα ερώτηση ή να επεξεργαστεί υπάρχουσα. Παράλληλα, εμφανίζονται και οι απαντήσεις που σχετίζονται με την ερώτηση, με δυνατότητα τροποποίησης ή διαγραφής.



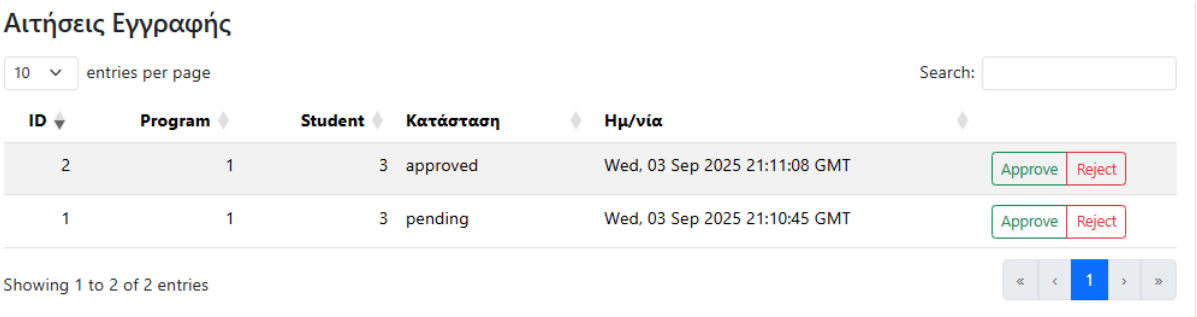
Εικόνα 3.26: Φόρμα για προσθήκη νέας ερώτησης με επιλογή πόντων και τύπου – Επιμορφωτής

Στην εικόνα 3.26 παρουσιάζεται η φόρμα μέσω της οποίας ο επιμορφωτής δημιουργεί μία νέα ερώτηση. Μπορεί να καθορίσει τον τύπο της ερώτησης (π.χ. πολλαπλής επιλογής), το κείμενό της και τη βαθμολογική της αξία (πόντους).



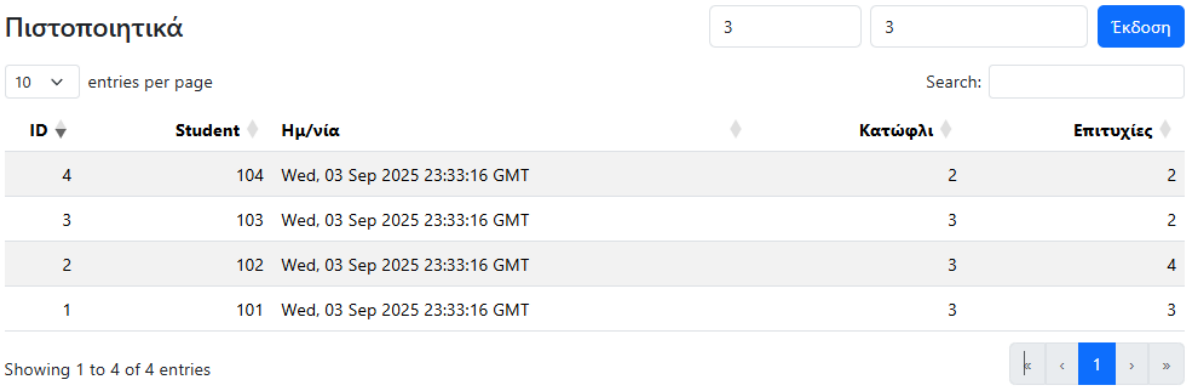
Εικόνα 3.27: Επεξεργασία απάντησης και καθορισμός κειμένου επιλογής και αν είναι η σωστή απάντηση – Επιμορφωτής

Η εικόνα 3.27 δείχνει τη φόρμα επεξεργασίας απάντησης σε ερώτηση quiz. Ο επιμορφωτής ορίζει το κείμενο της απάντησης και επιλέγει αν πρόκειται για τη σωστή επιλογή, ώστε να βαθμολογείται κατάλληλα κατά την αξιολόγηση.



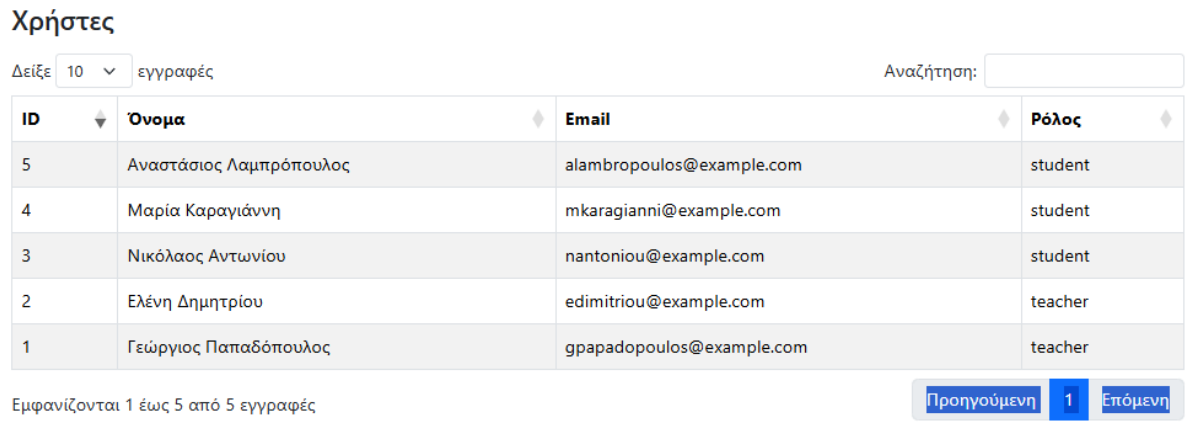
Εικόνα 3.28: Σελίδα Αιτήσεων Εγγραφής για αποδοχή ή απόρριψή – Επιμορφωτής

Η εικόνα 3.28 παρουσιάζει τη σελίδα όπου ο επιμορφωτής μπορεί να δει όλες τις αιτήσεις εγγραφής στα προγράμματά του. Από τη σελίδα αυτή έχει τη δυνατότητα να αποδεχθεί ή να απορρίψει κάθε αίτηση, διαχειριζόμενος έτσι τη συμμετοχή των επιμορφούμενων.



Εικόνα 3.29: Σελίδα για έκδοση Πιστοποιητικών – Επιμορφωτής

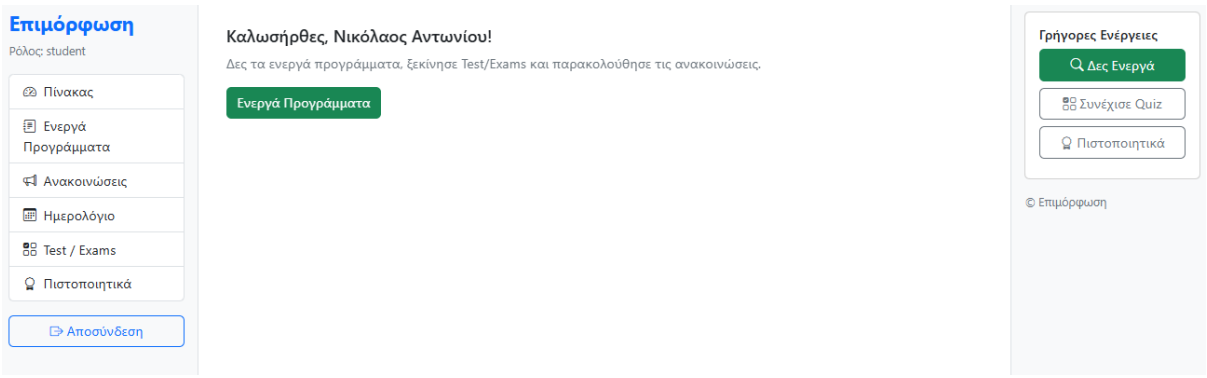
Στην εικόνα 3.29 εμφανίζεται η σελίδα από την οποία ο επιμορφωτής μπορεί να εκδώσει πιστοποιητικά παρακολούθησης. Για κάθε συμμετέχοντα εμφανίζονται στοιχεία που σχετίζονται με την πρόοδο και την επιτυχία του στο πρόγραμμα, πριν την τελική έκδοση.



Εικόνα 3.30: Σελίδα με όλους τους χρήστες (Admin) – Επιμορφωτής

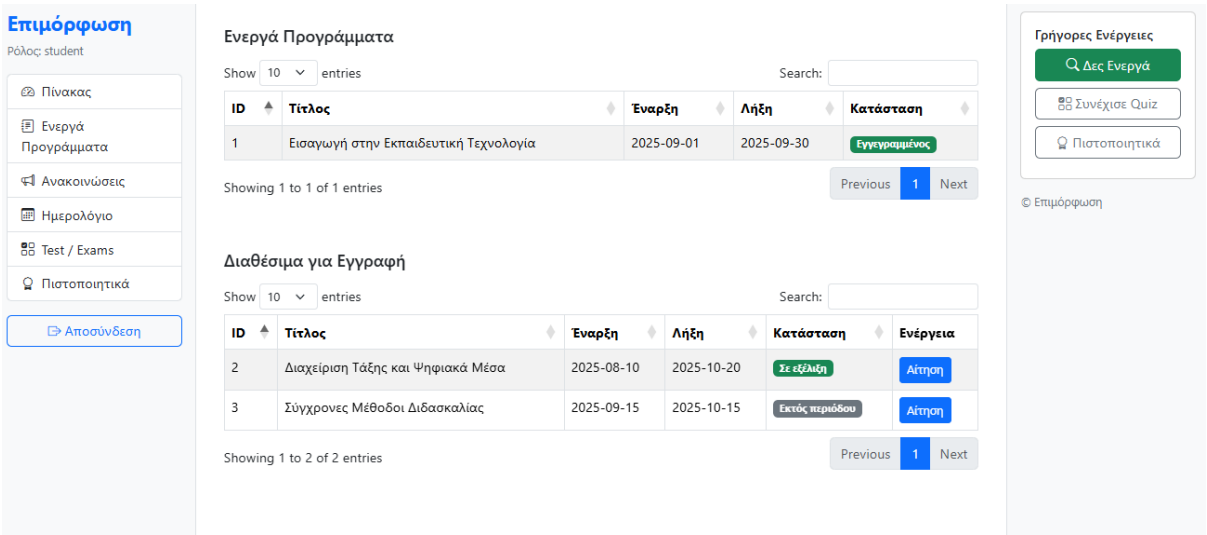
Η εικόνα 3.30 δείχνει τη σελίδα διαχείρισης χρηστών, στην οποία έχει πρόσβαση ο διαχειριστής του συστήματος. Εμφανίζονται όλα τα καταχωρημένα μέλη, με δυνατότητες επεξεργασίας των στοιχείων και εποπτείας της συμμετοχής τους.

3.2.2 Από την πλευρά του Student – Αυτός που ακολουθεί ένα πρόγραμμα επιμόρφωσης



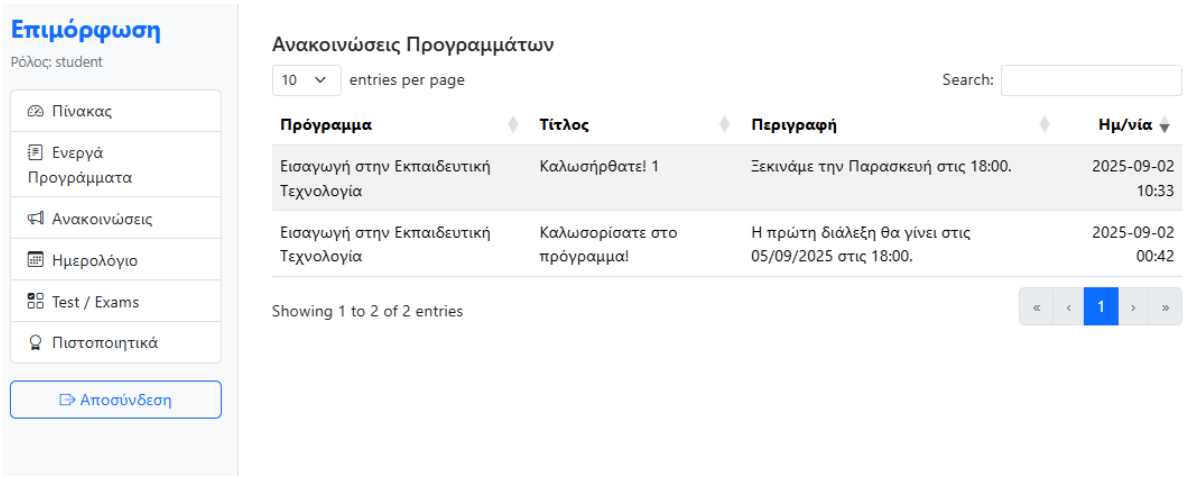
Εικόνα 3.31: Κεντρική σελίδα – dashboard του επιμορφωμένου χρήστη – Χρήστης

Η εικόνα 3.31 απεικονίζει την αρχική σελίδα του επιμορφωμένου χρήστη μετά τη σύνδεσή του. Από το dashboard έχει πρόσβαση στα προγράμματα στα οποία συμμετέχει, καθώς και σε ανακοινώσεις, αξιολογήσεις και εκπαιδευτικό υλικό.



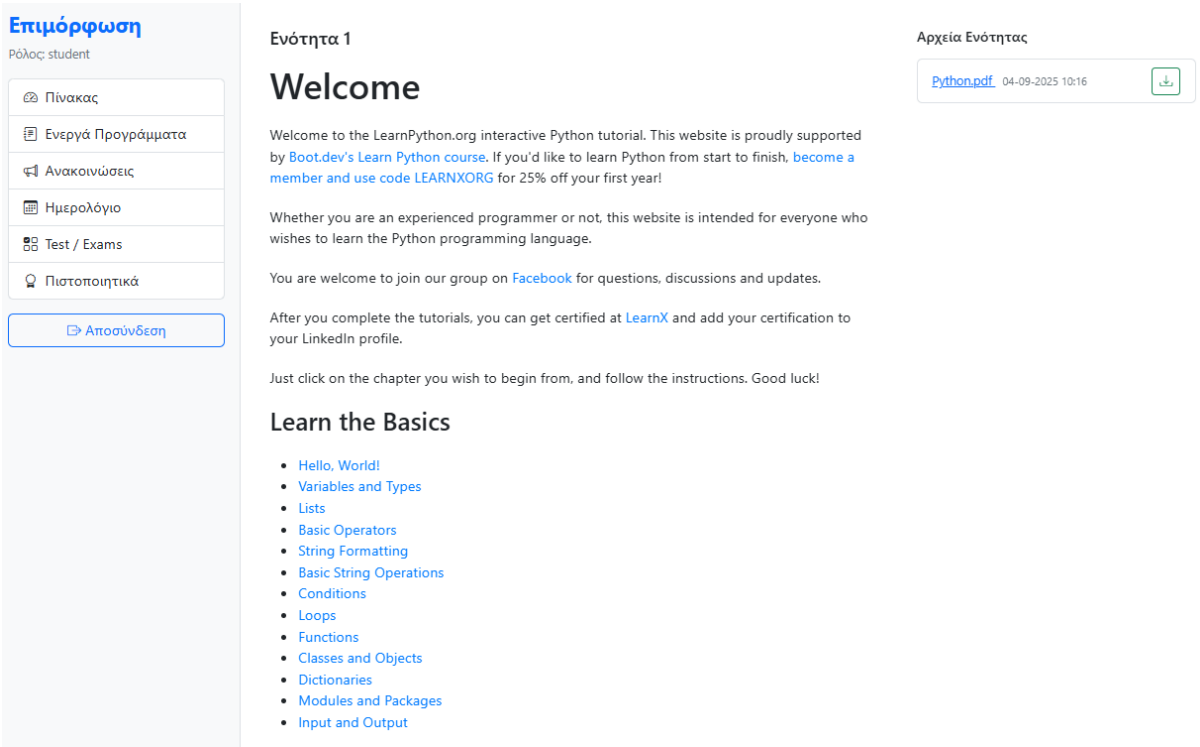
Εικόνα 3.32: Σελίδα με όλα τα επιμορφωτικά προγράμματα – Ενεργά και Διαθέσιμα – Επιλογή αίτησης εγγραφής – Χρήστης

Στην εικόνα 3.32 παρουσιάζεται η σελίδα με τα προγράμματα που είναι διαθέσιμα προς εγγραφή. Ο χρήστης μπορεί να δει πληροφορίες για κάθε πρόγραμμα και να υποβάλει αίτηση συμμετοχής, αν δεν είναι ήδη εγγεγραμμένος.



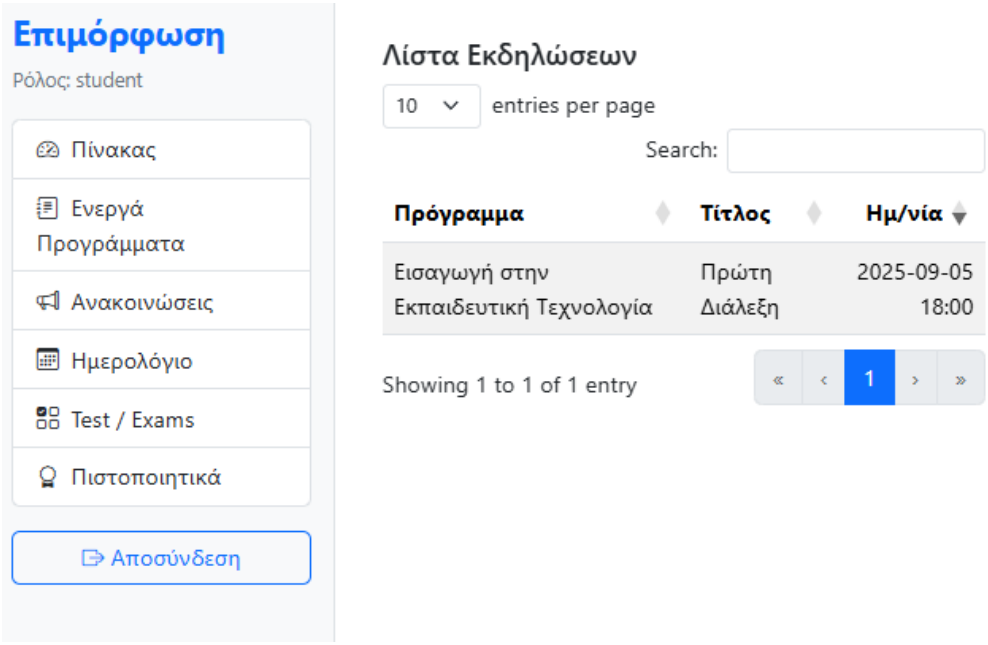
Εικόνα 3.33: Ανακοινώσεις των προγραμμάτων – Χρήστης

Η εικόνα 3.33 δείχνει τις ανακοινώσεις που σχετίζονται με τα προγράμματα στα οποία συμμετέχει ο χρήστης. Οι ανακοινώσεις εμφανίζονται με τίτλο, περιγραφή, συνδεδεμένο πρόγραμμα και δυνατότητα λήψης τυχόν επισυναπτόμενων αρχείων.



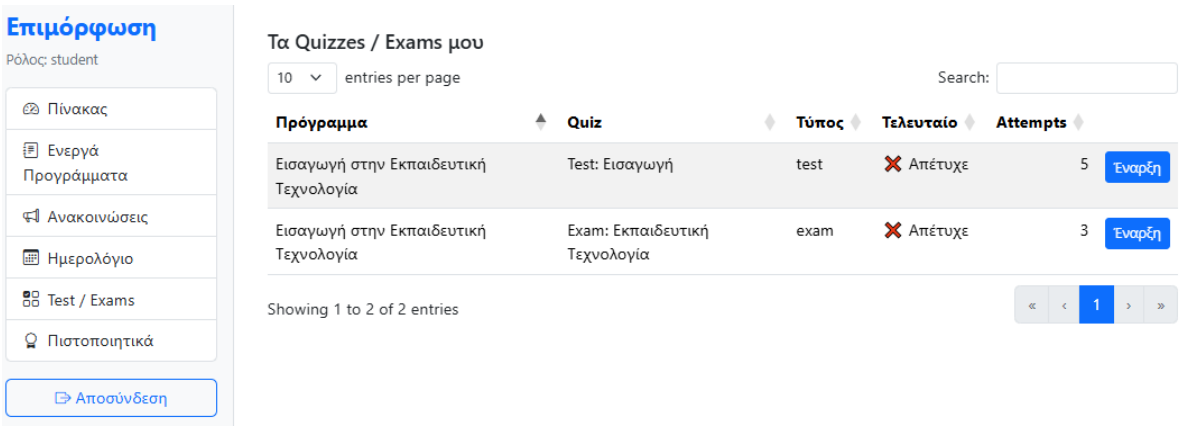
Εικόνα 3.34: Σελίδα με το περιεχόμενο επιμόρφωσης – και τα αρχεία – Χρήστης

Όπως φαίνεται στην εικόνα 3.34, ο χρήστης έχει πρόσβαση στο εκπαιδευτικό περιεχόμενο που έχει αναρτήσει ο επιμορφωτής. Για κάθε ενότητα εμφανίζονται τα σχετικά αρχεία, με δυνατότητα προεπισκόπησης ή λήψης.



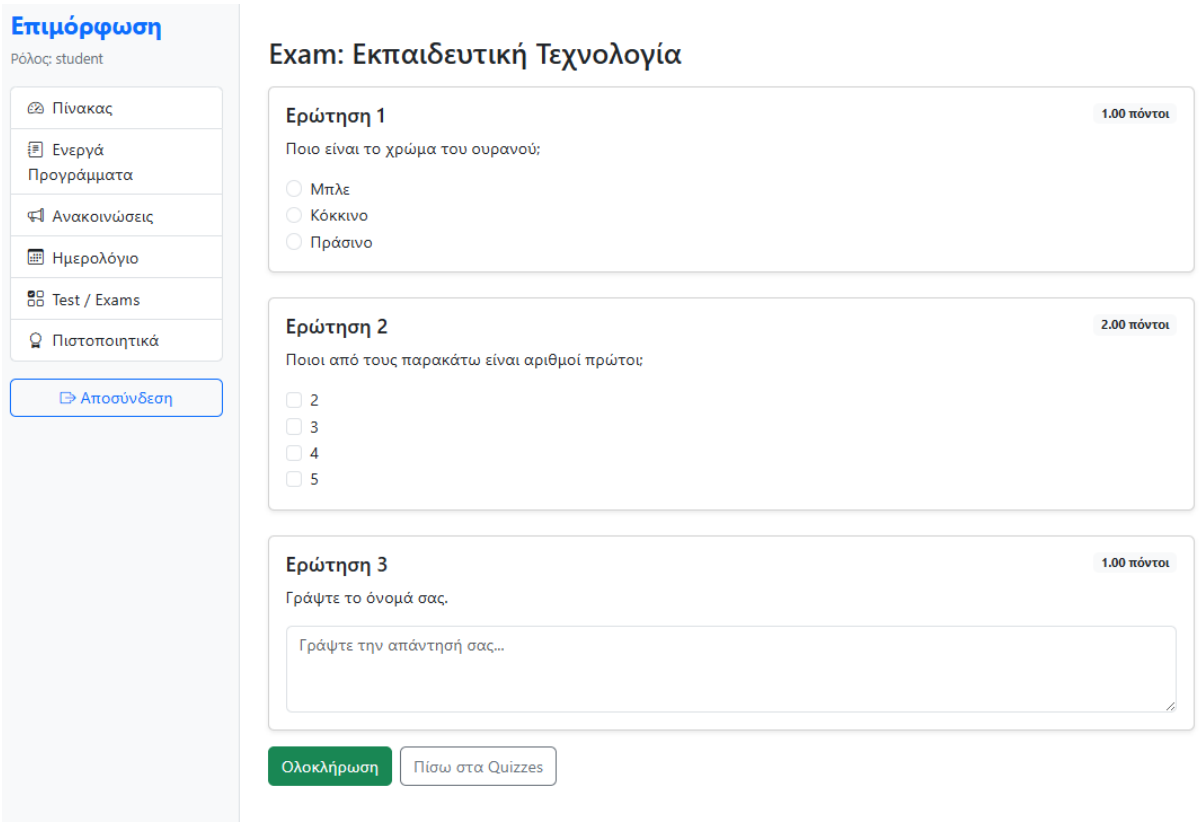
Εικόνα 3.35: Σελίδα με τα Events - εκδηλώσεις – Χρήστης

Η εικόνα 3.35 παρουσιάζει τη σελίδα στην οποία συγκεντρώνονται όλα τα προγραμματισμένα events που αφορούν τον χρήστη. Εμφανίζονται με χρονολογική σειρά και συνοδεύονται από τίτλο, περιγραφή και σχετικές ημερομηνίες.



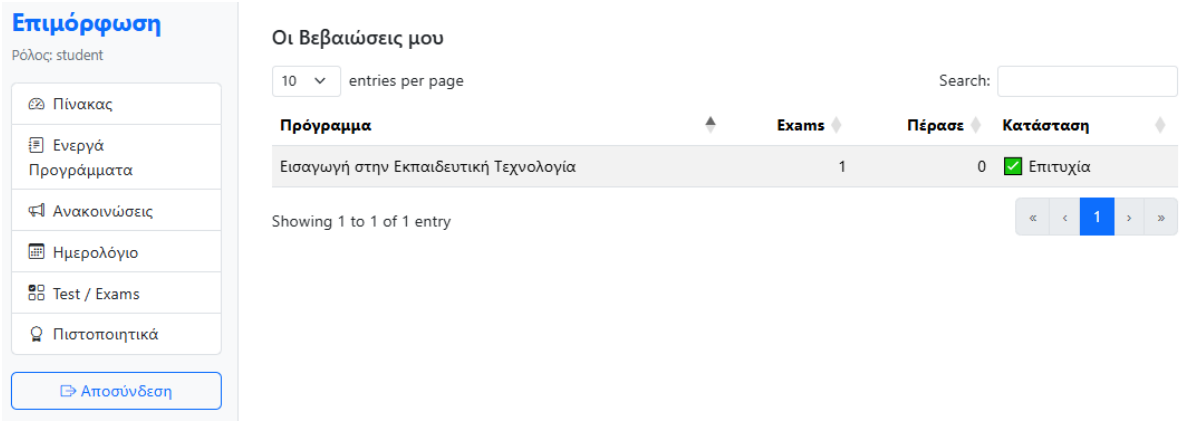
Εικόνα 3.36: Σελίδα με τα διαθέσιμα Quizzes/Exams ανα μάθημα που είναι εγγεγραμμένος ο χρήστης – Χρήστης

Η εικόνα 3.36 εμφανίζει όλα τα διαθέσιμα Quizzes και Exams που σχετίζονται με τα προγράμματα στα οποία είναι εγγεγραμμένος ο χρήστης. Κάθε αξιολόγηση συνοδεύεται από βασικά στοιχεία όπως τίτλος, είδος, χρονικό όριο και κατάσταση.



Εικόνα 3.37: Σελίδα με ένα Exam εκτελεί ο χρήστης – Χρήστης

Στην εικόνα 3.37 παρουσιάζεται η διεπαφή που βλέπει ο χρήστης κατά την εκτέλεση ενός Exam. Εμφανίζεται η ερώτηση, οι διαθέσιμες απαντήσεις και ο διαθέσιμος χρόνος, προσφέροντας μια απλή και λειτουργική εμπειρία αξιολόγησης.



Εικόνα 3.38: Σελίδα με τις βεβαιώσεις που έχει ο χρήστης από κάποιο επιμορφωτικό πρόγραμμα – Χρήστης

Η εικόνα 3.38 δείχνει τις βεβαιώσεις που έχει αποκτήσει ο χρήστης από την επιτυχία του σε επιμορφωτικά προγράμματα. Οι βεβαιώσεις συνοδεύονται από βασικές πληροφορίες, όπως ημερομηνία έκδοσης και όριο επιτυχίας.

3.3 Περιγραφή της πλατφόρμας από πλευρά προγραμματισμού

Παρακάτω φαίνεται η Δομή και Περιγραφή των Κύριων Αρχείων της Πλατφόρμας:

- **app.py**: Κεντρικό αρχείο εκκίνησης της εφαρμογής Flask και εγγραφής των Blueprints.
- **auth.py**: Περιλαμβάνει τη λογική αυθεντικοποίησης χρηστών (login/logout).
- **teacher.py**: Περιέχει όλα τα routes και λειτουργίες για τον ρόλο του επιμορφωτή.
- **student.py**: Περιέχει όλα τα routes και λειτουργίες για τον ρόλο του επιμορφούμενου.
- **db.py**: Διαχειρίζεται τη σύνδεση με τη βάση δεδομένων και βοηθητικές συναρτήσεις για SQL queries.
- **/templates/**: Περιέχει όλα τα HTML αρχεία της διεπαφής, οργανωμένα ανά ρόλο και χρήση.
- **/static/**: Περιλαμβάνει αρχεία CSS, JavaScript και εικόνες που χρησιμοποιούνται από το frontend.

Η εφαρμογή υλοποιείται με το Flask, ένα ελαφρύ framework για web εφαρμογές σε Python. Στο αρχείο app.py ορίζεται η βασική συνάρτηση create_app(), η οποία δημιουργεί το αντικείμενο της εφαρμογής, θέτει το μυστικό κλειδί για τα sessions και εγγράφει τρία βασικά Blueprints — ένα για την αυθεντικοποίηση (auth_bp), ένα για τον ρόλο του επιμορφωτή (teacher_bp) και ένα για τον

επιμορφούμενο (student_bp). Η δομή αυτή εξασφαλίζει καθαρό και διαχωρισμένο κώδικα ανάλογα με τον ρόλο χρήστη.

Εκκίνηση της εφαρμογής Flask – Αρχείο app.py

```
from flask import Flask
from auth import auth_bp
from teacher import teacher_bp
from student import student_bp

def create_app():
    app = Flask(__name__)
    app.secret_key = 'dev-key'  # μυστικό κλειδί για τα session

    # Καταχώρηση των Blueprint για τους τρεις ρόλους
    app.register_blueprint(auth_bp)
    app.register_blueprint(teacher_bp, url_prefix='/teacher')
    app.register_blueprint(student_bp, url_prefix='/student')

    return app
```

Το **db.py**, το οποίο είναι υπεύθυνο για τη σύνδεση της πλατφόρμας με τη βάση δεδομένων και την εκτέλεση SQL εντολών με ασφάλεια και ευκολία.

```
from flask import g
import mysql.connector

# Δημιουργία σύνδεσης με τη βάση δεδομένων
def get_db():
    if 'db' not in g:
        g.db = mysql.connector.connect(
            host='localhost',
            user='root',
            password='',
            database='epimorfosi2',
            charset='utf8'
        )
    return g.db

# Εκτέλεση εντολής χωρίς επιστροφή (INSERT, UPDATE, DELETE)
def execute(sql, params=None):
    db = get_db()
    cursor = db.cursor()
    cursor.execute(sql, params or ())
```

```
    db.commit()
    cursor.close()

# Εκτέλεση SELECT και επιστροφή όλων των γραμμών
def fetch_all(sql, params=None):
    db = get_db()
    cursor = db.cursor(dictionary=True)
    cursor.execute(sql, params or ())
    result = cursor.fetchall()
    cursor.close()
    return result

# Εκτέλεση SELECT και επιστροφή μίας μόνο γραμμής
def fetch_one(sql, params=None):
    db = get_db()
    cursor = db.cursor(dictionary=True)
    cursor.execute(sql, params or ())
    result = cursor.fetchone()
    cursor.close()
    return result
```

Στο αρχείο db.py υλοποιούνται όλες οι βασικές λειτουργίες επικοινωνίας με τη βάση δεδομένων MySQL. Η συνάρτηση get_db() δημιουργεί ή επιστρέφει μια υπάρχουσα σύνδεση ανά request, αποθηκευμένη στο αντικείμενο g της Flask. Οι execute(), fetch_all() και fetch_one() είναι βοηθητικές συναρτήσεις που απλοποιούν τη χρήση SQL εντολών. Οι συναρτήσεις επιστρέφουν αποτελέσματα σε μορφή dictionary, ώστε να είναι εύκολα προσπελάσιμα από τον υπόλοιπο κώδικα.

Αυθεντικοποίηση χρηστών – Αρχείο `auth.py`

```
from flask import Blueprint, render_template, request, redirect, session, url_for
from db import fetch_one

auth_bp = Blueprint('auth', __name__)

# Σελίδα σύνδεσης
@auth_bp.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
```

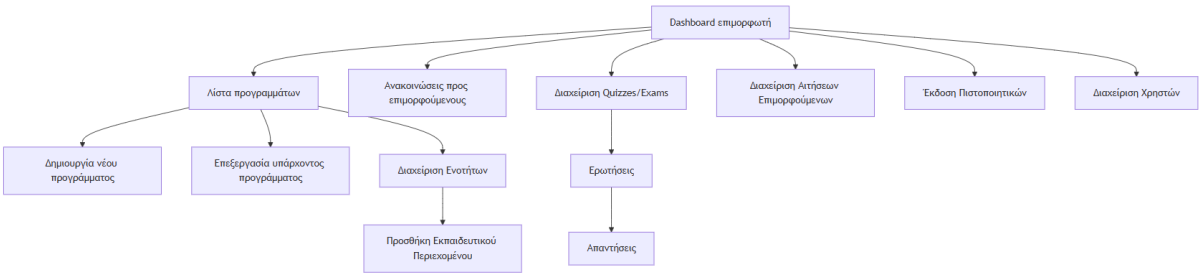
```
    # Έλεγχος στοιχείων χρήστη στη βάση
    user = fetch_one("SELECT * FROM users WHERE email = %s AND password = %s", (email, password))
    if user:
        session['user_id'] = user['id']
        session['role'] = user['role']

        # Ανακατεύθυνση ανάλογα με τον ρόλο
        if user['role'] == 'teacher':
            return redirect('/teacher')
        elif user['role'] == 'student':
            return redirect('/student')
    else:
        return render_template('auth/login.html', error='Λανθασμένα στοιχεία')
  return render_template('auth/login.html')

# Αποσύνδεση χρήστη
@auth_bp.route('/logout')
def logout():
  session.clear()
  return redirect(url_for('auth.login'))
```

Το αρχείο auth.py αποτελεί το σημείο ελέγχου για την πρόσβαση στην πλατφόρμα. Η βασική route / χειρίζεται τη σύνδεση των χρηστών, ελέγχοντας τα διαπιστευτήριά τους μέσω ερωτήματος SQL. Αν τα στοιχεία είναι σωστά, αποθηκεύονται στο session το user_id και ο ρόλος του χρήστη (teacher ή student). Στη συνέχεια γίνεται ανακατεύθυνση στον αντίστοιχο πίνακα ελέγχου. Η route /logout διαγράφει τα session δεδομένα και επιστρέφει τον χρήστη στη σελίδα σύνδεσης.

Το **teacher.py**, το οποίο περιλαμβάνει τις λειτουργίες που σχετίζονται με τον ρόλο του **επιμορφωτή**: τη διαχείριση προγραμμάτων, ενοτήτων, περιεχομένου, ανακοινώσεων, αξιολογήσεων κ.ά.



Εικόνα 3.39: Διάγραμμα για teacher.py

```
from flask import Blueprint, render_template, request, session, redirect, abort, jsonify
from db import fetch_all, fetch_one, execute
```

```
teacher_bp = Blueprint('teacher', __name__)

# Έλεγχος αν ο χρήστης είναι επιμορφωτής
def _require_teacher():
    if session.get('role') != 'teacher':
        abort(403)

# Αρχική σελίδα επιμορφωτή (Dashboard)
@teacher_bp.route('/')
def dashboard():
    resp = _require_teacher()
    if resp: return resp
    return render_template('teacher/dashboard.html')
```

Η βασική δομή του teacher.py περιλαμβάνει τον ορισμό ενός Blueprint για τον επιμορφωτή και μια βοηθητική συνάρτηση _require_teacher() που ελέγχει αν ο τρέχων χρήστης έχει ρόλο επιμορφωτή. Αν όχι, επιστρέφεται σφάλμα 403 (απαγορευμένη πρόσβαση). Το dashboard προσφέρεται μέσω της route /teacher/, και είναι η κεντρική σελίδα διαχείρισης του επιμορφωτή.

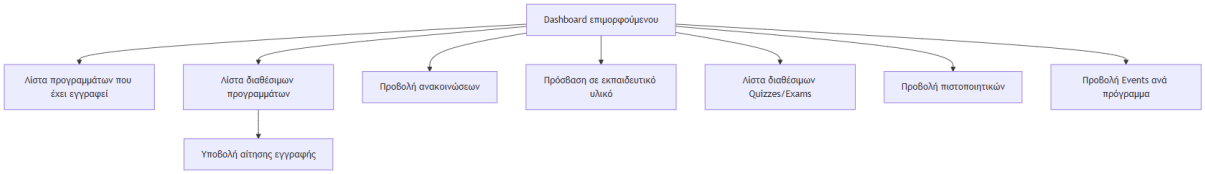
Λίστα με τα προγράμματα του επιμορφωτή (API)

```
@teacher_bp.route('/api/programs')
def api_programs():
    _require_teacher()
    teacher_id = session.get('user_id')
    rows = fetch_all("""
        SELECT * FROM programs
        WHERE teacher_id = %s
        ORDER BY start_date DESC
    """, (teacher_id,))
    return jsonify({'ok': True, 'data': rows})
```

Η παραπάνω route επιστρέφει σε μορφή JSON όλα τα προγράμματα που ανήκουν στον επιμορφωτή που έχει συνδεθεί. Η χρήση αυτής της διαδρομής γίνεται κυρίως μέσω AJAX για εμφάνιση δεδομένων σε δυναμικό πίνακα (DataTable) στο frontend.

Το student.py, το οποίο υλοποιεί τις βασικές λειτουργίες του **επιμορφούμενου χρήστη**, όπως:

- Προβολή και αίτηση συμμετοχής σε προγράμματα
- Πρόσβαση σε περιεχόμενο ενοτήτων
- Αξιολογήσεις (quizzes/exams)
- Ανακοινώσεις, πιστοποιητικά και events



Εικόνα 3.40: Διάγραμμα για student.py

Λειτουργίες επιμορφούμενου – Αρχείο student.py (βασική δομή)

```
from flask import Blueprint, render_template, session, redirect, abort, request, jsonify
from db import fetch_all, fetch_one, execute

student_bp = Blueprint('student', __name__)

# Έλεγχος αν ο χρήστης είναι επιμορφούμενος
def _require_student():
    if session.get('role') != 'student':
        abort(403)

# Αρχική σελίδα επιμορφούμενου (Dashboard)
@student_bp.route('/')
def dashboard():
    _require_student()
    return render_template('student/dashboard.html')
```

Όπως και στο teacher.py, έτσι και εδώ ορίζεται Blueprint για τις διαδρομές του επιμορφούμενου. Η βοηθητική _require_student() προστατεύει τις διαδρομές από μη εξουσιοδοτημένη πρόσβαση. Η αρχική σελίδα /student/ οδηγεί στον πίνακα ελέγχου του χρήστη.

Λίστα με τα προγράμματα στα οποία είναι εγγεγραμμένος

```
@student_bp.route('/api/programs-enrolled')
def api_programs_enrolled():
    _require_student()
    sid = session.get('user_id')
    rows = fetch_all("""
        SELECT p.id, p.title, p.start_date, p.end_date
        FROM enrollments e
        JOIN programs p ON p.id = e.program_id
        WHERE e.student_id = %s
    """, (sid,))
    return jsonify({'ok': True, 'data': rows})
```

Η παραπάνω διαδρομή επιστρέφει όλα τα προγράμματα στα οποία έχει εγγραφεί ο επιμορφούμενος. Χρησιμοποιείται στο frontend για τη δημιουργία πίνακα ενεργών προγραμμάτων και προβολή σχετικού περιεχομένου.

3.4 Περιγραφή των πινάκων

Η εφαρμογή χρησιμοποιεί μια σχεσιακή βάση δεδομένων MySQL με την ονομασία epimorfosi2. Η βάση περιλαμβάνει πίνακες που καλύπτουν τις βασικές οντότητες της πλατφόρμας: χρήστες, προγράμματα, εγγραφές, περιεχόμενο, αξιολογήσεις και πιστοποιητικά. Παρακάτω περιγράφονται οι πιο σημαντικοί πίνακες που χρησιμοποιούνται στην υλοποίηση:

users

Ο πίνακας users αποθηκεύει όλα τα μέλη της πλατφόρμας, είτε είναι επιμορφωτές, είτε επιμορφούμενοι, είτε διαχειριστές.

Πεδίο	**Περιγραφή**
id	Μοναδικό αναγνωριστικό χρήστη
role	Ο ρόλος του χρήστη (student, teacher, admin)
name	Ονοματεπώνυμο
email	Ηλεκτρονική διεύθυνση (μοναδική)
password_hash	Κρυπτογραφημένος κωδικός πρόσβασης
status	Κατάσταση λογαριασμού (active, inactive, pending)

programs

Ο πίνακας programs περιέχει τα εκπαιδευτικά προγράμματα που δημιουργούν οι επιμορφωτές.

Πεδίο	**Περιγραφή**
id	Μοναδικό αναγνωριστικό προγράμματος
teacher_id	ID του επιμορφωτή
title	Τίτλος προγράμματος
description	Περιγραφή
start_date / end_date	Ημερομηνίες διεξαγωγής
is_published	Κατάσταση δημοσίευσης

Πεδίο	Περιγραφή
required_pass_exams	Αριθμός απαιτούμενων επιτυχών αξιολογήσεων

registrations

Ο πίνακας registrations καταγράφει τις αιτήσεις εγγραφής των επιμορφούμενων σε προγράμματα.

Πεδίο	Περιγραφή
id	Μοναδικό αναγνωριστικό
program_id	ID προγράμματος
student_id	ID επιμορφούμενου
status	Κατάσταση αίτησης (pending, approved, rejected)
decision_by	ID επιμορφωτή που έλαβε την απόφαση
decision_at	Ημερομηνία απόφασης

enrollments

Ο πίνακας enrollments καταγράφει την οριστική εγγραφή επιμορφούμενων σε προγράμματα (μετά την αποδοχή).

Πεδίο	Περιγραφή
id	Μοναδικό αναγνωριστικό
program_id	ID προγράμματος
student_id	ID επιμορφούμενου
is_active	Ενεργή εγγραφή
enrolled_at	Ημερομηνία εγγραφής

announcements

Περιέχει τις ανακοινώσεις που δημιουργούν οι επιμορφωτές για τα προγράμματά τους.

Πεδίο	Περιγραφή
id	ID ανακοίνωσης
program_id	Πρόγραμμα στο οποίο ανήκει
title	Τίτλος
body	Κείμενο ανακοίνωσης

Πεδίο	Περιγραφή
attachments_json	Συννημένα αρχεία (σε μορφή JSON)
is_published	Δημοσιευμένη ή όχι
created_by	ID δημιουργού

materials και media

Δύο πίνακες για το περιεχόμενο επιμόρφωσης:

- materials → αρχεία (PDF, PPTX, DOCX)
- media → πολυμέσα (YouTube, video, links)

quizzes, quiz_questions, quiz_options, quiz_attempts, quiz_attempt_answers

Η ομάδα αυτών των πινάκων διαχειρίζεται το σύστημα αξιολογήσεων της πλατφόρμας:

Πίνακας	Ρόλος
quizzes	Πληροφορίες για κάθε αξιολόγηση (τίτλος, τύπος, όριο επιτυχίας)
quiz_questions	Ερωτήσεις ανά quiz
quiz_options	Επιλογές ανά ερώτηση
quiz_attempts	Προσπάθειες των χρηστών σε quiz
quiz_attempt_answers	Απαντήσεις ανά ερώτηση και προσπάθεια

certificates

Αποθηκεύει τα πιστοποιητικά που έχουν εκδοθεί για επιμορφούμενους.

Πεδίο	Περιγραφή
id	Μοναδικό αναγνωριστικό
student_id	ID επιμορφούμενου
issued_at	Ημερομηνία έκδοσης
threshold	Απαραίτητα επιτυχή quizzes
passed_exams	Πλήθος περασμένων εξετάσεων
meta	Λεπτομέρειες αποθηκευμένες σε JSON

Κεφάλαιο 4ο: Συμπεράσματα και βελτιώσεις

Η πλατφόρμα που υλοποιήθηκε στο πλαίσιο της παρούσας πτυχιακής εργασίας έχει ως σκοπό να προσφέρει μια πλήρη και λειτουργική λύση για την επιμόρφωση χρηστών σε ψηφιακό περιβάλλον. Μέσα από μια εύχρηστη διεπαφή τόσο οι επιμορφωτές όσο και οι επιμορφούμενοι μπορούν να αλληλεπιδράσουν με το σύστημα αξιοποιώντας μια σειρά από λειτουργίες που αφορούν τη δημιουργία, την παρακολούθηση και την αξιολόγηση επιμορφωτικών προγραμμάτων.

Το έργο μας βασίστηκε στην ανάγκη οργάνωσης της εκπαιδευτικής διαδικασίας σε ένα ενιαίο πληροφοριακό σύστημα. Ο επιμορφωτής έχει τη δυνατότητα να δημιουργεί και να διαχειρίζεται προγράμματα, να ανεβάζει υλικό, να δημοσιεύει ανακοινώσεις, να ορίζει αξιολογήσεις και να παρακολουθεί την πορεία των επιμορφούμενων. Από την άλλη πλευρά, οι επιμορφούμενοι έχουν πρόσβαση στα προγράμματα όπου έχουν εγγραφεί, βλέπουν τις ανακοινώσεις, μελετούν το υλικό και συμμετέχουν σε quizzes ή εξετάσεις. Και όλα αυτά μπορούν να οδηγήσουν στην έκδοση ενός πιστοποιητικού παρακολούθησης.

Το σύστημα που δημιουργήθηκε καλύπτει με επιτυχία τις βασικές ανάγκες μιας πλατφόρμας επιμόρφωσης με σχετικά απλό τεχνολογικό υπόβαθρο: Flask (Python) για backend, MySQL για αποθήκευση δεδομένων και Bootstrap στο frontend.

Είναι εφικτό να δημιουργηθεί μια ολοκληρωμένη εκπαιδευτική πλατφόρμα χωρίς την ανάγκη για περίπλοκα εργαλεία ή πλαίσια. Η χρήση του Flask και της MySQL προσέφερε αρκετή ευελιξία και έλεγχο, χωρίς περιττές πολυπλοκότητες. Αυτό δείχνει ότι με καλή οργάνωση, ακόμα και ένα μικρό project μπορεί να έχει ουσιαστικό αποτέλεσμα.

Να τονίζουμε ότι η δομή των ρόλων και η διαχείριση δικαιωμάτων πρόσβασης παίζουν σημαντικό ρόλο στη σωστή λειτουργία του συστήματος. Ο διαχωρισμός ανάμεσα σε επιμορφωτή, επιμορφούμενο και διαχειριστή επιτρέπει την καθαρή οριοθέτηση των δυνατοτήτων κάθε χρήστη και διατηρεί την ασφάλεια και τη σταθερότητα της πλατφόρμας.

Φτιάξαμε μια απλή και ξεκάθαρη διεπαφή χρήστη διευκολύνει την πρόσβαση και τη χρήση από άτομα χωρίς ιδιαίτερες γνώσεις. Η πλατφόρμα έχει σχεδιαστεί ώστε να είναι φιλική και κατανοητή, μειώνοντας έτσι την ανάγκη για εξωτερική υποστήριξη ή οδηγίες.

Παρότι το σύστημα καλύπτει τις βασικές απαιτήσεις, υπάρχουν αρκετά σημεία που μπορούν να εξελιχθούν στο μέλλον.

Η προσθήκη ειδοποιήσεων μέσω email ή στο σύστημα (π.χ. ειδοποίηση για νέα ανακοίνωση, έναρξη quiz κτλ.) θα ενίσχυε την αλληλεπίδραση και την ενημέρωση των χρηστών. Η δυνατότητα υποβολής αρχείων από τους επιμορφούμενους (π.χ. εργασίες, συμπληρωματικό υλικό) θα προσέφερε μεγαλύτερη

αμφίδρομη επικοινωνία και εξατομίκευση της εμπειρίας. Η ενσωμάτωση στατιστικών προόδου (μέσοι όροι, ποσοστά επιτυχίας, συμμετοχή) θα έδινε χρήσιμες πληροφορίες τόσο στον επιμορφωτή όσο και στον διαχειριστή του προγράμματος. Να βελτιώσουμε τα quizzes με τυχαία σειρά ερωτήσεων, αρνητική βαθμολογία ή χρονικά όρια ανά ερώτηση, θα επέτρεπε πιο ευέλικτες μορφές αξιολόγησης. Επίσης η βελτιστοποίηση της διεπαφής για κινητά θα καθιστούσε την πλατφόρμα περισσότερο προσβάσιμη, ανεξαρτήτως συσκευής.

Ισως και μια ενότητα για ανατροφοδότηση των επιμορφούμενων προς τον επιμορφωτή θα μπορούσε να ενισχύσει τη διαφάνεια και να προσφέρει ποιοτική ανατροφοδότηση για τη βελτίωση των προγραμμάτων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://docs.moodle.org/en/About_Moodle

[2] https://blogs.e-me.edu.gr/tsormpat/2019/09/29/%cf%84%ce%b9-%ce%b5%ce%af%ce%bd%ce%b1%ce%b9-%cf%84%ce%bf-moodle/

[3] https://www.instructure.com/canvas

[4] https://www.instructure.com/lms-learning-management-system

[5] https://openedx.org

[6] https://en.wikipedia.org/wiki/Open_edX

[7] https://www.learnpython.org/

[8] https://www.w3schools.com/python

[9] https://www.python.org/about/gettingstarted/

[10] https://www.codecademy.com/learn/learn-python-3

[11] https://flask.palletsprojects.com/en/stable/tutorial/

[12] https://www.geeksforgeeks.org/python/flask-tutorial/

[13] https://www.tutorialspoint.com/flask/index.htm

[14] https://flask.palletsprojects.com/en/stable/

[15] https://www.w3schools.com/MySQL/default.asp

[16] https://www.mysql.com/training/

ΚΩΔΙΚΑΣ

Το κεντρικό αρχείο app.py

```
# ================================
# app.py
# ================================
from flask import Flask, redirect, url_for, session, jsonify, request
from datetime import timedelta
import os

from auth import auth_bp
from teacher import teacher_bp
from student import student_bp

def create_app():
    app = Flask(__name__, static_url_path='/static', static_folder='static', template_folder='templates')
    app.secret_key = os.environ.get('SECRET_KEY', 'dev-secret-change-me')

    # MySQL config via env vars (adjust as needed)
    app.config['MYSQL_HOST'] = os.environ.get('MYSQL_HOST', '127.0.0.1')
    app.config['MYSQL_PORT'] = int(os.environ.get('MYSQL_PORT', '3306'))
    app.config['MYSQL_USER'] = os.environ.get('MYSQL_USER', 'root')
    app.config['MYSQL_PASSWORD'] = os.environ.get('MYSQL_PASSWORD', '')
    app.config['MYSQL_DB'] = os.environ.get('MYSQL_DB', 'epimorfosi2')

    app.permanent_session_lifetime = timedelta(hours=8)

    # Blueprints
    app.register_blueprint(auth_bp)
    app.register_blueprint(teacher_bp, url_prefix='/teacher')
    app.register_blueprint(student_bp, url_prefix='/student')

    @app.route('/')
    def index():
        # If logged in, route to role dashboard; else welcome
        if 'user_id' in session:
            role = session.get('role', 'student')
```

```
            return redirect(url_for(f'{role}.dashboard'))
        return redirect(url_for('auth.welcome'))

    # ---------- Unified JSON error handler for API routes ----------
    @app.errorhandler(401)
    @app.errorhandler(403)
    @app.errorhandler(404)
    @app.errorhandler(500)
    def handle_errors(e):
        path = (request.path or '')
        if path.startswith('/teacher/api/') or path.startswith('/student/api/'):
            code = getattr(e, 'code', 500)
            return jsonify({"ok": False, "error": str(e)}), code
        # For non-API routes keep default HTML responses
        return e

    return app

if __name__ == '__main__':
    app = create_app()
    # Debug μόνο σε ανάπτυξη
    app.run(host='0.0.0.0', port=int(os.environ.get('PORT', 5000)), debug=True)
```

Teacher.py – οι πρώτες γραμμές

```
import traceback
from flask import Blueprint, render_template, session, jsonify, abort, request
from db import fetch_all, fetch_one, execute
import os
from werkzeug.utils import secure_filename

teacher_bp = Blueprint('teacher', __name__, template_folder='templates')

UPLOAD_FOLDER = os.path.join(os.getcwd(), 'static', 'uploads', 'programs')
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

ALLOWED_EXTENSIONS = {'pdf','doc','docx','ppt','pptx','xls','xlsx','png','jpg','jpeg','zip'}
```

```
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS


# ---------------- API για αρχεία προγραμμάτων ----------------

@teacher_bp.route('/api/programs/<int:pid>/files')
def api_program_files(pid):
    resp = _require_teacher()
    if resp: return resp
    rows = fetch_all("""
        SELECT id, filename, uploaded_at
        FROM program_files
        WHERE program_id=%s
        ORDER BY uploaded_at DESC
    """, (pid,))
    return jsonify({"ok": True, "files": rows})

@teacher_bp.route('/api/programs/<int:pid>/files/upload', methods=['POST'])
def api_program_file_upload(pid):
    resp = _require_teacher()
    if resp: return resp
    if 'file' not in request.files:
        return jsonify({"ok": False, "error": "Δεν βρέθηκε αρχείο"}), 400
    file = request.files['file']
    if file.filename == '':
        return jsonify({"ok": False, "error": "Κενό όνομα αρχείου"}), 400
    if not allowed_file(file.filename):
        return jsonify({"ok": False, "error": "Μη αποδεκτός τύπος αρχείου"}), 400


    filename = secure_filename(file.filename)


    # Αν υπάρχει ήδη αρχείο με το ίδιο όνομα → βάλε suffix
    savepath = os.path.join(UPLOAD_FOLDER, filename)
    base, ext = os.path.splitext(filename)
    counter = 1
    while os.path.exists(savepath):
        filename = f"{base}_{counter}{ext}"
```

```
        savepath = os.path.join(UPLOAD_FOLDER, filename)
        counter += 1

    file.save(savepath)

    execute("""
        INSERT INTO program_files (program_id, filename, filepath, uploaded_by)
        VALUES (%s,%s,%s,%s)
    """, (pid, filename, savepath, session.get('user_id')))

    return jsonify({"ok": True, "filename": filename})

@teacher_bp.route('/api/programs/files/<int:fileid>/delete', methods=['POST'])
def api_program_file_delete(fileid):
    resp = _require_teacher()
    if resp: return resp
    row = fetch_one("SELECT filepath FROM program_files WHERE id=%s", (fileid,))
    if not row:
        return jsonify({"ok": False, "error": "Δεν βρέθηκε"}), 404

    # Διαγραφή αρχείου από δίσκο
    try:
        if os.path.exists(row['filepath']):
            os.remove(row['filepath'])
    except Exception as ex:
        print("Delete file error:", ex)

    execute("DELETE FROM program_files WHERE id=%s", (fileid,))
    return jsonify({"ok": True})

# ---------- Helpers ----------
def _require_teacher():
    """
    Role check. Για API routes δίνουμε JSON 401 αντί για HTML,
    ώστε DataTables/Chart.js να μην παίρνουν HTML και να «σπάνε».
    """
    if session.get('role') != 'teacher':
        if request.path.startswith('/teacher/api/'):
```

```
            return jsonify({"ok": False, "error": "Unauthorized"}), 401
        abort(403)

def _fmt_date(d):
    try:
        return d.strftime('%d-%m-%Y') if d else None
    except Exception:
        return str(d) if d else None

def _fmt_datetime(dt):
    try:
        return dt.strftime('%d-%m-%Y %H:%M') if dt else None
    except Exception:
        return str(dt) if dt else None

# ---------- Pages ----------
@teacher_bp.route('/dashboard')
def dashboard():
    resp = _require_teacher()
    if resp: return resp
    return render_template('dashboard_teacher.html')

@teacher_bp.route('/programs')
def programs():
    resp = _require_teacher()
    if resp: return resp
    return render_template('programs_teacher.html')

@teacher_bp.route('/announcements')
def announcements_page():
    resp = _require_teacher()
    if resp: return resp
    return render_template('announcements_teacher.html')

# ---------- Programs API ----------

@teacher_bp.route('/api/programs')
def api_programs():
```

```
    """
    Ασφαλές για DataTables:
    - LEFT JOIN για enrollments
    - Formatting ημερομηνιών στην Python (όχι MySQL DATE_FORMAT)
    - try/except με traceback σε console
    """
    resp = _require_teacher()
    if resp: return resp
    try:
        uid = session.get('user_id')
        if not uid:
            return jsonify({"ok": False, "error": "Session missing user_id"}), 401

        rows = fetch_all("""
            SELECT p.id, p.title, p.start_date, p.end_date, p.is_published,
                   COUNT(e.id) AS enrollments
            FROM programs p
            LEFT JOIN enrollments e
              ON e.program_id = p.id AND e.is_active=1
            WHERE p.teacher_id=%s
            GROUP BY p.id
            ORDER BY p.start_date DESC
        """, (uid,))

        data = [{
            "id": r["id"],
            "title": r["title"],
            "start_date": _fmt_date(r["start_date"]),
            "end_date": _fmt_date(r["end_date"]),
            "is_published": int(r.get("is_published", 0)),
            "enrollments": int(r.get("enrollments", 0))
        } for r in rows]

        return jsonify({"ok": True, "data": data})
    except Exception as ex:
        print("API /teacher/api/programs EXCEPTION:\n", traceback.format_exc())
        return jsonify({"ok": False, "error": str(ex)}), 500
```

```
@teacher_bp.route('/programs/<int:program_id>')
def program_view(program_id):
    resp = _require_teacher()
    if resp: return resp
    row = fetch_one(
        "SELECT * FROM programs WHERE id=%s AND teacher_id=%s",
        (program_id, session.get('user_id'))
    )
    if not row:
        abort(404)
    return render_template('programs_teacher.html', focus_id=program_id)


@teacher_bp.route('/api/programs/<int:program_id>')
def api_program_get(program_id):
    resp = _require_teacher()
    if resp: return resp
    try:
        row = fetch_one(
            "SELECT id, title, description, start_date, end_date, is_published, required_pass_exams "
            "FROM programs WHERE id=%s AND teacher_id=%s",
            (program_id, session.get('user_id'))
        )
        if not row:
            return jsonify({"ok": False, "error": "Not found"}), 404

        # Μορφοποίηση για <input type="date">
        sd = row.get('start_date')
        ed = row.get('end_date')
        row['start_date'] = sd.strftime('%Y-%m-%d') if sd else None
        row['end_date'] = ed.strftime('%Y-%m-%d') if ed else None

        return jsonify({"ok": True, **row})
    except Exception as ex:
        import traceback
        print("API program get EXCEPTION:\n", traceback.format_exc())
        return jsonify({"ok": False, "error": "Server error"}), 500


@teacher_bp.route('/api/programs/create', methods=['POST'])
```

```
def api_program_create():
    resp = _require_teacher()
    if resp: return resp
    try:
        title = (request.form.get('title') or '').strip()
        description = request.form.get('description') or ''
        start_date = request.form.get('start_date')  # YYYY-MM-DD
        end_date = request.form.get('end_date')      # YYYY-MM-DD
        is_published = 1 if request.form.get('is_published') in ('on', '1', 'true', 'True') else 0

        if not title or not start_date or not end_date:
            return jsonify({"ok": False, "error": "Συμπλήρωσε τίτλο/ημ/νίες."}), 400

        new_id = execute(
            "INSERT INTO programs (teacher_id,title,description,start_date,end_date,is_published,created_at) "
            "VALUES (%s,%s,%s,%s,%s,%s,NOW())",
            (session.get('user_id'), title, description, start_date, end_date, is_published)
        )
        return jsonify({"ok": True, "id": int(new_id)})
    except Exception as ex:
        print("API create program EXCEPTION:\n", traceback.format_exc())
        return jsonify({"ok": False, "error": str(ex)}), 500
```

student.py – οι πρώτες γραμμές

```
from flask import Blueprint, render_template, session, jsonify, abort, request
from db import fetch_all, fetch_one, execute

from datetime import datetime, timedelta,date
import random

student_bp = Blueprint('student', __name__, template_folder='templates')

def _require_student():
    if session.get('role') != 'student':
        abort(403)
```

```
@student_bp.route('/dashboard')
def dashboard():
    _require_student()
    return render_template('dashboard_student.html')

@student_bp.route('/programs')
def programs():
    _require_student()
    return render_template('programs_student.html')

@student_bp.route('/announcements')
def announcements_page():
    _require_student()
    return render_template('announcements_student.html')

@student_bp.route('/events')
def events_page():
    _require_student()
    return render_template('events_student.html')

@student_bp.route('/quizzes', endpoint='quizzes')
def quizzes_page():
    _require_student()
    return render_template('quizzes_student.html')

@student_bp.route('/api/announcements')
def api_announcements():
    resp = _require_student()
    if resp: return resp
    sid = session.get('user_id')
    try:
        rows = fetch_all("""
            SELECT a.id, a.title, a.body, a.created_at, p.title AS program_title
            FROM announcements a
            JOIN programs p ON p.id=a.program_id
            JOIN enrollments e ON e.program_id=p.id AND e.student_id=%s
            ORDER BY a.created_at DESC
        """, (sid,))
```

```
        # format ημερομηνία σε string
        for r in rows:
            if r.get('created_at'):
                r['created_at'] = r['created_at'].strftime('%Y-%m-%d %H:%M')
        return jsonify({"ok": True, "data": rows})
    except Exception as ex:
        import traceback; print("api_announcements EXC:\n", traceback.format_exc())
        return jsonify({"ok": False, "error": "Server error"}), 500


# ---------- Εκδηλώσεις ----------
@student_bp.route('/api/events')
def api_events():
    resp = _require_student()
    if resp: return resp
    sid = session.get('user_id')
    try:
        rows = fetch_all("""
            SELECT ev.id, ev.title, ev.start_at, ev.end_at, p.title AS program_title
            FROM events ev
            JOIN programs p ON p.id=ev.program_id
            JOIN enrollments e ON e.program_id=p.id AND e.student_id=%s
            ORDER BY ev.start_at DESC
        """, (sid,))
        for r in rows:
            if r.get('start_at') and hasattr(r['start_at'], 'strftime'):
                r['start_at'] = r['start_at'].strftime('%Y-%m-%d %H:%M')
            if r.get('end_at') and hasattr(r['end_at'], 'strftime'):
                r['end_at'] = r['end_at'].strftime('%Y-%m-%d %H:%M')
        return jsonify({"ok": True, "data": rows})
    except Exception as ex:
        import traceback; print("api_events EXC:\n", traceback.format_exc())
        return jsonify({"ok": False, "error": str(ex)}), 500


# ---------- Quizzes ----------
# Quizzes list (student-visible)
@student_bp.route('/api/quizzes')
def api_quizzes():
```

```
    resp = _require_student()
    if resp: return resp
    sid = session.get('user_id')

    rows = fetch_all("""
        SELECT q.id, q.title, q.quiz_type, q.description,
               q.pass_threshold_percent, q.open_at, q.close_at,
               q.time_limit_minutes, q.attempts_allowed,
               p.title AS program_title,
               -- τελευταία κατάσταση attempt
               COALESCE((
                 SELECT CASE WHEN qa.passed=1 THEN ' Πέτυχε'
                             ELSE 'Απέτυχε' END
                 FROM quiz_attempts qa
                 WHERE qa.quiz_id=q.id AND qa.student_id=%s
                 ORDER BY qa.attempted_at DESC LIMIT 1
               ), '—') AS last_status,
               -- πόσα attempts έχει κάνει
               (SELECT COUNT(*) FROM quiz_attempts qa
                 WHERE qa.quiz_id=q.id AND qa.student_id=%s) AS attempts_done
        FROM quizzes q
        JOIN programs p ON p.id = q.program_id
        JOIN enrollments e ON e.program_id = p.id AND e.student_id = %s
        WHERE q.is_published = 1
          AND (q.open_at IS NULL OR q.open_at <= NOW())
          AND (q.close_at IS NULL OR q.close_at >= NOW())
        ORDER BY q.id DESC
    """, (sid, sid, sid))

    # business flag για "μπορώ να ξεκινήσω"
    for r in rows:
        allowed = r['attempts_allowed']
        r['can_start'] = (allowed is None) or (r['attempts_done'] < allowed)
    return jsonify({"ok": True, "data": rows})

# Start a quiz attempt
# helper για insert + lastrowid
def execute_return_id(query, params=None):
```

```
    import mysql.connector
    from flask import current_app
    conn = mysql.connector.connect(
        host=current_app.config['MYSQL_HOST'],
        user=current_app.config['MYSQL_USER'],
        password=current_app.config['MYSQL_PASSWORD'],
        database=current_app.config['MYSQL_DB'],
        port=current_app.config['MYSQL_PORT']
    )
    cur = conn.cursor()
    cur.execute(query, params or ())
    conn.commit()
    last_id = cur.lastrowid
    cur.close()
    conn.close()
    return last_id

@student_bp.post('/api/quizzes/<int:quiz_id>/start')
def start_quiz_attempt(quiz_id):
    resp = _require_student()
    if resp:
        return resp
    sid = session.get('user_id')

    # 1) Έλεγξε ότι ο φοιτητής είναι εγγεγραμμένος στο πρόγραμμα του quiz
    q = fetch_one("""
        SELECT q.id, q.program_id, q.title, q.is_published,
               q.open_at, q.close_at, q.time_limit_minutes,
               q.attempts_allowed, q.shuffle_questions, q.shuffle_options
        FROM quizzes q
        JOIN enrollments e ON e.program_id = q.program_id AND e.student_id = %s
        WHERE q.id = %s
    """, (sid, quiz_id))
    if not q:
        return jsonify({"ok": False, "error": "Δεν έχετε πρόσβαση σε αυτό το quiz."}), 403

    # 2) Έλεγχοι διαθεσιμότητας
    now = datetime.now()
```

```
if q['is_published'] != 1:
    return jsonify({"ok": False, "error": "Το quiz δεν είναι διαθέσιμο."}), 400
if q['open_at'] and q['open_at'] > now:
    return jsonify({"ok": False, "error": "Το quiz δεν έχει ανοίξει ακόμη."}), 400
if q['close_at'] and q['close_at'] < now:
    return jsonify({"ok": False, "error": "Το quiz έχει κλείσει."}), 400

# 3) Έλεγχος attempts
done = fetch_one("""
    SELECT COUNT(*) AS c FROM quiz_attempts
    WHERE quiz_id = %s AND student_id = %s
""", (quiz_id, sid))['c']
if q['attempts_allowed'] is not None and done >= q['attempts_allowed']:
    return jsonify({"ok": False, "error": "Έχετε εξαντλήσει τα attempts."}), 400

# 4) Δημιουργία attempt και λήψη id
attempt_id = execute_return_id(
    "INSERT INTO quiz_attempts (quiz_id, student_id) VALUES (%s, %s)",
    (quiz_id, sid)
)

# 5) Φόρτωση ερωτήσεων & επιλογών
questions = fetch_all("""
    SELECT qq.id AS question_id, qq.question_text, qq.question_type, qq.points, qq.position
    FROM quiz_questions qq
    WHERE qq.quiz_id = %s
    ORDER BY qq.position ASC, qq.id ASC
""", (quiz_id,))
for qu in questions:
    opts = fetch_all("""
        SELECT qo.id AS option_id, qo.option_text, qo.position
        FROM quiz_options qo
        WHERE qo.question_id = %s
        ORDER BY qo.position ASC, qo.id ASC
    """, (qu['question_id'],))
    qu['options'] = opts

# 6) Shuffle αν χρειάζεται
```

```
    if q['shuffle_questions']:
        import random
        random.shuffle(questions)
    if q['shuffle_options']:
        import random
        for qu in questions:
            random.shuffle(qu['options'])

    return jsonify({
        "ok": True,
        "attempt_id": attempt_id,
        "quiz": {
            "id": q['id'],
            "title": q['title'],
            "time_limit_minutes": q['time_limit_minutes'],
        },
        "questions": questions
    })

# Πιστοποιητικά / Βεβαιώσεις
@student_bp.route('/certificates')
def certificates():
    _require_student()
    return render_template('certificates_student.html')

@student_bp.route('/api/certificates')
def api_certificates():
    _require_student()
    sid = session.get('user_id')
    rows = fetch_all("""
        SELECT p.id AS program_id, p.title,
               p.required_pass_exams,
               (SELECT COUNT(*) FROM quizzes q WHERE q.program_id=p.id AND q.quiz_type='exam') AS total_exams,
               (SELECT COUNT(DISTINCT qa.quiz_id)
                  FROM quiz_attempts qa
                  JOIN quizzes q ON q.id=qa.quiz_id
                 WHERE q.program_id=p.id AND qa.student_id=%s AND qa.passed=1) AS passed_exams
        FROM programs p
```

```
        JOIN enrollments e ON e.program_id=p.id AND e.student_id=%s
        WHERE e.is_active=1
    """, (sid, sid))
    # υπολογισμός status
    for r in rows:
        req = r['required_pass_exams'] or 0
        if r['passed_exams'] >= req:
            r['status'] = f"Επιτυχία "
            # ({r['passed_exams']}/{r['total_exams']})
            r['success'] = True
        else:
            r['status'] = f"Αποτυχία ({r['passed_exams']}/{r['total_exams']}, απαιτούνταν {req})"
            r['success'] = False
    return jsonify({"ok": True, "data": rows})

@student_bp.route('/api/programs-available')
def api_programs_available():
    _require_student()
    sid = session.get('user_id')

    rows = fetch_all("""
        SELECT p.id, p.title, p.start_date, p.end_date
        FROM programs p
        WHERE p.is_published=1
          AND NOT EXISTS (
            SELECT 1 FROM enrollments e
            WHERE e.program_id = p.id AND e.student_id = %s
          )
        ORDER BY p.start_date DESC
    """, (sid,))

    today = date.today()
    for r in rows:
        sd, ed = r.get('start_date'), r.get('end_date')
        r['start_date'] = sd.strftime('%Y-%m-%d') if sd else None
        r['end_date']   = ed.strftime('%Y-%m-%d') if ed else None
        r['active_now'] = bool(sd and ed and sd <= today <= ed)
        r['status'] = 'Σε εξέλιξη' if r['active_now'] else 'Εκτός περιόδου'
```

```
    return jsonify({"ok": True, "data": rows})


@student_bp.route('/api/programs-enrolled')
def api_programs_enrolled():
    resp = _require_student()
    if resp: return resp
    sid = session.get('user_id')
    try:
        rows = fetch_all("""
            SELECT p.id, p.title, p.start_date, p.end_date,
                   e.is_active, e.enrolled_at
            FROM enrollments e
            JOIN programs p ON p.id = e.program_id
            WHERE e.student_id = %s
            ORDER BY p.start_date DESC
        """, (sid,))
        today = date.today()
        for r in rows:
            sd, ed = r.get('start_date'), r.get('end_date')
            r['start_date'] = sd.strftime('%Y-%m-%d') if sd else None
            r['end_date']   = ed.strftime('%Y-%m-%d') if ed else None
            r['active_now'] = bool(sd and ed and sd <= today <= ed)
            r['status'] = 'Σε εξέλιξη' if r['active_now'] else 'Εκτός περιόδου'
        return jsonify({"ok": True, "data": rows})
    except Exception as ex:
        import traceback
        print(" /api/programs-enrolled ERROR:\n", traceback.format_exc())
        return jsonify({"ok": False, "error": "Σφάλμα στον server"}), 500
```