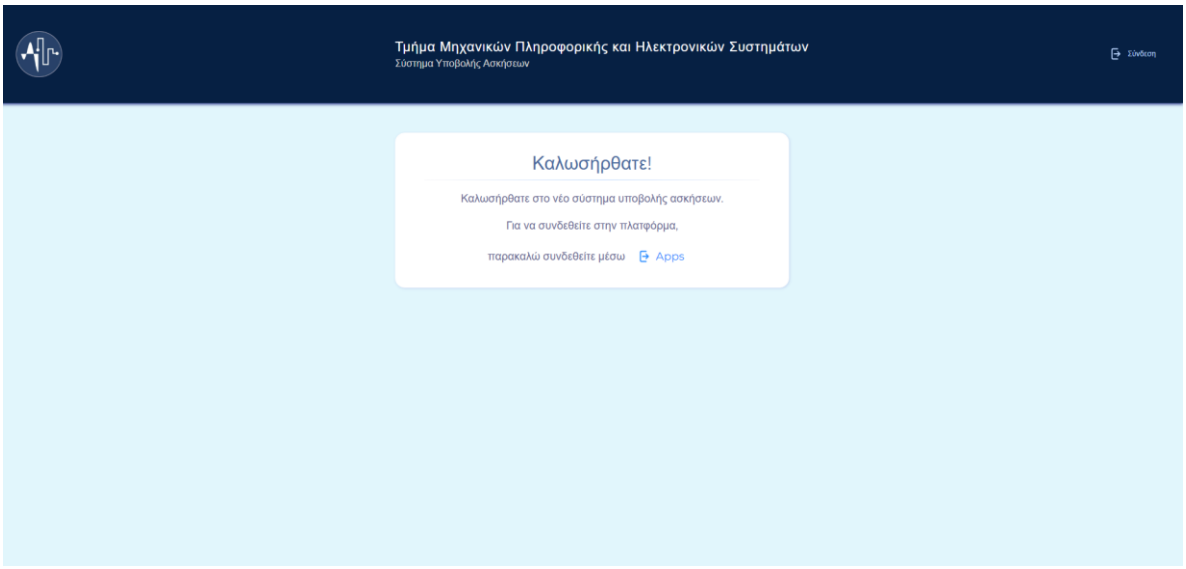
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σύστημα αποστολής ασκήσεων μέσω Web»



Των φοιτητών
Κορδατζάκης Θοδωρής
Αρ. Μητρώου: 185392
Δραμιτινός Βασίλειος
Αρ.Μητρώου: 185430

Επιβλέπων
Σιδηρόπουλος Αντώνης
Αναπληρωτής Καθηγητής

Ημερομηνία 31/5/2025

Τίτλος Δ.Ε. Σύστημα αποστολής ασκήσεων μέσω Web
Κωδικός Π.Ε. **24259**
Ονοματεπώνυμο φοιτητών
Κορδατζάκης Θοδωρής
Δραμιτινός Βασίλειος
Ονοματεπώνυμο εισηγητή Σιδηρόπουλος Αντώνης
Ημερομηνία ανάληψης Π.Ε. 16-10-2024
Ημερομηνία περάτωσης Π.Ε. 30-5-2025

Βεβαιώνουμε ότι είμαστε οι συγγραφείς αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχουμε καταγράψει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνουμε ότι αυτή η εργασία προετοιμάστηκε από εμάς προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής 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.

«Στις οικογένειες μας και στους φίλους που μας στήριξαν»

Πρόλογος

Η ανάγκη για οργάνωση και αποδοτικότητα στη διαδικασία υποβολής ασκήσεων αποτελεί διαχρονικό ζητούμενο στον χώρο της τριτοβάθμιας εκπαίδευσης. Με την ανάπτυξη των τεχνολογιών πληροφορικής και επικοινωνιών, η δυνατότητα δημιουργίας ολοκληρωμένων συστημάτων διαχείρισης υποβολών καθίσταται περισσότερο εφικτή και αναγκαία από ποτέ.

Η παρούσα εργασία επικεντρώνεται στον σχεδιασμό και την ανάπτυξη ενός σύγχρονου συστήματος υποβολής ασκήσεων, το οποίο φιλοδοξεί να εξυπηρετήσει τις ανάγκες τόσο των φοιτητών όσο και των διδασκόντων, προσφέροντας μια οργανωμένη και εύχρηστη πλατφόρμα διαχείρισης της εκπαιδευτικής διαδικασίας. Το σύστημα που αναπτύχθηκε επιτρέπει στους φοιτητές να υποβάλλουν τις ασκήσεις τους με ασφάλεια και συνέπεια, ενώ παρέχει στους διδάσκοντες τα εργαλεία για πλήρη πρόσβαση και έλεγχο επί των υποβολών. Ιδιαίτερη έμφαση δόθηκε στην ευχρηστία της διεπαφής, στην ασφάλεια των δεδομένων και στη δυνατότητα εύκολης προσαρμογής του συστήματος σε διαφορετικά γνωστικά αντικείμενα και εκπαιδευτικά σενάρια.

Η ανάπτυξη βασίστηκε σε σύγχρονες τεχνολογικές πρακτικές, λαμβάνοντας υπόψη τις ανάγκες τόσο των φοιτητών όσο και των διδασκόντων. Η εργασία αυτή δεν περιορίζεται μόνο στην τεχνική ανάπτυξη ενός πληροφοριακού συστήματος, αλλά επιχειρεί επίσης να αναδείξει τη σημασία της τεχνολογικής καινοτομίας στον εκσυγχρονισμό της εκπαίδευσης. Στόχος είναι το παρόν έργο να αποτελέσει μια βάση για περαιτέρω βελτιώσεις και εφαρμογές σε ένα ευρύτερο εκπαιδευτικό πλαίσιο, ανταποκρινόμενο στις ανάγκες ενός διαρκώς εξελισσόμενου ακαδημαϊκού περιβάλλοντος.

Περίληψη

Κάθε εξάμηνο, η αξιολόγηση των φοιτητών μέσω εργασιών, ασκήσεων και εξετάσεων αποτελεί βασικό και αναγκαίο κομμάτι της εκπαιδευτικής διαδικασίας. Για την σωστή διαχείριση και ομαλή ροή της διαδικασίας αυτής, είναι αναγκαία η ύπαρξη ενός συστήματος που να προσφέρει την δυνατότητα ψηφιακής υποβολής των ασκήσεων από τους φοιτητές αλλά και τον έλεγχο και την παρακολούθηση τους από το εκπαιδευτικό προσωπικό. Η ύπαρξη του ήδη υπάρχοντος συστήματος αποτέλεσε ικανοποιητική και αποτελεσματική λύση των βασικών λειτουργιών που απαιτούνταν για ένα τέτοιο σύστημα. Συγχρόνως, οι συνεχώς αυξανόμενες απαιτήσεις της εκπαιδευτικής διαδικασίας, δημιουργούν την ανάγκη μια νέα πιο ανανεωμένη εφαρμογή που θα καλύπτει τόσο τις απαιτήσεις και ανάγκες των φοιτητών όσο και των καθηγητών, εξελίσσοντας τις ήδη υπάρχουσες λειτουργείες και χαρακτηριστικά ενώ παράλληλα γίνεται προσθήκη νέων δυνατοτήτων. Η παρούσα πτυχιακή εργασία έχει ως στόχο την υλοποίηση ενός τέτοιου συστήματος, προσιτό προς τον χρήστη, με την χρήση νέων τεχνολογιών και εργαλείων που προσφέρουν τα frameworks της Angular και της NestJS.

«Submission Upload Web System»

«Kordatzakis Thodoris»

«Dramitinos Vasileios»

Abstract

Each academic semester, the evaluation of students through assignments, exercises, and examinations constitutes a fundamental and necessary part of the educational process. In order to manage this process efficiently and ensure its smooth operation, a system is required that enables students to submit their assignments digitally, while also allowing academic staff to monitor and assess them. The existing system has provided a satisfactory and effective solution for the basic functionalities required in such a context. However, the continuously increasing demands of the educational process highlight the need for a more modern and enhanced application, one that addresses the evolving needs of both students and instructors. This includes improving existing features while also introducing new capabilities. The aim of this thesis is to implement such a system, designed to be user-friendly and accessible, by utilizing modern technologies and tools offered by the Angular and NestJS frameworks.

Ευχαριστίες

Ευχαριστούμε θερμά τον Καθηγητή και Επιβλέπων Σιδηρόπουλο Αντώνη , του οποίου η καθοδήγηση και γνώση υπήρξαν καθοριστικές για την επιτυχημένη ολοκλήρωση της εργασίας αυτής. Μέσα από τη συνεργασία μας είχαμε την ευκαιρία να διευρύνουμε τις γνώσεις μας και να εξερευνήσουμε νέες προοπτικές.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Κώδικα

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
API	Application Programming Interface
UI	User Interface
UX	User Experience
DOM	Document Object Model
URL	Uniform Resource Locator
SPA	Single Page Application
SQL	Structured Query Language
NPM	Node Package Manager

Κεφάλαιο 1ο: Μελέτη και Εξέλιξη του Exams Submission Tool

1.1 Εισαγωγή

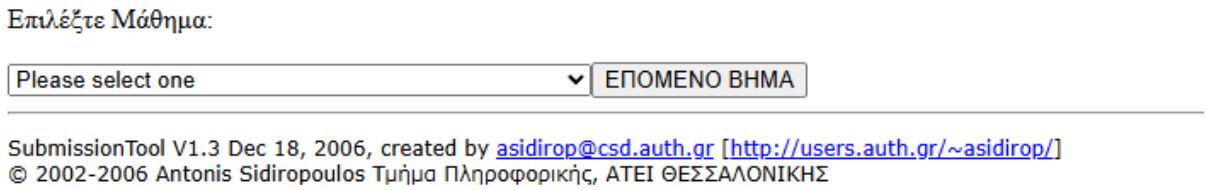
Το σύστημα Exams Submission Tool, το οποίο δημιουργήθηκε από τον καθηγητή Σιδηρόπουλο Αντώνη και χρησιμοποιείται από το τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος , είναι ένα σύστημα το οποίο είχε φτιαχτεί στα μέσα της δεκαετίας του 2000 με σκοπό τον εκσυγχρονισμό της εκπαιδευτικής διαδραστικότητας και της άμεσης αξιολόγησης ανάμεσα σε καθηγητή και φοιτητή. Η ύπαρξη ενός τέτοιου συστήματος στην εκπαιδευτική διαδικασία του πανεπιστημίου είναι ιδιαίτερα σημαντική και χρήσιμη, καθώς επιτρέπει στον καθηγητή να ενημερώνει άμεσα τον φοιτητή για νέες ασκήσεις και υποβολές που αναμένονται να γίνουν από αυτόν και στον φοιτητή να υποβάλλει τις ασκήσεις του με αμεσότητα, διαφάνεια και ασφάλεια. Είναι ένα εργαλείο, του οποίου η συνεισφορά βελτιώνει σε μεγάλο βαθμό την ποιότητα της εκπαιδευτικής εμπειρίας του φοιτητή, καθώς και την διδασκαλία του καθηγητή.

1.2 Δομή του Συστήματος

Η δομή και η λειτουργία ενός τέτοιου συστήματος είναι απαραίτητο να σχεδιαστούν και να υλοποιηθούν με τρόπο απλό και κατανοητό, έτσι ώστε να είναι πραγματικά φιλικό προς τον τελικό χρήστη.

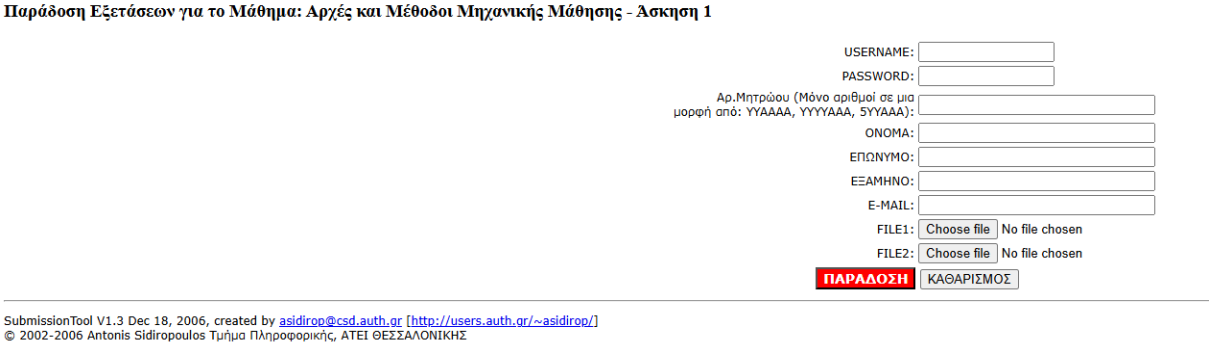
Ο στόχος είναι κάθε χρήστης, ανεξάρτητα από το επίπεδο εξοικείωσης του με την τεχνολογία, να μπορεί να το χρησιμοποιήσει με άνεση και χωρίς να χρειάζεται ιδιαίτερες γνώσεις ή εκπαίδευση. Ιδανικά, το σύστημα θα πρέπει να καθοδηγεί τον χρήστη με φυσικό τρόπο, να μην προκαλεί σύγχυση και να μην τον δυσκολεύει κατά τη διάρκεια της αλληλεπίδρασης. Έτσι, ακόμα και ο πιο απλός ή άπειρος χρήστης θα μπορεί να αξιοποιήσει πλήρως τις δυνατότητές του, χωρίς να αποθαρρύνεται ή να χρειάζεται βοήθεια για τη βασική του χρήση. Στην προκειμένη περίπτωση του Exams Submission Tool, η εμπειρία του χρήστη και ο τρόπος με τον οποίο είναι δομημένο είναι αρκετά φιλικό και εύχρηστο.

Με την είσοδο του στην πλατφόρμα (Εικονα 1.1) , ο χρήστης καλείται να επιλέξει την άσκηση για το μάθημα που τον ενδιαφέρει και επιθυμεί να κάνει μία υποβολή. Η επιλογή πραγματοποιείται μέσω ενός πεδίου επιλογής (select box). Με το πάτημα επάνω στο πεδίο, εμφανίζεται μία λίστα με όλα τα διαθέσιμα μαθήματα, από την οποία ο χρήστης μπορεί να διαλέξει εκείνο που τον αφορά. Η διαδικασία αυτή έχει ως στόχο να διασφαλίσει ότι η υποβολή του χρήστη θα καταχωρηθεί σωστά και θα αντιστοιχιστεί στο σωστό μάθημα.



Εικόνα 1.1 Στιγμιότυπο από την πρώτη οθόνη που βλέπει ο χρήστης

Αφού ο χρήστης πατήσει το κουμπί με την ένδειξη «ΕΠΟΜΕΝΟ ΒΗΜΑ», μεταφέρεται αυτόματα στην επόμενη οθόνη της διαδικασίας. Σε αυτήν την οθόνη εμφανίζεται μια φόρμα, την οποία καλείται να συμπληρώσει ώστε να μπορέσει να προχωρήσει στην οριστική υποβολή των στοιχείων του. Η φόρμα αυτή είναι σχεδιασμένη με τρόπο ώστε να συγκεντρώνει όλες τις απαραίτητες πληροφορίες που απαιτούνται για την ολοκλήρωση της διαδικασίας. Συγκεκριμένα, περιλαμβάνει πεδία για την εισαγωγή ονόματος χρήστη (Username), κωδικού πρόσβασης (Password), αριθμού μητρώου, ονόματος, επωνύμου, εξαμήνου φοίτησης, του ιδρυματικού email του χρήστη και μεταφόρτωσης των απαραίτητων αρχείων που απαιτούνται. Όλα τα παραπάνω συνθέτουν ένα βήμα σημαντικό και απαραίτητο για την επιτυχή ολοκλήρωση της διαδικασίας από τον χρήστη.



Εικόνα 1.2 Στιγμιότυπο οθόνης με τη φόρμα προς συμπλήρωση

Τέλος, ο χρήστης προχωρά στην ολοκλήρωση της διαδικασίας πατώντας το κουμπί με την ένδειξη «ΠΑΡΑΔΟΣΗ». Με την ενέργεια αυτή, το σύστημα καταχωρεί τα δεδομένα που έχουν εισαχθεί και εκτελεί την υποβολή της άσκησης. Εφόσον η υποβολή ολοκληρωθεί επιτυχώς, εμφανίζεται στην οθόνη ένα κατάλληλο μήνυμα επιβεβαίωσης, το οποίο ενημερώνει τον χρήστη ότι η διαδικασία πραγματοποιήθηκε με επιτυχία.

1.3 Στόχος εργασίας

Στόχος της συγκεκριμένης πτυχιακής εργασίας είναι η ανακατασκευή και εξέλιξη του υπάρχοντος συστήματος διαχείρισης ασκήσεων, το οποίο αποτέλεσε το σημείο εκκίνησης για την ανάπτυξη μιας πιο σύγχρονης και αποδοτικής λύσης. Πριν ξεκινήσει η διαδικασία δημιουργίας του νέου συστήματος, έγινε λεπτομερής μελέτη και ανάλυση της υπάρχουσας πλατφόρμας, προκειμένου να κατανοηθούν πλήρως οι λειτουργίες της, τα δυνατά της σημεία, αλλά και οι περιορισμοί της.

Βασισμένο σε αυτά τα δεδομένα, σχεδιάστηκε από την αρχή ένα νέο σύστημα υλοποιημένο με πιο σύγχρονες τεχνολογίες, το οποίο δεν αναπαράγει απλώς τις βασικές δυνατότητες του προηγούμενου, αλλά τις εξελίσσει και τις ενισχύει με επιπλέον χαρακτηριστικά. Οι νέες λειτουργίες ενσωματώθηκαν με σκοπό να καλύψουν πιο αποτελεσματικά τις ανάγκες των χρηστών, όπως αυτές διαμορφώνονται στο σημερινό, απαιτητικότερο εκπαιδευτικό περιβάλλον.

1.4 Προσθήκες και Εξέλιξη

Αφού προηγήθηκε προσεκτικός σχεδιασμός και σκέψη πάνω στις ανάγκες του συστήματος, αποφασίστηκε πως θα ήταν χρήσιμο και ουσιαστικό να προστεθούν ή να βελτιωθούν ορισμένα χαρακτηριστικά. Τα παρακάτω στοιχεία κρίθηκαν ως τα πιο σημαντικά για να ενισχύσουν τη λειτουργικότητα και τη συνολική εμπειρία χρήσης της πλατφόρμας.

- **Δυνατότητα σύνδεσης του χρήστη**. Πλέον, ο χρήστης μπορεί να συνδεθεί στο σύστημα χρησιμοποιώντας τα στοιχεία του ιδρυματικού του λογαριασμού. Ανάλογα με τον ρόλο του, μεταφέρεται αυτόματα στο κατάλληλο περιβάλλον. Αν πρόκειται για φοιτητή, οδηγείται σε μια φόρμα που καλείται να συμπληρώσει για να ανεβάσει τις ασκήσεις που επιθυμεί. Αντίθετα, αν είναι διδάσκων, μεταφέρεται στην αντίστοιχη πλατφόρμα του προσωπικού, όπου έχει τη δυνατότητα να διαχειρίζεται πλήρως τις υποβολές των φοιτητών και τα μαθήματα που διδάσκει.
- **Ιστορικό του φοιτητή**. Όταν ένας χρήστης, που έχει ρόλο φοιτητή, συνδεθεί στην πλατφόρμα με τα ιδρυματικά του στοιχεία, αποκτά άμεση πρόσβαση στο προσωπικό του ιστορικό. Εκεί μπορεί να δει συγκεντρωμένα όλες τις πληροφορίες ασκήσεων που έχει ανεβάσει μέχρι σήμερα. Η λίστα αυτή περιλαμβάνει χρήσιμες πληροφορίες, όπως το όνομα του μαθήματος στο οποίο ανήκει κάθε άσκηση, την ημερομηνία που πραγματοποιήθηκε η υποβολή, καθώς και τα ίδια τα αρχεία που ανέβασε. Με αυτόν τον τρόπο, ο φοιτητής έχει πάντα τον έλεγχο και μπορεί να ανατρέχει εύκολα στις προηγούμενες υποβολές του.
- **Μεταφορά ασκήσεων.** Ο διδάσκων, μέσα από το προσωπικό του περιβάλλον στην πλατφόρμα, έχει πλέον τη δυνατότητα να επιλέγει και να μεταφέρει εύκολα τις ασκήσεις από ένα εξάμηνο σε κάποιο άλλο της επιλογής του. Η δυνατότητα αυτή τον απαλλάσσει από την κουραστική και χρονοβόρα διαδικασία της εκ νέου χειροκίνητης δημιουργίας των ίδιων ασκήσεων, προσφέροντάς του ευκολία και εξοικονόμηση χρόνου.

1.5 Οργάνωση εργασίας

Στο επόμενο κεφάλαιο θα αναφέρουμε όλες τις τεχνολογίες και αρχιτεκτονικές που χρησιμοποιήθηκαν για την σχεδίαση και ανάπτυξη του συστήματος. Αυτές περιλαμβάνουν γλώσσες προγραμματισμού, περιβάλλον ανάπτυξης, frameworks κλπ.

Στο τρίτο κεφάλαιο θα αναλύσουμε τον τρόπο και τη ροή ανάπτυξης του συστήματος. Θα αναφερθούμε στην αρχιτεκτονική της εφαρμογής, τα βήματα δημιουργίας των συστατικών του κώδικα, τον σχεδιασμό και την δομή της βάσης δεδομένων, καθώς και την διαδικασία μέσα από την οποία περιηγείται ο χρήστης στην εφαρμογή.

Τέλος, στο τέταρτο κεφάλαιο αναφέρουμε τα συμπεράσματα στα οποία καταλήξαμε μέσα από την ανάπτυξη της εφαρμογής, τα αποτελέσματα της εργασίας και προτάσεις-βελτιώσεις για πιθανή μελλοντική επέκταση και εξέλιξη του συστήματος.

Κεφάλαιο 2ο: Τεχνολογίες

2.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζονται και αναλύονται οι τεχνολογίες που αξιοποιήθηκαν για την υλοποίηση της εφαρμογής. Η αρχιτεκτονική της βασίζεται σε δύο κύρια μέρη: την εφαρμογή ιστού (frontend), η οποία έχει αναπτυχθεί με το framework Angular, και το σύστημα εξυπηρέτησης (backend), το οποίο υλοποιήθηκε με χρήση Node.js και πιο συγκεκριμένα του NestJS framework.

Η αρχιτεκτονική της εφαρμογής ακολουθεί τον διαχωρισμό σε frontend (εφαρμογή ιστού) και backend (σύστημα εξυπηρέτησης). Συγκεκριμένα, το frontend έχει αναπτυχθεί με τη χρήση του Angular, ενός σύγχρονου, δυναμικού framework που βασίζεται στη γλώσσα TypeScript και είναι σχεδιασμένο για την κατασκευή επεκτάσιμων και διαδραστικών εφαρμογών ιστού. Από την άλλη πλευρά, το backend έχει υλοποιηθεί με τη Node.js πλατφόρμα και συγκεκριμένα πάνω στο framework της NestJS, ένα modular framework που προσφέρει σταθερότητα, ευκολία στη δομή του κώδικα και εξαιρετική υποστήριξη για την ανάπτυξη RESTful APIs.

Η εφαρμογή υποστηρίζει δύο βασικά περιβάλλοντα λειτουργίας: ένα για τον διδάσκοντα και ένα για τον φοιτητή. Κατά την είσοδο στην εφαρμογή, ο χρήστης αυθεντικοποιείται και βάσει των διαπιστευτηρίων του, μεταφέρεται αυτόματα στο αντίστοιχο περιβάλλον. Αυτός ο διαχωρισμός επιτρέπει την καλύτερη οργάνωση και ασφαλέστερη χρήση των λειτουργιών.

Το περιβάλλον του διδάσκοντα λειτουργεί ως ένα διαχειριστικό πάνελ, μέσα από το οποίο μπορεί να παρακολουθεί, να οργανώνει και να διαχειρίζεται τις υποβολές των φοιτητών, καθώς και τα διαθέσιμα μαθήματα. Του δίνεται η δυνατότητα ελέγχου των καταχωρημένων δεδομένων, προβολής των υποβολών και παρακολούθησης της συνολικής πορείας των φοιτητών.

Το περιβάλλον του φοιτητή, αντίστοιχα, είναι σχεδιασμένο με έμφαση στην ευχρηστία και τη φιλικότητα προς τον χρήστη. Μέσω μιας απλής και κατανοητής διεπαφής, κάθε φοιτητής μπορεί εύκολα να συμπληρώσει και να αποστείλει τις απαιτούμενες πληροφορίες για την υποβολή μιας άσκησης για το μάθημα που τον ενδιαφέρει..

Η επιλογή των τεχνολογιών Angular και NestJS κρίθηκε ιδιαίτερα σημαντική, καθώς προσφέρουν υψηλή απόδοση, καθαρή δομή, ευκολία συντήρησης του κώδικα, καθώς και δυνατότητα μελλοντικής επέκτασης του συστήματος, σύμφωνα με τις ανάγκες που ενδέχεται να προκύψουν σε επόμενες φάσεις ανάπτυξης. Επιπλέον, οι κοινότητες χρηστών και οι ομάδες υποστήριξης των τεχνολογιών αυτών είναι αρκετά διαδεδομένες, με αποτέλεσμα να είναι πιο εύκολη η ενσωμάτωση νέων λειτουργιών και η προσαρμογή της εφαρμογής σε νέα περιβάλλοντα χρήσης.

2.2 Angular

Η Angular είναι ένα web framework, το οποίο δίνει την δυνατότητα στον προγραμματιστή να χτίσει αποδοτικά, γρήγορα και αξιόπιστα συστήματα web. Η δημιουργία και η συντήρηση της γίνεται από την Google και παρέχει μία αρκετά μεγάλη γκάμα από έτοιμα εργαλεία, βιβλιοθήκες και APIs με στόχο να απλοποιήσει, να διευκολύνει και να θέσει μια συγκεκριμένη και λογικευμένη ροή ανάπτυξης και εργασίας. Βασιζόμενη στην γλώσσα Typescript, η οποία είναι ένα υπερσύνολο της Javascript προσφέρει περισσότερα εργαλεία και ασφάλεια τύπων (type safety), είναι ένα framework με σαφή πλεονεκτήματα,

ενώ επίσης η καθορισμένη δομή που παρέχει βοηθάει στην εύκολη αλληλοϋποστήριξη των προγραμματιστών , ειδικότερα σε μεγάλες επεκτάσιμες και συντηρήσιμες εφαρμογές.

Η αρχιτεκτονική της Angular χρησιμοποιεί μια δομική προσέγγιση κατά την οποία γίνεται διαχωρισμός ανάμεσα στη λογική και την παρουσίαση, ενώ κρίσιμο συστατικό της είναι η επαναχρησιμοποίηση τμημάτων του κώδικα.

Ειδικότερα, τα συστατικά του κώδικα της Angular βαφτίζονται ανάλογα με την λογική τους. Ακολουθούν ως εξής :

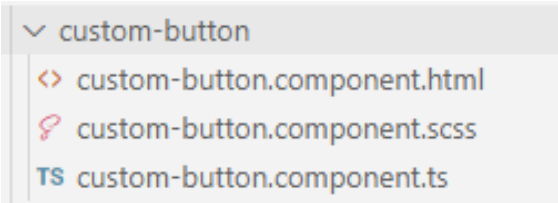
1. **Components:** Τα βασικά συστατικά του UI. Κάθε component περιλαμβάνει ένα HTML αρχείο όπου κατοικούν τα elements του UI , ένα αρχείο που περιέχει την κλάση που φιλοξενεί τον κώδικα Typescript για τη λογική λειτουργίας ,ένα CSS αρχείο για μορφοποίηση και ένα προαιρετικό αρχείο testing για σκοπούς testing.
2. **Services**: Περιέχουν κοινή λογική και λειτουργίες (π.χ. διαχείριση δεδομένων, επικοινωνία με APIs) και μπορούν να χρησιμοποιηθούν από πολλά components. Επίσης, επιτρέπουν την επικοινωνία σε components που δεν έχουν σχέση γονέα-παιδιού.
3. **Modules:** Ομαδοποιούν components και άλλες λειτουργίες, διευκολύνοντας έτσι την διαχείριση και τη συντήρηση της εφαρμογής. Επιπλέον, συμβάλλει στην βέλτιστη δομή και απόδοση του κώδικα , καθώς τμηματοποιεί την εφαρμογή σε πολλά modules, με κάθε module να είναι υπεύθυνο για ένα συγκεκριμένο κομμάτι.
4. **Directives**: Τα directives είναι κλάσεις, οι οποίες προσδίδουν επιπλέον συμπεριφορά στον τρόπο με τον οποίο διαχειρίζονται και προβάλλονται τα δεδομένα στο UI.
5. **Routing:** Με το routing της angular, αλλάζει το περιεχόμενο-σελίδα που βλέπει ο χρήστης, το οποίο συνδέεται με συγκεκριμένα components, χωρίς να πρέπει να γίνει κατέβασμα της σελίδας από τον διακομιστή (server).
6. **Guards**: Αρχεία που προστατεύουν διαδρομές από μη εξουσιοδοτημένους χρήστες.

Παρακάτω, αναλύεται λεπτομερώς και επεξηγείται η λειτουργία και η λογική των συστατικών κώδικα στο framework της Angular.

2.2.1 Components

Τα components της Angular είναι συστατικά του UI, τα οποία προβάλλουν περιεχόμενο και εκτελούν λογική κώδικα Typescript. Κάθε component αποτελείται από 4 αρχεία:

1. Ένα αρχείο HTML, το οποίο περιέχει τα συστατικά του DOM που συνιστούν το component.
2. Ένα αρχείο Typescript, που αποτελεί την λογική που εκτελεί το component.
3. Ένα αρχείο CSS στο οποίο περιέχεται ο κώδικας για την μορφοποίηση του περιεχομένου.
4. Ένα προαιρετικό αρχείο test ing, το οποίο χρησιμοποιείται για testing σκοπούς.



Εικόνα 2.1 Παράδειγμα δομής ενός component

Η λογική της δομής του κώδικα είναι να τμηματοποιεί η λογική της εφαρμογής σε components, καθένα το οποίο έχει συγκεκριμένο σκοπό και λογική. Η λογική αυτή των components είναι σχεδιασμένη έτσι ώστε κάθε component να είναι επαναχρησιμοποιήσιμο, να μπορεί δηλαδή να χρησιμοποιηθεί ξανά και ξανά το ίδιο κομμάτι κώδικα χωρίς να χρειάζεται να το ξανά γράψει με το χέρι από την αρχή ο προγραμματιστής. Άπαξ και δημιουργήσει το component με τη λογική που επιθυμεί, μπορεί μετά να εφαρμόσει αυτό το component σε όποιο μέρος θέλει. Ο σχεδιασμός αυτός βοηθάει στην συνεργασία μεταξύ των προγραμματιστών και τους γλιτώνει πολύτιμο χρόνο και κόπο.

2.2.1.1 HTML

Το HTML είναι μια Γλώσσα Χαρακτηρισμού Υπερ-Κειμένου (HyperText Markup Language), με την οποία δημιουργείται και δομείται μία ιστοσελίδα στο διαδίκτυο. Με το HTML ορίζονται τα βασικά συστατικά του DOM , όπως τίτλοι, παράγραφοι, σύνδεσμοι, φόρμες, εικόνες κτλ. Δεν είναι γλώσσα προγραμματισμού, είναι μία γλώσσα σήμανσης που περιγράφει τη δομή και το περιεχόμενο που φιλοξενείται από μια ιστοσελίδα έτσι ώστε να μπορεί να διαβαστεί από έναν φυλλομετρητή.

2.2.1.2 CSS

Το CSS (Cascading Style Sheets) είναι η γλώσσα που χρησιμοποιείται για να προστεθεί μορφοποίηση σε ένα αρχείο HTML. Δηλαδή, με τη χρήση CSS σε ένα αρχείο HTML καθορίζεται η εμφάνιση των στοιχείων HTML και ορίζονται ιδιότητες όπως χρώματα, γραμματοσειρές, αποστάσεις, διατάξεις, animations και άλλα οπτικά χαρακτηριστικά, κάνοντας έτσι τις ιστοσελίδες πιο ευχάριστες και φιλικές προς τον χρήστη. Στην ουσία, το CSS είναι αυτό που διαχωρίζει σε μία σελίδα την εμφάνιση από το περιεχόμενο, δίνοντας έτσι την δυνατότητα να σχεδιαστούν και να μορφοποιηθούν σελίδες με γρήγορο και εύκολο τρόπο.

2.2.1.3 Typescript

Typescript είναι γλώσσα προγραμματισμού, όπου δημιουργήθηκε από την Microsoft και χρησιμοποιείται για την εκτέλεση της λογικής της εφαρμογής. Είναι ένα υπερσύνολο της Javascript και την επεκτείνει με τρόπους που βελτιώνουν την ασφάλεια και τον εντοπισμό σφαλμάτων κατά τον χρόνο μεταγλώττισης (compile time) σε αντίθεση με την Javascript που το κάνει κατα τον χρόνο εκτέλεσης (runtime). Προσδίδει επιπλέον συντακτικούς ελέγχους και χρήση στατικής τυποποίησης (static typing), δηλαδή ορίζονται ρητά οι τύποι των μεταβλητών, παραμέτρων και επιστρεφόμενων τιμών με αποτέλεσμα τον άμεσο εντοπισμό λαθών. Ο κώδικας Typescript μεταγλωτίζεται στη συνέχεια σε κώδικα Javascript, έτσι ώστε να μπορεί να μεταφραστεί και να αποδοθεί σε οποιοδήποτε περιβάλλον Javascript , όπως φυλλομετρητές. Η Typescript χρησιμοποιείται στις εφαρμογές Angular, καθώς και σε Backend εφαρμογές λόγω της αξιοπιστίας και της επεκτασιμότητας της.

2.2.1.4 Inputs / Outputs

Σε ένα Angular component, χρησιμοποιούνται inputs και outputs για την επικοινωνία μεταξύ δύο component που έχουν σχέση γονέα-παιδιού.

Ειδικότερα, για τα inputs χρησιμοποιείται το decorator @Input() για να ορίσει μία μεταβλητή η οποία αναμένεται να περαστεί στο component αυτό από κάποιο γονικό component. Η λειτουργία αυτή είναι χρήσιμη για να λαμβάνονται τιμές, όπως μεταβλητές ή αντικείμενα, στο child component μέσω των attributes του HTML tag του.

```
export class ModalContainerComponent {
@Input() title?:string
}
```

Κώδικας 2.1 Παράδειγμα ορισμού Input μεταβλητής σε component

```
<app-modal-container
 [title]="'Εισαγωγή ασκήσεων παλαιότερης περιόδου'"
 (cancelChange)="dialogRef.close()"
 (submitChange)="submit()">
</app-modal-container>
```

Κώδικας 2.2 Παράδειγμα χρήσης Input του child component από το parent component

Όσον αφορά τα @Output() decorators , χρησιμοποιούνται από το child component για να εκπέμψει γεγονότα και δεδομένα προς το parent component χρησιμοποιώντας το EventEmitter(). Τέτοια γεγονότα μπορεί να περιλαμβάνουν το πάτημα ενός κουμπιού ή οποιαδήποτε άλλη δράση θα ορίσει ο προγραμματιστής.

Ο συμβατικός τρόπος ονομασίας @Output decorators είναι <όνομα γεγονότος>Change όπως φαίνεται στον Κώδικα 2.3

```
export class CustomButtonComponent {
  @Output() clickChange = new EventEmitter<void>();
}
```

Κώδικας 2.3 Παράδειγμα ορισμού Output στο child component που ενημερώνει το parent component για το πάτημα ενός κουμπιού

```
<app-custom-button
      (clickChange)="handleAddCoursePeriod()">
</app-custom-button>
```

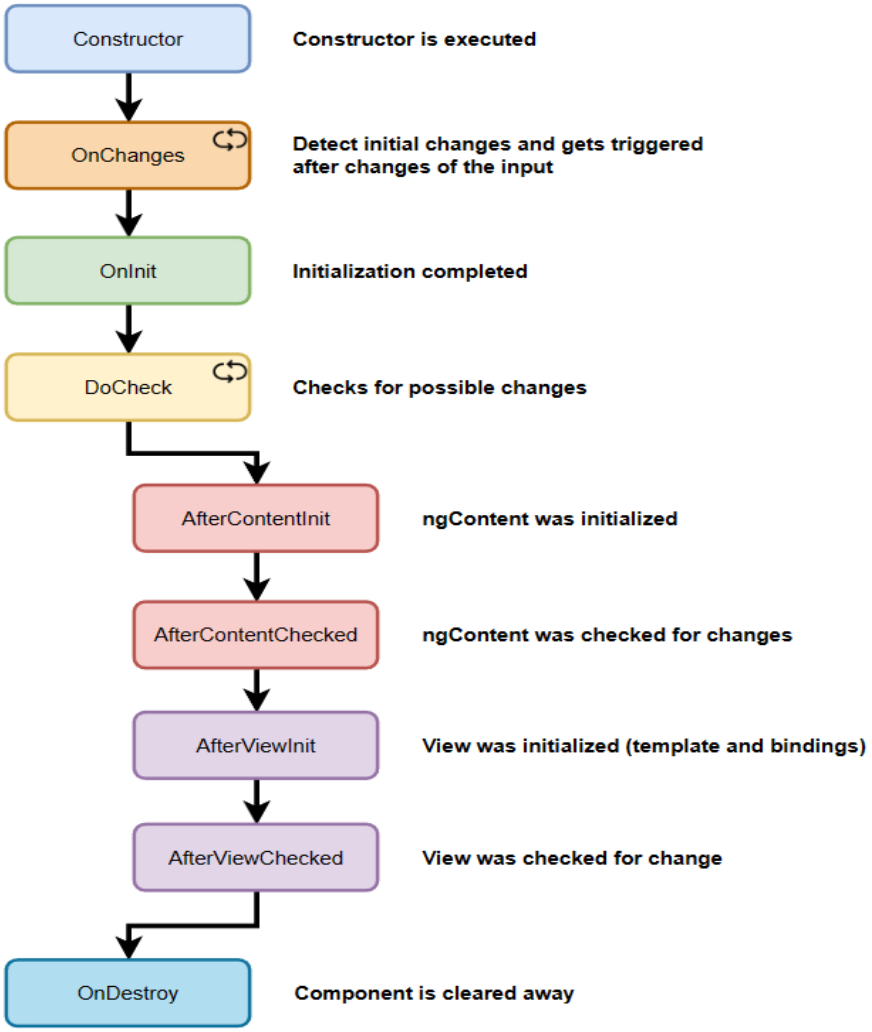
Κώδικας 2.4 Παράδειγμα χρήσης output event από το parent component

2.2.1.5 Lifecycle Hooks

Ο κύκλος ζωής ενός component είναι τα στάδια που μεσολαβούν από τη στιγμή δημιουργίας του component μέχρι τη στιγμή της καταστροφής του. Για κάθε στάδιο ζωής ενός component, υπάρχει και το αντίστοιχο LifeCycle hook, το οποίο είναι μία μέθοδος η οποία καλείται όταν φτάσει το component στο αντίστοιχο στάδιο κύκλου ζωής. Αφού εκτελεστεί ο δομητής (constructor) του component, ακολουθεί η εκτέλεση κατα σειρά των κύκλων ζωής ως εξής:

1. ngOnChanges

2. ngOnInit
3. ngDoCheck
4. ngAfterContentInit
5. ngAfterContentInitChecked
6. ngAfterViewInit
7. ngAfterViewInitChecked
8. ngOnDestroy



Εικόνα 2.2 Κύκλος ζωής ενός Angular Component

2.2.1.6 ngOnChanges

Η μέθοδος ngOnChanges εκτελείται μία φορά κατά την δημιουργία του component πριν την ngOnInit και κάθε φορά που αλλάζουν τα Inputs του component. Η μέθοδος αυτή περιέχει μία παράμετρο SimpleChanges , η οποία περιέχει τις καινούριες και τις προηγούμενες τιμές των Inputs μεταβλητών.

2.2.1.7 ngOnInit

Η μέθοδος ngOnInit τρέχει μόνο μία φορά κατά τη διάρκεια ζωής ενός component, αφού η Angular έχει αρχικοποιήσει όλες τις μεταβλητές Input με τις αρχικές τους τιμές. Αυτό το στάδιο ζωής πραγματοποιείται πριν αρχικοποιηθεί το πρότυπο (template) του component. Αυτό σημαίνει ότι μπορεί να γίνει ενημέρωση της κατάστασης του component βάσει των αρχικών τιμών των Inputs.

2.2.1.8 ngDoCheck

Η μέθοδος ngDoCheck εκτελείται πριν από κάθε φορά που η Angular ελέγχει ένα component για αλλαγές στο πρότυπο του. Το hook αυτό χρησιμοποιείται για να γίνει manual έλεγχος για αλλαγές στην κατάσταση ενός component εκτός του κανονικού αυτόματου ελέγχου για αλλαγές κατάστασης που έχει η Angular. Η μέθοδος αυτή συνιστάται να αποφεύγεται καθώς εκτελείται ιδιαίτερα συχνά και μπορεί να έχει βλαβερή επίδραση στην απόδοση της σελίδας. Χρησιμοποιείται μόνο όταν δεν υπάρχει διαφορετική εναλλακτική.

2.2.1.9 ngAfterContentInit

Η ngAfterContentInit εκτελείται μόλις αρχικοποιηθούν όλα τα child components μέσα σε ένα component. Αντιδρά όταν η Angular στείλει εξωτερικό περιεχόμενο στο view του component ή στο view στο οποίο υπάρχει ένα directive.

2.2.1.10 ngAfterContentChecked

Η μέθοδος ngAfterContentInitChecked εκτελείται κάθε φορά που τα εμφωλευμένα παιδιά του component έχουν ελεγχθεί για αλλαγές. Αυτό το lifecycle hook τρέχει πολύ συχνά και ενδέχεται να έχει αρνητικό αντίκτυπο στην απόδοση της σελίδας. Για το λόγο αυτό, συνίσταται να μην χρησιμοποιείται συχνά και μόνο όταν δεν υπάρχει άλλη εναλλακτική.

2.2.1.11 ngAfterViewInit

Το lifecycle hook ngAfterViewInit τρέχει μία φορά αφού έχουν αρχικοποιηθεί τα child components στο view του component. Είναι χρήσιμο όταν χρειάζεται η αλληλεπίδραση με τα στοιχεία του view του component, αφού έχουν αποδοθεί (rendered) στην οθόνη.

2.2.1.12 ngAfterViewChecked

Το ngAfterViewChecked lifecycle hook της Angular εκτελείται κάθε φορά που τα παιδιά του view του component έχουν ελεγχθεί για αλλαγές, το οποίο συνίσταται να αποφεύγεται η χρήση του όταν αυτό είναι δυνατό και να χρησιμοποιείται μόνο όταν δεν υπάρχει εναλλακτική λύση, καθώς εκτελείται πολύ συχνά με αποτέλεσμα να ζημιώνεται η απόδοση της εφαρμογής.

2.2.1.13 ngOnDestroy

Η μέθοδος ngOnDestroy ενός Component της Angular εκτελείται μία φορά μόλις πριν καταστραφεί ένα component. Η Angular καταστρέφει ένα component όταν ο χρήστης της εφαρμογής αλλάξει σελίδα στην οποία δεν υπάρχει πλέον το component ή όταν η απόδοση ενός component εξαρτάται από μια συνθήκη if.

2.2.2 Services

Τα services της Angular είναι κλάσεις, οι οποίες περιέχουν λογική για έναν συγκεκριμένο, καθορισμένο σκοπό και είναι στοχευμένες στο να εξυπηρετούν ένα μεμονωμένο χαρακτηριστικό (feature) της

εφαρμογής. Για το λόγο αυτό, είναι κρίσιμης σημασίας να κάνουν αυτό για το οποίο είναι προορισμένες να κάνουν, σωστά και αποδοτικά. Στην Angular διαχωρίζεται η λογική των components από την λογική των services για να αυξηθεί η κατάτμηση και η επαναχρησιμοποίηση (reusability) του κώδικα.

Ιδανικά, ο ρόλος του component είναι στοχευμένος προς την εμπειρία του χρήστη (User Experience) και στο να παρουσιάζει ιδιότητες και μεθόδους για τα δεδομένα που μεσολαβούν ανάμεσα στο view και στην λογική της εφαρμογής. Το view είναι υπεύθυνο για την απόδοση και την παρουσίαση των δεδομένων στην οθόνη και η λογική της εφαρμογής είναι υπεύθυνη για την διαχείριση και την χειραγώγηση αυτών των δεδομένων.

Υπάρχουν κάποιες εργασίες, οι οποίες δεν περιλαμβάνουν αλληλεπίδραση με το view ή την λογική της εφαρμογής, δηλαδή την κλάση του component. Τέτοιες εργασίες είναι κατάλληλες για να εκτελούνται στα services και περιλαμβάνουν την απόκτηση δεδομένων από έναν διακομιστή, την επικύρωση των στοιχείων που έχει εισάγει ο χρήστης ή ακόμα και καταγραφή δεδομένων στην κονσόλα. Τα services αυτά είναι injectable κλάσεις, το οποίο σημαίνει ότι μπορούν να γίνουν inject σε οποιοδήποτε component. Έτσι, μπορεί και αποκτά πρόσβαση στις εργασίες του service οποιοδήποτε component.

```
export class Logger {
 log(msg: any) { console.log(msg); }
 error(msg: any) { console.error(msg); }
 warn(msg: any) { console.warn(msg); }
}
```

Κώδικας 2.5 Παράδειγμα κλάσης service

Στον Κώδικα 2.5 δίνεται ένα παράδειγμα ενός service με όνομα logger , το οποίο φιλοξενεί τρεις μεθόδους, οι οποίες καταγράφουν ένα μήνυμα στην κονσόλα.

```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

Κώδικας 2.6 Παράδειγμα χρήσης ενός service σε ένα component

Στον Κώδικα 2.6 φαίνεται ο τρόπος με τον οποίο γίνεται inject ένα service σε ένα component με την χρήση της μεθόδου inject() και στη συνέχεια το πως χρησιμοποιούνται οι μέθοδοι του service αυτού.

Εναλλακτικός τρόπος για να γίνει το service injection στο component, είναι η δήλωση ενός instance του service στον δομητή του component (Κώδικας 2.7)

```
constructor(private service: HeroService) { }
```

Κώδικας 2.7 Service injection μέσω του δομητή

2.2.2.1 Dependency Injection

Το Dependency Injection είναι το κομμάτι του framework της Angular, το οποίο επιτρέπει στα components να έχουν πρόσβαση στις λειτουργίες των services και άλλων πόρων. Η Angular παρέχει την δυνατότητα για χειροκίνητο injection ενός service μέσα σε ένα component έτσι ώστε να αποκτήσει πρόσβαση στις λειτουργίες του.

Με το @Injectable() decorator πάνω από μία κλάση, ενημερώνεται η Angular ότι αυτή η κλάση του service μπορεί να γίνει inject σε ένα component ως dependency (Κώδικας 2.8)

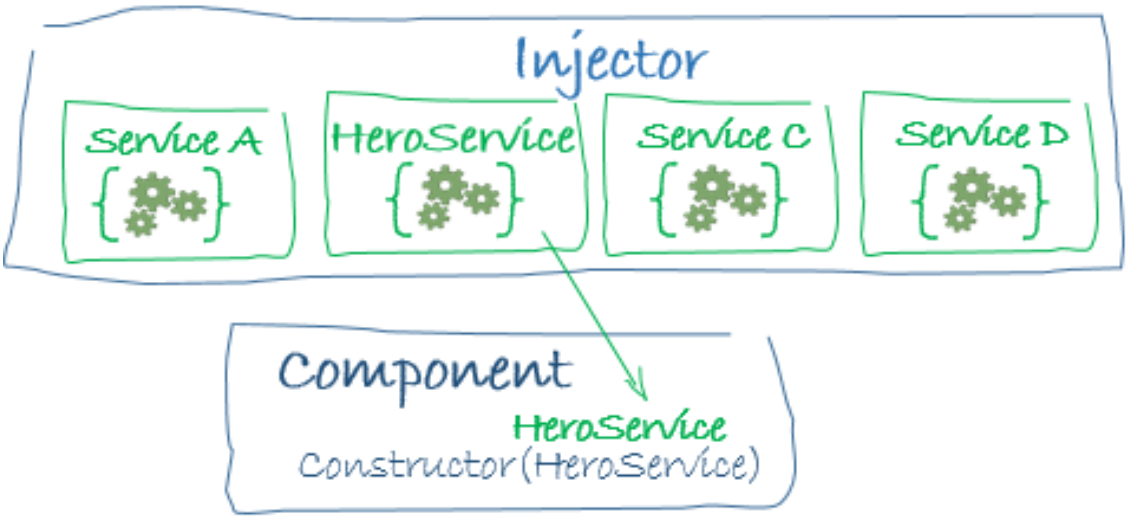
```
@Injectable({
  providedIn: 'root'
})
export class BucketService {}
```

Κώδικας 2.8 Χρήση Injectable() decorator πάνω από την κλάση

Η παράμετρος “providedIn” ενημερώνει την Angular για το επίπεδο στο οποίο θα γίνει καταχώρηση της κλάσης αυτής.

Όταν η Angular δημιουργεί ένα νέο instance μιας κλάσης component, προσδιορίζει ποια services ή άλλα dependencies χρειάζεται το component εξετάζοντας τους τύπους των παραμέτρων του δομητή. Όταν διαπιστωθεί ότι ένα component εξαρτάται από ένα service, ελέγχεται πρώτα αν ο injector έχει ήδη υπάρχοντα instances αυτού του service.

Αν δεν υπάρχει ακόμη κάποιο ζητούμενο instance του service, ο injector δημιουργεί ένα, χρησιμοποιώντας τον καταχωρημένο provider και το προσθέτει στον injector πριν επιστρέψει το service στην Angular. Αφού όλα τα ζητούμενα services έχουν επιλυθεί και επιστραφεί, η Angular μπορεί να καλέσει τον constructor του component με αυτά τα services ως ορίσματα.



Εικόνα 2.3 Παράδειγμα διαδικασίας service injection σε ένα component

2.2.3 Modules

Στην Angular, ένα module είναι μια βασική αρχιτεκτονική μονάδα που οργανώνει τον κώδικα μιας εφαρμογής σε λογικές ενότητες, επιτρέποντας τη διαχείριση και επαναχρησιμοποίηση των διαφόρων στοιχείων (components, directives, pipes και services).

Οποιοδήποτε Angular app περιλαμβάνει τουλάχιστον ένα module: το root module που ονομάζεται συνήθως AppModule. Μέσα στο module αυτό, δηλώνονται και ορίζονται όλα τα στοιχεία και συστατικά που φιλοξενούνται σε αυτό το module.

Ομαδοποιούν συναφή components, services, directives και pipes. Καθορίζουν ποια μέρη του κώδικα είναι ορατά και διαθέσιμα σε άλλα μέρη της εφαρμογής (μέσω exports και imports). Βοηθούν στη βελτιστοποίηση της φόρτωσης (lazy loading) τμημάτων της εφαρμογής. Παρέχουν διαφορετικά επίπεδα DI (Dependency Injection) μέσω των providers.

Μία εφαρμογή Angular θα μπορούσε να παρομοιαστεί με ένα παζλ, όπου κάθε κομμάτι του παζλ είναι ένα module. Χωρίς ένα από αυτά τα κομμάτια του παζλ (module) δεν είναι δυνατό να δούμε την πλήρη εικόνα. Ουσιαστικά, χωρίς την ομαδοποίηση και την κοινή διαχείριση που προσφέρουν τα modules στο framework της Angular , δεν είναι πιθανό να δούμε την πλήρη εικόνα.

```
@NgModule({
imports:       [CommonModule ],
declarations:  [CustomerComponent, NewItemDirective, OrdersPipe ], exports:
[CustomerComponent,     NewItemDirective,     OrdersPipe,     CommonModule,
FormsModule ] })
```

Κώδικας 2.9 Παράδειγμα ενός module αρχείου

Στο παράδειγμα του Κώδικα 2.9 υπάρχει ένα module αρχείο, το οποίο εισάγει κάποια components μέσω του πεδίου imports, τα οποία μπορούν να χρησιμοποιηθούν από τα κοινά components που βρίσκονται στο declarations πεδίο του ίδιου αρχείου. Τέλος, το πεδίο exports ορίζει τα components που εξάγει αυτό το module αρχείου και μπορούν να χρησιμοποιηθούν ως imports σε άλλα module αρχεία.

2.2.4 Standalone Components

Τα standalone components είναι μια νέα αρχιτεκτονική στα components που εφαρμόστηκε στις τελευταιες εκδόσεις της Angular. Προσφέρουν έναν πιο απλοποιημένο τρόπο με τον οποίο μπορούν να υλοποιηθούν Angular εφαρμογές. Με την εφαρμογή των standalone components, η ανάγκη για χρήση modules μειώνεται δραστικά, καθώς κάθε component που είναι standalone δεν χρειάζεται να φιλοξενείται μέσα σε κανένα module για να μπορέσει να χρησιμοποιηθεί. Ουσιαστικά, έχει δική του υπόσταση και είναι ανεξάρτητο, μπορεί δηλαδή να χρησιμοποιηθεί σε οποιοδήποτε μέρος χωρίς την ανάγκη ύπαρξης module.

Επίσης, τα standalone components φέρουν κάποιες από τις λειτουργίες των modules. Για παράδειγμα, ένα standalone component έχει στα ορίσματα του κι ένα πεδίο imports, το οποίο δέχεται σαν τιμή έναν πίνακα από άλλα standalone components, modules, directives και pipes, τα οποία χρειάζεται να

χρησιμοποιηθούν από το component αυτό. Είναι, δηλαδή, ένας τρόπος να χρησιμοποιεί το ένα συστατικό της Angular το άλλο.

```
@Component({
  selector: 'app-courses',
  standalone: true,
  imports: [
    CustomTableComponent,
    MatDialogModule,
    LoaderComponent,
    ProfessorDashboardContainerComponent,
    CustomButtonComponent,
  ],
  templateUrl: './courses.component.html',
  styleUrl: './courses.component.scss',
})
```

Κώδικας 2.10 Παράδειγμα standalone component

Όπως φαίνεται και στην Εικόνα 2.13, για να οριστεί ένα component ως standalone, πρέπει να οριστεί "true" το πεδίο "standalone" στα ορίσματα του component. Φαίνεται, επιπλέον, και ο πίνακας "imports", ο οποίος ορίζει ποια standalone components ή και modules, χρειάζεται να χρησιμοποιηθούν από το component. Τα πεδία "templateUrl" και "styleUrl" είναι τα μονοπάτια προς το αρχείο προτύπου και το αρχείο μορφοποίησης αντίστοιχα και τέλος, το πεδίο "selector" είναι η ονομασία του html tag με το οποίο χρησιμοποιείται αυτό το component στο html άλλων components.

Συνοπτικά, τα standalone components έχουν έρθει για να προσφέρουν μια πιο εύχρηστη και φιλική εμπειρία ως προς την διαχείριση της δομής των συστατικών της Angular στον κώδικα, ενώ προσδίδουν και στην απόδοση της εφαρμογής.

2.2.5 Directives

Τα directives είναι κλάσεις στο framework της Angular, οι οποίες προσφέρουν επιπλέον συμπεριφορά και αντιδραστικότητα στα στοιχεία και συστατικά σε μία εφαρμογή Angular. Η Angular παρέχει στον προγραμματιστή κάποια ενσωματωμένα built-in directives, τα οποία είναι αμέσως έτοιμα προς χρήση.

Η λειτουργία των directives ποικίλει από διαχείριση φορμών, λιστών και μορφοποίησης. Οι τύποι τους είναι οι εξής:

1. **Component Directives**: Χρησιμοποιείται μαζί με πρότυπο (template) και είναι ο πιο συνηθισμένος τύπος directive. Είναι τα components που έχουν ήδη αναφερθεί.
2. **Attribute Directives**: Αλλάζει την συμπεριφορά ή την εμφάνιση ενός στοιχείου (element)
3. **Structural Directives**: Χειραγωγεί την δομή του DOM αφαιρώντας ή προσθέτωντας DOM elements.

2.2.5.1 Attribute Directives

Τα attribute directives που παρέχει η Angular ακούν και επεξεργάζονται την συμπεριφορά άλλως HTML στοιχείων, attributes, πεδίων και components. Τα πιο συνηθισμένα attribute directives είναι το "NgClass", το "NgStyle" και το "NgModel".

Το "**NgClass**" προσθέτει και αφαιρεί κλάσεις CSS βάσει συνθήκης για τη δυναμική μορφοποίηση ενός στοιχείου. Για παράδειγμα, στον Κώδικα 2.11 αποδίδεται δυναμικά η CSS κλάση "submit-button" αν η

τιμή της μεταβλητής “isSubmit” είναι “true” και αντίστοιχα αν η τιμή της είναι “false”, αποδίδεται η CSS κλάση “button”.

```
<button [ngClass]="isSubmit?'submit-button':'button'">
</button>
```

Κώδικας 2.11 Παράδειγμα χρήσης NgClass attribute

Το “**NgStyle**” είναι το attribute με το οποίο δίνεται η δυνατότητα να οριστούν πολλαπλά δυναμικά inline css styles, προαιρετικά βάσει συνθήκης, απευθείας μέσα από το html tag του component. Για παράδειγμα, στον Κώδικα 2.12 ορίζονται δυναμικά οι CSS ιδιότητες “color” και “font-size”. Η ιδιότητα “color” δένεται δυναμικά με την τιμή της μεταβλητής “textColor” και η ιδιότητα “font-size” παίρνει την αντίστοιχη τιμή εξαρτώμενη από την συνθήκη της boolean τιμής της μεταβλητής “isH1”.

```
<span [style]="{'color':textColor,'font-
size':isH1?'15px':'10px'}">{{title}}
</span>
```

Κώδικας 2.12 Παράδειγμα χρήσης NgStyle attribute

Το “**NgModel**” προσφέρει την δυνατότητα να “δεθεί“ η τιμή ενός πεδίου εισαγωγής (input) με την επιθυμητή μεταβλητή και να προβληθεί στην οθόνη η τιμή του δεδομένου. Στο παράδειγμα του Κώδικα 2.13 δένεται η μεταβλητή “value” , η οποία έχει δηλωθεί στην κλάση typescript του component, με την τιμή του πεδίου input.

```
<input [type]="'text'" [(ngModel)]="value">
```

Κώδικας 2.13 Παράδειγμα χρήσης NgModel attribute

2.2.5.2 Structural Directives

Τα structural directives είναι directives, τα οποία είναι υπεύθυνα για την διαμόρφωση του DOM χειραγωγώντας HTML στοιχεία. Επεξεργάζονται το σχήμα της DOM ιεραρχίας προσθέτοντας, επεξεργάζοντας ή αφαιρώντας στοιχεία από το HTML. Τα πιο συνηθισμένα structural directives της Angular είναι το “NgIf”, το “NgFor” και το “NgSwitch”.

Το “**NgIf**” attribute προσθέτει ή αφαιρεί ένα HTML στοιχείο από ένα γονικό στοιχείο. Όταν η τιμή της μεταβλητής με την οποία είναι δεμένο το “NgIf” είναι false, τότε η Angular αφαιρεί το στοιχείο και τους απογόνους του από το DOM και στη συνέχεια “ξεφορτώνεται” το component , ελευθερώνοντας έτσι μνήμη και πόρους.

```
<app-item-detail *ngIf="isActive" [item]="item"></app-item-detail>
```

Κώδικας 2.14 Παράδειγμα χρήσης NgIf attribute

Το “**NgFor**” χρησιμοποιείται για την αναπαράσταση μιας λίστας στοιχείων στην σελίδα. Έχει την λογική της επαναληπτικής διαδικασίας for της Javascript. Πραγματοποιείται μία επαναλαμβανόμενη διαδικασία ενός πίνακα δεδομένων και αναπαριστώνται τα δεδομένα της τρέχουσας επανάληψης.

```
<div *ngFor="let item of items">{{ item.name }}</div>
```

Κώδικας 2.15 Παράδειγμα χρήσης NgFor attribute

Επίσης, μπορεί να χρησιμοποιηθεί για την επανάληψη ενός component όπως φαίνεται στον Κώδικα 2.16

```
<app-item-detail *ngFor="let item of items" [item]="item">
</app-item-detail>
```

Κώδικας 2.16 Παράδειγμα NgFor για επανάληψη component

Με παρόμοιο τρόπο με το switch της Javascript, το "**NgSwitch**" attribute χρησιμοποιείται για την προβολή ενός στοιχείου βασισμένο σε μία συνθήκη switch. Όπως φαίνεται στον Κώδικα 2.17, γίνεται χρήση του "NgSwitch" το οποίο έχει ως συνθήκη την τιμή της μεταβλητής "currentItem.feature". Ανάλογα με την τιμή της μεταβλητής αυτής, εμφανίζεται το κατάλληλο στοιχείο ή component με τη χρήση του "ngSwitchCase" , το οποίο ελέγχει την τιμή με την οποία αντιστοιχίζεται η προβολή της ανάλογης οθόνης.

```
<div [ngSwitch]="currentItem.feature">
 <app-stout-item    *ngSwitchCase="'stout'"
[item]="currentItem"></app-stout-item>

<app-device-item   *ngSwitchCase="'slim'"     [item]="currentItem"></app-
device-item>

 <app-lost-item     *ngSwitchCase="'vintage'"
[item]="currentItem"></app-lost-item>

 <app-best-item     *ngSwitchCase="'bright'"
[item]="currentItem"></app-best-item>

...

 <app-unknown-item  *ngSwitchDefault
[item]="currentItem"></app-unknown-item>
</div>
```

Κώδικας 2.17 Παράδειγμα χρήσης NgSwitch attribute

2.2.6 Routing

Σε μια Angular εφαρμογή, η αλλαγή του περιεχομένου που βλέπει ο χρήστης γίνεται με την εμφάνιση ή απόκρυψη συγκεκριμένων τμημάτων της οθόνης που αντιστοιχούν σε διάφορα components, χωρίς να απαιτείται φόρτωση νέας σελίδας από τον διακομιστή. Καθώς οι χρήστες εκτελούν ενέργειες μέσα στην εφαρμογή, χρειάζεται να μετακινούνται ανάμεσα σε διαφορετικά views που έχουν οριστεί.

Για να γίνει αυτή η πλοήγηση από το ένα view στο άλλο, χρησιμοποιείται το **Angular Router**. Το Router διευκολύνει την πλοήγηση, ερμηνεύοντας το URL του browser ως εντολή για να εντοπίσει το περιεχόμενο που αντιστοιχεί στο συγκεκριμένο URL και να το προβάλλει στην οθόνη..

Ουσιαστικά, το Angular Router προσφέρει τη δυνατότητα να υλοποιηθεί μια εμπειρία εφαρμογής που μοιάζει με desktop εφαρμογή, καθώς ο χρήστης κινείται άμεσα και ομαλά ανάμεσα σε σελίδες, χωρίς

καθυστερήσεις στην φόρτωση των σελίδων. Με τη χρήση routes, δίνεται η δυνατότητα να χαρτογραφηθούν συγκεκριμένα URLs σε συγκεκριμένα components, δημιουργώντας μια δομημένη και ευέλικτη αρχιτεκτονική πλοήγησης.

Για να διαμορφωθεί η δομή της πλοήγησης της εφαρμογής, χρησιμοποιούνται αρχεία με κατάληξη ".routes.ts". Στα αρχεία αυτά, ορίζονται τα URLs και τα components με τα οποία αντιστοιχίζονται.

Στο παράδειγμα της Κώδικα 2.18 , υπάρχει ένα αρχείο το οποίο περιέχει έναν πίνακα από αντικείμενα που περιέχουν την πληροφορία της δρομολόγησης. Τα αντικείμενα αυτά στην προκειμένη περίπτωση, αποτελούνται από το πεδίο "path" , το οποίο είναι το URL που ενεργοποιεί και φέρνει στο view το component με το οποίο έχει αντιστοιχηθεί στο πεδίο "component".

```
const routes: Routes = [
 { path: '', component: HomeComponent },
 { path: 'about', component: AboutComponent },
 { path: 'products', component: ProductsComponent];
```

Κώδικας 2.18 Παράδειγμα αρχείου routes.ts

Το route που ενεργοποιείται, ξεκινάει να κατοικεί στην θέση που καταλαμβάνει το tag <router-outlet> στον κώδικα HTML. Το <router-outlet> ουσιαστικά δρα ως placeholder για το component που ενεργοποιείται ανάλογα με την τρέχουσα κατάσταση της δρομολόγησης.

2.2.7 Guards

Τα guards είναι μια λειτουργία στο πλαίσιο Angular που χρησιμοποιείται για την προστασία διαδρομών (routes) από μη εξουσιοδοτημένους χρήστες. Μπορούν να χρησιμοποιηθούν για να επιτρέπουν ή να αποτρέπουν πρόσβαση σε μία συγκεκριμένη διαδρομή ανάλογα με τον ρόλο του χρήστη ή άλλες παραμέτρους.

Στην Angular υπάρχουν πέντε είδη guards:

1. **CanActivate**: Ελέγχει αν ο χρήστης έχει πρόσβαση στη διαδρομή.
2. **CanActivateChild**: Ελέγχει αν ο χρήστης έχει πρόσβαση στα παιδικά μονοπάτια της διαδρομής.
3. **CanDeactivate**: Ελέγχει αν ο χρήστης μπορεί να φύγει από τη διαδρομή.
4. **Resolve**: Φορτώνει δεδομένα πριν από την φόρτωση της διαδρομής .
5. **CanLoad**: Αποτρέπει την φόρτωση των lazy-loaded modules μέχρι να ικανοποιηθούν συγκεκριμένες προϋποθέσεις.

```
export const AuthenticationGuard: CanActivateFn = (
  route: ActivatedRouteSnapshot,
  state: RouterStateSnapshot
) => {
  const router = inject(Router);
  const authService = inject(AuthService);

  return authService.user$.pipe(
      map((user) => {
      if (!user) {
        return router.parseUrl('home');
      }
      return true;
    })
  );
};
```

Κώδικας 2.19 Παράδειγμα ενός guard αρχείου

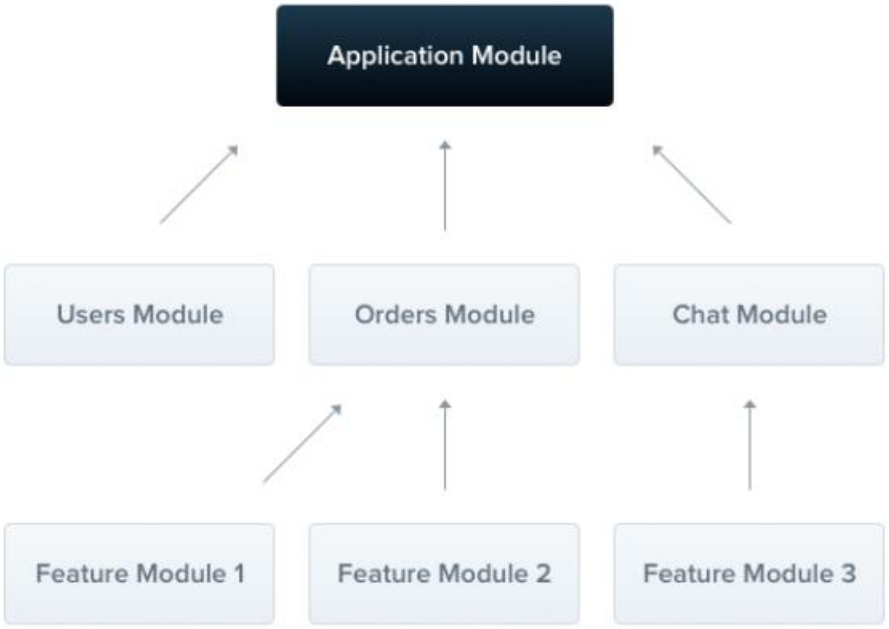
2.3 NestJS

Η Nestjs είναι ένα open source nodejs framework για την ανάπτυξη server-side εφαρμογών και διεπαφών προγραμματισμού εφαρμογών (API). Βασίζεται στην γλώσσα javascript και περιέχει πλήρη υποστήριξη της typescript, η οποία προσφέρει πολλά πλεονεκτήματα για την συγγραφή κώδικα όπως type safety των μεταβλητών και εντοπισμό σφαλμάτων. Αποτελεί μια ολοκληρωμένη λύση με ποικίλες δυνατότητες για την δημιουργία server side εφαρμογών και APIs. Επιπλέον χάρη στην σαφή δομή της διευκολύνει την κατανόηση του κώδικα επιτρέποντας μεγάλες ομάδες προγραμματιστών να αλληλοεπιδρούν ομαλότερα αλλά και η συντήρηση του κώδικα να γίνετε πιο αποτελεσματική, ειδικότερα σε μεγάλες και πολύπλοκες εφαρμογές.

Η αρχιτεκτονική της NestJS έχει επιρροές από μεγάλα frameworks, όπως της ExpressJS και της Angular. Συγκεκριμένα ακολουθεί ένα δομικό μοντέλο, χωρίζοντας τον κώδικα σε κομμάτια για καλύτερη οργάνωση και κατανομή. Επίσης συνδυάζει στοιχεία όπως αντικειμενοστραφής προγραμματισμός (Object Oriented Programming) και Συναρτησιακός Προγραμματισμός (Functional Programming) στα δομικά της στοιχεία.

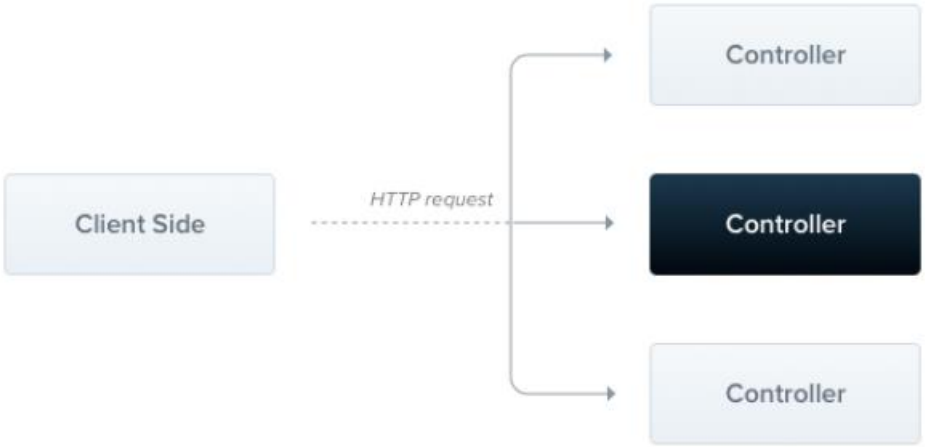
Ειδικότερα τα δομικά στοιχεία της NestJS περιλαμβάνουν :

1. **Modules**: Στην NestJS τα Modules αποτελούν τα θεμέλια κάθε πόρου της εφαρμογής. Στα modules ομαδοποιούνται και δηλώνονται όλες οι λειτουργίες που περιλαμβάνει κάθε πόρος, όπως για παράδειγμα controllers, services και providers. Επιπλέον τα Modules ακολουθούν μια ιεραρχική δομή και όλα έχουν ρίζα το αρχικό module της εφαρμογής.



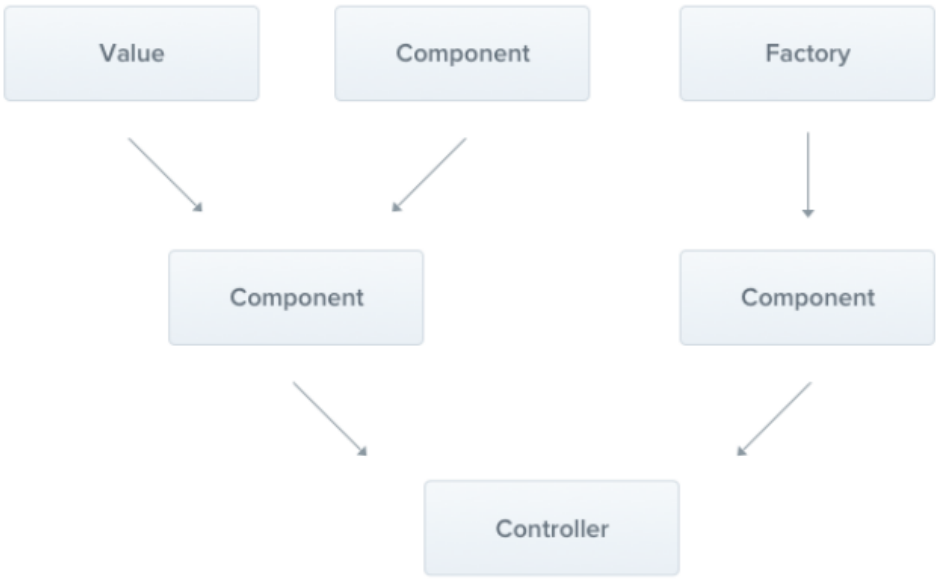
Εικόνα 2.4 Ιεραρχία modules στη NestJS

2. **Controllers**: Σκοπός των controllers στην NestJS είναι η ομαδοποίηση, η διαχείριση και ο καθορισμός των διαδρομών του API. Συνήθως αντιστοιχίζεται ένας controller ανά πόρο της εφαρμογής και περιλαμβάνει όλες τις διαδρομές προς αυτόν. Κάθε διαδρομή έπειτα καλεί να εκτελεστούν συγκεκριμένα κομμάτια κώδικα που αφορά την διαδρομή.



Εικόνα 2.5 Δομή controllers στη NestJS

3. **Providers**: Οι Providers είναι από τις βασικές έννοιες που ακολουθεί η NestJS. Σύμφωνα με την NestJS κάθε είδους κλάση μπορεί να θεωρηθεί σαν πάροχος (provider) ο οποίος μπορεί να εισαχθεί σε σημεία του κώδικα.



Εικόνα 2.6 Ιεραρχική δομή των providers στη NestJS

4. **Middlewares**: Τα middlewares είναι μέθοδοι κώδικα που εκτελούνται πριν από κάθε αίτηση σε κάποια διαδρομή (route). Έχουν πρόσβαση στο ερχόμενο request και εξερχόμενο response κάθε αίτησης και τους παρέχεται η δυνατότητα ανάγνωσης, προσθήκης και μεταβολής των πληροφοριών που περιέχουν, ενώ παράλληλα μπορούν να εκτελέσουν οποιοδήποτε τμήμα κώδικα. Επιπλέον τα Middlewares επιτρέπουν την αλυσίδωση των middleware (middleware chaining). Το Middleware chaining δίνει την δυνατότητα εκτέλεσης πολλών διαφορετικών middleware με αλυσιδωτή μορφή. Μετά από κάθε ολοκλήρωση του κώδικα που περιέχει ένα middleware, καλεί αυτομάτως την εκτέλεση του επόμενου κατά σειρά middleware που υπάρχει στην αλυσίδα. Αυτό επιτρέπει την οργάνωση των τμημάτων κώδικα αλλά και την αποφυγή ενός μεγάλου και πολύπλοκου middleware.

   Είναι μια προσιτή λύση αν υπάρχει ανάγκη για προεπεξεργασία των δεδομένων που εισέρχονται πριν την εκτέλεση του κώδικα της διαδρομής. Ένα παράδειγμα που καθιστά κατάλληλη λύση την χρήση των middleware είναι όταν υπάρχει ανάγκη μηχανισμού πιστοποίησης του πελάτη που έκανε αίτηση σε μια διαδρομή. Επεξεργάζονται τα δεδομένα που εισήλθαν και αν η ταυτοποίηση του πελάτη ήταν επιτυχής, επιτρέπεται η πρόσβαση στην διαδρομή.

Εικόνα 2.7 Ροή εκτέλεσης middleware στη NestJS

5. **Guards**: Τα guards είναι επίσης ένας μηχανισμός που εκτελείται πριν από κάθε αίτηση σε κάποια διαδρομή(route). Ειδικότερα εκτελούνται αφότου ολοκληρωθεί η εκτέλεση των middlewares και σκοπός τους είναι να προσδιορίσουν αν ο ταυτοποιημένος πελάτης μπορεί να έχει πρόσβαση στην αιτούμενη διαδρομή με βάση τις ιδιότητες του.

   Είναι μια προσιτή λύση αν υπάρχει ανάγκη καθορισμού πρόσβασης με βάση τα δικαιώματα ή με βάση τον ρόλο του κάθε χρήστη. Ένα παράδειγμα που καθιστά κατάλληλη λύση την χρήση των guards είναι όταν υπάρχει ανάγκη μηχανισμού εξουσιοδότησης του πελάτη που έκανε

αίτηση σε μια διαδρομή. Επεξεργάζονται τα δεδομένα του και αν η εξουσιοδότηση ήταν επιτυχής, επιτρέπεται η πρόσβαση στην διαδρομή.

Εικόνα 2.8 Ροή εκτέλεσης guards στη NestJS

2.3.1 Συστατικά

Η NestJS περιέχει πολλές έτοιμες λειτουργίες και εργαλεία για την διευκόλυνση των προγραμματιστών στην ανάπτυξη κώδικα. Παρακάτω γίνεται αναφορά και επεξήγηση σε κάποια από τα κύρια εργαλεία και λειτουργίες της NestJS , τα οποία χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής.

1. **Dependency Injection**: Το dependency injection είναι μια τεχνική που βασίζεται πολύ η δομή της NestJS. Συγκεκριμένα είναι μια τεχνική που χρησιμοποιείται στον αντικειμενοστραφή προγραμματισμό, η οποία επιτρέπει στην πρόσβαση μεθόδων και χαρακτηριστικών μιας κλάσης χωρίς απαραίτητα την δημιουργία αντικειμένου. Με αυτόν τον τρόπο επιτυγχάνεται η εύκολη επαναχρησιμοποίηση ιδιοτήτων μιας κλάσης.
2. **Decorators**: Η NestJS βασίζεται επίσης πολύ στους Decorators. Οι decorators είναι μέθοδοι που προσθέτουν λειτουργία σε κλάσης και μεθόδους με δηλωτικό τρόπο.Η NestJs προσφέρει μια μεγάλη συλλογή από decorators και δίνει την δυνατότητα δημιουργίας custom decorator ανάλογα τις απαιτούμενες ανάγκες.
3. **Exception handling**: Η NestJS έχει ενσωματωμένο ένα βασικό exception handling, για σφάλματα που μπορεί να προκύψουν κατά την διάρκεια που η εφαρμογή είναι ενεργή προβλέποντας ασυνήθιστα σφάλματα που μπορεί να σταματήσουν την ομαλή εκτέλεση της εφαρμογής αλλά και ενημερώνει με κατάλληλο http status code τον παραλήπτη.
4. **Data validation**: Ένα βασικό εργαλείο που προσφέρει η NestJS είναι το validation των δεδομένων που εισέρχονται στην εφαρμογή μέσω του ενσωματωμένου πακέτου class-validator. Με το εργαλείο αυτό το validation των δεδομένων γίνεται με απλό και αυτόματο τρόπο, μέσα από δηλωτικούς decorator που αναλαμβάνουν να ελέγξουν αν τα δεδομένα που απαιτούνται είναι σωστά.
5. **Swagger integration**: Το swagger είναι ένα δημοφιλές εργαλείο ανοιχτού κώδικα για την συγγραφή API documentation. Η NestJS έχει ενσωματώσει το swagger στο framework, έτσι ώστε η χρήση του να αντικατοπτρίζει την δομή συγγραφής της, με αποτέλεσμα η δημιουργία API documentation να γίνεται αυτόματα μέσα από decorators ειδικά σχεδιασμένους για το swagger.

2.3.2 NodeJS

Η NestJS είναι ένα framework το οποίο στηρίζεται στην NodeJS για να λειτουργήσει. Η NodeJS είναι ένα περιβάλλον εκτέλεσης (environment runtime) ανοιχτού κώδικα, το οποίο δίνει την δυνατότητα εκτέλεσης javascript κώδικα και σε server side περιβάλλον. Ειδικότερα η NodeJS έγινε ιδιαίτερα δημοφιλές διότι προσφέρει την επιλογή χρήσης μιας κοινής γλώσσας προγραμματισμού και σε frontend και σε backend περιβάλλοντα, διευκολύνοντας την ανάπτυξη εφαρμογών και την μακροχρόνια συντήρηση τους. Επιπλέον η NodeJS περιλαμβάνεται από ένα μεγάλο πλήθος διαθέσιμων βιβλιοθηκών

για κάθε είδους ανάγκη αλλά και από ένα ενεργό οικοσύστημα, καθιστώντας ένα βασικό εργαλείο για πολλούς προγραμματιστές.

2.3.3 Βιβλιοθήκες

Καθόλη την διάρκεια ανάπτυξης της εφαρμογής, έγινε η χρήση έτοιμων βιβλιοθηκών και εργαλείων για την επίτευξη αναγκαίων και απαραίτητων λειτουργιών. Ειδικότερα οι βιβλιοθήκες είναι έτοιμα κομμάτια κώδικα όπου προσφέρουν έτοιμα εργαλεία για μια ή περισσότερες λειτουργίες. Διευκολύνουν και επιταχύνουν την ανάπτυξη μιας εφαρμογής ιδίως σε περίπλοκες λειτουργίες που απαιτούν εξειδικευμένη γνώση για την υλοποίηση τους. Μπορούν να εγκατασταθούν με την χρήση ενός διαχειριστής πακέτων (Package Manager), ο οποίος περιλαμβάνει μια μεγάλη συλλογή από βιβλιοθήκες ανοιχτού κώδικα που μπορούν να χρησιμοποιηθούν σε μια εφαρμογή. Επιπλέον υπάρχουν διάφοροι διαθέσιμοι διαχειριστές πακέτων ανάλογα το περιβάλλον ανάπτυξης και την γλώσσα προγραμματισμού που χρησιμοποιείται στην εφαρμογή. Ειδικότερα στην nodeJS ο πιο δημοφιλής διαχειριστής πακέτων, ο οποίος χρησιμοποιήθηκε και στην συγκεκριμένη εφαρμογή, είναι ο npm (Node Package Manager). Ο npm αναλαμβάνει κύριός την διαχείριση των βιβλιοθηκών και παρέχει πρόσβαση και δυνατότητα εγκατάστασης πολλών και διαφόρων βιβλιοθηκών.

2.3.3.1 Mysql2

Η nodeJS δεν περιέχει άμεση υποστήριξη για την επικοινωνία και αλληλεπίδραση μεταξύ της βάσης δεδομένων και της εφαρμογής, παραπέμποντας υποχρεωτικά στην εξερεύνηση έτοιμων βιβλιοθηκών που αναλαμβάνουν την δημιουργία της συγκεκριμένης λειτουργίας. Έπειτα από έρευνα και ανάλυση των διαθέσιμων βιβλιοθηκών, έγινε επιλογή της βιβλιοθήκης mysql2. Η συγκεκριμένη βιβλιοθήκη προσφέρει πολλές λειτουργίες και στοχεύει στην αντικατάσταση της επίσημης βιβλιοθήκης της mysql. Κάποιες από τις βασικές λειτουργίες που προσφέρει περιλαμβάνουν:

- Δημιουργία σύνδεσης με την mysql βάση δεδομένων.
- Εκτέλεση ασύγχρονων ερωτημάτων προς την βάση.
- Προετοιμασμένες δηλώσεις (prepare statements)

2.3.3.2 JSZip

Για την επίτευξη της λειτουργίας εξαγωγής και εισαγωγής υποβολών σε συμπιεσμένα αρχεία έγινε η χρήση του πακέτου JSZip. Η βιβλιοθήκη JSZip προσφέρει έτοιμες μεθόδους, οι οποίες μπορούν να χρησιμοποιηθούν για την συμπίεση ενός ή περισσοτέρων αρχείων και φακέλων. Επιπλέον προσφέρει μια μεγάλη ποικιλία επιλογών ως προς τον τρόπο συμπίεσης των αρχείων αλλά και για τον βαθμό συμπίεσης τους.

Κεφάλαιο 3ο: Ανάπτυξη εφαρμογής

3.1 Εισαγωγή

Η διαδικτυακή εφαρμογή που αναπτύχθηκε έχει ως βασικό στόχο την υποστήριξη της διαδικασίας υποβολής ασκήσεων σε εκπαιδευτικό περιβάλλον. Η υλοποίηση βασίζεται στο σύγχρονο Angular framework, το οποίο προσφέρει τη δυνατότητα δημιουργίας δυναμικών και επεκτάσιμων εφαρμογών τύπου Single Page Application (SPA).

Κατά την είσοδο του χρήστη στην πλατφόρμα, πραγματοποιείται αρχικά η αυθεντικοποίηση του μέσω συστήματος σύνδεσης που παρέχεται από υπάρχον σύστημα της σχολής. Μόλις ολοκληρωθεί με επιτυχία η ταυτοποίηση, ελέγχεται ο ρόλος που του έχει αποδοθεί από το σύστημα. Ο χρήστης μπορεί να έχει είτε τον ρόλο του φοιτητή είτε του καθηγητή, και με βάση αυτόν τον ρόλο ανακατευθύνεται στο κατάλληλο περιβάλλον εντός της εφαρμογής.

Για τους χρήστες με ρόλο φοιτητή, η πλατφόρμα παρέχει ένα απλό και λειτουργικό περιβάλλον μέσω του οποίου μπορούν να επιλέξουν το μάθημα για το οποίο θέλουν να υποβάλουν εργασία. Στη διάθεσή τους υπάρχει σχετική φόρμα υποβολής, όπου καταχωρούν τα απαραίτητα στοιχεία και προχωρούν στην αποστολή της εργασίας τους.

Αντιθέτως, οι χρήστες με ρόλο καθηγητή οδηγούνται σε ένα περιβάλλον διαχείρισης, το οποίο παρέχει προηγμένες λειτουργίες για την επίβλεψη και τη διαχείριση τόσο των μαθημάτων όσο και των υποβολών των φοιτητών. Οι δυνατότητες του διαχειριστή θα παρουσιαστούν αναλυτικά στα επόμενα τμήματα της εργασίας. Το διαχειριστικό περιβάλλον έχει σχεδιαστεί με τρόπο ώστε να διευκολύνει τη χρήση και να προσφέρει άμεση εποπτεία όλων των σχετικών λειτουργιών.

Ως backend εργαλείο επιλέχθηκε το NestJS. Το NestJS είναι ένα σύγχρονο εργαλείο για τη δημιουργία εφαρμογών που «τρέχουν» στην πλευρά του διακομιστή (backend). Έχει σχεδιαστεί έτσι ώστε να διευκολύνει την οργανωμένη και καθαρή ανάπτυξη κώδικα, κάτι που είναι ιδιαίτερα σημαντικό όταν μια εφαρμογή μεγαλώνει και γίνεται πιο σύνθετη. Βασίζεται σε αρχές της παραδοσιακής μηχανικής λογισμικού, όπως η οργάνωση σε δομικές ενότητες (modules), η ευκολία στην επαναχρησιμοποίηση και η σαφής διαχείριση των dependencies. Παράλληλα, προσφέρει μια εμπειρία ανάπτυξης που θυμίζει άλλα σύγχρονα εργαλεία, όπως το Angular, κάνοντας ευκολότερη την εκμάθηση και τη συνεργασία μεταξύ προγραμματιστών frontend και backend. Είναι ιδανικό για την κατασκευή διαδικτυακών υπηρεσιών, APIs και κάθε είδους backend εφαρμογών.

Συνοπτικά, μπορούμε να πούμε πως η ανάπτυξη της παρούσας εφαρμογής ανέδειξε τη σημασία της σωστής αρχιτεκτονικής σχεδίασης, της ορθής διασύνδεσης των τεχνολογιών και της αποτελεσματικής χρήσης σύγχρονων εργαλείων λογισμικού. Μέσα από τη χρήση των frameworks Angular και NestJS, επιτεύχθηκε η δημιουργία ενός λειτουργικού και ευέλικτου περιβάλλοντος, τόσο για τους φοιτητές όσο και για τους καθηγητές, καλύπτοντας τις ανάγκες μιας σύγχρονης ψηφιακής πλατφόρμας εκπαιδευτικής υποβολής.

3.2 Σχεδιασμός και υλοποίηση της βάσης δεδομένων

Στο αρχικό στάδιο σχεδιασμού της εφαρμογής, δώσαμε ιδιαίτερη έμφαση στη δημιουργία μιας σταθερής και αποτελεσματικής βάσης δεδομένων. Θεωρήσαμε πως μια σωστά δομημένη βάση αποτελεί

το θεμέλιο πάνω στο οποίο μπορεί να χτιστεί ολόκληρο το σύστημα, εξασφαλίζοντας την αξιόπιστη αποθήκευση, εύκολη ανάκτηση και συνεχή ενημέρωση των δεδομένων. Παράλληλα, η σωστή σχεδίαση της βάσης διευκολύνει και τη μελλοντική εξέλιξη και επέκταση της εφαρμογής.Μια σχεδίαση βάσης δεδομένων που δεν λαμβάνει υπόψιν μελλοντικές απαιτήσεις και δεν εχει σωστή δόμηση θα μπορούσε να προκαλέσει πολλαπλά προβλήματα.

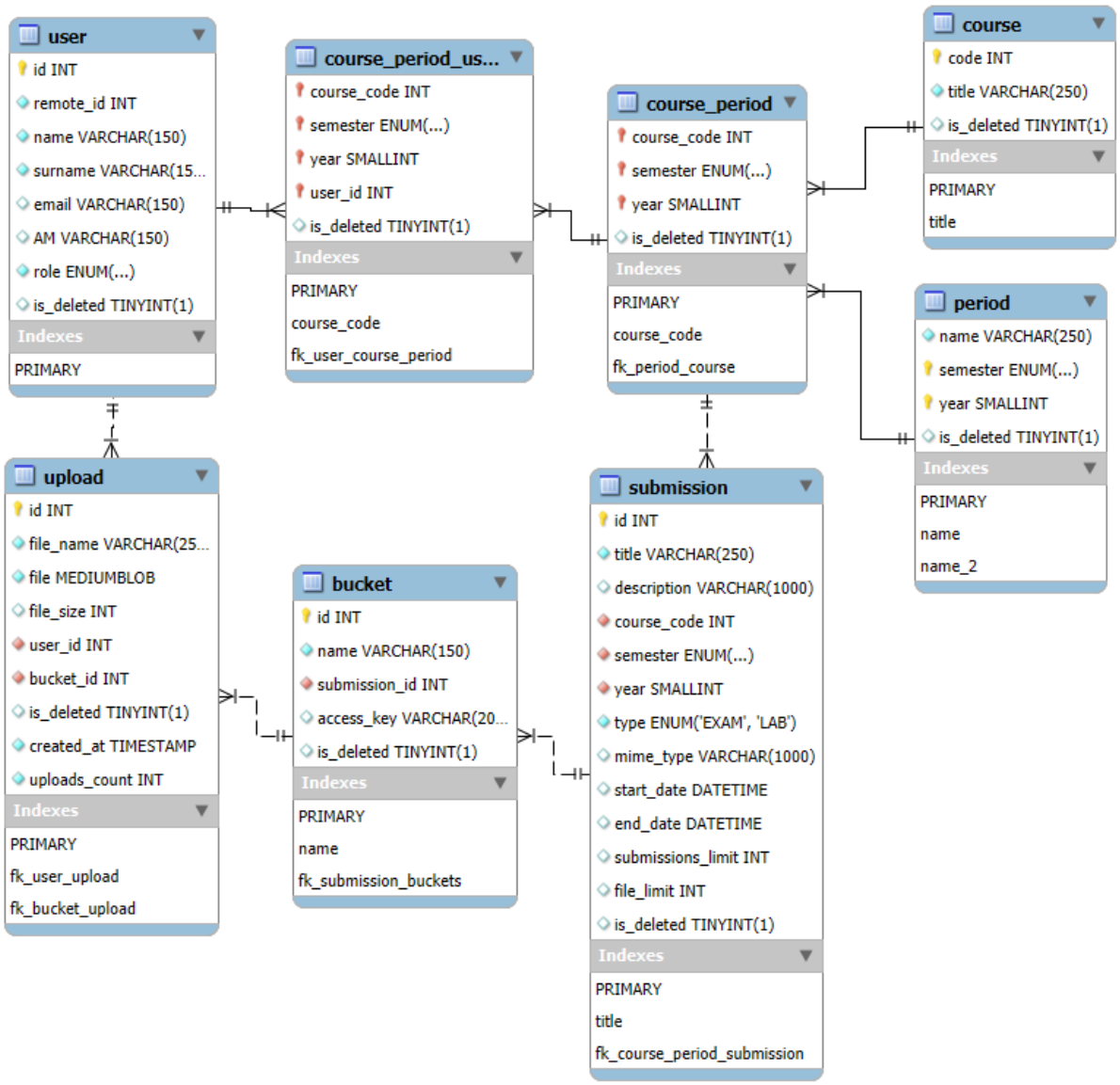
Ειδικότερα προβλήματα που μπορεί να εμφανιστούν περιλαμβάνουν τα εξής:

1. **Ταχύτητα**: Η ταχύτητα είναι ένα μείζον πρόβλημα που μπορεί να προκληθεί από μια απρόσεκτη σχεδίαση βάσης δεδομένων. Τα ερωτήματα προς την βάση θα πρέπει να εκτελούνται με τον πιο αποτελεσματικό και βέλτιστο τρόπο και επιπλέον με την χρήση λίγων πόρων του συστήματος.
2. **Ανάπτυξη εφαρμογής**: Επιπλέον εμποδίζει την ομαλή και γρήγορη ανάπτυξη της εφαρμογής. Αυτό συμβαίνει διότι αν δεν έχει ληφθεί υπόψη η μακροχρόνια ανάπτυξη και επέκταση της, μεταγενέστερα η προσθήκη νέων χαρακτηριστικών και νέων ιδιοτήτων είναι αισθητά χρονοβόρα και δύσκολη στην υλοποίηση της.
3. **Αναγνωσιμότητα**: Η αναγνωσιμότητα των ερωτημάτων είναι βασικός παράγοντας της ομαλής αλληλεπίδρασης μεταξύ των προγραμματιστών. Μια απρόσεκτη σχεδίαση βάσης δεδομένων μπορεί να επιφέρει την ανάγκη δημιουργίας σύνθετων ερωτημάτων, ακόμη και για απλά εννοιολογικά ερωτήματα. Αυτό έχει ως αποτέλεσμα την δυσνόητη κατανόηση των ερωτημάτων που εκτελούνται στην εφαρμογή, ειδικότερα σε πολύπλοκα ερωτήματα. Ως εκ τούτου δημιουργεί προβλήματα στην ομαλή αλληλεπίδραση των ατόμων που περιβάλλουν την εφαρμογή και την καθυστερεί την ανάπτυξη της.

Με βάση τις απαιτήσεις της εφαρμογής, θεωρήθηκε κατάλληλη επιλογή η χρήση μιας σχεσιακής βάσης δεδομένων για την αποθήκευση και διαχείριση των δεδομένων. Η σχεσιακή βάση δεδομένων είναι ένα σύνολο δεδομένων οργανωμένα σε πίνακες με γραμμές και στήλες, όπου ο πίνακας αντιπροσωπεύει μια οντότητα, κάθε στήλη αντιπροσωπεύει μια ιδιότητα/χαρακτηριστικό του πίνακα και κάθε γραμμή αντιπροσωπεύει μια εγγραφή πίνακα. Επιπλέον κάθε πίνακας ακολουθεί συγκεκριμένους κανόνες ως προς την δομή των εγγραφών και παράλληλα έχει την δυνατότητα να δημιουργεί συσχετίσεις/σχέσεις μεταξύ άλλων πινάκων της βάσης. Επίσης χρησιμοποιεί την γλώσσα SQL για την δημιουργία και εκτέλεση ερωτημάτων προς τα δεδομένα, προσφέροντας δυνατότητες δημιουργίας, μεταβολής και διαγραφής των δεδομένων, των πινάκων ή των σχέσεων. Την καθιστά κατάλληλη επιλογή για αποθήκευση δεδομένων με ακριβής δομή και οργάνωση. Ειδικότερα επιλέχθηκε η χρήση της Mysql relational database διότι προσφέρει απλότητα και ταχύτητα.

3.2.1 Δομή και σχεδίαση

Παρακάτω παρουσιάζεται αναλυτικά η δομή και η σχεδίαση της βάσης δεδομένων που δημιουργήθηκε για τις ανάγκες της εφαρμογής. Αναφέρονται οι πίνακες που δημιουργήθηκαν, τα πεδία που ήταν απαραίτητα για κάθε πίνακα, οι σχέσεις μεταξύ τους και οι κανόνες που πρέπει να ακολουθούν.



Εικόνα 3.1 Schema βάσης

3.2.1.1 Δομή Πινάκων

Κατά την σχεδίαση της βάσης δεδομένων, ελήφθη με μεγάλη σοβαρότητα η δόμηση των πινάκων. Η βάση δεδομένων είναι τα θεμέλια μιας εφαρμογής και μια απρόσεκτη δόμηση των πινάκων, θα μπορούσε να προκαλέσει πολλαπλά προβλήματα κατά την πορεία ανάπτυξης της.

Συγκεκριμένα οι πίνακες/οντότητες που περιλαμβάνονται στην βάση είναι οι εξής:

1. **user**: Ο πίνακας user (χρήστης) αναλαμβάνει την αποθήκευση των χρηστών της εφαρμογής και των χαρακτηριστικών τους.

   Περιλαμβάνει παρακάτω πεδία:

   - *id (κύριο κλειδί)*: Το κύριο αναγνωριστικό του χρήστη.
   - *name*: Το όνομα του χρήστη.
   - *surname*: Το επίθετο του χρήστη.
   - *email*: Το mail του χρήστη.
   - *AM*: Αριθμός μητρώου του χρήστη.

- *role*: Ο ρόλος του χρήστη. Οι χρήστες διαφοροποιούνται σε 2 ρόλους, τους φοιτητές και τους καθηγητές.
- *remore_id*: Το αναγνωριστικό του χρήστη με βάση τα αποτελέσματα της απομακρυσμένης διαδρομής για την ανάκτηση των πληροφοριών του.
- *is_deleted*: αναγνωριστικό εικονικής διαγραφής.

2. **course**: Ο πίνακας course (μάθημα) αναλαμβάνει την αποθήκευση τον τίτλων μαθημάτων του προγράμματος σπουδών της σχολής.

   Περιλαμβάνει τα παρακάτω πεδία:

   - *code (κύριο κλειδι)*: Κωδικός μαθήματος.
   - *title*: Τίτλος μαθήματος.
   - *is_deleted*: αναγνωριστικό εικονικής διαγραφής.

3. **period**: Ο πίνακας period (περίοδος) αναλαμβάνει την αποθήκευση των ετήσιων περιόδων που περιλαμβάνει κάθε χρονιά. Ο συγκεκριμένος πίνακας έχει δημιουργηθεί με την ύπαρξη δύο κύριων κλειδιών. Θεωρήθηκε περιττή η δημιουργία ενός αναγνωριστικού εγγραφής διότι ο συνδυασμός των δύο πεδίων semester (εξάμηνο) και year (χρόνος) μπορεί να περιγράψει την μοναδικότητα κάθε εγγραφής.

   Περιλαμβάνει τα παρακάτω πεδία:

   - *name*: Όνομα περιόδου. Μπορεί να προσδιοριστεί με οποιονδήποτε τρόπο χωρίς κάποιον περιορισμό.
   - *semester (κύριο κλειδί)*: Το εξάμηνο της περιόδου. Επιτρέπονται οι τιμές ‘winter’ για το χειμερινό εξάμηνο και ‘spring’ για το εαρινό εξάμηνο.
   - *year (κύριο κλειδί)*: Η χρονιά που περιβάλλεται η περίοδος.

4. **submission**: Ο Submission (άσκηση) πίνακας αναλαμβάνει την αποθήκευση των ασκήσεων που μπορεί να υπάρχουν σε ένα μάθημα μιας συγκεκριμένης περιόδου. Οι ασκήσεις χωρίζονται σε ασκήσεις ‘lab’ (εργαστηρίου) και σε ασκήσεις ‘exam’ (εξέτασης).

   Περιλαμβάνει τα παρακάτω πεδία:

   - *id (κύριο κλειδί)*: Αναγνωριστικό άσκησης.
   - *title*: Τίτλος/Ονομασία άσκησης.
   - *description*: Περιγραφή άσκησης. Μπορεί να προσδιοριστεί με οποιονδήποτε τρόπο με τον περιορισμό να είναι μικρότερο από χίλιους χαρακτήρες.
   - *course_code*: Το μάθημα στο οποίο ανήκει η άσκηση.
   - *semester*: Το εξάμηνο στο οποίο ανήκει η άσκηση.
   - *year*: Η χρονική περίοδος που ανήκει η άσκηση.
   - *type*: Ο Τύπος άσκησης. Επιτρέπονται οι τιμές ‘lab’ για άσκηση εργαστηρίου και ‘exam’ για άσκηση εξεταστικής.
   - *mime_type*: Το πεδίο mime_type, περιλαμβάνει όλους τους αποδεκτούς τύπους αρχείων που επιτρέπει μια άσκηση. Αν δεν καταχωρηθεί κάποια τιμή στο πεδίο, τότε αυτομάτως θεωρούνται αποδεκτοί όλοι οι τύποι αρχείων.
   - *start_date*: Το πεδίο start_date, περιέχει την ημερομηνία έναρξης της άσκησης, δηλαδή την ημερομηνία από την οποία η άσκηση επιτρέπει την υποβολή αρχείων.
   - *end_date*: Αναλόγως το πεδίο end_date, περιέχει την ημερομηνία λήξης της άσκησης, δηλαδή την ημερομηνία από την οποία η άσκηση δεν θα επιτρέπει την υποβολή αρχείων.

- *submission_limit*: Το πεδίο submission_limit περιλαμβάνει τον αριθμό μέγιστων επαναλαμβανόμενων υποβολών που μπορούν να γίνουν από έναν χρήστη. Επιτρεπόμενες τιμές είναι ακέραιοι αριθμοί ενώ αν δεν καταχωρηθεί κάποια τιμή στο πεδίο, αυτομάτως είναι αποδεκτές άπειρες υποβολές απο τον χρήστη.
- *file_limit*: Το πεδίο file_limit περιλαμβάνει τον μέγιστο επιτρεπόμενο αριθμό αρχείων που μπορούν να γίνουν σε μια υποβολή. Αν δεν καταχωρηθεί κάποια τιμή στο πεδίο, αυτομάτως το πεδίο εκχωρεί την τιμη 1 ως μέγιστο αριθμό αρχείων.
- *is_deleted*: Αναγνωριστικό εικονικής διαγραφής.

5. **bucket**: Ο bucket πίνακας δημιουργήθηκε σαν μια γενική έννοια στην βάση δεδομένων. Αυτό που αναλαμβάνει να αποθηκεύσει και να περιγράψει είναι η κατανομή κάθε τμήματος ενός η περισσοτέρων καθηγητών σε μια άσκηση. Συγκεκριμένα κάθε άσκηση, μπορεί να είναι κατανεμημένη σε πολλά διάφορα τμήματα φοιτητών και κάθε καθηγητής μπορεί να αναλαμβάνει ένα ή περισσότερα από αυτά τα τμήματα. Το bucket ως γενική έννοια είναι οι υποβολές που υπάρχουν ανά τμήμα και κατανέμονται αναλόγως τον καθηγητή που τα δημιούργησε. Είναι ο συνδετικός κρίκος μεταξύ υποβολών ανά τμήμα και ασκήσεων.

   Περιλαμβάνει τα παρακάτω πεδία:

   - *id (κύριο κλειδι)*: Αναγνωριστικό bucket.
   - *name*: Τίτλος bucket. Μπορεί να προσδιοριστεί με οποιονδήποτε τρόπο.
   - *submission_id*: Η άσκηση στην οποία ανήκει το bucket.
   - *access_key*: Το πεδίο access_key περιγράφει τον κωδικό που μπορεί να περιέχει ένα bucket για μια άσκηση. Δεν είναι απαιτούμενο να υπάρχει ένας κωδικός για κάποιο bucket. Αν δεν καταχωρηθεί κάποια τιμή, αυτομάτως θεωρείται ότι είναι ανοιχτό προς όλους.
   - *is_deleted*: Αναγνωριστικό εικονικής διαγραφής.
6. **upload**: Ο πίνακας upload (υποβολή) αναλαμβάνει την αποθήκευση τις πληροφορίες των υποβολών που έγιναν ανά χρήστη σε ένα bucket μιας συγκεκριμένης άσκησης.

   Περιλαμβάνει τα παρακάτω πεδία:

   - *id (κύριο κλειδί)*: Αναγνωριστικό υποβολής.
   - *file_name*: Όνομα αρχείου/αρχείων που υποβλήθηκαν.
   - *file*: Τα αρχεία που υποβλήθηκαν, αποθηκεύονται σε μορφή medium blob.
   - *file_size*: Συνολικό μέγεθος αρχείων που υποβλήθηκαν.
   - *user_id*: Αναγνωριστικό χρήστη που έκανε την υποβολή.
   - *bucket_id*: Το αναγνωριστικό του bucket στο οποίο έγινε μια υποβολή.
   - *created_at*: Ημερομηνία και ώρα δημιουργίας της υποβολής.
   - *uploads_count*: Αριθμός επανυποβολής.

Συνεχίζοντας υπάρχουν οι πίνακες σχέσεων. Οι πίνακες σχέσεων είναι πίνακες οι οποίοι βασίζονται κυρίως στην αποθήκευση αναγνωριστικών 2 διαφορετικών πινάκων. Το πλεονέκτημα που προσφέρουν οι πίνακες σχέσεων είναι η δημιουργία μιας ειδικότερης πληροφορίας ανάμεσα στους 2 πίνακες αλλά και στην αποφυγή άσκοπης επανάληψης πληροφορίας μέσα στην βάση δεδομένων.

Ειδικότερα οι πίνακες σχέσεων που συμπεριλαμβάνει η βάση δεδομένων είναι οι εξής:

1. **course_period**: Ο πίνακας course_period αναλαμβάνει την αποθήκευση της σχέσης μεταξύ των πινάκων course (μάθημα) και period (χρονική περίοδος). Στον πίνακα αποθηκεύονται οι αναφορές στα ξένα κλειδιά των πινάκων όπου ως αποτέλεσμα έχει την παραγωγή νέας

πληροφορίας ανάμεσα τους. Ειδικότερα αντιστοιχεί κάθε course (μάθημα) σε μία ή περισσότερες period (χρονική περίοδος) αλλά και το αντίστροφο, αντιστοιχεί κάθε period (χρονική περίοδος) σε ένα ή περισσότερα course (μάθημα). Επιπλέον ο πίνακας δημιουργήθηκε με την ύπαρξη τριών κύριων κλειδιών τα οποία μπορούν να αναδείξουν την μοναδικότητα μιας εγγραφής χωρίς την απαραίτητη προσθήκη νέου πεδίου αναγνώρισης (id).

Περιλαμβάνει τα παρακάτω πεδία:

- *course_code (κύριο κλειδί):* Αναφορά ξένου κλειδιού του πίνακα course (μάθημα).
- *semester (κύριο κλειδί)*: Αναφορά ξένου κλειδιού του πίνακα period (περίοδος).
- *year (κύριο κλειδί)*: Αναφορά δεύτερου συνδυαστικού ξένου κλειδιού του πίνακα period (περίοδος).

2. **course_period_user**: Ο πίνακας course_period_user αναλαμβάνει την αποθήκευση της σχέσης μεταξύ του χρήστη (user) και των μαθημάτων περιόδου (course period). Στον πίνακα αποθηκεύονται τα ξένα κλειδιά των πινάκων user (χρήστη) και course period (μαθημάτων περιόδου) όπου ως αποτέλεσμα έχει την παραγωγή νέας πληροφορίας στην βάση. Συγκεκριμένα αντιστοιχεί κάθε user (χρήστη) σε ένα ή περισσότερα course period (μάθημα περιόδου) αλλά και κάθε course period (μάθημα περιόδου) σε έναν ή περισσότερους user (χρήστης). Επιπρόσθετα ο πίνακας δημιουργήθηκε με την ύπαρξη τεσσάρων κύριων κλειδιών ως μοναδικό αναγνωριστικό για κάθε εγγραφής τα οποία μπορούν να αναδείξουν την μοναδικότητα κάθε εγγραφής χωρίς να είναι απαραίτητη η προσθήκη νέου πεδίου αναγνώρισης (id).

   Περιλαμβάνει τα παρακάτω πεδία:

   - *course_code (κύριο κλειδί)*: Αναφορά ξένου κλειδιού του πίνακα course_period (μάθημα περιόδου).
   - *semester (κύριο κλειδί)*: Αναφορά ξένου κλειδιού του πίνακα course_period (μάθημα περιόδου).
   - *year (κύριο κλειδί)*: Αναφορά ξένου κλειδιού του πίνακα course_period (μάθημα περιόδου).
   - *user_id (κύριο κλειδί)*: Αναφορά ξένου κλειδιού του πίνακα user (χρήστη).

3.2.1.2 Συσχετίσεις πινάκων

Ένα μεγάλο πλεονέκτημα των σχεσιακών βάσεων δεδομένων είναι η δυνατότητα δημιουργίας συσχετίσεων/σχέσεων μεταξύ των πινάκων. Οι σχέσεις είναι ένα είδος σύνδεσης μεταξύ των πινάκων, οι οποίες καθορίζουν τον τρόπο με τον οποίο είναι συνδεδεμένες οι εγγραφές 2 πινάκων.

Τα κύρια πλεονεκτήματα που προσφέρει η ύπαρξη σχέσεων μεταξύ των πινάκων μιας βάσης δεδομένων είναι τα εξής:

- ➢ **Ακεραιότητα αναφορών**: Οι συνδέσεις μεταξύ πινάκων προσδιορίζουν συγκεκριμένους κανόνες μεταξύ των πινάκων, δημιουργώντας αλληλεξάρτηση μεταξύ των δεδομένων των πινάκων. Επομένως παράγεται ακρίβεια και αξιοπιστία μεταξύ των δεδομένων και των συνδέσεων τους.
- ➢ **Πολυπλοκότητα ερωτημάτων**: Η ύπαρξη συνδέσεων μεταξύ πινάκων, διευκολύνει αισθητά την πολυπλοκότητα και την ταχύτητα εκτέλεσης ερωτημάτων.

➢ **Κανονικοποίηση**: Με την υλοποίηση σχέσεων επιτυγχάνεται η έννοια της κανονικοποίησης. Η κανονικοποίηση είναι η μετατροπή μεγάλων σε πληροφορία πινάκων, σε μικρότερους, καλύτερα οργανωμένους πίνακες.

3.2.1.2.1 Τύποι σχέσεων

Οι σχέσεις πινάκων κατανέμονται σε τύπους ανάλογα με το είδος σχέσεις που είναι απαραίτητη να γίνει μεταξύ δύο πινάκων.

Συγκεκριμένα διαφοροποιούνται στους εξής τύπους:

1. **Ένα-Προς-Ένα (1:1)**: Ο τύπος σχέσης ένα-προς-ένα προσδιορίζει κάθε μια εγγραφή ενός πίνακα Α να αναλογεί σε μία και μόνο μια εγγραφή ενός άλλου πίνακα Β.
2. **Ένα-Προς-Πολλά (1:M)**: Ο τύπος σχέσης ένα-προς-πολλά προσδιορίζει κάθε μια εγγραφή ενός πίνακα Α να αναλογεί σε καμία ή περισσότερες εγγραφές ενός άλλου πίνακα Β.
3. **Πολλά-Προς-Ένα (M:1)**: Η αντιστροφή έννοια του τύπου σχέσης Ένα-προς-Πολλά. Κάθε εγγραφή ενός πίνακα Β επιτρέπεται να αναλογεί σε μία και μόνο μια εγγραφή του πίνακα Α.
4. **Πολλά-Προς-Πολλά (M:M)**: Στον τύπο σχέσης πολλά-προς-πολλά, κάθε εγγραφή ενός πίνακα Α αναλογεί καμία ή πολλές εγγραφές ενός άλλου πίνακα Β. Αντίθετα και ο πίνακας Β μπορεί να αναλογεί σε καμία ή πολλές εγγραφές του πίνακα Α.

3.2.1.3 Συσχετίσεις Βάσης

Καθόλη την διάρκεια σχεδίασης της βάσης δεδομένων, δόθηκε μεγάλη έμφαση στην ορθή δόμηση και δημιουργία σχέσεων μεταξύ των πινάκων. Ήταν σημαντικό να δημιουργηθούν άρτιες εννοιολογικές σχέσεις ώστε να επιτευχθεί η σωστή σημασιολογική δομή ανάμεσα στα δεδομένα, χωρίς άσκοπες και λανθασμένες σχέσεις.

3.2.1.3.1 Πίνακας user

- Σχέση **πολλά-προς-πολλά (M:M)** με πίνακα course_period_user
    - Μια εγγραφή user (χρήστη) μπορεί να έχει κανένα ή περισσότερα course_period (μαθήματα περιόδου). Δημιουργήθηκε για την αναπαράσταση των μαθημάτων περιόδου που έχει κάθε μια εγγραφή χρήστη ρόλου (role πεδίο στον πίνακα user) ‘professor’. Σημασιολογικά κάθε καθηγητής μπορεί να έχει κανένα ή περισσότερα μαθήματα περιόδου.
    - Δημιουργία αναφοράς κύριου κλειδιού id του πίνακα user με ξένο κλειδί user_id του πίνακα course_period_user
- Σχέση **ένα-προς-πολλά (1:M)** με πίνακα upload
    - Μια εγγραφή user (χρήστη) μπορεί να έχει καμία ή περισσότερες upload (υποβολή). Αντίθετα κάθε upload (υποβολή) μπορεί να έχει έναν και μόνο έναν user (χρήστη). Δημιουργήθηκε για την αναπαράσταση των υποβολών που έχει κάθε μια εγγραφή χρήστη ρόλου (role πεδίο στον πίνακα user) ‘student’. Σημασιολογικά κάθε φοιτητής μπορεί να έχει καμία ή περισσότερες υποβολές και κάθε υποβολή μπορεί να έχει έναν χρηστή.
    - Δημιουργία αναφοράς κύριου κλειδιού id του πίνακα user (χρήστης) με ξένο κλειδί user_id του πίνακα course_period_user.

3.2.1.3.2 Πίνακας course_period_user

Ο πίνακας course_period_user είναι ένας ενδιάμεσος πίνακας ο οποίος αναλαμβάνει την δημιουργία σχέσης πολλά-προς-πολλά μεταξύ των πινάκων user (χρήστη) και course_period (μάθημα περιόδου). Εννοιολογικά κάθε χρήστης μπορεί να έχει πολλά μαθήματα και αντίθετα πολλά μαθήματα μπορούν να έχουν πολλούς χρήστες. Ειδικότερα για την επίτευξη της σχέσης αυτής δημιουργούνται οι παρακάτω σχέσεις μεταξύ των δύο πινάκων:

- Σχέση **πολλά-προς-ένα (M:1)** με πίνακα user
    - Μια εγγραφή course_period_user μπορεί να έχει σε έναν και μόνον έναν user(χρήστη). Αντίθετα κάθε user (χρήστης) μπορεί να έχει κανένα ή περισσότερα course_period_user.
- Σχέση **πολλά-προς-ένα (M:1)** με πίνακα course_period
    - Μια εγγραφή course_period_user μπορεί να έχει έναν και μόνο έναν user (χρήστη). Αντίθετα κάθε course_period (μάθημα περιόδου) μπορεί να έχει κανένα ή περισσότερα course_period_user.

3.2.1.3.3 Πίνακας course

- Σχέση **ένα-προς-πολλά (1:M)** με πίνακα course_period
    - Μια εγγραφή course (μάθημα) μπορεί να έχει κανένα ή περισσότερα course_period (μαθήμα περιόδου). Αντίθετα κάθε course_period μπορεί να έχει ένα και μόνο ένα course (μάθημα). Σημασιολογικά κάθε μάθημα μπορεί να υπάρχει και να έχει πολλές περιόδους. Για παράδειγμα ‘Δομημένος προγραμματισμός-χειμερινό εξάμηνο-2025’, Δομημένος προγραμματισμός-εαρινό-εξάμηνο-2025’.

3.2.1.3.4 Πίνακας period

- Σχέση **ένα-προς-πολλά (1:M)** με πίνακα course_period
    - Μια εγγραφή period (περίοδος) μπορεί να έχει κανένα ή περισσότερα course_period (μάθημα περιόδου). Αντίθετα κάθε course_period μπορεί να έχει ένα και μόνο ένα period (περίοδος). Σημασιολογικά κάθε περίοδος μπορεί να υπάρχει και να έχει πολλά μαθήματα. Για παράδειγμα ‘Δομημένος προγραμματισμός-χειμερινό εξάμηνο-2025’, Δομημένος προγραμματισμός-εαρινό-εξάμηνο-2025’.

3.2.1.3.5 Πίνακας course_period

Ο πίνακας course_period είναι επίσης ένας ενδιάμεσος πίνακας ο οποίος αναλαμβάνει την δημιουργία σχέσης πολλά-προς-πολλά μεταξύ των πινάκων course (μάθημα) και period (περίοδος).Εννοιολογικά κάθε μάθημα μπορεί να έχει πολλές περιόδους και αντίθετα κάθε περίοδος μπορεί να έχει πολλά μαθήματα. Ειδικότερα για την επίτευξη της σχέσης αυτής δημιουργούνται οι παρακάτω σχέσεις μεταξύ των δύο πινάκων:

- Σχέση **πολλά-προς-ένα (M:1)** με πίνακα course
    - Μια εγγραφή course_period μπορεί να έχει ένα και μόνο ένα course (μάθημα). Αντίθετα κάθε course (μάθημα) μπορεί να έχει πολλά course_period.
- Σχέση **πολλά-προς-ένα(M:1)** με πίνακα period
    - Μια εγγραφή course_period μπορεί να έχει ένα και μόνο ένα period (περίοδος). Αντίθετα κάθε period (περίοδος) μπορεί να έχει πολλά course_period.
- Σχέση **ένα-προς-πολλά (1:M)** με πίνακα submission

- Μια εγγραφή course_period (μάθημα περιόδου) μπορεί να έχει κανένα ή περισσότερα submission (άσκηση). Αντίθετα ένα submission (άσκηση) μπορεί να έχει ένα και μόνο ένα course_period. Σημασιολογικά κάθε μάθημα περιόδου μπορεί να έχει πολλές ασκήσεις.

3.2.1.3.6 Πίνακας submission

- Σχέση **πολλά-προς-ένα (M:1)** με τον πίνακα course_period
  - Πολλές εγγραφές submission (άσκηση) μπορούν να έχουν ένα και μόνο ένα course_period.
- Σχέση **ένα-προς-πολλά (1:M)** με τον πίνακα bucket
  - Μια εγγραφή submission (άσκηση) μπορεί να έχει κανένα ή περισσότερα bucket καθηγητών. Αντίθετα κάθε bucket μπορεί να έχει ένα και μόνο ένα submission (άσκηση). Σημασιολογικά κάθε άσκηση μπορεί έχει πολλά διαφορετικά 'αποθετήρια', έτσι ώστε να υπάρχει καλύτερη οργάνωση των τμημάτων φοιτητών που μπορεί κάθε καθηγητής να έχει για μια άσκηση.

3.2.1.3.7 Πίνακας bucket

- Σχέση **πολλά-προς-ένα (M:1)** με τον πίνακα submission
  - Μια εγγραφή bucket μπορεί να έχει ένα και μόνο ένα submission (άσκηση).
- Σχέση **ένα-προς-πολλά (1:M)** με τον πίνακα upload
  - Μια εγγραφή bucket μπορεί να έχει κανένα ή πολλά upload (υποβολή). Αντίθετα πολλά upload (υποβολή) μπορούν να έχουν ένα και μόνο ένα bucket. Σημασιολογικά κάθε bucket/αποθετήριο μιας άσκησης, μπορεί να έχει καμία ή περισσότερες υποβολές.

3.2.1.3.8 Πίνακας upload

- Σχέση **πολλά-προς-ένα (M:1)** με τον πίνακα user
  - Μια εγγραφή upload (υποβολή) μπορεί να έχει ένα και μόνο ένα user (χρήστη).
- Σχέση **πολλά-προς-ένα (M:1)** με τον πίνακα bucket
  - Μια εγγραφή upload (υποβολή) μπορεί να έχει ένα και μόνο ένα bucket.

3.3 Υλοποίηση web συστήματος

3.3.1 Περιβάλλον ανάπτυξης

Τα πρώτα βήματα που πάρθηκαν για την υλοποίηση του διαδικτυακού συστήματος ήταν η αρχικοποίηση ενός αποθετηρίου κώδικα (repository) στην εφαρμογή ελέγχου εκδόσεων (version control system) GitHub. Το στάδιο αυτό κρίθηκε ιδιαίτερα σημαντικό για την σωστή συνεργασία και συντονισμό μεταξύ των μελών της ομάδας ανάπτυξης.

Στη συνέχεια, προχωρήσαμε στην ενσωμάτωση ορισμένων βιβλιοθηκών και εργαλείων, τα οποία θα ήταν ιδιαίτερα χρήσιμα και σημαντικά στην υλοποίηση λειτουργιών.

Κατά την ανάπτυξη της εφαρμογής ενσωματώθηκαν διάφορες βιβλιοθήκες που ενίσχυσαν τη λειτουργικότητα και την αισθητική του περιβάλλοντος χρήστη.

Η Angular Material χρησιμοποιήθηκε για την παροχή έτοιμων γραφικών στοιχείων και διεπαφών, διευκολύνοντας σημαντικά τη δημιουργία μοντέρνου και ευχάριστου UI.

Επιπλέον, επιλέχθηκε η βιβλιοθήκη Angular Svg Icon, η οποία απλοποίησε τον χειρισμό SVG εικονιδίων, προσφέροντας μεγαλύτερη ευελιξία στην προσαρμογή των χαρακτηριστικών τους μέσω κώδικα.

Στη συνέχεια, ενσωματώθηκε η βιβλιοθήκη της Tailwind, η οποία είναι είναι ένα utility-first CSS framework που επιτρέπει την ταχεία ανάπτυξη διεπαφών χρήστη με την χρήση προκαθορισμένων κλάσεων στο HTML. Αυτή η προσέγγιση διευκολύνει τη συντήρηση του κώδικα, προσφέροντας υψηλό επίπεδο παραμετροποίησης και ταχύτητα σχεδιασμού.

Τελευταία βιβλιοθήκη που προστέθηκε είναι η Ngx Material Timepicker, μια βιβλιοθήκη που μας επέτρεψε να ενσωματώσουμε εύχρηστα και καλαίσθητα συστατικά(components) επιλογής ώρας στο περιβάλλον της εφαρμογής.

Όσον αφορά την παγκόσμια γραφική διεπαφή της εφαρμογής, είναι σημαντικό να σημειωθεί ότι τα dashboards του φοιτητή και του διδάσκοντα περικλείονται και τα δύο από το **app-global-container component,** το οποίο είναι ένα επαναχρησιμοποιήσιμο (reusable) component και αποτελεί το παγκόσμιο container για το view της εφαρμογής. Αποτελείται από τα components app-header και app-user-info.

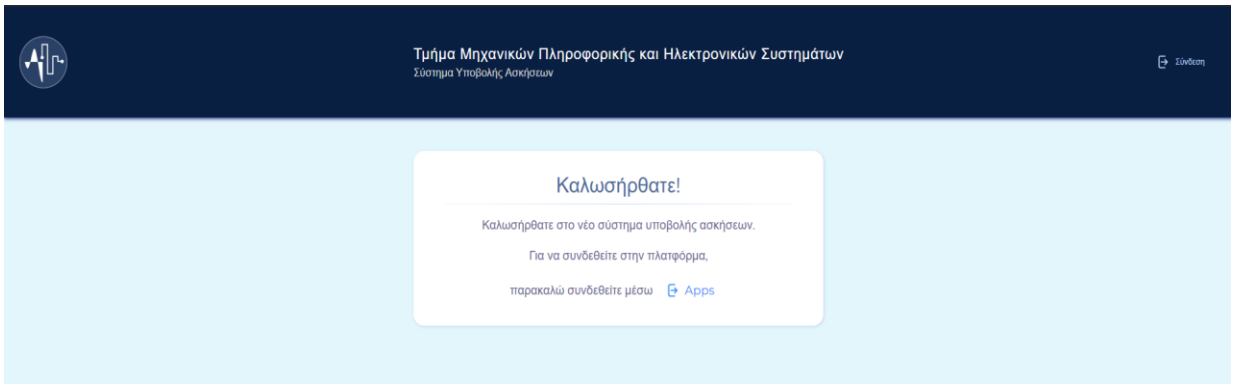
Το app-header είναι το component που φιλοξενεί την κεφαλή της σελίδας που περιέχει πληροφορίες όπως logo και όνομα του πανεπιστημίου,καθώς και τα κουμπιά σύνδεσης-αποσύνδεσης και το app-user-info που παρέχει οπτικές πληροφορίες για τις πληροφορίες του συνδεδεμένου χρήστη.

```
<div class="flex flex-col items-center " [ngClass]="isProfessor?'gap-
0':'gap-12'">
    <app-header class="w-full">
        <app-user-info [user]="user"></app-user-info>
    </app-header>
    <ng-content></ng-content>
</div>
```

Κώδικας 3.1 HTML κώδικας του global-container component

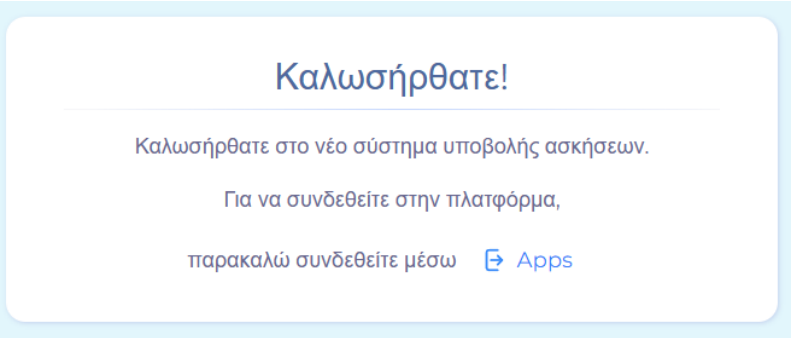
3.3.2 Αυθεντικοποίηση χρηστών

Πρώτος μας στόχος ήταν να υλοποιηθεί η λειτουργία της αυθεντικοποίησης του χρήστη. Ένας χρήστης μπορεί να ξεκινήσει την διαδικασία της αυθεντικοποίησης του μέσα από την αρχική οθόνη της εφαρμογής.



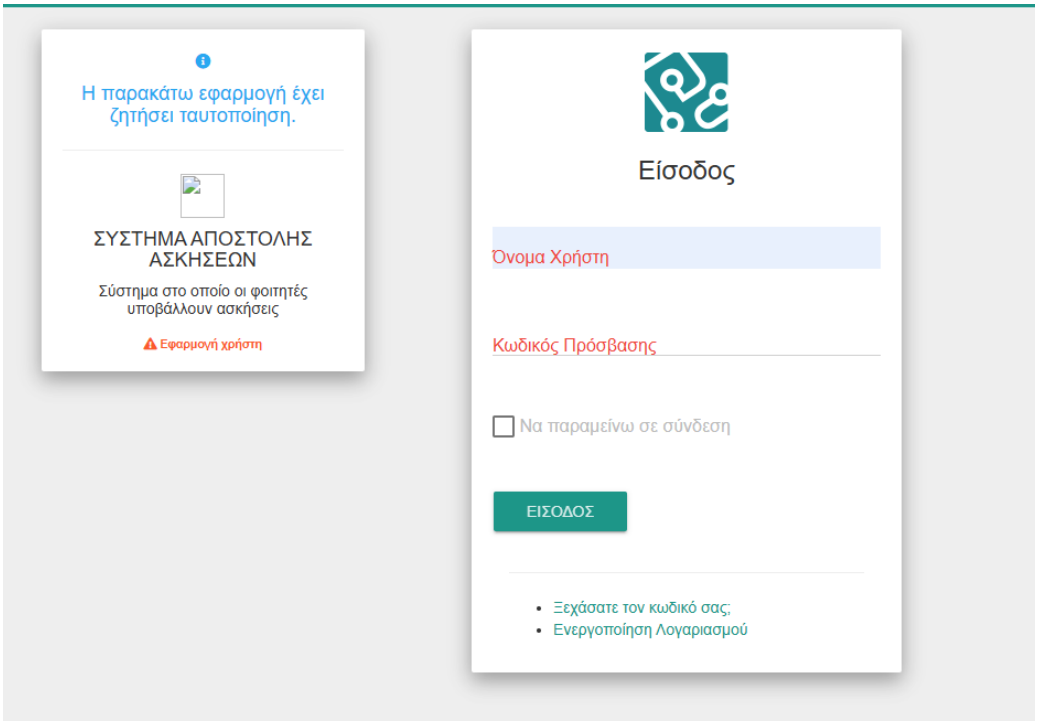
Εικόνα 3.2 Αρχική οθόνη της εφαρμογής

Στην οθόνη αυτή, του δίνεται η δυνατότητα μέσω δύο κουμπιών να ανακατευθυνθεί στο περιβάλλον αυθεντικοποίησης της σχολης. Τα κουμπιά αυτά βρίσκονται στο κεντρικό πλαίσιο καλωσορίσματος (Εικόνα 3.4) και στην πάνω δεξιά γωνία της κεφαλής της σελίδας.



Εικόνα 3.3 Πλαίσιο καλωσορίσματος

Στο πάτημα του κουμπιού σύνδεσης, ο χρήστης μεταφέρεται στο περιβάλλον σύνδεσης του πανεπιστημίου, όπου φαίνεται πως η εφαρμογή του συστήματος αποστολής ασκήσεων ζητάει ταυτοποίηση(Εικόνα 3.5).



Εικόνα 3.4 Ανακατεύθυνση χρήστη στο περιβάλλον σύνδεσης του πανεπιστημίου

Όταν εισάγει τα στοιχεία του και γίνει επιτυχής σύνδεση, μεταφέρεται πίσω στο περιβάλλον της εφαρμογής έχοντας προστεθεί στο url μια παράμετρος με έναν κωδικό. Αρχικά, γίνεται έλεγχος για την ύπαρξη του κωδικού αυτού και αν υπάρχει καλείται η μέθοδος getAuth() του auth service, η οποία λαμβάνει σαν παράμετρο τον κωδικό αυτόν και στη συνέχεια γίνεται κλήση προς το API του πανεπιστημίου για την απόκτηση access και refresh token, τα οποία αποθηκεύονται στα cookies. Με την απόκτηση του access token , γίνεται εκ νέου κλήση στο πανεπιστημιακό API για την απόκτηση των

στοιχείων του χρήστη. Ελέγχεται, στη συνέχεια, αν υπάρχει ο χρήστης στη βάση δεδομένων της εφαρμογής και αν δεν υπάρχει, δημιουργείται εκείνη τη στιγμή , αν υπάρχει γίνεται ενημέρωση. Στο τέλος, γίνεται καθαρισμός της παραμέτρου του κωδικού από το url.

Στην περίπτωση που δεν βρεθεί η παράμετρος του κωδικού, δηλαδή δεν έχει προηγηθεί σύνδεση στο περιβάλλον του πανεπιστημίου, εκτελείται η μέθοδος refreshToken() του auth service, η οποία αναλαμβάνει να επιστρέψει τις πληροφορίες του χρήστη εκμεταλλεύοντας το cookie του refresh token. Αν υπάρχει το cookie, δηλαδή έχει προηγηθεί σύνδεση, καλείται το ακαδημαϊκό API και επιστρέφεται εκ νέου ο χρήστης.

Επιπλέον, παρέχεται η πληροφορία πως το access token έχει διάρκεια ζωής 2 λεπτά μέχρι να λήξει.

Σε κάθε αίτημα προς τον διακομιστή, ελέγχεται η εγκυρότητα του access token. Όταν περάσουν 2 λεπτά από τη στιγμή δημιουργίας του, κάθε χρήση του token αυτού θα καταλήγει σε 401 unauthorized error και απορρίπτεται η επικοινωνία. Για το λόγο αυτό υπάρχει το refresh token, το οποίο σε περίπτωση που έχει λήξει το access token, χρησιμοποιείται για την δημιουργία ενός νέου access και refresh token και έπειτα επαναλαμβάνεται η προηγούμενη αποτυχημένη κλήση προς τον διακομιστή, αυτή τη φορά με το νέο access token. Για τη λειτουργία αυτή είναι υπεύθυνο το αρχείο auth interceptor, το οποίο λαμβάνει και ελέγχει κάθε αίτηση που πραγματοποιείται προς τον διακομιστή, παρέχοντας έτσι τη δυνατότητα να χειραγωγείται κάθε αίτημα με οποιονδήποτε τρόπο όπως τον έλεγχο σφαλμάτων των http αποκρίσεων και την προσθήκη κεφαλίδων ταυτοποίησης (authrozition bearer token).

Στο τέλος και των δύο περιπτώσεων, εκτελείται η μέθοδος redirectUser(), η οποία ελέγχει τον ρόλο του χρήστη και τον ανακατευθύνει στο κατάλληλο περιβάλλον, το οποίο προστατεύεται από το auth guard και το role guard, τα οποία προστατεύουν τα περιβάλλοντα αυτά από χρήστες που είναι μη συνδεδεμένοι και από χρήστες των οποίων ο ρόλος δεν επιτρέπεται για την πρόσβαση του περιβάλλοντος, αντίστοιχα.

Σε περίπτωση που ο χρήστης επιθυμεί να αποσυνδεθεί από την εφαρμογή, καλείται η logout μέθοδος του auth service , η οποία διαγράφει το refresh και το access token από τα cookies και στη συνέχεια ο χρήστης μεταφέρεται στην αρχική σελίδα, όντας πλέον αποσυνδεμένος έχοντας καθαρίσει όλα τα στοιχεία της σύνδεσης του.

3.3.3 Περιβάλλον του φοιτητή

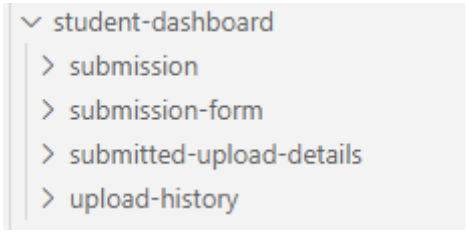
Στο σημείο αυτό, ξεκίνησε η υλοποίηση του γραφικού περιβάλλοντος του φοιτητή.

Αρχικά, ο χρήστης μεταφέρεται στο περιβάλλον αυτό αφού γίνει έλεγχος μέσω του φύλακα ταυτοποίησης γνωστό στην Angular κι ως Auth Guard, ότι ο χρήστης είναι συνδεδεμένος και ,ταυτόχρονα, έχει ρόλο φοιτητή.

Στη συνέχεια, προχωρήσαμε στην αρχικοποίηση όλων **Angular Components** που αντιπροσωπεύουν την οθόνη του φοιτητή. Η διαδικασία αυτή συνέβαλε στη διαμόρφωση μιας πλήρους εικόνας για το view του φοιτητή, παρέχοντας τη δυνατότητα πρόσβασης σε κάθε σημαντική λειτουργία.

3.3.3.1 Φόρμα υποβολής

Όπως φαίνεται και στην Εικόνα 3.5, το dashboard του φοιτητή αποτελείται από τέσσερα κύρια components. Το submission component λειτουργεί ως το container για την οθόνη του φοιτητή (Κώδικας 3.2) καθώς περικλείει τα submission-form και upload-history components.

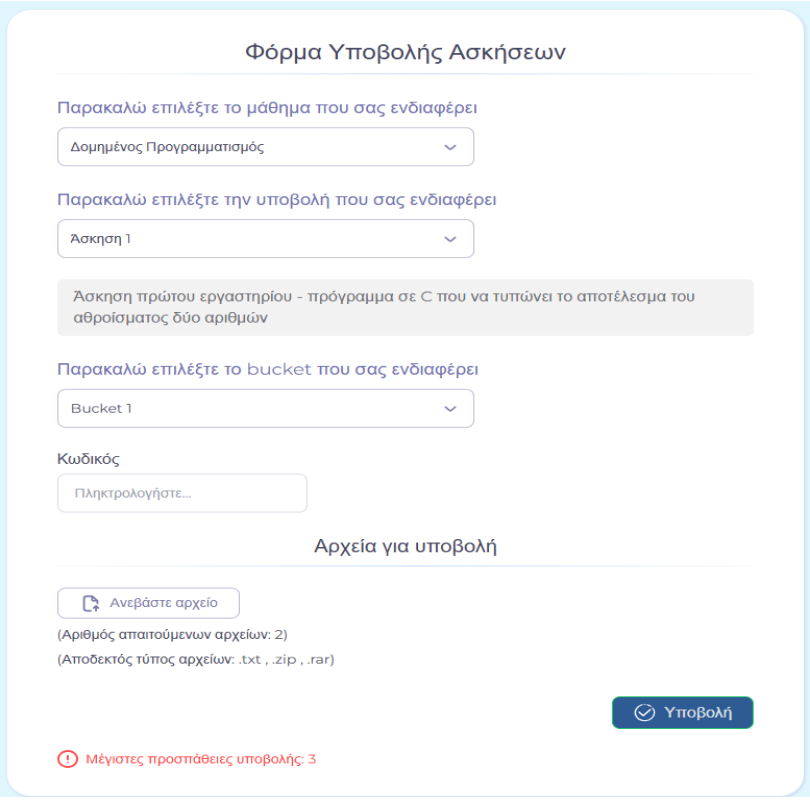


Εικόνα 3.5 Ιεραρχία των components του view του φοιτητή

```
<app-global-container>
    <div class="flex flex-col gap-0  items-center w-full justify-end">
        <app-submission-form></app-submission-form>
        <app-upload-history></app-upload-history>
    </div>
</app-global-container>
```

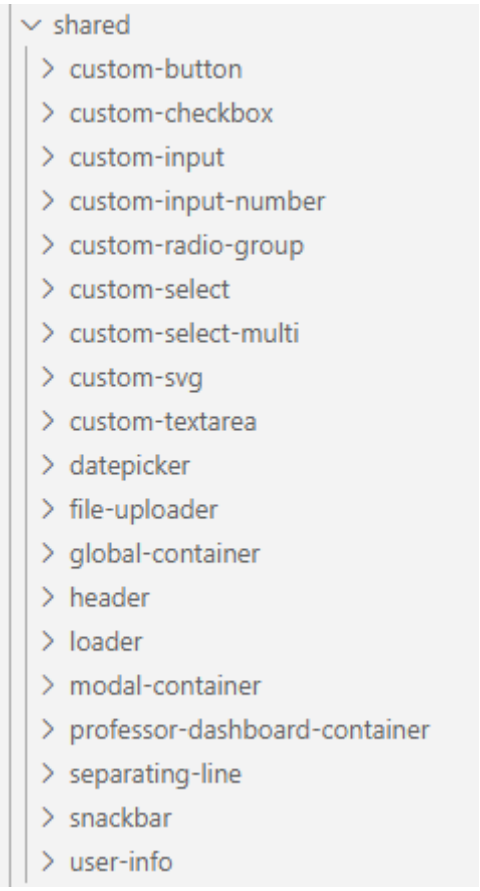
Κώδικας 3.2 HTML κώδικας του submission component

Το **submission-form** component είναι υπεύθυνο για την απόδοση και τη διαχείριση όλων των λειτουργιών που σχετίζονται με τη φόρμα υποβολής του φοιτητή. Συγκεκριμένα, αυτό το component χειρίζεται την εμφάνιση της φόρμας στην οποία ο φοιτητής μπορεί να εισαγάγει τα απαιτούμενα στοιχεία για την υποβολή της άσκησης ή της εργασίας του. Επιπλέον, διασφαλίζει τη σωστή επικύρωση των δεδομένων, την υποβολή τους στον διακομιστή, καθώς και την εμφάνιση τυχόν μηνυμάτων επιτυχίας ή σφαλμάτων. Είναι το σημείο αναφοράς για τον φοιτητή, καθώς παρέχει τις αναγκαίες διεπαφές και λειτουργίες για την ολοκλήρωση της διαδικασίας υποβολής.



Εικόνα 3.6 Γραφική διεπαφή φόρμας

Η φόρμα δημιουργήθηκε με τέτοιο τρόπο έτσι ώστε να δίνει στον φοιτητή την αίσθηση της εξέλιξης και της οπτικής ικανοποίησης. Έχει προγραμματιστεί με τέτοιο τρόπο που να μη δίνεται στον φοιτητή εξ αρχής ολόκληρη η πληροφορία που πρέπει να καταβάλει αλλά βήμα βήμα. Αρχικά, ο φοιτητής βλέπει ένα select-box, το οποίο ζητάει να επιλέξει το μάθημα που ενδιαφέρει τον φοιτητή. Το συγκεκριμένο select-box υλοποιείται μέσω του custom-select component, το οποίο είναι ένα από τα components που βρίσκονται στον φάκελο **shared** (Εικόνα 3.9) **,** δηλαδή ανήκει στα components εκείνα τα οποία μοιράζονται και επαναχρησιμοποιούνται σε πολλά σημεία της εφαρμογής.



Εικόνα 3.7 Ιεραρχία φακέλου για τα shared components

Λαμβάνει τέσσερα Inputs. Το “value” που αντιπροσωπεύει την τιμή που έχει, το “label” για την ετικέτα που εμφανίζεται σαν περιγραφή, το “data” για τα δεδομένα που εμφανίζονται σαν επιλογές στο dropdown μενού, το “hasSearch” που υποδηλώνει αν μέσα δίνεται η επιλογή για αναζήτηση των διαθέσιμων επιλογών και τα “secondaryUI” και “isResponsive”, τα οποία πραγματοποιούν κάποιες προσαρμογές στην εμφάνιση για κάποιες συγκεκριμένες περιπτώσεις.

```
@Input() value?: T;
  @Input() label?: string;
  @Input() data: T[] = [];
  @Input() secondaryUI = false;
  @Input() isResponsive = false;
  @Input() hasSearch = true;
```

Κώδικας 3.3 Τα inputs που δέχεται το custom-select component

Στη συνέχεια, αφού γίνει επιλογή μαθήματος, γίνεται κλήση προς τον διακομιστή για να στείλει τις υποβολές που συνδέονται με το επιλεγμένο μάθημα(Κώδικας 3.4), ενημερώνεται το HTML και μέσω

της συνθήκης If που κατοικεί στον κώδικα του HTML(Κώδικας 3.5) εμφανίζεται το επόμενο select-box που ζητάει από τον φοιτητή να επιλέξει από το μενού την υποβολή που τον ενδιαφέρει για το μάθημα που επέλεξε προηγουμένως. Η σύνταξη @If έχει αντικαταστήσει το directive NgIf στις νεότερες εκδόσεις της Angular.

```
getCourseSubmissions(courseCode:number,period:Period):Observable<Submissi
onResponse>{
    return
this.http.get<SubmissionResponse>(`courses/${courseCode}/semesters/${peri
od.semester}/years/${period.year}/submissions`);
  }
```

Κώδικας 3.4 Στιγμιότυπο μεθόδου του submission service που ζητάει και επιστρέφει τις υποβολές ενός μαθήματος

```
 @if (selectedCourse$.value) {
    <app-custom-select
      class="animate-slideDown"
      [data]="submissions"
      [label]="'Παρακαλώ επιλέξτε την υποβολή που σας ενδιαφέρει'"
      [isResponsive]="true"
      [loader]="submissionsLoader"
      [value]="selectedSubmission$.value"
      (valueChange)="selectedSubmission$.next($event);
       selectedBucket$.next(undefined)"
    ></app-custom-select>
```

Κώδικας 3.5 Χρήση σύνταξης @if statement για δυναμική απόδοση HTML

Με την επιλογή της ενδιαφερόμενης υποβολής, γίνεται και πάλι κλήση προς τον server για την αποστολή αιτήματος και λήψης των buckets-αποθετηρίων που αντιστοιχούν στην επιλεγμένη υποβολή. Με την επιλογή ενός bucket, γίνεται έλεγχος αν το bucket αυτό απαιτεί κωδικό για να γίνει υποβολή και στην περίπτωση που απαιτείται κωδικός, αποδίδεται δυναμικά στην οθόνη το περιεχόμενο του shared component **custom-input** στο οποίο ο φοιτητής καλείται να εισάγει τον κωδικό του bucket. Στην φόρμα εμφανίζονται ,επίσης, τα συστατικά που αφορούν το ανέβασμα αρχείων.

Ειδικότερα, αποδίδεται στην οθόνη το **file-uploader** component, το οποίο περιέχει το πρότυπο και την λογική για το ανέβασμα ενός αρχείου. Λαμβάνει 3 inputs, “label”, “maxFiles” και “mimeTypes”, τα οποία αντιπροσωπεύουν την ετικέτα περιγραφής, τον μέγιστο αριθμό αρχείων που μπορεί να ανεβάσει ο φοιτητής και τον τύπο των αποδεκτών αρχείων (Κώδικας 3.6).

```
  @Input() label?: string;
  @Input() maxFiles = 5
  @Input() mimeTypes?:string
  @Output() filesChange = new EventEmitter<File[]>()
```

Κώδικας 3.6 Inputs και Outputs του file-uploader component

Την στιγμή που ανεβάσει ένα αρχείο ο χρήστης, καλείται η μέθοδος του Κώδικα 3.7, η οποία αποθηκεύει τα αρχεία που έχουν ανέβει μέχρι τώρα τοπικά σε έναν πίνακα που αποτελείται από αντικείμενα τύπου File και τα στέλνει στο γονικό component δηλαδή το submission-form component μέσω του filesChange Event Emitter και ειδικότερα της μεθόδου emit() η οποία εκπέμπει δεδομένα προς τα έξω.

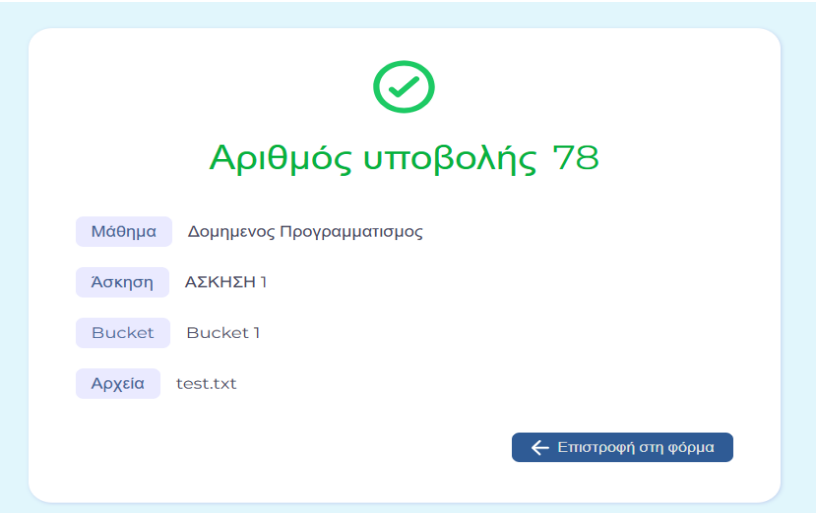
```
 onFileSelected(event: any): void {
    event.preventDefault();
    if(!this.checkForErrors(event.target.files))return
    this.uploading = true;
      const newFiles= Array.from(event.target.files) as File[]
      this.files.push(...newFiles)
      this.uploading = false;
    newFiles.length &&  this.snackbarService.showSnackbar('success','Τα
αρχεία ανέβηκαν επιτυχώς')
    this.filesChange.emit(this.files)
  }
```

Κώδικας 3.7 Μέθοδος που καλείται στο ανέβασμα ενός αρχείου

Πλέον, ο φοιτητής έχει συμπληρώσει όλες τις απαραίτητες πληροφορίες και είναι έτοιμος να πατήσει το κουμπί “Υποβολή”. Στο πάτημα, ελέγχεται η ορθότητα των δεδομένων και στη συνέχεια αποστέλλεται μια κλήση προς τον server με τα περιεχόμενα της υποβολής και του φοιτητή μέσω του **uploads service**. Τα δεδομένα αποστέλλονται στη μορφή **form data**, δηλαδή με μια δομή που χρησιμοποιείται ευρέως για την υποβολή φορμών μέσω του πρωτοκόλλου HTTP. Η μορφή αυτή επιτρέπει την αποστολή πεδίων κειμένου αλλά και αρχείων, όπως εικόνες ή έγγραφα, σε έναν server.

3.3.3.2 Μήνυμα επιτυχίας

Μόλις η φόρμα υποβληθεί και τα δεδομένα καταχωρηθούν με επιτυχία, εμφανίζεται στην οθόνη το περιεχόμενο του **submitted-upload-details** component που περιέχει ένα πλαίσιο με τον αριθμό αποστολής της υποβολής, καθώς και όλες τις πληροφορίες που είχε συμπληρώσει ο φοιτητής. Η εμφάνιση αυτού του πλαισίου γίνεται δυναμικά, μόνο όταν επιβεβαιωθεί ότι η διαδικασία καταχώρησης ολοκληρώθηκε σωστά (Εικόνα 3.15).



Εικόνα 3.8 Πλαίσιο μηνύματος επιτυχίας

3.3.3.3 Ιστορικό υποβολών

Η τελευταία βασική δυνατότητα που προσφέρεται στον φοιτητή μέσα από το περιβάλλον της εφαρμογής είναι η προβολή του ιστορικού των υποβολών του. Δηλαδή, ο φοιτητής μπορεί να δει συγκεντρωμένες όλες τις υποβολές που έχει πραγματοποιήσει στο παρελθόν, μαζί με σχετικές

πληροφορίες όπως η ημερομηνία υποβολής, το μάθημα για το οποίο έγιναν και άλλα σχετικά στοιχεία. Αυτό του δίνει τη δυνατότητα να ελέγχει τι έχει ήδη υποβάλει και να παρακολουθεί την πορεία των εργασιών του. Τη λειτουργία του ιστορικού αναλαμβάνει το **upload-history** component, το οποίο εμφανίζει μια λίστα με όλες τις υποβολές που έχει κάνει ο φοιτητής, αξιοποιώντας την μέθοδο getUserUploadHistory του **user service** για την ανάκτηση των σχετικών δεδομένων (Κώδικας 3.8). Επιπλέον, παρέχεται ένα κουμπί ανανέωσης που επιτρέπει στον χρήστη να φορτώσει εκ νέου τις υποβολές, εξασφαλίζοντας ότι βλέπει την πιο πρόσφατη πληροφορία. Η προβολή ή απόκρυψη της λίστας γίνεται διαδραστικά μέσω ενός εικονιδίου με σχήμα "ματιού", στο οποίο ο χρήστης μπορεί να κάνει κλικ για να εναλλάσσει την ορατότητα της λίστας.

```
getUserUploadHistory(userId:number):Observable<Upload[]>{
   return   this.http.get<Upload[]>(`users/uploads`)
  }
```

Κώδικας 3.8 Μέθοδος που επιστρέφει το ιστορικό υποβολών ενός φοιτητή

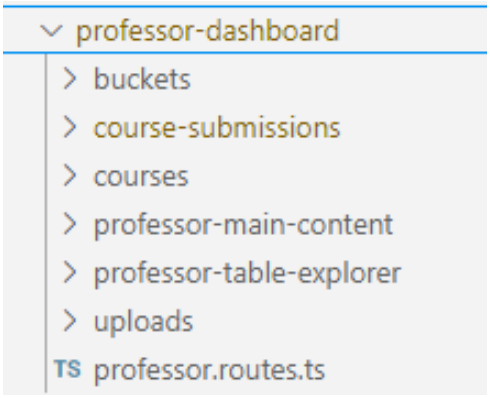
3.3.4 Διαχειριστικό περιβάλλον διδάσκοντα

Το διαχειριστικό περιβάλλον του διδάσκοντα αποτελεί ένα ειδικά διαμορφωμένο τμήμα της εφαρμογής, σχεδιασμένο για να καλύπτει τις ανάγκες εποπτείας και διαχείρισης των υποβολών και των μαθημάτων. Μέσα από ένα φιλικό και λειτουργικό περιβάλλον εργασίας, ο διδάσκων έχει τη δυνατότητα να παρακολουθεί τις υποβολές των φοιτητών, να ελέγχει τα δεδομένα που έχουν καταχωρηθεί, να οργανώνει τις δραστηριότητες του ανά μάθημα και να εκτελεί βασικές ενέργειες. Στόχος του διαχειριστικού περιβάλλοντος είναι να προσφέρει στον διδάσκοντα έναν εύχρηστο και αποδοτικό τρόπο διαχείρισης της εκπαιδευτικής διαδικασίας μέσω της πλατφόρμας.

Οι ενέργειες που μπορεί να εκτελέσει ο διδάσκοντας μέσω του διαχειριστικού του περιβάλλοντος είναι οι εξής:

- **Προβολή λίστας μαθημάτων**: Δυνατότητα προβολή λίστας μαθημάτων με επιλογή φιλτραρίσματος ανά περίοδο (συνδυασμός εξαμήνου και χρονιάς).
- **Δημιουργία μαθήματος**: Ο διδάσκοντας έχει την δυνατότητα να δημιουργεί μαθήματα αφού επιλέξει το εξάμηνο και την χρονιά στην οποία θέλει να προσθέσει το μάθημα.
- **Προσθήκη διδάσκοντα σε μάθημα**: Δυνατότητα προσθήκης ενός διδάσκοντα σε ένα μάθημα ,συγκεκριμένης περιόδου, που έχει ήδη δημιουργηθεί ήδη από άλλο διδάσκοντα.
- **Προβολή λίστας υποβολών ενός μαθήματος**: Δυνατότητα προβολής λίστας των υποβολών που έχουν δημιουργηθεί για ένα μάθημα.
- **Διαχείριση υποβολών**: Δυνατότητα δημιουργίας - επεξεργασίας υποβολών καταχωρώντας τις απαραίτητες πληροφορίες καθώς και δυνατότητα διαγραφής.
- **Μεταφορά υποβολών**: Δυνατότητα μεταφοράς υποβολών από μάθημα παλαιότερης περιόδου σε μάθημα επιλεγμένης περιόδου.
- **Προβολή λίστας bucket μίας υποβολής**: Δυνατότητα προβολής σε λίστα όλα τα bucket που κατοικούν σε μια υποβολή.
- **Διαχείριση bucket**: Δυνατότητα δημιουργίας - επεξεργασίας bucket για μια υποβολή καταχωρώντας τις απαραίτητες πληροφορίες καθώς και δυνατότητα διαγραφής.
- **Προβολή λίστας upload φοιτητών**: Προβολή λίστας με τα uploads που έχουν υποβάλει οι φοιτητές στο επιλεγμένο bucket.
- **Κατέβασμα αρχείων**: Δυνατότητα κατεβάσματος ενός ή όλων των αρχείων που έχουν ανεβάσει οι φοιτητές στο επιλεγμένο bucket.

Στην Εικόνα 3.17 φαίνεται ο τρόπος με τον οποίο έχει δομηθεί η ιεραρχία των components για το περιβάλλον του διδάσκοντα. Αποτελείται από 4 κύρια components και ένα αρχείο που περιέχει τα routes του περιβάλλοντος.

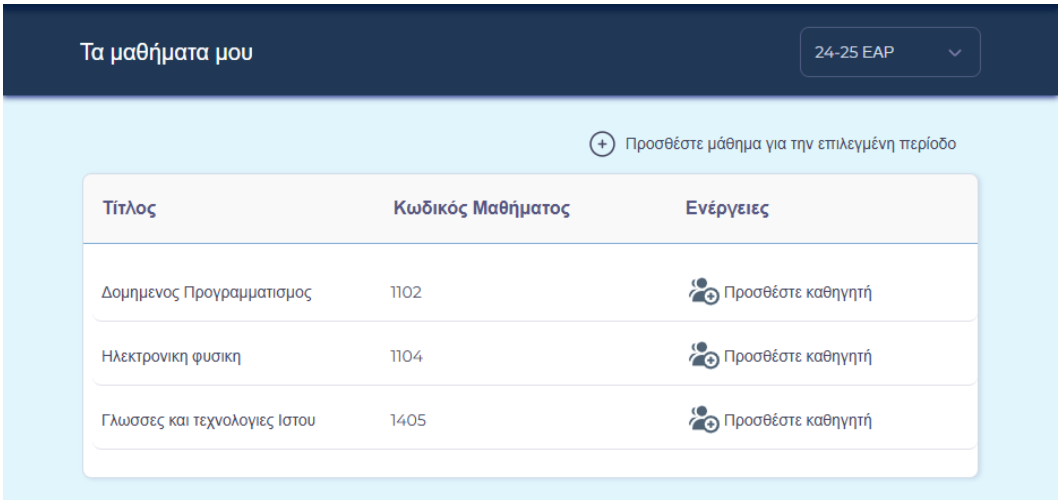


Εικόνα 3.9 Ιεραρχία components του περιβάλλοντος του διδάσκοντα

Στα επόμενα υποκεφάλαια, θα εξηγηθούν λεπτομερώς οι λειτουργίες κάθε οθόνης του διαχειριστικού του διδάσκοντα.

3.3.4.1 Οθόνη μαθημάτων

Η οθόνη των μαθημάτων είναι η πρώτη οθόνη που βλέπει ο διδάσκοντας με το που γίνει η αυθεντικοποίηση του και μεταφερθεί στο περιβάλλον του. Το component που αποδίδεται στην οθόνη είναι το **courses component**. Έχει στη διάθεση του έναν πίνακα (custom-table component), ο οποίος προβάλλει σε μορφή λίστας τα μαθήματα(Εικόνα 3.18) τα οποία είναι διασυνδεδεμένα με τον συγκεκριμένο διδάσκοντα. Έχει τη δυνατότητα να φιλτράρει τα αποτελέσματα ανά την περίοδο, η οποία αποτελείται από εξάμηνο και έτος και αποθηκεύεται στα url query params.



Εικόνα 3.10 Γραφική διεπαφή οθόνης μαθημάτων

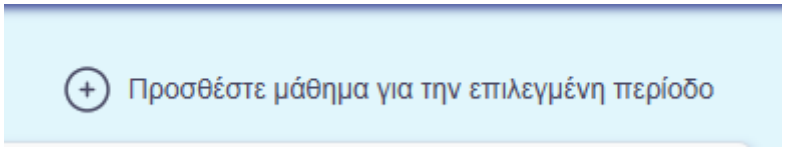
Η μέθοδος handlePeriodSelection (Κώδικας 3.9) είναι υπεύθυνη για τη σωστή ενεργοποίηση της περιόδου που έχει επιλεγεί. Αν υπάρχει ήδη αποθηκευμένη κάποια πληροφορία για την περίοδο στη διεύθυνση της σελίδας (query params), όπως όταν ο χρήστης έχει επιλέξει περίοδο και κάνει ανανέωση της σελίδας, τότε η μέθοδος ενεργοποιεί αυτή την περίοδο αυτόματα. Αν όμως ο χρήστης ανοίγει τη σελίδα για πρώτη φορά και δεν υπάρχει καμία προηγούμενη επιλογή, τότε εμφανίζεται ως προεπιλεγμένη η τρέχουσα περίοδος. Τέλος, όταν ο χρήστης επιλέγει χειροκίνητα κάποια περίοδο από

το μενού, η μέθοδος καταγράφει και εφαρμόζει αυτή την επιλογή. Με την επιστροφή της μεθόδου, γίνεται κλήση προς τον server μέσω του **coursesService** για την λήψη των μαθημάτων της περιόδου.

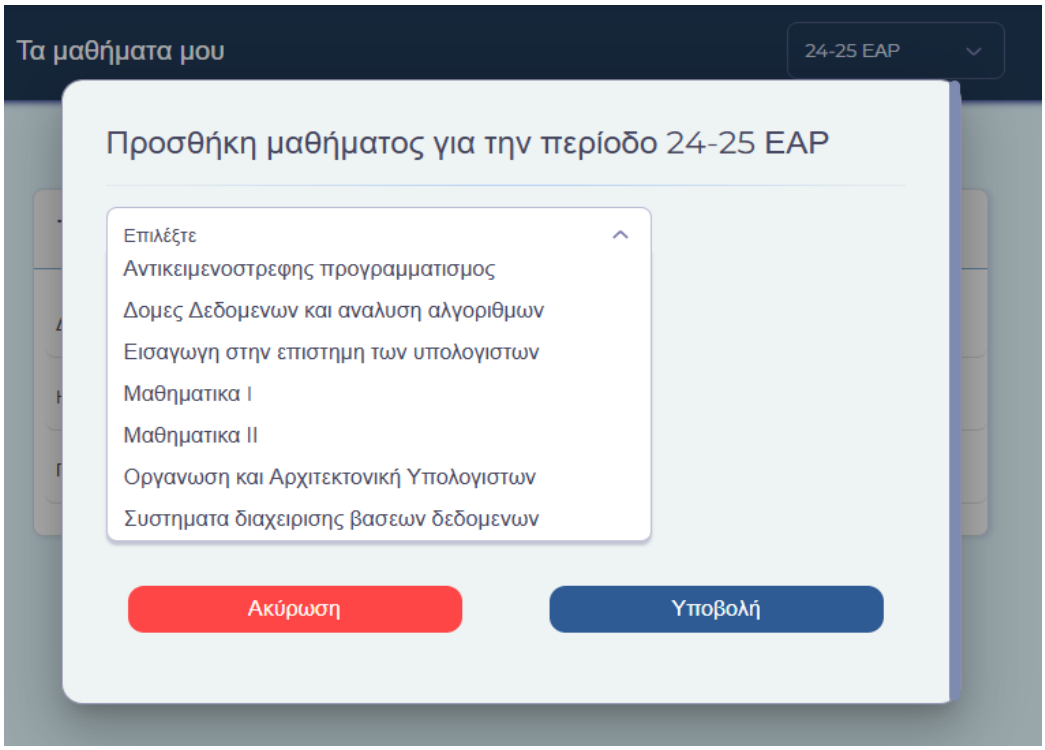
```
handlePeriodSelection(selectedPeriod: Period | null): Period {
    if (!selectedPeriod) {
      let [year, semester] = [
        this.route.snapshot.queryParams['year'],
        this.route.snapshot.queryParams['semester'],
      ];
      //if year and semester dont exist  in url params then it is the first
time the page is loaded
      if (!year && !semester) {
        const defaultPeriod = this.periods[0]; //save the newest period as
the default
        this.updateParams(defaultPeriod);
        return defaultPeriod;
      }
      //if year and semester already exist in url params
      selectedPeriod =
      {
        year: Number(year),
        semester: semester as SemesterType,
          name: this.periods.find(p => p.year == year && p.semester ==
semester)?.name
      };
      return selectedPeriod;
    }
    //if year and semester is manually selected from the dropdown
    this.updateParams(selectedPeriod);
    return selectedPeriod;
  }
```

Κώδικας 3.9 Μέθοδος handlePeriodSelection()

Επιπλέον,από την ίδια οθόνη ο καθηγητής έχει τη δυνατότητα να προσθέτει μαθήματα για την περίοδο που είναι επιλεγμένη. Στο πάτημα του κουμπιού “Προσθέστε μάθημα για την επιλεγμένη περίοδο”(Εικόνα 3.20) ανοίγει ένα παράθυρο modal (Εικόνα 3.21) με τη χρήση του MatDialog service, το οποίο διαθέτει ένα dropdown μενού με τα διαθέσιμα μαθήματα προς προσθήκη. Από τη στιγμή που προστεθεί ένα μάθημα από κάποιον καθηγητή σε μια συγκεκριμένη περίοδο, το μάθημα αυτό σταματάει να υπάρχει σαν διαθέσιμη επιλογή. Με το κλείσιμο του παραθύρου, γίνεται εκ νέου κλήση προς το server για την λήψη των νέων δεδομένων.

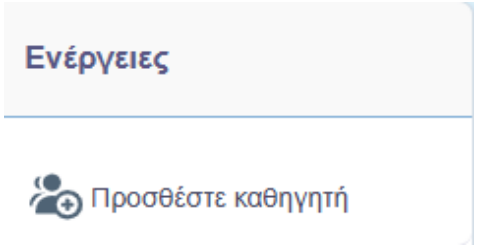


Εικόνα 3.11 Κουμπί πρόσθεσης μαθήματος

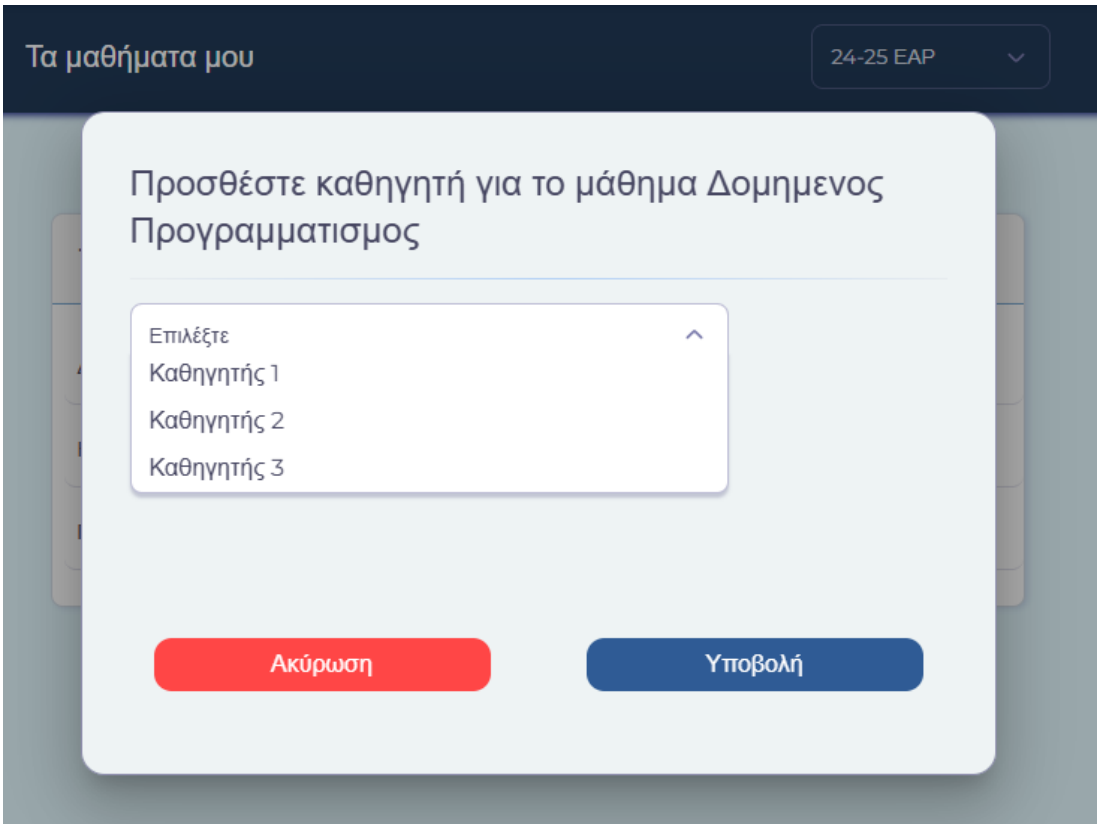


Εικόνα 3.12 Γραφική διεπαφή αναδυόμενου παραθύρου για προσθήκη μαθήματος

Τελευταία λειτουργία της οθόνης, είναι η λειτουργία πρόσθεσης διδάσκοντα σε ένα μάθημα. Όταν ο χρήστης πατήσει το αντίστοιχο κουμπί (Εικόνα 3.22), ανοίγει ένα αναδυόμενο παράθυρο (Εικόνα 3.23) στο οποίο εμφανίζονται τα δεδομένα των καθηγητών, όπως έχουν ληφθεί από το API που παρέχει το πανεπιστήμιο, μέσω ενός dropdown μενού. Αφού επιλεγεί ο ενδιαφερόμενος καθηγητής , η εφαρμογή επικοινωνεί με το API του backend συστήματος και προσθέτει τον καθηγητή στο μάθημα που έχει επιλεγεί μέσω POST request.



Εικόνα 3.13 Κουμπί προσθήκης καθηγητή στο μάθημα



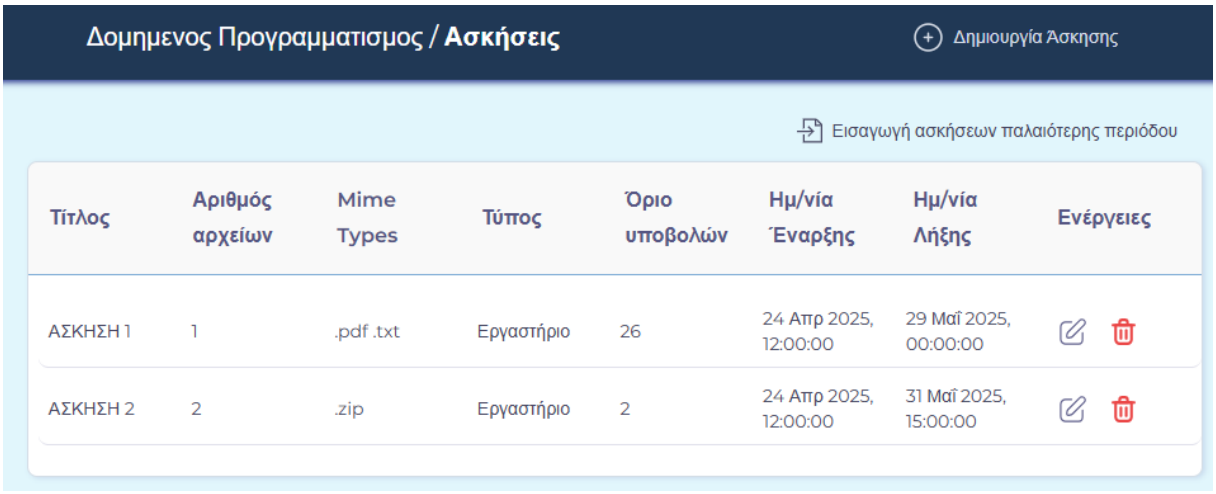
Εικόνα 3.14 Γραφική διεπαφή αναδυόμενου παραθύρου για προσθήκη καθηγητή στο μάθημα

Όταν ο χρήστης πατήσει πάνω σε μία εγγραφή του πίνακα, μεταφέρεται στην επόμενη διαδρομή που είναι οι ασκήσεις του μαθήματος.

3.3.4.2 Οθόνη ασκήσεων

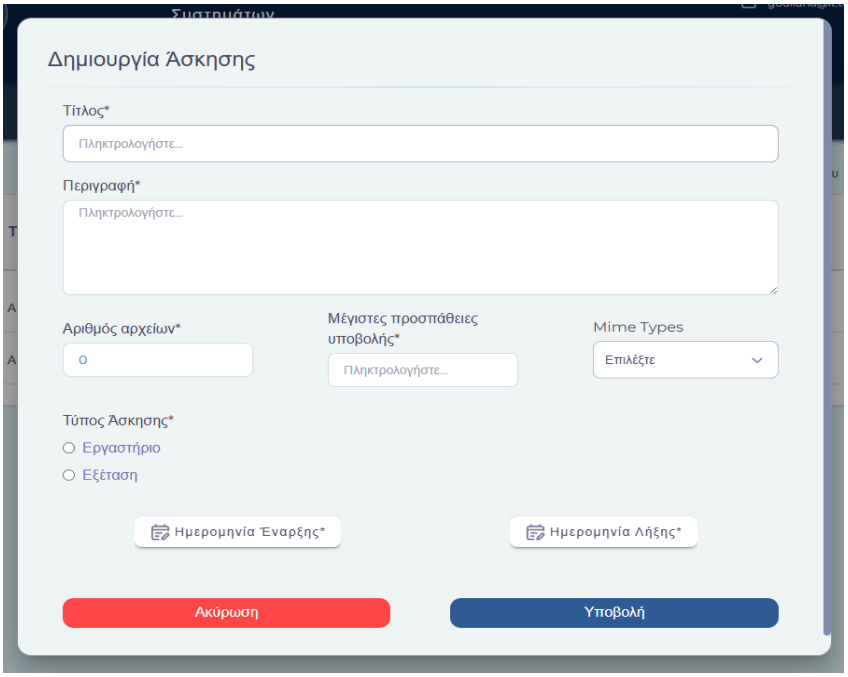
Η οθόνη των ασκήσεων αποτελεί ένα βασικό σημείο λειτουργικότητας για τον διδάσκοντα, καθώς του παρέχει τη δυνατότητα να διαχειρίζεται τις υποβολές των φοιτητών με οργανωμένο και αποτελεσματικό τρόπο.

Στην οθόνη αυτή εμφανίζεται το component **course-submissions**, το οποίο είναι υπεύθυνο για την παρουσίαση των ασκήσεων που έχουν δημιουργηθεί για κάθε μάθημα. Μέσω ενός πίνακα (table component) που προβάλλεται στην οθόνη, ο διδάσκοντας έχει τη δυνατότητα να δει αναλυτικά τις διαθέσιμες υποβολές. Ο πίνακας αυτός περιλαμβάνει χρήσιμες πληροφορίες για κάθε άσκηση, όπως τον τίτλο και την περιγραφή της, τον τύπο της (αν πρόκειται για εργαστήριο ή εξέταση), τους αποδεκτούς τύπους αρχείων (mime types), τον μέγιστο αριθμό αρχείων που μπορεί να ανεβάσει ο φοιτητής, το επιτρεπόμενο αριθμό υποβολών από τον ίδιο φοιτητή, καθώς και τις ημερομηνίες έναρξης και λήξης της υποβολής (Εικόνα 3.24).



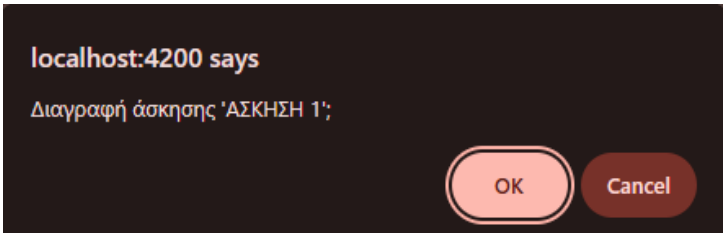
Εικόνα 3.15 Πίνακας δεδομένων των ασκήσεων

Όταν ο διδάσκοντας επιλέξει να δημιουργήσει ή να επεξεργαστεί μια άσκηση μέσω του ανάλογου κουμπιού, ανοίγει ένα αναδυόμενο παράθυρο (modal) με τη χρήση του MatDialog της Angular Material. Στο παράθυρο αυτό εμφανίζεται το component **submission-modal component** μια φόρμα συμπλήρωσης (Εικόνα 3.25), μέσω της οποίας μπορεί να εισάγει όλα τα απαραίτητα στοιχεία για την άσκηση. Τα διαθέσιμα πεδία της φόρμας περιλαμβάνουν τον τίτλο και την περιγραφή της άσκησης (custom-input component), τον τύπο της με επιλογή μεταξύ "Εργαστήριο" και "Εξέταση"(custom-radio-group component), τους αποδεκτούς τύπους αρχείων - mime types (custom-select-multi component), το μέγιστο πλήθος αρχείων που μπορεί να ανεβάσει ο φοιτητής (custom-input-number component), το όριο υποβολών που επιτρέπεται να κάνει για την ίδια άσκηση, καθώς και την ημερομηνία (datepicker component της Angular Material) και ώρα (ngx-material-timepicker) component έναρξης και λήξης υποβολών. Η φόρμα έχει σχεδιαστεί ώστε να είναι εύχρηστη και κατανοητή, διευκολύνοντας τον διδάσκοντα στη διαχείριση των απαιτούμενων στοιχείων. Με την υποβολή της φόρμας, καλείται η αντίστοιχη μέθοδος του submissionService και λαμβάνονται από τον server τα ανανεωμένα δεδομένα.



Εικόνα 3.16 Γραφική διεπαφή φόρμας δημιουργίας άσκησης

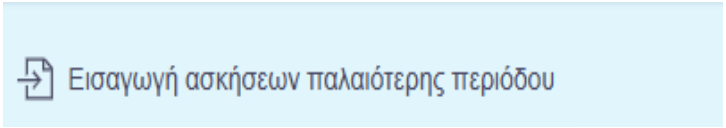
Όταν ο χρήστης επιλέξει το κουμπί διαγραφής μιας άσκησης, εμφανίζεται ένα παράθυρο προειδοποίησης (alert) που του ζητάει να επιβεβαιώσει την ενέργεια.



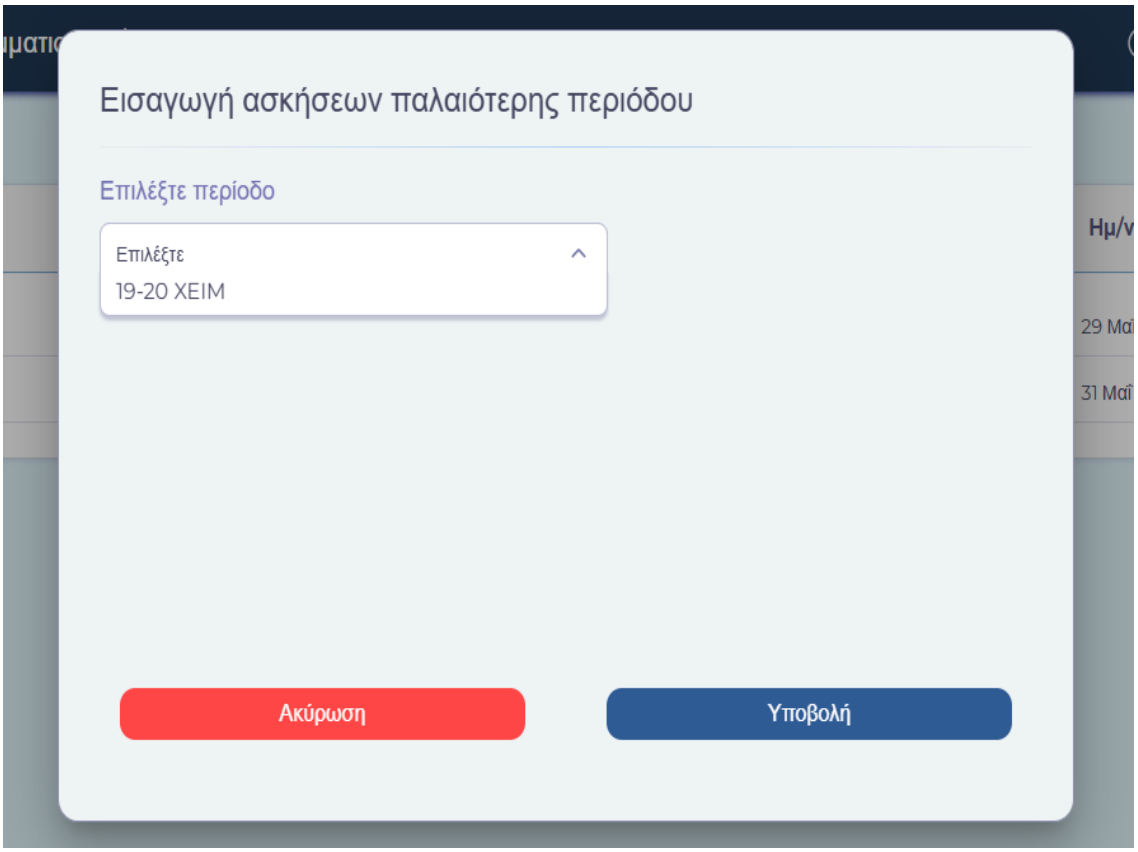
Εικόνα 3.17 Παράθυρο επιβεβαίωσης διαγραφής άσκησης

Αν ο χρήστης επιβεβαιώσει, τότε αποστέλλεται ένα HTTP αίτημα τύπου DELETE προς τον διακομιστή μέσω της deleteSubmission μεθόδου που λαμβάνει ως παράμετρο το id της άσκησης, με σκοπό τη διαγραφή της συγκεκριμένης άσκησης από το σύστημα. Το σύστημα στη συνέχεια δεν πραγματοποιεί πραγματική διαγραφή, αλλά soft delete δηλαδή αλλάζει την τιμή του πεδίου isDeleted της άσκησης, έτσι ώστε να υπάρχει backup στην περίπτωση που χρειαστεί. Μόλις ολοκληρωθεί επιτυχώς η διαδικασία, γίνεται ανανέωση των δεδομένων, ώστε να αποτυπώνεται η τρέχουσα κατάσταση και να μην εμφανίζεται πλέον η διαγραμμένη άσκηση στη λίστα.

Τελευταία λειτουργία της οθόνης των ασκήσεων είναι η λειτουργία μεταφοράς ασκήσεων παλαιότερης περιόδου στην επιλεγμένη περίοδο. Η διαδικασία ξεκινάει με το πάτημα του αντίστοιχου κουμπιού (Εικόνα 3.27), το οποίο εκκινεί την εμφάνιση ενός αναδυόμενου παραθύρου **submission-import-modal** component, μέσα απ το οποίο ο χρήστης καλείται να επιλέξει πρώτα την περίοδο από την οποία θέλει να μεταφέρει ασκήσεις (Εικόνα 3.28). Στο μενού εμφανίζονται μόνο οι περίοδοι εκείνες για τις οποίες το μάθημα αυτό υπάρχει και έχει ασκήσεις. Αν υπάρχει το μάθημα σε μια περίοδο αλλά δεν έχει ασκήσεις, τότε η περίοδος αυτή δεν θα εμφανιστεί καθόλου σαν διαθέσιμη επιλογή. Υπεύθυνη για την λειτουργία της λήψης των διαθέσιμων περιόδων είναι η μέθοδος getPeriodsOfAvailableSubmissions(Κώδικας 3.10) του periodService, το οποίο λαμβάνει σαν παραμέτρους τον κωδικό του μαθήματος και το αντικείμενο period.



Εικόνα 3.18 Κουμπί εισαγωγής παλαιότερων ασκήσεων

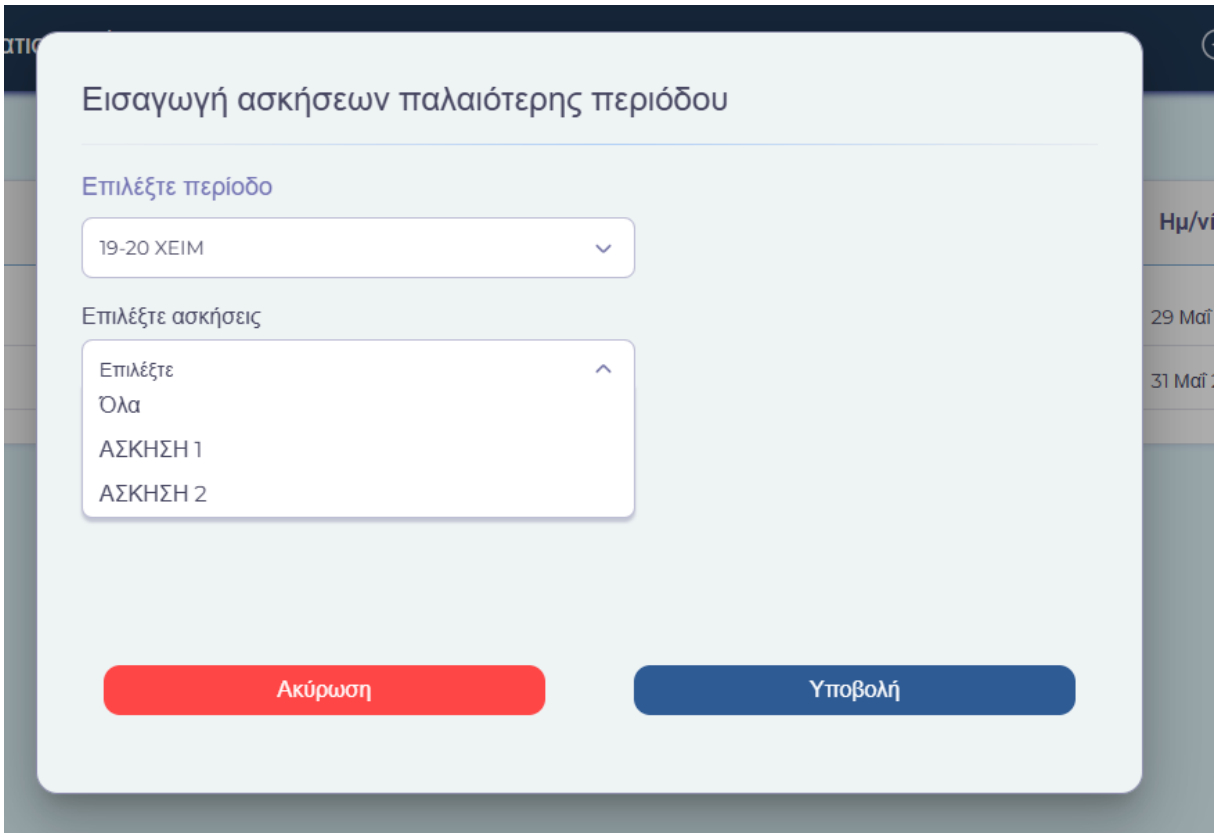


Εικόνα 3.19 Γραφική διεπαφή παραθύρου εισαγωγής παλαιότερων ασκήσεων

```
getPeriodsOfAvailableSubmissions(courseCode:number,period:Period):Observa
ble<Period[]>{
    return
this.http.get<Period[]>(`courses/${courseCode}/semesters/${period.semeste
r}/years/${period.year}/submission/periods`)
  }
```

Κώδικας 3.10 Μέθοδος getPeriodsOfAvailableSubmissions()

Αφού ο χρήστης επιλέξει την επιθυμητή περίοδο μέσω του μενού, το σύστημα προχωρά στην ανάκτηση των ασκήσεων που έχουν δημιουργηθεί για το συγκεκριμένο μάθημα κατά τη διάρκεια της επιλεγμένης περιόδου με την βοήθεια της getCourseSubmissions() του submissionService. Τα δεδομένα αυτά αποστέλλονται από τον διακομιστή και προβάλλονται στην εφαρμογή με τη μορφή ενός multi-select στοιχείου διεπαφής **multi-select component**. Μέσα από αυτό το component, ο χρήστης έχει τη δυνατότητα να επιλέξει μία ή περισσότερες από τις διαθέσιμες ασκήσεις, ανάλογα με το ποιες τον ενδιαφέρουν να μεταφέρει (Εικόνα 3.30).



Εικόνα 3.20 Γραφική διεπαφή παραθύρου εισαγωγής παλαιότερων ασκήσεων αφού έχει επιλεχθεί περίοδος

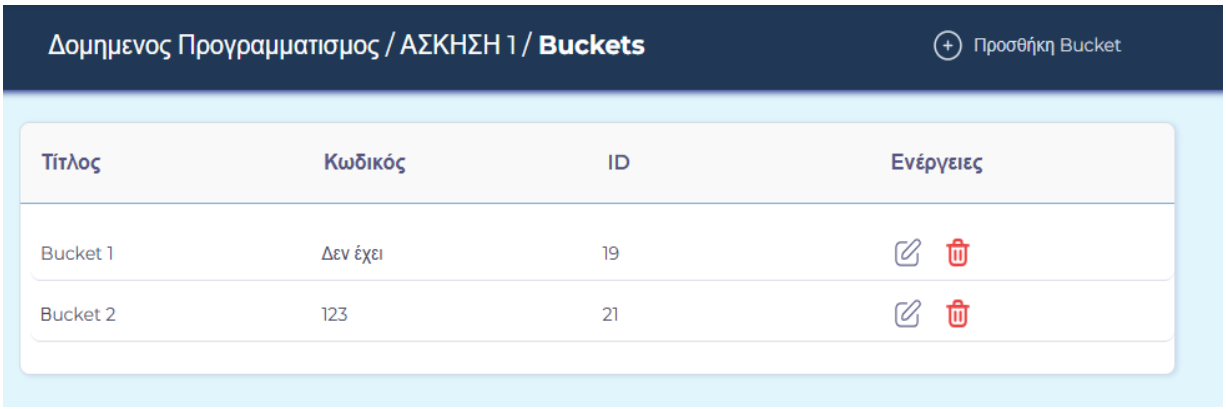
Η συγκεκριμένη λειτουργία συμβάλλει σημαντικά στην ευελιξία και την ταχύτητα διαχείρισης των ασκήσεων, καθώς απαλλάσσει τον διδάσκοντα από την επαναλαμβανόμενη και χρονοβόρα διαδικασία της χειροκίνητης δημιουργίας των ίδιων ασκήσεων για κάθε περίοδο. Με τη χρήση αυτής της δυνατότητας, οι ασκήσεις μπορούν να μεταφερθούν εύκολα από παλαιότερες περιόδους σε νέες. Κατά τη διαδικασία μεταφοράς, δεν μεταφέρονται μόνο οι ίδιες οι ασκήσεις αλλά και τα αντίστοιχα buckets που τις συνοδεύουν, εξαιρουμένων όμως των uploads που έχουν πραγματοποιήσει οι φοιτητές. Αυτό διασφαλίζει ότι το νέο περιβάλλον ξεκινά καθαρό, διατηρώντας όμως τη δομή και τις παραμέτρους των προηγούμενων ασκήσεων.

Όταν ο χρήστης πατήσει πάνω σε μία εγγραφή του πίνακα, μεταφέρεται στην επόμενη διαδρομή που είναι τα buckets της άσκησης.

3.3.4.3 Οθόνη buckets

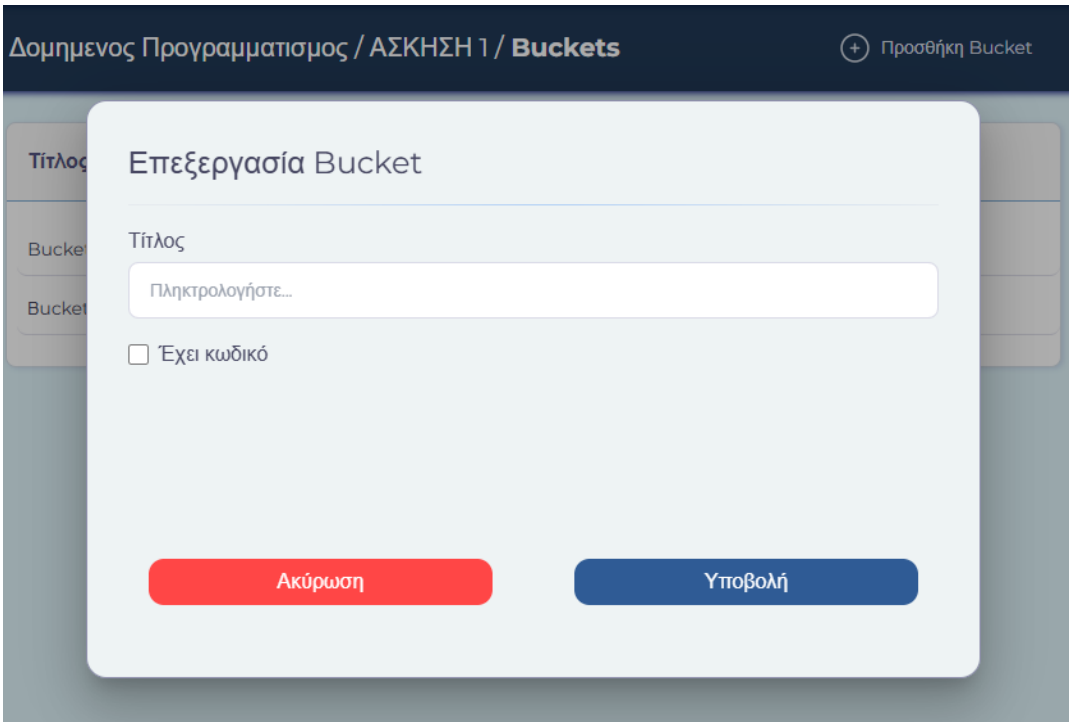
Στην οθόνη των buckets αποδίδεται το **buckets component** , το οποίο προσφέρει στον διδάσκοντα την δυνατότητα να διαχειρίζεται τα buckets στα οποία αποθηκεύονται τα uploads των φοιτητών.

Ειδικότερα, μέσω του bucketService και της μεθόδου getBuckets(), η οποία λαμβάνει σαν παράμετρο το id του submission, πραγματοποιείται επικοινωνία με τον διακομιστή και λαμβάνονται τα buckets που ανήκουν στην άσκηση που ζητήθηκε μέσω του submission id. Στον πίνακα που εμφανίζεται, προβάλλονται πληροφορίες για τα buckets όσον αφορά τον τίτλο, τον κωδικό και το id, το οποίο αποτελεί μοναδικό αναγνωριστικό (Εικόνα 3.31).

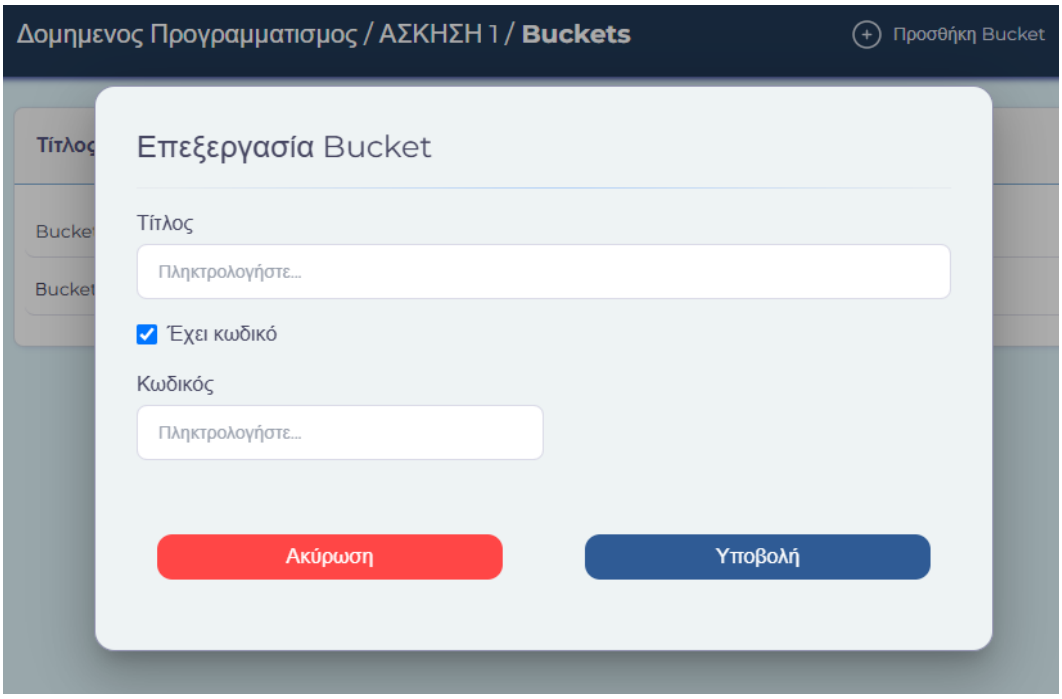


Εικόνα 3.21 Γραφική διεπαφή οθόνης του component buckets component

Στο πάτημα των ανάλογων κουμπιων, ανοίγει το παράθυρο **bucket-modal component** , το οποίο είναι μία φόρμα που ζητάει από τον χρήστη να συμπληρώσει πεδία όπως ο τίτλος του bucket και ένα checkbox που αντιπροσωπεύει την ύπαρξη κωδικού (Εικόνα 3.32). Αν το ενεργοποιήσει ο χρήστης, εμφανίζεται από κάτω ένα πεδίο εισαγωγής κωδικού, το οποίο αποτελεί τον κωδικό που θα κληθεί να συμπληρώσει ο φοιτητής όταν επιχειρήσει να υποβάλει μία άσκηση (Εικόνα 3.33). Με την υποβολή της φόρμας, καλείται η μέθοδος bucketAction() του bucket service,η οποία διαχειρίζεται δημιουργίες και επεξεργασίες bucket και στη συνέχεια ανανεώνονται τα δεδομένα με κλήση προς τον διακομιστή.



Εικόνα 3.22 Γραφική διεπαφή παραθύρου δημιουργίας bucket

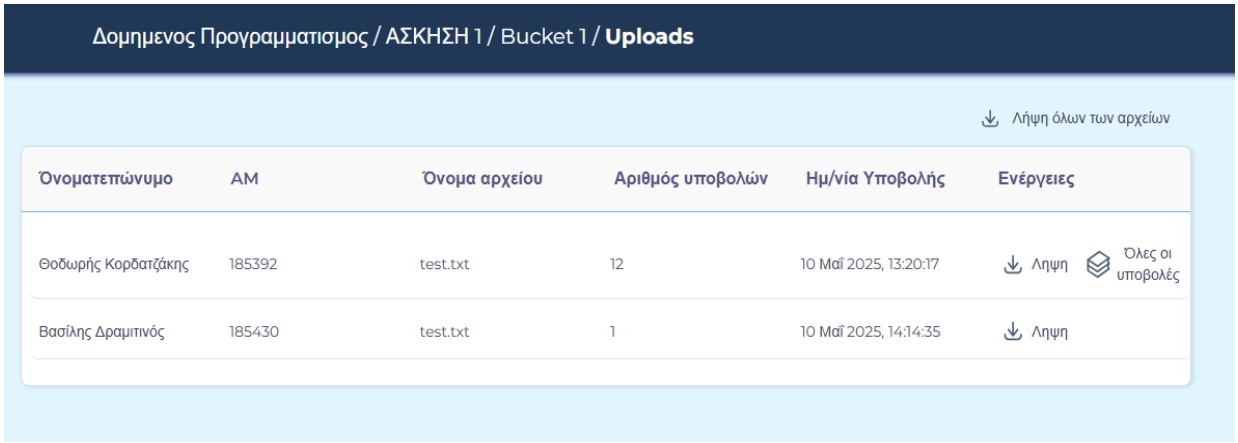


Εικόνα 3.23 Γραφική διεπαφή παραθύρου δημιουργίας bucket με ενεργοποιημένο το checkbox του κωδικού

Όταν ο χρήστης πατήσει πάνω σε μία εγγραφή του πίνακα, μεταφέρεται στην επόμενη διαδρομή που είναι τα uploads του bucket.

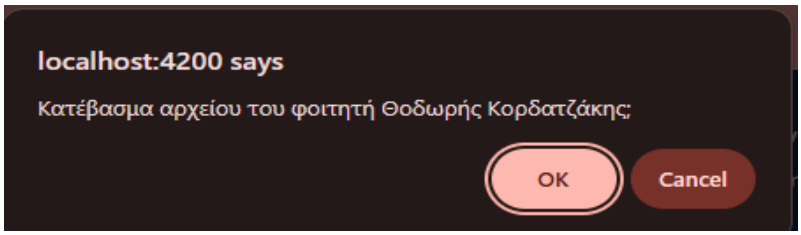
3.3.4.4 Οθόνη uploads

Όταν ο χρήστης μεταφερθεί στην διαδρομή των uploads, αναλαμβάνει το **uploads component** , το οποίο σε συνεργασία με το uploads service και τη μέθοδο getBucketUploads, περνώντας σαν παράμετρο το bucket id, λαμβάνονται από τον διακομιστή τα δεδομένα των uploads των φοιτητών για το bucket που ζητήθηκε. Σαν αποτέλεσμα εμφανίζεται ένας πίνακας που περιλαμβάνει πληροφορίες για τα uploads κάθε φοιτητή, όπως ονοματεπώνυμο φοιτητή, αριθμό μητρώου, ονόματα των αρχείων που ανέβασε, ημερομηνία και ώρα υποβολής και τον αριθμό των υποβολών που έκανε για την συγκεκριμένη άσκηση, ο οποίος δεν μπορεί να ξεπεράσει τον αριθμό των μέγιστων υποβολών που είχε καταχωρηθεί στο αντίστοιχο πεδίο της φόρμας δημιουργίας της άσκησης (Εικόνα 3.34).

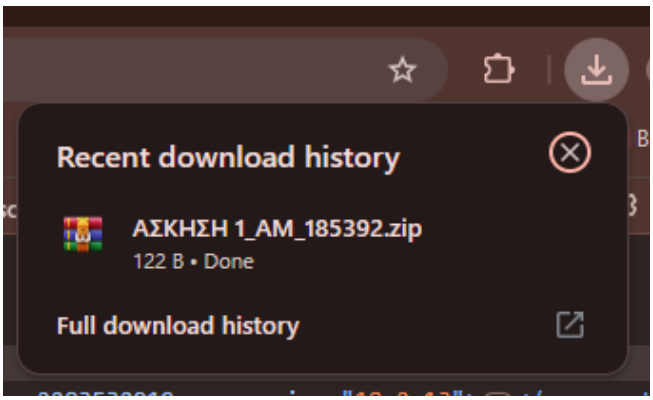


Εικόνα 3.24 Γραφική διεπαφή οθόνης του component uploads component

Στην κάθε εγγραφή του πίνακα, υπάρχει ένα κουμπί., το οποίο κατεβάζει στον υπολογιστή του χρήστη το περιεχόμενο του upload του φοιτητή σε μορφή .zip. Η διαδικασία που πραγματοποιείται είναι να συμπιέζονται τα αρχεία του upload στο backend σύστημα μέσω της downloadFile μεθόδου του upload service, η οποία ζητάει από το backend το συμπιεσμένο αρχείο. Με το που ληφθεί η απάντηση, δημιουργείται ένα προσωρινό URL για ένα αρχείο τύπου blob και την ενεργοποιείται η λειτουργία του browser να το κατεβάσει τοπικά. Ειδικότερα, Δημιουργείται ένα προσωρινό URL (τύπου blob) από το σώμα της απόκρισης (response.body), που περιέχει τα δεδομένα του αρχείου και δημιουργείται ένα στοιχείο τύπου <a> (άγκυρα/σύνδεσμος) και ορίζεται το href του να δείχνει στο blob URL που μόλις δημιουργήθηκε. Έπειτα, αποθηκεύεται η κεφαλίδα Content-Disposition, η οποία περιλαμβάνει το όνομα του αρχείου (filename) που στάλθηκε από τον διακομιστή. Αν υπάρχει όνομα αρχείου στην κεφαλίδα, εξάγεται με χρήση regular expression και αποκωδικοποιείται (σε περίπτωση που είναι σε UTF-8 ή περιέχει ειδικούς χαρακτήρες). Αν δεν βρεθεί, χρησιμοποιείται το προεπιλεγμένο όνομα αρχείου. Τέλος, προστίθεται προσωρινά ο σύνδεσμος στο DOM, "κλικάρεται" προγραμματιστικά για να ξεκινήσει η λήψη και στη συνέχεια αφαιρείται από το DOM και απελευθερώνεται η μνήμη καταργώντας το προσωρινό blob URL που είχε δημιουργηθεί νωρίτερα.

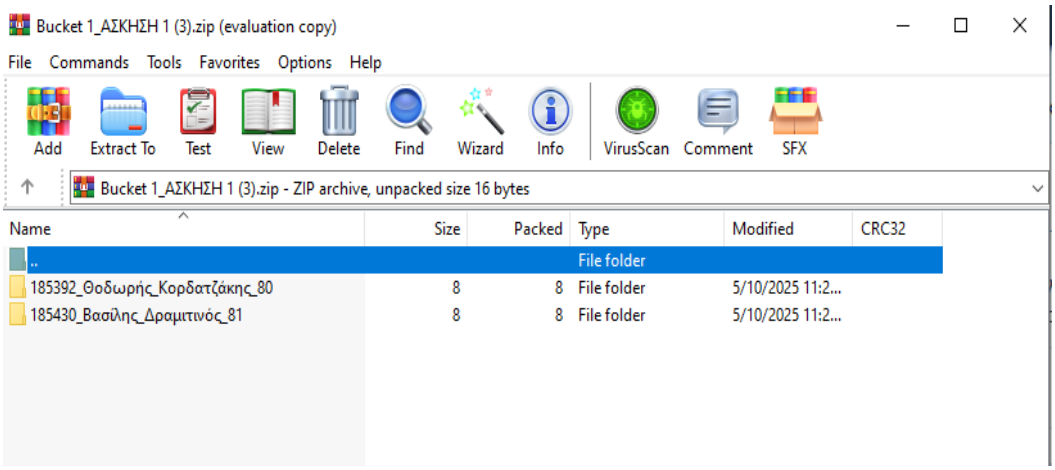


Εικόνα 3.25 Μήνυμα επιβεβαίωσης λήψης των αρχείων ενός φοιτητή



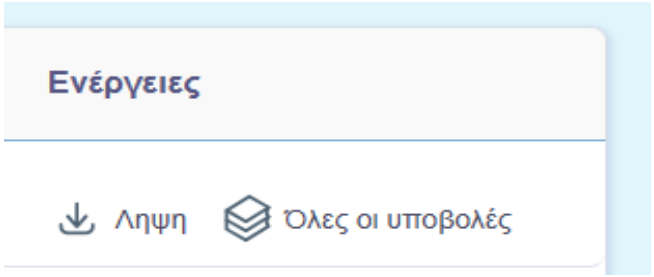
Εικόνα 3.26 Αρχείο λήψης της υποβολής του φοιτητή

Η ίδια λειτουργία πραγματοποιείται και στο πάτημα του αντίστοιχου κουμπιού για να γίνει λήψη όλων των uploads όλων των φοιτητών με την μόνη διαφορά ότι γίνεται κλήση σε διαφορετικό API endpoint του διακομιστή και το περιεχόμενο του ληφθέντος αρχείου αποτελείται από φακέλους, όπου κάθε φάκελος αντιπροσωπεύει τα αρχεία υποβολής του κάθε φοιτητή.



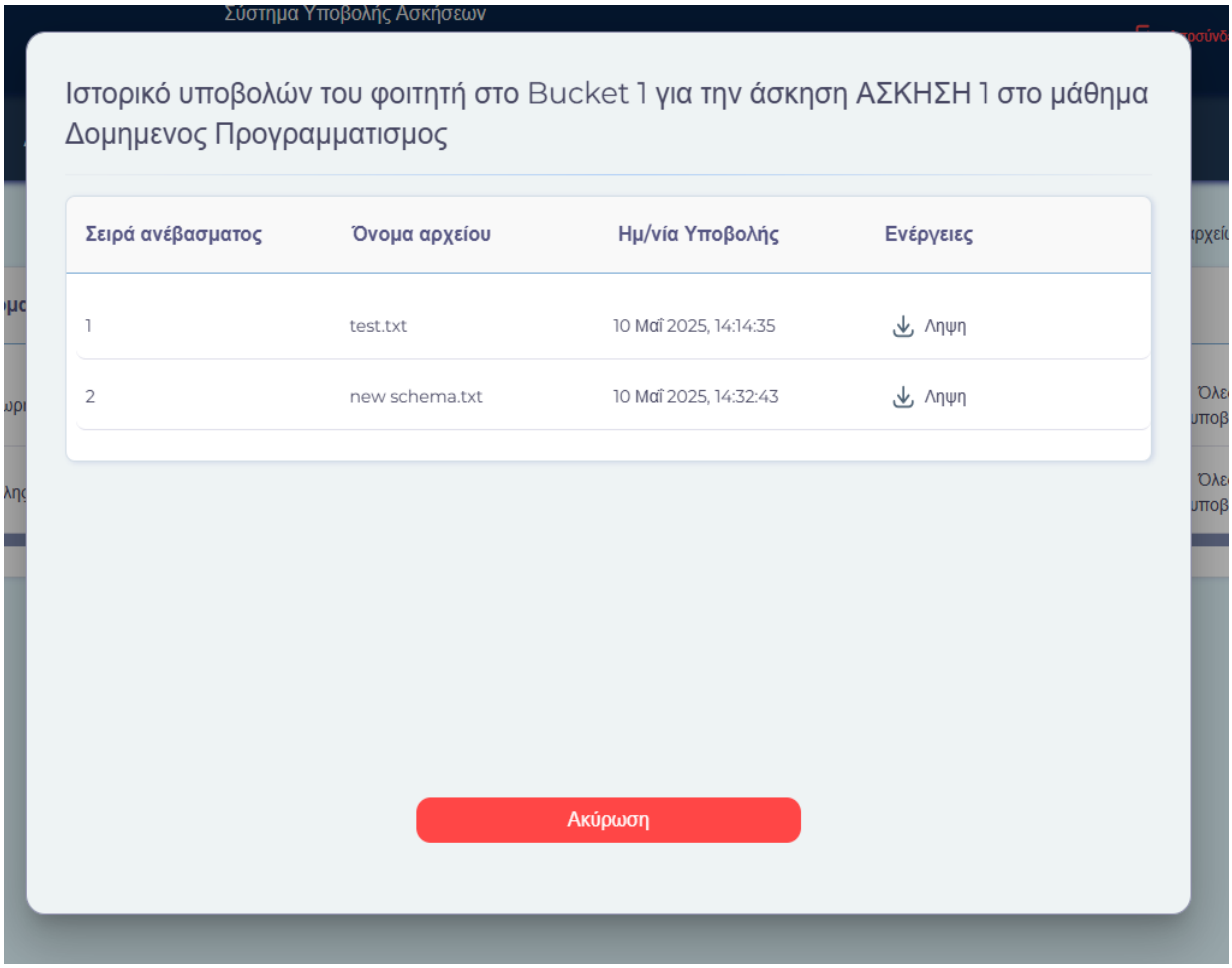
Εικόνα 3.27 Περιεχόμενο συμπιεσμένου αρχείου λήψης των αρχείων όλων των φοιτητών

Μια ακόμα σημαντική δυνατότητα στην οθόνη των uploads είναι η προβολή του ιστορικού υποβολών ενός φοιτητή για ένα συγκεκριμένο bucket, στην περίπτωση που έχει κάνει περισσότερες από μία υποβολές. Όταν διαπιστωθεί ότι υπάρχει ιστορικό πολλαπλών uploads, εμφανίζεται ένα επιπλέον κουμπί δίπλα από το κουμπί λήψης αρχείου (Εικόνα 3.38). Πατώντας το κουμπί αυτό, ανοίγει ένα αναδυόμενο παράθυρο που υλοποιείται μέσω του component **user-submission-uploads-modal.**



Εικόνα 3.28 Κουμπί για προβολή όλων των υποβολών του φοιτητή στην άσκηση

Στο άνοιγμα του παραθύρου, πραγματοποιείται εκ νέου αίτημα προς τον διακομιστή για λήψη των δεδομένων του ιστορικού του φοιτητή, μέσω της μεθόδου userBucketUploads του uploadsService, η οποία απαιτεί ως παράμετρο το αναγνωριστικό του φοιτητή και του bucket. Τα δεδομένα που επιστρέφονται εμφανίζονται με τη μορφή πίνακα, στον οποίο παρουσιάζονται όλες οι υποβολές που έχουν πραγματοποιηθεί ταξινομημένες με τη σειρά υποβολής. Επιπλέον, από τον πίνακα αυτό ο χρήστης έχει τη δυνατότητα να κατεβάσει όποιο από τα αρχεία επιθυμεί.



Εικόνα 3.29 Γραφική διεπαφή παραθύρου προβολής ιστορικού υποβολών του φοιτητή στην άσκηση

Σε αυτό το σημείο ολοκληρώνεται η παρουσίαση των λειτουργιών που προσφέρει το διαχειριστικό περιβάλλον του διδάσκοντα. Μέσα από αυτό το περιβάλλον, ο καθηγητής έχει στη διάθεσή του όλα τα απαραίτητα εργαλεία για τη δημιουργία, παρακολούθηση και διαχείριση μαθημάτων και ασκήσεων, καθώς και για την εποπτεία των υποβολών των φοιτητών, διευκολύνοντας έτσι την εκπαιδευτική διαδικασία.

Κεφάλαιο 4ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

4.1 Συμπεράσματα

Το σύστημα της πλατφόρμας υποβολής ασκήσεων που υλοποιήθηκε αποτελεί έναν πολύτιμο πόρο και εργαλείο του πανεπιστημίου, το οποίο στοχεύει στην διευκόλυνση των φοιτητών, καθώς και των καθηγητών στα πλαίσια της εκπαιδευτικής διαδικασίας. Είναι μια πρακτική λύση της διαδικασίας κατά την οποία οι φοιτητές υποβάλλουν τις ασκήσεις των εργαστηρίων τους ή τις ασκήσεις των εξετάσεων με τρόπο ασφαλή και εύκολο. Οι νέες δυνατότητες που προσφέρονται είναι σίγουρο πως θα βελτιώσουν σε μεγάλο βαθμό την εμπειρία του χρήστη και θα αποτελέσουν το βήμα για νέες μελλοντικές βελτιώσεις.

Η ανάπτυξη του διαχειριστικού περιβάλλοντος με Angular στο frontend και NestJS στο backend ήταν μια ευχάριστη και δημιουργική πρόκληση για εμάς. Μας έδωσε την ευκαιρία να κατανοήσουμε σε βάθος τη σωστή οργάνωση και διαχείριση δεδομένων, ειδικά όταν αυτά σχετίζονται μεταξύ τους έχοντας ως βάση τις σχέσεις που ορίζονται από τον διαχειριστή. Παράλληλα, η δημιουργία του περιβάλλοντος για τον φοιτητή αποτέλεσε πρόκληση από πλευράς σχεδιασμού και εμπειρίας χρήστη, καθώς θέλαμε να προσφέρουμε μια διεπαφή που να είναι φιλική, απλή στη χρήση και κατανοητή για τον τελικό χρήστη.

Μέσα από την ανάπτυξη της εφαρμογής κερδίσαμε εμπειρία τόσο σε τεχνικό όσο και σε σχεδιαστικό επίπεδο, ενισχύοντας τις δεξιότητές μας στον συντονισμό frontend και backend τεχνολογιών και στη δημιουργία μιας λειτουργικής και φιλικής διεπαφής χρήστη. Παρά τα θετικά αποτελέσματα, αναγνωρίζουμε ότι κάθε εφαρμογή μπορεί να εξελιχθεί περαιτέρω. Με αυτή την σκέψη, παραθέτουμε τις προτάσεις μας για βελτιώσεις, επεκτάσεις και εναλλακτικές λύσεις.

4.2 Μελλοντικές Επεκτάσεις

Για την περαιτέρω εξέλιξη της εφαρμογής, θα ήταν ωφέλιμο να εξεταστεί η πιθανότητα επέκτασης των παρακάτω λειτουργιών:

- **Χρονικό παράθυρο επεξεργασίας της υποβολής από τον φοιτητή**: Θα ήταν χρήσιμο να παρέχεται στον φοιτητή ένα χρονικό παράθυρο, μέσα στο οποίο θα έχει την δυνατότητα να μπορεί να επεξεργαστεί την υποβολή που έχει κάνει και στη λήξη του παραθύρου να υποβάλλεται οριστικά η υποβολή του. Αυτό θα γλιτώσει τον φοιτητή από το να πραγματοποιεί νέα υποβολή για να διορθώσει ένα λάθος που έχει κάνει, όπως το να έχει επιλέξει κατά λάθος άλλο μάθημα από αυτό που τον ενδιαφέρει ή να έχει ανεβάσει αρχείο το οποίο είναι ελλειπές και θέλει διορθώσεις. Θα έχει στην διάθεση του μια οθόνη ή έναν πίνακα στον οποίο θα φαίνονται όλες οι υποβολές που έχει κάνει, οι οποίες δεν έχουν ακόμα υποβληθεί οριστικά. Ο ορισμός του χρονικού παραθύρου θα γίνεται από το διαχειριστικό του καθηγητή.
- **Δυνατότητα βαθμολογίας ασκήσεων**: Μια λειτουργία που θα βοηθούσε ακόμα παραπάνω στην διαδικασία αξιολόγησης του φοιτητή θα ήταν να παρέχεται στον καθηγητή η δυνατότητα να βαθμολογήσει τις υποβολές που έχουν πραγματοποιήσει οι φοιτητές και να προσθέσει σχόλια σχετικά με προτάσεις βελτίωσης. Με την κατάθεση της βαθμολογίας, θα αποστέλλεται μήνυμα στο μειλ του φοιτητή με τις πληροφορίες της βαθμολογίας.
- **Οθόνη βαθμολογιών**: Η εμπειρία του χρήστη ως φοιτητής, θα ήταν πιο ολοκληρωμένη με την υλοποίηση μιας οθόνης στην οποία θα μπορούσε να βλέπει όλες τις υποβολές του, οι οποίες

έχουν βαθμολογηθεί από τον καθηγητή του. Ο χρήστης θα μπορεί να ταξινομήσει τις βαθμολογίες ανά ημερομηνια, μάθημα και βαθμό.

- **Ενημέρωση φοιτητή για την έναρξη και λήξη μιας υποβολής**: Κάτι που θα συνέβαλε στην ενημέρωση και εμπειρία του φοιτητή, θα ήταν η υλοποίηση μιας λειτουργίας που θα ενημερώνει τον φοιτητή την στιγμή που ανοίξει μία υποβολή. Αντίστοιχα, ο φοιτητής θα ενημερώνεται για την επικείμενη λήξη μιας υποβολής , δηλαδή ότι πρόκειται να κλείσει σε 24 ώρες για παράδειγμα.

Συνοψίζοντας, η εφαρμογή αποτελεί ένα πολύτιμο εργαλείο που συμβάλλει στην επιτυχημένη οργάνωση και διαχείριση της εκπαιδευτικής εμπειρίας των φοιτητών καθώς και των διδασκόντων του ΔΙ.ΠΑ.Ε.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Google, "*Angular Documentation: Overview"*, [Online]. Available: https://angular.dev/overview [Accessed: March 13, 2025].

[2] Google, "*Angular Documentation: Component Overview"*, [Online]. Available: https://v17.angular.io/guide/component-overview [Accessed: March 18, 2025].

[3] Google, "*Angular Documentation: Architecture – Services"*, [Online]. Available: https://v17.angular.io/guide/architecture-services [Accessed: April 2, 2025].

[4] Google, "*Angular Documentation: NgModules"*, [Online]. Available: https://v17.angular.io/guide/ngmodules [Accessed: April 2, 2025].

[5] Google, "*Angular Documentation: Directives"*, [Online]. Available: https://angular.dev/guide/directives [Accessed: April 5, 2025].

[6] Google, "*Angular Documentation: Routing Overview"*, [Online]. Available: https://v17.angular.io/guide/routing-overview [Accessed: April 6, 2025].

[7] Google, "*Angular API: CanActivate"*, [Online]. Available: https://angular.dev/api/router/CanActivate [Accessed: April 8, 2025].

[8] S. Astari, "What Is HTML? Hypertext Markup Language Basics Explained," *Hostinger Tutorials*, Apr. 28, 2025. [Online]. Available: https://www.hostinger.com/tutorials/what-is-html [Accessed: May 1, 2025].

[9] G. Domantas, "What Is CSS and How Does It Work?" *Hostinger Tutorials*, May 15, 2023. [Online]. Available: https://www.hostinger.com/ph/tutorials/what-is-css [Accessed: May 2, 2025].

[10] Motunrayo, "What is TypeScript?" *Hygraph Blog*, May 10, 2024. [Online]. Available: https://hygraph.com/blog/what-is-typescript [Accessed: May 2, 2025].

[11] Google, "*Angular Documentation: Inputs and Outputs"*, [Online]. Available: https://v17.angular.io/guide/inputs-outputs [Accessed: May 6, 2025].

[12] P. Lamb, "Angular Lifecycle Hooks," *Medium (ng-conf)*, Jun. 8, 2022. [Online]. Available: https://medium.com/ngconf/angular-lifecycle-hooks-977b73fc66af [Accessed: May 8, 2025].

[13] Google, "*Angular Documentation: Lifecycle Hooks"*, [Online]. Available: https://v17.angular.io/guide/lifecycle-hooks [Accessed: May 9, 2025].

[14] L. K. Singh, "Understanding Access Tokens and Refresh Tokens," *Medium*, Mar. 23, 2024. [Online]. Available: https://medium.com/@lakshyakumarsingh.2003.va/understanding-access-tokens-and-refresh-tokens-2ec4bc4f9a4f [Accessed: May 13, 2025].

[15] S. Kumar, "JWT Authentication in Nodejs — Refresh JWT with Cookie-based Token," *Medium*, Feb. 3, 2023. [Online]. Available: https://medium.com/@techsuneel99/jwt-authentication-in-nodejs-refresh-jwt-with-cookie-based-token-37348ff685bf [Accessed: May 14, 2025].

[16] The NestJS Team, "*NestJS Documentation: Overview"*, [Online]. Available: https://docs.nestjs.com/ [Accessed: March 15, 2025].

[17] The NestJS Team, "*NestJS Documentation: Modules"*, [Online]. Available: https://docs.nestjs.com/modules [Accessed: March 20, 2025].

[18] The NestJS Team, "*NestJS Documentation: Controllers"*, [Online]. Available: https://docs.nestjs.com/controllers [Accessed: March 25, 2025].

[19] The NestJS Team, "*NestJS Documentation: Providers"*, [Online]. Available: https://docs.nestjs.com/providers [Accessed: April 2, 2025].

[20] The NestJS Team, "*NestJS Documentation: Middleware"*, [Online]. Available: https://docs.nestjs.com/middleware [Accessed: April 4, 2025].

[21] The NestJS Team, "*NestJS Documentation: Guards"*, [Online]. Available: https://docs.nestjs.com/guards [Accessed: April 8, 2025].

[23] "Dependency injection," *Wikipedia, The Free Encyclopedia*, [Online]. Available: https://en.wikipedia.org/wiki/Dependency_injection [Accessed: April 10, 2025].

[24] The NestJS Team, "*NestJS Documentation: Exception Filters"*, [Online]. Available: https://docs.nestjs.com/exception-filters [Accessed: April 15, 2025].

[25] The NestJS Team, "*NestJS Documentation: Validation Techniques"*, [Online]. Available: https://docs.nestjs.com/techniques/validation [Accessed: April 17, 2025].

[26] The NestJS Team, "*NestJS Documentation: OpenAPI Introduction"*, [Online]. Available: https://docs.nestjs.com/openapi/introduction [Accessed: April 25, 2025].

[27] "Node.js," *Wikipedia, The Free Encyclopedia*, [Online]. Available: https://en.wikipedia.org/wiki/Node.js [Accessed: May 2, 2025].

[28] npm, Inc., "About npm," *npm Docs*, [Online]. Available: https://docs.npmjs.com/about-npm [Accessed: May 4, 2025].

[29] Mozilla Contributors, "Package management basics," *MDN Web Docs*, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Client-side_tools/Package_management [Accessed: May 4, 2025].

[30] S. Sidorarev, "*mysql2 Documentation"*, [Online]. Available: https://sidorares.github.io/node-mysql2/docs [Accessed: May 10, 2025].

[31] D. Stuk, "*JSZip Documentation"*, [Online]. Available: https://stuk.github.io/jszip/ [Accessed: May 15, 2025].

[32] D. Stuk, "J file(name, data [,options])," *JSZip Documentation*, [Online]. Available: https://stuk.github.io/jszip/documentation/api_jszip/file_data.html [Accessed: May 20, 2025].

[33] H. Gunawan, "The Cost of Poor Database Design," *CoderBased*, Apr. 26, 2023. [Online]. Available: https://www.coderbased.com/p/the-cost-of-poor-database-design [Accessed: May 24, 2025].

[34] "MySQL," *Wikipedia, The Free Encyclopedia*, May 21, 2025. [Online]. Available: https://en.wikipedia.org/wiki/MySQL [Accessed: May 18, 2025].

[35] "Relationships in SQL: One to One, One to Many, Many to Many," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/relationships-in-sql-one-to-one-one-to-many-many-to-many/ [Accessed: May 22, 2025].