ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σύστημα Δήλωσης Εργαστηριακών Τμημάτων»

Των φοιτητών

Μαυρομάτη Σταύρου

Αρ. Μητρώου: 174896

Κύρκου Πέτρου

Αρ. Μητρώου: 164692

Επιβλέπων

Ονοματεπώνυμο

Αμανατιάδης Δημήτριος

Ημερομηνία 10/09/2023

Τίτλος Π.Ε. Σύστημα Δήλωσης Εργαστηριακών Τμημάτων

Κωδικός Π.Ε. 22275

Ονοματεπώνυμο φοιτητή/των Μαυρομάτης Σταύρος, Κύρκος Πέτρος

Ονοματεπώνυμο εισηγητή Αμανατίδης Δημήτριος

Ημερομηνία ανάληψης Π.Ε. 21/10/2022

Ημερομηνία περάτωσης Π.Ε. 10/09/2023

Βεβαιώνουμε ότι είμαστε οι συγγράφεις 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 πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η επιλογή αυτού του θέματος Πτυχιακής εργασίας προέκυψε μετά από συζήτηση με τον συντονιστή του θέματος, κατά την οποία διαπιστώθηκε η ύπαρξη ενός σημαντικού προβλήματος στις δηλώσεις των Εργαστηρίων του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Μετά από μια εκτενή ανάλυση του θέματος, καταλήξαμε σε μια λύση μέσω της υλοποίησης μιας εφαρμογής στο πλαίσιο της Πτυχιακής εργασίας.

Η συγκεκριμένη εφαρμογή έχει ως στόχο να παρέχει λύσεις που αυτή τη στιγμή δεν υπάρχουν στο τμήμα, με σκοπό την ευκολία και τη βελτίωση της εκπαιδευτικής διαδικασίας.

Μέσα από αυτήν την πτυχιακή εργασία, θα επιτευχθεί η ανάπτυξη μιας λειτουργικής εφαρμογής που θα προσφέρει πρακτικές λύσεις για τα προβλήματα που αντιμετωπίζουν οι εκπαιδευτικοί.

Συνοψίζοντας, η επιλογή αυτού του θέματος Πτυχιακής εργασίας πηγάζει από την ανάγκη επίλυσης του προβλήματος στις δηλώσεις των Εργαστηρίων. Με την υλοποίηση της εφαρμογής, στοχεύετέ να βελτιωθεί η διαδικασία εκπαίδευσης

Περίληψη

Η Πτυχιακή αυτή εργασία προέκυψε από την ανάγκη του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων για τη δημιουργία μιας ενιαίας πλατφόρμας δηλώσεων Εργαστηριακών Τμημάτων. Παρατηρήθηκε ότι κατά την έναρξη κάθε εξαμήνου, οι καθηγητές χρησιμοποιούσαν διάφορες πλατφόρμες τρίτων, όπως το Google Drive ή το Moodle. Ο στόχος της εφαρμογής είναι να δημιουργηθεί ένα ολοκληρωμένο σύστημα, όπου θα βρίσκονται όλες οι πληροφορίες για όλα τα εργαστηριακά μαθήματα, χωρίς την ανάγκη για εξωτερικές λύσεις εκτός του τμήματος. Η πλατφόρμα αντιμετωπίζει αρκετά προβλήματα, όπως την ευκολία δημιουργίας και διαχείρισης μαθημάτων, ακόμη και κατά την περίοδο δηλώσεων, και παρέχει εύκολη πρόσβαση σε πληροφορίες για την πληρότητα των εργαστηριακών τμημάτων ανά μάθημα, καθώς και για τους φοιτητές που τα έχουν δηλώσει, μέσω εξαγωγής σε αρχείο λογιστικού φυλλου (excel).

Από την πλευρά των φοιτητών, παρέχεται ένα εύκολο περιβάλλον για τις δηλώσεις μαθημάτων, με έναν αλγόριθμο δικαιοσύνης στην προτεραιότητα των δηλώσεων. Η πλατφόρμα διαχωρίζει τους φοιτητές ανάλογα με την πρόοδό τους στις δηλώσεις και τη θεωρία, παρέχοντας μια καθαρή λίστα με τους φοιτητές που έχουν δικαίωμα να παρακολουθήσουν το εργαστήριο. Επιπλέον, παρέχεται η δυνατότητα εμφάνισης και αποθήκευσης του προγράμματος εργαστηρίων σε μορφή PDF, παραμετροποιώντας το ανάλογα με τον ρόλο του χρήστη. Έτσι, ένας φοιτητής θα βλέπει τα μαθήματα, τις ώρες και τις ημέρες που έχει δηλώσει εργαστήρια, ενώ ένας καθηγητής θα βλέπει τα μαθήματα που έχει να διδάξει και τις αντίστοιχες ημέρες/ώρες.

Η περίοδος δηλώσεων ξεκινά από τον διαχειριστή της εφαρμογής και μπορεί να λήξει αυτόματα μετά την παρέλευση της προκαθορισμένης ημερομηνίας ή χειροκίνητα.

«Laboratory Departments Submission System»

«Stavros Mavromatis»

«Petros Kyrkos»

Abstract

This thesis is focused on addressing the Department of Computer Engineering and Electronic Systems' need for a comprehensive platform for lab department submissions. Currently, faculty members utilize various third-party platforms like Google Drive, Google Docs and Moodle at the beginning of each semester to achieve their goal successfully. The objective of this application is to create a unified system that eliminates the reliance on the previously mentioned external solutions, allowing all lab course information and their submissions to be centralized within the department.

The platform resolves multiple challenges, including streamlined course creation and management, during the registration period or not. It provides easy access to information regarding the completeness of each lab section per course, as well as a comprehensive list of students who have registered, which can be exported to a .xlsx file.

For students, the platform offers a user-friendly environment for declaring the submission of their choice in a Laboratory course, implementing a customly designed algorithm to prioritize registrations. At the end of each declaration period, a clear list is being generated that contains only eligible students who can attend the course.

Furthermore, the platform enables users to view and optionally save their workshop schedules as a custom PDF file. The schedule is customizable based on the user's role. Students can view their submitted lab departments of choice, along with the corresponding times and days, while teachers can access information on the courses they are assigned, including the associated days and times.

The registration period is initiated by the application administrator and can either automatically end on a specified date or be concluded on demand.

By implementing this platform, the Department of Computer Engineering and Electronic Systems aims to enhance the efficiency and accessibility of the submissionaly periods, providing a centralized and user-friendly solution for both faculty members and students.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Α.Δ.Μ.Φ	Αριθμός Διδακτικών Μονάδων Φοιτητή
Α.Μ	Αριθμός Μητρώου
Α.Τ.Ε.Ι.Θ	Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλονίκης
Δ.Μ	Διδακτικές Μονάδες
Δ.Π.	Δείκτης Προόδου
ΔΙ.ΠΑ.Ε	Διεθνές Πανεπιστήμιο Ελλάδος
Ε.Φ.	Εξάμηνο Φοίτησης
Μ.Ε.Π.Δ.Δ.Μ.Α.Ε	Μέγιστες Επιτρεπόμενες Προς Δήλωση Δ.Μ. Ανά Εξάμηνο
Μ.Σ.Δ.Μ.Ε.	Μέγιστες Συμπληρωμένες Διδακτικές Μονάδες Εξαμήνου
Π.Ε.	Πτυχιακή Εργασία
Ρ.Π.	Ρυθμός Προόδου
API	Application Programming Interface
CSS	Cascading Style Sheets
CSV	Comma Separated Values
DI	Dependency Injection
DIC	Dependency Injection Container
DOM	Document Object Model
DTO	Data Transfer Object
EF Core	Entity Framework Core
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
HTTPS	Hyper-Text Transfer Protocol Secure
IDE	Integrated Development Environment
JS	JavaScript
JSON	JavaScript Object Notation
JWT	JavaScript Web Token

MVC	Model-View-Controller
PDF	Portable Document Format
RDBMS	Relational DataBase Management System
SPA	Single Page Application
SQL	Structured Query Language
UI	User Interface

Κεφάλαιο 1ο: Εισαγωγή Στο Πρόβλημα

1.1 Εισαγωγή

Η παρούσα πτυχιακή εργασία παρουσιάζει μια λύση για ένα σημαντικό πρόβλημα που αντιμετωπίζεται στο τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων καθώς και παλαιότερα το Τμήμα Μηχανικών Πληροφορικής που υπάγονταν στο πρώην Αλεξάνδρειο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Θεσσαλονίκης Το πρόβλημα αυτό αφορά τη διαδικασία εγγραφής των φοιτητών στα Εργαστηριακά Μαθήματα.

1.2 Ιστορική Αναδρομή Δηλώσεων Εργαστήριών

Πριν την Πλατφόρμα Uniportal και την ένταξη του πρώην Α.Τ.Ε.Ι.Θ στο ΔΙ.ΠΑ.Ε οι δηλώσεις για τα μαθήματα ως σύνολα γίνονταν μέσω ενός συστήματος με το όνομα ΠΥΘΙΑ. Αν και αυτό το σύστημα προσέφερε καινοτόμες λύσεις και δυνατότητες την τότε εποχή για τη δήλωση θεωρητικών και εργαστηριακών μαθημάτων ανά εξάμηνο φοίτησης, αντιμετώπιζε κρίσιμα προβλήματα που δεν επιλύθηκαν ποτέ.

Ένα από τα βασικά προβλήματα ήταν η υπερβολική χρήση που καλούταν να εξυπηρετήσει μέσω πολλαπλών ταυτόχρονων συνδέσεων που υπήρχε στις περιόδους δηλώσεων με αποτέλεσμα την υπερφόρτωση του συστήματος, καθώς δέχονταν έναν μεγάλο όγκο αιτήσεων από φοιτητές. Ας σημειωθεί ότι η περίοδος δηλώσεων ξεκινούσε ταυτόχρονα για κάθε Τμήμα που στεγαζότανε στο Α.Τ.Ε.Ι.Θ. Η αίτηση του κάθε φοιτητή, που μπορεί να ανήκει σε οποιοδήποτε τμήμα, εξυπηρετούνταν από έναν φυσικό διακομιστή και συνεπώς αυτό οδηγούσε σε καθυστερήσεις και ανεπάρκεια απόκρισης, προκαλώντας προβλήματα τόσο στους Φοιτητές όσο και στα Τμήματα γενικότερα. Οι φοιτητές συχνά αντιμετώπιζαν προβλήματα χρόνου και κατέληγαν σε Timeout Exceptions, χάνοντας όλες τις δηλώσεις και τα εργαστηριακά τμήματα που είχαν επιλέξει μέχρι εκείνη τη στιγμή. Σε αυτή την περίπτωση, ο φοιτητής ήταν αναγκασμένος να δηλώσει ξανα όλα τα μαθήματα με τον κίνδυνο να μην βρει θέση στα εργαστηριακά τμήματα που επιθυμούσε καθώς και να υποστεί δεύτερο Timeout λόγω του υπερβολικού φόρτου.

Πραγματοποιήθηκαν μερικές προσπάθειες βελτίωσης του συστήματος, αλλά αυτές δεν ήταν επαρκείς για μια τόσο απαιτητική διαδικασία. Ένας από τους τρόπους βελτίωσης ήταν η διαίρεση των δηλώσεων σε χρονικές υποκατηγορίες ανάλογα με έναν συντελεστή προόδου που προσπαθούσε να ποσοτικοποιήσει το ποσοστό επιτυχίας φοίτησης του κάθε φοιτητή μέχρι την δεδομένη στιγμή σε μορφή ακέραιου δείκτη, γνωστός στο πρώην Α.Τ.Ε.Ι.Θ ως Δείκτης Προόδου. Η ιδέα πίσω από αυτή την υλοποίηση ήταν η επιτροπή δηλώσεων να επιτρέψει σε όλους τους φοιτητές να υποβάλουν τα θεωρητικά μαθήματα όποτε επιθυμούν, αλλά να εισέρχονται σε σειρά με βάση τον Δείκτη Προόδου τους για να δηλώσουν τα εργαστηριακά τμήματα που επιθυμούσαν. Έτσι, όταν ένας φοιτητής έκανε μεγάλη προσπάθεια και περνούσε ένα μεγάλο ποσοστό μαθημάτων, του δίνονταν προτεραιότητα για να δηλώσει στο πρώτο γκρουπ και ούτο καθεξής για τις υπόλοιπες προτεραιότητες. Με αυτόν τον τρόπο, ο φοιτητής είχε κίνητρο να επιτύχει στα μαθήματα και παράλληλα μειωνόταν η φόρτιση του συστήματος κατά τρεις φορές, λόγω των τριών γκρουπ που υπήρχαν. Παρόλα αυτά, ακόμα υπήρχαν προβλήματα με την γενικότερη υπηρεσία του ΠΥΘΙΑ (παλιό λογισμικό, μη υπάρχουσα συντήρηση, μη επάρκεια αρκετών ισχυρών διακομιστών που μπορούσαν να υποστηρίξουν τον φόρτο κατά την διάρκεια δηλώσεων κ.α.) που δεν λύθηκαν με την ύπαρξη του Δείκτη Προόδου, γεγονός που ανάγκασε τους διδάσκοντες να αναζητήσουν άλλες καλύτερες λύσεις.

Με το σχέδιο της ένταξης του πρώην Α.Τ.Ε.Ι.Θ. στο ΔΙ.ΠΑ.Ε. και για να αντιμετωπιστούν αυτά τα προβλήματα και να βελτιωθεί το σύστημα, εισήχθη η πλατφόρμα Uniportal. Το Uniportal ήταν ένα νέο σύστημα, ενιαίο για όλα τα τμήματα του Πανεπιστημίου σε όλες τις πόλεις της Ελλάδας. Κατασκευάστηκε από Τρίτο πάροχο για την διαχείριση των δηλώσεων που αντικατέστησε πλήρως το ΠΥΘΙΑ με την ολοκλήρωση του σχεδίου ένταξης στο ΔΙ.ΠΑ.Ε. Αυτό το νέο σύστημα προσέφερε πλήρη υποστήριξη για τις δηλώσεις θεωρητικών και εργαστηριακών μαθημάτων ανά εξάμηνο φοίτησης και περίοδο μαθημάτων. Επιπλέον, προσφέρεται βελτιωμένη απόδοση και αντοχή στο σύστημα, καθώς είχε σχεδιαστεί για να ανταπεξέλθει σε μεγάλο αριθμό φοιτητών και να αποφεύγει τα προβλήματα που παλαιότερα προκαλούσε το ΠΥΘΙΑ.

Με την εισαγωγή της πλατφόρμας Uniportal, οι φοιτητές είχαν πλέον ένα αξιόπιστο σύστημα για τις δηλώσεις τους, χωρίς τον κίνδυνο να χάσουν αυτές ή τα εργαστηριακά τμήματα που είχαν επιλέξει λόγω προβλημάτων που κάποτε υπήρχαν. Επίσης η νέα πλατφόρμα προσέφερε βελτιωμένη εμπειρία χρήσης και περισσότερες λειτουργίες, που διευκόλυναν τους φοιτητές στη διαδικασία των δηλώσεων μαθημάτων.

Συνολικά, η εισαγωγή του Uniportal βελτίωσε σημαντικά τη διαδικασία δηλώσεων μαθημάτων και επέλυσε μερικά προβλήματα που αντιμετώπιζαν οι φοιτητές και το Τμήμα. Ωστόσο στο Uniportal όντως εξωτερικό λογισμικό Τρίτου δεν μπορούσε να εφαρμοστεί ένας αντίστοιχος αλγόριθμος όπως αυτός του Δείκτη Προόδου για την σωστή κατανομή των φοιτητών με βάση κάποια κριτήρια. Οπότε ενώ το Uniportal κρατήθηκε για τις δηλώσεις θεωρίας, οι δηλώσεις των εργαστηρίων προτιμήθηκε να διεξάγονται με διαφορετικό τρόπο και με προσωρινές λύσεις για το κάθε εργαστηριακό μάθημα από κάθε διδάσκοντα.

Στο Τμήμα, η διαδικασία εγγραφής των φοιτητών στα Εργαστηριακά Μαθήματα πραγματοποιείται με τρόπο που απαιτεί σημαντικό χρόνο και προσπάθεια. Οι φοιτητές πρέπει να υποβάλλουν ένα ηλεκτρονικό αρχείο εγγραφής, το οποίο περιλαμβάνει πληροφορίες όπως το επιθυμητό Εργαστηριακό Τμήμα, τον αριθμό μητρώου τους και το εξάμηνο σπουδών τους. Κατόπιν, οι αιτήσεις αξιολογούνται με βάση διάφορα κριτήρια, όπως ο αριθμός μητρώου, οι προηγούμενες αποτυχημένες προσπάθειες παρακολούθησης και οι διαθέσιμες θέσεις στα Εργαστηριακά Τμήματα της επιλογής του φοιτητή. Τέλος, οι φοιτητές ενημερώνονται ασύγχρονα για την αποδοχή ή μη της αίτησής τους μετά την οριστικοποίηση αποστολής.

Οι διδάσκοντες κάνουν χρήση διαφόρων τεχνολογιών για τη διαχείριση και την διεκπεραίωση των αιτήσεων αυτών. Αυτές περιλαμβάνουν τη χρήση ηλεκτρονικών φορμών για τη συλλογή των πληροφοριών από τους φοιτητές, τη χρήση βάσεων δεδομένων για την αποθήκευση και ανάκτηση των δεδομένων των φοιτητών και τη χρήση αλγορίθμων αξιολόγησης για την όσο το δυνατόν αυτοματοποίηση της διαδικασίας αξιολόγησης των αιτήσεων.

1.3 Προσωρινές Λύσεις / Προβλήματα

Με την ένταξη στο ΔΙ.ΠΑ.Ε πραγματοποιήθηκαν αλλαγές σε πολλαπλούς κανονισμούς. Έτσι πραγματοποιήθηκε και αλλαγή στον κανονισμό που εφαρμόζεται στην περίπτωση που ένας φοιτητής αποτύχει σε ένα Μικτό Μάθημα ως σύνολο. Τόσο στον καιρό του Α.Τ.Ε.Ι.Θ όσο και τώρα τα μαθήματα χωρίζονται σε Απλά Θεωρητικά και σε Μικτά Μαθήματα. Το Μικτό Μάθημα χωρίζεται σε 2 διακριτά μέρη (το θεωρητικό και το εργαστηριακό). Ο κανόνας που ίσχυε στο πρώην Α.Τ.Ε.Ι.Θ ήταν ότι εάν ο φοιτητής πετύχαινε σε ένα εκ των δύο μερών του Μαθήματος μπορούσε να κατοχυρωθεί ο βαθμός του στο μέρος που επέτυχε και να γίνεται προσπάθεια εκ νέου μόνο στο μέρος που είχε αποτύχει

πρωταρχικά. Κανονισμός που πλέον δεν ισχύει. Από την ολοκλήρωση της ένταξης και πέρα ο βαθμός δεν κατοχυρώνεται και η αποτυχία σε ένα μέρος του μαθήματος υποχρεώνει τον φοιτητή να επαναλάβει την διαδικασία εξέτασης στο Μάθημα ως σύνολο ακόμη και στο μέρος του όπου είχε επιτύχει.

Ανέκαθεν η διαδικασία δηλώσεων μαθημάτων χωριζόταν σε 2 διακριτά μέρη. Στις δηλώσεις θεωρίας και σε αυτές του εργαστηρίου. Η διαφορά των δύο είναι ότι ενώ οι δηλώσεις θεωρίας ήταν και είναι ανοικτές για όλους χωρίς την επιβάρυνση ορισμένων θέσεων τα εργαστηριακά τμήματα δεν ακολουθούν την ίδια φιλοσοφία καθώς τόσο οι εργαστηριακές θέσεις που έχει το κάθε Τμήμα όσο και το διδακτικό προσωπικό είναι περιορισμένο.

Ιστορικά και προς καλύτερη κατανόηση από τον αναγνώστη αναφέρεται ότι οι προεγγεγραμμένοι φοιτητές (Α Τυπικού Εξαμήνου) είχαν πάντα μια δεσμευμένη θέση σε ένα εργαστηριακό τμήμα του τυπικού εξαμήνου τους προσυμπληρωμένα κατά αλφαβητική σειρά χωρίς να προβούν οι ίδιοι σε κάποια δήλωση. Όσοι φοιτητές μεγαλύτερων εξαμήνων ωστόσο είχαν αποτύχει και έπρεπε να επαναλάβουν το εργαστήριο και να ξεκινήσουν αναγκαστικά εκ νέου την διαδικασία της δήλωσης εργαστηριακών τμημάτων δεν μπορούσαν να δεσμεύσουν μια θέση έναντι αυτών που ήταν προεγγεγραμμένοι (και λόγω του ότι οι διαθέσιμες θέσεις ήταν λίγες) με αποτέλεσμα να μην καταλαμβάνουν μια θέση σε εργαστηριακό μάθημα όπου είχαν αποτύχει μόνο μία φορά και με απώτερο αποτέλεσμα οι σπουδές αυτών να μένουν πίσω καθώς έπρεπε να περιμένουν το νέο ακαδημαϊκό έτος για την επανεκκίνηση της διαδικασίας.

Με την διακοπή λειτουργίας του ΠΥΘΙΑ και για την αντιμετώπιση των παραπάνω προβλημάτων οι καθηγητές άρχισαν να κατασκευάζουν δικές τους λύσεις, με αποτέλεσμα άθελα τους να επιβαρύνουν την κατάσταση. Ο κάθε διδάσκων ανέπτυσσε τον δικό του τρόπο καταχώρησης των προτιμήσεων των φοιτητών για δήλωση εργαστηριακού τμήματος. Κάποιοι από αυτούς προέβησαν σε χρήση φόρμας τύπου Google Forms και Moodle και για ορισμένους άλλους, οι οποίοι είχαν περισσότερο χρόνο και πόρους στη διάθεσή τους, ένα απλουστευμένο νέο σύστημα προσαρμοσμένο και αφοσιωμένο μόνο στις εργαστηριακές δηλώσεις των μαθημάτων τους. Ένα πρόβλημα ακόμη που γεννήθηκε και ειδικότερα για τη χρήση των Google Forms και σε περιπτώσεις που υπήρχε η δυνατότητα επιλογής περισσότερων του ενός τμημάτων ως προτίμηση από τους φοιτητές, αλλά όχι μόνο, ο αλγόριθμος που ανέπτυσσε ο κάθε συντονιστής καθηγητής, ώστε να κατανέμει τους φοιτητές στις προτιμήσεις τους, δεν ήταν τις περισσότερες φορές δίκαιος και πόσο μάλλον κοινός με αλγορίθμους άλλων συντονιστών με αρνητικά αποτέλεσμα τόσο από πλευράς φοιτητών όσο και από πλευράς καθηγητών.

Επιπλέον, σε περίπτωση που είχε ανακοινωθεί το πρόγραμμα των εργαστηριακών τμημάτων για επιλογή εργαστηριακών τμημάτων, οι φοιτητές δεν διάλεγαν διαφορετική μέρα και ώρα εργαστηρίου όταν υπήρχε η δυνατότητα επιλογής περισσότερων της μιας προτίμησης. Αυτό έφερε ως αποτέλεσμα να δημιουργούνται εμβόλιμα τμήματα που διδάσκονταν ταυτόχρονα σε διαφορετικές αίθουσες και μόνο από έναν διδάσκων (τις περισσότερες φορές) την ίδια μέρα και ώρα με αποτέλεσμα να μην ικανοποιούνται πρώτον (1), όλοι οι φοιτητές/τριες στις προτιμήσεις τους, παρόλο που ενδέχεται να είχαν δηλώσει από τους πρώτους το εργαστηριακό τμήμα της επιλογής τους και τελικά να κατέληγαν σε κάποιο εμβόλιμο καθώς και δεύτερον (2) οι διδάσκοντες καθώς έκαναν το καλύτερο δυνατό για την εξυπηρέτηση όλων.

Τα προβλήματα εμμέναν και αυτές οι εναλλακτικές λύσεις δεν ήταν επίσημες ή δομημένες έτσι ώστε να εξασφαλίζουν την εύκολη ανάκτηση των πληροφοριών χωρίς περιττή απώλεια χρόνου και προσπάθειας από τον κάθε διδάσκων αλλά και φοιτητή καθώς και ανανέωση και επικαιροποίηση των στοιχείων κάθε εξάμηνο.

Ακόμη ένα πρόβλημα στο οποίο δεν υπήρχε η δυνατότητα λύσης ούτε μέσω του νέου συστήματος Uniportal και χαρακτηρίστηκε ως πρόβλημα μείζονος σημασίας ήταν η δέσμευση θέσεων σε εργαστηριακά μαθήματα από φοιτητές περασμένων εξαμήνων που είχαν αποτύχει στην εξέταση του εργαστηρίου παραπάνω από μία φορές έναντι σε φοιτητές που βρισκόντουσαν στο τυπικό εξάμηνο διδασκαλίας του μαθήματος ή είχαν αποτύχει λιγότερες φορές στην εξέταση του.

Η πτυχιακή εργασία εφαρμόζει μια καινοτόμα λύση για το πρόβλημα αυτό. Συγκεκριμένα, αναπτύχθηκε μια διαδικτυακή πλατφόρμα που θα επιτρέπει την ηλεκτρονική υποβολή των αιτήσεων εγγραφής από τους φοιτητές. Η πλατφόρμα θα ενσωματώνει μια σχεσιακή και πάραυτα, ευέλικτη βάση δεδομένων που θα διαχειρίζεται τις αιτήσεις και θα παρέχει αυτόματη αξιολόγηση βάση των καθορισμένων κριτηρίων που θα παρουσιαστούν στην συνέχεια. Έτσι, η διαδικασία εγγραφής επιταχύνεται και ελαχιστοποιείται η προσπάθεια που απαιτείται τόσο από φοιτητές όσο και από τους καθηγητές, προσφέροντας πλήρη αυτοματοποίηση.

Με την εφαρμογή αυτής της πρότασης, αναμένεται η βελτίωση , η αποτελεσματικότητα και η ακρίβεια της διαδικασίας εγγραφής στα Εργαστηριακά Τμήματα, μειώνοντας τα χρονικά περιθώρια και τις πιθανές ανθρώπινες παρατυπίες. Επιπλέον, η πλατφόρμα θα προσφέρει περισσότερη διαφάνεια και ευκολία στη διαχείριση των αιτήσεων από τους διδάσκοντες και θα εξασφαλίζει μια αξιοκρατική διαδικασία επιλογής βάση νέων και προυπάρχοντών παραμέτρων που θα αναλυθούν εν συνεχεία.

1.4 Επίλογος

Στο πρώτο κεφάλαιο έγινε η εισαγωγή στα πρόβλημα που αντιμετωπίζει το Τμήμα εδώ και καιρό μία ιστορική αναδρομή του τι επικρατούσε στις περιόδους δηλώσεων μαθημάτων και τα προσωρινά αντίμετρα στα προβλήματα αυτά που εισήχθησαν από διδάσκοντες των εργαστηριακών μαθημάτων. Μερικά εν μέρη είχαν λυθεί στο απαρχαιωμένο σύστημα της Πυθίας με τον Αλγόριθμο του Δείκτη Προόδου, μερικά άλλα με την εισαγωγή του Uniportal και τέλος τα υπόλοιπα με τις προσαρμοσμένες λύσεις των καθηγητών. Έτσι γεννήθηκε μια νέα ανάγκη δημιουργίας μιας πλατφόρμας που έρχεται να συνδυάσει τις μεμονωμένες λύσεις σε όλα τα προβλήματα από όλες τις όψεις και που θα εφαρμόζει έναν διαφορετικό και πιο δίκαιο Αλγόριθμο από αυτό του Δείκτη Προόδου με αποτέλεσμα να αποτρέπει αθέμιτες συμπεριφορές και να εισάγει μια καινοτόμα και επεκτάσιμη λύση για το θέμα των Δηλώσεων των Εργαστηριακών Τμημάτων.

Κεφάλαιο 2ο: Ανάλυση Απαιτήσεων

2.1 Εισαγωγή

Σκοπός του παρακάτω κεφαλαίου είναι η παρουσίαση των απαιτήσεων οπού η νέα εφαρμογή πρέπει να προσφέρει ώστε να εξαλειφθούν τα προβλήματα που υπάρχουν μέχρι και σήμερα στην διαδικασία δηλώσεων εργαστηριακών τμημάτων κατά κύριο λόγο. Επίσης αναφέρεται συνοπτικά η κάλυψη των πιο μείζονων ζητημάτων όπως η Εξουσιοδότηση (Authorization) και η Αυθεντικοποίηση (Authentication).

2.2 Απαιτήσεις και Λύσεις που Προσφέρει η Εφαρμογή

Ο λόγος υλοποίησης της παρούσας πτυχιακής εργασίας ήταν η αντιμετώπιση των προβλημάτων που ταλανίζουν το Τμήμα εδώ και τόσο καιρό, που αναφέρθηκαν προηγουμένως, μέσω μιας ευέλικτης εφαρμογής που σκοπό έχει να κρατήσει όλα τα καλά των όσων παλαιότερων υλοποιήσεων καθώς και την προσθήκη νέων επεκτάσεων που πλέον απαιτούνται.

Μετά από εκτεταμένη συζήτηση με τον επιβλέποντα καθηγητή, συμφωνήθηκαν οι απαιτήσεις του νέου αυτού λογισμικού καθώς και τα προβλήματα που πρέπει να επιλύουν. Μολονότι στην πορεία ανάπτυξης του λογισμικού προξενήθηκαν και περαιτέρω άτυπες απαιτήσεις, παρακάτω παρατίθενται ότι ιδανικά θα πρέπει:

- Να δημιουργηθεί Web Εφαρμογή για την υποστήριξη της υλοποίησης
- Να δημιουργηθεί Web Application Program Interface που θα εξυπηρετεί την εφαρμογή του Web.
- Να χρησιμοποιεί την υπάρχουσα υποδομή αυθεντικοποίησης του ΔΙ.ΠΑ.Ε.
- Να υπάρχει αυθεντικοποίηση με βάση ρόλους, όπου ο κάθε ρόλος είναι διαβαθμισμένος και θα επιτρέπει είτε απαγορεύει ορισμένες λειτουργίες.
- Να λειτουργεί με βάση Περιόδους Δηλώσεων εαρινού είτε χειμερινού εξαμήνου, που θα δηλώνονται, θα ξεκινούν και θα τερματίζονται αυτοματοποιημένα όταν έρθει ανάλογα η εκάστοτε ημερομηνία έναρξης/λήξης.
- Να υπάρχουν δηλώσεις με βάση προτεραιότητα / δικαιοσύνη η οποία θα υπολογίζεται για τον κάθε φοιτητή / εξεταζόμενο ξεχωριστά και θα γίνεται καλύτερα και δικαιότερα από τον παλιό αλγόριθμο του Δείκτη Προόδου.
- Να αναπτυχθεί ο προαναφερθέν αλγόριθμος δικαιοσύνης που βάση διαφορετικών παραμέτρων του κάθε φοιτητή που ήδη κατέχονται για εκείνη την χρονική στιγμή, να αποτρέπει, να επικροτεί και να δίνει δεύτερες ευκαιρίες.
- Να υπάρχει δυνατότητα για τους καθηγητές να προσθέτουν και να τροποποιούν τα εργαστηριακά τμήματα που θα διδάξουν και που οι φοιτητές επρόκειτο να δηλώσουν , ακόμη και κατά την στιγμή που έχει γίνει η έναρξη των δηλώσεων.
- Να υπάρχει ένα σύστημα παρακολούθησης (monitoring) που πληροφορεί, αν είσαι ο διδάσκοντας καθηγητής , το φόρτο κατανομής που δέχεται το κάθε τμήμα ενός εργαστηρίου κατά την διάρκεια της πορείας δηλώσεων.
- Απόρροια της παραπάνω απαίτησης είναι επίσης, να υπάρχει δυνατότητα τροποποίησης των θέσεων ανάλογα με τον φόρτο που παρουσιάζεται κατά την πορεία των δηλώσεων.
- Να υπάρχει δυνατότητα, τόσο κατά την εξέλιξη των δηλώσεων όσο και μετά το πέρας τους, να γίνει εξαγωγή της αναλυτικής κατάστασης δηλώσεων του κάθε εργαστηριακού μαθήματος, με έναν κοινώς αποδεκτό τρόπο όπως είναι το Microsoft Excel.
- Να υπάρχει η δυνατότητα εκτύπωση ενός Αποδεκτού Εγγράφου για τον κάθε φοιτητή που πιστοποιεί την δήλωση του για μια δεδομένη Περίοδο Δηλώσεων.

- Να αποτρέπει διπλοεγγραφές ή δήλωση του ίδιου φοιτητή σε άλλο εργαστηριακό τμήμα του ίδιου εργαστηριακού μαθήματος.

2.3 Ικανοποίηση Απαιτήσεων

2.3.1 Αυθεντικοποίηση / Εξουσιοδότηση

Η εφαρμογή κάνει εκτεταμένη χρήση του συστήματος αυθεντικοποίησης του ΔΙ.ΠΑ.Ε. και προσφέρει διαφορετικές λειτουργίες ανάλογα με την κατηγορία του κάθε χρήστη εγγεγραμμένου στο Ihu Login Portal. Οι ρόλοι χρηστών που εξυπηρετεί το Ihu Login Portal είναι Φοιτητής (Student) και το Προσωπικό (Staff). Αν και χρήσιμοι έπρεπε να επεκταθούν για την κάλυψη των απαιτήσεων της προηγούμενης ενότητας. Έτσι προστέθηκε ως νέος επαυξητικός ρόλος ο Διαχειριστής (Admin).

Αναφέρεται επαυξητικός καθώς μπορεί να ανατεθεί είτε σε υπάρχων καθηγητή, χωρίς όμως αυτό να κρίνεται αναγκαίο. Αυτό που κρίθηκε αναγκαίο ωστόσο είναι η ύπαρξη τουλάχιστον ενός διαχειριστή που θα μπορεί να χρήσει άλλους διαχειριστές με την σειρά του. Κάθε χρήστης επιβάλλεται να εισέλθει στην εφαρμογή με τη χρήση του ονόματος χρήστη και του κωδικού πρόσβασής από το Ihu Login Portal. Μέσω αυτής της αυθεντικοποίησης, η εφαρμογή μπορεί επιτυχώς να προσδιορίσει την κατηγορία του χρήστη (φοιτητής, καθηγητής ή διαχειριστής). Κατά την πρώτη είσοδό, όλοι οι χρήστες αποθηκεύονται σε μια βάση δεδομένων, με πληροφορίες όπως το ονοματεπώνυμο, μοναδικό εσωτερικό τετραψήφιο κωδικό από την ταυτότητά τους, διεύθυνση email (αν υπάρχει) και, για τους φοιτητές, αριθμό μητρώου, τυπικό εξάμηνο φοίτησης και πολλές άλλες που θα αναφερθούν σε επόμενο κεφάλαιο.

2.3.2 Ρόλοι χρηστών

- Φοιτητής (Student): Ρόλος με την πιο μειωμένη πρόσβαση είναι αυτός του φοιτητή. Όποιος εξαρχής είναι δηλωμένος με αυτόν τον ρόλο θεωρείται ως ο Απλός Χρήστης της εφαρμογής που οι μόνες λειτουργίες που μπορεί να εκτελέσει είναι οι δηλώσεις εργαστηριακών τμημάτων καθώς και η εξαγωγή της δήλωσης του για την τρέχουσα περίοδο. Επιτρέπεται η πλοήγηση μόνο σε συγκεκριμένα τμήματα της εφαρμογής εάν η περίοδος δηλώσεων είναι ενεργή και σε άλλα όταν η περίοδος είναι ανενεργή.
- Καθηγητές / Προσωπικό (Staff): Ρόλος με επαυξημένη πρόσβαση είναι αυτού που δηλώνεται ως Staff. Όποιος κατέχει τον ρόλο αυτό έχει την δυνατότητα πλοήγησης σε περισσότερες ενότητες της εφαρμογής καθώς και αυξητικά από τον Φοιτητή, έχει την δυνατότητα Προσθήκης Εργαστηριακών Μαθημάτων κάτω από το μοναδικό αναγνωριστικό που του παρέχει η σχεσιακή βάση δεδομένων αυτόματα. Επιπροσθέτως έχει πρόσβαση στο σύστημα παρακολούθησης της πορείας των δηλώσεων. Ο χρήστης που κατέχει τον ρόλο του καθηγητή, μπορεί όπως προαναφέρθηκε, να προσθέσει το δικό του εργαστηριακό μάθημα και επίσης να δηλωθεί ως συνεργάτης στην διδασκαλία ενός εργαστηριακού τμήματος που έχει προστεθεί από κάποιον άλλο διαφορετικό χρήστη που κατέχει τον Ρόλο του Καθηγητή.

  Συνεπώς εφόσον έχει ρόλο στην συνδιδασκαλία του εργαστηριακού μαθήματος έχει και πρόσβαση στο σύστημα παρακολούθησης της πορείας των δηλώσεων, ωστόσο δεν μπορεί να προβεί σε οποιαδήποτε τροποποίηση του εργαστηριακού τμήματος καθώς δεν είναι ο συντονιστής του μαθήματος, ή καλύτερα διατυπωμένα, δεν είναι αυτός που το πρόσθεσε εξ αρχής. Να σημειωθεί ότι ένας χρήστης δεν μπορεί να έχει ταυτόχρονα τον Ρόλο του Φοιτητή και του Καθηγητή.
- Διαχειριστής Εφαρμογής (Admin): Ο Ρόλος με την ύψιστη πρόσβαση είναι αυτός του διαχειριστή εφαρμογής , έχει πρόσβαση σε όλες τις ενότητες της εφαρμογής. Ας σημειωθεί ότι ο όρος επαυξητικός δεν σημαίνει ότι όποιος τον κατέχει, χάνει τον προηγούμενο που κατείχε.

Για παράδειγμα αν ένας χρήστης που κατείχε τον Ρόλο του Καθηγητή δηλωθεί και ως διαχειριστής εφαρμογής, δεν παύει να είναι καθηγητής και δεν γίνεται ανάκληση των δικαιωμάτων και προσβάσεων

που κατείχε πρωτίστως. Ο χρήστης με τον ρόλο αυτό έχει το δικαίωμα και την πρόσβαση να διαχειρίζεται τις περιόδους δηλώσεων. Μπορεί να δηλώσει , να κάνει έναρξη και να λήξει μια περίοδο κατά το δοκούν. Επίσης έχει την δυνατότητα να προσθέσει και τον Ρόλο του Διαχειριστή σε άλλους χρήστες που δεν τον κατέχουν και κατέχουν μόνο τον Ρόλο Καθηγητή

2.3.3 Εξαγωγή Σε Υπολογιστικά Φύλλα

Η μετατροπή πληροφορίας σε κατάλληλη μορφή και έπειτα μετατροπή αυτής σε ένα κοινώς αποδεκτό πρότυπο όπως είναι εδώ και χρόνια τα αρχεία υπολογιστικών φύλλων του Microsoft Excel εξυπηρετούνται από το Backend της εφαρμογής με χρήση βιβλιοθηκών που υποστηρίζουν την λειτουργία αυτή. Ωστόσο λόγω συνέπειας και αποφυγής προβλημάτων όπως Memory Leaks που προέρχονται από την χρήση Open Source λογισμικού όπως η βιβλιοθήκη που χρησιμοποιήθηκε (CsvHelper) η λεπτομερής υλοποίηση γίνεται σε παρακάτω κεφάλαιο.

2.4 Επίλογος

Στο κεφάλαιο αυτό περιεγράφηκε επιγραμματικά η κύρια λειτουργία του συστήματος και οι τρεις κατηγορίες χρηστών που υπάρχουν και χρησιμοποιούνται στο σύνολο της εφαρμογής.

Παρουσιάστηκαν οι απαιτήσεις ανάπτυξης του λογισμικού καθώς και μια πρωταρχική προσπάθεια ανάλυσης του πως καλύφθηκαν αυτές δίχως να γίνει εκτεταμένη αναφορά σε τεχνολογίες και βιβλιοθήκες. Όσες απαιτήσεις δεν καλύφθηκαν πλήρως σε αυτό το κεφάλαιο θα καλυφθούν στα επόμενα κεφάλαια.

Επίσης, πρέπει να δοθεί ιδιαίτερη σημασία στην ασφάλεια και την αυθεντικοποίησης των χρηστών που παρέχει η εν λόγω εφαρμογή. Η είσοδος στην εφαρμογή απαιτεί την προσθήκη ονόματος χρήστη και κωδικού πρόσβασης, ενώ καταναλώνεται εξ ολοκλήρου το Ihu Login Portal του Τμήματος για την τελική επιβεβαίωση της ταυτότητας του. Αυτό εξασφαλίζει ότι μόνο εγκεκριμένοι χρήστες, που έχουν ήδη εγγραφεί στις δομές του Τμήματος μπορούν να αποκτήσουν πρόσβαση στην εφαρμογή και να χρησιμοποιήσουν τις σχετικές λειτουργίες που θα περιγράφουνε στην συνέχεια

Κεφάλαιο 3ο: Υλοποίηση Σχήματος και Βάσης με PostgreSQL

3.1 Εισαγωγή

Για τη βάση δεδομένων της εφαρμογής, επιλέχθηκε η PostgreSQL. Η PostgreSQL είναι ένα ισχυρό σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) ανοικτού κώδικα, γνωστό για την ευρωστία, την επεκτασιμότητα και το εκτεταμένο σύνολο χαρακτηριστικών του. Αναπτύχθηκε στις αρχές της δεκαετίας του 1980 και έχει εξελιχθεί σε μια ώριμη και ευρέως αποδεκτή λύση βάσης δεδομένων που χρησιμοποιείται από οργανισμούς όλων των μεγεθών σε διάφορους κλάδους.

3.2 Πλεονεκτήματα PostgreSQL Έναντι Ανταγωνιστών

Η PostgreSQL είναι εξαιρετικά αξιόπιστη και παρέχει ισχυρή ακεραιότητα δεδομένων μέσω της υποστήριξής της για συναλλαγές (transactions) ACID (Atomicity, Consistency, Isolation, Durability). Εξασφαλίζει ότι οι λειτουργίες της βάσης δεδομένων είτε ολοκληρώνονται πλήρως είτε ανατρέπονται σε περίπτωση αποτυχίας, διατηρώντας την ακεραιότητα και τη συνέπεια των δεδομένων.

Το συγκεκριμένο DBMS σε σύγκριση με άλλα επιλέχθηκε λόγου των βέλτιστων τεχνικών διαχείρισης των TCP Connections που εγκαθιδρύει, της εσωτερική διαχείρισης primary και complex types που πραγματοποιεί στο παρασκήνιο για την διαχείριση των τιμών που αποθηκεύονται εν τέλη στους σχεσιακούς πίνακες καθώς και για τον λόγο ότι είναι το DBMS που ξέρουν καλύτερα στην χρήση οι συγγραφείς της εφαρμογής .

Ένα από τα βασικά πλεονεκτήματα της PostgreSQL είναι η ικανότητά της να χειρίζεται πολύπλοκα δεδομένα και να υποστηρίζει προηγμένα και περίπλοκα SQL ερωτήματα. Τηρεί αυστηρά το πρότυπο της SQL και προσφέρει ένα πλούσιο σύνολο τύπων δεδομένων, συμπεριλαμβανομένης της υποστήριξης JSON αντικειμένων, πινάκων και γεωμετρικών τύπων.

Το DBMS χειρίζεται πολύ αποτελεσματικά μεγάλα σύνολα δεδομένων και υψηλούς αριθμούς ταυτόχρονων αιτήσεων για προσπελάσεις σε αυτά. Υποστηρίζει διάφορες τεχνικές ευρετηρίασης, συμπεριλαμβανομένων των B-trees, των hash indexes και των γενικευμένων δέντρων αναζήτησης (GiST), επιτρέποντας τη βελτιστοποίηση και την απόδοση των SQL queries ακόμη περισσότερο. Επιπλέον, προσφέρει ενσωματωμένες λειτουργίες για παράλληλη εκτέλεση queries , καθιστώντας την κατάλληλη για πολύ απαιτητικές εφαρμογές.

Ένα άλλο δυνατό σημείο της είναι η υποστήριξή της για προηγμένα χαρακτηριστικά, όπως η κατάτμηση πινάκων, η λογική αντιγραφή και τα wrappers ξένων δεδομένων.

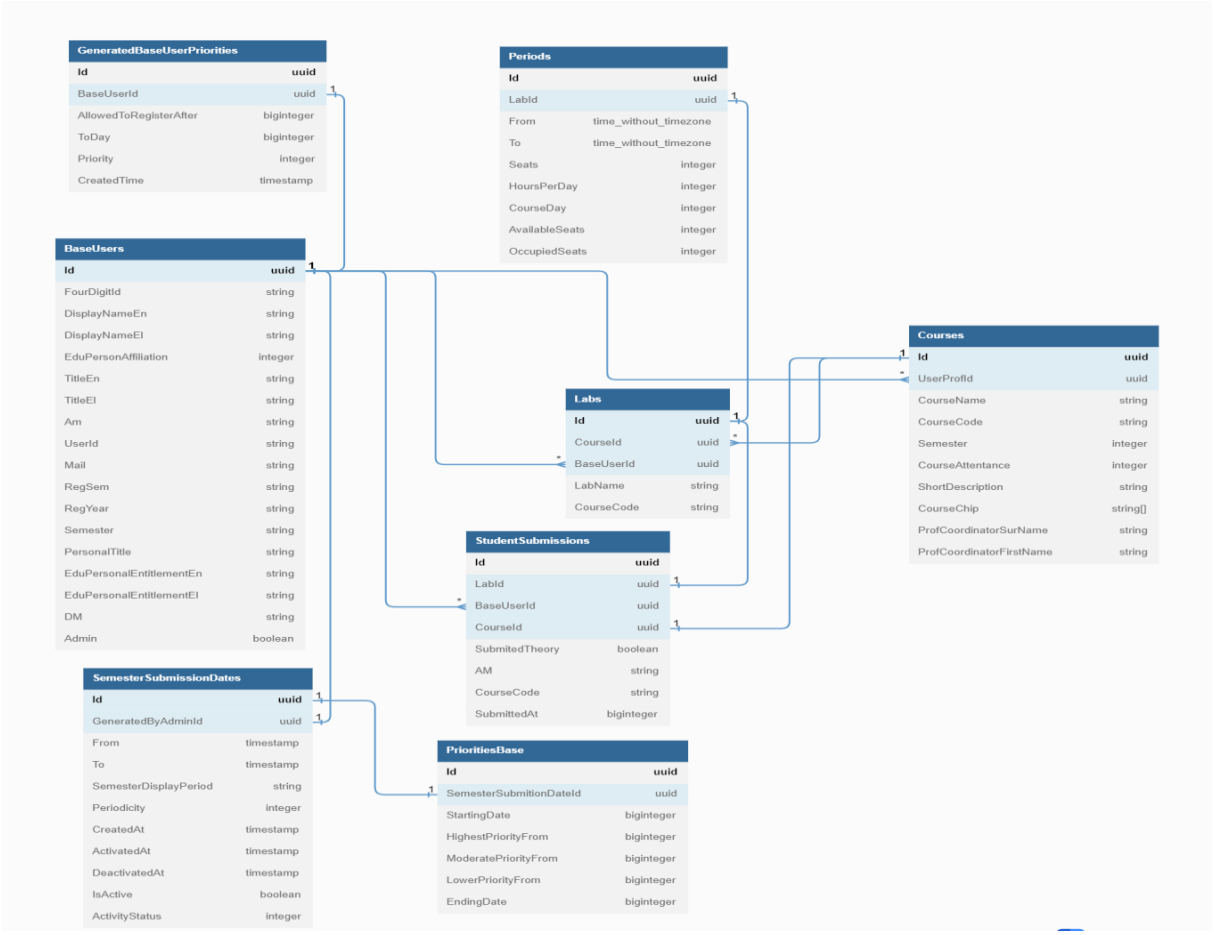
- Η διαίρεση πινάκων επιτρέπει την κατανομή των δεδομένων σε πολλαπλά φυσικά αρχεία, βελτιώνοντας την απόδοση των ερωτημάτων και απλοποιώντας τη διαχείριση των δεδομένων.
- Η λογική αντιγραφή επιτρέπει την επιλεκτική αντιγραφή συγκεκριμένων πινάκων ή βάσεων δεδομένων, διευκολύνοντας τη διανομή και την αντιγραφή δεδομένων σε κατανεμημένα συστήματα.
- Τα wrappers ξένων δεδομένων επιτρέπουν την πρόσβαση και την αλληλεπίδραση με δεδομένα σε εξωτερικές βάσεις δεδομένων ή συστήματα, παρέχοντας μια ενοποιημένη διεπαφή για την πρόσβαση σε ετερογενείς πηγές.

Συνολικά, είναι ένα αξιόπιστο DBMS, επεκτάσιμο και πλούσιο σε χαρακτηριστικά που προσφέρει ισχυρές δυνατότητες διαχείρισης δεδομένων. Η εκτεταμένη υποστήριξη SQL, η επεκτασιμότητα και η

ενεργή κοινότητα την καθιστούν εξαιρετική επιλογή για ένα ευρύ φάσμα εφαρμογών, από μικρά έργα έως συστήματα επιχειρησιακών επιπέδων.

3.3 Ανάλυση Σχεσιακού Σχήματος Εφαρμογής

Στην ενότητα αυτή θα γίνει ανάλυσης και επεξήγηση του σχεσιακού σχήματος που αποφασίστηκε και εφαρμόστηκε για την εξυπηρέτηση των απαιτήσεων που καλύφθηκαν από την εφαρμογή.



Σχήμα 3.1: Διάγραμμα Οντοτήτων - Συσχετίσεων

3.3.1 Πίνακας BaseUsers

Σε αυτόν τον πίνακα καταγράφονται όλες οι εγγραφές των χρηστών που πρέπει να αποθηκευτούν στα πλαίσια της παρούσας πτυχιακής εργασίας. Κατά την πρώτη εκτέλεση της εφαρμογής, θα πραγματοποιηθούν αυτόματες εγγραφές για όλους τους καθηγητές που υπάρχουν και αποδίδονται στο σύστημα του Τμήματος.

Όταν ένας νέος χρήστης συνδεθεί για πρώτη φορά, προστίθεται μια νέα εγγραφή στον πίνακα και ανάλογα με τα στοιχεία που έχουν μεταφερθεί από το Token της υπηρεσίας του Τμήματος, αποκτά τα αντίστοιχα χαρακτηριστικά ως φοιτητής ή καθηγητής. Οι στήλες του πίνακα περιλαμβάνουν τα εξής:

- ID: Μια μοναδική αναγνωριστική τιμή για κάθε εγγραφή στον πίνακα (κύριο κλειδί).
- FourDigitId: Τα τελευταία 4 ψηφία της ακαδημαϊκής ταυτότητας.
- DisplayNameEn: Το όνομα και επώνυμο του χρήστη στα Αγγλικά.
- DisplayNameEl: Το όνομα και επώνυμο του χρήστη στα Ελληνικά.

- EduPersonAffiliation: Η ιδιότητα του χρήστη, είτε είναι φοιτητής (Student) είτε προσωπικό(Staff).
- TitleEn: Ο τίτλος του χρήστη στα Αγγλικά, π.χ. Καθηγητής, Λέκτορας, Φοιτητής, Γραμματεία, Αναπληρωτής Καθηγητής κ.λπ.
- TitleEl: Ο τίτλος του χρήστη στα Ελληνικά.
- AM: Το Αριθμό Μητρώου του χρήστη (σε περίπτωση που είναι καθηγητής, αυτή η τιμή είναι null).
- UserId: Ο Αριθμός Μητρώου του χρήστη με το χαρακτηριστικό της σχολής, π.χ. It... (σε περίπτωση που είναι καθηγητής, αυτή η τιμή θα είναι null).
- Mail: Το ηλεκτρονικό ταχυδρομείο του χρήστη.
- RegSem: Το εξάμηνο εγγραφής του χρήστη.
- RegYear: Το έτος εισαγωγής του χρήστη.
- Semester: Το τρέχον εξάμηνο του χρήστη
- PersonalTitle: Ο τίτλος που έχει ως καθηγητής.
- EduPersonalEntitlementEN: Η εξειδίκευση του καθηγητή στα Αγγλικά.
- EduPersonalEntitlementEl: Η εξειδίκευση του καθηγητή στα Ελληνικά.
- DM: Οι διδακτικές μονάδες που έχει ο χρήστης σε περίπτωση που είναι Φοιτητής
- Admin: Ενδεικτικό χαρακτηριστικό που υποδηλώνει αν ο χρήστης έχει δηλωθεί ως διαχειριστής ή όχι.

Ούτε ο καθηγητής ούτε ο φοιτητής είναι υποχρεωτικό να έχει όλες τις πληροφορίες συμπληρωμένες όμως είναι αναγκαστικό στις δύο περιπτώσεις χρηστών να υπάρχουν διαθέσιμες κάποιες κρίσιμες πληροφορίες, όπου η απώλεια αυτών καθιστά αδύνατο την εφαρμογή να εξυπηρετήσει.

Τα κοινά αναγκαστικά πεδία που πρέπει να φέρουν τιμή και στους 2 τύπους χρηστών είναι:

- ID
- FourDigitId
- DisplayNameEn.
- DisplayNameEl.
- EduPersonAffiliation

Η απώλεια τιμής στα υπόλοιπα πεδία του πίνακα εάν ο χρήστης είναι καθηγητής δεν φέρει κάποια επίπτωση στην λειτουργία της εφαρμογής. Ωστόσο η απώλεια τιμής στα παρακάτω πεδία αν ο χρήστης έχει χαρακτηριστεί ως φοιτητής καθιστά την εφαρμογή μη ικανή να παράγει και να σερβίρει ορθά αποτελέσματα. Τα πεδία αυτά είναι τα παρακάτω:

- Am
- UserId
- RegSem
- RegYear
- Semester
- Dm

3.3.2 Πίνακας Courses

Ο παρακάτω πίνακας περιέχει αντιπροσωπεύει όλα τα εννοιολογικά εργαστηριακά μαθήματα που έχουν καταχωρήσει οι καθηγητές. Τα πεδία του πίνακα περιλαμβάνουν:

- ID: Μια μοναδική αναγνωριστική τιμή για κάθε εγγραφή στον πίνακα (κύριο κλειδί)
- UserProfId: Το ID του καθηγητή στον πίνακα των χρηστών (BaseUsers), ο οποίος έχει δημιουργήσει την εγγραφή του εργαστηριακού. μαθήματος. Λειτουργεί ως ξένο κλειδί για τον Πίνακα BaseUsers
- CourseName: Η ονομασία του μαθήματος.
- CourseCode: Η κωδική αναπαράσταση του μαθήματος που εισήχθη από τον διδάσκοντα Καθηγητή (π.χ.1242-2332.).
- Semeste: Το εξάμηνο διεξαγωγής του μαθήματος.
- CourseAttendanc: Η υποχρέωση παρακολούθησης του κάθε εργαστηριακού μαθήματος.
- ShortDescription: Μικρή περιγραφή του εργαστηριακού μαθήματος.
- CourseChip: Μια λίστα με τα βασικά χαρακτηριστικά του μαθήματος, όπως το εξάμηνο, η υποχρεωτικότητα και άλλες γενικές πληροφορίες.
- ProfCoordinatorSurName: Το επώνυμο του συντονιστή του μαθήματος.
- ProfCoordinatorFirstName: Το μικρό όνομα του συντονιστή του μαθήματος.

Κάθε μάθημα είναι μια μοναδική εγγραφή στον πίνακα και δεν μπορεί να υπάρχει ίδιος κωδικός μαθήματος περισσότερες από μία φορές. Υπάρχει σύνδεση 1 προς Πολλά με τον πίνακα Labs που περιγράφεται παρακάτω καθώς 1 Course διαθέτει 1 ή περισσότερα εργαστηριακά τμήματα.

3.3.3 Πίνακας Labs

Ο Πίνακας Labs αντιπροσωπεύει ένα Εργαστηριακό Τμήμα ως οντότητα. Αυτός ο πίνακας συνδέεται με τον πίνακα Courses. Το ID του πίνακα Course είναι το ξένο κλειδί σε αυτόν τον πίνακα, δημιουργώντας έτσι μια σχέση ένα προς πολλά. Δηλαδή, ένα εργαστηριακό μάθημα μπορεί να περιέχει πολλά Εργαστηριακά Τμήματα (Course HasMany Labs). Τα πεδία αυτού του πίνακα είναι τα εξής:

- ID: Μια μοναδική αναγνωριστική τιμή για κάθε εγγραφή στον πίνακα (κύριο κλειδί).
- CourseId: Το ξένο κλειδί της εγγραφής του μαθήματος στην εγγραφή του εργαστηριακού τμήματος
- BaseUserId: Το ID του καθηγητή που διδάσκει τμήμα. Χρησιμοποιείται ως ξένο κλειδί καθώς κάνει αναφορά σε εγγραφές του πίνακα BaseUsers.
- LabName: Η κωδική ονομασία του κάθε τμήματος, π.χ. T1, T2, E1, E2.
- CourseCode: Η κωδική αναπαράσταση του μαθήματος που ανήκει.

3.3.4 Πίνακας Periods

Ως Period ενός εργαστηριακού τμήματος ορίζεται η περίοδος διεξαγωγής του συγκεκριμένου τμήματος μαζί με πληροφορίες για τις θέσεις παρακολούθησης που έχουν δηλωθεί. Ο πίνακας Periods υλοποιεί μια σχέση ένα-προς-ένα με τον πίνακα Labs. Επεξηγηματικά, ένα εργαστηριακό τμήμα μπορεί να έχει ένα και μόνο ένα Period (Labs HasOne Period). Τα πεδία του οποίου είναι τα εξής:

- ID: Μια μοναδική αναγνωριστική τιμή για κάθε εγγραφή στον πίνακα (κύριο κλειδί).
- LabId: Το ξένο κλειδί του πίνακα, δηλώνοντας ότι αυτή η εγγραφή περιόδου ανήκει σε συγκεκριμένο εργαστηριακό τμήμα.
- From: Η ώρα έναρξης του συγκεκριμένου εργαστηριακού τμήματος.
- To: Η ώρα λήξης του συγκεκριμένου εργαστηριακού τμήματος.
- Seats: Ο μέγιστος αριθμός θέσεων/εγγραφών που μπορεί να δεχτεί το εργαστηριακό τμήμα.
- HoursPerDay: Η διάρκεια του μαθήματος, υπολογιζόμενη από την αφαίρεση των πεδίων To και From (To - From).

- CourseDay: Η ημέρα διεξαγωγής του μαθήματος, αναπαριστάτε με έναν ακέραιο αριθμό, όπου 1 αντιστοιχεί στη Δευτέρα, 2 στην Τρίτη, 3 στην Τετάρτη, 4 στην Πέμπτη και 5 στην Παρασκευή.
- AvailableSeats: Οι διαθέσιμες θέσεις που έχουν απομείνει προς πλήρωση στο εργαστηριακό τμήμα.
- OccupiedSeats: Ο Αριθμός των θέσεων που έχουν πληρωθεί για το εργαστηριακό τμήμα.

3.3.5 Πίνακας StudentSubmissions

Σημειώνεται ότι μια εγγραφή στον Πίνακα StudentSubmissions μπορεί να πραγματοποιηθεί μόνο από χρήστη που κατέχει τον Ρόλο του Φοιτητή. Ο παρακάτω πίνακας περιλαμβάνει όλες τις εγγραφές των φοιτητών που έχουν κάνει κάποια δήλωση σε ένα εργαστηριακό τμήμα που υπάγεται σε μία οντότητα εργαστηριακού μαθήματος. Οι δηλώσεις γίνονται κατά τη πρόοδο της περιόδου, ενώ κατά τη λήξη αυτής οι τιμές του πίνακα διαγράφονται και αναμένονται πλέον εκ νέου εγγραφές. Περιλαμβάνει τα εξής πεδία:

- ID: Το μοναδικό αναγνωριστικό για κάθε εγγραφή του πίνακα (κύριο κλειδί).
- LabId: Το ξένο κλειδί που αναφέρεται στον πίνακα Lab και υποδηλώνει σε ποιο εργαστηριακό τμήμα έχει πραγματοποιηθεί η δήλωση. Συσχέτιση 1-1 με τον Πίνακα Labs (StudentSubmissions HasOne Lab).
- BaseUserId: Το ξένο κλειδί που αναφέρεται στον πίνακα BaseUsers και υποδηλώνει ποιος χρήστης έχει κάνει την δήλωση. Συσχέτιση 1-1 με τον Πίνακα BaseUsers, (StudentSubmissions HasOne BaseUser).
- CourseId: Το ξένο κλειδί που αναφέρεται στον πίνακα Courses και υποδηλώνει με ποιο εργαστηριακό μάθημα έχει συνδεθεί η δήλωση. Συσχέτιση 1-1 με τον Πίνακα Courses, (StudentSubmissions HasOne Course).
- SubmitedTheory: Είναι μία boolean τιμή που υποδηλώνει εάν έχει δηλωθεί η θεωρία του αντίστοιχου συνδεδεμένου εργαστηριακού μαθήματος ως οντότητα για τον συγκεκριμένο χρήστη που πραγματοποίησε την δήλωση.
- AM: Το A.M. του χρήστη - φοιτητή που πραγματοποίησε την δήλωση.
- CourseCode: Η κωδική αναπαράσταση του εργαστηριακού μαθήματος που εισήχθη από τον διδάσκοντα καθηγητή.
- SubmitedAt: Η ημερομηνία καταχώρησης της εγγραφής στον πίνακα.

3.3.6 Πίνακας GeneratedBaseUserPriorities

Ο Πίνακας GeneratedBaseUserPriorities αποθηκεύει την προτεραιότητα που αποφασίστηκε ανά φοιτητή από τον Αλγόριθμο Δικαιοσύνης για την συγκεκριμένη περίοδο δηλώσεων, υπενθυμίζοντας ότι ο αλγόριθμος ενεργεί μόνο πάνω σε φοιτητές. Με αυτόν τον τρόπο αποσαφηνίζεται από πότε μέχρι πότε μπορεί να πραγματοποιήσει τη δήλωσή του ο κάθε χρήστης-φοιτητής της εφαρμογής. Τα πεδία αυτού είναι τα εξής:

- ID: Μια μοναδική αναγνωριστική τιμή για κάθε εγγραφή στον πίνακα (κύριο κλειδί).
- BaseUserId: Το ID του φοιτητή από τον πίνακα BaseUsers. Λειτουργεί ως Ξένο κλειδί στον Πίνακα BaseUsers και σχετίζεται 1 – 1.
- AllowedToRegisterAfter: Μια μεγάλη ακέραια τιμή που αναπαριστά την ημερομηνία σε UnixTimestamp, όπου έπειτα από αυτή ο φοιτητής έχει δικαίωμα να κάνει τη υποβάλει δήλωσή του.
- ToDate: Μια μεγάλη ακέραια τιμή που αναπαριστά την ημερομηνία σε UnixTimestamp, όπου πριν το πέρας της ο χρήστης-φοιτητής έχει το δικαίωμα να υποβάλλει τη δήλωσή του.

- Priority: Μια ακέραια τιμή που υποδηλώνει την προτεραιότητα που του έχει ανατεθεί. Με την τιμή 10 ορίζεται η Ύψιστη προτεραιότητα, με την τιμή 20 ορίζεται η Μέτρια προτεραιότητα και με την τιμή 30 ορίζεται η Χαμηλότερη προτεραιότητα.
- CreatedTime: Η ημερομηνία δημιουργίας καταχώρησης της συγκεκριμένης εγγραφής στο Πίνακας για τον συγκεκριμένο φοιτητή.

Κάθε εγγραφή αντιστοιχεί σε έναν μοναδικό χρήστη - φοιτητή. Δεν υπάρχει δυνατότητα πραγματοποίησης δήλωσης ενός φοιτητή σε ένα εργαστηριακό μάθημα δεύτερη φορά από την στιγμή που έχει ήδη δεσμεύσει μία θέση σε ένα εργαστηριακό τμήμα του ίδιου εργαστηριακού μαθήματος.

3.3.7 Πίνακας PrioritiesBase

Σκοπός του πίνακα PrioritiesBase είναι να καταγράφει τις εγγραφές κάθε φορά που εκκινείτε μια περίοδος δηλώσεων. Κατά τη δημιουργία της ημερομηνίας έναρξης, αυτόματα υπολογίζονται οι τρεις κατηγορίες προτεραιότητας βάσει της ημερομηνίας έναρξης και λήξης που περιεγράφηκε σε προηγούμενη ενότητα . Όλες οι ημερομηνίες σε αυτόν τον πίνακα είναι σε μορφή ακέραιου αριθμού υπό μορφή UnixTimestamp. Τα πεδία του πίνακα είναι τα εξής:

- ID: Το μοναδικό αναγνωριστικό κάθε εγγραφής του πίνακα (κύριο κλειδί).
- SemesterSubmissionDateId: Είναι το ξένο κλειδί που συσχετίζει 1 προς 1 για εγγραφή του πίνακα SemesterSubmissionDates, με την συγκεκριμένη εγγραφή προτεραιοτήτων(PrioritiyBase HasOne SemesterSubmissionDate).
- StartingDate: Η ημερομηνία έναρξης της περιόδου δηλώσεων.
- HighestPriorityFrom: Η ημερομηνία από την οποία και έπειτα οι φοιτητές με υψηλή προτεραιότητα θα μπορούν να υποβάλουν δηλώσεις.
- ModeratePriorityFrom: Η ημερομηνία από την οποία και έπειτα οι φοιτητές με μέτρια προτεραιότητα θα μπορούν να υποβάλουν δηλώσεις.
- LowerPriorityFrom: Η ημερομηνία από την οποία και έπειτα οι φοιτητές με χαμηλή προτεραιότητα θα μπορούν να υποβάλουν δηλώσεις.
- EndingDate: Ημερομηνία ολοκλήρωσης της διενέργειας των δηλώσεων.

3.3.8 Πίνακας SemesterSubmissionsDates

Ο πίνακας SemesterSubmissionDates χρησιμοποιείται για την αποτύπωση της περιόδου δηλώσεων στο σχεσιακό σχήμα. Σε αυτόν τον πίνακα θα υπάρχει πάντα μία μοναδική εγγραφή της περιόδου που πρόκειται να ή έχει εκκινηθεί καθώς η προηγούμενη που έχει έρθει εις πέρας, θα διαγράφεται κατά την ολοκλήρωση της μετά την ανάλογη πράξη από τον Διαχειριστή Εφαρμογής. Ο πίνακας έχει την ακόλουθη μορφή:

- ID: Το μοναδικό αναγνωριστικό για κάθε εγγραφή του πίνακα (κύριο κλειδί).
- GeneratedByAdmin: Το ID του καθηγητή που δημιούργησε την εγγραφή για εκκίνηση ή λήξη της περιόδου δηλώσεων. Το ID αντιστοιχεί ως ξένο κλειδί που αναφέρεται στον πίνακα "BaseUsers" και σχετίζεται 1 προς 1 με αυτόν.
- From: Η ημερομηνία έναρξης της περίδου δηλώσεων.
- To: Η ημερομηνία λήξης της περιόδου δηλώσεων.
- SemesterDisplayPeriod: Το όνομα της περιόδου δηλώσεων ανά εξάμηνο και ακαδημαϊκό έτος, για παράδειγμα "2023_2024 ΕΑΡΙΝΟ" ή "2023_2024 ΧΕΙΜΕΡΙΝΟ".
- Periodicity: Η περίοδος που γίνονται οι δηλώσεις, όπου 1 = Χειμερινό και 2 = Εαρινό εξάμηνο.
- CreatedAt: Η ημερομηνία καταχώρησης της εγγραφής στον πίνακα.

- ActivatedAt: Η ημερομηνία που πραγματοποιήθηκε αυτοματοποιημένη ενεργοποίηση της περιόδου δηλώσεων.
- DeactivatedAt: Η ημερομηνία που πραγματοποιήθηκε αυτοματοποιημένη λήξη της περιόδου δηλώσεων.
- IsActive: Ένδειξη για το εάν η τρέχουσα εγγραφή που υποδηλώνει μια περίοδο δηλώσεων είναι ενεργή ή όχι.
- ActivityStatus: Ένδειξη για την τωρινή κατάσταση της περιόδου με ακέραιο αριθμό 1,2,3. Όπου 1=Ανενεργή, 2=Ενεργή, 3=Προγραμματισμένη προς ενεργοποίηση.

3.4 Επίλογος

Σε αυτό το κεφάλαιο αναλύεται το σχεσιακό σχήμα της εφαρμογής. Αναφέρονται οι συσχετίσεις υπό την μορφή κυρίων και ξένων κλειδιών μεταξύ των πινάκων που την εξυπηρετούν. Χρησιμοποιήθηκαν κυρίως συσχετίσεις 1 προς 1 αλλά και 1 προς πολλά σε συγκεκριμένες μόνο περιπτώσεις. Όλοι οι πίνακες σχεδιάστηκαν ώστε να αντιπροσωπεύουν την δική τους αυτόνομη και ξεχωριστή οντότητα. Με την παραπάνω λογική διαίρεση των οντοτήτων που είναι αποτέλεσμα ανάλυσης των απαιτήσεων αυτών καθαυτών, σε αυτόνομες υπό-μονάδες που συσχετίζονται αυστηρά με άλλους πίνακες, διασφαλίζεται η ακεραιότητα των δεδομένων καθώς υποστηρίζεται πιο εύκολα η επέκταση του σχήματος κάτω από κάθε εννοιολογικό σχήμα - πίνακα με παραπάνω πληροφορίες χωρίς να επηρεάζεται η ακεραιότητα της εφαρμογής.

Κεφάλαιο 4ο: Υλοποίηση Backend με .NET CORE

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα παρουσιαστούν οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του BackEnd τμήματος της εφαρμογής της παρούσας Π.Ε. Θα αναλυθούν οι λόγοι που επιλέχθηκαν οι παρακάτω τεχνολογίες, καθώς και διάφορες τεχνικές που μπορούν να επιτευχθούν από την χρήση των τεχνολογιών αυτών. Τέλος, θα εξεταστούν ορισμένα τμήματα της υλοποίησης που αξίζουν αναφοράς.

4.2 Τεχνολογία .NET CORE 6.0 / Entity Framework Core

4.2.1 .NET Core 6.0 Framework

Για το Backend (API) της εφαρμογής, επιλέχθηκε συγκεκριμένα το .NET Core 6.0 , ένα ισχυρό Microsoft Framework για ανάπτυξη εφαρμογών σε γλώσσα προγραμματισμού C#. Παρέχει έναν πλήρη σετ εργαλείων για την ανάπτυξη, την προστασία και τη διαχείριση ενός Web API, καθώς η γλώσσα προγραμματισμού C# είναι ισχυρή και ευέλικτη, προσφέροντας πολλές δυνατότητες για την υλοποίηση της λειτουργικότητας της εφαρμογής.

Το .NET Core 6.0 είναι ένα διαπλατφορμικό Framework ανοικτού κώδικα που αναπτύχθηκε από τη Microsoft. Έχει σχεδιαστεί για τη δημιουργία σύγχρονων, cloud-native και υψηλής απόδοσης εφαρμογών που μπορούν να τρέξουν σε διάφορες πλατφόρμες, όπως Windows, macOS και συστήματα Unix. Το .NET Core 6.0 κυκλοφόρησε τον το 2021 στην αγορά και φέρνει μια σειρά από νέα χαρακτηριστικά, βελτιώσεις και επεκτάσεις στο γενικότερο οικοσύστημα της .NET που επιζητούσαν προγραμματιστές απο την Microsoft εδώ και καιρό. Κάποιες από αυτές τις πιο κύριες βελτιώσεις είναι:

- Οι Βελτιωμένες επιδόσεις: Το .NET Core 6.0 εισάγει διάφορες βελτιώσεις επιδόσεων, συμπεριλαμβανομένων ταχύτερων χρόνων εκκίνησης, μειωμένου αποτυπώματος στη μνήμη του μηχανήματος και βελτιωμένου Just-In-Time-Compilation (JIT).
- Εφαρμογές Ενιαίου Αρχείου: Όπου πλέον όλα τα εξαρτόμενα τμήματα μιας εφαρμογής συγκεντρώνονται κάτω από ένα εκτελέσιμο αρχείο.
- Ανάπτυξη βελτιωμένων Web APIs: παρέχοντας το νέο Framework που χρησιμοποιείται για την Π.Ε. με όνομα ASP.NET Core Web API. Το ASP.NET Core Web API ενσωματώνει και απλοποιεί πολλά πράγματα για την ανάπτυξη μιας τέτοιας υπηρεσίας.
- Βελτίωση στην Γλώσσα Υλοποίησης και Μείωση Χρόνου Εκτελέσεων: όπου η παρούσα έκδοση έρχεται να εκσυγχρονίσει ακόμη περισσότερο την γλώσσα C# με νέα χαρακτηριστικά όπως έναν.
- Βελτιωμένο Garbage Collector και ενισχυμένη υποστήριξη για υλοποιήσεις στην αρχιτεκτονική ARM64.

Κάθε Web API χτισμένο σε .NET Core Web API Framework οφείλει να είναι σχεδιασμένο αποκεντρωμένα ακολουθώντας ένα Design Pattern όπως το MVC που είναι από τα πιο γνωστά και με την μορφή των Injectable Services που περιγράφονται παρακάτω. Οι τεχνικές αυτές , όπου και ακολουθήθηκαν πιστά, κάνουν την εφαρμογή πιο ευέλικτη, επαναχρησιμοποιήσιμη και λιγότερο κοστοβόρα σε πόρους και απαιτήσεις του μηχανήματος όπου θα τρέξει. Αν και η περιγραφή των MVC, Services και του Dependency Injection που θα περιγραφούν στην συνέχεια και θα γίνει προσπάθεια αναλυτικής επεξήγησης όλων των παραπάνω στις παρακάτω ενότητες.

4.2.2 Services και Dependency Injection Pattern στη .Net Core 6.0

Οι υπηρεσίες (Services) στη .NET Core 6.0 ορίζονται συνήθως ως κλάσεις που ενθυλακώνουν ένα συγκεκριμένο σύνολο συναφών λειτουργιών, όπως η πρόσβαση σε δεδομένα, η επιχειρησιακή λογική, οι ενσωματώσεις λειτουργιών που παρέχονται από εξωτερικά API ή οποιεσδήποτε άλλες εργασίες προσανατολισμένες σε μεμονωμένες ανεξάρτητες υπηρεσίες. Αυτές οι υπηρεσίες έχουν σχεδιαστεί ώστε να είναι επαναχρησιμοποιήσιμες και αρθρωτές, επιτρέποντας την εύκολη κατανάλωσή τους από διάφορα μέρη της εφαρμογής, όπως Controllers, άλλα Services ή Background Tasks.

Ο όρος "dependency injection" (DI) αναφέρεται στην πρακτική του .NET Core 6.0 να εισάγει αυτόματα τα απαιτούμενα services ως εξαρτήσεις (dependencies) σε ένα αντικείμενο που πρέπει να κάνει χρήση του. Στη .NET Core 6.0, η διαχείριση των εξαρτήσεων αυτών γίνεται μέσω του μηχανισμού του Dependency Injection Container (DIC), όπου τα απαιτούμενα dependencies (όπως οι υπηρεσίες) παρέχονται αυτόματα στα μέρη που χρειάζονται.

Το DI με το DIC που εφαρμόζει η .NET Core 6.0 αναπτύχθηκε ώστε να επιτρέπει την χρήση και κατανάλωση των εξαρτήσεων με έναν ενιαίο τρόπο δίχως να κάνει δύσκολη την ζωή του προγραμματιστή. Ουσιαστικά το DIC έρχεται να αποκρύψει την low-level λειτουργικότητα που πρέπει να εκτελείται κάθε φορά όταν θα πρέπει να δηλωθεί ένα στοιχείο ως εξάρτηση σε ένα άλλο στοιχείο, επιτρέποντας τον προγραμματιστή να επικεντρωθεί με την πραγματική υλοποίηση του έργου του και όχι με διαδικασίες που ορίζονται από το Framework για τόσο low-level λειτουργίες. Το DIC μπορεί να χαρακτηριστεί ως ένα ξεχωριστό Framework στο περιβάλλον της .NET και καθίσταται απαραίτητη η κατανόηση λειτουργίας του απο οποιονδήποτε θέλει να κάνει αποτελεσματική χρήση του .NET Framework.

Το επόμενο βήμα που πρέπει να γίνει ώστε να δηλωθεί ένα Service στο DIC είναι η δήλωση του “για πόσο θα ζει” ή αλλιώς Service Lifetimes, κατά την εκτέλεση της εφαρμογής.

4.2.3 Lifetimes

Το πρώτο αρχείο που καλείται κατά το run-time του προγράμματος είναι ένα αρχείο που ονομάζεται Program.cs. Αυτό το αρχείο περιλαμβάνει όλες τις δηλώσεις λειτουργιών, από τη σύνδεση της εφαρμογής με τη βάση δεδομένων μέχρι τις δηλώσεις του Cors Policy, των Services και των Lifetimes αυτών καθώς και πολλά άλλα.

Ως Service Lifetime ορίζεται ο χρόνος ζωής που θα ζει ένα Service και θα είναι διαθέσιμο προς χρήση από ένα στοιχείο στο οποίο έχει δηλωθεί ως εξάρτηση. Με το πέρας του χρόνου αυτού το αντικείμενο του Service (Service Instance) που δημιουργείται καταστρέφεται και όλα τα δεδομένα και μνήμη που καταλαμβάνει απελευθερώνονται. Ωστόσο το Framework παρέχει διαφορετικά τέτοια Lifetimes που μπορούν να εξυπηρετούν διαφορετικά Services για διαφορετικές λειτουργίες ανάλογα με τις ανάγκες υλοποίησης της εκάστοτε εφαρμογής. Αυτά είναι τρία (3) στο πλήθος και περιγράφονται απο την Microsoft ως εξής:

- Singleton Lifetime: Το Service που δηλώνεται ως Singleton δεσμεύει τους πόρους που χρειάζεται και είναι διαθέσιμο απο το StartUp μέχρι και την λήξη εκτέλεσης της εφαρμογής.
- Transient Lifetime: Το Service που δηλώνεται ως Transient δεσμεύει τους πόρους που χρειάζεται και είναι διαθέσιμο μόνο στην περίπτωση που ζητηθεί για εκτέλεση μια λειτουργία που στεγάζεται μέσα του απο κάποιο στοιχείο που του έχει δηλωθεί ως εξάρτηση. Με το πέρας της εκτέλεσης της λειτουργίας αυτής, οι πόροι που καταλαμβάνει απελευθερώνονται και το αντικείμενο του Service παύει πλέον να υπάρχει.

- Scoped Lifetime: Είναι ένα ενδιάμεσο Liifetime που παρουσιάζει βελτιωμένες λειτουργίες και χαρακτηριστικά του Singleton και του Transient Lifetime συνδυαστικά. Το scoped service δημιουργείται κατα το αρχικό HTTP Request ενός πόρους και απελευθερώνει τους πόρους που δεσμεύει μετά την αποστολή της απόκρισης για αυτό το HTTP Request. Είναι το Lifetime που προτείνεται από την ίδια την Microsoft για κατα κόρον χρήση σε Services που χρησιμοποιούνται σε εφαρμογές τύπου Web API.

Στην εν λόγω εφαρμογή, όλα τα services, εκτός από το Service που υλοποιεί τη λειτουργία της In-Memory Cache, έχουν δηλωθεί ως Scoped. Αυτό σημαίνει ότι τα services αυτά δημιουργούνται κατά την παραλάβει του HTTP Request απο το Backend, χρησιμοποιούνται, απαντούν σε αιτήσεις και στη συνέχεια καταστρέφονται, ελευθερώνοντας χώρο στη RAM του server με την τελική επιστροφή του Http Response σε αυτόν που υπέβαλε το HTTP Request. Με αυτόν τον τρόπο, η εφαρμογή λειτουργεί με κατάλληλη αποδοτικότητα χωρίς να υπερφορτώνεται το σύστημα με πληροφορίες από services που μένουν ενεργά και δεν χρειάζονται άμεσα.

Όσον αφορά τη λειτουργία της In-Memory Cache, αναγκαστικά έχει δηλωθεί με Singleton Lifetime. Το παραπάνω είναι απαραίτητο για τη λειτουργία της τεχνικής caching, καθώς πρέπει να παραμένει ενεργή και να ανταποκρίνεται στην ανταλλαγή πληροφοριών καθ' όλη τη διάρκεια ζωής του προγράμματος. Με την χρήση της τεχνικής caching, μειώνονται σημαντικά οι βαριές και χρονοβόρες λειτουργίες στη βάση δεδομένων για πληροφορίες που είναι κοινές και όχι συχνά μεταβαλλόμενες. Ένα παράδειγμα είναι οι πληροφορίες για τα μαθήματα κάθε εξαμήνου που θα πρέπει να εμφανίζονται σε όλους τους χρήστες. Αντί να γίνεται συνεχής κλήση στη βάση δεδομένων ανά χρήστη για να ανακτηθούν τα μαθήματα αυτά, η πληροφορία κρατείται στην cache από την πρώτη κλήση και επιστρέφεται στους υπόλοιπους χρήστες που θα την αναζητήσουν αργότερα καθώς υπό φυσιολογικά πλαίσια δεν θα αλλάξουν οι πληροφορίες των μαθήματων αυτών.

4.2.4 Entity Framework Core

Το Entity Framework Core είναι ένα ισχυρό και δημοφιλές πλαίσιο αντικειμενοσχεσιακής απεικόνισης (ORM) που αναπτύχθηκε επίσης από τη Microsoft. Πρόκειται για μια ελαφριά, διαπλατφορμική έκδοση του Entity Framework που επιτρέπει στους προγραμματιστές να αλληλοεπιδρούν με σχεσιακές βάσεις δεδομένων χρησιμοποιώντας το .NET Core Framework. Το EF Core εν συντομία, παρέχει ένα ισχυρό σύνολο χαρακτηριστικών που απλοποιούν την πρόσβαση και τον χειρισμό δεδομένων, επιτρέποντας στους προγραμματιστές να επικεντρωθούν περισσότερο στην επιχειρησιακή λογική παρά στις λειτουργίες χαμηλού επιπέδου της βάσης δεδομένων που χρησιμοποιούν.

Το EF Core προσφέρει μια σειρά χαρακτηριστικών και πλεονεκτημάτων που το καθιστούν προτιμώμενη επιλογή για πολλούς. Πρώτον, υποστηρίζει πολλαπλούς παρόχους βάσεων δεδομένων, συμπεριλαμβανομένων των SQL Server, SQLite, MySQL, PostgreSQL και άλλων, επιτρέποντας την εργασία με διάφορα συστήματα βάσεων δεδομένων χρησιμοποιώντας ένα συνεπές και καθολικό API. Αυτή η ευελιξία επιτρέπει την ανάπτυξη εφαρμογών σε διαφορετικές πλατφόρμες και περιβάλλοντα.

Το EF Core ακολουθεί την αρχή convention-over-configuration, η οποία σημαίνει ότι συμπεραίνει αυτόματα αντιστοιχίσεις μεταξύ των πινάκων της βάσης δεδομένων και των κλάσεων οντοτήτων-συσχετίσεων με βάση συμβάσεων ονοματοδοσίας (Name Conventions). Ωστόσο, παρέχει επίσης ένα εύχρηστο API που ονομάζεται FluentAPI το οποίο επιτρέπει στους προγραμματιστές να ορίζουν ρητά αντιστοιχίσεις, σχέσεις και άλλες επιλογές διαμόρφωσης όταν αυτό κριθεί αναγκαίο. Αυτός ο συνδυασμός σύμβασης και διαμόρφωσης παρέχει τον πλήρη έλεγχο στην διαδικασία δημιουργίας ενός σχεσιακού σχήματος.

Ένα άλλο σημαντικό χαρακτηριστικό του EF Core είναι η υποστήριξή που παρέχει για migrations. Τα migrations επιτρέπουν στους προγραμματιστές να εξελίσσουν το σχήμα της βάσης δεδομένων με την πάροδο του χρόνου δημιουργώντας σταδιακές αλλαγές στα μοντέλα/πίνακες. Το EF Core μπορεί να δημιουργήσει τα σενάρια SQL που είναι απαραίτητα για την ενημέρωση του σχήματος της βάσης δεδομένων με βάση τις αλλαγές που γίνονται στις κλάσεις οντοτήτων. Αυτό απλοποιεί τη διαδικασία διαχείρισης των αλλαγών στο σχήμα της βάσης δεδομένων, ειδικά σε έργα με πολλούς προγραμματιστές ή σε περιβάλλοντα παραγωγής.

Το EF Core υποστηρίζει επίσης το LINQ (Language Integrated Query), το οποίο επιτρέπει την υποβολή SQL ερωτημάτων και την επεξεργασία δεδομένων χρησιμοποιώντας ένα έντονα τυποποιημένο και αντικειμενοστραφές συντακτικό. Αυτή η προσέγγιση εξαλείφει την ανάγκη συγγραφής ακατέργαστων ερωτημάτων SQL και παρέχει ασφάλεια κατά τη μεταγλώττιση και στην υποστήριξη της Microsoft Intellisense. Τα ερωτήματα LINQ μεταφράζονται από το EF Core στις κατάλληλες ακατέργαστες εντολές SQL βάση του DBMS που έχει δηλωθεί προς χρήση, βελτιστοποιώντας την εκτέλεση και την απόδοση τους.

Εκτός από την αναζήτηση δεδομένων, το EF Core παρέχει επίσης δυνατότητες εισαγωγής, ενημέρωσης και διαγραφής εγγραφών. Περιλαμβάνει έναν μηχανισμό παρακολούθησης αλλαγών (Tracking) ο οποίος παρακολουθεί τις αλλαγές που πραγματοποιούνται στις οντότητες και παράγει αυτόματα τις απαραίτητες εντολές SQL για τη διατήρηση και αποθήκευση αυτών των αλλαγών στη βάση δεδομένων.

Επιπλέον, το EF Core υποστηρίζει ασύγχρονες λειτουργίες, οι οποίες μπορούν να βελτιώσουν την απόδοση και την ανταπόκριση των εφαρμογών, ιδίως σε σενάρια όπου επεξεργάζονται πολλές ταυτόχρονες αιτήσεις προς την βάση δεδομένων απο την εφαρμογή.

Το EF Core συντηρείται ενεργά και διαθέτει ένα αυξανόμενο οικοσύστημα επεκτάσεων και βιβλιοθηκών τρίτων που παρέχουν πρόσθετη λειτουργικότητα, όπως strict type checking, προσωρινή αποθήκευση και βελτιωμένο Tracking.

Συνολικά, το EF Core είναι ένα ισχυρό και ευέλικτο ORM Framework για εφαρμογές στο περιβάλλον της .NET. Απλοποιεί την πρόσβαση στα δεδομένα, μειώνει την ποσότητα του boilerplate κώδικα και παρέχει ένα συνεπές API σε διαφορετικούς παρόχους βάσεων δεδομένων. Είτε δημιουργείτε μια μικρή εφαρμογή είτε ένα μεγάλο εταιρικό σύστημα, το EF Core μπορεί να βοηθήσει στη βελτίωση της παραγωγικότητα και στην επικέντρωση στην ανάπτυξη των βασικών χαρακτηριστικών της εφαρμογής χωρίς χαμένο χρόνο στις περίπλοκες ενέργειες πρόσβασης στα δεδομένα.

4.2.5 Middleware Pipeline – Middlewares

Τα Middlewares κατέχουν ανώτερο ιεραρχικά ρόλο, και ορίζονται ως εννοιολογικά αυτοτελή και αποκεντρωμένα κομμάτια κώδικα εκτελώντας μία μοναδική λειτουργία που θα τους οριστεί. Συνήθως δεν υπάρχει μόνο ένα σε μία εφαρμογή και εκτελούνται ως σύνολο μέσα από μία σειριακή σειρά εκτέλεσης που ονομάζεται Middleware Pipeline. Ως Middleware Pipeline ορίζεται η συνεχής ροή εκτέλεση πολλαπλών middleware συστατικών βάσει ενός Http αιτήματος κατα κύριο λόγο. Υπό την ολοκλήρωση εκτέλεσης του pipeline η αίτηση μαρκάρετε επιτυχώς ή όχι και είτε της επιτρέπεται να προχωρήσει στα ενδότερα της εφαρμογής είτε όχι.

Το Middleware οπότε, χρησιμοποιείται στην εφαρμογή ως ένας ενδιάμεσος που εκτελείται ιεραρχικά πρώτος κατα την παραλαβή της αίτησης και επιτρέπει τον έλεγχο και την επεξεργασία διαφόρων παραμέτρων των αιτημάτων πριν φτάσουν στον βασικό κώδικα της εφαρμογής ή πριν αποσταλούν στον

χρήστη. Σε μια εφαρμογή βασισμένη στην .NET Core 6.0, μια χρήση ενός Middleware μπορεί είναι για τον έλεγχο πρόσβασης ενός χρήστη σε πόρους που είναι προσβάσιμοι μόνο σε αυθεντικοποιημένους χρήστες.

Ένα πιθανό σενάριο είναι η χρήση ενός τέτοιου Middleware για τον έλεγχο των Cookies που μεταφέρονται από τον χρήστη. Αν ένας χρήστης δεν έχει ακολουθήσει την διαδικασία δημιουργίας ένα απαιτούμενο Cookie, τότε ο έλεγχος αυθεντικοποίησης δεν χρειάζεται να γίνει εξ ολοκλήρου μέσα στον κώδικα της εφαρμογής. Αντίθετα, μπορεί να πραγματοποιηθεί έλεγχος κατά την παραλαβή της αιτήσεως του χρήστη για έναν προστατευμένο πόρο και, αν δεν υπάρχει το αναμενόμενο Token, να επιστρέφεται ένα μήνυμα σφάλματος με Http Status Code "401 Unauthorized". Το παραπάνω υποθετικό σενάριο πρέπει να εκτελείται πάντα υπό μία μορφή πάνω σε όλα τα ευαίσθητα σημεία και δεδομένα της εφαρμογής, εκτός όμως από εκείνους τους πόρους όπου τέτοιου είδους αυθεντικοποίηση δεν απαιτείται. Φυσικά ένα αντίστοιχο Middleware ελέγχει την πρόσβαση σε προστατευμένους πόρους και στην παρούσα εφαρμογή.

4.2.6 Δημιουργία Σχήματος

Το EF Core παρέχει μεγάλη αυτοματοποίηση στην εκ νέου δημιουργία ενός ολόκληρου σχεσιακού σχήματος που περιέχει περίπλοκες εξαρτήσεις.

Η εφαρμογή έχει σχεδιαστεί με το Code-First approach. Το Code-First approach υποδηλώνει ότι πρώτα χτίζονται τα μοντέλα / κλάσεις σε κώδικα και έπειτα μέσω τετριμμένων διαδικασιών το σχήμα δημιουργείται εξ ολοκλήρου αυτόματα από το EF Core ακολουθώντας τις οδηγίες του προγραμματιστή.

Με την παραπάνω τακτική επιτυγχάνεται η εύκολη δημιουργία του σχήματος χάρη στο EF Core. Ο τύπος κάθε πεδίου της κλάσης μοντέλου μπορεί να είναι είτε πρωταρχικού τύπου είτε σύνθετου, πράγμα που υποστηρίζεται εξ ολοκλήρου απο το EF Core. Δημιουργώντας το μοντέλο / κλάση που πρέπει να αποτυπωθεί στη βάση δεδομένων υπό τη μορφή πίνακα, μπορεί να εκκινηθεί το επόμενο στάδιο δημιουργίας. Το στάδιο αυτό εκκινείτε με την εντολή "EntityFrameworkCore\Add-Migration", όπου η επιτυχία εκτέλεσης δημιουργεί το σκαρίφημα του σχήματος που πρόκειται να δημιουργηθεί τελικά και να αποτυπωθεί στην βάση αυτή καθαυτή υπό την μορφή πινάκων. Επιπλέον, κατά την ολοκλήρωση της διαδικασίας του Migration, υπάρχει μια πλήρη εικόνα όχι μόνο των πινάκων που επρόκειτο να δημιουργηθούν αλλά ,των τύπων κάθε πεδίου, των περιορισμών των ξένων και κύριων κλειδιών καθώς και των δεικτών (indexes) που δημιουργούνται αυτόματα το Entity Framework Core.

Τέλος, η διαδικασία επικυρώνεται οριστικά και απεικονίζεται στην συνδεδεμένη βάση με την εντολή "EntityFrameworkCore\Update-Database".

4.2.7 Αυθεντικοποιηση με βάση Cookie και Token

Κατά την είσοδο του χρήστη στην εφαρμογή, γίνεται μια κλήση στον εξυπηρετητή του πανεπιστημίου για αυθεντικοποίηση ακριβώς όπως έχει περιγράφει σε προηγούμενο κεφάλαιο. Αυτή η κλήση επιστρέφει ένα Token, το οποίο περιέχει τις πληροφορίες του χρήστη. Αρχικά, γίνεται η εισαγωγή του χρήστη στη βάση δεδομένων της εφαρμογής, παράλληλα με την δημιουργία ενός νέου Cookie.

Το Cookie αποθηκεύεται τοπικά στον φυλλομετρητή του χρήστη και μπορεί να διαγραφεί με τρεις τρόπους:

- Κατά τη λήξη του: Το Cookie έχει μια διάρκεια ζωής μίας ημέρας από την ημερομηνία δημιουργίας του. Αν ο χρήστης δεν συνδεθεί ξανά μέσα σε αυτό το διάστημα, τότε το Cookie αυτομάτως διαγράφεται από τον φυλλομετρητή.

- Χειροκίνητη Διαγραφή από τον φυλλομετρητή: Ο χρήστης έχει τη δυνατότητα να διαγράψει το Cookie από τον φυλλομετρητή. Σε αυτή την περίπτωση, όταν επανασυνδεθεί, θα χρειαστεί να πραγματοποιήσει νέα αυθεντικοποίηση και να δημιουργηθεί εκ νέου ένα Cookie.
- Αποσύνδεση από την εφαρμογή: Όταν ο χρήστης αποσυνδέεται από την εφαρμογή, το Cookie αυτόματα καταργείται. Σε αυτή την περίπτωση, αν ο χρήστης συνδεθεί ξανά, θα χρειαστεί να πραγματοποιήσει νέα αυθεντικοποίηση και να λάβει ένα νέο Cookie για την πλοήγησή του στην εφαρμογή.

Το εν λόγω Cookie αυτό καθαυτό δεν είναι τίποτα παραπάνω απο το μεταφορικό μέσο που μεταφέρει συγκεκριμένα μία πληροφορία απο το FrontEnd στο BackEnd της εφαρμογής σε κάθε HTTP Request που εν γένη υποβάλλεται για διεκπεραίωση απο το BackEnd.

Η πληροφορία που μεταφέρεται από το Cookie είναι ένα Token. Το Token αυτό υπάγεται κάτω από την τεχνική του Web Token Authorization που χρησιμοποιείται σε web εφαρμογές.

Το Token αυτό δημιουργείται εξ ολοκλήρου απο το BackEnd της εφαρμογής και κουβαλάει κρυπτογραφημένες πληροφορίες για τον χρήστη όπως ο ρόλος του (π.χ. καθηγητής, φοιτητής, διαχειριστής), το ΑΜ του και άλλες χρήσιμες πληροφορίες όπως εσωτερικές αναγνωρίσεις (Internal Ids) , οι οποίες χρησιμοποιούνται για την αναγνώριση και διαχείριση των χρηστών στην εφαρμογή.

Το μοντέλο κρυπτογράφησης που χρησιμοποιείται για την κρυπτογράφηση του Token είναι το Json Web Token Format (JWT), όπου με συγκεκριμένο ισχυρό ασύμμετρο αλγόριθμο εγγυάται την ασφαλή διακίνηση ευαίσθητων δεδομένων και την δυσκολία αθέμιτης αποκρυπτογράφησης των πληροφοριών από τρίτους.

Οπότε συνοψίζοντας τα παραπάνω, σε ένα υποθετικό σενάριο πρόσβασης ενός χρήστη που ισχυρίζεται ότι είναι αυθεντικοποιημένος και απαιτεί πρόσβαση σε έναν ευαίσθητο πόρο της εφαρμογής , αποστέλλεται το συγκεκριμένο Token στο BackEnd μέσω Cookie και εφόσον δεν έχει λήξει, δεν έχει “‘πειραχτεί” από κάποιο λογισμικό και φέρει πληροφορία έγκυρη που αντιστοιχίζεται με μία επίσης έγκυρη εγγραφή χρήστη στην βάση δεδομένων ο έλεγχος εγκυρότητας είναι επιτυχημένος και αποκτάται πρόσβαση στον εν λόγω πόρο για τον χρήστη που το ζήτησε. Η παραπάνω λειτουργία είναι εξ ολοκλήρου υλοποιημένη απο ένα αποκεντρωμένο Middleware που προστατεύει συγκεκριμένους πόρους και εκτελεί μόνο τετριμμένους λειτουργικούς ελέγχους.

4.2.8 Αλγόριθμος Δικαιοσύνης

Ένα από τα μεγαλύτερα προβλήματα που προέκυψαν κατά τη δημιουργία της εφαρμογής ήταν η ανάγκη κάλυψης της απαίτησης για ένα σύστημα δικαιοσύνης που θα ισχύει για όλους τους φοιτητές κατά την περίοδο των δηλώσεων. Για την επίλυση αυτού του προβλήματος, αναπτύχθηκε ένας αλγόριθμος που παράγει ορθό αποτελέσματα και εξαρτάται από το Ε.Φ, τις Δ.Μ. και το δεκαδικό αποτέλεσμα μιας φόρμουλας που ονομάστηκε Ρυθμός Προόδου (Ρ.Π.).

Το αποτέλεσμα του παραπάνω αλγορίθμου είναι η κατανομή του φοιτητή σε ένα ημερομηνιακό γκρουπ βάση προτεραιότητας. Το κάθε γκρουπ έχει διαφορετική ημερομηνία έναρξης, όμως όλα έχουν μια κοινή ημερομηνία λήξης που δεν είναι άλλη από την ημερομηνία λήξης της περιόδου δηλώσεων. Τα γκρουπ και οι τρεις ομάδες προτεραιότητας είναι αντιστοιχισμένες αναλογικά ως εξής.

Έστω ότι ημερομηνία έναρξης δηλώσεων ορίζεται η 01.04.2023 και ημερομηνία λήξης η 30.04.23.

Οι υπολογιζόμενές ημερομηνιακά θυρίδες θα έχουν την εξής κατανομή με βάση την προτεραιότητα που ξεκινάει από την Υψηλότερη για την πρώτη ημερομηνιακή θυρίδα και την Χαμηλότερη για την τελευταία ημερομηνιακή θυρίδα.

Η ημερομηνιακή απόσταση της ημερομηνίας έναρξης μιας ομάδας από την επόμενη κατα σειρά υπολογίζεται βάση της ακέραιας διαίρεσης των συνολικών ημερών διενέργειας της περιόδου που έχει δηλωθεί δια το πλήθος των ομάδων προτεραιοτήτων που πρέπει να μοιραστούν και στρογγυλοποίηση του αποτελέσματος της διαιρέσεως πάντα προς τον επόμενο ακέραιο αριθμό.

Ο παραπάνω υπολογισμός σε κώδικα C# αποτυπώνεται στο Παράρτημα Β του κειμένου. Η παραπάνω επιτυχημένη εκτέλεση των γραμμών κώδικα που αναφέρονται στο Παράρτημα Β διαμορφώνει το παρακάτω ημερομηνιακό χρονοδιάγραμμα που θα αποθηκευτεί στη βάση δεδομένων.

01.04.2023 - 30.04.2023 -> Υψηλή Προτεραιότητα

10.04.2023 - 30.04.2023 -> Μέτρια Προτεραιότητα

19.04.2023 - 30.04.2023 -> Χαμηλότερη Προτεραιότητα

Ο αλγόριθμος όπως προαναφέρθηκε απαιτεί την ύπαρξη κρίσιμων πληροφοριών του φοιτητή για την δίκαιη κατανομή του σε ένα από τα παραπάνω γκρουπ. Χάρη σε τρίτη εφαρμογή του Τμήματος που έχει αναφερθεί πρωτίστως, αποκτάται ο αριθμός των διδακτικών μονάδων που έχει ο κάθε χρήστης μέχρι το κατα περίπτωση προηγούμενο εξάμηνο, κατά την είσοδό του στην εφαρμογή καλύπτοντας την πρώτη απαίτηση του αλγορίθμου.

Η δεύτερη απαίτηση, το εξάμηνο φοίτησης καλύπτεται, μέσω του HTTP Response που λαμβάνεται κατά την είσοδο του φοιτητή στην εφαρμογή και αποτυπώνεται στην βάση δεδομένων.

Με βάση τις δύο πληροφορίες, υπολογίζεται ένας ρυθμό προόδου που λαμβάνει υπόψη τον τυπικό αριθμό διδακτικών μονάδων που κατέχει κάθε χρήστης και το εξάμηνο που βρίσκεται, καθώς και τον μέγιστο αριθμό δηλωμένων διδακτικών μονάδων που μπορεί να δηλώσει σε κάθε εξάμηνο. Το αποτέλεσμα της φόρμουλας είναι αυστηρά δεκαδικό αποτέλεσμα και πάντα μεγαλύτερο του μηδενός.

Η τελική φόρμουλα του ρυθμού Προόδου έχει την ακόλουθη μορφή:

$$\textbf{Ρυθμός Προόδου(Ρ.Π.)}=\frac{\textbf{Αριθμός Διδακτικών Μονάδων Φοιτή (Α.Δ.Μ.Φ)}}{\textbf{(Εξάμηνο Φοίτησης (Ε.Φ) − 1)} * \textbf{(Μέγιστες Επιτρεπόμενες Προς Δήλωση Δ.Μ.Ανά Εξάμηνο (Μ.Ε.Π.Δ.Δ.Μ.Α.Ε))}}$$

Σχήμα 4.1: Φόρμουλα Υπολογισμού Ρυθμού Προόδου

Η λογική του αλγορίθμου δικαιοσύνης συνοπτικά εκτελείται σειριακά ώστε να αποδοθεί η κατάλληλη προτεραιότητα / γκρουπ στον φοιτητή. Οι προτεραιότητες κατανέμονται ως εξής:

- **ΑΝ** ο φοιτητής είναι Α εξαμήνου => Ύψιστη Προτεραιότητα.

Για εξάμηνα μεγαλύτερα του Α και μικρότερα ίσα του Θ:

- **ΑΝ** (Α.Δ.Μ.Φ.>= Μ.Σ.Δ.Μ.Ε.) **Η** Ρ.Π. >= 0.65 => Ύψιστη Προτεραιότητα.
- **ΑΛΛΙΩΣ ΑΝ** Α.Δ.Μ.Φ. < Μ.Σ.Δ.Μ.Ε. **ΚΑΙ** 0.35 < Ρ.Π. < 0.65 => Μέτρια Προτεραιότητα.
- **ΑΛΛΙΩΣ** => Χαμηλή Προτεραιότητα

Για μεγαλύτερα εξάμηνα του Θ ο έλεγχος γίνεται μόνο τον Ρ.Π οπότε έχουμε την εξής διαμόρφωση:

- **ΑΝ** Ρ.Π. >= 0.65 **ΚΑΙ** το εξάμηνο του φοιτητή είναι >= 9 => Ύψιστη Προτεραιότητα.
- **ΑΛΛΙΩΣ ΑΝ** 0.35 < ΡΠ < 0.65 **ΚΑΙ** το εξάμηνο του φοιτητή είναι >= 9 => Μέτρια Προτεραιότητα.
- **ΑΛΛΙΩΣ**=> Χαμηλή Προτεραιότητα.

Ας σημειωθεί ότι τα ποσοστά 0.65 και 0.35 ως ανώτερο και κατώτερο καθοριστικό σημείου απόφασης έχουν συμφωνηθεί απο το επιβλέποντα καθηγητή της πτυχιακής εργασίας μετά από εκτεταμένες επαναλήψεις εκτέλεσης του αλγορίθμου για διαφορετικά παραδείγματα. Επιπλέον οι δύο αυτές δεκαδικές τιμές είναι μεταβαλλόμενες μέσα από ένα Αρχείο Παραμετροποίησης του BackEnd και όχι στατικές. Αυτό σημαίνει ότι μπορούν να αλλάξουν με εύκολο τρόπο σε περίπτωση που αποδειχθεί ότι ο αλγόριθμος φέρει πιο δίκαια αποτελέσματα με διαφορετικές τιμές.

Ακολουθούν δύο (2) παραδείγματα φοιτητών για να αποσαφηνιστεί η κατάσταση:

Όταν ένα εξάμηνο λήγει και ξεκινάει το επόμενο, το σύστημα ανανεώνει το εξάμηνο του φοιτητή μόνο αν το τρέχων εξάμηνο έχει ανανεωθεί και στην υπηρεσία του Apps όπου και χρησιμοποιεί η εφαρμογή. Για να γίνει ο σωστός υπολογισμός του Ρ.Π. πρέπει το τρέχον εξάμηνο να είναι πάντα στο μείον ένα (-1) από τρέχων λόγο του ότι ιδανικά θα πρέπει να λαμβάνεται υπόψιν η μέχρις στιγμής επικυρωμένη πορεία του . Ο υπολογισμός του Μ.Ε.Π.Δ.Δ.Μ.Α.Ε. πιο πάνω οπού βάση ψηφίων του ΑΜ ο φοιτητής καταλογίζεται στο κατάλληλο πρόγραμμα σπουδών που ακολουθεί (Α.Τ.Ε.Ι.Θ / ΔΙ.ΠΑ.Ε). Ο παραπάνω έλεγχος χωρίζεται σε 2 ακόμη υποκατηγορίες όπου είναι οι εξής:

- Οι φοιτητές του Πρώην Τμήματος που είχαν μέχρι 160 Δ.Μ. είχαν δικαίωμα να δηλώσουν μέχρι 30 μονάδες ανά εξάμηνο. Σε περίπτωση που έχει πάρα πάνω από 160 Δ.Μ. τότε έχει δικαίωμα να δηλώσει μέχρι 42 ανά εξάμηνο.
- Οι φοιτητές του Νέο Τμήματος που έχουν μέχρι 210 Δ.Μ. έχουν δικαίωμα να δηλώσουν μέχρι 30 αντίστοιχα και για πάνω από 210 Δ.Μ. έχουν δικαίωμα να δηλώσουν μέχρι 42 Δ.Μ. ανά εξάμηνο.

Επίσης σε κάθε αλλαγή εξάμηνου υπολογίζονται εκ νέου πόσες οι Δ.Μ. που ο εκάστοτε φοιτητής θα είχε συμπληρώσει αν είχε παρακολουθήσει επιτυχώς όλα τα μέχρι στιγμής μαθήματα, μέτρο που αναφέρεται ως Μ.Σ.Δ.Μ.Ε (Μέγιστες Συμπληρωμένες Διδακτικές Μονάδες Εξαμήνου). Για παράδειγμα, σε φοιτητή Γ εξάμηνου πραγματοποιείται ο υπολογισμός (30 ή 42) * (3-1) με αποτέλεσμα 60 ή 82 μονάδες (αναλόγως το πρόγραμμα σπουδών), όπου και τελικά με την συνδρομή του αποτελέσματος αυτού και του υπολογιζόμενου Ρ.Π μπορεί να ληφθεί η απόφαση.

Αν ένας φοιτητής, φοιτεί στο παλιό πρόγραμμα σπουδών στο Γ τυπικό εξάμηνο, που έχει περάσει όλα τα μαθήματα των προηγουμένων 2 εξάμηνων φοίτησης επιτυχώς έχει 60 Δ.Μ. Όταν ξεκινήσει ο υπολογισμός των προτεραιοτήτων αυτός ο φοιτητής θα έχει:

Δ.Μ >= Μ.Σ.Δ.Μ.Ε (60 =60) **ΚΑΙ** Ρ.Π. >= 65, όπου Ρ.Π. = 60 / (2-1) * 30 = 2.0. Ο φοιτητής αποφασίζεται ότι λαμβάνει την Υψίστη Προτεραιότητα.

Τώρα αν ο φοιτητής δεν ήταν τόσο συνεπείς στις υποχρεώσεις και έχει περάσει μόνο τα πέντε (5) από τα δέκα (10) μαθήματα αυτά τα δυο (2) εξάμηνα τότε ο υπολογισμός γίνεται ως εξής:

Δ.Μ < Μ.Σ.Δ.Μ.Ε (30 < 60) **ΚΑΙ** Ρ.Π. >= 0.35 **ΚΑΙ** Ρ.Π. < 0.65, όπου Ρ.Π. = 30 / (2-1) * 30 = 0,5 => Ο φοιτητής λαμβάνει την Μέτρια Προτεραιότητα.

Αντιστοίχως αν ο φοιτητής δεν πληρεί καμία από τις δύο παραπάνω συνθήκες λαμβάνει την Χαμηλή Προτεραιότητα.

Στον φοιτητής με εξάμηνο μεγαλύτερο του 8 τότε δεν συμπεριλαμβάνεται η λογική του Μ.Σ.Δ.Μ.Ε και γίνεται αποκλειστικός έλεγχος μόνο του Ρ.Π.

Έστω, ο φοιτητής είναι στο εξάμηνο 9 της σχολής και έχει Ρ.Π. = 240 / (9-1) * 30 = 1 => οπότε παίρνει Μέγιστη Προτεραιότητα.

Αντίστοιχα αν έχει Ρ.Π < 0,65 **ΚΑΙ** Ρ.Π > 0,35 => Μέτρια Προτεραιότητά αλλιώς παίρνει την Χαμηλή Προτεραιότητα

Με τον τρόπο αυτό έχει προσομοιωθεί όσο δυνατόν καλύτερα ένα δίκαιο σύστημα που βασίζεται σε πλήρως παραμετρικό αλγόριθμο που αντιμετωπίζει όσο καλύτερα το πρόβλημα της δικαιοσύνης στις δηλώσεις των φοιτητών κατά την περίοδο των δηλώσεων, συγχωρώντας, τιμωρώντας και δίνοντας δεύτερες ευκαιρίες σε φοιτητές που όντως αποτυπώνουν την προσπάθεια τους.

4.2.9 Αρχεία Παραμετροποίησης / Configuration Files

Τα αρχεία παραμετροποιήσεις (configuration file) χρησιμοποιούνται για να παρέχουν γενικές παραμετροποιήσιμες ρυθμίσεις και παραμέτρους συστήματος στην εφαρμογή που αναπτύσσεται στο πλαίσιο μιας εφαρμογής. Αυτά τα αρχεία περιέχουν πληροφορίες και παραμέτρους ζωτικής σημασίας που μπορούν να προσαρμοστούν ανάλογα με τις απαιτήσεις του συστήματος στο παγκοσμίως αποδεκτό JSON Format.

Μία πληροφορία που χρησιμοποιείται καθολικά στην εφαρμογή και είναι ύψιστης σημασίας είναι οι ρυθμίσεις σύνδεσης στην βάση δεδομένων που χρησιμοποιείται. Σε αυτήν την περίπτωση, ένα .json αρχείο παραμετροποίησης περιέχει τις απαραίτητες πληροφορίες, όπως η διεύθυνση του διακομιστή όπου φιλοξενείται η βάση δεδομένων, το όνομα της βάσης και τα διαπιστευτήρια σύνδεσης. Με αυτόν τον τρόπο, η εφαρμογή μπορεί να ανακτά αυτές τις πληροφορίες μέσω μιας κλήσης στο αρχείο κατά την εκκίνηση του προγράμματος και να χρησιμοποιεί την πληροφορία για την σύνδεση στη βάση δεδομένων αυτόματα μέχρι την λήξη λειτουργίας της.

Ο βασικός σκοπός των αρχείων παραμετροποίησης είναι να διευκολύνει την προσαρμογή και τη διαμόρφωση της εφαρμογής, χωρίς να απαιτείται καμία τροποποίηση του πηγαίου κώδικα. Αυτό επιτρέπει σε ρυθμίσεις να αλλάζουν παραμετρικά κατα το runtime της εφαρμογής, χωρίς πρώτον να επηρεάζεται ο κώδικας και δεύτερον χωρίς να χρειαστεί να γίνει restart της εφαρμογής καθαυτής. Οι παραπάνω λειτουργίες οπότε, προσφέρουν έναν ευέλικτο τρόπο παραμετροποίησης του συστήματος σε εκάστοτε διαφορετικές απαιτήσεις. Επίσης υπάρχει η δυνατότητα να προστεθούν περαιτέρω παράμετροι και ρυθμίσεις σε διάφορα αρχεία διαμόρφωσης, κάνοντας την εφαρμογή ακόμη πιο αφηρημένα συνδεδεμένη απο τις υποκείμενες λειτουργίες που χρησιμοποιεί.

Ακόμη δύο ρυθμίσεις μεταξύ άλλων που έχουν χρησιμοποιηθεί και είναι ζωτικής σημασίας για την εφαρμογή πέρα από το ανώτερο και κατώτερο όριο του Ρ.Π. απο τον Αλγόριθμο Δικαιοσύνης, είναι οι αποδεκτοί τύποι αρχείων και mime types που επιτρέπονται να μεταφορτωθούν στην εφαρμογή καθώς και τα URI επικοινωνίας με την Τρίτη Υπηρεσία του Τμήματος που είναι υπεύθυνη για το σερβίρισμα των Δ.Μ. του κάθε φοιτητή.

Με την ανύψωση των παραπάνω απαιτήσεων σε αρχεία παραμέτρων υπάρχει η δυνατότητα να αλλάξει το URI που πλέον σερβίρονται οι πόροι των Δ.Μ. κατα το runtime της εφαρμογής καθώς και πολύ εύκολα να προστεθεί ή να αφαιρεθεί ακόμη ένας τύπος αρχείου που κρίθηκε ότι πρέπει πλέον να εξυπηρετείται ή όχι.

Απο πιο τεχνολογική όψη η .NET Core 6.0 παρέχει τις παραπάνω δυνατότητες μέσω του προ εγκατεστημένου Service IOptionsMonitor<T> το οποίο γίνεται διαθέσιμο σε ένα σημείο του κώδικα μέσο του DIC που έχει προαναφερθεί. Όταν υπάρχει η απαίτησή να ανακτηθεί πληροφορία από ένα configuration file το IOptionsMonitor Service αναλαμβάνει την αναπαράστασή του απο την χρησιμοποιούμενη μορφή .json σε αρχείο .cs που μπορεί να προσπελαστεί επιτυχώς απο το Framework. Επιπλέον όταν γίνει κάποια αλλαγή στο configuration file κατά την εκτέλεση του προγράμματος το

Service αναλαμβάνει την κατα το runtime ενημέρωση των τιμών σε όλα τα σημεία κώδικα που αυτές χρησιμοποιούνται, αυτόματα χωρίς την απαίτηση επανεκκίνησης της εφαρμογής.

Τα αρχεία διαμόρφωσης μπορεί επίσης να περιλαμβάνουν και άλλες πληροφορίες όπως τις ρυθμίσεις γλώσσας, γενικές ρυθμίσεις εμφάνισης ή συμπεριφοράς της εφαρμογής, κανόνες για την ασφάλεια και την πρόσβαση, καθώς και παραμέτρους που επηρεάζουν άλλες επιμέρους συμπεριφορές της εφαρμογής.

Συνοψίζοντας οπότε, τα αρχεία διαμόρφωσης είναι ένα απλό αρχείο κειμένου σε μορφή JSON που περιέχουν κρίσιμες πληροφορίες, ρυθμίσεις και παραμέτρους που επηρεάζουν τη λειτουργία μιας εφαρμογής ή ενός συστήματος. Χρησιμοποιούνται για την ευκολία προσαρμογής και αλλαγής των ρυθμίσεων χωρίς την ανάγκη τροποποίησης του κώδικα, προσφέροντας μεγαλύτερη ευελιξία και δυνατότητα παραμετροποίησης της εφαρμογής.

4.2.10 Cron / Workers

Στο .NET Core 6.0, τόσο τα CronJobs όσο και οι Workers έχουν σημαντικό ρόλο στην εκτέλεση και τη διαχείριση των αυτοματοποιημένων εργασιών σε μια εφαρμογή.

Τα Cron, συντομογραφία του "χρονογράφου", ιστορικά είναι ένας χρονοπρογραμματιστής εκτέλεσης αυτοματοποιημένων εργασιών με βάση το χρόνο σε λειτουργικά συστήματα τύπου Unix. Επιτρέπει στους χρήστες να προγραμματίζουν και να αυτοματοποιούν την εκτέλεση επαναλαμβανόμενων εργασιών ή σεναρίων. Στο πλαίσιο μιας εφαρμογής .NET Core 6.0, το Cron αναφέρεται σε μια παρόμοια έννοια που υλοποιείται μέσω βιβλιοθηκών. Επιτρέπει στους προγραμματιστές να προγραμματίζουν και να εκτελούν συγκεκριμένες εργασίες σε προκαθορισμένα χρονικά διαστήματα ή σύμφωνα με ένα καθορισμένο χρονοδιάγραμμα. Αυτό είναι ιδιαίτερα χρήσιμο για την αυτοματοποίηση διαδικασιών ρουτίνας, όπως η δημιουργία αντιγράφων ασφαλείας δεδομένων, η δημιουργία αναφορών ή ο συγχρονισμός δεδομένων.

Από την άλλη πλευρά, ένας Worker χρησιμοποιείται συνήθως για την αποφόρτιση εργασιών που διαφορετικά θα μπλοκάραν το Main Execution Thread και θα επηρέαζαν αρνητικά την απόκριση της εφαρμογής. Εκτελώντας αυτές τις εργασίες στο παρασκήνιο, οι workers διασφαλίζουν ότι η κύρια εφαρμογή παραμένει ευέλικτη και μπορεί να συνεχίσει να εξυπηρετεί τα αιτήματα των χρηστών.

Οι δύο παραπάνω έννοιες συνδυαστικά αναλαμβάνουν να εκτελέσουν διαφορετικά αυτοματοποιημένα καθήκοντα στο Backend υπό πλήρη συντονισμό.

Ένα Cron αναλαμβάνει κάθε μέρα στις 00:05 να ελέγξει αν η συγκεκριμένη ημερομηνία χρήζει συγκεκριμένης σημασίας για την περίοδο δηλώσεων. Με τον ορό συγκεκριμένης σημασίας εννοείται ότι μια ημερομηνία μπορεί να είναι είτε η ημερομηνία έναρξης είτε η ημερομηνία λήξης μιας περιόδου δηλώσεων. Εφόσον ενεργοποιείται λοιπόν το Cron ανακτά απο την βάση δεδομένων όλες τις απαραίτητες εγγραφές που πρέπει συνδυαστικά να διεκπεραιωθούν απο την εφαρμογή για την λήψη μιας απόφασης. Με την ολοκλήρωση της ανάκτησης πληροφορίας από το Cron, οι εν λόγω εγγραφές προωθούνται σε έναν worker μηχανισμό, καθώς αποφασίστηκε αυστηρά ότι πρέπει να αποφευχθεί η απασχόληση του Main Thread της εφαρμογής από μία τόσο data-intensive διαδικασία καθώς απώτερος σκοπός είναι επίσης οι εν λόγω διαδικασία να εκτελεστεί παράλληλα στο παρασκήνιο της εφαρμογής. Εν τέλη με την ολοκλήρωση εκτέλεσης της λειτουργίας του worker λαμβάνεται η τελική απόφαση του αν η περίοδος πρέπει να ξεκινήσει ή να λήξει με βάση την κατα εκτέλεση ημερομηνία του μηχανήματος, αποφεύγοντας το Main Thread block και την μείωση ανταπόκρισης της εφαρμογής στο σύνολο της.

Ο συνδυασμός του Cron και των Workers παρέχει έναν ισχυρό μηχανισμό για τον προγραμματισμό εργασιών και την επεξεργασία στο παρασκήνιο σε εφαρμογές .NET Core 6.0. Το Cron επιτρέπει την αυτοματοποίηση επαναλαμβανόμενων εργασιών, ενώ οι workers χειρίζονται ασύγχρονα λειτουργίες με υψηλή ένταση πόρων, βελτιώνοντας τη συνολική απόδοση και την εμπειρία του χρήστη της εφαρμογής. Η κατανόηση και η αποτελεσματική αξιοποίηση αυτών των εννοιών είναι ζωτικής σημασίας για τη δημιουργία εύρωστων και αποδοτικών εφαρμογών.

4.2.11 Migrations / Μεταστάσεις στο EF Core

Τα Migrations αναφέρονται στη διαδικασία διαχείρισης και ενημέρωσης σχημάτων βάσεων δεδομένων στο Entity Framework Core. Κατά την ανάπτυξη εφαρμογών, το σχήμα της βάσης δεδομένων με την πάροδο του χρόνου μπορεί να εξελίσσεται λόγω μεταβαλλόμενων απαιτήσεων ή ενημερώσεων του μοντέλου δεδομένων.

Τα Migrations στο EF Core επιτρέπουν την πραγματοποίηση αυτών των αλλαγών στο σχήμα της βάσης δεδομένων διατηρώντας τα υπάρχοντα δεδομένα και εξασφαλίζοντας τη συμβατότητα του κώδικα της εφαρμογής με το ενημερωμένο σχήμα. Αυτοματοποιεί τη δημιουργία και την εκτέλεση σεναρίων SQL που εφαρμόζουν τις απαραίτητες τροποποιήσεις στη δομή του σχήματος.

Η διαδικασία του Migration στο EF Core περιλαμβάνει διάφορα βήματα. Αρχικά, ορίζονται οι επιθυμητές αλλαγές στο μοντέλο δεδομένων χρησιμοποιώντας Code-First Migration ή χρησιμοποιώντας μια προσέγγιση που βασίζεται σε στιγμιότυπα (Snapshot-Based Migrations). Τα Code-First migrations περιλαμβάνουν τη συγγραφή κώδικα C# που καθορίζει τις αλλαγές, ενώ τα Snapshot-Based Migrations αποτυπώνουν την τρέχουσα κατάσταση του μοντέλου δεδομένων και δημιουργούν migration με βάση τις διαφορές μεταξύ αυτών.

Μόλις καθοριστούν τα Migration, χρησιμοποιώντας τις αντίστοιχες εντολές που παρέχει το EF Core για την δημιουργία των ανάλογων αυτών αρχείων, παρέχονται τα απαραίτητα σενάρια SQL για την εφαρμογή των αλλαγών αυτών. Αυτά τα αρχεία μπορούν να εκδοθούν και να διαχειριστούν απο συστήματα ελέγχου πηγαίου κώδικα (IDEs).

Κατά την ανάπτυξη της εφαρμογής, τα αρχεία εκτελούνται στη στοχευμένη βάση δεδομένων, είτε χειροκίνητα είτε ως μέρος μιας αυτοματοποιημένης διαδικασίας. Το EF Core εντοπίζει τα Migrations και διασφαλίζει ότι εκτελούνται μόνο τα Migrations αυτά που είναι σε εκκρεμότητα. Αυτό βοηθά στη διατήρηση της ακεραιότητας της βάσης δεδομένων και αποφεύγει τυχόν συγκρούσεις ή απώλεια δεδομένων κατά τη διάρκεια αλλεπάλληλων ενημερώσεων του σχήματος.

Τα Migrations στο EF Core προσφέρουν και άλλα πλεονεκτήματα. Επιτρέπουν την αποτελεσματική συνεργασία μεταξύ ομάδων προγραμματιστών που εργάζονται στο ίδιο έργο, καθώς οι αλλαγές στο σχήμα της βάσης δεδομένων μπορούν εύκολα να παρακολουθούνται και να εφαρμόζονται στα τοπικά περιβάλλοντα ανάπτυξης του καθενός. Απλοποιεί επίσης τη διαδικασία ανάπτυξης με την αυτοματοποίηση των ενημερώσεων του σχήματος, μειώνοντας τις πιθανότητες σφαλμάτων ή ασυνεπειών από κάποιο μέλος της ομάδας ανάπτυξης.

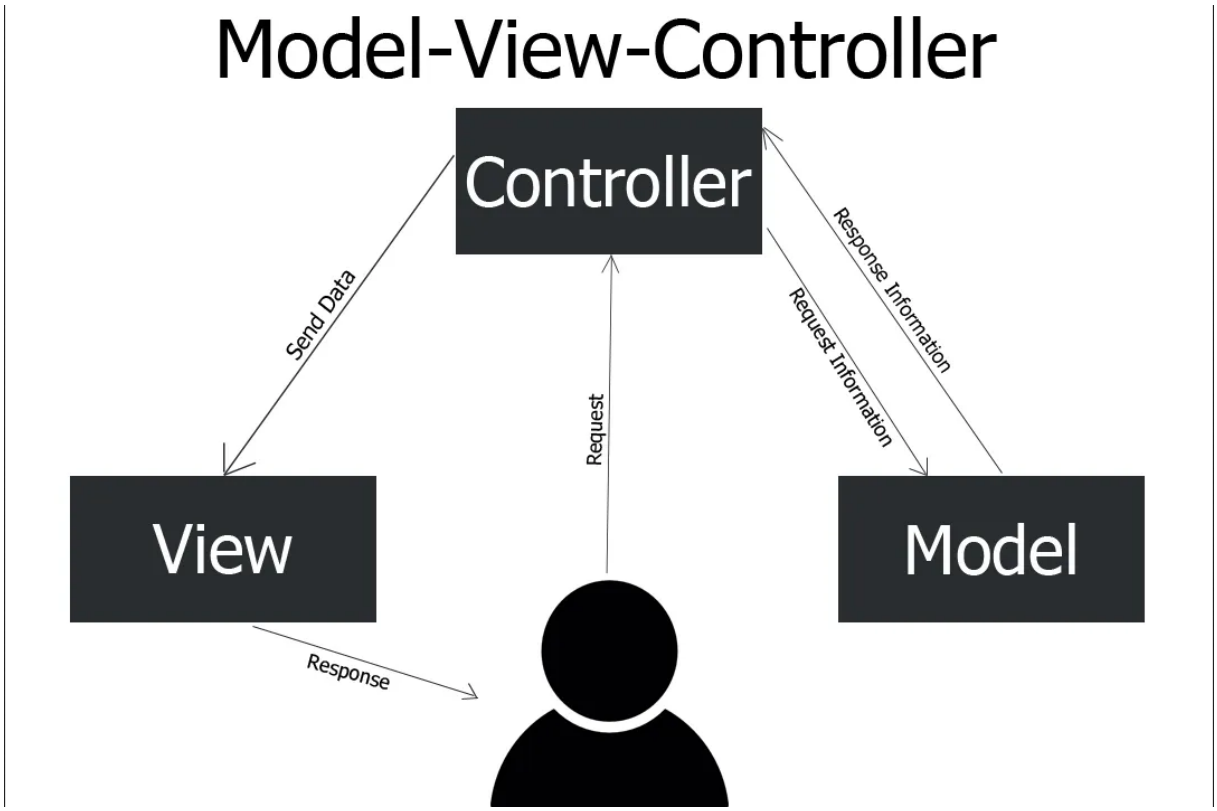
Ωστόσο, υπάρχουν προκλήσεις που σχετίζονται με τη χρήση τους. Ο χειρισμός σύνθετων σεναρίων, όπως η μετονομασία ή η διαγραφή οντοτήτων βάσης δεδομένων, απαιτεί προσεκτικό σχεδιασμό και μελέτη. Πρέπει να διασφαλιστεί ότι η διαδικασία μετάβασης δεν θα σπάσει την υπάρχουσα λειτουργικότητα ή δεν θα επηρεάσει αρνητικά τις επιδόσεις. Επιπλέον, η έκδοση και η διατήρηση της προς τα πίσω συμβατότητας του σχήματος της βάσης δεδομένων μπορεί να είναι πρόκληση όταν χρησιμοποιούνται πολλαπλές εκδόσεις της εφαρμογής.

Συνοψίζοντας, τα Migrations είναι ένα ισχυρό χαρακτηριστικό που απλοποιεί τη διαχείριση και την εξέλιξη των σχημάτων βάσεων δεδομένων σε εφαρμογές λογισμικού. Παρέχει μια βελτιωμένη προσέγγιση για την εφαρμογή αλλαγών σχήματος, διατηρώντας παράλληλα την ακεραιότητα των δεδομένων και εξασφαλίζοντας τη συμβατότητα με το code - base της εφαρμογής. Η κατανόηση και η αξιοποίηση των Migrations είναι απαραίτητη για αυτούς που εργάζονται με το περιβάλλον που παρέχει η .NET, καθώς διευκολύνει τις ελεγχόμενες ενημερώσεις του σχήματος της βάσης δεδομένων.

4.2.12 Γενικά για το MVC Design Pattern

Το αρχιτεκτονικό πρότυπο MVC (Model-View-Controller) είναι ένα ευρέως χρησιμοποιούμενο πρότυπο αρχιτεκτονικής σχεδίασης για τη δημιουργία εφαρμογών λογισμικού. Διαχωρίζει μια εφαρμογή σε τρία κύρια στοιχεία: το Μοντέλο (Model), την Προβολή (View) και τον Ελεγκτή (Controller). Κάθε συστατικό έχει συγκεκριμένο ρόλο και ευθύνη, με αποτέλεσμα τον σαφή διαχωρισμό των λειτουργιών που πρέπει το καθένα να εκτελεί.

- Model: Το Μοντέλο αναπαριστά τα δεδομένα και την επιχειρησιακή λογική μιας εφαρμογής. Ενθυλακώνει τις δομές δεδομένων, την κατάσταση και τους κανόνες της εφαρμογής. Είναι υπεύθυνο για το χειρισμό της αποθήκευσης, ανάκτησης και επεξεργασίας των δεδομένων. Το Μοντέλο είναι ανεξάρτητο από τη διεπαφή χρήστη (User Interface) και επικοινωνεί με την Προβολή και τον Ελεγκτή μέσω καλά καθορισμένων και αυστηρών διεπαφών.
- View: Η προβολή αντιπροσωπεύει τη διεπαφή χρήστη της εφαρμογής. Είναι υπεύθυνο για την απεικόνιση των δεδομένων στον χρήστη και την καταγραφή των αλληλεπιδράσεων του. Το View λαμβάνει τα δεδομένα ενός Model μέσω ενός Ελεγκτή και τα παρουσιάζει σε μορφή κατάλληλη και κατανοητή από τον χρήστη. Είναι συνήθως παθητικό και δεν περιέχει επιχειρησιακή λογική. Σε πολλές εφαρμογές, το επίπεδο προβολής είναι υπεύθυνο για τη δημιουργία των HTML αναπαραστάσεων μιας εφαρμογής.
- Controller: Ο Ελεγκτής ενεργεί ως ενδιάμεσος μεταξύ του Μοντέλου και της Προβολής. Χειρίζεται την είσοδο του χρήστη, επεξεργάζεται τα αιτήματα και ενορχηστρώνει τη ροή των δεδομένων μεταξύ του Μοντέλου και της Προβολής. Ο Ελεγκτής λαμβάνει ενέργειες του χρήστη από την Προβολή, καλεί τις κατάλληλες λειτουργίες στο Μοντέλο και ενημερώνει την Προβολή με τα τροποποιημένα δεδομένα. Περιέχει τη λογική της εφαρμογής για το χειρισμό των αλληλεπιδράσεων του χρήστη, τη λήψη αποφάσεων και το συντονισμό της συνολικής συμπεριφοράς του συστήματος.



Σχήμα 4.2: Το Αρχιτεκτονικό Πρότυπο MVC

Το πρότυπο MVC προωθεί το διαχωρισμό των εννοιολογικών ρόλων, καθιστώντας τις εφαρμογές πιο συντηρήσιμες και ευκολότερα ελεγχόμενες. Επιτρέπει την επαναχρησιμοποίηση των στοιχείων και υποστηρίζει την παράλληλη ανάπτυξη, επιτρέποντας σε διαφορετικές ομάδες να εργάζονται ανεξάρτητα σε διαφορετικά τμήματα μιας εφαρμογής. Επιπλέον, το MVC παρέχει ευελιξία για την προσαρμογή της διεπαφής του χρήστη ή της επιχειρησιακής λογικής χωρίς να επηρεάζονται άλλα στοιχεία που είναι εν γένη συνδεδεμένα μαζί της.

4.2.13 Ειδικά για το MVC Design Pattern σε Web API στην .NET Core 6.0

Όσον αφορά την ανάπτυξη της Backend εφαρμογή επιλέχθηκε το **MVC (Model View Controller)** ως αρχιτεκτονικό πρότυπο**.** Το συγκεκριμένο πρότυπο επιλέχθηκε, γιατί όπως προαναφέρθηκε επιτρέπει την παράλληλη ανάπτυξη του λογισμικού από μεγάλες ομάδες προγραμματιστών, και είναι δομημένο αρκετά αυστηρά ώστε να γίνεται διακριτοποίηση μεταξύ διαφόρων σημείων της εφαρμογής. Επίσης, λόγω της δομής του συγκεκριμένου προτύπου διευκολύνεται η δυνατότητα επαναχρησιμοποίηση ήδη υπάρχοντος κώδικα καθώς, χωρίζεται σε μικρότερες ξεχωριστές εννοιολογικές υπηρεσίες.

Το πρότυπο MVC είναι ένα πρότυπο που χρησιμοποιείται ευρέως σε διαδικτυακές εφαρμογές και για αυτό το λόγο βρήκε χώρο υλοποίησης και στην παρούσα Backend εφαρμογή που με πιο τεχνικούς όρους συγκαταλέγεται κάτω από την κατηγορία των Web APIs. Αν και εξ ορισμού τα Web APIs δεν έχουν αυτοτελή Views όπως προστάζει το MVC Pattern δεν απαγορεύεται η χρήση αυτών. Στην εν λόγω εφαρμογή το View κομμάτι εξυπηρετείται πλήρως απο ξεχωριστή και αυτόνομη FrontEnd εφαρμογή. Παρακάτω θα αναφερθούν με ορισμούς το πως υλοποιούνται τα 2 επόμενα κομμάτια που προστάζει το σχεδιαστικό Πρότυπο, δηλαδή τα Models και τα Controllers.

4.2.14 Controllers

Στα Web APIs της .NET Core, οι ελεγκτές είναι υπεύθυνοι για τη λήψη αιτήσεων HTTP και τον καθορισμό της κατάλληλης ενέργειας που πρέπει να εκτελεστεί. Ενεργούν ως μεσάζοντες μεταξύ του πελάτη ,των επιπέδων λογικής και δεδομένων της εφαρμογής. Όταν ένας πελάτης στέλνει ένα αίτημα σε ένα συγκεκριμένο τελικό σημείο ή αλλιώς endpoint, υπάρχει ένας αντίστοιχος ελεγκτής που είναι υπεύθυνος για την επεξεργασία και την απάντηση στο αίτημα αυτό.

Οι ελεγκτές στη .NET Core είναι υπεύθυνοι για την ερμηνεία και την επικύρωση της εισόδου, την κλήση της απαραίτητης επιχειρησιακής λογικής και την κατασκευής μιας κατάλληλης απάντησης. Οι ελεγκτές επίσης ενθυλακώνουν τη συμπεριφορά και τις ενέργειες που μπορεί να εκτελέσει η εφαρμογή.

Οι API Controllers ειδικότερα, που χρησιμοποιούνται εκτενώς στην εφαρμογή, ορίζονται ως κλάσεις που κληρονομούν από μια βασική κλάση ControllerBase που παρέχεται από το Framework. Κάθε κλάση Controller περιέχει μεθόδους δράσης, οι οποίες είναι δημόσιες μέθοδοι που αντιστοιχούν σε συγκεκριμένες ενέργειες που μπορεί να εκτελέσει η εφαρμογή εάν αυτές ζητηθούν φυσικά. Αυτές οι μέθοδοι δράσης σχολιάζονται με χαρακτηριστικά (Attributes) για να καθορίζουν τις πιο γνωστές HTTP Μεθόδους όπως GET, POST, PUT, DELETE στις οποίο και ανταποκρίνονται.

Οπότε όταν λαμβάνεται ένα αίτημα, το σύστημα δρομολόγησης κατευθύνει την αίτηση στον κατάλληλο ελεγκτή με βάση το URI και την HTTP μέθοδο που χρησιμοποιήθηκε. Στη συνέχεια, καλείται η αντίστοιχη μέθοδος δράσης του ελεγκτή για να χειριστεί το αίτημα. Εντός της μεθόδου δράσης, ο Ελεγκτής έχει πρόσβαση στα δεδομένα της αίτησης, μπορεί να εκτελέσει οποιαδήποτε απαραίτητη επεξεργασία και να επιστρέψει στο τέλος την κατάλληλη απόκριση, όπως η απόδοση μιας προβολής ή η επιστροφή δεδομένων JSON σε περίπτωση ενός RESTful Web API. Οι ελεγκτές παρέχουν επίσης πρόσθετα χαρακτηριστικά και δυνατότητες, όπως δέσμευση μοντέλου (data-binding), επικύρωση εισόδου (input validation) και ελέγχους εξουσιοδότησης (user authorization). Η δέσμευση μοντέλου αντιστοιχίζει αυτόματα τα εισερχόμενα δεδομένα αίτησης σε παραμέτρους της μεθόδου διαχείρισης ή αντικείμενα μοντέλου, απλοποιώντας την εξαγωγή και τη χρήση των δεδομένων του πελάτη. Η επικύρωση εισόδου διασφαλίζει ότι τα δεδομένα που λαμβάνονται είναι έγκυρα και πληρούν τα προκαθορισμένα κριτήρια που βάση αυτών χτίστηκε το εκάστοτε μοντέλο. Οι έλεγχοι εξουσιοδότησης επαληθεύουν τα δικαιώματα του πελάτη πριν επιτρέψουν την πρόσβαση σε ορισμένες ενέργειες ή πόρους ενδότερα της εφαρμογής.

Συνοπτικά, οι Controllers είναι βασικά συστατικά των εφαρμογών ιστού που χειρίζονται τις εισερχόμενες αιτήσεις και συντονίζουν τη συμπεριφορά της εφαρμογής. Ερμηνεύουν και επικυρώνουν την είσοδο, επικαλούνται την απαραίτητη επιχειρησιακή λογική και παράγουν την κατάλληλη απόκριση. Χρησιμοποιώντας αποτελεσματικά τους Controllers, υπάρχει η δυνατότητα δημιουργίας ισχυρών και κλιμακούμενων εφαρμογών ιστού. Πολύ συχνά σε αυτό το επίπεδο θα συναντάται και η χρήση των Services όπως έχουν πλήρως αναλυθεί προηγουμένως, τα οποία εξυπηρετούν μεμονωμένα ξεχωριστές δομές υπηρεσιών και επιχειρησιακής λογικής.

```
[Route("api/[controller]")]
[ApiController]
3 references
public class AuthClientController : ControllerBase
{
    private readonly ILogger<AuthClientController> _logger;
    private readonly ClientUserAuthService _clientAuthService;

    0 references
    public AuthClientController(
        ILogger<AuthClientController> logger,
        ClientUserAuthService clientAuthService)
    {
        _logger = logger;
        _clientAuthService = clientAuthService;
    }

    [HttpPost("sign-in")]
    0 references
    public async Task<APIResult<string>> SignInClient([FromBody] BaseUserRequest clientLogin)
    {
        IMPLEMENTATION
    }
```

Σχήμα 4.3: Ελεγκτής AuthClient

Παρακάτω θα αναλυθούν επεξηγηματικά οι πιο κρίσιμες γραμμές κώδικα του σχήματος 3.3.

- **Χαρακτηριστικό [Route("api/[controller]")]**: Το attribute (χαρακτηριστικό) που υποδηλώνει ότι η παρακάτω κλάση εξυπηρετείται κάτω από το URI που αναγκαστικά θα περιέχει το τμήμα **api/[controller]**. Το σημείο **[controller]** είναι ένα wildcard που συμπληρώνεται αυτόματα από το Framework κατα την εκτέλεση του προγράμματος βάση της υποκείμενης ονομασίας Κλάσης του Ελεγκτή.
- **Χαρακτηριστικό [ApiController]:** Το attribute (χαρακτηριστικό) αυτό υποδηλώνει ότι η παρακάτω κλάση είναι ένας Ελεγκτής και ειδικότερα ένας API Ελεγκτής που εξ ορισμού απο το Framework επιστρέφει απαντήσεις υπό τη μορφή application/json.
- **Κλάση public class AuthClientController : ControllerBase:** Η Ονομασία του Controller που γίνεται διαθέσιμος και μπορεί να κληθεί απο ένα Http Request. Σε αυτό το σημείο επίσης συμπληρώνεται το [controller] wildcard που αναφέρθηκε προηγουμένως. Η δυναμική τιμή που θα πάρει το wildcard είναι η AuthClient ή αλλιώς η ονομασία της κλάσης αν αφαιρεθεί η κατάληξη Controller που η ύπαρξη της είναι αναγκαστική απο το Framework για την σωστή λειτουργία του.
- **Χαρακτηριστικό [HttpPost("sign-in")]:** Το attribute (χαρακτηριστικό) που υποδηλώνει ότι η παρακάτω μέθοδος είναι ένα web endpoint που ανήκει και εξυπηρετείται απο τον ελεγκτή μέσο μόνο της Http μεθόδου POST. Το αλφαριθμητικό που δηλώνεται παραμετρικά είναι το τελικό καταληκτικό σημείο του URI που αν κληθεί ως αίτημα θα εξυπηρετηθεί από την παρακάτω μέθοδο διαχείρισης αιτήματος.
- **Μέθοδος Διαχείρισης SignInClient([FromBody] BaseUserRequest clientLoginReq):** Η μέθοδος διαχείρισης που εξυπηρετεί το συγκεκριμένο endpoint και έχει εσωτερικό όνομα SignInClient. Δέχεται παραμετρικά ως payload ένα BaseUserRequest Model καθώς καλείται μέσω μεθόδου POST και υπάρχει η δυνατότητα επισύναψης δεδομένων στο Body Section του HTTP Request. Το ακριβώς παραπάνω υποδηλώνει το χαρακτηριστικό **[FromBody]**. Δηλαδή το **[FromBody]** attribute ενημερώνει τον Controller ότι το παρών μοντέλο θα πρέπει να συμπληρωθεί από τα δεδομένα που αποστέλλονται από τον χρήστη και φιλοξενούνται μέσα στο HTTP Body Section.

Συμπερασματικά λοιπόν, αν θέλαμε να πραγματοποιήσουμε κλήση στο συγκεκριμένο endpoint

δεδομένου ενός domain name που λόγου χάρης παραδείγματος είναι **https://iee.ihu.gr** ή τελική μορφή του URI Endpoint θα ήταν η εξής:

POST **https://iee.ihu.gr/api/AuthClient/sign-in** και επιπλέον θα έπρεπε να επισυνάψουμε ειδικές πληροφορίες ως payload στο Http Body ώστε η απάντηση που θα λάβουμε απο τον Controller να είναι θετική.

4.2.15 Models

Σε μια εφαρμογή Web API που δομείται βάση του ASP.NET Core 6.0 Web API και ακολουθεί το πρότυπο MVC, τα Μοντέλα αντιπροσωπεύουν τις δομές δεδομένων που χρησιμοποιεί η εφαρμογή και την επιχειρησιακή λογική της. Τα Μοντέλα ενθυλακώνουν τα δεδομένα προς χρήση, καθορίζουν τη δομή και παρέχουν μεθόδους για τον χειρισμό και την πρόσβαση σε αυτά.

Επίσης αντιπροσωπεύουν συνήθως τις σχεσιακές οντότητες ή τους πόρους με τους οποίους ασχολείται το API. Οι οντότητες αυτές είναι που στην τελική αντιστοιχίζονται σε πίνακες βάσεων δεδομένων. Τα Μοντέλα επιπροσθέτως καθορίζουν τις ιδιότητες και τις σχέσεις των οντοτήτων, καθώς και τυχόν κανόνες επικύρωσης ή επιχειρησιακή λογική που σχετίζονται με αυτές. Το EF Core κάνει αποκλειστική χρήση Μοντέλων και τα Migration ,που έχουν προαναφερθεί, χτίζονται με βάση αυτά.

BaseUserRequestModel: Παρακάτω παρατίθεται ένα παράδειγμα ενός μοντέλου εν ονόματι BaseUserRequest που πρωτίστως παρουσιάστηκε στην υπό ενότητα των ελεγκτών. Σε ένα τέτοιο Model το .NET Core 6.0 προσπαθεί να “δέσει” (bind) την πληροφορία που λαμβάνει από το HTTP Post Body. Αν οι πληροφορίες που απαιτούνται απο το Model στέλνονται με το ορθό πρότυπο και με τους σωστούς τύπους δεδομένων που αναμένονται από τον αρμόδιο μηχανισμό του Framework, τότε η αντικατόπτριση των δεδομένων γίνεται με επιτυχία και οι τιμές είναι διαθέσιμες προς χρήση σε παρακάτω κομμάτια κώδικα. Ένα τέτοιο Model σε κώδικα αναπαρίσταται ως την παρακάτω εικόνα:

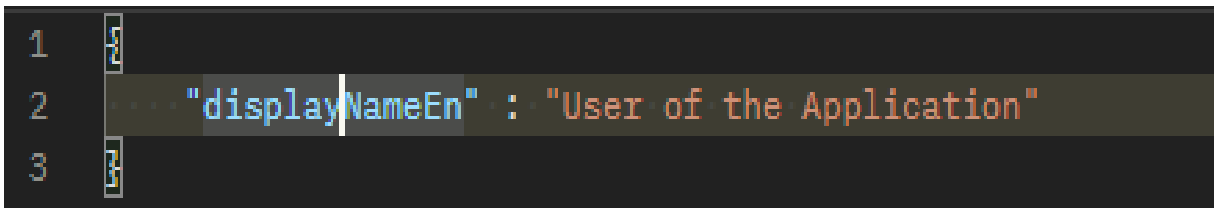
```
public class BaseUserRequest
{
    [JsonProperty("id")]
    3 references
    public string FourDigitId { get; set; } // Κωδικός Μοναδικής Σημασίας
    [JsonProperty("displayNameEn")]
    1 reference
    public string DisplayNameEn { get; set; } // Ονοματεπώνυμο στα Αγγλικά
    [JsonProperty("displayNameEl")]
    1 reference
    public string DisplayNameEl { get; set; } // Ονοματεπώνυμο στα Ελληνικά
    [JsonProperty("eduPersonAffiliation")]
    1 reference
    public PersonAffiliation EduPersonAffiliation { get; set; }  // Enumerator ρόλου Χρήστη
    [JsonProperty("titleEn")]
    1 reference
    public string TitleEn { get; set; }  // Τίτλος στα Αγγλικά
    [JsonProperty("titleEl")]
    1 reference
    public string TitleEl { get; set; } // Τίτλος στα Αγγλικά
    [JsonProperty("am")]
    1 reference
    public string? Am { get; set; } = null;  // Αριθμός Μητρώου Φοιτητή (Προαιρετικό)
    [JsonProperty("userId")]
    1 reference
    public string? UserId { get; set; } = null; // Εξωτερική Ταυτότητα Χρήστη (Προαιρετικό)
    [JsonProperty("mail")]
    1 reference
    public string? Mail { get; set; } = null; // Διεύθυνση Ηλ.Ταχ. Χρήστη (Προαιρετικό)
    [JsonProperty("regsem")]
    1 reference
    public string? RegSem { get; set; } = null;  // Εξάμηνο εγγραφής Φοιτητή (Προαιρετικό)
    [JsonProperty("regyear")]
    1 reference
    public string? RegYear { get; set; } = null;  // Έτος εισαγωγής Φοιτητή (Προαιρετικό)
    [JsonProperty("semester")]
    1 reference
    public string? Semester { get; set; } = null;  // Τρέχων εξάμηνο Φοιτητή (Προαιρετικό)
    [JsonProperty("personalTitle")]
    1 reference
    public string? PersonalTitle { get; set; } = null; // Προσωπικός Τίτλος (Προαιρετικό)

    [JsonProperty("eduPersonalEntitlementEn")]
    1 reference
    public string? EduPersonalEntitlementEn { get; set; } = null; // Ακαδημαϊκές Διακρίσεις στα Αγγλικά (Προαιρετικό)

    [JsonProperty("eduPersonalEntitlementEl")]
    1 reference
    public string? EduPersonalEntitlementEl { get; set; } = null; // Ακαδημαϊκές Διακρίσεις στα Ελληνικά (Προαιρετικό)
}
```

Σχήμα 4.4: Μοντέλο BaseUserRequest

Επίσης προηγουμένως αναφέρθηκε ότι τα Model είναι ουσιαστικά η δομή των πληροφοριών που εν τέλη αποθηκεύονται ως εγγραφή σε μία βάση δεδομένων. Κάτι που πρέπει να γίνει αντιληπτό είναι ότι όλα τα Model μιας εφαρμογής δεν είναι αναγκαστικό να αντικατοπτρίζονται στο σχήμα και να αποθηκεύονται μόνιμα στην βάση. Έτσι ,το παραπάνω Model δεν είναι αυτό που τελικά θα αποθηκευτεί. Ανταυτού στην συγκεκριμένη εφαρμογή και μετά από ανάλογες τροποποιήσεις το Model BaseUserRequest αντικατοπτρίζεται σε ένα Νέο Μοντέλο με όνομα BaseUser που τελικά θα αποθηκευτεί μόνιμα στην βάση δεδομένων και θα αντικατοπτρίζει ένα έγγραφο Χρήστη της εφαρμογής.

Χαρακτηριστικό [JsonProperty("...")]: Το χαρακτηριστικό αυτό αποτελεί ένα υποβοήθημα στον Binding Μηχανισμό Μοντέλων της .NET Core 6.0. Η κύρια χρήση του υποδηλώνει ότι με την δήλωση ενός διαφορετικού παραμετρικού αλφαριθμητικού που "δένεται" με κάθε ξεχωριστό πεδίο της C# κλάσης ο μηχανισμός έχει καλύτερες πιθανότητες να βρει και να δέσει ένα πεδίο που ως έχει αυστηρά

ως τίτλο ένα αλφαριθμητικό που η τιμή του ισούται με το αλφαριθμητικό που δίνεται παραμετρικά στο χαρακτηριστικό. Προς περαιτέρω κατανόηση αν υποθετικά στέλνοντας ένα JSON Object ως POST Request payload με την παρακάτω μορφή:



Σχήμα 4.5: Σωστή Μορφή Αναμενόμενης Τιμής σε Μορφή JSON

Ο μηχανισμός θα επιτύχει στο να δέσει το JSON πεδίο “displayNameEn” με το C# πεδίο DisplayNameEn καθώς το όρισμα που δίνεται στο **[JsonProperty]** χαρακτηριστικό ισούται αλφαριθμητικά με κάποιο κλειδί του αντικειμένου που αποστέλλεται.

Σε περίπτωση όμως που το JSON Object ακολουθούσε την παρακάτω μορφή:

Σχήμα 4.6: Λάθος Μορφή Αναμενόμενης Τιμής σε Μορφή JSON

Ο μηχανισμός θα αποτύχει να δέσει την τιμή καθώς αναμένει αυστηρά το JSON πεδίο να έχει όνομα που ισούται με “displayNameEn”. Σε περίπτωση αποτυχίας του μηχανισμού δεν εγείρεται κάποιο κρίσιμο σφάλμα που διακόπτει την λειτουργία της εφαρμογής, αλλά η τιμή του πεδίου δείχνει στην περιοχή μνήμης που ζει το Null Reference σε κάθε γλώσσα προγραμματισμού, και είναι χρέος του προγραμματιστή να διαχειριστή ορθά τέτοιες περιπτώσεις αποτυχίας του μηχανισμού ώστε να μην συμβεί κάποιο μοιραίο σφάλμα κατα την διάρκεια εκτέλεσης του προγράμματος.

4.2.16 Data Transfer Objects (DTO), APIResult<T>

Το APIResult αντικείμενο που αναφέρεται σε προηγούμενη ενότητα αναφέρεται σε ένα κοινό πρότυπο ή προσέγγιση που χρησιμοποιείται για την τυποποίηση της μορφής αποκρίσεων των τελικών API Endpoints σε εφαρμογές ιστού. Είναι ένας τρόπος για την ενθυλάκωση του αποτελέσματος μιας λειτουργίας του API και την παροχή συνεπών και δομημένων απαντήσεων στους χρήστες της εφαρμογής. Το APIResult όπως και οποιαδήποτε άλλα Model που ακολουθεί την ίδια φιλοσοφία αναφέρεται ως Data Transfer Objects στο περιβάλλον της .NET

Ως ορισμός που θα μπορούσε να δοθεί για τα DTOs είναι ο εξής:

Ως DTO ορίζεται ένα Wrapper Model που πρωταρχική του λειτουργία είναι η αποστολή και παραλαβή δεδομένων τόσο εκτός της εφαρμογή όσο και εντός αυτής από διαφορετικές υπό ενότητες της. Για παράδειγμα ένα ξεχωριστό DTO μπορεί να χρησιμοποιεί για το σερβίρισμα των HTTP αποκρίσεων στον χρήστη έξω απο την εφαρμογή και ένα διαφορετικό για τον σερβίρισμα δεδομένων που

προέρχονται από ένα συγκεκριμένο τμήμα της εφαρμογής και στην συνέχεια τροφοδοτούνται ως όρισμα σε ένα άλλο διαφορετικό τμήμα της.

Μία καλή τεχνική που ακολουθείται είναι η κατασκευή ενός ξεχωριστού DTO που κατασκευάζεται μία φορά και χρησιμοποιείται ως επιστρεφόμενο Model μετά το πέρας εκτέλεσης μιας μεθόδου ενός Service ή παραμετρικό Model εισαγωγής σε άλλες μεθόδους που μπορεί να υλοποιούνται σε ξεχωριστά Services. Με τον παραπάνω τρόπο αποσαφηνίζεται ότι θα υπάρχει ένα κοινό εσωτερικό πρωτόκολλο επικοινωνίας με έναν κοινό παρονομαστή που θα πρέπει αναγκαστικά να τηρείται από όλες τις υπό ενότητες της εφαρμογής.

Το ίδιο ακριβώς σκεπτικό αναγάγετέ και στην παράδοση του HTTP Response μέσω κάποιου DTO στον χρήστη που υποβάλει το HTTP Request. Τα DTOs απαιτούνται να χτίζονται με τέτοιο τρόπο απο τους προγραμματιστές ώστε να καλύπτουν πολλαπλά σενάρια αποτελέσματος. Αναλύοντας περαιτέρω, ένα HTTP Response μπορεί να είναι επισημασμένο ως επιτυχές είτε ως αποτυχημένο. Έτσι πρέπει το DTO που το εξυπηρετεί, να χτιστεί με τέτοια υποδομή ώστε να μπορεί να στεγάσει πολλαπλές απόρροιες ενός HTTP Reponse ακολουθώντας τον κοινό παρονομαστή που αναφέρθηκε.

Το APIResult DTO συγκεκριμένα , αλλά και γενικώς κάθε DTO, υλοποιείται συνήθως ως μια Generic Class ή ένα προσαρμοσμένο αντικείμενο που περιέχει σχετικές πληροφορίες σχετικά με το αποτέλεσμα της λειτουργίας του API. Περιλαμβάνει ιδιότητες όπως ο κωδικός κατάστασης (Http Status Code), το μήνυμα (StatusMessage) και το payload (Data) των δεδομένων που είναι μορφής τύπου T δηλαδή παραμετροποιήσιμος αναλόγως της περιστάσεως.

Ο κύριος σκοπός του APIResult είναι να παρέχει μια ενιαία μορφή απόκρισης που μπορεί να μεταφέρει την επιτυχία, την αποτυχία ή οποιοδήποτε άλλο συγκεκριμένο αποτέλεσμα μιας κλήσης API (π.χ. Ανακατεύθυνση σε άλλο πόρο του συστήματος). Επιτρέπει στους προγραμματιστές να κοινοποιούν την κατάσταση και τις λεπτομέρειες της λειτουργίας στον πελάτη με συνεπή τρόπο, ανεξάρτητα από το συγκεκριμένο τελικό endpoint που καλείται.

Αναλύοντας περαιτέρω, η κλάση APIResult<T> περιλαμβάνει συνήθως τις ακόλουθες ιδιότητες , οι οποίες δεν αποκλείεται να διαφέρουν απο υλοποίηση σε υλοποίηση. Σε γενικές γραμμές οι πολλαπλές εν γένη υλοποιήσεις θα πρέπει να παρέχουν πεδία όπως είναι τα παρακάτω:

- StatusCode: Αυτή η ιδιότητα αντιπροσωπεύει τον κωδικό κατάστασης HTTP της απόκρισης. Δείχνει αν η κλήση API ήταν επιτυχής (π.χ. 200 για “OK”) ή αν συνάντησε σφάλμα (π.χ. 400 για Bad Request, 500 για Internal Server Error κ.α.).
- StatusMessage: Η ιδιότητα StatusMessage παρέχει ένα περιγραφικό μήνυμα σχετικά με το αποτέλεσμα της λειτουργίας API. Μπορεί να περιλαμβάνει πρόσθετες πληροφορίες ή λεπτομέρειες σφάλματος για να βοηθήσει τους χρήστες να κατανοήσουν το αποτέλεσμα.
- Data: Η ιδιότητα Data περιέχει το API Response Payload, το οποίο μπορεί να διαφέρει απο endpoint σε endpoint. Μπορεί να περιέχει όλα τα σχετικά δεδομένα ή αντικείμενα που επιστρέφονται από την δεδομένη κλήση στο API.
- Με τη χρήση του APIResult, μπορούν να διασφαλιστούν συνεκτικές δομές απόκρισης σε διαφορετικά endpoints της εφαρμογής. Αυτό διευκολύνει τους πελάτες που χρησιμοποιούν το API να κατανοούν και να χειρίζονται τις απαντήσεις με συνέπεια.

Το APIResult μπορεί να είναι ιδιαίτερα επωφελές σε σενάρια επίσης όπου οι πελάτες πρέπει να επεξεργάζονται και να ερμηνεύουν την απόκριση προγραμματιστικά. Με την τήρηση μιας τυποποιημένης μορφής απόκρισης, οι χρήστες-προγραμματιστές μπορούν να βασίζονται στη δομή και τις ιδιότητες του APIResult για τον συνεπή χειρισμό διαφορετικών σεναρίων και σφαλμάτων.

Επιπλέον, το APIResult μπορεί να επεκταθεί ή να προσαρμοστεί ανάλογα με την υλοποίηση ώστε να περιλαμβάνει πρόσθετες ιδιότητες ή metadata σύμφωνα με τις ειδικές απαιτήσεις της εφαρμογής. Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να ανταποκρίνονται σε διάφορες περιπτώσεις χρήσης, διατηρώντας παράλληλα μια συνεπή μορφή απόκρισης.

Συνοπτικά, το APIResult είναι ένα DTO και ακολουθεί μια τυποποιημένη προσέγγιση για την ενθυλάκωση του αποτελέσματος των λειτουργιών του API. Παρέχει μια ενοποιημένη μορφή απόκρισης που περιλαμβάνει ιδιότητες όπως ο κωδικός κατάστασης, το μήνυμα και το payload δεδομένων, εξασφαλίζοντας συνέπεια και ευκολία ερμηνείας για πελάτες όσο και για προγραμματιστές που καταναλώνουν το API

4.2.17 Χαρακτηριστικά Επικύρωσης – Validation Attributes

Τα Validation Attributes κατέχουν σημαντικό ρόλο στην επικύρωση δεδομένων ενός μοντέλου και το χειρισμό διαφόρων σφαλμάτων που μπορούν να συμβούν σε μία εφαρμογή. Παρέχουν μηχανισμούς που διασφαλίζουν ότι τα δεδομένα που υποβάλλονται προς επεξεργασία συμμορφώνονται με καθορισμένους κανόνες και περιορισμούς, ενισχύοντας την αξιοπιστία και την ακεραιότητα της εφαρμογής.

Τα Validation Attributes έχουν την μορφή annotation στην C# που μπορούν να εφαρμοστούν σε πεδία ενός μοντέλου και όχι μόνο, για τον ορισμό κανόνων επικύρωσης. Αυτά τα χαρακτηριστικά αποτελούν μέρος του namespace System.ComponentModel.DataAnnotations και προσφέρουν ένα ευρύ φάσμα επιλογών επικύρωσης. Παραδείγματα συχνά χρησιμοποιούμενων χαρακτηριστικών επικύρωσης στην .NET περιλαμβάνουν:

- RequiredAttribute: όπου καθορίζει ότι μια ιδιότητα πρέπει να έχει μια τιμή και δεν μπορεί να είναι μηδενική ή null.
- StringLengthAttribute: όπου καθορίζει τους περιορισμούς μέγιστου και ελάχιστου μήκους για μια ιδιότητα τύπου συμβολοσειράς.
- RangeAttribute: όπου επικυρώνει ότι μια αριθμητική ιδιότητα εμπίπτει σε ένα καθορισμένο εύρος ψηφίων.
- RegularExpressionAttribute: όπου ελέγχει αν μια ιδιότητα συμβολοσειράς ταιριάζει με ένα καθορισμένο μοτίβο κανονικής έκφρασης (regular expression).

Με αυτά τα χαρακτηριστικά επικύρωσης, διευκολύνεται ο ορισμός και η επιβολή κανόνων επικύρωσης δεδομένων. Κατά τη διάρκεια του αντικατοπτρισμού δεδομένων ή κατά τη ρητή εκτέλεση ελέγχων επικύρωσης, το Framework αξιολογεί αυτόματα τα χαρακτηριστικά και επικυρώνει τα δεδομένα ανάλογα.

Ας σημειωθεί ότι τα Validation Attributes δεν βρίσκουν χρήση μόνο μέσα σε Model, καθώς μπορούν να δηλωθούν και να προστατέψουν συγκεκριμένα κομμάτια κώδικα που έχουν οριστεί ως μέθοδοι ή ακόμη και Controllers. Τα χαρακτηριστικά που αναφέρθηκαν δεν είναι τίποτα άλλο παρά Validation Attributes ειδικού σκοπού που χτίστηκαν απο την .NET Development Team και εξυπηρετούν ένα συγκεκριμένο σκοπό. Επίσης το Framework επιτρέπει την δημιουργία προσαρμοσμένων τέτοιων attributes που εξυπηρετεί έναν συγκεκριμένο σκοπό που θα δηλώσει ο δημιουργός του εκάστοτε attribute.

Συνοψίζοντας λοιπόν, τα Validation Attributes προσφέρουν έναν ισχυρό και βολικό τρόπο για την υλοποίηση της επικύρωσης δεδομένων και του χειρισμού σφαλμάτων, τόσο σε μοντέλα όσο και σε μεθόδους. Τα attributes αυτά επιτρέπουν σε προγραμματιστές να ορίζουν κανόνες επικύρωσης χρησιμοποιώντας annotations για την εκτέλεση ελέγχων επικύρωσης και χειρισμό σφαλμάτων.

Χρησιμοποιώντας αυτά τα χαρακτηριστικά, μπορεί να διασφαλιστεί η ακεραιότητα των δεδομένων και να βελτιωθεί η συνολική ποιότητα των εφαρμογών.

4.2.18 Βοηθητικές Κλάσεις / Μέθοδοι

Οι Helpers είναι βοηθητικές κλάσεις και μέθοδοι που παρέχουν επαναχρησιμοποιήσιμη λειτουργικότητα για να βοηθήσουν στην εκτέλεση κοινών εργασιών κατά την ανάπτυξη της εφαρμογής. Έχουν σχεδιαστεί για να απλοποιούν ή να ενθυλακώνουν πολύπλοκες ή επαναλαμβανόμενες λειτουργίες, διευκολύνοντας τη συγγραφή καθαρού, αποτελεσματικού και συντηρήσιμου κώδικα.

Οι Helpers μπορούν να λάβουν υπόσταση ως στατικές κλάσεις με στατικές μεθόδους ή ως μη στατικές κλάσεις με μεθόδους τύπου Instance. Μπορούν να περιλαμβάνουν ένα ευρύ φάσμα λειτουργιών όπου μερικά στιγμιότυπα από αυτές είναι:

- Ο Χειρισμός συμβολοσειρών: για κοινές λειτουργίες συμβολοσειρών, όπως μορφοποίηση, ανάλυση, περικοπή ή συνένωση.
- Ο Χειρισμός ημερομηνίας και ώρας: για την επεξεργασία ημερομηνιών και ωρών, όπως ανάλυση, μορφοποίηση, υπολογισμό διαφορών ή μετατροπή μεταξύ διαφορετικών ζωνών ώρας.
- Για λειτουργίες αρχείων και καταλόγων: Σε εργασίες που σχετίζονται με τη διαχείριση αρχείων και καταλόγων, όπως η ανάγνωση και εγγραφή αρχείων, η δημιουργία καταλόγων ή ο έλεγχος δικαιωμάτων σε αρχεία.
- Η Καταγραφή και χειρισμός σφαλμάτων: για την καταγραφή μηνυμάτων ή εξαιρέσεων, διευκολύνοντας την παρακολούθηση και την αντιμετώπιση προβλημάτων κατά τη διάρκεια της εκτέλεσης.
- Το Serialization / Deserialization: για τη μετατροπή αντικειμένων σε και από διαφορετικές μορφές δεδομένων, όπως για παράδειγμα JSON και XML, απλοποιώντας την ανταλλαγή δεδομένων.
- Η Κρυπτογράφηση και Hashing Digest: για την κρυπτογράφηση και αποκρυπτογράφηση δεδομένων, καθώς και για τη δημιουργία ασφαλών κωδικών hash για κωδικούς πρόσβασης ή άλλες ευαίσθητες πληροφορίες.
- Ο Χειρισμός αιτήσεων και αποκρίσεων HTTP: για την ανάλυση παραμέτρων ερωτήματος, το JSON Serialization / Deserialization ή ο καθορισμός επικεφαλίδων απάντησης (HTTP Request/Response Headers).

Με την ενθυλάκωση αυτών των κοινών λειτουργιών σε βοηθητικές κλάσεις ή συναρτήσεις, μπορεί να μειωθεί η εκτεταμένη επανάληψη κώδικα, να βελτιωθεί η αναγνωσιμότητα του καθώς και να βελτιωθεί η συντηρησιμότητα και η δυνατότητα ελέγχου των εφαρμογών τους. Οι Helpers προωθούν την επαναχρησιμοποίηση και επιτρέπουν στους προγραμματιστές να επικεντρωθούν στην υλοποίηση της επιχειρηματικής λογικής αντί να ασχολούνται με επαναλαμβανόμενες ή χαμηλού επιπέδου εργασίες.

Τέλος, αξίζει να σημειωθεί ότι ενώ οι βοηθοί μπορεί να είναι εξαιρετικά χρήσιμοι, είναι σημαντικό να χρησιμοποιούνται με σύνεση και να αποφεύγεται η δημιουργία υπερβολικά πολύπλοκων ή tightly coupled βοηθητικών κλάσεων. Η σωστή οργάνωση και τεκμηρίωση των Helpers είναι η χρυσή τομή που πρέπει να βρεθεί ώστε για να διασφαλιστεί η αποτελεσματική χρήση τους σε όλο το εύρος της εφαρμογής

4.3 Επίλογος

Σε αυτό το κεφάλαιο παρουσιάστηκαν αναλυτικά οι τεχνολογίες, οι τεχνικές και μερικές προσαρμοσμένες υλοποιήσεις που εφαρμόστηκαν για την ανάπτυξη της ξεχωριστής εφαρμογής του

BackEnd. Αν και το κεφάλαιο καταλαμβάνει μεγάλη έκταση στο κείμενο, καλύφθηκε ένα πολύ μικρό ποσοστό των τεχνικών που ακολουθήθηκαν ώστε η εφαρμογή να καλύψει πλήρως τις απαιτήσεις που προτάθηκαν. Μολονότι ισχύει το παραπάνω, αφετέρου ένα κείμενο δεν είναι εφικτό να καλύψει πλήρως το ευρύ φάσμα τακτικών που μπορούν να χρησιμοποιηθούν γενικότερα για την υλοποίηση μιας ολόκληρης εφαρμογής καθώς και την γνώση που χρειάστηκε να αποκτηθεί ώστε να έρθουν εις πέρας οι απαιτήσεις. Τελικός σκοπός μετά την ανάγνωση του κεφαλαίου από έναν αναγνώστη δεν είναι η εξ ολοκλήρου κατανόηση ενός ολόκληρου περιβάλλοντος με ότι αυτό παρέχει όπως ακριβώς είναι η .NET. Ως σκοπός θα μπορούσε να οριστεί η επιτυχημένη παρουσίαση της εννοιολογίας υλοποίησης που ακολουθήθηκε και πως ο συνδυασμός διαφόρων εργαλείων, τακτικών και τεχνολογιών μπορούν να συνεργαστούν αρμονικά στο να επιφέρουν την επίτευξη ενός άρτιου αποτελέσματος.

Κεφάλαιο 5ο: Υλοποίηση Frontend με VueJs

5.1 Εισαγωγή

Το Vue.js Framework είναι ένα ισχυρό και ευέλικτο Framework της γλώσσας JavaScript που χρησιμεύει στη διευκόλυνση της ανάπτυξης Εφαρμογών Ενιαίας Σελίδας ειδάλλως γνωστές και ως Single Page Applications (SPA). Ο σύγχρονος σχεδιασμός που έχει ακολουθηθεί στην ανάπτυξή του Framework έχει διευκολύνει πολύ την εμπειρία ανάπτυξης του προγραμματιστή που το χρησιμοποιεί.

Ένα από τα καθοριστικά χαρακτηριστικά του Framework είναι ο προοδευτικός του χαρακτήρας. Σχεδιάστηκε ειδικά για να μπορεί να υιοθετηθεί σταδιακά από προγραμματιστές, επιτρέποντας τους να εισάγουν το νεοσύστατο framework σε ήδη υπάρχοντα έργα, σε αντίθεση με ορισμένα framework-ανταγωνιστές που υπαγορεύουν μια αυστηρή και συγκεκριμένη αρχιτεκτονική που πρέπει να εφαρμοστεί.

5.2 Πλεονεκτήματα / Αρχιτεκτονική του Vue Framework

Η βασική βιβλιοθήκη (core module) του Framework είναι ελαφριά και εύκολη στην ενσωμάτωση με άλλες βιβλιοθήκες και frameworks. Αυτή η δια λειτουργικότητα επιτρέπει σε προγραμματιστές να αξιοποιούν το framework παράλληλα με τα εργαλεία που προτιμούν και με τις υπάρχουσες βάσεις κώδικα (code-bases), είτε χρησιμοποιούνται δημοφιλείς βιβλιοθήκες JavaScript όπως η Axios για HTTP Requests είτε ενσωματώνονται βιβλιοθήκες γραφικών UI όπως η Vuetify και η Element UI.

Το Vue Framework υλοποιεί το σχεδιαστικό πρότυπο Model View Controller (MVC) του οποίου η γενική φιλοσοφία περιεγράφηκε σε προηγούμενη ενότητα . Ακολουθώντας το πρότυπο αυτό, μια διεπαφή (template)του project μπορεί να προβληθεί είτε πρόκειται για μια εφαρμογή κινητού ή για μια εφαρμογή για desktops , είτε για έναν ιστότοπο γενικής χρήσης.

Το Framework επίσης χαρακτηρίζεται από το πόσο φιλικό προς εκμάθηση είναι καθώς είναι ένα εξαιρετικό μέρος για να ξεκινήσουν οι αρχάριοι που δεν είχαν προηγούμενη τριβή με άλλα frameworks. Το μόνο που χρειάζεται να κατέχει ένας αρχάριος απο γνώσεις για να ξεκινήσει είναι τη θεμελιώδης κατανόηση της HTML,CSS,Javascript.Η σχεδιαστική αρχιτεκτονική του Framework και μερικές λειτουργίες του παρουσιάζονται παρακάτω

- Ευκολία εκμάθησης: Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο η Vue έχει μια ήπια καμπύλη εκμάθησης, καθιστώντας το προσιτό σε προγραμματιστές διαφορετικών επιπέδων και δεξιοτήτων να ξεκινήσουν την χρήση της. Ουσιαστική και μόνη βασική απαίτηση είναι η στοιχειώδης κατανόηση HTML,CSS,Javascript.
- Αρχιτεκτονική βασισμένη σε Συστατικά (Components): Η Vue ακολουθεί μια αρχιτεκτονική βασισμένη σε συστατικά, η οποία πρεσβεύει την επαναχρησιμοποίηση και τον αρθρωτό κώδικα. Αυτή η προσέγγιση επιτρέπει σε χρήστες της να δημιουργούν αυτοτελή compoments που μπορούν εύκολα να επαναχρησιμοποιηθούν στο ίδιο ή ακόμη και σε διαφορετικό project. Το Component Based Architecture καθιστά κομμάτια κώδικα / λογικής διατηρήσιμα και εύκολα κλιμακούμενα.
- Reactivity και Two-Way data binding: Παρέχει ένα πρωτοπόρο reactivity system , που βασίζεται σε Javascript Proxy objects, επιτρέποντας την πιο αποτελεσματική ενημέρωση των reactive δεδομένων και την δέσμευση αυτών(data-binding). Οπότε ,οποιαδήποτε αλλαγή υφίσταται στα δεδομένα αντανακλάται αυτόματα στο UI και το αντίστροφο με πολύ αποδοτικό τρόπο για το μηχάνημα που εκτελείται η εφαρμογή. Η αμφίδρομη δέσμευση δεδομένων ,εν ολίγοις διασφαλίζει ότι πολλαπλά Components που χρησιμοποιούν κοινά δεδομένα από την

ίδια πηγή, παραμένουν πάντα συγχρονισμένα και αντιδρούν σχεδόν ακαριαία στις αλλαγές που υποβάλλονται από οποιοδήποτε άλλο Component πάνω στο ίδιο δεδομένο. Αυτό το χαρακτηριστικό απλοποιεί τη διαχείριση του Global State της εφαρμογής ,μειώνοντας τον boilerplate κώδικα που σχετίζεται με τον δύσκολο χειροκίνητο συγχρονισμό δεδομένων.

- Βελτιστοποιημένες Επιδόσεις: Το Framework έχει σχεδιαστεί με πρωταρχικό γνώμονα την απόδοση. Χρησιμοποιεί την τεχνική / έννοια του Virtual DOM. Το Virtual DOM σε επέκταση του ήδη γνωστού DOM, ενημερώνει και απεικονίζει αποτελεσματικά μόνο τα στοιχεία της εφαρμογής που επηρεάζονται από τις αλλαγές δεδομένων που λαμβάνουν χώρα στα Components. Αυτή η προσέγγιση ελαχιστοποιεί τους περιττούς χειρισμούς που πρέπει να γίνονται στο DOM, με αποτέλεσμα η εφαρμογή να είναι ταχύτερη.
- Επίσημες Βιβλιοθήκες / Εργαλεία: Το Framework τέλος, παρέχει επίσημα εργαλεία όπως το Vue Router για δρομολόγηση στο SPA και το Vuex Store για διαχείριση του Global State της εφαρμογής. Αυτά τα εργαλεία ενσωματώνονται απρόσκοπτα και αναγκαστικά τις περισσότερες φορές με το Framework, προσφέροντας ένα συνεπές και καλά τεκμηριωμένο οικοσύστημα για τη δημιουργία σύνθετων εφαρμογών.

5.3 Φιλοσοφία των Συστατικών (Components)

Στην ενότητα αυτή θα περιγραφεί όσο πιο ολοκληρωμένα γίνεται ο ορισμός του τι είναι ένα Component καθώς και ο λόγος χρήσης του. Δεν είναι καθόλου τυχαίο πως τα Components έχουν εγκαθιδρυθεί ως η κύρια αρχιτεκτονική που ακολουθείται σε κάθε σύγχρονο Major Javascript Framework όπως React, Angular, Vue, Svelte κ.α.

Αν μπορούσε να τυπωθεί αυστηρά ενας ορισμός για τα components , αυτός θα ορίζονταν ως μια εννοιολογική υποδιαιρεση λειτουργιών και HTML Markup που χτίζεται με αυτοσκοπό την αυτοτέλεια του συστατικού και της λογικής του με σκοπό την επεκτασιμότητα, την εύκολη συντήρηση και τέλος την επαναχρησιμοποίηση του.

Ιστορικά τα Components μετέβησαν απο πολλαπλές εκδόσεις σχεδιαστικής υλοποίησης, όπου κάθε μια βρήκε χώρα σε διαφορετικά Javascript Frameworks. Ένα component εννοιολογικά και πέρα από συγκεκριμένες σχεδιαστικές υλοποιήσεις, πρεσβεύει την ομαδοποίηση HTML Markup, Javascript κώδικα και template styling (π.χ. CSS) κάτω από ένα namespace το οποίο ορίζεται από τον συγγραφέα / σχεδιαστή του.

Η προηγούμενη σχεδιαστική υλοποίηση που χρησιμοποιείται μέχρι και σήμερα, είναι η συγγραφή ξεχωριστών αρχείων .html , .js, .css κάτω από το ίδιο Component namespace. Για παράδειγμα έστω ότι το namespace του Component είναι TheHeader. Τότε έπρεπε να δηλωθούν 3 διαφορετικά αρχεία με διαφορετικές καταλήξεις που “δείχνουν” στο κύριο Component namespace. Δηλαδή η αντιστοιχία θα ήταν the-header.component.html , the-header.components.js, the-header.component.css και τέλος όλα αυτά ομαδοποιημένα κάτω από έναν φάκελο με το αντίστοιχο όνομα the-header-component .

Με την πάροδο του χρόνο ωστόσο και με την κατασκευή πολλαπλών έργων με την προαναφερθείσα αρχιτεκτονική αποδείχθηκε ότι ο παραπάνω τρόπος δεν είναι και τόσο ευέλικτος σε μεγάλες υλοποιήσεις, ή τουλάχιστον όχι τόσο όσο αρχικά σχεδιάστηκε να είναι. Έτσι η τελευταία σχεδιαστική υλοποίηση ενός Component που ονομάζεται Single File Component (SFC) ή Συστατικό Ενιαίου Αρχείου και ακολουθείται αυστηρά απο το Vue Framework, ομαδοποιεί τα 3 ξεχωριστά αρχεία σε ένα. Με το SFC οπότε χρειάζεται ένα αρχείο του οποίου το namespace θα ήταν TheHeader , με κατάληξη .vue εφόσον γράφεται για να αξιοποιηθεί από το Vue Framework και όχι 3 ξεχωριστά αρχεία.

Τα Single File Components που χρησιμοποιεί αποκλειστικά πλέον η Vue και ιδιαίτερα στις τελευταίες της εκδόσεις διαιρούν το μοναδικό αυτό αρχείο σε 3 διακριτά μέρη. Το Template Section , Script Section και Style Section.

- Template Section: Υποδηλώνεται με HTML tag δεσμευμένης σημασίας απο το framework με αναπαράσταση <template> </template>. Μέσα στο παρόν tag υπάρχει η δυνατότητα να εμφωλευθούν πολλαπλά άλλα HTML Tags που στο σύνολο απαρτίζουν το HTML Template (DOM) του συστατικού. Επίσης πέραν από τα γνωστά διαθέσιμα html tags της HTML5 μπορούν να δηλωθούν προς χρήση και components που έχουν διαφορετικά namespaces. Τα εν λόγω components αναφέρονται ως Child Components του Component , και επιπροσθέτως το Component αυτό καθ' αυτό αναφέρεται σε τέτοιες περιπτώσεις ως Parent ή Root Component.
- Script Section: Υποδηλώνεται με HTML tag δεσμευμένης σημασίας απο το framework με αναπαράσταση <script> </script>. Μέσα στο tag αυτό δηλώνεται όλη η επιχειρησιακή λογική που πρέπει να κατέχει ένα Component. Πιο συγκεκριμένα μέθοδοι ,LifeCycle Hooks, χρήση Composables καθώς και τα δεδομένα reactive ή μη , που θα χρησιμοποιηθούν απο το DOM (Template). Η συνηθέστερη μορφή οργάνωσης της προαναφερθείσας λογικής είναι υπό μορφή μεθόδων.
- Style Section: Υποδηλώνεται με HTML tag δεσμευμένης σημασίας απο το framework με αναπαράσταση <style> </style>. Μέσα στο tag αυτό δηλώνονται τα styling css rules που θα ακολουθούν τα HTML elements που δηλώθηκαν μέσα στο <template> tag του Component κάνοντας χρήση μιας στιλιστικής μηχανής. Ευρέως χρησιμοποιούμενες είναι η κλασσική CSS, ενώ υπάρχει και η δυνατότητα χρήσης πιο νέων σύγχρονων μηχανών και pre-processors, όπως Less, Sass, Scss, PostCss κ.α. Αξίζει επίσης να σημειωθεί ότι αυτοί οι στιλιστικοί κανόνες επηρεάζουν μόνο τα elements του Component όπου έχουν δηλωθεί ότι ανήκουν.

Συνοψίζοντας, το Component Based Structure και η φιλοσοφία των Single File Components που εφαρμόζει το Vue Framework βοηθάει στην εύκολη επέκταση της λογικής ενός component, στην συντήρηση του εάν αυτή καταστεί αναγκαία και στην επαναχρησιμοποίηση του σε διάφορα σημεία της εφαρμογής. Τέλος ας σημειωθεί ότι η δομή των SFC συμπεριφέρεται πολύ καλύτερα σε επαγγελματικά projects που απαρτίζονται από εκατοντάδες components σε αντίθεση με την προηγούμενη επικρατέστερη αρχιτεκτονική όπου χώριζε το μοναδικό αρχείο σε 3 ξεχωριστά.

5.4 Γεγονότα Κύκλου Ζωής (Lifecycle Hooks)

Τα γεγονότα κύκλου ζωής ακολουθούνται ως φιλοσοφία σε πολλαπλά frameworks που βρίσκονται στο εμπόριο καθώς επίσης υλοποιούνται και στο Vue Framework με μερικές διαφοροποιήσεις. Ας επισημανθεί ότι τα lifecycle hooks είναι άρρηκτα συνδεδεμένα με την λογική του Components Based Architecture σε κάθε Framework. Ως ιδέα ένα Lifecycle Hook ορίζεται ως η υπογραφή μιας μεθόδου που δηλώνεται ως ένα event στον μηχανισμό του εκάστοτε Framework και κύρια μέριμνα της συγκεκριμένης μεθόδου είναι να εκτελείται σε ένα συγκεκριμένο **σημείο ζωής** ενός Component. Με ορθή χρήση των αντίστοιχων hooks ένας προγραμματιστής μπορεί να διατάξει το Component ,σε διάφορα σημεία της ζωής του που αντικατοπτρίζονται απο τα hooks αυτά, να εκτελέσει συγκεκριμένα κομμάτια κώδικα και συγκεκριμένες διαδικασίες αυτοματοποιημένα.

5.4.1 Components Lifecycle στο Vue Framework

Τα hooks δηλώνονται διαφορετικά σε κάθε Javascript Framework όμως , όλα ακολουθούν την ίδια λογική. Ο τρόπος που δηλώνονται καθορίζεται αποκλειστικά και μόνο από το Development Team του κάθε framework. Έτσι μερικά από τα πιο κυρίως χρησιμοποιούμενα lifecycle hooks στο Vue Framework και συγκεκριμένα στο Composition API, που παρουσιάζονται ως Composable Functions, περιγράφονται ως εξής:

- onBeforeMount(): Η onBeforeMount() μέθοδος καλείται ακριβώς πριν από το render του Component στο DOM.
- onMounted(): Η μέθοδος onMounted() ενεργοποιείται όταν το Component έχει τοποθετηθεί επιτυχώς στο DOM. Η κλήση της μεθόδου αυτής υποδηλώνει οτι το template και το script κάθε component έχει γίνει compiled, έχει απεικονιστεί στον φυλλομετρητή και είναι έτοιμο να ξεκινήσει την εκτέλεση τη επιχειρησιακής λογικής του.
- onBeforeUpdate(): Η συγκεκριμένη μέθοδος εκτελείτε πριν από κάθε ενημέρωση που υφίσταται ένα Component. Οι ενημερώσεις αυτές λαμβάνουν χώρα λόγω αλλαγών στα reactive δεδομένα που εξαρτάται το εκάστοτε Component. Χρησιμοποιείται για την εκτέλεση λειτουργιών ή για λειτουργίες προετοιμασίας πριν από το νέο render του Component με τα ενημερωμένα δεδομένα.
- onUpdated(): Η onUpdated() καλείται μετά το επιτυχημένο re-render και την επιτυχημένη επαναπροβολή του template του Component. Σηματοδοτεί ότι όντως έχουν εφαρμοστεί στο DOM οποιεσδήποτε αλλαγές προστάζουν οι αλλαγές στα reactive δεδομένα / εξαρτήσεις του Component.
- onBeforeUnmount(): Η μέθοδος αυτή ενεργοποιείται ακριβώς πριν το Component πρόκειται να εξαλειφθεί από το DOM. Παρέχει την ευκαιρία να εκτελεστούν εργασίες καθαρισμού.
- onUnmounted(): Η μέθοδος onUnmounted() ενεργοποιείται όταν το Component έχει εξαλειφθεί πλήρως από το DOM. Σε αυτό το στάδιο, το Component και τα δεδομένα του δεν αποτελούν πλέον μέρος της εφαρμογής.
- onErrorCaptured(): Η μέθοδος onErrorCaptured() επιτρέπει τη σύλληψη και τον χειρισμό σφαλμάτων που συμβαίνουν μέσα σε ένα Component ή σε κάποιο Child Component του κατα τη διάρκεια εκτέλεσης. Παρέχει έναν καθολικά αποδεκτό τρόπο για την ασφαλή διαχείριση σφαλμάτων.

5.5 Vue APIS

Πριν ξεκινήσει η περαιτέρω ανάλυση ας αναφερθεί ότι η Frontend εφαρμογή χρησιμοποιεί κατα κόρον το Composition API, όπου παρακάτω θα γίνει η ανάλυση του, και την έκδοση 3.2.5 του Vue.js Framework.

5.5.1 Ορισμός Vue Api

Ως Vue API (Application Programming Interface) αναφέρεται ένα σύνολο κανόνων και πρωτοκόλλων που πρέπει αναγκαστικά να τηρηθούν ώστε να είναι δυνατή η επικοινωνία προγραμματιστή με framework με απώτερο σκοπό την υλοποίηση μιας εφαρμογής. Ένα API, καθορίζει τις διαθέσιμες μεθόδους, διαδικασίες, πρακτικές και τεχνικές που ο χρήστης του μπορεί να υιοθετήσει ώστε να φέρει το framework στις απαιτήσεις της εφαρμογής του.

5.5.2 Ορισμός Api

Το Options API του Framework, όντας η πρώτη προσπάθεια του Vue Development Team είναι το interface που χρησιμοποιείται κατα κόρον σε παλαιότερες εκδόσεις του Framework (Vue 2.X) αλλά χωρίς να είναι απαγορευτικό, μπορεί να χρησιμοποιηθεί και στην έκδοση που χρησιμοποιεί η εφαρμογή. Ένα Component (Συστατικό) που δηλώνεται οτι θα χρησιμοποιεί το Options API, δεσμεύεται ότι θα χρησιμοποιήσει τις επιλογές (options) data, methods, watch και computed όπου και θα αναλυθούν παρακάτω.

- Data: Η επιλογή που επιτρέπει στον προγραμματιστή να δηλώσει τα δεδομένα που θα χρησιμοποιήσει το component. Ορίζεται ως μέθοδος που επιστρέφει ένα αντικείμενο που περιέχει τα δεδομένα αυτά ως ζευγάρια κλειδιού-τιμής. Επίσης στο αντικείμενο αυτό

επιστρέφονται και τα reactive (αντιδραστικά) δεδομένα που παρακολουθούνται αυτόματα από τον μηχανισμό του Framework.

- Computed Properties: Η επιλογή που επιτρέπει να δηλωθούν computed properties.
- Ως computed property ορίζεται το δεδομένο το οποίο η τιμή του εξαρτάται και προέρχεται από άλλα δεδομένα είτε άλλα computed properties. Τα δεδομένα αυτά μπορεί να είναι reactive, με αποτέλεσμα μετά την αλλαγή της τιμής του δεδομένου να επαναυπολογίζεται αυτόματα και η τιμή του αποτελέσματος του computed property. Άρα εν τέλη το αποτέλεσμα ενος computed property αλλάζει εφόσον κάποια εξαρτώμενη τιμή που συμβάλλει στον υπολογισμό του αλλάζει επίσης. Είναι η βασική πρωταρχική λειτουργία που παρείχε το Framework στο ξεπερασμένο πλέον Options API ώστε να μπορεί ένας χρήστης του να μετατρέπει μια μεταβλητή απο στατική σε reactive.
- Methods: Η επιλογή του methods επιτρέπει στον προγραμματιστή να δηλώσει ένα σύνολο ενεργειών με διάφορες λειτουργίες που μπορούν να κληθούν μέσω άλλων μεθόδων ή μέσω του HTML Template του Συστατικού. Μία μέθοδος ορίζεται ότι έχει μία και μόνο συγκεκριμένη σημασιολογική λειτουργία σε όλη την έκταση του Component.
- Watchers: Η επιλογή του watcher παρέχει έναν αυτοματοποιημένο τρόπο αντίδρασης στην μεταβολή τιμής ενός δεδομένου του Component. Ουσιαστικά ενας watcher μπορεί να εκτελέσει κώδικα βάση της αλλαγής τιμής μιας reactive μεταβλητής. Μια πολυχρησιμοποιημένη τεχνική που χρησιμοποιεί watchers είναι στην εναλλαγή του UI απο Desktop Mode σε Laptop Mode και τέλος σε Mobile Mode. Αν και παρουσιάζονται ως πολύ παρόμοιοι με τα computed properties που παρουσιάστηκαν προηγουμένως, υπάρχουν διαφορές που μπορεί κανείς να αντιληφθεί μόνο όταν αποκτήσει ένα μεγάλο μέρος εμπειρίας από την συνεχή τριβή του με το Framework.

Το Options API, παρείχε μια αρκετά απλή δομή υλοποίησης ενος SFC στο Framework. Ωστόσο σε μεγάλες εφαρμογές με εκατοντάδες Components αποδείχθηκε εν τέλη ότι είναι αρκετά δύσκολο στην συντήρηση και δύσκολα επεκτάσιμο. Για του παραπάνω λόγους το Vue Development Team απο την Έκδοση Vue 3 και έπειτα παρουσίασε το Composition API που φτιάχτηκε για να λύσει θέματα που παρουσιάστηκαν απο το Options API και επίσης να προσθέσει, να βελτιστοποιήσει και να απλουστεύσει ορισμένες λειτουργίες.

5.5.3 Comsposition API και Πλεονεκτήματα έναντι του Options API

Με το Composition API που εισήχθη στην έκδοση της Vue 3 και έπειτα, οι χρήστες του Framework πλέον μπορούν να χρησιμοποιούν μόνο μεταβλητές και μεθόδους μέσα σε μία setup μέθοδο που έχει οριστεί απο το Development Team. Αυτές οι μεταβλητές και οι μέθοδοι έρχονται να καλύψουν πλήρως κάθε επιλογή που παρείχε προηγουμένως το Options API. Επίσης στο Composition API έγινε εισαγωγή της έννοιας του Composable που ουσιαστικά αφαιρεί την λογική από ένα Component και την αφαιρεί εννοιολογικά (abstraction) κάνοντας τα συγκεκριμένα κομμάτια κώδικα επαναχρησιμοποιήσιμα σε όλα τα Συστατικά της εφαρμογής είτε ακόμη και σε εξ ολοκλήρου άλλες εφαρμογές.

Ενας καλός παραλληλισμός του Composable είναι τα Services και η ιδεολογία τους που παρέχοντα εδώ και χρόνια από ανταγωνιστικά Framework όπως η Angular.

Επιπλέον κάθε άλλο option όπως computed watcher και data έχει αντικατασταθεί από τις built-in μεθόδους ref() και reactive() που αναπτύχθηκαν από το Vue Development Team και δεν είναι τίποτα άλλο παρά composable functions. Μια σύντομη σύνοψη των προσθηκών / αλλαγών στο Composition API είναι:

- Τα Composable Functions. Μέθοδοι που ενθυλακώνουν λογική και ομαδοποιούνται αντίστοιχα για την δόμηση περίπλοκων υλοποιήσεων.

- Το Βελτιστοποιημένο Reactivity System. Το Composition API αξιοποιεί καλύτερα το reactivity system που προϋπήρχε, επιτρέποντας στους χρήστες του να δημιουργούν μεταβλητές με αντιδραστικό state χρησιμοποιώντας τις νέες μεθόδους ref() και reactive() και όχι τα μόνο τα template refs και τα computed properties που ήταν δεσμευτικά στο Options API.
- Πλήρης Υποστήριξη TypeScript. Το Composition API προσθέτει βελτιωμένη υποστήριξη για Typescript. Χρησιμοποιώντας μεθόδους και τύπους πλέον, η TypeScript μπορεί να συμπεράνει συν επαγωγικά τους τύπους μεταβλητών και τύπους αποτελεσμάτων που επιστρέφονται απο μεθόδους. Έτσι εν τέλη υπάρχει καλύτερη δυνατότητα auto-complete κατά την διαδικασία συγγραφής και πιο αυστηρό type checking που προστατεύει από λάθη κατα την εκτέλεση του πηγαίου κώδικα.

5.6 Προστάξεις (Directives)

Τα directives είναι concept που επίσης προϋπήρχε σε πολλαπλά παλαιότερα frameworks και φυσικά βρήκε έδαφος υλοποίησης και στο Vue Framework. Ως directives ορίζονται αυστηρές οδηγίες που έχει εισάγει το Development Team με σκοπό την αντικατόπτριση αλλαγών στο HTML Template του Component. Αυτή η οδηγία είναι ένα ειδικό χαρακτηριστικό (attribute) που μπορεί να εφαρμοστεί σε HTML Elements και Child Components ώστε να προστεθεί προσαρμοσμένη συμπεριφορά και λειτουργικότητα. Τα directives χρησιμοποιούνται για τον χειρισμό του DOM, την εφαρμογή δυναμικού στυλ σε elements , δέσμευση δεδομένων με μεταβλητές του script (data-binding) , διαχείριση Javascript Events και πολλά άλλα.

Ένα directive στο Framework δηλώνεται προστέθοντας πάντα μπροστά το πρόθεμα "**v-**" ακολουθούμενο από το όνομα του directive προς χρήση. Για παράδειγμα μερικά σε ισχύ και πολυχρησιμοποιημένα directives είναι το **v**-if και το **v**-for. Τα directives αυτά χρησιμοποιούνται για να δηλώσουν συγκεκριμένο τρόπο ή συμπεριφορά όπου τα elements ή τα child components θα πρέπει να συμπεριφέρονται με τα δεδομένα του Component.

Όταν ένα directive εφαρμόζεται σε ένα στοιχείο, επεξεργάζεται από τον complier της γλώσσας και το runtime της Vue, το οποίο με τη σειρά του μετασχηματίζει τη συμπεριφορά ή την εμφάνιση του στοιχείου με βάση το τι προστάζει το εκάστοτε directive. Τα directives όπως προαναφέρθηκε, μπορούν να χρησιμοποιηθούν για την υπό συνθήκη εμφάνιση στοιχείων (conditional render), την επανάληψη συλλογών δεδομένων (looping), τη δέσμευση δεδομένων σε μεταβλητές (data-binding), την διαχείριση συμβάντων (events) και το χειρισμό του DOM απο τα πιο βασικά.

Το Development Team έχει μεριμνήσει και παρέχει αρκετά ενσωματωμένες directives. Επίσης υπάρχει και η δυνατότητα να δημιουργηθούν και προσαρμοσμένα επαναχρησιμοποιούμενα directives απο τον κάθε προγραμματιστή προς κάλυψη των αναγκών του project.

Παρακάτω αναφέρονται τα πιο διάσημα από αυτά που η χρήση τους καθίσταται αναγκαίας για την εκπλήρωση οποιουδήποτε project.

v-if: Χρησιμοποιείται για την υπό συνθήκη εμφάνιση ή επανεμφάνιση ενός element ή block απο elements, βάση του boolean αποτελέσματος μιας συνθήκης όπου υπολογίζεται κάθε φορά κατα το runtime.

Παράδειγμα: <div v-if= " 1 < 2 " >Hello, Vue!

Στο παραπάνω απλουστευμένο παράδειγμα εφόσον το αποτέλεσμα της ανισότητας είναι αληθές ο μηχανισμός της Vue θα εμφανίσει το div element στο DOM.

v-show: Χρησιμοποιείται για την στιλιστική απόκρυψη ενός element επηρεάζοντας το css display property με βάση το αληθές ή μη αποτέλεσμα μιας συνθήκης.

Παράδειγμα: <div v-show="1 > 2">This is a hidden message.

Στο παραπάνω παράδειγμα αν το αποτέλεσμα είναι ψευδές θα προστεθεί αυτόματα στο div element το css property display:none; ειδάλλως θα κρατηθεί η default τιμή που είναι display:block.

v-for: Χρησιμοποιείται για την εμφάνιση ενός element ή ενός block απο elements με βάση ένα πίνακα, ένα αντικείμενο ή έναν πίνακα αντικειμένων. Το παρών directive κάνει loop βάση του input που δέχεται σαν πίνακα και εμφανίζει τόσα elements όσο και το μήκος του πίνακα.

Παράδειγμα:

```
<ul>
    <li v-for="item in items">{{ item }}</li>
</ul>
```

Για το παραπάνω παράδειγμα έστω ότι το items είναι ένας πίνακας απο αλφαριθμητικά με μορφή [‘1’,‘2’,’3’,’4’]. Το directive δεδομένου του μήκους του πίνακα αυτού θα εμφανίσει 4 διαφορετικά elements που ως innerHTML θα έχουν την τιμή που φέρει το κελί του πίνακα στην κάθε μεταβολή του δείκτη της επανάληψης. Προς καλύτερη κατανόηση παρατίθεται το αποτέλεσμα εκτέλεσης που θα εμφανιστεί στην ιστοσελίδα.

```
<ul>
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
</ul>
```

v-bind (:attr): Το directive αυτό χρησιμοποιείται για τη δυναμική δέσμευση (data-binding) και αντικατάσταση της τιμής ενός attribute ενός element με την τιμή μεταβλητής που βρίσκεται στο script tag του Component

Παράδειγμα:

```
<template>
    <a v-bind:src="url" >
</template>
<script>
setup(){
    const url= “https://www.google.com”;
    return {url}
}
```

```
</script>
```

Στο παραπάνω παράδειγμα η τιμή του src attribute στο <a> tag δεσμεύεται δυναμικά κατα το runtime του application με την τιμή της μεταβλητής “url” που υφίσταται στο script tag. Κατα το κλικ στον υπερσύνδεσμο <a> θα πραγματοποιηθεί πλοήγηση στο δηλωθέντα ιστότοπο.

Λόγω της εκτεταμένης χρήσης του παραπάνω directive στο Framework το Vue Development team έχει εισάγει συντόμευση του directive και αντί για v-bind μπορεί να χρησιμοποιηθεί το σύμβολο “ : ” πριν απο το επιθυμητό attribute με ακριβώς το ίδιο αποτέλεσμα.

v-on (@evt): Το v-on directive επιτρέπει την σύνδεση ακροατών σε διάφορα javascript γεγονότα σε ένα element ή component αντίστοιχα. Η δήλωση του v-on ακολουθούμενο από ένα javascript event επιτρέπει την κλήση της μεθόδου διαχείρισης του event όταν συμβεί ενεργοποίηση του συγκεκριμένου γεγονότος.

Παράδειγμα:

```
<template>
        <button v-on:click="handleClick">Click me</button>
</template>
<script>
setup(){
        function handleClick(){
        console.log(“I was clicked”);
        };
        return {handleClick}
}
</script>
```

Στο παραπάνω παράδειγμα αν πραγματοποιηθεί click στο button τότε η αντίστοιχη μέθοδος διαχείρισης που έχει δηλωθεί στο script σαν event handler του onClick event του συγκεκριμένου element, καλείται επιτυχώς με αποτέλεσμα την εμφάνιση του μηνύματος στην κονσόλα.

Λόγω της εκτεταμένης χρήσης του παραπάνω directive απο την κοινότητα, το Vue Development team έχει εισάγει συντόμευση του directive και αντί για v-on μπορεί να χρησιμοποιηθεί το σύμβολο “ @ ” πριν απο το επιθυμητό event που πρέπει να πραγματοποιηθεί η σύνδεση με ένα event handler method.

v-model: Το v-model directive χρησιμοποιείται για την δέσμευση δεδομένων σε διπλή κατεύθυνση (two-way-data-binding) ανάμεσα σε τιμές attribute των html elements του template στο Component και στην αντίστοιχη δεσμευμένη τιμή της μεταβλητής που βρίσκεται στο script του Component. Η διαφορά με το κανονικό v-bind directive είναι ότι πραγματοποιώντας μια αλλαγή στην τιμή του attribute που είναι συνδεδεμένη με την τιμή μιας μεταβλητής στο script tag,τόσο απο το template όσο και απο το script, η τιμή αντικατοπτρίζεται και είναι διαθέσιμη ακαριαία και στο δύο αυτά σημεία, σε αντίθεση με το συμβατικό v-bind που επιτρέπει την αλλαγή της τιμής μόνο μέσα από το script tag και εμφάνιση της αλλαγής στο template (one-way-data-binding). Το συγκεκριμένο directive βρίσκει χρήση σε child

components ανάλογα με την απαίτηση αλλά δεσμευτικά μόνο σε <input>,<select> και <textarea> απο HTML5 tags.

v-text: Το v-text directive επιτρέπει την ανάθεση μιας αλφαριθμητικής τιμής ως το innerText attribute ενός HTML5 tag. Καθίσταται ιδιαίτερα χρήσιμο directive καθώς υφίσταται pre build escaping μηχανισμός στο directive που προστατεύει την εφαρμογή από επιθέσεις τύπου injection. Επίσης ο ίδιος μηχανισμός κάνει μετατροπή οποιασδήποτε εντολής και τιμής μεταβλητής σε αλφαριθμητικό.

v-htm: Το v-html directive επιτρέπει την δυναμική ανάθεση HTML κώδικα ως το innerHTML ενός HTML5 tag απο το script tag. Εν ολίγοις το v-html επιτρέπει την δυναμική αναπαράσταση (render) HTML5 Markup σε ένα HTML block.

Παράδειγμα:

```
<template>
    <div v-html="htmlToRender"></button>
</template>
<script>
setup(){
    const htmlToRender = <div><h1> I am dynamically rendered </h1></div>
    return {htmlToRender }
}
```

Ως αποτέλεσμα εκτέλεσης του παραπάνω παραδείγματος ο φυλλομετρητής θα εμφανίσει το παρακάτω αποτέλεσμα.

```
<div>
    <div>
        <h1> I am dynamically rendered </h1>
    </div>
</div>
```

v-slot: Το v-slot directive χρησιμοποιείται σε συνδυασμό με τα λεγόμενα named slots των Components. Το directive παρέχει έναν τρόπο για τη μετάδοση περιεχομένου από το Parent Component στο Child Component, επιτρέποντας πιο ευέλικτες και επαναχρησιμοποιήσιμες συνθέσεις Components.

5.7 Χρήση του Vue Framework στην Υλοποίηση της Εφαρμογής

Το FrontEnd application ,όπως έχει αναφερθεί σε σημεία , έχει υλοποιηθεί πλήρως με την χρήση του Vue Framework. Κάνοντας εκτεταμένη χρήση των τεχνολογιών και των καινοτομιών του συγκεκριμένου Javascript Framework και ειδικότερα της έκδοσης 3 με Composition API επιτεύχθηκε η ανάπτυξη μιας πλήρως δυναμικής , διαδραστικής και φιλικής διεπαφής προς τον χρήστη.

Σε περισσότερες τεχνικές λεπτομέρειες ακολουθήθηκε σε μεγάλο βαθμό το Vue Style Guide με τις προτεινόμενες καθώς και ισχυρά προτεινόμενες συστάσεις του με σκοπό την εύκολη αναγνωσιμότητα του κώδικα από άλλους προγραμματιστές. Επίσης τηρώντας τον παραπάνω οδηγό και ακολουθόντας

συνεπείς και καθολικά αποδεκτές τεχνικές ανάπτυξής έχει επιτευχθεί ένα καθαρό , συντηρήσιμο και εύκολα κατανοητό codebase τόσο από κάποιον με μικρή όσο και από κάποιον με μεγάλη εμπειρία χρήσης του Framework.

Το application ακολουθεί αυστηρό Component Based Architecture οργανωμένο σε φακέλους. Ο κάθε φάκελος απευθύνεται σε μία εννοιολογικά ξεχωριστή λειτουργικότητα. Ενας άγραφος κανόνας / τεχνική που ακολουθήθηκε είναι ότι το κάθε front-end route θα πρέπει ιδανικά να είναι ένα ξεχωριστό Root Component με περισσότερα του ενός child components που διαιρούν όσο γίνεται καλύτερα της πολυπλοκότητα της απαίτησης.

Μία σύσταση που εφαρμόστηκε είναι ότι όλα τα κοινά επαναχρησιμοποιήσιμα Components στεγάζονται κάτω από τον φάκελο **Base** και έχουν επίσης ως πρώτο συνθετικό αυτή την κωδική ονομασία. Για παράδειγμα ένα Loading Spinner Component που ακολουθεί το Vue Style Guide πρέπει να μετονομαστεί σε PascalCase ώς **BaseLoadingSpinner.vue**. Γενικότερα, το συγκεκριμένο πατρόν ονομασίας υποδηλώνει στον προγραμματιστή ότι όλα τα Components που ξεκινούν από Base χρησιμοποιούνται σε περισσότερα του ενός σημείου μέσα στην εφαρμογή.

Μια επιπλέον σύσταση που εφαρμόστηκε για την αναγνώριση τον μοναδικών Component σε όλη την έκταση της εφαρμογής είναι η πρόσθεση του προθέματος **The** πριν την τελική ονομασία του. Για παράδειγμα ένα Component με όνομα **TheHeader.vue** υποδηλώνει ότι το component αυτό εμφανίζεται μία και μόνο μία φορά σε όλη την εφαρμογή.

Περαιτέρω έχει γίνει αποκλειστική χρήση του Composition API σε όλα τα Components και όλων των νεοσύστατων τεχνικών που φέρει αυτό. Για παράδειγμα η δια δραστικότητα που φέρει η εφαρμογή είναι αποτέλεσμα του νέου εκλεπτυσμένου reactivity system που προσφέρει την παρακολούθηση των αλλαγών σε περίπλοκα δια δραστικά δεδομένα πιο βελτιστοποιημένα απο προηγούμενες εκδόσεις.

Για την διατήρηση του Global State της εφαρμογής χρησιμοποιείται η επίσημα υποστηριζόμενη βιβλιοθήκη από το ίδιο το Vue Development Team που ονομάζεται Vuex Store.

Υλοποιώντας τα ανάλογα getters, mutations και actions επιτεύχθηκε η παρακολούθηση πολλαπλών States σε όλη την εφαρμογή καθώς υπάρχουν σημεία και μεταβλητές που επηρεάζονται από πολλαπλά σημεία μέσα στην εφαρμογή και η μεταβολή συγκεκριμένων τιμών φέρει αλλαγές σε περισσότερα του ενός Component. Ένα τρανταχτό παράδειγμα προς κατανόηση του τι ορίζεται ως State μιας εφαρμογής είναι το παρακάτω.

Για παράδειγμα η πληροφορία του αν η εφαρμογή παρουσιάζεται σε κάποιον που έχει κάνει log-in ή όχι μπορεί να χαρακτηριστεί ως δύο διαφορετικά διακριτά State που πρέπει να ανταποκριθεί η εφαρμογή. Μερικά components αναλόγως με την τιμή μιας μεταβλητής ονόματί **IsLoggedIn:boolean** που έχει δηλωθεί στο **Vuex Store** πρέπει να αντιδρούν διαφορετικά και να προβαίνουν σε διαφορετικές ενέργειες. Καθώς το Store μπορεί να αποθηκεύσει απο απλές boolean μεταβλητές μέχρι περίπλοκα object η φιλοσοφία είναι καθολική και περιγράφεται παρακάτω. Οι μεταβλητές που δηλώνονται στο Store μπορούν να αλλάξουν τιμές από πολλά σημεία μέσα σε πολλαπλά Components μέσα από ανάλογα αυστηρά σημεία πρόσβαση ως μέθοδοι που ονομάζονται actions , μπορούν να ανακτηθούν απο τις ανάλογες αυστηρές μεθόδους που ονομάζονται getters αλλά είναι αναγκαστικό να υπάρχει μόνο ένα διακριτό instance της μεταβλητής που αντικατοπτρίζει το εν ενεργεία State της κάθε φορά και επιβάλλεται να είναι γνωστό σε όλο το μήκος της εφαρμογής μόνο σε αυτούς που τους ενδιαφέρει η συγκεκριμένη τιμή

Τέλος η εφαρμογή κάνει εκτεταμένη χρήση Προσαρμοσμένων Composable Μεθόδων όπου η καθεμία είναι γραμμένη για να εξυπηρετεί μια συγκεκριμένη λογική λειτουργία. Εφόσον τα composables όπως έχει αναφερθεί προηγουμένως έχουν την φιλοσοφία των Services , είναι λογικό να είναι εννοιολογικά αποκεντρωμένα απο οποιοδήποτε Component και να έχουν την δυνατότητα να κληθούν μέσα σε και από οποιοδήποτε από αυτά. Τα Composable function έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να σερβίρουν δεδομένα και να εκτελούν λογική που έχει γραφτεί μία φορά και μπορεί η εκάστοτε λογική καθώς και όλα τα δεδομένα που κουβαλάνε να επαναχρησιμοποιηθούν σε όλη την εφαρμογή απο οποιοδήποτε Component κριθεί αναγκαίο. Όλα τα προηγούμενα σε συνδυασμό με το reactivity system του Framework προσφέρουν αποκεντρωμένο κώδικα και reactive δεδομένα που γίνονται inject σε ένα Component και το εκάστοτε Component αντιδρά αναλόγως σε αλλαγές που μπορούν να υποβληθούν από τρίτα Components πάνω στα ίδια δεδομένα που εν γένη μπορεί να εξαρτώνται και άλλα Components.

5.8 Επίλογος

Συνοψίζοντας για το συγκεκριμένο κεφάλαιο στο σύνολο του η ορθή χρήση ενός FrontEnd Framework καθώς και των τεχνικών που παρέχει όπως για παράδειγμα η Vue 3 με τα αντίστοιχα composables, directives, επαναχρησιμοποιήσιμα components, ορθή χρήση του VueStyleGuide και του reactivity system μπορούν να αποφέρουν εκπληκτικά τελικά αποτελέσματα κάνοντας την εφαρμογή βελτιωμένη ως προς την απαίτηση στου πόρους που απαιτούνται για την εκτέλεση της, πλήρως δια δραστική και το σημαντικότερο φιλική και εύχρηστη τόσο προς τον χρήστη όσο και προς τον προγραμματιστή που το χρησιμοποιεί.

Κεφάλαιο 6ο: Λειτουργία / Φιλοσοφία Εφαρμογής

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα παρουσιαστεί η λειτουργία της εφαρμογής στο σύνολό της. Για λόγους απομόνωσης των λειτουργιών της θα πραγματοποιηθεί ένας διαχωρισμός σε τέσσερις υπό ενότητες, όπου θα αναλυθούν με περισσότερες λεπτομέρειες οι λειτουργίες ανά κατηγορία χρήστη, τα δικαιώματα που φέρουν αυτοί εντός της εφαρμογής, καθώς και γενικότερες πληροφορίες σχετικά με τη λειτουργία της.

Στην πρώτη υπό ενότητα θα περιγραφεί η πραγματοποίηση της λειτουργίας του Log In μέσω του Ihu Login Portal καθώς και η ενσωμάτωση της ήδη υπάρχουσας υπηρεσίας του Τμήματος τόσο με το μπροστά μέρος (FrontEnd), όσο και με το πίσω μέρος της (BackEnd) της εφαρμογής.

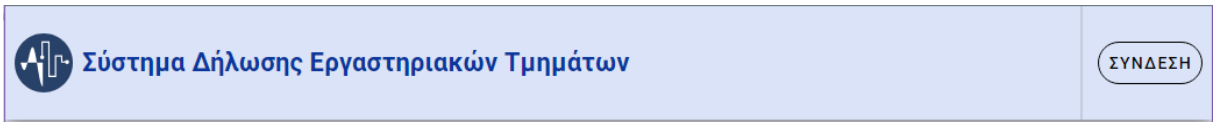
Πέραν της πρώτης υπό ενότητας και μέχρι τέλους του Κεφαλαίου κάθε επόμενη υπό ενότητα θα αναφέρει αναλογικά τα δικαιώματα και τις λειτουργίες που έχει ο κάθε χρήστης όταν πλοηγείτε στην εφαρμογή και του έχει ανατεθεί κατα περίπτωση ένας διαφορετικός ρόλος. Η ανάλυση ξεκινάει από τον Ρόλο του Διαχειριστή καθώς αυτός φέρει τις πιο αποκλειστικές λειτουργίες και τελειώνει στον Ρόλο του Φοιτητή καθώς είναι αυτός που φέρει την πιο μειωμένη πρόσβαση στην εφαρμογή.

Αξίζει να πραγματοποιηθεί η ενθύμηση ότι οι Ρόλοι ενός χρήστη της εφαρμογής λειτουργούν επαυξητικά και ότι η παραπάνω αναβάθμιση ενός Καθηγητή σε Διαχειριστή δεν σημαίνει επ ουδενί ότι χάνει τα προηγούμενα δικαιώματα και πρόσβαση που κατείχε ως Καθηγητής.

6.2 Είσοδος στην Εφαρμογή

Η είσοδος στη εφαρμογή γίνεται από το FrontEnd τμήμα της εφαρμογής και εξυπηρετείται από το κουμπί “ΣΥΝΔΕΣΗ” που στεγάζεται στην κεφαλίδα της ιστοσελίδας κατα την έναρξη της συνεδρίας πλοήγησης. Κατα την ενέργεια του κλικ γίνονται αλλεπάλληλα κάποιες διαδικασίες που εξυπηρετούνται από το πρωτόκολλο HTTPS. Ως επιτυχές αποτέλεσμα των διαδικασιών αυτών είναι η επιτυχημένη αυθεντικοποίηση και η ανάθεση ενός Ρόλου στον χρήστη.

Οι αλλεπάλληλες διαδικασίες αυτές περιγράφονται με απλουστευμένο τρόπο παρακάτω.



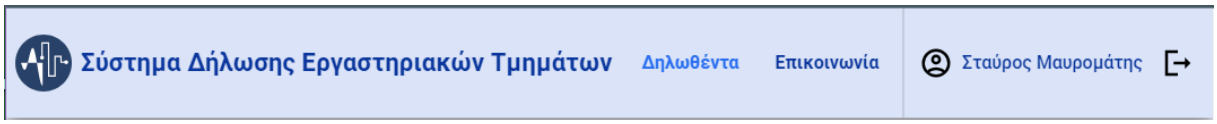
Σχήμα 6.1: Κεφαλίδα Ιστοσελίδας μη Εξουσιοδοτημένου χρήστη

Η πρώτη HTTPS διαδικασία είναι πρός το portal του τμήματος που με χρήση συγκεκριμένων Tokens, που έχουν εκδοθεί πρωτίστως από άλλη έμπιστη υποδομή του Τμήματος για την εφαρμογή, πραγματοποιείται το ανάλογο HTTPS Redirect στην σελίδα του Τμήματος που είναι υπεύθυνη για την αποδοχή χρήσης της εφαρμογής από τον χρήστη. Ενας χρήστης οφείλει να επιτρέψει την προώθηση των δεδομένων του προς χρήση από την εφαρμογή για να συνεχιστεί η διαδικασία της Αυθεντικοποίησης/Εξουσιοδότησης.

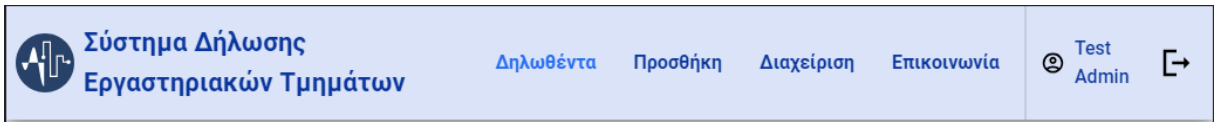
Η δεύτερη HTTPS διαδικασία είναι η κλήση προς το portal του Τμήματος με βάση ενός Time Based Token για απόκτηση της πληροφορίας του χρήστη που προσπαθεί να αυθεντικοποιηθεί. Η επιτυχία της

κλήσης αυτής έχει ως αποτέλεσμα να παρέχει την προσωποποιημένη πληροφορία του χρήστη σε JSON format στο Frontend της εφαρμογής.

Τρίτη και τελευταία διαδικασία είναι το POST Sign-In action που προωθείται στο BackEnd της εφαρμογής. Επισυνάπτονται τα δεδομένα που λήφθηκαν από την επιτυχημένη Δεύτερη διαδικασία έπειτα από μερική επεξεργασία και έπειτα αναλαμβάνει την διεκπεραίωση της αίτησης το ανάλογο τμήμα της Backend εφαρμογής. Η επιτυχημένη Τρίτη διαδικασία αυθεντικοποιεί τον χρήστη προσθέτοντας ένα Browser Session υπο την γνωστή μορφή Browser Cookie. Το Cookie αυτό περιέχει κρυπτογραφημένη πληροφορία με ισχυρό ασύμμετρο αλγόριθμο κρυπτογράφησης, το οποίο από εδώ και στο εξής θα προστίθεται σε κάθε επόμενη διαδικασία που θα ζητηθεί να εξυπηρετηθεί από το ίδιο Backend Application. Σχεδόν όλες οι διαδικασίες που παρέχει και υποστηρίζει το BackEnd είναι προστατευμένες υπο την έννοια ότι εξυπηρετούνται μόνο αν ο χρήστης έχει Αυθεντικοποιηθεί επιτυχώς και όντως είναι αυτός που ισχυρίζεται ότι είναι. Στο Sign-In action επίσης πραγματοποιούνται και ακόμη δύο κρίσιμες για την εφαρμογή διαδικασίες. Η πρώτη είναι η ανάθεση ρόλου στον χρήστη λαμβάνοντας υπόψη το JSON object που παραλαμβάνεται παραμετρικά και δεύτερη η ανάκτηση των Δ.Μ. μόνο αν κατέχει τον Ρόλο του Φοιτητή. Η ύπαρξη της πληροφορίας είναι κρίσιμη για την ορθή λειτουργία της εφαρμογής. Για την επιτυχημένη ανάκτηση της πληροφορίας αυτής γίνεται χρήση ενός Τρίτου API του Τμήματος που εξυπηρετεί μόνο το σερβίρισμα των Διδακτικών Μονάδων ανά Α.Μ. φοιτητή.



Σχήμα 6.2: Κεφαλίδα Ιστοσελίδας Εξουσιοδοτημένου Φοιτητή



Σχήμα 6.3: Κεφαλίδα Ιστοσελίδας Εξουσιοδοτημένου Καθηγητή - Διαχειριστή

Καθώς οι Δ.Μ. για τον φοιτητή είναι τόσο κρίσιμες ως πληροφορία για πολλά σημεία της εφαρμογής , αξίζει να επισημανθεί ότι σε περίπτωση που η Τρίτη εφαρμογή αυτή καταργηθεί ή δεν ανανεωθεί με τις κατα περίπτωση νεότερες για το εξάμηνο Δ.Μ. για κάθε εγγεγραμμένο φοιτητή, δεν υπάρχει καμία διασφάλιση ότι η εφαρμογή που παρουσιάζεται και ο αλγόριθμος θα λειτουργήσει ορθά

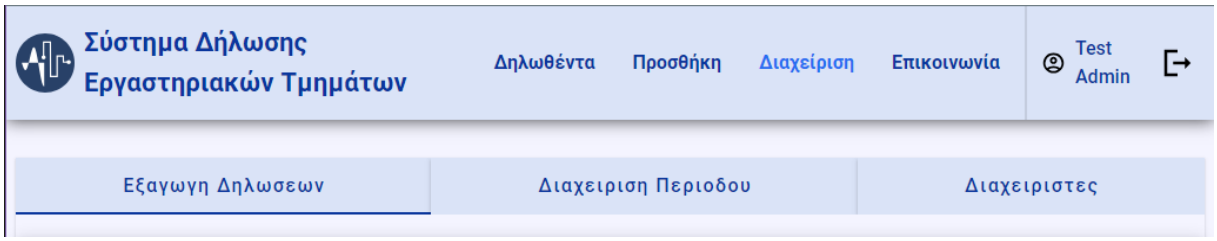
6.3 Πρόσβαση στην Εφαρμογή σαν Διαχειριστής

Στο προηγούμενο κεφάλαιο αναφέρθηκε ότι η εφαρμογή διαθέτει τουλάχιστον έναν διαχειριστή. Ο χρήστης που κατέχει τον ρόλο του Διαχειριστή έχει πρόσβαση στον Πίνακα Διαχείρισης στο Frontend, που εμφανίζεται στην επικεφαλίδα με το όνομα “Διαχείριση”.

Η Επιλογή “Διαχείριση” όμως δεν είναι κλειδωμένη και προσβάσιμη μόνο για τους χρήστες που κατέχουν τον ρόλο του Διαχειριστή. Οι χρήστες με ρόλο Καθηγητή έχουν επίσης πρόσβαση στην συγκεκριμένη ενότητα της ιστοσελίδας, ωστόσο έχουν μειωμένη πρόσβαση στις επιλογές και στις ενέργειες που μπορούν να προβούν από τον Πίνακα Διαχείρισης.

Οι ενέργειες για τους χρήστες με ρόλο Διαχειριστή, που πλοηγούνται στην συγκεκριμένη ενότητα επαυξάνονται και εμφανίζονται δύο επιπλέον επιλογές Διαχείρισης. Αυτές οι δύο επιλογές ονομαστικά

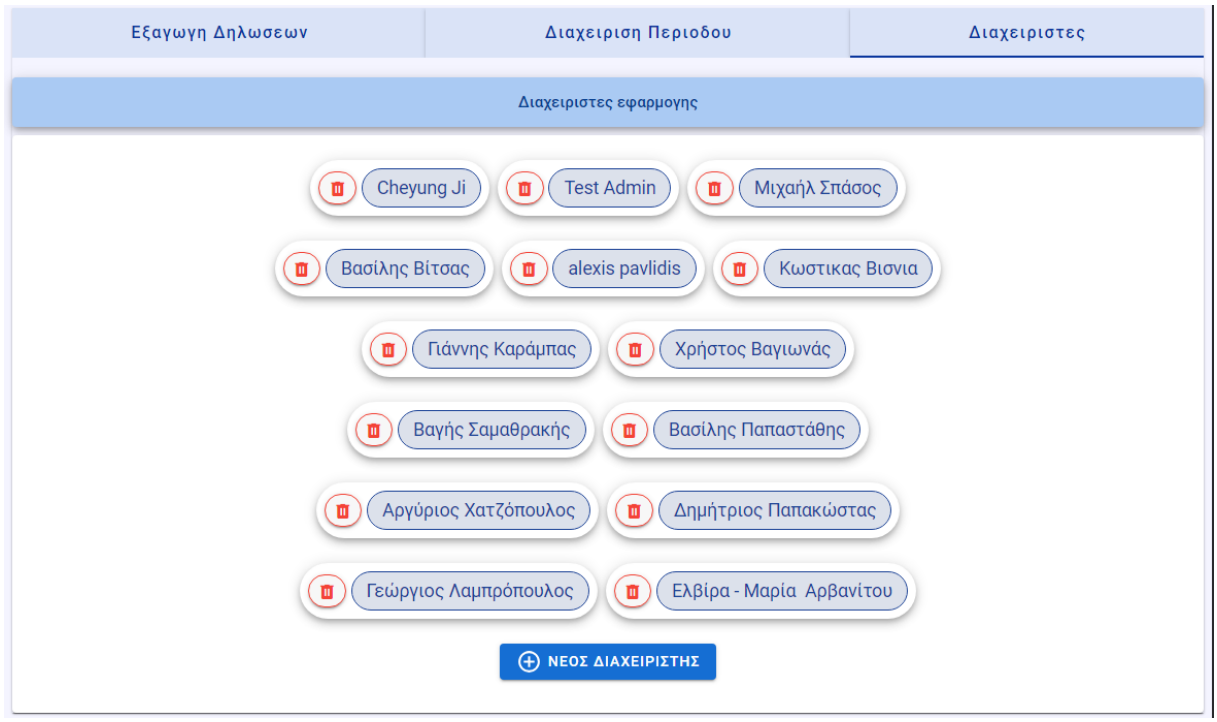
είναι : “Διαχειριστές” και η “Διαχείριση Περιόδου”. Στις ακόλουθες υπο-ενότητες αναλύονται εκτενώς οι λειτουργίες αυτών των δύο επιλογών καθώς η τρίτη και κοινή επιλογή μεταξύ Διαχειριστών και Καθηγητών ονόματι Εξαγωγή Δηλώσεων αναλύεται σε επόμενη υπό ενότητα.



Σχήμα 6.4: Πίνακας Διαχείρισης Καθηγητών - Διαχειριστών

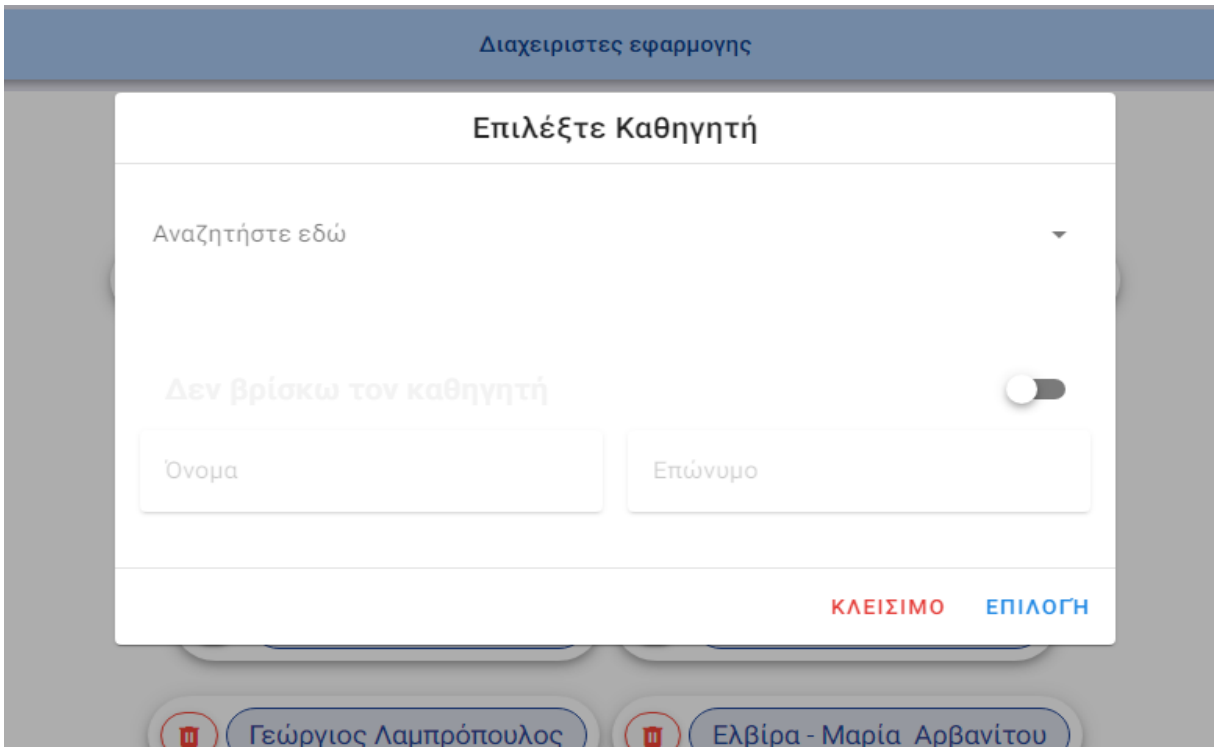
6.3.1 Επιλογή Διαχείρισης Διαχειριστών

Η φιλοσοφία ανάπτυξης της επιλογής αυτής είναι η ευέλικτη παρουσίαση των προσώπων που έχουν δηλωθεί ως διαχειριστές καθώς και ένας εξίσου ευέλικτος τρόπος προσθαφαίρεσης ενός χρήστη που είτε κατέχει ,είτε δεν πρέπει πλέον να κατέχει τον ρόλο του Διαχειριστή.



Σχήμα 6.5: Επιλογή Διαχειριστές Εφαρμογής

Συνεπώς αναπτύχθηκε μια εύκολη μέθοδος για την εισαγωγή ενός καθηγητή ως διαχειριστή μέσω ενός αναπτυσσόμενου μενού διαλόγου (modal menu dialog), όπου εμφανίζονται όλοι οι διαθέσιμοι καθηγητές που έχουν δηλωθεί στη Backend εφαρμογή. Ο διαχειριστής μπορεί να επιλέξει το όνομα του καθηγητή που επιθυμεί να αναβαθμίσει την πρόσβαση και με ένα κλικ να τον θέσει πλέον ως διαχειριστή.



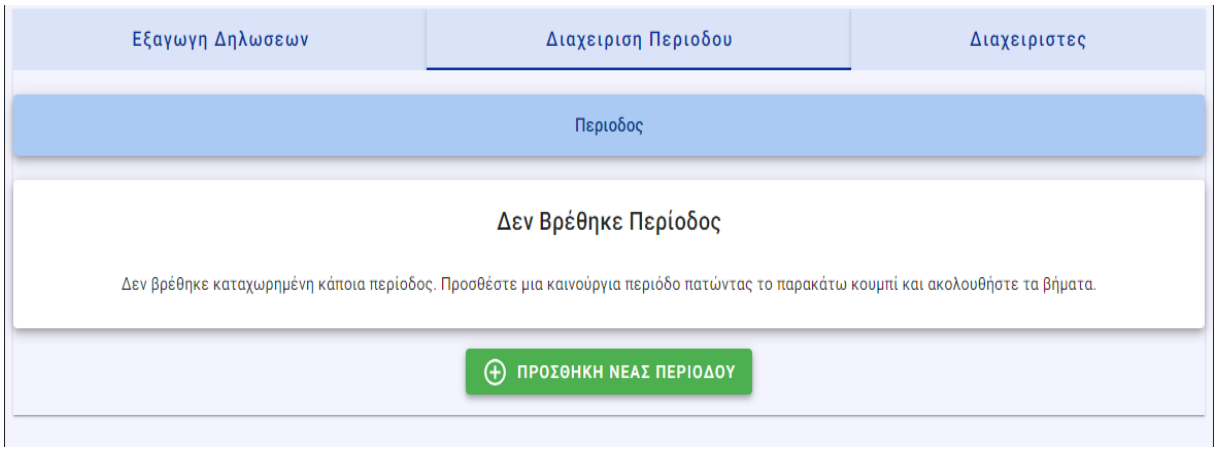
Σχήμα 6.6: Πτυσσόμενο Μενού Διαλόγου Επιλογής

Σε περίπτωση που δεν βρεθεί ο καθηγητής στο μενού, μπορεί ο χρήστης να προβεί στην εισαγωγή του εκ νέου. Η εισαγωγή περιλαμβάνει μόνο το όνομα και το επώνυμο του καθηγητή προς εισαγωγή. Ιδιαίτερα χρήσιμη λειτουργία καθώς εξυπηρετεί περιπτώσεις εκτάκτων καθηγητών που δεν "γνωρίζει" ακόμη το Portal του Τμήματος, όμως έχουν ενεργό ρόλο στην διδασκαλία εργαστηριακών τμημάτων. Μόνη προϋπόθεση ο νέος καθηγητής να μπορεί να αυθεντικοποιηθεί μέσω του Ihu Login, όπως και όλοι οι υπόλοιποι. Ο νέο-εισαχθέντας καθηγητής δηλώνεται αυτόματα ως έκτακτος.

Σε αυτή την ενέργεια της εκ νέου εισαγωγής ενός καθηγητή που δεν γνωρίζει το Portal του τμήματος, υπάρχει επικαιροποίηση του ονόματος και του επωνύμου τόσο απο το FrontEnd όσο και απο το Backend της εφαρμογής. Επιπλέον, ο διαχειριστής έχει τη δυνατότητα να αφαιρέσει άλλους καθηγητές που έχουν ήδη τον ρόλο του Διαχειριστή. Δεν υφίσταται η εφαρμογή ωστόσο να μην έχει τουλάχιστον ένα χρήστη με τον ρόλο του διαχειριστή. Αν γίνει προσπάθεια να διαγραφεί και ο τελευταίος διαχειριστής, εμφανίζεται ανάλογη προτροπή λάθους συστήματος

6.3.2 Επιλογή Διαχείρισης Περιόδου

Ενας διαχειριστής, κατ' απαίτηση, πρέπει να έχει την αποκλειστική δυνατότητα να ξεκινήσει μια περίοδο δηλώσεων, ορίζοντας την ημερομηνία έναρξης και λήξης. Απαίτηση που καλύφθηκε μέσω της διαχειριστικής επιλογής "Διαχείριση Περιόδου". Η ημερομηνία έναρξης αποφασίστηκε ότι πρέπει να είναι τουλάχιστον μία ημέρα μετά την ημέρα δημιουργίας της φόρμας έναρξης δηλώσεων και υπολογισμού προτεραιοτήτων με βάση τον Αλγόριθμο δικαιοσύνης. Επίσης, ο διαχειριστής μπορεί να ορίσει μια ημερομηνία λήξης για την περίοδο δηλώσεων τουλάχιστον μία εβδομάδα αργότερα από την ημερομηνία έναρξης των δηλώσεων. Επιπλέον μπορεί να την ακυρώσει με κλικ στην αντίστοιχη επιλογή.

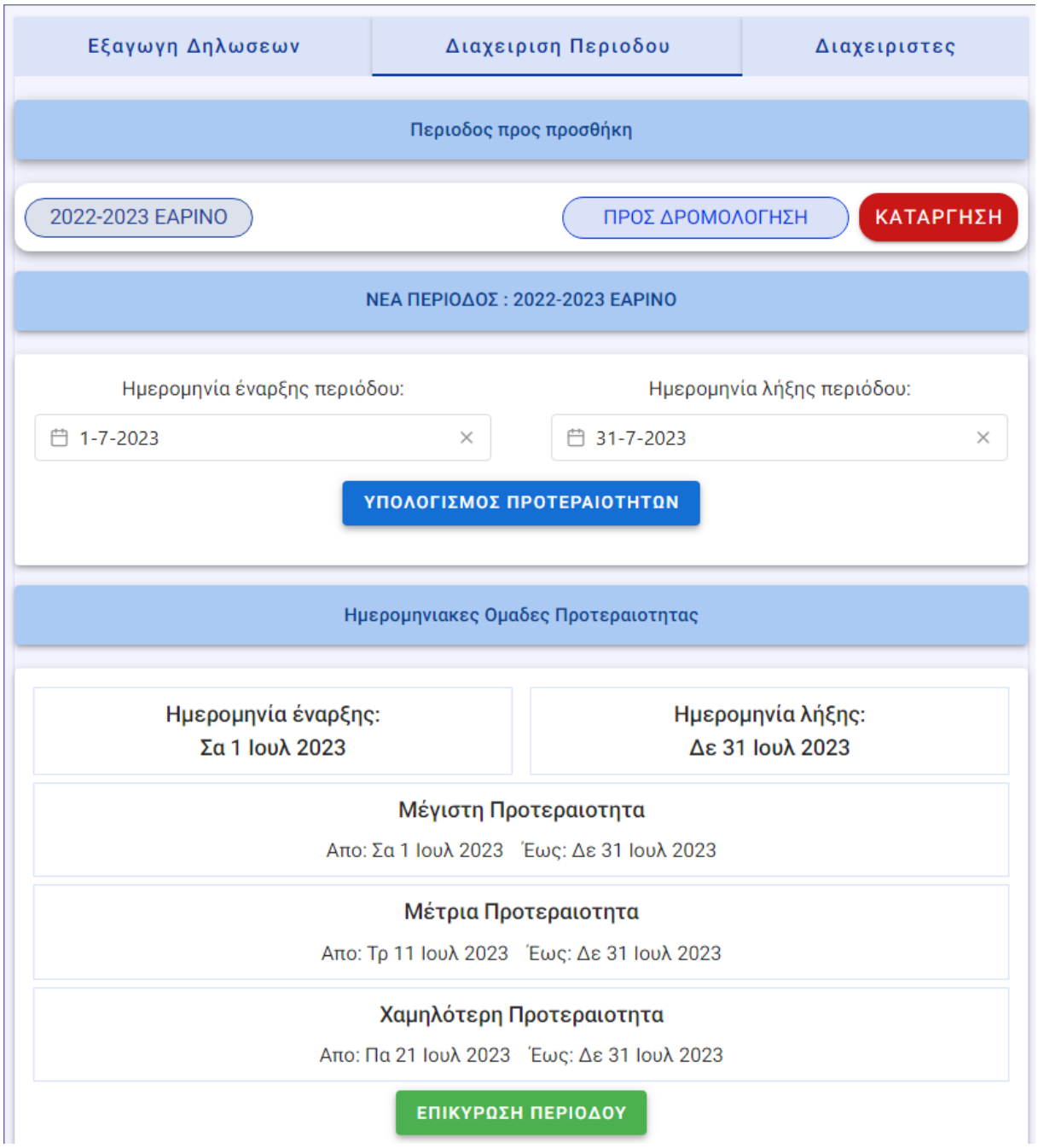


Σχήμα 6.7: Επιλογή Διαχείρισης Περιόδου

Η προαναφερθείσα φόρμα έναρξης είναι βασισμένη σε εισαγωγή ημερομηνιών και συγκεκριμένα ημερομηνίες έναρξης / λήξης από τον χρήστη. Με κλικ στην ενέργεια "Υπολογισμός Προτεραιοτήτων" εμφανίζονται οι υπολογισμένες χρονικές δεσμίδες (time frames) που ο κάθε φοιτητής μπορεί να δηλώσει, ανάλογα με την ομάδα προτεραιότητας που του έχει καταλογιστεί ως αποτέλεσμα απο τον Αλγόριθμο Δικαιοσύνης. Επιγραμματικά αναφέρονται ότι οι ομάδες προτεραιότητας απαρτίζονται απο 3 υποδιαιρέσεις.

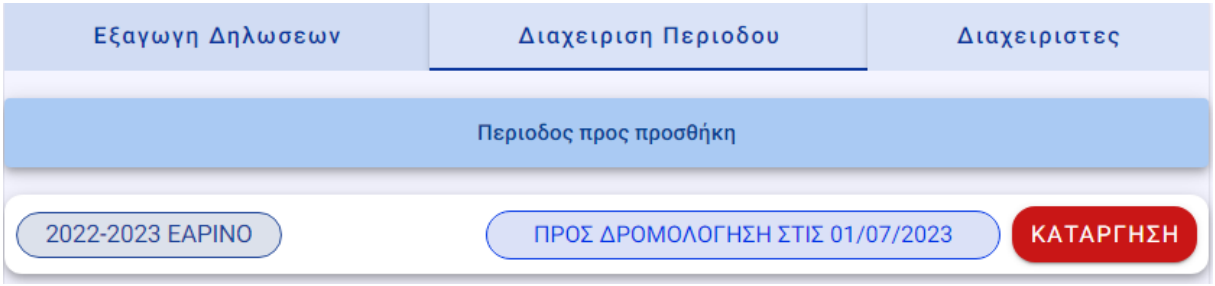
- Ύψιστη προτεραιότητα
- Μέτρια προτεραιότητα
- Χαμηλή προτεραιότητα

Περαιτέρω ανάλυση για τον Αλγόριθμο Δικαιοσύνης και τις ομάδες προτεραιότητας έχει πραγματοποιηθεί σε παραπάνω κεφάλαιο.



Σχήμα 6.8: Φόρμα Δημιουργίας Νέας Περιόδου

Η έναρξη της περιόδου γίνεται αυτοματοποιημένα απο το Backend Application μόλις η τωρινή ημερομηνία ισούται με αυτή που έχει εκχωρηθεί ως η ημερομηνία έναρξης. Η ίδια φιλοσοφία ακολουθείται και για την ημερομηνία λήξης της περιόδου.



Σχήμα 6.9: Νέα Περίοδος Δηλώσεων προς Ενεργοποίηση

6.4 Είσοδος στην Εφαρμογή σαν Καθηγητής

Όταν ένας καθηγητής συνδέεται στην εφαρμογή, οι λειτουργίες του μπορούν να χωριστούν σε δύο υποκατηγορίες: αυτές που μπορεί να εκτελέσει πριν την έναρξη της περιόδου δηλώσεων και αυτές που μπορεί να εκτελέσει κατά τη διάρκεια ή μετά το πέρας της περιόδου δηλώσεων.

Πριν την έναρξη της περιόδου, ένας καθηγητής μπορεί να πραγματοποιήσει τις παρακάτω ενέργειες:

- Προβολή πληροφοριών προηγούμενων δηλώσεων: Ο καθηγητής μέσω του Πίνακα Διαχείρισης που αποσαφηνίστηκε προηγουμένως ότι έχει πρόσβαση, μπορεί να δει τα στοιχεία των μαθημάτων του απο την προηγούμενη περίοδο δηλώσεων που πλέον είναι ανενεργή αν και μόνο δεν έχει διαγραφεί από κάποιον Διαχειριστή της Εφαρμογής η εν λόγω προηγούμενη περίοδος δηλώσεων και οι πληροφορίες που κουβαλάει είναι ακόμη διαθέσιμες προς λήψη
- Προετοιμασία για τις δηλώσεις: Ο καθηγητής μπορεί να προσθέσει τα μαθήματα που θα διδάξει πριν την περίοδο έναρξης αν δεν το έχει κάνει ήδη ή να ενημερώσει / τροποποιήσει τα υπάρχοντα. Άρα απαιτείται να ετοιμάσει τις λεπτομέρειες των μαθημάτων που πρόκειται να διδάξει κατά την τρέχουσα περίοδο όπως:
- Κωδικοί Τμημάτων,
- Περιγραφές,
- Τίτλοι,
- Διδάσκοντες καθηγητές,
- Ειδικές λεπτομέρειες για το κάθε εργαστηριακό τμήμα κ.α.

Κατά τη διάρκεια ή μετά το πέρας των δηλώσεων, ο καθηγητής μπορεί να εκτελέσει τις εξής ενέργειες:

Πρόσβαση στο σύστημα παρακολούθησης για τα μαθήματα του. Υπάρχει δηλαδή η δυνατότητα αυτός ο καθηγητής να βλέπει την ώρα των δηλώσεων το ποσοστό πληρότητας ενός τμήματος και εν γένη να το τροποποιήσει .

Εξαγωγή των μέχρι στιγμής αποτελεσμάτων των δηλώσεων. Άρα υπάρχει η δυνατότητα ο καθηγητής μέσω του Πίνακα Διαχείρισης να κατεβάσει τα στοιχεία ανά μάθημα για την τρέχουσα κατάσταση της συγκεκριμένης περιόδου δηλώσεων.

Φιλτράρισμα του αρχείου εξαγωγής με βάση την εισαγωγή ενός αντίστοιχου αρχείου που υποδηλώνει όλους τους φοιτητές που έχουν δηλώσει την θεωρία του μαθήματος αν και μόνο αν αυτές έχουν προηγηθεί.

Και τέλος πρόσβαση στο μενού επισκόπησης των μαθημάτων του. Στην επισκόπηση αυτή του παρουσιάζονται και οι επιλογές τροποποίησης και διαγραφής ενός συγκεκριμένου τμήματος.

6.4.1 Προσθήκη Μαθήματος

Στην κατηγορία πριν την έναρξη των δηλώσεων, ένα από τα δικαιώματα που έχει ο καθηγητής στην εφαρμογή είναι η δημιουργία ενός Εργαστηριακού Μαθήματος. Το κάθε μάθημα θεωρείται μοναδικό ως οντότητα και μπορεί να έχει περισσότερα από ένα εργαστηριακό τμήμα, το καθένα από τα οποία διαθέτει μία ώρα διεξαγωγής το εξάμηνο. Το σχεσιακό σχήμα που εξυπηρετεί την λειτουργικότητα αυτή αναλύεται σε παρακάτω υπό ενότητα.

Οι πληροφορίες που περιλαμβάνονται σε ένα Εργαστηριακό Μάθημα ως οντότητα είναι οι εξής:

- Εξάμηνο Διεξαγωγής Μαθήματος.
- Κωδικός Εργαστηριακού Μαθήματος.
- Τίτλος Εργαστηριακού Μαθήματος.
- Σύντομη Περιγραφή Μαθήματος.
- Παρακολούθηση Μαθήματος

Αυτές οι πληροφορίες εμφανίζονται στην αρχική καρτέλα του μαθήματος, παρέχοντας χρήσιμες πληροφορίες στους φοιτητές και τους καθηγητές για κάθε μάθημα.

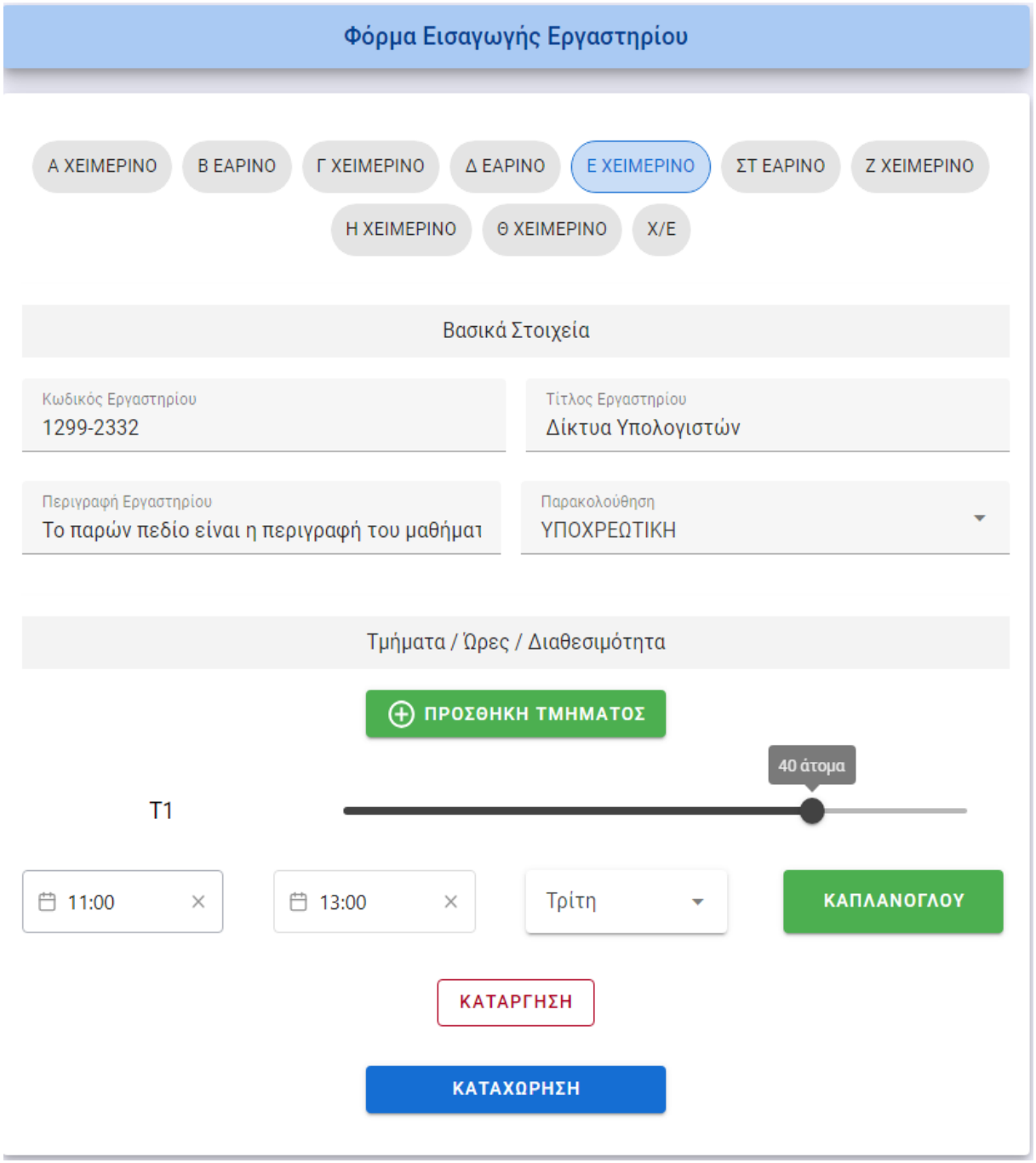
Ο χρήστης όπου κάνει την πρωταρχική προσθήκη του εργαστηριακού μαθήματος στο σύστημα θεωρείται ο Βασικός συντονιστής του μαθήματος αυτού.

Στο επόμενο βήμα της δημιουργίας του εργαστηριακού μαθήματος, ο χρήστης υποχρεούται να εισάγει τουλάχιστον ένα εργαστήριο τμήμα για να θεωρηθεί έγκυρη η φόρμα εισαγωγής του μαθήματος και να μπορεί να προχωρήσει στην τελική προσθήκη του στο σύστημα. Η προσθήκη ενός τέτοιου εργαστηριακού τμήματος γίνεται με κλικ στην επιλογή “Προσθήκη Τμήματος”.

Κάθε εργαστηριακό τμήμα που προστίθεται θα πρέπει να έχει συμπληρωμένες τις παρακάτω τιμές ώστε να θεωρηθεί έγκυρο. Τα πεδία του τμήματος είναι τα εξής:

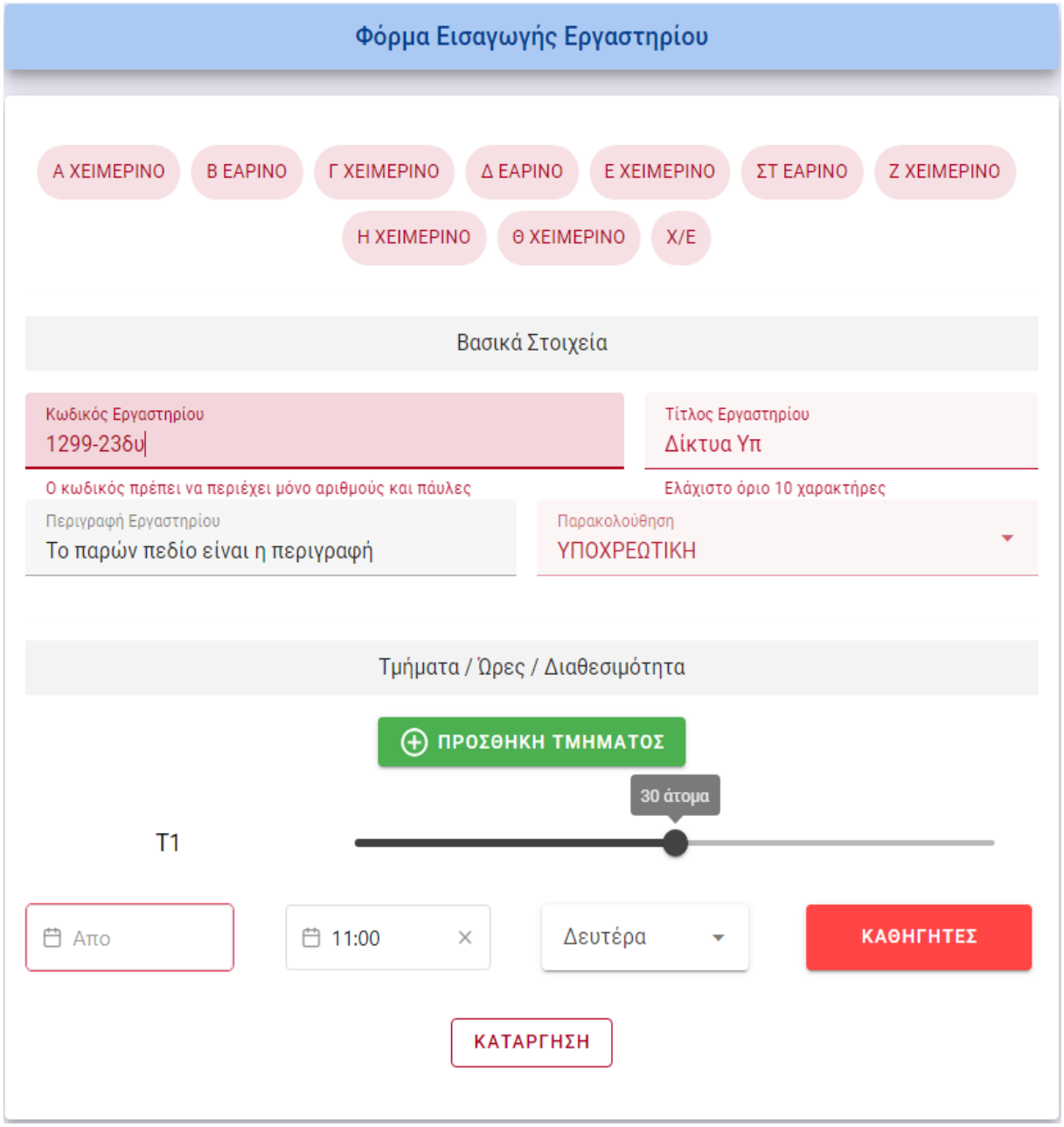
- Το Εργαστηριακό Όνομα (π.χ. Τ1).
- Η Ημέρα Διεξαγωγής του Εργαστηρίου.
- Η Ώρα Έναρξης Διεξαγωγής του Εργαστηρίου.
- Η Ώρα Λήξης Διεξαγωγής του Εργαστηρίου.
- Ο Διδάσκων Καθηγητής.
- Ο Μέγιστος Αριθμός Ατόμων Παρακολούθησης του Τμήματος.

Ο συντονιστής είναι υποχρεωμένος να εισάγει τον διδάσκων καθηγητή, **ακόμα και στην περίπτωση που είναι ο ίδιος ο γενικός συντονιστής του Εργαστηριακού Μαθήματος ,** επιπλέον σημειώνεται ότι ο καθηγητής που διδάσκει το εργαστηριακό τμήμα μπορεί να είναι διαφορετικός από αυτόν που είναι υπεύθυνος για το εργαστηριακό μάθημα στο σύνολο του, ώστε γίνει η ανάλογη εγγραφή του στο σύστημα. Εάν ο επιθυμητός καθηγητής δεν υπάρχει στο σύστημα, μπορεί να προστεθεί με το ονοματεπώνυμό του. Κατα συνέπεια πραγματοποιείται μια νέα εγγραφή χρήστη ως καθηγητή στη βάση δεδομένων, επιτρέποντας τη δήλωσή του ως διδάσκων του τμήματος κανονικά. Επίσης αναφέρεται ότι, μπορεί να δηλωθεί ως διδάσκων σε ένα εργαστηριακό τμήμα ενας χρήστης με ρόλο καθηγητή όπου είναι επίσης γενικός συντονιστής σε ένα διαφορετικό Εργαστηριακό Μάθημα χωρίς καμία παραλλαγή. Κατά την ολοκλήρωση όλων των υποχρεωτικών πεδίων, γίνεται η καταχώρηση του μαθήματος και εμφανίζεται ανάλογο μήνυμα επιτυχίας.



Σχήμα 6.10: Έγκυρη Φόρμα Εισαγωγής Μαθήματος

Εάν κάποιο από τα υποχρεωτικά πεδία δεν συμπληρωθεί, τότε η δημιουργία δεν μπορεί να προχωρήσει μέχρι να συμπληρωθούν ορθώς όλα τα λανθασμένα πεδία. Σε περίπτωση που ο χρήστης δεν επιθυμεί να δημιουργήσει το μάθημα μπορεί να μεταβεί σε άλλη σελίδα της εφαρμογής, όπου και πριν από αυτή την ενέργεια θα λάβει ένα μήνυμα επιβεβαίωσης ότι όλες οι πληροφορίες που έχει εισάγει στη φόρμα μέχρι στιγμής θα διαγραφούν.



Σχήμα 6.11: Άκυρη Φόρμα Εισαγωγής Μαθήματος

Σημαντική λεπτομέρεια είναι ότι κατά τη διάρκεια της περιόδου δήλωσης, ένας καθηγητής δεν μπορεί να δημιουργήσει νέο μάθημα. Η δημιουργία νέου μαθήματος πρέπει να γίνει τουλάχιστον μία ημέρα πριν την έναρξή της περιόδου.

6.4.2 Δηλωθέντα Μαθήματα

Σε αυτή την ενότητα της εφαρμογής, ο καθηγητής έχει πρόσβαση κατά τη διάρκεια της περιόδου δηλώσεων όσο και μετά τη λήξη της. Μπορεί να διαχειριστεί όλα τα μαθήματα που έχει δημιουργήσει και να παρακολουθεί την πορεία των εργαστηριακών τμημάτων που του έχουν ανατεθεί από άλλους συντονιστές. Η κύρια λειτουργία αυτής της σελίδας είναι να παρέχει στον καθηγητή μια ολοκληρωμένη επισκόπηση και έλεγχο των εργαστηριακών μαθημάτων και εργαστηριακών τμημάτων αντίστοιχα.

Σχήμα 6.12: Τα Δηλωθέντα Εργαστηριακά Τμήματα

Συγκεκριμένα, ο συντονιστής καθηγητής έχει πρόσβαση σε κρίσιμες και χρήσιμες πληροφορίες για κάθε εργαστηριακό τμήμα, που τον βοηθούν να το αναγνωρίσει και να το διαχειριστεί αναλόγως. Οι πληροφορίες παρουσιάζονται ταξινομημένες ανά εξάμηνο με ευέλικτο τρόπο και είναι οι εξής:

- Κωδικός Εργαστηριακού Μαθήματος.
- Τίτλος Μαθήματος.
- Ώρα Έναρξης και Λήξης Τμήματος.
- Ημέρα διεξαγωγής.
- Εξάμηνο Μαθήματος.
- Συμμετοχή στο Μάθημα.

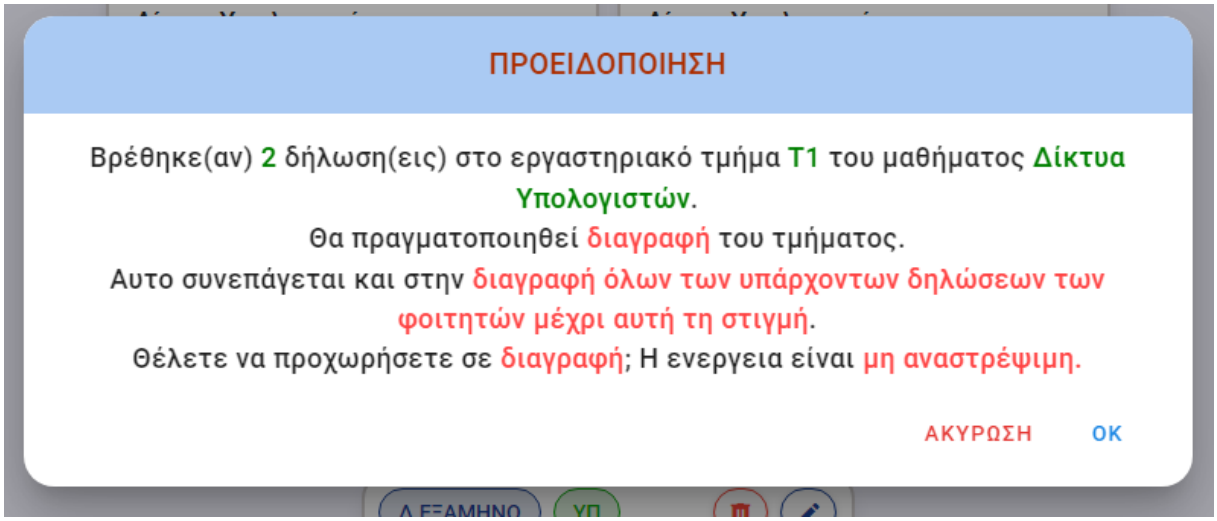
Επιπλέον, ένας καθηγητής έχει πρόσβαση σε δύο κρίσιμες λειτουργίες (διαγραφή, τροποποίηση) που μπορεί να εκτελέσει πάνω στο μάθημα συνολικά, **μόνο και μόνο αν είναι ο κύριος συντονιστής του εργαστηριακού μαθήματος και όχι απλώς ένας διδάσκων σε ένα εργαστηριακό τμήμα.** Η παρούσα απόφαση λήφθηκε ως μέτρο προστασίας.

6.4.3 Διαγράφη Μαθημάτων

Στην παρούσα ενότητα της εφαρμογής, Ο καθηγητής μπορεί εάν το επιθυμεί να διαγράψει εξ ολοκλήρου ένα ολόκληρο εργαστηριακό τμήμα από την εννοιολογική οντότητα του εργαστηριακού μαθήματος. Ωστόσο, είναι σημαντικό να σημειωθεί πρωταρχικά ότι αυτή η ενέργεια είναι μη αναστρέψιμη και

οδηγεί σε μόνιμη απώλεια των σχετικών δεδομένων. Όλες οι πληροφορίες που αφορούν στο επιλεγμένο μάθημα χάνονται οριστικά.

Σε περίπτωση που χρειαστεί να γίνει κατάργηση ενος τμήματος, εν μέσω ενεργής περιόδου δηλώσεων, υπάρχει ανάλογη προτροπή που ενημερώνει κατάλληλα αν υπάρχουν μέχρι στιγμής συνδεδεμένες δηλώσεις φοιτητών πάνω σε αυτό το εργαστηριακό τμήμα που είναι προς κατάργηση, όπως επίσης παρουσιάζονται και οι ενέργειες επικύρωσης είτε ακύρωσης της ενέργειας που επρόκειτο να εκτελεστούν. Μόλις ο καθηγητής επιλέξει μία από τις δύο επιλογές, "ΟΚ" ή "Ακύρωση", θα εκτελεστεί η αντίστοιχη ενέργεια.



Σχήμα 6.13: Προειδοποίηση Διαγραφής

Σε περίπτωση που πατηθεί το "ΟΚ", θα πραγματοποιηθεί ολική διαγραφή του τμήματος, καθώς και όλων των σχετικών δηλώσεων που είναι συνδεδεμένο, εάν αυτές φυσικά υπάρχουν. Σε περίπτωση που υπάρχουν δηλώσεις για το εν λόγω μάθημα, θα εμφανιστεί ένα μήνυμα ενημέρωσης, επισημαίνοντας ότι το συγκεκριμένο μάθημα έχει δηλωθεί από τουλάχιστον έναν φοιτητή, και οι σχετικές δηλώσεις θα διαγραφούν επίσης. Να σημειωθεί ισχυρά σε αυτό το σημείο ότι η εν λόγω ενέργεια μόλις ολοκληρωθεί είναι επίσης μη αναστρέψιμη.

Αντίθετα, εάν ο καθηγητής επιλέξει την επιλογή "Ακύρωση", καμία ενέργεια δεν θα πραγματοποιηθεί, το μάθημα θα παραμείνει ανέπαφο και η πλοήγηση στην εφαρμογή συνεχίζει κανονικά.

Οι παραπάνω συνδυαστικές λειτουργίες και προτροπές παρέχουν στον καθηγητή έναν αποτελεσματικό και ασφαλή τρόπο διαχείρισης των μαθημάτων του με μεγαλύτερη έμφαση στην αποφυγή κρίσιμων λαθών που προκαλούν σύγχυση.

Η λειτουργία της διαγραφής επιτρέπει στον καθηγητή να διαχειρίζεται τα μαθήματά του με ευελιξία, προσφέροντας τη δυνατότητα διαγραφής πληροφορίας όταν και αν το θεωρήσει απαραίτητο, με την επιφύλαξη ότι οποιαδήποτε απώλεια δεδομένων είναι μη αναστρέψιμη.

6.4.4 Τροποποίηση Μαθημάτων

Στην ενότητα της Τροποποίησης, που είναι διαθέσιμη είτε είναι ενεργή μια περίοδος δηλώσεων είτε όχι, ο καθηγητής έχει την ελευθερία να αλλάξει οποιαδήποτε πληροφορία θέλει για ένα υπάρχων Εργαστηριακό μάθημα. Μπορεί να ενημερώσει όλα τα πεδία, υπό την προϋπόθεση ότι δεν αφήνει κενά ή πληροφορίες που δεν αντιστοιχούν σε συγκεκριμένα πεδία.

Ωστόσο, υπάρχουν ορισμένοι περιορισμοί που τέθηκαν κατ' απαίτηση. Δεν μπορεί να μειώσει τον ελάχιστο αριθμό ατόμων που μπορούν να δηλώσουν ένα εργαστηριακό τμήμα, ωστόσο, επιτρέπεται η αύξηση του αριθμού αυτού μέχρι μία συγκεκριμένη τιμή. Η εν λόγω τιμή εκκινεί αυτόματα από τον ήδη δηλωμένο μέγιστο αριθμό που ορίστηκε κατά τη δημιουργία του εργαστηρίου και μπορεί να αυξηθεί μέχρι το μέγιστο όριο των πενήντα (50) ατόμων ανά τμήμα.

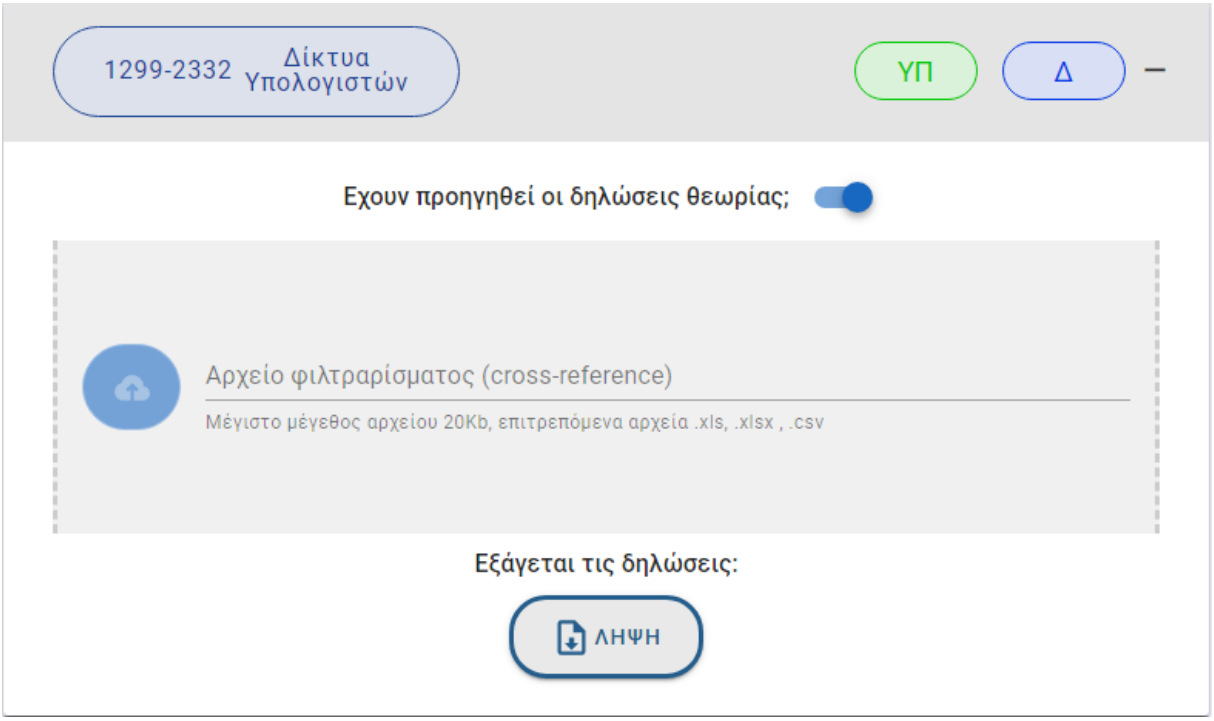
Όταν ο καθηγητής ολοκληρώσει τις επιθυμητές αλλαγές, θα πρέπει να πατήσει το κουμπί "Ενημέρωση" ώστε να εφαρμοστούν όλες οι αλλαγές σε όλα τα πεδία του μαθήματος. Αν η ενέργεια της ενημέρωσης είναι επιτυχημένη θα παρουσιαστεί το ανάλογο μήνυμα επιτυχίας. Το ίδιο θα συμβεί επίσης αν προκύψει και κάποιο σφάλμα στην εφαρμογή κατα την ενημέρωση του μαθήματος. Αν ωστόσο, γίνει αλλαγή σελίδας ενώ δεν έχουν ενημερωθεί οι αλλαγές αυτές, θα εμφανιστεί επιλογή που ενημερώνει ότι όλες οι μη-ενημερωμένες αλλαγές θα χαθούν. Μέσω κλικ στην "ΟΚ" επιλογή πραγματοποιείται η ακύρωση και η μεταφορά σε άλλη σελίδα της εφαρμογής, ενώ με κλικ στην επιλογή "Ακύρωση" ακυρώνονται όλες οι ενέργειες πλοήγησης και παραμένει στην σελίδα της τροποποίησης.

Επιπλέον, ο καθηγητής μπορεί να καταργήσει ένα εργαστηριακό τμήμα μέσα από την φόρμα τροποποίησης. Αυτή η επιλογή ακολουθεί η ίδια διαδικασία με αυτή που αναφέρθηκε στην ενότητα της Διαγραφής Μαθημάτων. Γίνεται εμφάνιση της επιλογή "Κατάργηση" που βρίσκεται δίπλα από κάθε εργαστηριακό τμήμα. Ας σημειωθεί ότι οι δύο παραπάνω ενέργειες μπορεί να λειτουργήσουν συνδυαστικά χωρίς να παρεμποδίζει η μία την άλλη.

6.4.5 Λειτουργία Εισαγωγής / Εξαγωγής Αρχείων

Ένας χρήστης με τον ρόλο του καθηγητή έχει στη διάθεσή του τη λειτουργία Εισαγωγής και Εξαγωγής δεδομένων ανά εργαστηριακό μάθημα, προκειμένου να διατηρεί μια ακριβή και ενημερωμένη λίστα που θα του φανεί χρήσιμη. Οι λειτουργίες παρουσιάζονται κάτω από την επιλογή "Διαχείριση". Με τη χρήση αυτών των λειτουργιών, ο καθηγητής έχει τη δυνατότητα να διατηρεί μια ενημερωμένη και ακριβή λίστα για κάθε εργαστηριακό μάθημα του. Αυτό επιτρέπει την ευκολότερη παρακολούθηση και διαχείριση των δραστηριοτήτων των εργαστηρίων, προσφέροντας ένα αξιόπιστο εργαλείο για τον καθηγητή και βελτιώνοντας την επικοινωνία και τη συνεργασία μεταξύ των εμπλεκομένων φοιτητών και καθηγητών.

Εισαγωγή Δεδομένων μέσω αρχείων: Η Λειτουργία του κεφαλαίου αυτού στεγάζεται κάτω από την επιλογή "Εξαγωγή Δηλώσεων" που είναι μέρος της ενότητας "Διαχείρισης". Ο ενδιαφερόμενος μπορεί να επιλέξει το εργαστηριακό μάθημα στο οποίο θέλει να ανεβάσει αρχείο και με την ενεργοποίηση του διακόπτη "Έχουν προηγηθεί οι δηλώσεις θεωρίας;" να του εμφανιστεί το κατάλληλο τμήμα της ιστοσελίδας που είναι υπεύθυνο για αυτή την λειτουργία.



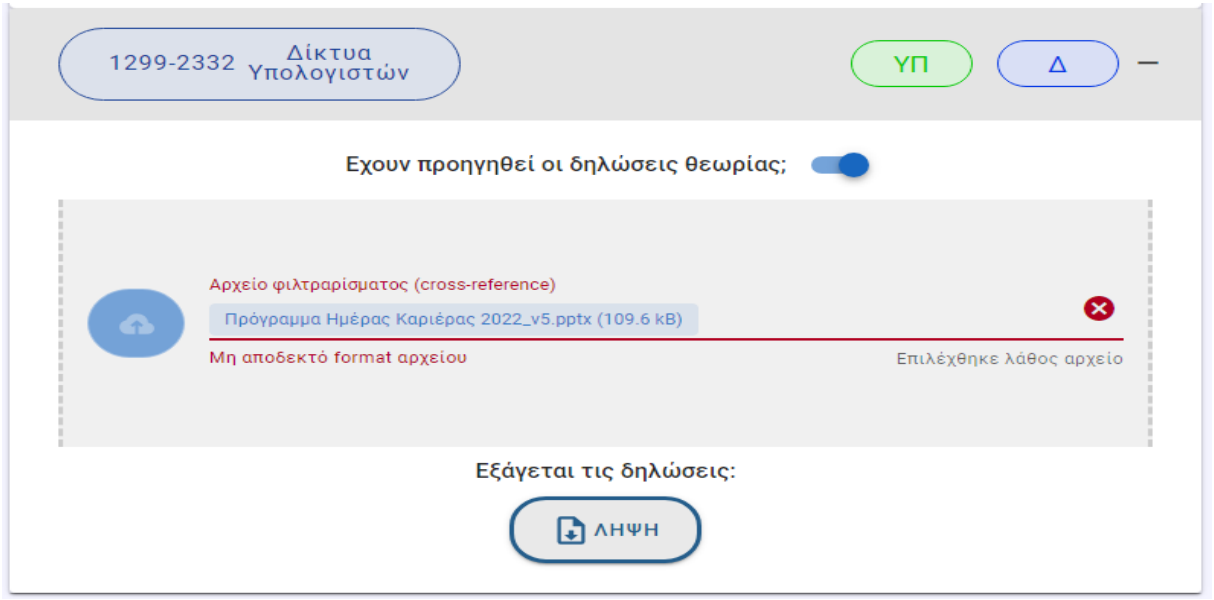
Σχήμα 6.14: Εμφάνιση επιλογής μεταφόρτωσης Αρχείου

Μέσω αυτής της λειτουργίας, ο καθηγητής μπορεί να εισάγει νέα δεδομένα για κάθε εργαστηριακό τμήμα που διδάσκει υπο την μορφή αρχείου .csv, .xlx, .xlsx. Η πληροφορία που επιβάλλεται να εισαχθεί είναι ο Αριθμός Μητρώου των φοιτητών, όπου η ύπαρξη του εκάστοτε ΑΜ στο αρχείο αυτό υποδηλώνει ότι έχει πραγματοποιηθεί η δήλωση του συγκεκριμένου φοιτητή ενός φοιτητή στο θεωρητικό μέρος του μαθήματος. Όταν ο καθηγητής εισάγει το αρχείο που περιέχει την απαιτούμενη στήλη απο ΑΜ, αυτόματα εκτελείται μια λειτουργία που δημιουργεί έναν κωδικό μαθήματος. Ο κωδικός αυτός αντλεί όλες τις πληροφορίες που είναι γνωστές για τις συγκεκριμένες δηλώσεις και ενημερώνει μια Boolean μεταβλητή με την ονομασία "SubmittedTheory" στο έγγραφο δήλωσης του χρήστη για το συγκεκριμένο εργαστηριακό μάθημα.

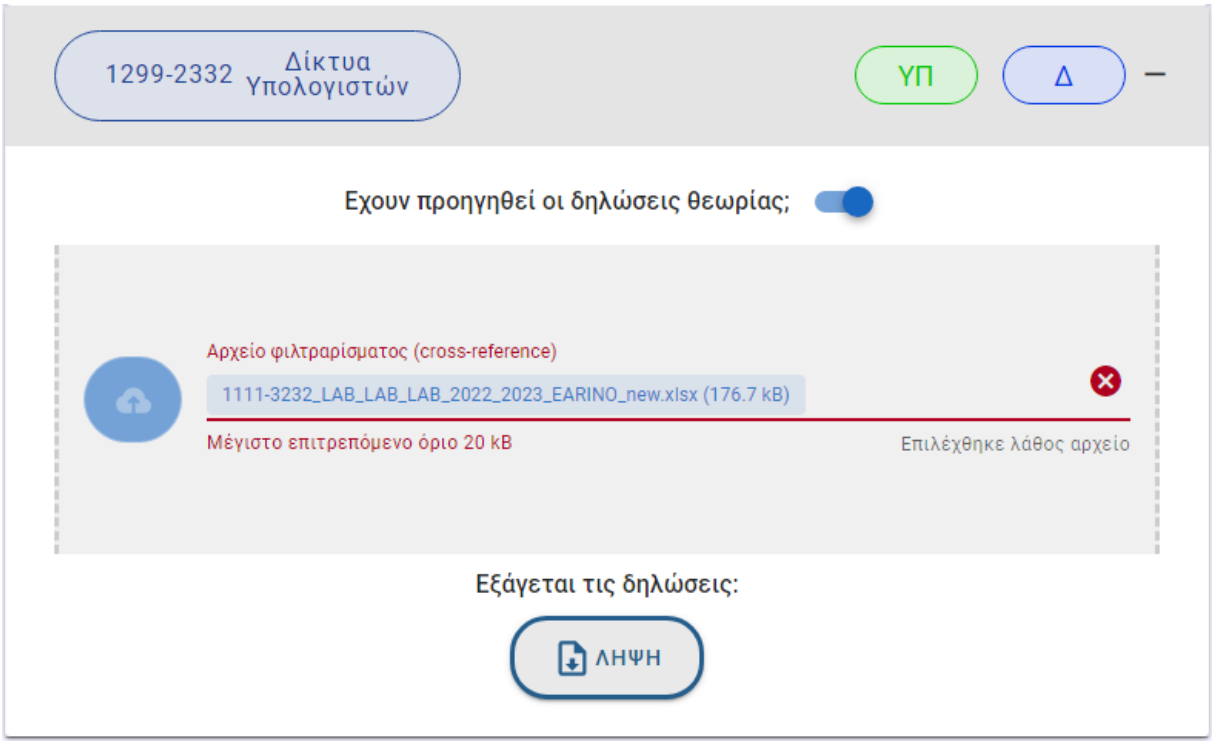
Σε αυτό το σημείο αξίζει να σημειωθεί ότι η μεταφόρτωση ενός αρχείου στην BackEnd εφαρμογή που σκοπό έχει το φιλτράρισμα / ανανέωση των εγγραφών των δηλώσεων γίνεται υπό όρους. Υπάρχουν κανόνες επικύρωσης οι οποίοι παρουσιάζονται με ανάλογο μήνυμα λάθους τόσο στο FrontEnd όσο και στο BackEnd της εφαρμογής. Σε περίπτωση που δεν τηρούνται δεν είναι δυνατή η μεταφόρτωση. Αυτοί οι κανόνες έχουν εφαρμοστεί με σκοπό την καλύτερη και ασφαλέστερη μεταφόρτωση αρχείων στην εφαρμογή ώστε να μην παρουσιαστούν κενά ασφαλείας.

Οι κανόνες που δεν επιτρέπουν την μεταφόρτωση και ακυρώνουν την μεταφόρτωση αρχείου είναι οι εξής:

- Το μέγιστο μέγεθος του προς ανάρτηση αρχείου είναι <= 20kB.
- Το πεδίο πρέπει να είναι συμπληρωμένο με υπαρκτό αρχείο.
- Οι επιτρεπόμενες καταλήξεις αρχείων είναι αυστηρά .xlsx, .xls, .csv.
- Το Mime Type του αρχείου πρέπει να συμπεριλαμβάνεται στην λίστα των MimeTypes που παρατίθενται στο Παράρτημα Α.



Σχήμα 6.15: Ενημέρωση μη αποδεκτής κατάληξης / mime-type Αρχείου προς μεταφόρτωση



Σχήμα 6.16: Ενημέρωση μη αποδεκτού μεγέθους Αρχείου προς μεταφόρτωση

Σε περίπτωση που κανένας από τους παραπάνω κανόνες δεν παραβιάζεται δίνεται η επιλογή μεταφόρτωσης του αρχείου στο BackEnd της εφαρμογής μέσω κλικ στο ενεργοποιημένο πλέον κουμπί του Upload που βρίσκεται δίπλα από το πεδίο εισαγωγής του αρχείου

Η λογική αναπτύχθηκε έτσι ώστε να διαβάζονται οι αριθμοί μητρώου των φοιτητών μέσω της στήλης που βρίσκεται στο αρχείο. Έπειτα, **αν υπάρχει** η εγγραφή της δήλωσης του φοιτητή με το αντίστοιχο ΑΜ στη βάση δεδομένων, αλλάζει η τιμή της μεταβλητής "SubmittedTheory" σε True. Αυτό σημαίνει ότι ο συγκεκριμένος φοιτητής έχει δηλώσει επιτυχώς το μάθημα θεωρίας και έχει δικαίωμα παρακολούθησης του εργαστηρίου. Σε περίπτωση που ο αριθμός μητρώου που υπάρχει στο αρχείο δεν

αντιστοιχεί σε κάποια εγγραφή δήλωσης σε εργαστηριακό μάθημα, δεν γίνεται καμία άλλη λειτουργία. Κατά την δημιουργία της αρχικής εγγραφής δήλωσης για έναν φοιτητή όταν αυτή πραγματοποιηθεί, ας σημειωθεί ότι η μεταβλητή "SubmittedTheory" που ελέγχεται σε αυτή την λειτουργία αρχικοποιείται ως False. Δηλαδή, όλοι οι φοιτητές που προβαίνουν σε δήλωση μέσω της εφαρμογής αντιμετωπίζονται ως οντότητες που δεν έχουν υποβάλει πρωτίστως δήλωση θεωρίας.

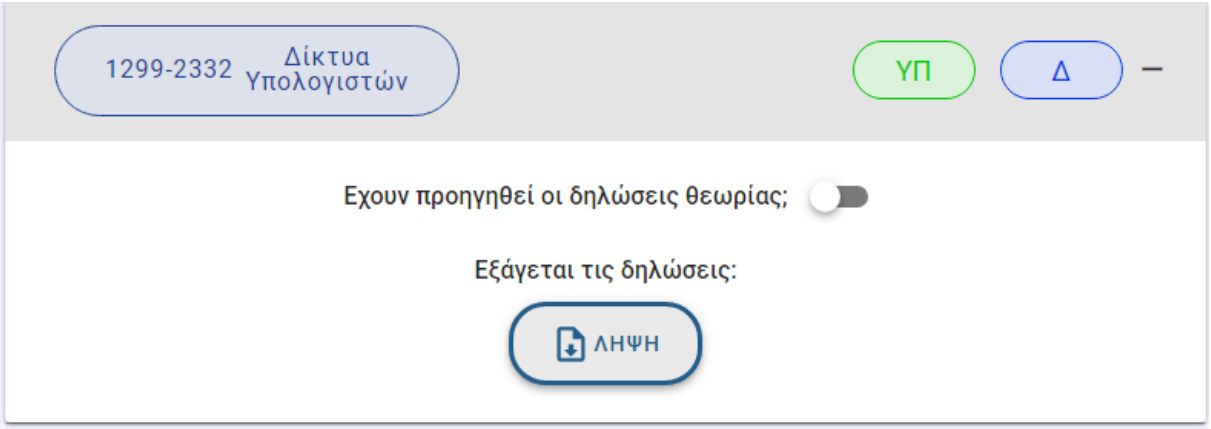
Ωστόσο, η παραπάνω διαδικασία δεν είναι δεσμευτικό οτι εκτελείται μόνο μια φορά κατα την λήξη της περιόδου δηλώσεων. Υπάρχει η δυνατότητα να εκτελείται περιοδικά, με αποτέλεσμα να ενημερώνεται επίσης περιοδικά η βάση δεδομένων με τις δηλώσεις θεωρίας των φοιτητών και κατα συνέπεια το αρχείο εξαγωγής που αναφέρεται στην παρακάτω ενότητα.

Με αυτόν τον τρόπο, παράγεται κατα περίπτωση μια ενημερωμένη λίστα με όσους φοιτητές έχουν δηλώσει τόσο το μάθημα θεωρίας συνδυαστικά με το εργαστήριο, όσο και αυτούς που δεν έχουν δηλώσει το μάθημα θεωρίας και επιθυμούν να καλύψουν μια θέση σε κάποιο εργαστηριακό τμήμα.

Εν κατακλείδι, η προαναφερθείσα λειτουργία σχεδιάστηκε με τέτοιο τρόπο ώστε να είναι ευέλικτη στη χρήσης της και μέσω συνδυαστικής πληροφορίας από άλλες πηγές ώστε να εξάγεται πάντα η ορθή τελική πληροφορία για κάθε εργαστηριακό τμήμα

<u>Εξαγωγή Δεδομένων σε αρχείο</u>: Αρχικά σημειώνεται πως η υλοποίηση της λειτουργίας εξαγωγής εξυπηρετείται απο το Backend της εφαρμογής. Το BackEnd μορφοποιεί την πληροφορία κατάλληλα και την σερβίρει μέσω του πρωτοκόλλου HTTPS στο Frontend της εφαρμογής ώς bytes με κατάληξη .csv. Έπειτα η Frontend εφαρμογή με χρήση Javascript βιβλιοθήκης, αναλαμβάνει την μετατροπή του .csv που αποστέλλεται σε .xlsx αρχείο και τέλος αποθήκευση του, μετά απο προτροπή, στο σύστημα του χρήστη της εφαρμογής.

Στην περίπτωση που ο χρήστης με Ρόλο Καθηγητή είτε με επαυξημένο ρόλο Διαχειριστή επιθυμεί να κάνει εξαγωγή των δηλώσεων ώστε να έχει την εικόνα του πως βαίνουν οι δηλώσεις μέχρι στιγμής με το πόσοι φοιτητές και ποιοι έχουν κάνει δήλωση για ένα συγκεκριμένο εργαστηριακό μάθημα, υπάρχει η δυνατότητα στην ανάλογη ενότητα του μενού “Διαχείρισης” στην επιλογή “Εξαγωγή Δηλώσεων” να πραγματοποιηθεί κλικ στην επιλογή “Λήψη” του κάθε εργαστηριακού μαθήματος που παρουσιάζεται και αποθηκεύει το προς στιγμήν ανανεωμένο αρχείο. Το αρχείο αυτό περιέχει τα στοιχεία των φοιτητών, όπως ο Αριθμός Μητρώου (ΑΜ), το Ονοματεπώνυμο, το Τμήμα, και πληροφορίες για τη δήλωση τους για το το αντίστοιχο εργαστηριακό τμήμα, ταξινομημένα ανά Κωδικό Εργαστηριακού Τμήματος



Σχήμα 6.17: Επιλογή Λήψης Αρχείου ανά Εργαστηριακό Μάθημα

Είναι σημαντικό να σημειωθεί ότι η εισαγωγή του αρχείου σε .csv , .xslx,.xls πρέπει να προηγηθεί της εξαγωγής των δεδομένων, προκειμένου να υπάρχει μια πλήρως ενημερωμένη εικόνα.

6.5 Είσοδος στην Εφαρμογή σαν Φοιτητής

Κατά την είσοδο στην εφαρμογή, ο φοιτητής έχει πρόσβαση σε 2 υποκατηγορίες, 1 από τις οποίες είναι ελεύθερα προσβάσιμες ενώ ή άλλη προστατεύεται και είναι προσβάσιμη μόνο κατά την περίοδο των δηλώσεων μαθημάτων. Οι υποκατηγορίες που μπορεί να επισκεφθεί είναι οι εξής:

- Εργαστήρια.
- Δηλωθέντα.

Στην υποκατηγορία “Εργαστήρια”, ο φοιτητής μπορεί να δει τη λίστα των διαθέσιμων εργαστηρίων για κάθε μάθημα, μόνο και μόνο αν η περίοδος δηλώσεων του εξαμήνου είναι σε ενεργή κατάσταση. Αν η περίοδος δηλώσεων είναι ενεργή, ο φοιτητής μπορεί να κάνει τη δήλωση του εργαστηρίου που επιθυμεί να παρακολουθήσει μέσω κλικ στο ανάλογο εργαστηριακό τμήμα.

Στην υποκατηγορία “Δηλωθέντα”, ο φοιτητής (και ο καθηγητής) έχει πάντα πρόσβαση, είτε η περίοδος είναι ενεργή είτε έχει λήξει απο τον αυτοματοποιημένο μηχανισμό. Απο την συγκεκριμένη ενότητα μπορεί να δει τη δήλωση των εργαστηριακών τμημάτων που έχει πραγματοποιήσει σε μορφή πίνακα καθώς και σε μορφή καρτών που περιέχουν την κρίσιμη πληροφορία που σίγουρα ενδιαφέρει έναν φοιτητή με πιο φιλικό τρόπο.

Οι δύο παραπάνω υποκατηγορίες επιτρέπουν στον φοιτητή να έχει μια ευέλικτη και οργανωμένη προσέγγιση για τις δηλώσεις των εργαστηριακών τμημάτων του. Μέσω αυτών των λειτουργιών, μπορεί να επιλέξει και να διαχειριστεί τα μαθήματα και τα εργαστηριακά τμήματα που επιθυμεί να παρακολουθήσει κατά τη διάρκεια του ακαδημαϊκού εξαμήνου.

6.5.1 Επιλογή Εργαστήρια

Για φοιτητές: Όταν ο φοιτητής εισέρχεται στην εφαρμογή, θα ζητηθεί από αυτόν να επιλέξει από ποιο εξάμηνο θέλει να του εμφανιστούν τα εργαστήρια. Τα εξάμηνα θα εμφανίζονται αυτόματα ανάλογα με την περίοδο κατά την οποία γίνονται οι δηλώσεις. Για παράδειγμα, κατά την περίοδο των χειμερινών δηλώσεων θα εμφανίζονται μόνο τα εξάμηνα Α, Γ, Ε, Ζ και Θ, ενώ κατά την εαρινή περίοδο δηλώσεων θα εμφανίζονται τα εξάμηνα Β, Δ, Ε, ΣΤ, Η. Επίσης, θα υπάρχει η επιλογή "Χ/Ε" που θα εμφανίζεται και στις δύο περιόδους.

Ο χρήστης θα έχει τη δυνατότητα να επιλέξει ένα ή περισσότερα εξάμηνα. Κατά την επιλογή του εξαμήνου, θα εμφανίζονται όλα τα μαθήματα με την ονομασία, τον κωδικό τους, αν είναι υποχρεωτικής παρακολούθησης και το εξάμηνο διδασκαλίας τους. Ο φοιτητής θα έχει τη δυνατότητα να επιλέξει ένα μάθημα, και τότε θα εμφανίζεται μια αναλυτική περιγραφή του μαθήματος και ένα κουμπί που του επιτρέπει να κάνει τη δήλωσή του.

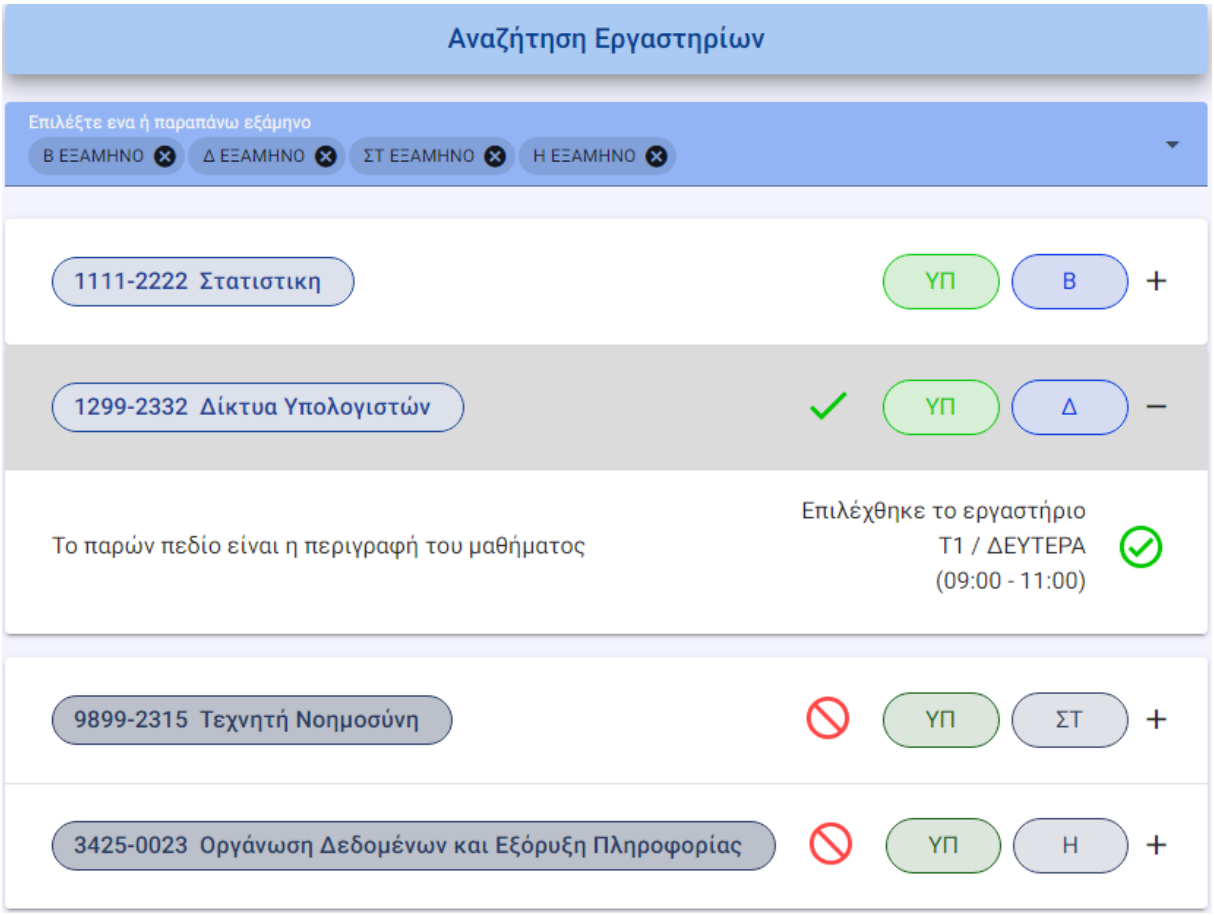
Η εμφάνιση των διαθέσιμων εργαστηριακών μαθημάτων για τα εξάμηνα που έχουν επιλεχθεί για κάθε φοιτητή μπορούν να μεταβάλλονται καθώς υπάρχουν πολλές μεταβλητές που είτε επιτρέπουν την πρόσβαση στην ενότητα της τελικής υποβολή είτε την απαγορεύουν εμφανίζοντας στον φοιτητή την ανάλογη αιτιολογία.

Πέραν της λεκτικής αιτιολογίας, γίνεται εμφάνιση και ανάλογων εικονιδίων που προδιαθέτουν την απόφαση αυτή. Αυτή η λειτουργία έρχεται να προστατέψει τους φοιτητές και κατα συνέπεια και τους

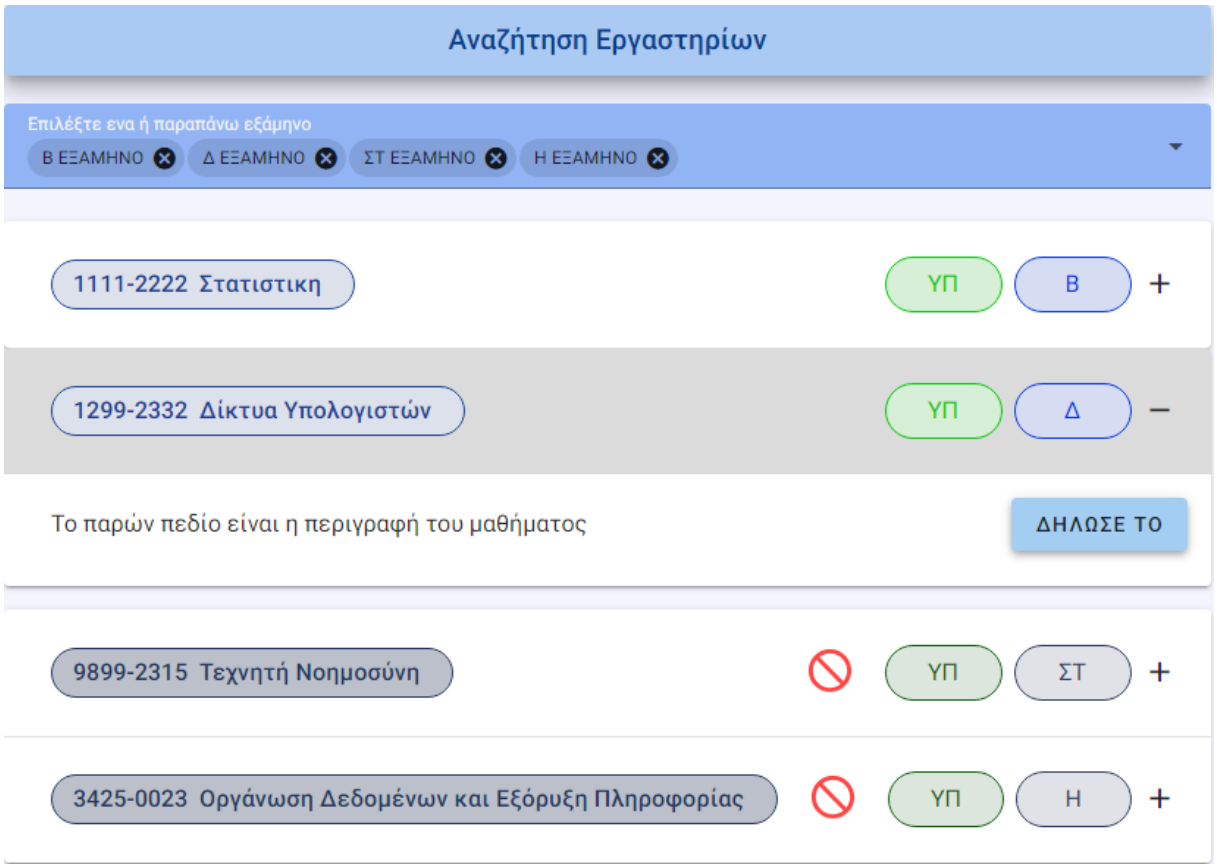
καθηγητές, έτσι ώστε να μην επιτρέπει την πρόσβαση στην ενότητα τελικής δήλωσης ενός φοιτητή όπου:

- Δεν βρίσκεται στο τυπικό εξάμηνο του μαθήματος.
- Δεν είναι ακόμη στην σωστή ημερομηνιακή ομάδα προτεραιότητας.
- Έχει ήδη καλύψει μια θέση στο συγκεκριμένο εργαστηριακό τμήμα.

Αν καμία από τις 3 παραπάνω συνθήκες δεν είναι αληθής τότε η πρόσβαση στην επόμενη ενότητα με ονομασία “Δήλωσε το” του επιτρέπεται.



Σχήμα 6.18: Αναζήτηση Εργαστηριακού Μαθήματος ανά Εξάμηνο



Σχήμα 6.19: Επιλογή Δήλωσης Τμήματος σε Συγκεκριμένο Εργ. Μάθημα

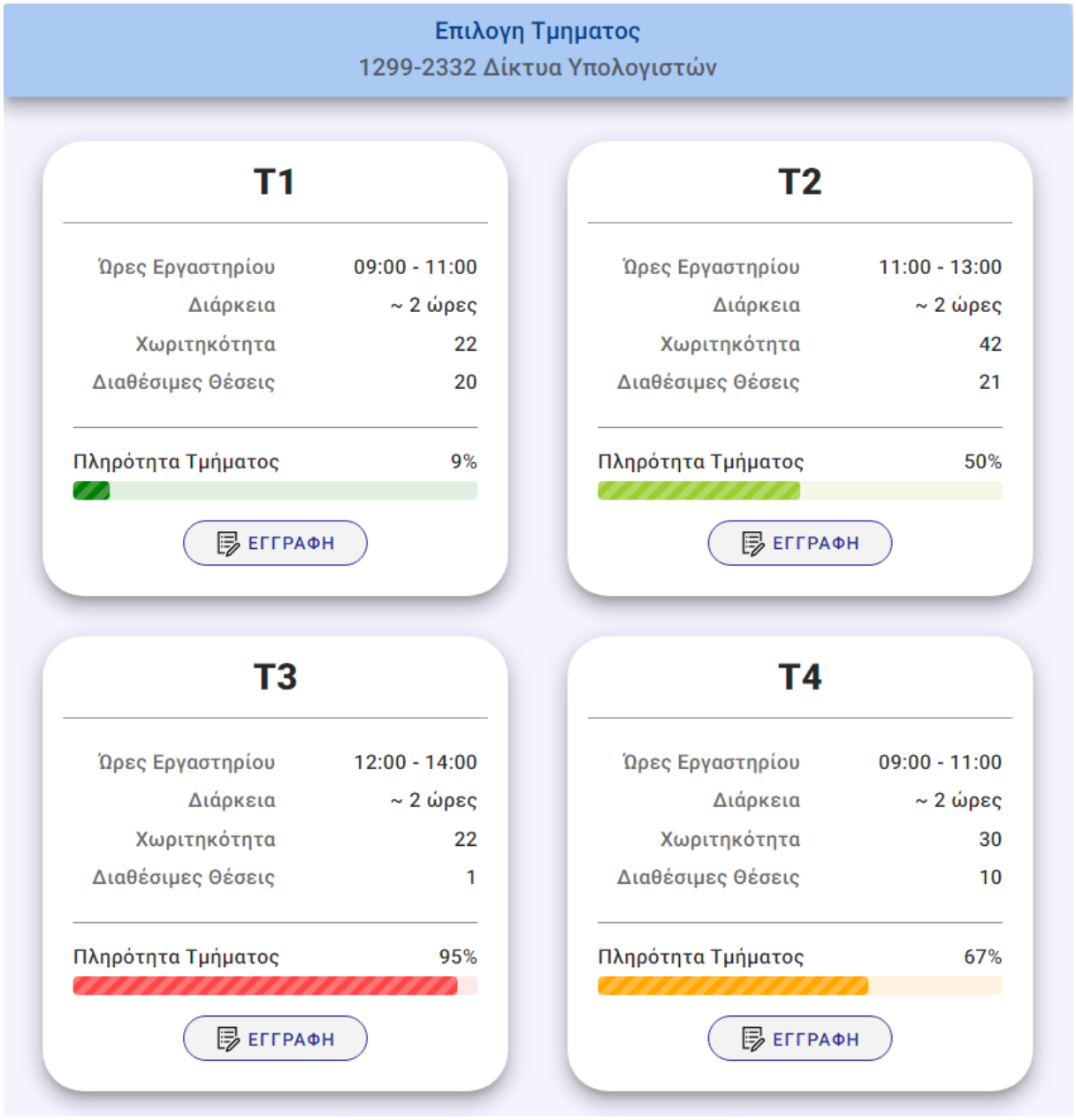
Αφού ο φοιτητής πατήσει το κουμπί "Δήλωσε το", θα μεταφερθεί σε μια νέα σελίδα όπου θα βρίσκονται όλα τα εργαστηριακά τμήματα του μαθήματος υπο μορφή καρτών. Σε αυτές τις κάρτες, ο φοιτητής θα μπορεί να δει διάφορες χρήσιμες πληροφορίες για κάθε τμήμα. Οι εικονιζόμενες κάρτες περιλαμβάνουν:

- Το όνομα του τμήματος, για παράδειγμα "T1", "T2".
- Την ώρα διεξαγωγής του εργαστηριακού μαθήματος, για παράδειγμα "10:00 - 12:00".
- Τη κατα προσέγγιση διάρκεια του μαθήματος, για παράδειγμα ~2 ώρες.
- Τη μέγιστη χωρητικότητα του τμήματος, για παράδειγμα 40.
- Τις υπολειπόμενες διαθέσιμες θέσεις, για παράδειγμα 20.
- Το ποσοστό πλήρωσης του εργαστηρίου εκείνη την χρονική στιγμή με μπάρα προόδου.

Η μπάρα προόδου θα εμφανίζει το ποσοστό πληρότητας τη συγκεκριμένη χρονική στιγμή με φιλικό και κατανοητό τρόπο, με ανάλογη χρωματική κωδικοποίηση. Για παράδειγμα, αν το εργαστηριακό τμήμα “T1” έχει μέγιστη χωρητικότητα 40 άτομα και υπάρχουν διαθέσιμες 20 θέσεις, η μπάρα θα καταλαμβάνει χώρο μέχρι τη μέση της, με ποσοστό πληρότητας 50% και χρώμα κιτρινοπράσινο. Ανάλογα με την μεταβλητή πληρότητα, θα υπάρχει και η αντίστοιχη χρωματική κωδικοποίηση όπως προαναφέρθηκε ως εξής:

- Ανοικτό Πράσινο για ποσοστό >= 0% και < 20%
- Σκούρο πράσινο για ποσοστό >= 20% και < 40%
- Κιτρινοπράσινο για ποσοστό >=40% και <60%
- Πορτοκαλί για ποσοστό >= 60% και < 80%

- Ανοικτό κόκκινο για ποσοστό >= 80%.



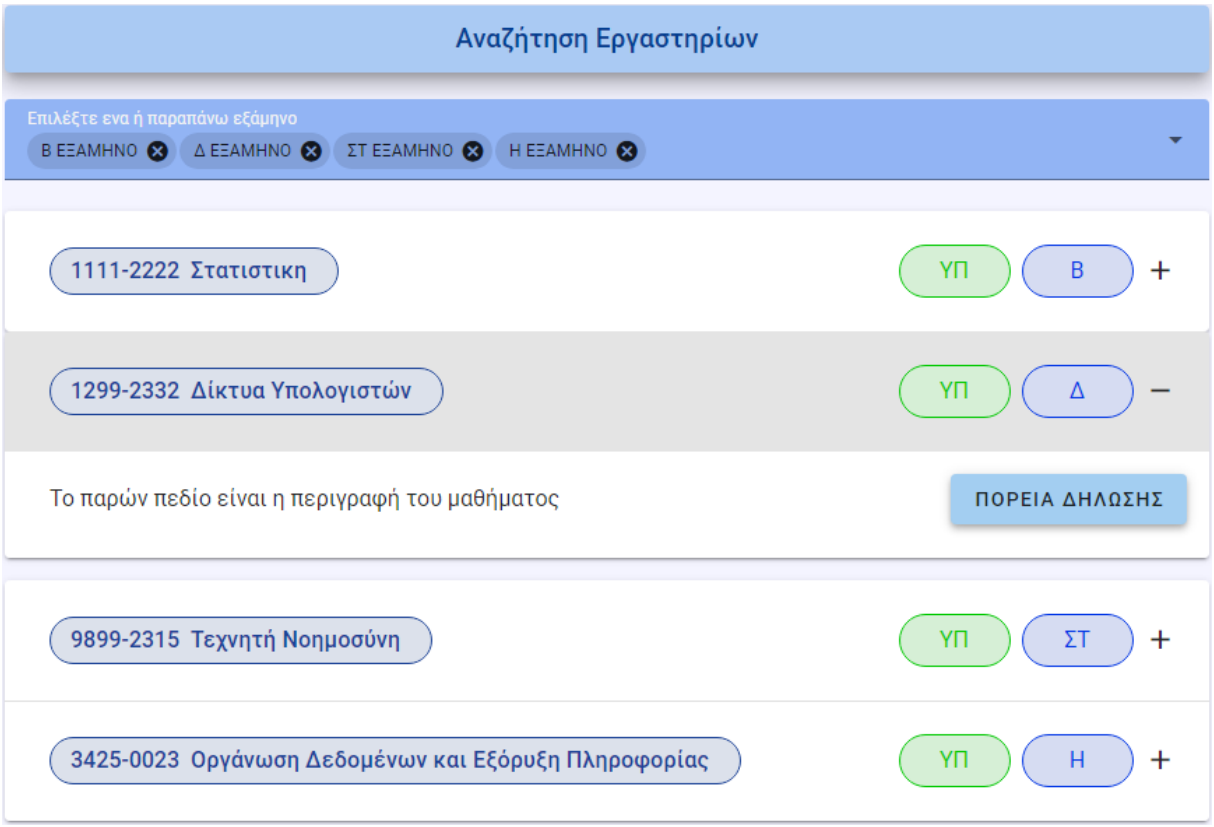
Σχήμα 6.20: Επιλογές Εργαστηρίων προς Δήλωση

Με αυτόν τον τρόπο, ο φοιτητής έχει όλες τις πληροφορίες που χρειάζεται για να αποφασίσει πριν προχωρήσει στη δήλωση του μαθήματος. Επίσης ας σημειωθεί ότι όταν το ποσοστό πληρότητας είναι < 100% η μπάρα προόδου παρουσιάζεται με animation που υποδηλώνει ότι υπάρχουν ακόμη θέσεις για να πληρωθούν, ενώ όταν το τμήμα πληρωθεί πλήρως το animation παύει να υφίσταται για το συγκεκριμένο εργαστήριο.

Για καθηγητές: Η συγκεκριμένη ενότητα εφαρμογής βρίσκει παρόμοια χρήση με τους φοιτητές και για τους καθηγητές, ωστόσο χρήζουν ειδοποιών διαφορών που θα περιγραφούν παρακάτω.

Ο καθηγητής έχει την δυνατότητα επιλογής πολλαπλών εξαμήνων όπως ακριβώς γίνεται με τους φοιτητές, με την διαφορά ότι στον εκάστοτε καθηγητή εμφανίζονται μόνο τα μαθήματα που κατέχει ως

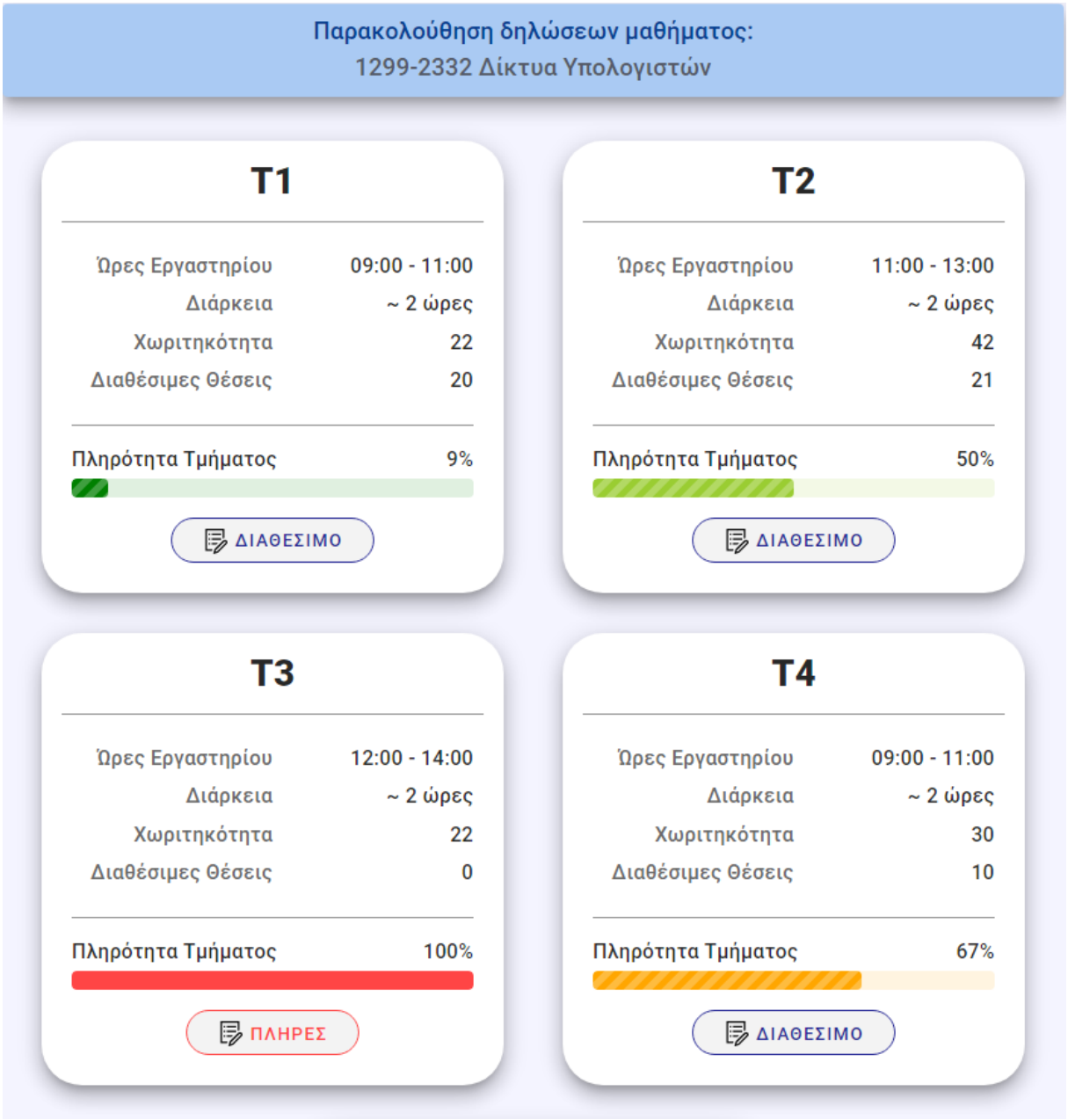
συντονιστής είτε έχει δηλωθεί ως συνεργάτης σε ένα ή περισσότερα τμήματα στο επιλεγμένο και εμφανιζόμενο εξάμηνο.



Σχήμα 6.21: Πρόσβαση στο Σύστημα Παρακολούθησης από Καθηγητή

Ο καθηγητής έχει την επιλογή να παρακολουθήσει την πορεία των δηλώσεων του μαθήματος του μέσω κλικ στο κουμπί “Πορεία Δήλωσης”.

Όταν γίνει επιτυχημένα η πλοήγηση στην συγκεκριμένη ενότητα θα εμφανιστούν οι ίδιες πληροφορίες που εμφανίζονται στους φοιτητές με την διαφορά ότι σε κάθε κάρτα τμήματος θα αναφέρεται η επιλογή τροποποίησης η οποία αν επιλεχθεί, μεταφέρει τον χρήστη στην φόρμα Τροποποίησης του Εργαστηριακού Μαθήματος

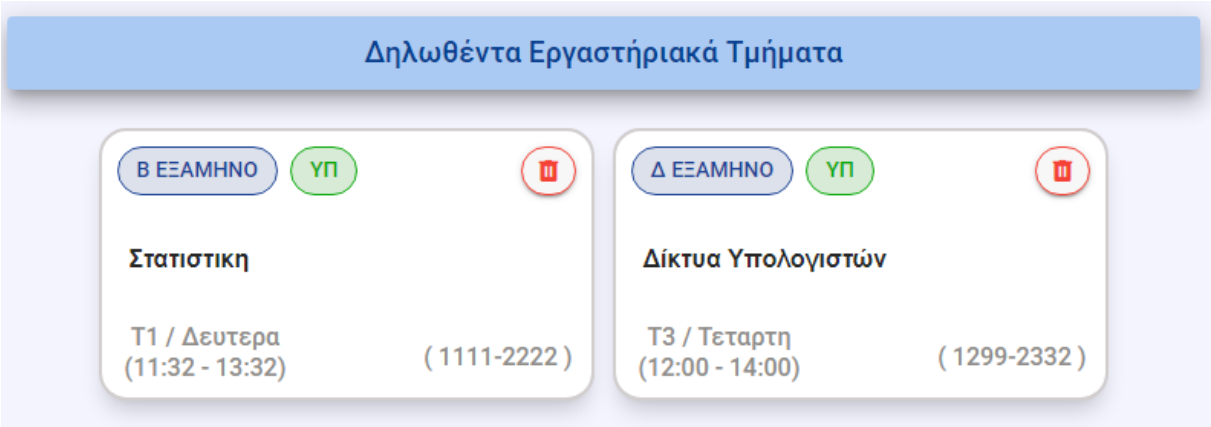


Σχήμα 6.22: Σύστημα Παρακολούθησης Πορείας Δηλώσεων Μαθήματος

Ας σημειωθεί ότι η αλλαγή πληροφορίας από την προαναφερθείσα υποβολή της φόρμα με αλλαγές σε οποιοδήποτε πεδίο βρίσκει άμεσο αντίκτυπο σε αυτή την ενότητα της εφαρμογής. Ο χρήστης πλέον μπορεί να επικυρώσει ότι οι αλλαγές υποβλήθηκαν σωστά με το να ξανά επισκεφτεί στην συγκεκριμένη ενότητα.

6.5.2 Επιλογή Δηλωθέντα

Αφού ολοκληρωθεί χωρίς κάποιο πρόβλημα η δήλωση, ο φοιτητής θα μεταφέρεται αυτόματα στην καρτέλα "Δηλωθέντα". Σε αυτήν την καρτέλα, ο φοιτητής θα δει τη δήλωσή του για όλα τα μαθήματα που έχει επιλέξει. Επίσης ένα πρόγραμμα που θα έχει δημιουργηθεί βάσει των επιλογών αυτών. Αυτό το πρόγραμμα, που λειτουργεί και ως Αποδεικτικό Δήλωσης, μπορεί να καταφορτωθεί στο σύστημα του χρήστη σε μορφή PDF από την επιλογή Μετατροπή σε Pdf που εμφανίζεται υπο την μορφή στατικού κουμπιού.



Σχήμα 6.23: Δηλωθέντα Εργαστήρια Φοιτητή

Εάν, για κάποιο λόγο, ο χρήστης αλλάξει γνώμη και θέλει να αποδεσμεύσει ένα συγκεκριμένο εργαστηριακό τμήμα ή το μάθημα γενικότερα, μπορεί να διαγράψει τη δήλωσή του για το συγκεκριμένο μάθημα από την επιλογή “Κάδος” που βρίσκεται στην εικονιζόμενη κάρτα. Μετά τη διαγραφή, εμφανίζεται ένα μήνυμα προτροπής που επιβεβαιώνει ότι η δήλωση θα χαθεί με την αποδοχή του αιτήματος διαγραφής και ότι η ενέργεια είναι μη αναστρέψιμη. Μόλις το αποδεχτεί, η δήλωση διαγράφεται από τη βάση δεδομένων και οι διαθέσιμες θέσεις στο εργαστηριακό μάθημα αποδεσμεύονται.

Σχήμα 6.24: Αποδεικτικό Δήλωσης και Λειτουργία Εκτύπωσης

Η πλοήγηση εκ νέου του φοιτητή στην επιλογή “Εργαστήρια” **δεν απαγορεύεται** και η διαδικασία μπορεί να ξεκινήσει εκ νέου για αυτόν χωρίς καμία επίπτωση.

Μετά την ολοκλήρωση της περιόδου δηλώσεων ή κατα την διάρκεια της, ο φοιτητής μπορεί να επισκεφτεί κανονικά αυτήν την καρτέλα και να δει τα εργαστηριακά μαθήματα που έχει δηλώσει, καθώς και το ωρολόγιο πρόγραμμα τους σε ανύποπτη χρονική στιγμή. Η μόνη ενέργεια που δεν μπορεί να πραγματοποιήσει είναι η διαγραφή, καθώς μετά το πέρας της περιόδου δηλώσεων δεν υπάρχει το δικαίωμα διαγραφής της δήλωσής του συγκεκριμένου τμήματος του.

6.6 Επίλογος

Σε αυτό το κεφάλαιο πραγματοποιήθηκε εκτενής ανάλυση στη λειτουργία της εφαρμογής και του πως αυτές καλύπτουν τις απαιτήσεις λογισμικού που είχαν τεθεί. Αναλύθηκαν τα δικαιώματα που κατέχει κάθε χρήστης καθώς και τον τρόπο που γίνεται ο διαχωρισμός ρόλων στην εφαρμογή.

Επιπλέον, παρουσιάστηκαν οι ενέργειες που μπορεί να εκτελέσει ο καθηγητής, δηλαδή να δημιουργήσει ένα μάθημα, να το τροποποιήσει και να το διαγράψει, είτε όταν οι δηλώσεις είναι ενεργές, είτε πριν από την έναρξή τους ή ακόμα και μετά τη λήξη τους. Ο καθηγητής μπορεί να έχει σε μορφή υπολογιστικού φύλλου όλες τις πληροφορίες που γνωρίζει για τα μαθήματα του ανά εργαστήριο. Έχει τη δυνατότητα, μέσω της εισαγωγής ενός αρχείου .csv, .xls ή .xlsx με τα άτομα που έχουν πραγματοποιήσει δήλωση για το μάθημα θεωρίας, να φιλτράρει και να οπτικοποιήσει το ποιοι φοιτητές έχουν εν τέλη δικαίωμα να παρακολουθήσουν το εργαστήριο που δήλωσαν και ποιοι όχι.

Τέλος, ο φοιτητής ο οποίος έχει τη δυνατότητα να μπει στην εφαρμογή και να κάνει τη δήλωση του εργαστηρίου που επιθυμεί, αν πληρεί τα ανάλογα κριτήρια. Αν έχει δικαίωμα δήλωσης, μπορεί να αποκτήσει πρόσβαση και ανάλογα με το εξάμηνό που βρίσκεται να δηλώσει μαθήματα έως και αυτό. Κοινή λειτουργία που διαθέτει ο φοιτητής, καθώς και ο καθηγητής, είναι η εξαγωγή του προγράμματος των εργαστηρίων με τις λεπτομέρειες τους σε μορφή .pdf ως “Αποδεικτικού Δήλωσης”.

Κεφάλαιο 7ο: Συμπεράσματα και Προτάσεις Βελτίωσης

Ξεκινώντας, ως πρώτη πρόταση βελτίωσης για την εφαρμογή είναι η υλοποίηση ενός μηχανισμού που θα παρέχει ενημέρωση Real-Time κατά τις δηλώσεις των εργαστηριακών τμημάτων, όπου κάθε φορά που πληρώνεται μια θέση, θα ενημερώνονται αυτόματα όλοι οι χρήστες που βρίσκονται ενεργοί μέσα στην εφαρμογή για την αλλαγή αυτή, χωρίς να απαιτείται ενημέρωση από τον χρήστη, δηλαδή χωρίς να απαιτείται να σταλεί εκ νέου ένα HTTP Request για τον συγκεκριμένο πόρο. Αυτή η λειτουργία θα είναι αρκετά δύσκολη και χρονοβόρα, καθώς θα πρέπει να υλοποιηθεί ένας μηχανισμός όπου η σύνδεση του πελάτη (client) με τον διακομιστή (server) θα είναι συνεχώς ανοιχτή και θα επικεντρώνεται σε επίπεδο χρήστη. Συνεπώς, θα χρειαστεί να καθαιρεθεί η λογική των HTTP Request - Response και του πρωτοκόλλου HTTP γενικότερα και να υιοθετηθεί ένα εντελώς διαφορετικό και φτιαγμένο για τέτοιες απαιτήσεις που ονομάζεται WebSockets Protocol. Αν και υπάρχει η δυνατότητα να εφαρμοστεί το HTTP μέσω Long Polling ή Server Side Event (SSE) η διαδικασία ανάπτυξης τους είναι πιο χρονοβόρα και δύσκολη απο την εξ ολοκλήρου ανάπτυξη με κάποιο Sockets πρωτόκολλο. Έτσι κάθε μία από αυτές τις επιλογές έχει τα πλεονεκτήματά της και τα αντίστοιχα μειονεκτήματά.

Η δυσκολία υλοποίησης της παραπάνω πρότασης σε αυτήν την κατάσταση της εφαρμογής οφείλεται κυρίως σε δύο (2) λόγους:

- Όλη η υπάρχουσα υλοποίηση τόσο στην πλευρά του FrontEnd όσο και στην πλευρά του BackEnd έχει υλοποιηθεί με τη αποκλειστική χρήση των αιτημάτων HTTP (HttpRequest). Η αλλαγή ή η επέκταση αυτής της λογικής σε κάποια από τις παραπάνω λειτουργίες θα καταλογιζόταν κατα πολύ εκτός χρονικού προγραμματισμού. Θα απαιτούσε αρκετό χρόνο για την εκ νέου υλοποίησή και συμμόρφωση των ήδη υπάρχοντών με αυτή καθώς και το ανάλογο testing που θα έπρεπε να γίνει.
- Ο δεύτερος λόγος είναι ότι η υποστήριξη μιας τέτοιας υλοποίησης που απαιτεί τις ταυτόχρονες παράλληλες συνδέσεις πολλαπλών ενεργών χρηστών απαιτεί ένα πολύ ισχυρό μηχάνημα για την σωστή υποστήριξη του τόσο στο FrontEnd που θα εκδοθεί όσο και στο BackEnd που βασίζεται. Οπότε απαιτείται αρκετή μνήμη RAM και Επεξεργαστική Ισχύ που δεν είναι είς γνώση αν έχει στην διάθεση του το Τμήμα.

Επιπλέον, με μερικές ακόμα τροποποιήσεις, η εφαρμογή θα μπορούσε να αναχθεί και να φιλοξενεί δηλώσεις μαθημάτων που διεξάγονται και από άλλα Τμήματος του Πανεπιστήμιου.

Κάποιες άλλες ιδέες που θα μπορούσαν να υλοποιηθούν και να προστεθούν στον αρχικό σχεδιασμό και να βελτιώσουν τη λειτουργία της εφαρμογής, με γενικό στόχο την προσθήκη ευκολιών και τη βελτίωση της απόδοσης είναι:

- Βελτιστοποίηση της σύνδεσης με τη βάση δεδομένων. Αξιοποιώντας κατάλληλες τεχνικές caching, thread-pooling και καλύτερων τεχνικών indexing, μπορεί να επιτευχθεί βελτίωση της απόκρισης της εφαρμογής , της απόδοσης των ερωτημάτων προς τη σχεσιακή βάση δεδομένων και την απόκριση αυτών πίσω στην εφαρμογή.
- Χρήση μεμονωμένης βάσης για Cache. Η προσθήκη μιας μεμονωμένης Caching Β.Δ. όπως η πολύ γνωστή Redis θα βελτίωνε επίσης πολύ την απόκριση των ερωτημάτων που προέρχονται από τους χρήστες της εφαρμογής σερβίροντας στατικά και μη συχνά μεταβαλλόμενα δεδομένα σε πολύ μικρό χρόνο αποκρίσεως χωρίς να χρειάζεται να απασχολείται εκ νέου η εφαρμογή για τον υπολογισμό και το σερβίρισμα τους.

- Βελτίωση της ασφάλειας: Με την υλοποίηση μηχανισμών ασφαλείας, όπως τη χρήση κρυπτογράφησης και την επαλήθευση ταυτότητας, μπορεί να ενισχυθεί η ασφάλεια της εφαρμογής και να προστατευθούν τα προσωπικά δεδομένα των χρηστών ακόμη καλύτερα.
- Ανάπτυξη ενός δυναμικού συστήματος αναζήτησης: Με την εφαρμογή ενός προηγμένου συστήματος αναζήτησης, οι χρήστες θα μπορούν να εντοπίζουν εύκολα διαθέσιμες θέσεις σε εργαστήρια με βάση κριτήρια, όπως ημερομηνία, ώρα, κατηγορία ακόμη και Τμήματος αν η εφαρμογή αναχθεί για χρήση σε περισσότερα του ενός Τμήματα.
- Ανάπτυξη μιας ξεχωριστής διαχειριστικής διεπαφής με περισσότερες λειτουργίες: Η προσθήκη μιας ξεχωριστής διαχειριστικής διεπαφής για τους διαχειριστές του συστήματος και μόνο σε ξεχωριστή ιστοσελίδα / εφαρμογή θα μπορούσε να φιλοξενήσει περισσότερες επιλογές που θα αποφασίζονταν μέσα από απαιτήσεις των διδασκόντων συμπεριλαμβάνοντας και ότι ήδη παρέχει η υπάρχουσα υλοποίηση.
- Επέκταση λειτουργικότητας: Με την προσθήκη νέων λειτουργιών, όπως τη δυνατότητα αποθήκευσης και κοινής χρήσης αρχείων, τη δυνατότητα σύνδεσης με εξωτερικά συστήματα (π.χ. ηλεκτρονικό ημερολόγιο) και τη δυνατότητα επικοινωνίας με άλλους χρήστες μέσω μηνυμάτων, η εφαρμογή έπεται να γίνει πιο πλήρης και ακόμη περισσότερο λειτουργική.

Μέσω της υλοποίησης όλων των προηγουμένων προτάσεων βελτίωσης (και σίγουρα πολλών άλλων που μπορούν να προταχθούν), η εφαρμογή μπορεί να ενισχύσει την απόδοση, την ασφάλεια, την ευκολία χρήσης και την εμπειρία των χρηστών, ενισχύοντας έτσι τη λειτουργικότητα και την αξία της στο σύνολο της για τον τελικό χρήστη. Για το λόγω του αληθές σε περίπτωση που καλυφθούν οι παραπάνω προτάσεις η εφαρμογή ανάγεται από ένα απλό σύστημα δηλώσεων εργαστηρίων σε ένα νέο υβρίδιού Uniportal (με περισσότερες λειτουργίες) με Μέσο Κοινωνικής Δικτύωσης εξ ολοκλήρου σχεδιασμένο απο τελειόφοιτους φοιτητές.

Ωστόσο, αν και οι παραπάνω προτάσεις βελτίωσης ακούγονται δελεαστικές και όντως θα ενισχύσουν τη λειτουργικότητα και όχι μόνο πρέπει να ληφθεί υπόψη ότι η υλοποίησή τους μπορεί να απαιτήσει υπέρμετρο πρόσθετο χρόνο υλοποίησης και πόρους, καθώς και την προσαρμογή του υπάρχοντος κώδικα και σχήματος της εφαρμογής με τέτοιο τρόπο ώστε να μπορούν να υποστηριχθούν όλα όσα υποστηριζόταν και πριν την προσθήκη των νέων λειτουργιών. Επομένως, η εκτέλεση αυτών των προτάσεων θα πρέπει να γίνει με προσεκτική σχεδίαση και ανάλυση, λαμβάνοντας υπόψη τόσο τις υπάρχουσες νέες ανάγκες όσο και τις προηγούμενες που υπήρχαν μαζί με τους περιορισμούς που αυτές έθεταν στο έργο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] “Entity Framework Core Documentation”, Microsoft, May 25, 2021.[Online]. Available: https://learn.microsoft.com/en-us/ef/core/.
[2] “.NET Core Overview”, TutorialsTeacher.com.[Online]. Available: https://www.tutorialsteacher.com/core/dotnet-core.
[3] “Dependency injection in ASP.NET Core”, Microsoft, May 09, 2023.[Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-7.0
[4]H. S. Domareski, “Dependency Injection and Service Lifetimes in .NET Core”, Medium.com, Nov. 25 2020.[Online]. Available: https://henriquesd.medium.com/dependency-injection-and-service-lifetimes-in-net-core-ab9189349420.
[5] “What is Web API?”, TutorialsTeacher.com.[Online]. Available: https://www.tutorialsteacher.com/webapi/what-is-web-api.
[6] R. Peres, “Entity Framework Core Succinctly”, Syncfusion.[Online]. Available: https://www.syncfusion.com/succinctly-free-ebooks/entity-frame-work-core-succinctly/setting-up.
[7] I. Tiwari, “ASP.NET Core 6 Web API CRUD With Entity Framework”, C# Corner, Nov. 24, 2022.[Online]. Available: ASP.NET Core 6 Web API CRUD With Entity Framework.
[8] G. Karroy, “Web API Token Based Authentication”, C# Corner, Jan. 08, 2021.[Online]. Available: https://www.c-sharpcorner.com/article/web-ap/.
[9] S. Agarwal, “Middleware in ASP.NET Core ”, DNC Magazine(dotnetcurry), Mar. 15, 2023. [Online]. Available: https://www.dotnetcurry.com/aspnet-core/middleware-in-aspnetcore.
[10] dotConnectTeam, “Scheduling Cron Jobs in ASP.NET 6 and C#”, Devart , Jul. 5, 2022. [Online]. Available: Scheduling Cron Jobs in ASP.NET 6 and C#.
[11]S. Gordon, “What Are .NET Worker Services?”, Steve Gordon – Code With Steve, Mar. 30, 2020.[Online]. Available: https://www.stevejgordon.co.uk/what-are-dotnet-worker-services.
[12] “Introduction to Worker Services in ASP.NET Core”, The Tech Platform, Sep 6, 2022. [Online]. Available: https://www.thetechplatform.com/post/worker-services-in-asp-net-core.
[13] “Overview of ASP.NET Core MVC” ,Microsoft, Jun. 27, 2022.[Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0
[14] The PostgreSQL Global Development Group ,“What is PostgreSQL”, Postgresql.org.[Online]. Available: https://www.postgresql.org/about/
[15] R.Kumar, “What is PostgreSQL – Introduction”, Geeksforgeeks.org.[Online]. Available: https://www.geeksforgeeks.org/what-is-postgresql-introduction/
[16] T.Davidson, ”Why Use PostgreSQL For Your Next Project?”, CleanComit.com, Nov. 13,2022.[Online].
Available: https://cleancommit.io/blog/why-use-postgresql-for-your-next-project/
[17] “The Vue.js 3 API Tutorial”, KoderHQ.com.[Online]. Available: https://www.koderhq.com/tutorial/vue/api/
[18] P. Gillin, ”What is Component-Based Architecture?”, Mendix.com, Apr. 8, 2022.[Online].
Available: https://www.mendix.com/blog/what-is-component-based-architecture/
[19] “Components Basics”, Vuejs.org.[Online].
Available: https://vuejs.org/guide/essentials/component-basics.html
[20] M. Maribojoc, “A Complete Guide to Vue Lifecycle Hooks - with Vue 3 Updates”, LearnVue.co, Dec. 25, 2020.[Online].
Available: https://learnvue.co/articles/vue-lifecycle-hooks-guide
[21] “Composition API: Lifecycle Hooks”, Vuejs.org.[Online].
Available: https://vuejs.org/api/composition-api-lifecycle.html
[22] “Style Guide”, Vuejs.org.[Online].
Available: https://vuejs.org/style-guide/
[23] “Single File Components *.vue”, Worldline.github.io.[Online].

Available: https://worldline.github.io/vuejs-training/views/#single-file-components-vue
[24] “SFC Syntax Specification”, Vuejs.org.[Online].
Available: https://vuejs.org/api/sfc-spec.html#overview
[25] H. Zakelšek, “Options API vs. Composition API”, Medium.com, Oct.18, 2022.[Online]. Available: https://medium.com/codex/options-api-vs-composition-api-4a745fb8610
[26] J. Bland, ”Vue 3 Composition API vs Options API”, JenifferBland.com.[Online].
Available: https://www.jenniferbland.com/vue-3-composition-api-vs-options-api/
[27] “Build-in Directives”, Vuejs.org .[Online].
Available: https://vuejs.org/api/built-in-directives.html
[28] “Custom Directives”, Vuejs.org.[Online].
Available: https://vuejs.org/guide/reusability/custom-directives.html

ΠΑΡΑΡΤΗΜΑ Α : Αποδεκτά MIME TYPES προς Μεταφόρτωση Αρχείων

```
const _acceptableMimeTypes = [
    "application/vnd.ms-excel",
    "application/msexcel",
    "application/x-msexcel",
    "application/x-ms-excel",
    "application/x-excel",
    "application/x-dos_ms_excel",
    "application/xls",
    "application/x-xls",
    "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
    "text/csv"
]
```

ΠΑΡΑΡΤΗΜΑ Β : Αλγόριθμος Διαχωρισμού Ημερομηνιών σε Γλώσσα C#

```
>var from = new DateTimeOffset(2023, 04, 01); //Αποτέλεσμα 01.04.2023

>var to = new DateTimeOffset(2023,04,30); //Αποτέλεσμα 30.04.2023

> var res = to.Subtract(from); //Αποτέλεσμα 29 ημέρες

>var calcDays = Math.Floor((res.TotalDays / 3)); //Αποτέλεσμα 9

>var highestPriorityDt = from; //Αποτέλεσμα 01.04.2023

>var moderatePriorityDt = highestPriorityDt.AddDays(calcDays); //Αποτέλεσμα  10.04.2023

>var lowestPriorityDt = moderatePriorityDt.AddDays(calcDays); //Αποτέλεσμα  19.04.2023

>var newEndingToTimestamp = lowestPriorityDt.AddDays(calcDays); //Αποτέλεσμα 28.04.23

> var endOfPeriodDto = to.CompareTo(newEndingToTimestamp) < 0

? newEndingToTimestamp : to;  //Αποτέλεσμα 30.04.2023
```