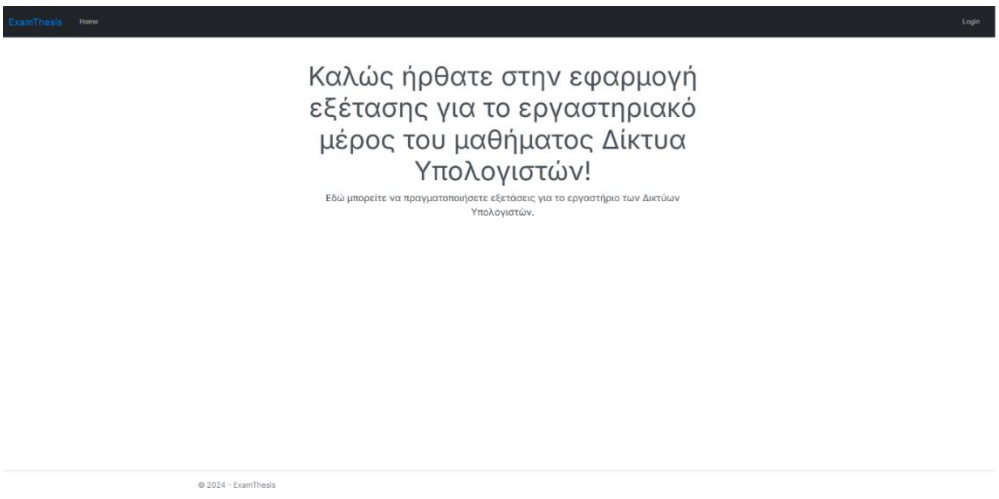
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟ ΣΥΣΤΗΜΑ ΕΞΕΤΑΣΗΣ ΕΡΓΑΣΤΗΡΙΟΥ ΔΙΚΤΥΩΝ ΥΠΟΛΟΓΙΣΤΩΝ»



Του φοιτητή
Φωτιάδη Δημήτριου
Αρ. Μητρώου: 164772

Επιβλέπων
Αμανατιάδης Δημήτριος

Θεσσαλονίκη, Σεπτέμβριος 2024

Τίτλος Π.Ε.: Αυτοματοποιημένο σύστημα εξέτασης εργαστηρίου Δικτύων Υπολογιστών
Κωδικός Π.Ε. 22323
Ονοματεπώνυμο φοιτητή: Φωτιάδης Δημήτριος
Ονοματεπώνυμο εισηγητή: Αμανατιάδης Δημήτριος
Ημερομηνία ανάληψης Π.Ε. 09-11-2022
Ημερομηνία περάτωσης Π.Ε. 10-09-2024

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

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

Πρόλογος

Κατά την αναζήτηση θέματος για την εκπόνηση της Πτυχιακής εργασίας ήρθαμε σε επικοινωνία με τον συντονιστή του συγκεκριμένου θέματος, όπου και παρατηρήθηκε ένα σημαντικό πρόβλημα στην διαδικασία των εξετάσεων του εργαστηριακού μέρους του μαθήματος Δικτύων Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Έπειτα από την ανάλυση του προβλήματος, προχωρήσαμε στην εύρεση λύσης μέσω μίας εφαρμογής αυτοματοποιημένου συστήματος εξέτασης στο πλαίσιο της Πτυχιακής εργασίας.

Το σύστημα στοχεύει στην απλοποίηση της διενέργειας εξετάσεων του εργαστηριακού μέρους των Δικτύων Υπολογιστών, παρέχοντας λύσεις για τα προβλήματα που αντιμετωπίζουν οι καθηγητές και φοιτητές αλλά και για την συνολική βελτίωση της διαδικασίας.

Περίληψη

Η παρούσα Πτυχιακή εργασία προήλθε από το πρόβλημα που είχε παρατηρηθεί στην διαδικασία της εξέτασης του εργαστηριακού μέρους των Δικτύων Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Η τωρινή διαδικασία της εξέτασης του εργαστηριακού μέρους του μαθήματος γίνεται δια ζώσης μέσω των εργαστηριακών αιθουσών του τμήματος με ένα μεγάλο πλήθος φοιτητών να παίρνει κάθε χρόνο μέρος σε αυτήν. Σκοπός της εφαρμογής είναι να δημιουργηθεί ένα Αυτοματοποιημένο Σύστημα Εξέτασης το οποίο θα είναι ενιαίο για όλους του καθηγητές που διδάσκουν το μάθημα των Δικτύων Υπολογιστών αλλά και για τους φοιτητές.

Με την υλοποίηση του Αυτοματοποιημένου Συστήματος Εξέτασης, παρέχεται ένα απλοποιημένο περιβάλλον για την διαδικασία των εξετάσεων με την δυνατότητα της αυτόματης βαθμολόγησης. Το σύστημα εξέτασης χωρίζει τους χρήστες σε δύο ρόλους. Αυτή του καθηγητή και του φοιτητή. Από την πλευρά του καθηγητή, έχει την δυνατότητα δημιουργίας εξέτασης με τις επιλογές χρόνου έναρξης και λήξης καθώς επίσης και τον ορισμό ελάχιστης βάσης επιτυχίας στην κάθε εξέταση ανάλογα με τον βαθμό δυσκολίας που αυτός έχει ορίσει κατά την δημιουργία των ερωτήσεων της εξέτασης. Επίσης μετά την λήξη της εξέτασης έχει την δυνατότητα εξαγωγής των αποτελεσμάτων σε μορφή Spreadsheet (.xlsx). Από την πλευρά του φοιτητή, δίνεται η δυνατότητα να παίρνει μέρος στην εξέταση του μαθήματος μέσω ενός εύχρηστου περιβάλλοντος στο οποίο θα του δίνεται η δυνατότητα μετά το πέρας της εξέτασης, να μπορεί να δει εάν έχει επιτύχει στην εξέταση ή όχι. Έτσι ο φοιτητής δεν θα χρειάζεται να περιμένει για την ανακοίνωση των αποτελεσμάτων αλλά και οι καθηγητές δεν θα χρειάζεται να βαθμολογούν ένα μεγάλο πλήθος γραπτών.

Η περίοδος της εξέτασης ορίζεται αρχικά από τον καθηγητή και μπορεί να λήξει αυτόματα μετά την πέρας της προκαθορισμένης ώρας.

«AUTOMATED COMPUTER NETWORK LABORATORY EXAMINATION SYSTEM»

«Dimitrios Fotiadis»

Abstract

This thesis was the result of a problem that had been observed in the examination process of the laboratory part of the Computer Networks of the Department of Computer Engineering and Electronic Systems. The current process of examining the laboratory part of the course is conducted face-to-face through the department's laboratory classrooms with a large number of students taking part in it every year. The purpose of the application is to create an Automated Examination System which will be uniform for all the professors teaching the Computer Networks course as well as for the students.

With the implementation of the Automated Examination System, a simplified environment is provided for the examination process with the possibility of automatic grading. The examination system divides users into two roles. That of the teacher and the student. On the professor's side, he has the ability to create an exam with the options of start and end time as well as setting a minimum pass mark in each exam depending on the difficulty level he has set while creating the exam questions. Also, after the end of the exam he has the ability to export the results in Spreadsheet (.xlsx) format.

On the student's side, the student is given the opportunity to take part in the examination of the course through an easy-to-use environment in which he/she will be able to see after the examination whether he/she has succeeded in the examination or not. Thus the student will not have to wait for the announcement of the results and the teachers will not have to grade a large number of papers.

The examination period is initially set by the professor and can be automatically ended after the predetermined time.

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα καθηγητή, Αμανατιάδη Δημήτριο για την καθοδήγηση αλλά και την εμπιστοσύνη που έδειξε στο πρόσωπο μου για την ανάθεση της εν λόγω πτυχιακής εργασίας. Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου που είναι πάντα δίπλα μου. Τέλος, θα ήθελα προσωπικά να ευχαριστήσω τον φίλο μου Δημήτρη για την ώθηση που μου έδωσε για να συνεχίσω και να ολοκληρώσω την πτυχιακή εργασία.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

API	Application Programming Interface
CSS	Cascading Style Sheets
DOM	Document Object Model
EF Core	Entity Framework Core
HTML	Hypertext Markup Language
IOC	Inversion of Control
IP	Internet Protocol
JS	JavaScript
MVC	Model-View-Controller
PDF	Portable Document Format
SEO	Search Engine Optimization
SQL	Structured Query Language
TDD	Test Driven Development
UI	User Interface
URL	Uniform Resource Locator

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθεί μία λύση για ένα από τα σημαντικότερα προβλήματα που αντιμετωπίζουν φοιτητές και καθηγητές του τμήματος Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων. Το σημαντικό αυτό πρόβλημα αφορά την διενέργεια εξετάσεων στο εργαστηριακό μέρος του μαθήματος των Δικτύων Υπολογιστών.

1.2 Τωρινή Διαδικασία Διενέργειας Εξετάσεων

Οι εξετάσεις στο εργαστηριακό μέρος του μαθήματος των Δικτύων Υπολογιστών διενεργείται δια ζώσης στις εργαστηριακές αίθουσες του τμήματος. Ο καθηγητής μοιράζει τα θέματα και οι φοιτητές μέσω του υπολογιστή χρησιμοποιούν τα διάφορα εργαλεία που χρειάζονται για την επίλυση των θεμάτων. Συμπληρώνουν τα θέματα και τα παραδίδουν στον καθηγητή μετά την λήξη της εξέτασης. Ο συγκεκριμένος τρόπος εξέτασης του εργαστηριακού μέρους του μαθήματος αντιμετωπίζει κρίσιμα προβλήματα τα οποία μέχρι στιγμής δεν έχουν επιλυθεί.

Η δυσκολία του αντικειμένου καθώς και η ανάγκη για συστηματική παρακολούθηση που απαιτούνται για την κατανόηση του μαθήματος των Δικτύων Υπολογιστών έχουν οδηγήσει στο αποτέλεσμα πολλοί φοιτητές να αντιμετωπίζουν δυσκολίες και αρκετοί από αυτούς αποτυγχάνουν στις εξετάσεις του εργαστηρίου. Αυτό το γεγονός δημιουργεί μεγάλο πρόβλημα διότι κάθε χρόνο φοιτητές που έχουν αποτύχει στην εξέταση του εργαστηρίου θα πρέπει να επανεξεταστούν συνεπώς να δημιουργείται μεγάλο πλήθος εξεταζόμενων. Αυτή η αύξηση των φοιτητών επιβαρύνει την διαχείριση των εξετάσεων του μαθήματος και της αξιολόγησης τον καθηγητή επηρεάζοντας ενδεχομένως και την ποιότητα της αξιολόγησης.

Ένα από τα κυριότερα προβλήματα της τωρινής διαδικασίας εξετάσεων είναι ότι ο κάθε καθηγητής που έχει αναλάβει ένα από τα τμήματα του μαθήματος δημιουργεί διαφορετική εξέταση από τους άλλους καθηγητές. Αυτό το πρόβλημα δημιουργεί άνισες συνθήκες αξιολόγησης μεταξύ των φοιτητών καθώς κάποιοι από αυτούς μπορεί να εξετάζονται σε πιο απαιτητικά θέματα και οι υπόλοιποι σε λιγότερο. Το συγκεκριμένο πρόβλημα επιδεινώνει την αξιοπιστία της αξιολόγησης του μαθήματος. Επιπρόσθετα, αρκετοί φοιτητές εφόσον αποτύχουν στην πρώτη εξέταση του εργαστηρίου, καταφεύγουν στην δεύτερη εξέταση σε άλλους καθηγητές οι οποίοι μπορεί να μην βαθμολογούν κατά τα λεγόμενα τους αυστηρά τα γραπτά τους.

1.3 Προσωρινές Λύσεις Συστημάτων Εξέτασης

Την εποχή του κορονοϊού όπου τα τμήματα είχανε παραμείνει κλειστά λόγο της ολικής καραντίνας της χώρας, είχε δημιουργηθεί η ανάγκη για ηλεκτρονικές πλατφόρμες εξέτασης εξ αποστάσεως. Μία από αυτές τις πλατφόρμες εξέτασης είναι το «exams-iee.the.ihu.gr» όπου δημιουργήθηκε από τρίτους για όλα τα τμήματα του Διεθνούς Πανεπιστημίου και χρησιμοποιήθηκε συνολικά στο μεγαλύτερο μέρος των εξετάσεων των μαθημάτων. Το συγκεκριμένο σύστημα εξέτασης δίνει την δυνατότητα διεξαγωγής εξέτασης με ερωτήσεις πολλαπλής επιλογής, ερωτήσεις ανάπτυξης, αντιστοίχισης και σωστού/λάθους με αυτόματη βαθμολόγηση. Επίσης το συγκεκριμένο σύστημα δίνει στον καθηγητή την επιλογή εμφάνισης του τελικού βαθμού του φοιτητή ή την απόκρυψη του καθώς και την επιλογή για το αν ήθελε να εμφανίζει τις σωστές απαντήσεις στο τέλος της κάθε εξέτασης.

Το συγκεκριμένο σύστημα όπως και πολλά άλλα που δημιουργήθηκαν από την μία πλευρά έλυναν το πρόβλημα της εξέτασης του μεγάλου πλήθους φοιτητών αλλά δεν παρείχαν στους καθηγητές εργαστηρίων όπως στο μάθημα των Δικτύων Υπολογιστών την πλήρη ελευθερία μορφοποίησης της εξέτασης με βάσει τις ανάγκες του κάθε μαθήματος με αποτέλεσμα στις εξετάσεις των εργαστηρίων πολλοί καθηγητές προσπάθησαν να δημιουργήσουν εξετάσεις με την μορφή της δια ζώσης εξέτασης δίνοντας θέματα σε ηλεκτρονική μορφή τύπου PDF ή σε μορφή Word και καλούσαν τους φοιτητές να δημιουργήσουν ένα αρχείο κειμένου όπου εκεί θα αποτύπωναν τις απαντήσεις τους. Όπως γίνεται αντιληπτό ο συγκεκριμένος τρόπος δημιουργούσε μία σύγχυση στις τάξεις των φοιτητών αλλά και των καθηγητών.

1.4 Αντικείμενο Εργασίας

Το αντικείμενο της παρούσας πτυχιακής εργασίας εστιάζει στην κατασκευή ενός αυτοματοποιημένου συστήματος εξέτασης για το μάθημα του εργαστηρίου των Δικτύων Υπολογιστών. Μέσα από αυτό το σύστημα ο καθηγητής θα μπορεί να δημιουργεί και να διαχειρίζεται ερωτήσεις των εξετάσεων του μαθήματος, βάζοντας τις ερωτήσεις σε κατηγορίες. Στην συνέχεια θα μπορεί να δημιουργεί πακέτα ερωτήσεων με σκοπό την δημιουργία πολλαπλών θεμάτων για μία ή και παραπάνω κατηγορίες.

Ο καθηγητής επίσης θα μπορεί να δημιουργεί εξετάσεις ορίζοντας την διάρκειας της, δηλαδή τον χρόνο έναρξης και λήξης της εξέτασης, τον μέγιστο βαθμό που μπορεί να λάβει κάποιος φοιτητής, την ελάχιστη βάση επιτυχίας καθώς και να επιλέξει ποιες κατηγορίες ερωτήσεων θα συμπεριληφθούν στην εξέταση. Οι ερωτήσεις θα βαθμολογούνται αυτόματα από το σύστημα εξέτασης βάσει των πόντων που έχει ορίσει ο καθηγητής. Για κάθε ερώτηση θα υπάρχει και αρνητική βαθμολογία σε περίπτωση λάθους απάντησης για την αποφυγή τυχαίων απαντήσεων. Στο τέλος της εξέτασης θα μπορεί να κάνει εξαγωγή των αποτελεσμάτων ανά εξέταση όπου θα του εμφανίζει τον μοναδικό κωδικό της εξέτασης, το αναγνωριστικό του φοιτητή και τον τελικό βαθμό του.

1.5 Στόχος εργασίας

Ο στόχος της εργασίας είναι η δημιουργία ενός αυτοματοποιημένου συστήματος εξέτασης που θα δίνει την δυνατότητα στον καθηγητή όσο μεγάλο και να είναι το πλήθος των εξεταζόμενων να μπορεί να ανταπεξέλθει στις απαιτήσεις των εξετάσεων καθώς και στην γρήγορη και αξιόπιστη αξιολόγηση τους.

Το ζητούμενο είναι να υπάρχει ένα απλό και εύχρηστο περιβάλλον όπου θα επιτρέπει στον καθηγητή να δημιουργεί εξετάσεις με βάσει τις ανάγκες του εργαστηρίου καθώς και να μειώσει τις διαδικασίες αξιολόγησης μεγάλου πλήθους φοιτητών ώστε να κυλούν οι διαδικασίες όσο το δυνατόν γρηγορότερα και με απόλυτη αξιοπιστία. Επίσης, θα εξαλειφθεί και το φαινόμενο της αλλαγής προτίμησης των φοιτητών σε συγκεκριμένους εξεταστές καθώς οι αξιολόγηση της εξέτασης θα γίνεται αυτοματοποιημένα από το σύστημα.

1.6 Επίλογος

Σε αυτό το κεφάλαιο αναφέρθηκαν αρχικά τα προβλήματα που αντιμετωπίζει το Τμήμα στην διαδικασία εξέτασης του εργαστηριακού μέρους του μαθήματος Δικτύων Υπολογιστών. Έπειτα παρουσιάστηκε η τωρινή διαδικασία εξέτασης και οι προσωρινές λύσεις που υπάρχουν. Τέλος, παρουσιάζεται το αντικείμενο της Πτυχιακής εργασίας καθώς και ποια προβλήματα στοχεύει να επιλύσει.

Κεφάλαιο 2ο: Εισαγωγή στις Διαδικτυακές Εφαρμογές

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα αναλυθεί περιληπτικά το τι είναι μια διαδικτυακή εφαρμογή τις δημοφιλέστερες τεχνολογίες ανάπτυξης καθώς επίσης τα πλεονεκτήματα που έχει η κάθε μία και τους λόγους για τους οποίους είναι οι πιο διαδεδομένες.

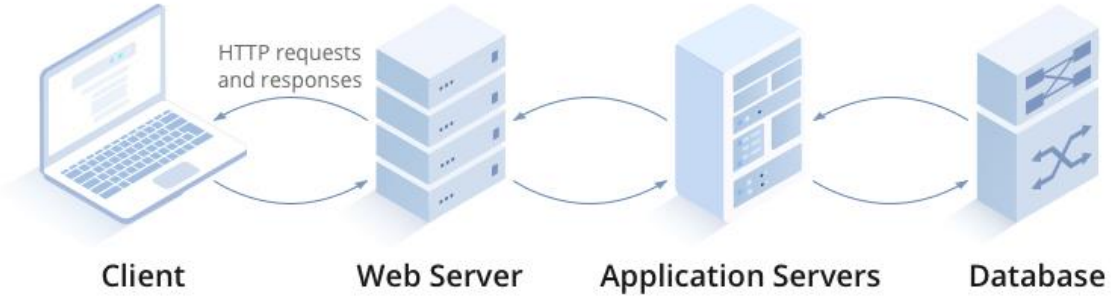
2.2 Τι είναι μια διαδικτυακή εφαρμογή

Μία διαδικτυακή εφαρμογή (web application) είναι ένα πρόγραμμα εφαρμογής που αποθηκεύεται σε έναν απομακρυσμένο διακομιστή (server) και παρέχεται μέσω του διαδικτύου μέσω μιας διεπαφής ενός προγράμματος περιήγησης (browser interface).

Οι προγραμματιστές σχεδιάζουν διαδικτυακές εφαρμογές για μια μεγάλη ποικιλία χρήσεων και χρηστών, από έναν οργανισμό έως ένα άτομο για πολλούς λόγους. Οι διαδικτυακές εφαρμογές που χρησιμοποιούνται συνήθως μπορεί να περιλαμβάνουν ηλεκτρονικό ταχυδρομείο, ηλεκτρονικές αριθμομηχανές ή καταστήματα ηλεκτρονικού εμπορίου. Ενώ οι χρήστες μπορούν να έχουν πρόσβαση σε ορισμένες εφαρμογές ιστού μόνο από ένα συγκεκριμένο πρόγραμμα περιήγησης, οι περισσότερες είναι διαθέσιμες ανεξάρτητα από το πρόγραμμα περιήγησης.

Οι διαδικτυακές εφαρμογές δεν χρειάζεται να εγκατασταθούν σαν ένα πρόγραμμα σε κάποιον ηλεκτρονικό υπολογιστή του χρήστη, καθώς η πρόσβαση σε αυτές γίνεται μέσω δικτύου. Οι χρήστες μπορούν να έχουν πρόσβαση σε μια διαδικτυακή εφαρμογή μέσω ενός προγράμματος περιήγησης ιστού, όπως το Google Chrome, το Mozilla Firefox ή το Safari.

Για να λειτουργήσει μία διαδικτυακή εφαρμογή, χρειάζεται ένας Διακομιστής Ιστού (Web Server) ένας Διακομιστής Εφαρμογών (Application Server) και μία Βάση Δεδομένων (Database) του Σχήματος 2.1. Οι Διακομιστές Ιστού διαχειρίζονται τις αιτήσεις που λαμβάνουν από τους χρήστες ενώ ο Διακομιστής Εφαρμογών ολοκληρώνει τις αιτήσεις που ζητήθηκαν. Η Βάση Δεδομένων είναι ο χώρος στον οποίο αποθηκεύονται όλες οι απαραίτητες πληροφορίες που χρειάζεται μια διαδικτυακή εφαρμογή.



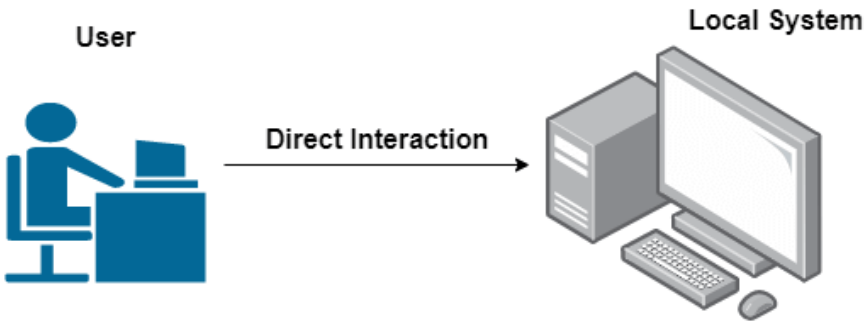
Σχήμα 2.1:Λειτουργία Διαδικτυακής Εφαρμογής

2.3 Αρχιτεκτονική Εφαρμογών

Η Αρχιτεκτονική Λογισμικού είναι ένα υποσύνολο του σχεδιασμού λογισμικού. Καθορίζει πώς θα οργανωθούν και θα συναρμολογηθούν τα σημαντικά στοιχεία του λογισμικού. Πρέπει να καταστεί σαφές ότι η αρχιτεκτονική εφαρμογών δεν περιλαμβάνει την κατασκευή λογισμικού, απλώς σχεδιάζει τη διάταξη των βασικών στοιχείων του λογισμικού. Η αρχιτεκτονική λογισμικού προσπαθεί να διασφαλίσει την αποτελεσματική επικοινωνία μεταξύ αυτών των στοιχείων. Περιλαμβάνει επίσης την εύρεση των περιορισμών που πρέπει να ληφθούν υπόψη κατά την κατασκευή του λογισμικού.

2.4 Αρχιτεκτονική Ενός Επιπέδου

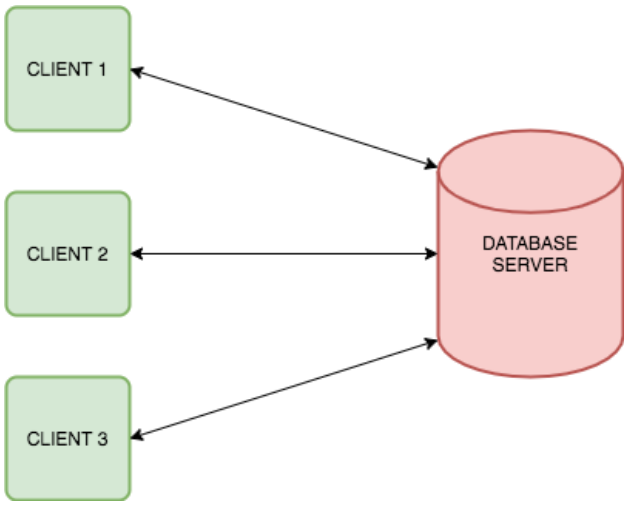
Η αρχιτεκτονική ενός επιπέδου, αναφέρεται σε εκείνο το είδος αρχιτεκτονικής λογισμικού στο οποίο όλα τα απαιτούμενα στοιχεία για τη λειτουργία της εφαρμογής είναι διαθέσιμα στο ίδιο πακέτο. Σημαίνει ότι η διεπαφή χρήστη, η επιχείρηση και τα επίπεδα είναι προσβάσιμα από την εφαρμογή κάτω από την ίδια τοπική μονάδα δίσκου. Τόσο ο πελάτης όσο και ο διακομιστής βρίσκονται στο ίδιο μηχάνημα όπως παρουσιάζεται στο Σχήμα 2.2. Είναι η απλούστερη αρχιτεκτονική εφαρμογών που χρησιμοποιείται αλλά αυτό το επίπεδο μειονεκτεί σε μια διαδικτυακή εφαρμογή, καθώς μπορεί να έχει πρόσβαση σε δεδομένα που είναι διαθέσιμα μόνο σε έναν υπολογιστή ή διακομιστή.



Σχήμα 2.2:1-Επίπεδο Αρχιτεκτονικής

2.5 Αρχιτεκτονική Δύο Επιπέδων

Η αρχιτεκτονική δύο επιπέδων είναι αυτή στην οποία το επίπεδο διεπαφής χρήστη και το επίπεδο βάσης δεδομένων βρίσκονται σε δύο διαφορετικά μηχανήματα όπως αναφέρεται στο Σχήμα 2.3. Αυτό σημαίνει ότι ο πελάτης και ο διακομιστής δεν βρίσκονται στον ίδιο υπολογιστή. Ο πελάτης, που είναι η συσκευή από την πλευρά του χρήστη δίνει τις οδηγίες. Στη συνέχεια, ο διακομιστής που αποθηκεύει όλα τα δεδομένα και τις πληροφορίες καλείται να παράσχει τα απαιτούμενα δεδομένα ή να κάνει αλλαγές στα υπάρχοντα δεδομένα. Το επιχειρηματικό επίπεδο υπάρχει στο μηχάνημα διακομιστή. Τόσο η διεπαφή χρήστη όσο και το επίπεδο παρουσίασης και το επίπεδο βάσης δεδομένων επικοινωνούν μεταξύ τους μέσω του Διαδικτύου, του Πρωτοκόλλου Ελέγχου Μεταφοράς, του Πρωτοκόλλου Διαδικτύου. Είναι εύκολο να διατηρηθεί και να τροποποιηθεί η αρχιτεκτονική εφαρμογών δύο επιπέδων. Ακόμη και η επικοινωνία μεταξύ πελάτη και διακομιστή με τη μορφή αιτήματος και απάντησης είναι πολύ γρήγορη. Αλλά το μειονέκτημα αυτής της αρχιτεκτονικής είναι ότι, εάν ο αριθμός των πελατών υπερβαίνει τη χωρητικότητα?, τότε ο διακομιστής δεν μπορεί να ανταποκριθεί στα αιτήματα που υποβάλλουν οι πελάτες και μειώνει την παραγωγικότητα του. Η εφαρμογή πρέπει επίσης να επανεγκατασταθεί στον υπολογιστή-πελάτη εάν γίνουν κάποιες αλλαγές στην εφαρμογή. Δεδομένου ότι το επιχειρηματικό επίπεδο βρίσκεται στην πλευρά του πελάτη, επομένως ο πελάτης πρέπει να έχει υψηλή επεξεργαστική ισχύ.

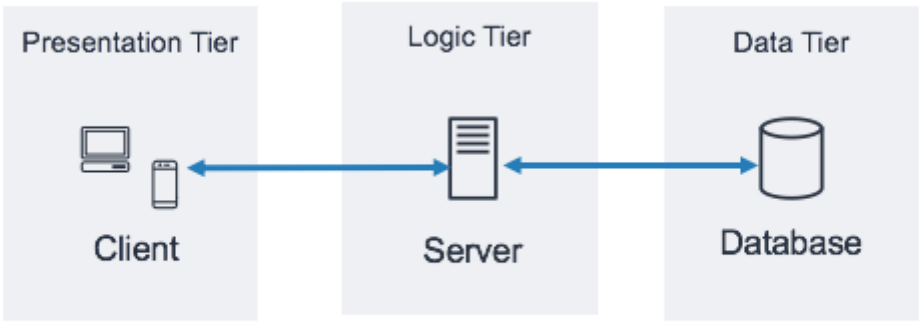


Σχήμα 2.3:2-Επίπεδα Αρχιτεκτονικής

2.6 Αρχιτεκτονική Τριών Επιπέδων

Η αρχιτεκτονική τριών επιπέδων είναι καθιερωμένη αρχιτεκτονική εφαρμογών λογισμικού που οργανώνει τις εφαρμογές σε τρία λογικά και φυσικά επίπεδα υπολογιστών του Σχήματος 2.4:

- **Επίπεδο παρουσίασης** ή διεπαφή χρήστη (UI-User Interface),
- **Επίπεδο εφαρμογής**, όπου γίνεται η επεξεργασία των δεδομένων,
- **Επίπεδο δεδομένων**, όπου αποθηκεύονται και διαχειρίζονται τα δεδομένα της εφαρμογής.



Σχήμα 2.4: 3-Επιπέδων Αρχιτεκτονική

2.6.1 Επίπεδο Παρουσίασης

Το επίπεδο παρουσίασης είναι η διεπαφή χρήστη (User Interface) και το επίπεδο επικοινωνίας της εφαρμογής. Εκεί ο τελικός χρήστης αλληλοεπιδρά με την εφαρμογή. Ο κύριος σκοπός του είναι να

εμφανίζει πληροφορίες και να συλλέγει αιτήματα από τον χρήστη. Αυτό είναι το ανώτατο επίπεδο που μπορεί να εκτελεστεί σε ένα πρόγραμμα περιήγησης ιστού, ως εφαρμογή εγκατεστημένη στον υπολογιστή του ή μια γραφική διεπαφή χρήστη (GUI). Το επίπεδο παρουσίασης ιστού αναπτύσσεται με χρήση Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) και JavaScript.

2.6.2 Επίπεδο Εφαρμογής

Το επίπεδο εφαρμογής, γνωστό και ως λογικό επίπεδο ή μεσαίο επίπεδο, είναι η καρδιά της εφαρμογής. Σε αυτό το επίπεδο, τα αιτήματα που συλλέγονται στο επίπεδο παρουσίασης υποβάλλονται σε επεξεργασία - μερικές φορές σε σχέση με άλλες πληροφορίες στο επίπεδο δεδομένων - χρησιμοποιώντας επιχειρηματική λογική, ένα συγκεκριμένο σύνολο επιχειρηματικών κανόνων. Το επίπεδο εφαρμογής μπορεί επίσης να προσθέσει, να διαγράψει ή να τροποποιήσει δεδομένα στο υποκείμενο επίπεδο δεδομένων. Το επίπεδο εφαρμογής συνήθως αναπτύσσεται χρησιμοποιώντας γλώσσες προγραμματισμού όπως Python, Java, C#, Perl, PHP ή Ruby και επικοινωνεί με το επίπεδο δεδομένων χρησιμοποιώντας κλήσεις Application Programming Interface (API).

2.6.3 Επίπεδο Δεδομένων

Το επίπεδο δεδομένων, που μερικές φορές ονομάζεται επίπεδο βάσης δεδομένων, επίπεδο πρόσβασης δεδομένων ή back-end, είναι το σημείο όπου αποθηκεύονται και διαχειρίζονται οι πληροφορίες που υποβάλλονται σε επεξεργασία από την εφαρμογή. Αυτό μπορεί να είναι ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων με Structured Query Language (SQL) όπως PostgreSQL, MySQL, MariaDB, Oracle, Db2, Informix ή Microsoft SQL Server είτε σε διακομιστή βάσης δεδομένων NoSQL όπως Cassandra, CouchDB ή MongoDB.

2.6.4 Πλεονεκτήματα της αρχιτεκτονικής τριών επιπέδων

Το κύριο πλεονέκτημα της αρχιτεκτονικής τριών επιπέδων είναι ο λογικός και φυσικός διαχωρισμός της λειτουργικότητας. Κάθε επίπεδο μπορεί να εκτελείται σε ξεχωριστό λειτουργικό σύστημα και πλατφόρμα διακομιστή - για παράδειγμα, διακομιστή ιστού, διακομιστή εφαρμογών, διακομιστή βάσης δεδομένων - που ταιριάζει καλύτερα στις λειτουργικές απαιτήσεις του. Και κάθε επίπεδο εκτελείται σε τουλάχιστον ένα αποκλειστικό υλικό διακομιστή ή εικονικό διακομιστή, έτσι ώστε οι υπηρεσίες κάθε επιπέδου να μπορούν να προσαρμοστούν και να βελτιστοποιηθούν χωρίς να επηρεάζονται τα άλλα επίπεδα.

Άλλα οφέλη (σε σύγκριση με την αρχιτεκτονική ενός ή δύο επιπέδων) περιλαμβάνουν:

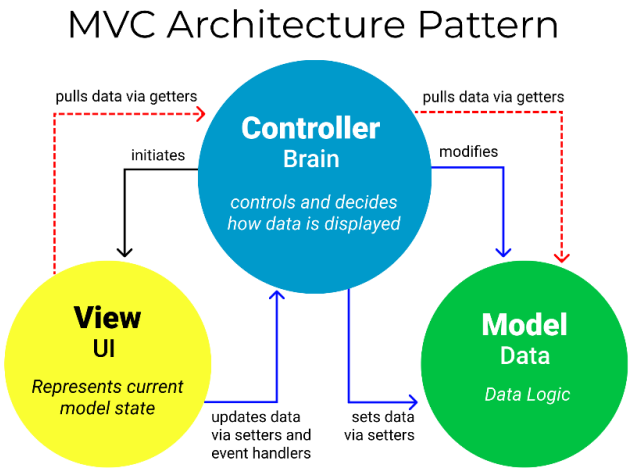
- **Ταχύτερη ανάπτυξη**: Επειδή κάθε επίπεδο μπορεί να αναπτυχθεί ταυτόχρονα από διαφορετικές ομάδες, ένας οργανισμός μπορεί να φέρει την εφαρμογή στην αγορά πιο γρήγορα. Οι προγραμματιστές μπορούν να χρησιμοποιούν τις πιο πρόσφατες και καλύτερες γλώσσες και εργαλεία για κάθε επίπεδο.
- **Βελτιωμένη επεκτασιμότητα**: Οποιαδήποτε βαθμίδα μπορεί να κλιμακωθεί ανεξάρτητα από τις άλλες, όπως απαιτείται
- **Βελτιωμένη αξιοπιστία**: Μια διακοπή λειτουργίας σε ένα επίπεδο είναι λιγότερο πιθανό να επηρεάσει τη διαθεσιμότητα ή την απόδοση των άλλων επιπέδων.
- **Βελτιωμένη ασφάλεια**: Επειδή το επίπεδο παρουσίασης και το επίπεδο δεδομένων δεν μπορούν να επικοινωνήσουν απευθείας, ένα καλά σχεδιασμένο επίπεδο εφαρμογής μπορεί να λειτουργήσει ως εσωτερικό τείχος προστασίας για προστασία και απο άλλες κακόβουλες πράξεις.

2.7 Αρχιτεκτονική Model-View-Controller(MVC)

Το MVC είναι το επικρατέστερο σχεδιαστικό μοτίβο που διαχωρίζει μια εφαρμογή σε τρία κύρια λογικά στοιχεία Model που αντιστοιχεί στο επίπεδο δεδομένων, View που αντιστοιχεί στο επίπεδο παρουσίασης και Controller όπου αντιστοιχεί στο επίπεδο εφαρμογής όπως φαίνεται και στο Σχήμα 2.5. Κάθε αρχιτεκτονικό στοιχείο είναι κατασκευασμένο για να χειρίζεται συγκεκριμένες πτυχές ανάπτυξης μιας εφαρμογής. Απομονώνει την επιχειρηματική λογική και το επίπεδο παρουσίασης μεταξύ τους. Χρησιμοποιήθηκε παραδοσιακά για γραφικές διεπαφές χρήστη (GUIs) για επιτραπέζιους υπολογιστές. Σήμερα, το MVC είναι ένα από τα πιο συχνά χρησιμοποιούμενα βιομηχανικά πρότυπα ανάπτυξης ιστοσελίδων για τη δημιουργία επεκτάσιμων έργων. Χρησιμοποιείται επίσης για το σχεδιασμό εφαρμογών για εφαρμογές κινητών συσκευών.

Τα χαρακτηριστικά που παρέχει το MVC :

- Έναν σαφή διαχωρισμό της επιχειρηματικής λογικής, της λογικής διεπαφής χρήστη και της λογικής εισόδου.
- Έναν πλήρη έλεγχο του HTML και των διευθύνσεων Uniform Resource Locator (URL), γεγονός που καθιστά εύκολο τον σχεδιασμό της αρχιτεκτονικής εφαρμογών ιστού.
- Είναι ένα ισχυρό στοιχείο αντιστοίχισης διευθύνσεων URL χρησιμοποιώντας το οποίο μπορεί να δημιουργηυούν εφαρμογές που έχουν κατανοητές διευθύνσεις URL.
- Υποστηρίζει Test Driven Development (TDD).



Σχήμα 2.5:Αρχιτεκτονική MVC

Το πλαίσιο MVC περιλαμβάνει τα ακόλουθα 3 στοιχεία:

- **Model**
- **View**
- **Controller**

Το στοιχείο **Model** αντιστοιχεί σε όλη τη λογική που σχετίζεται με τα δεδομένα με την οποία εργάζεται ο χρήστης. Αυτό μπορεί να αντιπροσωπεύει είτε τα δεδομένα που μεταφέρονται μεταξύ των στοιχείων Προβολή και Ελεγκτή είτε οποιαδήποτε άλλα δεδομένα που σχετίζονται με τη λογική της επιχείρησης. Μπορεί να προσθέσει ή να ανακτήσει δεδομένα από τη βάση δεδομένων. Απαντά στο αίτημα του ελεγκτή επειδή ο ελεγκτής δεν μπορεί να αλληλεπιδράσει μόνος του με τη βάση δεδομένων. Το μοντέλο αλληλεπιδρά με τη βάση δεδομένων και δίνει τα απαιτούμενα δεδομένα πίσω στον ελεγκτή.

Το στοιχείο **View** χρησιμοποιείται για όλη τη λογική διεπαφής χρήστη της εφαρμογής. Δημιουργεί μια διεπαφή χρήστη για τον χρήστη. Οι προβολές δημιουργούνται από τα δεδομένα που συλλέγονται από το στοιχείο model, αλλά αυτά τα δεδομένα δεν λαμβάνονται απευθείας αλλά μέσω του ελεγκτή. Αλληλεπιδρά μόνο με τον ελεγκτή.

Ο **Controller** είναι το στοιχείο που επιτρέπει τη διασύνδεση μεταξύ των προβολών και του μοντέλου, επομένως λειτουργεί ως ενδιάμεσος. Ο controller δεν χρειάζεται να ανησυχεί για τον χειρισμό της λογικής δεδομένων, απλώς λέει στο μοντέλο τι να κάνει. Επεξεργάζεται όλη την επιχειρηματική λογική και τα εισερχόμενα αιτήματα, χειρίζεται δεδομένα χρησιμοποιώντας το στοιχείο Model και αλληλεπιδρά με την προβολή για να αποδώσει το τελικό αποτέλεσμα.

2.7.1 Πλεονεκτήματα και μειονεκτήματα της αρχιτεκτονικής MVC

Πλεονεκτήματα:

- Ο κώδικας διατηρείται εύκολα και μπορεί να επεκταθεί εύκολα.
- Το MVC Model μπορεί ο χρήστης να το ελέγχει ξεχωριστά.
- Τα στοιχεία του MVC μπορούν να αναπτυχθούν ταυτόχρονα.
- Μειώνει την πολυπλοκότητα διαιρώντας μια εφαρμογή σε τρεις ενότητες. Μοντέλο, προβολή και ελεγκτής.
- Λειτουργεί καλά για εφαρμογές Ιστού που υποστηρίζονται από μεγάλες ομάδες σχεδιαστών και προγραμματιστών.
- Αυτή η αρχιτεκτονική βοηθά στον έλεγχο των στοιχείων ανεξάρτητα, καθώς όλες οι κλάσεις και τα αντικείμενα είναι ανεξάρτητα μεταξύ τους
- Search Engine Optimization (SEO) Friendly.

Μειονεκτήματα:

- Είναι δύσκολο να διαβαστεί, να αλλαχτεί, να δοκιμαστεί και να επαναχρησιμοποιηθεί αυτό το μοντέλο
- Δεν είναι κατάλληλο για κατασκευή μικρών εφαρμογών
- Αυξημένη πολυπλοκότητα και αναποτελεσματικότητα δεδομένων

2.8 Εισαγωγή στα APIs και την Διασύνδεση Εφαρμογών

Ένα API είναι ένα σύνολο κώδικα προγραμματισμού που επιτρέπει τη μετάδοση δεδομένων μεταξύ ενός προϊόντος λογισμικού και ενός άλλου. Περιλαμβάνει επίσης τους όρους αυτής της ανταλλαγής δεδομένων. Η διεπαφή προγραμματισμού εφαρμογής πρέπει να διακρίνεται σαφώς από τη διεπαφή χρήστη. Η διεπαφή χρήστη δέχεται δεδομένα από τους χρήστες, τα προωθεί στην εφαρμογή για επεξεργασία και επιστρέφει τα αποτελέσματα στον χρήστη. Το API δεν αλληλεπιδρά με τον χρήστη, αλλά επεξεργάζεται τα δεδομένα που λαμβάνονται από μια λειτουργική μονάδα προγράμματος και μεταδίδει τα αποτελέσματα πίσω στην άλλη λειτουργική μονάδα.

2.8.1 Η λειτουργικότητα του API

Η αρχή λειτουργίας ενός API εκφράζεται συνήθως μέσω της επικοινωνίας αιτήματος-απόκρισης μεταξύ ενός πελάτη και ενός διακομιστή. Ο πελάτης είναι οποιαδήποτε εφαρμογή front-end με την οποία αλληλεπιδρά ένας χρήστης. Ο διακομιστής είναι υπεύθυνος για τις λειτουργίες λογικής υποστήριξης και βάσης δεδομένων. Σε αυτό το σενάριο, ένα API λειτουργεί ως μεσαίο επίπεδο μεταξύ του πελάτη και του διακομιστή, καθιστώντας δυνατή την αποστολή αιτημάτων δεδομένων και απαντήσεων.

2.8.2 Τα Στοιχεία ενός API

Οι διεπαφές προγραμματισμού εφαρμογών αποτελούνται από δύο στοιχεία:

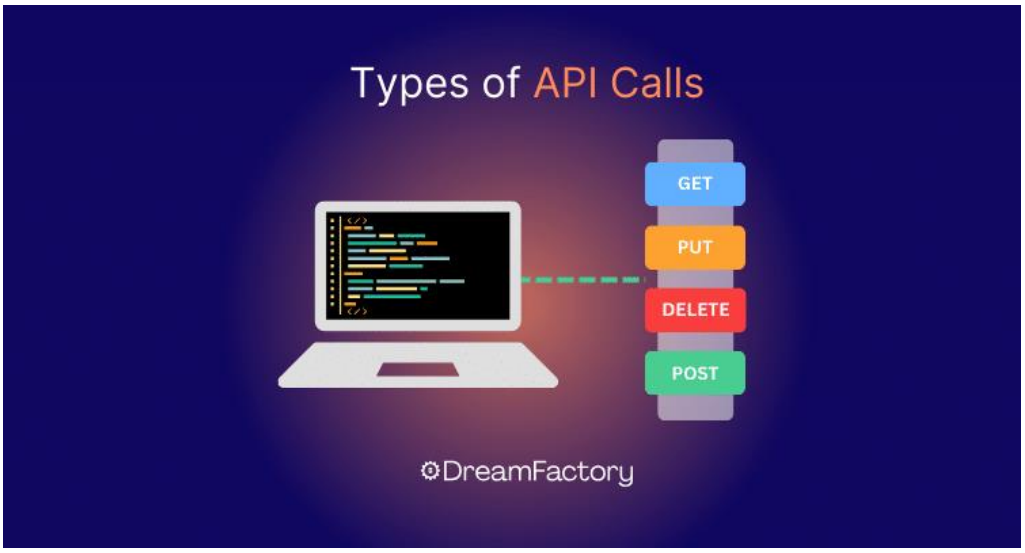
- **τεχνική προδιαγραφή**: που περιγράφει τις επιλογές ανταλλαγής δεδομένων μεταξύ λύσεων με την προδιαγραφή να γίνεται με τη μορφή αίτησης για πρωτόκολλα επεξεργασίας
- **παράδοσης δεδομένων**: διεπαφή λογισμικού, γραμμένο σύμφωνα με τις προδιαγραφές που το αντιπροσωπεύουν.

Το λογισμικό που χρειάζεται πρόσβαση σε πληροφορίες ή λειτουργικότητα από άλλο λογισμικό καλεί το API του καθορίζοντας τις απαιτήσεις για το πώς πρέπει να παρέχονται τα δεδομένα/λειτουργικότητα. Το άλλο λογισμικό επιστρέφει δεδομένα/λειτουργικότητα που ζητήθηκε από την προηγούμενη εφαρμογή. Η διεπαφή μέσω της οποίας επικοινωνούν αυτές οι δύο εφαρμογές είναι αυτό που καθορίζει το API.

2.8.3 Κλήσεις συνάρτησης API

Κάθε API περιέχει και υλοποιείται από κλήσεις συναρτήσεων – δηλώσεις γλώσσας που ζητούν από λογισμικό να εκτελεί συγκεκριμένες ενέργειες και υπηρεσίες. Οι κλήσεις συναρτήσεων αποτελούνται από ρήματα (π.χ. BEGIN, GET, DELETE, κ.λπ.) του Σχήματος 2.6 και ουσιαστικά (πχ, Data, Access, κ.λπ.) που επιτρέπουν σε ένα μηχάνημα να καταλάβει τι πρέπει να κάνει στη συνέχεια όπως οι παρακάτω δύο ενέργειες:

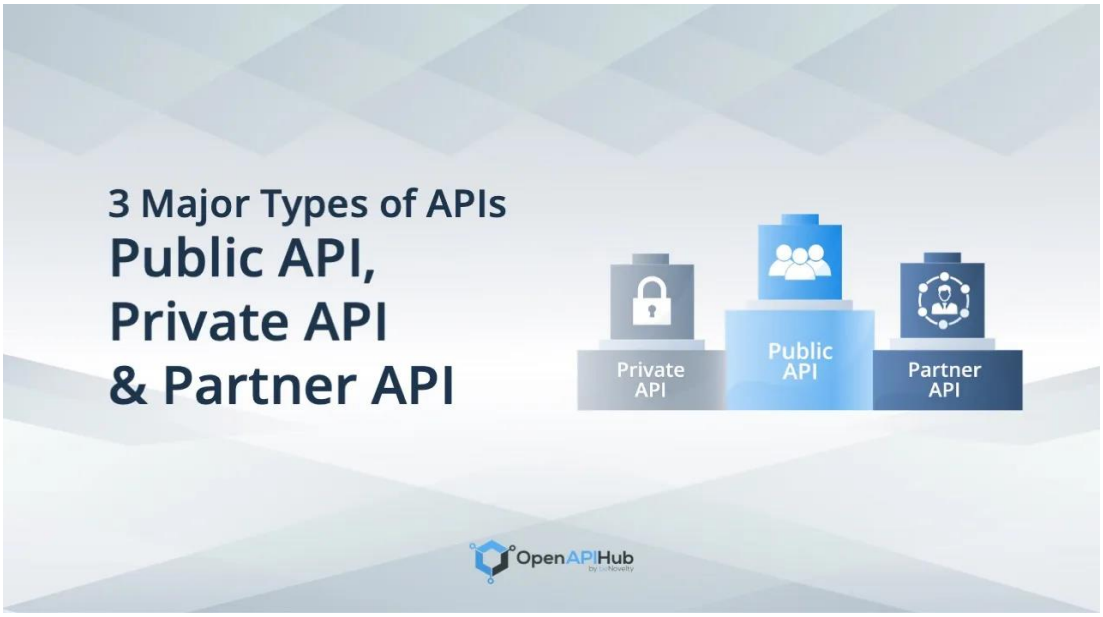
- Έναρξη ή ολοκλήρωση μιας συνεδρίας,
- Επαναφορά ή ανάκτηση αντικειμένων από διακομιστή



Σχήμα 2.6: Τύποι Κλήσεων API

2.8.4 Τύποι API

Υπάρχουν διάφοροι τύποι API όπως παρουσιάζεται και στο Σχήμα 2.7 που μπορούν να κατηγοριοποιηθούν με βάση τους τρόπους με τους οποίους είναι διαθέσιμα για χρήση και σύμφωνα με τους αρχικούς σχεδιαστικούς τους σκοπούς.



Σχήμα 2.7: Τύποι API

Ιδιωτικά API: Αυτές οι διεπαφές λογισμικού εφαρμογών έχουν σχεδιαστεί για τη βελτίωση των οργανωτικών λύσεων και υπηρεσιών. Οι εσωτερικοί προγραμματιστές ή εργολάβοι μπορούν να χρησιμοποιήσουν αυτά τα API για να ενσωματώσουν συστήματα ή εφαρμογές πληροφορικής μιας εταιρείας, καθώς και να δημιουργήσουν νέα συστήματα ή εφαρμογές που απευθύνονται σε πελάτες αξιοποιώντας υπάρχοντα συστήματα. Ακόμα κι αν οι εφαρμογές είναι δημόσια προσβάσιμες, η ίδια η διεπαφή παραμένει διαθέσιμη μόνο για όσους εργάζονται απευθείας με τον εκδότη API. Η ιδιωτική στρατηγική επιτρέπει σε μια εταιρεία να ελέγχει πλήρως τη χρήση του API.

API συνεργατών: Αυτός ο τύπος API προωθείται ανοιχτά αλλά μοιράζεται με επιχειρηματικούς συνεργάτες που έχουν υπογράψει συμφωνία με τον εκδότη. Η κοινή περίπτωση χρήσης για συνεργαζόμενα API είναι η ενοποίηση λογισμικού μεταξύ δύο μερών. Μια εταιρεία που παρέχει στους συνεργάτες πρόσβαση σε δεδομένα ή δυνατότητες επωφελείται από επιπλέον ροές εσόδων. Ταυτόχρονα, μπορεί να παρακολουθεί πώς χρησιμοποιούνται τα εκτεθειμένα ψηφιακά στοιχεία, να διασφαλίζει εάν οι λύσεις τρίτων που χρησιμοποιούν τα API τους παρέχουν αξιοπρεπή εμπειρία χρήστη και να διατηρεί την εταιρική ταυτότητα στις εφαρμογές τους.

Δημόσια API: Γνωστά ως προγραμματιστές ή εξωτερικά, αυτά τα API είναι διαθέσιμα για οποιονδήποτε τρίτο προγραμματιστή. Ένα δημόσιο πρόγραμμα API επιτρέπει την αύξηση της αναγνωρισιμότητας της επωνυμίας και τη λήψη πρόσθετης πηγής εισοδήματος όταν εκτελείται σωστά.

Υπάρχουν δύο τύποι δημόσιων API – ανοιχτά (δωρεάν) και εμπορικά:

- **Τα ανοιχτά (δώρεαν) δημόσια API**, όπως προτείνει ο ορισμός του Open API, είναι αυτά με όλες τις λειτουργίες δημόσιες και διαθέσιμα για χρήση χωρίς περιοριστικούς όρους και προϋποθέσεις. Για παράδειγμα, είναι δυνατή η δημιουργία μιας εφαρμογής που χρησιμοποιεί το API χωρίς ρητή έγκριση από τον προμηθευτή API ή υποχρεωτικά τέλη αδειοδότησης. Ο ορισμός αναφέρει επίσης ότι η περιγραφή του API και οποιαδήποτε σχετική τεκμηρίωση πρέπει να είναι ανοιχτά και διαθέσιμα. Επιπλέον, αυτά τα API μπορούν να χρησιμοποιηθούν ελεύθερα για τη δημιουργία και τη δοκιμή εφαρμογών.
- **Εμπορικά**, οι εμπορικοί χρήστες πληρώνουν τέλη συνδρομής ή χρησιμοποιούν API σε βάση πληρωμής. Μια δημοφιλής προσέγγιση μεταξύ των εκδοτών είναι η προσφορά δωρεάν δοκιμών, ώστε οι χρήστες να μπορούν να αξιολογούν τα API πριν αγοράσουν συνδρομές

2.9 Front-End Τεχνολογίες Ανάπτυξης Διαδικτυακής Εφαρμογής

Στο πλαίσιο της Ανάπτυξης Διαδικτυακής Εφαρμογής, το Front-End μιας εφαρμογής Ιστού αναφέρεται στο τμήμα της εφαρμογής με το οποίο οι χρήστες αλληλοεπιδρούν άμεσα. Περιλαμβάνει όλα όσα βλέπουν, αγγίζουν οι χρήστες στις οθόνες τους, συμπεριλαμβανομένου του σχεδιασμού, της διάταξης, του περιεχομένου και των στοιχείων διεπαφής χρήστη, όπως κουμπιά, φόρμες, μενού και άλλα.

2.9.1 Βασικές Τεχνολογίες Front-End: HTML, CSS και JavaScript

Το Front-End αποτελείται από πολλές διαφορετικές γλώσσες και βιβλιοθήκες. Ενώ αυτές διαφέρουν από εφαρμογή σε εφαρμογή, υπάρχουν μόνο μερικές γενικές γλώσσες κατανοητές από όλα τα προγράμματα περιήγησης Ιστού. Αυτές οι τρεις κύριες γλώσσες κωδικοποίησης Front-End είναι η HTML, η CSS και η JavaScript. Μαζί, δημιουργούν έναν υποκείμενο σκελετό που χρησιμοποιούν τα προγράμματα περιήγησης ιστού για την απόδοση των ιστοσελίδων με τις οποίες αλληλοεπιδρούμε καθημερινά. Όλες οι άλλες βιβλιοθήκες και το Front-End βασίζονται σε αυτές τις τρεις κύριες γλώσσες, γεγονός που τις καθιστά απαραίτητες δεξιότητες για κάθε προγραμματιστή Front-End.

2.9.2 HTML

Η HTML ή αλλιώς HyperText Markup Language είναι μία γλώσσα σήμανσης που χρησιμοποιείται ως βασική γλώσσα με την οποία δημιουργείται η δόμηση των σελίδων του Ιστού. Η HTML παρέχει την δομή σε μία ιστοσελίδα όπως παρουσιάζεται στο Σχήμα 2.8 έτσι ώστε να την κάνει προσβάσιμη στου χρήστες του Διαδικτύου. Η HTML γράφεται υπό την μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείνονται μέσα σε σύμβολα < >, στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες της HTML συνήθως λειτουργούν σαν ζεύγη (<a>) με την πρώτη ετικέτα να ονομάζεται ετικέτα έναρξης και με την τελευταία λήξης. Ενδιάμεσα από τις ετικέτες οι σχεδιαστές μπορούν να τοποθετήσουν κείμενα, πίνακες, εικόνες.

Simple HTML Document Example

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
</head>
<body>
    <!-- Content goes here -->
</body>
</html>
```

Σχήμα 2.8: Η δομή της HTML

Η HTML είναι η κύρια γλώσσα που χρησιμοποιείται για την δημιουργία ιστοσελίδων διότι:

- Είναι απλή και εύκολη στην εκμάθηση και χρήση
- Έχει υποστήριξη από όλες τις πλατφόρμες ανεξαρτήτου λογισμικού συστήματος
- Συνδυάζεται και με άλλες γλώσσες προγραμματισμού όπως CSS και JavaScript.

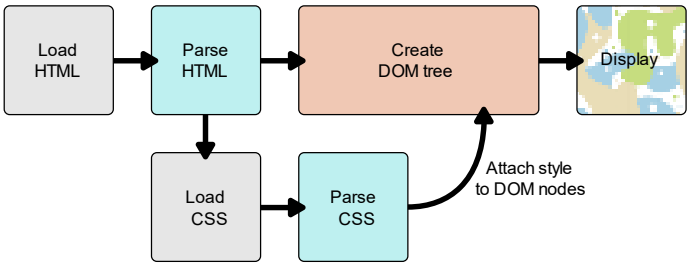
2.9.3 CSS

Το CSS είναι μια γλώσσα φύλλου που χρησιμοποιείται για να περιγράψει την εμφάνιση και τη μορφοποίηση ενός εγγράφου γραμμένο στην γλώσσα HTML. Είναι απαραίτητο για το σχεδιασμό ιστοσελίδων, καθώς επιτρέπει στους προγραμματιστές να ελέγχουν τα τυπογραφικά χαρακτηριστικά, τις ιδιότητες χρώματος και τη συνολική διάταξη των ιστοσελίδων. Το CSS μπορεί να χρησιμοποιηθεί εσωτερικά μέσα σε ένα στοιχείο, ενσωματωμένα μέσα σε στοιχεία HTML ή εξωτερικά μέσω συνδεδεμένων αρχείων CSS. Το CSS λειτουργεί εφαρμόζοντας στυλ σε στοιχεία μιας ιστοσελίδας. Αυτά τα στυλ γράφονται σε ένα αρχείο CSS, το οποίο στη συνέχεια συνδέεται με ένα αρχείο HTML. Αυτός ο διαχωρισμός μεταξύ στυλ και δομής επιτρέπει στους προγραμματιστές να εφαρμόζουν συνεπή στυλ σε πολλές σελίδες, προσαρμόζοντας την εμφάνιση σε διαφορετικά μεγέθη οθόνης και τύπους συσκευών.

Η βάση του CSS βρίσκεται στη σύνταξή του, η οποία αποτελείται από δύο κύρια μέρη: επιλογής και δηλώσεις. Οι επιλογής υποδεικνύουν το στοιχείο HTML που θα διαμορφωθεί και οι δηλώσεις καθορίζουν το στυλ που θα εφαρμοστεί. Μια δήλωση χωρίζεται περαιτέρω σε μια ιδιότητα (τι να διαμορφώσετε) και μια τιμή (πώς να την διαμορφώσετε).

Τα πλεονεκτήματα του CSS είναι:

- **Ταχύτητα σελίδας**: Προσφέρει μεγαλύτερη ταχύτητα στην σελίδα από άλλες τεχνολογίες. Με τη βοήθεια του CSS, μπορεί να εφαρμοσθεί σε όλες τις εμφανίσεις συγκεκριμένων ετικετών σε έγγραφα HTML.
- **Καλύτερη εμπειρία χρήστη**: Το CSS κάνει μια ιστοσελίδα πολύ ελκυστική στα μάτια. Επίσης, το CSS το καθιστά φιλικό προς τον χρήστη. Όταν το κουμπί ή το κείμενο είναι σε σωστή μορφή, βελτιώνει την εμπειρία του χρήστη.
- **Ταχύτερος χρόνος ανάπτυξης**: Με τη βοήθεια του CSS, μπορεί να καθορισθεί η μορφή και το στυλ των πολλαπλών σελίδων σε μια συμβολοσειρά κώδικα.
- **Εύκολες αλλαγές μορφοποίησης**: Στο CSS, αν χρειαστεί να πραγματοποιηθούν αλλαγές στη μορφή χρειάζεται μόνο να αλλάξει η μορφή μιας σελίδας που θα εφαρμόζεται αυτόματα στις άλλες σελίδες του CSS. Δεν χρειάζεται να διορθωθούν μεμονωμένες σελίδες σε ένα φύλλο στυλ CSS. Εάν γίνει διόρθωση σε ένα φύλλο στυλ CSS, θα ενημερώσει αυτόματα το άλλο φύλλο στυλ CSS.
- **Συμβατότητα:** Η συμβατότητα είναι πολύ σημαντική στη σημερινή εποχή. Εάν δημιουργηθεί οποιαδήποτε ιστοσελίδα, θα πρέπει να είναι πολύ εύχρηστη και φιλική προς το χρήστη. Το CSS συνεργάζεται με το HTML για να ανταποκρίνεται ο σχεδιασμός της ιστοσελίδας.



Σχήμα 2.9: Λειτουργία CSS

2.9.4 JavaScript

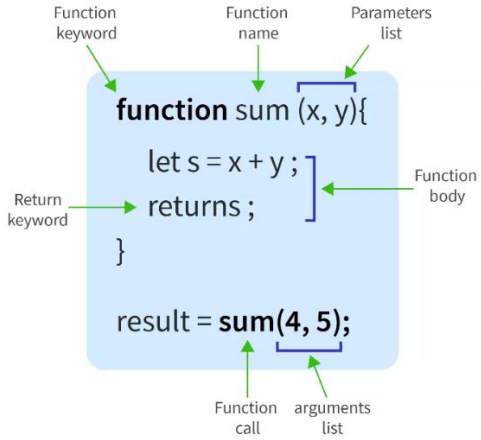
Η JavaScript (js) είναι μια ελαφριά αντικειμενοστραφής γλώσσα προγραμματισμού που χρησιμοποιείται από πολλούς ιστότοπους για τη δημιουργία σεναρίων στις ιστοσελίδες. Είναι μια

ερμηνευμένη, πλήρης γλώσσα προγραμματισμού που επιτρέπει τη δυναμική διαδραστικότητα σε ιστότοπους όταν εφαρμόζεται σε ένα έγγραφο HTML. Εισήχθη το έτος 1995 για την προσθήκη προγραμμάτων στις ιστοσελίδες στο πρόγραμμα περιήγησης Netscape Navigator. Έκτοτε, έχει υιοθετηθεί από όλα τα άλλα προγράμματα περιήγησης ιστού με γραφικά. Με τη JavaScript, οι χρήστες μπορούν να δημιουργήσουν σύγχρονες εφαρμογές Ιστού για να αλληλεπιδρούν οι χρήστες απευθείας χωρίς να φορτώνουν ξανά τη σελίδα κάθε φορά. Ο παραδοσιακός ιστότοπος χρησιμοποιεί JavaScript για να παρέχει διάφορες μορφές διαδραστικότητας και απλότητας.

Η σύνταξη της JavaScript είναι παρόμοια με αρκετές γλώσσες προγραμματισμού όπως η C και η Java. Χρησιμοποιεί λέξεις κλειδιά, μεταβλητές και συναρτήσεις για να δημιουργεί δηλώσεις (statements) και εκφράσεις (expressions). Μία δήλωση είναι μια γραμμή κώδικα που εκτελεί μία συγκεκριμένη ενέργεια ενώ μία έκφραση είναι μία γραμμή κώδικα που αξιολογείται σε μία τιμή.

Μία από τις βασικές δομές της JavaScript είναι η μεταβλητή. Οι μεταβλητές χρησιμοποιούνται για την αποθήκευση και τον χειρισμό δεδομένων. Δηλώνονται χρησιμοποιώντας τη λέξη-κλειδί "var", "let" ή "const". Η διαφορά μεταξύ "var", "let" και "const" είναι το εύρος και η δυνατότητα εκ νέου εκχώρησης της τιμής. Το "Var" είναι το εύρος συνάρτησης (είναι προσβάσιμο μόνο εντός της συνάρτησης στην οποία έχουν δηλωθεί), το "let" είναι το εύρος μπλοκ (είναι προσβάσιμο μόνο εντός του μπλοκ κώδικα στο οποίο έχουν δηλωθεί) και το "const" είναι το μπλοκ εύρος και δεν μπορεί να ανατεθεί εκ νέου (αν προσπαθήσετε να εκχωρήσετε ξανά μια τιμή σε μια μεταβλητή που δηλώνεται με "const", θα λάβετε ένα σφάλμα).

Μια άλλη σημαντική δομή της JavaScript είναι η συνάρτηση (function) όπως βλέπουμε στο Σχήμα 2.10. Οι συναρτήσεις είναι μπλοκ κώδικα που μπορούν να εκτελεστούν πολλές φορές. Δηλώνονται χρησιμοποιώντας τη λέξη κλειδί "function" ακολουθούμενη από ένα όνομα και ένα σύνολο παρενθέσεων.



Σχήμα 2.10: Μορφή Συνάρτησης JavaScript

Η JavaScript χρησιμοποιείται κυρίως για τους παρακάτω λόγους:

- **Εξοικονομεί χρόνο και bandwidth**: Ανεξάρτητα από το πού φιλοξενείται JavaScript, εκτελείται πάντα στο περιβάλλον πελάτη για εξοικονόμηση bandwidth και για γρήγορη διαδικασία εκτέλεσης.
- **Εύκολη αποστολή αιτημάτων HTTP**: Στο JavaScript, το XMLHttpRequest είναι ένα σημαντικό αντικείμενο που σχεδιάστηκε από τη Microsoft. Οι κλήσεις αντικειμένου πραγματοποιούνται από το XMLHttpRequest ως ασύγχρονο αίτημα HTTP στον διακομιστή για μεταφορά των δεδομένων και στις δύο πλευρές χωρίς επαναφόρτιση της σελίδας.
- **Συμβατό για όλα τα προγράμματα περιήγησης**: Το μεγαλύτερο πλεονέκτημα της JavaScript έχει τη δυνατότητα να υποστηρίζει όλα τα σύγχρονα προγράμματα περιήγησης και να παράγει ένα ισοδύναμο αποτέλεσμα.
- **Υποστήριξη Κοινότητας**: Οι παγκόσμιες εταιρείες υποστηρίζουν την ανάπτυξη της κοινότητας δημιουργώντας σημαντικά έργα συνήθως όπως πλαίσια (framework). Ένα παράδειγμα είναι η Google (δημιουργήθηκε το framework Angular) ή το Facebook (δημιουργήθηκε το framework React.js).

2.9.5 ASP.NET Core Razor Pages

Η ανάπτυξη διεπαφής στο ASP.NET Core MVC περιλαμβάνει τη δημιουργία ανταποκρινόμενων και οπτικά ελκυστικών διεπαφών χρήστη χρησιμοποιώντας HTML, CSS και JavaScript. Οι προγραμματιστές αξιοποιούν το HTML για τη δομή, το CSS για το στυλ και το JavaScript για διαδραστικότητα και δυναμική συμπεριφορά από την πλευρά του πελάτη.

Το ASP.NET Core MVC χρησιμοποιεί σελίδες και προβολές Razor για τη δημιουργία δυναμικού περιεχομένου ιστού. Οι προγραμματιστές χρησιμοποιούν τη σύνταξη Razor (C#) στη σήμανση HTML για να ενσωματώσουν λογική από την πλευρά του διακομιστή, να έχουν πρόσβαση σε δεδομένα μοντέλου και να αποδώσουν δυναμικό περιεχόμενο. Οι Σελίδες Razor παρέχουν μια βελτιωμένη προσέγγιση για τη δημιουργία ιστοσελίδων, ενώ οι Προβολές προσφέρουν μεγαλύτερη ευελιξία και επιλογές προσαρμογής.

Αρχιτεκτονική MVC στο ASP.NET Core

Η αρχιτεκτονική MVC στο ASP.NET Core διαχωρίζει μια εφαρμογή σε τρία στοιχεία: Μοντέλα, Προβολές και Ελεγκτές. Αυτό το μοτίβο βοηθά στον διαχωρισμό των ανησυχιών. Σε αυτό το μοτίβο, τα αιτήματα των χρηστών δρομολογούνται σε έναν ελεγκτή. Ένας ελεγκτής καλεί το μοντέλο για να εκτελέσει ενέργειες χρήστη ή να ανακτήσει δεδομένα. Στη συνέχεια, ο ελεγκτής μεταβιβάζει αυτό το μοντέλο σε μια προβολή και επιστρέφεται στον χρήστη.

MODEL

Στο ASP.NET Core MVC, το μοντέλο αντιπροσωπεύει τα δεδομένα και την επιχειρηματική λογική της εφαρμογής. Οι προγραμματιστές ορίζουν κατηγορίες μοντέλων για να ενσωματώσουν οντότητες δεδομένων, να ορίσουν σχέσεις και να εφαρμόσουν επιχειρηματικούς κανόνες. Τα μοντέλα αλληλεπιδρούν με τη βάση δεδομένων μέσω του Entity Framework Core ή άλλων βιβλιοθηκών πρόσβασης δεδομένων.

VIEW

Οι προβολές στο ASP.NET Core MVC είναι υπεύθυνες για την παρουσίαση της διεπαφής χρήστη και την απόδοση δυναμικού περιεχομένου στον πελάτη. Οι προγραμματιστές δημιουργούν πρότυπα προβολής χρησιμοποιώντας τη σύνταξη Razor όπως αναφέρεται στο Σχήμα 2.11, η οποία συνδυάζει τη σήμανση HTML με κώδικα C# για τη δημιουργία δυναμικών ιστοσελίδων.

```
<ul>
  @foreach (var p in products) {
     <li>
         @p.ProductName

         @if (p.UnitsInStock == 0) {
            @: (Out of stock!)
         }
         else if (p.UnitsInStock < 4) {
            @: (Only @p.UnitsInStock left!)
         }
     </li>
  }
</ul>
```

Σχήμα 2.11: Ενσωμάτωση C# σε HTML

CONTROLLER

Οι ελεγκτές λειτουργούν ως ενδιάμεσοι μεταξύ του μοντέλου και των στοιχείων προβολής στο ASP.NET Core MVC. Διαχειρίζονται αιτήματα χρηστών, εκτελούν επιχειρηματική λογική και ενορχηστρώνουν λειτουργίες ανάκτησης και χειρισμού δεδομένων. Οι ελεγκτές λαμβάνουν δεδομένα από τον χρήστη μέσω του προγράμματος περιήγησης, τα επεξεργάζονται και επιστρέφουν μια κατάλληλη απόκριση, όπως απόδοση μιας προβολής ή ανακατεύθυνση σε άλλη σελίδα.

2.9.6 Bootsrap

Το Bootstrap είναι ένα δωρεάν open source πλαίσιο (framework) ανάπτυξης Front-End για την δημιουργία ιστοσελίδων και εφαρμογών ιστού. Το Bootsrap παρέχει μια συλλογή σύνταξης για σχέδια προτύπων.

Ως πλαίσιο, το Bootstrap περιλαμβάνει τα βασικά στοιχεία για την ανταποκρινόμενη ανάπτυξη εφαρμογών, επομένως οι προγραμματιστές χρειάζεται μόνο να εισάγουν τον κώδικα σε ένα προκαθορισμένο σύστημα πλέγματος (grid system). Το πλαίσιο Bootsrap έχει βασιστεί στην γλώσσα σήμανσης HTML, CSS και JavaScript. Οι προγραμματιστές που χρησιμοποιούν το Bootsrap μπορούν να δημιουργούν πολύ πιο γρήγορα εφαρμογές ιστού χωρίς να ξοδεύουν χρόνο για βασικές εντολές και συναρτήσεις.

Τα πλεονεκτήματα του Bootsrap είναι:

- **Εύκολο στην χρήση**: Ένας από τους λόγους για τους οποίους το Bootstrap είναι τόσο δημοφιλές μεταξύ των προγραμματιστών και των σχεδιαστών ιστού είναι ότι έχει μια απλή δομή αρχείων. Τα αρχεία του είναι μεταγλωττισμένα για εύκολη πρόσβαση και απαιτούνται μόνο βασικές γνώσεις HTML, CSS και JS για να τα τροποποιήθουν.
- **Ανταποκρινόμενο Πλέγμα** (Responsive Grid): Το Bootstrap έρχεται με ένα προκαθορισμένο σύστημα πλέγματος. Το σύστημα πλέγματος αποτελείται από γραμμές και στήλες, επιτρέποντάς να δημιουργεί ένα πλέγμα μέσα στο υπάρχον αντί να εισάγονται ερωτήματα πολυμέσων στο αρχείο CSS.
- **Συμβατότητα:** Το Bootstrap έχει σχεδιαστεί για να διασφαλίζει τη συμβατότητα μεταξύ προγραμμάτων περιήγησης, ελαχιστοποιώντας τα προβλήματα σε διαφορετικά προγράμματα περιήγησης.
- **Προκατασκευασμένα εξαρτήματα και πρότυπα:** Το Bootstrap συνοδεύεται από ένα πλούσιο σύνολο προκατασκευασμένων στοιχείων (components) και προτύπων (templates), επιταχύνοντας τη διαδικασία ανάπτυξης.

2.9.7 AngularJS

Το AngularJS είναι ένα δομικό πλαίσιο για δυναμικές εφαρμογές ιστού. Επιτρέπει να χρησιμοποιείται η HTML ως γλώσσα προτύπου και να επεκτείνεται η σύνταξη της HTML για να εκφράζονται τα στοιχεία της εφαρμογής καθαρά και συνοπτικά. Η σύνδεση δεδομένων και το Dependency Injection του AngularJS εξαλείφουν μεγάλο μέρος του κώδικα που διαφορετικά θα έπρεπε να γραφτεί. Όλα αυτά συμβαίνουν μέσα στο πρόγραμμα περιήγησης, καθιστώντας το ιδανικό συνεργάτη με οποιαδήποτε τεχνολογία διακομιστή.

Η αναντιστοιχία σύνθετης αντίστασης μεταξύ δυναμικών εφαρμογών και στατικών εγγράφων επιλύεται συχνά με:

- **Βιβλιοθήκη** - μια συλλογή λειτουργιών που είναι χρήσιμες κατά τη σύνταξη εφαρμογών ιστού. Ο κώδικας είναι υπεύθυνος και καλείται στη βιβλιοθήκη όταν αυτό κρίνεται σκόπιμο
- **Πλαίσια** - μια συγκεκριμένη υλοποίηση μιας διαδικτυακής εφαρμογής, όπου ο κώδικάς συμπληρώνει τις λεπτομέρειες. Το πλαίσιο είναι υπεύθυνο και καλεί τον κώδικά όταν χρειάζεται κάτι συγκεκριμένο για την εφαρμογή

Το AngularJS ακολουθεί μια άλλη προσέγγιση. Προσπαθεί να ελαχιστοποιήσει την αναντιστοιχία σύνθετης αντίστασης μεταξύ του εγγράφου HTML και του τι χρειάζεται μια εφαρμογή δημιουργώντας νέες κατασκευές HTML. Το AngularJS μαθαίνει στο πρόγραμμα περιήγησης νέα σύνταξη μέσω μιας κατασκευής που ονομάζεται οδηγίες.

Τα παραδείγματα περιλαμβάνουν:

- Δομές ελέγχου Document Object Model (DOM) για επανάληψη, εμφάνιση και απόκρυψη DOM Fragments
- Υποστήριξη για φόρμες και επικύρωση φορμών
- Προσάρτηση νέας συμπεριφοράς σε στοιχεία DOM, όπως ο χειρισμός συμβάντων DOM
- Ομαδοποίηση HTML σε επαναχρησιμοποιήσιμα στοιχεία
- Data binding

Το AngularJS χρησιμοποιεί Javascript και HTML για την υλοποίηση του MVC, όπου η HTML φροντίζει το τμήμα προβολής, ενώ η Javascript διαχειρίζεται το τμήμα μοντέλου και ελεγκτή. Στο τέλος, το AngularJS μεταγλωττίζει το μη μεταγλωττισμένο πρότυπο HTML και τις βασικές οδηγίες στο πρόγραμμα περιήγησης. Οι οδηγίες συνήθως βοηθούν στη σύζευξη των τιμών των πεδίων εισαγωγής, όπως αυτές στο πεδίο κειμένου, με το αντίστοιχο στοιχείο HTML.

2.9.8 VueJS

Το Vue JS, που συνήθως αναφέρεται ως Vue, είναι ένα πλαίσιο ανοιχτού κώδικα JavaScript που δημιουργήθηκε από τον Evan You το 2014 ως εναλλακτική λύση σε βαρύτερα πλαίσια όπως το AngularJS και το React. Το Vue συνδυάζει προσεγγίσεις που επηρεάζονται από το Angular και βελτιστοποιημένες δυνατότητες για διεπαφή στο front-end μέρος για την ανάπτυξη εφαρμογών. Η βασική βιβλιοθήκη του Vue εστιάζει μόνο στο επίπεδο προβολής και έχει σχεδιαστεί για να υιοθετείται σταδιακά σε έργα.

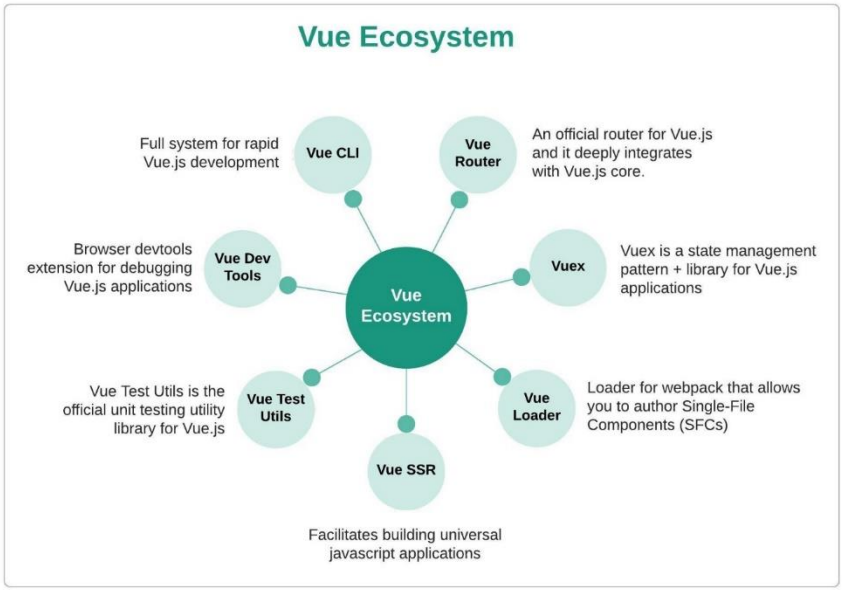
Το Vue είναι ένα πλαίσιο JavaScript που διευκολύνει την ανάπτυξη User Interface (UI)) ιστότοπων και εφαρμογών μιας σελίδας. Ένα προοδευτικό πλαίσιο JavaScript, το Vue κάνει τη δημιουργία διεπαφών χρήστη απλούστερη και πιο ευχάριστη.

Το Vue είναι ένα πλαίσιο που καλύπτει τα περισσότερα από τα κοινά χαρακτηριστικά που απαιτούνται για την ανάπτυξη του frontend. Ο Ιστός είναι εξαιρετικά διαφορετικός - τα πράγματα που αναπτύσσονται στον Ιστό μπορεί να διαφέρουν δραστικά σε μορφή και κλίμακα. Το Vue έχει σχεδιαστεί για να είναι ευέλικτο και να υιοθετείται σταδιακά. Ανάλογα με την περίπτωση χρήσης, το Vue μπορεί να χρησιμοποιηθεί με διάφορους τρόπους:

- Βελτίωση στατικού HTML χωρίς βήμα κατασκευής
- Ενσωμάτωση ως στοιχεία Web σε οποιαδήποτε σελίδα
- Εφαρμογή μιας σελίδας (SPA) Fullstack / Απόδοση από την πλευρά του διακομιστή (SSR)
- Jamstack / Δημιουργία στατικής τοποθεσίας (SSG)

Πλεονεκτήματα του VueJS:

- **Αρχιτεκτονική βασισμένη σε στοιχεία**: Όπως το Angular και το React, το πλαίσιο Vue js ακολουθεί μια αρχιτεκτονική που βασίζεται σε στοιχεία. Αυτό σημαίνει ότι όλος ο κώδικας εφαρμογής frontend μπορεί να χωριστεί σε ανεξάρτητα στοιχεία. Αυτά τα στοιχεία, που αποτελούνται από πρότυπο, λογική και στυλ, συνδέονται μεταξύ τους για να σχηματίσουν την εφαρμογή Ιστού.
- **Επαναχρησιμοποίηση:** Η προσέγγιση βάσει στοιχείων του Vue επιτρέπει τον σχηματισμό επαναχρησιμοποιήσιμων στοιχείων ενός αρχείου. Μέσα σε ένα στοιχείο, το πρότυπο, η λογική και τα στυλ συνδέονται εγγενώς. Αντί να διαχωρίζει τον κώδικα σε αυθαίρετα επίπεδα, το Vue συγκεντρώνει στοιχεία που μπορούν να επαναχρησιμοποιηθούν και σε μια συνάρτηση.
- **Υψηλή Απόδοση**: Το πλαίσιο Vue js είναι εξαιρετικά ελαφρύ, ομαδοποιώντας περίπου 20 KB. Αλλά δεν θέτει σε κίνδυνο την απόδοση ή την παραγωγικότητα. Είναι ένα από τα πιο γρήγορα διαθέσιμα πλαίσια για τη δημιουργία διεπαφών ιστού.
- **Εικονικό Dom**: Το Vue js χρησιμοποιεί μια εικονική αναπαράσταση του πραγματικού DOM μιας ιστοσελίδας, που δημιουργήθηκε χρησιμοποιώντας αντικείμενα Javascript. Έτσι τα αντικείμενα DOM μπορούν εύκολα να αποδοθούν χωρίς να τροποποιείται και να ανανεώνεται ολόκληρο το δέντρο κάθε φορά.



Σχήμα 2.12: Οικοσύστημα Vue

2.10 Back-End Τεχνολογίες Ανάπτυξης Διαδικτυακής Εφαρμογής

Το Back-End είναι ο κώδικας που εκτελείται στον διακομιστή, που λαμβάνει αιτήματα από τους πελάτες και περιέχει τη λογική για να στείλει τα κατάλληλα δεδομένα πίσω στον πελάτη. Το Back-End περιλαμβάνει επίσης τη βάση δεδομένων, η οποία θα αποθηκεύει όλα τα δεδομένα για την εφαρμογή.

Το Back-End είναι όλη η τεχνολογία που απαιτείται για την επεξεργασία του εισερχόμενου αιτήματος και τη δημιουργία και αποστολή της απάντησης στον πελάτη. Αυτό περιλαμβάνει συνήθως τρία κύρια μέρη:

- **Διακομιστής** (Server). Αυτός είναι ο υπολογιστής που λαμβάνει αιτήματα.
- **Εφαρμογή** (Application). Αυτή είναι η εφαρμογή που εκτελείται στον διακομιστή και ακούει αιτήματα, ανακτά πληροφορίες από τη βάση δεδομένων και στέλνει μια απάντηση.
- **Βάση δεδομένων** (Database). Οι βάσεις δεδομένων χρησιμοποιούνται για την οργάνωση και διατήρηση δεδομένων

Υπάρχουν πολλές τεχνολογίες Back-End για την ανάπτυξη Διαδικτυακών Εφαρμογών η κάθε μία με τα δικά της χαρακτηριστικά και ανάλογα την ανάγκη της εφαρμογής διαλέγεται συνήθως μία από αυτές, όπως για παράδειγμα το ASP.NET CORE, το Node.js και η Java.

2.10.1 ASP.NET CORE MVC

Το ASP.NET MVC Core είναι ένα Full-Stack πλαίσιο ανάπτυξης εφαρμογών ιστού από τη Microsoft που βασίζεται στην αρχιτεκτονική MVC για τη διευκόλυνση της δημιουργίας εφαρμογών για υπολογιστές, κινητά και διαδικτυακές εφαρμογές. Κατά κύριο λόγο, είναι ένα πλαίσιο, ελαφρύ μεταξύ όλων των άλλων τεχνολογιών ανάπτυξης της Microsoft και είναι επίσης ανοιχτού κώδικα.

Τα βασικά στοιχεία του ASP.NET Core MVC είναι:

- **Δρομολόγηση**
- **Model Validation**
- **Model Binding**
- **Dependency injection**
- **Φίλτρα**
- **Περιοχές**
- **Testability**

Δρομολόγηση

Για να γίνει το ASP.NET Core MVC αποδοτικό, βασίζεται αποκλειστικά στη δρομολόγηση του ASP.NET Core. Η κύρια λειτουργία της δρομολόγησης είναι η ενσωμάτωση λειτουργιών URL με δυνατότητα αναζήτησης για την βελτιστοποίηση μηχανών αναζήτησης. Επιπλέον, δίνεται η δυνατότητα δημιουργίας συνδέσμων, ανεξάρτητα από τη δομή του αρχείου στον διακομιστή. Επίσης, η δρομολόγησή του κατηγοριοποιείται σε δρομολόγηση συμβάσεων και δρομολόγηση χαρακτηριστικών. Το ένα βοηθά με τον καθολικό ορισμό URL κατά την αποδοχή ενός αιτήματος και το άλλο βοηθά στη συσχέτιση του ελεγκτή, των ενεργειών και της διαδρομής.

Model Validation

Η επικύρωση μοντέλου (Model Validation) του ASP.NET Core MVC είναι μια προηγμένη δυνατότητα που επαληθεύει τα χαρακτηριστικά από την πλευρά του πελάτη πριν τα ανεβάσετε στον διακομιστή. Επίσης, η ίδια διαδικασία εκτελείται από την πλευρά του διακομιστή προτού η εφαρμογή καλέσει τον ελεγκτή ενεργειών. Επιπλέον, το μοντέλο επικύρωσης χρησιμοποιεί jQuery για να επιβάλει τον σχολιασμό του προγράμματος περιήγησης για να προσθέσει τύπο μοντέλου στο επίπεδο προβολής.

Model Binding

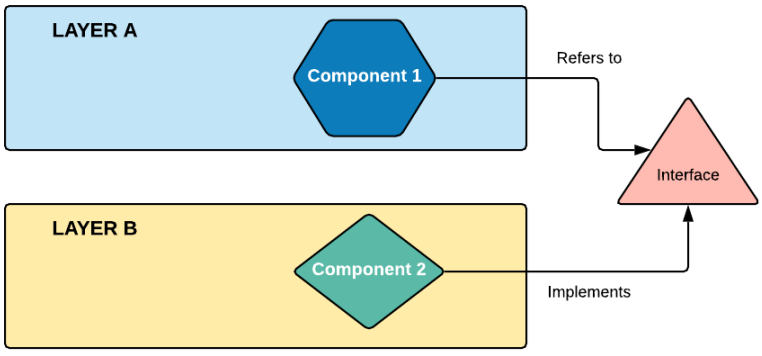
Η σύνδεση μοντέλου ASP.NET Core MVC στοχεύει στη μετατροπή των αιτημάτων των χρηστών σε αντικείμενα, οδηγώντας τον ελεγκτή να τα επεξεργάζεται αποτελεσματικά. Κυρίως, μετατρέπει τους ακόλουθους τύπους αιτημάτων:

- Τιμές που εισάγονται μέσω φορμών
- Δεδομένα που σχετίζονται με τη δρομολόγηση
- Κεφαλίδες HTTP
- Παράμετροι που σχετίζονται με το query string

Τα πλεονεκτήματα αυτής της δυνατότητας είναι ότι εξοικονομεί χρόνο στον ελεγκτή για να κατανοήσει το αίτημα και να το επεξεργαστεί.

Dependency injection

Ο λόγος για το Dependency Injection στο ASP.NET Core MVC του Σχήματος 2.13 είναι η χρήση του IoC (αντιστροφή ελέγχου) μεταξύ των κλάσεων και των σχετικών εξαρτήσεων. Επιπλέον, αξιοποιεί την εφαρμογή ώστε να ακολουθεί την αρχή της ρητής εξάρτησης. Λόγω αυτού, η εφαρμογή μπορεί να ζητήσει απαραίτητες/πρόσθετες υπηρεσίες και πόρους.



Σχήμα 2.13: Dependency Injection

Φίλτρα

Τα φίλτρα είναι ένα κρίσιμο στοιχείο του πλαισίου ASP.NET Core MVC. Βοηθά στην παροχή των ακόλουθων βασικών λειτουργιών:

- Βοηθά στην ενθυλάκωση εγκάρσιων ανησυχιών, όπως ο χειρισμός εξαιρέσεων.

- Υποστηρίζει την απρόσκοπτη λειτουργία της λογικής προ επεξεργασίας (pre-processing) και μετά επεξεργασίας (post-processing).
- Μπορεί να ρυθμιστεί να εκτελείται σε ένα συγκεκριμένο σενάριο. Ως εκ τούτου, μπορεί να ορισθεί μια συνθήκη για την ενεργοποίησή της κατά τη διάρκεια μιας μεμονωμένης διαδικασίας

Περιοχές

Στο ASP.NET MVC Core, οι περιοχές αναφέρονται σε ένα μέρος μιας εφαρμογής. Οι περιοχές βοηθούν στη διαίρεση του λογισμικού σε διαφορετικά μικρά μέρη, βοηθώντας τις ομάδες ανάπτυξης να δημιουργήσουν, να αναπτύξουν, να διατηρήσουν και να αναβαθμίσουν αποτελεσματικά.

Κάθε φορά που αναπτύσσεται μια μεγάλη εφαρμογή, όπως ένα κατάστημα ηλεκτρονικού εμπορίου ή ένα σύστημα διαχείρισης επιχειρήσεων, οι περιοχές μπαίνουν σε δράση. Λόγω αυτού, οι προγραμματιστές εργάζονται ανεξάρτητα σε κάθε στοιχείο και το δοκιμάζουν διεξοδικά για να διασφαλίσουν τη λειτουργικότητά του σύμφωνα με τις απαιτήσεις του πελάτη.

Testability

Όταν πρόκειται για τη δοκιμή μιας εφαρμογής ASP.NET MVC Core, διασφαλίζεται ότι θα αφαιρούνται τα μέγιστα κενά, θα χειρίζονται οι εξαιρέσεις και θα εξαλείφονται τα σφάλματα. Ως αποτέλεσμα των χαρακτηριστικών της περιοχής, οι δοκιμαστές μπορούν εύκολα να εξετάσουν κάθε στοιχείο, να το δοκιμάσουν και να βρουν πού υστερεί. Έτσι, η ομάδα λαμβάνει μια βαθιά εικόνα του κώδικα που πρέπει να διαμορφωθεί εκ νέου, να αναβαθμιστεί ή να ανανεωθεί.

Τα πλεονεκτήματα του ASP.NET CORE είναι:

- Βοηθά στην αξιόπιστη δοκιμή εφαρμογών, οδηγώντας στην εξάλειψη ευπαθειών, σφαλμάτων και δυσλειτουργιών.
- Παρέχει συμβατότητα μεταξύ πλατφορμών, βοηθώντας στην εκτέλεση λογισμικού σε Windows, iOS και Linux.
- Μπορεί να χρησιμοποιηθεί για τη δημιουργία API, τη σύνδεση με συστήματα τρίτων και την εκτέλεση λειτουργιών CRUD.
- Η ελαφριά αρχιτεκτονική βοηθά στην εξοικονόμηση κόστους εγκατάστασης και συντήρησης.
- Οι πόροι ανάπτυξης ASP.NET MVC Core είναι εύκολα διαθέσιμοι και οι οργανισμοί μπορούν εύκολα να αναθέσουν σε εξωτερικούς συνεργάτες τους προγραμματιστές.

2.10.2 Node.js

Το Node.js είναι ένα περιβάλλον χρόνου εκτέλεσης που επιτρέπει την εκτέλεση JavaScript από την πλευρά του διακομιστή. Είναι χτισμένο στη μηχανή JavaScript V8 του Chrome, η οποία μεταγλωττίζει τη JavaScript σε αποτελεσματικό κώδικα μηχανής. Το Node.js λειτουργεί σε μια αρχιτεκτονική μονού νήματος που βασίζεται σε συμβάντα, χρησιμοποιώντας έναν βρόχο συμβάντων για να χειρίζεται πολλές ταυτόχρονες λειτουργίες χωρίς αποκλεισμό.

Όταν ένας πελάτης στέλνει ένα αίτημα σε έναν διακομιστή Node.js, το αίτημα προστίθεται σε μια ουρά συμβάντων. Ο βρόχος συμβάντος ελέγχει συνεχώς αυτήν την ουρά και επεξεργάζεται κάθε αίτημα. Εάν ένα αίτημα περιλαμβάνει λειτουργία I/O, το Node.js το εκφορτώνει στον πυρήνα του συστήματος, ο

οποίος το χειρίζεται ασύγχρονα. Μόλις ολοκληρωθεί η λειτουργία I/O, ο πυρήνας ειδοποιεί το Node.js, εκτελώντας την αντίστοιχη συνάρτηση επανάκλησης.

Τα στοιχεία της αρχιτεκτονικής διακομιστή Node.js είναι:

- **Αιτήματα**
- **Node.js Server**
- **Ουρά συμβάντος**
- **Βρόχος συμβάντος**
- **Thread Pool**
- **Εξωτερικοί Πόροι**

Αιτήματα

Οι εισερχόμενες αιτήσεις μπορεί να είναι αποκλειστικές (σύνθετες) ή μη αποκλειστικές (απλές), ανάλογα με τις συγκεκριμένες εργασίες που θέλουν να εκτελέσουν οι χρήστες σε μια εφαρμογή Ιστού.

Node.js Server

Ο διακομιστής Node.js είναι το θεμέλιο της αρχιτεκτονικής. Ως πλατφόρμα από την πλευρά του διακομιστή, ο διακομιστής Node.js όχι μόνο δέχεται αιτήματα από χρήστες, αλλά επίσης επεξεργάζεται αυτά τα αιτήματα και στέλνει αυτές τις απαντήσεις στους αντίστοιχους χρήστες.

Ουρά Συμβάντων

Η ουρά συμβάντων στον διακομιστή Node.js αποθηκεύει τα εισερχόμενα αιτήματα πελατών και τα μεταβιβάζει ένα προς ένα στον βρόχο συμβάντων.

Βρόχος συμβάντος

Αυτός είναι ένας άπειρος βρόχος – που δεν τελειώνει ποτέ. Συνεχίζει να λαμβάνει αιτήματα από την ουρά συμβάντων, να τα επεξεργάζεται και να επιστρέφει την αντίστοιχη απάντηση στους πελάτες.

Αυτός ο βρόχος συμβάντος έχει έξι φάσεις που επαναλαμβάνονται μέχρι να μην μείνει κανένας κώδικας για εκτέλεση. Οι έξι φάσεις του βρόχου συμβάντων είναι:

1. Χρονοδιακόπτες
2. I/O Επανακλήσεις
3. Αναμονή / Προετοιμασία
4. I/O Polling
5. Επανακλήσεις setImmediate()
6. Κλείσιμο εκδηλώσεων

Thread Pool

Το Thread Pool στην αρχιτεκτονική Back-End Node.js περιέχει τα νήματα για την εκτέλεση εργασιών που απαιτούνται για την επεξεργασία αιτημάτων πελατών.

Εξωτερικοί Πόροι

Οι Εξωτερικοί Πόροι χρησιμοποιούνται για τον αποκλεισμό αιτημάτων πελατών. Γενικά χειρίζονται πολλαπλά αιτήματα αποκλεισμού, όπως αποθήκευση δεδομένων, υπολογισμοί κ.λπ.

Τα πλεονεκτήματα του Node.JS είναι:

- **Επεκτάσιμη ανάπτυξη εφαρμογών Ιστού**: Το Node.js δημιουργήθηκε με γνώμονα την επεκτασιμότητα. Επιτρέπει σε πολλαπλά νήματα να τρέχουν ταυτόχρονα και να αλληλεπιδρούν μεταξύ τους, κάτι που είναι ανώτερο από άλλες λύσεις ανάπτυξης υποστήριξης ιστού.
- **Υψηλή απόδοση**: Το Node.js είναι δημοφιλές για την παροχή εξαιρετικής απόδοσης και την αντιμετώπιση σημαντικών ποσοτήτων δεδομένων προς επεξεργασία ταυτόχρονα.
- **Ισχυρή επεξεργασία δεδομένων**: Το Node.js είναι μια γλώσσα ανάπτυξης Back-End και περιβάλλον χρόνου εκτέλεσης ικανό να επεξεργάζεται χιλιάδες αιτήματα ταυτόχρονα.
- **Βοηθά στη δημιουργία εφαρμογών μεταξύ πλατφορμών**: Χρησιμοποιώντας πλαίσια όπως το Electron και το NW.js, το Node.js επιτρέπει την ανάπτυξη διαδικτυακών εφαρμογών σε πραγματικό χρόνο σε πολλαπλές πλατφόρμες.

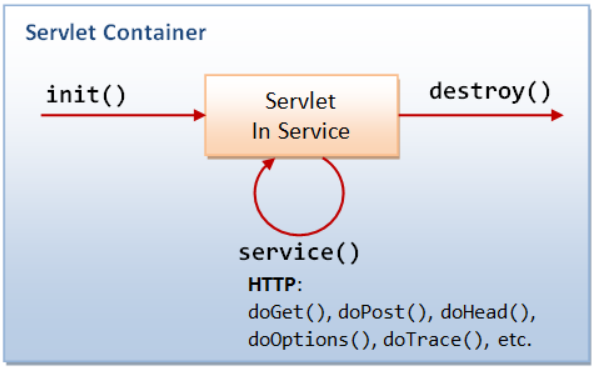
2.10.3 Java

Η Java είναι μια ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού για την ανάπτυξη διαδικτυακών εφαρμογών. Είναι μια δημοφιλής επιλογή μεταξύ των προγραμματιστών για πάνω από δύο δεκαετίες, με εκατομμύρια εφαρμογές Java που χρησιμοποιούνται σήμερα. Η Java είναι μια πολυπλατφορμική, αντικειμενοστρεφής και δικτυοκεντρική γλώσσα που μπορεί να χρησιμοποιηθεί ως πλατφόρμα από μόνη της. Είναι μια γρήγορη, ασφαλής, αξιόπιστη γλώσσα προγραμματισμού για την κωδικοποίηση όλων, από εφαρμογές για κινητά και εταιρικό λογισμικό μέχρι εφαρμογές μεγάλων δεδομένων και τεχνολογίες διακομιστή.

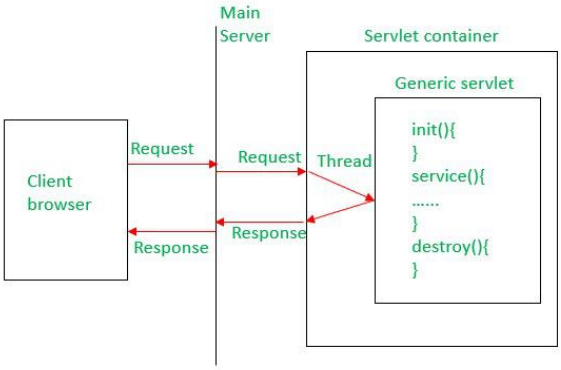
Η Java παρέχει ορισμένες τεχνολογίες όπως το Servlet και το JavaServer Pages (JSP) που επιτρέπουν να αναπτύχθει μια εφαρμογή Ιστού εύκολα σε έναν διακομιστή. Παρέχει επίσης ορισμένα πλαίσια όπως το Spring, το Spring Boot που απλοποιούν την εργασία και παρέχουν έναν αποτελεσματικό τρόπο ανάπτυξης μιας εφαρμογής web και μειώνουν την προσπάθεια του προγραμματιστή.

Java Servlet API

Το Java Servlet API επιτρέπει να ορίζονται κλάσεις ειδικές για το HTTP όπως στο Σχήμα 2.14. Μια κλάση servlet επεκτείνει τις δυνατότητες των διακομιστών που φιλοξενούν εφαρμογές που έχουν πρόσβαση μέσω ενός μοντέλου προγραμματισμού αιτήματος-απόκρισης όπως στο Σχήμα 2.15. Οι Servlets μπορούν να ανταποκριθούν σε οποιοδήποτε τύπο αιτήματος, χρησιμοποιούνται συνήθως για την επέκταση των εφαρμογών που φιλοξενούνται από διακομιστές Ιστού.



Σχήμα 2.14: Αιτήματα HTTP



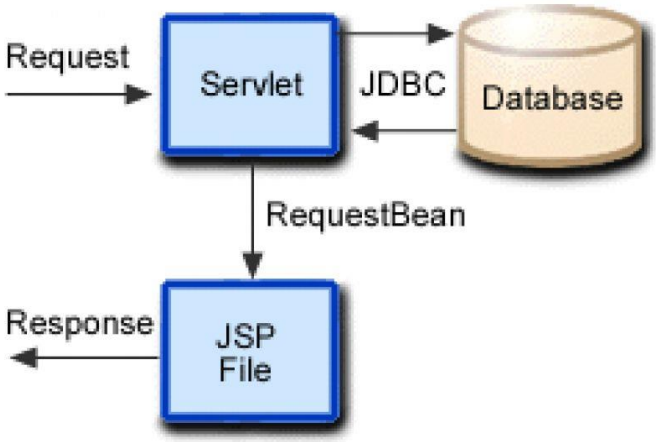
Σχήμα 2.15: Μοντέλο Αιτήματος-Απόκρισης

JavaServer Pages Technology

Η τεχνολογία JSP παρέχει έναν απλοποιημένο, γρήγορο τρόπο δημιουργίας δυναμικού περιεχομένου ιστού. Η τεχνολογία JSP επιτρέπει την ταχεία ανάπτυξη εφαρμογών που βασίζονται στο web που είναι ανεξάρτητες από διακομιστή και πλατφόρμα. Η τεχνολογία JSP επιτρέπει να προστίθενται αποσπάσματα κώδικα servlet απευθείας σε ένα έγγραφο που βασίζεται σε κείμενο όπως στο Σχήμα 2.16.

Συνήθως, μια σελίδα JSP είναι ένα έγγραφο που βασίζεται σε κείμενο και περιέχει δύο τύπους κειμένου:

- **Στατικά δεδομένα**, τα οποία μπορούν να εκφραστούν σε οποιαδήποτε μορφή που βασίζεται σε κείμενο, όπως HTML, Wireless Markup Language (WML) ή XML
- **Στοιχεία τεχνολογίας JSP**, τα οποία καθορίζουν τον τρόπο με τον οποίο η σελίδα κατασκευάζει δυναμικό περιεχόμενο



Σχήμα 2.16: Συνεργασία Servlet-JSP

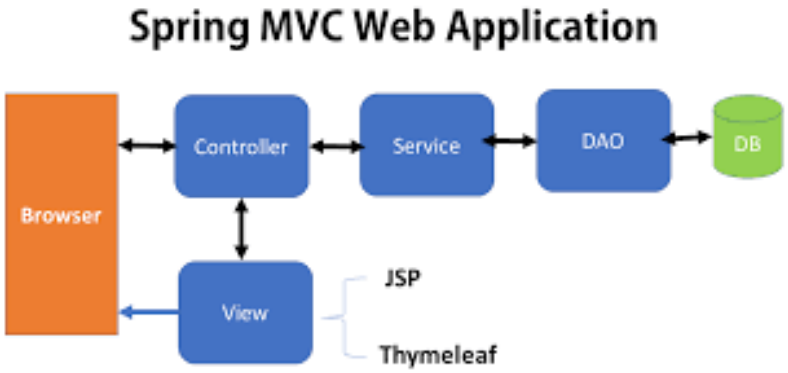
Spring

Το πλαίσιο Spring όπως φαίνεται στο Σχήμα 2.17 παρέχει ένα ολοκληρωμένο μοντέλο προγραμματισμού και διαμόρφωσης για σύγχρονες εταιρικές εφαρμογές που βασίζονται σε Java - σε κάθε είδους πλατφόρμα ανάπτυξης. Ένα βασικό στοιχείο του Spring είναι η υποστήριξη υποδομής σε επίπεδο εφαρμογών: Το Spring εστιάζει στην "υδραυλική εγκατάσταση" των εταιρικών εφαρμογών,

έτσι ώστε οι ομάδες να μπορούν να επικεντρωθούν στην επιχειρηματική λογική σε επίπεδο εφαρμογής, χωρίς περιττούς δεσμούς με συγκεκριμένα περιβάλλοντα ανάπτυξης.

Κάποια από τα χαρακτηριστικά του πλαισίου Spring είναι:

- **Βασικές τεχνολογίες**: Dependency Injection, συμβάντα, πόροι
- **Testing**: Εικονικά αντικείμενα, πλαίσιο TestContext, Spring MVC Test
- **Πρόσβαση δεδομένων**: συναλλαγές, υποστήριξη DAO, JDBC, ORM



Σχήμα 2.17: Αρχιτεκτονική Spring

Τα πλεονεκτήματα της Java:

- **Είναι εύκολο να κλιμακωθεί**: Η Java μπορεί να κλιμακωθεί πολύ εύκολα για την ανάπτυξη εφαρμογών ιστού, επειδή τα στοιχεία της είναι ευρέως προσβάσιμα και διαθέσιμα. Η εφαρμογή ιστού μπορεί να κλιμακωθεί οριζόντια και κάθετα ταυτόχρονα, ανάλογα με την απαίτηση.
- **Cross-Platform**: Όταν χρησιμοποιείται η γλώσσα Java, μπορεί να γραφτεί κώδικας και να χρησιμοποιηθεί σχεδόν οπουδήποτε. Αυτό κάνει την Java μια πολύ «φορητή» γλώσσα. Ως κώδικας πολλαπλών πλατφορμών, μπορεί να εκτελεστεί σε όλες τις πλατφόρμες χάρη στην Java Virtual Machine (JVM).
- **Multi-thread**: Οι εφαρμογές ιστού Java μπορούν να εξισορροπήσουν τη χρήση τους όταν έχουν πρόσβαση από πολλούς χρήστες ταυτόχρονα. Η δημιουργία νημάτων το κάνει αυτό για κάθε μεμονωμένο χρήστη εντός της ίδιας της εφαρμογής αντί να δημιουργεί πολλαπλά αντίγραφα στη συσκευή όπου γίνεται πρόσβαση στην εφαρμογή. Μέσω αυτού, τα νήματα παρακολουθούνται για να εξασφαλιστεί εξαιρετική απόδοση.

2.10.4 Βάσεις Δεδομένων

Ο όρος «βάση δεδομένων» επινοήθηκε από τον Peter Naur το 1960 για να περιγράψει την προσέγγισή του στην ανάπτυξη συστημάτων λογισμικού. Στις πρώτες μέρες της πληροφορικής, οι βάσεις δεδομένων ήταν συνώνυμες με αρχεία σε δίσκο. Ο όρος εξακολουθεί να χρησιμοποιείται συνήθως με αυτόν τον τρόπο.

Τα δεδομένα είναι το θεμέλιο μιας διαδικτυακής εφαρμογής. Μια βάση δεδομένων είναι μια συλλογή δεδομένων και πληροφοριών που αποθηκεύονται με οργανωμένο τρόπο για εύκολη ανάκτηση. Χρησιμοποιείται για την αποθήκευση πληροφοριών χρήστη, δεδομένων συνεδρίας και άλλων δεδομένων εφαρμογών. Η βάση δεδομένων είναι το κεντρικό αποθετήριο για όλα τα δεδομένα. Οι εφαρμογές Ιστού χρησιμοποιούν μια ποικιλία βάσεων δεδομένων για την αποθήκευση δεδομένων όπως επίπεδα αρχεία, σχεσιακές βάσεις δεδομένων, βάσεις δεδομένων αντικειμενικών σχέσεων και βάσεις

δεδομένων NoSQL. Κάθε τύπος βάσης δεδομένων έχει τα δικά του πλεονεκτήματα και μειονεκτήματα όσον αφορά την αποθήκευση και την ανάκτηση δεδομένων. Ο πρωταρχικός σκοπός μιας βάσης δεδομένων είναι η αποθήκευση, η ανάκτηση και η ενημέρωση πληροφοριών. Μια βάση δεδομένων μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων που σχετίζονται με οποιαδήποτε πτυχή των επιχειρηματικών λειτουργιών.

Υπάρχουν αρκετοί τύποι βάσεων δεδομένων που μπορεί ένας προγραμματιστής να επιλέξει ανάλογα με την χρήση της εφαρμογής που θα αναπτύξει. Κάποιες από αυτές είναι: Microsoft SQL Server, MySql και MongoDB.

Microsoft SQL Server

Ο Microsoft SQL Server βασίζεται στη γλώσσα SQL και περιορίζεται στα λειτουργικά συστήματα Windows και Linux. Αν και δεν είναι τόσο προηγμένη όσο άλλες σύγχρονες βάσεις δεδομένων, παρέχει αποτελεσματική διαχείριση φόρτου εργασίας. Η ίδια βάση δεδομένων μπορεί να χρησιμοποιηθεί από πολλούς χρήστες, διαθέτει εύκολη δημιουργία αντιγράφων ασφαλείας και ανάκτηση και είναι εξαιρετικά ασφαλής και συνεπής. Εάν η εφαρμογή περιλαμβάνει επεξεργασία συναλλαγών, επιχειρηματική ευφυΐα και αναλυτικά στοιχεία, όπως μια εφαρμογή που χρησιμοποιείται για εξόρυξη δεδομένων, το MS SQL μπορεί να αποδειχθεί εξαιρετική επιλογή. Ακόμη περισσότερο όταν το περιβάλλον ανάπτυξης είναι σε μεγάλο βαθμό ενσωματωμένο με διαφορετικά προϊόντα της Microsoft.

MySQL

Μία από τις πιο δημοφιλείς βάσεις δεδομένων που χρησιμοποιούνται για την ανάπτυξη εφαρμογών Ιστού, η MySQL είναι γνωστή για τη σταθερότητα, την αξιοπιστία, την ασφάλεια και την οικονομική αποδοτικότητά της. Διαθέτει επεξεργασία δεδομένων υψηλής ταχύτητας και ανάκτηση δεδομένων και είναι εύκολο στη χρήση. Επειδή είναι ανοιχτού κώδικα, ο κώδικας μπορεί να τροποποιηθεί ελεύθερα. Επιπλέον, η βάση δεδομένων παρέχει τεχνική υποστήριξη στους χρήστες και τους προγραμματιστές, υποστηρίζει την αρχιτεκτονική πελάτη-διακομιστή και επιτρέπει την εύκολη ενσωμάτωση με άλλες μηχανές.

MongoDB

Όταν κυκλοφόρησε, η MongoDB ήταν από τις πρώτες βάσεις δεδομένων που προσανατολίζονται σε έγγραφα και σήμερα είναι μια από τις πιο δημοφιλείς βάσεις δεδομένων ανοιχτού κώδικα που βασίζεται στο σύστημα NoSQL. Προτιμάται κυρίως από προγραμματιστές που εργάζονται με αποθήκευση δεδομένων μεγάλου όγκου. Το MongoDB είναι διαθέσιμο και ως μια πλήρως διαχειριζόμενη υπηρεσία cloud και μια αυτοδιαχειριζόμενη υποδομή. Τα κύρια χαρακτηριστικά του περιλαμβάνουν ενσωματωμένο failover και αναπαραγωγή, ασφάλεια δεδομένων από άκρο σε άκρο και πλήρη ελαστικότητα

2.11 Επίλογος

Σε αυτό το κεφάλαιο, αναλύθηκαν οι τεχνολογίες που χρησιμοποιούνται για την ανάπτυξη Διαδικτυακών Εφαρμογών. Αρχικά αναπτύχθηκαν οι ευρύτερες έννοιες των Διαδικτυακών Εφαρμογών και στην συνέχεια παρουσιάστηκαν οι πιο γνωστές τεχνολογίες Front-End και Back-End που χρησιμοποιούνται πιο συχνά στην ανάπτυξη εφαρμογών.

Κεφάλαιο 3ο: Front-End

3.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναλυθεί η υλοποίηση του Front-End μέρους του αυτοματοποιημένου συστήματος εξέτασης για το εργαστηριακό μέρος του μαθήματος των Δικτύων χρησιμοποιώντας κάποιες από τις τεχνολογίες που αναφέρθηκαν στο προηγούμενο κεφάλαιο για την ανάπτυξη του.

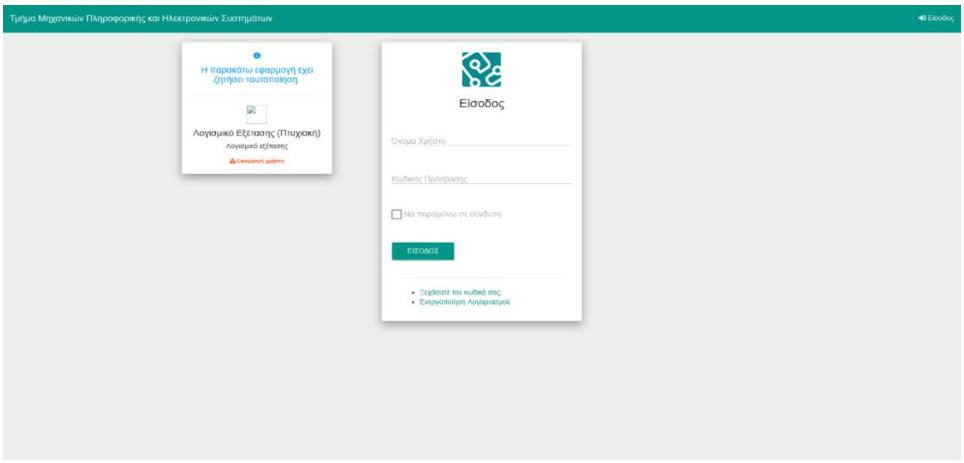
Για την ανάπτυξη του Front-End μέρους επιλέχθηκαν βασικές τεχνολογίες όπως η HTML, CSS, JavaScript, το Full-Stack πλαίσιο ASP.NET Core με Razor σελίδες και πλαίσια όπως η AngularJS, η Vue.js αλλά και το CSS πλαίσιο Bootsrap.

Τα πλαίσια που χρησιμοποιήθηκαν για την ανάπτυξη της διαδικτυακής εφαρμογής και του Front-End είναι το ASP.NET Core MVC όπου σε αυτό συμπεριλαμβάνεται και η HTML,CSS,JavaScript καθώς και το CSS πλαίσιο Bootstrap διότι οι τεχνολογίες που θα χρησιμοποιηθούν έπρεπε να πληρούν κάποιες συγκεκριμένες προδιαγραφές όπως:

- **Συντήρηση Κώδικα**: Ο κώδικας της εφαρμογής πρέπει να είναι ευανάγνωστος για να διευκολύνει την κατανόηση και την τροποποίηση του σε μελλοντικές αλλαγές που θα χρειαστούν. Το ASP.NET Core MVC μέσω των Razor σελίδων επιτρέπει την ξεχωριστή διαχείριση.
- **Επεκτασιμότητα**: Η επεκτασιμότητα και η υποδομή θα πρέπει να επιτρέπουν στους προγραμματιστές να δημιουργούν ελαφρύτερες, πιο ασφαλείς και εξαιρετικά παραγωγικές εφαρμογές.
- **Cross-Platform**: Η πλατφόρμα θα πρέπει να επιτρέπει την δημιουργία και ανάπτυξη εφαρμογών ASP.NET σε περιβάλλοντα Windows, macOS και Linux.
- **Ενεργή Κοινότητα**: Η τεχνολογία θα πρέπει να έχει μία ενεργή κοινότητα έτσι ώστε να υπάρχει και η ανάλογη υποστήριξη.

3.2 Σελίδα Εισόδου

Η είσοδος του χρήστη γίνεται μέσω του Authentication Server της σχολής «login.iee.ihu.gr» του Σχήματος 3.1. Ο χρήστης θα πρέπει να συμπληρώσει τα διαπιστευτήρια του όπου χρησιμοποιεί στην πλατφόρμα του «login.iee.ihu.gr» έτσι ώστε να συνδεθεί στην πλατφόρμα εξέτασης. Όταν ο χρήστης κάνει κλικ στο κουμπί «Είσοδος» και συμπληρώσει τα στοιχεία του κάνει σύνδεση και ανακατευθύνεται στην αρχική σελίδα της εφαρμογής.



Σχήμα 3.1: Είσοδος χρήστη

3.3 Διαχωρισμός Προβολής Καθηγητή και Φοιτητή

Μετά διαδικασία εισόδου στην πλατφόρμα εξέτασης θα δημιουργηθούν δύο ρόλοι μέσω των “Claims”. Ένας ρόλος είναι του καθηγητή και ένας του φοιτητή.

Ο ρόλος του καθηγητή έχει τα εξής δικαιώματα στην πλατφόρμα:

- Να μπορεί να δημιουργεί και να διαγράφει κατηγορίες ερωτήσεων.
- Να μπορεί να δημιουργεί και να διαγράφει ερωτήσεις.
- Να μπορεί να δημιουργεί και να διαγράφει πακέτα ερωτήσεων για την ομαδοποίηση των ερωτήσεων.
- Να μπορεί να δημιουργεί και να διαγράφει εξετάσεις.
- Να μπορεί να εξάγει τα αποτελέσματα των φοιτητών.

Ο ρόλος του φοιτητή έχει τα εξής δικαιώματα στην πλατφόρμα:

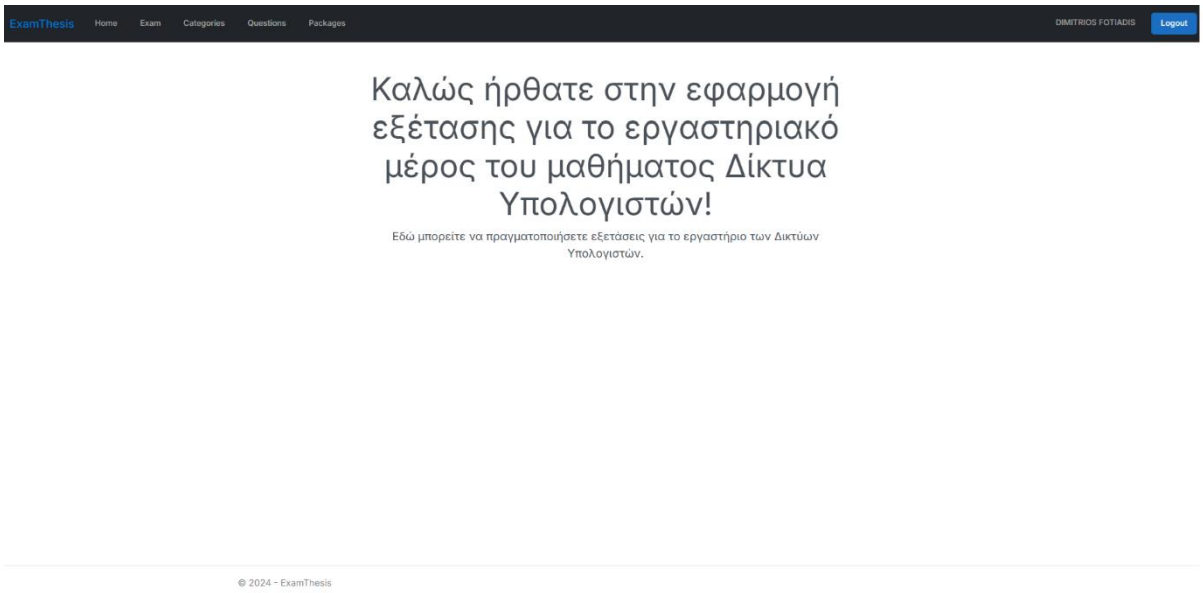
- Να μπορεί να εισέρχεται στην εξέταση
- Να μπορεί να παίρνει μέρος στην εξέταση και να υποβάλει τις απαντήσεις των ερωτήσεων
- Να μπορεί να βλέπει μετά την υποβολή των απαντήσεων αν ήταν επιτυχής ή ανεπιτυχής η εξέταση

3.4 Προβολή Καθηγητή

Ο καθηγητής, έχει πλήρη πρόσβαση στην πλατφόρμα εξέτασης. Μπορεί να δημιουργήσει ερωτήσεις, κατηγορίες, πακέτα ερωτήσεων και εξετάσεις του Σχήματος 3.2.

Ο καθηγητής έχε πρόσβαση στις σελίδες από το μενού:

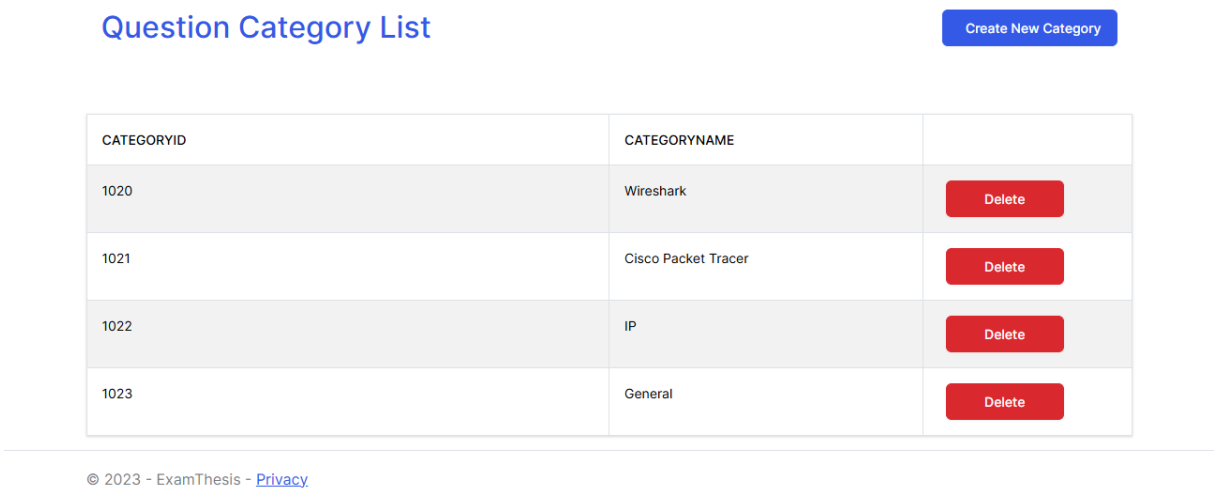
- Exam
- Categories
- Questions
- Packages



Σχήμα 3.2: Προβολή Μενού Καθηγητή

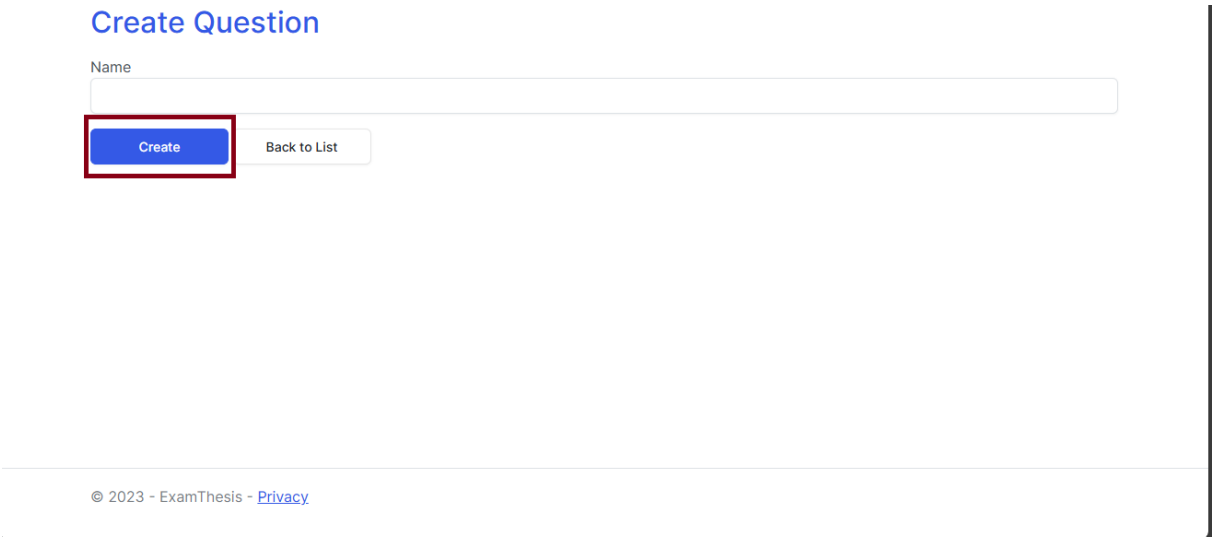
3.4.1 Σελίδα Δημιουργίας Κατηγοριών

Όταν ο χρήστης κάνει κλικ στο κουμπί “Categories” εισέρχεται στην σελίδα δημιουργίας κατηγοριών ερωτήσεων. Στην σελίδα «/QuestionCategory/Index» του Σχήματος 3.3 ο χρήστης μπορεί να δει σε λίστα τις υπάρχουσες κατηγορίες ερωτήσεων.



Σχήμα 3.3: Σελίδα Λίστας Κατηγοριών

Στην συνέχεια όταν ο χρήστης κάνει κλικ στο κουμπί “Create New Category” τον ανακατευθύνει στην σελίδα «/QuestionCategory/Create» του Σχήματος 3.4 όπου μπορεί να δημιουργήσει μία κατηγορία. Εφόσον συμπληρώσει το πεδίο “Name” κάνει κλικ στο κουμπί “Create” εκτελείτε η μέθοδος “Post” της φόρμας του Σχήματος 3.5. Η κατηγορία αποθηκεύεται και ανακατευθύνει τον χρήστη στην αρχική σελίδα των κατηγοριών.

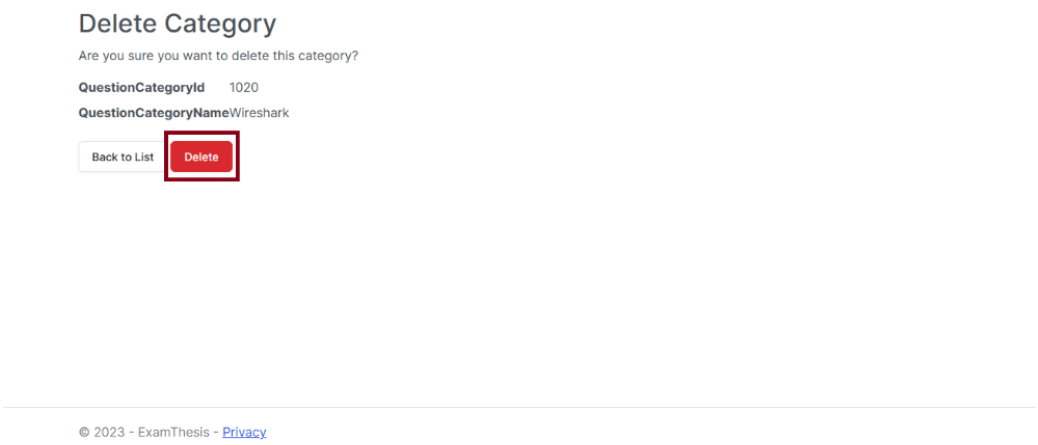


Σχήμα 3.4: Σελίδα Δημιουργίας Κατηγοριών

```
@model ExamThesis.Models.QuestionCategory
<form enctype="multipart/form-data" method="post">
        <div class="row pb-2">
            <h2 class="text-primary">Create Question</h2>
        </div>
        <div class="mb-3">
            <label asp-for="QuestionCategoryName">Name</label>
        <input asp-for="QuestionCategoryName" class="form-control" />
    </div>
        <button type="submit" class="btn btn-primary" style="width:150px">Create</button>
        <a asp-controller="QuestionCategory" asp-action="Index" class="btn btn-secondary" style="width:150px">Back to List</a>
</form>
```

Σχήμα 3.5: Κώδικας Σελίδας Δημιουργίας Κατηγοριών

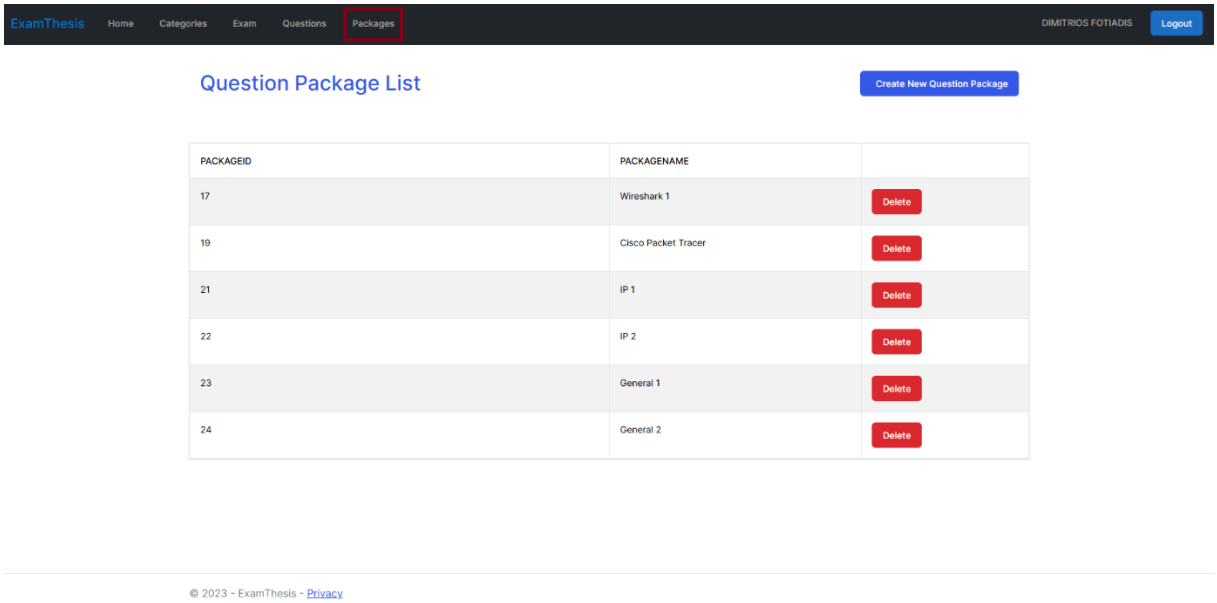
Ο χρήστης επίσης έχει την δυνατότητα διαγραφής κατηγοριών που έχουν δημιουργηθεί. Στην σελίδα «/QuestionCategory/Index» του Σχήματος 3.6 όταν ο χρήστης κάνει κλικ στο κουμπί “Delete” που βρίσκεται στην ίδια σειρά με την κατηγορία που θέλει να διαγράψει. Με την βοήθεια της συνάρτησης του Παραρτήματος Α: Σελίδα Διαγραφής κατηγορίας, μόλις κάνει κλικ ο χρήστης στο κουμπί με “id=deleteButton” ψάχνει την κατηγορία που θέλει να διαγράψει και εμφανίζει ένα μήνυμα επιβεβαίωσης. Εφόσον ο χρήστης κάνει κλικ στο κουμπί “OK” τότε αποστέλλεται ένα Ajax αίτημα τύπου “Delete” στον “Controller”. Στην συνέχεια όταν η διαδικασία ολοκληρωθεί ο χρήστης ανακατευθύνεται στην σελίδα “Index” με ανανεωμένη την λίστα κατηγοριών.



Σχήμα 3.6: Σελίδα Διαγραφής Κατηγοριών

3.4.2 Σελίδα Πακέτων Ερωτήσεων

Όταν ο χρήστης κάνει κλικ στο κουμπί “Packages” ανακατευθύνεται στην σελίδα πακέτων ερωτήσεων του Σχήματος 3.7. Εκεί δημιουργείτε μία λίστα με τα υπάρχοντα πακέτα ερωτήσεων όπως και την δυνατότητα δημιουργίας ενός καινούριου πακέτου άλλα και την διαγραφή ενός υπάρχοντος πακέτου με την βοήθεια μίας “for loop” του Σχήματος 3.8 όπου κάνει ανάκτηση του “PackageId” “PackageName” για κάθε πακέτο ερωτήσεων που υπάρχει. Τα πακέτα ερωτήσεων βοηθούν στην ομαδοποίηση ερωτήσεων έτσι ώστε ο καθηγητής να μπορεί να φτιάξει διάφορα πακέτα θεμάτων με σκοπό τον διαμοιρασμό ξεχωριστών πακέτων ερωτήσεων κατά την εξέταση στους φοιτητές.

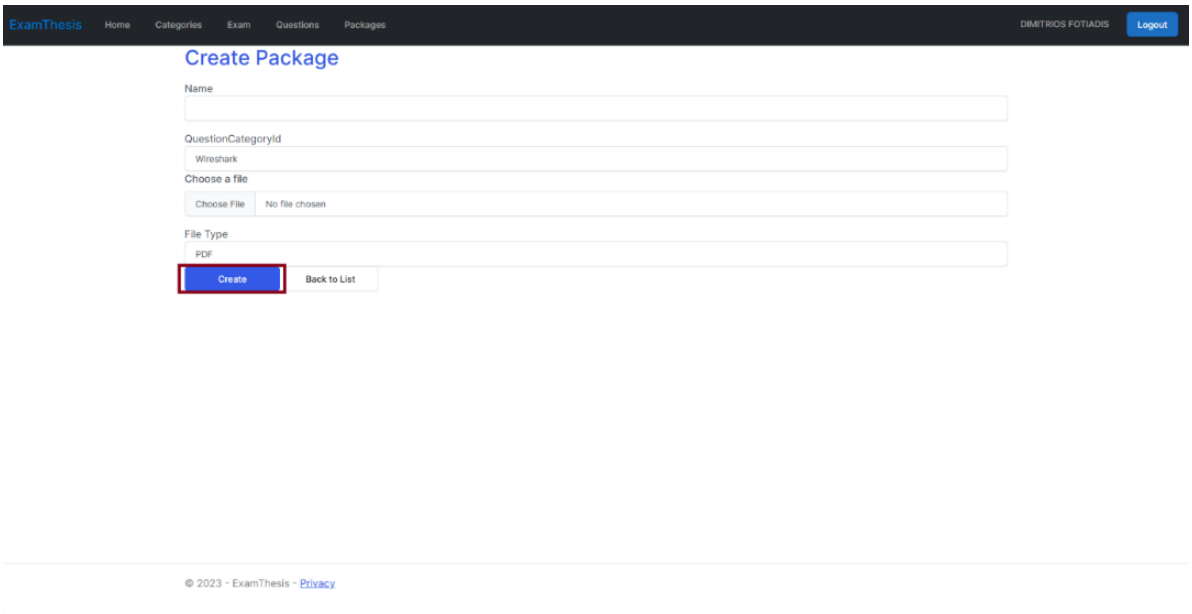


Σχήμα 3.7: Σελίδα Πακέτων Ερωτήσεων

```
@foreach (var obj in ViewBag.QuestionPackagesList)
{
    <tr>
        <td width="50%">
            @obj.PackageId
        </td>
        <td width="30%">
            @obj.PackageName
        </td>
```

Σχήμα 3.8: Κώδικας Σελίδας Πακέτων

Στην συνέχεια όταν ο χρήστης κάνει κλικ στο κουμπί “Create New Question Package” ανακατευθύνεται στην σελίδα δημιουργίας ενός πακέτου του Σχήματος 3.9, όπου πρέπει να συμπληρώσει το πεδίο όνομα πακέτου, να διαλέξει σε ποια κατηγορία ερωτήσεων ανήκει το συγκεκριμένο πακέτο μέσω των ήδη υπαρχόντων κατηγοριών με την βοήθεια της “ViewBag” του Σχήματος 3.10 αλλά επίσης του δίνεται και η δυνατότητα ανεβάσματος ενός αρχείου ανά πακέτο ερωτήσεων που μπορεί να περιέχει διάφορες πληροφορίες για τις ερωτήσεις που περιλαμβάνει το πακέτο ερωτήσεων. Εφόσον ο χρήστης συμπληρώσει τα πεδία της φόρμας, κάνει κλικ στο κουμπί “Create”, καλεί την μέθοδο “Post” της φόρμας. Όταν ολοκληρωθεί η αποθήκευση ανακατευθύνεται ο χρήστης στην σελίδα “Index” με ανανεωμένη την λίστα πακέτων ερωτήσεων.

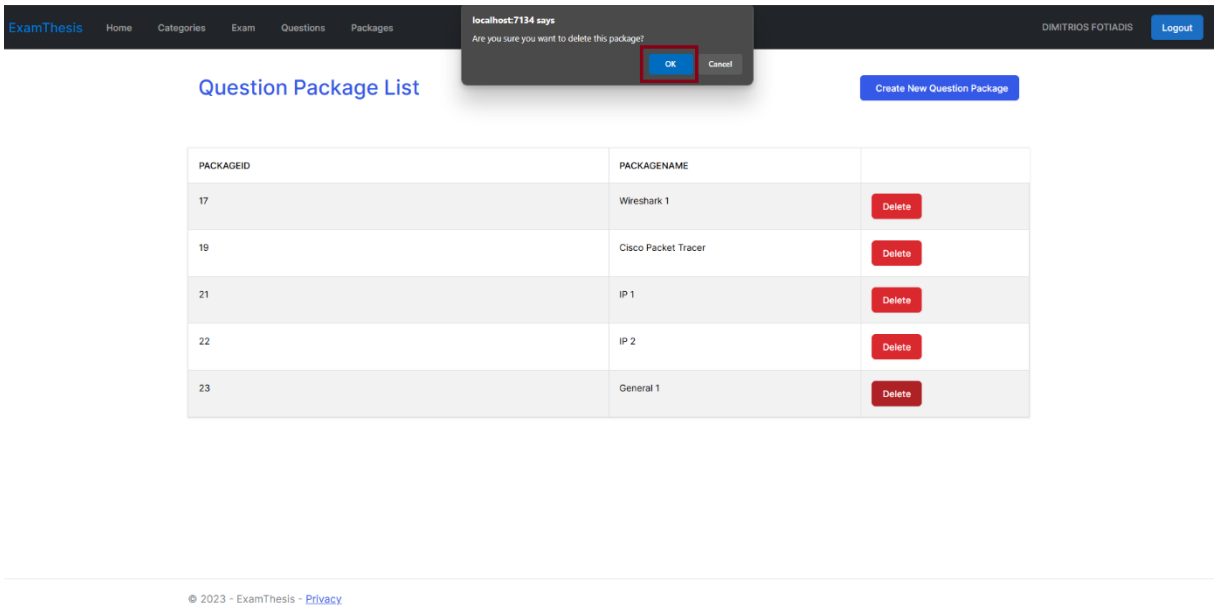


Σχήμα 3.9: Σελίδα Δημιουργίας Πακέτου

```
@model ExamThesis.Models.QuestionPackage
<form enctype="multipart/form-data" method="post">
        <div class="row pb-2">
            <h2 class="text-primary">Create Package</h2>
        </div>
        <div class="mb-3">
        <label asp-for="PackageName">Name</label>
        <input asp-for="PackageName" class="form-control" />
        </div>
        <div class="form-group">
            <label asp-for="QuestionCategoryId" class="control-label"></label>
            <select asp-items="ViewBag.QuestionCategories" asp-for="QuestionCategoryId" class="form-control"></select>
        </div>
        <div class="mb-3">
            <label asp-for="FileData" class="form-label">Choose a file</label>
            <input type="file" asp-for="FileData" class="form-control">
        </div>
        <div class="form-group">
            <label for="FileType">File Type</label>
            <select class="form-control" id="FileType" name="FileType">
                <option value=".pdf">PDF</option>
                <option value=".pcapng">PCAPNG</option>
                <option value=".pkt">PKT</option>
            </select>
        </div>

        <button type="submit" class="btn btn-primary" style="width:150px">Create</button>
        <a asp-controller="QuestionCategory" asp-action="Index" class="btn btn-secondary" style="width:150px">Back to List</a>
</form>
```

Σχήμα 3.10: Κώδικας Σελίδας Δημιουργίας

Επίσης ο χρήστης έχει την δυνατότητα διαγραφής ενός πακέτου ερωτήσεων. Όταν ο χρήστης κάνει κλικ στο κουμπί “Delete” εμφανίζεται ένα μήνυμα διαγραφής του Σχήματος 3.11 . Μόλις ο χρήστης κάνει κλικ στο κουμπί “OK” που εμφανίζεται στην οθόνη του η διαγραφή ολοκληρώνεται και ανανεώνεται η σελίδα “Index”.

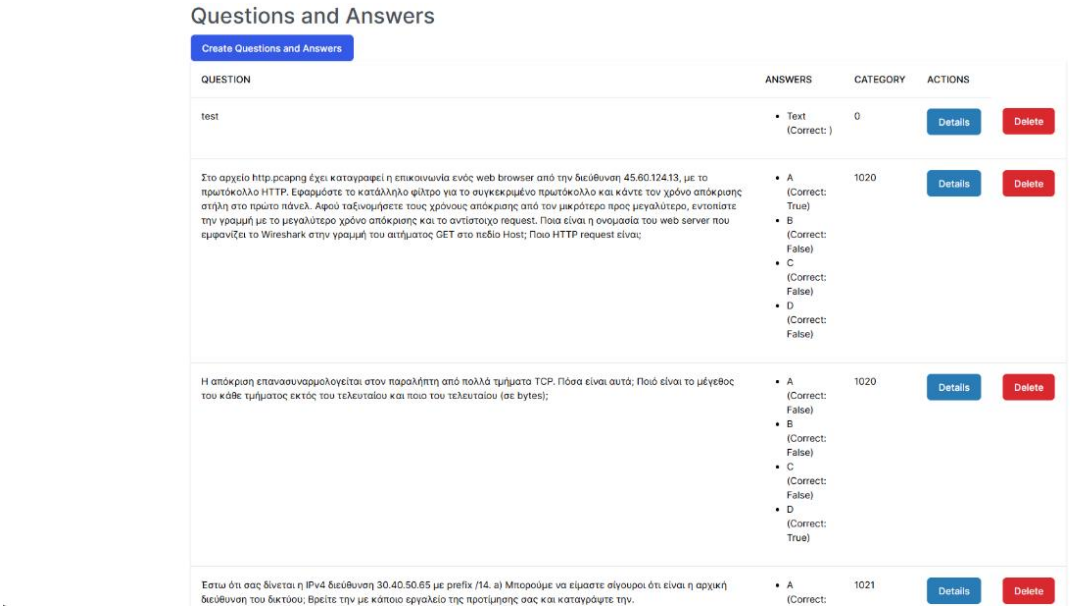


Σχήμα 3.11: Μήνυμα Διαγραφής Πακέτου

3.4.3 Σελίδα Ερωτήσεων

Όταν ο χρήστης κάνει κλικ στο κουμπί “Question” του μενού τον ανακατευθύνει στη σελίδα “Index” των ερωτήσεων. Εκεί ο χρήστης όπως παρουσιάζεται στο Σχήμα 3.12 θα βρει μία λίστα με όλες τις υπάρχουσες ερωτήσεις καθώς και τις επιλογές δημιουργίας αλλά και διαγραφής μίας ερώτησης με την

βοήθεια μίας “for loop” του Σχήματος 3.13. Ο χρήστης έχει επίσης την επιλογή ενός κουμπιού “Details” όπου όταν το κάνει κλικ τον ανακατευθύνει στην σελίδα “Details” με τις πληροφορίες της κάθε ερώτησης.

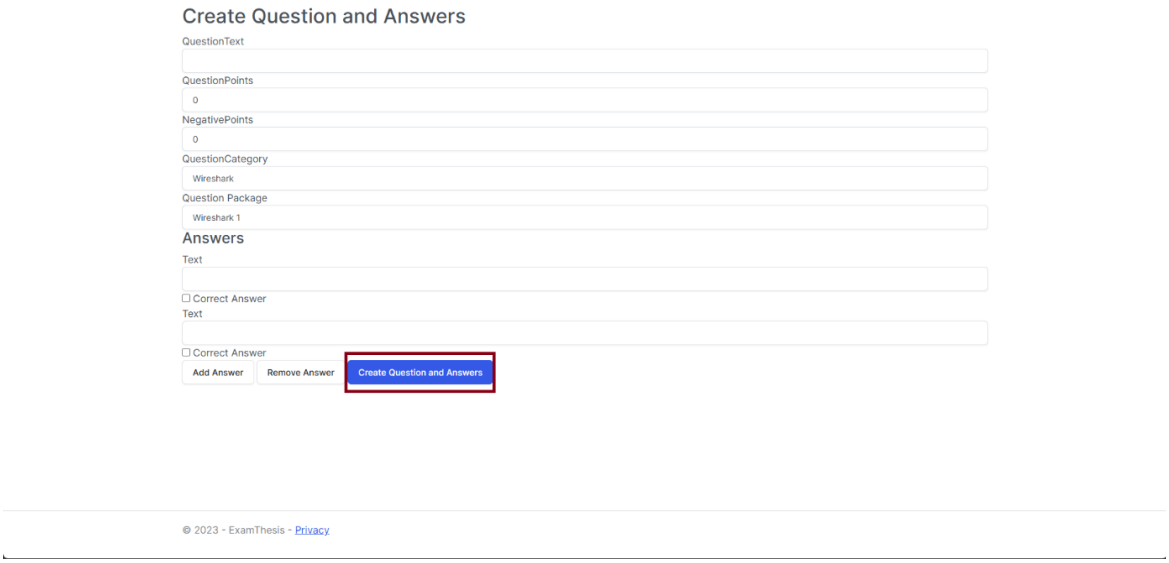


Σχήμα 3.12: Σελίδα Ερωτήσεων

```
@foreach (var question in Model)
{
    <tr>
        <td>@question.QuestionText</td>
        <td>
            <ul>
                @foreach (var answer in question.Answers)
                {
                    <li>@answer.Text (Correct: @answer.IsCorrect)</li>
                }
            </ul>
        </td>
        <td>
            @question.QuestionCategoryId
        </td>
        <td>
            <a asp-controller="QuestionCreate" asp-action="Details" asp-route-id="@question.QuestionId" class="btn btn-info">Details</a>
        </td>
        <td>
            <a asp-controller="QuestionCreate" asp-action="Delete" asp-route-id="@question.QuestionId" class="btn btn-danger">Delete</a>
        </td>
    </tr>
}
```

Σχήμα 3.13: Κώδικας σελίδας Ερωτήσεων

Στην συνέχεια όταν ο χρήστης κάνει κλικ στο κουμπί “Create Questions and Answers” τον ανακατευθύνει στην σελίδα «/QuestionCreate/Create». Στην σελίδα “Create” ο χρήστης έχει την δυνατότητα όπως παρουσιάζεται και στο Σχήμα 3.14 να συμπληρώσει τα πεδία “QuestionText” όπου είναι το κείμενο της ερώτησης, “QuestionPoints” και “NegativePoints” όπου είναι τα πεδία του βαθμού που θα έχει η ερώτηση αν είναι σωστή και ποια θα είναι αντίστοιχα η αρνητική βαθμολογία σε περίπτωση λανθασμένης απάντησης. Επίσης διαλέγει σε ποια κατηγορία θέλει να κατατάξει την συγκεκριμένη ερώτηση καθώς και σε ποιο πακέτο ερωτήσεων. Τέλος ο χρήστης μπορεί να προσθέσει δύο ή και παραπάνω απαντήσεις για την κάθε ερώτηση πατώντας το κουμπί “Add Answer”. Εφόσον ο χρήστης συμπληρώσει την φόρμα, κάνει κλικ στο κουμπί “Create Question and Answers” καλείται η μέθοδος “Post” της φόρμας του Παραρτήματος Α: Σελίδα Δημιουργίας Ερωτήσεων/Απαντήσεων. Έτσι αποθηκεύεται η φόρμα και ανακατευθύνει τον χρήστη πίσω στην σελίδα “Index”.



Σχήμα 3.14: Φόρμα Ερωτήσεων/Απαντήσεων

Όταν ο χρήστης κάνει κλικ στο κουμπί “Add Answers” για να προσθέσει έξτρα πεδίο απάντησης εκτελείται η συνάρτηση JavaScipt “addAnswer()” όπως φαίνεται και στο Σχήμα 3.15. Η συνάρτηση βρίσκει το στοιχείο με το “id” “answers-container” που θα προστεθούν οι καινούριες απαντήσεις. Στην συνέχεια υπολογίζει τον αριθμό των απαντήσεων που έχουν προστεθεί για να δημιουργήσει το επόμενο “index” μέσω του “var index = container.children().length”. Έπειτα γίνεται ένας έλεγχος αν κάποια από τις απαντήσεις που προστέθηκαν έχει επιλεγεί ως σωστή. Στο τελευταίο μέρος της συνάρτησης δημιουργεί το νέο HTML “var newAnswerHtml” και το προσθέτει στο “container”.

```
function addAnswer() {
    var container = $('#answers-container');
    var index = container.children().length;
    var isChecked = $(".answer-checkbox").is(":checked") ? "true" : "false";
    console.log(isChecked);
    // Κατασκευή νέας απάντησης ως HTML
    var newAnswerHtml = `
                <div class="form-group">
                    <label for="Answers_${index}__Text" class="control-label">Answer ${index + 1}</label>
                    <input type="text" name="Answers[${index}].Text" class="form-control" />
                    <label>
                        <input type="checkbox" class="answer-checkbox" name="Answers[${index}].IsCorrect" value="true" />
                        Correct Answer
                    </label>
                </div>
            `;

    // Προσθήκη της νέας απάντησης στο container
    container.append(newAnswerHtml);
}
```

Σχήμα 3.15: Συνάρτηση addAnswer()

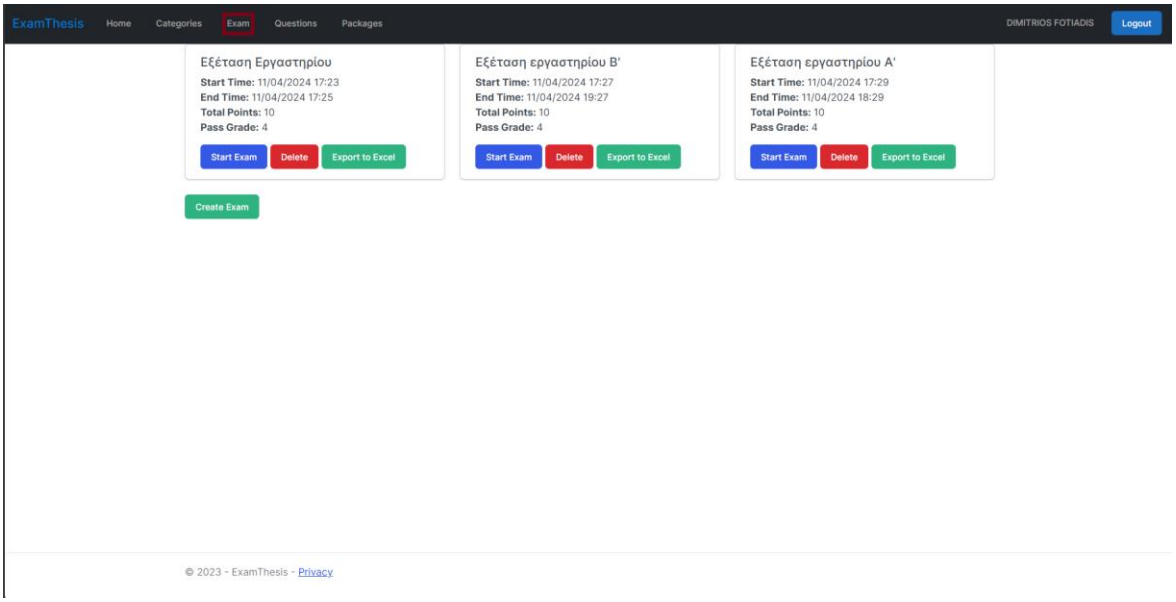
Όταν ο χρήστης κάνει κλικ στο κουμπί “Remove Answer” καλείται η συνάρτηση “removeAnswer()” του Σχήματος 3.16 όπου διαγράφει την τελευταία καταχώρηση απάντησης που δημιούργησε ο χρήστης με την βοήθεια του “container.children().last().remove()”.

```
function removeAnswer() {
    var container = $('#answers-container');
    var childrenCount = container.children().length;

    if (childrenCount > 1) {
        container.children().last().remove();
    }
}
```

Σχήμα 3.16: Συνάρτηση removeAnswer()

3.4.4 Σελίδα Εξέτασης

Όταν ο χρήστης κάνει κλικ στο κουμπί του μενού “Exam” ανακατευθύνεται στην σελίδα «/Exam/Index» του Σχήματος 3.17 όπου υπάρχει η λίστα των εξετάσεων με τις επιλογές δημιουργίας εξέτασης, διαγραφής εξέτασης καθώς και το κουμπί “Export to Excel” όπου δίνει την δυνατότητα στον χρήστη της εξαγωγής των αποτελεσμάτων των φοιτητών. Για να εμφανισθεί η λίστα των εξετάσεων δημιουργείται μία “for loop” του Σχήματος 3.18 όπου εκτελείται για κάθε “exam” στο “Model” και εμφανίζει το όνομα της εξέτασης, την ημερομηνία έναρξης και λήξης, τον μέγιστο βαθμό της εξέτασης καθώς και την ελάχιστη βάση επιτυχίας. Οι επιλογές “Create Exam”, “Delete”, “Export To Excel” τις ελέγχεται με την “User.IsInRole()” όπου αν ο χρήστης είναι καθηγητής τις εμφανίζει σαν επιλογή αλλιώς της απενεργοποιεί.

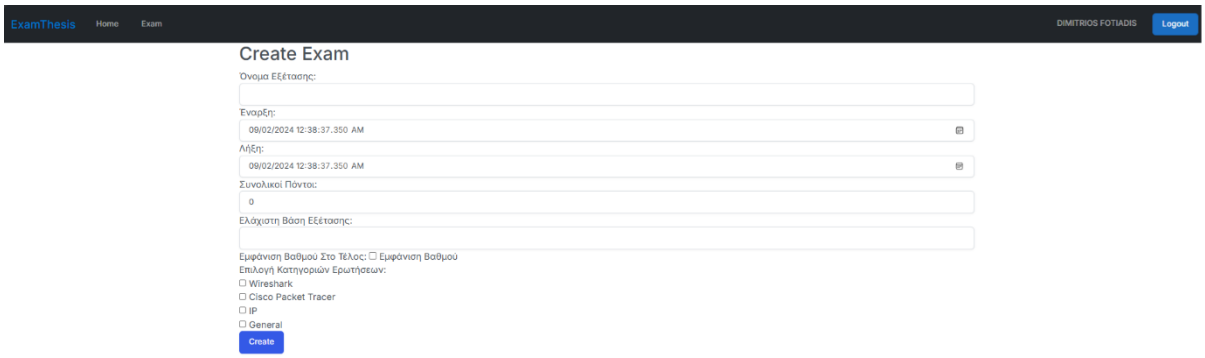


Σχήμα 3.17: Σελίδα Λίστας Εξετάσεων

```
@foreach (var exam in Model)
{
    <div class="col-md-4 mb-4">
        <div class="card">
            <div class="card-body">
                <h5 class="card-title">@exam.ExamName</h5>
                <p class="card-text">
                    <strong>Start Time:</strong> @exam.StartTime.ToString("dd/MM/yyyy HH:mm")<br />
                    <strong>End Time:</strong> @exam.EndTime.ToString("dd/MM/yyyy HH:mm")<br />
                    <strong>Total Points:</strong> @exam.TotalPoints<br />
                    <strong>Pass Grade:</strong> @exam.PassGrade
                </p>
```

Σχήμα 3.18: Κώδικας Σελίδας Εξετάσεων

Στην συνέχεια όταν ο χρήστης θέλει να δημιουργήσει μία εξέταση και κάνει κλικ στο κουμπί “Create Exam” ανακατευθύνετε στην σελίδα «/Exam/Create» τους Σχήματος 3.19. Εκεί ο χρήστης θα πρέπει να συμπληρώσει τα απαραίτητα πεδία όπως το όνομα της εξέτασης, την ημερομηνία έναρξης και λήξης της εξέτασης, τον μέγιστο βαθμό της εξέτασης, τον βαθμό της ελάχιστης βάσης επιτυχίας, ποιες κατηγορίες θέλει να υπάρχουν οι ερωτήσεις της εξέτασης όπου εμφανίζονται μία “for loop” του Σχήματος 3.20 που τυπώνει όλες τις υπάρχουσες κατηγορίες στο “Model” και εάν θέλει ο τελικός βαθμός να εμφανίζεται στο τέλος της εξέτασης στον φοιτητή. Όταν συμπληρώσει τα πεδία και κάνει κλικ στο κουμπί “Create” τότε καλείται η “Post” μέθοδος της φόρμας όπου αποθηκεύεται η καινούρια εξέταση και ανακατευθύνεται ο χρήστης στην σελίδα “Index” με την ανανεωμένη λίστα εξετάσεων.

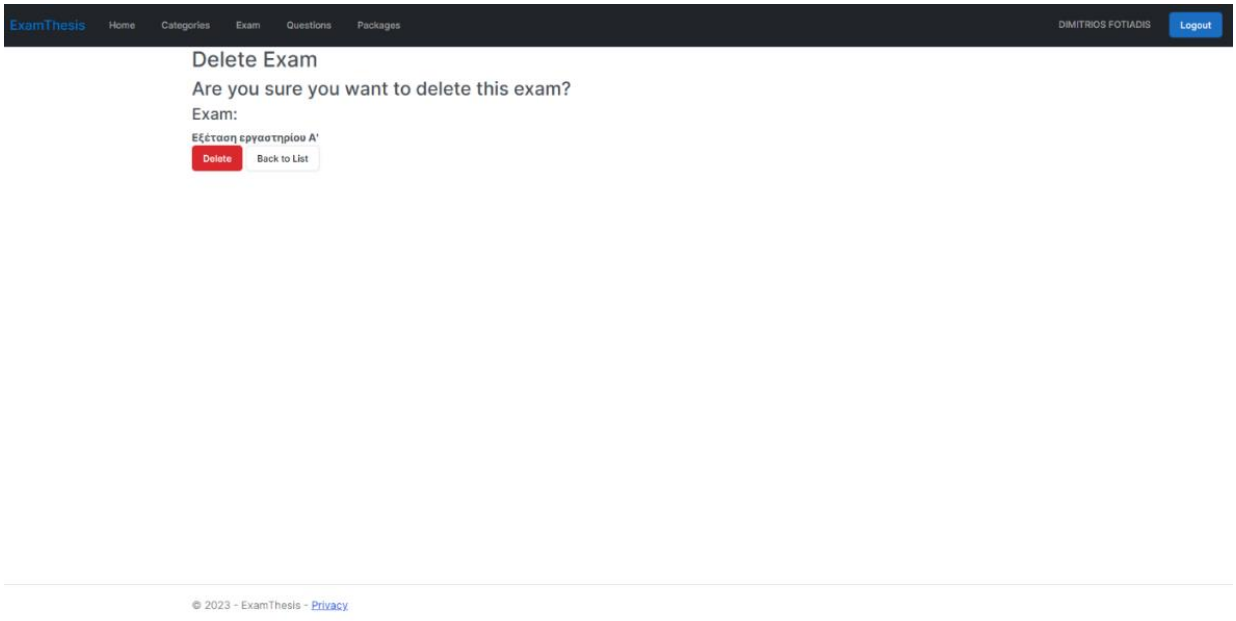


Σχήμα 3.19: Δημιουργία Εξέτασης

```
@for (var i = 0; i < Model.SelectedCategories.Count; i++)
{
    var currentcategory =Model.SelectedCategories[i];

   @*  <div class="form-group">
        <input type="checkbox" asp-for="SelectedCategories[]"
               name="@currentcategory.QuestionCategoryName"> @currentcategory.QuestionCategoryName

    </div> *@
    <div class="form-group">


            <input type="checkbox" asp-for="SelectedCategories[i].IsChecked"  />
            @currentcategory.QuestionCategoryName
            <input id="@currentcategory.QuestionCategoryId" asp-for="SelectedCategories[i].QuestionCategoryId" type="hidden"
        <input id="@currentcategory.QuestionCategoryName" asp-for="SelectedCategories[i].QuestionCategoryName" type="hidden"
    </div>
}
```

Σχήμα 3.20: Κώδικας Σελίδας Δημιουργίας

Όταν ο χρήστης κάνει κλικ στο κουμπί “Delete” στην εξέταση που θέλει να διαγράψει ανακατευθύνεται στην σελίδα «/Exam/Delete» του Σχήματος 3.21 όπου εκεί του εμφανίζεται η σελίδα με το όνομα της εξέτασης. Εφόσον κάνει κλικ στο κουμπί “Delete” καλείται η συνάρτηση με το “id” του κουμπιού “deleteButton” του Σχήματος 3.22 όπου εμφανίζει ένα μήνυμα επιβεβαίωσης διαγραφής. Μόλις ο χρήστης κάνει κλικ στο κουμπί “OK” τότε αποστέλλεται ένα αίτημα “Ajax” τύπου “DELETE” στο url «/Exam/DeleteConfirmed/», η εξέταση διαγράφεται και ο χρήστης ανακατευθύνεται στην σελίδα “Index” με την ανανεωμένη λίστα των εξετάσεων.



Σχήμα 3.21: Σελίδα Διαγραφής Εξέτασης

```
$(function () {
    $('#deleteButton').click(function () {
        if (confirm("Are you sure you want to delete this category?")) {
            $.ajax({
                url: '/Exam/DeleteConfirmed/' + @Model.ExamId,
                type: 'DELETE',
                headers: {
                    'RequestVerificationToken': $('input[name="__RequestVerificationToken"]').val()
                },
                success: function (data) {
                    console.log('Success:', data);
                    window.location.href = '/Exam/Index';
                },
                error: function (error) {
                    console.error('Error:', error);
                }
            });
        }
    });
});
```

Σχήμα 3.22: Κώδικας Σελίδας Διαγραφής

Μετά τις διαδικασίες δημιουργίας και διαγραφής της εξέτασης, δίνεται η δυνατότητα στον χρήστη να εξάγει τα αποτελέσματα των εξετάσεων βλέποντας το αναγνωριστικό της εξέτασης το αναγνωριστικό του φοιτητή καθώς και τον βαθμό του. Όταν ο χρήστης κάνει κλικ στο κουμπί “Export To Excel” κατεβαίνει στον υπολογιστή του ένα αρχείο Excel του Σχήματος 3.23 με τα στοιχεία που αναφέρθηκαν παραπάνω.

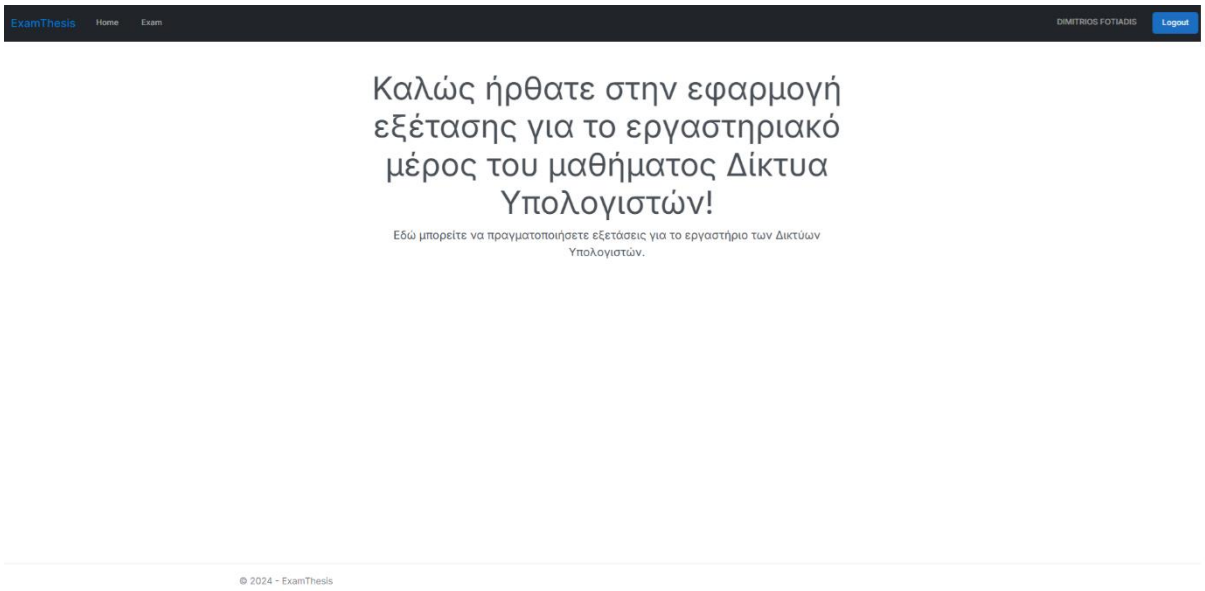
	A	B	C	D	E
1	StudentId	ExamId	Grade		
2	it164772	16	6.5		
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					

Σχήμα 3.23: Μορφή Αρχείου Excel

3.5 Προβολή Φοιτητή

Ό φοιτητής σε αντίθεση με τον καθηγητή έχει περιορισμένες δυνατότητες στην πλατφόρμα εξέτασης. Η πρόσβαση του περιορίζεται μόνο στην προβολή της αρχικής σελίδας και στην σελίδα με την λίστα των εξετάσεων μέσω του μενού του Σχήματος 3.24.

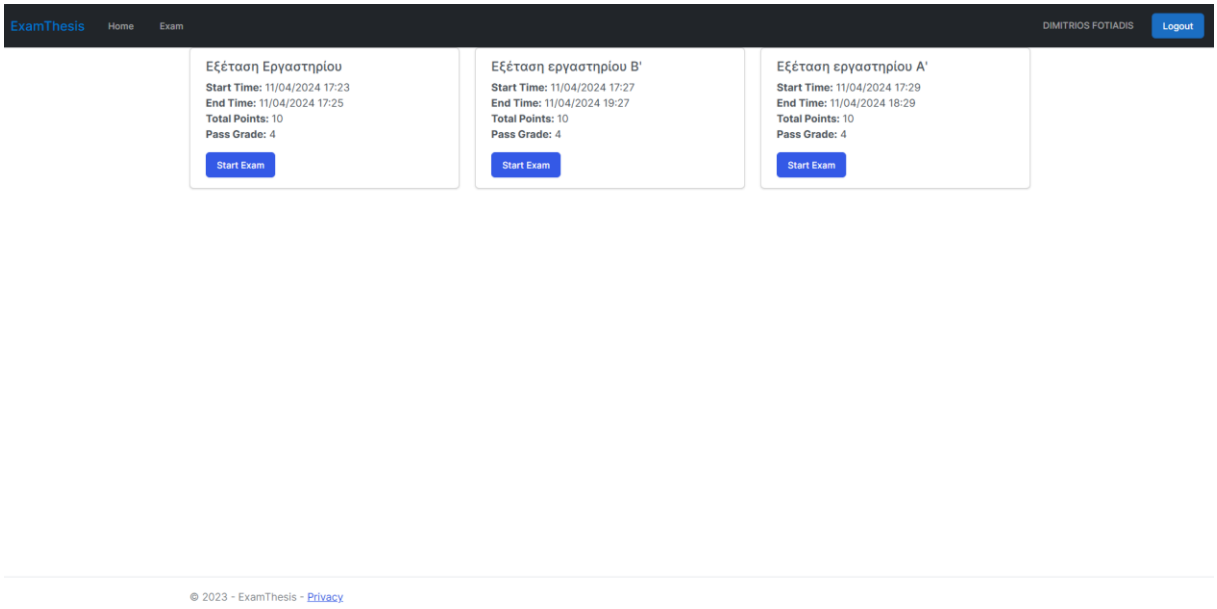
- Home
- Exam



Σχήμα 3.24: Προβολή Μενού Φοιτητή

3.5.1 Σελίδα Εξέτασης

Όταν ο χρήστης κάνει κλικ στο κουμπί του μενού “Exam” ο χρήστης ανακατευθύνεται στην σελίδα «Exam/Index» του Σχήματος 3.25. Στην σελίδα των εξετάσεων ο χρήστης μπορεί να δει τις υπάρχουσες εξετάσεις με τις πληροφορίες όπως το όνομα της εξέτασης, η ημερομηνία έναρξης και λήξης της εξέτασης, την μέγιστη βαθμολογία της εξέτασης καθώς και την ελάχιστη βάση επιτυχίας.

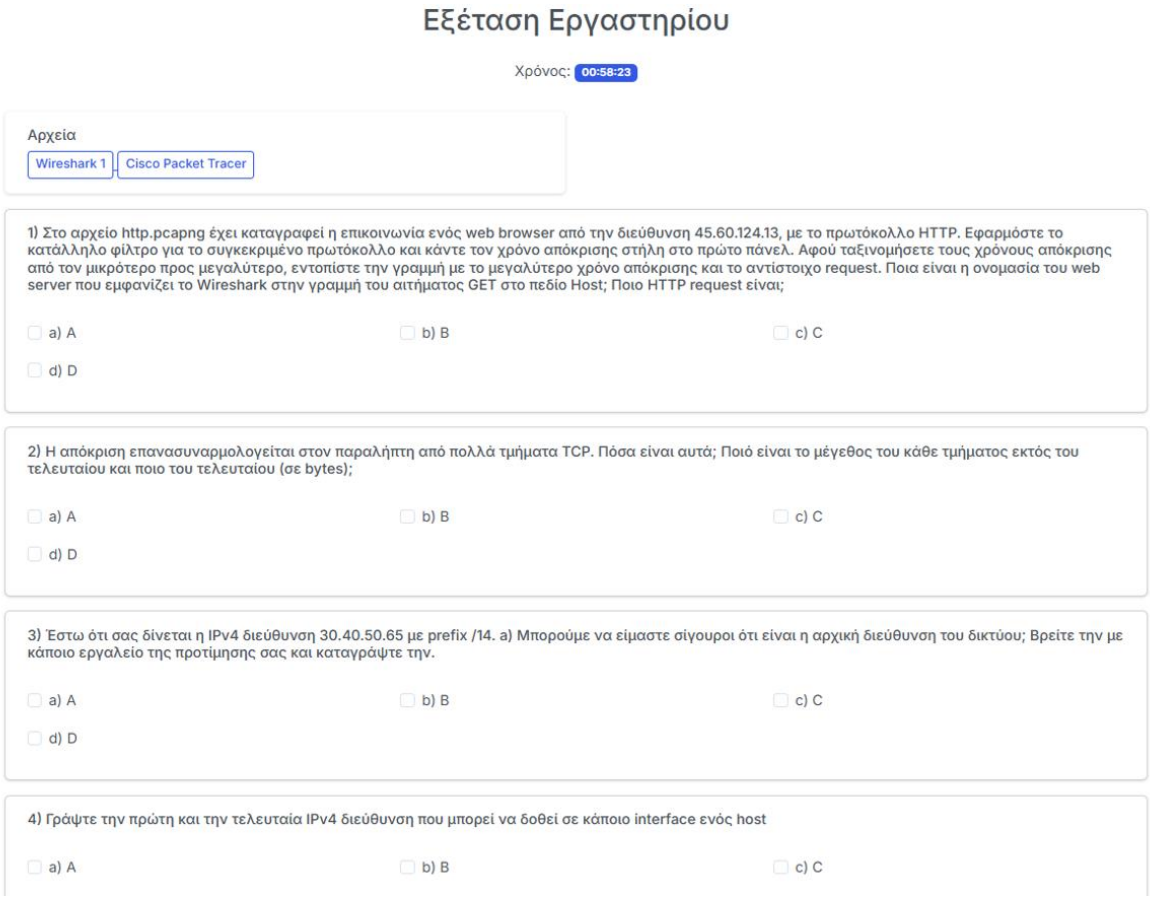


Σχήμα 3.25: Σελίδα Λίστας Εξετάσεων

```
<p class="card-text">
    <strong>Start Time:</strong> @exam.StartTime.ToString("dd/MM/yyyy HH:mm")<br />
    <strong>End Time:</strong> @exam.EndTime.ToString("dd/MM/yyyy HH:mm")<br />
    <strong>Total Points:</strong> @exam.TotalPoints<br />
    <strong>Pass Grade:</strong> @exam.PassGrade
</p>
```

Σχήμα 3.26: Κώδικας Σελίδας Εξετάσεων

Εφόσον ο χρήστης βρει την εξέταση του και κάνει κλικ στο κουμπί “Start Exam” ανακατευθύνεται στην σελίδα “Exam” του Σχήματος 3.27. Όταν ο χρήστης εισέλθει στην εξέταση βρίσκει τον εναπομείναντα χρόνο της εξέτασης όπου υπολογίζεται με βάσει την ώρα έναρξης και λήξης με την συνάρτηση του Σχήματος 3.28, τα βοηθητικά αρχεία που χρειάζονται για την επίλυση των ερωτήσεων, τις ερωτήσεις πολλαπλών απαντήσεων καθώς και το κουμπί υποβολής των απαντήσεων. Στην συνέχεια όταν ο χρήστης απαντήσει τις ερωτήσεις, για την υποβολή των απαντήσεων κάνει κλικ στο κουμπί “Submit Exam” καλείται η “Post” μέθοδος της φόρμας για την υποβολή των απαντήσεων.



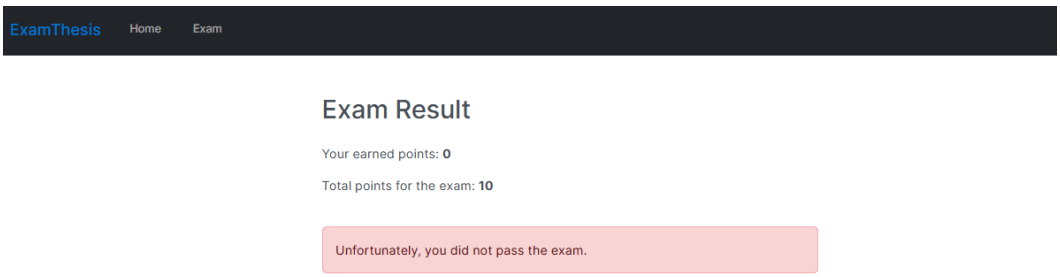
Σχήμα 3.27: Παράδειγμα Σελίδας Εξέτασης

```
function formatTime(milliseconds) {
    var hours = Math.floor(milliseconds / (1000 * 60 * 60));
    var minutes = Math.floor((milliseconds % (1000 * 60 * 60)) / (1000 * 60));
    var seconds = Math.floor((milliseconds % (1000 * 60)) / 1000);

    // Προσθήκη μηδέν μπροστά αν η τιμή είναι μικρότερη από 10
    var formattedHours = hours < 10 ? '0' + hours : hours;
    var formattedMinutes = minutes < 10 ? '0' + minutes : minutes;
    var formattedSeconds = seconds < 10 ? '0' + seconds : seconds;

    return formattedHours + ":" + formattedMinutes + ":" + formattedSeconds;
}

// Υπολογίζουμε τον εναπομείναντα χρόνο
var remainingTime = endTime - new Date();

// Εμφανίζουμε τον εναπομείναντα χρόνο στο στοιχείο HTML με id "timer"
$("#timer").text(formatTime(remainingTime));
```

Σχήμα 3.28: Υπολογισμός Χρονομέτρου

3.5.2 Σελίδα Αποτελεσμάτων

Όταν γίνει η υποβολή των απαντήσεων από την χρήστη, ανακατευθύνεται στην σελίδα «/Exam/Results» του Σχήματος 3.29. Στην σελίδα αυτή ο χρήστης μπορεί να δει τον τελικό του βαθμό και ένα μήνυμα επιτυχίας ή αποτυχίας . Αν ο βαθμός του φοιτητή είναι μεγαλύτερος από την ελάχιστη βάση επιτυχίας τότε του εμφανίζεται το μήνυμα ότι έχει επιτύχει στις εξετάσεις αντίστοιχα αν ο βαθμός του είναι μικρότερος τότε του εμφανίζει μήνυμα αποτυχίας μέσω της “ViewBag.Passed” του Σχήματος 3.30.



Σχήμα 3.29: Σελίδα Αποτελέσματος

```
<div class="container">
    <h2 class="mt-5">Exam Result</h2>

    <div class="row mt-4">
        <div class="col-md-6">
            @if (ViewBag.ShowGrade)
            {
                <p>Your earned points: <strong>@ViewBag.EarnedPoints</strong></p>
                <p>Total points for the exam: <strong>@ViewBag.TotalPoints</strong></p>
            }
        </div>
    </div>

    <div class="row mt-4">
        <div class="col-md-6">
            @if (ViewBag.Passed)
            {
                <div class="alert alert-success" role="alert">
                    Congratulations! You passed the exam.
                </div>
            }
            else
            {
                <div class="alert alert-danger" role="alert">
                    Unfortunately, you did not pass the exam.
                </div>
            }
        </div>
```

Σχήμα 3.30: Κώδικας Σελίδας Αποτελεσμάτων

3.6 Επίλογος

Σε αυτό το κεφάλαιο αναλύθηκαν αρχικά οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη του Front-End μέρους της εφαρμογής. Έπειτα περιληπτικά παρουσιάστηκαν οι λειτουργίες της εφαρμογής καθώς και μέρος του κώδικα των σελίδων

Κεφάλαιο 4ο: Back-End

4.1 Εισαγωγή

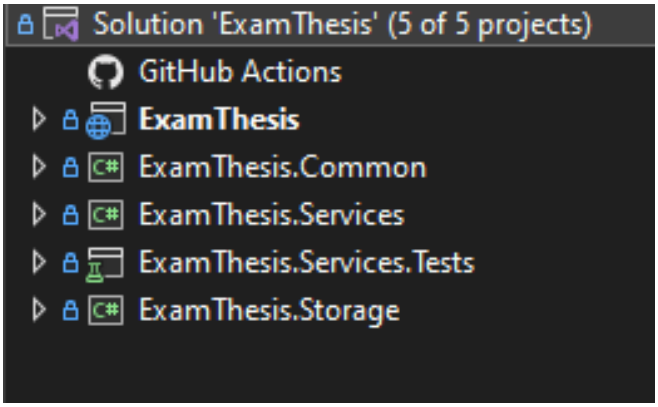
Για την ανάπτυξη του Back-End μέρους του αυτοματοποιημένου συστήματος εξέτασης διαλέξαμε όπως και στο Front-End το Full-Stack ASP.NET Core MVC. Η επιλογή της τεχνολογίας έπρεπε να έρχεται με κάποια απαραίτητα χαρακτηρίστηκα όπως:

- **Επίδοση**: Το ASP.NET Core είναι γνωστό για τις υψηλές του αποδόσεις σε απαιτητικές διαδικτυακές εφαρμογές όπως το αυτοματοποιημένο σύστημα εξέτασης που θα χρησιμοποιείται από μεγάλο όγκο φοιτητών.
- **Ασφάλεια**: Δίνει την δυνατότητα με ενσωματωμένες λύσεις για επιθέσεις και παρέχει πρωτόκολλα ασφαλείας όπως το HTTPS. Επίσης δίνει την δυνατότητα για διαχείριση χρηστών όπως είδαμε και στο 3ο Κεφάλαιο.
- **Δομή**: Το ASP.NET Core υποστηρίζει την αρχιτεκτονική MVC που διαχωρίζει την εφαρμογή σε επίπεδα μοντέλου, προβολής και ελεγκτών για την καλύτερη διαχείριση καθώς επίσης και παρέχει και υποστήριξη για Dependency Injection που βελτιώνει την διαχείριση των εξαρτήσεων.
- **Κοινότητα/Υποστήριξη**: Το ASP.NET Core παρέχει μια μεγάλη κοινότητα προγραμματιστών καθώς και την υποστήριξη ενός κολοσσού όπως η εταιρεία Microsoft.

4.2 Αρχιτεκτονική Εφαρμογής

Για την εφαρμογή επιλέχθηκε μία πολύ-επίπεδη αρχιτεκτονική (layered architecture). Αυτό βοηθάει στην καλύτερη οργάνωση του κώδικα την ευκολία στην συντήρηση και στην επαναχρησιμοποίηση του. Υπάρχουν τέσσερα projects του Σχήματος 4.1:

- ExamThesis
- ExamThesis.Services
- ExamThesis.Common
- ExamThesis.Storage



Σχήμα 4.1: Δομή Εφαρμογής

Όταν δημιουργείται μια εφαρμογή ASP.NET MVC, δεν πρέπει να τοποθετείται η λογική της βάσης δεδομένων μέσα στις ενέργειες του ελεγκτή (Controller). Η ανάμειξη της βάσης δεδομένων και της λογικής του ελεγκτή κάνει την εφαρμογή πιο δύσκολη στη συντήρηση με την πάροδο του χρόνου. Η

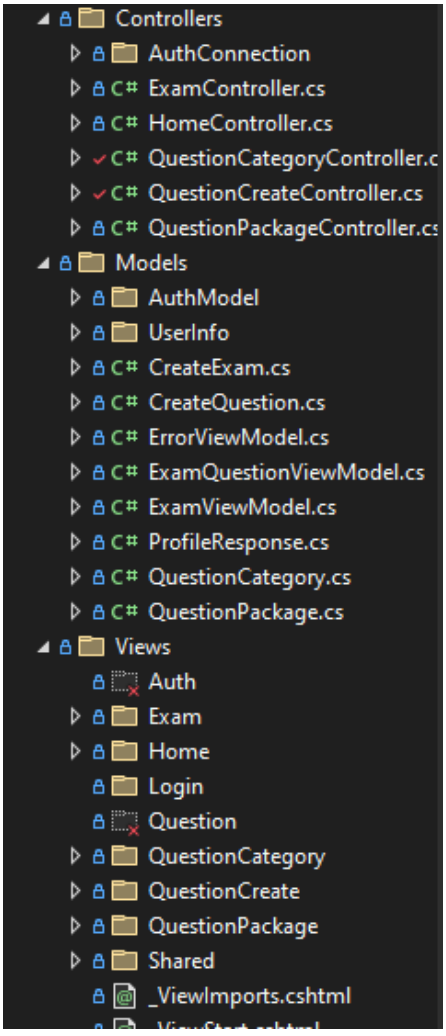
προτεινόμενη τακτική είναι να τοποθετείται όλη η λογική της βάσης δεδομένων σε ένα ξεχωριστό επίπεδο.

4.2.1 ExamThesis (Επίπεδο Εφαρμογής)

Στο επίπεδο εφαρμογής όπως αναφέρεται και στο Σχήμα 4.2 χρησιμοποιήθηκε η αρχιτεκτονική MVC όπου βοηθάει στο να χωριστεί αρχικά σε τρία μέρη τον project.

Τα μέρη αυτά είναι το Model,View,Controller:

- **Controllers**: Οι Controllers είναι υπεύθυνοι για την επεξεργασία και τον χειρισμό των αιτημάτων αλλά και για την επεξεργασία των δεδομένων.
- **View**: Το View όπως είδαμε και στο 3ο κεφάλαιο της ανάπτυξης του Front-end είναι υπεύθυνο για την προβολή στον χρήστη.
- **Model**: Τα Models χρησιμοποιούνται για την μεταφορά δεδομένων από τους Controllers στο View.

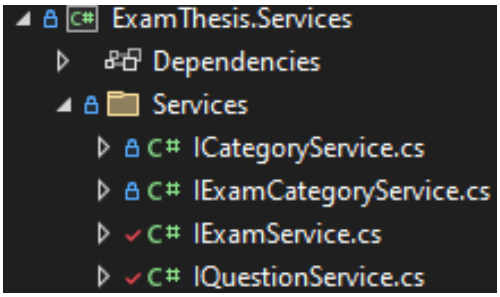


Σχήμα 4.2: Αρχιτεκτονική MVC ExamThesis

4.2.2 ExamThesis.Services (Επίπεδο Υπηρεσιών)

Ένα επίπεδο υπηρεσίας είναι ένα πρόσθετο επίπεδο σε μια εφαρμογή ASP.NET MVC που μεσολαβεί στην επικοινωνία μεταξύ ενός ελεγκτή και του επιπέδου Storage. Το επίπεδο υπηρεσιών περιέχει επιχειρηματική λογική. Συγκεκριμένα, περιέχει την λογική επικύρωσης. Οι υπηρεσίες αυτές είναι ενσωματωμένες μέσω του Dependency Injection (DI) όπου επιτρέπει την απλή αντικατάσταση των υπηρεσιών χωρίς να χρειάζεται να γίνει καμία αλλαγή στους Controllers. Έτσι το επίπεδο εφαρμογής μένει ανεπηρέαστο από τυχόν αλλαγές στην λογική. Στο Συγκεκριμένο επίπεδο δημιουργήθηκαν τέσσερα Interfaces (Διεπαφές) του Σχήματος 4.3:

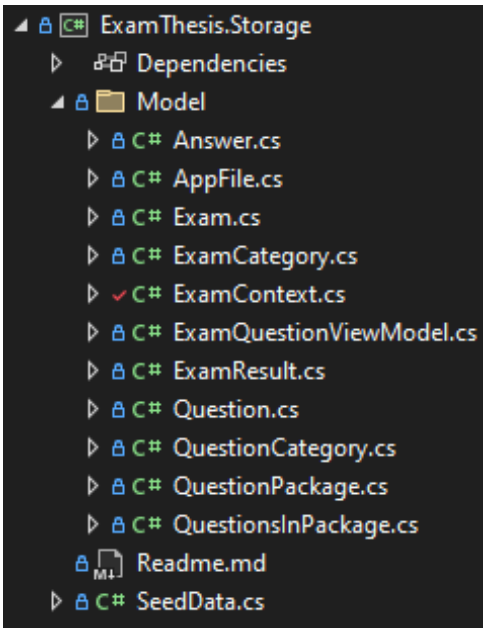
- **ICategoryService**: το οποίο είναι υπεύθυνο για τις λειτουργίες δημιουργίας διαγραφής των κατηγοριών
- **IExamCategoryService**: το οποίο είναι υπεύθυνο για την διαχείριση της λογικής στις επιλεγμένες κατηγορίες σε μία εξέταση
- **IExamService**: το συγκεκριμένο Interface είναι υπεύθυνο για την διαχείριση της λογικής στο ευρύ φάσμα των λειτουργιών των εξετάσεων. Όπως η δημιουργία, η διαγραφή αλλά και το πώς χειρίζεται την εξέταση από που αντλούνται οι ερωτήσεις κλπ.
- **IQuestionService**: το οποίο είναι υπεύθυνο για την διαχείρηση της λογικής και των αιτημάτων για την δημιουργία αλλα και διαγραφή των ερωτήσεων αλλα και των πακέτων ερωτήσεων



Σχήμα 4.3: Interfaces ExamThesis.Services

4.2.3 ExamThesis.Storage (Επίπεδο Αποθήκης)

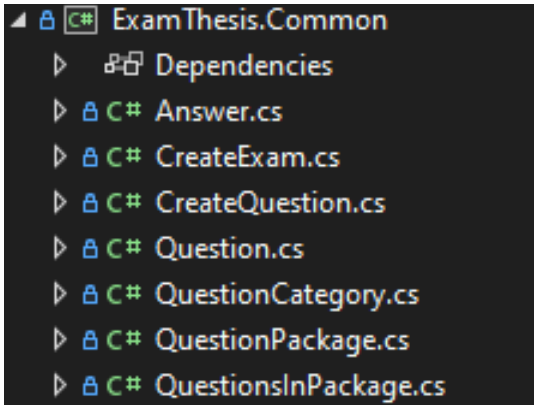
Το ExamThesis.Storage περιέχει τα μοντέλα για την επικοινωνία με την βάση δεδομένων του Σχήματος 4.4. Σε αυτό το project υπάρχουν οι κλάσεις που αντιπροσωπεύουν τα entities της βάσης δεδομένων. Αυτό γίνεται για τον διαχωρισμό των μοντέλων του επιπέδου εφαρμογής και του επιπέδου αποθήκης έτσι ώστε το επίπεδο εφαρμογής να μην έχει καμία ανάμειξη με την βάση δεδομένων. Μέσω του επιπέδου αυτού δημιουργήθηκαν τα αντίστοιχα μοντέλα προβολής στην συνέχεια.



Σχήμα 4.4: ExamThesis.Storage Models

4.2.4 ExamThesis.Common (Ενδιάμεσο Επίπεδο Μοντέλου)

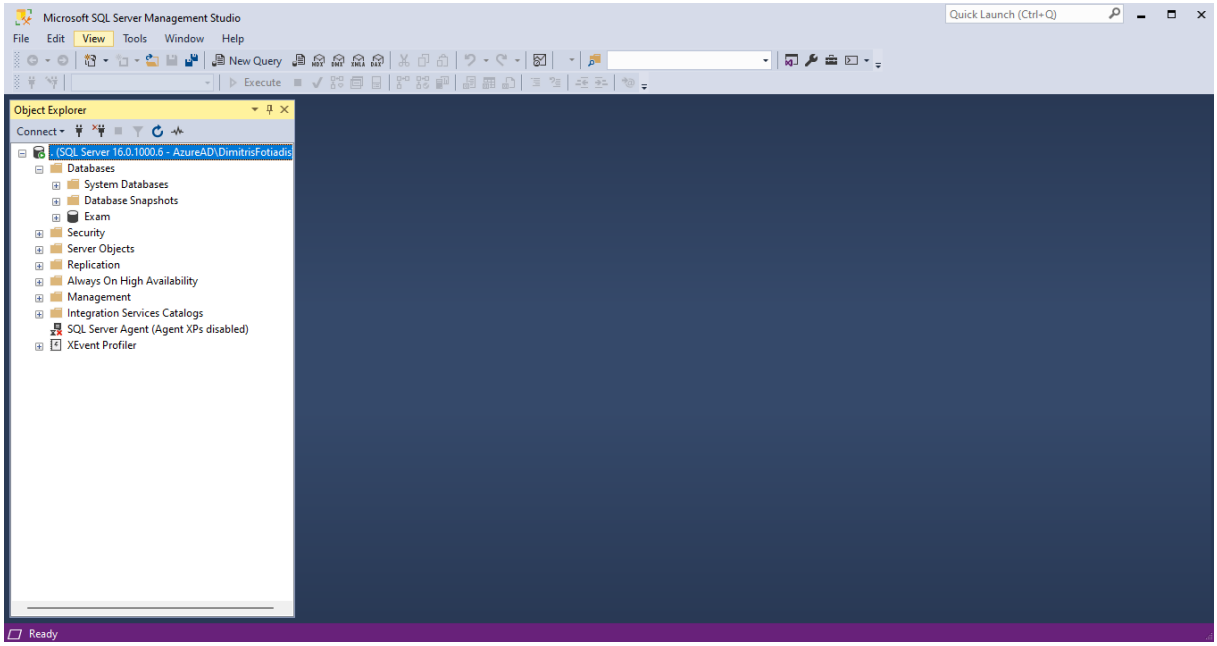
Το ExamThesis.Common χρησιμοποιείται ως ενδιάμεσο επίπεδο το οποίο παρέχει κοινά μοντέλα του Σχήματος 4.5 με τα υπόλοιπα Projects. Αυτό γίνεται για να βοηθάει στην επαναχρησιμοποίηση του κώδικα καθώς και σε περίπτωση αλλαγών δεν θα επηρεάζεται η υπόλοιπη εφαρμογή.



Σχήμα 4.5: ExamThesis.Common Models

4.3 Σχεδιασμός και Σύνδεση Βάσης Δεδομένων

Για την βάση δεδομένων χρησιμοποιήσαμε τον Microsoft SQL Server όπου είναι ένα από τα πιο δημοφιλή και αξιόπιστα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων και είναι ιδανικό για διαδικτυακές εφαρμογές. Για την δημιουργία της βάση δεδομένων χρειάσθηκε η εφαρμογή SQL Server Management (SMSS) η οποία δίνει την δυνατότητα του γραφικού περιβάλλοντος του Σχήματος 4.6.



Σχήμα 4.6: Microsoft SQL Server Management Studio (SSMS)

4.3.1 Σύνδεση Βάσης Δεδομένων

Για σύνδεση της βάσης δεδομένων με το πρόγραμμα καθορίσθηκαν στο αρχείο “appsettings.json” οι ρυθμίσεις με ένα “ConnectionString” του Σχήματος 4.7 όπου το “AllowedHosts” το χρησιμοποιείται για να καθοριστεί ποιοι είναι οι Hosts που επιτρέπεται να έχουν πρόσβαση στην εφαρμογή όπου στην προκειμένη περίπτωση έχει την τιμή “*” που σημαίνει ότι δεν υπάρχει κάποιος περιορισμός πρόσβασης. Στην συνέχεια το “ConnectionString” περιέχει την σύνδεση με την βάσης. Στο “string” υπάρχουν κάποια μέρη όπως:

- **Server**: Αυτό το πεδίο ορίζει τον διακομιστή της βάσης όπου στην προκειμένη περίπτωση έχει την τιμή “.” που σημαίνει ότι ο διακομιστής βρίσκεται τοπικά
- **Database**: Σε αυτό το πεδίο ορίζεται η βάση δεδομένων που θα χρησιμοποιηθεί για το πρόγραμμα, όπου εδώ υπάρχει η βάση δεδομένων που δημιουργήθηκε με όνομα “Exam”
- **Trusted_Connection**: Αυτό το πεδίο όταν η τιμή του είναι “true” σημαίνει ότι υπάρχει η ασφάλεια των Windows για τον έλεγχο ταυτότητας.

```
"AllowedHosts": "*",
"ConnectionStrings": {
  "DefaultConnection": "Server=.;Database=Exam;Trusted_Connection=True;TrustServerCertificate=True;"
}
```

Σχήμα 4.7: Connection String Βάσης Δεδομένων

4.3.2 Σχεδιασμός Βάσης Δεδομένων

Για την σχεδίαση της βάσης δεδομένων χρησιμοποιήθηκε η τεχνική “Database First” όπου σχεδιάσθηκε η βάση κατευθείαν στον “Microsoft SQL Server Studio”. Έπειτα η βάση γίνεται “scaffold” με την βοήθεια του “Entity Framework” έτσι ώστε να δημιουργηθεί από του πίνακες με τα πεδία τους τα αντίστοιχα μοντέλα στην εφαρμογή. Αυτή η τεχνική είναι ιδανική διότι, ότι αλλαγή γίνεται στην βάση,

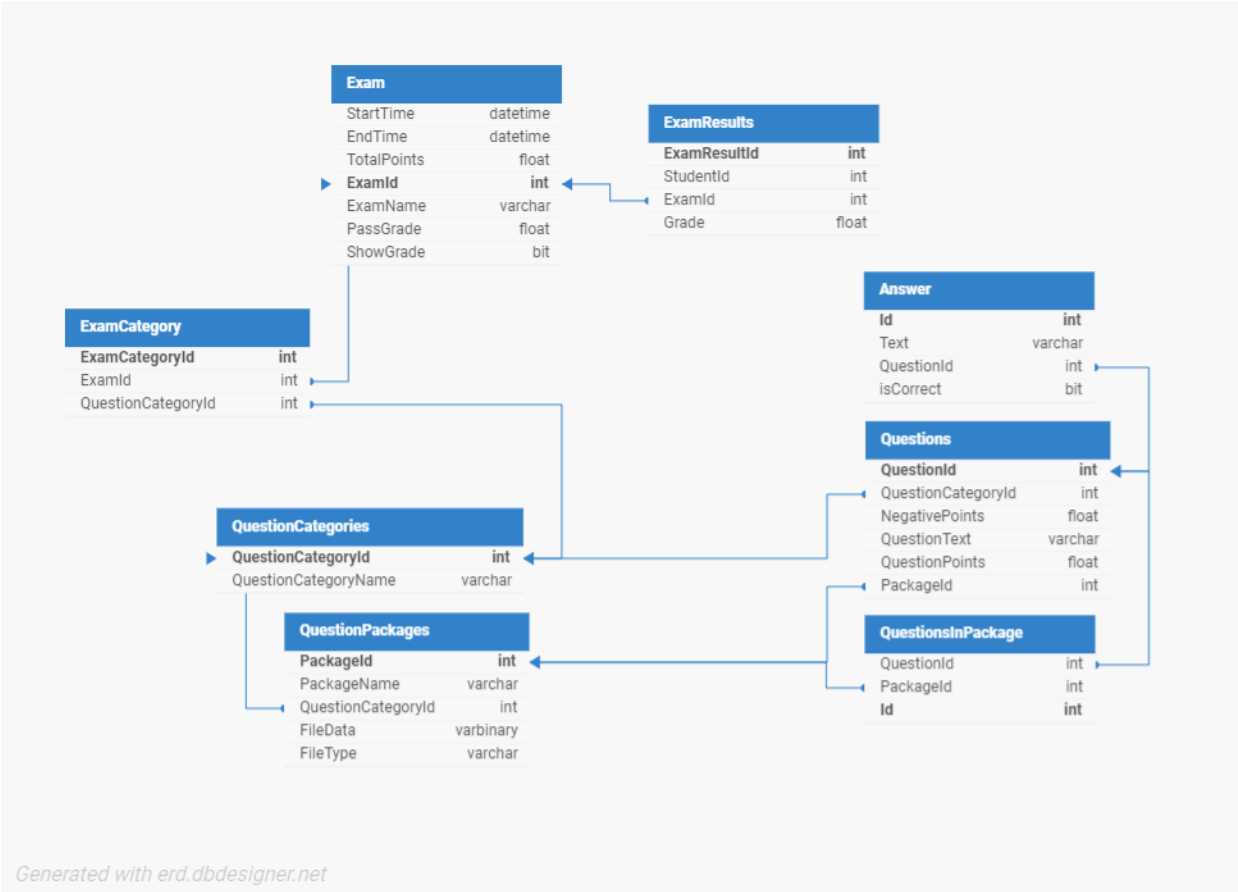
αυτόματα με την εντολή του Σχήματος 4.8 ανανεώνονται τα μοντέλα και έτσι το πρόγραμμα πάντα θα είναι ενημερωμένο χωρίς να υπάρχουν “Conflicts” μεταξύ της βάσης και των μοντέλων.

```
dotnet ef dbcontext scaffold "Server=.;Database=Exam;Trusted_Connection=True;TrustServerCertificate=True;"
Microsoft.EntityFrameworkCore.SqlServer -o Model
```

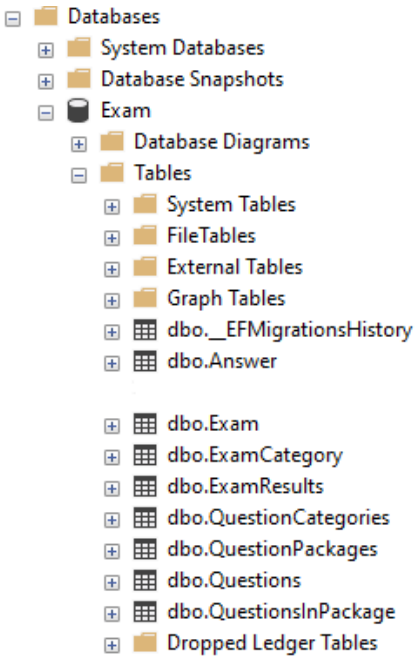
Σχήμα 4.8: Εντολή dotnet scaffold

Για να είναι πλήρως λειτουργικό το σύστημα εξέτασης δημιουργήθηκαν οι κατάλληλοι πίνακες με τα πεδία τους που βοηθάνε να αποθηκεύονται τα απαραίτητα δεδομένα. Με την βοήθεια ενός διαγράμματος ER του Σχήματος 4.9 σχεδιάσθηκαν οκτώ πίνακες του Σχήματος 4.10:

1. **Questions**: Στον πίνακα Questions αποθηκεύονται οι ερωτήσεις που δημιουργεί ο χρήστης με τα απαραίτητα πεδία του “*QuestionId,QuestionCategoryId,NegativePoints,QuestionPoints* και *PackageId*”
2. **Answer**: Στον πίνακα Answer αποθηκεύονται οι απαντήσεις των ερωτήσεων που δημιουργεί ο χρήστης με τα πεδία *Id,Text,QuestionId και IsCorrect*
3. **QuestionCategories**: Στον πίνακα *QuestionCategories* αποθηκεύονται οι κατηγορίες των ερωτήσεων με τα πεδία *QuestionCategoryId και QuestionCategoryName*
4. **QuestionPackages**: Στον πίνακα QuestionPackage αποθηκεύονται τα πακέτα ερωτήσεων που δημιουργεί ο χρήστης με τα πεδία *PackageId, PackageName, QuestionCategoryId, FileData και FileType*
5. **QuestionInPackages**: Στον πίνακα QuestionInPackages αποθηκεύονται οι ερωτήσεις στα πακέτα που διαλέγει ο χρήστης με τα πεδία *Id, PackageId και QuestionId.*
6. **Exam**: Στον πίνακα Exam αποθηκεύονται οι εξετάσεις που δημιουργεί ο χρήστης με τα πεδία *ExamId, StartTime ,EndTime, ExamName, TotalPoints , PassGrade και ShowGrade*
7. **ExamCategory**: Στον πίνακα ExamCategory αποθηκεύονται οι κατηγορίες που έχουν επιλεχθεί από τον χρήστη κατά την δημιουργία μιας εξέτασης με τα πεδία *ExamCategoryId, ExamId και QuestionCategoryId.*
8. **ExamResults**: Στον πίνακα ExamResults αποθηκεύονται τα αποτελέσματα των εξετάσεων με τα πεδία *ExamResultsId, StudentId, ExamId και Grade.*



Σχήμα 4.9: Διάγραμμα Σχεδίασης Βάσης



Σχήμα 4.10: Πίνακες Βάσης Δεδομένων

4.3.3 Σύνδεσμος Βάσης Δεδομένων

Με την βοήθεια του EntityFramework (EF Core) δημιουργείθηκε η σύνδεση της βάσης με το πρόγραμμα. Αρχικά, δημιουργήθηκε μία κεντρική κλάση που περιέχει τις απαραίτητες υπηρεσίες για την αλληλεπίδραση με την βάση δεδομένων. Μόλις εκτελεστεί η εντολή του EF Core δημιουργείται αυτόματα το αρχείο ExamContext του Σχήματος 4.11 όπου αντιπροσωπεύει την βάση και περιλαμβάνει τα “DbSet” για κάθε πίνακα. Στην Συνέχεια δημιουργήθηκαν οι κλάσεις για κάθε πίνακα της βάσης δεδομένων με τα αντίστοιχα πεδία της. Με την μέθοδο “OnModelCreating” διαμορφώνει την συμπεριφορά των οντοτήτων και την δημιουργία των μοντέλων με τις σχεσιακές ιδιότητες που έχουν δημιουργθεί κατά την σχεδίαση της βάσης δεδομένων.

```
15 references
public virtual DbSet<QuestionCategory> QuestionCategories { get; set; }

8 references
public virtual DbSet<QuestionPackage> QuestionPackages { get; set; }

5 references
public virtual DbSet<QuestionsInPackage> QuestionsInPackages { get; set; }

0 references
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
warning To protect potentially sensitive information in your connection string, you should move it out
    => optionsBuilder.UseSqlServer("Server=.;Database=Exam;Trusted_Connection=True;TrustServerCertif

0 references
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Answer>(entity =>
    {
        entity.ToTable("Answer");

        entity.HasIndex(e => e.QuestionId, "IX_Answer_QuestionId");

        entity.HasIndex(e => e.Id, "idx_answer_id");

        entity.HasIndex(e => e.QuestionId, "idx_question_id");

        entity.Property(e => e.IsCorrect).HasColumnName("isCorrect");

        entity.HasOne(d => d.Question).WithMany(p => p.Answers).HasForeignKey(d => d.QuestionId);
    });

    modelBuilder.Entity<AppFile>(entity =>
    {
        entity.ToTable("AppFile");
    });

    modelBuilder.Entity<Exam>(entity =>
    {
        entity.HasKey(e => e.ExamId).HasName("PK__Exam__297521C7826CB490");

        entity.ToTable("Exam");

        entity.Property(e => e.EndTime).HasColumnType("datetime");
        entity.Property(e => e.ExamName).HasMaxLength(255);
        entity.Property(e => e.StartTime).HasColumnType("datetime");
    });
```

Σχήμα 4.11: ExamContext Κλάση

4.4 Μοντέλα Επιπέδου Αποθήκης

Αφού έγινε η σύνδεση με την βάση δεδομένων και στην συνέχεια δημιουργήθηκε με την εντολή του Σχήματος 4.8 το αρχείο “ExamContext , δημιουργήθηκαν επίσης και τα μοντέλα στο επίπεδο αποθήκης τα οποία είναι ένα σύνολο κλάσεων που περιγράφουν τα δεδομένα όπως φαίνεται και στο παράδειγμα Μοντέλου “Question” του Σχήματος 4.12. Το μοντέλο αντιπροσωπεύει τα δεδομένα και την επιχειρηματική λογική της εφαρμογής. Ενσωματώνει την κατάσταση και τη συμπεριφορά της εφαρμογής, παρέχοντας μια δομημένη προσέγγιση στη διαχείριση δεδομένων. Τα αντίστοιχα μοντέλα που έχουν δημιουργηθεί είναι:

1) Question
2) Answer
3) Exam
4) ExamCategory
5) ExamResults
6) QuestionCategory
7) QuestionPackage
8) QuestionInPackage

```
19 references
public partial class Question
{
    9 references
    public int QuestionId { get; set; }

    5 references
    public int QuestionCategoryId { get; set; }

    5 references
    public double NegativePoints { get; set; }

    8 references
    public string QuestionText { get; set; } = null!;

    5 references
    public double QuestionPoints { get; set; }

    8 references
    public int? PackageId { get; set; }

    10 references
    public virtual ICollection<Answer> Answers { get; set; } = new List<Answer>();

    1 reference
    public virtual QuestionPackage? Package { get; set; }

    2 references
    public virtual ICollection<QuestionsInPackage> QuestionsInPackages { get; set; } = new List<QuestionsInPackage>();
}
```

Σχήμα 4.12: Μοντέλο Question

4.5 Controllers & Services

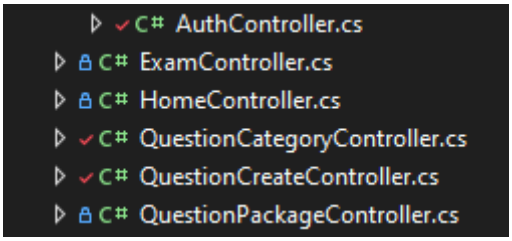
Η αρχιτεκτονική της εφαρμογής είναι πολύ-επίπεδη. Τους Controllers και τα Services στο επίπεδο εφαρμογής και υπηρεσίας χρειάστηκαν για να διαχειρίζονται την λογική και την αλληλεπίδραση τους με τα δεδομένα.

Controllers

Οι Controllers είναι υπεύθυνοι για την επεξεργασία των αιτημάτων (HTTP Post,Get,Delete) που στέλνει ο χρήστης και μέσω των Services να γίνεται η διαχείριση της λογικής για την επιστροφή των κατάλληλων αιτήσεων. Σε κάθε Controller του Σχήματος 4.13 βρίσκονται Action μεθόδους που διαχειρίζονται τις αιτήσεις όπου μπορούν να επιστρέψουν ένα View, κάποια JSON δεδομένα ή και να επιστρέψει τον χρήστη σε μία καινούρια σελίδα.

Οι Controllers που δημιουργήθηκαν στην εφαρμογή είναι:

1. AuthController
2. QuestionCategoryController
3. QuestionCreateController
4. QuestionPackageController
5. ExamController
6. HomeController



Σχήμα 4.13: Controllers

Services

Τα Services στο επίπεδο των υπηρεσιών αναλαμβάνουν την λογική για την εκτέλεση των λειτουργιών. Είναι υπεύθυνα για την διαχείριση των δεδομένων της εφαρμογής καθώς και για την επικοινωνία με την βάση. Ο διαχωρισμός αυτών των λειτουργιών γίνεται για την αποφόρτιση των Controllers και την επαναχρησιμοποίηση του κώδικα για μεταγενέστερες αλλαγές.

Για να χρησιμοποιηθούν τα Services χρειάζεται στο αρχείο “Program.cs” του Σχήματος 4.14 να δηλωθούν οι υπηρεσίες επιτρέποντας στο ASP.NET Core να τις δημιουργεί και να τις χειρίζεται αυτόματα . Στον κώδικα υπάρχει το αντικείμενο “builder.Services” το οποίο περιλαμβάνει όλες τις υπηρεσίες που πρέπει να δηλωθούν και με την βοήθεια του “AddTransient” καταχωρούνται .

Οι υπηρεσίες που δημιουργήθηκαν για τις ανάγκες της εφαρμογής είναι:

1. «IExamService, ExamService()»
2. «IExamCategoryService,ExamCategoryService()»
3. «IQuestionService,QuestionService()»
4. «ICategoryService,CategoryService()»

```
builder.Services.AddTransient<IExamService, ExamService>();
builder.Services.AddTransient<IExamCategoryService, ExamCategoryService>();
builder.Services.AddTransient<IQuestionService, QuestionService>();
builder.Services.AddTransient<ICategoryService, CategoryService>();
```

Σχήμα 4.14: Δήλωση Υπηρεσιών

1. AuthController (Σύνδεση και Διαχωρισμός Χρηστών)

Όταν ο χρήστης κάνει κλικ το κουμπί «Login» στην εφαρμογή τότε καλείται η μέθοδος του AuthController “Index” όπου αυτήν με την σειρά της καλεί την μέθοδο “GetUrl” του Σχήματος 4.15 και ανακατευθύνει τον χρήστη στο περιβάλλον εισόδου του «login.iee.ihu.gr». Η συγκεκριμένη μέθοδος δημιουργεί τον σύνδεσμο με τα όλα απαραίτητα στοιχεία που θα χρειαστούν όπως το “Client_id”, “Response_type”, “code”, “scope”, “state” και το “redirect_uri”.

```
[HttpGet]
0 references
public IActionResult Index()
{
    return Redirect(GetUrl());
}
1 reference
private string GetUrl()
{
    var url = $"https://login.iee.ihu.gr/login?client_id={CLIENT_ID}&response_type=code&scope=profile&state=12345&redirect_uri=https://localhost:7134/Auth/Callback";
    return url;
}
```

Σχήμα 4.15: Μέθοδος Index και GetUrl

Αφού ολοκληρωθεί επιτυχώς η ταυτοποίηση του χρήστη το «login.iee.ihu.gr» επιστρέφει πίσω τον χρήστη στην διεύθυνση που δόθηκε στο πεδίο “Redirect_uri” «/Auth/Callback» όπου επιστρέφει επίσης και έναν κωδικό εξουσιοδότησης “code” καθώς και ένα “state”. Αυτά τα δύο θα χρειαστούν για να ανακτηθεί το Access Token. Όταν ανακατευθυνθεί πίσω στην σελίδα ο χρήστης, η μέθοδος “Callback” λαμβάνει αυτά τα στοιχεία και καλεί την μέθοδο “GetAccessToken” του Σχήματος 4.16 για να μετατρέψει τον κωδικό εξουσιοδότησης “code” στο Access Token.

```
private async Task<TokenResponse> GetAccessToken(string code)
{
    var TokenUrl = $"https://login.iee.ihu.gr/token";
    var httpClient = new HttpClient();

    var payload = new Dictionary<string, string>()
    {
        { "grant_type", "authorization_code" },
        { "client_id", CLIENT_ID },
        { "client_secret", CLIENT_SECRET },
        { "redirect_uri", @"https://localhost:7134/Auth/Callback" },
        { "code", code }
    };

    var stringContent = JsonConvert.SerializeObject(payload);

    var tokenResponse = await httpClient.PostAsync(TokenUrl, new StringContent(stringContent, Encoding.UTF8,
        "application/json"));
    var tokenContent = await tokenResponse.Content.ReadAsStringAsync();
    var token = JsonSerializer.Deserialize<TokenResponse>(tokenContent);

    return token;


}
```

Σχήμα 4.16: Μέθοδος GetAccessToken

Εφόσον το “Access Token” ληφθεί επιτυχώς θα χρησιμοποιηθεί για να ανακτηθούν οι πληροφορίες που χρειάζονται από τον χρήστη την μεθόδο “GetProfile” του Σχήματος 4.17 . Η συγκεκριμένη μέθοδος δημιουργεί ένα αίτημα «https://api.iee.ihu.gr/profile» όπου με αυτό το αίτημα το «login.iee.ihu.gr» παρέχει τις πληροφορίες για τον χρήστη. Όταν επιστραφούν από το αίτημα οι πληροφορίες αποσυμπιέζονται (Deserialize) και επιστρέφονται μέσω του πεδίου “profile”.

```
[HttpGet]
0 references
public async Task<IActionResult> Callback(string code, string state)
{

    var token = await GetAccessToken(code);
    var profileResponse = await GetProfile(token.access_token);
        ....
private async Task<ProfileResponse> GetProfile(string token)
{
    var TokenUrl = $"https://api.iee.ihu.gr/profile";
    var httpClient = new HttpClient();
    httpClient.DefaultRequestHeaders
    .Accept
    .Add(new MediaTypeWithQualityHeaderValue("application/json"));
    httpClient.DefaultRequestHeaders.Add("x-access-token", token);
    var response = await httpClient.GetAsync(TokenUrl);
    var responseContent = await response.Content.ReadAsStringAsync();

    var profile = JsonSerializer.Deserialize<ProfileResponse>(responseContent);
```

Σχήμα 4.17: Μέθοδος GetProfile

Μετά την επιτυχή επιστροφή των απαραίτητων πληροφορίων των μεθόδων “GetAccessToken” και “GetProfile” στην μέθοδο “Callback” του Σχήματος 4.18 αποθηκεύει τις πληροφορίες του χρήστη στην συνεδρία (session) όπως το πεδίο “uid” όπου είναι το αναγνωριστικό του χρήστη. Στην συνέχεια δημιουργείται μία λίστα “Claims” όπου αποθηκεύονται τα στοιχεία του χρήστη όπως το “cn” που είναι το όνομα το “uid” οπου είναι το αναγνωριστικό του χρήστη καθώς και το “eduPersonAffiliation” που είναι η ιδιότητα του. Αφού γίνει η απαραίτητη εγγραφή των στοιχείων που ορίστηκαν στα “Claims” δημιουργούνται δύο ρόλοι χρηστών. Ένας ρόλος είναι ο “Teacher” εάν το πεδίο “eduPersonAffiliation” είναι “staff” και ο άλλος ρόλος είναι ο “Student” εάν το πεδίο «eduPersonAffiliation» είναι “student” έτσι ώστε να διαχωριστεί η πρόσβαση του καθηγητή στην εφαρμογή με του φοιτητή.

```
var token = await GetAccessToken(code);
var profileResponse = await GetProfile(token.access_token);
//Αποθήκευση των πληροφοριών του χρήστη στην συνεδρία
ViewData.Add("Name", profileResponse.cn);
ViewData.Add("UserId", profileResponse.uid);
HttpContext.Session.SetString("UserId", profileResponse.uid);


var claims = new List<Claim>
{
    new Claim(ClaimTypes.Name, profileResponse.cn),
    new Claim(ClaimTypes.NameIdentifier, profileResponse.id),
    new Claim("UserId", profileResponse.uid),
    new Claim("Edu", profileResponse.eduPersonAffiliation)
};

if (profileResponse.eduPersonAffiliation == "staff")
{
    claims.Add(new Claim(ClaimTypes.Role, UserRoles.Teacher));
}
else if(profileResponse.eduPersonAffiliation == "student")
{
    claims.Add(new Claim(ClaimTypes.Role, UserRoles.Student));
}
```

Σχήμα 4.18: Μέθοδος Callback

2. QuestionCategoryController (Δημιουργία και Διαγραφή Κατηγοριών)

Για την δημιουργία των κατηγοριών όπως φαίνεται στο Σχήμα 4.19 δημιουργήθηκε μία «HttpPost Create» μέθοδο όπου ελέγχει αν τα δεδομένα που υποβλήθηκαν είναι σωστά (ModelState.isValid). Έπειτα δημιουργήθηκε ένα αντικείμενο "categoryModel" τύπου QuestionCategory από το ενδιάμεσο επίπεδο Common. Στην συνέχεια καλείται η υπηρεσία "_categoryService.Create()". Στο «ICategoryService Create» δημιουργήθηκε και εκεί ένα αντικείμενο "model" αυτήν την φορά τύπου "QuestionCategory Storage" για να καταχωρηθούν και να αποθηκευτούν τα δεδομένα στην βάση με την μέθοδο "Add(model)" και την "SaveChangesAsync()" όπως παρουσιάζεται στο Σχήμα 4.20. Εφόσον ολοκληρωθεί η διαδικασία με επιτυχία καλείται η "RedirectToAction" όπου ανακατευθύνει τον χρήστη στην σελίδα "Index" του "View" με ένα μήνυμα επιτυχίας .

```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create(Models.QuestionCategory questionCategory)
{
    if (ModelState.IsValid)
    {
        var categoryModel = new ExamThesis.Common.QuestionCategory()
        {
            QuestionCategoryName = questionCategory.QuestionCategoryName,
        };

        await _categoryService.Create(categoryModel);

        TempData["SuccessMessage"] = "Question category created successfully.";
        return RedirectToAction("Index");
    }

    return View(questionCategory);
}
```

Σχήμα 4.19: Μέθοδος HttpPost Create-QuestionCategoryController

```
2 references
public async Task Create(ExamThesis.Common.QuestionCategory category)
{
    var model = new Storage.Model.QuestionCategory()
    {
        QuestionCategoryName = category.QuestionCategoryName,
    };
    _db.QuestionCategories.Add(model);
    await _db.SaveChangesAsync();

}
```

Σχήμα 4.20: Μέθοδος Create-ICategoryService

Για την διαγραφή των κατηγοριών καλείται η μέθοδος "DeletePost" όπως φαίνεται στο Σχήμα 4.21 του "QuestionCategoryController" όπου με βάσει το "id" της κατηγορίας καλείται το "_categoryService.DeleteById()" και ψάχνει με την μέθοδο "FindAsync()" του Σχήματος 4.22 την κατηγορία με βάσει το "id", διαγράφεται με την μέθοδο "Remove" και αποθηκεύονται οι αλλαγές με την "SaveChangesAsync()". Όταν η διαδικασία ολοκληρωθεί με επιτυχία καλείται η μέθοδος "RedirectToAction" και ανακατευθύνει τον χρήστη στην σελίδα "Index" του View".

```
[HttpDelete]
0 references
public async Task<IActionResult> DeletePost(int id)
{
    await _categoryService.DeleteById(id);

    TempData["SuccessMessage"] = "Category deleted successfully.";

    return RedirectToAction("Index");
}
```

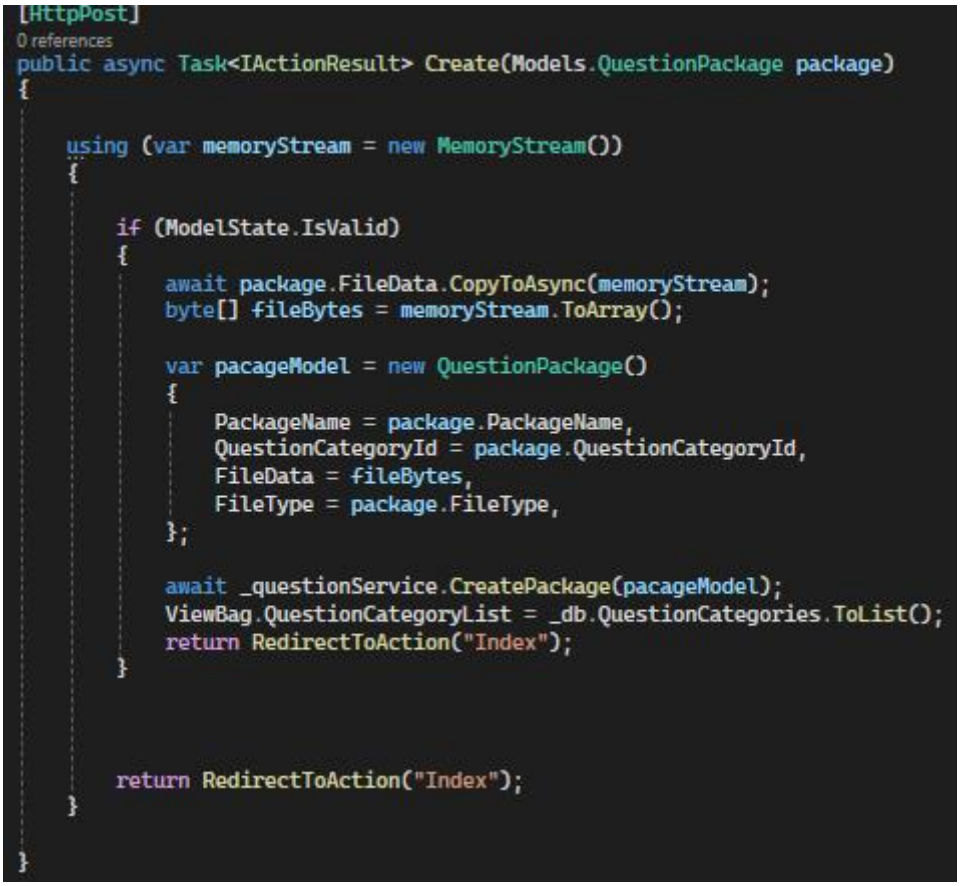
Σχήμα 4.21: Μέθοδος Http DeletePost-QuestionCategoryController

```
2 references
public async Task DeleteById(int id)
{
    var obj = await _db.QuestionCategories.FindAsync(id);
    _db.QuestionCategories.Remove(obj);
    await _db.SaveChangesAsync();
}
```

Σχήμα 4.22: Μέθοδος DeleteById-ICategoryService

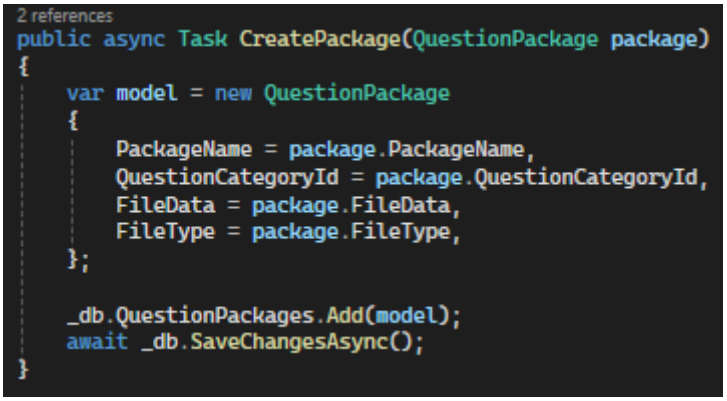
3. QuestionPackageController (Δημιουργία και Διαγραφή Πακέτων Ερωτήσεων)

Για την δημιουργία ενός πακέτου ερωτήσεων όταν καλείται η μέθοδος “HttpPost Create” του Σχήματος 4.23 δημιουργήθηκε ένα “MemoryStream()” για την προσωρινή αποθήκευση του αρχείου για να αποφευχθεί η προσωρινή αποθήκευση σε κάποιο φυσικό δίσκο και για την εύκολη μετατροπή του δεδομένων του αρχείου σε “byte” μέσω του “memoryStream.ToArray()”. Έπειτα ελέγχεται η εγκυρότητα του μοντέλου μέσω της “ModelState.IsValid” όπου ελέγχει αν το μοντέλο “QuestionPackage” είναι έγκυρο. Εάν τα δεδομένα της φόρμας δεν είναι έγκυρα, τότε η μέθοδος θα επιστρέψει ξανά στην φόρμα με μήνυμα σφάλματος. Εφόσον τα δεδομένα του μοντέλου είναι έγκυρα αντιγράφεται το αρχείο στο “MemoryStream” και μετατρέπεται σε “byte array” για να αποθηκευτεί αργότερα στην βάση. Μετά δημιουργείται ένα αντικείμενο τύπου “QuestionPackage” όπου περιέχει το όνομα του πακέτου, την κατηγορία ερωτήσεων καθώς επίσης τα δεδομένα του αρχείου και τον τύπο του. Τέλος καλείται η μέθοδος “_questionService.CreatePackage()” του Σχήματος 4.24 από το “IQuestionService” όπου και εκεί δημιουργείται ένα αντικείμενο “Model” τύπου “QuestionPackage Storage”, προστίθεται με την μέθοδο “Add()” το μοντέλο στην βάση και έπειτα αποθηκεύονται οι αλλαγές με την μέθοδο “SaveChangesAsync()”.



```
[HttpPost]
0 references
public async Task<IActionResult> Create(Models.QuestionPackage package)
{

    using (var memoryStream = new MemoryStream())
    {

        if (ModelState.IsValid)
        {
            await package.FileData.CopyToAsync(memoryStream);
            byte[] fileBytes = memoryStream.ToArray();

            var pacageModel = new QuestionPackage()
            {
                PackageName = package.PackageName,
                QuestionCategoryId = package.QuestionCategoryId,
                FileData = fileBytes,
                FileType = package.FileType,
            };

            await _questionService.CreatePackage(pacageModel);
            ViewBag.QuestionCategoryList = _db.QuestionCategories.ToList();
            return RedirectToAction("Index");
        }



        return RedirectToAction("Index");
    }

}
```

Σχήμα 4.23: Μέθοδος HttpPost Create-QuestionPackageController



```
2 references
public async Task CreatePackage(QuestionPackage package)
{
    var model = new QuestionPackage
    {
        PackageName = package.PackageName,
        QuestionCategoryId = package.QuestionCategoryId,
        FileData = package.FileData,
        FileType = package.FileType,
    };

    _db.QuestionPackages.Add(model);
    await _db.SaveChangesAsync();
}
```

Σχήμα 4.24: Μέθοδος CreatePackage-IQuestionService

Για την διαγραφή ενός πακέτου ερωτήσεων καλείται η μέθοδος του Σχήματος 4.25 “HttpPost Delete()” όπου με βάσει το “id” του πακέτου καλείται η υπηρεσία “_questionService.DeletePackage()” από το “Services” του Σχήματος 4.26 όπου με βάσει το “PackageId” πραγματοποιούνται αλυσιδωτές διαγραφές των ερωτήσεων, των απαντήσεων και τέλος το πακέτο όπου ανήκουν όλα αυτά με την μέθοδο “Remove” και αποθηκεύονται οι αλλαγές με την μέθοδο “SaveChangesAsync()”. Όταν ολοκληρωθεί η διαγραφή καλείται η μέθοδος “RedirectToAction()” όπου ανακατευθύνει τον χρήστη στην ανανεωμένη λίστα των πακέτων.

```
[HttpPost]
0 references
public async Task<IActionResult> Delete(int id)
{
    await _questionService.DeletePackage(id);
    return RedirectToAction("Index");
}
```

Σχήμα 4.25: Μέθοδος HttpPost Delete-QuestionPackageController

```
2 references
public async Task DeletePackage(int packageId)
{
    // Delete QuestionsInPackage
    var questionsInPackage = await _db.QuestionsInPackages.Where(qip => qip.PackageId == packageId).ToListAsync();
    _db.QuestionsInPackages.RemoveRange(questionsInPackage);

    // Delete Questions
    var questions = await _db.Questions.Where(q => q.PackageId == packageId).ToListAsync();
    _db.Questions.RemoveRange(questions);

    // Delete Answers of Question
    var answers = await _db.Answers.Where(a => a.Question.PackageId == packageId).ToListAsync();
    _db.Answers.RemoveRange(answers);

    // Delete QuestionPackage
    var package = await _db.QuestionPackages.FindAsync(packageId);
    _db.QuestionPackages.Remove(package);

    await _db.SaveChangesAsync();
}
```

Σχήμα 4.26: Μέθοδος DeletePackage-IQuestionService

4. QuestionCreateController (Δημιουργία και Διαγραφή Ερωτήσεων)

Σε αυτόν τον Controller δημιουργούνται τα αιτήματα HttpGet,HttpPost και HttpDelete για την δημιουργία και διαγραφή των ερωτήσεων. Κατά την δημιουργία μίας ερώτησης για αρχή καλείται η μέθοδος “HttpGet Create()”. Όπως παρουσιάζεται στο Σχήμα 4.27 δημιουργείται ένα “ViewModel” με αντικείμενο τύπου “CreateQuestion” που χρησιμοποιείται για να γεμίζει την φόρμα στην προβολή. Στην συνέχεια γίνεται ανάκτηση και προβολή όλων των πακέτων ερωτήσεων πού έχουν ήδη δημιουργηθεί καθώς και των κατηγοριών μέσω της “ViewBag”. Τέλος επιστρέφει όλα τα στοιχεία που χρειάζονται μέσω του “View(viewModel)”.

```
0 references
public IActionResult Create()
{
    var viewModel = new Models.CreateQuestion();
    var questionPackages = _db.QuestionPackages.ToList();
    ViewBag.QuestionCategories = new SelectList(_db.QuestionCategories, "QuestionCategoryId", "QuestionCategoryName");
    ViewBag.QuestionPackages = new SelectList(questionPackages, "PackageId", "PackageName");
    return View(viewModel);
}
```

Σχήμα 4.27: Μέθοδος HttpGet Create-QuestionCreateController

Στην συνέχεια για την δημιουργία της ερώτησης καλείται η μέθοδος “HttpPost Create()” του Σχήματος 4.28. Για αρχή γίνεται ένας έλεγχος του μοντέλου εάν τα στοιχεία είναι έγκυρα με την “ModelState.isValid”. Εάν τα δεδομένα του μοντέλου δεν είναι έγκυρα τότε θα ανακατευθύνει πάλι στην φόρμα συμπλήρωσης. Εφόσον τα δεδομένα είναι έγκυρα τότε δημιουργείται ένα αντικείμενο “CreateQuestion”. Τα πεδία του “createModel” αντιγράφονται από το μοντέλο “question” και στην συνέχεια η λίστα “Answers” φιλτράρεται για να περιέχει μόνο τις απαντήσεις που έχουν κείμενο και έπειτα μετατρέπεται σε ένα αντικείμενο τύπου “CreateAnswer”. Μετά και την δημιουργία του μοντέλου καλείται η υπηρεσία “_questionService.Create()” για να γίνει η αποθήκευση της φόρμας του Σχήματος 4.29. Στο “_questionService.Create()” δημιουργήθηκε ένα αντικείμενο “Model” τύπου Storage.Question αυτήν την φορά για την αντιγραφή των πεδίων. Έπειτα αφού γίνει η προσθήκη της ερώτησης με την μέθοδο “Add(model)” δημιουργήθηκε επίσης ένα αντικείμενο τύπου Storage.QuestionsInPackage, όπου και αντιγράφονται τα πεδία του “questionInPackage” από το μοντέλο “model” και προστίθενται στον πίνακα “QuestionInPackages” με την μέθοδο “Add()”. Αφού τελειώσουν όλες οι διαδικασίες αν το μοντέλο είναι έγκυρο ανακατευθύνει τον χρήστη στην σελίδα “Index” με την μέθοδο “RedirectToAction()”.

```
[HttpPost]
0 references
public async Task<IActionResult> Create(CreateQuestion question)
{
    if (ModelState.IsValid)
    {
        var createModel = new ExamThesis.Common.CreateQuestion()
        {
            QuestionText = question.QuestionText,
            Answers = question.Answers
            .Where(answer => !string.IsNullOrEmpty(answer.Text))
            .Select(answer => new ExamThesis.Common.CreateAnswer
            {
                Text = answer.Text,
                IsCorrect = answer.IsCorrect

            }).ToList(),
            QuestionPoints = question.QuestionPoints,
            NegativePoints = question.NegativePoints,
            QuestionCategoryId = question.QuestionCategoryId,
            PackageId = question.PackageId
        };
        await _questionService.Create(createModel);
        TempData["SuccessMessage"] = "Questions and Answers created successfully.";
        return RedirectToAction("Index");
    }
    TempData["FailMessage"] = "Questions and Answers created failed.";

    return View(question);
}
```

Σχήμα 4.28: Μέθοδος HttpPost Create-QuestionCreateController

```
2 references
public async Task Create(CreateQuestion question)
{
    var model = new ExamThesis.Storage.Model.Question
    {
        QuestionText = question.QuestionText,
        Answers = question.Answers
            .Where(answer => !string.IsNullOrEmpty(answer.Text))
            .Select(answer => new ExamThesis.Storage.Model.Answer
            {
                Text = answer.Text,
                IsCorrect = answer.IsCorrect

            }).ToList(),
        QuestionPoints = question.QuestionPoints,
        NegativePoints = question.NegativePoints,
        QuestionCategoryId = question.QuestionCategoryId,
        PackageId = question.PackageId
    };

    _db.Questions.Add(model);
    await _db.SaveChangesAsync();

    // Προσθήκη ερώτησης στο QuestionInPackage
    var questionInPackage = new QuestionsInPackage
    {
        PackageId = model.PackageId,
        QuestionId = model.QuestionId
    };

    _db.QuestionsInPackages.Add(questionInPackage);
    await _db.SaveChangesAsync();
}
```

Σχήμα 4.29: Μέθοδος Create-IQuestionService

Για την διαγραφή μίας ερώτησης δημιουργήθηκε μία μέθοδος “HttpGet Delete” του Σχήματος 4.30 όπου καταχωρεί σε μία μεταβλητή τύπου “questionFromDb” την ερώτηση η οποία έχει επιλεχθεί από τον χρήστη. Στην συνέχεια γίνεται έλεγχος αν το πεδίο είναι “null” επιστρέφει με την μέθοδο “NotFound()” ότι δεν υπάρχει αυτή η ερώτηση ενώ αν δεν είναι “null” επιστρέφει στο “View” την ερώτηση “questionFromDb”.

```
[HttpGet]
0 references
public IActionResult Delete(int id)
{
    var questionFromDb = _db.Questions.Find(id);

    if (questionFromDb == null)
    {
        return NotFound();
    }

    return View(questionFromDb);
}
```

Σχήμα 4.30: Μέθοδος HttpGet Delete-QuestionCreateController

Για την διαγραφή της επιλεγμένης ερώτησης δημιουργήθηκε η "HttpDelete DeleteConfirmed" του Σχήματος 4.31 όπου με βάσει το "id" της ερώτησης καλείται το Service "_questionService.DeleteById();". Όταν εκτελείται το service του Σχήματος 4.32 τοποθετούνται σε δύο πεδία η επιλεγμένη ερώτηση καθώς και οι απαντήσεις όπου ανήκουν σε αυτή. Στην συνέχεια καταχωρείται σε ένα τρίτο πεδίο την ερώτηση που βρίσκεται μέσα στον πίνακα "QuestionInPackage" , διαγράφεται από εκεί και έπειτα διαγράφεται η ερώτηση και οι απαντήσεις της από τον πίνακα "Questions" και "Answer". Τέλος εφόσον η διαγραφή ήταν επιτυχής ανακατευθύνει τον χρήστη στην ανανεωμένη σελίδα "Index" με την μέθοδο "RedirectToAction()".

```
[HttpDelete]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> DeleteConfirmed(int id)
{
    await _questionService.DeleteById(id);
    return RedirectToAction("Index");
}
```

Σχήμα 4.31: Μέθοδος HttpDelete DeleteConfirmed-QuestionCreateController

```
2 references
public async Task DeleteById(int id)
{
    var questionToDelete = _db.Questions.Find(id);
    var answersToDelete = _db.Answers.Where(a => a.QuestionId == id);

    var questionsInPackageToDelete = _db.QuestionsInPackages.Where(qip => qip.QuestionId == id);
    _db.QuestionsInPackages.RemoveRange(questionsInPackageToDelete);

    _db.Answers.RemoveRange(answersToDelete);

    _db.Questions.Remove(questionToDelete);
   await _db.SaveChangesAsync();

}
```

Σχήμα 4.32: Μέθοδος DeleteById-IQuestionService

4. ExamController (Δημιουργία,Διαγραφή και Διαχείριση της Εξέτασης)

Στον «ExamController» δημιουργήθηκαν όλα τα αιτήματα HttpGet, HttpPost, HttpDelete για την δημιουργία την διαγραφή και την διαχείριση της εξέτασης. Για την δημιουργία της εξέτασης στην αρχή καλείται η μέθοδος «HttpGet Create» του Σχήματος 4.33 όπου παρέχει τα δεδομένα που χρειάζονται για την εμφάνιση της φόρμας καθώς και για την δημιουργία της εξέτασης. Στην αρχή γίνεται ανάκτηση όλων των κατηγοριών, όπου κάθε κατηγορία μετατρέπεται σε ένα αντικείμενο τύπου "QuestionCategory" που περιέχει το "QuestionCategoryId" και "QuestionCategoryName" και προστίθενται σε μία λίστα τύπου "questionCategories". Στην συνέχεια δημιουργήθηκε ένα νέο αντικείμενο τύπου "SelectedCategories" του μοντέλου "CreateExam" όπου εκεί ορίζεται η λίστα των κατηγοριών. Αυτό το μοντέλο επιστρέφει με την "return View(model)" για να εμφανίσει στον χρήστη κατηγορίες ερωτήσεων ως επιλογές και τα πεδία.

```
0 references
public IActionResult Create()
{
    var questionCategories = _db.QuestionCategories.Select(qc => new ExamThesis.Models.QuestionCategory()
    {
        QuestionCategoryId=qc.QuestionCategoryId,
        QuestionCategoryName=qc.QuestionCategoryName,
    }).ToList();

    var model = new CreateExam()
    {
        SelectedCategories = questionCategories

    };

    return View(model);
}
```

Σχήμα 4.33: Μέθοδος HttpGet Create-ExamController

Αφού επιστραφεί σωστά το "Model" για την δημιουργία της εξέτασης, δημιουργήθηκε η μέθοδος "HttpPost Create" του Σχήματος 4.34. Αρχικά ελέγχει εάν τα δεδομένα του μοντέλου είναι έγκυρα με το "ModelState.isValid". Αν τα δεδομένα είναι έγκυρα προχωράει στην δημιουργία ενός αντικειμένου "examModel" τύπου "CreateExam" του ενδιάμεσου επιπέδου "Common" όπου και εκεί αντιγράφονται τα πεδία. Έπειτα καλείται το "_examService.CreateExam(examModel)" Service του Σχήματος 4.35.όπου προσθέτει στην βάση και αποθηκεύει τα νέα δεδομένα αλλά και τις επιλεγμένες κατηγορίες που επέλεξε ο χρήστης. Μετά την προσθήκη και αποθήκευση των αλλαγών καλείται η "RedirectToAction("Index")" όπου ανακατευθύνει τον χρήστη στην σελίδα "Index" με την ανανεωμένη λίστα των εξετάσεων. Σε περίπτωση που τα δεδομένα του μοντέλου δεν είναι έγκυρα, τότε εμφανίζει μήνυμα σφάλματος και επαναφέρει τον χρήστη ξανά στην φόρμα συμπλήρωσης για να ξανά συμπληρώσει τα πεδία με το "return View(model)".

```
if (!ModelState.IsValid)
{
    foreach (var error in ModelState.Values.SelectMany(v => v.Errors))
    {
        Console.WriteLine($"Model error: {error.ErrorMessage}");
    }
    return View(model);
}
ViewBag.ShowGrade = model.ShowGrade;
if (ModelState.IsValid)
{
    var examModel= new ExamThesis.Common.CreateExam(){
        ExamName = model.ExamName,
        StartTime = model.StartTime,
        EndTime = model.EndTime,
        TotalPoints = model.TotalPoints,
        PassGrade = model.PassGrade,
        ShowGrade = model.ShowGrade,
        SelectedCategories = model.SelectedCategories.Where(sc => sc.IsChecked==true).Select(sc => new Common.QuestionCategory() {
        QuestionCategoryId=sc.QuestionCategoryId}).ToList()

    };
 await _examService.CreateExam(examModel);

    return RedirectToAction("Index");
}

return View(model);
```

Σχήμα 4.34: Μέθοδος HttpPost Create-ExamController

```
public async Task CreateExam(CreateExam exam)
{

    var model = new Exam
    {
        ExamName = exam.ExamName,
        StartTime = exam.StartTime,
        EndTime = exam.EndTime,
        TotalPoints = exam.TotalPoints,
        PassGrade = exam.PassGrade,
        ShowGrade = exam.ShowGrade,

    };

    _db.Exams.Add(model);
    await _db.SaveChangesAsync();
    await _categoryService.AddCategoriesForExam(exam, model.ExamId);
}
```

Σχήμα 4.35: Μέθοδος CreateExam-IExamService

Για την διαγραφή μίας εξέτασης δημιουργήθηκε μία μέθοδο “HttpGet Delete” του Σχήματος 4.36 όπου καταχωρεί σε μία μεταβλητή τύπου “examFromDb” την εξέταση η οποία έχει επιλεχθεί από τον χρήστη. Στην συνέχεια γίνεται ο έλεγχος, εαν το πεδίο είναι “null” επιστρέφει με την μέθοδο “NotFound()” ότι δεν υπάρχει αυτή η εξέταση. Ενώ αν δεν είναι “null” επιστρέφει στο “View” την εξέταση “examFromDb”

```
[HttpGet]
0 references
public IActionResult Delete(int id)
{
    var examFromDb = _db.Exams.Find(id);

    if (examFromDb == null)
    {
        return NotFound();
    }

    return View(examFromDb);
}
```

Σχήμα 4.36: Μέθοδος HttpGet Delete-ExamController

Για την διαγραφή της επιλεγμένης εξέτασης δημιουργήθηκε η “HttpDelete DeleteConfirmed” του Σχήματος 4.37 όπου με βάσει το “id” της εξέτασης καλείται το Service “_examService.DeleteByExamId();”. Όταν εκτελείται το service του Σχήματος 4.38 καταχωρούνται σε δύο πεδία η επιλεγμένη εξέταση καθώς και τα αποτελέσματα όπου ανήκουν σε αυτή. Στην συνέχεια καταχωρείται επίσης σε ένα τρίτο πεδίο με τις επιλεγμένες κατηγορίες της εξέτασης που βρίσκονται μέσα στον πίνακα “ ExamCategories” , διαγράφονται από εκεί και έπειτα διαγράφεται η εξέταση με τις

επιλεγμένες κατηγορίες της από τον πίνακα “ ExamResults” και “Exams”. Τέλος εφόσον η διαγραφή ήταν επιτυχής ανακατευθύνει τον χρήστη στην ανανεωμένη σελίδα “Index” με την μέθοδο “RedirectToAction()”.

```
[HttpDelete]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> DeleteConfirmed(int id)
{
    await _examService.DeleteByExamId(id);
    TempData["SuccessMessage"] = "Exam deleted successfully.";
    return RedirectToAction("Index");
}
```

Σχήμα 4.37: Μέθοδος HttpDelete DeleteConfirmed-ExamController

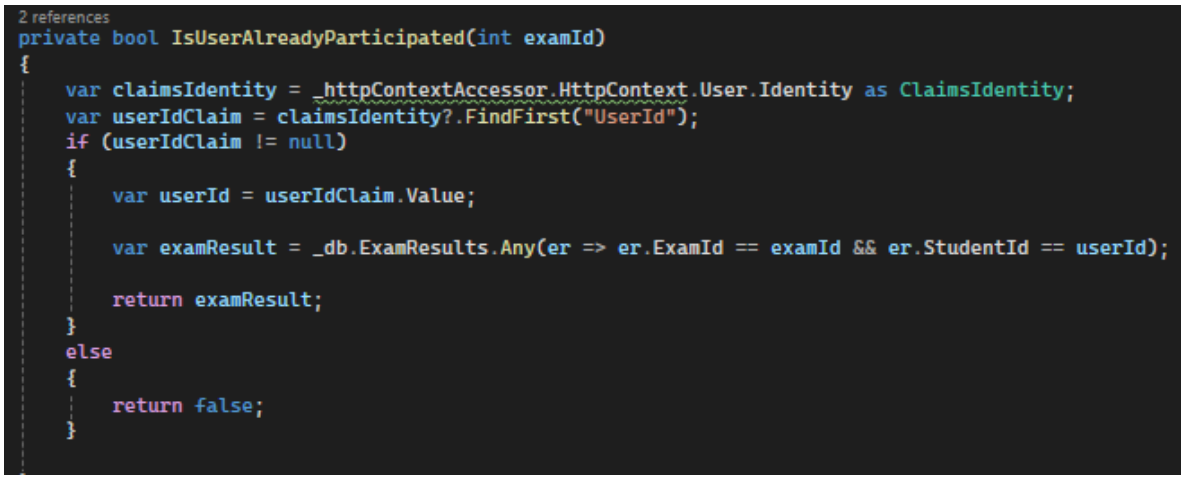
```
2 references
public async Task DeleteByExamId(int examId)
{
    var examToDelete = await _db.Exams.FindAsync(examId);
    if (examToDelete != null)
    {
        var examResultsToDelete = await _db.ExamResults.Where(er => er.ExamId == examId).ToListAsync();
        _db.ExamResults.RemoveRange(examResultsToDelete);

        var examCategoriesToDelete = await _db.ExamCategories.Where(ec => ec.ExamId == examId).ToListAsync();
        _db.ExamCategories.RemoveRange(examCategoriesToDelete);

        _db.Exams.Remove(examToDelete);

        await _db.SaveChangesAsync();

    }
}
```

Σχήμα 4.38: Μεθοδος DeleteByExamId-IExamService

Για την εκκίνηση της εξέτασης όταν ο φοιτητής θα προσπαθήσει εισέλθει στην εξέταση δημιουργούνται πρώτα δύο έλεγχοι. Ο πρώτος έλεγχος γίνεται όπως παρουσιάζεται και στο Σχήμα 4.39 στην μέθοδο “CanStartExam()” όπου με βάσει το “id” της εξέτασης ελέγχεται αν η ώρα που ο φοιτητής προσπαθεί να εισέλθει στην εξέταση είναι μέσα στα χρονικά πλαίσια που έχει ορίσει ο καθηγητής κατά την δημιουργία της. Ο δεύτερος έλεγχος γίνεται εάν ο φοιτητής έχει ολοκληρώσει την συγκεκριμένη εξέταση όπως παρουσιάζεται και στο Σχήμα 4.40 στην μέθοδο “IsUserAlreadyParticipated()”. Σε αυτήν την μέθοδο ελέγχεται στον πίνακα “ExamResults” αν ο χρήστης έχει κάνει υποβολή των απαντήσεων του και επιστρέφει το αποτέλεσμα σε “bool”.

```
0 references
private bool CanStartExam(int examId)
{
    var exam = _db.Exams.First(er => er.ExamId == examId);
    return DateTime.Now >= exam.StartTime || DateTime.Now <exam.EndTime;
}
```

Σχήμα 4.39: Μέθοδος Ελέγχου CanStartExam-ExamController



```
2 references
private bool IsUserAlreadyParticipated(int examId)
{
    var claimsIdentity = _httpContextAccessor.HttpContext.User.Identity as ClaimsIdentity;
    var userIdClaim = claimsIdentity?.FindFirst("UserId");
    if (userIdClaim != null)
    {
        var userId = userIdClaim.Value;

        var examResult = _db.ExamResults.Any(er => er.ExamId == examId && er.StudentId == userId);

        return examResult;
    }
    else
    {
        return false;
    }
```

Σχήμα 4.40: Μέθοδος Ελέγχου IsUserAlreadyParticipated-ExamController

Εφόσον έχουν δημιουργηθεί οι απαραίτητοι έλεγχοι, στην «HttpPost Exam» μέθοδο καλούνται οι έλεγχοι και με την βοήθεια της "if/else if" όπως φαίνεται στο Σχήμα 4.41 όπου ελέγχεται εάν είναι "true/false" οι τιμές των ελέγχων. Όταν περάσει από του δύο ελέγχους μέσα στην μεταβλητή "exam" αποθηκεύεται το αναγνωριστικό της εξέτασης που έχει επιλέξει ο χρήστης. Έπειτα ανακτάται το αναγνωριστικό του φοιτητή και καλείται το "_examService.GetExamQuestionsByExamId(id, userIdClaim);" Service με πεδία το "id" της εξέτασης και το αναγνωριστικό του φοιτητή. Σε αυτό το Service του Σχήματος 4.42 δημιουργήθηκε μία "Get" μέθοδο όπου ανακτά και επιστρέφει μέσω του μοντέλου "ExamQuestionViewModel" τις ερωτήσεις και απαντήσεις στον φοιτητή. Στην αρχή με βάσει το αναγνωριστικό του φοιτητή και της εξέτασης δημιουργήθηκε ένα κλειδί τύπου "cacheKey" για την προσωρινή αποθήκευση των δεδομένων στην μνήμη. Αφού προστέθηκε η βιβλιοθήκη "Microsoft.Extensions.Caching.Memory;" ελέγχεται με την "_memoryCache" αν οι ερωτήσεις για τον συγκεκριμένο διαγώνισμα και φοιτητή έχουν ήδη αποθηκευτεί στην μνήμη. Εάν δεν υπάρχουν δεδομένα με αυτά τα αναγνωριστικά τότε συνεχίζει στην ανάκτηση των ερωτήσεων και απαντήσεων για τον κάθε φοιτητή ξεχωριστά. Στην αρχή γίνεται η ανάκτηση της εξέτασης και προστίθεται το αναγνωριστικό της σε μία μεταβλητή τύπου "exam". Αφού γίνει ο απαραίτητος έλεγχος εάν είναι "null" γίνεται η ανάκτηση των κατηγοριών της εξέτασης που έχει επιλέξει ο χρήστης κατά την δημιουργία της μέσω του πίνακα "ExamCategories" και τα προστίθενται σε μία μεταβλητή τύπου "examCategories". Έπειτα για κάθε κατηγορία γίνονται ανάκτηση τα πακέτα ερωτήσεων όπου επιλέγεται τυχαία ένα πακέτο ερωτήσεων για κάθε κατηγορία με την βοήθεια της "Random()". Δημιουργήθηκε επίσης, ένα μοντέλο τύπου "ExamQuestionViewModel" του Σχήματος 4.43. Αφού έχει ολοκληρωθεί η ανάκτηση των ερωτήσεων και απαντήσεων. Στην συνέχεια, αποθηκεύονται στην μνήμη "Cache" όλα τα δεδομένα σε μορφή "JSON" έτσι ώστε σε περίπτωση που για κάποιον αδιευκρίνιστο λόγο ο φοιτητής χάσει την σύνδεση του με την εξέταση όταν θα προσπαθήσει να ξανά εισέλθει σε αυτήν η ανάκτηση των δεδομένων θα γίνει από την μνήμη Cache έτσι ώστε να ξανά έχει τις ίδιες ερωτήσεις και απαντήσεις. Τα αποθηκευμένα δεδομένα στην μνήμη "Cache" παραμένουν εκεί για 120 λεπτά μέσω του AbsuluteExpirationRelativeToNow. Τέλος επιστρέφει στον Controller το "ExamQuestionViewModel" και καταχωρείται στην μεταβλητή "model" και επιστρέφεται στο "View" το "model".

```
0 references
public async Task<IActionResult> Exam(int id)
{
    if (IsUserAlreadyParticipated(id))
    {
        TempData["ErrorMessage"] = "Έχετε ήδη ολοκληρώσει αυτήν την εξέταση.";
        return RedirectToAction("Index");
    }
    else if (!CanStartExam(id))
    {
        TempData["ErrorMessage"] = "Δεν έχει ξεκινήσει η εξέταση!";
        return RedirectToAction("Index");
    }
    else
    {
        var exam = await _db.Exams.FindAsync(id);
        if (exam != null)
        {
            ViewBag.EndTime = exam.EndTime.ToString("HH:mm:ss");
            ViewBag.StartTime = exam.StartTime.ToString("HH:mm:ss");
        }
        ViewBag.ExamId = id;
        var claimsIdentity = _httpContextAccessor.HttpContext.User.Identity as ClaimsIdentity;
        var userIdClaim = claimsIdentity?.FindFirst("UserId").Value;
        var model = await _examService.GetExamQuestionsByExamId(id, userIdClaim);
        return base.View(model);
    }
}
```

Σχήμα 4.41: Μέθοδος HttpPost Exam-ExamController

```
ExamQuestionViewModel examQuestionViewModel = null;
var cacheKey = $"{studentId}_examId_questions";

if (_memoryCache.TryGetValue<string>(cacheKey, out var examJson))
{
    return JsonConvert.DeserializeObject<List<ExamQuestionViewModel>>(examJson);
}
else
{
    var exam = await _db.Exams.FindAsync(examId);

    if (exam == null)
    {
        return Enumerable.Empty<ExamQuestionViewModel>();
    }
```

Σχήμα 4.42: Μέθοδος GetExamQuestionsByExam-IExamService 1/2

```
examQuestionViewModel = new ExamQuestionViewModel
{
    StartTime = exam.StartTime,
    EndTime = exam.EndTime,
    TotalPoints = exam.TotalPoints,
    ExamId = exam.ExamId,
    ExamName = exam.ExamName,
    PassGrade = (double)exam.PassGrade,
    QuestionId = question.QuestionId,
    QuestionCategoryId = question.QuestionCategoryId,
    NegativePoints = question.NegativePoints,
    QuestionText = question.QuestionText,
    QuestionPoints = question.QuestionPoints,
    PackageId = question.PackageId,
    Answers = question.Answers,
    QuestionsInPackages = question.QuestionsInPackages,
    FileData = selectedPackage.FileData,
    PackageName = selectedPackage.PackageName,
    FileType = selectedPackage.FileType,



examQuestionViewModels.Add(examQuestionViewModel);
```

Σχήμα 4.43: Μέθοδος GetExamQuestionsByExam-IExamService 2/2

Αφού έχουν ανακτηθεί όλα τα δεδομένα που χρειάζονται για την διαδικασία της εξέτασης όταν απαντηθούν οι ερωτήσεις από τον χρήστη καλείται η μέθοδος “Submit()” του Σχήματος 4.44. Στην αρχή ανακτάται το αναγνωριστικό του φοιτητή από το authentication του “Claim”. Έπειτα στην καινούρια μεταβλητή “earnedPoints” εισάχθηκαν τα αποτελέσματα από το “_examService.SubmitExam()” του Σχήματος 4.45. Το συγκεκριμένο “Service” είναι υπεύθυνο για την αξιολόγηση των απαντήσεων, τον συνολικό βαθμό καθώς και για την κατοχύρωση τους στην βάση δεδομένων. Πρώτα γίνεται η ανάκτηση όλων των ερωτήσεων της εξέτασης που έχει υποβάλλει ο φοιτητής στην μεταβλητή “examQuestions”. Έπειτα ανακτά την εξέταση που έχει υποβάλει ο φοιτητής και αποθηκεύεται το αναγνωριστικό της στην μεταβλητή “exam”. Δημιουργείται μία νέα μεταβλητή “earnedPoints” και γίνεται η αρχικοποίηση της με την τιμή “0.0” καθώς η συγκεκριμένη μεταβλητή θα επιστρέφει τον συνολικό βαθμό. Στην συνέχεια ελέγχεται εάν οι απαντήσεις που έχει δώσει ο φοιτητής περιλαμβάνουν την σωστή. Εάν η απάντηση που έδωσε είναι σωστή τότε προσθέτει στον συνολικό βαθμό τις μονάδες της ερώτησης, διαφορετικά εάν η απάντηση είναι λάθος αφαιρεί από τον συνολικό βαθμό την αρνητική βαθμολογία της ερώτησης. Με την βοήθεια του “Contain” δίνεται η δυνατότητα να υπολογίζονται μόνο οι ερωτήσεις που έχει συμπληρώσει ο φοιτητής έτσι ώστε στις απαντήσεις που έχουν μείνει κενές να μην αφαιρεθεί η αρνητική βαθμολογία. Αφού έχει ολοκληρωθεί η διαδικασία του συνολικού βαθμού δημιουργήθηκε ένα αντικείμενο “examResult” το οποίο αποθηκεύει το “studentId”, το “examId” και το “earnedPoints” και το προσθέτει στην βάση δεδομένων στον πίνακα “ExamResult”. Εφόσον επιστρέψει τον συνολικό βαθμό στον Controller υπολογίζει εάν ο συνολικός βαθμός του φοιτητή είναι μεγαλύτερος ή ίσος με την βάση επιτυχίας που έχει ορίσει ο καθηγητής και ανάλογα με το αποτέλεσμα επιστρέφει στο “View” το αντίστοιχο μήνυμα επιτυχίας ή αποτυχίας.

```
public async Task<IActionResult> Submit(int id, List<int> selectedAnswers,List<int> selectedQuestions,string studentId)
{
    var claimsIdentity = _httpContextAccessor.HttpContext.User.Identity as ClaimsIdentity;
    var userIdClaim = claimsIdentity?.FindFirst("UserId");
    studentId = userIdClaim.Value;
    try
    {
        var earnedPoints = await _examService.SubmitExam(id, selectedAnswers,studentId,selectedQuestions);

        var exam = await _db.Exams.FindAsync(id);
        if (exam == null)
        {
            return NotFound();
        }

        ViewBag.EarnedPoints = earnedPoints;
        ViewBag.TotalPoints = exam.TotalPoints;
        ViewBag.PassGrade = exam.PassGrade;

        if (earnedPoints >= exam.PassGrade)
        {
            ViewBag.Passed = true;
        }
        else
        {
            ViewBag.Passed = false;
        }

        return View("Result");
    }
    catch (Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Σχήμα 4.44: Μέθοδος HttpPost Submit-ExamController

```
foreach (var question in examQuestions)
{
    var correctAnswer = question.Answers.Where(a => a.IsCorrect == true).First();
    var userAnswers = selectedAnswers.ToList();

    // Υπολογισμός βαθμού μόνο για τις ερωτήσεις που έχουν επιλεγεί
    if (userAnswers.Contains(correctAnswer.Id))
    {
        earnedPoints += question.QuestionPoints;
    }
    else
    {
        earnedPoints -= question.NegativePoints;
    }
}
if(earnedPoints < 0)
{
    earnedPoints = 0.0;
}

    var examResult = new ExamResult
    {
        StudentId = studentId,
        ExamId = examId,
        Grade = earnedPoints
    };
```

Σχήμα 4.45: Μέθοδος SubmitExam-IExamService

Κατά την ολοκλήρωση της εξέτασης για να εξάγει ο καθηγητής τα αποτελέσματα των εξετάσεων καλείται η μέθοδος “ExportExamResultsToExcel()” του Σχήματος 4.46. Στην αρχή γίνεται ανάκτηση των αποτελεσμάτων της εξέτασης με βάσει το “id” της εξέτασης και αποθηκεύονται τα αποτελέσματα σε μία λίστα “examResults”. Έπειτα δημιουργήθηκε ένα “XLWorkbook” με την βοήθεια της βιβλιοθήκης “ClosedXML” και προστίθεται ένα νέο φύλλο εργασίας με όνομα “ExamResults”. Έπειτα ορίσθηκαν οι επικεφαλίδες των κελιών. Στην συνέχεια τροφοδοτούνται τα κελία με τα αποτελέσματα που είναι καταχωρημένα στην βάση ξεκινώντας από την δεύτερη γραμμή διότι στην πρώτη βρίσκονται οι επικεφαλίδες. Τέλος για την αποθήκευση του αρχείου δημιουργήθηκε ένα “MemoryStream” για την προσωρινή του αποθήκευση όπου αποθηκεύεται με την μέθοδο “SaveAs.ToArray()” ορίζοντας τον τύπο “MIME” του αρχείου “Excel” και επιστρέφεται στον χρήστη ως απάντηση το αρχείο με όνομα “ExamResultsExport”.

```
0 references
public IActionResult ExportExamResultsToExcel(int id)
{
    var examResults = _db.ExamResults.Where(er => er.ExamId == id).ToList();
    using (var workbook = new XLWorkbook())
    {



        var worksheet = workbook.Worksheets.Add("ExamResults");

        worksheet.Cell(1, 1).Value = "StudentId";
        worksheet.Cell(1, 2).Value = "ExamId";
        worksheet.Cell(1, 3).Value = "Grade";

        for (int i = 0; i < examResults.Count; i++)
        {
            var examResult = examResults[i];
            worksheet.Cell(i + 2, 1).Value = examResult.StudentId;
            worksheet.Cell(i + 2, 2).Value = examResult.ExamId;
            worksheet.Cell(i + 2, 3).Value = examResult.Grade;
        }

        using (var stream = new MemoryStream())
        {
            workbook.SaveAs(stream);
            var content = stream.ToArray();
            var mimeType = "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet";
            var fileName = "ExamResultsExport.xlsx";
            return File(content, mimeType, fileName);
        }

    }
}
```

Σχήμα 4.46: Μέθοδος ExportExamResultsToExcel

4.6 Επίλογος

Σε αυτό το κεφάλαιο αρχικά, αναφέρθηκαν οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του Back-End μέρους. Έπειτα αναλύθηκαν περιληπτικά οι διαδικασίες που γίνονται για την λειτουργικότητα της εφαρμογής.

Κεφάλαιο 5ο: Συμπεράσματα και Προτάσεις Βελτίωσης

Αρχικά, μετά και την ανάλυση της εφαρμογής και των λειτουργιών της, ως πρώτη πρόταση βελτίωσης της εφαρμογής και σημαντικότερη είναι η αποθήκευση των ήδη απαντημένων ερωτήσεων κάθε φορά που ο φοιτητής κάνει κλικ σε μία απάντηση ερώτησης. Αυτή η συγκεκριμένη υλοποίηση θα βοηθήσει σε περιπτώσεις αποσύνδεσης του φοιτητή έτσι ώστε να μπορεί κατά την είσοδο του ξανά στην εξέταση όπως επαναφέρονται μέσω της μνήμης Cache όπου αναλύσαμε στο προηγούμενο κεφάλαιο οι ερωτήσεις να επαναφέρονται έτσι και οι απαντήσεις. Αυτή η υλοποίηση θα είναι δύσκολη και χρονοβόρα καθώς η υπάρχουσα υποδομή αρχιτεκτονικής της εφαρμογής βασίζεται σε μία αντιμετώπιση όπου συνολικά οι απαντήσεις υποβάλλονται στον Server κατά την ολοκλήρωση της εξέτασης. Η Βάση Δεδομένων στην συγκεκριμένη περίπτωση θα πρέπει να είναι σχεδιασμένη να επιτρέπει την συνεχή ενημέρωση των απαντήσεων. Θα πρέπει να υπάρχει μία δομή όπου θα αποθηκεύονται οι απαντήσεις των ερωτήσεων σε πίνακες για κάθε φοιτητή ξεχωριστά. Μία άλλη λύση είναι αυτή της Cache όπως έγινε και με τις ερωτήσεις για να αποθηκεύονται προσωρινά σε αυτήν μέσω της βοήθειας βιβλιοθηκών όπως η «Memory Cache» το οποίο ήδη ήταν εκτός αρχικού σχεδιασμού και υλοποιήθηκε επιπλέον με κόστος τον χρόνο διεκπεραίωσης της εφαρμογής για την ομαλή διεξαγωγή των εξετάσεων.

Η πολυπλοκότητα και η δυσκολία πραγματοποίησης της συγκεκριμένης πρότασης στην εφαρμογή οφείλεται κυρίως σε δύο λόγους:

- **Session Management**: Για να διασφαλισθεί η ακεραιότητα των δεδομένων η εφαρμογή θα πρέπει να χειρίζεται με ασφάλεια όλες τις συνεδρίες των φοιτητών. Έτσι η κάθε απάντηση θα πρέπει να συνδέεται με την συνεδρία του φοιτητή ώστε να μειώνονται τα προβλήματα απώλειας δεδομένων. Η συγκεκριμένη αλλαγή θα μετατόπιζε κατά πολύ τον αρχικό προγραμματισμό εκτός χρονικού ορίου. Θα χρειαζόταν αρκετός χρόνος επιπλέον για να υλοποιηθεί και να γίνουν οι απαραίτητοι έλεγχοι έτσι ώστε να συμβαδίζει με την υπάρχουσα δομή της εφαρμογής.
- **Latency/Load Management**: Καθώς θα υπάρχει συνεχής επικοινωνία του Server με τον διακομιστή για να γίνεται σε Real-Time η αποθήκευση των απαντήσεων η συγκεκριμένη υλοποίηση απαιτεί λόγο του μεγάλου πλήθους των εξεταζόμενων την υποδομή η οποία να μπορεί να ανταπεξέλθει σε τόσες συνεχόμενες αιτήσεις. Θα πρέπει το δίκτυο αλλά και ο Server να είναι ικανοί να αντέξουν πολλαπλές αιτήσεις και να μην υπάρχουν καθυστερήσεις στην απόκριση.

Μέσα από κάποιες ακόμα αλλαγές θα μπορούσε η εφαρμογή να φιλοξενεί και όλες τις εξετάσεις των μαθημάτων του τμήματος έτσι ώστε να υπάρξει ένα ενιαίο σύστημα εξέτασης για όλους του καθηγητές αλλά και τους φοιτητές. Για να υλοποιηθεί αυτό θα πρέπει να υπάρξουν κάποιες ακόμα προτάσεις όπως:

- **Προσθήκη επιπλέον τρόπων εξέτασης**: Για να μπορεί η εφαρμογή να ανταπεξέλθει και στις απαιτήσεις των υπόλοιπων μαθημάτων θα πρέπει να μεγαλώσει το εύρος των επιλογών για την διαδικασία της εξέτασης με κυριότερη την επιλογή ερωτήσεων ανάπτυξης στην εφαρμογή. Αυτό θα βοηθήσει σε πιο θεωρητικά μαθήματα που απαιτούνται απαντήσεις ορισμών.
- **Λίστα Μαθημάτων**: Η εφαρμογή θα πρέπει να δίνει την δυνατότητα στους φοιτητές να κάνουν εγγραφή σε κάθε μάθημα που τους ενδιαφέρει. Με αυτήν την δυνατότητα θα γίνεται ο διαχωρισμός των εξετάσεων για κάθε μάθημα και δεν θα υπάρχει σύγχυση μεταξύ των μαθημάτων και των φοιτητών
- **Ιστορικό Εξετάσεων**: Για να μπορεί η εφαρμογή να ανταπεξέλθει σε ένα μεγαλύτερο εύρος φοιτητών αλλά και μαθημάτων θα πρέπει να υπάρχει η δυνατότητα αποθήκευσης των αποτελεσμάτων και η παραμονή τους στο σύστημα ανά μάθημα έτσι ώστε οι φοιτητές να μπορούν να βλέπουν και μέσω του συστήματος εξέτασης τα αποτελέσματα των μαθημάτων όπου έχουν εξεταστεί.

- **Ασφάλεια απομακρυσμένης εξέτασης**: Για να γίνει χρήση της εφαρμογής εξέτασης από όλα τα μαθήματα εξ αποστάσεως θα πρέπει να υπάρξουν επιπλέον μέτρα ασφάλειας και αξιοπιστίας του χρήστη. Η καταγραφή της IP καθώς και η ανάγκη περιορισμένης πρόσβασης στην εξέταση με βάσει κάποιο αναγνωριστικό του φοιτητή είναι αναγκαίο για να διασφαλιστεί η ομαλή εξέταση όλων των μαθημάτων.

Με την υλοποίηση των παραπάνω προτάσεων μπορεί να βελτιστοποιηθεί η εφαρμογή σε πολλούς τομείς όπως η ασφάλεια, η απόδοση αλλά και η ποικιλία της σε τρόπους εξέτασης δίνοντας στους φοιτητές ένα ολοκληρωμένο σύστημα εξέτασης ενιαίο όπου δεν θα χρειάζεται να γνωρίζουν πολλαπλές πλατφόρμες εξέτασης. Επίσης οι καθηγητές θα διαθέτουν ένα νέο εργαλείο προσαρμοσμένο στις ανάγκες του τμήματος όπου θα το χειρίζονται και θα το τροποποιούν αναλόγως χωρίς να χρειάζεται η υποστήριξη από τρίτους. Η συγκεκριμένη υλοποίηση θα μοιάζει αρκετά με το υπάρχων σύστημα εξέτασης «exams.iee.ihu.gr» όπου αναφέραμε προηγουμένως με παραπάνω λειτουργίες οι οποίες θα βοηθούν συγκεκριμένα την εξέταση για κάθε μάθημα του τμήματος.

Αν και οι προτάσεις που αναφέρθηκαν παραπάνω μπορούν να δώσουν την δυνατότητα στο τμήμα να διατηρεί το δικό του σύστημα εξέτασης χωρίς να χρειάζεται οι καθηγητές να δημιουργούν ξεχωριστά πλατφόρμες εξέτασης για το κάθε μάθημα τους ή να στηρίζονται σε πλατφόρμες τρίτων, θα πρέπει να επισημανθεί ότι το συγκεκριμένο εγχείρημα θα χρειαστεί αρκετό πρόσθετο χρόνο για την υλοποίηση του καθώς και την προσαρμογή τους στο υπάρχων σύστημα εξέτασης, έτσι ώστε να μπορούν να υποστηρίζονται οι ήδη υπάρχουσες λειτουργίες αλλά και να γίνουν οι προσθήκες των νέων λειτουργιών. Συνεπώς, η υλοποίηση όλων αυτών των προτάσεων που προαναφέρθηκαν θα πρέπει να γίνουν με σωστή μελέτη και ανάλυση των υπαρχόντων λειτουργιών αλλά και την εύρεση του τρόπου ένταξης των νέων λειτουργιών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] “Overview of ASP.NET Core MVC”, Microsoft, September 26, 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/.

[2] “Razor syntax reference for ASP.NET Core”, Microsoft, November 14, 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/views/.

[3] “Entity Framework Core”, Microsoft, May 25, 2021. [Online]. Available: https://learn.microsoft.com/en-us/ef/core/.

[4] Michael Goodwin, “What is an API”, IBM, April 09, 2024. [Online]. Available: https://www.ibm.com/topics/api.

[5] Sivakumar Mani, “Web application architecture types”, Medium.com, May 14, 2023. [Online]. Available: https://medium.com/@sivakumarjd/web-application-architecture-types-single-tier-architecture-monolithic-architecture-two-tier-f3d5c9f2aff6.

[6] “What is a Web Application?”, Geeksforgeeks.org, January 12, 2024. [Online]. Available: https://www.geeksforgeeks.org/what-is-web-app/.

[7] “HTML Introduction”, Geeksforgeeks.org, August 06, 2024. [Online]. Available: https://www.geeksforgeeks.org/html-introduction/.

[8] “What is CSS?”, Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS.

[9] “What is JavaScript?”, Mozilla. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[10] “Introduction to JavaScript”, Geeksforgeeks.org, July 26, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-javascript/.

[11] “Get Started With Boostrap”, Bootstrap. [Online]. Available: https://getbootstrap.com/docs/5.3/getting-started/introduction/.

[12] A.Jordana, “What Is Bootstrap?”, Hostinger.com, May 19, 2023. [Online]. Available: https://www.hostinger.com/tutorials/what-is-bootstrap/.

[13] “What Is AngularJS?”, Angularjs.org. [Online]. Available: https://docs.angularjs.org/guide/introduction.

[14] “What is Vue?”, Vuejs.org. [Online]. Available: https://vuejs.org/guide/introduction.html.

[15] “About Node.js”, nodejs.org. [Online]. Available: https://nodejs.org/en/about.

[16] Benjamin Semah, “What Exactly is Node.js? Explained for Beginners?”, Freecodecamp.org, December 05, 2022. [Online]. Available: https://www.freecodecamp.org/news/what-is-node-js/.

[17] “What Is a Database?”, Oracle.com, November 24, 2020. [Online]. Available: https://www.oracle.com/database/what-is-database/.

[18] “What is SQL Server?”, Microsoft, February 14, 2024. [Online]. Available: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver16.

[19] “What is NoSQL?”, MongoDB. [Online]. Available: https://www.mongodb.com/resources/basics/databases/nosql-explained .

[20] “What is MySQL?”, MySQL. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html.

ΠΑΡΑΡΤΗΜΑ Α : Front-End

Σελίδα Κατηγορίες Ερωτήσεων

```
@model IEnumerable<ExamThesis.Storage.Model.QuestionCategory>
@{
   ViewData["Title"] = "Index";
}
<div class="container p-3">
   <div class="row pt-4">
      <div class="col-6">
         <h2 class="text-primary">Question Category List</h2>
      </div>
      <div class="col-6 text-end">
         <a asp-controller="QuestionCategory" asp-action="Create" class="btn btn-primary">
            <i class="bi bg-plus-circle"></i>   Create New Category
         </a>
      </div>
   </div>
</div>
<br /><br />

<table class="table table-bordered table-striped" style="width:100%">
   <thead>
      <tr>
         <th>
            CategoryId
         </th>
         <th>
            CategoryName
         </th>
         <th></th>
      </tr>
   </thead>
   <tbody>
      @foreach (var categories in Model)
      {
         <tr>
            <td width="50%">
               @categories.QuestionCategoryId
            </td>
            <td width="30%">
               @categories.QuestionCategoryName
            </td>
            <td>
               <div class="w-75 btn-group" role="group">
                  <a asp-controller="QuestionCategory" asp-action="Delete" asp-route-
id="@categories.QuestionCategoryId"
                  class="btn btn-danger mx-2"> <i class="bi bi-pencil-square"></i> Delete </a>

               </div>
            </td>
         </tr>
      }
   </tbody>
```

Σελίδα Διαγραφής Κατηγοριών

```
@model ExamThesis.Storage.Model.QuestionCategory

<h2>Delete Category</h2>

<p>Are you sure you want to delete this category?</p>

<dl class="row">
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.QuestionCategoryId)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.QuestionCategoryId)
    </dd>

    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.QuestionCategoryName)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.QuestionCategoryName)
    </dd>
</dl>

<div class="form-group">
    <a asp-action="Index" class="btn btn-secondary">Back to List</a>
    <button id="deleteButton" class="btn btn-danger">Delete</button>
</div>

@section Scripts {
    <script>
        $(function () {
            $('#deleteButton').click(function () {
                if (confirm("Are you sure you want to delete this category?")) {
                    $.ajax({
                        url: '/QuestionCategory/DeletePost/' + @Model.QuestionCategoryId,
                        type: 'DELETE',
                        headers: {
                            'RequestVerificationToken': $('input[name="__RequestVerificationToken"]').val()
                        },
                        success: function (data) {
                            console.log('Success:', data);
                            window.location.href = '/QuestionCategory/Index';
                        },
                        error: function (error) {
                            console.error('Error:', error);
                        }
                    });
                    window.location.href = '/Exam/Index';
                }
            });
        });
    </script>
}
```

Σελίδα Δημιουργίας Ερωτήσεων/Απαντήσεων

```
@model ExamThesis.Models.CreateQuestion

<h2>Create Question and Answers</h2>

<form asp-action="Create" method="post">
    <div class="form-group">
        <label asp-for="QuestionText" class="control-label"></label>
        <input asp-for="QuestionText" class="form-control" />
    </div>

    <div class="form-group">
        <label asp-for="QuestionPoints" class="control-label"></label>
        <input asp-for="QuestionPoints" class="form-control" />
    </div>

    <div class="form-group">
        <label asp-for="NegativePoints" class="control-label"></label>
        <input asp-for="NegativePoints" class="form-control" />
    </div>

    <div class="form-group">
        <label asp-for="QuestionCategoryId" class="control-label">QuestionCategory</label>
        <select asp-items="ViewBag.QuestionCategories" asp-for="QuestionCategoryId" class="form-
control" ></select>
    </div>
    <div class="form-group">
        <label asp-for="PackageId" class="control-label">Question Package</label>
        <select asp-items="ViewBag.QuestionPackages" asp-for="PackageId" class="form-
control"></select>
    </div>

    <h4>Answers</h4>
    <div id="answers-container">
        @for (var i = 0; i < Model.Answers.Count; i++)
        {
            <div class="form-group">
                <label asp-for="Answers[i].Text" class="control-label"></label>
                <input asp-for="Answers[i].Text" class="form-control" />

                <label>
                    <input type="checkbox" asp-for="Answers[i].IsCorrect" />
                    Correct Answer
                </label>
            </div>
        }
    </div>

    <button type="button" class="btn btn-secondary" onclick="addAnswer()">Add Answer</button>
    <button type="button" class="btn btn-secondary" onclick="removeAnswer()">Remove
Answer</button>

    <button type="submit" class="btn btn-primary">Create Question and Answers</button>
```

```
@* @section scripts {
   <script>
      function addAnswer() {
         var container = $('#answers-container');
         var index = container.children().length+1;
         var count = @Model.Answers.Count();
         var test = new ExamThesis.Storage.Model.CreateAnswer();
         @Model.Answers.Add(test);

         var answerHtml =`
            <div class="form-group">
               <label asp-for="Answers[index].Text" class="control-label"></label>
               <input asp-for="Answers[index].Text" class="form-control" />

               <label>
                  <input type="checkbox" asp-for="Answers[${index}].IsCorrect" />
                  Correct Answer
               </label>
            </div>
         `;

         container.append(answerHtml);
      } *@
   @section scripts {
      <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>
      <script>

         function addAnswer() {
            var container = $('#answers-container');
            var index = container.children().length;
            var isChecked = $(".answer-checkbox").is(":checked") ? "true" : "false";
            console.log(isChecked);
            // Κατασκευή νέας απάντησης ως HTML
            var newAnswerHtml = `
                     <div class="form-group">
                        <label for="Answers_${index}__Text" class="control-label">Answer ${index +
1}</label>
                        <input type="text" name="Answers[${index}].Text" class="form-control" />
                        <label>
                           <input type="checkbox" class="answer-checkbox"
name="Answers[${index}].IsCorrect" value="true" />
                           Correct Answer
                        </label>
                     </div>
                  `;

            // Προσθήκη της νέας απάντησης στο container
            container.append(newAnswerHtml);
         }

      function removeAnswer() {
         var container = $('#answers-container');
         var childrenCount = container.children().length;
```

```
            if (childrenCount > 1) {
                container.children().last().remove();
            }
        }
    </script>
}
</form>
```

Σελίδα Δημιουργίας Εξέτασης

```
@model ExamThesis.Models.CreateExam
@using ExamThesis.Storage.Model
@{
    ViewData["Title"] = "Create Exam";
}

<h2>Create Exam</h2>

<form asp-action="Create" method="post">
    <div class="form-group">
        <label asp-for="ExamName" class="control-label">Όνομα Εξέτασης:</label>
        <input asp-for="ExamName" class="form-control" />
        <span asp-validation-for="ExamName" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="StartTime" class="control-label">Έναρξη:</label>
        <input asp-for="StartTime" class="form-control" />
        <span asp-validation-for="StartTime" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="EndTime" class="control-label">Λήξη:</label>
        <input asp-for="EndTime" class="form-control" />
        <span asp-validation-for="EndTime" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="TotalPoints" class="control-label">Συνολικοί Πόντοι:</label>
        <input asp-for="TotalPoints" class="form-control" />
        <span asp-validation-for="TotalPoints" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="PassGrade" class="control-label">Ελάχιστη Βάση Εξέτασης:</label>
        <input asp-for="PassGrade" class="form-control" />
        <span asp-validation-for="PassGrade" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label  class="control-label">Επιλογή Κατηγοριών Ερωτήσεων:</label>
        <div id="answers-container">
            @for (var i = 0; i < Model.SelectedCategories.Count; i++)
            {
```

```
            var currentcategory =Model.SelectedCategories[i];

            @*  <div class="form-group">
                <input type="checkbox" asp-for="SelectedCategories[]"
                    name="@currentcategory.QuestionCategoryName">
@currentcategory.QuestionCategoryName

            </div> *@
            <div class="form-group">


                <input type="checkbox" asp-for="SelectedCategories[i].IsChecked"  />
                @currentcategory.QuestionCategoryName
                <input                  id="@currentcategory.QuestionCategoryId"                  asp-
for="SelectedCategories[i].QuestionCategoryId"                                                    type="hidden"
value="@currentcategory.QuestionCategoryId"/>
              <input                  id="@currentcategory.QuestionCategoryName"                  asp-
for="SelectedCategories[i].QuestionCategoryName"                                                  type="hidden"
value="@currentcategory.QuestionCategoryName" />
            </div>
          }
      </div>
   </div>

   <button type="submit" class="btn btn-primary">Create</button>
</form>
```