ΠΑΝΕΠΙΣΤΗΜΙΟ
ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ
UNIVERSITY OF WEST ATTICA

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΥΠΟΔΟΜΗΣ ΠΑΡΟΧΗΣ WEB ΥΠΗΡΕΣΙΩΝ ΣΕ ΦΟΙΤΗΤΕΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΥΠΟΔΟΜΗΣ ΠΑΡΟΧΗΣ WEB ΥΠΗΡΕΣΙΩΝ ΣΕ ΦΟΙΤΗΤΕΣ

Αντώνιος Κάσσος

Α.Μ. 711141062

Εισηγητής:

Ιωάννης Βογιατζής, Καθηγητής

Εξεταστική Επιτροπή:

1. Ιωάννης Βογιατζής, Καθηγητής
2. Χρήστος Τρούσσας, Επίκουρος Καθηγητής
3. Γεώργιος Μελετίου, ΕΔΙΠ

Ημερομηνία Εξέτασης:

10/10/2023

Ioannis Voyiatzis

Digitally signed by Ioannis Voyiatzis
Date: 2024.01.09 07:59:57 +02'00'

Christos Troussas

Digitally signed by Christos Troussas
Date: 2024.01.09 10:42:37 +02'00'

Georgios Meletiou

Digitally signed by Georgios Meletiou
Date: 2024.01.09 22:00:35 +02'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Κάσσος Αντώνιος του Αλέξανδρου, με αριθμό μητρώου 711141062 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Μηχανικών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της σχεδίασης και ανάπτυξης ενός πληροφοριακού συστήματος. Την προσπάθειά μου αυτή υποστήριξε ο επιβλέποντας καθηγητής μου Ιωάννης Βογιατζής, τον οποίο θα ήθελα να ευχαριστήσω θερμά. Επίσης, θα ήθελα να ευχαριστήσω ιδιαίτερα τον κύριο Γεώργιο Μελετίου, για το χρόνο που αφιέρωσε και την καθοδήγηση που παρείχε κατά την εκπόνηση της παρούσας εργασίας. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για τη συμπαράσταση καθ' όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία ασχολείται με τον σχεδιασμό και την ανάπτυξη ενός πληροφοριακού συστήματος υποδομής χρησιμοποιώντας σύγχρονες τεχνολογίες, με σκοπό τη μηχανοργάνωση διαδικασιών που αφορούν τις πτυχιακές και διπλωματικές εργασίες των φοιτητών, όπως αυτές ορίζονται στον Εσωτερικό Κανονισμό του Πανεπιστημίου Δυτικές Αττικής. Το πληροφοριακό σύστημα καλείται να επιλύσει ορισμένα προβλήματα που αναδεικνύονται μέσα από την ανάλυση της υφιστάμενης κατάστασης, καθώς και να απλοποιήσει διαδικασίες όπως η αίτηση, η ανάθεση και η αξιολόγηση των πτυχιακών και διπλωματικών εργασιών, για όλους τους εμπλεκόμενους. Η εργασία εκπονήθηκε με στόχο τη πλήρη αξιοποίηση του υλοποιημένου συστήματος υποδομής από το Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών και τη διασύνδεση του τόσο με υπάρχουσες όσο και με μελλοντικές εφαρμογές του πανεπιστημίου, ώστε να διασφαλίζεται η ομαλή διεκπεραίωση των διπλωματικών εργασιών από την αρχή ως τέλος.

ABSTRACT

This thesis deals with the design and development of an infrastructure information system using modern technologies, with the purpose of computerizing procedures related to students' diploma theses, as defined in the Internal Regulations of the University of Western Attica. The information system is called upon to resolve certain problems that emerge through the analysis of the existing situation, as well as to simplify procedures such as the application, assignment, and evaluation of diploma theses, for all those involved. The project was created with the aim of fully utilizing the implemented infrastructure system by the Department of Information and Computer Engineering and its integration with both existing and future applications of the University, to ensure the smooth completion of diploma theses from start to finish.

Περιεχόμενα

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ 4
ΕΥΧΑΡΙΣΤΙΕΣ 5
ΠΕΡΙΛΗΨΗ 7
ABSTRACT 8
Κατάλογος Εικόνων 12
Κατάλογος Πινάκων 13
Κατάλογος Διαγραμμάτων 13
Συντομογραφίες 14
Κεφάλαιο 1 – Εισαγωγή 15
1.1. Περιγραφή του αντικειμένου της Διπλωματικής εργασίας 15
1.2. Ανασκόπηση της υφιστάμενη διαδικασίας 15
1.3. Διαπίστωση προβλημάτων 17
1.4. Προτεινόμενη Λύση 18
Κεφάλαιο 2 – Θεωρητικό Υπόβαθρο 20
2.1. Η γλώσσα προγραμματισμού C# 20
2.1.1. Ιστορική αναδρομή 20
2.1.2. Κύρια χαρακτηριστικά και πλεονεκτήματα 21
2.1.3. Πρακτικές εφαρμογές της C# 22
2.2. Η πλατφόρμα ανάπτυξης .NET 22
2.2.1. Αρχιτεκτονική του .NET 23
2.2.2. Ιστορικό Εκδόσεων 24
2.2.3. Πλεονεκτήματα του .NET 33
2.3. Σχεσιακές Βάσεις Δεδομένων 35
2.3.1. Εισαγωγή Σχεσιακό Μοντέλο Δεδομένων 35
2.3.2. Σχέσεις 35
2.3.3. Πεδίο τιμών 36
2.3.4. Σχήμα και Στιγμιότυπο Σχέσης 36
2.3.5. Κλειδιά 37
2.3.6. Περιορισμοί 38

2.3.7. Κανονικοποίηση ... 39
2.4. Η γλώσσα Βάσεων Δεδομένων SQL ... 39
2.4.1. Σύντομη ιστορική αναδρομή ... 39
2.4.2. Βασικές αρχές της SQL ... 40
2.4.3. Σύνταξη και βασικές λειτουργίες ... 41
2.4.4. Πρακτικές εφαρμογές της SQL ... 43
2.5. Ανταλλαγή δεδομένων και επικοινωνία μεταξύ συστημάτων ... 43
2.5.1. Υπηρεσίες Ιστού (Web Services) ... 43
2.5.2. Διεπαφές Προγραμματισμού Εφαρμογών (APIs) ... 44
2.5.3. Διαφορές μεταξύ Web Service και API ... 45
Κεφάλαιο 3 – Μεθοδολογία ... 47
3.1. Εργαλεία Ανάπτυξης ... 47
3.1.1. Visual Studio ... 47
3.1.2. SQL Server Management Studio ... 50
3.2. Σύστημα Διαχείρισης Εκδόσεων ... 52
3.2.1. Το σύστημα Git ... 52
3.2.2. Η πλατφόρμα GitHub ... 53
3.3. Επιπρόσθετες τεχνολογίες που χρησιμοποιήθηκαν ... 54
3.3.1. Entity Framework Core ... 54
3.3.2. Swagger ... 56
3.4. Σχεδιαστικά Πρότυπα & Βέλτιστες Πρακτικές ... 58
3.4.1. Η αρχιτεκτονική REST ... 58
3.4.2. Dependency Injection ... 60
3.4.3. Repository Pattern ... 60
Κεφάλαιο 4 – Σχεδιασμός & Ανάπτυξη Πληροφοριακού Συστήματος ... 62
4.1. Αρχιτεκτονική ... 62
4.2. Βάση Δεδομένων ... 63
4.2.1. Σχεδιάγραμμα Οντότητας/Σχέσης ... 63
4.2.2. Πίνακες της Βάσης Δεδομένων ... 64
4.3. REST API ... 70
4.3.1. Πίνακες κλήσεων του REST API ... 70
4.4. Ταυτοποίηση Χρηστών ... 86

4.4.1. Το πρωτόκολλο SAML 2.0 ... 86
4.4.2. Η πλατφόρμα Okta ως Identity Provider ... 88
4.4.3. Σύνδεση χρήστη μέσω του SSO ... 92
Κεφάλαιο 5 – Συμπεράσματα ... 95
5.1. Μελλοντικές προσθήκες και βελτιώσεις ... 95
Παράρτημα Α ... 97
Κώδικας SQL - Πίνακες ... 97
Παράρτημα Β ... 101
Κώδικας C# - Models & Classes ... 101
Κώδικας C# - Repositories ... 116
Κώδικας C# - Controllers ... 128
Κώδικας C# - Utilities ... 151
Κώδικας C# - Main ... 182
Configuration ... 187
Βιβλιογραφία & Πηγές ... 188

Κατάλογος Εικόνων

Εικόνα 2.1 – Το λογότυπο της C# 21
Εικόνα 2.2 – Αρχιτεκτονική του .NET 24
Εικόνα 2.3 – Επισκόπηση των εκδόσεων του .NET 25
Εικόνα 2.4 – Αρχικές εκδόσεις του .NET Framework 25
Εικόνα 2.5 – Ανάπτυξη εφαρμογής WinForms μέσω Visual Studio 26
Εικόνα 2.6 – Η έκδοση 3.0 του .NET Framework 27
Εικόνα 2.7 – Τελευταίες εκδόσεις του .NET Framework 28
Εικόνα 2.8 – Αρχικές εκδόσεις του .NET Core 30
Εικόνα 2.9 – Η νέα γενιά του .NET 31
Εικόνα 2.10 – Hosting models της τεχνολογίας Blazor 32
Εικόνα 2.11 – Συσχέτιση πινάκων με τη χρήση Foreign Key 38
Εικόνα 2.12 – Ανταλλαγή δεδομένων μέσω APIs & Web Services 46
Εικόνα 3.1 – Δυνατότητες του Visual Studio 47
Εικόνα 3.2 – Παράδειγμα αυτόματης συμπλήρωσης και συμβουλής του IntelliSense 48
Εικόνα 3.3 – Παράδειγμα ενός tooltip του IntelliSense 48
Εικόνα 3.4 – Παράδειγμα debugging εργαλείων του Visual Studio 49
Εικόνα 3.5 – Εγκατάσταση βιβλιοθηκών μέσω του Nuget Package Manager 49
Εικόνα 3.6 – Το περιβάλλον του Visual Studio 2022 50
Εικόνα 3.7 – Το περιβάλλον του SQL Server Management Studio 51
Εικόνα 3.8 – Ιστορικό τροποποιήσεων του κώδικα μέσω του Visual Studio 53
Εικόνα 3.9 – Αποθετήριο του κώδικα της εφαρμογής στη πλατφόρμα GitHub 54
Εικόνα 3.10 – Προσεγγίσεις Code-First & Database-First του Entity Framework 55
Εικόνα 3.11 – Η κλάση της οντότητας «Course» 55
Εικόνα 3.12 – Κώδικας migration «AddCoursesTable» 55
Εικόνα 3.13 – Τεκμηρίωση REST API μέσω του Swagger 56
Εικόνα 3.14 – Εκτέλεση δοκιμαστικής κλήσης μέσω του Swagger UI 57
Εικόνα 3.15 – Επικοινωνία client-server μέσω REST API 58
Εικόνα 3.16 – Εκτέλεση CRUD λειτουργιών βάσει URI 59
Εικόνα 3.17 – Διεπαφή αποθετηρίου θεμάτων 61
Εικόνα 3.18 – Υλοποίηση αποθετηρίου θεμάτων 61
Εικόνα 4.1 – Διαδικασία ταυτοποίησης χρήστη σύμφωνα με το SAML 2.0 87
Εικόνα 4.2 – Δημιουργία νέας εφαρμογής διασύνδεσης στη πλατφόρμα Okta 89
Εικόνα 4.3 – Ορισμός ονόματος εφαρμογής διασύνδεσης στη πλατφόρμα Okta 89
Εικόνα 4.4 – Παραμετροποίηση SAML 2.0 IdP στη πλατφόρμα Okta 90
Εικόνα 4.5 – Ορισμός προσαρμοσμένων χαρακτηριστικών χρήστη 90
Εικόνα 4.6 – Δήλωση των IdP Metadata στον κώδικα της εφαρμογής 91
Εικόνα 4.7 – Προσθήκη χρήστη στον IdP 91
Εικόνα 4.8 – Συμπλήρωση προσαρμοσμένων χαρακτηριστικών χρήστη 92
Εικόνα 4.9 – Με εξουσιοδοτημένη κλήση προς το REST API 92

Εικόνα 4.10 – Σύνδεση χρήστη μέσω Single Sign-On 93
Εικόνα 4.11 – Session Cookie συνδεδεμένου χρήστη 93
Εικόνα 4.12 – Εξουσιοδοτημένη κλήση προς το REST API 94

Κατάλογος Πινάκων

Πίνακας 2.1 - Η σχέση Students 36
Πίνακας 4.1 - Δομή του πίνακα «ThesisProjects» της Β.Δ. 65
Πίνακας 4.2 - Δομή του πίνακα «ThesisApplications» της Β.Δ. 65
Πίνακας 4.3 - Δομή του πίνακα «Assignments» της Β.Δ. 66
Πίνακας 4.4 - Δομή του πίνακα «AssignmentsHistory» της Β.Δ. 67
Πίνακας 4.5 - Δομή του πίνακα «ExaminationCommittees» της Β.Δ. 67
Πίνακας 4.6 - Δομή του πίνακα «Examinations» της Β.Δ. 68
Πίνακας 4.7 - Δομή του πίνακα «Courses» της Β.Δ. 69
Πίνακας 4.8 - Δομή του πίνακα «Lookup» της Β.Δ. 69
Πίνακας 4.9 - Δομή του πίνακα «EFMigrationHistory» της Β.Δ. 69
Πίνακας 4.10 - Πίνακας κλήσεων του ApplicationsController 74
Πίνακας 4.11 - Πίνακας κλήσεων του ProjectsController 77
Πίνακας 4.12 - Πίνακας κλήσεων του ValuesController 79
Πίνακας 4.13 - Πίνακας κλήσεων του AssignmentsController 84
Πίνακας 4.14 - Πίνακας κλήσεων του AuthenticationController 85

Κατάλογος Διαγραμμάτων

Διάγραμμα 1.1 - Διάγραμμα Ροής διαδικασίας Πτυχιακών/Διπλωματικών εργασιών 17
Διάγραμμα 4.1 - Αρχιτεκτονική του Πληροφοριακού Συστήματος 62
Διάγραμμα 4.2 - Διάγραμμα Οντοτήτων/Συσχετίσεων της Β.Δ. (ER Diagram) 64

Συντομογραφίες

Π.Σ. – Πληροφοριακό Σύστημα

Β.Δ. – Βάση Δεδομένων

IDE – Integrated Development Environments

GUI – Graphical User Interface

REST – Representational State Transfer

API – Application Programming Interface

DI – Dependency Injection

EF – Entity Framework

PK – Primary Key

FK – Foreign Key

DBMS – Database Management System

RDBMS – Relational Database Management System

SAML – Security Assertion Markup Language

IdP – Identity Provider

SP – Service Provider

Κεφάλαιο 1 – Εισαγωγή

Σε αυτό το κεφάλαιο περιγράφεται το αντικείμενο της παρούσας Διπλωματικής εργασίας και οι λόγοι για τους οποίους εκπονήθηκε, ενώ πραγματοποιείται ανάλυση της υφιστάμενης κατάστασης επί του θέματος. Μέσα από την ανάλυση γίνεται η διαπίστωση των προβλημάτων που αναδεικνύονται και στη συνέχεια περιγράφεται πως αποσκοπεί στην επίλυση αυτών η λύση που ακολουθήθηκε.

1.1. Περιγραφή του αντικειμένου της Διπλωματικής εργασίας

Αντικείμενο της παρούσας Διπλωματικής εργασίας είναι η σχεδίαση και ανάπτυξη ενός Πληροφοριακού Συστήματος (Π.Σ.) για την καλύτερη διαχείριση των Πτυχιακών και Διπλωματικών Εργασιών των φοιτητών του Τμήματος Μηχανικών Πληροφορικής. Πιο συγκεκριμένα, επικεντρώνεται στην υλοποίηση μίας Back-End υποδομής που αποτελείται από ένα REST API και μίας Βάσης Δεδομένων, με το οποίο θα μπορούν να επικοινωνούν υπάρχουσες ή και νέες Front-End εφαρμογές, προκειμένου οι χρήστες, δηλαδή οι φοιτητές και οι καθηγητές, να μπορούν να αλληλοεπιδρούν με το σύστημα. Για να μπορέσουν οι χρήστες να επικοινωνήσουν με το πληροφοριακό σύστημα, χρειάζεται να συνδεθούν σε έναν κεντρικό Identity Provider (ένα σύστημα διαχείρισης και αποθήκευσης στοιχείων χρήστη). Η αυθεντικοποίηση των χρηστών γίνεται σύμφωνα με το πρωτόκολλο SAML 2.0, το οποίο υποστηρίζεται από τη κεντρική υπηρεσία ταυτοποίησης των χρηστών του Πανεπιστημίου Δυτικής Αττικής, επιτρέποντας έτσι τη μελλοντική διασύνδεση της με το Π.Σ.

Το Π.Σ. σχεδιάστηκε με στόχο να βοηθήσει στην επίλυση των διαφόρων προβλημάτων που αντιμετωπίζουν οι καθηγητές και οι φοιτητές με την υπάρχουσα διαδικασία που ακολουθείται αναφορικά με τις Διπλωματικές και Πτυχιακές εργασίες, και παράλληλα να συμβάλλει στη καλύτερη μηχανοργάνωση του τμήματος.

1.2. Ανασκόπηση της υφιστάμενη διαδικασίας

Οι φοιτητές του Τμήματος Μηχανικών Πληροφορικής υποχρεούνται σε εκπόνηση Πτυχιακής ή Διπλωματικής εργασίας όπως προβλέπεται στο Πρόγραμμα Σπουδών του τμήματος. Συγκεκριμένα, οι παλαιοί φοιτητές που ακολουθούν κύκλο σπουδών ΤΕΙ υποχρεούνται σε εκπόνηση πτυχιακής εργασίας, ενώ οι νεότεροι που ακολουθούν το νέο πρόγραμμα σπουδών του ΠΑΔΑ σε εκπόνηση διπλωματικής εργασίας.

Βάσει του Άρθρου 40 του Εσωτερικού Κανονισμού του Πανεπιστήμιού Δυτικής Αττικής [1], στην αρχή του εξαμήνου κάθε μέλος του Διδακτικού Ερευνητικού Προσωπικού (Δ.Ε.Π.) εισηγείται τον αριθμό των Πτυχιακών/Διπλωματικών που επιθυμεί να επιβλέπει. Τα θέματα εγκρίνονται από τη Γενική Συνέλευση και αναρτώνται στην ιστοσελίδα του Τμήματος. Οι φοιτητές από τη μεριά τους επιλέγουν ένα θέμα και έπειτα από συνεννόηση με τον επιβλέποντα καθηγητή, υποβάλλουν τη σχετική αίτηση ανάληψης θέματος Πτυχιακής/Διπλωματικής εργασίας προς τη γραμματεία του Τμήματος. Εφόσον η αίτηση

ανάληψης εγκριθεί, η συνέλευση του τμήματος ορίζει τη Τριμελή Εξεταστική Επιτροπή για τη συγκεκριμένη εργασία, όπου το ένα μέλος είναι ο επιβλέπων καθηγητής.

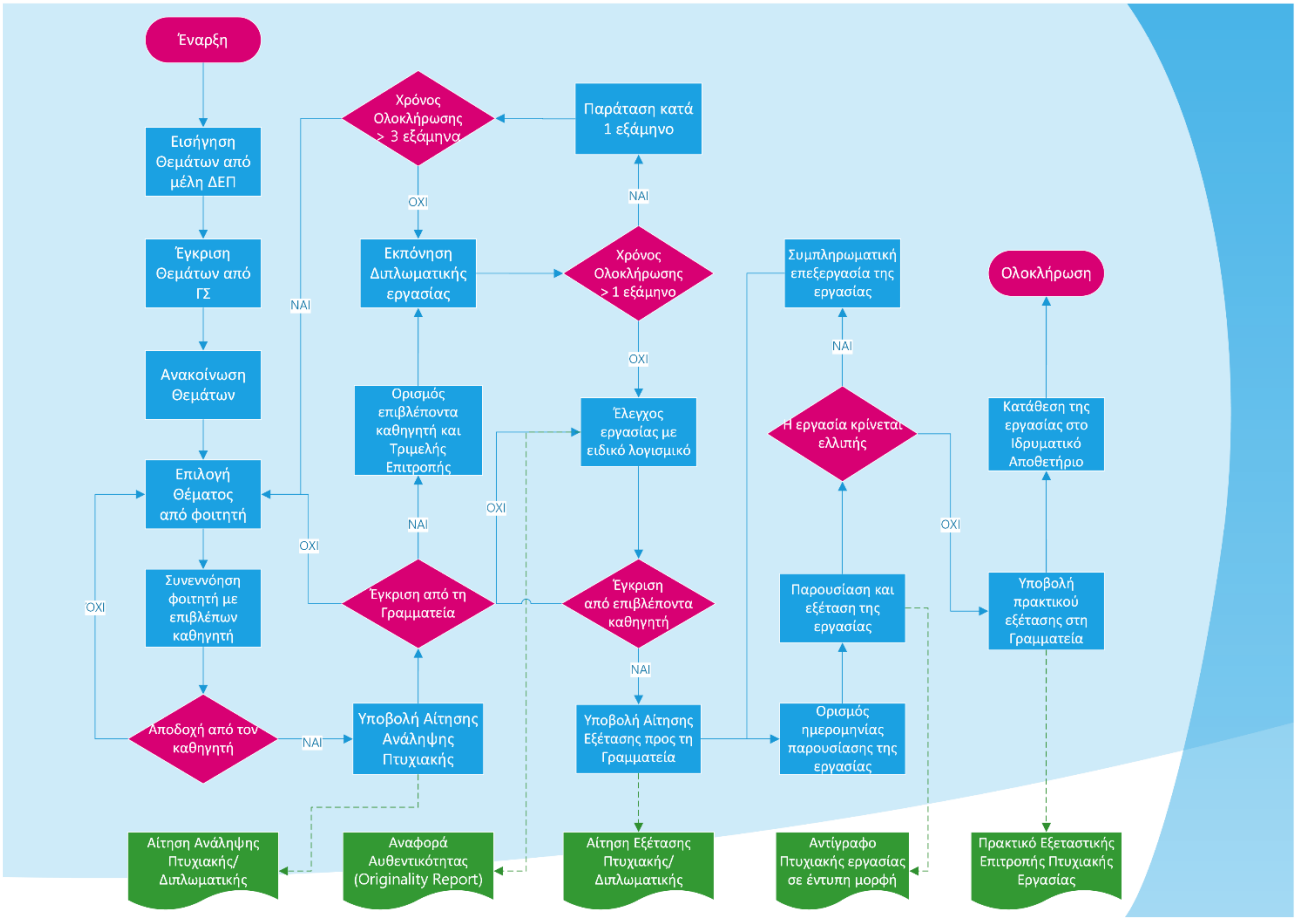
Κατά γενικό κανόνα, οι Πτυχιακές/Διπλωματικές εργασίες εκπονούνται ατομικά από τον κάθε φοιτητή. Αν η φύση ενός θέματος το απαιτεί, η εκπόνηση μπορεί κατ' εξαίρεση να γίνει από ομάδα το πολύ δύο φοιτητών, υπό την προϋπόθεση ότι είναι διακριτή η ατομική εργασία και συμβολή και των δύο φοιτητών τόσο κατά την εκπόνηση όσο και κατά την παρουσίαση της εργασίας.

Από τη στιγμή ανάληψης μιας Πτυχιακής/Διπλωματικής εργασίας, οι φοιτητές έχουν περιθώριο ενός ακαδημαϊκού εξαμήνου για να την ολοκληρώσουν. Ωστόσο, ο χρόνος ολοκλήρωσης επιδέχεται παράτασης κατά ένα εξάμηνο, εφόσον εγκριθεί από τον επιβλέποντα καθηγητή. Κατ' εξαίρεση και αφότου υπάρξει αιτιολογημένο αίτημα από το φοιτητή, ο χρόνος ολοκλήρωσης μπορεί να παραταθεί για ένα επιπλέον εξάμηνο. Εάν παρέλθει η προθεσμία των τριών εξαμήνων, τότε ο υποχρεούται να καταθέσει εκ νέου αίτηση ανάληψης θέματος Πτυχιακής/Διπλωματικής εργασίας.

Όλες οι εργασίες πριν εξεταστούν, υποβάλλονται σε έλεγχο με τη χρήση ειδικού λογισμικού ανίχνευσης λογοκλοπής ή ταύτισης με άλλες ακαδημαϊκές εργασίες ή πηγές από τη βάση δεδομένων που διαθέτει. Από τον έλεγχο αυτό προκύπτει η αναφορά αυθεντικότητας (Originality Report) στην οποία αναφέρεται το ποσοστό ταύτισης. Ο επιβλέποντας καθηγητής αξιολογεί την εν λόγω αναφορά και εφόσον επιβεβαιώσει την αυθεντικότητα και την εγκυρότητα της εργασίας, εγκρίνει την παρουσίαση και εξέταση της ενώπιον της Τριμελούς Εξεταστικής Επιτροπής. Ο φοιτητής υποβάλλει τη σχετική αίτηση εξέτασης και κατόπιν συνεννόησης με τον επιβλέποντα καθηγητή ορίζεται η ημερομηνία εξέτασης της εργασίας.

Ο φοιτητής έχει την ευθύνη να καταθέσει επί της Τριμελούς Εξεταστικής Επιτροπής αντίτυπο της Πτυχιακής/Διπλωματικής εργασίας σε ηλεκτρονική ή και έντυπη μορφή. Κατά τη παρουσίαση ο φοιτητής καλείται να απαντήσει σε σχετικές ερωτήσεις επί του θέματος, ώστε να διαπιστωθεί η ορθότητα και η πληρότητα της λύσης που δόθηκε στο σχετικό πρόβλημα, καθώς και ο βαθμός συμμετοχής του εφόσον αφορά ομαδική εργασία. Η επιτροπή κρίνει και βαθμολογεί, ως προς καθορισμένα κριτήρια, ανεξάρτητα καθέναν από τους συμμετέχοντες φοιτητές και ο τελικός βαθμός της Πτυχιακής/Διπλωματικής εργασίας προκύπτει από το μέσο όρο της βαθμολογίας των μελών τους Τριμελούς Επιτροπής. Το πρακτικό εξέτασης με την αναλυτική βαθμολογία υποβάλλεται στη Γραμματεία του Τμήματος από τον επιβλέπων καθηγητή. Σε περίπτωση που μια εργασία κριθεί ελλιπής, παραπέμπεται προς συμπληρωματική επεξεργασία από τον φοιτητή και επανεξετάζεται εκ νέου σε διαφορετική ημερομηνία, κατόπιν συνεννόησης. Τέλος, όλες οι Πτυχιακές/Διπλωματικές εργασίες κατατίθενται στο Ιδρυματικό Αποθετήριο του Πανεπιστημίου Δυτικής Αττικής.

Η ανωτέρω διαδικασία αποτυπώνεται συνοπτικά στο παρακάτω διάγραμμα ροής:



Διάγραμμα 1.1 - Διάγραμμα Ροής διαδικασίας Πτυχιακών/Διπλωματικών εργασιών

1.3. Διαπίστωση προβλημάτων

Είναι γεγονός πως το Πανεπιστήμιο Δυτικής Αττικής έχει εξελίξει σε μεγάλο βαθμό τα συστήματα και τις υποδομές του τα τελευταία χρόνια, ιδιαίτερα για θέματα που αφορούν την καθημερινότητα των φοιτητών και του εκπαιδευτικού προσωπικού του. Ωστόσο, ύστερα από συζητήσεις τόσο με τους φοιτητές όσο και τους καθηγητές του Τμήματος, διαπιστώθηκαν ορισμένα προβλήματα στη διαδικασία που ακολουθείται για τις Πτυχιακές και Διπλωματικές εργασίες.

Αμφότερες οι πλευρές συμφωνούν στο γεγονός ότι υπάρχει αρκετή γραφειοκρατία από την αρχή ως το τέλος της διαδικασίας που ακολουθείται και ότι χρήζει βελτίωσης και καλύτερης οργάνωσης. Σε αντίθεση για παράδειγμα με τη διαδικασία δήλωσης μαθημάτων που γίνεται ηλεκτρονικά, όλη η διαδικασία που αφορά τις Πτυχιακές/Διπλωματικές εργασίες γίνεται με τη χειρόγραφη συμπλήρωση διαφορετικών εντύπων (Αίτηση Ανάληψης Πτυχιακής/Διπλωματικής, Αίτηση Εξέτασης, Πρακτικό Αξιολόγησης κλπ) τα οποία πρέπει να υπογραφούν από όλους τους εμπλεκόμενους και στη συνέχεια κατατεθούν στην

Γραμματεία του Τμήματος, ώστε να πρωτοκολληθούν και να αρχειοθετηθούν. Η απλοποίηση της συνολικής διαδικασίας μέσω της μηχανοργάνωσης και της ψηφιοποίησης, θα μπορούσε να μειώσει σημαντικά τη γραφειοκρατία και ταυτόχρονα να βελτιώσει τη καθημερινότητα των φοιτητών και να μειώσει το φόρτο εργασίας του διοικητικού προσωπικού του Τμήματος.

Ένα συχνό πρόβλημα που αντιμετωπίζουν οι φοιτητές είναι πως η λίστα με τα προτεινόμενα θέματα Πτυχιακών/Διπλωματικών εργασιών που αναρτάται στην αρχή του κάθε εξαμήνου και στη συνέχεια δεν ενημερώνεται. Αυτό έχει ως αποτέλεσμα πολλοί φοιτητές να δηλώνουν ενδιαφέρον για θέματα που έχουν ήδη ανατεθεί, και να αναλώνονται σε άσκοπες επικοινωνίες με διαφορετικούς καθηγητές, προκειμένου να αναλάβουν ένα θέμα. Πρόβλημα επίσης αποτελεί η έλλειψη ενημέρωσης σχετικά με την πορεία εξέλιξης των αιτήσεων τους ή τη προθεσμία ολοκλήρωσης της εργασίας τους. Επιπλέον, επισήμαναν πως δεν υπάρχουν κάπου συγκεντρωμένα και εύκολα προσβάσιμα τα απαραίτητα έγγραφα που χρειάζεται να συμπληρώνουν και να καταθέτουν καθ' όλη τη διάρκεια της διαδικασίας.

Από τη πλευρά τους οι καθηγητές, ειδικά εκείνοι που αναλαμβάνουν την επίβλεψη πολλαπλών διαφορετικών θεμάτων, δεν διαθέτουν κάποιο ενιαίο τρόπο διαχείρισης και παρακολούθησης των Πτυχιακών/Διπλωματικών εργασιών για τις οποίες είναι υπεύθυνοι. Συγκεκριμένα, οι καθηγητές δεν μπορούν να επεξεργαστούν με άμεσο τρόπο τα αναρτημένα θέματα τους, όταν για παράδειγμα επιθυμούν να τροποποιήσουν την περιγραφή ενός θέματος ή την προτεινόμενη βιβλιογραφία, χωρίς να στείλουν σχετική ενημέρωση προς τη Γραμματεία του Τμήματος. Αντίστοιχα, δεν μπορούν να επισημάνουν ένα θέμα ως μη διαθέσιμο εφόσον έχει ήδη ανατεθεί σε κάποιον φοιτητή. Επιπρόσθετα, οι καθηγητές δεν έχουν τη δυνατότητα να παρακολουθούν και να ενημερώνονται ανά πάσα στιγμή τη πορεία εξέλιξης των ανατεθειμένων Πτυχιακών/Διπλωματικών εργασιών που επιβλέπουν, ώστε να τις οργανώνουν καλύτερα. Ένα ακόμα πρόβλημα που διαπιστώθηκε, είναι πως δεν είναι ξεκάθαρη η ιστορικότητα ενός θέματος, αν για παράδειγμα ανατέθηκε και δεν ολοκληρώθηκε στον προβλεπόμενο χρόνο ή ακυρώθηκε ύστερα από αίτημα του φοιτητή.

1.4. Προτεινόμενη Λύση

Με αφορμή τα προβλήματα που αναφέρθηκαν, γεννήθηκε η ανάγκη για την σχεδίαση και ανάπτυξη ενός πληροφοριακού συστήματος υποδομής με στόχο την απλοποίηση και βελτίωσης της διαδικασίας που αφορά τις Πτυχιακές και Διπλωματικές εργασίες των φοιτητών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών.

Το πληροφοριακό σύστημα αρχικά θα πρέπει να διαθέτει μια κεντρική Βάση Δεδομένων, στην οποία θα καταχωρούνται όλα τα θέματα Πτυχιακών και Διπλωματικών εργασιών, με σκοπό να δημιουργηθεί ένας επίσημος Πίνακας θεμάτων, για να μπορούν οι ενδιαφερόμενοι φοιτητές ανά πάσα στιγμή να αναζητούν και να ενημερώνονται με όλα τα διαθέσιμα θέματα. Στο Πίνακα Θεμάτων θα επιτρέπεται η επισκόπηση του κάθε θέματος, περιέχοντας όλες τις απαραίτητες πληροφορίες όπως τη περιγραφή, την ενδεικτική βιβλιογραφία, αν

προτείνεται για ομαδική ή ατομική εργασία καθώς και ποιον κύκλο σπουδών αφορά. Οι καθηγητές θα πρέπει να έχουν τη δυνατότητα να δημιουργούν νέα θέματα, να τα τροποποιούν και φυσικά να τα αναθέτουν στους ενδιαφερόμενους φοιτητές, κατόπιν συνεννόησης. Όταν ένα θέμα ανατίθεται σε έναν υποψήφιο φοιτητή, θα πρέπει αυτόματα να δηλώνεται ως μη διαθέσιμο, ώστε να ενημερώνονται οι ενδιαφερόμενοι φοιτητές και να μην χρειάζεται να έρθουν σε επικοινωνία με τον υπεύθυνο καθηγητή.

Προκειμένου να απλοποιηθεί η διαδικασία ανάληψης μίας Πτυχιακής ή Διπλωματικής εργασίας από έναν φοιτητή, προτείνεται όλες οι Αιτήσεις Ανάληψης Πτυχιακής/Διπλωματικής να συμπληρώνονται και να καταχωρούνται ηλεκτρονικά στη Βάση Δεδομένων, με δυνατότητα εκτύπωσης τους από το σύστημα. Αντίστοιχα, προτείνεται να καταχωρούνται ηλεκτρονικά και οι Αιτήσεις Εξέτασης από τους φοιτητές, εφόσον έχει η εγκριθεί η εξέταση τους από τον επιβλέποντα καθηγητή. Επιπλέον, προτείνεται το πληροφοριακό σύστημα να υποστηρίζει την ηλεκτρονική συμπλήρωση και εκτύπωση του Πρακτικού Αξιολόγησης κατά την διαδικασία εξέτασης από τους καθηγητές. Με την ηλεκτρονική υποβολή των ανωτέρω εγγράφων στο σύστημα, επιτυγχάνεται καλύτερη οργάνωση, μείωση της γραφειοκρατίας και του φόρτου των εμπλεκομένων, ενώ ταυτόχρονα απλοποιείται σημαντικά η διαδικασία. Παράλληλα, όλα τα προσωπικά στοιχεία και οι πληροφορίες των φοιτητών αποθηκεύονται με ασφαλή τρόπο σε μια κεντρική Βάση Δεδομένων, αντί σε χάρτινα έγγραφα.

Για τη διευκόλυνση των καθηγητών, πέραν της δημιουργίας και διαχείρισης των προσωπικών τους θεμάτων, θα πρέπει να επιτρέπεται η πλήρη διαχείριση των ανατεθειμένων Πτυχιακών/Διπλωματικών εργασιών για τις οποίες είναι υπεύθυνοι. Συγκεκριμένα, προτείνεται εκτός από την ηλεκτρονική ανάθεση θέματος σε υποψήφιο φοιτητή, να μπορούν να ορίζουν διαφορετικές καταστάσεις σε κάθε ανάθεση για καλύτερη οργάνωση και παρακολούθηση τους, για παράδειγμα θέτουν μια ανάθεση σε κατάσταση «Εξέλιξης» αν βρίσκεται σε εκπόνηση από τον φοιτητή ή «Προς Εξέταση» όταν έχει ολοκληρωθεί και ο καθηγητής εγκρίνει την εξέταση της. Οι αλλαγές κατάστασης θα καταγράφονται στη Βάση Δεδομένων, ώστε να υπάρχει πλήρη ιστορικότητα για τη πορεία εξέλιξης της εργασίας. Επίσης, προτείνεται η δυνατότητα ορισμού της Τριμελούς Εξεταστικής Επιτροπής ή και του ορισμού της ημερομηνίας εξέτασης και παρουσίασης της μιας Πτυχιακής/Διπλωματικής εργασίας στο σύστημα από το καθηγητή, καθώς και τη δυνατότητα παράτασης της προθεσμίας ολοκλήρωσης, με στόχο τη πλήρη διαχείριση των ανατεθειμένων εργασιών μέσω του πληροφοριακού συστήματος. Τέλος, το σύστημα θα επιτρέπει ερωτήματα προς την Βάση Δεδομένων από τους φοιτητές, σχετικά με την κατάσταση της ανάθεσης τους, την προθεσμία και όλη την σχετική πληροφορία που αφορά της Πτυχιακή/Διπλωματική τους εργασία.

Η υλοποίηση της ανωτέρω προτεινόμενης λύσης, στοχεύει στην επίλυση των προβλημάτων που διαπιστώθηκαν από την ανάλυση της υφιστάμενης κατάστασης, στην απλοποίηση της συνολικής διαδικασίας που ακολουθείται για τις Πτυχιακές/ Διπλωματικές εργασίες αλλά και στη βελτίωση της καθημερινότητας των φοιτητών και των καθηγητών του Τμήματος.

Κεφάλαιο 2 – Θεωρητικό Υπόβαθρο

Σε αυτό το κεφάλαιο πραγματοποιείται μια γενική ανασκόπηση των βασικών εννοιών και τεχνολογιών στις οποίες βασίζεται ο σχεδιασμός του συστήματος που υλοποιήθηκε. Αρχικά, γίνεται μια σύντομη ιστορική αναδρομή στη κύρια γλώσσα προγραμματισμού που χρησιμοποιήθηκε, την C#, καθώς και μια αναφορά των κύριων χαρακτηριστικών της. Στη συνέχεια, περιγράφεται η αρχιτεκτονική της πλατφόρμας ανάπτυξης .NET και των επιμέρους στοιχείων από τα οποία αποτελείται, πως εξελίχθηκε ανά τα χρόνιά μέσα από τις διάφορες εκδόσεις που έχουν κυκλοφορήσει, ενώ επισημαίνονται τα κύρια πλεονεκτήματα που προσφέρει. Τέλος, αναφέρονται βασικές έννοιες που αφορούν τις σχεσιακές βάσεις δεδομένων, τα Web Services και τα APIs.

2.1. Η γλώσσα προγραμματισμού C#

Η C# (προφέρεται ως C Sharp) είναι μια σύγχρονη και ευέλικτη αντικειμενοστρεφής γλώσσα προγραμματισμού η οποία έχει αποκτήσει σημαντική δημοτικότητα μεταξύ των προγραμματιστών τα τελευταία χρόνια.

2.1.1. Ιστορική αναδρομή

Η C# αναπτύχθηκε από την Microsoft και εμφανίστηκε πρώτη φορά το 2000. Η γλώσσα σχεδιάστηκε από μία ομάδα με επικεφαλής τον Anders Hejlsberg [2], ο οποίος είναι γνωστός για τη δουλειά του πάνω στον μεταγλωττιστή (compiler) Turbo Pascal και τη γλώσσα προγραμματισμού Delphi. Ο Hejlsberg είχε ως στόχο να δημιουργήσει μια γλώσσα προγραμματισμού που θα συνδύαζε την απλότητα και την εξοικείωση της C++ με την παραγωγικότητα και τη ισχύ της πλατφόρμας .NET της Microsoft.

Η C# είναι εμπνευσμένη από διάφορες γλώσσες προγραμματισμού, όπως η C, η C++ και η Java. Υιοθέτησε τη σύνταξη και τις δομές (constructs) της C και της C++, διευκολύνοντας έτσι τους προγραμματιστές που ήταν ήδη εξοικειωμένοι με αυτές τις γλώσσες. Επιπλέον, διαθέτει δυνατότητες και χαρακτηριστικά της Java για να επωφεληθεί από την ασφάλεια και την ανθεκτικότητα του Java Virtual Machine (JVM).

Αρχικά σχεδιάστηκε ως αναπόσπαστο μέρος του .NET Framework, της επαναστατικής πλατφόρμας της Microsoft για ανάπτυξη και εκτέλεση εφαρμογών λογισμικού. Η συγκεκριμένη πλατφόρμα διαθέτει ένα κοινό περιβάλλον εκτέλεσης (common runtime environment) που επιτρέπει στα προγράμματα γραμμένα σε C# να εκτελούνται απρόσκοπτα σε διαφορετικές πλατφόρμες, αλλά και να αξιοποιούν τις τεράστιες δυνατότητες και τη πληθώρα βιβλιοθηκών που παρέχονται στο οικοσύστημα του .NET [3]. Με τη πάροδο του χρόνου, έχουν κυκλοφορήσει διάφορες εκδόσεις της C# για να καλύψει τις συνεχώς εξελισσόμενες ανάγκες των προγραμματιστών. Κάθε νέα έκδοση εισήγαγε σημαντικές νέες δυνατότητες και βελτιώσεις ως προς την απόδοση και

τη παραγωγικότητα. Η πιο πρόσφατη και σταθερή έκδοση της C# μέχρι και σήμερα είναι η έκδοση 11 [4].

Τα τελευταία χρόνια, η Microsoft έχει κάνει σημαντικά βήματα για να φέρει την C# πέρα από το οικοσύστημα των Windows. Με την κυκλοφορία του .NET Core, μιας cross-platform έκδοσης ανοιχτού κώδικα (open-source) του .NET Framework, οι προγραμματιστές που χρησιμοποιούν την γλώσσα απέκτησαν την δυνατότητα να δημιουργούν εφαρμογές που τρέχουν σε διαφορετικές πλατφόρμες. Η δυνατότητα αυτή αύξησε τη δημοτικότητα της C# και ανέδειξε την ευελιξία της. Σήμερα, η C# έχει εδραιωθεί ως μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού στο κλάδο της Πληροφορικής. [5]

2.1.2. Κύρια χαρακτηριστικά και πλεονεκτήματα

Τα κύρια χαρακτηριστικά της γλώσσας C# είναι τα εξής [2] [3]:

1. **Ευκολία**: η C# σχεδιάστηκε να είναι φιλική προς τους αρχάριους, με μια σαφή και ευανάγνωστη σύνταξη. Η σύνταξή της είναι παρόμοια με άλλες δημοφιλείς γλώσσες προγραμματισμού όπως ο C++ και ο Java, καθιστώντας ευκολότερη τη μετάβαση μεταξύ διαφορετικών γλωσσών. Η γλώσσα παρέχει μια ισχυρή βάση για την κατανόηση θεμελιωδών εννοιών και αρχών προγραμματισμού.
2. **Αντικειμενοστρεφής Προγραμματισμός**: η C# ακολουθεί ένα ισχυρό πρότυπο αντικειμενοστρεφή προγραμματισμού, που εφαρμόζει έννοιες όπως η ενθυλάκωση, η κληρονομικότητα και ο πολυμορφισμός, επιτρέποντας στους προγραμματιστές να δημιουργούν επεκτάσιμο και επαναχρησιμοποιούμενο κώδικα.
3. **Ισχυρό σύστημα τύπων (strong typing) και ασφάλεια**: η γλώσσα ορίζει έναν ή περισσότερους περιορισμούς στον τρόπο που συνδυάζονται οι λειτουργίες που χρησιμοποιούν τιμές από διαφορετικούς τύπους δεδομένων. Παράλληλα εφαρμόζει στατικό έλεγχο τύπων, που σημαίνει πως ο κάθε τύπος μεταβλητής καθορίζεται πριν από την εκτέλεση του προγράμματος, επιτυγχάνοντας έτσι καλύτερη ασφάλεια και έγκαιρη ανίχνευση σφαλμάτων, με αποτέλεσμα πιο αξιόπιστο και συντηρήσιμο κώδικα.
4. **Παραγωγικότητα**: η γλώσσα παρέχει ένα ευρύ φάσμα έτοιμων βιβλιοθηκών και δυνατοτήτων όπως για παράδειγμα εκφράσεων Λάμδα (Lamda Expressions) για τη δημιουργία ανώνυμων συναρτήσεων, σύνταξη LINQ (Language Integrated Query) που δημιουργεί ένα κοινό μοτίβο εργασίας πάνω σε δεδομένα διαφορετικού τύπου, δημιουργία ασύγχρονων μεθόδων με τη χρήση των keywords async/await, αυτόματη διαχείριση της μνήμης από μέσω του Garbage Collector και πολλά άλλα. Τέτοιες δυνατότητες απλοποιούν την ανάπτυξη εφαρμογών και επιτρέπουν στους προγραμματιστές να γράφουν κατανοητό και συνοπτικό κώδικα.

5. **Εκτενής Τεκμηρίωση:** η Microsoft παρέχει πλήρης και εκτενής τεκμηρίωση για τη C# όπως και για όλες τις τεχνολογίες που αφορούν το .NET μέσω του διαδικτύου. Επίσης προσφέρει δωρεάν εκπαιδευτικό υλικό, διαδραστικά μαθήματα, παραδείγματα χρήσης καθώς και αναλυτικά άρθρα με που εξηγούν εις βάθος τις πιο περίπλοκες έννοιες και τεχνικές.
6. **Μεγάλη Κοινότητα:** όντας προϊόν ενός τεχνολογικού κολοσσού όπως η Microsoft και με τη συνεχή υποστήριξη και αναβαθμίσεις που επιδέχεται, η C# επιλέγεται από πάρα πολλούς προγραμματιστές, με αποτέλεσμα να έχει σχηματιστεί μια μεγάλη και ενεργή κοινότητα. Στο τομέα της πληροφορικής και ειδικά στην ανάπτυξη λογισμικού, είναι πολύ σημαντικό να υπάρχει μια κοινότητα, καθώς μέσω αυτής οι προγραμματιστές μπορούν να ανταλλάσσουν απόψεις, ιδέες και να συμβουλεύουν ο ένας τον άλλο.

2.1.3. Πρακτικές εφαρμογές της C#

Η C# είναι ευρέως διαδεδομένη σε διάφορους τομείς λόγω της αποτελεσματικότητα της σε διαφορετικούς τύπους εφαρμογών [5] όπως για παράδειγμα:

- **Desktop Εφαρμογές:** χρησιμοποιώντας τεχνολογίες όπως το WPF (Windows Presentation Foundation) και τα Windows Forms, η C# ενδείκνυται για την ανάπτυξη δυναμικών και διαδραστικών desktop εφαρμογών.
- **Web Εφαρμογές:** η C# μπορεί να χρησιμοποιηθεί για την ανάπτυξη server-side δικτυακών εφαρμογών χάρις στις τεχνολογίες ASP.NET και ASP.NET Core, που επιτρέπουν στους προγραμματιστές να χτίζουν εύκολα web services, RESTFul APIs και web εφαρμογές που τρέχουν σε πραγματικό χρόνο. Μπορεί επίσης να χρησιμοποιηθεί για την ανάπτυξη client-side web εφαρμογών με την χρήση της τεχνολογίας Blazor, μιας πλατφόρμας ανοιχτού κώδικα ανάπτυξης web εφαρμογών με τη χρήση C# και HTML/CSS.
- **Mobile Εφαρμογές:** χάρις στη καινοτόμα τεχνολογία Xamarin, οι προγραμματιστές μπορούν να αναπτύσσουν εγγενείς mobile εφαρμογές για τις πλατφόρμες Android και iOS, γράφοντας αποκλειστικά σε C#, καθιστώντας έτσι ευκολότερο τον διαμοιρασμό κώδικα μεταξύ των διαφορετικών πλατφορμών.
- **Ανάπτυξη Παιχνιδιών:** η C# αποτελεί δημοφιλής επιλογή για την ανάπτυξη παιχνιδιών και ο λόγος είναι η μηχανή ανάπτυξης παιχνιδιών Unity, η οποία αξιοποιεί τη C# ως τη κύρια γλώσσα της και παρέχει όλα τα απαραίτητα εργαλεία για τη δημιουργία καθηλωτικών και σύγχρονων παιχνιδιών που μπορούν να τρέξουν σε οποιαδήποτε πλατφόρμα.

2.2. Η πλατφόρμα ανάπτυξης .NET

Το .NET (προφέρεται ως Dot NET) είναι μια πλατφόρμα ανάπτυξης που έχει αναπτυχθεί από τη Microsoft και αποτελείται από διάφορα εργαλεία, γλώσσες προγραμματισμού και βιβλιοθήκες για την δημιουργία εφαρμογών διαφορετικού τύπου [6]. Υπάρχουν διάφορες

υλοποιήσεις του .NET. Κάθε υλοποίηση επιτρέπει στον κώδικα .NET να εκτελείται σε διαφορετικό λειτουργικό σύστημα, όπως για παράδειγμα σε Linux, macOS, Windows, iOS, Android και πολλά άλλα. Οι κύριες υλοποιήσεις είναι οι εξής:

- **.NET Framework -** αποτελεί την αρχική υλοποίηση του .NET και υποστηρίζει την δημιουργία και εκτέλεση ιστοσελίδων, διεργασιών, desktop εφαρμογών και άλλων σε περιβάλλον Windows.
- **.NET** ή .**NET Core -** αποτελεί τη νέας γενιάς υλοποίηση .NET και είναι cross-platform, δηλαδή υποστηρίζει την δημιουργία και εκτέλεση εφαρμογών και ιστοσελίδων σε λειτουργικό Windows, Linux αλλά και macOS. Η υλοποίηση αυτή είναι ανοιχτού κώδικα (open source).
- **Xamarin/Mono** - η υλοποίηση αυτή επιτρέπει την δημιουργία και εκτέλεση mobile εφαρμογών σε όλα τα μεγάλα λειτουργικά συστήματα για κινητά, συμπεριλαμβανομένων των iOS και Android.
- **.NET Standard** - είναι μια επίσημη προδιαγραφή των API (Application Programming Interfaces) που είναι κοινά μεταξύ των διάφορων υλοποιήσεων του .NET, επιτρέποντας έτσι την επαναχρησιμοποίηση του ίδιου κώδικά και βιβλιοθηκών σε διαφορετικού τύπου εφαρμογές.

2.2.1. Αρχιτεκτονική του .NET

Το .NET αφορά ένα οικοσύστημα που αποτελείται από διάφορα κύρια στοιχεία (components) που συνεργάζονται για να διευκολύνουν την ανάπτυξη εφαρμογών [6]. Η αρχιτεκτονική έχει σχεδιαστεί έτσι ώστε να προωθεί την επαναχρησιμοποίηση του κώδικα, τη συντηρησιμότητα και την αλληλεπίδραση μεταξύ διαφορετικών γλωσσών προγραμματισμού και πλατφορμών. Συγκεκριμένα, αποτελείται από τα εξής components:

Κοινή Γλώσσα Εκτέλεσης (Common Language Runtime - CLR)

Το Common Language Runtime (CLR) αποτελεί ένα κρίσιμο στοιχείο της αρχιτεκτονικής του .NET [7]. Λειτουργεί ως περιβάλλον εκτέλεσης για τις εφαρμογές που έχουν δημιουργηθεί σε .NET. Το CLR πραγματοποιεί αυτόματη διαχείριση μνήμης (συλλογή αχρησιμοποίητων αντικειμένων – garbage collection) και χειρίζεται τα σφάλματα ή τυχόν εξαιρέσεις. Παρέχει επίσης διάφορες υπηρεσίες, όπως ασφάλεια, επαλήθευση τύπων και μεταγλώττιση κατά τη διάρκεια της εκτέλεσης.

Βασική Βιβλιοθήκη Κλάσεων (Base Class Library - BCL)

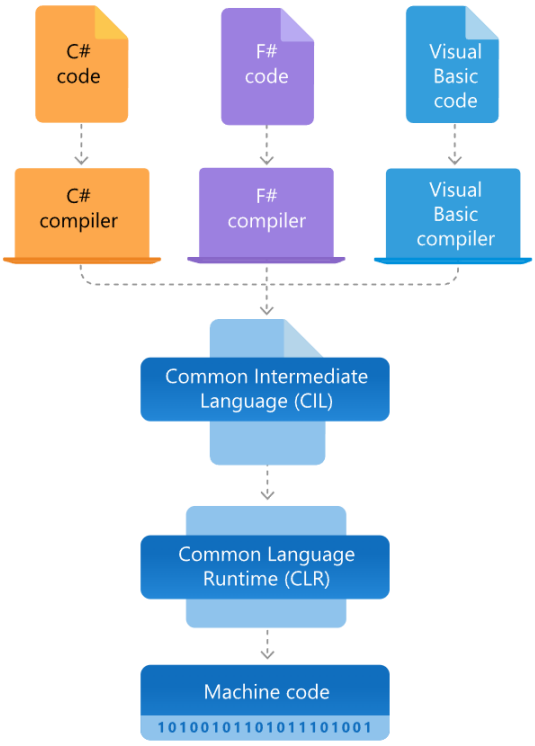
Η Βασική Βιβλιοθήκη Κλάσεων (BCL) αποτελεί μια συλλογή έτοιμων κλάσεων, τύπων και διεπαφών που παρέχονται από το .NET Framework [7]. Προσφέρει μια ευρεία γκάμα λειτουργιών, όπως αρχεία εισόδου/εξόδου, δικτυακή επικοινωνία, πρόσβαση σε δεδομένα και στοιχεία διεπαφής χρήστη. Η BCL απλοποιεί την ανάπτυξη παρέχοντας έτοιμες κλάσεις και μεθόδους, μειώνοντας τον κώδικα που απαιτείται να γραφεί από τους προγραμματιστές.

Κοινό Σύστημα Τύπων (Common Type System - CTS)

Το Κοινό Σύστημα Τύπων (CTS) καθορίζει ένα πρότυπο σύνολο τύπων δεδομένων και κανόνων που διασφαλίζουν την αλληλεπίδραση μεταξύ διαφορετικών γλωσσών προγραμματισμού του .NET [7]. Εξασφαλίζει ότι τα αντικείμενα που έχουν γραφεί σε μία γλώσσα μπορούν να χρησιμοποιηθούν από μία άλλη χωρίς προβλήματα, επιτρέποντας στους προγραμματιστές να συνεργάζονται χρησιμοποιώντας την προτιμώμενη τους γλώσσα προγραμματισμού.

Μεταγλώττιση Αμέσου Χρόνου (Just-In-Time - JIT)

Η αρχιτεκτονική του .NET χρησιμοποιεί τη μεταγλώττιση αμέσου χρόνου (JIT) για να μετατρέψει τον ενδιάμεσο κώδικα (Common Intermediate Language - CIL) σε γλώσσα μηχανής που μπορεί να εκτελεστεί από τον επεξεργαστή του υπολογιστή [7]. Η μεταγλώττιση JIT πραγματοποιείται κατά τη διάρκεια της εκτέλεσης του προγράμματος και βελτιστοποιεί τον κώδικα με βάση τους διαθέσιμους πόρους του συστήματος από το οποίο εκτελείται, βελτιώνοντας έτσι την απόδοση.



Εικόνα 2.2 – Αρχιτεκτονική του .NET

2.2.2. Ιστορικό Εκδόσεων

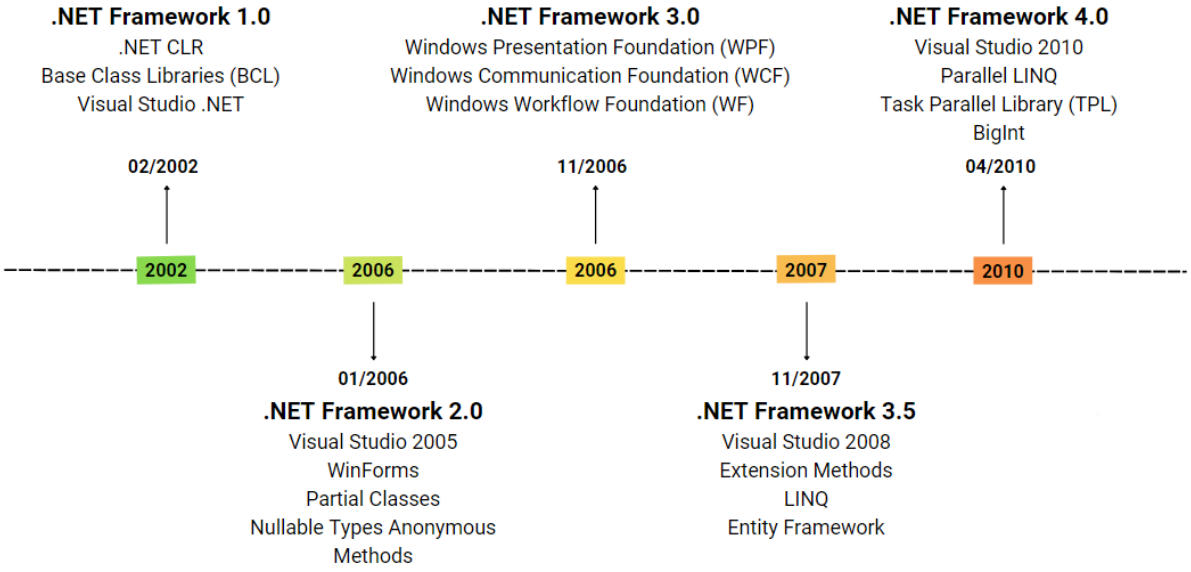
Το .NET υπάρχει για πάνω από μια εικοσαετία και ως εκ τούτου έχουν κυκλοφορήσει πολλές διαφορετικές εκδόσεις και ενημερώσεις. Οι κυριότερες εξ αυτών, είναι φυσικά το .NET Framework που εμφανίστηκε για πρώτη φορά το 2002 [8], ακολουθεί το .NET Core

που κυκλοφόρησε το 2016 και εν τέλει καθιερώθηκε ως .NET στην 5η έκδοση του Framework που κυκλοφόρησε το 2020 [9].

Εικόνα 2.3 – Επισκόπηση των εκδόσεων του .NET

Ωστόσο, έχει είναι ενδιαφέρον να παρακολουθήσει κάνεις πως εξελίχτηκε η πλατφόρμα με τη πάροδο του χρόνου και να κατανοήσει που οφείλονται οι συχνές μετονομάσεις αλλά και τις διαφορές μεταξύ των εκδόσεων.

Εκδόσεις .NET Framework



Εικόνα 2.4 – Αρχικές εκδόσεις του .NET Framework

Έκδοση 1.0

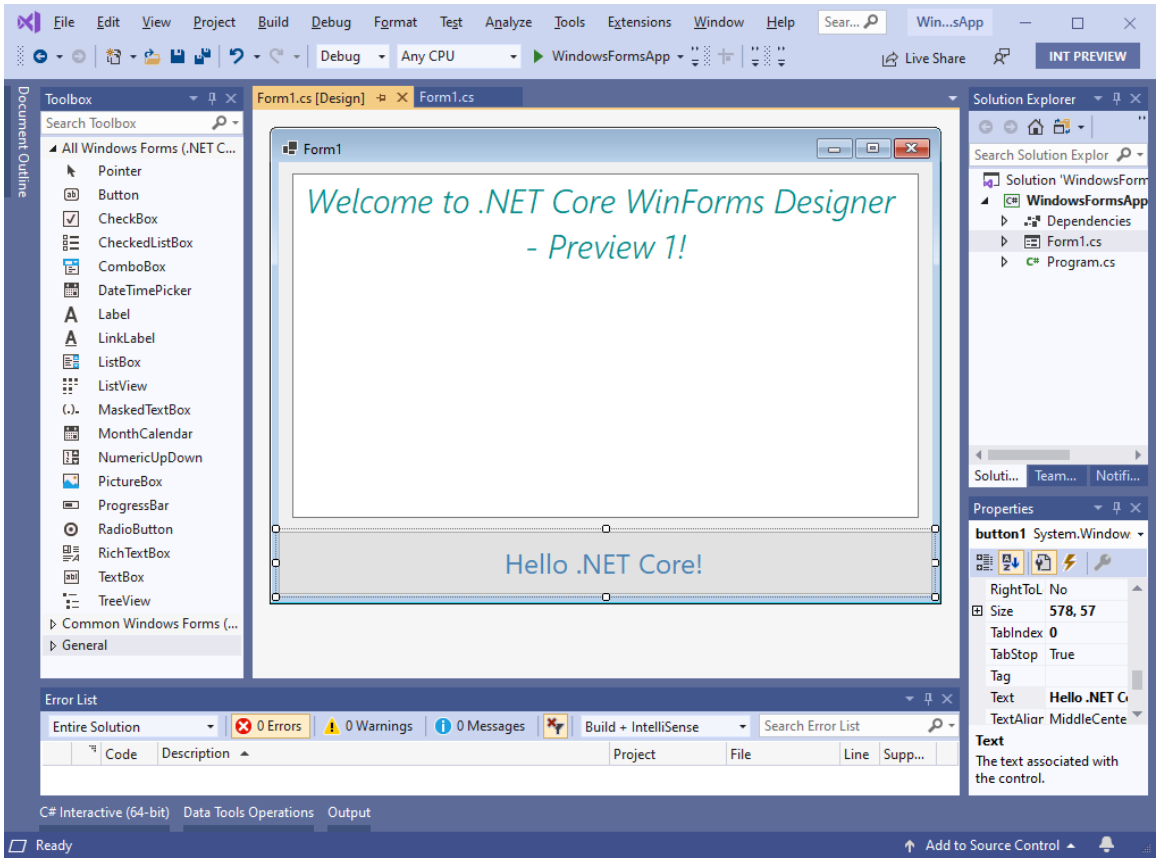
Όπως έχει ήδη αναφερθεί, όλα ξεκίνησαν με την πρωταρχική έκδοση του .NET Framework που κυκλοφόρησε τον Φλεβάρη του 2002. Η πλατφόρμα τότε αφορούσε αποκλειστικά συστήματα με λογισμικό Windows και συγκεκριμένα για τις εκδόσεις Windows 98, Window ME (Millenium Edition) και Windows XP. Η έκδοση 1.0 εισήγαγε το CLR, τη BCL, το ASP.NET αλλά και τη πρώτη έκδοση του Visual Studio [8].

Έκδοση 2.0

Η δεύτερη έκδοση του .NET Framework κυκλοφόρησε ταυτόχρονα με το Visual Studio 2005 τον Γενάρη του 2006 [8]. Εισήγαγε νέες δυνατότητες για την γλώσσα C#, πιο συγκεκριμένα:

- Τμηματικές κλάσεις
- Nullable τύπους μεταβλητών
- Ανώνυμες μεθόδους
- Πίνακες δεδομένων ή DataTables

Ωστόσο, η μεγαλύτερη προσθήκη ήταν η εισαγωγή των Windows Forms (ή **WinForms**) [10]. Τα WinForms είναι μια βιβλιοθήκη κλάσεων γραφικού περιβάλλοντος χρήστη (Graphical User Interface – GUI) που παρέχει μια ευκολότερη διεπαφή για τη ανάπτυξη desktop εφαρμογών. Διαθέτει μια πλήρης εργαλειοθήκη με όλα τα απαραίτητα στοιχεία όπως buttons, labels, text boxes, lists κλπ τα οποία είναι πλήρως προγραμματιζόμενα και παραμετροποιήσιμα καθώς και ένα περιβάλλον σχεδίασης όπου ο χρήστης μπορεί να τα τοποθετεί εύκολα με τη χρήση του ποντικιού.



Εικόνα 2.5 – Ανάπτυξη εφαρμογής WinForms μέσω Visual Studio

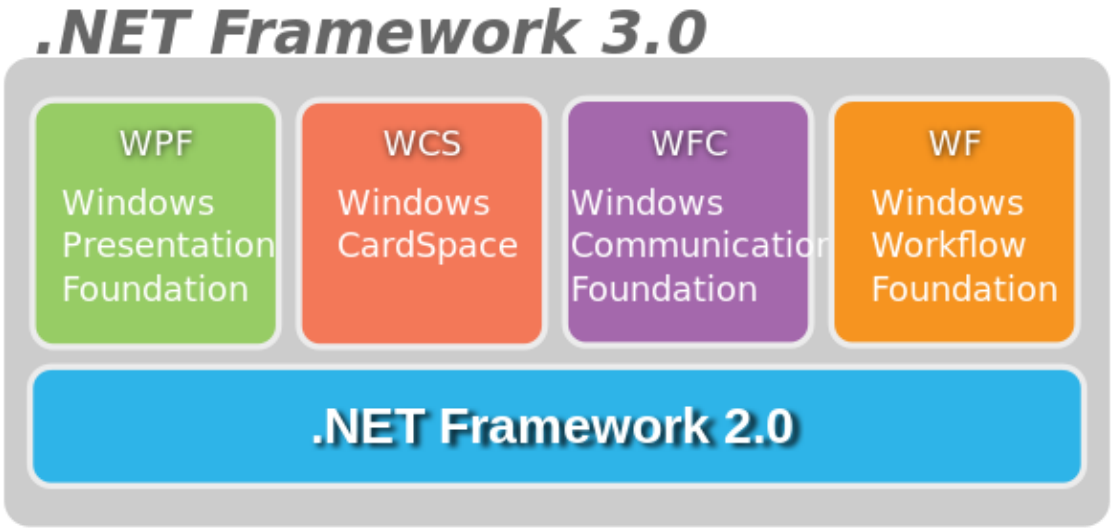
Έκδοση 3.0

Η έκδοση 3.0 κυκλοφόρησε στις 5 Νοεμβρίου του 2006 και εισήγαγε σημαντικές τεχνολογίες όπως το Windows Presentation Foundation (**WPF**), το Windows Communication Foundation (**WCF**) και το Windows Workflow Foundation (**WWF**) [8].

Το WPF είναι μία πιο μοντέρνα έκδοση των WinForms, που χρησιμοποιεί μια μηχανή απόδοσης γραφικών που βασίζεται σε διανύσματα (vectors), προσφέροντας έτσι πιο ελκυστικό και σύγχρονο UI, σε αντίθεση με το WinForms που χρησιμοποιεί μια πιο παραδοσιακή προσέγγιση, που βασίζεται σε φόρμες και στοιχεία ελέγχου [10].

Όσον αφορά το WWF, πρόκειται για ένα framework δημιουργίας και διαχείρισης ροών εργασίας ή workflows. Ένα workflow είναι μια σειρά από προγραμματιστικά βήματα που εκτελούν μια διαδικασία. Κάθε βήμα μιας διαδικασίας ορίζεται ως μια δραστηριότητα (ή activity). Το WWF προσφέρει μια βιβλιοθήκη από activities, ενώ υπάρχει η δυνατότητα της ανάπτυξης προσαρμοσμένων δραστηριοτήτων για πρόσθετη λειτουργικότητα. Μέσω του εργαλείου Workflow Designer του Visual Studio, επιτυγχάνεται η σχεδίαση και οπτικοποίηση των ροών εργασίας [11].

Τέλος, το WCF είναι ένα framework ανάπτυξης δικτυακών υπηρεσιών (web services) [12]. Μέσω του WCF, μπορεί να γίνει αποστολή δεδομένων ως ασύγχρονα μηνύματα από ένα endpoint σε ένα άλλο.



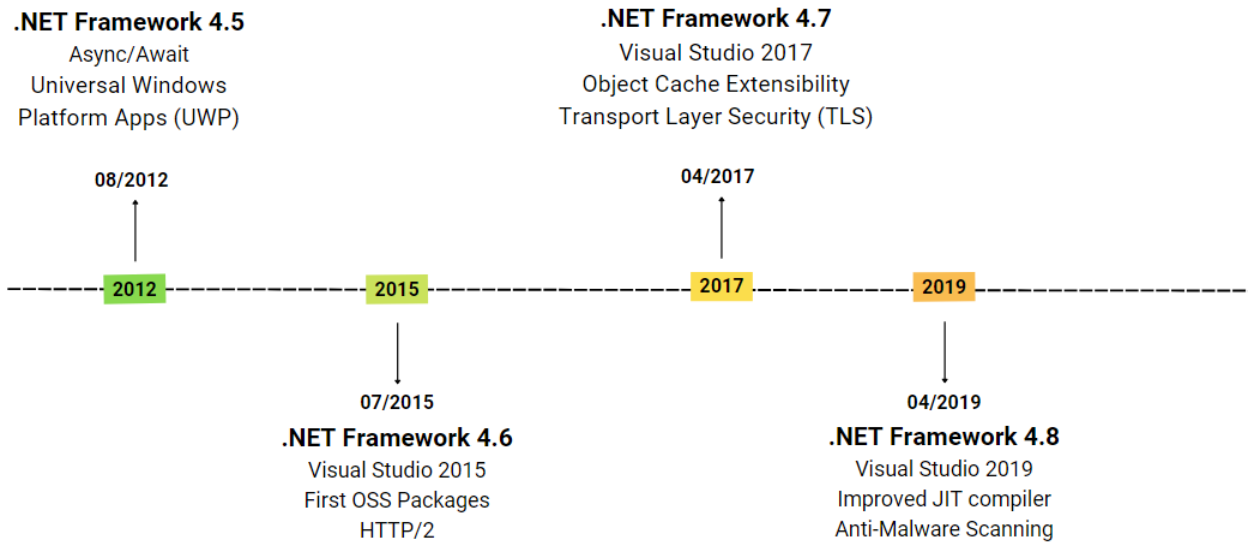
Εικόνα 2.6 – Η έκδοση 3.0 του .NET Framework

Έκδοση 3.5

Στις 19 Νοεμβρίου του 2007, κυκλοφόρησε η έκδοση 3.5 και έφερε σημαντικές βελτιώσεις και νέες δυνατότητες για υπάρχουσες τεχνολογίες όπως το WPF και το WCF [8]. Ωστόσο, οι πιο αξιοσημείωτες προσθήκες ήταν η έκδοση 2008 του Visual Studio και η εισαγωγή των Extension Methods αλλά και των τεχνολογιών **LINQ** και του **Entity Framework.**

Έκδοση 4.0

Το .NET Framework 4.0 κυκλοφόρησε στις 12 Απριλίου του 2010. Ήταν μια έκδοση που έφερε μεγάλες βελτιώσεις και εμπλουτισμούς και κυκλοφόρησε παράλληλα με το Visual Studio 2010 [8]. Επιπρόσθετα, εισήγαγε τη Parallel LINQ και τη βιβλιοθήκη παράλληλων εργασιών (Task Parallel Library - TPL) που ενίσχυσαν σημαντικά τις δυνατότητες παράλληλου προγραμματισμού του framework. Τέλος, προστέθηκε και ο τύπος δεδομένων BigInt που χρησιμοποιείται για την αρχικοποίηση αυθαίρετα μεγάλων προσημασμένων ακέραιων αριθμών.



Εικόνα 2.7 – Τελευταίες εκδόσεις του .NET Framework

Έκδοση 4.5

Η έκδοση 4.5 του .NET Framework κυκλοφόρησε στις 15 Αυγούστου του 2012 μαζί με το Visual Studio 2012 [8]. Έφερε την ενσωματωμένη υποστήριξη για των Async/Await keywords για τη διαχείριση ασύγχρονων λειτουργιών. Επίσης, εισήγαγε τα Universal Windows Apps (UWP), που επέτρεπε την ανάπτυξη ενιαίων εφαρμογών που τρέχουν σε διαφορετικές συσκευές χωρίς να χρειάζεται κάποια αναπροσαρμογή του κώδικα.

Έκδοση 4.6

Το .NET Framework 4.6 κυκλοφόρησε στις 20 Ιουλίου 2015, μαζί με το Visual Studio 2015 [8]. Προσέθεσε τον μεταγλωττιστή Just in Time (JIT) που ονομάζεται RyuJIT, καθώς και τα πρώτα πακέτα ανοιχτού κώδικα .NET Framework. Πέρα από τη νέα έκδοση του Visual Studio, η πιο αξιοσημείωτη προσθήκη ήταν η υποστήριξη του πρωτοκόλλου HTTP/2.

Έκδοση 4.7

Το .NET Framework 4.7 κυκλοφόρησε στις 15 Απριλίου 2017, ταυτόχρονα με το Visual Studio 2017 [8]. Περιλάμβανε μερικά νέα χαρακτηριστικά, όπως την επεκτασιμότητα της προσωρινής μνήμης (cache) των αντικειμένων ή τη βελτιωμένη ασφάλεια με την υποστήριξη του TLS (Transport Layer Security), δίχως ωστόσο κάποια νέα καινοτομία.

Έκδοση 4.8

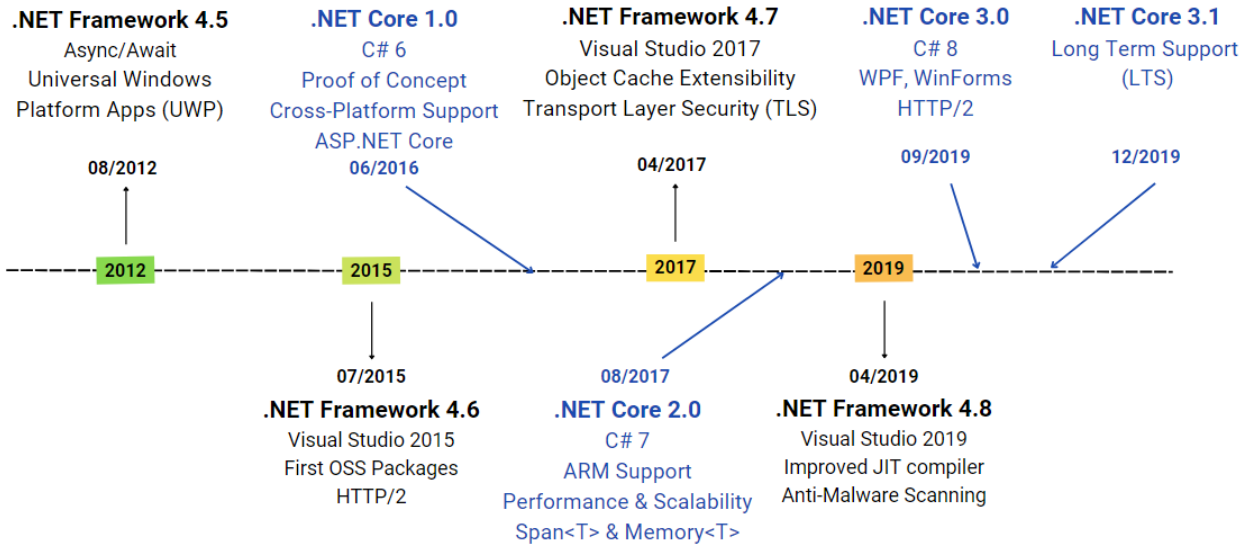
Το .NET Framework 4.8 κυκλοφόρησε στις 18 Απριλίου 2019 και ήταν η τελευταία έκδοση της εποχής .NET Framework [8]. Περιλάμβανε την κυκλοφορία του Visual Studio 2019 αλλά και βελτιώσεις για τον μεταγλωττιστή JIT, ενώ πρόσθεσε σημαντικές ενημερώσεις ασφαλείας από κακόβουλο λογισμικό. Η έκδοση 4.8.1 που κυκλοφόρησε στις 9 Αυγούστου 2022 ήταν και η τελευταία έκδοση ενημέρωσης κώδικα μέχρι σήμερα.

Εκδόσεις .NET Core

Το .NET Framework αποτέλεσε μια τεράστια επιτυχία της Microsoft για πάνω από 15 χρόνια. Όλο και περισσότεροι προγραμματιστές και εταιρείες υιοθέτησαν τις γλώσσες προγραμματισμού και άρχισαν να αναπτύσσουν εφαρμογές βασισμένες στο .NET Framework. Ωστόσο, είναι στενά συνδεδεμένο με το λειτουργικό σύστημα των Windows, περιορίζοντας έτσι τις εφαρμογές που αξιοποιούν το .NET Framework. Επίσης, ο κύκλος εκδόσεων εξαρτάται από τα Windows.

Με τη σύγχρονη ανάπτυξη λογισμικού, απαιτείται ταχεία ανάπτυξη και μικρές και σταδιακές αλλαγές. Μια άλλη τάση είναι ότι σχεδόν όλες οι σύγχρονες πλατφόρμες προγραμματιστών είναι ανοιχτού κώδικα. Η μετατροπή του .NET Framework σε μια πλατφόρμα ανοιχτού κώδικα δεν ήταν ποτέ επιλογή λόγω των προβλημάτων αδειοδότησης που συνδέονται με τα Windows. Παράλληλά, άλλοι περιοριστικοί παράγοντες, όπως ο περιορισμός ως προς το λειτουργικό σύστημα, αποτέλεσαν μεγαλύτερο πρόβλημα με την πάροδο του χρόνου.

Τα προβλήματα αυτά ήρθε να επιλύσει το .NET Core [13]. Σχεδιάστηκε για να είναι η μετεξέλιξη του .NET. Μια δωρεάν πλατφόρμα ανοιχτού κώδικα γενικού σκοπού και πάνω από όλα απεξαρτημένη από το λειτουργικό σύστημα των Windows, επιτρέποντας έτσι σε προγραμματιστές που χρησιμοποιούν Linux ή macOS να αναπτύσσουν εφαρμογές που εκτελούνται σε .NET Core. Αυτά τα χαρακτηριστικά καθιστούν το .NET μια βιώσιμη επιλογή για ανάπτυξη cloud εφαρμογών.



Εικόνα 2.8 – Αρχικές εκδόσεις του .NET Core

Έκδοση Core 1.0

Η πρώτη έκδοση του .NET Core ανακοινώθηκε το 2014 και κυκλοφόρησε στις 27 Ιουνίου του 2016 παράλληλα με την 6 έκδοση της C#, συνεπώς κυκλοφόρησε στο ενδιάμεσο της έκδοσης 4.6 και 4.7 του .NET Framework [9]. Περιείχε έναν περιορισμένο αριθμό υλοποιημένων APIs από την μηδέν και αποτελεί τα θεμέλιά του .NET που χρησιμοποιείται σήμερα.

Έφερε την cross-platform υποστήριξη, ένα ελαφρύ περιβάλλον εκτέλεσης βελτιστοποιημένο για σενάρια υπολογιστικού νέφους καθώς και το **ASP.NET Core**, ένα framework ανάπτυξης δικτυακών εφαρμογών, το οποίο αποτέλεσε τη πρώτη νέα τεχνολογία που βασίστηκε πάνω από το .NET Core.

Έκδοση Core 2.0

Το .NET Core 2.0 κυκλοφόρησε ένα χρόνο αργότερα, στις 14 Αυγούστου 2017, λίγους μήνες μετά τις εκδόσεις .NET Framework 4.7 και Visual Studio 2017, η οποία περιλάμβανε και την 7η έκδοση της C# [9].

Τα κύρια χαρακτηριστικά της δεύτερης έκδοσης ήταν η υποστήριξη ARM (Advanced RISC Machine) για σενάρια IoT, η βελτιωμένη απόδοση και η επεκτασιμότητα, συμπεριλαμβανομένης της υποστήριξης των τύπων δεδομένων Span<T> και Memory<T> για την ευκολότερη διαχείριση της μνήμης.

Έκδοση Core 3.0

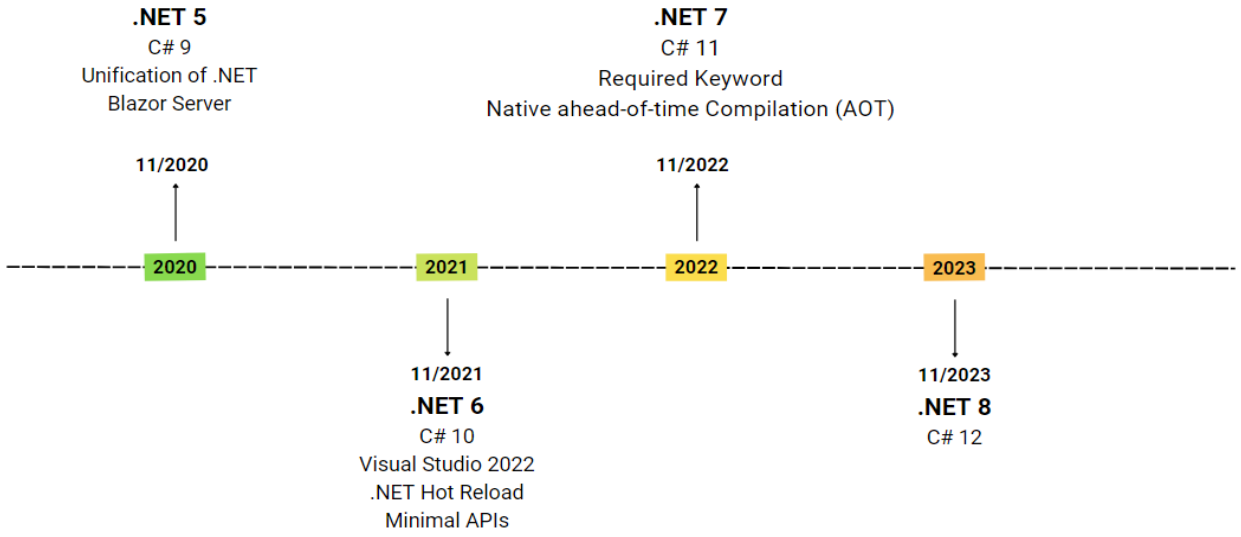
Το NET Core 3.0 κυκλοφόρησε δύο χρόνια αργότερα, στις 23 Σεπτεμβρίου 2019, λίγους μήνες μετά την τελική έκδοση .NET Framework 4.8, ενώ συμπεριλάμβανέ την 8η έκδοση της

C# [9]. Προσέθεσε επίσης υποστήριξη για την ανάπτυξη desktop εφαρμογών των Windows μέσω WPF και WinForms, αλλά και υποστήριξη του πρωτόκολλου HTTP/2.

Έκδοση Core 3.1

Το .NET Core 3.1 είναι μια έκδοση μακροπρόθεσμης υποστήριξης (Long-term Support - LTS) που υιοθετήθηκε ιδιαίτερα από την κοινότητα, κυκλοφόρησε στις 3 Δεκεμβρίου 2019 [9]. Οι προηγούμενες εκδόσεις .NET Core ήταν κυρίως για άτομα που πειραματίζονταν με τη νέα υλοποίηση της πλατφόρμας. Ξεκινώντας με το .NET Core 3.1, υπήρξε μια τεράστια στροφή προς την ανάπτυξη εφαρμογών στο νέο .NET Core αντί για το παλιό .NET Framework.

Έκδοση .NET 5



Εικόνα 2.9 – Η νέα γενιά του .NET

Τον Νοέμβριο του 2020 κυκλοφόρησε το .NET 5, όπου για πρώτη φορά το **.NET Core** μετονομάστηκε σε απλά **.NET**, ενώ υπήρξε η μεταπήδηση από την έκδοση 3 στην έκδοση 5 για να αποφευχθεί η σύγχυση του .NET Framework με το .NET Core [9].

Με την έλευση του .NET 5 ήρθε επίσης και η 9η έκδοση της C# και η πρωτοποριακή τεχνολογία **ASP.NET Core Blazor** [14], ένα framework για την δημιουργία client-side εφαρμογών με σύγχρονη και διαδραστική διεπαφή χρήστη (User Interface - UI), χρησιμοποιώντας C# αντί Javascript. Το Blazor προσφέρει δύο διαφορετικά μοντέλα φιλοξενίας (hosting models), το **Blazor Server** και το **Blazor WebAssembly**.

1. **Blazor Server**: αφορά το μοντέλο φιλοξενίας σε κάποιο διακομιστή (server) από τον οποίο και εκτελείται μέσω μιας εφαρμογής ASP.NET Core. Ο server επικοινωνεί με τον client μέσω μιας SignalR διασύνδεσης. Όταν γίνεται κάποια ενέργεια (event) από την μεριά του client, όπως για παράδειγμα ένα κλικ του ποντικιού, οι πληροφορίες

σχετικά με την ενέργεια αυτή αποστέλλονται στον server μέσω της SignalR διασύνδεσης. Ο server με τη σειρά του διαχειρίζεται το event και υπολογίζει τη διαφορά που δημιουργείται σε επίπεδο HTML. Έτσι δεν αποστέλλεται ολόκληρο το HTML πίσω στο client, παρά μόνο η διαφορά, με βάση την οποία ενημερώνεται η διεπαφή χρήστη (UI) από το πρόγραμμα περιήγησης (browser).

2. **Blazor WebAssembly**: αφορά το μοντέλο φιλοξενίας απευθείας στο browser του χρήστη και εκτελείται μέσω του WebAssembly. Έτσι, όλα όσα χρειάζεται η εφαρμογή, όπως ο μεταγλωττισμένος κώδικας εφαρμογής, οι εξαρτήσεις (βιβλιοθήκες κλπ) και ο εκτελέσιμος κώδικας .NET φορτώνονται στο πρόγραμμα περιήγησης του χρήστη.

Εικόνα 2.10 – Hosting models της τεχνολογίας Blazor

Το Blazor βασίζεται στην αρχιτεκτονική SPA (Single Page Application) όπου ενημερώνει τον κώδικα της ίδιας σελίδας ως απόκριση στις ενέργειες του χρήστη. Δεδομένου ότι μόνο η διαφορά απαιτείται για την ενημέρωση του UI, η εφαρμογή γίνεται πιο γρήγορη και ανταποκρίνεται άμεσα στον χρήστη.

Η τεχνολογία αυτή είναι πολύ σημαντική καθώς επέτρεψε στους προγραμματιστές που γράφουν εφαρμογές σε C# και .NET να μπορούν να αξιοποιήσουν τις δεξιότητες τους για front-end εφαρμογές αντί να μάθουν κάποιο framework της Javascript που έχουν μεγάλη καμπύλη εκμάθησης.

Αξίζει να σημειωθεί πως από την κυκλοφορία του .NET 5 και έπειτα, η Microsoft ακολουθεί μία νέα πολιτική όσον αφορά τις εκδόσεις του .NET. Συγκεκριμένα, κυκλοφορεί μια νέα έκδοση .NET και μια νέα έκδοση του μεταγλωττιστή C# κάθε χρόνο. Οι ζυγοί αριθμοί αφορούν εκδόσεις μακροπρόθεσμης υποστήριξης (LTS) με υποστήριξη τριών ετών ενώ οι μονοί αριθμοί είναι βραχυπρόθεσμες εκδόσεις με υποστήριξη 18 μηνών [13].

Έκδοση .NET 6

Το .NET 6 κυκλοφόρησε στις 8 Νοεμβρίου του 2021 [9]. Ήταν η πρώτη έκδοση μακροπρόθεσμης υποστήριξης (LTS) μετά την μετονομασία σε «.NET». Παράλληλα, έφερε την 10η έκδοση της C# και το Visual Studio 2022, που συμπεριλάμβανέ την τεχνολογία **Hot Reload** [15]**,** που επιτρέπει στους προγραμματιστές να κάνουν αλλαγές στον κώδικα ενώ τρέχει σε debug mode, με το πάτημα ενός κουμπιού, βελτιώνοντας έτσι την εμπειρία του προγραμματιστή και επιταχύνοντας την ανάπτυξη λογισμικού. Επιπλέον, κυκλοφόρησαν τα Minimal APIs για την ανάπτυξη σε ASP.NET Core, τα top-level statements και τα file-scoped namespaces.

Έκδοση .NET 7

Στις 8 Νοεμβρίου του 2022 κυκλοφόρησε το .NET 7, ως η τρέχουσα έκδοση βραχυπρόθεσμης υποστήριξης, μαζί με την 11η έκδοση της C# [9]. Έφερε συγκεκριμένες προσθήκες όπως η λέξη κλειδί “required” και η ενσωμάτωση της μεταγλώττισης AOT (Ahead-of-Time compilation), για τη παραγωγή αυτόνομων εκτελέσιμων αρχείων. Στο άμεσο μέλλον, θα υπάρχει μια νέα έκδοση .NET και C# κάθε χρόνο γύρω στο Νοέμβριο. Το .NET 8 και η 12η έκδοση της C# αναμένεται να κυκλοφορήσει τον Νοέμβριο του 2023.

2.2.3. Πλεονεκτήματα του .NET

Το .NET προσφέρει αρκετά πλεονεκτήματα που συμβάλλουν στη δημοτικότητα και ευρεία υιοθέτησή του από τους προγραμματιστές [16]. Μερικά από τα κύρια πλεονεκτήματα της χρήσης της πλατφόρμας .NET είναι:

1. **Υποστήριξη Πολλαπλών Γλωσσών**: Ένα από τα μεγάλα πλεονεκτήματα της αρχιτεκτονικής του .NET είναι η υποστήριξη πολλαπλών γλωσσών προγραμματισμού, όπως C#, Visual Basic .NET (VB.NET), F# και άλλες. Αυτή η δυνατότητα διαλειτουργικότητας μεταξύ διαφορετικών γλωσσών επιτρέπει στους προγραμματιστές να χρησιμοποιούν την προτιμώμενη τους γλώσσα, ενώ ταυτόχρονα μπορούν να συνεργάζονται με εφαρμογές που έχουν γραφτεί σε άλλες γλώσσες.
2. **Ανεξαρτησία Πλατφόρμας**: Όπως αναφέρθηκε στη προηγούμενη ενότητα, μέσω του .NET Core, της cross-platform έκδοσης του .NET Framework, οι προγραμματιστές έχουν τη δυνατότητα να δημιουργούν εφαρμογές που μπορούν να εκτελούνται σε διάφορες πλατφόρμες, όπως Windows, macOS και Linux. Αυτό απλοποιεί τη διαδικασία ανάπτυξης εφαρμογών που μπορούν να χρησιμοποιηθούν σε διάφορα λειτουργικά συστήματα χωρίς περιορισμούς.
3. **Επαναχρησιμοποίηση Κώδικα**: Η αρχιτεκτονική του .NET προωθεί την επαναχρησιμοποίηση του κώδικα μέσω της χρήσης βιβλιοθηκών και κλάσεων από τη Βασική Βιβλιοθήκη Κλάσεων (BCL) και από βιβλιοθήκες υλοποιημένες από τρίτους. Οι προγραμματιστές μπορούν να εκμεταλλευτούν αυτά τα έτοιμα στοιχεία,

εξοικονομώντας χρόνο και κόπο αποφεύγοντας την ανάγκη να γράψουν κώδικα από την αρχή.

4. **Παραγωγικότητα**: Με την πλούσια σε εργαλεία, βιβλιοθήκες και φυσικά την υποστήριξη από ισχυρά Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης (Integrated Development Environments - IDEs), όπως το Visual Studio και το Visual Studio Code. η πλατφόρμα .NET ενισχύει την παραγωγικότητα και την αποδοτικότητα των προγραμματιστών. Αυτά τα περιβάλλοντα παρέχουν στους προγραμματιστές ισχυρά εργαλεία για την επεξεργασία κώδικα, εντοπισμό σφαλμάτων (debugging), εκτέλεση δοκιμών (testing), αυτόματη διαχείριση μνήμης και έτοιμα templates εφαρμογών. Προσφέρουν επίσης δυνατότητες όπως επισήμανση σύνταξης, αυτόματη συμπλήρωση κώδικα και ενσωματωμένο Git.
5. **Επιδόσεις και Επεκτασιμότητα**: Οι εφαρμογές υλοποιημένες σε .NET είναι εξαιρετικά αποδοτικές και επεκτάσιμες. Χάρις το CLR, βελτιστοποιείται η εκτέλεση κώδικα και η διαχείριση της μνήμης, ενώ η μεταγλώττιση άμεσου χρόνου (JIT) αξιοποιεί τους πόρους του συστήματος με κατάλληλο τρόπο επιτυγχάνοντας κατά αυτό το τρόπο καλύτερες επιδόσεις. Επιπλέον, η πλατφόρμα υποστηρίζει παράλληλο προγραμματισμό και ασύγχρονες μεθόδους, δίνοντας στους προγραμματιστές τη δυνατότητα να δημιουργούν επεκτάσιμες που αδιάλειπτες εφαρμογές.
6. **Ασφάλεια**: Το .NET Framework δίνει προτεραιότητα στην ασφάλεια και διαθέτει ενσωματωμένους μηχανισμούς προστασίας των εφαρμογών. Παρέχει λειτουργίες περιορισμού πρόσβασης κώδικα, κρυπτογραφημένη κωδικοποίηση και ελέγχους ταυτότητας και εξουσιοδότησης. Επιπλέον, η Microsoft φροντίζει να εκδίδει τακτικά ενημερώσεις ασφαλείας και βελτιώσεις, έτσι ώστε να ακολουθούνται οι πλέον βέλτιστες πρακτικές και οι εφαρμογές υλοποιημένες σε .NET να είναι ασφαλείς και θωρακισμένες από κοινές απειλές ασφαλείας.
7. **Διασύνδεση με το Οικοσύστημα της Microsoft:** Το .NET διασυνδέεται πλήρως με άλλες τεχνολογίες και εργαλεία της Microsoft, συμπεριλαμβανομένων των Visual Studio IDEs, των cloud υπηρεσιών μέσω της πλατφόρμας Microsoft Azure, του SQL Server, του Active Directory, των εφαρμογών του Microsoft Office ή Office 365, και άλλα πολλά. Η διασύνδεση αυτή απλοποιεί την ανάπτυξη και την εγκατάσταση των εφαρμογών στο οικοσύστημα της Microsoft και επιτρέπει στους προγραμματιστές να αξιοποιούν πρόσθετες δυνατότητες και υπηρεσίες που προσφέρει η Microsoft.
8. **Συμβατότητα με Προηγούμενες Εκδόσεις**: Το .NET framework προσφέρει συμβατότητα με προηγούμενες εκδόσεις, διασφαλίζοντας ότι οι υπάρχουσες εφαρμογές συνεχίζουν να λειτουργούν ομαλά σε κάθε νέα έκδοση ή ενημέρωση. Αυτό σημαίνει ότι οι οργανισμοί μπορούν να αναβαθμίσουν τις παλαιότερες εφαρμογές τους σε νεότερες εκδόσεις του framework χωρίς σημαντικές δυσκολίες ή την ανάγκη επέμβασης στον υπάρχον κώδικα. Το .NET υποστηρίζει επίσης τη διαλειτουργικότητα με υπάρχοντα κώδικα και συστήματα, διευκολύνοντας την ενοποίηση με παλαιότερες εφαρμογές και υποδομές.

9. **Κατάλληλο για Εταιρικές Λύσεις**: Το .NET ενδείκνυται για ανάπτυξη εταιρικών εφαρμογών, καθώς παρέχει υψηλή ασφάλεια και επιλογές επεκτασιμότητας. Επιπλέον, προσφέρει ισχυρή υποστήριξη για τη δημιουργία web services, τη δημιουργία API και την εύκολη διασύνδεση τους με άλλα εταιρικά συστήματα. Αυτό το καθιστά ιδανική επιλογή για την ανάπτυξη εφαρμογών μεγάλης κλίμακας, που απαιτούν υψηλή απόδοση, αξιοπιστία και επεκτασιμότητα.

2.3. Σχεσιακές Βάσεις Δεδομένων

Οι Σχεσιακές Βάσεις Δεδομένων (Relational Databases) αποτελούν τη ραχοκοκαλιά των σύγχρονων συστημάτων διαχείρισης βάσεων δεδομένων ΣΒΔΒ (DBMS, Database Management System), που υποστηρίζουν διάφορες εφαρμογές, ιστότοπους και επιχειρήσεις σε όλο τον κόσμο. Αυτές οι βάσεις δεδομένων βασίζονται σε ένα καλά καθορισμένο σύνολο αρχών που επιτρέπουν την αποτελεσματική οργάνωση, αποθήκευση, ανάκτηση και συντήρηση των δεδομένων.

2.3.1. Εισαγωγή Σχεσιακό Μοντέλο Δεδομένων

Το σχεσιακό μοντέλο (relational model) είναι ένα θεμελιώδες αποτελεί θεμελιώδη αρχή στον τομέα της διαχείρισης βάσεων δεδομένων, παρέχοντας μια δομημένη προσέγγιση για την οργάνωση και τη διαχείριση δεδομένων. Εισάγει την έννοια των πινάκων, όπου τα δεδομένα αποθηκεύονται σε γραμμές και στήλες. Αυτό το μοντέλο, που αναπτύχθηκε από τον Edgar F. Codd τη δεκαετία του 1970 [17], αποτελεί τη βάση για τα περισσότερα σύγχρονα συστήματα βάσεων δεδομένων και είναι γνωστό για την απλότητα, την ευελιξία και την αποτελεσματικότητά του στην αναπαράσταση πολύπλοκων σχέσεων δεδομένων. Αποτελεί τον ακρογωνιαίο λίθο της οργάνωσης δεδομένων, επιτρέποντας την αποτελεσματική ανάκτηση, αποθήκευση και χειρισμό δεδομένων με λογικό και συνεπή τρόπο.

2.3.2. Σχέσεις

Στο σχεσιακό μοντέλο δεδομένων, μια «σχέση» (relation) είναι μια μαθηματική έννοια που αντιστοιχεί σε αυτό που συνήθως αναφέρεται ως «πίνακας» σε ένα σύστημα σχεσιακής βάσης δεδομένων [18]. Ο όρος «σχέση» χρησιμοποιείται για να τονίσει τη μαθηματική βάση του μοντέλου. Ουσιαστικά, μια σχέση είναι ένα σύνολο δεδομένων οργανωμένων σε γραμμές (πλειάδες) και στήλες (χαρακτηριστικά ή πεδία) που τηρούν συγκεκριμένους κανόνες και ιδιότητες. Πιο συγκεκριμένα, κάθε γραμμή του πίνακα αντιπροσωπεύει μια μεμονωμένη εγγραφή δεδομένων μιας οντότητας και κάθε στήλη αντιστοιχεί σε ένα χαρακτηριστικό της οντότητας. Για παράδειγμα, η σχέση «Students» που φαίνεται στον πίνακα 2.1, αντιπροσωπεύει το σύνολο των οντοτήτων Students, όπου κάθε γραμμή αντιστοιχεί σε μια οντότητα Student και κάθε στήλη αντιστοιχεί σε ένα από τα χαρακτηριστικά του συνόλου οντοτήτων, όπως είναι ο Αριθμός Μητρώου, το Όνομα και το Επίθετο.

Students			
ID	FirstName	LastName	Department
1234	Antonios	Kassos	Informatics
5678	George	Papadopoulos	Informatics

Πίνακας 2.1 - Η σχέση Students

Η διάταξη των πλειάδων ή των χαρακτηριστικών σε μια σχέση δεν έχει απολύτως καμία σημασία. Συνεπώς, ακόμα και αν η σχέση Students είχε διαφορετική σειρά ως προς τα τις στήλες ή τις γραμμές, θα πρόκυπτε μια ισοδύναμη σχέση.

2.3.3. Πεδίο τιμών

Το σχεσιακό μοντέλο δεδομένων απαιτεί να υπάρχει ομοιογένεια των χαρακτηριστικών μιας οντότητας ως προς το τύπο δεδομένων. Αυτό σημαίνει πως όλες οι τιμές μιας στήλης σε έναν πίνακα πρέπει να έχουν ένα συγκεκριμένο πεδίο τιμών (domain), δηλαδή ένα σύνολο επιτρεπόμενων τιμών [18]. Για παράδειγμα, το πεδίο τιμών για τη στήλη "ID" της σχέσης Students θα πρέπει να είναι οι ακέραιοι αριθμοί, ενώ για τις στήλες “FirstName” και “LastName” τα αλφαριθμητικά. Επιπλέον, οι τιμές σε κάθε στήλη θα πρέπει να είναι ατομικές, που σημαίνει πως θα πρέπει να ανήκουν σε κάποιο στοιχειώδη τύπο δεδομένων και όχι σε κάποιο σύνολο, δομή ή λίστα δεδομένων που να μπορούν να διαιρεθούν περαιτέρω.

2.3.4. Σχήμα και Στιγμιότυπο Σχέσης

Το όνομα και το σύνολο το χαρακτηριστικών μιας σχέσης, ονομάζεται «Σχήμα» (Schema) [18]. Το σχήμα καθορίζει τις στήλες που συνθέτουν έναν πίνακα, συμπεριλαμβανομένου των ονομάτων και τους τύπους δεδομένων τους. Ένα σχήμα μπορεί επίσης να περιέχει τυχόν περιορισμούς ακεραιότητας που εφαρμόζονται σε μία σχέση, όπως για παράδειγμα το Πρωτεύον Κλειδί της σχέσης. Για παράδειγμα, το σχήμα της σχέσης Student είναι το εξής:

Student (ID, FirstName, LastName, Department)

εναλλακτικά

Student (ID: integer, FirstName: string, LastName: string, Department: string)

Στη δεύτερη περίπτωση, επισημαίνεται και ο τύπος δεδομένων κάθε στήλης. Το υπογραμμισμένο χαρακτηριστικό «ID» δηλώνει ότι είναι Πρωτεύον Κλειδί της σχέσης, που σημαίνει ότι οι γραμμές του πίνακα δεν μπορούν να έχουν την ίδια τιμή στο πεδίο «ID», διασφαλίζοντας την μοναδικότητα της κάθε γραμμής.

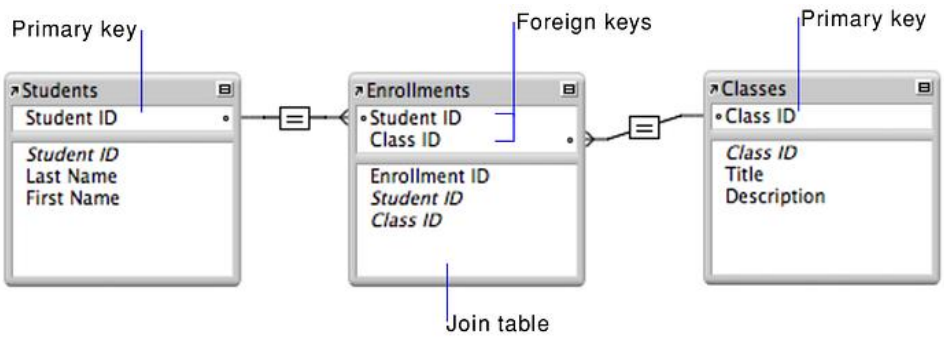
Στο σχεσιακό μοντέλο, μια βάση δεδομένων αποτελείται από μία ή περισσότερες σχέσεις/πίνακες. Το σύνολο των σχημάτων των σχέσεων που την απαρτίζουν ονομάζεται «Σχήμα της Βάσης Δεδομένων» (Database Schema).

Ο αριθμός των γραμμών σε έναν πίνακα είναι δυναμικός και μπορεί να αλλάξει, καθώς εισάγονται, ενημερώνονται και διαγράφονται δεδομένα. Ως «Στιγμιότυπο Σχέσης» (Relation Instance), ονομάζουμε τα δεδομένα που εμπεριέχονται σε έναν πίνακα σε μια συγκεκριμένη χρονική στιγμή. Για παράδειγμα, οι δύο πλειάδες που φαίνονται στον πίνακα 2.1 αποτελούν ένα στιγμιότυπο της σχέσης Student.

2.3.5. Κλειδιά

Τα κλειδιά είναι χαρακτηριστικά ή συνδυασμοί χαρακτηριστικών που χρησιμοποιούνται για τον μοναδικό προσδιορισμό γραμμών (πλειάδων) μέσα σε έναν πίνακα. Διασφαλίζουν την ακεραιότητα των δεδομένων, δημιουργούν συσχετίσεις (relationships) μεταξύ των πινάκων και διευκολύνουν την αποτελεσματική ανάκτηση δεδομένων [18]. Υπάρχουν διάφοροι τύποι κλειδιών που χρησιμοποιούνται συνήθως στο σχεσιακό μοντέλο:

- **Πρωτεύον Κλειδί (Primary Key – PK):** Ως πρωτεύον κλειδί ορίζεται ένα πεδίο ενός πίνακα που χαρακτηρίζει μοναδικά κάθε εγγραφή του. Κάθε πίνακας πρέπει να έχει ένα πρωτεύον κλειδί. Στη σχέση Students για παράδειγμα, το πρωτεύον κλειδί είναι το χαρακτηριστικό «ID», δηλαδή ο αριθμός μητρώου.
- **Υποψήφιο Κλειδί (Candidate Key):** Ως υποψήφιο ή εναλλακτικό κλειδί ονομάζεται το πεδίο ή το σύνολο πεδίων του πίνακα που θα μπορούσε ενδεχομένως να αξιοποιηθεί ως πρωτεύον κλειδί. Εάν ένας πίνακας έχει πολλά υποψήφια κλειδιά, ένα από αυτά επιλέγεται ως πρωτεύον κλειδί.
- **Υπερκλειδί (Superkey)**: Ένα υπερκλειδί είναι ένα σύνολο πεδίων που μπορεί να προσδιορίσει μοναδικά εγγραφές σε έναν πίνακα, αλλά μπορεί να περιλαμβάνει περισσότερα πεδία από όσα χρειάζεται. Οποιοδήποτε υποσύνολο ενός υπερκλειδιού είναι επίσης ένα υπερκλειδί. Για παράδειγμα, στο πίνακα Students, ένα υπερκλειδί μπορεί να αποτελέσει ο συνδυασμός των πεδίων ID, FirstName και LastName.
- **Ξένο Κλειδί (Foreign key – FK):** Ένα ξένο κλειδί είναι ένα πεδίο ή ένα σύνολο πεδίων σε έναν πίνακα που αναφέρεται στο πρωτεύον κλειδί ενός άλλου πίνακα. Τα ξένα κλειδιά χρησιμοποιούνται για τη δημιουργία συσχετίσεων (relationships) μεταξύ διαφορετικών πινάκων. Για παράδειγμα, ο πίνακας «Enrollments» της εικόνας 11 διαθέτει δύο ξένα κλειδιά, τα πεδία «StudentID» και «ClassID» που αναφέρονται στα πρωτεύοντα κλειδιά των πινάκων «Students» και «Classes» αντίστοιχα.
- **Σύνθετο Κλειδί (Composite Key)**: Ως σύνθετο κλειδί ορίζεται ένα πρωτεύον κλειδί που αποτελείται από δύο ή περισσότερα πεδία. Χρησιμοποιείται όταν ένα μεμονωμένο χαρακτηριστικό δεν μπορεί να προσδιορίσει μοναδικά τις εγγραφές ενός πίνακα, αλλά ο συνδυασμός πολλαπλών χαρακτηριστικών το κάνει.



Εικόνα 2.11 – Συσχέτιση πινάκων με τη χρήση Foreign Key

2.3.6. Περιορισμοί

Στο σχεσιακό μοντέλο, οι «Περιορισμοί» (Constraints) είναι κανόνες και συνθήκες που ορίζονται και επιβάλλονται για να διατηρηθεί η ακεραιότητα και η συνέπεια των δεδομένων. Οι περιορισμοί διασφαλίζουν πως τα δεδομένα αποθηκεύονται στη βάση δεδομένων παραμένουν ακριβή, αξιόπιστα και συμμορφώνονται σε προκαθορισμένους κανόνες [18]. Υπάρχουν διάφοροι τύποι περιορισμών όπως:

- **Περιορισμός Πρωτεύοντος Κλειδιού**: Ο περιορισμός πρωτεύοντος κλειδιού εξασφαλίζει τη μοναδικότητα των τιμών σε μία ή και περισσότερες στήλες ενός πίνακα, που έχουν οριστεί ως πρωτεύον κλειδί της σχέσης. Ό περιορισμός αυτός εγγυάται επίσης πως αυτές οι στήλες δεν περιέχουν NULL τιμές. Σε κάθε πίνακα μπορεί να οριστεί μονάχα ένα Πρωτεύον Κλειδί.
- **Περιορισμός Μοναδικότητας**: Σε αντίθεση με τον περιορισμό Πρωτεύοντος Κλειδιού, ο περιορισμός μοναδικότητας (Unique constraint) διασφαλίζει ότι οι τιμές σε μία ή περισσότερες στήλες, που έχουν δηλωθεί ως UNIQUE, είναι μοναδικές σε όλες τις γραμμές ενός πίνακα, χωρίς ωστόσο να αποτρέπει τις NULL τιμές. Επίσης, ένας πίνακας μπορεί να έχει οποιοδήποτε αριθμό από δηλώσεις UNIQUE.
- **Περιορισμός Ξένου Κλειδιού**: Ο περιορισμός αυτός επιβάλλει την ακεραιότητα αναφοράς διασφαλίζοντας ότι οι τιμές στη στήλη ξένου κλειδιού ενός πίνακα ταιριάζουν με τις τιμές στη στήλη πρωτεύοντος κλειδιού ενός άλλου πίνακα. Για παράδειγμα, προκειμένου να καταχωρηθεί μία νέα εγγραφή στο πίνακα «Enrollments» της εικόνας 11, θα πρέπει τα πεδία «StudentID» και «ClassID» να αντιστοιχούν σε έγκυρες εγγραφές των αντίστοιχων πινάκων.
- **Περιορισμός Ελέγχου**: Ο περιορισμός ελέγχου καθορίζει μια συνθήκη που πρέπει να ικανοποιούν οι τιμές σε μία ή περισσότερες στήλες για να θεωρούνται έγκυρες. Χρησιμοποιείται για την επιβολή συγκεκριμένων κανόνων ή προϋποθέσεων σχετικά με τις τιμές των δεδομένων. Για παράδειγμα, μια στήλη "Βαθμολογία" σε έναν πίνακα "Εργασίες" μπορεί να έχει έναν περιορισμό ελέγχου για να διασφαλιστεί ότι οι τιμές των βαθμολογιών είναι μεταξύ 0 και 10.

- **Περιορισμός Not-Null:** Ο περιορισμός Not-Null απαιτεί μια στήλη να μην μπορεί να περιέχει τιμές NULL, διασφαλίζοντας τη παρουσία δεδομένων σε συγκεκριμένες στήλες όπου δεν επιτρέπονται μηδενικές τιμές.

2.3.7. Κανονικοποίηση

Ο Edgar F. Codd εισήγαγε επίσης για πρώτη φορά την έννοια της Κανονικοποίησης, ως μέρος του σχεσιακού μοντέλου [18]. Η Κανονικοποίηση είναι μια κρίσιμη έννοια στις σχεσιακές βάσεις δεδομένων που εστιάζει στη βελτιστοποίηση της δομής της βάσης δεδομένων για την εξάλειψη του πλεονασμού και τη διασφάλιση της ακεραιότητας των δεδομένων. Η διαδικασία περιλαμβάνει τη διάσπαση μεγάλων πινάκων σε μικρότερους, συσχετισμένους πίνακες και την οργάνωση των δεδομένων με αποτελεσματικό τρόπο.

Η Κανονικοποίηση, τυπικά χωρίζεται σε πολλές κανονικές μορφές, από την πρώτη κανονική μορφή (1NF) έως τα υψηλότερα επίπεδα όπως η τρίτη κανονική μορφή (3NF) και πέρα. Κάθε κανονική μορφή έχει συγκεκριμένους κανόνες και οδηγίες για την εξάλειψη του πλεονασμού και τη διασφάλιση της συνέπειας των δεδομένων. Για παράδειγμα, σε μια μη κανονικοποιημένη βάση δεδομένων, η διεύθυνση ενός πελάτη μπορεί να αποθηκευτεί τόσο στον πίνακα "Πελάτες" όσο και στον πίνακα "Παραγγελίες", οδηγώντας σε πλεονασμό. Με την κανονικοποίηση της βάσης δεδομένων, οι πληροφορίες διεύθυνσης μπορούν να αποθηκευτούν σε έναν ξεχωριστό πίνακα "Διευθύνσεις" και ο πίνακας "Παραγγελίες" μπορεί να αναφέρεται σε αυτόν χρησιμοποιώντας ένα ξένο κλειδί, μειώνοντας τα διπλότυπα δεδομένα.

2.4. Η γλώσσα Βάσεων Δεδομένων SQL

Η Structured Query Language ή SQL είναι μια ισχυρή και τυποποιημένη γλώσσα που παίζει κεντρικό ρόλο στον κόσμο της διαχείρισης δεδομένων. Η SQL είναι το κύριο μέσο με το οποίο οι χρήστες αλληλεπιδρούν με σχεσιακές βάσεις δεδομένων, επιτρέποντάς τους να ανακτούν, να χειρίζονται και να οργανώνουν δεδομένα αποτελεσματικά. Στο παρακάτω κεφάλαιο γίνεται αναφορά στις βασικές αρχές, τη σύνταξη και τις πρακτικές εφαρμογές της SQL, τονίζοντας την σημασία της στη λήψη αποφάσεων βάσει των δεδομένων σε διάφορους κλάδους.

2.4.1. Σύντομη ιστορική αναδρομή

Η προέλευση της SQL μπορεί να εντοπιστεί στις αρχές της δεκαετίας του 1970, όταν ο ερευνητής της IBM Dr. Edgar F. Codd εισήγαγε την έννοια του μοντέλου σχεσιακής βάσης δεδομένων. Το πρωτοποριακό έργο του Codd έθεσε τα θεμέλια για αυτό που θα γινόταν η γλώσσα SQL. Η πρώτη επίσημη προδιαγραφή της SQL δημοσιεύτηκε από την IBM το 1974 ως Structured English Query Language (SEQUEL). Ωστόσο, λόγω ζητημάτων εμπορικών σημάτων, η SEQUEL μετονομάστηκε αργότερα σε SQL [17].

Η SQL κέρδισε γρήγορα δημοτικότητα και έγινε βιομηχανικό πρότυπο για συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS). Εγκρίθηκε επίσημα ως πρότυπο αρχικά από το Αμερικανικό Εθνικό Ινστιτούτο Προτύπων (ANSI) το 1986 και ακολούθησε ο Διεθνής Οργανισμός Τυποποίησης (ISO) το 1987 [19]. Έκτοτε, η SQL έχει υποστεί αρκετές αναθεωρήσεις, με τις SQL-92, SQL:1999 , SQL:2003, SQL:2008, SQL:2011, SQL:2016 και SQL:2019 είναι οι πιο αξιοσημείωτες εκδόσεις.

2.4.2. Βασικές αρχές της SQL

Η SQL βασίζεται σε ένα σύνολο βασικών αρχών [18] που καθορίζουν τη λειτουργικότητα και τη δομή της:

Ανάκτηση δεδομένων

Η κύρια λειτουργία της SQL είναι η ανάκτηση δεδομένων από σχεσιακές βάσεις δεδομένων. Οι χρήστες μπορούν να καθορίσουν τα δεδομένα που χρειάζονται χρησιμοποιώντας τη δήλωση SELECT, η οποία επιτρέπει το φιλτράρισμα, την ταξινόμηση και τη συγκέντρωση δεδομένων. Για παράδειγμα, ένα απλό ερώτημα (query) SQL μπορεί να ανακτήσει μια λίστα μαθημάτων από μια βάση δεδομένων ενός Πανεπιστημίου.

Χειρισμός δεδομένων

Η SQL υποστηρίζει λειτουργίες χειρισμού δεδομένων, συμπεριλαμβανομένης της εισαγωγής δεδομένων (INSERT), της τροποποίησης (UPDATE) και της διαγραφής (DELETE). Αυτές οι λειτουργίες επιτρέπουν στους χρήστες να προσθέτουν, να ενημερώνουν και να αφαιρούν δεδομένα στη βάση δεδομένων.

Ορισμός δεδομένων

Η SQL επιτρέπει στους χρήστες να ορίζουν και να διαχειρίζονται τη δομή μιας βάσης δεδομένων χρησιμοποιώντας δηλώσεις όπως CREATE TABLE, ALTER TABLE και DROP TABLE. Αυτές οι δηλώσεις διέπουν τον τρόπο οργάνωσης των δεδομένων σε πίνακες.

Ασφάλεια δεδομένων

Η SQL παρέχει μηχανισμούς για τη διαχείριση της πρόσβασης και της ασφάλειας στη βάση δεδομένων. Οι χρήστες μπορούν να ορίσουν ρόλους χρήστη, δικαιώματα και περιορισμούς πρόσβασης για να ελέγχουν ποιος μπορεί να εκτελεί συγκεκριμένες λειτουργίες στη βάση δεδομένων.

Συναλλαγές

Η SQL υποστηρίζει συναλλαγές (transactions), οι οποίες είναι ακολουθίες εντολών SQL που αντιμετωπίζονται ως ενιαία, ατομική μονάδα εργασίας. Οι συναλλαγές διασφαλίζουν την ακεραιότητα των δεδομένων είτε πραγματοποιώντας όλες τις αλλαγές είτε επαναφέροντάς τις σε περίπτωση σφάλματος.

Τύποι δεδομένων

Η SQL υποστηρίζει διάφορους τύπους δεδομένων, συμπεριλαμβανομένων ακεραίων, συμβολοσειρών, ημερομηνιών και άλλων. Οι τύποι δεδομένων διασφαλίζουν ότι τα δεδομένα αποθηκεύονται και ανακτώνται στην κατάλληλη μορφή.

2.4.3. Σύνταξη και βασικές λειτουργίες

Η σύνταξη της SQL είναι ισχυρή και φιλική προς το χρήστη. Ανάλογα το ΣΔΒΔ μπορεί να διαφέρει ελάχιστα, αλλά η βασική ιδέα παραμένει ίδια. Ακολουθούν μερικά παραδείγματα ερωτημάτων SQL για κάποιες από τις βασικότερες λειτουργίες της:

- Η δήλωση **SELECT**: Η δήλωση SELECT χρησιμοποιείται για την ανάκτηση δεδομένων από έναν ή περισσότερους πίνακες. Επιτρέπει στους χρήστες να επιλέξουν ποιες στήλες θα ανακτήσουν και θα εφαρμόσουν συνθήκες χρησιμοποιώντας τον όρο **WHERE**. Για τον συνδυασμό δεδομένων από πολλούς πίνακες χρησιμοποιείται η δήλωση **JOIN**.
  Παράδειγμα ερωτήματος για την ανάκτηση όλων των στηλών ενός πίνακα:

  Παράδειγμα ερωτήματος για την ανάκτηση συγκεκριμένων στηλών ενός πίνακα:

  Παράδειγμα εφαρμογή συνθήκης για το φιλτράρισμα των αποτελεσμάτων:

  Παράδειγμα ανάκτησης δεδομένων από πολλαπλούς πίνακες:

- Η δήλωση **INSERT**: Η δήλωση INSERT χρησιμοποιείται για την προσθήκη νέων εγγραφών σε έναν πίνακα. Οι χρήστες αναφέρουν τον πίνακα προορισμού και ορίζουν τιμές για τις στήλες στις οποίες πρόκειται να εισαχθούν τα δεδομένα.

  Παράδειγμα προσθήκης μιας νέας εγγραφής σε έναν πίνακα:

- Η δήλωση **UPDATE**: Η δήλωση UPDATE τροποποιεί τις υπάρχουσες εγγραφές σε έναν πίνακα. Οι χρήστες αναφέρουν τον πίνακα προορισμού και τις στήλες που θα ενημερωθούν και ορίζουν τις νέες τιμές. Οι συνθήκες του όρου WHERE καθορίζουν ποιες εγγραφές επηρεάζονται.

Παράδειγμα ενημέρωσης εγγραφών ενός πίνακα:

- Η δήλωση **DELETE**: Η δήλωση DELETE αφαιρεί εγγραφές από έναν πίνακα. Όπως και η δήλωση UPDATE, χρησιμοποιεί τον όρο WHERE για να καθορίσει ποιες εγγραφές θα διαγραφούν.

  Παράδειγμα

- Η δήλωση **CREATE TABLE**: Η δήλωση CREATE TABLE ορίζει τη δομή ενός νέου πίνακα. Οι χρήστες ορίζουν το όνομα του πίνακα, τα ονόματα στηλών, τους τύπους δεδομένων και τους αντίστοιχους περιορισμούς, όπως τα πρωτεύοντα και ξένα κλειδιά.

  Παράδειγμα:

- Η δήλωση **ALTER TABLE**: Η δήλωση ALTER TABLE χρησιμοποιείται για την τροποποίηση της δομής ενός υπάρχοντος πίνακα. Οι χρήστες μπορούν να προσθέσουν, να τροποποιήσουν ή να διαγράψουν στήλες αλλά και περιορισμούς ή δείκτες.

  Παράδειγμα

- Η δήλωση **DROP TABL**E: Η δήλωση DROP TABLE χρησιμοποιείται για την διαγραφή ενός πίνακα από τη βάση δεδομένων.

  Παράδειγμα διαγραφής ενός πίνακα:

2.4.4. Πρακτικές εφαρμογές της SQL

Η SQL χρησιμοποιείται εκτεταμένα σε διάφορους κλάδους και τομείς [20] όπως:

- Το Business Intelligence (BI): Η SQL αποτελεί τη βάση της επιχειρηματικής ευφυΐας και των αναλυτικών στοιχείων. Οι αναλυτές δεδομένων και άλλοι επαγγελματίες χρησιμοποιούν την SQL για να εξαγάγουν, να μετασχηματίσουν και να φορτώσουν δεδομένα, καθώς και για να δημιουργήσουν αναφορές και να αντλούν χρήσιμες πληροφορίες από δεδομένα.
- Το ηλεκτρονικό εμπόριο: Οι διαδικτυακοί έμποροι λιανικής βασίζονται σε βάσεις δεδομένων SQL για τη διαχείριση καταλόγων προϊόντων, παραγγελιών πελατών και του αποθεματικού τους. Η SQL επιτρέπει εξατομικευμένες εμπειρίες αγορών, αποτελεσματική επεξεργασία παραγγελιών και διαχείριση του αποθέματος.
- Την Υγεία: Οι οργανισμοί υγειονομικής περίθαλψης χρησιμοποιούν βάσεις δεδομένων SQL για να αποθηκεύουν αρχεία ασθενών, ιατρικά ιστορικά και θεραπευτικές αγωγές. Αυτές οι βάσεις δεδομένων βοηθούν στη λήψη κλινικών αποφάσεων, στη συμμόρφωση με τους κανονισμούς της υγειονομικής περίθαλψης και τη φροντίδα των ασθενών.
- Την Οικονομία: Τα χρηματοπιστωτικά ιδρύματα χρησιμοποιούν την SQL για το χειρισμό μεγάλων όγκων δεδομένων συναλλαγών, τον εντοπισμό δόλιων δραστηριοτήτων, τη δημιουργία οικονομικών αναφορών αλλά και τη διασφάλιση της ακρίβειας και της εγκυρότητας των οικονομικών αρχείων.
- Την Εκπαίδευση: Τα εκπαιδευτικά ιδρύματα χρησιμοποιούν βάσεις δεδομένων SQL για τη διαχείριση αρχείων μαθητών, καταλόγων μαθημάτων και δεδομένων βαθμολόγησης. Αυτές οι βάσεις δεδομένων διευκολύνουν τις ακαδημαϊκές λειτουργίες, την παρακολούθηση της προόδου των μαθητών και την δημιουργία αναφορών.

2.5. Ανταλλαγή δεδομένων και επικοινωνία μεταξύ συστημάτων

Στο συνεχώς εξελισσόμενο τοπίο της πληροφορικής, οι όροι «Web Service» και «API» αποτελούν βασικά στοιχεία της σύγχρονης ανάπτυξης λογισμικού και ανταλλαγής δεδομένων. Αυτές οι δύο έννοιες παίζουν κεντρικό ρόλο στην επικοινωνία και τη διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων, επιτρέποντάς τους να αλληλεπιδρούν και να συνεργάζονται απρόσκοπτα. Οι Υπηρεσίες Ιστού και τα API διαφέρουν ως προς τη λειτουργία, την υλοποίηση και το πεδίο εφαρμογής τους παρά το γεγονός ότι είναι έννοιες αλληλένδετες.

2.5.1. Υπηρεσίες Ιστού (Web Services)

Σύμφωνα με τη Κοινοπραξία του Παγκόσμιου Ιστού (World Wide Web Consortium - W3C) [21], οι Υπηρεσίες Ιστού ή αλλιώς Web Services είναι ένας ευρύς όρος που περιλαμβάνει ένα σύνολο από πρότυπα και πρωτόκολλα που διευκολύνουν την επικοινωνία

και την ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων μέσω του διαδικτύου ή ενός ιδιωτικού δικτύου. Ο πρωταρχικός στόχος των Web Services είναι να επιτρέψουν τη διαλειτουργικότητα (interoperability) μεταξύ διαφορετικών συστημάτων, ανεξάρτητα από τις υποκείμενες τεχνολογίες, τις γλώσσες προγραμματισμού ή τις πλατφόρμες τους. Υπάρχουν διάφοροι τύποι Web Services, ωστόσο οι δύο πιο ευρέως διαδεδομένοι είναι οι εξής:

1. **SOAP Web Services**: Πρόκειται για τα Web Services που χρησιμοποιούν το Simple Object Access Protocol (SOAP) ως το κύριο πρωτόκολλο επικοινωνίας τους. Το SOAP είναι ένα πρωτόκολλο ανταλλαγής δομημένων πληροφοριών στη μορφή μηνυμάτων. Χρησιμοποιεί την XML (Extensible Markup Language) για τη μορφοποίηση μηνυμάτων και βασίζεται συνήθως στο HTTP ή σε άλλα πρωτόκολλα για την μετάδοση τους. Τα SOAP Web Services είναι γνωστά για τα αυστηρά τους πρότυπα, την ασφάλεια και την πολυπλοκότητα τους, και χρησιμοποιούνται συχνά για την επικοινωνία μεταξύ επιχειρήσεων (B2B communication) και σε εφαρμογές εταιρικού επιπέδου.
2. **RESTful Web Services**: Αυτά τα Web Services χρησιμοποιούν την αρχιτεκτονική REST (Representational State Transfer) για την επικοινωνία με άλλα συστήματα, η οποία βασίζεται στην χρήση των τυπικών μεθόδων του πρωτοκόλλου HTTP (GET, POST, PUT, DELETE) για την εκτέλεση λειτουργιών και συνήθως χρησιμοποιούν απλές και ευέλικτές μορφές για την μορφοποίηση των δεδομένων όπως για παράδειγμα το JSON (JavaScript Object Notation). Τα RESTful Web Services είναι δημοφιλή λόγω της απλότητας, την επεκτασιμότητας και την ευκολίας χρήσης τους έναντι των SOAP Web Services, και χρησιμοποιούνται συχνά στην ανάπτυξη εφαρμογών web ή κινητών συσκευών.

2.5.2. Διεπαφές Προγραμματισμού Εφαρμογών (APIs)

Μία Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface – API), είναι ένα σύνολο κανόνων, πρωτοκόλλων και εργαλείων που επιτρέπουν σε διαφορετικές εφαρμογές λογισμικού να επικοινωνούν μεταξύ τους [22]. Καθορίζει τις μεθόδους και τις μορφές δεδομένων που μπορούν να χρησιμοποιήσουν οι προγραμματιστές για να αλληλοεπιδρούν με ένα συγκεκριμένο στοιχείο λογισμικού, υπηρεσία ή σύστημα, χωρίς να χρειάζεται να κατανοήσουν την εσωτερική λειτουργία αυτού του στοιχείου.

Τα APIs μπορούν να χαρακτηριστούν ως μια γέφυρα που επιτρέπει σε δύο διαφορετικά εφαρμογές λογισμικού να ανταλλάσσουν πληροφορίες και λειτουργικότητα απρόσκοπτά. Χρησιμοποιούνται κυρίως για να ζητούν και να ανταλλάσσουν δεδομένα, να εκτελούν συγκεκριμένες λειτουργίες ή να έχουν πρόσβαση σε διάφορες δυνατότητες ενός συστήματος, καθιστώντας τα ένα θεμελιώδες στοιχείο της σύγχρονης ανάπτυξης λογισμικού. Τα APIs χρησιμοποιούνται σε ένα ευρύ φάσμα εφαρμογών, από την ανάπτυξη εφαρμογών web και κινητών συσκευών έως τη διασύνδεση διαφορετικών συστημάτων. Παράλληλα, προωθούν την διαλειτουργικότητα και τη δημιουργία πάνω σε υπάρχοντα

στοιχεία λογισμικού, επιτρέποντας στους προγραμματιστές να αξιοποιήσουν τις δυνατότητες άλλων συστημάτων χωρίς να χρειάζεται να εφεύρουν εκ νέου τον τροχό.

Παρότι τα APIs έπαιξαν καθοριστικό ρόλο στην ανάπτυξη λογισμικού, η εμφάνιση των Web APIs αποτέλεσε σημείο καμπής. Τα Web APIs ουσιαστικά είναι ένα είδος API που χρησιμοποιούν δικτυακά πρωτόκολλα όπως το HTTP για να εκθέσουν τις λειτουργίες τους μέσω του Διαδικτύου. Αυτή η καινοτομία επέτρεψε στους προγραμματιστές να έχουν πρόσβαση σε δεδομένα και υπηρεσίες, επιτρέποντας στους προγραμματιστές να αξιοποιήσουν τη δύναμη των απομακρυσμένων συστημάτων με ευκολία. Τα Web APIs βοήθησαν σε μεγάλο βαθμό την ανάπτυξη δικτυακών εφαρμογών που βασίζονται στην έννοια του "mashup". Οι προγραμματιστές έχουν πλέον την δυνατότητα να συνδυάσουν δεδομένα και υπηρεσίες από διάφορες πηγές για να δημιουργήσουν νέες, καινοτόμες εφαρμογές. Για παράδειγμα, το Google Maps API έδωσε τη δυνατότητα στους προγραμματιστές να ενσωματώσουν χάρτες στις εφαρμογές τους, μεταμορφώνοντας τον τρόπο πλοήγησης και αλληλεπίδρασης με υπηρεσίες που βασίζονται στην τοποθεσία.

2.5.3. Διαφορές μεταξύ Web Service και API

Παρότι τα Web Services και τα APIs μπορούν να διευκολύνουν τη μεταφορά δεδομένων μεταξύ συστημάτων μέσω του Διαδικτύου, δεν είναι το ίδιο, και οι όροι δεν πρέπει να χρησιμοποιούνται ως ταυτόσημοι. Η κυριότερη διαφορά τους αποτυπώνεται στην εξής πρόταση [23]:

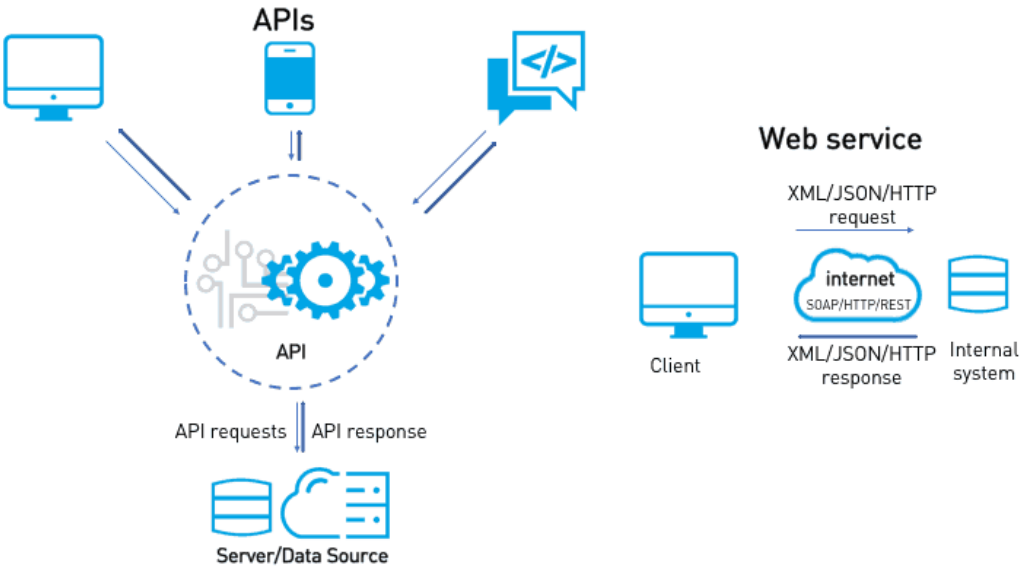
«Όλα τα Web Services είναι APIs, αλλά όχι όλα τα APIs είναι Web services.»

Πέραν τούτου υπάρχουν μερικές βασικές διαφορές μεταξύ Web Service και API [24] όπως:

1. **Πεδίο εφαρμογής**: Τα Web Services έχουν σχεδιαστεί κυρίως για απομακρυσμένη αλληλεπίδραση και ανταλλαγή δεδομένων μεταξύ διαφορετικών συστημάτων. Χρησιμοποιούνται κυρίως σε μεγάλης κλίμακας εταιρικών εφαρμογών και στην επικοινωνία B2B. Τα APIs, από την άλλη πλευρά, αναφέρονται συγκεκριμένα σε διεπαφές που επιτρέπουν την επικοινωνία μεταξύ εφαρμογών λογισμικού. Τα Web API, ειδικότερα, εστιάζουν σε αλληλεπιδράσεις μέσω του διαδικτύου και χρησιμοποιούνται συνήθως στην ανάπτυξη εφαρμογών web και κινητών συσκευών αλλά και σε υπηρεσίες cloud.
2. **Πρωτόκολλα επικοινωνίας**: Τα Web Services βασίζονται στην ύπαρξη ενός δικτύου και σε καθιερωμένα πρωτόκολλα επικοινωνίας, συμπεριλαμβανομένων των SOAP, HTTP, SMTP και άλλων. Αντιθέτως, τα APIs δεν συνδέονται με ένα συγκεκριμένο πρωτόκολλο επικοινωνίας. Μπορούν να χρησιμοποιηθούν για επικοινωνία μέσω του διαδικτύου (Web APIs), εσωτερικά σε μια εφαρμογή, ή ακόμα και μεταξύ στοιχείων υλικού.
3. **Μορφή μηνύματος**: Τα Web Services, ιδιαίτερα αυτά που βασίζονται στο SOAP, είναι γνωστά για την αυστηρή δομή μηνυμάτων και τη μορφή δεδομένων XML, η οποία μπορεί να είναι πιο περίπλοκη. Τα RESTful Web Services, από την άλλη πλευρά, επιτρέπουν

μεγαλύτερη ευελιξία υποστηρίζοντας πολλαπλές μορφές δεδομένων, συμπεριλαμβανομένων των JSON και XML, και γενικά θεωρούνται απλούστερες στην εργασία.. Τα APIs δεν ορίζουν μια συγκεκριμένη μορφή δεδομένων, επιτρέποντας στους προγραμματιστές να επιλέξουν τη μορφή που ταιριάζει καλύτερα στις ανάγκες τους. Αυτή η ευελιξία μπορεί να κάνει τα API πιο «ελαφριά» και πιο εύχρηστα για συγκεκριμένες εργασίες.

4. **Ασφάλεια**: Τα Web services διαθέτουν ενσωματωμένες δυνατότητες ασφαλείας, όπως το WS-Security για υπηρεσίες που βασίζονται στο SOAP, οι οποίες παρέχουν ασφάλεια και κρυπτογράφηση των μηνυμάτων. Ωστόσο, η παραμετροποίηση και η εφαρμογή της ασφάλειας στις υπηρεσίες ιστού μπορεί να είναι περίπλοκη. Αντίθετα, η ασφάλεια των APIs εξαρτάται από την υλοποίηση. Για παράδειγμα τα Web APIs, χρησιμοποιούν συνήθως τυπικές πρακτικές ασφάλειας δικτύων, όπως το HTTPS για κρυπτογράφηση δεδομένων και ταυτοποίηση χρησιμοποιώντας tokens ή API keys. Η απλότητα των API μπορεί να κάνει την εφαρμογή της ασφάλειας πιο εύκολη .
5. **Προσβασιμότητα**: Τα Web services περιλαμβάνουν μεταδεδομένα (metadata) και τεκμηρίωση που βοηθούν τους clients να ανακαλύψουν τις λειτουργίες και πως χρησιμοποιούνται. Η WSDL (Web Services Description Language) χρησιμοποιείται συνήθως για αυτόν τον σκοπό σε υπηρεσίες που βασίζονται στο SOAP. Τα APIs μπορεί να παρέχουν ή να μην παρέχουν λεπτομερή τεκμηρίωση και η δυνατότητα ανακάλυψής τους μπορεί να ποικίλλει πολύ. Ενώ ορισμένα APIs συνοδεύονται από εκτενή τεκμηρίωση, άλλα μπορεί να απαιτούν περισσότερη προσπάθεια για να κατανοηθούν και να χρησιμοποιηθούν αποτελεσματικά.



Εικόνα 2.12 – Ανταλλαγή δεδομένων μέσω APIs & Web Services

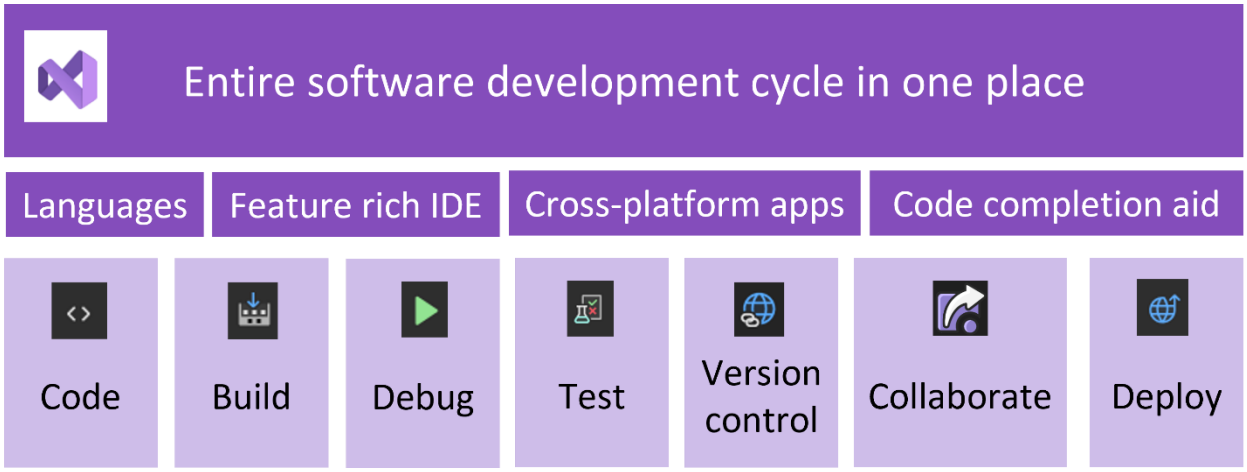
Κεφάλαιο 3 – Μεθοδολογία

Σε αυτό το κεφάλαιο περιγράφεται η γενική μεθοδολογία που ακολουθήθηκε κατά την ανάπτυξη του πληροφοριακού συστήματος, όπως τα εργαλεία ανάπτυξης και οι επιπρόσθετες τεχνολογίες που χρησιμοποιήθηκαν, καθώς και τα σχεδιαστικά πρότυπα και οι βέλτιστές πρακτικές που εφαρμόστηκαν.

3.1. Εργαλεία Ανάπτυξης

3.1.1. Visual Studio

Για την ανάπτυξη του κώδικα της εφαρμογής χρησιμοποιήθηκε το Visual Studio 2022. Το Visual Studio είναι ένα ισχυρό εργαλείο για προγραμματιστές που μπορεί να χρησιμοποιηθεί για ολόκληρο τον κύκλο ανάπτυξης ενός έργου [25]. Είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) το οποίο χρησιμοποιούν οι προγραμματιστές για τη συγγραφή και επεξεργασία κώδικα, τον εντοπισμό λαθών (debugging), τη δημιουργία εκτελέσιμου κώδικα (build) και της δημοσίευση (deployment) των εφαρμογών. Υποστηρίζει πολλαπλές γλώσσες προγραμματισμού και μπορεί να χρησιμοποιηθεί για την ανάπτυξη εφαρμογών κάθε τύπου.

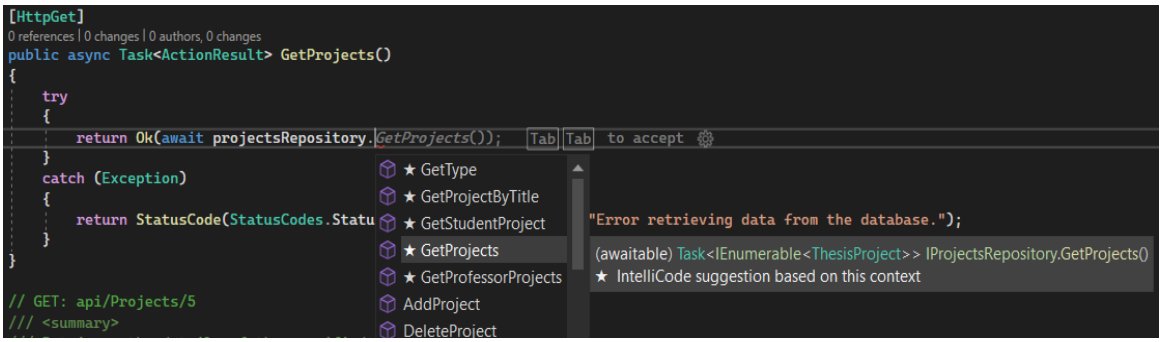


Εικόνα 3.1 – Δυνατότητες του Visual Studio

Μερικά από τα σημαντικότερα χαρακτηριστικά του Visual Studio που αξιοποιήθηκαν κατά την ανάπτυξη του κώδικα της εφαρμογής είναι:

- **Code Editor & IntelliSense**: Το Visual Studio προσφέρει έναν ισχυρό επεξεργαστή κώδικα με πολλές δυνατότητες χάρις την λειτουργία IntelliSense. Το IntelliSense είναι μια έξυπνη λειτουργία που βελτιώνει σημαντικά την εμπειρία ανάπτυξης κώδικα για τους προγραμματιστές. Πρόκειται για ένα έξυπνο βοηθό ανάπτυξης και αυτόματης συμπλήρωσης κώδικα που παρέχει χρήσιμες προτάσεις και πληροφορίες. Για παράδειγμα, καθώς πληκτρολογούμε κώδικα όπως φαίνεται στην Εικόνα 3.2, το IntelliSense προβλέπει τι προσπαθούμε να πληκτρολογήσουμε και προσφέρει

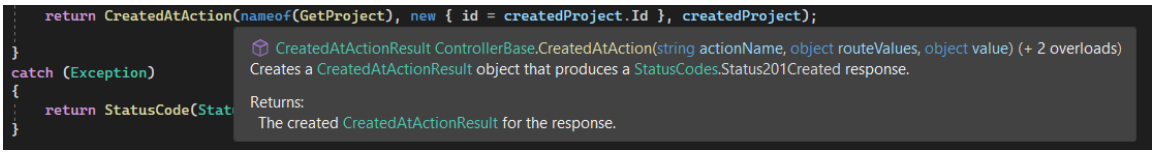
προτάσεις για τη συμπλήρωση του κώδικα. Αυτό περιλαμβάνει την πρόταση ονομάτων μεταβλητών, ονομάτων μεθόδων, ονομάτων κλάσεων και πολλά άλλα.



Εικόνα 3.2 – Παράδειγμα αυτόματης συμπλήρωσης και συμβουλής του IntelliSense

Το IntelliSense αντιλαμβάνεται το γενικό πλαίσιο του κώδικά και προσφέρει προτάσεις που σχετίζονται με τη γλώσσα προγραμματισμού και τα διάφορα frameworks που χρησιμοποιούνται. Λαμβάνει υπόψη τις κλάσεις, τις μεθόδους, τις ιδιότητες και τις μεταβλητές που είναι διαθέσιμες στον κώδικά, καθώς και τυχόν εισαγόμενες βιβλιοθήκες.

Επίσης, το IntelliSense διαθέτει συμβουλές εργαλείων (tooltips). Όταν τοποθετείτε ο δείκτης του ποντικιού πάνω από μια κλάση, μέθοδο, μεταβλητή ή οποιοδήποτε στοιχείο κώδικα (Εικόνα 3.3), το IntelliSense εμφανίζει μια επεξήγηση εργαλείου με πρόσθετες πληροφορίες, όπως τις υπογραφές των υπερφορτωμένων μεθόδων, ποια είναι η έξοδος που επιστρέφεται, αλλά και τεκμηρίωση ή παραδείγματα χρήσης.



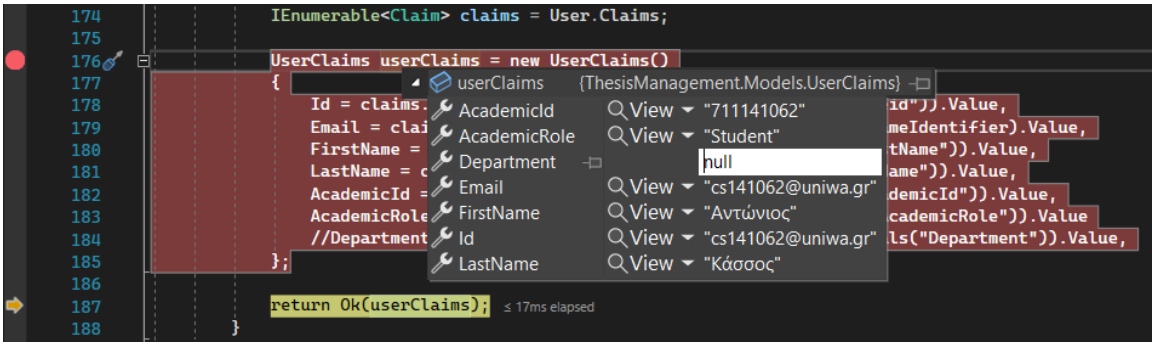
Εικόνα 3.3 – Παράδειγμα ενός tooltip του IntelliSense

Μια ακόμα πολύ χρήσιμη δυνατότητα, είναι η επισήμανση σημείων του κώδικα που επιδέχονται διόρθωση για προληπτική αποτροπή σφαλμάτων ή εναλλακτικής συγγραφής για την εφαρμογή των βέλτιστων πρακτικών.

Επιπλέον, το IntelliSense επιπλέον, εύκολη αναζήτηση και αποτελεσματική πλοήγηση στο κώδικα ενός project. Για παράδειγμα, μπορούμε εύκολα να μεταβούμε γρήγορα στον ορισμό μιας κλάσης ή μιας μεθόδου, ή να αναζητήσουμε και να αντικαταστήσουμε σημεία του κώδικα μέσω του “Find & Replace”.

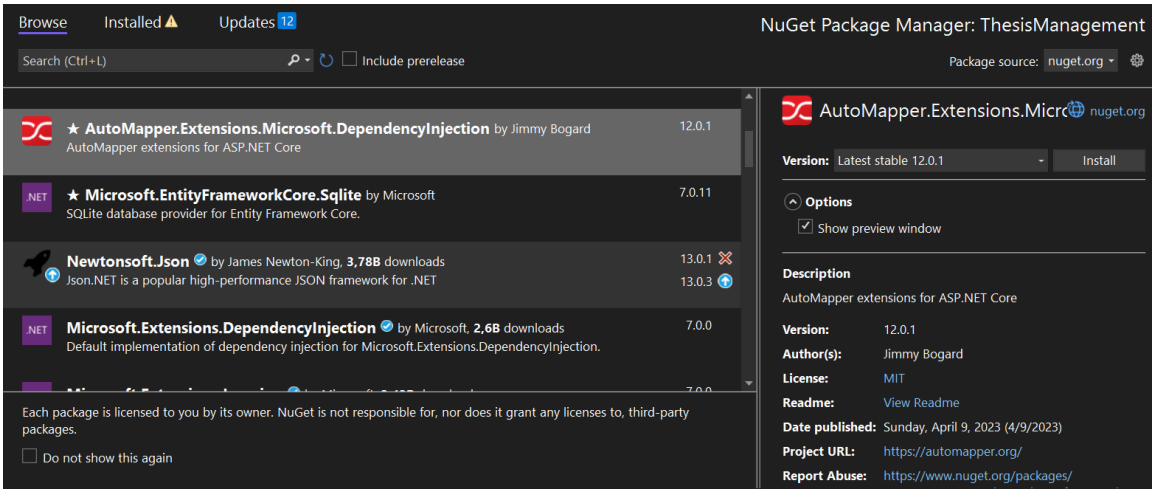
- **Εργαλεία Debugging**: Τα ισχυρά εργαλεία εντοπισμού σφαλμάτων που διαθέτει το Visual Studio, επιτρέπουν στους προγραμματιστές να εντοπίζουν και να διορθώνουν γρήγορα και εύκολα σφάλματα στον κώδικά τους.

  Αυτά τα εργαλεία περιλαμβάνουν σημεία διακοπής (breakpoints), παρατηρητές (watchers) και ενδιάμεσο εντοπισμό σφαλμάτων μέσω της ελεγχόμενης εκτέλεσης του κώδικα βήμα προς βήμα (Εικόνα 3.4).



Εικόνα 3.4 – Παράδειγμα debugging εργαλείων του Visual Studio

- **NuGet Package Manager**: Οι προγραμματιστές μπορούν εύκολα να εγκαθιστούν και να διαχειρίζονται βιβλιοθήκες και plugins τρίτων, χρησιμοποιώντας τον διαχειριστή πακέτων NuGet που είναι ενσωματωμένος στο Visual Studio (Εικόνα 3.5).



Εικόνα 3.5 – Εγκατάσταση βιβλιοθηκών μέσω του Nuget Package Manager

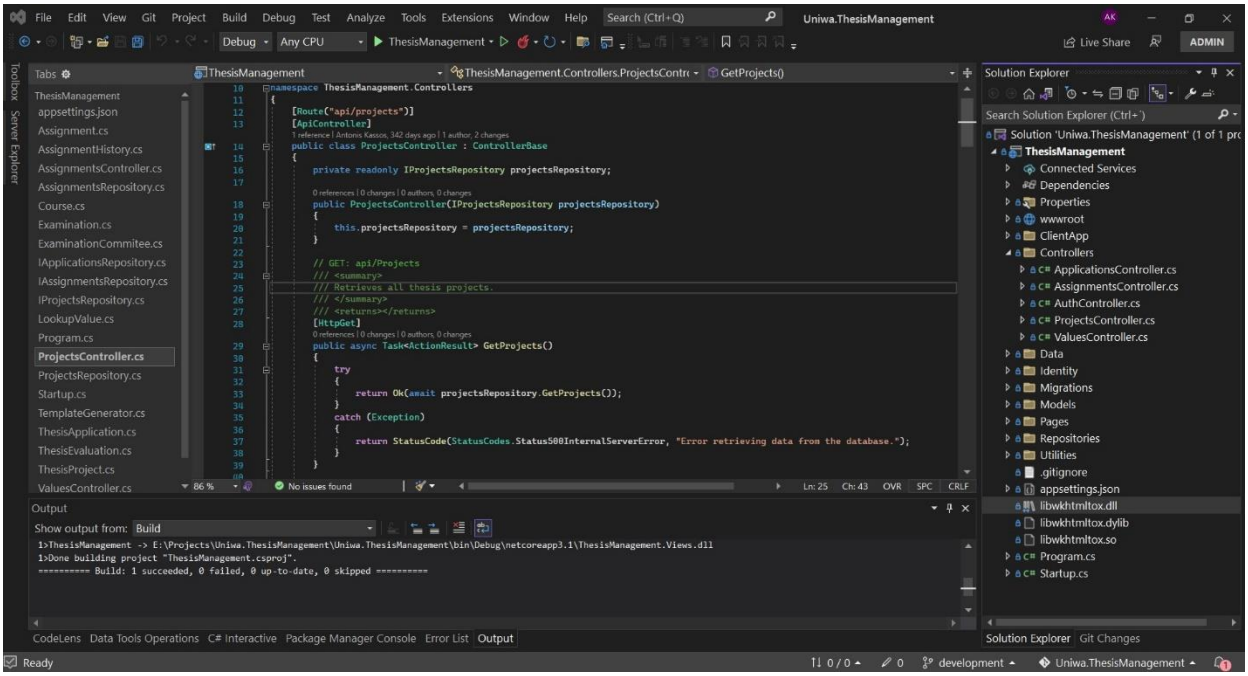
- **Version & Source Control**: Το Visual Studio διαθέτει ενσωματωμένη υποστήριξη του Git, επιτρέποντας στους προγραμματιστές να διαχειρίζονται αποθετήρια πηγαίου κώδικα (repositories), να πραγματοποιούν διακλαδώσεις (branches) και συγχωνεύσεις (merge), και να συνεργάζονται μεταξύ τους.
- **Επεκτάσεις**: Το Visual Studio υποστηρίζει ένα πλούσιο οικοσύστημα επεκτάσεων και plugins που διατίθενται μέσω του Visual Studio Marketplace, ενισχύοντας τη λειτουργικότητα και τη παραγωγικότητα κατά την φάση ανάπτυξη.

Αξίζει να σημειωθεί πως το Visual Studio διατίθεται σε τρεις διαφορετικές εκδόσεις, όπου η κάθε μία διαθέτει επιπλέον δυνατότητες και λειτουργίες:

- **Visual Studio Community**: Δωρεάν έκδοση για μεμονωμένους προγραμματιστές, κατάλληλο για προσωπική χρήση και μικρές ομάδες.

- **Visual Studio Professional**: Σχεδιασμένο για επαγγελματίες προγραμματιστές και για μικρές έως μεσαίες ομάδες. Επιλέχτηκε η συγκεκριμένη έκδοση για την ανάπτυξη της εφαρμογής της διπλωματικής εργασίας, κυρίως λόγω του ενσωματωμένου Git.
- **Visual Studio Enterprise**: Απευθύνεται σε μεγαλύτερους οργανισμούς με προηγμένες ανάγκες ανάπτυξης και συνεργασίας.

Στην Εικόνα 3.6, φαίνεται το πλήρες περιβάλλον ανάπτυξης του Visual Studio 2022.



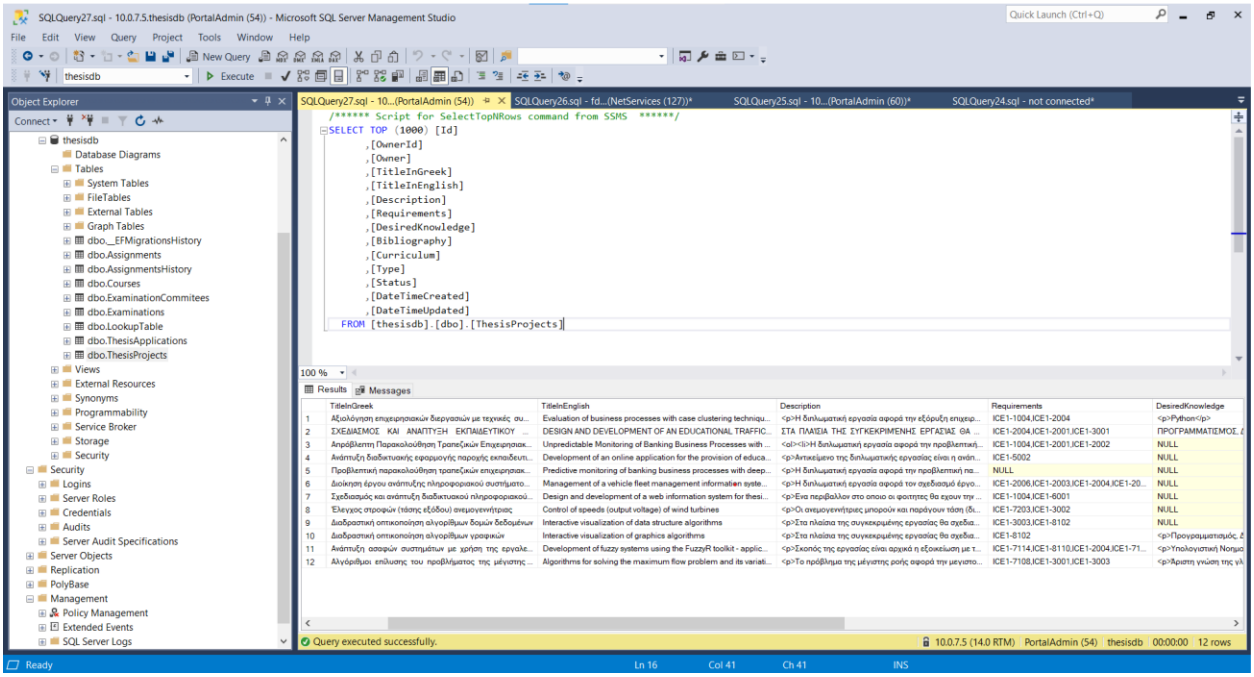
Εικόνα 3.6 – Το περιβάλλον του Visual Studio 2022

Συνοψίζοντας, το Visual Studio είναι ένα ευέλικτο και πλούσιο σε δυνατότητες IDE που επιτρέπει στους προγραμματιστές να αναπτύσσουν ένα ευρύ φάσμα εφαρμογών λογισμικού, από desktop εφαρμογές και εφαρμογές web έως εφαρμογές για κινητά και υπηρεσίες στο cloud. Η ενσωμάτωσή του με τις τεχνολογίες της Microsoft και τις υπηρεσίες cloud το καθιστά πολύτιμο εργαλείο για προγραμματιστές που εργάζονται στο οικοσύστημα της Microsoft.

3.1.2. SQL Server Management Studio

Η δημιουργία και η διαχείριση της βάσης δεδομένων του πληροφοριακού συστήματος που αναπτύχθηκε, πραγματοποιήθηκε μέσω του SQL Server Management Studio (Εικόνα 3.7). Το SQL Server Management Studio (SSMS) είναι μια εφαρμογή λογισμικού που αναπτύχθηκε από τη Microsoft και χρησιμοποιείται για τη διαχείριση βάσεων δεδομένων Microsoft SQL [26]. Παρέχει μια γραφική διεπαφή χρήστη (GUI) για την αλληλεπίδραση με τον SQL Server και την εκτέλεση διαφόρων εργασιών που σχετίζονται με τις βάσεις δεδομένων. Το SQL Server Management Studio χρησιμοποιείται συνήθως από διαχειριστές

βάσεων δεδομένων, προγραμματιστές και άλλους επαγγελματίες που εργάζονται πάνω σε βάσεις δεδομένων του SQL Server.



Εικόνα 3.7 – Το περιβάλλον του SQL Server Management Studio

Ακολουθούν ορισμένες από τις βασικές δυνατότητες και λειτουργίες του SQL Server Management Studio:

- Ανάπτυξη βάσεων δεδομένων: Το SSMS επιτρέπει τη δημιουργία, τη τροποποίηση και τη διαχείριση στοιχείων που αφορούν τις βάσεις δεδομένων, όπως πίνακες, αποθηκευμένες διαδικασίες (stored procedures) και συναρτήσεις.
- Εκτέλεση ερωτημάτων (queries): Μέσω του SSMS οι χρήστες μπορούν να συντάσσουν και να εκτελούν ερωτήματα ή σενάρια SQL. Παρέχει ένα πλήρες πρόγραμμα επεξεργασίας ερωτημάτων με δυνατότητες επισήμανσης σύνταξης και δημιουργίας έτοιμων βασικών ερωτημάτων SQL.
- Διαχείριση διακομιστή (server): Επιτρέπει τη δημιουργία τοπικής βάσης δεδομένων μέσω του τοπικού διακομιστή SQLExpress, τη παραμετροποίηση των ρυθμίσεων διακομιστή, καθώς και την εκτέλεση εργασιών όπως η δημιουργία αντιγράφων ασφαλείας, η επαναφορά και η διαχείριση των χρηστών.
- Εισαγωγή και εξαγωγή δεδομένων: Μέσω του SSMS οι χρήστες μπορούν να εισάγουν δεδομένα από διάφορες πηγές στις βάσεις δεδομένων του SQL Server και να εξάγουν δεδομένα από τον SQL Server σε διαφορετικές μορφές αρχείων.
- Δημιουργία ροών δεδομένων: Υποστηρίζει τα SQL Server Integration Services (SSIS) για το σχεδιασμό και τη δημιουργία ροών δεδομένων και διαδικασιών ETL (Extract, Transform, Load).

- Παρακολούθηση απόδοσης: Το SSMS παρέχει εργαλεία για την παρακολούθηση και τη βελτιστοποίηση της απόδοσης των βάσεων δεδομένων, όπως τη δυνατότητα προβολής σχεδιαγραμμάτων εκτέλεσης, τη παρακολούθηση της χρήσης των διαθέσιμων πόρων αλλά και τη διάγνωση προβλημάτων απόδοσης.

Συνοψίζοντας, το SQL Server Management Studio είναι ένα ισχυρό εργαλείο που απλοποιεί τη διαχείριση και την ανάπτυξη βάσεων δεδομένων SQL Server. Διατίθεται δωρεάν από την Microsoft και αποτελεί ένα απαραίτητο εργαλείο για όποιον εργάζεται με Microsoft SQL βάσεις δεδομένων.

3.2. Σύστημα Διαχείρισης Εκδόσεων

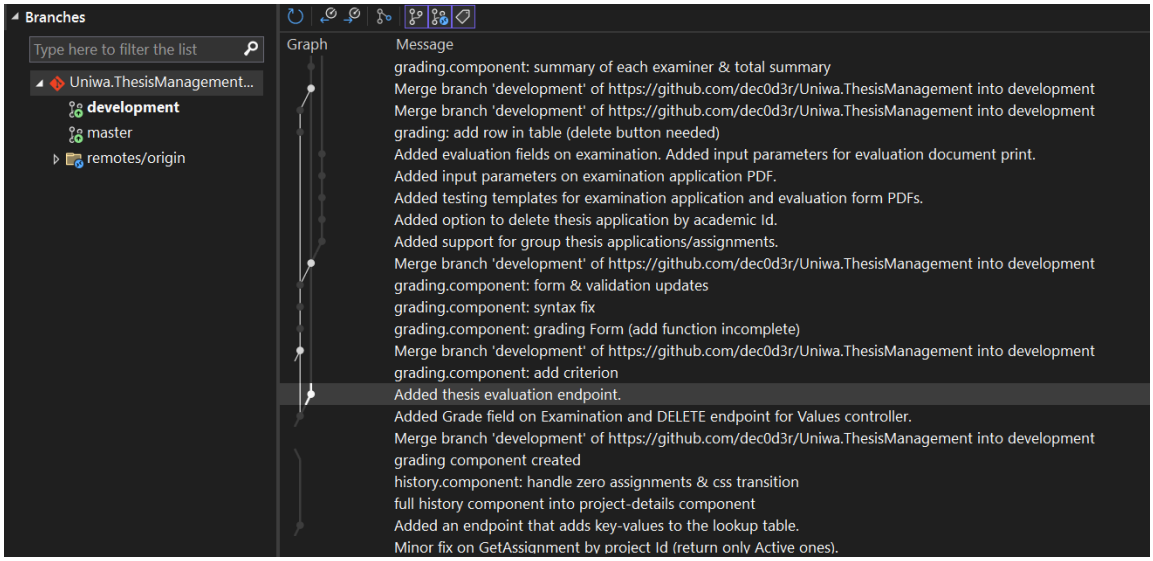
Ο βασικός λόγος ύπαρξης των συστημάτων διαχείρισης εκδόσεων (version control systems) κατά την ανάπτυξη ενός έργου, είναι η παρακολούθηση και η διαχείριση των τροποποιήσεων του κώδικα. Στο κόσμο των προγραμματιστών, είναι γνωστό πως η διόρθωση ή οι αλλαγές σε κομμάτια του κύριου πηγαίου κώδικα μιας εφαρμογής εγκυμονεί κινδύνους. Επίσης, ένα σύνηθες φαινόμενο, κυρίως κατά την υλοποίηση μεγάλων έργων, είναι ο διαχωρισμός των εργασιών, με σκοπό την παράλληλη ανάπτυξη κώδικα από μια ομάδα προγραμματιστών. Την λύση σε αυτά τα ζητήματα παρέχει το version control με τη δημιουργία διακλαδώσεων (branches) [27]. Οι διακλαδώσεις αντιγράφουν μέρος του πηγαίου κώδικα επιτρέποντας την τροποποίηση του, χωρίς να επηρεαστεί την κύρια πηγή από την οποία δημιουργήθηκαν. Κατά την υλοποίηση μίας εργασίας που ορίστηκε για το εκάστοτε branch, ο προγραμματιστής έχει τη δυνατότητα να μεταφέρει τις αλλαγές του στον κύριο πηγαίο κώδικα – ή σε κάποια άλλη διακλάδωση που επιθυμεί – εκτελώντας μία συγχώνευση (merge).

3.2.1. Το σύστημα Git

Το Git είναι ένα κατανεμημένο σύστημα διαχείρισης εκδόσεων (Distributed Version Control System - DVCS) που χρησιμοποιείται για την παρακολούθηση και διαχείριση των αλλαγών στον πηγαίο κώδικα κατά την ανάπτυξη λογισμικού. Δημιουργήθηκε από τον Linus Torvalds το 2005 και σήμερα αποτελεί το πιο ευρέως διαδεδομένο σύστημα διαχείρισης εκδόσεων στον κόσμο [27]. Το Git είναι ένα θεμελιώδες εργαλείο για τη συνεργασία μεταξύ των προγραμματιστών, επιτρέποντας σε πολλούς ανθρώπους να εργάζονται στο ίδιο έργο παράλληλα.

Το Git διατηρεί ένα ιστορικό όλων των αλλαγών, δίνοντας της δυνατότητα επαναφορά κώδικα σε περίπτωση κάποιας δυσλειτουργίας. Υποστηρίζει λειτουργίες όπως οι διακλαδώσεις και οι συγχωνεύσεις, ενώ εισάγει την έννοια των καταχωρήσεων (commits). Ένα commit, είναι ένα στιγμιότυπο του κώδικα σε μια συγκεκριμένη χρονική στιγμή. Οι προγραμματιστές δημιουργούν commits για να καταγράφουν τις αλλαγές που έχουν κάνει στη βάση του κώδικα. Κάθε commit έχει ένα μοναδικό αναγνωριστικό και το σύνολο τους, σχηματίζει ένα χρονοδιάγραμμα της ιστορίας του έργου.

Το Visual Studio που χρησιμοποιήθηκε για την ανάπτυξη του κώδικα στα πλαίσια της Διπλωματικής εργασίας, διαθέτει ενσωματωμένο Git, το οποίο μας επιτρέπει να εκτελούμε εύκολα λειτουργίες όπως τη δημιουργία διακλαδώσεων, τη συγχώνευση και τη καταχώρηση αλλαγών, σε ένα συνδεδεμένο αποθετήριο κώδικα, απευθείας από το περιβάλλον του Visual Studio. Για τη δημιουργία του αποθετήριού κώδικα της εφαρμογής, χρησιμοποιήθηκε η πλατφόρμα GitHub. Στην Εικόνα 3.8, φαίνεται ένα μέρος του ιστορικού των commits του κώδικα της εφαρμογής, μέσα από το Visual Studio.

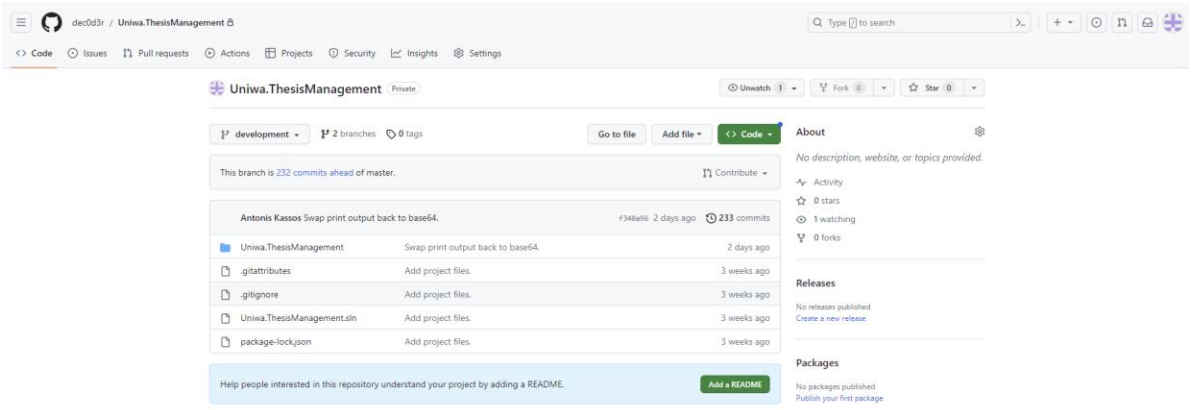


Εικόνα 3.8 – Ιστορικό τροποποιήσεων του κώδικα μέσω του Visual Studio

3.2.2. Η πλατφόρμα GitHub

Το GitHub είναι μια διαδικτυακή πλατφόρμα για τη φιλοξενία και τη διαχείριση αποθετηρίων Git. Παρέχει ένα φιλικό περιβάλλον για προγραμματιστές μέσω του οποίου μπορούν να συνεργάζονται πάνω σε έργα λογισμικού και υποστηρίζει δυνατότητες διαμοιρασμού κώδικα, παρακολούθησης αλλαγών, διαχείρισης προβλημάτων και πολλά άλλα. Οι χρήστες μπορούν να δημιουργούν δωρεάν λογαριασμούς στον ιστότοπο του GitHub και να ανεβάζουν τον πηγαίο κώδικα από διάφορα έργα λογισμικού, δημιουργώντας αποθετήρια κώδικα (code repositories), τα οποία μπορεί να είναι ιδιωτικά ή διαθέσιμα προς το κοινό. Η λειτουργία του βασίζεται στις αρχές του Version Control και του Git.

Το GitHub αποτελεί μια από τις πιο δημοφιλείς πλατφόρμες διαχείρισης εκδόσεων και συνεργασίας στην κοινότητα των προγραμματιστών [28]. Συνεισφέρει σε μεγάλο βαθμό στην ανάπτυξη λογισμικού ανοιχτού κώδικα, παρέχοντας μια πλατφόρμα όπου οι προγραμματιστές μπορούν να συνεργάζονται, να μοιράζονται τη δουλειά τους και να συνεισφέρουν σε διάφορα έργα. Αποτελεί επίσης, ένα πολύτιμο εργαλείο για οργανισμούς και ομάδες που εργάζονται σε ιδιωτικά έργα, προσφέροντας τους έναν ασφαλή και αποτελεσματικό τρόπο διαχείρισης των έργων κατά τη φάση ανάπτυξης του λογισμικού τους. Ο αντίκτυπος του GitHub στην κοινότητα ανάπτυξης λογισμικού το έχει καταστήσει κεντρικό κόμβο για τη συνεργασία ανάπτυξης κώδικα και την διαχείριση εκδόσεων.



Εικόνα 3.9 – Αποθετήριο του κώδικα της εφαρμογής στη πλατφόρμα GitHub

3.3. Επιπρόσθετες τεχνολογίες που χρησιμοποιήθηκαν

3.3.1. Entity Framework Core

Το Entity Framework Core ή EF Core είναι ένας Object-Relational Mapper (ORM). Πρόκειται για ένα ελαφρύ, επεκτάσιμο και λογισμικό ανοιχτού κώδικα [29]. Όπως το .NET Core, έτσι το EF Core είναι επίσης cross-platform, δηλαδή λειτουργεί σε όλα τα λειτουργικά συστήματα (Windows, Mac OS και Linux). Το EF Core χρησιμοποιήθηκε στα πλαίσια της διπλωματικής εργασίας για την ανάπτυξη της βάσης δεδομένων του πληροφοριακού συστήματος.

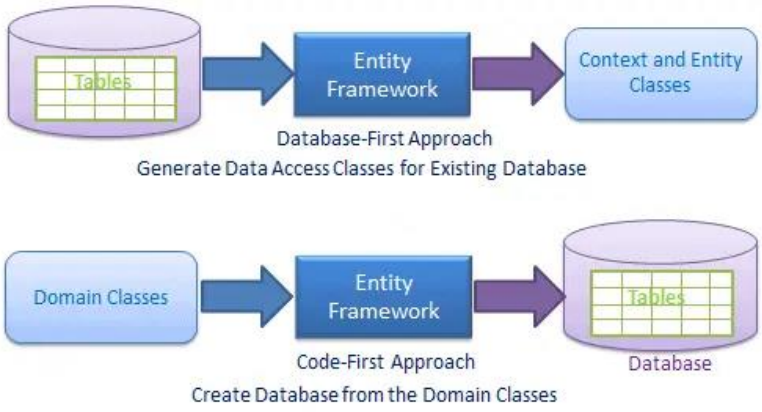
Ένας ORM παρέχει ένα σύνολο εργαλείων και βιβλιοθηκών στους προγραμματιστές για να εργάζονται πάνω σε βάσεις δεδομένων με αντικειμενοστραφή τρόπο. Το EF Core απλοποιεί τον προγραμματισμό της βάσης δεδομένων επιτρέποντας στους προγραμματιστές να αλληλεπιδρούν με αυτές, χρησιμοποιώντας κώδικα C# ή VB.NET αντί να γράφουν ερωτήματα SQL, εξαλείφοντας την ανάγκη για το μεγαλύτερο μέρος του κώδικα πρόσβασης δεδομένων που συνήθως χρειάζεται να γράψουν.

Αναλυτικότερα, το EF αντιστοιχίζει τους πίνακες βάσεων δεδομένων σε αντικείμενα .NET. Οι οντότητες (entities) περιγράφονται από κλάσεις C# ή VB.NET που αντιπροσωπεύουν πίνακες βάσης δεδομένων και οι ιδιότητες αυτών των κλάσεων αντιστοιχούν στις στήλες του πίνακα. Παράλληλα, οι σχέσεις μεταξύ οντοτήτων αντιπροσωπεύονται ως ιδιότητες πλοήγησης (navigation properties) μιας κλάσης, καθιστώντας εύκολη την εργασία με δεδομένα που συσχετίζονται μεταξύ τους.

Το EF Core υποστηρίζει δύο προσεγγίσεις ανάπτυξης (Εικόνα 3.8) [29]:

1. **Code-First προσέγγιση**: Σε αυτή τη προσέγγιση, οι προγραμματιστές ορίζουν πρώτα τις κλάσεις C# των οντοτήτων και στη συνέχεια το EF Core δημιουργεί τη βάση δεδομένων και τους πίνακες βάσει αυτών. Αυτή η μέθοδος χρησιμοποιείται συνήθως για νέα έργα. Στα πλαίσια της διπλωματικής εργασίας, ακολουθήθηκε η Code-First προσέγγιση.

2. **Database-First προσέγγιση**: Αντίστροφα, σε αυτή τη προσέγγιση το EF Core δημιουργεί τις κλάσεις οντοτήτων βάσει μιας υπάρχουσας βάσης δεδομένων. Αυτή η μέθοδος χρησιμοποιείται συνήθως για έργα που υπάρχει ήδη κάποια βάση δεδομένων. Ωστόσο, στην Core έκδοση του Entity Framework, η μέθοδος αυτή έχει περιορισμένη υποστήριξη.



Εικόνα 3.10 – Προσεγγίσεις Code-First & Database-First του Entity Framework

Στη περίπτωση της Code-First προσέγγισης, το EF Core υποστηρίζει τα λεγόμενα «Data Migrations». Τα migrations επιτρέπουν στους προγραμματιστές να κάνουν αλλαγές στις βάσεις δεδομένων, τροποποιώντας τις κλάσεις των οντοτήτων. Μπορούν να δημιουργούν, να ενημερώνουν, να διαγράφουν ή και να επαναφέρουν σχήματα μιας βάσης δεδομένων, διατηρώντας παράλληλα την ακεραιότητα των δεδομένων. Τα migrations εφαρμόζονται στη βάση δεδομένων μέσω εντολών και μεταφράζονται σε εντολές SQL αυτόματα. Για παράδειγμα, για τη δημιουργία ενός πίνακα μαθημάτων στη Β.Δ. του πληροφοριακού συστήματος που αναπτύχθηκε, ορίστηκε η κλάση «Course», όπως φαίνεται στην Εικόνα 3.9:

```
public class Course
{
    0 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }
    0 references | Antonis Kassos, 14 days ago | 1 author, 1 change
    public string Code { get; set; }
    0 references | Antonis Kassos, 14 days ago | 1 author, 1 change
    public string Title { get; set; }
    0 references | Antonis Kassos, 14 days ago | 1 author, 1 change
    public string Department { get; set; }
    0 references | Antonis Kassos, 14 days ago | 1 author, 1 change
    public string Curriculum { get; set; }
    0 references | Antonis Kassos, 14 days ago | 1 author, 1 change
    public string Period { get; set; }
    0 references | Antonis Kassos, 14 days ago | 1 author, 1 change
    public string Type { get; set; }
}
```

Εικόνα 3.11 – Η κλάση της οντότητας «Course»

```
public partial class AddCoursesTable : Migration
{
    0 references | 0 changes | 0 authors, 0 changes
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Courses",
            columns: table => new
            {
                Id = table.Column<int>(nullable: false)
                    .Annotation("SqlServer:Identity", "1, 1"),
                Code = table.Column<string>(nullable: true),
                Title = table.Column<string>(nullable: true),
                Department = table.Column<string>(nullable: true),
                Curriculum = table.Column<string>(nullable: true),
                Period = table.Column<string>(nullable: true),
                Type = table.Column<string>(nullable: true)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Courses", x => x.Id);
            });
    }

    0 references | 0 changes | 0 authors, 0 changes
    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropTable(
            name: "Courses");
    }
}
```

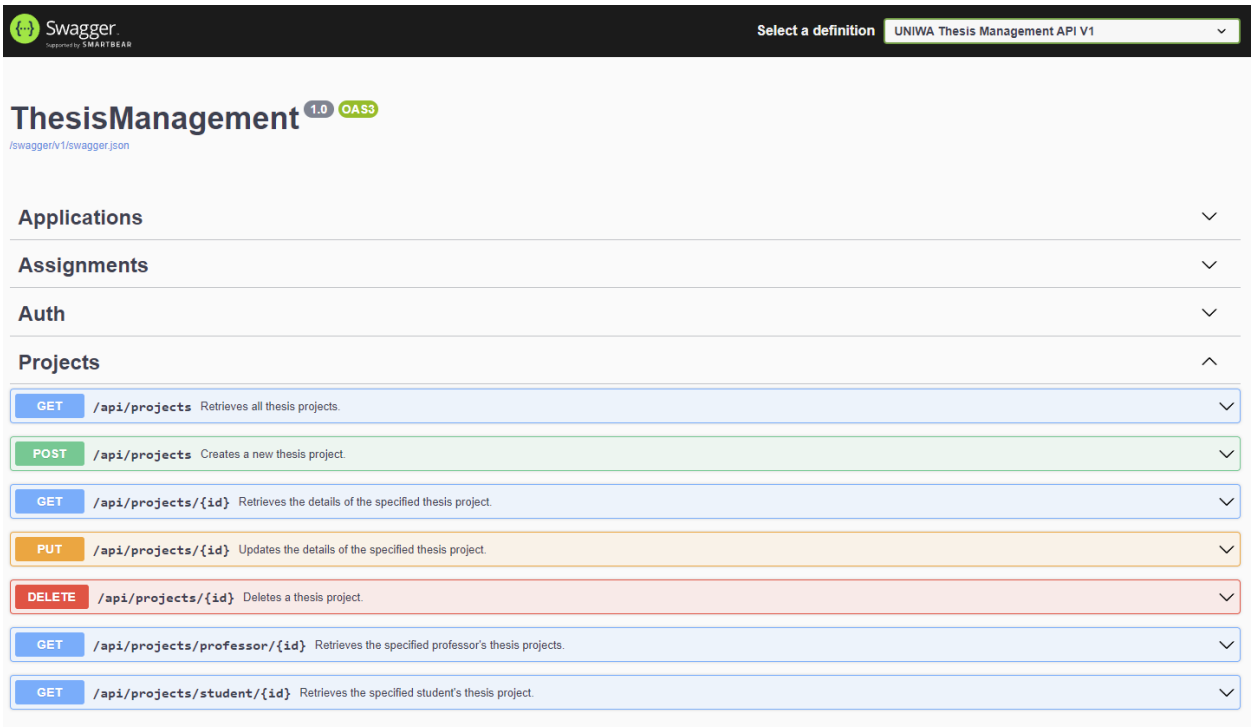
Εικόνα 3.12 – Κώδικας migration «AddCoursesTable»

Με το EF Core, εκτελώντας την εντολή ***'add-migration AddCoursesTable'***, δημιουργείται το migration που φαίνεται στην Εικόνα 3.10. Για την εφαρμογή του στη βάση δεδομένων, εκτελείται η εντολή '***update-database***', η οποία μεταφράζει το migration σε εντολές SQL και δημιουργεί τον πίνακα στη βάση δεδομένων αυτόματα. Κατά αυτό το τρόπο, μπορούμε να κάνουμε αλλαγές στη βάση δεδομένων εύκολα, γρήγορα και χωρίς τη χρήση της SQL.

Το EF Core υποστηρίζει διαφορετικών ειδών σχεσιακές αλλά και μη σχεσιακές βάσεις δεδομένων, χρησιμοποιώντας έτοιμες βιβλιοθήκες που ονομάζονται Database Providers. Συγκεκριμένα, υποστηρίζει πολλούς Database Providers, συμπεριλαμβανομένων των SQL Server, MySQL, SQLite , PostgreSQL και άλλων. Οι βιβλιοθήκες αυτές διατίθενται ως πακέτα μέσω του Nuget Package Manager.

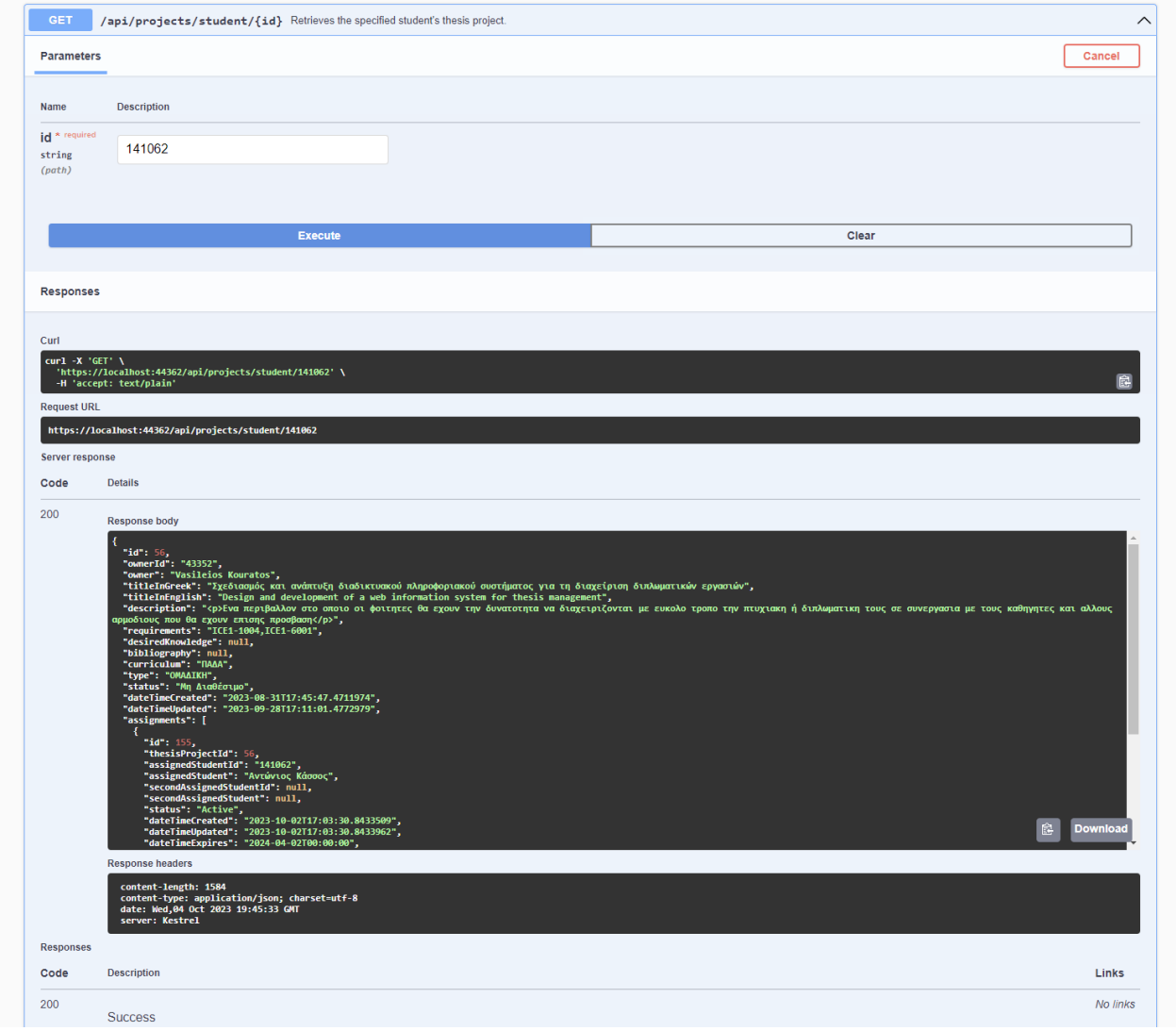
3.3.2. Swagger

Το Swagger είναι ένα δημοφιλές εργαλείο ανοιχτού κώδικα για την τεκμηρίωση και την δοκιμή των APIs [30]. Στο πλαίσιο του .NET Core, το Swagger χρησιμοποιείται συχνά για τη δημιουργία μιας διαδραστικής και φιλικής προς το χρήστη τεκμηρίωσης για ένα Web API. Επιτρέπει στους προγραμματιστές να οπτικοποιούν και να αλληλεπιδρούν με τις διάφορες κλήσεις και τις παραμέτρους των HTTP αιτημάτων και απαντήσεων του API, ενώ παρέχει έναν τρόπο δοκιμής των κλήσεων του API απευθείας μέσω μίας διαδραστικής διεπαφής. Στην Εικόνα 3.13, φαίνεται μέρος της τεκμηρίωσης του REST API που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας, μέσω της διεπαφής του Swagger.



Εικόνα 3.13 – Τεκμηρίωση REST API μέσω του Swagger

Το Swagger αυτόματα ένα αρχείο JSON (ή μερικές φορές YAML) για να περιγράψει τα τελικά σημεία (endpoints) του API, τα μοντέλα αιτημάτων/απαντήσεως και άλλα μεταδεδομένα. Αυτό το αρχείο JSON δημιουργείται με βάση τον κώδικα και τα σχόλια του API, και βάσει αυτού, το Swagger δημιουργεί τη διαδραστική διεπαφή χρήστη ή αλλίως Swagger UI. Στην Εικόνα 3.14, φαίνεται η εκτέλεση μιας δοκιμαστικής κλήσης από το REST API που αναπτύχθηκε μέσω του Swagger UI.



Εικόνα 3.14 – Εκτέλεση δοκιμαστικής κλήσης μέσω του Swagger UI

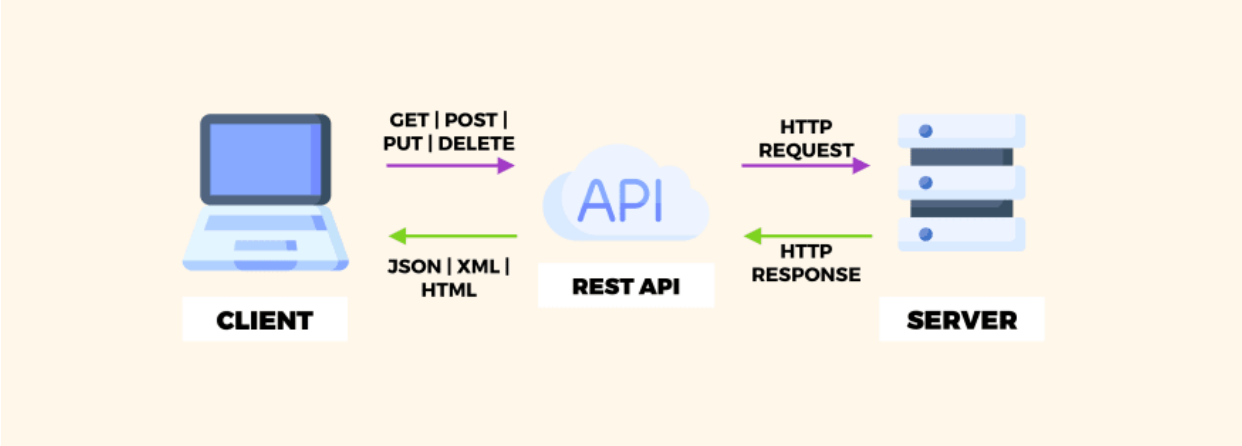
3.4. Σχεδιαστικά Πρότυπα & Βέλτιστες Πρακτικές

3.4.1. Η αρχιτεκτονική REST

Το Representational State Transfer, ή REST, είναι μια αρχιτεκτονική προσέγγιση που έχει παίξει καθοριστικό ρόλο στη διαμόρφωση του Παγκόσμιου Ιστού. Επινοήθηκε από τον Roy Fielding στη διδακτορική του διατριβή το 2000 [31], και έκτοτε το REST αποτελεί βασικό πυλώνα για το σχεδιασμό υπηρεσιών ιστού, προωθώντας τη διαλειτουργικότητα, την επεκτασιμότητα και την απλότητα στον ψηφιακό κόσμο. Το REST βασίζεται σε ένα σύνολο θεμελιωδών αρχών που καθοδηγούν το σχεδιασμό και την εφαρμογή του.

Η αρχιτεκτονική REST διαχωρίζει τον πελάτη και τον διακομιστή σε διακριτές οντότητες που μπορούν να αναπτυχθούν ανεξάρτητα. Αυτός ο διαχωρισμός προωθεί την τμηματική δομή και απλοποιεί την αρχιτεκτονική, καθώς κάθε στοιχείο έχει ένα καλά καθορισμένο σύνολο ευθυνών. Χρησιμοποιώντας μια διεπαφή REST, οι διαφορετικοί πελάτες καλούν τα ίδια τελικά σημεία (endpoints), εκτελούν τις ίδιες ενέργειες και λαμβάνουν τις ίδιες απαντήσεις.

Επιπλέον, κάθε αίτημα από έναν πελάτη προς έναν διακομιστή πρέπει να περιέχει όλες τις πληροφορίες που απαιτούνται για την κατανόηση και την επεξεργασία του αιτήματος. Ο διακομιστής δεν πρέπει να αποθηκεύει πληροφορίες σχετικά με την κατάσταση του πελάτη μεταξύ των αιτημάτων. Ο περιορισμός αυτός εξασφαλίζει πως στην επικοινωνία μεταξύ πελάτη-διακομιστή δεν λαμβάνεται υπόψιν η κατάσταση τους (stateless communication), ενισχύοντας την επεκτασιμότητα και την ανοχή σε σφάλματα, καθώς ο διακομιστής δεν χρειάζεται να διατηρεί συνεδριάσεις πελατών (client sessions).



Εικόνα 3.15 – Επικοινωνία client-server μέσω REST API

Μια ακόμα βασική αρχή του REST είναι η διατήρηση μιας ομοιόμορφης και συνεπής διεπαφής. Αυτό σημαίνει ότι οι πόροι προσδιορίζονται από URIs (Uniform Resource Identifiers) και οι τυπικές μέθοδοι του πρωτοκόλλου HTTP (GET, POST, PUT, DELETE) χρησιμοποιούνται για την αλληλεπίδραση με αυτούς τους πόρους. Αυτή η ομοιομορφία απλοποιεί την υλοποίηση τόσο του πελάτη όσο και του διακομιστή.

Για παράδειγμα, στο REST API που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας, η λίστα με όλα τα θέματα πτυχιακών και διπλωματικών εργασιών βρίσκεται μέσω του παρακάτω URI:

https://localhost:44362/api/projects

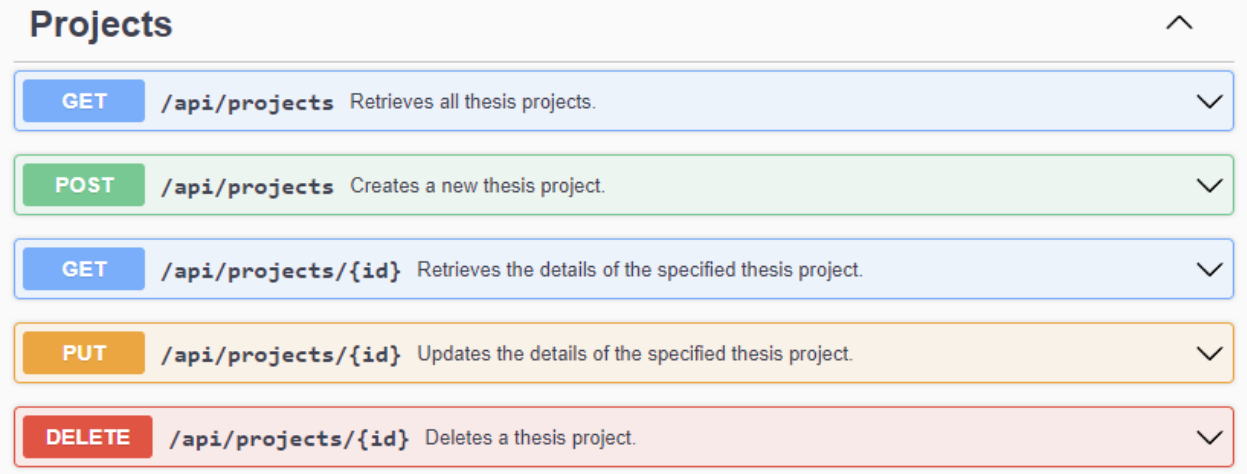
Κάθε URI, αποτελείται από 3 μέρη, στη προκειμένη περίπτωση:

- HTTPS δηλώνει το πρωτόκολλο επικοινωνίας
- localhost:44362 δηλώνει το domain name
- api/projects δηλώνει το endpoint

Ανάλογα τη μέθοδο HTTP που αποστέλλεται με το αίτημα, ο πελάτης δηλώνει στον διακομιστή τι να κάνει με το συγκεκριμένο πόρο. Κατά γενική σύμβαση, οι μέθοδοι HTTP αντιστοιχίζονται με τις βασικές λειτουργίες δεδομένων CRUD (Create, Read, Update, Delete) ως εξής:

- GET = READ
- POST = CREATE
- PUT = UPDATE
- DELETE = DELETE

Αντιστοίχως, στο REST API που αναπτύχθηκε εφαρμόζεται η ίδια λογική, ο συνδυασμός URI και HTTP μεθόδου δηλώνει στο σύστημα ποια ενέργεια να εκτελέσει για τον συγκεκριμένο πόρο, στη προκειμένη περίπτωση, για ένα θέμα πτυχιακής/διπλωματικής εργασίας όπως φαίνεται παρακάτω (Εικόνα 3.16):



Εικόνα 3.16 – Εκτέλεση CRUD λειτουργιών βάσει URI

3.4.2. Dependency Injection

Το Dependency Injection (DI) είναι ένα μοτίβο σχεδιασμού λογισμικού και μια τεχνική που χρησιμοποιείται στην ανάπτυξη λογισμικού για τη διαχείριση των εξαρτήσεων μεταξύ διαφορετικών στοιχείων ή κλάσεων με πιο ευέλικτο και διατηρήσιμο τρόπο. Με τον όρο εξάρτηση (dependency) εννοούμε ένα αντικείμενο ή στοιχείο στο οποίο βασίζεται ένα άλλο αντικείμενο για να εκτελέσει τη λειτουργία του [32]. Για παράδειγμα, μια κλάση «Car» που περιγράφει ένα αυτοκίνητο, μπορεί να εξαρτάται από ένα αντικείμενο «Engine» για να λειτουργήσει.

Ο πρωταρχικός στόχος του Dependency Injection είναι να επιτύχει την αρχή «Inversion of Control» (IoC), που σημαίνει πως ο έλεγχος της δημιουργίας και διαχείρισης αντικειμένων ή εξαρτήσεων μετατοπίζεται από το εξαρτημένο στοιχείο σε μια εξωτερική οντότητα, που συχνά ονομάζεται "container" ή "DI container" [32]. Ένας container επιλύει τις εξαρτήσεις και τις παρέχει στα στοιχεία που τις χρειάζονται. Κατά αυτό το τρόπο, αντί να δημιουργείται το εξαρτημένο αντικείμενο (αντικείμενο Engine σε αυτήν την περίπτωση) εντός της κλάσης Car, παρέχεται στη κλάση Car από τον container. Αυτό γίνεται συνήθως μέσω της εισχώρησης κατασκευαστή (constructor injection), της εισχώρησης μεθόδου (method injection) ή της εισχώρησης ιδιοτήτων (property injection).

Το Dependency Injection είναι μια τεχνική που χρησιμοποιείται κατά κόρον στο .NET Core Framework (C#), συνεπώς και στην εφαρμογή που αναπτύχθηκε, αλλά και σε πολλές άλλες γλώσσες και τεχνολογίες, όπως το Spring Framework της Java ή το Dagger Framework του Android. Είναι μια πρακτική που επιφέρει αρκετά πλεονεκτήματα, όπως ότι διευκολύνει τη διαδικασία unit testing σε μεγάλο βαθμό, καθώς επιτρέπει την εύκολη αντικατάσταση τις πραγματικές εξαρτήσεις με εικονικά αντικείμενα. Επίσης, επιτρέπει την αλλαγή ή την αντικατάσταση εξαρτήσεων χωρίς την τροποποίηση στο κώδικα του εξαρτημένων στοιχείων. Παράλληλα, τα στοιχεία με σαφώς καθορισμένες εξαρτήσεις μπορούν να επαναχρησιμοποιηθούν πιο εύκολα σε διαφορετικά μέρη μιας εφαρμογής ή ακόμα και σε άλλα έργα. Γενικά, ο κώδικας μιας εφαρμογής που ακολουθεί το μοτίβο DI, είναι συνήθως πιο τμηματικός και ευκολότερος στη συντήρηση, καθώς έχει μειωμένη σύζευξη μεταξύ των στοιχείων.

3.4.3. Repository Pattern

Το Repository Pattern είναι ένα μοτίβο σχεδιασμού που χρησιμοποιείται στην ανάπτυξη λογισμικού, ιδιαίτερα στον αντικειμενοστραφή προγραμματισμό, για τον διαχωρισμό της λογική που ανακτά δεδομένα από μια πηγή δεδομένων (όπως μια βάση δεδομένων) από την υπόλοιπη εφαρμογή [32]. Χρησιμοποιείται συνήθως στην C# και σε άλλες γλώσσες προγραμματισμού, ώστε ο κώδικας μιας εφαρμογής να είναι πιο ευανάγνωστος και εύκολα συντηρήσιμος, ενώ βοηθάει στην εκτέλεση δοκιμών, ειδικά όταν απαιτείται εργασία πάνω σε δεδομένα.

Η τεχνική εφαρμόζεται με τον ορισμό διαφορετικών διεπαφών αποθετηρίων (repository interfaces). Κάθε διεπαφή περιγράφει ένα σύνολο μεθόδων που καθορίζουν τον τρόπο πρόσβασης και χειρισμού των δεδομένων, αποκρύπτοντας τις λεπτομέρειες για το πώς ακριβώς αποθηκεύονται ή ανακτώνται τα δεδομένα από την υποκείμενη πηγή δεδομένων. Οι λεπτομέρειες για τον τρόπο αποθήκευσης και ανάκτησης των δεδομένων βρίσκονται στο αντίστοιχο αποθετήριο (repository). Για παράδειγμα, στην εφαρμογή που αναπτύχθηκε υπάρχουν διάφορα αποθετήρια, όπως το αποθετήριο θεμάτων. Η διεπαφή του αποθετηρίου θεμάτων (IProjectRepository) περιλαμβάνει τους ορισμούς των διαφόρων μεθόδων για λειτουργίες CRUD (Create, Read, Update, Delete) και αναζήτησης επί των θεμάτων (Εικόνα 3.17). Η υλοποίηση αυτών των μεθόδων υπάρχει στο αντίστοιχο αποθετήριο (ProjectRepository) (Εικόνα 3.18).

```
public interface IProjectsRepository
{
    2 references | 0 changes | 0 authors, 0 changes
    Task<IEnumerable<ThesisProject>> GetProjects();
    2 references | 0 changes | 0 authors, 0 changes
    Task<IEnumerable<ThesisProject>> GetProfessorProjects(string professorId);
    2 references | 0 changes | 0 authors, 0 changes
    Task<ThesisProject> GetStudentProject(string studentId);
    4 references | 0 changes | 0 authors, 0 changes
    Task<ThesisProject> GetProject(int projectId);
    2 references | 0 changes | 0 authors, 0 changes
    Task<ThesisProject> AddProject(ThesisProject project);
    2 references | 0 changes | 0 authors, 0 changes
    Task<ThesisProject> UpdateProject(ThesisProject project);
    2 references | 0 changes | 0 authors, 0 changes
    Task<ThesisProject> DeleteProject(int projectId);
    2 references | 0 changes | 0 authors, 0 changes
    Task<ThesisProject> GetProjectByTitle(string title);
}
```

Εικόνα 3.17 – Διεπαφή αποθετηρίου θεμάτων

```
public class ProjectsRepository : IProjectsRepository
{
    private readonly UniwaDbContext dbContext;

    0 references | 0 changes | 0 authors, 0 changes
    public ProjectsRepository(UniwaDbContext dbContext)
    {
        this.dbContext = dbContext;
    }

    2 references | 0 changes | 0 authors, 0 changes
    public async Task<ThesisProject> AddProject(ThesisProject project)
    {
        project.DateTimeCreated = DateTime.Now;
        project.DateTimeUpdated = DateTime.Now;

        var result = await dbContext.ThesisProjects.AddAsync(project);
        await dbContext.SaveChangesAsync();
        return result.Entity;
    }

    2 references | 0 changes | 0 authors, 0 changes
    public async Task<ThesisProject> DeleteProject(int projectId)
    {
        var result = await dbContext.ThesisProjects
            .FirstOrDefaultAsync(p => p.Id == projectId);

        if (result != null)
        {
            dbContext.ThesisProjects.Remove(result);
            await dbContext.SaveChangesAsync();
            return result;
        }

        return null;
    }
```

Εικόνα 3.18 – Υλοποίηση αποθετηρίου θεμάτων

Κεφάλαιο 4 – Σχεδιασμός & Ανάπτυξη Πληροφοριακού Συστήματος

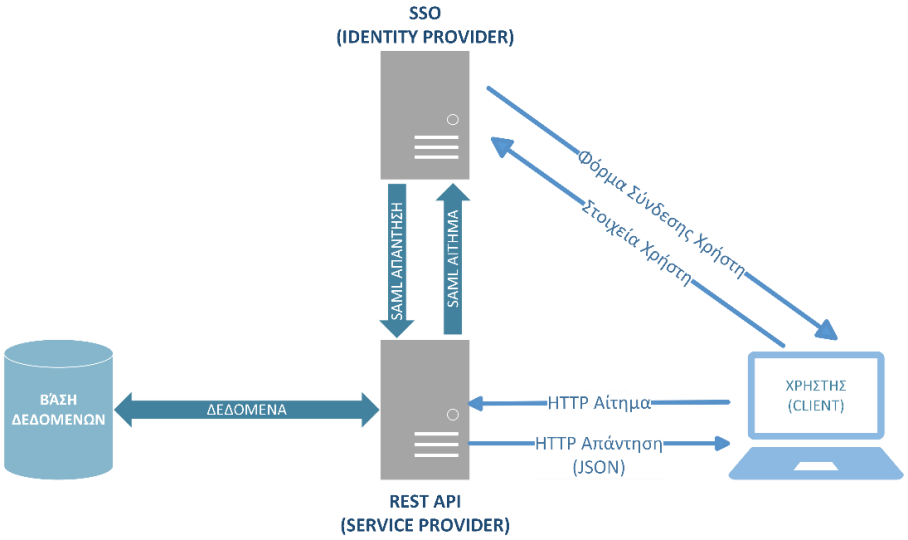
Σε αυτό το κεφάλαιο περιγράφεται η αρχιτεκτονική του πληροφοριακού συστήματος υποδομής που σχεδιάστηκε και αναπτύχθηκε στα πλαίσια της παρούσας Διπλωματικής εργασίας και στη συνέχεια αναλύονται σε βάθος τα επιμέρους στοιχεία που το απαρτίζουν.

4.1. Αρχιτεκτονική

Στο Διάγραμμα 4.1 απεικονίζεται η αρχιτεκτονική του πληροφοριακού συστήματος υποδομής που αναπτύχθηκε. Ουσιαστικά πρόκειται για ένα πληροφοριακό σύστημα που εφαρμόζει το μοντέλο πελάτη (client) και εξυπηρετητή (server) και χρησιμοποιεί μια κεντρική υπηρεσία Single Sign-On (SSO) για την αυθεντικοποίηση των χρηστών.

Συγκεκριμένα, το Π.Σ. διαθέτει μία βάση δεδομένων SQL, στην οποία καταχωρούνται όλα τα απαραίτητα δεδομένα, όπως για παράδειγμα τα θέματα των Πτυχιακών/Διπλωματικών εργασιών και οι αιτήσεις ανάληψης των φοιτητών. Για την διασύνδεση και την αλληλεπίδραση ενός τρίτου συστήματος ή εφαρμογής (clients), όπως ένα περιβάλλον Front-End, με την βάση δεδομένων του Π.Σ., αναπτύχθηκε ένα REST API. Μια διαδικτυακή υπηρεσία που επιτρέπει την ανταλλαγή δεδομένων μέσω HTTP αιτημάτων. Τα δεδομένα που ανταλλάσσονται μεταξύ του REST API και των διαφόρων clients είναι σε μορφή JSON.

Προκειμένου οι χρήστες να μπορούν επικοινωνήσουν με το Π.Σ., χρειάζεται να συνδεθούν σε έναν κεντρικό σύστημα ταυτοποίησης (Identity Provider), μέσω μίας SSO υπηρεσίας. Η ταυτοποίηση των χρηστών γίνεται σύμφωνα με το πρωτόκολλο SAML 2.0, το οποίο υποστηρίζεται από τη κεντρική υπηρεσία ταυτοποίησης των χρηστών του Πανεπιστημίου Δυτικής Αττικής, επιτρέποντας έτσι τη μελλοντική διασύνδεση της με το Π.Σ.



Διάγραμμα 4.1 - Αρχιτεκτονική του Πληροφοριακού Συστήματος

4.2. Βάση Δεδομένων

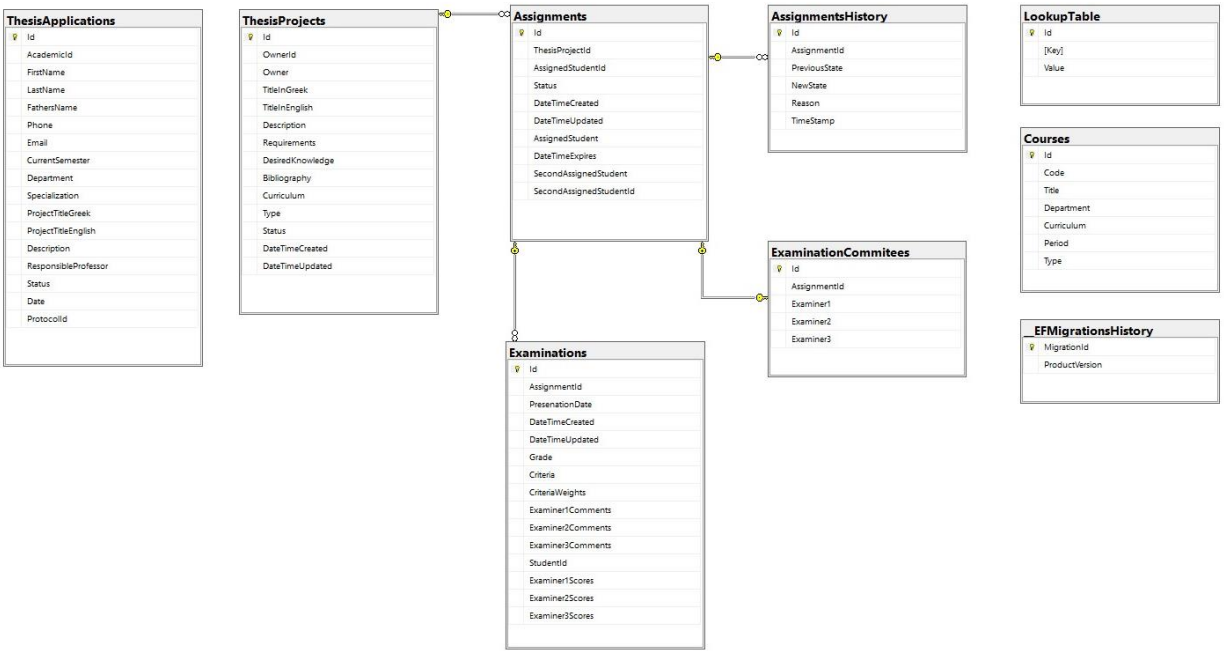
Η βάση δεδομένων αποτελεί τα θεμέλια του συστήματος υποδομής που υλοποιήθηκε. Ο σχεδιασμός της βάσης δεδομένων βασίστηκε στις ανάγκες που προέκυψαν από την ανάλυση της υφιστάμενης διαδικασίας που ακολουθείται από το Τμήμα όσον αφορά τις Πτυχιακές/Διπλωματικές εργασίες των φοιτητών, όπως αυτή περιγράφεται στο 1ο Κεφάλαιο.

Το πρώτο βήμα του σχεδιασμού της βάσης δεδομένων, είναι η διάκριση των βασικών εμπλεκόμενων οντοτήτων. Από το Διάγραμμα 1.1, γίνεται εύκολα αντιληπτό πως η πρώτη οντότητα που σηματοδοτεί και την έναρξη της διαδικασίας είναι τα «Θέματα» Πτυχιακών/Διπλωματικών εργασιών. Τα θέματα καταχωρούνται, τροποποιούνται και δημοσιεύονται από τους υπεύθυνους καθηγητές. Στην συνέχεια, οι φοιτητές έρχονται σε συνεννόηση με κάποιον καθηγητή για να αναλάβουν την εκπόνηση ενός συγκεκριμένου θέματος. Εφόσον πάρει έγκριση από τον υπεύθυνο καθηγητή, γίνεται ανάθεση του θέματος στον φοιτητή, ο οποίος με τη σειρά του χρειάζεται να συμπληρώσει και να καταθέσει την Αίτηση Ανάληψης. Σε αυτό το σημείο, προκύπτουν δύο οντότητες, η μία αφορά τις «Αναθέσεις» των εργασιών και η άλλη τις «Αιτήσεις» των φοιτητών. Οι Αναθέσεις συσχετίζουν τους φοιτητές με το αντίστοιχο θέμα Πτυχιακής/Διπλωματικής εργασίας και τον επιβλέποντα καθηγητή, ο οποίος και τις διαχειρίζεται. Οι Αιτήσεις καταχωρούνται από τους φοιτητές και αποθηκεύονται για λόγους μηχανοργάνωσης, μείωσης της γραφειοκρατίας και αρχειοθέτησης. Επιπλέον, ο καθηγητής ορίζει την Τριμελή Εξεταστική Επιτροπή, την ημερομηνία εξέτασης και τελικά την αναλυτική βαθμολογία της εργασίας. Συνεπώς, η οντότητα «Αναθέσεις» μπορεί να αναλυθεί σε μικρότερες σχετικές οντότητες, όπως οι «Εξεταστικές Επιτροπές» και οι «Εξετάσεις».

Φυσικά, υπάρχουν και δευτερεύοντες οντότητες όπως για παράδειγμα η λίστα των μαθημάτων, που χρησιμεύει στη συμπλήρωση της λίστας προ απαιτούμενων μαθημάτων ενός θέματος Πτυχιακής/Διπλωματικής εργασίας. Αξίζει να σημειωθεί, πως φυσικά συμμετέχουν και οι οντότητες «Φοιτητής» και «Καθηγητής» σε αυτή τη διαδικασία, ωστόσο αυτές δεν συμπεριλαμβάνονται στη βάση δεδομένων του συστήματος υποδομής, καθώς θα γίνεται διασύνδεση με κεντρική υπηρεσία για την ταυτοποίηση των χρηστών και την λήψη των απαραίτητων στοιχείων τους. Στις παρακάτω ενότητες περιγράφονται αναλυτικά οι πίνακες, τα πεδία τους και οι μεταξύ τους σχέσεις.

4.2.1. Σχεδιάγραμμα Οντότητας/Σχέσης

Στο Διάγραμμα 4.1, απεικονίζεται το διάγραμμα Οντοτήτων/Συσχετίσεων (Entity-Relationship Diagram) της βάσης δεδομένων του πληροφοριακού συστήματος.



Διάγραμμα 4.2 - Διάγραμμα Οντοτήτων/Συσχετίσεων της Β.Δ. (ER Diagram)

4.2.2. Πίνακες της Βάσης Δεδομένων

Το δεύτερο βήμα του σχεδιασμού της βάσης δεδομένων είναι η δημιουργία των σχετικών πινάκων. Η βάση δεδομένων του συστήματος υποδομής αποτελείται από 8 πίνακες δεδομένων και έναν επιπλέον πίνακα που αφορά το ιστορικό των migrations του Entity Framework. Πιο συγκεκριμένα, αποτελείται από τους εξής πίνακες:

Πίνακας "ThesisProjects"

Στον πίνακα «ThesisProjects» αποθηκεύονται όλα τα θέματα Πτυχιακών και Διπλωματικών εργασιών που καταχωρούνται από τους υπευθύνους καθηγητές και αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό θέματος (Αύξων αριθμός)	Primary Key
OwnerId	Nvarchar(20)	Κωδικός υπεύθυνου καθηγητή	NOT NULL
Owner	Nvarchar(50)	Ονοματεπώνυμο υπεύθυνου καθηγητή	
TitleInGreek	Nvarchar(250)	Τίτλος θέματος στα Ελληνικά	
TitleInEnglish	Nvarchar(250)	Τίτλος θέματος στα Αγγλικά	
Description	Nvarchar(1000)	Περιγραφή θέματος	
Requirements	Nvarchar(250)	Προ-απαιτούμενα μαθήματα	
DesiredKnowledge	Nvarchar(250)	Επιθυμητές γνώσεις	

Bibliography	Nvarchar(1000)	Βιβλιογραφία	
Curriculum	Nvarchar(20)	Κύκλος Σπουδών (ΠΑΔΑ/ΤΕΙ)	
Type	Nvarchar(10)	Ατομική/Ομαδική	
Status	Nvarchar(25)	Διαθέσιμο/Μη Διαθέσιμο/Απενεργοποιημένο	
DateTimeCreated	Datetime	Ημερομηνία δημιουργίας	NOT NULL
DateTimeUpdated	Datetime	Ημερομηνία τροποποίησης	NOT NULL

Πίνακας 4.1 - Δομή του πίνακα «ThesisProjects» της Β.Δ.

Πίνακας «ThesisApplications»

Στον πίνακα «ThesisApplications», αποθηκεύονται όλες οι καταχωρημένες Αιτήσεις Ανάληψης Θέματος Πτυχιακής/Διπλωματικής εργασίας των φοιτητών. Αξίζει να σημειωθεί, πως πολλά από τα στοιχεία της Αίτησης Ανάληψης χρησιμοποιούνται και στις υπόλοιπες αιτήσεις, όπως η Αίτηση Εξέτασης και το Πρακτικό Βαθμολόγησης. Επομένως, με χρήση συνδυαστικών queries, μπορεί να γίνει ανάκτηση των απαραίτητων δεδομένων για τη δημιουργία και την ηλεκτρονική εκτύπωση όλων των απαραίτητων αιτήσεων/εγγράφων της διαδικασίας. Ο πίνακας αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό αίτησης (Αύξων αριθμός)	Primary Key, NOT NULL
AcademicId	Nvarchar(20)	Αριθμός μητρώου φοιτητή	
FirstName	Nvarchar(25)	Όνομα φοιτητή	
LastName	Nvarchar(25)	Επίθετο φοιτητή	
FathersName	Nvarchar(25)	Όνομα πατρός φοιτητή	
Phone	Nvarchar(20)	Τηλέφωνο επικοινωνίας	
Email	Nvarchar(50)	Email επικοινωνίας	
CurrentSemester	Nvarchar(5)	Τρέχον εξάμηνο σπουδών	
Department	Nvarchar(100)	Τμήμα	
Specialization	Nvarchar(100)	Κατεύθυνση	
ProjectTitleGreek	Nvarchar(250)	Τίτλος θέματος στα Ελληνικά	
ProjectTitleEnglish	Nvarchar(250)	Τίτλος θέματος στα Αγγλικά	
Description	Nvarchar(1000)	Περιγραφή θέματος	
ResponsibleProfessor	Nvarchar(50)	Επιβλέποντας καθηγητής	
Status	Nvarchar(25)	Κατάσταση αίτησης	
Date	Date	Ημερομηνία καταχώρησης	
ProtocolId	Nvarchar(20)	Αριθμός πρωτοκόλλου	

Πίνακας 4.2 - Δομή του πίνακα «ThesisApplications» της Β.Δ.

Πίνακας «Assignments»

Κάθε εγγραφή του πίνακα «Assignments», αντιπροσωπεύει μια Ανάθεση Πτυχιακής/Διπλωματικής εργασίας. Ουσιαστικά, ο πίνακας αυτός συσχετίζει ένα θέμα Πτυχιακής/Διπλωματικής με τον φοιτητή που το αναλαμβάνει. Ένας καθηγητής μπορεί να δημιουργεί μια ανάθεση για κάθε ένα από τα διαθέσιμα θέματα του, σε έναν ή και δύο ενδιαφερόμενους φοιτητές σε περίπτωση ομαδικής εργασίας. Ο καθηγητής μπορεί έτσι να παρακολουθεί την πορεία εξέλιξης και να διαχειρίζεται τις ανατεθειμένες εργασίες του, ορίζοντας διαφορετικές καταστάσεις, ημερομηνία εξέτασης και εξεταστικής επιτροπής καθώς και να τις βαθμολογεί ή ακόμα και να τις ακυρώνει. Ο πίνακας αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό ανάθεσης (Αύξων αριθμός)	Primary Key, NOT NULL
ThesisProjectId	Integer	Κωδικός θέματος	Foreign Key, NOT NULL
AssignedStudentId	Nvarchar(20)	Αριθμός μητρώου φοιτητή	NOT NULL
AssignedStudent	Nvarchar(50)	Ονοματεπώνυμο φοιτητή	
SecondAssignedStudentId	Nvarchar(20)	Αριθμός μητρώου δεύτερου φοιτητή (περίπτωση ομαδικής εργασίας)	
SecondAssignedStudent	Nvarchar(50)	Ονοματεπώνυμο δεύτερου φοιτητή (περίπτωση ομαδικής εργασίας)	
Status	Nvarchar(10)	Κατάσταση ανάθεσης	
DateTimeCreated	DateTime	Ημερομηνία δημιουργίας	NOT NULL
DateTimeUpdated	DateTime	Ημερομηνία τροποποίησης	NOT NULL
DateTimeExpires	DateTime	Ημερομηνία προθεσμίας	NOT NULL

Πίνακας 4.3 - Δομή του πίνακα «Assignments» της Β.Δ.

Πίνακας «AssignmentsHistory»

Ο πίνακας AssignmentsHistory χρησιμεύει στη διατήρηση της ιστορικότητας των διαφορετικών καταστάσεων μιας ανατεθειμένης Πτυχιακής/Διπλωματικής εργασίας. Συνεπώς, έχει άμεση συσχέτιση με τον πίνακα Assignments, λόγω του ότι κάθε φορά που αλλάζει η κατάσταση μιας ανάθεσης, δημιουργείται μια εγγραφή στον πίνακα AssignmentsHistory με τη προηγούμενη και τη νέα κατάσταση της ανάθεσης. Ο πίνακας αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό (Αύξων αριθμός)	Primary Key, NOT NULL
AssignmentId	Integer	Κωδικός ανάθεσης	Foreign Key, NOT NULL
PreviousState	Nvarchar(20)	Προηγούμενη κατάσταση ανάθεσης	
NewState	Nvarchar(50)	Νέα κατάσταση ανάθεσης	
Reason	Nvarchar(20)	Λόγος αλλαγής/Σχόλια	
TimeStamp	DateTime	Ημερομηνία	NOT NULL

Πίνακας 4.4 - Δομή του πίνακα «AssignmentsHistory» της Β.Δ.

Πίνακας «ExaminationCommittees»

Στον πίνακα ExaminationCommittees αποθηκεύονται οι συνθέσεις των Τριμελών Εξεταστικών Επιτροπών που ορίζονται από τους επιβλέποντες καθηγητές για κάθε ανατεθειμένο θέμα Πτυχιακής/Διπλωματικής εργασίας. Έχει άμεση συσχέτιση με τον πίνακα Assignments και αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό (Αύξων αριθμός)	Primary Key, NOT NULL
AssignmentId	Integer	Κωδικός ανάθεσης	Foreign Key, NOT NULL
Examiner1	Nvarchar(50)	Ονοματεπώνυμο πρώτου εξεταστή	
Examiner2	Nvarchar(50)	Ονοματεπώνυμο δεύτερου εξεταστή	
Examiner3	Nvarchar(50)	Ονοματεπώνυμο τρίτου εξεταστή	

Πίνακας 4.5 - Δομή του πίνακα «ExaminationCommittees» της Β.Δ.

Πίνακας «Examinations»

Ο πίνακας Examinations χρησιμεύει στην αποθήκευση της αναλυτικής βαθμολογίας και της ημερομηνίας παρουσίασης/εξέτασης των ανατεθειμένων Πτυχιακών/Διπλωματικών εργασιών. Χρησιμεύει επίσης, στην ανάκτηση της απαραίτητης πληροφορίας για την ηλεκτρονική δημιουργία και εκτύπωση του Πρακτικού Αξιολόγησης της Εξεταστικής Επιτροπής. Συσχετίζεται άμεσα με τον πίνακα Assignments και αποτελείται από τις στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό (Αύξων αριθμός)	Primary Key, NOT NULL
AssignmentId	Integer	Κωδικός ανάθεσης	Foreign Key, NOT NULL
StudentId	Nvarchar(20)	Αριθμός μητρώου εξεταζόμενου φοιτητή	
Criteria	Nvarchar(250)	Κριτήρια αξιολόγησης	
CriteriaWeights	Nvarchar(50)	Βαρύτητα κριτηρίων αξιολόγησης	
Examiner1Scores	Nvarchar(50)	Αναλυτική βαθμολογία πρώτου εξεταστή	
Examiner2Scores	Nvarchar(50)	Αναλυτική βαθμολογία δεύτερου εξεταστή	
Examiner3Scores	Nvarchar(50)	Αναλυτική βαθμολογία τρίτου εξεταστή	
Examiner1Comments	Nvarchar(100)	Σχόλια πρώτου εξεταστή	
Examiner2Comments	Nvarchar(100)	Σχόλια δεύτερου εξεταστή	
Examiner3Comments	Nvarchar(100)	Σχόλια τρίτου εξεταστή	
Grade	Decimal 3,1	Τελικός βαθμός	
PresentationDate	DateTime	Ημερομηνία εξέτασης	NOT NULL
DateTimeCreated	DateTime	Ημερομηνία δημιουργίας	NOT NULL
DateTimeUpdated	DateTime	Ημερομηνία τροποποίησης	NOT NULL

Πίνακας 4.6 - Δομή του πίνακα «Examinations» της Β.Δ.

Πίνακας «Courses»

Ο πίνακας Courses περιέχει την συνολική λίστα μαθημάτων του Τμήματος Μηχανικών Πληροφορικής και Η/Υ. Είναι ένας βοηθητικός πίνακας που μπορεί να αξιοποιηθεί από το σύστημα Front-End, για την συμπλήρωση των προ-απαιτούμενων μαθημάτων κατά τη δημιουργία ενός θέματος Πτυχιακής/Διπλωματικής εργασίας και αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό ανάθεσης (Αύξων αριθμός)	Primary Key, NOT NULL

Code	Nvarchar(20)	Κωδικός μαθήματος	
Title	Nvarchar(100)	Αριθμός μητρώου φοιτητή	
Department	Nvarchar(100)	Ονοματεπώνυμο φοιτητή	
Curriculum	Nvarchar(25)	Αριθμός μητρώου δεύτερου φοιτητή (περίπτωση ομαδικής εργασίας)	
Period	Nvarchar(10)	Ονοματεπώνυμο δεύτερου φοιτητή (περίπτωση ομαδικής εργασίας)	
Type	Nvarchar(50)	Κατάσταση ανάθεσης	

Πίνακας 4.7 - Δομή του πίνακα «Courses» της Β.Δ.

Πίνακας «Lookup»

Ο πίνακας Lookup, είναι ένας βοηθητικός πίνακας τιμών στον οποίο αποθηκεύονται διαφορετικά ζευγάρια κλειδιών-τιμών (key-value pairs), που μπορεί να αξιοποιηθεί από ένα σύστημα Front-End ως ευρετήριο, για να ανακτήσει διάφορες λίστες τιμών όπως οι διαφορετικές πιθανές καταστάσεις μιας ανάθεσης, η λίστα κριτηρίων αξιολόγησης και η λίστα κατευθύνσεων του Τμήματος. Ο πίνακας που αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
ID	Integer	Μοναδικό αναγνωριστικό (Αύξων αριθμός)	Primary Key, NOT NULL
Key	Nvarchar(50)	Κλειδί ζευγαριού	
Value	Nvarchar(250)	Τιμή ζευγαριού	

Πίνακας 4.8 - Δομή του πίνακα «Lookup» της Β.Δ.

Πίνακας «EFMigrationHistory»

Πρόκειται για έναν συστημικό πίνακα που δημιουργείται αυτόματα από το Entity Framework. Χρησιμεύει ως πίνακας ιστορικότητας για το ποια migrations έχουν εφαρμοστεί στη βάση δεδομένων και αποτελείται από τις εξής στήλες:

Όνομα	Τύπος	Περιγραφή	Περιορισμοί
MigrationId	Nvarchar(150)	Κλειδί ζευγαριού	Primary Key, NOT NULL
ProductVersion	Nvarchar(30)	Τιμή ζευγαριού	NOT NULL

Πίνακας 4.9 - Δομή του πίνακα «EFMigrationHistory» της Β.Δ.

4.3. REST API

Η επικοινωνία και η αλληλεπίδραση ενός τρίτου συστήματος ή μιας εφαρμογής με τη βάση δεδομένων του πληροφοριακού συστήματος υποδομής, πραγματοποιείται μέσω ενός REST API. Το REST API δέχεται HTTP αιτήματα, τα επεξεργάζεται και απαντάει με τα κατάλληλα δεδομένα. Για τη μορφοποίηση των δεδομένων που ανταλλάσσονται μέσω των HTTP Requests και HTTP Responses, χρησιμοποιήθηκε η μορφή JSON. Το REST API αναπτύχθηκε με τη χρήση της γλώσσας C# και του ASP.NET Core Framework (έκδοση 3.1 για μέγιστη συμβατότητα).

Η υπηρεσία είναι οργανωμένη σε πέντε διαφορετικούς ρυθμιστές (controllers), όπου ο κάθε ένας περιέχει ένα σύνολο από τελικά σημεία (endpoints) που εκτελούν μια συγκεκριμένη λειτουργία, ώστε να γίνεται εύκολα αντιληπτό ποια οντότητα της βάσης δεδομένων αφορούν. Ακολουθούν αναλυτικοί πίνακες για κάθε controller, όπου επεξηγείται η λειτουργία του κάθε endpoint του, καθώς και η αναμενόμενη μορφή των αιτημάτων και των απαντήσεων του.

4.3.1. Πίνακες κλήσεων του REST API

ApplicationsController:

Endpoint	HTTP Method	Περιγραφή	Request Body	Response Body
/api/applications	GET	Επιστρέφει τα στοιχεία όλων των αιτήσεων ανάληψης πτυχιακής / διπλωματικής των φοιτητών.	-	[{ "id": int, "academicId": "string", "firstName": "string", "lastName": "string", "fathersName": "string", "phone": "string", "email": "string", "currentSemester": "string", "department": "string", "specialization": "string", "projectTitleGreek": "string", "projectTitleEnglish": "string", "description": "string",

				"responsibleProfessor": "string", "status": "string", "date": "string", "protocolId": "string" }, …]
/api/applications/thesis/{id}	GET	Επιστρέφει τα στοιχεία μιας συγκεκριμένης αίτησης ανάληψης βάσει του 'id' της αίτησης.	-	{ "id": int, "academicId": "string", "firstName": "string", "lastName": "string", "fathersName": "string", "phone": "string", "email": "string", "currentSemester": "string", "department": "string", "specialization": "string", "projectTitleGreek": "string", "projectTitleEnglish": "string", "description": "string", "responsibleProfessor": "string", "status": "string", "date": "string", "protocolId": "string" }
/api/applications/thesis	POST	Δημιουργεί μια νέα αίτηση Ανάληψης πτυχιακής / διπλωματικής εργασίας.	{ "id": 0, "academicId": "string", "firstName": "string", "lastName": "string", "fathersName": "string", "phone": "string", "email": "string", "currentSemester": "strin",	HTTP 201 (Created)

			<pre>"department": "string", "specialization": "string", "projectTitleGreek": "string", "projectTitleEnglish": "string", "description": "string", "responsibleProfessor": "string", "status": "string", "date": "string", "protocolId": "string" }</pre>	
/api/applications/thesis/{id}	DELETE	Διαγράφει μια συγκεκριμένη αίτηση ανάληψης βάσει του αριθμού (id) της αίτησης.	-	HTTP 200
/api/applications/thesis/student/{academicId}	GET	Επιστρέφει τα στοιχεία της αίτησης ανάληψης ενός συγκεκριμένου φοιτητή, βάσει του Α.Μ. του φοιτητή (academicId).	-	<pre>{ "id": int, "academicId": "string", "firstName": "string", "lastName": "string", "fathersName": "string", "phone": "string", "email": "string", "currentSemester": "string", "department": "string", "specialization": "string", "projectTitleGreek": "string", "projectTitleEnglish": "string", "description": "string", "responsibleProfessor": "string", "status": "string", "date": "string",</pre>

				<pre>"protocolId": "string" }</pre>
/api/applications/thesis/student/{academicId}	DELETE	Διαγράφει την αίτηση ανάληψης ενός συγκεκριμένου φοιτητή, βάσει του Α.Μ. του φοιτητή (academicId).	-	HTTP 200
/api/applications/thesis/print/student/{academicId} ?secondAcademicId={secondAcademicId}	GET	Επιστρέφει την αίτηση ανάληψης ενός συγκεκριμένου φοιτητή σε μορφή base64, βάσει του Α.Μ. του φοιτητή (academicId). Σε περίπτωση ομαδικής εργασίας απαιτείται και το Α.Μ. του δεύτερου φοιτητή (secondAcademicId).	-	<pre>{ "base64":"string" }</pre>
/api/applications/thesis/print/examination/{academicId} ?secondAcademicId={secondAcademicId}	GET	Επιστρέφει την αίτηση ανάληψης ενός συγκεκριμένου φοιτητή σε μορφή base64, βάσει του Α.Μ. του φοιτητή (academicId). Σε περίπτωση ομαδικής εργασίας απαιτείται και το Α.Μ. του δεύτερου φοιτητή (secondAcademicId).	-	<pre>{ "base64":"string" }</pre>
/api/applications/thesis/print/extension/{academicId} ?secondAcademicId={secondAcademicId}	GET	Επιστρέφει την αίτηση παράτασης ενός συγκεκριμένου φοιτητή σε μορφή base64, βάσει του Α.Μ. του φοιτητή (academicId). Σε περίπτωση ομαδικής εργασίας απαιτείται και το Α.Μ. του δεύτερου φοιτητή (secondAcademicId).	-	<pre>{ "base64":"string" }</pre>

/api/applications/thesis/print/evaluation/{academicId}	GET	Επιστρέφει το πρακτικό βαθμολόγησης ενός συγκεκριμένου φοιτητή σε μορφή base64, βάσει του Α.Μ. του φοιτητή (academicId).	-	`{ "base64":"string" }`

Πίνακας 4.10 - Πίνακας κλήσεων του ApplicationsController

Projects Controller:

Endpoint	HTTP Method	Περιγραφή	Request Body	Response Body
/api/projects	GET	Επιστρέφει όλες τις αιτήσεις ανάληψης πτυχιακής / διπλωματικής των φοιτητών.	-	`[{ "id": int, "ownerId": "string", "owner": "string", "titleInGreek": "string", "titleInEnglish": "string", "description": "string", "requirements": "string", "desiredKnowledge": "string", "bibliography": "string", "curriculum": "string", "type": "string", "status": "string", "dateTimeCreated": 2023-01-01T00:00:00", "dateTimeUpdated": "2023-01-01T00:00:00" }, …]`
/api/projects	POST	Δημιουργεί ένα νέο θέμα πτυχιακής / διπλωματικής εργασίας.	`{ "ownerId": "string", "owner": "string", "titleInGreek": "string", "titleInEnglish": "string", "description": "string", "requirements": "string", "desiredKnowledge": "string", "bibliography": "string", "curriculum": "string", "type": "string", "status": "string" }`	`HTTP 201 (Created)`

/api/projects/{id}	GET	Επιστρέφει τα στοιχεία ενός συγκεκριμένου θέματος πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού θέματος (id).	-	<pre>{ "id": int, "ownerId": "string", "owner": "string", "titleInGreek": "string", "titleInEnglish": "string", "description": "string", "requirements": "string", "desiredKnowledge": "string", "bibliography": "string", "curriculum": "string", "type": "string", "status": "string", "dateTimeCreated": "2023-01-01T00:00:00", "dateTimeUpdated": "2023-01-01T00:00:00" }</pre>
/api/projects/{id}	PUT	Τροποποιεί τα στοιχεία ενός συγκεκριμένου θέματος πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού θέματος (id).	-	<pre>{ "id": int, "ownerId": "string", "owner": "string", "titleInGreek": "string", "titleInEnglish": "string", "description": "string", "requirements": "string", "desiredKnowledge": "string", "bibliography": "string", "curriculum": "string", "type": "string", "status": "string", "dateTimeCreated": "2023-01-01T00:00:00", "dateTimeUpdated": "2023-01-01T00:00:00" }</pre>
/api/projects/{id}	DELETE	Διαγράφει ένα θέμα πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού θέματος (id).	-	HTTP 200

/api/projects/professor/{id}	GET	Επιστρέφει όλα τα θέματα πτυχιακών / διπλωματικών εργασιών ενός καθηγητή, βάσει του αριθμού μητρώου του (id).	-	<pre>[{ "id": int, "ownerId": "string", "owner": "string", "titleInGreek": "string", "titleInEnglish": "string", "description": "string", "requirements": "string", "desiredKnowledge": "string", "bibliography": "string", "curriculum": "string", "type": "string", "status": "string", "dateTimeCreated": 2023-01-01T00:00:00", "dateTimeUpdated": "2023-01-01T00:00:00" }, …]</pre>
/api/projects/student/{id}	GET	Επιστρέφει τα στοιχεία του θέματος πτυχιακής / διπλωματικής εργασίας που έχει αναλάβει ένας συγκεκριμένος φοιτητής, βάσει του αριθμού μητρώου του (id).	-	<pre>{ "id": int, "ownerId": "string", "owner": "string", "titleInGreek": "string", "titleInEnglish": "string", "description": "string", "requirements": "string", "desiredKnowledge": "string", "bibliography": "string", "curriculum": "string", "type": "string", "status": "string", "dateTimeCreated": 2023-01-01T00:00:00", "dateTimeUpdated": "2023-01-01T00:00:00" }</pre>

Πίνακας 4.11 - Πίνακας κλήσεων του ProjectsController

ValuesController:

Endpoint	HTTP Method	Περιγραφή	Request Body	Response Body
/api/values/courses	GET	Επιστρέφει όλες τις αιτήσεις ανάληψης πτυχιακής / διπλωματικής των φοιτητών.	-	[{ "id": 3, "code": "ICE1-1003", "title": "ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ", "department": "ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ", "curriculum": "ΠΡΟΓΡΑΜΜΑ 5 ΕΤΩΝ ΣΠΟΥΔΩΝ (2019)", "period": "Χειμερινή", "type": "Υποχρεωτικό" }, { "id": 4, "code": "ICE1-1004", "title": "ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ", "department": "ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ", "curriculum": "ΠΡΟΓΡΑΜΜΑ 5 ΕΤΩΝ ΣΠΟΥΔΩΝ (2019)", "period": "Χειμερινή", "type": "Υποχρεωτικό" }, …]

/api/values/{key}	GET	Επιστρέφει μια λίστα τιμών, βάσει ενός συγκεκριμένου κλειδιού (key).	-	[
				{
				"id": 1,
				"key": "project-status",
				"value": "Διαθέσιμο"
				},
				{
				"id": 2,
				"key": "project-status",
				"value": "Μη Διαθέσιμο"
				},
				{
				"id": 3,
				"key": "project-status",
				"value": "Απενεργοποιημένο"
				}
				]

/api/values	POST	Δημιουργεί ένα νέο ζευγάρι κλειδιού/τιμής.	`{ "key": "string", "value": "string" }`	`HTTP 201 (Created)`
/api/values/{id}	DELETE	Διαγράφει ένα συγκεκριμένο ζευγάρι κλειδιού/τιμής, βάσει του αριθμού ζευγαριού (id).	-	`HTTP 200`

Πίνακας 4.12 - Πίνακας κλήσεων του ValuesController

AssignmentsController:

Endpoint	HTTP Method	Περιγραφή	Request Body	Response Body
/api/assignments	GET	Επιστρέφει όλες τις αναθέσεις πτυχιακών / διπλωματικών εργασιών.	-	`[{ "id": int, "thesisProjectId": int, "assignedStudentId": "string", "assignedStudent": "string", "secondAssignedStudentId": "string", "secondAssignedStudent": "string", "status": "string", "dateTimeCreated": "2023-01-01T00:00:00", "dateTimeUpdated": "2023-01-01T00:00:00", "dateTimeExpires": "2023-01-01T00:00:00" }, …]`
/api/assignments	POST	Δημιουργεί μια νέα ανάθεση πτυχιακής / διπλωματικής εργασίας.	`{ "thesisProjectId": int, "assignedStudentId": "string", "assignedStudent": "string", "secondAssignedStudentId": "string", "secondAssignedStudent": "string" }`	HTTP 201 (Created)
/api/assignments/{id}	GET	Επιστρέφει όλα τα στοιχεία μιας συγκεκριμένης ανάθεσης, βάσει του αριθμού ανάθεσης (id).	-	`{ "id": int, "thesisProjectId": int, "assignedStudentId": "string", "assignedStudent": "string", "secondAssignedStudentId": "string", "secondAssignedStudent": "string",`

				<pre>"status": "string",
				"dateTimeCreated": "2023-01-01T00:00:00"
				"dateTimeUpdated": "2023-01-01T00:00:00"
				"dateTimeExpires": "2023-01-01T00:00:00"
				"thesisProject": {
				"id": int,
				"ownerId": "string",
				"owner": "string",
				"titleInGreek": "string",
				"titleInEnglish": "string",
				"description": "string",
				"requirements": "string",
				"desiredKnowledge": "string",
				"bibliography": "string",
				"curriculum": "string",
				"type": "string",
				"status": "string",
				"dateTimeCreated": "2023-01-01T00:00:00"
				"dateTimeUpdated": "2023-01-01T00:00:00"
				},
				"examinationCommitee": {
				"id": int,
				"assignmentId": int,
				"examiner1": "string",
				"examiner2": "string",
				"examiner3": "string"
				},
				"examinations": [
				{
				"id": int,
				"assignmentId": int,
				"studentId": "string",
				"grade": decimal,
				"criteria": "string",
				"criteriaWeights": "string",
				"examiner1Scores": "string",
				"examiner2Scores": "string",
				"examiner3Scores": "string",
				"examiner1Comments": "string",
				"examiner2Comments": "string",</pre>

				<pre>"examiner3Comments": "string",
				"presenationDate": "2023-01-01T00:00:00",
				"dateTimeCreated": "2023-01-01T00:00:00",
				"dateTimeUpdated": "2023-01-01T00:00:00"
				}
				],
				"statusChanges": [
				{
				"id": int,
				"assignmentId": int,
				"previousState": "string",
				"newState": "string",
				"reason": "string",
				"timeStamp": "2023-01-01T00:00:00"
				}
				]
				}</pre>

/api/assignments/{id}	DELETE	Ακυρώνει μια ανάθεση πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	-	HTTP 200
/api/assignments/{projectId}/history	GET	Επιστρέφει το ιστορικό αναθέσεων ενός συγκεκριμένου θέματος πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού θέματος (projectId).	-	<pre>[{ "id": int, "thesisProjectId": int, "assignedStudentId": "string", "assignedStudent": "string", "secondAssignedStudentId": "string", "secondAssignedStudent": "string", "status": "string", "dateTimeCreated": "2023-01-01T00:00:00" "dateTimeUpdated": "2023-01-01T00:00:00" "dateTimeExpires": "2023-01-01T00:00:00" }, "statusChanges": [{ "id": int, "assignmentId": int, "previousState": "string",</pre>

				<pre>"newState": "string", "reason": "string", "timeStamp": "2023-01-01T00:00:00" }, { "id": int, "assignmentId": int, "previousState": "string", "newState": "string", "reason": "string", "timeStamp": "2023-01-01T00:00:00" }, …] }, …]</pre>
/api/assignments/{id}/change-status	POST	Αλλάζει την κατάσταση ανάθεσης μιας συγκεκριμένης πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	<pre>{ "newStatus": "string", "reason": "string" }</pre>	HTTP 201 (Created)
/api/assignments/{id}/extend	POST	Παρατείνει την προθεσμία ολοκλήρωσης μιας ανατεθειμένης πτυχιακής / διπλωματικής εργασίας κατά 1 εξάμηνο, βάσει του αριθμού ανάθεσης (id).	-	HTTP 201 (Created)
/api/assignments/{id}/presentation	POST	Ορίζει την ημερομηνία εξέτασης/παρουσίασης μιας ανατεθειμένης πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	<pre>{ "date": "2023-01-01" }</pre>	HTTP 201 (Created)
/api/assignments/{id}/presentation	PUT	Τροποποιεί την ημερομηνία εξέτασης/παρουσίασης μιας ανατεθειμένης πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	<pre>{ "date": "2023-01-01" }</pre>	HTTP 200

/api/assignments/{id}/examination-committee	POST	Ορίζει τα μέλη της Τριμελούς Εξεταστικής Επιτροπής μιας ανατεθειμένης πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	`{ "examiner1": "string", "examiner2": "string", "examiner3": "string" }`	`HTTP 201 (Created)`
/api/assignments/{id}/examination-committee	PUT	Τροποποιεί τα μέλη της Τριμελούς Εξεταστικής Επιτροπής μιας ανατεθειμένης πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	`{ "examiner1": "string", "examiner2": "string", "examiner3": "string" }`	`HTTP 200`
/api/assignments/{id}/evaluation	PUT	Ορίζει ή τροποποιεί την αναλυτική βαθμολογία μιας ανατεθειμένης πτυχιακής / διπλωματικής εργασίας, βάσει του αριθμού ανάθεσης (id).	`{ "studentId": "string", "grade": decimal, "criteria": "string", "criteriaWeights": "string", "examiner1Scores": "string", "examiner2Scores": "string", "examiner3Scores": "string", "examiner1Comments": "string", "examiner2Comments": "string", "examiner3Comments": "string" }`	`HTTP 200`

Πίνακας 4.13 - Πίνακας κλήσεων του AssignmentsController

AuthenticationController:

Endpoint	HTTP Method	Περιγραφή	Request Body	Response Body
/auth/login?returnUrl=””	GET	Ανακατευθύνει τον χρήστη στη κεντρική υπηρεσία ταυτοποίησης. Μετά την ταυτοποίηση ο χρήστης επιστρέφει στην διεύθυνση που δηλώνεται στη παράμετρο “returnUrl”.	-	`HTTP 200`
/auth/logout	POST	Τερματίζει το session του συνδεδεμένου χρήστη.	-	`HTTP 200`
/auth/loggedOut	GET	Ανακατευθύνει τον χρήστη στην αρχική σελίδα.	-	`HTTP 200`
/auth/assertionConsumerService	POST	Λαμβάνει και επικυρώνει το απαντητικό SAML 2.0 μήνυμα από την κεντρική υπηρεσία ταυτοποίησης για την επιτυχή / ανεπιτυχή ταυτοποίηση του χρήστη.	-	`HTTP 200`
/auth/claims	GET	Επιστρέφει τα στοιχεία του συνδεδεμένου χρήστη.	-	`{ "id": "string", "firstName": "string", "lastName": "string", "email": "string", "department": "string", "academicId": "string", "academicRole": "string" }`

Πίνακας 4.14 - Πίνακας κλήσεων του AuthenticationController

4.4. Ταυτοποίηση Χρηστών

Στις παρακάτω ενότητες περιγράφεται το πρωτόκολλο αυθεντικοποίησης SAML 2.0 και πως αυτό εφαρμόστηκε στο πληροφοριακό σύστημα που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας.

4.4.1. Το πρωτόκολλο SAML 2.0

Το πρωτόκολλο Security Assertion Markup Language ή αλλιώς SAML, που αναπτύχθηκε από την Τεχνική Επιτροπή Υπηρεσιών Ασφαλείας του Οργανισμού για την Προώθηση των Προτύπων Δομημένης Πληροφορίας (Organization for the Advancement of Structured Information Standards - OASIS) [33], είναι ένα ανοιχτό πρότυπο που βασίζεται στην XML για την ανταλλαγή δεδομένων ελέγχου ταυτότητας και εξουσιοδότησης μεταξύ διαφορετικών συστημάτων. Το SAML 2.0, η δεύτερη έκδοση του πρωτοκόλλου, αποτελεί de facto πρότυπο για ασφαλή έλεγχο ταυτότητας και εξουσιοδότησης λόγω της στιβαρότητας και της ευρείας υιοθέτησής του.

Το SAML 2.0 επιτρέπει το Signle Sign-On (SSO), το οποίο επιτρέπει στους χρήστες να έχουν πρόσβαση σε πολλά συστήματα, χωρίς να απαιτείται να εισαγάγουν εκ νέου τους κωδικούς τους για κάθε σύστημα. Αυτό εξαλείφει την ανάγκη για πολλαπλές συνδέσεις και κωδικούς πρόσβασης, καθιστώντας τη διαδικασία ταυτοποίησης πιο απλή και ευκολότερη. Επιπλέον, το πρωτόκολλο παρέχει λεπτομερή έλεγχο πρόσβασης, δίνοντας στους διαχειριστές τη δυνατότητα να διαχειρίζονται τα δικαιώματα χρήστη για διαφορετικά συστήματα.

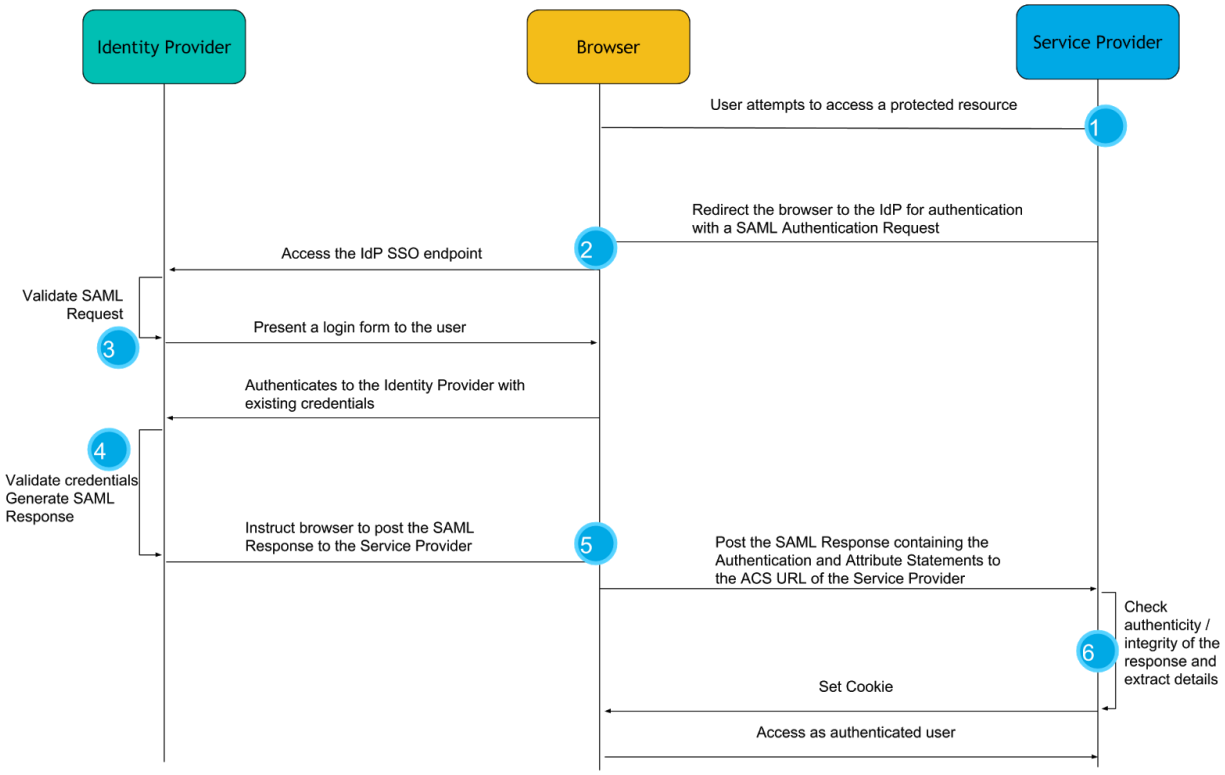
Το πρωτόκολλο SAML 2.0 αποτελείται από τρία κύρια στοιχεία, τον πάροχο ταυτότητας (Identity Provider - IdP), τον πάροχο υπηρεσιών (Service Provider - SP) και τον χρήστη [34]. Ο IdP είναι υπεύθυνος για ταυτοποίηση του χρήστη και τη δημιουργία ενός «ισχυρισμού» (assertion), δηλαδή ενός XML αρχείου που περιέχει πληροφορίες σχετικά με την ταυτότητα και τα δικαιώματα του, το οποίο αποστέλλει στον SP. Ο SP λαμβάνει αυτόν τον ισχυρισμό και παραχωρεί στον χρήστη πρόσβαση στο ζητούμενο σύστημα με βάση τις πληροφορίες που περιέχονται σε αυτό. Οι ισχυρισμοί πέραν της ταυτότητας του χρήστη, περιέχουν βασικές πληροφορίες όπως η ταυτότητα του IdP και μια ψηφιακή υπογραφή που επαληθεύει την αυθεντικότητα του ισχυρισμού. Μπορεί επίσης να περιέχουν πρόσθετες πληροφορίες, όπως οι ομάδες χρηστών που ανήκει ο χρήστης, τα δικαιώματα του και άλλα χαρακτηριστικά που μπορεί να χρειάζονται για τον έλεγχο πρόσβασης.

Η διαδικασία ταυτοποίησης χρήστη σύμφωνα με το πρωτόκολλο SAML 2.0 περιγράφεται αναλυτικότερα παρακάτω:

1. Αρχικά ένας χρήστης επιχειρεί να αποκτήσει πρόσβαση σε έναν προστατευμένο πόρο ενός SP. Ο SP, αναγνωρίζοντας ότι ο χρήστης δεν έχει ταυτοποιηθεί, ανακατευθύνει τον χρήστη στην υπηρεσία ελέγχου ταυτότητας του IdP.

2. Ο χρήστης εισάγει τα διαπιστευτήριά του (όνομα χρήστη και κωδικό πρόσβασης) στην υπηρεσία ελέγχου ταυτότητας (SSO) του IdP.
3. Ο IdP επαληθεύει τα διαπιστευτήρια του χρήστη και, εφόσον είναι έγκυρα, δημιουργεί έναν ισχυρισμό SAML που περιέχει πληροφορίες σχετικά με την ταυτότητα και τα χαρακτηριστικά του χρήστη. Αυτός ο ισχυρισμός υπογράφεται ψηφιακά από τον IdP για να διασφαλιστεί η αυθεντικότητα και η ακεραιότητά του.
4. Ο IdP αποστέλλει τον ισχυρισμό SAML πίσω στο πρόγραμμα περιήγησης του χρήστη ως απάντηση στο αίτημα ελέγχου ταυτότητας.
5. Το πρόγραμμα περιήγησης του χρήστη προωθεί τον ισχυρισμό SAML στην υπηρεσία του SP.
6. Ο SP λαμβάνει τον ισχυρισμό SAML και επικυρώνει τη γνησιότητά του ελέγχοντας την ψηφιακή υπογραφή και διασφαλίζοντας ότι προέρχεται από έναν αξιόπιστο IdP.
7. Ο SP δημιουργεί ένα session για τον χρήστη, επιτρέποντάς του να έχει πρόσβαση σε πολλούς πόρους εντός του οικοσυστήματος του SP χωρίς να χρειάζεται να επαναλάβει την διαδικασία ταυτοποίησης.
8. SP έχοντας επαληθεύσει με επιτυχία τον ισχυρισμό SAML, παραχωρεί πρόσβαση στον πόρο ή την υπηρεσία που ζητήθηκε, βασιζόμενος στην επαληθευμένη ταυτότητα του χρήστη.

Στην εικόνα 4.1 φαίνεται το διάγραμμα ροής της διαδικασίας ταυτοποίησης ενός χρήστη σύμφωνα με το πρωτόκολλο SAML 2.0.



Εικόνα 4.1 – Διαδικασία ταυτοποίησης χρήστη σύμφωνα με το SAML 2.0

Αξίζει να σημειωθεί, πως δημοφιλή παραδείγματα IdP αποτελούν οι πλατφόρμες Micosoft Azure Active Directory, Google Identity και Okta. Αντίστοιχα, παραδείγματα SP αποτελούν οι εφαρμογές web και κινητών συσκευών, οι υπηρεσίες cloud, και τα διάφορα ειδών APIs. Στη περίπτωση του Π.Σ. υποδομής που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας, ως IdP χρησιμοποιήθηκε η πλατφόρμα Okta, ενώ το REST API που υλοποιήθηκε αποτελεί τον SP.

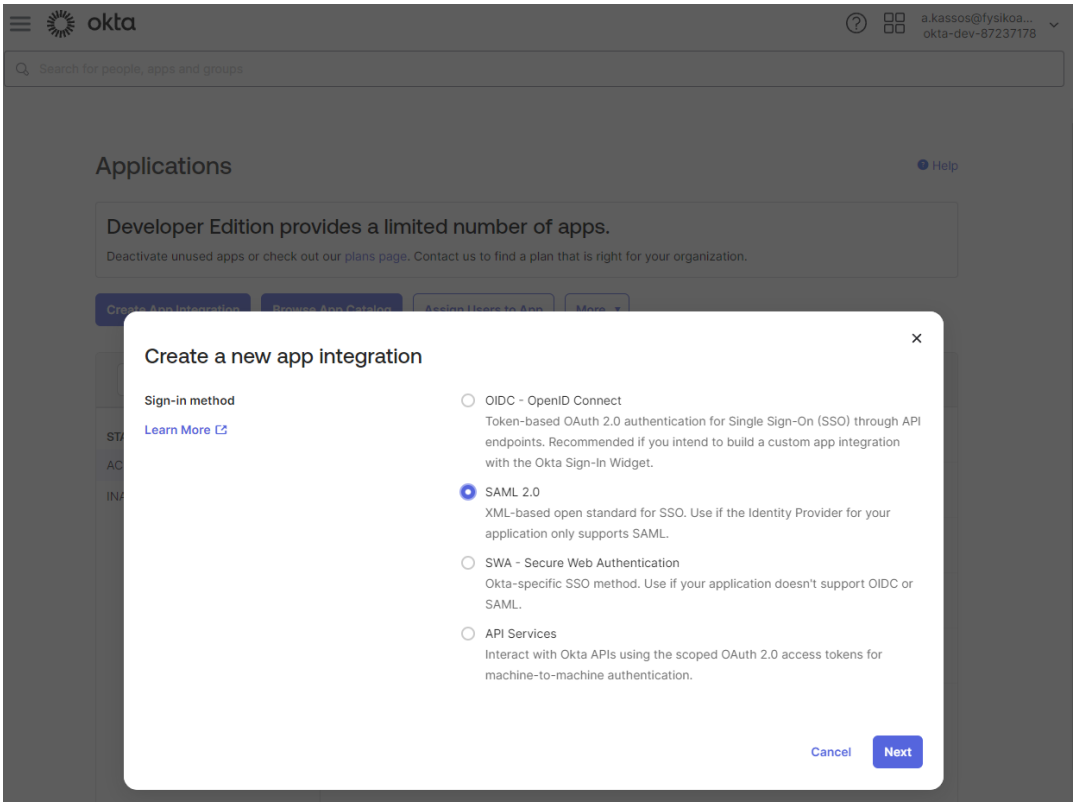
4.4.2. Η πλατφόρμα Okta ως Identity Provider

Η Okta είναι μια εταιρεία που παρέχει λύσεις διαχείρισης ταυτότητας και πρόσβασης (Identity and Access Management - IAM) για επιχειρήσεις και οργανισμούς. Το IAM είναι ένα κρίσιμο στοιχείο της κυβερνοασφάλειας και περιλαμβάνει τη διαχείριση και τον έλεγχο της πρόσβασης σε διάφορους ψηφιακούς πόρους και υπηρεσίες εντός ενός οργανισμού. Η πλατφόρμα της Okta προσφέρει μια σειρά από λειτουργίες και υπηρεσίες που βοηθούν τις επιχειρήσεις και τους οργανισμούς να προστατεύουν τα συστήματα και τα δεδομένα τους, ενώ παράλληλα διευκολύνει τους υπαλλήλους, τους πελάτες και τους συνεργάτες να έχουν πρόσβαση στους πόρους που χρειάζονται.

Στα πλαίσια ανάπτυξης του πληροφοριακού συστήματος υποδομής για τις ανάγκες της διπλωματικής εργασίες, χρησιμοποιήθηκε η πλατφόρμα της Okta ως κεντρική υπηρεσία ταυτοποίησης, προσομοιώνοντας κατά αυτό το τρόπο την αντίστοιχη υπηρεσία ταυτοποίησης που χρησιμοποιεί το Πανεπιστήμιο Δυτικής Αττικής για τους φοιτητές και το προσωπικό του. Πιο συγκεκριμένα, δημιουργήθηκε μέσω της πλατφόρμας της Okta ένας custom SAML 2.0 Identity Provider, ώστε να μπορεί να διασυνδεθεί το Π.Σ. υποδομής που αναπτύχθηκε ως Service Provider και να εφαρμοστεί το πρωτόκολλο SAML 2.0 [35].

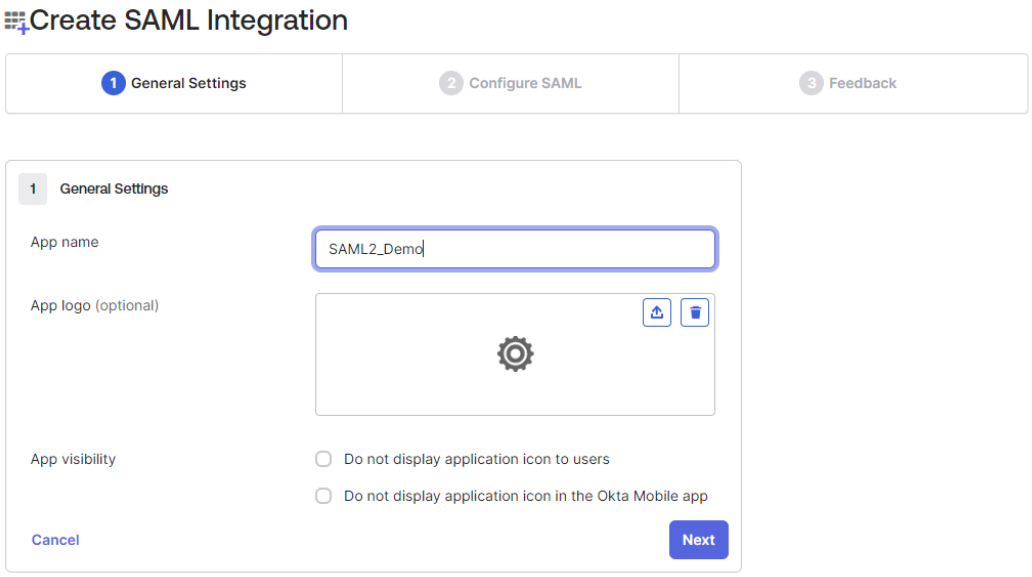
Η πλατφόρμα της Okta επιλέχτηκε κυρίως διότι υποστηρίζει πλήρως το SAML 2.0 με δυνατότητα SSO, αλλά και λόγω των πολλών διευκολύνσεων που προσφέρει, όπως ότι μπορούμε να ορίσουμε custom attributes στους χρήστες για τις ανάγκες της εφαρμογής, ή ότι οι χρήστες μπορούν να γίνουν self-registered στον IdP με οποιοδήποτε προσωπικό τους email. Επιπλέον, η διαδικασία δημιουργίας και παραμετροποίησης ενός SAML 2.0 IdP στη πλατφόρμα της Okta είναι πολύ απλή και εύκολη [36]. Παρακάτω, περιγράφεται η διαδικασία που ακολουθήθηκε για τη δημιουργία του IdP και τη διασύνδεση με το Π.Σ. υποδομής:

Αρχικά, έχοντας κάνει εγγραφή στην πλατφόρμα της Okta, αποκτάμε πρόσβαση στο περιβάλλον για προγραμματιστές. Έπειτα, δημιουργούμε μια νέα διασύνδεση εφαρμογής έχοντας επιλέξει το SAML 2.0 ως μέθοδο σύνδεσης:



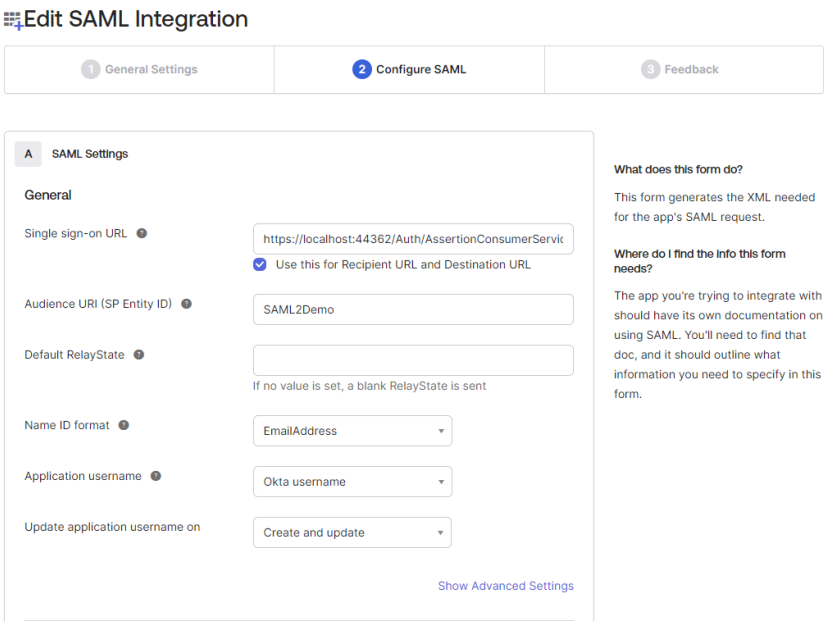
Εικόνα 4.2 – Δημιουργία νέας εφαρμογής διασύνδεσης στη πλατφόρμα Okta

Θέτουμε ένα όνομα εφαρμογής και συνεχίζουμε στο επόμενο βήμα.



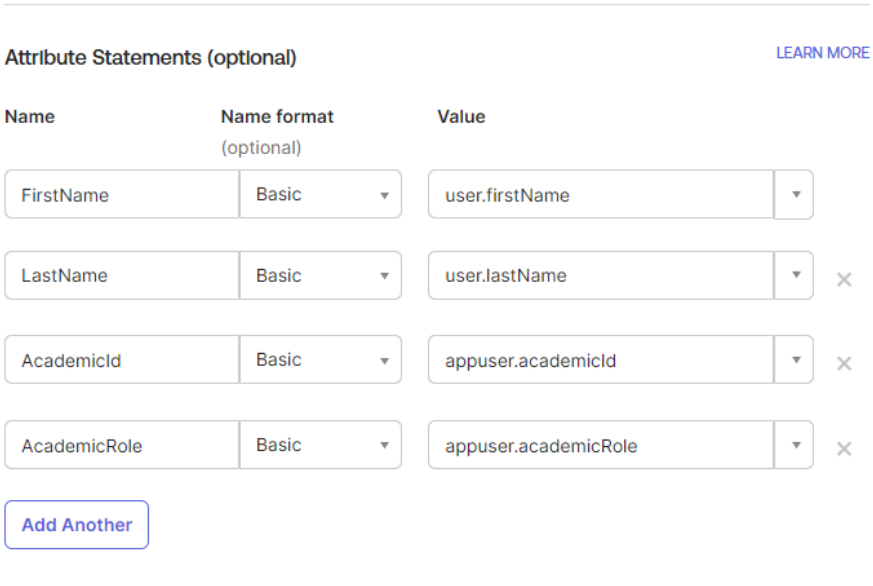
Εικόνα 4.3 – Ορισμός ονόματος εφαρμογής διασύνδεσης στη πλατφόρμα Okta

Ορίζουμε στο Single Sign-On URL το endpoint που είναι υπεύθυνο για την λήψη και την επικύρωση του SAML 2.0 απαντητικού του IdP, δηλαδή το …/auth/assertionConsumerService endpoint του REST API, καθώς και ένα ενδεικτικό Audience URI:



Εικόνα 4.4 – Παραμετροποίηση SAML 2.0 IdP στη πλατφόρμα Okta

Στη συνέχεια, ορίζουμε τα απαραίτητα custom attributes για τις ανάγκες του Π.Σ. υποδομής, όπως για παράδειγμα ο αριθμός μητρώου, το όνομα, το επίθετο και ο ρόλος (φοιτητής/καθηγητής) του χρήστη:



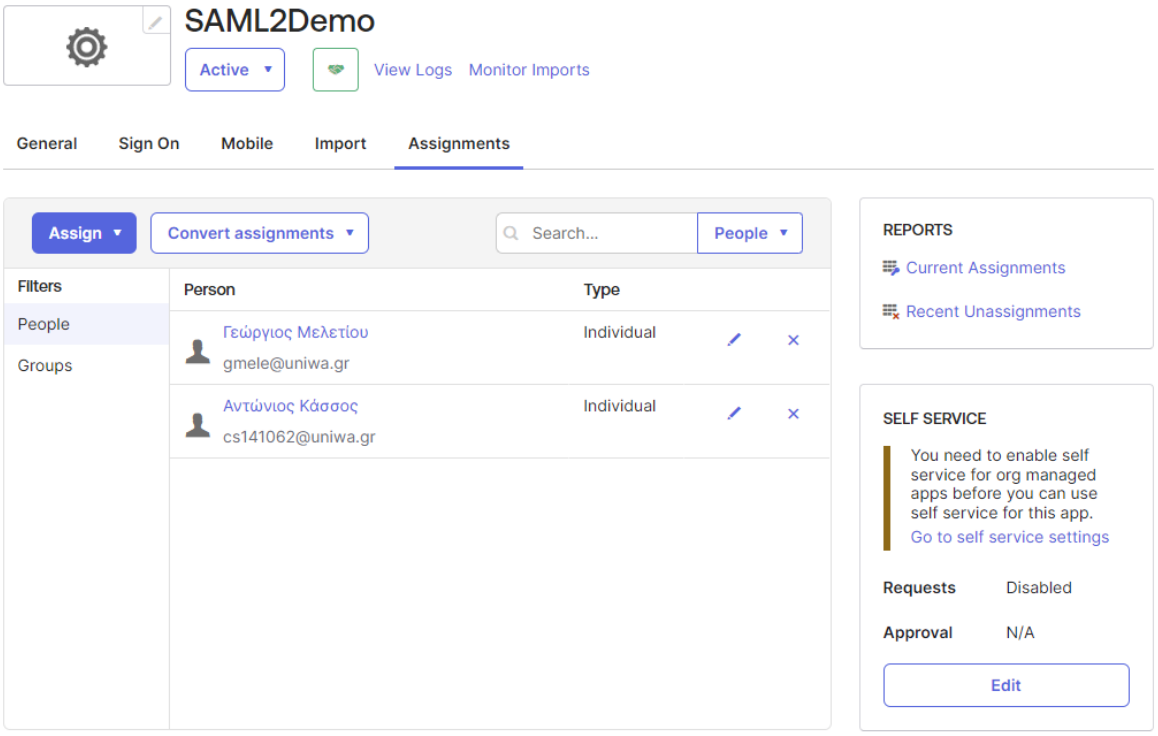
Εικόνα 4.5 – Ορισμός προσαρμοσμένων χαρακτηριστικών χρήστη

Σε αυτό το σημείο, δημιουργείται ένας σύνδεσμος με τα απαραίτητα metadata του IdP που χρειάζεται να γνωρίζει ο SP, δηλαδή η το Π.Σ. υποδομής, προκειμένου να μπορούν να επικοινωνήσουν, ο οποίος δηλώνεται στο αρχείο παραμετροποίησης appSettings.json του κώδικα της εφαρμογής:

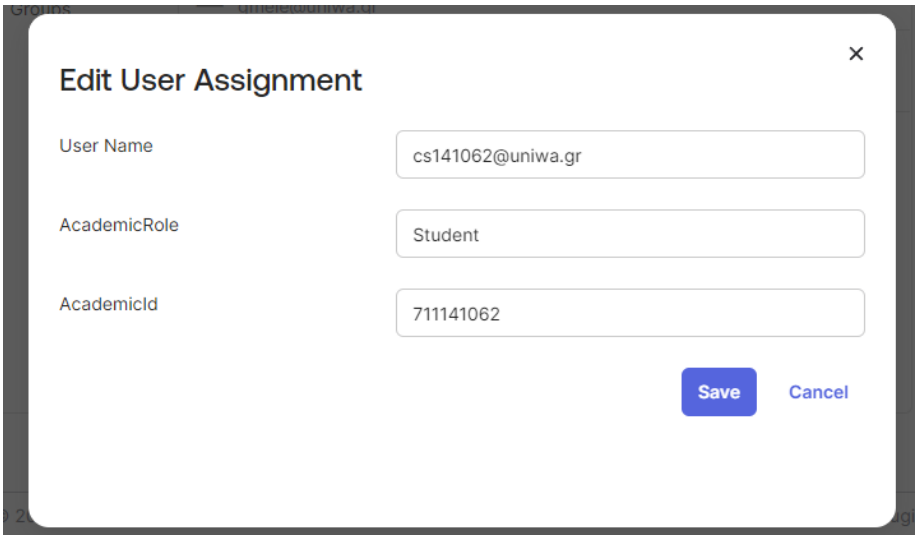
```
"Saml2": {
  // Okta IDP
  "IdPMetadata": "https://dev-87237178.okta.com/app/exk1i8nguzKVB3PjB5d7/sso/saml/metadata",
  "Issuer": "SAML2Demo",
  "SignatureAlgorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
  // Use only with a valid certificate
  //"SigningCertificateFile": "certificate.pfx",
  //"SigningCertificatePassword": "******",
  //"SignatureValidationCertificateFile": "certificate.cer",
  "CertificateValidationMode": "ChainTrust",
  "RevocationMode": "NoCheck"
}
}
```

Εικόνα 4.6 – Δήλωση των IdP Metadata στον κώδικα της εφαρμογής

Τέλος, προσθέτουμε τους χρήστες που μπορούν να χρησιμοποιήσουν αυτή τη νέα διασύνδεση και συμπληρώνουμε τιμές για τα custom attributes που δημιουργήσαμε για τις ανάγκες της εφαρμογής:



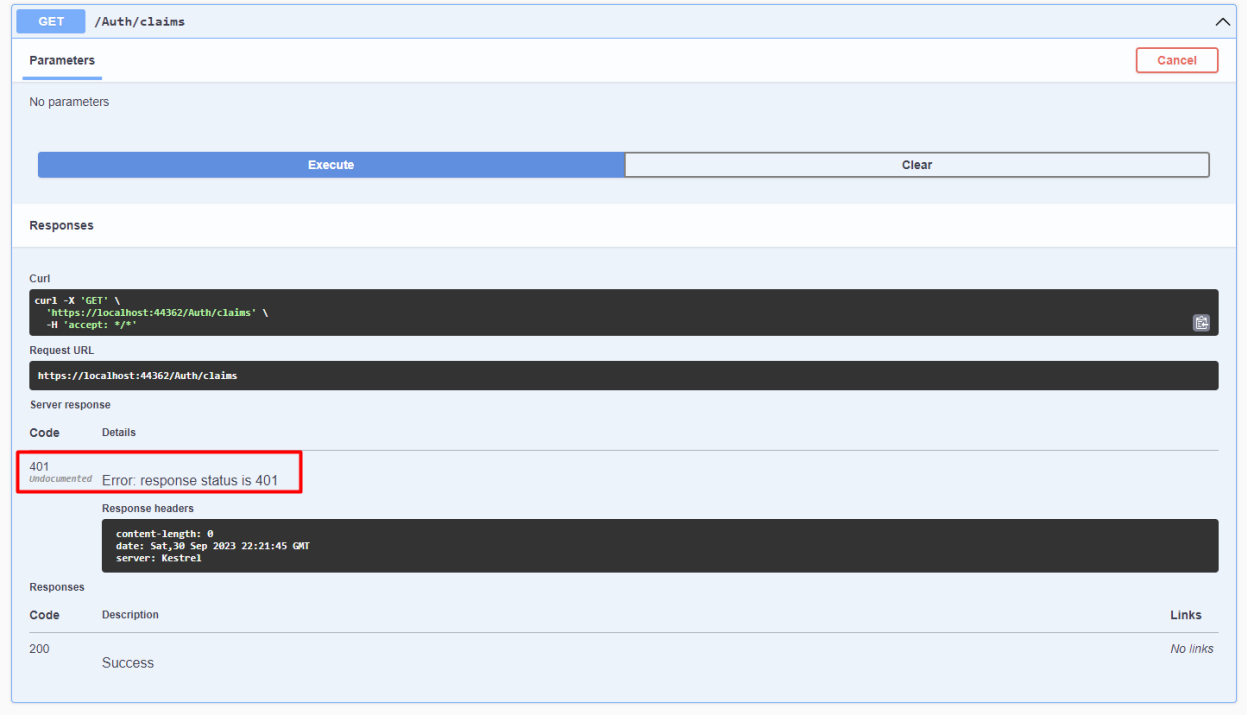
Εικόνα 4.7 – Προσθήκη χρήστη στον IdP



Εικόνα 4.8 – Συμπλήρωση προσαρμοσμένων χαρακτηριστικών χρήστη

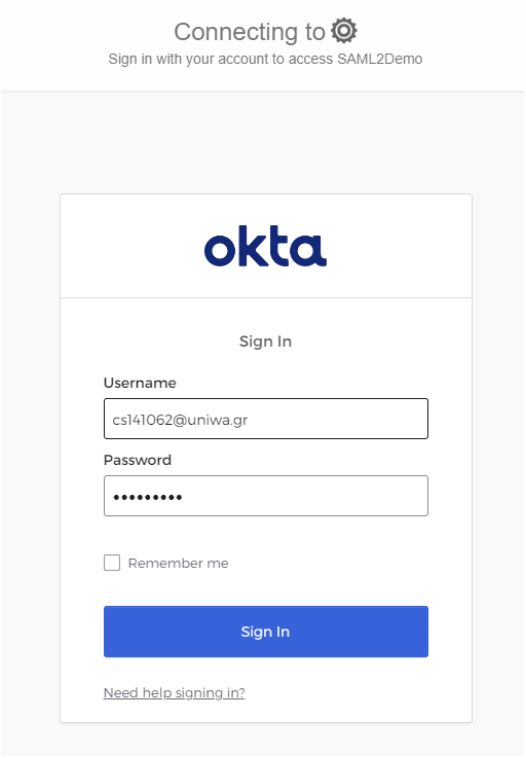
4.4.3. Σύνδεση χρήστη μέσω του SSO

Προκειμένου ένας χρήστης να αποκτήσει πρόσβαση και να μπορεί να αλληλεπιδράσει με το REST API της εφαρμογής, θα πρέπει να έχει συνδεθεί μέσω του SSO του Identity Provider. Σε διαφορετική περίπτωση, δεν έχει το δικαίωμα να καλέσει τα endpoints του REST API και λαμβάνει HTTP απάντηση με «Status 401 – Unauthorized».

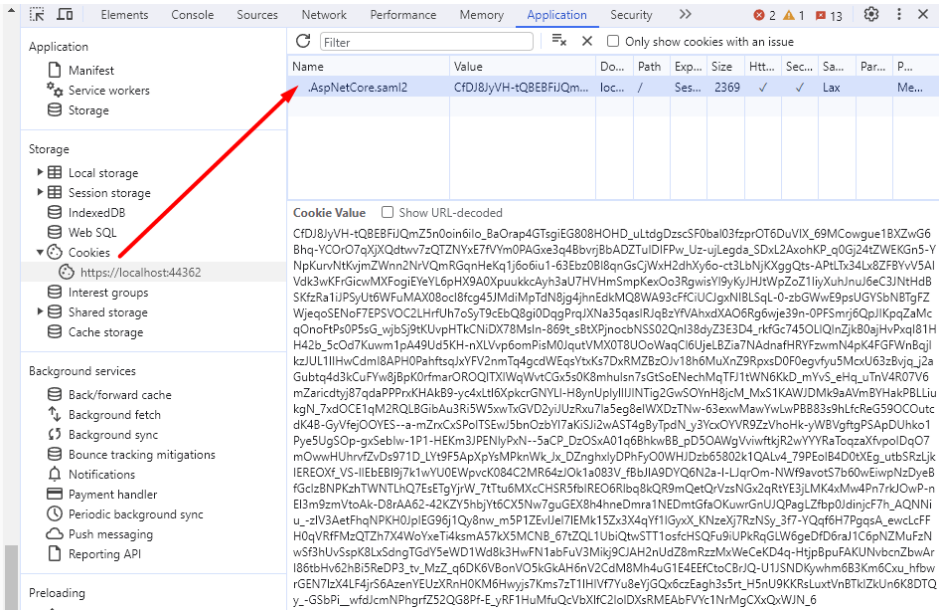


Εικόνα 4.9 – Με εξουσιοδοτημένη κλήση προς το REST API

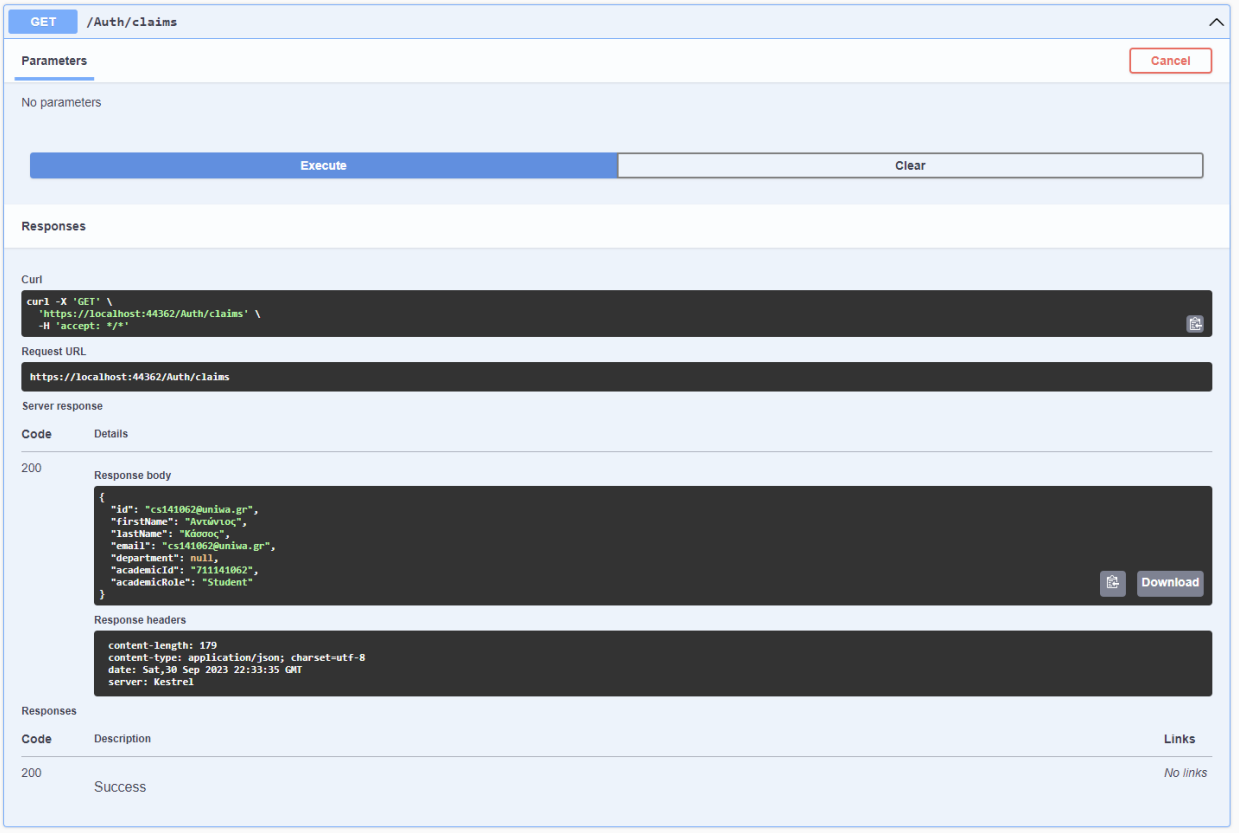
Οι χρήστες που συνδέονται επιτυχώς μέσω του SSO (Εικόνα 4.10) στον IdP, αποκτούν ένα Session Cookie (Εικόνα 4.11), που τους δίνει την απαραίτητη πρόσβαση προκειμένου να καλούν τα endpoints του REST API της εφαρμογής. Επίσης, όπως φαίνεται στην Εικόνα 4.12, η εφαρμογή ή αλλιώς ο Service Provider, έχει πλέον αποκτήσει πρόσβαση στα custom attributes του χρήστη που δημιουργήσαμε για τις ανάγκες του Π.Σ. υποδομής.



Εικόνα 4.10 – Σύνδεση χρήστη μέσω Single Sign-On



Εικόνα 4.11 – Session Cookie συνδεδεμένου χρήστη



Εικόνα 4.12 – Εξουσιοδοτημένη κλήση προς το REST API

Κεφάλαιο 5 – Συμπεράσματα

Στη παρούσα Διπλωματική εργασία αναλύσαμε την υφιστάμενη διαδικασία που ακολουθείται από το Πανεπιστήμιο Δυτικής Αττικής όσον αφορά τις Πτυχιακές και Διπλωματικές εργασίες των φοιτητών, διαπιστώνοντας μια σειρά από προβλήματα και δυσκολίες. Για την επίλυση των προβλημάτων αυτών, καθώς και για την απλοποίηση της διαδικασίας αλλά και για την καλύτερη μηχανοργάνωση του Τμήματος Μηχανικών Πληροφορικής, σχεδιάστηκε και αναπτύχθηκε ένα Πληροφοριακό Σύστημα υποδομής παροχής υπηρεσιών μέσω διαδικτύου.

Μέσω του Πληροφοριακού Συστήματος που αναπτύχθηκε, επιτυγχάνεται η ηλεκτρονική καταχώρηση θεμάτων Πτυχιακών και Διπλωματικών εργασιών από τους καθηγητές σε μια κεντρική βάση δεδομένων, με σκοπό τη δημιουργία ενός ηλεκτρονικού πίνακα θεμάτων που θα έχουν πρόσβαση ανά πάσα στιγμή όλοι οι φοιτητές και οι καθηγητές του Τμήματος. Παράλληλα, παρέχει τη δυνατότητα ηλεκτρονικής υποβολής και εκτύπωσης αιτήσεων από τους φοιτητές, όπως η Αίτηση Ανάληψης ενός θέματος, αλλά και την ανάθεση των επιμέρους θεμάτων από τους υπεύθυνους καθηγητές. Επιπλέον, το Πληροφοριακό Σύστημα υποστηρίζει την πλήρη διαχείριση των ανατεθειμένων εργασιών από τους καθηγητές, για την καλύτερη οργάνωση και παρακολούθηση της πορείας εξέλιξης τους. Η διαχείριση των εργασιών περιλαμβάνει λειτουργίες όπως τον ορισμό διαφορετικών καταστάσεων εξέλιξης, τον ορισμό Τριμελούς Εξεταστικής Επιτροπής και ημερομηνίας εξέτασης, την αναλυτική βαθμολόγηση και την παράταση της προθεσμίας ολοκλήρωσης.

Όλες οι λειτουργίες του Πληροφοριακού Συστήματος είναι διαθέσιμες μέσω μίας διαδικτυακής υπηρεσίας τύπου REST, επιτρέποντας τη διασύνδεση του με υπάρχουσα συστήματα και εφαρμογές του Τμήματος, με σκοπό τη δημιουργία μιας ολοκληρωμένης διαδικτυακής πλατφόρμας για την διαχείριση των Πτυχιακών και Διπλωματικών εργασιών των φοιτητών του Τμήματος Μηχανικών Πληροφορικής και Υπολογιστών.

5.1. Μελλοντικές προσθήκες και βελτιώσεις

Το σύστημα υποδομής που αναπτύχθηκε στα πλαίσια της Διπλωματικής εργασίας, παρέχει τα θεμέλια για δημιουργία ενός ολοκληρωμένου συστήματος που θα μπορεί να αξιοποιηθεί από το Τμήμα Μηχανικών Πληροφορικής. Για παράδειγμα, σύστημα υποστηρίζει το πρωτόκολλο ταυτοποίησης χρηστών SAML 2.0, που χρησιμοποιείται από την κεντρική υπηρεσία αυθεντικοποίησης χρηστών του Πανεπιστημίου. Μια πολύ σημαντική προσθήκη θα ήταν η διασύνδεση του συστήματος με τη κεντρική υπηρεσία αυθεντικοποίησης, ώστε οι χρήστες να συνδέονται με τους κωδικούς χρήστη που τους παρέχει το Πανεπιστήμιο. Εξίσου σημαντική προσθήκη, θα ήταν η αξιοποίηση της διαδικτυακής υπηρεσίας τύπου REST από μια Front-End εφαρμογή, μέσω της οποίας οι χρήστες θα μπορούν να αλληλεπιδρούν με το σύστημα.

Το υλοποιημένο σύστημα υποδομής επιδέχεται ορισμένες βελτιώσεις σε μερικές λειτουργίες του. Για παράδειγμα, θα μπορούσε να χρησιμοποιηθεί ένας SMTP Server για την

αποστολή ηλεκτρονικών μηνυμάτων (email), σε λειτουργίες όπως η ηλεκτρονική υποβολή της Αίτησης Ανάληψης θέματος από τους φοιτητές, ώστε να αποστέλλονται αυτόματα προς τη Γραμματεία του Τμήματος. Επιπλέον, η δυνατότητα ηλεκτρονικής υπογραφής των αιτήσεων θα μπορούσε βελτιώσει σημαντικά την εμπειρία των χρηστών και να απλοποιήσει ακόμα περισσότερο τη διαδικασία αίτησης και ανάληψης. Τέλος, αξίζει να σημειωθεί πως το σύστημα υποδομής υποστηρίζει κυρίως λειτουργίες που αφορούν τις ανάγκες των φοιτητών και καθηγητών, ωστόσο θα μπορούσε κάλλιστα να υποστηρίξει λειτουργίες που αφορούν και την Γραμματεία του Τμήματος, όπως για παράδειγμα η πρωτοκόλληση των αιτήσεων των φοιτητών.

Παράρτημα Α

Κώδικας SQL - Πίνακες

Πίνακας ThesisProjects

```
CREATE TABLE [dbo].[ThesisProjects](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [OwnerId] [nvarchar](20) NULL,
    [Owner] [nvarchar](50) NULL,
    [TitleInGreek] [nvarchar](250) NULL,
    [TitleInEnglish] [nvarchar](250) NULL,
    [Description] [nvarchar](1000) NULL,
    [Requirements] [nvarchar](250) NULL,
    [DesiredKnowledge] [nvarchar](250) NULL,
    [Bibliography] [nvarchar](1000) NULL,
    [Curriculum] [nvarchar](20) NULL,
    [Type] [nvarchar](10) NULL,
    [Status] [nvarchar](25) NULL,
    [DateTimeCreated] [datetime2](7) NOT NULL,
    [DateTimeUpdated] [datetime2](7) NOT NULL,
 CONSTRAINT [PK_ThesisProjects] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Πίνακας ThesisApplications

```
CREATE TABLE [dbo].[ThesisApplications](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [AcademicId] [nvarchar](20) NULL,
    [FirstName] [nvarchar](25) NULL,
    [LastName] [nvarchar](25) NULL,
    [FathersName] [nvarchar](25) NULL,
    [Phone] [nvarchar](20) NULL,
    [Email] [nvarchar](50) NULL,
    [CurrentSemester] [nvarchar](5) NULL,
    [Department] [nvarchar](100) NULL,
    [Specialization] [nvarchar](100) NULL,
    [ProjectTitleGreek] [nvarchar](250) NULL,
    [ProjectTitleEnglish] [nvarchar](250) NULL,
    [Description] [nvarchar](1000) NULL,
    [ResponsibleProfessor] [nvarchar](50) NULL,
    [Status] [nvarchar](25) NULL,
    [Date] [nvarchar](10) NULL,
    [ProtocolId] [nvarchar](20) NULL,
 CONSTRAINT [PK_ThesisApplications] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Πίνακας Assignments

```
CREATE TABLE [dbo].[Assignments](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [ThesisProjectId] [int] NOT NULL,
    [AssignedStudentId] [nvarchar](20) NULL,
    [Status] [nvarchar](10) NULL,
    [DateTimeCreated] [datetime2](7) NOT NULL,
    [DateTimeUpdated] [datetime2](7) NOT NULL,
    [AssignedStudent] [nvarchar](50) NULL,
    [DateTimeExpires] [datetime2](7) NOT NULL,
    [SecondAssignedStudent] [nvarchar](50) NULL,
    [SecondAssignedStudentId] [nvarchar](20) NULL,
 CONSTRAINT [PK_Assignments] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Assignments] ADD  DEFAULT ('0001-01-01T00:00:00.0000000') FOR
[DateTimeExpires]
GO

ALTER TABLE [dbo].[Assignments]  WITH CHECK ADD  CONSTRAINT
[FK_Assignments_ThesisProjects_ThesisProjectId] FOREIGN KEY([ThesisProjectId])
REFERENCES [dbo].[ThesisProjects] ([Id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Assignments] CHECK CONSTRAINT
[FK_Assignments_ThesisProjects_ThesisProjectId]
GO
```

Πίνακας AssignmentsHistory

```
CREATE TABLE [dbo].[AssignmentsHistory](
    [Id] [bigint] IDENTITY(1,1) NOT NULL,
    [AssignmentId] [int] NOT NULL,
    [PreviousState] [nvarchar](20) NULL,
    [NewState] [nvarchar](20) NULL,
    [Reason] [nvarchar](250) NULL,
    [TimeStamp] [datetime2](7) NOT NULL,
 CONSTRAINT [PK_AssignmentsHistory] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[AssignmentsHistory]  WITH CHECK ADD  CONSTRAINT
[FK_AssignmentsHistory_Assignments_AssignmentId] FOREIGN KEY([AssignmentId])
REFERENCES [dbo].[Assignments] ([Id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[AssignmentsHistory] CHECK CONSTRAINT
[FK_AssignmentsHistory_Assignments_AssignmentId]
GO
```

Πίνακας ExaminationCommittees

```
CREATE TABLE [dbo].[ExaminationCommitees](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [AssignmentId] [int] NOT NULL,
    [Examiner1] [nvarchar](50) NULL,
    [Examiner2] [nvarchar](50) NULL,
    [Examiner3] [nvarchar](50) NULL,
 CONSTRAINT [PK_ExaminationCommitees] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[ExaminationCommitees]  WITH CHECK ADD  CONSTRAINT
[FK_ExaminationCommitees_Assignments_AssignmentId] FOREIGN KEY([AssignmentId])
REFERENCES [dbo].[Assignments] ([Id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[ExaminationCommitees] CHECK CONSTRAINT
[FK_ExaminationCommitees_Assignments_AssignmentId]
GO
```

Πίνακας Examinations

```
CREATE TABLE [dbo].[Examinations](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [AssignmentId] [int] NOT NULL,
    [PresenationDate] [datetime2](7) NOT NULL,
    [DateTimeCreated] [datetime2](7) NOT NULL,
    [DateTimeUpdated] [datetime2](7) NOT NULL,
    [Grade] [decimal](3, 1) NULL,
    [Criteria] [nvarchar](250) NULL,
    [CriteriaWeights] [nvarchar](50) NULL,
    [Examiner1Comments] [nvarchar](100) NULL,
    [Examiner2Comments] [nvarchar](100) NULL,
    [Examiner3Comments] [nvarchar](100) NULL,
    [StudentId] [nvarchar](20) NULL,
    [Examiner1Scores] [nvarchar](50) NULL,
    [Examiner2Scores] [nvarchar](50) NULL,
    [Examiner3Scores] [nvarchar](50) NULL,
 CONSTRAINT [PK_Examinations] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Examinations]  WITH CHECK ADD  CONSTRAINT
[FK_Examinations_Assignments_AssignmentId] FOREIGN KEY([AssignmentId])
REFERENCES [dbo].[Assignments] ([Id])
ON DELETE CASCADE
GO

ALTER TABLE [dbo].[Examinations] CHECK CONSTRAINT
[FK_Examinations_Assignments_AssignmentId]
GO
```

Πίνακας Courses

```
CREATE TABLE [dbo].[Courses](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Code] [nvarchar](20) NULL,
    [Title] [nvarchar](100) NULL,
    [Department] [nvarchar](100) NULL,
    [Curriculum] [nvarchar](50) NULL,
    [Period] [nvarchar](10) NULL,
    [Type] [nvarchar](25) NULL,
 CONSTRAINT [PK_Courses] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Πίνακας Lookup

```
CREATE TABLE [dbo].[LookupTable](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Key] [nvarchar](50) NULL,
    [Value] [nvarchar](250) NULL,
 CONSTRAINT [PK_LookupTable] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Πίνακας EFMigrationsHistory

```
CREATE TABLE [dbo].[__EFMigrationsHistory](
    [MigrationId] [nvarchar](150) NOT NULL,
    [ProductVersion] [nvarchar](32) NOT NULL,
 CONSTRAINT [PK___EFMigrationsHistory] PRIMARY KEY CLUSTERED
(
    [MigrationId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
```

Παράρτημα Β

Κώδικας C# - Models & Classes

Assignment.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class Assignment
    {
        public int Id { get; set; }
        public int ThesisProjectId { get; set; }
        [MaxLength(20)]
        public string AssignedStudentId { get; set; }
        [MaxLength(50)]
        public string AssignedStudent { get; set; }
        [MaxLength(20)]
        public string SecondAssignedStudentId { get; set; }
        [MaxLength(50)]
        public string SecondAssignedStudent { get; set; }

        [MaxLength(10)]
        public string Status { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeUpdated { get; set; }
        public DateTime DateTimeExpires { get; set; }

        public ThesisProject ThesisProject { get; set; }
        public ExaminationCommitee ExaminationCommitee { get; set; }
        public ICollection<Examination> Examinations { get; set; }
        public ICollection<AssignmentHistory> StatusChanges { get; set; }
    }
}
```

AssignmentsHistory.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class AssignmentHistory
    {
        public long Id { get; set; }
        public int AssignmentId { get; set; }
        [MaxLength(20)]
        public string PreviousState { get; set; }
        [MaxLength(20)]
        public string NewState { get; set; }
        [MaxLength(250)]
        public string Reason { get; set; }
```

```
        public DateTime TimeStamp { get; set; }
    }
}
```

AssignmentStatus.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class AssignmentStatus
    {
        public string NewStatus { get; set; }
        public string Reason { get; set; }
    }
}
```

Course.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class Course
    {
        public int Id { get; set; }
        [MaxLength(20)]
        public string Code { get; set; }
        [MaxLength(100)]
        public string Title { get; set; }
        [MaxLength(100)]
        public string Department { get; set; }
        [MaxLength(50)]
        public string Curriculum { get; set; }
        [MaxLength(10)]
        public string Period { get; set; }
        [MaxLength(25)]
        public string Type { get; set; }
    }
}
```

Examination.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class Examination
    {
        public int Id { get; set; }
```

```
        public int AssignmentId { get; set; }
        [MaxLength(20)]
        public string StudentId { get; set; }
        [Column(TypeName = "decimal(3,1)")]
        public decimal? Grade { get; set; }
        [MaxLength(250)]
        public string Criteria { get; set; }
        [MaxLength(50)]
        public string CriteriaWeights { get; set; }
        [MaxLength(50)]
        public string Examiner1Scores { get; set; }
        [MaxLength(50)]
        public string Examiner2Scores { get; set; }
        [MaxLength(50)]
        public string Examiner3Scores { get; set; }
        [MaxLength(100)]
        public string Examiner1Comments { get; set; }
        [MaxLength(100)]
        public string Examiner2Comments { get; set; }
        [MaxLength(100)]
        public string Examiner3Comments { get; set; }
        public DateTime PresenationDate { get; set; }
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeUpdated { get; set; }
    }
}
```

ExaminationCommittee.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class ExaminationCommitee
    {
        public int Id { get; set; }
        public int AssignmentId { get; set; }
        [MaxLength(50)]
        public string Examiner1 { get; set; }
        [MaxLength(50)]
        public string Examiner2 { get; set; }
        [MaxLength(50)]
        public string Examiner3 { get; set; }
    }
}
```

ExaminationCommitteeDto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class ExaminationCommiteeDto
    {
        public string Examiner1 { get; set; }
        public string Examiner2 { get; set; }
```

```
        public string Examiner3 { get; set; }
    }
}
```

LookupValue.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class LookupValue
    {
        public int Id { get; set; }
        [MaxLength(50)]
        public string Key { get; set; }
        [MaxLength(250)]
        public string Value { get; set; }
    }
}
```

PresentationDate.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class PresentationDate
    {
        public DateTime Date { get; set; }
    }
}
```

Specialization.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class Specialization
    {
        public int Id { get; set; }
        public string Value { get; set; }
    }
}
```

Status.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace ThesisManagement.Models
{
    public class Status
    {
        public int Id { get; set; }
        public string Value { get; set; }
    }
}
```

ThesisApplication.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class ThesisApplication
    {
        public int Id { get; set; }
        [MaxLength(20)]
        public string AcademicId { get; set; }
        [MaxLength(25)]
        public string FirstName { get; set; }
        [MaxLength(25)]
        public string LastName { get; set; }
        [MaxLength(25)]
        public string FathersName { get; set; }
        [MaxLength(20)]
        public string Phone { get; set; }
        [MaxLength(50)]
        public string Email { get; set; }
        [MaxLength(5)]
        public string CurrentSemester { get; set; }
        [MaxLength(100)]
        public string Department { get; set; }
        [MaxLength(100)]
        public string Specialization { get; set; }
        [MaxLength(250)]
        public string ProjectTitleGreek { get; set; }
        [MaxLength(250)]
        public string ProjectTitleEnglish { get; set; }
        [MaxLength(1000)]
        public string Description { get; set; }
        [MaxLength(50)]
        public string ResponsibleProfessor { get; set; }
        [MaxLength(25)]
        public string Status { get; set; }
        [MaxLength(10)]
        public string Date { get; set; }
        [MaxLength(20)]
        public string ProtocolId { get; set; }
    }
}
```

ThesisApplicationPDF.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
namespace ThesisManagement.Models
{
    public class ThesisApplicationPDF
    {
        public string Base64 { get; set; }
    }
}
```

ThesisEvaluation.cs

```
namespace ThesisManagement.Models
{
    public class ThesisEvaluation
    {
        public string StudentId { get; set; }
        public decimal Grade { get; set; }
        public string Criteria { get; set; }
        public string CriteriaWeights { get; set; }
        public string Examiner1Scores { get; set; }
        public string Examiner2Scores { get; set; }
        public string Examiner3Scores { get; set; }
        public string Examiner1Comments { get; set; }
        public string Examiner2Comments { get; set; }
        public string Examiner3Comments { get; set; }
    }
}
```

ThesisProject.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class ThesisProject
    {
        public int Id { get; set; }
        [MaxLength(20)]
        public string OwnerId { get; set; }
        [MaxLength(50)]
        public string Owner { get; set; }
        [MaxLength(250)]
        public string TitleInGreek { get; set; }
        [MaxLength(250)]
        public string TitleInEnglish { get; set; }
        [MaxLength(1000)]
        public string Description { get; set; }
        [MaxLength(250)]
        public string Requirements { get; set; }
        [MaxLength(250)]
        public string DesiredKnowledge { get; set; }
        [MaxLength(1000)]
        public string Bibliography { get; set; }
        [MaxLength(20)]
        public string Curriculum { get; set; }
        [MaxLength(10)]
        public string Type { get; set; }
        [MaxLength(25)]
        public string Status { get; set; }
```

```
        public DateTime DateTimeCreated { get; set; }
        public DateTime DateTimeUpdated { get; set; }

        public ICollection<Assignment> Assignments { get; set; }
    }
}
```

UpdateProjectStatusDto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class UpdateProjectStatusDto
    {
        public string ProjectId { get; set; }
        public string Status { get; set; }
    }
}
```

UserClaims.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ThesisManagement.Models
{
    public class UserClaims
    {
        public string Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
        public string Department { get; set; }
        public string AcademicId { get; set; }
        public string AcademicRole { get; set; }

        public UserClaims()
        {

        }

        public UserClaims(string id, string firstName, string lastName, string email,
string department, string academicId, string academicRole)
        {
            Id = id;
            FirstName = firstName;
            LastName = lastName;
            Email = email;
            Department = department;
            AcademicId = academicId;
            AcademicRole = academicRole;
        }
    }
}
```

ClaimsTransform.cs

```
using ITfoxtec.Identity.Saml2.Claims;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace ThesisManagement.Identity
{
    public static class ClaimsTransform
    {
        public static ClaimsPrincipal Transform(ClaimsPrincipal incomingPrincipal)
        {
            if (!incomingPrincipal.Identity.IsAuthenticated)
            {
                return incomingPrincipal;
            }

            return CreateClaimsPrincipal(incomingPrincipal);
        }

        private static ClaimsPrincipal CreateClaimsPrincipal(ClaimsPrincipal
incomingPrincipal)
        {
            var claims = new List<Claim>();

            // All claims
            claims.AddRange(incomingPrincipal.Claims);

            // Or custom claims
            //claims.AddRange(GetSaml2LogoutClaims(incomingPrincipal));
            //claims.Add(new Claim(ClaimTypes.NameIdentifier,
GetClaimValue(incomingPrincipal, ClaimTypes.NameIdentifier)));

            return new ClaimsPrincipal(new ClaimsIdentity(claims,
incomingPrincipal.Identity.AuthenticationType, ClaimTypes.NameIdentifier,
ClaimTypes.Role)
            {
                BootstrapContext =
((ClaimsIdentity)incomingPrincipal.Identity).BootstrapContext
            });
        }

        private static IEnumerable<Claim> GetSaml2LogoutClaims(ClaimsPrincipal
principal)
        {
            yield return GetClaim(principal, Saml2ClaimTypes.NameId);
            yield return GetClaim(principal, Saml2ClaimTypes.NameIdFormat);
            yield return GetClaim(principal, Saml2ClaimTypes.SessionIndex);
        }

        private static Claim GetClaim(ClaimsPrincipal principal, string claimType)
        {
            return ((ClaimsIdentity)principal.Identity).Claims.Where(c => c.Type ==
claimType).FirstOrDefault();
        }

        private static string GetClaimValue(ClaimsPrincipal principal, string
claimType)
        {
            var claim = GetClaim(principal, claimType);
```

```
            return claim != null ? claim.Value : null;
        }
    }
}
```

UniwaDbContext.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;

namespace ThesisManagement.Data
{
    public class UniwaDbContext : DbContext
    {
        public UniwaDbContext(DbContextOptions<UniwaDbContext> options)
            : base(options)
        {
        }

        public DbSet<ThesisProject> ThesisProjects { get; set; }
        public DbSet<ThesisApplication> ThesisApplications { get; set; }
        public DbSet<Assignment> Assignments { get; set; }
        public DbSet<AssignmentHistory> AssignmentsHistory { get; set; }
        public DbSet<Examination> Examinations { get; set; }
        public DbSet<ExaminationCommitee> ExaminationCommitees { get; set; }
        public DbSet<LookupValue> LookupTable { get; set; }
        public DbSet<Course> Courses { get; set; }
    }
}


UniwaDbContextModelSnapshot.cs
// <auto-generated />
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using ThesisManagement.Data;

namespace ThesisManagement.Migrations
{
    [DbContext(typeof(UniwaDbContext))]
    partial class UniwaDbContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {
#pragma warning disable 612, 618
            modelBuilder
                .HasAnnotation("ProductVersion", "3.1.28")
                .HasAnnotation("Relational:MaxIdentifierLength", 128)
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            modelBuilder.Entity("ThesisManagement.Models.Assignment", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
```

```
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<string>("AssignedStudent")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("AssignedStudentId")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<DateTime>("DateTimeCreated")
                        .HasColumnType("datetime2");

                    b.Property<DateTime>("DateTimeExpires")
                        .HasColumnType("datetime2");

                    b.Property<DateTime>("DateTimeUpdated")
                        .HasColumnType("datetime2");

                    b.Property<string>("SecondAssignedStudent")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("SecondAssignedStudentId")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("Status")
                        .HasColumnType("nvarchar(10)")
                        .HasMaxLength(10);

                    b.Property<int>("ThesisProjectId")
                        .HasColumnType("int");

                    b.HasKey("Id");

                    b.HasIndex("ThesisProjectId");

                    b.ToTable("Assignments");
                });

            modelBuilder.Entity("ThesisManagement.Models.AssignmentHistory", b =>
                {
                    b.Property<long>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("bigint")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<int>("AssignmentId")
                        .HasColumnType("int");

                    b.Property<string>("NewState")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("PreviousState")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("Reason")
                        .HasColumnType("nvarchar(250)")
```

```
                        .HasMaxLength(250);

                    b.Property<DateTime>("TimeStamp")
                        .HasColumnType("datetime2");

                    b.HasKey("Id");

                    b.HasIndex("AssignmentId");

                    b.ToTable("AssignmentsHistory");
                });

            modelBuilder.Entity("ThesisManagement.Models.Course", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<string>("Code")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("Curriculum")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Department")
                        .HasColumnType("nvarchar(100)")
                        .HasMaxLength(100);

                    b.Property<string>("Period")
                        .HasColumnType("nvarchar(10)")
                        .HasMaxLength(10);

                    b.Property<string>("Title")
                        .HasColumnType("nvarchar(100)")
                        .HasMaxLength(100);

                    b.Property<string>("Type")
                        .HasColumnType("nvarchar(25)")
                        .HasMaxLength(25);

                    b.HasKey("Id");

                    b.ToTable("Courses");
                });

            modelBuilder.Entity("ThesisManagement.Models.Examination", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<int>("AssignmentId")
                        .HasColumnType("int");

                    b.Property<string>("Criteria")
                        .HasColumnType("nvarchar(250)")
                        .HasMaxLength(250);
```

```
                    b.Property<string>("CriteriaWeights")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<DateTime>("DateTimeCreated")
                        .HasColumnType("datetime2");

                    b.Property<DateTime>("DateTimeUpdated")
                        .HasColumnType("datetime2");

                    b.Property<string>("Examiner1Comments")
                        .HasColumnType("nvarchar(100)")
                        .HasMaxLength(100);

                    b.Property<string>("Examiner1Scores")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Examiner2Comments")
                        .HasColumnType("nvarchar(100)")
                        .HasMaxLength(100);

                    b.Property<string>("Examiner2Scores")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Examiner3Comments")
                        .HasColumnType("nvarchar(100)")
                        .HasMaxLength(100);

                    b.Property<string>("Examiner3Scores")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<decimal?>("Grade")
                        .HasColumnType("decimal(3,1)");

                    b.Property<DateTime>("PresenationDate")
                        .HasColumnType("datetime2");

                    b.Property<string>("StudentId")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.HasKey("Id");

                    b.HasIndex("AssignmentId");

                    b.ToTable("Examinations");
                });

            modelBuilder.Entity("ThesisManagement.Models.ExaminationCommitee", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<int>("AssignmentId")
                        .HasColumnType("int");
```

```
                    b.Property<string>("Examiner1")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Examiner2")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Examiner3")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.HasKey("Id");

                    b.HasIndex("AssignmentId")
                        .IsUnique();

                    b.ToTable("ExaminationCommitees");
                });

            modelBuilder.Entity("ThesisManagement.Models.LookupValue", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<string>("Key")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Value")
                        .HasColumnType("nvarchar(250)")
                        .HasMaxLength(250);

                    b.HasKey("Id");

                    b.ToTable("LookupTable");
                });

            modelBuilder.Entity("ThesisManagement.Models.ThesisApplication", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<string>("AcademicId")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("CurrentSemester")
                        .HasColumnType("nvarchar(5)")
                        .HasMaxLength(5);

                    b.Property<string>("Date")
                        .HasColumnType("nvarchar(10)")
                        .HasMaxLength(10);

                    b.Property<string>("Department")
                        .HasColumnType("nvarchar(100)")
```

```
                        .HasMaxLength(100);

                    b.Property<string>("Description")
                        .HasColumnType("nvarchar(1000)")
                        .HasMaxLength(1000);

                    b.Property<string>("Email")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("FathersName")
                        .HasColumnType("nvarchar(25)")
                        .HasMaxLength(25);

                    b.Property<string>("FirstName")
                        .HasColumnType("nvarchar(25)")
                        .HasMaxLength(25);

                    b.Property<string>("LastName")
                        .HasColumnType("nvarchar(25)")
                        .HasMaxLength(25);

                    b.Property<string>("Phone")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("ProjectTitleEnglish")
                        .HasColumnType("nvarchar(250)")
                        .HasMaxLength(250);

                    b.Property<string>("ProjectTitleGreek")
                        .HasColumnType("nvarchar(250)")
                        .HasMaxLength(250);

                    b.Property<string>("ProtocolId")
                        .HasColumnType("nvarchar(20)")
                        .HasMaxLength(20);

                    b.Property<string>("ResponsibleProfessor")
                        .HasColumnType("nvarchar(50)")
                        .HasMaxLength(50);

                    b.Property<string>("Specialization")
                        .HasColumnType("nvarchar(100)")
                        .HasMaxLength(100);

                    b.Property<string>("Status")
                        .HasColumnType("nvarchar(25)")
                        .HasMaxLength(25);

                    b.HasKey("Id");

                    b.ToTable("ThesisApplications");
                });

            modelBuilder.Entity("ThesisManagement.Models.ThesisProject", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);
```

```
            b.Property<string>("Bibliography")
                .HasColumnType("nvarchar(1000)")
                .HasMaxLength(1000);

            b.Property<string>("Curriculum")
                .HasColumnType("nvarchar(20)")
                .HasMaxLength(20);

            b.Property<DateTime>("DateTimeCreated")
                .HasColumnType("datetime2");

            b.Property<DateTime>("DateTimeUpdated")
                .HasColumnType("datetime2");

            b.Property<string>("Description")
                .HasColumnType("nvarchar(1000)")
                .HasMaxLength(1000);

            b.Property<string>("DesiredKnowledge")
                .HasColumnType("nvarchar(250)")
                .HasMaxLength(250);

            b.Property<string>("Owner")
                .HasColumnType("nvarchar(50)")
                .HasMaxLength(50);

            b.Property<string>("OwnerId")
                .HasColumnType("nvarchar(20)")
                .HasMaxLength(20);

            b.Property<string>("Requirements")
                .HasColumnType("nvarchar(250)")
                .HasMaxLength(250);

            b.Property<string>("Status")
                .HasColumnType("nvarchar(25)")
                .HasMaxLength(25);

            b.Property<string>("TitleInEnglish")
                .HasColumnType("nvarchar(250)")
                .HasMaxLength(250);

            b.Property<string>("TitleInGreek")
                .HasColumnType("nvarchar(250)")
                .HasMaxLength(250);

            b.Property<string>("Type")
                .HasColumnType("nvarchar(10)")
                .HasMaxLength(10);

            b.HasKey("Id");

            b.ToTable("ThesisProjects");
        });

    modelBuilder.Entity("ThesisManagement.Models.Assignment", b =>
        {
            b.HasOne("ThesisManagement.Models.ThesisProject", "ThesisProject")
                .WithMany("Assignments")
                .HasForeignKey("ThesisProjectId")
                .OnDelete(DeleteBehavior.Cascade)
                .IsRequired();
        });
```

```
            modelBuilder.Entity("ThesisManagement.Models.AssignmentHistory", b =>
                {
                    b.HasOne("ThesisManagement.Models.Assignment", null)
                        .WithMany("StatusChanges")
                        .HasForeignKey("AssignmentId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });

            modelBuilder.Entity("ThesisManagement.Models.Examination", b =>
                {
                    b.HasOne("ThesisManagement.Models.Assignment", null)
                        .WithMany("Examinations")
                        .HasForeignKey("AssignmentId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });

            modelBuilder.Entity("ThesisManagement.Models.ExaminationCommitee", b =>
                {
                    b.HasOne("ThesisManagement.Models.Assignment", null)
                        .WithOne("ExaminationCommitee")
                        .HasForeignKey("ThesisManagement.Models.ExaminationCommitee",
"AssignmentId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });
#pragma warning restore 612, 618
        }
    }
}
```

Κώδικας C# - Repositories

IApplicationsRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public interface IApplicationsRepository
    {
        Task<IEnumerable<ThesisApplication>> GetApplications();
        Task<ThesisApplication> AddApplication(ThesisApplication application);
        Task<ThesisApplication> DeleteApplication(int applicationId);
        Task<ThesisApplication> DeleteApplication(string academicId);
        Task<ThesisApplication> GetApplication(int applicationId);
        Task<ThesisApplication> GetApplication(string academicId);
    }
}
```

ApplicationsRepository.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Data;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public class ApplicationsRepository : IApplicationsRepository
    {
        private readonly UniwaDbContext dbContext;

        public ApplicationsRepository(UniwaDbContext dbContext)
        {
            this.dbContext = dbContext;
        }

        public async Task<ThesisApplication> AddApplication(ThesisApplication
application)
        {
            // Check if there is already a thesis application for this student
            var appInDB = dbContext.ThesisApplications
                .Where(a => a.AcademicId == application.AcademicId && a.Status ==
"Προς Έγκριση")
                .Any();

            if (appInDB)
            {
                return null;
            }

            application.Status = "Προς Έγκριση";
            application.Date = DateTime.Now.ToString("yyyy-MM-dd");

            var result = await dbContext.ThesisApplications.AddAsync(application);
            await dbContext.SaveChangesAsync();
            return result.Entity;
        }

        public async Task<ThesisApplication> DeleteApplication(int applicationId)
        {
            var result = await dbContext.ThesisApplications
                .FirstOrDefaultAsync(p => p.Id == applicationId);

            if (result != null)
            {
                dbContext.ThesisApplications.Remove(result);
                await dbContext.SaveChangesAsync();
                return result;
            }

            return null;
        }

        public async Task<ThesisApplication> DeleteApplication(string academicId)
        {
            var result = await dbContext.ThesisApplications
                .FirstOrDefaultAsync(p => p.AcademicId == academicId);

            if (result != null)
            {
                dbContext.ThesisApplications.Remove(result);
                await dbContext.SaveChangesAsync();
                return result;
```

```
            }

            return null;
        }

        public async Task<ThesisApplication> GetApplication(string academicId)
        {
            return await dbContext.ThesisApplications
                .FirstOrDefaultAsync(a => a.AcademicId == academicId);
        }

        public async Task<ThesisApplication> GetApplication(int applicationId)
        {
            return await dbContext.ThesisApplications
                .FirstOrDefaultAsync(a => a.Id == applicationId);
        }

        public async Task<IEnumerable<ThesisApplication>> GetApplications()
        {
            return await dbContext.ThesisApplications.ToListAsync();
        }

    }
}
```

IAssignmentsRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public interface IAssignmentsRepository
    {
        Task<Assignment> AddAssignment(Assignment assignment);
        Task<AssignmentHistory> ChangeAssignmentStatus(int assignmentId,
AssignmentStatus assignmentStatus);
        Task<Assignment> ExtendCompletionPeriod(int assignmentId);
        Task<Assignment> DeleteAssignment(int assignmentId);
        Task<IEnumerable<Assignment>> GetAssignments();
        Task<Assignment> GetAssignment(int assignmentId);
        Task<Assignment> GetAssignmentByProject(int projectId);
        Task<IEnumerable<Assignment>> GetAssignmentsHistoryAsync(int projectId);
        Task<Assignment> GetStudentAssignment(string studentId);
        Task<IEnumerable<Assignment>> GetProfessorAssignments(string professorId);
        Task<Assignment> SetExaminationCommitee(ExaminationCommitee commitee);
        Task<Assignment> SetExaminationDate(ICollection<Examination> exams);
        Task<Assignment> SetGrade(int assignmentId, ThesisEvaluation grade);
        Task<Assignment> UpdateExaminationCommitee(ExaminationCommitee commitee);
        Task<Assignment> UpdateExaminationDate(int assignmentId, PresentationDate
presentationDate);
    }
}
```

AssignmentsRepository.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Threading.Tasks;
using ThesisManagement.Data;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public class AssignmentsRepository : IAssignmentsRepository
    {
        private readonly UniwaDbContext dbContext;

        public AssignmentsRepository(UniwaDbContext dbContext)
        {
            this.dbContext = dbContext;
        }

        public async Task<Assignment> AddAssignment(Assignment assignment)
        {
            assignment.DateTimeCreated = DateTime.Now;
            assignment.DateTimeUpdated = DateTime.Now;
            assignment.DateTimeExpires =
DateTime.Parse(DateTime.Now.AddMonths(6).ToString("yyyy-MM-dd"));
            assignment.Status = "Active";

            var result = await dbContext.Assignments.AddAsync(assignment);
            await dbContext.SaveChangesAsync();

            if (result != null)
            {
                // Set initial assignment state
                var stateUpdate = new AssignmentHistory()
                {
                    AssignmentId = result.Entity.Id,
                    PreviousState = "Έναρξη",
                    NewState = "Προς Έγκριση",
                    Reason = null,
                    TimeStamp = DateTime.Now
                };
                await dbContext.AssignmentsHistory.AddAsync(stateUpdate);

                // Change project status to 'Not Available'
                var project = dbContext.ThesisProjects.FirstOrDefault(p => p.Id ==
assignment.ThesisProjectId);
                if (project != null)
                {
                    project.Status = "Μη Διαθέσιμο";
                }
            }

            await dbContext.SaveChangesAsync();
            return result.Entity;
        }

        public async Task<AssignmentHistory> ChangeAssignmentStatus(int assignmentId,
AssignmentStatus assignmentStatus)
        {
            var history = await dbContext.AssignmentsHistory
                .Where(p => p.AssignmentId == assignmentId)
                .ToListAsync();

            if (history != null)
            {
                var lastChange = history.OrderByDescending(s =>
s.TimeStamp).FirstOrDefault();
```

```
                var stateUpdate = new AssignmentHistory()
                {
                    AssignmentId = assignmentId,
                    PreviousState = lastChange.NewState,
                    NewState = assignmentStatus.NewStatus,
                    Reason = assignmentStatus.Reason,
                    TimeStamp = DateTime.Now
                };
                var result = await dbContext.AssignmentsHistory.AddAsync(stateUpdate);
                await dbContext.SaveChangesAsync();

                return result.Entity;
            }

            return null;
        }

        public async Task<Assignment> DeleteAssignment(int assignmentId)
        {
            var result = await dbContext.Assignments
                .FirstOrDefaultAsync(p => p.Id == assignmentId);

            if (result != null)
            {
                //dbContext.Assignments.Remove(result);

                result.DateTimeUpdated = DateTime.Now;
                result.Status = "Inactive";

                // Change project status to 'Available'
                var project = dbContext.ThesisProjects.FirstOrDefault(p => p.Id ==
result.ThesisProjectId);
                if (project != null)
                {
                    project.Status = "Διαθέσιμο";
                }

                await dbContext.SaveChangesAsync();
                return result;
            }

            return null;
        }

        public async Task<Assignment> ExtendCompletionPeriod(int assignmentId)
        {
            var result = await dbContext.Assignments
                .FirstOrDefaultAsync(a => a.Id == assignmentId);

            if (result != null)
            {
                result.DateTimeExpires = result.DateTimeExpires.AddMonths(6);
            }

            await dbContext.SaveChangesAsync();

            return result;
        }

        public async Task<Assignment> GetAssignment(int assignmentId)
        {
            var assignment = await dbContext.Assignments
```

```
                .Include(a => a.StatusChanges)
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => a.Id == assignmentId);

            if (assignment != null && assignment.StatusChanges != null)
            {
                assignment.StatusChanges =
assignment.StatusChanges.OrderByDescending(s => s.TimeStamp).ToList();
            }

            return assignment;
        }

        public async Task<Assignment> GetAssignmentByProject(int projectId)
        {
            return await dbContext.Assignments
                .FirstOrDefaultAsync(a => a.ThesisProjectId == projectId &&
a.Status.Equals("Active"));
        }

        public async Task<IEnumerable<Assignment>> GetAssignments()
        {
            return await dbContext.Assignments
                .ToListAsync();
        }

        public async Task<IEnumerable<Assignment>> GetAssignmentsHistoryAsync(int
projectId)
        {
            return await dbContext.Assignments
                .Include(h => h.StatusChanges)
                .Where(a => a.ThesisProjectId == projectId)
                .ToListAsync();
        }

        public async Task<IEnumerable<Assignment>> GetProfessorAssignments(string
professorId)
        {
            return await dbContext.Assignments
                .Include(a => a.ThesisProject)
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .Where(a => a.ThesisProject.OwnerId == professorId)
                .ToListAsync();
        }

        public async Task<Assignment> GetStudentAssignment(string studentId)
        {
            return await dbContext.Assignments
                .Include(a => a.ThesisProject)
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => (a.AssignedStudentId == studentId ||
a.SecondAssignedStudentId == studentId) && a.Status.Equals("Active"));
        }

        public async Task<Assignment> SetExaminationCommitee(ExaminationCommitee
commitee)
        {
            var result = await dbContext.ExaminationCommitees.AddAsync(commitee);
            await dbContext.SaveChangesAsync();
```

```
            return await dbContext.Assignments
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => a.Id == commitee.AssignmentId);
        }

        public async Task<Assignment> SetExaminationDate(ICollection<Examination>
exams)
        {
            foreach (var exam in exams)
            {
                var result = await dbContext.Examinations.AddAsync(exam);
                await dbContext.SaveChangesAsync();
            }

            return await dbContext.Assignments
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => a.Id == exams.First().AssignmentId);
        }

        public async Task<Assignment> SetGrade(int assignmentId, ThesisEvaluation
evaluation)
        {
            var result = await dbContext.Examinations
                .FirstOrDefaultAsync(a => a.AssignmentId == assignmentId &&
a.StudentId == evaluation.StudentId);

            if (result != null)
            {
                result.Grade = evaluation.Grade;
                result.Criteria = evaluation.Criteria;
                result.CriteriaWeights = evaluation.CriteriaWeights;
                result.Examiner1Scores = evaluation.Examiner1Scores;
                result.Examiner2Scores = evaluation.Examiner2Scores;
                result.Examiner3Scores = evaluation.Examiner3Scores;
                result.Examiner1Comments = evaluation.Examiner1Comments;
                result.Examiner2Comments = evaluation.Examiner2Comments;
                result.Examiner3Comments = evaluation.Examiner3Comments;
                result.DateTimeUpdated = DateTime.Now;
            }

            await dbContext.SaveChangesAsync();

            return await dbContext.Assignments
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => a.Id == assignmentId);
        }

        public async Task<Assignment> UpdateExaminationCommitee(ExaminationCommitee
commitee)
        {
            var result = await dbContext.ExaminationCommitees
                .FirstOrDefaultAsync(a => a.AssignmentId == commitee.AssignmentId);

            if (result != null)
            {
                result.Examiner1 = commitee.Examiner1;
                result.Examiner2 = commitee.Examiner2;
                result.Examiner3 = commitee.Examiner3;
            }
```

```
            await dbContext.SaveChangesAsync();

            return await dbContext.Assignments
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => a.Id == commitee.AssignmentId);
        }

        public async Task<Assignment> UpdateExaminationDate(int assignmentId,
PresentationDate presentationDate)
        {
            var exams = await dbContext.Examinations
                .Where(a => a.AssignmentId == assignmentId).ToListAsync();

            if (exams != null)
            {
                foreach (var exam in exams)
                {
                    exam.PresenationDate = presentationDate.Date;
                    exam.DateTimeUpdated = DateTime.Now;
                    await dbContext.SaveChangesAsync();
                }
            }

            return await dbContext.Assignments
                .Include(a => a.Examinations)
                .Include(a => a.ExaminationCommitee)
                .FirstOrDefaultAsync(a => a.Id == assignmentId);
        }
    }
}
```

IProjectsRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public interface IProjectsRepository
    {
        Task<IEnumerable<ThesisProject>> GetProjects();
        Task<IEnumerable<ThesisProject>> GetProfessorProjects(string professorId);
        Task<ThesisProject> GetStudentProject(string studentId);
        Task<ThesisProject> GetProject(int projectId);
        Task<ThesisProject> AddProject(ThesisProject project);
        Task<ThesisProject> UpdateProject(ThesisProject project);
        Task<ThesisProject> DeleteProject(int projectId);
        Task<ThesisProject> GetProjectByTitle(string title);
    }
}
```

ProjectsRepository.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Data;
```

```
using ThesisManagement.Models;
using Z.EntityFramework.Plus;

namespace ThesisManagement.Repositories
{
    public class ProjectsRepository : IProjectsRepository
    {
        private readonly UniwaDbContext dbContext;

        public ProjectsRepository(UniwaDbContext dbContext)
        {
            this.dbContext = dbContext;
        }

        public async Task<ThesisProject> AddProject(ThesisProject project)
        {
            project.DateTimeCreated = DateTime.Now;
            project.DateTimeUpdated = DateTime.Now;

            var result = await dbContext.ThesisProjects.AddAsync(project);
            await dbContext.SaveChangesAsync();
            return result.Entity;
        }

        public async Task<ThesisProject> DeleteProject(int projectId)
        {
            var result = await dbContext.ThesisProjects
                .FirstOrDefaultAsync(p => p.Id == projectId);

            if (result != null)
            {
                dbContext.ThesisProjects.Remove(result);
                await dbContext.SaveChangesAsync();
                return result;
            }

            return null;
        }

        public async Task<ThesisProject> GetStudentProject(string studentId)
        {
            var project = await dbContext.ThesisProjects
                .IncludeFilter(p => p.Assignments
                    .Where(a => (a.AssignedStudentId == studentId ||
a.SecondAssignedStudentId == studentId) && a.Status.Equals("Active"))
                        .SelectMany(a => a.StatusChanges))
                .IncludeFilter(p => p.Assignments
                    .Where(a => (a.AssignedStudentId == studentId ||
a.SecondAssignedStudentId == studentId) && a.Status.Equals("Active"))
                        .SelectMany(a => a.Examinations))
                .IncludeFilter(p => p.Assignments
                    .Where(a => (a.AssignedStudentId == studentId ||
a.SecondAssignedStudentId == studentId) && a.Status.Equals("Active"))
                        .Select(a => a.ExaminationCommitee))
                .Where(p => p.Assignments
                    .Any(a => (a.AssignedStudentId == studentId ||
a.SecondAssignedStudentId == studentId) && a.Status.Equals("Active")))
                .FirstOrDefaultAsync();

            if (project != null && project.Assignments != null)
            {
```

```
                project.Assignments.First().StatusChanges =
project.Assignments.First().StatusChanges.OrderByDescending(s =>
s.TimeStamp).ToList();
            }

            return project;
        }

        public async Task<IEnumerable<ThesisProject>> GetProfessorProjects(string
professorId)
        {
            return await dbContext.ThesisProjects
                .Where(p => p.OwnerId == professorId)
                .OrderByDescending(p => p.DateTimeCreated)
                .ToListAsync();
        }

        public async Task<ThesisProject> GetProject(int projectId)
        {
            var project = await dbContext.ThesisProjects
                .IncludeFilter(p => p.Assignments
                    .Where(a => a.Status.Equals("Active"))
                        .SelectMany(a => a.StatusChanges))
                .IncludeFilter(p => p.Assignments
                    .Where(a => a.Status.Equals("Active"))
                        .SelectMany(a => a.Examinations))
                .IncludeFilter(p => p.Assignments
                    .Where(a => a.Status.Equals("Active"))
                        .Select(a => a.ExaminationCommitee))
                .FirstOrDefaultAsync(p => p.Id == projectId);

            if (project != null && project.Assignments != null &&
project.Assignments.Count > 0)
            {
                project.Assignments.First().StatusChanges =
project.Assignments.First().StatusChanges.OrderByDescending(s =>
s.TimeStamp).ToList();
            }

            return project;
        }

        public async Task<ThesisProject> GetProjectByTitle(string title)
        {
            return await dbContext.ThesisProjects
                .FirstOrDefaultAsync(p => p.TitleInGreek == title || p.TitleInEnglish
== title);
        }

        public async Task<IEnumerable<ThesisProject>> GetProjects()
        {
            return await dbContext.ThesisProjects.OrderByDescending(p =>
p.DateTimeCreated).ToListAsync();
        }

        public async Task<ThesisProject> UpdateProject(ThesisProject project)
        {
            var result = await dbContext.ThesisProjects
                .FirstOrDefaultAsync(p => p.Id == project.Id);

            if (result != null)
            {
                result.TitleInGreek = project.TitleInGreek;
```

```
                result.TitleInEnglish = project.TitleInEnglish;
                result.Description = project.Description;
                result.Requirements = project.Requirements;
                result.DesiredKnowledge = project.DesiredKnowledge;
                result.Bibliography = project.Bibliography;
                result.Curriculum = project.Curriculum;
                result.Type = project.Type;
                result.Status = project.Status;
                result.DateTimeUpdated = DateTime.Now;
                result.OwnerId = project.OwnerId;

                await dbContext.SaveChangesAsync();
                return result;
            }

            return null;
        }
    }
}
```

IValuesRepository.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public interface IValuesRepository
    {
        Task<LookupValue> AddKeyValue(LookupValue value);
        Task<LookupValue> DeleteKeyValue(int id);
        Task<LookupValue> GetValue(int id);
        Task<IEnumerable<Course>> GetCourses();
        Task<IEnumerable<LookupValue>> GetListOfValues(string key);
    }
}
```

ValuesRepository.cs

```
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Data;
using ThesisManagement.Models;

namespace ThesisManagement.Repositories
{
    public class ValuesRepository : IValuesRepository
    {
        private readonly UniwaDbContext dbContext;

        public ValuesRepository(UniwaDbContext dbContext)
        {
            this.dbContext = dbContext;
        }

        public async Task<LookupValue> AddKeyValue(LookupValue keyValue)
        {
```

```
            var check = dbContext.LookupTable.Any(x => x.Key == keyValue.Key.ToLower()
&& x.Value == keyValue.Value.ToUpper());

            if (!check)
            {
                var result = await dbContext.LookupTable.AddAsync(keyValue);
                await dbContext.SaveChangesAsync();

                return result.Entity;
            }

            return null;
        }

        public async Task<LookupValue> DeleteKeyValue(int id)
        {
            var result = await dbContext.LookupTable
                .FirstOrDefaultAsync(p => p.Id == id);

            if (result != null)
            {
                dbContext.LookupTable.Remove(result);
                await dbContext.SaveChangesAsync();
                return result;
            }

            return null;
        }

        public async Task<IEnumerable<Course>> GetCourses()
        {
            return await dbContext.Courses.ToListAsync();
        }

        public async Task<IEnumerable<LookupValue>> GetListOfValues(string key)
        {
            return await dbContext.LookupTable.Where(v => v.Key == key).ToListAsync();
        }

        public async Task<LookupValue> GetValue(int id)
        {
            return await dbContext.LookupTable.FirstOrDefaultAsync(v => v.Id == id);
        }
    }
}
```

Κώδικας C# - Controllers

ApplicationsController.cs

```
using DinkToPdf;
using DinkToPdf.Contracts;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;
using ThesisManagement.Repositories;
using ThesisManagement.Utilities;

namespace ThesisManagement.Controllers
{
    [Route("api/applications")]
    [ApiController]
    public class ApplicationsController : ControllerBase
    {
        private readonly IApplicationsRepository applicationsRepository;
        private readonly IAssignmentsRepository assignmentsRepository;
        private readonly IConverter converter;

        public ApplicationsController(IApplicationsRepository applicationsRepository,
IAssignmentsRepository assignmentsRepository, IConverter converter)
        {
            this.applicationsRepository = applicationsRepository;
            this.assignmentsRepository = assignmentsRepository;
            this.converter = converter;
        }

        // GET: api/applications
        /// <summary>
        /// Retrieves all thesis applications.
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        public async Task<ActionResult> GetApplications()
        {
            try
            {
                return Ok(await applicationsRepository.GetApplications());
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }

        // POST: api/applications
        /// <summary>
        /// Creates a new thesis application.
        /// </summary>
        [HttpPost("thesis")]
        public async Task<ActionResult> AddThesisApplication([FromBody]
ThesisApplication application)
        {
```

```
            try
            {
                if (application == null)
                {
                    return BadRequest();
                }

                var createdApplication = await
applicationsRepository.AddApplication(application);

                if (createdApplication == null)
                {
                    return BadRequest("There is already a pending application for this
Academic ID.");
                }

                return CreatedAtAction(nameof(GetThesisApplication), new { id =
createdApplication.Id }, createdApplication);

            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
creating new thesis application.");
            }
        }

        /// <summary>
        /// Print thesis application.
        /// </summary>
        /// <returns></returns>
        [HttpGet("thesis/print/student/{academicId}", Name =
"PrintThesisApplication")]
        public async Task<IActionResult> PrintApplication(string academicId, string
secondAcademicId)
        {
            try
            {
                ThesisApplication application = new ThesisApplication();
                ThesisApplication secondApplication = new ThesisApplication();

                application = await applicationsRepository.GetApplication(academicId);

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

                if (secondAcademicId != null)
                {
                    secondApplication = await
applicationsRepository.GetApplication(secondAcademicId);

                    if (secondApplication == null)
                    {
                        return NotFound();
                    }
                    // Check if students apply for the same project
                    if (secondApplication.ProjectTitleEnglish !=
application.ProjectTitleEnglish)
                    {
                        return BadRequest();
                    }
```

```
                }

                var globalSettings = new GlobalSettings()
                {
                    ColorMode = ColorMode.Color,
                    Orientation = Orientation.Portrait,
                    PaperSize = PaperKind.A4,
                    Margins = new MarginSettings
                    {
                        Top = 10
                    },
                    DocumentTitle = "Thesis Application"
                    //Out =
@$"C:\Temp\PDFCreator\Thesis_Application_{DateTime.Now.ToString("yyyy_MM_dd_HH_mm_ss")
}.pdf"
                };

                var objectSettings = new ObjectSettings()
                {
                    PagesCount = true,
                    HtmlContent = (secondApplication.AcademicId != null) ?
TemplateGenerator.GetGroupThesisApplicationPDF(application, secondApplication) :
TemplateGenerator.GetThesisApplicationPDF(application),
                    WebSettings = { DefaultEncoding = "utf-8", UserStyleSheet =
Path.Combine(Directory.GetCurrentDirectory(), "assets", "styles.css") },
                    HeaderSettings = { FontName = "Arial", FontSize = 9 },
                    FooterSettings = { FontName = "Arial", FontSize = 9 }
                };

                var pdf = new HtmlToPdfDocument
                {
                    GlobalSettings = globalSettings,
                    Objects = { objectSettings }
                };

                var file = converter.Convert(pdf);

                var base64 = new ThesisApplicationPDF() { Base64 =
Convert.ToBase64String(file) };

                return Ok(base64);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
creating thesis application PDF.");
            }
        }

        /// <summary>
        /// Print thesis examination application.
        /// </summary>
        /// <returns></returns>
        [HttpGet("thesis/print/examination/{academicId}", Name =
"PrintExaminationApplication")]
        public async Task<IActionResult> PrintExaminationApplication(string
academicId, string secondAcademicId)
        {
            try
            {
                ThesisApplication application = new ThesisApplication();
                ThesisApplication secondApplication = new ThesisApplication();
```

```
                application = await applicationsRepository.GetApplication(academicId);

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

                if (secondAcademicId != null)
                {
                    secondApplication = await
applicationsRepository.GetApplication(secondAcademicId);

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

                var globalSettings = new GlobalSettings()
                {
                    ColorMode = ColorMode.Color,
                    Orientation = Orientation.Portrait,
                    PaperSize = PaperKind.A4,
                    Margins = new MarginSettings
                    {
                        Top = 10
                    },
                    DocumentTitle = "Thesis Examination Application"
                    //Out =
@$"C:\Temp\PDFCreator\Thesis_Examination_Application_{DateTime.Now.ToString("yyyy_MM_d
d_HH_mm_ss")}.pdf"
                };

                var objectSettings = new ObjectSettings()
                {
                    PagesCount = true,
                    HtmlContent =
TemplateGenerator.GetExaminationApplicationPDF(application, secondApplication),
                    WebSettings = { DefaultEncoding = "utf-8", UserStyleSheet =
Path.Combine(Directory.GetCurrentDirectory(), "assets", "styles.css") },
                    HeaderSettings = { FontName = "Arial", FontSize = 9 },
                    FooterSettings = { FontName = "Arial", FontSize = 9 }
                };

                var pdf = new HtmlToPdfDocument
                {
                    GlobalSettings = globalSettings,
                    Objects = { objectSettings }
                };

                var file = converter.Convert(pdf);

                var base64 = new ThesisApplicationPDF() { Base64 =
Convert.ToBase64String(file) };

                return Ok(base64);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
creating thesis application PDF.");
            }
        }
```

```
        /// <summary>
        /// Print thesis extension application.
        /// </summary>
        /// <returns></returns>
        [HttpGet("thesis/print/extension/{academicId}", Name =
"PrintExtensionApplication")]
        public async Task<IActionResult> PrintExtensionApplication(string academicId,
string secondAcademicId)
        {
            try
            {
                ThesisApplication application = new ThesisApplication();
                ThesisApplication secondApplication = new ThesisApplication();

                application = await applicationsRepository.GetApplication(academicId);

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

                if (secondAcademicId != null)
                {
                    secondApplication = await
applicationsRepository.GetApplication(secondAcademicId);

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

                var globalSettings = new GlobalSettings()
                {
                    ColorMode = ColorMode.Color,
                    Orientation = Orientation.Portrait,
                    PaperSize = PaperKind.A4,
                    Margins = new MarginSettings
                    {
                        Top = 10
                    },
                    DocumentTitle = "Thesis Extension Application"
                    //Out =
@$"C:\Temp\PDFCreator\Thesis_Extension_Application_{DateTime.Now.ToString("yyyy_MM_dd_
HH_mm_ss")}.pdf"
                };

                var objectSettings = new ObjectSettings()
                {
                    PagesCount = true,
                    HtmlContent =
TemplateGenerator.GetExtensionApplicationPDF(application, secondApplication),
                    WebSettings = { DefaultEncoding = "utf-8", UserStyleSheet =
Path.Combine(Directory.GetCurrentDirectory(), "assets", "styles.css") },
                    HeaderSettings = { FontName = "Arial", FontSize = 9 },
                    FooterSettings = { FontName = "Arial", FontSize = 9 }
                };

                var pdf = new HtmlToPdfDocument
                {
                    GlobalSettings = globalSettings,
                    Objects = { objectSettings }
```

```
                };

                var file = converter.Convert(pdf);

                var base64 = new ThesisApplicationPDF() { Base64 =
Convert.ToBase64String(file) };

                return Ok(base64);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
creating thesis application PDF.");
            }
        }

        /// <summary>
        /// Print thesis evaluation form.
        /// </summary>
        /// <returns></returns>
        [HttpGet("thesis/print/evaluation/{academicId}", Name =
"PrintEvaluationForm")]
        public async Task<IActionResult> PrintEvaluationForm(string academicId)
        {
            try
            {
                var application = await
applicationsRepository.GetApplication(academicId);

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

                var assignment = await
assignmentsRepository.GetStudentAssignment(academicId);

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

                var globalSettings = new GlobalSettings()
                {
                    ColorMode = ColorMode.Color,
                    Orientation = Orientation.Portrait,
                    PaperSize = PaperKind.A4,
                    Margins = new MarginSettings
                    {
                        Top = 10
                    },
                    DocumentTitle = "Thesis Evaluation"
                    //Out =
@$"C:\Temp\PDFCreator\Thesis_Evaluation_{DateTime.Now.ToString("yyyy_MM_dd_HH_mm_ss")}
.pdf"
                };

                var objectSettings = new ObjectSettings()
                {
                    PagesCount = true,
                    HtmlContent = TemplateGenerator.GetEvaluationPDF(application,
assignment),
```

```
                    WebSettings = { DefaultEncoding = "utf-8", UserStyleSheet =
Path.Combine(Directory.GetCurrentDirectory(), "assets", "styles.css") },
                    HeaderSettings = { FontName = "Arial", FontSize = 9 },
                    FooterSettings = { FontName = "Arial", FontSize = 9 }
                };

                var pdf = new HtmlToPdfDocument
                {
                    GlobalSettings = globalSettings,
                    Objects = { objectSettings }
                };

                var file = converter.Convert(pdf);

                var base64 = new ThesisApplicationPDF() { Base64 =
Convert.ToBase64String(file) };

                return Ok(base64);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
creating thesis application PDF.");
            }
        }

        /// <summary>
        /// Retrieves the details of the specified student's thesis application.
        /// </summary>
        [HttpGet("thesis/student/{academicId}", Name = "GetStudentThesisApplication")]
        public async Task<ActionResult<ThesisApplication>>
GetStudentThesisApplication(string academicId)
        {
            try
            {
                var result = await applicationsRepository.GetApplication(academicId);

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

                return result;
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving application from the database.");
            }
        }

        /// <summary>
        /// Deletes a student's thesis application.
        /// </summary>
        /// <returns></returns>
        [HttpDelete("thesis/student/{academicId}")]
        public async Task<ActionResult<ThesisApplication>>
DeleteStudentApplication(string academicId)
        {
            try
            {
                var applicationToDelete = await
applicationsRepository.GetApplication(academicId);
```

```
                if (applicationToDelete == null)
                {
                    return NotFound($"Application of student with Academic ID =
{academicId} not found.");
                }

                return await applicationsRepository.DeleteApplication(academicId);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
deleting thesis application data.");
            }
        }

        // Get: api/applications/thesis/5
        /// <summary>
        /// Retrieves the details of the specified thesis application.
        /// </summary>
        [HttpGet("thesis/{id}", Name = "GetThesisApplication")]
        public async Task<ActionResult<ThesisApplication>> GetThesisApplication(int
id)
        {
            try
            {
                var result = await applicationsRepository.GetApplication(id);

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

                return result;
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving application from the database.");
            }
        }

        // DELETE: api/applications/thesis/5
        /// <summary>
        /// Deletes a thesis application.
        /// </summary>
        /// <returns></returns>
        [HttpDelete("thesis/{id}")]
        public async Task<ActionResult<ThesisApplication>> DeleteApplication(int id)
        {
            try
            {
                var applicationToDelete = await
applicationsRepository.GetApplication(id);

                if (applicationToDelete == null)
                {
                    return NotFound($"Application with ID = { id } not found.");
                }

                return await applicationsRepository.DeleteApplication(id);
            }
            catch (Exception)
```

```
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
deleting thesis application data.");
            }
        }
    }
}
```

AssignmentsController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;
using ThesisManagement.Repositories;

namespace ThesisManagement.Controllers
{
    [Route("api/assignments")]
    [ApiController]
    public class AssignmentsController : ControllerBase
    {
        private readonly IAssignmentsRepository assignmentsRepository;

        public AssignmentsController(IAssignmentsRepository assignmentsRepository)
        {
            this.assignmentsRepository = assignmentsRepository;
        }

        // GET: api/Assignments
        /// <summary>
        /// Retrieves all thesis assignments.
        /// </summary>
        [HttpGet]
        public async Task<ActionResult> GetAssignments()
        {
            try
            {
                return Ok(await assignmentsRepository.GetAssignments());
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }

        // GET: api/Assignments/5
        /// <summary>
        /// Retrieves the specified thesis assignment.
        /// </summary>
        [HttpGet("{id}", Name = "GetAssignment")]
        public async Task<ActionResult<Assignment>> GetAssignment(int id)
        {
            try
            {
                var result = await assignmentsRepository.GetAssignment(id);

                if (result == null)
                {
```

```
                    return NotFound();
                }

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }

        /// <summary>
        /// Retrieves all assignments history for the specified thesis project.
        /// </summary>
        [HttpGet("{projectId}/history")]
        public async Task<ActionResult<IEnumerable<Assignment>>>
GetAssignmentsHistory(int projectId)
        {
            try
            {
                return Ok(await
assignmentsRepository.GetAssignmentsHistoryAsync(projectId));
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }
        [HttpPost]
        public async Task<ActionResult> AssignProject([FromBody] Assignment
assignment)
        {
            try
            {
                if (assignment == null)
                {
                    return BadRequest();
                }

                var existingAssignment = await
assignmentsRepository.GetAssignmentByProject(assignment.ThesisProjectId);

                if (existingAssignment != null)
                {
                    ModelState.AddModelError("error", "This project is already
assigned.");
                    return BadRequest(ModelState);
                }

                var studentAssignment = await
assignmentsRepository.GetStudentAssignment(assignment.AssignedStudentId);

                if (studentAssignment != null)
                {
                    ModelState.AddModelError("error", $"The student with Academic ID
'{assignment.AssignedStudentId}' has already a project assigned to them.");
                    return BadRequest(ModelState);
                }

                studentAssignment = await
assignmentsRepository.GetStudentAssignment(assignment.SecondAssignedStudentId);
```

```
                if (studentAssignment != null)
                {
                    ModelState.AddModelError("error", $"The student with Academic ID
'{assignment.SecondAssignedStudentId}' has already a project assigned to them.");
                    return BadRequest(ModelState);
                }

                var newAssignment = await
assignmentsRepository.AddAssignment(assignment);

                return CreatedAtAction(nameof(GetAssignment), new { id =
newAssignment.Id }, newAssignment);

            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
assigning thesis project to student.");
            }
        }

        /// <summary>
        /// Changes the status of a thesis assignment.
        /// </summary>
        [HttpPost("{id}/change-status")]
        public async Task<ActionResult> ChangeStatus(int id, [FromBody]
AssignmentStatus assignmentStatus)
        {
            try
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }

                var statusChange = await
assignmentsRepository.ChangeAssignmentStatus(id, assignmentStatus);

                if (statusChange == null)
                {
                    throw new Exception();
                }

                return Ok(statusChange);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
changing status to thesis assignment.");
            }
        }

        /// <summary>
        /// Extends a thesis assignment completion period by a semester.
        /// </summary>
        [HttpPost("{id}/extend")]
        public async Task<ActionResult> Extend(int id)
        {
            try
```

```
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }

                var assignment = await
assignmentsRepository.ExtendCompletionPeriod(id);

                if (assignment == null)
                {
                    throw new Exception();
                }

                return Ok(assignment);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
extending thesis completion period.");
            }
        }

        /// <summary>
        /// Sets the presantation date of the specified thesis assignment.
        /// </summary>
        [HttpPost("{id}/presentation")]
        public async Task<ActionResult> SetPresentationDate(int id, [FromBody]
PresentationDate presentationDate)
        {
            try
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }
                if (existingAssignment.Examinations.Count > 0)
                {
                    return BadRequest("Examination date is already defined for this
thesis assignment.");
                }

                var project = assignmentsRepository

.GetStudentAssignment(existingAssignment.AssignedStudentId).Result.ThesisProject;
                ICollection<Examination> exams = new List<Examination>();
                if (project.Type.Equals("ATOMIKH"))
                {
                    Examination exam = new Examination()
                    {
                        AssignmentId = id,
                        StudentId = existingAssignment.AssignedStudentId,
                        DateTimeCreated = DateTime.Now,
                        DateTimeUpdated = DateTime.Now,
                        PresenationDate = presentationDate.Date
                    };
                    exams.Add(exam);
```

```
                }
                else
                {
                    Examination exam = new Examination()
                    {
                        AssignmentId = id,
                        StudentId = existingAssignment.AssignedStudentId,
                        DateTimeCreated = DateTime.Now,
                        DateTimeUpdated = DateTime.Now,
                        PresenationDate = presentationDate.Date
                    };
                    exams.Add(exam);

                    exam = new Examination()
                    {
                        AssignmentId = id,
                        StudentId = existingAssignment.SecondAssignedStudentId,
                        DateTimeCreated = DateTime.Now,
                        DateTimeUpdated = DateTime.Now,
                        PresenationDate = presentationDate.Date
                    };
                    exams.Add(exam);
                }
                var result = await assignmentsRepository.SetExaminationDate(exams);

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
setting presentation date.");
            }
        }

        /// <summary>
        /// Sets the examination committee of the specified thesis assignment.
        /// </summary>
        [HttpPost("{id}/examination-committee")]
        public async Task<ActionResult> SetExaminationCommittee(int id, [FromBody]
ExaminationCommiteeDto examiners)
        {
            try
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }
                if (existingAssignment.ExaminationCommitee != null)
                {
                    return BadRequest("Examination commitee is already defined for
this thesis assignment.");
                }

                var commitee = new ExaminationCommitee()
                {
                    AssignmentId = id,
                    Examiner1 = examiners.Examiner1,
                    Examiner2 = examiners.Examiner2,
                    Examiner3 = examiners.Examiner3
                };
```

```
                var result = await
assignmentsRepository.SetExaminationCommitee(commitee);

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
setting examination commitee.");
            }
        }

        /// <summary>
        /// Updates the presantation date of the specified thesis assignment.
        /// </summary>
        [HttpPut("{id}/presentation")]
        public async Task<ActionResult> UpdatePresentationDate(int id, [FromBody]
PresentationDate presentationDate)
        {
            try
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }

                var result = await assignmentsRepository.UpdateExaminationDate(id,
presentationDate);

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
updating presentation date.");
            }
        }

        /// <summary>
        /// Evaluates the specified thesis assignment.
        /// </summary>
        [HttpPut("{id}/evaluation")]
        public async Task<ActionResult> SetThesisGrade (int id, [FromBody]
ThesisEvaluation evaluation)
        {
            try
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }

                var result = await assignmentsRepository.SetGrade(id, evaluation);

                return Ok(result);
            }
```

```
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
updating thesis grade.");
            }
        }

        /// <summary>
        /// Updates the examination committee of the specified thesis assignment.
        /// </summary>
        [HttpPut("{id}/examination-committee")]
        public async Task<ActionResult> UpdateExaminationCommittee(int id, [FromBody]
ExaminationCommiteeDto examiners)
        {
            try
            {
                var existingAssignment = await
assignmentsRepository.GetAssignment(id);

                if (existingAssignment == null)
                {
                    return NotFound("Thesis assignment not found.");
                }

                var commitee = new ExaminationCommitee()
                {
                    AssignmentId = id,
                    Examiner1 = examiners.Examiner1,
                    Examiner2 = examiners.Examiner2,
                    Examiner3 = examiners.Examiner3
                };

                var result = await
assignmentsRepository.UpdateExaminationCommitee(commitee);

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
updating examination commitee.");
            }
        }

        // DELETE: api/ApiWithActions/5
        /// <summary>
        /// Cancels the specified thesis assignment.
        /// </summary>
        [HttpDelete("{id}")]
        public async Task<ActionResult<Assignment>> Delete(int id)
        {
            try
            {
                var assignmentToDelete = await
assignmentsRepository.GetAssignment(id);

                if (assignmentToDelete == null)
                {
                    return NotFound($"Assignment with ID = { id } not found.");
                }

                return await assignmentsRepository.DeleteAssignment(id);
            }
```

```
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
deleting thesis project data.");
            }
        }
    }
}
```

AuthController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Authentication;
using System.Security.Claims;
using System.Threading.Tasks;
using ITfoxtec.Identity.Saml2;
using ITfoxtec.Identity.Saml2.MvcCore;
using ITfoxtec.Identity.Saml2.Schemas;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.Options;
using ThesisManagement.Identity;
using ThesisManagement.Models;

namespace ThesisManagement.Controllers
{
    [AllowAnonymous]
    [Route("Auth")]
    public class AuthController : Controller
    {
        const string relayStateReturnUrl = "ReturnUrl";
        private readonly Saml2Configuration config;

        public AuthController(IOptions<Saml2Configuration> configAccessor)
        {
            config = configAccessor.Value;
        }

        [HttpGet]
        [Route("Login")]
        public IActionResult Login(string returnUrl = null)
        {
            var binding = new Saml2RedirectBinding();
            binding.SetRelayStateQuery(new Dictionary<string, string> { {
relayStateReturnUrl, returnUrl ?? Url.Content("~/") } });

            return binding.Bind(new Saml2AuthnRequest(config)).ToActionResult();
        }

        [HttpPost]
        [Route("AssertionConsumerService")]
        public async Task<IActionResult> AssertionConsumerService()
        {
            var binding = new Saml2PostBinding();
            var saml2AuthnResponse = new Saml2AuthnResponse(config);

            binding.ReadSamlResponse(Request.ToGenericHttpRequest(),
saml2AuthnResponse);
            if (saml2AuthnResponse.Status != Saml2StatusCodes.Success)
            {
```

```
                throw new AuthenticationException($"SAML Response status:
{saml2AuthnResponse.Status}");
            }
            binding.Unbind(Request.ToGenericHttpRequest(), saml2AuthnResponse);
            await saml2AuthnResponse.CreateSession(HttpContext, claimsTransform:
(claimsPrincipal) => ClaimsTransform.Transform(claimsPrincipal), lifetime: new
TimeSpan(1, 0, 0));

            var relayStateQuery = binding.GetRelayStateQuery();
            var returnUrl = relayStateQuery.ContainsKey(relayStateReturnUrl) ?
relayStateQuery[relayStateReturnUrl] : Url.Content("~/");
            return Redirect(returnUrl);
        }

        [HttpPost("Logout")]
        public async Task<IActionResult> Logout()
        {
            if (!User.Identity.IsAuthenticated)
            {
                return Unauthorized();
            }

            var binding = new Saml2PostBinding();
            var saml2LogoutRequest = await new Saml2LogoutRequest(config,
User).DeleteSession(HttpContext);

            return binding.Bind(saml2LogoutRequest).ToActionResult();
        }

        [HttpGet]
        [Route("LoggedOut")]
        public IActionResult LoggedOut()
        {
            var binding = new Saml2PostBinding();
            binding.Unbind(Request.ToGenericHttpRequest(), new
Saml2LogoutResponse(config));

            return Redirect(Url.Content("~/"));
        }

        [HttpGet]
        [Route("SingleLogout")]
        public async Task<IActionResult> SingleLogout()
        {
            Saml2StatusCodes status;
            var requestBinding = new Saml2PostBinding();
            var logoutRequest = new Saml2LogoutRequest(config, User);
            try
            {
                requestBinding.Unbind(Request.ToGenericHttpRequest(), logoutRequest);
                status = Saml2StatusCodes.Success;
                await logoutRequest.DeleteSession(HttpContext);
            }
            catch (Exception exc)
            {
                // log exception
                Console.WriteLine("SingleLogout error: " + exc.ToString());
                status = Saml2StatusCodes.RequestDenied;
            }

            var responsebinding = new Saml2PostBinding();
            responsebinding.RelayState = requestBinding.RelayState;
            var saml2LogoutResponse = new Saml2LogoutResponse(config)
```

```
            {
                InResponseToAsString = logoutRequest.IdAsString,
                Status = status,
            };
            return responsebinding.Bind(saml2LogoutResponse).ToActionResult();
        }

        [Authorize]
        [HttpGet("claims")]
        public IActionResult GetClaims()
        {
            if (!User.Identity.IsAuthenticated)
            {
                return Unauthorized();
            }


            IEnumerable<Claim> claims = User.Claims;

            UserClaims userClaims = new UserClaims()
            {
                Id = claims.FirstOrDefault(c => c.Type.Contains("saml2nameid")).Value,
                Email = claims.FirstOrDefault(c => c.Type ==
ClaimTypes.NameIdentifier).Value,
                FirstName = claims.FirstOrDefault(c =>
c.Type.Equals("FirstName")).Value,
                LastName = claims.FirstOrDefault(c =>
c.Type.Equals("LastName")).Value,
                AcademicId = claims.FirstOrDefault(c =>
c.Type.Equals("AcademicId")).Value,
                AcademicRole = claims.FirstOrDefault(c =>
c.Type.Equals("AcademicRole")).Value
                //Department = User.Claims.FirstOrDefault(c =>
c.Type.Equals("Department")).Value,
            };

            return Ok(userClaims);
        }
    }
}
```

ProjectsController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;
using ThesisManagement.Repositories;

namespace ThesisManagement.Controllers
{
    [Route("api/projects")]
    [ApiController]
    public class ProjectsController : ControllerBase
    {
        private readonly IProjectsRepository projectsRepository;

        public ProjectsController(IProjectsRepository projectsRepository)
        {
            this.projectsRepository = projectsRepository;
```

```
        }

        // GET: api/Projects
        /// <summary>
        /// Retrieves all thesis projects.
        /// </summary>
        /// <returns></returns>
        [HttpGet]
        public async Task<ActionResult> GetProjects()
        {
            try
            {
                return Ok(await projectsRepository.GetProjects());
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }

        // GET: api/Projects/5
        /// <summary>
        /// Retrieves the details of the specified thesis project.
        /// </summary>
        /// <returns></returns>
        [HttpGet("{id}", Name = "GetProject")]
        public async Task<ActionResult<ThesisProject>> GetProject(int id)
        {
            try
            {
                var result = await projectsRepository.GetProject(id);

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

                return result;
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
            }

        /// <summary>
        /// Retrieves the specified professor's thesis projects.
        /// </summary>
        /// <returns></returns>
        [HttpGet("professor/{id}")]
        public async Task<ActionResult<IEnumerable<ThesisProject>>>
GetProfessorProjects(string id)
        {
            try
            {
                var result = await projectsRepository.GetProfessorProjects(id);

                if (result == null)
                {
                    return NotFound();
                }
```

```
                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }

        /// <summary>
        /// Retrieves the specified student's thesis project.
        /// </summary>
        /// <returns></returns>
        [HttpGet("student/{id}")]
        public async Task<ActionResult<ThesisProject>> GetStudentProject(string id)
        {
            try
            {
                var result = await projectsRepository.GetStudentProject(id);

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

                return Ok(result);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
retrieving data from the database.");
            }
        }

        // POST: api/Projects
        /// <summary>
        /// Creates a new thesis project.
        /// </summary>
        /// <returns></returns>
        [HttpPost]
        public async Task<ActionResult> CreateProject(ThesisProject project)
        {
            try
            {
                if (project == null)
                {
                    return BadRequest();
                }

                var existingProject = await
projectsRepository.GetProjectByTitle(project.TitleInGreek);

                if (existingProject != null)
                {
                    ModelState.AddModelError("title", "Project title already
exists.");
                    return BadRequest(ModelState);
                }

                var createdProject = await projectsRepository.AddProject(project);
```

```
                return CreatedAtAction(nameof(GetProject), new { id =
createdProject.Id }, createdProject);

            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
creating new thesis project.");
            }
        }

        // PUT: api/Projects/5
        /// <summary>
        /// Updates the details of the specified thesis project.
        /// </summary>
        /// <returns></returns>
        [HttpPut("{id}")]
        public async Task<ActionResult<ThesisProject>> UpdateProject(int id,
ThesisProject project)
        {
            try
            {
                if (id != project.Id)
                {
                    return BadRequest("Project ID mismatch.");
                }

                var projectToUpdate = await projectsRepository.GetProject(project.Id);

                if (projectToUpdate == null)
                {
                    return NotFound($"Project with ID = { project.Id } not found.");
                }

                return await projectsRepository.UpdateProject(project);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
updating thesis project data.");
            }
        }

        // DELETE: api/Projects/5
        /// <summary>
        /// Deletes a thesis project.
        /// </summary>
        /// <returns></returns>
        [HttpDelete("{id}")]
        public async Task<ActionResult<ThesisProject>> DeleteProject(int id)
        {
            try
            {
                var projectToDelete = await projectsRepository.GetProject(id);

                if (projectToDelete == null)
                {
                    return NotFound($"Project with ID = { id } not found.");
                }

                return await projectsRepository.DeleteProject(id);
            }
            catch (Exception)
```

```
            {
                return StatusCode(StatusCodes.Status500InternalServerError, "Error
deleting thesis project data.");
            }
        }
    }
}
```

ValuesController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ThesisManagement.Models;
using ThesisManagement.Repositories;

namespace ThesisManagement.Controllers
{
    [Authorize]
    [Route("api/values")]
    [ApiController]
    public class ValuesController : ControllerBase
    {
        private readonly IValuesRepository valuesRepository;

        public ValuesController(IValuesRepository valuesRepository)
        {
            this.valuesRepository = valuesRepository;
        }

        // GET: api/lovs/courses
        /// <summary>
        /// Retrieves a list of courses.
        /// </summary>
        /// <returns></returns>
        [HttpGet("courses")]
        public async Task<IEnumerable<Course>> GetCourses()
        {
            return await valuesRepository.GetCourses();
        }

        /// <summary>
        /// Retrieves a list of values for the lookup value specified.
        /// </summary>
        /// <returns></returns>
        [HttpGet("{key}")]
        public async Task<IEnumerable<LookupValue>> GetListOfValues(string key)
        {
            return await valuesRepository.GetListOfValues(key);
        }

        /// <summary>
        /// Adds a key-value to the lookup table.
        /// </summary>
        /// <returns></returns>
        [HttpPost]
        public async Task<IActionResult> AddKeyValues(LookupValue value)
        {
            var result = await valuesRepository.AddKeyValue(value);
```

```
            try
            {
                if (result == null)
                {
                    return BadRequest("Key-Value already exists!");
                }
                else
                {
                    return StatusCode(StatusCodes.Status201Created, result);
                }
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError);
            }
        }

        /// <summary>
        /// Deletes a key-value from the lookup table.
        /// </summary>
        /// <returns></returns>
        [HttpDelete]
        public async Task<ActionResult<LookupValue>> DeleteKeyValue(int id)
        {
            try
            {
                var keyValueToDelete = await valuesRepository.GetValue(id);

                if (keyValueToDelete == null)
                {
                    return NotFound($"Lookup value with ID = {id} not found.");
                }

                return await valuesRepository.DeleteKeyValue(id);
            }
            catch (Exception)
            {
                return StatusCode(StatusCodes.Status500InternalServerError);
            }
        }
    }
}
```

Κώδικας C# - Utilities

TemplateGenerator.cs

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ThesisManagement.Models;

namespace ThesisManagement.Utilities
{
    public static class TemplateGenerator
    {
        private static string imgLogo =
"
HBwYIDAoMDAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8XGBYUGBIUFRT/2wBDAQMEBAUEBQkFBQkUDQsNFBQUF
BQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBT/wAARCAB7Aj0DASIAAhEBAxE
B/8QAHwAAAQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhM
UEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RFRkdISUpTVFVWV1hZWmN
kZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW1
9jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAECAwQFBgcICQoL/8QAtREAAgE
CBAQDBAcFBAQAAQJ3AAECAxEEBSExBhJBUQdhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYkNOEl8RcYGRomJygpK
jU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6goOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKm
qsrO0tba3uLm6wsPExcbHyMnK0tPU1dbX2Nna4uPk5ebn6Onq8vP09fb3+Pn6/9oADAMBAAIRAxEAPwD9UqKKO
1AAKKKa0iqcGgBxximgj1rD8VeNNG8FaX/aWu6jDpmn+akP2mc4QM7BVBPbJI5PFeW698dptQ1Pxx4W8P2/2bx
To1v9o0x7oB4NT2xJK6x4/i2uox/tqemaajKWxLklue3MwA5OBSbhjqK+PvE3xA0nxTe67rWv+J9f0O6uLCG58
JWmmzzQxzK9urgokfyyy+buVlbOMYwM19I+BNT1l/hto19rtvJ/b39nRS3lui4bzvLBdcdMk56cVcocpEZ3Onj
1W0luLi3juoXnt8edGsgLRZGRuHUZHPNTRzJKqsjq6sMgqcgivlT4K6N4p0v4px+IdT8Na3BdeK1u/wDhImuoC
tvasH3WgUk8qqAx4HrmrGh+GbzwL8NPiJ4h1S/1WK10uTULDQLIytCbe2WZvK2YxlnchQxzhQoHfLcEtmCmz6l
8xc8mngg9K+bZvizqXwU0pLPxHq0uuponh46trd3fsomNzKQlvbIygAFnWTqCcAGvXPAHxR0nx3o/261MsMSrH
5kssbLBvccrHKwCyYPGV4qXCS16DU03Y7eimLKCvf8AKnioNAooxRigAoo7UUAHeiiigAoooxQAUCijFABRRji
igAooooAKKKKACijFGKACijFFABRQKMUAHagUAUd6ACig0UAFFGKKACjvRRQAUUYooAOlFGKMUAHWijFFAAKKM
UYoAKKKKACiijFAB1ooxRQAUUYooAKKKMUAFFFFABRRRQAnpS0UYoAO9HejFGKACijFAoAKO1FIxwKAGu+0+te
d+LvipaW2v3/hTw9Nbaj44jsGvodNlciNQMbRIw+6W3DH1BPFW/it4uuNB0ddO0nUtM07xNqqvBpL6xI8du02B
1ZVPIzkL3NeLadGvjfWF8DeI7OfQ/idobRXMHiLSW89tpiGLp5iibkfBjZGHOMY440jG+rMpytojm7698S/HOP
xK1nod6/g/wAQ40uexv7hZp9L1WEHE3k8+XCHRFYE5PDbQDz7D8Pf2dtH8LPJdXBnaWee11GK0MpZdOuI4tjJC
/3vLx8u08beOlenaF4ZstFNzPDbQR316yy3tzDEIzcyhAu9seyitgRgU5VH8MdhRp21bMvSPDenaHYQWNhZw2t
nASYoIkwqZYscfUkmr/kY78d+KmCijFZamtkcV8T/AIjWPwp8JXniLUbeafF7ZstfFTwj4bvobKLUvEulWV7NIsUdtNdoJGZjhRtznk0OzfuijeC95nXUVFFN5mPpmpag1CiiigAFHWk
/Ggk0AL2opBzRk0CFo7U0E07tQMKKTNLnNK6AKO9Jk0tFwCik5pRTAKKQk0AmgQtFJk0ZNK4xaKaSaATRcB3ai
ikyaYC0UmTSk0AFApMmgGgBaDRRQAUUhJ7UopXAKKKQmgBaKTk0ZNFwFopOaMmi4C0GkyfWjJHemAtFJzQc5pX
AWjrRmimAUUGigAopMnNKTSuAUUgNB9qLoBaKTmjJpiF7UUgzS0DCikPWjJpXAWjOKTJxSbqYDqhu5VhhZ2cIq
8ksQBUhwFJryv9om7t3+H8+mXF5qWmLeug+3afpr3wh2uHxJGgJ2Nt2npwTyKaV2kTLRHkU0mq+P9Z1vTdd1oa
X4l1C6l2eCvFMG7TprZTiEW7jkvtAYyRNnc33eBX0J8O/BUPg7QLW3PmTXixLHJPcy+fMEBYrF5pAZ1TcQpbnH
1NeM/s+3PiPxvrU02qa74f8W6Dpk884v7B2kka6kcFEMUgzbeUoYBFOOevWvpOMYQA1rUfRGdNdWKBhahmmSKJ
5HkVI0BZmY4AA6kmpXAKkH0715F8avEOq6v4K8feHNA0+4XUrfRZH+3XcZSzffGwaNJAR+8Ckn0Bxmsopykka
ajFs7vQPHmjeJNf1rR9NvFvL3R2jS9EfKxO4JClum7AyR2yK32fn39Aa+Pf8AgnLN2MqSpV
vd6nuYKq6lPXoMuraO7hlhmjWWKRSjowyGBGCK+XfEHwq4/Cf4hoR67qCW+l+ElmdzqpP2p2PzPEp+aUR1Z5
YbH99H0RY1SQENtBYvgbif4lOOK+/cxA58ffjZ8V/12QPN
LtIUZG47jjFfVPauc8Y2l+2i3Nxpb6dbarBGzW13qURkig/vMcEHoOx5xXnQk4nbKCluc98IfHus+NLDUj4g0C
58OajbXTiK3ukCNNbN80MmMnB2nDDPDA16Er7q+a/hnq1hefFi21eLxnrPjjVbm3axu5bDTNukwpkOuHVQqhWD
```

c7mb56+kYxx1qai5WKDvoS0jfdNKOlIx+WpNDyH45/Ha4+D194YsbDwpqHi3Utelnit7LTpAsgMaqx6g54b9K8
+1T9rbxromnXOoX/wAB/FtrZW0bSzTySoFjRRkk/L2xXQ/HP/k4D4F/9f2pf+k6V6B8d/8Aki3jf/sEXP8A6La
tvd5Y3W5zvmbdnsdL4N8RJ4s8KaTraRNbx6hax3SxO2SgdQwGe+M1y/hj4w6L4s+KHizRZyg6n4djt5JzuGH8
wEkD/d+UH3NZvhLxZZ+Bf2c9G8QXziO003w7DdSMT2WAHH4185+C/Dt38Hp/hn8V9UDxXvivUJofErvxhb8h7c
v/wBcysa+gBPrTjTjJSv8hSqONkj6h+L3xZ0v4N+HLPXdb3rpkl/BZTzL/wAsRKSokI7gHGfaux07UYNStIbq1
nS5t50EkckTblSMgg9wQa8G/bYsrbV/hRo9ndRLPa3HibS4Z3GVZGnAIPsRmsnwHq17+zL45tvh/rs8lx4B1
iQ/8ACM6tcNu+ySk5NlKx/wDHCfp9EoKUFbcvmkpuL2PXB8Vbd/jRJ8pFsp5PtS6MdZN5vGwr5qx+XtxnOWzn2r
yab9rrxDeeI/EemeHfhF4k8i8T2T2+ialPpkt/p8yGN5N5v/ADD+ND4Gb/0sirzD4OfGHWvwn/wC
KPizYaf8ADfxLqvgfxlqMxvqfxlqMxvNHiV4kPmEbDkj5u9aqnFu1BVFZ2TdjQ0/59+QVGnOP7vaHb/1HwUEGzEnSAfjEdACe
aFdyBbhdrlQC2MfMACOO9edW37Vnjq+Mi+WCZBJHIsqEMp6nL4asTYQkQa6H7v41fhMf+39AMBC4FpNZvOwR2O3zQdwk5DkJzvpGBtdjl99WoHJ0+9fHOw/sqJt79+fLogFlTyrOnpAnLRZNhtyrhjpfu7NrHrBgdoZzYZQnxxY0qwl0B
aFdyBbhdrlQC2MfMACOO9edW37Vnjy+tYri2+Avi+WCZBJHIsqEMp6H7v41ufH/AMR3Pi39kzxfqt5o174fuLn
S5GfTtRULPB82AGA4zwK9f8A/8iP4f/7B9v8A+ilrP3YxvYp80IpwWTE8u7Mlt3gvbuybS1llBMjwSMn3gOxAxX04zXCfsvD/TvjJ/2Ot3/6Igo/Yp/
kV5VkREc52jj74FaPwu+Kw+Kvgi+1/TvKtg+Kvgi+1/TtMlt3gvbuySlllBMjwSMn3gOxAxX04zXCfsvD/TvjJ/2Ot3/6Igo/Yp/
5I9qH/Yx6r/6VPRKMYxk1xhuovqcloX7Z/ijxO1+mk+Q1hxbuovqcloX7Z/ijxO1+mk+Bt5kN5UqfeVvlGDzXqWjfGHxNqYx
1fxNP8ADPXbHWLKXD4cmYfabofL8yHGMcnt/Ca4fVUb4U/tY6P/YwFzY+P7eQanpcP3oJ4fI5yoFI+Vj3we4
FfR56UqnKkmFPmd22FLVl+2T4rv/ABFfa8wUaS3NokyFPSH+4WG3Azg17L42+vBxNnM3iL+K6+B4fBxvNMnM3iL
UbfThDuAa2eUZy2euOn4xVxPgDn9rX4o/wDYI0tX4q/tL63wtqcOnX0Gry2nimTFzPQzD0czK9TVumZ
yUEuguZxjds+g9SvRpum3V2QXEETS7R1bAzj9lMPkm3V2QXEETS7R1bAzjQiplePLhIX5/cYrBw9ee
+40S/hPwW8cQq9vIpke2TC5Q8nmr3wc1o+H/wBh/T9TB2vbeHLm7Ab0VfuyRg8eKzn2TUVfuv7ROVkeyfC74haf
fA+l+JdMb/Rb6MtsbBkhuO/MGdKu5B5B9q8Q8ZXzNsgbByImctBtUf8Cr5x/Yma8+Gjy8+Gjy/8D3V
Z3ZdR0m08UaYZD/AAzxr9pjH+5nBM4iRaPw54B8R6bpA2n5JZ5Xzcn/gP7tPruFaKgnV/
5b6Euq+VNLU+qPiv8UYPhJaxwQdXabEWMmtNMMZzDqnvLJEgzWnPsDbGeEp0fMsvLxOJyksYgtQ+WqeK2vvGY
GSSa8S/bDH/ABQPHmt4s0z/wBGGmftfqRrfhR6VsrWcXCu7otD/wCs9zfM0P0hO+pU7pX/6FqqEbLQo4u
xf2uNX8RA6h4M+FHifxvkxV4ZVVj/AMThdtusyjjQ0KMCXHHGSv5V6p8N9pt7GYrmyl7p
Ih6HryMg44NdlZ2UGn2sVtVawpDBEoRIk0jXGEoRIkXCqBwAB2rwPxhpkfw9/a0CalUXoIi8QJI3ACl8Gc
z6fWklGV0zi70Zh70Zh8nbs+iVO4Up702nrWJuef/2Fe8o/Hft8d6bjiAl9LNl/3jkuauMEk23J42Kc7uzTc+Ji8k4F
jHlPmJHHNfL/wAWvB8H7SHxq1TwMS+i+EtDld3Xoup1XSERH03Ig3D3b2r1H9mzxvP44+4jN6kgWA+Bh5CK1JCe
VuID5b5787Q3OYGtZQSgmtzGMm5n8Ng8Kfi/Z/E0+ILX7HlpGr6Fq2IW+44I6qw5BeEpgBhtheeTdU9+jNhnBwt
KniLwdp2nz3Eeg28cl7qgkHkpO4ysAHUtt+Y+grxP8AaJj1xVXrRtJ1ixt9+7Ck2kuPdh
sJ6/nXq/7N3w5vfAXw0mPDwa9t8hFdEAyAXw5hklkiXxRrUz6rq856856tcScVz6KCF/CrlBKPpmk3nwT8UQX2p
ySRWMLzorXRQZbYNvOFwa9t8HfELUvEHw9m8R6Xv4x/AAveRRzSPpN8wMyhAT1AA5xxfx0/50F+Av/X9X/X9qf/
Olew+OMf8Ibr3/Xhcf8AM5tCt/j5aT/s4+lnE/4sy3v3/zp3bx1FvcyXm69YGF2QsUWxbtbMydezAEMdOUwA+J
F9t441dpLeyuLSG4itV0a5UDJCgH3nJYDS/4Jwe8GbIjE5V2OfLtZu/4ghRjjTQW5VvsX2vqZvTm11+0Ak5yjH4
aY5aGdYlBup17u2Pk7AdK09kuVz6Jk+0xXxXt8rW28JmMRU6XMSNSTe11g0YCe7JsR7+0z+VVrYxd6DrEV2IlXCqBEgRgKRjjtXBc
HtXGv6Fp2ppG0SXltHcKcKhOodO8gQjJETxVfddycs/ePG8uUtwgmbwHuBbXwQ6KdJH7+cR/ZXAH0e15Fx66Lkv5qtjV0n1T5+fqqs8HHqPx6D0UKi1BHWVgvH6ukhsYIxPDbIUkAjU
Bl56HtWMYc8bpGjGzfE+fEwEdyuU6uz16V1ov+dDrW15SSh5tt/h+/DkahgsrY16sSMq38h8h7dlZcGtE07pyzD5LJLZ8x0V6GTsxm6Yk6Ow4rY7SaGCuC3XDZsmpGDb0o0FkMEZEQ3i0Q
k0u6ezu/s1wjeVKvVT8vHGD+NaX7IPiGbxZrnxb140q70Sa68QuQO8QtBrC/ULNCfIQYYDoak8UBvhT+1J4bvdJUT2
/j5Hs9V0uL76ywrlLzHQAL8rHvxWijGMnB0HrVENykJM9U+E3xB1l4bHrp6RxJUFhtQE+oaPxXEMNFFEM/lLz6owLyLgHeMADH
OPwrv6jToAOlSCua99TdKxHMcP8a4DQfi/o3iD4eVqHiiKz8gJGNS1xAbTXLkbXX5bbuQzVbrFGrTOURqbxD0FdfyqxQxZXAPNmEDYqA5l5Y
rUHEdlp9vJczE/wB1FJP48+BVZ8w3iL6GkBNlhGFapeXuX+fq88iLxsqGHQ4o95D2RjJG
pfil8Q9a+Hei2t7oVVGPtqp7omCOongJh6Vn5iFfRWuxJbzxVOqJ1hmdFjYZv+SygCqkiyW9Y1RIm3wd1wyYEnOCM2oB0VnmrFlERgrOqt4oQqOiSCGt1xkhcN/bwpbsJzBntr4uzcFbwEUt/fVKZx9DNKEpucexd1Pbc+tHb2bnwvvcDsRJYiqtcL0PxgnJqU2Zk8tnKwdhzUuhhsjR8k6QKBEQaDRa
rb9k+rp5/FtCf7v8AfxzitY0lJN9jOvxZo8/OVVxt5n2j8Jvi2vxcg1PU90Om6tPD0Vw0Fhqdw2BqAXhpI0xkJngEnmrX
wg+KkPxY0DUtTt7KVWwy1O401o5XDFmiYAtkdjmuo0jTbTQ9NttsPaLoO1s7eMRRQ9j/jCyqAgIBgADtXxl+z78ddW+H
+h+KNKs/hp4o8pMQDdF8HHfy/2hpMCvSzg7k9RUhzpuqKpNf16ahHM+ByrBwEtc2bIyqRrGGyRYmV
SpHXJ315b8Rv2p9a8D8A/ABSuPA1j8Mta+nWGeDXX+Ib6uExc1gBfdDXSpCyPjJu4BCP9LjPXhkA+9aeSUrUDnJcBbv
beHr5Fh1mMmIO2ZIjuXB6DFdT+15Zjwv4b0f4mMf4MafNHa+IvCcd0kttubBvIpGHCUuRagqUNo52mGSjJdCHJyTaZ1
Pwp+NXinx/4Tda+Fuv+DLfV/v4+DLVYGlW/1N1MbMCPkAByc/pXr2fas/Rb5tV0uxvHt5ESRW5eHjUJUhafcdK0q
5pWe2h0RTW7rTW7EdvkPFeBftKuKaUuoCzuLr4nW3gzTypOpyJMUaWPtYDt93HgbaW8N8+y7Z2DhN
PF2t/2ZbeKU8Z29qtneXmh6tGVG3LHKpbBTKHpBuetaUdJEVPhGqSr7sqW7BkgpgQtyG3LqL4k0E1B6lrfo9kbuG4Tyz7jw+G
DEBwHAYkgZOfXqetej1znw8nvLzWoc2owtb3zWcRmiaAwMj7BkGMklT7EnFdGaze7kzE1Ga7LjshHOFJHpXgvxv8M+LN
U0nVNN0aC/vYNUvLcypeZeZ8Z2W0jFyDBPB8kRkZRFuMrglzKxa5/wDB74S5+Hfg
T4JsfCuh1Ll1Ll3s5kuHBluJSNNMfU4HQdzAK8z0P9kzw9p9/q2fF+kP2zytB8qPGFSnnhnA3YKqrfGK1fdVOwe
6+IfD6axeaNcM5Q6fd4Z4pmx7pVBGVnaAXc4EN1OtfEsxWmJd73ARyJbVxN2Tv9wgqDBhbR4tJHahvtIMMDdBdwhjEL9ydM/ZfHD1Vm
a6KUqrlaG7MpqnGN5bI+Kv+Cl0EcX7R/PVs6bcvolsSBwc+ZKfw/lXyh1+tev/AB2+JGkVxt
Ysl/aRwo1kpbyGyhwX5+b3bivIM96xneOu8+xwsXCjG+Mu6itJTneOx96/8ABMv4v/4ABMv4nv4H/2Os3Oqf8nUnW86fT7g2Om
AySfXDAD0ya/ReMBr56/4YM8sL4hz4i4cb1qiUMS+Et67D+LzJGIJ/4DtH4V7Jo6lT5zT29DG38hZCaf9zv1gG1IdWhttNGWjcA
yMrAMpGCp6cEVUNucmeuM6z5Rb4R+NNT8aan2aanrx8Q6h/Y2kgdvtj/NMBbb7BGmRhMhgQuSOa+p7CUy2q
OVCsyglVYMBx2I6/vlz4pfDO48SfF5fdU0F4XabrttYXTPqV7fa3EsV9vt8bTCdxjKMVPKjO3MHNfSS3hQE+H7
DdawP7hP8ARRraQSRxcfdVgACB0Areo20rmEFqzXXpQ33TQvSlIyMViao+f/jiwb9oD4Fc/wDL9qf/pJHWKtIsrvh/j
xJj4L+Nxj/mEXP/os1282mW008M8kEck0JJid1BMeeu3PTp2p09tHcwvFLGskTjayPnTjayOMhh3BHerctF5ecp8lfEV

pPH3wx+DfwmtJpYG8U21rLqbwNh4rCCJXkOe2WCgH1wK1fHX7E9hq3gfUrK18beM7yZLZns7a/1hpbcSouY90e
MEBgOPavphNItI54pktolmiTy45Agyif3R6D29qt7eMVarSi/dJdGL3Pjvx38QX+Jf7Lfw81e5JGop4k0m1vo2
+9Hcw3ISUN77lNfTHxG+HekfFDwZfeHNbg86xu0xuHDxOPuyKezKcEGt4eH9O8kQmxtzD5nneX5S7d+c7sY655
zV4xKfXj0qZVL2sNQ1d2fG/wH07xr4e/a6vdI8a5vZ9L8ISWdlrAH/IQthdQmORv9vB2t7j3q38B/jl4C+Gmv/
FrTvFHiiw0S/k8Z6jOkF1LtYoZMAgelfWsmlWr30d4YIzdxxtEs5UF1RiCyg+hKqcewrOm8FaFcTPNLounyyux
d3e2QszE5JJxya1lVUt0ZxpuOx5B+0J4w0bx1+yn401rw/qMGq6VcabKIrq2bcj7X2nB9iD+Ves+BH2+CPDw45
0+3/wDRa1px6HYRaf8AYEsrdLI5zbrEojOTk/LjHXmrMdskMaoi7UUYAB6D0FYSl7tjVR1ueC/svORffGTj/md
bvr/1wgqD9jrUYNL+Cur3NxIsVtBr+ryySudqoouXJYk9AB/Kve7fTbaz8429vHCZnMkhjQKXYgDcSOp4HNJFp
NpDavbR20Udu5JeJEAVs9cjvnPNW6l7q3YXI073Pjv4O/tE/DfW/iV4t+JHizxjpmnX91KdK0awvJgHtbCNuGx
2aRvmPsBX1N4D+J/hf4n6dPfeFdatNcs4JBFJNaSblR8ZwT64q9/wg3h7/oB6dn/r1T/Cr+naLY6SjR2NnBZxs
cssEaoCfXgUVJKWwqcZR3PFfh82f2tPiiR/0CNMH6PTf2ocDV/hKTj/AJHC06/Rq9zWwt4bmS4jhRZ5AA8gUbm
A6ZPekubC3vDGbiCOYxtvjMihtjeoz0PvS5/e5h8ulmZ/icf8Uzqpx/y6S9f9w18oT6g9n/wTrsLeJ/LuNQ0uO
wiI675bjYMfnX2K0SuhVhuUjBB6GqT6HYvZJZmzg+yoQVh8tdikHIwMY4PNOM7NA4XPnP8Aae0jUvh94N8IfEb
wxZ/aNa8F4ha3A5ntJYxE6HHUBije3Nc/4++Gk3gT9iS8tbwmTXFa31/U585Z7o3Mc8xJ745XPoBX1jd6dBf20
lvcQpPbuMNFIoKn8KLjT4Lu2e3uIkngkG14pFDKw9COhHtVKs1FLzF7LW9z5/8A2qtQGq/C3wLdKdwn8TaRLxz
95ic/rXS/tJ+Atc8TeGtG17wqizeKvC+oJq2nwE4FztBWSHP+0hYV6xNpNrcwxxTW8csUZDIjoCqkdCB0BHarJ
jzUe0atboV7M8R8O/tgfDHU9IafVfEdt4a1O2Xbe6Tq7GC5tpB95CjcnB4BGQa5/wCHc958f/jPafEX7Nc2fgf
w5bS22gG5jMbX88wAludp52YAC+v4c+26r8OfC+u38d9qPh7S769j5W4ubSN5B/wIjPaugt7ZLeNY40WONRhVQ
YAHpRzxV+VCUZfaJY+B0rL8U+JLTwp4c1TWb5xHZ2FtJcyuxwAqqSf5VqgYqK7s4L23lguIknhkUq8cihlYehB
7VmjRnx/+zJ+0J8M/DfgzUda8SeN9ItPFPibUp9X1CGWfDxb2PlxH/dQDjsSa1PgR8WfCbftL+NtA8La5Z6zof
ie3TXbf7G+5YbxfkuE+rAB/wFfSX/CDeHjydD04/W1j/wAKnsvCei6ddLcWulWVvOvSWK3RWH4gV0upTfNZbnO
oSsvI8T/bYw/wesc4H/FQ6X1/6+kr362XFvH/ALoqK7022v0VLmCO4jUhgkqhhkHIOParSjC49K5m/dsbpa3PA
Pjlz+0N8Bf+v7U//SdK9h8cnHgzXj/04T/+i2rUm063uLmCeSCN5oCTFIygsmeuD2zU0kQmQo6hlIwQR1q+a9v
IhQsn5nyFYnP/AATdbj/mVm/HrVvXPB2qfDHRPB3xe8E2zzXUGi2cXiXSIBxqNmIkBkA/56xjkeor6o/sezFgb
L7JD9j27Ps+weXt9NvTHtU6WsSQCEIFiC7QgAwB0xj0q3Vtp5k+yvZnjPxk8a6R8QP2WfFev6JdLeabf6LLLDK
h7Ecg+jA8EdiDXovw6G34e+Gcj/mG23X/AK5LWzHoenw2Bso7OCOzIINusYEZB6jb0xVmK2SGJY0UJGoACrwAB
0x6Vm5WjZGnLrc8C/Z/uY7Hx18cLmeQRwReIw7SNwAot0JJPtzXmXwt/aJ+HHiL4teLfiJ4p8YaZpsysdH0Gyv
JtrwWSctLt7GVufoAK+w10q0T7SFtok+0HdNhB+8OMZb149azl8C+HcY/sPTiP+vWP/CtlUjd8yM3TfTYpeAPi
j4V+J1jPeeFdctNdtrdxFLLaPuCuRkA/hXWM2FzVDT9FsdIRksbOCzRzllgiVAT6nAq+VDDBrn06G54D+1deSe
LrDwt8MbOWSK68YaisFy8Jw8djFiSdwe3AVQf9r2rmPFn7Emmat4P1DSoPG/jS5K2xW1tLzWWkt/MUZjDIRgqG
A/yK+mp9LtZruG6e3ia5hBWOYoC6A9QD1GanEK7SDwPY1qqsoJKJk6ak22eVfsz/EKX4ifBvQr29J/tezQ6dqK
N95biE7Hz7nAb/gVc1+x+w/4Rjx3jnPjHUz/48le5Wml2un+aLW3it1kcyOIkC7mPVjjqT60+00+2sQ4t4I4A7
mRxGoAZj1J9/ek5Xv5go7PseM/tpE/8M0eMsf3LX/0qhr1vQn/4kenj/p2j/wDQRV2+sLfULZre5gjuIHxuilU
MrY5GQfenxwqgwBgDtU391Irl1ueB/s/H/i7Hx1PIz4ii6/8AXrHXkvwt+Dz/ABN+DmtX+jXI0rxpofi7VLzRd
TQcxyib7jHuj9CPx7V9oQ6ba2ss0kMEcUkzb5XjUKZGxjLEdTgDrTLbTbeyQpbQR26M5crEgUFj1OAOtae1avy
mbpJtHmvwG+MMXxV8LSi+tf7M8UaTJ9i1nSn4e1uF4OB/dbqp9D7VyX7F/wA/w98Ten/CUalg+v7wV7rb6VaW1
xNcQ20UU85/eyogDPjpk9TU1lp9tYIyW0EdujMXKxIFBY9TgdzSlNO/KrXCMXo2eJeKsL+154JGeT4a1AfX97F
/hXm3xu+L/gzxB+0RoXhPxT4isdJ8MeEgNVvlvJdq3V//AMsovcIDuPufavraWwgku0uTDG1wilFlKjcqnqAaz
bnwfol5cPNcaRYzzOdzPJbozE+pJFEaiTvIcoPozlPA3x8+HvxF1oaT4Y8V6brGo+W0n2a1l3PtXqcV6IfpWdp
/hbSNKnE9nplpazYx5kECo2PTIFae0VnKz2NI3+0IR8pry/WPg1cX/ivV9Z0/xnr2gDVNjXFrppgCl1QIGBeNi
DgDpXqNRXFxHbRl5XEaDksxwBSi3Eb8yroGmvo+kWllLe3GoyQRrGbq7IMspA+8xAAz+FaArN07XLPV/M+xzfa
FQ4MiKdhPs2MH8K0AaXXUE09h1NYUFxUVzMkcTFiAoGSW4GKYXtqeX/HLw/4s1nwlez6D4hfSGsLmHUlS2jIee
GEb5IGIOcOR2+nevgjxl8L9f/bC/a58XWcF2+naRpjeRJezRllghi/dbVHcmQPgZ7n0r6M+In7ZelzeJNUv/C8
l1daF4MhuDqtw2FtL+eRfKt7dO7fvSDu4GFbFbOjeLZ/C3xV+HdlqVrp+m65428OXK39tZnYtvchmuFcZOeWkk
U+rHNejRc6Culr0OCso1Xvp1PlD9rn9jfQv2fPAOi+IND1y71I3F0LK4juwvzsVZtybeg+U5HNfJOP/ANde1P4
o1vxf8PvGvh3xBrV1eaZ4ala+06xdss11LcJCeSCSoVnbbnr6ZrxbBzivqMJz8nLVd2eLiFFSvBH3b8Mv+CiGg
fCj4Q+7DvDvCwN2uaPqvhx5WCmYhbmFPdipDAfRT
/ADr8vu+B1ruPhZ8INe+LHiiz0fTojZpcJJI19cxusCIilmJYD2rkq5fhtZz0ubU8ZWdox2R+4uia3ZeIdJtdR
067hvrK5jEkVxA4dHU9CCOtXG5U84OK/Dz4Y/Gr1wbKDwn4g1GBlnCJpsbtJBOxYDYYSdrAn2zX7ZaTeTN
odpc6iq29z9nWS4XOFRtoLfgDmvnsVhHhmtb32PZoYhVU11R5R4g/Zh8O+Jtb8RateX+rR3WtzmW7SzvZbeKZP
KSNIpFRgHVTHuHQkse1eoeCfDy+EvC2laKkpnSwtY7YSsMF9qgZ/SpLLxXpGo3CQ22o288rjKojglh7VroQRxX
G5StaR0JK90x3SiipRYdKDSUtMAwKTApaKACk96XvR2oAMUYFHaigApMZpetBoATAowKWjnNACYFLRmigAoxm
mu4Qc0zzgRkHjpQBKOKPxqLzRnGeaXz0YcMD9KAJMCjAzUXmjkA5I60LMvc4GfwoAl4pDigHcMimbxzz060APw
KXFRmZSBtOc9MUglDDgigCWiozOqj5mxTfOHHPXpQBLgfWjApjTKB1pBOo5J4xnNAEtFQvcIgyWAHqaBcKwBDA
g96AJutAqETDIGfyqXNAC0mBSij1oAMCig0c0AJgUYApaDQAYFFFFAAeaMUUUAFGBRRQAUUUUAJxRgZpaBQAmB
6UtH1ooAKMZoozQAUUUdKAE7Vm+INKXWdMltHYoJAPmAzgjnOD16dK06a4yAOn0ovYTV9Dz/Ttans9ft4ZLiW7
jYMjSMgtreJF6lV5J5AGScc8V3sbh0Vl5DDIIPWuR8Dx6Ac+DiF4Bqd7L0K0awvJgHtbCNuGx
sEt2kkhZjEHYGVox0ZwOBk8gdgRWko8yujGL5XZmbqvinxXq2pTWnhnQYFtIX2Nq2rztFE7d/LjUF3A/vHaD2z
XFeLfg78SPiBq0Vv4Z0GBbSF9jatq8nhnQYFtIX2Nq2rztFE7d/LjUF3A/vHaD2z
m8MaLoluulynfcLdIJmuX/vyFvvH9B2AryT9qX9y5j5/jVrGkeJ/DhOuHw14r0Ozt4aLoluulynfcLdIJmuX/vyFvvH9B2AryT9qX9j5/jVrGkeJ/DOuHw14r0OzX4aLoluulynfcLdIJmuX/vyFvvH9B2AryT9qX9j5/jVrGkeJ/DOuHw14r0OzX4
1BtpuwE57+tVCrOE+dbkSpxlHlex+UvxH/ZV+I/7LE9l47s7iz8X2scm69KWzyJGcggzR5y6EjO7sQPbPy3rOo
/2xql1ffZbaxNxIX+z2qssSE9kBJwPQE1+/M9rFcxvHMgljcbWRxlSPQivNtV/Zk+FWt373154B0Ge7c7nlnNjG
Cx98CvYoZly61Y3Z5lXA83wPQ+Cv2R/2FU+L+HwXIoTxdfyWnh2WUm2062BWW6RTjLMfuqSCOOSB719E/tpfF/R

```
v2e/g8ngzwxBbWGravbNY2trbKF+y2uNskmB7fKPUsfQ19A+O/FPh74FfDHUNamgjsND0S1ylrbKEUAcKiL0BJ
IAr4J+B/wAGPEH7ZvxWvvih8QY3h8JLcfurZsgXAU/LBH/0zUcM3c5x3xkqzxFR1qz91dDWVNUoKnBas1/+CfP
7LJ1C8t/if4ntP9FgYnRbWVf9Y3T7QQew6L789hX374h1f+zvsSlo0inuBFK8nKqu1jz6ZK7ee5pzyWXhvTbe3
gjjhjQLDbW0Q2qSBhUUdh0HtXNrYXeo6iY9RLWmpycxzxr5ltPHnPlsOnHocHuD2rjq1pYip7SWxvTpqjDkjqz
W8M6XYW+q6hLp8m6ElVZAAUV+pCN2HzdOgzxXUKuB/hVexs4rKFYokSNFH3UG0fkKtVyt3Z1RVkFGaKBSKDvRR
RigAooooAB1oNFHpQACiiigAoo96SgBfWjvRR3oADRRRQB45+1f4d0zWvhFd3moIss2kXEOoWcD5K3M6sAtsyj
lxNny9vq4I5ArJ+KPh/T/AIZfBJ9Y8Paavh3UNBZNYstPtskS3fCm1wv3jLuMWPV1I5Fek/En4aWfxLstKt7zU
9T0z+zb1NQgl0yZY285AQpO5WBxuJHHBweoFV4PhXZPbaXDqmratr406/Gowtqc6OTKFKpu2ooZVyWAI4bnsKA
PJtLdtU8BfDfSzq1w1r41vpLvWb6CVopJC0MtxJAG4ZAXUR4GCApHHNdFaeFNN+DfxR8I6f4Tiex0nxE11a32j
xys0CmKBpUulUsdjAoI2I+95q55Arsbv4MeHL3w1daFJHcrYy38mpwmKcxyWdwzl98DrgxkMSRj1I6Eiqug/BH
TtKbU7m+13XvEGs39m+n/ANsardI9zbwMMFIdiIkfYkhMkgFiSBQB578NfGer3Hxbk1/UL2STwr44a4tdGgc/u
4XsyRGy/wDXxGJpAe6xA965q98YeLJPg9aWr+GZ10/+3oov7a/tNC2z+1AM7MbsHpjPSvoG++GGh3egeHtISGS
0s9Antp9OFs2xoGgwEwcdNuVI7hiKJPhlo0/hFPDbrN/ZqXK3YAk+feLjzxzjpv8A0oA6yPGwY6V8v6N8XvEHh
af4kRahIL6fVtQupfCMDnh3+0mzFuMfwiQQy59JnPRa+oQu0AVw0Xwb8Mq/hl5bN7qbw7dTXunTTPlopZQwdj6
8O36elAHi/gbxX4stdB+E7fZJvFeuxR6nZagv2pbczSQsY2lYtkHJTOP9qqt3rerSeBPit4hvraXRdc0LxFHf2
Nh9pM5M0cMBjtwV6iUnZtA/jr37R/hho2g31pd2ouPNtZ7u4iDyZCtcNul4x0z09KrXPwk0K7vbi4lFw/2nVo9
amhMv7uS4RAiZGOVG1Wx6qPpQB4hq/je28Mfsv2Wq3fiAWureOZkjn1JmZfs91dtibaG5VYEDjB5Ah55zWePGG
q/En4R/DODw/wCLHh8QWPioaNNq9vhlnlt7e6Ad1/ijlVY3IPVJOxr33Sfg/oGj6/a6tALrfaXV3eW1q0uYIJr
kL5zqmOCcMRzx5smPvVJefCnRL3xMuuOLhbsX8Op7ElxGbiKGSFX246lJCD67V9KAPMLP4iv4x8daDZ3kT6V4h
07T9Ug1bSWfmGYJFtcdN0bD5kfup9c15lfeNdV0rWfhRpSX9zpGna/4TWy1HX9+RYB54QG54WRz+7VzwrODzwK
+o9S+GHh/VfGlt4smsguvW9jNpou4ztZoJCCyN64K5GemT61nRfBXwuLSC0ms2vLSLR20I29ywdHtSysVYEcnK
jmgDkPGHhTTNQ8XeB/h5fRyDwc2mXlwbIzMFvJYGgWKCRs5cbZJJCpJ3GME5wa4Xx7preD9E+JPg/wrqF3pOi2
9ppdxbSWsxY6XcXFyUkjiJztAVI5Ah4HmHjBxXsl98FdM1SDwlDcavrLxeGZEltAblS0rIMKZWKFmO3K5BGQTn
OahtfgToVrD4nt2vdTubTxBepf3VvPOhVJVkD5QhAwyVUck8KAMUAeba1461Z9c8CeFPEcq23ivTvFNqJ2gJSP
UrYwz7LmMZ+62PmXnaykehr6SHWuW8U/DbQfGWs+HtV1SyWXUtBuxeafdLxJE+CCM91IPKng8HqAa6kCgBaO9A
FFABRRRigA70UUUAFFFFABRSUtABRRiigAooooAO1FGKMUAJ6UtJS0AFFFFABmjrRRnFAAcUYyKKKAILq0ivIX
hmjWSJ1KsjjIYHqDXE6/oKpfBJFistCtYxN5ca7Y3I+9vIwQQMFe2fpXe9agnt0nVldA6N1VhkH/OKabTJcU0c
b4W8bpfeYlx+4toV/107EMnzbQkmQMOeDj612kMyTIro4dGG5SpyCK5XVvDFy6RxWSwrF9pF0zSlt6PuyWHrkE
jHHWqPn6nBHE1vKbW6vbvZb2rICiQqD1XqOAW47kVo4p6ozUpLc7zIpa5qx8Sl9VfTJoS96jAN5IyoXaDvOeg5
xir8niTT4L37G9zifcEPyNtVj0UtjAJ7AnNZtWL5k9jT707jFZ1rrtnevcrDKXNuWWQ7GABUkEZxzgg9Kzrjxn
psc3kxyGaRkikUIvVZJBGpH4mizG5K1y74j8M6R4t0x9O1rTrbVLB2DNbXcYkQkdCQeOKi0u10jQI7fRtNitrK
OCP91ZW4CCNB6KOgrAGt6hr+nXIt9tnLbzESLlkwqkhh5hXbnvxnvUuhWn9o3UGtadCtqZXeC7QtkSqCcMCOCQ
eh9Cau2lmZ86eyM6+efV/EF7p9/pzXawMvlXVp8r26sQUbk8njqvPHIruNNt5rezijuJRcTIuGlC7dx9cdqelo
glM/lgTuqoz9yBkgH8z+dWh0qXK+iKjC2o1Rg0+iipNAooooAPxooozQAUUUUAFGKM0dqACijpRQAUUUUAFGKM
RQAUdaKM0cOaKKKAA0UUUZoAxOOtFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQ
AUUUUAFFFFABRRRQAYooo0ooAKKKKACkpaKACiiloASijoKDQAhUEVG1urOrlVLr0YjkeuK
lpaVgM630mK0u7u5jH765Kl2PPQYA+nFYFz4Ru5p7mETQ/wBn3N2l3ITu80Fdp2DtjKDnPc8V1zdaSquyeVMw9
B0BtJW6MkryPNcTS43sVUNIzABScDhucVjr8Ooft7mETQ/wBn3N2l3ITu80Fdp2DtjKDnPc8V1zdaSquyeVMw9
g8N2Vve3FysbF5yWdS5KZIwTt6c961Ik2LgcAdh0FO70q1LdykkthcUUCjvUpWGFFLSGMAUUUtACD6UUAFA
S0g6CgA70dqKBQAUlpKAEooAKEooFBoAKKPSgUAFABRRRigAooooAE
oz70tJQB//Z";

        public static string GetThesisApplicationPDF(ThesisApplication application)
        {
            var strBuilder = new StringBuilder();

            strBuilder.Append($@"
                <!DOCTYPE html
                  PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN""
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
                <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""el""
lang=""el"">

                <head>
                  <meta http-equiv=""Content-Type"" content=""text/html; charset=utf-
8"" />
                  <title>ISO 9001:2000</title>
                  <meta name=""description"" content=""Corrective Action Request
(CAR)"" />
                  <style type=""text/css"">
                    * {{
                      margin: 0;
```

```
  padding: 0;
  text-indent: 0;
  box-sizing: border-box;
}}

body {{
  font-family: Arial, sans-serif;
  line-height: 1.5;
  text-align: left;
}}

table {{
  width: 100%;
}}

.first-details-td {{
  width: 16%;
}}

.primary-details-td {{
  font-weight: 700;
  width: 20%;
}}

.position-relative {{
  position: relative;
  height: 50px;
}}

.col,
   [class*='col-'] {{
       padding-right: 0;
       padding-left: 0;
  width: 100%;
  position: relative;
   }}

.mb-1 {{
  margin-bottom: .25rem;
}}

.mt-1 {{
  margin-top: .25rem;
}}

.mt-5 {{
  margin-top: 3rem;
}}

.mx-5 {{
  margin-left: 3rem;
  margin-right: 3rem;
}}

.p-0 {{
  padding: 0;
}}

.pb-1 {{
  padding-bottom: .25rem;
}}

.pr-1 {{
```

```
  padding-right: .25rem;
}}

.pr-5 {{
  padding-right: 3rem;
}}

.pb-2 {{
  padding-bottom: .5rem;
}}

.pb-3 {{
  padding-bottom: 1rem;
}}

.pb-4 {{
  padding-bottom: 1.5rem;
}}

.font-weight-bold {{
  font-weight: 700;
}}

.text-right {{
  text-align: right;
}}

.text-center {{
  text-align: center;
}}

.border-bottom {{
  border-bottom: .5px dashed #8e8d8d !important;
}}

.undeline-dashed {{
  text-decoration: underline dashed #8e8d8d .5px;
}}

.signature-body {{
  position: absolute;
  text-align: center;
}}

.signature-body-first {{
  right: 0;
}}

.signature-body-second {{
  right: 150px;
}}

.signature-body-third {{
  right: 300px;
}}

.signature-body-fourth {{
  top: 106px;
}}

.s1 {{
  color: black;
  font-style: normal;
```

```
                    font-weight: normal;
                    text-decoration: none;
                    font-size: 16px;
                  }}

                  .s2 {{
                    color: black;
                    font-style: normal;
                    font-weight: bold;
                    text-decoration: none;
                    font-size: 12px;
                  }}

                  .s3 {{
                    color: black;
                    font-style: normal;
                    font-weight: normal;
                    text-decoration: none;
                    font-size: 12px;
                  }}

                  .s4 {{
                    color: black;
                    font-style: normal;
                    font-weight: bold;
                    text-decoration: none;
                    font-size: 10px;
                  }}

                  p {{
                    color: black;
                    font-style: normal;
                    font-weight: normal;
                    text-decoration: none;
                    font-size: 10px;
                    margin: 0px;
                  }}

                  .date-body {{
                    position: absolute;
                    right: 0;
                  }}

                  table,
                  tbody {{
                    vertical-align: top;
                    overflow: visible;
                  }}
                </style>
              </head>

              <body>
                <div class=""mx-5 p-0"">
                  <div class=""text-center"">
                    <img width=""573"" height=""123""
                    src={imgLogo} />
                  </div>
                  <div class=""s1 text-right pb-3"">Αίτηση Ανάληψης
Διπλωματικής/Πτυχιακής Εργασίας</div>
                  <table class=""s2 pb-3"">
                    <tr>
                      <td class=""first-details-td"">ΤΜΗΜΑ:</td>
                      <td>{application.Department}</td>
```

```
                        </tr>
                        <tr>
                          <td class=""first-details-td"">ΠΡΟΣ:</td>
                          <td>ΤΗΝ ΓΡΑΜΜΑΤΕΙΑ ΤΟΥ ΤΜΗΜΑΤΟΣ</td>
                        </tr>
                        <tr>
                          <td class=""first-details-td"">ΚΑΤΕΥΘΥΝΣΗ:</td>
                          <td>{application.Specialization}</td>
                        </tr>
                      </table>
                      <div class=""position-relative s3"">
                        <div class=""date-body"">
                          <div>Ημερομηνία:
<span>{DateTime.Parse(application.Date).ToString("dd/MM/yyyy")}</span></div>
                          <div>Αρ.Πρωτ.: <span></span></div>
                        </div>
                      </div>
                      <table class=""s3 pb-3"">
                        <tr>
                          <td class=""primary-details-td"">Επώνυμο</td>
                          <td class=""border-bottom"">{application.LastName}</td>
                        </tr>
                        <tr>
                          <td class=""primary-details-td"">Όνομα</td>
                          <td>{application.FirstName}</td>
                        </tr>
                        <tr>
                          <td class=""primary-details-td"">Όνομα Πατέρα</td>
                          <td>{application.FathersName}</td>
                        </tr>
                        <tr>
                          <td class=""primary-details-td"">Τηλέφωνο Επικοινωνίας</td>
                          <td>{application.Phone}</td>
                        </tr>
                        <tr>
                          <td class=""primary-details-td"">Email Επικοινωνίας</td>
                          <td>{application.Email}</td>
                        </tr>
                        <tr>
                          <td class=""primary-details-td"">Εξάμηνο Σπουδών</td>
                          <td>{application.CurrentSemester}</td>
                        </tr>
                        <tr>
                          <td class=""primary-details-td"">Αριθμός Μητρώου</td>
                          <td>{application.AcademicId}</td>
                        </tr>
                      </table>
                      <div class=""s3 pb-2"">Παρακαλούμε να ενεργήσετε για την έγκριση
της εκπόνησης της ακόλουθης εργασίας</div>
                      <div class=""s3 pb-2"">
                        <div class=""font-weight-bold"">Τίτλος διπλωματικής/πτυχιακής
στην ελληνική γλώσσα</div>
                        <div class=""undeline-dashed"">
                          {application.ProjectTitleGreek}
                        </div>
                        <div class=""font-weight-bold mt-1"">Τίτλος
διπλωματικής/πτυχιακής στην αγγλική γλώσσα</div>
                        <div class=""undeline-dashed"">
                          {application.ProjectTitleEnglish}
                        </div>
                        <div class=""font-weight-bold mt-1"">Περίληψη</div>
                        <div class=""undeline-dashed"">
                          {application.Description}
```

```
                    </div>
                  </div>
                  <div class=""s3 position-relative mt-5"">
                    <div class=""signature-body signature-body-first"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-second"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-third"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-fourth"">
                      <div>Ο/Η Επιβλέπων/ουσα<br>Καθηγητής/τρια</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(Ον/μο - Υπογραφή)</div>
                      </div>
                    </div>
                  </div>
                </body>

                </html>
            ");

            return strBuilder.ToString();
        }

        public static string GetGroupThesisApplicationPDF(ThesisApplication
application, ThesisApplication secondApplication)
        {
            var strBuilder = new StringBuilder();

            strBuilder.Append($@"
                <!DOCTYPE html
                  PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN""
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
                <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""el""
lang=""el"">

                <head>
                  <meta http-equiv=""Content-Type"" content=""text/html; charset=utf-
8"" />
                  <title>ISO 9001:2000</title>
                  <meta name=""description"" content=""Corrective Action Request
(CAR)"" />
                  <style type=""text/css"">
                    * {{
                      margin: 0;
                      padding: 0;
```

```
  text-indent: 0;
  box-sizing: border-box;
}}

body {{
  font-family: Arial, sans-serif;
  line-height: 1.5;
  text-align: left;
}}

table {{
  width: 100%;
}}

.first-details-td {{
  width: 16%;
}}

.primary-details-td {{
  font-weight: 700;
  width: 20%;
}}

.position-relative {{
  position: relative;
  height: 50px;
}}

.col,
   [class*='col-'] {{
       padding-right: 0;
       padding-left: 0;
  width: 100%;
  position: relative;
   }}

.mb-1 {{
  margin-bottom: .25rem;
}}

.mt-1 {{
  margin-top: .25rem;
}}

.mt-5 {{
  margin-top: 3rem;
}}

.mx-5 {{
  margin-left: 3rem;
  margin-right: 3rem;
}}

.p-0 {{
  padding: 0;
}}

.pb-1 {{
  padding-bottom: .25rem;
}}

.pr-1 {{
  padding-right: .25rem;
```

```
}}

.pr-5 {{
  padding-right: 3rem;
}}

.pb-2 {{
  padding-bottom: .5rem;
}}

.pb-3 {{
  padding-bottom: 1rem;
}}

.pb-4 {{
  padding-bottom: 1.5rem;
}}

.font-weight-bold {{
  font-weight: 700;
}}

.text-right {{
  text-align: right;
}}

.text-center {{
  text-align: center;
}}

.border-bottom {{
  border-bottom: .5px dashed #8e8d8d !important;
}}

.undeline-dashed {{
  text-decoration: underline dashed #8e8d8d .5px;
}}

.signature-body {{
  position: absolute;
  text-align: center;
}}

.signature-body-first {{
  right: 0;
}}

.signature-body-second {{
  right: 150px;
}}

.signature-body-third {{
  right: 300px;
}}

.signature-body-fourth {{
  top: 106px;
}}

.s1 {{
  color: black;
  font-style: normal;
  font-weight: normal;
```

```
                    text-decoration: none;
                    font-size: 16px;
                  }}

                  .s2 {{
                    color: black;
                    font-style: normal;
                    font-weight: bold;
                    text-decoration: none;
                    font-size: 12px;
                  }}

                  .s3 {{
                    color: black;
                    font-style: normal;
                    font-weight: normal;
                    text-decoration: none;
                    font-size: 12px;
                  }}

                  .s4 {{
                    color: black;
                    font-style: normal;
                    font-weight: bold;
                    text-decoration: none;
                    font-size: 10px;
                  }}

                  p {{
                    color: black;
                    font-style: normal;
                    font-weight: normal;
                    text-decoration: none;
                    font-size: 10px;
                    margin: 0px;
                  }}

                  .date-body {{
                    position: absolute;
                    right: 0;
                  }}

                  table,
                  tbody {{
                    vertical-align: top;
                    overflow: visible;
                  }}

                  .w-25 {{
                    width: 25%;
                    padding-right: 2rem;
                  }}
                </style>
              </head>

              <body>
                <div class=""mx-5 p-0"">
                  <div class=""text-center"">
                    <img width=""573"" height=""123""
                    src={imgLogo} />
                  </div>
                  <div class=""s1 text-right pb-3"">Αίτηση Ανάληψης
Διπλωματικής/Πτυχιακής Εργασίας</div>
```

```
                    <table class=""s2 pb-3"">
                      <tr>
                        <td class=""first-details-td"">ΤΜΗΜΑ:</td>
                        <td>{application.Department}</td>
                      </tr>
                      <tr>
                        <td class=""first-details-td"">ΠΡΟΣ:</td>
                        <td>ΤΗΝ ΓΡΑΜΜΑΤΕΙΑ ΤΟΥ ΤΜΗΜΑΤΟΣ</td>
                      </tr>
                      <tr>
                        <td class=""first-details-td"">ΚΑΤΕΥΘΥΝΣΗ:</td>
                        <td>{application.Specialization}</td>
                      </tr>
                    </table>
                    <div class=""position-relative s3"">
                      <div class=""date-body"">
                        <div>Ημερομηνία:
<span>{DateTime.Parse(application.Date).ToString("dd/MM/yyyy")}</span></div>
                        <div>Αρ.Πρωτ.: <span></span></div>
                      </div>
                    </div>
                    <table class=""s3 pb-3"">
                      <tr>
                        <td class=""primary-details-td"">Επώνυμο</td>
                        <td class=""w-25 border-bottom"">{application.LastName}</td>
                        <td>{secondApplication.LastName}</td>
                      </tr>
                      <tr>
                        <td class=""primary-details-td"">Ονομα</td>
                        <td class=""w-25 border-bottom"">{application.FirstName}</td>
                        <td>{secondApplication.FirstName}</td>
                      </tr>
                      <tr>
                        <td class=""primary-details-td"">Ονομα Πατέρα</td>
                        <td class=""w-25 border-
bottom"">{application.FathersName}</td>
                        <td>{secondApplication.FathersName}</td>
                      </tr>
                      <tr>
                        <td class=""primary-details-td"">Τηλέφωνο Επικοινωνίας</td>
                        <td class=""w-25 border-bottom"">{application.Phone}</td>
                        <td>{secondApplication.Phone}</td>
                      </tr>
                      <tr>
                        <td class=""primary-details-td"">Email Επικοινωνίας</td>
                        <td class=""w-25 border-bottom"">{application.Email}</td>
                        <td>{secondApplication.Email}</td>
                      </tr>
                      <tr>
                        <td class=""primary-details-td"">Εξάμηνο Σπουδών</td>
                        <td class=""w-25 border-
bottom"">{application.CurrentSemester}</td>
                        <td>{secondApplication.CurrentSemester}</td>
                      </tr>
                      <tr>
                        <td class=""primary-details-td"">Αριθμός Μητρώου</td>
                        <td class=""w-25 border-bottom"">{application.AcademicId}</td>
                        <td>{secondApplication.AcademicId}</td>
                      </tr>
                    </table>
                    <div class=""s3 pb-2"">Παρακαλούμε να ενεργήσετε για την έγκριση
της εκπόνησης της ακόλουθης εργασίας</div>
                    <div class=""s3 pb-2"">
```

```
                    <div class=""font-weight-bold"">Τίτλος διπλωματικής/πτυχιακής
στην ελληνική γλώσσα</div>
                    <div class=""undeline-dashed"">
                      {application.ProjectTitleGreek}
                    </div>
                    <div class=""font-weight-bold mt-1"">Τίτλος
διπλωματικής/πτυχιακής στην αγγλική γλώσσα</div>
                    <div class=""undeline-dashed"">
                      {application.ProjectTitleEnglish}
                    </div>
                    <div class=""font-weight-bold mt-1"">Περίληψη</div>
                    <div class=""undeline-dashed"">
                      {application.Description}
                    </div>
                  </div>
                  <div class=""s3 position-relative mt-5"">
                    <div class=""signature-body signature-body-first"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-second"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-third"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-fourth"">
                      <div>Ο/Η Επιβλέπων/ουσα<br>Καθηγητής/τρια</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(Ον/μο - Υπογραφή)</div>
                      </div>
                    </div>
                  </div>
                </body>

                </html>
            ");

            return strBuilder.ToString();
        }

        public static string GetExaminationApplicationPDF(ThesisApplication
application, ThesisApplication secondApplication)
        {
            var strBuilder = new StringBuilder();

            strBuilder.Append($@"
                <!DOCTYPE html
                  PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN""
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
```

```
                <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""el""
lang=""el"">

                <head>
                  <meta http-equiv=""Content-Type"" content=""text/html; charset=utf-
8"" />
                  <title>ISO 9001:2000</title>
                  <meta name=""description"" content=""Corrective Action Request
(CAR)"" />
                  <style type=""text/css"">
                    * {{
                      margin: 0;
                      padding: 0;
                      text-indent: 0;
                      box-sizing: border-box;
                    }}

                    body {{
                      color: black;
                      font-family: Arial, sans-serif;
                      line-height: 1.5;
                      text-align: left;
                    }}

                    table {{
                      width: 100%;
                    }}

                    .first-details-td {{
                      width: 16%;
                    }}

                    .data-table,
                    .details-table {{
                      border: 1px solid black;
                    }}

                    .data-table td {{
                      border: 1px solid black;
                      padding: 0 .25rem;
                    }}

                    .position-relative {{
                      position: relative;
                      height: 50px;
                    }}

                    .col,
                       [class*='col-'] {{
                          padding-right: 0;
                          padding-left: 0;
                      width: 100%;
                      position: relative;
                       }}

                    .mb-1 {{
                      margin-bottom: .25rem;
                    }}

                    .mt-5 {{
                      margin-top: 3rem;
                    }}
```

```
.mx-5 {{
  margin-left: 3rem;
  margin-right: 3rem;
}}

.my-5 {{
  margin-top: 3rem;
  margin-bottom: 3rem;
}}

.p-0 {{
  padding: 0;
}}

.pb-1 {{
  padding-bottom: .25rem;
}}

.pr-1 {{
  padding-right: .25rem;
}}

.pr-5 {{
  padding-right: 3rem;
}}

.pl-1 {{
  padding-left: .25rem;
}}

.pl-3 {{
  padding-left: 1rem;
}}

.pb-2 {{
  padding-bottom: .5rem;
}}

.pb-3 {{
  padding-bottom: 1rem;
}}

.pb-4 {{
  padding-bottom: 1.5rem;
}}

.font-weight-bold {{
  font-weight: 700;
}}

.text-right {{
  text-align: right;
}}

.text-center {{
  text-align: center;
}}

.border-bottom {{
  border-bottom: .5px dashed #8e8d8d !important;
}}

.undeline-dashed {{
```

```
                    text-decoration: underline dashed #8e8d8d .5px;
                  }}

                  .signature-body {{
                    position: absolute;
                    text-align: center;
                  }}

                  .signature-body-first {{
                    right: 0px;
                  }}

                  .signature-body-second {{
                    right: 200px;
                  }}

                  .signature-body-third {{
                    right: 400px;
                  }}

                  .signature-body-fourth {{
                    top: 106px;
                  }}

                  .s1 {{
                    color: black;
                    font-style: normal;
                    font-weight: normal;
                    text-decoration: none;
                    font-size: 20px;
                  }}

                  .s2 {{
                    font-weight: 700;
                    font-size: 16px;
                  }}

                  .s3 {{
                    font-weight: 400;
                    font-size: 16px;
                  }}

                  .date-body {{
                    position: absolute;
                    right: 0;
                  }}

                  table,
                  tbody {{
                    vertical-align: top;
                    overflow: visible;
                  }}
                </style>
              </head>

              <body>
                <div class=""mx-5 p-0"">
                  <div class=""text-center"">
                    <img width=""573"" height=""123""
                    src={imgLogo} />
                  </div>
                  <div class=""s1 text-right pb-3"">Αίτηση εξέτασης
Διπλωματικής/Πτυχιακής εργασίας</div>
```

```
                    <div class=""s2 pb-3"">
                      <div>ΤΜΗΜΑ {application.Department}</div>
                      <div>ΠΡΟΣ ΓΡΑΜΜΑΤΕΙΑ</div>
                    </div>
                    <div class=""position-relative s3 mb-1"">
                      <div class=""date-body"">
                        <div>Ημερομηνία:
<span>{DateTime.Parse(application.Date).ToString("dd/MM/yyyy")}</span></div>
                        <div>Αρ.Πρωτ.: <span></span></div>
                      </div>
                    </div>
                    <table class=""data-table s3"">
                      <colgroup>
                        <col>
                        <col span=""2"" width=""40%"">
                      </colgroup>
                      <tr>
                        <td class=""font-weight-bold"">Επώνυμο</td>
                        <td>{application.LastName}</td>
                        <td>{secondApplication.LastName}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Όνομα</td>
                        <td>{application.FirstName}</td>
                        <td>{secondApplication.FirstName}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Όνομα Πατέρα</td>
                        <td>{application.FathersName}</td>
                        <td>{secondApplication.FathersName}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Τηλέφωνο Επικοινωνίας</td>
                        <td>{application.Phone}</td>
                        <td>{secondApplication.Phone}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Email Επικοινωνίας</td>
                        <td>{application.Email}</td>
                        <td>{secondApplication.Email}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Εξάμηνο Σπουδών</td>
                        <td>{application.CurrentSemester}</td>
                        <td>{secondApplication.CurrentSemester}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Αριθμός Μητρώου</td>
                        <td>{application.AcademicId}</td>
                        <td>{secondApplication.AcademicId}</td>
                      </tr>
                    </table>
                    <div class=""s3 pb-2 my-5"">
                      <div>
                        Παρακαλούμε να ενεργήσετε για τον ορισμό της επιτροπής
εξέτασης και την αξιολόγηση της εργασίας μας με
                        <span class=""font-weight-bold""> θέμα:</span>
                      </div>
                      <div>{application.ProjectTitleGreek}</div>
                    </div>
                    <table class=""details-table s3"">
                      <tr>
                        <td class=""border-bottom pl-3"">1.
```

```
                        <span class=""pl-1"">Η συγκεκριμένη αίτηση υποβάλλεται μετά
την υποβολή σχετικής
                        <span class=""font-weight-bold""> εισήγησης από τον
Επιβλέποντα Καθηγητή</span>.</span>
                      </td>
                    </tr>
                    <tr>
                      <td class=""pl-3"">2.
                        <span class=""pl-1"">Συνυπόβαλλεται το
                        <span class=""font-weight-bold""> cd </span>με την
εργασία.</span>
                      </td>
                    </tr>
                  </table>
                  <div class=""s3 position-relative mt-5"">
                    <div class=""signature-body signature-body-first"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-second"">
                      <div>Ο/Η Αιτών/ούσα</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(υπογραφή)</div>
                      </div>
                    </div>
                    <div class=""signature-body signature-body-fourth"">
                      <div>Ο/Η Επιβλέπων/ουσα<br>Καθηγητής/τρια</div>
                      <div>
                        <div class=""border-bottom mb-1 mt-5""></div>
                        <div>(Ον/μο - Υπογραφή)</div>
                      </div>
                    </div>
                  </div>
                </body>

                </html>
            ");

            return strBuilder.ToString();
        }

        public static string GetExtensionApplicationPDF(ThesisApplication application,
ThesisApplication secondApplication)
        {
            var strBuilder = new StringBuilder();

            strBuilder.Append($@"
                <!DOCTYPE html
                  PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN""
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
                <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""el""
lang=""el"">

                <head>
                  <meta http-equiv=""Content-Type"" content=""text/html; charset=utf-
8"" />
                  <title>ISO 9001:2000</title>
```

```
                <meta name=""description"" content=""Corrective Action Request
(CAR)"" />
                <style type=""text/css"">
                  * {{
                    margin: 0;
                    padding: 0;
                    text-indent: 0;
                    box-sizing: border-box;
                  }}

                  body {{
                    color: black;
                    font-family: Arial, sans-serif;
                    line-height: 1.5;
                    text-align: left;
                  }}

                  table {{
                    width: 100%;
                  }}

                  .first-details-td {{
                    width: 16%;
                  }}

                  .data-table,
                  .details-table {{
                    border: 1px solid black;
                  }}

                  .data-table td {{
                    border: 1px solid black;
                    padding: 0 .25rem;
                  }}

                  .position-relative {{
                    position: relative;
                    height: 50px;
                  }}

                  .col,
                     [class*='col-'] {{
                        padding-right: 0;
                        padding-left: 0;
                    width: 100%;
                    position: relative;
                     }}

                  .mb-1 {{
                    margin-bottom: .25rem;
                  }}

                  .mt-5 {{
                    margin-top: 3rem;
                  }}

                  .mx-5 {{
                    margin-left: 3rem;
                    margin-right: 3rem;
                  }}

                  .my-5 {{
                    margin-top: 3rem;
```

```
  margin-bottom: 3rem;
}}

.p-0 {{
  padding: 0;
}}

.pb-1 {{
  padding-bottom: .25rem;
}}

.pr-1 {{
  padding-right: .25rem;
}}

.pr-5 {{
  padding-right: 3rem;
}}

.pl-1 {{
  padding-left: .25rem;
}}

.pl-3 {{
  padding-left: 1rem;
}}

.pb-2 {{
  padding-bottom: .5rem;
}}

.pb-3 {{
  padding-bottom: 1rem;
}}

.pb-4 {{
  padding-bottom: 1.5rem;
}}

.font-weight-bold {{
  font-weight: 700;
}}

.text-right {{
  text-align: right;
}}

.text-center {{
  text-align: center;
}}

.border-bottom {{
  border-bottom: .5px dashed #8e8d8d !important;
}}

.undeline-dashed {{
  text-decoration: underline dashed #8e8d8d .5px;
}}

.signature-body {{
  position: absolute;
  text-align: center;
}}
```

```
                    .signature-body-first {{
                      right: 0px;
                    }}

                    .signature-body-second {{
                      right: 200px;
                    }}

                    .signature-body-third {{
                      right: 400px;
                    }}

                    .signature-body-fourth {{
                      top: 106px;
                    }}

                    .s1 {{
                      color: black;
                      font-style: normal;
                      font-weight: normal;
                      text-decoration: none;
                      font-size: 20px;
                    }}

                    .s2 {{
                      font-weight: 700;
                      font-size: 16px;
                    }}

                    .s3 {{
                      font-weight: 400;
                      font-size: 16px;
                    }}

                    .date-body {{
                      position: absolute;
                      right: 0;
                    }}

                    table,
                    tbody {{
                      vertical-align: top;
                      overflow: visible;
                    }}
                  </style>
                </head>

                <body>
                  <div class=""mx-5 p-0"">
                    <div class=""text-center"">
                      <img width=""573"" height=""123""
                      src={imgLogo} />
                    </div>
                    <div class=""s1 text-right pb-3"">Αίτηση εξέτασης
Διπλωματικής/Πτυχιακής εργασίας</div>
                    <div class=""s2 pb-3"">
                      <div>ΤΜΗΜΑ {application.Department}</div>
                      <div>ΠΡΟΣ ΓΡΑΜΜΑΤΕΙΑ</div>
                    </div>
                    <div class=""position-relative s3 mb-1"">
                      <div class=""date-body"">
```

```
                        <div>Ημερομηνία:
<span>{DateTime.Parse(application.Date).ToString("dd/MM/yyyy")}</span></div>
                        <div>Αρ.Πρωτ.: <span></span></div>
                      </div>
                    </div>
                    <table class=""data-table s3"">
                      <colgroup>
                        <col>
                        <col span=""2"" width=""40%"">
                      </colgroup>
                      <tr>
                        <td class=""font-weight-bold"">Επώνυμο</td>
                        <td>{application.LastName}</td>
                        <td>{secondApplication.LastName}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Όνομα</td>
                        <td>{application.FirstName}</td>
                        <td>{secondApplication.FirstName}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Όνομα Πατέρα</td>
                        <td>{application.FathersName}</td>
                        <td>{secondApplication.FathersName}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Τηλέφωνο Επικοινωνίας</td>
                        <td>{application.Phone}</td>
                        <td>{secondApplication.Phone}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Email Επικοινωνίας</td>
                        <td>{application.Email}</td>
                        <td>{secondApplication.Email}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Εξάμηνο Σπουδών</td>
                        <td>{application.CurrentSemester}</td>
                        <td>{secondApplication.CurrentSemester}</td>
                      </tr>
                      <tr>
                        <td class=""font-weight-bold"">Αριθμός Μητρώου</td>
                        <td>{application.AcademicId}</td>
                        <td>{secondApplication.AcademicId}</td>
                      </tr>
                    </table>
                    <div class=""s3 pb-2 my-5"">
                      <div>
                        Παρακαλούμε να ενεργήσετε για την παράταση της προσθεσμίας
ολοκλήρωσης της εργασίας μας με
                        <span class=""font-weight-bold""> θέμα:</span>
                      </div>
                      <div>{application.ProjectTitleGreek}</div>
                    </div>
                    <table class=""details-table s3"">
                      <tr>
                        <td class=""border-bottom pl-3"">1.
                          <span class=""pl-1"">Η συγκεκριμένη αίτηση υποβάλλεται με
την
                          <span class=""font-weight-bold""> σύμφωνη γνώμη του
Επιβλέποντα Καθηγητή</span>.</span>
                        </td>
                      </tr>
```

```
                    </table>
                    <div class=""s3 position-relative mt-5"">
                      <div class=""signature-body signature-body-first"">
                        <div>Ο/Η Αιτών/ούσα</div>
                        <div>
                          <div class=""border-bottom mb-1 mt-5""></div>
                          <div>(υπογραφή)</div>
                        </div>
                      </div>
                      <div class=""signature-body signature-body-second"">
                        <div>Ο/Η Αιτών/ούσα</div>
                        <div>
                          <div class=""border-bottom mb-1 mt-5""></div>
                          <div>(υπογραφή)</div>
                        </div>
                      </div>
                      <div class=""signature-body signature-body-fourth"">
                        <div>Ο/Η Επιβλέπων/ουσα<br>Καθηγητής/τρια</div>
                        <div>
                          <div class=""border-bottom mb-1 mt-5""></div>
                          <div>(Ον/μο - Υπογραφή)</div>
                        </div>
                      </div>
                    </div>
                  </body>

                  </html>
            ");

            return strBuilder.ToString();
        }

        public static string GetEvaluationPDF(ThesisApplication application,
Assignment assignment)
        {
            var examination = assignment.Examinations.FirstOrDefault(e => e.StudentId
== application.AcademicId);
            var strBuilder = new StringBuilder();

            strBuilder.Append($@"
                <!DOCTYPE html
                      PUBLIC ""-//W3C//DTD XHTML 1.0 Transitional//EN""
""http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"">
                    <html xmlns=""http://www.w3.org/1999/xhtml"" xml:lang=""el""
lang=""el"">

                    <head>
                      <meta http-equiv=""Content-Type"" content=""text/html;
charset=utf-8"" />
                      <title>Thesis Examination</title>
                      <meta name=""description"" content=""Corrective Action Request
(CAR)"" />
                      <style type=""text/css"">
                        * {{
                          margin: 0;
                          padding: 0;
                          text-indent: 0;
                          box-sizing: border-box;
                        }}

                        body {{
                          color: black;
```

```
  font-family: Arial, sans-serif;
  line-height: 1.5;
  text-align: left;
}}

table {{
  width: 100%;
}}

.first-details-td {{
  width: 8%;
}}

.data-table,
.details-table {{
  border: 1px solid black;
}}

.data-table td {{
  border: 1px solid black;
  padding: 0 .25rem;
}}

.light-gray-td {{
  background: #dedede;
}}

.dark-gray-td {{
  background: #7b7b7b;
}}

.position-relative {{
  position: relative;
  height: 50px;
}}

.col,
   [class*='col-'] {{
      padding-right: 0;
      padding-left: 0;
  width: 100%;
  position: relative;
   }}

.mb-1 {{
  margin-bottom: .25rem;
}}

.mt-5 {{
  margin-top: 3rem;
}}

.mx-5 {{
  margin-left: 3rem;
  margin-right: 3rem;
}}

.my-5 {{
  margin-top: 3rem;
  margin-bottom: 3rem;
}}

.my-3 {{
```

```
  margin-top: 1rem;
  margin-bottom: 1rem;
}}

.p-0 {{
  padding: 0;
}}

.py-3 {{
  padding-top: 1rem;
  padding-bottom: 1rem;
}}

.pb-1 {{
  padding-bottom: .25rem;
}}

.pr-1 {{
  padding-right: .25rem;
}}

.pr-5 {{
  padding-right: 3rem;
}}

.pl-1 {{
  padding-left: .25rem;
}}

.pl-3 {{
  padding-left: 1rem;
}}

.pb-2 {{
  padding-bottom: .5rem;
}}

.pb-3 {{
  padding-bottom: 1rem;
}}

.pb-4 {{
  padding-bottom: 1.5rem;
}}

.font-weight-bold {{
  font-weight: 700;
}}

.text-right {{
  text-align: right;
}}

.text-center {{
  text-align: center;
}}

.border-bottom {{
  border-bottom: .5px dashed #8e8d8d !important;
}}

.undeline-dashed {{
  text-decoration: underline dashed #8e8d8d .5px;
```

```
    }}

    .signatures-container {{
      margin-top: 2rem;
    }}

    .signature-body {{
      position: absolute;
      text-align: center;
    }}

    .signature-body-left {{
      left: 0px;
    }}

    .signature-body-center {{
      right: 333px;
    }}

    .signature-body-right {{
      right: 0px;
    }}

    .s1 {{
      color: black;
      font-style: normal;
      font-weight: normal;
      text-decoration: none;
      font-size: 20px;
    }}

    .s2 {{
      font-weight: 700;
      font-size: 16px;
    }}

    .s3 {{
      font-weight: 400;
      font-size: 16px;
    }}

    table,
    tbody {{
      vertical-align: top;
      overflow: visible;
    }}

    .line-height-extend {{
      line-height: 1.4;
    }}

    .date-body {{
      position: absolute;
      right: 0px;
    }}
  </style>
</head>

<body>
  <div class=""mx-5 p-0"">
    <div class=""text-center"">
      <img width=""573"" height=""123""
      src={imgLogo} />
```

```
                        </div>
                        <div class=""s1 text-right pb-3"">Πρακτικό Αξιολόγησης
Πτυχιακής Εργασίας</div>
                        <table class=""s2 pb-3"">
                          <tr>
                            <td class=""first-details-td"">ΤΜΗΜΑ:</td>
                            <td>{application.Department}</td>
                          </tr>
                          <tr>
                            <td class=""first-details-td"">ΠΡΟΣ:</td>
                            <td>ΤΗΝ ΓΡΑΜΜΑΤΕΙΑ ΤΟΥ ΤΜΗΜΑΤΟΣ</td>
                          </tr>
                          <tr>
                            <td class=""first-details-td"">ΚΑΤΕΥΘΥΝΣΗ:</td>
                            <td>{application.Specialization}</td>
                          </tr>
                        </table>
                        <div class=""position-relative"">
                          <div class=""date-body"">
                            <div>Ημερομηνία:
<span>{DateTime.Now.ToString(""dd/MM/yyyy"")}</span></div>
                            <div>Αρ. Πρωτ.: <span></span></div>
                          </div>
                        </div>
                        <table class=""data-table s3"">
                          <colgroup>
                            <col span=""1"" width=""15%"">
                            <col span=""1"" width=""35%"">
                            <col span=""1"" width=""15%"">
                            <col span=""1"" width=""35%"">
                          </colgroup>
                          <tr>
                            <td class=""font-weight-bold"">Επώνυμο</td>
                            <td>{application.LastName}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold"">Ονομα</td>
                            <td>{application.FirstName}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold"">Αριθμός Μητρώου</td>
                            <td colspan=""3"">{application.AcademicId}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold"">Θέμα Πτυχιακής
Εργασίας</td>
                            <td colspan=""3"">{application.ProjectTitleGreek}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold"">Subject of Thesis</td>
                            <td colspan=""3"">{application.ProjectTitleEnglish}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold"">Επιβλέπων/ουσα
Καθηγητής/τρια</td>
                            <td>{application.ResponsibleProfessor}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold"">Ημ/νία Παρουσίασης
Πτυχιακής Εργασίας</td>
                            <td
colspan=""3"">{assignment.Examinations.FirstOrDefault(e => e.StudentId ==
application.AcademicId).PresenationDate.ToString(""dd/MM/yyyy"")}</td>
```

```
                        </tr>
                      </table>
                      <div class=""font-weight-bold my-3 text-
center"">ΒΑΘΜΟΛΟΓΙΑ</div>
                      <table class=""data-table s3"">
                        <colgroup>
                          <col span=""1"" width=""25%"">
                          <col span=""1"" width=""10%"">
                          <col>
                        </colgroup>
                        <tr>
                          <td rowspan=""3"" class=""font-weight-bold light-gray-
td"">Κριτήριο*</td>
                          <td rowspan=""3"" class=""font-weight-bold text-center
light-gray-td"">Βαρύτητα<br>(W)<br>Κριτηρίου*</td>
                          <td colspan=""2"" class=""font-weight-bold text-center
light-gray-td"">Αξιολογητής 1</td>
                          <td colspan=""2"" class=""font-weight-bold text-center
light-gray-td"">Αξιολογητής 2</td>
                          <td colspan=""2"" class=""font-weight-bold text-center
light-gray-td"">Αξιολογητής 3</td>
                          <tr>
                            <td colspan=""2"" class=""text-
center"">{assignment.ExaminationCommitee.Examiner1}</td>
                            <td colspan=""2"" class=""text-
center"">{assignment.ExaminationCommitee.Examiner2}</td>
                            <td colspan=""2"" class=""text-
center"">{assignment.ExaminationCommitee.Examiner3}</td>
                          </tr>
                          <tr>
                            <td class=""font-weight-bold text-center light-gray-
td"">Βαθμός<br>(B1)</td>
                            <td class=""font-weight-bold text-center light-gray-
td"">W*B1</td>
                            <td class=""font-weight-bold text-center light-gray-
td"">Βαθμός<br>(B2)</td>
                            <td class=""font-weight-bold text-center light-gray-
td"">W*B2</td>
                            <td class=""font-weight-bold text-center light-gray-
td"">Βαθμός<br>(B3)</td>
                            <td class=""font-weight-bold text-center light-gray-
td"">W*B3</td>
                          </tr>
                        </tr>
                        {PrintEvaluationTable(examination)}
                      </table>
                      <div class=""s3 py-3 pl-3 line-height-extend"">
                        * Τα κριτήρια και η βαρύτητά τους αποφασίζονται από τη
Συνέλευση του Τμήματος<br>
                        Η Επιτροπή, αφού παρακολούθησε την παρουσίαση της εργασίας,
αξιολόγησε τον/την παραπάνω φοιτητή/φοιτήτρια και απονέμει το βαθμό (ολογράφως και
αριθμητικά): <strong>{(examination.Grade.HasValue ?
ConvertGradeToText(examination.Grade.Value) : string.Empty)}
({(examination.Grade.HasValue ? examination.Grade.Value.ToString("0.#") : "-
")})</strong><br>
                        Συνημμένα: CD με την Πτυχιακή/Διπλωματική Εργασία, ελεγμένο
από τον/την επιβλέπων Καθηγητή/επιβλέπουσα Καθηγήτρια:
{assignment.ThesisProject.Owner}<br>
                      </div>
                      <div class=""font-weight-bold text-center"">ΤΑ ΜΕΛΗ ΤΗΣ
ΕΠΙΤΡΟΠΗΣ ΕΞΕΤΑΣΗΣ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ</div>
                      <div class=""s3 position-relative signatures-container"">
```

```
                            <div class=""signature-body signature-body-left"">(Ονομ/μο,
Υπογραφή)</div>
                            <div class=""signature-body signature-body-
center"">(Ονομ/μο, Υπογραφή)</div>
                            <div class=""signature-body signature-body-right"">(Ονομ/μο,
Υπογραφή)</div>
                        </div>
                      </div>
                    </body>

                    </html>
            ");

            return strBuilder.ToString();
        }

        static string PrintEvaluationTable(Examination examination)
        {
            var provider = CultureInfo.InvariantCulture;
            var criteria = examination.Criteria.Split(';');
            var criteriaWeights = examination.CriteriaWeights.Split(';');
            var examiner1scores = examination.Examiner1Scores.Split(';');
            var examiner2scores = examination.Examiner2Scores.Split(';');
            var examiner3scores = examination.Examiner3Scores.Split(';');

            decimal score1 = 0;
            decimal score2 = 0;
            decimal score3 = 0;

            string result = string.Empty;
            for (int i = 0; i < criteria.Count(); i++)
            {
                result += "<tr>";
                result += $"<td class=\"text-center\">{criteria[i]}</td>";
                result += $"<td class=\"text-
center\">{decimal.Parse(criteriaWeights[i], provider).ToString("0.#")}</td>";
                result += $"<td class=\"text-
center\">{decimal.Parse(examiner1scores[i], provider).ToString("0.#")}</td>";
                result += $"<td class=\"text-
center\">{(decimal.Parse(criteriaWeights[i], provider) *
decimal.Parse(examiner1scores[i], provider)).ToString("0.#")}</td>";
                result += $"<td class=\"text-
center\">{decimal.Parse(examiner2scores[i], provider).ToString("0.#")}</td>";
                result += $"<td class=\"text-
center\">{(decimal.Parse(criteriaWeights[i], provider) *
decimal.Parse(examiner2scores[i], provider)).ToString("0.#")}</td>";
                result += $"<td class=\"text-
center\">{decimal.Parse(examiner3scores[i], provider).ToString("0.#")}</td>";
                result += $"<td class=\"text-
center\">{(decimal.Parse(criteriaWeights[i], provider) *
decimal.Parse(examiner3scores[i], provider)).ToString("0.#")}</td>";

                score1 += (decimal.Parse(criteriaWeights[i], provider) *
decimal.Parse(examiner1scores[i], provider));
                score2 += (decimal.Parse(criteriaWeights[i], provider) *
decimal.Parse(examiner2scores[i], provider));
                score3 += (decimal.Parse(criteriaWeights[i], provider) *
decimal.Parse(examiner3scores[i], provider));
                result += "</tr>";
            }

            result += @$"<tr>
```

```
                            <td colspan=""2"" class=""light-gray-td"">Βαθμολογία
Αξιολογητή<br>Σχόλιο Αξιολογητών</td>
                            <td class=""dark-gray-td text-center""></td>
                            <td class=""text-center"">{score1.ToString("0.#")}</td>
                            <td class=""dark-gray-td text-center""></td>
                            <td class=""text-center"">{score2.ToString("0.#")}</td>
                            <td class=""dark-gray-td text-center""></td>
                            <td class=""text-center"">{score3.ToString("0.#")}</td>
                        </tr>
                        <tr>
                            <td colspan=""2"" class=""light-gray-td"">Σχόλια
Αξιολογητή 1</td>
                            <td colspan=""6"">{examination.Examiner1Comments}</td>
                        </tr>
                        <tr>
                            <td colspan=""2"" class=""light-gray-td"">Σχόλια
Αξιολογητή 2</td>
                            <td colspan=""6"">{examination.Examiner2Comments}</td>
                        </tr>
                        <tr>
                            <td colspan=""2"" class=""light-gray-td"">Σχόλια
Αξιολογητή 3</td>
                            <td colspan=""6"">{examination.Examiner3Comments}</td>
                        </tr>
                        <tr>
                            <td colspan=""2"" class=""light-gray-td"">Βαθμολογία
Πτυχιακής Εργασίας</td>
                            <td colspan=""6"">{(examination.Grade.HasValue ?
examination.Grade.Value.ToString("0.#") : string.Empty)}</td>
                        </tr>";

            return result;
        }

        public static string ConvertGradeToText(decimal grade)
        {
            if (grade < 0 || grade > 10)
                return "ΜΗ ΕΓΚΥΡΟΣ ΒΑΘΜΟΣ";

            int wholePart = (int)grade;
            int decimalPart = (int)((grade - wholePart) * 10);

            string wholeText = GetWholeText(wholePart);
            string decimalText = GetDecimalText(decimalPart);

            if (decimalPart == 0)
                return wholeText;
            else
                return $"{wholeText} ΚΑΙ {decimalText} ΔΕΚΑΤΑ";
        }

        private static string GetWholeText(int wholePart)
        {
            switch (wholePart)
            {
                case 0:
                    return "ΜΗΔΕΝ";
                case 1:
                    return "ΕΝΑ";
                case 2:
                    return "ΔΥΟ";
                case 3:
                    return "ΤΡΙΑ";
```

```
                case 4:
                    return "ΤΕΣΣΕΡΑ";
                case 5:
                    return "ΠΕΝΤΕ";
                case 6:
                    return "ΕΞΙ";
                case 7:
                    return "ΕΠΤΑ";
                case 8:
                    return "ΟΚΤΩ";
                case 9:
                    return "ΕΝΝΕΑ";
                case 10:
                    return "ΔΕΚΑ";
                default:
                    return "";
            }
        }

        private static string GetDecimalText(int decimalPart)
        {
            switch (decimalPart)
            {
                case 0:
                    return "";
                case 1:
                    return "ΕΝΑ";
                case 2:
                    return "ΔΥΟ";
                case 3:
                    return "ΤΡΙΑ";
                case 4:
                    return "ΤΕΣΣΕΡΑ";
                case 5:
                    return "ΠΕΝΤΕ";
                case 6:
                    return "ΕΞΙ";
                case 7:
                    return "ΕΠΤΑ";
                case 8:
                    return "ΟΚΤΩ";
                case 9:
                    return "ΕΝΝΕΑ";
                case 10:
                    return "ΔΕΚΑ";
                default:
                    return "";
            }
        }
    }
}
```

Κώδικας C# - Main

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

namespace ThesisManagement
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).Build().Run();
        }

        public static IHostBuilder CreateHostBuilder(string[] args) =>
            Host.CreateDefaultBuilder(args)
                .ConfigureWebHostDefaults(webBuilder =>
                {
                    webBuilder.UseStartup<Startup>();
                });
    }
}
```

Statup.cs

```
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.HttpsPolicy;
using Microsoft.AspNetCore.SpaServices.AngularCli;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using ITfoxtec.Identity.Saml2;
using ITfoxtec.Identity.Saml2.Schemas.Metadata;
using System;
using System.Linq;
using ITfoxtec.Identity.Saml2.MvcCore.Configuration;
using Microsoft.AspNetCore.Authentication;
using ITfoxtec.Identity.Saml2.Schemas;
using System.Net;
using System.Threading.Tasks;
using Microsoft.IdentityModel.Logging;
using Newtonsoft.Json;
using ThesisManagement.Repositories;
using Microsoft.EntityFrameworkCore;
using ThesisManagement.Data;
using System.Reflection;
using System.IO;
using DinkToPdf;
using DinkToPdf.Contracts;
using ITfoxtec.Identity.Saml2.Util;

namespace ThesisManagement
{
    public class Startup
    {
        public Startup(IConfiguration configuration, IWebHostEnvironment env)
        {
            Configuration = configuration;
            ContentRoot = env.ContentRootPath;
        }

        public IConfiguration Configuration { get; }
```

```
        public string ContentRoot;

        // This method gets called by the runtime. Use this method to add services to
the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.Configure<Saml2Configuration>(Configuration.GetSection("Saml2"));
            services.Configure<Saml2Configuration>(saml2Configuration =>
            {
                saml2Configuration.SignAuthnRequest = true;

                // Use only with signing certificate
                //saml2Configuration.SigningCertificate =
CertificateUtil.Load(Path.Combine(ContentRoot,
Configuration["Saml2:SigningCertificateFile"]),
Configuration["Saml2:SigningCertificatePassword"]);

//saml2Configuration.SignatureValidationCertificates.Add(CertificateUtil.Load(AppEnvir
onment.MapToPhysicalFilePath(Configuration["Saml2:SignatureValidationCertificateFile"]
)));_hostingEnvironment

                saml2Configuration.AllowedAudienceUris.Add(saml2Configuration.Issuer);

                var entityDescriptor = new EntityDescriptor();
                entityDescriptor.ReadIdPSsoDescriptorFromUrl(new
Uri(Configuration["Saml2:IdPMetadata"]));
                if (entityDescriptor.IdPSsoDescriptor != null)
                {
                    //saml2Configuration.AllowedIssuer = entityDescriptor.EntityId;
                    saml2Configuration.SingleSignOnDestination =
entityDescriptor.IdPSsoDescriptor.SingleSignOnServices.First().Location;
                    //saml2Configuration.SingleLogoutDestination =
entityDescriptor.IdPSsoDescriptor.SingleLogoutServices.First().Location;

saml2Configuration.SignatureValidationCertificates.AddRange(entityDescriptor.IdPSsoDes
criptor.SigningCertificates);
                }
                else
                {
                    throw new Exception("IdPSsoDescriptor not loaded from metadata.");
                }
            });

            services.AddSaml2();

            // Add Database
            services.AddDbContext<UniwaDbContext>(options =>

options.UseSqlServer(Configuration.GetConnectionString("DBConnection")));

            services.AddControllersWithViews().AddNewtonsoftJson(options => {
                options.SerializerSettings.ReferenceLoopHandling =
ReferenceLoopHandling.Ignore;
            });
            // In production, the Angular files will be served from this directory
            services.AddSpaStaticFiles(configuration =>
            {
                configuration.RootPath = "ClientApp/dist";
            });

            // Add DinkToPDF
            services.AddSingleton(typeof(IConverter), new SynchronizedConverter(new
PdfTools()));
```

```
            // Add Repositories
            services.AddScoped<IProjectsRepository, ProjectsRepository>();
            services.AddScoped<IApplicationsRepository, ApplicationsRepository>();
            services.AddScoped<IAssignmentsRepository, AssignmentsRepository>();
            services.AddScoped<IValuesRepository, ValuesRepository>();

            // Register the Swagger generator, defining 1 or more Swagger documents
            services.AddSwaggerGen(c =>
            {
                // Set the comments path for the Swagger JSON and UI.
                var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
                var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
                c.IncludeXmlComments(xmlPath);
            });
        }

        // This method gets called by the runtime. Use this method to configure the
HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
                IdentityModelEventSource.ShowPII = true;
            }
            else
            {
                app.UseDeveloperExceptionPage();

                app.UseExceptionHandler("/Error");
                // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }

            app.UseHttpsRedirection();
            app.UseStaticFiles();
            if (!env.IsDevelopment())
            {
                app.UseSpaStaticFiles();
            }

            // Enable middleware to serve generated Swagger as a JSON endpoint.
            app.UseSwagger();

            // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
            // specifying the Swagger JSON endpoint.
            app.UseSwaggerUI(c =>
            {
                c.SwaggerEndpoint("/swagger/v1/swagger.json", "UNIWA Thesis Management
API V1");
                c.RoutePrefix = "swagger";
            });

            app.UseRouting();

            app.UseSaml2();

            app.MapWhen(
                context =>
                {
```

```
                    return !context.User.Identity.IsAuthenticated &&
context.Request.Path.Value.StartsWith("/api/", StringComparison.OrdinalIgnoreCase);
                },
                config =>
                {
                    config.Run(async context =>
                    {
                        context.Response.StatusCode =
(int)HttpStatusCode.Unauthorized;
                        await Task.FromResult(string.Empty);
                    });
                }
            );

            app.UseAuthorization();

            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute(
                    name: "default",
                    pattern: "{controller}/{action=Index}/{id?}");
            });

            //Require SAML 2.0 authorization before SPA load.
            app.Use(async (context, next) =>
            {
                if (!context.User.Identity.IsAuthenticated)
                {
                    await context.ChallengeAsync(Saml2Constants.AuthenticationScheme);
                }
                else
                {
                    await next();
                }
            });

            app.UseSpa(spa =>
            {
                // To learn more about options for serving an Angular SPA from ASP.NET
Core,
                // see https://go.microsoft.com/fwlink/?linkid=864501

                spa.Options.SourcePath = "ClientApp";

                if (env.IsDevelopment())
                {
                    spa.UseAngularCliServer(npmScript: "start");
                }
            });


        }
    }
}
```

Configuration

appSettings.json

```
{
  "ConnectionStrings": {
    "DBConnection": "Data Source=(localdb)\\MSSQLLocalDB;Initial
Catalog=UniwaThesisManagementDb;Integrated Security=True"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*",
  "Saml2": {
    // Okta IDP
    "IdPMetadata": "https://dev-
87237178.okta.com/app/exk1i8nguzKVB3PjB5d7/sso/saml/metadata",
    "Issuer": "SAML2Demo",
    "SignatureAlgorithm": "http://www.w3.org/2001/04/xmldsig-more#rsa-sha256",
    // Use only with a valid certificate
    //"SigningCertificateFile": "certificate.pfx",
    //"SigningCertificatePassword": "******",
    //"SignatureValidationCertificateFile": "certificate.cer",
    "CertificateValidationMode": "ChainTrust",
    "RevocationMode": "NoCheck"
  }
}
```

launchSettings.json

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:61484",
      "sslPort": 44362
    }
  },
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "ThesisManagement": {
      "commandName": "Project",
      "launchBrowser": true,
      "applicationUrl": "https://localhost:44362;http://localhost:5000",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

Βιβλιογραφία & Πηγές

[1] Η Σύγκλητος του Πανεπιστημίου Δυτικής Αττικής, «Εσωτερικός κανονισμός λειτουργίας του Πανεπιστημίου Δυτικής Αττικής,» 21 Οκτωβρίου 2020. [Ηλεκτρονικό]. Available: https://ecec.uniwa.gr/foititika-2/esoterikos-kanonismos-leitoyrgias-pada/. [Πρόσβαση 1 Σεπτέμβριου 2023].

[2] Wikipedia, «C_Sharp_(programming_language),» 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Πρόσβαση 2 Σεπτεμβρίου 2023].

[3] Microsoft, «A tour of the C# language,» 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. [Πρόσβαση 2 Σεπτέμριος 2023].

[4] Microsoft, «The history of C#,» 9 Μάρτιου 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-version-history. [Πρόσβαση 2023 Σεπτεμβρίου 2023].

[5] ByteHide, «C# in 2023: The MOST POPULAR Programming Language?,» 2022. [Ηλεκτρονικό]. Available: https://www.bytehide.com/blog/c-wants-to-become-the-most-popular-programming-language-in-2022. [Πρόσβαση 2 Σεπτεμβρίου 2023].

[6] Microsoft, «What is .NET Framework?,» Microsoft, 2023. [Ηλεκτρονικό]. Available: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework. [Πρόσβαση 3 Σεπτεμβρίου 2023].

[7] Microsoft, «Overview of .NET Framework,» 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-gb/dotnet/framework/get-started/overview. [Πρόσβαση 3 Σεπτεμβρίου 2023].

[8] Wikipedia, «.NET Framework version history,» 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/.NET_Framework_version_history. [Πρόσβαση 4 Σεπτεμβρίου 2023].

[9] Wikipedia, «.NET,» 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/.NET. [Πρόσβαση 4 Σεπτεμβρίου 2023].

[10] ByteHide, «WPF vs WinForms,» 2023. [Ηλεκτρονικό]. Available: https://www.bytehide.com/blog/wpf-vs-winforms. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[11] M. Rouse, «Windows Workflow Foundation,» 2 Μαρτίου 2015. [Ηλεκτρονικό]. Available: https://www.techopedia.com/definition/16476/windows-workflow-foundation-wf. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[12] Microsoft, «What Is Windows Communication Foundation,» 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[13] Microsoft, «What is .NET Core? Introduction and overview,» 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/dotnet/core/introduction. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[14] Microsoft, «ASP.NET Core Blazor,» 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/aspnet/core/blazor/?view=aspnetcore-7.0. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[15] D. Lyalin, «Introducing the .NET Hot Reload experience for editing code at runtime,» 25 Μαίου 2021. [Ηλεκτρονικό]. Available: https://devblogs.microsoft.com/dotnet/introducing-net-hot-reload/. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[16] GraffersID, «Advantages and Disadvantages of Using .NET in 2023,» 2023. [Ηλεκτρονικό]. Available: https://graffersid.com/advantages-and-disadvantages-of-using-net/. [Πρόσβαση 5 Σεπτεμβρίου 2023].

[17] IBM, «What is a relational database?,» 2022. [Ηλεκτρονικό]. Available: https://www.ibm.com/topics/relational-databases. [Πρόσβαση 6 Σεπτεμβρίου 2023].

[18] J. D. Ullman, Βασικές Αρχές για τα Συστήματα Βάσεων Δεδομένων, Αθήνα: Εκδόσεις Κλειδάριθμος, 2008.

[19] Wikipedia, «SQL,» 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/SQL. [Πρόσβαση 6 Σεπτεμβρίου 2023].

[20] Daivi, «7 Real-World Applications of SQL Across Industries,» 23 Αυγούστου 2023. [Ηλεκτρονικό]. Available: https://www.projectpro.io/article/applications-of-sql/834. [Πρόσβαση 7 Σεπτεμβρίου 2023].

[21] W3C, «Web Services Architecture,» 11 Φεβρουαρίου 2004. [Ηλεκτρονικό]. Available: https://www.w3.org/TR/ws-arch/#id2260892. [Πρόσβαση 8 Σεπτεμβρίου 2023].

[22] IBM, «What is an API?,» 2022. [Ηλεκτρονικό]. Available: https://www.ibm.com/topics/api. [Πρόσβαση 8 Σεπτεμβρίου 2023].

[23] K. Arellano, «API vs Web Service: What's the Difference?,» 20 Απριλίου 2021. [Ηλεκτρονικό]. Available: https://rapidapi.com/blog/api-vs-web-service/. [Πρόσβαση 8 Σεπτεμβρίου 2023].

[24] A. Walker, «API vs Web Service – Difference Between Them,» 4 Οκτωβρίου 2022. [Ηλεκτρονικό]. Available: https://www.guru99.com/api-vs-web-service-difference.html. [Πρόσβαση 9 Σεπτεμβρίου 2023].

[25] Microsoft, «Visual Studio 2022 IDE - Programming Tool for Software Developers,» 2023. [Ηλεκτρονικό]. Available: https://visualstudio.microsoft.com/vs/. [Πρόσβαση 10 Σεπτεμβρίου 2023].

[26] Microsoft, «What is SQL Server Management Studio (SSMS)?,» 31 Μαρτίου 2023. [Ηλεκτρονικό]. Available: https://learn.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver16. [Πρόσβαση 10 Σεπτεμβρίου 2023].

[27] S. Chacon και B. Straub, Pro Git, New York City: Apress, 2023.

[28] Wikipedia, «GitHub,» 2023. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/GitHub. [Πρόσβαση 12 Σεπτεμβρίου 2023].

[29] EntityFrameworkTutorial, «Entity Framework Core,» 2020. [Ηλεκτρονικό]. Available: https://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx. [Πρόσβαση 14 Σεπτεμβρίου 2023].

[30] SmartBear Software, «Swagger Documentation,» 2023. [Ηλεκτρονικό]. Available: https://swagger.io/docs/. [Πρόσβαση 15 Σεπτεμβρίου 2023].

[31] CodeAcademy Team, «What is REST?,» 4 Απριλίου 2018. [Ηλεκτρονικό]. Available: https://www.codecademy.com/article/what-is-rest. [Πρόσβαση 16 Σεπτεμβρίου 2023].

[32] M. Spasojevic και V. Pecanac, ULTIMATE ASP.NET CORE WEB API, Novi Sad: CodeMaze, 2023.

[33] OASIS, «Security Assertion Markup Language (SAML) V2.0 Technical Overview,» 25 Μαρτίου 2008. [Ηλεκτρονικό]. Available: http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html. [Πρόσβαση 20 Σεπτεμβρίου 2023].

[34] I. Kakavas, «SAML Authentication and the Elastic Stack,» 20 Μαίου 2021. [Ηλεκτρονικό]. Available: https://www.elastic.co/blog/how-to-enable-saml-authentication-in-kibana-and-elasticsearch. [Πρόσβαση 20 Σεπτεμβρίου 2023].

[35] Okta, «SAML app integrations,» 2023. [Ηλεκτρονικό]. Available: https://help.okta.com/en-us/content/topics/apps/apps-about-saml.htm. [Πρόσβαση 22 Σεπτεμβρίου 2023].

[36] N. Gamb, «How to Authenticate with SAML in ASP.NET Core and C#,» 23 Οκτωβρίου 2020. [Ηλεκτρονικό]. Available: https://developer.okta.com/blog/2020/10/23/how-to-authenticate-with-saml-in-aspnet-core-and-csharp. [Πρόσβαση 22 Σεπτεμβρίου 2023].