TECHNICAL MANUAL

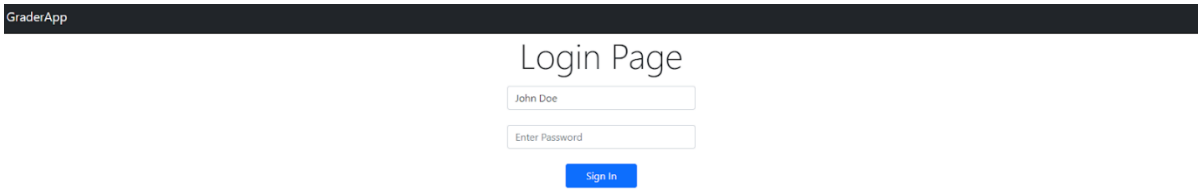
Πίνακας Περιεχομένων

Εισαγωγή

Η εφαρμογή που δημιουργήσαμε είναι μία διαδικτυακή εφαρμογή που υλοποιεί διάφορες λειτουργίες για τους φοιτητές, τους καθηγητές και το γραμματειακό προσωπικό ενός πανεπιστημίου. Χρησιμοποιήσαμε τη Visual C# (ASP.NET Core Web Application) σε συνδυασμό με την αρχιτεκτονική σχεδίασης Model View Controller στο περιβάλλον ανάπτυξης Visual Studio 2022. Επίσης το άλλο εργαλείο που χρησιμοποιήσαμε είναι το Microsoft SQL Server 2022 μαζί με το SQL Server Management Studio 19 για την δημιουργία και επεξεργασία της βάσης δεδομένων που χρειάζεται. Η εφαρμογή μας υλοποιεί όλες τις λειτουργίες που ζητούνται για τους φοιτητές, τους καθηγητές και τις γραμματείες.

Login Page

Αρχικά, ο χρήστης της εφαρμογής συνδέεται στην σελίδα Login μέσω του UserController του μοντέλου MVC. Συγκεκριμένα, με το που ξεκινάει την εφαρμογή του βαθμολογίου κατευθύνεται στην Index σελίδα του Home δηλαδή την Login σελίδα μας. Εφόσον συμπληρώσει την φόρμα με το Username και το Password τότε πατάει το κουμπί Sign In το οποίο τον κάνει redirect στο action User Check του UserController. Εκεί, ο UserController παίρνει ως ορίσματα τα δεδομένα που του έδωσε ο χρήστης και μέσα από το μοντέλο User παίρνει το context.Users, δηλαδή όλους του users που υπάρχουν μέσα στην βάση και ελέγχει αν τα στοιχεία που μας έδωσε ο χρήστης ταυτίζονται με κάποιον από αυτούς. Αν ταυτίζονται τότε ελέγχουμε τον ρόλο του συγκεκριμένου χρήστη, δηλαδή αν είναι “Student”, “Professor” ή “Secretary”, και τον κατευθύνουμε στην αντίστοιχη Index σελίδα του αντίστοιχου View, κρατώντας στα RouteData.Values το username του ώστε να μπορέσουμε να το χρησιμοποιήσουμε και στα άλλα Views. Στην περίπτωση που δεν ταυτίζονται τότε επιστρέφουμε ένα μήνυμα σφάλματος στον χρήστη ότι τα δεδομένα που μας έχει δώσει είναι λανθασμένα και ότι πρέπει να ξαναπροσπαθήσει.

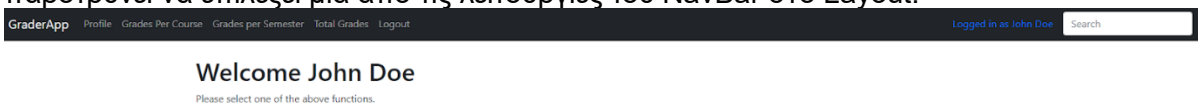


Φοιτητές

Στην περίπτωση που ο χρήστης που συνδέθηκε στην εφαρμογή μας είναι καθηγητής ανακατευθύνεται στην index σελίδα του Professor Controller όπου τον καλωσορίζουμε και του αναφέρουμε να επιλέξει μία από τις παραπάνω λειτουργίες οι οποίες βρίσκονται στο navigation bar και είναι οι εξής: **Profile, Grades Per Course, Grades Per Semester, Total Grades, Logout.**

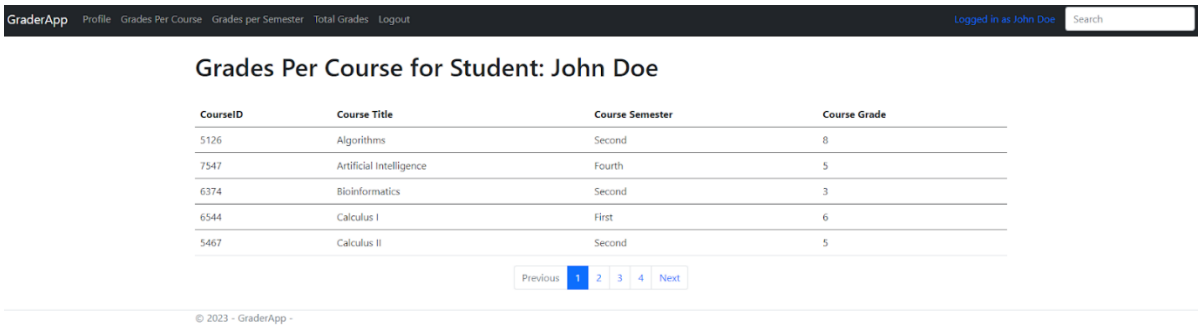
Profile

Profile είναι η Index σελίδα του StudentController που απλά καλωσορίζει το φοιτητή και τον παροτρύνει να επιλέξει μία από τις λειτουργίες του NavBar στο Layout.



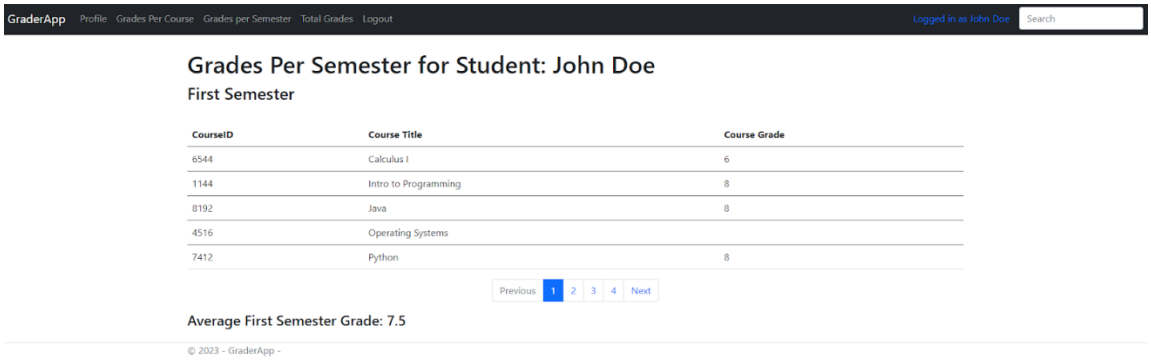
Grades Per Course

Σε αυτή τη σελίδα ο συνδεδεμένος φοιτητής μπορεί να δει την βαθμολογία όλων των μαθημάτων στα οποία έχει γραφτεί. Αυτό γίνεται μέσω του StudentsController και της συνάρτησης GradesPerCourse, στην οποία παίρνουμε από το μοντέλο Students όλους τους φοιτητές και συγκεκριμένα τα πεδία RegistrationNumber και UsersUsername. Στη συνέχεια, επειδή έχουμε στείλει από την Login σελίδα το username του φοιτητή ελέγχουμε μέσω ενός βρόχου το username που έχουμε σε ποιον Student από το μοντέλο αντιστοιχεί και αποθηκεύουμε το RegistrationNumber του. Εφόσον, θέλουμε να εμφανίσουμε τα μαθήματα παίρνουμε από τα μοντέλα Course και CourseHasStudent τα αντίστοιχα πεδία από την βάση που χρειαζόμαστε. Επειδή θέλουμε να εμφανίσουμε στο View δεδομένα από διαφορετικά μοντέλα, δημιουργούμε ένα καινούριο μοντέλο το StudentCourseView που συμπεριλαμβάνει τα πεδία των άλλων μοντέλων που θέλουμε να χρησιμοποιήσουμε. Μέσω ενός βρόχου πάλι ελέγχουμε για τα αντικείμενα CourseHasStudent της βάσης αυτά που συμπίπτουν με το RegistrationNumber του φοιτητή μας και έπειτα ελέγχουμε από αυτά ποια συμπίπτουν με το CourseId των αντικειμένων Course της βάσης και δημιουργούμε μία λίστα τύπου StudentCourseView βάζοντας μέσα όλα τα πεδία αντικειμένων που προκύπτουν μετά τις συγκρίσεις.
Επιπλέον, επειδή θέλουμε να προσθέσουμε και σελιδοποίηση και μπάρα αναζήτησης στη σελίδα έχουμε προσθέσει μερικές ακόμα αλλαγές. Παίρνουμε το περιεχόμενο της μπάρας αναζήτησης από το Layout μέσω του ViewData και μέσω ενός απλού query παίρνουμε από την λίστα StudentCourseView μόνο τα αντικείμενα στα οποία το CourseTitle συμπίπτει με το περιεχόμενο αυτό. Στην συνέχεια, κάνουμε μία ταξινόμηση της λίστας με βάση τον τίτλο και την μετατρέπουμε σε PagedList ώστε να πραγματοποιηθεί η σελιδοποίηση. Τέλος, επιστρέφουμε στο View μας την λίστα αυτή.
Στο View μας τώρα εμφανίζουμε μέσω ενός table αρχικά τα ονόματα των πεδίων που θέλουμε και στη συνέχεια μέσω ενός βρόχου foreach που εξετάζει όλα τα αντικείμενα στην λίστα που μας επέστρεψε ο StudentController εμφανίζουμε τα αντίστοιχα πεδία που επιθυμούμε. Τα αντικείμενα αυτά είναι ταξινομημένα κατα όνομα του CourseTitle και εμφανίζονται σελιδοποιημένα με βάση τις αντίστοιχες παραμέτρους που έχουμε προσθέσει 5 αντικείμενα ανά σελίδα.



Grades Per Semester

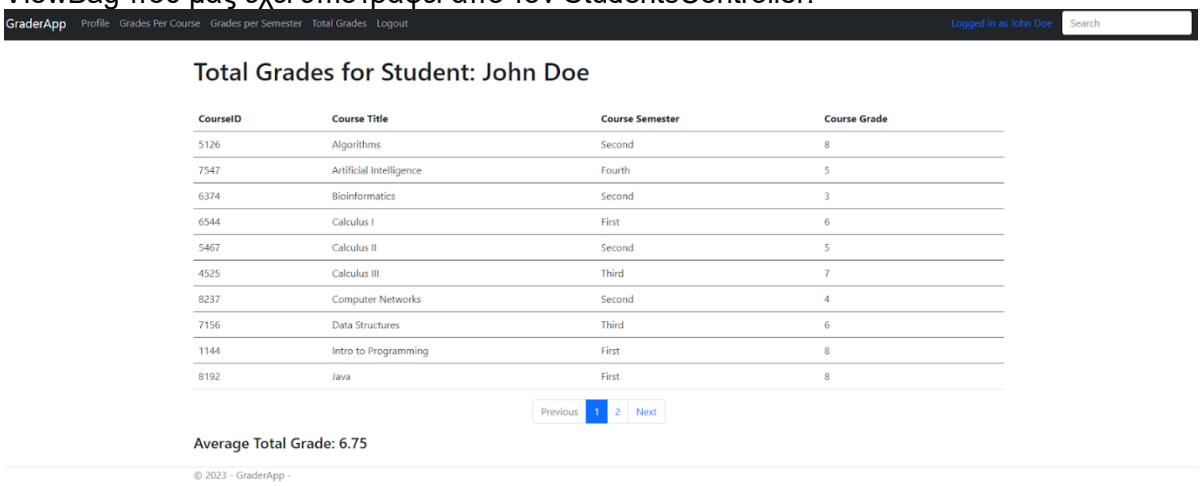
Σε αυτή τη σελίδα ο συνδεδεμένος φοιτητής μπορεί να δει την βαθμολογία όλων των μαθημάτων στα οποία έχει γραφτεί ανα εξάμηνο με βάση την σελιδοποίηση. Αυτό γίνεται μέσω του StudentsController και της συνάρτησης GradesPerSemester, στην οποία ακολουθούμε την ίδια διαδικασία του GradesPerCourse και παίρνουμε από το μοντέλο Students όλους τους φοιτητές και συγκεκριμένα τα πεδία RegistrationNumber και UsersUsername. Στη συνέχεια, επειδή έχουμε στείλει από την Login σελίδα το username του φοιτητή ελέγχουμε μέσω ενός βρόχου το username που έχουμε σε ποιον Student από το μοντέλο αντιστοιχεί και αποθηκεύουμε το RegistrationNumber του. Εφόσον, θέλουμε να εμφανίσουμε τα μαθήματα κάθε εξαμήνου παίρνουμε από τα μοντέλα Course και CourseHasStudent τα αντίστοιχα πεδία από την βάση που χρειαζόμαστε. Επειδή θέλουμε πάλι να εμφανίσουμε στο View δεδομένα από διαφορετικά μοντέλα χρησιμοποιούμε ξανά το μοντέλο StudentCourseView που συμπεριλαμβάνει τα πεδία των άλλων μοντέλων που θέλουμε να χρησιμοποιήσουμε. Μέσω ενός βρόχου ελέγχουμε για τα αντικείμενα CourseHasStudent της βάσης αυτά που συμπίπτουν με το RegistrationNumber του φοιτητή μας και έπειτα ελέγχουμε από αυτά ποια συμπίπτουν με το CourseId των αντικειμένων Course της βάσης και δημιουργούμε μία λίστα τύπου StudentCourseView βάζοντας μέσα όλα τα πεδία αντικειμένων που προκύπτουν μετά τις συγκρίσεις.
Επιπλέον, επειδή θέλουμε να προσθέσουμε και σελιδοποίηση ανα εξάμηνο και μπάρα αναζήτησης στη σελίδα έχουμε προσθέσει μερικές ακόμα αλλαγές. Παίρνουμε το περιεχόμενο της μπάρας αναζήτησης από το Layout μέσω του ViewData και μέσω ενός απλού query παίρνουμε από την λίστα StudentCourseView μόνο τα αντικείμενα στα οποία το CourseTitle συμπίπτει με το περιεχόμενο αυτό. Στην συνέχεια, κάνουμε μία ταξινόμηση της λίστας με βάση τον τίτλο και την μετατρέπουμε σε PagedList ώστε να πραγματοποιηθεί η σελιδοποίηση. Πραγματοποιούμε επίσης έναν έλεγχο για το εξάμηνο στο οποίο βρισκόμαστε ανάλογα με τη σελίδα (π.χ. αν είμαστε στην πρώτη σελίδα εμφανίζουμε τα μαθήματα του πρώτου εξαμήνου) και αποθηκεύουμε το εξάμηνο σε ένα ViewBag. Τέλος, επιστρέφουμε στο View μας την λίστα αυτή.
Στο View μας τώρα εμφανίζουμε μέσω ενός table αρχικά τα ονόματα των πεδίων που θέλουμε και στη συνέχεια μέσω ενός βρόχου foreach που εξετάζει όλα τα αντικείμενα στην λίστα που μας επέστρεψε ο StudentController εμφανίζουμε τα αντίστοιχα πεδία που συμπίπτουν με το εξάμηνο που αναγράφεται στην αντίστοιχη σελίδα συγκρίνοντας με το ViewBag που μας έχει επιστραφεί.
Στον βρόχο επίσης έχουμε ένα μετρητή count και ένα σύνολο sum για να εμφανίζουμε στον φοιτητή τον μέσο όρο του κάθε εξαμήνου. Τα αντικείμενα αυτά είναι ταξινομημένα κατα όνομα του CourseTitle και εμφανίζονται σελιδοποιημένα με βάση τις αντίστοιχες παραμέτρους που έχουμε προσθέσει και εφόσον έχουμε τέσσερα διαφορετικά εξάμηνα στην βάση τα εμφανίζουμε ανα σελίδα του PagedListPager.



Total Grades

Σε αυτή τη σελίδα ο συνδεδεμένος φοιτητής μπορεί να δει την συνολική βαθμολογία όλων των μαθημάτων στα οποία έχει δώσει εξετάσεις και έχει λάβει βαθμό. Αυτό γίνεται μέσω του StudentsController και της συνάρτησης TotalGrades, στην οποία ακολουθούμε την ίδια διαδικασία του GradesPerCourse και παίρνουμε από το μοντέλο Students όλους τους φοιτητές και συγκεκριμένα τα πεδία RegistrationNumber και UsersUsername. Στη συνέχεια, επειδή έχουμε στείλει από την Login σελίδα το username του φοιτητή ελέγχουμε μέσω ενός βρόχου το username που έχουμε σε ποιον Student από το μοντέλο αντιστοιχεί και αποθηκεύουμε το RegistrationNumber του. Εφόσον, θέλουμε να εμφανίσουμε όλα τα μαθήματα που έχει δώσει εξετάσεις παίρνουμε από τα μοντέλα Course και CourseHasStudent τα αντίστοιχα πεδία από την βάση που χρειαζόμαστε. Επειδή θέλουμε πάλι να εμφανίσουμε στο View δεδομένα από διαφορετικά μοντέλα χρησιμοποιούμε ξανά το μοντέλο StudentCourseView που συμπεριλαμβάνει τα πεδία των άλλων μοντέλων που θέλουμε να χρησιμοποιήσουμε. Μέσω ενός βρόχου ελέγχουμε για τα αντικείμενα CourseHasStudent της βάσης αυτά που συμπίπτουν με το RegistrationNumber του φοιτητή μας και έπειτα ελέγχουμε από αυτά ποια συμπίπτουν με το CourseId των αντικειμένων Course της βάσης και ποια έχουν βαθμό διαφορετικό του NULL και δημιουργούμε μία λίστα τύπου StudentCourseView βάζοντας μέσα όλα τα πεδία αντικειμένων που προκύπτουν μετά τις συγκρίσεις.
Επιπλέον, επειδή θέλουμε να προσθέσουμε και σελιδοποίηση ανα εξάμηνο και μπάρα αναζήτησης στη σελίδα έχουμε προσθέσει μερικές ακόμα αλλαγές. Παίρνουμε το περιεχόμενο της μπάρας αναζήτησης από το Layout μέσω του ViewData και μέσω ενός απλού query παίρνουμε από την λίστα StudentCourseView μόνο τα αντικείμενα στα οποία το CourseTitle συμπίπτει με το περιεχόμενο αυτό. Στην συνέχεια, κάνουμε μία ταξινόμηση της λίστας με βάση τον τίτλο και την μετατρέπουμε σε PagedList ώστε να πραγματοποιηθεί η σελιδοποίηση. Θέλουμε να εμφανίσουμε ξανά τον μέσο όρο της συνολικής βαθμολογίας του φοιτητή οπότε πάλι μέσω ενός count και sum στον StudentController αυτή την φόρα, (επειδή θέλουμε να συμπεριλάβουμε όλες τις βαθμολογίες, όχι μόνο ανα σελίδα) βρίσκουμε το average και το αποθηκεύουμε σε ένα Viewbag. Τέλος, επιστρέφουμε στο View μας την λίστα τύπου StudentsCourseView.
Στο View μας τώρα εμφανίζουμε μέσω ενός table αρχικά τα ονόματα των πεδίων που θέλουμε και στη συνέχεια μέσω ενός βρόχου foreach που εξετάζει όλα τα αντικείμενα στην λίστα που μας επέστρεψε ο StudentController εμφανίζουμε τα αντίστοιχα πεδία που επιθυμούμε. Τα αντικείμενα αυτά είναι ταξινομημένα κατα όνομα του CourseTitle και εμφανίζονται σελιδοποιημένα με βάση τις αντίστοιχες παραμέτρους που έχουμε προσθέσει

10 αντικείμενα ανά σελίδα. Τέλος, εμφανίζουμε και τον συνολικό μέσο όρο μέσω του ViewBag που μάς έχει επιστραφεί από τον StudentsController.



Logout

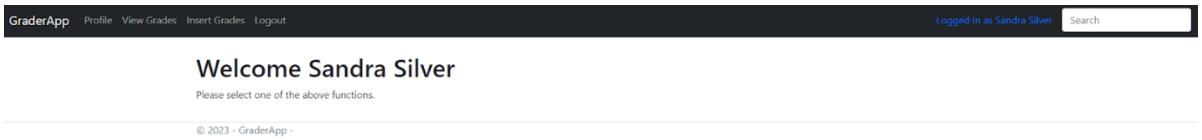
Ανακατεύθυνση στην αρχική σελίδα της εφαρμογής (Login Page).

Καθηγητές

Στην περίπτωση που ο χρήστης που συνδέθηκε στην εφαρμογή μας είναι καθηγητής ανακατευθύνεται στην index σελίδα του Professors Controller όπου τον καλωσορίζουμε και του αναφέρουμε να επιλέξει μία από τις παραπάνω λειτουργίες οι οποίες βρίσκονται στο navigation bar και είναι οι εξής: **Profile, View Grades, Insert Grades, Logout**.

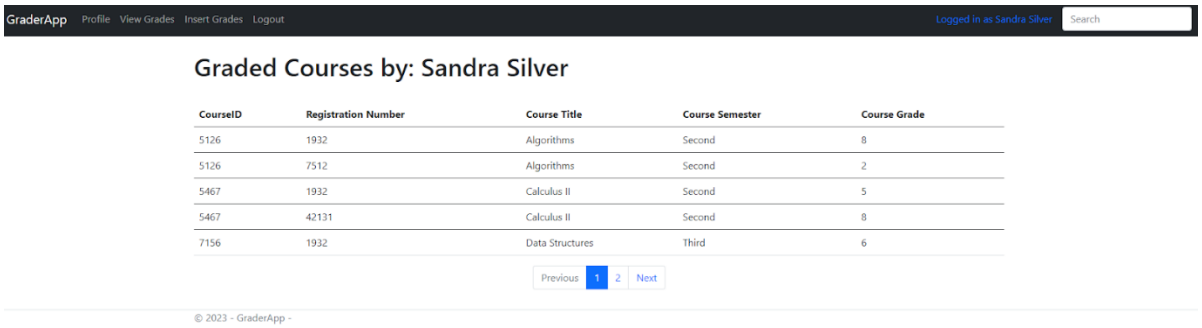
Profile

Index σελίδα του Professors Controller.



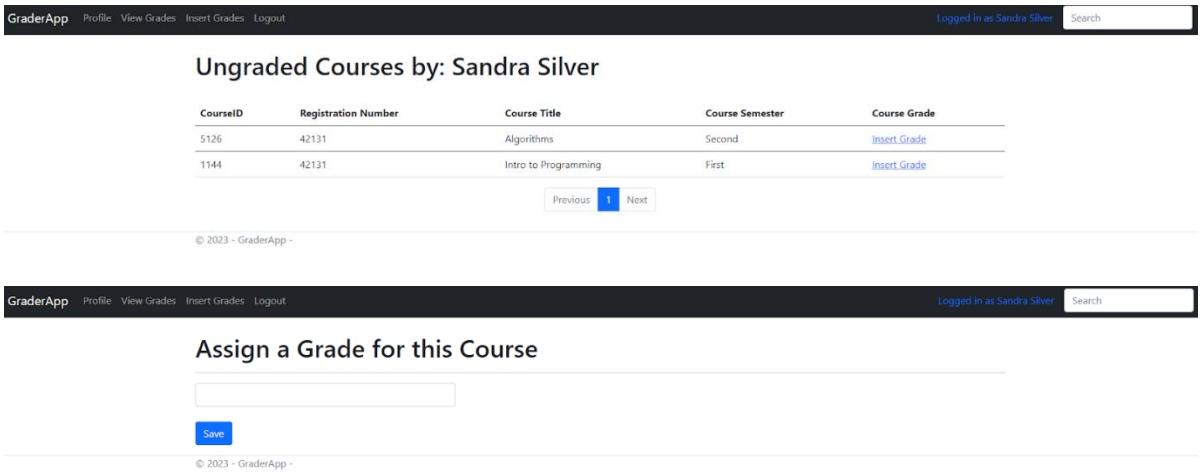
View Grades

Σε αυτή τη σελίδα ο συνδεδεμένος καθηγητής μπορεί να δει τους βαθμούς που έχει βάλει στα μαθήματα. Με το κλικ πάνω σε αυτήν την λειτουργία δημιουργούνται αντικείμενα τύπου ProfessorCourseView στην συνάρτηση ViewGrades() τα οποία αποθηκεύονται σε μία λίστα και περιέχουν το id του κάθε μαθήματος, τον αριθμό μητρώου και το βαθμό του φοιτητή που έχει πάρει στο μάθημα, τον τίτλο και το εξάμηνο του μαθήματος. Αυτά τα δεδομένα τα παίρνει η συνάρτηση μέσω του _context, δηλαδή μέσω της βάσης δεδομένων. Η τελική λίστα που δημιουργείται που είναι τύπου ProfessorCourseView μετατρέπεται σε PagedList (ώστε να εμφανίζονται οι εγγραφές με σελιδοποίηση) και μεταφέρεται στο αντίστοιχο View ViewGrades.cshtml. Επίσης η συνάρτηση αυτή υλοποιεί και την λειτουργία της αναζήτησης μέσω του τίτλου του μαθήματος. Στο ViewGrades.cshtml εμφανίζονται όλα τα μαθήματα που έχει βαθμολογήσει ο καθηγητής μαζί με το id, τον τίτλο και το εξάμηνο του κάθε course καθώς και τον αριθμό μητρώου του αντίστοιχου φοιτητή.



Insert Grades

Σε αυτή τη σελίδα ο καθηγητής μπορεί να δει όλα τα μαθήματα που δεν έχει βαθμολογήσει, δηλαδή τα μαθήματα στην βάση που έχουν NULL τιμή στο πεδίο του βαθμού. Η συνάρτηση που καλείται είναι η InsertGrades() που βρίσκεται στον Professor Controller και με παρόμοιο τρόπο όπως στην προηγούμενη συνάρτηση αντλεί τα δεδομένα από την βάση και τα τοποθετεί σε μία λίστα τύπου ProfessorCourseView. Τοποθετεί τις εγγραφές που δεν έχουν βαθμό (το πεδίο GradeCourseStudent του course_has_students είναι NULL). Όπως προηγουμένως επιστρέφει στο View (InsertGrades.cshtml) μία λίστα τύπου PagedList η οποία εμφανίζει στον χρήστη το id, τον τίτλο, το εξάμηνο του μαθήματος και τον αριθμό μητρώου του φοιτητή που δεν έχει εξεταστεί. Στο τελευταίο πεδίο του δίνει τη δυνατότητα να εισαγάγει βαθμό στο αντίστοιχο μάθημα με το κλικ πάνω στο “Insert Grade”. Με το πάτημα αυτό ανακατευθύνεται στον Controller του CourseHasStudents στην σελίδα Edit. Εκεί μέσω του ViewData[] μεταφέρονται τα απαραίτητα δεδομένα ώστε να μπορέσει να βαθμολογήσει ο συνδεδεμένος καθηγητής το αντίστοιχο μάθημα. Εφόσον επιλέξει το βαθμό και πατήσει Save, ο βαθμός αποθηκεύεται στη βάση και ανακατευθύνεται στη προηγούμενη σελίδα (Insert Grades).



Logout

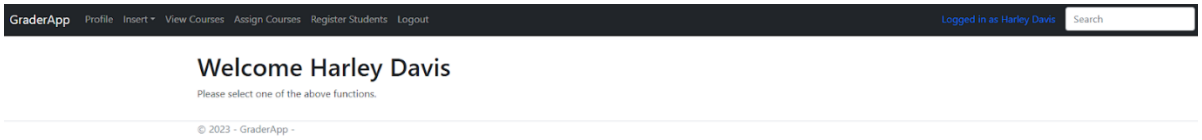
Ανακατεύθυνση στην αρχική σελίδα της εφαρμογής (Login Page).

Γραμματείες

Στην περίπτωση που ο χρήστης που συνδέθηκε στην εφαρμογή μας ανήκει στο γραμματειακό προσωπικό ανακατευθύνεται στην index σελίδα του Secretaries Controller όπου τον καλωσορίζουμε και του αναφέρουμε να επιλέξει μία από τις παραπάνω λειτουργίες οι οποίες βρίσκονται στο navigation bar και είναι οι εξής: **Profile, Insert, View Courses, Assign Courses, Register Students, Logout**.

Profile

Index σελίδα του Secretaries Controller.

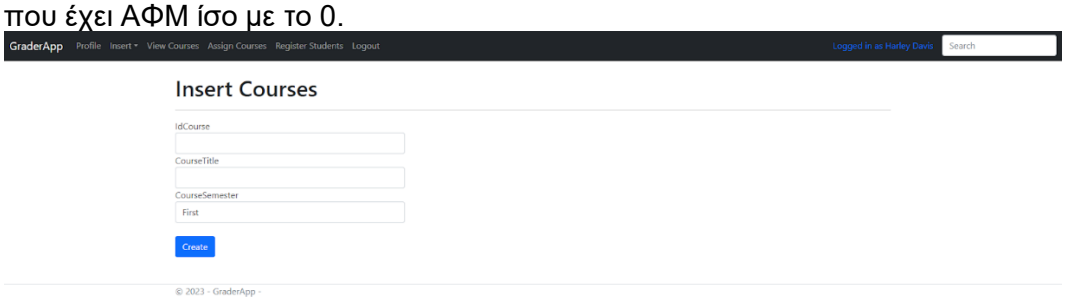


Insert

Με το κλικ πάνω στο Insert εμφανίζονται στο συνδεδεμένο χρήστη 3 επιλογές: Courses, Professors, Students

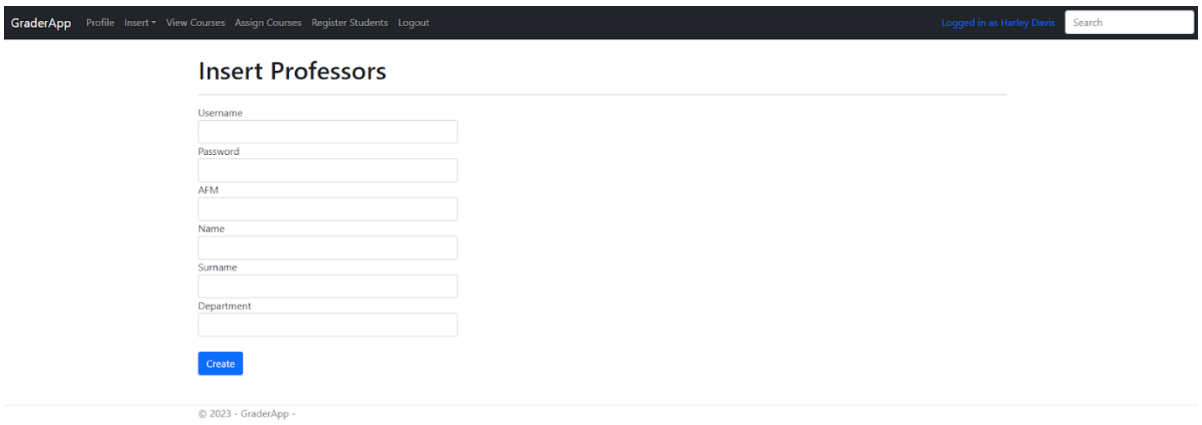
Courses

Σε αυτή τη σελίδα ο χρήστης μπορεί να εισαγάγει καινούργια μαθήματα στη βάση δεδομένων τοποθετώντας στα πεδία IdCourse, CourseTitle, CourseSemester τις αντίστοιχες τιμές. Η συνάρτηση που καλείται είναι η InsertCourses() που βρίσκεται στον Secretaries Controller και εμφανίζει στο χρήστη το View InsertCourses.cshtml. Εκεί ο χρήστης εφόσον συμπληρώσει τις αντίστοιχες τιμές και πατήσει το κουμπί “Create” πραγματοποιείται έλεγχος μέσω της συνάρτησης InsertCourses([Bind("IdCourse,CourseTitle,CourseSemester")] Course course) για την επιτυχία ή αποτυχία της εισαγωγής του μαθήματος. Αν αποτύχει (επειδή ο χρήστης έβαλε idCourse που ήδη υπάρχει στη βάση για κάποιο μάθημα) του εμφανίζει στο κάτω μέρος An “Error has occurred!”. Διαφορετικά του εμφανίζει μήνυμα επιτυχίας “Course created successfully!”. Η παραπάνω λειτουργία υλοποιήθηκε με τη χρήση ενός Professor στην βάση ο οποίος έχει AFM = 0. Τα μαθήματα που δημιουργούνται μέσω αυτής της λειτουργίας αναθέτονται αρχικά σε αυτόν και ύστερα με τη λειτουργία ανάθεσης μαθήματος σε καθηγητή, το γραμματειακό προσωπικό μπορεί να επιλέξει σε ποιον καθηγητή ανήκει το κάθε μάθημα. Στον κώδικά μας, δηλαδή γίνονται οι απαραίτητοι έλεγχοι, ώστε να εμφανίζονται στους χρήστες της εφαρμογής μόνο οι “πραγματικοί” καθηγητές και όχι αυτός που έχει ΑΦΜ ίσο με το 0.



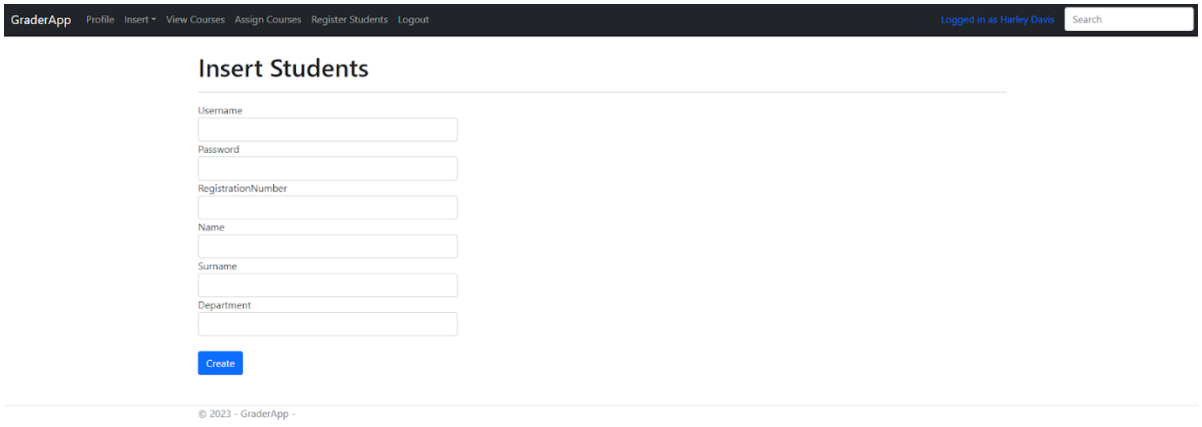
Professors

Σε αυτή τη σελίδα ο χρήστης μπορεί να καθηγητές στη βάση δεδομένων τοποθετώντας στα πεδία Username, Password, AFM, Name, Surname, Department τις αντίστοιχες τιμές. Η συνάρτηση που καλείται είναι η InsertProfessors() που βρίσκεται στον Secretaries Controller και εμφανίζει στο χρήστη το View InsertProfessors.cshtml. Εκεί ο χρήστης εφόσον συμπληρώσει τις αντίστοιχες τιμές και πατήσει το κουμπί “Create” πραγματοποιείται έλεγχος μέσω της συνάρτησης InsertProfessors([Bind("Username,Password,AFM,Name,Surname,Department")] ProfessorUser professorUser) για την επιτυχία ή αποτυχία της εισαγωγής του καθηγητή. Αν αποτύχει του εμφανίζει στο κάτω μέρος An “Error has occurred!”. Διαφορετικά του εμφανίζει μήνυμα επιτυχίας “Professor created successfully!”. Η λειτουργία αυτή αρχικά φτιάχνει έναν User και ύστερα έναν Professor, διότι χρειάζεται ο Professor να έχει ένα USERS_username(ξένο κλειδί) ώστε να εισαχθεί στη βάση δεδομένων.



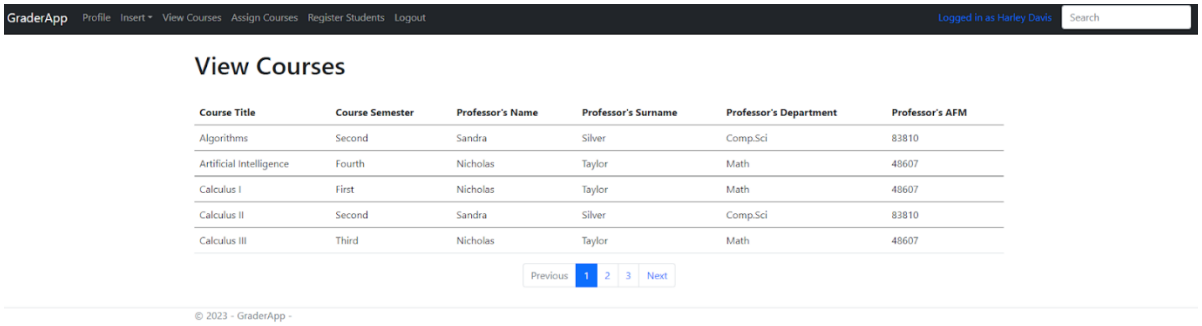
Students

Σε αυτή τη σελίδα ο χρήστης μπορεί να φοιτητές στη βάση δεδομένων τοποθετώντας στα πεδία Username, Password, RegistrationNumber, Name, Surname, Department τις αντίστοιχες τιμές. Η συνάρτηση που καλείται είναι η InserStudents() που βρίσκεται στον Secretaries Controller και εμφανίζει στο χρήστη το View InsertStudents.cshtml. Εκεί ο χρήστης εφόσον συμπληρώσει τις αντίστοιχες τιμές και πατήσει το κουμπί “Create” πραγματοποιείται έλεγχος μέσω της συνάρτησης InsertStudents([Bind("Username,Password,RegistrationNumber,Name,Surname,Department")] StudentUser studentUser) για την επιτυχία ή αποτυχία της εισαγωγής του φοιτητή. Αν αποτύχει του εμφανίζει στο κάτω μέρος An “Error has occurred!”. Διαφορετικά του εμφανίζει μήνυμα επιτυχίας “Student created successfully!”. Η λειτουργία αυτή αρχικά φτιάχνει έναν User και ύστερα έναν Student, διότι χρειάζεται ο Student να έχει ένα USERS_username(ξένο κλειδί) ώστε να εισαχθεί στη βάση δεδομένων.



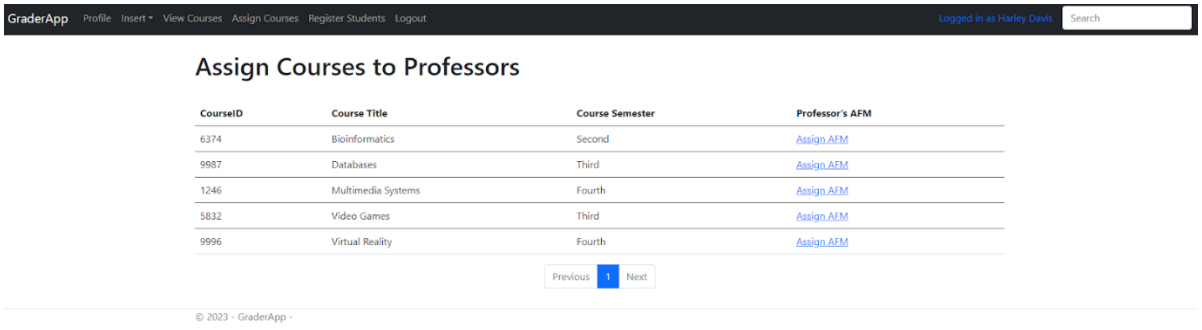
View Courses

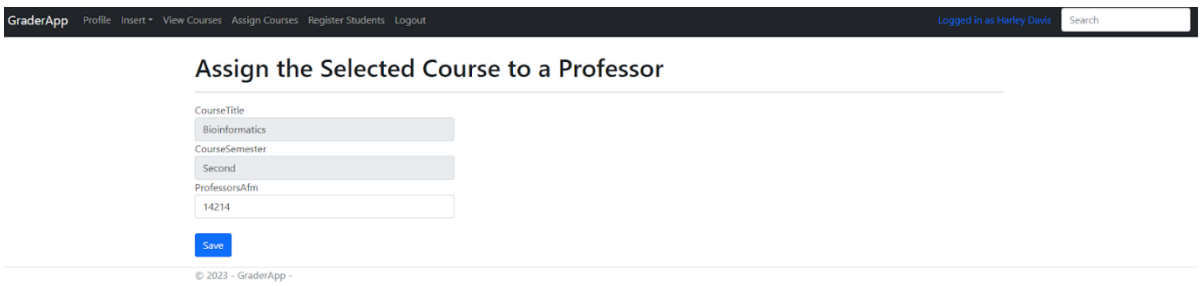
Σε αυτήν τη σελίδα ο γραμματέας μπορεί να δει όλα τα διαθέσιμα μαθήματα στην βάση και τους αντίστοιχους καθηγητές οι οποίοι τα έχουν αναλάβει. Με το κλικ πάνω σε αυτήν την λειτουργία δημιουργούνται αντικείμενα τύπου SecretaryCourseView στην συνάρτηση ViewCourses() τα οποία αποθηκεύονται σε μία λίστα και περιέχουν τον τίτλο και το εξάμηνο του κάθε μαθήματος, το όνομα, το επώνυμο, το τμήμα και το ΑΦΜ του καθηγητή που έχει αναλάβει στο μάθημα.. Αυτά τα δεδομένα τα παίρνει η συνάρτηση μέσω του _context, δηλαδή μέσω της βάσης δεδομένων. Η τελική λίστα που δημιουργείται που είναι τύπου SecretaryCourseView μετατρέπεται σε PagedList (ώστε να εμφανίζονται οι εγγραφές με σελιδοποίηση) και μεταφέρεται στο αντίστοιχο View ViewCourses.cshtml. Επίσης η συνάρτηση αυτή υλοποιεί και την λειτουργία της αναζήτησης μέσω του τίτλου του μαθήματος, του ονόματος και του επώνυμου του καθηγητή. Στο ViewCourses.cshtml εμφανίζονται όλα τα διαθέσιμα μαθήματα μαζί με τον τίτλο, το εξάμηνο του κάθε course καθώς και το όνομα, επώνυμο, τμήμα και ΑΦΜ του αντίστοιχου καθηγητή.



Assign Courses

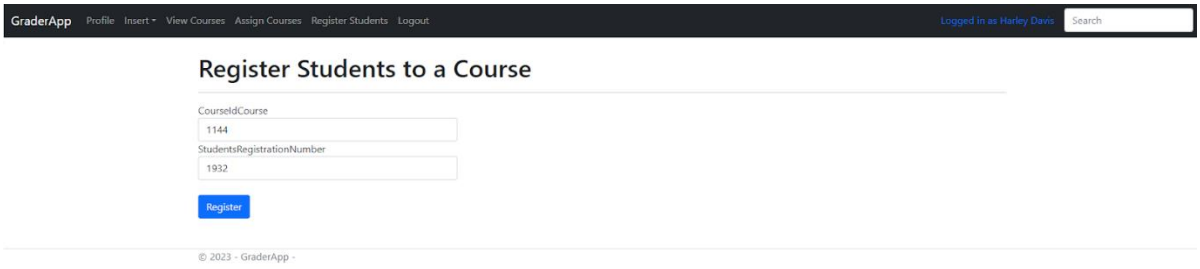
Σε αυτή τη σελίδα ο γραμματέας μπορεί να δει όλα τα μαθήματα που δεν έχουν ανατεθεί σε καθηγητή, δηλαδή τα μαθήματα στην βάση που έχουν 0 τιμή στο πεδίο του ProfessorsAFM. Όπως αναφέραμε και προηγουμένως στον κώδικά μας γίνονται οι απαραίτητοι έλεγχοι, ώστε να εμφανίζονται στους χρήστες της εφαρμογής μόνο οι “πραγματικοί” καθηγητές και όχι αυτός που έχει ΑΦΜ ίσο με το 0. Η συνάρτηση που καλείται είναι η AssignCourses() που βρίσκεται στον Secretary Controller και με παρόμοιο τρόπο όπως στην προηγούμενη συνάρτηση αντλεί τα δεδομένα από την βάση και τα τοποθετεί σε μία λίστα τύπου Course. Τοποθετεί τις εγγραφές που δεν έχουν ProfessorsAFM(το πεδίο ProfessorsAFM του course είναι 0). Όπως προηγουμένως επιστρέφει στο View (AssignGrades.cshtml) μία λίστα τύπου PagedList η οποία εμφανίζει στον χρήστη το id, τον τίτλο, το εξάμηνο του μαθήματος και το ΑΦΜ του ψεύτικου καθηγητή. Στο τελευταίο πεδίο του δίνει τη δυνατότητα να εισαγάγει ΑΦΜ Καθηγητή στο αντίστοιχο μάθημα με το κλικ πάνω στο “Assign AFM”. Με το πάτημα αυτό ανακατευθύνεται στον Controller του Course στην σελίδα Edit. Εκεί μέσω του ViewData[] μεταφέρονται τα απαραίτητα δεδομένα ώστε να μπορέσει ο συνδεδεμένος γραμματέας να καταχωρήσει ΑΦΜ κάποιου καθηγητή που βρίσκεται ήδη στην βαση για το αντίστοιχο μάθημα. Εφόσον επιλέξει ΑΦΜ από την λίστα και πατήσει Save, το ΑΦΜ καθηγητή αποθηκεύεται στη βάση και ανακατευθύνεται στη προηγούμενη σελίδα (Assign Courses).





Register Students

Σε αυτή τη σελίδα ο γραμματέας μπορεί να δηλώσει ένα μάθημα σε έναν φοιτητή ο οποίος δεν είναι ήδη εγγεγραμμένος σε αυτό το μάθημα. Η συνάρτηση που καλείται είναι η Create() που βρίσκεται στον CourseHasStudent Controller. Εκεί μέσω του ViewData[] μεταφέρονται τα απαραίτητα δεδομένα ώστε να μπορέσει ο συνδεδεμένος γραμματέας να εγγράψει τον φοιτητή στο αντίστοιχο μάθημα. Συγκεκριμένα, μέσω του _context παίρνουμε από όλα τα courses τα IdCourse και από τους Students όλα τα RegistrationNumber. Αυτά επιστρέφονται στο View μέσω ViewData και ο γραμματέας μπορεί να επιλέξει courseId και registrationNumber από την λίστα και να πατήσει Register, έτσι δημιουργείται καινούριο αντικείμενο CourseHasStudent στην βάση με τα αντίστοιχα πεδία CourseIdCourse και StudentsRegistrationNumber και ένα κενό πεδίο GradeCourseStudent και του επιστρέφουμε το αντίστοιχο μήνυμα. Σημαντικό είναι να αναφέρουμε ότι στον CourseHasStudent Controller ελέγχουμε την περίπτωση που ο φοιτητής είναι ήδη εγγεγραμμένος στο μάθημα στο οποίο επιλέξαμε να τον γράψουμε και επιστρέφουμε το αντίστοιχο μήνυμα σφάλματος στον γραμματέα.



Logout

Ανακατεύθυνση στην αρχική σελίδα της εφαρμογής (Login Page).

Παρατήρηση

Κάποιοι από τους ήδη υπάρχοντες χρήστες (που χρησιμοποιούνται στα παραδείγματα) για την διευκόλυνσή σας στην περιήγηση της εφαρμογής μας είναι οι εξής:

Φοιτητής

Username: John Doe
Password: john123

Καθηγητής

Username: Sandra Silver
Password: sandra

Γραμματεία

Username: Harley Davis
Password: hardav