ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

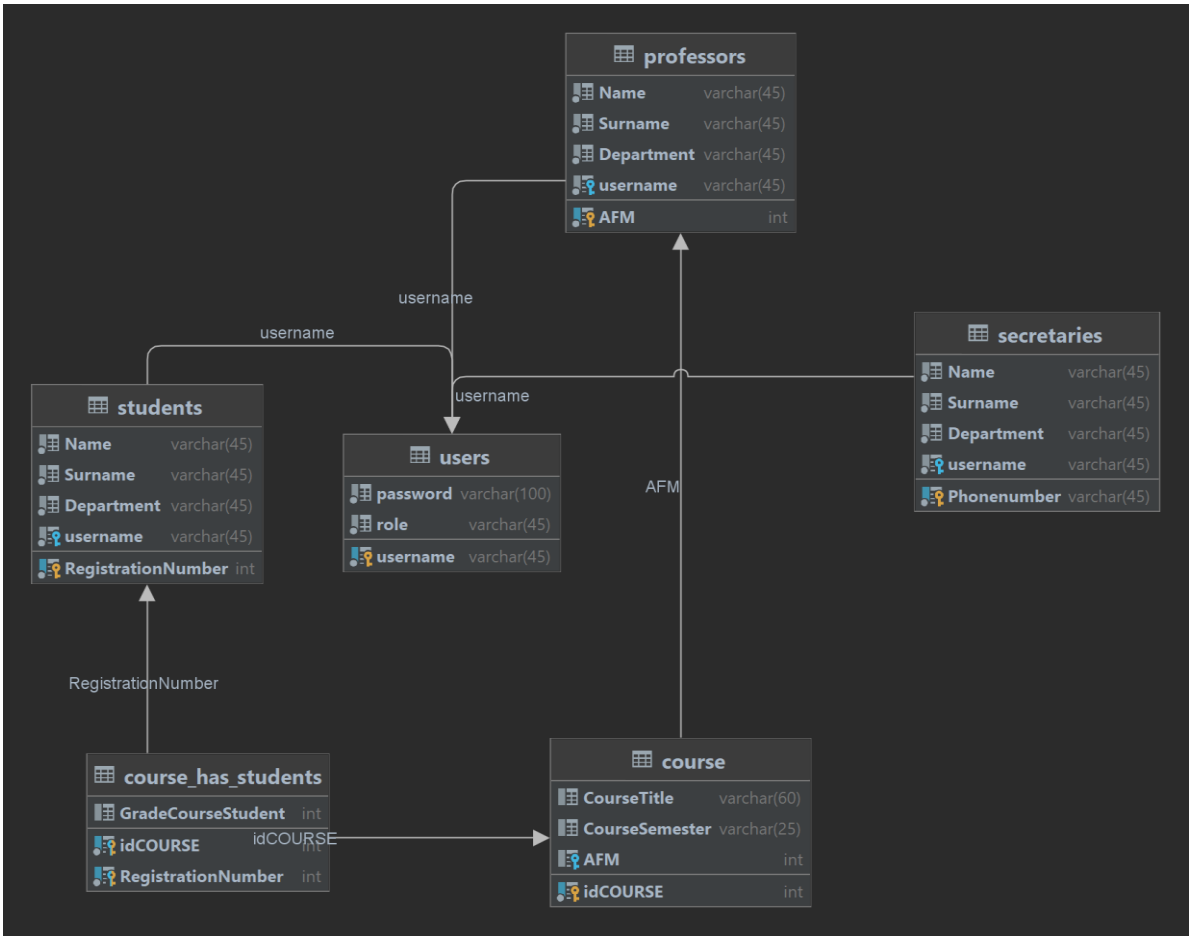
ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ

«ΣΥΓΧΡΟΝΑ ΘΕΜΑΤΑ ΤΕΧΝΟΛΟΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ»

ΠΕΙΡΑΙΑΣ 2023

ΣΤΟΙΧΕΙΑ ΟΜΑΔΑΣ	
ΑΥΓΕΡΙΝΟΣ ΧΡΗΣΤΟΣ	Π19020
ΒΙΤΑΚΗΣ ΑΘΑΝΑΣΙΟΣ	Π19247
ΠΑΝΑΓΙΩΤΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ	Π19130

Διάγραμμα Βάσης Δεδομένων



Πίνακες:

- Users (Χρηστες)
- Students (Φοιτητές)
- Professors(Καθηγητές)
- Secretaries(Γραμματεία)
- Course (Μάθημα)
- Course_has_students (Βαθμοί Μαθητών ανά μάθημα)

Το αντικείμενο της βάσης δηλώνεται από μια κλάση DBContext και χρησιμοποιείται από πολλούς controllers.

Πιο συγκεκριμένα:

1. UsersController

- UsersLogin() :

```
user = _context.Users.FirstOrDefault(user => user.Username.Equals(model.Username)
 && user.Password.Equals(model.Password));
```

Στη μέθοδο αυτή χρησιμοποιείται αναζήτηση στη βάση δεδομένων στο πίνακα Users που περιέχει όλους τους χρήστες με στόχο να βρεθεί και να αποθηκευτεί το πρωτεύον κλειδί του πίνακα που είναι το username με βάση τα στοιχεία που στάλθηκαν από τη φόρμα. Στη συνέχεια , ανάλογα το ρόλο του χρήστη που βρέθηκε γίνεται ανακατεύθυνση στη σελίδα Index που του αντιστοιχεί.

Για κάθε έναν από τους Students,Professors,Secretaries υπάρχει η μέθοδος Index όπου μας μεταβιβάζει στην αρχική σελίδα του κάθε χρήστη. Στη μέθοδο αυτή χρησιμοποιούμε το αποθηκευμένο username που δόθηκε στη UsersLogin από το χρήστη ώστε να κάνουμε αναζήτηση στη βάση, να βρούμε την εγγραφή , να κρατήσουμε το αντίστοιχο αντικείμενο και να εμφανίσουμε τα χαρακτηριστικά του στην εξατομικευμένη σελίδα του κάθε χρήστη.

1. StudentsController

- SelectAll()

```
var allGrades = _context.CourseHasStudents.Include(s => s.Student).Include(s => s.Course).
    Where(s=>s.Student.Username.Equals(username) && s.GradeCourseStudent!=null);
```

Στη μέθοδο αυτή του φοιτητή χρησιμοποιούμε το username που αποθηκεύσαμε απο τη UsersLogin και κάνουμε αναζήτηση στο πίνακα CourseHasStudents σε σύνδεση με τους πίνακες Students, Course ώστε να πάρουμε μια λίστα με όλους του βαθμούς(σε όποιο μάθημα έχει βάθμο) και τα μαθήματα του συγκεκριμένου φοιτητή.

- SelectBySemester()

```
var gradesBySemester = _context.CourseHasStudents.Include(s => s.Student).Include(s => s.Course).
   Where(s => s.Course.CourseSemester.Equals(Selected.ToString()) && s.Student.Username.Equals(username) && s.GradeCourseStudent != null);
```

Στη μέθοδο αυτή του φοιτητή χρησιμοποιούμε το username που αποθηκεύσαμε απο τη UsersLogin και κάνουμε αναζήτηση στο πίνακα CourseHasStudents σε σύνδεση με τους πίνακες Students, Course ώστε να πάρουμε μια λίστα με όλους του βαθμούς(σε όποιο μάθημα έχει βάθμο) και τα μαθήματα του συγκεκριμένου φοιτητή μόνο στο συγκεκριμένο εξάμηνο που επιλέχθηκε.

- GradesAverage()

```
var gradesAvg = _context.CourseHasStudents.Include(s => s.Student).Include(s => s.Course).
    Where(s => s.Student.Username.Equals(username) && s.GradeCourseStudent >= 5).Average(s=>s.GradeCourseStudent);

int coursesCount = _context.CourseHasStudents.Include(s => s.Student).
    Where(s => s.Student.Username.Equals(username) && s.GradeCourseStudent>=5).Count();
```

Στη τελευταία μέθοδο του φοιτητή χρησιμοποιούμε τη βάση για δύο λόγους. Αρχικά κάνουμε αναζήτηση στο πίνακα CourseHasStudents σε σύνδεση με τους πίνακες Student, Course τη πρώτη φορά για να πάρουμε το μέσο όρο μόνο των περασμένων μαθημάτων (βαθμός >=5) του φοιτητή και την δεύτερη ώστε να βρούμε το πλήθος αυτών των μαθημάτων.

2. ProfessorsController

- SelectByLesson()

```
var gradesByLesson = _context.CourseHasStudents.Include(s => s.Student).Include(s => s.Course).ThenInclude(s => s.Professor).
    Where(s => s.Course.Professor.Username.Equals(username) && s.GradeCourseStudent != null && s.IdCourse == SelectedCourseId);
```

Ο καθηγητής σε αυτή τη σελίδα έχει τη δυνατότητα να βλέπει τα μαθήματα που διδάσκει με όλους τους φοιτητές που το έχουν δηλώσει και έχουν βαθμολογηθεί . Επομένως χρειαζόμαστε τους πίνακες CourseHasStudents, σε σύνδεση με τους πίνακες Student , Course , Professor. Η αναζήτηση γίνεται με το συγκεκριμένο username του καθηγητή σε συνδυασμό με το μάθημα που επιλέγεται κάθε φορά .

- InsertGrades()

```
CourseHasStudent crs = new CourseHasStudent();

crs = _context.CourseHasStudents.
    FirstOrDefault(u => u.RegistrationNumber == RegNum && u.IdCourse == SelectedCourseId);

crs.GradeCourseStudent = grade;
_context.Update(crs);
_context.SaveChanges();
```

Ο καθηγητής έχει επίσης τη δυνατότητα να βαθμολογεί τους φοιτητές για τα μαθήματά του. Σε αυτή τη περίπτωση ο καθηγητής αρχικά πρέπει να επιλέξει το μάθημα , το φοιτητή τον οποίο θα βαθμολογήσει και το βαθμό. Γίνεται μια αναζήτητη στη βάση ώστε να βρεθεί η εγγραφή με το δηλωμένο μάθημα και ύστερα βαθμολογείται και αποθηκεύεται στη βάση.

3. SecretariesController

- SelectCourses()

```
var courses = _context.Courses.Include(s => s.Professor).
    Where(s => s.Professor.Department.Equals(department) && s.Afm != null);
```

Ο γραμματέας σε αυτή τη μέθοδο έχει τη δυνατότητα να προβάλει όλα τα μαθήματα του τμήματος του οποίου ανήκει στα οποία έχει ανατεθεί καθηγητής.Για να πραγματοποιηθεί αυτό πρέπει να κάνουμε μια σύνδεση στους πίνακες Courses και Professors και επιστρέφουμε τη λίστα με τα μαθήματα.

- InsertCourses()

```
_context.Add(course);
```

Ο γραμματέας μπορεί επίσης να προσθέσει μαθήματα αφού συμπληρώσει τα απαραίτητα πεδία στη φόρμα για το συγκεκριμένο μάθημα.

- InsertProfessors()

```
professor.Username = "p" + professor.Afm;
User user = _context.Users.FirstOrDefault(u => u.Username.Equals(professor.Username));
```

```
user = new User();
user.Username = professor.Username;
user.Password = professor.Username;
user.Role = "Professor";
professor.User = user;

_context.Add(professor);
await _context.SaveChangesAsync();
```

Ο γραμματέας μπορεί επίσης να προσθέσει καθηγητές αφού συμπληρώσει τα απαραίτητα πεδία στη φόρμα για το συγκεκριμένο καθηγητή.
Ως username και password θα χρησιμοποιηθεί το ΑΦΜ που δίνει ο γραμματέας με ένα χαρακτήρα “p” στην αρχή. Έπειτα ελέγχεται αν ήδη υπάρχει στη βάση και αν δεν υπάρχει δημιουργούμε το αντικείμενο και το προσθέτουμε στη βάση.

- InsertStudents()

```
User user = _context.Users.FirstOrDefault(u => u.Username.Equals(student.Username));
```

```
user = new User();
user.Username = student.Username;
user.Password = student.Username;
user.Role = "Student";
student.User = user;

_context.Add(student);
await _context.SaveChangesAsync();
```

Στη περίπτωση της προσθήκης μαθητή το username και το password είναι ίδια , αυτά δηλαδή που θα δώσει ο γραμματέας κατά συμπλήρωση της φόρμας.Αφού συμπληρώσει όλα τα άλλα απαραίτητα στοιχεία ελέγχεται αν ήδη υπάρχει στη βάση χρήστης με το ίδιο username και αν δεν υπάρχει δημιουργούμε το αντικείμενο του φοιτητή και το προσθέτουμε.

- AssignCourseToProfessor()

```
Course course = _context.Courses.FirstOrDefault(s => s.IdCourse == SelectedCourseId);
```

```
course.Afm = SelectedAfm;
_context.Update(course);
_context.SaveChanges();
```

Άλλη μία δυνατότητα του γραμματέα είναι να αναθέτει μάθημα σε καθηγητή.Σε αυτή τη περίπτωση γίνεται μια αναζήτηση στο πίνακα Courses και αφού βρεθεί το μάθημα στη βάση από το επιλεγμένο id που έδωσε ο χρήστης , προστίθεται το ΑΦΜ του επιλεγμένου καθηγητή και ενημερώνεται στη βάση.

- AssignCourseToStudent()

```
CourseHasStudent cs = new CourseHasStudent();
cs.IdCourse = (int)SelectedCourseId;
cs.RegistrationNumber = SelectedStudentId;

CourseHasStudent tmpcourse;
tmpcourse=_context.CourseHasStudents.
    FirstOrDefault(c => c.IdCourse == (int)SelectedCourseId && c.RegistrationNumber==SelectedStudentId);
if (tmpcourse == null)
{
    _context.Add(cs);
    _context.SaveChanges();
}
```

Τελευταία μέθοδος του γραμματέα είναι να δηλώνει μαθήματα σε φοιτητές. Πιο συγκεκριμένα , για να γίνει η δήλωση, αρκεί να δημιουργήσουμε μια εγγραφή στον πίνακα CourseHasStudents με το κωδικος εισαγωγής του φοιτητή και το κωδικό του μαθήματος.Αφού ελεγχθεί πως δεν υπάρχει στη βάση, εισάγεται η εγγραφή.