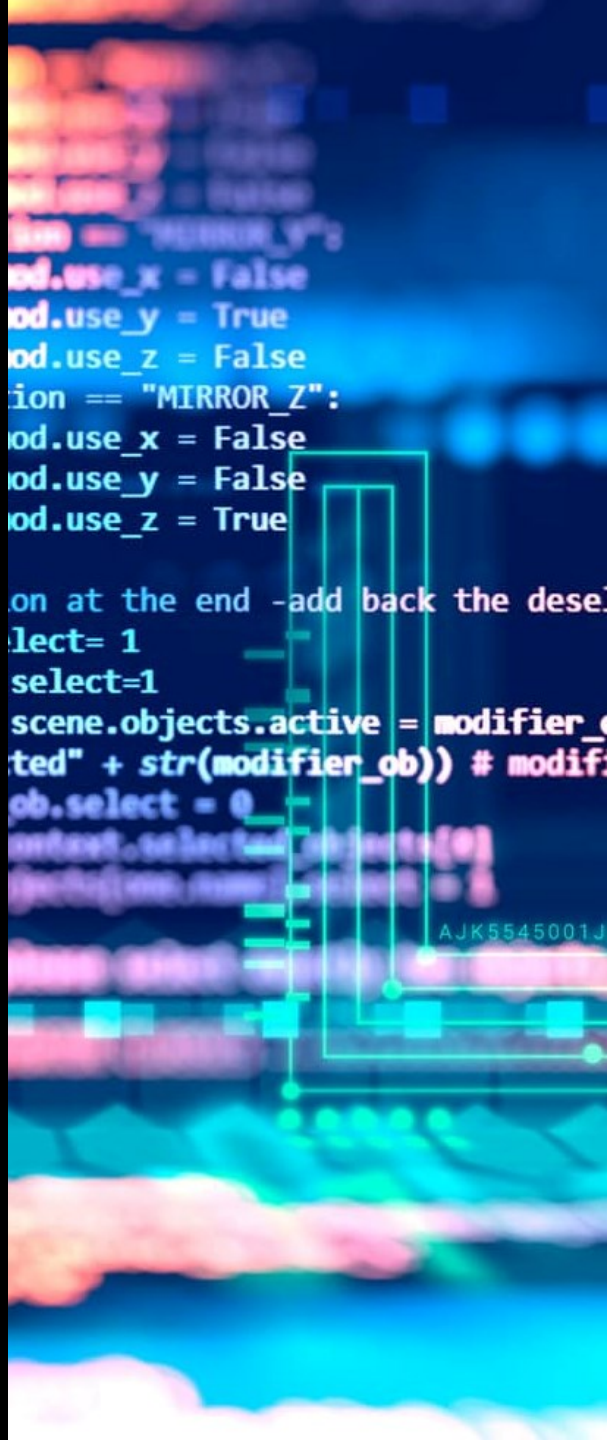
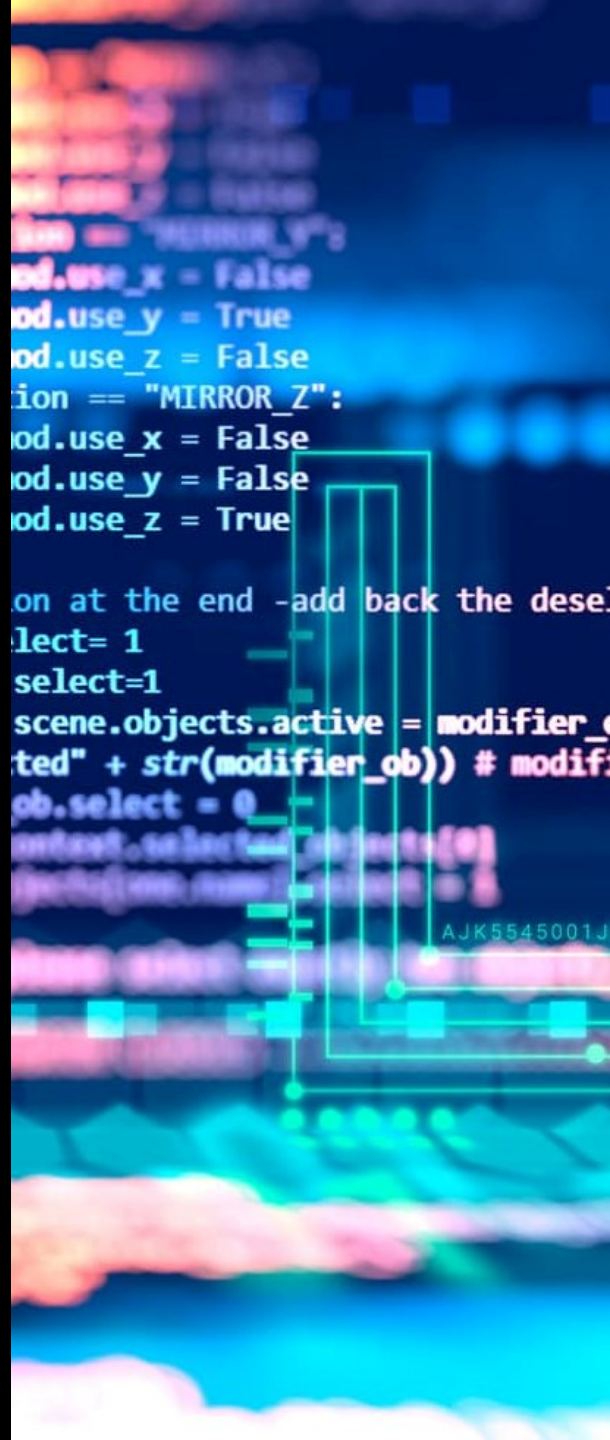
Final Project Report

ΟΝΟΜΑΤΕΠΩΝΥΜΟ DEVELOPER:
ΠΑΝΤΑΖΗΣ ΑΝΑΣΤΑΣΙΟΣ

ΕΙΔΙΚΟΤΗΤΑ: ΤΕΧΝΙΚΟΣ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ



Περιεχόμενα



Εισαγωγή

Στόχος μας είναι να υλοποιήσουμε μιά Full Stack εφαρμογή βασισμένη στην επέκταση της εφαρμογής που δημιουργήσαμε στα μαθήματα στα πλαίσια τους προγράμματος του ΣΕΒ στο AUEB.

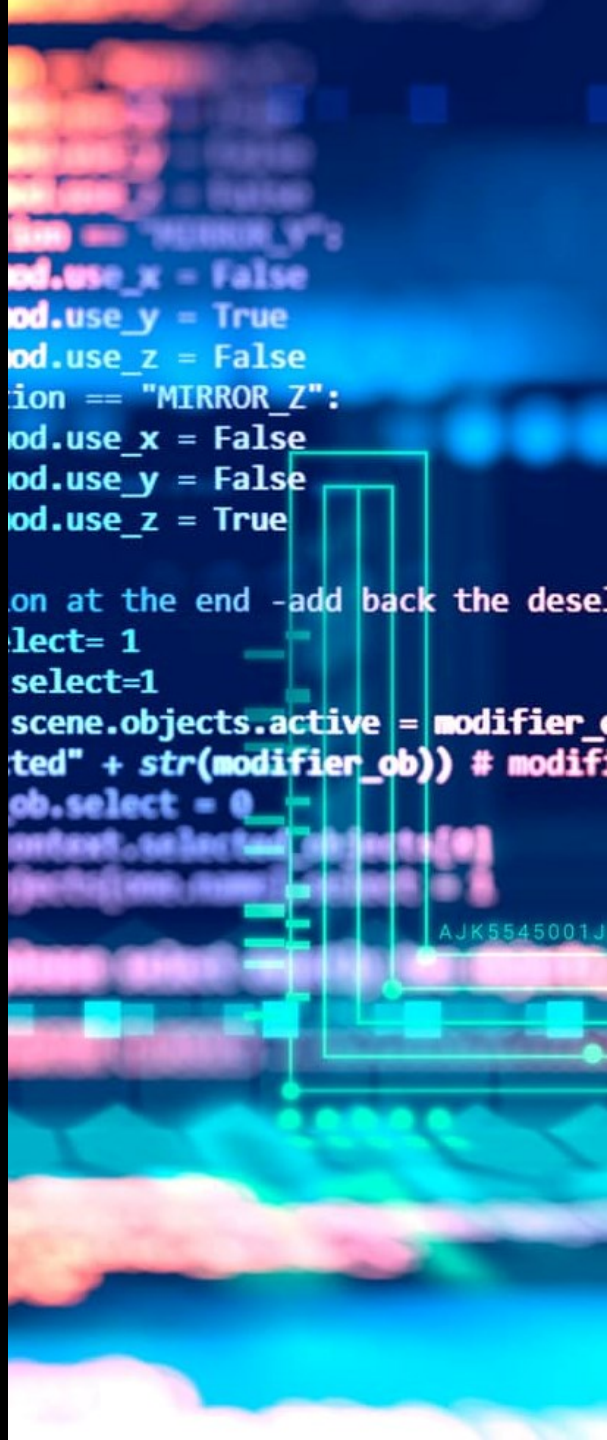
Σύμφωνα με το domain model που μας δώθηκε πρέπει να δημιουργήσουμε τα αντίστοιχα

DAO/DTO/Service Layer/Controllers/Web Pages

χρησιμοποιώντας βάση δεδομένων MSSQLSERVER,

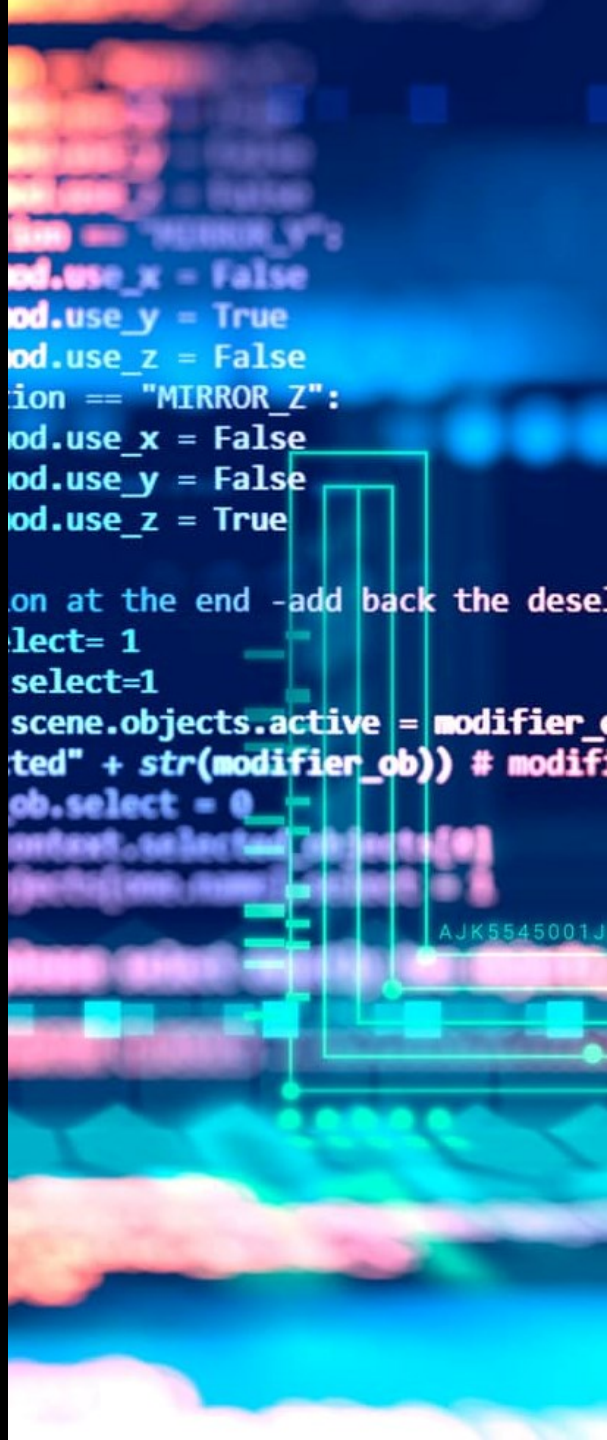
C# και ASP .NET με Razor Pages και Web Front-End

HTML/CSS/JavaScript



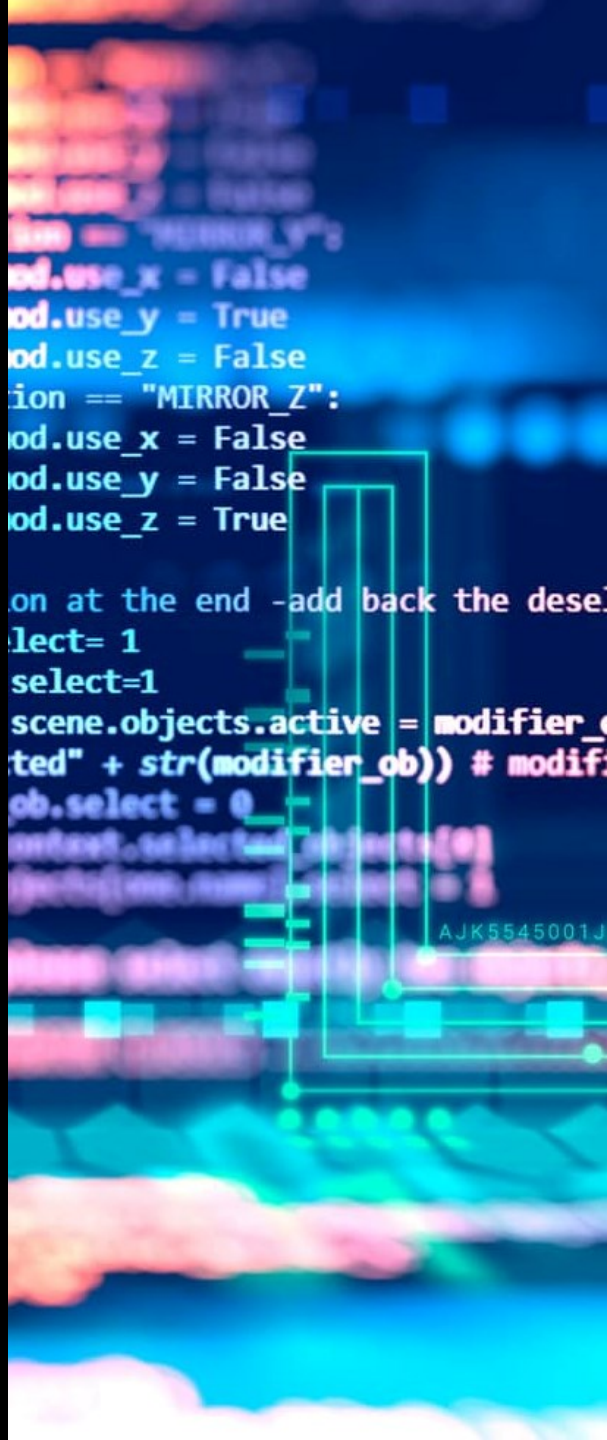
Ανάλυση Προβλήματος

Η μεγαλύτερη δυσκολία σε τέτοιου είδους εφαρμογές είναι ο μεγάλος όγκος τους καθώς και το πάντρεμα πολλών τεχνολογιών και μεθοδολογιών μαζί όπως και η απροβλημάτιστη λειτουργία όλων αυτών χωρίς σφάλματα



Σχεδιασμός κομμάτι 1ο: Εφαρμογή

Θα υλοποιήσουμε την εφαρμογή μας χρησιμοποιώντας την γλώσσα προγραμματισμού C# μαζί με ASP .NET με Razor Pages. Θα δημιουργήσουμε τις pages για να φτιάξουμε ένα μενού φιλικό ως προς στους χρήστες και θα έχουμε τους controllers που αλληλεπιδρούν για κάθε διεργασία μεταξύ τους σύμφωνα πάντα με τις κλάσεις του domain model και με βάση τα CRUD



Σχεδιασμός κομμάτι 2ο: Κώδικας

Η αρχιτεκτονική που θα χρησιμοποιήσουμε για τον κωδικά μας είναι η service-oriented architecture η αλλιώς (SOA).

Για να μεταφέρουμε δεδομένα μεταξύ διεργασιών θα χρησιμοποιήσουμε ένα Data transfer object(DTO) το οποίο το χρησιμοποιούμε για να μειώσουμε τις κλησεις μεθόδων.

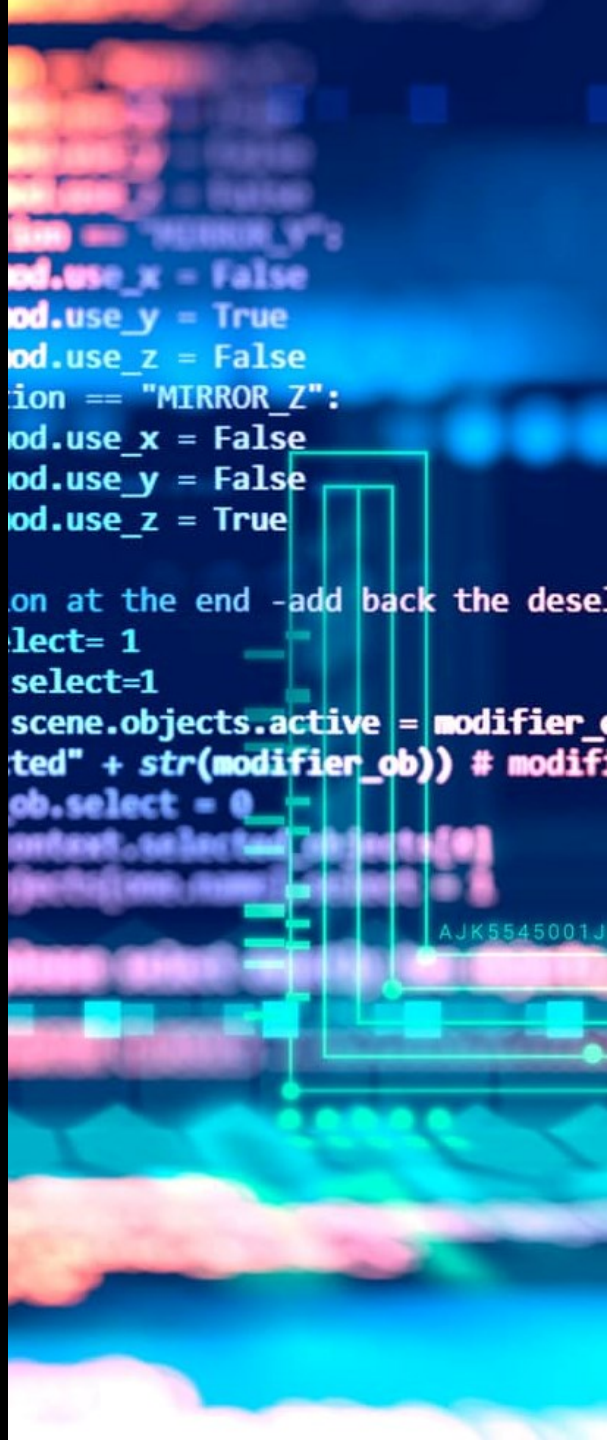
Data Access Object (Dao) Layer: χρησιμοποιείται για να έχουμε πρόσβαση στην βάση δεδομένων

Model Layer: περιέχει τα δεδομένα με τα οποία δουλεύει ο χρήστης(σχήμα,διεπαφές,βάσεις δεδομένων μαζί με τα πεδία τους

Service Layer: Το public API μας. Είναι μια γέφυρα ανάμεσα στα υψηλότερα και χαμηλότερα layer και το απαρτίζει μεγάλος αριθμός services όπου το καθένα εκτελεί τις δικές του business function

Controller Layer: είναι ο μαέστρος του Service Layer καθώς διαχειρίζεται τα εισερχόμενα request

View Layer: Οι cshtml pages που βλέπει ο χρήστης



Σχεδιασμός κομμάτι 3ο: Βάση Δεδομένων

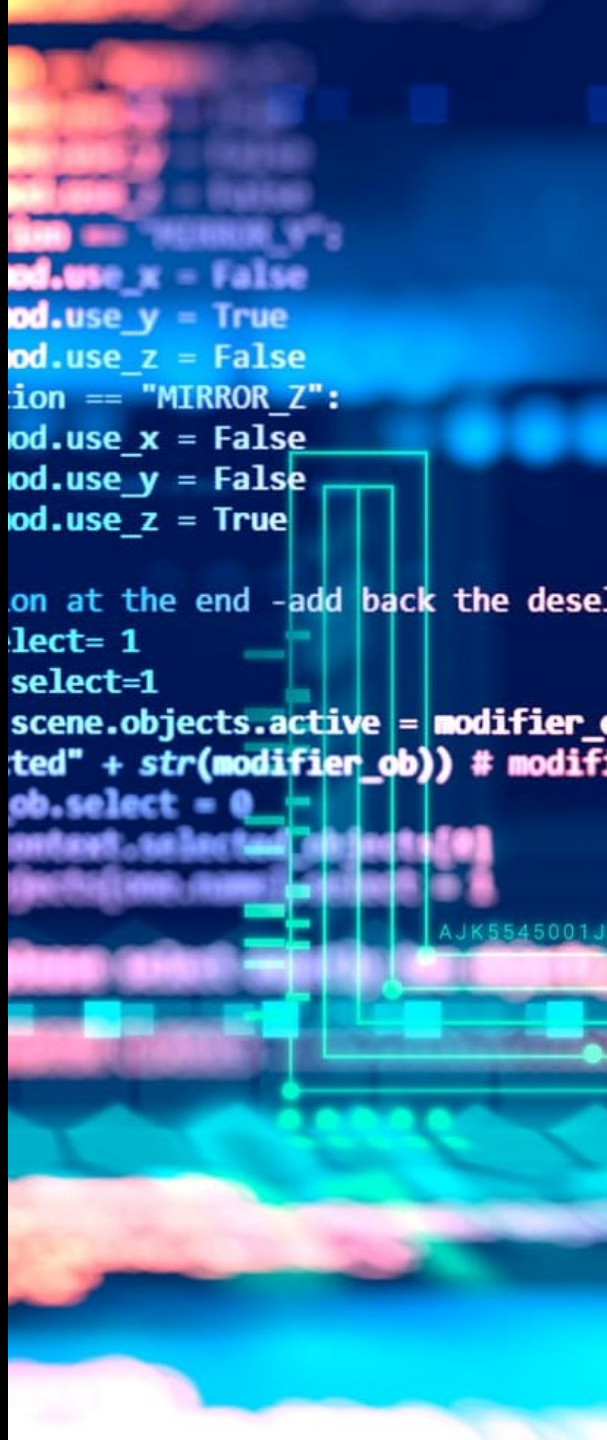
Η βάση δεδομένων που θα δημιουργήσουμε θα περιέχει 4 πίνακες σύμφωνα με τις 4 κλάσσεις που αναγράφονται στο Domain Model, οι οποίες είναι οι εξής:

Students Table(Μαθητές)

Teachers Table(Καθηγητές)

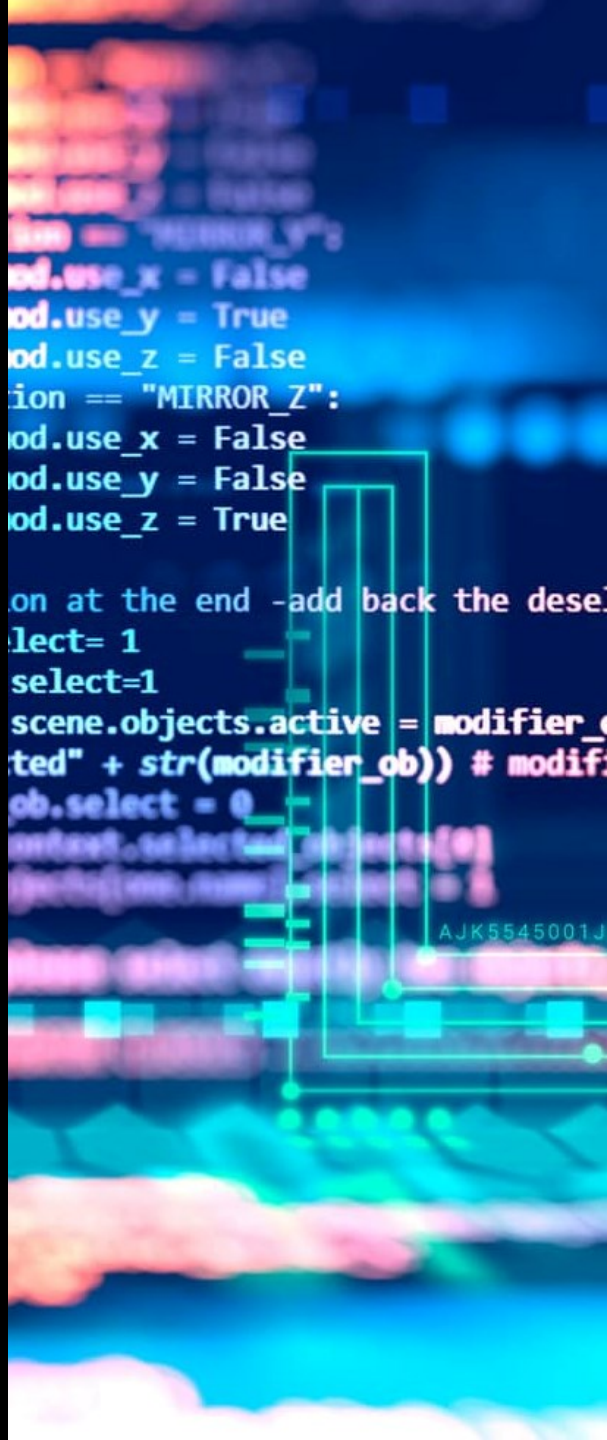
Courses Table(Μαθήματα)

StudentCourses(Εγγραφές)



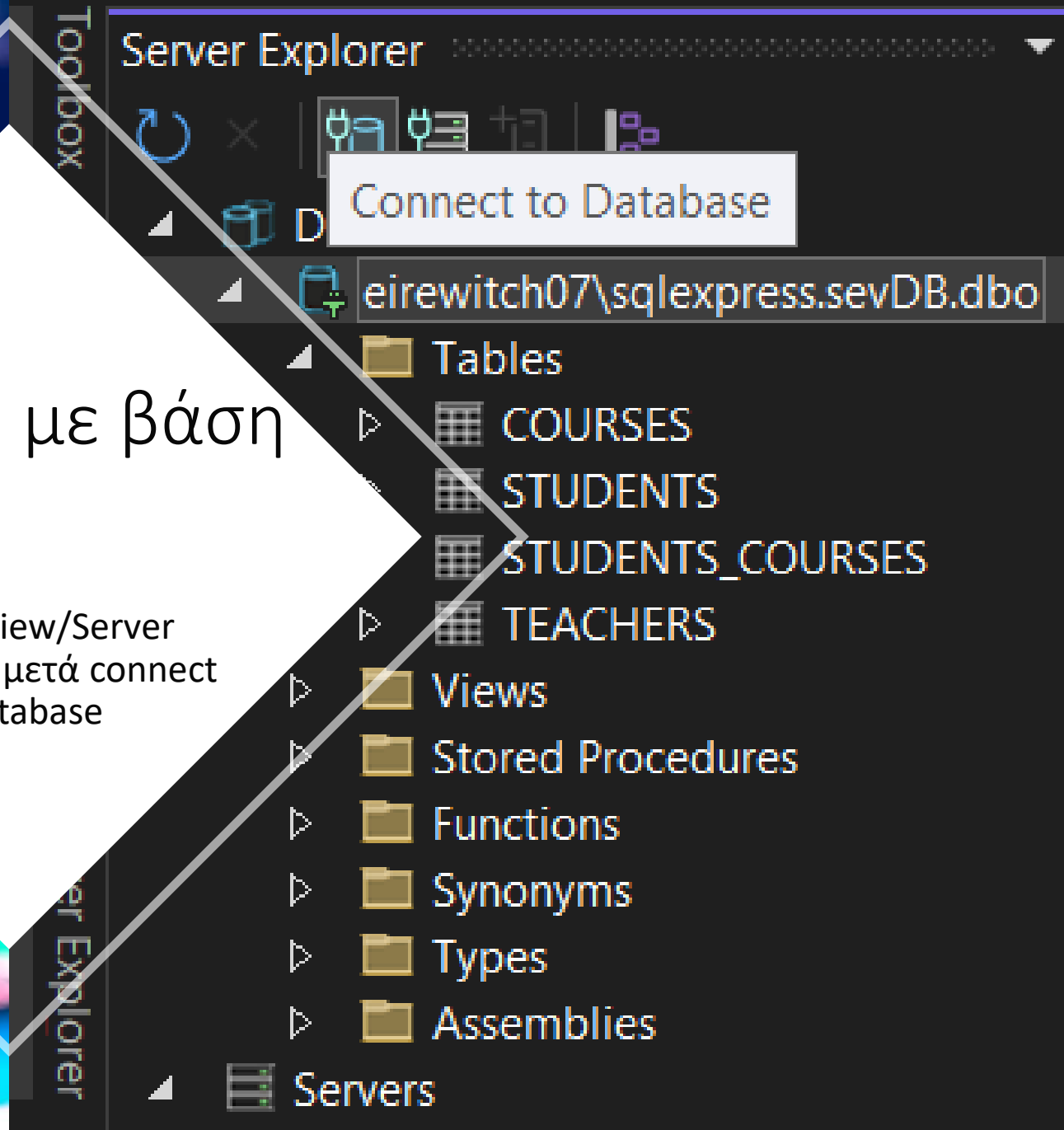
Υλοποίηση

Ξεκινάμε δημιουργώντας ένα νέο Web App σε .Net 6.0
Μέσα σε αυτό το Project θα υλοποιήσουμε την βάση μας

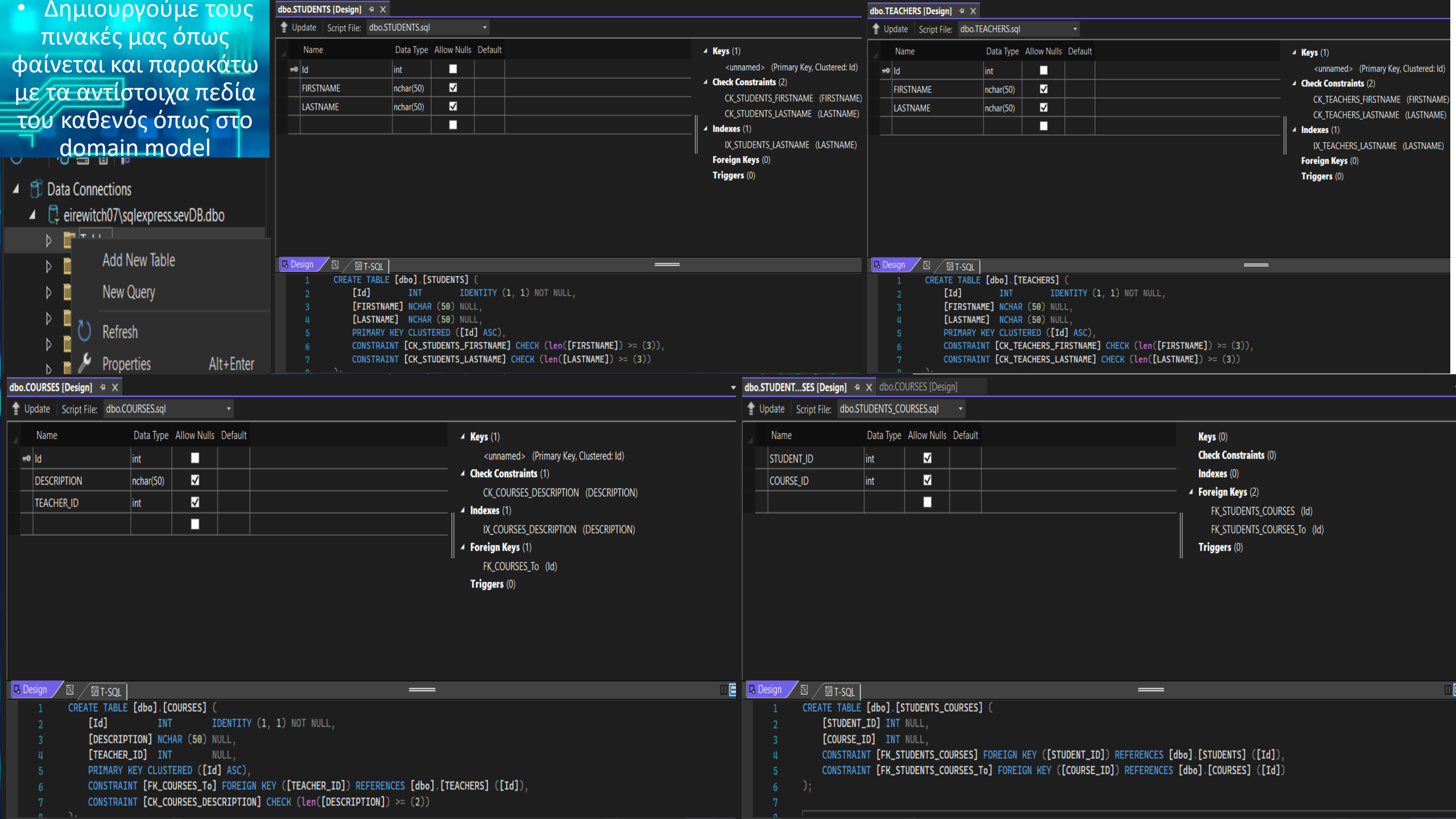


Σύνδεση με βάση

Πατάμε View/Server Explorer και μετά connect to database

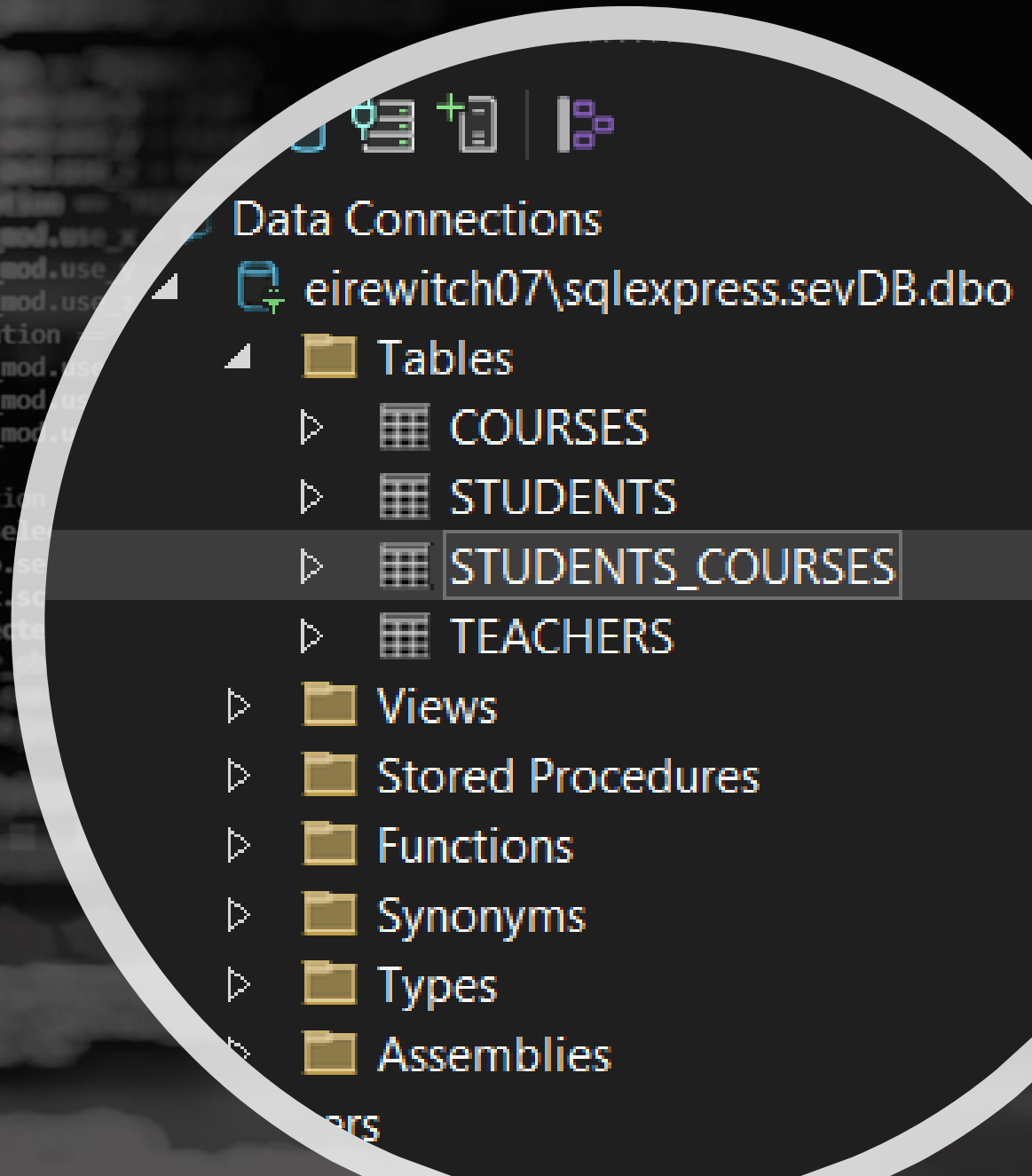


- Δημιουργούμε τους πινακές μας όπως φαίνεται και παρακάτω με τα αντίστοιχα πεδία του καθενός όπως στο domain model



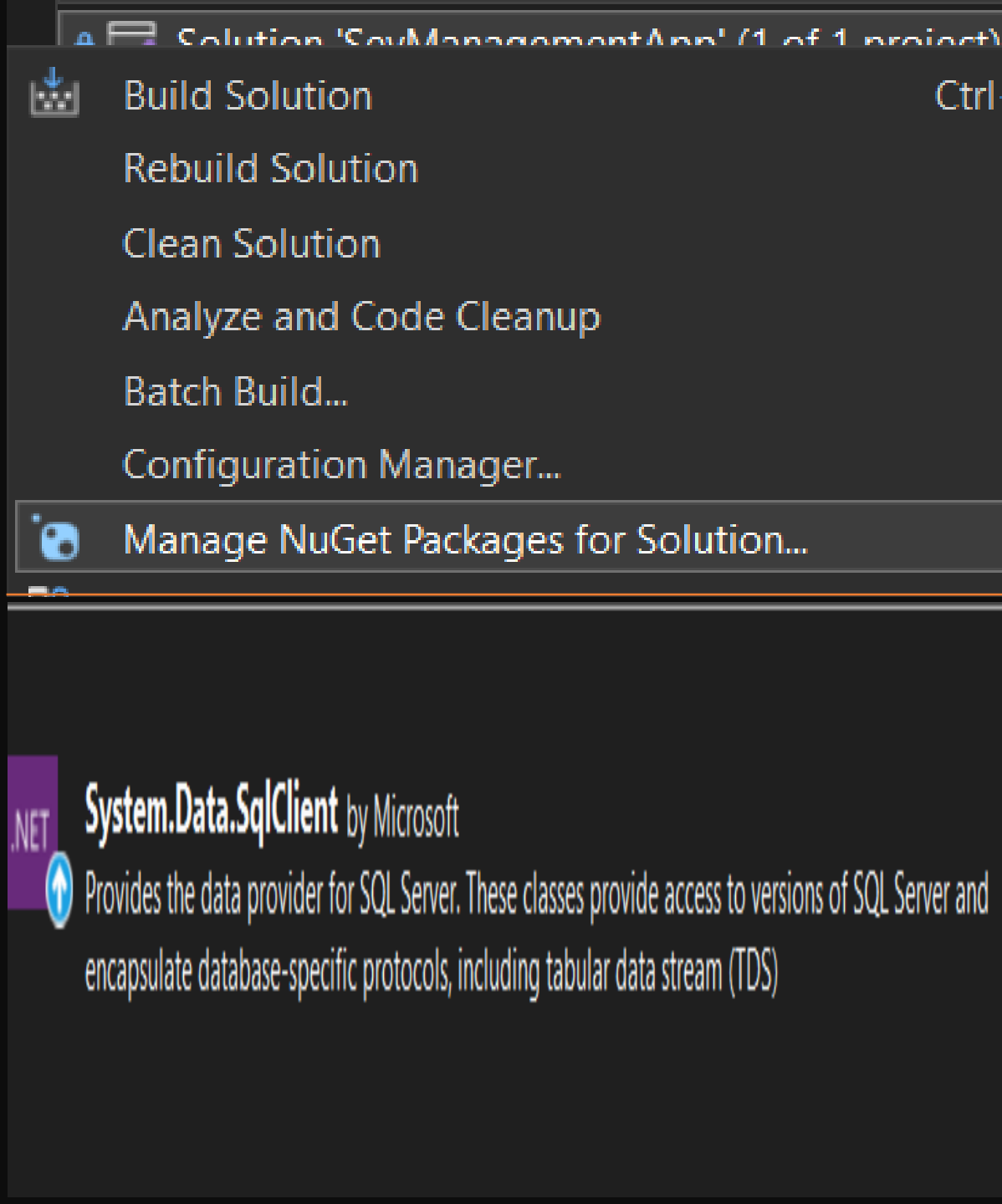
Update Database

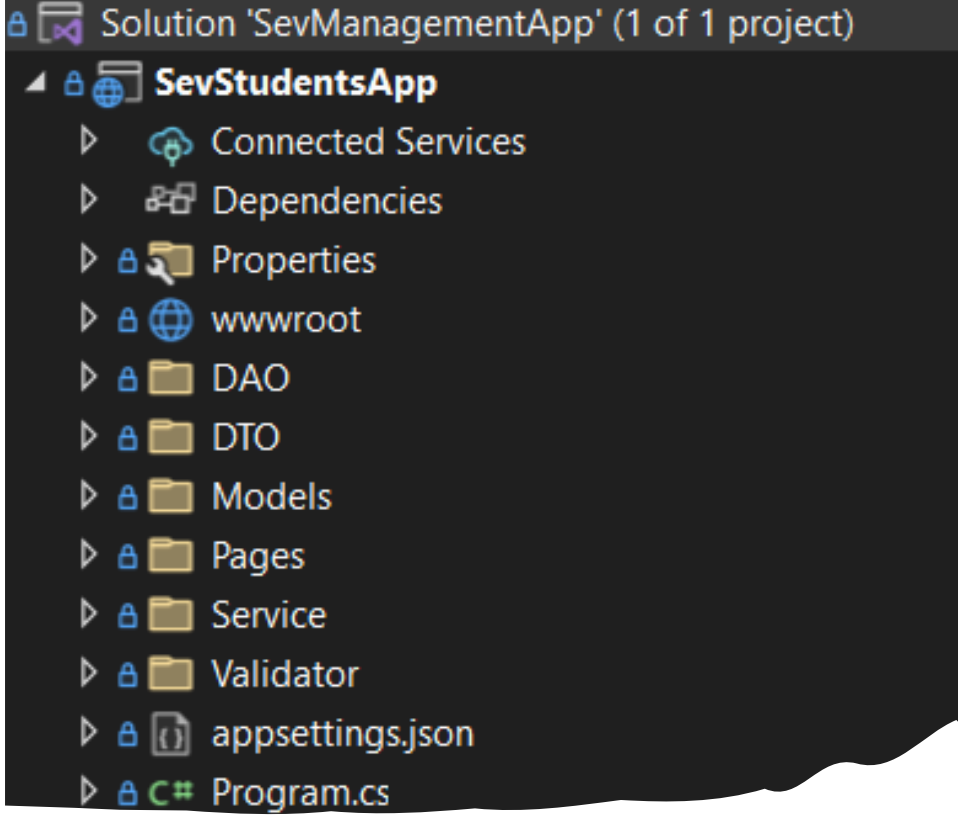
- Πατώντας Update στην συνέχεια πρέπει να έχουν δημιουργηθεί οι παρακάτω πίνακες



System.Data.SQLClient

- Το χρησιμοποιούμε ως driver σύνδεσης με ΒΔ MS SQLServer. Εισάγουμε το dependecy injection στο project με Nuget

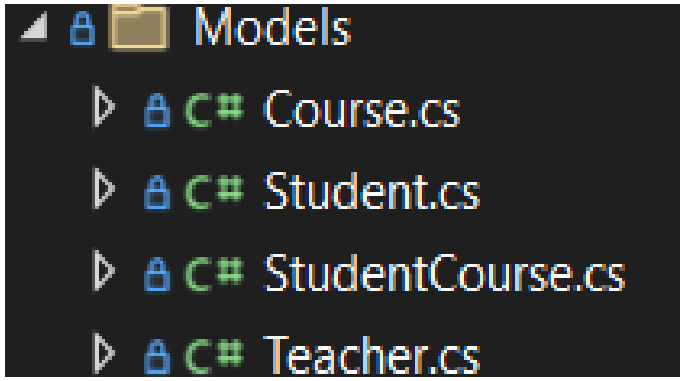




Δομή Project

Model

- Επειδή έχουμε 4 διαφορετικούς πίνακες χρειαζόμαστε και 4 διαφορετικά models



Student.cs

SevStudentsApp

```
namespace SevStudentsApp.Models
{
    34 references
    public class Student
    {
        9 references
        public int Id { get; set; }
        7 references
        public string? Firstname { get; set; }
        7 references
        public string? Lastname { get; set; }
    }
}
```

Teacher.cs

SevStudentsApp

```
namespace SevTeachersApp.Models
{
    34 references
    public class Teacher
    {
        9 references
        public int Id { get; set; }
        7 references
        public string? Firstname { get; set; }
        7 references
        public string? Lastname { get; set; }
    }
}
```

Course.cs

SevStudentsApp

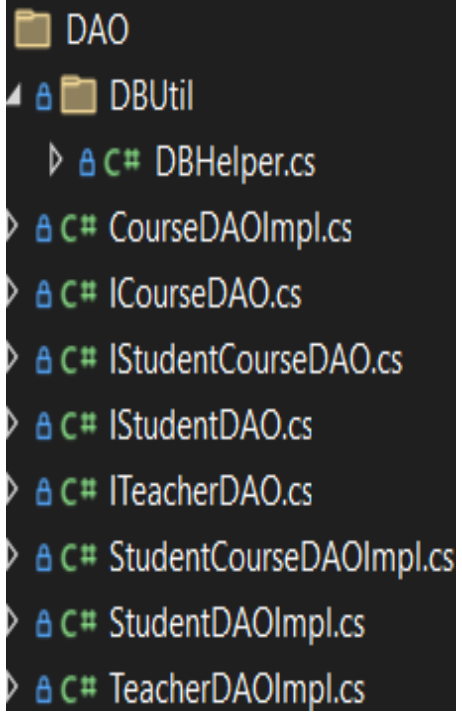
```
namespace SevCoursesApp.Models
{
    34 references
    public class Course
    {
        9 references
        public int Id { get; set; }
        7 references
        public string? Description { get; set;
        7 references
        public int TeacherId { get; set; }
    }
}
```

StudentCourse.cs

SevStudentsApp

```
namespace SevStudentCoursesApp.Models
{
    34 references
    public class StudentCourse
    {
        10 references
        public int StudentId { get; set; }
        7 references
        public int CourseId { get; set; }
    }
}
```

DAO-IStudentDAO
DAO-ITeacherDAO
DAO-ICourseDAO
DAO
IStudentCourseDAO




```cs
using SevStudentsApp.Models;

namespace SevStudentsApp.DAO
{
    7 references
    public interface IStudentDAO
    {
        2 references
        void Insert(Student? student);
        2 references
        void Update(Student? student);
        2 references
        Student? Delete(Student? student);
        2 references
        Student? GetStudent(int id);
        2 references
        List<Student> GetAll();
    }
}
```

```cs
using SevTeachersApp.Models;

namespace SevTeachersApp.DAO
{

    public interface ITeacherDAO
    {

        void Insert(Teacher? teacher);

        void Update(Teacher? teacher);

        Teacher? Delete(Teacher? teacher);

        Teacher? GetTeacher(int id);

        List<Teacher> GetAll();
    }
}
```

```cs
using SevCoursesApp.Models;

namespace SevCoursesApp.DAO
{

    public interface ICourseDAO
    {

        void Insert(Course? course);

        void Update(Course? course);
        2 references
        Course? Delete(Course? course);
        2 references
        Course? GetCourse(int id);
        2 references
        List<Course> GetAll();
    }
}
```

```cs
using SevStudentCoursesApp.Models;

namespace SevStudentCoursesApp.DAO
{
    7 references
    public interface IStudentCourseDAO
    {
        2 references
        void Insert(StudentCourse? studentcourse);
        2 references
        void Update(StudentCourse? studentcourse);
        2 references
        StudentCourse? Delete(StudentCourse? studentcourse);
        2 references
        StudentCourse? GetStudentCourse(int student_id);
        2 references
        List<StudentCourse> GetAll();
    }
}
```

- Επειδή έχουμε 4 διαφορετικούς πίνακες χρειαζόμαστε και 4 διαφορετικά DAO

StudentDAOImpl.cs | CourseDAOImpl.cs

SevStudentsApp | SevStudentsApp

```
1   using SevStudentsApp.DAO.DBUtil;
2   using SevStudentsApp.Models;
3   using System.Data.SqlClient;
4
5   namespace SevStudentsApp.DAO
6   {
        4 references
7       public class StudentDAOImpl : IStudentDAO
8       {
            2 references
9           public Student? Delete(Student? student)...
34
            2 references
35          public List<Student> GetAll()...
71
            2 references
72          public Student? GetStudent(int id)...
108
        2 references
109     public void Insert(Student? student)...
136
            2 references
137         public void Update(Student? student)...
164     }
165 }
```

TeacherDAOImpl.cs | StudentDAOImpl.cs | CourseDAOImpl.cs

SevStudentsApp | SevTeachersApp

```
1   using SevStudentsApp.DAO.DBUtil;
2   using SevTeachersApp.Models;
3   using System.Data.SqlClient;
4
5   namespace SevTeachersApp.DAO
6   {
        4 references
7       public class TeacherDAOImpl : ITeacherDAO
8       {
            2 references
9           public Teacher? Delete(Teacher? teacher)...
34
            2 references
35          public List<Teacher> GetAll()...
71
            2 references
72          public Teacher? GetTeacher(int id)...
108
            2 references
109         public void Insert(Teacher? teacher)...
136
            2 references
137         public void Update(Teacher? teacher)...
164     }
165 }
```

CourseDAOImpl.cs

SevStudentsApp | SevCoursesApp.I

```
1   using SevStudentsApp.DAO.DBUtil;
2   using SevCoursesApp.Models;
3   using System.Data.SqlClient;
4
5   namespace SevCoursesApp.DAO
6   {
        4 references
7       public class CourseDAOImpl : ICourseDAO
8       {
            2 references
9           public Course? Delete(Course? course)...
34
            2 references
35          public List<Course> GetAll()...
71
            2 references
72          public Course? GetCourse(int id)...
108
            2 references
109         public void Insert(Course? course)...
136
            2 references
137         public void Update(Course? course)...
164     }
165 }
```

StudentCourseDAOImpl.cs | TeacherDAOImpl.cs | StudentDAOImpl.cs | CourseDAOImpl.cs

SevStudentsApp | SevStudentCoursesApp.DAO.StudentCourseDA

```
1   using SevStudentsApp.DAO.DBUtil;
2   using SevStudentCoursesApp.Models;
3   using System.Data.SqlClient;
4
5   namespace SevStudentCoursesApp.DAO
6   {
        4 references
7       public class StudentCourseDAOImpl : IStudentCourseDAO
8       {
            2 references
9           public StudentCourse? Delete(StudentCourse? studentcourse)...
34
            2 references
35          public List<StudentCourse> GetAll()...
70
            2 references
71          public StudentCourse? GetStudentCourse(int student_id)...
106
            2 references
107         public void Insert(StudentCourse? studentcourse)...
134
            2 references
135         public void Update(StudentCourse? studentcourse)...
161     }
162 }
```

- Και τα implementations για κάθε ένα από τα 4 διαφορετικά Interface του DAO για κάθε πίνακα τα οποία τρέχουν queries

DBHelper Utility Class

DBHelper.cs

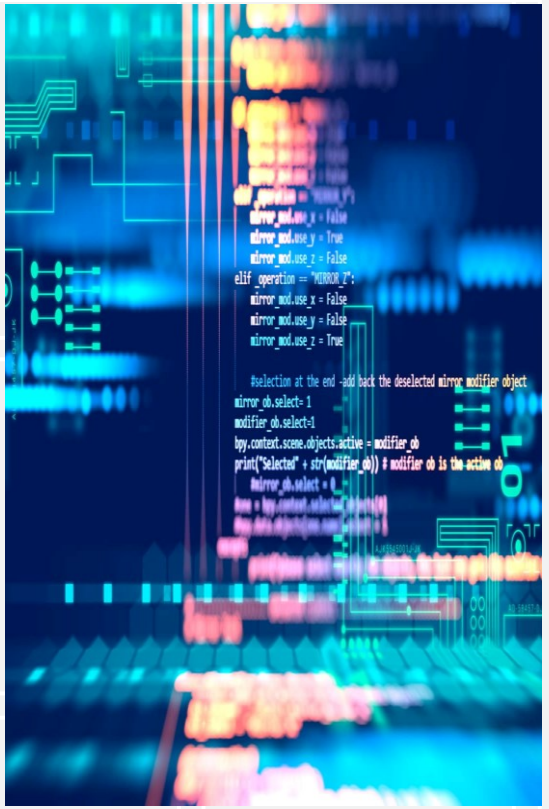
SevStudentsApp | SevStudentsApp.DAO.DBUtil.DBHelper

```
1   using System.Data.SqlClient;
2
3   namespace SevStudentsApp.DAO.DBUtil
4   {
        21 references
5       public class DBHelper
6       {
7           private static SqlConnection? conn;
8
9           //No instances of this class should be available
            0 references
10          private DBHelper() { }
11
            20 references
12          public static SqlConnection? GetConnection()
13          {
14              try
15              {
16                  ConfigurationManager configurationManager = new();
17                  configurationManager.AddJsonFile("appsettings.json");
18                  string url = configurationManager.GetConnectionString("DefaultConnection"
19                  //string url = "Data Source=localhost\\sqlexpress;Initial Catalog=sevDB;I
20                  conn = new SqlConnection(url);
21                  return conn;
22
23              } catch (Exception e)
24              {
```

DBHelper.cs

SevStudentsApp | SevStudentsApp

```
23              } catch (Exception e)
24              {
25                  Console.WriteLine(e.StackTrace);
26                  return null;
27              }
28          }
29
            0 references
30          public static void CloseConnection()
31          {
32              if (conn is not null)
33              {
34                  conn.Close();
35              }
36          }
```



appsettings.json

Schema: https://json.schemastore.org/appsettings.json

```
1   {
2     "Logging": {
3       "LogLevel": {
4         "Default": "Information",
5         "Microsoft.AspNetCore": "Warning"
6       }
7     },
8     "AllowedHosts": "*",
9     "ConnectionStrings": {
10      "DefaultConnection": "Data Source=localhost\\sqlexpress;Ir
11    }
12  }
13
```

Η DBHelper Utility Class είναι μια κλάση την οποία χρησιμοποιούμε για να συνδέσουμε την εφαρμογή μας και την βάση μας χρησιμοποιώντας ένα Connection String το οποίο αποθηκεύεται σε ένα config file, το appsettings.json

INSERT METHOD

StudentCourseDAOImpl.cs | TeacherDAOImpl.cs | **StudentDAOImpl.cs** | CourseDAOImpl.cs

SevStudentsApp — SevStudentsApp.DAO.StudentDAOImpl

```
2 references
109  public void Insert(Student? student)
110  {
111      if (student == null) return;
112
113      try
114      {
115          using SqlConnection? conn = DBHelper.GetConnection();
116
117          if (conn is not null) conn.Open(); else return;
118
119          string sql = "INSERT INTO STUDENTS " +
120              "(FIRSTNAME, LASTNAME) VALUES " +
121              "(@firstname, @lastname)";
122
123          using SqlCommand command = new SqlCommand(@sql, conn);
124
125          command.Parameters.AddWithValue("@firstname", student.Firstname);
126          command.Parameters.AddWithValue("@lastname", student.Lastname);
127
128          command.ExecuteNonQuery();
129      }
130      catch (Exception e)
131      {
132          Console.WriteLine(e.StackTrace);
133          throw;
134      }
```

StudentCourseDAOImpl.cs | TeacherDAOImpl.cs | StudentDAOImpl.cs | CourseDAOImpl.cs

SevStudentsApp — SevTeachersApp.DAO.TeacherDAOImpl

```
109  public void Insert(Teacher? teacher)
110  {
111      if (teacher == null) return;
112
113      try
114      {
115          using SqlConnection? conn = DBHelper.GetConnection();
116
117          if (conn is not null) conn.Open(); else return;
118
119          string sql = "INSERT INTO TEACHERS " +
120              "(FIRSTNAME, LASTNAME) VALUES " +
121              "(@firstname, @lastname)";
122
123          using SqlCommand command = new SqlCommand(@sql, conn);
124
125          command.Parameters.AddWithValue("@firstname", teacher.Firstname);
126          command.Parameters.AddWithValue("@lastname", teacher.Lastname);
127
128          command.ExecuteNonQuery();
129      }
130      catch (Exception e)
131      {
132          Console.WriteLine(e.StackTrace);
133          throw;
134      }
135  }
```

CourseDAOImpl.cs

SevStudentsApp — SevCoursesApp.DAO.CourseDAOImpl

```
2 references
109  public void Insert(Course? course)
110  {
111      if (course == null) return;
112
113      try
114      {
115          using SqlConnection? conn = DBHelper.GetConnection();
116
117          if (conn is not null) conn.Open(); else return;
118
119          string sql = "INSERT INTO COURSES " +
120              "(DESCRIPTION, TEACHER_ID) VALUES " +
121              "(@description, @teacherid)";
122
123          using SqlCommand command = new SqlCommand(@sql, conn);
124
125          command.Parameters.AddWithValue("@description", course.Description);
126          command.Parameters.AddWithValue("@teacherid", course.TeacherId);
127
128          command.ExecuteNonQuery();
129      }
130      catch (Exception e)
131      {
132          Console.WriteLine(e.StackTrace);
133          throw;
134      }
```

StudentCourseDAOImpl.cs

SevStudentsApp — SevStudentCoursesApp.DAO.StudentCourseDAOImpl

```
2 references
107  public void Insert(StudentCourse? studentcourse)
108  {
109      if (studentcourse == null) return;
110
111      try
112      {
113          using SqlConnection? conn = DBHelper.GetConnection();
114
115          if (conn is not null) conn.Open(); else return;
116
117          string sql = "INSERT INTO STUDENTS_COURSES " +
118              "(STUDENT_ID, COURSE_ID) VALUES " +
119              "(@student_id, @course_id)";
120
121          using SqlCommand command = new SqlCommand(@sql, conn);
122
123          command.Parameters.AddWithValue("@student_id", studentcourse.StudentId);
124          command.Parameters.AddWithValue("@course_id", studentcourse.CourseId);
125
126          command.ExecuteNonQuery();
127      }
128      catch (Exception e)
129      {
130          Console.WriteLine(e.StackTrace);
131          throw;
132      }
```

Η Insert Method εισάγει δεδομένα στον πίνακα
Αντιστοιχεί στο γράμμα C(Create) από το CRUD

GET METHOD

StudentDAOImpl.cs

```
SevStudentsApp  SevStudentsApp.DAO.StudentDAOImpl
2 references
72  public Student? GetStudent(int id)
73  {
74      Student? student = null;
75
76      try
77      {
78          using SqlConnection? conn = DBHelper.GetConnection();
79
80          if (conn is not null) conn.Open();
81
82          string sql = "SELECT * FROM STUDENTS WHERE ID = @id";
83
84          using SqlCommand command = new SqlCommand(sql, conn);
85
86          command.Parameters.AddWithValue("@id", id);
87
88          using SqlDataReader reader = command.ExecuteReader();
89
90          if (reader.Read())
91          {
92              student = new Student()
93              {
94                  Id = reader.GetInt32(0),
95                  Firstname = reader.GetString(1),
96                  Lastname = reader.GetString(2)
97              };
```

TeacherDAOImpl.cs

```
SevStudentsApp  SevTeachersApp.DAO.TeacherDAOImpl
72  public Teacher? GetTeacher(int id)
73  {
74      Teacher? teacher = null;
75
76      try
77      {
78          using SqlConnection? conn = DBHelper.GetConnection();
79
80          if (conn is not null) conn.Open();
81
82          string sql = "SELECT * FROM TEACHERS WHERE ID = @id";
83
84          using SqlCommand command = new SqlCommand(sql, conn);
85
86          command.Parameters.AddWithValue("@id", id);
87
88          using SqlDataReader reader = command.ExecuteReader();
89
90          if (reader.Read())
91          {
92              teacher = new Teacher()
93              {
94                  Id = reader.GetInt32(0),
95                  Firstname = reader.GetString(1),
96                  Lastname = reader.GetString(2)
97              };
98          }
```

CourseDAOImpl.cs TeacherDAOImpl.cs

```
SevStudentsApp  SevCoursesApp.DAO.CourseDAOImpl
2 references
72  public Course? GetCourse(int id)
73  {
74      Course? course = null;
75
76      try
77      {
78          using SqlConnection? conn = DBHelper.GetConnection();
79
80          if (conn is not null) conn.Open();
81
82          string sql = "SELECT * FROM COURSES WHERE ID = @id";
83
84          using SqlCommand command = new SqlCommand(sql, conn);
85
86          command.Parameters.AddWithValue("@id", id);
87
88          using SqlDataReader reader = command.ExecuteReader();
89
90          if (reader.Read())
91          {
92              course = new Course()
93              {
94                  Id = reader.GetInt32(0),
95                  Description = reader.GetString(1),
96                  TeacherId = reader.GetInt32(2)
97              };
```

StudentCourseDAOImpl.cs

```
SevStudentsApp  SevStudentCoursesApp.DAO.StudentCourseDAOImpl
2 references
107 public void Insert(StudentCourse? studentcourse)
108 {
109     if (studentcourse == null) return;
110
111     try
112     {
113         using SqlConnection? conn = DBHelper.GetConnection();
114
115         if (conn is not null) conn.Open(); else return;
116
117         string sql = "INSERT INTO STUDENTS_COURSES " +
118             "(STUDENT_ID, COURSE_ID) VALUES " +
119             "(@student_id, @course_id)";
120
121         using SqlCommand command = new SqlCommand(@sql, conn);
122
123         command.Parameters.AddWithValue("@student_id", studentcourse.StudentId);
124         command.Parameters.AddWithValue("@course_id", studentcourse.CourseId);
125
126         command.ExecuteNonQuery();
127     }
128     catch (Exception e)
129     {
130         Console.WriteLine(e.StackTrace);
131         throw;
132     }
```

Η Get Method διαβάζει και επιστρέφει μια στήλη μέσα στονπίνακα

Αντιστοιχεί στο γράμμα R(Read) από το CRUD

GET ALL METHOD

StudentDAOImpl.cs

SevStudentsApp — SevStudentsApp.DAO.StudentDAOImpl

```
35  public List<Student> GetAll()
36  {
37      List<Student> students = new List<Student>();
38
39      try
40      {
41          using SqlConnection? conn = DBHelper.GetConnection();
42
43          if (conn is not null) conn.Open();
44
45          string sql = "SELECT * FROM STUDENTS";
46
47          using SqlCommand command = new SqlCommand(sql, conn);
48          using SqlDataReader reader = command.ExecuteReader();
49
50          while (reader.Read())
51          {
52              Student student = new Student()
53              {
54                  Id = reader.GetInt32(0),
55                  Firstname = reader.GetString(1),
56                  Lastname = reader.GetString(2)
57              };
58
59              students.Add(student);
60          }
61
```

TeacherDAOImpl.cs

SevStudentsApp — SevTeachersApp.DAO.TeacherDAOImpl

```
2 references
35  public List<Teacher> GetAll()
36  {
37      List<Teacher> teachers = new List<Teacher>();
38
39      try
40      {
41          using SqlConnection? conn = DBHelper.GetConnection();
42
43          if (conn is not null) conn.Open();
44
45          string sql = "SELECT * FROM TEACHERS";
46
47          using SqlCommand command = new SqlCommand(sql, conn);
48          using SqlDataReader reader = command.ExecuteReader();
49
50          while (reader.Read())
51          {
52              Teacher teacher = new Teacher()
53              {
54                  Id = reader.GetInt32(0),
55                  Firstname = reader.GetString(1),
56                  Lastname = reader.GetString(2)
57              };
58
59              teachers.Add(teacher);
60          }
```

CourseDAOImpl.cs

SevStudentsApp — SevCoursesApp.DAO.CourseDAOImpl

```
2 references
35  public List<Course> GetAll()
36  {
37      List<Course> courses = new List<Course>();
38
39      try
40      {
41          using SqlConnection? conn = DBHelper.GetConnection();
42
43          if (conn is not null) conn.Open();
44
45          string sql = "SELECT * FROM COURSES";
46
47          using SqlCommand command = new SqlCommand(sql, conn);
48          using SqlDataReader reader = command.ExecuteReader();
49
50          while (reader.Read())
51          {
52              Course course = new Course()
53              {
54                  Id = reader.GetInt32(0),
55                  Description = reader.GetString(1),
56                  TeacherId = reader.GetInt32(2)
57              };
58
59              courses.Add(course);
60          }
```

StudentCourseDAOImpl.cs | CourseDAOImpl.cs

SevStudentsApp — SevStudentCoursesApp.DAO.StudentCourseDAOImpl

```
2 references
35  public List<StudentCourse> GetAll()
36  {
37      List<StudentCourse> studentcourses = new List<StudentCourse>();
38
39      try
40      {
41          using SqlConnection? conn = DBHelper.GetConnection();
42
43          if (conn is not null) conn.Open();
44
45          string sql = "SELECT * FROM STUDENTS_COURSES";
46
47          using SqlCommand command = new SqlCommand(sql, conn);
48          using SqlDataReader reader = command.ExecuteReader();
49
50          while (reader.Read())
51          {
52              StudentCourse studentcourse = new StudentCourse()
53              {
54                  StudentId = reader.GetInt32(0),
55                  CourseId = reader.GetInt32(1)
56              };
57
58              studentcourses.Add(studentcourse);
59          }
60
```

Η Get All Method διαβάζει και επιστρέφει όλες τις στήλες μέσα στον πίνακα

Αντιστοιχεί στο γράμμα R(Read) από το CRUD

UPDATE METHOD

StudentDAOImpl.cs

```
SevStudentsApp  SevStudentsApp.DAO.StudentDAOImpl
137  public void Update(Student? student)
138  {
139      if (student == null) return;
140
141      try
142      {
143          using SqlConnection? conn = DBHelper.GetConnection();
144
145          if (conn is not null) conn.Open(); else return;
146
147          string sql = "UPDATE STUDENTS SET FIRSTNAME = @firstname, " +
148              "LASTNAME = @lastname WHERE ID = @id";
149
150          using SqlCommand command = new SqlCommand(@sql, conn);
151
152          command.Parameters.AddWithValue("@firstname", student.Firstname);
153          command.Parameters.AddWithValue("@lastname", student.Lastname);
154          command.Parameters.AddWithValue("@id", student.Id);
155
156          command.ExecuteNonQuery();
157      }
158      catch (Exception e)
159      {
160          Console.WriteLine(e.StackTrace);
161          throw;
162      }
163  }
```

TeacherDAOImpl.cs

```
SevStudentsApp  SevTeachersApp.DAO.TeacherDAOImpl
137  public void Update(Teacher? teacher)
138  {
139      if (teacher == null) return;
140
141      try
142      {
143          using SqlConnection? conn = DBHelper.GetConnection();
144
145          if (conn is not null) conn.Open(); else return;
146
147          string sql = "UPDATE TEACHERS SET FIRSTNAME = @firstname, " +
148              "LASTNAME = @lastname WHERE ID = @id";
149
150          using SqlCommand command = new SqlCommand(@sql, conn);
151
152          command.Parameters.AddWithValue("@firstname", teacher.Firstname);
153          command.Parameters.AddWithValue("@lastname", teacher.Lastname);
154          command.Parameters.AddWithValue("@id", teacher.Id);
155
156          command.ExecuteNonQuery();
157      }
158      catch (Exception e)
159      {
160          Console.WriteLine(e.StackTrace);
161          throw;
162      }
163  }
```

CourseDAOImpl.cs

```
SevStudentsApp  SevCoursesApp.DAO.CourseDAOImpl
2 references
137  public void Update(Course? course)
138  {
139      if (course == null) return;
140
141      try
142      {
143          using SqlConnection? conn = DBHelper.GetConnection();
144
145          if (conn is not null) conn.Open(); else return;
146
147          string sql = "UPDATE COURSES SET DESCRIPTION = @description, " +
148              "TEACHER_ID = @teacherid WHERE ID = @id";
149
150          using SqlCommand command = new SqlCommand(@sql, conn);
151
152          command.Parameters.AddWithValue("@description", course.Description);
153          command.Parameters.AddWithValue("@teacherid", course.TeacherId);
154          command.Parameters.AddWithValue("@id", course.Id);
155
156          command.ExecuteNonQuery();
157      }
158      catch (Exception e)
159      {
160          Console.WriteLine(e.StackTrace);
161          throw;
162      }
```

StudentCourseDAOImpl.cs

```
SevStudentsApp  SevStudentCoursesApp.DAO.StudentCourseDAOImpl
2 references
135  public void Update(StudentCourse? studentcourse)
136  {
137      if (studentcourse == null) return;
138
139      try
140      {
141          using SqlConnection? conn = DBHelper.GetConnection();
142
143          if (conn is not null) conn.Open(); else return;
144
145          string sql = "UPDATE STUDENTS_COURSES SET STUDENT_ID = @student_id, " +
146              "COURSE_ID = @course_id";
147
148          using SqlCommand command = new SqlCommand(@sql, conn);
149
150          command.Parameters.AddWithValue("@student_id", studentcourse.StudentId);
151          command.Parameters.AddWithValue("@course_id", studentcourse.CourseId);
152
153          command.ExecuteNonQuery();
154      }
155      catch (Exception e)
156      {
157          Console.WriteLine(e.StackTrace);
158          throw;
159      }
160  }
```

Η Update Method τροποποιεί μιά στήλημέσα στον πίνακα

Αντιστοιχεί στο γράμμα U(Update) από το CRUD

DELETE METHOD

StudentDAOImpl.cs

SevStudentsApp — SevStudentsApp.DAO.StudentDAOImpl

```
public Student? Delete(Student? student)
{
    if (student == null) return null;

    try
    {
        using SqlConnection? conn = DBHelper.GetConnection();

        if (conn is not null) conn.Open(); else return null;

        string sql = "DELETE FROM STUDENTS WHERE ID = @id";

        using SqlCommand command = new SqlCommand(@sql, conn);

        command.Parameters.AddWithValue("@id", student.Id);

        int rowsAffected = command.ExecuteNonQuery();
        return (rowsAffected > 0) ? student : null;
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

SevStudentsApp — SevTeachersApp.DAO.TeacherDAOImpl

```
public Teacher? Delete(Teacher? teacher)
{
    if (teacher == null) return null;

    try
    {
        using SqlConnection? conn = DBHelper.GetConnection();

        if (conn is not null) conn.Open(); else return null;

        string sql = "DELETE FROM TEACHERS WHERE ID = @id";

        using SqlCommand command = new SqlCommand(@sql, conn);

        command.Parameters.AddWithValue("@id", teacher.Id);

        int rowsAffected = command.ExecuteNonQuery();
        return (rowsAffected > 0) ? teacher : null;
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

CourseDAOImpl.cs

SevStudentsApp — SevCoursesApp.DAO.CourseDAOImpl

```
public Course? Delete(Course? course)
{
    if (course == null) return null;

    try
    {
        using SqlConnection? conn = DBHelper.GetConnection();

        if (conn is not null) conn.Open(); else return null;

        string sql = "DELETE FROM COURSES WHERE ID = @id";

        using SqlCommand command = new SqlCommand(@sql, conn);

        command.Parameters.AddWithValue("@id", course.Id);

        int rowsAffected = command.ExecuteNonQuery();
        return (rowsAffected > 0) ? course : null;
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

StudentCourseDAOImpl.cs

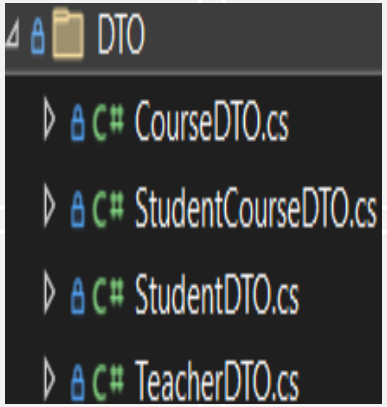
SevStudentsApp — SevStudentCoursesApp.DAO.StudentCourseDAOImpl — Delete(

```
2 references
public StudentCourse? Delete(StudentCourse? studentcourse)
{
    if (studentcourse == null) return null;

    try
    {
        using SqlConnection? conn = DBHelper.GetConnection();

        if (conn is not null) conn.Open(); else return null;

        string sql = "DELETE FROM STUDENTS_COURSES WHERE STUDENT_ID = @student_id";

        using SqlCommand command = new SqlCommand(@sql, conn);

        command.Parameters.AddWithValue("@student_id", studentcourse.StudentId);

        int rowsAffected = command.ExecuteNonQuery();
        return (rowsAffected > 0) ? studentcourse : null;
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

Η Delete Method διαγράφει μιά στήλη μέσα στον πίνακα

Αντιστοιχεί στο γράμμα D(Delete) από το CRUD

DTO



StudentDTO.cs

```
SevStudentsApp
namespace SevStudentsApp.DTO
{
    16 references
    public class StudentDTO
    {
        5 references
        public int Id { get; set; }
        7 references
        public string? Firstname { get; set; }
        7 references
        public string? Lastname { get; set; }
    }
}
```

TeacherDTO.cs

```
SevStudentsApp
namespace SevTeachersApp.DTO
{
    16 references
    public class TeacherDTO
    {
        5 references
        public int Id { get; set; }
        7 references
        public string? Firstname { get; set; }
        7 references
        public string? Lastname { get; set; }
    }
}
```

CourseDTO.cs

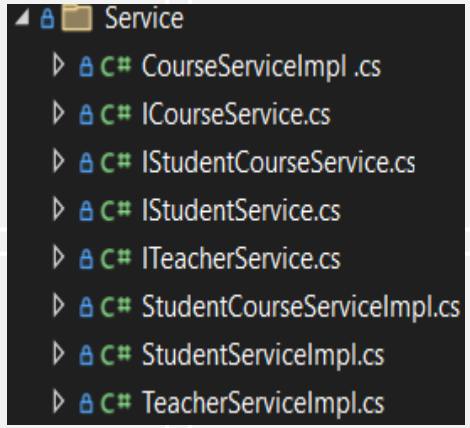
```
SevStudentsApp
namespace SevCoursesApp.DTO
{
    16 references
    public class CourseDTO
    {
        5 references
        public int Id { get; set; }
        7 references
        public string? Description { get; set;
        6 references
        public int TeacherId { get; set; }
    }
}
```

StudentCourseDTO.cs

```
SevStudentsApp
namespace SevStudentCoursesApp.DTO
{

    public class StudentCourseDTO
    {

        public int StudentId { get; set; }

        public int CourseId { get; set; }
    }
}
```

SERVICE LAYER

Το Service Layer είναι το Public API μας. Περιέχεις τα interfaces και τα implementations για κάθε μας πίνακα κάνωντας dependecy injection τις methods του καλώντας το DAO



IStudentService.cs

```
using SevStudentsApp.DTO;
using SevStudentsApp.Models;

namespace SevStudentsApp.Service
{
    public interface IStudentService
    {
        List<Student> GetAllStudents();
        void InsertStudent(StudentDTO? dto);
        void UpdateStudent(StudentDTO? dto);
        2 references
        Student? DeleteStudent(StudentDTO? dto);
        2 references
        Student? GetStudent(int id);


    }
}
```

ITeacherService.cs

```
using SevTeachersApp.DTO;
using SevTeachersApp.Models;

namespace SevTeachersApp.Service
{
    5 references
    public interface ITeacherService
    {
        2 references
        List<Teacher> GetAllTeachers();
        2 references
        void InsertTeacher(TeacherDTO? dto);
        2 references
        void UpdateTeacher(TeacherDTO? dto);
        2 references
        Teacher? DeleteTeacher(TeacherDTO? dto);
        2 references
        Teacher? GetTeacher(int id);


    }
}
```

ICourseService.cs

```
using SevCoursesApp.DTO;
using SevCoursesApp.Models;

namespace SevCoursesApp.Service
{
    public interface ICourseService
    {
        List<Course> GetAllCourses();
        void InsertCourse(CourseDTO? dto);
        void UpdateCourse(CourseDTO? dto);
        Course? DeleteCourse(CourseDTO? dto);
        Course? GetCourse(int id);


    }
}
```

IStudentCourseService.cs

```
using SevStudentCoursesApp.DTO;
using SevStudentCoursesApp.Models;

namespace SevStudentCoursesApp.Service
{
    public interface IStudentCourseService
    {
        List<StudentCourse> GetAllStudentCourses();
        void InsertStudentCourse(StudentCourseDTO? dto);
        void UpdateStudentCourse(StudentCourseDTO? dto);
        StudentCourse? DeleteStudentCourse(StudentCourseDTO? dto);
        StudentCourse? GetStudentCourse(int student_id);


    }
}
```

DEPENDECY INJECTIONS

To Dependency Injection κάθε πίνακα

StudentServiceImpl.cs

SevStudentsApp — SevStudentsApp.Servic

```
1  using SevStudentsApp.DAO;
2  using SevStudentsApp.DTO;
3  using SevStudentsApp.Models;
4
5  namespace SevStudentsApp.Service
6  {
7      public class StudentServiceImpl : IStudentService
8      {
9          private readonly IStudentDAO dao;
10
11         public StudentServiceImpl(IStudentDAO dao)
12         {
13             this.dao = dao;
14         }
```

TeacherServiceImpl.cs

SevStudentsApp — SevTeachersApp.Servic

```
1  using SevTeachersApp.DAO;
2  using SevTeachersApp.DTO;
3  using SevTeachersApp.Models;
4
5  namespace SevTeachersApp.Service
6  {
       5 references
7      public class TeacherServiceImpl : ITeacherService
8      {
9          private readonly ITeacherDAO dao;
10
           4 references
11         public TeacherServiceImpl(ITeacherDAO dao)
12         {
13             this.dao = dao;
14         }
```

CourseServiceImpl .cs | TeacherServiceImpl.cs

SevStudentsApp — SevCoursesApp.Servi

```
1  using SevCoursesApp.DAO;
2  using SevCoursesApp.DTO;
3  using SevCoursesApp.Models;
4
5  namespace SevCoursesApp.Service
6  {
7      public class CourseServiceImpl : ICourseService
8      {
9          private readonly ICourseDAO dao;
10
11         public CourseServiceImpl(ICourseDAO dao)
12         {
13             this.dao = dao;
14         }
```

StudentCourseServiceImpl.cs

SevStudentsApp — SevStudentCoursesA

```
1  using SevStudentCoursesApp.DAO;
2  using SevStudentCoursesApp.DTO;
3  using SevStudentCoursesApp.Models;
4
5  namespace SevStudentCoursesApp.Service
6  {
       5 references
7      public class StudentCourseServiceImpl : IStudent
8      {
9          private readonly IStudentCourseDAO dao;
10
           4 references
11         public StudentCourseServiceImpl(IStudentCou
12         {
13             this.dao = dao;
14         }
```

SERVICE LAYER INSERT METHOD

StudentServiceImpl.cs

SevStudentsApp

```
2 references
public Student? GetStudent(int id)...

2 references
public void InsertStudent(StudentDTO? dto)
{
        if (dto is null) return;

        try
        {
        Student? student = Convert(dto);
            dao.Insert(student);
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
            throw;
        }
}
```

TeacherServiceImpl.cs

SevStudentsApp

```
2 references
public void InsertTeacher(TeacherDTO? dto)
{
    if (dto is null) return;

    try
    {
        Teacher? teacher = Convert(dto);
        dao.Insert(teacher);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

CourseServiceImpl .cs

SevStudentsApp

```
2 references
public void InsertCourse(CourseDTO? dto)
{
    if (dto is null) return;

    try
    {
        Course? course = Convert(dto);
        dao.Insert(course);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

StudentCourseServiceImpl.cs

SevStudentsApp

```
public void InsertStudentCourse(StudentCourseDTO? dto)
{
    if (dto is null) return;

    try
    {
        StudentCourse? studentcourse = Convert(dto);
        dao.Insert(studentcourse);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

SERVICE LAYER GET METHOD

StudentServiceImpl.cs

SevStudentsApp — SevStudentsApp.Service.StudentServ

```
44  public Student? GetStudent(int id)
45  {
46      try
47      {
48          return dao.GetStudent(id);
49      } catch (Exception e)
50      {
51          Console.WriteLine(e.Message);
52          throw;
53      }
54  }
```

TeacherServiceImpl.cs

SevStudentsApp — SevTeachersApp.Service.TeacherServ

```
45  public Teacher? GetTeacher(int id)
46  {
47      try
48      {
49          return dao.GetTeacher(id);
50      }
51      catch (Exception e)
52      {
53          Console.WriteLine(e.Message);
54          throw;
55      }
56  }
```

CourseServiceImpl.cs

SevStudentsApp — SevCoursesApp.Service.CourseServic

```
45  public Course? GetCourse(int id)
46  {
47      try
48      {
49          return dao.GetCourse(id);
50      }
51      catch (Exception e)
52      {
53          Console.WriteLine(e.Message);
54          throw;
55      }
56  }
```

StudentCourseServiceImpl.cs

SevStudentsApp — SevStudentCoursesApp.Service.Stud

```
45  public StudentCourse? GetStudentCourse(int student_id)
46  {
47      try
48      {
49          return dao.GetStudentCourse(student_id);
50      }
51      catch (Exception e)
52      {
53          Console.WriteLine(e.Message);
54          throw;
55      }
56  }
```

SERVICE LAYER GET ALL METHOD

StudentServiceImpl.cs

SevStudentsApp

```
32  public List<Student> GetAllStudents()
33  {
34      try
35      {
36          return dao.GetAll();
37      } catch (Exception e)
38      {
39          Console.WriteLine(e.Message);
40          return new List<Student>();
41      }
42  }
```

TeacherServiceImpl.cs

SevStudentsApp

```
32  public List<Teacher> GetAllTeachers()
33  {
34      try
35      {
36          return dao.GetAll();
37      }
38      catch (Exception e)
39      {
40          Console.WriteLine(e.Message);
41          return new List<Teacher>();
42      }
43  }
```

CourseServiceImpl .cs

SevStudentsApp

```
32  public List<Course> GetAllCourses()
33  {
34      try
35      {
36          return dao.GetAll();
37      }
38      catch (Exception e)
39      {
40          Console.WriteLine(e.Message);
41          return new List<Course>();
42      }
43  }
```

StudentCourseServiceImpl.cs

SevStudentsApp

```
2 references
32  public List<StudentCourse> GetAllStudentCourses()
33  {
34      try
35      {
36          return dao.GetAll();
37      }
38      catch (Exception e)
39      {
40          Console.WriteLine(e.Message);
41          return new List<StudentCourse>();
42      }
43  }
```

SERVICE LAYER UPDATE METHOD

StudentServiceImpl.cs

SevStudentsApp — SevStudentsApp.Se

```
2 references
85      public void UpdateStudent(StudentDTO? dto)
86      {
87          if (dto is null) return;
88
89          try
90          {
91              Student? student = Convert(dto);
92              dao.Update(student);
93          }
94          catch (Exception e)
95          {
96              Console.WriteLine(e.Message);
97              throw;
98          }
99      }
100 }
101 }
```

TeacherServiceImpl.cs

SevStudentsApp — SevTeachersApp.Se

```
87      public void UpdateTeacher(TeacherDTO? dto)
88      {
89          if (dto is null) return;
90
91          try
92          {
93              Teacher? teacher = Convert(dto);
94              dao.Update(teacher);
95          }
96          catch (Exception e)
97          {
98              Console.WriteLine(e.Message);
99              throw;
100         }
101     }
102 }
103 }
```

CourseServiceImpl .cs

SevStudentsApp — SevCoursesApp.

```
2 references
87      public void UpdateCourse(CourseDTO? dto)
88      {
89          if (dto is null) return;
90
91          try
92          {
93              Course? course = Convert(dto);
94              dao.Update(course);
95          }
96          catch (Exception e)
97          {
98              Console.WriteLine(e.Message);
99              throw;
100         }
101     }
102 }
103 }
```

StudentCourseServiceImpl.cs

SevStudentsApp — SevStudentCoursesApp.Servi

```
2 references
32      public List<StudentCourse> GetAllStudentCourses()
33      {
34          try
35          {
36              return dao.GetAll();
37          }
38          catch (Exception e)
39          {
40              Console.WriteLine(e.Message);
41              return new List<StudentCourse>();
42          }
43      }
```

SERVICE LAYER DELETE METHOD

StudentServiceImpl.cs

SevStudentsApp | SevStudentsApp.Service

```
2 references
public Student? DeleteStudent(StudentDTO? dto)
{
    if (dto is null) return null;

    try
    {
        Student? student = Convert(dto);
        return dao.Delete(student);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

TeacherServiceImpl.cs

SevStudentsApp | SevTeachersApp.Service

```
public Teacher? DeleteTeacher(TeacherDTO? dto)
{
    if (dto is null) return null;

    try
    {
        Teacher? teacher = Convert(dto);
        return dao.Delete(teacher);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

CourseServiceImpl .cs

SevStudentsApp | SevCoursesApp.Service.(

```
public Course? DeleteCourse(CourseDTO? dto)
{
    if (dto is null) return null;

    try
    {
        Course? course = Convert(dto);
        return dao.Delete(course);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

StudentCourseServiceImpl.cs

SevStudentsApp | SevStudentCoursesApp.S

```
public StudentCourse? DeleteStudentCourse(Stud
{
    if (dto is null) return null;

    try
    {
        StudentCourse? studentcourse = Convert
        return dao.Delete(studentcourse);
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

SERVICE LAYER CONVERT METHOD

StudentServiceImpl.cs

SevStudentsApp

```
3 references
private Student? Convert(StudentDTO dto)
{

    if (dto == null) return null;

    return new Student()
    {
        Id = dto.Id,
        Firstname = dto.Firstname,
        Lastname = dto.Lastname,
    };
}
```

TeacherServiceImpl.cs

SevStudentsApp

```
3 references
private Teacher? Convert(TeacherDTO dto)
{

    if (dto == null) return null;

    return new Teacher()
    {
        Id = dto.Id,
        Firstname = dto.Firstname,
        Lastname = dto.Lastname,
    };
}
```

CourseServiceImpl .cs

SevStudentsApp

```
private Course? Convert(CourseDTO dto)
{

    if (dto == null) return null;

    return new Course()
    {
        Id = dto.Id,
        Description = dto.Description,
        TeacherId = dto.TeacherId,
    };
}
```

StudentCourseServiceImpl.cs | CourseServiceImpl .cs

SevStudentsApp

```
private StudentCourse? Convert(StudentCourseDTO dto)
{

    if (dto == null) return null;

    return new StudentCourse()
    {
        StudentId = dto.StudentId,
        CourseId = dto.CourseId,
    };
}
```

VALIDATOR

StudentValidator.cs

SevStudentsApp — SevStudentsApp.Validator.StudentValidator

```
using SevStudentsApp.DTO;

namespace SevStudentsApp.Validator
{

    public class StudentValidator
    {
        //No instances of this class should be available

        private StudentValidator() { }


        public static string Validate(StudentDTO? dto)
        {
            if ((dto!.Firstname!.Length <= 2) ||
                (dto!.Lastname!.Length <= 2))
            {
                return "Firstname or Lastname should not be less than three characters";
            }

            return "";
        }
    }
}
```

TeacherValidator.cs

SevStudentsApp — SevTeachersApp.Validator.TeacherValidator

```
using SevTeachersApp.DTO;

namespace SevTeachersApp.Validator
{
    3 references
    public class TeacherValidator
    {
        //No instances of this class should be available
        0 references
        private TeacherValidator() { }

        2 references
        public static string Validate(TeacherDTO? dto)
        {
            if ((dto!.Firstname!.Length <= 2) ||
                (dto!.Lastname!.Length <= 2))
            {
                return "Firstname or Lastname should not be less than three characters";
            }

            return "";
        }
    }
}
```

CourseValidator.cs

SevStudentsApp — SevCoursesApp.Validator.CourseValidator

```
using SevCoursesApp.DTO;

namespace SevCoursesApp.Validator
{

    public class CourseValidator
    {
        //No instances of this class should be available

        private CourseValidator() { }


        public static string Validate(CourseDTO? dto)
        {
            if ((dto!.Description!.Length <= 1))
            {
                return "Description should not be less than two characters";
            }

            return "";
        }
    }
}
```

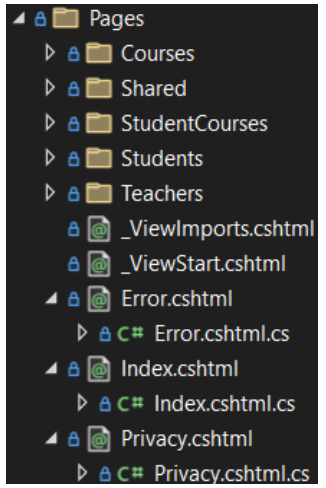
StudentCourseValidator.cs

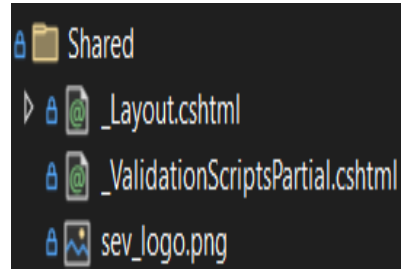
SevStudentsApp — SevStudentCoursesApp.Validator.StudentCourseValidato

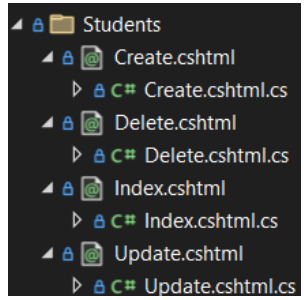
```
using SevStudentCoursesApp.DTO;

namespace SevStudentCoursesApp.Validator
{

    public class StudentCourseValidator
    {
        //No instances of this class should be available
        0 references
        private StudentCourseValidator() { }

        2 references
        public static string Validate(StudentCourseDTO? dto)
        {
            if ((dto!.StudentId!.ToString().Length <= 1))
            {
                return "Student id should not be less than two digits";
            }

            return "";
        }
    }
}
```

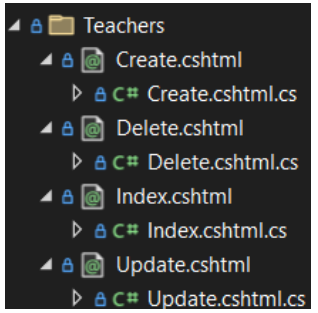
RAZOR PAGES
(CONTROLLER AND VIEW LAYERS)

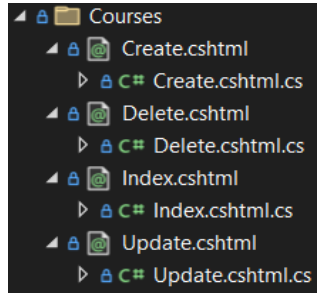
Φτιάχνουμε 4 φακέλους 1 για κάθε πίνακα και για τον κάθε φάκελο πατάμε δεξί κλικ στον φάκελο και new Razor Page. Razor Page – Empty και ξανά Razor Page empty και το κάνουμε αυτό 4 φορές για το index.cshtml, create.cshtml, update.cshtml, delete.cshtml

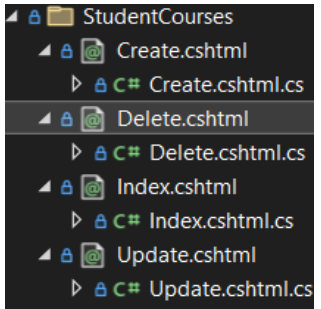












STUDENT RAZOR PAGES(INDEX, CREATE) (CONTROLLER AND VIEW LAYERS)

```cs
namespace SevStudentsApp.Pages.Students
{
    6 references
    public class IndexModel : PageModel
    {

        private readonly IStudentDAO studentDAO= new StudentDAOImpl();
        private readonly IStudentService? service;

        internal List<Student> students = new();

        0 references
        public IndexModel()
        {
            service = new StudentServiceImpl(studentDAO);
        }

        0 references
        public IActionResult OnGet()
        {
            students = service!.GetAllStudents();
            return Page();
        }
    }
}
```

```cshtml
@page
@model SevStudentsApp.Pages.Students.IndexModel
@{
    ViewData["Title"] = "Student Index";
}

<h2>List of Students</h2>

<a class="btn btn-primary btn-sm" href="/Students/Create">New Student</a>
<div style="background: #611ff0">
<table class="table table-bordered table-hover text-white">
    <thead>
        <tr>
            <th>ID</th>
            <th>Firstname</th>
            <th>Lastname</th></tr>
        </thead>
        <tbody>
            @if (Model.students is not null)
            {
                @foreach (var student in Model.students)
                {
                    <tr>
                        <td>@student.Id</td>
                        <td>@student.Firstname</td>
                        <td>@student.Lastname</td>
                        <td>
                                <a class="btn btn-outline-light btn-sm"
                                href="/Students/Update?id=@student.Id">Update</a>
                                 <a class="btn btn-outline-light btn-sm"
                                href="/Students/Delete?id=@student.Id">Delete</a>
                        </td>
                    </tr>
                }
            }

        </tbody>

    </table>
    </div>
```

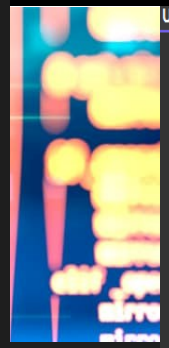
Create.cshtml.cs

```cs
namespace SevStudentsApp.Pages.Students
{
    6 references
    public class CreateModel : PageModel
    {
        private readonly IStudentDAO studentDAO = new StudentDAOImpl();
        private readonly IStudentService service;

        0 references
        public CreateModel()
        {
            service = new StudentServiceImpl(studentDAO);
        }

        internal StudentDTO studentDto = new();
        internal string errorMessage = "";

        0 references
        public void OnGet()
        {
        }

        0 references
        public void OnPost()
        {
            studentDto.Firstname = Request.Form["firstname"];
            studentDto.Lastname = Request.Form["lastname"];

            errorMessage = StudentValidator.Validate(studentDto);

            if (!errorMessage.Equals("")) return;

            try
            {
                service.InsertStudent(studentDto);
                Response.Redirect("/Students/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
}
```

Create.cshtml

```cshtml
@page
@model SevStudentsApp.Pages.Students.CreateModel
@{
    ViewData["Title"] = "Student Create";

}

<h2>New Student</h2>
@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <label for="firstname" class="col-md-1 col-form-label">First Name</label>
        <div>
            <input type="text" class="form-control"
                   name="firstname" id="firstname" placeholder="Enter your first name"
                   value="@Model.studentDto.Firstname" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="lastname" class="col-md-1 col-form-label">Last Name</label>
        <div>
            <input type="text" class="form-control"
                   name="lastname" id="lastname" placeholder="Enter your last name"
                   value="@Model.studentDto.Lastname" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="col-md-1 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="col-md-1 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

STUDENT RAZOR PAGES(UPDATE, DELETE) (CONTROLLER AND VIEW LAYERS)



Update.cshtml.cs

```
SevStudentsApp
SevStudentsApp.Pages.Students.UpdateModel
namespace SevStudentsApp.Pages.Students
{
    6 references
    public class UpdateModel : PageModel
    {

        private readonly IStudentDAO studentDAO = new StudentDAOImpl();
        private readonly IStudentService service;


        0 references
        public UpdateModel()
        {
            service = new StudentServiceImpl(studentDAO);
        }

        internal StudentDTO studentDto = new();
        internal string errorMessage = "";


        0 references
        public void OnGet()
        {
            try
            {
                Student? student;

                int id = int.Parse(Request.Query["id"]);
                student = service.GetStudent(id);

                if (student != null)
                {
                    studentDto = ConvertToDto(student);
                }
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                    return;
                }
            }

            0 references
            public void OnPost()
            {
                errorMessage = "";
                // Get DTO
                studentDto.Id = int.Parse(Request.Form["id"]);
                studentDto.Firstname = Request.Form["firstname"];
                studentDto.Lastname = Request.Form["lastname"];

                // validate
                errorMessage = StudentValidator.Validate(studentDto);

                if (!errorMessage.Equals("")) return;

                try
                {
                    service.UpdateStudent(studentDto);
                    Response.Redirect("/Students/Index");
                }
                catch (Exception e)
                {
                    errorMessage = e.Message;
                    return;
                }
            }


            1 reference
            private StudentDTO ConvertToDto(Student student)
            {
                return new StudentDTO()
                {
                    Id = student.Id,
                    Firstname = student.Firstname,
                    Lastname = student.Lastname
                };
            }
        }
    }
}
```

Update.cshtml

```
@page
@model SevStudentsApp.Pages.Students.UpdateModel
@{
    ViewData["Title"] = "Student Update";
}

<h2>New Student</h2>
@if (!Model.errorMessage.Equals(""))
{
        <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <input type="hidden" class="form-control" name="id"
                value="@Model.studentDto.Id" />

    </div>
    <div class="row mb-3">
        <label for="firstname" class="col-md-1 col-form-label">Όνομα</label>
        <div>
            <input type="text" class="form-control"
                name="firstname" id="firstname" placeholder="Enter firstname"
                value="@Model.studentDto.Firstname" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="lastname" class="col-md-1 col-form-label">Επώνυμο</label>
        <div>
            <input type="text" class="form-control"
                name="lastname" id="lastname" placeholder="Enter lastname"
                value="@Model.studentDto.Lastname" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="offset-md-1 col-md-3 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="offset-md-1 col-md-3 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

Delete.cshtml

```
@page
@model SevStudentsApp.Pages.Students.DeleteModel
@{
    ViewData["Title"] = "Student Delete";

}

@if (!Model.errorMessage.Equals(""))
{
        <h2><strong>@Model.errorMessage</strong></h2>
}
```

Delete.cshtml.cs

```
namespace SevStudentsApp.Pages.Students
{
    6 references
    public class DeleteModel : PageModel
    {
        private readonly IStudentDAO studentDAO = new StudentDAOImpl();
        private readonly IStudentService? service;

        internal string errorMessage = "";

        0 references
        public DeleteModel()
        {
            service = new StudentServiceImpl(studentDAO);
        }

        0 references
        public void OnGet()
        {
            StudentDTO studentDTO = new();

            try
            {
                Student? student;

                int id = int.Parse(Request.Query["id"]);

                studentDTO.Id = id;
                student = service?.DeleteStudent(studentDTO);
                Response.Redirect("/Students/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
}
```

TEACHER RAZOR PAGES(INDEX, CREATE) (CONTROLLER AND VIEW LAYERS)

Index.cshtml.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using SevTeachersApp.DAO;
using SevTeachersApp.Models;
using SevTeachersApp.Service;

namespace SevTeachersApp.Pages.Teachers
{
    6 references
    public class IndexModel : PageModel
    {

        private readonly ITeacherDAO teacherDAO= new TeacherDAOImpl();
        private readonly ITeacherService? service;

        internal List<Teacher> teachers = new();

        0 references
        public IndexModel()
        {
            service = new TeacherServiceImpl(teacherDAO);
        }

        0 references
        public IActionResult OnGet()
        {
            teachers = service!.GetAllTeachers();
            return Page();
        }
    }
}
```

Create.cshtml.cs

```
namespace SevTeachersApp.Pages.Teachers
{
    6 references
    public class CreateModel : PageModel
    {
        private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
        private readonly ITeacherService service;


        0 references
        public CreateModel()
        {
            service = new TeacherServiceImpl(teacherDAO);
        }

        internal TeacherDTO teacherDto = new();
        internal string errorMessage = "";

        0 references
        public void OnGet()
        {
        }

        0 references
        public void OnPost()
        {
            teacherDto.Firstname = Request.Form["firstname"];
            teacherDto.Lastname = Request.Form["lastname"];

            errorMessage = TeacherValidator.Validate(teacherDto);

            if (!errorMessage.Equals("")) return;

            try
            {
                service.InsertTeacher(teacherDto);
                Response.Redirect("/Teachers/Index");

                catch (Exception e)
                {
                    errorMessage = e.Message;
                    return;
                }
            }
        }
    }
}
```

Index.cshtml

```
@page
@model SevTeachersApp.Pages.Teachers.IndexModel
@{
    ViewData["Title"] = "Teacher Index";

}

<h2>List of Teachers</h2>
<a class="btn btn-primary btn-sm" href="/Teachers/Create">New Teacher</a>
<div style="background: #611ff0">
<table class="table table-bordered table-hover text-white">
    <thead>
        <tr>
            <th>ID</th>
            <th>Firstname</th>
            <th>Lastname</th></tr>
        </thead>
        <tbody>
            @if (Model.teachers is not null)
            {
                @foreach (var teacher in Model.teachers)
                {
                    <tr>
                        <td>@teacher.Id</td>
                        <td>@teacher.Firstname</td>
                        <td>@teacher.Lastname</td>
                        <td>
                                <a class="btn btn-outline-light btn-sm"
                                href="/Teachers/Update?id=@teacher.Id">Update</a>
                                 <a class="btn btn-outline-light btn-sm"
                                href="/Teachers/Delete?id=@teacher.Id">Delete</a>
                        </td>
                    </tr>
                }

        </tbody>

    </table>
    </div>
```

Create.cshtml

```
@page
@model SevTeachersApp.Pages.Teachers.CreateModel
@{
    ViewData["Title"] = "Teacher Create";

}

<h2>New Teacher</h2>
@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <label for="firstname" class="col-md-1 col-form-label">First Name</label>
        <div>
            <input type="text" class="form-control"
                   name="firstname" id="firstname" placeholder="Enter your first name"
                   value="@Model.teacherDto.Firstname" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="lastname" class="col-md-1 col-form-label">Last Name</label>
        <div>
            <input type="text" class="form-control"
                   name="lastname" id="lastname" placeholder="Enter your last name"
                   value="@Model.teacherDto.Lastname" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="col-md-1 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="col-md-1 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

TEACHER RAZOR PAGES(UPDATE, DELETE) (CONTROLLER AND VIEW LAYERS)

Update.cshtml.cs

```
namespace SevTeachersApp.Pages.Teachers
{
    6 references
    public class UpdateModel : PageModel
    {

        private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
        private readonly ITeacherService service;


        0 references
        public UpdateModel()
        {
            service = new TeacherServiceImpl(teacherDAO);
        }

        internal TeacherDTO teacherDto = new();
        internal string errorMessage = "";


        0 references
        public void OnGet()
        {
            try
            {
                Teacher? teacher;

                int id = int.Parse(Request.Query["id"]);
                teacher = service.GetTeacher(id);

                if (teacher != null)
                {
                    teacherDto = ConvertToDto(teacher);
                }
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }

        0 references
        public void OnPost()
        {
            errorMessage = "";
            // Get DTO
            teacherDto.Id = int.Parse(Request.Form["id"]);
            teacherDto.Firstname = Request.Form["firstname"];
            teacherDto.Lastname = Request.Form["lastname"];

            // validate
            errorMessage = TeacherValidator.Validate(teacherDto);

            if (!errorMessage.Equals("")) return;

            try
            {
                service.UpdateTeacher(teacherDto);
                Response.Redirect("/Teachers/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }


        1 reference
        private TeacherDTO ConvertToDto(Teacher teacher)
        {
            return new TeacherDTO()
            {
                Id = teacher.Id,
                Firstname = teacher.Firstname,
                Lastname = teacher.Lastname
            };
        }
    }
}
```

Update.cshtml

```
@page
@model SevTeachersApp.Pages.Teachers.UpdateModel
@{
    ViewData["Title"] = "Teacher Update";
}

<h2>New Teacher</h2>
@if (!Model.errorMessage.Equals(""))
{
        <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <input type="hidden" class="form-control" name="id"
                value="@Model.teacherDto.Id" />

    </div>
    <div class="row mb-3">
        <label for="firstname" class="col-md-1 col-form-label">Όνομα</label>
        <div>
            <input type="text" class="form-control"
                name="firstname" id="firstname" placeholder="Enter firstname"
                value="@Model.teacherDto.Firstname" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="lastname" class="col-md-1 col-form-label">Επώνυμο</label>
        <div>
            <input type="text" class="form-control"
                name="lastname" id="lastname" placeholder="Enter lastname"
                value="@Model.teacherDto.Lastname" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="offset-md-1 col-md-3 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="offset-md-1 col-md-3 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

Delete.cshtml

```
@page
@model SevTeachersApp.Pages.Teachers.DeleteModel
@{
    ViewData["Title"] = "Teacher Delete";

}
@if (!Model.errorMessage.Equals(""))
{
        <h2><strong>@Model.errorMessage</strong></h2>
}
```

Delete.cshtml.cs

```
namespace SevTeachersApp.Pages.Teachers
{
    6 references
    public class DeleteModel : PageModel
    {
        private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
        private readonly ITeacherService? service;

        internal string errorMessage = "";

        0 references
        public DeleteModel()
        {
            service = new TeacherServiceImpl(teacherDAO);
        }

        0 references
        public void OnGet()
        {
            TeacherDTO teacherDTO = new();

            try
            {
                Teacher? teacher;

                int id = int.Parse(Request.Query["id"]);

                teacherDTO.Id = id;
                teacher = service?.DeleteTeacher(teacherDTO);
                Response.Redirect("/Teachers/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
}
```

COURSE RAZOR PAGES(INDEX, CREATE (CONTROLLER AND VIEW LAYERS)

Index.cshtml.cs

SevStudentsApp — SevCoursesApp.Pages.Courses.IndexModel

```
namespace SevCoursesApp.Pages.Courses
{
    6 references
    public class IndexModel : PageModel
    {

        private readonly ICourseDAO courseDAO= new CourseDAOImpl();
        private readonly ICourseService? service;

        internal List<Course> courses = new();

        0 references
        public IndexModel()
        {
            service = new CourseServiceImpl(courseDAO);
        }

        0 references
        public IActionResult OnGet()
        {
            courses = service!.GetAllCourses();
            return Page();
        }
    }
}
```

Index.cshtml

```
@page
@model SevCoursesApp.Pages.Courses.IndexModel
@{
    ViewData["Title"] = "Course Index";
}

<h2>List of Courses</h2>
<a class="btn btn-primary btn-sm" href="/Courses/Create">New Course</a>
<div style="background:#611ff0">
<table class="table table-bordered table-hover text-white">
    <thead>
        <tr>
            <th>ID</th>
            <th>Description</th>
            <th>Teacher ID</th></tr>
        </thead>
        <tbody>
            @if (Model.courses is not null)
            {
                @foreach (var course in Model.courses)
                {
                    <tr>
                        <td>@course.Id</td>
                        <td>@course.Description</td>
                        <td>@course.TeacherId</td>
                        <td>
                                <a class="btn btn-outline-light btn-sm"
                                href="/Courses/Update?id=@course.Id">Update</a>
                                <a class="btn btn-outline-light btn-sm"
                                href="/Courses/Delete?id=@course.Id">Delete</a>
                        </td>
                    </tr>
                }
            </tbody>
    </table>
    </div>
```

Create.cshtml.cs

SevStudentsApp — SevCoursesApp.Pages.Courses.CreateMode

```
namespace SevCoursesApp.Pages.Courses
{
    6 references
    public class CreateModel : PageModel
    {
        private readonly ICourseDAO courseDAO = new CourseDAOImpl()
        private readonly ICourseService service;


        0 references
        public CreateModel()
        {
            service = new CourseServiceImpl(courseDAO);
        }

        internal CourseDTO courseDto = new();
        internal string errorMessage = "";

        0 references
        public void OnGet()
        {
        }

        0 references
        public void OnPost()
        {
            courseDto.Description = Request.Form["description"];
            courseDto.TeacherId = int.Parse(Request.Form["teacherid"])

            errorMessage = CourseValidator.Validate(courseDto);

            if (!errorMessage.Equals("")) return;

            try
            {
                service.InsertCourse(courseDto);
                Response.Redirect("/Courses/Index");

            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
```

Create.cshtml

```
@page
@model SevCoursesApp.Pages.Courses.CreateModel
@{
    ViewData["Title"] = "Course Create";
}

<h2>New Course</h2>
@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <label for="description" class="col-md-1 col-form-label">Description</label>
        <div>
            <input type="text" class="form-control"
                name="description" id="description" placeholder="Enter your course description"
                value="@Model.courseDto.Description" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="teacherid" class="col-md-1 col-form-label">Teacher ID</label>
        <div>
            <input type="number" class="form-control"
                name="teacherid" id="teacherid" placeholder="Enter your Teacher Id"
                value="@Model.courseDto.TeacherId" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="col-md-1 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="col-md-1 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

COURSE RAZOR PAGES(UPDATE, DELETE) (CONTROLLER AND VIEW LAYERS)

Update.cshtml.cs

```
namespace SevCoursesApp.Pages.Courses
{
    6 references
    public class UpdateModel : PageModel
    {

        private readonly ICourseDAO courseDAO = new CourseDAOImpl();
        private readonly ICourseService service;


        0 references
        public UpdateModel()
        {
            service = new CourseServiceImpl(courseDAO);
        }

        internal CourseDTO courseDto = new();
        internal string errorMessage = "";


        0 references
        public void OnGet()
        {
            try
            {
                Course? course;
                int id = int.Parse(Request.Query["id"]);
                course = service.GetCourse(id);

                if (course != null)
                {
                    courseDto = ConvertToDto(course);
                }
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }

        0 references
        public void OnPost()
        {
            errorMessage = "";
            // Get DTO
            courseDto.Id = int.Parse(Request.Form["id"]);
            courseDto.Description = Request.Form["description"];
            courseDto.TeacherId = int.Parse(Request.Form["teacherid"]);

            // validate
            errorMessage = CourseValidator.Validate(courseDto);
            if (!errorMessage.Equals("")) return;

            try
            {
                service.UpdateCourse(courseDto);
                Response.Redirect("/Courses/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }


        1 reference
        private CourseDTO ConvertToDto(Course course)
        {
            return new CourseDTO()
            {
                Id = course.Id,
                Description = course.Description,
                TeacherId = course.TeacherId
            };
        }
    }
}
```

Update.cshtml

```
@page
@model SevCoursesApp.Pages.Courses.UpdateModel
@{
    ViewData["Title"] = "Course Update";
}

<h2>New Course</h2>
@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <input type="hidden" class="form-control" name="id"
               value="@Model.courseDto.Id" />

    </div>
    <div class="row mb-3">
        <label for="description" class="col-md-1 col-form-label">Περιγραφή</label>
        <div>
            <input type="text" class="form-control"
                   name="description" id="description" placeholder="Enter description"
                   value="@Model.courseDto.Description" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="teacherid" class="col-md-1 col-form-label">Κωδικός Καθηγητή</label>
        <div>
            <input type="number" class="form-control"
                   name="teacherid" id="teacherid" placeholder="Enter Teacher Id"
                   value="@Model.courseDto.TeacherId" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="offset-md-1 col-md-3 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="offset-md-1 col-md-3 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

Delete.cshtml

```
@page
@model SevCoursesApp.Pages.Courses.DeleteModel
@{
    ViewData["Title"] = "Course Delete";
}

@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}
```

Delete.cshtml.cs

```
namespace SevCoursesApp.Pages.Courses
{
    6 references
    public class DeleteModel : PageModel
    {
        private readonly ICourseDAO courseDAO = new CourseDAOImpl();
        private readonly ICourseService? service;

        internal string errorMessage = "";

        0 references
        public DeleteModel()
        {
            service = new CourseServiceImpl(courseDAO);
        }

        0 references
        public void OnGet()
        {
            CourseDTO courseDTO = new();

            try
            {
                Course? course;

                int id = int.Parse(Request.Query["id"]);
                courseDTO.Id = id;
                course = service?.DeleteCourse(courseDTO);
                Response.Redirect("/Courses/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
}
```

STUDENTCOURSE RAZOR PAGES(INDEX, CREATE) (CONTROLLER AND VIEW LAYERS)

IndexModel.cs

```
namespace SevStudentCoursesApp.Pages.StudentCourses
{
    6 references
    public class IndexModel : PageModel
    {

        private readonly IStudentCourseDAO studentcourseDAO= new StudentCourseDAOImpl();
        private readonly IStudentCourseService? service;

        internal List<StudentCourse> studentcourses = new();

        0 references
        public IndexModel()
        {
            service = new StudentCourseServiceImpl(studentcourseDAO);
        }

        0 references
        public IActionResult OnGet()
        {
            studentcourses = service!.GetAllStudentCourses();
            return Page();
        }
    }
}
```

Index.cshtml

```
@page
@model SevStudentCoursesApp.Pages.StudentCourses.IndexModel
@{
    ViewData["Title"] = "Student Course Index";

}

<h2>List of Student Courses</h2>
<a class="btn btn-primary btn-sm" href="/StudentCourses/Create">New Student Course</a>
<div style="background:#611ff0">
<table class="table table-bordered table-hover text-white">

    <thead>
        <tr>
            <th>Student ID</th>
            <th>Course ID</th>
        </tr>
        </thead>
        <tbody>
            @if (Model.studentcourses is not null)
            {
                @foreach (var studentcourse in Model.studentcourses)
                {
                    <tr>
                        <td>@studentcourse.StudentId</td>
                        <td>@studentcourse.CourseId</td>
                        <td>
                                                <a class="btn btn-outline-light btn-sm"
                            href="/StudentCourses/Update?student_id=@studentcourse.StudentId">Update</a>
                                    <a class="btn btn-outline-light btn-sm"
                            href="/StudentCourses/Delete?student_id=@studentcourse.StudentId">Delete</a>
                        </td>
                    </tr>
                }
            }

        </tbody>

</table>
</div>
```

CreateModel.cshtml.cs

```
namespace SevStudentCoursesApp.Pages.StudentCourses
{
    6 references
    public class CreateModel : PageModel
    {
        private readonly IStudentCourseDAO studentcourseDAO = new StudentCourseDAOImpl();
        private readonly IStudentCourseService service;


        0 references
        public CreateModel()
        {
            service = new StudentCourseServiceImpl(studentcourseDAO);
        }

        internal StudentCourseDTO studentcourseDto = new();
        internal string errorMessage = "";

        0 references
        public void OnGet()
        {
        }

        0 references
        public void OnPost()
        {
            studentcourseDto.StudentId = int.Parse(Request.Form["student_id"]);
            studentcourseDto.CourseId = int.Parse(Request.Form["course_id"]);

            errorMessage = StudentCourseValidator.Validate(studentcourseDto);

            if (!errorMessage.Equals("")) return;

            try
            {
                service.InsertStudentCourse(studentcourseDto);
                Response.Redirect("/StudentCourses/Index");

            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
}
```

Create.cshtml

```
@page
@model SevStudentCoursesApp.Pages.StudentCourses.CreateModel
@{
    ViewData["Title"] = "Student Course Create";

}

<h2>New Student Course</h2>
@if (!Model.errorMessage.Equals(""))
{
        <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <label for="student_id" class="col-md-1 col-form-label">Student ID</label>
        <div>
            <input type="number" class="form-control"
                   name="student_id" id="student_id" placeholder="Enter student id"
                   value="@Model.studentcourseDto.StudentId" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="course_id" class="col-md-1 col-form-label">Course ID</label>
        <div>
            <input type="number" class="form-control"
                   name="course_id" id="course_id" placeholder="Enter course_id"
                   value="@Model.studentcourseDto.CourseId" />
                      value="@Model.studentcourseDto.CourseId" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="col-md-1 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="col-md-1 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

STUDENTCOURSE RAZOR PAGES(UPDATE, DELETE) (CONTROLLER AND VIEW LAYERS)

Update.cshtml.cs

```
SevStudentsApp | SevStudentCoursesApp.Pages.StudentCourses.UpdateModel
namespace SevStudentCoursesApp.Pages.StudentCourses
{
    6 references
    public class UpdateModel : PageModel
    {

        private readonly IStudentCourseDAO studentcourseDAO = new StudentCourseDAOImpl();
        private readonly IStudentCourseService service;


        0 references
        public UpdateModel()
        {
            service = new StudentCourseServiceImpl(studentcourseDAO);
        }

        internal StudentCourseDTO studentcourseDto = new();
        internal string errorMessage = "";


        0 references
        public void OnGet()
        {
            try
            {
                StudentCourse? studentcourse;
                int student_id = int.Parse(Request.Query["student_id"]);
                studentcourse = service.GetStudentCourse(student_id);

                if (studentcourse != null)
                {
                    studentcourseDto = ConvertToDto(studentcourse);
                }
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }

        0 references
        public void OnPost()
        {
            errorMessage = "";
            // Get DTO
            studentcourseDto.StudentId = int.Parse(Request.Form["student_id"]);
            studentcourseDto.CourseId = int.Parse(Request.Form["course_id"]);

            // validate
            errorMessage = StudentCourseValidator.Validate(studentcourseDto);
            if (!errorMessage.Equals("")) return;

            try
            {
                service.UpdateStudentCourse(studentcourseDto);
                Response.Redirect("/StudentCourses/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }


        1 reference
        private StudentCourseDTO ConvertToDto(StudentCourse studentcourse)
        {
            return new StudentCourseDTO()
            {
                StudentId = studentcourse.StudentId,
                CourseId = studentcourse.CourseId
            };
        }
    }
}
```

Update.cshtml

```
@page
@model SevStudentCoursesApp.Pages.StudentCourses.UpdateModel
@{
    ViewData["Title"] = "Student Course Update";
}

<h2>New Student Course</h2>
@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}

<form method="POST">
    <div class="row mb-3">
        <label for="student_id" class="col-md-1 col-form-label">Κωδικός Μαθητή</label>
        <div>
            <input type="number" class="form-control"
                   name="student_id" id="student_id" placeholder="Enter Student Id"
                   value="@Model.studentcourseDto.StudentId" />
        </div>
    </div>
    <div class="row mb-3">
        <label for="course_id" class="col-md-1 col-form-label">Κωδικός Μαθήματος</label>
        <div>
            <input type="number" class="form-control"
                   name="course_id" id="course_id" placeholder="Enter Course Id"
                   value="@Model.studentcourseDto.CourseId" />
        </div>
    </div>
    <div class="row mb-3">
        <div class="offset-md-1 col-md-3 d-grid">
            <button type="submit" class="btn btn-primary">Submit</button>
        </div>
        <div class="offset-md-1 col-md-3 d-grid">
            <a href="/" role="button" class="btn btn-primary">Cancel</a>
        </div>
    </div>
</form>
```

Delete.cshtml

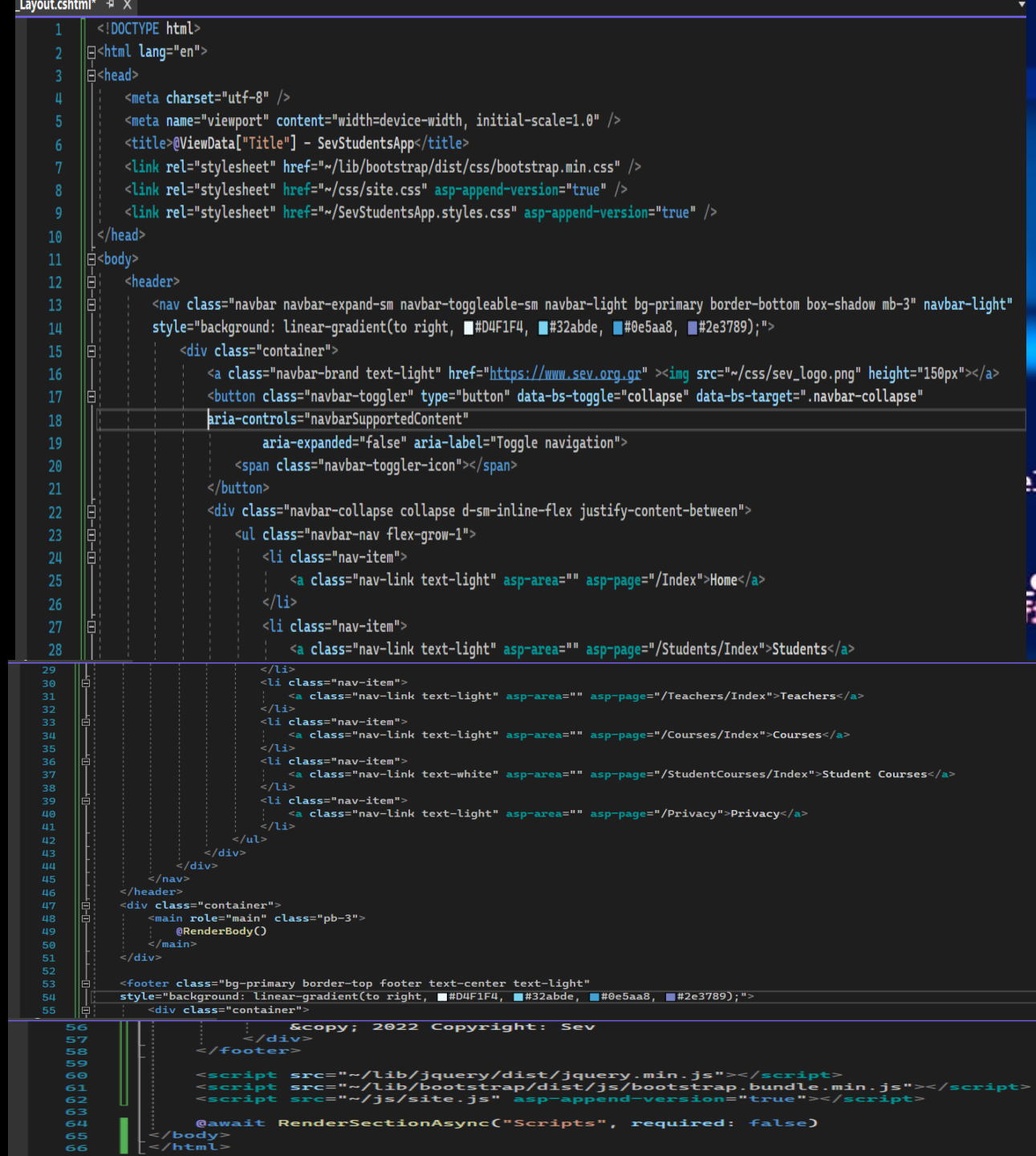
```
@page
@model SevStudentCoursesApp.Pages.StudentCourses.DeleteModel
@{
    ViewData["Title"] = "Student Course Delete";

}

@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}
```

Delete.cshtml.cs

```
SevStudentsApp | SevStudentCoursesApp.Pages.StudentCourses.DeleteModel
namespace SevStudentCoursesApp.Pages.StudentCourses
{
    6 references
    public class DeleteModel : PageModel
    {
        private readonly IStudentCourseDAO studentcourseDAO = new StudentCourseDAOImpl();
        private readonly IStude   interface SevStudentCoursesApp.DAO.IStudentCourseDAO

        internal string errorMessage = "";

        0 references
        public DeleteModel()
        {
            service = new StudentCourseServiceImpl(studentcourseDAO);
        }

        0 references
        public void OnGet()
        {
            StudentCourseDTO studentcourseDTO = new();

            try
            {
                StudentCourse? studentcourse;

                int student_id = int.Parse(Request.Query["student_id"]);
                studentcourseDTO.StudentId = student_id;
                studentcourse = service?.DeleteStudentCourse(studentcourseDTO);
                Response.Redirect("/StudentCourses/Index");
            }
            catch (Exception e)
            {
                errorMessage = e.Message;
                return;
            }
        }
    }
}
```

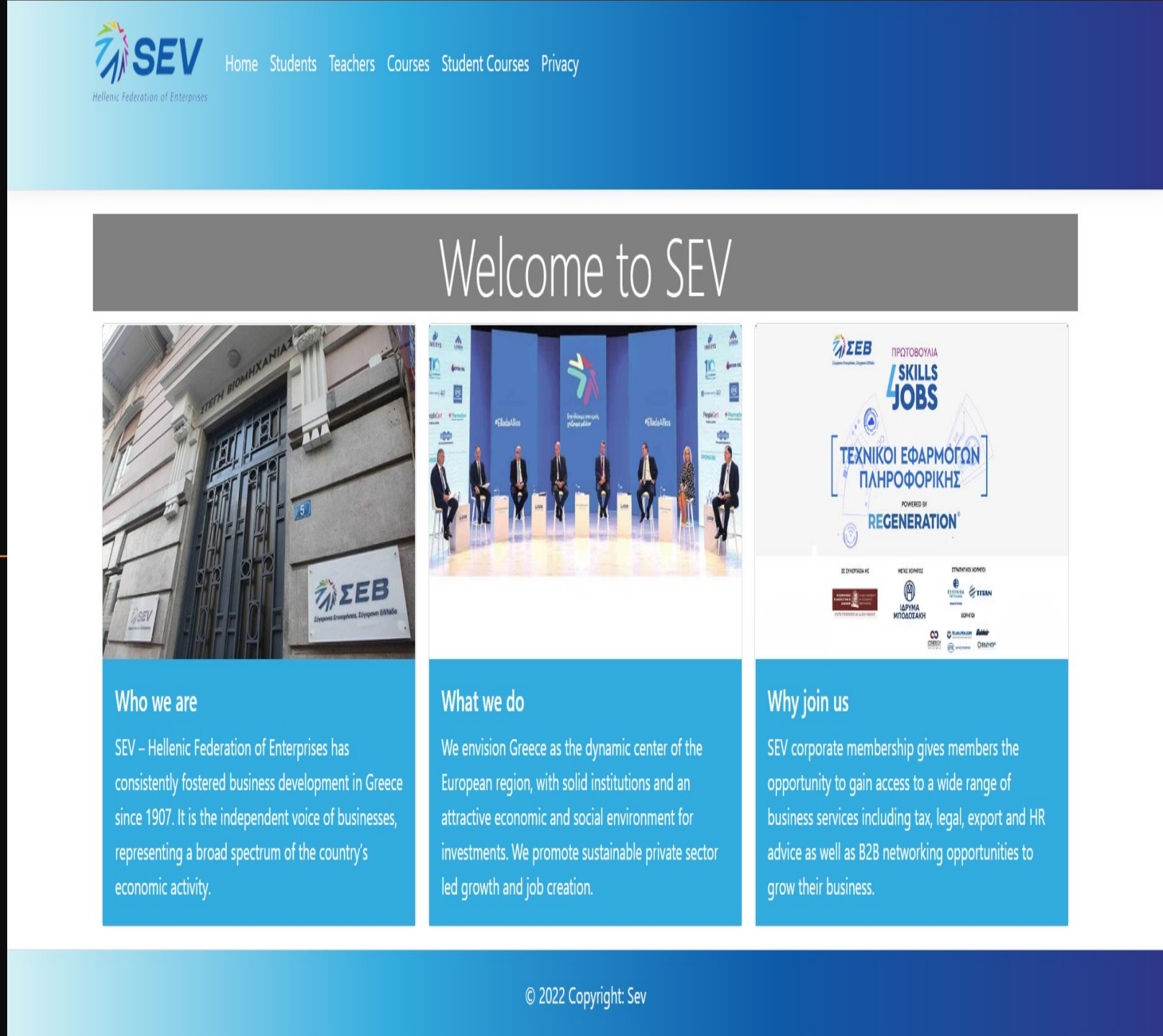
SHARED/LAYOUT

Το κοινό layout όλων των σελίδων το οποίο περιέχει το menu και το footer



ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

- Αρχική σελίδα εμφάνιση



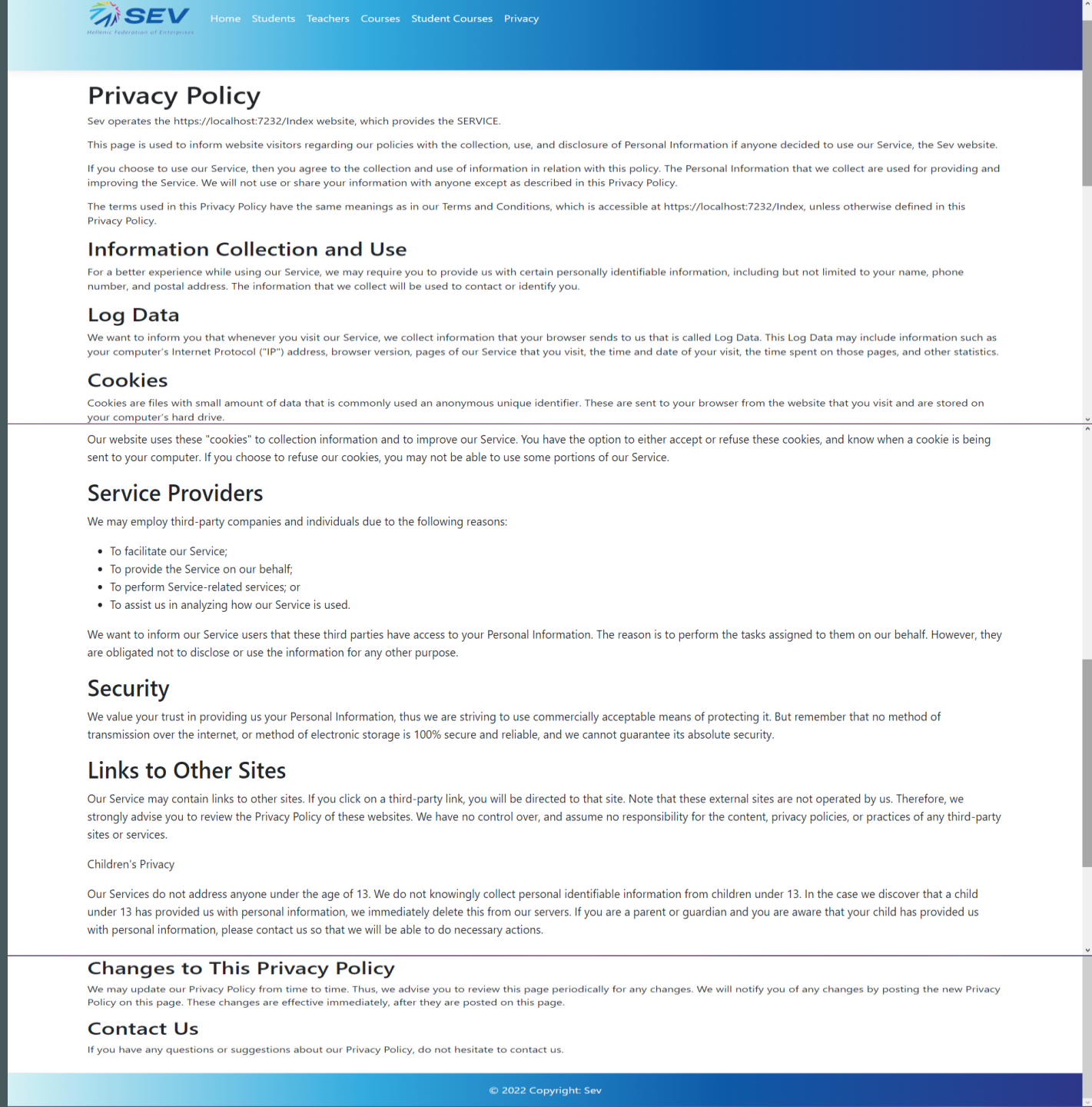
ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

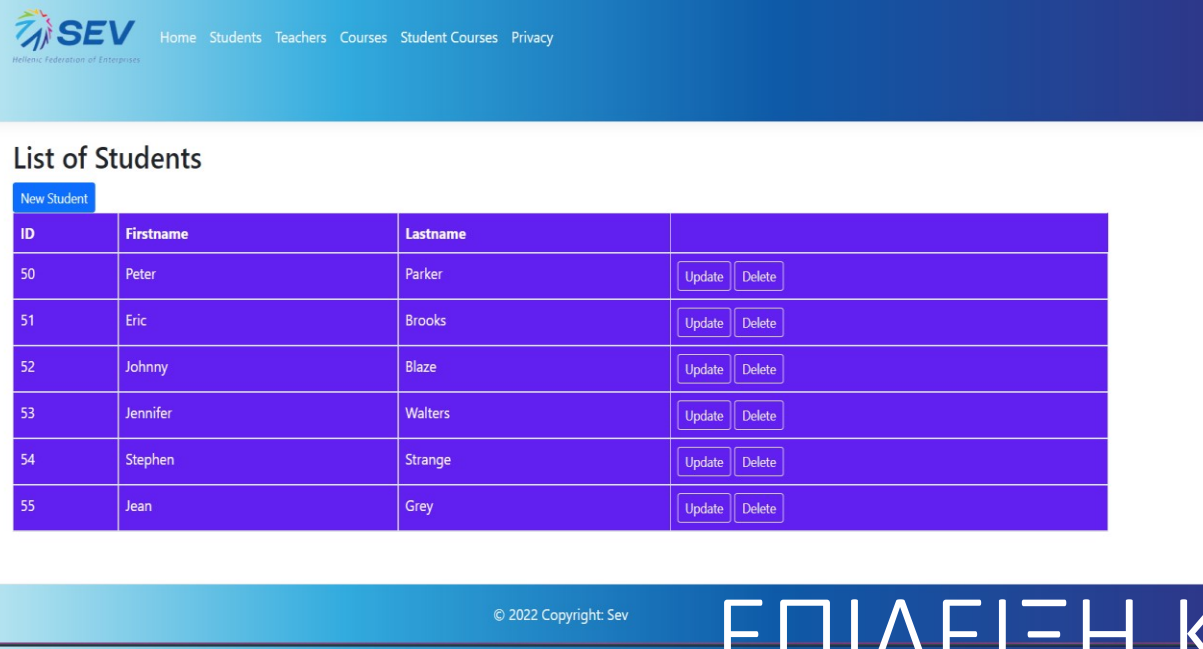
- Αρχική σελίδα κώδικας

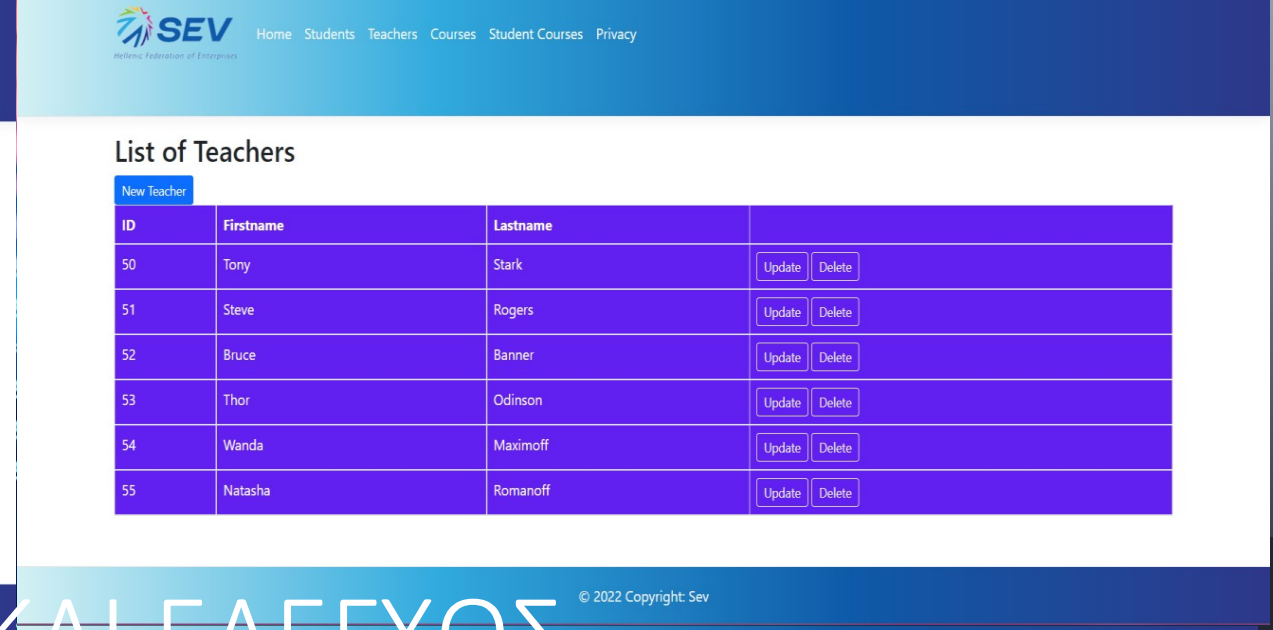
```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center text-white">
 <div style="background: #808080">
        <h1 class="display-4">Welcome to SEV</h1>
        </div>
        </div>
<div class="container">
    <div class="row g-3">
        <div class="col-12 col-md-6 col-lg-4">
            <div class="card">
                <div style="background: #32abde">
                <a href="https://en.sev.org.gr/who-we-are"><img src="~/css/sev_building.jpg" alt="Sev-Grid1" class="card-img-top"></a>
                <div class="card-body">
                    <h5 class="card-title text-white">Who we are</h5>
                    <p class="card-text text-white" >
                      SEV – Hellenic Federation of Enterprises has consistently fostered business development in Greece since 1907.
                      It is the independent voice of businesses, representing a broad spectrum of the country's economic activity.
                    </p>
                </div>
            </div>
        </div>
        </div>
        <div class="col-12 col-md-6 col-lg-4">
            <div class="card">
                <div style="background: #32abde">
                    <a href="https://en.sev.org.gr/sev-business-priorities/"><img src="~/css/sev_conference.png" alt="Sev-Grid2"
                        class="card-img-top"></a>
                <div class="card-body">
                    <h5 class="card-title text-white">What we do</h5>
                    <p class="card-text text-white">
                    We envision Greece as the dynamic center of the European region, with solid institutions and an attractive economic
                    and social environment for investments. We promote sustainable private sector led growth and job creation.
                    </p>
                </div>
            </div>
        </div>
        <div class="col-12 col-md-6 col-lg-4">
            <div class="card">
                <div style="background: #32abde">
                <a href="https://en.sev.org.gr/why-join-us"><img src="~/css/sev_skills4jobs.jpg" alt="Sev-Grid3" class="card-img-top"></a
                <div class="card-body">
                    <h5 class="card-title text-white">Why join us</h5>
                    <p class="card-text text-white">
                        SEV corporate membership gives members the opportunity to gain access to a wide range of business services
                        legal, export and HR advice as well as B2B networking opportunities to grow their business.
                    </p>
                </div>
            </div>
        </div>
    </div>
</div>
</div>
```

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

- Privacy Policy εμφάνιση

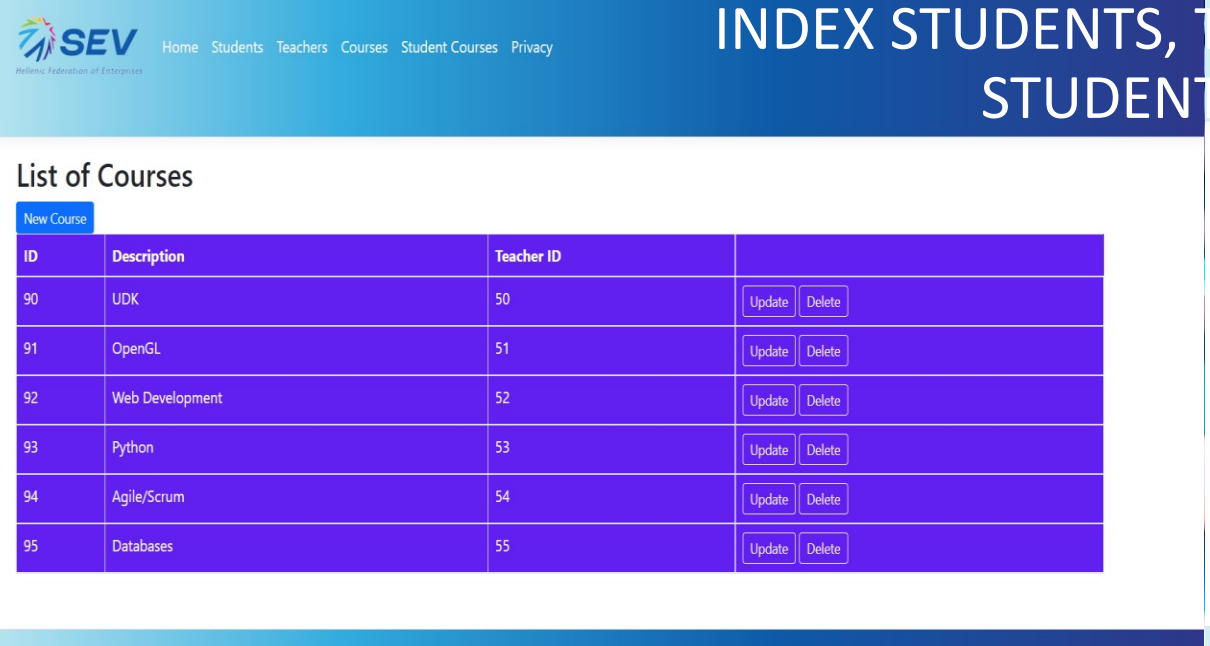


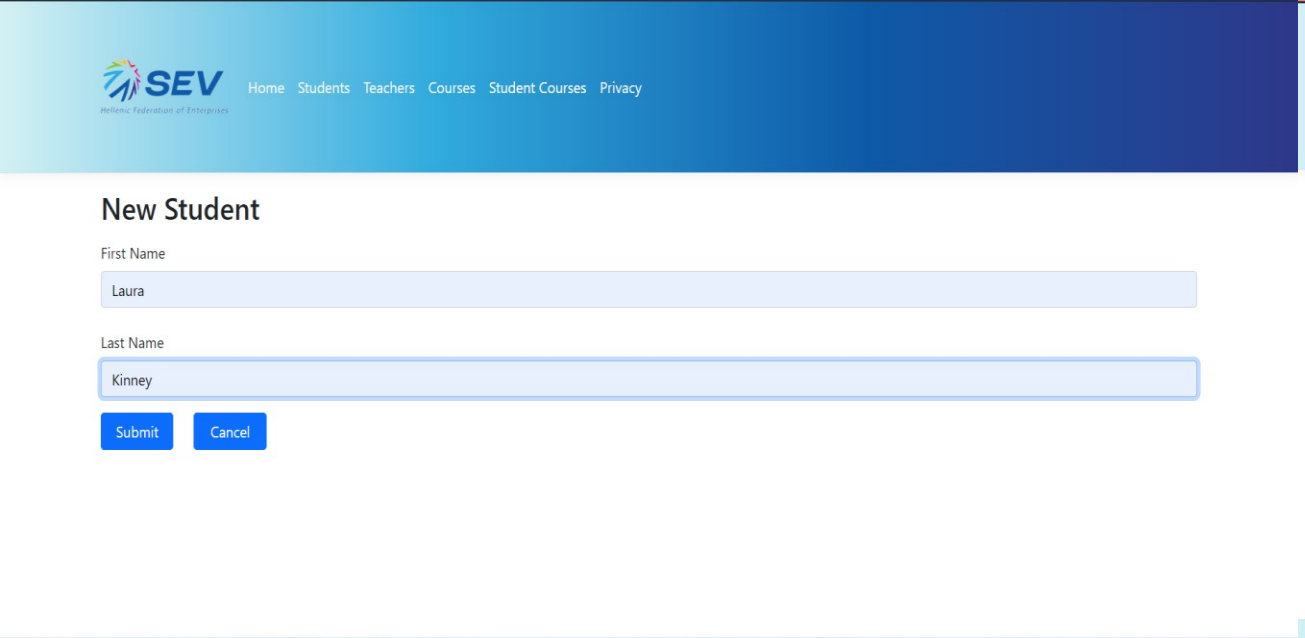


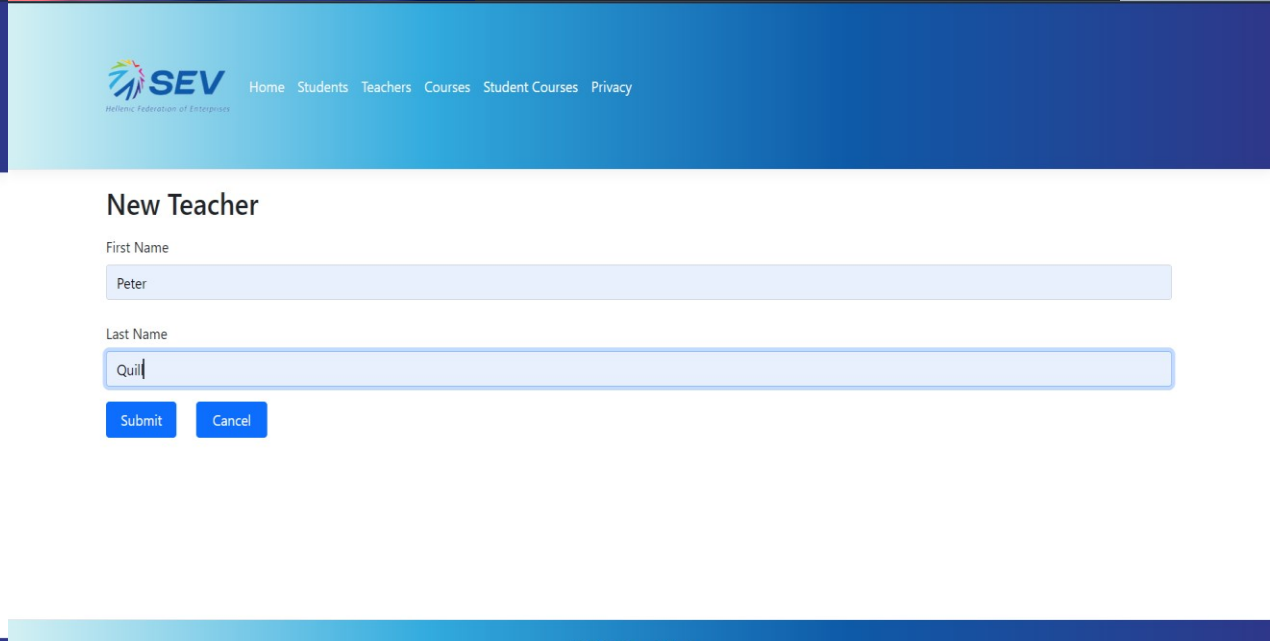


ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

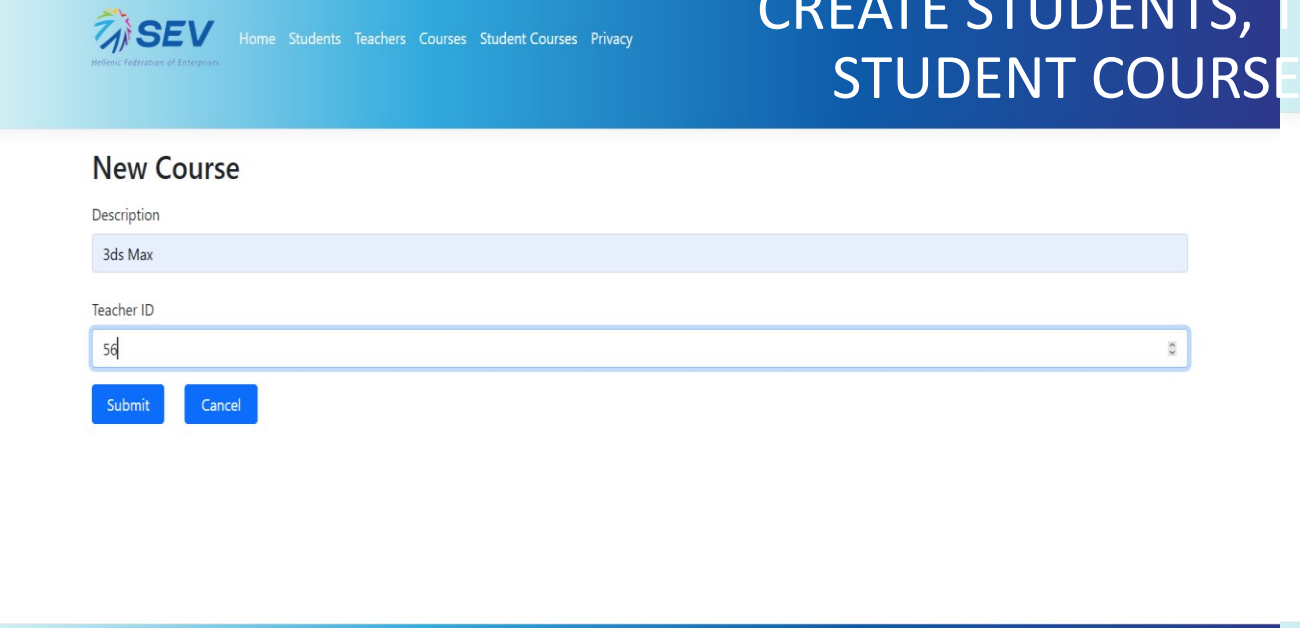
INDEX STUDENTS, TEACHERS, COURSES, STUDENT COURSES

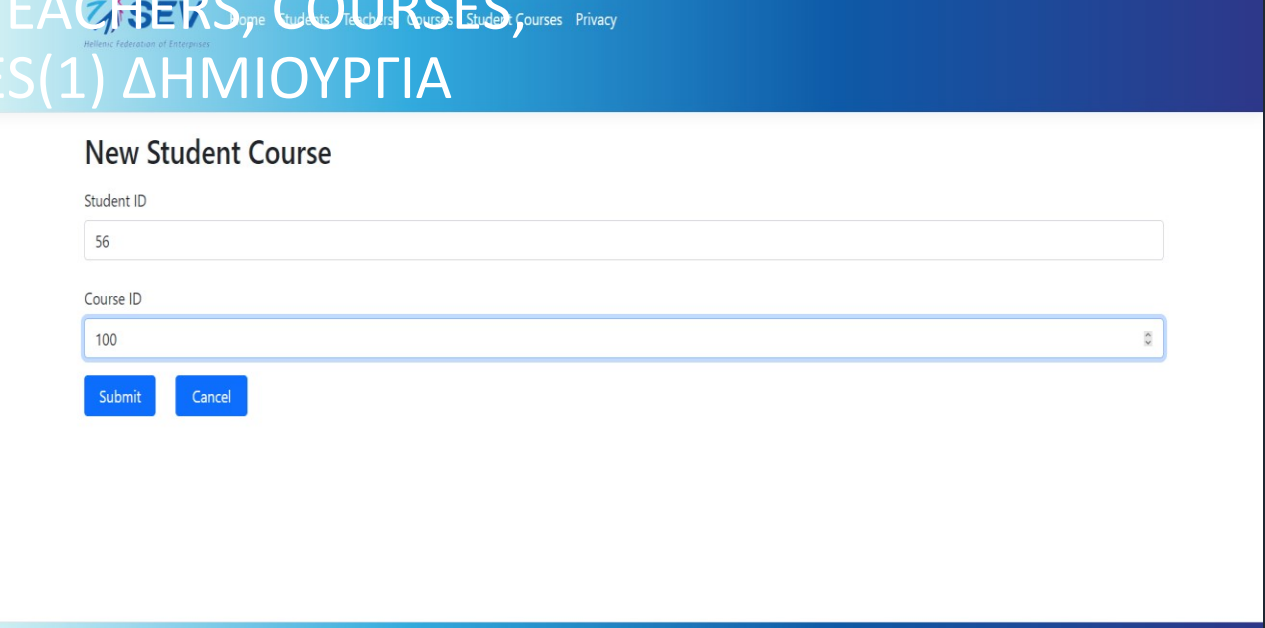






ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ
CREATE STUDENTS, TEACHERS, COURSES, STUDENT COURSES(1) ΔΗΜΙΟΥΡΓΙΑ





SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Students

New Student

ID	Firstname	Lastname	
50	Peter	Parker	Update Delete
51	Eric	Brooks	Update Delete
52	Johnny	Blaze	Update Delete
53	Jennifer	Walters	Update Delete
54	Stephen	Strange	Update Delete
55	Jean	Grey	Update Delete
56	Laura	Kinney	Update Delete

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Teachers

New Teacher

ID	Firstname	Lastname	
50	Tony	Stark	Update Delete
51	Steve	Rogers	Update Delete
52	Bruce	Banner	Update Delete
53	Thor	Odinson	Update Delete
54	Wanda	Maximoff	Update Delete
55	Natasha	Romanoff	Update Delete
56	Peter	Quill	Update Delete

ΕΠΙΔΕΙΞΗ ΚΑΤΑΛΛΗΛΟΤΗΤΑΣ

CREATE STUDENTS, TEACHERS, COURSES, STUDENT COURSES(2) ΔΗΜΙΟΥΡΓΗΘΗΚΕ

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Courses

New Course

ID	Description	Teacher ID	
90	UDK	50	Update Delete
91	OpenGL	51	Update Delete
92	Web Development	52	Update Delete
93	Python	53	Update Delete
94	Agile/Scrum	54	Update Delete
95	Databases	55	Update Delete
100	3ds Max	56	Update Delete

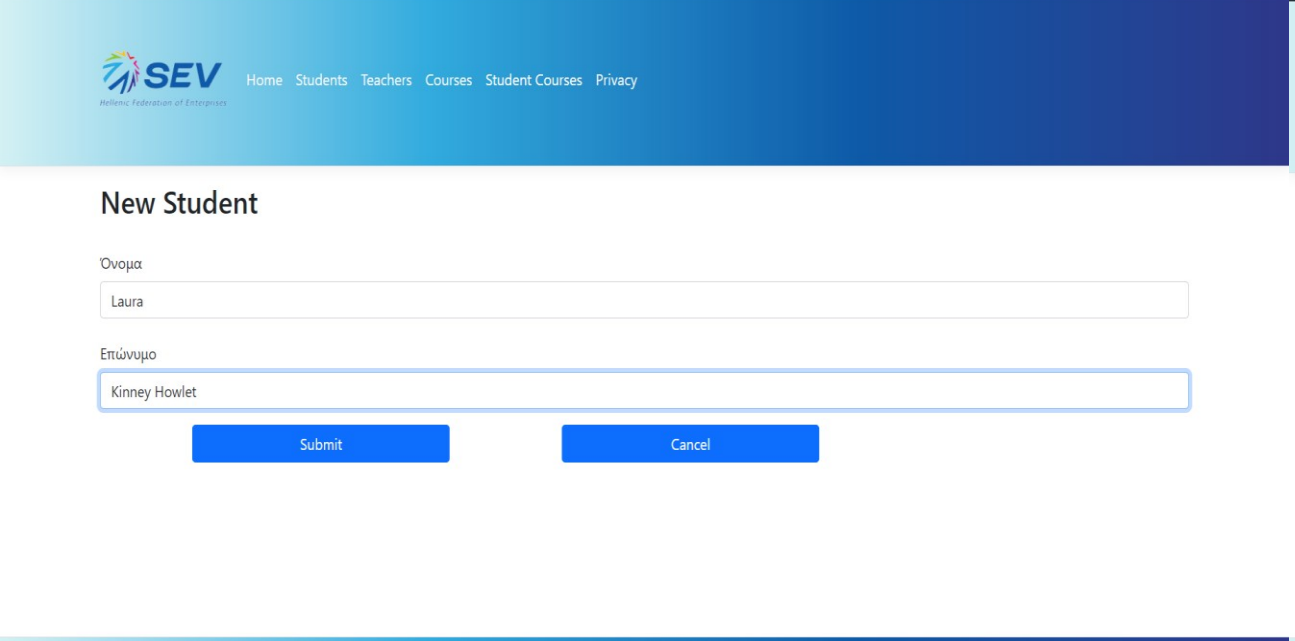
SEV Hellenic Federation of Enterprises

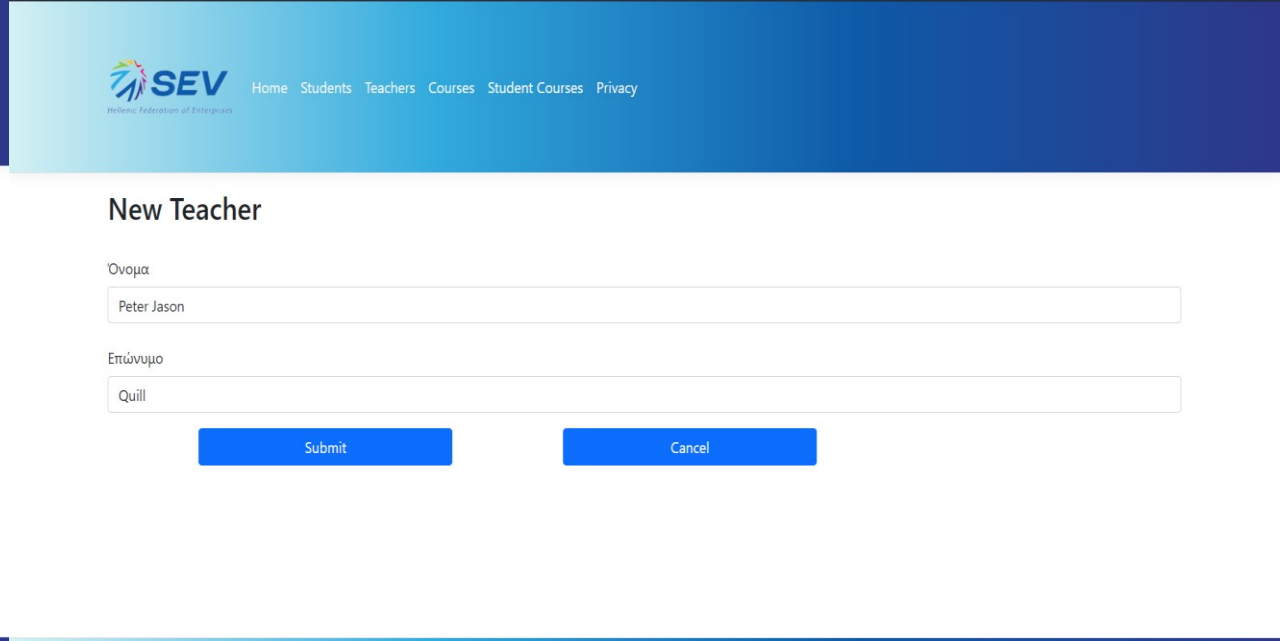
Home Students Teachers Courses Student Courses Privacy

List of Student Courses

New Student Course

Student ID	Course ID	
50	90	Update Delete
51	91	Update Delete
52	92	Update Delete
53	93	Update Delete
54	94	Update Delete
55	95	Update Delete
56	100	Update Delete

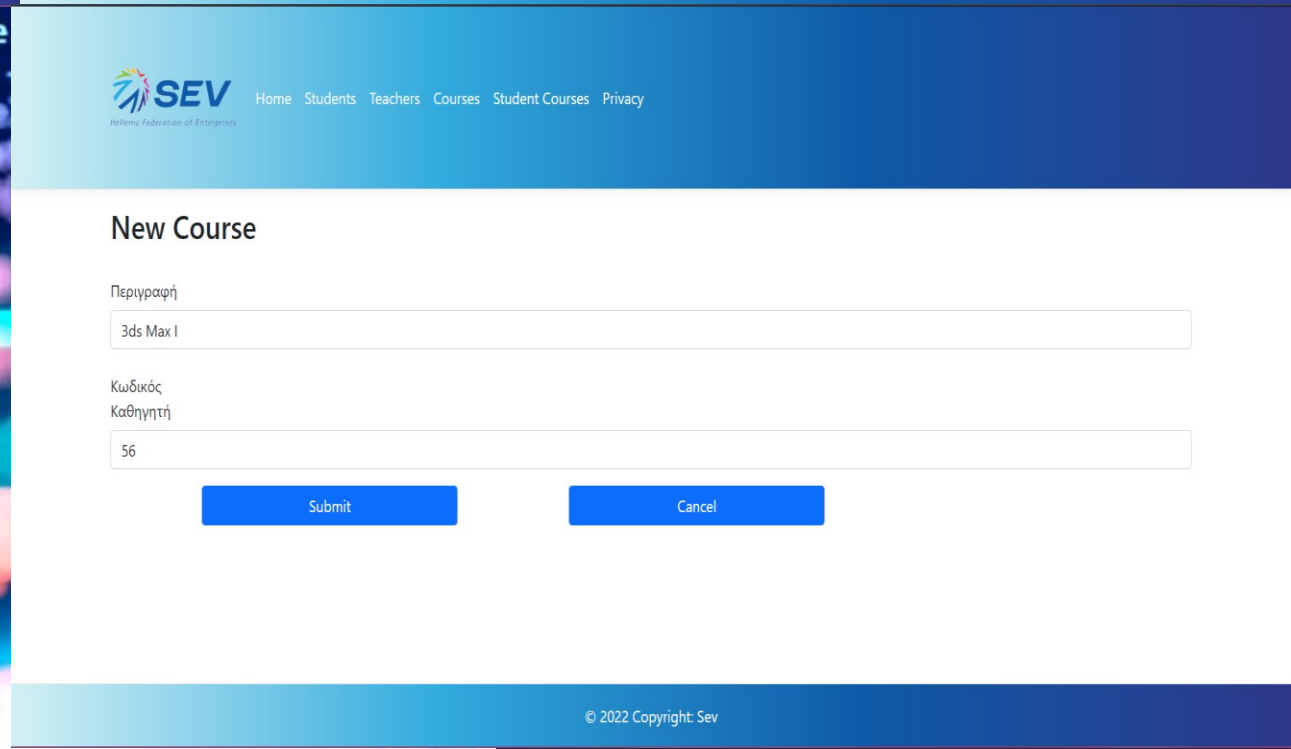




ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

UPDATE STUDENTS, TEACHERS, COURSES (1) ΔΗΜΙΟΥΡΓΙΑ

Πατάμε το κουμπί update δίπλα από το row που θέλουμε για να γίνουν οι αλλανές που θέλουμε



SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Students

New Student

ID	Firstname	Lastname	
50	Peter	Parker	Update Delete
51	Eric	Brooks	Update Delete
52	Johnny	Blaze	Update Delete
53	Jennifer	Walters	Update Delete
54	Stephen	Strange	Update Delete
55	Jean	Grey	Update Delete
56	Laura	Kinney Howlet	Update Delete

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Teachers

New Teacher

ID	Firstname	Lastname	
50	Tony	Stark	Update Delete
51	Steve	Rogers	Update Delete
52	Bruce	Banner	Update Delete
53	Thor	Odinson	Update Delete
54	Wanda	Maximoff	Update Delete
55	Natasha	Romanoff	Update Delete
56	Peter Jason	Quill	Update Delete

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

UPDATE STUDENTS, TEACHERS, COURSES (2) ΔΗΜΙΟΥΡΓΗΘΗΚΕ

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Courses

New Course

ID	Description	Teacher ID	
90	UDK	50	Update Delete
91	OpenGL	51	Update Delete
92	Web Development	52	Update Delete
93	Python	53	Update Delete
94	Agile/Scrum	54	Update Delete
95	Databases	55	Update Delete
100	3ds Max I	56	Update Delete

© 2022 Copyright: Sev

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Students

New Student

ID	Firstname	Lastname	
50	Peter	Parker	Update Delete
51	Eric	Brooks	Update Delete
52	Johnny	Blaze	Update Delete
53	Jennifer	Walters	Update Delete
54	Stephen	Strange	Update Delete
55	Jean	Grey	Update Delete

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Teachers

New Teacher

ID	Firstname	Lastname	
50	Tony	Stark	Update Delete
51	Steve	Rogers	Update Delete
52	Bruce	Banner	Update Delete
53	Thor	Odinson	Update Delete
54	Wanda	Maximoff	Update Delete
55	Natasha	Romanoff	Update Delete

DELETE STUDENTS, TEACHERS, COURSES, STUDENT COURSES

Πατάμε το κουμπί delete για να διαγράψουμε το row που θέλουμε

SEV Hellenic Federation of Enterprises

Home Students Teachers Courses Student Courses Privacy

List of Courses

New Course

ID	Description	Teacher ID	
90	UDK	50	Update Delete
91	OpenGL	51	Update Delete
92	Web Development	52	Update Delete
93	Python	53	Update Delete
94	Agile/Scrum	54	Update Delete
95	Databases	55	Update Delete

SEV Hellenic Federation of Enterprises

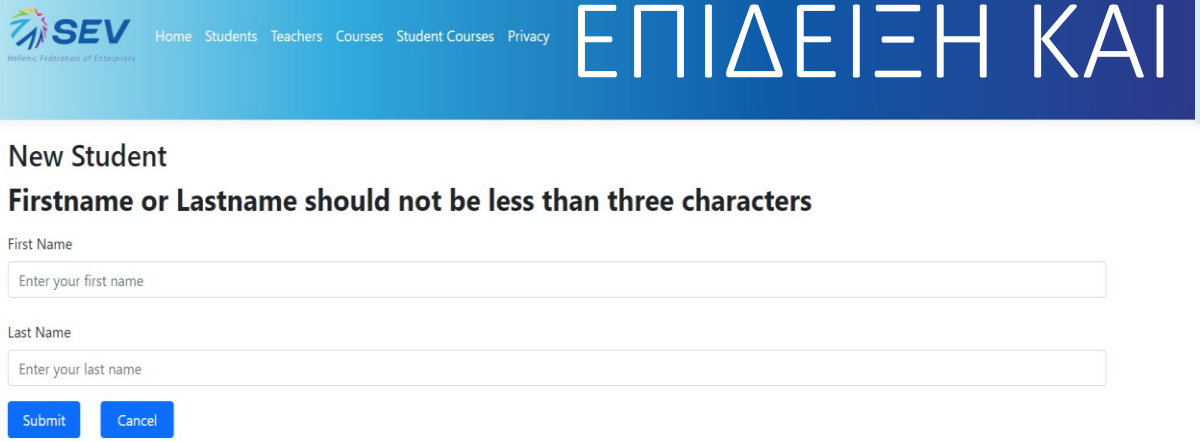
Home Students Teachers Courses Student Courses Privacy

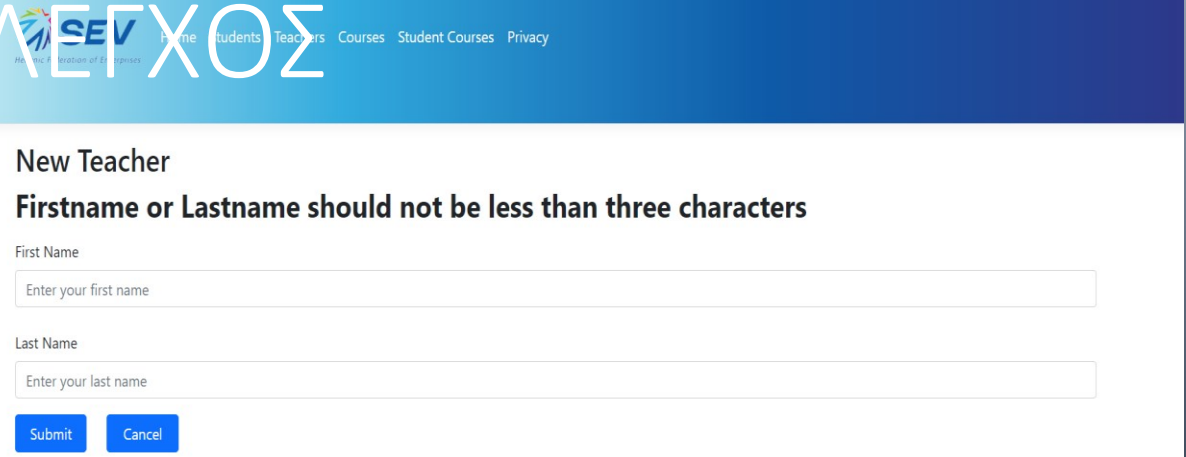
List of Student Courses

New Student Course

Student ID	Course ID	
50	90	Update Delete
51	91	Update Delete
52	92	Update Delete
53	93	Update Delete
54	94	Update Delete
55	95	Update Delete

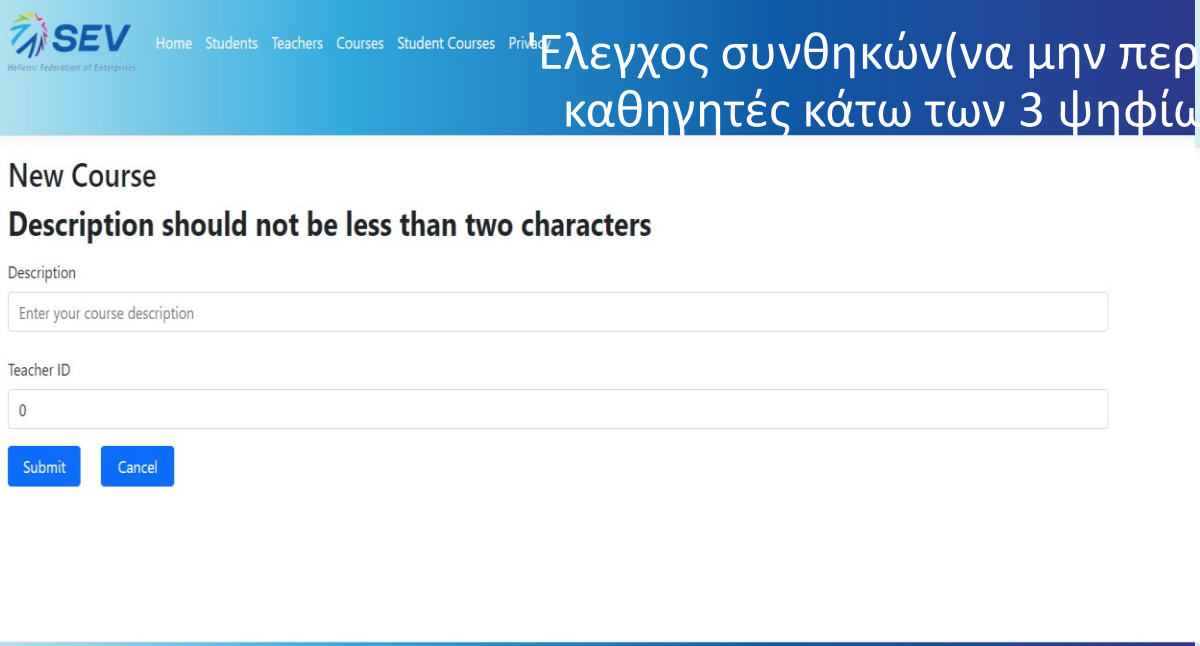
ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

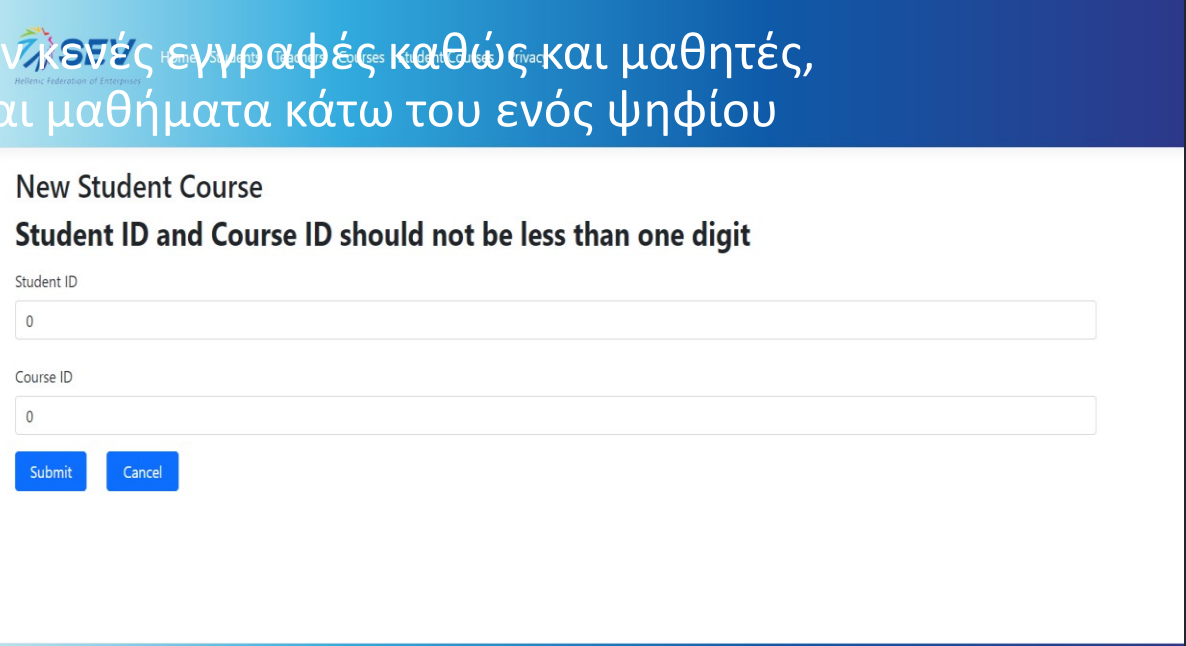




VALIDATOR STUDENTS, TEACHERS, COURSES, STUDENT COURSES

Έλεγχος συνθηκών(να μην περνούν κενές εγγραφές καθώς και μαθητές, καθηγητές κάτω των 3 ψηφίων και μαθήματα κάτω του ενός ψηφίου



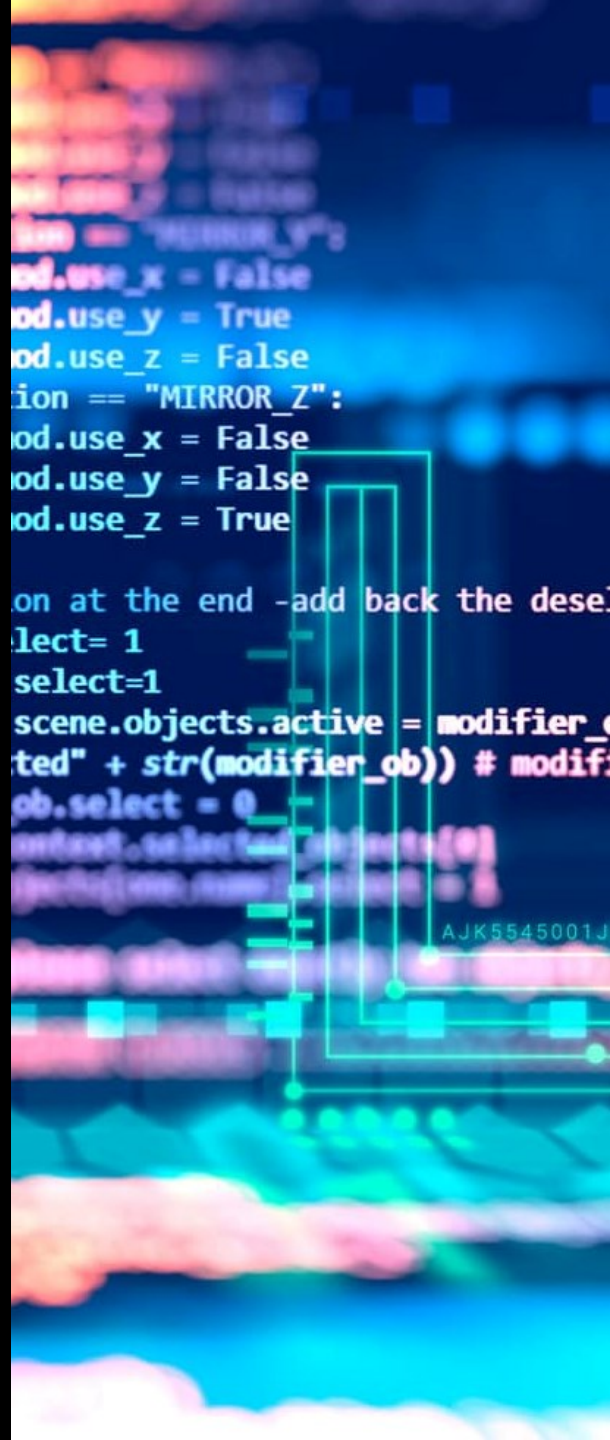


GITHUB PROJECT LINK

https://github.com/AnastasisPan/SevManagementApp

GITHUB PROFILE

https://github.com/AnastasisPan



ΒΙΒΛΙΟΓΡΑΦΙΑ

Προγραμματισμός με C# .NET Core - Αθανάσιος Ανδρούτσος

Προγραμματισμός με Java/JavaEE - Αθανάσιος Ανδρούτσος

Προγραμματισμός στο Web - Μάρκος Καραμπάτσης

Προγραμματισμός Βάσεων Δεδομένων - Χρυσόστομος Καπέτης

https://getbootstrap.com

https://www.privacypolicytemplate.net