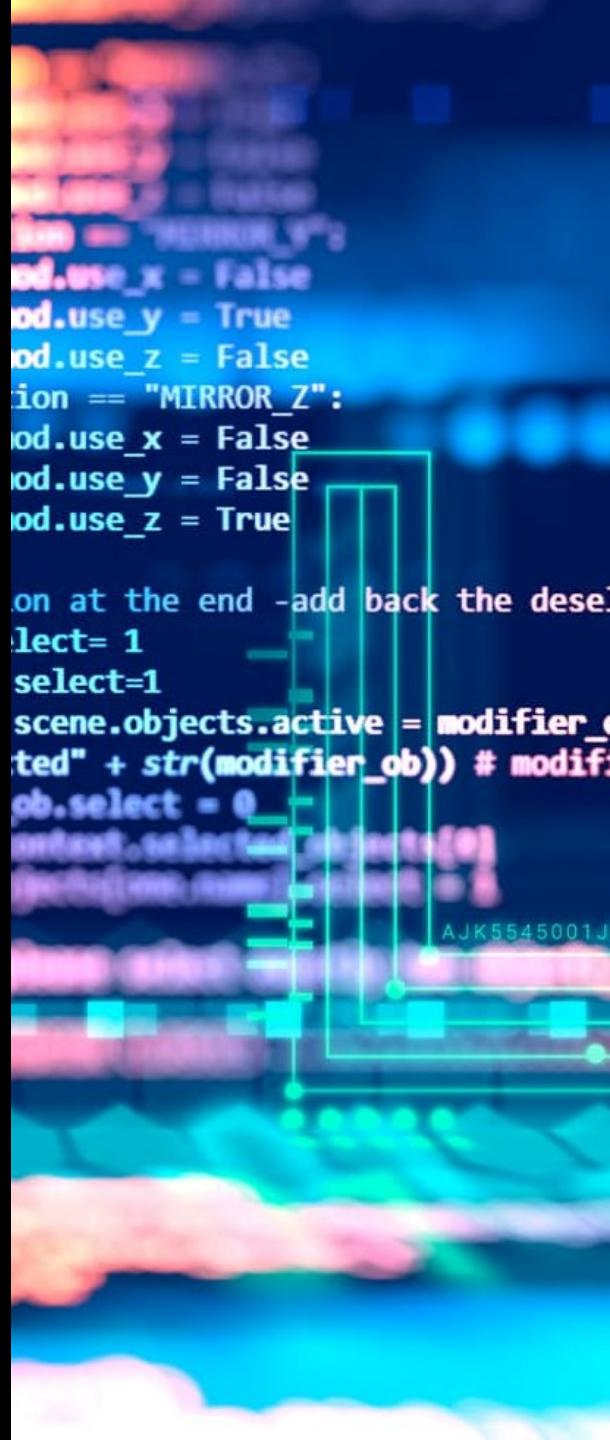


Final Project Report

ΟΝΟΜΑΤΕΠΩΝΥΜΟ DEVELOPER:
ΠΑΝΤΑΖΗΣ ΑΝΑΣΤΑΣΙΟΣ

ΕΙΔΙΚΟΤΗΤΑ: ΤΕΧΝΙΚΟΣ
ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ



```
    od.use_x = False
    od.use_y = True
    od.use_z = False
    ion == "MIRROR_Y":
        od.use_x = False
        od.use_y = True
        od.use_z = False
    ion == "MIRROR_Z":
        od.use_x = False
        od.use_y = False
        od.use_z = True
on at the end -add back the deselected
select= 1
select=1
scene.objects.active = modifier_object
ted" + str(modifier_ob)) # modifier
ob.select = 0
unselect_selected()
unselect_all()
AJK5545001J
```

Περιεχόμενα

- Εισαγωγή
 - Ανάλυση του προβλήματος
 - Σχεδιασμός
 - Εφαρμογή
 - Κώδικας
 - Βάση Δεδομένων
 - Υλοποίηση
 - Επίδειξη και Έλεγχος
 - Βιβλιογραφία

```
    if direction == "MIRROR_X":  
        mod.use_x = True  
        mod.use_y = False  
        mod.use_z = False  
    elif direction == "MIRROR_Y":  
        mod.use_x = False  
        mod.use_y = True  
        mod.use_z = False  
    elif direction == "MIRROR_Z":  
        mod.use_x = False  
        mod.use_y = False  
        mod.use_z = True  
  
on at the end -add back the deselected objects  
select= 1  
select=1  
scene.objects.active = modifier_object  
selected=" + str(modifier_ob)) # modifier object  
ob.select = 0  
context.selected_objects.add(ob)  
ob.select = 1  
ob.select = 0  
ob.select = 1  
ob.select = 0
```

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 ΑUEB.

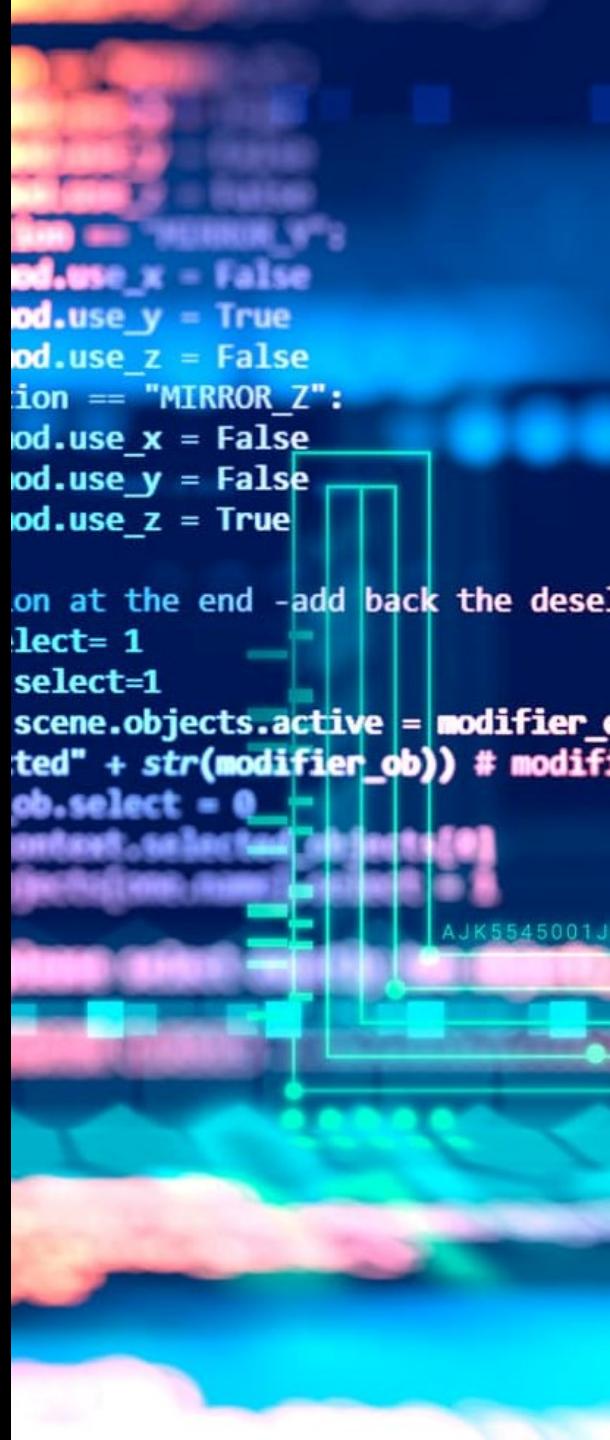
Σύμφωνα με το domain model που μας δώθηκε πρέπει να δημιουργήσουμε τα αντίστοιχα

DAO/DTO/Service Layer/Controllers/Web Pages

χρησιμοποιώντας βάση δεδομένων MSSQLSERVER,

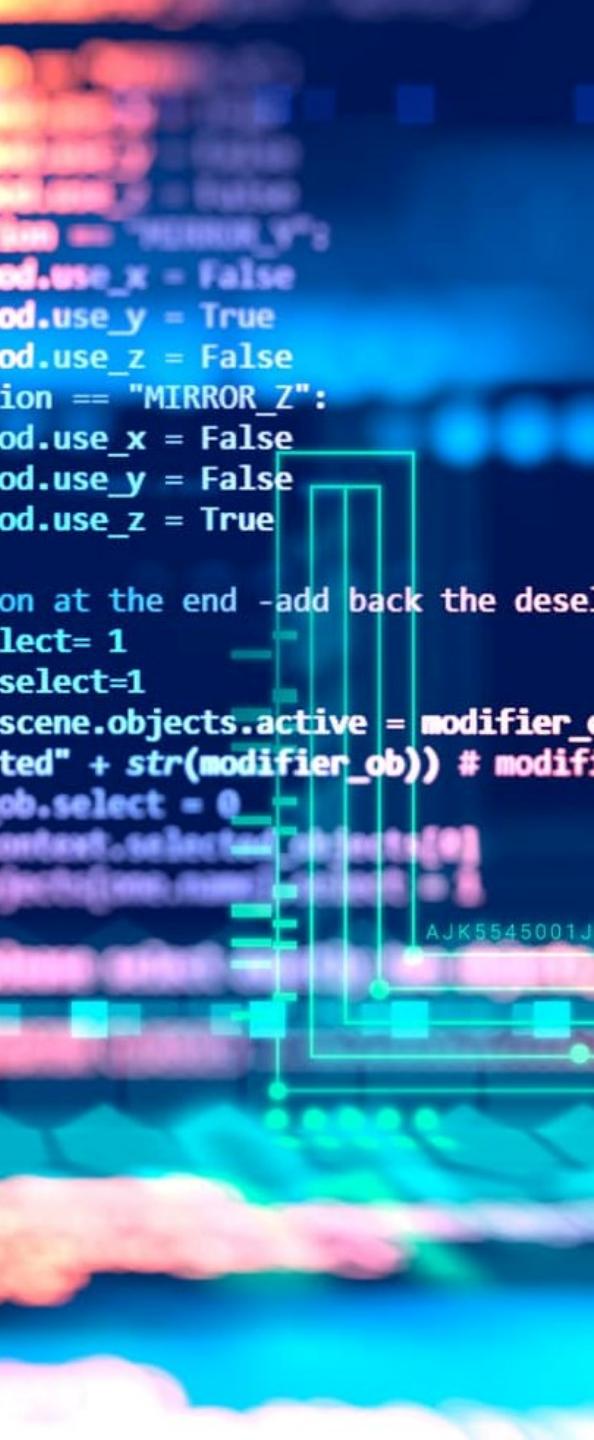
C# και ASP .NET με Razor Pages και Web Front-End

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

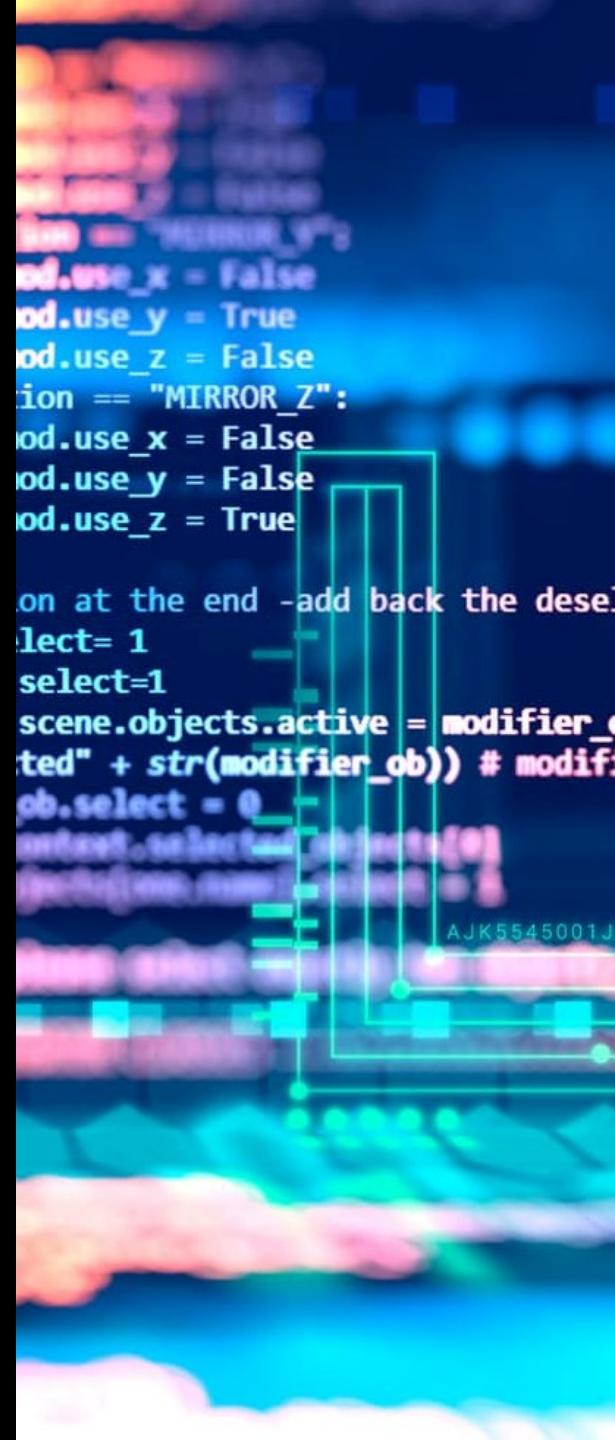


```
    od.use_x = False
    od.use_y = True
    od.use_z = False
    ion == "MIRROR_Y":
        od.use_x = False
        od.use_y = True
        od.use_z = False
    ion == "MIRROR_Z":
        od.use_x = False
        od.use_y = False
        od.use_z = True
on at the end -add back the deselected
select= 1
select=1
scene.objects.active = modifier_selected" + str(modifier_ob)) # modifier
ob.select = 0
modifier.select = modifier.select
modifier.select = modifier.select
AJK5545001J
```

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 ή 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.

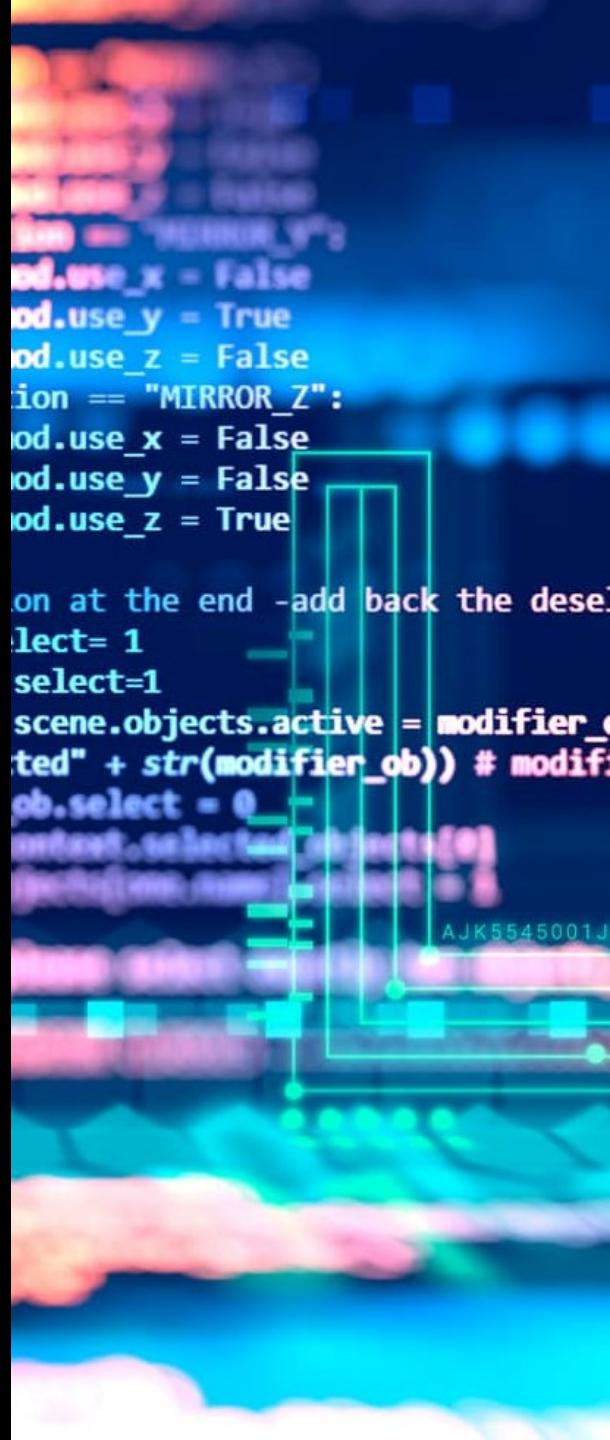
Data Access Object (Dao) Layer: χρησιμοποιείται για να έχουμε πρόσβαση στην βάση δεδομένων

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

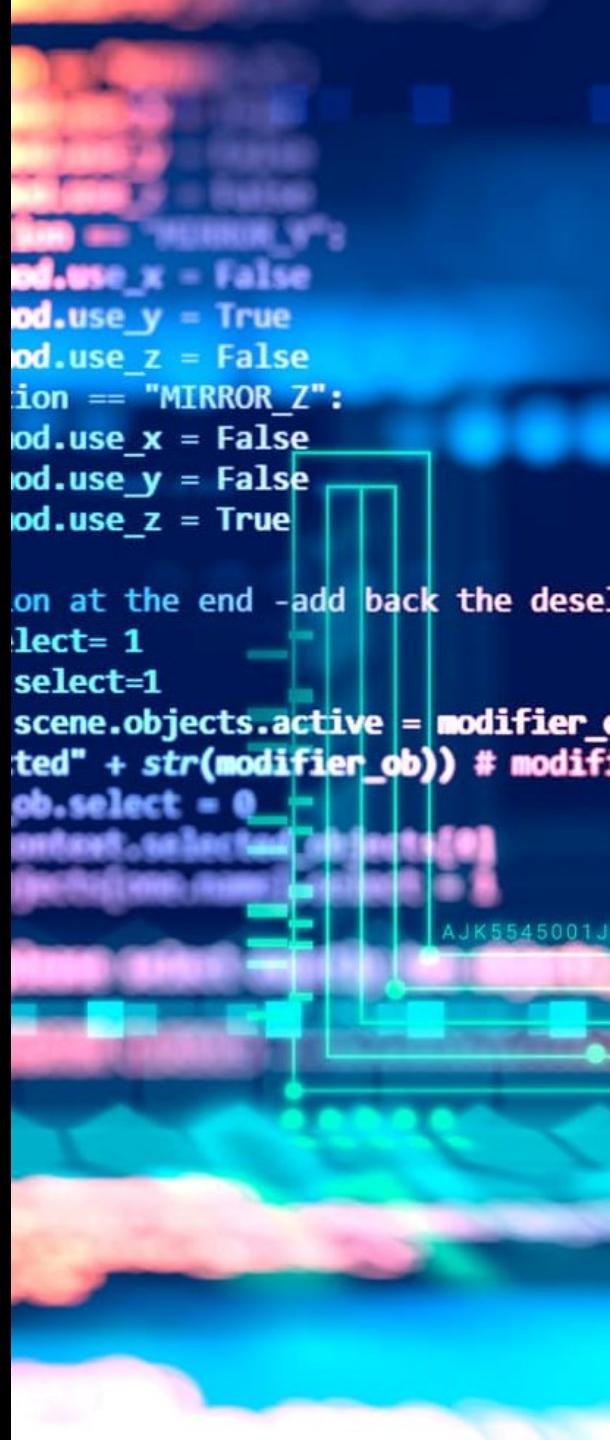
Η βάση δεδομένων που θα δημιουργήσουμε θα περιέχει 4 πίνακες σύμφωνα με τις 4 κλάσσεις που αναγράφονται στο Domain Model, οι οποίες είναι οι εξής:

Students Table(Μαθητές)

Teachers Table(Καθηγητές)

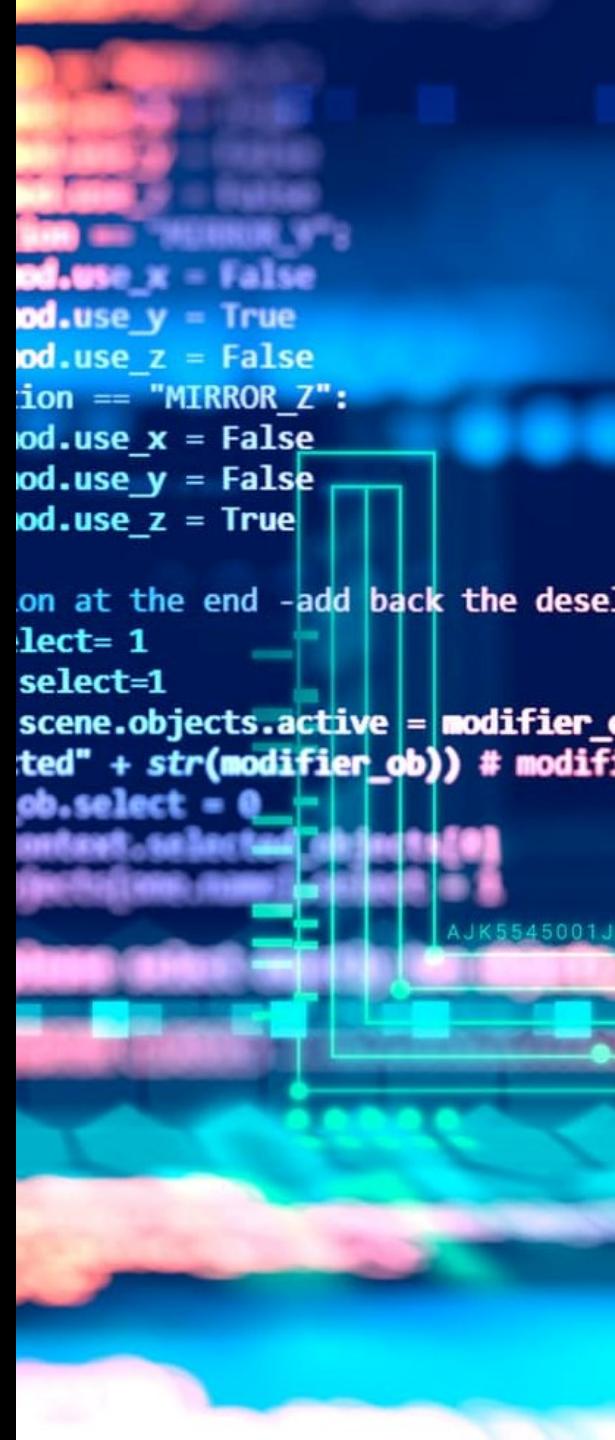
Courses Table(Μαθήματα)

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

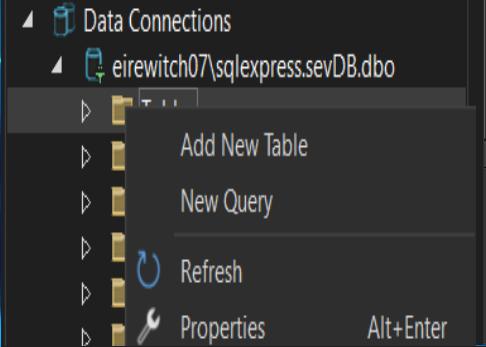
Πατάμε View/Server
Explorer και μετά connect
to database

Server Explorer



- Connect to Database
- eirewitch07\sqlexpress.sevDB.dbo
 - Tables
 - COURSES
 - STUDENTS
 - STUDENTS_COURSES
 - TEACHERS
 - Views
 - Stored Procedures
 - Functions
 - Synonyms
 - Types
 - Assemblies
- Servers

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



dbo.STUDENTS [Design]

Update | Script File: dbo.STUDENTS.sql

Name	Data Type	Allow Nulls	Default
Id	int	<input checked="" type="checkbox"/>	
FIRSTNAME	nchar(50)	<input checked="" type="checkbox"/>	
LASTNAME	nchar(50)	<input checked="" type="checkbox"/>	

Keys (1)
 <unnamed> (Primary Key, Clustered: Id)

Check Constraints (2)
 CK_STUDENTS_FIRSTNAME (FIRSTNAME)
 CK_STUDENTS_LASTNAME (LASTNAME)

Indexes (1)
 IX_STUDENTS_LASTNAME (LASTNAME)

Foreign Keys (0)

Triggers (0)

T-SQL

```
1 CREATE TABLE [dbo].[STUDENTS] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [FIRSTNAME] NCHAR (50) NULL,
4     [LASTNAME] NCHAR (50) NULL,
5     PRIMARY KEY CLUSTERED ([Id] ASC),
6     CONSTRAINT [CK_STUDENTS_FIRSTNAME] CHECK (len([FIRSTNAME]) >= (3)),
7     CONSTRAINT [CK_STUDENTS_LASTNAME] CHECK (len([LASTNAME]) >= (3))
```

dbo.TEACHERS [Design]

Update | Script File: dbo.TEACHERS.sql

Name	Data Type	Allow Nulls	Default
Id	int	<input checked="" type="checkbox"/>	
FIRSTNAME	nchar(50)	<input checked="" type="checkbox"/>	
LASTNAME	nchar(50)	<input checked="" type="checkbox"/>	

Keys (1)
 <unnamed> (Primary Key, Clustered: Id)

Check Constraints (2)
 CK_TEACHERS_FIRSTNAME (FIRSTNAME)
 CK_TEACHERS_LASTNAME (LASTNAME)

Indexes (1)
 IX_TEACHERS_LASTNAME (LASTNAME)

Foreign Keys (0)

Triggers (0)

T-SQL

```
1 CREATE TABLE [dbo].[TEACHERS] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [FIRSTNAME] NCHAR (50) NULL,
4     [LASTNAME] NCHAR (50) NULL,
5     PRIMARY KEY CLUSTERED ([Id] ASC),
6     CONSTRAINT [CK_TEACHERS_FIRSTNAME] CHECK (len([FIRSTNAME]) >= (3)),
7     CONSTRAINT [CK_TEACHERS_LASTNAME] CHECK (len([LASTNAME]) >= (3))
```

dbo.COURSES [Design]

Update | Script File: dbo.COURSES.sql

Name	Data Type	Allow Nulls	Default
Id	int	<input checked="" type="checkbox"/>	
DESCRIPTION	nchar(50)	<input checked="" type="checkbox"/>	
TEACHER_ID	int	<input checked="" type="checkbox"/>	

Keys (1)
 <unnamed> (Primary Key, Clustered: Id)

Check Constraints (1)
 CK_COURSES_DESCRIPTION (DESCRIPTION)

Indexes (1)
 IX_COURSES_DESCRIPTION (DESCRIPTION)

Foreign Keys (1)
 FK_COURSES_To (Id)

Triggers (0)

T-SQL

```
1 CREATE TABLE [dbo].[COURSES] (
2     [Id] INT IDENTITY (1, 1) NOT NULL,
3     [DESCRIPTION] NCHAR (50) NULL,
4     [TEACHER_ID] INT NULL,
5     PRIMARY KEY CLUSTERED ([Id] ASC),
6     CONSTRAINT [FK_COURSES_To] FOREIGN KEY ([TEACHER_ID]) REFERENCES [dbo].[TEACHERS] ([Id]),
7     CONSTRAINT [CK_COURSES_DESCRIPTION] CHECK (len([DESCRIPTION]) >= (2))
```

dbo.STUDENT...SES [Design]

Update | Script File: dbo.STUDENTS_COURSES.sql

Name	Data Type	Allow Nulls	Default
STUDENT_ID	int	<input checked="" type="checkbox"/>	
COURSE_ID	int	<input checked="" type="checkbox"/>	

Keys (0)

Check Constraints (0)

Indexes (0)

Foreign Keys (2)

FK_STUDENTS_COURSES (Id)

FK_STUDENTS_COURSES_To (Id)

Triggers (0)

dbo.STUDENTS_COURSES [Design]

Update | Script File: dbo.STUDENTS_COURSES.sql

1 CREATE TABLE [dbo].[STUDENTS_COURSES] (
2 [STUDENT_ID] INT NULL,
3 [COURSE_ID] INT NULL,
4 CONSTRAINT [FK_STUDENTS_COURSES] FOREIGN KEY ([STUDENT_ID]) REFERENCES [dbo].[STUDENTS] ([Id]),
5 CONSTRAINT [FK_STUDENTS_COURSES_To] FOREIGN KEY ([COURSE_ID]) REFERENCES [dbo].[COURSES] ([Id])
6);
7
8

Update Database

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



Data Connections

eirewitch07\sqlexpress.sevDB.dbo

Tables

COURSES

STUDENTS

STUDENTS_COURSES

TEACHERS

Views

Stored Procedures

Functions

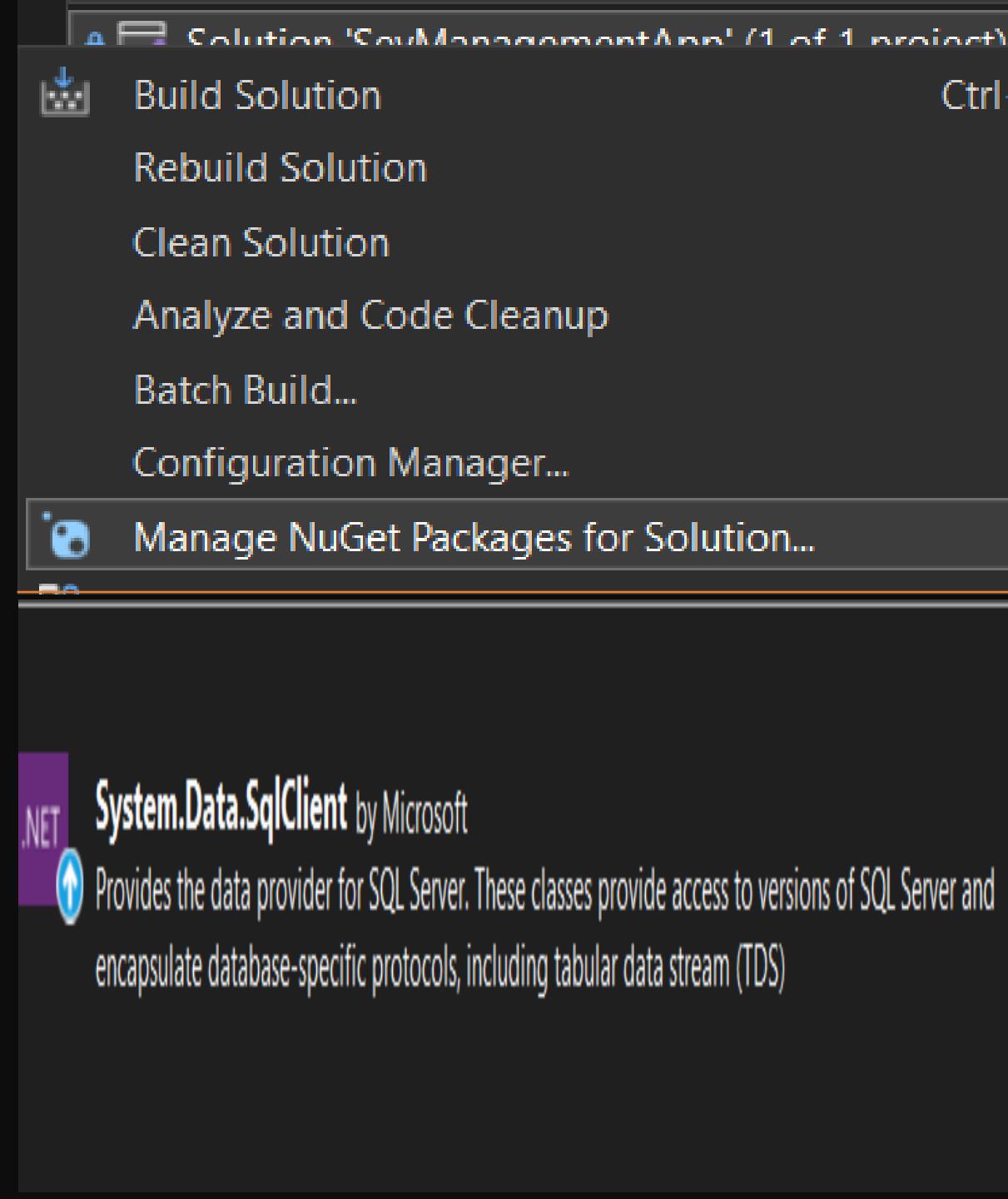
Synonyms

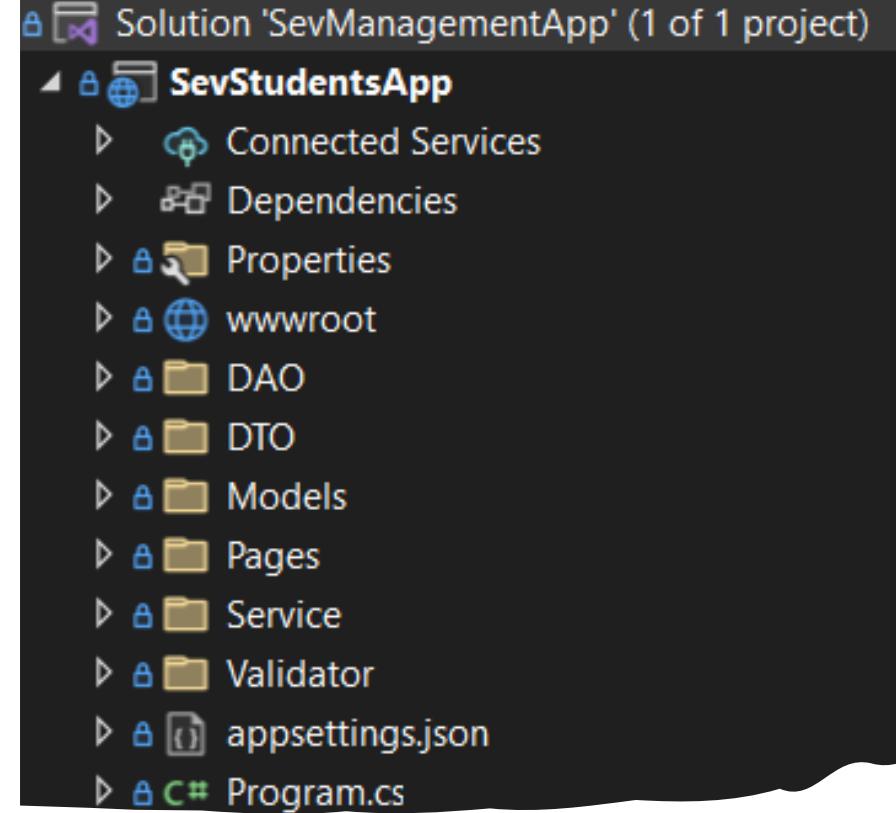
Types

Assemblies

System.Data.SqlClient

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





Δομή Project

Model

- Επειδή έχουμε 4 διαφορετικούς πίνακες χρειαζόμαστε και 4 διαφορετικά models

```
Models
  ▶ Course.cs
  ▶ Student.cs
  ▶ StudentCourse.cs
  ▶ Teacher.cs
```

```
Teacher.cs
namespace SevTeachersApp.Models
{
    public class Teacher
    {
        public int Id { get; set; }
        public string? Fristname { get; set; }
        public string? Lastname { get; set; }
    }
}
```

```
StudentCourse.cs
namespace SevStudentCoursesApp.Models
{
    public class StudentCourse
    {
        public int StudentId { get; set; }
        public int CourseId { get; set; }
    }
}
```

```
Student.cs
namespace SevStudentsApp.Models
{
    public class Student
    {
        public int Id { get; set; }
        public string? Fristname { get; set; }
        public string? Lastname { get; set; }
    }
}
```

```
Course.cs
namespace SevCoursesApp.Models
{
    public class Course
    {
        public int Id { get; set; }
        public string? Description { get; set; }
        public int TeacherId { get; set; }
    }
}
```



DAO-ISTudentDAO
 DAO-ITeacherDAO
 DAO-ICourseDAO

DAO

```

using SevStudentsApp.Models;

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

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();
    }
}
```

IStudentCourseDAO

```

using SevCoursesApp.Models;

namespace SevCoursesApp.DAO
{
    public interface ICourseDAO
    {
        void Insert(Course? course);
        void Update(Course? course);
        Course? Delete(Course? course);
        Course? GetCourse(int id);
        List<Course> GetAll();
    }
}

using SevStudentCoursesApp.Models;

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

- Επειδή έχουμε 4 διαφορετικούς πίνακες χρειαζόμαστε και 4 διαφορετικά DAO

```
DAO-IStudentDAOImpl  
DAO-ITeacherDAOImpl  
DAO-ICourseDAOImpl  
O-IStudentCourseDAOImpl
```

```
StudentDAOImpl.cs  CourseDAOImpl.cs
SevStudentsApp
{ 1  using SevStudentsApp.DAO.DBUtil;
 2  using SevStudentsApp.Models;
 3  using System.Data.SqlClient;
 4
 5  namespace SevStudentsApp.DAO
 6  {
 7      {
 8          4 references
 9          public class StudentDAOImpl : IStudentDAO
10          {
11              2 references
12              public Student? Delete(Student? student)
13
14              34
15
16              2 references
17              public List<Student> GetAll()
18
19              71
20
21              2 references
22              public Student? GetStudent(int id)
23
24              108
25
26              2 references
27              public void Insert(Student? student)
28
29              136
30
31              2 references
32              public void Update(Student? student)
33
34              164
35
36              165
37      }
38  }
```

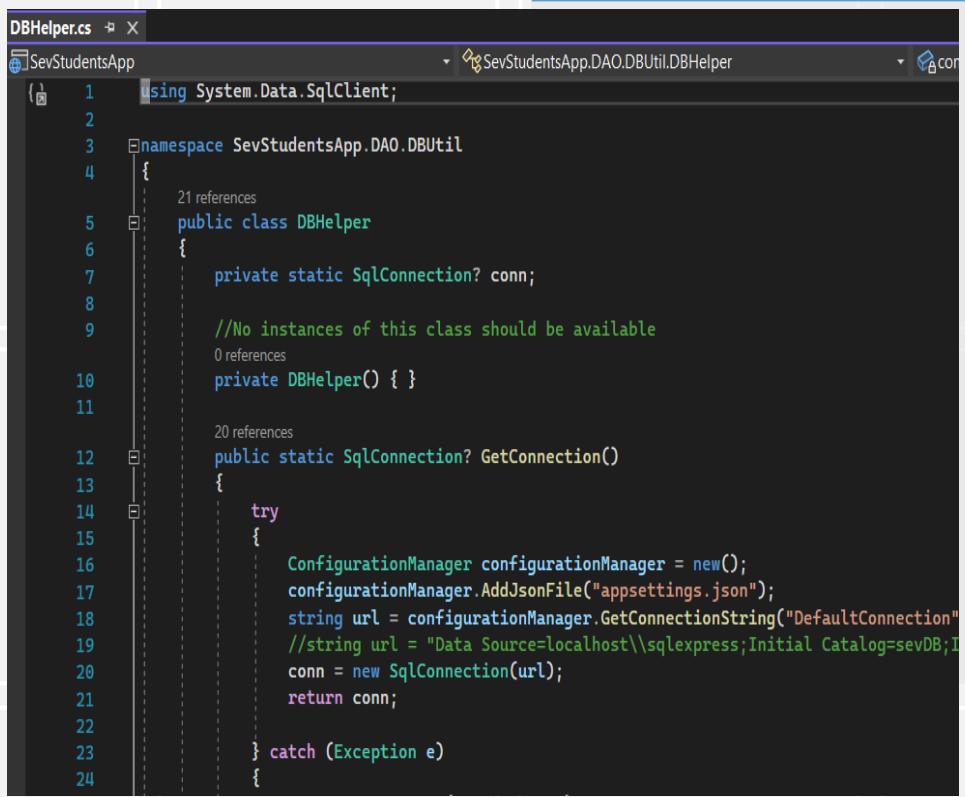
```
1  using SevStudentsApp.DAO.DBUtil;
2  using SevTeachersApp.Models;
3  using System.Data.SqlClient;
4
5  namespace SevTeachersApp.DAO
6  {
7      public class TeacherDAOImpl : ITeacherDAO
8      {
9          public Teacher? Delete(Teacher? teacher)
10         {
11             return null;
12         }
13
14         public List<Teacher> GetAll()
15         {
16             return new List<Teacher>();
17         }
18
19         public Teacher? GetTeacher(int id)
20         {
21             return null;
22         }
23
24         public void Insert(Teacher? teacher)
25         {
26             // Implementation
27         }
28
29         public void Update(Teacher? teacher)
30         {
31             // Implementation
32         }
33     }
34 }
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165 }
```

```
CourseDAOImpl.cs  ✘ X
SevStudentsApp
SevCoursesApp.DAODAO
CourseDAOImpl : ICourseDAO
Delete(Course? course)
GetAll()
GetCourse(int id)
Insert(Course? course)
Update(Course? course)
```

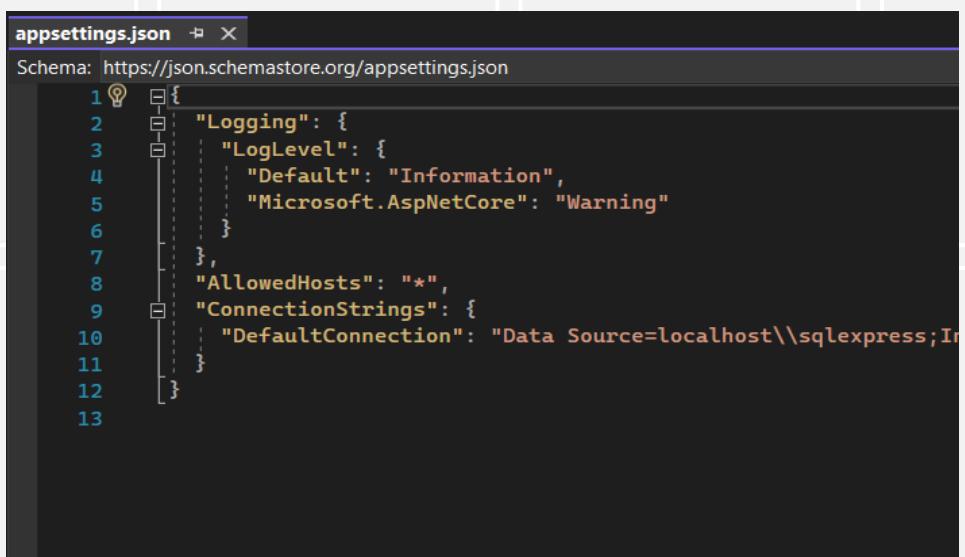
```
StudentCourseDAOImpl.cs  ✘ x TeacherDAOImpl.cs          StudentDAOImpl.cs          CourseDAOImpl.cs
SevStudentsApp
1  using SevStudentsApp.DAO.DBUtil;
2  using SevStudentCoursesApp.Models;
3  using System.Data.SqlClient;
4
5  namespace SevStudentCoursesApp.DAO
6  {
7      public class StudentCourseDAOImpl : IStudentCourseDAO
8      {
9          public StudentCourse? Delete(StudentCourse? studentcourse)
10         ...
11
12         public List<StudentCourse> GetAll()
13         ...
14
15         public StudentCourse? GetStudentCourse(int student_id)
16         ...
17
18         public void Insert(StudentCourse? studentcourse)
19         ...
20
21         public void Update(StudentCourse? studentcourse)
22         ...
23     }
24 }
```

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

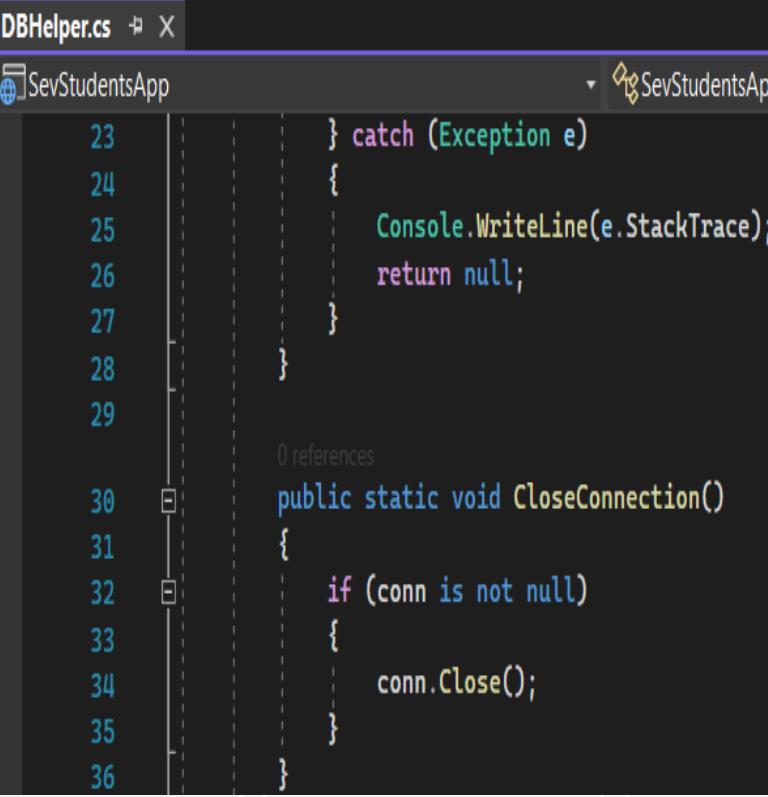
DBHelper Utility Class



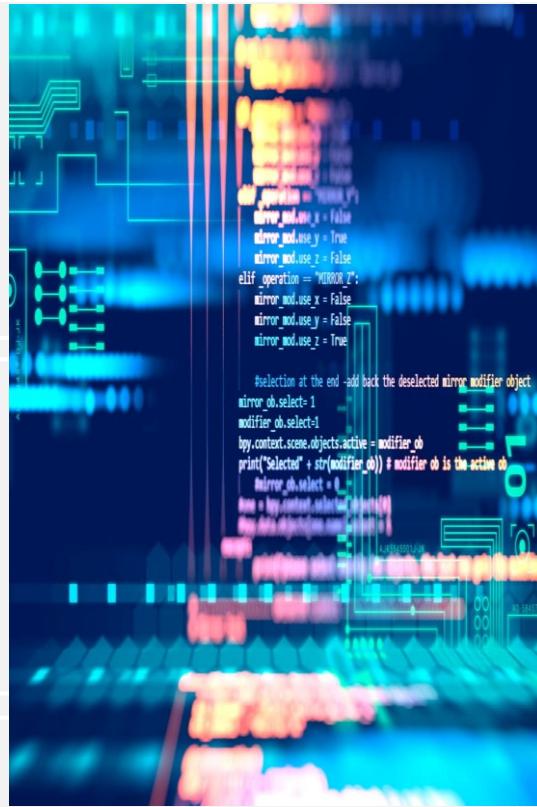
```
1 using System.Data.SqlClient;
2
3 namespace SevStudentsApp.DAO.DBUtil
4 {
5     public class DBHelper
6     {
7         private static SqlConnection? conn;
8
9         //No instances of this class should be available
10        private DBHelper() { }
11
12        public static SqlConnection? GetConnection()
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";
20                conn = new SqlConnection(url);
21                return conn;
22            } catch (Exception e)
23            {
24            }
25        }
26    }
}
```



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



```
13         } catch (Exception e)
14         {
15             Console.WriteLine(e.StackTrace);
16             return null;
17         }
18     }
19
20     0 references
21     public static void CloseConnection()
22     {
23         if (conn is not null)
24         {
25             conn.Close();
26         }
27     }
28
29
30     0 references
31     public static void CloseConnection()
32     {
33         if (conn is not null)
34         {
35             conn.Close();
36         }
37     }
38 }
```



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
public void Insert(StudentCourse? studentcourse)
{
    if (studentcourse == null) return;

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

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

        string sql = "INSERT INTO STUDENTS_COURSES " +
                    "(STUDENT_ID, COURSE_ID) VALUES " +
                    "(@student_id, @course_id)";

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

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

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}

TeacherDAOImpl.cs
public void Insert(Teacher? teacher)
{
    if (teacher == null) return;

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

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

        string sql = "INSERT INTO TEACHERS " +
                    "(FIRSTNAME, LASTNAME) VALUES " +
                    "(@firstname, @lastname)";

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

        command.Parameters.AddWithValue("@firstname", teacher.Firstname);
        command.Parameters.AddWithValue("@lastname", teacher.Lastname);

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}

StudentDAOImpl.cs
public void Insert(Student? student)
{
    if (student == null) return;

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

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

        string sql = "INSERT INTO STUDENTS " +
                    "(FIRSTNAME, LASTNAME) VALUES " +
                    "(@firstname, @lastname)";

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

        command.Parameters.AddWithValue("@firstname", student.Firstname);
        command.Parameters.AddWithValue("@lastname", student.Lastname);

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}

CourseDAOImpl.cs
public void Insert(Course? course)
{
    if (course == null) return;

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

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

        string sql = "INSERT INTO COURSES " +
                    "(DESCRIPTION, TEACHER_ID) VALUES " +
                    "(@description, @teacherid)";

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

        command.Parameters.AddWithValue("@description", course.Description);
        command.Parameters.AddWithValue("@teacherid", course.TeacherId);

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

```
public void Insert(StudentCourse? studentcourse)
{
    if (studentcourse == null) return;

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

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

        string sql = "INSERT INTO STUDENTS_COURSES " +
                    "(STUDENT_ID, COURSE_ID) VALUES " +
                    "(@student_id, @course_id)";

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

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

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

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

GET METHOD

```
StudentDAOImpl.cs
public Student? GetStudent(int id)
{
    Student? student = null;

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

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

        string sql = "SELECT * FROM STUDENTS WHERE ID = @id";

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

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

        using SqlDataReader reader = command.ExecuteReader();

        if (reader.Read())
        {
            student = new Student()
            {
                Id = reader.GetInt32(0),
                Firstname = reader.GetString(1),
                Lastname = reader.GetString(2)
            };
        }
    }
}

TeacherDAOImpl.cs
public Teacher? GetTeacher(int id)
{
    Teacher? teacher = null;

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

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

        string sql = "SELECT * FROM TEACHERS WHERE ID = @id";

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

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

        using SqlDataReader reader = command.ExecuteReader();

        if (reader.Read())
        {
            teacher = new Teacher()
            {
                Id = reader.GetInt32(0),
                Firstname = reader.GetString(1),
                Lastname = reader.GetString(2)
            };
        }
    }
}

CourseDAOImpl.cs
public Course? GetCourse(int id)
{
    Course? course = null;

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

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

        string sql = "SELECT * FROM COURSES WHERE ID = @id";

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

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

        using SqlDataReader reader = command.ExecuteReader();

        if (reader.Read())
        {
            course = new Course()
            {
                Id = reader.GetInt32(0),
                Description = reader.GetString(1),
                TeacherId = reader.GetInt32(2)
            };
        }
    }
}

StudentCourseDAOImpl.cs
public void Insert(StudentCourse? studentcourse)
{
    if (studentcourse == null) return;

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

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

        string sql = "INSERT INTO STUDENTS_COURSES " +
                    "(STUDENT_ID, COURSE_ID) VALUES " +
                    "(@student_id, @course_id)";

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

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

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

Η Get Method διαβάζει και επιστρέφει μια στήλη μέσα στον πίνακα

Αντιστοιχεί στο γράμμα R(Read) από το CRUD

GET ALL METHOD

```
StudentDAOImpl.cs
public List<Student> GetAll()
{
    List<Student> students = new List<Student>();

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

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

        string sql = "SELECT * FROM STUDENTS";

        using SqlCommand command = new SqlCommand(sql, conn);
        using SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            Student student = new Student()
            {
                Id = reader.GetInt32(0),
                Firstname = reader.GetString(1),
                Lastname = reader.GetString(2)
            };

            students.Add(student);
        }
    }
}

TeacherDAOImpl.cs
public List<Teacher> GetAll()
{
    List<Teacher> teachers = new List<Teacher>();

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

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

        string sql = "SELECT * FROM TEACHERS";

        using SqlCommand command = new SqlCommand(sql, conn);
        using SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            Teacher teacher = new Teacher()
            {
                Id = reader.GetInt32(0),
                Firstname = reader.GetString(1),
                Lastname = reader.GetString(2)
            };

            teachers.Add(teacher);
        }
    }
}

CourseDAOImpl.cs
public List<Course> GetAll()
{
    List<Course> courses = new List<Course>();

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

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

        string sql = "SELECT * FROM COURSES";

        using SqlCommand command = new SqlCommand(sql, conn);
        using SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            Course course = new Course()
            {
                Id = reader.GetInt32(0),
                Description = reader.GetString(1),
                TeacherId = reader.GetInt32(2)
            };

            courses.Add(course);
        }
    }
}

StudentCourseDAOImpl.cs
public List<StudentCourse> GetAll()
{
    List<StudentCourse> studentcourses = new List<StudentCourse>();

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

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

        string sql = "SELECT * FROM STUDENTS_COURSES";

        using SqlCommand command = new SqlCommand(sql, conn);
        using SqlDataReader reader = command.ExecuteReader();

        while (reader.Read())
        {
            StudentCourse studentcourse = new StudentCourse()
            {
                StudentId = reader.GetInt32(0),
                CourseId = reader.GetInt32(1)
            };

            studentcourses.Add(studentcourse);
        }
    }
}
```

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
public void Update(Student? student)
{
    if (student == null) return;

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

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

        string sql = "UPDATE STUDENTS SET FIRSTNAME = @firstname, " +
                     "LASTNAME = @lastname WHERE ID = @id";

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

        command.Parameters.AddWithValue("@firstname", student.Firstname);
        command.Parameters.AddWithValue("@lastname", student.Lastname);
        command.Parameters.AddWithValue("@id", student.Id);

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}

TeacherDAOImpl.cs
public void Update(Teacher? teacher)
{
    if (teacher == null) return;

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

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

        string sql = "UPDATE TEACHERS SET FIRSTNAME = @firstname, " +
                     "LASTNAME = @lastname WHERE ID = @id";

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

        command.Parameters.AddWithValue("@firstname", teacher.Firstname);
        command.Parameters.AddWithValue("@lastname", teacher.Lastname);
        command.Parameters.AddWithValue("@id", teacher.Id);

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}

CourseDAOImpl.cs
public void Update(Course? course)
{
    if (course == null) return;

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

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

        string sql = "UPDATE COURSES SET DESCRIPTION = @description, " +
                     "TEACHER_ID = @teacherid WHERE ID = @id";

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

        command.Parameters.AddWithValue("@description", course.Description);
        command.Parameters.AddWithValue("@teacherid", course.TeacherId);
        command.Parameters.AddWithValue("@id", course.Id);

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}

StudentCourseDAOImpl.cs
public void Update(StudentCourse? studentcourse)
{
    if (studentcourse == null) return;

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

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

        string sql = "UPDATE STUDENTS_COURSES SET STUDENT_ID = @student_id, " +
                     "COURSE_ID = @course_id";

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

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

        command.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.StackTrace);
        throw;
    }
}
```

Η Update Method τροποποιεί μιά στήλημέσα στον πίνακα

Αντιστοιχεί στο γράμμα U(Update) από το CRUD

DELETE METHOD

```
StudentDAOImpl.cs
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;
    }
}

TeacherDAOImpl.cs
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
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
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;
    }
}
```

H Delete Method διαγράφει μιά στήλη μέσα στον πίνακα

Αντιστοιχεί στο γράμμα D(Delete) από το CRUD

DTO

DTO

- ▷ C# CourseDTO.cs
- ▷ C# StudentCourseDTO.cs
- ▷ C# StudentDTO.cs
- ▷ C# TeacherDTO.cs

StudentDTO.cs X

SevStudentsApp SevStudents

```
1  namespace SevStudentsApp.DTO
2  {
3      public class StudentDTO
4      {
5          public int Id { get; set; }
6          public string? Firstname { get; set; }
7          public string? Lastname { get; set; }
8      }
9  }
```

TeacherDTO.cs X

SevStudentsApp SevTeachers

```
1  namespace SevTeachersApp.DTO
2  {
3      public class TeacherDTO
4      {
5          public int Id { get; set; }
6          public string? Firstname { get; set; }
7          public string? Lastname { get; set; }
8      }
9  }
```

CourseDTO.cs X

SevStudentsApp SevCourses

```
1  namespace SevCoursesApp.DTO
2  {
3      public class CourseDTO
4      {
5          public int Id { get; set; }
6          public string? Description { get; set; }
7          public int TeacherId { get; set; }
8      }
9  }
```

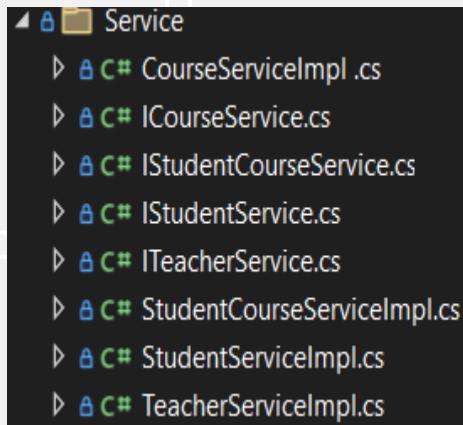
StudentCourseDTO.cs X

SevStudentsApp SevStudentCourses

```
1  namespace SevStudentCoursesApp.DTO
2  {
3      public class StudentCourseDTO
4      {
5          public int StudentId { get; set; }
6          public int CourseId { get; set; }
7      }
8  }
```

SERVICE LAYER

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



The image displays four code editors side-by-side, each showing a public interface definition:

- IStudentService.cs**: Located in the `SevStudentsApp` namespace. It defines methods for getting all students, inserting a student, updating a student, deleting a student, and getting a student by ID.
- ITeacherService.cs**: Located in the `SevTeachersApp` namespace. It defines methods for getting all teachers, inserting a teacher, updating a teacher, deleting a teacher, and getting a teacher by ID.
- ICourseService.cs**: Located in the `SevCoursesApp` namespace. It defines methods for getting all courses, inserting a course, updating a course, deleting a course, and getting a course by ID.
- IStudentCourseService.cs**: Located in the `SevStudentCoursesApp` namespace. It defines methods for getting all student courses, inserting a student course, updating a student course, deleting a student course, and getting a student course by ID.

DEPENDENCY INJECTIONS

To Dependency
Injection κάθε
πίνακα

```
StudentServiceImpl.cs ✎ X
SevStudentsApp
SevStudentsApp.Service
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         }
15     }
16 }
```

```
TeacherServiceImpl.cs ✎ X
SevStudentsApp
SevTeachersApp.Service
1  using SevTeachersApp.DAO;
2  using SevTeachersApp.DTO;
3  using SevTeachersApp.Models;
4
5  namespace SevTeachersApp.Service
6  {
7      public class TeacherServiceImpl : ITeacherService
8      {
9          private readonly ITeacherDAO dao;
10
11         public TeacherServiceImpl(ITeacherDAO dao)
12         {
13             this.dao = dao;
14         }
15     }
16 }
```

```
CourseServiceImpl.cs ✎ X TeacherServiceImpl.cs
SevCoursesApp
SevCoursesApp.Service
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         }
15     }
16 }
```

```
StudentCourseServiceImpl.cs ✎ X
SevStudentCoursesApp
SevStudentCoursesApp.Service
1  using SevStudentCoursesApp.DAO;
2  using SevStudentCoursesApp.DTO;
3  using SevStudentCoursesApp.Models;
4
5  namespace SevStudentCoursesApp.Service
6  {
7      public class StudentCourseServiceImpl : IStudentCourseService
8      {
9          private readonly IStudentCourseDAO dao;
10
11         public StudentCourseServiceImpl(IStudentCourseDAO dao)
12         {
13             this.dao = dao;
14         }
15     }
16 }
```

SERVICE LAYER INSERT METHOD

StudentServiceImpl.cs

```
SevStudentsApp.SevStudentsApp.Service.StudentService
```

```
public Student? GetStudent(int id){}

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.SevTeachersApp.SevTeachersApp.Service.TeacherService
```

```
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

```
SevCoursesApp.SevCoursesApp.Service.CourseService
```

```
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

```
SevStudentCoursesApp.SevStudentCoursesApp.Service.StudentCourseService
```

```
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

```
public Student? GetStudent(int id)
{
    try
    {
        return dao.GetStudent(id);
    } catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

TeacherServiceImpl.cs

```
public Teacher? GetTeacher(int id)
{
    try
    {
        return dao.GetTeacher(id);
    } catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

CourseServiceImpl.cs

```
public Course? GetCourse(int id)
{
    try
    {
        return dao.GetCourse(id);
    } catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

StudentCourseServiceImpl.cs

```
public StudentCourse? GetStudentCourse(int student_id)
{
    try
    {
        return dao.GetStudentCourse(student_id);
    } catch (Exception e)
    {
        Console.WriteLine(e.Message);
        throw;
    }
}
```

SERVICE LAYER GET ALL METHOD

StudentServiceImpl.cs

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

CourseServiceImpl.cs

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

StudentCourseServiceImpl.cs

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

SERVICE LAYER UPDATE METHOD

```
StudentServiceImpl.cs  ✘ X
SevStudentsApp
  ↳ SevStudentsApp

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

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

```
TeacherServiceImpl.cs  X
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  X
SevStudentsApp  SevCoursesApp

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

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

```
StudentCourseServiceImpl.cs  X
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.
16  public Student? DeleteStudent(StudentDTO? dto)
17  {
18      if (dto is null) return null;
19
20      try
21      {
22          Student? student = Convert(dto);
23          return dao.Delete(student);
24      }
25      catch (Exception e)
26      {
27          Console.WriteLine(e.Message);
28          throw;
29      }
30 }
```

TeacherServiceImpl.cs

```
SevStudentsApp      ↴ SevTeachersApp.Service.
16  public Teacher? DeleteTeacher(TeacherDTO? dto)
17  {
18      if (dto is null) return null;
19
20      try
21      {
22          Teacher? teacher = Convert(dto);
23          return dao.Delete(teacher);
24      }
25      catch (Exception e)
26      {
27          Console.WriteLine(e.Message);
28          throw;
29      }
30 }
```

CourseServiceImpl.cs

```
SevStudentsApp      ↴ SevCoursesApp.Service.
16  public Course? DeleteCourse(CourseDTO? dto)
17  {
18      if (dto is null) return null;
19
20      try
21      {
22          Course? course = Convert(dto);
23          return dao.Delete(course);
24      }
25      catch (Exception e)
26      {
27          Console.WriteLine(e.Message);
28          throw;
29      }
30 }
```

StudentCourseServiceImpl.cs

```
SevStudentsApp      ↴ SevStudentCoursesApp.Service.
16  public StudentCourse? DeleteStudentCourse(StudentCourseDTO? dto)
17  {
18      if (dto is null) return null;
19
20      try
21      {
22          StudentCourse? studentcourse = Convert(dto);
23          return dao.Delete(studentcourse);
24      }
25      catch (Exception e)
26      {
27          Console.WriteLine(e.Message);
28          throw;
29      }
30 }
```

SERVICE LAYER CONVERT METHOD

StudentServiceImpl.cs X

```
SevStudentsApp      SevStudentsApp
```

```
    3 references
  72     private Student? Convert(StudentDTO dto)
  73     {
  74
  75         if (dto == null) return null;
  76
  77         return new Student()
  78         {
  79             Id = dto.Id,
  80             Firstname = dto.Firstname,
  81             Lastname = dto.Lastname,
  82         };
  83     }
```

TeacherServiceImpl.cs X

```
SevStudentsApp      SevTeachersApp
```

```
    3 references
  74     private Teacher? Convert(TeacherDTO dto)
  75     {
  76
  77         if (dto == null) return null;
  78
  79         return new Teacher()
  80         {
  81             Id = dto.Id,
  82             Firstname = dto.Firstname,
  83             Lastname = dto.Lastname,
  84         };
  85     }
```

CourseServiceImpl.cs X

```
SevStudentsApp      SevCoursesApp
```

```
    private Course? Convert(CourseDTO dto)
{
}
if (dto == null) return null;
return new Course()
{
    Id = dto.Id,
    Description = dto.Description,
    TeacherId = dto.TeacherId,
};
```

StudentCourseServiceImpl.cs X CourseServiceImpl.cs

```
SevStudentsApp      SevStudentCoursesApp.Service.St
```

```
    private StudentCourse? Convert(StudentCourseDTO dto)
{
}
if (dto == null) return null;
return new StudentCourse()
{
    StudentId = dto.StudentId,
    CourseId = dto.CourseId,
};
```

VALIDATOR

The image shows a code editor with four tabs open, each containing a C# class definition for validation logic:

- StudentValidator.cs**: Validates Student DTO objects. It contains a private constructor and a static Validate method that checks if Firstname or Lastname are less than three characters.
- TeacherValidator.cs**: Validates Teacher DTO objects. It contains a private constructor and a static Validate method that checks if Firstname or Lastname are less than three characters.
- CourseValidator.cs**: Validates Course DTO objects. It contains a private constructor and a static Validate method that checks if Description is less than two characters.
- StudentCourseValidator.cs**: Validates StudentCourse DTO objects. It contains a private constructor and a static Validate method that checks if StudentId is less than two digits.

```
StudentValidator.cs ✘ X
SevStudentsApp Validator.StudentValidator
1  using SevStudentsApp.DTO;
2
3  namespace SevStudentsApp.Validator
4  {
5
6      public class StudentValidator
7      {
8          //No instances of this class should be available
9
10         private StudentValidator() { }
11
12
13         public static string Validate(StudentDTO? dto)
14         {
15             if ((dto!.Firstname!.Length <= 2) ||
16                 (dto!.Lastname!.Length <= 2))
17             {
18                 return "Firstname or Lastname should not be less than three characters";
19             }
20
21         }
22     }
23 }
```

```
TeacherValidator.cs ✘ X
SevTeachersApp Validator.TeacherValidator
1  using SevTeachersApp.DTO;
2
3  namespace SevTeachersApp.Validator
4  {
5
6      public class TeacherValidator
7      {
8          //No instances of this class should be available
9
10         private TeacherValidator() { }
11
12
13         public static string Validate(TeacherDTO? dto)
14         {
15             if ((dto!.Firstname!.Length <= 2) ||
16                 (dto!.Lastname!.Length <= 2))
17             {
18                 return "Firstname or Lastname should not be less than three characters";
19             }
20
21         }
22     }
23 }
```

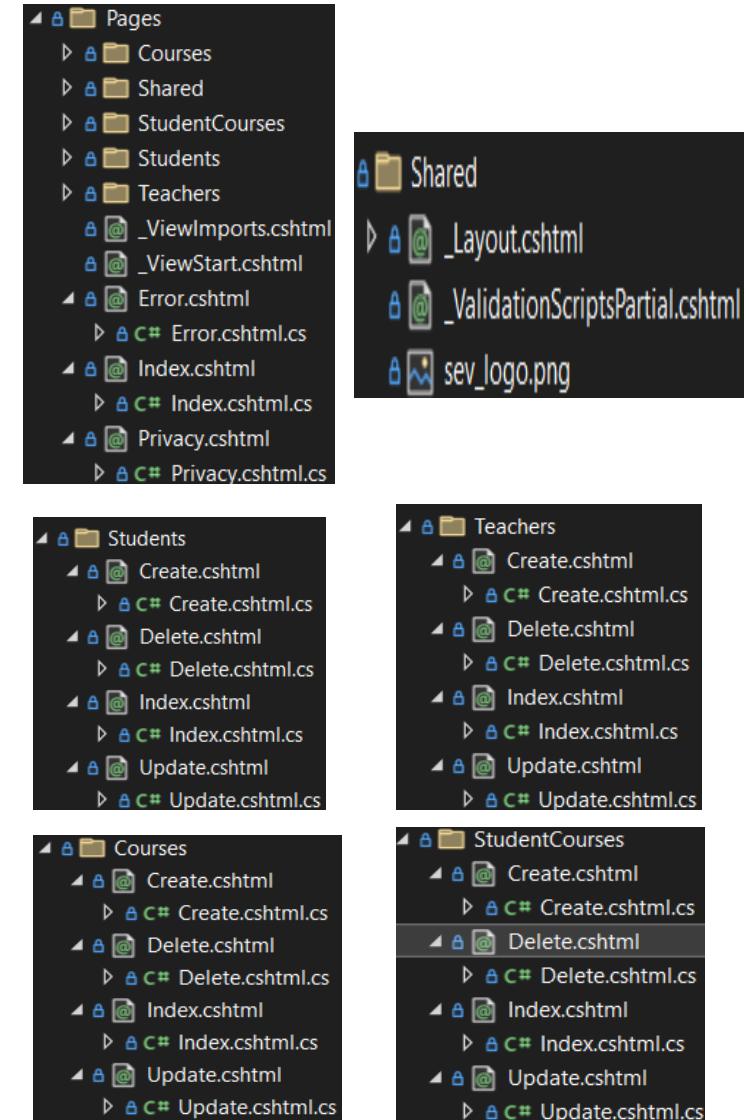
```
CourseValidator.cs ✘ X
SevCoursesApp Validator.CourseValidator
1  using SevCoursesApp.DTO;
2
3  namespace SevCoursesApp.Validator
4  {
5
6      public class CourseValidator
7      {
8          //No instances of this class should be available
9
10         private CourseValidator() { }
11
12
13         public static string Validate(CourseDTO? dto)
14         {
15             if ((dto!.Description!.Length <= 1))
16             {
17                 return "Description should not be less than two characters";
18             }
19
20         }
21     }
22 }
```

```
StudentCourseValidator.cs ✘ X
SevStudentCoursesApp Validator.StudentCourseValidator
1  using SevStudentCoursesApp.DTO;
2
3  namespace SevStudentCoursesApp.Validator
4  {
5
6      public class StudentCourseValidator
7      {
8          //No instances of this class should be available
9
10         private StudentCourseValidator() { }
11
12
13         public static string Validate(StudentCourseDTO? dto)
14         {
15             if ((dto!.StudentId!.ToString().Length <= 1))
16             {
17                 return "Student id should not be less than two digits";
18             }
19
20         }
21     }
22 }
```

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)

```
7  namespace SevStudentsApp.Pages.Students
8  {
9      6 references
10     public class IndexModel : PageModel
11     {
12         private readonly IStudentDAO studentDAO = new StudentDAOImpl();
13         private readonly IStudentService? service;
14
15         internal List<Student> students = new();
16
17         0 references
18         public IndexModel()
19         {
20             service = new StudentServiceImpl(studentDAO);
21         }
22
23         0 references
24         public IActionResult OnGet()
25         {
26             students = service!. GetAllStudents();
27             return Page();
28         }
29     }
30 }
```

```
1 @page
2 @model SevStudentsApp.Pages.Students.IndexModel
3 @{
4     ViewData["Title"] = "Student Index";
5 }
6
7 <h2>List of Students</h2>
8 [
9     <a class="btn btn-primary btn-sm" href="/Students/Create">New Student</a>
10    <div style="background-color: #611ff0">
11        <table class="table table-bordered table-hover text-white">
12            <thead>
13                <tr>
14                    <th>ID</th>
15                    <th>Firstname</th>
16                    <th>Lastname</th></tr>
17            </thead>
18            <tbody>
19                @if (Model.students != null)
20                {
21                    @foreach (var student in Model.students)
22                    {
23                        <tr>
24                            <td>@student.Id</td>
25                            <td>@student.Firstname</td>
26                            <td>@student.Lastname</td>
27                            <td>
28                                <a class="btn btn-outline-light btn-sm"
29                                    href="/Students/Update?id=@student.Id">Update</a>
30                                <a class="btn btn-outline-light btn-sm"
31                                    href="/Students/Delete?id=@student.Id">Delete</a>
32                            </td>
33                        </tr>
34                    }
35                }
36            </tbody>
37        </table>
38    </div>
39
40
```

```
 7  namespace SevStudentsApp.Pages.Students
 8  {
 9      6 references
10      public class CreateModel : PageModel
11      {
12          private readonly IStudentDAO studentDAO = new StudentDAOImpl();
13          private readonly IStudentService service;
14
15          0 references
16          public CreateModel()
17          {
18              service = new StudentServiceImpl(studentDAO);
19          }
20
21          internal StudentDTO studentDto = new();
22          internal string errorMessage = "";
23
24          0 references
25          public void OnGet()
26          {
27
28          0 references
29          public void OnPost()
30          {
31              studentDto.Firstname = Request.Form["firstname"];
32              studentDto.Lastname = Request.Form["lastname"];
33
34              errorMessage = StudentValidator.Validate(studentDto);
35
36              if (!errorMessage.Equals("")) return;
37
38              try
39              {
40                  service.InsertStudent(studentDto);
41                  Response.Redirect("/Students/Index");
42              }
43          }
44      }
45  }
```

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


```

AJK

```
 Create.cshtml
1  @page
2  @model SevStudentsApp.Pages.Students.CreateModel
3  @{
4      ViewData["Title"] = "Student Create";
5  }
6
7  <h2>New Student</h2>
8  @if (!Model.errorMessage.Equals(""))
9  {
10     <h2><strong>@Model.errorMessage</strong></h2>
11 }
12
13
14 <form method="POST">
15     <div class="row mb-3">
16         <label for="firstname" class="col-md-1 col-form-label">First Name</label>
17         <div>
18             <input type="text" class="form-control" name="firstname" id="firstname" placeholder="Enter your first name" value="@Model.studentDto.Firstname" />
19         </div>
20     </div>
21     <div class="row mb-3">
22         <label for="lastname" class="col-md-1 col-form-label">Last Name</label>
23         <div>
24             <input type="text" class="form-control" name="lastname" id="lastname" placeholder="Enter your last name" value="@Model.studentDto.Lastname" />
25         </div>
26     </div>
27     <div class="row mb-3">
28         <div class="col-md-1 d-grid">
29             <button type="submit" class="btn btn-primary">Submit</button>
30         </div>
31         <div class="col-md-1 d-grid">
32             <a href="/" role="button" class="btn btn-primary">Cancel</a>
33         </div>
34     </div>
35 
```

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

The image displays the structure of a .NET Core application for managing student data. It includes:

- Update.cshtml**: A Razor page for updating student information. It features a form with input fields for student ID, first name, and last name, and buttons for submit and cancel.
- UpdateModel.cs**: The C# controller for the update operation. It handles GET requests to fetch a student by ID and POST requests to update the student's information. It also contains a private method to convert a Student object to a StudentDTO.
- Delete.cshtml**: A Razor page for deleting a student. It shows a confirmation message and links for back and cancel operations.
- DeleteModel.cs**: The C# controller for the delete operation. It handles GET requests to get a student by ID and POST requests to delete the student.
- StudentDTO.cs**: A class definition for StudentDTO, which contains properties for Id, Firstname, and Lastname.

```

Update.cshtml.cs
SevStudentsApp
  - Pages
    - Students
      - Update.cshtml
      - UpdateModel.cs
Delete.cshtml
SevStudentsApp
  - Pages
    - Students
      - Delete.cshtml
      - DeleteModel.cs
StudentDTO.cs
SevStudentsApp
  - Pages
    - Students
      - StudentDTO.cs
  
```

Index.cshtml.cs

```

1  using Microsoft.AspNetCore.Mvc;
2  using Microsoft.AspNetCore.Mvc.RazorPages;
3  using SevTeachersApp.DAO;
4  using SevTeachersApp.Models;
5  using SevTeachersApp.Service;
6
7  namespace SevTeachersApp.Pages.Teachers
8  {
9      public class IndexModel : PageModel
10     {
11
12         private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
13         private readonly ITeacherService? service;
14
15         internal List<Teacher> teachers = new();
16
17         public IndexModel()
18         {
19             service = new TeacherServiceImpl(teacherDAO);
20         }
21
22         public IActionResult OnGet()
23         {
24             teachers = service!. GetAllTeachers();
25             return Page();
26         }
27     }
28 }

```

Index.cshtml

```

1 @page
2 @model SevTeachersApp.Pages.Teachers.IndexModel
3 @{
4     ViewData["Title"] = "Teacher Index";
5 }
6
7 <h2>List of Teachers</h2>
8 <a class="btn btn-primary" href="/Teachers/Create">New Teacher</a>
9 <div style="background-color: #6611ff00">
10 <table class="table table-bordered table-hover text-white">
11     <thead>
12         <tr>
13             <th>ID</th>
14             <th>Firstname</th>
15             <th>Lastname</th>
16         </tr>
17     </thead>
18     <tbody>
19         @if (Model.teachers is not null)
20         {
21             @foreach (var teacher in Model.teachers)
22             {
23                 <tr>
24                     <td>@teacher.Id</td>
25                     <td>@teacher.Firstname</td>
26                     <td>@teacher.Lastname</td>
27                     <td>
28                         <a class="btn btn-outline-light btn-sm" href="/Teachers/Update?id=@teacher.Id">Update</a>
29                         <a class="btn btn-outline-light btn-sm" href="/Teachers/Delete?id=@teacher.Id">Delete</a>
30                     </td>
31                 </tr>
32             }
33         }
34     </tbody>
35 </table>
36 </div>

```

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

Create.cshtml.cs

```

7  Create.cshtml.cs
8  namespace SevTeachersApp.Pages.Teachers
9  {
10    public class CreateModel : PageModel
11    {
12        private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
13        private readonly ITeacherService service;
14
15        internal TeacherDTO teacherDto = new();
16        internal string errorMessage = "";
17
18        public CreateModel()
19        {
20            service = new TeacherServiceImpl(teacherDAO);
21        }
22
23        public void OnGet()
24        {
25        }
26
27        public void OnPost()
28        {
29            teacherDto.Firstname = Request.Form["firstname"];
30            teacherDto.Lastname = Request.Form["lastname"];
31
32            errorMessage = TeacherValidator.Validate(teacherDto);
33
34            if (!errorMessage.Equals(""))
35            {
36                try
37                {
38                    service.InsertTeacher(teacherDto);
39                    Response.Redirect("/Teachers/Index");
40                }
41                catch (Exception e)
42                {
43                    errorMessage = e.Message;
44                }
45            }
46        }
47    }
48 }

```

Create.cshtml

```

1 @page
2 @model SevTeachersApp.Pages.Teachers.CreateModel
3 @{
4     ViewData["Title"] = "Teacher Create";
5 }
6
7 <h2>New Teacher</h2>
8 @if (!Model.errorMessage.Equals(""))
9 {
10     <h2><strong>@Model.errorMessage</strong></h2>
11 }
12
13 <form method="POST">
14     <div class="row mb-3">
15         <label for="firstname" class="col-md-1 col-form-label">First Name</label>
16         <div>
17             <input type="text" class="form-control" name="firstname" id="firstname" placeholder="Enter your first name" value="@Model.teacherDto.Firstname" />
18         </div>
19     </div>
20     <div class="row mb-3">
21         <label for="lastname" class="col-md-1 col-form-label">Last Name</label>
22         <div>
23             <input type="text" class="form-control" name="lastname" id="lastname" placeholder="Enter your last name" value="@Model.teacherDto.Lastname" />
24         </div>
25     </div>
26     <div class="row mb-3">
27         <div class="col-md-1 d-grid">
28             <button type="submit" class="btn btn-primary">Submit</button>
29         </div>
30         <div class="col-md-1 d-grid">
31             <a href="/" role="button" class="btn btn-primary">Cancel</a>
32         </div>
33     </div>
34 </form>

```

Update.cshtml.cs

```

SevStudentsApp
  ↳ SevTeachersApp.Pages.Teachers.UpdateModel
    8  namespace SevTeachersApp.Pages.Teachers
    9  {
    10     6 references
    11     public class UpdateModel : PageModel
    12     {
    13         private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
    14         private readonly ITeacherService service;
    15
    16         0 references
    17         public UpdateModel()
    18         {
    19             service = new TeacherServiceImpl(teacherDAO);
    20         }
    21
    22         internal TeacherDTO teacherDto = new();
    23         internal string errorMessage = "";
    24
    25         0 references
    26         public void OnGet()
    27         {
    28             try
    29             {
    30                 Teacher? teacher;
    31
    32                 int id = int.Parse(Request.Query["id"]);
    33                 teacher = service.GetTeacher(id);
    34
    35                 if (teacher != null)
    36                 {
    37                     teacherDto = ConvertToDto(teacher);
    38                 }
    39             catch (Exception e)
    40             {
    41                 errorMessage = e.Message;
    42             }
    43         }
    44
    45         0 references
    46         public void OnPost()
    47         {
    48             errorMessage = "";
    49             // Get DTO
    50             teacherDto.Id = int.Parse(Request.Form["id"]);
    51             teacherDto.Firstname = Request.Form["firstname"];
    52             teacherDto.Lastname = Request.Form["lastname"];
    53
    54             // validate
    55             errorMessage = TeacherValidator.Validate(teacherDto);
    56
    57             if (!errorMessage.Equals("")) return;
    58
    59             try
    60             {
    61                 service.UpdateTeacher(teacherDto);
    62                 Response.Redirect("/Teachers/Index");
    63             }
    64             catch (Exception e)
    65             {
    66                 errorMessage = e.Message;
    67             }
    68         }
    69
    70     }
    71
    72     1 reference
    73     private TeacherDTO ConvertToDto(Teacher teacher)
    74     {
    75         return new TeacherDTO()
    76         {
    77             Id = teacher.Id,
    78             Firstname = teacher.Firstname,
    79             Lastname = teacher.Lastname
    80         };
    81     }
    82
    83 }

```

Delete.cshtml.cs

```

SevStudentsApp
  ↳ SevTeachersApp.Pages.Teachers.DeleteModel
    1  @page
    2  @model SevTeachersApp.Pages.Teachers.DeleteModel
    3  @{
    4      ViewData["Title"] = "Teacher Delete";
    5
    6  }
    7
    8  @if (!Model.errorMessage.Equals(""))
    9  {
    10     <h2><strong>@Model.errorMessage</strong></h2>
    11  }

```

Update.cshtml

```

1  @page
2  @model SevTeachersApp.Pages.Teachers.UpdateModel
3  @{
4      ViewData["Title"] = "Teacher Update";
5
6  <h2>New Teacher</h2>
7  @if (!Model.errorMessage.Equals(""))
8  {
9      <h2><strong>@Model.errorMessage</strong></h2>
10 }
11
12 <form method="POST">
13     <div class="row mb-3">
14         <input type="hidden" class="form-control" name="id"
15                         value="@Model.teacherDto.Id" />
16
17     </div>
18     <div class="row mb-3">
19         <label for="firstname" class="col-md-1 col-form-label">Όνομα</label>
20         <div>
21             <input type="text" class="form-control"
22                         name="firstname" id="firstname" placeholder="Enter firstname"
23                         value="@Model.teacherDto.Firstname" />
24         </div>
25     </div>
26     <div class="row mb-3">
27         <label for="lastname" class="col-md-1 col-form-label">Επώνυμο</label>
28         <div>
29             <input type="text" class="form-control"
30                         name="lastname" id="lastname" placeholder="Enter lastname"
31                         value="@Model.teacherDto.Lastname" />
32         </div>
33     </div>
34     <div class="row mb-3">
35         <div class="offset-md-1 col-md-3 d-grid">
36             <button type="submit" class="btn btn-primary">Submit</button>
37         </div>
38         <div class="offset-md-1 col-md-3 d-grid">
39             <a href="/" role="button" class="btn btn-primary">Cancel</a>
40         </div>
41     </div>
42
43 </form>

```

Delete.cshtml

```

SevStudentsApp
  ↳ SevTeachersApp.Pages.Teachers.DeleteModel
    7  namespace SevTeachersApp.Pages.Teachers
    8  {
    9      6 references
    10     public class DeleteModel : PageModel
    11     {
    12         private readonly ITeacherDAO teacherDAO = new TeacherDAOImpl();
    13         private readonly ITeacherService? service;
    14
    15         internal string errorMessage = "";
    16
    17         0 references
    18         public DeleteModel()
    19         {
    20             service = new TeacherServiceImpl(teacherDAO);
    21         }
    22
    23         0 references
    24         public void OnGet()
    25         {
    26             TeacherDTO teacherDTO = new();
    27
    28             try
    29             {
    30                 Teacher? teacher;
    31
    32                 int id = int.Parse(Request.Query["id"]);
    33
    34                 teacherDTO.Id = id;
    35                 teacher = service?.DeleteTeacher(teacherDTO);
    36                 Response.Redirect("/Teachers/Index");
    37             }
    38             catch (Exception e)
    39             {
    40                 errorMessage = e.Message;
    41             }
    42         }
    43     }

```

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



```

Index.cshtml.cs  # X
SevStudentsApp
    SevCoursesApp.Pages.Courses.IndexModel
        7  namespace SevCoursesApp.Pages.Courses
        8  {
        9      public class IndexModel : PageModel
        10     {
        11         private readonly ICourseDAO courseDAO = new CourseDAOImpl();
        12         private readonly ICourseService? service;
        13
        14         internal List<Course> courses = new();
        15
        16         public IndexModel()
        17         {
        18             service = new CourseServiceImpl(courseDAO);
        19         }
        20
        21         public IActionResult OnGet()
        22         {
        23             courses = service!.GetAllCourses();
        24             return Page();
        25         }
        26     }
        27 }
        28

Create.cshtml.cs  # X
SevStudentsApp
    SevCoursesApp.Pages.Courses.CreateModel
        7  namespace SevCoursesApp.Pages.Courses
        8  {
        9      public class CreateModel : PageModel
        10     {
        11         private readonly ICourseDAO courseDAO = new CourseDAOImpl();
        12         private readonly ICourseService service;
        13
        14         public CreateModel()
        15         {
        16             service = new CourseServiceImpl(courseDAO);
        17         }
        18
        19         internal CourseDTO courseDto = new();
        20         internal string errorMessage = "";
        21
        22         public void OnGet()
        23         {
        24         }
        25
        26         public void OnPost()
        27         {
        28             courseDto.Description = Request.Form["description"];
        29             courseDto.TeacherId = int.Parse(Request.Form["teacherid"]);
        30             errorMessage = CourseValidator.Validate(courseDto);
        31             if (!errorMessage.Equals(""))
        32             {
        33                 try
        34                 {
        35                     service.InsertCourse(courseDto);
        36                     Response.Redirect("/Courses/Index");
        37                 }
        38                 catch (Exception e)
        39                 {
        40                     errorMessage = e.Message;
        41                 }
        42             }
        43         }
        44     }
        45 }
        46
        47
        48

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


## List of Courses

New Course

| ID | Description | Teacher ID |
|----|-------------|------------|
|----|-------------|------------|


```

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

```

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


## List of Courses

New Course

| ID | Description | Teacher ID |
|----|-------------|------------|
|----|-------------|------------|


```

```

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


## New Course


@if (!Model.errorMessage.Equals(""))
{
    <h2><strong>@Model.errorMessage</strong></h2>
}
<form method="POST">


<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>



<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 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>


</form>

```

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

```
Update.cshtml.cs
```

```
SevStudentsApp
  8  namespace SevCoursesApp.Pages.Courses
  9  {
 10     public class UpdateModel : PageModel
 11     {
 12         private readonly ICourseDAO courseDAO = new CourseDAOImpl();
 13         private readonly ICourseService service;
 14
 15         public UpdateModel()
 16         {
 17             service = new CourseServiceImpl(courseDAO);
 18         }
 19
 20         internal CourseDTO courseDto = new();
 21         internal string errorMessage = "";
 22
 23         public void OnGet()
 24         {
 25             try
 26             {
 27                 Course? course;
 28                 int id = int.Parse(Request.Query["id"]);
 29                 course = service.GetCourse(id);
 30
 31                 if (course != null)
 32                 {
 33                     courseDto = ConvertToDto(course);
 34                 }
 35             catch (Exception e)
 36             {
 37                 errorMessage = e.Message;
 38                 return;
 39             }
 40
 41             errorMessage = "";
 42
 43             public void OnPost()
 44             {
 45                 // Get DTO
 46                 courseDto.Id = int.Parse(Request.Form["id"]);
 47                 courseDto.Description = Request.Form["description"];
 48                 courseDto.TeacherId = int.Parse(Request.Form["teacherid"]);
 49
 50                 // validate
 51                 errorMessage = CourseValidator.Validate(courseDto);
 52
 53                 if (errorMessage.Equals("")) return;
 54
 55                 try
 56                 {
 57                     service.UpdateCourse(courseDto);
 58                     Response.Redirect("/Courses/Index");
 59                 }
 60                 catch (Exception e)
 61                 {
 62                     errorMessage = e.Message;
 63                     return;
 64                 }
 65
 66             }
 67
 68             private CourseDTO ConvertToDto(Course course)
 69             {
 70                 return new CourseDTO()
 71                 {
 72                     Id = course.Id,
 73                     Description = course.Description,
 74                     TeacherId = course.TeacherId
 75                 };
 76             }
 77
 78         }
 79
 80     }
 81
 82 }
```

```
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">


<input type="hidden" class="form-control" name="id" value="@Model.courseDto.Id" />



<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>



<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 class="row mb-3">
    <button type="submit" class="btn btn-primary">Submit</button>
</div>
<div class="row mb-3">
    <a href="/" role="button" class="btn btn-primary">Cancel</a>
</div>
</form>
```

```
Delete.cshtml.cs
```

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

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

```
Delete.cshtml.cs
```

```
SevStudentsApp
  7  namespace SevCoursesApp.Pages.Courses
  8  {
  9      public class DeleteModel : PageModel
 10      {
 11          private readonly ICourseDAO courseDAO = new CourseDAOImpl();
 12          private readonly ICourseService? service;
 13
 14          internal string errorMessage = "";
 15
 16          public DeleteModel()
 17          {
 18              service = new CourseServiceImpl(courseDAO);
 19          }
 20
 21          public void OnGet()
 22          {
 23              CourseDTO courseDTO = new();
 24
 25              try
 26              {
 27                  Course? course;
 28                  int id = int.Parse(Request.Query["id"]);
 29
 30                  courseDTO.Id = id;
 31                  course = service?.DeleteCourse(courseDTO);
 32                  Response.Redirect("/Courses/Index");
 33
 34              }
 35              catch (Exception e)
 36              {
 37                  errorMessage = e.Message;
 38                  return;
 39              }
 40
 41          }
 42
 43      }
 44
 45  }
```

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

```

7  namespace SevStudentCoursesApp.Pages.StudentCourses
8  {
9      public class IndexModel : PageModel
10     {
11         private readonly IStudentCourseDAO studentcourseDAO= new StudentCourseDAOImpl();
12         private readonly IStudentCourseService? service;
13
14         internal List<StudentCourse> studentcourses = new();
15
16         0 references
17         public IndexModel()
18         {
19             service = new StudentCourseServiceImpl(studentcourseDAO);
20         }
21
22         0 references
23         public IActionResult OnGet()
24         {
25             studentcourses = service!. GetAllStudentCourses();
26             return Page();
27         }
28     }

```

```

1  @page
2  @model SevStudentCoursesApp.Pages.StudentCourses.IndexModel
3  @{
4      ViewData["Title"] = "Student Course Index";
5  }
6
7  <h2>List of Student Courses</h2>
8  <a class="btn btn-primary btn-sm" href="/StudentCourses/Create">New Student Course</a>
9  <div style="background: #611ff0">
10 <table class="table table-bordered table-hover text-white">
11 <thead>
12 <tr>
13 <th>Student ID</th>
14 <th>Course ID</th>
15 </tr>
16 </thead>
17 <tbody>
18     @if (Model.studentcourses is not null)
19     {
20         @foreach (var studentcourse in Model.studentcourses)
21         {
22             <tr>
23                 <td>@studentcourse.StudentId</td>
24                 <td>@studentcourse.CourseId</td>
25                 <td>
26                     <a class="btn btn-outline-light btn-sm" href="/StudentCourses/Update?student_id=@studentcourse.StudentId">Update</a>
27                     <a class="btn btn-outline-light btn-sm" href="/StudentCourses/Delete?student_id=@studentcourse.StudentId">Delete</a>
28                 </td>
29             </tr>
30         }
31     }
32     </tbody>
33 </table>
34 </div>
35
36
37
38
39
40

```

```

7  namespace SevStudentCoursesApp.Pages.StudentCourses
8  {
9      6 references
10     public class CreateModel : PageModel
11     {
12         private readonly IStudentCourseDAO studentcourseDAO = new StudentCourseDAOImpl();
13         private readonly IStudentCourseService service;
14
15         0 references
16         public CreateModel()
17         {
18             service = new StudentCourseServiceImpl(studentcourseDAO);
19         }
20
21         internal StudentCourseDTO studentcourseDto = new();
22         internal string errorMessage = "";
23
24         0 references
25         public void OnGet()
26         {
27         }
28
29         0 references
30         public void OnPost()
31         {
32             studentcourseDto.StudentId = int.Parse(Request.Form["student_id"]);
33             studentcourseDto.CourseId = int.Parse(Request.Form["course_id"]);
34
35             errorMessage = StudentCourseValidator.Validate(studentcourseDto);
36
37             if (!errorMessage.Equals(""))
38             {
39                 try
40                 {
41                     service.InsertStudentCourse(studentcourseDto);
42                     Response.Redirect("/StudentCourses/Index");
43                 }
44                 catch (Exception e)
45                 {
46                     errorMessage = e.Message;
47                 }
48             }
49         }
50     }
51
52
53
54
55
56
57
58
59
59

```

```

1  @page
2  @model SevStudentCoursesApp.Pages.StudentCourses.CreateModel
3  @{
4      ViewData["Title"] = "Student Course Create";
5  }
6
7  <h2>New Student Course</h2>
8  @if (!Model.errorMessage.Equals(""))
9  {
10     <h2><strong>@Model.errorMessage</strong></h2>
11 }
12
13 <form method="POST">
14     <div class="row mb-3">
15         <label for="student_id" class="col-md-1 col-form-label">Student ID</label>
16         <div>
17             <input type="number" class="form-control" name="student_id" id="student_id" placeholder="Enter student id" value="@Model.studentcourseDto.StudentId" />
18         </div>
19     </div>
20     <div class="row mb-3">
21         <label for="course_id" class="col-md-1 col-form-label">Course ID</label>
22         <div>
23             <input type="number" class="form-control" name="course_id" id="course_id" placeholder="Enter course_id" value="@Model.studentcourseDto.CourseId" />
24             <input type="number" class="form-control" name="course_id" id="course_id" placeholder="Enter course_id" value="@Model.studentcourseDto.CourseId" />
25         </div>
26     </div>
27     <div class="row mb-3">
28         <div class="col-md-1 d-grid">
29             <button type="submit" class="btn btn-primary">Submit</button>
30         </div>
31         <div class="col-md-1 d-grid">
32             <a href="/" role="button" class="btn btn-primary">Cancel</a>
33         </div>
34     </div>
35 </form>
36
37
38
39

```

Update.cshtml.cs

```

    8  namespace SevStudentCoursesApp.Pages.StudentCourses
    9  {
        6 references
       10 public class UpdateModel : PageModel
        {
        }

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

        0 references
       17 public UpdateModel()
       18 {
            service = new StudentCourseServiceImpl(studentcourseDAO);
        }

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

        0 references
       26 public void OnGet()
       27 {
            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
       47 public void OnPost()
       48 {
            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
       72 private StudentCourseDTO ConvertToDto(StudentCourse studentcourse)
       73 {
            return new StudentCourseDTO()
            {
                StudentId = studentcourse.StudentId,
                CourseId = studentcourse.CourseId
            };
        }
    }

```

Update.cshtml

```

    1  @page
    2  @model SevStudentCoursesApp.Pages.StudentCourses.UpdateModel
    3  @{
        ViewData["Title"] = "Student Course Update";
    }

    7  <h2>New Student Course</h2>
    8  @if (!Model.errorMessage.Equals(""))
    9  {
        <h2><strong>@Model.errorMessage</strong></h2>
    }

    13 <form method="POST">
    14 <div class="row mb-3">
    15     <label for="student_id" class="col-md-1 col-form-label">Κωδικός Μαθητή</label>
    16     <div>
    17         <input type="number" class="form-control"
    18             name="student_id" id="student_id" placeholder="Enter Student Id"
    19             value="@Model.studentcourseDto.StudentId" />
    20     </div>
    21 </div>
    22 <div class="row mb-3">
    23     <label for="course_id" class="col-md-1 col-form-label">Κωδικός Μαθήματος</label>
    24     <div>
    25         <input type="number" class="form-control"
    26             name="course_id" id="course_id" placeholder="Enter Course Id"
    27             value="@Model.studentcourseDto.CourseId" />
    28     </div>
    29 </div>
    30 <div class="row mb-3">
    31     <div class="offset-md-1 col-md-3 d-grid">
    32         <button type="submit" class="btn btn-primary">Submit</button>
    33     </div>
    34     <div class="offset-md-1 col-md-3 d-grid">
    35         <a href="/" role="button" class="btn btn-primary">Cancel</a>
    36     </div>
    37 </div>
    38 </form>

```

Delete.cshtml

```

    1  @page
    2  @model SevStudentCoursesApp.Pages.StudentCourses.DeleteModel
    3  @{
        ViewData["Title"] = "Student Course Delete";
    }

    8  @if (!Model.errorMessage.Equals(""))
    9  {
        <h2><strong>@Model.errorMessage</strong></h2>
    }

    11 </div>

```

Delete.cshtml.cs

```

    7  namespace SevStudentCoursesApp.Pages.StudentCourses
    8  {
        6 references
       9 public class DeleteModel : PageModel
        {
        }

        private readonly IStudentCourseDAO studentcourseDAO = new StudentCourseDAOImpl();
        private readonly IStudentCourseService service;
        interface SevStudentCoursesApp.DAO.IStudentCourseDAO
        {
        }

        internal string errorMessage = "";

        0 references
       16 public DeleteModel()
       17 {
            service = new StudentCourseServiceImpl(studentcourseDAO);
        }

        0 references
       21 public void OnGet()
       22 {
            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

```
Layout.cshtml* + X
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="utf-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>@ ViewData["Title"] - SevStudentsApp</title>
7      <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
8      <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
9      <link rel="stylesheet" href="~/SevStudentsApp.styles.css" asp-append-version="true" />
10     </head>
11     <body>
12         <header>
13             <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-primary border-bottom box-shadow mb-3" navbar-light"
14                 style="background: linear-gradient(to right, #D4F1F4, #32abde, #0e5aa8, #2e3789);"
15                 >
16                 <div class="container">
17                     <a class="navbar-brand text-light" href="https://www.sev.org.gr" ></a>
18                     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse"
19                         aria-controls="navbarSupportedContent"
20                         aria-expanded="false" aria-label="Toggle navigation">
21                         <span class="navbar-toggler-icon"></span>
22                     </button>
23                     <div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
24                         <ul class="navbar-nav flex-grow-1">
25                             <li class="nav-item">
26                                 <a class="nav-link text-light" asp-area="" asp-page="/Index">Home</a>
27                             <li class="nav-item">
28                                 <a class="nav-link text-light" asp-area="" asp-page="/Students/Index">Students</a>
29                             </li>
30                             <li class="nav-item">
31                                 <a class="nav-link text-light" asp-area="" asp-page="/Teachers/Index">Teachers</a>
32                             </li>
33                             <li class="nav-item">
34                                 <a class="nav-link text-light" asp-area="" asp-page="/Courses/Index">Courses</a>
35                             </li>
36                             <li class="nav-item">
37                                 <a class="nav-link text-white" asp-area="" asp-page="/StudentCourses/Index">Student Courses</a>
38                             </li>
39                             <li class="nav-item">
40                                 <a class="nav-link text-light" asp-area="" asp-page="/Privacy">Privacy</a>
41                             </li>
42                         </ul>
43                     </div>
44                 </nav>
45             </header>
46             <div class="container">
47                 <main role="main" class="pb-3">
48                     @RenderBody()
49                 </main>
50             </div>
51
52             <footer class="bg-primary border-top footer text-center text-light"
53                 style="background: linear-gradient(to right, #D4F1F4, #32abde, #0e5aa8, #2e3789);"
54                 >
55                 <div class="container">
56                     &copy; 2022 Copyright: Sev
57                 </div>
58             </footer>
59
60             <script src="~/lib/jquery/dist/jquery.min.js"></script>
61             <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
62             <script src="~/js/site.js" asp-append-version="true"></script>
63
64             @await RenderSectionAsync("Scripts", required: false)
65
66         </body>
67     </html>
```

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

- Αρχική σελίδα εμφάνιση



Hellenic Federation of Enterprises

[Home](#) [Students](#) [Teachers](#) [Courses](#) [Student Courses](#) [Privacy](#)

Welcome to SEV



Who we are

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.



What we do

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.



Why join us

SEV corporate membership gives members the opportunity to gain access to a wide range of business services including tax, legal, export and HR advice as well as B2B networking opportunities to grow their business.

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

- Αρχική σελίδα κώδικας

```
1 @page
2 @model IndexModel
3 @{
4     ViewData["Title"] = "Home Page";
5 }
6
7 <div class="text-center text-white">
8     <div style="background: #808080">
9         <h1 class="display-4">Welcome to SEV</h1>
10    </div>
11 </div>
12 <div class="container">
13     <div class="row g-3">
14         <div class="col-12 col-md-6 col-lg-4">
15             <div class="card">
16                 <div style="background: #32abde">
17                     <a href="https://en.sev.org.gr/who-we-are"><img src "~/css/sev_building.jpg" alt="Sev-Grid1" class="card-img-top"></a>
18                     <div class="card-body">
19                         <h5 class="card-title text-white">Who we are</h5>
20                         <p class="card-text text-white" >
21                             SEV - Hellenic Federation of Enterprises has consistently fostered business development in Greece since 1907.
22                             It is the independent voice of businesses, representing a broad spectrum of the country's economic activity.
23                         </p>
24                     </div>
25                 </div>
26             </div>
27         </div>
28         <div class="col-12 col-md-6 col-lg-4">
29             <div class="card">
30                 <div style="background: #32abde">
31                     <a href="https://en.sev.org.gr/sev-business-priorities/"><img src "~/css/sev_conference.png" alt="Sev-Grid2" class="card-img-top"></a>
32                     <div class="card-body">
33                         <h5 class="card-title text-white">What we do</h5>
34                         <p class="card-text text-white" >
35                             We envision Greece as the dynamic center of the European region, with solid institutions and an attractive economic
36                             and social environment for investments. We promote sustainable private sector led growth and job creation.
37                         </p>
38                     </div>
39                 </div>
40             </div>
41         </div>
42         <div class="col-12 col-md-6 col-lg-4">
43             <div class="card">
44                 <div style="background: #32abde">
45                     <a href="https://en.sev.org.gr/why-join-us"><img src "~/css/sev_skills4jobs.jpg" alt="Sev-Grid3" class="card-img-top"></a>
46                     <div class="card-body">
47                         <h5 class="card-title text-white">Why join us</h5>
48                         <p class="card-text text-white" >
49                             SEV corporate membership gives members the opportunity to gain access to a wide range of business services :
50                             legal, export and HR advice as well as B2B networking opportunities to grow their business.
51                         </p>
52                     </div>
53                 </div>
54             </div>
55         </div>
56     </div>
57 </div>
58 </div>
59
```

Privacy Policy

Sev operates the <https://localhost:7232/Index> website, which provides the SERVICE.

This page is used to inform website visitors regarding our policies with the collection, use, and disclosure of Personal Information if anyone decided to use our Service, the Sev website.

If you choose to use our Service, then you agree to the collection and use of information in relation with this policy. The Personal Information that we collect are used for providing and improving the Service. We will not use or share your information with anyone except as described in this Privacy Policy.

The terms used in this Privacy Policy have the same meanings as in our Terms and Conditions, which is accessible at <https://localhost:7232/Index>, unless otherwise defined in this Privacy Policy.

Information Collection and Use

For a better experience while using our Service, we may require you to provide us with certain personally identifiable information, including but not limited to your name, phone number, and postal address. The information that we collect will be used to contact or identify you.

Log Data

We want to inform you that whenever you visit our Service, we collect information that your browser sends to us that is called Log Data. This Log Data may include information such as your computer's Internet Protocol ("IP") address, browser version, pages of our Service that you visit, the time and date of your visit, the time spent on those pages, and other statistics.

Cookies

Cookies are files with small amount of data that is commonly used an anonymous unique identifier. These are sent to your browser from the website that you visit and are stored on your computer's hard drive.

Our website uses these "cookies" to collection information and to improve our Service. You have the option to either accept or refuse these cookies, and know when a cookie is being sent to your computer. If you choose to refuse our cookies, you may not be able to use some portions of our Service.

Service Providers

We may employ third-party companies and individuals due to the following reasons:

- To facilitate our Service;
- To provide the Service on our behalf;
- To perform Service-related services; or
- To assist us in analyzing how our Service is used.

We want to inform our Service users that these third parties have access to your Personal Information. The reason is to perform the tasks assigned to them on our behalf. However, they are obligated not to disclose or use the information for any other purpose.

Security

We value your trust in providing us your Personal Information, thus we are striving to use commercially acceptable means of protecting it. But remember that no method of transmission over the internet, or method of electronic storage is 100% secure and reliable, and we cannot guarantee its absolute security.

Links to Other Sites

Our Service may contain links to other sites. If you click on a third-party link, you will be directed to that site. Note that these external sites are not operated by us. Therefore, we strongly advise you to review the Privacy Policy of these websites. We have no control over, and assume no responsibility for the content, privacy policies, or practices of any third-party sites or services.

Children's Privacy

Our Services do not address anyone under the age of 13. We do not knowingly collect personal identifiable information from children under 13. In the case we discover that a child under 13 has provided us with personal information, we immediately delete this from our servers. If you are a parent or guardian and you are aware that your child has provided us with personal information, please contact us so that we will be able to do necessary actions.

Changes to This Privacy Policy

We may update our Privacy Policy from time to time. Thus, we advise you to review this page periodically for any changes. We will notify you of any changes by posting the new Privacy Policy on this page. These changes are effective immediately, after they are posted on this page.

Contact Us

If you have any questions or suggestions about our Privacy Policy, do not hesitate to contact us.

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

- Privacy Policy εμφάνιση

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

© 2022 Copyright Sev

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

© 2022 Copyright Sev

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

© 2022 Copyright Sev

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

INDEX STUDENTS, TEACHERS, COURSES,

STUDENT COURSES

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

© 2022 Copyright Sev

New Student

First Name

Laura

Last Name

Kinney

[Submit](#)

[Cancel](#)

New Teacher

First Name

Peter

Last Name

Quill

[Submit](#)

[Cancel](#)

© 2022 Copyright: Sev

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

© 2022 Copyright: Sev

New Course

Description

3ds Max

Teacher ID

5d

[Submit](#)

[Cancel](#)

New Student Course

Student ID

56

Course ID

100

[Submit](#)

[Cancel](#)

© 2022 Copyright: Sev

© 2022 Copyright: Sev

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

© 2022 Copyright: Sev

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ CREATE STUDENTS, TEACHERS, COURSES, STUDENT COURSES(2) ΔΗΜΙΟΥΡΓΗΘΗΚΕ

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

© 2022 Copyright: Sev

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

© 2022 Copyright: Sev

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

© 2022 Copyright: Sev

New Student

Όνομα

Laura

Επώνυμο

Kinney Howlet

[Submit](#)[Cancel](#)

New Teacher

Όνομα

Peter Jason

Επώνυμο

Quill

[Submit](#)[Cancel](#)

© 2022 Copyright: Sev

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ

UPDATE STUDENTS, TEACHERS, COURSES (1) ΔΗΜΙΟΥΡΓΙΑ

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

```
bpy.context.scene  
print("Selected"  
#mirror_ob.  
bone = bpy.context
```

New Course

Περιγραφή

3ds Max I

Κωδικός

Καθηγητή

56

[Submit](#)[Cancel](#)

© 2022 Copyright: Sev

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

© 2022 Copyright: Sev

ΕΠΙΔΕΙΞΗ ΚΑΙ ΕΛΕΓΧΟΣ
UPDATE STUDENTS, TEACHERS, COURSES (2)
ΔΗΜΙΟΥΡΓΗΘΚΕ

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

© 2022 Copyright: Sev

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

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

© 2022 Copyright Sev

DELETE STUDENTS, TEACHERS, COURSES, STUDENT COURSES

© 2022 Copyright Sev

Πατάμε το κουμπί delete για να διαγράψουμε το
row που θέλουμε

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

© 2022 Copyright Sev

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

© 2022 Copyright Sev

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	

© 2022 Copyright Sev

New Student

Firstname or Lastname should not be less than three characters

First Name

Enter your first name

Last Name

Enter your last name

[Submit](#)

[Cancel](#)

New Teacher

Firstname or Lastname should not be less than three characters

First Name

Enter your first name

Last Name

Enter your last name

[Submit](#)

[Cancel](#)

© 2022 Copyright: Sev

VALIDATOR STUDENTS, TEACHERS, COURSES, STUDENT COURSES

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

New Course

Description should not be less than two characters

Description

Enter your course description

Teacher ID

0

[Submit](#)

[Cancel](#)

New Student Course

Student ID and Course ID should not be less than one digit

Student ID

0

Course ID

0

[Submit](#)

[Cancel](#)

© 2022 Copyright: Sev

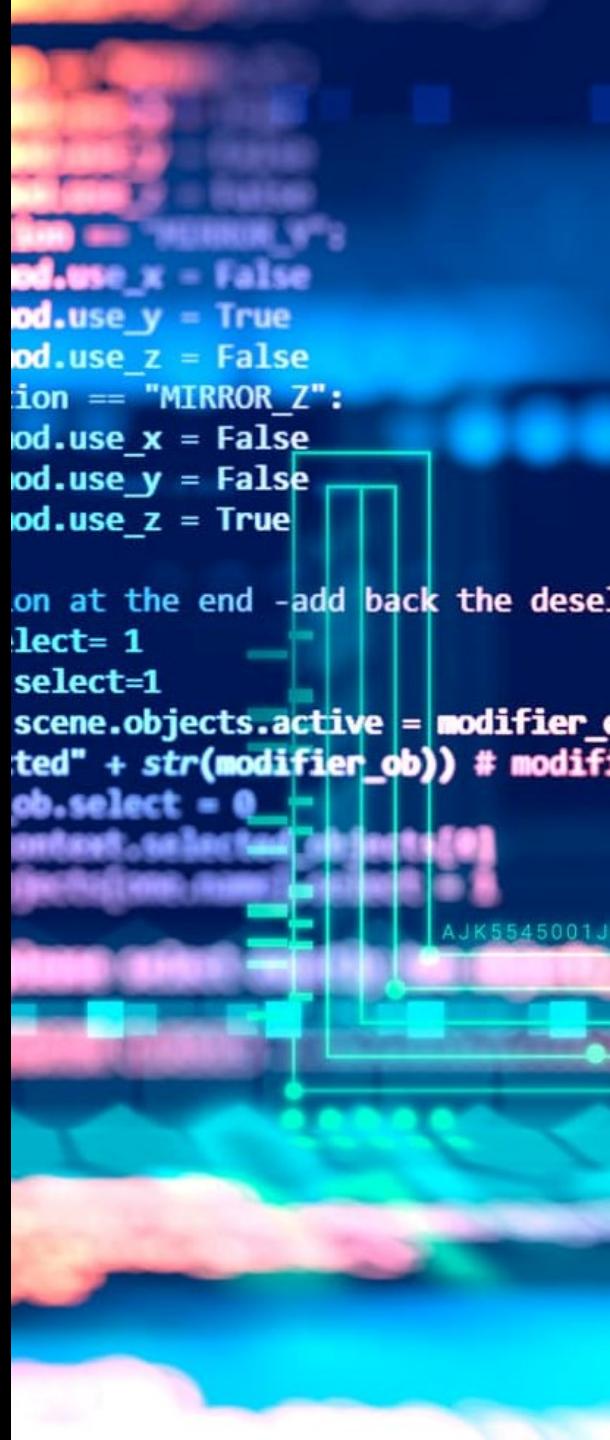
© 2022 Copyright: Sev

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>

