
STUDENTS MANAGEMENT APP

A PROJECT MADE IN THE CONTEXT OF SEV – REGENERATION
SKILLS4JOBS INITIATIVE AT UNIVERSITY OF ECONOMICS AND
BUSINESS FOR SOFTWARE ENGINEERS

BEGIANA ELMAZAI

INTRODUCTION

There was a need to create a web Application to manage the enrollments of students in courses and the teachers teaching each course.

So , I used ASP.NET CORE framework coding with C# , as this can integrate a unified solution of our code/business logic and graphical interfaces.

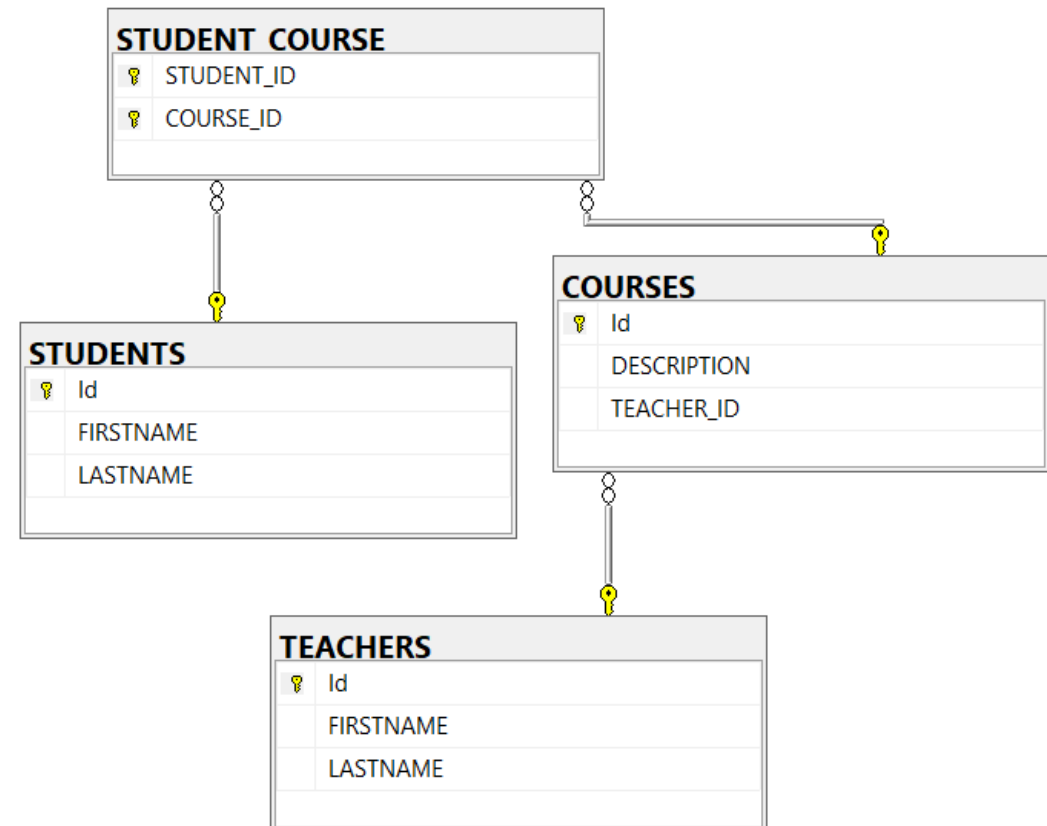
APPLICATION COMPONENTS

1. A DATABASE AT SQL SERVER MANAGEMENT STUDIO,
2. C# code
3. Razor pages


I. DATABASE

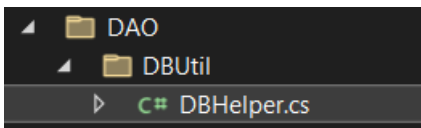
Consist's of four tables

- STUDENTS (**Id**, FIRSTNAME, LASTNAME) [**Id Primary Key**]
- TEACHERS (**Id**, FIRSTNAME, LASTNAME) [**Id Primary Key**]
- COURSES (**Id**, DESCRIPTION, **TEACHER_ID**) [**Id Primary Key**]
TEACHER_ID Foreing key to TEACHERS Table at **Id**
- STUDENT_COURSE (**STUDENT_ID** , **COURSE_ID**)
[Both fields compose **Primary Key** and
STUDENT_ID Foreing key to STUDENTS Table at **Id**
COURSE_ID Foreing key to COURSES Table at **Id**]



I. DATABASE CONNECTION WITH VS CODE

 **System.Data.SqlClient** by Microsoft 4.8.4
Provides the data provider for SQL Server. These classes provide access to versions of SQL Server and encapsulate database-specific protocols, inclu...



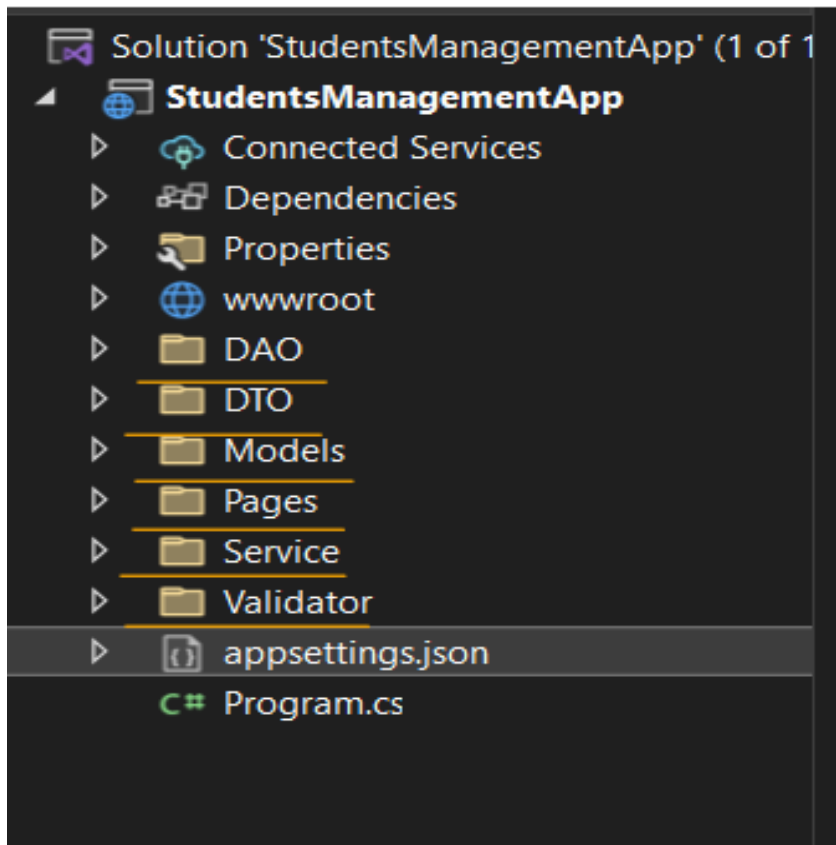
Connection with DB is done by Installing Nuget Package **SQLClient** and provided the classes and interfaces that create connection with database, command queries, read or send data.

In this project, DB Helper Class attempts the connection to by sending a connection string.

```
1 using System.Data.SqlClient;
2
3 namespace StudentsManagementApp.DAO.DBUtil
4 {
5     23 references
6     public class DBHelper
7     {
8         private static SqlConnection? conn;
9
10        //No instances of this class available .UTILITY CLASS
11        0 references
12        private DBHelper() { }
13
14        22 references
15        public static SqlConnection? GetConnection()
16        {
17            try
18            {
19                ConfigurationManager configurationManager = new();
20                configurationManager.AddJsonFile("appsettings.json");
21                string url = configurationManager.GetConnectionString("DefaultConnection");
22                //string url = "Data Source=localhost\\sqlexpress;Initial Catalog=SevDB;Integrated Security=True";
23                conn = new SqlConnection(url);
24                return conn;
25            }
26            catch (Exception e)
27            {
28                Console.WriteLine(e.StackTrace);
29                return null;
30            }
31        }
32
33        0 references
34        public static void CloseConnection()
35        {
36            if (conn != null)
37            {
38                conn.Close();
39            }
40        }
41    }
42 }
```

2. CODE DESIGN

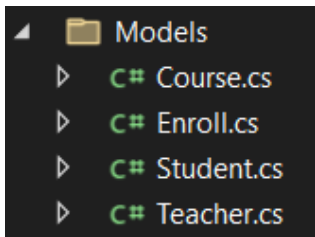
Namespaces



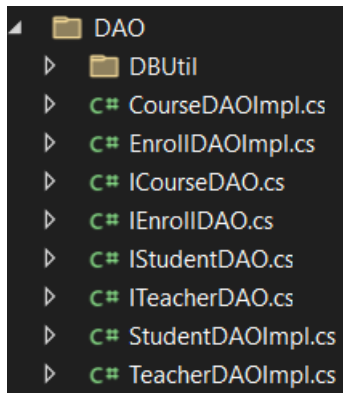
The application follows the design architecture SOA (Service-oriented-architecture) which obeys the design principle Separation of Concern. This is achieved by separating the code into small and self-contained layers (namespaces), which makes it easier to debug, maintain, extend and reuse for future additions. The data binding between the layers that have to communicate is loosely coupled and in our case is done with dependency injection via interfaces.

2.A NAMESPACES

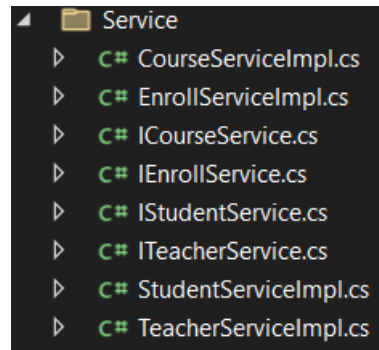
MODELS « DAO « SERVICE « VALIDATOR « DTO « PAGES



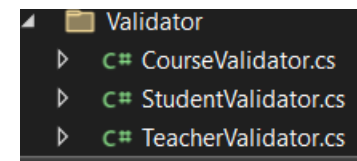
The classes in models are representations of data in the database



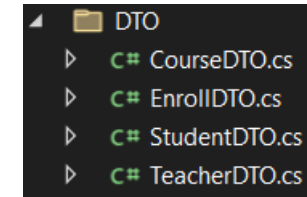
Data Access layer communicates with the database via DBUtil Class.



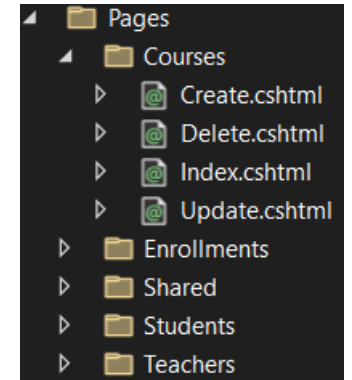
Service layer represents the public API with documentation communicating with DAO layer



Helper class that checks if the object transferred is valid



Data Transfer Object tranfers data from user-input to service layer



Holds for views(csHtml) and controllers that manages requests from user

2.B.MODELS

Classes
representing
each table of
database with
fields as fields
of the tables

```
namespace StudentsManagementApp.Models
{
    46 references
    public class Student
    {
        13 references
        public int Id { get; set; }
        10 references
        public string? Firstname { get; set; }
        10 references
        public string? Lastname { get; set; }
    }
}
```

```
1 namespace StudentsManagementApp.Models
2 {
3     46 references
4     public class Teacher
5     {
6         14 references
7         public int Id { get; set; }
8         11 references
9         public string? Firstname { get; set; }
10        11 references
11        public string? Lastname { get; set; }
12    }
13 }
```

```
namespace StudentsManagementApp.Models
{
    46 references
    public class Course
    {
        13 references
        public int Id { get; set; }
        10 references
        public string? Description { get; set; }
        7 references
        public int TeacherId { get; set; }
    }
}
```

```
1 namespace StudentsManagementApp.Models
2 {
3     23 references
4     public class Enroll
5     {
6         7 references
7         public int StudentId { get; set; }
8         7 references
9         public int CourseId { get; set; }
10    }
11 }
```


2.B.DAO

DAO layer contains interfaces and classes that implement them. For example ..

```
using StudentsManagementApp.Models;

namespace StudentsManagementApp.DAO
{
    7 references
    public interface IStudentDAO
    {
        2 references
        void Insert(Student? student);
        2 references
        void Update(Student? student);
        2 references
        Student? Delete(Student? student);
        2 references
        Student? GetStudent(int id);

        2 references
        List<Student> GetAll();
    }
}
```

Interface

Each CRUD method uses DB helper first to connect to database and then to command queries and read or send data

```
4 references
public class StudentDAOImpl : IStudentDAO
{
    2 references
    public Student? Delete(Student? student)
    {
        if (student == null) return null;

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

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

            string sql1 = "DELETE FROM STUDENT_COURSE WHERE STUDENT_ID = @id";
            string sql2 = "DELETE FROM STUDENTS WHERE ID = @id";

            using SqlCommand command1 = new(sql1, conn);
            using SqlCommand command2 = new(sql2, conn);

            command1.Parameters.AddWithValue("@id", student.Id);
            command2.Parameters.AddWithValue("@id", student.Id);

            command1.ExecuteNonQuery();
            int rowsAffected = command2.ExecuteNonQuery();
            return (rowsAffected > 0) ? student : null;
        }
        catch (Exception e)
        {
            Console.WriteLine(e.StackTrace);
            throw;
        }
    }
}
```

Delete method implementation

Service Interface

```
1 using StudentsManagementApp.DTO;
2 using StudentsManagementApp.Models;
3
4 namespace StudentsManagementApp.Service
5 {
6     5 references
7     public interface IStudentService
8     {
9         /// <summary>
10        /// A method that returns a list with all the students
11        /// </summary>
12        /// <returns> A list of Student objects</returns>
13        2 references
14        List<Student> GetAllStudents();
15        /// <summary>
16        /// A method that inserts a record (student) with values Id,Firstname,Lastname at
17        /// the students table
18        /// </summary>
19        /// <param name="dto"> DTOStudent object that is to be
20        /// converted to Student</param>
21        2 references
22        void InsertStudent(StudentDTO? dto);
23        /// <summary>
24        /// A method that updates a record (Student) with values Firstname,Lastname at
25        /// the Students table
26        /// </summary>
27        /// <param name="dto">
28        /// DTOStudent that is to be
29        /// converted to Student</param>
30        2 references
31        void UpdateStudent(StudentDTO? dto);
32
33        /// <summary>
34        /// A method that brings one student ,searched by Id, from the student table
35        /// </summary>
36        /// <param name="id">Id of the student</param>
37        /// <returns> a Student object</returns>
38        2 references
39        Student? GetStudent(int id);
40
41        /// <summary>
42        /// It deletes a student record from Students table
43        /// </summary>
44        /// <param name="dto">
45        /// DTOStudent Object that is to be
46        /// converted to Student</param>
47        /// <returns>Returns the deleted Student </returns>
48        2 references
49        Student? DeleteStudent(StudentDTO? dto);
50    }
51 }
```

2.B.SERVICE

Service layer also contains interfaces and classes that implement them but also holds for public API and documentation explaining utility .

Service Delete Method Implementation

Service layer calls methods from DAO layer with dependency injection

```
public Student? DeleteStudent(StudentDTO? dto)
{
    if (dto == null) return null;

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

```
public class StudentServiceImpl : IStudentService
{
    private readonly IStudentDAO dao;

    4 references
    public StudentServiceImpl(IStudentDAO dao)
    {
        this.dao = dao;
    }
}
```

2.B.DTO

DTO like model classes but serves communication between controllers and service layer

```
namespace StudentsManagementApp.Models
{
    46 references
    public class Student
    {
        13 references
        public int Id { get; set; }
        10 references
        public string? Firstname { get; set; }
        10 references
        public string? Lastname { get; set; }
    }
}
```

```
1 namespace StudentsManagementApp.Models
2 {
3     46 references
4     public class Teacher
5     {
6         14 references
7         public int Id { get; set; }
8         11 references
9         public string? Firstname { get; set; }
10        11 references
11        public string? Lastname { get; set; }
12    }
13 }
```

```
namespace StudentsManagementApp.Models
{
    46 references
    public class Course
    {
        13 references
        public int Id { get; set; }
        10 references
        public string? Description { get; set; }
        7 references
        public int TeacherId { get; set; }
    }
}
```

```
1 namespace StudentsManagementApp.Models
2 {
3     23 references
4     public class Enroll
5     {
6         7 references
7         public int StudentId { get; set; }
8         7 references
9         public int CourseId { get; set; }
10    }
11 }
```

2.B.VALIDATOR

Checks if DTOS are valid

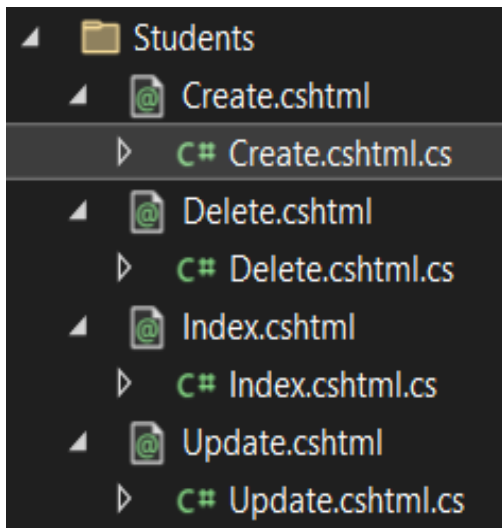
```
public class CourseValidator
{
    0 references
    private CourseValidator() { }

    2 references
    public static string Validate(CourseDTO? dto)
    {
        if (dto!.Description!.Length <= 6)
        {
            return "Course should not be less than 6 characters ";
        }
        else return "";
    }
}
```

```
1 using StudentsManagementApp.DTO;
2
3 namespace StudentsManagementApp.Validator
4 {
5     3 references
6     public class StudentValidator
7     {
8         //checks if a DTO is valid - UTILITY CLASS
9
10        0 references
11        private StudentValidator(){ }
12
13        2 references
14        public static string Validate(StudentDTO? dto)
15        {
16            if ((dto!.Firstname!.Length <=2) || (dto!.Lastname!.Length <= 4))
17            {
18                return "Firstname or Lastname should not be less than 2 and 4 characters respectively";
19            }
20            return "";
21        }
22    }
23 }
24
```

3. RAZOR PAGES

Razor pages contains the view (cshtml) file and the controller class that manages the requests of the user.



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

        internal List<Student> students = new();

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

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

Every time a request is made a new html page is sent back. Each razor pages represent a CRUD action .
For example IndexModel calls GetAllStudents() from Service layer

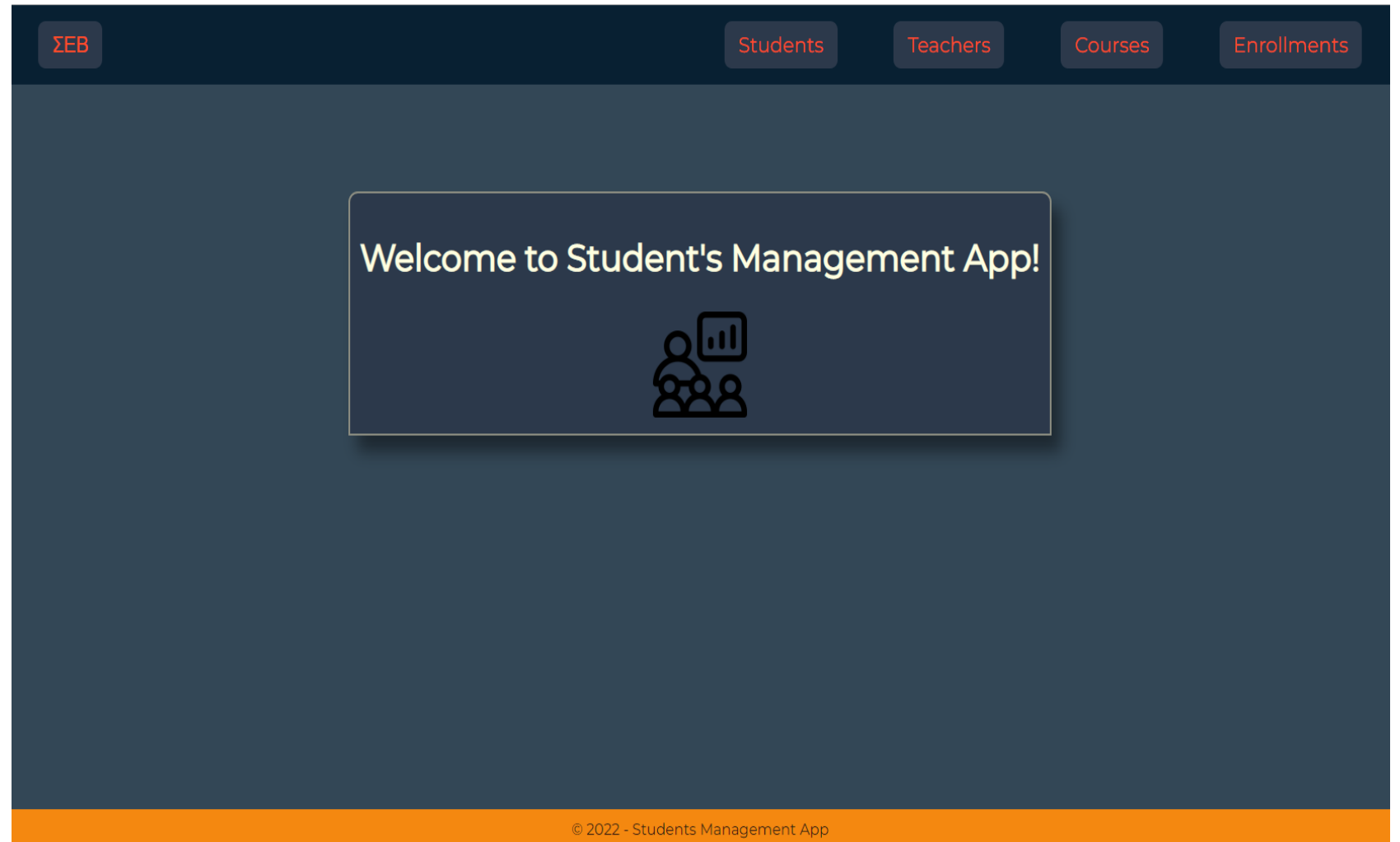
The HTML page that is sent to user

```
@page
@model StudentsManagementApp.Pages.Students.IndexModel
@{
}

<h2>List of Students</h2>
<div class="database">
    <div>
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Firstname</th>
                    <th>Lastname</th>
                </tr>
            </thead>
            <tbody>
                @if (Model.students is not null )
                {
                    @foreach (var student in Model.students)
                    {
                        <tr>
                            <td>@student.Id</td>
                            <td>@student.Firstname</td>
                            <td>@student.Lastname</td>
                            <td class="btns">
                                <a class="button" href="/Students/Update?id=@student.Id">Update</a>
                                <a class="button" href="/Students/Delete?id=@student.Id">Delete</a>
                            </td>
                        </tr>
                    }
                }
            </tbody>
        </table>
    </div>
</div>
<div class="button-container">
    <div class="button" id="new">
        <a href="/Students/Create">New Student</a>
    </div>
</div>
```

DEMONSTRATION

HOMEPAGE



DEMONSTRATION

Students page

The screenshot displays the 'Students' page of a management application. At the top, a navigation bar includes a 'ΣΕΒ' logo and buttons for 'Students', 'Teachers', 'Courses', and 'Enrollments'. The main heading is 'List of Students'. Below it, a table lists two students with 'Update' and 'Delete' buttons for each. A 'New Student' button is positioned at the bottom of the table area. The footer shows the URL 'https://localhost:7230/Students' and the copyright notice '© 2022 - Students Management App'.

ID	Firstname	Lastname		
77	Σπύρος	Νάτσιος	Update	Delete
78	Μπεγιάνα	Ελμάζαι	Update	Delete

New Student

https://localhost:7230/Students © 2022 - Students Management App

DEMONSTRATION

Create student

It throws error message:
name or lastname is under
2 or 4 characters

ΣΕΒ Students Teachers Courses Enrollments

New Student

Name
Λεφτέρης

Lastname
M

Submit Cancel

Firstname or Lastname should not be less than 2 and 4 characters respectively

© 2022 - Students Management App

DEMONSTRATION

Student created

ΣΕΒ

Students

Teachers

Courses

Enrollments

List of Students

ID	Firstname	Lastname		
77	Σπύρος	Νάτσιος	Update	Delete
78	Μπεγιάνα	Ελμάζαϊ	Update	Delete
80	Λεφτέρης	Μυλωνάκης	Update	Delete

New Student

© 2022 - Students Management App

DEMONSTRATION

Same goes with
Teacher

ΣΕΒ Students Teachers Courses Enrollments

List of Teachers

ID	Firstname	Lastname		
25	Θανάσης	Ανδρούτσος	Update	Delete
26	Κίμων	Ασκαρίδης	Update	Delete

New Teacher

© 2022 - Students Management App

DEMONSTRATION

Update Teacher

ΣΕΒStudentsTeachersCoursesEnrollments

Update Teacher

Firstname

Αλέξανδρος

Lastname

Ασκαριδης

SubmitCancel

© 2022 - Students Management App

DEMONSTRATION

ΣΕΒ

Students

Teachers

Courses

Enrollments

List of Teachers

ID	Firstname	Lastname		
25	Θανάσης	Ανδρούτσος	Update	Delete
26	Αλέξανδρος	Ασκαρίδης	Update	Delete

New Teacher

Teacher
Updated

DEMONSTRATION

ΣΕΒ

Students

Teachers

Courses

Enrollments

List of Courses

ID	Description	Teacher	
75	Αλγοριθμική	Αλέξανδρος	Ασκαρίδης
76	Machine Learning	Θανάσης	Ανδρούτσος

Update

Delete

Update

Delete

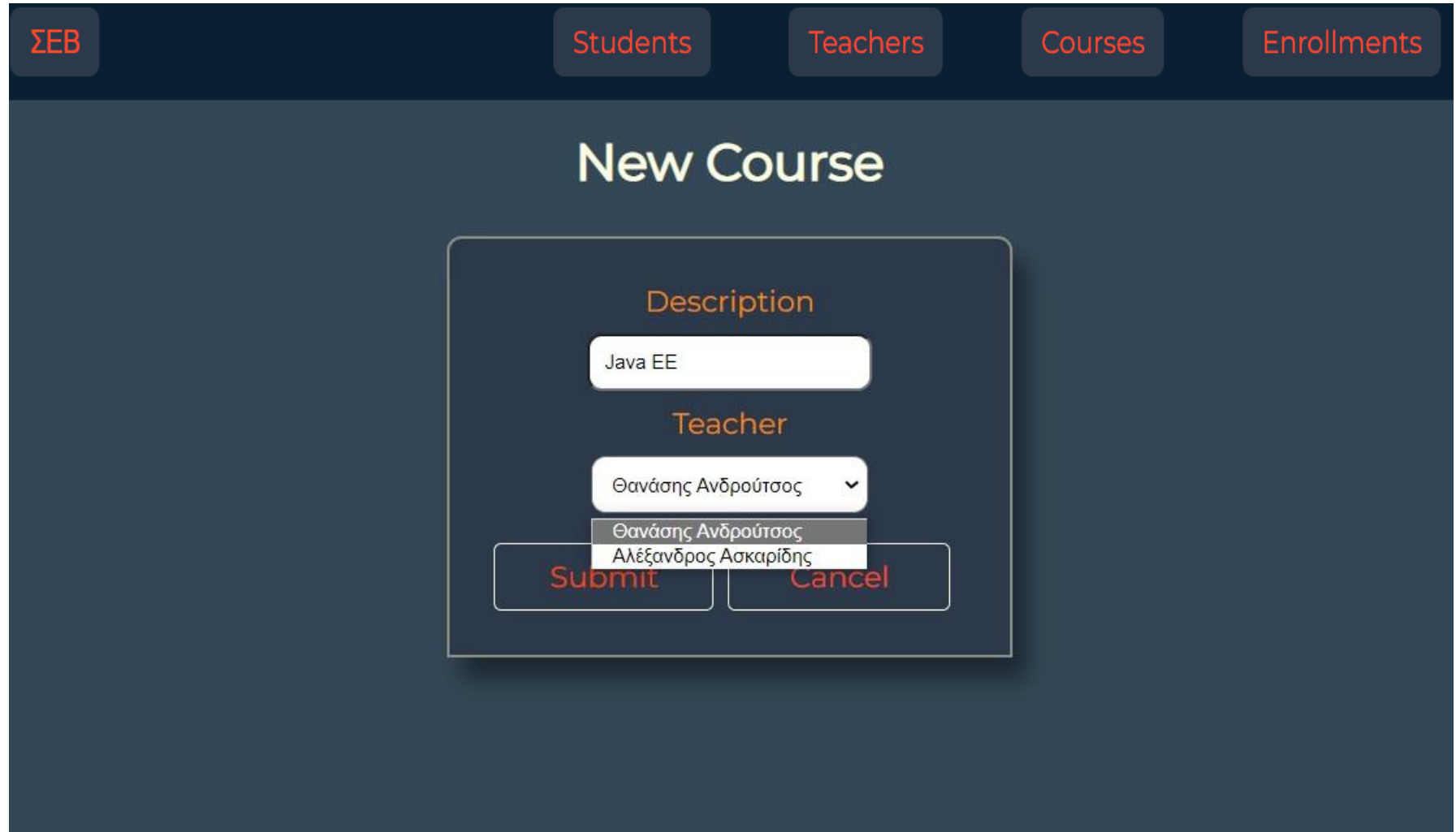
New Course

Courses

DEMONSTRATION

Create Course

It is required to
choose an already
inserted teacher



The screenshot shows a web application interface for creating a new course. At the top, there is a navigation bar with five tabs: ΣΕΒ, Students, Teachers, Courses, and Enrollments. The 'Courses' tab is active. Below the navigation bar, the title 'New Course' is displayed. The form itself is a dark blue box with rounded corners. It contains two input fields: 'Description' with the text 'Java EE' and 'Teacher' with a dropdown menu. The dropdown menu is open, showing two options: 'Θανάσης Ανδρούτσος' (highlighted) and 'Αλέξανδρος Ασκαρίδης'. At the bottom of the form are two buttons: 'Submit' and 'Cancel'.

ΣΕΒ Students Teachers Courses Enrollments

New Course

Description

Java EE

Teacher

Θανάσης Ανδρούτσος ▼

Θανάσης Ανδρούτσος

Αλέξανδρος Ασκαρίδης

Submit Cancel

DEMONSTRATION

Course Created

It is required to choose an already inserted teacher

ΣΕΒStudentsTeachersCoursesEnrollments

List of Courses

ID	Description	Teacher			
75	Αλγοριθμική	Αλέξανδρος	Ασκαρίδης	Update	Delete
76	Machine Learning	Θανάσης	Ανδρούτσος	Update	Delete
77	Java EE	Θανάσης	Ανδρούτσος	Update	Delete

New Course

DEMONSTRATION

Update Course

It throws error message: course should not be under 6 characters

ΣΕΒStudentsTeachersCoursesEnrollments

Update Course

Description

Java

Teacher

Κώστας Κωστόπουλος

Submit

Cancel

Course should not be less than 6 characters

DEMONSTRATION

ΣΕΒ

Students

Teachers

Courses

Enrollments

List of Courses

ID	Description	Teacher			
75	Αλγοριθμική	Αλέξανδρος	Ασκαρίδης	Update	Delete
76	Machine Learning	Θανάσης	Ανδρούτσος	Update	Delete
77	Java Enterprise Edition	Κώστας	Κωστόπουλος	Update	Delete

New Course

Course
Updated

DEMONSTRATION

ΣΕΒ

Students

Teachers

Courses

Enrollments

Enrollments

ID	Firstname	Lastname	Course Id	Course
78	Μπεγιάνα	Ελμάζαϊ	77	Java Enterprise Edition

Delete

New Enrollment

Enrollments

DEMONSTRATION

Create Enroll

It is required to
choose already
inserted Students and
Courses

The screenshot shows a web application interface with a dark blue header and a sidebar. The header contains navigation buttons: ΣΕΒ, Students, Teachers, Courses, and Enrollments. The main content area is titled "New Enroll" and features a modal form for creating a new enrollment. The form has a title "Student" and a dropdown menu with the following options: Σπύρος Νάτσιος, Μπεγιάννα Ελμάζαι, Λεφτέρης Μυλωνάκης, and Αλγοριθμική. Below the dropdown are two buttons: Submit and Cancel.

ΣΕΒ Students Teachers Courses Enrollments

New Enroll

Student

Σπύρος Νάτσιος ▼

Μπεγιάννα Ελμάζαι

Λεφτέρης Μυλωνάκης

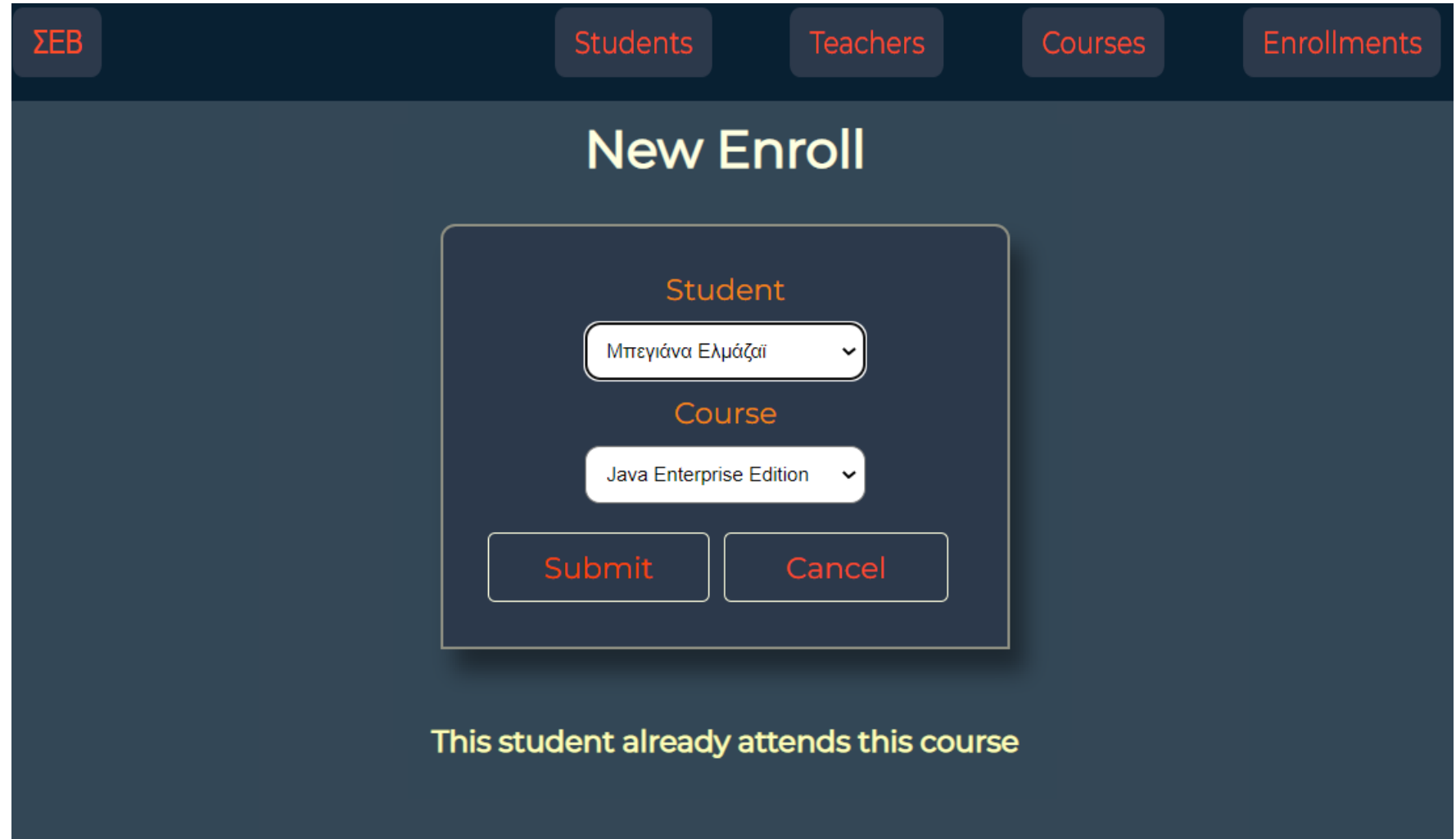
Αλγοριθμική ▼

Submit Cancel

DEMONSTRATION

Create Enroll

It throws error if you make the same enroll



The screenshot shows a web application interface for creating a new enrollment. At the top, there is a navigation bar with five tabs: ΣΕΒ, Students, Teachers, Courses, and Enrollments. The 'Enrollments' tab is currently selected. Below the navigation bar, the main heading is 'New Enroll'. In the center, there is a form with two dropdown menus. The first dropdown is labeled 'Student' and has 'Μπεγιάνα Ελμάζαϊ' selected. The second dropdown is labeled 'Course' and has 'Java Enterprise Edition' selected. Below these dropdowns are two buttons: 'Submit' and 'Cancel'. At the bottom of the form, a message in yellow text states: 'This student already attends this course'.

ΣΕΒ Students Teachers Courses Enrollments

New Enroll

Student

Μπεγιάνα Ελμάζαϊ ▼

Course

Java Enterprise Edition ▼

Submit Cancel

This student already attends this course

DEMONSTRATION

ΣΕΒ

Students

Teachers

Courses

Enrollments

Enrollments

ID	Firstname	Lastname	Course Id	Course	
77	Σπύρος	Νάτσιος	77	Java Enterprise Edition	Delete
78	Μπεγιάνα	Ελμάζαϊ	77	Java Enterprise Edition	Delete

New Enrollment

Enrollment
Created

DEMONSTRATION

When you delete a course also enrollments in this course are deleted

List of Courses

ID	Description	Teacher		
75	Αλγοριθμική	Αλέξανδρος Ασκαρίδης	Update	Delete
76	Machine Learning	Θανάσης Ανδρούτσος	Update	Delete
77	Java Enterprise Edition	Κώστας Κωστούπουλος	Update	Delete

New Course

Enrollments

ID	Firstname	Lastname	Course Id	Course	
77	Σπύρος	Νάτσιος	77	Java Enterprise Edition	Delete
78	Μπεγιάνα	Ελμάζαϊ	77	Java Enterprise Edition	Delete

New Enrollment

List of Courses

ID	Description	Teacher		
75	Αλγοριθμική	Αλέξανδρος Ασκαρίδης	Update	Delete
76	Machine Learning	Θανάσης Ανδρούτσος	Update	Delete

New Course

Enrollments

ID	Firstname	Lastname	Course Id	Course
----	-----------	----------	-----------	--------

New Enrollment

DEMONSTRATION

When you delete a student also enrollments of this student are deleted

List of Students

ID	Firstname	Lastname		
77	Σπύρος	Νάτσιος	Update	Delete
78	Μπεγιάνα	Ελμάζαϊ	Update	Delete
80	Λεφτέρης	Μυλωνάκης	Update	Delete

Enrollments

ID	Firstname	Lastname	Course Id	Course	
80	Λεφτέρης	Μυλωνάκης	75	Αλγοριθμική	Delete

New Enrollment

List of Students

ID	Firstname	Lastname		
77	Σπύρος	Νάτσιος	Update	Delete
78	Μπεγιάνα	Ελμάζαϊ	Update	Delete

New Student

Enrollments

ID	Firstname	Lastname	Course Id	Course
----	-----------	----------	-----------	--------

New Enrollment

DEMONSTRATION

When you delete a teacher also courses and enrollments in this course are deleted

List of Teachers

ID	Firstname	Lastname		
25	Θανάσης	Ανδρούτσος	Update	Delete
26	Αλέξανδρος	Ασκαρίδης	Update	Delete
29	Κώστας	Κωστόπουλος	Update	Delete

New Teacher

List of Courses

ID	Description	Teacher		
75	Αλγοριθμική	Αλέξανδρος Ασκαρίδης	Update	Delete
76	Machine Learning	Θανάσης Ανδρούτσος	Update	Delete
78	Μαθηματικά	Κώστας Κωστόπουλος	Update	Delete

New Course

Enrollments

ID	Firstname	Lastname	Course Id	Course	
77	Σπύρος	Νάτσιος	78	Μαθηματικά	Delete

New Enrollment

List of Teachers

ID	Firstname	Lastname		
25	Θανάσης	Ανδρούτσος	Update	Delete
26	Αλέξανδρος	Ασκαρίδης	Update	Delete

New Teacher

List of Courses

ID	Description	Teacher		
75	Αλγοριθμική	Αλέξανδρος Ασκαρίδης	Update	Delete
76	Machine Learning	Θανάσης Ανδρούτσος	Update	Delete

New Course

Enrollments

ID	Firstname	Lastname	Course Id	Course
----	-----------	----------	-----------	--------

New Enrollment

SOURCES

- Notes from **Thanasis Androutsos** , also the teacher leading this course
- https://en.wikipedia.org/wiki/Service-oriented_architecture