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

Github : https://github.com/Begiana/CSharp-Student-Management-App

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

1. DATABASE

Consist's of four tables

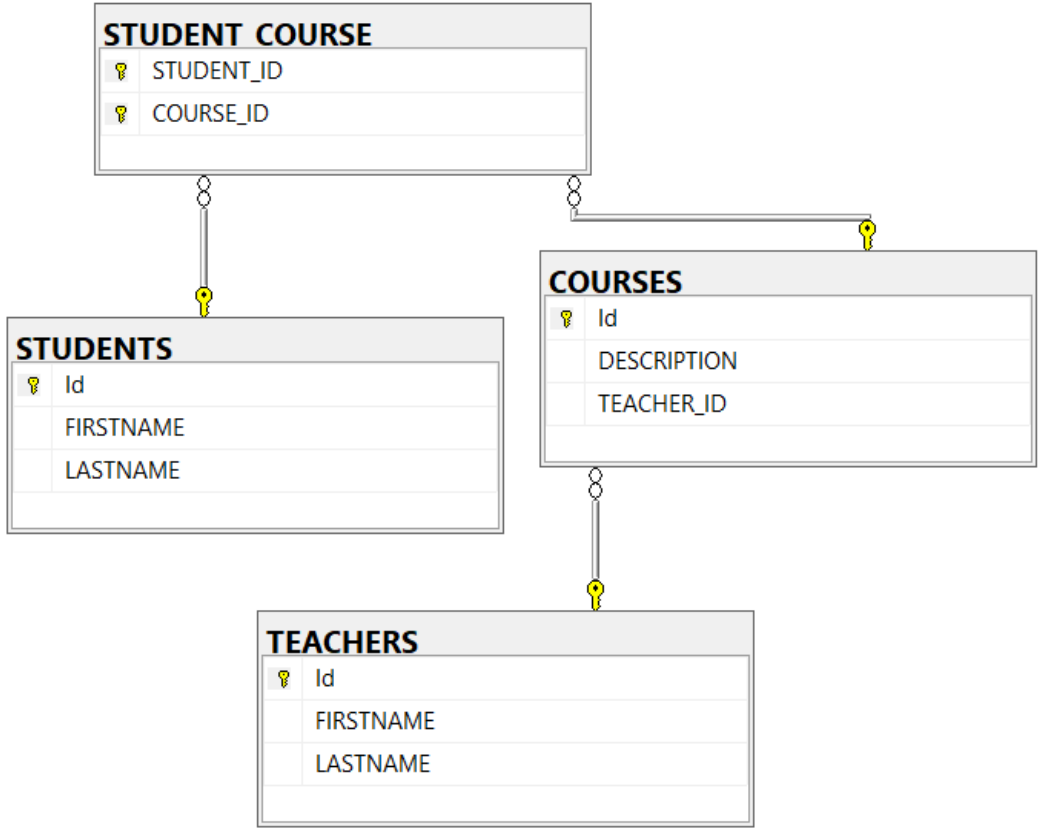
- **STUDENTS (Id, FIRSTNAME, LASTNAME) [Id Primary Key]**
- **TEACHERS (Id, FIRSTNAME, LASTNAME) [Id Primary Key]**
- **COURSES (Id, DESCRIPTION, TEACHER_ID) [Id Primary Key]**

  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]



1. 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...

- DAO
  - DBUtil
    - DBHelper.cs

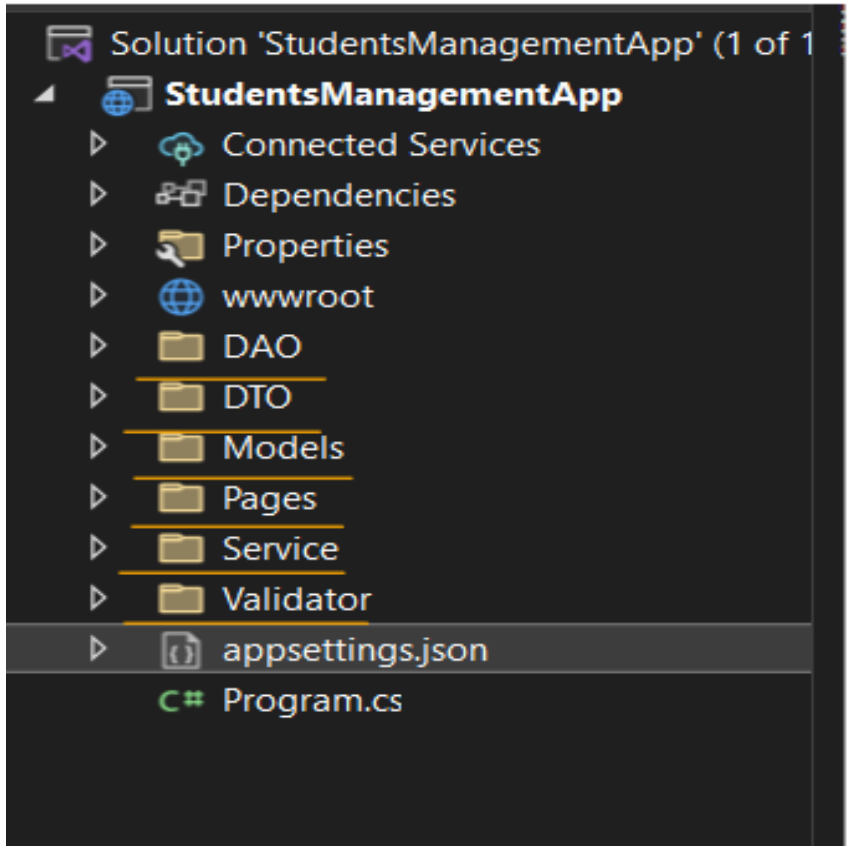
Connection with DB is done by Installing Nuget Package SQLClient and provided the classes and interfaces that create connection with database, command queries, read or send data.

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

```
using System.Data.SqlClient;

namespace StudentsManagementApp.DAO.DBUtil
{
    23 references
    public class DBHelper
    {
        private static SqlConnection? conn;

        //No instances of this class available .U T I L I T Y   C L A S S
        0 references
        private DBHelper() { }

        22 references
        public static SqlConnection? GetConnection()
        {
            try
            {
                ConfigurationManager configurationManager = new();
                configurationManager.AddJsonFile("appsettings.json");
                string url = configurationManager.GetConnectionString("DefaultConnection");
                //string url = "Data Source=localhost\\sqlexpress;Initial Catalog=SevDB;Integrated Security=True";
                conn = new SqlConnection(url);
                return conn;
            }
            catch ( Exception e)
            {
                Console.WriteLine(e.StackTrace);
                return null;
            }
        }

        0 references
        public static void CloseConnection()
        {
            if (conn != null)
            {
                conn.Close();
            }
        }
    }
}
```

2. CODE DESIGN

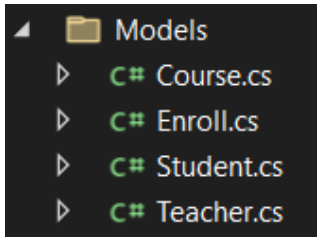
Namespaces



The application follows the design architecture SOA (Service-oriented-architecture) which obeys the design principle Seperation 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.

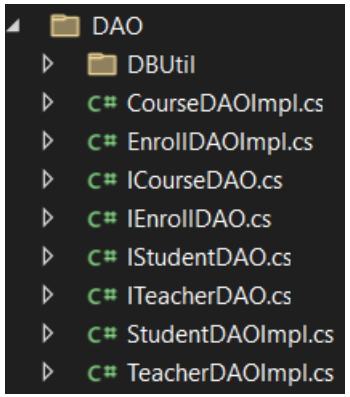
2.A NAMESPACES

MODELS



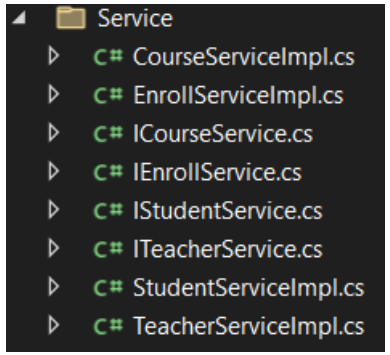
The classes in models are representations of data in the database

DAO



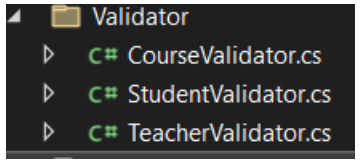
Data Access layer communicates with the database via DBUtil Class.

SERVICE



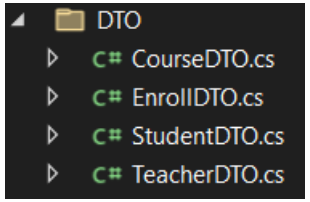
Service layer represents the public API with documentation communicating with DAO layer

VALIDATOR



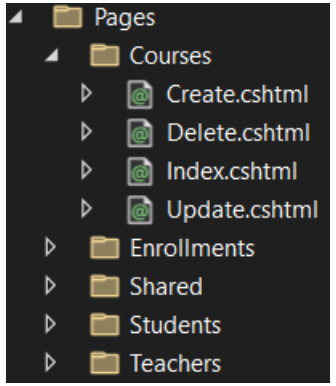
Helper class that checks if the object transferred is valid

DTO



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

PAGES



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; }
    }
}
```

```
namespace StudentsManagementApp.Models
{
    46 references
    public class Teacher
    {
        14 references
        public int Id { get; set; }
        11 references
        public string? Firstname { get; set; }
        11 references
        public string? Lastname { get; set; }
    }
}
```

```
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; }

    }
}
```

```
namespace StudentsManagementApp.Models
{
    23 references
    public class Enroll
    {
        7 references
        public int StudentId { get; set; }
        7 references
        public int CourseId { get; set; }

    }
}
```

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

2.B.SERVICE

Service Interface

```
using StudentsManagementApp.DTO;
using StudentsManagementApp.Models;

namespace StudentsManagementApp.Service
{
    5 references
    public interface IStudentService
    {
        /// <summary>
        /// A method that returns a list with all the students
        /// </summary>
        /// <returns> A list of Student objects</returns>
        2 references
        List<Student> GetAllStudents();
        /// <summary>
        /// A method that inserts a record (student) with values Id,Firstname,Lastname at
        /// the students table
        /// </summary>
        /// <param name="dto"> DTOStudent object that is to be
        /// converted to Student</param>
        2 references
        void InsertStudent(StudentDTO? dto);
        /// <summary>
        /// A method that updates a record (Student) with values Firstname,Lastname at
        /// the Students table
        /// </summary>
        /// <param name="dto">
        /// DTOStudent that is to be
        /// converted to Student</param>
        2 references
        void UpdateStudent(StudentDTO? dto);

        /// <summary>
        /// A method that brings one student ,searched by Id, from the student table
        /// </summary>
        /// <param name="id">Id of the student</param>
        /// <returns> a Student object</returns>
        2 references
        Student? GetStudent(int id);

        /// <summary>
        /// It deletes a student record  from Students table
        /// </summary>
        /// <param name="dto">
        /// DTOStudent Object that is to be
        /// converted to Student</param>
        /// <returns>Returns the deleted Student </returns>
        2 references
        Student? DeleteStudent(StudentDTO? dto);
    }
}
```

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

Service Delete Method Implementation

```
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;
    }
}
```

Service layer calls methods from DAO layer with dependency injection

```
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; }
    }
}
```

```
namespace StudentsManagementApp.Models
{
    46 references
    public class Teacher
    {
        14 references
        public int Id { get; set; }
        11 references
        public string? Firstname { get; set; }
        11 references
        public string? Lastname { get; set; }
    }
}
```

```
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; }

    }
}
```

```
namespace StudentsManagementApp.Models
{
    23 references
    public class Enroll
    {
        7 references
        public int StudentId { get; set; }
        7 references
        public int CourseId { get; set; }

    }
}
```

2.B.VALIDATOR

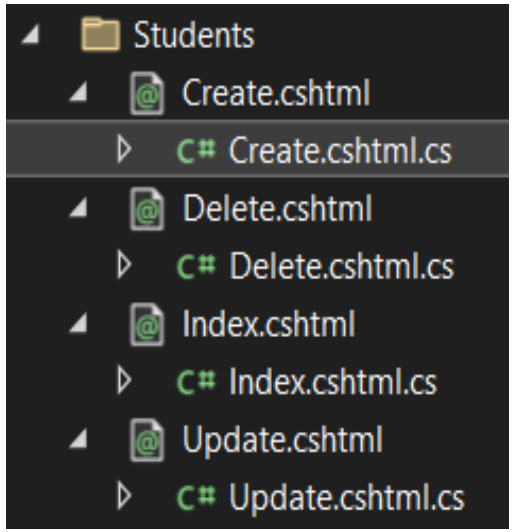
Checks if DTOS are valid

```
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 "";
    }
}
```

```
using StudentsManagementApp.DTO;

namespace StudentsManagementApp.Validator
{
    3 references
    public class StudentValidator
    {
        //checks if a DTO is valid  -  U T I L I T Y   C L A S S

        0 references
        private StudentValidator(){ }


        2 references
        public static string Validate(StudentDTO? dto)
        {
            if ((dto!.Firstname!.Length <=2) || (dto!.Lastname!.Length <= 4))
            {
                return "Firstname or Lastname should not be less than 2 and 4 characters respectively";
            }
            return "";
        }


    }
}
```

3.RAZOR PAGES

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



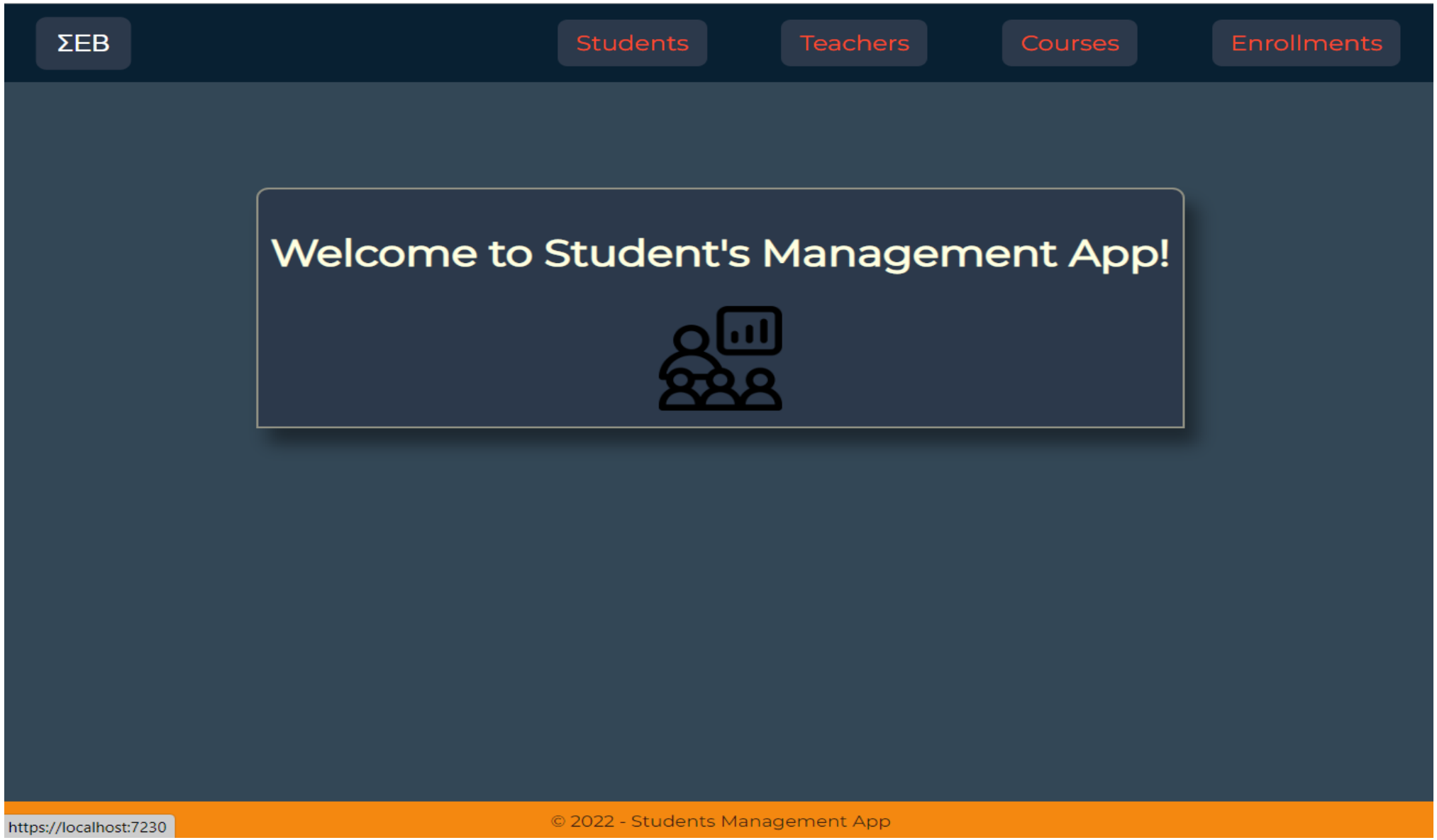

```
namespace StudentsManagementApp.Pages.Students
{
    6 references
    public class IndexModel : PageModel
    {
        private readonly IStudentDAO studentDAO = new StudentDAOImpl();
        private readonly IStudentService? service;

        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

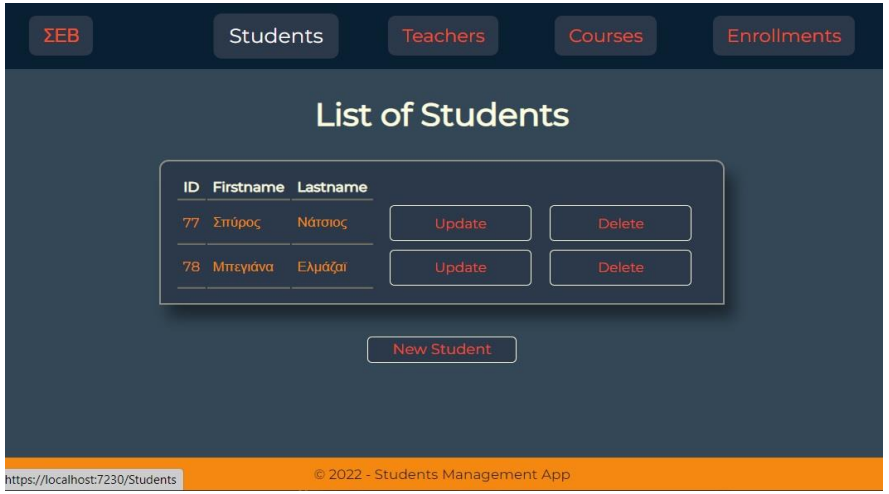
The HTML page that is sent to user

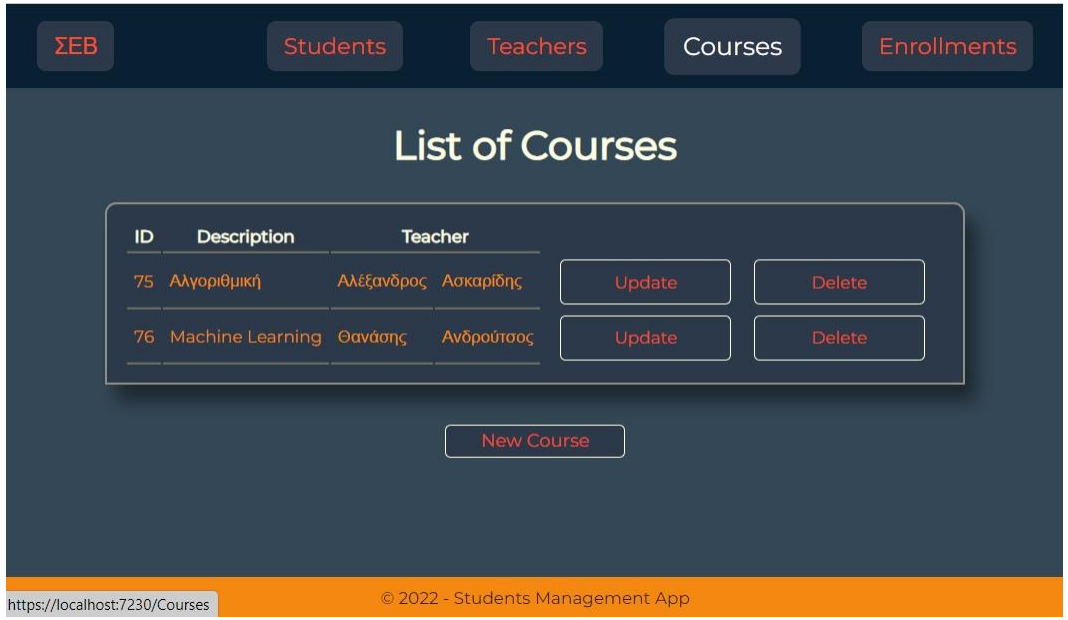
```
@page
@model StudentsManagementApp.Pages.Students.IndexModel
@{
}

<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>
```

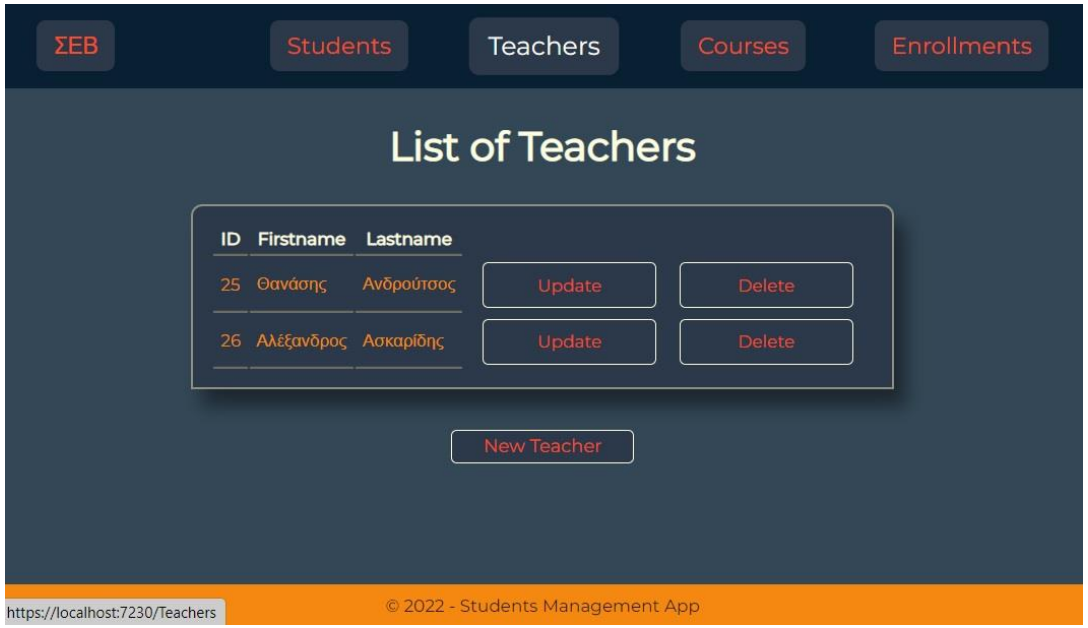
HOMEPAGE

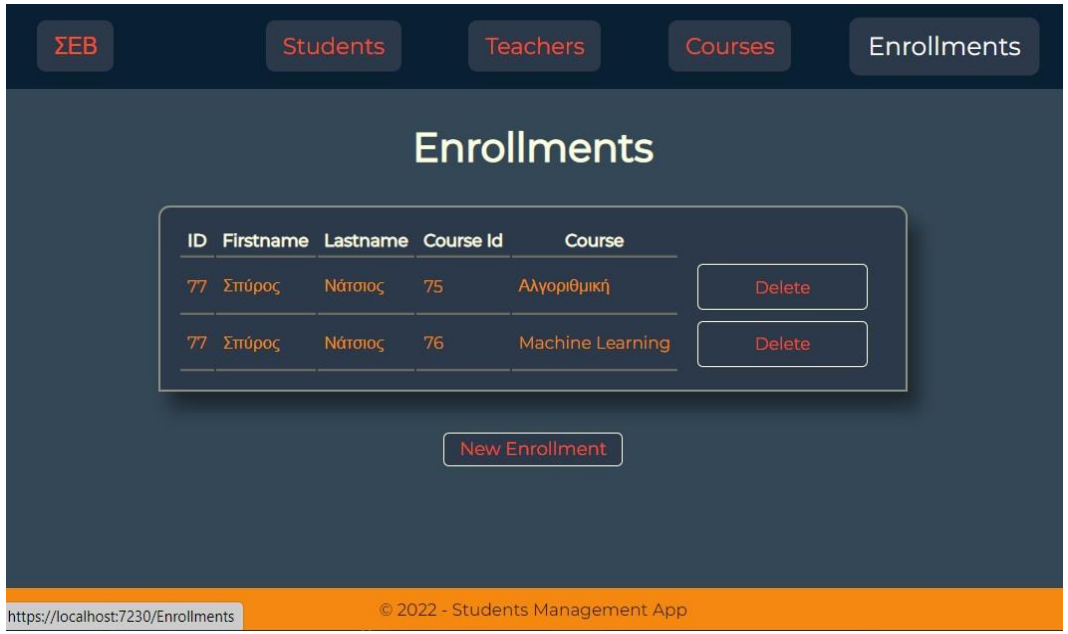

ΣEB
Students
Teachers
Courses
Enrollments
Welcome to Student's Management App!
https://localhost:7230
© 2022 - Students Management App

INDEX Pages

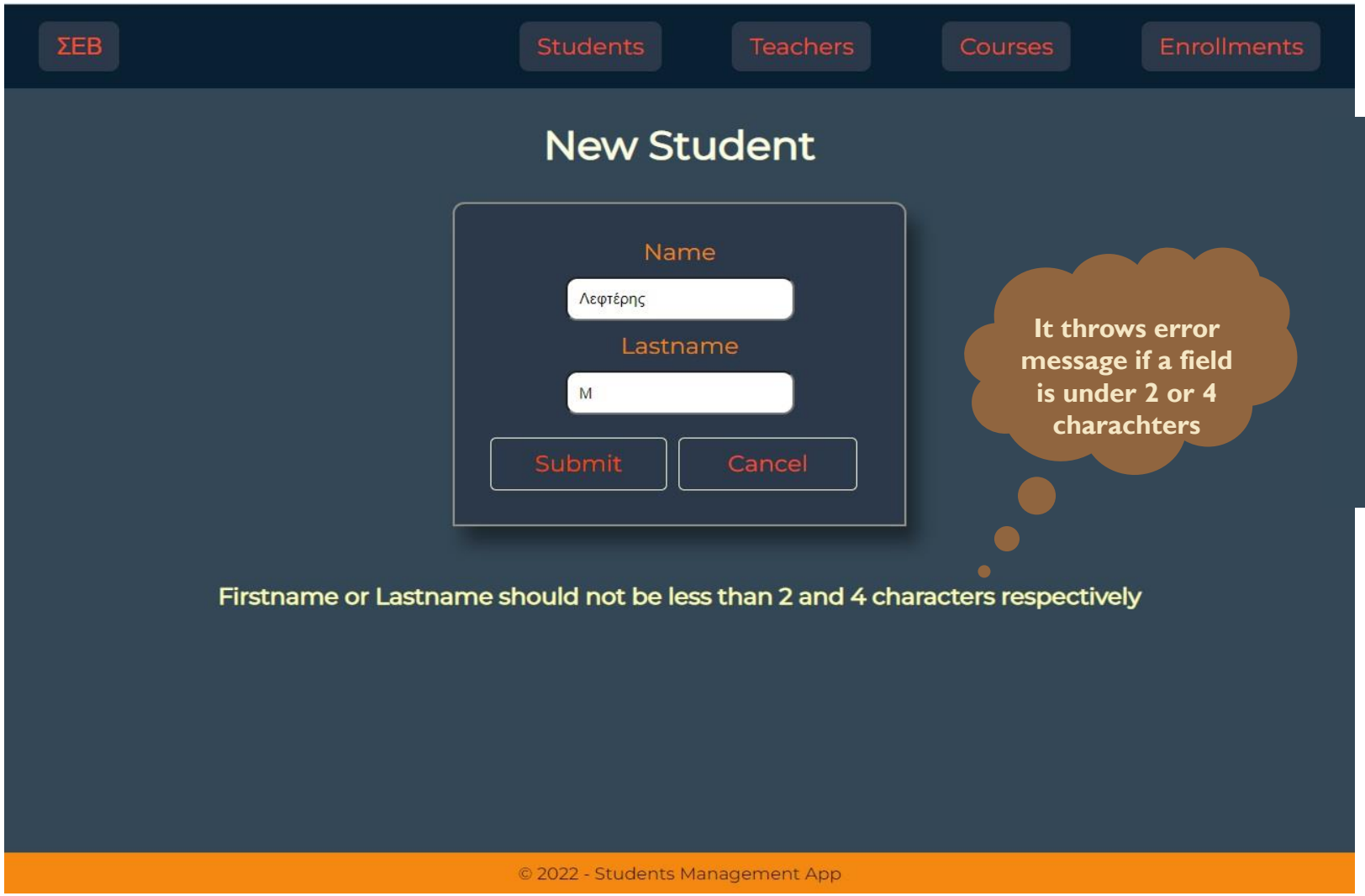

ΣΕΒ
Students
Teachers
Courses
Enrollments
List of Students
ID Firstname Lastname
77 Σπύρος Νάτσιος
Update
Delete
78 Μπεγιάνα Ελμάζαι
Update
Delete
New Student
https://localhost:7230/Students
© 2022 - Students Management App


ΣΕΒ
Students
Teachers
Courses
Enrollments
List of Courses
ID Description Teacher
75 Αλγοριθμική Αλέξανδρος Ασκαρίδης
Update
Delete
76 Machine Learning Θανάσης Ανδρούτσος
Update
Delete
New Course
https://localhost:7230/Courses
© 2022 - Students Management App

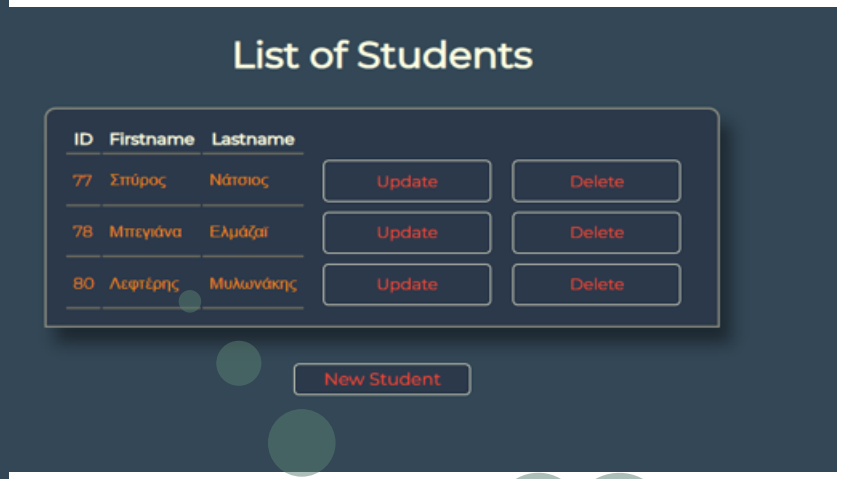

ΣΕΒ
Students
Teachers
Courses
Enrollments
List of Teachers
ID Firstname Lastname
25 Θανάσης Ανδρούτσος
Update
Delete
26 Αλέξανδρος Ασκαρίδης
Update
Delete
New Teacher
https://localhost:7230/Teachers
© 2022 - Students Management App


ΣΕΒ
Students
Teachers
Courses
Enrollments
Enrollments
ID Firstname Lastname Course Id Course
77 Σπύρος Νάτσιος 75 Αλγοριθμική
Delete
77 Σπύρος Νάτσιος 76 Machine Learning
Delete
New Enrollment
https://localhost:7230/Enrollments
© 2022 - Students Management App

DEMONSTRATION : STUDENT CREATION

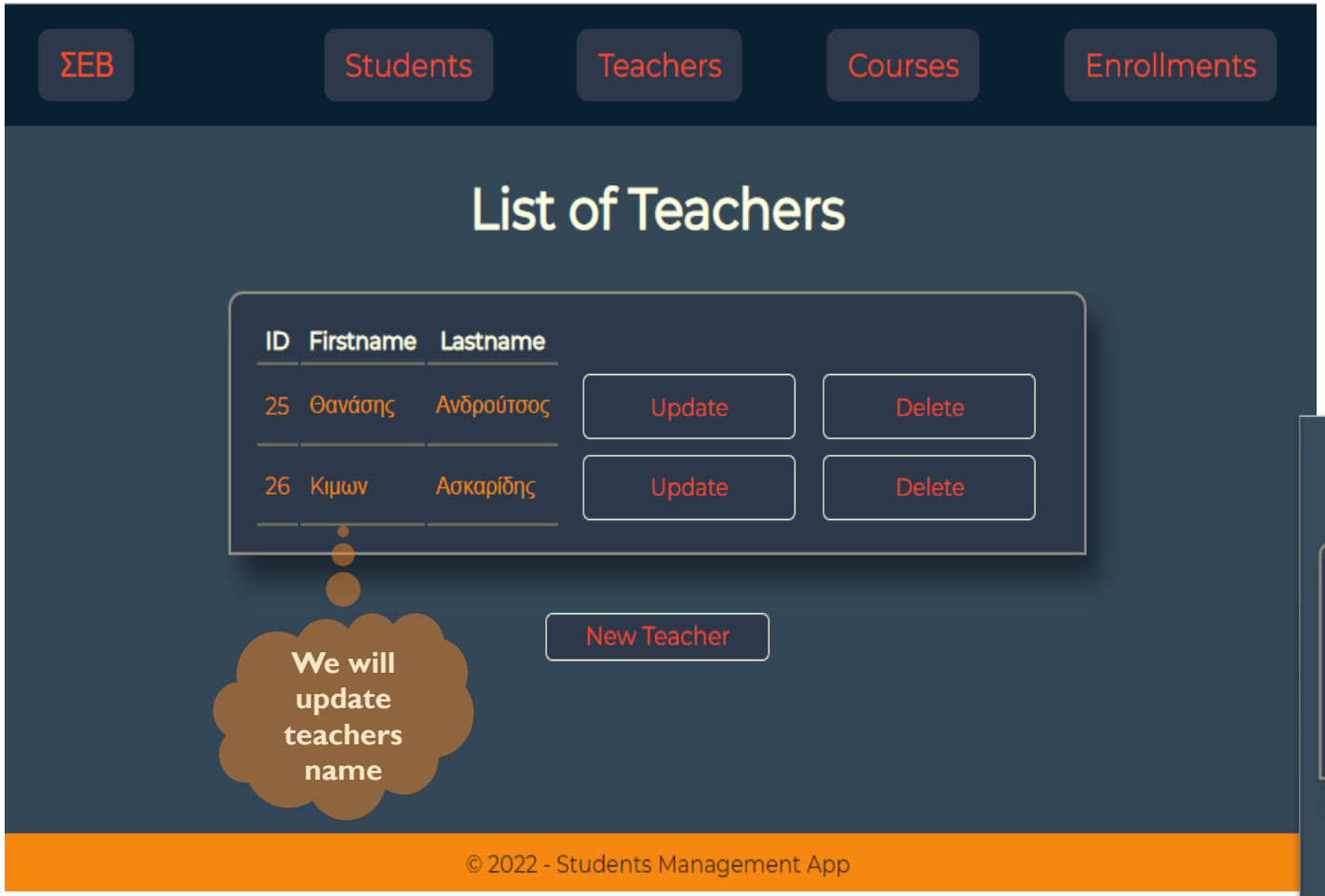


It throws error message if a field is under 2 or 4 charachters



Student created

DEMONSTRATION : TEACHER UPDATE



DEMONSTRATION

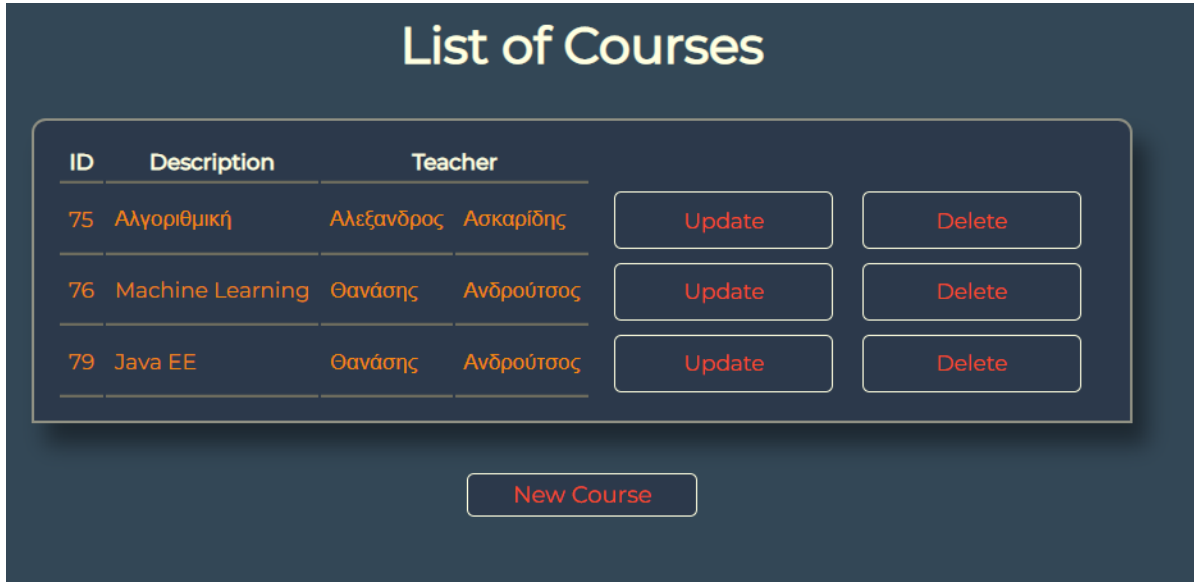
CREATE COURSE

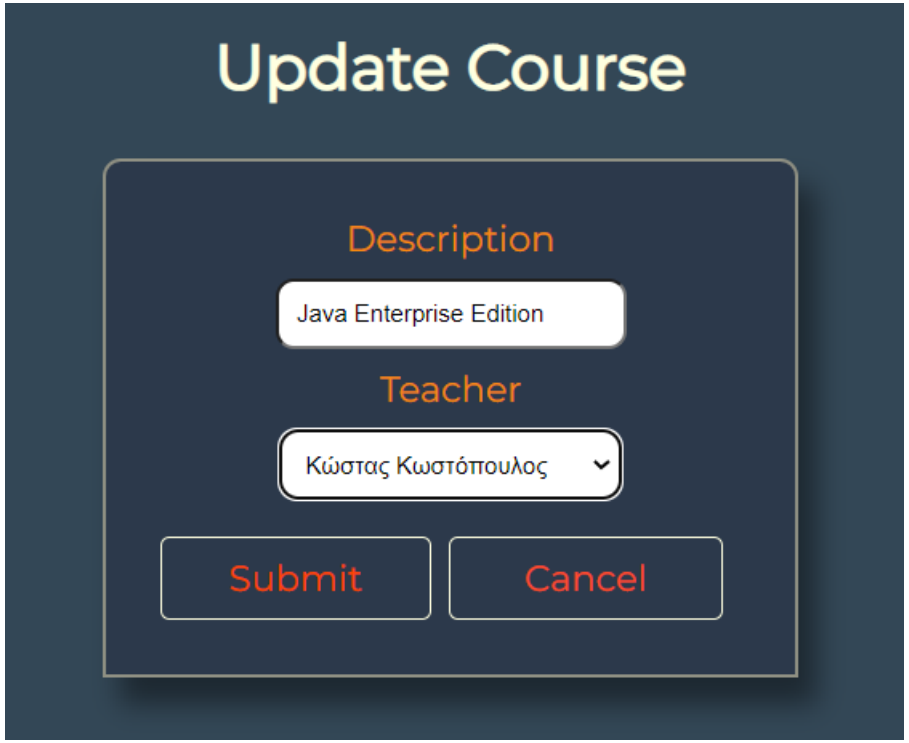
It is required to choose an already inserted teacher

Course created

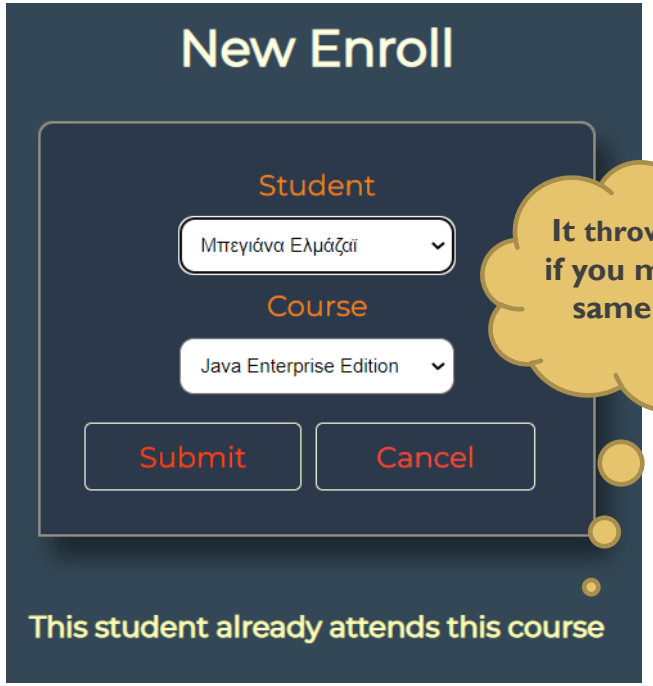
DEMONSTRATION

UPDATE COURSE

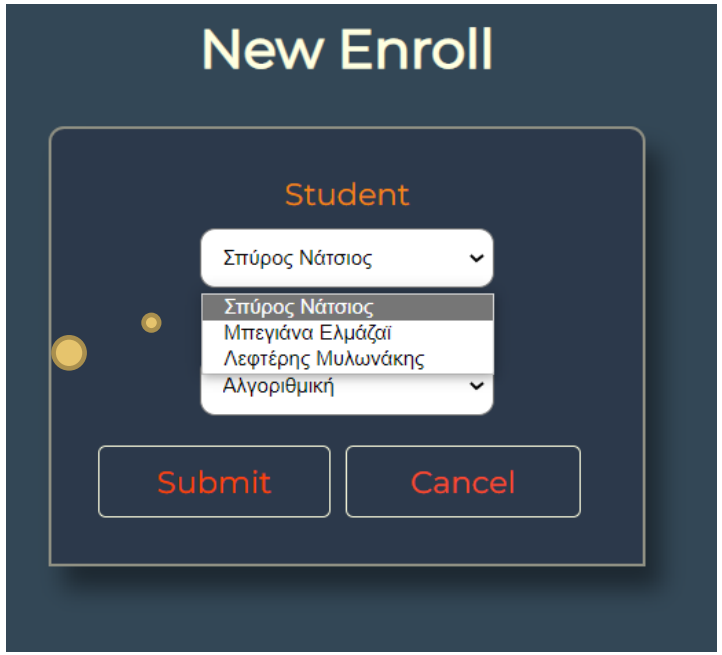




DEMONSTRATION CREATE ENROLL



It throws error if you make the same enroll



It is required to choose already inserted Students and Courses

Enrollment Created

DEMONSTRATION DELETE COURSES

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	Κωστόπουλος	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 DELETE STUDENT

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 DELETE TEACHER

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