

# Documentație pentru Student Registration

## I. Descriere

Studentreg este o aplicație de management a studenților care permite utilizatorilor să înregistreze, să vizualizeze, să editeze și să șteargă informații despre studenți. Aplicația folosește un serviciu web pentru a comunica cu o bază de date și oferă o interfață grafică utilizatorului (GUI) pentru a interacționa cu datele.

## II. Tehnologii utilizate

- C#
- Windows Forms
- ASP.NET Web API
- Microsoft SQL Server

## III. Funcționalități

### 1. Înregistrare de studenți

- Utilizatorul poate adăuga un nou student completând informațiile în câmpurile de nume, curs și taxă, apoi apăsând butonul "Save".
- Informațiile introduse sunt trimise către serviciul web pentru a fi înregistrate în baza de date.

### 2. Vizualizare a studenților

- Lista cu studenți este afișată într-un tabel cu coloane pentru ID, nume, curs și taxă.
- Datele sunt încărcate din baza de date folosind serviciul web și afișate în interfața utilizatorului.

### 3. Editare a studenților

- Utilizatorul poate selecta un student din tabel și apăsa butonul "Edit" pentru a modifica informațiile despre acel student.
- Informațiile selectate sunt încărcate în câmpurile de editare, iar utilizatorul poate face modificările dorite și apăsa butonul "Save" pentru a actualiza datele.

#### 4. Ștergere a studenților

- Utilizatorul poate selecta un student din tabel și apăsa butonul "Delete" pentru a șterge acel student din baza de date.
- După confirmare, înregistrarea studentului este ștearsă din baza de date.

### IV. **Conectarea la serviciul web**

- Aplicația comunică cu un serviciu web folosind HTTP pentru a efectua operațiile CRUD (Create, Read, Update, Delete) asupra datelor de student.
- Adresa URL a serviciului web este `http://localhost:5262`.

### V. **Interfața utilizatorului**

- Interfața utilizatorului este construită folosind Windows Forms și oferă o experiență simplă și intuitivă pentru utilizatori.
- Utilizatorii pot naviga ușor între diferitele funcționalități ale aplicației folosind butoanele și interacțiunile din GUI.

### IV. **Codul sursă**

#### 1. Studentreg (Windows Forms) Form1.cs:

```
using System; using
System.Drawing.Text; using
System.Net.Http; using
System.Net.Http.Json; using
System.Text; using
System.Threading.Tasks;
```

```

using System.Windows.Forms;
using Newtonsoft.Json;

namespace Studentreg
{
    public partial class Form1 : Form
    {
        private readonly HttpClient _httpClient;
        private const string BaseUrl = "http://localhost:5262"; // Adresa URL a
serviciului web

        public bool Mode { get; private set; }

        public Form1()
        {
            InitializeComponent();
            Load();
            _httpClient = new HttpClient();
        }
        public new async Task Load()
        {
try
            {
                HttpResponseMessage response = await
_httpClient.GetAsync($"{BaseUrl}/api/students");
response.EnsureSuccessStatusCode(); var
students = await
response.Content.ReadFromJsonAsync<Student[]>();

                dataGridView1.Rows.Clear();
foreach (var student in students)
                {
                    dataGridView1.Rows.Add(student.Id, student.Name, student.Course,
student.Fee);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        public async Task<Student> GetStudentById(int id)
        {
            var response = await
_httpClient.GetAsync($"{BaseUrl}/api/students/{id}");
response.EnsureSuccessStatusCode();
            return await
response.Content.ReadFromJsonAsync<Student>();
        }
        private async void button1_Click(object sender, EventArgs e)
        {
            string name = txtName.Text;
            string course = txtCourse.Text;
            string fee = txtFee.Text;

            if (Mode == true)
            {
                var student = new Student { Name = name, Course = course, Fee =

```

```

(double)decimal.Parse(fee) });
        await SendHttpRequest(HttpMethod.Post, "/api/students", student);
        MessageBox.Show("Record Added");
    }
else
    {
        string id = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        var student = new Student { Id = int.Parse(id), Name = name, Course
= course, Fee = (double)decimal.Parse(fee) };
        await SendHttpRequest(HttpMethod.Put, $"/api/students/{id}",
student);
        MessageBox.Show("Record Updated");
    }

    await Load();
}

private async void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.ColumnIndex == dataGridView1.Columns["Edit"].Index && e.RowIndex
>= 0)
    {
        Mode = false;
        string id = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        var student = await GetStudentById(int.Parse(id));
        txtName.Text = student.Name;                txtCourse.Text = student.Course;
        txtFee.Text = student.Fee.ToString();        button1.Text = "Edit";
    }
    else if (e.ColumnIndex == dataGridView1.Columns["Delete"].Index &&
e.RowIndex >= 0)
    {
        string id = dataGridView1.CurrentRow.Cells[0].Value.ToString();
        await SendHttpRequest(HttpMethod.Delete, $"/api/students/{id}");
        MessageBox.Show("Record Deleted");
        await Load();
    }
}

private async Task SendHttpRequest(HttpMethod method, string
requestUri, object data = null)
{
    var json = data != null ? JsonConvert.SerializeObject(data) : null;
    var content = new StringContent(json, Encoding.UTF8,
"application/json");
    var response = await _httpClient.SendAsync(new HttpRequestMessage
{
        Method = method,
        RequestUri = new Uri(BaseUrl + requestUri),
        Content = content
    });

    response.EnsureSuccessStatusCode();
}
}
}

```

## 2. StudentManagementService.Web (ASP.NET Web API)

```
using System;
using System.Collections.Generic;
using System.Linq; using
System.Threading.Tasks; using
Microsoft.AspNetCore.Http; using
Microsoft.AspNetCore.Mvc; using
Microsoft.EntityFrameworkCore; using
StudentManagementService.Data; using
StudentManagementService.Models;

namespace StudentManagementService.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class StudentsController : ControllerBase
    {
        private readonly StudentManagementServiceContext _context;

        public StudentsController(StudentManagementServiceContext context)
        {
            _context = context;
        }

        // GET: api/Students
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Student>>> GetStudent()
        {
            return await _context.Student.ToListAsync();
        }

        // GET: api/Students/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Student>> GetStudent(int id)
        {
            var student = await _context.Student.FindAsync(id);

            if (student == null)
            {
                return NotFound();
            }

            return student;
        }

        // PUT: api/Students/5
        [HttpPut("{id}")]
        public async Task<ActionResult> PutStudent(int id, Student student)
        {
            if (id != student.Id)
            {
                return BadRequest();
            }
        }
    }
}
```

```

        }

        _context.Entry(student).State = EntityState.Modified;

        try
        {
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!StudentExists(id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }

        return NoContent();
    }

    // POST: api/Students

    [HttpPost]
    public async Task<ActionResult<Student>> PostStudent(Student student)
    {
        _context.Student.Add(student);
        await _context.SaveChangesAsync();

        return CreatedAtAction("GetStudent", new { id = student.Id }, student);
    }

    // DELETE: api/Students/5
    [HttpDelete("{id}")]
    public async Task<IActionResult> DeleteStudent(int id)
    {
        var student = await _context.Student.FindAsync(id);
        if (student == null)
        {
            return NotFound();
        }

        _context.Student.Remove(student);
        await _context.SaveChangesAsync();

        return NoContent();
    }

    private bool StudentExists(int id)
    {
        return _context.Student.Any(e => e.Id == id);
    }
}

```

## VI. Capturi de ecran

Interfața principală:

Student Registration

Student Name

Course

Fee

	ID:	Student name	Course:	Fee:	Edit:	Delete:
*						

CRUD:

Form1

# Student Registration

Student Registration

Student Name

Sergiu

Course

React

Fee

790

Save

Clear

	ID	Student name	Course	Fee	Edit	Delete
▶	1	Mouad	math	1000	Edit	Delete
	2	Narcis	Java	500	Edit	Delete
✱						

Refresh

Form1

# Student Registration

Student Registration

Student Name

Course

Fee

Save

Clear

	ID	Student name	Course	Fee	Edit	Delete
▶	1	Mouad	math	1000	Edit	Delete
	2	Narcis	Java	500	Edit	Delete
	3	Sergiu	React	790	Edit	Delete
✱						

Refresh



Form1

Student Registration

Student Name

Course

Fee

Save

Clear

Student Registration

ID	Student name	Course	Fee	Edit	Delete
2	Narcis	Java	500	Edit	Delete
3	Sergiu	React	790	Edit	Delete
*					

Refresh

Form1

Student Registration

Student Name

Course

Fee

Save

Clear

Student Registration

ID	Student name	Course	Fee	Edit	Delete
2	Narcis	Java/Net	1000	Edit	Delete
3	Sergiu	React	790	Edit	Delete
*					

Refresh

MS SQL SERVER:

Object Explorer

Connect

MOUAD\_PC\SQLEXPRESS (SQL Server)

- Databases
  - System Databases
  - Database Snapshots
  - gcbt
    - Database Diagrams
    - Tables
      - System Tables
      - FileTables
      - External Tables
      - Graph Tables
      - dbo.student
        - Columns
        - Keys
        - Constraints
        - Triggers
        - Indexes
        - Statistics
  - Views
  - External Resources
  - Synonyms
  - Programmability

SQLQuery6.sql - M...AD\_PC\mouad (63)

```
SELECT TOP (1000) [id]
, [stdname]
, [course]
, [fee]
FROM [gcbt].[dbo].[student]
```


100 %

Results Messages

	id	stdname	course	fee
1	1	Mouad	math	1000
2	2	Narcis	Java	500
3	3	Sergiu	React	790

Serviciu web:

Aplicația mea utilizează un serviciu web care are ca referință OpenAI pentru a furniza funcționalități avansate. Acesta este motivul pentru care găsiți documentația și interfața de utilizare (Swagger) în care sunt prezentate și accesibile diferitele funcții ale serviciului.

 Swagger  
OpenAPI Specification

Select a definition: StudentManagementService v1

## StudentManagementService 1.0 OK

<http://localhost:5002/swagger/v1/swagger.json>

### Students

GET

/api/Students

POST

/api/Students

GET

/api/Students/{id}

PUT

/api/Students/{id}

DELETE

/api/Students/{id}

### WeatherForecast

GET

/WeatherForecast

### Schemas

DateOnly

DayOfWeek

Student

WeatherForecast

