МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №8

по дисциплине: **«**Программирование сетевых приложений**»**

на тему: «Программирование приложений с использованием

*WebSocket*»

Выполнил: студент гр. ИТП-41

Расшивалов Н.И.

Принял: преподаватель

Гуменников Е. Д.

Гомель 2022

**Цель:** изучить создание *web*-сервиса.

**Задание:**

Разработать HTTP­сервис по выбранной теме ­ реализовать CRUD (Create Read Update Delete) операции. Обязательно оолжны использоваться запросы в стиле REST:

GET ­ для получения данных

POST ­ для создания ресурса (сохранения данных)

PUT ­ для обновления ресурса (обновление данных)

DELETE ­ для удаления ресурса (удаления строк)

(рекомендуется использовать .Net, Java или Python, можно использовать другие языки программирования по согласованию с преподавателем). Приложение должно поставлять и получать данные в одном из следующих форматов: XML, JSON, YAML. На сервере использовать любую БД для хранения данных, рекомендуется использовать MySQL, PostgreSQL, MS SQL Server. Приветствуется использование не реляционных СУБД (например, MongoDB или Casandra). Клиентское приложение: это html­страница, где запросы (вызовы сервиса) осуществляются с использованием AJAX (рекомендуется использовать jQuery). Для студентов претендующих на оценку 9­10 следует

использовать любой из перечисленных JavaScript framework: AngularJS, Backbone, KnockoutJS, AmberJS или разработать мобильный клиент для Android, iOS или Windows Phone. Экзамен: ФИО студента, дата, оценка, группа.

**Ход работы:**

Созданный API представлен на рисунке 1.

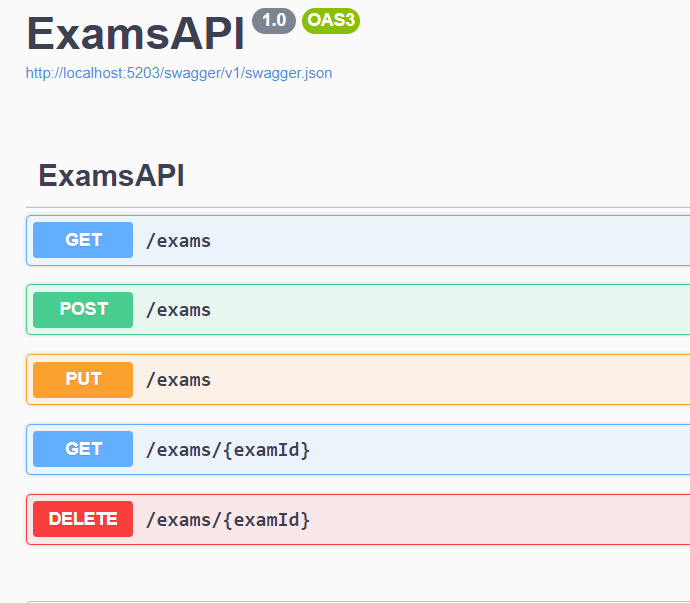


Рисунок 1 – Созданный *API*

Созданный клиент представлен на рисунке 2.

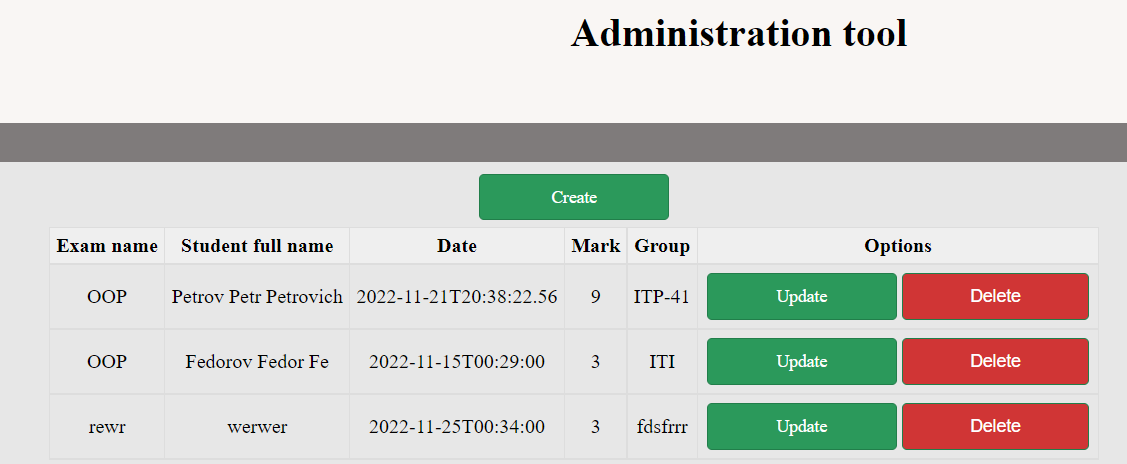


Рисунок 2 – Созданный клиент

Листинг программы представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы получены навыки работы с *web*-сервисами с использованием *REST*-архитектуры.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

using ExamsAPI.Models;

using ExamsAPI.Services;

using Microsoft.AspNetCore.Http.HttpResults;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddCors(options =>

{

options.AddDefaultPolicy(

policy =>

{

policy.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod();

});

});

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var examsSonnectionString = builder.Configuration.GetConnectionString("Exams");

builder.Services.AddSingleton(new ExamServiceOptions { ConnectionString = examsSonnectionString });

builder.Services.AddTransient<IExamService, ExamService>();

var app = builder.Build();

app.UseCors();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.MapGet("/exams/", async (IExamService examService) =>

{

return await examService.GetAsync();

})

.WithOpenApi();

app.MapGet("/exams/{examId}", async (int examId, IExamService examService) =>

{

var exam = await examService.GetAsync(examId);

if(exam == null)

{

return Results.NotFound();

}

return Results.Ok(exam);

})

.WithOpenApi();

app.MapDelete("/exams/{examId}", async (int examId, IExamService examService) =>

{

await examService.DeleteAsync(examId);

})

.WithOpenApi();

app.MapPost("/exams/", async (Exam exam, IExamService examService) =>

{

return await examService.CreateAsync(exam);

})

.WithOpenApi();

app.MapPut("/exams/", async (Exam exam, IExamService examService) =>

{

await examService.UpdateAsync(exam);

})

.WithOpenApi();

app.Run();