Integração Sistemas

Relatório Parte 4 dos Trabalhos Práticos

Autores:

João de Castro - **210100342**

Daniel Nunes - **200100306**

Alexandre Silva - **190100307**



**Filipe Madeira**

Ano Letivo 2023/2024

Contents

[Programas Utilizados 3](#_Toc168172302)

[3](#_Toc168172303)

[API 3](#_Toc168172304)

[Ficheiros 3](#_Toc168172305)

[WebSocketExtensions.cs 4](#_Toc168172306)

[WebSocketHandler.cs 4](#_Toc168172307)

[WebSocketHandler.cs 5](#_Toc168172308)

[Program.cs 5](#_Toc168172309)

[Cliente 9](#_Toc168172310)

[Ecrãs: 10](#_Toc168172311)

[Formulário de Login. 10](#_Toc168172312)

[Login feito com sucesso. 10](#_Toc168172313)

[Página principal e como aceder a página da parte 4 do trabalho. 11](#_Toc168172314)

[Ligar a conexão. 12](#_Toc168172315)

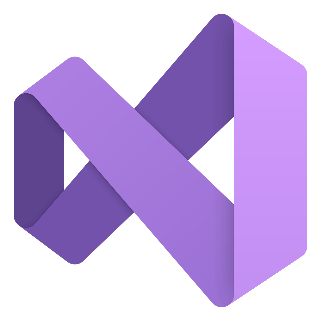
[Receber as informações enviadas. 12](#_Toc168172316)

[Ficheiros: 13](#_Toc168172317)

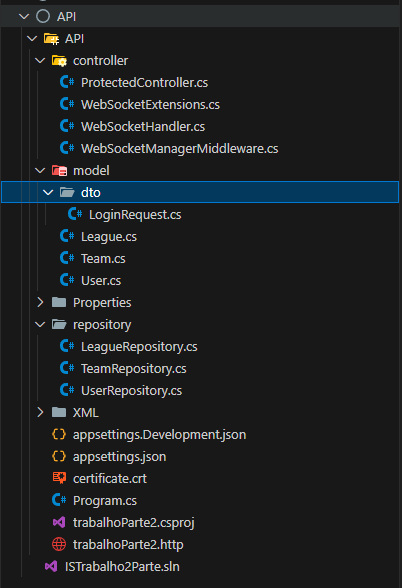
[Part4.cs: 13](#_Toc168172318)

# Programas Utilizados

# XAMPP - WikipediaVisual Studio Code icon PNG and SVG Vector Free Download



# API

 Nesta Imagem pode -se observar o E*xplorer* do visual Studio, com os ficheiros necessário para a API. Tendo uma importancia maior as pastas “controller”, que irá controlar a maneira como o utilizador irá interagir com a aplicação. “model”, que é responsavel por amazenar os dados que vem do back-end. “repository”, onde se encontra todas as chamadas SQL, e por ultimo o “Program.cs”, onde a API corre, pode ser associado a um *Main.*

## Ficheiros

### WebSocketExtensions.cs

public static class WebSocketExtensions {

    public static IServiceCollection AddWebSocketManager(this IServiceCollection services) {

        services.AddTransient<WebSocketHandler>();

        return services;

    }

    public static IApplicationBuilder MapWebSocketManager(this IApplicationBuilder app, PathString path, WebSocketHandler handler) {

        return app.Map(path, (x) => x.UseMiddleware<WebSocketManagerMiddleware>(handler));

    }

}

### WebSocketHandler.cs

using System.Net.WebSockets;

using System.Text;

public class WebSocketHandler {

    public async Task HandleAsync(HttpContext context, WebSocket socket) {

        var buffer = new byte[1024];

        WebSocketReceiveResult result;

        do {

            result = await socket.ReceiveAsync(new ArraySegment<byte>(buffer), CancellationToken.None);

            var message = Encoding.UTF8.GetString(buffer, 0, result.Count);

            // Handle received message (e.g., process it and send response)

            var responseMessage = $"Received: {message}";

            var responseBytes = Encoding.UTF8.GetBytes(responseMessage);

            await socket.SendAsync(new ArraySegment<byte>(responseBytes),

            result.MessageType, result.EndOfMessage, CancellationToken.None);

        }

        while (!result.CloseStatus.HasValue);

        await socket.CloseAsync(result.CloseStatus.Value, result.CloseStatusDescription, CancellationToken.None);

    }

}

### WebSocketHandler.cs

public class WebSocketManagerMiddleware {

    private readonly RequestDelegate \_next;

    private readonly WebSocketHandler \_webSocketHandler;

    public WebSocketManagerMiddleware(RequestDelegate next, WebSocketHandler webSocketHandler) {

        \_next = next;

        \_webSocketHandler = webSocketHandler;

    }

    public async Task InvokeAsync(HttpContext context) {

        if (!context.WebSockets.IsWebSocketRequest)

            return;

        var socket = await context.WebSockets.AcceptWebSocketAsync();

        await \_webSocketHandler.HandleAsync(context, socket);

    }

}

### Program.cs

using LeagueApi;

using TeamApi;

using UserApi;

using Microsoft.AspNetCore.Mvc;

using MySqlConnector;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.IdentityModel.Tokens;

using System.Text;

using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddWebSocketManager(); // parte 4

builder.Services.AddControllers();

builder.Services.AddTransient<UserRepository>();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(c => {

    c.SwaggerDoc("v1", new OpenApiInfo { Title = "IS", Version = "v1" });

    c.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme() {

        Name = "Authorization",

        Type = SecuritySchemeType.ApiKey,

        Scheme = "Bearer",

        BearerFormat = "JWT",

        In = ParameterLocation.Header,

        Description = "JWT Authorization header using the Bearer scheme.\r\n\r\n Enter 'Bearer' [space] and then your token in the text input below.\r\n\r\nExample: \"Bearer 12345abcdef\"",

    });

    c.AddSecurityRequirement(new OpenApiSecurityRequirement { {

        new OpenApiSecurityScheme {

            Reference = new OpenApiReference {

                Type = ReferenceType.SecurityScheme,

                Id = "Bearer"

            }

        },

        new string[] {}

    } });

});

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme).AddJwtBearer(options => {

    options.TokenValidationParameters = new TokenValidationParameters {

        ValidateIssuer = false,

        ValidateAudience = false,

        ValidateLifetime = true,

        ValidateIssuerSigningKey = true,

        //ValidIssuer = "your\_issuer",

        //ValidAudience = "your\_audience",

        IssuerSigningKey = new

        SymmetricSecurityKey(Encoding.UTF8.GetBytes("SecretKeywqewqeqqqqqqqqqqqweeeeeeeeeeeeeeeeeeeqweqe"))

    };

});

builder.Services.AddMySqlDataSource(builder.Configuration.GetConnectionString("Default")!);

var app = builder.Build();

var webSocketPort = builder.Configuration.GetValue<int>("WebSocketServer:Port");  // parte 4

if (app.Environment.IsDevelopment()) {

    app.UseSwagger();

    app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.MapControllers();

// ------------- User ------------- //

app.MapGet("/api/user", async (MySqlDataSource db) => {

    var repository = new UserRepository(db);

    return await repository.GetAllUserAsync();

});

app.MapGet("/api/user/xml", async (MySqlDataSource db) => {

    var repository = new UserRepository(db);

    return await repository.GetAllUserAsync();

});

app.MapPost("/api/user", async ([FromServices] MySqlDataSource db, [FromBody] User body) => {

    var repository = new UserRepository(db);

    await repository.InsertUserAsync(body);

    return body;

});

// . . .

// ------------- League ------------- //

app.MapGet("/api/league", async (MySqlDataSource db) => {

    var repository = new LeagueRepository(db);

    return await repository.GetAllLeagueAsync();

});

app.MapGet("/api/league/xml", async (MySqlDataSource db) => {

    var repository = new LeagueRepository(db);

    return await repository.GetAllLeagueAsync();

});

app.MapPost("/api/league", async ([FromServices] MySqlDataSource db, [FromBody] League body) => {

    var repository = new LeagueRepository(db);

    await repository.InsertLeagueAsync(body);

    return body;

});

app.MapPut("/api/league/{id}", async (int id, [FromServices] MySqlDataSource db, [FromBody] League body) => {

    var repository = new LeagueRepository(db);

    var result = await repository.FindOneAsync(id);

    if (result is null)

        return Results.NotFound();

    result.leagueName = body.leagueName;

    result.leagueCountry = body.leagueCountry;

    result.leagueMaxCap = body.leagueMaxCap;

    await repository.UpdateAsync(result);

    return Results.Ok(result);

});

app.MapDelete("/api/league/{id}", async ([FromServices] MySqlDataSource db, int id) => {

    var repository = new LeagueRepository(db);

    var result = await repository.FindOneAsync(id);

    if (result is null)

        return Results.NotFound();

    await repository.DeleteAsync(result);

    return Results.NoContent();

});

// ------------- Team ------------- //

app.MapGet("/api/team", async (MySqlDataSource db) => {

    var repository = new TeamRepository(db);

    return await repository.GetAllTeamAsync();

});

app.MapPost("/api/team", async ([FromServices] MySqlDataSource db, [FromBody] Team body) => {

    var repository = new TeamRepository(db);

    await repository.InsertTeamAsync(body);

    return body;

});

app.MapPut("/api/team/{id}", async (int id, [FromServices] MySqlDataSource db, [FromBody] Team body) => {

    var repository = new TeamRepository(db);

    var result = await repository.FindOneAsync(id);

    if (result is null)

        return Results.NotFound();

    result.teamName = body.teamName;

    result.FkIdLeague = body.FkIdLeague;

    await repository.UpdateAsync(result);

    return Results.Ok(result);

});

app.MapDelete("/api/team/{id}", async ([FromServices] MySqlDataSource db, int id) => {

    var repository = new TeamRepository(db);

    var result = await repository.FindOneAsync(id);

    if (result is null)

        return Results.NotFound();

    await repository.DeleteAsync(result);

    return Results.NoContent();

});

app.UseWebSockets();

app.MapWebSocketManager("/ws", new WebSocketHandler());

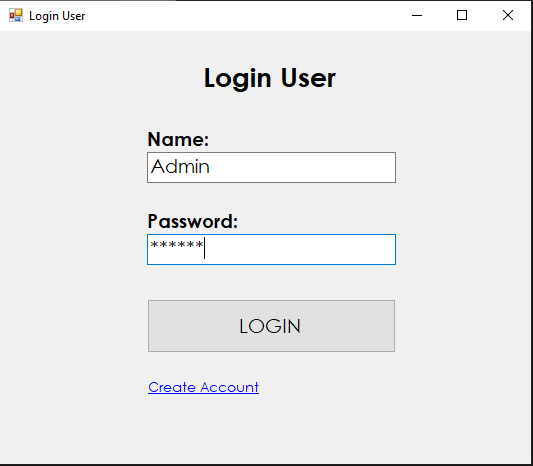
app.Run($"http://localhost:{webSocketPort}");

// app.Run();

# Cliente

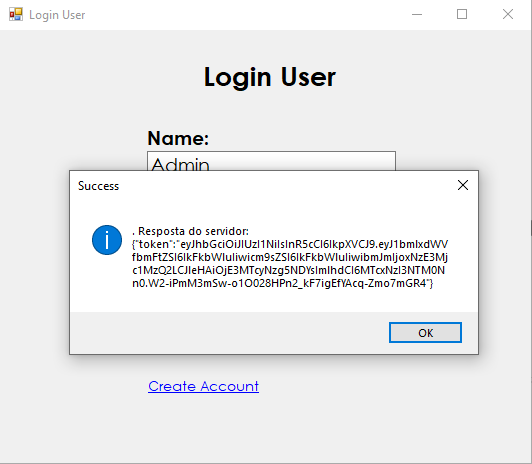
## Ecrãs:

### Formulário de Login.



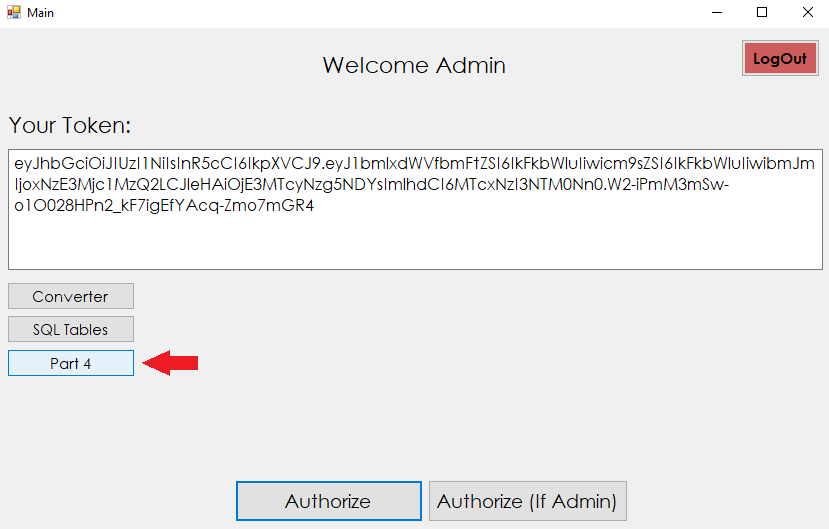
### Login feito com sucesso.

Ao serem introduzidos um nome e password correspondete ao nome colocado a aplicação irá mostrar uma mendagem de sucesso, e nela um token corresponde a esse utilziador, para porterior poder autorizar a sua presença.

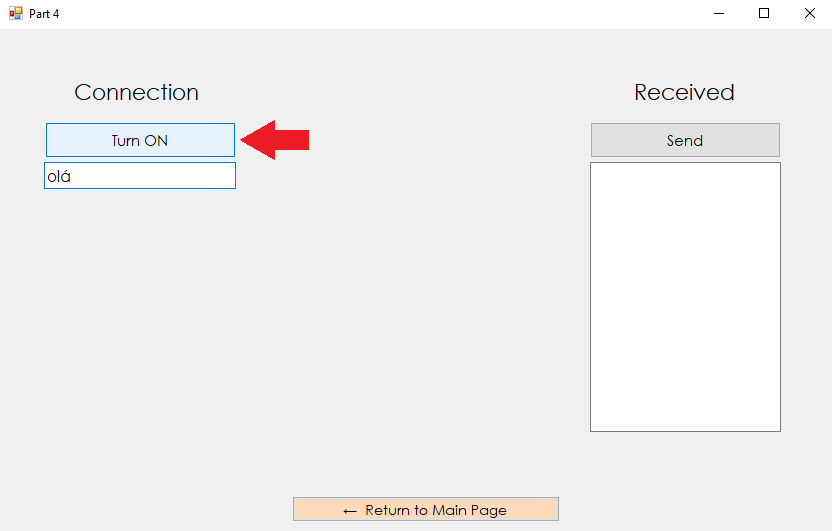


### Página principal e como aceder a página da parte 4 do trabalho.

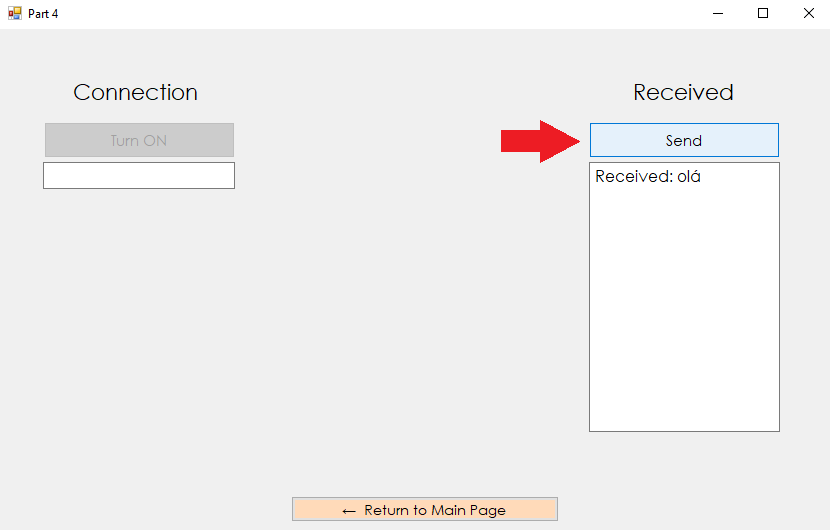
De modo a aceder ao *Form* da parte 4 é apenas necessário carregar no botão que diz “Part 4”.



### Ligar a conexão.

**

### Receber as informações enviadas.

**

## Ficheiros:

### Part4.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net.Http;

using System.Net.WebSockets;

using System.Text;

using System.Text.Json;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms;

using static System.Net.Mime.MediaTypeNames;

namespace WindowsFormsIS

{

public partial class Part4 : Form

{

private ClientWebSocket \_clientWebSocket;

public Part4()

{

InitializeComponent();

}

private async void BtnConnection\_Click(object sender, EventArgs e)

{

try

{

\_clientWebSocket = new ClientWebSocket(); //cria uma instância de ClientWebSocket

await \_clientWebSocket.ConnectAsync(new Uri("ws://localhost:5000/ws"), CancellationToken.None); //estabelece uma conexão com o servidor WebSocket em ws://localhost:5000/ws

\_ = ReceiveMessagesAsync(); //Inicia a receção de mensagens chamando o método ReceiveMessagesAsync()

btnConnection.Enabled = false; //desabilita o botão de conexão após estabelecimento de conexão

textBoxConnection.Enabled = true; //habilita a caixa de mensagem de entrada após estabelecimento de conexão

btnSend.Enabled = true; // habilita o botão de envio após estabelecimento de conexão

}

catch (Exception ex)

{

MessageBox.Show($"Erro na conexão: {ex.Message}");

}

}

private async Task ReceiveMessagesAsync() //responsável por receber mensagens do servidor WebSocket enquanto a conexão estiver aberta

{

var buffer = new byte[1024];

while (\_clientWebSocket.State == WebSocketState.Open) //aguarda até que o estado da conexão seja aberto

{

var result = await \_clientWebSocket.ReceiveAsync(new ArraySegment<byte>(buffer), CancellationToken.None); //chama ReceiveAsync() para receber uma mensagem do servidor

var message = Encoding.UTF8.GetString(buffer, 0, result.Count);

//converte os dados numa string UTF-8

textBoxSend.Text = message; //atualiza a caixa de texto txtReceivedMessage com a mensagem recebida

}

}

private async void btnSend\_Click(object sender, EventArgs e)

{

if (\_clientWebSocket.State == WebSocketState.Open) //verifica se a conexão WebSocket está aberta

{

var message = Encoding.UTF8.GetBytes(textBoxConnection.Text); // converte o texto da caixa txtMessage.Text em bytes UTF-8

await \_clientWebSocket.SendAsync(new ArraySegment<byte>(message), WebSocketMessageType.Text, true, CancellationToken.None); //envia a mensagem para o servidor usando SendAsync()

textBoxConnection.Text = ""; // limpa o conteúdo da caixa de mensagem de entrada

}

}

private void BtnGoBack\_Click(object sender, EventArgs e)

{

Main mainPage = new Main();

this.Hide();

mainPage.Show();

}

}

}