**Приложение В**

**Листинги**

**Листинг В.1 – Класс аутентификации пользователей**

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Claims;

using System.Security.Cryptography;

using System.Threading.Tasks;

using YourVitebskWebServiceApp.APIServiceInterfaces;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Text;

using Microsoft.Extensions.Configuration;

using YourVitebskWebServiceApp.APIModels;

using System.IO;

using Microsoft.AspNetCore.Hosting;

using YourVitebskWebServiceApp.Repositories;

using Microsoft.AspNetCore.Http;

namespace YourVitebskWebServiceApp.APIServices

{

public class AuthService : IAuthService

{

private readonly YourVitebskDBContext \_context;

private readonly IConfiguration \_configuration;

private readonly ImageService \_imageService;

private readonly IWebHostEnvironment \_appEnvironment;

public AuthService(YourVitebskDBContext context, IConfiguration configuration, IWebHostEnvironment appEnvironment)

{

\_context = context;

\_configuration = configuration;

\_appEnvironment = appEnvironment;

\_imageService = new ImageService(appEnvironment);

}

public static void CreatePasswordHash(string password, out byte[] passwordHash, out byte[] passwordSalt)

{

using (var hmac = new HMACSHA512())

{

passwordSalt = hmac.Key;

passwordHash = hmac.ComputeHash(Encoding.UTF8.GetBytes(password));

}

}

public static bool VerifyPassword(string password, byte[] passwordHash, byte[] passwordSalt)

{

using (var hmac = new HMACSHA512(passwordSalt))

{

var computedHash = hmac.ComputeHash(Encoding.UTF8.GetBytes(password));

return computedHash.SequenceEqual(passwordHash);

}

}

public string CreateToken(Models.User user)

{

string image = "";

if (Directory.Exists($"{\_appEnvironment.WebRootPath}/images/users/{user.UserId}"))

{

image = Directory.GetFiles($"{\_appEnvironment.WebRootPath}/images/users/{user.UserId}").Select(x => Path.GetFileName(x)).First();

}

var claims = new List<Claim>()

{

new Claim(nameof(user.UserId), user.UserId.ToString()),

new Claim(nameof(user.Email), user.Email),

new Claim(nameof(user.FirstName), user.FirstName),

new Claim(nameof(user.LastName), user.LastName),

new Claim(nameof(user.PhoneNumber), user.PhoneNumber),

new Claim(nameof(user.IsVisible), user.IsVisible.ToString()),

new Claim("Image", image)

};

var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration.GetSection("AppSettings:Token").Value));

var credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha512Signature);

var token = new JwtSecurityToken(claims: claims,expires: DateTime.Now.AddDays(1), signingCredentials: credentials);

return new JwtSecurityTokenHandler().WriteToken(token);

}

public async Task<string> Register(UserRegisterDTO userData)

{

try

{

if (await \_context.Users.AnyAsync(x => x.Email == userData.Email))

{

throw new ArgumentException("Пользователь с таким email уже существует!");

}

CreatePasswordHash(userData.Password, out byte[] passwordHash, out byte[] passwordSalt);

var user = new Models.User

{

UserId = null,

Email = userData.Email,

PasswordHash = passwordHash,

PasswordSalt = passwordSalt,

FirstName = userData.FirstName,

LastName = userData.LastName,

PhoneNumber = null,

RoleId = 1,

IsVisible = true,

};

\_context.Users.Add(user);

await \_context.SaveChangesAsync();

return CreateToken(user);

}

catch (ArgumentException e) { throw e; }

}

public async Task<string> Login(UserLoginDTO userData)

{

Models.User user = await \_context.Users.FirstOrDefaultAsync(x => x.Email == userData.Email);

if (user == null) { throw new ArgumentException("Неверные логин и(или) пароль"); }

if (!VerifyPassword(userData.Password, user.PasswordHash, user.PasswordSalt)) { throw new ArgumentException("Неверные логин и(или) пароль"); }

return CreateToken(user);

}

}

}