**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА №13**

**Дисциплина:** BackEnd-разработка

**Тема:** Добавление авторизации и аутентификации в веб-приложение на основе ASP.NET Core

**Выполнил: студент группы 231-332**

**Чесноков Александр Евгеньевич**

(Фамилия И.О. обучающегося)

**Дата, подпись** 01.10.2025 

(Дата)(Подпись)

**Проверил:** \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_*

(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_ \_\_\_\_\_\_\_\_*

(Дата)(Подпись)

**Москва**

**2025**

**Цель:**

Освоить процесс добавления механизмов аутентификации и авторизации в веб-приложение на платформе ASP.NET Core для обеспечения безопасности приложения и данных пользователей.

**Ход работы:**

На первом этапе был создан новый проект типа ASP.NET Core Web API, в котором настроена инфраструктура аутентификации с использованием JWT (JSON Web Token). Данный способ широко применяется в современных API, так как позволяет проверять подлинность пользователей без постоянного хранения состояния на сервере.

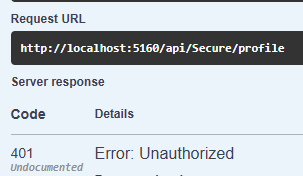
В файле Program.cs были добавлены сервисы для аутентификации (AddAuthentication) и авторизации (AddAuthorization). Настройка JWT выполнена на основе симметричного ключа, заданного в конфигурационном файле appsettings.json.

Контроллер AuthController реализует базовые механизмы регистрации и входа пользователей. При регистрации создаётся пользователь с определённой ролью (например, “User” или “Admin”), а при входе система проверяет его данные и выдает **JWT-токен**, который клиент должен использовать в последующих запросах.

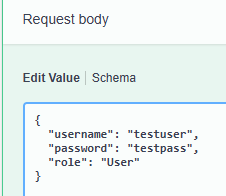
Контроллер SecureController содержит два метода, доступ к которым ограничен с помощью атрибутов [Authorize] и [Authorize(Roles = "Admin")]. Первый метод (/api/secure/profile) доступен всем авторизованным пользователям, второй (/api/secure/admin) — только тем, у кого установлена роль “Admin”.

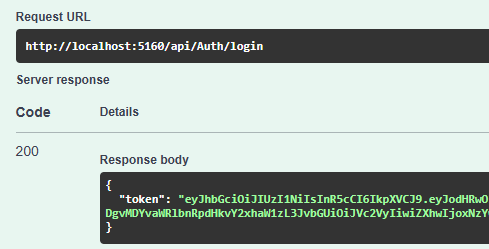
Тестирование приложения проводилось через **Swagger UI**, который поддерживает ввод токенов для авторизации. После регистрации и получения токена, он вставляется в раздел “Authorize”, после чего запросы к защищённым маршрутам выполняются успешно. При попытке обращения без токена или с неправильными правами сервер возвращает ошибку 401 Unauthorized или 403 Forbidden.

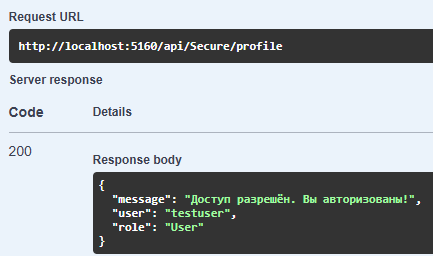
**Скриншоты работы приложения:**

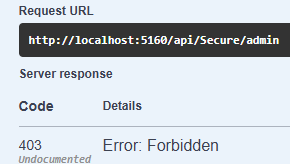
Попытка GET запроса без авторизации:  
GET /profile:  
****

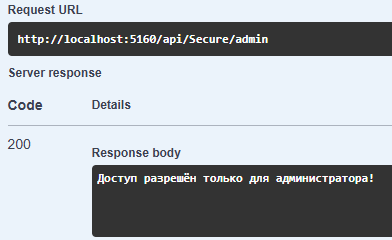
Авторизация и попытка запроса с правами “User”:

POST /register:  


POST /login:  


GET /profile:  


GET /admin:  


Попытка запроса POST /admin: с правами “Admin”:  


**Листинг:**

**Program.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Authentication.JwtBearer;  using Microsoft.IdentityModel.Tokens;  using Microsoft.OpenApi.Models;  using System.Text;  var builder = WebApplication.CreateBuilder(args);  // *контроллеры*  builder.Services.AddControllers();  // *Swagger*  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen(options =>  {      options.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme      {          Name = "Authorization",          Type = SecuritySchemeType.ApiKey,          Scheme = "Bearer",          BearerFormat = "JWT",          In = ParameterLocation.Header,          Description = "Введите токен в формате: Bearer {токен}"      });      options.AddSecurityRequirement(new OpenApiSecurityRequirement      {          {              new OpenApiSecurityScheme              {                  Reference = new OpenApiReference                  {                      Type = ReferenceType.SecurityScheme,                      Id = "Bearer"                  }              },              Array.Empty<string>()          }      });  });  // *JWT-аутентификация*  var key = Encoding.UTF8.GetBytes(builder.Configuration["Jwt:Key"]!);  builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)      .AddJwtBearer(options =>      {          options.TokenValidationParameters = new TokenValidationParameters          {              ValidateIssuer = true,              ValidateAudience = true,              ValidateLifetime = true,              ValidateIssuerSigningKey = true,              ValidIssuer = builder.Configuration["Jwt:Issuer"],              ValidAudience = builder.Configuration["Jwt:Audience"],              IssuerSigningKey = new SymmetricSecurityKey(key)          };      });  builder.Services.AddAuthorization();  var app = builder.Build();  if (app.Environment.IsDevelopment())  {      app.UseSwagger();      app.UseSwaggerUI();  }  // *Подключаем аутентификацию и авторизацию*  app.UseAuthentication();  app.UseAuthorization();  app.MapControllers();  app.Run(); |

**Controllers/AuthController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.IdentityModel.Tokens;  using System.IdentityModel.Tokens.Jwt;  using System.Security.Claims;  using System.Text;  using Lab13.Models;  namespace Lab13.Controllers;  /// *Контроллер для регистрации и входа пользователей*  [ApiController]  [Route("api/[controller]")]  public class AuthController : ControllerBase  {      private static readonly List<User> Users = new();      private readonly IConfiguration \_config;      public AuthController(IConfiguration config)      {          \_config = config;      }      [HttpPost("register")]      public IActionResult Register(RegisterModel model)      {          if (Users.Any(u => u.Username == model.Username))              return BadRequest("Пользователь с таким именем уже существует.");          Users.Add(new User          {              Username = model.Username,              Password = model.Password,              Role = model.Role          });          return Ok("Регистрация успешна.");      }      [HttpPost("login")]      public IActionResult Login(LoginModel model)      {          var user = Users.FirstOrDefault(u => u.Username == model.Username && u.Password == model.Password);          if (user == null)              return Unauthorized("Неверное имя пользователя или пароль.");          var token = GenerateJwtToken(user);          return Ok(new { token });      }      private string GenerateJwtToken(User user)      {          var claims = new List<Claim>          {              new Claim(ClaimTypes.Name, user.Username),              new Claim(ClaimTypes.Role, user.Role)          };          var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_config["Jwt:Key"]!));          var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);          var token = new JwtSecurityToken(              issuer: \_config["Jwt:Issuer"],              audience: \_config["Jwt:Audience"],              claims: claims,              expires: DateTime.Now.AddMinutes(30),              signingCredentials: creds);          return new JwtSecurityTokenHandler().WriteToken(token);      }  } |

**Controllers/SecureController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  using System.Security.Claims;  namespace Lab13.Controllers  {  [ApiController]  [Route("api/[controller]")]  public class SecureController : ControllerBase  {  // Доступ только для авторизованных пользователей  [Authorize]  [HttpGet("profile")]  public IActionResult GetProfile()  {  // Находим роль безопасно через ClaimTypes.Role  var role = User.FindFirst(ClaimTypes.Role)?.Value;  return Ok(new  {  message = "Доступ разрешён. Вы авторизованы!",  user = User.Identity?.Name,  role = role  });  }  // Доступ только для админов  [Authorize(Roles = "Admin")]  [HttpGet("admin")]  public IActionResult AdminAccess()  {  return Ok("Доступ разрешён только для администратора!");  }  }  } |

**Models/LoginModel.cs:**

|  |
| --- |
| namespace Lab13.Models;  public class LoginModel  {      public string Username { get; set; } = string.Empty;      public string Password { get; set; } = string.Empty;  } |

**Models/RegisterModel.cs:**

|  |
| --- |
| namespace Lab13.Models;  public class RegisterModel  {      public string Username { get; set; } = string.Empty;      public string Password { get; set; } = string.Empty;      public string Role { get; set; } = string.Empty;  } |

**Models/User.cs:**

|  |
| --- |
| namespace Lab13.Models;  /// *Простая модель пользователя*  public class User  {      public string Username { get; set; } = string.Empty;      public string Password { get; set; } = string.Empty;      public string Role { get; set; } = string.Empty;  } |

**Ссылка на репозиторий:** <https://github.com/avoryyy/backend_mospoly>