**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 №15**

**Дисциплина:** BackEnd-разработка

**Тема:** Контроль доступа к адресам 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:**

Научиться реализовывать механизмы перенаправления запросов через промежуточное программное обеспечение (middleware) для контроля доступа к различным адресам в веб-приложении на платформе [ASP.NET](https://asp.net/) Core.

**Ход работы:**

В файле Program.cs настроена аутентификация с использованием cookie и добавлено базовое управление ролями пользователей без подключения к базе данных. Для демонстрации работы механизма были созданы два пользователя: admin с ролью Admin и user с ролью User.

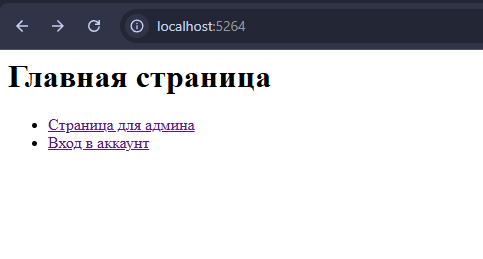
Для контроля доступа реализован middleware, который проверяет роль пользователя при обращении к маршрутам, начинающимся с /Admin. Если пользователь не имеет роли Admin, он перенаправляется на страницу с сообщением о запрете доступа. Этот подход позволяет централизованно управлять доступом без необходимости дублировать проверку в каждом контроллере.

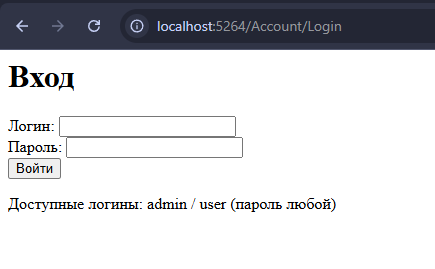
Контроллер AdminController дополнительно снабжён атрибутом [Authorize(Roles = "Admin")], что обеспечивает защиту на уровне маршрута и демонстрирует комбинирование middleware и встроенной авторизации ASP.NET Core.

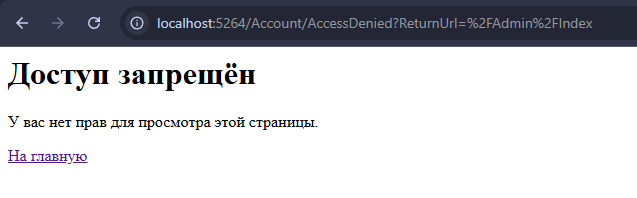
Контроллер AccountController реализует вход и выход пользователей, используя claims для хранения имени и роли, что позволяет middleware корректно перенаправлять пользователей на основе их прав. Вход реализован через простую форму HTML, где логины admin и user определяют соответствующую роль, а пароль не проверяется для упрощённого примера.

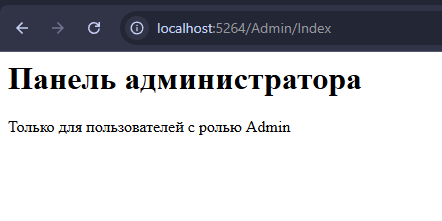
Приложение протестировано: пользователь admin имеет доступ к панели администратора, а обычный пользователь перенаправляется на страницу “Доступ запрещён”. Также проверено, что после выхода из аккаунта доступ к защищённым страницам блокируется.

**Скриншоты работы приложения:**



****





**Листинг:**

**Program.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Authentication.Cookies;  using Microsoft.AspNetCore.Authorization;  var builder = WebApplication.CreateBuilder(args);  builder.Services.AddControllersWithViews();  builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)  .AddCookie(options =>  {  options.LoginPath = "/Account/Login";  options.AccessDeniedPath = "/Account/AccessDenied";  });  builder.Services.AddAuthorization();  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen();  var app = builder.Build();  app.UseStaticFiles();  app.UseSwagger();  app.UseSwaggerUI();  app.UseRouting();  app.UseAuthentication();  app.UseAuthorization();  // Middleware для перенаправления пользователей в зависимости от роли  app.Use(async (context, next) =>  {  if (context.User.Identity?.IsAuthenticated == true)  {  if (context.Request.Path.StartsWithSegments("/Admin") &&  !context.User.IsInRole("Admin"))  {  context.Response.Redirect("/Account/AccessDenied");  return;  }  }  await next();  });  app.MapControllerRoute(  name: "default",  pattern: "{controller=Home}/{action=Index}/{id?}"  );  app.Run(); |

**Controllers/HomeController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  namespace Lab15.Controllers  {      public class HomeController : Controller      {          public IActionResult Index()          {              return Content(@"                  <html>                      <body>                          <h1>Главная страница</h1>                          <ul>                              <li><a href='/Admin/Index'>Страница для админа</a></li>                              <li><a href='/Account/Login'>Вход в аккаунт</a></li>                          </ul>                      </body>                  </html>", "text/html; charset=utf-8");          }      }  } |

**Controllers/AdminController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Authorization;  using Microsoft.AspNetCore.Mvc;  namespace Lab15.Controllers  {      [Authorize(Roles = "Admin")]      public class AdminController : Controller      {          public IActionResult Index()          {              return Content("<h1>Панель администратора</h1><p>Только для пользователей с ролью Admin</p>", "text/html; charset=utf-8");          }      }  } |

**Controllers/AccountController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Authentication;  using Microsoft.AspNetCore.Authentication.Cookies;  using Microsoft.AspNetCore.Mvc;  using System.Security.Claims;  namespace Lab15.Controllers  {      public class AccountController : Controller      {          // *Вход пользователя*          [HttpGet]          public IActionResult Login()          {              return Content(@"                  <html>                      <body>                          <h1>Вход</h1>                          <form method='post'>                              Логин: <input name='username' /><br/>                              Пароль: <input name='password' type='password' /><br/>                              <button type='submit'>Войти</button>                          </form>                          <p>Доступные логины: admin / user (пароль любой)</p>                      </body>                  </html>", "text/html; charset=utf-8");          }          [HttpPost]          public async Task<IActionResult> Login(string username, string password)          {              // *Простая проверка в памяти*              string role = username.ToLower() == "admin" ? "Admin" : "User";              var claims = new List<Claim>              {                  new Claim(ClaimTypes.Name, username),                  new Claim(ClaimTypes.Role, role)              };              var identity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);              var principal = new ClaimsPrincipal(identity);              await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, principal);              return Redirect("/");          }          public async Task<IActionResult> Logout()          {              await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);              return Redirect("/");          }          public IActionResult AccessDenied()          {              return Content("<h1>Доступ запрещён</h1><p>У вас нет прав для просмотра этой страницы.</p><a href='/'>На главную</a>", "text/html; charset=utf-8");          }      }  } |

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