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

**Выполнил(а): студент(ка) группы \_\_231-336\_\_**

\_\_\_\_\_\_\_\_Канищев И.М\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия И.О.)

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

(Дата) (Подпись)

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

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

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

(Дата) (Подпись)

**Замечания:**

**Москва**

**2025**

## **Описание реализации**

В рамках данного практического занятия была реализована система аутентификации и авторизации в веб-приложении на платформе ASP.NET Core 8.0. Были выполнены следующие задачи:

### 1. Настройка механизма аутентификации

Для настройки аутентификации использовалась встроенная система Identity в ASP.NET Core, которая предоставляет готовые решения для управления пользователями, ролями и аутентификацией.

### 2. Реализация механизма авторизации

Были созданы различные роли пользователей (Admin, User) и настроены соответствующие политики доступа для каждой роли.

### 3. Использование атрибутов авторизации

Для ограничения доступа к контроллерам и методам применялись атрибуты [Authorize], [AllowAnonymous], а также кастомные политики авторизации.

### Главный класс Program.cs

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.DataProtection;

var builder = WebApplication.CreateBuilder(args);

// ВАЖНО: Data Protection ДО аутентификации

builder.Services.AddDataProtection()

.SetApplicationName("AuthExample");

// Настройка аутентификации

builder.Services.AddAuthentication(options =>

{

options.DefaultScheme = CookieAuthenticationDefaults.AuthenticationScheme;

options.DefaultChallengeScheme = CookieAuthenticationDefaults.AuthenticationScheme;

options.DefaultSignInScheme = CookieAuthenticationDefaults.AuthenticationScheme;

options.DefaultAuthenticateScheme = CookieAuthenticationDefaults.AuthenticationScheme;

})

.AddCookie(CookieAuthenticationDefaults.AuthenticationScheme, options =>

{

options.Cookie.Name = "AuthCookie";

options.LoginPath = "/Account/Login";

options.AccessDeniedPath = "/Account/AccessDenied";

options.ExpireTimeSpan = TimeSpan.FromMinutes(30);

options.SlidingExpiration = true;

// Критически важные настройки

options.Cookie.HttpOnly = true;

options.Cookie.SecurePolicy = CookieSecurePolicy.None; // Для HTTP

options.Cookie.SameSite = SameSiteMode.Lax;

options.Cookie.Path = "/";

// Отладка

options.Events = new CookieAuthenticationEvents

{

OnSigningIn = context =>

{

Console.WriteLine($"🔐 SIGNING IN: {context.Principal.Identity.Name}");

return Task.CompletedTask;

},

OnSignedIn = context =>

{

Console.WriteLine($"✅ SIGNED IN: {context.Principal.Identity.Name}");

return Task.CompletedTask;

},

OnValidatePrincipal = context =>

{

Console.WriteLine($"🔍 VALIDATING: {context.Principal?.Identity?.Name}");

return Task.CompletedTask;

}

};

});

builder.Services.AddAuthorization();

builder.Services.AddControllersWithViews();

builder.Services.AddHttpContextAccessor();

builder.Services.AddScoped<IUserService, UserService>();

var app = builder.Build();

// Конфигурация pipeline

app.UseStaticFiles();

app.UseRouting();

// ВАЖНО: Правильный порядок!

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

### app.Run();

### Сервисы

### UserService.cs

public interface IUserService

{

User Authenticate(string username, string password);

}

public class UserService : IUserService

{

// В реальном приложении - база данных

private readonly List<User> \_users = new()

{

new User { Id = 1, Username = "admin", Password = "admin123", Role = "Admin" },

new User { Id = 2, Username = "user", Password = "user123", Role = "User" }

};

public User Authenticate(string username, string password)

{

return \_users.FirstOrDefault(x =>

x.Username == username && x.Password == password);

}

### }

## **Модели**

User.cs

public class User

{

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Role { get; set; }

}

public class LoginModel

{

public string Username { get; set; }

public string Password { get; set; }

}

**Контроллеры**

AccountController.cs

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Authentication;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Security.Claims;

public class AccountController : Controller

{

private readonly IUserService \_userService;

public AccountController(IUserService userService)

{

\_userService = userService;

}

[HttpGet]

[AllowAnonymous]

public IActionResult Login()

{

Console.WriteLine("📄 Login page loaded");

return View();

}

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginModel model)

{

Console.WriteLine($"🔐 Login attempt: {model.Username}");

if (!ModelState.IsValid)

{

Console.WriteLine("❌ Model state invalid");

return View(model);

}

var user = \_userService.Authenticate(model.Username, model.Password);

if (user == null)

{

Console.WriteLine("❌ User not found or wrong password");

ModelState.AddModelError("", "Неверные учетные данные");

return View(model);

}

Console.WriteLine($"✅ User authenticated: {user.Username}, Role: {user.Role}");

// Создаем claims

var claims = new List<Claim>

{

new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),

new Claim(ClaimTypes.Name, user.Username),

new Claim(ClaimTypes.Role, user.Role),

new Claim("AuthenticatedAt", DateTime.Now.ToString())

};

var claimsIdentity = new ClaimsIdentity(

claims,

CookieAuthenticationDefaults.AuthenticationScheme);

var authProperties = new AuthenticationProperties

{

IsPersistent = true,

ExpiresUtc = DateTimeOffset.UtcNow.AddMinutes(30),

IssuedUtc = DateTimeOffset.UtcNow,

AllowRefresh = true

};

try

{

Console.WriteLine("🚀 Attempting SignInAsync...");

await HttpContext.SignInAsync(

CookieAuthenticationDefaults.AuthenticationScheme,

new ClaimsPrincipal(claimsIdentity),

authProperties);

Console.WriteLine($"🎉 SignInAsync SUCCESS for {user.Username}");

// Проверяем сразу

var authResult = await HttpContext.AuthenticateAsync();

Console.WriteLine($"📊 Immediate auth check: {authResult.Succeeded}");

return RedirectToAction("Index", "Home");

}

catch (Exception ex)

{

Console.WriteLine($"💥 SignInAsync ERROR: {ex.Message}");

ModelState.AddModelError("", "Ошибка при входе в систему");

return View(model);

}

}

[HttpPost]

[Authorize]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

Console.WriteLine("👋 User logged out");

return RedirectToAction("Index", "Home");

}

[HttpGet]

[AllowAnonymous]

public IActionResult AccessDenied()

{

return View();

}

[HttpGet]

[AllowAnonymous]

public async Task<IActionResult> DebugAuth()

{

var authResult = await HttpContext.AuthenticateAsync();

var debugInfo = new

{

User = new

{

IsAuthenticated = User.Identity.IsAuthenticated,

UserName = User.Identity.Name,

AuthenticationType = User.Identity.AuthenticationType

},

AuthenticationResult = new

{

Succeeded = authResult.Succeeded,

Principal = authResult.Principal != null ? new

{

IsAuthenticated = authResult.Principal.Identity.IsAuthenticated,

Name = authResult.Principal.Identity.Name

} : null,

Failure = authResult.Failure?.Message

},

Cookies = Request.Cookies.Select(c => new { c.Key, ValueLength = c.Value?.Length }).ToList(),

Headers = Request.Headers.Where(h => h.Key.StartsWith("Cookie")).ToDictionary(h => h.Key, h => h.Value)

};

return Json(debugInfo);

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> TestLogin()

{

// Тестовый метод для прямого входа

var claims = new List<Claim>

{

new Claim(ClaimTypes.NameIdentifier, "999"),

new Claim(ClaimTypes.Name, "testuser"),

new Claim(ClaimTypes.Role, "Admin"),

new Claim("Test", "DirectLogin")

};

var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);

await HttpContext.SignInAsync(

CookieAuthenticationDefaults.AuthenticationScheme,

new ClaimsPrincipal(claimsIdentity),

new AuthenticationProperties { IsPersistent = true });

return RedirectToAction("Index", "Home");

}

}

HomeController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller

{

private readonly IAuthorizationService \_authorizationService;

public HomeController(IAuthorizationService authorizationService)

{

\_authorizationService = authorizationService;

}

[AllowAnonymous] // Доступно без авторизации

public IActionResult Index()

{

return View();

}

[Authorize] // Требуется любая авторизация

public IActionResult Profile()

{

return View();

}

[Authorize(Roles = "Admin")] // Только для администраторов

public IActionResult AdminPanel()

{

return View();

}

[Authorize(Roles = "User,Admin")] // Для пользователей и администраторов

public IActionResult UserArea()

{

return View();

}

[Authorize(Policy = "AdminOnly")] // Использование политики

public IActionResult AdminSettings()

{

return View();

}

[Authorize]

public async Task<IActionResult> CheckPermissions()

{

var result = new

{

IsAuthenticated = User.Identity.IsAuthenticated,

UserName = User.Identity.Name,

IsInRoleAdmin = User.IsInRole("Admin"),

IsInRoleUser = User.IsInRole("User"),

CanAccessAdminPanel = (await \_authorizationService.AuthorizeAsync(User, "AdminOnly")).Succeeded

};

return Json(result);

}

}

**Представение информации**

Home/Index.cshtml

@using Microsoft.AspNetCore.Authorization

@inject IAuthorizationService AuthorizationService

@using System.Security.Claims

<h2>Главная страница</h2>

<div style="background: #fff3cd; padding: 15px; margin: 10px 0; border: 1px solid #ffeaa7;">

<h4>🔍 ДИАГНОСТИКА АУТЕНТИФИКАЦИИ:</h4>

<p><strong>IsAuthenticated:</strong> <span style="color: @(User.Identity.IsAuthenticated ? "green" : "red")">@User.Identity.IsAuthenticated</span></p>

<p><strong>UserName:</strong> @(User.Identity.Name ?? "null")</p>

<p><strong>AuthenticationType:</strong> @(User.Identity.AuthenticationType ?? "null")</p>

<p><strong>Role:</strong> @(User.FindFirst(ClaimTypes.Role)?.Value ?? "null")</p>

<p><strong>Cookies Count:</strong> @Context.Request.Cookies.Count</p>

<div style="margin-top: 10px;">

<a href="/Account/DebugAuth" target="\_blank" class="btn btn-info btn-sm">Подробная диагностика</a>

<form method="post" action="/Account/TestLogin" style="display: inline;">

@Html.AntiForgeryToken()

<button type="submit" class="btn btn-warning btn-sm">Тестовый вход</button>

</form>

</div>

</div>

@if (User.Identity.IsAuthenticated)

{

<div style="background: #d4edda; padding: 20px; margin: 15px 0; border: 2px solid #c3e6cb;">

<h3>✅ ВЫ АВТОРИЗОВАНЫ!</h3>

<p><strong>Имя:</strong> @User.Identity.Name</p>

<p><strong>Роль:</strong> @User.FindFirst(ClaimTypes.Role)?.Value</p>

<div style="margin-top: 15px;">

<a href="/Home/Profile" class="btn btn-primary">Профиль</a>

@if (User.IsInRole("Admin"))

{

<a href="/Home/AdminPanel" class="btn btn-danger">Админ панель</a>

}

<form method="post" action="/Account/Logout" style="display: inline;">

@Html.AntiForgeryToken()

<button type="submit" class="btn btn-warning">Выйти</button>

</form>

</div>

</div>

}

else

{

<div style="background: #f8d7da; padding: 20px; margin: 15px 0; border: 2px solid #f5c6cb;">

<h3>❌ ВЫ НЕ АВТОРИЗОВАНЫ</h3>

<p>Для доступа к функциям системы необходимо войти.</p>

<a href="/Account/Login" class="btn btn-success">Войти в систему</a>

</div>

}

Home/Profile.cshtml

@using System.Security.Claims

<h2>Профиль пользователя</h2>

<p>Эта страница доступна только авторизованным пользователям.</p>

<p><strong>Имя:</strong> @User.Identity.Name</p>

<p><strong>Роль:</strong> @User.FindFirst(ClaimTypes.Role)?.Value</p>

<a href="/">На главную</a>

Home/AdminPanel.cshtml

<h2>Админ панель</h2>

<div style="background: #ffebee; padding: 15px;">

<p>✅ Эта страница доступна только администраторам!</p>

<p>Вы имеете доступ к админ функциям.</p>

</div>

<a href="/">На главную</a>

**Account/AccessDenied.cshtml**

<h2>Доступ запрещен</h2>

<div style="background: #fff3cd; padding: 15px;">

<p>❌ У вас недостаточно прав для просмотра этой страницы.</p>

<p>Требуемая роль: Администратор</p>

</div>

<a href="/">На главную</a>

Account/Login.cshtml

@model LoginModel

<h2>Вход в систему</h2>

<form method="post">

@Html.AntiForgeryToken()

<div style="margin-bottom: 15px;">

<label for="Username">Имя пользователя:</label>

<input type="text" id="Username" name="Username" value="@Model?.Username" class="form-control" style="width: 200px;" />

<span asp-validation-for="Username" class="text-danger"></span>

</div>

<div style="margin-bottom: 15px;">

<label for="Password">Пароль:</label>

<input type="password" id="Password" name="Password" class="form-control" style="width: 200px;" />

<span asp-validation-for="Password" class="text-danger"></span>

</div>

<button type="submit" class="btn btn-primary">Войти</button>

</form>

@if (!ViewData.ModelState.IsValid)

{

<div style="color: red; margin-top: 15px;">

<h4>Ошибки валидации:</h4>

@foreach (var error in ViewData.ModelState.Values.SelectMany(v => v.Errors))

{

<p>@error.ErrorMessage</p>

}

</div>

}

<div style="margin-top: 20px; padding: 10px; background: #f5f5f5;">

<p><strong>Тестовые пользователи:</strong></p>

<p>👤 admin / admin123 (Администратор)</p>

<p>👤 user / user123 (Пользователь)</p>

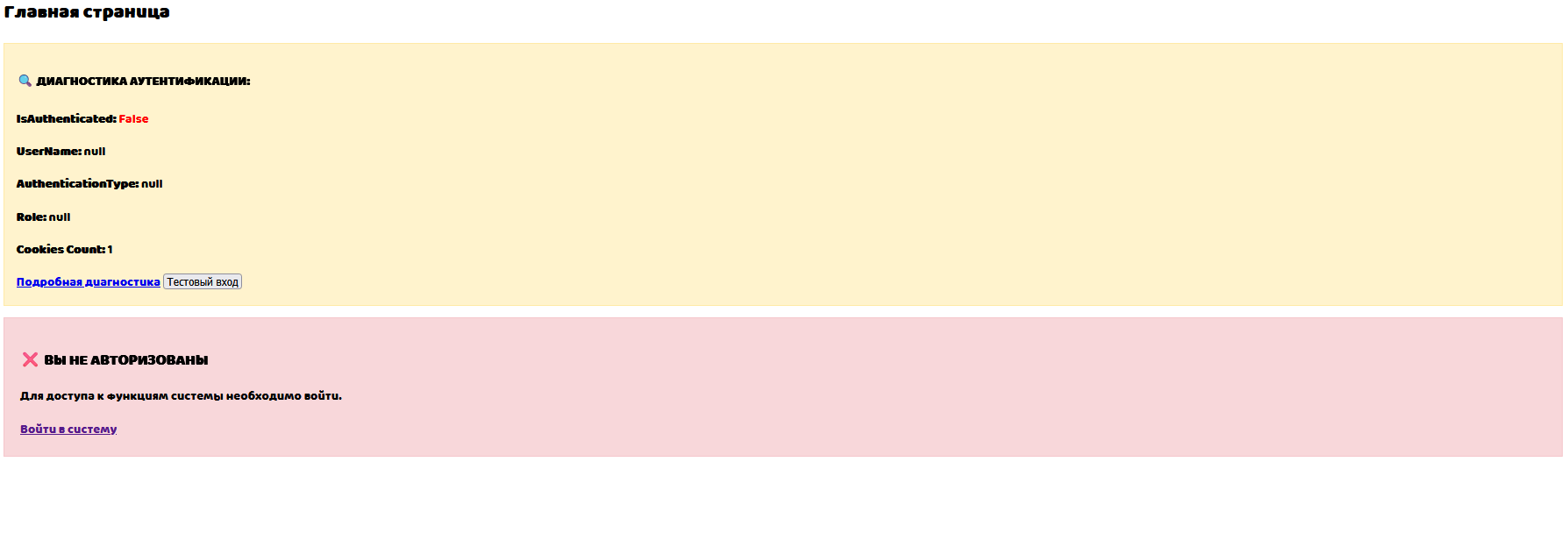
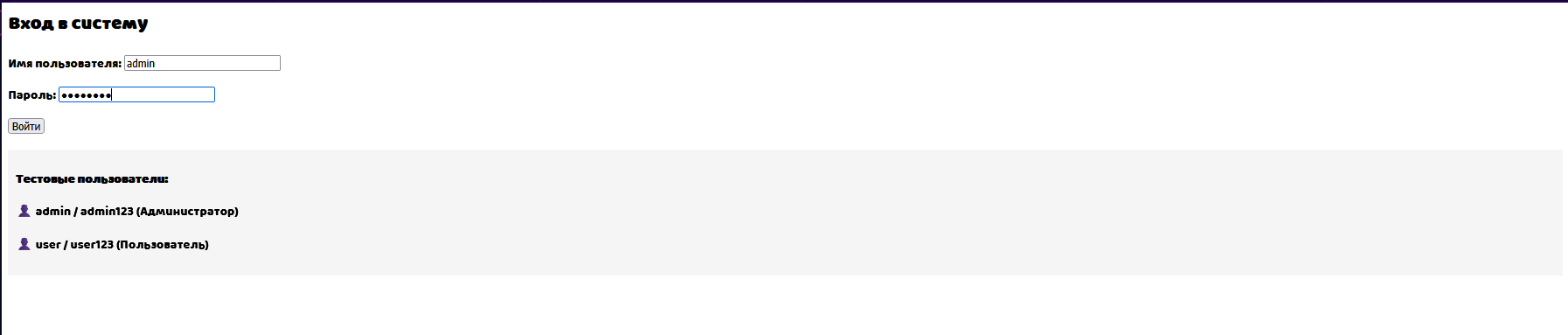
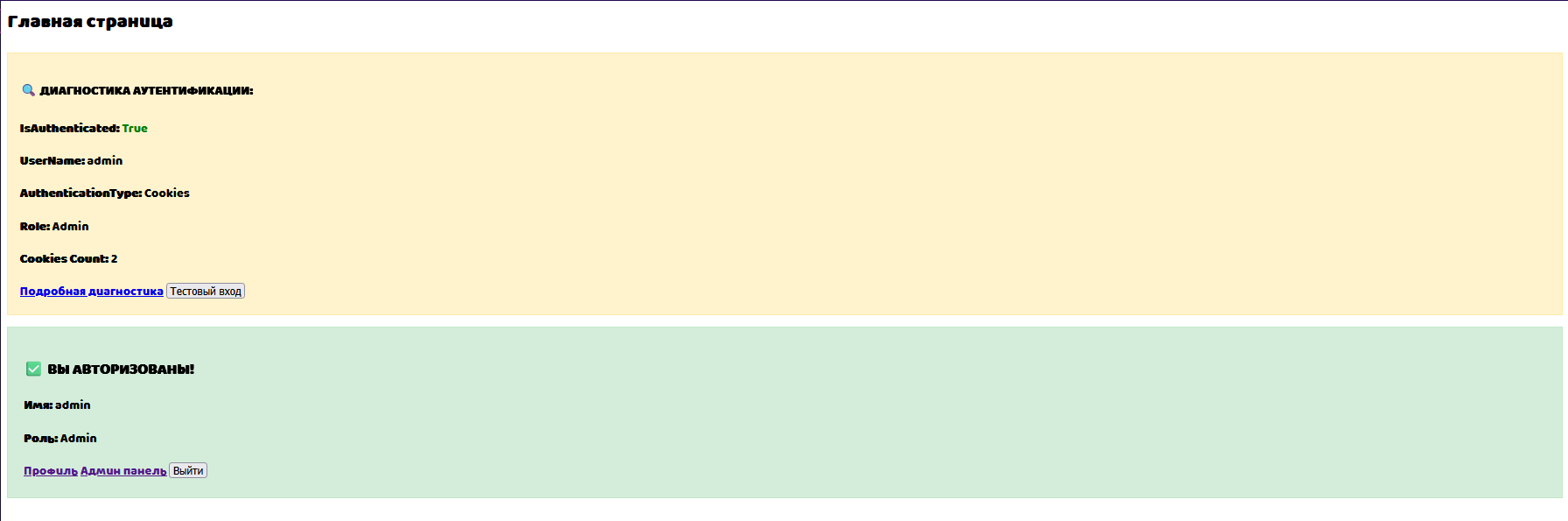
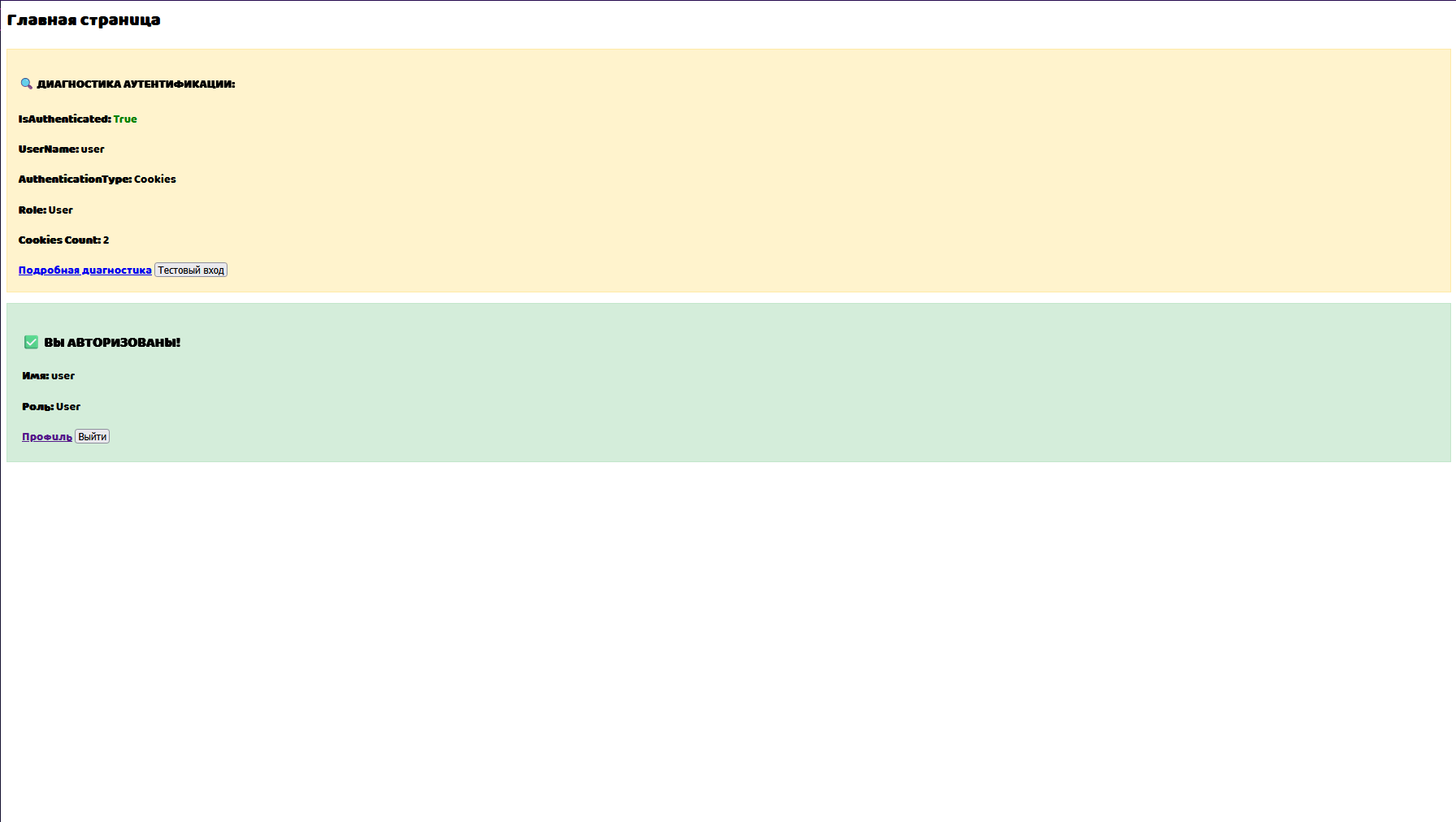
</div>

@section Scripts {

<partial name="\_ValidationScriptsPartial" />

}

## **Тестирование системы**

### Сценарии тестирования:

1. **Анонимный пользователь**:
   * Может просматривать главную страницу
   * Не может получить доступ к защищенным разделам
   * При попытке доступа к защищенной странице перенаправляется на страницу входа
2. **Обычный пользователь (роль User)**:
   * Может входить в систему и выходить из нее
   * Имеет доступ к пользовательским разделам
   * Не может получить доступ к панелям администратора и модератора
3. **Администратор (роль Admin)**:
   * Имеет полный доступ ко всем разделам системы
   * Может управлять пользователями и настройками