Министерство науки и высшего образования Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

**(ВлГУ)**

**Кафедра информационных систем и программной инженерии**

Лабораторная работа №6

по дисциплине «Технологии программирования»

Тема: «Реализация аутентификации в MVC с ASP.NET»

Выполнил:

студент гр. ПРИ-123

А.Ц. Нямаа

Принял:

Ст. пр. кафедры ИСПИ

В.В. Данилов

Владимир, 2025 г.

ЦЕЛЬ РАБОТЫ:

Познакомиться с реализацией аутентификации MVC на платформе ASP.NET.

ЗАДАНИЕ (вариант 12).

В созданном проекте MVC реализовать аутентификацию согласно следующим требованиями: создать как минимум две роли, создать пользователя для каждой роли и организовать ролям и пользователям доступ к определённым действиям на сайте.

ВЫПОЛНЕНИЕ РАБОТЫ:

В начале выполнения данной работы я создал новый .NET проект с MVC шаблоном, где установил тип проверки подлинности «Индивидуальные учетные записи».

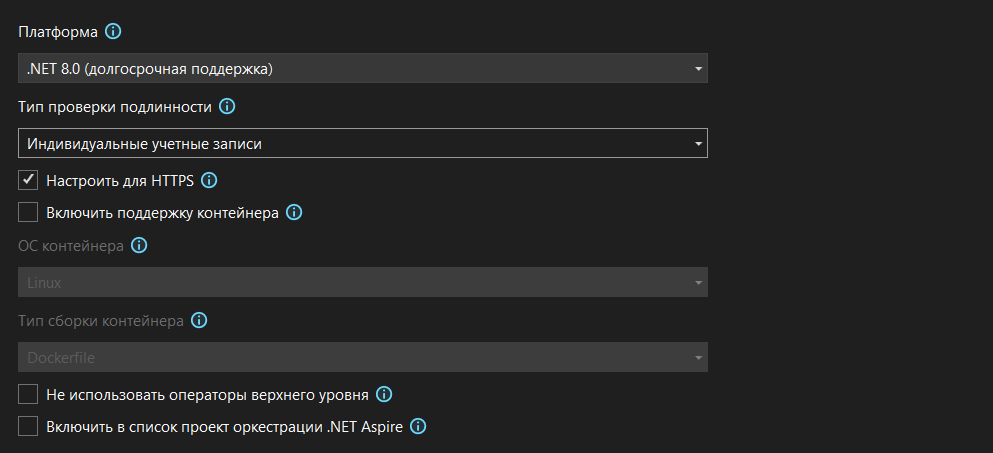


Рисунок 1. Создание нового проекта.

Далее перенес весь код из предыдущей лабораторной работы, а также добавил некоторый новый, который необходим для полноценной работы веб-приложения с пользователями и их ролями.

Листинг Program.cs:

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using task.Data;

using task.Models;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection") ?? throw new InvalidOperationException("Connection string 'DefaultConnection' not found.");

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(connectionString));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddIdentity<User, IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>()

;

builder.Services.AddControllersWithViews();

builder.Services.AddRazorPages();

builder.Services.AddAuthentication("Cookies")

.AddCookie("Cookies", options =>

{

options.LoginPath = "/Account/Login";

options.AccessDeniedPath = "/Account/AccessDenied";

});

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseMigrationsEndPoint();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.MapRazorPages();

app.Run();

Листинг ApplicationDbContext.cs:

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using task.Models;

namespace task.Data

{

public class ApplicationDbContext : IdentityDbContext<User>

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

Database.EnsureCreated();

}

}

}

Листинг AccountController.cs:

using Microsoft.AspNetCore.Mvc;

using CustomIdentityApp.ViewModels;

using Microsoft.AspNetCore.Identity;

using task.Models;

namespace CustomIdentityApp.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

public AccountController(UserManager<User> userManager, SignInManager<User> signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

[HttpGet]

public IActionResult Register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new User { Email = model.Email, UserName = model.Email, Year = model.Year };

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

await \_signInManager.SignInAsync(user, false);

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

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

[HttpGet]

public IActionResult Login(string returnUrl = null)

{

return View(new LoginViewModel { ReturnUrl = returnUrl });

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result =

await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.ReturnUrl))

{

return Redirect(model.ReturnUrl);

}

else

{

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

}

}

else

{

ModelState.AddModelError("", "Неправильный логин и (или) пароль");

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

await \_signInManager.SignOutAsync();

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

}

public IActionResult AccessDenied()

{

return View();

}

}

}

Листинг deleteController.cs (пример организации доступа к определенным действиям по ролям):

using Microsoft.AspNetCore.Identity;

namespace task\_beta.Controllers

{

[Authorize(Roles = "Admin")]

public class deleteController : Controller

{

private string connection;

...

В конце добавил два пользователя при помощи кнопки «Register», создал роль для каждого пользователя через базу данных, а также привязал роли и пользователей, добавив необходимые Id в таблице AspNetUserRoles.

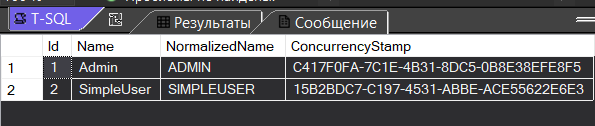


Рисунок 2. Таблица с добавленными ролями.

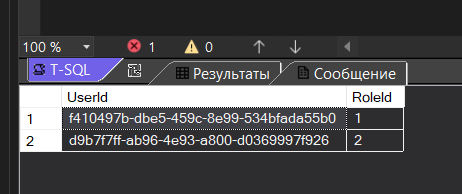


Рисунок 3. Таблица AspNetUserRoles.

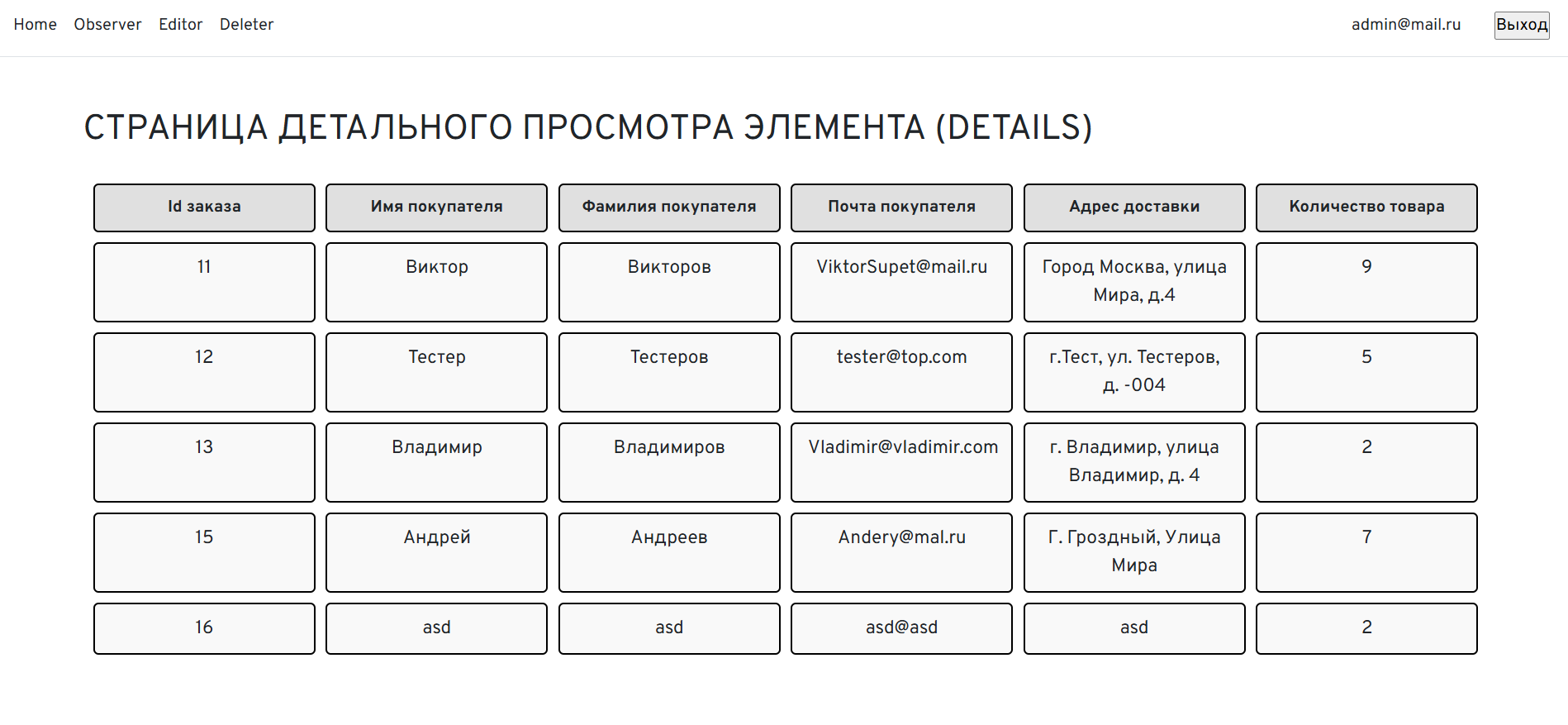


Рисунок 4. Демонстрация доступа к странице Deleter с ролью “Admin”.

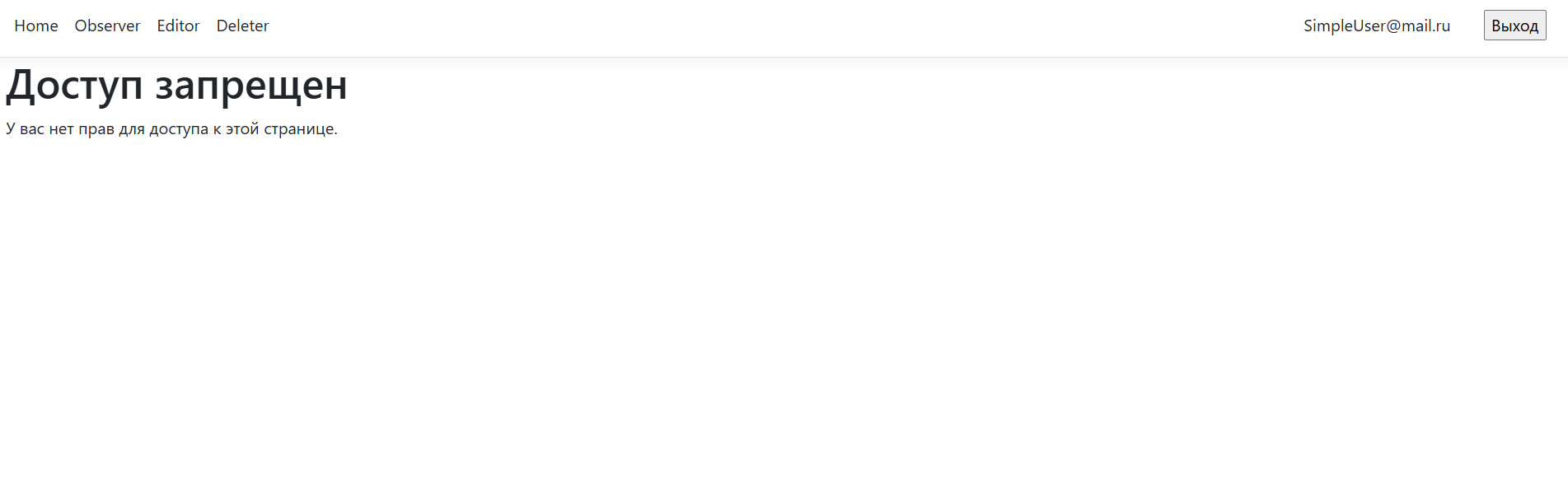


Рисунок 5. Демонстрация запрета на доступ с ролью “SimpleUser”.

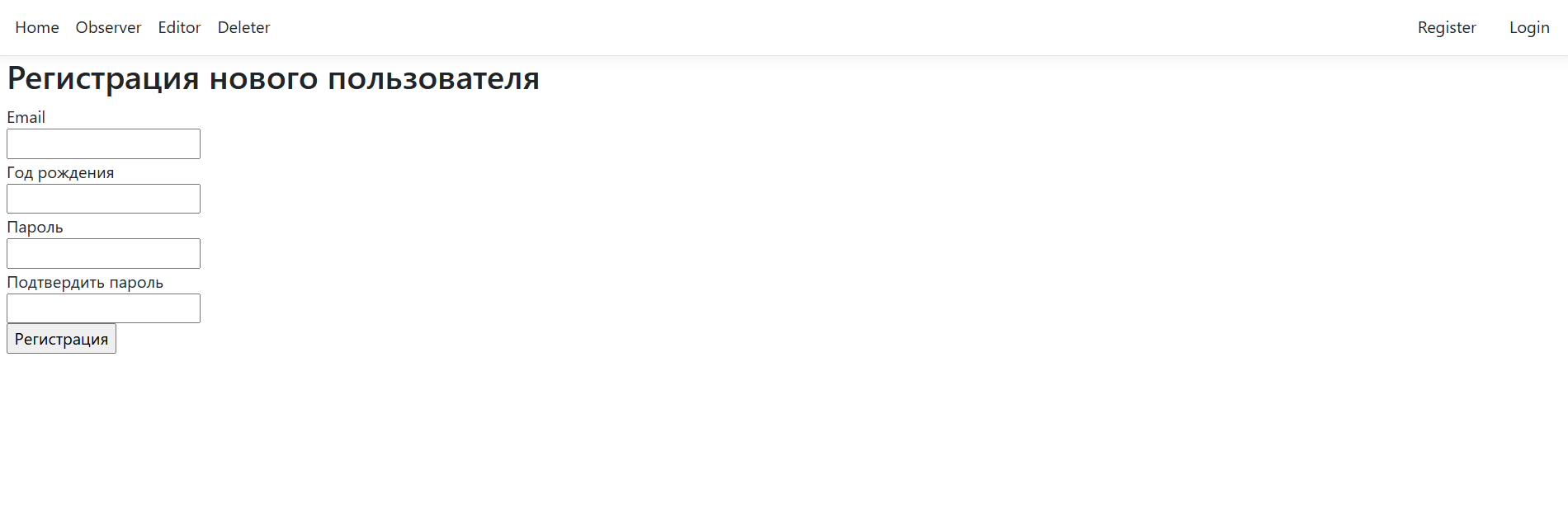


Рисунок 6. Страница регистрации.

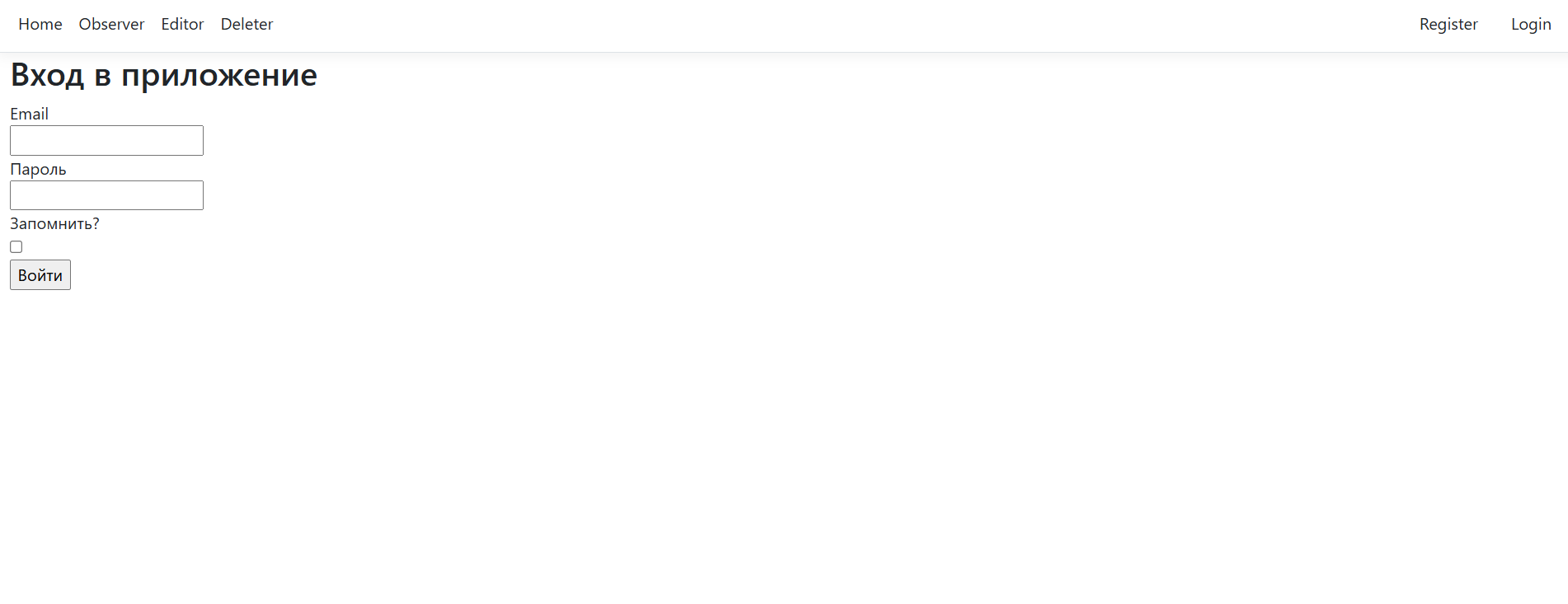


Рисунок 7. Страница авторизации.

ВЫВОД К РАБОТЕ:

Я познакомился с реализацией аутентификации MVC на платформе ASP.NET.