Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

ЗВІТ

з лабораторної роботи № 2

Тема: “Автоматизований інтернет магазин”

Дисципліна «Об’єктно-орієнтоване конструювання програм»

Підготував:

студент гр. ІПЗ-23(1)

Гоша Давід Олександрович

Дата : 30.01.22

Перевірила:

доц. Зубик Л.В.

# Київ – 2022

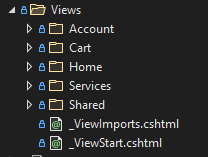
**Тема:** MVC

**Завдання**

1. Створити MVC ASP-застосунок. Для теми проекту з попереднього семестру (або іншої самостійно обраної теми, яка буде закріплена за студентом до кінця поточного семестру після її погодження з викладачами) розробити його модель, подання і контролер(-и)

**Код**

**Представлення**



<!DOCTYPE HTML>

<!--

Future Imperfect by HTML5 UP

html5up.net | ajlkn

Free for personal and commercial use under the CCA 3.0 license (html5up.net/license)

-->

<html>

<head>

@await Html.PartialAsync("MetaTagsPartial")

@await Html.PartialAsync("CssPartial")

</head>

<body class="is-preload">

<!-- Wrapper -->

<div id="wrapper">

<!-- Header -->

@await Html.PartialAsync("HeaderPartial")

<!-- Menu -->

@await Html.PartialAsync("MenuPartial")

<!-- Main -->

<div id="main">

<!-- Post -->

@\*@await Html.PartialAsync("PostSample")\*@

@RenderBody()

<!-- Pagination -->

@\*<ul class="actions pagination">

<li><a href="" class="disabled button large previous">Previous Page</a></li>

<li><a href="#" class="button large next">Next Page</a></li>

</ul>\*@

</div>

<!-- Sidebar -->

@await Html.PartialAsync("SideBarPartial")

@\*@await Html.PartialAsync("FooterPartial")\*@

</div>

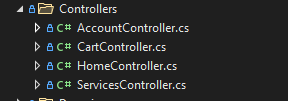
<!-- Scripts -->

@await Html.PartialAsync("ScriptsPartial")

</body>

</html>

**Контроллери**



using System.Threading.Tasks;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using ThirdShop.Models;

using ThirdShop.Services;

using ThirdShop.Models.ViewModels;

namespace ThirdShop.Controllers

{

[Authorize]

public class AccountController : Controller

{

private readonly UserManager<IdentityUser> userManager;

private readonly SignInManager<IdentityUser> signInManager;

public AccountController(UserManager<IdentityUser> userMgr, SignInManager<IdentityUser> signinMgr)

{

userManager = userMgr;

signInManager = signinMgr;

}

[AllowAnonymous]

public IActionResult Login(string returnUrl)

{

ViewBag.returnUrl = returnUrl;

return View(new LoginViewModel());

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl)

{

if (ModelState.IsValid)

{

IdentityUser user = await userManager.FindByNameAsync(model.UserName);

if (user != null)

{

await signInManager.SignOutAsync();

Microsoft.AspNetCore.Identity.SignInResult result = await signInManager.PasswordSignInAsync(user, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

return Redirect(returnUrl ?? "/");

}

}

ModelState.AddModelError(nameof(LoginViewModel.UserName), "Неверный логин или пароль");

}

return View(model);

}

[Authorize]

public async Task<IActionResult> Logout()

{

await signInManager.SignOutAsync();

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

}

[HttpGet]

[AllowAnonymous]

public IActionResult Register()

{

return View();

}

[HttpPost]

[AllowAnonymous]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

IdentityUser user = new IdentityUser { Email = model.Email, UserName = model.Name, };

// добавляем пользователя

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

if (result.Succeeded)

{

// генерация токена для пользователя

var code = await userManager.GenerateEmailConfirmationTokenAsync(user);

var callbackUrl = Url.Action(

"ConfirmEmail",

"Account",

new { userId = user.Id, code = code },

protocol: HttpContext.Request.Scheme);

EmailService emailService = new EmailService();

await emailService.SendEmailAsync(model.Email, "Confirm your account",

$"Подтвердите регистрацию, перейдя по ссылке: <a href='{callbackUrl}'>link</a>");

return View("MailCheck");

}

else

{

foreach (var error in result.Errors)

{

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

}

}

}

return View(model);

}

[HttpGet]

[AllowAnonymous]

public IActionResult ForgotPassword()

{

return View();

}

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ForgotPassword(ForgotPasswordViewModel model)

{

if (ModelState.IsValid)

{

var user = await userManager.FindByEmailAsync(model.Email);

if (user == null || !(await userManager.IsEmailConfirmedAsync(user)))

{

// пользователь с данным email может отсутствовать в бд

// тем не менее мы выводим стандартное сообщение, чтобы скрыть

// наличие или отсутствие пользователя в бд

return View("ForgotPasswordConfirmation");

}

/\*Для сброса пароля пройдите по ссылке\*/

var code = await userManager.GeneratePasswordResetTokenAsync(user);

var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code }, protocol: HttpContext.Request.Scheme);

EmailService emailService = new EmailService();

await emailService.SendEmailAsync(model.Email, "Reset Password",

$"For reset password use link: <a href='{callbackUrl}'>link</a>");

return View("ForgotPasswordConfirmation");

}

return View(model);

}

[HttpGet]

[AllowAnonymous]

public IActionResult ResetPassword(string code = null)

{

return code == null ? View("Error") : View();

}

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ResetPassword(ResetPasswordViewModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

var user = await userManager.FindByEmailAsync(model.Email);

if (user == null)

{

return View("ResetPasswordConfirmation");

}

var result = await userManager.ResetPasswordAsync(user, model.Code, model.Password);

if (result.Succeeded)

{

return View("ResetPasswordConfirmation");

}

foreach (var error in result.Errors)

{

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

}

return View(model);

}

[HttpGet]

[AllowAnonymous]

public async Task<IActionResult> ConfirmEmail(string userId, string code)

{

if (userId == null || code == null)

{

return View("Error");

}

var user = await userManager.FindByIdAsync(userId);

if (user == null)

{

return View("Error");

}

var result = await userManager.ConfirmEmailAsync(user, code);

if (result.Succeeded)

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

else

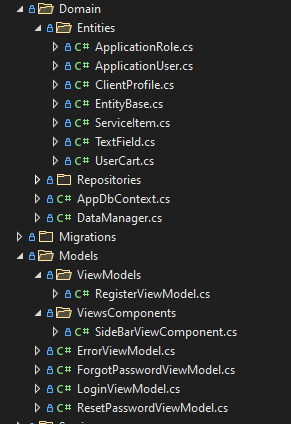
return View("Error");

}

}

}

Моделі



**Контекст бази даних**

using System;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using ThirdShop.Domain.Entities;

namespace ThirdShop.Domain

{

public class AppDbContext : IdentityDbContext<IdentityUser>

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<TextField> TextFields { get; set; }

public DbSet<ServiceItem> ServiceItems { get; set; }

public DbSet<UserCart> UserCarts { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<IdentityRole>().HasData(new IdentityRole

{

Id = "44546e06-8719-4ad8-b88a-f271ae9d6eab",

Name = "admin",

NormalizedName = "ADMIN"

});

modelBuilder.Entity<IdentityUser>().HasData(new IdentityUser

{

Id = "3b62472e-4f66-49fa-a20f-e7685b9565d8",

UserName = "admin",

NormalizedUserName = "ADMIN",

Email = "my@email.com",

NormalizedEmail = "MY@EMAIL.COM",

EmailConfirmed = true,

PasswordHash = new PasswordHasher<IdentityUser>().HashPassword(null, "admin"),

SecurityStamp = string.Empty

});

modelBuilder.Entity<IdentityUserRole<string>>().HasData(new IdentityUserRole<string>

{

RoleId = "44546e06-8719-4ad8-b88a-f271ae9d6eab",

UserId = "3b62472e-4f66-49fa-a20f-e7685b9565d8"

});

modelBuilder.Entity<TextField>().HasData(new TextField

{

Id = new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"),

CodeWord = "PageIndex",

Title = "Главная"

});

modelBuilder.Entity<TextField>().HasData(new TextField

{

Id = new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"),

CodeWord = "PageServices",

Title = "Наши услуги"

});

modelBuilder.Entity<TextField>().HasData(new TextField

{

Id = new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"),

CodeWord = "PageContacts",

Title = "Контакты"

});

modelBuilder.Entity<TextField>().HasData(new TextField

{

Id = new Guid("c8353f35-6dc4-4bb7-b447-279e6bbb9ea6"),

CodeWord = "PageAboutUs",

Title = "О нас"

});

}

}

}

**Приклад моделі тектових полів**

using System.ComponentModel.DataAnnotations;

namespace ThirdShop.Domain.Entities

{

public class TextField : EntityBase

{

[Required]

public string CodeWord { get; set; }

[Display(Name = "Название страницы (заголовок)")]

public override string Title { get; set; } = "Информационная страница";

[Display(Name = "Cодержание страницы")]

public override string Text { get; set; } = "Содержание заполняется администратором";

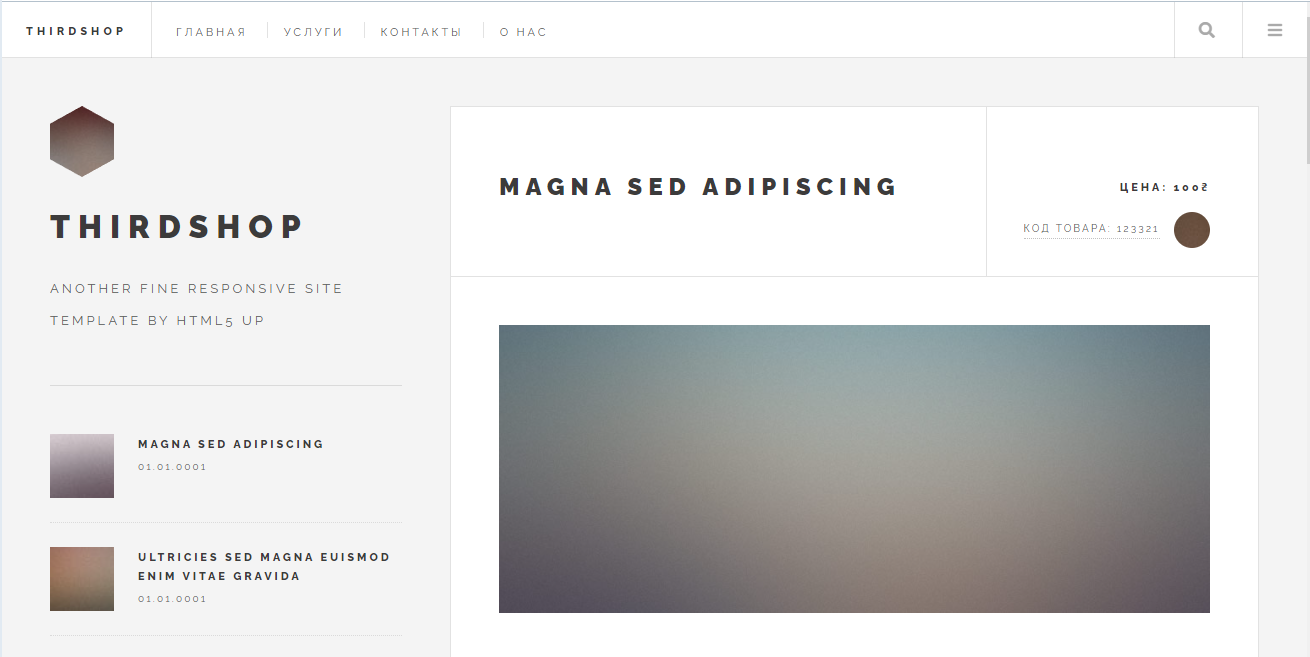
}

}

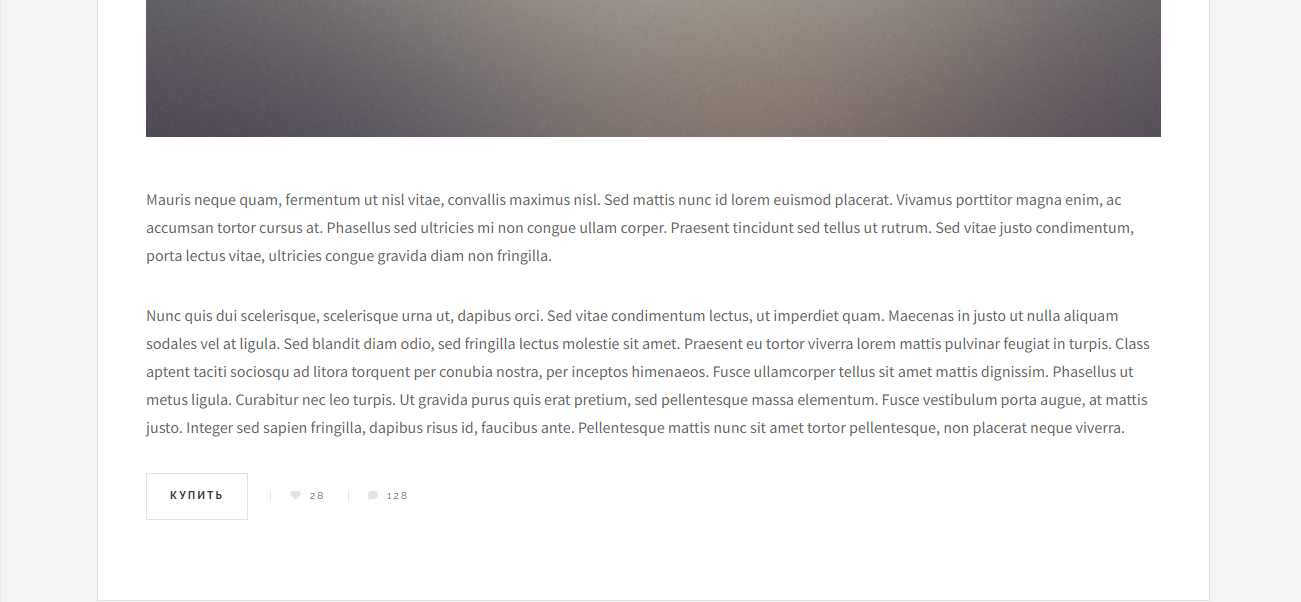
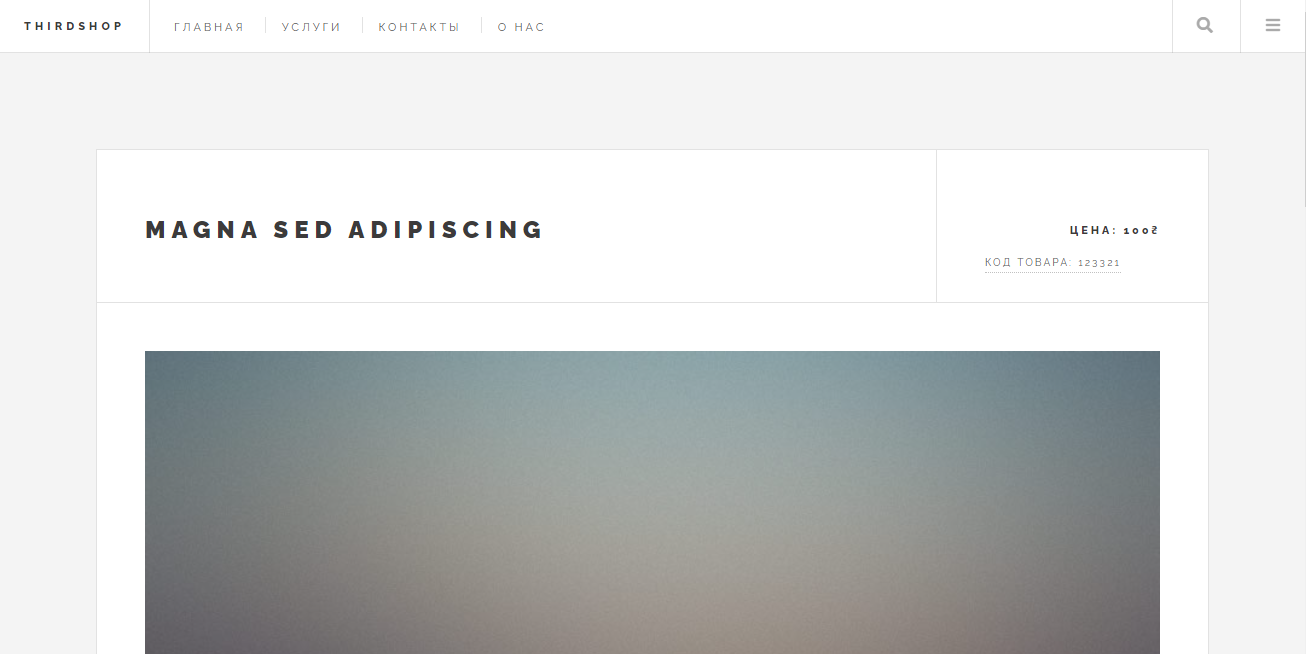
**Скриншоти Виконання**



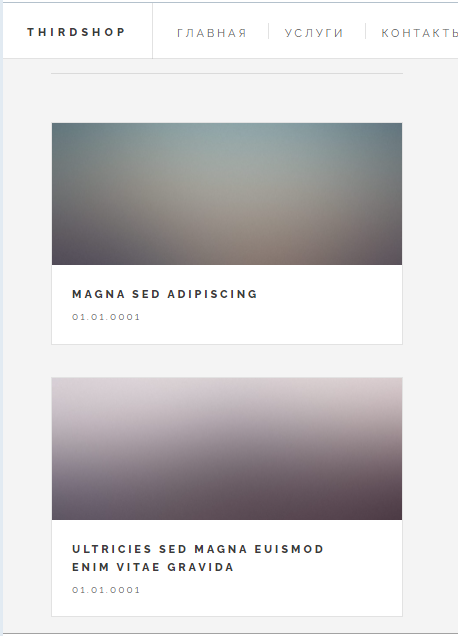
Приклад сторінки послуг



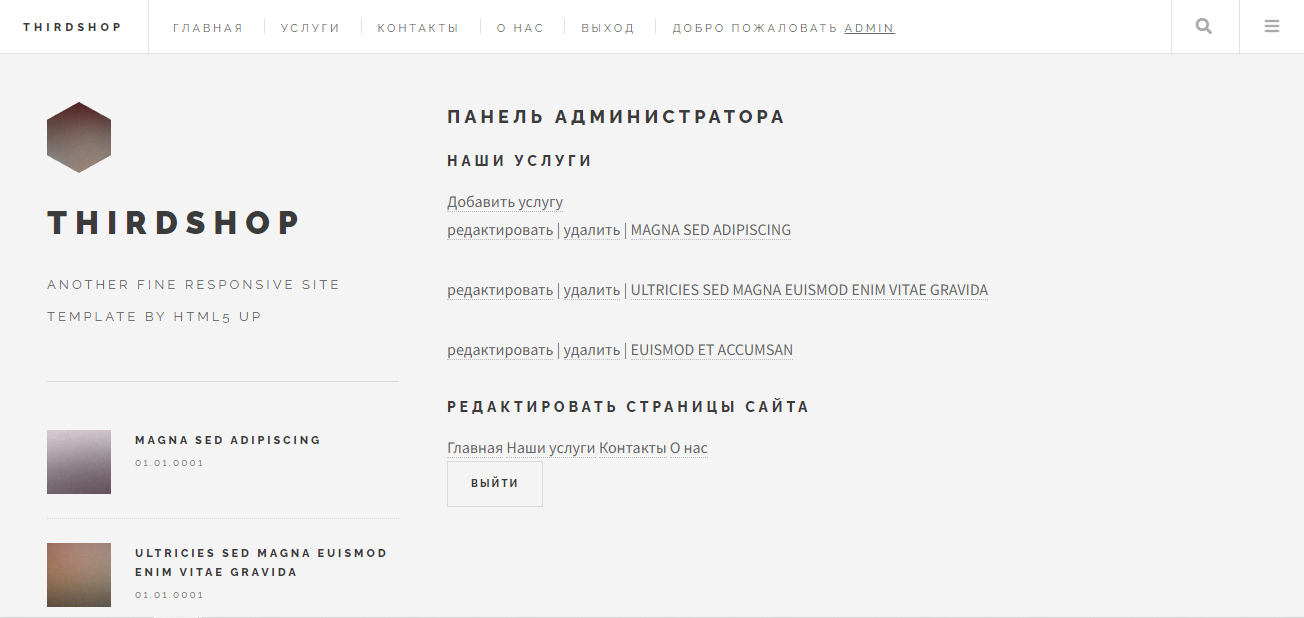
Приклад самої послуги



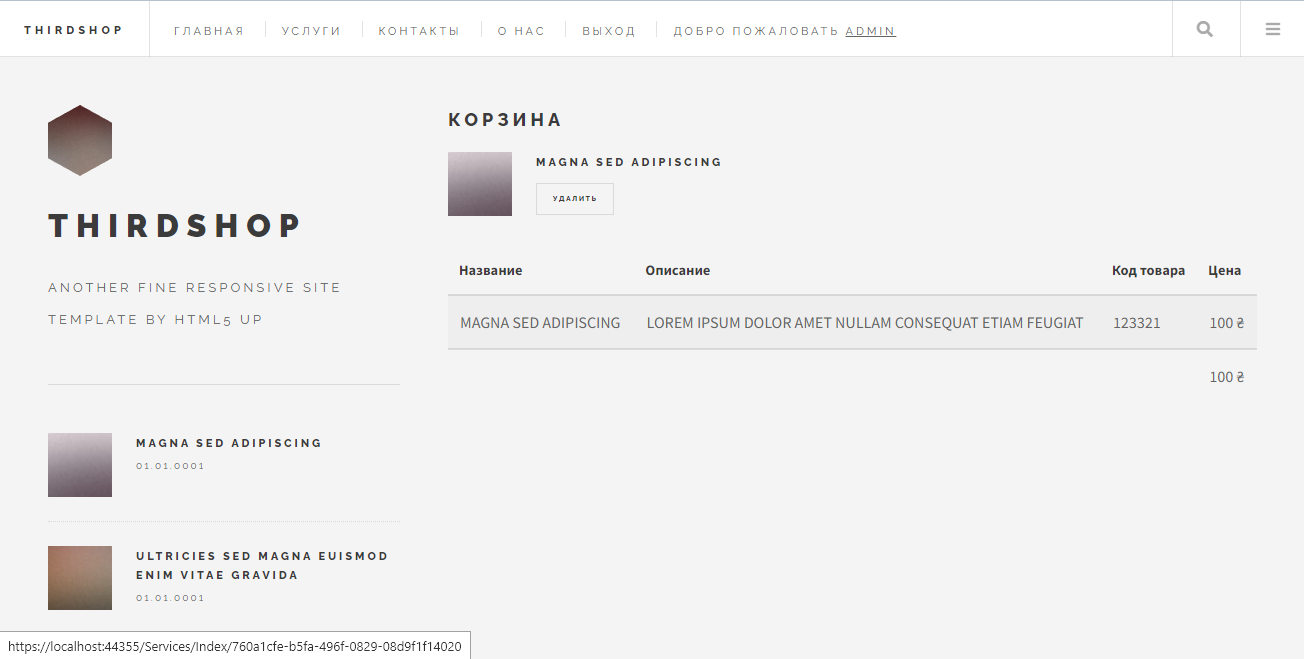
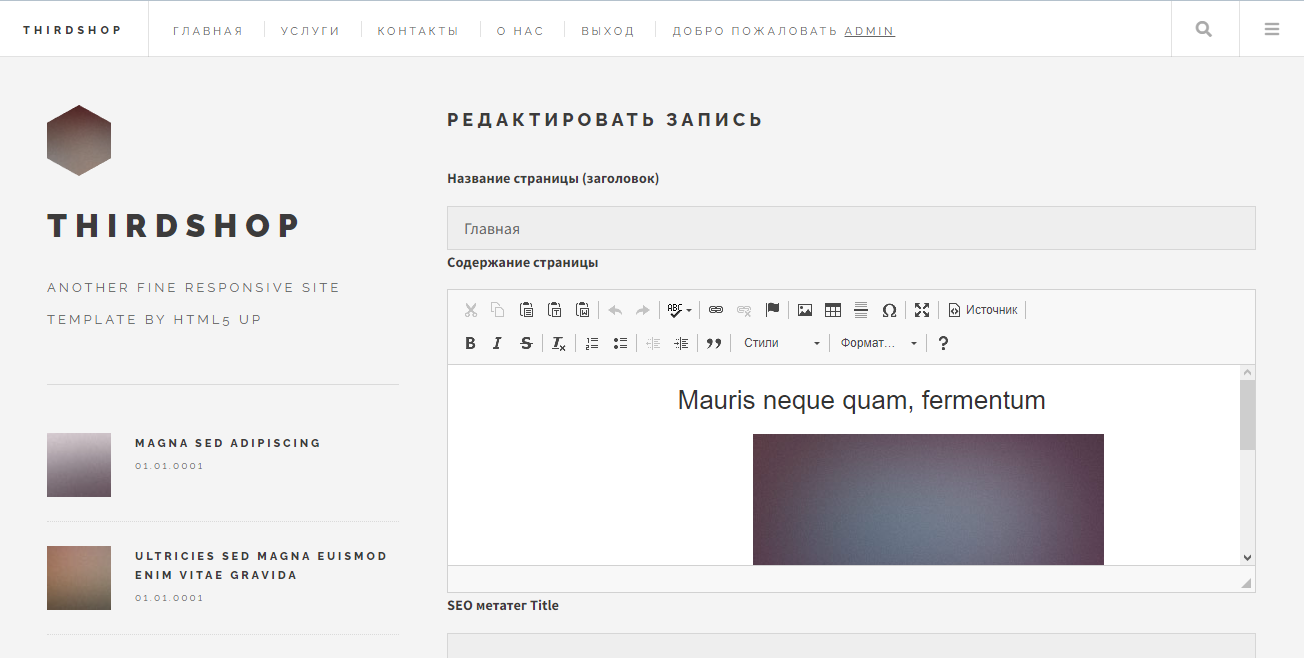
Приклад сайдбара



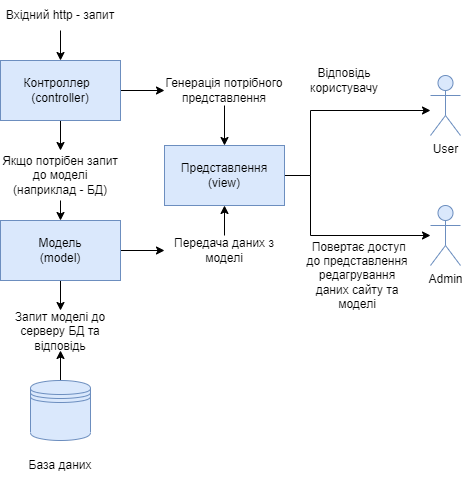
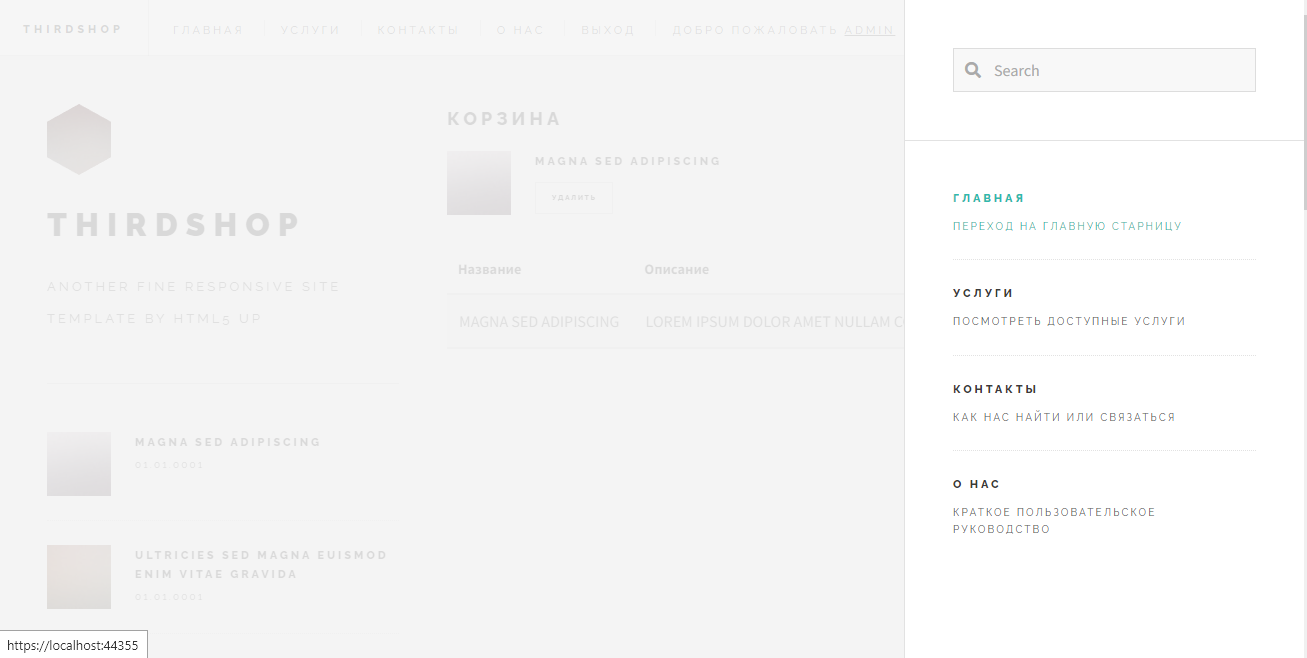
Адмін панель



Приклад редагування головної сторінки



Приклад роботи меню



Діаграма роботи компонентів

Висновки:

У цій лабораторній я розробив веб застосунок на платформі asp.net core за паттерном MVC. Було створено прототип інтернет магазину, який динамічно заповнюється адміністраторами. Додано ViewComponent у вигляді сайдбару.

Технології які використовувались:

1. Середовище розробки Visual Studio 2022
2. Сервер бд MS SQL Server 2019
3. Мова прогрумування C#
4. Entity Framework Core + Migrations + Identity
5. Тип застосунку ASP.NET Core MVC 3.1
6. HTML5 + SASS (CSS)
7. JavaScript + jQuery