

Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и информационных технологий
Кафедра 805

Курсовой проект

По дисциплине «ООП»

Вариант №3: «Разработка сложных сайтов с использованием
технологии ASP.NET Core MVC»

Студент: Манташев Асадулла Уллубиевич

Группа: М8О-205Б-20

Руководитель: доц. Кузнецова С. В.

Оценка:

Дата:

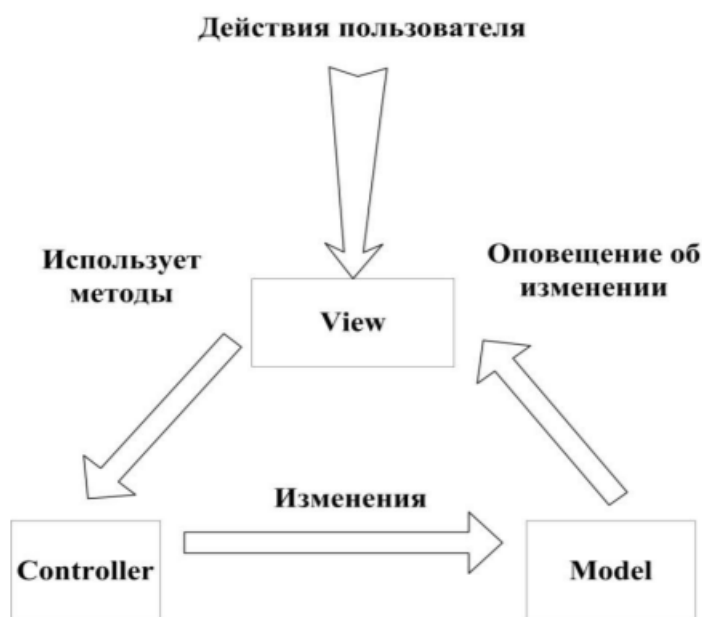
Цель: разработать сайт, использующий технологию ASP.Net MVC. В качестве примера я разработаю сайт онлайн магазин по продаже окон.

Описание технологии Asp.Net MVC

Model-view-controller (MVC, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких паттернов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данная схема проектирования часто используется для построения архитектурного каркаса, когда переходят от теории к реализации в конкретной предметной области.

Основная цель применения этой концепции состоит в отделении бизнес-логики (*модели*) от её визуализации (*представления, вида*). За счет такого разделения повышается возможность повторного использования.

На следующей схеме показаны три основных компонента и существующие между ними связи.



Структура MVC

Такое распределение обязанностей позволяет масштабировать приложение в контексте сложности, так как проще писать код, выполнять отладку и тестирование компонента (модели, представления или контроллера) с одним заданием. Гораздо труднее обновлять, тестировать и отлаживать код, зависимости которого находятся в двух или трех этих областях. Например, логика пользовательского интерфейса, как правило, подвергается изменениям чаще, чем бизнес-логика. Если код представления и бизнес-логика объединены в один объект, содержащий бизнес-логику, объект

необходимо изменять при каждом обновлении пользовательского интерфейса. Это часто приводит к возникновению ошибок и необходимости повторно тестировать бизнес-логику после каждого незначительного изменения пользовательского интерфейса.

Функции модели

Модель в приложении MVC представляет состояние приложения и бизнес-логику или операций, которые должны в нем выполняться. Бизнес-логика должна быть включена в состав модели вместе с логикой реализации для сохранения состояния приложения. Как правило, строго типизированные представления используют типы `ViewModel`, предназначенные для хранения данных, отображаемых в этом представлении. Контроллер создает и заполняет эти экземпляры `ViewModel` из модели.

Функции представления

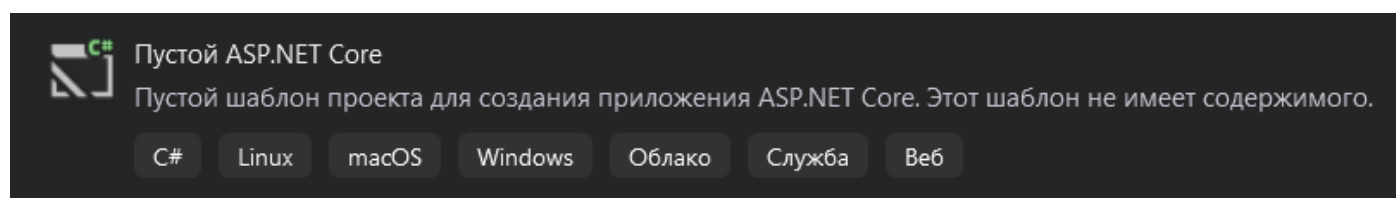
Представления отвечают за представление содержимого через пользовательский интерфейс. Они используют `Razor` обработчик представлений для внедрения кода `.NET` в разметку `HTML`. Представления должны иметь минимальную логику, которая должна быть связана с представлением содержимого. Если есть необходимость выполнять большую часть логики в представлении для отображения данных из сложной модели, рекомендуется воспользоваться компонентом представления, `ViewModel` или шаблоном представления, позволяющими упростить представление.

Функции контроллера

Контроллеры — это компоненты для управления взаимодействием с пользователем, работы с моделью и выбора представления для отображения. В приложении MVC представление служит только для отображения информации. Обработку введенных данных, формирование ответа и взаимодействие с пользователем обеспечивает контроллер. В структуре MVC контроллер является начальной отправной точкой и отвечает за выбор рабочих типов моделей и отображаемых представлений (именно этим объясняется его название — он контролирует, каким образом приложение отвечает на конкретный запрос).

Создание и настройка сайта. Необходимое программное обеспечение

Для создания сайта я сформировал пустой шаблон `ASP.NET Core` в который впоследствии добавлю все нужные папки для реализации технологии MVC



Далее произвожу настройку проекта. С помощью контроллера пакетов NuGet добавляю технологию Microsoft.AspNet.Mvc и Microsoft.AspNetCore.StaticFiles (позволяет отображать на сайте статические файлы, например картинки). Далее подключим все нужные сервисы в файле Startup.cs.

Startup.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;

namespace CourseProject
{
    public class Startup
    {
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddTransient<IAllWindows, MockWindows>(); //позволяет связать
интерфейс с его реализацией
            services.AddTransient<IWindowsCategory, MockCategory>();
            services.AddMvc();
        }

        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            app.UseStatusCodePages();
            app.UseStaticFiles(); // Подключение Microsoft.AspNetCore.StaticFiles
            app.UseMvcWithDefaultRoute(); // Осуществляет переадресацию на главную страницу
        }
    }
}
```

Program.cs

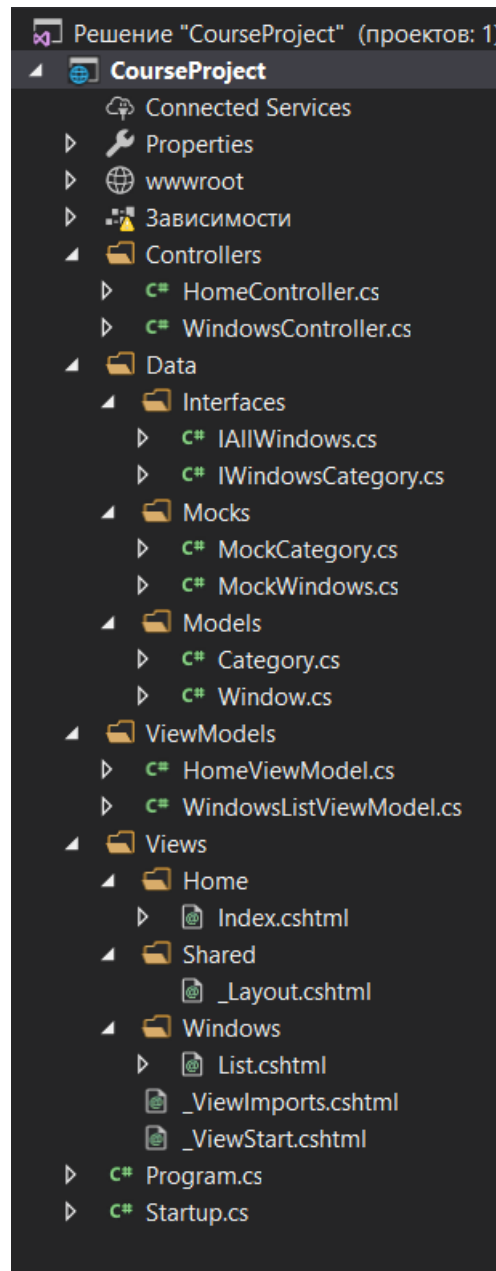
```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace CourseProject
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

Структура проекта web-приложения.

Модели, контроллеры, представления



wwwroot – папка для хранения статических файлов

Controllers – контроллеры

Data – папка хранящая в себе все, что связано с моделью

Interface – интерфейс моделей

Mocks – информация о моделях

Models – модель

ViewModels – модели, использующиеся при отображении информации на сайте

View – все, что связано с представлением

Windows - html шаблон для представления страницы со всеми товарами

Home – html шаблон для представления главной страницы

Shared – html шаблон, содержащий дизайн сайта

Файлы **_ViewImport.cshtml** и **_ViewStart.cshtml** осуществляют связь между контроллером и представлением

Теперь подробнее о содержании других файлов:

WindowsController.cs

```
using CourseProject.Data.Interfaces;
using CourseProject.Data.Models;
using CourseProject.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Controllers
{
    public class WindowsController : Controller
    {
        private readonly IAllWindows _allWindows;
        private readonly IWindowsCategory _allCategories;

        public WindowsController(IAllWindows iAllWindows, IWindowsCategory iWindowsCat)
        {
            _allWindows = iAllWindows;
            _allCategories = iWindowsCat;
        }

        [Route("Windows/List")] //Указание ссылок, которые будет использовать представление
        [Route("Windows/List/{category}")]
        public IActionResult List(string category)
        {
            string _category = category;
            IEnumerable<Window> windows = null;
            string currCategory = "";
            string currDesc = "";
            if (string.IsNullOrEmpty(category))
            {
                windows = _allWindows.Windows.OrderBy(i => i.id); //Запись в переменную
                //всех элементов если category не указана
            }
            else
            {
                //Запись в переменную всех элементов соответствующих указанной category
                if (string.Equals("plastic", category,
                    StringComparison.OrdinalIgnoreCase))
                {
                    windows = _allWindows.Windows.Where(i =>
                        i.Category.categoryName.Equals("Пластиковые окна")).OrderBy(i => i.id);
                    currCategory = "Пластиковые окна";
                    currDesc = "Пластиковые окна отличные конструкции, обеспечивающие
                        качественную звукоизоляцию и предотвращающие потерю тепла.";
                }
                else if (string.Equals("Wood", category,
                    StringComparison.OrdinalIgnoreCase))
                {

```

```

        windows = _allWindows.Windows.Where(i =>
i.Category.categoryName.Equals("Деревянные окна")).OrderBy(i => i.id);
        currCategory = "Деревянные окна";
        currDesc = "Красота и надежность от природы по выгодной цене";
    }
    else if (string.Equals("aluminum", category,
StringComparison.OrdinalIgnoreCase))
    {
        windows = _allWindows.Windows.Where(i =>
i.Category.categoryName.Equals("Алюминиевые окна")).OrderBy(i => i.id);
        currCategory = "Алюминиевые окна";
        currDesc = "Алюминиевые окна – это легкая и надежная конструкция,
в основе которой лежит изготовленный из алюминия профиль с установленным в него стеклопакетом.";
    }
}

var windowObj = new WindowsListViewModel
{
    allWindows = windows,
    currCategory = currCategory,
    currDesc = currDesc
};
ViewBag.Title = "Страница с окнами";

return View(windowObj); //Передача полученного списка окон в html шаблон
}
}
}

```

HomeController.cs

```

using CourseProject.Data.Interfaces;
using CourseProject.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Controllers
{
    public class HomeController : Controller
    {
        //Создание и ввод переменной, хранящей список всех окон, при помощи конструктора
        private IAllWindows _windowRep;

        public HomeController(IAllWindows windowRep)
        {
            _windowRep = windowRep;
        }
        //Вызов html шаблона
        public IActionResult Index()
        {
            var homeWindows = new HomeViewModel //Ввод переменной, хранящей в себе те окна,
у которых isFavourite == true;
            {
                favWindows = _windowRep.Windows.Where(p => p.isFavorite) //Отбор этих
окон и запись в переменную
            };
            return View(homeWindows); //Передача нужных окон в html шаблон
        }
    }
}

```

IAllWindows.cs

```
using CourseProject.Data.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Data.Interfaces
{
    public interface IAllWindows //интерфейс окон
    {
        IEnumerable<Window> Windows { get; }
        IEnumerable<Window> getFavWindows { get; }
        Window getObjectWindow(int windowid);
    }
}
```

IWindowsCategory.cs

```
using CourseProject.Data.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Data.Interfaces
{
    public interface IWindowsCategory //интерфейс категорий
    {
        IEnumerable<Category> AllCategories{ get; }
    }
}
```

MockCategory.cs

```
using CourseProject.Data.Interfaces;
using CourseProject.Data.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Data.Mocks
{
    public class MockCategory : IWindowsCategory //реализация интерфейса категорий
    {
        public IEnumerable<Category> AllCategories
        {
            get //указываем категории, которые буду
            //возвращаться при вызове ф-ции
            {
                return new List<Category>
                {
                    new Category { categoryName = "Пластиковые окна", desc =
                    "Пластиковые окна отличные конструкции, обеспечивающие качественную звукоизоляцию и предотвращающие
                    потерю тепла."},
                    new Category { categoryName = "Деревянные окна", desc = "Красота и
                    надежность от природы по выгодной цене"},
                    new Category { categoryName = "Алюминиевые окна", desc =
                    "Алюминиевые окна – это легкая и надежная конструкция, в основе которой лежит изготовленный из
                    алюминия профиль с установленным в него стеклопакетом."},
                };
            }
        }
    }
}
```


MockWindows.cs

MockWindows.cs

```

        longDesc = "Они эффектно смотрятся в
интерьере и на фасаде, делают вид проема более солидным. Цветное покрытие в оттенках RAL-палитры не
выгорает на солнце.",
        img = "/img/5.jpg",
        price = 18000,
        isFavorite = true,
        Category =

_categoryWindows.AllCategories.First()
    },
    new Window { name = "Одностворчатое окно",
        size = "600x1415 мм",
        longDesc = "Сделано из дуба. Среди всех
пород дерева дуб является бесспорным лидером по показателю прочности, плотности и долговечности.",
        img = "/img/6.jpg",
        price = 35000,
        isFavorite = false,
        Category =

_categoryWindows.AllCategories.ElementAt(1)
    },
    new Window { name = "Раздвижное окно",
        size = "800x1700 мм",
        longDesc = "Небольшой вес. Устанавливается
на любые балконы без риска их деформации.",
        img = "/img/7.jpg",
        price = 5600,
        isFavorite = true,
        Category =

_categoryWindows.AllCategories.Last()
    },
    new Window { name = "Раздвижное из 3 створок",
        size = "2400x1700 мм",
        longDesc = "Защищает от проникновения в дом
пыли, осадков, уличного шума. Раздвижные модели имеют увеличенный световой проем",
        img = "/img/8.jpg",
        price = 14920,
        isFavorite = false,
        Category =

_categoryWindows.AllCategories.Last()
    }
};
}
}
}
public IEnumerable<Window> getFavWindows { get; set; }

public Window getObjectWindow(int windowid)
{
    throw new NotImplementedException();
}
}
}
}

```

Category.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Data.Models
{
    public class Category //модель категории окон
    {
        public int id { set; get; }

        public string categoryName { set; get; }

        public string desc { set; get; }
    }
}

```

```

        public List<Window> Windows { set; get; }
    }
}

```

Window.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.Data.Models
{
    public class Window    //модель окна
    {
        internal bool isFavourite;

        public int id { set; get; }

        public string name { set; get; }

        public string size { set; get; }

        public string longDesc { set; get; }

        public string img { set; get; }

        public int price { set; get; }

        public int categoryID { set; get; }

        public bool available { set; get; }

        public bool isFavorite { set; get; }

        public virtual Category Category { set; get; }
    }
}

```

WindowsListViewModel.cs

```

using CourseProject.Data.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.ViewModels
{
    public class WindowsListViewModel
    {
        // введение функций, возвращающих все окна и текущую категорию с её описанием
        public IEnumerable<Window> allWindows { get; set; }

        public string currCategory { get; set; }

        public string currDesc { get; set; }
    }
}

```

HomeViewModel.cs

```

using CourseProject.Data.Models;

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace CourseProject.ViewModels
{
    //Введение функций, возвращающих все окна, у которых isFavorite==true
    public class HomeViewModel
    {
        public IEnumerable<Window> favWindows { get; set; }
    }
}

```

List.cshtml

```

<div class="row m-3 p-3">
    <h3>@Model.currCategory</h3>
    <h4>@Model.currDesc</h4>
</div>
<div class="row m-3 p-5">
    @{
        foreach (var window in Model.allWindows)
        {
            <div class="col-lg-4">
                
                <h2>@window.name</h2>
                <p><u>Цена:</u> @window.price.ToString("c")</p>
                <p><u>Размер:</u> @window.size</p>
                <p><u>Описание:</u> @window.longDesc</p>
            </div>
        }
    }
</div>
<!--блок кода(характеристика каждого отдельного товара) , который мы вставим в общий шаблон -->

```

Index.cshtml

```

@model HomeViewModel
<p>&nbsp;</p>
<!-- html шаблон для главной страницы-->
<h1>Самые продаваемые варианты</h1>
<div class="row mt-5 mb-2">
    @{
        foreach (var window in Model.favWindows)
        {
            <div class="col-lg-4">
                
                <h2>@window.name</h2>
                <p>Цена: @window.price.ToString("c")</p>
            </div>
        }
    }
</div>

```

_Layout.cshtml

```

<!DOCTYPE html>
<!-- общий шаблон -->
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>@ViewBag.Title</title>
    <link href="~/css/bootstrap.min.css" rel="stylesheet" type="text/css" />
    <link href="~/css/Style.css" rel="stylesheet" type="text/css" />
</head>
<body>

```

```

<header>
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container-fluid">
            <a class="navbar-brand" href="/">Главная</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-
label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarCollapse">
                <ul class="navbar-nav me-auto mb-2 mb-md-0">
                    <li class="nav-item">
                        <a class="nav-link active" aria-current="page"
href="/Windows/List">Все окна</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active"
href="/Windows/List/plastic">Пластиковые окна</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active"
href="/Windows/List/wood">Деревянные окна</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active"
href="/Windows/List/aluminum">Алюминиевые окна</a>
                    </li>
                </ul>
            </div>
        </div>
    </nav>
</header>
<div class="container mt-5">
    @RenderBody()
</div>
<footer class="my-5 pt-5 text-muted text-center text-small">
    <p class="mb-1">2021 Курсовая работа</p>
    <p class="float-right"><a href="#">Back to top</a></p>
</footer>
<script src="~/js/bootstrap.min.js"></script>
</body>
</html>

```

_ViewImports.cshtml

```

@using CourseProject.ViewModels
@using CourseProject.Data.Models
<!-- в этом файле мы подключаем доп файлы -->

```

_ViewStart.cshtml

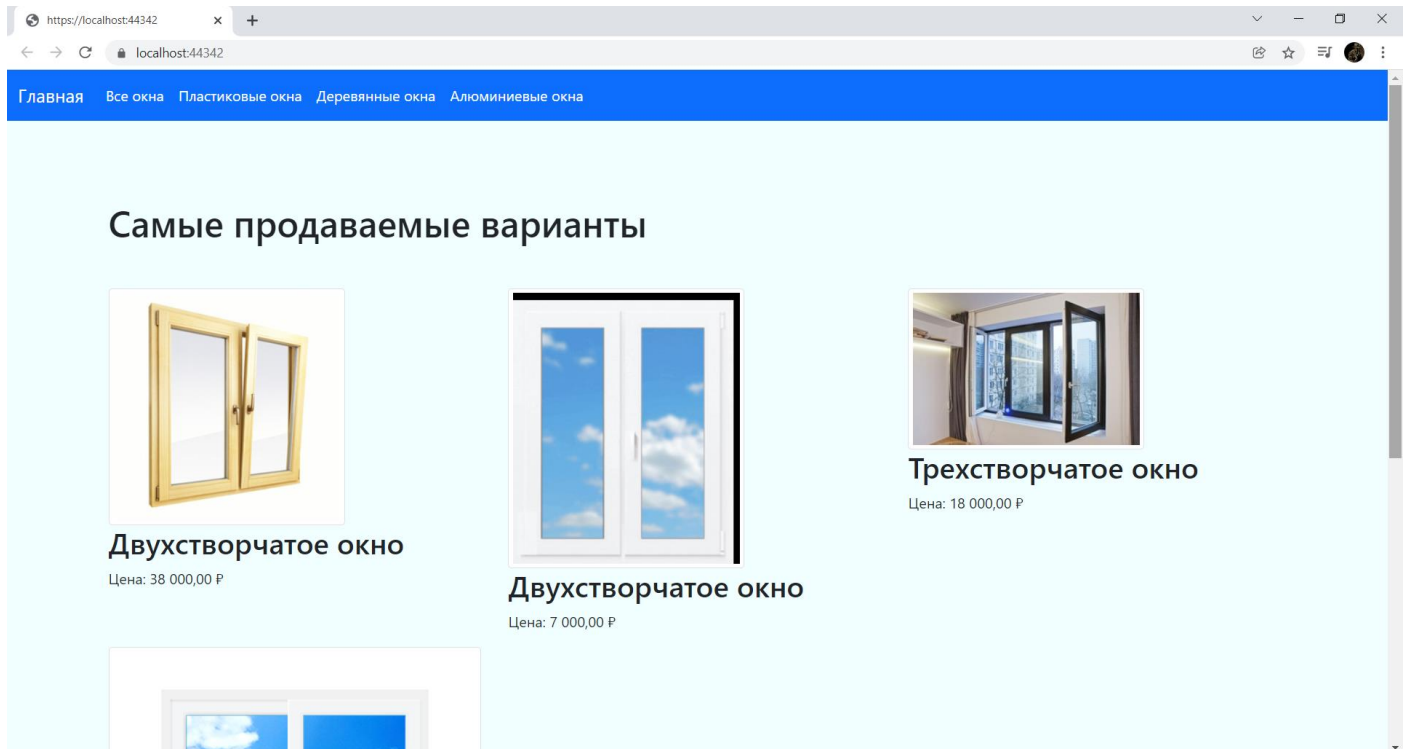
```

@{
    Layout = "_Layout";
}
<!-- отвечает за то, в какой шаблон будут вставлены блоки -->

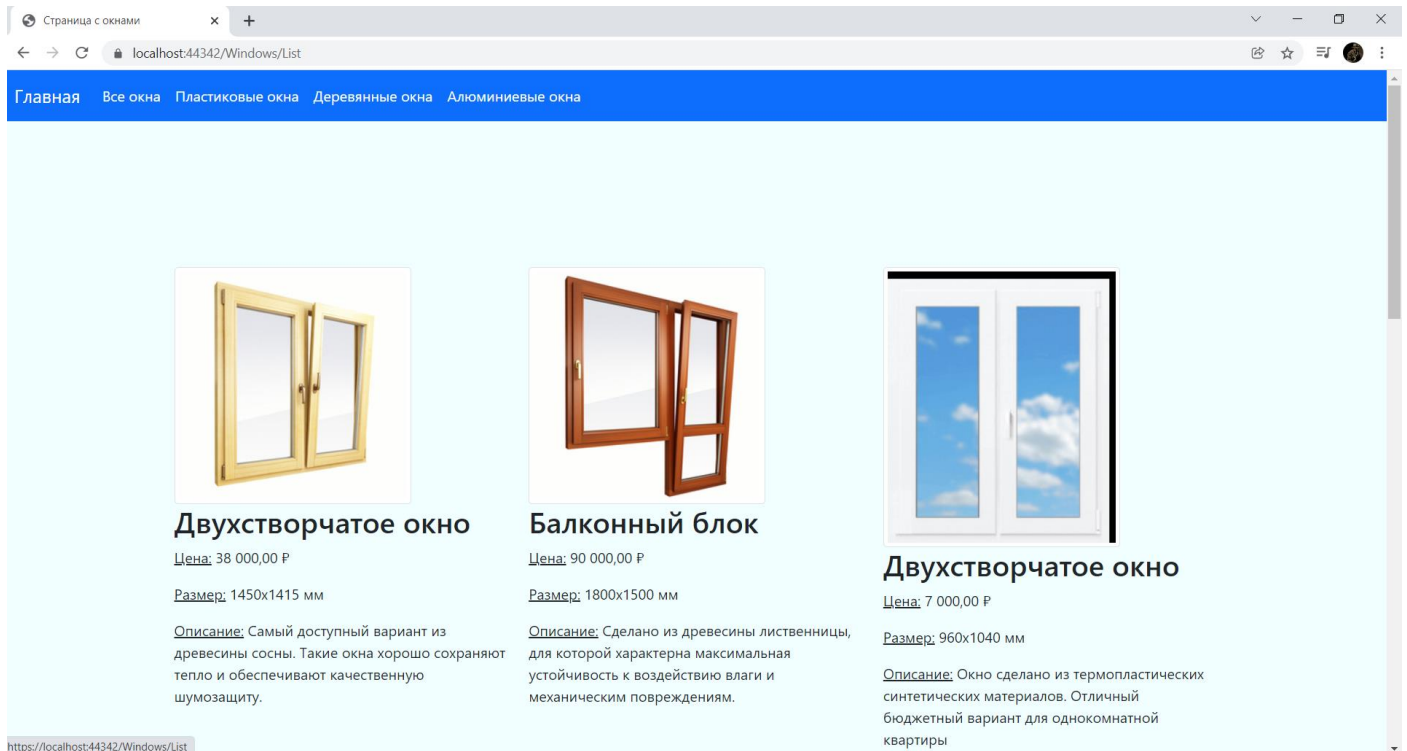
```

Тестирование сайта. Структура сайта. Демонстрационные примеры.

Главная страница



Страница со всеми окнами



Страница с пластиковыми окнами


Страница с окнами

localhost:44342/Windows/List/plastic

ГлавнаяВсе окнаПластиковые окнаДеревянные окнаАлюминиевые окна

Пластиковые окна

Пластиковые окна отличные конструкции, обеспечивающие качественную звукоизоляцию и предотвращающие потерю тепла.

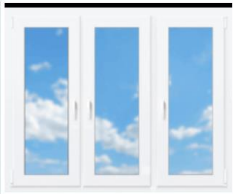


Двухстворчатое окно

Цена: 7 000,00 Р

Размер: 960x1040 мм

Описание: Окно лакировано в белом цвете. Мультифункциональный стеклопакет в подарок.




Трехстворчатое окно

Цена: 15 000,00 Р

Размер: 1560x2160 мм

Описание: Окно лакировано в белом цвете. Мультифункциональный стеклопакет в подарок.



Трехстворчатое окно

Цена: 18 000,00 Р

Размер: 1560x2160 мм

Описание: Они эффектно смотрятся в интерьере и на фасаде, делают вид проема более солидным. Цветное покрытие в оттенках RAL-палитры не выгорает на солнце.

Страница с деревянными окнами


Страница с окнами

localhost:44342/Windows/List/wood

ГлавнаяВсе окнаПластиковые окнаДеревянные окнаАлюминиевые окна

Деревянные окна

Красота и надежность от природы по выгодной цене




Двухстворчатое окно

Цена: 38 000,00 Р

Размер: 1450x1415 мм

Описание: Самый доступный вариант из древесины сосны. Такие окна хорошо сохраняют тепло и обеспечивают качественную звукоизоляцию.




Балконный блок

Цена: 90 000,00 Р

Размер: 1800x1500 мм

Описание: Сделано из древесины лиственницы, для которой характерна максимальная устойчивость к воздействию влаги и механическим повреждениям.



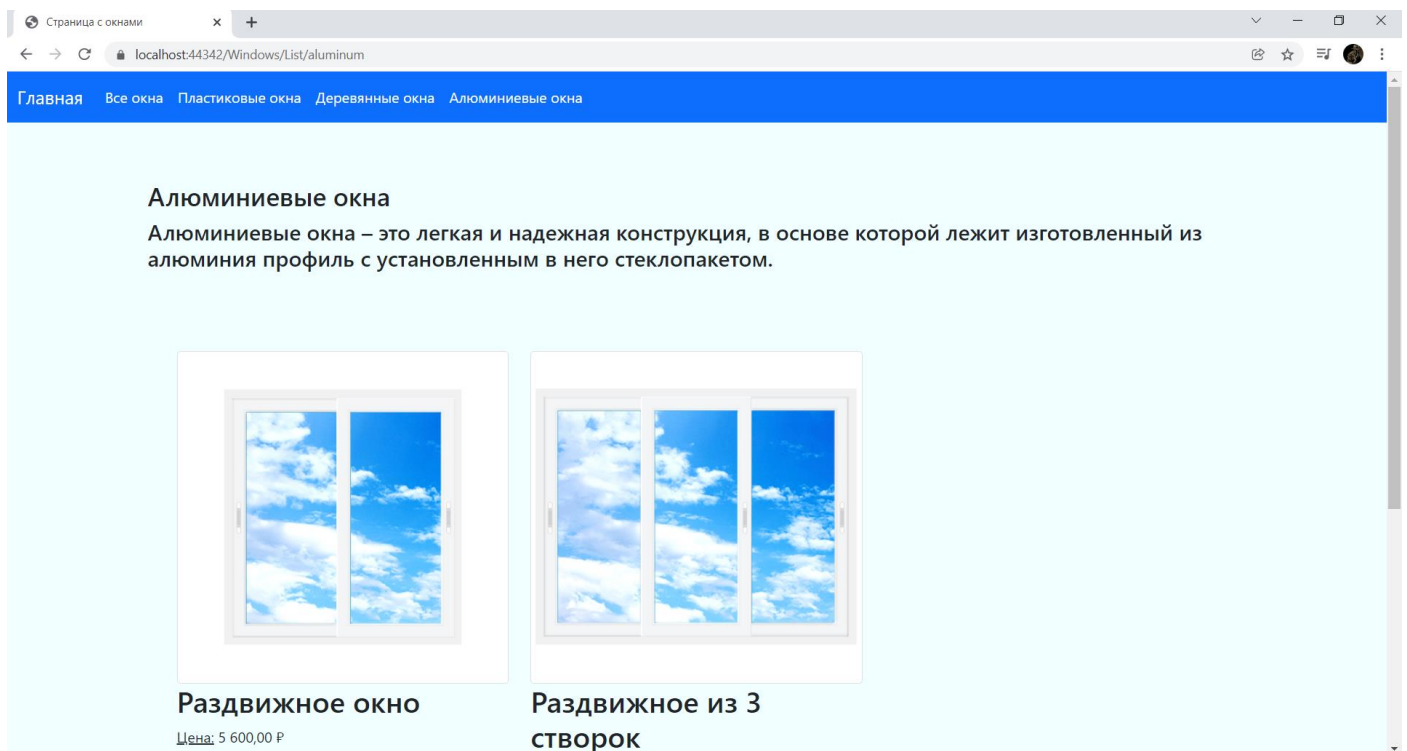
Одностоворчатое окно

Цена: 35 000,00 Р

Размер: 600x1415 мм

Описание: Сделано из дуба. Среди всех пород дерева дуб является бесспорным лидером по показателю прочности, плотности и влагостойкости.

Страница с алюминиевыми окнами



Вывод

В ходе курсового проекта мною была освоена технология Asp.Net MVC и я написал сайт для интернет-магазина.

СПИСОК ИСТОЧНИКОВ

- <https://cloud.mail.ru/public/Af3p/grG7LCtFh>
- <https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-3.1>
- https://ru.wikipedia.org/wiki/ASP.NET_MVC_Framework
- <https://ru.wikipedia.org/wiki/Model-View-Controller>