



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки

09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 1

Дисциплина: «Backend»

Тема: *Создание приложения на основе класса WebApplication на основе*

ASP.NET Core

Выполнил: студент группы: 231-339

Карапетян Нвер Каренович

(Фамилия И.О.)

Дата, подпись: 16.02.25

(Дата)

(Подпись)

Проверил: _____

(Фамилия И.О., степень, звание)

(Оценка)

Дата, подпись _____

(Дата)

(Подпись)

Москва
2025

Цель:

Ознакомиться с базовыми шагами создания веб-приложения на основе класса `WebApplication` в ASP.NET Core.

Задачи:

1. Создать новый проект ASP.NET Core приложения, используя класс `WebApplication`.
2. Протестировать работу приложения локально.

Ход работы

MVC в ASP.NET Core

MVC (Model-View-Controller) — это архитектурный шаблон, который разделяет приложение на три основных компонента:

- **Model** — модель, которая представляет данные и логику приложения. Это классы, которые содержат данные в виде полей.
- **View** — представление, которое отображает данные пользователю. В ASP.NET Core это обычно HTML-шаблоны, которые содержат код `Razor` для динамической генерации контента.
- **Controller** — контроллер, который обрабатывает запросы от пользователей, манипулирует данными модели и передает их в представления.

ASP.NET Core использует этот шаблон для структурирования веб-приложений, обеспечивая разделение логики, представлений и данных.

Для того, чтобы создать проект на основе `WebApplication` с использованием шаблона MVC, необходимо открыть среду разработки Visual Studio 2022 и создать новый проект с типом «Веб-приложение ASP.NET Core (модель-представление-контроллер) (Майкрософт)»:

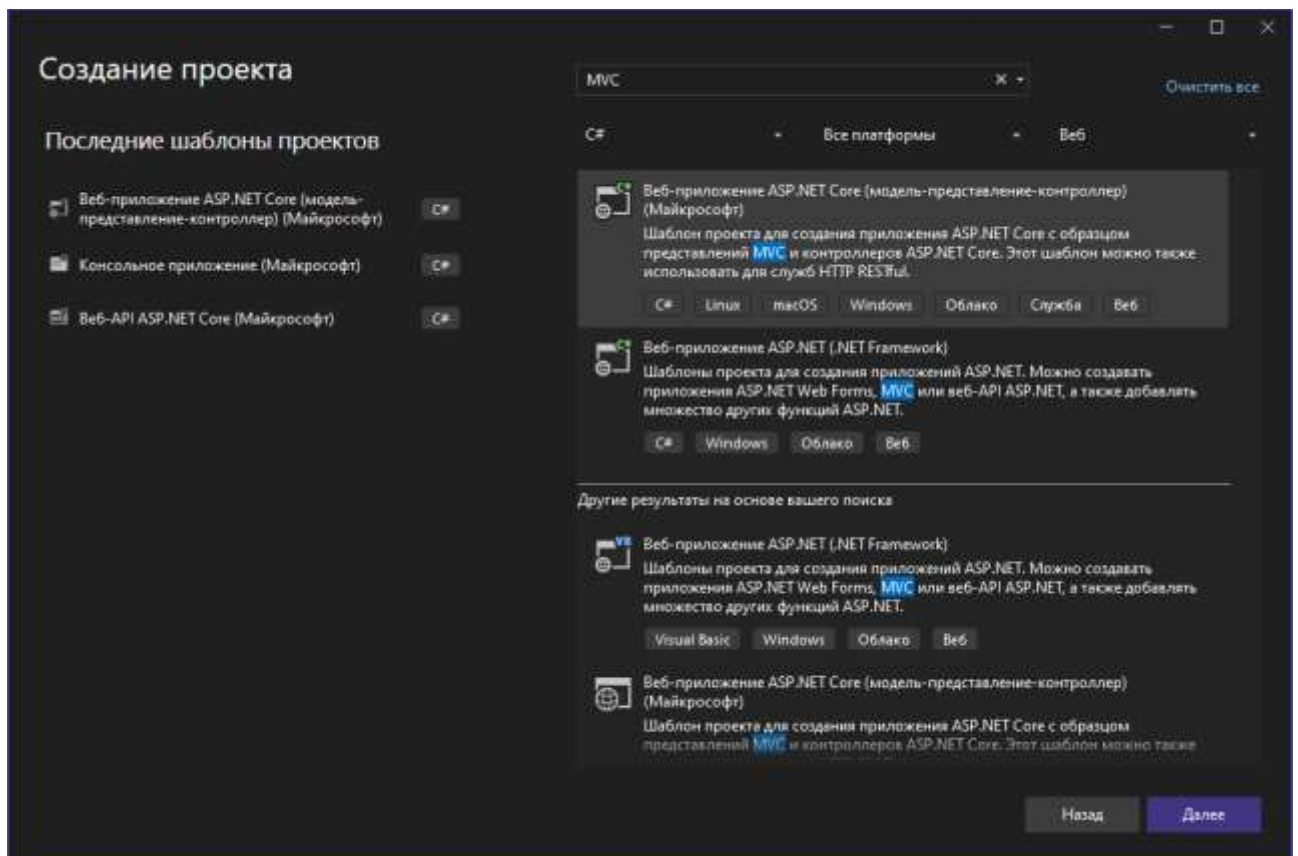


Рисунок 1. Создание проекта с применением шаблоном MVC.

Структура проекта

После этого Visual Studio создает проект с базовой структурой, которая включает в себя папки для моделей, контроллеров и представлений, а также базовый код для маршрутизации и работы с представлениями.

Базовая структура проекта состоит из следующих элементов ([см. Приложение А](#)):

- **Connected Services** — этот раздел используется для подключения к внешним сервисам, API, облачным платформам (например, Azure) и другим источникам данных. Здесь настраиваются и регистрируются внешние зависимости, что упрощает интеграцию с ними (например, подключение WCF-сервисов, REST API, сторонних SDK).
- **Properties** — содержит файлы, отвечающие за метаданные и настройки проекта.

- **launchSettings.json** — наиболее важный файл, который задает конфигурации для отладки и запуска (например URL-адреса, переменные окружения, профили запуска). Эти настройки помогают определить, как будет запускаться приложение в режиме разработки.
- **wwwroot** — корневая директория для статических файлов, доступных напрямую из браузера.
 - **css** — папка, в которой хранятся файлы с таблицами стилей для оформления сайта.
 - **js** — здесь хранятся скрипты JavaScript, отвечающие за интерактивность сайта.
 - **lib** — раздел со сторонними библиотеками (Bootstrap, jQuery и т.д.), установленные через LibMan или вручную.
 - Нередко здесь создают и другие папки для хранения медиафайлов (например, **images**, **fonts**, **icons** и др.).
- **Dependencies (Зависимости)** — этот узел показывает все внешние зависимости проекта. Здесь видны подключенные NuGet-пакеты, ссылки на сборки, COM-компоненты и другие ресурсы. Фактически, это не физическая папка, а представление всех библиотек, используемых в проекте.
- **Controllers** — эта папка содержит классы-контроллеры, которые обрабатывают входящие HTTP-запросы. Контроллеры реализуют логику приложения: получают запросы, обрабатывают данные (взаимодействуя с моделями) и выбирают соответствующее представление для рендеринга ответа пользователю.
- **Models** — предназначена для хранения классов, описывающих данные и бизнес-логику. Модели могут включать свойства (данные), методы валидации, бизнес-правила и даже взаимодействие с базой данных. Они являются связующим звеном между контроллерами и представлениями.
- **Views** — содержит Razor-представления (.cshtml), которые отвечают за отображение данных и формирование HTML-кода, отправляемого клиенту.

- **Папки, соответствующие контроллерам** — например, папка **Home** содержит представления, используемые HomeController.
- **_Layout.cshtml** — общий шаблон для страниц, содержащий базовую разметку (навигация, футер, подключение стилей и скриптов). Представления вставляются в него через `@RenderBody()` (см. [Приложение В](#)).
- **Shared** — папка для общих представлений (например, частичные представления, ошибки), которые могут использоваться несколькими контроллерами.
- **appsettings.json** — файл конфигурации приложения. Содержит настройки в формате JSON, такие как строки подключения к базе данных, параметры логирования, настройки сервисов и другие параметры, которые приложение использует при запуске. При необходимости можно использовать и дополнительные файлы (например, `appsettings.Development.json`) для различных сред.
- **Program.cs** — точка входа в приложение. Здесь содержится следующий скрипт по умолчанию:

```

1  namespace Laba2
2  {
3      class Program
4      {
5          // СОЗДАНИЕ ЭНТЕРПРИЗА WebApplication
6          public static void Main(string[] args)
7          {
8              var builder = WebApplication.CreateBuilder(args);
9
10             // РЕГИСТРАЦИЯ СЕРВИСОВ (осуществляется через Dependency Injection)
11             builder.Services.AddControllersWithViews();
12
13             var app = builder.Build();
14
15             // Configure the HTTP request pipeline.
16             if (!app.Environment.IsDevelopment())
17             {
18                 app.UseExceptionHandler("/Home/Error");
19                 // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
20                 app.UseHsts();
21             }
22             // КОНФИГУРАЦИЯ middleware (например, маршрутизация, обработка статических файлов, обработка ошибок)
23             app.UseHttpsRedirection();
24             app.UseRouting();
25
26             app.UseAuthorization();
27
28             app.MapStaticAssets();
29             app.MapControllerRoute(
30                 name: "default",
31                 pattern: "{controller=Home}/{action=Index}/{id?}");
32             app.UseStaticAssets();
33
34             // ЗАПУСК ПРИЛОЖЕНИЯ
35             app.Run();
36         }
37     }
38 }

```

Рисунок 2. Листинг скрипта Program.cs

Когда мы запускаем проект, в браузере отображается главная страница, определенная в **HomeController** в методе **Index**. Приложение автоматически маршрутизирует запросы и отображает представления.

По умолчанию контроллер **HomeController** управляет двумя основными представлениями: **Index.cshtml** и **Privacy.cshtml**. Страница по адресу / загружает **HomeController.Index**, а страница /Home/Privacy — **HomeController.Privacy**.

Шаблон **_Layout.cshtml** используется для обертки всех страниц, чтобы обеспечить единообразный внешний вид (например, меню навигации, футер и стили).

Запустив приложение, нас встретит следующая страница, на которой есть шапка с навигацией по страницам (по умолчанию есть две страницы: Home и Privacy), основной контент и футер:

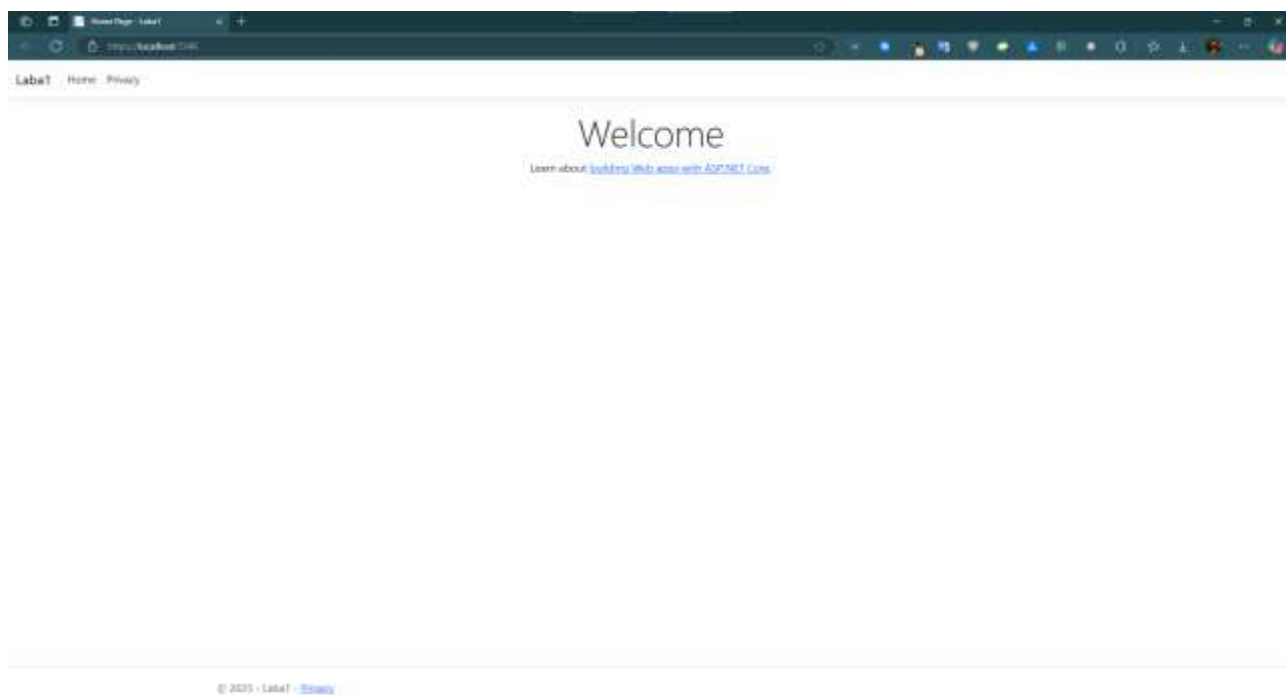


Рисунок 3. Главная страница нашего проекта.

Создание новых страниц

В папку **Controllers** добавим несколько новых контроллеров: **ContactsController.cs**, **FaqController.cs** и **AboutController.cs** для страниц с контактной информацией, часто задаваемыми вопросами и информации о компании соответственно.

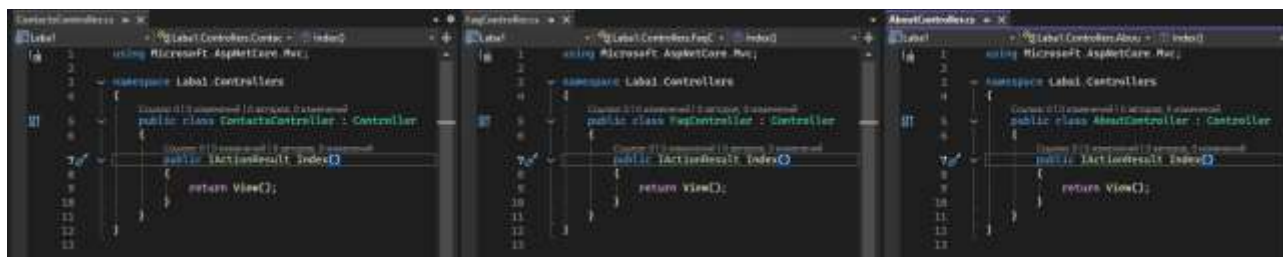


Рисунок 4. Листинги трех скриптов-контроллеров.

Нажав правой кнопкой мыши на метод **Index**, для каждого контроллера нажмем «Добавить представление», тем самым создав в папке **Views** подпапки с названиями контроллеров (но без слова «Controller»). Внутри созданной подпапки автоматически появится файл **Index.cshtml**, который будет являться представлением для метода **Index** данного контроллера. Этот файл можно будет использовать для определения разметки и логики отображения данных, связанных с данным действием.

С помощью компонентов Bootstrap (можно найти на сайте с [официальной документацией](#) по Bootstrap) в каждом из файлов **Index.cshtml** для каждой из страниц сверстаем небольшое содержимое (например, форму для страницы контактов, «аккордеон» на странице FAQ и небольшое описание на странице «О компании»).

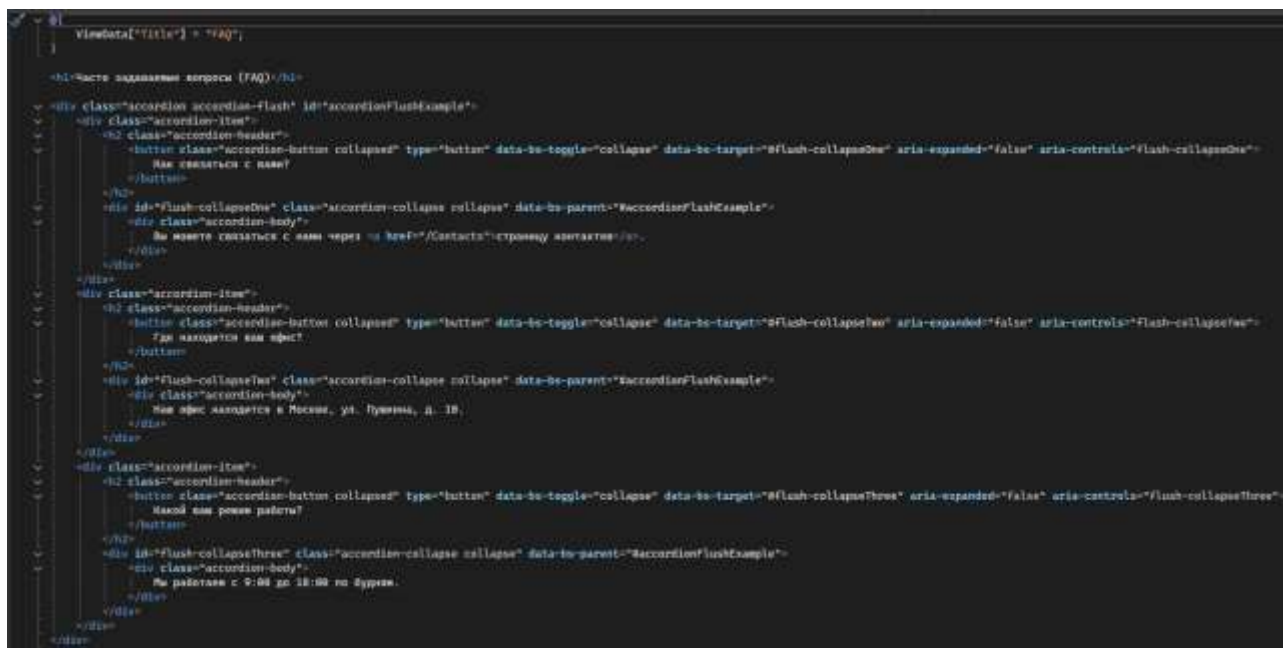


Рисунок 5. Пример листинга файла Index.cshtml для страницы FAQ.

Как работает MVC-приложение

При поступлении HTTP-запроса система маршрутизации анализирует URL и определяет, какой контроллер и действие должны обработать запрос. Например, запрос к `/Contacts` будет направлен в метод **Index** контроллера **ContactsController**.

Контроллеры, в свою очередь, содержат методы (действия), которые обрабатывают запросы, взаимодействуют с данными (через модели) и выбирают соответствующее представление для формирования ответа. В нашем случае мы создали контроллеры, такие как **ContactsController**, **FaqController** и **AboutController**. Метод **Index** каждого контроллера возвращает представление, которое будет отображаться пользователю.

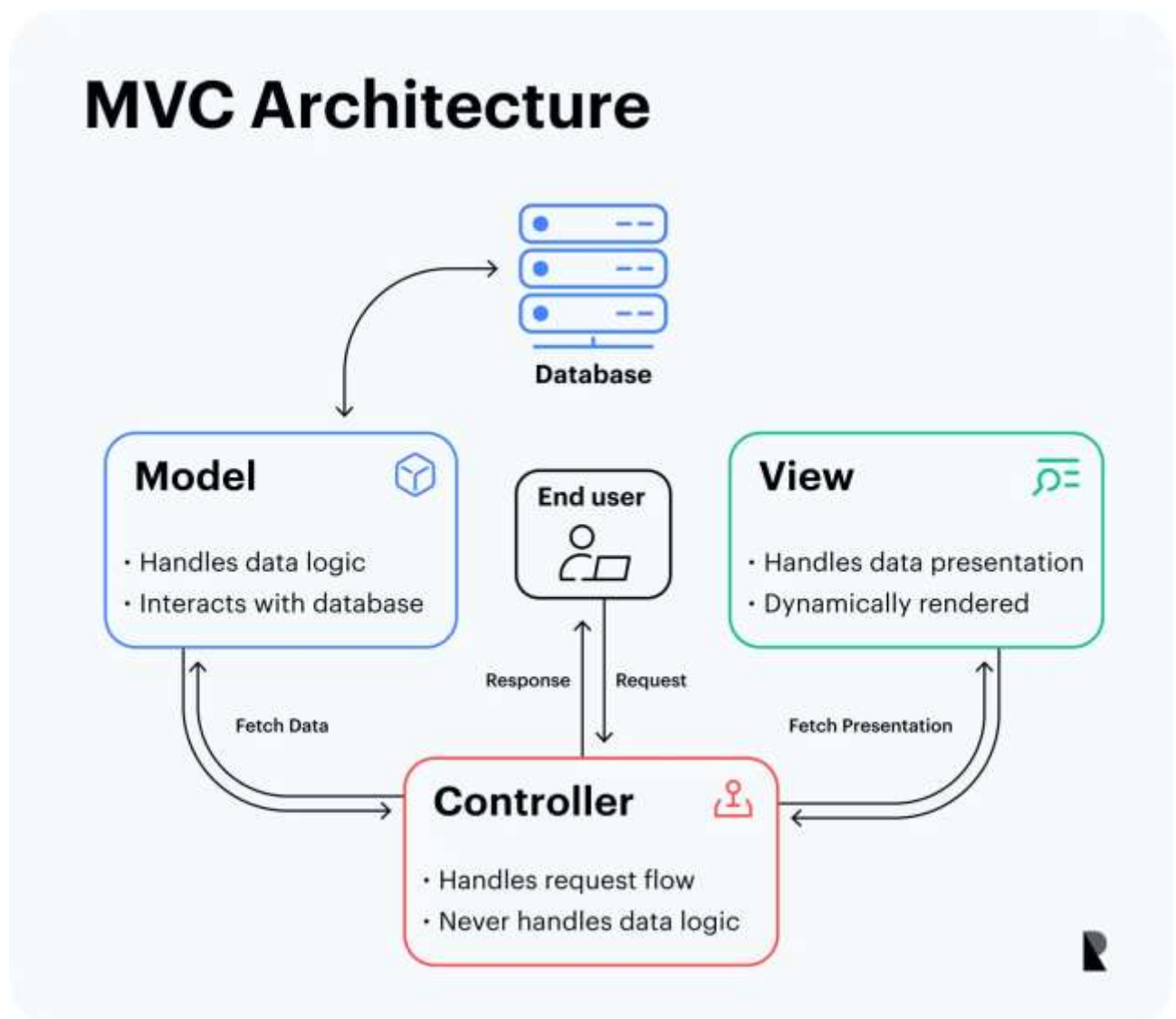
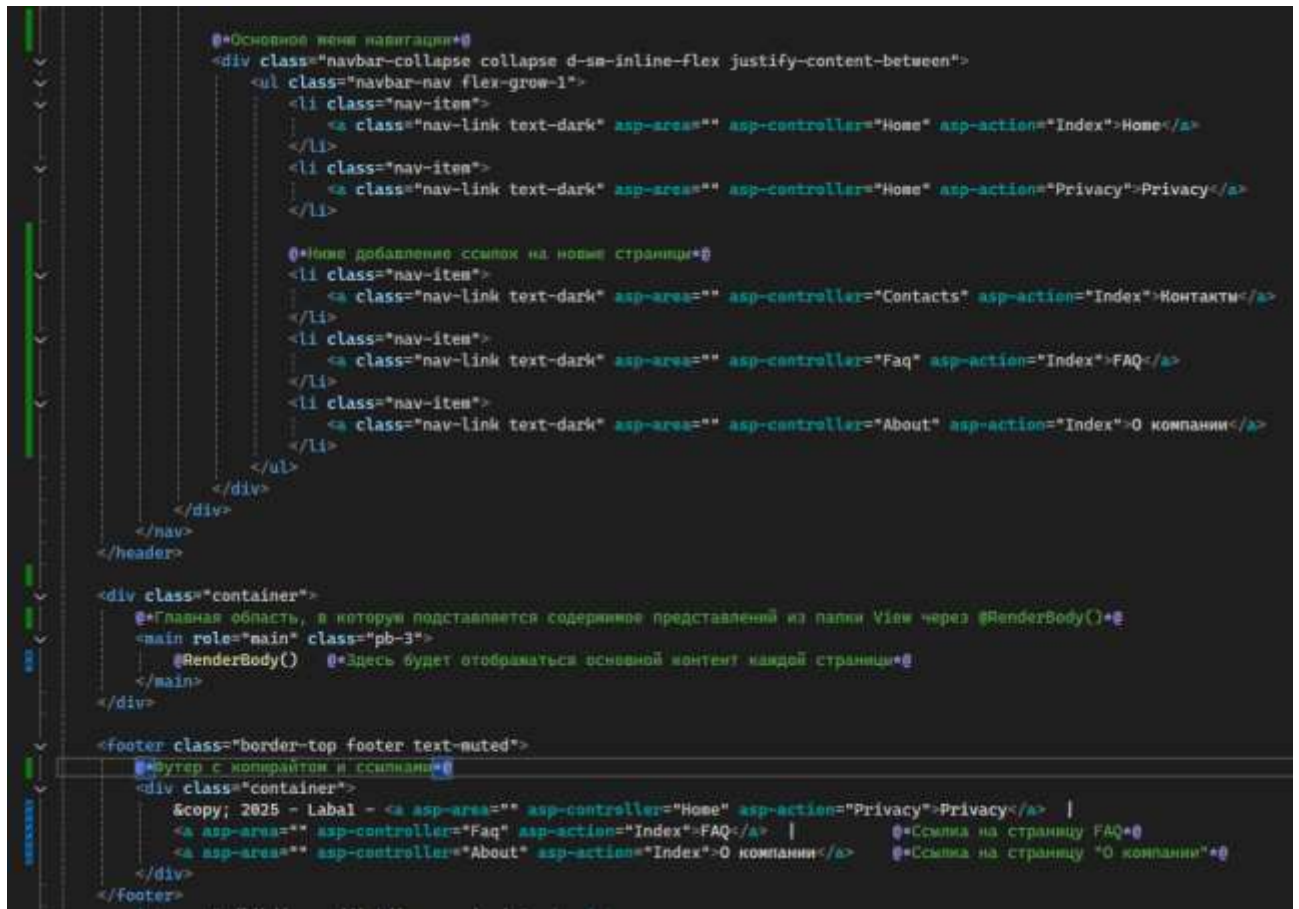


Рисунок 6. Схема работы MVC-приложений.

Следующим шагом нам предстоит привнести некоторые изменения в шаблоне **_Layout.cshtml**, добавив ссылки в шапке навигационного меню и футера для перехода на созданные нами страницы:



```
@*Основное меню навигации*@
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </li>
    @*Добавление ссылок на новые страницы*@
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Contacts" asp-action="Index">Контакты</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Faq" asp-action="Index">FAQ</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="About" asp-action="Index">О компании</a>
    </li>
  </ul>
</div>
</nav>
</header>

<div class="container">
  @*Главная область, в которую подставляется содержимое представлений из папки View через @RenderBody()*@
  <main role="main" class="pb-3">
    @RenderBody() @*Здесь будет отображаться основной контент каждой страницы*@
  </main>
</div>

<footer class="border-top footer text-muted">
  @*Футер с копирайтом и ссылками*@
  <div class="container">
    &copy; 2025 - Labal - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a> |
    <a asp-area="" asp-controller="Faq" asp-action="Index">FAQ</a> | @*Ссылка на страницу FAQ*@
    <a asp-area="" asp-controller="About" asp-action="Index">О компании</a> @*Ссылка на страницу "О компании"*@
  </div>
</footer>
```

Рисунок 7. Измененный фрагмент кода шаблона **_Layout.cshtml** с ссылками на новые страницы.

Запустив наше приложение, мы можем увидеть наши новые страницы и перейти на них, нажав на соответствующие ссылки в шапке навигационного меню или в футере:

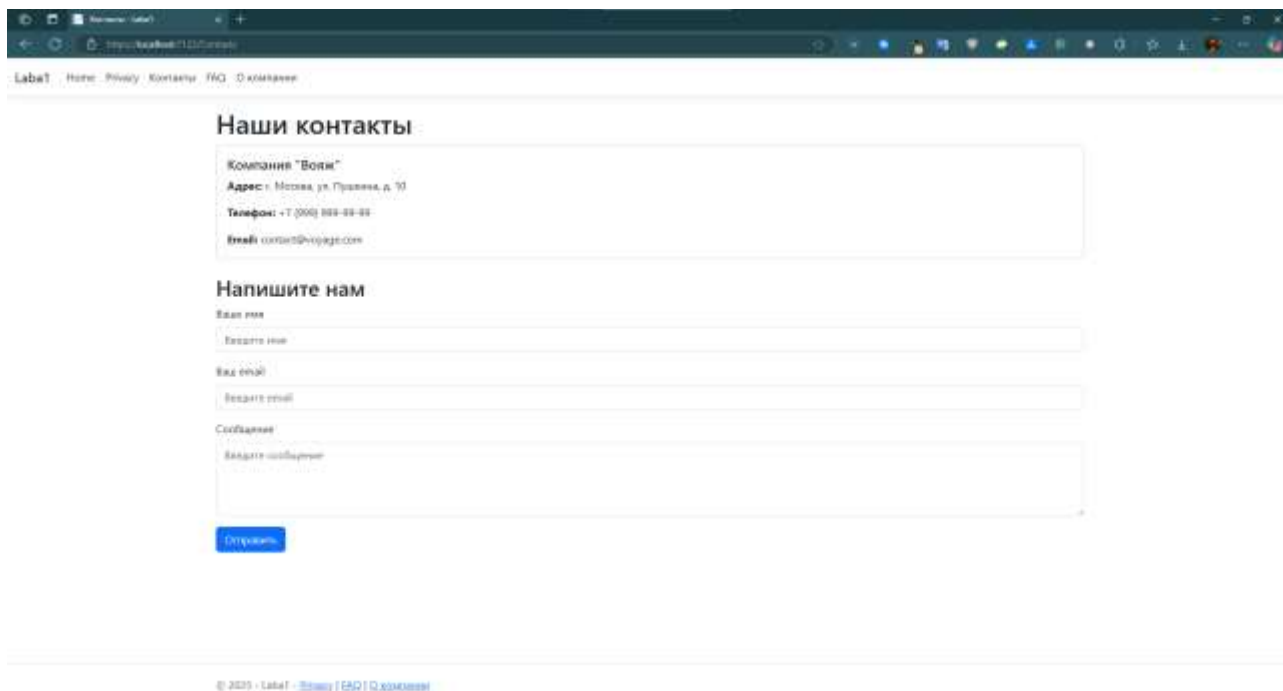


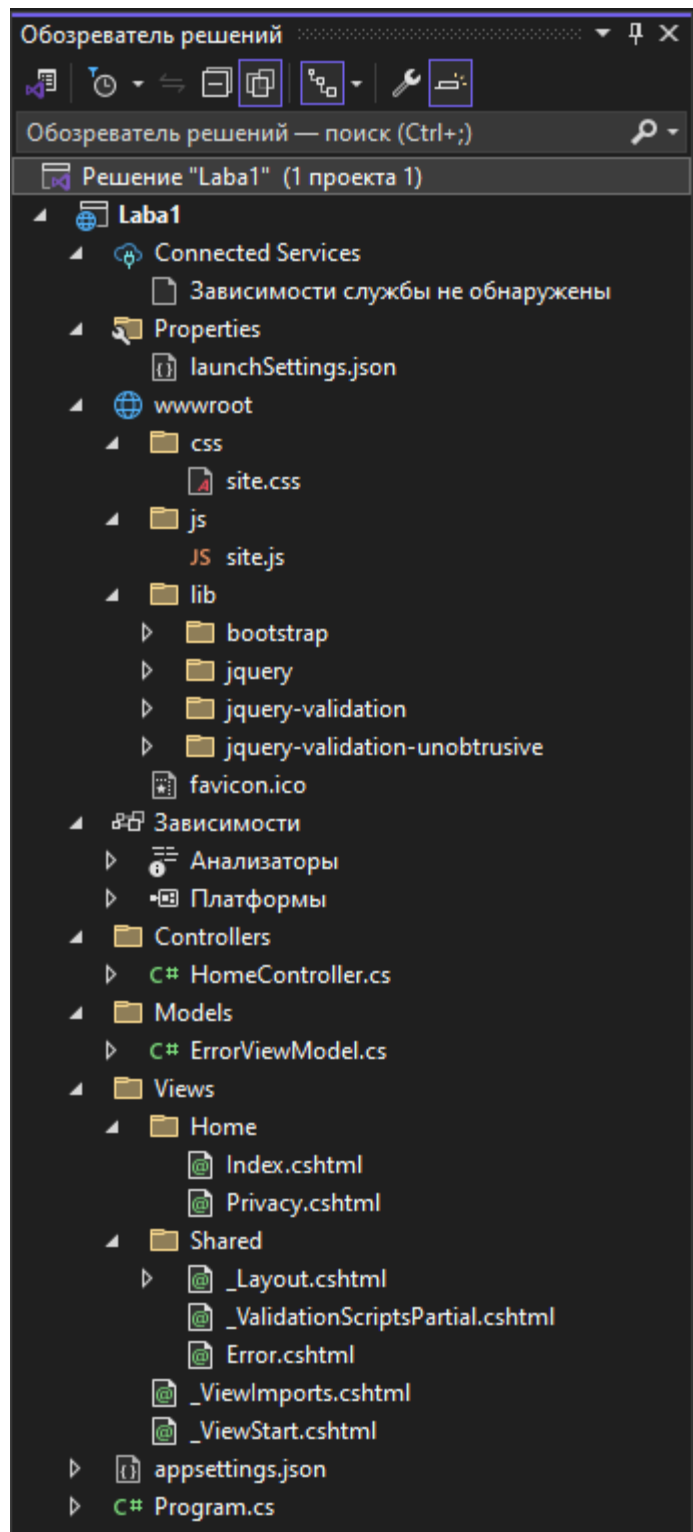
Рисунок 8. Демонстрация готовой страницы «Контакты».

Заключение

В данной лабораторной работе было создано веб-приложение на основе класса WebApplication с использованием паттерна MVC в ASP.NET Core, изучена базовая структура проекта, реализована маршрутизация, добавлены новые страницы через контроллеры и представления, а также обновлен общий шаблон навигации. В результате был получен простенький прототип многостраничного веб-приложения.

Приложение

А) Базовая структура MVC-проекта:



В) Листинг с комментариями файла-шаблона _Layout.cshtml

```
<!DOCTYPE html>  
<html lang="en">
```

```

<head>
    @*Указание кодировки и настройка отображения страницы*@
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    @*Заголовок страницы с динамическим значением благодаря C#*@
    <title>@ViewData["Title"] - Laba2</title>

    <script type="importmap"></script>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
    <link rel="stylesheet" href="~/Laba2.styles.css" asp-append-version="true"
/>
</head>
<body>
    <header>
        @*Навигационная панель с использованием Bootstrap-классов*@
        <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light
bg-white border-bottom box-shadow mb-3">
            <div class="container-fluid">
                @*Логотип или название сайта с переходом на главную страницу*@
                <a class="navbar-brand" asp-area="" asp-controller="Home" asp-
action="Index">Laba2</a>

                @*Кнопка для раскрытия меню на мобильных устройствах*@
                <button class="navbar-toggler" type="button" data-bs-tog-
gle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupported-
Content"

                    aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
                </button>

                @*Основное меню навигации*@
                <div class="navbar-collapse collapse d-sm-inline-flex justify-
content-between">
                    <ul class="navbar-nav flex-grow-1">
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-con-
troller="Home" asp-action="Index">Home</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link text-dark" asp-area="" asp-con-
troller="Home" asp-action="Privacy">Privacy</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>

```

```

    <div class="container">
        @*Главная область, в которую подставляется содержимое представлений из
        папки View через @RenderBody()*@
        <main role="main" class="pb-3">
            @RenderBody()    @*Здесь будет отображаться основной контент каждой
            страницы*@
        </main>
    </div>

    @*Футер сайта с копирайтом и ссылками*@
    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2025 - Laba2 - <a asp-area="" asp-controller="Home" asp-ac-
            tion="Privacy">Privacy</a>
        </div>
    </footer>

    <script src="~/lib/jquery/dist/jquery.min.js"></script>
    <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
    <script src="~/js/site.js" asp-append-version="true"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body>
</html>

```