**1 Создание проекта на основе шаблона *MVC*.**

Данный проект представляет собой веб приложение туристического агентства для написания *frontend* используется *CSS* – стили проекта, *HTML* – разметка страницы и *JavaScript* – динамическое взаимодействие. *Beckend* будет написан на языке *C#* а в качестве системы управления базами данных выбран *PostgreSql.*

Для начала необходимо зайти в *Visual studio* и выбрать пункт «Создание проекта» как показано на рисунке 1.

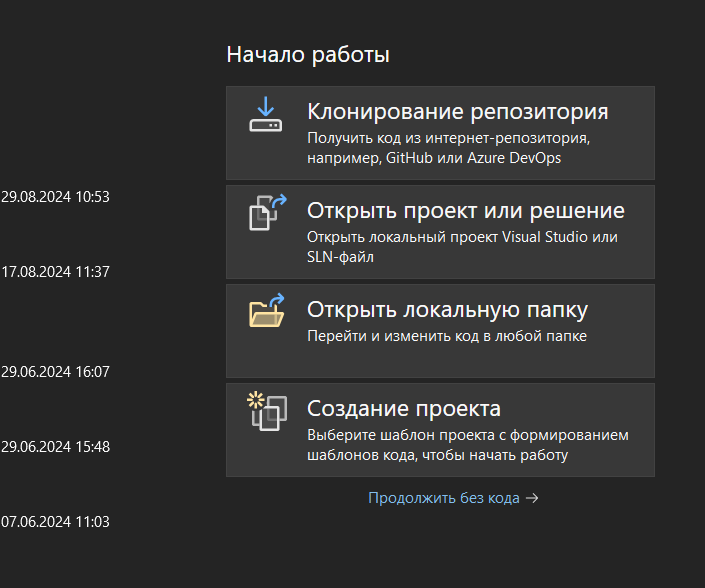


Рисунок 1 – Создание проекта

После выбираем необходимые параметры для поиска нужного шаблона либо вводим название «Веб-приложение *ASP.NET Core* (модель-представление-контроллер» в строку поиска как можно увидеть на рисунке 2. Данный проект будет написан с использованием архитектуры *MVC*.

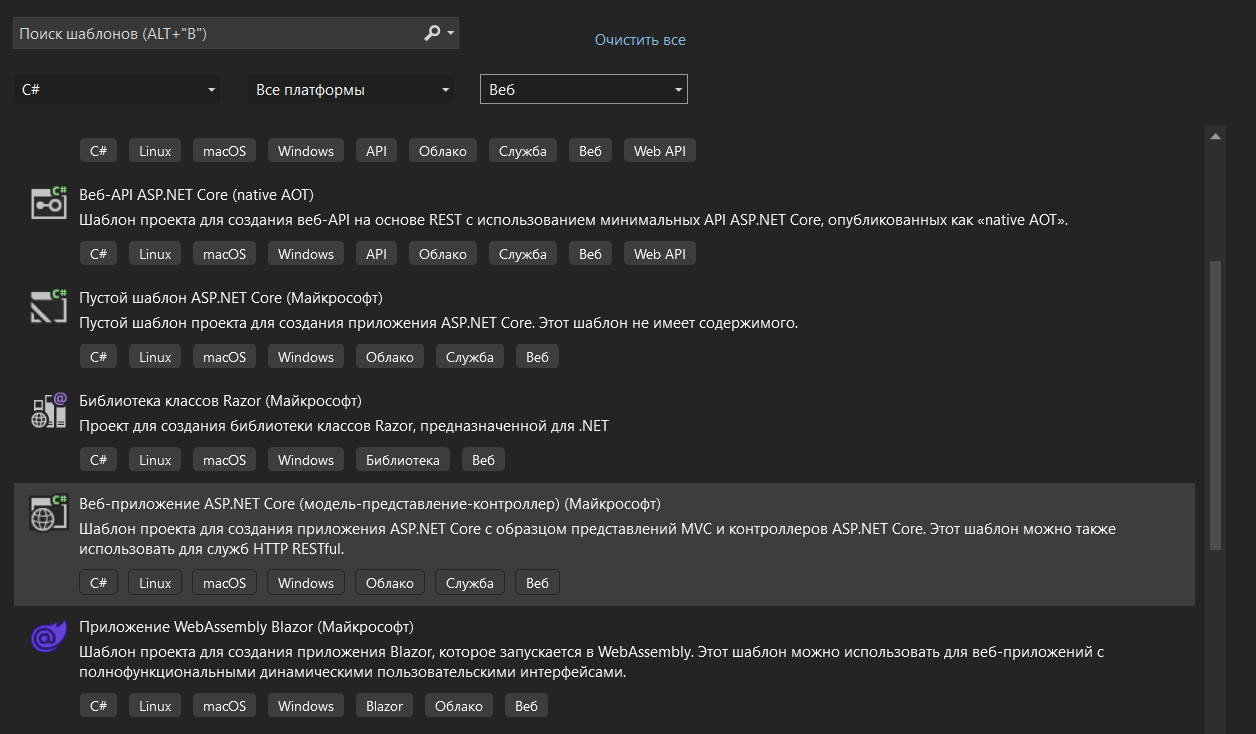


Рисунок 2 – Шаблоны

*MVC* расшифровывается как «модель-представление-контроллер» (от англ. *model-view-controller*). Это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.

Когда нужный шаблон был выбран указываем имя проекта «*TravelAgency*» и путь к проекту. После нажимаем кнопку «Далее» в появившемся окне необходимо выбрать платформу .*NET* для нашего проекта (.Net 6 или выше).

В обозревателе решений на рисунке 3 показана структура данного шаблона на начальном этапе разработки. Если обозреватель решений не закреплен на панели его можно найти в Вид/Обозреватель решений.

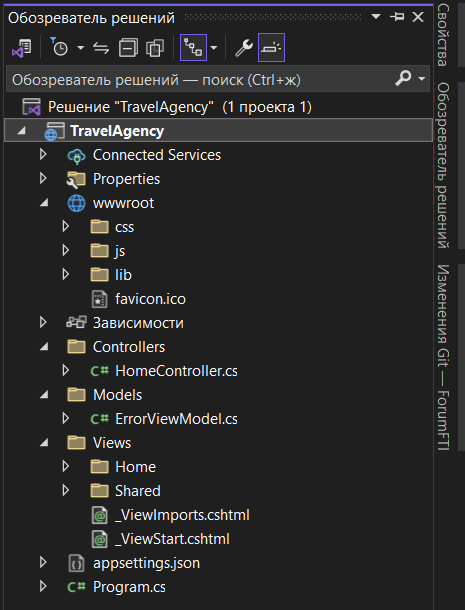


Рисунок 3 – Обозреватель решений

Здесь есть несколько основных папок с которыми мы будем работать.

Папка *wwwroot* содержит в себе папку для стилей проекта, папку, содержащую скрипты для динамического взаимодействия с проектом, и папку с дополнительными файлами которые могут упростить работу если знать, как их использовать. В данном проекте мы будем использовать чистый JavaScript.

Папка *Controllers* – обрабатывает входящие запросы. Во фреймворке это может заключаться в определении конкретных *URL*, на которые попадает пользователь при переходе по ссылке или при нажатии кнопки.

Папка *Models* – отвечает за данные, которые хранятся и обрабатываются на сервере.

Папка *Views* – это *HTML*-шаблон, который возвращает сервер после обработки запроса.

В обозревателе решений выбираем *Views/Shered/\_Layout.cshtml*. Файлы *layout* или мастер-страницы позволяют определить единый шаблон для страниц *Razor* и применяются для создания единообразного, унифицированного вида приложения. Для определения интерфейса мастер-страницы также применяют код *Razor* и *html*, как обычные страницы *razor*, но при этом упрощают создание приложения. Например, можно определить на мастер-странице общие для всех остальных страниц *Razor* меню и другие элементы, а также подключить общие стили и скрипты. В итоге нам не придется на каждой отдельной странице прописывать путь к файлам стилей, а потом при необходимости его изменять.

Тег <*head*> показанный на рисунке 4 предназначен для хранения служебной информации о странице. Он располагается первым в теге <*html*>, сразу перед <*body*>. Внутри <*head*> обычно содержится заголовок, ключевые слова, описание страницы и другие служебные данные.

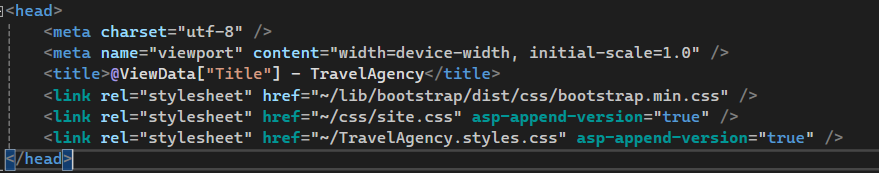


Рисунок 4 – тег <head>

<*header*> – это семантичный элемент, который нужен для отделения вводного содержимого или навигации от основного содержимого целой страницы или секционного блока. Например, «шапка» с навигацией для всего сайта или имя автора и дата публикации в превью статьи или новости.

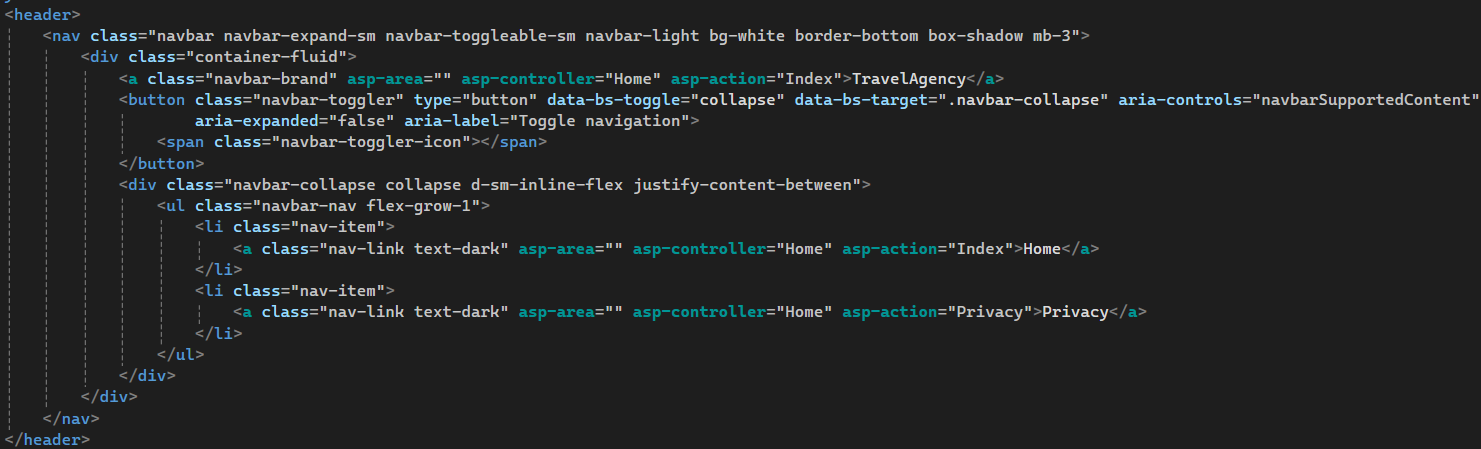


Рисунок 5 – тег <header>

Отличительной особенностью файлов *layout* является использование метода @*RenderBody()*. Этот метод будет вставлять содержимое страниц *Razor*, которые используют данную мастер-страницу. В итоге мы сможем легко установить для всех страниц единообразный стиль оформления.

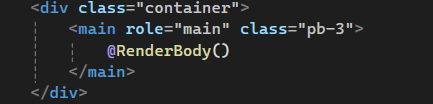


Рисунок 6 – функция *@RenderBody*()

Tег <*footer*> служит для создания футера сайта или же подвала сайта. Это нижняя часть сайта, в которой может находиться какая-нибудь контактная информация, нижнее меню, ссылки или же что-то в этом духе.

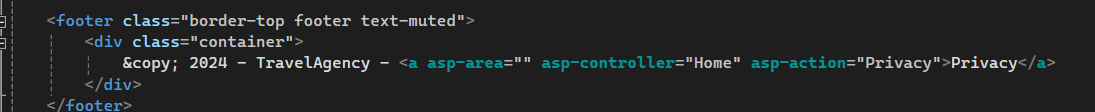


Рисунок 7 – тег <*footer*>

Скрипты чаще всего подключают из внешних файлов с расширением .*js*. Для этого используют тег <*script*> с атрибутом *src*, в котором указывается путь к файлу. Внешние скрипты лучше подключать перед закрывающим тегом </*body*>.

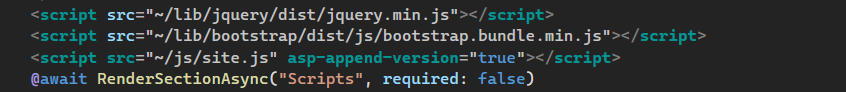


Рисунок 8 – подключение скриптов

Рассмотрим еще два файла находящихся в папке *Views/Shered/* *\_ViewImports.cshtml* и *\_ViewStart.cshtml.* В первом файле находятся всего две строки кода показанные на рисунке 9. Эти немногочисленные строки предназначены для глобального подключения пространств имен и дополнительных возможностей, которые применяются ко всем представлениям в вашем проекте *ASP.NET Core MVC*.



Рисунок 9 – содержимое файла *\_ViewImports*

В файле *\_ViewStart.cshtml* так же всего пару строк, как и показано на рисунке 10. Эта строка в *Razor*-представлении *ASP.NET Core* используется для указания файла макета (*layout*), который будет применен к текущему представлению.

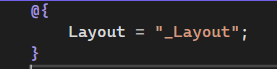


Рисунок 10 – содержимое файла *\_ViewStart.cshtml*

Рассмотрим еще одну папку в разделе представлений – папка *Home.* В этой папке хранятся представления, которые связаны с действиями *(Action Methods)* контроллера *HomeController.* Этот контроллер обычно отвечает за основные страницы веб-приложения, такие как главная страница, страница контактов и т.д. Так же по умолчанию при запуске проекта вызывается именно этот контролер и действие *Index* это можно изменить в файле *Program.cs* в строках показанных на рисунке 11.

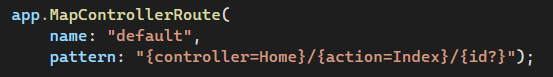


Рисунок 11 – фрагмент кода из файла *Program.cs*