**МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ ПМР**

**ГОУ СПО «ТИРАСПОЛЬСКИЙ ТЕХНИКУМ ИНФОРМАТИКИ И ПРАВА»**

**ДНЕВНИК**

**ПРОХОЖДЕНИЯ практики**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**УП.01.01\_Учебная практика**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

вид практики

Профессиональный модуль \_\_\_ПМ.01 Разработка модулей программного\_\_\_ обеспечения для компьютерных систем \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

индекс, наименование профессионального модуля

Междисциплинарный курс \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_МДК 01.01. Разработка программных модулей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_МДК 01.02. Поддержка и тестирование программных модулей\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

индекс, наименование междисциплинарного курса

Ф.И.О. студента(ки)\_\_\_\_Сиглов Артем Витальевич \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность \_\_2.09.02.07 Информационные системы и программирование\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

код, наименование специальности

Курс \_\_\_\_\_\_3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ группа \_\_\_315\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Время прохождения практики: с «\_1\_» \_\_11\_\_ 20\_24\_ г. по «\_12\_» \_\_12\_\_ 20\_24\_ г.

База практики \_\_ГОУ СПО «Тираспольский техникум информатики и права»\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№**  **п/п** | **Дата** | **Описание выполненной работы** | **Количество часов** | **Оценка и подпись руководителя практики** |
|  | 01.11.2024 | Роль практического обучения при формировании компетенций обучающихся.  Инструктаж по технике безопасности и правилам поведения на учебной практике. | 2 |  |
|  | 01.11.2024 | Создание проекта на основе шаблона MVC. | 2 |  |
|  | 01.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 02.11.2024 | Использование Git для контроля версий. | 2 |  |
|  | 02.11.2024 | Использование Git для контроля версий. | 2 |  |
|  | 02.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 04.11.2024 | Определение структуры MVC-приложения, взаимодействие моделей, контроллеров и представлений. | 2 |  |
|  | 04.11.2024 | Определение структуры MVC-приложения, взаимодействие моделей, контроллеров и представлений. | 2 |  |
|  | 04.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 05.11.2024 | Frontend разработка основная страница: написание шапки сайта, секции «О нас». | 2 |  |
|  | 05.11.2024 | Frontend разработка основная страница: написание шапки сайта, секции «О нас». | 2 |  |
|  | 05.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 06.11.2024 | Frontend разработка основная страница: написание секций «Услуги», «Контакты». | 2 |  |
|  | 06.11.2024 | Frontend разработка основная страница: написание секций «Услуги», «Контакты». | 2 |  |
|  | 06.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 07.11.2024 | Frontend разработка основная страница: написание форм входа/регистрации. | 2 |  |
|  | 07.11.2024 | Frontend разработка основная страница: написание форм входа/регистрации. | 2 |  |
|  | 07.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 08.11.2024 | Frontend разработка основная страница: написание секции «Написать сообщение» и подвала сайта. | 2 |  |
|  | 08.11.2024 | Frontend разработка основная страница: написание секции «Написать сообщение» и подвала сайта. | 2 |  |
|  | 08.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 09.11.2024 | Использование fetch-запросов для обновления данных без перезагрузки страницы. | 2 |  |
|  | 09.11.2024 | Использование fetch-запросов для обновления данных без перезагрузки страницы. | 2 |  |
|  | 09.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 11.11.2024 | Создание адаптивного интерфейса с использованием медиазапросов CSS. | 2 |  |
|  | 11.11.2024 | Создание адаптивного интерфейса с использованием медиазапросов CSS. | 2 |  |
|  | 11.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 12.11.2024 | Знакомство с PostgreSQL. | 2 |  |
|  | 12.11.2024 | Знакомство с PostgreSQL. | 2 |  |
|  | 12.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 13.11.2024 | Создание базы данных в СУБД PostgreSQL. | 2 |  |
|  | 13.11.2024 | Создание базы данных в СУБД PostgreSQL. | 2 |  |
|  | 13.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 14.11.2024 | Подключение PostgreSQL через appsettings.json. Определение сущностей и их атрибутов в C# для базы данных с использованием Entity Framework. | 2 |  |
|  | 14.11.2024 | Подключение PostgreSQL через appsettings.json. Определение сущностей и их атрибутов в C# для базы данных с использованием Entity Framework. | 2 |  |
|  | 14.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 15.11.2024 | Разработка базовых операций CRUD для всех сущностей. Добавление моделей. | 2 |  |
|  | 15.11.2024 | Разработка базовых операций CRUD для всех сущностей. Добавление моделей. | 2 |  |
|  | 15.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 16.11.2024 | Разработка сервисов для авторизации и аутентификации. | 2 |  |
|  | 16.11.2024 | Разработка сервисов для авторизации и аутентификации. | 2 |  |
|  | 16.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 18.11.2024 | Подключение ASP.NET Identity для управления доступом пользователей и роли. | 2 |  |
|  | 18.11.2024 | Подключение ASP.NET Identity для управления доступом пользователей и роли. | 2 |  |
|  | 18.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 19.11.2024 | Использование C#, FluentValidation и JavaScript для валидации вводимых данных при авторизации.  Оформление отчетной документации. | 2 |  |
|  | 19.11.2024 | Использование C#, FluentValidation и JavaScript для валидации вводимых данных при регистрации. | 2 |  |
|  | 19.11.2024 | Использование C#, FluentValidation и JavaScript для валидации вводимых данных при регистрации. | 2 |  |
|  | 20.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 20.11.2024 | Реализация аутентификации пользователя при регистрации. | 2 |  |
|  | 20.11.2024 | Реализация аутентификации пользователя при регистрации. | 2 |  |
|  | 21.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 21.11.2024 | Интеграция сторонних провайдеров, таких как Google, для регистрации и входа пользователей. | 2 |  |
|  | 21.11.2024 | Интеграция сторонних провайдеров, таких как Google, для регистрации и входа пользователей. | 2 |  |
|  | 22.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 22.11.2024 | Frontend разработка страницы с профилем пользователя: написание вкладки с запросами. | 2 |  |
|  | 22.11.2024 | Frontend разработка страницы с профилем пользователя: написание вкладки с запросами. | 2 |  |
|  | 23.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 23.11.2024 | Frontend разработка страницы с профилем пользователя: написание вкладки с корзиной. | 2 |  |
|  | 23.11.2024 | Frontend разработка страницы с профилем пользователя: написание вкладки с корзиной. | 2 |  |
|  | 25.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 25.11.2024 | Реализация функционала редактирования профиля и обновления данных пользователя. | 2 |  |
|  | 25.11.2024 | Реализация функционала редактирования профиля и обновления данных пользователя. | 2 |  |
|  | 26.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 26.11.2024 | Frontend разработка страницы со странами. | 2 |  |
|  | 26.11.2024 | Frontend разработка страницы со странами. | 2 |  |
|  | 27.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 27.11.2024 | Frontend разработка страницы с турами: реализация фильтров и сортировки данных на страницах с турами. | 2 |  |
|  | 27.11.2024 | Frontend разработка страницы с турами: реализация фильтров и сортировки данных на страницах с турами. | 2 |  |
|  | 28.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 28.11.2024 | Разработка сервисов для реализации фильтров и сортировки данных на страницах с турами. | 2 |  |
|  | 28.11.2024 | Разработка сервисов для реализации фильтров и сортировки данных на страницах с турами. | 2 |  |
|  | 29.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 29.11.2024 | Frontend разработка страницы с турами: секция отображения туров. | 2 |  |
|  | 29.11.2024 | Frontend разработка страницы с турами: секция отображения туров. | 2 |  |
|  | 30.11.2024 | Оформление отчетной документации. | 2 |  |
|  | 30.11.2024 | Frontend разработка конкретная страница с туром. | 2 |  |
|  | 30.11.2024 | Frontend разработка конкретная страница с туром. | 2 |  |
|  | 02.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 02.12.2024 | Реализация поискового механизма с использованием JS. | 2 |  |
|  | 02.12.2024 | Реализация поискового механизма с использованием JS. | 2 |  |
|  | 03.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 03.12.2024 | Разделение прав доступа для администраторов, обычных пользователей. | 2 |  |
|  | 03.12.2024 | Разделение прав доступа для администраторов, обычных пользователей. | 2 |  |
|  | 04.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 04.12.2024 | Разработка интерфейса для администрирования пользователей и управление запросами. | 2 |  |
|  | 04.12.2024 | Разработка интерфейса для администрирования пользователей и управление запросами. | 2 |  |
|  | 05.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 05.12.2024 | Разработка интерфейса для администрирования странами и турами. | 2 |  |
|  | 05.12.2024 | Разработка интерфейса для администрирования странами и турами. | 2 |  |
|  | 06.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 06.12.2024 | Разработка функций для управления контентом сайта, таких как создание новых туров. | 2 |  |
|  | 06.12.2024 | Разработка функций для управления контентом сайта, таких как создание новых туров. | 2 |  |
|  | 07.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 07.12.2024 | Разработка модульных тестов на xUnit для проверки корректности работы логики приложения при аутентификации. | 2 |  |
|  | 07.12.2024 | Разработка модульных тестов на xUnit для проверки корректности работы логики приложения при аутентификации. | 2 |  |
|  | 09.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 09.12.2024 | Разработка модульных тестов на xUnit для проверки корректности работы логики приложения для сервисов, связанных с профилем. | 2 |  |
|  | 09.12.2024 | Разработка модульных тестов на xUnit для проверки корректности работы логики приложения для сервисов, связанных с профилем. | 2 |  |
|  | 10.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 10.12.2024 | Разработка модульных тестов на xUnit для проверки корректности работы логики приложения для сервисов, связанных с турами. | 2 |  |
|  | 10.12.2024 | Разработка модульных тестов на xUnit для проверки корректности работы логики приложения для сервисов, связанных с турами. | 2 |  |
|  | 11.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 11.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 11.12.2024 | Оформление отчетной документации. | 2 |  |
|  | 12.12.2024 | Оформление индивидуального проекта и презентации к защите работы. | 2 |  |
|  | 12.12.2024 | Оформление индивидуального проекта и презентации к защите работы. | 2 |  |
|  | 12.12.2024 | Дифференцированный зачет. | 2 |  |

Содержание объемов выполненных работ подтверждаю

Руководитель практики

от техникума (предприятия) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_С.Г. Патеров\_\_/

подпись Ф.И.О.

**01.11.2024**

**Тема:** Роль практического обучения при формировании компетенций обучающихся. Инструктаж по технике безопасности и правилам поведения на учебной практике. Создание проекта на основе шаблона MVC. Оформление отчетной документации.

**Ход Работы:**

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

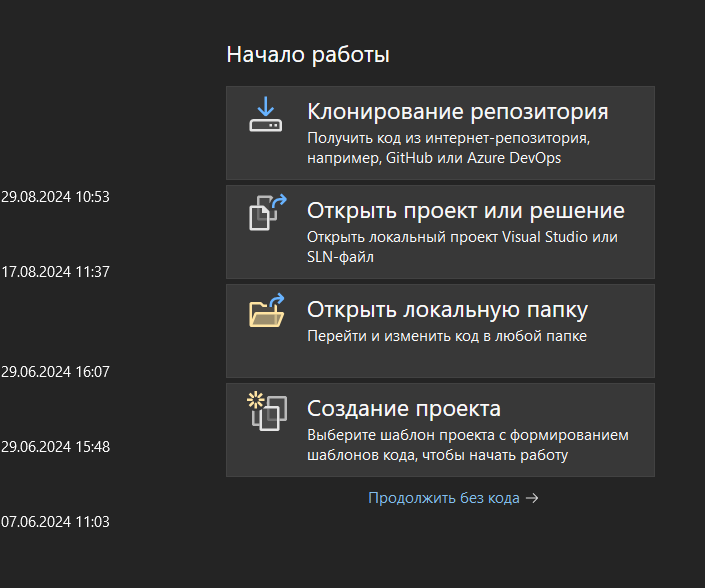


Рисунок 1 – Создание проекта

**Шаг 2.** 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*.

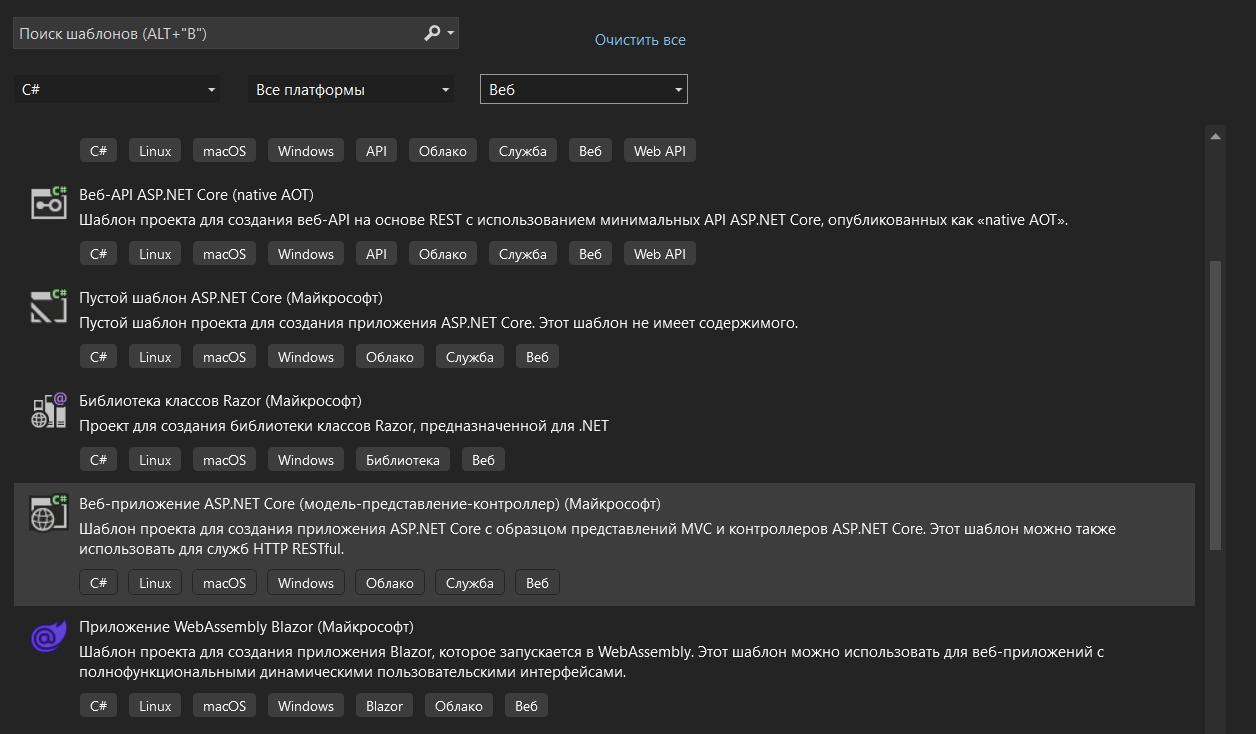


Рисунок 2 – Шаблоны

**Шаг 3.**  *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.

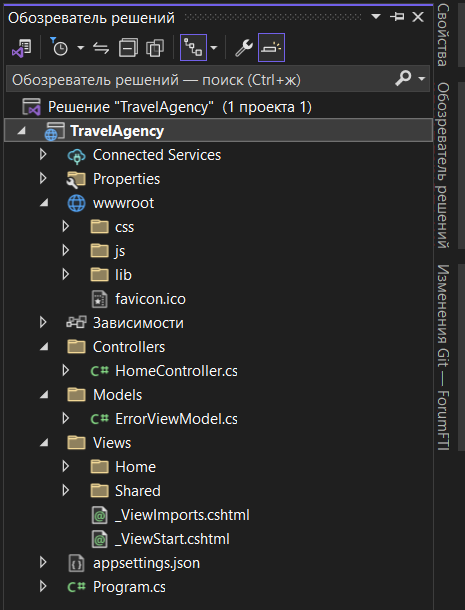


Рисунок 3 – Обозреватель решений

**Шаг 4.** 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.

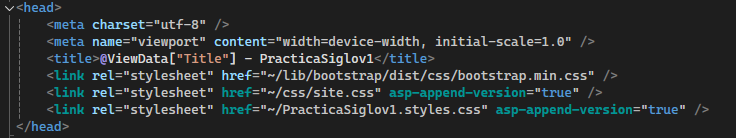


Рисунок 4 – тег <head>

<*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.

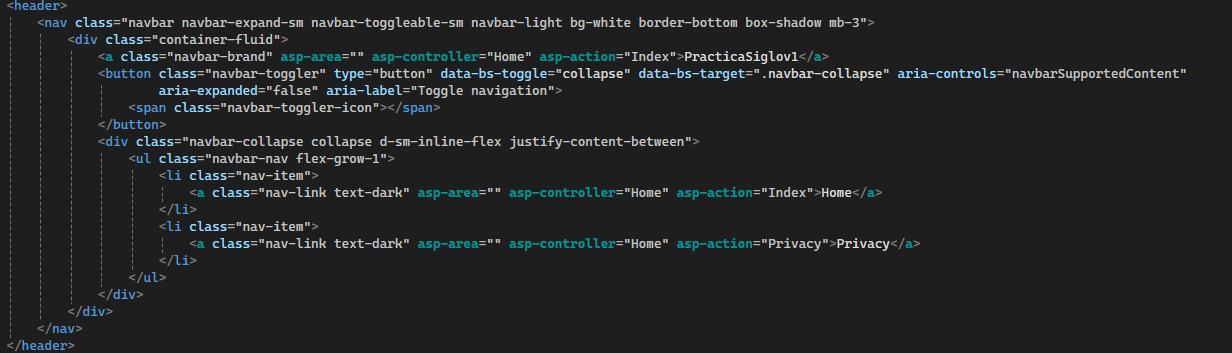


Рисунок 5 – тег <header>

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.

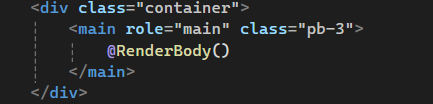


Рисунок 6 – функция *@RenderBody*()

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.

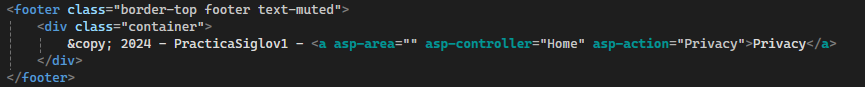


Рисунок 7 – тег <*footer*>

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*>.

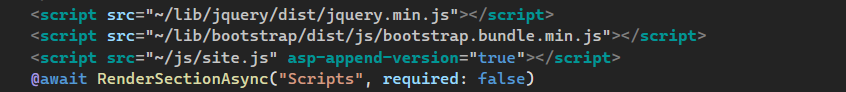


Рисунок 8 – подключение скриптов

**Шаг 5.** 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*.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXeFuPF3Qr9MzC3AiEd8WMh79bEOYUYoJ2riWK0I9t5x_8ME8FjFHVA8NONU5XDaUyapZgsYgbn6U7jiqY9U0wD-Ebevd8kuLkeEpviT5tKkxXPLt-MNy4x8SvzrEhKYn-P_y5k3FZrlWl25ZkgQrKMVUviP0NZBVs5tSPW80nigxsk?key=dVzRd2Dt4PuAbWPPkpYmh6OO

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.

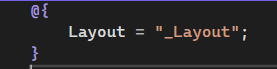


Рисунок 10 – содержимое файла *\_ViewStart.cshtml*

**Шаг 6.** 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.

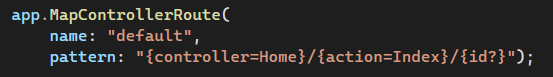


Рисунок 11 – фрагмент кода из файла *Program.cs*

В результате создания проекта при запуске проекта открывается страница браузера показанная на рисунке 12

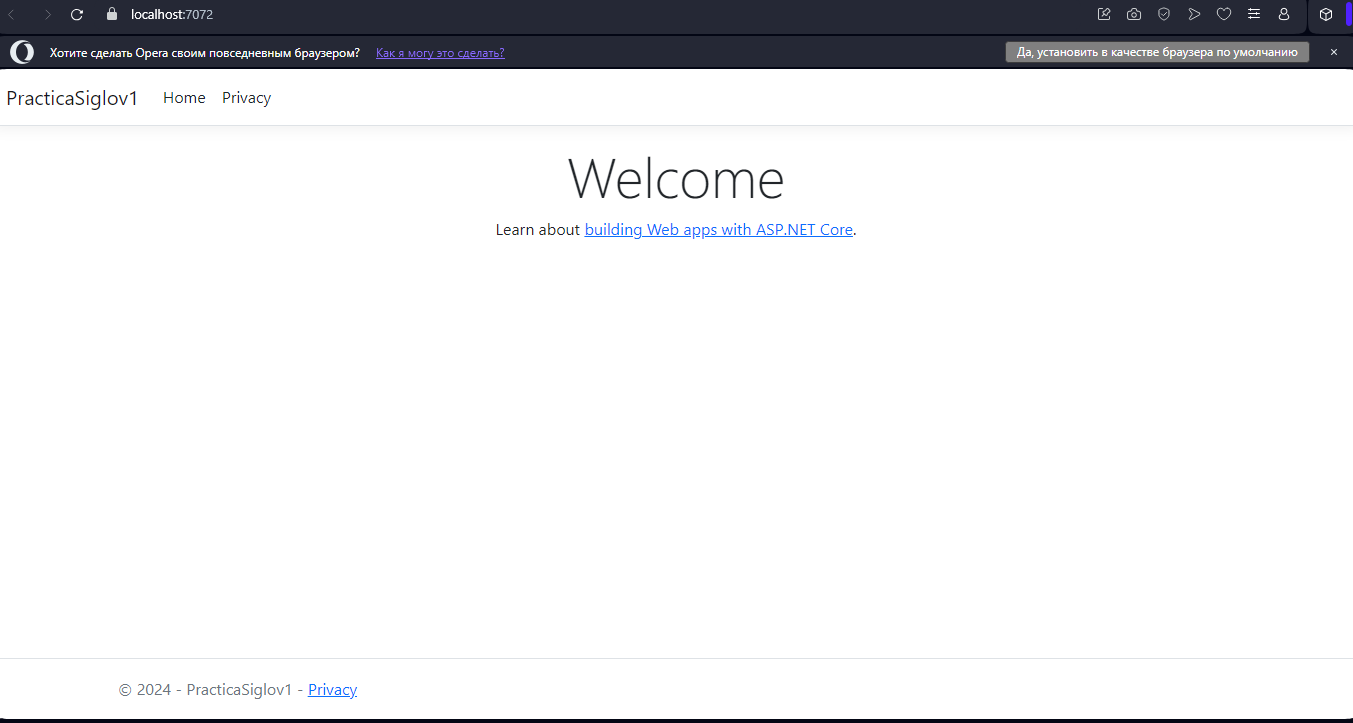


Рисунок 12 – результат работы кода при создании

**Вывод:** Создал проект на основе MVC ознакомился с архитектурой проекта.

**02.11.2024**

**Тема:** Использование Git для контроля версий. Оформление отчетной документации.

**Ход работы:**

**Шаг 1.** Во время регистрации вам будет предложено проверить адрес электронной почты. Без проверенного адреса электронной почты вы не сможете выполнить некоторые основные задачи *GitHub*, например, создание репозитория. Для начала создадим репозиторий для этого в правом верхнем углу открываем меню пользователя и выбираем пункт *«Your repositories».*

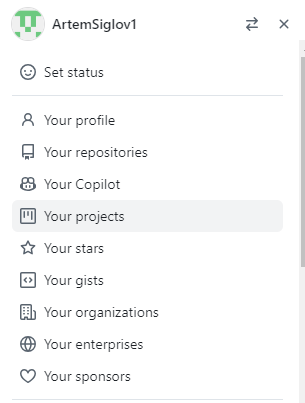


Рисунок 12 – ваш профиль

**Шаг 2.** На странице с репозиториями нажимаем на кнопку *«New».* Откроется страница с созданием репозитория. Задаем название репозитория *TravelAgency* как показано на рисунке 13. Остальные поля и пункты можно оставить в исходном положении.

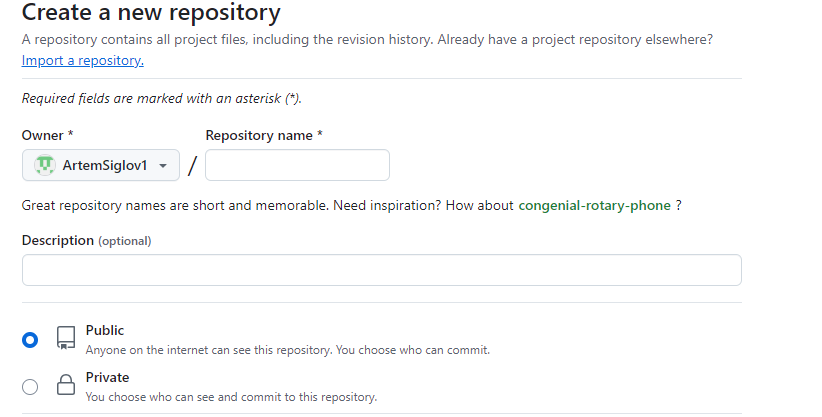
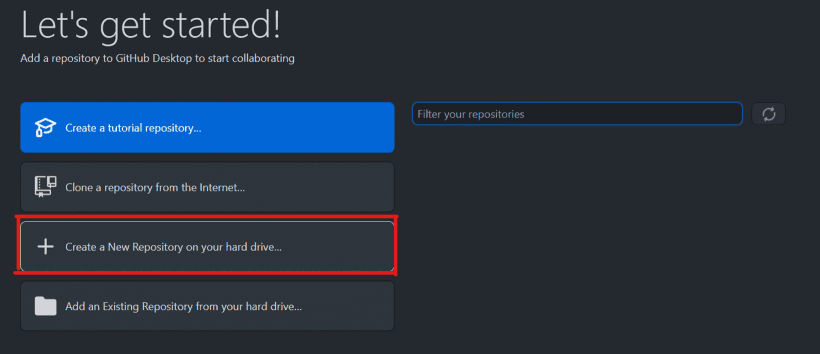


Рисунок 13 – создание удаленного репозитория

**Шаг 3.** После создания, удаленного репозитория необходимо создать локальный репозиторий. Для того что бы разобраться как работать с *GitHub Desktop* можно перейти на сайт [*https://htmlacademy.ru/blog/git/github-desktop*](https://htmlacademy.ru/blog/git/github-desktop)либо просмотреть информацию ниже, она взята с этого сайта.

После первого входа в *GitHub Desktop* вас попросят ввести ваши [логин и пароль от *GitHub.com*](https://htmlacademy.ru/blog/git/github-account). После этого у вас появится доступ ко всем репозиториям, сохранённым в профиле.

Создание репозитория: если вы никогда не пользовались *GitHub*-ом, нужно будет создать репозиторий для работы над проектом. На главном экране *GitHub Desktop* выберите пункт *«Create a New Repository on your hard drive».*

 Рисунок 14 – начало работы

**Шаг 4.** Нужно будет ввести название репозитория, его описание и выбрать папку на компьютере, куда будут сохраняться файлы.

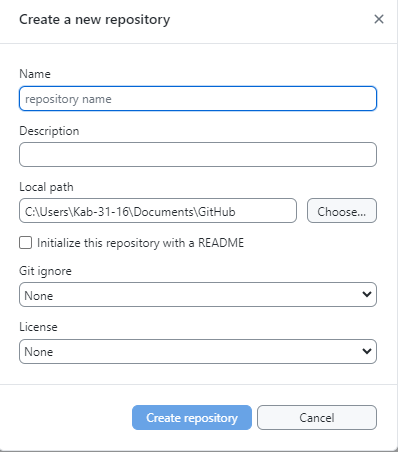


Рисунок 15 – создание локального репозитория

**Шаг 5.** После этого нажимаем на *Create repository*, ждём несколько секунд и готово – на компьютере появилась папка, которой можно пользоваться для разработки вашего проекта.

Клонирование репозитория: если у вас уже какой-нибудь репозиторий на Гитхабе, его можно клонировать. Клонировать – это скачать все файлы к себе на компьютер, чтобы можно было их изменять и потом загружать обратно.

Выбираем *Add -> Clone Repository...*

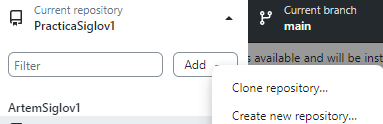


Рисунок 16 – клонирование репозитория

**Шаг 6.** В открывшемся окне выбираем один из имеющихся репозиториев. В данном случае он называется *zaverstai,* но у вас может быть любой другой.

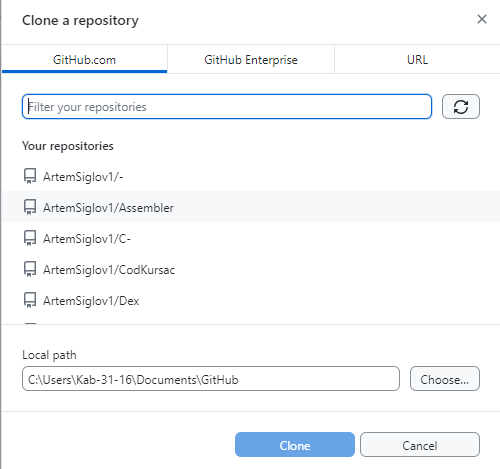


Рисунок 17 – выбор репозитория

После этого файлы репозитория начнут скачиваться – если их много, то это займет некоторое время.



Рисунок 18 – скачивание файлов

**Шаг 7.** Работа с репозиторием. Меняем файлы и сохраняем обратно: вне зависимости от того, создали вы репозиторий или клонировали его, так выглядит *GitHub Desktop* с открытым репозиторием, в котором мы пока ничего не меняли.

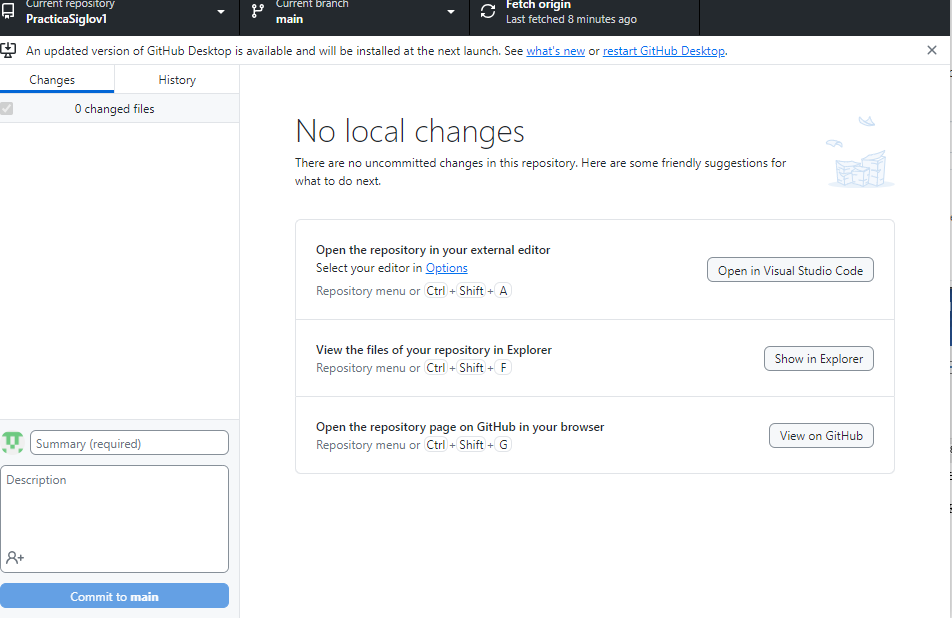


Рисунок 19 – вид приложения с открыты репозиторием

*Слева* – *поле для измененных файлов, справа* – *служебная информация. Слева снизу* – *поле для коммитов.*

**Шаг 8.** Давайте добавим какой-нибудь файл. Например, я добавил в локальный репозиторий (скопировал в папку) файл index.html. Вы можете добавить текстовый файл.

Сразу после добавления или изменения файла в окне GitHub Desktop будет видно, что изменилось – если мы добавили целый новый файл, то все строчки будут с плюсиками и зелёные. Это значит, что они были добавлены в файл и GitHub Desktop раньше их никогда не видел.

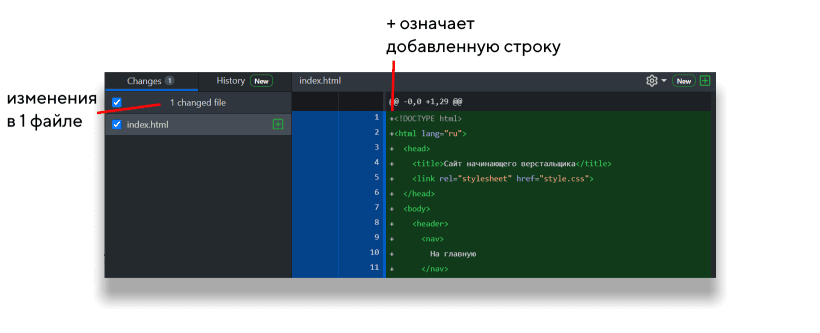


Рисунок 20 – вид при добавлении файла

**Шаг 9.** Загружаем новый репозиторий на *GitHub*: после того, как мы добавили какой-то код в свежесозданный репозиторий, нужно сделать коммит, то есть зафиксировать все сохранённые изменения и дать им название. Текст должен быть лаконичным и в то же время сообщать о том, что делает коммит. Например, «добавляет имя наставника в *Readme*», «вводит функцию сортировки изображений», «правит ошибку в поиске городов на карте». Вводим имя жмём большую синюю кнопку *«Commit to main»*

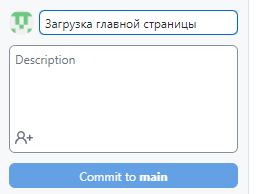


Рисунок 21 – создание коммита

**Шаг 10.** Изменения, которые мы внесли и сохранили, пока локальны. Их нужно послать на *GitHub.* Чтобы опубликовать свежесозданный репозиторий на *GitHub*, нажмите *Publish repository*.

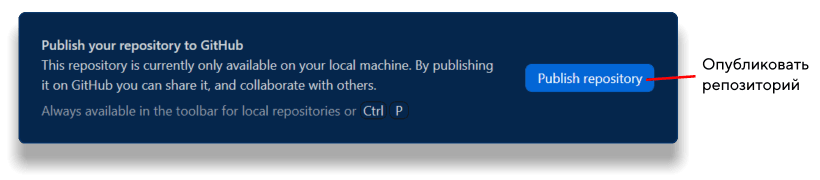


Рисунок 22 – опубликовать репозиторий

**Шаг 11.** Появится подтверждение о публикации репозитория – проверяем название и описание, если нужно, ставим галочку о том, что код приватный, и публикуем.



Рисунок 23 – подтверждение о публикации репозитория

Готово – после этого репозиторий появится в вашем профиле на *GitHub. com.*

**Шаг 12.** Добавляем код и коммитим изменения: репозиторий создан и загружен на GitHub, теперь нужно добавить немного кода.

Когда вы допишете код в файлы, которые находятся в репозитории, вы сможете просмотреть все их изменения в окне *GitHub Desktop*. Вот здесь, например, мы изменили «второй» на «третий» в тексте страницы – и изменения сразу видны, можно проверить, что всё исправленное будет загружено.

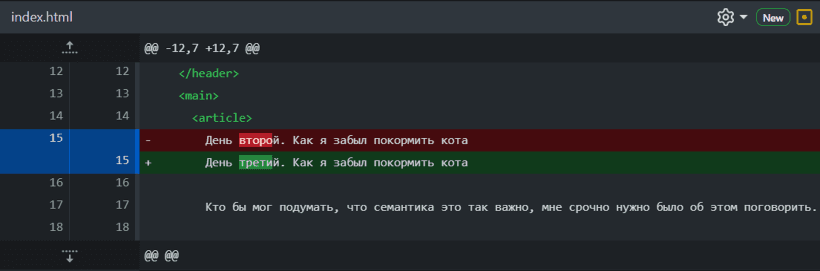


Рисунок 24 – просмотр исправлений

**Шаг 13.** Дальше действуем по проверенной схеме – коммитим изменения.

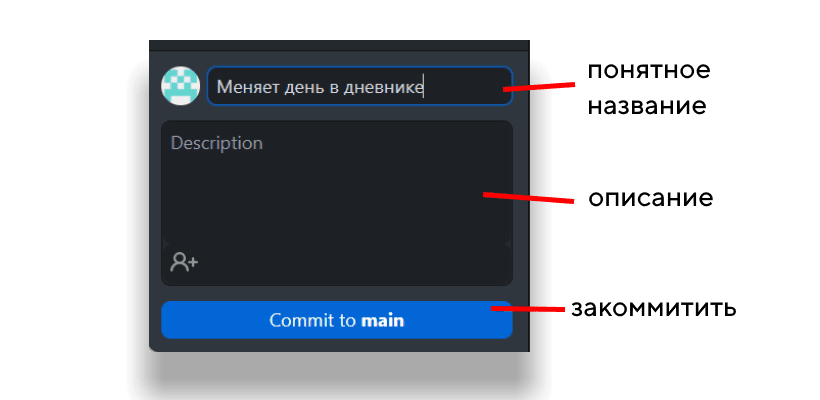


Рисунок 25 – коммит изменений

**Шаг 14.** В центре главного экрана появится предложение запушить коммит в удалённый репозиторий. Соглашаемся и жмём *Push origin*.

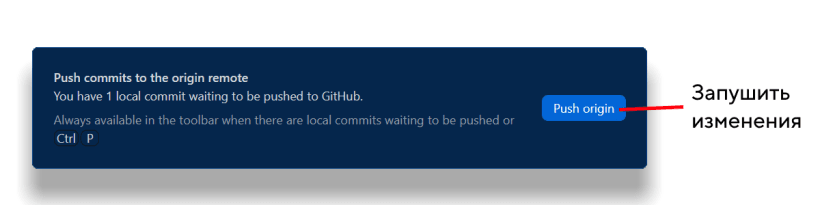


Рисунок 26 – отправка изменений

Готово! Теперь, если зайти на [*GitHub.com*](http://github.com/), в наш репозиторий, увидим изменённый файл, который мы только что отправили.

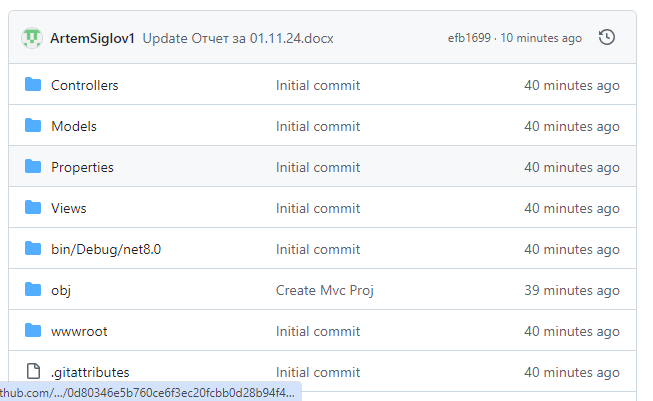


Рисунок 27 – проверка отправки изменений

**Вывод:** Научился и закрепил навыки создания или клонирования репозиториев, чтобы предоставить доступ к проекту в любом месте где есть доступ к интернету.

**04.11.24**

**Тема:** Определение структуры *MVC*-приложения, взаимодействие моделей, контроллеров и представлений. Оформление отчетной документации.

**Ход работы:**

**Шаг 1.** Изучим такие вопросы как: чистая архитектура, правило зависимостей, сущности, сценарии, интерфейс-адаптеры, фреймворки и драйверы, пересечение границ. Разберем может ли быть более четырех кругов, а также как данные пересекают границы.

**Шаг 2.** В обозревателе решений ПКМ кликаем на решение. Выбираем пункт «Добавить», «Создать папку решения». Как показано на рисунке 28.

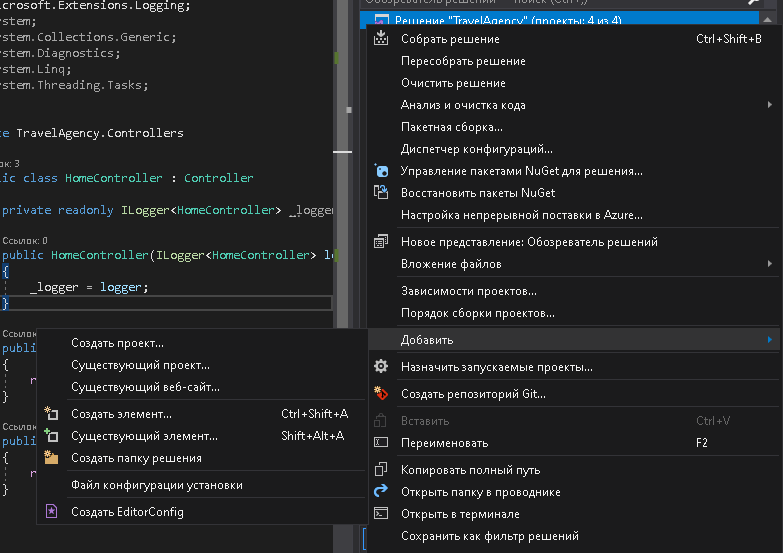


Рисунок 28 –­ создание папки решения.

**Шаг 3.** В обозревателе решений найдем созданную папку в предыдущем шаге. Кликаем на нее ПКМ. Выбираем пункт «Добавить», «Создать проект» проект должен быть типа «Библиотека классов» как показано на рисунке 29.

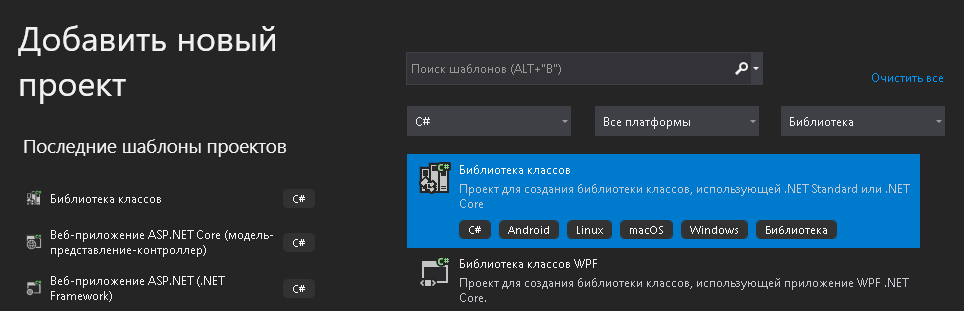


Рисунок 29 – создание проекта «Библиотека классов»

Добавить что получилось в результате

**Вывод:** Определил структуры *MVC*-приложения, взаимодействие моделей, контроллеров и представлений. Оформил отчетную документацию.

**05.11.24**

**Тема:** Frontend разработка основная страница: написание шапки сайта, секции «О нас».

Оформление отчетной документации.

**Ход работы:**

Продолжим с разметки главной страницы, а именно в файле *\_Layot.cshtml* приведем содержимое тега *<body>* к следующему виду в соответствии с рисунком 30. В данном фрагменте представлена разметка шапки сайта.



Рисунок 30 – разметка шапки сайта

На данный момент если вы запустите приложение, то оно будет выглядеть ужасно так как проект еще не содержит никаких *CSS* свойств. Что бы это исправить в папке «*css*» удалим имеющийся файл «*site.css*» и создадим новый файл *«shared.css».* Нажмем правой кнопкой мыши по папке в которую хотим добавить элемент и выберем Добавить/Создать элемент… как показано на рисунке 31.

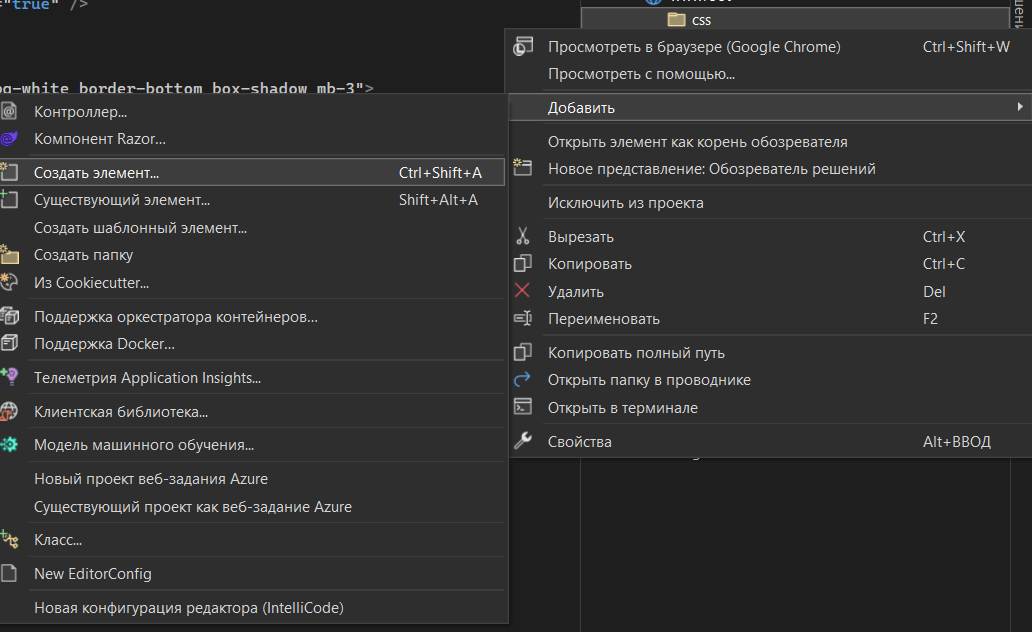


Рисунок 31 – открытие окна «добавление нового элемента»

В открывшемся окне необходимо выбрать в левом меню *C#/ASP.NET Core*/Веб/Содержимое/Таблица стилей. Выбранный файл нужно назвать *shared.css* как показано на рисунке 32.

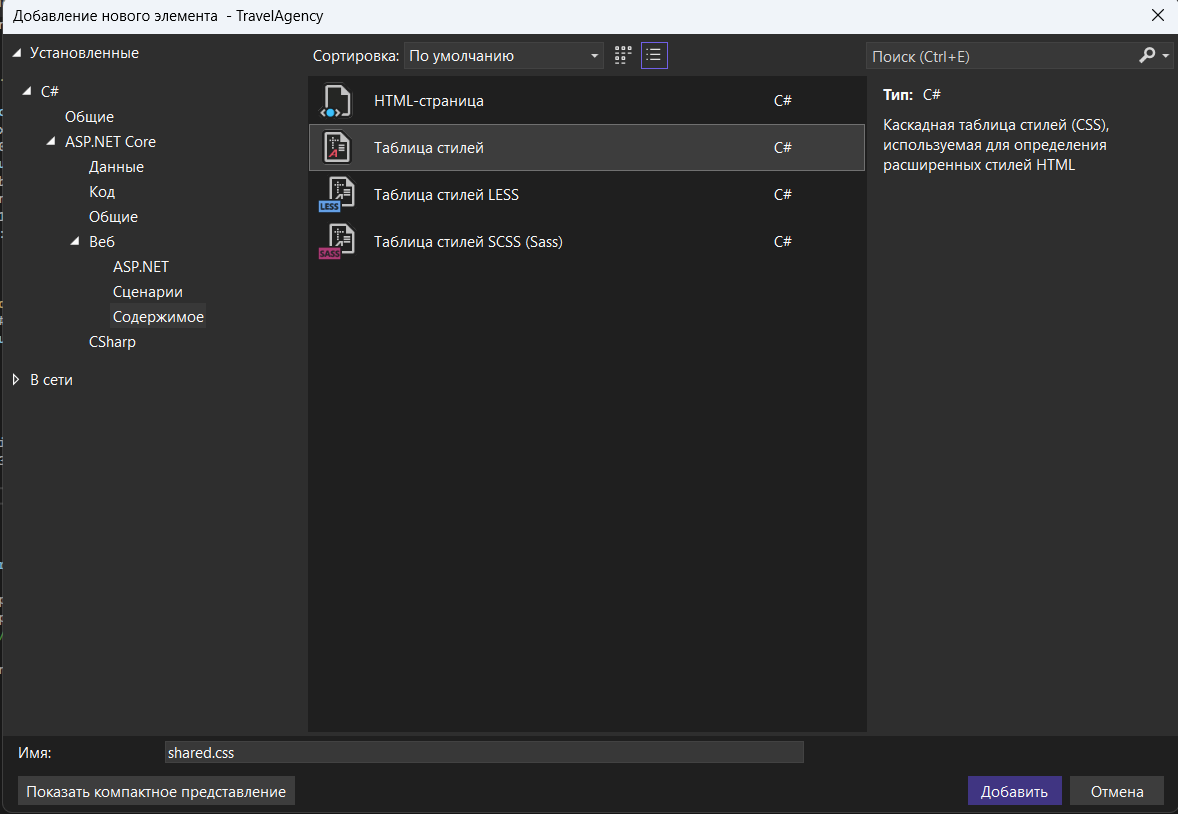


Рисунок 32 – добавление файла таблицы стилей

В появившемся файле прописываем пару стилей для тега *<html>, \*,\*::before,\*::after* и *body* в соответствиис рисунком 33.

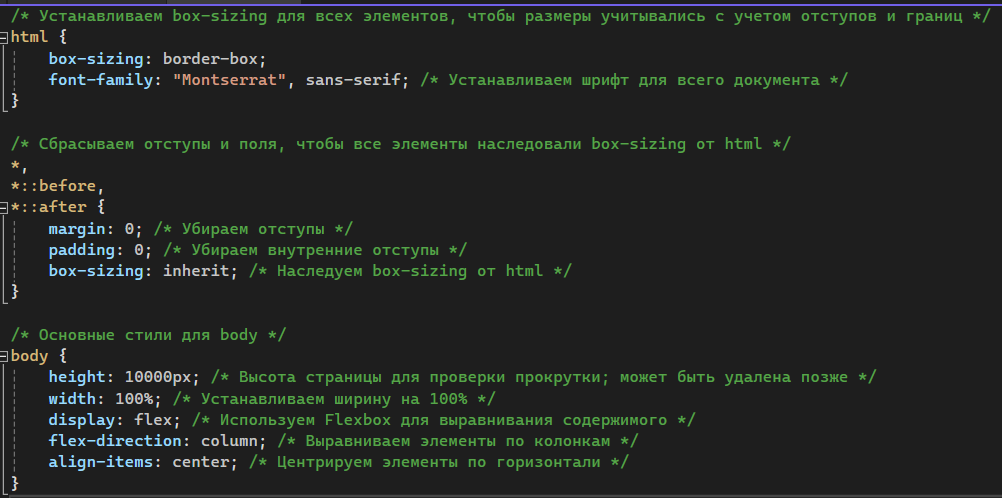


Рисунок 34 – стили *<html>*, *\*,\*::before,\*::after* и *body*

Далее прописываем стили для логотипа как показано на рисунке 35 в нашем случае это просто текст.

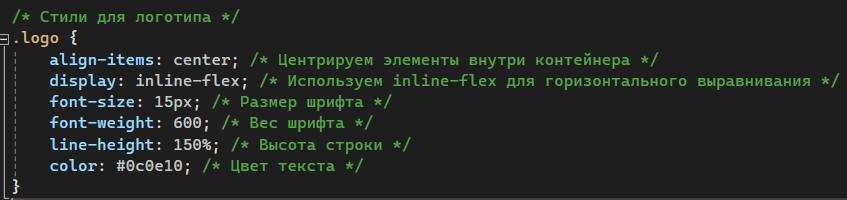


Рисунок 35 – стили для логотипа сайта

В данном проекте было принято решение разделить шапку сайта на две части. Это несет в себе исключительно эстетический смысл. В верхней части будут располагаться слева логотип, а справа две кнопки «Войти» и «Зарегистрироваться», далее по центру расположена картинка и под ней вторая часть шапки, где расположено меню. Для данного элемента сайта показаны стили на рисунке 36.

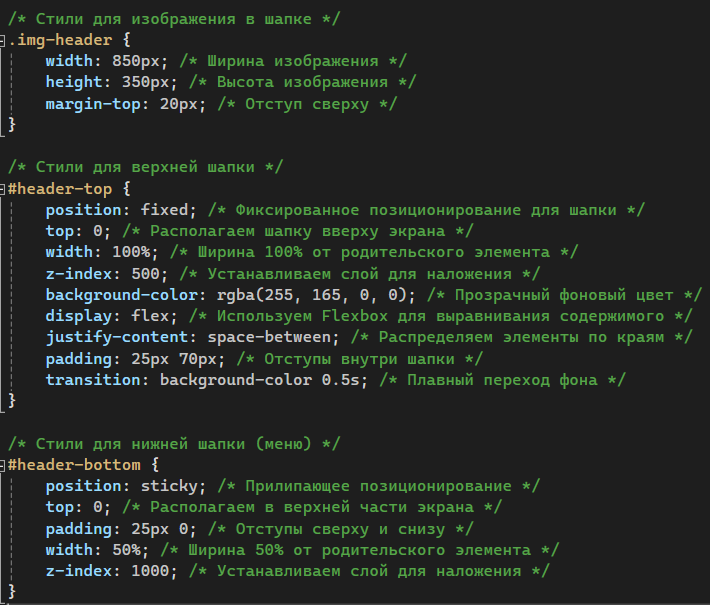


Рисунок 36 – стили шапки сайта

Ниже на рисунке 37 представлены стили для кнопок в верхней шапке.

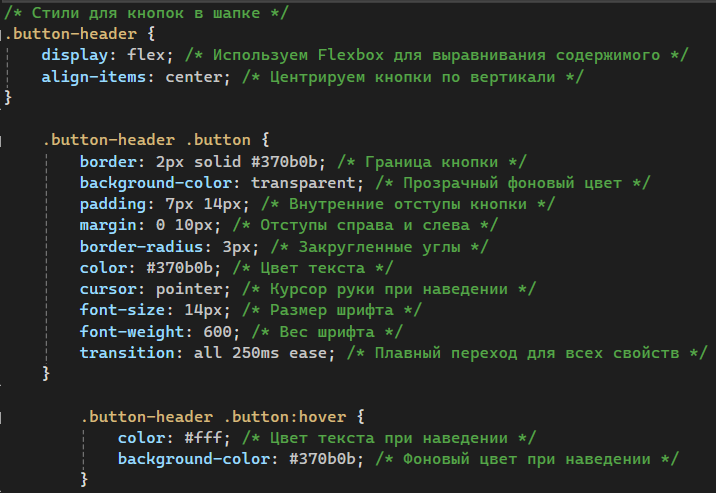


Рисунок 37 – стили для кнопок верхней шапки

Далее на рисунке 38 продемонстрированы стили для самого меню в нижней шапке.

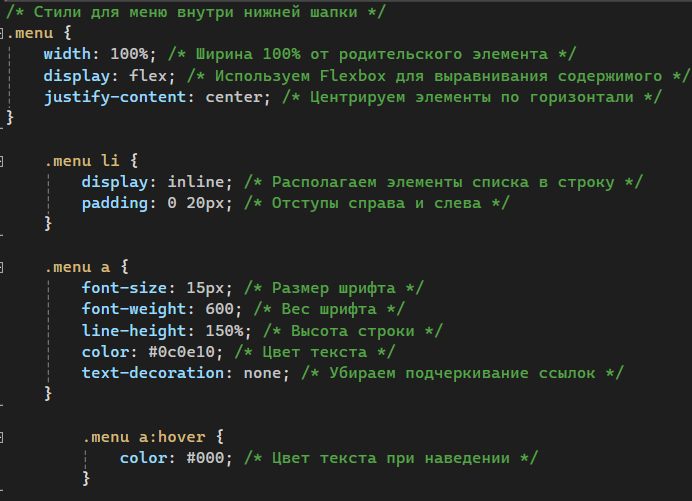


Рисунок 38 – стили для меню нижней шапки

Чтобы создать эффект, при котором шапка изначально прозрачная, а затем постепенно становится оранжевой при прокрутке, мы будем использовать *CSS* вместе с *JavaScript*. Для создания *js* файла откроем окно «Добавление нового элемента» и выберем *C#/ASP.NET Core*/Веб/Сценарии/Файл *JavaScript.* Выбранный файл необходимо назвать «*shared\_script.js»* как показано на рисунке 39.



Рисунок 39 – создание файла *JavaScript*

Далее в созданном файле прописываем следующий скрипт как показано на рисунке 40. Теперь если запустите проект, то при прокрутке шапка будет постепенно изменять свой цвет.

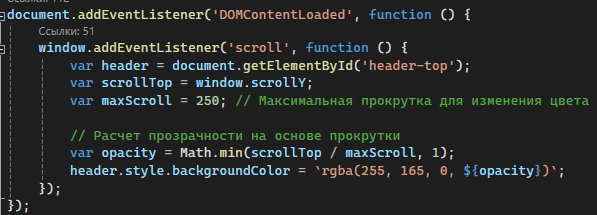


Рисунок 40 – скрипт для изменения цвета шапки с прозрачного на оранжевый

Определим стиль контейнера для тела сайта как показанно на рисунк 41.

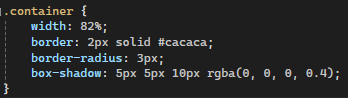


Рисунок 41 – стили для контейнера

Теперь можно перейти к реализации основного контента для этого создадим файл *SiteInformation.cshtml* в паке Home как показано на рисунке 42.

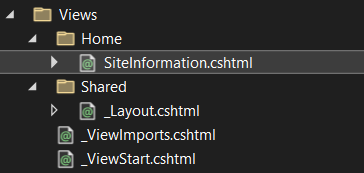


Рисунок 42 – структура папки Views на данном этапе разработки

Начнем с информации «О нас» пропишем разметку страницы как показано на рисунке как показано на рисунке 43. В выделенное место необходимо скопировать текст 1 из приложения А.

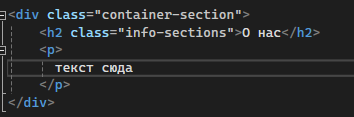


Рисунок 43 – «О нас» *html-*разметка

Применим стили к информации «О нас» в соответствии с рисунком 44

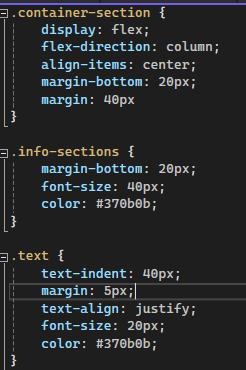
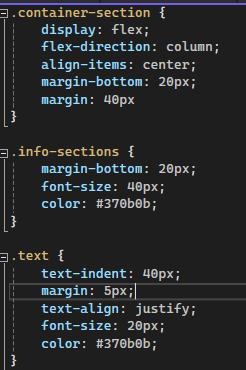


Рисунок 44 – «О нас» стили

В конце данной главы у вас должен получится вот такой результат.

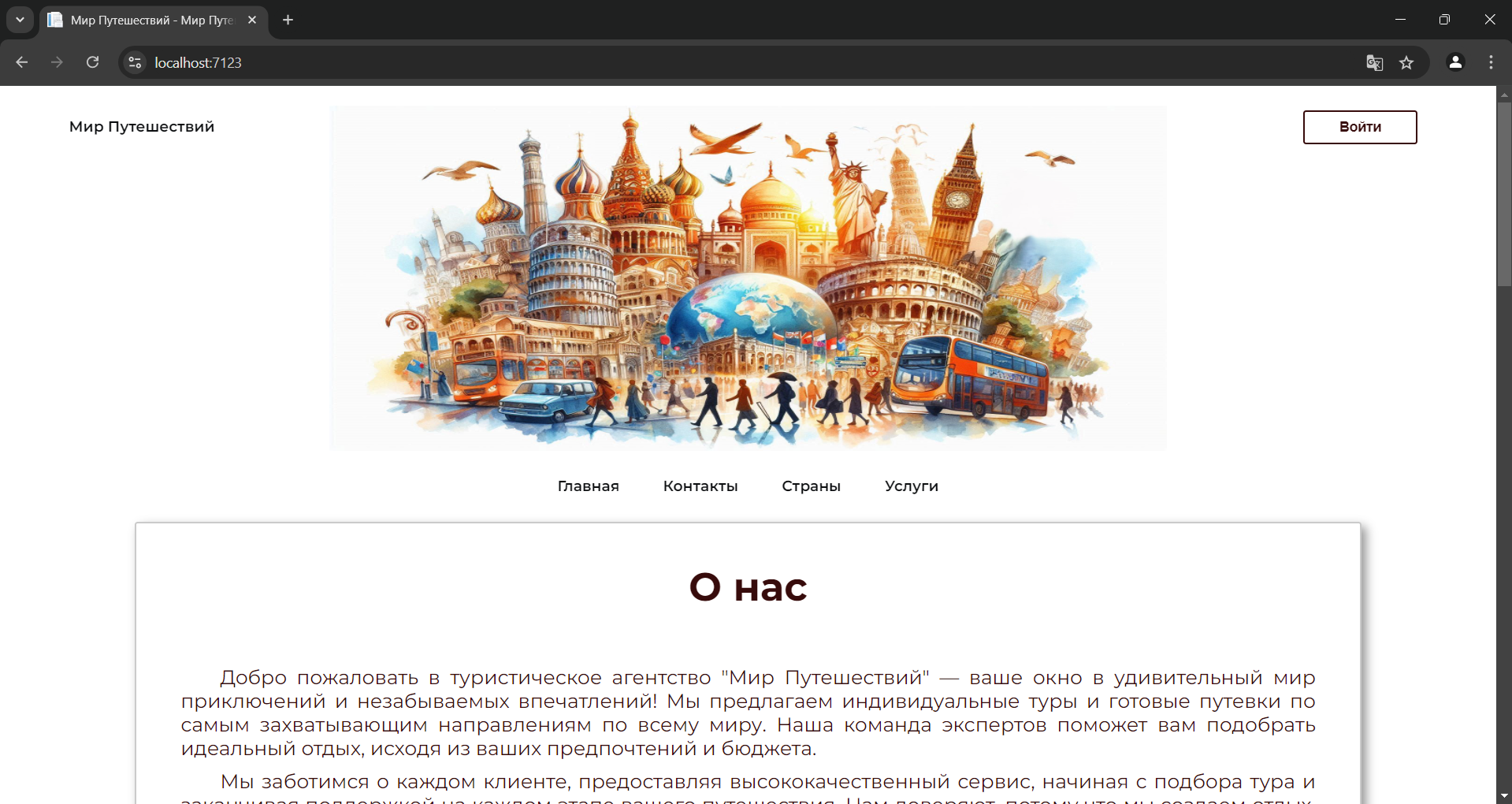


Рисунок 45 – Результат

**06.11.24**

**Тема:** Frontend разработка основная страница: написание секций «Услуги», «Контакты». Оформление отчетной документации.

**Ход работы:**

**Шаг 1.** Разработаем секцию с услугами в виде карточек с картинками и текстом как показано на рисунке 46. Карточки в данной секции будут расположены в контейнере по три. Всего карточек будет шесть для просмотра будет реализована прокрутка карточек при нажатии на стрелочку. Информацию для карточек ищите в приложении А текст 2. Картинки можете найти в интернете и добавить их в вашу папку с картинками, в пути укажите свое название картинок.  Прописываем в файл *SiteInformation.cshtml.*

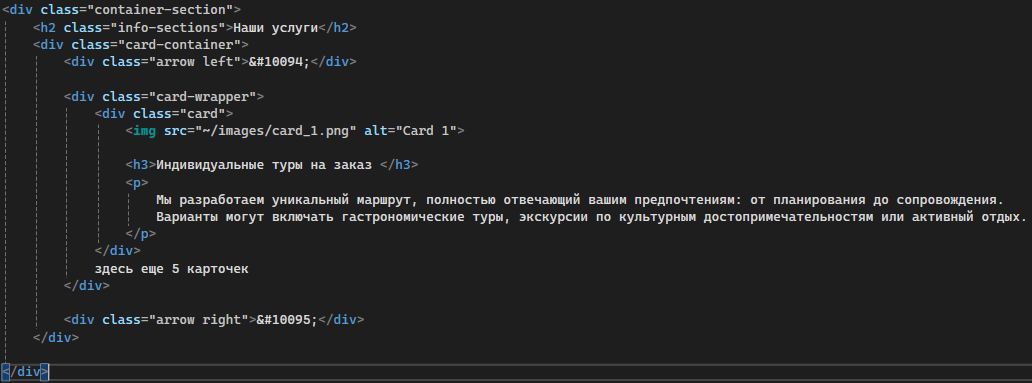


Рисунок 46 – «Наши услуги» разметка

**Шаг 2.** Ниже на рисунке 47 показаны стили для карточек с услугами. Прописываем все стили в главе в файл *home.css.* Чтобы не допускать опечатки копируем соответствующие названия классов из *SiteInformation.cshtml.* Не забываем ставить точки у классов.

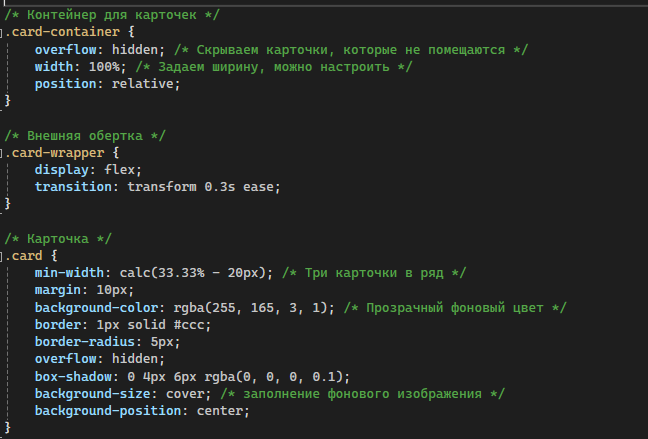


Рисунок 47 – «Наши услуги» стили для карточек

**Шаг 3.** Далее приведены стили для картинок, заголовка карточки и текста с описанием услуги как показано на рисунке 48.

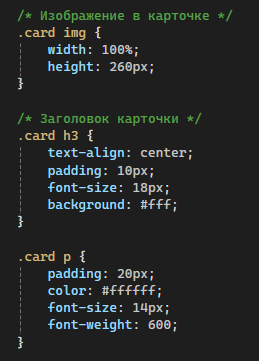


Рисунок 48 – «Наши услуги» стили для картинок и текста карточки

**Шаг 4.** Для прокрутки используются элементы с классом «*arrow».* Ниже на рисунке 49 показаны стили для стрелочек.

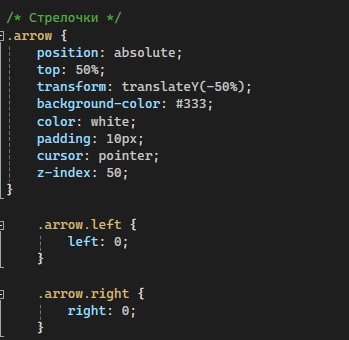


Рисунок 49 – «Наши услуги» стили для стрелочек

**Шаг 5.** Для того что бы при нажатии на стрелочку происходила прокрутка необходимо задействовать *JS* для этого создадим файл с названием «*home\_script.js»* и в него пропишем скрипт как показано на рисунке 50.



Рисунок 50 – скрипт для прокрутки

**Шаг 6.** Запустите проект и проверьте работу секции «Наши услуги»

Далее нам необходимо разместить карту на сайте. Как правило для это используют картографические плагины от внешних сервисов, например, Google Maps или Яндекс. Карты.

Воспользуемся картами от Яндекс по нескольким причинам:

* у Яндекс. Карт документация на русском языке;
* бесплатное подключение;
* лучше наполнены адреса для Приднестровья;
* процесс регистрации в сервисах Google слегка усложняется и требует под рукой данные кредитной карты, несмотря на бесплатное пользование сервисом. Но платёжные данные теперь являются обязательным условием пользования сервисом.

Конечно, без сети интернет плагин работать не будет. Поэтому для начала перейдём по ссылке <https://yandex.ru/map-constructor/>

В окне приветствия выбираем «Создать карту». Система попросит нас авторизоваться. Авторизируемся под своей учётной записью, или регистрируем новый ящик.

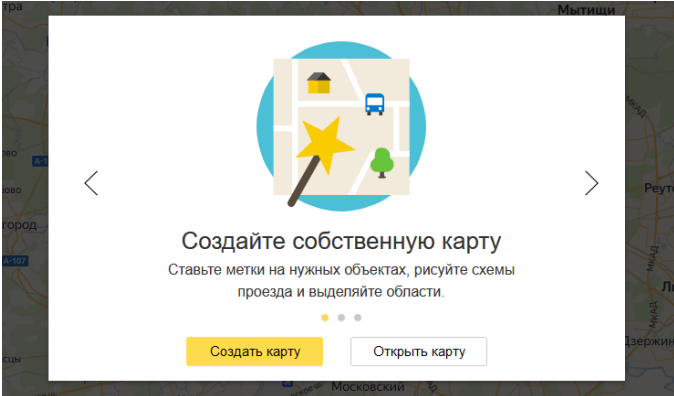


Рисунок 51 – Главная страница конструктора карт

**Шаг 7.** Здесь заполняем поля с названием карты, можем добавить описание, а

также добавить метки кликом по значку «Метки» в верхнем левом углу карты.

Перемещаем метку, настраиваем её внешний вид, а затем жмём «Сохранить и продолжить». Этот этап показан на рисунке 52.

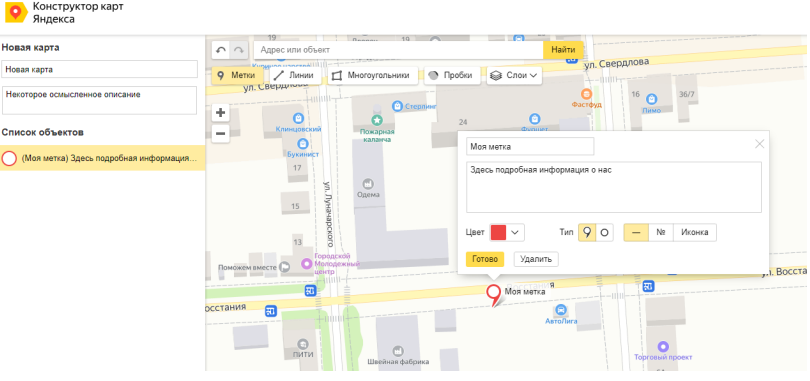


Рисунок 52 – Создание проекта Карты

**Шаг 8.** На следующем этапе, показанном на рисунке 53 находим на карте нужный город. Колесиком мыши устанавливаем в окне предпросмотра масштаб. В настройках оставляем активной интерактивную карту, установим галочку «Растянуть по ширине», а высоту установим 500px. Отлично, жмем «Получить код карты».

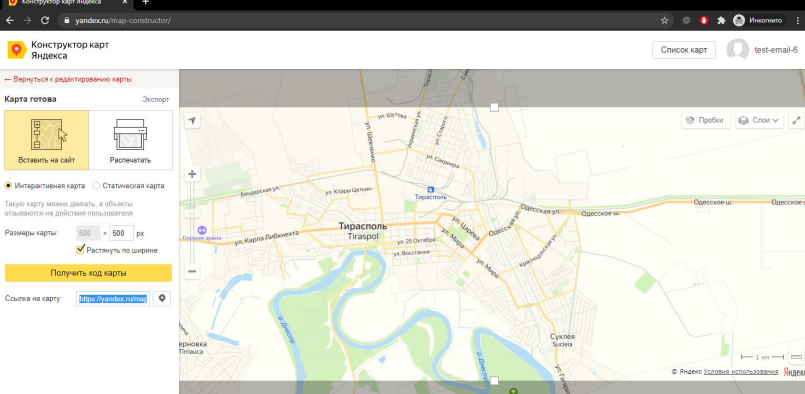


Рисунок 53 – Настройки карты

После чего появится окно, где можно будет скопировать код как показано на рисунке 54.

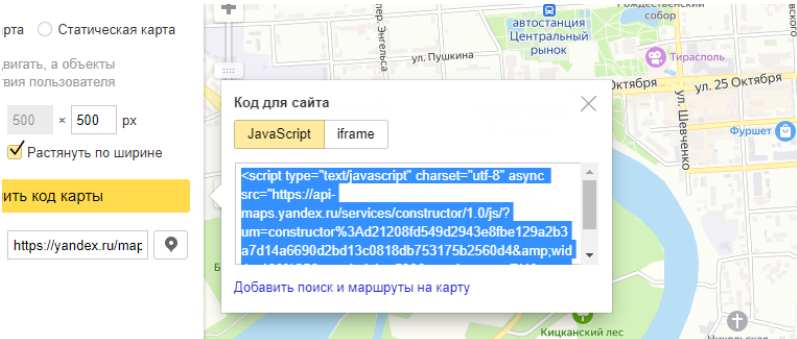


Рисунок 54 – Получение кода карты

**Шаг 9.** Теперь копируем данный код и вставим его в созданную секцию «Мы находимся» как показано на рисунках 55-56. Прописываем в файл *SiteInformation.cshtml.*

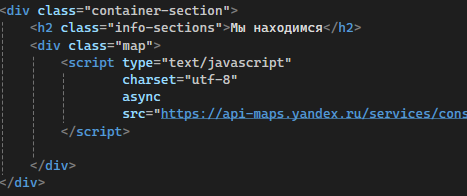


Рисунок 55 – секция контактов местоположение разметка

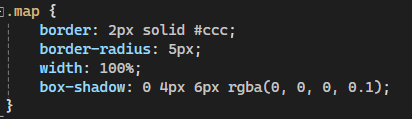


Рисунок 56 – секция контактов местоположение стили

**Шаг 10.** Добавим в верх секции «container-section» еще блок с кодом как показано на рисунке 72-73. Прописываем в файл *SiteInformation.cshtml.*

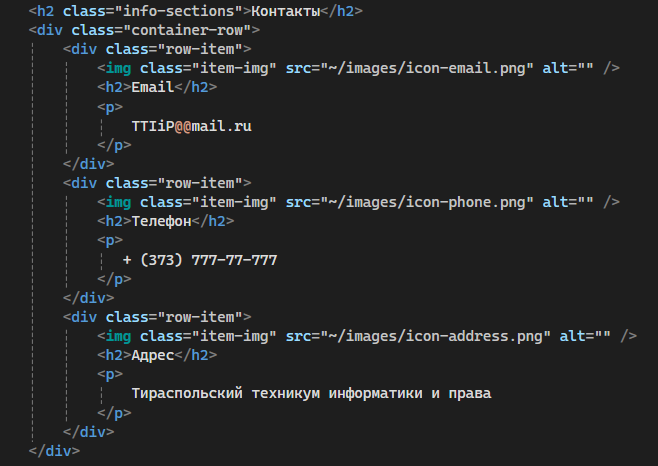


Рисунок 72 – контакты html/css

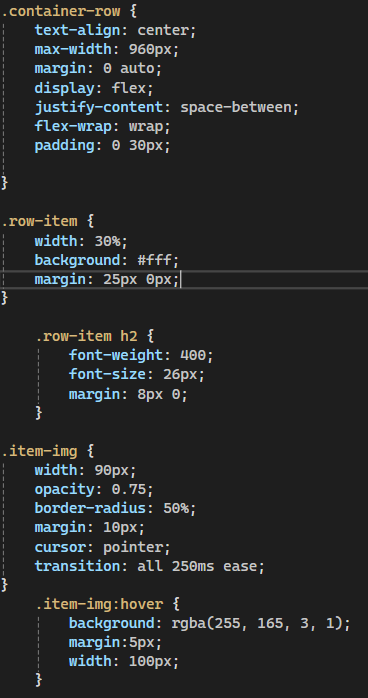
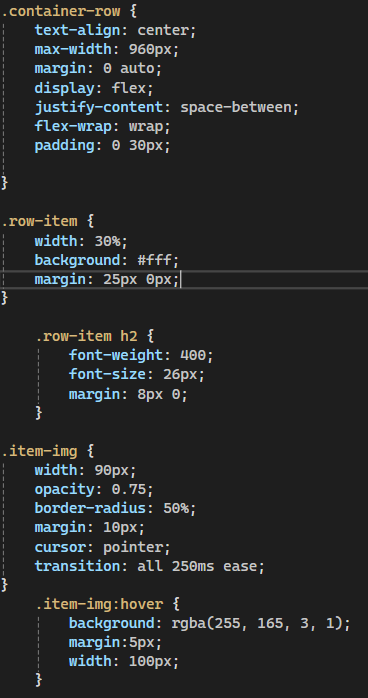


Рисунок 73 – контакты html/css

**Шаг 11.** В конце перед закрывающим тегом </body> стили будем указывать следующим образом используя *asp-append-version="true".* Это должно быть прописано для того чтобы браузер всегда использовал только новую версию файлов js.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXf42qR-bIjlWAw3qbShU6_2q_IYEyiG37UIp_uGEOZbLC-aHiNDcUNTyiyilpXT_zy-Zz2-er0wpSGPN4UmmG0EUjN5lSS85i_j1Fffj0tVRpD1I7Sarij5e0VPRnXTzo7knk94Ca17R0M6CFlk2VFNank?key=5ZQSu0k68aSBAR-kkCO7pcWI

В конце данной главы у вас должен получится вот такой результат.

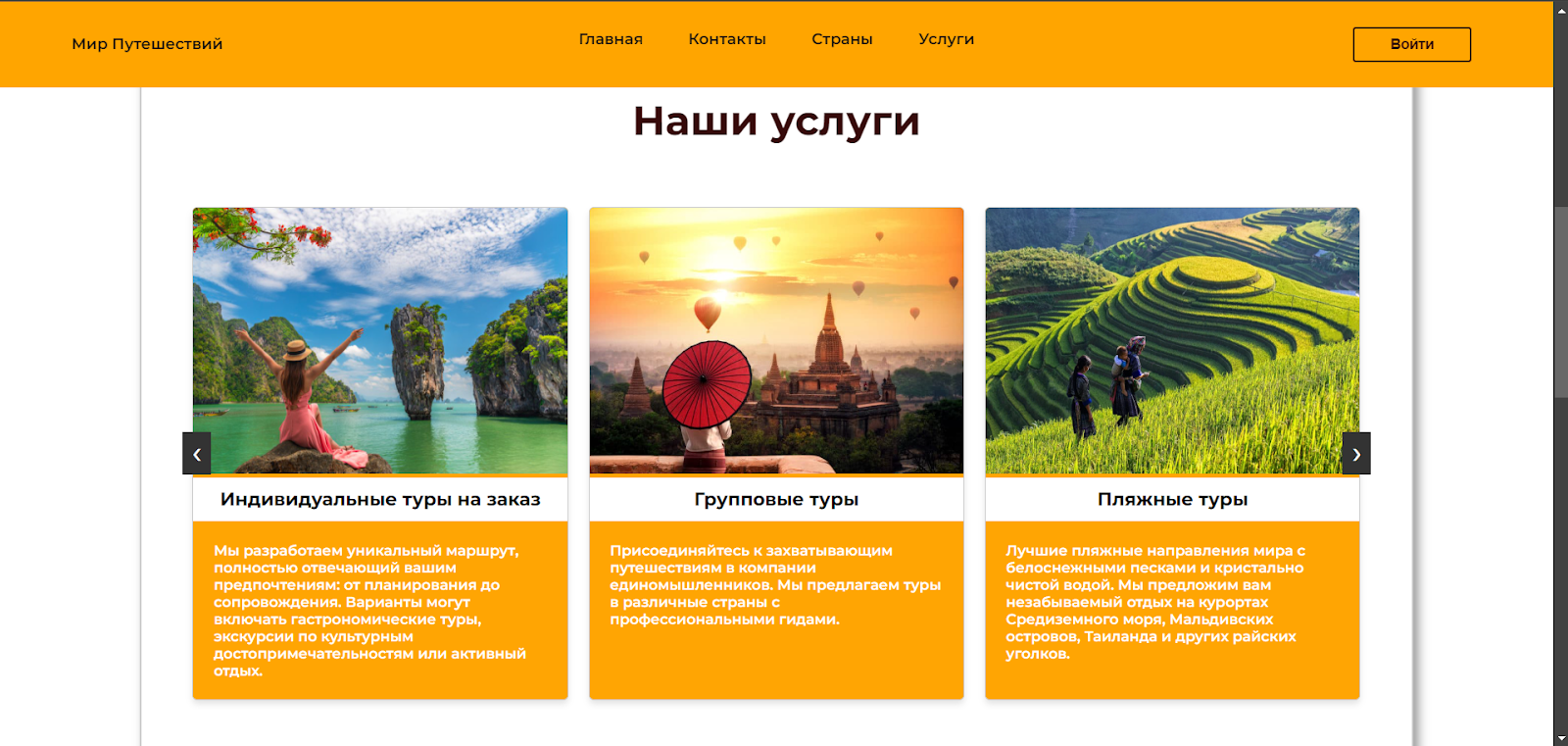
****

Рисунок 57 – Результат

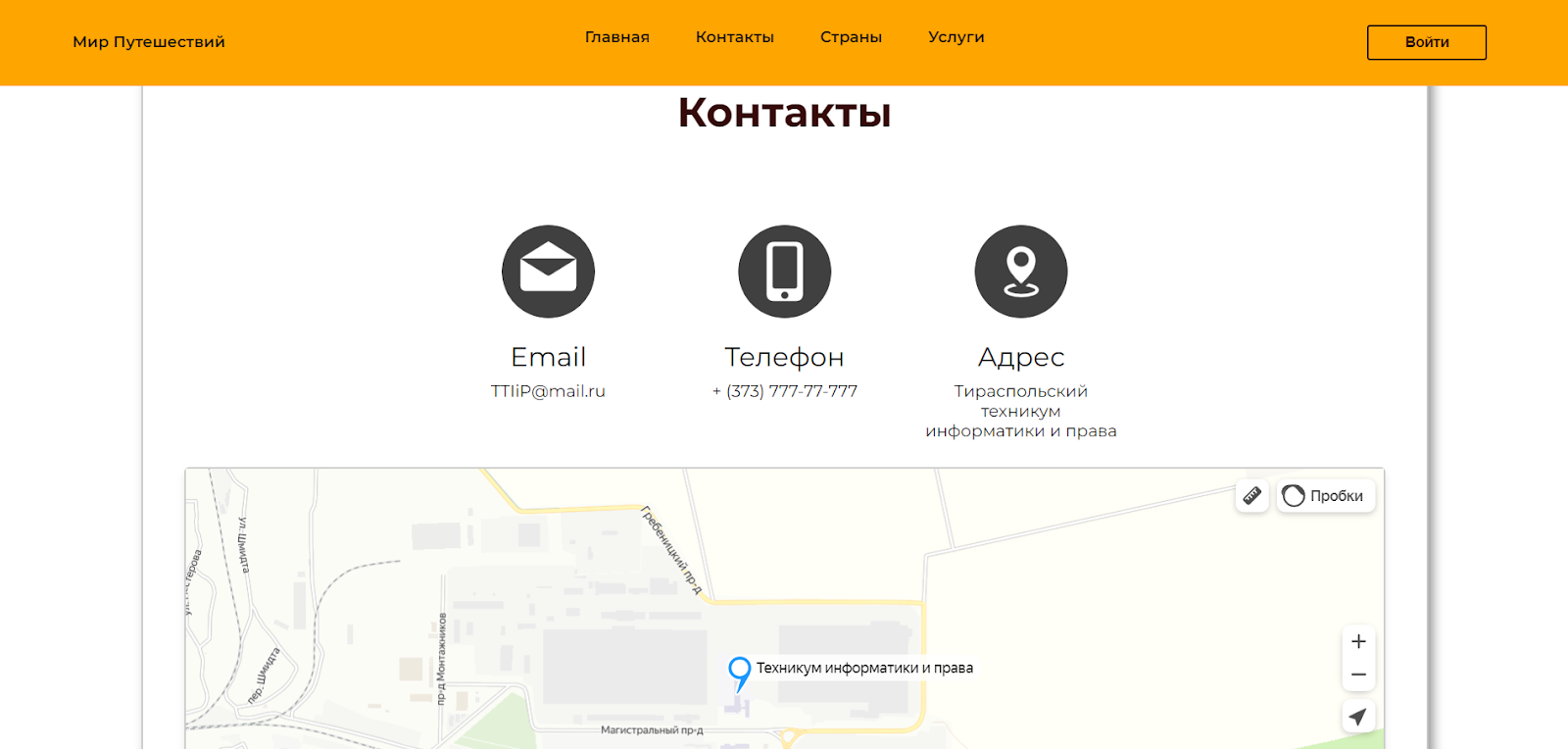
****

Рисунок 58 – Результат

**Вывод:** Выполнил Frontend разработку основной страницы: написал секций «Услуги», «Контакты».

**07.11.24**

**Тема:** Frontend разработка основная страница: написание форм входа/регистрации. Оформление отчетной документации.

**Ход работы:**

**Шаг 1.** В данной главе мы реализуем внешний вид формы входа и регистрации. Создадим новое представление \_LoginRegistrationPartial.cshtml в папке \Views\Shared. В этом файле для начала пропишем разметку контейнера формы как показано на рисунке 59. А также сразу в файле \_Layout.cshtml пропишем строку как показано на рисунке 60.

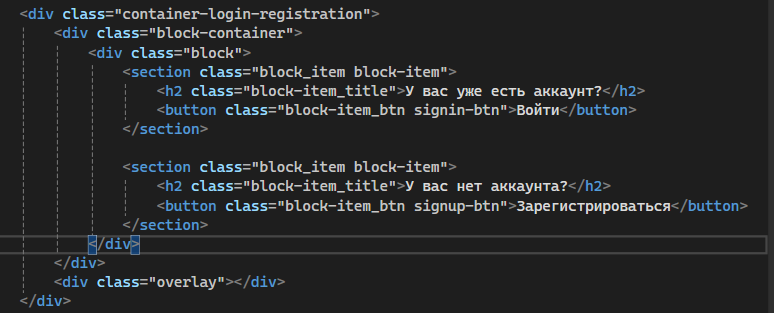
******

Рисунок 59 – контейнер формы

**https://lh7-rt.googleusercontent.com/docsz/AD_4nXcT-ywfUgKusajPBP-Ty48Lk-rbyGxTi1yOjtpZMq-y3trw6px-Ked4ez9tw2yothZZEFItYgQxcZzMc49CGZIJCXGgewADqkfMKzpak1fkfngG34SXQxINbc65c7PT22748ym3rzbmwm3yJVJA6Hq0YtI?key=INBRQAIJ6oNZ936DLEKIIy_J**

Рисунок 60 – вызов частичного представления

**Шаг 2.** Зафиксируем контейнер по центру разделив его на две части. В левой части будет располагаться кнопка «Войти» а в правой части контейнера будет «Зарегистрироваться». А также здесь показан div с классом overlay который необходим для того, что после открытия формы можно было ее закрыть, нажав за пределы формы.Для стилей связанных с входом и регистрацией создадим файл css с названием login\_and\_registration. Не забудьте что при создании нового css/js файла необходимо их подключить.

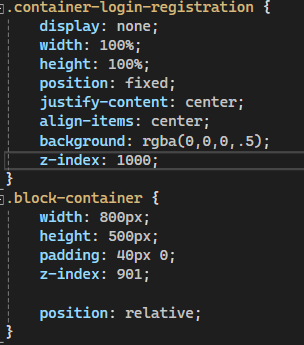
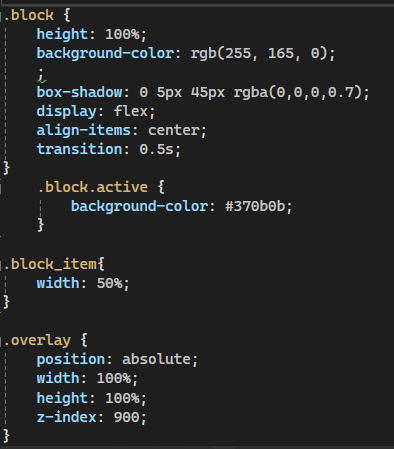
******

Рисунок 61 – стили для контейнера формы

**Шаг 3.** На рисунке 62 показано скрипт для открытия формы при нажатии на кнопку «Войти». Здесь так же создаем новый файл js с названием login\_and\_registration\_script. Для того что бы скрипт срабатывал нашей кнопке «Войти» необходимо добавить *id="click-to-hide".*

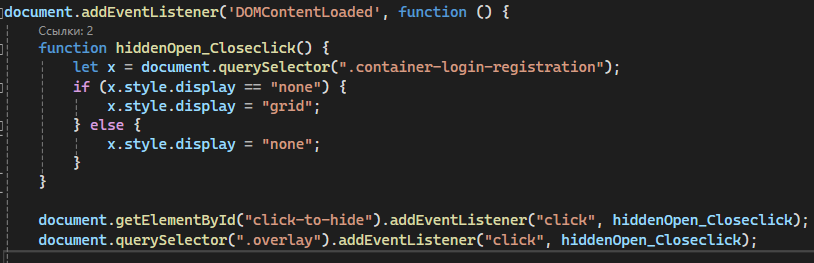


Рисунок 62 – скрипт открытия формы

**Шаг 4.** Далее зададим стили для текста и кнопок левой и правой части контейнера.

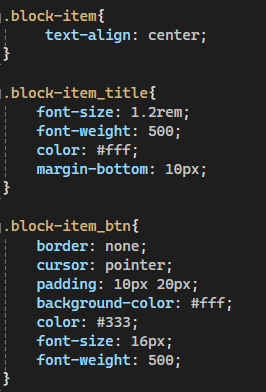


Рисунок 63 – стили для блоков контейнера

**Шаг 5.** На рисунках 64-65 представлена разметка форм входа и регистрации.

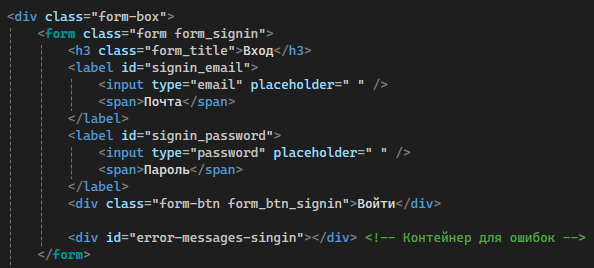
******

Рисунок 64 – разметка формы входа

******

Рисунок 65 – разметка формы регистрация

**Шаг 6.** Данные стили оформляют саму форму, текст и поля ввода на ней (рис. 66). При нажатии на поле подсказка будет подниматься над полем (рис. 67).

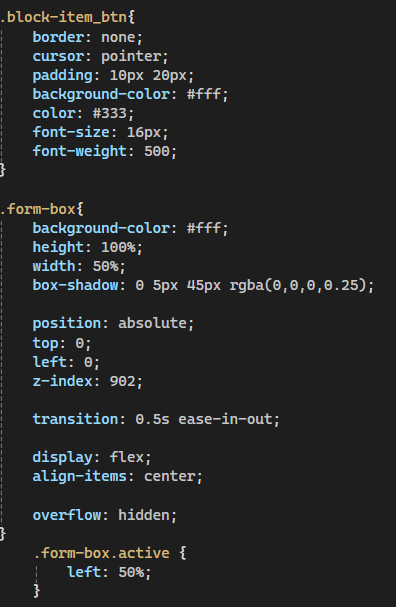
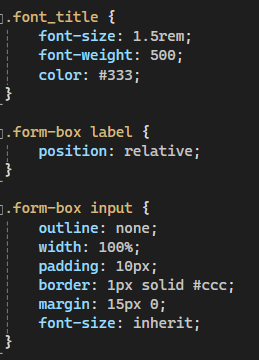
 ******

Рисунок 66 – стили для формы

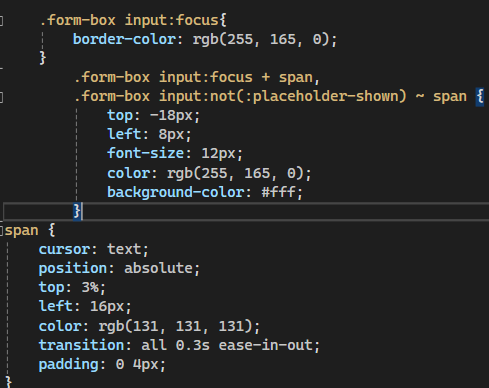
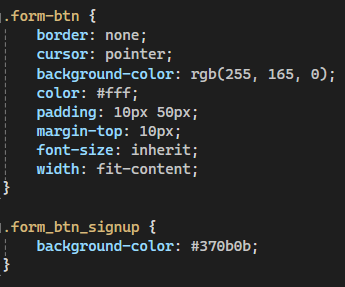
******

 Рисунок 67 – стили для кнопок и полей ввода

При этом также присутствуют стили, благодаря которым форма красиво будет переключаться между регистрацией и входом. Все классы active предназначены для изменения цвета и положения формы.

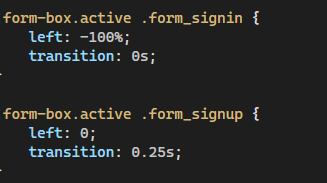
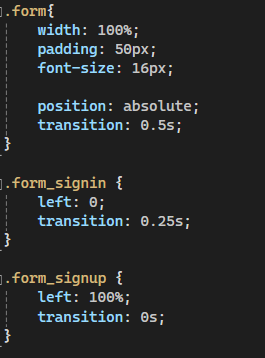
******

Рисунок 68 – стили для смены формы

**Шаг 7.** Чтобы при нажатии на кнопку появлялась форма нужно прописать скрипт как показано на рисунке 69.

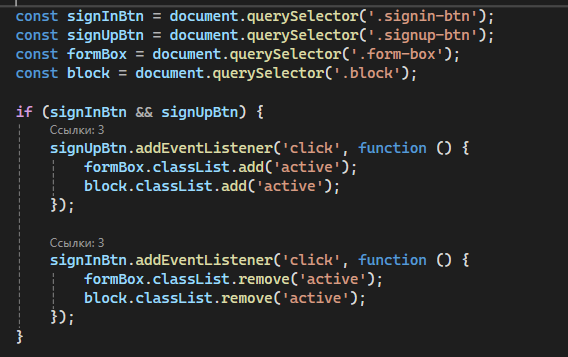
******

 Рисунок 69 – скрипт перехода от одной формы к другой

**Шаг 8.** В конце данной главы у вас должен получится вот такой результат.

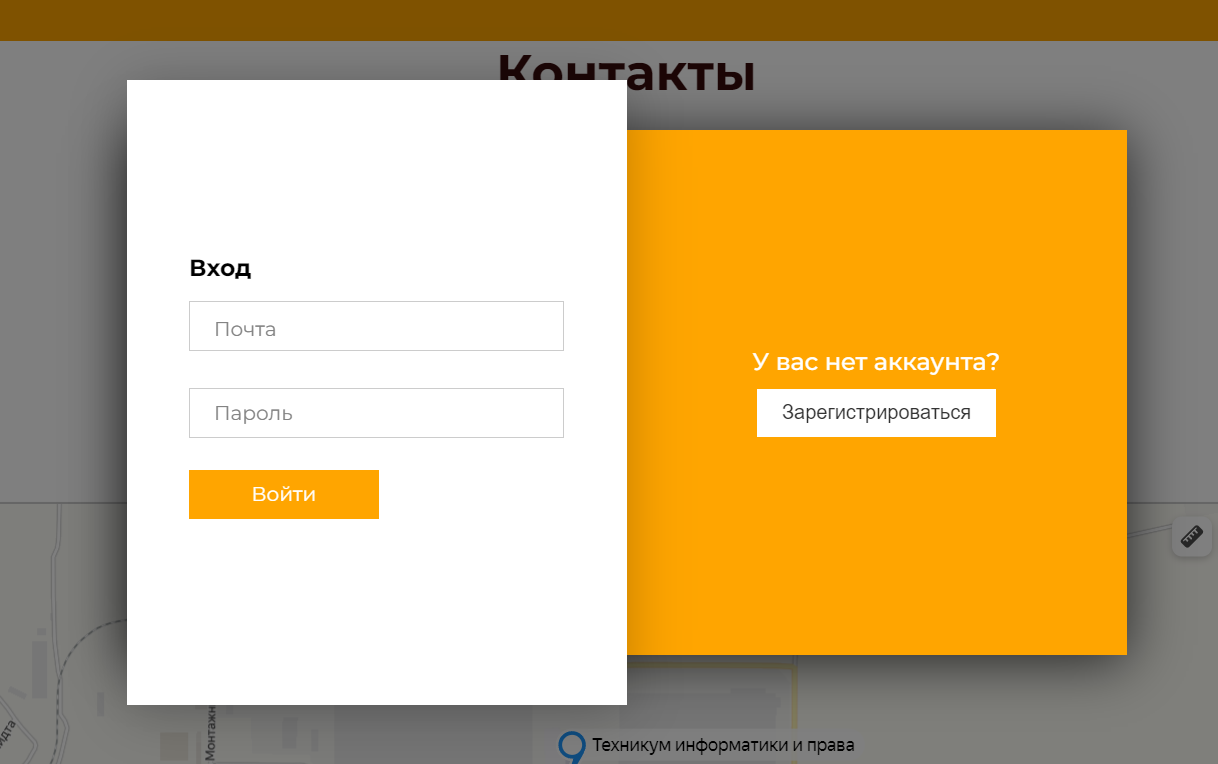


Рисунок 70 – Результат

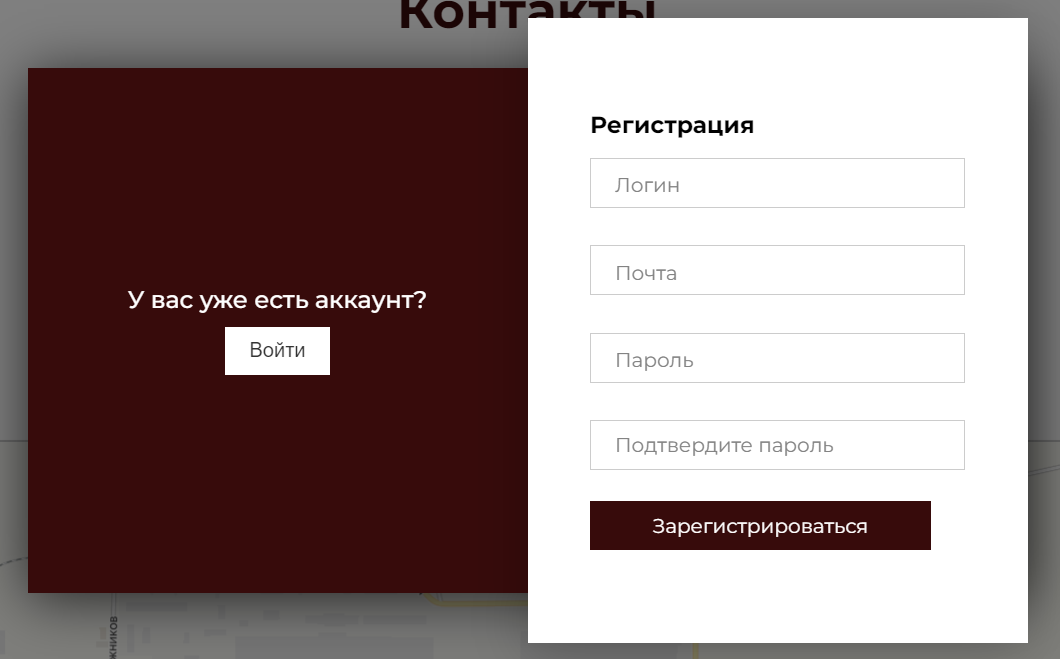


Рисунок 71 – Результат

**Вывод:** Выполнил Frontend разработку основной страницы: написал формы входа/регистрации. Оформил отчетную документацию.

**08.11.24**

**Тема:** Frontend разработка основная страница: написание секции «Написать сообщение» и подвала сайта. Оформление отчетной документации.

**Ход работы:**

**Шаг 1.** Разместим ниже секции с контактами секцию с отправкой сообщения на почту. Сам функционал отправки мы рассмотрим позже. Этот этап показан на рисунках 74-75.

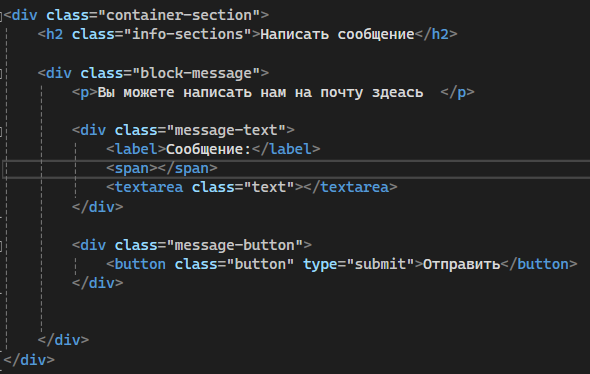


Рисунок 74 – секция «Написать сообщение» разметка

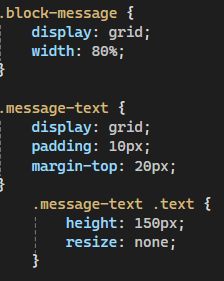
 

Рисунок 75 – секция «Написать сообщение» стили

**Шаг 2.** На рисунке 76 показан код для подвала сайта. Там будет размещена минимальная информация и дублирование меню.



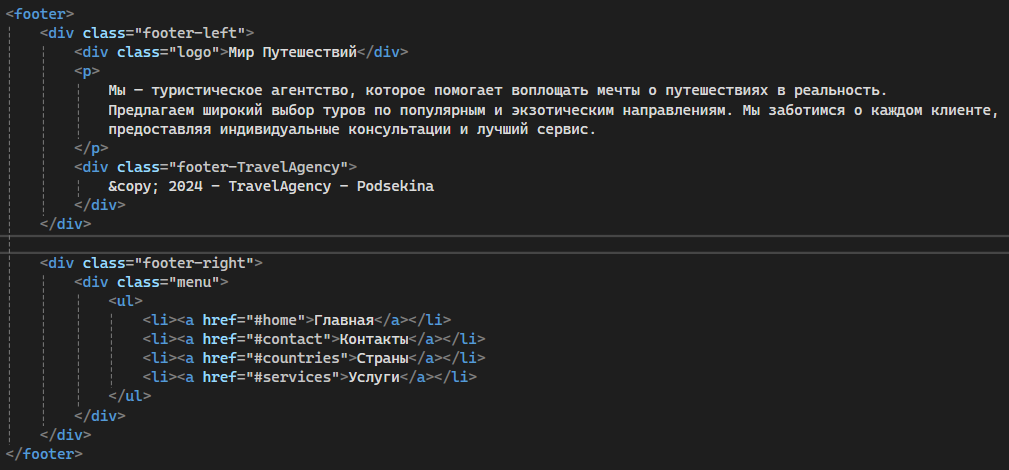


Рисунок 76 – footer для сайта

**Шаг 3.** В конце данной главы у вас должен получится вот такой результат.

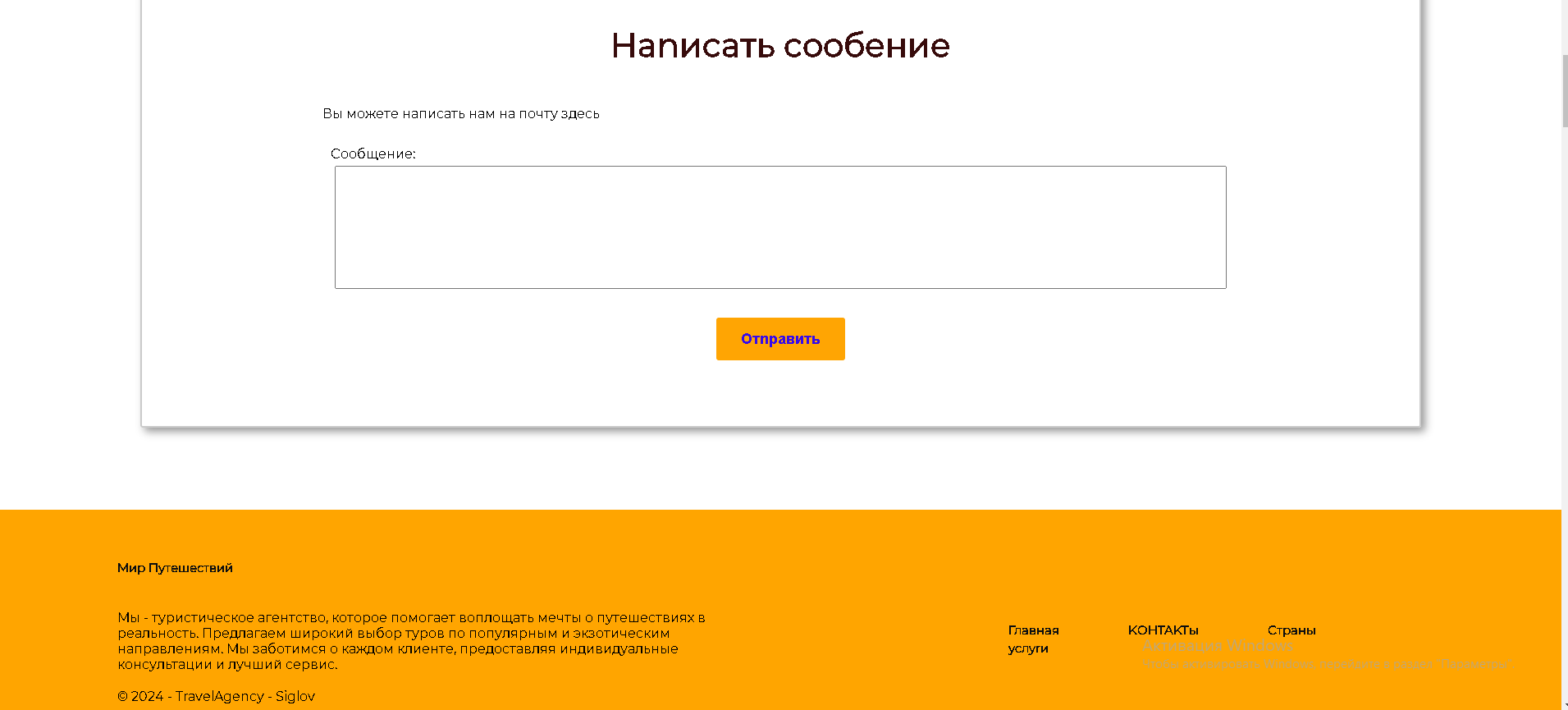
******

Рисунок 77 – Результат

**Вывод:** Frontend разработка основная страница: написание секции «Написать сообщение» и подвала сайта. Оформление отчетной документации

**09.11.24**

**Тема:** Использование fetch-запросов для обновления данных без перезагрузки страницы. Оформление отчетной документации.

**Ход работы:**

**Шаг 1.** Для начала создадим объекты, которые будем передавать, а именно это модели для входа и регистрации пользователя. В *Domain* (имеется в виду проект *TravelАgency.Domain*) создадим папки *ViewModels/LoginAndRegistration.*  Создадим класс *LoginViewModel* который будет содержать некоторые свойства и атрибуты для проверки валидности как показано на рисунке 78.

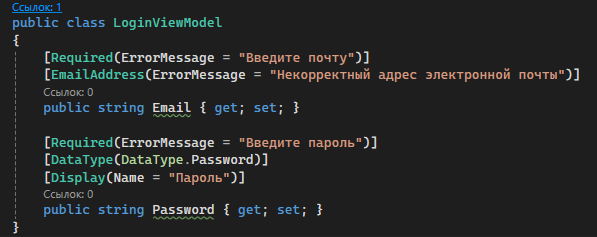


Рисунок 78 – класс *LoginViewModel*

**Шаг 2.** Далее создаем класс *RegisterViewModel* аналогично предыдущему этот класс также содержит некоторые свойства и атрибуты, которые совпадают с прописанными ранее формами.

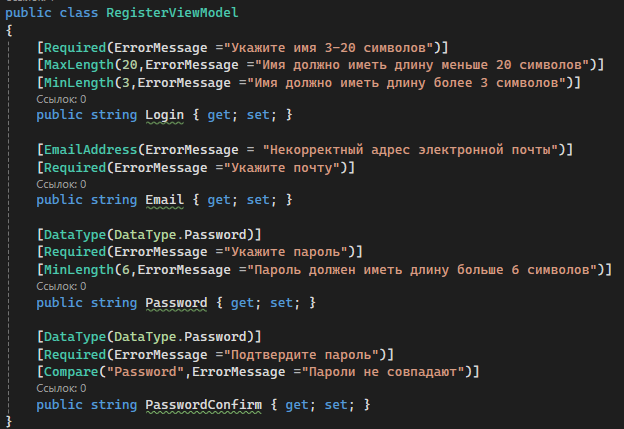
****

Рисунок 79 – класс *RegisterViewModel*

**Шаг 3.** Чтобы наш mvc-проект получил доступ к другому проекту ему необходимо добавить ссылку. Правой кнопкой мыши нажимаем на проект TravelАgency и выбираем пункт Добавить/Ссылка на проект…

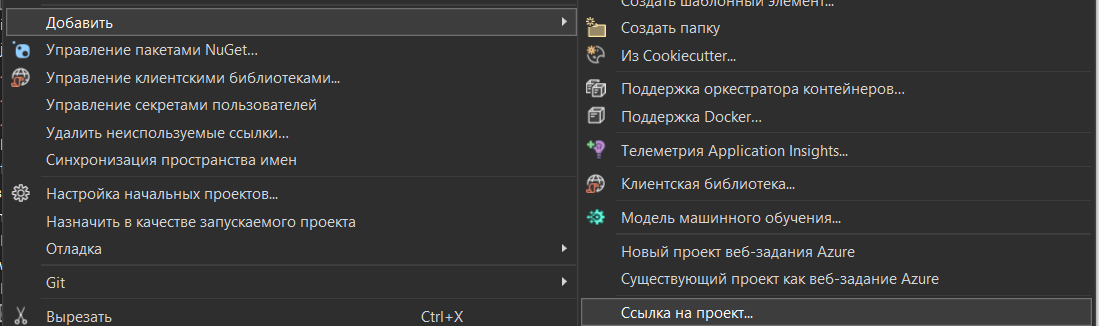
****

Рисунок 80 – Добавить/Ссылка на проект…

**Шаг 4.** В появившемся окне Диспетчера ссылок выбираем проект, на который мы хотим установить ссылку ставя галочку слева после чего необходимо нажать OK.

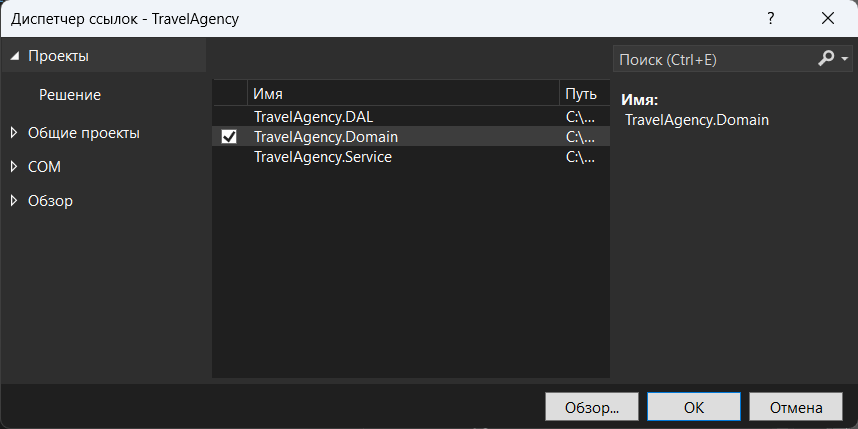
****

Рисунок 81 – Диспетчер ссылок

**Шаг 5.** После того как мы добавили ссылку на проект мы можем обратится к модели для входа из контроллера. Добавим в контроллер действие *Login.*

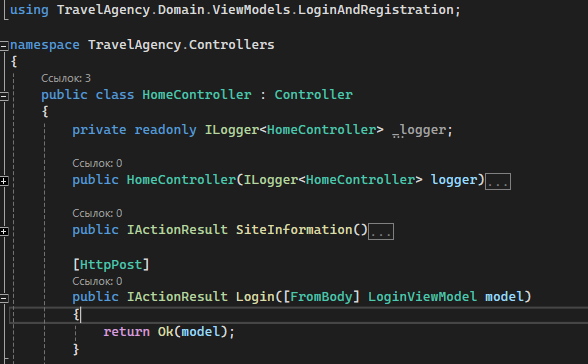


Рисунок 82 – Действие *Login*

**Шаг 6.** В файле *login\_and\_registration\_script* прописываем следующий код. Здесь при нажатии на кнопку формируются все необходимые данные для *fetch*-запроса, а именно *url, body, method.*

****

Рисунок 83 – обработка нажатия на кнопку войти

**Шаг 7.** Далее представлена реализация метода, вызываемого на предыдущем рисунке. Здесь непосредственно реализован fetch-запрос.

****

Рисунок 84 – функция *sendRequest*

**Шаг 8.** Теперь можно посмотреть через точки останова как передаются данные или сразу посмотреть в консоли через панель разработчик в браузере. Добавим в контроллер проверку валидности по атрибутам.

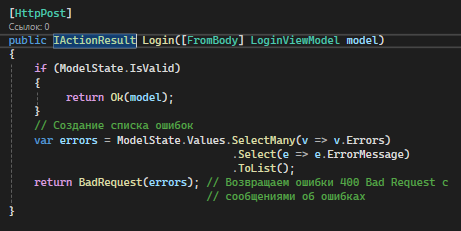


Рисунок 85 – проверка валидности

**Шаг 9.** Добавим в обработчик нажатия кнопки для вызова функции для отображения ошибок (рис. 86). Так же в конце формы *form\_signin* в файле \_*LoginRegistrationPartial* добавим строку *<div id="error-messages-singin"></div>* тоже самое сделаем для формы *form\_signup.*



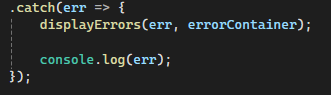


Рисунок 86 – вызов функции *displayErrors*

**Шаг 10.** Далее представлена сама функции для отображения ошибок (рис 87). Она очищает предыдущие ошибки и создает элемент с классом *error* для каждой ошибки.

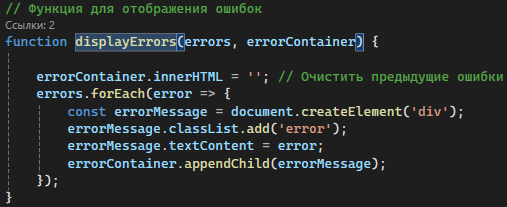


Рисунок 87 – Функция *displayErrors*

**Шаг 11.** Добавим совсем немного стилей для отображения ошибок как показано на рисунке 88.

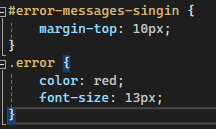


Рисунок 88 – Стили для отображения ошибок

**Шаг 12.** Теперь добавим вызов еще одного метода, который будет вызываться если все данные были введены верно. *location.reload()* обновит страницу.

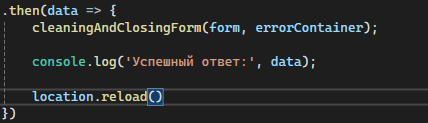


Рисунок 89 – Вызов функции *CleaningAndClosingForm*

**Шаг 13.** Данная функция предназначена чтобы при успешном ответе запроса очищать все поля и закрывать форму.

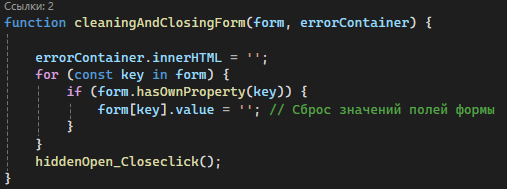


Рисунок 90 – Функция *CleaningAndClosingForm*

**Шаг 14.** Теперь пора проделать все те же махинации только для формы регистрации. Ниже приведен пример обработки нажатия на кнопку зарегистрироваться.



Рисунок 91 – обработка нажатия на кнопку зарегистрироваться



**Шаг 15.** А также добавим действие в контроллер в соответствии с рисунком 92.

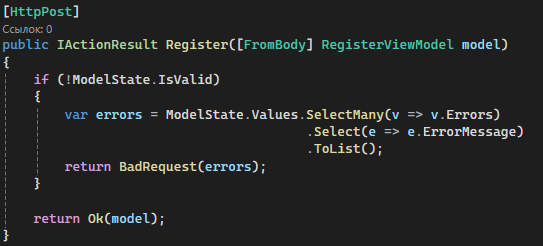


Рисунок 92 – действие *Register*

**Вывод:** Использовал fetch-запросы для обновления данных без перезагрузки страницы. Оформил отчетную документацию.

**11.11.2024**

**Тема:** Создание адаптивного интерфейса с использованием медиазапросов *CSS*. Оформление отчетной документации.

**Ход работы:**

1. Для создания адаптивного интерфейса необходимо использовать медиазапросы *CSS*. Медиа-запрос состоит из ключевого слова, описывающего тип устройства (необязательный параметр) и выражения, проверяющего характеристики данного устройства. Из всех характеристик чаще всего проверяется ширина устройства *width*. Медиа-запрос является логическим выражением, которое возвращает истину или ложь. Для использования медиазапросов в нашем проекте необходимо создать новый файл, подключить его и написать следующий код:



Рисунок 55. Медиазапрос.

1. Далее для нашего всплывающего меню необходимо написать следующую разметку:

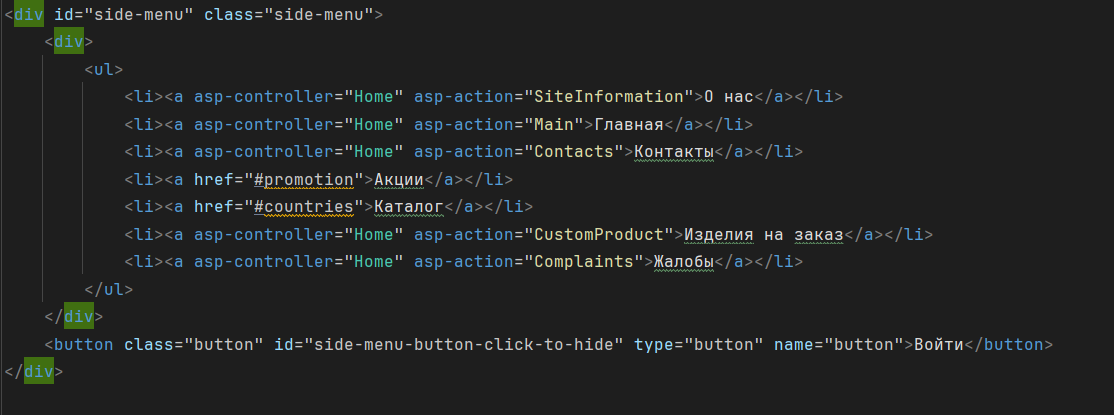


Рисунок 56. Разметка всплывающего меню.

1. Для работы нажатия на кнопку необходимо написать следующий скрипт в файл *shared\_site.js*:

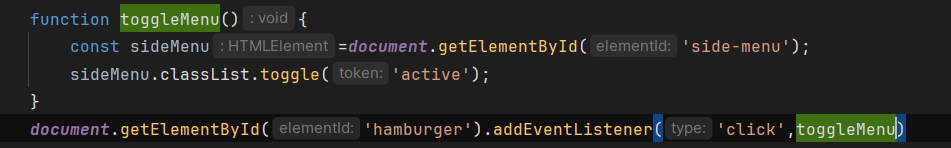


Рисунок 57. Скрипт обрабатывающий нажатие на кнопку.

1. После необходимо написать медиазапросы, которые относятся ко всем разделам сайта, пример приведен ниже:

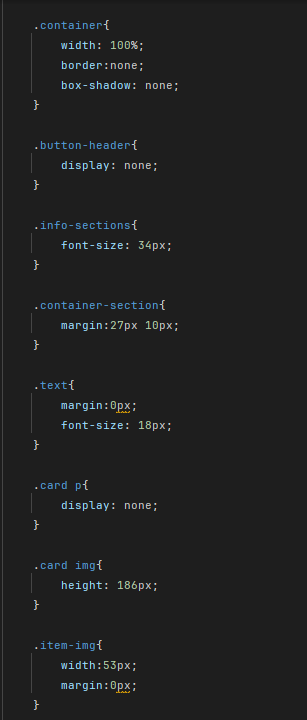


Рисунок 58. Медиазапросы к другим разделам сайта.

1. После всех проделанных действий можно просмотреть результат работы:

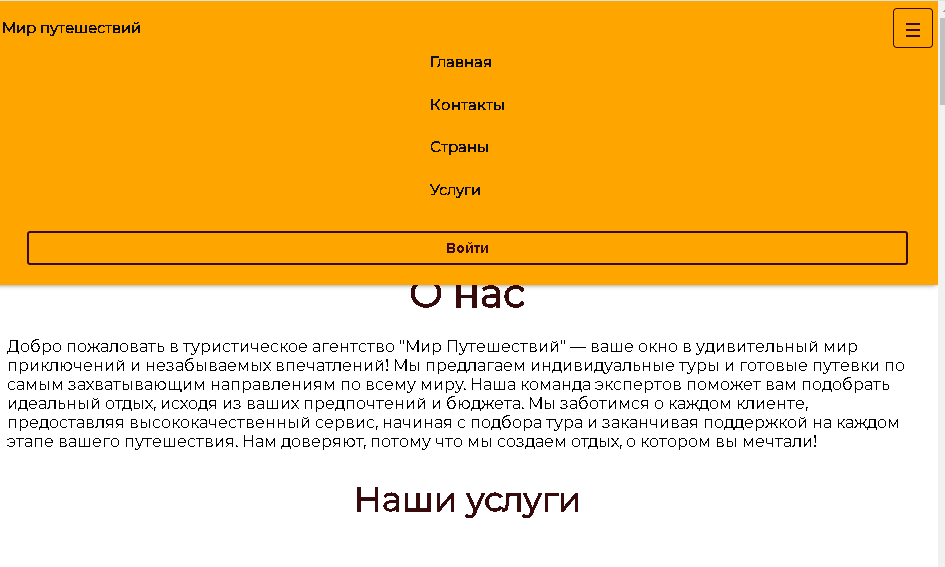


Рисунок 59. Результат работы.

**Вывод:** в ходе проделанной работы мы смогли адаптировать наш интерфейс под экраны различных размеров.

**12.11.2024**

**Тема:** Знакомство с *PostgreSQL*. Оформление отчетной документации.

**Ход работы:**

1. Для начала работы с *PostgreSQL* необходимо ознакомиться что это такое и для чего необходимо. *PostgreSQL* – это объектно-реляционная система управления базами данных (*ORDBMS*), наиболее развитая из открытых [СУБД](https://blog.skillfactory.ru/glossary/subd/) в мире. Имеет открытый исходный код и является альтернативой коммерческим базам данных. Она предоставляет следующие функции:

* Гибкий доступ к базам данных, их организация и хранение.
* Управление записями в базах данных: создание, редактирование и удаление, обновление версий и так далее.
* Просмотр нужной информации из базы по запросу, например, для ее отправки на сайт или в интерфейс приложения.
* Отправка транзакций, последовательных запросов, собранных в подобие скрипта.
* Настройка и контроль доступа к той или иной информации, группировка пользователей по уровню прав.
* Контроль версий и организация одновременного доступа к базе из разных источников так, чтобы предотвратить сбои.
* Защита информации от возможных утечек и потерь.
* Контроль состояния базы в целом.

Также необходимо понимать, что такое база данных. База данных- совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Ознакомимся с установкой *PostgreSQL.* Сперва необходимо перейти на сайт установщика и скачать установщик.



Рисунок 60. Скачивание установщика.

1. Далее необходимо запустить установщик.

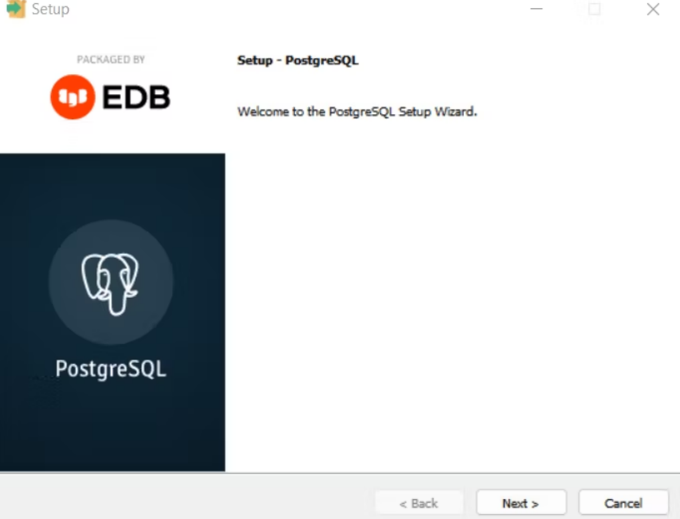


Рисунок 61. Запуск установщика.

1. После надо выбрать папку куда будет установлено *PostgreSQL*:

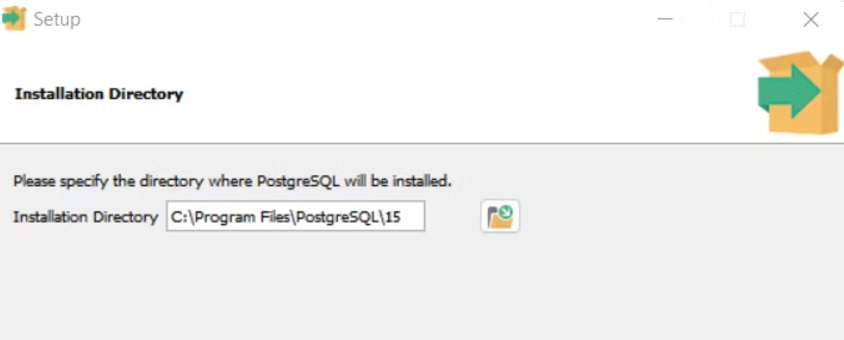


Рисунок 62. Выбор папки куда будет установлено *PostgreSQL*.

1. Вместе с СУБД можно установить следующие компоненты:

* *PostgreSQL Server* — это сам сервер баз данных *PostgreSQL*;
* *pgAdmin 4* — приложение с графическим интерфейсом для администрирования СУБД;
* *Stack Builder* — утилита, которая поможет установить дополнительные библиотеки и инструменты в *PostgreSQL*;
* *Command Line Tools* — инструменты для работы с СУБД через командную строку.

После чего необходимо дождаться установки и уже начинать работу в этом приложении.



Рисунок 63. Установка основных пакетов.

**Вывод:** ознакомились с *PostgreSQL*, научились его устанавливать.

**13.11.2024**

**Тема:** Создание базы данных в СУБД *PostgreSQL*. Оформление отчетной документации.

**Ход работы:**

1. Для создания базы данных необходимо нажать на кнопку Создать:

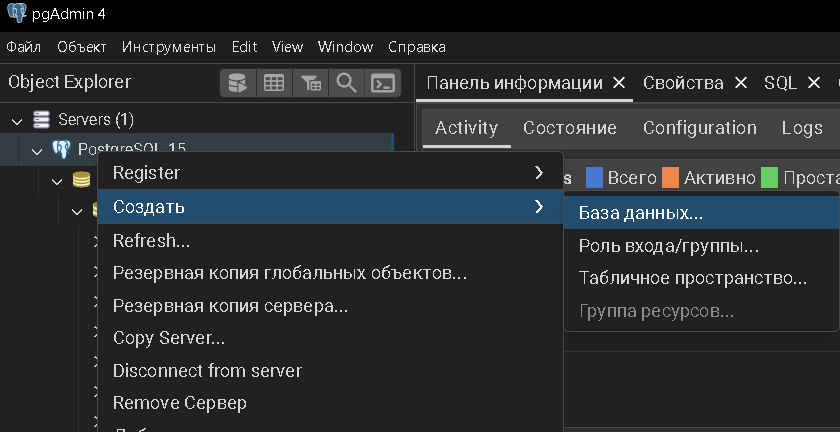


Рисунок 64. Создание базы данных.

1. Далее, нам необходимо создать таблицы нашей базы данных. Для этого необходимо открыть пункт Схемы нашей базы данных и нажать пункт Создать:

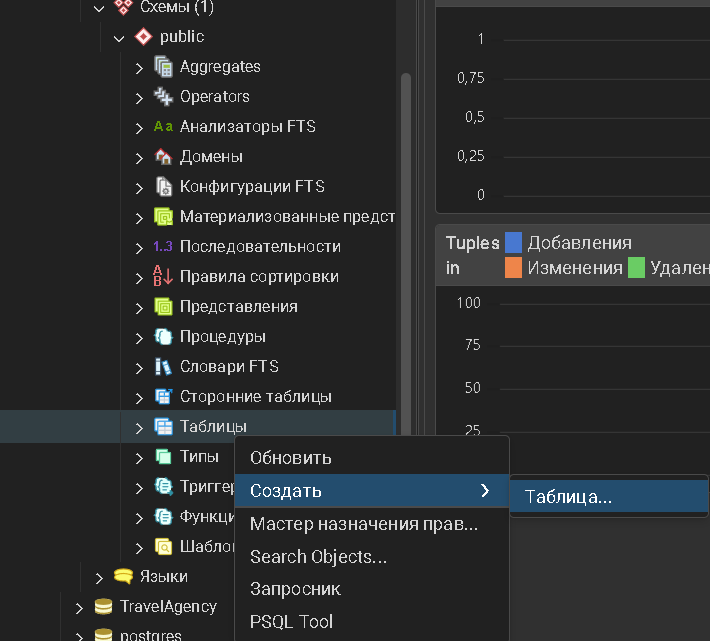


Рисунок 65. Создание таблицы.

1. После этого необходимо заполнить таблицу полями и задать этим полям значение, что показано на следующем рисунке:

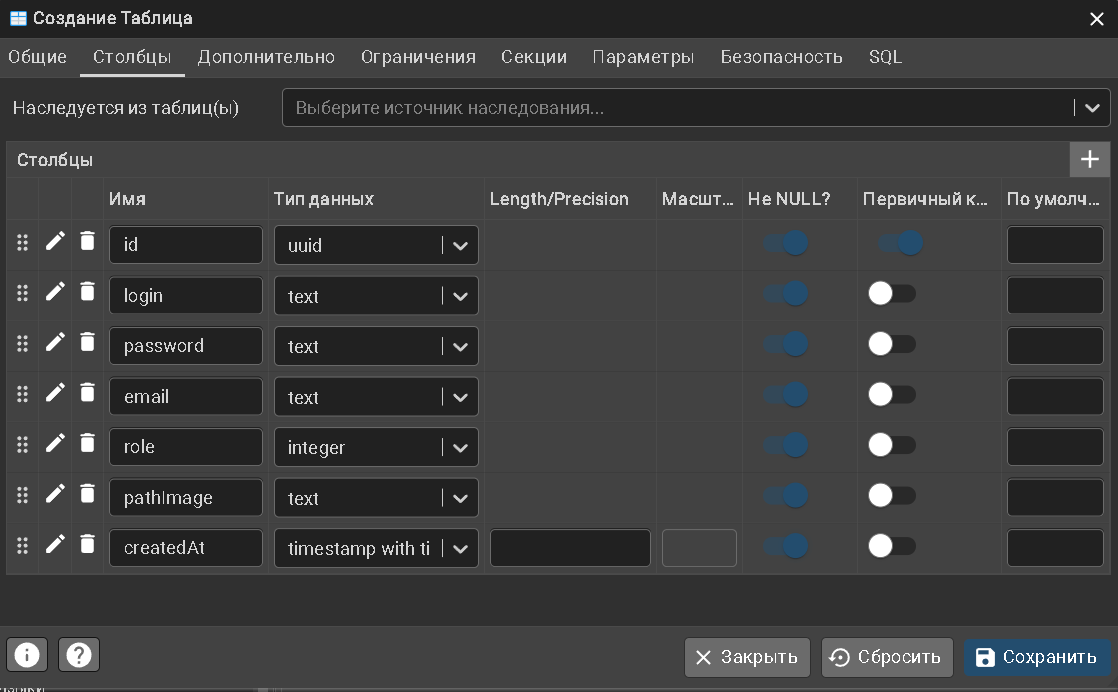


Рисунок 66. Заполнение таблицы строками.

1. Первая таблица- таблица пользователей, которые зарегистрировались на сайте. Структура указана в таблице 1.

Таблица 1-Структура таблицы пользователей.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| id | uuid | Идентификатор пользователя |
| login | text | Имя пользователя |
| password | text | Пароль пользователя |
| email | text | Электронная почта пользователя |
| role | integer | Роль пользователя |
| pathImage | text | Картинка пользователя |
| createdAt | timestamp with time zone | Дата создания |

1. Вторая таблица- таблица продукции нашего магазина ювелирных изделий. Структура находиться в таблице 2.

Таблица 2-Структура таблицы продукции.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| *id* | *uuid* | Идентификатор товара |
| *idCategories* | *uuid* | Идентификатор категории товара |
| *name* | *text* | Название товара |
| *cost* | *numeric* | Стоимость товара |
| *idImg* | *uuid* | Идентификатор фотографии |
| *createdAt* | *timestamp with time zone* | Дата создания профиля |

1. Третья таблица- таблица категорий товаров ювелирных изделий. Структуру можно увидеть в таблице 3.

Таблица 3-Структура таблицы категории товаров

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| *id* | *uuid* | Идентификатор категории товаров |
| *name* | *text* | Описание категории товаров |
| *idImg* | *text* | Картинка категории товаров |
| *createdAt* | *timestamp with time zone* | Дата создания товара |

1. Четвертая таблица- таблица заказов наших пользователей, которая указывает на товар, пользователя и цену заказа. Структура показана в таблице 4.

Таблица 4-Структура таблицы заказов.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| *id* | *uuid* | Идентификатор заказа |
| *idUser* | *uuid* | Идентификатор пользователя |
| *idProduct* | *uuid* | Идентификатор товара |
| *name* | *text* | Название заказа |
| *cost* | *numeric* | Стоимость заказа |
| *createdAt* | *timestamp with time zone* | Дата создания заказа |

1. Пятая таблица- таблица галереи картинок определенного товаров. Структура показана в таблице 5.

Таблица 5-Структура таблицы картинок товара.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| *id* | *uuid* | Идентификатор фото |
| *idProduct* | *uuid* | Идентификатор товара |
| *pathImg* | *text* | Картинка товара |

1. Шестая таблица- таблица запросов наших пользователей, которые сделали заказ. Ее структура показана в таблице 6.

Таблица 6-Структура таблицы запросов.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| *id* | *uuid* | Идентификатор запроса |
| *idUser* | *uuid* | Идентификатор пользователя |
| *decription* | *text* | Описание запроса |
| *status* | *integer* | Статус запроса |
| *createdAt* | *timestamp with time zone* | Дата создания запроса |

1. Седьмая таблица- таблица жалоб пользователя на заказ. Структура этой таблицы указана в таблице 7.

Таблица 7-Структура таблицы жалоб.

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| *id* | *uuid* | Идентификатор жалобы |
| *idUser* | *uuid* | Идентификатор пользователя |
| *idOrders* | *uuid* | Идентификатор заказа |
| *decription* | *text* | Описание жалобы |

1. Далее необходимо установить связи между таблицами. Для этого необходимо нажать ПКМ на таблицу, нажать пункт *properties*, пункт Ограничения:

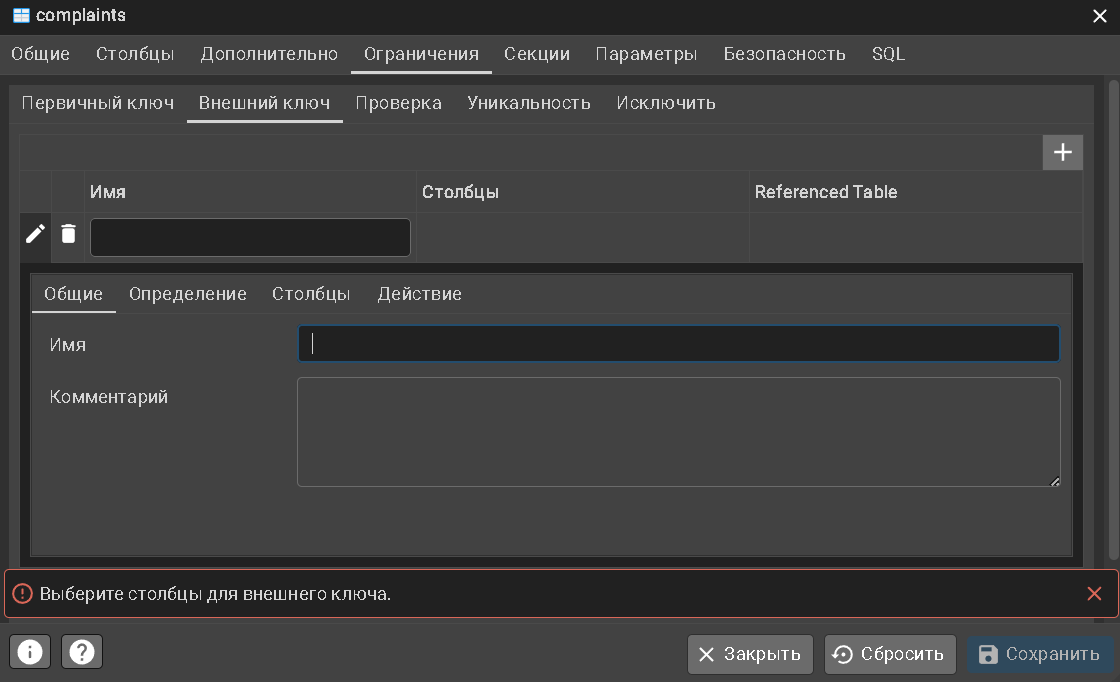


Рисунок 67. Установка зависимостей между таблицами.

1. В результате получится следующая структура.

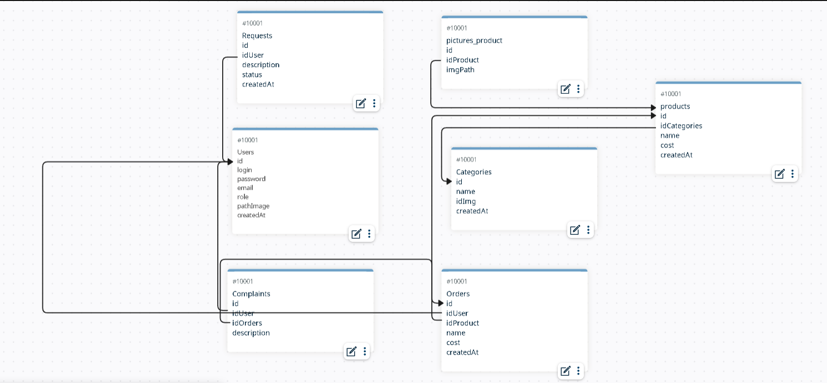


Рисунок 68. Структура взаимосвязей между таблицами.

**Вывод:** в результате была создана база данных со всеми необходимыми таблицами.