**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)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

код, наименование специальности

Курс \_\_\_\_\_\_3\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ группа \_\_\_313\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_ТСПодсекина\_\_/

подпись Ф.И.О.

**Индивидуальное задание**

**на практику \_\_\_\_\_\_\_\_\_УП.01.01 УЧЕБНАЯ ПРАКТИКА\_\_\_\_\_\_\_\_\_\_\_**

индекс практики, вид практики

по профессиональному модулю (междисциплинарному курсу)

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

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

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

Ф.И.О. студента Тизул Игорь ,

Учебная группа \_\_\_\_\_\_\_313\_\_\_\_\_\_\_ курс \_\_\_\_\_III\_\_\_\_\_\_\_\_

Специальность \_\_\_**2.09.02.07 Информационные системы и программирование**

код и наименование специальности

Объем 216 часов с «01» ноября 2024 г. по «12» декабря 2024 г.

|  |  |  |
| --- | --- | --- |
| № задания | Содержание задания | ОК, ПК |
| 19 | Выполнить задания по проектированию, разработке сайта на основе шаблона MVC «Интернет-магазин для продажи школьных принадлежностей»  Разработать интернет-магазин, специализирующийся на школьных товарах, таких как рюкзаки, пеналы, канцелярия, учебники и тетради.  Сайт должен включать удобную навигацию с фильтрацией товаров по категориям, корзину и систему отзывов. | ПК 1.1-  ПК 1.5 |
| 1. Разработать Frontend с использованием CSS и JS:основная страница, регистрация/авторизация, вкладка с запросами и корзиной, страница со странами, фильтрация, страница с выбранным элементом, отзывы, панель администратора.  2. Разработать базу данных.  3. Разработать проект с моделями  4. Разработать проект с подключением БД и запросами к ней.  5. Разработать проект с сервисами, реализующими бизнес-логику приложения.  6. Реализовать контролеры.  7. Провести модульное тестирование проекта. |

Руководитель практики Т.С.Подсекина.

**01.11.24**

**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: Создаём проект входя в программу Visial Studio 2024.

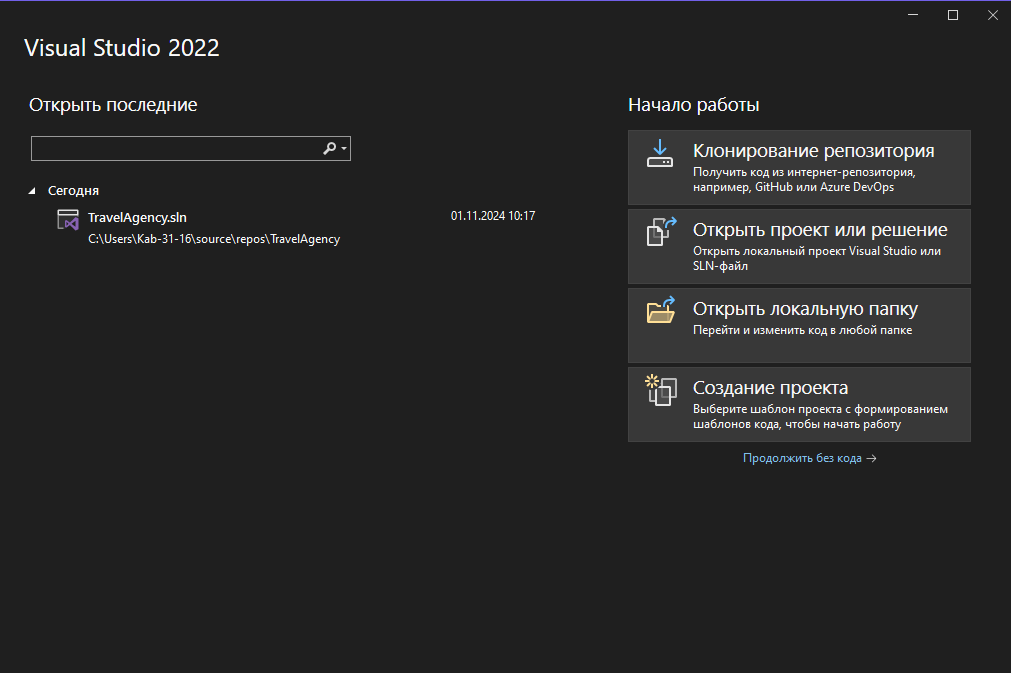


Рис. 1. Создание проекта.

Шаг 2: Нажав на кнопку «Создание проекта», мы переходим в меню шаблонов. Выбираем шаблон как на скриншоте.

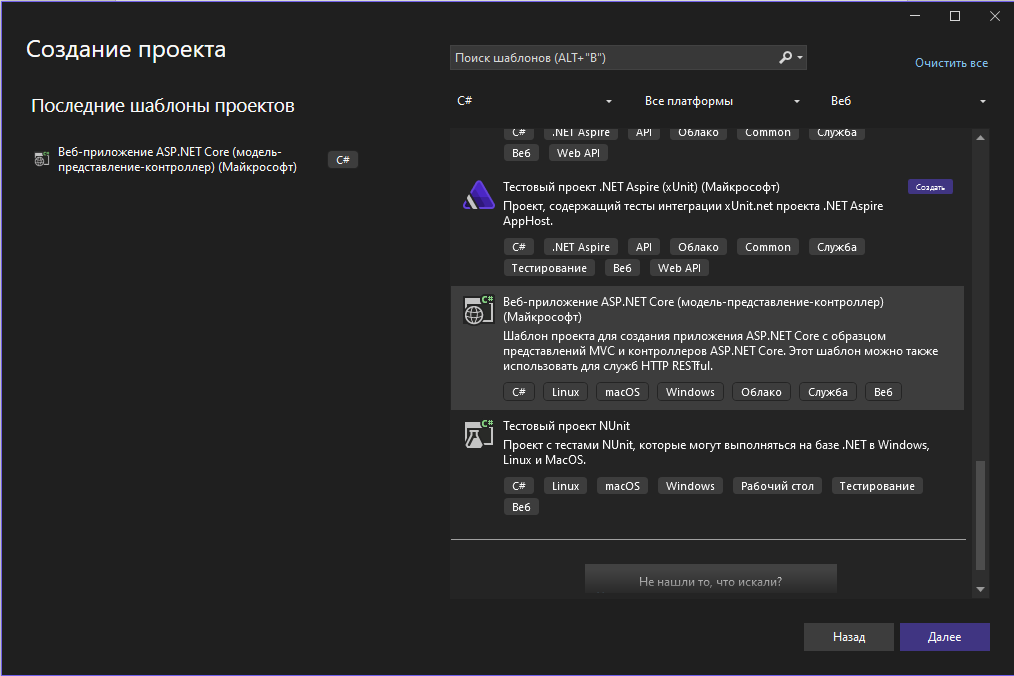


Рис. 2. Установка Шаблонов.

Шаг 3: Создав проект, мы переходи в вкладку обозреватель решений. В данной вкладке мы можем управлять, удалять, и добавлять наши файлы программы.

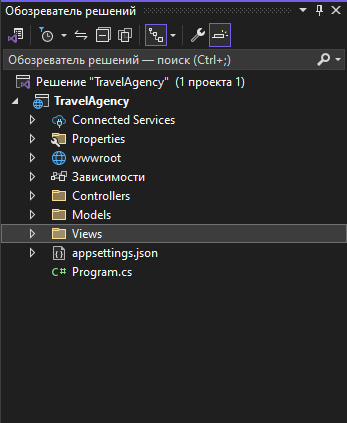


Рис. 3. Обозреватель решений.

Шаг 4: Находим в обозревателе решений папку Views, потом переходим в следующую папку Shered, а затем в другую \_Layout.cshtml. В данном файле у нас будет содержаться именно этот код.

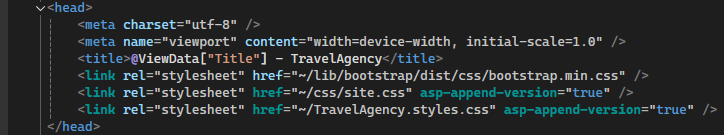


Рис. 4. Тег <head> в папке: Views/Shered/\_Layout.cshtml.

Шаг 5:Находим в обозревателе решений папку Views, потом переходим в следующую папку Shered, а затем в другую \_Layout.cshtml. В данном файле у нас будет содержаться именно этот код.

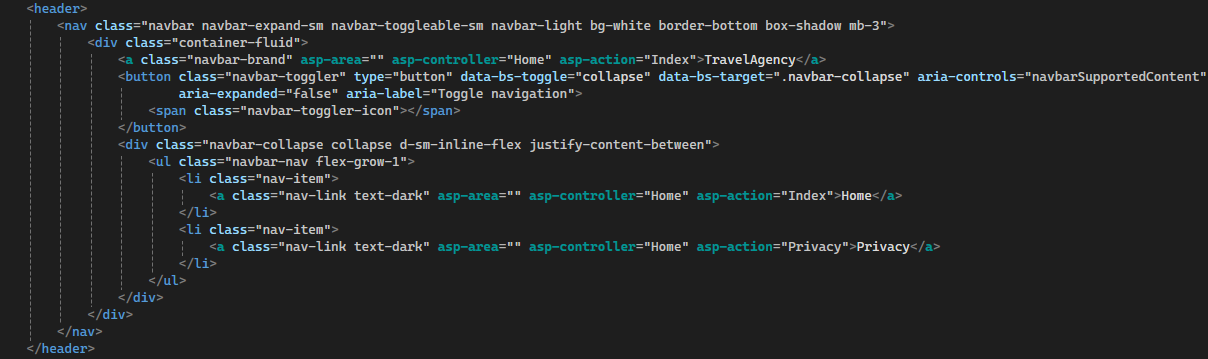


Рис. 5. Тег <header> в папке: Views/Shered/\_Layout.cshtml

Шаг 6: Находим в обозревателе решений папку Views, потом переходим в следующую папку Shered, а затем в другую \_Layout.cshtml. В данном файле у нас будет содержаться именно этот код.

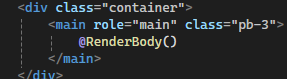


Рис. 6. Функция @RenderBody()

Шаг 7: Заходим в папку Views, потом переходим в следующую папку Shered, а затем в \_Layout.cshtml. В файле будет содержаться информация о данном теге **<**footer**>**.

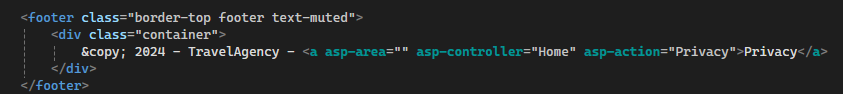


Рис. 7. Тег <footer>

Шаг 8: Находясь также в папкеViews, мы находим строчки кода, которые обозначают подключение скриптов к программе.

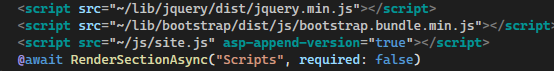


Рис. 8. Подключение скриптов.

Шаг 9: Заходим в папку Views, потом переходим в следующую папку Shered, а затем в \_ViewImports.cshtml.Открываем этот файл и мы сможем увидеть его содержимое (код).



Рис. 9. Содержание файла\_ViewImports

Шаг 10: Заходим в папку Views, потом переходим в следующую папку Shered, а затем в \_ViewStart.cshtml**.** Открываем этот файл и мы сможем увидеть его содержимое (код).



Рис. 10. Содержание файла\_ViewStart.cshtml

Шаг 11:Находим в обозревателе решений Файл Program.cs.

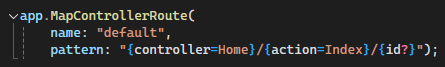


Рис. 11. Фрагмент кода из файла Program.cs

**Вывод:** Я, познакомился с mvc структурой файлов, их содержимого кода, а так же их функциональностью.

**01.11.24**

**Тема:** Создание проекта на основе шаблона MVC.

**Ход Работы:**

Шаг 1: Переходим в зарегистрированный аккаунт ГитХаба. Переходим в вкладку нашего профиля.

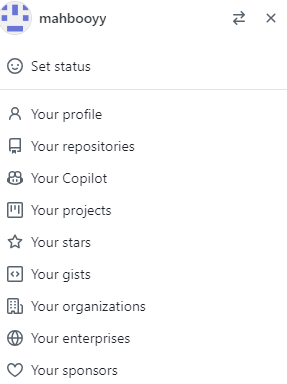


Рис.12. Профиль Гитхаба.

Шаг 2: Зайдя в меню ГитХаба, выбираем создать репозиторий, и называем его.

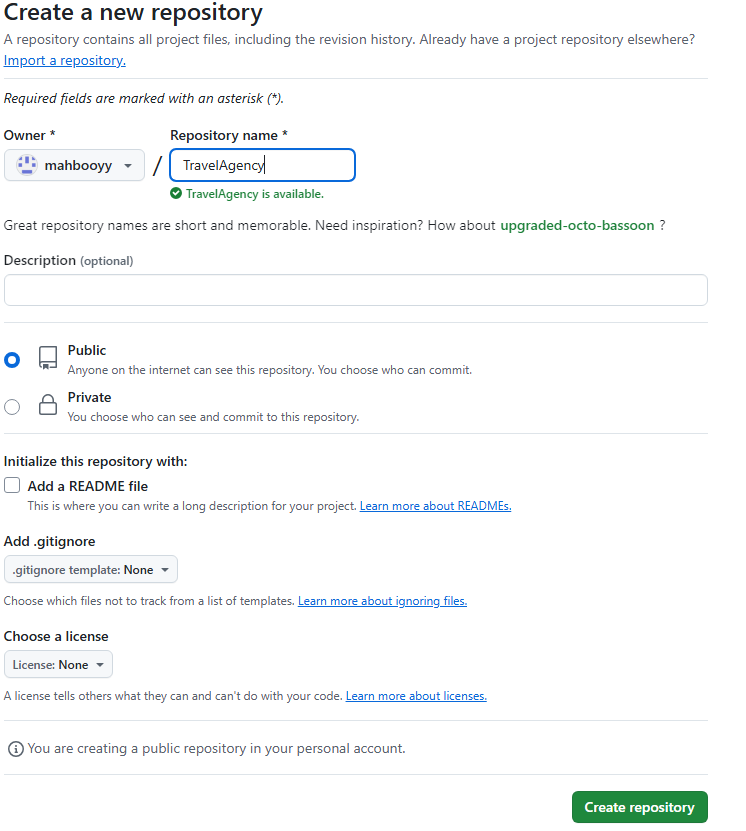


Рис.13. Создание репозитория.

Шаг 3: После того как мы создали репозиторий, то дальше мы клонируем данный репозиторий на веб версию, с этим же репозиторием. Переносим файлы.

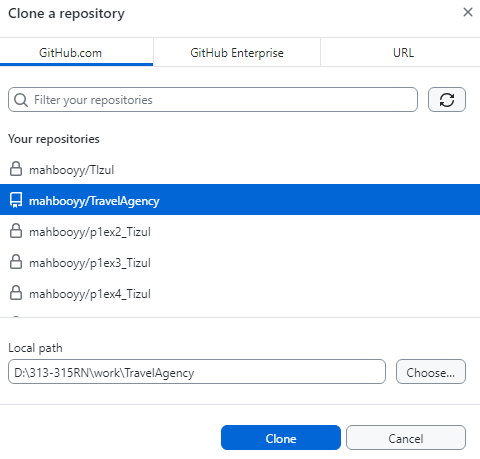


Рис.14. Клонирование репозитория на свой ПК.

Шаг 4: После того как мы склонировали файлы на веб-репозиторий, и если мы сделали всё правильно, то вид приложения с открытым репозиторием будет выглядеть так как на скриншоте.

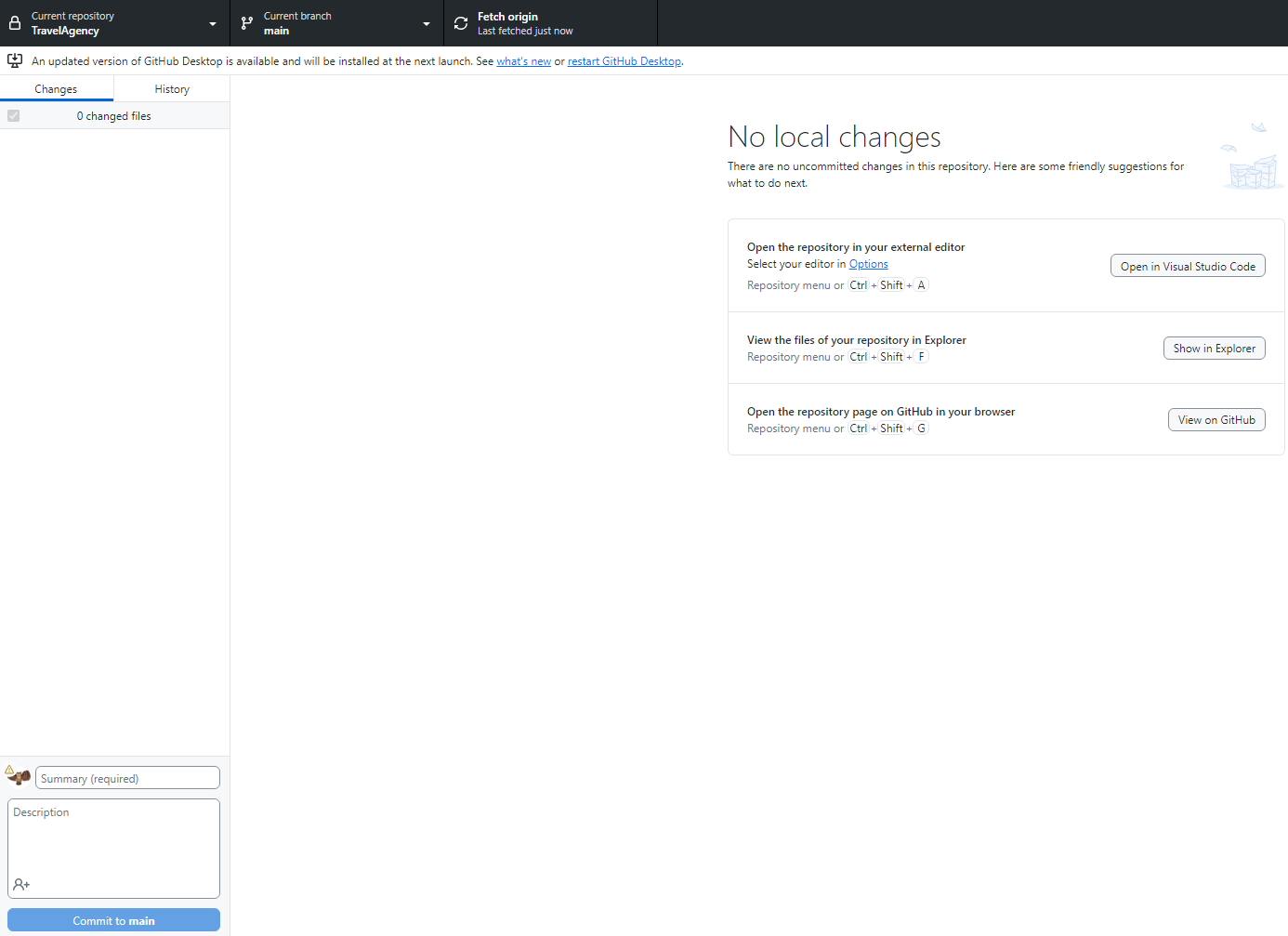


Рис.15. Вид приложения с открытым репозиторием.

Шаг 5: При добавлении нового файла, будут показаны строки, а то есть плюсики (+), это значит, что в репозиторий и во внутренней структуре файла были добавлены новые строки кода.

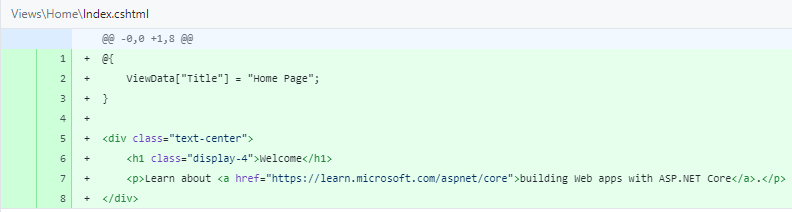


Рис.16. Вид при добавлении нового файла Index.cshtml

Шаг 6: Выбрав файл для публикации, нам необходимо его опубликовать, и подтвердить его публикацию в репозиторий.

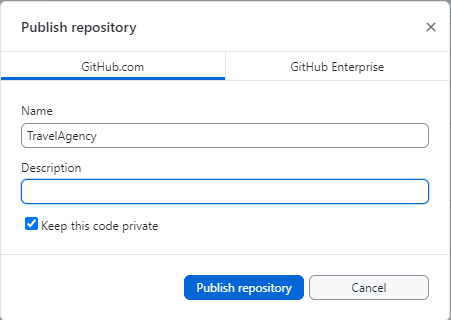


Рис.17. Подтверждение публикации репозитория

Шаг 7: После того, как мы отправили репозиторий, в декстопе будут отображены некоторые исправления в файлах. Необходимо о них знать, и видеть.



Рис.18. Просмотр исправлений

Шаг 8: Далее нам необходимо проверить, отправились ли файлы, которые мы отправляли, для этого переходим в браузер, заходим в ГитХаб, потом находим наш репозиторий, который мы опубликовали, и просматриваем файлы (его содержимое).

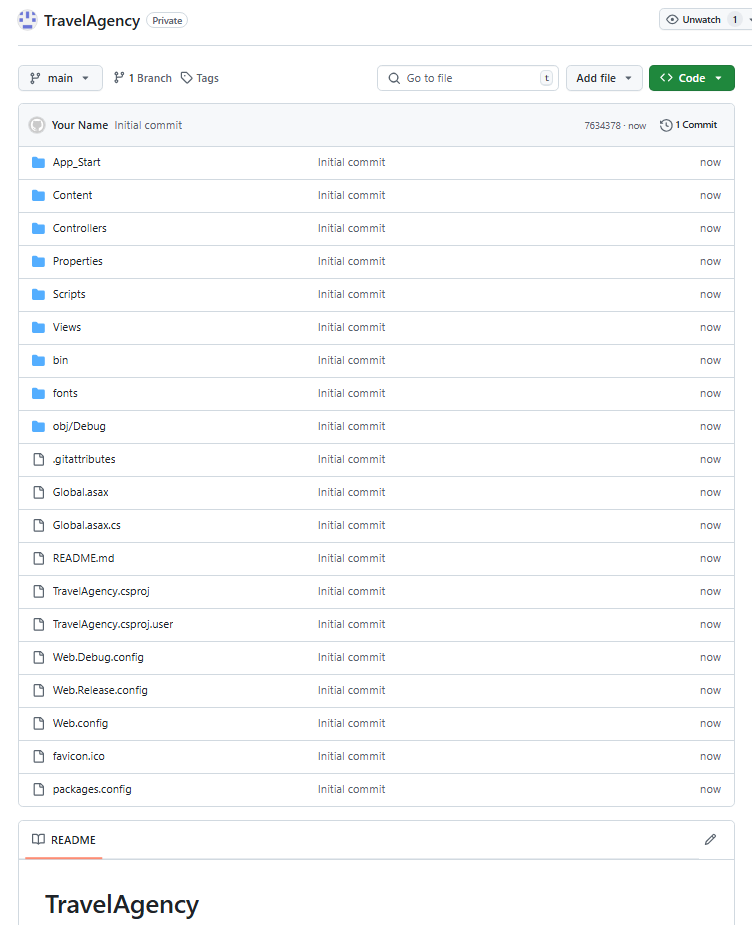


Рис.19. Проверка отправки изменений

**Вывод:** Я научился пользоваться Github и узнал его функионал.

**04.11.2024**

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

**Ход работы:**

Шаг 1: Первым делом я удалил папку `Models` из проекта, так как модели будут находиться в отдельном месте. Затем создал три новые папки: `DAL`, `Domain` и `Service`. Это позволяет четко разделить ответственность каждого слоя.

Шаг 2: В каждой из созданных папок я добавил проекты типа «библиотека классов». Это важно для того, чтобы каждый слой был независимым и модульным. Теперь у меня есть следующие проекты:

- `YourProject.DAL`

- `YourProject.Domain`

- `YourProject.Service`

Шаг 3: Далее я начал работать над определением сущностей. Я создал бизнес-объекты, которые отражают основные правила моего приложения. Например, сущностью может быть `User`, который содержит основные атрибуты и методы, связанные с пользователем. Это важно, так как сущности должны оставаться неизменными при изменениях во внешних слоях.

Шаг 4: На следующем этапе я определил сценарии, которые будут реализовывать бизнес-правила. Они инкапсулируют потоки данных между слоями сущностей и адаптеров. Я убедился, что изменения в этом слое не затрагивают сущности, и они изолированы от внешних изменений, таких как изменения в базе данных или пользовательском интерфейсе.

\Шаг 5: Затем я перешел к реализации интерфейс-адаптеров. Это слой, который будет преобразовывать данные между форматом, удобным для сценариев и сущностей, и форматом, используемым для хранения, например, в базе данных. Я осознал, что в этом слое будет реализована архитектура MVC, где контроллеры взаимодействуют со сценариями, а модели передают данные.

Шаг 6: Я начал интегрировать контроллеры и представления, определив, как они будут взаимодействовать со сценарием. Важно, чтобы контроллеры не знали о конкретных реализациях представлений, а только о интерфейсах, что позволяет соблюдать правило зависимостей.

Шаг 7: Далее я определил, что наружный слой будет включать фреймворки, базы данных и интерфейс пользователя. Я старался минимизировать код в этом слое, оставляя его лишь для взаимодействия с внутренними кругами.

Шаг 8: Пересечение границ

Я обратил внимание на то, как контроллеры и представления взаимодействуют с сценариями. Я применил принцип инверсии зависимостей, чтобы контроллеры обращались к сценариям через интерфейсы, а не напрямую. Это позволяет соблюдать правило зависимостей, где внутренние круги не знают о внешних.

Шаг 9: Передача данных между слоями

Я осознал, что при передаче данных между слоями нужно использовать структуры данных, которые изолированы от внутренней реализации. Я избегал передачи сущностей напрямую и старался использовать объекты передачи данных (DTO).

\Шаг 10: Подготовка к реализации

Наконец, я подготовился к реализации проекта, удалив папку `Models`, создав необходимые папки и проекты. Теперь у меня есть четкая структура, позволяющая организовать код и сделать его поддерживаемым и гибким.

Этот подход к разработке с использованием чистой архитектуры и MVC позволил мне четко организовать код и сделать его более устойчивым к изменениям.

**Вывод:** В этой работе я освоил организацию проекта с использованием принципов чистой архитектуры и паттерна MVC. Понял важность разделения ответственности между слоями, что позволяет создавать гибкую и поддерживаемую структуру. Также научился применять правило зависимостей для изоляции бизнес-логики и правильно передавать данные через объекты передачи данных (DTO). Это значительно повысило мои навыки разработки и понимание архитектурных принципов.

**04.11.2024**  
**Тема:** Frontend разработка основной страницы: написание шапки сайта, секции «О нас»

**Ход работы:**

Шаг 1: Разметка шапки сайта

В первую очередь я открыл файл \_Layout.cshtml и добавил разметку для шапки сайта. Шапка состоит из двух частей:

В верхней части шапки расположены логотип и кнопки «Войти» и «Зарегистрироваться».

В нижней части размещено меню.

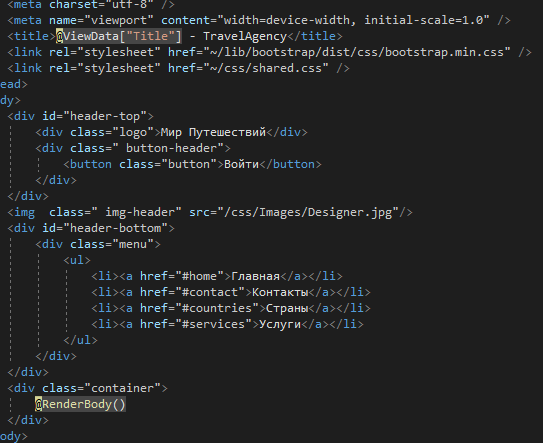


Рисунок 20 – Пример кода разметка шапки.

Шаг 2: Создание и подключение CSS

Для того чтобы сайт выглядел аккуратно, я удалил старый файл site.css из папки css и создал новый файл стилей shared.css.

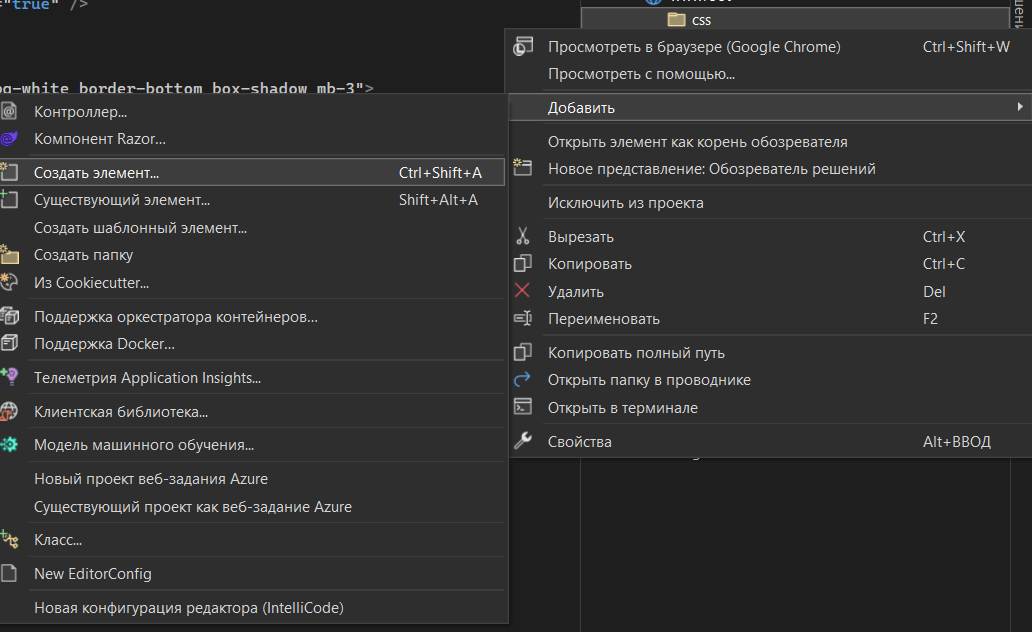


Рисунок 22 – открытие окна «добавление нового элемента»

Для этого я щелкнул правой кнопкой мыши по папке css и выбрал пункт "Добавить/Создать элемент…". В открывшемся окне выбрал C#/ASP.NET Core/Веб/Содержимое/Таблица стилей и назвал файл shared.css.

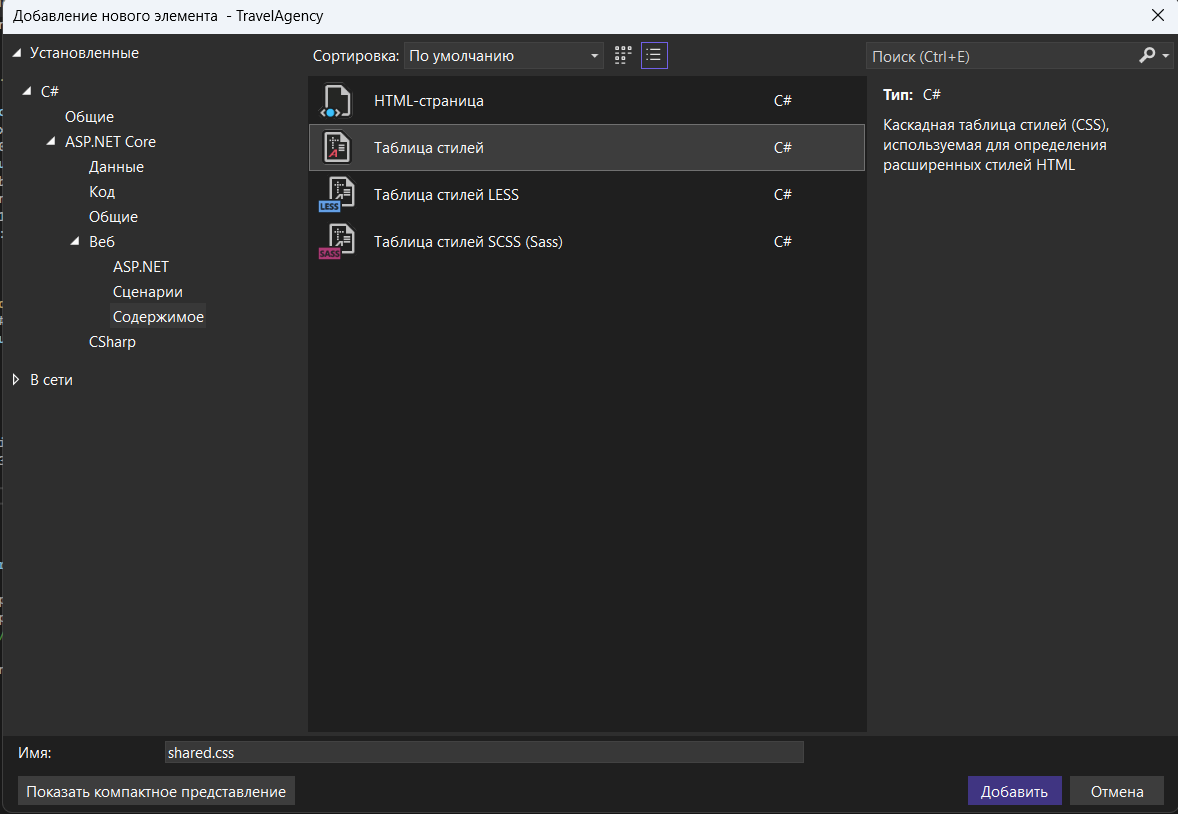


Рисунок 23 – добавление файла таблицы стилей.

Шаг 3: Применение базовых стилей

В новом файле shared.css я прописал базовые стили для элементов html, \*, \*::before, \*::after и body для нормализации отступов и шрифтов на всех страницах. Это позволило избавиться от лишних отступов и задать единый шрифт для всего сайта.

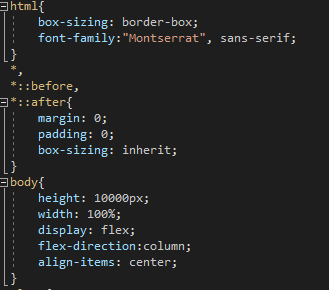


Рисунок 24 – пример стилей.

Шаг 4: Стилизация логотипа

Так как в этом проекте логотип представляет собой текст, я добавил стили для элемента logo. Он теперь имеет более крупный шрифт и жирное начертание.

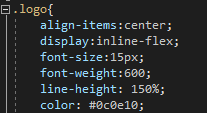


Рисунок 25 – пример стилей.

Шаг 5: Стилизация шапки сайта

Шапка сайта была разделена на две части, и для каждой из них я прописал отдельные стили:

Для верхней части шапки, в которой находятся логотип и кнопки «Войти» и «Зарегистрироваться», я использовал Flexbox для выравнивания элементов по бокам.

Пример стилей для верхней части шапки:

Для нижней части шапки, где расположено меню, я применил стили для выравнивания ссылок по центру и добавления эффектов при наведении.

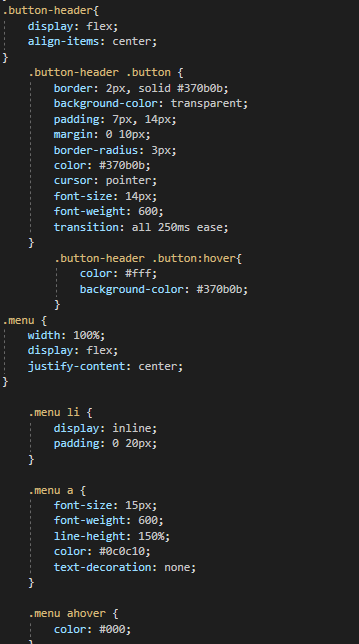
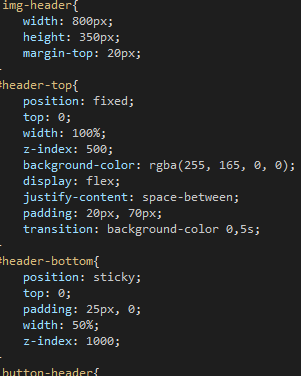


Рисунок 26,27 – пример стилей.

Шаг 6: Стилизация кнопок

Для кнопок в верхней части шапки я прописал стили, чтобы они выглядели как интерактивные элементы.

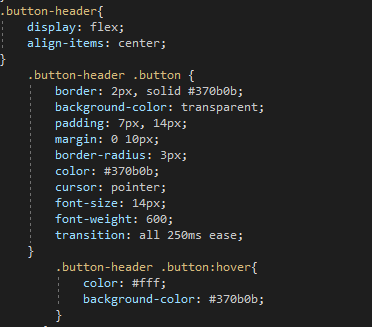


Рисунок 28 – пример стилей для кнопок.

Шаг 7: Создание эффекта прокрутки для шапки

Чтобы создать эффект, при котором шапка изначально будет прозрачной, а затем постепенно становиться оранжевой при прокрутке страницы, я добавил JavaScript.

Для этого создал новый файл shared\_script.js, выбрав в проекте "Добавить элемент" → "Файл JavaScript", и прописал скрипт для изменения цвета шапки при прокрутке.

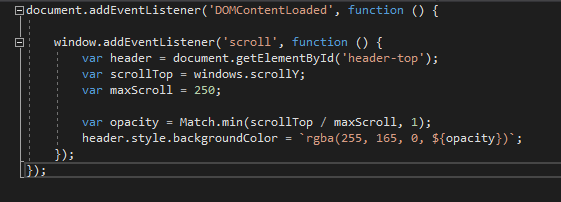


Рисунок 29 – пример реализации кода.

Шаг 8: Стилизация контейнера

Для оформления основного контента страницы я добавил стили для контейнера, в котором будет располагаться основной контент сайта.

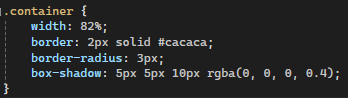
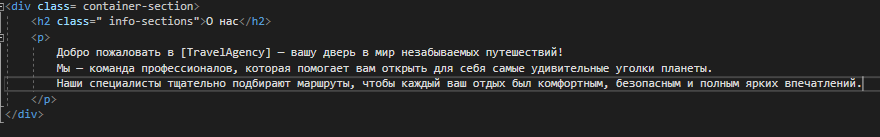


Рисунок 30 – стили для контейнера.

Шаг 9: Разметка и стили для секции «О нас»

Для создания страницы с информацией «О нас», я создал новый файл SiteInformation.cshtml в папке Views/Home. В этом файле прописал разметку секции «О нас», добавив текст и заголовок.



. Рисунок 31 – пример кода

Шаг 10: Результат

После выполнения всех шагов и настройки стилей, проект выглядит следующим образом:

Шапка сайта теперь состоит из логотипа, кнопок и меню с эффектом изменения цвета при прокрутке.

Секция «О нас» оформлена с использованием аккуратных стилей и хорошо интегрирована в общую структуру страницы.

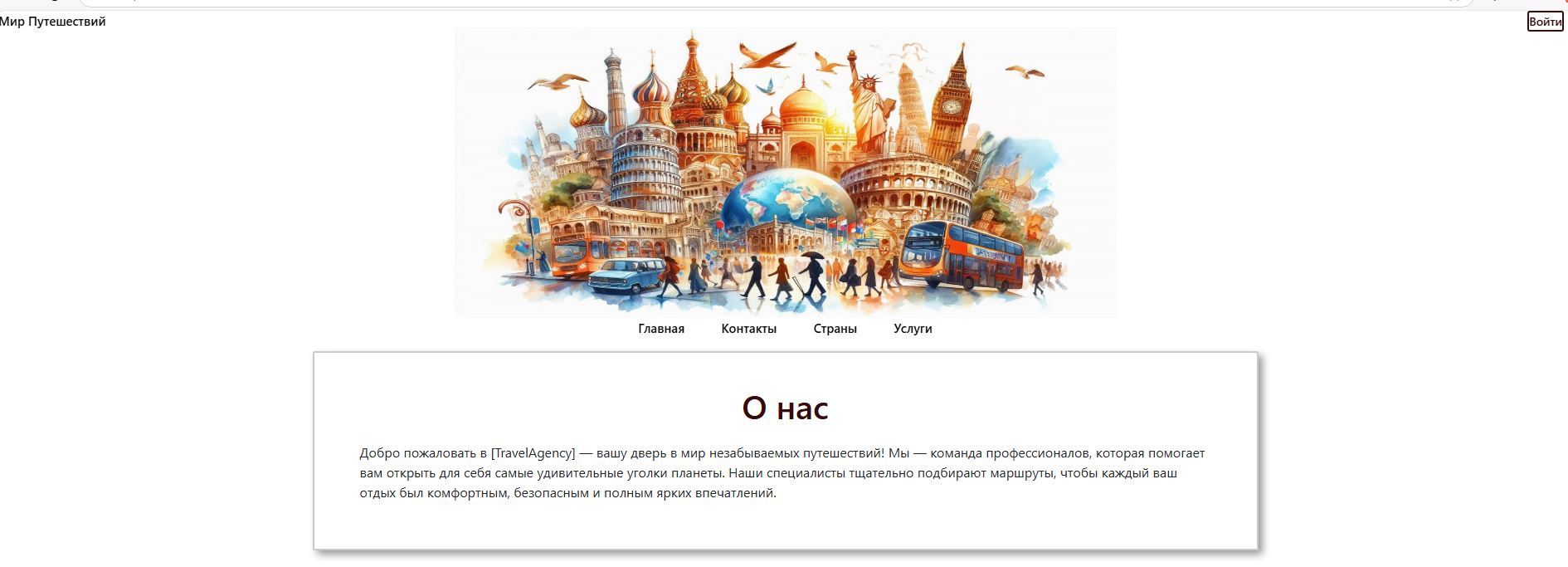


Рисунок 32 - результат.

**Вывод:** В процессе работы я научился. Работать с разметкой и стилями для создания адаптивных и интерактивных элементов на сайте. Применять JavaScript для создания эффектов при прокрутке страницы. Структурировать сайт, разделяя его на компоненты, такие как шапка и секция «О нас», с соответствующей стилизацией. Использовать методы и технологии для создания динамичного и функционального интерфейса.

**06.11.24**

**Тема:** *Frontend* разработка основная страница: написание формы вход/регистрация.

**Ход работы:**

Шаг 1: В данной главе мы реализуем внешний вид формы входа и регистрации. Создадим новое представление \_LoginRegistrationPartial.cshtml в папке \Views\Shared.

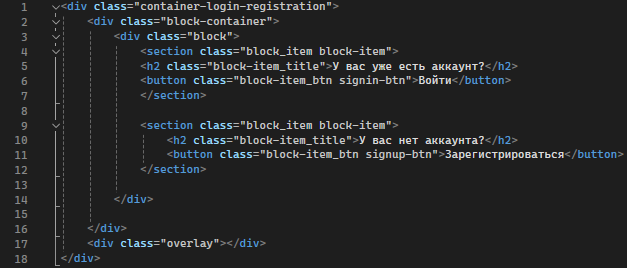


Рис.33.Контейнер формы

Шаг 2: А также сразу в файле \_Layout.cshtml пропишем строку как показано на рисунке 38.



Рис.34. Вызов частичного представления

Шаг 3: 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.

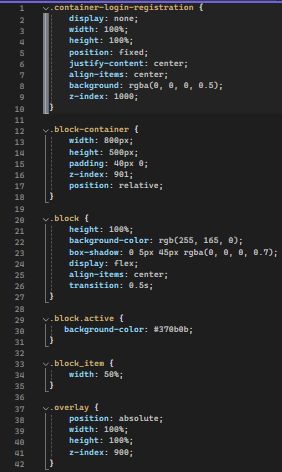


Рис.35. Стили для контейнера формы

Шаг 4:  Здесь так же создаем новый файл js с названием login\_and\_registration\_script. Для того что бы скрипт срабатывал в нашей кнопке.

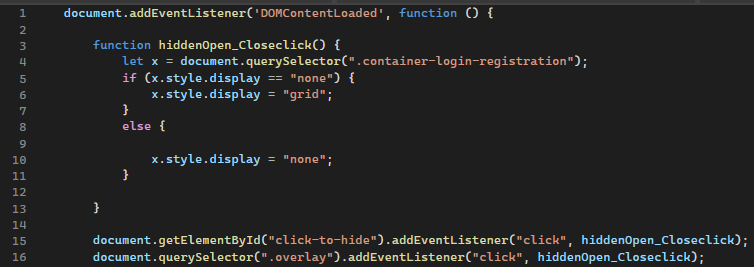


Рис.36. Стили для контейнера формы

Шаг 5:  Далее зададим стили для текста и кнопок левой и правой части контейнера.



Рис.37. Стили для блоков контейнера

Шаг 6: Создаём разметку страницы.



Рис.38. Разметка формы входа

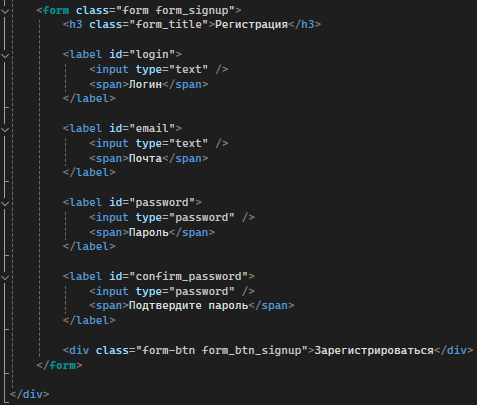


Рис.39. Разметка формы регистрация

Шаг 6: Создаём стили для форм. Дополняем качественными полями и шрифтами наши формы.

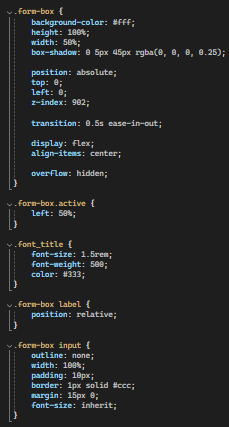


Рис.40. Стили для форм

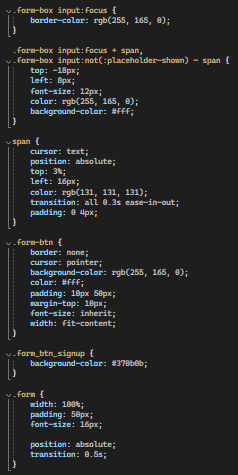


Рис.41. Стили для кнопок и полей ввода

Шаг 7: Используем другие стили для более плавного и красивого перехода между панелями.

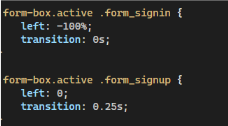
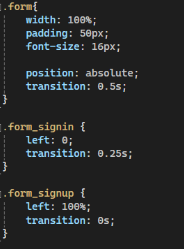


Рис.42.Стили для смены формы

Шаг 8: Для того чтобы мы могли увидеть при нажатии на кнопку форму, нам необходимо прописать скрипт, показанный на рисунке 47.

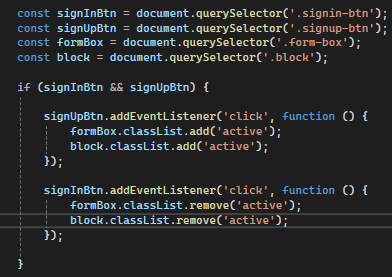


Рис.47.Скрипт для открытия формы от одной к другой

Шаг 9: В завершение у нас должно получиться так:

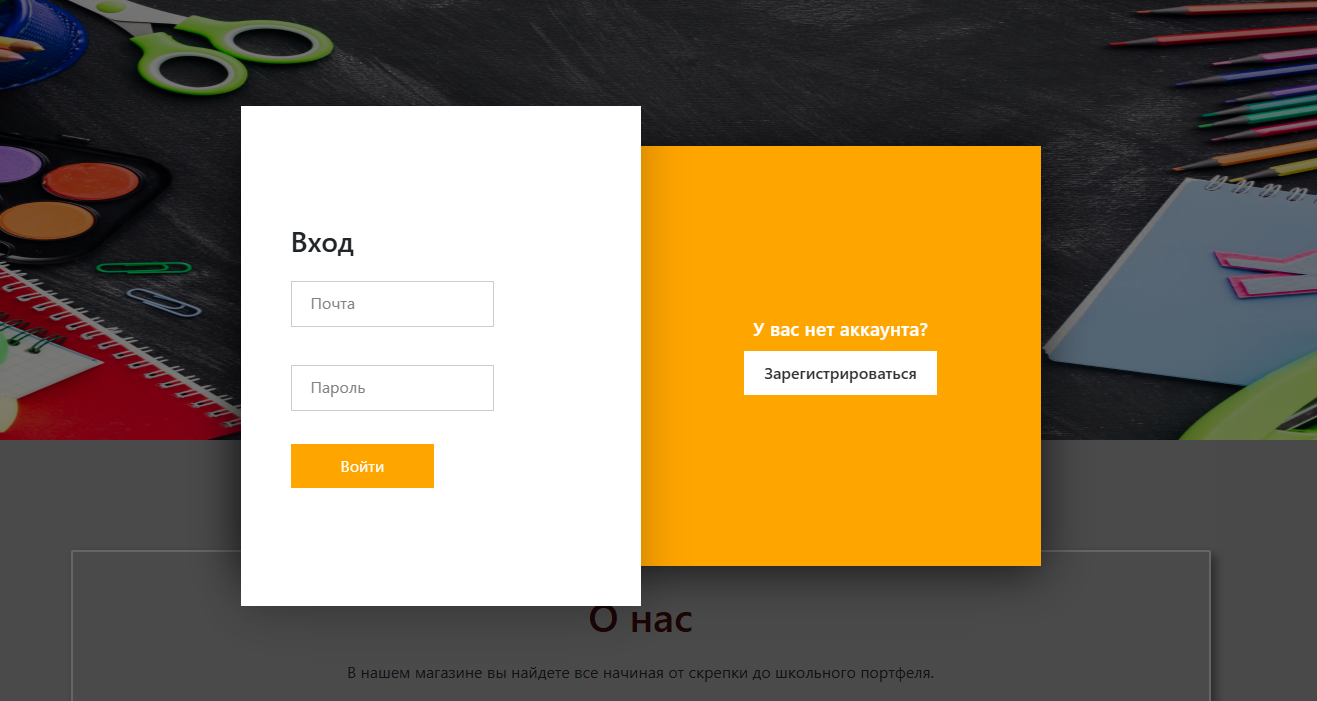


Рис.48.Результат работы

**Вывод:** В ходе выполнения этой практической работы я освоил процесс подключения различных стилей к формам, научился настраивать их внешний вид, а также работать с компонентами и привязывать их к конкретным ID кнопок. Например, я реализовал функциональность "click-to-hide", где кнопка, привязанная к определенному ID, скрывает или отображает элемент на странице.

**07.11.24**

**Тема:** *Frontend* разработка, основная страница: написание секции «Написать сообщение» а также подвал сайта. Оформление отчетной документации.

**Ход работы:**

Шаг 1: Разместим ниже секции с контактами секцию с отправкой сообщения на почту. Напишем для данного задания разметку, а также стили для этой разметки. Этот шаг показан на рисунках 49-50.

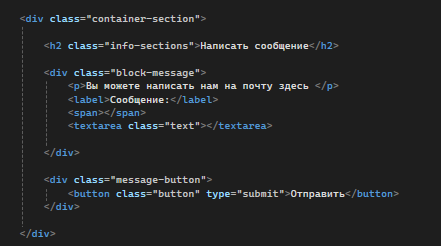


Рис.49. Секция «Написать сообщение» разметка

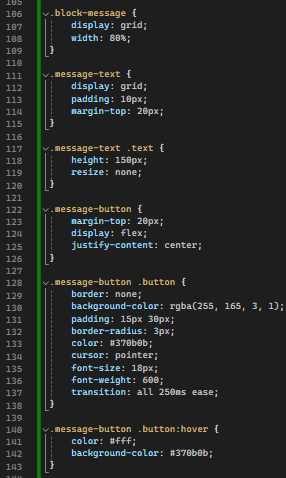


Рис.50. Секция «Написать сообщение» стили

Шаг 2: Пропишем для подвала нашего сайта стили. Для этого переходим в файл Home.css и прописываем туда следующий код:

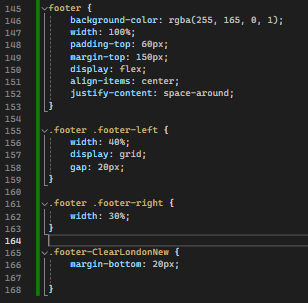


Рис.51. Стили для подвала сайта

Шаг 3: Далее пропишем footer для нашего подвала. Чтобы одна его часть располагалась в правой стороне, то есть, то что нужно нам, а что необходимо расположить в левой части (например значок защиты авторских прав), соответственно в другой стороне.

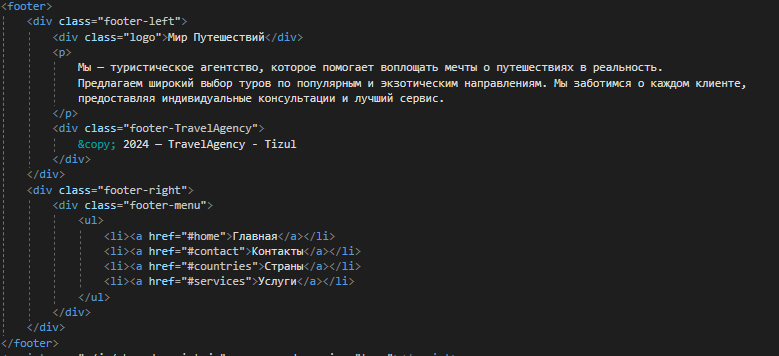


Рис.52. footer для подвала сайта

Шаг 4: Если всё было прописано правильно, то в результате у нас должна получиться данная структура подвала:

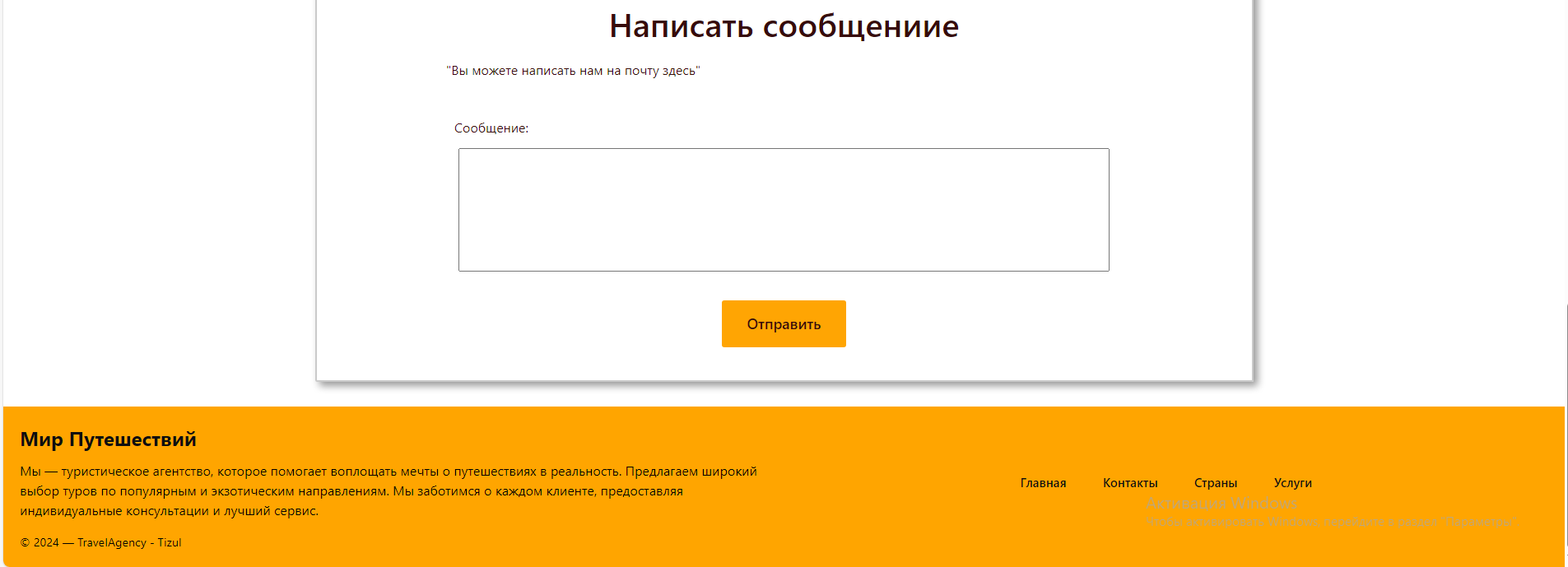


Рис.53. Результат работы

**Вывод:** В ходе выполнения этой практической работы я научился работать с секциями подвала сайта, осваивая их стилизацию и структуру. Также я понял, какие ошибки могут возникнуть при разработке подвала, и научился эффективно их устранять.

**08.11.24**

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

**Ход работы:**

JavaScript предоставляет возможность отправлять асинхронные запросы на сервер, позволяя обновлять данные на странице без её перезагрузки. Это особенно полезно для таких операций, как отправка заказов, получение информации о пользователе, обновление контента и другие.

Шаг 1: Метод fetch() позволяет отправлять сетевые запросы и получать ответы в формате Promise. Мы будем использовать его для отправки данных в формате JSON. Для отправки JSON в теле запроса мы указываем заголовок Content-Type: application/json.

Заметим, что по умолчанию, при отправке строки в теле запроса (body), заголовок Content-Type будет иметь значение text/plain;charset=UTF-8. Однако, так как мы отправляем данные в формате JSON, мы должны указать правильный тип содержимого — application/json. Для этого в параметре headers запроса укажем этот тип.

Шаг 2: Для начала создадим модели, которые будем использовать для передачи данных о пользователе при входе и регистрации. В проекте InteractivePortal.Domain добавим папку ViewModels/LoginAndRegistration. Внутри создадим два класса: LoginViewModel и RegisterViewModel. Каждый из этих классов будет содержать необходимые свойства и атрибуты для проверки валидности данных, как показано на рисунке 54 и 55.

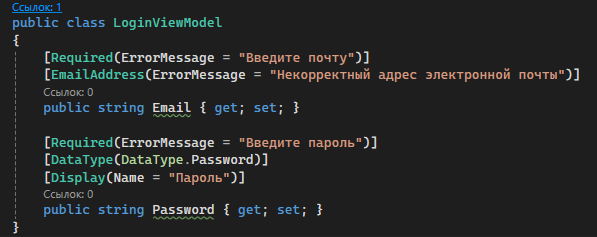


Рис.54. Класс *LoginViewModel*

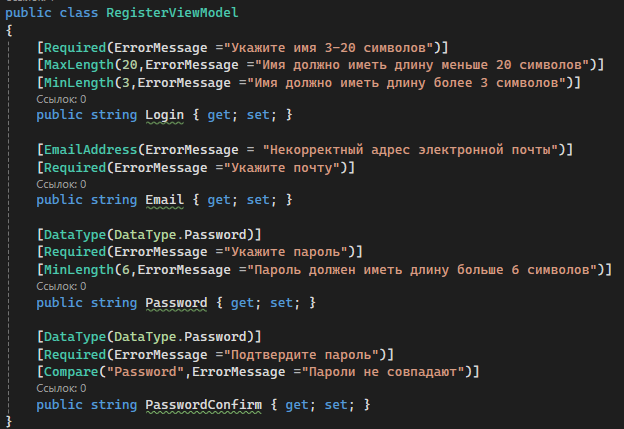
****

Рис.55.  Класс *RegisterViewModel*

Шаг 3: Чтобы проектInteractiveCources.Web мог использовать модели из проекта InteractivePortal.Domain, необходимо установить ссылку на этот проект. Для этого нажмите правой кнопкой мыши на проект InteractiveCources и выберите опцию "Добавить ссылку на проект". В открывшемся окне диспетчера ссылок выберите нужный проект и нажмите OK, как показано на рисунке 56 и 57.

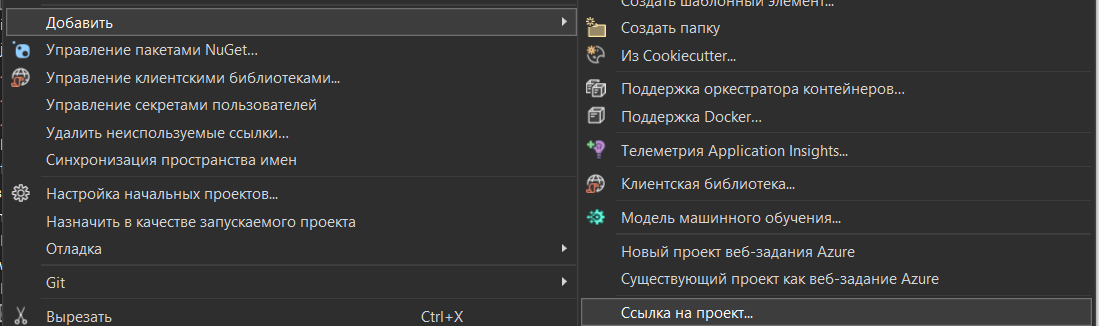
****

Рис.56.  Добавить/Ссылка на проект

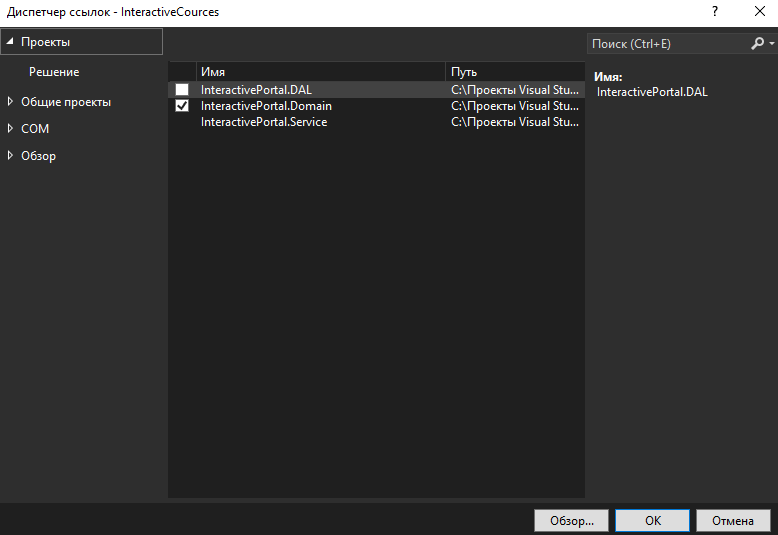
****

Рис.57.  Диспетчер ссылок

Шаг 4: После того как ссылка на проект будет добавлена, мы сможем использовать модели в контроллере. Создадим метод Login в контроллере, который будет отвечать за обработку данных, полученных с формы входа, как показано на рисунке 58.

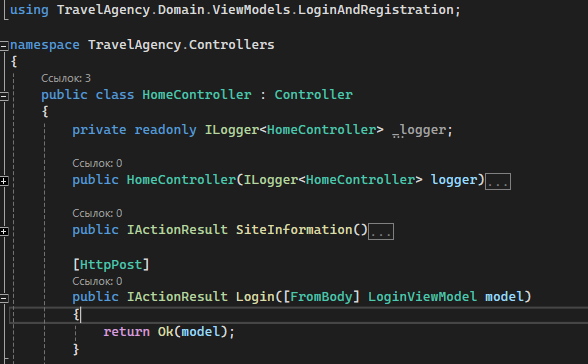


Рис.58.  Действие *Login*

Шаг 5: В файле login\_and\_registration\_script.js добавим код для обработки нажатия на кнопку "Войти". При нажатии на кнопку мы будем собирать необходимые данные для отправки через fetch: определим URL, метод и тело запроса, как показано на рисунке 59.

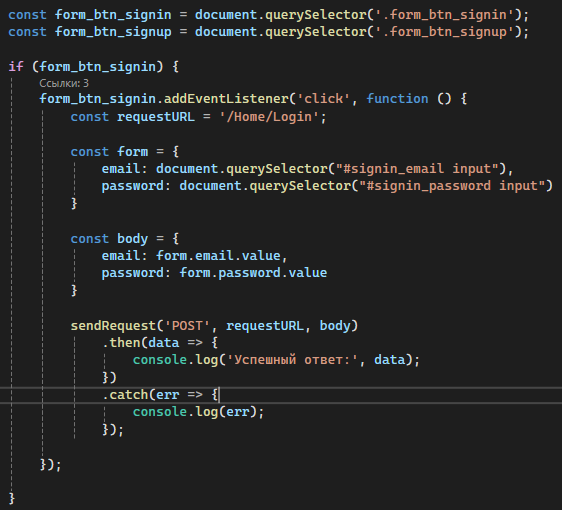
****

Рис.59.  Обработка нажатия на кнопку войти

Шаг 6: Далее, в той же части скрипта, реализуем метод sendRequest, который будет отправлять запрос на сервер, как показано на рисунке 60. Мы сможем отследить процесс передачи данных, используя точки останова или посмотреть результаты в консоли браузера через инструменты разработчика.

****

Рис.60.Функция *sendRequest*

Шаг 7: После получения данных с клиента, в контроллере необходимо будет провести проверку на валидность, используя атрибуты в модели. Это можно сделать, проверив состояние модели, как показано на рисунке 61.

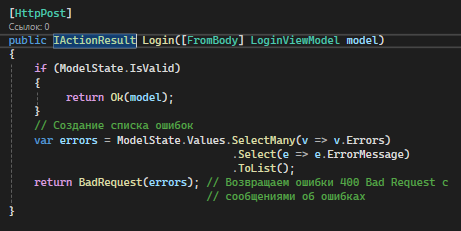


Рис.61. проверка валидности

Шаг 8: Если в процессе валидации данных возникнут ошибки, их нужно отобразить на клиенте. Для этого создадим функцию displayErrors(), которая будет вызываться после нажатия на кнопку. Она очистит предыдущие ошибки и отобразит новые в виде элементов с классом error. Также в разметку формы добавим контейнер для ошибок, как показано на рисунке 62 и 63. Внешний вид ошибок можно настроить с помощью стилей, как показано на рисунке 64.

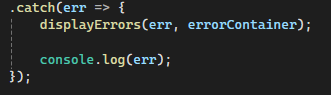


Рис.62. Вызов функции *displayErrors*

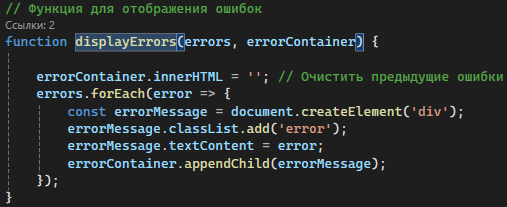


Рис.63. Функция *displayErrors*

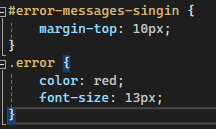


Рис.64. Стили для отображения ошибок

Шаг 9: Когда данные успешно проходят проверку и запрос успешно выполнен, необходимо обновить страницу. Для этого используем метод location.reload(), который вызовется в функции CleaningAndClosingForm, как показано на рисунке 65 и 66. Эта функция также очистит все поля формы и закроет ее.

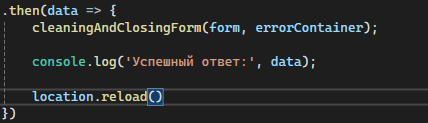


Рис.65. Вызов функции *CleaningAndClosingForm*

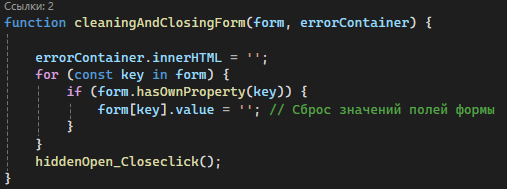


Рис.66. Функция *CleaningAndClosingForm*

Шаг 10: Аналогично тому, как мы обрабатываем форму входа, реализуем обработку для формы регистрации. Добавим обработчик для кнопки "Зарегистрироваться", как показано на рисунке 67. Также создадим соответствующее действие в контроллере для регистрации пользователя, как показано на рисунке 68.

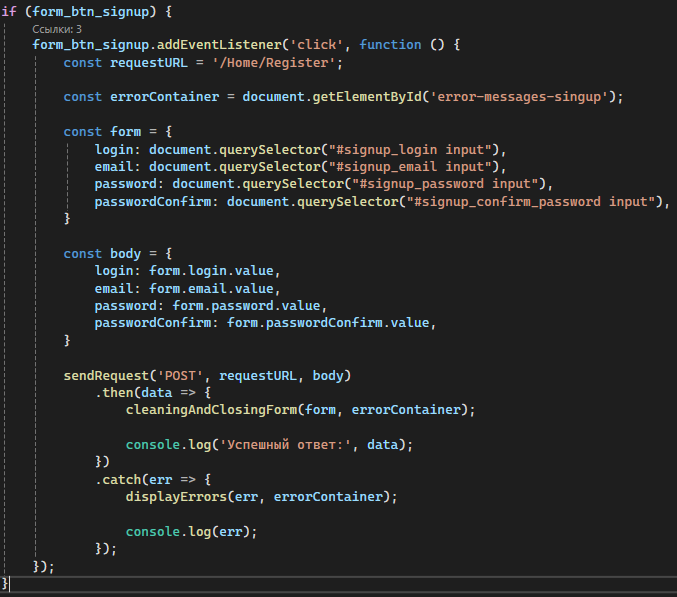


Рис.67. Обработка нажатия на кнопку зарегистрироваться

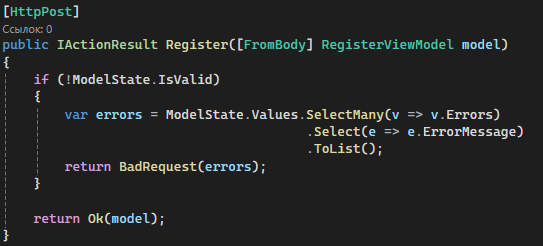


Рис.68. Действие *Register*

**Вывод:** Используя метод fetch(), я реализовал асинхронную отправку данных на сервер, что позволяет обновлять информацию на странице без её перезагрузки. Это значительно повышает производительность, улучшает пользовательский опыт и делает взаимодействие с сайтом более быстрым и плавным.

**13.11.2024**

**Тема**: Создание адаптивного интерфейса для веб-страницы с использованием медиа-запросов CSS для улучшения отображения на мобильных устройствах.

#### Ход работы:

**Шаг 1: Понимание структуры viewport и медиа-запросов:**

**Viewport** – это область на экране, видимая пользователю, которая изменяется в зависимости от размера экрана устройства.

До появления мобильных устройств веб-страницы в основном разрабатывались для десктопных экранов. Однако с увеличением популярности мобильных устройств стало необходимым создание адаптивных сайтов, которые могут изменять своё отображение в зависимости от устройства.

**Шаг 2: Подключение медиа-запросов:**

Медиа-запросы в CSS позволяют применять стили, основанные на характеристиках устройства (например, ширина экрана). Это помогает адаптировать страницу под различные разрешения экрана.

Для начала был создан новый файл стилей media\_style.css, в который добавлены правила для адаптации элементов на мобильных устройствах.

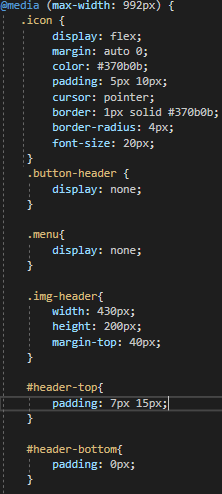


Рис. 69-Пример кода для стилизации шапки

**Шаг 3: Добавление элемента "гамбургер-меню"**

Для улучшения мобильной версии добавлен элемент "гамбургер", который скрывает меню, а при нажатии открывает его.

В HTML-разметке добавлена кнопка, которая будет управлять видимостью бокового меню на мобильных устройствах.

**Шаг 4: Стилизация гамбургер-меню:**

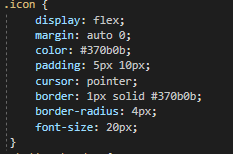


Рис. 70 – Пример стилей для гамбургер-меню

В файл shared.css добавлены стили для скрытия гамбургера на экранах больше 992px:

В media\_style.css добавлены стили, которые делают гамбургер-меню видимым на устройствах с меньшей шириной экрана:

**Шаг 5: Создание выезжающего меню:**

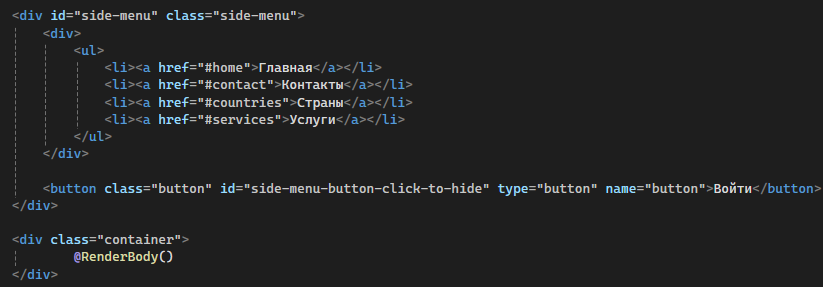


Рис. 71 – разметка появляющегося меню

Для того чтобы боковое меню не мешало пользователю при просмотре сайта, оно скрывается и выезжает при нажатии на гамбургер.

Стили для выезжающего меню добавлены в shared.css:

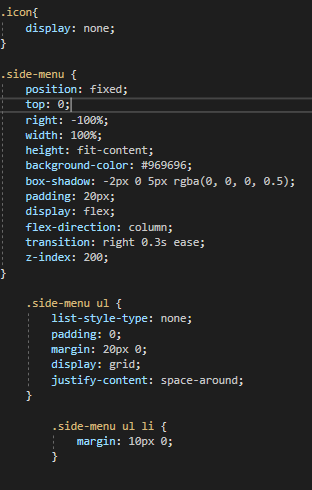


Рис. 72 - Стили для выезжающего меню

**Шаг 6: Добавление JavaScript для функционала меню:**

В chared\_script.js добавлена функция для открытия/закрытия меню при нажатии на гамбургер:

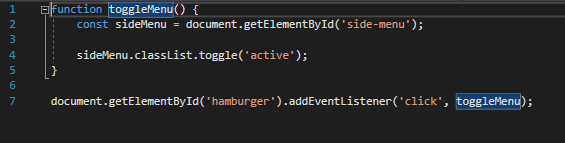


Рис. 73 – Пример скрипта

**Шаг Применение существующих стилей:**

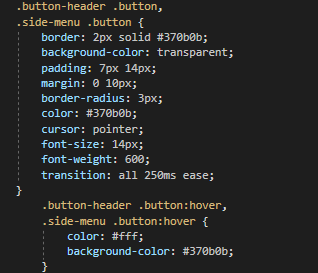


Рис. 74 – пример стилей для кнопок.  
Для того чтобы не дублировать стили, использованные ранее для других частей страницы, были добавлены нужные классы и комбинированные стили. Это позволило использовать уже прописанные правила для различных элементов в новом контексте.

**Добавление обработчиков для взаимодействия с кнопкой:**

В файле login\_and\_registration\_script.js добавлен обработчик для кнопки, чтобы она корректно реагировала на клики:



Рис. 75 – пример реализации кода

Вывод: Я научился использовать медиа-запросы CSS для создания адаптивных веб-страниц, реализовывать гамбургер-меню для мобильных устройств и адаптировать элементы интерфейса под разные размеры экранов, а также использовать JavaScript для взаимодействия с элементами страницы.

**Тема:** Создание базы данных в СУБД *PostgreSQL.*

**Ход работы**

Шаг 1: Мы познакомились с *PostgreSQL* в теории пришло время поработать с данной СУБД на практике. Чтобы начать, необходимо открыть нашу СУБД, ярлык этого приложения называется *pgAdmin*.

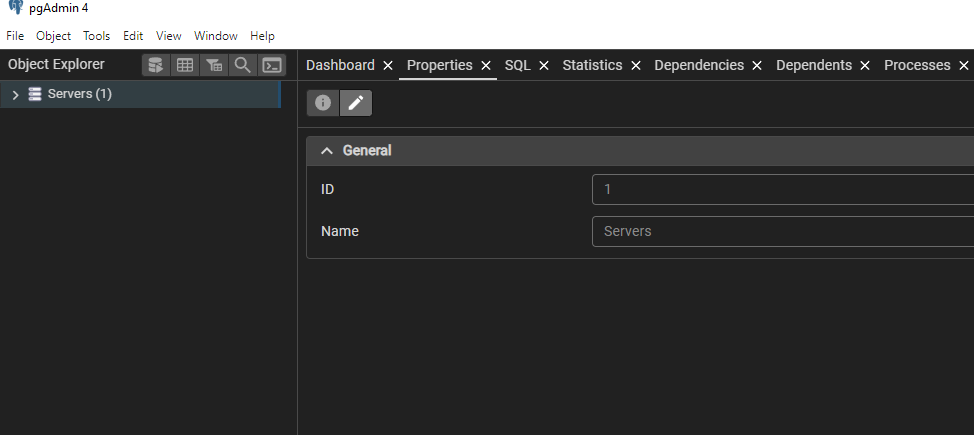


Рис.76. Так выглядит начальное окно *pgAdmin*

Шаг 2: Нужно раскрыть список Servers и ввести пароль если он имеется. Далее, как показано на рисунке 106 выбираем Создать/База данных… В появившемся окне вводим название travel\_agency и нажимаем кнопку сохранить.

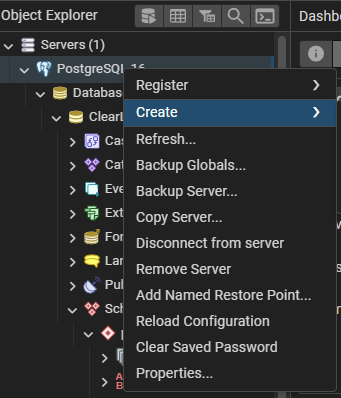


Рис.77. Создание базы данных

Шаг 3: Далее находим пункт «Схемы» в созданной БД в нем нажимае правой кнопкой мыши на «Таблицы» и выбираем Создать/Таблица…

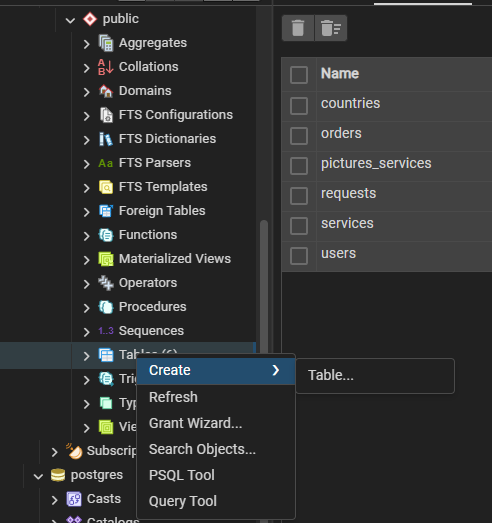


Рис.78. Создание базы данных

Шаг 4: Создаём таблицы которые будут нам необходимы в будущей разработки продукта. Во вкладке General задаем название таблицы. Далее во вкладке Столбцы в верхнем правом углу нажимаем плюсик и прописываем название и тип столбца. Также для всех таблиц действует принцип что поле id является первичным ключом. Добавляем такие таблицы как:

1. Таблица пользователей (users) будет хранить информацию зарегистрированного пользователя, а именно столбцы: id, логин, пароль, почта, роль, путь к картинке и дату записи.

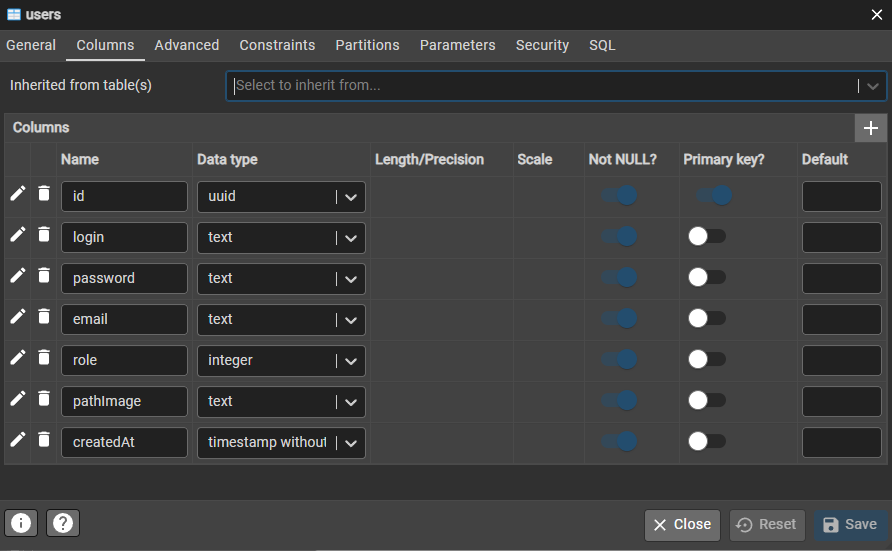


Рис.79. Создание таблицы пользователи

2. Таблица запросов (requests) хранит информацию о индивидуальных турах пользователя, а именно столбцы: id, id пользователя, описание, картинка, статус запроса, дату записи. *Таблица запросов связана с таблицей пользователи.*

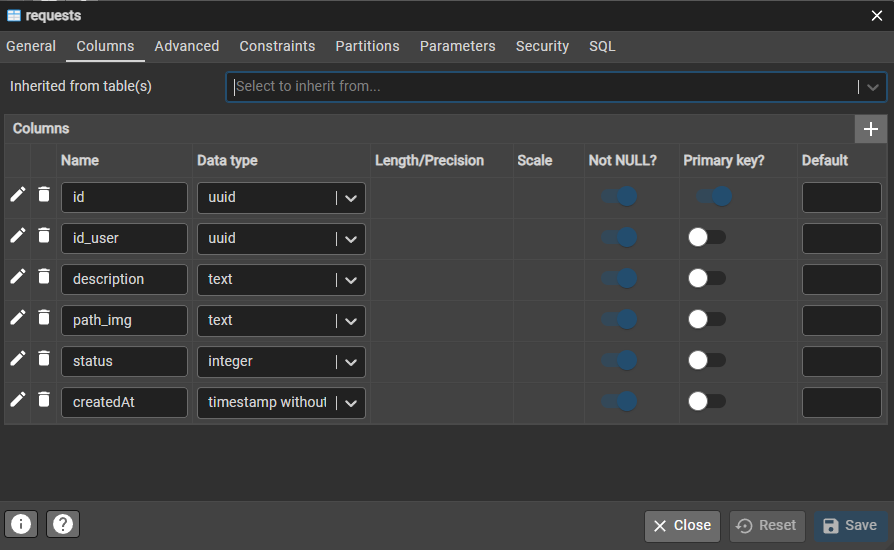


Рис.80. Создание таблицы запросы

3. Таблица заказов (orders) предназначена для записей, которые будут отображаться в корзине. Содержит столбцы: id, id пользователя, id тура название, цена, дата записи. *Эта таблица имеет отношения с такими таблицами как пользователи и services.*

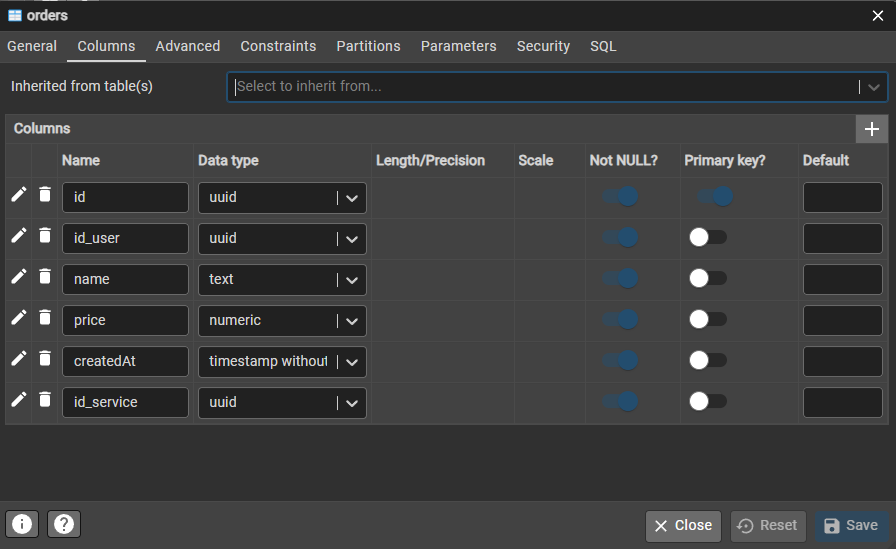


Рис.81. Создание таблицы заказы

4. Таблица услуг (services) содержит столбцы: id, id страны, город, название компании, цена уборки дома, цена уборки квартиры, цена уборки территории, цена уборки офиса, картинку и дату записи. *Эта таблица зависит от таблицы со странами.*

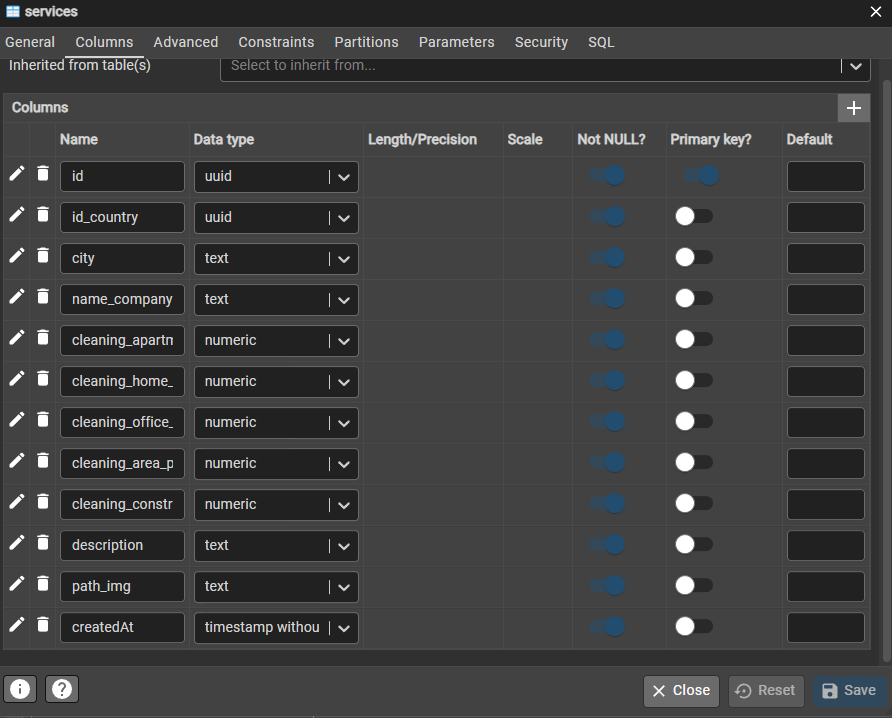


Рис.82. Создание таблицы услуг

5. Таблица стран (countries) включает в себя такие столбцы как id, название, картинку, количество туров, дата записи.

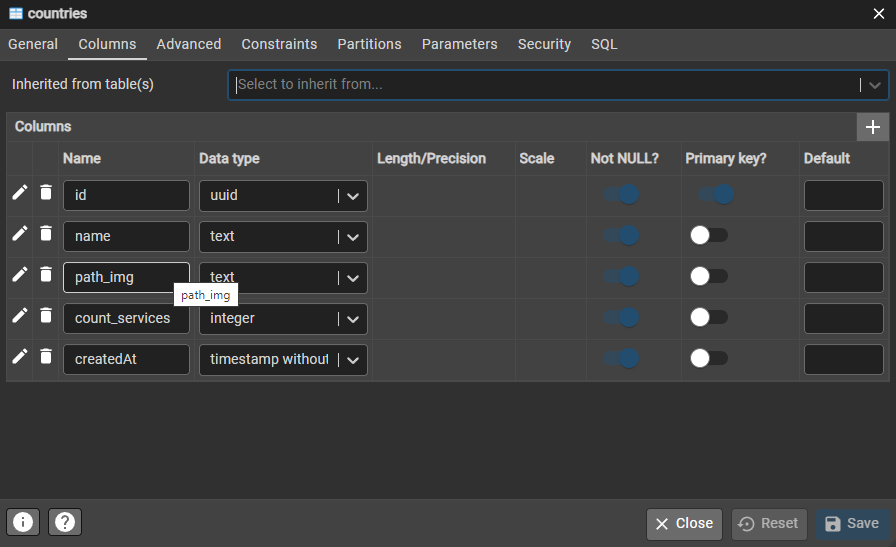


Рис.82. Создание таблицы страны

6. Таблица с картинками для конкретной услуги (pictures\_services) она содержит следующие столбцы: id, id тура, путь к картинке. *Эта таблица зависит от таблицы с турами.*

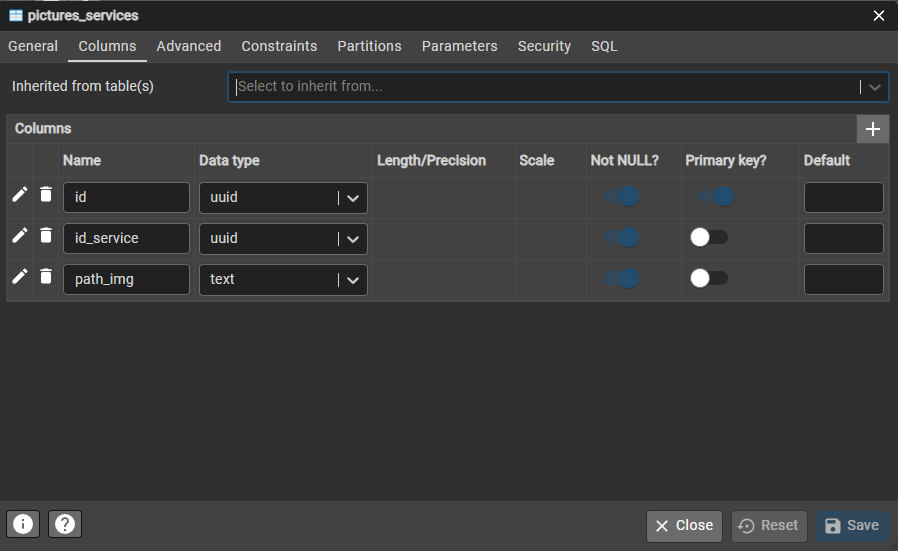


Рис.83. Создание таблицы картинки для услуг

Шаг 5: Создаём вторичные ключи в наших таблицах, для дальнейшего ссылания на конкретные элементы таблиц. Для этого нужно нажать правой кнопкой мыши по нужной таблице (request, services, orders, pictures\_services) и выбрать пункт properties. Далее нужно перейти во вкладку Ограничения/Внешний ключ после этого нажать на плюсик и в появившемся поле задать имя столбца. Внизу Нужно перейти во вкладку Столбцы и выбрать из предложенных какие столбцы связать. Пример показан на рисунке 90.

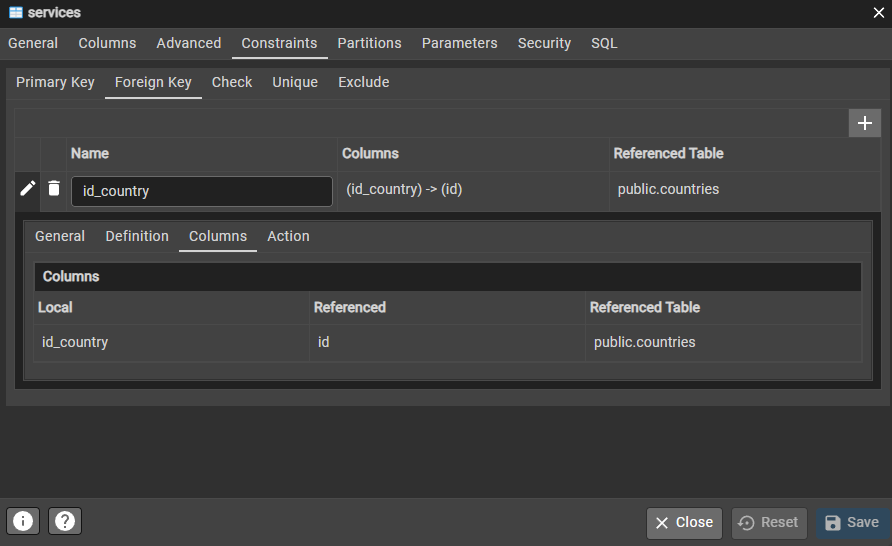


Рис.84. Задание внешнего ключа таблице

**Вывод:** На данной практической работе я научился создавать СУБД, создавать и настраивать таблицы нужным образом, подключать их взаимосвязь между собой, отправлять с помощью ссылки элемент одной таблицы на ссылающийся на другой элемент. (Например: элемент таблицы Services=id\_country🡪Counrties=id)

**14.11.24**

**Тема:** Подключение *PostgreSQL* через *appsettings.json*. Определение сущностей и их атрибутов в *C#* для базы данных с использованием *Entity Framework.*

**Ход работы**

Чтобы подключить *PostgreSQL* в качестве базы данных для проекта с использованием *Entity Framework Core*, необходимо настроить строку подключения и параметры конфигурации в *appsettings.json*, а также установить соответствующие пакеты. Давайте разберем, как это сделать, а также обсудим альтернативные способы подключения.

Для работы с *PostgreSQL* в *ASP.NET Core* и *Entity Framework Core* вам потребуется пакет *Npgsql.EntityFrameworkCore.PostgreSQL*. Установить его можно через *NuGet Package Manager* или *CLI*. Через *NuGet* это можно сделать следующим образом:

Шаг 1: Нажмите правой кнопкой мыши по проекту DAL и выберите пункт Управление пакетами *NuGet*В строке поиска введите 2. *Npgsql.EntityFrameworkCore.PostgreSQL* во вкладке Обзор должны отображаться результаты поиска3.После выберите версию и нажмите установить. Далее согласитесь со всеми пунктами.

После установки пакета откройте файл *appsettings.json* и добавьте строку подключения. Строка подключения к *PostgreSQL* должна включать такие параметры, как хост, имя базы данных, имя пользователя, пароль, и порт.

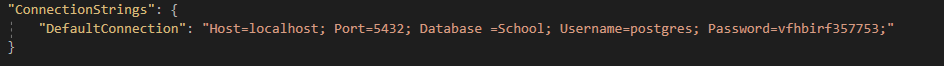


Рис.85. Cтрока подключения

Шаг 2: Добавим класс в проект *DAL* в соответствии с рисунком 92. Так же не забудем указать ссылку на этот проект для *TravelAgency*

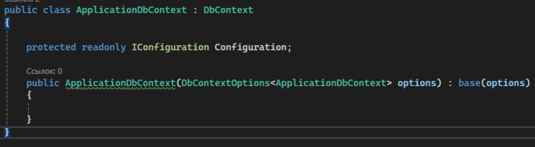


Рис.86. Класс *ApplicationDbContext*

Шаг 3: Чтобы Entity Framework знал, какую базу данных использовать, добавьте настройку в Startup.cs

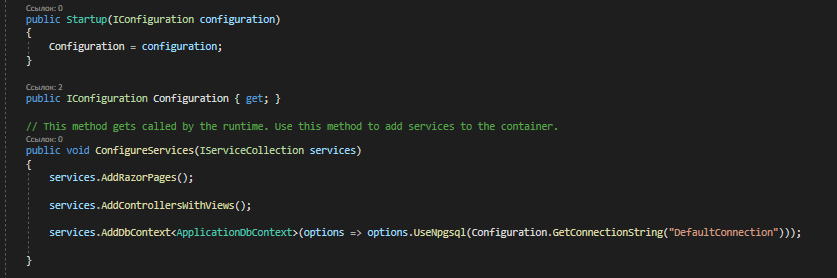


Рис.87. Настройка в *Startup.cs*

Шаг 4: Разработаем модели для базы данных. Для этого в проекте Domain создадим папку *ModelsDb* в нее добавим класс *UserDb.cs* в нем добавим поля соответствующие столбцам таблицы с пользователями.

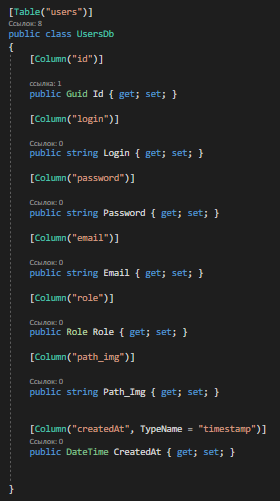


Рис.88. Модель бд таблицы пользователи

Шаг 5: Также добавим папку *Enum* и добавим в нее перечисление ролей, статуса и перечисление услуг.

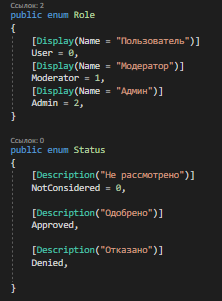


Рис.89. Перечисление ролей и статуса

Шаг 6: Реализовываем таким же образом все остальные классы.

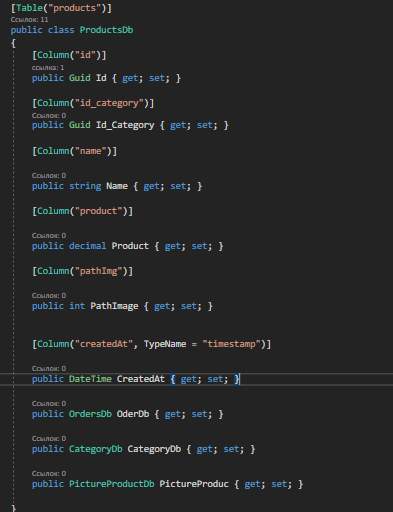


Рис.90. Перечисление класса ServiceDb

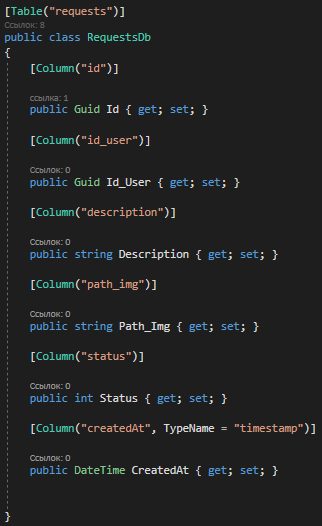


Рис.91. Перечисление класса RequestDb

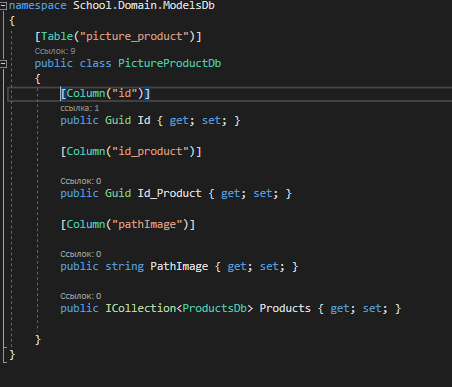


Рис.92. Перечисление класса PictureProductDb

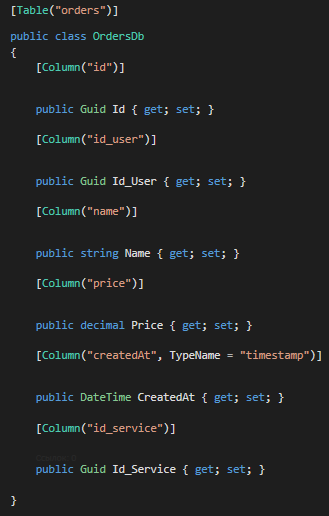


Рис.93. Перечисление класса OrderDb

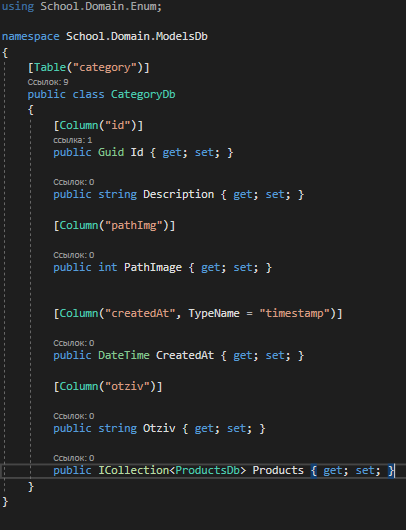


Рис.94. Перечисление класса CategoryDb

Шаг 7: Вернемся к нашему классу *ApplicationDbContext* и пропишем в нем определение моделей, которые будут представлять таблицы в базе данных. Но перед этим установим пакеты *Microsoft.EntityFrameworkCore* и *Microsoft.EntityFrameworkCore.Tools* в проект *DAL* и *Microsoft.EntityFrameworkCore.Design в проекст TravelAgency*



Рис.95. Класс содержащие все содержимое таблиц

Шаг 8: Находим вверху проекта строку с функциями. Заходим в поиск, и вводим Консоль Диспетчера Пакетов. Внизу открывается меню, куда мы заходим, и выбираем проект TravelAgency.DAL, после чего переходим к следующему шагу.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXfEtuFpBe_mu_7PPJ_B7yOvV7sC-Uv6H03Fpyb1h6X5vhoEgHEKSkqkU-HyCdBFjK984iC87D3GlSGZBizzOrHiinuuo_sKNCdngfbYeoE1L3QogIkeJNDoyofIYNMI1PJXHBN065RJgX1wizAUFQ?key=Y1UtSktmd6chbRw8S8U60G5a

Рис.102. Вкладки проекта

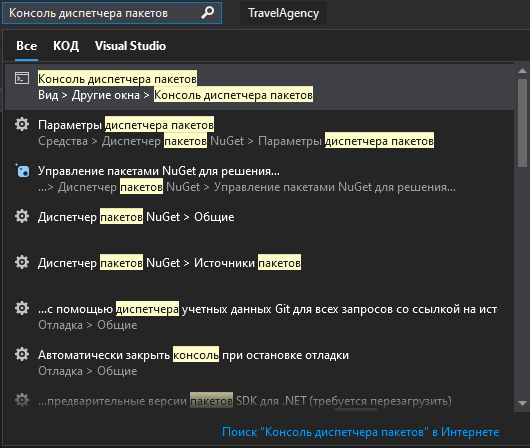


Рис.96.Консоль диспетчера пакетов

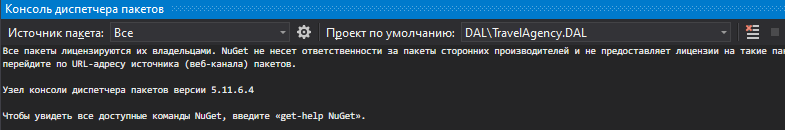


Рис.97. Поиск Консоли Диспетчера Пакетов

Шаг 9: Для проверки связи с нашей бд можем произвести миграцию в Консоли диспетчера пакетов введем следующую команду. После того как в проекте DAL появится папка с миграциями выбираем миграцию Init и удаляем содержимое методов Up и Down так как мы уже создали таблицы вручную в бд и эту миграцию делаем только для проверки соединения и инициализации. Вводим данную команду в строку консоли - Add-Migration Init

https://lh7-rt.googleusercontent.com/docsz/AD_4nXfnGAA6FaL4HCzhjFdMB1TkHodAP0JaYNpHPpZmk09Jggw3M7RkTMWvrdP2oWomt6i8sf0s7ofi7zNoy5GcsSgWfJ-AFKMfTwJcwmqKuwKS2sfXIEk-pvax8N21RbTvN5z7esw45vfL_GOmNeyVgJU?key=Y1UtSktmd6chbRw8S8U60G5a

Рис.98. Команда добавления миграции

Шаг 10: После того как содержимое было удалено необходимо выполнить еще одну команду так как команда с рисунка 105 выполняет локальное фиксирование, а чтобы отправить изменения в бд нужно выполнить команду как на рисунке 106.

https://lh7-rt.googleusercontent.com/docsz/AD_4nXchZ9Z5a53y2RjuQd597BS7sH8kUv03l1qA2QQlCNTDrKd5SaxnbHMutXOX_dhbnAH6X11b_qsW295SQNHqUS_QxuRE6DeWClETzTv8uLmg9YhdRZBCsi2A_GhH_ausDFL5z0COGptWN0y5bce7GYY?key=Y1UtSktmd6chbRw8S8U60G5a

Рис.99. Команда обновления БД (Правильный процесс update)

**Вывод:** На данной практической работе я научился работать с локальной базой данных, использовать её элементы, подключать их в файл Startup.cs. А также я понял, как работает данная структура обращения запросов, обновления базы данных и её миграция (в случае локального тестирования).