Приложение 1

**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 \_\_\_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»\_\_\_\_\_

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

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

**Оглавление**

[01.11.2024 6](#_Toc181710802)

[02.11.2024 11](#_Toc181710803)

[04.11.24 17](#_Toc181710804)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№**  **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) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_C.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:**

В данном отчёте рассматривается процесс создания веб-приложения туристического агентства с использованием архитектуры *MVC* (модель-представление-контроллер). *MVC* позволяет эффективно организовывать код, разделяя его на три основных компонента: модель (данные), представление (интерфейс) и контроллер (логика приложения). Мы будем использовать язык *C#* для разработки *backend* и *PostgreSQL* для управления базами данных, а также *HTML*, *CSS* и *JavaScript* для создания *frontend*. - упрости без использования слов *backend*, *PostgreSQl*.

1. Первым шагом является открытие среды разработки *Visual* *Studio*. Необходимо выбрать пункт «Создание проекта», как показано на рисунке 1.

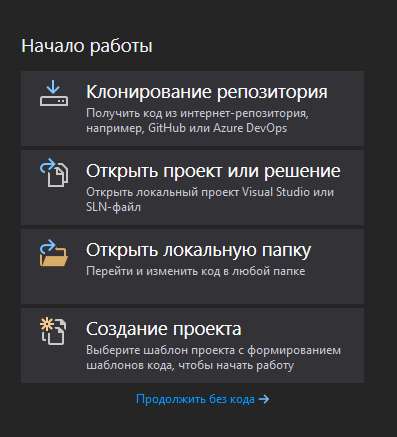


Рисунок 1 – Создание проекта

1. На следующем этапе в окне поиска вводим название шаблона: «Веб-приложение *ASP.NET Core* (модель-представление-контроллер)», как можно увидеть ниже на рисунке 2, и выбираем его. Это позволит создать проект, использующий архитектуру *MVC*. После нажимаем кнопку «Далее».

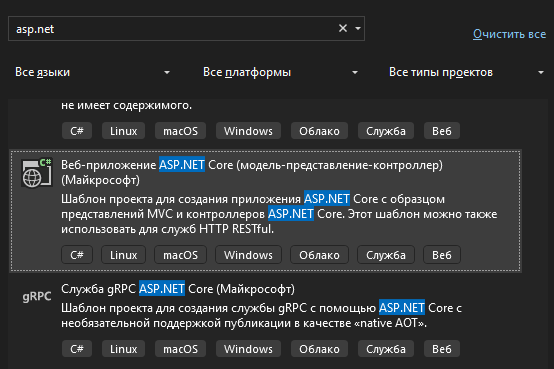


Рисунок 2 – Шаблоны

Архитектура *MVC (Model-View-Controller)* представляет собой паттерн проектирования, который помогает организовать код веб-приложений, разделяя его на три основных компонента: модель, представление и контроллер. Это разделение упрощает разработку, тестирование и поддержку приложений.

3. Переходим к настройке проекта: указываем имя *«TravelAgency»* и путь, где будет находиться проект. Ниже выбираем платформу *.NET* (рекомендуется *.NET 6* или выше), как видно на рисунке 3. Это важно для обеспечения совместимости и получения всех преимуществ новейших возможностей *framework*. Затем нажимаем кнопку «Создать».

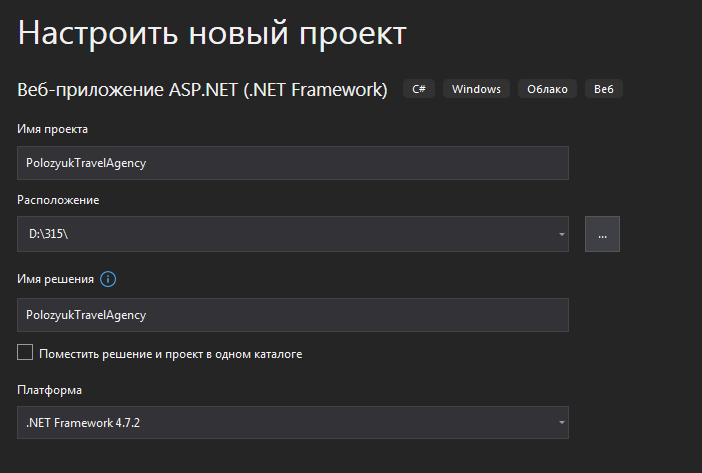


Рисунок 3. Настройка проекта

4. В открывшемся окне справа можно увидеть обозреватель решений, как видно ниже на рисунке 4.

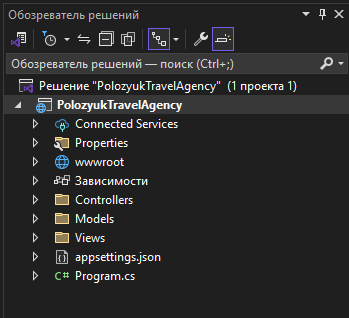


Рисунок 4 – Обозреватель решений

1. При его отсутствии нажмите *Ctrl+Alt+L* либо же перейдите в меню Вид => Обозреватель решений, как показано ниже.



В проекте есть несколько ключевых папок, с которыми мы будем работать:

* Папка *wwwroot* содержит файлы стилей, скрипты для динамического взаимодействия и другие вспомогательные файлы, которые могут облегчить разработку, если знать, как их использовать. В этом проекте мы будем применять чистый *JavaScript*.
* Папка *Controllers* отвечает за обработку входящих запросов. В рамках *framework* это включает в себя определение конкретных *URL*, по которым пользователи попадают при переходе по ссылкам или нажимая кнопки.
* Папка *Models* содержит данные, которые хранятся и обрабатываются на сервере.
* Папка *Views* представляет собой HTML-шаблон, который сервер возвращает после обработки запроса.

В обозревателе решений нам необходимо выбрать файл *Views/Shered/\_Layout.cshtml*. Файлы макетов или мастер-страницы играют ключевую роль в создании единого шаблона для страниц Razor, позволяя разработчикам обеспечивать согласованный и унифицированный вид всего приложения. Используя эти файлы, мы можем применять как *Razor*-код, так и *HTML*, подобно обычным страницам *Razor*, при этом упрощая процесс разработки.

С помощью мастер-страницы можно задавать общие элементы интерфейса, такие как меню и другие компоненты, которые будут присутствовать на всех страницах. Это значительно упрощает задачу подключения общих стилей и скриптов, поскольку нам не нужно будет заново прописывать пути к ним на каждой отдельной странице. Вместо этого все необходимые подключения можно сделать в одном месте, что не только экономит время, но и облегчает возможные изменения в будущем.

Кроме того, следует обратить внимание на тег <*head*>, который предназначен для хранения служебной информации о странице. Этот тег располагается в самом начале документа, сразу перед открывающим тегом <*body*>. Внутри тега <*head*> обычно содержатся важные элементы, такие как заголовок страницы, ключевые слова, описание и другие метаданные, которые помогают браузерам и поисковым системам правильно интерпретировать содержимое страницы.

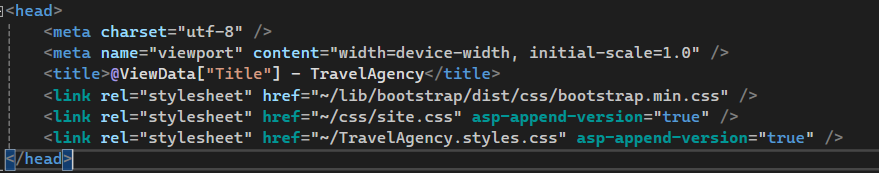


Рисунок 5 – тег <*head*>

Элемент <*header*> является семантическим компонентом, который выполняет важную функцию в структуре *HTML*-документа. Он служит для отделения вводного содержимого или навигационных элементов от основного контента страницы или отдельного секционного блока.

Например, в веб-приложении <*header*> может содержать "шапку" сайта, где располагаются ссылки на основные разделы навигации, логотип и название компании. Также он может включать информацию об авторе, дату публикации и заголовок в превью статей или новостей.

Использование <*header*> не только улучшает читаемость кода, но и помогает поисковым системам и вспомогательным технологиям лучше понимать структуру страницы. Это, в свою очередь, способствует более удобной навигации для пользователей и улучшает общую доступность сайта. Кроме того, наличие хорошо организованного заголовка может положительно сказаться на пользовательском опыте, делая интерфейс более интуитивным и легким для восприятия.

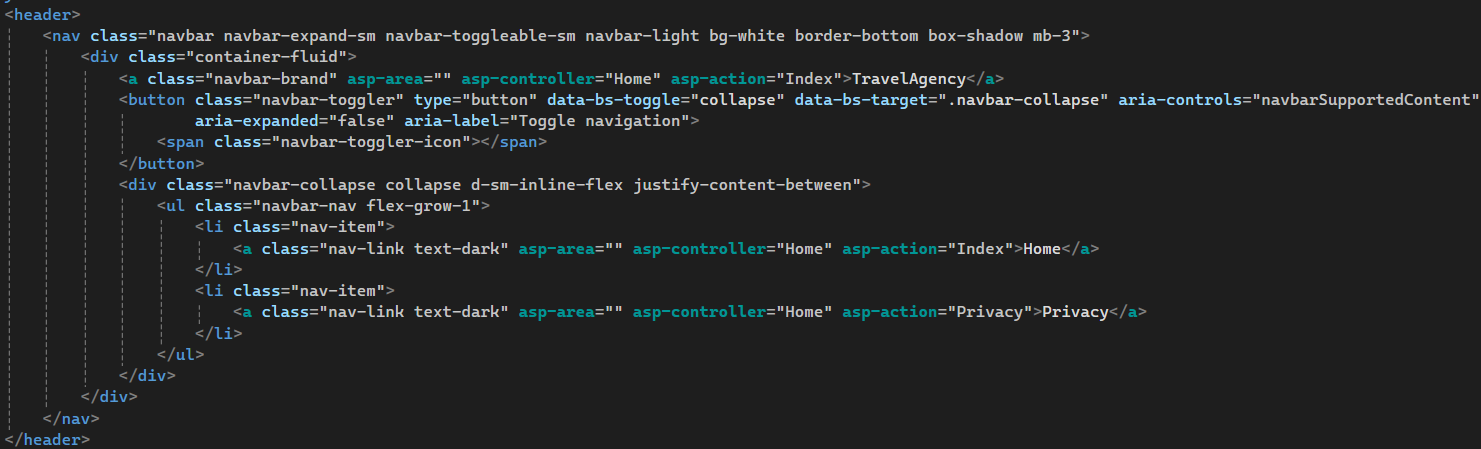


Рисунок 6 – тек <*header*>

Одной из ключевых особенностей файлов макетов является использование метода *@RenderBody*. Этот метод позволяет вставлять содержимое страниц *Razor*, которые используют данную мастер-страницу. Благодаря этому мы можем легко и удобно установить единообразный стиль оформления для всех страниц приложения.

Когда мы определяем структуру в файле макета, метод *@RenderBody()* служит как своеобразное "место" для вставки уникального контента каждой отдельной страницы. Это упрощает процесс разработки и поддержания приложения, поскольку изменения в оформлении или структуре можно внести в одном месте — в файле макета. Таким образом, мы достигаем консистентности в дизайне и минимизируем дублирование кода, что делает приложение более управляемым и легким в поддержке.

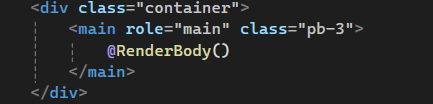


Рисунок 7 – функция *@RenderBody()*

Тег <*footer*> предназначен для создания футера сайта, также известного как подвал. Эта область располагается в нижней части страницы и часто содержит важную информацию, такую как контактные данные, нижнее меню, ссылки на политику конфиденциальности или условия использования, а также авторские права.

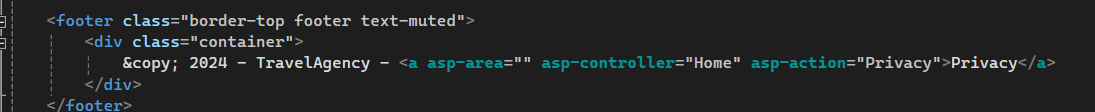


Рисунок 8 - тег <*footer*>

Скрипты обычно подключаются из внешних файлов с расширением .js. Для этого используется тег <*script*> с атрибутом *src*, в котором указывается путь к соответствующему файлу. Рекомендуется подключать внешние скрипты перед закрывающим тегом </*body*>

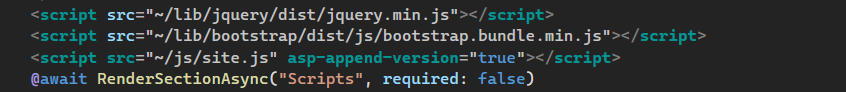


Рисунок 9 - подключение скриптов

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

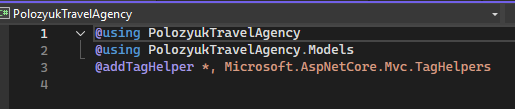


Рисунок 10 - содержимое файла *\_ViewImports*

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.

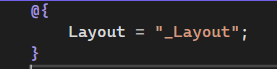


Рисунок 11 – содержимое файла *\_ViewStart.cshtml*

В разделе представлений имеется ещё одна папка — **Home**. В этой папке хранятся представления, связанные с действиями *(Action Methods)* контроллера ***HomeController***. Этот контроллер обычно отвечает за основные страницы веб-приложения, такие как главная страница и страница контактов. При запуске проекта по умолчанию вызывается именно этот контроллер, а действие ***Index***. Изменить этот параметр можно в файле ***Program.cs***, как показано на рисунке 12.

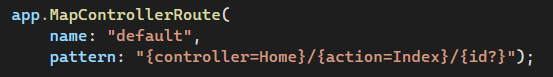


Рисунок 12 – фрагмент кода из файла *Program.cs*

При нажатии на кнопку "Пуск" в *Visual Studio* откроется стандартный веб-браузер, показывающий домашнюю страницу веб-приложения туристического агентства. Обычно *URL* будет выглядеть как *http://localhost:5000*, если в настройках проекта не указано иное. При первом запуске, если вы ничего не меняли, откроется главная страница, где отображается приветственный текст или название проекта с указанием, что приложение *ASP.NET Core* работает. На странице также будет представлена панель навигации — обычно это верхнее меню, откуда можно перейти на другие страницы. В стандартном шаблоне добавлены ссылки на главную страницу (*Home*) и страницу конфиденциальности (*Privacy*).

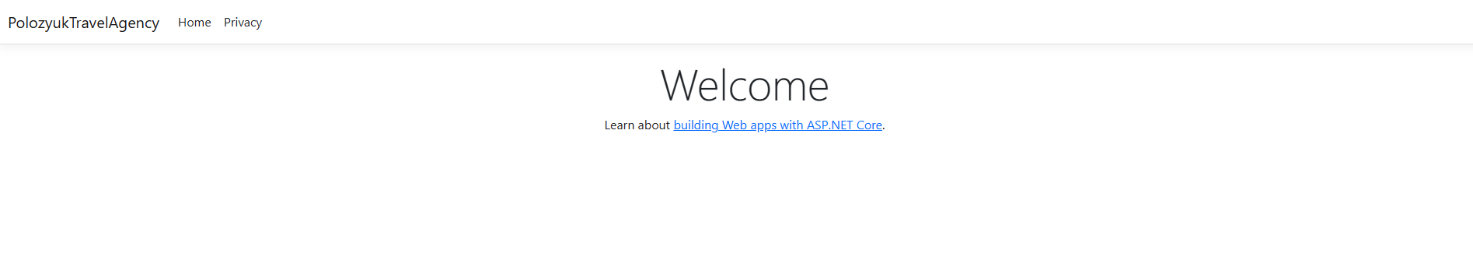


Рисунок 13 – Веб-приложение проекта

В процессе работы описаны ключевые этапы разработки, включая организацию структуры проекта, создание файлов макетов и использование семантических тегов для формирования удобного интерфейса.

**Вывод:** В данном отчёте была достигнута цель создания веб-приложения туристического агентства с использованием архитектуры *MVC*.

# 02.11.2024

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

**Ход работы:**

*Git* — это система контроля версий, предназначенная для отслеживания изменений в файлах проекта, командной работы над ними, а также для сохранения истории изменений с возможностью возврата к предыдущим версиям. В данной практике используется приложение *GitHub Desktop* для локального управления изменениями, для работы с которым необходимо иметь учётную запись на *GitHub* с подтверждённым адресом электронной почты.

Каждый пользователь *GitHub.com* идентифицируется через личную учётную запись, содержащую имя пользователя и профиль. Для создания новой учётной записи требуется перейти на [*https://github.com/*](https://github.com/), выбрать пункт регистрации и следовать инструкциям. Подтверждение адреса электронной почты является обязательным, так как без него невозможно выполнить некоторые ключевые действия на *GitHub*, такие как создание репозитория.

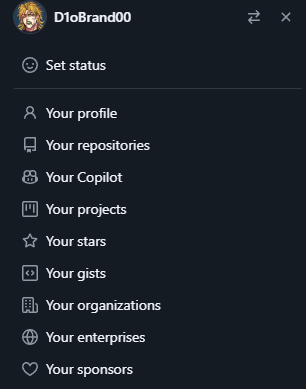


Рисунок 14 – профиль Github

Для создания репозитория сначала необходимо открыть меню пользователя в правом верхнем углу экрана и выбрать пункт *«Your repositories»*. На странице с существующими репозиториями нажимается кнопка *«New»*, после чего открывается страница для создания нового репозитория. В поле для названия указывается имя репозитория — *TravelAgency*, как показано на рисунке 15. Остальные поля и параметры можно оставить по умолчанию.

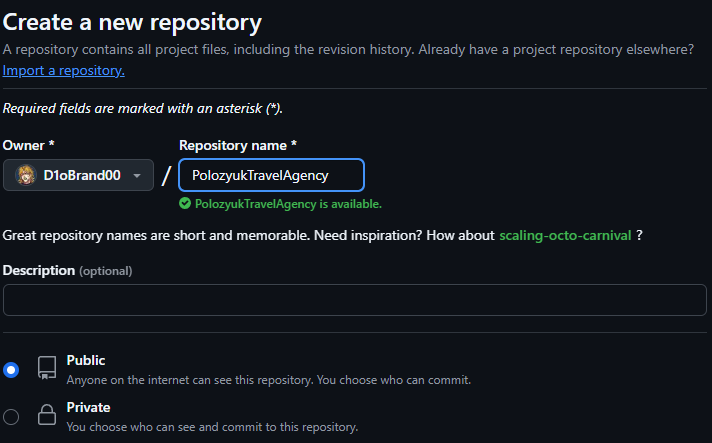


Рисунок 15 – создание удалённого репозитория

После создания удалённого репозитория необходимо перейти к созданию локального репозитория. Чтобы изучить основы работы с *GitHub Desktop*, можно посетить сайт *https://htmlacademy.ru/blog/git/github-desktop* или ознакомиться с информацией ниже, основанной на материале данного ресурса.

При первом входе в *GitHub Desktop* потребуется ввести логин и пароль от *GitHub.com*. После авторизации станет доступ ко всем репозиториям, сохранённым в профиле.

Для создания репозитория: если *GitHub* ранее не использовался, потребуется создать репозиторий для работы над проектом. На главном экране *GitHub Desktop* следует выбрать пункт *«Create a New Repository on your local drive».*

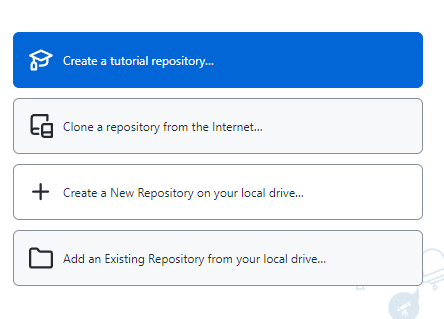


Рисунок 16 – начало работы

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

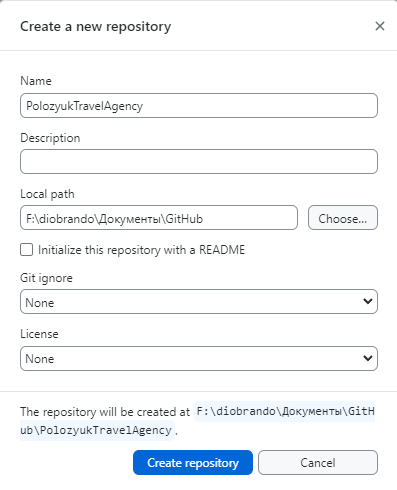


Рисунок 17 – создание локального репозитория

Далее нажимается кнопка *Create repository*, и через несколько секунд на компьютере создаётся папка, готовая для разработки проекта.

Клонирование репозитория: если уже существует репозиторий на *GitHub*, его можно клонировать. Клонирование позволяет скачать все файлы на компьютер, чтобы в дальнейшем их редактировать и загружать обратно. Для этого выбирается пункт Add -> Clone Repository....

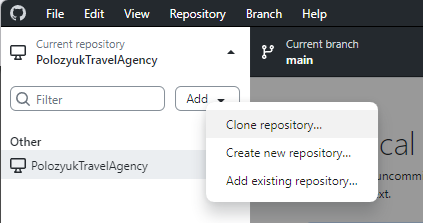


Рисунок 18 – клонирование репозитория

В открывшемся окне следует выбрать один из существующих репозиториев. В данном примере он называется PolozyukTravelAgency, однако название может отличаться в зависимости от доступных репозиториев.

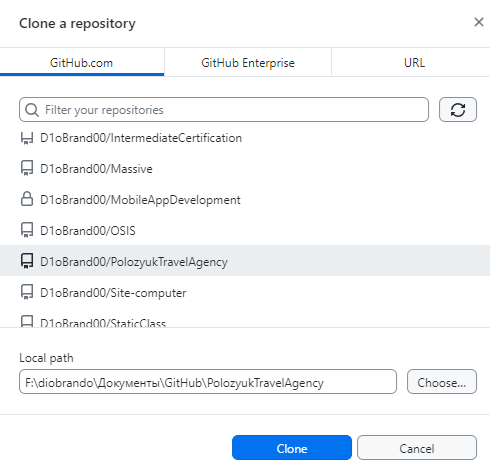


Рисунок 19 - выбор репозитория

После этого начнётся загрузка файлов репозитория на компьютер. Если файлов много, процесс может занять некоторое время.

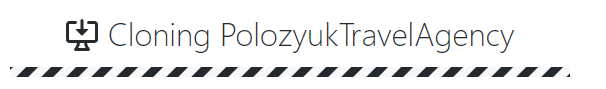


Рисунок 20 – скачивание файлов

Работа с репозиторием. Изменение файлов и сохранение обратно: независимо от того, был ли репозиторий создан или клонирован, интерфейс *GitHub Desktop* будет выглядеть следующим образом с открытым репозиторием, в котором пока не было внесено никаких изменений.

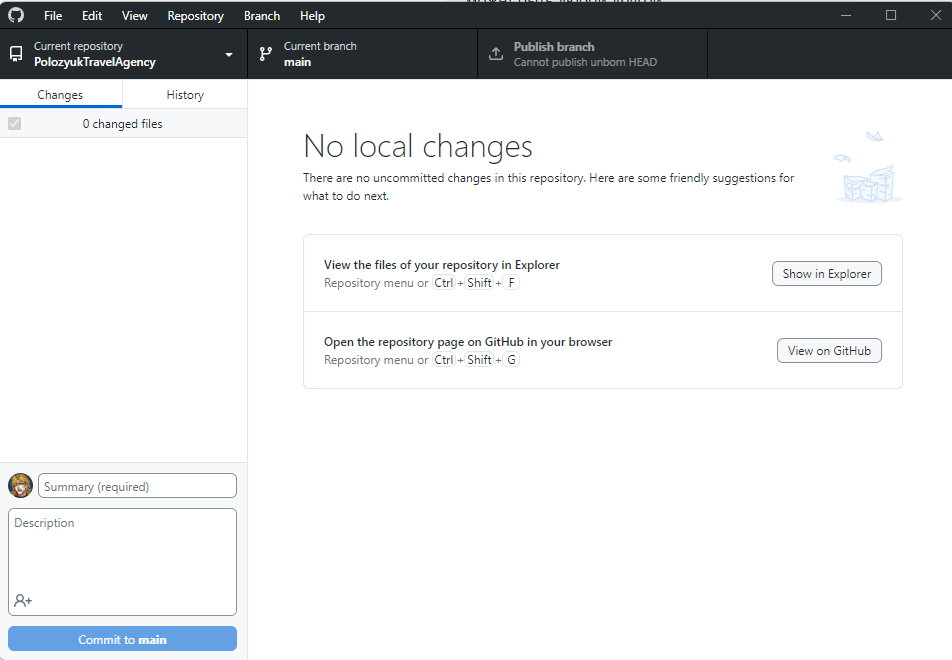


Рисунок 21 – вид приложения с открытым репозиторием

Слева располагается область для изменённых файлов, справа — служебная информация. В нижней части слева находится поле для коммитов.

Для примера можно добавить новый файл. Например, в локальный репозиторий был добавлен файл index.html. Также возможно добавить текстовый файл.

Сразу после добавления или изменения файла в окне *GitHub Desktop* отобразится информация о внесённых изменениях. Если был добавлен новый файл, все строки будут отмечены плюсиками и окрашены в зелёный цвет. Это указывает на то, что данные строки были добавлены в файл, и *GitHub Desktop* ранее их не фиксировал.

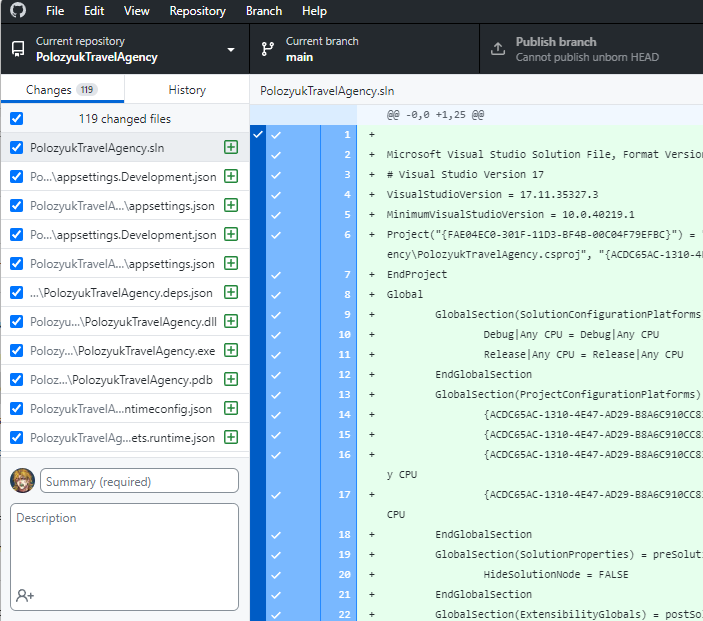


Рисунок 22 – вид при добавлении файла

Загрузка нового репозитория на *GitHub*: после добавления кода в только что созданный репозиторий необходимо выполнить коммит, то есть зафиксировать все сохранённые изменения и дать им название. Текст коммита должен быть лаконичным, но информативным, чтобы отражать суть внесённых изменений. Например, можно использовать формулировки «добавляет имя наставника в *Readme*», «вводит функцию сортировки изображений» или «исправляет ошибку в поиске городов на карте». После ввода текста коммита следует нажать на большую синюю кнопку «*Commit to main*».

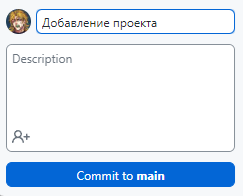


Рисунок 23 – создание коммита

Внесённые изменения и сохранённые данные остаются локальными. Чтобы отправить их на GitHub, необходимо опубликовать свежесозданный репозиторий. Для этого следует нажать кнопку Publish repository.

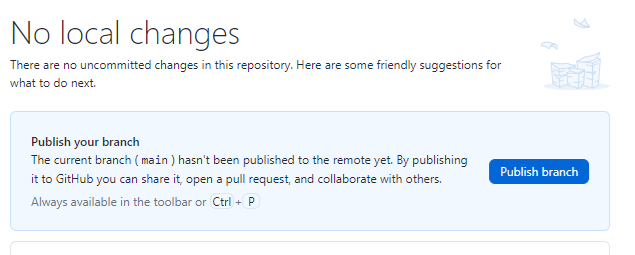


Рисунок 24 – опубликовать репозиторий

Появится окно подтверждения публикации репозитория. Следует проверить название и описание, при необходимости установить галочку, чтобы сделать код приватным, и затем нажать кнопку для публикации.

Готово — после этого репозиторий будет доступен в профиле на *GitHub.com*.

Добавление кода и коммит изменений: репозиторий успешно создан и загружен на *GitHub*, теперь необходимо внести изменения в код. После редактирования файлов, находящихся в репозитории, все изменения можно будет просмотреть в окне *GitHub Desktop*.

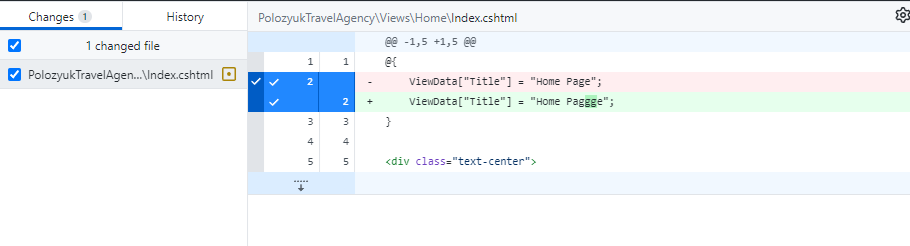


Рисунок 25 – просмотр исправлений

Дальше действуем по проверенной схеме – коммитим изменения.

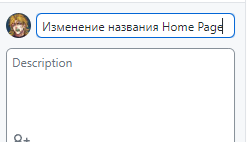


Рисунок 26 – коммит изменений

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

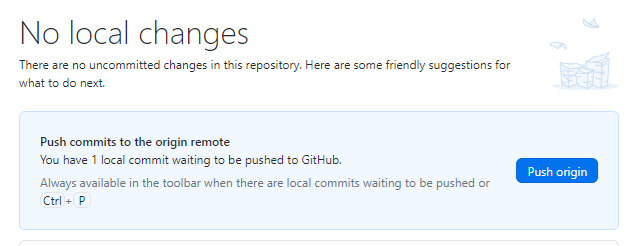


Рисунок 27 – пуш изменений

Готово! Теперь, если зайти на GitHub.com в созданный репозиторий, можно увидеть изменённый файл, который был только что загружен.

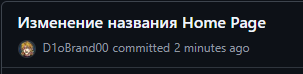


Рисунок 28 – проверка отправки изменений

Все успешно выполнено — теперь есть возможность создать или клонировать репозиторий, добавить в него файлы и опубликовать всё на GitHub.com, не используя консоль.

**Вывод:** В ходе практики было успешно изучено основное устройство и принципы работы с платформой *GitHub*. Освоены ключевые процессы, такие как создание репозитория, клонирование существующих репозиториев, выполнение коммитов для фиксации изменений и публикация репозитория на *GitHub*.

# 04.11.24

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

**Ход работы:**

В данном проекте будет использоваться чистая архитектура, показанная на рисунке 29.

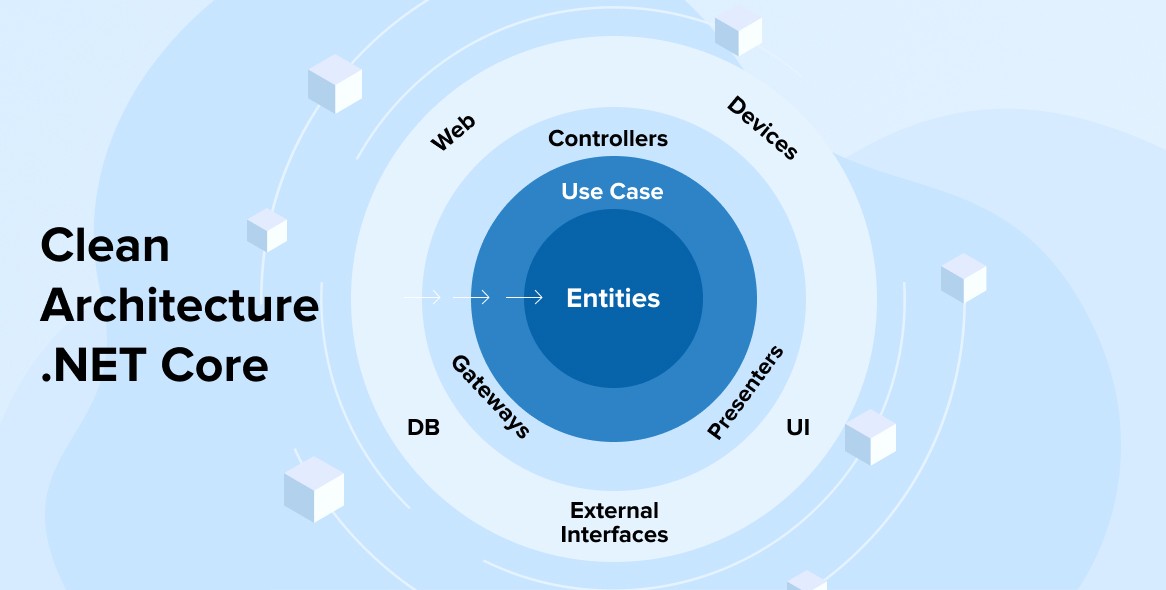


Рисунок 29 – чистая архитектура.

Правило зависимости.

Архитектура состоит из концентрических кругов, где внешний круг представляет детали реализации, а внутренний — абстрактные бизнес-правила. Основное правило — зависимости должны направляться только внутрь, то есть внутренние слои не могут зависеть от внешних. Внешние данные не должны использоваться в внутренних слоях.

Сущности.

Сущности определяются бизнес-правилами и инкапсулируют основные и стабильные правила системы. Они представляют собой бизнес-объекты, которые не зависят от изменений в внешних компонентах, таких как интерфейс или навигация.

Сценарии.

Сценарии реализуют бизнес-правила и управляют потоком данных между слоями. Они инкапсулируют все варианты использования системы. Изменения в логике приложения могут повлиять на этот слой, но изменения во внешних компонентах, как базы данных или *UI*, не должны затрагивать сценарии.

Интерфейс-Адаптеры

Этот слой преобразует данные между внутренними слоями и внешним миром (например, для хранения в базе данных). В нем реализуется архитектура MVC, где модели передаются от контроллеров к сценариям и обратно. Код этого слоя не должен зависеть от конкретных технологий, таких как *SQL*-запросы.

Фреймворки и драйверы.

Внешний слой включает фреймворки, базы данных и пользовательский интерфейс. Он взаимодействует с внутренними слоями и служит для подключения приложения к реальному миру, минимизируя влияние на основную логику.

Пересечение границ.

Контроллеры и представления взаимодействуют с более внутренними слоями, такими как сценарии, через интерфейсы. Это позволяет соблюдать правило зависимости, избегая прямых зависимостей между слоями.

Как данные пересекают границы.

Данные, пересекающие границы между слоями, должны быть изолированы от специфики каждого слоя. Передача данных должна происходить в формате, удобном для внутреннего слоя, чтобы избежать зависимости от внешних технологий, например, ORM.Реализация проекта:

Шаг 1. Находим в Обозревателе решений папку Models и, щёлкнув правой кнопкой мыги, удаляем её, как видно на рисунке 30.

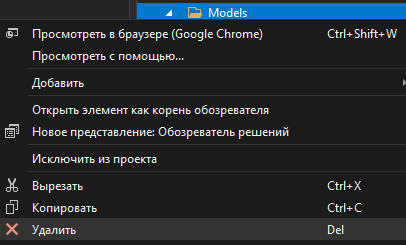


Рисунок 30 – удаление папки проекта

Шаг 2. В Обозревателе решений нажимаем правой кнопкой мыши по корневой папке проекта и выбираем Добавить -> Создать папку решения.

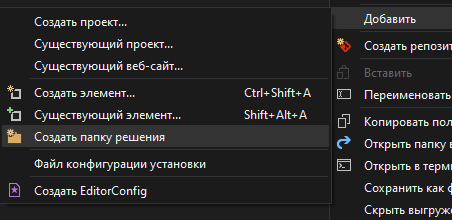


Рисунок 31 – создание папки решения

Таким образом создаём три новые папки с названиями: DAL, Domain, Service.

Шаг 3. Далее создадим в каждой папке проект «Библиотека классов», как показано на рисунке 32.

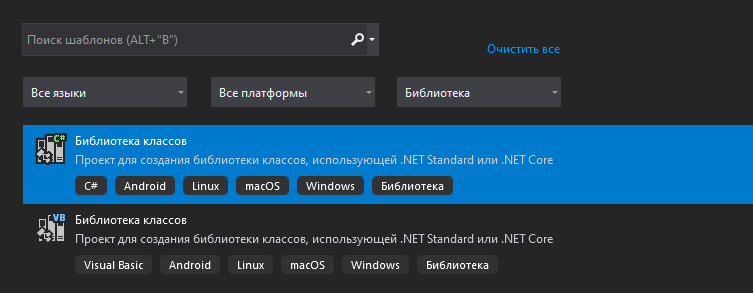


Рисунок 32 – создание проекта «Библиотека классов»

По итогу мы получаем трёхуровневовую архитектуру, что видно на рисунке 33.

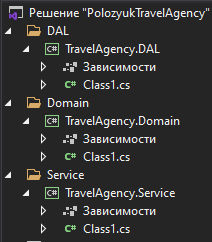


Рисунок 33 – трёхуровневая архитектура

**Вывод:** В ходе изучения теоретической части работы, были рассмотрены основные принципы, лежащие в основе архитектуры приложения, основанной на паттерне MVC, а также взаимодействие его ключевых компонентов: **моделей, контроллеров и представлений.**