Дипломная работа по теме:

Анализ и сравнение написания web-приложений с использованием разных фреймворков:Разработка простых веб-приложений с использованием Django, Flask и FastAPI, их сравнение.

Выполнил:

Шевелев А.Н.

2024 г.

|  |  |
| --- | --- |
| Введение | 3 |
| Обоснование выбора темы | 3 |
| Определение цели и задач исследования | 3 |
| Основные понятия и определения | 4 |
| Обзор инструментов для разработки веб-приложений | 5 |
| Django | 5 |
| Flask | 7 |
| FastAPI | 9 |
| Обзор проекта | 12 |
| Структура проекта | 12 |
| Главная страница | 12 |
| Страница расчета | 13 |
| Ссылки на сторонние сайты | 14 |
| Файловая структура проектов | 15 |
| Структура проекта Django | 15 |
| Структура проекта Flask | 16 |
| Структура проекта FastAPI | 16 |
| Шаблоны страниц | 17 |
| Анализ результатов | 17 |
| Django | 17 |
| Flask | 17 |
| FastAPI | 17 |
| Заключение | 18 |
| Источники | 19 |

ВВЕДЕНИЕ

**Обоснование выбора темы**

1. Актуальность: в настоящее время практически любой бизнес нуждается в своем веб-приложении. Фреймворки позволяют предоставить конечному пользователю готовый к использованию продукт.

2. Потребности рынка: существует потребность в специалистах, способных работать с помощью популярных фреймворков, а также обладающих навыками написания HTML и CSS кода.

3. Личный интерес: создание веб-приложений всегда было интересно мне самому, в связи с этим я планирую развивать навык работы с фреймворками, создавать более сложные и масштабные проекты, а также, надеюсь, что это поможет мне в карьерном росте.

Таким образом, выбор темы «Анализ и сравнение написания web-приложений с использованием разных фреймворков» обусловлен её актуальностью, личным интересом и потребностью рынка, что делает ее подходящей для дипломной работы.

**Определение цели и задач исследования**

Цель исследования: создать веб-приложения на трёх фреймворках, и провести их сравнение.

Задачи исследования:

1. Обзор фреймворков Django, Flask и FastAPI: провести анализ, выделить плюсы и минусы.

3. Создание веб-приложений: создать идентичные веб-приложения, и оценить их работу.

4. Оценить сложность кода: сравнить количество и сложность кода в зависимости от фреймворка.

4. Написание дипломной работы: составить дипломную работу, включающую в себя введение, обзор литературы, анализ результатов и вывод.

Цели и задачи исследования направлены на получение практических результатов, которые позволяют сравнить и оценить фреймворки Django, Flask и FastAPI.

**Основные понятия и определения**

Основные понятия и определения в области написания web-приложений с использованием фреймворков

1. **Фреймворк**(англ. framework — «каркас, структура») — **э**то динамически пополняемая библиотека языка программирования, в которой собраны его базовые модули. Фреймворки упрощают процесс разработки приложений, сайтов, сервисов.

2. **Веб-приложение** (Web Application): Программное приложение, которое работает на веб-сервере и доступно через браузер. В контексте данного проекта, веб-приложение будет использоваться для сравнения и демонстрации различных фреймворков.

3. **Веб-сервер** — это программное и аппаратное обеспечение, которое использует протокол HTTP (Hypertext Transfer Protocol) и другие протоколы для ответа на запросы клиентов, сделанные через браузер

3. **Модель** (Model) — место, где хранятся данные приложения.

4. **Представление** (View) — определяет взаимодействие с пользователем через модель, отвечает за отображение данных, внешний вид проекта.

5. **Контроллер** (Controller) — отвечает за реакцию двух предыдущих объектов на действия пользователя.

**Обзор инструментов для разработки веб-приложений на Python, используемых в дипломной работе**

Используемые фреймворки – Django, Flask, FastAPI

Django — это Python-фреймворк для разработки веб-приложений. Он предоставляет разработчикам огромный выбор готовых модулей, надстроек и инструментов, которые значительно ускоряют и упрощают процесс создания сложных, многофункциональных веб-приложений.

Из чего состоит Django

Этот фреймворк предлагает разработчику готовый набор инструментов для создания сайтов. Он использует принцип MVT или Model-View-Template, который дает возможность разделить логику работы сайта, его внешнее отображение для пользователей и реакции.

Как это работает:

1. Шаблоны — они определяют, каким будет внешний вид страницы сайта. Это файлы с HTML-кодом, отображающим содержимое. Они могут быть как статическими, так и динамическими;
2. Представление — отвечает за данные, а также принимает, обрабатывает и реагирует на HTTP-запросы. За счет него приложение отвечает на действие пользователей;
3. Модель — это источник информации о данных. Она обеспечивает внутренние функции, например, подключение к базе данных или форматирование информации. Ее задача — отвечать за бизнес-логику, свойства и другие манипуляции.

В состав фреймворка входит много модулей, позволяющих реализовывать функционал сайта. В их числе:

* Модули защиты от всех популярных сетевых атак;
* Веб-сервер, позволяющий проводить разработку и тестирование;
* Интерфейс администратора, через который можно вносить изменения во внутреннее содержимое сайта;
* Встроенные модули авторизации и аутентификации;
* Диспетчер, который обменивается событиями между компонентами;
* Фильтры обработки запросов — перенаправления, кэширования и так далее;
* Модули для подключения внешних блоков к сайту.

**Плюсы Django:**

1. **Быстрота:** Django был разработан, чтобы помочь разработчикам создать приложение настолько быстро, на сколько это возможно. Это включает в себя формирование идеи, разработку и выпуск проекта, где Django экономит время и ресурсы на каждом из этих этапов. Таким образом, его можно назвать идеальным решением для разработчиков, для которых вопрос дедлайна стоит в приоритете.
2. **Полная комплектация:** Django работает с десятками дополнительных функций, которые заметно помогают с аутентификацией пользователя, картами сайта, администрированием содержимого, RSS и многим другим. Данные аспекты помогают осуществить каждый этап веб разработки.
3. **Безопасность:** Работая в Django, вы получаете защиту от ошибок, связанных с безопасностью и ставящих под угрозу проект. Я имею ввиду такие распространенные ошибки, как инъекции SQL, кросс-сайт подлоги, clickjacking и кросс-сайтовый скриптинг. Для эффективного использования логинов и паролей, система пользовательской аутентификации является ключом.
4. **Масштабируемость:** фреймворк Django наилучшим образом подходит для работы с самыми высокими трафиками. Следовательно, логично, что великое множество загруженных сайтов используют Django для удовлетворения требований, связанных с трафиком.
5. **Разносторонность:** менеджмент контента, научные вычислительные платформы, даже крупные организации – со всем этим можно эффективно справляться при помощи Django.

**Минусы Django:**

1. Некоторая ограниченность: есть вещи, которые в Django заданы строго, и изменить их нельзя. Если нет желания писать их с нуля, придется смириться с их ограничениями.
2. Требовательность к ресурсам: из-за обилия встроенных инструментов Django требует больше мощностей для запуска, чем нативный код на пайтон.
3. Слишком большая сложность для простых проектов: маленький одностраничный сайт может быть быстрее собрать с нуля, чем писать полную структуру проекта и программировать на Django.

Из всего вышеперечисленного я делаю вывод, что Django подходит более опытным программистам и для больших проектов.

С помощью фреймворка Django реализовано очень много популярных сервисов и проектов, например Youtube, страница поиска Google, Mozilla, Reddit, Pinterest, Dropbox, сайты онлайн-магазинов вроде «Спортмастера».

Flask — микрофреймворк для Python. Приставка «микро» означает, что инструмент предоставляет минимальный набор функций для создания веб-приложений. Чтобы получить доступ к дополнительным возможностям, нужно подключить сторонние плагины и библиотеки.

Фреймворк позволяет создавать веб-приложения по шаблонам с гибкой настройкой функциональности. Его легко использовать, а ограниченный встроенный инструментарий можно быстро расширить: например, добавить модули аутентификации, обработки форм и работы с базами данных.

Плюсы Flask:

1. Простота: Flask легко установить и настроить.
2. Гибкость: микрофреймворк позволяет разработчикам самостоятельно выбирать технологии и инструменты, которые они хотят применять в своих проектах
3. Расширяемость: Flask позволяет расширять функциональность с помощью плагинов и модулей, которые можно легко интегрировать в проект.
4. Активное сообщество: Flask является одним из самых используемых фреймворков для Python, поэтому имеет большое комьюнити разработчиков.

Минусы Flask:

1. Отсутствие готовых решений: разработчики изначально могут использовать во Flask только минимальный набор функциональности. Если же программисту нужны более широкие возможности, такие как аутентификация пользователя, ему придется добавить дополнительные библиотеки или реализовать это самостоятельно.
2. Нет встроенной многопоточности: Flask был разработан как однопоточный фреймворк. И чтобы управлять многопоточными веб-приложениями, придется установить дополнительные библиотеки.
3. Ограниченные возможности для масштабирования: если проект начинает расти и усложняться, то могут появиться сложности в поддержке нужной функциональности.

Исходя из вышеперечисленного, можно сделать вывод, что Flask больше подходит для новичков и для написания небольших проектов.

Сайты, которые используют Flask:

* Социальная сеть Pinterest – pinterest.com
* Онлайн-журнал Creapassions – creapassions.com
* Образовательные онлайн-курсы Brilliant – brilliant.org
* Онлайн-магазин одежды и обуви Zalando – zalando.de
* Компания по разработке программного обеспечения IBM – ibm.com

FastAPI — это фреймворк для создания лаконичных и довольно быстрых HTTP API-серверов со встроенными валидацией, сериализацией и асинхронностью. Стоит он на плечах двух других фреймворков. Работой с web в FastAPI занимается Starlette, за валидацию отвечает Pydantic.

Стоит сразу отметить, что FastAPI работает асинхронно. Асинхронное программирование — это потоковая обработка программного обеспечения /пользовательского пространства, где приложение, а не процессор, управляет потоками и переключением контекста. В асинхронном программировании контекст переключается только в заданных точках переключения, а не с периодичностью, определённой CPU.

Основные возможности FastAPI

* FastAPI предлагает разработчикам широкий спектр возможностей для создания высокопроизводительных веб-сервисов и API на Python.Декларативные роуты: FastAPI использует декларативный подход к определению роутов с использованием аннотаций типов Python. Это позволяет разработчикам четко указывать типы данных входных и выходных параметров, что упрощает валидацию данных и автоматическую генерацию документации.
* Автоматическая валидация данных: благодаря интеграции с библиотекой pydantic фреймворк автоматически выполняет валидацию данных, полученных от клиентов, в соответствии с описанными моделями данных. Это помогает избежать ошибок и обеспечивает безопасность при обработке запросов.
* Интерактивная документация: FastAPI генерирует интерактивную документацию API автоматически на основе описаний Python (OpenAPI и Swagger). Разработчики могут легко просматривать доступные эндпоинты, параметры запросов и примеры использования, что значительно упрощает тестирование и использование API.
* Поддержка асинхронности: FastAPI полностью поддерживает асинхронное программирование с использованием asyncio. Это позволяет обрабатывать множество одновременных запросов эффективно, не блокируя потоки, что особенно важно для высоконагруженных приложений.
* Встроенная поддержка CORS: фреймворк предоставляет встроенную поддержку Cross-Origin Resource Sharing (CORS), что позволяет ограничивать доступ к ресурсам с разных источников и обеспечивать безопасность веб-приложений.
* Модульные зависимости: FastAPI поддерживает инъекцию зависимостей, что позволяет разделять код на независимые модули и легко управлять зависимостями между ними.
* Поддержка WebSocket: в последних версиях FastAPI добавлена экспериментальная поддержка WebSocket, что позволяет создавать веб-сокеты и обрабатывать двусторонние связи в режиме реального времени.

Плюсы FastAPI:

1. Быстрота и производительность: FastAPI использует asyncio и pydantic для достижения высокой производительности обработки запросов.
2. Декларативный стиль: для определения входных и выходных данных используются аннотации типов Python, что упрощает их валидацию и автодокументирование.
3. Поддержка OpenAPI и Swagger: FastAPI автоматически генерирует интерактивную документацию API на основе описаний Python, что упрощает разработку и тестирование API.
4. Асинхронная поддержка: благодаря использованию asyncio FastAPI позволяет эффективно обрабатывать множество одновременных запросов без блокировки потоков.
5. Интеграция с Starlette: FastAPI построен на базе Starlette, что обеспечивает надежность и расширяемость фреймворка.

Минусы FastAPI:

1. Для новичков FastAPI может показаться сложным из-за использования асинхронного программирования, аннотаций типов Python и других современных подходов.
2. В некоторых случаях настройка окружения и зависимостей для работы с FastAPI может потребовать дополнительных усилий по сравнению с более традиционными фреймворками.
3. Некоторые возможности FastAPI, такие как поддержка WebSocket, все еще могут быть в стадии эксперимента и не обеспечивать полной стабильности и поддержки.
4. По сравнению с другими более устоявшимися фреймворками, экосистема FastAPI может быть менее развитой. Это может затруднить поиск готовых решений и плагинов для специфических задач.
5. Поскольку FastAPI все еще активно развивается, могут происходить изменения в API и подходах к решению задач, что может потребовать пересмотра и адаптации кода.

Из всего этого можно сделать вывод, что FastAPI хоть и подходит скорее для опытных программистов, но при этом можно создавать как небольшие, так и крупные проекты.

Некоторые компании, которые используют FastAPI: Uber, Netflix и Microsoft.

**Обзор проекта**

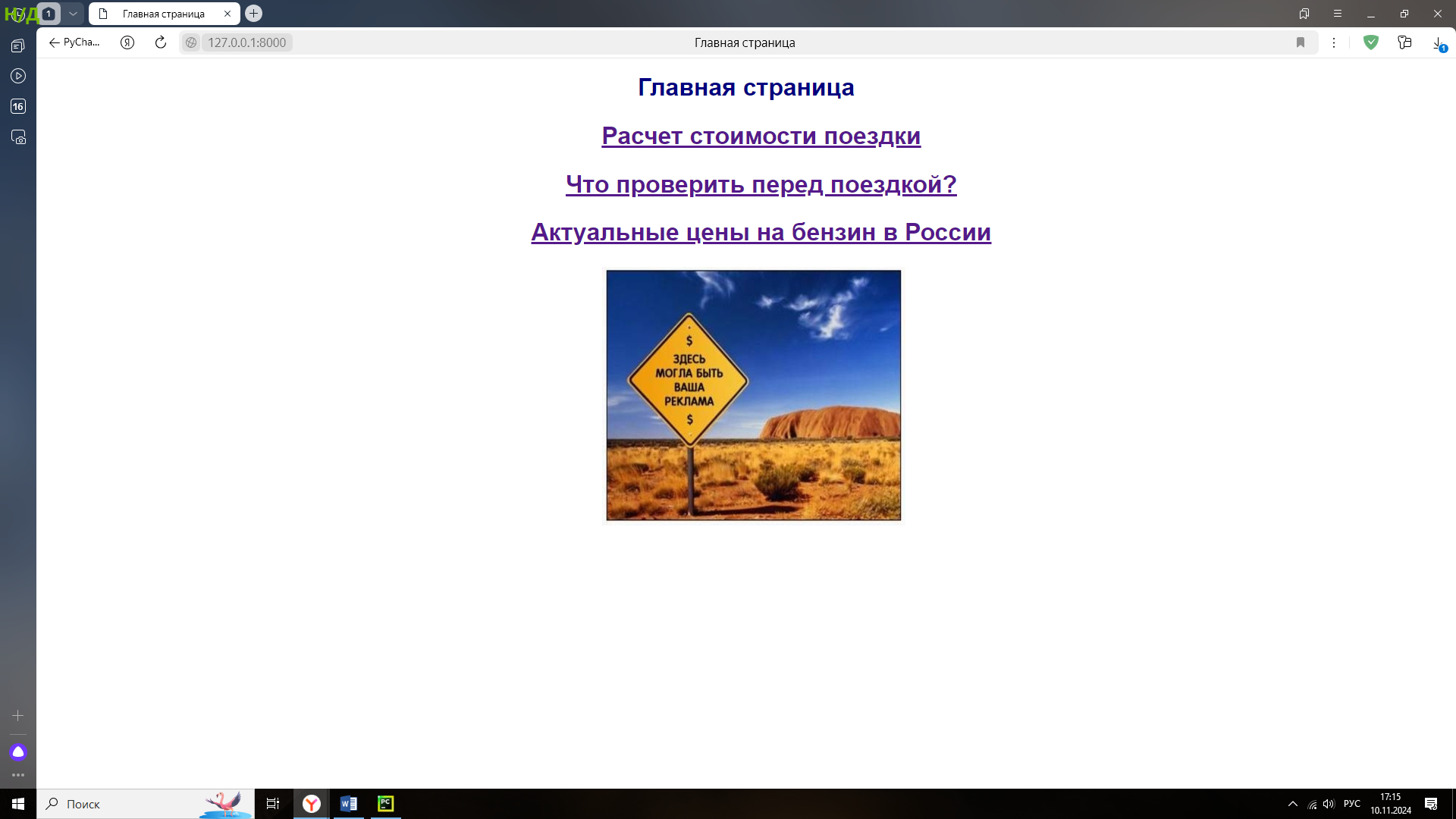
Моё веб-приложение предназначено для того, чтобы пользователь мог без знания формул посчитать сколько будет стоить поездка на автомобиле на определенное расстояние.

**Структура проекта:**

Проект включает в себя следующие ключевые компоненты:

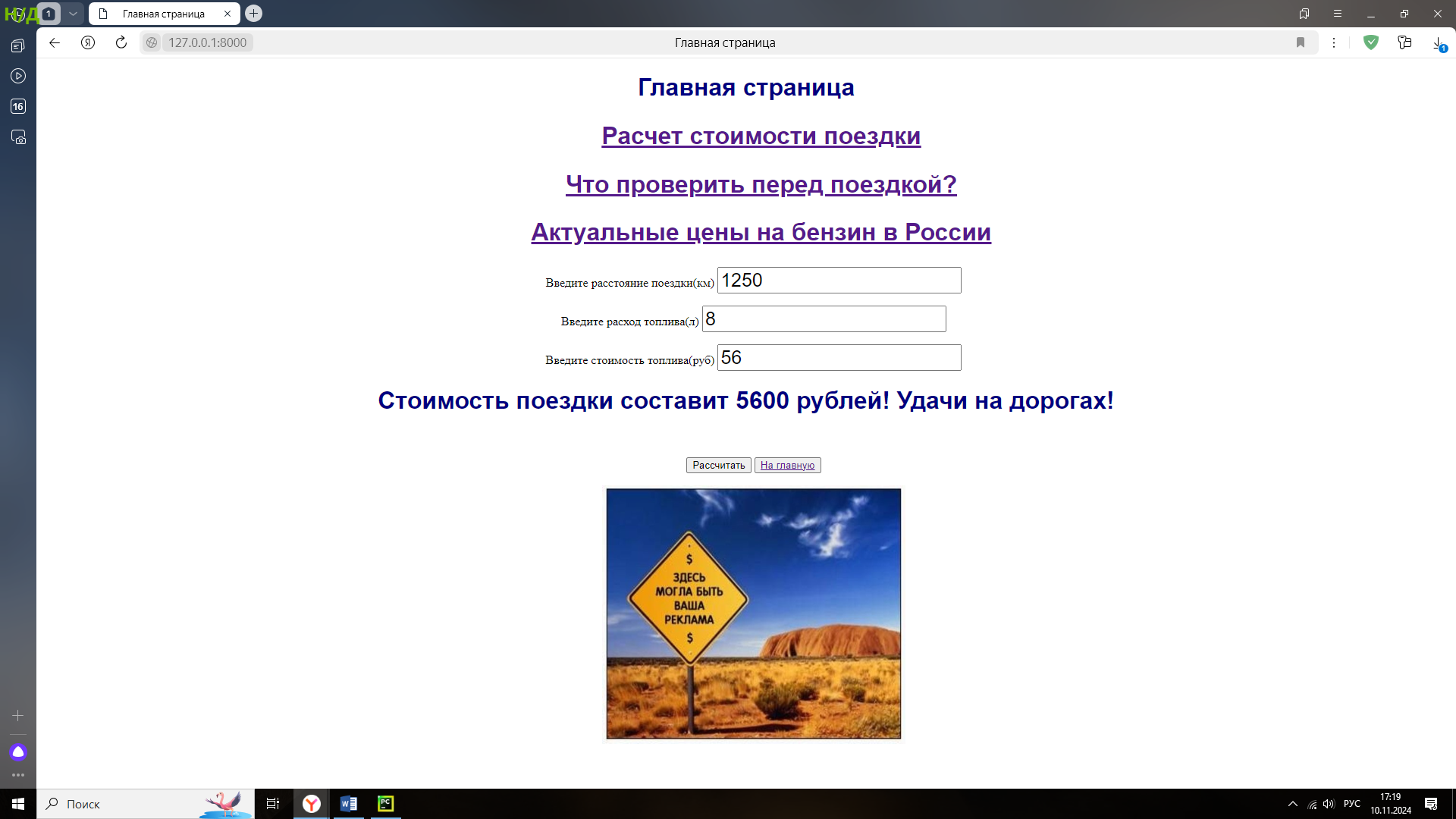
**Главная страница**

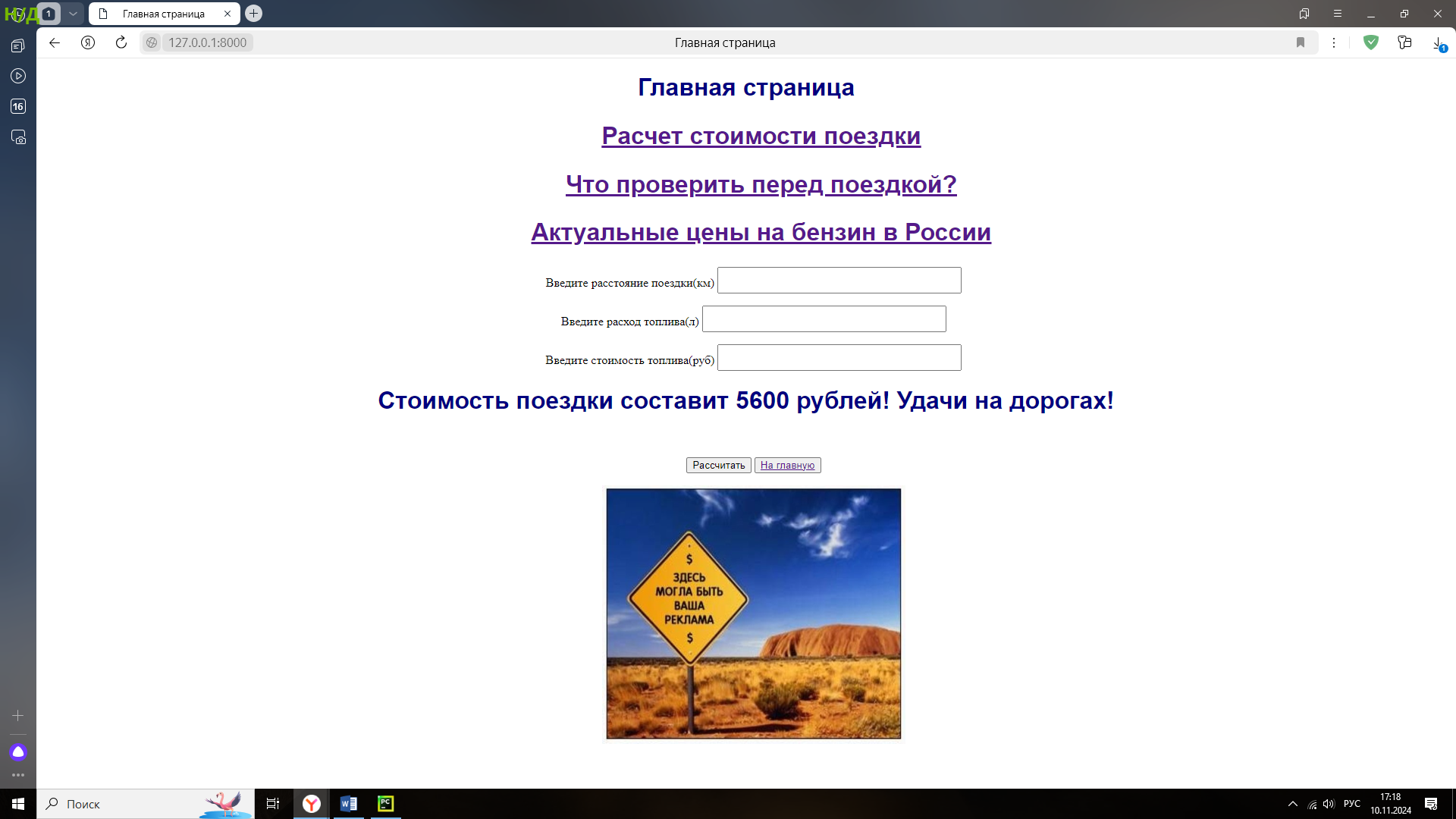
Здесь есть ссылка на страницу с формами ввода, актуальные цены на бензин, а также на статью о подготовке автомобиля к дальней поездке.



**Страница расчета**

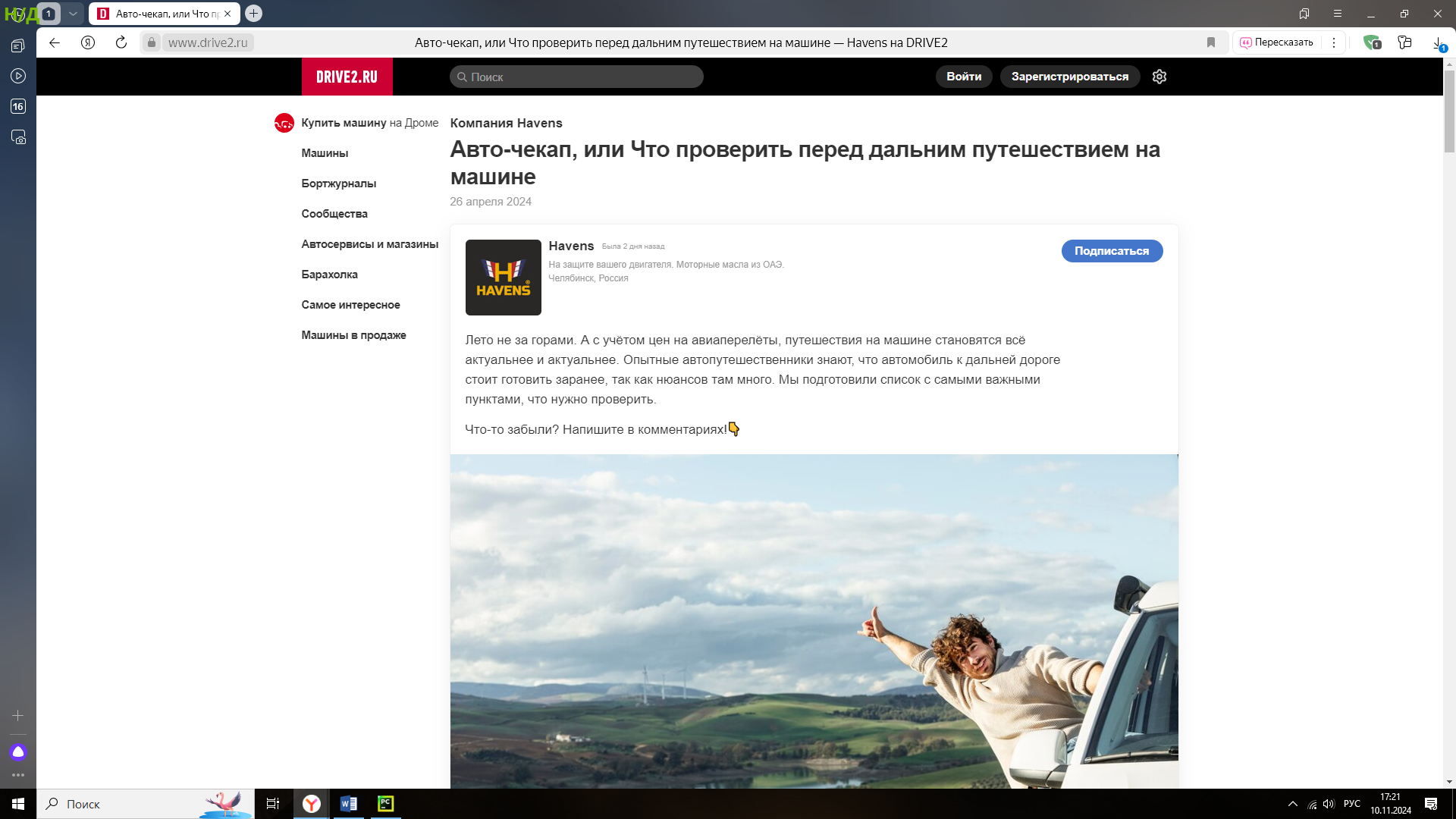
Страница с формами ввода, заполнив которые, и нажав кнопку «Рассчитать» пользователь получит стоимость своей поездки.

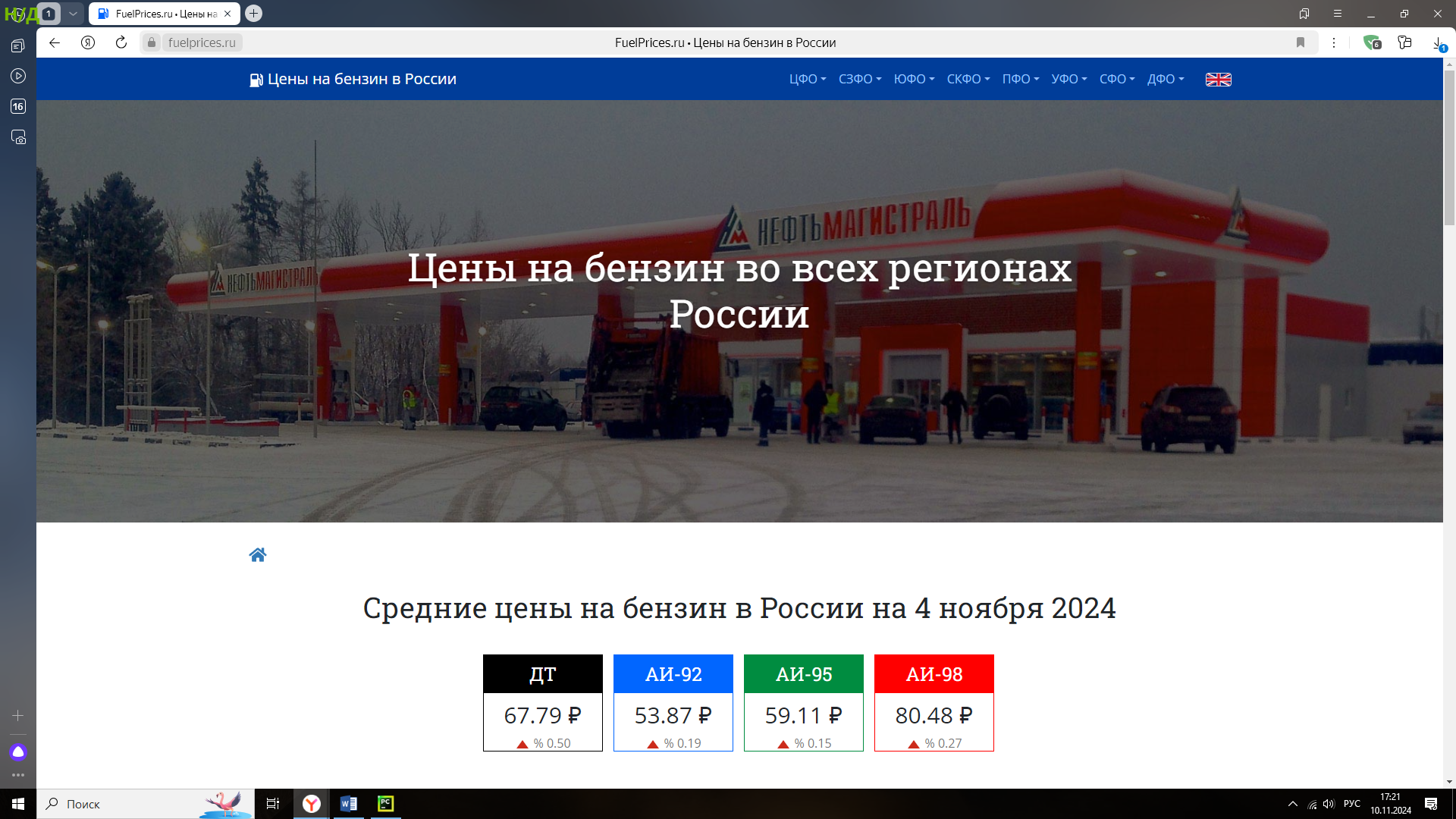




**«Что проверить перед поездкой» и «Актуальные цены на бензин»**

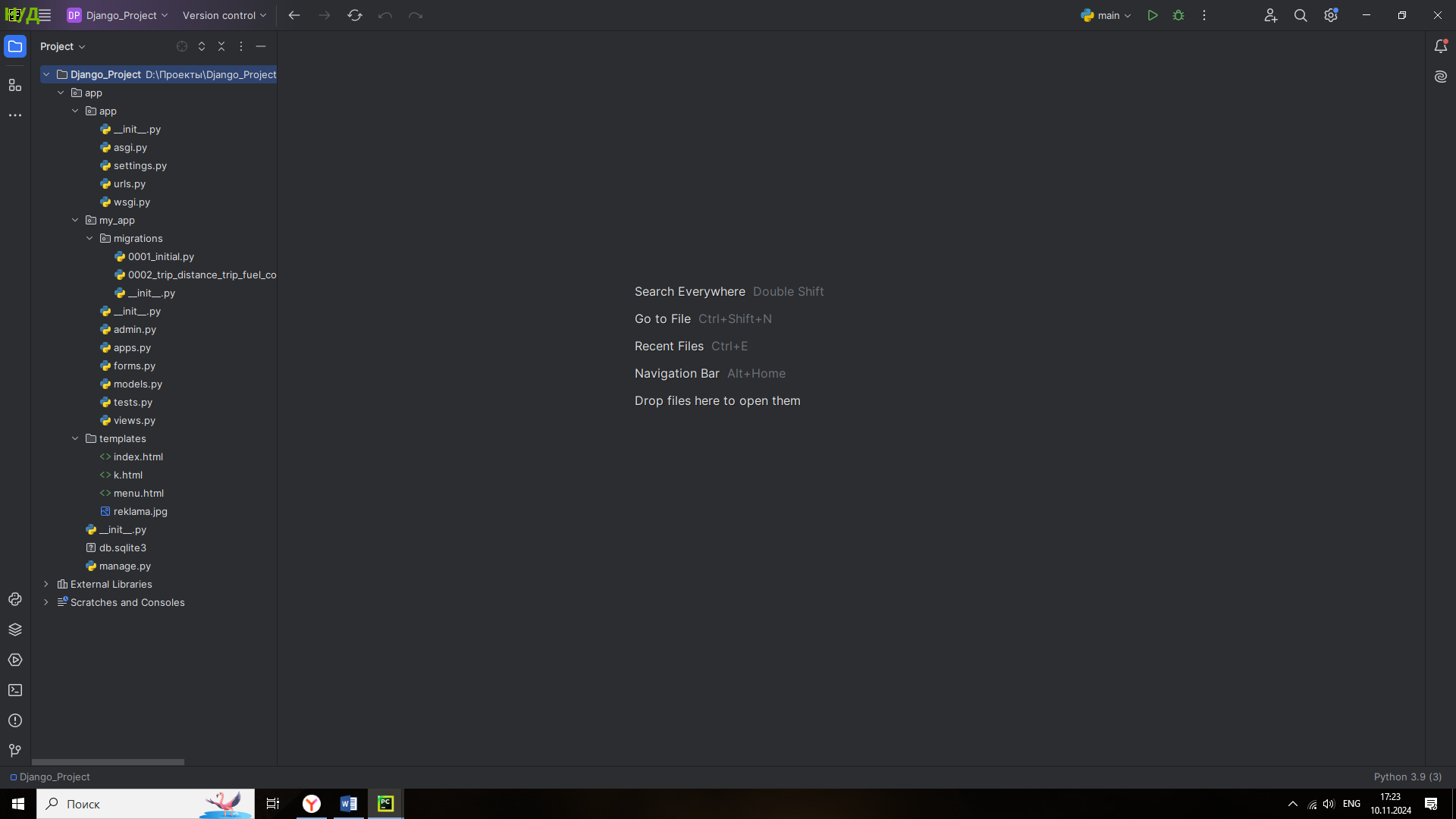
Ссылки, перенаправляющие на сторонние сайты.



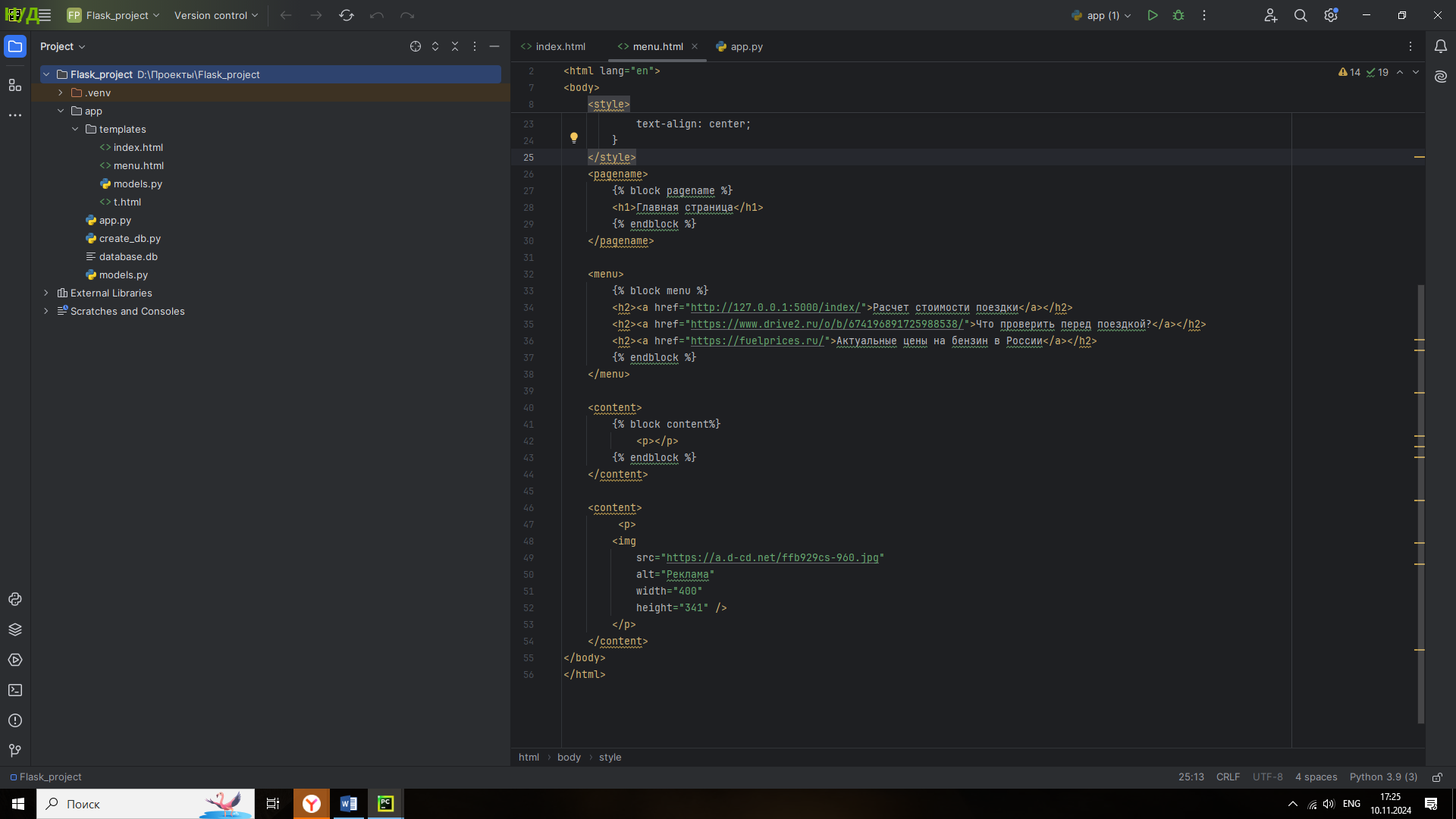


**Файловая структура проектов**

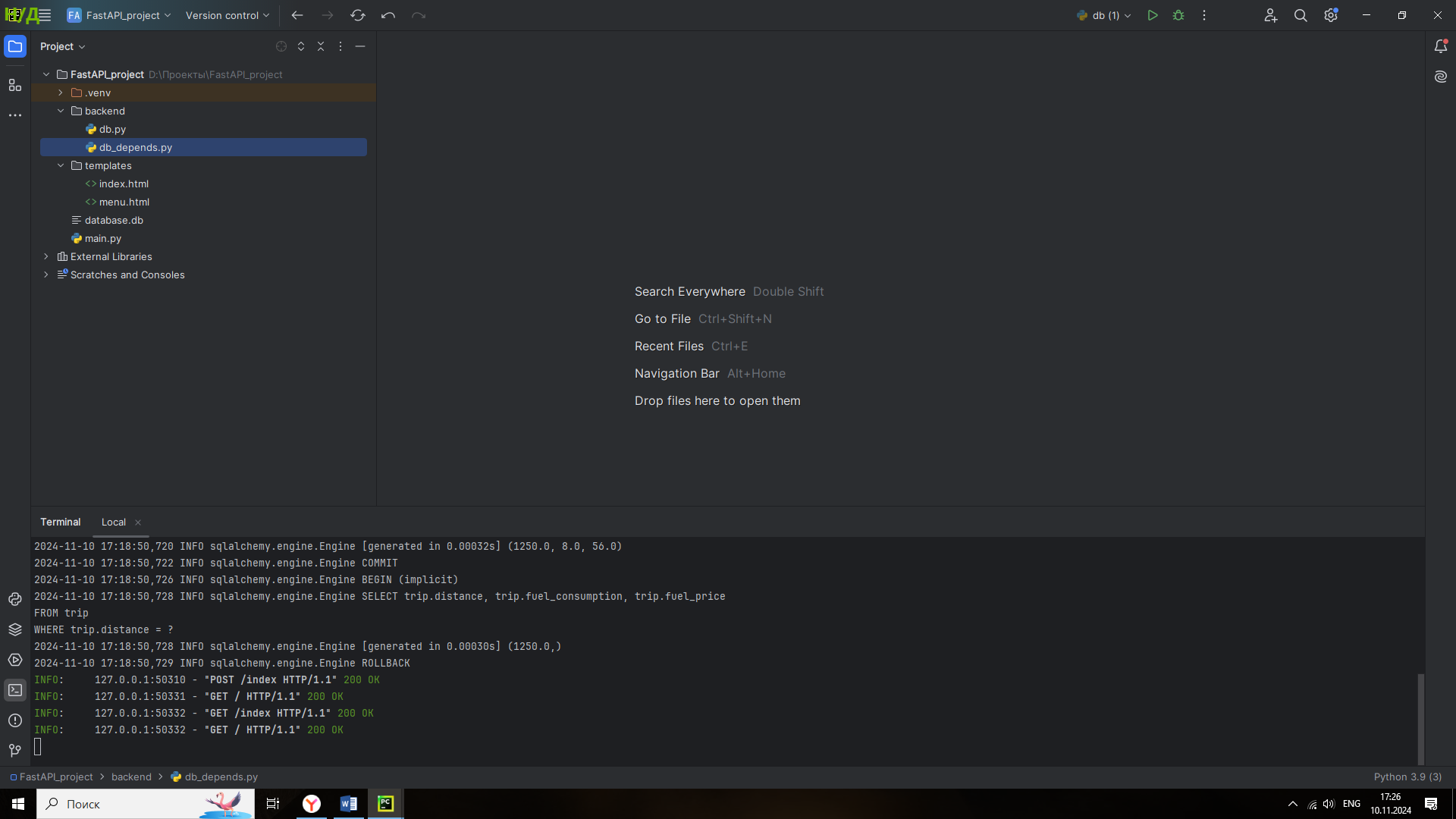
Структура проекта Django:



Структура проекта Flask:



Структура проекта FastAPI:



**Шаблоны страниц:**

menu.html – Шаблон главной страницы приложения.

Index.html – шаблон страницы с формами ввода.

**Анализ результатов**

Как уже было сказано – использовано три разных фреймворка для создания приложения: Django, Flask и FastAPI.

Поработав с каждым, я сделал определенные выводы:

**Django:**

Если взглянуть на скриншоты выше, по структуре проекта Django можно понять, насколько большой скелет нужен даже для такого маленького приложения. Фреймворк определенно требует некоторых навыков программирования, но не нужно быть специалистом чтобы в этом разобраться. Радует наличие встроенных миграций в базу данных, а также ORM.

**Flask:**

Для такого небольшого приложения Flask идеальный фреймворк. Всё максимально просто и понятно. В интернете бытует мнение, что Django проще для новичков, потому что загоняет пользователя в определенные рамки и не даёт свободу выбора. Для меня же Flask показался проще, если не учитывать подключение к базе данных.

**FastAPI:**

Самый быстрый, но при этом, по моему мнению, самый коварный из представленных инструментов. В интернете очень мало информации, очень много учебных статей построены на «Hello world» и не более. Подключение к базе данных вызвало целую бурю эмоций. Для такого маленького проекта не лучший выбор, асинхронность в этом случае не помогает. Для больших проектов возможно подойдёт отлично, но требует, по-моему, требует хороших навыков в программировании.

**ЗАКЛЮЧЕНИЕ**

Это приложение представляет собой платформу для пользователей, которым необходимо рассчитать стоимость своей поездки, зная всего три показателя – расстояние поездки, расход топлива автомобиля и стоимость топлива. Приложение полностью функционирует и готово к работе.

Дальнейшие планы:

Приложение можно развивать и дальше. Например, предлагать пользователю записаться на станцию технического обслуживания онлайн, или оповестить о погодных условиях в регионе на выбор пользователя. Также планируется сделать страницы регистрации и авторизации, чтобы расчеты пользователя могли сохраняться в его учетной записи.

Источники

* <https://practicum.yandex.ru/blog/framework-django/>
* <https://python-scripts.com/django-obzor>
* <https://ru.hexlet.io/blog/posts/freymvork-django-python-plyusy-minusy-vostrebovannost-na-rynke>
* <https://workspace.ru/tools/framework/flask/>
* <https://select-dev.ru/fastapi/>
* <https://tproger.ru/articles/pochemu-vam-stoit-vybrat-frejmvork-fastapi-dlja-svoego-sledujushhego-proekta>