**Дипломная работа**

**по теме: Анализ и сравнение написания web-приложений с использованием разных фреймворков.**

**Автор: Дмитриев Дмитрий Валерьянович**

**Группа №27: Python-разработчик**

**Содержание**

* Введение ... 3
* 1. Определение цели и задач исследования ... 4
* 1.1 Обзор проекта ... 5
* 2. Сравнение фреймворков ... 6
* 3. Методы и подходы к разработке ... 6
* 4. Обзор популярных фреймворков для разработки веб-приложений на Python ... 7
* 5. Разработка. Написание кода, реализующего функционал приложения ... 9
* 6. Разработка в соответствии с созданной документацией ... 10
* 7. Анализ и интерпретация результатов ... 16
* 8. Заключение ... 19
* Приложение 1. Пример файловой структуры проекта ... 21

**Введение**

**Обоснование выбора темы:**

1. В современном мире очень важно решать задачи быстро и удобно, в любой сфере нашего общества, будь то бизнес, творчество, наука и другие сферы. От этого зависит будущее развитие проекта и вовлечённость заинтересованных людей и обычных пользователей. Все современные проекты сопровождаются интернет-ресурсами и web-приложениями позволяющими увеличить аудиторию пользователей и упростить их вовлечённость в проект. И чем доступнее и понятнее для пользователя приложение, тем больше шансов, что оно станет хорошим помощником для решения тех или иных задач.

2. Потребности рынка: Программирование можно сравнить с творчеством. Если программист — писатель, то свой сюжет он формирует с помощью методов, классов и их взаимодействия. В настоящее время невозможно представить успешный проект, без хорошего web-приложения или интернет-сайта и разработчики в данном сегменте будут очень востребованы в обозримом будущем.

3. Практическая значимость: Работа в сегменте разработки web-приложений позволит применить знания на практике и отработать нужные качества, а в плане творческой части реализовать многие идеи, которые помогут привнести, что-то новое. В этом плане программисту может помочь использование фреймворков, которые упрощают разработку различных приложений и идей, за счёт подготовленного каркаса, чем фреймворк, по сути, и является. Фреймворки в программировании подходят для решения многих задач: создания интернет-магазинов, блогов, приложений с множеством тысяч активных пользователей и для проверки собственных идей. При разработке на базе фреймворка, программист должен учитывать его структуру и ограничения. Во фреймворке есть реализованные классы, предопределённые переменные, константы и готовые решения отдельной функциональности: валидация запросов, работа с БД, авторизация, работа с формами. По сути, программисту нужно взять интересующие блоки, связать их и использовать исходя из реализуемого проекта.

4. Личный интерес и перспективы: Разработка приложений является востребованной областью, в которой нужны специалисты с новыми идеями. Эта область позволит набраться опыта, развить и отработать личные навыки и навыки работы в команде, по созданию интересных проектов. А в плане творческой части, придумать и реализовать свои проекты, которые помогут открыть новые перспективы в данной отрасли.

Таким образом, выбор темы: “Анализ и сравнение написания web-приложений с использованием разных фреймворков.” становится не просто формальной задачей, а осознанным выбором актуальной темы с практической значимостью, потребностями рынка и личным интересом, что делает эту тему подходящей для написания дипломной работы.

**Определение цели и задач исследования**

Цель исследования: разработать web-приложение с использованием различных фреймворков и провести их сравнительный анализ.

Задачи исследования:

1. Обзор фреймворков для написания web-приложений и отработка модели на примере формы опросного листа: Провести сравнение использования базового функционала Django, Flask и FastAPI, по удобству отправки формы опросного листа.
2. Разработка сценариев сравнения: Разработать алгоритм и определить критерии сравнения базового функционала фреймворков.
3. Реализация проекта по созданию web-приложения и проведение тестирования: Создание приложения на разных фреймворках и отработка поставленных задач на практике.
4. Написание дипломной работы: Создание дипломной работы, оформленной по стандарту учебного заведения, с анализом результатов и выводом из полученных данных.

**Обзор проекта.**

Предполагается создание приложения на трёх разных фреймворках: Django, Flask и FastAPI. За основу взят сайт компании по продаже грузоподъёмного оборудования, с формой опросного листа для оформления заявок и регистрации запросов в базе данных. Таким образом администратор сможет быстро обрабатывать входящие заявки и распределять их по менеджерам, что существенно повысит работоспособность и сократит время обработки входящих запросов.

Оформление сайта будет реализовано с помощью Bootstrap.

Форма опросного листа, будет иметь внешний вид, с обязательными полями для заполнения:

* Название организации
* Контактное лицо
* Электронный адрес
* Адрес объекта
* Номер телефона
* Комментарий к заказу

**2. Основные понятия и определения**  
  
Обзор основных понятий в области создания web-приложений:  
  
1) Фреймворк (англ. framework — «каркас, структура»): готовая модель в программировании для быстрой разработки, на основе которой можно дописать собственный код. В разработке веб-приложений часто используются Django, FastAPI или Flask.

2) Веб-приложение (Web Application): клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.  
3) Бэкенд (серверная сторона) — это программно-аппаратная часть системы, которую не видят пользователи. Они не знают о работе внутренних процессов и не могут на них влиять.

4) Фронтенд (англ. frontend) — это область разработки ПО, которая фокусируется на создании пользовательского интерфейса и взаимодействии с пользователем. В основе фронтенда лежат языки HTML, CSS и JavaScript, которые применяются для создания веб-страниц, их оформления и добавления интерактивности.

5) CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete).

6) Базы данных - совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

7) ORM (англ. Object-Relational Mapping) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

8) API (англ. application programming interface) — программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими. (В противоположность пользовательскому интерфейсу, используемому для взаимодействия конечного пользователя с программой.) Обычно входит в описание какого-либо интернет-протокола, фреймворка или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

Данные термины помогают понять ключевые аспекты работы с написанием web-приложений.  
  
**3. Методы и подходы к разработке**

1) Планирование и анализ. Определение целей и требований к приложению.

За основу возьмём сайт по продаже грузоподъёмного оборудования, действующей организации с формой опросного листа, для создания входящих заявок от заказчиков.

2) Проектирование и дизайн. Разработка структуры приложения, его архитектуры и пользовательского интерфейса.

Приложение должно предоставлять информацию по товару, информацию о компании, контактную информацию и форму для заполнения, записывающую входящие заявки от заказчиков по интересующему товару, в базу данных. На данном этапе этого функционала будет достаточно. В дальнейшем можно будет расширить и улучшить модель.

Фронтенд будет реализован с помощью Bootstrap из которого возьмём основной стиль страниц. На главную страницу добавим карусель и пропишем ссылки для перехода. Подключим css стили и js скрипт.

Бэкенд реализуем с помощью Python, используя фреймворки Django, Flask и FastAPI, для реализации поставленных задач.

С помощью ORM подключим базу данных и пропишем модели. Поскольку нам не нужно создавать сложную БД, в классах будем использовать models.CharField()

**4. Обзор популярных фреймворков для разработки веб-приложений на Python**

**Django** — это веб-фреймворк с полным стеком с открытым исходным кодом, написанный на Python, который следует архитектурному шаблону MTV (модель-шаблон-представление) и принципу DRY (Не повторяйтесь). Он был создан и был публично опубликован в 2005 году.

Преимущества:

Полно функциональность: Django поставляется с множеством встроенных функций, включая административную панель, ORM и систему аутентификации.

Безопасность: Он предлагает мощные инструменты для обработки общих угроз безопасности.

Недостатки:

Меньшая гибкость: Его строгая структура может быть ограничивающей для некоторых проектов.

Сложность: Из-за большого числа функций кривая обучения может быть крутой для новичков.

**Flask** - это удобный для начинающих веб-фреймворк с открытым исходным кодом, написанный на Python, который следует архитектурному шаблону MVC (модель-представление-контроллер) и поддерживает шаблоны Jinja, которые можно использовать для создания представлений. О было сделано в 2004 году. Название “Flask” является отсылкой к более раннему фреймворку Bottle.

Преимущества:

Гибкость и простота: Flask предоставляет минималистичный подход к веб-разработке.

Легко расширяем: Можно легко добавить дополнительные функции с помощью расширений.

Недостатки:

Меньше встроенных функций: Необходимо вручную добавлять многие компоненты.

Масштабируемость: Для крупных проектов может потребоваться дополнительная настройка.

**FastAPI** также является микро-веб-фреймворком Starlette-ASGI (Asynchronous Server Gateway Interface) с открытым исходным кодом, написанным на Python и используемым для создания RESTful API. Он был сделан в декабре 2018 года. Название основано на скорости разработки API, отсюда и "FastAPI’.

Преимущества:

Быстрая производительность: Один из самых быстрых фреймворков для Python.

Поддержка асинхронного программирования: Идеально подходит для обработки параллельных запросов.

Недостатки:

Молодость: Еще не так широко принят, как Django или Flask.

Ограниченное количество ресурсов и сообщества: Не так много обучающих материалов и поддержки сообщества.

**5. Разработка. Написание кода, реализующего функционал приложения.**

Планирование и анализ требований  
  
Выбор фреймворка и инструментов: Поскольку цель задания сравнить приложение в разных фреймворках, были последовательно задействованы Django, Flask и FastAPI.  
  
Определение структуры приложения: Разработка схемы архитектуры, включающей фронтенд, бэкенд и обмен с базой данных.  
  
Разработка прототипа трёх приложений и адаптация фронтенда к ним.  
  
Реализация основного функционала: Создание базовой версии приложения с минимально необходимым функционалом для тестирования и демонстрации.  
  
Разработка ключевого функционала  
  
Создание post запроса из формы опросного листа с последующей записью в базу данных.  
  
Создание API для взаимодействия с фронтендом: Разработка REST API для обработки запросов от пользовательского интерфейса.  
  
Реализация пользовательского интерфейса: Создание форм и страниц, отображения результатов и взаимодействия с пользователем.  
  
Основные требования  
Создание : Пользователь должен иметь возможность заполнить форму запроса и получить подтверждение создания заявки.  
  
  
Отображение результатов: При успешной отправке запроса, пользователь должен получить подтверждение о регистрации заявки.  
  
Технические требования  
  
Фронтенд: использовать HTML, CSS, JavaScript, Bootstrap  
  
Бэкенд: Python с использованием фреймворков Django, Flask и FastAPI.  
  
База данных: Использование базы данных SQlite для хранения данных о зарегистрированных запросах.  
  
API: Разработка API для взаимодействия между фронтендом и бэкендом.

**6. Разработка в соответствии с созданной документацией**  
  
Планирование разработки  
  
Разработка была разделена на несколько основных этапов: проектирование интерфейса, реализация серверной логики и создания формы опросного листа с последующем post-запросом и записью в БД. Планирование задач и отслеживание прогресса осуществлялось в программе Microsoft Office.  
  
Разработка  
  
Фронтенд-разработка:  
  
Созданы пользовательский интерфейс с использованием Bootstrap для стилизации.

На Главной странице представлена карусель с возможностью перехода в каталог, информация о различных услугах с возможностью перехода на страницу с более подробной информацией и информация по выполненным объектам.

Рис.1 Главная страница.

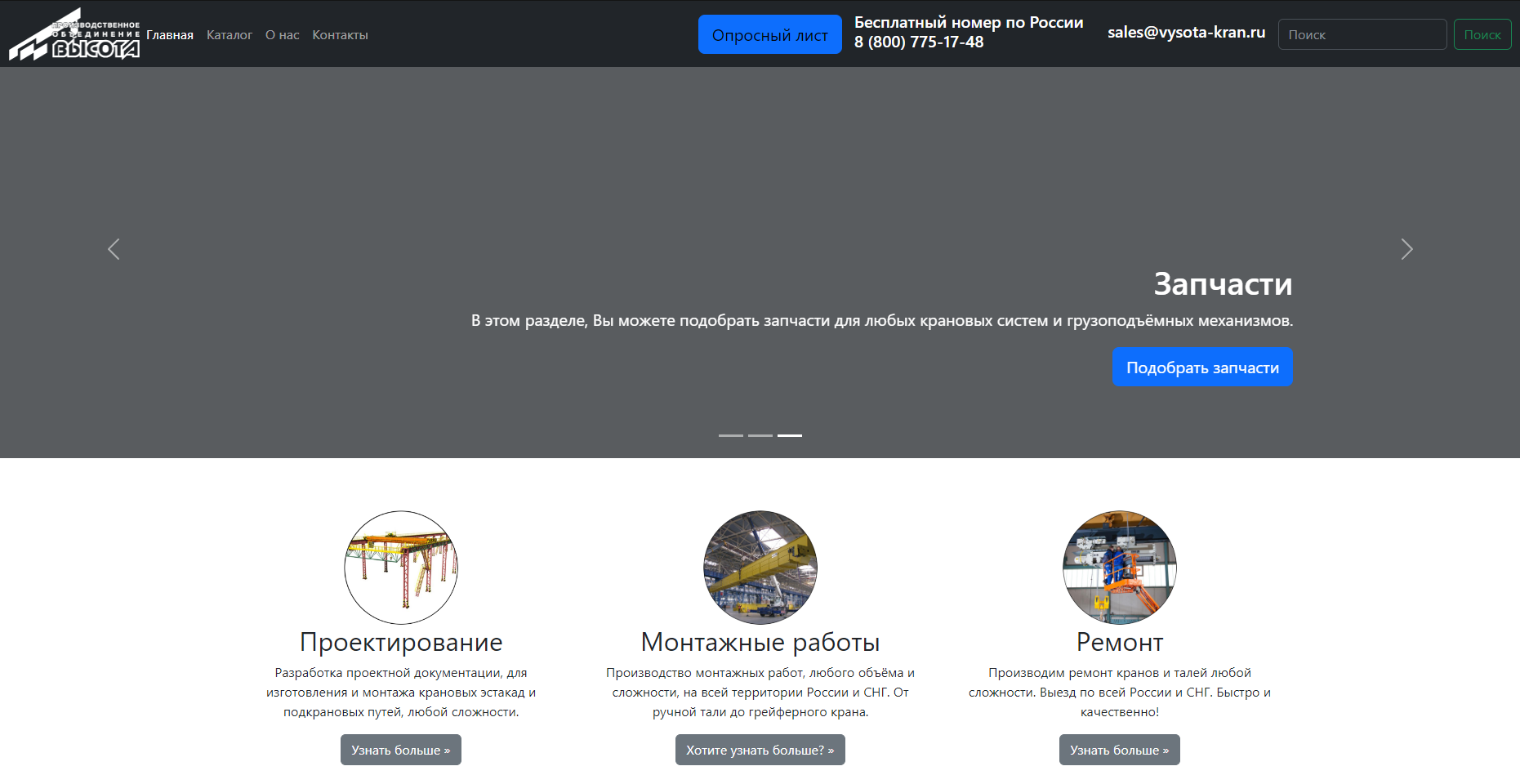
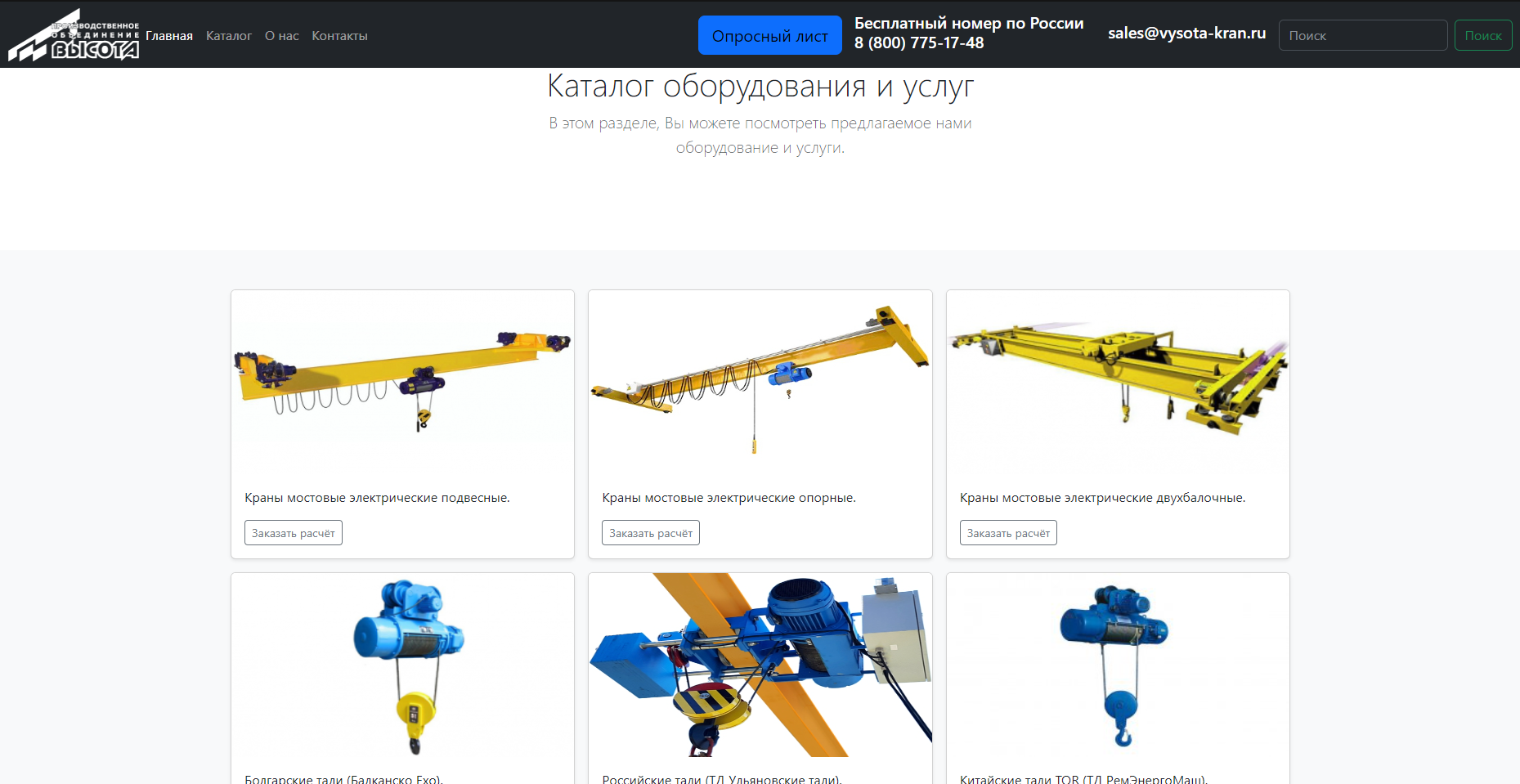
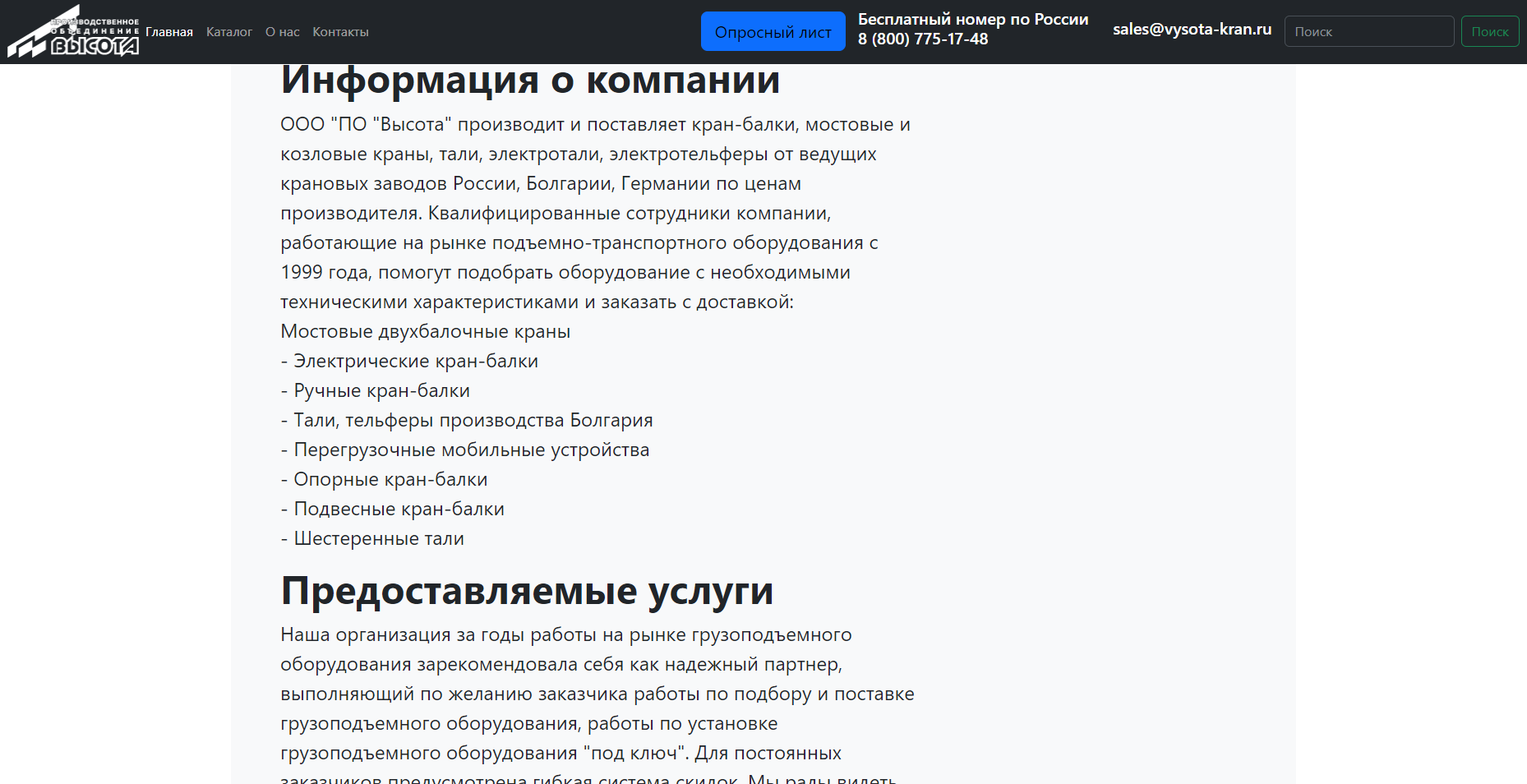
  
Каталог оборудования представлен с разными разделами и кнопками перехода.

Рис.2 Каталог оборудования.



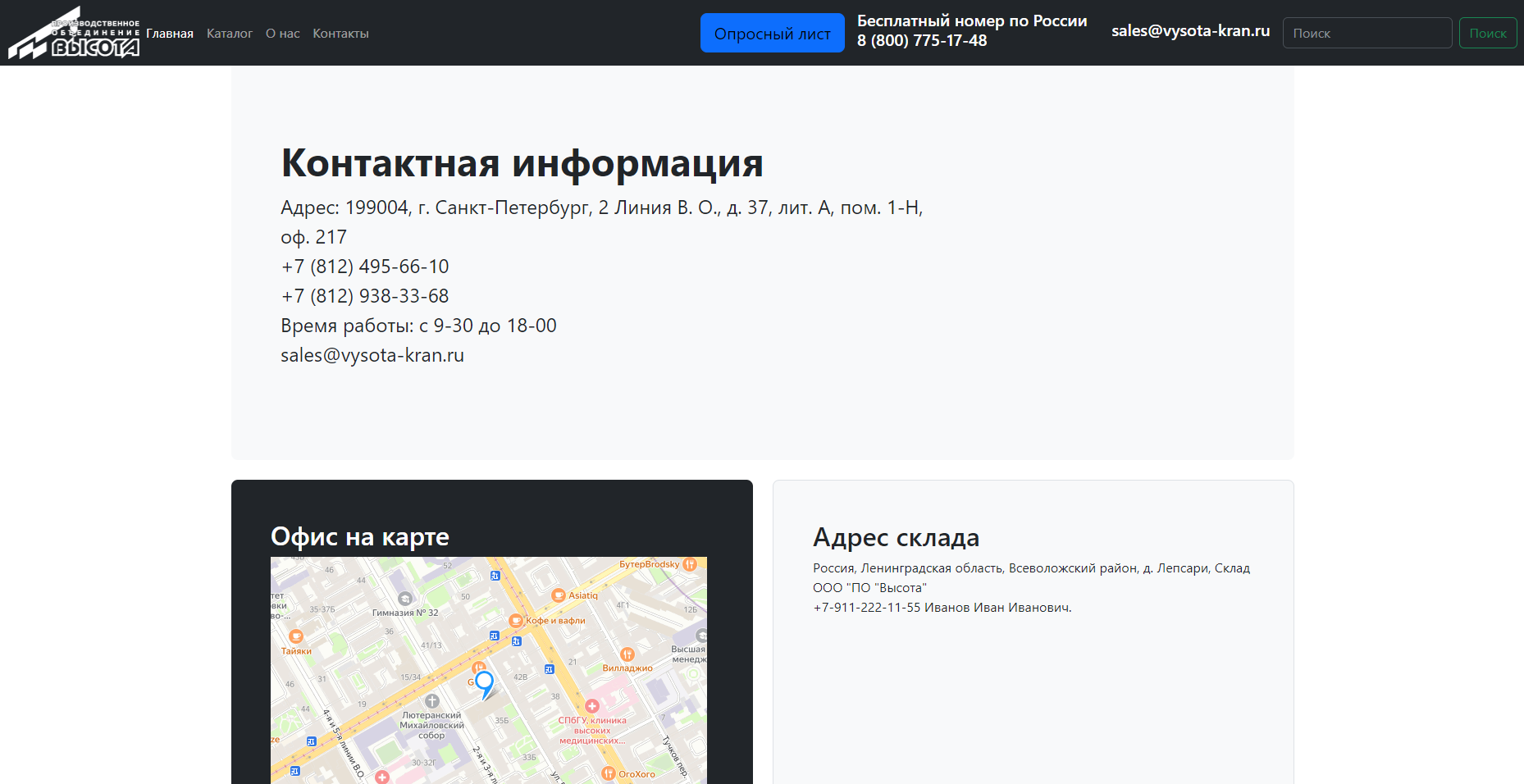
На странице информации отображена основная информация о компании и предоставляемых услугах.

Рис.3 Страница информации о компании.



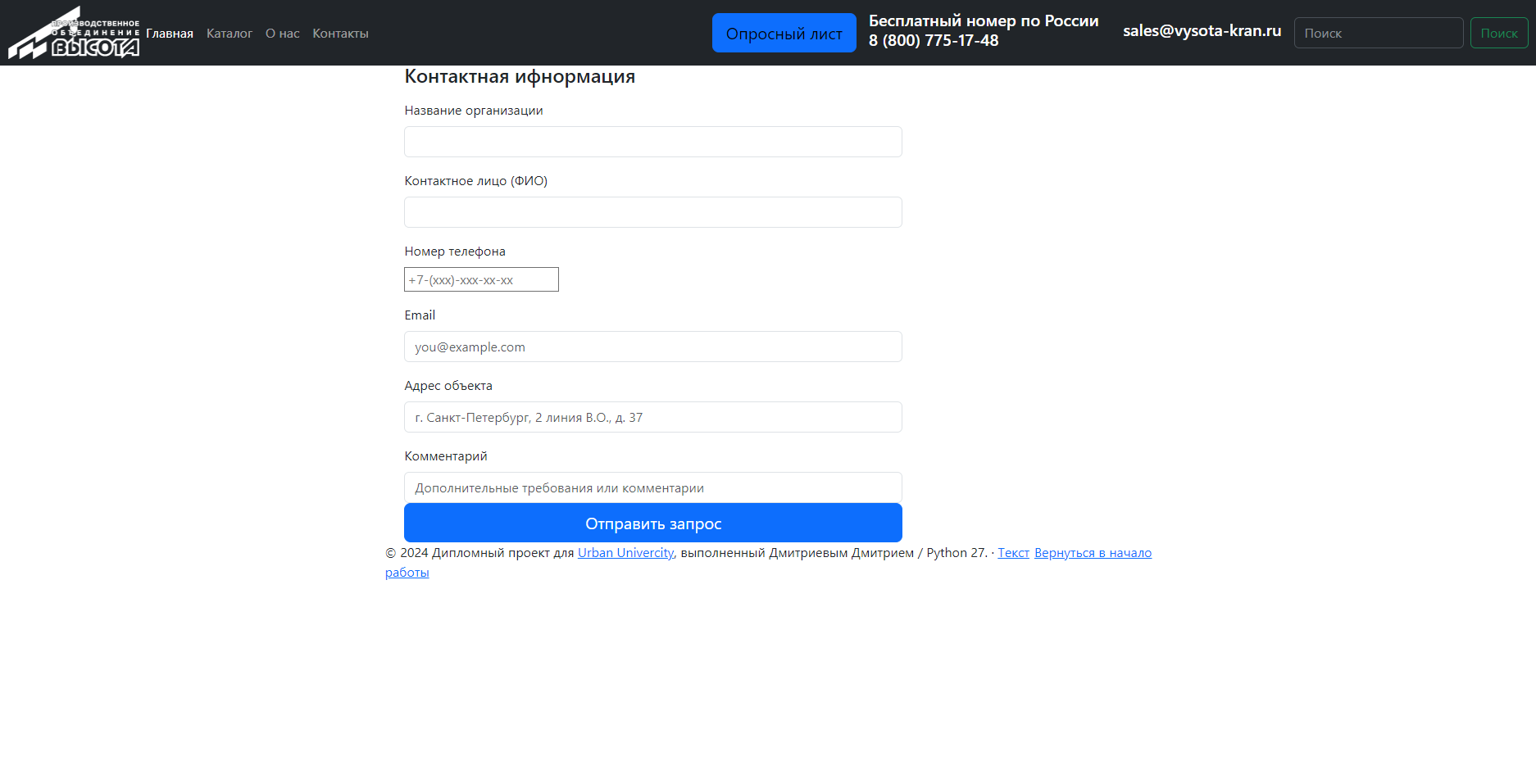
Страница контактной информации с указанием всех необходимых данных и расположением адреса на карте.

Рис.4 Страница контактной информации.



Разработана форма опросного листа, создан класс для заполнения формы и проработана логика обмена с базой данных.

Рис.5 Форма заполнения заявки.



**Бэкенд-разработка:**  
  
Реализована серверная логика с использованием фреймворков Django, Flask и FastAPI.  
  
Настроен маршрут для обработки формы опросного листа, записи в базу данных и передачи результата в фронтенд.  
  
Реализована обработка формы опросного листа и отправки подтверждения регистрации заявки, в используемых фреймворках.

Подключена и проверена база данных. Все, правильно заполненные, запросы регистрируются.

Рис.6 Файловая структура Django

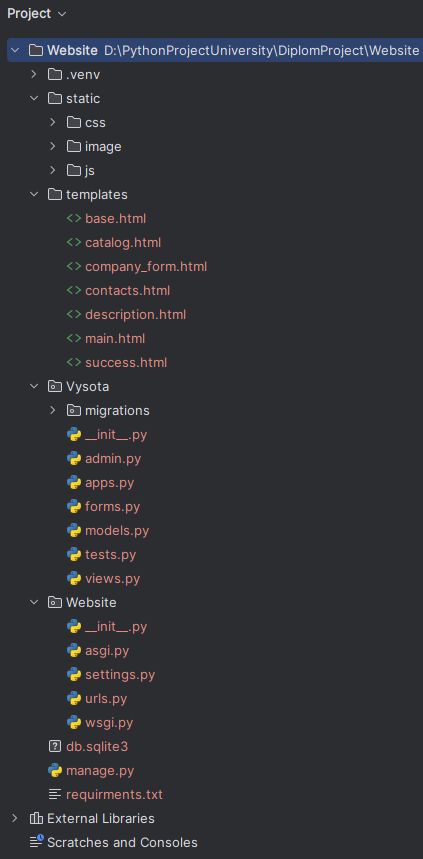


Рис.7 Файловая структура Flask

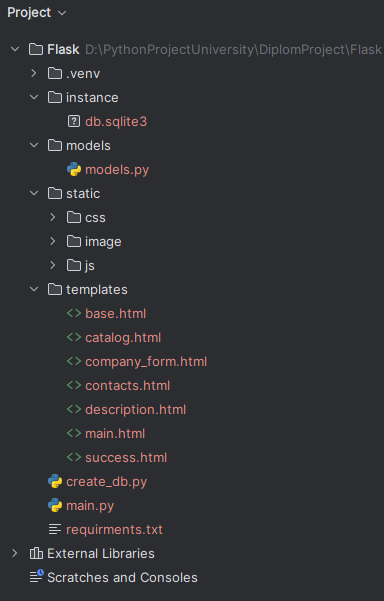
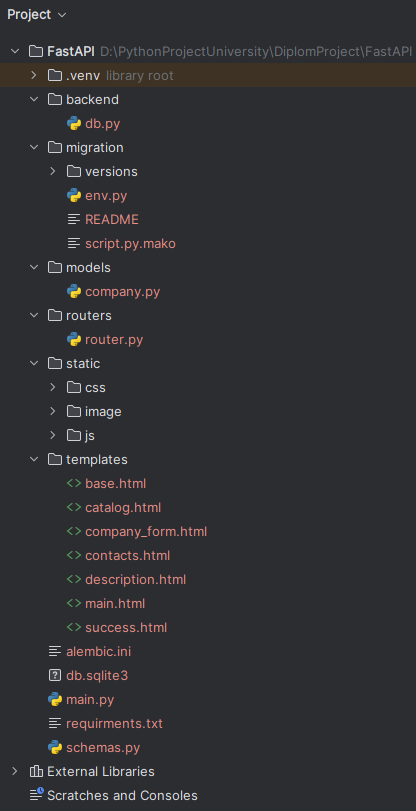


Рис.8 Файловая структура FastAPI



**7. Анализ и интерпретация результатов**  
  
Сравнение моделей  
  
Было протестировано web-приложение созданное на трёх фреймворках, Django, Flask и FastAPI.

Django:

* Удобное создание базовой структуры приложения
* Много настроек для работы с различными проектами
* Удобен для создания средних и больших приложений и при этом крайне неудобен для создания малых приложений по причине большого количества настроек
* Удобный интерфейс встроенного сервера с админ панелью
* Большая и подробная документация

Flask:

* Удобное использование CRUD-запросов, когда можно указать get и post запросы одновременно и настроить логику работы внутри функции.
* Удобный микро-фреймворк для создания средних и малых приложений
* Можно быстро создать и настроить приложение
* Достаточно прост в освоении
* Подробная документация и большая аудитория пользователей, которые могут дать совет или подсказку

FastAPI:

* Поддержка асинхронных и синхронных представлений
* Быстрое создание микросервисов или запросов
* Удобный встроенный Swagger
* Активно использует декораторы

**Интерпретация результатов**  
  
Все три фреймворка показали себя с хорошей стороны.

Реализация приложения на Django заняла больше всего времени. Но в процессе создания приложения возникло множество идей, как можно будет улучшить приложение в будущем. Удобно начинать новый проект и быстро создавать файловую структуру, подключать базу данных и прописывать связи и маршруты. Использование на маленьких приложениях не рекомендуется, так как для этого можно использовать более удобные фреймворки.

Реализация приложения на Flask не вызвала больших затруднений, так как на этом фреймворке можно использовать без модификации те же шаблоны, а CRUD-запросы мне показались удобнее, чем в том же FastAPI из-за возможности присваивать get и post запрос одновременно и прописывать логику его работы. Существенным минусом является, что у него нет админ панели или Swaggera как у Django и FastAPI, но к этому можно привыкнуть. Форма опросного листа, которую я реализовал в этих приложениях, запустилась сразу без всяких проблем.

Реализация приложения на FastAPI заставила искать информацию о том, каким образом можно отправить опросный лист в базу данных. В Swaggere post запрос создавался и записывался в базу данных, но через форму опросного листа запрос уходить отказывался. Но в конечном счёте проблему решить удалось. Функционал всех трёх фреймворков отлично себя показал.  
  
Рекомендации по выбору модели в зависимости от проекта  
  
Для больших приложений, требующих большого функционала, рекомендуется использовать Django. В маленьких приложениях его лучше не использовать, по причине большого количества настроек.  
  
Для приложений, где важна скорость создания, можно рассмотреть использование фреймворка FastAPI, поскольку он считается одним из самых быстрых. Имеет поддержку асинхронных и синхронных представлений, есть встроенная документация Swagger. Но как показала практика, по нему меньше документации и сложнее поиск решения проблем.

Flask же отличается своей простотой и гибкостью, делая его доступным для начинающих, в то же время предлагая мощные возможности для создания сложных приложений.

**8. Заключение**  
  
Обзор выполненной работы  
  
Проектирование и разработка веб-приложения на фреймворках Django, Flask и FastAPI были успешно завершены в соответствии с изначально созданной документацией. Приложение включает функционал CRUD-запросов и формы опросного листа с записью зарегистрированных заявок в базу данных.  
  
Реализованное приложение соответствует требованиям и демонстрирует стабильную работу во всех использованных фреймворках.  
  
  
**Дальнейшие планы**  
  
  
Планируется добавление новых функций в приложение на фреймворке Django, таких как: переход на postgreSQL, добавление встроенной CRM с созданием и присвоением каждому заказчику своих запросов, быстрый расчёт (внутри заказа) цены оборудования и быстрое формирование коммерческого предложения с присвоением исходящего номера в отдельную базу данных. Отслеживание пользователем своих заказчиков и добавление напоминаний о звонке или встрече.

**Приложение 1. Requirements**

Рис.10 Requirments Django

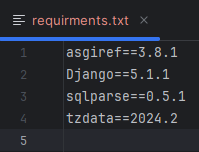


Рис.11 Requirments Flask

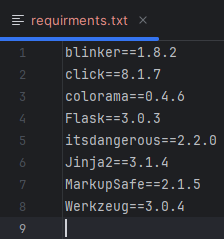


Рис.11 Requirments FastAPI

