Geekbrains

**Создание высокоэффективного сервиса для покупок одежды**

Программа:

Python-Разработчик

Григорович Даниил Андреевич

д. Заречье

2024

**Содержание**

[Содержание](#_heading=h.7mt9x4b37ce)

[Введение](#_heading=h.1fob9te)

[Теоретическая и практическая главы](#_heading=h.hst8x8f1r8tz)

[Заключение](#_heading=h.dhwmmf3wq1kk)

[Список используемой литературы](#_heading=h.m8jvbwhv7ujg)

[Приложения](#_heading=h.s6k7t2lwcp0k)

Введение (4 стр.)

Глава 1. Основы используемых технологий

* 1. Python
  2. JavaScript
  3. PostgreSQL, SQL
  4. Redis
  5. Docker
  6. CSS, HTML, Jinja2
  7. Git, GitHub
  8. FastAPI

Глава 2. Основы работы веб-приложений

2.1 Определение веб-приложения

2.2 Архитектура веб-приложений

2.3 Клиентская часть

2.4 Серверная часть

2.5 Взаимодействие с базой данных

2.6 Протоколы и форматы обмена данными

2.7 Аутентификация и авторизация

2.8 Обработка ошибок и безопасность

2.9 Разработка и развертывание

Глава 3. Основы бэкенд разработки с Python и FastAPI(5-7 стр.)

3.1 Патерны

3.2 Асинхронность

3.3 Валидация данных

3.4 Документация API

3.5 Управление зависимостями

3.6 Безопасность

3.7 Тестирование

3.8 Разработка в команде

3.9 Интеграция с базами данных

Глава 4. Разработка веб-приложения(8-стр.)

4.1 Система аутентификации

4.2 Система корзины и работа с Redis

4.3 Управление запасами товаров

4.4 Управление заказами и картами

4.5 Интеграция с платежным сервисом

4.6 Применение Jinja2

Заключение (~ 4 стр.)

Список используемой литературы

Приложения

**Введение**

Случается такое, что нужно разработать веб-приложение не имея полного технического задания. Этот проект представляет из себя веб-приложение, которое было разработано для изучения новых технологий и закрепления уже изученных знаний.

**Тема проекта:** Создания и организацию процесса создания веб-приложения в условиях отсутствия полного технического задания на примере создания веб-магазина одежды.

**Цель:** Изучение создания и организацию процесса создания веб-приложения в условиях отсутствия полного технического задания на примере создания веб-магазина одежды.

**Какую проблему решает проект:** Случается такое, что нужно разработать веб-приложение не имея полного технического задания.

**Задачи:**

1. Изучить литературу и другой материал касающийся темы иследования.
2. Рассмотреть основные технологии и фремворки для разработки веб-приложений.
3. Ознакомиться с основными принципами разработки и взаимодействия с используемыми технологиями и инструментами.
4. Начать разработку с использыванием обозначеными ниже технологиями и инструментами.

**Инструменты** **и технологии:**

* Языки программирования: Python, JavaScript
* Фреймворк: FastAPI
* Базы данных: PostgreSQL, Redis
* Развертывание: Docker
* Язык стилей, разметки и шаблонизатор: CSS, HTML, Jinja2
* Система управления версиями: Git

**Состав команды**:

Проект был разработан с учатием двух человек:

* Хохлов Андрей: Ответственный за разработку бэкенд-системы, API для регестрации, управления пользователями и товарами, а также общую архитектуру проекта.
* Даниил Григорович: Ответственный за разработку логики корзины покупок, включая добавление и удаление товаров, аутентификации, оформление заказов, интеграцию с платёжными системами, управление запасами товаров и оптимизацию производительности системы.

**Теоретическая и практическая главы**

1. **Основы используемых технологий**
   1. Python — мультипарадигмальный высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным в том плане, что всё является объектами. Необычной особенностью языка является выделение блоков кода отступами. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации. Сам же язык известен как интерпретируемый и используется в том числе для написания скриптов. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как C или C++.

Python является мультипарадигменным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, метапрограммирование, функциональное программирование и асинхронное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Аспектно-ориентированное программирование частично поддерживается через декораторы, более полноценная поддержка обеспечивается дополнительными фреймворками. Такие методики как контрактное и логическое программирование можно реализовать с помощью библиотек или расширений. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора (GIL), высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Python имеет обширную стандартную библиотеку, которая включает в себя модули для работы с файлами, сетями, базами данных, веб-разработкой и многими другими задачами. Это делает его особенно популярным для разработки веб-приложений (например, с использованием фреймворков Django и Flask), научных вычислений (с библиотеками NumPy и SciPy), анализа данных (с библиотеками Pandas и Matplotlib) и машинного обучения (с библиотеками TensorFlow и scikit-learn). Python также активно используется в автоматизации задач, написании скриптов и тестировании.

1.2. JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java. Языком JavaScript не владеет какая-либо организация или компания, что отличает его от ряда языков программирования, используемых в веб-разработке. В последние годы JavaScript стал основным языком для разработки как клиентских, так и серверных приложений благодаря таким технологиям, как Node.js, что позволило использовать его для создания масштабируемых сетевых приложений.

JavaScript также поддерживает асинхронное программирование через промисы и async/await, что упрощает работу с асинхронными операциями, такими как запросы к серверу. Существуют многочисленные библиотеки и фреймворки, такие как React, Angular и Vue.js, которые значительно упрощают разработку сложных пользовательских интерфейсов и одностраничных приложений. JavaScript продолжает развиваться, и новые стандарты ECMAScript добавляют новые возможности и улучшения, что делает его одним из самых популярных языков программирования в мире.

Кроме того, JavaScript активно используется в разработке мобильных приложений с помощью таких фреймворков, как React Native и Ionic. Он также находит применение в разработке серверных приложений с использованием Node.js, что позволяет создавать высокопроизводительные и масштабируемые приложения. JavaScript поддерживает работу с различными базами данных, включая NoSQL базы данных, такие как MongoDB, что делает его универсальным инструментом для разработки современных веб-приложений.

Среди недостатков JavaScript можно выделить:

* Слабая типизация: Поскольку JavaScript использует слабую типизацию, это может привести к неожиданным ошибкам, когда переменные могут принимать значения разных типов без явного указания. Это требует от разработчиков внимательности и тщательного тестирования.
* Проблемы с производительностью: Хотя JavaScript достаточно быстр, особенно с современными движками (например, V8 от Google), он все же может быть медленнее по сравнению с компилируемыми языками, такими как C или C++. Это может стать проблемой при выполнении вычислительно сложных задач.
* Безопасность: JavaScript, особенно в контексте веб-приложений, подвержен различным уязвимостям, таким как XSS (межсайтовый скриптинг) и CSRF (межсайтовая подделка запроса). Разработчики должны быть внимательны к безопасности своих приложений и использовать соответствующие меры защиты.
* Разнообразие фреймворков и библиотек: Хотя наличие множества фреймворков и библиотек является преимуществом, это также может привести к путанице. Разработчикам может быть сложно выбрать подходящий инструмент для своей задачи, особенно с учетом быстрого развития экосистемы JavaScript.
* Отладка: Отладка JavaScript-кода может быть сложной задачей, особенно в больших приложениях. Хотя современные инструменты разработчика в браузерах значительно упростили этот процесс, все же могут возникать трудности при отслеживании ошибок и производительности.

Несмотря на эти недостатки, JavaScript продолжает оставаться одним из самых популярных языков программирования в мире благодаря своей универсальности, поддержке со стороны сообщества и постоянному развитию. Он активно используется в веб-разработке, мобильных приложениях, серверных приложениях и даже в разработке игр. С появлением таких технологий, как WebAssembly, JavaScript также расширяет свои возможности, позволяя разработчикам использовать другие языки программирования в веб-приложениях.

1.3. PostgreSQL, SQL

SQL (аббр. от англ. Structured Query Language — «язык структурированных запросов») — декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. Является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных. В общем случае SQL (без ряда современных расширений) считается языком программирования неполным по Тьюрингу, но вместе с тем стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений.

Изначально SQL был основным способом работы пользователя с базой данных и позволял выполнять следующий набор операций:

* создание в базе данных новой таблицы;
* добавление в таблицу новых записей;
* изменение записей;
* удаление записей;
* выборка записей из одной или нескольких таблиц (в соответствии с заданным условием);
* изменение структур таблиц.

Со временем SQL усложнился — обогатился новыми конструкциями, обеспечил возможность описания и управления новыми хранимыми объектами (например, индексы, представления, триггеры и хранимые процедуры) — и стал приобретать черты, свойственные языкам программирования. Используется вместе с СУБД (Система Управления Базами Данных).

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД). Существует в реализациях для множества UNIX-подобных платформ, включая различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows. Сильными сторонами PostgreSQL считаются:

высокопроизводительные и надёжные механизмы транзакций и репликации;

расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке C;

Преимущества PostgreSQL:

* наследование;
* возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;
* встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
* расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

PostgreSQL также поддерживает сложные запросы, включая объединения, подзапросы и оконные функции, что делает его мощным инструментом для анализа данных. Благодаря своей надежности и гибкости, PostgreSQL широко используется в различных областях,Все таблицы изображающие структуру моделей находятся в “Приложения”

1.4. Redis

Redis — резидентная система управления базами данных класса NoSQL, работающая со структурами данных типа «ключ — значение». Используется как для баз данных, так и для реализации кэшей, брокеров сообщений. Изначально выпускалась под свободной лицензией, с 2024 года — под лицензией SSPL.

Ориентирована на достижение максимальной производительности на атомарных операциях (заявляется о приблизительно 100 тыс. SET- и GET-запросов в секунду на Linux-сервере начального уровня). Написана на Си, интерфейсы доступа созданы для большинства основных языков программирования. Хранит базу данных в оперативной памяти, снабжена механизмами снимков и журналирования для обеспечения постоянного хранения (на дисках, твердотельных накопителях).

Также предоставляет операции для реализации механизма обмена сообщениями в шаблоне «издатель — подписчик»: с его помощью приложения могут создавать каналы, подписываться на них и помещать в каналы сообщения, которые будут получены всеми подписчиками (как IRC-чат). Поддерживает репликацию данных с основных узлов на несколько подчинённых (англ. master — slave replication). Также поддерживает транзакции и пакетную обработку команд (выполнение пакета команд, получение пакета результатов).

Работает на большинстве POSIX-систем, таких как Linux, \*BSD, Mac OS X без каких-либо дополнений, компания-спонсор проекта поддерживает систему на Linux и Mac OS X. Официальной поддержки для сборок Windows нет, но доступны некоторые опции, позволяющие обеспечить работу Redis на этой системе, сообщается о работах Microsoft по переносу Redis на Windows.

В версии 2.6.0 добавлена поддержка Lua, позволяющего выполнять запросы на сервере. Lua позволяет атомарно совершить произвольную обработку данных на сервере и предназначена для использования в случае, когда нельзя достичь того же результата с использованием стандартных команд.

Среди языков программирования, имеющих библиотеки для работы с Redis, — Си, C++, C#, Clojure, Лисп, Erlang, Java, JavaScript, Haskell, Lua, Perl, PHP, Python, Ruby, Scala, Go, Tcl, Rust, Swift, Nim. Все данные Redis хранит в виде словаря, в котором ключи связаны со своими значениями. Одно из ключевых отличий Redis от других хранилищ данных заключается в том, что значения этих ключей не ограничиваются строками.

Redis также поддерживает различные структуры данных, такие как строки, списки, множества, отсортированные множества, хеши и битовые карты. Это позволяет разработчикам эффективно использовать Redis для различных сценариев, включая кэширование, управление сессиями, очереди задач и многое другое.

Одной из ключевых особенностей Redis является его поддержка механизма Pub/Sub, который позволяет приложениям обмениваться сообщениями в реальном времени. Это делает Redis отличным выбором для разработки приложений, требующих мгновенной передачи данных, таких как чаты или уведомления.

Redis также предлагает возможность использования кластеров, что позволяет распределять данные по нескольким узлам и обеспечивать высокую доступность и масштабируемость. В случае сбоя одного из узлов, Redis может автоматически перенаправлять запросы на другие узлы, что минимизирует время простоя.

Кроме того, Redis поддерживает различные механизмы управления памятью, включая LRU (Least Recently Used) и LFU (Least Frequently Used) для автоматического удаления старых данных, когда память переполняется. Это позволяет эффективно управлять ресурсами и поддерживать высокую производительность.

Redis имеет активное сообщество и множество инструментов для мониторинга и управления, таких как Redis CLI, Redis Desktop Manager и другие. Эти инструменты помогают разработчикам и администраторам легко управлять базами данных и отслеживать производительность.

Наконец, Redis активно используется в промышленности и имеет множество успешных примеров применения в таких компаниях, как GitHub, Snapchat, Twitter и многих других, что подтверждает его надежность и эффективность в реальных условиях.

1.5 Docker

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Позволяет «упаковать» приложение со всем своим окружением и зависимостями в контейнер, который может быть развёрнут на любой Linux-системе с поддержкой контрольных групп в ядре, а также предоставляет набор команд для управления этими контейнерами. Изначально использовал возможности LXC, с 2015 года начал использовать собственную библиотеку, абстрагирующую виртуализационные возможности ядра Linux — libcontainer. С появлением Open Container Initiative начался переход от монолитной к модульной архитектуре.

В данном проекте мы используем Docker для создания изолированной среды, в которой разворачиваются все необходимые сервисы. Это позволяет упростить процесс развертывания и обеспечить согласованность окружения на разных машинах. В файле docker-compose.yaml описаны два основных сервиса: Redis и PostgreSQL.

Сервис Redis используется как in-memory хранилище данных, что позволяет ускорить доступ к часто запрашиваемым данным и улучшить производительность приложения. В конфигурации указано, что Redis будет использовать образ redis:alpine, который является легковесной версией Redis. Порт 6379, на котором работает Redis, проброшен на хост-машину, что позволяет другим сервисам взаимодействовать с Redis. Данные Redis будут сохраняться в локальной папке ./data, что обеспечивает их сохранность при перезапуске контейнера.

Сервис PostgreSQL используется для хранения данных приложения. В конфигурации указаны переменные окружения, необходимые для инициализации базы данных, такие как имя пользователя, пароль и имя базы данных. Образ postgres:latest гарантирует, что мы используем последнюю стабильную версию PostgreSQL. Порт 5432, на котором работает PostgreSQL, также проброшен на хост-машину. Данные базы данных будут храниться в локальной папке ./pgdata, что позволяет избежать потери данных при перезапуске контейнера.

Использование Docker и Docker Compose в этом проекте позволяет легко управлять зависимостями и конфигурацией, а также упрощает процесс развертывания приложения на различных средах, таких как локальная разработка, тестирование и продакшн. Это обеспечивает гибкость и масштабируемость, что является важным аспектом современного подхода к разработке программного обеспечения. Содержимое Docker-compose.yaml можно посмотреть в “Приложения”.

1.6 CSS, HTML, Jinja2

HTML (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — стандартизированный язык гипертекстовой разметки документов для просмотра веб-страниц в браузере. Веб-браузеры получают HTML документ от сервера по протоколам HTTP/HTTPS или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора. Элементы HTML являются строительными блоками HTML страниц. С помощью HTML разные конструкции, изображения и другие объекты, такие как интерактивная веб-форма, могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы HTML выделяются тегами, записанными с использованием угловых скобок. Такие теги, как <img> и <input>, напрямую вводят контент на страницу. Другие теги, такие как <p>, окружают и оформляют текст внутри себя и могут включать другие теги в качестве подэлементов. Браузеры не отображают HTML-теги, но используют их для интерпретации содержимого страницы. Язык XHTML является более строгим вариантом HTML, он следует синтаксису XML и является приложением языка XML в области разметки гипертекста. В HTML можно встроить язык программирования JavaScript с помощью тега.

Также включение CSS в HTML позволяет задавать внешний вид и макет страницы. CSS — формальный язык декодирования и описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки (чаще всего HTML или XHTML). Также может применяться к любым XML-документам, например, к SVG или XUL. Использование CSS позволяет создателям веб-страниц задавать цвета, шрифты, стили, расположение отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS является ограждение и отделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана) или при выводе устройствами, использующими шрифт Брайля.

CSS также поддерживает различные методологии и подходы к стилизации, такие как BEM (Block Element Modifier), SMACSS (Scalable and Modular Architecture for CSS) и OOCSS (Object-Oriented CSS), которые помогают организовать и структурировать код, делая его более читаемым и поддерживаемым. С помощью CSS можно использовать медиа-запросы для создания адаптивного дизайна, который позволяет веб-страницам корректно отображаться на различных устройствах, включая мобильные телефоны, планшеты и настольные компьютеры.

Современные CSS-фреймворки, такие как Bootstrap и Tailwind CSS, предоставляют готовые компоненты и утилиты, которые значительно ускоряют процесс разработки и позволяют создавать стильные и функциональные интерфейсы без необходимости писать много пользовательского кода. CSS также поддерживает анимации и переходы, что позволяет добавлять динамические эффекты на веб-страницы, улучшая пользовательский опыт и взаимодействие с интерфейсом.

Jinja2 — это шаблонизатор для языка программирования Python. Он подобен шаблонизатору Django, но предоставляет Python-подобные выражения, обеспечивая исполнение шаблонов в песочнице. Это текстовый шаблонизатор, поэтому он может быть использован для создания любого вида разметки, а также исходного кода. Лицензирован под лицензией BSD.

Шаблонизатор Jinja позволяет настраивать теги, фильтры, тесты и глобальные переменные. В отличие от шаблонизатора Django, Jinja2 позволяет конструктору шаблонов вызывать функции с аргументами на объектах.

Jinja2, как и Smarty, также поставляется с простой в использовании системой фильтров, похожей на конвейер Unix.

1.7 Git, GitHub

Git — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года; координатор — Дзюн Хамано.

Среди проектов, использующих Git, — ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

Программа является свободной и выпущена под лицензией GNU GPL версии 2. По умолчанию используется TCP-порт 9418.

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки.

Веб-сервис основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub, Inc (ранее Logical Awesome). Сервис бесплатен для проектов с открытым исходным кодом и (с 2019 года) небольших частных проектов, предоставляя им все возможности (включая SSL[9]), а для крупных корпоративных проектов предлагаются различные платные тарифные планы.

Создатели сайта называют GitHub «социальной сетью для разработчиков».

Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых.

С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.

Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок.

Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.

Можно создавать приватные репозитории, которые будут видны только вам и выбранным вами людям. Раньше такая возможность была платной.

Есть возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса.

Код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта.

1.8 FastAPI

FastAPI — веб-фреймворк для создания API, написанный на Python. Один из самых быстрых и популярных (после Django и Flask) веб-фреймворков, написанных на Python (на 2023 год).

FastAPI активно использует декораторы, аннотации типов и интроспекцию кода, что позволяет уменьшить количество шаблонного кода в веб-приложении. FastAPI автоматически генерирует и отображает документацию согласно спецификации OpenAPI.

В основе FastAPI лежат две библиотеки — Starlette (ASGI-фреймворк) и Pydantic (для описания схем данных); FastAPI склеивает их и реализует некоторые дополнительные возможности — регистрацию представлений через внедрение зависимостей, работу с аутентификацией и авторизацией, автоматическую генерацию документации и другое. Возможно использование как асинхронных, так и синхронных представлений.

1. **Основы работы веб-приложения**

2.1 Определение веб-приложения

Веб-приложение — это программное обеспечение, которое работает на веб-сервере и доступно пользователям через веб-браузер. В отличие от традиционных настольных приложений, которые требуют установки на локальном устройстве, веб-приложения могут быть использованы на различных платформах и устройствах, включая компьютеры, планшеты и смартфоны. Это делает их более доступными и удобными для пользователей, так как они могут получить доступ к приложению из любой точки мира, где есть интернет. Веб-приложения могут быть статическими, где контент не меняется, или динамическими, где информация обновляется в зависимости от действий пользователя или данных из базы данных.

2.2 Архитектура веб-приложений

Веб-приложения обычно имеют трехуровневую архитектуру, состоящую из клиентского, серверного и базового уровней. Клиентский уровень представляет собой интерфейс, с которым взаимодействует пользователь. Он включает в себя элементы управления, такие как кнопки, формы и меню, которые позволяют пользователю выполнять действия. Серверный уровень обрабатывает запросы от клиента, выполняет бизнес-логику и взаимодействует с базой данных. Базовый уровень отвечает за хранение данных и может включать в себя реляционные или нереляционные базы данных, такие как MySQL, PostgreSQL, MongoDB или Redis. Эта архитектура позволяет разделить ответственность между различными компонентами, что упрощает разработку, тестирование и поддержку приложения.

2.3 Клиентская часть

Клиентская часть веб-приложения включает в себя HTML, CSS и JavaScript. HTML (Hypertext Markup Language) отвечает за структуру страницы, определяя, какие элементы будут отображаться и как они будут организованы. HTML-документ состоит из различных тегов, которые описывают заголовки, параграфы, изображения и другие элементы. CSS (Cascading Style Sheets) используется для стилизации этих элементов, позволяя разработчикам задавать цвета, шрифты, отступы и другие визуальные характеристики. CSS также поддерживает адаптивный дизайн, что позволяет веб-приложениям корректно отображаться на различных устройствах и экранах. JavaScript добавляет интерактивность на страницу, позволяя пользователям взаимодействовать с элементами, отправлять данные на сервер и обновлять содержимое без перезагрузки страницы. Это создает более плавный и отзывчивый пользовательский интерфейс, что особенно важно в условиях высокой конкуренции на рынке веб-приложений.

2.4 Серверная часть

Серверная часть веб-приложения обрабатывает запросы от клиента и выполняет бизнес-логику. Сервер может быть написан на различных языках программирования, таких как Python, Java, PHP, Ruby или Node.js. Каждый из этих языков имеет свои особенности и преимущества, что позволяет разработчикам выбирать наиболее подходящий инструмент для конкретной задачи. Когда клиент отправляет запрос, сервер принимает его, обрабатывает и отправляет ответ обратно. Это может включать в себя выполнение операций с базой данных, обработку данных и формирование HTML-страниц для отображения пользователю. Сервер также может управлять сессиями пользователей, обеспечивая сохранение состояния между запросами. Это особенно важно для приложений, где пользователи должны оставаться аутентифицированными и иметь доступ к своим данным на протяжении всего сеанса..

2.5 Взаимодействие с базой данных

Веб-приложения часто используют базы данных для хранения и управления данными. Серверная часть взаимодействует с базой данных через SQL (Structured Query Language) или другие языки запросов, чтобы извлекать, обновлять или удалять данные. Это позволяет веб-приложениям динамически изменять содержимое в зависимости от действий пользователя. Например, в интернет-магазине база данных может хранить информацию о товарах, заказах и пользователях, а сервер может выполнять запросы для отображения актуальной информации на страницах приложения.

Протоколы и форматы обмена данными: Веб-приложения используют различные протоколы для обмена данными между клиентом и сервером. Наиболее распространенным является HTTP (Hypertext Transfer Protocol), который определяет, как сообщения передаются по сети. В последние годы также стал популярным протокол HTTPS (HTTP Secure), который обеспечивает шифрование данных для повышения безопасности. Для передачи данных часто используются форматы JSON (JavaScript Object Notation) или XML (eXtensible Markup Language), которые позволяют структурировать информацию и легко обмениваться ею между клиентом и сервером.

2.6 Аутентификация и авторизация

Веб-приложения используют различные протоколы для обмена данными между клиентом и сервером. Наиболее распространенным является HTTP (Hypertext Transfer Protocol), который определяет, как сообщения передаются по сети. В последние годы также стал популярным протокол HTTPS (HTTP Secure), который обеспечивает шифрование данных для повышения безопасности. Это особенно важно для веб-приложений, которые обрабатывают конфиденциальную информацию, такую как пароли и платежные данные. Для передачи данных часто используются форматы JSON (JavaScript Object Notation) или XML (eXtensible Markup Language), которые позволяют структурировать информацию и легко обмениваться ею между клиентом и сервером. JSON стал особенно популярным благодаря своей простоте и легкости в использовании с JavaScript, что делает его идеальным для современных веб-приложений.

2.7 Аутентификация и авторизация

Веб-приложения часто требуют аутентификации пользователей для обеспечения безопасности. Аутентификация — это процесс проверки личности пользователя, который может включать в себя ввод логина и пароля. После успешной аутентификации пользователю может быть предоставлен доступ к определенным функциям приложения. Авторизация, в свою очередь, определяет, какие действия пользователь может выполнять в приложении. Для этого могут использоваться различные методы, такие как JWT (JSON Web Token), OAuth или сессии, которые помогают управлять правами доступа и защищать данные. Важно, чтобы разработчики внедряли надежные механизмы аутентификации и авторизации, чтобы предотвратить несанкционированный доступ к данным и функциям приложения.

2.8 Обработка ошибок и безопасность

Веб-приложения должны быть защищены от различных угроз, таких как SQL-инъекции, XSS (межсайтовый скриптинг) и CSRF (межсайтовая подделка запросов). SQL-инъекции происходят, когда злоумышленник вставляет вредоносный SQL-код в запрос, что может привести к утечке данных или повреждению базы данных. XSS-атаки позволяют злоумышленникам внедрять скрипты на страницы, которые затем выполняются в браузерах других пользователей. CSRF-атаки используют доверие пользователя к веб-приложению, чтобы выполнять нежелательные действия от его имени. Для защиты от этих угроз разработчики применяют различные методы, такие как валидация данных, использование HTTPS, регулярные обновления программного обеспечения и внедрение механизмов защиты, таких как Content Security Policy (CSP). Обработка ошибок также играет важную роль в обеспечении безопасности и удобства использования. Веб-приложения должны корректно обрабатывать ошибки, предоставляя пользователю понятные сообщения и не раскрывая внутренние детали системы, которые могут быть использованы злоумышленниками.

2.9 Разработка и развертывание

Разработка веб-приложений включает в себя проектирование, кодирование, тестирование и развертывание. Разработчики используют различные инструменты и фреймворки, такие как Flask, Django, React или Angular, чтобы ускорить процесс разработки и улучшить качество кода. Эти фреймворки предоставляют готовые решения для распространенных задач, таких как маршрутизация, управление состоянием и работа с базами данных, что позволяет разработчикам сосредоточиться на бизнес-логике приложения. Процесс тестирования включает в себя как функциональное, так и нагрузочное тестирование, чтобы убедиться, что приложение работает корректно и может справляться с ожидаемыми объемами трафика. После завершения разработки приложение развертывается на веб-сервере, чтобы стать доступным для пользователей. Это может включать в себя настройку серверного окружения, конфигурацию базы данных и обеспечение безопасности. Разработчики также должны учитывать вопросы масштабируемости и производительности, чтобы приложение могло справляться с увеличением нагрузки по мере роста числа пользователей.

1. **Основы бэкенд разработки с Python и FastAPI**

Бэкенд-разработка на Python с использованием FastAPI включает в себя несколько ключевых концепций и принципов, которые помогают создавать эффективные и масштабируемые веб-приложения. Вот основные аспекты этой теории.

3.1. Архитектурные паттерны

MVC (Model-View-Controller): Хотя FastAPI не навязывает конкретную архитектуру, многие разработчики используют MVC или его вариации для разделения логики приложения на модели (данные), представления (интерфейс) и контроллеры (бизнес-логика). Это позволяет улучшить организацию кода и упростить его сопровождение. Важно отметить, что использование MVC способствует более четкому разделению ответственности, что делает приложение более модульным и легким для тестирования.

RESTful API: FastAPI идеально подходит для создания RESTful API, где ресурсы представляются в виде URL, а действия над ними выполняются с помощью стандартных HTTP-методов (GET, POST, PUT, DELETE). RESTful подход позволяет создавать гибкие и масштабируемые приложения, которые легко интегрируются с другими сервисами и клиентами. Важно также учитывать принципы REST, такие как stateless взаимодействие и использование кэширования, что может значительно повысить производительность приложения.

3.2. Асинхронность

Асинхронное программирование: FastAPI поддерживает асинхронные функции, что позволяет обрабатывать множество запросов одновременно, не блокируя выполнение. Это особенно полезно для I/O-операций, таких как запросы к базе данных или внешним API. Асинхронность позволяет значительно повысить производительность приложения, особенно в условиях высокой нагрузки, когда требуется обрабатывать большое количество одновременных запросов.

Кроме того, использование асинхронных функций упрощает работу с библиотеками, которые также поддерживают асинхронность, такими как httpx для выполнения HTTP-запросов или databases для работы с базами данных. Это позволяет нам эффективно управлять ресурсами и минимизировать время ожидания, что в свою очередь улучшает пользовательский опыт.

Асинхронность также способствует более эффективному использованию серверных ресурсов. Вместо того чтобы блокировать потоки во время ожидания завершения операций ввода-вывода, FastAPI может обрабатывать другие запросы, что позволяет лучше масштабировать приложение и снижать затраты на инфраструктуру.

Важно отметить, что при разработке асинхронных приложений необходимо учитывать особенности работы с состоянием и синхронизацией данных. Мы применяем подходы, такие как использование асинхронных контекстных менеджеров и библиотек для управления состоянием, чтобы избежать проблем, связанных с конкурентным доступом к ресурсам.

Таким образом, асинхронное программирование в FastAPI не только повышает производительность и масштабируемость приложения, но и позволяет создавать более отзывчивые и эффективные веб-сервисы, способные справляться с высокими нагрузками и обеспечивать качественное взаимодействие с пользователями.

3.3. Валидация данных

Pydantic: FastAPI использует Pydantic для валидации и сериализации данных. Это позволяет автоматически проверять входные данные и генерировать документацию API на основе аннотаций типов. Pydantic обеспечивает строгую типизацию, что помогает избежать ошибок на ранних этапах разработки и улучшает качество кода. Кроме того, использование Pydantic позволяет легко управлять сложными структурами данных и обеспечивает высокую степень гибкости при работе с различными форматами данных.

В нашем проекте мы также использовали схемы (schemas) для работы с базой данных. Схемы представляют собой модели данных, которые описывают структуру и типы данных, используемых в приложении. Они позволяют нам четко определить, какие поля должны присутствовать в запросах и ответах API, а также обеспечивают валидацию данных на уровне модели.

Использование схем упрощает процесс работы с базой данных, так как мы можем легко преобразовывать данные между форматом, используемым в API, и форматом, необходимым для хранения в базе данных. Это позволяет избежать дублирования кода и делает его более читаемым и поддерживаемым.

Кроме того, схемы помогают в автоматизации процесса создания документации API, так как FastAPI может использовать аннотации типов и описания полей

для генерации документации, что делает API более понятным для разработчиков и пользователей.

Таким образом, интеграция Pydantic и схем в нашем проекте не только улучшает валидацию данных, но и способствует более эффективному взаимодействию между различными компонентами приложения, повышая его надежность и удобство в использовании.

3.4. Документация API

Автоматическая генерация документации: FastAPI автоматически создает интерактивную документацию (Swagger UI и ReDoc) на основе определений маршрутов и моделей данных, что упрощает тестирование и использование API. Это позволяет разработчикам и пользователям API быстро ознакомиться с доступными эндпоинтами, их параметрами и ответами, что значительно ускоряет процесс интеграции и тестирования.

В самом языке Python также предусмотрены возможности для документирования кода. Каждый объект в Python, будь то функция, класс или модуль, может содержать строки документации, известные как docstring. Эти строки размещаются сразу после определения функции или класса и служат для описания их назначения, параметров и возвращаемых значений. Это позволяет разработчикам легко понимать, как использовать различные компоненты кода, просто заглянув в их docstring.

Кроме того, Python поддерживает встроенные функции, такие как help(), которые позволяют пользователям получать информацию о функциях и классах, включая их docstring. Это делает процесс изучения кода более интуитивным, так как разработчики могут быстро получать справочную информацию прямо в интерпретаторе.

Также стоит отметить, что Python позволяет организовывать код в модули и пакеты, что способствует структурированию проекта и упрощает навигацию по коду. Каждый модуль может содержать свои собственные docstring, что позволяет документировать функциональность на уровне модуля.

Таким образом, возможности Python для документирования кода, такие как использование docstring и встроенные функции для получения справочной информации, делают процесс создания и поддержки документации более простым и эффективным. Это особенно важно в контексте командной разработки, где ясность и доступность информации о коде играют ключевую роль в успешной интеграции и сотрудничестве между разработчиками.

3.5. Управление зависимостями

Dependency Injection: FastAPI предоставляет механизм для управления зависимостями, что позволяет легко внедрять компоненты, такие как базы данных, сервисы и другие ресурсы, в ваши обработчики. Это способствует более чистой архитектуре приложения и упрощает тестирование, так как зависимости можно легко заменять на Mokcs(моки) или Stabs(стабы).

3.6. Безопасность

Аутентификация и авторизация: FastAPI поддерживает различные схемы аутентификации и предоставляет инструменты для защиты API и управления доступом к ресурсам. При разработке фронтенд части веб-приложения мы сделали защиту от XSS-атак, что является важным аспектом безопасности. Также стоит отметить, что FastAPI поддерживает OAuth2 и JWT (JSON Web Tokens), что позволяет реализовать безопасные механизмы аутентификации и авторизации пользователей.

3.7. Тестирование

Тестирование API: FastAPI упрощает процесс тестирования с помощью встроенных инструментов, позволяя разработчикам писать тесты для своих маршрутов и логики. Для тестирования в этом проекте мы использовали библиотеку “pytest”, которая предоставляет мощные возможности для написания и организации тестов. Также стоит отметить, что FastAPI позволяет легко интегрировать тесты с клиентами, что упрощает процесс проверки функциональности API.

В рамках тестирования мы создали несколько тестов, которые охватывают основные функции нашего приложения. Для начала мы настроили тестовую базу данных, используя асинхронные фикстуры, которые обеспечивают подключение и отключение от базы данных перед и после выполнения тестов. Это позволяет нам изолировать тесты и гарантировать, что они не влияют друг на друга.

Мы использовали TestClient из FastAPI для выполнения HTTP-запросов к нашему приложению. Это позволяет нам тестировать маршруты, как если бы они были вызваны реальными пользователями. Например, в тесте test\_homepage мы проверяем, что главная страница возвращает статус 200 и содержит ожидаемый текст.

Другие тесты, такие как test\_get\_edit\_item\_form и test\_edit\_item, проверяют функциональность редактирования товаров. Мы отправляем GET и POST запросы, проверяем статус код ответа и содержимое HTML, чтобы убедиться, что формы отображаются и обрабатываются корректно. В случае некорректных данных, таких как неверный формат цены, тест test\_edit\_item\_invalid\_data проверяет, что сервер возвращает статус 422, что указывает на ошибку валидации.

Также мы тестируем добавление новых товаров с помощью test\_add\_item\_valid\_data, где проверяем, что товар успешно добавляется при корректных данных. В случае отсутствия обязательных полей, тест test\_add\_item\_missing\_data подтверждает, что сервер возвращает статус 400 или 422, что указывает на ошибку в запросе.

Кроме того, мы проверяем функциональность поиска товаров с помощью тестов test\_search\_valid\_query и test\_search\_empty\_query, которые подтверждают, что запросы к поисковому эндпоинту возвращают ожидаемые результаты.

Таким образом, тестирование API с использованием FastAPI и pytest позволяет нам уверенно проверять функциональность приложения, обеспечивая его надежность и стабильность.

3.8. Разработка в команде

Для разработки в команде необходимо использовать системы контроля версий. В данном проекте мы решили использовать систему контроля версий “Git” и веб-сервис “GitHub”. Это позволяет эффективно управлять изменениями в коде, отслеживать историю изменений и облегчает совместную работу над проектом. Использование Git также позволяет организовать процесс ревью кода, что способствует повышению качества и безопасности приложения.

Кроме того, GitHub предоставляет множество инструментов для управления проектами, таких как Issues и Projects, которые помогают команде отслеживать задачи, планировать спринты и управлять рабочими процессами. Это позволяет всем участникам команды быть в курсе текущего статуса задач и упрощает коммуникацию между разработчиками.

Важным аспектом командной разработки является использование ветвления (branching) в Git. Команда может создавать отдельные ветки для новых функций или исправлений, что позволяет работать над несколькими задачами одновременно, не мешая основной кодовой базе. После завершения работы над задачей, изменения могут быть объединены с основной веткой через Pull Request, что также предоставляет возможность для обсуждения и ревью кода перед его интеграцией.

Кроме того, использование GitHub Actions для автоматизации процессов CI/CD (непрерывной интеграции и непрерывного развертывания) позволяет команде автоматически тестировать и развертывать приложение при каждом изменении в коде. Это значительно ускоряет процесс разработки и помогает выявлять ошибки на ранних этапах.

Таким образом, использование Git и GitHub не только упрощает управление кодом, но и способствует улучшению качества разработки, повышению прозрачности процессов и более эффективному взаимодействию между членами команды.

Существует еще множество веб-сервисов для управления системами контроля версий, и каждый из них имеет свои особенности и преимущества. Рассмотрим несколько популярных альтернатив GitHub, а затем выделим, почему именно GitHub является предпочтительным выбором для многих команд.

* GitLab: GitLab — это платформа, которая предлагает функции, аналогичные GitHub, включая управление репозиториями, инструменты для совместной работы и CI/CD. GitLab также предоставляет возможность хостинга на собственных серверах, что может быть важным для организаций, которые требуют повышенной безопасности и контроля над своими данными. Однако, несмотря на свои сильные стороны, GitLab может быть сложнее в настройке и использовании для новичков.
* Bitbucket: Bitbucket, принадлежащий компании Atlassian, также предлагает управление репозиториями и интеграцию с другими инструментами Atlassian, такими как Jira и Confluence. Bitbucket поддерживает как Git, так и Mercurial, что делает его гибким выбором для команд, использующих разные системы контроля версий. Однако Bitbucket имеет ограничения на количество пользователей в бесплатной версии, что может быть проблемой для небольших команд.
* SourceForge: SourceForge был одним из первых веб-сервисов для хостинга проектов с открытым исходным кодом. Он предлагает управление репозиториями, трекеры ошибок и другие инструменты для разработки. Однако в последние годы SourceForge потерял популярность из-за более современных и удобных платформ, таких как GitHub и GitLab. Интерфейс SourceForge может показаться устаревшим, и он не предлагает такой же уровень интеграции с современными инструментами разработки.
* Azure DevOps: Azure DevOps от Microsoft предоставляет полный набор инструментов для управления проектами, включая репозитории Git, CI/CD, управление задачами и тестирование. Это мощная платформа, особенно для команд, работающих в экосистеме Microsoft. Однако для небольших команд или стартапов Azure DevOps может быть избыточным и сложным в использовании.

Теперь давайте выделим, почему GitHub является предпочтительным выбором для многих команд:

* Широкое распространение и сообщество: GitHub является одной из самых популярных платформ для хостинга проектов с открытым исходным кодом. Это означает, что многие разработчики уже знакомы с его интерфейсом и функциональностью, что упрощает процесс обучения и интеграции новых членов команды.
* Интуитивно понятный интерфейс: GitHub предлагает простой и понятный интерфейс, который делает управление репозиториями и совместную работу более доступными для разработчиков всех уровней. Это особенно важно для команд, где могут быть новички.
* Интеграция с другими инструментами: GitHub легко интегрируется с множеством сторонних инструментов и сервисов, таких как CI/CD системы, инструменты для управления проектами и системы мониторинга. Это позволяет командам настраивать свои рабочие процессы в соответствии с их потребностями.
* Поддержка открытого исходного кода: GitHub активно поддерживает проекты с открытым исходным кодом, что позволяет разработчикам легко делиться своими проектами и получать вклад от сообщества. Это создает возможности для сотрудничества и обмена знаниями.
* GitHub Actions: Эта функция позволяет автоматизировать рабочие процессы, такие как тестирование и развертывание, прямо в репозитории. Это значительно упрощает настройку CI/CD и позволяет командам быстрее реагировать на изменения в коде.
* Социальные функции: GitHub предлагает множество социальных функций, таких как возможность следить за другими разработчиками, ставить звезды на репозитории и участвовать в обсуждениях. Это создает активное сообщество, которое может помочь в решении проблем и обмене опытом.

Таким образом, хотя существуют и другие веб-сервисы для управления системами контроля версий, GitHub выделяется благодаря своему удобству, широкому сообществу и множеству интеграций, что делает его идеальным выбором для команд, стремящихся к эффективной и качественной разработке.

3.9. Интеграция с базами данных

ORM (Object-Relational Mapping): Часто используются ORM, такие как SQLAlchemy, для упрощения работы с SQL базами данных, что позволяет разработчикам работать с данными на уровне объектов, а не писать сложные SQL-запросы. Это значительно ускоряет процесс разработки, так как разработчики могут сосредоточиться на бизнес-логике приложения, а не на деталях реализации базы данных. SQLAlchemy предоставляет мощные инструменты для работы с моделями данных, включая возможность создания сложных запросов, управления транзакциями и миграциями базы данных.

Также в проекте использовалась No-SQL база данных “Redis”. Redis идеально подходит для хранения временных данных, кэширования и работы с высоконагруженными приложениями благодаря своей высокой скорости и эффективности. В отличие от реляционных баз данных, Redis использует структуру данных ключ-значение, что позволяет быстро получать доступ к данным и выполнять операции записи и чтения.

В проекте, Redis, используется для реализации системы корзины, где данные о товарах, добавленных пользователем, хранятся в виде хешей. Это позволяет эффективно управлять состоянием корзины, добавлять и удалять товары, а также быстро получать информацию о текущем содержимом корзины.

Интеграция с обеими базами данных (SQL и No-SQL) позволяет проекту использовать преимущества каждой из них, обеспечивая гибкость и производительность. SQLAlchemy обеспечивает надежное управление структурированными данными, в то время как Redis позволяет эффективно работать с неструктурированными данными и временными хранилищами. Такой подход позволяет разработчикам создавать масштабируемые и высокопроизводительные приложения, которые могут легко адаптироваться к изменяющимся требованиям бизнеса.Эти принципы и концепции создают основу для эффективной бэкенд-разработки на Python с использованием FastAPI, позволяя создавать высокопроизводительные, безопасные и удобные в использовании веб-приложения.

1. **Разработка веб-приложения**

4.1. Система аутентификации

Система аутентификации — это механизм, который позволяет проверить личность пользователя или устройства, чтобы обеспечить доступ к защищенным ресурсам. Она играет ключевую роль в обеспечении безопасности информационных систем. Для ее реализации я использовал технологию JWT. JWT (JSON Web Token) — это открытый стандарт, который определяет компактный и автономный способ безопасной передачи информации между сторонами в формате JSON. Для работы с JWT в Python есть библиотека python-jose. JWT часто используется для аутентификации и авторизации в веб-приложениях, так как позволяет передавать данные о пользователе и его правах доступа. В данном проекте структура JWT представляет из себя id пользователя, email пользователя, имя пользователя, дату истечения токена и id токена. В файле cookies/jwt.py находятся все функции для управления токенами: создание, дешифровка и проверка срока истечения токена.

Для того чтобы получить JWT, пользователь должен пройти аутентификацию. Функция располагается по эндпоинту “/login/”. Пользователь должен ввести данные от аккаунта: email и пароль. Если данные совпадают с данными, которые хранятся в базе данных, то пользователю возвращается JWT. Пароли в базе данных хранятся в зашифрованном виде. Также есть система выхода из аккаунта. Когда пользователь попадает на эндпоинт “/logout/”, то сайт проверяет, есть ли у пользователя JWT. Если нет, то происходит переадресация на эндпоинт “/login/”; в ином случае у пользователя удаляется JWT и происходит переадресация на “/”.

Фронтенд часть для аутентификации реализована с помощью HTML и JavaScript. В файле login\_form.html создается форма для ввода email и пароля. При отправке формы JavaScript код, находящийся в login.js, обрабатывает данные и отправляет их на сервер. Если вход успешен, пользователю показывается сообщение об успешном входе, а в случае ошибки — сообщение с деталями проблемы. Это позволяет пользователю быстро реагировать на ошибки ввода и улучшает общий пользовательский опыт.

4.2. Система корзины и работа с Redis

При реализации системы корзины я использовал нереляционную базу данных “Redis”. Для ее запуска на операционной системе “Windows” необходимо использовать приложение для развертывания, для данного проекта был выбран “Docker”. Docker-compose файл находится по пути “docker/docker-compose.yaml”. В отличие от реляционных баз данных, где данные хранятся в таблицах с фиксированной схемой и связями между ними, нереляционные базы данных (или NoSQL базы данных) используют более гибкие структуры данных, такие как документы, графы или ключ-значение, что позволяет хранить неструктурированные и полуструктурированные данные. Также они имеют преимущество в скорости на запись, что делает их идеальными для приложений, требующих высокой производительности и быстрого доступа к данным.

Когда пользователь, имеющий JWT, нажимает на изображение предмета гардероба на главной странице или странице поиска, с помощью JavaScript происходит запрос на эндпоинт “/add/”, с указанием аргументов position\_id (id предмета гардероба) и amount (количество предметов для добавления, по умолчанию равно 1). Затем функция “updateCart”, находящаяся в файле “static/js/ajax\_add\_toc\_cart.js”, обновляет страницу корзины, делая запрос на эндпоинт “/cart/get/”. Все функции для управления корзиной находятся в файле “cart/cart.py”. Все функции для управления “Redis” находятся в файле “cart/redis\_client.py”.

Фронтенд часть корзины представлена в файле cart.html, где отображаются все добавленные товары. Каждый товар имеет кнопку "Заказать" и "Удалить из корзины", что позволяет пользователю легко управлять содержимым своей корзины. JavaScript код, находящийся в файлах add\_order.js, delete\_item.js и add\_order\_to\_all.js, обрабатывает действия пользователя, такие как добавление товаров в корзину и удаление их из нее, а также обновление состояния корзины в реальном времени.

4.3. Управление запасами товаров

Записи о товарах надо где-то хранить, для этого существуют базы данных. На этом проекте мы использовали SQL и No-SQL базы данных. Для хранения структурированных данных, таких как записи о пользователях, записи о товарах, записи о заказах, подойдет SQL база данных. На данном проекте мы использовали PostgreSQL. В базе данных товары хранятся с такой структурой: id, title, description, quantity, price, created\_at, is\_featured, gender\_category, item\_type\_id, image\_filename, status. Когда пользователь делает заказ, идет проверка количества товаров. Если количество товаров равно 0, то при заказе пользователь получит ошибку. Все функции для управления запасами товаров находятся в “app/crud.py”.

Фронтенд часть управления запасами товаров реализована через различные HTML-шаблоны, которые отображают информацию о товарах. Например, в шаблоне add\_order.html пользователю предоставляется возможность указать адрес доставки и количество предметов для заказа. JavaScript код в файле add\_order\_to\_db.js обрабатывает отправку данных формы на сервер, а также обновляет информацию о цене в зависимости от выбранного количества товара. Это позволяет пользователю видеть актуальную стоимость заказа перед его подтверждением.

4.4. Управление заказами и картами

Когда пользователь хочет сделать заказ на эндпоинте “/add-order/”, то сначала проверяется наличие платежной карты. Если карты нет, то пользователя перенаправляет на эндпоинт “/add-card/”, где он может ввести данные своей карты для дальнейших платежей. Этот процесс включает в себя валидацию введенных данных, чтобы убедиться, что информация о карте корректна и соответствует требованиям платежного сервиса. После успешного добавления карты пользователь получает уведомление о том, что карта успешно сохранена и теперь доступна для использования при оформлении заказов.

Фронтенд часть для управления картами представлена в шаблоне add\_card.html, где пользователю предлагается ввести данные своей карты. JavaScript код в файле add\_card.js обрабатывает отправку данных формы и отображает уведомление о результате операции. Это позволяет пользователю быстро понять, была ли карта успешно добавлена или произошла ошибка.

Если карта есть, то пользователь может сделать заказ из корзины: заказать 1 товар или все. Если пользователь выбирает заказать 1 товар, то его перебрасывает на эндпоинт “/add-order/”, где ему нужно указать адрес доставки и количество предметов. На этом этапе система также проверяет наличие товара на складе и его доступность для заказа. Если выбранное количество превышает доступное, пользователю будет предложено изменить количество или выбрать другой товар.

Если пользователь выбирает заказать все товары из корзины, то его переадресовывает на эндпоинт “/add-order/all”, где ему показывается предупреждение, что все предметы будут куплены с учетом их количества в корзине. Перед подтверждением заказа система выводит сводную информацию о заказе, включая список товаров, их количество, общую стоимость и адрес доставки. Пользователь должен подтвердить заказ, прежде чем он будет отправлен на обработку.

После подтверждения заказа система создает запись о заказе в базе данных и инициирует процесс обработки платежа через интегрированный платежный сервис. В случае успешного завершения платежа пользователю отправляется уведомление о статусе заказа, а также информация о предполагаемой дате доставки. Фронтенд часть для оформления заказа представлена в шаблоне add\_order\_all.html, где пользователю показывается сводная информация о заказе и возможность его подтверждения. JavaScript код в файле all\_order.js обрабатывает отправку формы и отображает уведомление о результате оформления заказа.

Таким образом, разработка фронтенд части веб-приложения включает в себя создание интуитивно понятных интерфейсов для пользователей, которые позволяют легко взаимодействовать с системой аутентификации, корзиной, управлением запасами и оформлением заказов. Использование JavaScript для обработки событий и динамического обновления контента значительно улучшает пользовательский опыт и делает приложение более отзывчивым.

4.5. Интеграция с платежным сервисом

Для примера интеграции с платежным сервисом был выбран платежный сервис ecommpay. Ecommpay предлагает широкий спектр возможностей для обработки платежей, включая поддержку различных методов оплаты, таких как кредитные карты, электронные кошельки и банковские переводы. Платежный сервис также обеспечивает высокий уровень безопасности транзакций, что является критически важным для защиты данных пользователей. Кроме того, ecommpay предоставляет инструменты для управления возвратами, отчетности и аналитики, что позволяет бизнесам эффективно отслеживать свои финансовые операции.

Для интеграции с ecommpay в Python есть библиотека ecommpay-sdk, которая упрощает взаимодействие с API платежного сервиса. Эта библиотека предоставляет разработчикам удобные методы для выполнения различных операций, таких как создание платежей, проверка статуса транзакций и обработка уведомлений о платежах. Поскольку этот проект является лишь имитацией магазина, никакие деньги не списываются, но система интеграции с платежным сервисом предусмотрена. Когда пользователь оформляет заказ, то при успешном выполнении данные о заказе отправляются в платежный сервис, что позволяет протестировать весь процесс без реальных финансовых операций. Это делает интеграцию с ecommpay удобной и безопасной для разработки и тестирования.

4.6. Применение Jinja2

В данном проекте Jinja2 используется для наследования всех .html файлов от базового index.html файла, а также он участвует в прорисовке index.html, add\_order.html, cart.html, edit\_request.html и search\_results.html файлов без использования JavaScript. Я использовал Jinja2 для создания своих .html файлов: cart.html, add\_order.html, index.html с 127 по 136 страницу и другие мелкие изменения, но остальная логика этих файлов работает с использованием JavaScript.

Jinja2 позволяет эффективно управлять структурой и содержимым страниц, обеспечивая возможность динамического отображения данных. Например, в index.html можно использовать блоки для отображения различных секций, таких как заголовок, меню и основной контент, что упрощает процесс обновления и изменения дизайна.

В add\_order.html и cart.html Jinja2 используется для отображения форм и списков товаров, позволяя пользователю взаимодействовать с приложением без необходимости перезагрузки страницы. Данные, передаваемые из серверной части, могут быть легко интегрированы в шаблоны, что делает процесс рендеринга более гибким и удобным.

Кроме того, в edit\_request.html и search\_results.html Jinja2 помогает отображать результаты поиска и редактирования запросов, обеспечивая пользователю интуитивно понятный интерфейс. Использование Jinja2 в этих файлах позволяет избежать дублирования кода и улучшает читаемость, так как общие элементы интерфейса могут быть вынесены в базовый шаблон.

Таким образом, Jinja2 не только упрощает процесс разработки, но и улучшает пользовательский опыт, позволяя создавать более интерактивные и динамичные веб-приложения.

# Заключение

В ходе разработки веб-приложения с использованием Python и FastAPI были получены важные теоретические и практические выводы. FastAPI продемонстрировало высокую производительность и простоту в использовании, что позволило значительно сократить время на разработку и тестирование приложения. Также было выявлено, что использование асинхронного программирования в FastAPI позволяет эффективно обрабатывать множество запросов одновременно, что является критически важным для современных веб-приложений.

Оценка проведённого исследования показала, что разработанное веб-приложение успешно выполняет поставленные задачи. Результаты тестирования подтвердили его стабильность и высокую скорость обработки запросов. Пользовательский интерфейс оказался интуитивно понятным, что положительно сказалось на пользовательском опыте.

Практическая значимость работы заключается в создании функционального веб-приложения, которое может быть использовано в реальных условиях. Рекомендации по дальнейшему развитию включают добавление новых функций, таких как интеграция с внешними API и улучшение системы аутентификации пользователей. Планы на дальнейшие исследования могут включать изучение возможностей масштабирования приложения и его адаптации для работы с большими объемами данных.

Общий итог работы подтверждает достижение поставленной цели — создание эффективного и производительного веб-приложения. Все задачи, поставленные в начале проекта, были успешно выполнены, а гипотеза о том, что использование FastAPI значительно ускоряет процесс разработки, была доказана.

В качестве предложений по совершенствованию объекта исследования можно выделить необходимость внедрения более сложных алгоритмов обработки данных и улучшения системы мониторинга производительности приложения. Также стоит рассмотреть возможность использования контейнеризации для упрощения развертывания и управления приложением в различных средах.

Кроме того, для повышения безопасности проекта рекомендуется внедрить следующие меры:

* Использование HTTPS: Шифрование данных, передаваемых между клиентом и сервером, поможет защитить информацию от перехвата.
* Защита от CSRF: Реализация токенов для защиты от межсайтовой подделки запросов (CSRF) обеспечит дополнительный уровень безопасности для форм, отправляемых пользователями.
* Ограничение доступа: Реализация механизма контроля доступа на основе ролей (RBAC) позволит ограничить доступ к определённым ресурсам в зависимости от прав пользователя.
* Регулярные обновления: Поддержка актуальности зависимостей и библиотек, используемых в проекте, поможет избежать уязвимостей, связанных с устаревшими версиями.
* Логирование и мониторинг: Внедрение системы логирования и мониторинга поможет отслеживать подозрительную активность и быстро реагировать на инциденты безопасности.

Эти меры помогут значительно повысить уровень безопасности веб-приложения и защитить его от потенциальных угроз.

* Любанович Билл. FastAPI: Веб-разработка на Python. Неизвестно: Sprint Book, 2024: 288.
* Снайдер Джон. SQL для начинающих: Полное руководство. Москва: БХВ-Петербург, 2021: 320.
* Любанович Билл. FastAPI: Веб-разработка на Python. Неизвестно: Sprint Book, 2024: 288.
* Мартин Роберт. Чистый код: Создание, анализ и рефакторинг. Санкт-Петербург: Питер, 2020: 464.
* Гуджон Дэвид. Python для анализа данных. Москва: Вильямс, 2019: 400.
* Кастро Мигель. Redis в действии. Москва: БХВ-Петербург, 2020: 352.
* Грэм Дэниел. Python и SQL: Полное руководство по работе с базами данных. Москва: ДМК Пресс, 2022: 280.
* Каплан Игорь. Redis: Быстрые и эффективные базы данных. Санкт-Петербург: Питер, 2021: 240.
* Митчелл Уэсли. Python для веб-разработки: от Flask до FastAPI. Москва: Вильямс, 2023: 320.
* Кормен Томас. Алгоритмы: построение и анализ. Санкт-Петербург: Питер, 2019: 992.
* Сантош С. SQL: Полное руководство. Москва: Вильямс, 2021: 480.
* Любанович Билл. FastAPI: Разработка RESTful API на Python. Неизвестно: Sprint Book, 2023: 256.
* Зелинский Алексей. Python для анализа данных: Научные вычисления и визуализация. Москва: БХВ-Петербург, 2022: 368.
* Каплан Игорь. Redis: Основы и продвинутые техники. Москва: ДМК Пресс, 2021: 300.
* Мартин Роберт. Совершенный код: Как писать программы, которые легко читать и поддерживать. Санкт-Петербург: Питер, 2020: 480.
* Смит Джон. SQL для профессионалов: Оптимизация и производительность. Москва: Вильямс, 2022: 400.
* Курт М. Python и Redis: Эффективные решения для хранения данных. Москва: БХВ-Петербург, 2023: 320.

# Приложения

Docker-compose.yaml

version: '3'

services:

redis:

image: redis:alpine

ports:

- "6379:6379"

volumes:

- ./data:/data

db:

image: postgres:latest

environment:

- POSTGRES\_USER=postgres

- POSTGRES\_PASSWORD=postgres

- POSTGRES\_DB=postgres

- POSTGRES\_HOST=localhost

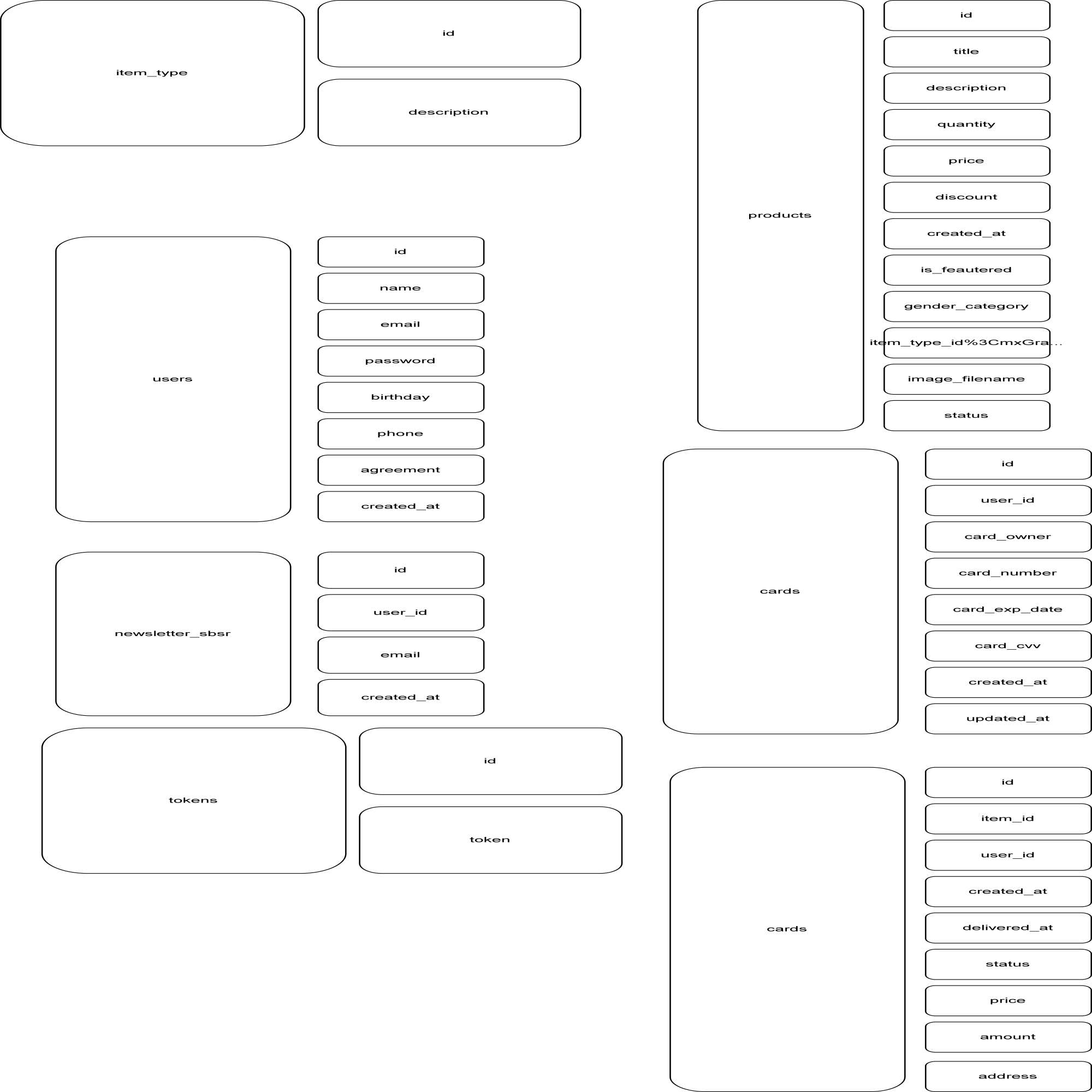
- POSTGRES\_PORT=5432

volumes:

- ./pgdata:/var/lib/postgresql/data

ports:

- "5432:5432"



Структура моделей в базе данных

GitHub: <https://github.com/AndreyKhokhlovGeekBrains/GraduationProject>