Реферат

РПЗ 44 с., 17 рис., 3 табл., 9 ист.

Данная работа отражает разработку ПО для интернет-магазина парфюмерии.

Формализовано задание, определен необходимый функционал. Проведен анализ типов СУБД по модели данных и выбрана наиболее подходящая модель. Описана структура базы данных, определены роли пользователей и их права. Создана и заполнена БД в соответствии с аналитическим и конструкторскими разделами. Реализованы необходимые для работы приложения функции и триггеры. Реализован web-интерфейс для доступа к БД и проведения с ней манипуляций. Разработано программное обеспечение, реализующую поставленную задачу. Проведено сравнение времени отклика БД с индексом и без него.

КЛЮЧЕВЫЕ СЛОВА

База данных, Интернет-магазин, Парфюмерия, PostgreSQL, Django, Django-ORM, Python

Содержание

Введение

Последние годы на рынке товаров народного потребления наблюдается довольно устойчивая тенденция снижения продаж в розничных магазинах, в том числе сетевых, при одновременном росте продаж товаров и услуг через интернет. Интернет-магазины предоставляют возможность покупателям совершать покупки в любое время суток и из любой точки мира. Интернет-магазины парфюмерии становятся все более популярными, так как клиенты имеют возможность получить доступ к широкому ассортименту товаров и выбрать наиболее подходящие ароматы без необходимости посещать физический магазин.

Существует множество факторов необходимости хорошей базы данных для интернет магазина, вот некоторые из них:

- Хранение и организация информации: База данных позволяет эффективно хранить и организовывать информацию о продуктах, клиентах, заказах, платежах и других аспектах работы интернет-магазина. Это обеспечивает систематизацию данных и упрощает доступ к соответствующим данным при необходимости;

- Управление инвентарем: База данных позволяет отслеживать наличие товаров, их количество, стоимость и другую информацию, связанную с продуктами, доступными для продажи. Это помогает управлять запасами и предотвращать ситуации, когда товары заканчиваются на складе или есть остатки, которые необходимо реализовать;

- Обработка заказов: База данных позволяет хранить информацию о заказах, включая детали заказа, данные клиента, способы оплаты и доставки. Это обеспечивает возможность эффективной обработки заказов, отслеживания их статуса, связи с клиентами и обеспечения актуальной информации о выполнении заказов;

- Персонализация и удобство для клиентов: База данных позволяет хранить информацию о предпочтениях клиентов, истории покупок, предыдущих заказах и других персональных данных. Это помогает создать более персонализированный опыт покупателя, предлагая ему рекомен-

дации товаров, специальные предложения или скидки в соответствии с его предпочтениями.

– Аналитика и прогнозирование: База данных позволяет проводить анализ данных и выявлять тенденции, например, наиболее популярные товары, поведение клиентов, эффективность маркетинговых кампаний и другие факторы. Эта информация может быть использована для принятия решений по улучшению процессов продажи, разработке маркетинговых стратегий и прогнозирования спроса на товары.

– Безопасность данных: База данных позволяет обеспечить безопасность хранения и обработки информации о клиентах, включая персональные данные, платежные данные и другие конфиденциальные сведения. Важно убедиться, что база данных обеспечивает защиту от несанкционированного доступа, взломов и утечек данных, чтобы сохранить доверие клиентов и соблюдать соответствующие правовые и организационные требования.

Цель данной курсовой работы — Разработка базы данных для хранения и анализа информации интернет-магазина парфюмерии.

Для достижения поставленной цели необходимо решить следующие задачи:

– формализовать задание, определить необходимый функционал;

– провести анализ СУБД;

– описать структуру базы данных;

– спроектировать приложение для доступа к БД;

– создать и заполнить БД;

– реализовать интерфейс для доступа к БД;

– разработать программное обеспечение, реализующую поставленную задачу.

1 Аналитический раздел

В данном разделе будет проанализирована поставленная задача, определён необходимый функционал програмного обеспечения, описаны роли пользователей программы и будет произведён анализ существующих моделей базы данных.

1.1 Постановка задачи

Необходимо разработать программу для отображения, манипуляции и эффективного хранения информации интернет-магазина парфюмерии. Покупатель должен иметь возможность просмотра каталога товаров, оформления зазаза выбранной продукции, написания отзывов к товарам и постановки оценок товарам и отзывам других покупетелей. Необходимо предусмотреть наличие ролей менеджера и администратора, осуществляющих модерацию и управление продукцией (добавление, изменение, удаление товаров), заказами(просмотр товаров, при необходимости корректировка заказа), модерации некорректных отзывов и регулярующих деятельность покупателей, в том числе доступ к профилям покупателей, при необходимости внесение корректировок или удаление профиля.

1.2 Формализация данных

Таблица 1.1 – Данные и сведения о них

Данные	**Сведения**
Пользователь	ФИО, никнейм, дата рождения, адрес, e-mail, пароль, права доступа, пол, аватар
Товар	Заголовок, описание, фото, категории, количество, стоимость, дата публикации, объем флакона, рейтинг
Заказ	Товары, дата оформления, дата исполнения, стоимость, статус, комментарий
Отзыв	Содержание, дата, пользователь, рейтинг, дата публикации
Оценка	Дата, значение, пользователь

1.3 Типы пользователей

Таблица 1.2 – Типы пользователей и их функционал

Тип пользователя	**Функционал**
Покупатель	Просмотр каталога продукции и информации о товарах, написание отзывов и постановка оценок товарам и комментариям
Менеджер	Просмотр каталога продукции и информации о товарах Просмотр информации покупателей, просмотр информации о заказах, подтверждение и отклонение заказов, обновление продукции и информации о ней, при необходимости модерация комментариев пользователей
Администратор	Просмотр каталога продукции и информации о товарах Просмотр информации покупателей, просмотр информации о заказах, подтверждение и отклонение заказов, обновление продукции и информации о ней, при необходимости модерация комментариев пользователей Добавление и удаление менеджеров и возможность изменение информации пользователей, либо их удаление

На рисунке 1.2 представлена Use-Case-диаграмма.

1.4 ER-диаграмма сущностей

На рисунке 1.1 представлена ER-диаграмма сущностей проекта.

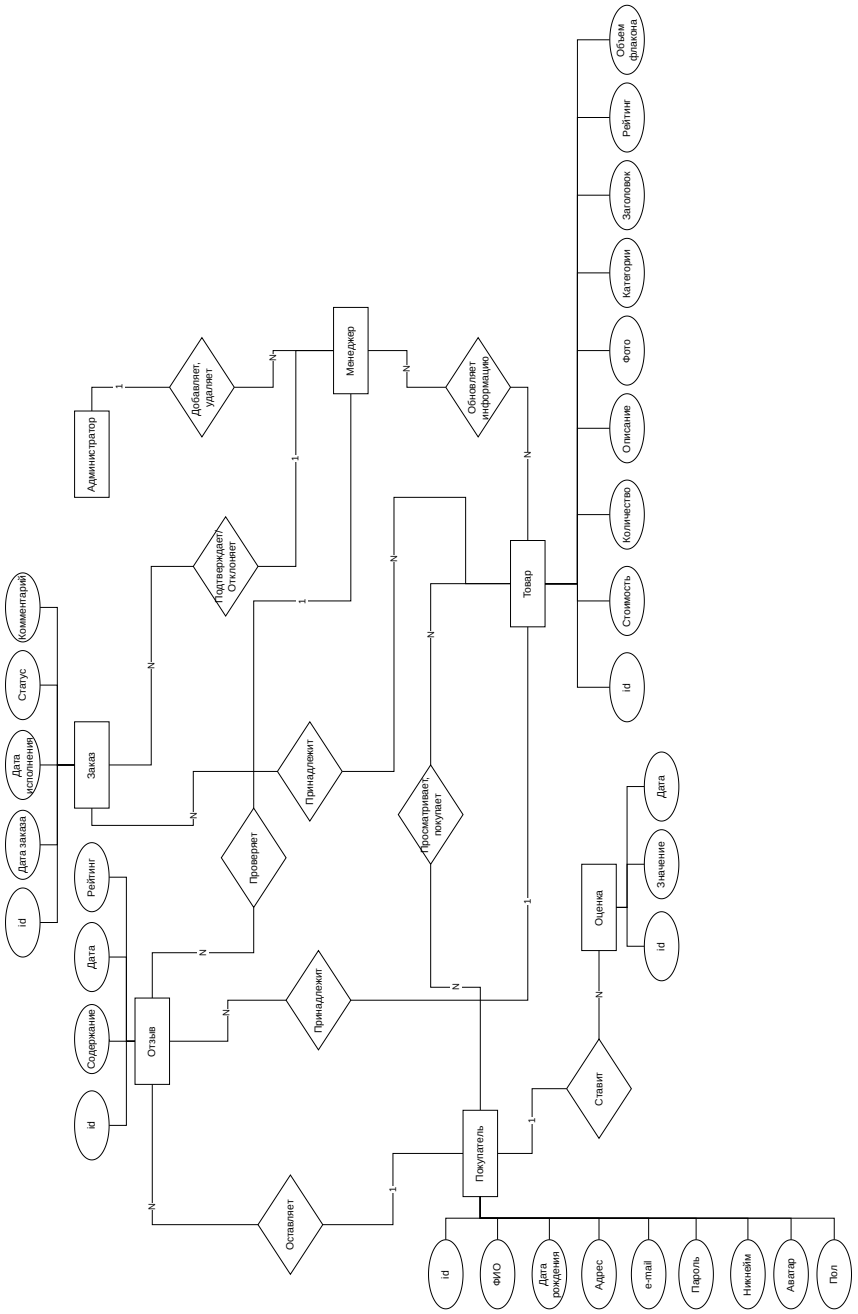

Рисунок 1.1 – ER-диаграмма сущностей

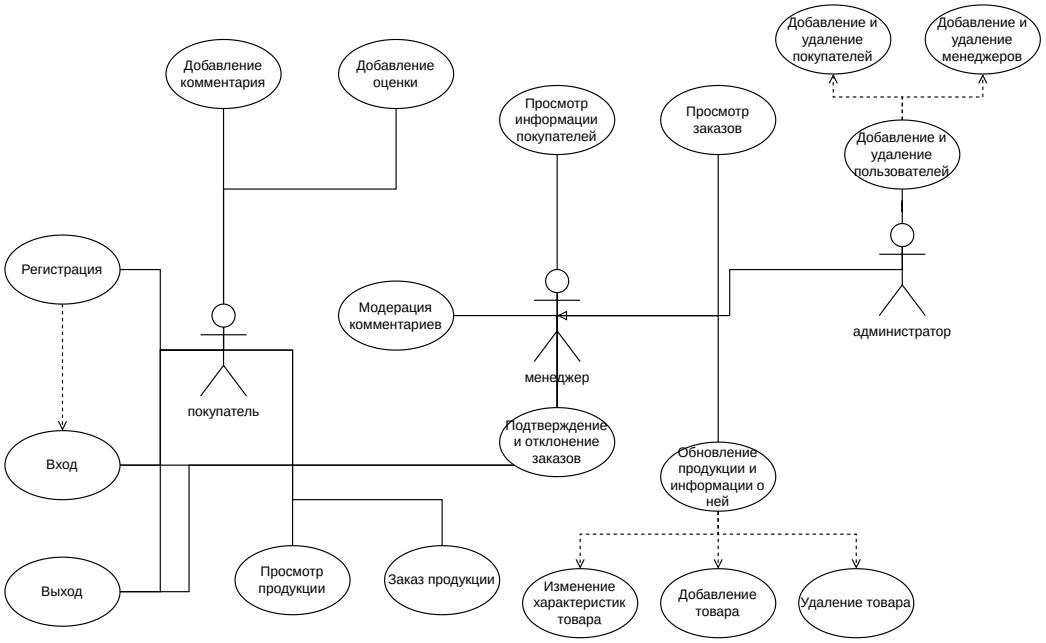

Рисунок 1.2 – Use-Case диаграмма

1.5 Описание существующих СУБД

Система управления базами данных — это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования БД многими пользователями [1].

1.5.1 Основные функции СУБД

Основынми функциями СУБД являются:

– создание, изменение и удаление структуры базы данных (таблиц, индексов, представлений и других объектов);

– организация и оптимизация хранения данных;

– реализация механизмов контроля целостности и согласованности данных (транзакции, ограничения, триггеры);

– обеспечение многопользовательского доступа к базе данных;

– мониторинг и управление производительностью базы данных;

– журнализация изменений, резервное копирование и восстановление базы данных после сбоев;

– поддержка языков БД.

1.5.2 Классификация СУБД по модели данных

Модель данных — это абстрактное, самодостаточное, логическое определение объектов, операторов и прочих элементов, в совокупности составляющих абстрактную машину доступа к данным, с которой взаимодействует пользователь. Эти объекты позволяют моделировать структуру данных, а операторы — поведение данных [2].

Основные типы моделей организации данных:

– иерархическая

  Модель данных, в основе которой лежит иерархическая структура типа дерева. Дерево — это орграф, в каждую вершину которого кроме первой (корневой), входит только одна дуга, а из любой вершины (кроме конечных) может исходить произвольное число дуг. В иерархической структуре подчиненный элемент данных всегда связан только с одним исходным [3]. Но при том, что у каждого дочернего элемента может быть только один родительский, у родительского может быть несколько дочерних.

  Преимущества иерархической модели данных:

  – простая организация данных отражает их естественную иерархическую структуру;

  – высокая производительность выполнения запросов, связанных с навигацией по иерархии;

  Недостатки:

  – ограничение на количество связей между узлами. Множественные связи между двумя узлами не предусмотрены;

  - сложность обновления данных, особенно в случае изменения структуры иерархии;
  - отсутствие гибкости, так как система сильно зависит от текущей модели данных;

- сетевая

  Основана на представлении информации в виде орграфа, в котором в каждую вершину может входить произвольное число дуг. Вершинам графа сопоставлены типы записей, дугам — связи между ними [3]. В отличие от иерархической модели, где каждая запись может иметь только одного родителя, в сетевой модели запись может иметь несколько родительских записей и несколько дочерних записей.

  Преимущества сетевой модели данных:

  - более гибкая структура данных по сравнению с иерархической моделью. Записи могут иметь несколько родителей, что позволяет представить более сложные связи между данными;
  - высокая производительность при доступе к связанным данным, поскольку нет необходимости проходить через все уровни иерархии;

  Недостатки:

  - сложность при проектировании и поддержке структуры данных, особенно при изменениях;
  - зависимость от физической структуры данных и последовательности прочтения записей;
  - ограниченная поддержка сложных запросов и агрегатных операций;

- графовая

  Графовая модель данных основана на математической теории графов. В этой модели данные представляются в виде набора вершин (узлов) и связей (ребер), которые соединяют эти вершины. Каждая вершина может иметь атрибуты, которые хранят информацию о ней. Связи также имеют атрибуты и определяют отношения между вершинами. Главная

особенность графовой модели данных состоит в том, что она позволяет представлять сложные связи и отношения между данными

Преимущества графовой модели данных:

- гибкая структура данных, способствующая моделированию сложных отношений;
- эффективность при выполнении запросов, особенно связанных с обходом и анализом графовых структур;
- наглядность и понятность представления данных;

Недостатки:

- нет единого языка запросов для графовых баз данных;
- затраты на хранение и обработку связей между вершинами, особенно в случае больших графовых структур;

– реляционная

В реляционной модели для отображения информации о предметной области используется таблица, называемая отношением. Строка такой таблицы называется кортежем, столбец — атрибутом. Каждый атрибут может принимать некоторое подмножество значений из определенной области — домена.

Табличная организация БД позволяет реализовать ее важнейшее преимущество перед другими моделями данных, а именно возможность использования точных математических методов манипулирования данными, и, прежде всего, аппарата реляционной алгебры и исчисления отношений.

К другим достоинствам реляционной модели можно отнести наглядность, простоту изменения данных и организации разграничения доступа к ним [3].

Вывод

Поскольку реляционная модель базы данных является наиболее широко используемой и удобной, имеет возможность изменения базы данных без глобальных изменений программного обеспечения, а также обеспечивает поддержку ограничений, таких как ограничения целостности, что позволяет гарантировать, что данные, хранящиеся в базе данных, всегда являются корректными, то для реализации данного проекта была выброана именно она.

В данном разделе была произведена формализация поставленной задачи, описан необходимый функционал с распределением по ролям, проанализорованы модели базы данных и выбрана реляционная модель.

2 Конструкторский раздел

В этом разделе будет проведено проектирование базы данных и приложения. Будут приведены созданные таблицы, поля таблиц с указанием их семантического смысла. Будет представлена ER-модель созданной базы данных.Будет выбран тип приложения. Будут представлены схемы основных алгоритмов, необходимых для функционирования приложения.

2.1 Проектирование базы данных

База данных приложения будет реализована с помощью следующих таблиц:

a) таблица продуктов Product

b) таблица заказов Order

c) таблица отзывов Review

d) таблица оценок продуктов Likeproduct

e) таблица оценок отзывов Likereview

f) таблица категорий продуктов Category

g) таблица профилей Profile

h) таблица пользователей User

Таблица **User** содержит информацию о пользователях приложения. Содержит следующие поля:

- id — первичный ключ, идентификатор пользователя;
- password — пароль пользователя;
- username — никнейм пользователя(логин);
- first_name — имя пользователя;

– last_name — фамилия пользователя;

– email — электронная почта пользователя;

– is_superuser — логическое поле, обозначающее есть ли у пользователя все разрешения без их явного назначения;

– is_staff — логическое поле, обозначающее есть ли у пользователя доступ к сайту администратора;

– is_active — логическое поле, обозначающее активность записи;

– data_joined — дата и время создания учетной записи;

– last_login — дата и время последнего входа пользователя в систему.

Таблица **Profile** содержит информацию о профилях пользователей. Содержит следующие поля:

– id — первичный ключ, идентификатор профиля;

– user_id — внешний ключ, соответствует id пользователя приложения;

– birth_date — дата рождения;

– address — адрес пользователя;

– sex — пол пользователя;

– avatar — путь до картинки с аватаром пользователя.

Таблица **Product** содержит информацию о товарах, доступных на сайте. Содержит следующие поля:

– id — первичный ключ, идентификатор товара;

– title — название товара;

– content — описание товара;

– count — количество товара;

– cost — цена за единицу товара;

– image_path — путь до изображения товара;

– pub_date — дата публикации товара;

– rating — оценка товара;

– volume — объем флакона.

Таблица **Order** содержит информацию о заказах пользователей. Содержит следующие поля:

– id — первичный ключ, идентификатор заказа;

– status — статус заказа;

– comment — комментарий к заказу;

– order_date — дата создания заказа;

– date_of_completion — дата выполнения заказа;

– profile_id — внешний ключ, соответствует id профиля заказчика.

Таблица **Review** содержит информацию об отзывах на товары. Содержит следующие поля:

– id — первичный ключ, идентификатор отзыва;

– content — описание товара;

– review_date — дата публикации отзыва;

– profile_id — внешний ключ, соответствует id профиля, оставившего отзыв;

– product_id — внешний ключ, соответствует id товара, на который оставили отзыв.

Таблица **Likeproduct** содержит информацию об оценках на товары. Содержит следующие поля:

– id — первичный ключ, идентификатор оценки на товар;

– mark — оценка, оставленная пользователем;

– profile_id — внешний ключ, соответствует id профиля, оставившего оценку;

– product_id — внешний ключ, соответствует id товара, на который оставили оценку.

Таблица **Likereview** содержит информацию об оценках на отзывы. Содержит следующие поля:

– id — первичный ключ, идентификатор оценки на отзыв;

– mark — оценка, оставленная пользователем;

– profile_id — внешний ключ, соответствует id профиля, оставившего оценку;

– review_id — внешний ключ, соответствует id отзыва, на который оставили оценку.

Таблица **Category** содержит информацию о категориях товаров. Содержит следующие поля:

– id — первичный ключ, идентификатор категории;

– name — название категории.

Таблица **OrdersProducts** содержит информацию о связи между заказами и товарами, а также хранит количество данного товара в заказе. Содержит следующие поля:

– id — первичный ключ, идентификатор категории;

– order_id — внешний ключ, соответствует id заказа;

– product_id — внешний ключ, соответствует id продукта;

– cnt — количество товара в заказе.

На рисунке 2.1 приведена инфологическая модель базы данных.

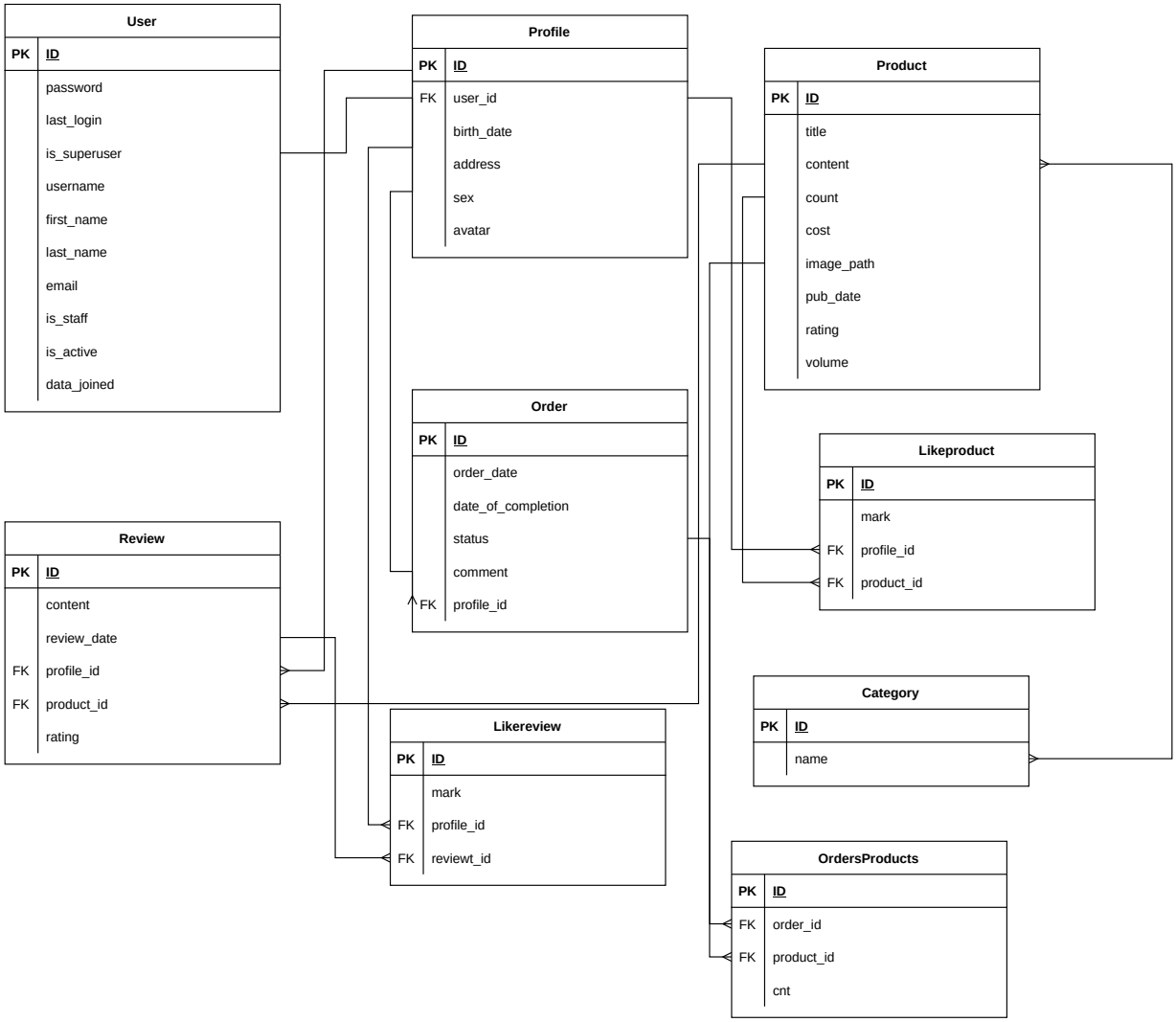

Рисунок 2.1 – Инфологическая модель базы данных

2.2 Схемы триггеров

На рисунке 2.2 приведена схема триггера, уменьшающего количество товара при заказе на количество единиц, заказанного товара.

Рисунок 2.2 – Схема триггера decrease_count

На рисунке 2.3 приведена схема триггера, изменяющего рейтинг товара при добавлении оценки на товар пользователем.

Рисунок 2.3 – Схема триггера like_product_insert

На рисунке 2.4 приведена схема триггера, изменяющего рейтинг товара на удвоенную оценку при изменеии оценки товара пользователем.

Рисунок 2.4 – Схема триггера like_product_update

3 Технологический раздел

В данном разделе приведены средства реализации, структура программного обеспечения, листинги кода и интерфейс ПО.

3.1 Средства реализации ПО

3.1.1 Выбор языка программирования

Для реализации ПО был выбран язык программирования python [4]. Данный выбор обусловлен следующими причинами:

– имеющийся опыт разработки на данном языке;

– python поддерживает объектно-ориентированный подход;

– обширный функционал языка, а также огромное количество библиотек и фреймворков, которые отлично подходят для быстрой разработки web-приложений;

– python имеет одно из самых больших и активных сообществ разработчиков, по нему есть большое количество литературы и информации.

3.1.2 Выбор СУБД

В роли СУБД было решено использовать PostgreSQL [5].

PostgreSQL [5] — СУБД с открытым исходным кодом, основой которого был код, написанный в Беркли. Она поддерживает большую часть стандарта SQL и предлагает множество современных функций:

– сложные запросы;

– поддержка многопоточности и параллелизма;

– внешние ключи;

– триггеры;

– изменяемые представления;

– транзакционная целостность;

– многоверсионность.

– поддержка восстановления после сбоев;

– возможность масштабируемости и репликации для дальнейшего роста интернет-магазина;

– является системой с открытым исходным кодом, что означает наличие большого сообщества разработчиков и пользователей, активно поддерживающих и обновляющих его.

Данный выбор обусловлен описанным выше функционалом, удовлетворяющим всем требованиям данного проекта, опытом разработки ПО с использованием данной СУБД и ее совместимостью с python.

3.1.3 Выбор платформы для реализации ПО

Для реализации ПО был выбран веб-фреймворк Django [6]. Django — высокоуровневый веб-фреймворк, способствующий быстрой разработке и чистому, прагматичному дизайну.

Выбор данного фреймворка обусловлен такими его достоинствами, как:

– автоматическая генерация административного интерфейса: Django автоматически создает административный интерфейс для управления данными в базе данных. Это упрощает администрирование и позволяет быстро создать функциональную панель управления;

– ORM: Django предоставляет ORM, который позволяет работать с базой данных через объектно-ориентированный интерфейс. Это позволяет разрабатывать приложения, не требуя написания SQL-запросов вручную, что сокращает время разработки и упрощает обслуживание кода;

– Встроенная аутентификация и авторизация: Django предоставляет функциональность для аутентификации пользователей и контроля доступа к различным частям приложения. Это помогает сделать проект безопасным и защищенным;

– Интеграция с другими технологиями: Django хорошо интегрируется с другими популярными технологиями и фреймворками, такими как Bootstrap jQuery, REST framework и другими. Это позволяет использовать широкий спектр инструментов при разработке проекта;

– Широкое сообщество: Django имеет большое и активное сообщество разработчиков. Существует множество документации, учебных пособий, видеоуроков и форумов, посвященных Django.

– Встроенная защита от внешних угроз: механизмы предотвращения SQL-инъекций и подделки межсайтовых запросов;

3.1.4 Выбор инструмента для работы с БД

В качестве инструмента для работы с БД был выбран Django-ORM, так как он ключен в базовый функционал фреймворка Django и обладает следующими достоинствами:

– позволяет определить модели, которые представляют таблицы в базе данных. Каждый атрибут модели соответствует полю в таблице, а методы предоставляют функциональность для работы с данными;

– позволяет автоматически создать схему базы данных на основе определенных моделей. Он обнаруживает изменения в моделях и создает или обновляет таблицы в соответствии с этими изменениями;

– предоставляет API для выполнения различных типов запросов к базе данных. Он поддерживает широкий спектр операций, таких как фильтрация, сортировка, агрегация и связи между моделями;

– использует концепцию ленивой загрузки данных, что означает, что запросы к базе данных выполняются только при необходимости. Это повышает производительность и уменьшает нагрузку на базу данных;

– поддерживает транзакции, что обеспечивает целостность данных при выполнении нескольких запросов. Он также предоставляет инструменты для создания и применения миграций, что позволяет изменять структуру базы данных без необходимости вручную редактировать схему.

Так как некоторые запросы к БД требуют вызовы функций базы, а Django-ORM, к сожалению, не поддерживает данный функционал, они были реализованы на чистом SQL. SQL-запросы отправлялись с помощью встроенного функционала Django, реализованного с использованием библиотеки **psycopg2**

3.1.5 Выбор средств отображения страниц браузера

Для отображения web-страниц в браузере был выбран HTML, CSS и JS фреймворк Bootstrap [7].

К плюсам Bootstrap можно отнести:

– кросс-браузерность;

– единство стилей элементов;

– ускорение верстки сайтов в сравнении с чистым CSS и JS.

В качестве среды разработки была выбрана среда Visual Studio Code.

3.2 Структура программного обеспечения

Структура проекта задается используемым фреймворком
Основные модули приложения:

– `views.py` — содержит функции для обработки запросов к приложению и отображению нужных веб-страниц;

– `models.py` — содержит классы, соответствующие таблицам базы данных для работы Django ORM и менеджеры, включающие функции запросов к данным таблицам;

– `forms.py` — содержит классы обработки и валидации форм;

– `urls.py` — содержит URL-адреса страниц приложения и соответствующие им обработчики;

– `cart.py` — содержит класс корзины товаров пользователя;

– `admin.py` — содержит параметры страницы администрирования приложения.

3.3 Листинги кода

3.3.1 Создание базы данных

В приложении была создана база данных, соответствующая описанию, приведенному в аналитическом и конструкторских отделах. Создание базы данных было реализованы с помощью Django ORM и механизмом миграций Django [6].

Код описания классов таблиц представлен в приложении А.

Также на поля таблиц были наложены ограничения, представленные в приложении В.

3.3.2 Функции

Для работы приложения было создано две функции:

– `popular_products()` — функция получения id товаров, отсортированных по популярности. Под популярностью подразумевается количество вхождений в заказы пользователей.

– `total_price(integer)` — функция расчета полной стоимости заказа. На вход функция получает id заказа, на выход возвращает полную стоимость с учетом количества каждого товара в заказе.

На листинге 3.1 представлена функция `popular_products()`.

Листинг 3.1 – Функция получения популярных товаров

```
CREATE OR REPLACE FUNCTION popular_products()
RETURNS TABLE (
    product int
) AS $$
BEGIN
    DROP TABLE IF EXISTS popular_products;

    CREATE TEMP TABLE popular_products(
        product int
    );

    INSERT INTO popular_products (product)
    WITH tmp AS (
    SELECT product_id, COUNT(app_orders_products.id) AS count_products
    FROM app_orders_products
    GROUP BY product_id
    ORDER BY count_products DESC)
    SELECT product_id FROM tmp;

    RETURN QUERY SELECT * FROM popular_products;
END;
$$ language PLPGSQL;
```

На листинге 3.2 представлена функция `total_price(integer)`.

Листинг 3.2 – Функция получения общей суммы заказа по его id

```
CREATE OR REPLACE FUNCTION total_price(integer)
RETURNS INT AS $$
BEGIN
    RETURN (WITH TMP(product_id, cost, cnt, total) AS
        (SELECT product_id, cost, cnt, cost*cnt AS total
        FROM app_order JOIN app_orders_products
        ON app_order.id=app_orders_products.order_id
        JOIN app_product ON app_product.id=app_orders_products.product_id
        WHERE app_order.id=$1)
    SELECT SUM(total) AS total_price FROM TMP);
END;
$$ language PLPGSQL;
```

3.3.3 Триггеры

При разработке ПО было создано 6 триггеров, позволяющих обновлять количество товара и рейтинг товаров и отзывов:

- `decrease_count()` — триггер, выполняющий уменьшения количества товара в наличии при заказе данного товара на количество заказанных единиц.
- `increase_count()` — триггер, выполняющий увеличения количества товара в наличии при удалении заказа с данным товаром на количество заказанных единиц.
- `like_product_insert()` — триггер, выполняющий изменение рейтинга товара на величину оценки при добавлении новой оценки данного товара пользователем.
- `like_product_update()` — триггер, выполняющий изменение рейтинга товара на величину удвоенной оценки при изменении пользователем оценки товара.
- `like_review_insert()` — триггер, выполняющий изменение рейтинга отзыва на величину оценки при добавлении новой оценки данного отзыва пользователем.

– `like_review_update()` — триггер, выполняющий изменение рейтинга отзыва на величину удвоенной оценки при изменении пользователем оценки товара.

На листинге 3.3 представлен триггер `decrease_count()`.

Листинг 3.3 – Триггер на уменьшение количества товаров

```
CREATE OR REPLACE FUNCTION decrease_count()
RETURNS TRIGGER
AS $decrease_count$
BEGIN
    UPDATE app_product
    SET count = count - NEW.cnt
    WHERE app_product.id = NEW.product_id;
    RETURN NEW;
END;
$decrease_count$ language PLPGSQL;

CREATE TRIGGER decrease_count AFTER INSERT ON app_orders_products
FOR EACH ROW EXECUTE PROCEDURE decrease_count();
```

На листинге 3.4 представлен триггер `like_product_insert()`.

Листинг 3.4 – Триггер на изменение рейтинга при добавлении оценки

```
CREATE OR REPLACE FUNCTION like_product_insert()
RETURNS TRIGGER
AS $like_product_insert$
BEGIN
    UPDATE app_product
    SET rating = rating + NEW.mark
    WHERE app_product.id = NEW.product_id;
    RETURN NEW;
END;
$like_product_insert$ language PLPGSQL;

CREATE TRIGGER like_product_insert AFTER INSERT ON app_likeproduct
FOR EACH ROW EXECUTE PROCEDURE like_product_insert();
```

На листинге 3.5 представлен триггер `like_product_update()`.

Листинг 3.5 – Триггер на обновление рейтинга при изменении оценки

```
CREATE OR REPLACE FUNCTION like_product_update()
RETURNS TRIGGER
AS $like_product_update$
BEGIN
    UPDATE app_product
    SET rating = rating + 2 * NEW.mark
    WHERE app_product.id = NEW.product_id;
    RETURN NEW;
END;
$like_product_update$ language PLPGSQL;

CREATE TRIGGER like_product_update AFTER UPDATE ON app_likeproduct
FOR EACH ROW EXECUTE PROCEDURE like_product_update();
```

3.3.4 Ролевая модель на уровне БД

В соответствии с типами пользователей, описанными в аналитическом отделе были созданы следующие роли на уровне БД:

a) Client — роль обычного пользователя. Роль Client предполагает доступ на добавление и изменение записей таблиц auth_user и app_profile, так как пользователь может осуществлять регистрацию. Пользователь может добавлять записи в таблицу app_review (комментирование товаров) и имеет права на добавление и изменение таблиц app_likeproduct и app_likereview, так как может оставлять оценки на товары и отзывы.

b) Manager — роль менеджера сайта. Менеджер выданы права на таблицы, связанные с товарами, заказами, отзывами, категориями и оценками, так как менеджер осуществляет модерацию содержимого сайта, следит за актуальностью информации товаров и изменяет информацию заказов.

c) Administrator — роль администратора сайта. Администратор сайта имеет права доступа на таблицы с товарами, заказами, отзывами, категориями и оценками, а также на таблицы auth_user и app_profile. Администратор имеет право на выдачу ролей другим пользователям.

Код создания ролей представлен в приложении В.

3.4 Демонстрация работы программы

3.4.1 Регистрация

На рисунке 3.1 представлена страница регистрации на сайте. Пользователь должен корректно заполнить все поля формы. Поля "username"и "email" должны быть уникальными, поля "password"и "repeat password"должны совпадать. Длина пароля составляет минимум 9 символов. Если поле "avatar" останется пустым, оно будет заполнено автоматически картинкой по умолчанию. Остальные поля обязательны для заполнения.

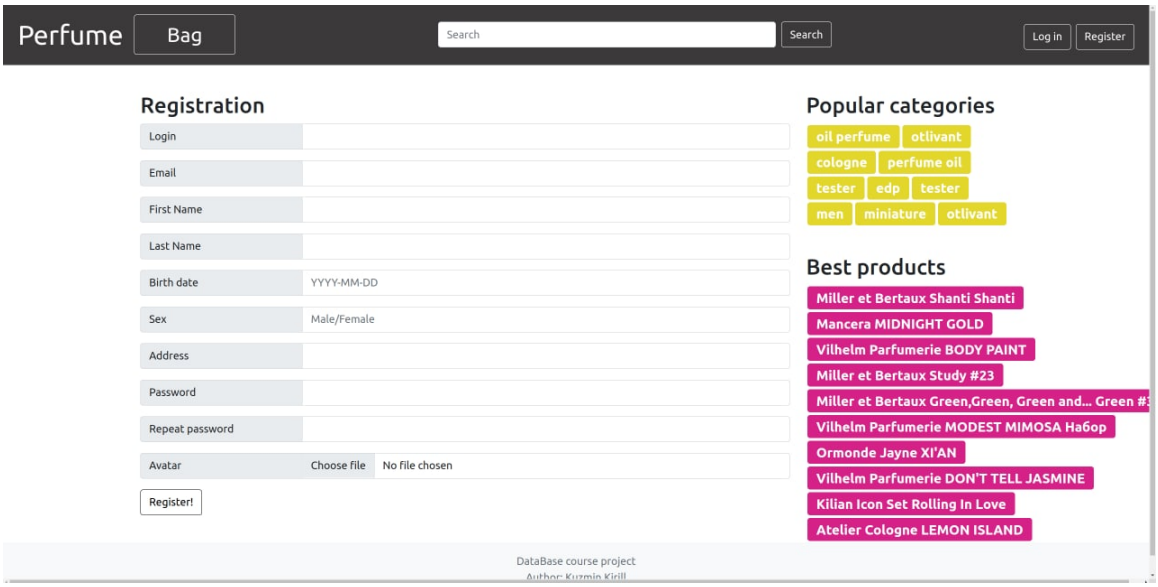

Рисунок 3.1 – Страница регистрации

3.4.2 Вход в систему

На рисунке 3.2 представлена страница входа в систему. Для авторизации необходимо ввести login и password пользователя, зарегистрированного в системе и нажать на кнопку "Login". При необходимости с этой страницы можно попасть на страницу регистрации, нажав кнопку "Create new account".

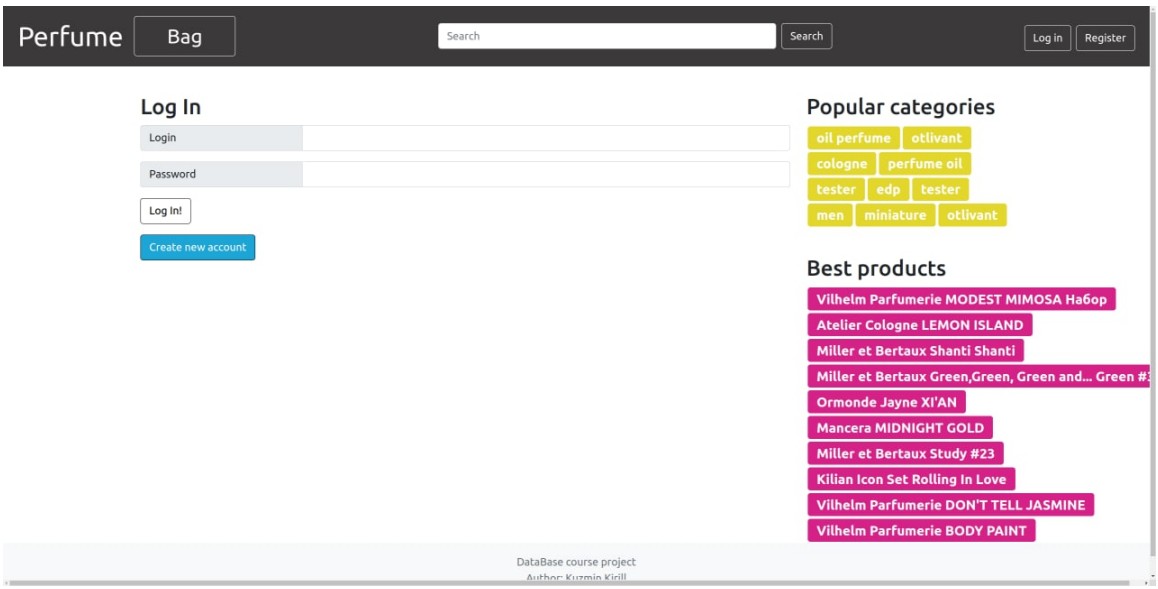

Рисунок 3.2 – Страница входа в систему

3.4.3 Изменение данных пользователя

На рисунке 3.3 представлена страница, позволяющая изменять данные пользователя. Для этого пользователь должен изменить соответсвующие поля. Поле "avatar"не обязательно для заполнения. После изменения информации следует нажать на кнопку "save"чтобы изменения вступили в силу.

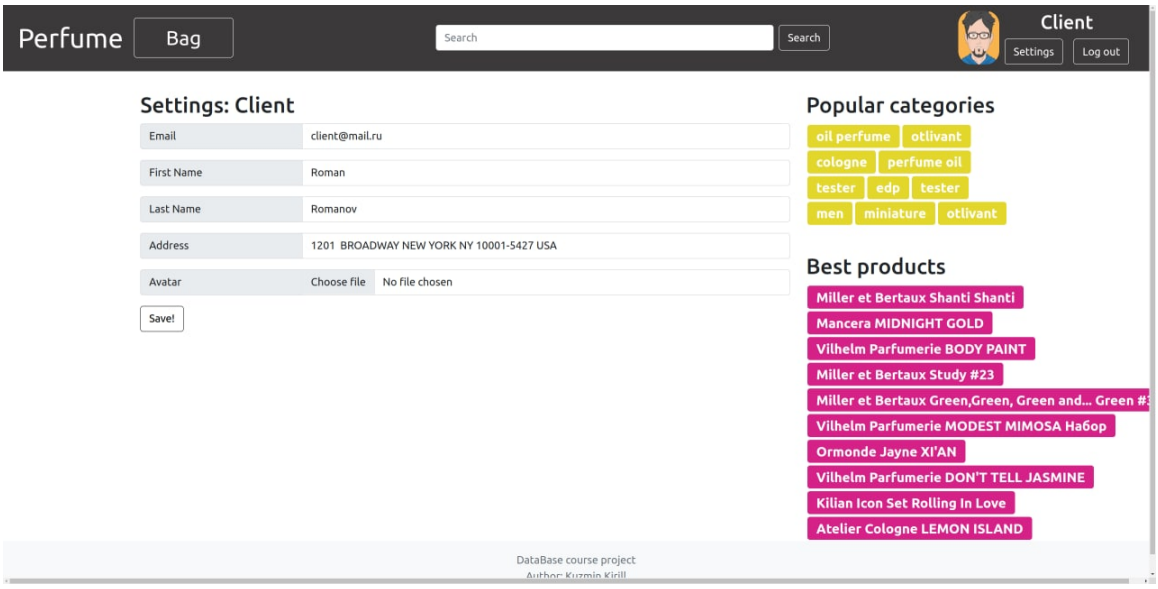

Рисунок 3.3 – Страница изменения данных пользователя

3.4.4 Поиск товаров

На рисунке 3.4 представлена страница с результатами поиска по строке "musk". Для выполнения поиска следует ввести поисковый запрос и нажать кнопку "Search". Поиск осуществляется по вхождению поискового запроса в название товара. Поиск не чувствителен к регистру.

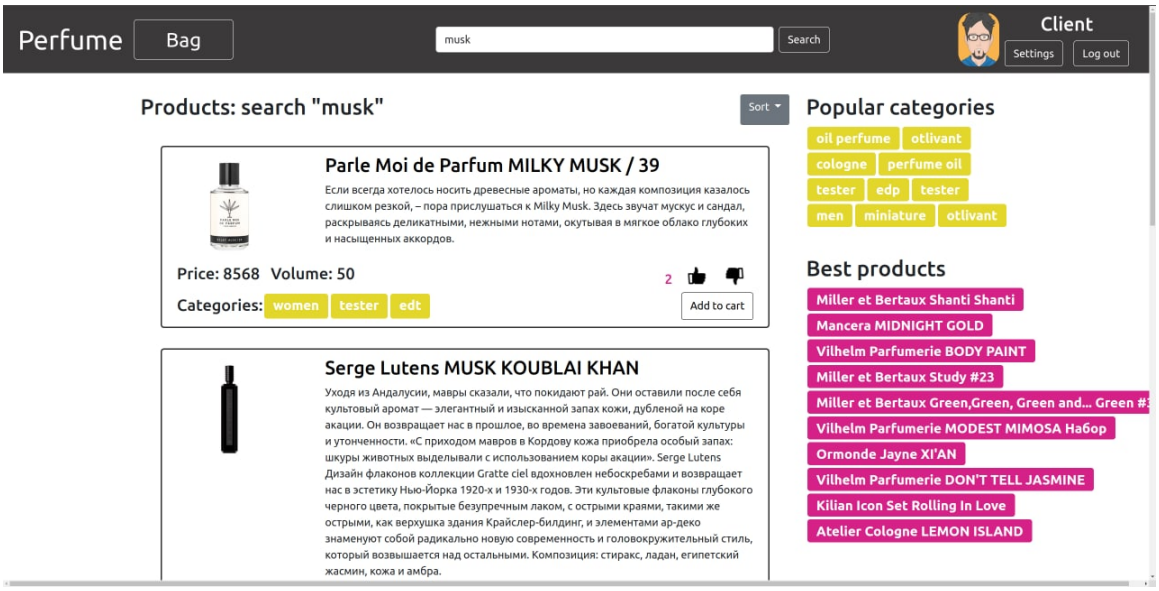

Рисунок 3.4 – Страница поиска товаров

3.4.5 Страница товара

На рисунке 3.5 представлена страница товара, на которой показано описание товара, цена, объем флакона, а также категории, к которым относится данный товар. Здесь можно поставить оценку товару в формате "нравится - не нравится". Чтобы добавить товар в корзину следует нажать кнопку "Add to cart". На странице товара представлены отзывы других пользователей на данный товар и оценки на отзывы, поставить которую можно так же, как и на товар.

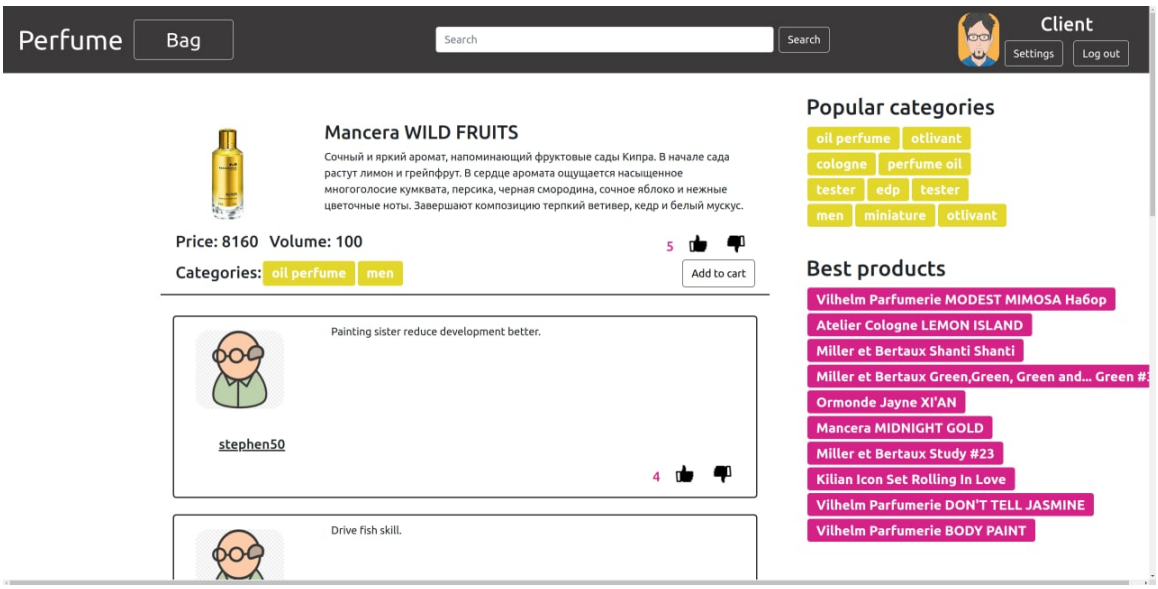

Рисунок 3.5 – Страница продукта

3.4.6 Создание отзыва

На рисунке 3.6 представлена страница создания отзыва. Для того, чтобы оставить отзыв пользователь должен войти в систему, зайти на страницу товара, заполнить форму Отзыва и нажать на кнопку "Review".

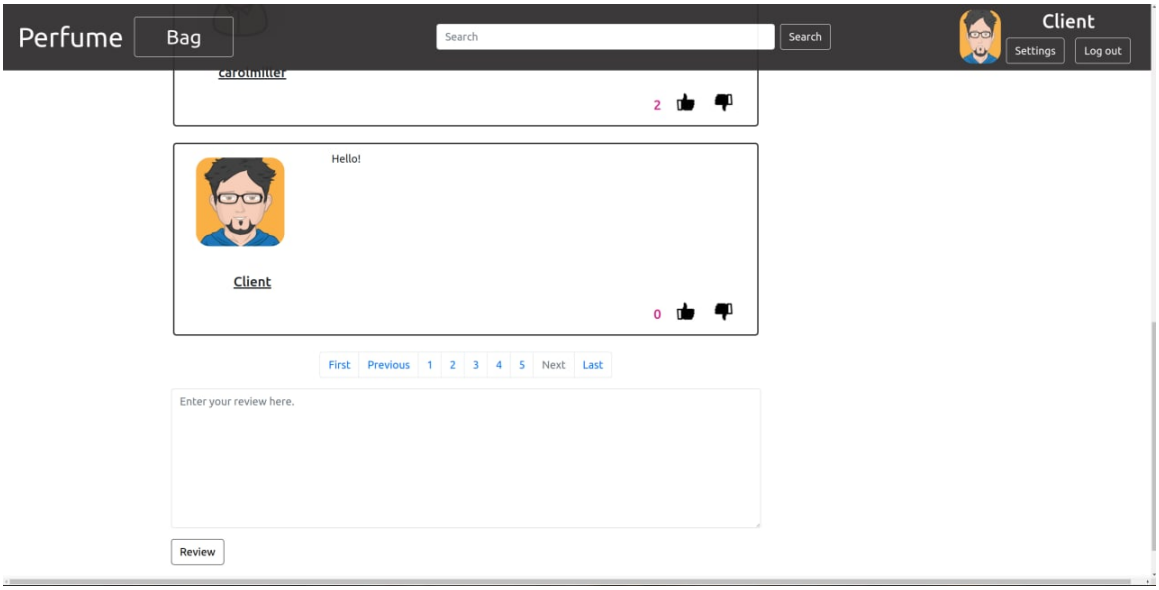

Рисунок 3.6 – Страница создания отзыва

3.4.7 Корзина товаров

На рисунке 3.7 представлена страница с корзиной товаров, добавленных пользователем. Для изменения количества товара в корзине следует воспользоваться кнопками - и +, расположенными напротив наименования товара. Когда количество товара в корзине станет равным нулю, товар будет удален из корзины. На данной странице также показывается итоговая стоимость заказа. Чтобы просмотреть информацию о товаре из корзины следует нажать на название товара. Для просмотра истории заказов нужно нажать кнопку "Order history".

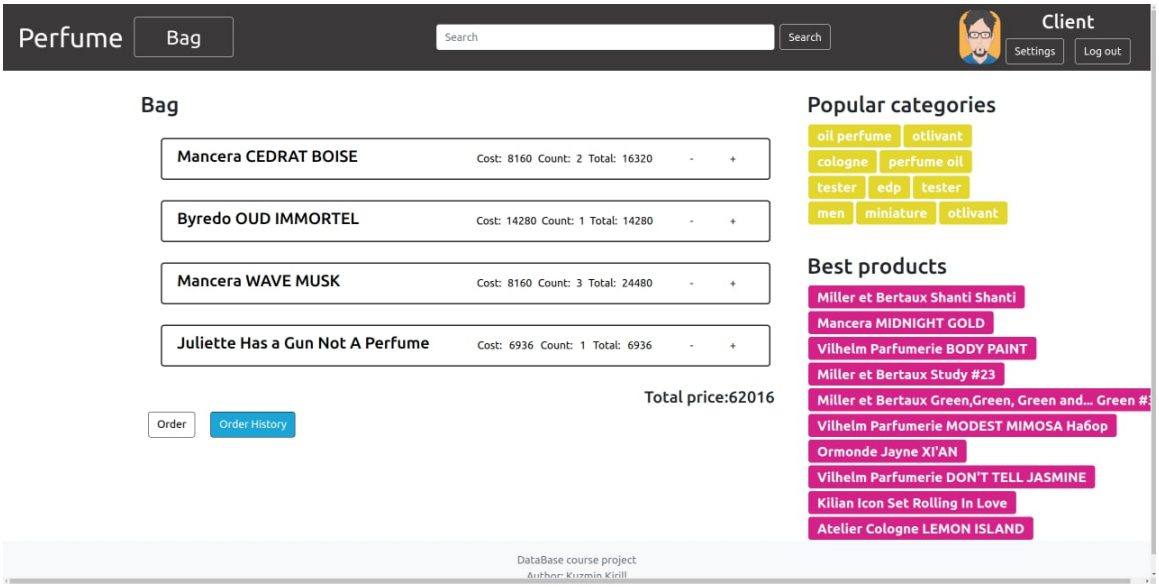

Рисунок 3.7 – Корзина товаров

3.4.8 История заказов пользователя

На рисунке 3.8 представлена страница истории заказов пользователя. Для отображения подробной информации о заказе следует нажать на нужный заказ, пользователь будет перенаправлен на страницу заказа.

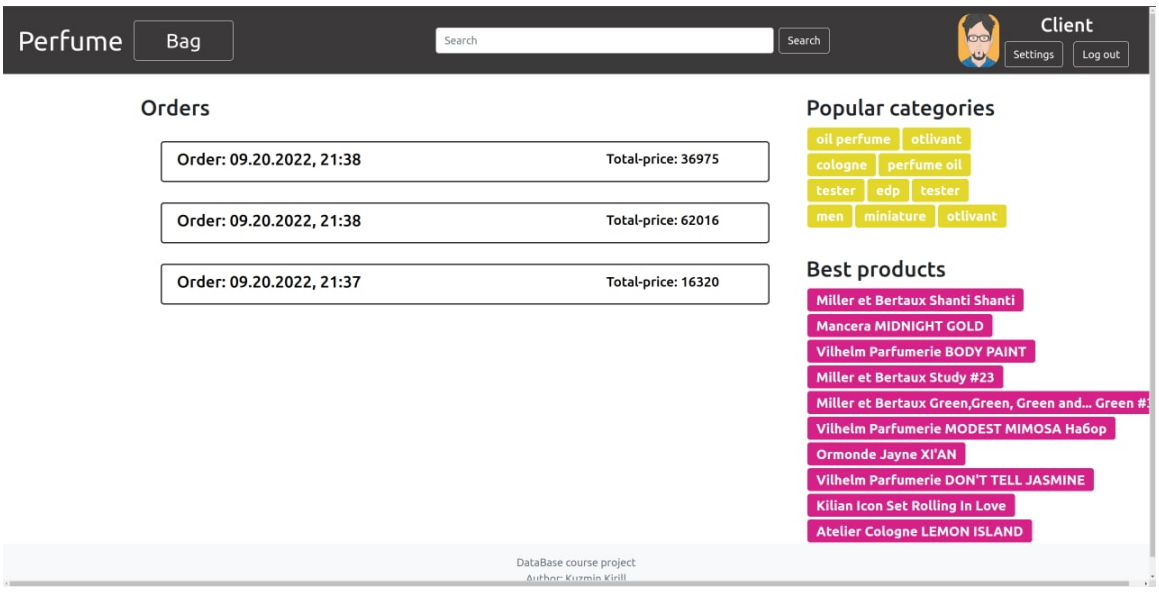

Рисунок 3.8 – Страница истории заказов

3.4.9 Страница управления товаром

На рисунке 3.9 представлена страница управления товаром с аккаунта администратора. Доступ к данному функционалу имеется у пользователей с ролью manager и administrator. Данная страница позволяет изменять характеристики товара, создавать и удалять товары.

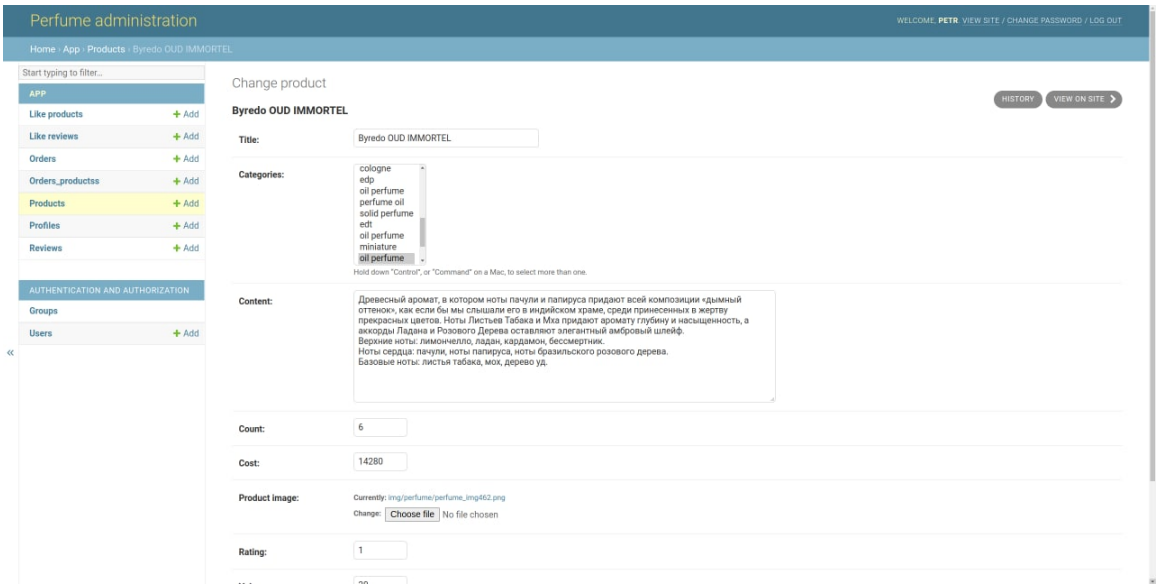

Рисунок 3.9 – Страница управления товаром

3.4.10 Страница управления пользователями

На рисунке 3.10 представлена страница управления пользователями. Пользователи с ролью administrator могут вносить изменения в поля пользователей, удалять и создавать новых пользователей. Manager может просматривать информацию пользователей.

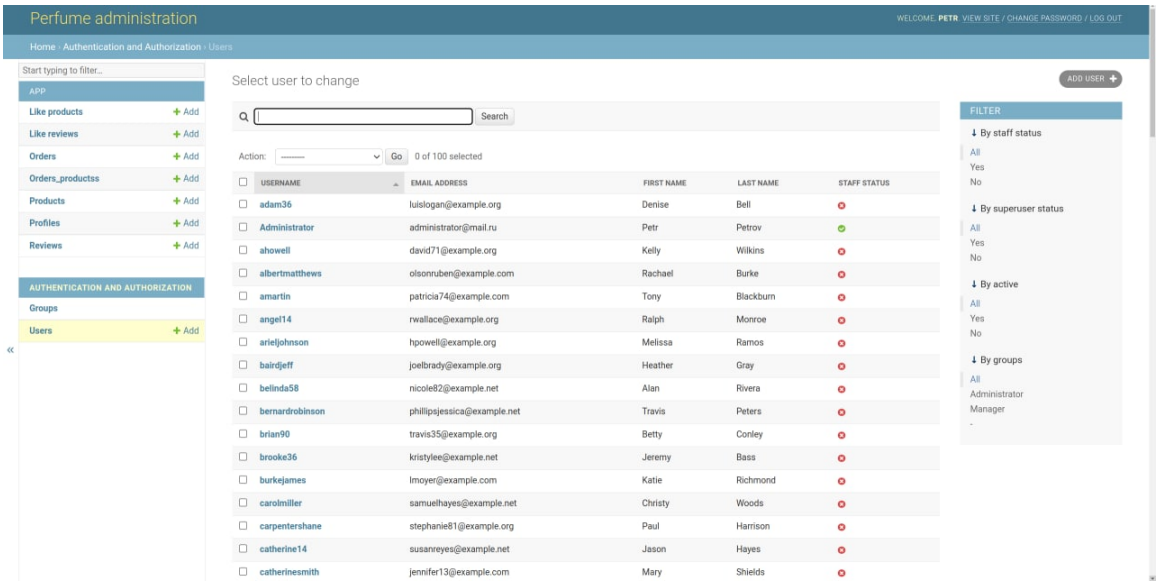

Рисунок 3.10 – Страница управления пользователями

4 Исследовательский раздел

В данном разделе будет проведено исследование зависимости времени исполнения запроса к базе данных от наличия индексации столбцов базы. Будет показан график, отображающий зависимость времени исполнения от индексации. В этом разделе также будут приведены технические характеристики устройства, на котором выполнялось измерение.

4.1 Технические характеристики

Тестирование выполнялось на устройстве со следующими техническими характеристиками:

– операционная система Ubuntu 20.04.3 LTS [8];

– память 7.5 GiB;

– процессор Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz × 8 [9].

Во время тестирования устройство было подключено к блоку питания и не нагружено никакими приложениями, кроме встроенных приложений окружения, окружением и системой тестирования.

4.2 Постановка эксперимента

Цель эксперимента — оценка времени исполнения запроса к базе данных с использованием индексации и без нее.

Оценка будет производиться для таблицы Product. Измеряться будет время поиска в данной таблице по полю title, оно же и будет проиндексировано.

4.3 Результаты эксперимента

В таблице 4.1 представлены результаты тестов.

Таблица 4.1 – Время исполнения запроса в зависимости от использования индексации

Количество записей в таблице	**Без индексации**	**С индексацией**
100	0.058	0.043
250	0.121	0.048
500	0.184	0.055
750	0.247	0.069
1000	0.361	0.081

На рисунке 4.1 представлен график зависимости времени исполнения запроса и использованием индексации и без нее от количества записей в таблице.

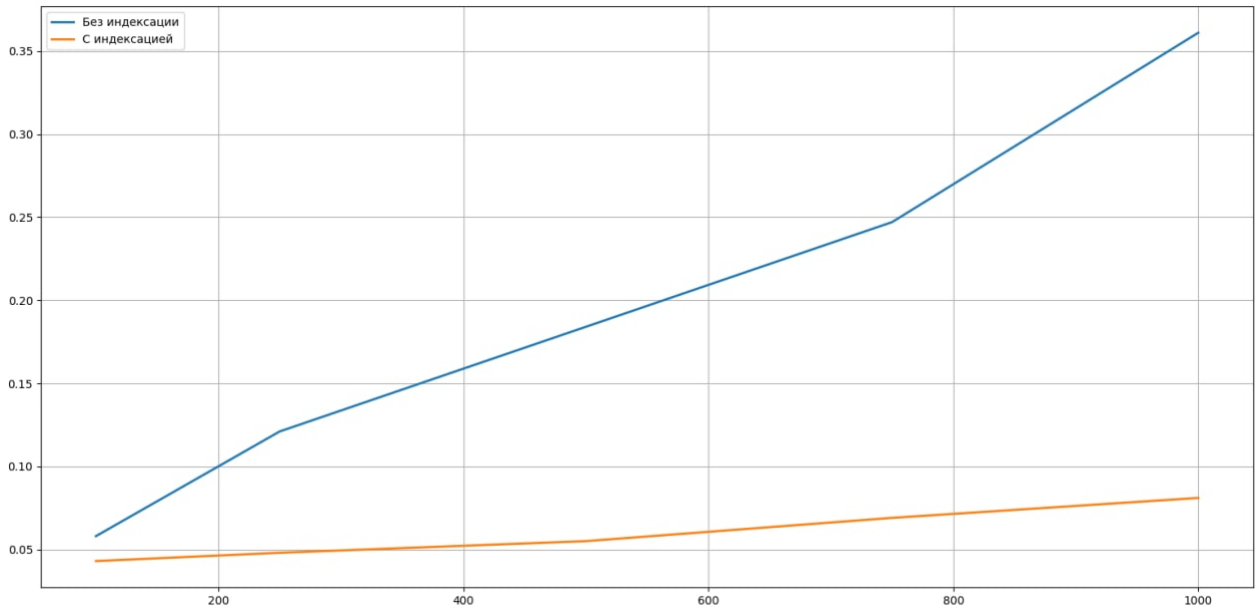

Рисунок 4.1 – График зависимости времени запроса от индексации

Вывод

По результатам эксперимента можно сделать вывод, что индексация существенно сокращает время ответа. При небольшой заполненности таблицы(100 записей) выигрыш по времени составляет 26%. При заполненности в 1000 записей выигрыш составляет 78%.

Заключение

В ходе выполнения курсовой работы было создано ПО для интернет-магазина парфюмерии.

Была спроектирована база данных для реализации требуемого функционала. Был изучен фреймворк Django и Django ORM для работы с базой данных. Было написано 2 функции и 6 триггеров, выделена ролевая модель. Было проведено исследование зависимости времени ответа БД от индексации.

В рамках выполнения работы решены следующие задачи:

- формализовано задание, определен необходимый функционал;
- проведен анализ СУБД;
- описана структура базы данных;
- спроектировано приложение для доступа к БД;
- создана и заполнена БД;
- реализован интерфейс для доступа к БД;
- разработано программное обеспечение, реализующую поставленную задачу.

Список использованных источников

1. Басов, А. С. Сравнение современных СУБД / А. С. Басов // Вестник науки. – 2020. – Т. 4. – № 7(28). – С. 50-54. – EDN LULYQU.

2. Дейт К. Дж. Введение в системы баз данных. – 8-е изд. – М.: «Вильямс», 2006.

3. Егорова, И. Е. Базы данных : учебное пособие / И. Е. Егорова, М. В. Коротеев. – Волгоград : Волгоградский государственный технический университет, 2016. – 120 с. – ISBN 978-5-9948-2256-2. – EDN XSAUNP.

4. Python 3.10.7 documentation [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/ свободный – (20.08.2022).

5. PostgreSQL: Документация. [Электронный ресурс]. – Режим доступа: https://postgrespro.ru/docs/postgresql/ свободный – (20.08.2022).

6. Django documentation [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/en/4.1/ свободный – (20.08.2022).

7. Bootstrap documentation [Электронный ресурс]. – Режим доступа: https://getbootstrap.com/docs/5.2/getting-started/introduction/ свободный – (20.08.2022).

8. Ubuntu: Enterprise Open Source and Linux [Электронный ресурс]. – Режим доступа: https://ubuntu.com/ свободный – (20.08.2022).

9. Процессор Intel® Core™ i7-8550U [Электронный ресурс]. – Режим доступа: https://ark.intel.com/content/www/us/en/ark/products/122589/intel-core-i78550u-processor-8m-cache-up-to-4-00-ghz.html свободный – (20.08.2022).

Приложение А

Листинг 4.1 – Классы таблиц базы данных(часть 1)

```
class Product(models.Model):
    title = models.CharField(max_length=100)
    categories = models.ManyToManyField('Category', related_name='products')
    content = models.TextField()
    count = models.IntegerField(default=0)
    cost = models.IntegerField(default=0)
    product_image = models.ImageField(
        upload_to='img/%Y/%m/%d/', default='img/1.jpg')
    pub_date = models.DateTimeField(auto_now_add=True)
    rating = models.IntegerField(default=0)
    volume = models.IntegerField(default=100)

class Order(models.Model):
    status = models.CharField(max_length=50, default="Pending")
    order_date = models.DateTimeField(auto_now_add=True)
    date_of_completion = models.DateTimeField(null=True, blank=True)
    comment = models.TextField(default="")
    profile = models.ForeignKey(
        'Profile', on_delete=models.CASCADE, related_name='orders')
    products = models.ManyToManyField(
        'Product', through="Orders_products", related_name='orders')

class Orders_products(models.Model):
    order = models.ForeignKey(
        'Order', on_delete=models.CASCADE, related_name='ordersproducts')
    product = models.ForeignKey(
        'Product', on_delete=models.CASCADE, related_name='ordersproducts')
    cnt = models.IntegerField(default=1)

class Review(models.Model):
    content = models.CharField(max_length=150)
    review_date = models.DateTimeField(auto_now_add=True)
    rating = models.IntegerField(default=0)
    profile = models.ForeignKey(
        'Profile', on_delete=models.CASCADE, related_name='reviews')
    product = models.ForeignKey(
        'Product', on_delete=models.CASCADE, related_name='reviews')
```

Листинг 4.2 – Классы таблиц базы данных(часть 2)

```
class LikeProduct(models.Model):
    mark = models.IntegerField(default=0)
    pub_date = models.DateTimeField(auto_now_add=True)
    product = models.ForeignKey(
        'Product', on_delete=models.CASCADE, related_name="likes")
    profile = models.ForeignKey(
        'Profile', on_delete=models.CASCADE, related_name="product_likes")

class LikeReview(models.Model):
    mark = models.IntegerField(default=0)
    pub_date = models.DateTimeField(auto_now_add=True)
    review = models.ForeignKey(
        'Review', on_delete=models.CASCADE, related_name="likes")
    profile = models.ForeignKey(
        'Profile', on_delete=models.CASCADE, related_name="review_likes")


class Category(models.Model):
    name = models.CharField(max_length=50)

class Profile(models.Model):
    user = models.OneToOneField(
        User, on_delete=models.CASCADE, null=False, related_name='profile')
    birth_date = models.DateField(null=True, blank=True)
    address = models.CharField(max_length=150)
    sex = models.CharField(max_length=10)
    avatar = models.ImageField(
        upload_to='img/%Y/%m/%d/', default='img/ava.jpg')
```

Приложение Б

Листинг 4.3 – Создание ограничений на таблицы

```
ALTER TABLE app_product
    ADD CONSTRAINT correct_count CHECK (count>=0);

ALTER TABLE app_product
    ADD CONSTRAINT correct_volume CHECK (volume>0 AND volume<=200);

ALTER TABLE app_product
    ADD CONSTRAINT correct_cost CHECK (cost>=0);

ALTER TABLE app_orders_products
    ADD CONSTRAINT correct_cnt CHECK (cnt>=0);

ALTER TABLE app_likeproduct
    ADD CONSTRAINT correct_mark_likeproduct CHECK (mark>=-1 AND mark<=1);

ALTER TABLE app_likereview
    ADD CONSTRAINT correct_mark_likereview CHECK (mark>=-1 AND mark<=1);

ALTER TABLE app_profile
    ADD CONSTRAINT correct_sex CHECK (sex = 'Male' OR sex = 'Female');
```

Приложение В

Листинг 4.4 – Роли на уровне БД

```
CREATE ROLE Client NOSUPERUSER NOCREATEDB;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO Client;
GRANT INSERT, UPDATE ON auth_user TO Client;
GRANT INSERT, UPDATE ON app_profile TO Client;
GRANT INSERT ON app_review TO Client;
GRANT INSERT, UPDATE ON app_likeproduct TO Client;
GRANT INSERT, UPDATE ON app_likereview TO Client;

CREATE ROLE Manager NOSUPERUSER NOCREATEDB;
GRANT USAGE, SELECT ON ALL SEQUENCES IN SCHEMA public TO Manager;
GRANT ALL PRIVILEGES ON app_order TO Manager;
GRANT ALL PRIVILEGES ON app_product TO Manager;
GRANT ALL PRIVILEGES ON app_category TO Manager;
GRANT ALL PRIVILEGES ON app_review TO Manager;
GRANT ALL PRIVILEGES ON app_product_categories TO Manager;
GRANT ALL PRIVILEGES ON app_orders_products TO Manager;

CREATE ROLE Administrator CREATEROLE NOSUPERUSER NOCREATEDB;
GRANT ALL PRIVILEGES ON auth_user TO Administrator;
GRANT ALL PRIVILEGES ON app_profile TO Administrator;
GRANT ALL PRIVILEGES ON app_product TO Administrator;
GRANT ALL PRIVILEGES ON app_order TO Administrator;
GRANT ALL PRIVILEGES ON app_category TO Administrator;
GRANT ALL PRIVILEGES ON app_review TO Administrator;
GRANT ALL PRIVILEGES ON app_likeproduct TO Administrator;
GRANT ALL PRIVILEGES ON app_likereview TO Administrator;
GRANT ALL PRIVILEGES ON app_product_categories TO Administrator;
GRANT ALL PRIVILEGES ON app_orders_products TO Administrator;
```