|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ МЕЖДУНАРОДНЫХ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ (ФМОП)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

***Магазин музыкальных альбомов***

Студент группы **ИУ7И-66Б** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Андрич Катарина**

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Романова Т. Н.**

*2023 г.*

**РЕФЕРАТ**

Расчетно-пояснительная записка содержит 34 страницы, 13 рисунков, 3 таблицы, 6 источников.

Цель курсовой работы — разработка и реализация магазина музыкальных альбомов.

Для этого нужно проанализировать и выбрать технологию для реализации приложения, разработать структуру базы данных и провести исследование.

Цель курсовой работы, проектирование и реализация магазина альбомов с помощью Python и PostgreSQL, достигнута.

Дальнейшее развитие проекта подразумевает:

* добавление фильтров для альбомов;
* добавление других видов подарков;
* добавление вариантов доставки.

КЛЮЧЕВЫЕ СЛОВА

*базы данных, разработка ПО, альбомы, музыка, заказ, Python, PostgreSQL*

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 5**](#_Toc28256)

[**1 Аналитическая часть 6**](#_Toc21473)

[1.1 Формализация задачи 6](#_Toc23296)

[1.2 ER модель разработанной базы данных 7](#_Toc13934)

[1.3 Глоссарий предметной области 8](#_Toc19126)

[1.4 Анализ существующих решений 9](#_Toc28500)

[1.5 Анализ типов и выбор СУБД 11](#_Toc7562)

[1.5.1 Реляционные базы данных 11](#_Toc15721)

[1.5.2 Нереляционные базы данных 12](#_Toc8763)

[1.6 Вывод 12](#_Toc8241)

[**2 Конструкторская часть 13**](#_Toc22753)

[2.1 Проектирование базе данных 13](#_Toc20938)

[2.2 Требования к программе 15](#_Toc16729)

[2.3 Вывод 16](#_Toc23097)

[**3 Технологическая часть 17**](#_Toc13724)

[3.1 Выбор СУБД 17](#_Toc23391)

[3.2 Выбор языка программирования и среды программирования 18](#_Toc1157)

[3.3 Детали реализации 19](#_Toc13108)

[3.4 Способы и этапы тестирования 23](#_Toc31858)

[3.5 Вывод 24](#_Toc26051)

[**4 Исследовательская часть 26**](#_Toc19424)

[4.1 Цель эксперимента 26](#_Toc8137)

[4.2 Постановка эксперимента 26](#_Toc3074)

[4.3 Результат эксперимента 27](#_Toc24428)

[4.4 Вывод 28](#_Toc27932)

[**ЗАКЛЮЧЕНИЕ 29**](#_Toc31667)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30**](#_Toc31397)

[**ПРИЛОЖЕНИЕ А 31**](#_Toc18590)

[**ПРИЛОЖЕНИЕ Б 32**](#_Toc23927)

[**ПРИЛОЖЕНИЕ В 33**](#_Toc17676)

[**ПРИЛОЖЕНИЕ Г 34**](#_Toc29856)

**ВВЕДЕНИЕ**

За последнее десятилетие интернет стал все более популярным. Вначале он использовался главным образом для связи и поиска информации, но сегодня он стал самым большим магазином в мире.

Онлайн-шопинг имеет множество преимуществ. Покупатели могут искать широкий ассортимент товаров, не выходя из дома. Кроме того, это позволяет покупателям преодолевать границы своей страны и покупать товары, которые в противном случае были бы недоступны. Сегодня легче, чем когда-либо, найти выгодные предложения и дешёвые цены.

С другой стороны, с развитием интернета основным источником музыки для многих людей стали цифровые платформы, а не физические носители. Однако в последние годы популярность компакт-дисков снова начала возрастать, и люди покупают несколько версий для своих коллекций.

Цель курсовой работы — разработка и реализация магазина музыкальных альбомов.

Для достижения цели необходимо:

* проанализировать существующие СУБД;
* выбрать технологию для реализации приложения;
* разработать структуру базы данных для хранения и организации данных;
* провести исследование.

**1 Аналитическая часть**

**1.1 Формализация задачи**

Необходимо создать и разработать приложение с функционалом, поддерживающим несколько типов пользователей.

Неавторизованному пользователю разрешено:

* просматривать каталог и описание товаров;
* зарегистрировать профиль.

Пользователю разрешено:

* просматривать каталог и описание товаров;
* добавить товары в корзину;
* оформить заказ;
* просматривать предыдущие заказы;
* изменить и удалить профиль.

Администратору разрешено:

* создать, изменить и удалить товары.

Менеджеру разрешено:

* просматривать все заказы;
* отменить и изменить заказ.

Нужно проверять правильность формата данных, которые вводятся.

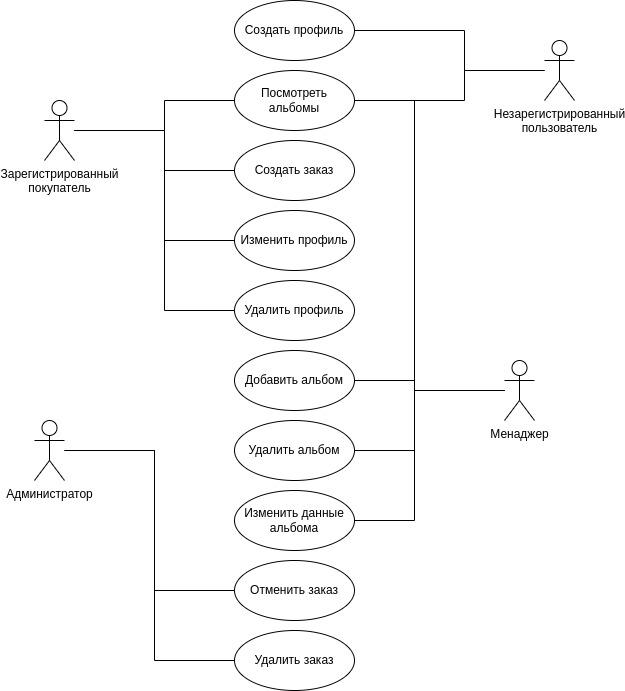


Рисунок 1 – Use-case диаграмма приложения

**1.2 ER модель разработанной базы данных**

ER-диаграмма (сокращение от Entity-Relationship diagram) – это графический инструмент, используемый для описания и моделирования связей между сущностями в базе данных. ER моделирование помогает систематически анализировать требования к данным для создания хорошо спроектированной базы данных [1]. Модель сущность-связь является концептуальной моделью данных, которая помогает организовать и представить связи между различными сущностями в системе.

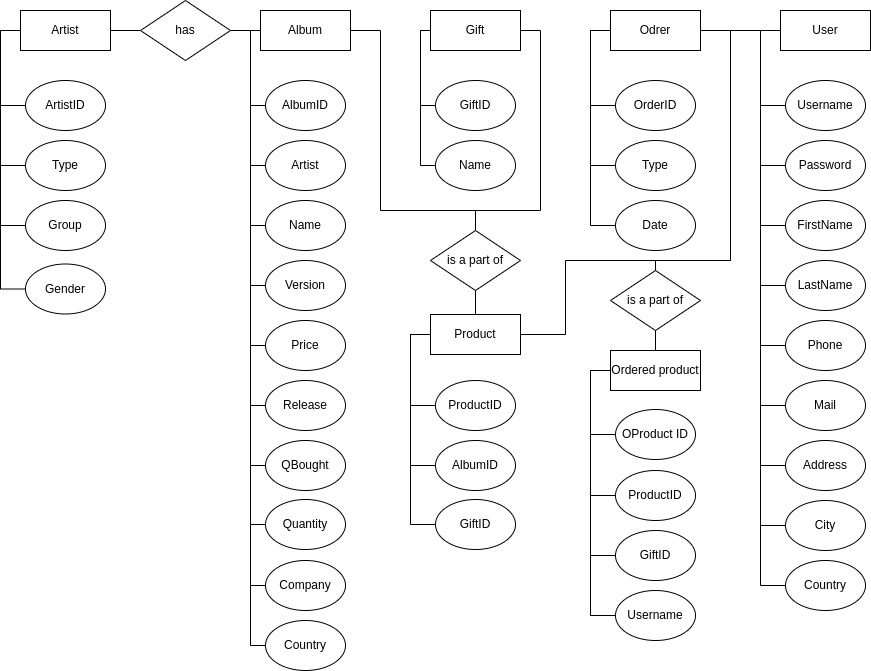


Рисунок 2 – ER-диаграмма приложения в нотации Чена

**1.3** **Глоссарий предметной области**

В таблице 1 представлен глоссарий предметной области.

Таблица 1 – глоссарий предметной области

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Товар | Любой продукт, который продаётся |
| Заказ | Запрос на приобретение товара или услуги |
| Магазин | Торговое помещение, где продаются товары |
| Пользователь | Человек, который пользуется услугами или покупает товары у продавца или поставщика |
| Администратор | Специалист, ответственный за управление товарами |
| Менеджер | Специалист, который занимается продажами товаров |

**1.4** **Анализ существующих решений**

Некоторые из самых популярных магазинов, ориентированные на продажу музыкальных альбомов:

* Kpop Srbija
* Candy Shop

Kpop Srbija — сайт для продажи альбомов.

Возможности — регистрация на сайте, просмотр товаров, фильтрация по исполнителям, пред-заказ альбомов.

Преимуществом является возможность пред-заказа с предзаказанными плюшками.

Недостатком является то, что невозможно сделать под-заказ.

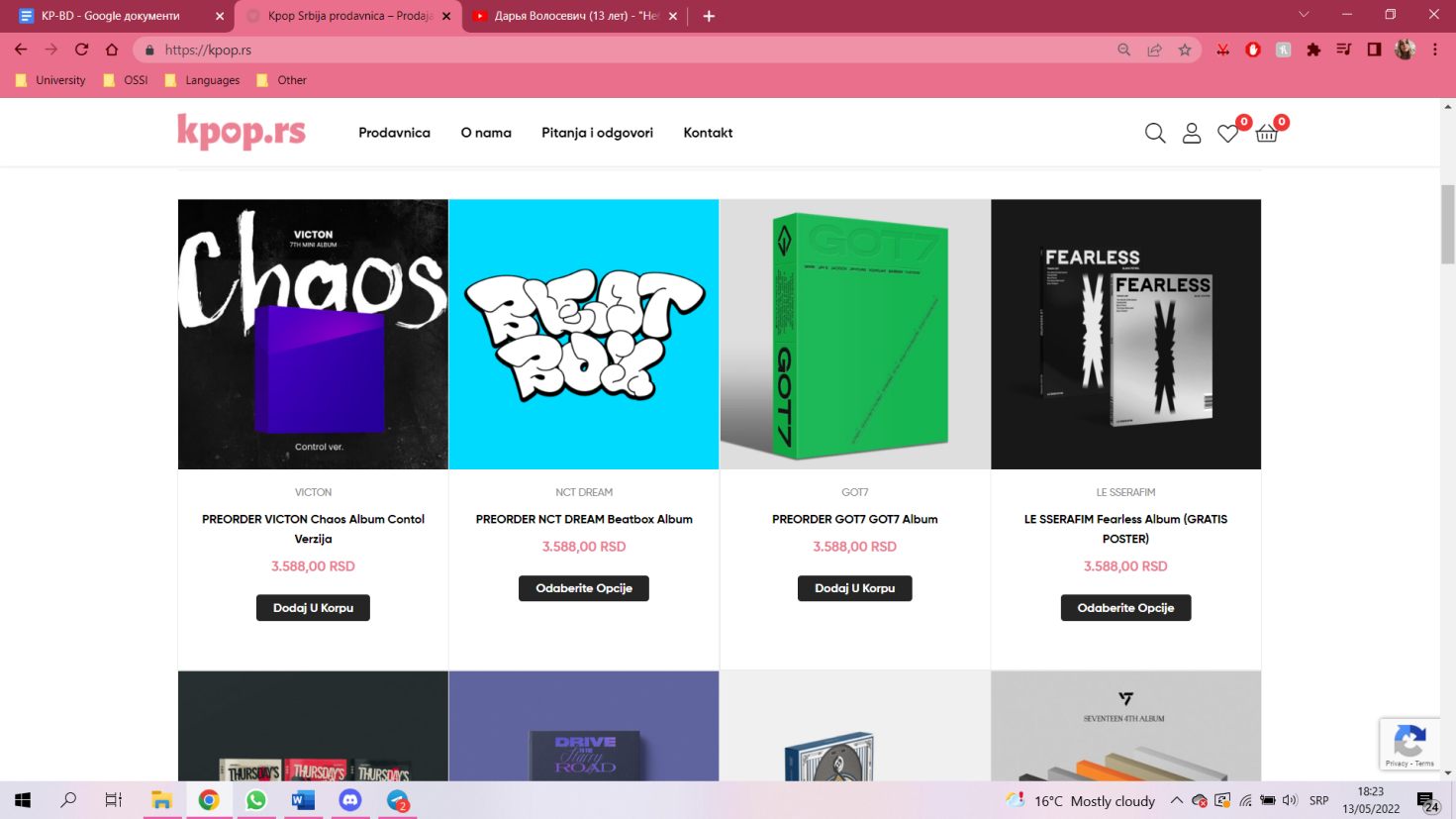


Рисунок 3 – интерфейс сайта Kpop Srbija.

Candy Shop — сайт для продажи альбомов.

Возможности — регистрация на сайте, просмотр товаров, фильтрация по исполнителям, пред-заказ и под-заказ альбомов.

Преимуществом является возможность пред-заказа и под-заказа

Недостатками являются невозможность сделать пред-заказ с предзаказанными плюшками, невозможность под-заказа определённой версии при наличии остальных и показ неудачных покупок, мешающие просмотру.

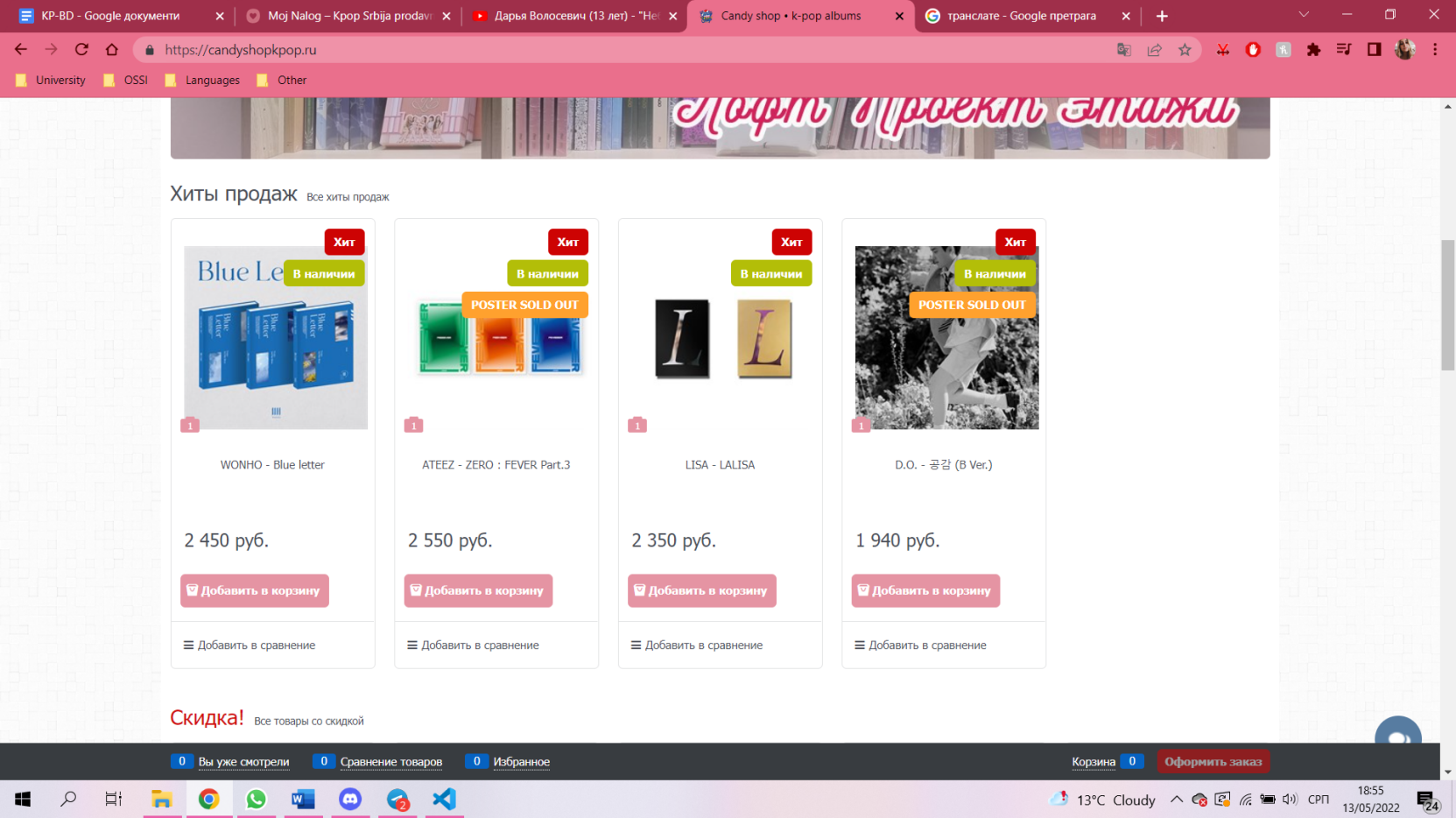


Рисунок 4 – интерфейс сайта Candy Shop

**1.5 Анализ типов и выбор СУБД**

Типы баз данных, называемых также моделями БД или семействами БД, представляют собой шаблоны и структуры, используемые для организации данных в системе управления базами данных (СУБД) [2].

Система управления базами данных (СУБД) – это программное обеспечение, предназначенное для организации и управления базами данных. Она обеспечивает функциональность по созданию, модификации, удалению и извлечению данных из базы данных, а также обеспечивает безопасность и целостность данных. СУБД позволяет эффективно хранить, обрабатывать и управлять большими объёмами информации, предоставляя удобный интерфейс для работы с данными [5]. Существуют различные виды СУБД, но в основном разделяются на:

* реляционные — SQL;
* нереляционные — NoSQL.

**1.5.1** **Реляционные базы данных**

Реляционная база данных (SQL) — база, где данные хранятся в формате таблиц, они строго структурированы и связаны друг с другом. В таблице есть строки и столбцы, каждая строка представляет отдельную запись, а столбец — поле с назначенным ей типом данных. В каждой ячейке информация записана по шаблону [3].

Реляционные системы управления базами данных (РСУБД) обеспечивают высокую производительность. Они обеспечивают надёжную защиту информации от потерь. Реляционные базы данных легко масштабируются, если они размещены на одном сервере. РСУБД подходят для работы с данными у которых ясная структура.

**1.5.2** **Нереляционные базы данных**

Нереляционные базы данных (NoSQL) хранят данные без чётких связей друг с другом и чёткой структуры и предлагают динамическую структуру данных, которые могут храниться несколькими способами: ориентированно по колонкам, документо-ориентированно, в виде графов или на основе пар «ключ-значение» [4].

NoSQL базы данных ускоряют процесс разработки и делают возможным поэтапную реализацию благодаря гибким схемам. Потому что пользуют гибкие модели данных, БД NoSQL представляют отличный выбор для обработки частично структурированных и неструктурированных данных.

**1.6 Вывод**

В этой части была представлена формализация задачи, изложены основные принципы реляционных и нереляционных баз данных, а также была выбрана модель хранения.

Проведён анализ существующих аналогичных продуктов. Однако ни одно из существующих решений не обеспечивает всех возможностей, таких как пред-заказ с предзаказанными плюшками и под-заказ определённой версии.

**2 Конструкторская часть**

**2.1 Проектирование базе данных**

База данных приложения содержит следующие таблицы:

* таблица с альбомами;
* таблица с исполнителями;
* таблица с заказами;
* таблица с пользователями;
* таблица с выгодами;
* таблица, объединяющая альбом и подарок;
* таблица, объединяющая заказ, пользователя и таблицу с альбомом и подарком.

Таблица с альбомами должна содержать информацию о:

* AlbumID – идентификатор альбома (PK);
* Version – версия альбома (text);
* Price – цена альбома (integer);
* ArtistID – идентификатор исполнителя (FK);
* Name – название альбома (text);
* Release – дата выпуска (date);
* QBought – купленное количество (integer);
* Quantity – количество доступных альбомов (integer);
* Company – компания, выпустившая альбом (varchar);
* Country – страна из которой альбом (varchar).

Таблица с исполнителями должна содержать информацию о:

* ArtistID – идентификатор исполнителя (PK);
* Type – тип исполнителя (varchar);
* Group – название группы исполнителя (text);
* Gender – пол исполнителя (char).

Таблица с заказами должна содержать информацию о:

* OdrerID – идентификатор заказа (PK);
* Date – дата заказа (date);
* Type – тип заказа (varchar).

Таблица с пользователями должна содержать информацию о:

* Username – идентификатор пользователя (PK);
* Password – пароль пользователя (varchar);
* Phone – номер телефона (int);
* Mail – адрес электронной почты (text);
* Address – адрес (text);
* City – город (varchar);
* Country – страна (varchar);
* FirstName – имя (varchar);
* LastName – фамилия (varchar).

Таблица с выгодами должна содержать информацию о:

* GiftID – идентификатор выгоды (PK);
* Name – имя выгоды (varchar).

Таблица, объединяющая альбом и подарок должна содержать информацию о:

* APGID – идентификатор комбинации (PK);
* AlbumID – идентификатор альбума (FK);
* GiftID – идентификатор выгоды (FK).

Таблица, объединяющая заказ, пользователя и таблицу с альбомом и подарком должна содержать информацию о:

* OrdAlbID – идентификатор комбинации (PK);
* OrderID – идентификатор заказа (FK);
* APGID – идентификатор комбинации альбома и подарка (FK);
* Username – идентификатор пользователя (FK).

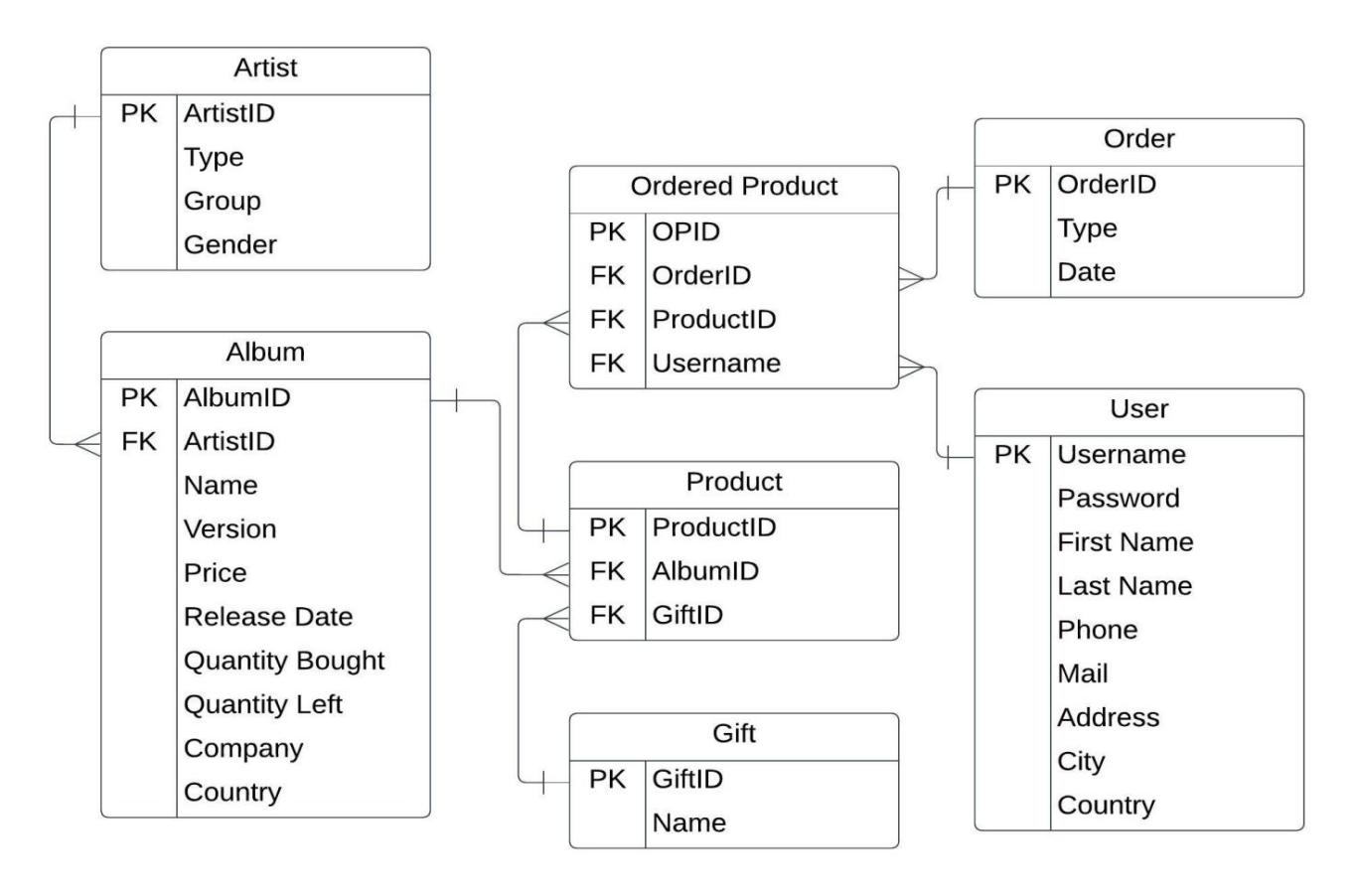


Рисунок 5 – Схема базы данных

**2.2 Требования к программе**

Разрабатываемое программное обеспечение должно обеспечивать следующие возможности:

* отображение списка альбомов (в соответствии с датой выпуска);
* отображение соответствующих подарков;
* регистрация нового пользователя;
* авторизация пользователя;
* добавление комбинации альбома и подарка в корзину;
* создание заказа;
* изменение данных пользователя;
* удаление пользователя;
* просмотр предыдущих заказов;
* удаление заказа;
* изменение заказа;
* добавление альбома;
* изменение альбома;
* удаление альбома.

Ограничения:

* пароль от профиля хранится в БД без шифрования;
* редактировать и удалять можно только заказы, сделанные в этот и предыдущий день.

**2.3 Вывод**

В данном разделе была выполнена разработка базы данных. В процессе разработки была создана схема базы данных, которая описывает структуру и отношения между таблицами. Кроме того, были перечислены требования к программе, которые определяют функциональные характеристики ПО. Разработка базы данных является важным этапом проектирования, поскольку она обеспечивает хранение и организацию данных, необходимых для работы программы. Благодаря схеме базы данных и перечисленным требованиям, можно приступить к созданию программного решения, соответствующего заданным условиям и функциональности.

**3 Технологическая часть**

**3.1 Выбор СУБД**

Существует несколько популярных реляционных СУБД, таких как PostgreSQL, MySQL, Microsoft SQL Server (MSSQL) и Oracle.

В данном случае, выбран PostgreSQL по следующим причинам:

1. Открытый исходный код и свободная лицензия: PostgreSQL является СУБД с открытым исходным кодом, что позволяет свободно использовать и модифицировать её. Это позволяет гибко настраивать СУБД под конкретные требования проекта, а также экономит затраты на лицензирование.
2. Мощная функциональность: PostgreSQL предлагает широкий спектр функций, поддерживая продвинутые возможности, такие как полнотекстовый поиск, триггеры, хранимые процедуры и другие расширения.
3. Надёжность и стабильность: PostgreSQL известен своей надёжностью и стабильностью. Он обладает хорошо разработанной архитектурой, обеспечивающей целостность данных и отказоустойчивость. PostgreSQL также обеспечивает механизмы резервного копирования и восстановления, что позволяет обеспечить сохранность данных.
4. Расширяемость и сообщество: PostgreSQL обладает активным сообществом разработчиков и пользователей, которые создают дополнительные модули и расширения. Это позволяет легко добавлять новые функциональные возможности и интегрировать PostgreSQL с другими системами.

В сравнении с MySQL, MSSQL и Oracle, PostgreSQL обладает схожим функционалом, однако предлагает некоторые преимущества, такие как большая гибкость настройки и открытый исходный код.

**3.2 Выбор языка программирования и среды программирования**

В качестве языка программирования был выбран Python. Выбор был сделан по нескольким причинам:

1. Простота и читаемость кода: Python имеет понятный и лаконичный синтаксис, который облегчает чтение и написание кода [6]. Это позволяет быстро разрабатывать и поддерживать программу.
2. Богатая экосистема библиотек: Python имеет обширную коллекцию библиотек, которые предоставляют готовые решения для различных задач. В проекте пользуется библиотека psycopg2 для взаимодействия с базой данных PostgreSQL. Кроме того, для создания графического интерфейса пользуется библиотека tkinter. Благодаря этому, можно создавать интуитивно понятные пользовательские интерфейсы.

В качестве среды разработки был выбран Visual Studio Code. Выбор был сделан по нескольким причинам:

1. Мощные функциональные возможности: Visual Studio Code (VS Code) предлагает широкий набор инструментов и расширений, которые помогают упростить и ускорить процесс разработки. Он обладает интегрированной системой контроля версий, подсветкой синтаксиса, автодополнением и отладчиком, что значительно повышает продуктивность и помогает в создании качественного кода.
2. Поддержка языка программирования Python: VS Code предоставляет полноценную поддержку для разработки на Python. Он обладает мощными функциями, такими как автодополнение кода, проверка синтаксиса, отладка и интеграция с виртуальными окружениями. Это делает VS Code идеальным выбором для разработки на языке Python.

Сочетание Python, библиотеки psycopg2 и tkinter, а также Visual Studio Code обеспечивает гибкость, удобство и эффективность в разработке проекта, основанного на использовании базы данных PostgreSQL и создании графического интерфейса для пользователей.

**3.3 Детали реализации**

В приложении A показано, как реализовано соединение с базой данных с использованием ролевой модели.

В приложении Б представлены примеры некоторых SQL-запросов, используемых в программе.

В приложениях В и Г показаны процедуры, используемые в программе для обновления купленных и доступных альбомов.

На изображениях ниже показаны примеры интерфейса.

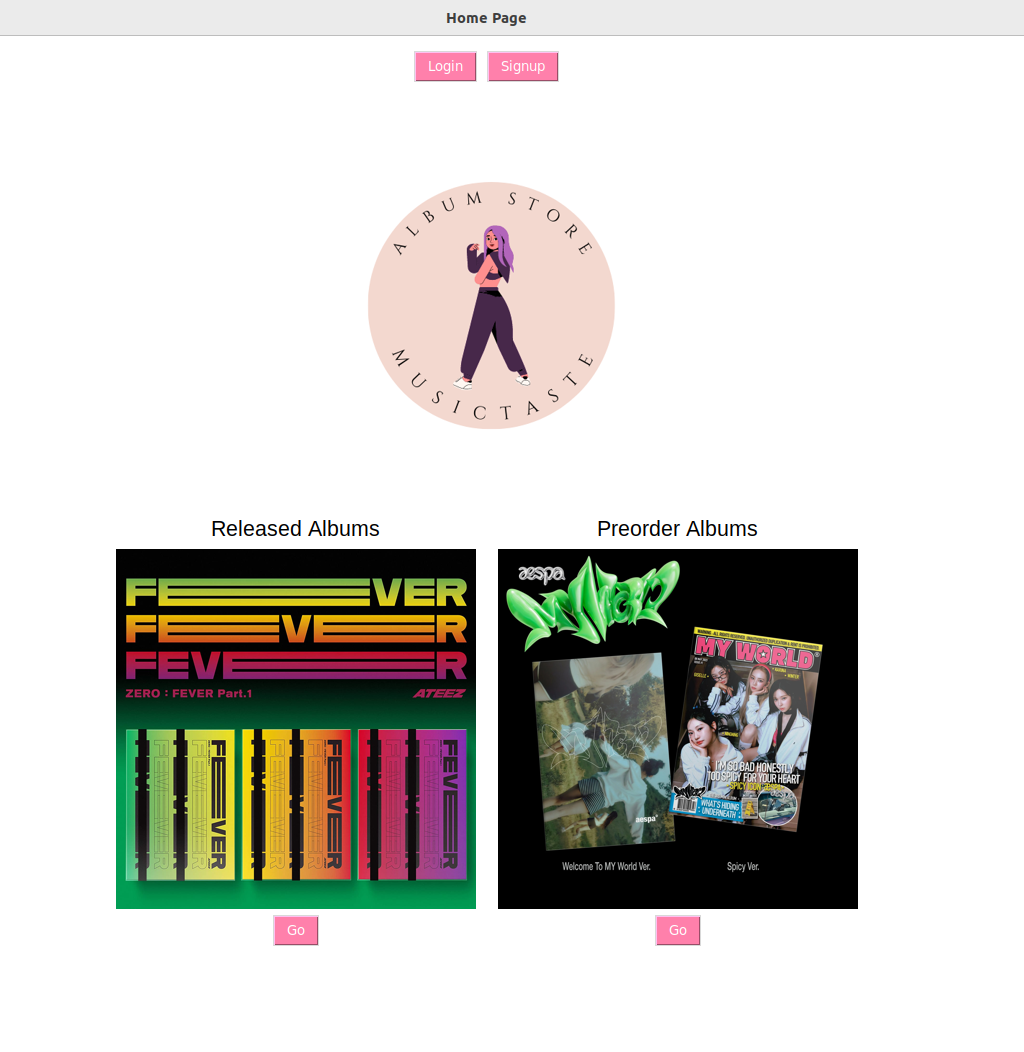
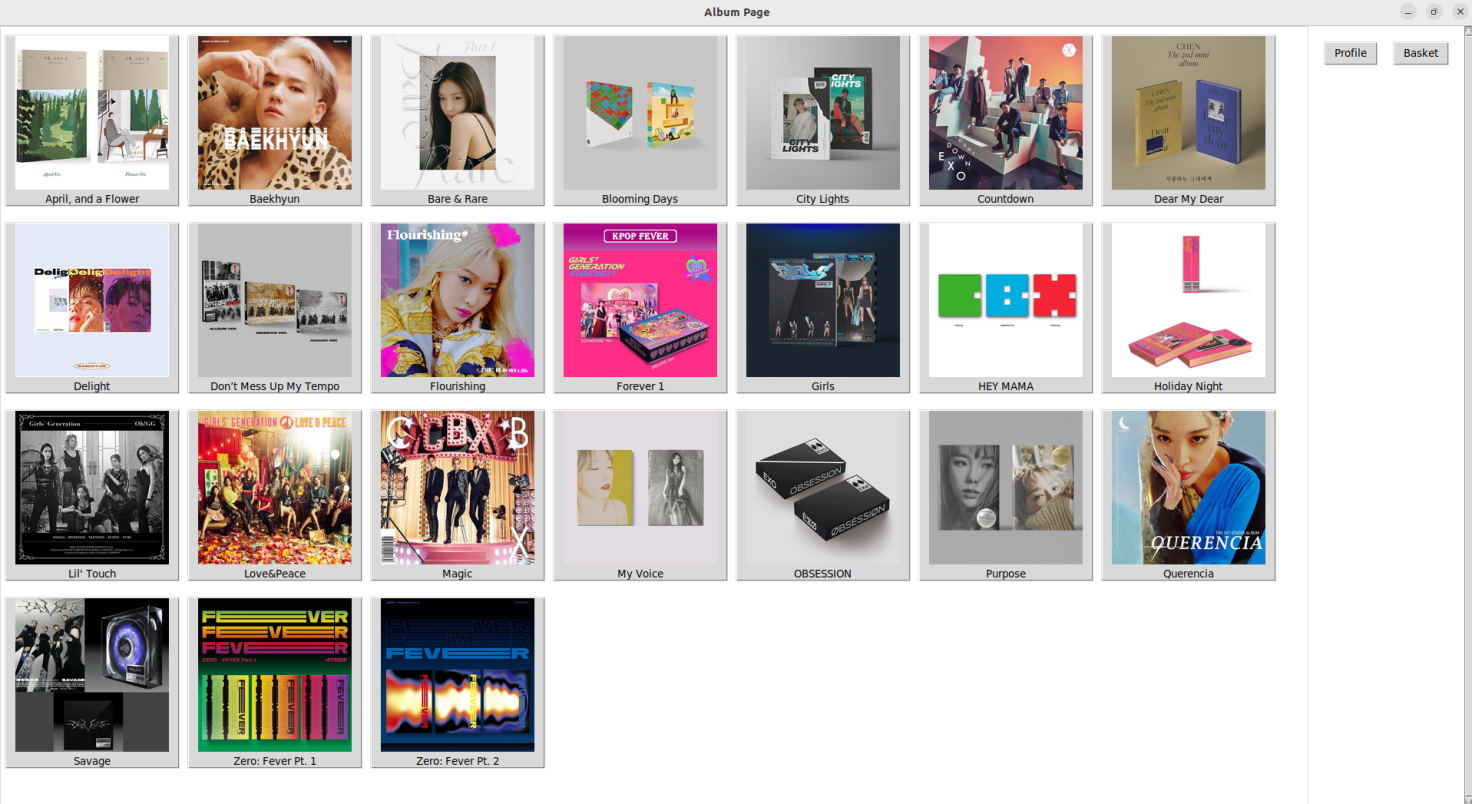


Рисунок 6 – Стартовый экран

Рисунок 7 – Окно продуктов

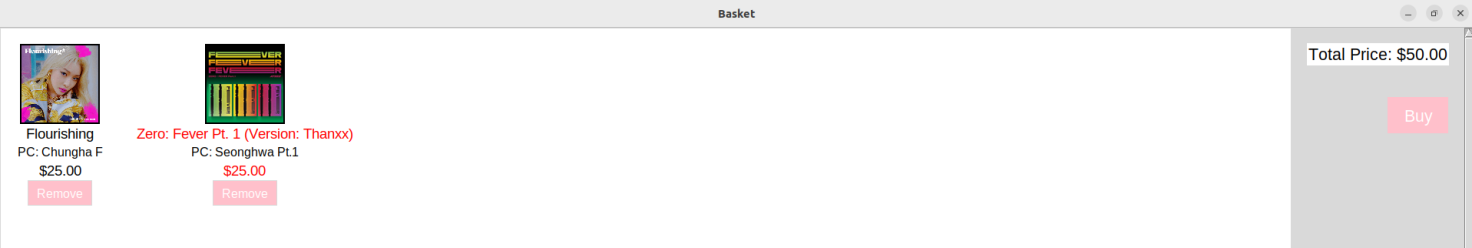


Рисунок 8 – Корзина

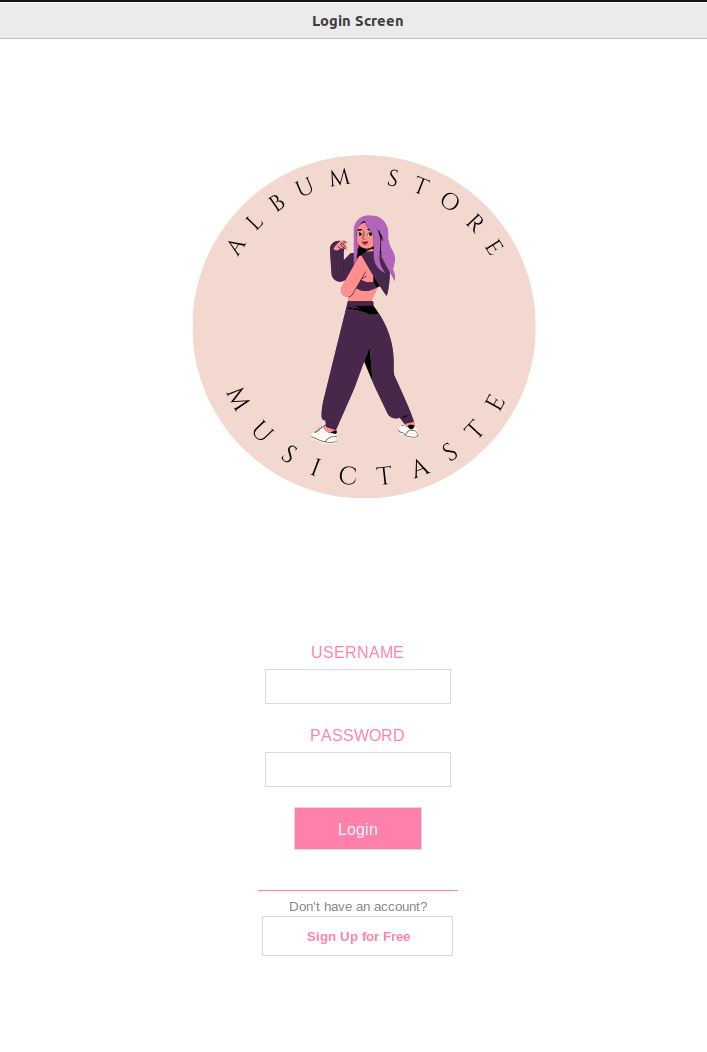
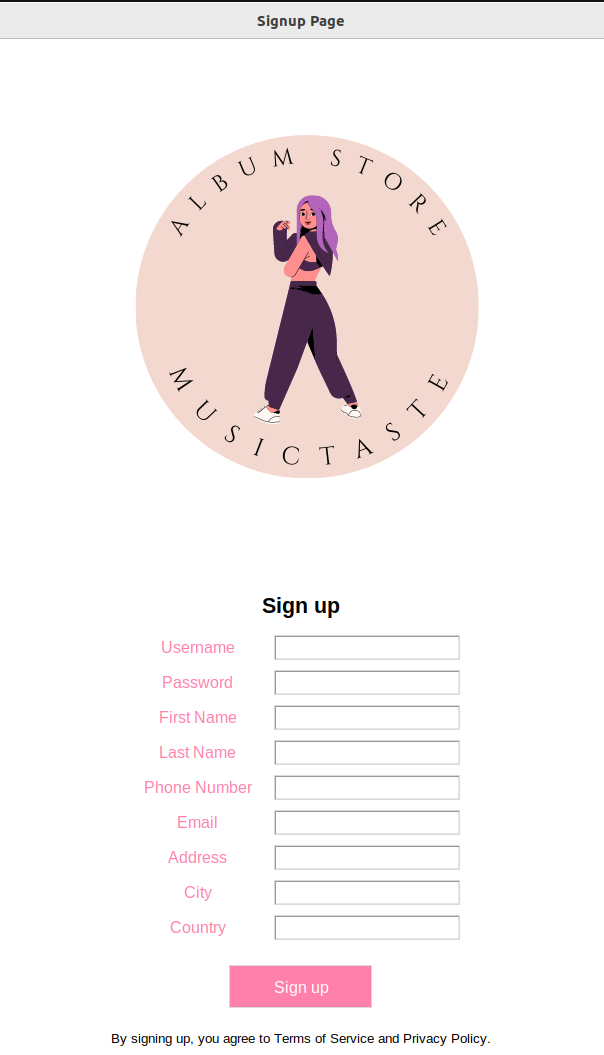
 

Рисунок 9 – Окна входа и регистрации

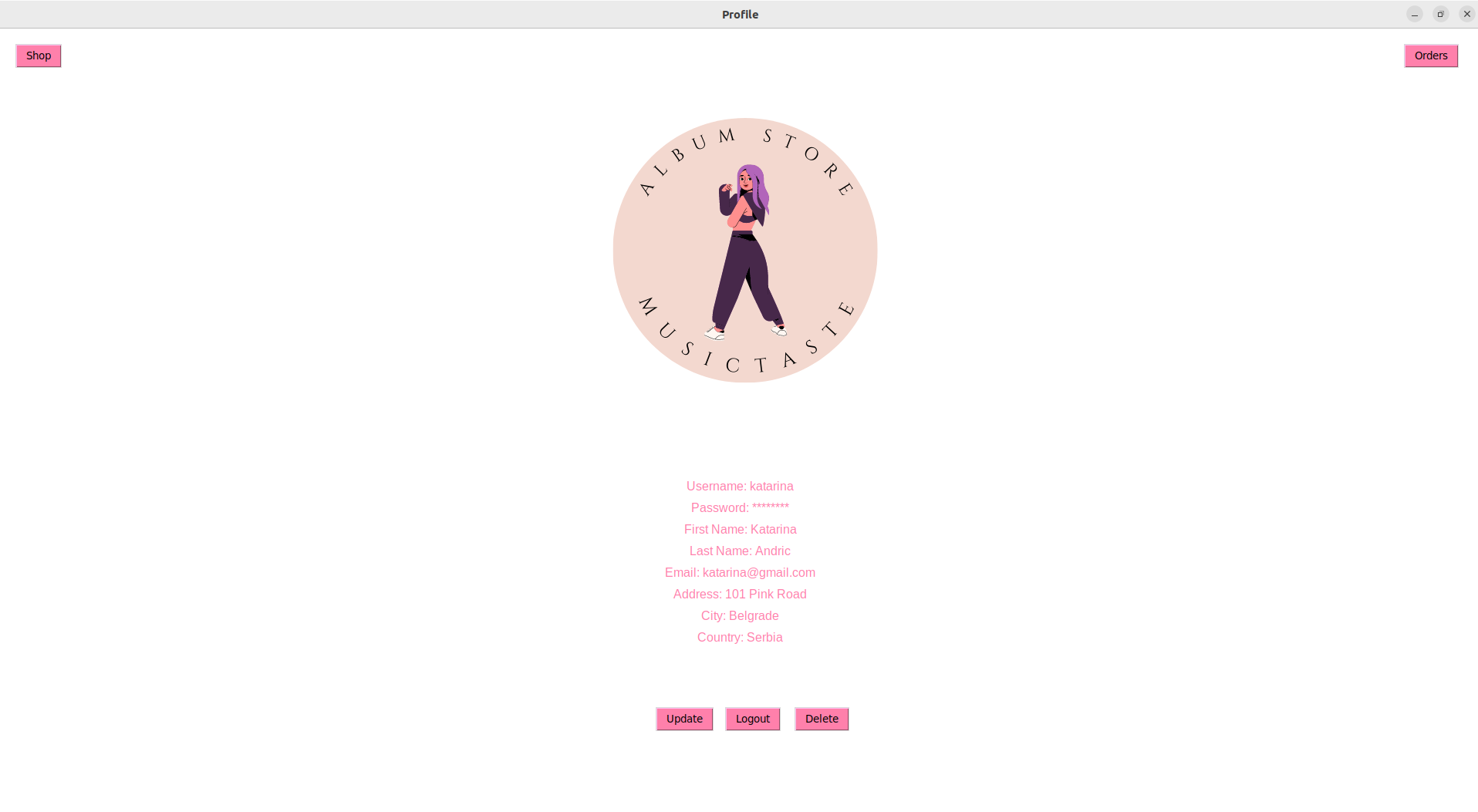


Рисунок 10 – Профиль



Рисунок 11 – Интерфейс менеджера



Рисунок 12 – Интерфейс администратора

**3.4 Способы и этапы тестирования**

Для проверки функциональности программного обеспечения будет использоваться функциональное тестирование.

Этапы тестирования:

* регистрация пользователя;
* авторизация пользователя;
  + попытка авторизоваться с неправильным именем пользователя или паролем;
* покупка альбома (доступный, под-заказ и предзаказ);
  + попытка купить альбом незарегистрированным пользователем;
* просмотр предыдущих заказов пользователем и менеджером;
* изменение и отмена заказа;
  + попытка изменить или удалить заказ, который не был сделан в текущий или предыдущий день;
* добавление, редактирование и удаление альбома;
  + попытка добавить существующий альбом.

**3.5 Вывод**

В данном разделе были выполнены следующие задачи:

1. Выбор СУБД: Был выбран PostgreSQL в качестве системы управления базами данных. Этот выбор основан на открытом исходном коде, свободной лицензии, мощной функциональности, надёжности, стабильности и расширяемости данной СУБД. В сравнении с другими популярными реляционными СУБД, PostgreSQL обладает схожим функционалом, но предлагает большую гибкость настройки и имеет активное сообщество разработчиков и пользователей.
2. Выбор языка программирования и среды программирования: В качестве языка программирования был выбран Python. Этот выбор обусловлен его простотой и читаемостью кода, а также наличием богатой экосистемы библиотек. Для разработки графического интерфейса была использована библиотека tkinter. В качестве среды разработки был выбран Visual Studio Code, благодаря его мощным функциональным возможностям и полноценной поддержке языка программирования Python.
3. Способы и этапы тестирования: Для проверки функциональности программного обеспечения будет использоваться функциональное тестирование. Этапы тестирования включают регистрацию и авторизацию пользователей, покупку альбомов, просмотр предыдущих заказов, изменение и отмену заказов, добавление, редактирование и удаление альбомов. Это позволит проверить работоспособность различных функций программы и обнаружить возможные ошибки или недочёты.

В результате выбора PostgreSQL в качестве СУБД, Python в качестве языка программирования, библиотеки psycopg2 и tkinter, а также использования Visual Studio Code в качестве среды разработки, были обеспечены гибкость, удобство и эффективность в разработке программного продукта. Тестирование функциональности программы позволит удостовериться в её правильной работе и соответствии требованиям проекта.

**4 Исследовательская часть**

**4.1 Цель эксперимента**

Цель эксперимента заключается в определении влияния числа записей в таблице и типа запроса на время выполнения. Задачи для достижения цели:

* выбор запросов;
* измерение времени выполнения каждого запроса в зависимости от количества записей в таблице;
* построение графика взаимосвязи между временем выполнения запросов и количеством записей;
* анализ результатов.

Технические характеристики устройства:

* операционная система: Ubuntu 22.04.2 LTS
* оперативная память: 9 GB
* процессор: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz

**4.2 Постановка эксперимента**

В ходе эксперимента были использованы два вида запросов.

Первый вид:

|  |
| --- |
| SELECT oa.ordalbid, o.orderid, o.orderdate, o.ordertype, oa.username, a.albumname, a.albumver, g.pcname  FROM ordalb oa  INNER JOIN orders o ON oa.orderid = o.orderid  INNER JOIN apg p ON oa.apg = p.apgid  INNER JOIN album a ON p.albumid = a.albumid  INNER JOIN gift g ON p.giftid = g.giftid |

Второй вид:

|  |
| --- |
| SELECT \*  FROM ordalb oa  INNER JOIN orders o ON oa.orderid = o.orderid  INNER JOIN apg p ON oa.apg = p.apgid  INNER JOIN album a ON p.albumid = a.albumid  INNER JOIN gift g ON p.giftid = g.giftid |

После каждых 100 измерений записей было вычислено среднее время выполнения, а потом количество записей увеличилось за 1000, до 10000. Время измерялось с использованием функции time (библиотека time).

**4.3 Результат эксперимента**

Результаты эксперимента представлены в таблицах 2 и 3. В таблице 2 приведены результаты измерений для первого запроса, где указано время выполнения для каждого количества записей. Аналогично, в таблице 3 представлены результаты измерений для второго запроса.

Таблица 2 – Результаты измерения первого запроса

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ко-личе-ство запи-сей | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10  000 |
| Вре-мя | 0.009931 | 0.011121 | 0.014892 | 0.018479 | 0.021430 | 0.025890 | 0.037692 | 0.044029 | 0.048321 | 0.051124 |

Таблица 3 – Результаты измерения второго запроса

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ко-личе-ство запи-сей | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10  000 |
| Вре-мя | 0.019872 | 0.024128 | 0.028439 | 0.033996 | 0.039122 | 0.045900 | 0.059679 | 0.067951 | 0.078347 | 0.083025 |

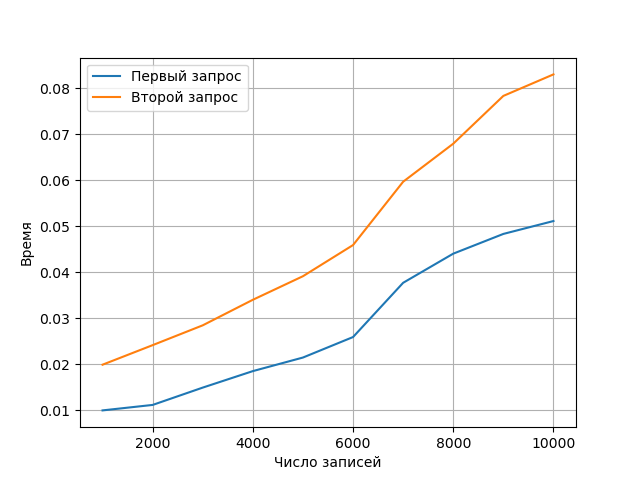


Рисунок 13 – Взаимосвязь между временем выполнения запросов и количеством записей

**4.4 Вывод**

В итоге было обнаружено, что время выполнения напрямую зависит от количества записей. Кроме того, было выяснено, что первый тип запроса выполняется быстрее второго. Исходя из полученных данных, можно сделать вывод, что как количество записей в таблице, так и тип запроса влияют на время выполнения запроса.

**ЗАКЛЮЧЕНИЕ**

Цель курсовой работы, проектирование и реализация магазина альбомов с помощью Python и PostgreSQL, достигнута.

В процессе работы были выполнены задачи:

* проведения анализа существующих СУБД;
* выбора набора технологий для реализации приложения;
* разработки структуры базы данных для хранения и организации данных;
* проектирования и разработки приложения для взаимодействия с базой данных;
* анализа времени выполнения различных запросов к базе данных в зависимости от количества записей в таблицах.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Диаграмма ER в СУБД [Электронный ресурс]. – Режим доступа: https://coderlessons.com/tutorials/bazy-dannykh/osnovy-subd/5-diagramma-er-v-subd#3. (Дата обращения: 23.03.2023).

2. СУБД (система управления базами данных) – виды, классификация и назначение [Электронный ресурс]. – Режим доступа: https://nauka.club/informatika/subd.html (Дата обращения: 15.03.2023).

3. Сравнение SQL и NoSQL: как выбрать систему хранения данных [Электронный ресурс]. – Режим доступа: https://mcs.mail.ru/blog/sravnenie-sql-i-nosql-kak-vybrat-sistemu-hraneniya-dannyh (Дата обращения: 15.03.2023)

4. Зачем нужны нереляционные базы данных: история появления и развития [Электронный ресурс]. – Режим доступа: https://www.bigdataschool.ru/wiki/nosql (Дата обращения: 15.03.2023)

5. database management system (DBMS) [Электронный ресурс]. – Режим доступа: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (Дата обращения: 23.03.2023)

6. Yogesh Rana. Python: Simple though an Important Programming language // International Research Journal of Engineering and Technology (IRJET), 2019.

**ПРИЛОЖЕНИЕ А**

Листинг 1 – Подключение к различным ролям в базе данных.

|  |
| --- |
| def get\_conn(username):  if username == 'manager':  connection = psycopg2.connect(  host="localhost",  database="postgres",  user="manager",  password="\*\*\*\*\*\*\*"  )  elif username == 'admin':  connection = psycopg2.connect(  host="localhost",  database="postgres",  user="admin",  password="\*\*\*\*\*"  )  else:  connection = psycopg2.connect(  host="localhost",  database="postgres",  user="user",  password="\*\*\*\*"  )  return connection |

**ПРИЛОЖЕНИЕ Б**

Листинг 2 – SQL-запрос на обновление даты заказа

|  |
| --- |
| UPDATE Orders SET orderdate = %s  WHERE orderid = (  SELECT orderid  FROM ordalb  WHERE ordalbid = %s) |

Листинг 3 – SQL-запрос для получения идентификаторов подарков на основе идентификатора альбома

|  |
| --- |
| SELECT g.pcname  FROM gift g  INNER JOIN apg a ON g.giftid = a.giftid  WHERE a.albumid = %s |

Листинг 4 – SQL-запрос для получения информации об альбоме и подарке на основе идентификатора продукта

|  |
| --- |
| SELECT a.albumid, a.albumname, a.albumver, a.price, a.qleft, g.pcname  FROM apg ag  INNER JOIN album a ON ag.albumid = a.albumid  INNER JOIN gift g ON ag.giftid = g.giftid  WHERE ag.apgid = %s |

**ПРИЛОЖЕНИЕ В**

Листинг 5 – Процедура уменьшения количества доступных альбомов и увеличения количества купленных альбомов

|  |
| --- |
| CREATE OR REPLACE PROCEDURE update\_album\_qbought(apgid\_val integer)  AS $$  BEGIN  UPDATE album  SET qbought = qbought + 1, qleft = qleft - 1  WHERE albumid IN (SELECT albumid FROM apg WHERE apgid = apgid\_val);  END;  $$ LANGUAGE plpgsql; |

Листинг 6 – Процедура увеличения количества купленных альбомов

|  |
| --- |
| CREATE OR REPLACE PROCEDURE update\_podorder\_qbought(apgid\_val integer)  AS $$  BEGIN  UPDATE album  SET qbought = qbought + 1  WHERE albumid IN (SELECT albumid FROM apg WHERE apgid = apgid\_val);  END;  $$ LANGUAGE plpgsql; |

**ПРИЛОЖЕНИЕ Г**

Листинг 7 – Применение процедур

|  |
| --- |
| for apg\_id in apg\_ids:  cur.execute("CALL update\_album\_qbought(%s)", (apg\_id,))  for addod in podorder:  cur.execute("CALL update\_podorder\_qbought(%s)", (addod,)) |