Аннотация

Курсовая работа на тему: «Разработка базы данных магазина алкогольной продукции». В ходе выполнения данной курсовой работы была спроектирована база данных, хранящая информацию об алкогольной продукции, брендах, заводах-производителях и регионах произрастания винограда, пользователях и их корзинах, а также приложение, предоставляющее пользователям графический интерфейс для доступа к разработанной базе данных.

Приложение было спроектировано с использованием языка программирования Python 3.7.0 и с помощью среды разработки Visual Studio Code 1.71.0. В качестве системы управления базой данных использовалась широко распространенная реляционная СУБД PostgreSQL.

Расчетно-пояснительная записка к курсовой работе содержит 37 страниц, 11 рисунков, 5 таблиц, 6 источников.

Содержание

Введение

Как известно, на сегодняшний день алкогольная продукция занимает огромную нишу на рынке производства и продажи продуктов питания. Среднестатистический выпивающий россиянин в год покупает восемь ящиков пива, двадцать бутылок водки, восемь бутылок вина, две бутылки коньяка и банку джин-тоника [1].

По данным Всемирной Организации Здравоохранения, лишь 27% россиян старше 15 лет никогда не пили и не собираются, а 15% уже бросили. Остальные 58% хотя бы раз в год употребляют алкогольные напитки, причем 60% из пьющих подвержены так называемому «тяжелому эпизодическому пьянству» [2].

Приняв во внимание все вышеуказанное, можно сделать вывод, что перед лицами, реализующими алкогольную продукцию на территории Российской Федерации, стоит задача продажи алкоголя в огромных количествах.

Для решения подобной проблемы на помощь может прийти специально разработанное программное обеспечение, которое поможет эффективно реализовывать на рынке алкогольную продукцию, при этом быть максимально удобным и для пользователя.

Цель работы – разработать базу данных, в полной мере предоставляющею интерфейс магазина алкогольной продукции.

Для достижения поставленной цели необходимо решить следующие задачи:

- четко сформулировать задачу, определить необходимый функционал;
- проанализировать существующие СУБД и выбрать наиболее подходящую для решения поставленной задачи;
- описать структуру базы данных алкогольной продукции;
- спроектировать описанную базу данных;

- разработать приложение с графическим пользовательским интерфейсом, позволяющее в полной мере взаимодействовать с реализуемой базой данных.

1 Аналитический раздел

В данном разделе будет четко поставлена задача, формализованы данных, которые в дальнейшем будут храниться в базе данных. Будет представлена ER-диаграмма проектируемой базы данных, произведена категоризация пользователей и алкогольной продукции.

Также будут классифицированы и рассмотрены различные системы управления базами данных, выявлены их преимущества и недостатки. Будет произведен выбор типа СУБД для решения поставленной задачи.

1.1 Постановка задачи

Необходимо разработать полноценную базу данных для хранения информации о широком спектре алкогольной продукции, а также разработать ПО, которое будет предоставлять полноценный пользовательский интерфейс магазина алкогольной продукции, взаимодействуя с разрабатываемой базой данных. Данный вид приложения в сочетании с размещением базы данных на сервере, может использовать данные базы данных, осуществляя подключение к базе данных и последующую выгрузку необходимых пользователю для работы данных.

При этом, ПО должно предоставлять отдельные опции пользовательского интерфейса каждому из имеющихся типов пользователей.

1.2 Анализ типа программного обеспечения

Разрабатываемое программное обеспечение будет представлено в виде клиентного Desktop-проложения. Desktop-приложение – это программное обеспечение, обработка данных которого осуществляется на стороне клиента, т.е. пользователя. Само ПО представлено в виде исполняемого файла или скрипта непосредственно на устройстве пользователя, персональном компьютере,

смартфоне или другом подобном устройстве.

Преимущество данного типа ПО заключается в его безграничной гибкости в реализации полноценного графического пользовательского интерфейса, что вносит львиную долю для восприятия пользователем приложения, ведь чаще всего обычные люди воспринимают программное обеспечение с красивым, дружелюбным для пользователя интерфейсом как более надежное и эффективное ПО. К тому же использование данного типа программного обеспечения позволяет команде разработчиков или разработчику не тратить дополнительные денежные ресурсы на аренду сервера, как, например, в случае с Web-приложением.

Для решения поставленной задачи как нельзя кстати подходит именно Desktop-приложение, т.к. перед программным обеспечением поставлены задачи доступа к объемной по структуре и информации базе данных, что будет крайне неудобно без графического пользовательского интерфейса.

1.3 Формализация данных

Разрабатываемая база данных должна содержать информацию о брендах, заводах-производителях, регионах, пользователях и их корзинах, самой алкогольной продукции. Корзины пользователей и информация о пользователях не входят в ER-диаграмму.

Ниже, на рисунке 1.1, представлена ER-диаграмма разрабатываемой базы данных.

Таблица 1.1 – Классификация сведений о данных

Данные	**Сведения**
Бренд	Название, страна, год основания
Завод-производитель	Название, страна, город, год основания
Регион	Название, страна
Пользователь	Имя, логин, пароль, тип
Алкоголь	Название, категория, тип, бренд, завод-производитель, крепость, дата розлива, цена, объем, количество в наличии, тара, фильтрация, регион, сорт, год урожая, сахар
Корзина	Алкоголь, количество в корзине

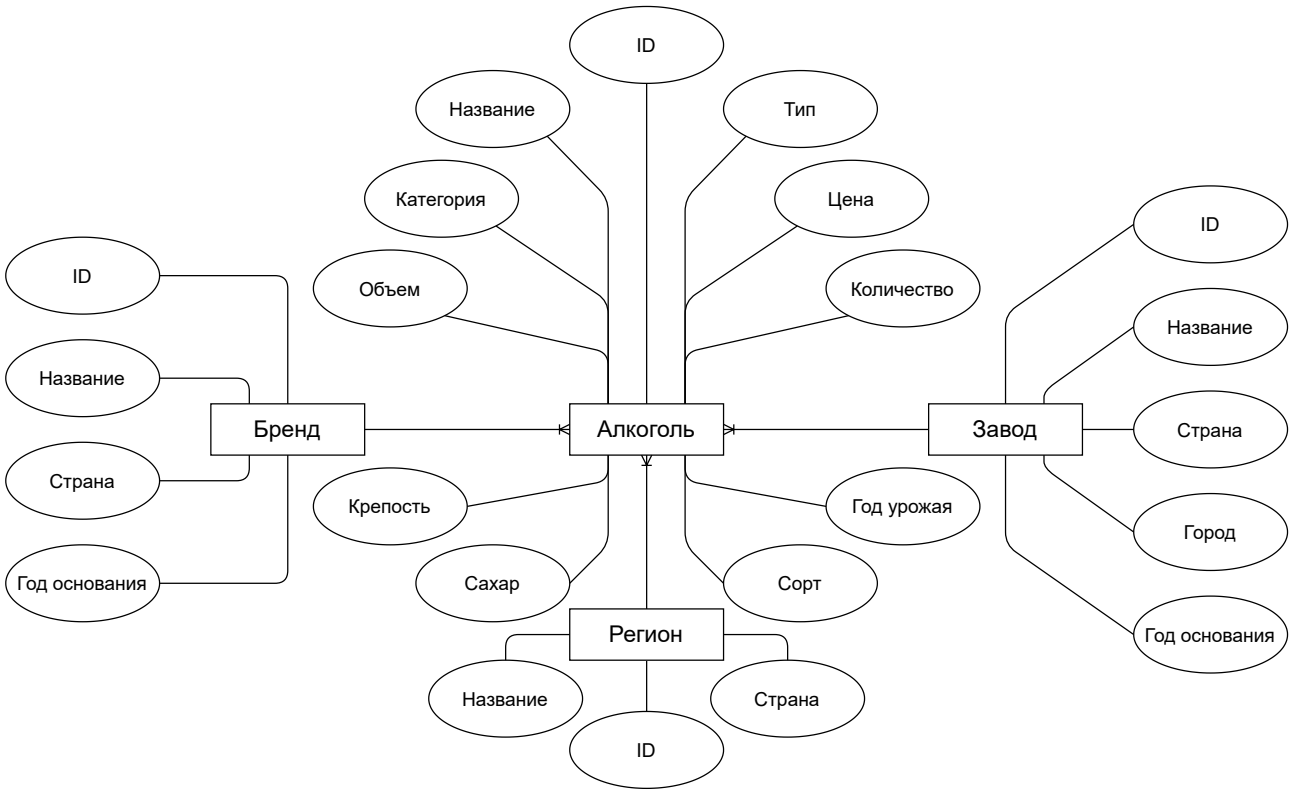

Рисунок 1.1 – ER-диаграмма

1.4 Категории пользователей и алкоголя

Для взаимодействия с приложением пользователям необходима авторизация. Исходя из этого, в системе будут присутствовать как авторизованные, так и неавторизованные (анонимы) пользователи. При этом, единственным

вариантом для неавторизированных пользователей будет являться авторизация (если сведения о них есть в системе), либо же непосредственно регистрация с последующим внесением пользователя в базу данных. После прохождения этапа авторизации или регистрации пользователю присваивается одна из имеющихся категорий пользователя (покупатель, менеджер продаж, главный менеджер) и предоставляется доступ к программному обеспечению посредством пользовательского интерфейса с учетом роли пользователя в системе.

Ниже, в таблице 1.2, представлена категоризация пользователей, содержащая информацию о типе роли пользователя и доступных данному типу пользователей действиях в приложении.

Таблица 1.2 – Пользователи и привилегии

Роль пользователя	**Привилегии**
Аноним	Регистрация/Авторизация.
Покупатель	Просмотр информации об алкогольной продукции. Взаимодействие с корзиной (добавление, удаление).
Менеджер продаж	Просмотр информации об алкогольной продукции. Взаимодействие с корзиной (добавление, удаление). Изменение цены товара и его наличия на складе.
Главный менеджер	Просмотр информации об алкогольной продукции. Взаимодействие с корзиной (добавление, удаление). Изменение цены товара и его наличия на складе. Добавление новых данных (алкоголь, бренд, ...).

Исходя из представленных данных в таблице 1.2, на рисунке 1.2 представлена Use-Case диаграмма, содержащая информацию действиях, доступных каждой категории пользователей.

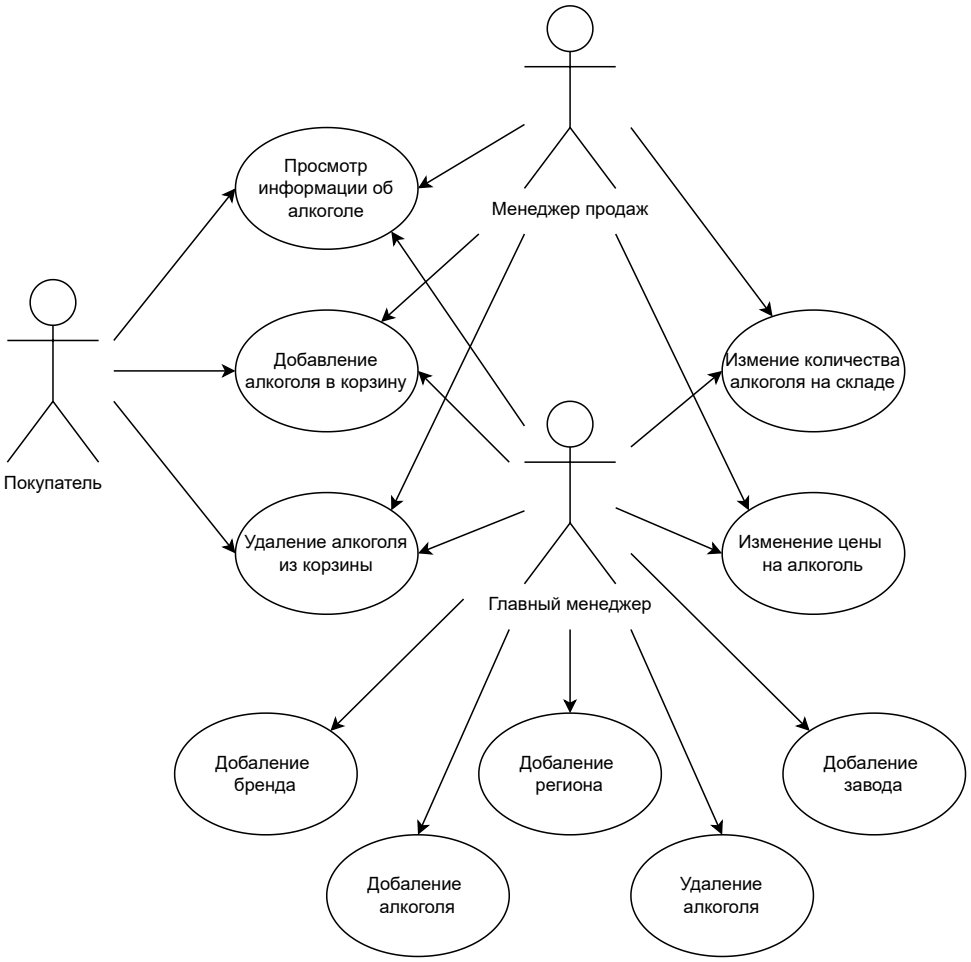

Рисунок 1.2 – Use-Case диаграмма

В таблице 1.3 приведены имеющиеся в базе данных категории и типы алкогольной продукции.

Таблица 1.3 – Категории и типы алкогольной продукции

Категория	**Типы**
Вино	Красное, белое, розовое, вермут, портвейн
Игристое	Шампанское, игристое, просекко, асти, ламбруско
Крепкий алкоголь	Ром, виски, коньяк, ликер, водка, текила, джин, абсент, саке
Пиво	Светлое, темное, сидр, пивной напиток

1.5 Базы данных

База данных (далее – БД) – набор данных, организованных определенным образом, предполагающий общеопределенные принципы описания, хранения и способы взаимодействия с данными.

К любой БД выдвигаются следующие требования:

- неизбыточность;
- эффективность доступа;
- совместное использование;
- безопасность;
- восстановление после сбоя;
- целостность;
- независимость от сторонних приложений.

1.6 СУБД

СУБД (система управления базами данных) – это специальное программное обеспечение для проектирования и создания баз данных, их непосредственного изменения, и получения из них определенной информации; программный комплекс, позволяющий администрировать базу данных, защищать ее целостность и обеспечивать конфиденциальность хранящихся в ней данных.

Функции СУБД:

- управление данными во внешней памяти и оперативной памяти;
- журнализация, резервное копирование и восстановление целостности информации базы данных после сбоев;
- поддержка множества языков БД.

1.7 Классификация СУБД

Под моделью данных обычно понимают абстрактное, в полной мере самодостаточное, логическое определение объектов, и прочих элементов, составляющих так называемую абстрактную машину доступа к данным, с которой взаимодействует тот или иной пользователь.

Классификация СУБД по модели данных:

- дореляционные;
- реляционные;
- постреляционные.

Классификация дореляционных СУБД:

- инвертированные списки;
- иерархические;
- сетевые.

1.7.1 Иерархическая модель

Иерархическая модель данных используется непосредственно для проектирования баз данных с древовидной иерархической структурой. В данной модели используется ориентация представления данных в виде древовидной структуры от корня к листьям. Каждый так называемый лист дерева модели хранит экземпляры сущностей (объекты) – записи.

К преимуществам использования иерархической модели данных для проектирования баз данных можно отнести простоту работы с моделью, а также высокую скорость доступа к данным посредством определенных шаблонов хранения информации.

Основной и пожалуй самый большой недостаток в работе с данной моделью данных заключается в многократном дублировании информации. Подобный аспект возникает в силу того, что каждая придаточная сущность (атрибут) может иметь отношение исключительно к одной родительской сущности.

1.7.2 Сетевая модель

Сетевая модель позволяет проектировать базы данных с графовой структурой общего вида. Каждый узел графа данной модели хранит представления сущностей и данные об отношениях с экземплярами других типов. Каждая запись хранит произвольный набор значений атрибутов, характеризующих представление сущности. Связи между записями в сетевой модели данных построены на указателях, в частности, каждая запись хранит у себя ссылку на другую однотипную запись и ссылки на прилегающие списки подчиненных записей, связывающиеся с ней определенными групповыми отношениями.

Представленная выше модель данных сложна в проектировании и поддержке. В сетевой модели не предусматривается физическая независимость данных. Обусловлено это тем, что наборы определены и организованы с использованием физических ссылок.

Преимуществом же данной модели является ее гибкость в работе с данными, но этот признак может обернуться недостатком, если база данных по своей структуре сильно нагружена таблицами и связями.

1.7.3 Реляционная модель

Реляционная модель данных представляет собой совокупность данных и состоит из конечного набора таблиц. При данной организации данных отсутствует явно наблюдаемая в других моделях данных иерархия. Таблицы представляют собой строки (записи) и столбцы (атрибуты). На пересечении строк и столбцов находятся значения атрибутов записей. Из того, что имеется возможность просмотра строк и столбцов в произвольном порядке и

направлении, достигается гибкость выбора подмножества среди представленных элементов.

Реляционная модель, как известно, самая удобная и наиболее распространенная модель представления данных. Отличается непосредственной независимостью данных, простотой хранения и удобством использования этих данных.

Недостатком данной модели данных является невысокая скорость обработки запросов, в особенности, если запрос сформирован неэффективно.

Ниже, в таблице 1.4 представлен анализ вышеупомянутых моделей данных с указанием их преимуществ и недостатков.

Таблица 1.4 – Сравнение моделей данных

Модель данных	**Преимущества**	**Недостатки**
Иерархическая	Высокая скорость доступа к данным	Многократное дублирование данных
Сетевая	Информационная гибкость	Сложность проектирования
Реляционная	Удобное представление данных, простота в использовании	Невысокая скорость выполнения некоторых запросов

Опираясь на вышеуказанные данные, можно прийти к выводу от том, что реляционная модель хранения данных как нельзя кстати подходит для решения поставленной задачи.

Вывод

В данном разделе была проанализирована поставленная задача и рассмотрены возможные способы для ее реализации. Также были рассмотрены различные типы СУБД, проанализированы их преимущества и недостатки. Для решения поставленной задачи была выбрана реляционная СУБД.

2 Конструкторский раздел

В данном разделе будет рассмотрено проектирование объектов базы данных, в частности таблиц с их полями и их типами данных, триггера и хранимой процедуры.

Для триггера и хранимой процедуры будут представлены схемы их работы, для базы данных будет представлена диаграмма базы данных. Также будут выдвинуты требования к реализуемому программному обеспечению.

2.1 Проектирование базы данных

В данном разделе будет описано проектирование таких элементов базы данных, как таблицы, триггер и процедура. Таблицы будут описаны в соответствии с полями, будут представлены схемы триггера и процедуры.

В описание таблиц войдут названия таблиц, названия полей таблиц с описанием хранимых ими данных, включая тип данных. Также будет представлена диаграмма базы данных.

2.1.1 Таблицы

База данных должна содержать рассмотренные в таблице 1.1 данные:

- таблица алкогольной продукции alcohol;
- таблица заводов brand;
- таблица брендов factory;
- таблица регионов region;
- таблица пользователей users.

Ниже, на рисунке 2.1, представлена диаграмма базы данных.

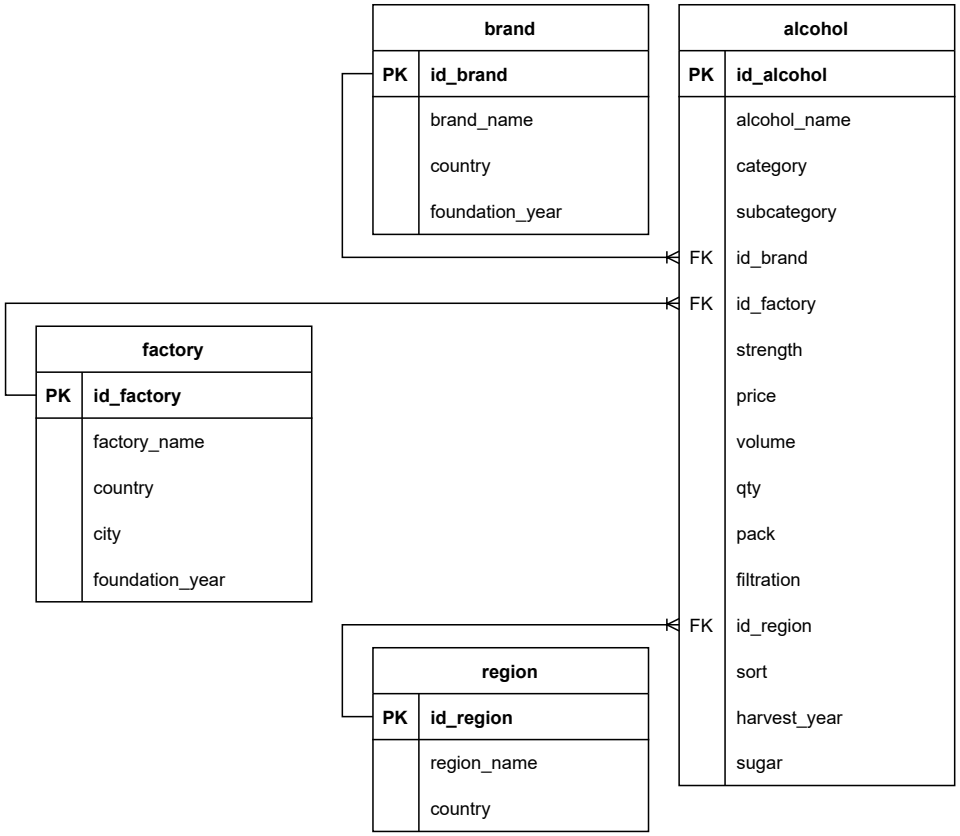

Рисунок 2.1 – Диаграмма базы данных

Таблица **alcohol** содержит информацию об алкогольной продукции:

- id_alcohol – уникальный идентификатор алкоголя, первичный ключ, INT PK;
- alcohol_name – название алкоголя, TEXT;
- category – категория алкоголя, INT;
- subcategory – тип алкоголя, INT;
- id_brand – идентификатор бренда, INT FK;
- id_factory – идентификатор завода-производителя, INT FK;
- strength – крепость (в градусах), INT;
- price – цена (в рублях), FLOAT;

- volume – объем тары, FLOAT;
- qty – количество алкоголя на складе, INT;
- pack – тара (банка, стекло) (для пива), INT;
- filtration – фильтрация (для пива), INT;
- id_region – идентификатор региона (для вин и игристых), INT FK;
- sort – сорт винограда (для вин и игристых), TEXT;
- harvest_year – год урожая (для вин и игристых), INT;
- sugar – содержание сахара (полусладкое, полусухое, сухое, брют) (для вин и игристых), INT.

Таблица **factory** содержит информацию о заводах-производителях:

- id_factory – уникальный идентификатор завода-производителя, первичный ключ, INT PK;
- factory_name – название завода, TEXT;
- country – страна, где находится завод, TEXT;
- city – город, в котором находится, TEXT;
- foundation_year – год основания завода, INT.

Таблица **brand** содержит информацию о брендах:

- id_brand – уникальный идентификатор бренда, первичный ключ, INT PK;
- brand_name – название бренда, TEXT;
- country – страна, в которой основан бренд, TEXT;
- foundation_year – год основания бренда, INT.

Таблица **region** содержит информацию о регионах произрастания винограда:

- id_region – уникальный идентификатор региона, первичный ключ, INT PK;
- region_name – название региона, TEXT;
- country – страна региона, TEXT.

Таблица **users** содержит информацию о пользователях:

- id_user – уникальный идентификатор пользователя, первичный ключ, INT PK;
- user_nick – имя пользователя, TEXT;
- user_login – логин пользователя, TEXT;
- user_password – пароль, TEXT;
- user_grant – роль пользователя, INT.

Таблица корзины пользователя содержит информацию об алкоголе в корзине пользователя:

- id_item – уникальный идентификатор ячейки товара, первичный ключ, INT PK;
- id_alcohol – идентификатор алкоголя, INT;
- qty – количество алкоголя в корзине, INT.

2.1.2 Триггер

Для таблицы users будет реализован специальный триггер на удаление, чтобы при попытке удалить пользователя из таблицы базы данных провоцировало исключение.

Данный триггер необходим для того, чтобы в случае невнимательности модератора защитить таблицу users от удаления из нее пользователей.

Ниже, на рисунке 2.2, представлена схема разрабатываемого триггера.

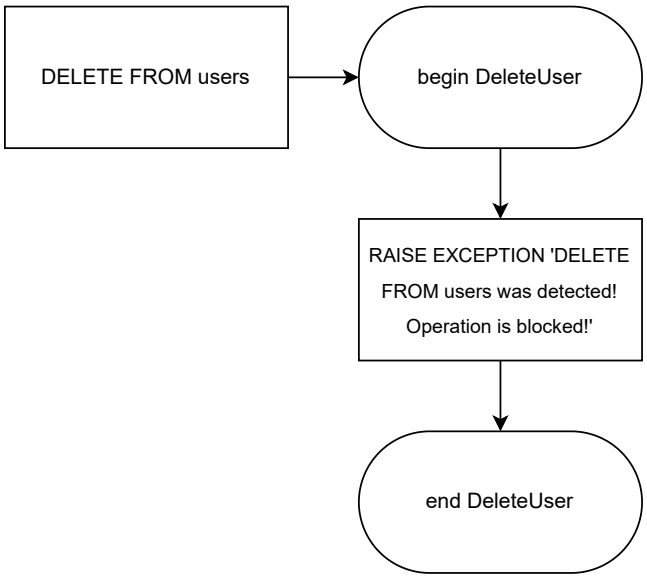

Рисунок 2.2 – Схема триггера

2.1.3 Хранимая процедура

Для корректной отработки изменения количества алкоголя в корзине пользователя будет создана хранимая процедура UpdateCart, позволяющая изменять количество алкоголя как в положительную, так и в отрицательную сторону. При этом, если количество алкоголя в строке меньше либо равно числу алкоголя на удаление из этой строки, произойдет удаление данной строки из корзины пользователя.

Ниже, на рисунке 2.3 представлена схема хранимой процедуры обновления количества алкоголя в корзине пользователя.

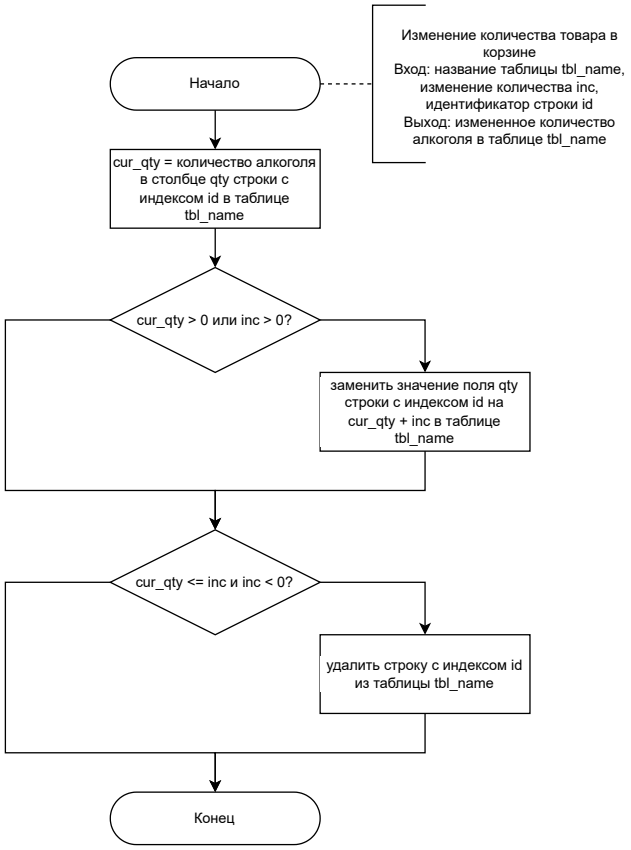

Рисунок 2.3 – Схема хранимой процедуры

2.2 Требования к ПО

Для доступа к спроектированной базе данных необходимо разработать специальное программное обеспечение. ПО должно предоставлять определенные возможности для пользователей.

Возможности для **покупателя**:

- авторизация или регистрация;
- просмотр всей алкогольной продукции;
- просмотр карточки отдельного алкоголя;
- добавление алкоголя в корзину;
- удаление алкоголя из корзины.

Возможности для **менеджера продаж**:

- возможности покупателя;
- изменение цены на продукцию;
- изменение количества продукции на складе.

Возможности для **главного менеджера**:

- возможности покупателя;
- возможности менеджера продаж;
- удаление алкогольной продукции;
- добавление алкогольной продукции;
- добавление региона;
- добавление завода-производителя;
- добавление бренда.

Вывод

В данном разделе было рассмотрено проектирование таких элементов базы данных, как таблицы, триггер и хранимая процедура. Также были обозначены требования к реализуемому программному обеспечению.

3 Технологический раздел

В данном разделе будет обоснован выбор средств реализации программного обеспечения, в частности будет выбран язык программирования, среда разработки, реляционная СУБД, а также библиотека для работы с СУБД.

Также будет рассмотрена структура разрабатываемого программного обеспечения, представлен программный код создания объектов базы данных и графический пользовательский интерфейс программного обеспечения.

3.1 Средства реализации

В качестве **языка программирования** был выбран *Python 3.7.0* [3], так как:

- имеет необходимые библиотеки для работы с различными СУБД;
- имеется много учебной литературы;
- данный язык использовался при выполнении лабораторных работ по курсу «Базы данных», следовательно, имеется определенный опыт работы.

Для работы с СУБД будет использоваться библиотека *psycopg2* [4].

В качестве **среды разработки** был выбран *Visual Studio Code 1.71.0* [5] из следующих соображений:

- имеется опыт работы с данной средой на протяжении нескольких лет, что сократит время изучения возможностей новой среды;
- она довольно гибкая и имеет много настраиваемых параметров для обеспечения комфортного процесса разработки ПО.

В качестве **СУБД** была выбрана *PostgreSQL* [6], исходя из следующих соображений:

- данная СУБД находится в открытом доступе;
- имеется опыт работы с данной СУБД в рамках курса «Базы данных»;
- удовлетворяет всем требованиям, необходимым для решения поставленной задачи;
- предоставляет обширный функционал для работы с базами данных (процедуры, функции, триггеры и т.п.).

3.2 Структура ПО

Программное обеспечение разрабатывается с использованием структурного подхода.

На рисунке 3.1 представлена функциональная структура ПО.

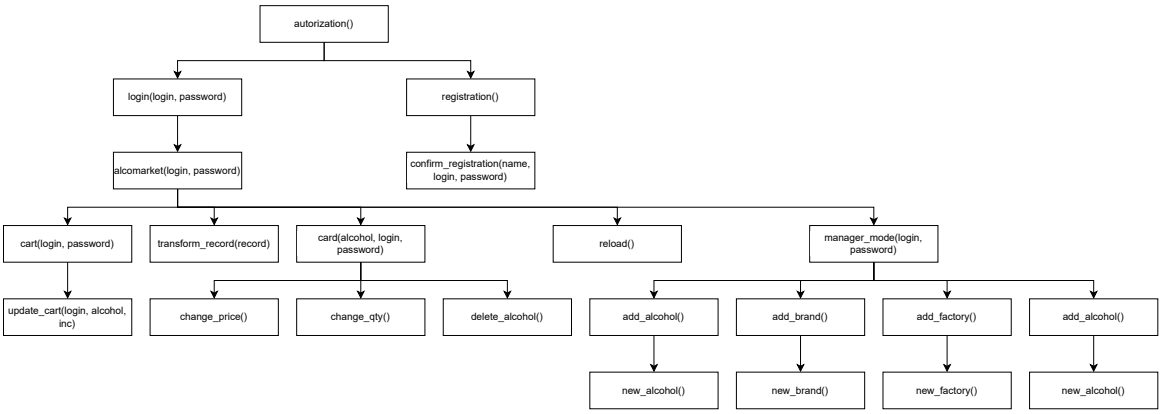

Рисунок 3.1 – Функциональная структура ПО

3.3 Сведения о модулях ПО

Реализованное ПО состоит из следующих модулей:

- *alcomarket.py* – содержит главную функцию и интерфейс главного рабочего экрана;

- *autorization.py* – содержит функции для регистрации и авторизации пользователей и соответствующий интерфейс;
- *connection.py* – содержит функции для подключения к базе данных;
- *requester.py* – содержит функции для выполнения запросов к базе данным;
- *card.py* – содержит функцию для просмотра карточки алкогольной продукции и соответствующий интерфейс;
- *cart.py* – содержит функции для работы с корзиной пользователя и соответствующий интерфейс;
- *manager.py* – содержит функции для основных возможностей главного менеджера и соответствующий интерфейс.

3.4 Создание объектов базы данных

В данном разделе будет описано создание объектов базы данных алкогольной продукции, в частности таблиц, ролевой модели, триггера на удаление и хранимой процедуры.

Будут представлены подробные листинги для создания каждой таблицы базы данных, будет рассмотрен листинг создания ролевой модели, создание процедуры, триггера на удаление и функции-реакции для триггера.

3.4.1 Создание таблиц

В данном разделе представлено создание таблиц базы данных. Для выполнения поставленной задачи необходимо создать таблицы.

В листинге 3.1 представлен код для создания таблиц базы данных.

Листинг 3.1 – Создание таблиц базы данных

```
CREATE TABLE brand(
    id_brand INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    brand_name TEXT NOT NULL UNIQUE,
    country TEXT NOT NULL,
    product INT NOT NULL,
    foundation_year INT NOT NULL
);

CREATE TABLE factory(
    id_factory INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    factory_name TEXT NOT NULL UNIQUE,
    country TEXT NOT NULL,
    city TEXT NOT NULL,
    foundation_year INT NOT NULL
);

CREATE TABLE region(
    id_region INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    region_name TEXT NOT NULL UNIQUE,
    country TEXT NOT NULL
);

CREATE TABLE alcohol(
    id_alcohol INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    alcohol_name TEXT NOT NULL,
    category INT NOT NULL,
    subcategory INT NOT NULL,
    id_brand INT NOT NULL,
    id_factory INT NOT NULL,
    strength INT NOT NULL,
    price FLOAT NOT NULL,
    volume FLOAT NOT NULL,
    qty INT NOT NULL,
    pack INT, filtration INT,
    id_region INT,
    sort TEXT,
    harvest_year INT,
    sugar INT
);

CREATE TABLE users(
    id_user INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    user_nick TEXT NOT NULL,
    user_login TEXT NOT NULL UNIQUE,
    user_password TEXT NOT NULL,
    user_grant INT NOT NULL
);
```

В листинге 3.2 представлен код изменения таблиц, в частности, назначение вторичных ключей и проверка некоторых данных.

Листинг 3.2 – Изменение таблиц базы данных

```
ALTER TABLE brand
    ADD CHECK (foundation_year > 0);

ALTER TABLE factory
    ADD CHECK (foundation_year > 0);

ALTER TABLE alcohol
    ADD CHECK (category > 0),
    ADD CHECK (subcategory > 0),
    ADD CHECK (strength > 0),
    ADD CHECK (price > 0),
    ADD CHECK (volume > 0),
    ADD CHECK (qty >= 0),
    ADD CHECK (harvest_year > 1900),
    ADD FOREIGN KEY (id_region) REFERENCES region(id_region),
    ADD FOREIGN KEY (id_brand) REFERENCES brand(id_brand),
    ADD FOREIGN KEY (id_factory) REFERENCES factory(id_factory);

ALTER TABLE users
    ADD CHECK (user_grant BETWEEN 1 AND 3);
```

3.4.2 Создание ролевой модели

Для возможности пользователей работать с базой данных необходима ролевая модель. Будут созданы роли пользователей с минимальными правами доступа, после чего они будут наделены правами манипуляций с данными.

В листинге 3.3 представлен код создания ролевой модели.

Листинг 3.3 – Создание ролевой модели

```
CREATE ROLE "buyer" NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE
   NOREPLICATION;
CREATE ROLE "manager" NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE
   NOREPLICATION;
CREATE ROLE "general_manager" NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE
   NOREPLICATION;
```

В листинге 3.4 описана выдача прав категориям пользователей.

Листинг 3.4 – Выдача прав для роли пользователя

```
GRANT SELECT ON TABLE alcohol, brand, factory, region TO GROUP "buyer", "
   manager", "general_manager";
GRANT UPDATE ON TABLE alcohol TO GROUP "manager", "general_manager";
GRANT DELETE ON TABLE alcohol TO GROUP "general_manager";
GRANT INSERT ON TABLE brand, factory, region TO GROUP "general_manager";
```

3.4.3 Создание триггера

Для триггера на удаление из таблицы users создается специальная триггерная функция Reaction. Данная функция возвращает исключение при попытке удаления пользователя из таблицы users.

В листинге 3.5 представлено создание функции-реакции на срабатывание триггера.

Листинг 3.5 – Создание функции-реакции для триггера

```
CREATE OR REPLACE FUNCTION Reaction() RETURNS TRIGGER
AS
$$
BEGIN
    IF TG_NAME = 'deleteuser' THEN
    RAISE EXCEPTION 'DELETE FROM users was detected! Operation is blocked!';
    END IF;
RETURN NEW;
END;
$$
LANGUAGE plpgsql;
```

В листинге 3.6, представлено создание триггера на удаление из таблицы users.

Листинг 3.6 – Создание триггера на удаление из users

```
CREATE TRIGGER DeleteUser
BEFORE DELETE
ON
users
FOR EACH ROW
EXECUTE PROCEDURE Reaction();
```

3.4.4 Создание хранимой процедуры

Как было указано выше, для корректной отработки изменения количества алкогольной продукции создается специальная хранимая процедура.

В листинге 3.7 представлен код для создания хранимой процедуры.

Листинг 3.7 – Создание хранимой процедуры

```
CREATE PROCEDURE UpdateCart (tbl_name text, inc INT, id INT)
AS
$$
    DECLARE cur_qty INT;
BEGIN
    EXECUTE format('SELECT qty FROM %s WHERE id_item = %s', tbl_name, id)
   INTO cur_qty;
    IF cur_qty > 1 OR inc > 0 THEN
        EXECUTE format('UPDATE %s SET qty = qty + %s WHERE id_item = %s',
   tbl_name, inc, id);
    END IF;
    IF cur_qty <= inc AND inc < 0 THEN
        EXECUTE format('DELETE FROM %s WHERE id_item = %s', tbl_name, id);
    END IF;
END;
$$
LANGUAGE plpgsql;
```

3.5 Интерфейс программы

Интерфейс главного меню программы представлен на рисунке 3.2.

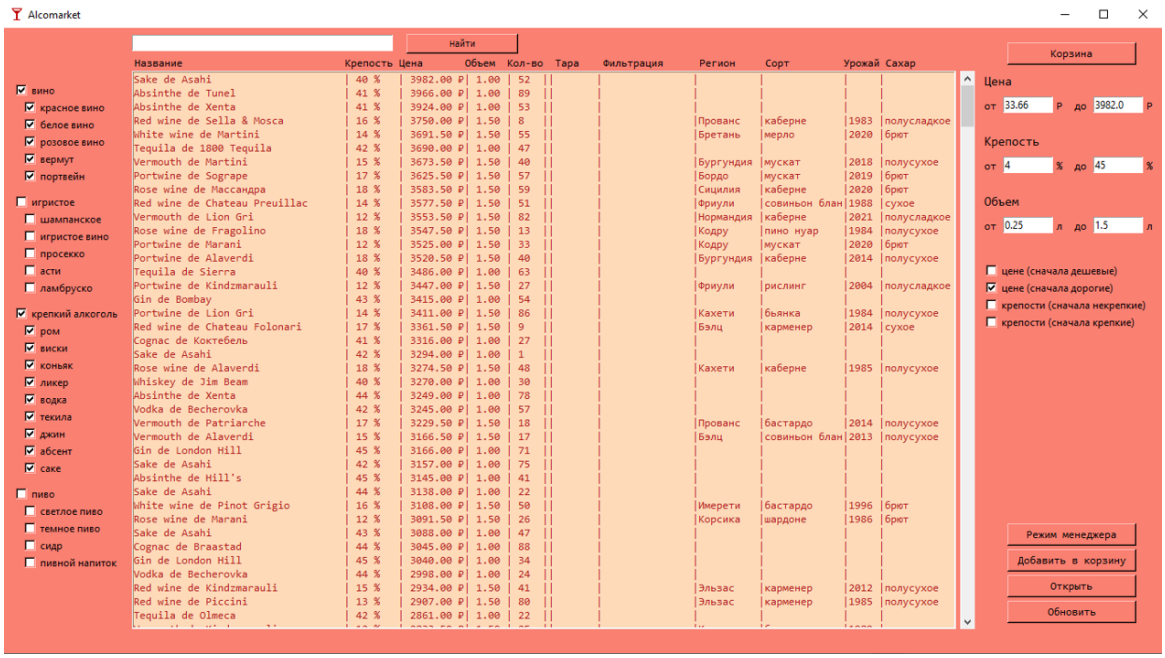

Рисунок 3.2 – Главное меню

В режиме главного менеджера в главном меню доступна кнопка «Режим менеджера», в других режимах она недоступна.

На рисунке 3.3 представлен интерфейс окна авторизации пользователя, а на рисунке 3.4 окно регистрации пользователя.

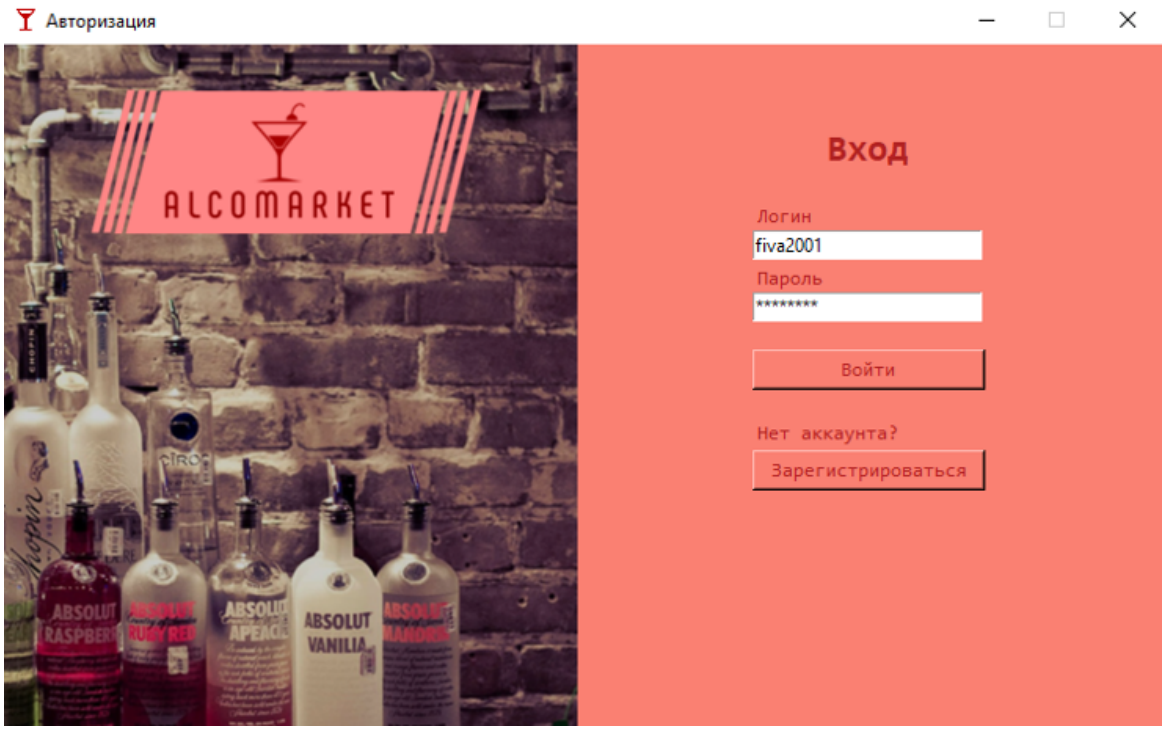

Рисунок 3.3 – Окно авторизации

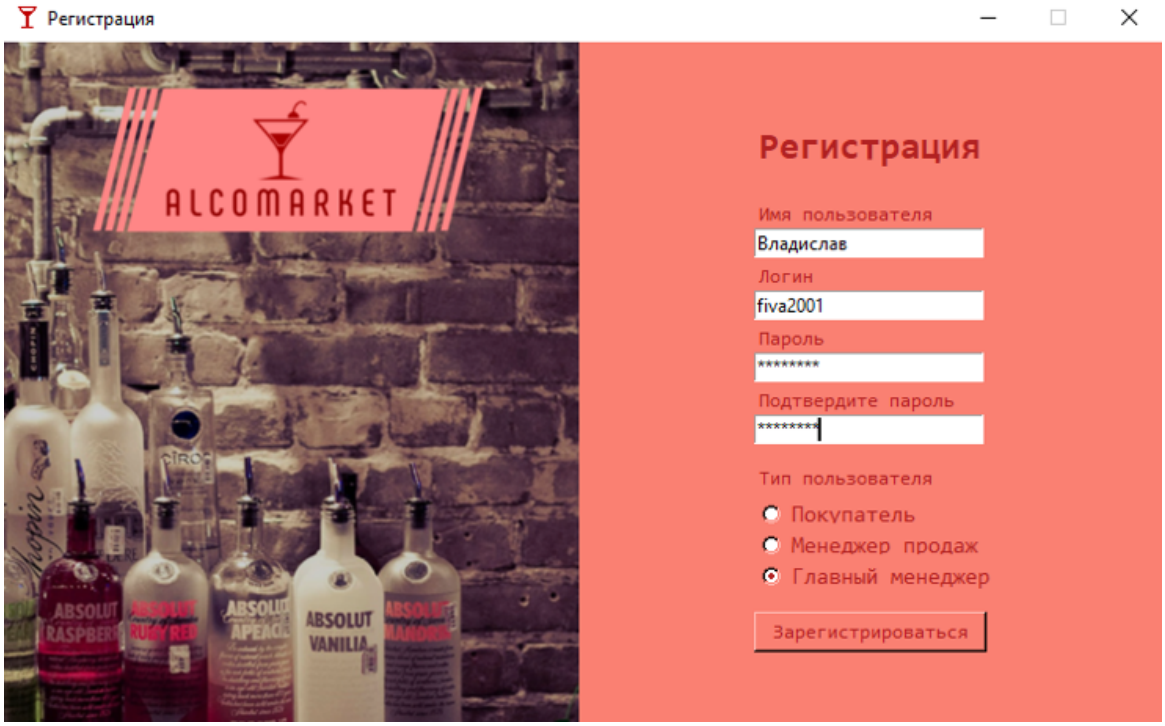

Рисунок 3.4 – Окно регистрации

На рисунке 3.5 представлено окно карточки алкогольной продукции. В окне карточки алкоголя для менеджера продаж доступно изменение цены продукции и его количества на складе кнопками «Изменить», а в режиме главного менеджера доступна кнопка «Удалить алкоголь». Простым покупателям данные действия недоступны.

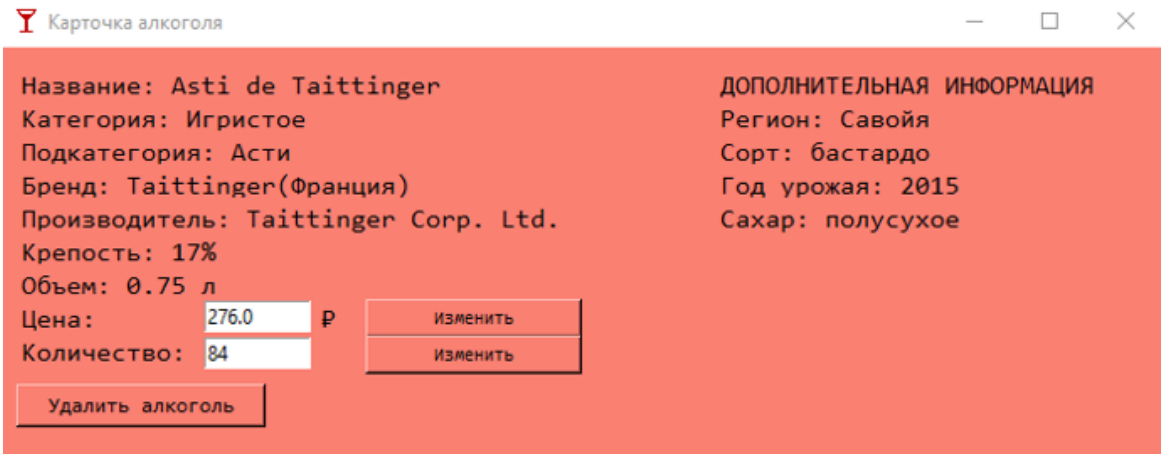

Рисунок 3.5 – Карточка алкогольной продукции

На рисунке 3.6 представлен функционал главного менеджера на добавление алкоголя в базу данных.

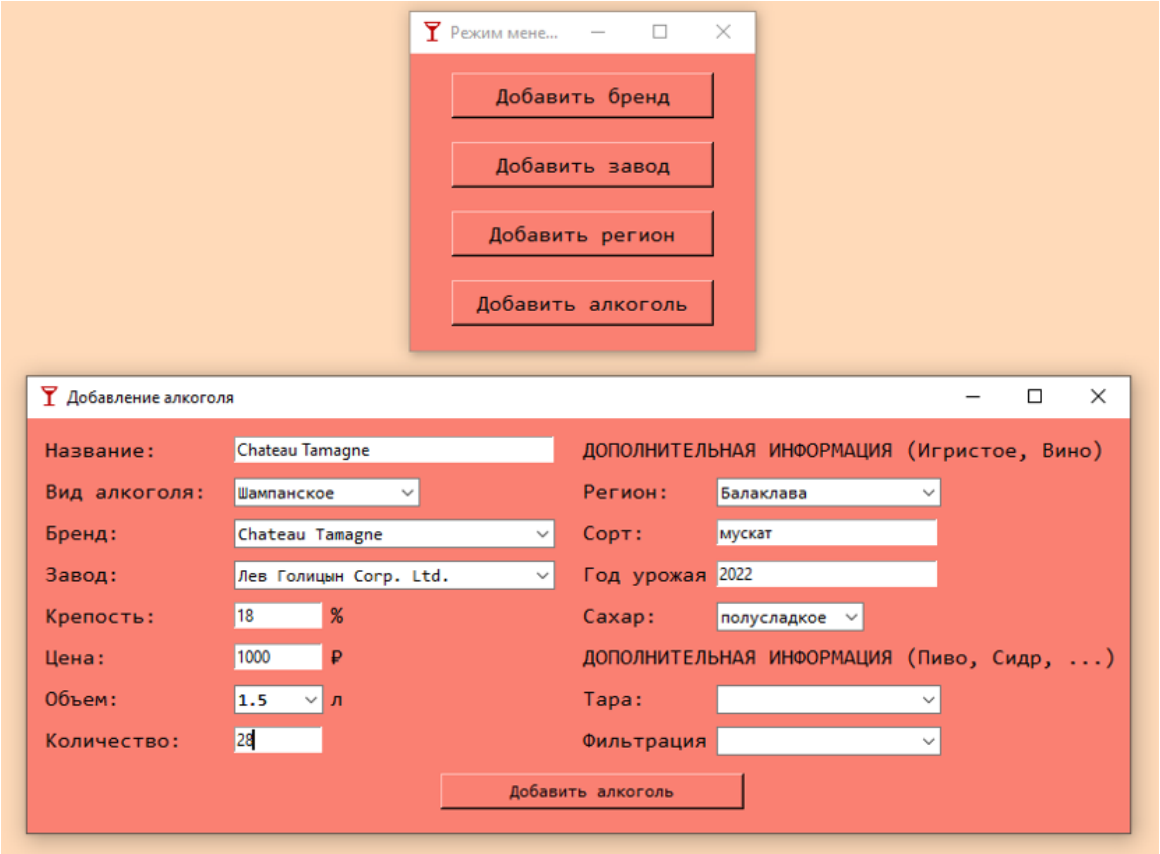

Рисунок 3.6 – Функционал главного менеджера

Вывод

В данном разделе были выбраны средства реализации: язык программирования Python, среда разработки Visual Studio 2022 и СУБД PostgreSQL. Для создания приложения использовалась библиотека psycopg2. Также была описана структура базы данных и приложения, а также рассмотрен графический интерфейс ПО.

4 Исследовательский раздел

4.1 Исследование производительности ПО

В разработанном ПО нередко встречаются select-запросы по полю с фильтрацией.

Чтобы повысить производительность ПО, можно использовать B-tree индексирование по полю. B-дерево – это автоматически балансирующаяся структура данных, построенная на определенном наборе правил поиска, вставки и удаления данных более быстрым и эффективным способом, непосредственно работающая с памятью. С его помощью можно проиндексировать абсолютно любые данные, для которых предусмотрена возможность сортировки. Использование B-дерева может значительно уменьшить время выполнения запроса, условием которого будет являться выражение, которое состоит из полей, входящих в индекс.

4.2 Технические характеристики устройства

Ниже представлены технические характеристики устройства, на котором выполнялись замеры времени выполнения select-запросов с индексированием и без иного:

- операционная система – Windows 10 Pro;
- память – 8 GB;
- процессор – Intel® Core™ i5 8250U CPU @ 1.6 GHz 1.8 GHz;
- видеокарта – NVIDIA® GeForce® MX130 2GB DDR3/GDDR5 VRAM.

При тестировании ноутбук был включен в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, а также системой тестирования.

4.3 Результаты эксперимента

Эксперимент производился на таблице alcohol.

Наиболее часто встречающийся запрос на данную таблицу: «SELECT * FROM alcohol WHERE alcohol_name LIKE '%searching%';».

Создадим индекс на поле alcohol_name: «CREATE INDEX alco_index ON alcohol USING BTREE(alcohol_name);».

В таблице 4.1 представлены замеры производительности времени выполнения select-запроса (мс) в зависимости от размера таблицы.

Таблица 4.1 – Исследование времени выполнения select-запросов

Размер	Время без индексирования	Время с индексированием
100	0.1437	0.0738
250	0.1562	0.0727
500	0.1757	0.0872
750	0.1935	0.0971
1000	0.2242	0.1068

Исходя из полученных данных можно сделать вывод о том, что время выполнения select-запросов при использовании индексирования уменьшилось в среднем в 2.038 раза относительно времени выполнения запросов без индексирования.

Заключение

Во процессе выполнения курсовой работы была достигнута поставленная цель и выполнены следующие задачи:

- сформулирована задача и определен необходимый функционал;
- проанализированы существующие СУБД и выбрана наиболее подходящая для решения поставленной задачи;
- описана структура базы данных алкогольной продукции;
- спроектирована описанная база данных;
- разработано приложение с графическим пользовательским интерфейсом, позволяющее в полной мере взаимодействовать с реализованной базой данных.

Также в ходе выполнения работы были изучены новые возможности языка Python и библиотека для работы с СУБД PostgreSQL – psycopg2. Приобретен расширенный опыт работы с СУБД PostgreSQL, получен довольно большой набор знаний в области проектирования и разработки баз данных.

В исследовательском разделе опытным путем было установлено, что использование B-дерева в индексировании таблиц увеличивает производительность ПО, в частности сокращает время выполнения select-запроса в среднем в 2.038 раза.

Список использованных источников

1. Сколько и какой алкоголь пьют в России [Электронный ресурс]. – 2019 – . Режим доступа: `https://journal.tinkoff.ru/drunkards/` (дата обращения: 14.04.2022).

2. The effects of alcohol control measures on mortality and life expectancy in the Russian Federation [Электронный ресурс]. – 2019 – . Режим доступа: `https://apps.who.int/iris/bitstream/handle/10665/328167/9789289054379-eng.pdf?sequence=1&isAllowed=y` (дата обращения: 14.04.2022).

3. Python 3.7.0: Документация. [Электронный ресурс]. Режим доступа: https://www.python.org/doc/ (дата обращения: 19.04.2022)

4. psycopg2: Документация. [Электронный ресурс]. Режим доступа: https://www.psycopg.org/ (дата обращения: 20.04.2022)

5. Visual Studio Code 1.71.0: Документация. [Электронный ресурс]. Режим доступа: https://code.visualstudio.com/docs (дата обращения: 20.04.2022)

6. PostgreSQL: Документация. [Электронный ресурс]. Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 24.04.2022)