

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
1 АНАЛИЗ ОБЛАСТИ ПРОЕКТИРОВАНИЯ	5
1.1 Описание предметной области	5
1.2 Обзор существующих решений	7
1.3 Анализ требований к системе и подготовка сопроводительной документации	9
2 РАЗРАБОТКА АРХИТЕКТУРЫ СИСТЕМЫ.....	16
2.1 Проектирование и создание диаграмм.....	16
2.2 Проектирование и создание базы данных	20
3 РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ ДАННЫМИ КОМПАНИИ ПО ПРОДАЖЕ ЭЛЕКТРОНИКИ.....	24
3.1 Реализация GUI-оболочки.....	24
3.2 Демонстрация работы системы	25
ЗАКЛЮЧЕНИЕ	29
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	30
ПРИЛОЖЕНИЯ.....	31
Приложение А	32
Приложение Б.....	33

ВВЕДЕНИЕ

В современном мире темпы развития информационных технологий достигли неописуемых высот, ускоряясь с каждым годом все больше и больше. Это оказывает сильное влияние на все сферы жизни общества, особенно на сферу бизнеса, ведь компании сталкиваются с необходимостью адаптироваться к новым условиям, обеспечивая при этом эффективное управление процессами и ресурсами.

В условиях постоянного роста рынка электроники и увеличения объема данных, с которыми работают компании, управление этими данными становится критически важной задачей. Современные технологии позволяют автоматизировать сбор, хранение и анализ информации, что значительно повышает эффективность работы. Системы управления данными играют ключевую роль в этом процессе, обеспечивая интеграцию различных источников информации и упрощая доступ к ней для пользователей. Эффективная система позволяет компании оптимизировать внутренние процессы, повысить качество обслуживания клиентов и, конечно же, увеличить прибыль. Без такой системы компания не способна выдержать конкуренцию на рынке.

Целью данной курсовой работы является создание системы обработки данных компании по продаже электроники.

Задачи, решаемые в данной курсовой работе:

- анализ требований к системе и подготовка сопроводительной документации;
- проектирование и создание архитектуры системы;
- создание триггеров, функций и процедур;
- разработка GUI оболочки.

1 АНАЛИЗ ОБЛАСТИ ПРОЕКТИРОВАНИЯ

1.1 Описание предметной области

Системы обработки данных представляют собой комплекс программных и аппаратных средств, предназначенных для сбора, обработки и анализа информации. В современном мире, где объемы данных растут с каждым днем, такие системы становятся неотъемлемой частью успешного функционирования организаций. Они позволяют эффективно управлять информацией, автоматизировать бизнес-процессы и принимать обоснованные решения на основе анализа данных.

Проектирование СОД включает в себя как проектирование базы данных, так и разработку прикладного программного обеспечения. База данных является центральным элементом и основным определяющим фактором при создании высокопроизводительных систем. В то же время, тенденция развития современных информационных технологий определяет постоянное возрастание сложности прикладных приложений ИС, усиления их роли в повышении качества, конкурентоспособности ИС, а также увеличения срока ее жизни. [1.1]

В жизненном цикле информационной системы проектирование базы данных и приложений выполняется параллельно, несмотря на большую разницу в используемых методах проектирования и средах реализации. В большинстве случаев проектирование приложений нельзя завершить до окончания проектирования базы данных. С другой стороны, база данных предназначена для поддержки приложений, а потому между фазами проектирования базы данных и проектирования приложений для этой базы данных должен постоянно происходить обмен информацией.

Пакет прикладных приложений представляет собой пользовательские интерфейсы и специальные прикладные программы, предназначенные для

работы с базой данных, например, обеспечивающие права доступа и т. д. Помимо проектирования способов, с помощью которых пользователь сможет получить доступ к необходимым ему функциональным возможностям, необходимо и на сегодняшний день гораздо важнее, разработать соответствующие пользовательские интерфейсы (ПИ) приложений базы данных. Разработка ПИ требует значительных затрат времени и ресурсов. В первую очередь это связано с тем, что от успешной разработки ПИ в большой степени зависит эффективность и конкурентоспособность всей ИС.

Системы обработки данных могут использоваться в различных областях, включая финансы, здравоохранение, производство и маркетинг. Они обеспечивают возможность интеграции данных из различных источников, что способствует получению более полной и точной информации для анализа.

Процесс разработки систем обработки данных включает несколько ключевых этапов, каждый из которых требует тщательного анализа и планирования.

Первым этапом является анализ требований к системе и подготовка сопроводительной документации. Этот шаг является фундаментом процесса разработки, на котором формируется четкое представление о том, что именно должно быть реализовано в системе. Он позволяет выявить особенности бизнес-процессов и технические условия, влияющие на проект. Правильное понимание требований позволяет избежать критических ошибок на поздних этапах разработки. Документация помогает избежать недопонимания между разработчиками и заказчиками, а также обеспечивает ясность на следующих этапах проекта. [1.1]

Следующим шагом является проектирование и создание архитектуры системы. На этом этапе разрабатывается общая структура объекта, включая выбор технологий, платформ и инструментов, которые будут использоваться во время разработки. Архитектура системы должна учитывать требования, выявленные на предыдущем этапе. Важно также определить, как различные компоненты системы будут взаимодействовать друг с другом. Этот этап

является критически важным, так как от правильно спроектированной архитектуры зависит эффективность работы всей системы.

Последним этапом разработки является создание графического пользовательского интерфейса (GUI) оболочки. GUI играет ключевую роль в взаимодействии пользователей с системой, и его дизайн должен быть интуитивно понятным и удобным. Хорошо спроектированный интерфейс обеспечивает легкий доступ к функциональности системы и позволяет пользователям эффективно выполнять свои задачи. Эффективный интерфейс не только улучшает восприятие системы, но и способствует более высокой производительности пользователей. [1.2]

Таким образом, процесс разработки систем обработки данных включает в себя несколько взаимосвязанных этапов, каждый из этих которых играет важную роль в создании эффективной и надежной системы, способной удовлетворить потребности пользователей и обеспечить высокую производительность.

Система обработки данных для компании по продаже электроники направлена на автоматизацию управления данными о товарах, клиентах и продажах. Это включает в себя учет и анализ данных о имеющихся товарах, отслеживание продаж, а также обслуживание клиентов. Основные задачи системы заключаются в улучшении качества обслуживания клиентов и оптимизации складских операций.

1.2 Обзор существующих решений

Рассмотрим примеры компаний по продаже электроники:

Начнем с онлайн-платформы Яндекс Маркет.

Преимущества:

- Широкий выбор товаров от множества продавцов, в том числе независимых.

- Система сравнения товаров по критериям, удобный поиск и фильтрация.
- Удобный и простой интерфейс для пользователей.

Недостатки:

- Высокие комиссии для продавцов, что может отразиться на ценах для покупателей.
- Некоторые пользователи жалуются на качество обслуживания.

Теперь перейдем к интернет-магазину Citilink.

Преимущества:

- Хорошие цены на компьютерную технику и комплектующие.
- Удобная система доставки и самовывоза.
- Информативные описания и оценки от пользователей.

Недостатки:

- На сайте не всегда хватает информации о некоторых товарах.
- Большинство товаров представлено только от одного продавца, что может ограничивать выбор.

Рассмотрим компанию Wildberries.

Преимущества:

- Широкий ассортимент товаров и бесплатная доставка для крупногабаритного товара.
- Удобный мобильный приложение и простой интерфейс сайта.
- Возможность возврата товара в течение 14 дней без объяснения причин.

Недостатки:

- Ограниченный выбор в некоторых категориях, особенно в таких сегментах, как игровые консоли и ноутбуки.
- Некоторые пользователи жалуются на ошибки при доставке и запоздалую отправку товаров.

Рассмотренные платформы предлагают широкий выбор товаров из различных категорий, включая электронику, предоставляют возможность

оставлять отзывы о товарах и продавцах, предлагают услуги по доставке товаров, предоставляют фильтры для удобного поиска товаров по различным параметрам.

1.3 Анализ требований к системе и подготовка сопроводительной документации

Назначение и цели создания (развития) системы

Система обработки данных компании по продаже электроники предназначена для автоматизации ключевых бизнес-процессов: управления запасами, продажами, взаимодействия с клиентами.

Основной целью создания ИС является улучшение обслуживания клиентов за счет проектирования системы управления данными.

Требования к системе

1. Требования к структуре и функционированию системы.

Система имеет модульную структуру, включающую в себя следующие модули:

- модуль раздела «Каталог товаров»;
- модуль раздела «Личный кабинет»;
- модуль раздела «История заказов»;
- модуль раздела «Оформление заказа»;
- модуль работы с базой данных;
- модуль коммуникации и поддержки.

Система должна выполнять следующие функции:

- работа с пользователями;
- управление продажами;
- обработка трафика большого объема;
- информирование о сбоях.

2. Показатели назначения.

Подсистемы, разработанные и доработанные в рамках данного раздела, обязательно должны отвечать следующим требованиям:

- Время на полный запуск (или перезапуск) системы и компонентов системы должно составлять не более 15 минут.
- Коэффициент юзабилити не менее 85%.
- Коэффициент интерактивности не менее 90%.
- Коэффициент достоверности информации не менее 95%.
- REST API подсистемы администрирования: 50 запросов в минуту при времени отклика не более трех секунд.

Требования к аппаратной части и масштабированию для обеспечения перечисленных показателей должны быть определены на этапе технического проектирования.

3. Требования к безопасности.

Система должна обеспечивать высокую степень защиты данных и конфиденциальности пользователей. Это включает такие меры, как шифрование данных, регулярное обновление политик безопасности, а также обеспечение устойчивости программно-технических средств к возможным кибератакам.

4. Требования к эргономике и технической эстетике.

Интерфейс системы должен быть интуитивно понятным и легким в использовании даже для пользователей без опыта работы с подобными системами. Навигация, поиск должны быть простыми и понятными. Система должна быть доступна и удобна для использования на разных устройствах, таких как компьютеры, планшеты и смартфоны. Это включает в себя

адаптивный дизайн, который автоматически подстраивается под размеры экрана и разрешение устройства пользователя.

5. Требования по стандартизации и унификации.

Для реализации статических страниц и шаблонов должны использоваться языки HTML и CSS. Исходный код должен разрабатываться в соответствии со стандартами W3C (HTML 5).

Для реализации интерактивных элементов клиентской части должны использоваться языки JavaScript.

Для реализации динамических страниц должен использоваться язык PHP.

6. Дополнительные требования.

Дополнительные требования не предъявляются.

Требования к функциям (задачам), выполняемым системой

Требования к функциям (задачам), выполняемым системой, приведены в Таблице 1.

Таблица 1.1 – Требования к функциям, выполняемым системой

Функция	Задача
Работа с пользователями	Регистрация пользователей
	Авторизация пользователей
	Техническая поддержка
Оформление заказа	Поиск товара
	Добавление товара в корзину
	Оформление заказа
Информирование о сбоях	Отправление уведомлений о сбое
Обработка трафика большого объема	Запись данных в БД
	Выгрузка данных в оперативную память
	Предоставление данных

Требования к видам обеспечения

1. Требования к математическому обеспечению системы.

Математическое обеспечение системы должно обеспечивать реализацию перечисленных в данном ТЗ функций, а также выполнение операций конфигурирования, программирования, управления базами данных и документирования. Алгоритмы должны быть разработаны с учетом возможности получения некорректной входной информации и предусматривать соответствующую реакцию на такие события

2. Требования к информационному обеспечению системы.

Состав, структура и способы организации данных в системе должны быть определены на этапе технического проектирования.

Данные, используемые системой, должны храниться в реляционной СУБД. Структура базы данных определяется с учетом особенностей внутренней модели системы принятия решений.

Информационный обмен между серверной и клиентской частями системы должен осуществляться по протоколу HTTPS.

3. Требования к лингвистическому обеспечению системы.

Платформа должна быть реализована на русском и английском языках. Должна быть предусмотрена возможность переключения между русским и английским языками через настройки внутри системы. Система ввода-вывода должна поддерживать английский и русский языки.

4. Требования к программному обеспечению системы.

Программное обеспечение клиентской части должно удовлетворять следующим требованиям:

- последняя версия браузера Google Chrome, Firefox или Safari;
- одна из следующих операционных систем: Windows 7, Mac OS X 10.7, Ubuntu 10 или их более поздние версии.

5. Требования к техническому обеспечению системы.

Платформа, на которой будет развернута серверная часть системы, должна удовлетворять следующим минимальным требованиям:

- не менее 8 GB оперативной памяти;
- не менее 500 GB свободного места на жестком диске;
- ОС на базе Linux или ОС Windows;
- поддерживаемый протокол передачи данных HTTP / HTTPS, скорость передачи данных 20 Мбит/с;
- процессор с тактовой частотой не менее 4.6 GHz.

Состав и содержание работ по созданию (развитию) системы

Разработка системы предполагается по календарному плану, приведенному на Рисунках 1.1-1.2.

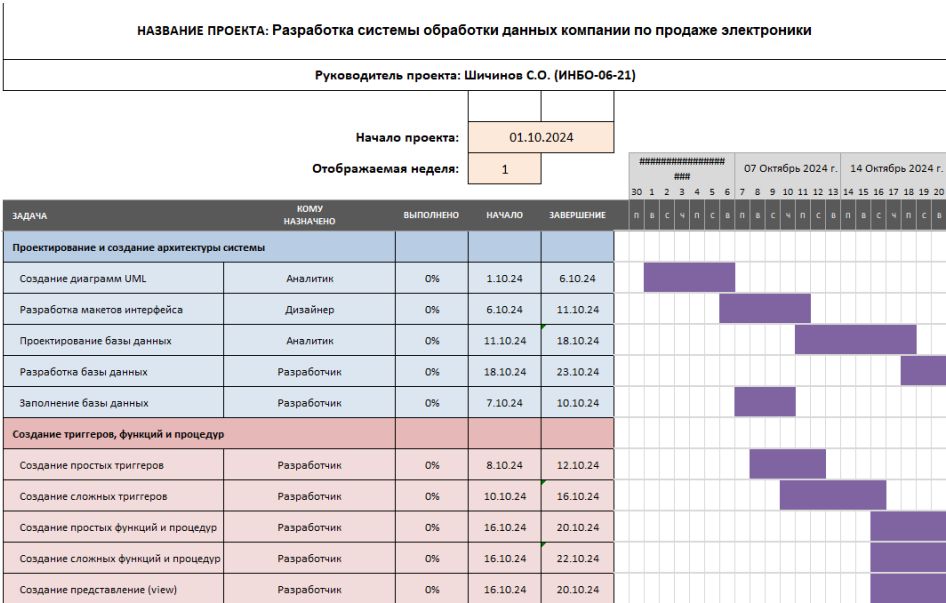


Рисунок 1.1 — Первый фрагмент диаграмма Ганта для системы

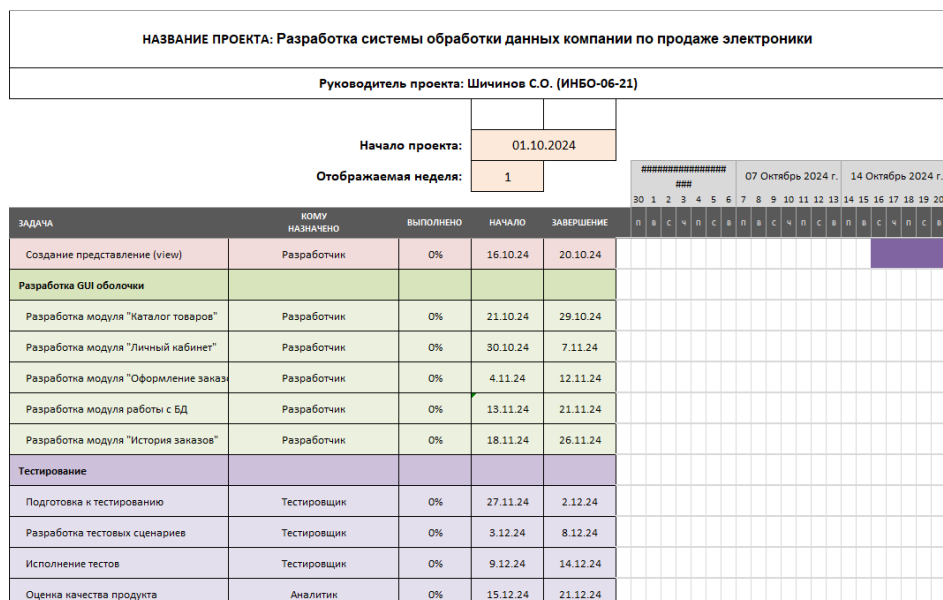


Рисунок 1.2 — Второй фрагмент диаграмма Ганта для системы

Требования к документированию

Проектная документация должна быть разработана в соответствии с ГОСТ 34.201-2020 и ГОСТ 7.32-2017.

Отчетные материалы должны включать в себя текстовые материалы (представленные в виде бумажной копии и на цифровом носителе в формате MS Word) и графические материалы.

Предоставить документы:

- 1) схема функциональной структуры автоматизируемой деятельности;
- 2) описание технологического процесса обработки данных;
- 3) описание информационного обеспечения;
- 4) описание программного обеспечения АС;
- 5) схема логической структуры БД;
- 6) руководство пользователя;
- 7) описание контрольного примера;
- 8) протокол испытаний.

Источники разработки

- ГОСТ 34.602-2020. Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
- ГОСТ Р 59793-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания.
- ГОСТ 34.201-2020. Информационные технологии. Комплекс стандартов на автоматизированные системы. Виды, комплектность и обозначение документов при создании автоматизированных систем.
- ГОСТ Р 59795-2021. Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Требования к содержанию документов.
- ГОСТ 19.106-78. Единая система программной документации. Требования к программным документам, выполненным печатным способом.
- ГОСТ 19.105-78. Единая система программной документации. Общие требования к программным документам.

В результате был проведен анализ области и требований к системе для разработки онлайн-платформы по продаже электроники. Изучение данной области помогло углубить понимание особенностей проектирования платформы, а также определить важные элементы, влияющие на ее функциональность и привлекательность для покупателей. Это понимание позволяет выявить необходимые технологии, которые следует интегрировать для улучшения пользовательского опыта. Все указанные требования будут учтены в дальнейшем этапе проектирования, что гарантирует соответствие платформы запросам целевой аудитории.

2 РАЗРАБОТКА АРХИТЕКТУРЫ СИСТЕМЫ

2.1 Проектирование и создание диаграмм

Для описания процесса обработки заказов в компании по продаже электроники была создана диаграмма в нотации IDEF0. Контекстный уровень диаграммы представлен на Рисунке 2.1.

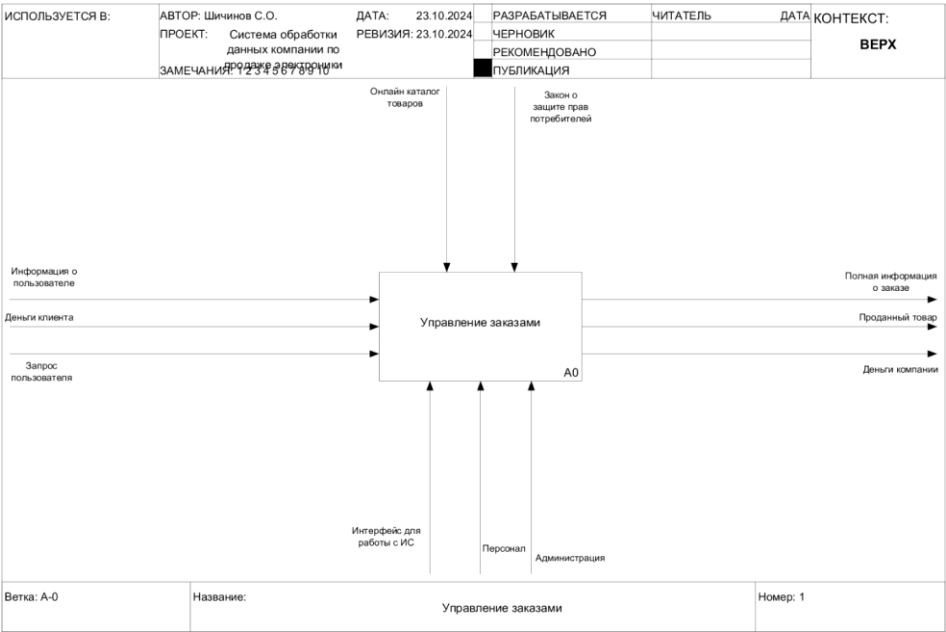


Рисунок 2.1 — Контекстный уровень диаграммы процесса управления заказами компании по продаже электроники в нотации IDEF0

При декомпозиции контекстного уровня были созданы следующие функциональные блоки:

1. Авторизация пользователя (A1).
2. Оформление заказа (A2).
3. Предоставление товара (A3).

Далее был декомпозирован функциональный блок «Оформление заказа» (A2) и были выделены следующие элементы:

1. Выбор товаров (A21).
2. Резервирование товара (A22).
3. Формирование заказа (A23).

На Рисунке 2.2 представлена декомпозиция контекстного уровня в нотации IDEF0.

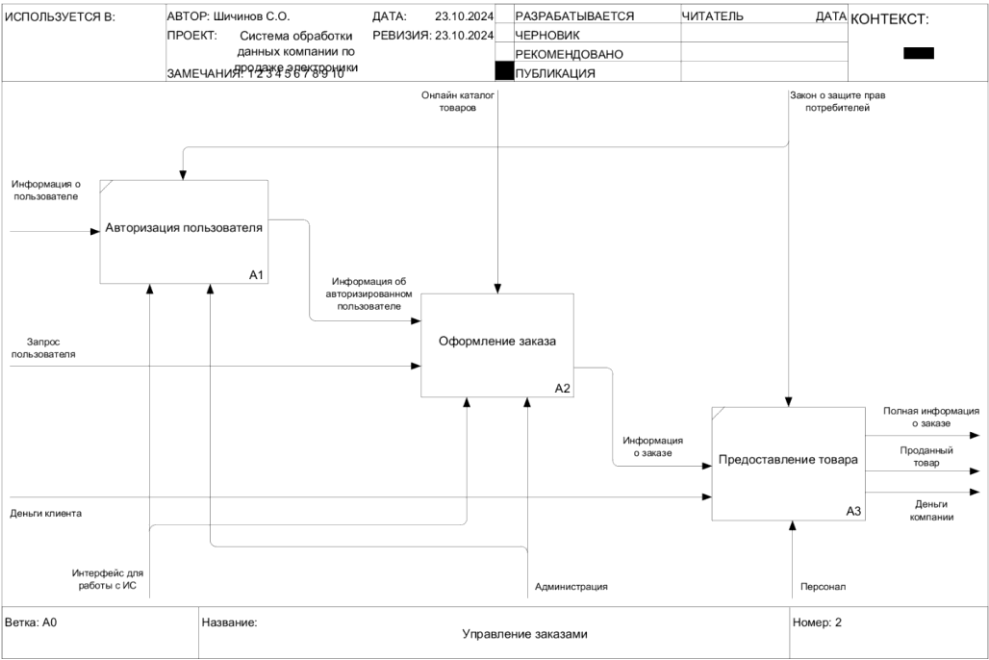


Рисунок 2.2 — Декомпозиция контекстного уровня диаграммы

Декомпозиция диаграммы второго уровня в нотации IDEF0 представлена на Рисунке 2.3.

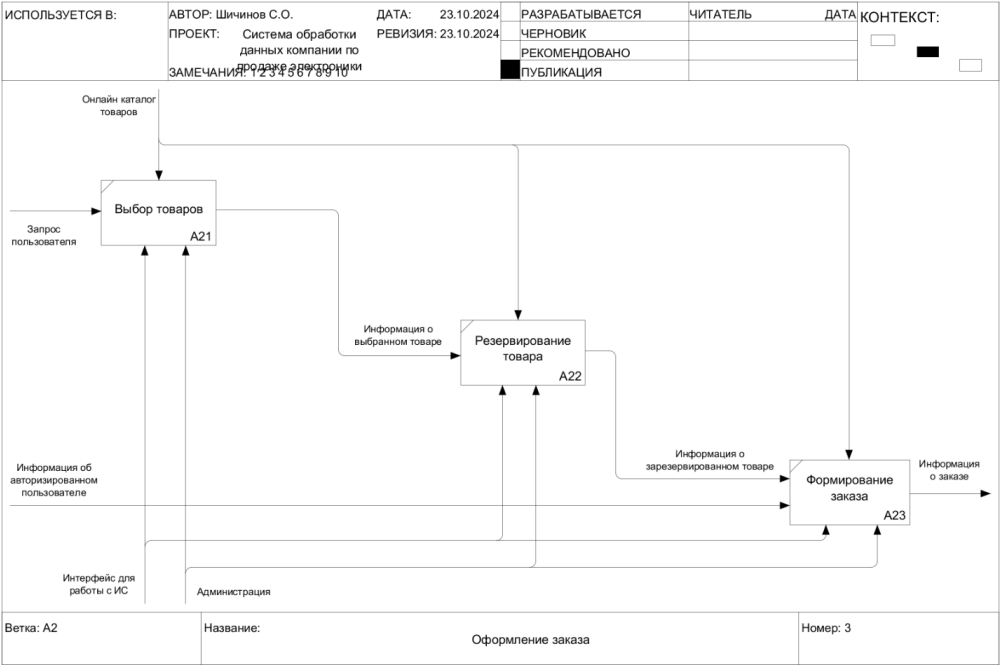


Рисунок 2.3 — Декомпозиция подпроцесса «Оформление заказа»

В процессе создания диаграмм UML будут спроектированы диаграмма компонентов, диаграмма классов и диаграмма последовательности.

Диаграмма классов для системы обработки данных компании по продаже электроники представлена на Рисунке 2.4.

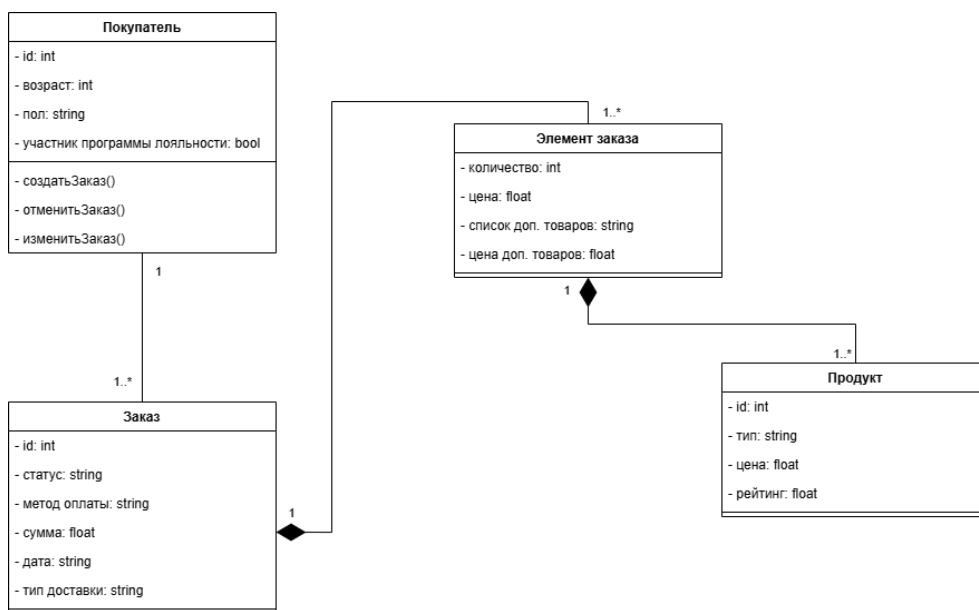


Рисунок 2.4 — Диаграмма классов для системы обработки данных компании по продаже электроники

Диаграмма состоит из взаимосвязанных классов:

1. Класс «Покупатель» определяется идентификатором, возрастом, полом и наличием карты лояльности, он может создать заказ, изменить его или вовсе отменить.
2. Класс «Заказ» обладает идентификатором, статусом, методом оплаты, общей суммой, датой оформления и типом доставки.
3. Класс «Элемент заказа» включает в себя количество товаров, цену, список и цену дополнительных товаров.
4. Класс «Продукт» обладает идентификатором, типом, ценой и рейтингом.

Спроектированная диаграмма компонентов, отражающая зависимость между программными компонентами, представлена на Рисунке 2.5.

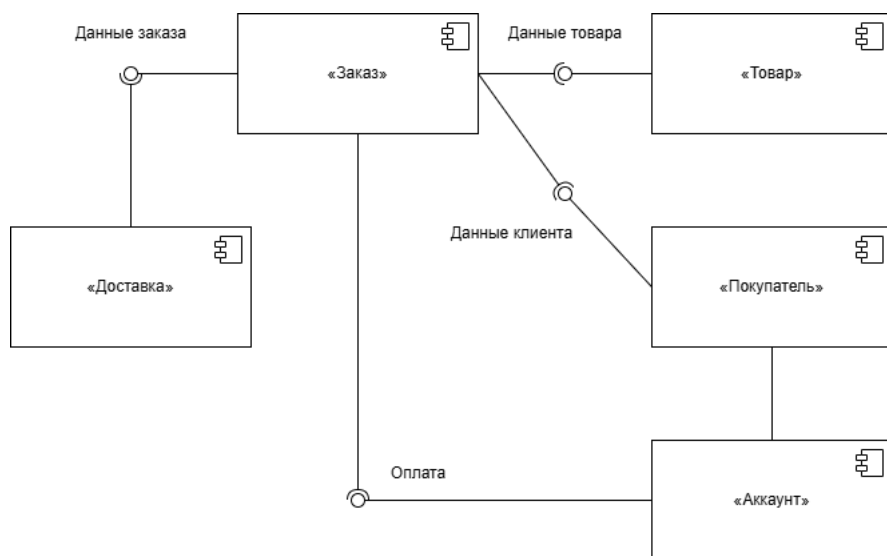


Рисунок 2.5 — Диаграмма компонентов для системы обработки данных компании по продаже электроники

Созданная диаграмма последовательности продемонстрирована на Рисунке 1.6.

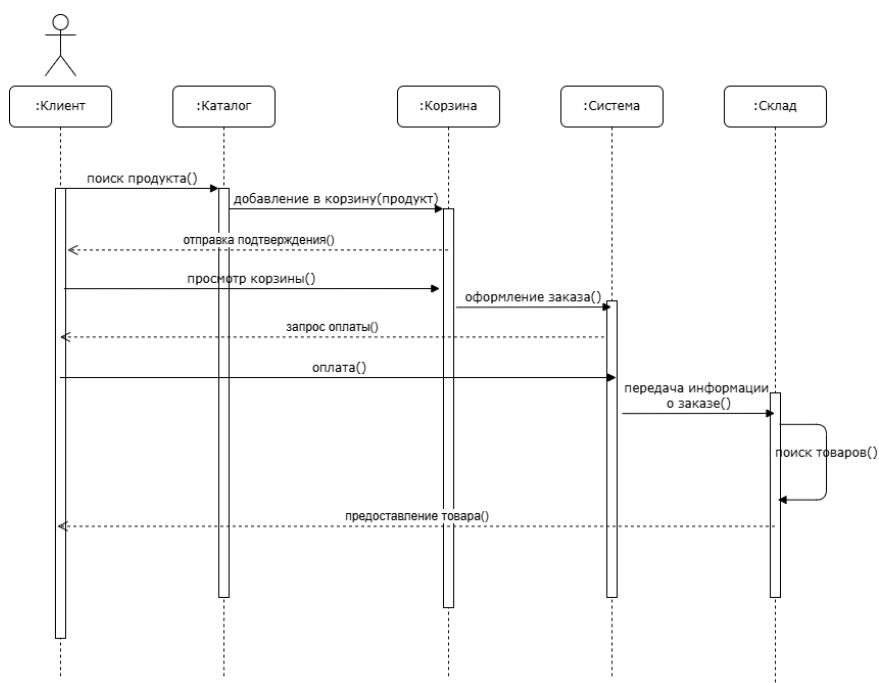


Рисунок 2.6 — Диаграмма последовательности для системы обработки данных компании по продаже электроники

Этап проектирования и создания диаграмм закончен. Теперь перейдем к работе с базой данных.

2.2 Проектирование и создание базы данных

В процессе создания базы данных были спроектированы логическая и физическая модель базы данных. Логическая модель базы данных представлена на Рисунке 2.7.

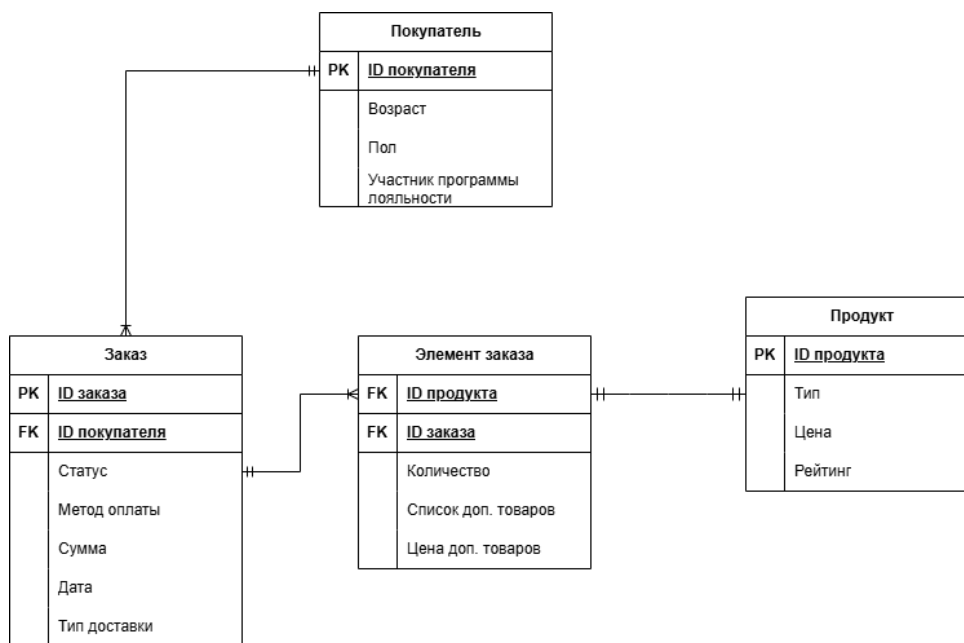


Рисунок 2.7 — Логическая модель базы данных

Физическая модель базы данных системы обработки данных по продаже электроники представлена на Рисунке 2.8.

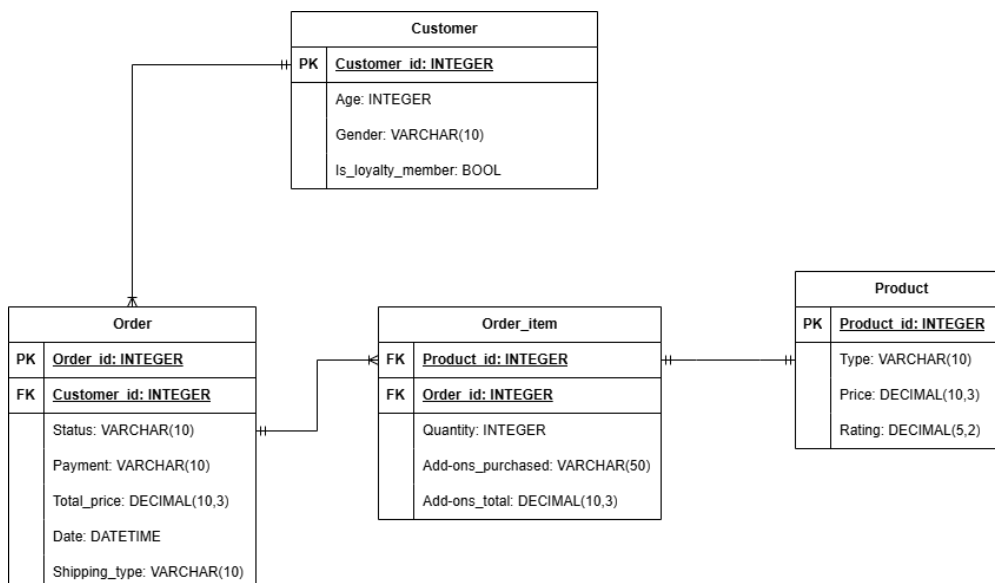


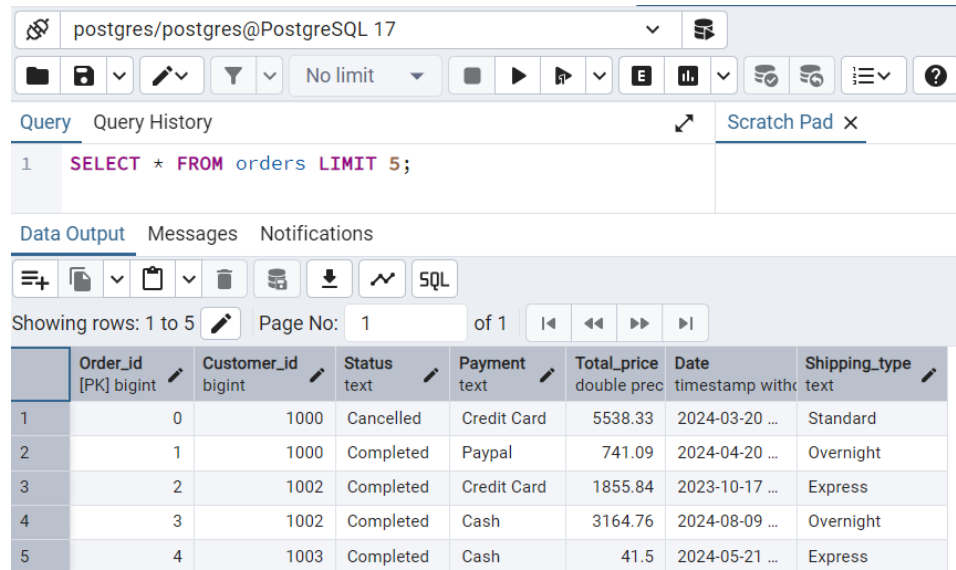
Рисунок 2.8 — Физическая модель базы данных

Исходный набор данных содержит записи о продажах электроники для компании в течение одного года, с сентября 2023 по сентябрь 2024 года. Он включает в себя подробную информацию о демографических данных клиентов, типах продуктов. Вот ключевые поля, входящие в этот набор данных:

- `Customer_id` — уникальный идентификатор клиента (целочисленный тип).
- `Age` — возраст клиента (целочисленный тип).
- `Gender` — пол клиента (строковый тип).
- `Is_loyalty_member` — клиент является участником программы лояльности (логический тип).
- `Type` — тип продаваемого электронного продукта (строковый тип).
- `Product_id` — уникальный код продукта (строковый тип).
- `Rating` — рейтинг продукта от клиента (целочисленный тип).
- `Status` — статус заказа (строковый тип).
- `Payment` — способ, используемый для оплаты (строковый тип).
- `Total_price` — общая стоимость транзакции (вещественный тип).
- `Price` — цена за единицу продукта (вещественный тип).
- `Quantity` — количество купленных единиц (целочисленный тип).
- `Date` — дата покупки (тип даты и времени).
- `Shipping_type` — тип доставки, выбранный клиентом (строковый тип).
- `Add-ons_purchased` — список любых дополнительных товаров, которые были куплены (строковый тип).
- `Add-on_total` — общая стоимость приобретенных дополнений (вещественный тип).

Теперь загрузим данные в PostgreSQL в соответствующие им таблицы, создадим первичные ключи там, где они необходимы.

Фрагмент загруженных в PostgreSQL данных компании по продаже электроники представлен на Рисунке 2.9.



The screenshot shows a PostgreSQL client interface. At the top, the connection is to 'postgres/postgres@PostgreSQL 17'. Below the connection bar is a toolbar with various icons. The 'Query' tab is active, showing a single query: `SELECT * FROM orders LIMIT 5;`. Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is active, displaying a table of 5 rows. The table has 8 columns: 'Order_id [PK] bigint', 'Customer_id bigint', 'Status text', 'Payment text', 'Total_price double prec', 'Date timestamp with', and 'Shipping_type text'. The data is as follows:

	Order_id [PK] bigint	Customer_id bigint	Status text	Payment text	Total_price double prec	Date timestamp with	Shipping_type text
1	0	1000	Cancelled	Credit Card	5538.33	2024-03-20 ...	Standard
2	1	1000	Completed	Paypal	741.09	2024-04-20 ...	Overnight
3	2	1002	Completed	Credit Card	1855.84	2023-10-17 ...	Express
4	3	1002	Completed	Cash	3164.76	2024-08-09 ...	Overnight
5	4	1003	Completed	Cash	41.5	2024-05-21 ...	Express

Рисунок 2.9 — Фрагмент данных компании по продаже электроники в PostgreSQL

Перейдем к описанию триггеров, методов и функция для созданной базы данных.

Триггеры:

1. `age_check_trigger` — срабатывает при вставке в таблицу `customers` или при ее обновлении, следит за тем, чтобы возраст клиента был не меньше 14;
2. `set_date_trigger` — срабатывает при вставке в таблицу `orders`, заменяет неправильно указанную дату на текущую;
3. `rating_check_trigger` — срабатывает при вставке в таблицу `products` или при ее обновлении, следит за тем, чтобы рейтинг был в диапазоне от 1 до 5;
4. `calculate_total_trigger` — срабатывает при вставке в таблицу `order_items` или при ее обновлении, подсчитывает сумму заказа для таблицы `orders` в соответствии с новыми данными;
5. `order_deletion_trigger` — срабатывает при удалении из таблицы `orders`, удаляет все записи, связанные с удаленным заказом, в таблице `order_items`.

Процедуры:

1. `add_order_item` — добавляет запись в таблицу `order_items` с заданными параметрами;
2. `complete_order` — обновляет запись в таблице `orders` (оформление заказа);
3. `add_product` — добавляет запись в таблицу `products` с заданными параметрами;
4. `update_product_info` — обновляет запись в таблицы `products` заданными значениями.

Функции:

1. `get_customer_orders` — возвращает таблицу со всеми заказами заданного клиента;
2. `customer_exists` — возвращает `True`, если `id` клиента существует;
3. `add_order` — создает запись в таблице `orders` и возвращает ее `id`;
4. `add_customer` — создает новую запись в таблице `customers` и возвращает ее `id`;
5. `get_top_products` — возвращает таблицу с заданным количеством самых продаваемых товаров;

Теперь перейдем к реализации системы управления данными компании по продаже электроники. Создадим пользовательский интерфейс и протестируем его.

3 РЕАЛИЗАЦИЯ СИСТЕМЫ УПРАВЛЕНИЯ ДАНЫМИ КОМПАНИИ ПО ПРОДАЖЕ ЭЛЕКТРОНИКИ

3.1 Реализация GUI-оболочки

В качестве языка программирования для написания пользовательского интерфейса был выбран Python, обладающий огромным количеством библиотек под все нужды пользователей.

Для создания самого интерфейса использовалась библиотека PyQt6. Этот инструмент позволяет разрабатывать кроссплатформенные приложения с интуитивно понятным и привлекательным пользовательским интерфейсом. PyQt6 обладает лучшей производительностью по сравнению со схожими инструментами, что делает его подходящим для проектов и приложений, требующих быстрого обновления пользовательского интерфейса.

Наконец, в качестве адаптера для работы с PostgreSQL на Python был выбран psycopg2. Модуль psycopg2 активно развивается, работает с высокой скоростью и обладает внушительным сообществом пользователей.

Интерфейс пользователя состоит из трех основных компонентов:

- окно авторизации пользователя;
- окно регистрации пользователя;
- личный кабинет пользователя.

Первое, что видит пользователь при запуске системы — это окно авторизации. Если он уже зарегистрирован, то клиент должен написать свой идентификатор и нажать на кнопку «Войти», после чего система проверяет наличие id в таблице с помощью функции «customer_exists». Если проверка прошла успешно, клиенту открывается окно с личным кабинетом, в противном

случае система сообщает об ошибке. Также из этого окна есть возможность перейти в окно регистрации с помощью кнопки «Зарегистрироваться».

Следующее окно содержит поля необходимые для регистрации пользователя, при нажатии кнопки срабатывает функция «add_customer», создающая новую запись в таблице клиентов на основе заполненных данных. После этого пользователь переходит в основную часть системы (личный кабинет).

Окно личного кабинета пользователя обладает обширным функционалом:

- просмотр каталога товаров;
- просмотр истории заказов, использующий функцию «get_customer_orders»;
- добавление выбранного товара в корзину (метод «add_order_item» и функция «add_order» для создания заказа);
- просмотр товаров, добавленных в корзину (функция «get_cart»);
- оформление текущего заказа (функция «complete_order»);
- просмотр самых популярных товаров (функция «get_top_products»).

Листинги с кодом реализации базы данных и GUI-оболочки представлены в Приложении А и Приложении Б.

Полученный интерфейс позволяет выполнять все необходимые операции удобно и эффективно, используя реализованные ранее триггеры, методы и функции. Теперь перейдем к демонстрации работы созданной системы.

3.2 Демонстрация работы системы

Продemonстрируем работу системы, рассмотрев вышеописанный функционал.

Первым делом в окне авторизации попытаемся войти под идентификатором существующего пользователя (Рисунок 3.1).

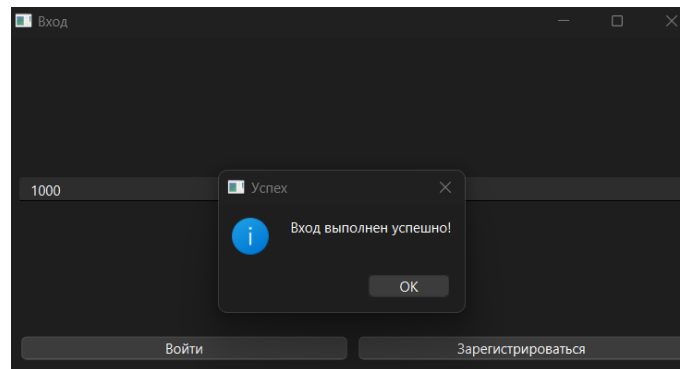


Рисунок 3.1 — Авторизация существующего пользователя

Теперь посмотрим на поведение системы при подаче несуществующего идентификатора (Рисунок 3.2).

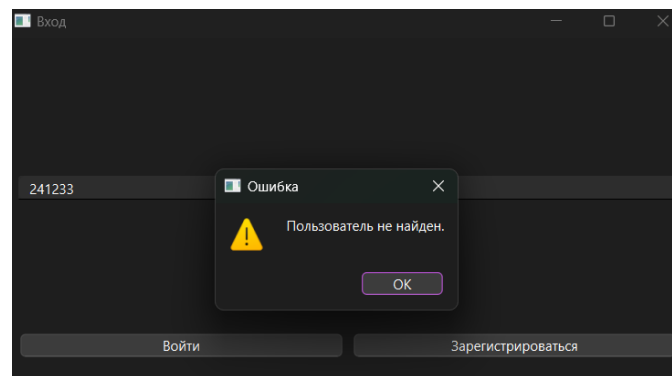


Рисунок 3.2 — Авторизация несуществующего пользователя

Зарегистрируем нового пользователя системы и получим его идентификатор (Рисунок 3.3).

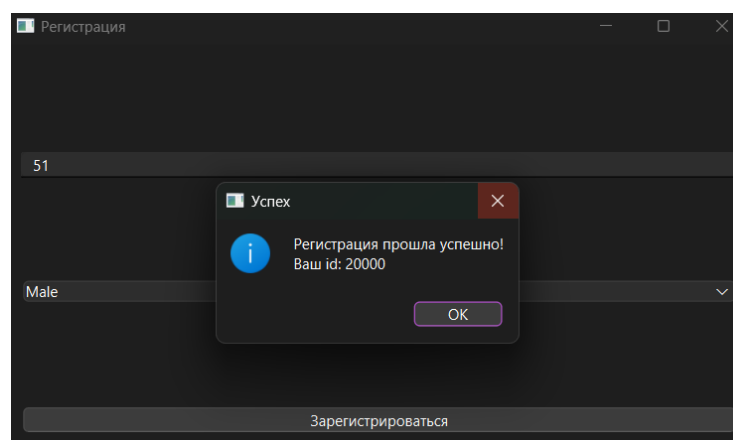


Рисунок 3.3 — Регистрация нового пользователя

Далее перейдем в личный кабинет, откроем каталог товаров и добавим определенный в корзину (Рисунок 3.4).

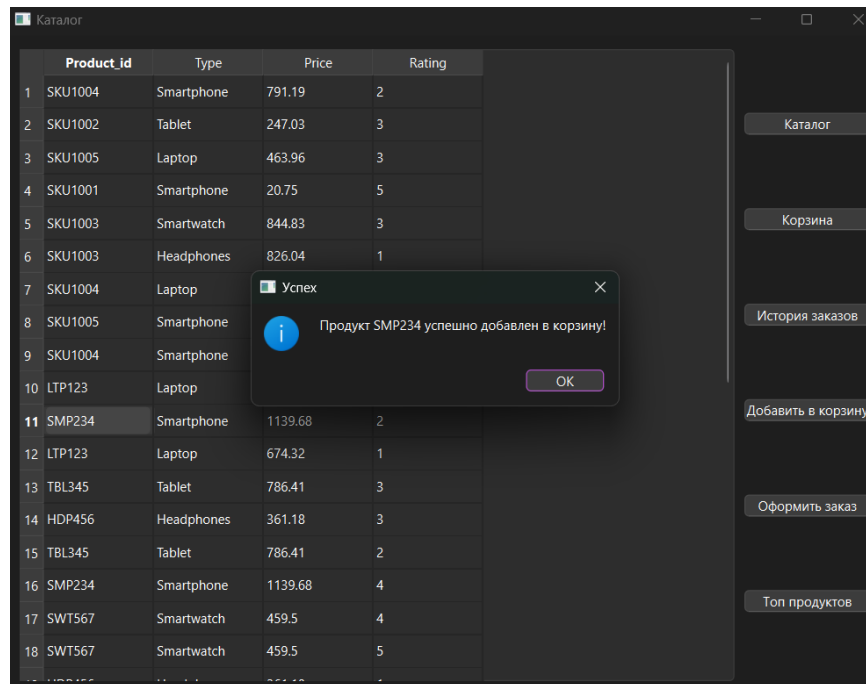


Рисунок 3.4 — Добавление товара из каталога в корзину

Также добавим в корзину несколько товаров из самых популярных (Рисунок 3.5).

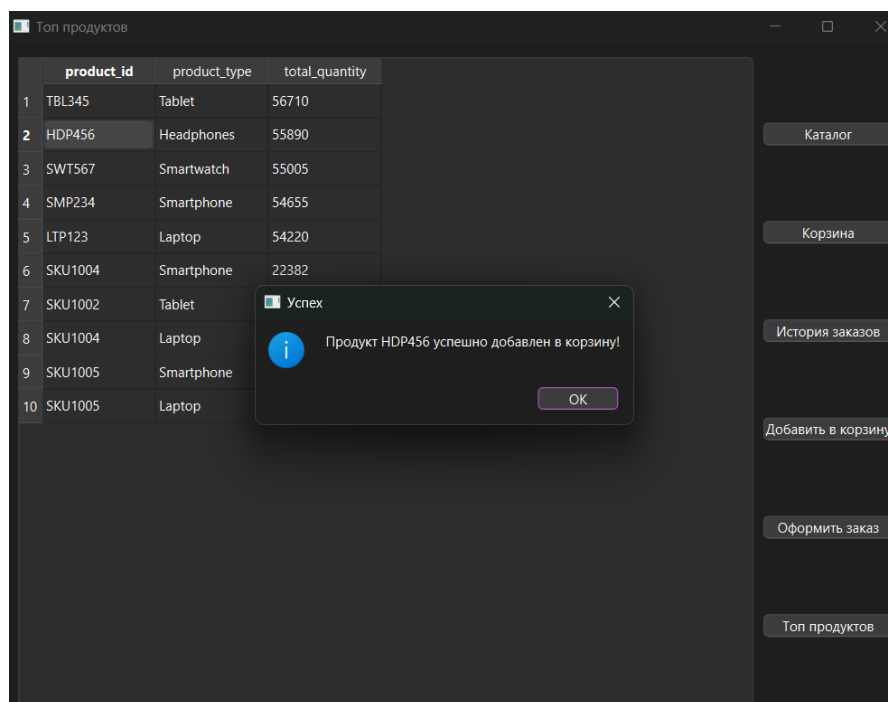
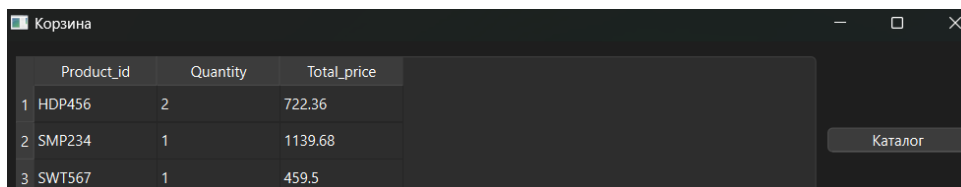


Рисунок 3.5 — Добавление товара из топа в корзину

Перейдем в корзину и убедимся, что товары соответствуют выбранным (Рисунок 3.6).

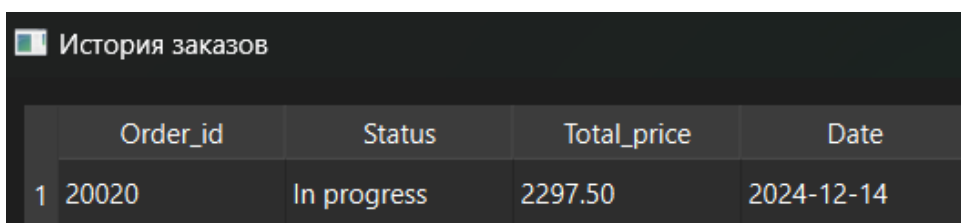


	Product_id	Quantity	Total_price
1	HDP456	2	722.36
2	SMP234	1	1139.68
3	SWT567	1	459.5

Каталог

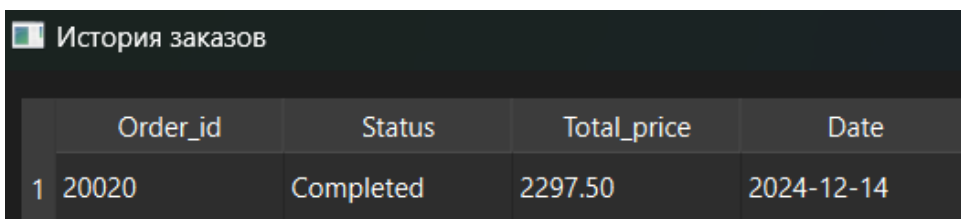
Рисунок 3.6 — Просмотр корзины товаров

Перейдем в историю заказов и посмотрим на данные до оформления заказа (Рисунок 3.7) и после (Рисунок 3.8).



	Order_id	Status	Total_price	Date
1	20020	In progress	2297.50	2024-12-14

Рисунок 3.7 — История заказов до оформления заказа



	Order_id	Status	Total_price	Date
1	20020	Completed	2297.50	2024-12-14

Рисунок 3.8 — История заказов после оформления заказа

Проведенное тестирование системы подтвердило работоспособность вышеописанного функционала как оболочки, так и базы данных. В дальнейшем можно добавить поиск товара в каталоге по критериям, а также пароль для авторизации клиентов.

ЗАКЛЮЧЕНИЕ

Успешное управление процессами и ресурсами в условиях постоянного технологического прогресса требует от компаний не только внедрения новых технологий, но и стратегического подхода к их использованию.

Современные технологии предоставляют возможности для автоматизации процессов сбора, хранения и анализа данных, что существенно повышает производительность работы. Системы управления данными являются центральным элементом этого процесса, обеспечивая интеграцию различных информационных источников и упрощая доступ к данным для пользователей.

Эффективная система управления данными позволяет организациям оптимизировать свои внутренние процессы, улучшить качество обслуживания клиентов и, безусловно, увеличить прибыль. В условиях конкурентного рынка отсутствие такой системы может негативно сказаться на способности компании сохранять свою конкурентоспособность.

Цель данной курсовой работы — создание системы обработки данных компании по продаже электроники — достигнута.

В ходе выполнения данной курсовой работы были выполнены следующие задачи:

- проанализированы требования к системе и подготовлена сопроводительная документация;
- спроектирована и создана архитектура системы;
- созданы триггеры, функции и процедуры;
- разработана GUI оболочка.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

АНАЛИЗ ОБЛАСТИ ПРОЕКТИРОВАНИЯ И ОБЗОР ИМЕЮЩИХСЯ РАЗРАБОТОК

- 1.1. Осипов Д. Л. Технологии проектирования баз данных. - М.: ДМК Пресс, 2019. - 498 с.: ил.
- 1.2. Базы данных: учебное пособие для обучающихся по направлению подготовки 09.03.03 «Прикладная информатика» / Сост.: Т.Ж. Базаржапова, О.А. Гармаева, А.Ю. Хаптахасев. - Улан-Удэ: ФГБОУ ВО БГСХА, 2022. - 84 с.
- 1.3. Oracle [Электронный ресурс] — Режим доступа:
<https://www.oracle.com/cis/database/what-is-data-management/>

ПРИЛОЖЕНИЯ

Приложение А — код создания базы данных.

Приложение Б — код реализации графической оболочки.

Приложение А

В Листинге А.1 представлен код создания базы данных на языке SQL для PostgreSQL.

Листинг А.1 — Код создания базы данных

```
CREATE DATABASE postgres;

CREATE TABLE customers (
  Customer_id SERIAL PRIMARY KEY,
  Age INT NOT NULL,
  Gender VARCHAR(10) NOT NULL,
  Is_loyalty_member BOOLEAN NOT NULL
);

CREATE TABLE products (
  Product_id VARCHAR(10) PRIMARY KEY,
  Type VARCHAR(20) NOT NULL,
  Price DECIMAL(10, 3) NOT NULL,
  Rating INT NOT NULL
);

CREATE TABLE orders (
  Order_id SERIAL PRIMARY KEY,
  Customer_id INT NOT NULL,
  Status VARCHAR(20) NOT NULL,
  Payment VARCHAR(20),
  Total_price DECIMAL(10, 3) NOT NULL,
  Date DATE NOT NULL,
  Shipping_type VARCHAR(20),
  FOREIGN KEY (Customer_id) REFERENCES customers (Customer_id) ON DELETE CASCADE
);

CREATE TABLE order_items (
  Order_id INT NOT NULL,
  Product_id VARCHAR(10) NOT NULL,
  Quantity INT NOT NULL,
  Add-ons_purchased VARCHAR(20),
  Add-ons_total DECIMAL(10, 3),
  FOREIGN KEY (Order_id) REFERENCES orders (Order_id) ON DELETE CASCADE,
  FOREIGN KEY (Product_id) REFERENCES products (Product_id) ON DELETE CASCADE
);
```

Приложение Б

В Листинге Б.1 представлен код реализации графической оболочки на языке Python.

Листинг Б.1 — Код реализации графической оболочки

```
import sys
import psycopg2
from PyQt6.QtWidgets import (
    QApplication,
    QWidget,
    QVBoxLayout,
    QHBoxLayout,
    QPushButton,
    QLineEdit,
    QTableWidgetItem,
    QTableWidgetItem,
    QMessageBox,
    QComboBox
)

conn = psycopg2.connect(
    dbname="postgres",
    user="postgres",
    password="postgres",
    host="localhost",
    port="5432"
)
cursor = conn.cursor()

class LoginWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Вход")
        self.setGeometry(100, 100, 600, 400)

        page_layout = QVBoxLayout()
        button_layout = QHBoxLayout()

        self.id_input = QLineEdit(self)
        self.id_input.setPlaceholderText("ID пользователя")
        page_layout.addWidget(self.id_input)
        page_layout.addLayout(button_layout)

        self.login_button = QPushButton("Войти", self)
        self.login_button.clicked.connect(self.login)
        button_layout.addWidget(self.login_button)

        self.register_button = QPushButton("Зарегистрироваться", self)
        self.register_button.clicked.connect(self.open_registration_window)
```

```

button_layout.addWidget(self.register_button)

self.setLayout(page_layout)

def login(self):
    user_id = int(self.id_input.text())
    cursor.execute("SELECT * FROM customer_exists(%s);", (user_id,))
    user = cursor.fetchone()[0]

    if user:
        QMessageBox.information(self, "Успех", "Вход выполнен успешно!")
        self.close()
        self.open_main_window(user_id)
    else:
        QMessageBox.warning(self, "Ошибка", "Пользователь не найден.")

def open_registration_window(self):
    self.registration_window = RegistrationWindow()
    self.registration_window.show()
    self.close()

def open_main_window(self, user_id):
    self.main_window = MainWindow(user_id)
    self.main_window.show()
    self.close()

class RegistrationWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Регистрация")
        self.setGeometry(100, 100, 600, 400)

        layout = QVBoxLayout()

        self.age_input = QLineEdit(self)
        self.age_input.setPlaceholderText("Возраст пользователя")
        layout.addWidget(self.age_input)

        self.gender_combo = QComboBox(self)
        self.gender_combo.addItem("Male", "Female")
        layout.addWidget(self.gender_combo)

        self.register_button = QPushButton("Зарегистрироваться")
        self.register_button.clicked.connect(self.register)
        layout.addWidget(self.register_button)

        self.setLayout(layout)

    def register(self):
        age = self.age_input.text()
        gender = self.gender_combo.currentText()

```



```

if age and gender:
    try:
        cursor.execute("SELECT add_customer(%s, %s);", (int(age), gender))
        customer_id = cursor.fetchone()[0]
        conn.commit()
        QMessageBox.information(self, "Успех", f"Регистрация прошла успешно!\nВаш
id:
{customer_id}")
        self.open_main_window(customer_id)
    except Exception as e:
        QMessageBox.warning(self, "Ошибка", str(e))
    else:
        QMessageBox.warning(self, "Ошибка", "Пожалуйста, заполните все поля.")

def open_main_window(self, user_id):
    self.main_window = MainWindow(user_id)
    self.main_window.show()
    self.close()

class MainWindow(QWidget):
    def __init__(self, user_id):
        super().__init__()
        self.user_id = user_id
        self.order_id = None
        self.setGeometry(100, 100, 800, 600)

        page_layout = QHBoxLayout()
        button_layout = QVBoxLayout()

        self.table_widget = QTableWidget()
        page_layout.addWidget(self.table_widget)

        self.catalog_button = QPushButton("Каталог")
        self.catalog_button.clicked.connect(self.show_catalog)
        button_layout.addWidget(self.catalog_button)

        self.cart_button = QPushButton("Корзина")
        self.cart_button.clicked.connect(self.show_cart)
        button_layout.addWidget(self.cart_button)

        self.history_button = QPushButton("История заказов")
        self.history_button.clicked.connect(self.show_history)
        button_layout.addWidget(self.history_button)

        self.add_to_cart_button = QPushButton("Добавить в корзину")
        self.add_to_cart_button.clicked.connect(self.add_to_cart)
        button_layout.addWidget(self.add_to_cart_button)

        self.complete_order_button = QPushButton("Оформить заказ")
        self.complete_order_button.clicked.connect(self.complete_order)
        button_layout.addWidget(self.complete_order_button)

```

```

self.top_products_button = QPushButton("Топ продуктов")
self.top_products_button.clicked.connect(self.show_top_products)
button_layout.addWidget(self.top_products_button)

self.show_catalog()

page_layout.addLayout(button_layout)
self.setLayout(page_layout)

def update_table(self, name):
    self.setWindowTitle(name)
    rows = cursor.fetchall()

    self.table_widget.setRowCount(len(rows))
    self.table_widget.setColumnCount(len(rows[0]))
    self.table_widget.setHorizontalHeaderLabels([desc[0] for desc in cursor.description])

    for i, row in enumerate(rows):
        for j, item in enumerate(row):
            self.table_widget.setItem(i, j, QTableWidgetItem(str(item)))

def show_catalog(self):
    cursor.execute("SELECT * FROM products;")
    self.update_table("Каталог")

def show_cart(self):
    if(self.order_id is None):
        QMessageBox.warning(self, "Ошибка", "Продукты не выбраны!")
        return
    cursor.execute(f"SELECT * FROM get_cart({self.order_id});")
    self.update_table("Корзина")

def show_history(self):
    cursor.execute(f"SELECT * FROM get_customer_orders({self.user_id});")
    self.update_table("История заказов")

def add_to_cart(self):
    if(self.windowTitle() == "История заказов"):
        QMessageBox.warning(self, "Ошибка", "Продукт не выбран!")
        return
    if(self.order_id is None):
        cursor.execute(f"SELECT add_order({self.user_id});")
        self.order_id = int(cursor.fetchone()[0])
        conn.commit()

    row = self.table_widget.currentRow()
    column = 0
    product_id = self.table_widget.item(row, column).text()
    cursor.execute("CALL add_order_item(%s, %s)", (self.order_id, product_id))
    conn.commit()
    QMessageBox.information(self, "Успех", f"Продукт {product_id} успешно добавлен в корзину!")

```

```

def complete_order(self):
    if (self.order_id is None):
        QMessageBox.warning(self, "Ошибка", "Продукты не выбраны!")
        return
    cursor.execute("CALL complete_order(%s, %s, %s, %s)", (self.order_id, self.user_id,
'Paypal', 'Overnight'))
    conn.commit()
    QMessageBox.information(self, "Успех", "Заказ успешно оформлен!")
    self.order_id = None

def show_top_products(self):
    cursor.execute("SELECT * FROM get_top_products(10);")
    self.update_table("Топ продуктов")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    login_window = LoginWindow()
    login_window.show()
    sys.exit(app.exec())

```