Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с

оценкой

В.Г.Лукоянычев

« » \_\_\_\_\_\_ 2022 г.

**ОТЧЕТ**

О производственной практике

Технологическая (проектно-технологическая) практика

По теме «Анализ покупательской активности»

в АО «Ритейл-Интеграция»

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент гр. ПИ-92 |  |  |  | И.И.Шинтяпин | | | |
|  | Подпись |  |  | И. О. Ф. | | |
| Руководитель от профильной |  | руководитель |  |  | | | |  |
| организации |  | отдела разработки | О.В.Ковальчук | | | |
|  | Подпись | Должность |  | И. О. Ф. | | |
| Руководитель от университета | доцент, к.т.н. |  | В.Г.Лукоянычев | | | |
|  | Должность, ученая степень |  |  | И. О. Ф. |  |

2022

ФГБОУ ВО «Алтайский государственный технический университет

им. И. И. Ползунова»

Кафедра прикладной математики

**Индивидуальное задание**

На производственную практику Технологическая (проектно-технологическая) практика

(вид и тип практики по УП)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| студенту | | |  | Шинтяпину И.И.. | | | группы | | | | ПИ-92 | | |  |
|  |  |  |  |  | (Ф.И.О.) | |  |  | | |  |  |  |  |
| **Профильная организация** | | | | | АО «Ритейл-Интеграция» | | | | | |  |  |  |  |
|  |  |  |  |  | (наименование) | | | |  |  |  |  |  |  |
| **График проведения практики:** | | | | |  | | | |  |  |  |  |  |  |
|  |  |  |  | | | | | | | |  |  |  |  |
|  | **№** |  | **Содержание работ, выполняемых на** | | | | | | | |  | **Сроки** |  |  |
|  | **п/** |  |  |  | **практике** | |  |  | | |  | **выполнения** |  |  |
|  | **п** |  |  |  |  | |  |  | | |  |  |  |  |
|  | 1 |  | Изучение организации работы предприятия и используемого на | | | | | | | |  | 20.06-21.06 |  |  |
|  |  |  | нем инструментария и ПО | | |  |  |  | | |  |  |  |  |
|  | 2 |  | Формулировка задач для решения в ходе практики, вида и объема | | | | | | | |  | 21.06 |  |  |
|  |  |  | результатов | |  | |  |  | | |  |  |  |  |
|  | 3 |  | Изучение и анализ предметной области, библиографический | | | | | | | |  | 21.06 – 23.06 |  |  |
|  |  |  | поиск, изучение литературы. | | | |  |  | | |  |  |  |  |
|  | 4 |  | Постановка задачи, проектирование состава и структуры ПО | | | | | | | |  | 24.06 – 26.06 |  |  |
|  |  |  |  | | | |  |  | | |  |  |  |  |
|  | 5 |  | Реализация программного обеспечения | | | |  |  | | |  | 27.06 -14.07 |  |  |
|  |  |  |  | | | |  |  | | |  |  |  |  |
|  | 6 |  | Тестирование программного обеспечения | | | |  |  | | |  | 14.07 – 15.07 |  |  |
|  |  |  |  | | | |  |  | | |  |  |  |  |
|  | 7 |  | Оформление и сдача отчета по практике | | | |  |  | | |  | 16.07-17.07 |  |  |
|  |  |  |  |  |  | |  |  | | |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель практики |  |  |  |  | В.Г.Лукоянычев, доцент, к.т.н. | | |
| от университета | | (подпись) | |  |  | (Ф.И.О., должность) | |
| Руководитель практики от | |  |  |  |  |  |  |
| профильной организации | |  |  |  |  |  | Ковальчук О.В., руководитель |
|  |  | (подпись) | |  |  |  | (Ф.И.О., должность) отдела  разработки |
| Задание принял к исполнению | | | |  |  |  | Шинтяпин И. И. |
|  |  |  |  | (подпись) | |  | (Ф.И.О.) |
|  |  | **Инструктаж по ОТ, ТБ, ПБ, ПВТР** | | | | | |

Инструктаж обучающегося по ознакомлению с требованиями охраны труда, техники безопасности, пожарной безопасности, а также правилами внутреннего трудового

распорядка проведен « » 2022 г.

Руководитель практики от

профильной организации Ковальчук О.В., руководитель отдела разработки



(подпись) (Ф.И.О., должность)

Оглавление

[**Введение 4**](#_Toc111393800)

[**1. О предприятии 5**](#_Toc111393801)

[**2. Описание предметной области и постановка задачи 11**](#_Toc111393802)

[2.1 Описание предметной области 11](#_Toc111393803)

[2.2 Постановка задачи 12](#_Toc111393804)

[**3. Проектирование 13**](#_Toc111393805)

[3.1 Модель данных 13](#_Toc111393806)

[3.2 Описание форм 15](#_Toc111393807)

[**4 Реализация 16**](#_Toc111393808)

[4.1 Инструменты и технологии 16](#_Toc111393809)

[4.2 Реализация программного обеспечения 16](#_Toc111393810)

[**Заключение 18**](#_Toc111393811)

[**Список использованных источников 19**](#_Toc111393812)

[**Приложение А. Тестирование программного обеспечения 20**](#_Toc111393813)

[**Приложение Б. Исходный текст программы 29**](#_Toc111393814)

# Введение

Покупательская активность напрямую зависит от значения покупательской способности. Под покупательской способностью понимают экономический показатель, обратно пропорциональный количеству валюты, необходимой для покрытия определённой потребительской корзины из товаров и услуг. Данный параметр показывает, сколько среднестатистический потребитель может купить на определённую сумму денег товаров и услуг при существующем уровне цен.

Покупательная способность зависит от уровня доходов населения и той их части, которая может быть выделена на покупки, но также обусловлена величиной цен и тарифов на услуги. Это очень важный экономический показатель, который во многом определяет уровень развития экономики страны и уровень благосостояния населения в целом.

Значение покупательской способности определяет значение покупательской активности, с увеличением первого показателя соответственно увеличивается второй и наоборот.

Целью данной работы является написание программного продукта, обеспечивающего хранение и обработку большого количества данных о покупках (чеков) и самих товаров, проводя тем самым анализ покупательской активности.

# О предприятии

Группа компаний «Ритейл Сервис» специализируется на проведении комплексной автоматизации предприятий розничной торговли, общественного питания, индустрии гостеприимства, развлекательных комплексов, производственных и других предприятий.

Основные направления деятельности компании, включают в себя:

* Внедрение систем автоматизации (как типовых решений, так и индивидуально разработанных под задачи клиента).
* Проектирование и монтаж структурированных кабельных сетей и систем контроля доступа.
* Продажа офисной техники, торговой техники и оборудования (онлайн-кассы, сканеры штрихкода, торговые весы, терминалы сбора данных и др.).
* Продажа, аренда и сопровождение программного обеспечения.
* Продажа, аренда, внедрение и сопровождение специализированного кассового программного обеспечения Artix.
* Продажа, внедрение и сопровождение программных продуктов. Фирмы 1С.
* Сервисное обслуживание: сопровождение программного обеспечения, техническое обслуживание и ремонт терминалов, источников бесперебойного питания и стабилизаторов, торгового и офисного оборудования, обслуживание серверов, сетей, персональных компьютеров и оргтехники.
* Создание и выдача сертификатов ключей электронной цифровой подписи для юридических лиц и индивидуальных предпринимателей.

Компания «Ритейл Сервис» сотрудничает с такими организациями, как

* **«Kaspersky»** - Ведущий мировой производитель программных решений для обеспечения информационной безопасности.
* **«1c-bitrix»** - Занимает лидирующие позиции на российском рынке в области разработки систем управления веб-проектами и корпоративными порталами.
* **«Microsoft»** - Мировой лидер в области программного обеспечения, IТ-услуг и решений.
* **«1C»** - Российская компания, специализирующаяся на разработке, дистрибьюции, издании и поддержке компьютерных программ делового и домашнего назначения.
* **«Атол»** - Одна из ведущих компаний на российском рынке автоматизации предприятий торговли, общественного питания и индустрии развлечений, аптек, АЗС, складов, производства и сферы услуг.
* **«Такском»** - специализируется на разработке, внедрении и сопровождении систем электронного документооборота, обеспечивающих обмен по каналам связи конфиденциальной информацией, защищенной электронной цифровой подписью.
* **«Cassida»** - Производитель профессионального банковского оборудования для ритейла, финансовых организаций банковского сектора и для всех компаний, чья деятельность связана с получением наличности.
* **«Русский проект»** - Лидер на российском рынке технологий и оборудования для предприятий общественного питания и торговли, предлагающий весь спектр услуг по проектированию, оснащению и техническому обслуживанию предприятий общественного питания и торговли любого формата и профиля.

Группа компаний «Ритейл-Сервис» начала свою деятельность 29 декабря 1999 года с момента возникновения ООО «Софт-Сиб Алтай», то есть данная фирма уже более 20ти лет предоставляет высококвалифицированные услуги по автоматизации розничной торговли, это говорит о колоссальном опыте организации. Численность компании в первые годы работы – 7 человек. Спустя несколько лет компания автоматизирует первые пять супермаркетов самообслуживания «Мария-Ра», получает статус сотрудника «1С», открывает филиалы в Рубцовске и Бийске. В 2005 году компания запускает собственное производство самоклеящихся этикеток для торговых и производственных предприятий Алтайского края. Компания растет, численность персонала увеличивается до 50 человек.

На протяжении следующего десятилетия происходит планомерное развитие организации: численность компании превышает 200ти человек, открываются представительства во многих городах России, реализуются масштабные проекты и подтверждается статус партнера со многими крупными корпорациями мира. За это время компания занимает лидирующие позиции по автоматизации бизнес-процессов не только в регионе, но и на федеральном уровне.

IT-отдел компании использует методологию SCRUM-разработки – гибкая методология управления проектами, содержит минимально необходимый набор мероприятий, артефактов, ролей, на которых строится процесс SCRUM-разработки, позволяющий за фиксированные небольшие промежутки времени, называемые спринтами (sprints), предоставлять конечному пользователю работающий продукт с новыми бизнес-возможностями, для которых определён наибольший приоритет.

Спринт — промежуток времени, достаточный для выполнения запланированной совокупности операций SCRUM, целью которой является создание инкремента бизнес-продукта. Жёстко фиксирован по времени. Длительность одного спринта от 1 до 4 недель. Чем короче спринт, тем более гибким является процесс разработки, релизы выходят чаще, быстрее поступают отзывы от потребителя, меньше времени тратится на работу в неправильном направлении, но много времени тратится на митинги планирования спринта, ретроспективы. С другой стороны, при более длительных спринтах команда (SCRUM Team) уменьшает издержки на совещания, демонстрации продукта и. т. п.

Преимущества Scrum заключаются в том, что он ориентирован на клиента, адаптивен, предостваляет клиенту возможность делать изменения в требованиях в любой момент времени. Возможность изменения требований привлекательна для многих заказчиков ПО.

Scrum достаточно прост в изучении, позволяет экономить время, за счет исключения не критичных активностей, позволяет получить потенциально рабочий продукт в конце каждого Sprint'а.

Scrum делает упор на самоорганизующуюся, многофункциональную команду, способную решить необходимые задачи с минимальной координацией. Это особенно привлекательно для малых компаний и стартапов, так как избавляет от необходимости от найма или обучения специализированного персонала руководителей.

Отдел разработки предприятия состоит из двух SCRUM команд, каждая команда в свою очередь состоит из трех тестировщиков и пяти разработчиков. Такое распределение наиболее эффективно и рационально в плане реализации поставленных задач.

Процесс разработки и написания готового ПО заключается в последовательной реализации задач из очереди с наивысшим приоритетом, на спринт дается 2 недели, это объективно достаточный срок на выполнение необходимой задачи.

Для разработки программного обеспечения отдел разработки использует следующие языки программирования: C++, Java, Python. На основе данных ЯП в фирме разработчиками используются следующие фреймворки и инструменты:

C++:

* Qt - фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++, позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

Java:

* Spring - универсальный фреймворк с открытым исходным кодом для Java-платформы, предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.
* Vaadin — свободно распространяемый фреймворк для создания RIA-веб-приложений, разрабатываемый одноимённой финской компанией. В отличие от библиотек на Javascript и специфических плагинов для браузеров, Vaadin предлагает сервер-ориентированную архитектуру, базирующуюся на Java Enterprise Edition. Одна из наиболее значимых функций в Vaadin -использование Java как единственного языка программирования при создании веб-приложений и веб-контента.
* Apache Maven — фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM являющемся подмножеством XML. Maven обеспечивает декларативную, а не императивную сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов.
* Liquibase — это независимая от базы данных библиотека с открытым исходным кодом для отслеживания, управления и применения изменений схемы базы данных.
* Hibernate — библиотека для языка программирования Java, предназначенная для решения задач объектно-реляционного отображения, самая популярная реализация спецификации JPA. Позволяет сократить объёмы низкоуровневого программирования при работе с реляционными базами данных; может использоваться как в процессе проектирования системы классов и таблиц, так и для работы с уже существующей базой.
* JUnit — фреймворк для модульного тестирования программного обеспечения на языке Java.

Также в фирме пользуется следующими базами данных:

* MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, является решением для малых и средних приложений. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей.
* PostgreSQL — свободная объектно-реляционная система управления базами данных, существует в реализациях для множества UNIX-подобных платформ. СУБД позволяет гибко управлять базами данных.
* MongoDB — документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Применяется в веб-разработке, в частности, в рамках JavaScript-ориентированного стека MEAN.

После написания программного обеспечения происходит процесс тестирования то тех пор, пока не будет сто процетного прохождения всех тестов. Затем происходит CodeReview более опытными разработчиками.

В качестве Git-хостинга в «Ритейл Сервис» используется GitLab - веб-инструмент жизненного цикла DevOps с открытым исходным кодом, представляющий систему управления репозиториями кода для Git с собственной вики, системой отслеживания ошибок, CI/CD пайплайном и другими функциями.

В компании проводятся регулярные совещания и деловые встречи с разбором заданий, достигнутых результатов и возникших трудностях в процессе работы, также в коллективе царит неформальная обстановка, все это благоприятно сказывается на рабочем процессе и, безусловно, увеличивает продуктивность работы сотрудников.

# Описание предметной области и постановка задачи

## Описание предметной области

Важное и ключевое значение для розничной и оптовой торговли является автоматизация процессов обслуживания клиентов, бухгалтерского учёта и анализа базы данных.

Процесс автоматизации торговли можно описать так:

Выписка первичных документов: чеки, счет-фактуры, расходные накладные на склад. Выписка связана с базой данных и бухгалтерским учётом, где фиксируются торговые операции. После чего расходные накладные автоматически поступают на склад, где кладовщики сразу начинают собирать заказ, не дожидаясь клиента. При перемещении товара между своими складами и филиалами, товарные накладные в режиме онлайн поступают должностным лицам. После прихода товара на склад от поставщиков, в режиме онлайн передаётся кладовщиком в общую базу данных, что позволяет оперативно начать реализацию клиентам. Поступление денежных средств из банковской программы автоматически выгружается в бухгалтерский и управленческий учёт.

Автоматизация торговли через интернет-магазин, позволяет дистанционно обслуживать клиентов, что привлекает больше покупателей и увеличивает скорость их обслуживания. Состоит из нескольких уровней:

1. Клиент через сайт делает заказ. Сайт связан с базой данных и бухгалтерским учётом, что позволяет корректно, на основание товарных остатков выставлять клиенту счёт.
2. При поступление денежных средств, происходит привязка к выставленному счету и программа передаёт на склад расходную накладную на сбор заказа. Если заказ собрался в полном объёме, склад даёт команду программе на выписку первичных документов и передаёт заказ в службу доставки.

Не менее значимым в торговле стоит автоматизация управленческого учёта. Он необходим для составления заявок поставщикам для пополнения товарных запасов, определения оптимального количества и ассортимента для закупа, составления графика закупок для сезонных товаров, учёта деловой переписки с контрагентами, составления рабочего плана и контроля управленческих решений.

Состоит из следующих блоков:

1. На основание базы данных программа производит анализ расхода товарной номенклатуры в заданный период времени, сравнивает со складскими остатками и отправляет заявки поставщикам, которые привязаны к этой номенклатуре. Менеджеру по закупу остаётся только следить за возможными ошибками программы, контролировать процесс доставки и оплату поставщику.
2. Программа из всего массива данных выявляет сезонный товар и на основе его расхода и остатков на складе, составляет график закупа, с учётом времени доставки.
3. Программа анализирует коммерческие предложения поставщиков, и выявляет наиболее выгодные по цене позиции.
4. Программа фиксирует дату отгрузки от поставщиков и ставит примерный срок прихода на склад с учётом прошлых поставок.
5. Деловая переписка с контрагентами загружается в базу данных в свой сектор.
6. Менеджер вносит в программу свой рабочий план и отмечает его выполнение, а его непосредственный начальник видит его исполнение и корректирует в случаи необходимости.

Перспективной областью для автоматизации является автоматизация крупных складских комплексов, где работа склада происходит без вмешательства человека.

## Постановка задачи

Необходимо разработать приложение для анализа покупательской активности. Приложение должно получать список чеков из базы данных и отображать результаты обработки этой информации, например, на какие товары растет спрос, а на какие падает. Параметров, по которым производится обработка, должно быть несколько, например, выручка по магазину, средний чек, кол-во продажи определенного товара, график кол-ва покупателей.

# Проектирование

## Модель данных

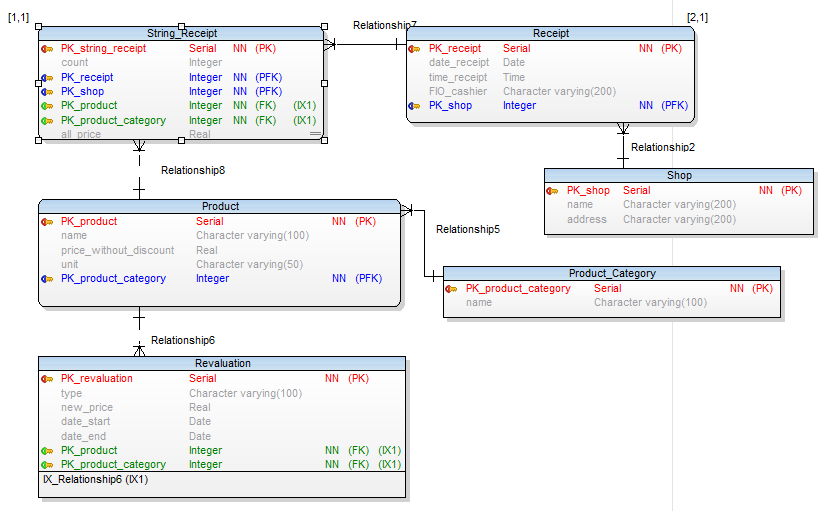


Рис. 3.1 – модель базы данных

В базе данных присутствуют сущности:

1. Shop – магазин, содержит информацию о магазине

Атрибуты:

* 1. PK\_shop (serial, primary key ) – первичный ключ магазина
  2. name (varchar) – название
  3. address (varchar) - адрес

1. Receipt – чек, включает в себя данные о покупке

Атрибуты:

* 1. PK\_receipt (serial, primary key ) – первичный ключ чека
  2. Date\_receipt (Date) – дата создания чека
  3. Time\_receipt (Time) – время создания чека
  4. FIO\_cashier (varchar) – ФИО кассира
  5. PK\_shop (int) – внешний ключ для связанной таблицы «Shop»

1. String\_Receipt – строка чека, содержит необходимую информация о каждом товаре в чеке

Атрибуты:

* 1. PK\_String\_receipt (serial, primary key) – первичный ключ строки чека
  2. PK\_receipt (int) – внешний ключ на связанную таблицу «Receipt»
  3. PK\_shop (int) – внешний ключ на связанную таблицу «Shop»
  4. PK\_product (int) – внешний клбч на связанную таблицу «Product»
  5. PK\_product\_category (int) – внешний ключ на связанную таблицу «Product\_Category»
  6. All\_price (real) – полная сумма за товар

1. Product – таблица, содержащая информацию о товарах

Атрибуты:

4.1 PK\_product (serial, primary key) – первичный ключ товара

4.2 name (varchar) – название товара

4.3 price\_without\_discount (real) – цена товара без скидки

4.4 unit (varchar) – единицы измерения товара (шт, л, кг)

4.5 PK\_product\_category (int) – внешний ключ для связанной таблицы «Product\_ Category»

1. Product\_Category – таблица категорий товара

Атрибуты:

* 1. PK\_product\_category (serial, primary key) – первичный ключ категории товара
  2. Name (varchar) – название категории

1. Revaluation – таблица переоценки товара, содержит всю необходимую информацию о скидки, акции или переоценки товара, включая сроки и цену

Атрибуты:

* 1. PK\_revaluation (serial, primary key) – первичный ключ переоценки
  2. Type (varchar) – тип (скидка, переоценка)
  3. New\_price (Real) – новая цена
  4. Date\_start (Date) – дата начала акции или переоценки
  5. Date\_end (Date) – дата окончания акции или переоценки
  6. PK\_product (int) - внешний ключ для связанной таблицы «Product»
  7. PK\_product\_category (int) - внешний ключ для связанной таблицы «Product\_Category»

## Описание форм

1. MainWindow – стартовая и главная форма. Она выводит списки чеков, товаров и магазинов с общей информацией о них.
2. InfoWindow – форма для показа детальной информации о товарах и магазинах.
3. CheckForm – форма для показа информации о чеках.

# Реализация

## Инструменты и технологии

При разработке использовались:

1. **Qt** — фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++.

Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML.

1. **QCustomPlot** — это виджет Qt C++ для построения графиков и визуализации данных. Он не имеет дополнительных зависимостей и хорошо документирован. Эта библиотека построения графиков ориентирована на создание привлекательных 2D-графиков, графиков и диаграмм высокого качества для публикации, а также предлагает высокую производительность для приложений визуализации в реальном времени.
2. **PostgreSQL** – свободная объектно-реляционная система управления базами данных (СУБД), наиболее развитая из открытых СУБД в мире и являющаяся реальной альтернативой коммерческим базам данных.

## Реализация программного обеспечения

При запуске приложения открывается главная форма, на которой отображается список чеков, с общей информацией о них. Сверху есть три кнопки позволяющие изменять содержимое списка на товары, магазины и чеки. Под ними расположены кнопки позволяющие сортировать и фильтровать список по параметрам, выбранным в списках рядом с этими кнопками. При переходе между списками фильтры и сортировки сбрасываются. Также если два раза нажать левой кнопкой мыши на конкретный товар, чек или магазин то откроется форма, показывающая всю информацию о данном объекте.

Чеки:

В списке для каждого чека выводятся: дата и время закрытия чека, итоговая сумма чека и адрес магазина, в котором была произведена покупка. Фильтровать этот список можно по адресу магазина, а сортировать по дате и времени и итоговой сумме. На форме с детальной информацией можно также увидеть список товаров входящих в чек. Для каждой строки чека в списке выводится название товара, количество и сумма.

Товары:

В списке для каждого товара выводятся: название, категория, сколько продано за этот месяц, цена и доход с этого товара за этот месяц. Также рядом с количеством проданного товара и доходом выводится сравнение в процентах этих показателей с предыдущим месяце. Фильтровать этот список можно по категории товара, а сортировать по цене, продажам за месяц и доходу за месяц. На форме с детальной информацией можно также увидеть графики продаж и доходов для этого товара за этот год.

Магазины:

В списке для каждого магазина выводятся: адрес, количество покупателей за этот месяц и доход этого магазина за этот месяц. Также рядом с количеством покупателей и доходом выводится сравнение в процентах этих показателей с предыдущим месяцем. Фильтровать этот список нельзя, а сортировать можно по покупателям и доходу за месяц. На форме с детальной информацией можно также увидеть графики покупателей и доходов для этого магазина за этот год.

# Заключение

В процессе работы удалось реализовать приложение, способное обрабатывать информацию о продажах (чеках) и отображать результаты обработки.

Преимущества:

1. Возможность фильтровать и сортировать чеки, товары и магазины по разным критериям
2. Вывод информации о продажах за предыдущий год в виде графиков
3. Понятный вывод данных в виде таблиц.

В дальнейшем, возможно, усовершенствовать систему, добавив для товаров вывод информации о том, в каком магазине данный товар продается лучше всего, а в каком хуже всего. Для магазинов можно выводить информацию о лучше всего продающихся товаров в этом магазине по всем категориям товаров.

# Список использованных источников

1. Qt Documentation [Электронный ресурс]. – Режим доступа: https://doc.qt.io/, свободный.
2. Postgresql.org [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs, свободный.
3. Stack Overflow [Электронный ресурс]. – Режим доступа: https://stackoverflow.com/, свободный.
4. Habr [Электронный ресурс]. – Режим доступа: https://habr.com, свободный.

# Приложение А. Тестирование программного обеспечения

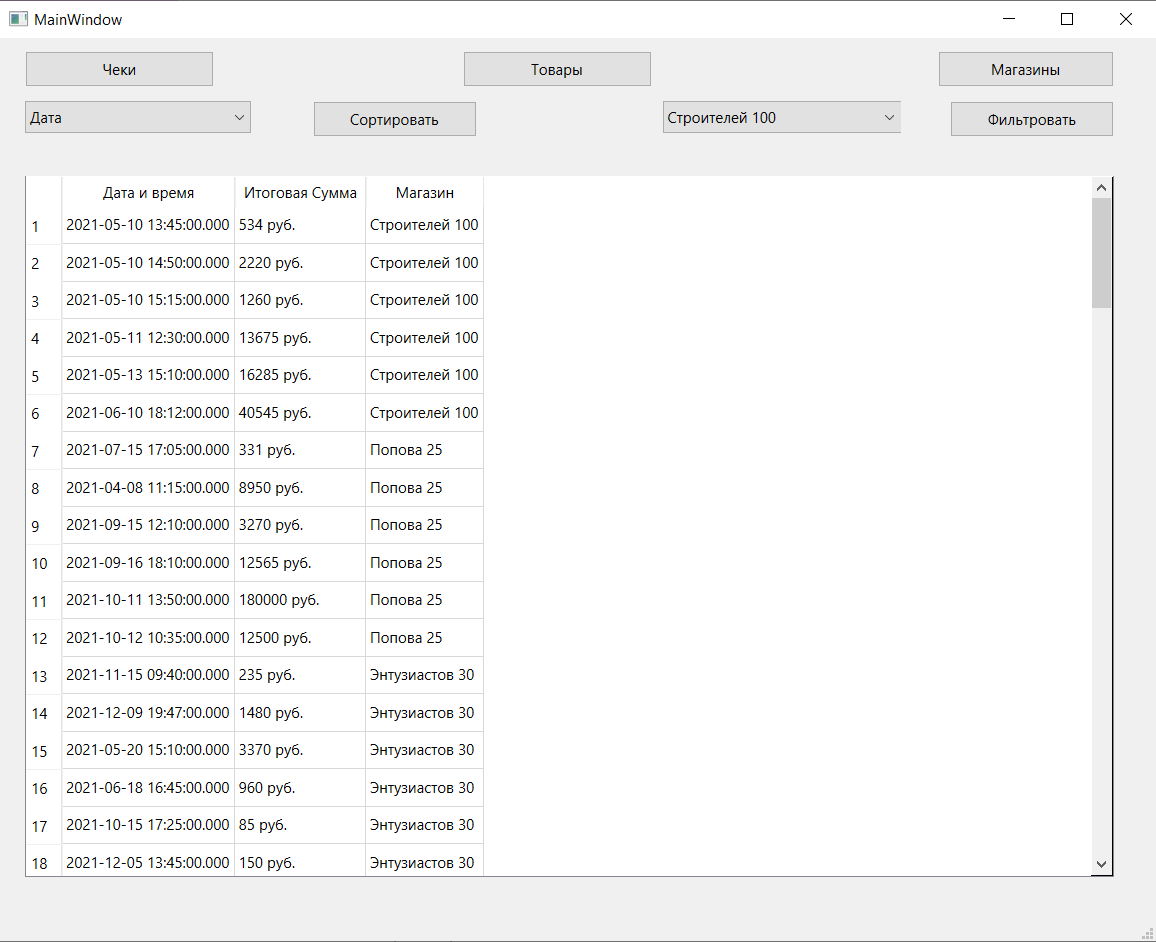


Рис. 1 – Список чеков

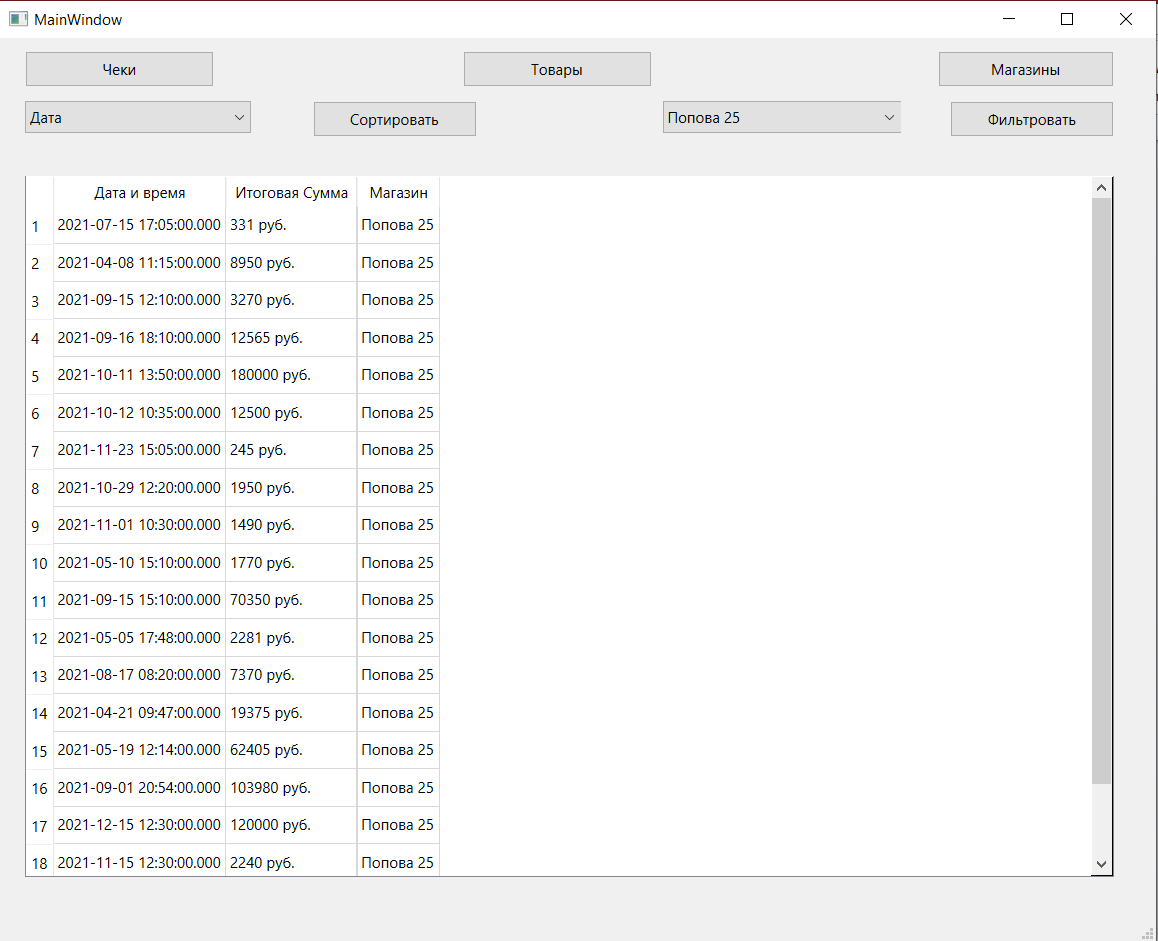


Рис. 2 – Фильтрация чеков

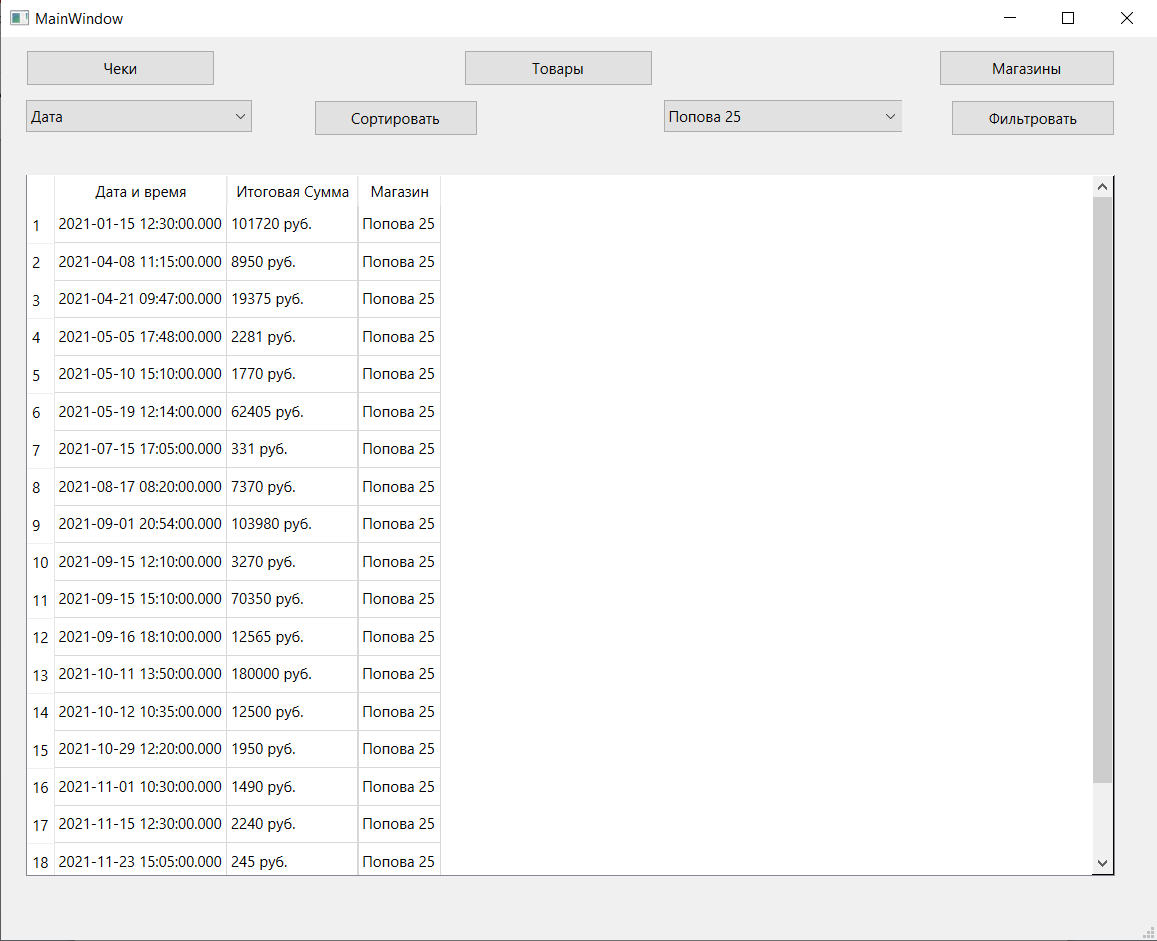


Рис. 3 - Сортировка чеков

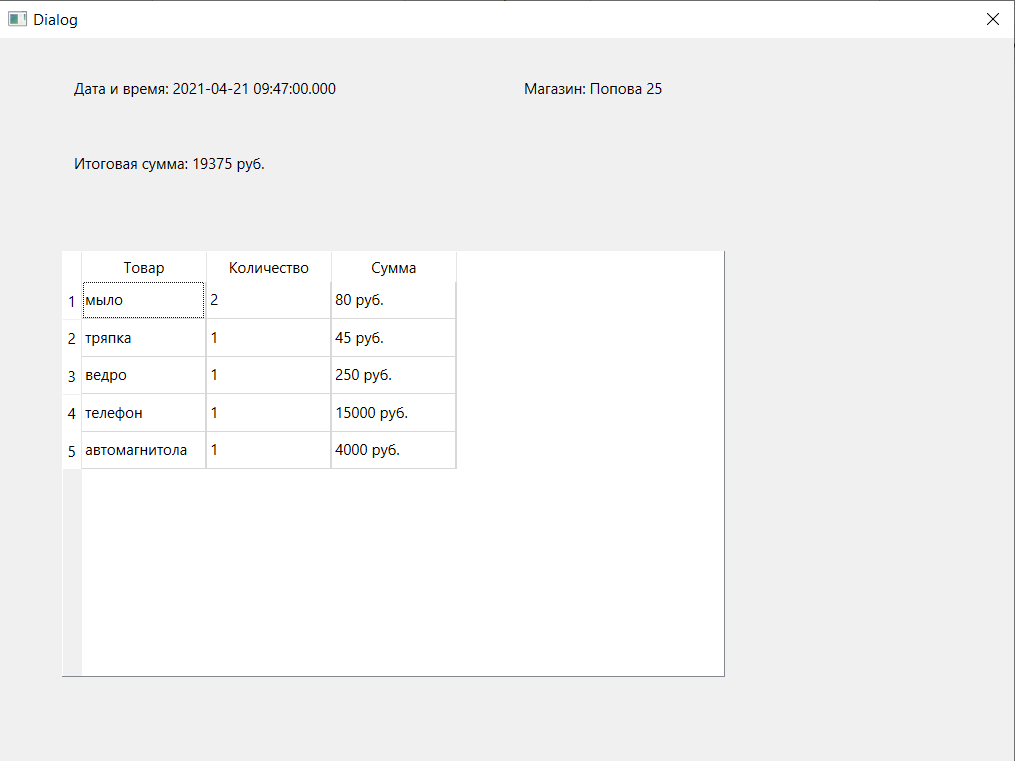


Рис. 4 – Информация о чеке

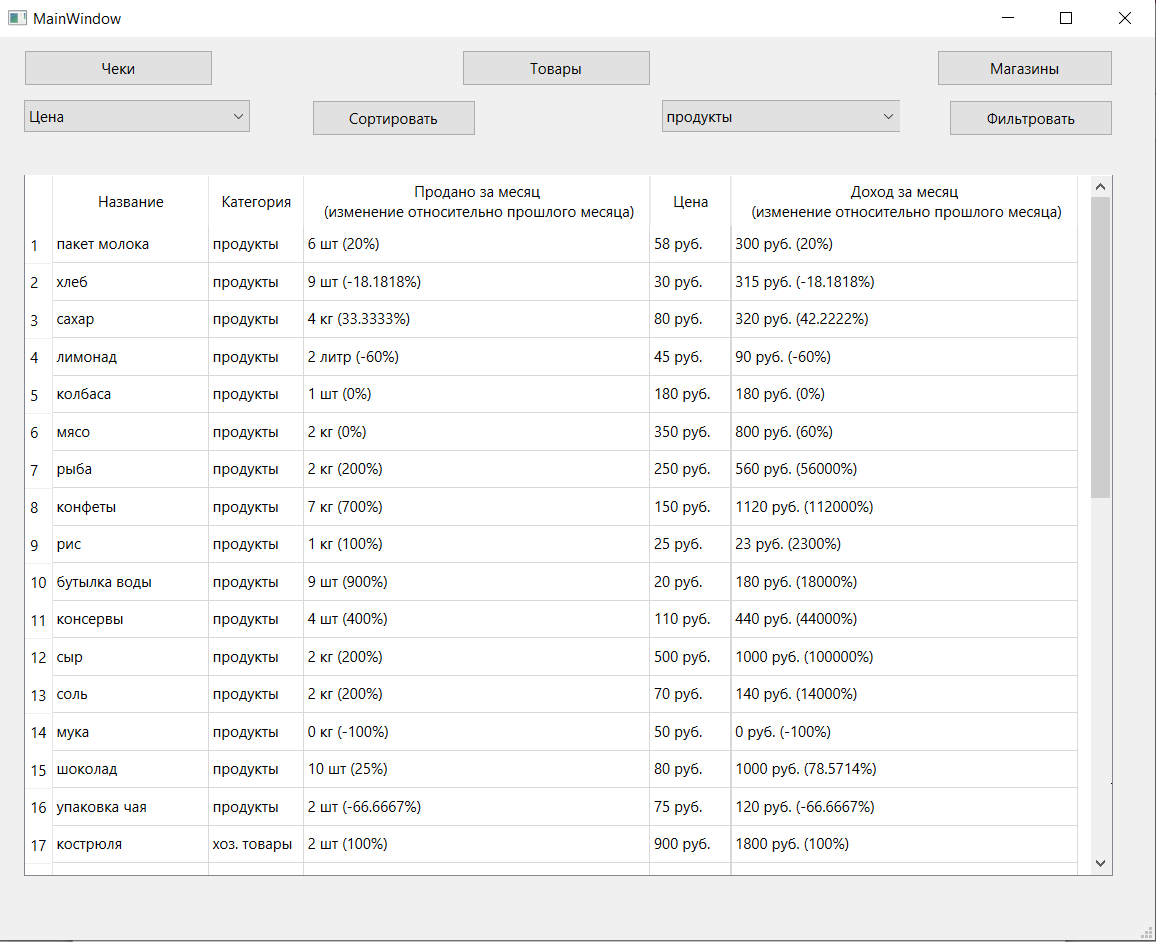


Рис. 5 – Список товаров

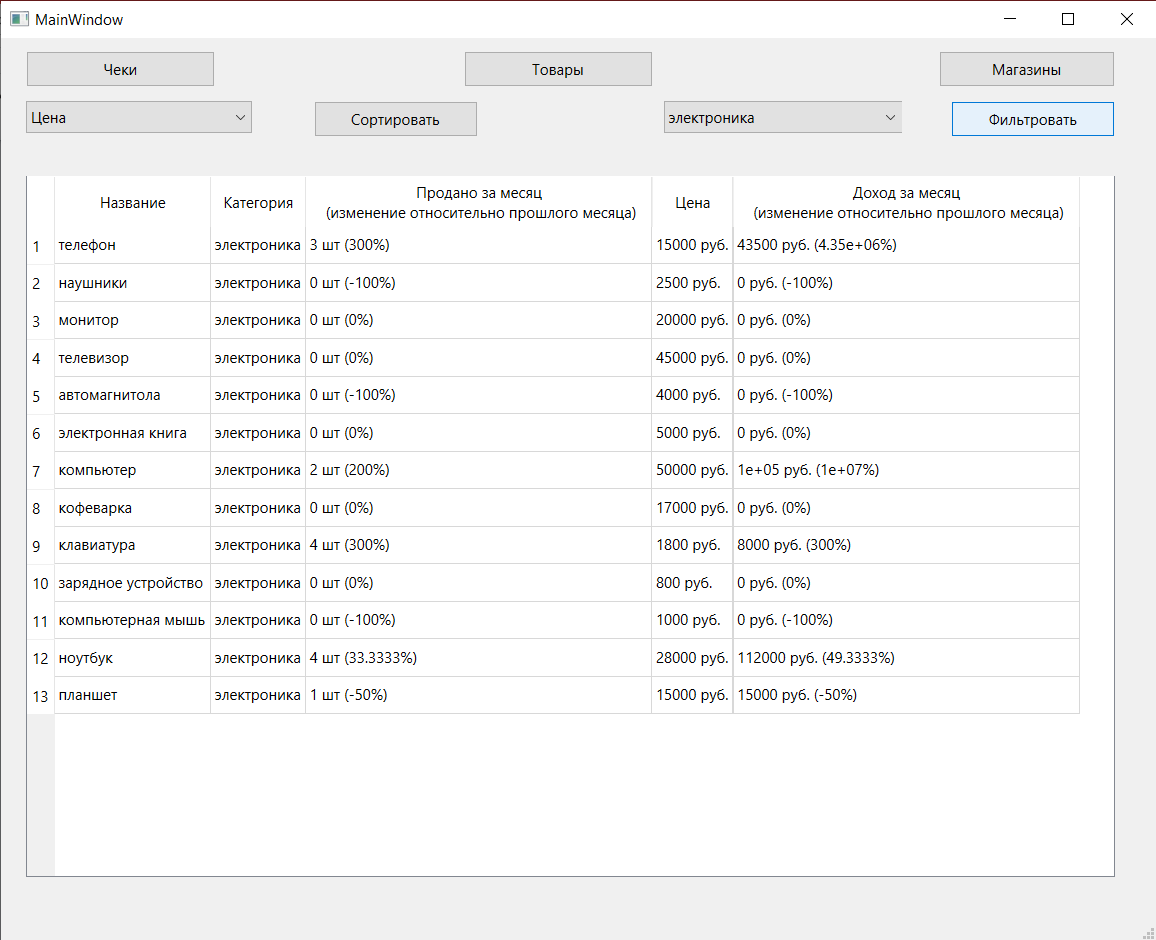


Рис. 6 – Фильтрация товаров

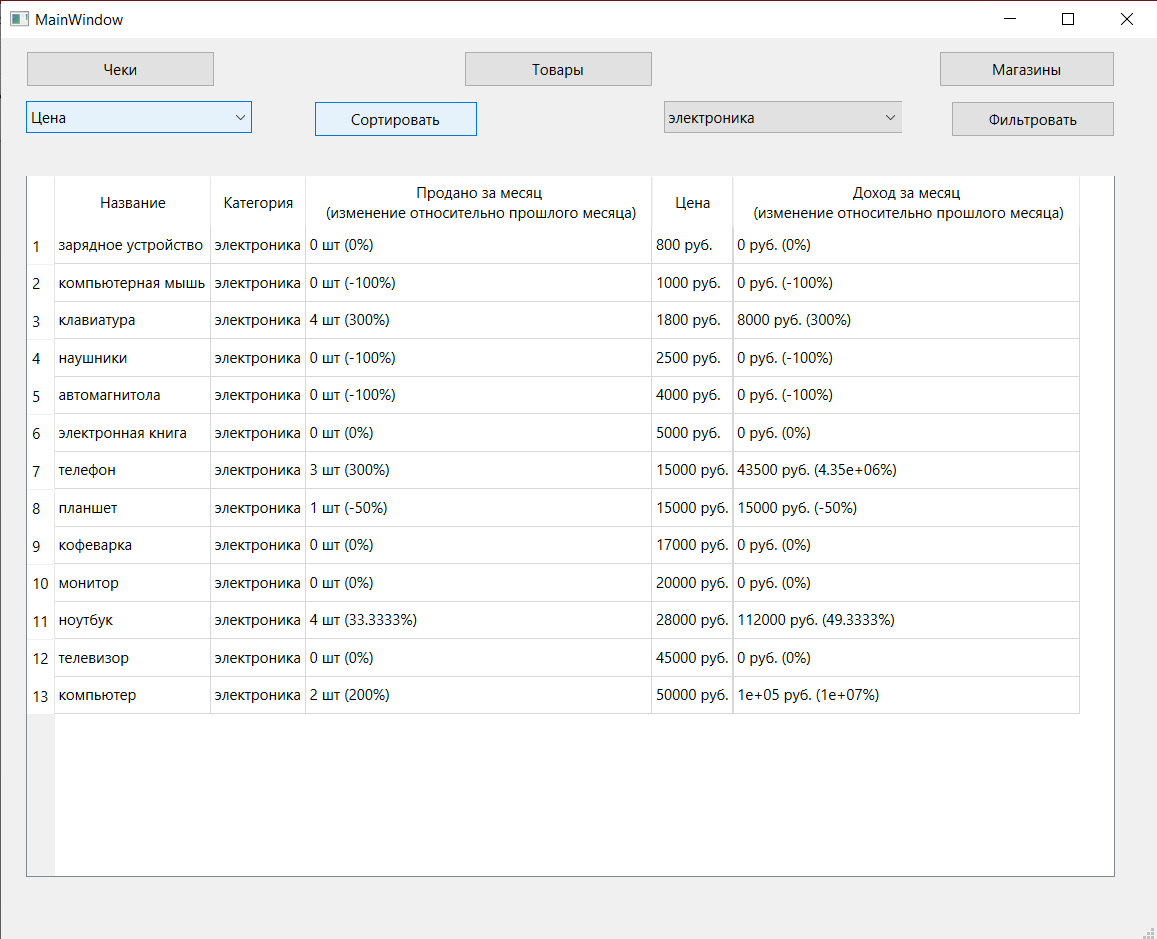


Рис. 7 – Сортировка товаров

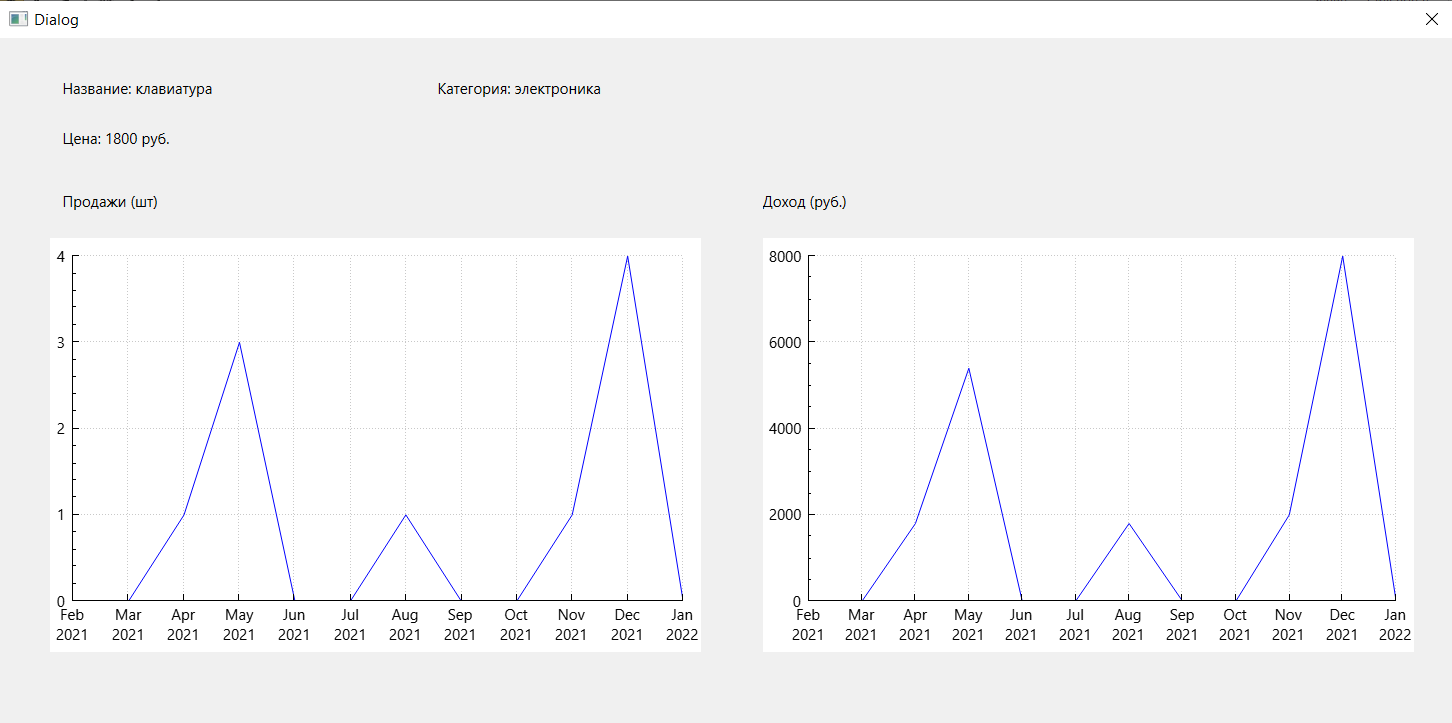


Рис. 8 – Информация о конкретном товаре

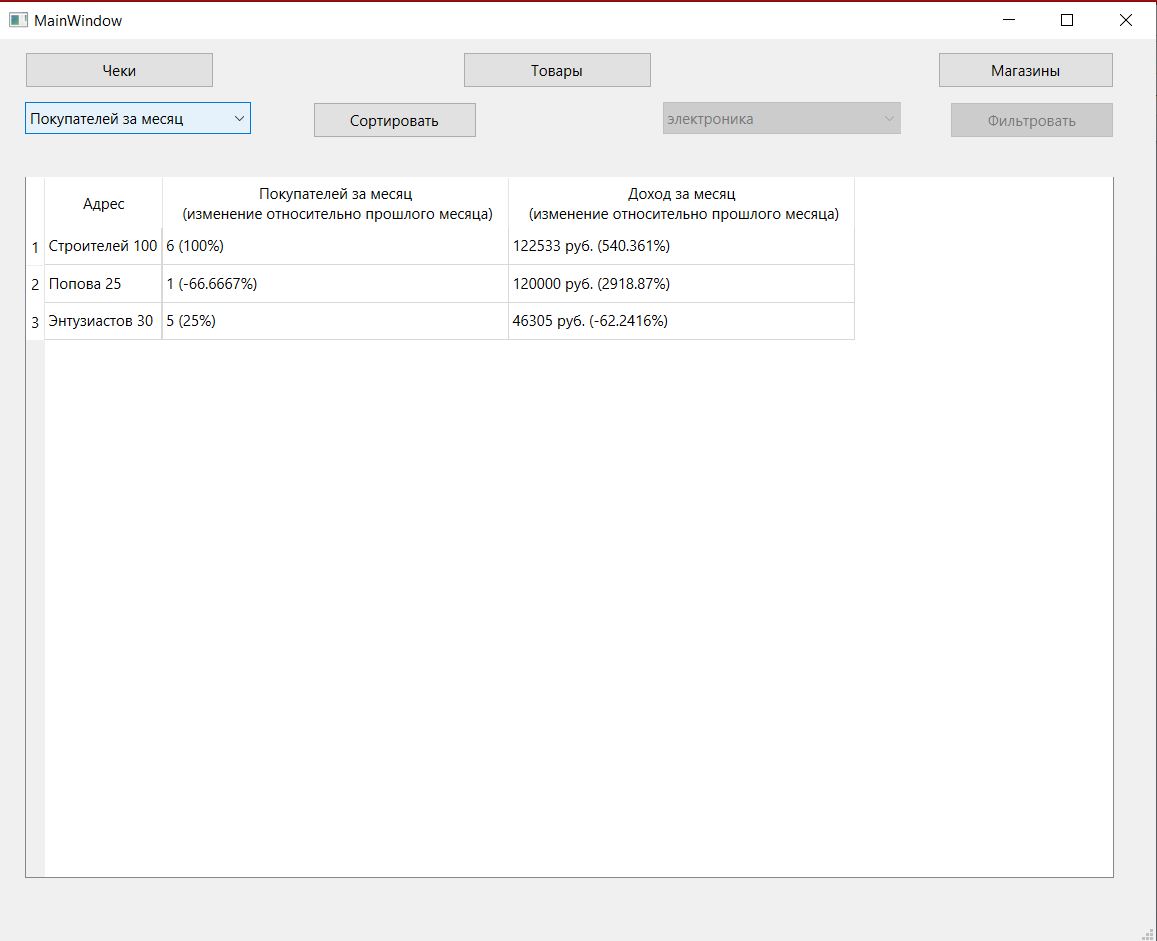


Рис. 9 – Список магазинов

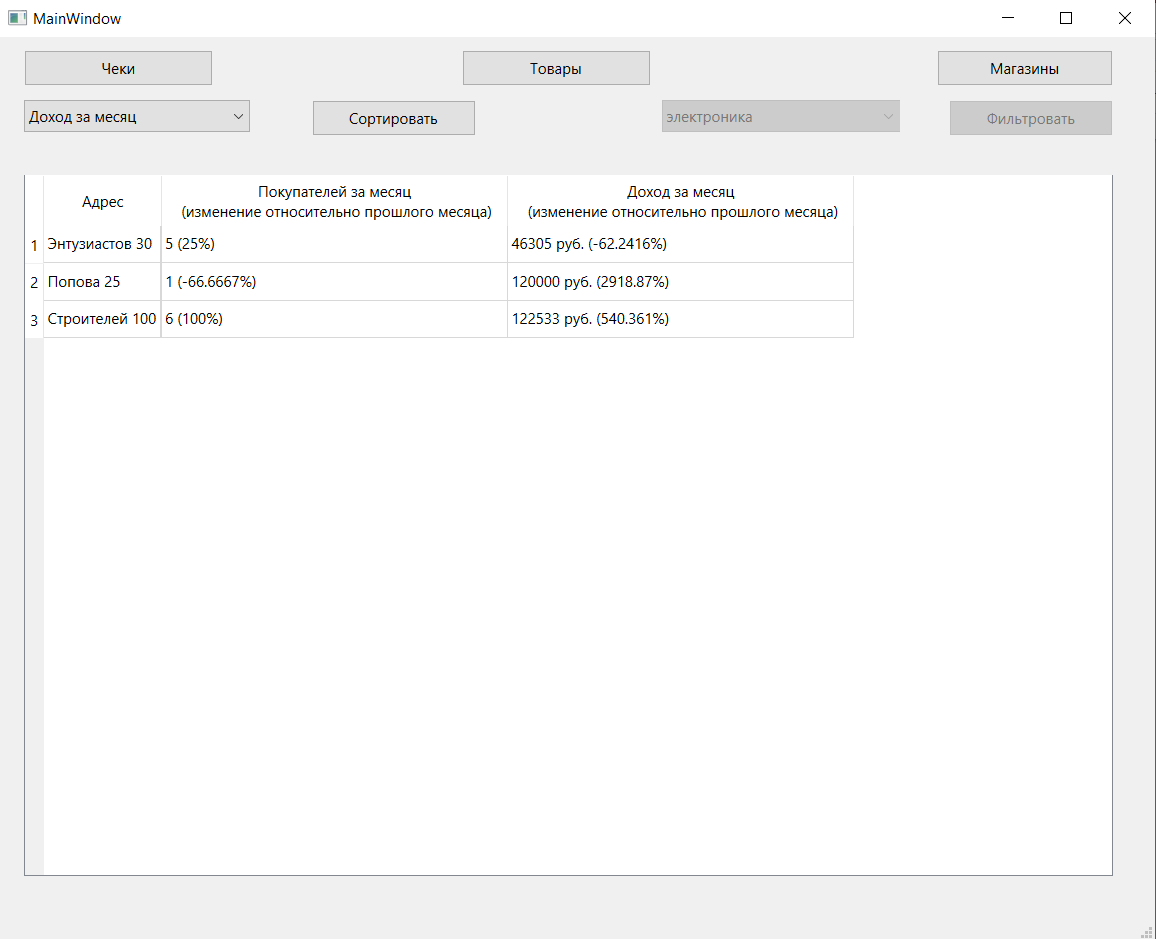


Рис. 10 – Сортировка магазинов

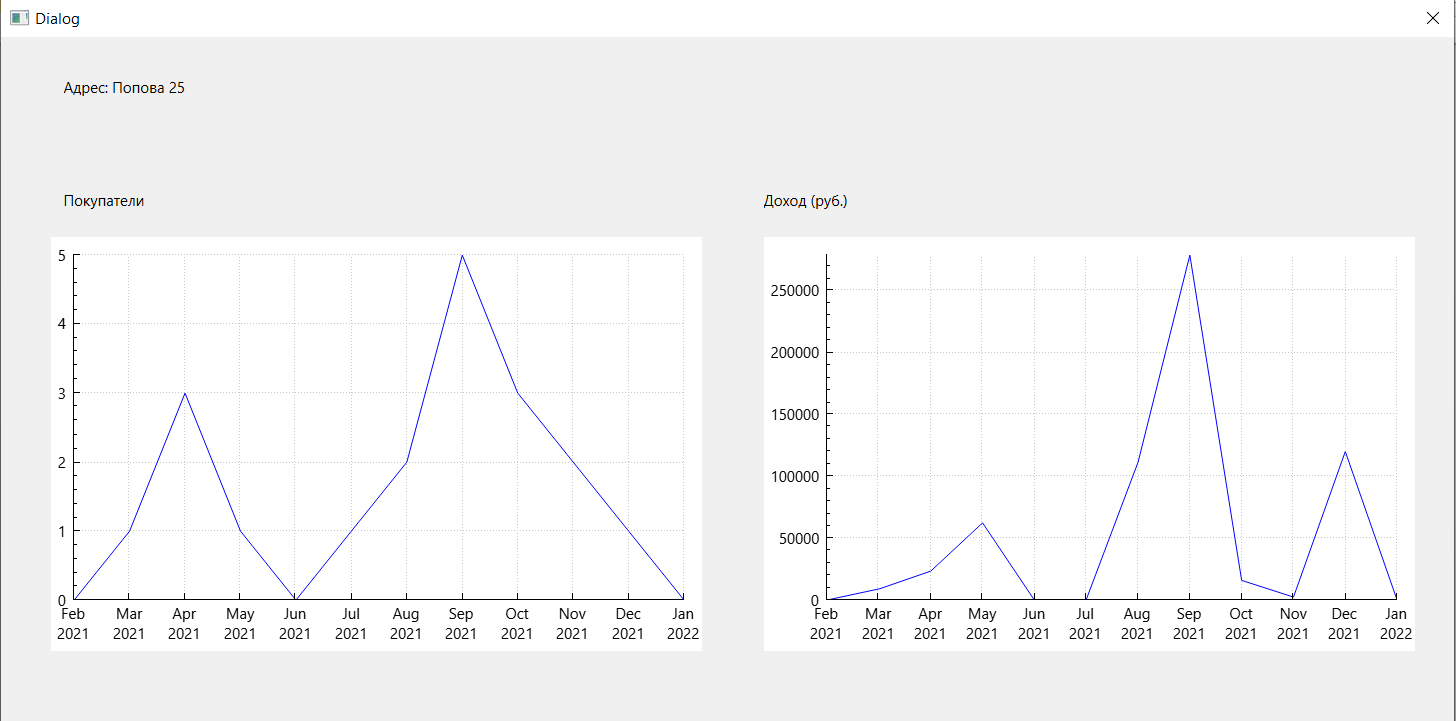


Рис. 11 – Информация о магазине

# Приложение Б. Исходный текст программы

**MainWindow.h**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QMessageBox>

#include <QDateTime>

#include <QListWidgetItem>

#include <infowindow.h>

#include <checkform.h>

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_GoodsButton\_clicked();

void on\_CheckButton\_clicked();

void on\_SortButton\_clicked();

void on\_FiltButton\_clicked();

void on\_ShopButton\_clicked();

void GoodsShow(QString filt, QString sort);

void CheckShow(QString filt, QString sort);

void ShopShow(QString sort);

void on\_tableWidget\_cellDoubleClicked(int row, int column);

private:

Ui::MainWindow \*ui;

QSqlDatabase db;

QString OpenedTabel;

QString Filt;

QString Sort;

};

#endif // MAINWINDOW\_H

**MainWindow.cpp**

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

db = QSqlDatabase::addDatabase("QPSQL");

db.setHostName("localhost");

db.setDatabaseName("Shop");

db.setUserName("postgres");

db.setPassword("1324");

bool ok = db.open();

if(ok){

on\_CheckButton\_clicked();

ui->tableWidget->setShowGrid(true);

ui->tableWidget->setSelectionMode(QAbstractItemView::SingleSelection);

ui->tableWidget->setSelectionBehavior(QAbstractItemView::SelectRows);

ui->tableWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);

}

else{

QMessageBox msBox;

msBox.setText("Не получилось открыть базу данных");

msBox.setWindowTitle("Ошибка");

msBox.exec();

this->setAttribute(Qt::WA\_DeleteOnClose);

this->close();

}

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_GoodsButton\_clicked()

{

GoodsShow("", "");

OpenedTabel = "Product";

ui->SortBox->clear();

ui->SortBox->addItem("Цена");

ui->SortBox->addItem("Продажи за месяц");

ui->SortBox->addItem("Доход за месяц");

ui->FiltBox->setEnabled(1);

ui->FiltButton->setEnabled(1);

ui->FiltBox->clear();

QSqlQuery query("Select prod.\* from \"Product\_Category\" as prod");

while(query.next()){

ui->FiltBox->addItem(query.value(1).toString());

}

Filt = Sort = "";

}

void MainWindow::on\_CheckButton\_clicked()

{

CheckShow("", "");

OpenedTabel = "Receipt";

ui->SortBox->clear();

ui->SortBox->addItem("Дата");

ui->SortBox->addItem("Сумма");

ui->FiltBox->setEnabled(1);

ui->FiltButton->setEnabled(1);

ui->FiltBox->clear();

QSqlQuery query("Select sh.\* from \"Shop\" as sh");

while(query.next()){

ui->FiltBox->addItem(query.value(2).toString());

}

Filt = Sort = "";

}

void MainWindow::on\_ShopButton\_clicked()

{

ShopShow("");

OpenedTabel = "Shop";

ui->SortBox->clear();

ui->SortBox->addItem("Покупателей за месяц");

ui->SortBox->addItem("Доход за месяц");

ui->FiltBox->setDisabled(1);

ui->FiltButton->setDisabled(1);

Filt = Sort = "";

}

void MainWindow::on\_SortButton\_clicked()

{

QString sort("");

QString filt("");

if(OpenedTabel == "Receipt"){

if(ui->SortBox->currentIndex() == 0){

sort = "Order by rep.\"date\_receipt\", rep.\"time\_receipt\"";

}

else if(ui->SortBox->currentIndex() == 1){

sort = "Order by prwd";

}

if(Filt != ""){

filt = "Where sh.\"address\"=\'" + Filt + "\'";

}

CheckShow(filt, sort);

}

else if(OpenedTabel == "Product"){

if(Filt != ""){

filt = "Where cat.\"name\"=\'" + Filt + "\'";

}

if(ui->SortBox->currentIndex() == 0){

sort = "Order by prod.\"price\_without\_discount\"";

}

if(ui->SortBox->currentIndex() == 1){

sort = "Order by cspNow";

}

if(ui->SortBox->currentIndex() == 2){

sort = "Order by iwpNow";

}

GoodsShow(filt, sort);

}

else if(OpenedTabel == "Shop"){

if(ui->SortBox->currentIndex() == 0){

sort = "Order by crwsNow";

}

if(ui->SortBox->currentIndex() == 1){

sort = "Order by siNow";

}

ShopShow(sort);

}

Sort = sort;

}

void MainWindow::on\_FiltButton\_clicked()

{

if(OpenedTabel == "Product"){

GoodsShow("Where cat.\"name\"=\'" + ui->FiltBox->currentText() + "\'", Sort);

Filt = ui->FiltBox->currentText();

}

else if(OpenedTabel == "Receipt"){

CheckShow("Where sh.\"address\"=\'" + ui->FiltBox->currentText() + "\'", Sort);

Filt = ui->FiltBox->currentText();

}

}

void MainWindow::GoodsShow(QString filt, QString sort)

{

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(5);

QStringList header;

header << "Название" << "Категория" << "Продано за месяц \n (изменение относительно прошлого месяца)" << "Цена" <<

"Доход за месяц \n (изменение относительно прошлого месяца)";

ui->tableWidget->setHorizontalHeaderLabels(header);

QString date1, date2, date3;

//date1 = QDateTime::currentDateTime().addMonths(-1).toString();

//date2 = QDateTime::currentDateTime().toString();

//date3 = QDateTime::currentDateTime().addMonths(-2).toString();

date1 = "2021-12-01";

date2 = "2022-01-01";

date3 = "2021-11-01";

double tent1, tent2;

QSqlQuery query("SELECT cat.\"name\", "

"count\_sale\_product(prod.\"name\", \'" + date1 + "\', \'" + date2 + "\') as cspNow, "

"income\_with\_product(prod.\"name\", \'" + date1 + "\', \'" + date2 + "\') as iwpNow, "

"count\_sale\_product(prod.\"name\", \'" + date3 + "\', \'" + date1 + "\') as cspOld, "

"income\_with\_product(prod.\"name\", \'" + date3 + "\', \'" + date1 + "\') as iwpOld, "

"prod.\* "

"From \"Product\" as prod "

"Join \"Product\_Category\" as cat on prod.\"PK\_product\_category\"=cat.\"PK\_product\_category\"" + filt + sort);

int i;

i = 0;

while(query.next()){

tent1 = query.value(3).toDouble();

if(tent1 == 0) tent1++;

tent1 = ((query.value(1).toDouble() - query.value(3).toDouble())/tent1) \* 100;

tent2 = query.value(4).toDouble();

if(tent2 == 0) tent2++;

tent2 = ((query.value(2).toDouble() - query.value(4).toDouble())/tent2) \* 100;

ui->tableWidget->insertRow(i);

ui->tableWidget->setItem(i, 0, new QTableWidgetItem(query.value(6).toString()));

ui->tableWidget->setItem(i, 1, new QTableWidgetItem(query.value(0).toString()));

ui->tableWidget->setItem(i, 2, new QTableWidgetItem(query.value(1).toString() + " " + query.value(8).toString() + " (" + QString::number(tent1) + + "\%)"));

ui->tableWidget->setItem(i, 3, new QTableWidgetItem(query.value(7).toString() + " руб."));

ui->tableWidget->setItem(i, 4, new QTableWidgetItem(query.value(2).toString() + " руб." + " (" + QString::number(tent2) + "\%)"));

i++;

}

ui->tableWidget->resizeColumnsToContents();

}

void MainWindow::CheckShow(QString filt, QString sort)

{

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(3);

QStringList header;

header << "Дата и время" << "Итоговая Сумма" << "Магазин";

ui->tableWidget->setHorizontalHeaderLabels(header);

QSqlQuery query("SELECT price\_receipt\_with\_discount(rep.\"PK\_receipt\") as prwd, "

"sh.\"address\" , rep.\* "

"From \"Receipt\" as rep "

"Join \"Shop\" as sh on rep.\"PK\_shop\"=sh.\"PK\_shop\" " + filt + sort);

int i;

i = 0;

while(query.next()){

ui->tableWidget->insertRow(i);

ui->tableWidget->setItem(i, 0, new QTableWidgetItem(query.value(3).toString() + " " + query.value(4).toString()));

ui->tableWidget->setItem(i, 1, new QTableWidgetItem(query.value(0).toString() + " руб."));

ui->tableWidget->setItem(i, 2, new QTableWidgetItem(query.value(1).toString()));

i++;

}

ui->tableWidget->resizeColumnsToContents();

}

void MainWindow::ShopShow(QString sort)

{

ui->tableWidget->setRowCount(0);

ui->tableWidget->setColumnCount(3);

QStringList header;

header << "Адрес" << "Покупателей за месяц \n (изменение относительно прошлого месяца)"

<< "Доход за месяц \n (изменение относительно прошлого месяца)";

ui->tableWidget->setHorizontalHeaderLabels(header);

QString date1, date2, date3;

//date1 = QDateTime::currentDateTime().addMonths(-1).toString();

//date2 = QDateTime::currentDateTime().toString();

//date3 = QDateTime::currentDateTime().addMonths(-2).toString();

date1 = "2021-12-01";

date2 = "2022-01-01";

date3 = "2021-11-01";

double tent1, tent2;

QSqlQuery query("Select count\_receipt\_with\_shop(sh.\"address\", \'" + date1 + "\', \'" + date2 + "\') as crwsNow, "

"store\_income(sh.\"address\", \'" + date1 + "\', \'" + date2 + "\') as siNow, "

"count\_receipt\_with\_shop(sh.\"address\", \'" + date3 + "\', \'" + date1 + "\') as crwsOld, "

"store\_income(sh.\"address\", \'" + date3 + "\', \'" + date1 + "\') as siOld, sh.\* "

"From \"Shop\" as sh " + sort);

int i;

i = 0;

while(query.next()){

tent1 = query.value(2).toDouble();

if(tent1 == 0) tent1++;

tent1 = ((query.value(0).toDouble() - query.value(2).toDouble())/tent1) \* 100;

tent2 = query.value(3).toDouble();

if(tent2 == 0) tent2++;

tent2 = ((query.value(1).toDouble() - query.value(3).toDouble())/tent2) \* 100;

ui->tableWidget->insertRow(i);

ui->tableWidget->setItem(i, 0, new QTableWidgetItem(query.value(6).toString()));

ui->tableWidget->setItem(i, 1, new QTableWidgetItem(query.value(0).toString() + " (" + QString::number(tent1) + "\%)"));

ui->tableWidget->setItem(i, 2, new QTableWidgetItem(query.value(1).toString() + " руб. (" + QString::number(tent2) + + "\%)"));

i++;

}

ui->tableWidget->resizeColumnsToContents();

}

void MainWindow::on\_tableWidget\_cellDoubleClicked(int row, int column)

{

if(OpenedTabel == "Product"){

InfoWindow \*Form = new InfoWindow;

Form->UpdateInfo(ui->tableWidget->item(row, 0)->text(), "Product");

Form->exec();

}

else if(OpenedTabel == "Shop"){

InfoWindow \*Form = new InfoWindow;

Form->UpdateInfo(ui->tableWidget->item(row, 0)->text(), "Shop");

Form->exec();

}

else if(OpenedTabel == "Receipt"){

CheckForm \*Form = new CheckForm;;

Form->UpdateInfo(ui->tableWidget->item(row, 0)->text(), ui->tableWidget->item(row, 2)->text());

Form->exec();

}

}

**InfoWindow.h**

#ifndef INFOWINDOW\_H

#define INFOWINDOW\_H

#include <QDialog>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QDateTime>

namespace Ui {

class InfoWindow;

}

class InfoWindow : public QDialog

{

Q\_OBJECT

public:

InfoWindow(QWidget \*parent = nullptr);

~InfoWindow();

void UpdateInfo(QString name, QString type);

private:

Ui::InfoWindow \*ui;

};

#endif // INFOWINDOW\_H

**InfoWindow.cpp**

#include "infowindow.h"

#include "ui\_infowindow.h"

InfoWindow::InfoWindow(QWidget \*parent) :

QDialog(parent),

ui(new Ui::InfoWindow)

{

ui->setupUi(this);

}

InfoWindow::~InfoWindow()

{

delete ui;

}

void InfoWindow::UpdateInfo(QString name, QString type)

{

QVector<double> x1, y1, x2, y2;

QVector<double> ticks;

QVector<QString> labels;

if(type == "Product"){

QString date1, date2;

QSqlQuery query("SELECT cat.\"name\", "

"prod.\* "

"From \"Product\" as prod "

"Join \"Product\_Category\" as cat on prod.\"PK\_product\_category\"=cat.\"PK\_product\_category\""

"Where prod.\"name\"=\'" + name + "\'");

query.next();

ui->NameLabel->setText("Название: " + query.value(2).toString());

ui->TypeLabel->setText("Категория: " + query.value(0).toString());

ui->PriceLabel->setText("Цена: " + query.value(3).toString() + " руб.");

ui->firstGraphLabel->setText("Продажи (" + query.value(4).toString() + ")");

ui->SecondGraphLabel->setText("Доход (руб.)");

for(int i = 0; i < 12; i++){

//date1 = QDateTime::currentDateTime().addMonths(-12 + i).toString();

//date2 = QDateTime::currentDateTime().addMonths(-12 + i + 1).toString();

date1 = QDateTime::currentDateTime().addMonths(-18 + i).toString();

date2 = QDateTime::currentDateTime().addMonths(-18 + i + 1).toString();

ticks.push\_back(i + 1);

labels.push\_back(date1.split(" ").value(1) + "\n" + date1.split(" ").value(4));

query.clear();

query.exec("Select "

"count\_sale\_product(prod.\"name\", \'" + date1 + "\', \'" + date2 + "\') as cspNow, "

"income\_with\_product(prod.\"name\", \'" + date1 + "\', \'" + date2 + "\') as iwpNow, "

"prod.\* "

"From \"Product\" as prod "

"Where prod.\"name\"=\'" + name + "\'");

query.next();

x1.push\_back(i + 1);

y1.push\_back(query.value(0).toDouble());

x2.push\_back(i + 1);

y2.push\_back(query.value(1).toDouble());

}

}

else {

QString date1, date2;

QSqlQuery query("SELECT sho.\* "

"From \"Shop\" as sho "

"Where sho.\"address\"=\'" + name + "\'");

query.next();

ui->NameLabel->setText("Адрес: " + query.value(2).toString());

ui->TypeLabel->setText("");

ui->PriceLabel->setText("");

ui->firstGraphLabel->setText("Покупатели");

ui->SecondGraphLabel->setText("Доход (руб.)");

for(int i = 0; i < 12; i++){

//date1 = QDateTime::currentDateTime().addMonths(-12 + i).toString();

//date2 = QDateTime::currentDateTime().addMonths(-12 + i + 1).toString();

date1 = QDateTime::currentDateTime().addMonths(-18 + i).toString();

date2 = QDateTime::currentDateTime().addMonths(-18 + i + 1).toString();

ticks.push\_back(i + 1);

labels.push\_back(date1.split(" ").value(1) + "\n" + date1.split(" ").value(4));

query.clear();

query.exec("Select "

"count\_receipt\_with\_shop(sh.\"address\", \'" + date1 + "\', \'" + date2 + "\') as crwsNow, "

"store\_income(sh.\"address\", \'" + date1 + "\', \'" + date2 + "\') as siNow "

"From \"Shop\" as sh "

"Where sh.\"address\"=\'" + name + "\'");

query.next();

x1.push\_back(i + 1);

y1.push\_back(query.value(0).toDouble());

x2.push\_back(i + 1);

y2.push\_back(query.value(1).toDouble());

}

}

QSharedPointer<QCPAxisTickerText> tex(new QCPAxisTickerText);

tex->addTicks(ticks, labels);

ui->FirstGraph->xAxis->setTicker(tex);

ui->FirstGraph->addGraph();

ui->FirstGraph->graph(0)->addData(x1, y1);

ui->FirstGraph->yAxis->rescale();

ui->FirstGraph->graph(0)->rescaleAxes();

ui->FirstGraph->replot();

ui->SecondGraph->xAxis->setTicker(tex);

ui->SecondGraph->addGraph();

ui->SecondGraph->graph(0)->addData(x2, y2);

ui->SecondGraph->graph(0)->rescaleAxes();

ui->SecondGraph->replot();

}

**Checkform.h**

#ifndef CHECKFORM\_H

#define CHECKFORM\_H

#include <QDialog>

#include <QSqlDatabase>

#include <QSqlQuery>

namespace Ui {

class CheckForm;

}

class CheckForm : public QDialog

{

Q\_OBJECT

public:

explicit CheckForm(QWidget \*parent = nullptr);

~CheckForm();

void UpdateInfo(QString date, QString shop);

private:

Ui::CheckForm \*ui;

};

#endif // CHECKFORM\_H

**Checkform.cpp**

#include "checkform.h"

#include "ui\_checkform.h"

CheckForm::CheckForm(QWidget \*parent) :

QDialog(parent),

ui(new Ui::CheckForm)

{

ui->setupUi(this);

}

CheckForm::~CheckForm()

{

delete ui;

}

void CheckForm::UpdateInfo(QString date, QString shop)

{

QSqlQuery query("SELECT price\_receipt\_with\_discount(rep.\"PK\_receipt\") as prwd, "

"price\_receipt(rep.\"PK\_receipt\") as pr, "

"sh.\"address\" , rep.\* "

"From \"Receipt\" as rep "

"Join \"Shop\" as sh on rep.\"PK\_shop\"=sh.\"PK\_shop\" "

"Where rep.\"date\_receipt\"='" + date.split(" ").value(0) + "' and "

"rep.\"time\_receipt\"='" + date.split(" ").value(1) + "' and sh.\"address\"='" + shop + "'");

query.next();

int rep;

rep = query.value(3).toInt();

ui->dateLabel->setText("Дата и время: " + query.value(4).toString() + " " + query.value(5).toString());

ui->shopLabel->setText("Магазин: " + shop);

ui->discountLabel->setText("Итоговая сумма: " + query.value(0).toString() + " руб.");

ui->tableWidget->setShowGrid(true);

ui->tableWidget->setSelectionMode(QAbstractItemView::SingleSelection);

ui->tableWidget->setSelectionBehavior(QAbstractItemView::SelectRows);

ui->tableWidget->setEditTriggers(QAbstractItemView::NoEditTriggers);

ui->tableWidget->setColumnCount(3);

QStringList header;

header << "Товар" << "Количество" << "Сумма";

ui->tableWidget->setHorizontalHeaderLabels(header);

query.exec("Select prod.\"name\", str.\* "

"From \"String\_Receipt\" as str "

"Join \"Product\" as prod on prod.\"PK\_product\"=str.\"PK\_product\" "

"Where str.\"PK\_receipt\"=" + QString::number(rep));

int i;

i = 0;

while(query.next()){

ui->tableWidget->insertRow(i);

ui->tableWidget->setItem(i, 0, new QTableWidgetItem(query.value(0).toString()));

ui->tableWidget->setItem(i, 1, new QTableWidgetItem(query.value(2).toString()));

ui->tableWidget->setItem(i, 2, new QTableWidgetItem(query.value(7).toString() + " руб."));

i++;

}

}

**Main.cpp**

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}