Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

У Т В Е Р Ж Д А Ю

**Зав. кафедрой ВММБ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Ю. Столбов

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

***ЗАДАНИЕ***

***НА ПОРТФОЛИО***

***по дисциплине***

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Рачев Руслан Игоревич; ИСТ-19-2б

(фамилия, имя, отчество студента; группа)

1. **Тема портфолио**

Корпоративная информационная система PC-Master

2. **Срок сдачи студентом отчета:** « » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_г.

3. **Содержание отчета:**

Техническое задание, хранилище данных, диаграмма классов,

реализация, тестирование, вывод

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (расшифровка)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Пермский национальный** **исследовательский политехнический университет»**

Факультет: Прикладной математики и механики

Кафедра: Вычислительной математики, механики и биомеханики

Направление: 09.03.02 «Информационные системы и технологии»

Профиль бакалавриата: «Информационные системы и технологии»

**ПОРТФОЛИО**

по дисциплине

**«КОРПОРАТИВНЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

Тема: **«Корпоративная информационная система PC-Master»**

Выполнил:

студент гр. ИСТ-19-2б

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Рачев Руслан Игоревич

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(Ф.И.О.)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*(подпись)*

Принял:

ст. преподаватель, Банников Р.Ю.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(должность, ФИО руководителя)*

*\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Пермь 2023**

**Оглавление**

[1 Техническое задание 1](#_Toc132769593)

[2 Хранилище данных 8](#_Toc132769594)

[3 Диаграмма классов 18](#_Toc132769595)

[4 Реализация 20](#_Toc132769596)

[5 Тестирование 32](#_Toc132769597)

[6 Вывод 38](#_Toc132769598)

Техническое задание

1. Общие сведения
   1. Наименование информационной системы

Информационно-справочная система магазина «PC-Master»

* 1. Основания для проведения работ

Работа выполняется на основании договора №159 от 10.02.2023г.

* 1. Наименование организаций - Заказчика и Разработчика
     1. Заказчик

PC-Master;

Адрес фактический: ул. Профессора Поздеева 13, г. Пермь;

Телефон/Факс: +79048450444.

* + 1. Разработчик

ИП «Рачев Р.И.»;

Адрес фактический: ул. Профессора Дедюкина 24, г. Пермь;

Телефон/Факс: +79194497649.

* 1. Порядок оформления и предъявления заказчику результатов работ

Работы по разработке «Информационно-справочная система PC-Master» сдаются Разработчиком поэтапно в соответствии с календарным планом Проекта.

1. Назначение и цели конфигурирования системы
   1. Назначение системы

Система предназначена для повышения эффективности и качества деятельности магазина «PC-Master». Основным назначением системы является автоматизация рабочего места пользователей.

* 1. Цели создания системы

Информационно-справочная система PC-Master создается с целью:

1. Уменьшение возможности ошибок при редактировании и добавлении информации о товарах и информации о поставщиках;
2. Демонстрации данных, обеспечивающее практичность в работе с информацией;
3. Автоматизированная отчетность сведений о сотрудниках, товарах и поставщиках.
4. Требования к системе
   1. Требования к системе в целом
      1. Требования к структуре и функционированию системы

Автоматизированная система должна обеспечивать возможность выполнения следующих функций:

1. Функция аутентификации;
2. Функция, предоставляющая сведения о товарах;
3. Функция, предоставляющая сведения о продажах;
4. Функция, предоставляющая сведения о поставщиках;
5. Функция, предоставляющая сведения о поставках;
6. Функция, предоставляющая сведения о сотрудниках;
7. Функция, обеспечивающая добавление, удаление и изменение товаров в базе;
8. Функция, обеспечивающая добавление, удаление и изменение поставщиков в базе;
9. Функция добавления, удаления и изменения сотрудников в базе;
10. Функция, обеспечивающая добавление, удаление и изменение закупок в базе;
11. Функция, обеспечивающая добавление, удаление и изменение продаж в базе;
12. Функция генерации отчетов – не менее пяти;
13. Формирование отчетов в формате Excel;
14. Реакция системы на ошибочный ввод;
15. Интерфейс должен быть доступным и понятным Пользователю.
    * 1. Требования к численности и квалификации персонала системы и режиму его работы

Для работы системы должен быть выделен ответственный оператор. Им может являться конечный пользователь, знающий предметную область и правила работы с системой.

* + 1. Требования к надежности

Надежное функционирование автоматизированной системы обеспечивается выполнением Заказчиком следующих организационно-технических мероприятий:

1. Организация бесперебойного электропитания технических средств;
2. Использование лицензированного программного обеспечения;
3. Регулярное выполнение рекомендаций Министерства труда и социального развития РФ, изложенных в Постановлении от 23 июля 1998 года об утверждении межотраслевых типовых норм времени на работы по сервисному обслуживанию ПК, и оргтехники, и сопровождению программных средств;
4. Регулярное выполнение требований ГОСТ 51188-98. Защита информации, испытание программных средств на наличие вирусов;
5. Предварительное обучение пользователей и обслуживающего персонала.
6. Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 30-ти минут при условии соблюдения условий эксплуатации технических и программных средств. Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.
7. Проверка выполнения требований по надежности должна производиться на этапе проектирования расчетным путем, а на этапах испытаний и эксплуатации - по методике Разработчика, согласованной с Заказчиком.
   * 1. Требования к эргономике и технической эстетике

Система должна обеспечивать удобный для конечного пользователя интерфейс, отвечающий следующим требованиям:

1. Интерфейс системы должен быть единым;
2. Должно быть обеспечено наличие локализованного (русскоязычного) интерфейса пользователя;
3. В отчетности должен использоваться шрифт Calibri;
4. Размер шрифта отчета должен быть: 11пт;
5. В интерфейсе должен использоваться шрифт Microsoft Sans Serif;
6. Цветовая палитра интерфейса должна быть без использования яркого цвета фона;
7. При возникновении ошибок в работе подсистемы на экран монитора должно выводиться сообщение с наименованием ошибки на русском языке.
   1. Управление системой
      1. Администрирование

Администратор системы будет являться привилегированным пользователем, то есть пользователем, у которого будут права на работу со всей системой.

* + 1. Пользователи системы

В системе должна быть возможность просмотра и изменения данных в справочниках для пользователей (продавцов). Изменение и просмотр данных сотрудников и аналитических отчетов должно быть недоступно для пользователей.

* + 1. Роли пользователей

Роли пользователей используются для разграничения прав пользователей. Будут доступны следующие роли:

1. Администратор;
2. Продавец.
   * 1. Права

На каждую роль в системе будет встроен необходимый набор функций согласно их должностным обязанностям.

* + 1. Продавец

Продавец может добавлять/удалять/изменять записи по товарам, закупкам, поставщикам и продажам.

* + 1. Администратор

Администратор может добавлять/удалять/изменять учетные записи сотрудников. Просматривать все существующие отчеты. Добавлять/удалять/изменять записи по товарам, закупкам, поставщикам и продажам.

* + 1. Отчеты

Отчетные формы для администраторов и пользователей системы, составляются как аналитический отчет по продажам товаров за период, отчет по остаткам товара, отчет о заказанных у поставщиков товаров, отчет по продажам за период по сотрудникам, отчет по продажам за период всех сотрудников в формате xlsx.

* + 1. Отчет по товарам

Интерфейс отчета содержит наименование отчета. Столбцы: номер товара, название, тип товара, поставщик, дата поставки, закупленное количество, проданное количество и оставшееся на складе количество товара.

* + 1. Отчет по продажам товаров

Интерфейс отчета содержит наименование отчета. Столбцы: номер продажи, дата, товар, проданное количество, цена за единицу товара, общая стоимость, продавец.

* + 1. Отчет по закупкам товаров

Интерфейс отчета содержит наименование отчета. Столбцы: номер закупки, дата, товар, закупленное количество, закупочная цена, общая стоимость, поставщик, город поставщика.

* + 1. Отчет по продавцам

Интерфейс отчета содержит наименование отчета. Столбцы: номер сотрудника, ФИО, количество продаж, сумма продаж, прибыль продавца. Строки сортируются по убыванию прибыли.

* + 1. Отчет по прибыли

Интерфейс отчета содержит наименование отчета. Столбцы: номер товара, название, тип товара, закупочная цена, розничная цена, проданное количество, сумма продаж, прибыль. Отчет содержит данные по прибыли с каждого товара.

* + 1. Поставщики

В системе должна быть возможность создания, удаления и изменения учетных записей поставщиков.

* + 1. Товары

В системе должна быть возможность создания, удаления и изменения учетных записей для товара.

1. Инструменты разработки

Предполагается использование следующих инструментов при разработке:

1. VisualStudio Community;
2. MySQL как сервер базы данных;
3. C#, WindowsForm.
4. Состав и содержание работ по созданию системы

Стадии и этапы разработки по созданию КИС PC-Master:

* 1. Стадии разработки

Разработка должна быть проведена в три стадии:

1. Разработка технического задания;
2. Рабочее проектирование;
3. Внедрение.
   1. Этапы разработки

На стадии разработки технического задания должны быть выполнены следующие этапы:

1. Разработка технического задания.
2. Согласование технического задания.
3. Утверждение технического задания.
4. На стадии рабочего проектирования должны быть выполнены следующие этапы:
5. Разработка программы.
6. Разработка программной документации.
7. Испытания программы.
8. На стадии внедрение должны быть выполнены следующие этапы:
9. Подготовка программы.
10. Передача программы.
11. Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1. Постановка задачи.
2. Определение и уточнение требований к техническим средствам.
3. Определение требований к программе.
4. Определение стадий, этапов и сроков разработки программы и документации на нее.
5. Согласование и утверждение технического задания.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями к составу документации.

На этапе тестирования автоматизированной системы должно осуществляться следующим образом:

1. Необходимо проверить точность следования всем алгоритмам.
2. Проверить реакцию системы при вводе некорректных значений.
3. Необходимо проверить корректность добавления, редактирования, удаления данных в системе.

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию на объектах Заказчика.

Таблица 1 Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| **Стадии разработки** | **Этапы работ** | **Содержание работ** | **Время выполнения** |
| Проектирование БД | Проектирование и разработка логической и физической модели БД | Описание предметной области с выделением основных сущностей и связей между ними | 10.02.2023- 16.02.2023 |
| Проектирование интерфейса | Проектирование интерфейса | Спроектировать интерфейс согласно обозначенному функционалу | 17.02.2023-23.02.2023 |
| Проектирование структуры классов | Разработка структуры классов | Разработать структуры классов для интерфейсной и функциональной части | 24.02.2023-02.03.2023 |
| Реализация | Программная реализация интерфейсной и функциональной части | Запрограммировать обозначенный функционал с учетом созданной БД и разработанных структур классов | 03.03.2023- 16.03.2023 |
| Тестирование | Испытание продукта | Проведение тестирования разработанного приложения | 17.03.2023-23.03.2023 |

1. Порядок контроля и приемки системы

После передачи Исполнителем отдельного функционального модуля программы Заказчику, последний имеет право тестировать модуль в течение 7 дней. После тестирования Заказчик должен принять работу по данному этапу или в письменном виде изложить причину отказа от принятия. В случае обоснованного отказа Исполнитель обязуется доработать модуль. В противном случае после проведения испытаний Исполнитель совместно с Заказчиком подписывает Акт приемки-сдачи автоматизированной системы в эксплуатацию.

Таблица 2. Виды, состав, объем и методы испытаний системы и ее составных частей

|  |  |  |  |
| --- | --- | --- | --- |
| **Контрольные вопросы** | **Показатель качества** | **Код** | **Вид тестирования** |
| Соответствует ли реализация функций программного обеспечения задачам пользователя? Насколько полно автоматизированы задачи пользователя? | Пригодность | 1.1 | Функциональное |
| Насколько функционирование программного обеспечения и получаемые результаты (число десятичных знаков, округление) соответствуют требованиям приложения? | Правильность | 1.2 | Функциональное |
| Насколько легко и эффективно осуществляется взаимодействие с другим программным обеспечением в среде пользователя? | Способность к взаимодействию | 1.3 | Интеграционное |
| Обеспечивает ли программное обеспечение средства санкционирования доступа и выполняет ли требования приложения? | Защищенность | 1.4. | Функциональное, Стресс-тестирование |
| Функционирует ли система надежно в соответствии с требованиями поддержки приложения и технологичности, включая управление аномалиями (с оценкой средств управления аномалиями: определение ошибочных ситуаций системы и условий, требующих специальной обработки для подтверждения целостности системы; особенности восстановления и работы в условиях неполной работоспособности)? | Надежность | 2 | Функциональное, Стресс-тестирование |
| Применимо ли программное обеспечение в заданной операционной и поддерживающей среде? | Практичность | 3 | Интеграционное |
| Функционирует ли система эффективно, минимизируя издержки, с минимальным временем отклика и максимальной производительностью системы (с оценкой использования данных, оценкой эффективности по памяти, оценкой выполнения итераций и проверкой требований технологичности)? | Эффективность | 4 | Функциональное |
| Насколько легко исправлять ошибки и устранять недостатки? Насколько легко расширять возможности или технологию путем развития существующих функций или добавления новых функций или данных? | Изменяемость | 5.1 | Стресс-тестирование |
| Насколько легко переносить программное обеспечение для использования в другой среде (конфигурация КТС и/или среда программной системы)? | Мобильность | 6.1 | Интеграционное |

1. Требования к документированию

Для системы на различных стадиях создания должны быть выпущены следующие документы из числа предусмотренных в ГОСТ 34.201-89 «Информационная технология. Комплекс стандартов на автоматизированные системы».

Виды, комплектность и обозначения документов при создании Информационно-справочная система PC-Master приведены в таблице 5:

Таблица 3. Виды, комплектность и обозначения документов при создании Информационно-справочная система PC-Master

|  |  |  |
| --- | --- | --- |
| Наименование документа | Код документа | Часть проекта |
| Техническое задание | ТЗ | Предпроектное исследование |
| Инструкция пользователя | ИП | Рабочее проектирование |

1. Источники разработки

Настоящее Техническое Задание разработано на основе следующих документов и информационных материалов:

1. ГОСТ 24.701-86 «Надежность автоматизированных систем управления»
2. ГОСТ 15150-69 «Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды»
3. ГОСТ 21958-76 «Система «Человек-машина». Зал и кабины операторов. Взаимное расположение рабочих мест. Общие эргономические требования»
4. ГОСТ 12.1.004-91 «ССБТ. Пожарная безопасность. Общие требования»
5. ГОСТ Р. 50571.22-2000 «Электроустановки зданий».

Хранилище данных

Предметная область

Для повышения эффективности деятельности любого магазина, необходим строгий учет и контроль всех процессов. Магазину PC-Master необходима информационно-справочная система, содержащая полную информацию о сотрудниках, товарах, а также его поставках и продажах.

Помимо хранения, нужна возможность для ввода, редактирования и удаления информации из базы данных. Также для администраторов системы необходима функция формирования следующих итоговых отчетов:

1. Отчет по продажам;
2. Отчет по закупкам;
3. Отчет по остаткам товаров;
4. Отчет по продажам сотрудников;
5. Отчет по прибыли.

Без возможности хранения и обработки всей информации по функционированию магазина PC-Master есть малая вероятность успешности данного предприятия. Автоматизация процесса учета в магазине PC-Master позволит уменьшить ручной труд, погасить процент ошибок, снизить время на обработку информации и работу с документацией.

Логическая модель базы данных

На основе данной предметной области было составлена логическая модель со связями (рис.1).

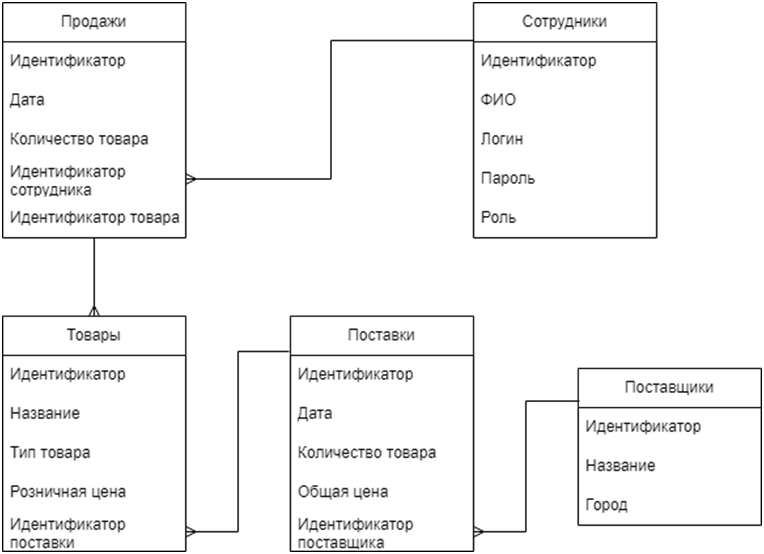


Рисунок 1 – Логическая модель базы данных

Физическая модель базы данных

Для разработки программного продукта была выбрана СУБД MySQL Workbench. Ее функциональные возможности определены следующим образом:

1. Интерпретация баз данных в виртуальной среде
2. Изменение таблиц в графическом редакторе
3. Создание связей между таблицами
4. Реализация построения ER-диаграмм из скриптов SQL, которые создавались ранее

MySQL Workbench позволяет выполнять простейшие операции с данными:

1. Добавить в таблицу одну или несколько записей;
2. Удалить из таблицы одну или несколько записей;
3. Обновить значения некоторых полей в одной или нескольких записях;
4. Найти одну или несколько записей, удовлетворяющих заданному условию.

На основе предметной области и логической модели базы данных была составлена физическая модель базы данных. Физическая модель диаграммы базы данных (рис.2).

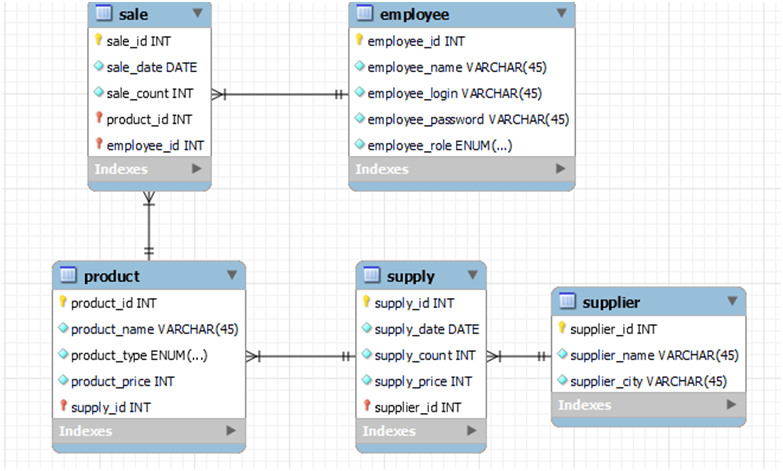


Рисунок 2 – Физическая модель базы данных

Описание основных сущностей и типы полей

1. Таблица «Поставщики» - supplier. Таблица включает в себя такие поля как идентификатор, название, город поставщика.

|  |  |
| --- | --- |
| **Столбец** | **Тип данных** |
| supplier\_id | PK, INT |
| supplier\_name | VARCHAR (45) |
| supplier\_city | VARCHAR (45) |

1. Таблица «Поставки» – supply. Таблица включает в себя такие поля как идентификатор, дата поставки, количество товара, стоимость поставки и идентификатор поставщика.

|  |  |
| --- | --- |
| **Столбец** | **Тип данных** |
| supply\_id | PK, INT |
| supply\_date | DATE |
| supply\_count | INT |
| supply\_price | INT |
| supplier\_id | FK, INT |

1. Таблица «Товары» – product. Таблица включает в себя такие поля как идентификатор, название товара, тип товара, цена, идентификатор поставки.

|  |  |
| --- | --- |
| **Столбец** | **Тип данных** |
| product\_id | PK, INT |
| product\_name | VARCHAR (45) |
| product\_type | ENUM |
| product\_price | INT |
| supply\_id | FK, INT |

1. Таблица «Продажи» - sale. Таблица является информацией о продажах товаров, которые связаны с продавцами. Таблица включает в себя столбцы идентификатор, дата продажи, количество, идентификатор товара, идентификатор продавца.

|  |  |
| --- | --- |
| **Столбец** | **Тип данных** |
| sale\_id | PK, INT |
| sale\_date | DATE |
| sale\_count | INT |
| product\_id | FK, INT |
| employee\_id | FK, INT |

1. Таблица «Сотрудники» – employee. Таблица включает в себя такие поля как идентификатор, ФИО, логин, пароль, роль сотрудника.

|  |  |
| --- | --- |
| **Столбец** | **Тип данных** |
| employee\_id | PK, INT |
| employee\_name | VARCHAR (100) |
| employee\_login | VARCHAR (100) |
| employee\_password | VARCHAR (100) |
| employee\_role | ENUM |

Тип данных для идентификаторов (первичные и внешние ключи) выбран INT, который занимает 8 бит памяти. Выбран такой тип данных т.к. записей в таблице может быть огромное количество. Для строковых типов данных заданы размеры – 45 и 100, например, для наименования поставщика, товара, роли пользователя т.к. названия могут быть длинными. Для дат выбран тип данных DATE, содержащий день, месяц, год. Для информации «о количестве» задан целочисленный тип данных INT, т.к. количество может быть до нескольких тысяч.

Связи между сущностями в таблице базы данных:

|  |  |  |
| --- | --- | --- |
| **Название таблицы 1** | **Название таблицы 2** | **Связь между таблицами** |
| supplier | supply | Один ко многим идентифицирующая связь |
| supply | product | Один ко многим идентифицирующая связь |
| product | sale | Один ко многим идентифицирующая связь |
| employee | sale | Один ко многим идентифицирующая связь |

Содержимое базы данных

Заполненная таблица «Поставщики» (рис.3)

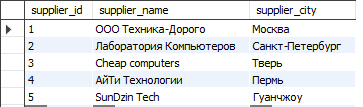


Рисунок 3 – Таблица «Поставщики»

Заполненная таблица «Поставки» (рис.4)

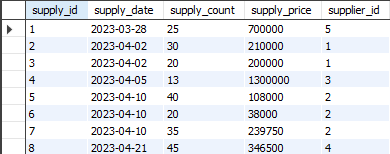


Рисунок 4 – Таблица «Поставки»

Заполненная таблица «Товары» (рис.5)

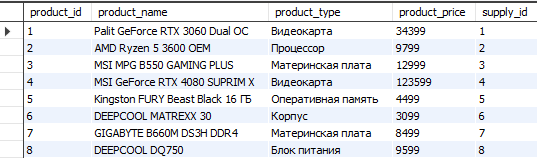


Рисунок 5 – Таблица «Товары»

Заполненная таблица «Продажи» (рис.6)

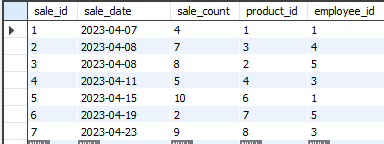


Рисунок 6 – Таблица «Продажи»

Заполненная таблица «Сотрудники» (рис.7)



Рисунок 7 – Таблица «Сотрудники»

Код заполнения базы данных находится в приложении 1.

Постановка задачи

### Назначение системы

Система предназначена для повышения эффективности и качества деятельности магазина «PC-Master». Основным назначением является автоматизация рабочего места пользователей системы.

### Цели создания системы

Информационно-справочная система создается с целью:

* Уменьшение возможности ошибок при редактировании и добавлении информации о товарах и информации о поставщиках;
* Демонстрация данных, обеспечивающая практичность в работе с информацией;
* Автоматизированная отчетность сотрудников, товаров и поставщиков.

### Требования к структуре и функционированию системы

Автоматизированная система должна обеспечивать возможность выполнения следующих функций:

* Функция авторизации и аутентификации;
* Функция, обеспечивающая добавление, изменение и удаление товаров;
* Функция, предоставляющая сведения о товарах магазина;
* Функция, обеспечивающая добавление, изменение и удаление информации о поставках товаров;
* Функция, предоставляющая сведения о поставках товаров;
* Функция, обеспечивающая добавление, изменение и удаление информации о поставщиках;
* Функция, предоставляющая сведения о поставщиках;
* Функция, обеспечивающая добавление, изменение и удаление информации о продажах;
* Функция, предоставляющая сведения о продажах;
* Функция, обеспечивающая добавление, изменение и удаление информации о сотрудниках магазина;
* Функция, предоставляющая сведения о сотрудниках магазина;
* Функция формирования отчетов – не менее пяти;
* Функция выгрузки отчетов в формате Excel;
* Реакция системы на ошибочный ввод;
* Интерфейс должен быть доступным и понятным пользователю.

Проектирование интерфейса

Окно авторизации системы для всех пользователей системы (рис.10)

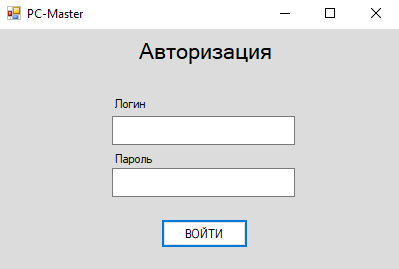


Рисунок 10 – Окно авторизации системы

Окно справочника «Сотрудники» (рис.11). Данную страницу видит только Администратор системы. Данные о сотруднике в виде таблицы.

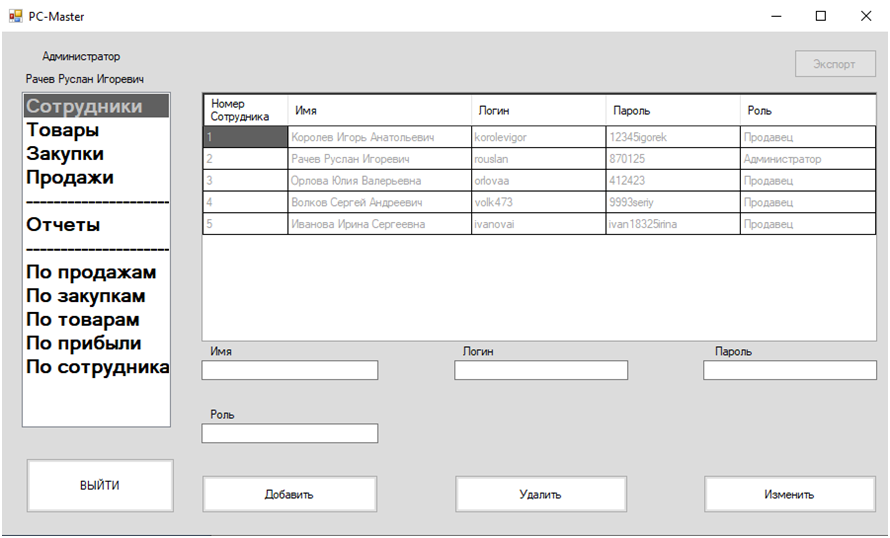


Рисунок 11 – Справочник «Сотрудники»

Справочник «Товары». Доступ на добавление, изменение, удаление к этому справочнику есть у администратора и сотрудника (рис.12).

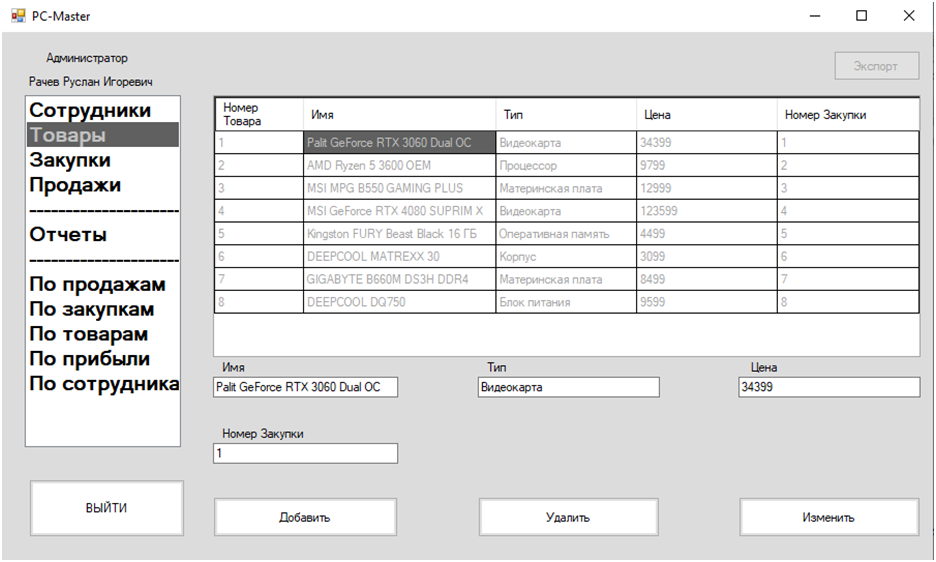


Рисунок 12 ­– Справочник «Товары»

Ответом на ошибочный ввод является вывод окна ошибки, с просьбой проверить введенные пользователем данные (рис.13)

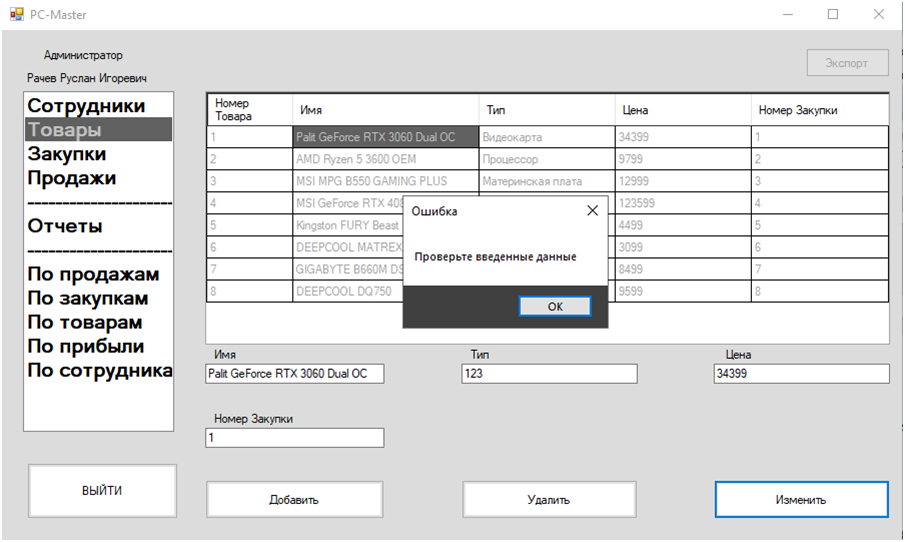


Рисунок 13 – Вывод окна ошибки

На рисунке 14 представлен интерфейс одного из предоставляемых отчетов. Доступ к отчетам есть только у администраторов системы.

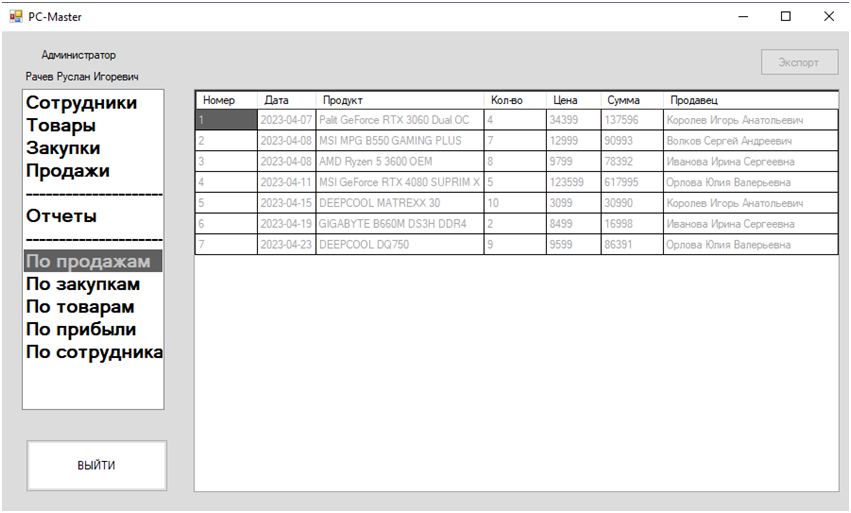


Рисунок 14 – Отчет по продажам товаров

Все представленные отчеты можно выгрузить в формате xlsx по нажатию кнопки «Экспорт» в правом верхнем углу. Пример выгруженного отчета (рис.15):

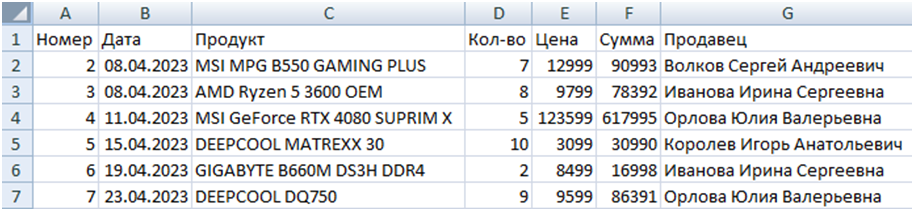


Рисунок 15 – Отчет по продажам в формате xlsx

Диаграмма классов

Проектирование диаграммы классов

Диаграмма классов представлена на рисунке 1.

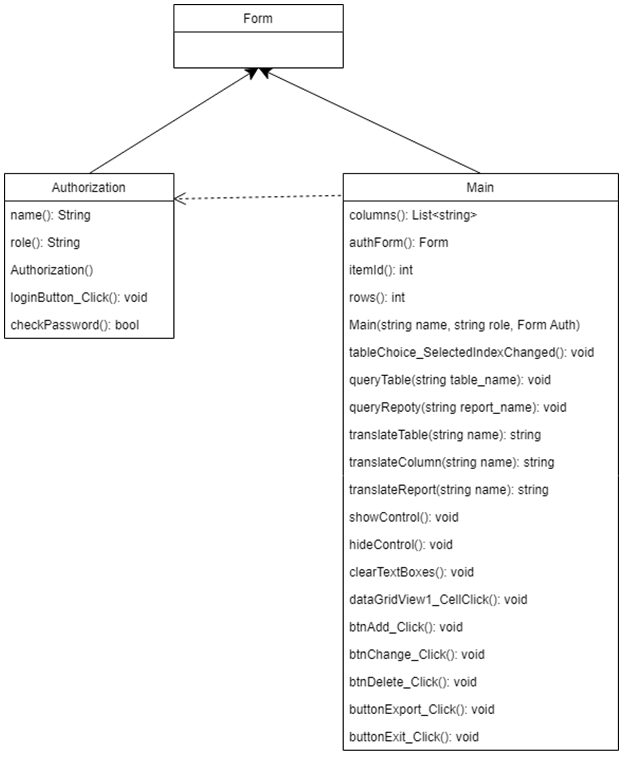


Рисунок 1 – Диаграмма классов информационной системы PC-Master

Сущности структуры классов

1. Authorization – класс для авторизации пользователя системы путем сравнения логина и пароля в базе данных. Содержит следующие методы:

* name – Получение значения фамилии имя и отчества пользователя;
* role – Получение значения роли пользователя;
* Authorization – Процедура создания формы Authorization;
* loginButton\_Click – Получение введенных пользователем данных для авторизации, проверка их достоверности и переход к форме Main;
* checkPassword – Запрос к базе данных и сравнение введенного логина и пароля.

2. Main – основной класс для работы пользователя с системой. Содержит следующие методы:

* columns – Получение списка колонок таблицы;
* authForm – Скрытая форма авторизации;
* itemId – Получение порядкового номера выбранной строки таблицы;
* rows – Получение количества строк таблицы;
* Main – Процедура создания формы Main;
* tableChoice\_SelectedIndexChanged – Процедура выбора запрашиваемой таблицы или отчета;
* queryTable – Процедура запроса к базе данных для отображения выбранной таблицы;
* queryReport – Процедура запроса к базе данных для отображения выбранной отчета;
* translateTable – Процедура для перевода названия таблицы;
* translateColumn – Процедура для перевода названия колонок таблицы;
* translateReport – Процедура для перевода названия отчета;
* showControl – Процедура для отображения части интерфейса, отвечающей за изменение таблицы;
* hideControl – Процедура для скрытия части интерфейса, отвечающей за изменение таблицы;
* clearTextBoxes – Процедура для очистки полей для ввода данных;
* btnAdd\_Click – Процедура запроса к базе данных для добавления новой строки в таблицу;
* btnChange\_Click – Процедура запроса к базе данных для изменения записей в таблице;
* btnDelete\_Click – Процедура запроса к базе данных для удаления строк из таблицы;
* buttonExport\_Click – Процедура для выгрузки отчета в формат xlsx;
* buttonExit\_Click – Процедура для выхода из системы и перехода обратно к странице авторизации;

3. Form – класс представляющий окно-интерфейс приложения.

Связи между классами:

|  |  |  |
| --- | --- | --- |
| Имя класса 1 | Имя класса 2 | Связь |
| Form | Authorization | Наследование |
| Form | Main | Наследование |
| Main | Authorization | Зависимость |

Реализация

Функции, которые должны быть реализованы в системе:

1. Функция авторизации

Внешний интерфейс для функции авторизации представлен на рисунке 1. В данном окне пользователю необходимо ввести логин и пароль для того, чтобы войти в приложение.

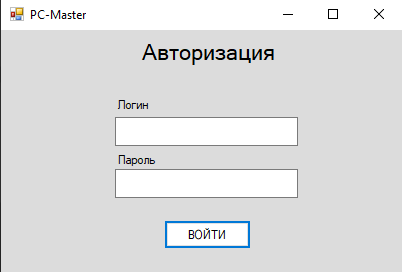


Рисунок 1 – Интерфейс функции авторизации

Программный код для аутентификации с помощью запроса к базе данных:

private bool checkPassword()

{

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

string sql = "Select \* from employee WHERE employee\_login='"+loginInput.Text+"' AND employee\_password='"+passwordInput.Text+"'";

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

name = Convert.ToString(reader.GetValue(1));

role = Convert.ToString(reader.GetValue(4));

}

}

}

Console.WriteLine(name);

conn.Close();

conn.Dispose();

if (name == "") return false;

else return true;

}

1. Функция для просмотра записей в базе данных

В системе должна быть функция для просмотра данных в базе по всем товарам, продажам, закупкам, поставщикам и сотрудникам магазина. Интерфейс просмотра справочника представлен на рисунке 2.

Программный код для отображения базы:

private void queryTable(String table\_name)

{

itemId = 0;

dataGridView1.Columns.Clear();

dataGridView1.Rows.Clear();

columns.Clear();

clearTextBoxes();

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

table\_name = translateTable(table\_name);

string sql = "Select \* from " + table\_name;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

columns = new List<string>();

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

int count = 0;

DataTable schema = null;

schema = reader.GetSchemaTable();

foreach (DataRow col in schema.Rows)

{

string сname = translateColumn(col.Field<String>("ColumnName"));

dataGridView1.Columns.Add(col.Field<String>("ColumnName").ToUpper(), сname);

columns.Add(col.Field<String>("ColumnName"));

count++;

}

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;

for (int i = 0; i < count; i++)

dataGridView1.Columns[i].Width = (int)(dataGridView1.Width \* 1 / count);

dataGridView1.Columns[0].Width -= 50;

dataGridView1.Columns[1].Width += 50;

dataGridView1.Columns[count - 1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

int row = 0;

while (reader.Read())

{

dataGridView1.Rows.Add();

for (int column = 0; column < count; column++)

{

string value = "";

if (reader.GetValue(column).GetType() == typeof(DateTime))

{

DateTime time = (DateTime) reader.GetValue(column);

value = time.ToString("yyyy-MM-dd");

}

else value = Convert.ToString(reader.GetValue(column));

dataGridView1.Rows[row].Cells[column].Value = value;

}

row++;

}

rows = row;

}

}

conn.Close();

conn.Dispose();

showControl();

}

1. Функция добавления, изменения и удаления записей из базы данных

В системе должна быть функция, обеспечивающая добавление, изменение и удаление записей в каждой таблице. При закупке нового товара или при устройстве нового сотрудника, его данные должны быть добавлены в базу, также и при его увольнении и прочих событиях данные должны быть актуализированы.

Интерфейс ИС «PC-Master» для добавления, удаления и изменения записей в базе данных (рис. 2).

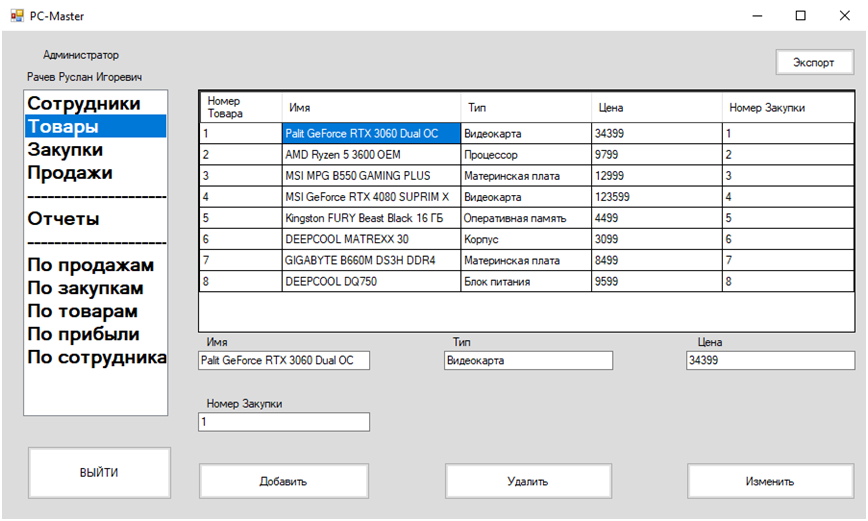


Рисунок 2 – Пример интерфейса для раздела «Товары»

Для добавления записи в базу необходимо ввести данные во все предложенные поля, после чего нажать кнопку «Добавить». После этого отображаемая таблица обновится и в ней появится новая строка.

Программный код для добавления записи:

private void btnAdd\_Click(object sender, EventArgs e)

{

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

String table\_name = translateTable(tableChoice.SelectedItem.ToString());

String clmns = "";

for (int i = 1; i < columns.Count; i++)

{

clmns += columns[i] + ", ";

}

clmns = clmns.Substring(0, clmns.Length - 2);

String text = "";

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

for (int i = 0; i < columns.Count - 1; i++)

{

text += "'" + textBoxes[i].Text + "', ";

}

text = text.Substring(0, text.Length - 2);

string sql = "Insert into " + table\_name + " (" + clmns + ") values (" + text + ")";

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка добавлена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

conn.Close();

conn.Dispose();

queryTable(table\_name);

}

Для удаления нужно щёлкнуть по необходимой строке и нажать на кнопку «Удалить». После этого отображаемая таблица обновится и из нее исчезнет удаленная строка.

Программный код для удаления записи:

private void btnDelete\_Click(object sender, EventArgs e)

{

if (itemId != 0)

{

String table\_name = translateTable(tableChoice.SelectedItem.ToString());

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

string sql = "DELETE FROM " + table\_name + " WHERE " + columns[0] + "=" + itemId;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка удалена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

conn.Close();

queryTable(table\_name);

} else MessageBox.Show("Ни одна строка не выбрана!", "Ошибка");

}

Для изменения строки таблицы, необходимо кликнуть на необходимую строку, данные ее столбцов отобразятся в полях снизу. После этого нужно изменить данные строки на необходимые данные и нажать кнопку «Изменить».

Программный код для изменения записи:

private void btnChange\_Click(object sender, EventArgs e)

{

if (itemId != 0)

{

String table\_name = translateTable(tableChoice.SelectedItem.ToString());

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

String text = "";

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

for (int i = 1; i < columns.Count; i++)

{

text += columns[i] + "='" + textBoxes[i - 1].Text + "',";

}

text = text.Substring(0, text.Length - 2);

string sql = "UPDATE " + table\_name + " SET " + text +

" WHERE " + columns[0] + "=" + itemId;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка изменена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

conn.Close();

queryTable(table\_name);

} else MessageBox.Show("Ни одна строка не выбрана!", "Ошибка");

}

1. Функция генерации отчетов

В системе присутствует 5 видов отчета (рис.3). Каждый из них предназначен для отображения своего набора данных.

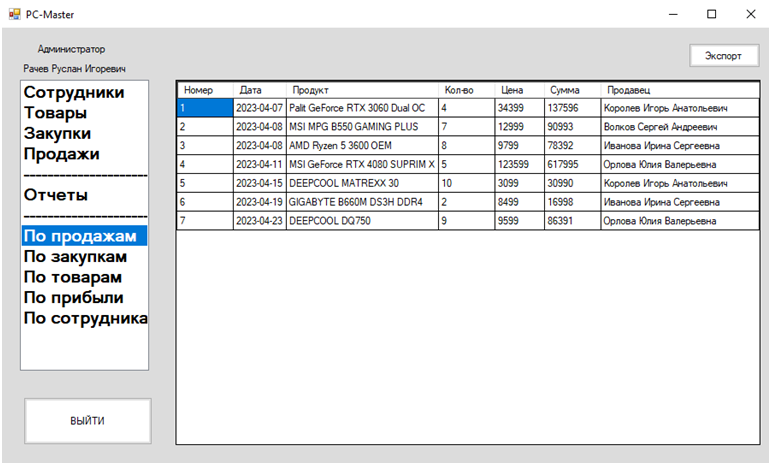


Рисунок 3 – Пример интерфейса для отображения отчета

Отчет по продажам предоставляет данные по всем совершенным продажам.

SQL-код для отчета по продажам:

CREATE VIEW sales\_report

as SELECT

sale.sale\_id as "Номер",

sale.sale\_date as "Дата",

product.product\_name as "Продукт",

sale.sale\_count as "Кол-во",

product.product\_price as "Цена",

(sale.sale\_count\*product.product\_price) as "Сумма",

employee.employee\_name as "Продавец"

FROM sale

JOIN product ON sale.product\_id = product.product\_id

JOIN employee ON sale.employee\_id = employee.employee\_id;

Отчет по закупкам предоставляет данные по всем совершенным закупкам.

SQL-код для отчета по закупкам:

CREATE VIEW supplies\_report

as SELECT

supply.supply\_id as "Номер",

supply.supply\_date as "Дата",

product.product\_name as "Продукт",

supply.supply\_count as "Кол-во",

REPLACE(FORMAT((supply.supply\_price/supply.supply\_count), '######'), ",", "") as "Цена закупки",

supply.supply\_price as "Сумма",

supplier.supplier\_name as "Поставщик",

supplier.supplier\_city as "Город"

FROM supply

JOIN product ON supply.supply\_id = product.supply\_id

JOIN supplier ON supply.supplier\_id = supplier.supplier\_id

ORDER BY supply.supply\_date;

Отчет по товарам предоставляет данные по всем товарам на складе.

SQL-код для отчета по товарам:

CREATE VIEW products\_report

as SELECT

product.product\_id as "Номер",

product.product\_name as "Продукт",

product.product\_type as "Тип",

supplier.supplier\_name as "Поставщик",

supply.supply\_date as "Дата поставки",

supply.supply\_count as "Кол-во",

SUM(sale.sale\_count) as "Продано",

(supply.supply\_count-SUM(sale.sale\_count)) as "На складе"

FROM product

JOIN supply ON supply.supply\_id = product.supply\_id

JOIN supplier ON supply.supplier\_id = supplier.supplier\_id

JOIN sale ON sale.product\_id = product.product\_id

GROUP BY product.product\_id, supply.supply\_id, supplier.supplier\_id, sale.sale\_id;

Отчет по прибыли предоставляет данные по прибыли, полученной с продаж каждого товара.

SQL-код для отчета по прибыли:

CREATE VIEW profits\_report

as SELECT

product.product\_id as "Номер",

product.product\_name as "Продукт",

product.product\_type as "Тип",

REPLACE(FORMAT((supply.supply\_price/supply.supply\_count), '######'), ",", "") as "Закупочная цена",

product.product\_price as "Розничная цена",

SUM(sale.sale\_count) as "Продано",

(SUM(sale.sale\_count)\*product.product\_price) as "Сумма продаж",

(SUM(sale.sale\_count)\*(product.product\_price-REPLACE(FORMAT((supply.supply\_price/supply.supply\_count), '######'), ",", ""))) as "Прибыль"

FROM product

JOIN supply ON supply.supply\_id = product.supply\_id

JOIN supplier ON supply.supplier\_id = supplier.supplier\_id

JOIN sale ON sale.product\_id = product.product\_id

GROUP BY product.product\_id, supply.supply\_id, supplier.supplier\_id, sale.sale\_id;

Отчет по сотрудникам предоставляет данные обо всех работниках магазина.

SQL-код для отчета по сотрудникам:

CREATE VIEW employees\_report as

SELECT

abc.employee\_id as "Номер",

abc.employee\_name as "ФИО",

SUM(abc.summ\_count) as "Кол-во продаж",

SUM(abc.summ\_price) as "Сумма продаж",

SUM(abc.summ\_profit) as "Заработал"

from(

SELECT

employee.employee\_id,

employee.employee\_name,

SUM(sale.sale\_count) as summ\_count,

(SUM(sale.sale\_count)\*product.product\_price) as summ\_price,

(SUM(sale.sale\_count)\*(product.product\_price-REPLACE(FORMAT((supply.supply\_price/supply.supply\_count), '######'), ",", ""))) as summ\_profit

FROM employee

JOIN sale ON sale.employee\_id = employee.employee\_id

JOIN product ON sale.product\_id = product.product\_id

JOIN supply ON supply.supply\_id = product.supply\_id

GROUP BY product.product\_id, sale.sale\_id, supply.supply\_id, supply.supply\_price, supply.supply\_count, employee.employee\_id) as abc

GROUP BY abc.employee\_id;

Каждый из разработанных отчетов необходимо отображать в приложении.

Программный код для отображения отчетов:

private void queryReport(String report\_name)

{

dataGridView1.Columns.Clear();

dataGridView1.Rows.Clear();

columns.Clear();

clearTextBoxes();

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

report\_name = translateReport(report\_name);

string sql = "Select \* from " + report\_name;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

columns = new List<string>();

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

int count = 0;

DataTable schema = null;

schema = reader.GetSchemaTable();

foreach (DataRow col in schema.Rows)

{

string сname = col.Field<String>("ColumnName");

dataGridView1.Columns.Add(col.Field<String>("ColumnName").ToUpper(), сname);

count++;

}

for (int i = 0; i < count; i++)

dataGridView1.Columns[i].Width = (int)(dataGridView1.Width \* 1 / count);

if (report\_name == "sales\_report" || report\_name == "employees\_report")

dataGridView1.Columns[count - 1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

int row = 0;

while (reader.Read())

{

dataGridView1.Rows.Add();

for (int column = 0; column < count; column++)

{

string value = "";

if (reader.GetValue(column).GetType() == typeof(DateTime))

{

DateTime time = (DateTime)reader.GetValue(column);

value = time.ToString("yyyy-MM-dd");

}

else value = Convert.ToString(reader.GetValue(column));

dataGridView1.Rows[row].Cells[column].Value = value;

}

row++;

}

rows = row;

}

}

conn.Close();

conn.Dispose();

}

1. Функция выгрузки отчетов в формат Excel

В программе должна быть функция выгрузки отчета в формат xlsx. Для переноса необходимого отчета необходимо нажать кнопку «Экспорт».

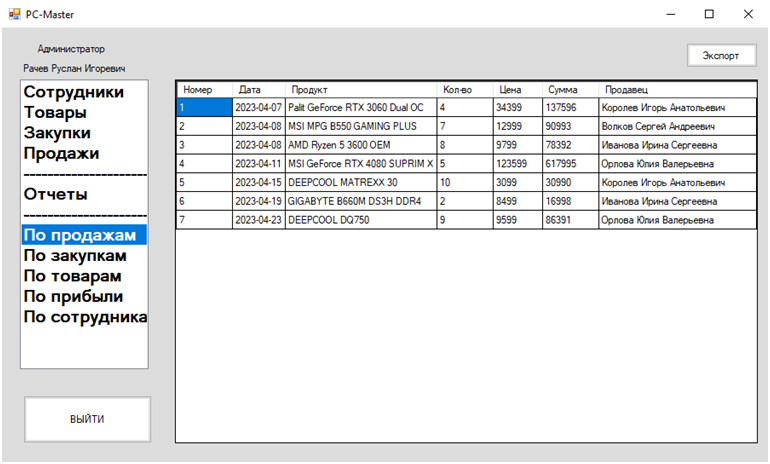


Рисунок 4 – Интерфейс для выгрузки отчета

Программный код для выгрузки отчетов:

private void buttonExport\_Click(object sender, EventArgs e)

{

Excel.Application ExcelApp = new Excel.Application();

Excel.Workbook ExcelWorkBook;

Excel.Worksheet ExcelWorkSheet;

ExcelWorkBook = ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);

ExcelWorkSheet = (Excel.Worksheet)ExcelWorkBook.Worksheets.get\_Item(1);

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

ExcelApp.Cells[1, j + 1] = dataGridView1.Columns[j].HeaderText.ToString();

ExcelApp.Cells[i + 2, j + 1] = dataGridView1.Rows[i].Cells[j].Value;

}

}

ExcelApp.Visible = true;

ExcelApp.UserControl = true;

ExcelApp.Rows.AutoFit();

ExcelApp.Columns.AutoFit();

}

После выгрузки открывается Excel и отчет автоматически копируется в новый документ (рис.5). После этого, сделав все необходимые изменения, пользователь может сохранить отчет в формате xlsx.

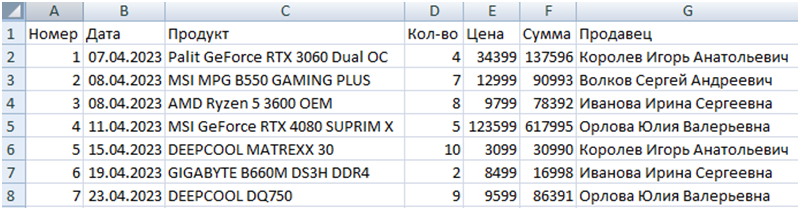


Рисунок 5 – Просмотр отчета в Excel

1. Реакция системы на ошибочный ввод.

При неправильном вводе выдается сообщение при аутентификации пользователя (рис.6), при не правильном типе данных выводится надпись (рис.7).

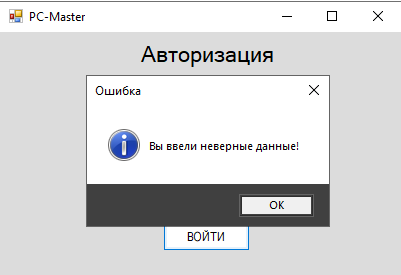


Рисунок 6 – Вывод ошибки при авторизации

Программный код реакции на ошибку при авторизации:

if (loginInput.Text != "" && passwordInput.Text != "")

{

bool userExists = checkPassword();

if (userExists)

{

loginInput.Text = "";

passwordInput.Text = "";

this.Hide();

Main mainForm = new Main(name, role, this);

mainForm.Show();

}

else

{

MessageBox.Show("Вы ввели неверные данные!",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

else

{

MessageBox.Show("Введите логин и пароль!",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

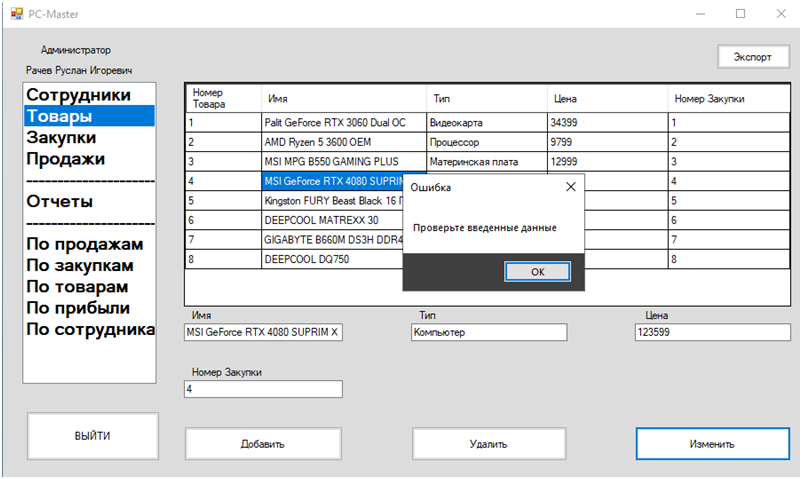


Рисунок 7 – Вывод ошибки в типе данных

Программный код реакции на ошибку в типе данных:

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка добавлена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

Тестирование

Обозначение объекта тестирования

В системе будет проведено тестирование:

1. Функции авторизации;
2. Функции формирования каждого из отчетов;
3. Функции выгрузки в Excel;
4. Реакция системы на ошибочный ввод данных авторизации и добавления записей в таблицу.

В рамках обозначенных объектов тестирования будет проведено функциональное тестирование. После него будет проведено нефункциональное тестирование по интерфейсу системы.

Тестирование будет проводить тестировщик Рачев Р.И.

Функциональное тестирование

## Функция авторизации

Для проверки корректности работы авторизации, зайдем в систему в качестве администратора, с соответствующим логином и паролем (рис.1).

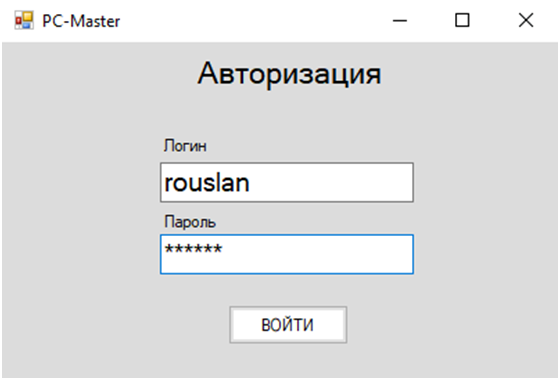


Рисунок 1 – Вход в роли администратора

Вход в систему выполнен под администратором, это видно в левом верхнем углу, где на экран выведено ФИО и роль пользователя. В этой роли отображены вкладки сотрудники, товары, закупки, продажи, поставщики, а также все отчеты (рис. 2). В каждой вкладке кроме отчетов можно добавлять, удалять и изменять данные.



Рисунок 2 – Интерфейс в роли администратора

Попробуем зайти в роли продавца магазина, используя данные для авторизации одного из пользователей системы (рис. 3).

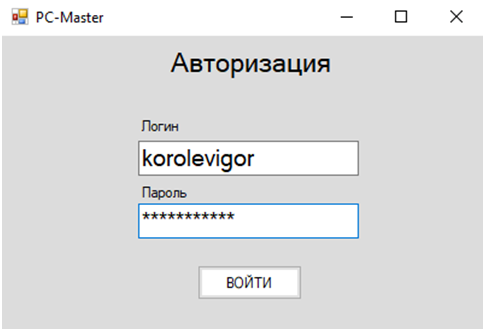


Рисунок 3 – Вход в роли сотрудника

Вход в систему выполнен под сотрудником, это видно в левом верхнем углу где на экран выведено ФИО и роль пользователя. В этой роли пользователю доступны вкладки товары, закупки и продажи (рис. 4). В каждой из вкладок можно добавлять, удалять и изменять данные



Рисунок 4 – Интерфейс в роли сотрудника

Авторизация и разграничение ролей работает корректно. Программа верно определяет пользователя, под которым осуществляется вход и его данные (роль и ФИО). Также пользователям корректно выдаются права на просмотр и редактирование таблиц, доступных им в зависимости от их роли.

## Функции формирования отчетов

В системе есть возможность формировать любой из 5 представленных отчетов. Данная возможность предоставляется только пользователям с ролью «Администратор». Разные отчеты отображают разные наборы данных обо всех процессах магазина. Для вывода на экран одного из них, сотруднику необходимо нажать на соответствующую вкладку в панели слева (рис.5).

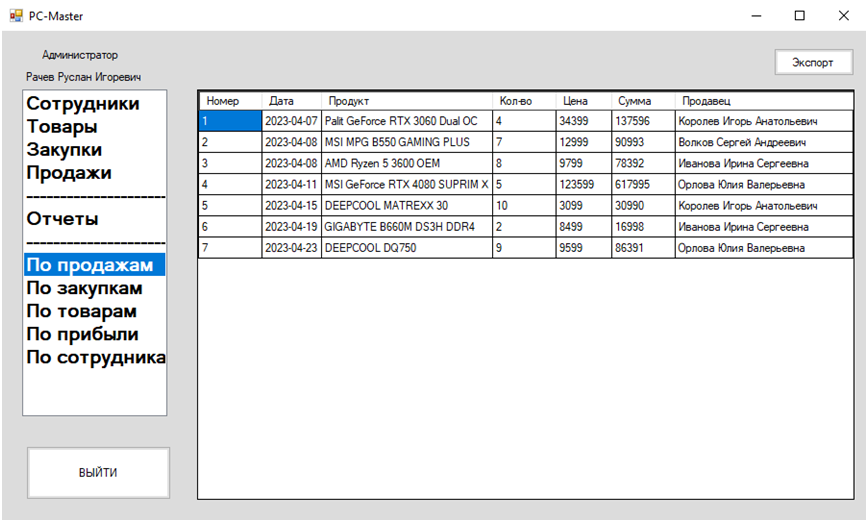


Рисунок 5 – Интерфейс по вкладке «Отчет»

Все отчеты формируются и отображаются корректно. Все пользователи, имеющие доступ к ним, могут в любое время обратиться к любому из предоставленных отчетов без каких-либо ошибок и багов.

## Выгрузка отчетов в формат Excel

После того как мы выбрали необходимый отчет и он был выведен на экран, кроме просто его просмотра у пользователя есть также возможность выгрузки его в Excel. Для этого необходимо нажать кнопку «Экспорт» в верхнем правом углу. Отчет сразу же откроется в новом документе Excel (рис.7). После внесения необходимых изменений, пользователь может сохранить выгруженный отчет в формате xlsx.

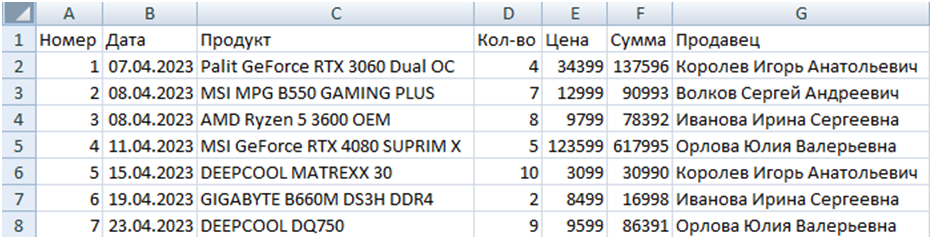


Рисунок 7 – Отчет в формате Excel

Отчеты выгружаются корректно, никаких ошибок не было выявлено. Выгруженный отчет содержит верную информацию и выглядит также как и отчет в программе.

## Реакция системы на ошибочный ввод данных

При попытке входа в систему с неверными или пустыми данными авторизации на экран выводится информация об ошибке (рис.8). Данная функция важна для того, чтобы к системе не получили доступ лица, не являющиеся сотрудниками магазина.

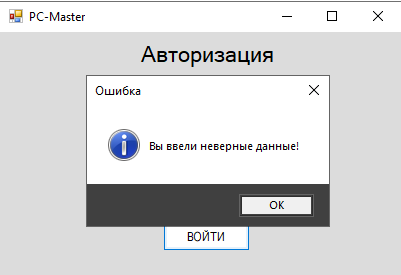


Рисунок 8 – Вывод ошибки при неправильном вводе данных

При добавлении или изменении строк справочников, введенные пользователями данные также проверяются на корректность. Если сотрудник магазина вводит данные, которые не соответствуют своему типу, на экран выводится ошибка об этом (рис.9).

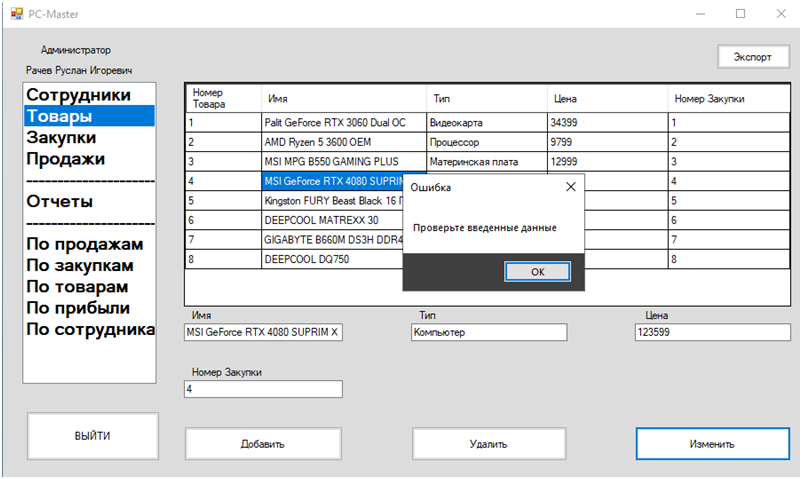


Рисунок 9 – Интерфейс вкладки «Товары» при неправильном вводе данных

Система корректно реагирует на неверно введенные данные. Все подобные ошибки обрабатываются, что позволяет избегать нарушения работы программы.

Нефункциональное тестирование

На рисунке 10 представлен интерфейс сотрудника с открытой вкладкой «Товар». Неудобным можно посчитать только интерфейс добавления и редактирования данных в базе. Он представлен в виде 6 отдельных полей ввода, каждое из которых отвечают за свой столбец таблицы. Для пользователей могло бы быть удобнее редактировать данные непосредственно в самой таблице.

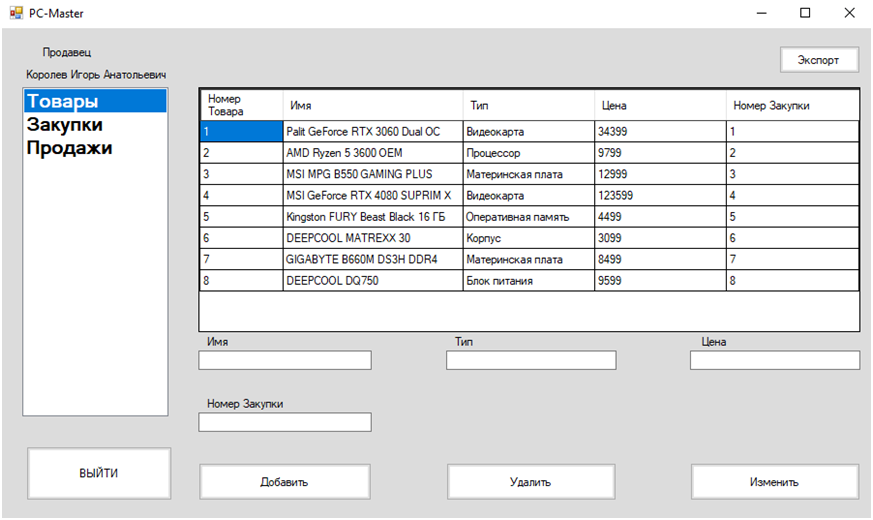


Рисунок 10 – Интерфейс сотрудника вкладка «Товары»

Весь интерфейс системы разработан на русском языке. Нефункциональных ошибок больше не выявлено.

Вывод

В данной работе была спроектирована база данных, интерфейс, разработана диаграмма классов системы. Все эти шаги обеспечили корректную программную реализацию корпоративной информационной системы PC-Master. Также было проведено функциональное и нефункциональное тестирование.

В системе был реализован основной функционал: аутентификация, вид под разными пользователями (в зависимости от роли), редактировании базы данных, формирование отчетов, реакция системы на ошибочный ввод.

Разработанная система полностью отвечает всем требования технического задания. Сроки разработки были соблюдены.

Приложение 1

SQL-код заполнения базы данных:

INSERT INTO `mydb`.`employee` (`employee\_name`, `employee\_login`, `employee\_password`, `employee\_role`) VALUES ('Королев Игорь Анатольевич', 'korolevigor', '12345igorek', 'Продавец');

INSERT INTO `mydb`.`employee` (`employee\_name`, `employee\_login`, `employee\_password`, `employee\_role`) VALUES ('Рачев Руслан Игоревич', 'rouslan', '870125', 'Администратор');

INSERT INTO `mydb`.`employee` (`employee\_id`, `employee\_name`, `employee\_login`, `employee\_password`, `employee\_role`) VALUES ('3', 'Орлова Юлия Валерьевна', 'orlovaa', '412423', 'Продавец');

INSERT INTO `mydb`.`employee` (`employee\_id`, `employee\_name`, `employee\_login`, `employee\_password`, `employee\_role`) VALUES ('4', 'Волков Сергей Андреевич', 'volk473', '9993seriy', 'Продавец');

INSERT INTO `mydb`.`employee` (`employee\_id`, `employee\_name`, `employee\_login`, `employee\_password`, `employee\_role`) VALUES ('5', 'Иванова Ирина Сергеевна', 'ivanovai', 'ivan18325irina', 'Продавец');

INSERT INTO `mydb`.`supplier` (`supplier\_name`, `supplier\_city`) VALUES ('ООО Техника-Дорого', 'Москва');

INSERT INTO `mydb`.`supplier` (`supplier\_name`, `supplier\_city`) VALUES ('Лаборатория Компьютеров', 'Санкт-Петербург');

INSERT INTO `mydb`.`supplier` (`supplier\_name`, `supplier\_city`) VALUES ('Cheap computers', 'Тверь');

INSERT INTO `mydb`.`supplier` (`supplier\_name`, `supplier\_city`) VALUES ('АйТи Технологии', 'Пермь');

INSERT INTO `mydb`.`supplier` (`supplier\_name`, `supplier\_city`) VALUES ('SunDzin Tech', 'Гуанчжоу');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-03-28', '25', '700000', '5');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-05', '30', '210000', '1');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-05', '20', '200000', '1');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-02', '13', '1300000', '3');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-10', '40', '108000', '2');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-10', '20', '38000', '2');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-10', '35', '239750', '2');

INSERT INTO `mydb`.`supply` (`supply\_date`, `supply\_count`, `supply\_price`, `supplier\_id`) VALUES ('2023-04-21', '45', '346500', '4');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('Palit GeForce RTX 3060 Dual OC', 'Видеокарта', '34399', '1');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('AMD Ryzen 5 3600 OEM', 'Процессор', '9799', '2');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('MSI MPG B550 GAMING PLUS', 'Материнская плата', '12999', '3');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('MSI GeForce RTX 4080 SUPRIM X', 'Видеокарта', '123599', '4');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('Kingston FURY Beast Black 16 ГБ', 'Оперативная память', '4499', '5');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('DEEPCOOL MATREXX 30', 'Корпус', '3099', '6');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('GIGABYTE B660M DS3H DDR4', 'Материнская плата', '8499', '7');

INSERT INTO `mydb`.`product` (`product\_name`, `product\_type`, `product\_price`, `supply\_id`) VALUES ('DEEPCOOL DQ750', 'Блок питания', '9599', '8');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-07', '4', '1', '1');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-08', '7', '3', '4');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-08', '8', '2', '5');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-11', '5', '4', '3');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-15', '10', '6', '1');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-19', '2', '7', '5');

INSERT INTO `mydb`.`sale` (`sale\_date`, `sale\_count`, `product\_id`, `employee\_id`) VALUES ('2023-04-23', '9', '8', '3');

Код класса Autorization:

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PC\_Master

{

public partial class Authorization : Form

{

String name = "";

String role = "";

public Authorization()

{

InitializeComponent();

}

private void loginButton\_Click(object sender, EventArgs e)

{

if (loginInput.Text != "" && passwordInput.Text != "")

{

bool userExists = checkPassword();

if (userExists)

{

loginInput.Text = "";

passwordInput.Text = "";

this.Hide();

Main mainForm = new Main(name, role, this);

mainForm.Show();

}

else

{

MessageBox.Show("Вы ввели неверные данные!",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

else

{

MessageBox.Show("Введите логин и пароль!",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly);

}

}

private bool checkPassword()

{

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

string sql = "Select \* from employee WHERE employee\_login='"+loginInput.Text+"' AND employee\_password='"+passwordInput.Text+"'";

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

name = Convert.ToString(reader.GetValue(1));

role = Convert.ToString(reader.GetValue(4));

}

}

}

Console.WriteLine(name);

conn.Close();

conn.Dispose();

if (name == "") return false;

else return true;

}

}

}

Код класса Main:

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Common;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace PC\_Master

{

public partial class Main : Form

{

List<string> columns = new List<string>();

Form authForm;

int itemId = 0;

int rows = 0;

public Main(String name, String role, Form Auth)

{

InitializeComponent();

label1.Text = role;

label2.Text = name;

authForm = Auth;

int tableCount = tableChoice.Items.Count;

if (role != "Администратор")

{

for (int i = 4; i < tableCount; i++)

{

tableChoice.Items.Remove(tableChoice.Items[tableChoice.Items.Count-1]);

}

tableChoice.Items.Remove(tableChoice.Items[0]);

}

}

private void tableChoice\_SelectedIndexChanged(object sender, EventArgs e)

{

if (tableChoice.SelectedIndex < 4)

{

dataGridView1.Height = 253;

queryTable(tableChoice.SelectedItem.ToString());

}

if (tableChoice.SelectedIndex > 6)

{

hideControl();

dataGridView1.Height = 427;

queryReport(tableChoice.SelectedItem.ToString());

}

}

private void queryTable(String table\_name)

{

itemId = 0;

dataGridView1.Columns.Clear();

dataGridView1.Rows.Clear();

columns.Clear();

clearTextBoxes();

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

table\_name = translateTable(table\_name);

string sql = "Select \* from " + table\_name;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

columns = new List<string>();

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

int count = 0;

DataTable schema = null;

schema = reader.GetSchemaTable();

foreach (DataRow col in schema.Rows)

{

string сname = translateColumn(col.Field<String>("ColumnName"));

dataGridView1.Columns.Add(col.Field<String>("ColumnName").ToUpper(), сname);

columns.Add(col.Field<String>("ColumnName"));

count++;

}

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.None;

for (int i = 0; i < count; i++)

dataGridView1.Columns[i].Width = (int)(dataGridView1.Width \* 1 / count);

dataGridView1.Columns[0].Width -= 50;

dataGridView1.Columns[1].Width += 50;

dataGridView1.Columns[count - 1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

int row = 0;

while (reader.Read())

{

dataGridView1.Rows.Add();

for (int column = 0; column < count; column++)

{

string value = "";

if (reader.GetValue(column).GetType() == typeof(DateTime))

{

DateTime time = (DateTime) reader.GetValue(column);

value = time.ToString("yyyy-MM-dd");

}

else value = Convert.ToString(reader.GetValue(column));

dataGridView1.Rows[row].Cells[column].Value = value;

}

row++;

}

rows = row;

}

}

conn.Close();

conn.Dispose();

showControl();

}

private void queryReport(String report\_name)

{

dataGridView1.Columns.Clear();

dataGridView1.Rows.Clear();

columns.Clear();

clearTextBoxes();

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

report\_name = translateReport(report\_name);

string sql = "Select \* from " + report\_name;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

columns = new List<string>();

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

int count = 0;

DataTable schema = null;

schema = reader.GetSchemaTable();

foreach (DataRow col in schema.Rows)

{

string сname = col.Field<String>("ColumnName");

dataGridView1.Columns.Add(col.Field<String>("ColumnName").ToUpper(), сname);

count++;

}

for (int i = 0; i < count; i++)

dataGridView1.Columns[i].Width = (int)(dataGridView1.Width \* 1 / count);

if (report\_name == "sales\_report" || report\_name == "employees\_report")

dataGridView1.Columns[count - 1].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;

int row = 0;

while (reader.Read())

{

dataGridView1.Rows.Add();

for (int column = 0; column < count; column++)

{

string value = "";

if (reader.GetValue(column).GetType() == typeof(DateTime))

{

DateTime time = (DateTime)reader.GetValue(column);

value = time.ToString("yyyy-MM-dd");

}

else value = Convert.ToString(reader.GetValue(column));

dataGridView1.Rows[row].Cells[column].Value = value;

}

row++;

}

rows = row;

}

}

conn.Close();

conn.Dispose();

}

private string translateTable(String name)

{

if (name == "Сотрудники") name = "employee";

if (name == "Товары") name = "product";

if (name == "Продажи") name = "sale";

if (name == "Закупки") name = "supply";

if (name == "Поставщики") name = "supplier";

return name;

}

private string translateColumn(String name)

{

if (name == "employee\_id") name = "Номер Сотрудника";

if (name == "product\_id") name = "Номер Товара";

if (name == "sale\_id") name = "Номер Продажи";

if (name == "supply\_id") name = "Номер Закупки";

if (name == "supplier\_id") name = "Номер Поставщика";

if (name.EndsWith("\_name")) name = "Имя";

if (name.EndsWith("\_login")) name = "Логин";

if (name.EndsWith("\_password")) name = "Пароль";

if (name.EndsWith("\_role")) name = "Роль";

if (name.EndsWith("\_date")) name = "Дата";

if (name.EndsWith("\_price")) name = "Цена";

if (name.EndsWith("\_count")) name = "Количество";

if (name.EndsWith("\_type")) name = "Тип";

return name;

}

private string translateReport(String name)

{

if (name == "По продажам") name = "sales\_report";

if (name == "По закупкам") name = "supplies\_report";

if (name == "По товарам") name = "products\_report";

if (name == "По прибыли") name = "profits\_report";

if (name == "По сотрудникам") name = "employees\_report";

return name;

}

private void showControl()

{

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

Label[] labels = new Label[6] { column1, column2, column3, column4, column5, column6 };

for (int i = 0; i < columns.Count-1; i++)

{

textBoxes[i].Visible = true;

labels[i].Visible = true;

labels[i].Text = translateColumn(columns[i+1]);

}

btnDelete.Visible = true;

btnAdd.Visible = true;

btnChange.Visible = true;

}

private void hideControl()

{

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

Label[] labels = new Label[6] { column1, column2, column3, column4, column5, column6 };

for (int i = 0; i < 6; i++)

{

textBoxes[i].Visible = false;

textBoxes[i].Text = "";

labels[i].Visible = false;

labels[i].Text = "";

}

btnDelete.Visible = false;

btnAdd.Visible = false;

btnChange.Visible = false;

}

private void clearTextBoxes()

{

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

Label[] labels = new Label[6] { column1, column2, column3, column4, column5, column6 };

for (int i = 0; i < 6; i++)

{

textBoxes[i].Visible = false;

textBoxes[i].Text = "";

labels[i].Text = "";

}

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

Int32.TryParse(dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString(), out itemId);

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

for (int i = 0; i < columns.Count-1; i++)

textBoxes[i].Text = dataGridView1.Rows[e.RowIndex].Cells[i+1].Value.ToString();

}

private void btnAdd\_Click(object sender, EventArgs e)

{

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

String table\_name = translateTable(tableChoice.SelectedItem.ToString());

String clmns = "";

for (int i = 1; i < columns.Count; i++)

{

clmns += columns[i] + ", ";

}

clmns = clmns.Substring(0, clmns.Length - 2);

String text = "";

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

for (int i = 0; i < columns.Count - 1; i++)

{

text += "'" + textBoxes[i].Text + "', ";

}

text = text.Substring(0, text.Length - 2);

string sql = "Insert into " + table\_name + " (" + clmns + ") values (" + text + ")";

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка добавлена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

conn.Close();

conn.Dispose();

queryTable(table\_name);

}

private void btnChange\_Click(object sender, EventArgs e)

{

if (itemId != 0)

{

String table\_name = translateTable(tableChoice.SelectedItem.ToString());

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

String text = "";

TextBox[] textBoxes = new TextBox[6] { textBox1, textBox2, textBox3, textBox4, textBox5, textBox6 };

for (int i = 1; i < columns.Count; i++)

{

text += columns[i] + "='" + textBoxes[i - 1].Text + "', ";

}

text = text.Substring(0, text.Length - 2);

string sql = "UPDATE " + table\_name + " SET " + text +

" WHERE " + columns[0] + "=" + itemId;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка изменена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

conn.Close();

queryTable(table\_name);

} else MessageBox.Show("Ни одна строка не выбрана!", "Ошибка");

}

private void btnDelete\_Click(object sender, EventArgs e)

{

if (itemId != 0)

{

String table\_name = translateTable(tableChoice.SelectedItem.ToString());

MySqlConnection conn = DBConnection.GetDBConnection();

conn.Open();

string sql = "DELETE FROM " + table\_name + " WHERE " + columns[0] + "=" + itemId;

MySqlCommand cmd = new MySqlCommand();

cmd.Connection = conn;

cmd.CommandText = sql;

try

{

cmd.ExecuteNonQuery();

MessageBox.Show("Строка удалена!", "Message");

}

catch (Exception)

{

MessageBox.Show("Проверьте введенные данные", "Ошибка");

}

conn.Close();

queryTable(table\_name);

} else MessageBox.Show("Ни одна строка не выбрана!", "Ошибка");

}

private void buttonExport\_Click(object sender, EventArgs e)

{

Excel.Application ExcelApp = new Excel.Application();

Excel.Workbook ExcelWorkBook;

Excel.Worksheet ExcelWorkSheet;

ExcelWorkBook = ExcelApp.Workbooks.Add(System.Reflection.Missing.Value);

ExcelWorkSheet = (Excel.Worksheet)ExcelWorkBook.Worksheets.get\_Item(1);

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

for (int j = 0; j < dataGridView1.ColumnCount; j++)

{

ExcelApp.Cells[1, j + 1] = dataGridView1.Columns[j].HeaderText.ToString();

ExcelApp.Cells[i + 1, j + 1] = dataGridView1.Rows[i].Cells[j].Value;

}

}

ExcelApp.Visible = true;

ExcelApp.UserControl = true;

ExcelApp.Rows.AutoFit();

ExcelApp.Columns.AutoFit();

}

private void buttonExit\_Click(object sender, EventArgs e)

{

this.Close();

authForm.Show();

}

}

}