Оглавление

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 2](#_Toc132766374)

[1. Общие сведения 2](#_Toc132766375)

[2. Назначение и цели конфигурирования системы 2](#_Toc132766376)

[3. Требования к системе 3](#_Toc132766377)

[4. Инструменты разработки 6](#_Toc132766378)

[5. Состав и содержание работ по созданию системы 6](#_Toc132766379)

[6. Согласование и утверждение технического задания. 7](#_Toc132766380)

[7. Порядок контроля и приемки системы 8](#_Toc132766381)

[8. Требования к документированию 12](#_Toc132766382)

[9. Источники разработки 13](#_Toc132766383)

[БАЗА ДАННЫХ 14](#_Toc132766384)

[1. Предметная область 15](#_Toc132766385)

[2. Логическая модель базы данных 16](#_Toc132766386)

[3. Физическая модель базы данных 17](#_Toc132766387)

[4. Описание основных сущностей и типы полей 18](#_Toc132766388)

[5. Содержимое таблиц базы данных 19](#_Toc132766389)

[6. Выводы 22](#_Toc132766390)

[ИНТЕРФЕЙС 23](#_Toc132766391)

[1. Постановка задачи 24](#_Toc132766392)

[2. Проектирование интерфейса 25](#_Toc132766393)

[3. Выводы 27](#_Toc132766394)

[ДИАГРАММА КЛАССОВ 28](#_Toc132766395)

[1. Проектирование диаграммы классов 29](#_Toc132766396)

[2. Вывод 32](#_Toc132766397)

[РЕАЛИЗАЦИЯ ПРОГРАММЫ 33](#_Toc132766398)

[1. Разработка программы 34](#_Toc132766399)

[2. Выводы 40](#_Toc132766400)

[ТЕСТИРОВАНИЕ 41](#_Toc132766401)

[1. Обозначение объекта тестирования 42](#_Toc132766402)

[2. Функциональное тестирование 43](#_Toc132766403)

[3. Выводы 48](#_Toc132766404)

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1. **Общие сведения**
   1. **Наименование информационной системы**

Информационно-справочная система магазина «Beauty house»

* 1. **Основания для проведения работ**

Работа выполняется на основании договора №555 от 02.02.2023г.

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

ПНИПУ

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

Телефон/Факс: +79024725286;

* + 1. **Разработчик**

ИП «Клименко А.А.»;

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

Телефон/Факс: +79027963299

* 1. **Порядок оформления и предъявления заказчику результатов работ:**

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

1. **Назначение и цели конфигурирования системы**
   1. **Назначение системы**

Система предназначена для автоматизации работы магазина «Beauty house».

* 1. **Цели создания системы**

Информационно-справочная система Beauty house создается с целью:

1. Удобное редактирование и добавление информации о товарах и заказах;
2. Демонстрации данных, обеспечивающее практичность в работе с информацией;
3. Автоматизированная отчетность товаров и заказов.
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. Функции, обеспечивающая добавление, изменение и удаление товаров, заказов, покупателей;
3. Функция, предоставляющая отчёты о товарах, заказах, покупателей;
4. Функция, предоставляющая возможность менять пароли для пользователй;
5. Функция генерации отчетов – не менее пяти;
6. Формирование отчетов в формате Excel;
7. Реакция системы на ошибочный ввод;
8. 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 Bookman Old Style;
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 создания и изменения учетных записей для пользователей (покупателей). Пользователи системы могут просматривать товары и создавать заказы.

* + 1. **Роли пользователей**

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

1. Администратор;
2. Покупатель
   * 1. **Права**

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

* + - 1. **Покупатель**

Покупатель может добавлять/удалять/изменять свои заказы, просматривать товары, менять свой пароль. Просматривать отчеты по товарам.

* + - 1. **Администратор**

Администратор может добавлять/удалять/изменять товары, заказы, учетные записи покупателей. Просматривать все существующие отчеты.

* + 1. **Отчеты**

Отчетные формы для администраторов составляются как аналитический отчет по товарам, отчет по данным пользователей, отчёт о корзинах пользователей, отчет по данным брендов товаров, отчет по данным типов товаров в формате xslx.

* + - 1. **Отчет по товару**

Столбцы – наименование товара, цена, объём тары, тип, бренд.

* + - 1. **Отчет по данным пользователей**

Столбцы – ФИО, Email, номер телефона, пароль, роль.

* + - 1. **Отчет о корзинах пользователей**

Столбцы – номе закала, ФИО пользователя, товар.

* + - 1. **Отчет отчет по данным брендов товаров**

Столбцы – наименование типа товара.

* + - 1. **Отчет по данным типов**

Столбцы – наименование бренда 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 Beauty house:

* 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.

На стадии рабочего проектирования должны быть выполнены следующие этапы:

1. Разработка программы.
2. Разработка программной документации.
3. Испытания программы.

На стадии внедрение должны быть выполнены следующие этапы:

1. Подготовка программы.
2. Передача программы.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **№ этапа** | **Этап** | **Форма отчётности** | **Срок выполнения** |
| 1 | Проектирование, разработка логической и физической модели БД | Отчёт | 10.02.2023- 20.02.2023 |
| 2 | Проектирование интерфейса | Отчёт | 21.02.2023-27.02.2023 |
| 3 | Разработка структуры классов | Отчёт | 28.02.2023-01.03.2023 |
| 4 | Программная реализация интерфейсной и функциональной части | Отчёт | 02.03.2023- 26.03.2023 |
| 5 | Испытание продукта | Отчёт | 27.03.2023-30.03.2023 |

## 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 Pro-complect приведены в таблице 5:

Таблица 3. Виды, комплектность и обозначения документов при создании Информационно-справочная система Pro-complect

|  |  |  |
| --- | --- | --- |
| 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».

БАЗА ДАННЫХ

**Цель**: спроектировать базу данных для интернет магазина уходовой косметики.

**Задачи**:

1. Выделить предметную область проектируемой базы данных;
2. Спроектировать физическую и логическую модель базы данных;
3. Обосновать выбор типов полей;
4. Заполнить базу данных тестовыми данными.
5. **Предметная область**

Для автоматизированной работы магазина Beauty house необходима информационно-справочная система, основанная на современной базе данных, включающая в себя данные о покупателях, товаре и заказах.

В базе данных необходимо хранить разнообразную информацию о товаре и покупателях. Информация о товарах магазина должна быть полной.

Деятельность направлена на работу с товарами и работу с покупателями. Отчетные формы для администраторов составляются как аналитический отчет по товарам, отчет по данным пользователей, отчёт о корзинах пользователей, отчет по данным брендов товаров, отчет по данным типов товаров.

1. **Логическая модель базы данных**

На основе данной предметной области было составлена логическая модель со связями. Логическая модель базы данных представлена на рисунке 2.1.

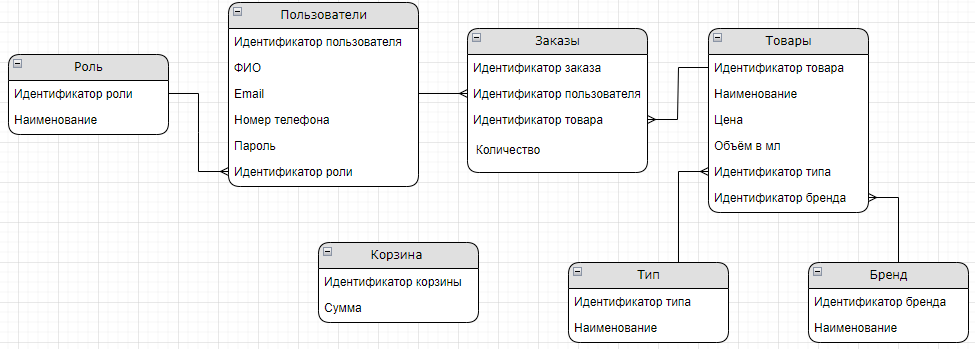


Рис. 2.1. Логическая модель

1. **Физическая модель базы данных**

Физическая модель базы данных была разработана с помощью СУБД MySQL Workbench.

На основе предметной области и логической модели базы данных была составлена физическая модель базы данных. Физическая модель диаграммы базы данных, рисунок 2.2.

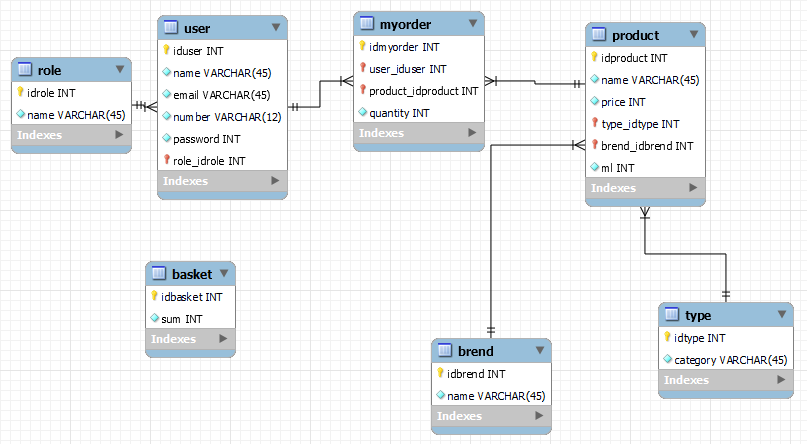


Рис. 2.2. Физическая модель

1. **Описание основных сущностей и типы полей**

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 12, например, для наименования товара, ФИО пользователя 45 символов будет достаточно, а 12 – для номера телефона. Для информации «о количестве, стоимости» задан целочисленный тип данных INT, т.к. количество может быть до нескольких тысяч.

В таблице 3, описаны связи между сущностями в данной базе данных.

Таблица 3 Связи между сущностями в базе данных

|  |  |  |
| --- | --- | --- |
| **Таблица №1** | **Таблица №2** | **Связь** |
| роль | пользователи | 1:М, идентифицирующая |
| пользователи | заказы | 1:М, идентифицирующая |
| заказы | товары | 1:М, идентифицирующая |
| тип | товары | 1:М, идентифицирующая |
| тип | товары | 1:М, идентифицирующая |

1. **Содержимое таблиц базы данных**

Код заполнения базы представлен приложении 1.

Заполненная таблица «Пользователи», рисунок 2.3.

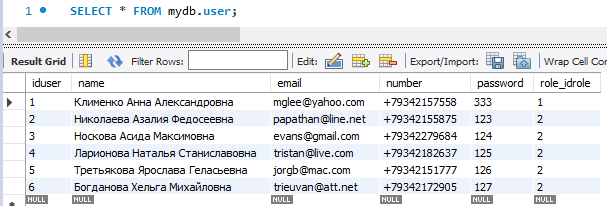


Рис. 2.3. Таблица «Пользователи»

Заполненная таблица «Роли», рисунок 2.4.

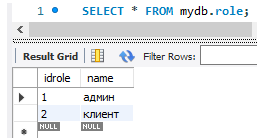


Рис. 2.4. Таблица «Роли»

Заполненная таблица «Товары», рисунок 2.5.

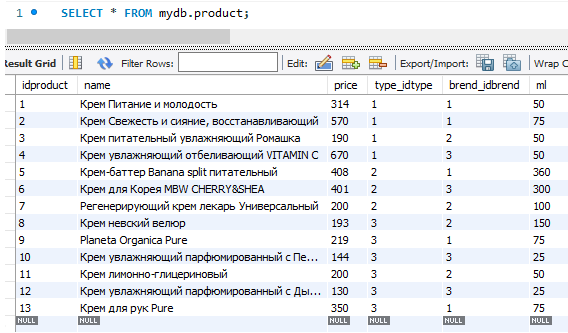
****

Рис. 2.5. Таблица «Товары»

Заполненная таблица «Тип», рисунок 2.6.

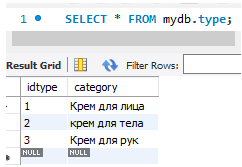
****

Рис. 2.6. Таблица «Тип»

Заполненная таблица «Бренд», рисунок 2.7.

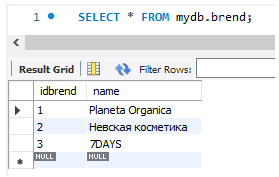


Рис. 2.5. Таблица «Бренд»

Заполненная таблица «Заказы», рисунок 2.8.

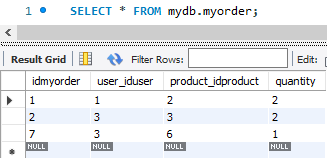


Рис. 2.8. Таблица «Заказы»

Заполненная таблица «Корзина», рисунок 2.9.

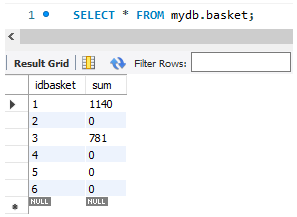


Рис. 2.9. Таблица «Корзина»

1. **Выводы**

В данной лабораторной работе была спроектирована логическая и физическая модель данных, описаны сущности и заполнены таблицы базы данных.

**ИНТЕРФЕЙС**

**Цель работы:** спроектировать интерфейс согласно обозначенному функционалу в техническом задании.

**Задачи:**

1. Разработать прототип входа для администратора и пользователей;
2. Разработать прототипы отображения таблиц всех сущностей;
3. Разработать прототип смены пароля;
4. Разработать прототип реакции на ошибочный ввод данных.
5. **Постановка задачи**

**Назначение системы**

Система предназначена для автоматизации работы магазина «Beauty house».

**Цели создания системы**

Информационно-справочная система Beauty house создается с целью:

1. Удобное редактирование и добавление информации о товарах и заказах;
2. Демонстрации данных, обеспечивающее практичность в работе с информацией;
3. Автоматизированная отчетность товаров и заказов.

**Требования к структуре и функционированию системы**

Автоматизированная система должна обеспечивать возможность выполнения следующих функций:

1. Функция аутентификации;
2. Функции, обеспечивающая добавление, изменение и удаление товаров, заказов, покупателей;
3. Функция, предоставляющая отчёты о товарах, заказах, покупателей;
4. Функция, предоставляющая возможность менять пароли для пользователей;
5. Функция генерации отчетов – не менее пяти;
6. Формирование отчетов в формате Excel;
7. Реакция системы на ошибочный ввод;
8. Интерфейс должен быть доступным и понятным Пользователю.
9. **Проектирование интерфейса**

Ниже представлены окна интерфейса, которые были спроектированы в Figma.

Разделение пользователей рисунок 3.1:



Рис. 2.1. Окно «Разделение пользователей»

Авторизация клиента рисунок 3.2.



Рис.3.2. Окно «Авторизация клиента»

Авторизация администратора рисунок 3.3.

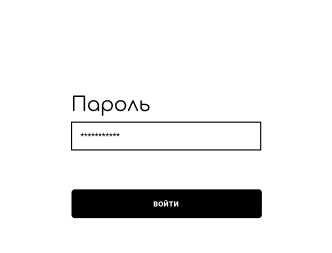


Рис. 3.3. Окно «Авторизация администратора»

Отображение таблиц, тут же кнопка “Отчёт” рисунок 4.

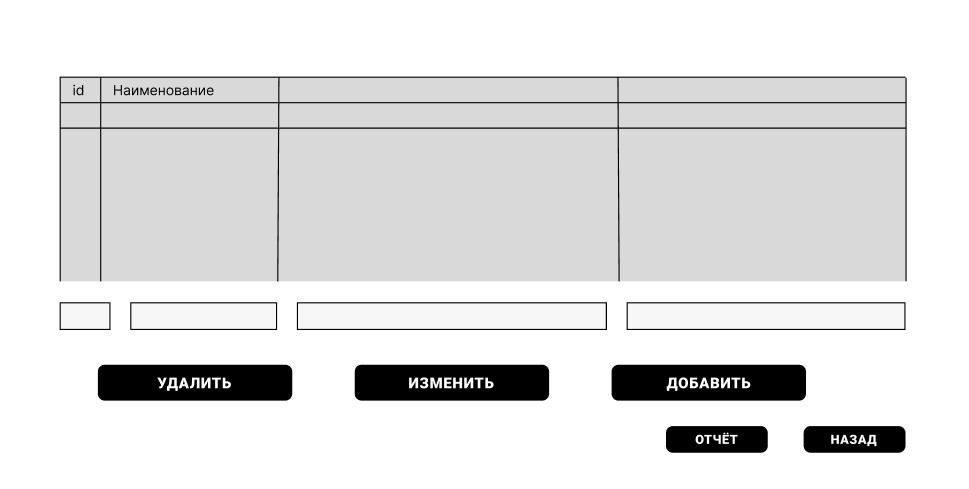


Рис. 3.4. Окно «Отображение таблиц»

Реакция на ошибочный ввод данных рисунок 3.5.

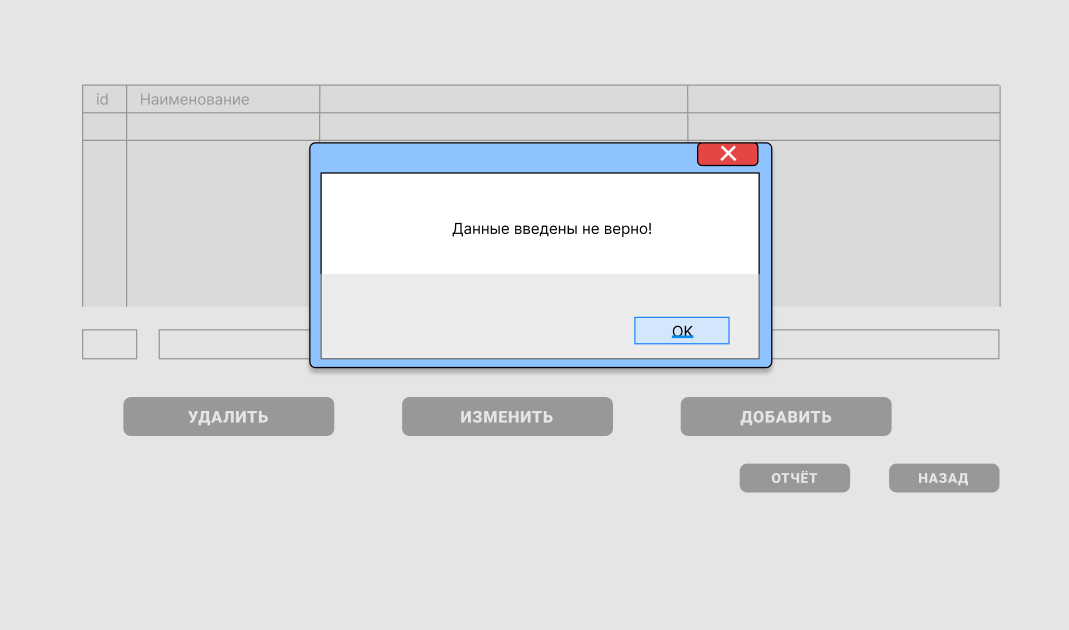


Рис. 3.5. Окно «Реакция на ошибочный ввод данных»

1. **Выводы**

В данной лабораторной работе был спроектирован интерфейс и отображение отчетов согласно техническому заданию.

**ДИАГРАММА КЛАССОВ**

**Цель работы:** разработать структуру классов согласно обозначенному функционалу в техническом задании.

**Задачи:**

1. Определить основной функционал системы;
2. Разработать диаграмму классов для интерфейсной и функциональной части КИС;
3. Описать классы и методы;
4. **Проектирование диаграммы классов**

Ниже представлена диаграмма классов, которая была спроектирована на платформе diagrams.net, рисунок 4.1.

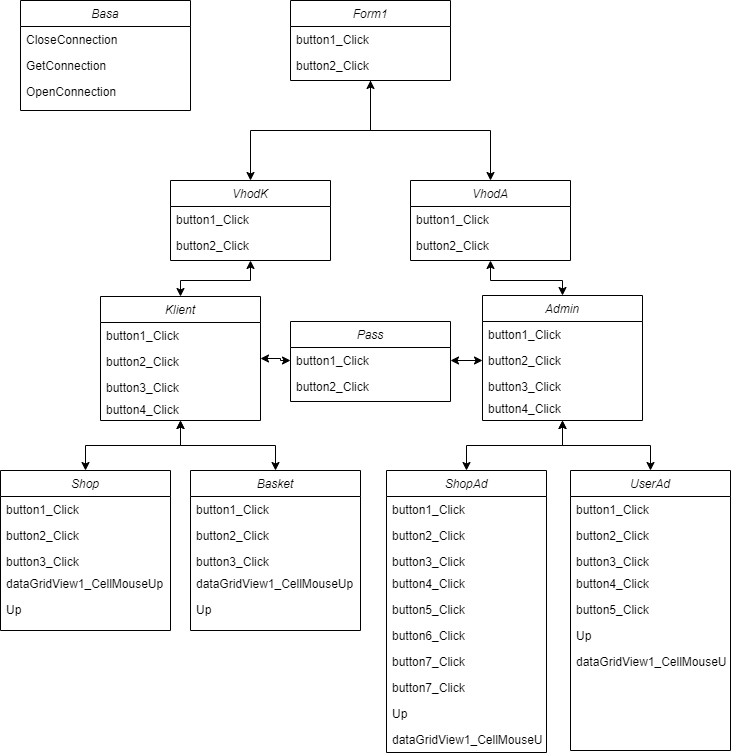


Рис. 4.1. Диаграмма классов

Назначение классов и их методов:

* *Basa –* [*подключение к базе данных MySQL*](https://betacode.net/10513/connect-to-mysql-database-in-csharp)
  + OpenConnection – открытие БД
  + CloseConnection – закытие БД
  + GetConnection – подключение к БД
* *Form1 – форма главного окна*
  + button1\_Click – переход на вход под клиентом
  + button2\_Click – переход на вход под администратором
* *VhodK– авторизация пользователя*
  + button1\_Click – вход, переход под клиента
  + button2\_Click – переход назад
* *VhodA– авторизация* администратора
  + button1\_Click – вход, переход под администратора
  + button2\_Click – переход назад
* *Klient– окно пользователя*
  + button1\_Click – переход на форму с товарами
  + button2\_Click – переход на форму с корзиной
  + button3\_Click – переход на форму для смены пароля
  + button4\_Click – переход назад
* *Admin– окно* администратора
  + button1\_Click – переход на форму с товарами
  + button2\_Click – переход на форму с корзиной
  + button3\_Click – переход на форму для смены пароля
  + button4\_Click – переход назад
* *Pass– смена пароля*
  + button1\_Click – подтверждение смены
  + button2\_Click – переход назад
* *Shop –* форма с товарами
  + button1\_Click – добавление товара в корзину
  + button2\_Click – вывод отчёта
  + button3\_Click – переход назад
  + dataGridView1\_CellMouseUp - записывание данных строки по нажатию на ячейку
  + Up – отрисовка таблицы с нужными данными из базы данных
* *Basket –* форма с корзиной
  + button1\_Click – удаление товара из корзины
  + button2\_Click – вывод отчёта
  + button3\_Click – переход назад
  + dataGridView1\_CellMouseUp - записывание данных строки по нажатию на ячейку
  + Up – отрисовка таблицы с нужными данными из базы данных
* *ShopAd – форма с товарами для администратора*
  + button1\_Click – удаление товара
  + button2\_Click – изменение товара
  + button3\_Click – добавление товара
  + button4\_Click – переход на форму типов
  + button5\_Click – переход на форму брендов
  + button6\_Click – вывод отчёта
  + button7\_Click – переход назад
  + dataGridView1\_CellMouseUp - записывание данных строки по нажатию на ячейку
  + Up – отрисовка таблицы с нужными данными из базы данных
* *UserAd– форма с пользователями для администратора*
  + button1\_Click – удаление строки
  + button2\_Click – изменение строки
  + button3\_Click – добавление строки
  + button4\_Click – вывод отчёта
  + button5\_Click – переход назад
  + dataGridView1\_CellMouseUp - записывание данных строки по нажатию на ячейку
  + Up – отрисовка таблицы с нужными данными из базы данных

1. **Вывод**

В данной лабораторной работы была разработана диаграмма классов и расписаны назначение классов и методов для корпоративной информационной системы Beauty house.

**РЕАЛИЗАЦИЯ ПРОГРАММЫ**

**Цель работы:** запрограммировать обозначенный в ТЗ функционал с учётом созданной БД и разработанных структур классов.

**Задачи:**

1. Запрограммировать обозначенный в ТЗ функционал с учётом созданной БД, спроектированного интерфейса и разработанных структур классов;
2. Привести программный код, согласно функциям.
3. **Разработка программы**

Программа была написана в Visual Studio на C#. Ниже представлена работа основного функционала. Программный код для каждой функции представлен в приложении 2.

Авторизация для клиента и для администратора, для двух ролей созданы соответственно два разных окна.

Авторизация клиента рисунок 5.1.

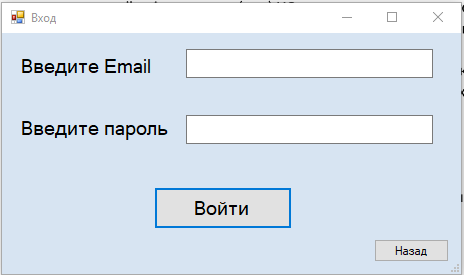


Рис. 5.1. Авторизация клиента

Авторизация администратора рисунок 5.2.

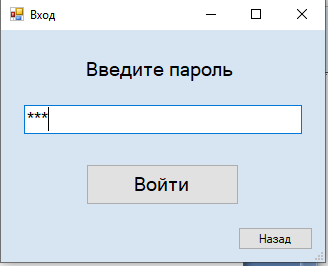


Рис. 5.2. Авторизация администратора

Для клиента доступен переход к трём окнам рисунок 5.3.

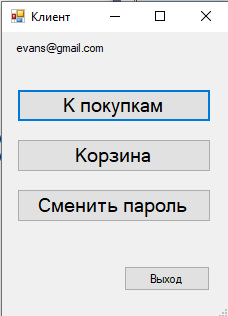


Рис. 5.3. Главное окно для клиента

Отображение товара для клиента, а также функция добавления в корзину на рисунке 5.4.

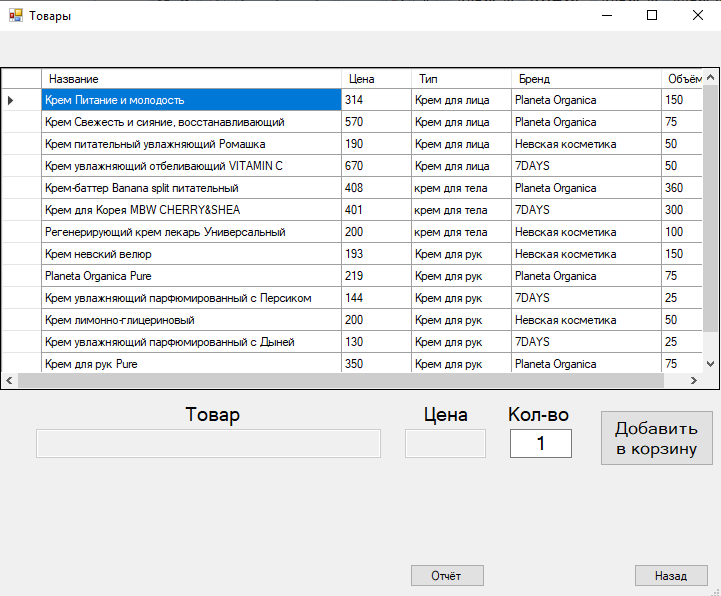


Рис. 5.4. Окно товаров для клиента

Отображение товара для клиента, а также функция удаления из корзины на рисунке 5.5.

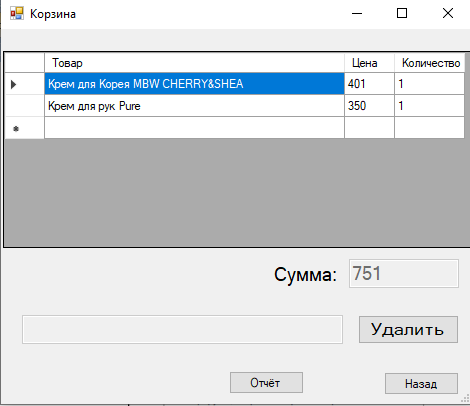


Рис. 5.5. Окно корзины для клиента

Отображение окна смены пароля рисунок 5.6.

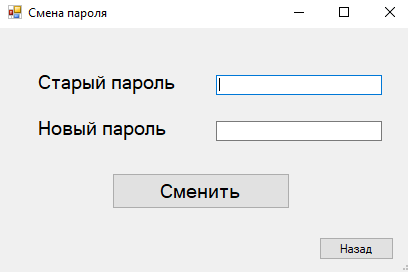


Рис. 5.6. Окно смены пароля

Для администратора доступен переход к трём окнам рисунок 5.7:

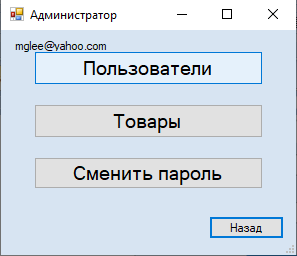


Рис. 5.7. Главное окно администратора

Отображение пользователей для администратора, а также функции изменения, добавления и удаления данных, рисунок 5.8.

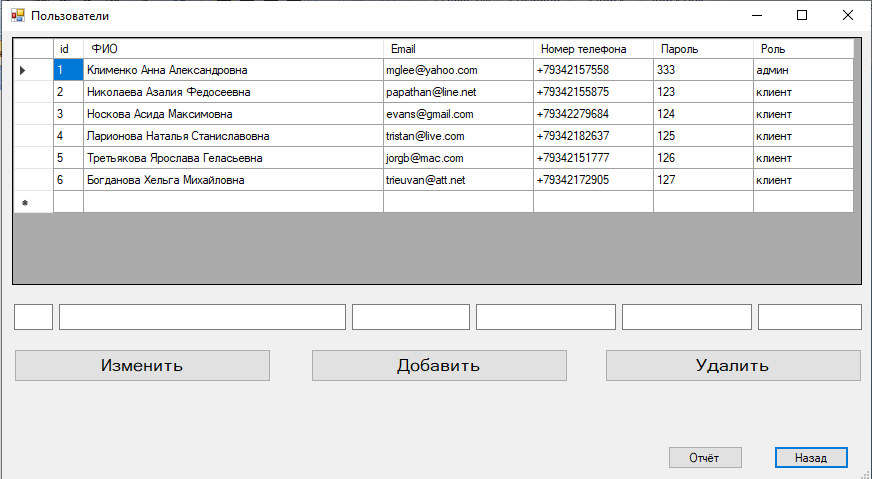


Рис. 5.8. Окно пользователи для администратора

Отображение товаров для администратора, а также функции изменения, добавления и удаления данных, рисунок 5.9.

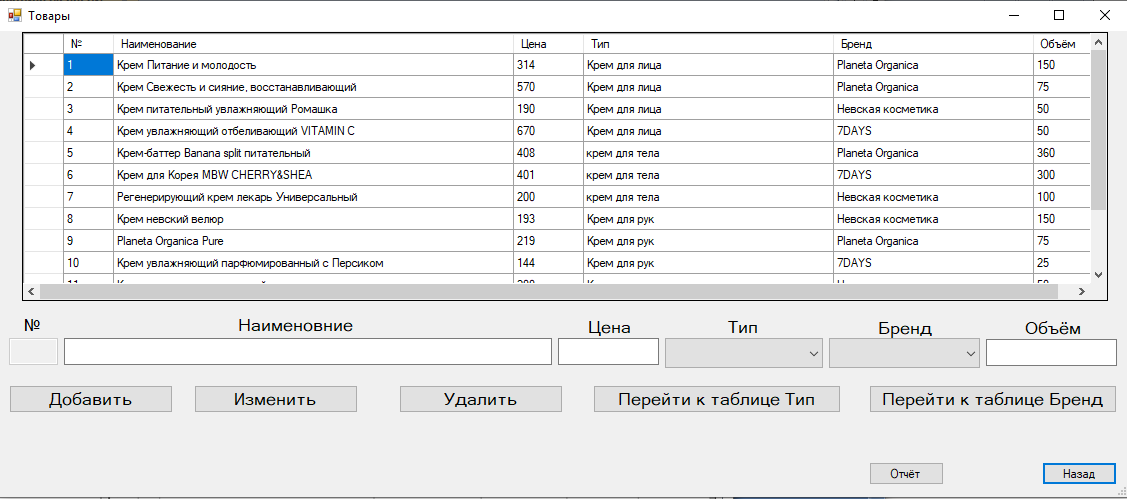


Рис. 5.9. Окно пользователи для администратора

Пример отображение отчётов, в данном случае отчёт по всем пользователям, рисунок 5.10.

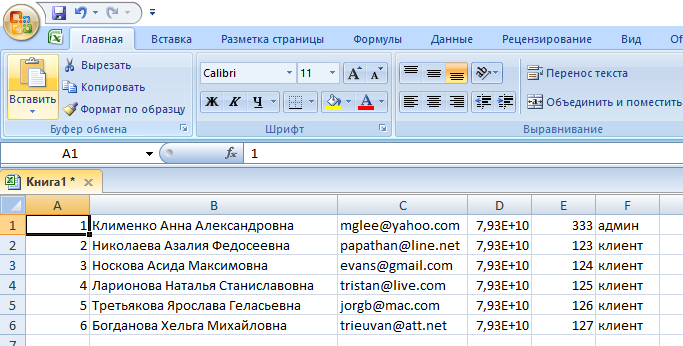


Рис. 5.10. Отчёт по пользователям

Отображение реакции на неправильный ввод данных, рисунки 5.11-5.13.

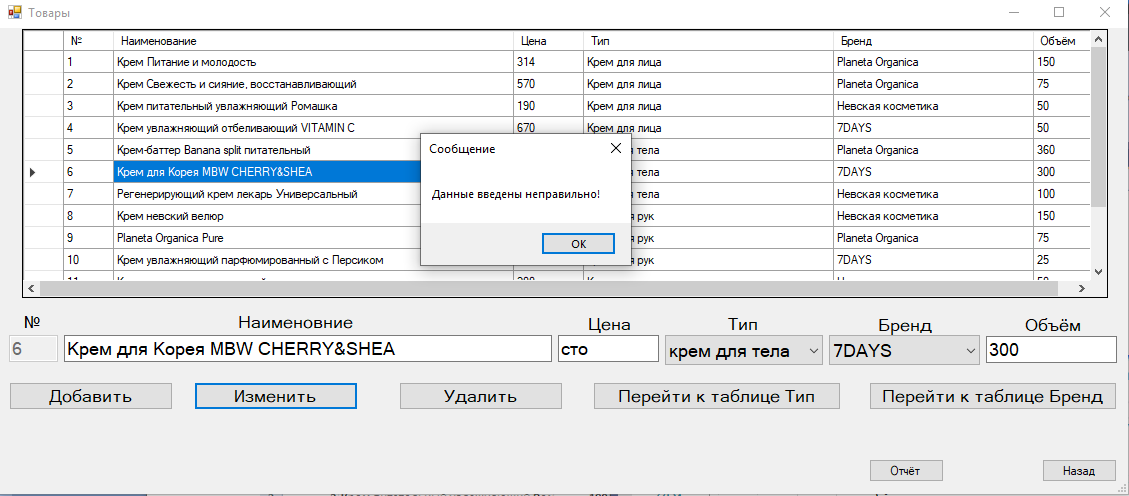


Рис. 5.11. Неправильный ввод при изменении данных о товаре

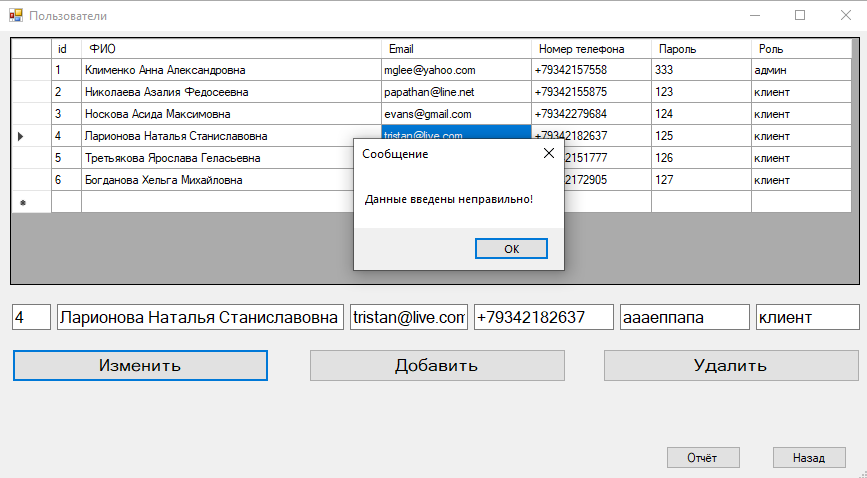


Рис. 5.12. Неправильный ввод при изменении данных о пользователе

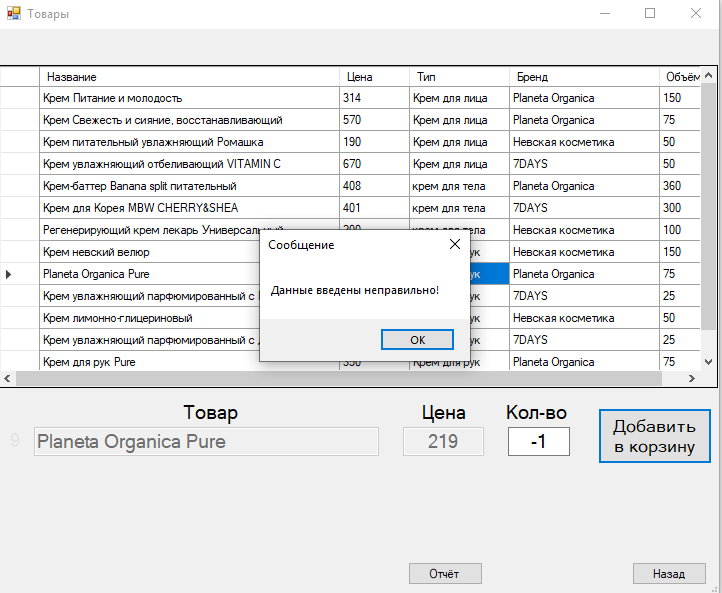


Рис. 5.13. Неправильный ввод при добавлении товара в корзину

1. **Выводы**

В данной лабораторной работе была разработана программа информационной системы Beauty house. Реализован основной функционал: аутентификация, вид под разными пользователями (в зависимости от роли), редактировании базы данных, формирование отчетов, реакция системы на ошибочный ввод.

**ТЕСТИРОВАНИЕ**

**Цель работы:** обозначить назначение и функционал тестирования (объект тестирования)

**Задачи:**

1. Определить объект тестирования
2. Определить вид тестирования
3. Сделать выводы о программном продукте
4. **Обозначение объекта тестирования**

В системе будет проведено тестирование:

1. Функции авторизации;
2. Функции формирования отчетов;
3. Функции изменения, добавления, удаления данных из базы;
4. Реакция системы на ошибочный ввод данных авторизации и добавления медикамента;

В рамках обозначенных объектов тестирования будет проведено функциональное тестирование.

1. **Функциональное тестирование**

Функция авторизации клиента введём логин: jorgb@mac.com и пароль: 126, рисунок 6.1.

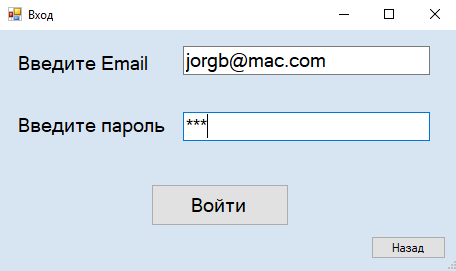


Рис. 6.1. Вход клиента

Вход в систему осуществлён под клиентом, логин видно в левом верхнем углу, рисунок 6.2.

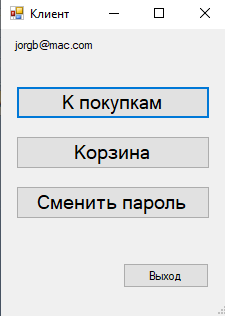


Рис. 6.2. Окно клиента

Проверка на не правильный ввод логина, введём jorgbbb@mac.com, пароль: 126, рисунок 6.3.

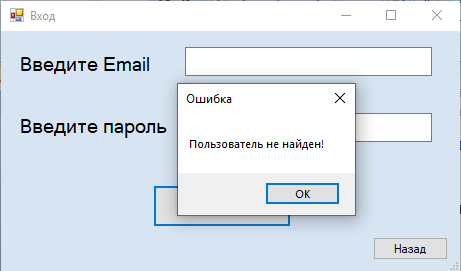


Рис. 6.3. Вывод ошибки о неправильном логине

Вход не произошёл, система вывела сообщение.

Проверка при правильном вводе логина, но несоответствующим паролем, введём jorgb@mac.com, пароль 1266, рисунок 6.4.

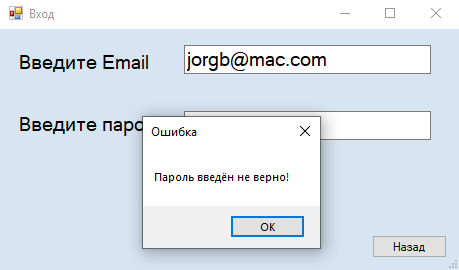


Рис. 6.4. Реакция на неправильный пароль

Для клиента доступен просмотр товара и добавление в корзину. Добавим позицию с количеством: 2. При нажатии на ячейку, выбирается вся стока и отображается в строках ниже, рисунок 6.5.

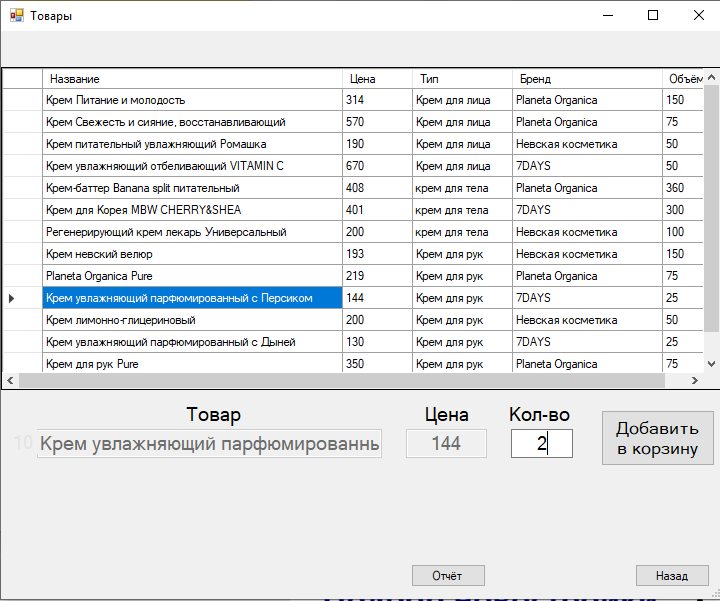


Рис. 6.5. Окно товары для клиента

Просмотрим, что товар добавился в корзину и удалим его, рисунок 6.6. Перед удалением выходит окно подтверждения.

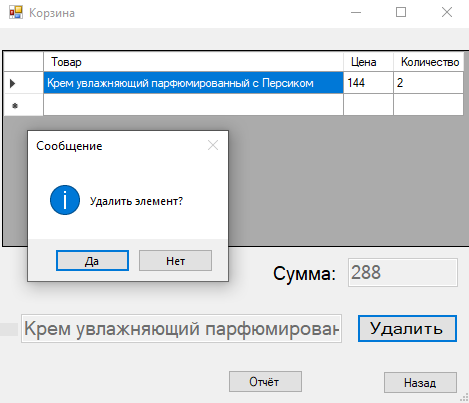


Рис. 6.6. Удаление позиции из корзины

Товар удалён в базе, в окне также пропала строка, рисунок 6.7.

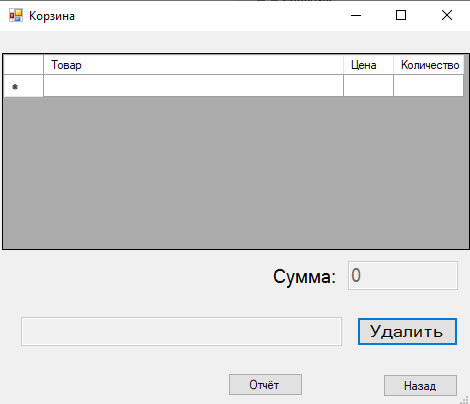


Рис. 6.7. Пустая корзина

Для администратора доступен просмотр пользователей, данные можно изменят, добавлять и удалять. Добавим пользователя рисунок 6.8.

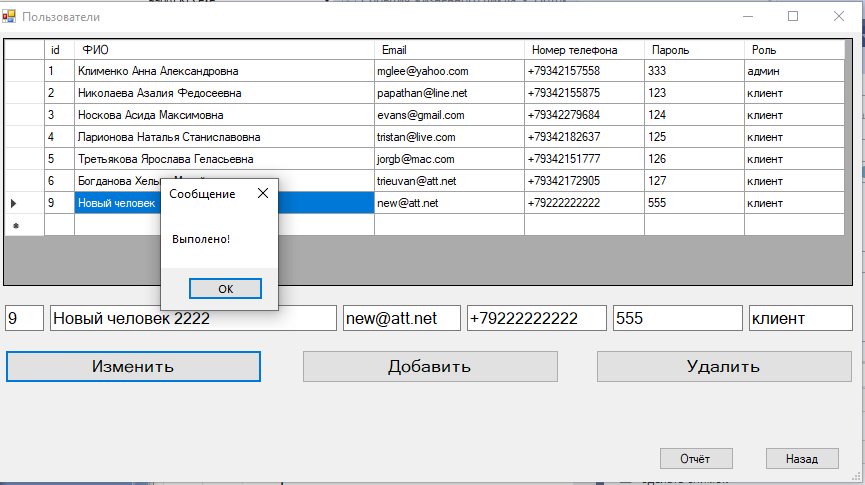


Рис. 6.8. Добавление пользователя

Пользователь добавился базу и отображается в окне, удалим его, рисунок 6.9. Перед удалением выходит окно подтверждения.

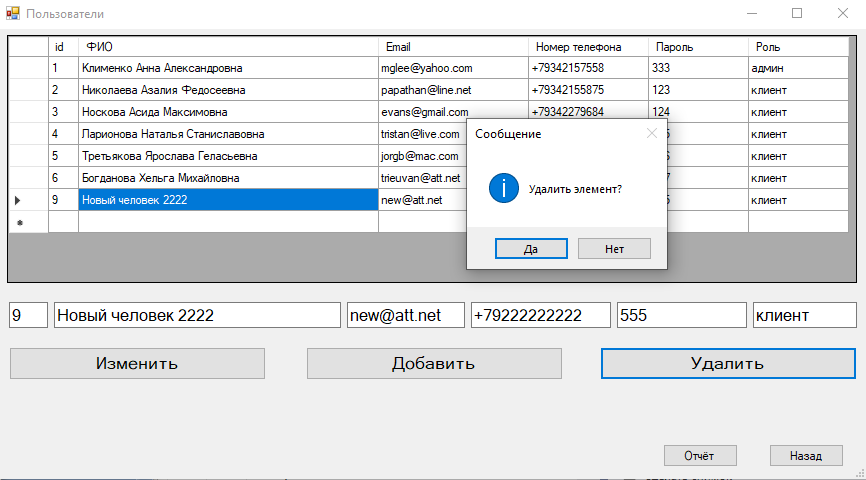


Рис. 6.9. Удаление пользователя

Пользователь удалён в базе, в окне также пропала строка, рисунок 6.10.

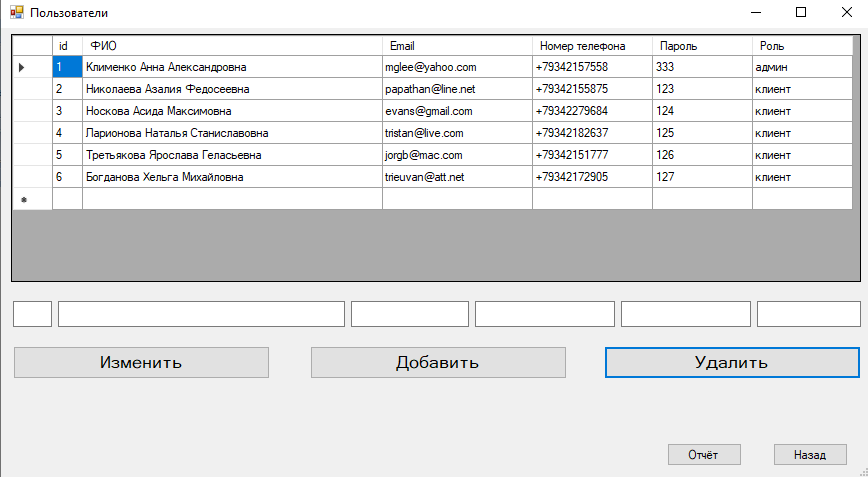


Рис. 6.10. Окно администратора для отображения данных пользователей

Аналогичная работа с формой Товары, только данные Тип и Бренд нельзя записать вручную, они выбираются из списка, рисунок 6.11.

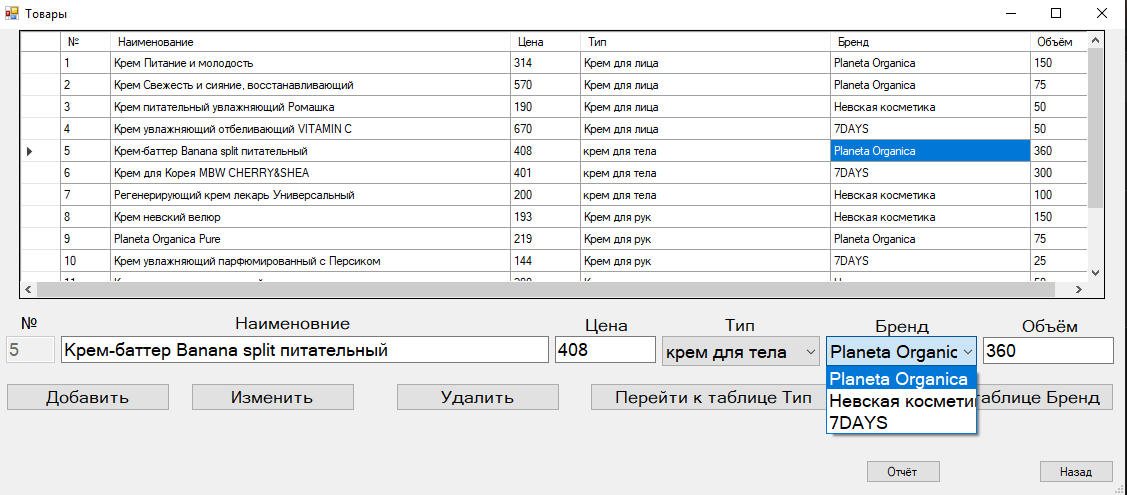


Рис. 6.11. Окно товары для клиента

1. **Выводы**

В данной лабораторной работе проведено функциональное тестирование: аутентификация, вид под разными пользователями (в зависимости от роли), редактировании базы данных, реакция системы на ошибочный ввод.

Приложение 1

INSERT into role(name)

values

('админ'),

('клиент');

select \*from role;

INSERT into user(name, email, number, password, role\_idrole)

values

('Клименко Анна Александровна', 'mglee@yahoo.com', '+79342157558', 333,1),

('Николаева Азалия Федосеевна', 'papathan@line.net', '+79342155875', 123, 2),

('Носкова Асида Максимовна', 'evans@gmail.com', '+79342279684', 124, 2),

('Ларионова Наталья Станиславовна', 'tristan@live.com', '+79342182637', 125, 2),

('Третьякова Ярослава Геласьевна', 'jorgb@mac.com', '+79342151777', 126, 2),

('Богданова Хельга Михайловна', 'trieuvan@att.net', '+79342172905', 127, 2);

select \*from user;

INSERT into type(category)

values

('крем для лица'),

('крем для тела'),

('крем для рук');

select \*from type;

INSERT into brend(name)

values

('Planeta Organica'),

('Невская косметика'),

('7DAYS');

select \*from brend;

INSERT into product(name,price,type\_idtype,brend\_idbrend, ml)

values

('Крем Питание и молодость', 314, 1, 1, 50),

('Крем Свежесть и сияние, восстанавливающий', 570, 1, 1, 75),

('Крем питательный увлажняющий Ромашка', 190, 1, 2, 50),

('Крем увлажняющий отбеливающий VITAMIN С', 670, 1, 3, 50),

('Крем-баттер Banana split питательный', 408, 2, 1, 360),

('Крем для Корея MBW CHERRY&SHEA', 401, 2, 3, 300),

('Регенерирующий крем лекарь Универсальный', 200, 2, 2, 100),

('Крем невский велюр', 193, 3, 2, 50),

('Planeta Organica Pure', 219, 3, 1, 75),

('Крем увлажняющий парфюмированный с Персиком', 144, 3, 3, 25),

('Крем лимонно-глицериновый', 200, 3, 2, 50),

('Крем увлажняющий парфюмированный с Дыней', 130, 3, 3, 25),

('Крем для рук Pure', 350, 3, 1, 75);

select \*from product;

INSERT into basket(sum)

values

(0),

(0),

(0),

(0),

(0),

(0);

select \*from basket;

---добавление суммы заказа в корзину

DELIMITER //

CREATE TRIGGER Myorder\_UPDATE

AFTER INSERT ON myorder

FOR EACH ROW

BEGIN

DECLARE id INT;

DECLARE cc INT;

DECLARE ss INT;

DECLARE k INT;

SELECT new.user\_iduser INTO id ;

SELECT new.quantity INTO k ;

SELECT product.price INTO cc FROM product WHERE new.product\_idproduct = product.idproduct;

SELECT basket.sum INTO ss FROM basket,user WHERE new.user\_iduser = basket.idbasket AND basket.idbasket=user.iduser;

UPDATE basket SET sum=ss+cc\*k WHERE idbasket = id;

END//

INSERT into myorder(user\_iduser,product\_idproduct,quantity)

values

(1,2,2);

select \*from myorder;

select \*from basket;

select \*from product;

---удаление суммы заказа из корзины

DELIMITER //

CREATE TRIGGER Myorder\_DELETE

AFTER DELETE ON myorder

FOR EACH ROW

BEGIN

DECLARE id INT;

DECLARE cc INT;

DECLARE ss INT;

DECLARE k INT;

SELECT OLD.user\_iduser INTO id ;

SELECT OLD.quantity INTO k ;

SELECT product.price INTO cc FROM product WHERE OLD.product\_idproduct = product.idproduct;

SELECT basket.sum INTO ss FROM basket,user WHERE OLD.user\_iduser = basket.idbasket AND basket.idbasket=user.iduser;

UPDATE basket SET sum=ss-cc\*k WHERE idbasket = id;

END//

--удаление корзины при удалении клиента

DELIMITER //

CREATE TRIGGER User\_DELETE

AFTER DELETE ON user

FOR EACH ROW

BEGIN

DECLARE id INT;

SELECT OLD.iduser INTO id ;

DELETE FROM basket WHERE basket.idbasket = id;

END//

DROP TRIGGER User\_INSERT;

---создание козины пи добавлении клиента

DELIMITER //

CREATE TRIGGER User\_INSERT

AFTER INSERT ON user

FOR EACH ROW

BEGIN

INSERT into basket(sum) values (0);

END//

select \*from user;

DELIMITER //

CREATE PROCEDURE insert\_user(f VARCHAR(45), e VARCHAR(45), n VARCHAR(45), p INT, r VARCHAR(45))

BEGIN

DECLARE rl int;

SELECT role.idrole into rl FROM role WHERE r = namee;

INSERT INTO user(name,email,number,password,role\_idrole)

VALUES

(f, e, n, p, rl);

END//

DELIMITER //

CREATE PROCEDURE insert\_product(n VARCHAR(45), p int, t VARCHAR(45), b VARCHAR(45), m INT)

BEGIN

DECLARE ti int;

DECLARE bi int;

SELECT idtype into ti FROM type WHERE t = category;

SELECT idbrend into bi FROM brend WHERE b = name;

INSERT INTO product(name, price, type\_idtype, brend\_idbrend, ml)

VALUES

(n, p, ti, bi, m);

END//

DELIMITER //

CREATE PROCEDURE SELECT\_product(i int)

BEGIN

SELECT name AS 'Наименование' FROM product,myorder WHERE i = myorder.user\_iduser AND myorder.product\_idproduct=product.idproduct;

END

DELIMITER //

CREATE PROCEDURE SELECT\_tov\_brend(i int)

BEGIN

SELECT name AS 'Название', price AS 'Цена', category AS 'Тип', ml AS 'Объём' , name AS 'Бренд' FROM product, type, brend WHERE i = product.brend\_idbrend AND type.idtype=product.type\_idtype AND product.brend\_idbrend=brend.idbrend;

END

----вывод клиентов

CREATE VIEW view1

AS SELECT name AS 'Клиент', email AS 'Email', number AS 'Номер', password AS 'Пароль' FROM user where iduser>1;

select \*from view1

DELIMITER //

CREATE PROCEDURE update\_product(id int, n VARCHAR(45), p int, t VARCHAR(45), b VARCHAR(45), m INT)

BEGIN

DECLARE ti int;

DECLARE bi int;

SELECT idtype into ti FROM type WHERE t = category;

SELECT idbrend into bi FROM brend WHERE b = name;

UPDATE product

SET name = n,

price = p,

type\_idtype = ti,

brend\_idbrend = bi,

ml=m

WHERE idproduct = id;

END//

DELIMITER //

CREATE PROCEDURE update\_user(id int, n VARCHAR(45), e VARCHAR(45), nm VARCHAR(45), p int, r VARCHAR(45))

BEGIN

DECLARE ri int;

SELECT idrole into ri FROM role WHERE r = name;

UPDATE user

SET name = n,

email = e,

number = nm,

password = p,

role\_idrole = ri

WHERE iduser = id;

END//

Приложение 2

Программный код для авторизации клиента:

db.OpenConnection();

string sql = "SELECT iduser FROM user WHERE email=" +"'"+textBox1.Text + "'";

command = new MySqlCommand(sql, db.GetConnection());

if (command.ExecuteScalar() == null)

{

this.textBox1.Text = "";

this.textBox2.Text = "";

MessageBox.Show("Пользователь не найден!", "Ошибка");

}

else {

string iduser = command.ExecuteScalar().ToString();

if (iduser == "1") {

this.textBox1.Text = "";

this.textBox2.Text = "";

MessageBox.Show("Пользователь не найден!", "Ошибка");

}

else

{

sql = "SELECT password FROM user WHERE iduser=" + iduser;

command = new MySqlCommand(sql, db.GetConnection());

string pas = command.ExecuteScalar().ToString();

if (textBox2.Text == pas)

{

this.Visible = false;

Klient form2 = new Klient(int.Parse(iduser));

form2.ShowDialog();

}

else

{

this.textBox2.Text = "";

MessageBox.Show("Пароль введён не верно!", "Ошибка");

}

}

}

Программный код для авторизации администратора:

db.OpenConnection();

string sql = "SELECT password FROM user WHERE iduser=1";

command = new MySqlCommand(sql, db.GetConnection());

string pas = command.ExecuteScalar().ToString();

if (textBox1.Text == pas)

{

this.Visible = false;

Admin form2 = new Admin();

form2.ShowDialog();

}

else

{

this.textBox1.Text = "";

MessageBox.Show("Пароль введён не верно!", "Ошибка");

}

Программный код отображения товаров для клиента:

private void Up()

{

dataGridView1.Rows.Clear();

db.OpenConnection();

string sql = "SELECT idproduct, name, price, (SELECT category FROM type WHERE type.idtype = product.type\_idtype), (SELECT name FROM brend WHERE brend.idbrend = product.brend\_idbrend), ml FROM product";

command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

Программный код реализации функции добавления в корзину:

private void button1\_Click(object sender, EventArgs e)

{

if (int.Parse(textBox3.Text) < 1) { MessageBox.Show("Данные введены неправильно!", "Сообщение"); }

else {

try

{

db.OpenConnection();

string sql = "INSERT into myorder(user\_iduser,product\_idproduct,quantity) VALUES (";

sql = sql + id + "," + label4.Text + "," + textBox3.Text + ")";

MySqlCommand command = new MySqlCommand(sql, db.GetConnection());

try

{

command.ExecuteNonQuery();

}

catch

{

MessageBox.Show("Данные введены неправильно!", "Сообщение"); }

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "1";

label4.Text = "";

}

catch { MessageBox.Show("Выберите товар!", "Ошибка"); }

}

}

Программный код отображения корзины для клиента и удаления из корзины:

private void Up()

{

dataGridView1.Rows.Clear();

db.OpenConnection();

string sql = "SELECT idmyorder, user\_iduser, (SELECT name FROM product WHERE myorder.product\_idproduct = product.idproduct), (SELECT price FROM product WHERE myorder.product\_idproduct = product.idproduct), quantity FROM myorder WHERE user\_iduser="+id;

command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[5]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

}

reader.Close();

sql = "SELECT sum FROM basket WHERE idbasket=" + id;

command = new MySqlCommand(sql, db.GetConnection());

textBox1.Text = command.ExecuteScalar().ToString();

db.CloseConnection();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

Программный код удаления товара из корзины:

private void button2\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Удалить элемент?",

"Сообщение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1);

if (result == DialogResult.Yes)

{

try

{

db.OpenConnection();

string sql = "DELETE FROM myorder WHERE idmyorder =" + label2.Text;

command = new MySqlCommand(sql, db.GetConnection());

command.ExecuteNonQuery();

textBox2.Text = "";

label2.Text = "";

db.CloseConnection();

Up();

}

catch { MessageBox.Show("Выберите товар!", "Ошибка"); }

}

}

Программный код смены пароля:

private void button1\_Click(object sender, EventArgs e)

{

command = new MySqlCommand("SELECT password FROM user WHERE iduser=" + id, db.GetConnection());

string pas = command.ExecuteScalar().ToString();

if (textBox1.Text == pas)

{

string sql = "UPDATE user SET password = " + textBox2.Text + " WHERE iduser = " + id;

command = new MySqlCommand(sql, db.GetConnection());

int number = command.ExecuteNonQuery();

this.textBox1.Text = "";

this.textBox2.Text = "";

MessageBox.Show("Пароль изменён! ("+ number+")");

}

else

{

this.textBox1.Text = "";

MessageBox.Show("Пароль введён не верно!", "Ошибка");

}

}

Программный код отображения пользователей для администратора:

private void Up()

{

dataGridView1.Rows.Clear();

db.OpenConnection();

string sql = "SELECT iduser, name, email, number, password, (SELECT name FROM role WHERE role.idrole = user.role\_idrole) FROM user";

command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

Программный код реализации функции добавления данных для администратора:

private void button3\_Click(object sender, EventArgs e)

{

db.OpenConnection();

string procedure\_name = "insert\_user";

MySqlCommand command = new MySqlCommand(procedure\_name, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

try

{

MySqlParameter f = new MySqlParameter

{

ParameterName = "f",

Value = textBox2.Text

};

MySqlParameter em = new MySqlParameter

{

ParameterName = "e",

Value = textBox3.Text

};

MySqlParameter n = new MySqlParameter

{

ParameterName = "n",

Value = textBox4.Text

};

MySqlParameter p = new MySqlParameter

{

ParameterName = "p",

Value = int.Parse(textBox5.Text)

};

MySqlParameter r = new MySqlParameter

{

ParameterName = "r",

Value = textBox6.Text

};

command.Parameters.Add(f);

command.Parameters.Add(em);

command.Parameters.Add(n);

command.Parameters.Add(p);

command.Parameters.Add(r);

command.ExecuteNonQuery();

MessageBox.Show("Выполено!", "Сообщение");

}

catch

{ MessageBox.Show("Данные введены неправильно!", "Сообщение"); }

db.CloseConnection();

Up();

Clean();

}

Программный код реализации функции изменения данных для администратора:

private void button2\_Click(object sender, EventArgs e)

{

db.OpenConnection();

string procedure\_name = "update\_user";

MySqlCommand command = new MySqlCommand(procedure\_name, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

try

{

MySqlParameter id = new MySqlParameter

{

ParameterName = "id",

Value = int.Parse(textBox1.Text)

};

MySqlParameter n = new MySqlParameter

{

ParameterName = "n",

Value = textBox2.Text

};

MySqlParameter em = new MySqlParameter

{

ParameterName = "e",

Value = textBox3.Text

};

MySqlParameter nm = new MySqlParameter

{

ParameterName = "nm",

Value = textBox4.Text

};

MySqlParameter p = new MySqlParameter

{

ParameterName = "p",

Value = int.Parse(textBox5.Text)

};

MySqlParameter r = new MySqlParameter

{

ParameterName = "r",

Value = textBox6.Text

};

command.Parameters.Add(id);

command.Parameters.Add(n);

command.Parameters.Add(em);

command.Parameters.Add(nm);

command.Parameters.Add(p);

command.Parameters.Add(r);

command.ExecuteNonQuery();

MessageBox.Show("Выполено!", "Сообщение");

}

catch

{ MessageBox.Show("Данные введены неправильно!", "Сообщение"); }

db.CloseConnection();

Up();

Clean();

}

Программный код реализации функции удаления данных для администратора:

private void button4\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Удалить элемент?",

"Сообщение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1);

if (result == DialogResult.Yes)

{

try

{

db.OpenConnection();

string sql = "DELETE FROM user WHERE iduser =" + textBox1.Text;

command = new MySqlCommand(sql, db.GetConnection());

command.ExecuteNonQuery();

db.CloseConnection();

Up();

}

catch { MessageBox.Show("Выберите товар!", "Ошибка"); }

Clean();

}

}

Программный код отображения товаров для администратора:

private void Up()

{

dataGridView1.Rows.Clear();

db.OpenConnection();

string sql = "SELECT idproduct, name, price, (SELECT category FROM type WHERE type.idtype = product.type\_idtype), (SELECT name FROM brend WHERE brend.idbrend = product.brend\_idbrend), ml FROM product";

command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

MySqlDataReader reader = command.ExecuteReader();

List<string[]> data = new List<string[]>();

while (reader.Read())

{

data.Add(new string[6]);

data[data.Count - 1][0] = reader[0].ToString();

data[data.Count - 1][1] = reader[1].ToString();

data[data.Count - 1][2] = reader[2].ToString();

data[data.Count - 1][3] = reader[3].ToString();

data[data.Count - 1][4] = reader[4].ToString();

data[data.Count - 1][5] = reader[5].ToString();

}

reader.Close();

sql = "SELECT category FROM type";

command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

reader = command.ExecuteReader();

while (reader.Read())

{

comboBox2.Items.Add(reader[0].ToString());

}

reader.Close();

sql = "SELECT name FROM brend";

command = new MySqlCommand(sql, db.GetConnection());

msda.SelectCommand = command;

msda.Fill(dt);

reader = command.ExecuteReader();

while (reader.Read())

{

comboBox3.Items.Add(reader[0].ToString());

}

reader.Close();

db.CloseConnection();

foreach (string[] s in data)

dataGridView1.Rows.Add(s);

}

Программный код реализации функции добавления данных о товаре для администратора:

private void button6\_Click(object sender, EventArgs e)

{

db.OpenConnection();

string procedure\_name = "insert\_product";

MySqlCommand command = new MySqlCommand(procedure\_name, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

try

{

MySqlParameter n = new MySqlParameter

{

ParameterName = "n",

Value = textBox1.Text

};

MySqlParameter p = new MySqlParameter

{

ParameterName = "p",

Value = int.Parse(textBox2.Text)

};

MySqlParameter t = new MySqlParameter

{

ParameterName = "t",

Value = comboBox2.Text

};

MySqlParameter b = new MySqlParameter

{

ParameterName = "b",

Value = comboBox3.Text

};

MySqlParameter m = new MySqlParameter

{

ParameterName = "m",

Value = int.Parse(textBox3.Text)

};

command.Parameters.Add(n);

command.Parameters.Add(p);

command.Parameters.Add(t);

command.Parameters.Add(b);

command.Parameters.Add(m);

command.ExecuteNonQuery();

MessageBox.Show("Выполено!", "Сообщение");

}

catch

{ MessageBox.Show("Данные введены неправильно!", "Сообщение"); }

db.CloseConnection();

Up();

Clean();

}

Программный код реализации функции изменения данных о товаре для администратора:

private void button7\_Click(object sender, EventArgs e)

{

db.OpenConnection();

string procedure\_name = "update\_product";

MySqlCommand command = new MySqlCommand(procedure\_name, db.GetConnection());

command.CommandType = CommandType.StoredProcedure;

try

{

MySqlParameter id = new MySqlParameter

{

ParameterName = "id",

Value = int.Parse(textBox4.Text)

};

MySqlParameter n = new MySqlParameter

{

ParameterName = "n",

Value = textBox1.Text

};

MySqlParameter p = new MySqlParameter

{

ParameterName = "p",

Value = int.Parse(textBox2.Text)

};

MySqlParameter t = new MySqlParameter

{

ParameterName = "t",

Value = comboBox2.Text

};

MySqlParameter b = new MySqlParameter

{

ParameterName = "b",

Value = comboBox3.Text

};

MySqlParameter m = new MySqlParameter

{

ParameterName = "m",

Value = int.Parse(textBox3.Text)

};

command.Parameters.Add(id);

command.Parameters.Add(n);

command.Parameters.Add(p);

command.Parameters.Add(t);

command.Parameters.Add(b);

command.Parameters.Add(m);

command.ExecuteNonQuery();

MessageBox.Show("Выполено!", "Сообщение");

}

catch

{ MessageBox.Show("Данные введены неправильно!", "Сообщение"); }

db.CloseConnection();

Up();

Clean();

}

Программный код реализации функции удаления данных о товаре для администратора:

private void button8\_Click(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Удалить элемент?",

"Сообщение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1);

if (result == DialogResult.Yes)

{

try

{

db.OpenConnection();

string sql = "DELETE FROM product WHERE idproduct =" + textBox4.Text;

command = new MySqlCommand(sql, db.GetConnection());

command.ExecuteNonQuery();

db.CloseConnection();

Up();

}

catch { MessageBox.Show("Выберите товар!", "Ошибка"); }

Clean();}}

Программный код отображения отчётов:

private void button5\_Click(object sender, EventArgs e)

{

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

ExcelApp.Application.Workbooks.Add(Type.Missing);

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

{

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

{

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

}

}

ExcelApp.Visible = true;

ExcelApp.UserControl = true;}