**СОДЕРЖАНИЕ**

[Постановка задачи 4](#_Toc132287305)

[Основные этапы реализации 6](#_Toc132287306)

[1. Средства разработки 7](#_Toc132287307)

[2. Структура базы данных 12](#_Toc132287308)

[2.1 Предметная область 12](#_Toc132287309)

[2.2 Логическая модель 12](#_Toc132287310)

[2.3 Физическая модель 13](#_Toc132287311)

[2.4 Связи между сущностями 17](#_Toc132287312)

[3. Программный продукт 19](#_Toc132287313)

[3.1 Функционал 19](#_Toc132287314)

[Выводы 30](#_Toc132287315)

[Список литературы 31](#_Toc132287316)

[Приложение 1 33](#_Toc132287317)

Постановка задачи

### Корпоративная информационная система, осуществляющая функции редактирования, создания, чтения, удаления и обновления товаров предназначена для рекламы и в последующем заказов товаров определенной категории.

### Цель создания системы:

* реклама товаров;
* прием заказов;
* отслеживание количества заказов;
* обновление товаров.

### **Состав системы**

### В состав системы должны входить следующие компоненты:

### Подсистема управления. Подсистема, предназначенная для работы Администратора системы с данными БД (редактирование, удаление, обновление и добавление), выполнения настроек.

* Подсистема сбора и обработки данных. Подсистема, предназначенная для регистрации заявок.
* Подсистема защиты информации. Подсистема представляет собой программно-технический комплекс, предназначенный для защиты технических средств, программного обеспечения и данных от несанкционированного доступа к данным системы. Выполняет функции по идентификации и аутентификации сторон, функции по разграничению прав доступа к информационным ресурсам.
* База данных системы. Подсистема, предназначена для хранения и управления данными
* Подсистема отчетности. Подсистема, предназначенная для формирования отчета, который содержит данные, генерируемые в процессе функционирования системы, сохранение отчета в форматах DOCX.

(Вывод отзывов определенного пользователя; Вывод количества товаров; Вывод зарегистрированных пользователей; Вывод о количестве администраторов; Вывод категории товаров).

### **Требования к способам и средствам связи для информационного обмена между подсистемами**

### Информационный обмен между подсистемами должен осуществляться

### посредством стандартных протоколов и интерфейсов электронного взаимодействия.

### Система должна обеспечивать предоставление информации, находящейся в ней, при наличии соответствующих прав и привилегий.

### **Требования к режимам функционирования системы**

### Система должна функционировать круглогодично в автоматическом режиме, обеспечивая непрерывный круглосуточный режим работы, за исключением регламентных остановов для проведения процедур технического обслуживания.

### Длительность и периодичность регламентных остановов системы регулируется Заказчиком на основании внутренних регламентов Заказчика.

### **Требования к модернизации системы**

### Система должна предусматривать возможность адаптации без существенных доработок;

### Система должна поддерживать возможность дальнейшего развития информационной основы;

### Система должна поддерживать возможность развития технологической основы (модернизация и обновление серверов и рабочих станций, переход на новые версии операционных систем).

Основные этапы реализации

### На основе сформулированного задания были выделены следующие основные этапы реализации системы и определены сроки начала и окончания выполнения каждого этапа. Табл.1.

Таблица 1 – Сроки реализации

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование этапа** | **Сроки** | | **Форма отчетности** |
| **Начало** | **Окончание** |
| Проектирование хранилища данных КИС | 10.02.2023 | 19.02.2023 | Отчет |
| Проектирование интерфейса КИС | 20.02.2023 | 1.03.2023 | Отчет |
| Разработка структур классов | 2.03.2023 | 11.03.2023 | Отчет |
| Программная реализация КИС | 12.03.2023 | 9.04.2023 | Отчет |
| Тестирование КИС | 10.04.2023 | 14.04.2023 | Отчет |

1. Средства разработки

### В качестве языка запросов для работы с базой данных выбран язык структурированных запросов SQL.[1]

### *Язык структурированных запросов* – был создан фирмой IBM в начале 70-х годов прошлого века для работы с реляционными базами данных. К середине 80-х годов появились чрезвычайно эффективные реляционные системы управления базами данных (СУБД), такие как Db2, Oracle, MS SQL Server, Sybase, Informix. Несколько позже в связи с развитием всемирной паутины (WWW – World Wide Web) широко стала применяться СУБД MySQL. Все перечисленные СУБД – сетевые, многопользовательские. На персональных ЭВМ применяются однопользовательские СУБД Paradox, FoxPro, Access. Во все реляционные СУБД встроен язык SQL. Изначально многие СУБД, например, FoxPro, имели собственный, отличный от SQL язык, но простота, эффективность, универсальность сделала SQL сначала де-факто, а затем в 1986 году де-юре стандартным языком запросов для реляционных СУБД. Благодаря своей (относительной) простоте эксплуатации, простоте написания и очень высокой скорости выполнения запросов реляционная модель данных вместе с языком SQL применяется в настоящее время наиболее широко, практически вытеснив все другие модели (иерархические, сетевые и т.д.).

### В рамках разработки информационной системы необходимо создать базу данных для хранения, создания и редактирования данных, на основе которых, будет осуществляться выполнение запросов пользователей. На данный момент двумя наиболее популярными СУБД являются *MySQL* и *PostgreSQL*.

### Для того чтобы выбрать лучшую СУБД, которая подойдет под задачи приложения, необходимо их сравнить.

### *MySQL* работает под управлением практически любой операционной системы. Начать работу можно с бесплатным сервером, перейдя дальше на платную версию. Данная СУБД является одной из самых популярных в мире, известная, как самая надежная и безопасная система по управлению базами данных.

### Преимущества данной системы:

### Безопасность;

### Простота в использовании;

### Кроссплатформенность;

### Масштабируемость.

### Из недостатков можно выделить:

### Ограниченный функционал в бесплатной версии;

### Низкая скорость разработки.

### *PostgreSQL* – реляционная база данных, работающая под управлением любой операционной системы. Является альтернативой коммерческим базам данных.

### Положительные стороны данной СУБД:

### Восстановление данных на определенный момент времени;

### Большое количество документации;

### Имеет множество дополнительных плагинов.

### Минусы:

### Сложная установка;

### Нестабильность сервера;

### Тяжелый синтаксис.

### Исходя из сравнения СУБД, была выбрана *MySQL*, так как она имеет ряд преимуществ по сравнению с другими: простота в использовании, безопасность, масштабируемость, доступность.

### Для разработки базы данных на *MySQL* необходим интерфейс. Наиболее популярными являются: *Adminer*, *PHPMyAdmin*, *Workbench*.

*Adminer* – бесплатный инструмент, служащий для администрирования и управления базами данных. Является облегченным вариантом *PHPMyAdmin*. Поддерживает *MySQL*, *PostgreSQL*, *SQLite*, *MS SQL*, *Oracle*, *SimpleDB*, *Elasticsearch*, *MongoDB*.

Возможности функционала *Adminer*:

* Абсолютно бесплатная;
* Поддерживает мультиязычность;
* Работа с базами данных, с их структурой и индексами;
* Сохранение сессии при аутентификации;
* Поддержка различных тем оформления.

Минусы *Adminer*:

* Присутствие уязвимостей в защите;
* Неполная поддержка *PostgreSQL*.

*PHPMyAdmin* – бесплатный инструмент, предназначенный для управления базами данных. Поддерживает широкий спектр операций на *MySQL*. Наиболее популярные операции (управления базами данных, таблицами, индексами, полями и т.д.) могут быть выполнены с помощью пользовательского интерфейса.

Достоинства PHPMyAdmin:

* Управление базами данных без непосредственного ввода SQL запроса;
* Частое обновление и поддержка системы;
* Возможность интегрировать в собственные проекты.

Недостатки:

* Для тонкой настройки необходимы глубокие знания.

*Workbench* – визуальный инструмент, объединяющий создание, администрирование, проектирование и обслуживание баз данных в единую среду разработки.

Достоинства данной системы:

* Возможность представления и редактирования моделей баз данных в графическом виде;
* Простое создание связей между полями таблиц, за счет функционального механизма;
* Восстановление структур таблиц и связей при помощи функции Reverse Engineering.

К недостаткам относятся:

* Сложный интерфейс;
* Добавление данных в таблицы возможно исключительно при использовании SQL запросов.

Сравнив популярные интерфейсы, и исходя из потребностей по разработке системы, для организации работы с базой данных был выбран *Workbench*, так как он обладает знакомым разработчикам интерфейсом, является бесплатным и часто обновляется [2-5].

### В качестве языка программирования был выбран C#. Язык программирования С# был разработан в 1993–2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft.

### C# поддерживает много полезных функций:

### инкапсуляция,

### наследование,

### полиморфизм,

### перегрузка операторов,

### статическая типизация.

### К преимуществам данного языка можно отнести:

### Поддержка подавляющего большинства продуктов Microsoft

### Бесплатность ряда инструментов для небольших компаний и некоторых индивидуальных разработчиков – Visual Studio, облако Azure, Windows Server, Parallels Desktop для Mac Pro и др.

### Типы данных имеют фиксированный размер (32-битный int и 64-битный long), что повышает «мобильность» языка и упрощает программирование.

### Автоматическая «сборка мусора»  Это значит, что в большинстве случаев не придётся заботиться об освобождении памяти. Общеязыковая среда CLR сама вызовет сборщик мусора и очистит память.

### Низкий порог вхождения. Синтаксис C# имеет много схожего с другими языками программирования, благодаря чему облегчается переход для программистов. Язык C# часто признают наиболее понятным и подходящим для новичков.

### Но есть у C# и некоторые недостатки:

### Приоритетная ориентированность на платформу Windows;

### Язык бесплатен только для небольших фирм, индивидуальных программистов, стартапов и учащихся. Крупной компании покупка лицензионной версии этого языка обойдётся в круглую сумму [6,7].

1. Структура базы данных

2.1 Предметная область

### Для корпоративной информационной системы необходимо хранилище данных, которое содержит в себе информацию о товарах, которые принадлежат одной из нескольких категорий, также к товарам есть рекомендации по уходу. Имеется информации о пользователях системы, а именно: имя; фамилия; пароль; электронная почта; телефон; тип пользователя. Под типом пользователя подразумевается, то какими функциями он будет обладать (администратор или обычный пользователь). У пользователя есть возможность оставлять отзывы и оформлять заказы.

2.2 Логическая модель

### На основании описанной предметной области была построена логическая модель хранилища данных. Рис.1. В нее включены такие сущности как:

### Товары;

### Категории;

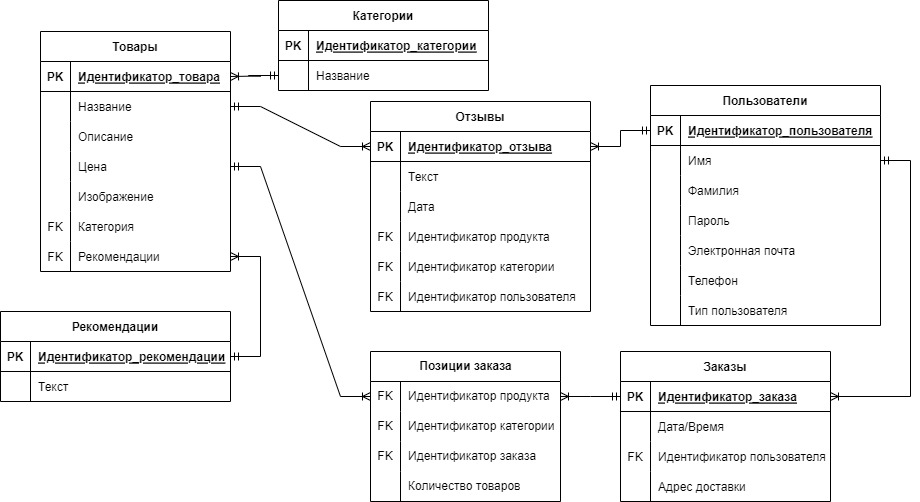
### Отзывы;

### Пользователи;

### Рекомендации;

### Позиции заказа;

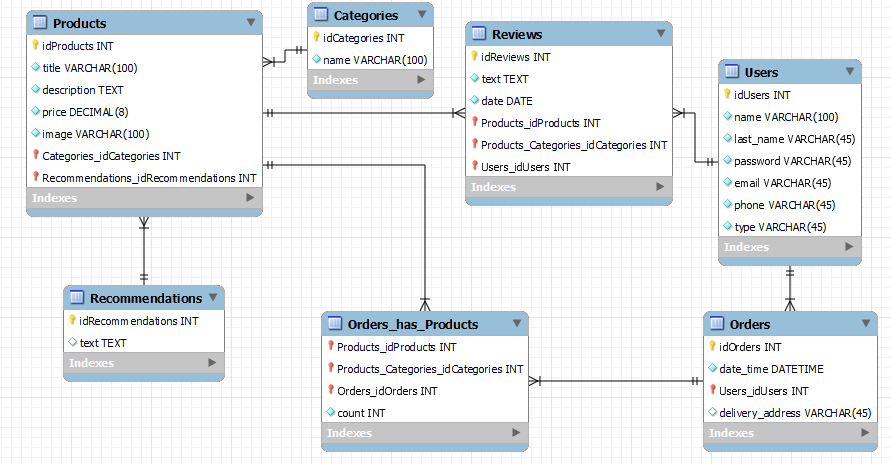
### Заказы.



### Рисунок 1 – Логическая модель хранилища данных.

2.3 Физическая модель

### Физическая модель представлена на рис.2. Рассмотрим подробнее поля каждой сущности.



### Рисунок 2 – Физическая модель хранилища данных

### **1. Products (товары)** – данная сущность содержит такие поля как:

### Идентификатор товара – выбран тип INT, так как чаще всего идентификаторы являются целочисленными.

### Название – выбран тип VARCHAR(100), так как для названия хватит 100 символов.

### Описание – выбран тип TEXT, так как подразумевается, что описание товара может быть объемным, как раз такой тип позволяет хранить большие данные.

### Цена – выбран тип DECIMAL(8), так как 8 символов хватит для товаров, которые планируется продавать, так же данный тип позволяет хранить точное вещественное значение данных, он используется, когда точность важна.

### Изображение – выбран тип VARCHAR(100), так как для описания пути изображения 100 символов будет достаточно.

### Идентификатор категории – выбран тип INT, аналогично идентификатору товара.

### Идентификатор рекомендации – выбран тип INT, аналогично идентификатору товара.

### **2. Recommendation (рекомендации)** – данная сущность содержит такие поля как:

### Идентификатор рекомендации – выбран тип INT, аналогично идентификатору товара.

### Текст – выбран тип TEXT, так как объем рекомендации может быть достаточно большим.

### **3. Categories (категории)** – данная сущность содержит такие поля как:

### Идентификатор категории – выбран тип INT, аналогично идентификатору товара.

### Название – выбран тип VARCHAR(100), так как для описания названия этого достаточно.

### **4. Reviews (отзывы)** – данная сущность содержит такие поля как:

### Идентификатор отзыва – выбран тип INT, аналогично идентификатору товара.

### Текст – выбран тип TEXT, так как отзыв может быть большим по содержанию.

### Дата – выбран тип DATE, который позволяет хранить дату отзыва.

### Идентификатор товара – выбран тип INT, аналогично идентификатору товара.

### Идентификатор категории – выбран тип INT, аналогично идентификатору товара.

### Идентификатор пользователя – выбран тип INT, аналогично идентификатору товара.

### **5. Users (пользователи)** – данная сущность содержит такие поля как:

### Идентификатор пользователя – выбран тип INT, аналогично идентификатору товара.

### Имя – выбран тип VARCHAR(100), так как для этого поля такое количество символов будет вполне достаточно.

### Фамилия – выбран тип VARCHAR(45), данное количество символов позволит описать это поле 45 символами.

### Пароль – выбран тип VARCHAR(45), аналогично фамилии.

### Электронная почта – выбран тип VARCHAR(45), аналогично фамилии.

### Номер телефона – выбран тип VARCHAR(45), аналогично фамилии.

### Тип пользователя – выбран тип VARCHAR(45), аналогично фамилии.

### **6. Orders (заказы)** – данная сущность содержит такие поля как:

### Идентификатор заказа – выбран тип INT, аналогично идентификатору товара.

### Дата/время – выбран тип DATETIME, что позволяет в одном поле хранить сразу два значения.

### Идентификатор пользователя – выбран тип INT, аналогично идентификатору товара.

### Адрес доставки – выбран тип VARCHAR(45), аналогично фамилии.

### **7. Orders\_has\_Products (позиции заказа)** – данная сущность содержит такие поля как:

### Идентификатор товара – выбран тип INT, аналогично идентификатору товара.

### Идентификатор категории – выбран тип INT, аналогично идентификатору товара.

### Идентификатор заказа – выбран тип INT, аналогично идентификатору товара.

### Количество товаров – выбран тип INT, так как количество товаров не может быть дробным числом.

2.4 Связи между сущностями

### Так как у одной категории может быть несколько товаров, то выбрана связь «один ко многим». Аналогичным образом связаны таблицы «Рекомендации» и «Товары», основываясь на том, что у многих товаров может быть одна и та же рекомендация.

### Такую же связь имеют таблицы «Товары» и «Отзывы», к одному товару относится много отзывов.

### Таблицы «Пользователи» и «Отзывы» связаны по принципу, у одного пользователя может быть много отзывов.

### Таблицы «Пользователи» и «Заказы» также имеют связь «один ко многим», исходя из того, что у одного пользователя может быть много заказов.

### Таблицы «Заказы» и «Товары» связаны между собой связью «многие ко многим» и имеют отдельную связывающую таблицу «Позиции в заказе».

1. Программный продукт

### Разработка велась в среде Visual Studio на языке C# с использованием Windows Form.

### База данных реализована в СУБД MySQL Workbench.

3.1 Функционал

### Задача авторизации сверить введенные данные пользователя с данными в базе и открыть в зависимости от этого личный кабинет. Если данные введены некорректно, то открывается окно с ошибкой. Рис. 3.

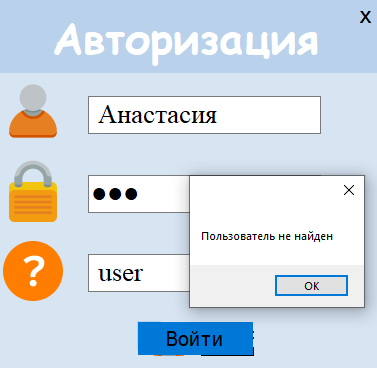


Рисунок 3 – Авторизация.

При успешной авторизации как администратор пользователь видит следующее окно, представленное на рис.4.

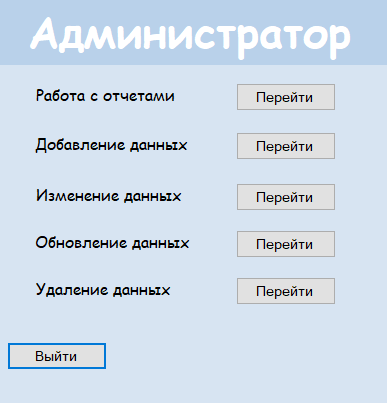


Рисунок 4 – Личный кабинет администратора.

У администратора имеется пять функций такие как:

* Работа с отчетами;
* Добавление данных;
* Изменение данных;
* Обновление данных;
* Удаление данных.

Рассмотрим подробнее каждую.

1. Работа с отчетами представлена на рис. 5.

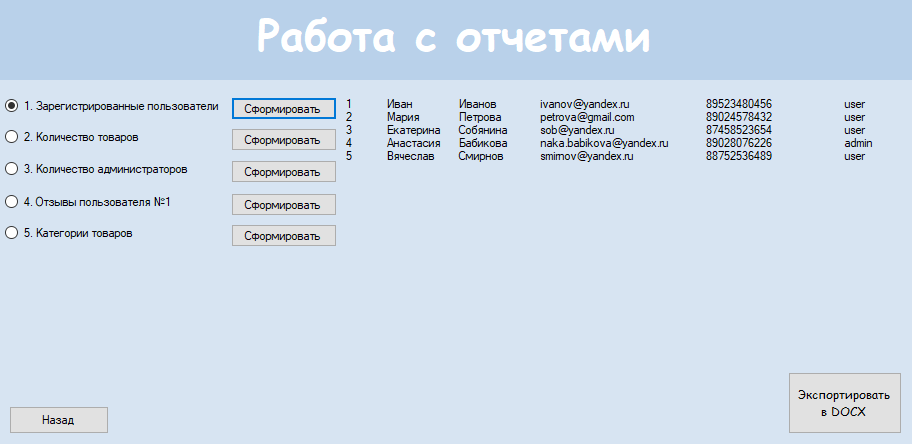


Рисунок 5 – Работа с отчетами.

При нажатии на кнопку «Сформировать», к базе данных отправляется запрос, например чтобы получить зарегистрированных пользователей запрос выглядит следующим образом:

MySqlCommand command = new MySqlCommand("select \* from users", db.getConnection());

### Где первая часть отвечает за сам запрос, а вторая за подключение к базе. Аналогичным образом реализованы остальные запросы.

Для того чтобы экспортировать данный отчет в DOCX необходимо нажать слева от отчета кнопку (radioButton), а затем кнопку «Экспортировать в DOCX». При успешном экспорте выводится всплывающее окно как на рис. 6.

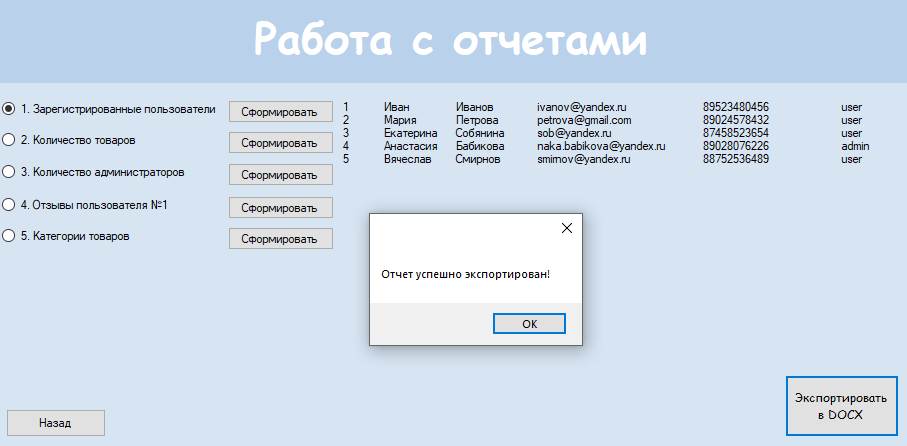


Рисунок 6 – Экспорт отчета.

Отчет в формате DOCX не удалось реализовать в заявленном формате. Вариант отчета представлен на рис.7.

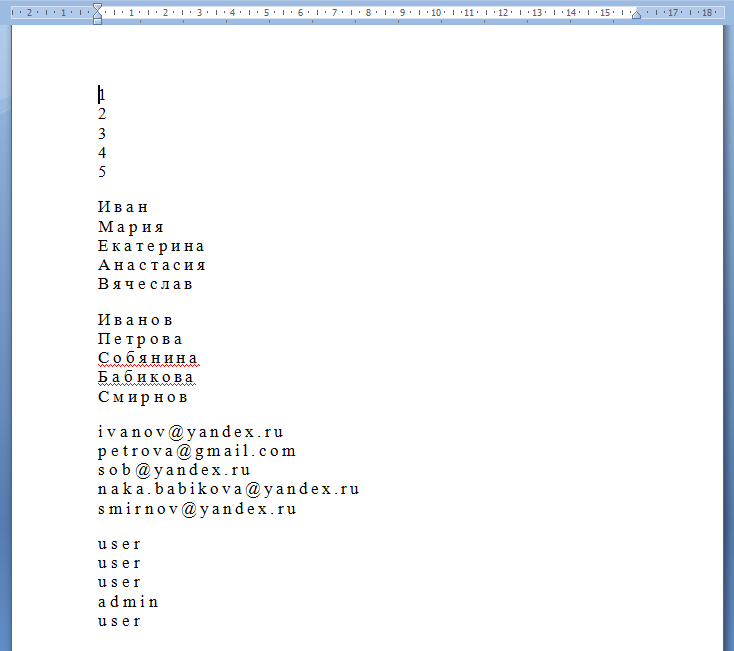


Рисунок 7 – Отчет в формате DOCX.

2. Добавление данных представлено на рис. 8.

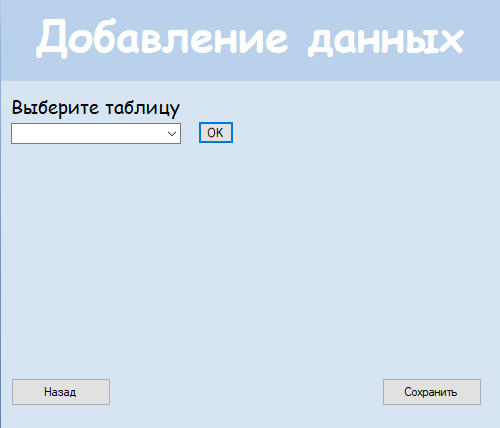


Рисунок 8 – Добавление данных.

Для того чтобы добавить данные для начала необходимо выбрать таблицу и нажать кнопку «ОК».

Если таблица была не выбрана и нажата кнопка «ОК», то выводится всплывающее окно представленное на рис.9.

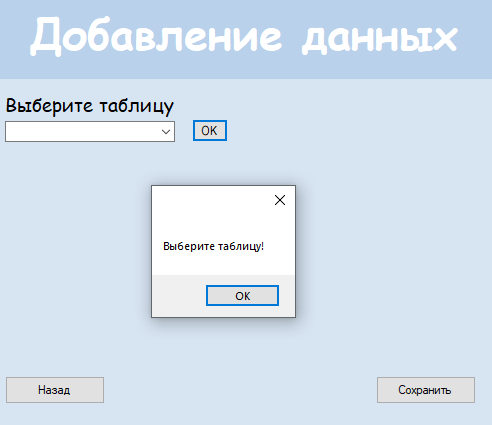


Рисунок 9 – Всплывающее окно с выбором таблицы.

При успешном выборе таблицы для каждой таблицы выводятся соответствующие поля, которые необходимо заполнить. Рис. 10.

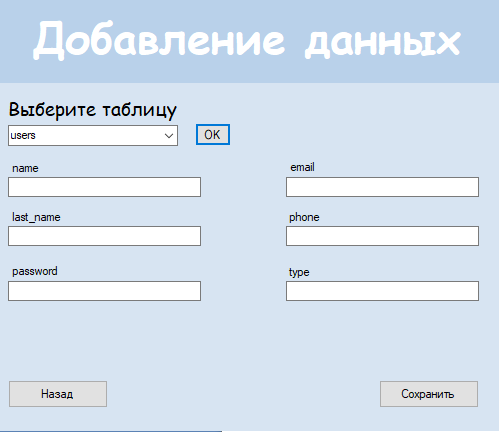


Рисунок 10 – Добавление данных в таблицу «users».

Для некоторых полей реализована проверка на пустоту, если пользователь не заполнил поле и нажал кнопку «Сохранить», то всплывает окно с предупреждением. Рис.11.

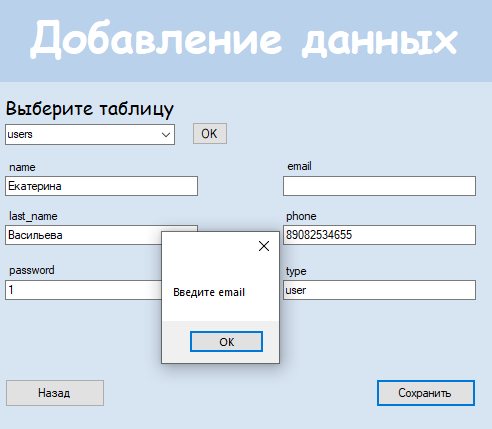


Рисунок 11 – Окно с предупреждением.

При успешном заполнение всех полей после нажатия на кнопку «Сохранить» получаем следующее сообщение. Рис. 12.

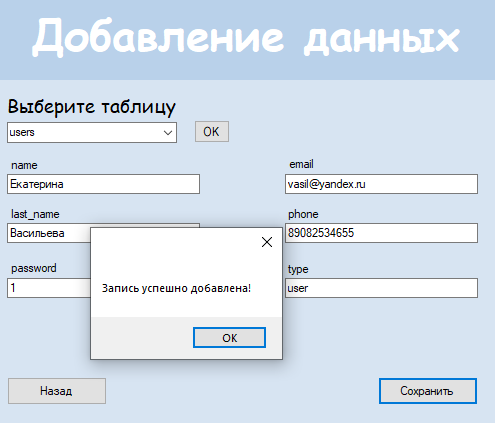


Рисунок 12 – Успешное добавление в базу.

3. Изменение данных представлено на рис. 13.

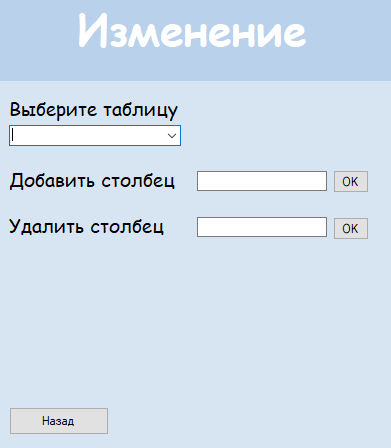


Рисунок 13 – Изменение данных.

4. Обновление данных представлено на рис.14.

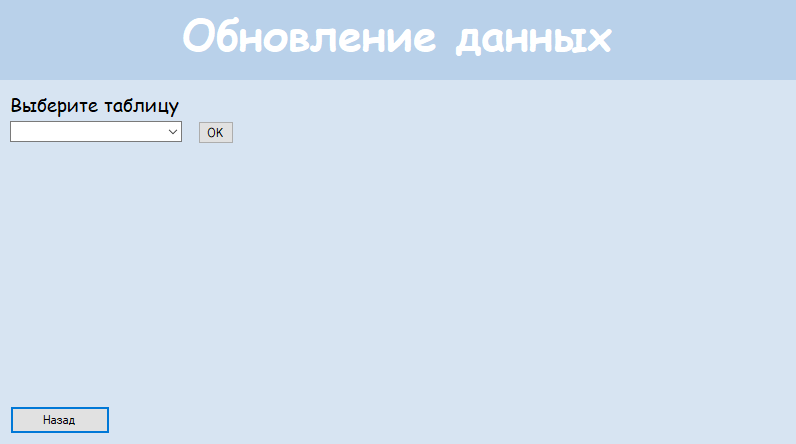


Рисунок 14 – Обновление данных.

Для обновления необходимо выбрать таблицу и нажать кнопку «ОК». Для каждой таблицы обновляемые поля отличаются. На рис. 15 обновление выполнялось для таблицы «orders».

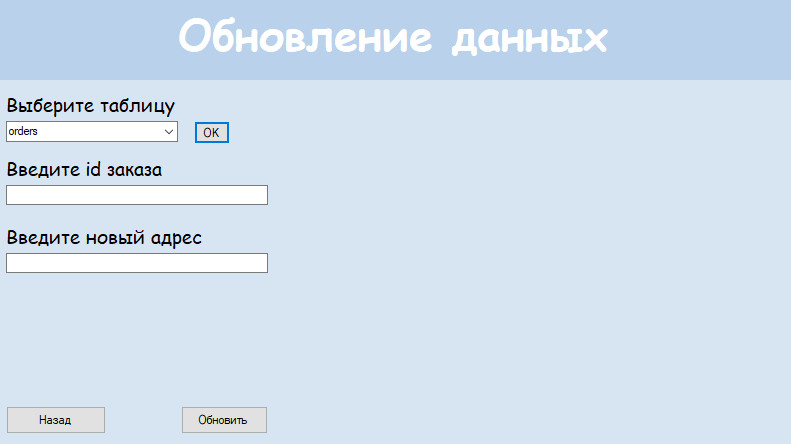


Рисунок 15 – Обновление таблицы «orders».

До обновления таблица в базе выглядела следующим образом. Рис. 16.

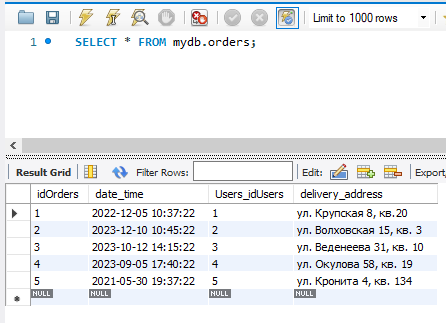


Рисунок 16 – Таблица «orders».

Заполнили поля и нажали кнопку «Обновить». К базе отправился запрос на обновление

MySqlCommand command = new MySqlCommand("UPDATE `orders` SET `delivery\_address`=@adress WHERE `idOrders`=@id", db.getConnection()); Рис.17.

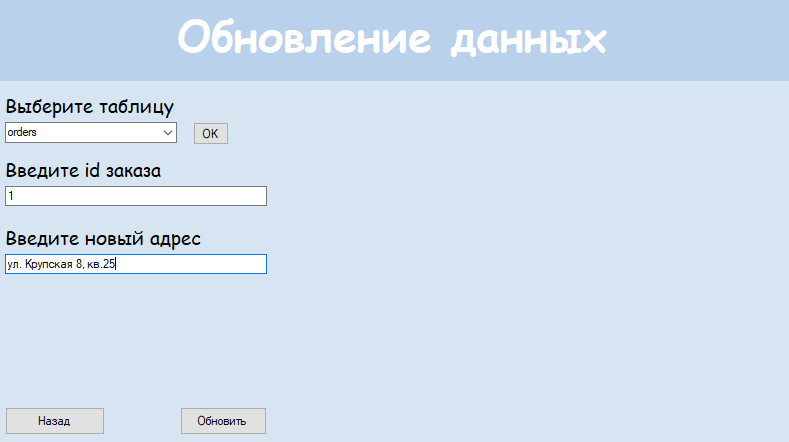


Рисунок 17 – Заполненные поля для обновления.

После успешного обновления было получено всплывающее окно представленное на рис. 18.

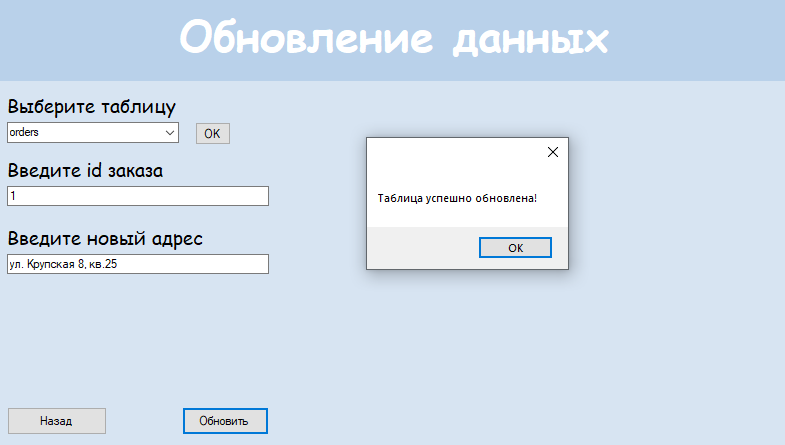


Рисунок 18 – Успешное обновление.

После этого база выглядит следующим образом. Рис. 19.

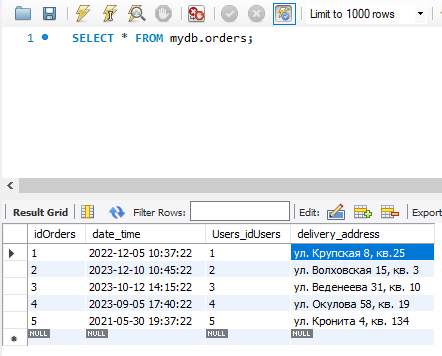


Рисунок 19 – Обновленная таблица «orders»

5. Удаление данных представлено на рис. 20.

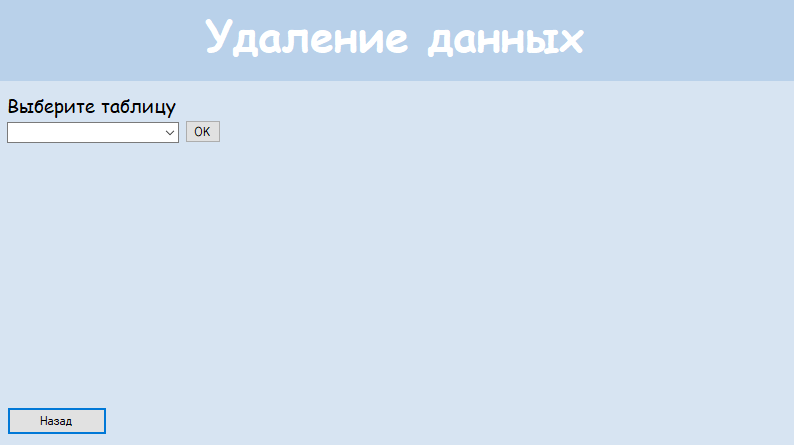


Рисунок 20 – Удаление данных.

Для того чтобы удалить данные необходимо выбрать таблицу и нажать кнопку «ОК». После этого появляется поле, пример представлен на рис.21.

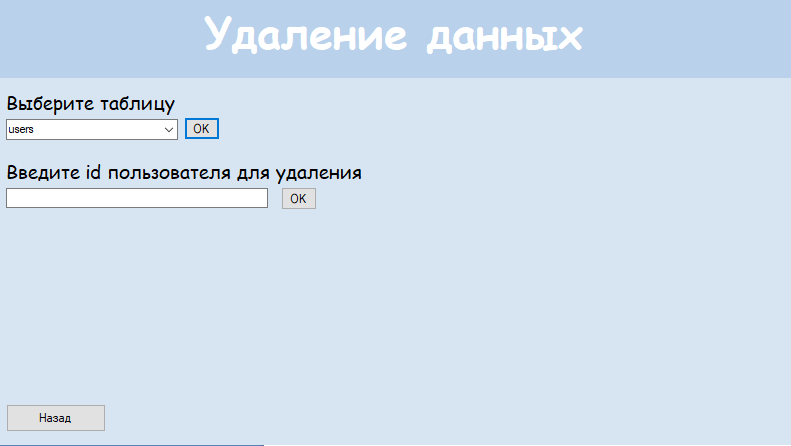


Рисунок 21 – Удаление записи по id.

При успешном удалении всплывает окно, представленное на рис.22.

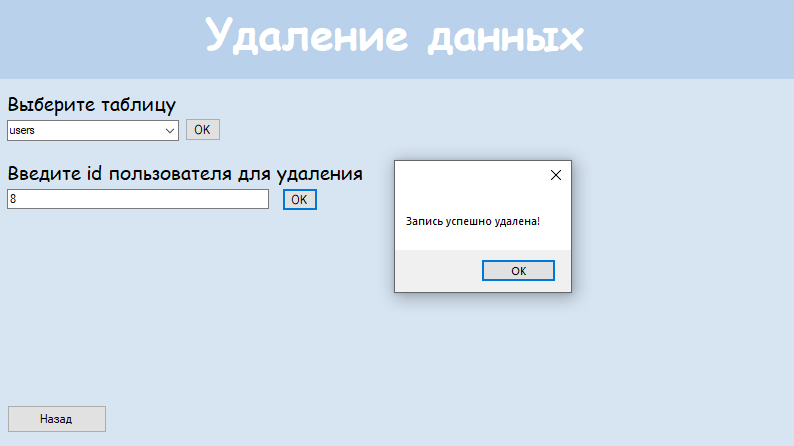


Рисунок 22 – Успешное удаление записи.

При успешной авторизации как пользователь отображается следующее окно, представленное на рис.23.

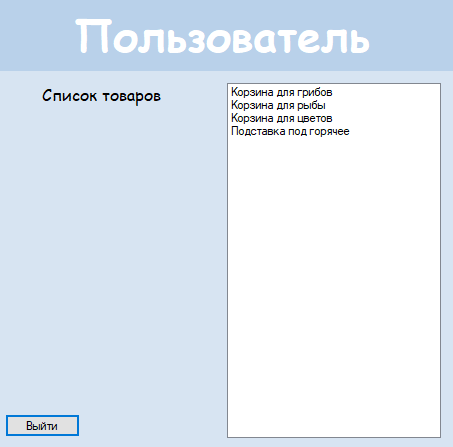


Рисунок 23 – Личный кабинет пользователя.

Пользователь имеет возможность просматривать товары. При нажатии на кнопку «Выйти» пользователь попадает на окно с авторизацией.

Выводы

Таким образом, в ходе курсовой работы была проанализирована предметная область и на основании этого сформулировано техническое задание, на основе которого велась разработка системы. Разработаны логическая и физическая модели хранилища данных. Построена схема базы данных в СУБД MySQL Workbench. Разработан понятный и удобный интерфейс. В ходе работы было проведено ручное тестирование основного функционала системы, были выявлены недостатки, которые в последующем планируется исправить и выполнить повторное тестирование.

список литературы

1. Проектирование и разработка реляционных баз данных : Учебно-методическое пособие по курсу «Базы данных» для студентов, обучающихся по направлениям 01.03.04 «Прикладная математика», 27.03.03 «Системный анализ и управление». – Ижевск : Ижевский государственный технический университет имени М.Т. Калашникова, 2017. – 47 с. – EDN IHEQCI.
2. Галанина, В. А. Базы данных : введение в теорию реляционных баз данных : учебное пособие / В. А. Галанина ; В. А. Галанина ; Федеральное агентство по образованию, Гос. образовательное учреждение высш. проф. образования Санкт-Петербургский гос. ун-т аэрокосмического приборостроения. – Санкт-Петербург : ГУАП, 2008. – 107 с. – ISBN 978-5-8088-0396-1. – EDN QMTHIH.
3. Мартишин, С. А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench : методы и средства проектирования информационных систем и технологий : инструментальные средства информационных систем : учебное пособие для студентов высших учебных заведений, обучающихся по направлению 230400 "Информационные системы и технологии" / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко ; С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ИД "Форум", 2012. – (Высшее образование). – ISBN 978-5-8199-0517-3. – EDN QMXDLT.
4. Иванов, К. А. Основные аспекты работы с инструментом для визуального проектирования баз данных MYSQL Workbench / К. А. Иванов, А. В. Тарутин // Инновационные технологии: теория, инструменты, практика. – 2020. – Т. 1. – С. 4-9. – EDN UNYTNY.
5. Авдиль, С. Л. Возможности объектно-ориентированного языка программирования С# (С SHARP) / С. Л. Авдиль, Э. А. Бекирова // . – 2018. – № 3(21). – С. 113-117. – EDN YQVQFN.
6. Марченко, А. Л. Основы программирования на C# 2.0 : Учебное пособие / А. Л. Марченко. – Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. – 552 с. – ISBN 978-5-4487-0084-2. – EDN ZUZDVP.
7. Троелсен, Э. C# и платформа .NET 3.0 : специальное издание / Э. Троелсен ; Эндрю Троелсен ; [пер. с англ. В. Щербинин]. – Москва [и др.] : Питер, 2008. – ISBN 978-5-91180-518-0. – EDN QMSQNJ.

*Приложение 1*

### **1. Разделенный доступ к информационной базе.**

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

{

String loginUser = loginField.Text;

String passUser = passField.Text;

String typeUser = typeField.Text;

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `users` WHERE `name` = @uL AND `password` = @uP AND `type` = @uT ", db.getConnection());

command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = loginUser;

command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = passUser;

command.Parameters.Add("@uT", MySqlDbType.VarChar).Value = typeUser;

adapter.SelectCommand = command;

adapter.Fill(table);

if(table.Rows.Count > 0)

{

if(typeUser == "admin")

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

if(typeUser == "user")

{

Form3 form3 = new Form3();

form3.Show();

this.Hide();

}

}

else

{

MessageBox.Show("Пользователь не найден");

}

}

### **2. Добавление, изменение и удаление данных БД.**

### *Добавление*

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

{

var table = comboBoxTables.Text;

if (table == "categories")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите название");

return;

}

if (isCategoriesExist())

{

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `categories` (`name`) VALUES (@name)", db.getConnection());

command.Parameters.Add("@name", MySqlDbType.VarChar).Value = textBox1.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Запись успешно добавлена!");

}

else

{

MessageBox.Show("Запись НЕ добавлена!");

}

db.closeConnection();

}

*Изменение*

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

{

var table = comboBoxTables.Text;

if (table == "categories")

{

DB db = new DB();

MySqlCommand command\_1 = new MySqlCommand("SELECT count(\*) FROM information\_schema.columns WHERE table\_name = 'categories'", db.getConnection());

MySqlCommand command = new MySqlCommand("ALTER TABLE `categories` ADD COLUMN new NVARCHAR(50) NOT NULL DEFAULT 'Неизвестен'", db.getConnection());

MySqlCommand command\_2 = new MySqlCommand("SELECT count(\*) FROM information\_schema.columns WHERE table\_name = 'categories'", db.getConnection());

String field = textBox1.Text;

command.Parameters.Add("@name", MySqlDbType.VarChar).Value = field;

db.openConnection();

command.ExecuteNonQuery();

if (command\_1 != command\_2)

{

MessageBox.Show("Столбец успешно добавлен!");

}

else

{

MessageBox.Show("Столбец НЕ добавлен!");

}

db.closeConnection();

}

}

*Удаление*

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

{

var table = comboBoxTables.Text;

if (table == "categories")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("DELETE FROM `categories` WHERE `idCategories` = @idCategories", db.getConnection());

command.Parameters.Add("@idCategories", MySqlDbType.VarChar).Value = textBox1.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Запись успешно удалена!");

}

else

{

MessageBox.Show("Запись НЕ удалена!");

}

db.closeConnection();

}

### **3. Формирование отчетов.**

### //Зарегистрированные пользователи

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

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("select \* from users", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

idLabel.Visible = true;

nameLabel.Visible = true;

lastNameLabel.Visible = true;

emailLabel.Visible = true;

phoneLabel.Visible = true;

typeLabel.Visible = true;

int indicator = 0;

foreach (DataRow row in table.Rows)

{

if (indicator > 0)

{

int id = (int)row[0];

idLabel.Text += Convert.ToString(id) + Environment.NewLine;

string name = (string)row[1];

this.nameLabel.Text += name + Environment.NewLine;

string lastName = (string)row[2];

this.lastNameLabel.Text += lastName + Environment.NewLine;

string email = (string)row[4];

this.emailLabel.Text += email + Environment.NewLine;

string phone = (string)row[5];

this.phoneLabel.Text += phone + Environment.NewLine;

string type = (string)row[6];

this.typeLabel.Text += type + Environment.NewLine;

}

else

{

indicator += 1;

int id = (int)row[0];

idLabel.Text = Convert.ToString(id) + Environment.NewLine;

string name = (string)row[1];

this.nameLabel.Text = name + Environment.NewLine;

string lastName = (string)row[2];

this.lastNameLabel.Text = lastName + Environment.NewLine;

string email = (string)row[4];

this.emailLabel.Text = email + Environment.NewLine;

string phone = (string)row[5];

this.phoneLabel.Text = phone + Environment.NewLine;

string type = (string)row[6];

this.typeLabel.Text = type + Environment.NewLine;

}

}

}

//Экспорт в DOCX

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

{

//1.Зарегистрированные пользователи

if (radioButton1.Checked)

{

// путь к документу

string pathDocument = AppDomain.CurrentDomain.BaseDirectory + "report1.docx";

// создаём документ

DocX document = DocX.Create(pathDocument);

// вставляем параграф и передаём текст

document.InsertParagraph(idLabel.Text).

// устанавливаем шрифт

Font("Times new Roman").

//// устанавливаем размер шрифта

FontSize(14).

//// устанавливаем интервал между символами

Spacing(3);

document.InsertParagraph(nameLabel.Text).

// устанавливаем шрифт

Font("Times new Roman").

//// устанавливаем размер шрифта

FontSize(14).

//// устанавливаем интервал между символами

Spacing(3);

document.InsertParagraph(lastNameLabel.Text).

// устанавливаем шрифт

Font("Times new Roman").

//// устанавливаем размер шрифта

FontSize(14).

//// устанавливаем интервал между символами

Spacing(3);

document.InsertParagraph(emailLabel.Text).

// устанавливаем шрифт

Font("Times new Roman").

//// устанавливаем размер шрифта

FontSize(14).

//// устанавливаем интервал между символами

Spacing(3);

document.InsertParagraph(typeLabel.Text).

// устанавливаем шрифт

Font("Times new Roman").

//// устанавливаем размер шрифта

FontSize(14).

//// устанавливаем интервал между символами

Spacing(3);

// сохраняем документ

document.Save();

MessageBox.Show("Отчет успешно экспортирован!");

}

### **4. Обработка исключительных ситуаций.**

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

{

var table = comboBoxTables.Text;

if (table == "orders")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

if (textBox2.Text == "")

{

MessageBox.Show("Введите адрес");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("UPDATE `orders` SET `delivery\_address`=@adress WHERE `idOrders`=@id", db.getConnection());

command.Parameters.Add("@adress", MySqlDbType.VarChar).Value = textBox2.Text;

command.Parameters.Add("@id", MySqlDbType.VarChar).Value = textBox1.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Таблица успешно обновлена!");

}

else

{

MessageBox.Show("Таблица НЕ обновлена!");

}

db.closeConnection();

}