**СОДЕРЖАНИЕ**

[1. Техническое задание 5](#_Toc132286756)

[1.1 Общие сведения 5](#_Toc132286757)

[1.1.1 Наименование системы 5](#_Toc132286758)

[1.1.2 Исполнитель 5](#_Toc132286759)

[1.1.3 Плановые сроки начала и окончания работ по разработке системы 5](#_Toc132286760)

[1.2 Назначение и цели создания системы 5](#_Toc132286761)

[1.2.1 Назначение системы 5](#_Toc132286762)

[1.3 Цели создания 5](#_Toc132286763)

[1.4 Требования к системе 6](#_Toc132286764)

[1.4.1 Требования к системе в целом 6](#_Toc132286765)

[1.4.1.1 Требования к структуре и функционированию системы 6](#_Toc132286766)

[1.4.1.2 Требования к показателям назначения 7](#_Toc132286767)

[1.4.1.3 Требования к надежности 8](#_Toc132286768)

[1.4.1.4 Требования к безопасности 11](#_Toc132286769)

[1.4.1.5 Требования к эргономике и технической эстетике 11](#_Toc132286770)

[1.4.1.6 Требования к эксплуатации 11](#_Toc132286771)

[1.4.1.7 Требования к защите информации от несанкционированного доступа 12](#_Toc132286772)

[1.4.1.8 Требования по сохранности информации при авариях 13](#_Toc132286773)

[1.5 Порядок разработки автоматизированной системы 14](#_Toc132286774)

[2. Хранилище данных 17](#_Toc132286775)

[2.1 Предметная область 17](#_Toc132286776)

[2.2 Логическая модель 17](#_Toc132286777)

[2.3 Физическая модель 19](#_Toc132286778)

[2.4 Связи между сущностями 20](#_Toc132286779)

[2.5 Содержимое базы данных 20](#_Toc132286780)

[Вывод 22](#_Toc132286781)

[3. Интерфейс 23](#_Toc132286782)

[3.1 Средства разработки 23](#_Toc132286783)

[3.2 Элементы интерфейса 23](#_Toc132286784)

[3.2.1 Форма авторизации. 23](#_Toc132286785)

[3.2.2 Вид личного кабинета пользователя. 23](#_Toc132286786)

[3.2.3 Вид личного кабинета администратора. 24](#_Toc132286787)

[3.2.4 Вид отчета в файле docx. 27](#_Toc132286788)

[3.2.5 Реакция ИС на ошибочный ввод данных. 27](#_Toc132286789)

[Вывод 28](#_Toc132286790)

[4. Диаграмма классов 29](#_Toc132286791)

[4.1 Назначение классов и их методов 29](#_Toc132286792)

[Вывод 31](#_Toc132286793)

[5. Реализация 32](#_Toc132286794)

[5.1 Средства разработки 32](#_Toc132286795)

[5.2 Функционал 32](#_Toc132286796)

[Вывод 43](#_Toc132286797)

[6. Тестирование 44](#_Toc132286798)

[6.1 Реализация 44](#_Toc132286799)

[6.1.1 Тестировщик и вид тестирования 44](#_Toc132286800)

[6.1.2 Тестирование основного функционала 44](#_Toc132286801)

[Вывод 53](#_Toc132286802)

[Приложение 1 54](#_Toc132286803)

1. Техническое задание

*1.1 Общие сведения*

1.1.1 Наименование системы

### Полное наименование системы – Корпоративная информационная система, осуществляющая функции редактирования, создания, чтения, удаления и обновления товаров.

1.1.2 Исполнитель

### Разработчиком системы является Бабикова Анастасия Сергеевна.

1.1.3 Плановые сроки начала и окончания работ по разработке системы

10.02.2023 – 14.04.2023

*1.2 Назначение и цели создания системы*

1.2.1 Назначение системы

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

*1.3 Цели создания*

### Цель создания системы:

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

*1.4 Требования к системе*

1.4.1 Требования к системе в целом

1.4.1.1 Требования к структуре и функционированию системы

### **Требования к общей схеме архитектуры системы**

### Архитектура системы должна представлять собой клиент-серверную архитектуру.

### **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.

1.4.1.2 Требования к показателям назначения

### В качестве основного параметра, характеризующего степень соответствия системы ее назначению, необходимо принять информационно-технологическую емкость системы, которая выражается в поддержке системой следующих параметров на момент ввода системы в эксплуатацию:

* период хранения архивных данных – не менее 5 лет;
* период накопления и оперативной обработки информации – не менее 5 лет.
* минимальный срок эксплуатации, при котором сохраняется целевое назначение системы – не менее 5 лет.

### **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).

1.4.1.3 Требования к надежности

### **Основные показатели надежности системы**

### Под надежностью системы следует понимать комплексное свойство системы сохранять во времени в установленных нормативно-технической и/или конструкторской документацией пределах значения параметров, характеризующих способность системы выполнять свои функции, определяемые ее назначением, режимами и условиями эксплуатации.

### Система должна обеспечивать:

### − целостность и корректность данных при разрыве соединения во время взаимодействия со смежными системами;

### − возможность создания резервных копий конфигурации и данных с использованием внешних средств резервного копирования;

### − возможность восстановления резервных копий конфигурации и данных с использованием внешних средств резервного копирования;

### − обеспечивать восстановление работоспособности при появлении сбоев, аварий и отказов, возникающих на аппаратном обеспечении;

### − обеспечивать восстановление работоспособности при появлении сбоев, аварий и отказов, возникающих на сервере базы данных посредством стандартных механизмов резервного копирования и восстановления данных;

### − система в целом не должна терять работоспособность в случае возникновения сбоев, аварий и отказов, возникающих на рабочих станциях пользователей и печатающих устройствах;

### − система в целом должна сохранять работоспособность при некорректных действиях пользователей;

### − обеспечивать при отказе интерфейса администрирования сохранность работоспособности системы как в целом, так и её отдельных компонентов;

### − в целях восстановления системы в целом, или её компонентов после аварийного случая должны быть разработаны процедуры резервного копирования. система должна обеспечивать круглосуточный режим функционирования 7 дней в неделю.

### Состав и количественные значения показателей надежности для Системы в целом должны быть следующими:

### − время восстановления после сбоя Системы должно определяться как промежуток времени в минутах, в течение которого обслуживающий персонал, в случае необходимости производит восстановление базы данных Системы из резервной копии, и запустит комплекс программ;

### − совокупный простой в год Системы должен определяться как промежуток времени в часах, в течение, которого Система находится в неработоспособном состоянии, вызванном аварийной обслуживанием.

### Количественные значения показателей надежности Системы:

### − время восстановления после сбоя должно быть не более 60 минут;

### − совокупный простой не должен превышать 8 часов в год.

### Минимальный срок эксплуатации системы не менее 5 лет, отдельных подсистем – не менее 5 лет.

### **Перечень аварийных ситуаций**

### Под аварийной ситуацией системе следует понимать такое состояние, которое характеризуется:

### полным или частичным прекращением выполнения функциональных задач системы;

### полным или частичным нарушением взаимодействия между компонентами системы

### нештатным режимом работы всей системы или ее основных компонентов;

### полной или частичной потерей данных;

### нелегитимным доступом к данным системы, и предумышленным ее искажением или уничтожением.

### Перечень возможно предпринимаемых мер, направленных на предотвращение аварийных ситуаций:

### принятие мер по резервированию устройств и модулей системы для немедленного или постепенного восстановления работы системы;

### своевременное реагирование на обращение по аварийным ситуациям и по поддержке системы в целом с участием квалифицированного персонала.

1.4.1.4 Требования к безопасности

### Общие требования пожарной безопасности клиентских рабочих мест системы должны соответствовать нормам на бытовое электрооборудование.

1.4.1.5 Требования к эргономике и технической эстетике

### Для персонального компьютера предусматривается использование графического дисплея с видеорежимом не менее 1024x768 точек или выше с частотой вертикальной развертки не менее 60 Гц.

### Пользовательский интерфейс системы реализуется на русском языке. В разделах программного обеспечения в составе системы, рассчитанных на взаимодействие с администраторами, инженерами и техническим обслуживающим персоналом, допускается использование интерфейса на английском языке.

1.4.1.6 Требования к эксплуатации

### Требования настоящего раздела являются общими для средств вычислительной техники, применяемых в составе системы.

### **Условия и режимы эксплуатации**

### Условия эксплуатации оборудования системы должны обеспечивать использование технических средств системы с заданными техническими показателями, включающими состояние окружающей среды, параметры электропитания, периодичность и характер технического обслуживания, а также иные условия, если это является требованием производителя оборудования.

### **Требования к электропитанию**

### Первичными источниками электропитания системы, включая средства представления информации, являются сети переменного тока напряжением 220В, частотой 50Гц объектов муниципального и промышленного хозяйства.

### В целях обеспечения непрерывной работы технических средств клиентских рабочих мест системы возможно применение схем подключения, обеспечивающих их бесперебойное электроснабжение (с применением аккумуляторных батарей).

### Электропитание устройств, обеспечивающих работу: СУБД, систем хранения ключевой информации, систем информационной безопасности и систем технологических защит должно осуществляться с применением схем подключения, обеспечивающих их гарантированное и бесперебойное электроснабжение.

### **Требования к регламентам обслуживания**

### Обслуживание технических средств системы производится Заказчиком согласно внутренним регламентам Заказчика.

### Объем, трудозатраты и порядок выполнения обслуживания должны соответствовать техническим условиям на эксплуатацию применяемых программно-технических средств.

1.4.1.7 Требования к защите информации от несанкционированного доступа

### Несанкционированный доступ к данным системы должен быть ограничен следующими средствами:

### административными и организационными средствами: серверное и коммуникационное оборудование БД системы должно быть размещено в физически защищенном помещении и оснащено средствами обеспечения ее бесперебойной работы. Доступ в серверное помещение должен быть ограничен с помощью технических средств контроля. Должны быть разработаны специальные административные регламенты, определяющие порядок доступа в указанные помещения, а также регулирующие доступ к данным системы;

### административными программными средствами операционной системы к отдельным ее компонентам и приложениям;

### административными программными средствами СУБД в соответствии с ролями пользователей;

1.4.1.8 Требования по сохранности информации при авариях

### Перечень событий, при которых должна обеспечиваться сохранность информации в системе:

### выключение электропитания в сетях общего пользования. Стабильность питания должна быть обеспечена устройствами резервного питания, обеспечивающими требуемые показатели по уровню, качеству, бесперебойности электропитания, в соответствии с регламентами обеспечения бесперебойной работы системы;

### выход из строя серверного оборудования, в результате механического повреждения его компонентов. Сохранность данных должна быть обеспечена в результате проведения резервного копирования данных, хранящихся в системе.

### При наступлении событий, связанных с физическим уничтожением серверного оборудования системы в целом, либо отдельных его частей, восстановление функционирования системы должно осуществляться в сроки и в порядке, определяемом соответствующими регламентами. В качестве источников информации для восстановления данных должны использоваться резервные копии данных, дистрибутивы системы, операционной системы, СУБД и прочего ПО, задействованного при реализации системы.

*1.5 Порядок разработки автоматизированной системы*

### **1. Проектирование хранилища данных**

### В хранилища данных помещают данные, которые редко меняются. Хранилища ориентированы на выполнение аналитических запросов, обеспечивающих поддержку принятия решений для руководителей и менеджеров. При проектировании хранилищ данных необходимо выполнять следующие требования:

### хранилище должно иметь понятную для пользователей структуру данных;

### должны быть выделены статические данные, которые модифицируются по расписанию (ежедневно, еженедельно, ежеквартально);

### должны быть упрощены требования к запросам для исключения запросов, требующих множественных утверждений SQL в традиционных реляционных СУБД;

### должна обеспечиваться поддержка сложных запросов SQL, требующих обработки миллионов записей.

### **2. Проектирование интерфейса**

### При разработке интерфейса следует учесть следующие критерии:

### Естественность интерфейса;

### Согласованность интерфейса;

### Дружественность интерфейса (Принцип «прощения пользователя»);

### Принцип «обратной связи»;

### Простота интерфейса;

### Гибкость интерфейса;

### Эстетическая привлекательность.

### **3. Создание диаграммы классов**

### Диаграмма классов предназначена для представления внутренней структуры программы в виде классов и связей между ними.

### **4. Реализация системы**

### **5. Тестирование системы**

### Тестирование системы будет проводиться вручную.

### Основная цель ручного тестирования – убедиться, что приложение работает нормально, без каких-либо функциональных дефектов и ошибок, а также в соответствии с требованиями спецификации. 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. Хранилище 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;

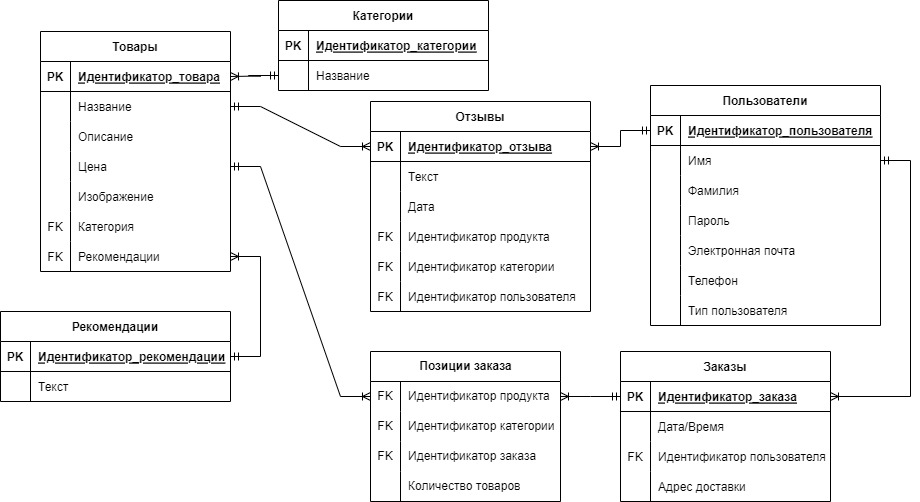
### Отзывы;

### Пользователи;

### Рекомендации;

### Позиции заказа;

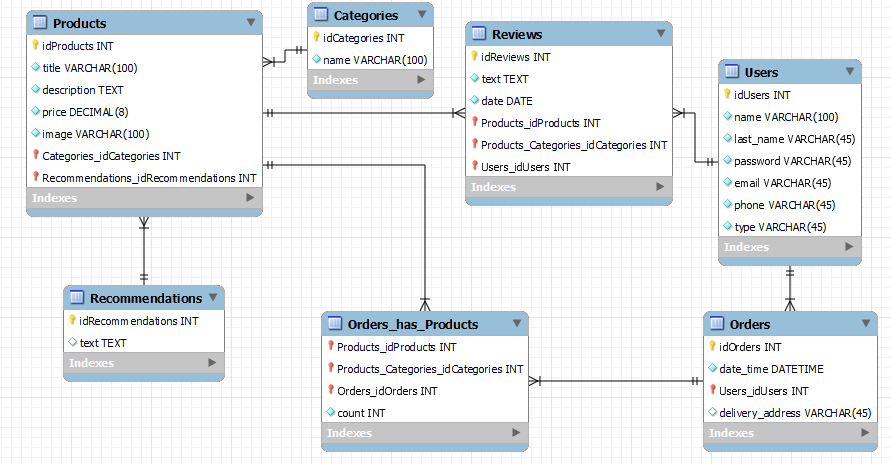
### Заказы.



### 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.

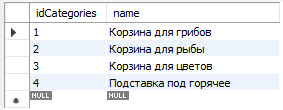
### Таблицы «Пользователи» и «Отзывы» связаны по принципу, у одного пользователя может быть много отзывов.

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

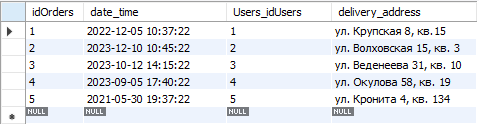
### Таблицы «Заказы» и «Товары» связаны между собой связью «многие ко многим» и имеют отдельную связывающую таблицу «Позиции в заказе».

2.5 Содержимое базы данных

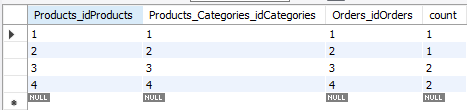
### После того как было спроектировано хранилище, в него были внесены данные. Рис. 3-9.



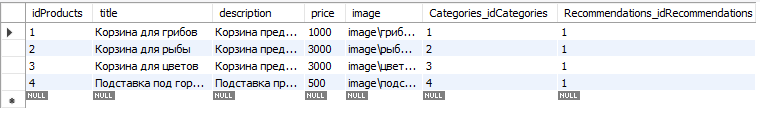
### Рисунок 3 – Таблица «Категории» с заполненными данными



### Рисунок 4 – Таблица «Заказы» с заполненными данными



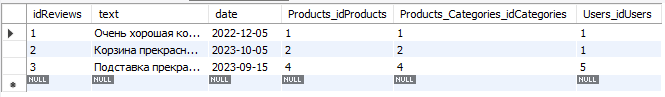
### Рисунок 5 – Таблица «Позиции заказа» с заполненными данными



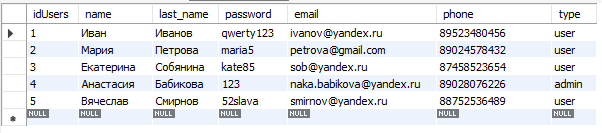
### Рисунок 6 – Таблица «Товары» с заполненными данными



### Рисунок 7 – Таблица «Рекомендации» с заполненными данными



### Рисунок 8 – Таблица «Отзывы» с заполненными данными



### Рисунок 9 – Таблица «Пользователи» с заполненными данными

Вывод

### В ходе данного задания были разработаны логическая и физическая модели хранилища КИС. После чего физическая модель базы была заполнена данными.

### В последующем планируется разработка интерфейса, который подразумевает работу с базой данных.

1. Интерфейс

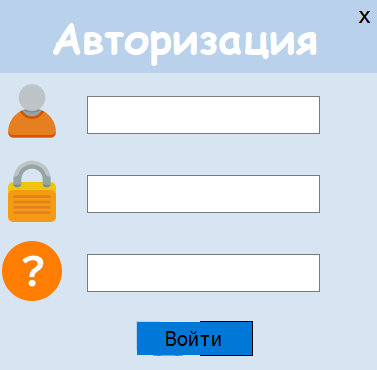
3.1 Средства разработки

### Интерфейс был разработан в среде Visual Studio на языке C# с применением Windows Form.

3.2 Элементы интерфейса

3.2.1 Форма авторизации.

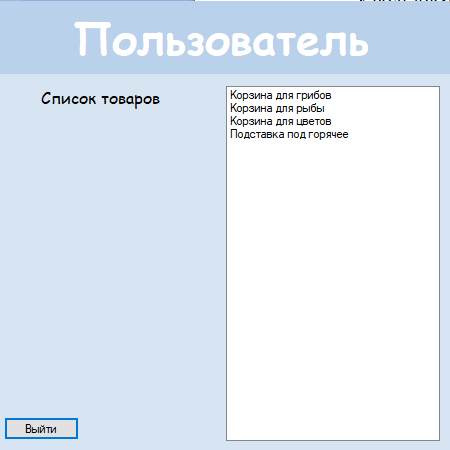
### Для авторизации пользователю необходимо ввести правильные логин и пароль, а также тип пользователя и нажать кнопку войти. Рис.10.



### Рисунок 10 – Авторизация

3.2.2 Вид личного кабинета пользователя.

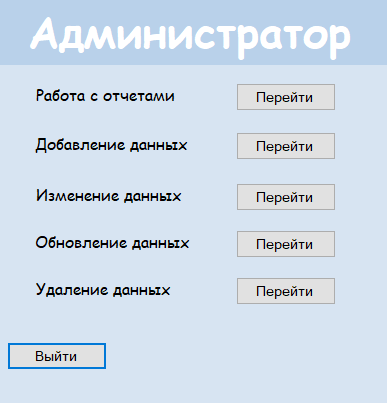
### У пользователя есть возможность просматривать список товаров. При нажатии на кнопку выйти, пользователь выходит из системы и автоматически возвращается к окну авторизации. Рис.11.



### Рисунок 11 – Страница мои данные

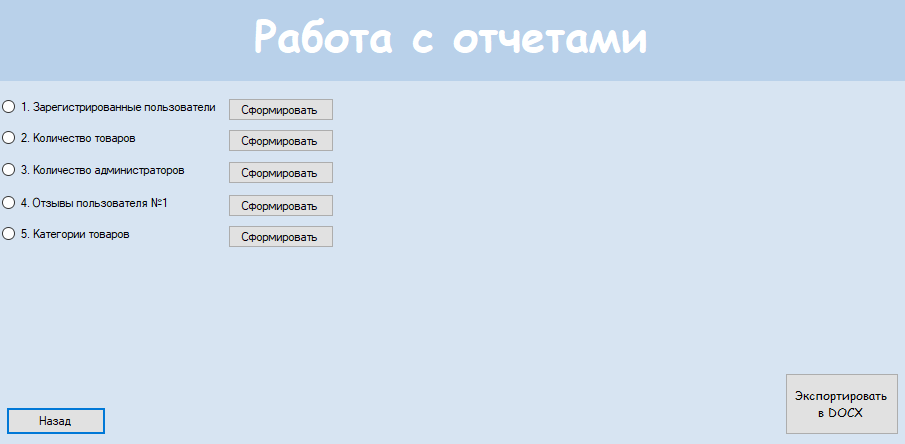
3.2.3 Вид личного кабинета администратора.

### По сравнению с пользователем, у администратора появляется возможность формировать различные отчеты, а так же работать с данными. Так же как и у пользователя при нажатии на кнопку выход, пользователь выходит из системы и автоматически возвращается к окну авторизации. На рис.12 представлен личный кабинет администратора.



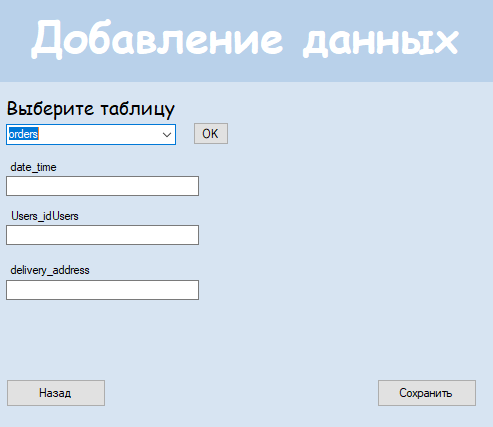
### Рисунок 12 – Личный кабинет администратора

### Окно для работы с отчетами выглядит следующим образом. Рис.13. Администратор может выбрать необходимые отчеты и выбрать режим просмотра. Кнопки «Сформировать» отвечают за просмотр в приложении. При нажатии на кнопку «Экспортировать в DOCX» будет происходить экспорт в файл формата DOCX.При нажатии на кнопку «Назад» администратор возвращается на первоначальную страницу.

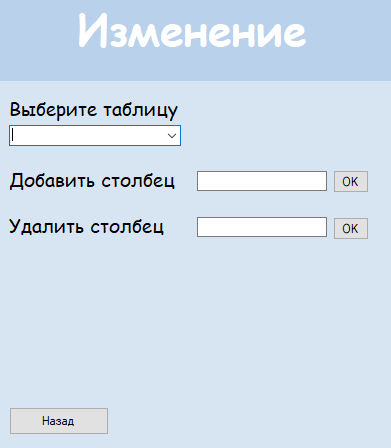


### Рисунок 13 – Окно отчеты

### Для работы с данными предусмотрено четыре различных страницы, которые представлены на рис.14-17. Для того чтобы начать работать с данными необходимо выбрать таблицу из выпадающего списка.



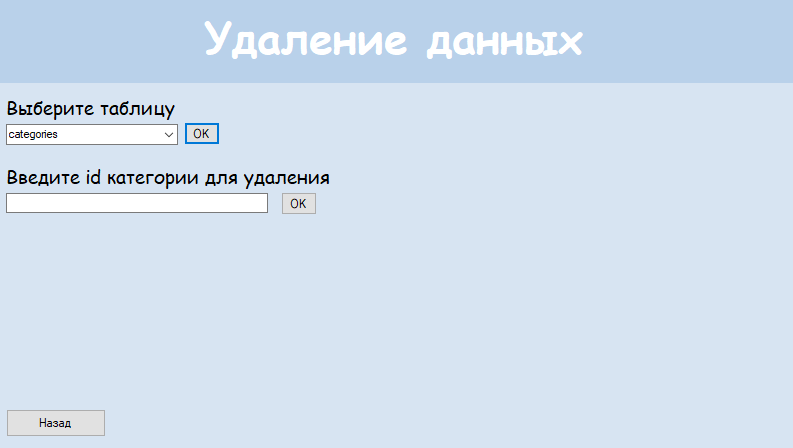
### Рисунок 14 – Добавление данных



### Рисунок 15 – Изменение данных

### 

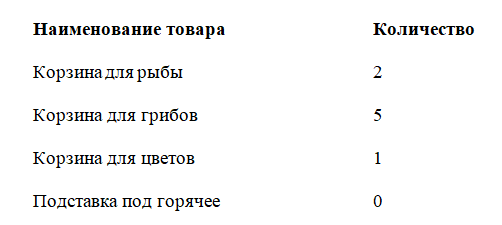
### Рисунок 16 – Обновление данных



### Рисунок 17 – Удаление данных

3.2.4 Вид отчета в файле docx.

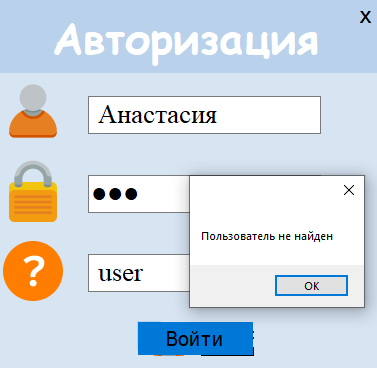
### Примерный формат отчета в docx. Рис.18.



### Рисунок 18 – Отчет «количество товаров»

3.2.5 Реакция ИС на ошибочный ввод данных.

### Если пользователь ошибочно введет логин, пароль или тип пользователя, то получит следующее сообщение. Рис.19.



### Рисунок 19 – Модальное окно при неправильном вводе данных

Вывод

### В ходе данного задания был разработан интерфейс КИС, а именно форма авторизации, личный кабинет, как для пользователя, так и для администратора. При вводе ошибочных данных пользователю отображается модальное окно.

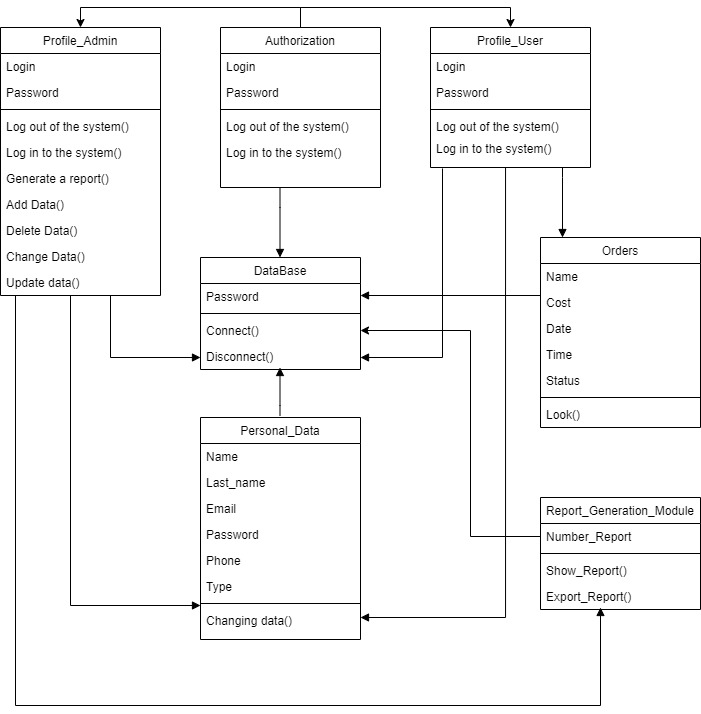
### В ходе разработки возможны изменения интерфейса. В последующем планируется подключение базы данных, которая была спроектирована на предыдущем этапе.

1. Диаграмма классов

### Диаграмма классов спроектирована средствами Diagrams.net. Diagrams.net – это бесплатное приложение для построения диаграмм, позволяющее пользователям создавать схемы, модели и диаграммы и обмениваться ими в веб-браузере.

4.1 Назначение классов и их методов

### На рис.20. представлена диаграмма классов.



### Рисунок 20 – Диаграмма классов

### **Класс Profile\_Admin** – предназначен для работы пользователя с типом «admin».

### Метод **«Log out of the system»** – метод, с помощью которого осуществляется выход из системы.

### Метод **«Log in to the system»** – метод, с помощью которого осуществляется вход в систему.

### Метод **«Generate a report»** – метод, с помощью которого формируется запрос к модулю генерации отчетов.

### Метод **«Add Data»** – метод, который позволяет добавлять данные в базу данных.

### Метод **«Delete Data»** – метод, который позволяет удалять данные из базы данных.

### Метод **«Change Data»** – метод, который позволяет изменять данные в базе данных.

### Метод **«Update Data»** – метод, который позволяет обновлять данные в базе данных.

### **Класс Authorization** – предназначен для авторизации пользователей в системе.

### Метод **«Log out of the system»** – метод, с помощью которого осуществляется выход из системы.

### Метод **«Log in to the system»** – метод, с помощью которого осуществляется вход в систему.

### **Класс Profile\_User** – предназначен для работы пользователя с типом «user».

### Метод **«Log out of the system»** – метод, с помощью которого осуществляется выход из системы.

### Метод **«Log in to the system»** – метод, с помощью которого осуществляется вход в систему.

### **Класс Orders** – предназначен для пользователя, который имеет доступ к просмотру заказов.

### Метод **«Look»** – метод, с помощью которого пользователь может просматривать свои заказы.

### **Класс Report\_Generation\_Module** – предназначен для взаимодействия администратора с отчетами.

### Метод **«Show\_Report»** – метод, с помощью которого отчет выводится в личный кабинет.

### Метод **«Export\_Report»** – метод, с помощью которого можно экспортировать отчет в формате DOCX.

### **Класс Personal\_Data** – предназначен для редактирования персональных данных пользователя.

### Метод **«Changing data»** – метод, который сохраняет изменения после редактирования данных.

### **Класс DataBase** – предназначен для взаимодействия всех компонентов системы с базой данных.

### Метод **«Connect»** – метод, с помощью которого происходит подключение к базе данных.

### Метод **«Disconnect»** – метод, с помощью которого происходит отключение от базы данных.

Вывод

### В ходе данного задания была разработана структура классов. Описаны классы и их методы.

### В последующем планируется реализация системы с объединением интерфейса и базы данных.

1. Реализация

5.1 Средства разработки

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

### База данных реализована в СУБД MySQL Workbench.

5.2 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. 21.

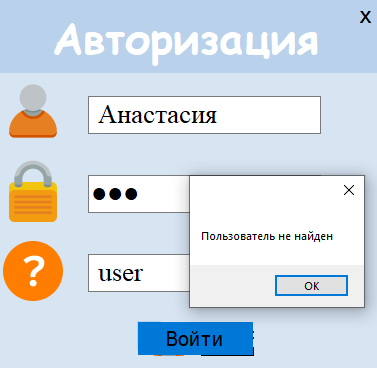


Рисунок 21 – Авторизация.

При успешной авторизации как администратор пользователь видит следующее окно, представленное на рис.22.

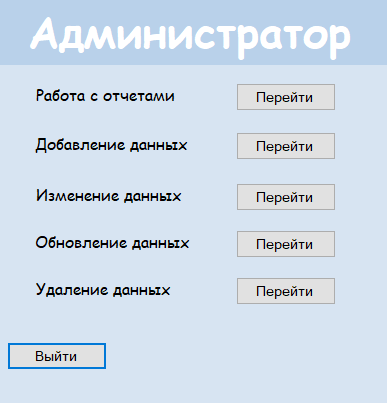


Рисунок 22 – Личный кабинет администратора.

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. 23.

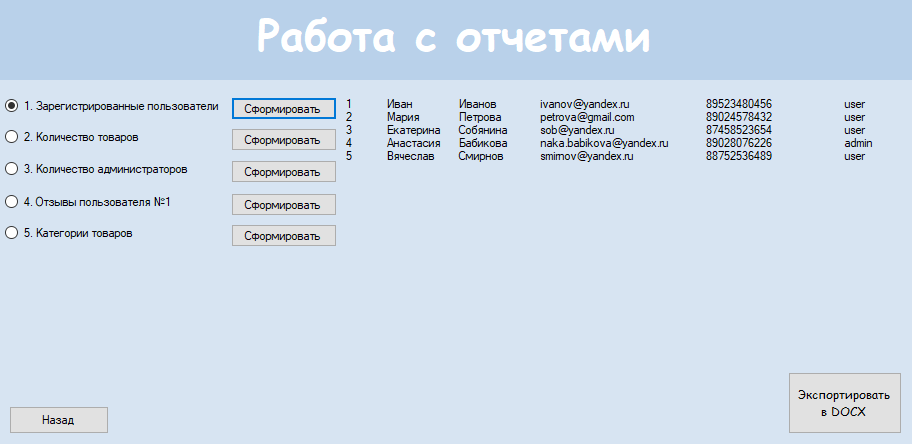


Рисунок 23 – Работа с отчетами.

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. 24.

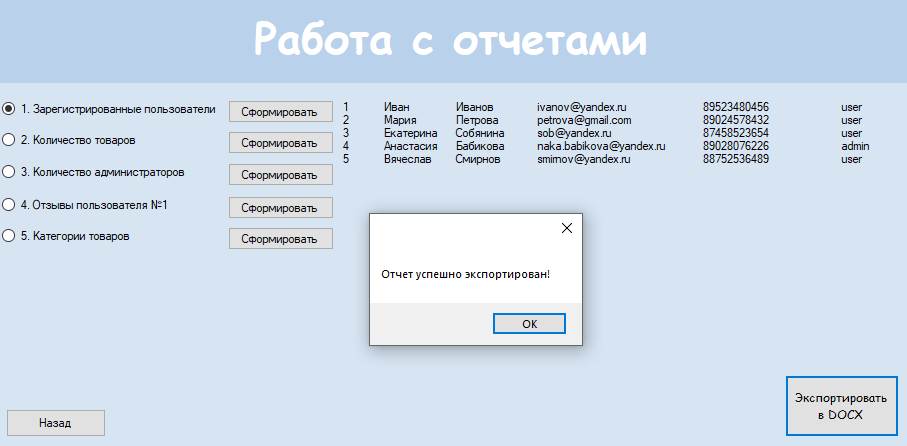


Рисунок 24 – Экспорт отчета.

Отчет в формате DOCX не удалось реализовать в заявленном формате. Вариант отчета представлен на рис.25.

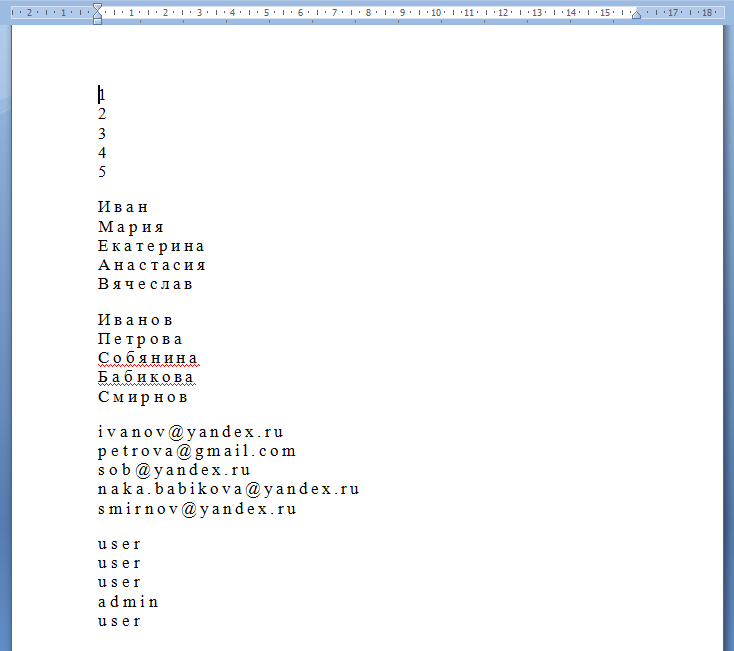


Рисунок 25 – Отчет в формате DOCX.

2. Добавление данных представлено на рис. 26.

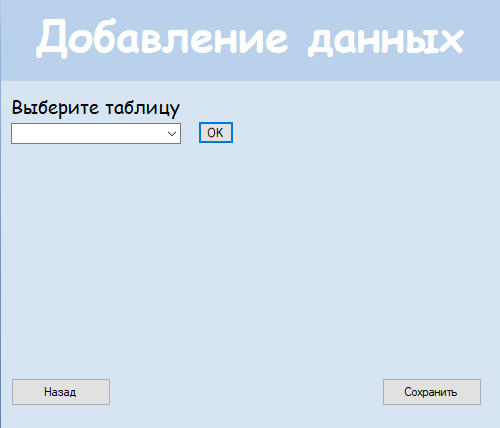


Рисунок 26 – Добавление данных.

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.27.

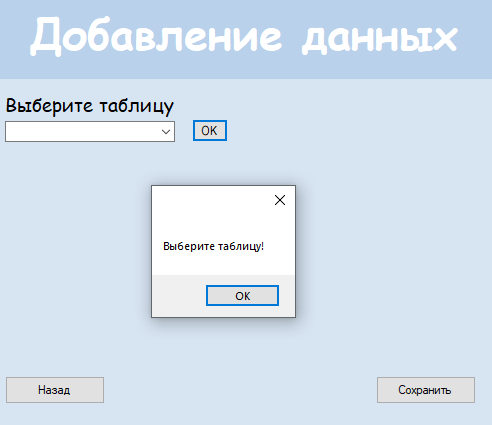


Рисунок 27 – Всплывающее окно с выбором таблицы.

При успешном выборе таблицы для каждой таблицы выводятся соответствующие поля, которые необходимо заполнить. Рис. 28.

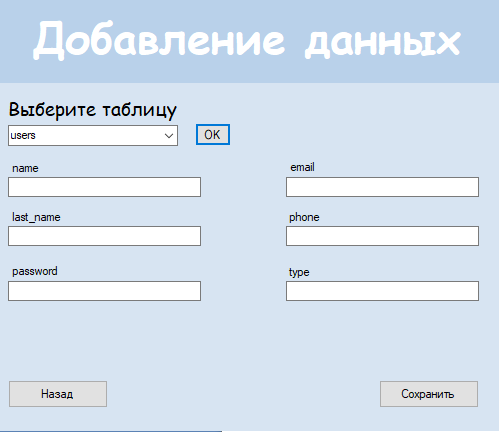


Рисунок 28 – Добавление данных в таблицу «users».

Для некоторых полей реализована проверка на пустоту, если пользователь не заполнил поле и нажал кнопку «Сохранить», то всплывает окно с предупреждением. Рис.29.

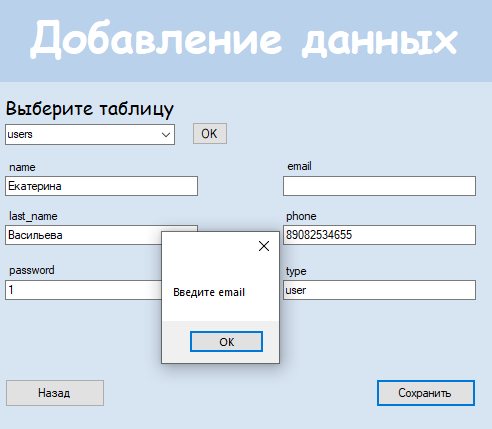


Рисунок 29 – Окно с предупреждением.

При успешном заполнение всех полей после нажатия на кнопку «Сохранить» получаем следующее сообщение. Рис. 30.

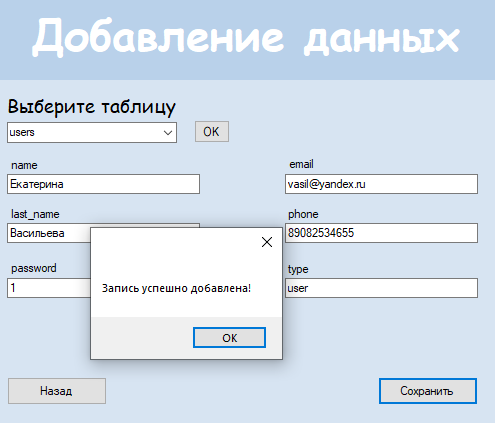


Рисунок 30 – Успешное добавление в базу.

3. Изменение данных представлено на рис. 31.

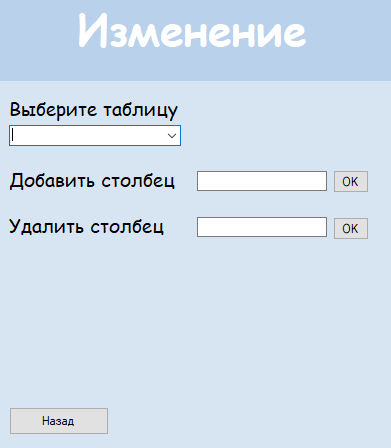


Рисунок 31 – Изменение данных.

4. Обновление данных представлено на рис.32.

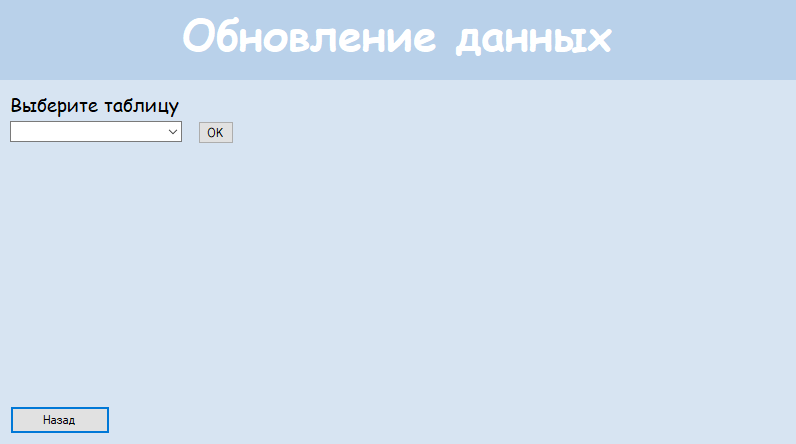


Рисунок 32 – Обновление данных.

Для обновления необходимо выбрать таблицу и нажать кнопку «ОК». Для каждой таблицы обновляемые поля отличаются. На рис. 33 обновление выполнялось для таблицы «orders».

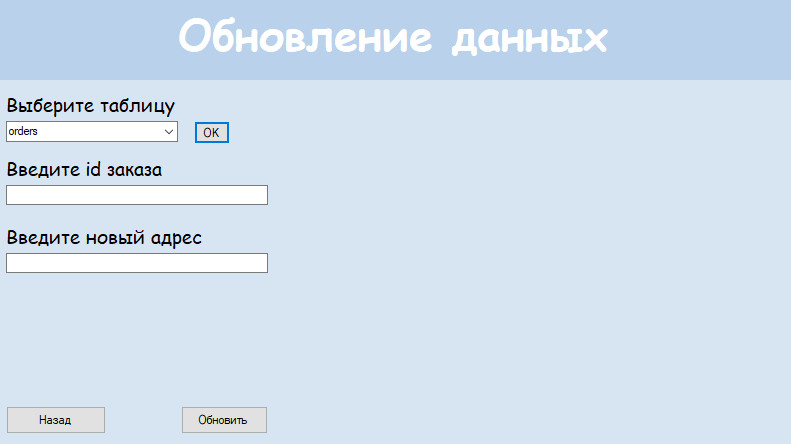


Рисунок 33 – Обновление таблицы «orders».

До обновления таблица в базе выглядела следующим образом. Рис. 34.

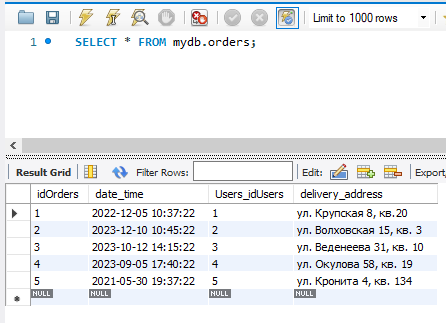


Рисунок 34 – Таблица «orders».

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.35.

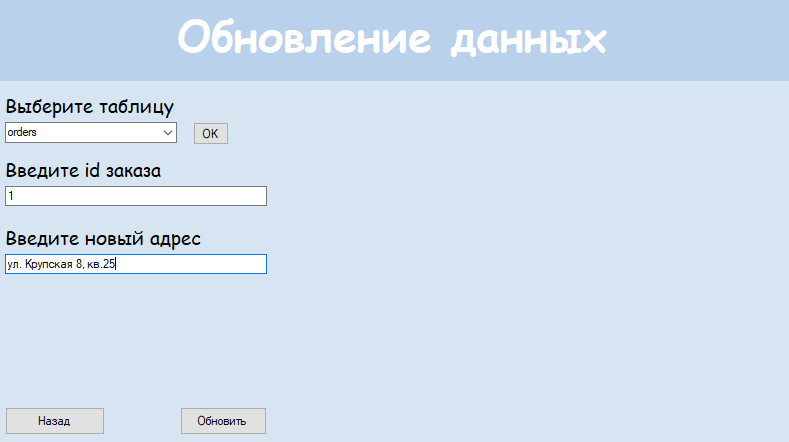


Рисунок 35 – Заполненные поля для обновления.

После успешного обновления было получено всплывающее окно представленное на рис. 36.

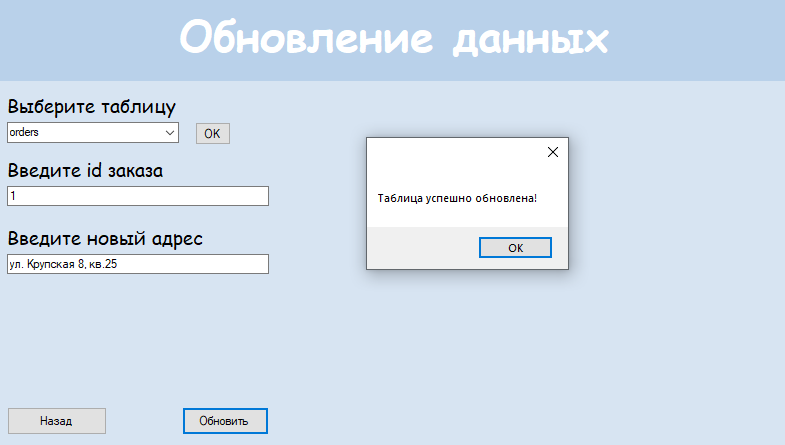


Рисунок 36 – Успешное обновление.

После этого база выглядит следующим образом. Рис. 37.

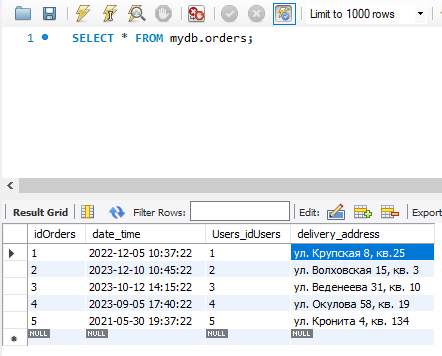


Рисунок 37 – Обновленная таблица «orders»

5. Удаление данных представлено на рис. 38.

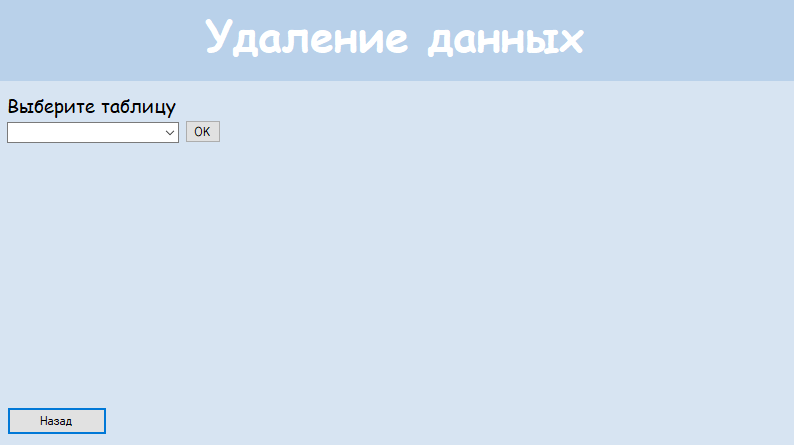


Рисунок 38 – Удаление данных.

Для того чтобы удалить данные необходимо выбрать таблицу и нажать кнопку «ОК». После этого появляется поле, пример представлен на рис.39.

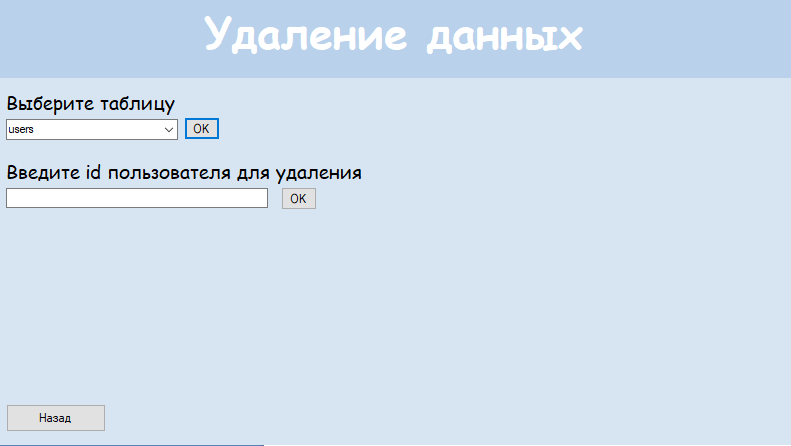


Рисунок 39 – Удаление записи по id.

При успешном удалении всплывает окно, представленное на рис.40.

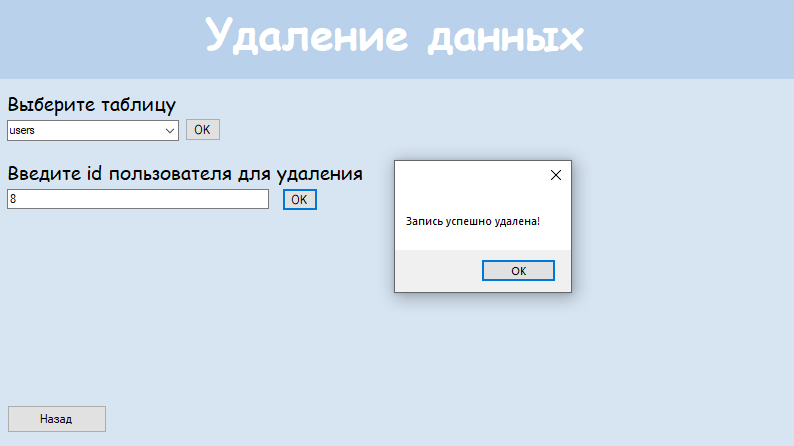


Рисунок 40 – Успешное удаление записи.

При успешной авторизации как пользователь отображается следующее окно, представленное на рис.41.

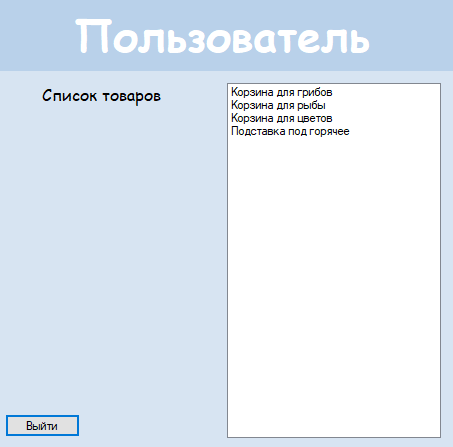


Рисунок 41 – Личный кабинет пользователя.

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

Вывод

### В ходе данного задания была реализована КИС на языке C#. А именно личные кабинеты пользователя и администратора, реализован отдельный класс для взаимодействия с базой данных (обновление, удаление, добавление, изменение данных), ответы в виде всплывающих окон на различные действия пользователей, формирование отчетов.

На данном этапе обновление реализовано у одной таблицы, некорректно выводятся отчеты в формат DOCX.

На следующем этапе планируется провести ручное тестирование для проверки заявленного функционала в техническом задании.

1. Тестирование

6.1 Реализация

6.1.1 Тестировщик и вид тестирования

### Тестирование проводилось в ручном режиме.

### Тестировщик – Бабикова Анастасия Сергеевна.

6.1.2 Тестирование основного функционала

*1. Авторизация*

|  |
| --- |
| **Тестируемая функция:** авторизация пользователя («user»);  **Входные данные:**  Текстовое поле № 1 = Иван  Текстовое поле № 2 = qwerty123  Текстовое поле № 3 = user  **Результат теста:**  Открытие личного кабинета пользователя  **Комментарии по результату теста:**  Успешная авторизация пользователя |

|  |
| --- |
| **Тестируемая функция:** авторизация пользователя («user») Рис.42.;  **Входные данные:**  Текстовое поле № 1 = Ваня  Текстовое поле № 2 = qwerty1234  Текстовое поле № 3 = user  **Результат теста:**  Всплывающее окно, что пользователь не найден  **Комментарии по результату теста:**  Авторизация не произошла |

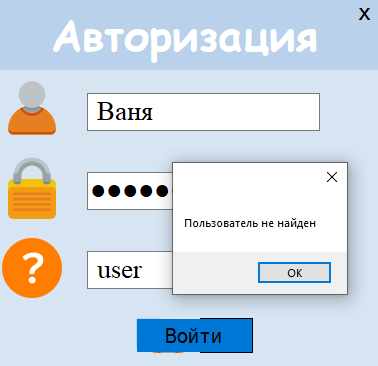


Рисунок 42 – Авторизация не существующего пользователя.

|  |
| --- |
| **Тестируемая функция:** авторизация пользователя («admin»)  **Входные данные:**  Текстовое поле № 1 = Анастасия  Текстовое поле № 2 = 123  Текстовое поле № 3 = admin  **Результат теста:**  Открытие личного кабинета администратора  **Комментарии по результату теста:**  Успешная авторизация пользователя |

*2. Функционал (вид) ИС под разными пользователями.*

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («user»)  **Входные данные: -**  **Результат теста**  Просмотр товаров, находящихся в базе данных  **Комментарии по результату теста**  Успешный просмотр товаров. |

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («user»)  **Входные данные:** нажатие на кнопку «выйти»  **Результат теста:**  Открытие окна с авторизацией |

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Работа с отчетами  **Входные данные:** нажатие на кнопку «сформировать»  **Результат теста:**  Вывод определенного отчета  **Комментарии по результату теста**  Справа от кнопки «сформировать» вывод отчета. Рис.43. |

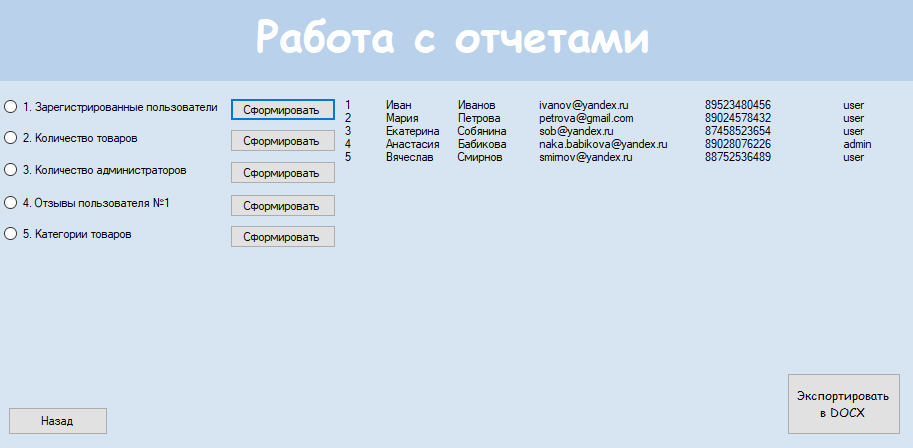


Рисунок 43 – Отчет о зарегистрированных пользователях.

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Работа с отчетами. Экспорт отчета в DOCX.  **Входные данные:** нажатие на кнопку с определенным отчетом, нажатие на кнопку «Экспортировать в DOCX»  **Результат теста:**  Всплывающее окно о том, что отчет был успешно экспортирован. Рис.44  **Комментарии по результату теста**  При просмотре отчета в формате DOCX выявлен баг системы, которая выводит столбцы отчета друг под другом. Рис.45. |

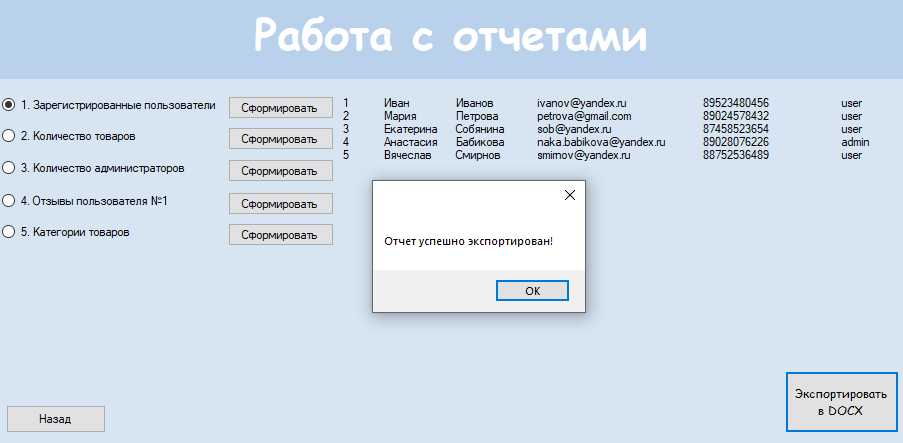


Рисунок 44 – Всплывающее окно «Отчет успешно экспортирован».

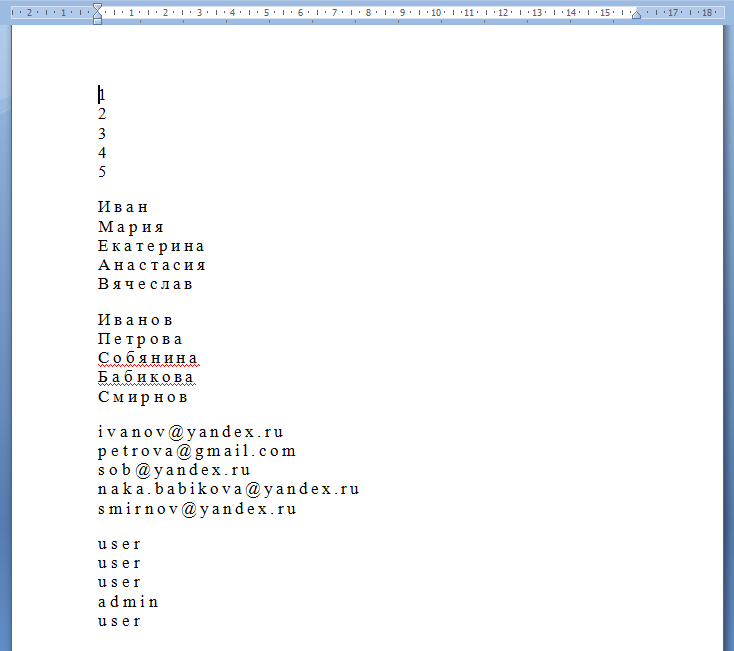


Рисунок 45 – Отчет в формате DOCX.

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Добавление данных  **Входные данные:** Таблица «categories», поле name = Дровница, нажатие на кнопку «сохранить». Рис.46.  **Результат теста:**  Всплывающее окно о том, что запись успешно добавлена.  **Комментарии по результату теста**  Запись успешно добавлена, что показывает база данных на рис.47. |

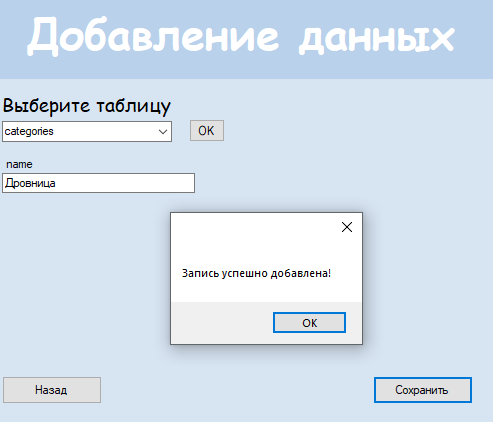


Рисунок 46 – Успешное добавление данных.

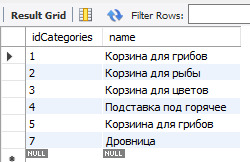


Рисунок 47 – Таблица «categories» в базе данных.

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Добавление данных  **Входные данные:** Таблица «categories», поле name = null, нажатие на кнопку «сохранить». Рис. 48.  **Результат теста:**  Всплывающее окно о том, что поле name было не заполнено.  **Комментарии по результату теста**  Запись не добавлена, так как пользователь не ввел название новой категории. |

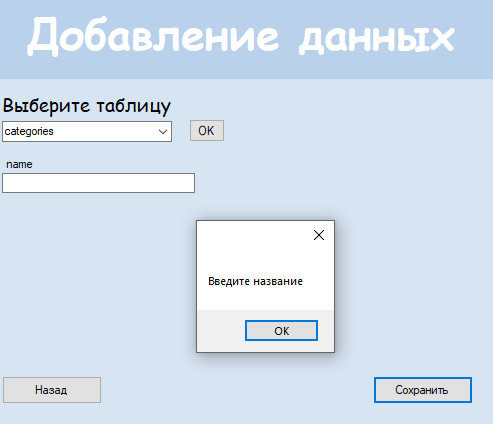


Рисунок 48 – Всплывающее окно «Введите название».

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Изменение данных  **Входные данные:** таблица «categories»  **Результат теста**  Всплывающее окно «Столбец успешно добавлен» Рис.49.  **Комментарии по результату теста**  Данная функция работает некорректно, так как она не считывает текстовое поле. На данном этапе при запросе к базе, поле установлено по умолчанию и может добавляться только один раз.  MySqlCommand command = new MySqlCommand("ALTER TABLE `categories` ADD COLUMN new NVARCHAR(50) NOT NULL DEFAULT 'Неизвестен'", db.getConnection());  Аналогичным образом работает удаление столбца. Рис. 50. |

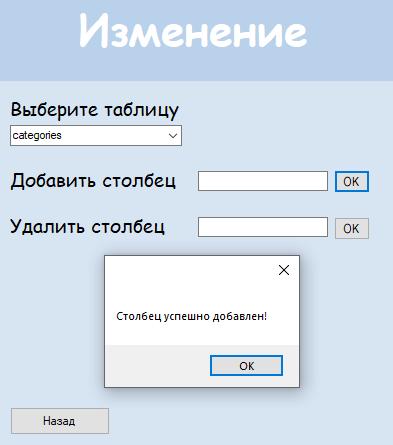


Рисунок 49 – Успешное изменение таблицы.

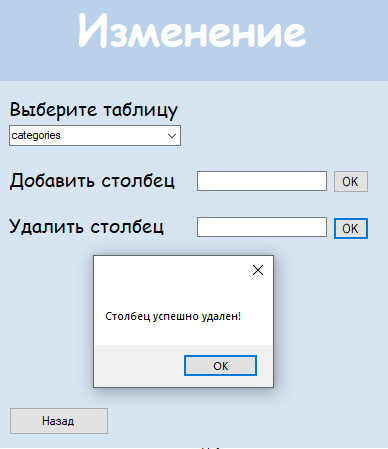


Рисунок 50 – Успешное изменение таблицы.

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Обновление данных  **Входные данные:** таблица «orders», id=1, новый адрес = ул. Крупская 8, кв.15. Нажатие на кнопку «Обновить».  **Результат теста**  Всплывающее окно о том, что таблица успешно обновлена. Рис.51.  **Комментарии по результату теста**  Таблица успешно обновлена, что подтверждает запрос к базе данных. Рис. 52-53. |

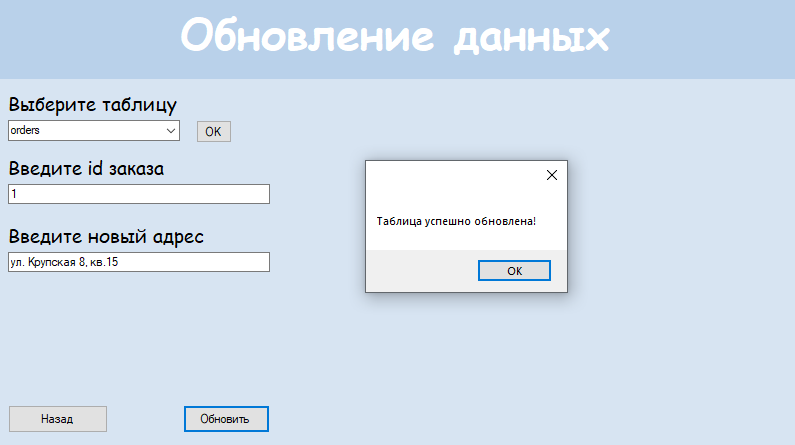


Рисунок 51 – Успешное обновление таблицы.

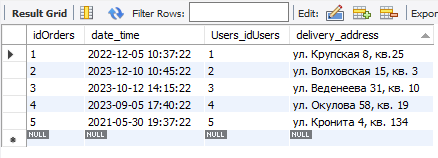


Рисунок 52 – Таблица до обновления.

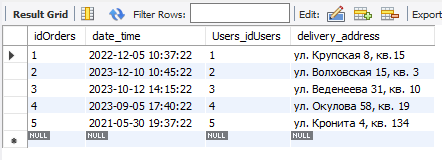


Рисунок 53 – Таблица после обновления.

|  |
| --- |
| **Тестируемая функция:** личный кабинет пользователя («admin»). Удаление данных.  **Входные данные: таблица «categories», id=7**  **Результат теста**  Всплывающее окно об успешном удалении записи. Рис.54.  **Комментарии по результату теста**  Запись успешно удалена, что подтверждает запрос к базе данных. Рис. 55-56. |

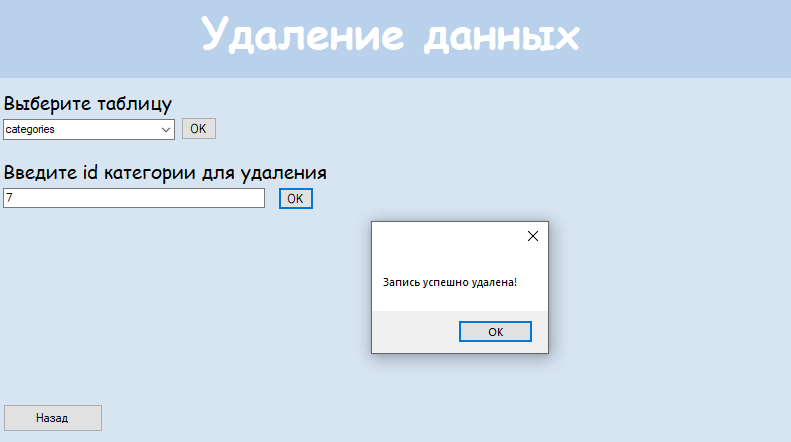


Рисунок 54 – Успешное удаление.

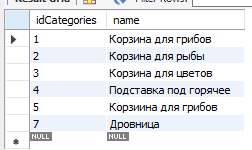


Рисунок 55 – Таблица до удаления.

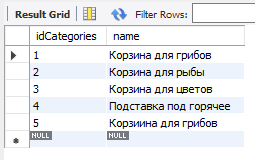


Рисунок 56 – Таблица после удаления.

Вывод

### В ходе данного задания было проведено ручное тестирование основного функционала системы, которое показывает, что система не всегда работает так, как заявлено в техническом задании. В последующем планируется исправить найденные недостатки и выполнить повторное тестирование.

*Приложение 1*

### **Авторизация**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KIS

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.passField.AutoSize = false;

this.passField.Size = new Size(this.passField.Size.Width,38);

}

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

{

Application.Exit();

}

private void closeButton\_MouseEnter(object sender, EventArgs e)

{

closeButton.ForeColor = Color.Red;

}

private void closeButton\_MouseLeave(object sender, EventArgs e)

{

closeButton.ForeColor = Color.Black;

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if(e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

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");

}

}

}

}

### **Личный кабинет администратора**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace KIS

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

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

{

Form1 form1 = new Form1();

form1.Show();

this.Hide();

}

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

{

this.Close();

}

private void closeButton\_MouseEnter(object sender, EventArgs e)

{

closeButton.ForeColor = Color.Red;

}

private void closeButton\_MouseLeave(object sender, EventArgs e)

{

closeButton.ForeColor = Color.Black;

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

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

{

Reports report = new Reports();

report.Show();

this.Hide();

}

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

{

AddData add = new AddData();

add.Show();

this.Hide();

}

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

{

ChangeData change = new ChangeData();

change.Show();

this.Hide();

}

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

{

UpdateData update = new UpdateData();

update.Show();

this.Hide();

}

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

{

DeleteData delete = new DeleteData();

delete.Show();

this.Hide();

}

}

}

### **Личный кабинет пользователя**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KIS

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

listBox();

}

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

{

this.Close();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

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

{

Form1 form1 = new Form1();

form1.Show();

this.Hide();

}

private void listBox()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("select title from products", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

foreach (DataRow row in table.Rows)

{

string tablename = (string)row[0];

listBox1.Items.Add(tablename);

}

}

}

}

### **Добавление данных**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KIS

{

public partial class AddData : Form

{

public AddData()

{

InitializeComponent();

comboBox();

}

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

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void comboBox()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SHOW TABLES", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

foreach (DataRow row in table.Rows)

{

string tablename = (string)row[0];

comboBoxTables.Items.Add(tablename);

}

}

//OkButton

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

{

var table = comboBoxTables.Text;

if(table == "")

{

MessageBox.Show("Выберите таблицу!");

}

if(table == "categories")

{

label1.Visible = true;

label3.Visible = false;

label4.Visible = false;

label5.Visible = false;

label6.Visible = false;

label7.Visible = false;

textBox1.Visible = true;

textBox2.Visible = false;

textBox3.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

textBox6.Visible = false;

DB db = new DB();

DataTable tableFields = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("show fields from categories", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(tableFields);

foreach (DataRow row in tableFields.Rows)

{

string field1 = (string)row[0];

label1.Text = field1;

}

}

if (table == "orders")

{

label1.Visible = true;

label1.Text = "date\_time";

label3.Visible = true;

label3.Text = "Users\_idUsers";

label4.Visible = true;

label4.Text = "delivery\_address";

label5.Visible = false;

label6.Visible = false;

label7.Visible = false;

textBox1.Visible = true;

textBox2.Visible = true;

textBox3.Visible = true;

textBox4.Visible = false;

textBox5.Visible = false;

textBox6.Visible = false;

}

if(table == "orders\_has\_products")

{

label1.Visible = true;

label1.Text = "Products\_idProducts";

label3.Visible = true;

label3.Text = "Products\_Categories\_idCategories";

label4.Visible = true;

label4.Text = "Orders\_idOrders";

label5.Visible = true;

label5.Text = "count";

label6.Visible = false;

label7.Visible = false;

textBox1.Visible = true;

textBox2.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

textBox5.Visible = false;

textBox6.Visible = false;

}

if(table == "products")

{

label1.Visible = true;

label1.Text = "title";

label3.Visible = true;

label3.Text = "description";

label4.Visible = true;

label4.Text = "price";

label5.Visible = true;

label5.Text = "image";

label6.Visible = true;

label6.Text = "Categories\_idCategories";

label7.Visible = true;

label7.Text = "Recommendations\_idRecommendations";

textBox1.Visible = true;

textBox2.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

textBox5.Visible = true;

textBox6.Visible = true;

}

if(table == "recommendations")

{

label1.Visible = true;

label1.Text = "text";

label3.Visible = false;

label4.Visible = false;

label5.Visible = false;

label6.Visible = false;

label7.Visible = false;

textBox1.Visible = true;

textBox2.Visible = false;

textBox3.Visible = false;

textBox4.Visible = false;

textBox5.Visible = false;

textBox6.Visible = false;

}

if(table == "reviews")

{

label1.Visible = true;

label1.Text = "text";

label3.Visible = true;

label3.Text = "date";

label4.Visible = true;

label4.Text = "Products\_idProducts";

label5.Visible = true;

label5.Text = "Products\_Categories\_idCategories";

label6.Visible = true;

label6.Text = "Users\_idUsers";

label7.Visible = false;

textBox1.Visible = true;

textBox2.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

textBox5.Visible = true;

textBox6.Visible = false;

}

if(table == "users")

{

label1.Visible = true;

label1.Text = "name";

label3.Visible = true;

label3.Text = "last\_name";

label4.Visible = true;

label4.Text = "password";

label5.Visible = true;

label5.Text = "email";

label6.Visible = true;

label6.Text = "phone";

label7.Visible = true;

label7.Text = "type";

textBox1.Visible = true;

textBox2.Visible = true;

textBox3.Visible = true;

textBox4.Visible = true;

textBox5.Visible = true;

textBox6.Visible = true;

}

}

//Сохранить

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();

}

if(table == "orders")

{

if (textBox3.Text == "")

{

MessageBox.Show("Введите адрес");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `orders` (`date\_time`,`Users\_idUsers`,`delivery\_address`) VALUES (@datatime, @idUsers, @address)", db.getConnection());

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

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

command.Parameters.Add("@address", MySqlDbType.VarChar).Value = textBox3.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if(table == "orders\_has\_products")

{

if (textBox4.Text == "")

{

MessageBox.Show("Введите количество");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `orders\_has\_products` (`Products\_idProducts`,`Products\_Categories\_idCategories`,`Orders\_idOrders`,`count`) VALUES (@idProducts, @idCategories, @idOrders, @count)", db.getConnection());

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

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

command.Parameters.Add("@idOrders", MySqlDbType.VarChar).Value = textBox3.Text;

command.Parameters.Add("@count", MySqlDbType.VarChar).Value = textBox4.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if(table == "products")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите название");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `products` (`title`,`description`,`price`,`image`, `Categories\_idCategories`, `Recommendations\_idRecommendations`) VALUES (@title, @description, @price, @image, @idCategories, @idRecommendations)", db.getConnection());

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

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

command.Parameters.Add("@price", MySqlDbType.VarChar).Value = textBox3.Text;

command.Parameters.Add("@image", MySqlDbType.VarChar).Value = textBox4.Text;

command.Parameters.Add("@idCategories", MySqlDbType.VarChar).Value = textBox5.Text;

command.Parameters.Add("@idRecommendations", MySqlDbType.VarChar).Value = textBox6.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if (table == "recommendations")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите текст");

return;

}

DB db = new DB();

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

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if(table == "reviews")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите текст");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `reviews` (`text`,`date`,`Products\_idProducts`,`Products\_Categories\_idCategories`, `Users\_idUsers`) VALUES (@text, @date, @idProducts, @idCategories, @idUsers)", db.getConnection());

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

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

command.Parameters.Add("@idProducts", MySqlDbType.VarChar).Value = textBox3.Text;

command.Parameters.Add("@idCategories", MySqlDbType.VarChar).Value = textBox4.Text;

command.Parameters.Add("@idUsers", MySqlDbType.VarChar).Value = textBox5.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if(table == "users")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите имя");

return;

}

if (textBox2.Text == "")

{

MessageBox.Show("Введите фамилию");

return;

}

if (textBox3.Text == "")

{

MessageBox.Show("Введите пароль");

return;

}

if (textBox4.Text == "")

{

MessageBox.Show("Введите email");

return;

}

if (textBox5.Text == "")

{

MessageBox.Show("Введите телефон");

return;

}

if (textBox6.Text == "")

{

MessageBox.Show("Введите тип пользователя");

return;

}

if (isUserExist())

{

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`name`,`last\_name`,`password`,`email`, `phone`, `type`) VALUES (@name, @last\_name, @password, @email, @phone, @type)", db.getConnection());

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

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

command.Parameters.Add("@password", MySqlDbType.VarChar).Value = textBox3.Text;

command.Parameters.Add("@email", MySqlDbType.VarChar).Value = textBox4.Text;

command.Parameters.Add("@phone", MySqlDbType.VarChar).Value = textBox5.Text;

command.Parameters.Add("@type", MySqlDbType.VarChar).Value = textBox6.Text;

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

}

public Boolean isCategoriesExist()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `categories` WHERE `name` = @name ", db.getConnection());

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

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Данная категория уже существует!");

return true;

}

else

{

return false;

}

}

public Boolean isUserExist()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `users` WHERE `email` = @email ", db.getConnection());

command.Parameters.Add("@email", MySqlDbType.VarChar).Value = textBox4.Text;

adapter.SelectCommand = command;

adapter.Fill(table);

if (table.Rows.Count > 0)

{

MessageBox.Show("Пользователь с введенным email уже существует!");

return true;

}

else

{

return false;

}

}

}

}

### **Изменение данных**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KIS

{

public partial class ChangeData : Form

{

public ChangeData()

{

InitializeComponent();

comboBox();

}

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

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void comboBox()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SHOW TABLES", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

foreach (DataRow row in table.Rows)

{

string tablename = (string)row[0];

comboBoxTables.Items.Add(tablename);

}

}

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();

}

}

private void deleteColumnButton\_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` DROP COLUMN new", db.getConnection());

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

db.openConnection();

command.ExecuteNonQuery();

if (command\_1!= command\_2)

{

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

}

else

{

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

}

db.closeConnection();

}

}

}

}

### **Удаление данных**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KIS

{

public partial class DeleteData : Form

{

public DeleteData()

{

InitializeComponent();

comboBox();

}

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

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void comboBox()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SHOW TABLES", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

foreach (DataRow row in table.Rows)

{

string tablename = (string)row[0];

comboBoxTables.Items.Add(tablename);

}

}

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

{

var table = comboBoxTables.Text;

if (table == "")

{

MessageBox.Show("Выберите таблицу!");

}

if (table == "categories")

{

label1.Visible = true;

label1.Text = "Введите id категории для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "orders")

{

label1.Visible = true;

label1.Text = "Введите id заказа для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "orders\_has\_products")

{

label1.Visible = true;

label1.Text = "Введите id заказа для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "products")

{

label1.Visible = true;

label1.Text = "Введите id товара для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "recommendations")

{

label1.Visible = true;

label1.Text = "Введите id рекомендации для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "reviews")

{

label1.Visible = true;

label1.Text = "Введите id отзыва для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "users")

{

label1.Visible = true;

label1.Text = "Введите id пользователя для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

}

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();

}

if (table == "orders")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

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

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if (table == "orders\_has\_products")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

MySqlCommand command = new MySqlCommand("DELETE FROM `orders\_has\_products` WHERE `Products\_idProducts` = @Products\_idProducts", db.getConnection());

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if (table == "products")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

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

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if (table == "recommendations")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

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

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if (table == "reviews")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

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

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

if (table == "users")

{

if (textBox1.Text == "")

{

MessageBox.Show("Введите id");

return;

}

DB db = new DB();

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

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

db.openConnection();

if (command.ExecuteNonQuery() == 1)

{

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

}

else

{

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

}

db.closeConnection();

}

}

}

}

### **Обновление данных**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KIS

{

public partial class UpdateData : Form

{

public UpdateData()

{

InitializeComponent();

comboBox();

}

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

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

private void comboBox()

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SHOW TABLES", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

foreach (DataRow row in table.Rows)

{

string tablename = (string)row[0];

comboBoxTables.Items.Add(tablename);

}

}

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

{

var table = comboBoxTables.Text;

if (table == "")

{

MessageBox.Show("Выберите таблицу!");

}

if (table == "orders")

{

label1.Visible = true;

label1.Text = "Введите id заказа";

label3.Visible = true;

label3.Text = "Введите новый адрес";

textBox1.Visible = true;

textBox2.Visible = true;

button2.Visible = true;

}

if (table == "orders\_has\_products")

{

label1.Visible = true;

label1.Text = "Введите id заказа для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "products")

{

label1.Visible = true;

label1.Text = "Введите id товара для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "recommendations")

{

label1.Visible = true;

label1.Text = "Введите id рекомендации для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "reviews")

{

label1.Visible = true;

label1.Text = "Введите id отзыва для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

if (table == "users")

{

label1.Visible = true;

label1.Text = "Введите id пользователя для удаления";

textBox1.Visible = true;

button2.Visible = true;

}

}

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();

}

}

}

}

### **Формирование отчетов**

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

using Xceed.Words.NET;

using Xceed.Document.NET;

namespace KIS

{

public partial class Reports : Form

{

public Reports()

{

InitializeComponent();

}

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

{

Form2 form2 = new Form2();

form2.Show();

this.Hide();

}

Point lastPoint;

private void panel1\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void panel1\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

//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;

}

}

}

//Количество товаров

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

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

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

adapter.SelectCommand = command;

adapter.Fill(table);

idLabel.Visible = true;

nameLabel.Visible = false;

lastNameLabel.Visible = false;

emailLabel.Visible = false;

phoneLabel.Visible = false;

typeLabel.Visible = false;

idLabel.Text = Convert.ToString(table.Rows.Count) ;

}

//Количество администраторов

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

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("select \* from users WHERE `type`='admin'", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

idLabel.Visible = true;

nameLabel.Visible = false;

lastNameLabel.Visible = false;

emailLabel.Visible = false;

phoneLabel.Visible = false;

typeLabel.Visible = false;

idLabel.Text = Convert.ToString(table.Rows.Count);

}

//Отзывы пользователя с id=1

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

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("select \* from reviews WHERE `Users\_idUsers`=1", db.getConnection());

adapter.SelectCommand = command;

adapter.Fill(table);

idLabel.Visible = true;

nameLabel.Visible = true;

lastNameLabel.Visible = true;

emailLabel.Visible = false;

phoneLabel.Visible = false;

typeLabel.Visible = false;

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];

nameLabel.Text += Convert.ToString(name) + Environment.NewLine;

DateTime lastName = (DateTime)row[2];

lastNameLabel.Text += Convert.ToString(lastName) + Environment.NewLine;

}

else

{

indicator += 1;

int id = (int)row[0];

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

string name = (string)row[1];

nameLabel.Text = Convert.ToString(name) + Environment.NewLine;

DateTime lastName = (DateTime)row[2];

lastNameLabel.Text = Convert.ToString(lastName) + Environment.NewLine;

}

}

}

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

{

DB db = new DB();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

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

adapter.SelectCommand = command;

adapter.Fill(table);

idLabel.Visible = true;

nameLabel.Visible = true;

lastNameLabel.Visible = false;

emailLabel.Visible = false;

phoneLabel.Visible = false;

typeLabel.Visible = false;

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];

nameLabel.Text += Convert.ToString(name) + Environment.NewLine;

}

else

{

indicator += 1;

int id = (int)row[0];

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

string name = (string)row[1];

nameLabel.Text = Convert.ToString(name) + 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!");

}

//2.Количество товаров

if (radioButton2.Checked)

{

// путь к документу

string pathDocument = AppDomain.CurrentDomain.BaseDirectory + "report2.docx";

// создаём документ

DocX document = DocX.Create(pathDocument);

document.InsertParagraph(idLabel.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!");

}

//3.Количество администраторов

if (radioButton3.Checked)

{

// путь к документу

string pathDocument = AppDomain.CurrentDomain.BaseDirectory + "report3.docx";

// создаём документ

DocX document = DocX.Create(pathDocument);

document.InsertParagraph(idLabel.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.Отзывы пользователя №1

if (radioButton4.Checked)

{

// путь к документу

string pathDocument = AppDomain.CurrentDomain.BaseDirectory + "report4.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);

// сохраняем документ

document.Save();

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

}

//5.Категории товаров

if (radioButton5.Checked)

{

// путь к документу

string pathDocument = AppDomain.CurrentDomain.BaseDirectory + "report5.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);

// сохраняем документ

document.Save();

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

}

}

}

}