

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Заведующий кафедрой | | «ПОВТиАС» |
|  | | В.В. Долгов |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г. | |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе по дисциплине (модулю) Базы данных

(наименование учебной дисциплины (модуля))

на тему: Проектирование и реализация базы данных интернет мазагина хлебной продукции

Автор проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.Д.Булыгин

(подпись) И.О.Ф.

Направление/специальность, профиль/специализация:

09.03.04 Программная инженерия

код направления наименование направления (специальности)

наименование профиля (специализации)

Обозначение курсовой работы 09.03.04.720000.000 КР Группа ВПР32

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ст. преподаватель С.П. Новиков\_\_\_\_\_\_\_

подпись (должность, И.О.Ф.)

Проект (работа) защищен(а) \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка подпись

Ростов-на-Дону

20\_\_



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Программное обеспечение вычислительной техники и автоматизированных систем

(наименование кафедры)

|  |  |  |
| --- | --- | --- |
|  | | |
| Заведующий кафедрой | | «ПОВТиАС» |
|  | | В.В. Долгов |
| (подпись) | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_г. | |

**ЗАДАНИЕ**

на курсовой проект (работу)

Студент Р.Д. Булыгин Код 09.03.04 Группа ВПР32

Тема Проектирование и реализация базы данных интернет магазина хлебной продукции

Срок представления проекта (работы) к защите «\_\_» \_\_\_\_\_\_\_\_\_ 20\_\_ г.

Исходные данные для курсовой работы

1. Задание на выполнение курсовой работы

2. Лекционный материал по дисциплине «Базы данных»; Ульман Дж. Основы систем баз данных. -М., Финансы и статистика, 2014. —438 с. Карпова Т.С. Базы данных: модели, разработка, реализация –СПб: Питер, 2013. –304 с. Коннолли Томас, БеггКарелин. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание: Пер. с англ. —М.: Издательский дом "Вильямс", 2011. —1440 с. Советов, Б.Я. Базы данных: теория и практика: Учебник для бакалавров / Б.Я. Советов, В.В. Цехановский, В.Д. Чертовской. —М.: Юрайт, 2013. —463 c. Кузин, А.В. Базы данных: Учебное пособие для студентов высших учебных заведений / А.В. Кузин, С.В. Левонисова —М.: ИЦ Академия, 2012. —320 c

Содержание пояснительной записки

ВВЕДЕНИЕ:

В данном разделе содержится: описание ключевых понятий, которые фигурируют в контексте баз данных, постановка задачи, выполнение которой сводится к разработке базы данных интернет магазина хлебной продукци, краткие описания каждой главы.

Наименование и содержание разделов:

1. Анализ предметной области и проектирование структуры базы данных. Этот раздел посвящен описанию выбранной предметной области. Приведено подробное описание видения базы данных конечными пользователями, описаны их основные функции. Описана модель уровня представлений.

2. Физическая реализация разработанной модели. В данном разделе описаны основные функции, используемые для работы с базой данных, спроектированы таблицы в системе управления базой данных.

3. Разработка клиентского интерфейса для работы с базой данных. Данный раздел содержит описание классов, необходимых для создания приложения, их методов. Произведено тестирование приложения.

ЗАКЛЮЧЕНИЕ:

В заключении курсовой работы отображены результаты решения поставленной задачи – была разработана база данных интренет магазина хлебной продукции, а также программное средство с графическим интерфейсом в виде оконного приложения, соответствующего требованиям, предъявляемыми конечными пользователями.

Руководитель проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_С.П. Новиков\_\_

подпись, дата И.О.Ф.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Р.Д. Булыгин\_\_

подпись, дата И.О.Ф.

**СОДЕРЖАНИЕ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

09.03.04.720000.000 КР

Разраб.

Булугын Р.Д.

Провер.

Новиков С.П.

Проектирование и реализация базы данных интернет магазина хлебной продукции

Пояснительная записка

Лит.

Листов

119

[ВВЕДЕНИЕ](#_Toc63342421)

[1. Анализ предметной области и проектирование структуры базы данных (БД) 9](#_Toc63342422)

[1.1 Формирование уровня представлений БД интернет магазина хлебной продукции 9](#_Toc63342423)

[1.1.1 Описание представлений пользователей и выделение основных сущностей 9](#_Toc63342424)

[1.1.2 Описание атрибутов сущностей 12](#_Toc63342425)

[1.1.3 Описание доменов атрибутов 13](#_Toc63342426)

[1.1.4 Создание схемы в виде ER-диаграммы 17](#_Toc63342427)

[1.1.5 Описание связей между объектами 20](#_Toc63342428)

[1.2 Построение концептуального уровня БД интернет магазина хлебной продукции 21](#_Toc63342429)

[1.2.1 Описание функциональных зависимостей, имеющих место в предметной области 21](#_Toc63342430)

[1.2.2 Нормализация отношений 22](#_Toc63342431)

[1.2.2.1 Нормализация до первой нормальной формы (1НФ) 22](#_Toc63342432)

[1.2.2.2 Обоснование выбора первичных ключей и приведение ко второй нормальной форме (2НФ) 23](#_Toc63342433)

[1.2.2.3 Нормализация до третьей нормальной формы (3НФ). Выводы](#_Toc63342434) 24

[1.2.3 Логическое (даталогическое) проектирование базы данных 25](#_Toc63342435)

[2. Физическая реализация модели БД интернет магазина 26](#_Toc63342436)

[2.1 Обоснование выбора целевой системы управления базами данных (СУБД) 26](#_Toc63342437)

[2.2 Создание таблиц данных в среде целевой СУБД 27](#_Toc63342438)

[3. Разработка клиентского интерфейса для работы с базой данных 31](#_Toc63342439)

[3.1 Выбор среды для разработки приложения 31](#_Toc63342440)

[3.2 Описание основных классов и методов интерфейса 32](#_Toc63342441)

[3.3 Пример работы графического интерфейса 36](#_Toc63342442)

[ЗАКЛЮЧЕНИЕ](#_Toc63342443) 47

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc63342444)

[ПРИЛОЖЕНИЕ А – ER-диаграмма 48](#_Toc63342445)

[ПРИЛОЖЕНИЕ Б – Даталогическая схема 50](#_Toc63342446)

[ПРИЛОЖЕНИЕ В – UML-диаграмма 51](#_Toc63342447)

[ПРИЛОЖЕНИЕ Г – Код базы данных в MSQL 52](#_Toc63342448)

[ПРИЛОЖЕНИЕ Д – Код клиентского интерфейса 53](#_Toc63342449)

*ДГТУ*

*Кафедра «ПОВТиАС»*

[ПРИЛОЖЕНИЕ Е – Отзыв руководителя 119](#_Toc63342450)

## **ВВЕДЕНИЕ**

Современный мир невозможно представить без использования компьютерных технологий. Каждую компьютеризированную технологию можно разложить на две компоненты: алгоритмы и данные. Сегодня ежесекундно генерируются гигантские объемы данных, работать с подобными объемами данных без систем хранения и структурирования не предоставляется возможным. Для структурированной работы с полученной информацией необходимы инструменты, организующие работу, взаимодействие, хранение и доступ к данным. Таким инструментом и являются базы данных.

База данных — это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств [1]. Или, более узко: БД – именованная совокупность взаимосвязанных данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений. БД состоит из множества связанных файлов и используется в целях отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Данные запоминаются так, чтобы они были независимы от программ, использующих эти данные. Для добавления новых или модификации существующих данных, а также для поиска данных в БД применяется система управления базами данных (СУБД).

СУБД – это программное обеспечение, предназначенное для создания, ведения и совместного использования БД многими пользователями. Простым языком, система управления базами данных выступает в качестве посредника между базой данных и ее пользователями [1]. СУБД осуществляют ввод, проверку, систематизацию, поиск и обработку данных, распечатку их в виде отчётов. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Первичным источником связанных данных при проектировании баз данных любых разновидностей является соответствующая предметная область, которая будет рассматриваться как совокупность знаний и данных об объектах и процессах, подлежащих проектированию и хранению в БД.

Целью данной работы является разработка базы данных и графического клиентского интерфейса для интернет-магазина хлебной продукции.

В первой главе выполнен анализ предметной области, выделены основные сущности и их атрибуты, описана база данных на внешнем и концептуальном уровнях.

Во второй главе описана реализация базы данных интернет-магазина хлебной продукции на физическом уровне.

В третьей главе описана реализация клиентского интерфейса и рассмотрена работа приложения на тестовом примере.

# **1. Анализ предметной области и проектирование структуры базы данных (БД)**

Процесс проектирования базы данных принято разбивать на несколько уровней абстракций.

## **1.1 Формирование уровня представлений БД интернет-магазина хлебной продукции**

### **1.1.1 Описание представлений пользователей и выделение основных сущностей**

**Предметной областью** данной курсовой работы является сфера торговли, связанных с быстрым оформлением заказа в магазине.

Пользователю достаточно просто сформировать свой заказ в онлайн магазине, и в дальнейшем просто забрать его в ближайшем к нему магазине.

Интернет-магазин — сайт или приложение, торгующий товарами в интернете. Позволяет пользователям сформировать заказ на покупку, выбрать способ оплаты и доставки заказа в сети Интернет.

* задачами онлайн магазина являются:
* увеличение продаж за счет притока трафика в компанию;
* упрощение процесса покупки и заказа онлайн;
* сокращение расходов на рекламу и продвижение;
* реализация дополнительных услуг;
* информирование потребителей.

Сперва рассмотрим предметную область на уровне представления.

База данных онлайн магазина необходима прежде всего для хранения информации о пользователях, доступных товарах, выполненных операциях, выполненных заказах.

Так как по принципам онлайн магазина покупателем является сам конечный пользователь, информация о пользователе должна содержать следующие данные: имя, фамилию, а также необходимо иметь информацию о его заказах. Данные в заказе заполняет конечный пользователь при его создании путем заполнения простой формы.

Данные о продукте должны содержать его название, вес, цена, описание, отзывы пользователей о продукте (комментарии) и обзорах на них (система рецензий). Так как конечный клиент магазина является покупателем, отзыв или комментарий к продукты заполняются также конечным пользователем при его добавлении путем заполнения данных в форме.

Для удобства хранения и поиска продукта была введена система категорий и подкатегорий. В категориях хранится список подкатегорий, в подкатегориях хранится список товаров, присущих ему. Список товаров определяется администратором БД, путем указания товару номера подкатегории, в которой он находится.

Данные о заказе пользователя должны содержать имя, фамилию, уникальный номер пользователя, список товаров, который выбрал пользователь, окончательную сумму заказа. Некоторые данные нужные для заказа, такие как: список товаров пользователь выбирает путем заполнения в форме.

Так же некоторым пользователям может потребоваться оставить своё мнение о продукте со всеми его минусами и плюсами, для этого мы вводим понятие - отзыв. Его пользователь может оставить выбрав продукт и написав сам отзыв в специальной форме.

На основе предметной области можно выделить следующих конечных пользователей:

* конечный пользователь;
* модератор;
* администратор.

Процесс заполнения данных начинается с регистрации нового конечного пользователя.

**Конечный пользователь:** заполняет данные о себе через интерфейс программы, при помощи простой регистрационной формы.

Далее конечный пользователь может создать новый заказ путем заполнения списка продуктов, написать свой комментарий или отзыв к любому продукту.

**Модератор:** является пользователям, имеющим доступ к просмотру базы данных. Принимает решение о допустимости публикуемого отзыва или комментария. Таким образом для покрытия целей модератора система должна хранить информацию:

* о клиенте;
* о заказах;
* о комментариях
* о отзывах.

**Администратор**: является суперпользователем, то есть он имеет полный доступ ко всей информации, хранящейся в базе данных.

Итого выделено: «Пользователь», «Заказ», «Отзыв», «Комментарий», «Категория», «Подкатегория».

## **1.1.2 Описание атрибутов сущностей**

Ниже будут описаны необходимые сущности и атрибуты для создания БД интернет магазина.

**Объект «Пользователь»:**

TABLE `user` (  
 `email` VARCHAR(255) NOT NULL,  
 `password` VARCHAR(32) NOT NULL,  
 `create\_time` TIMESTAMP NULL DEFAULT CURRENT\_TIMESTAMP,  
 `firstname` VARCHAR(45) NOT NULL,  
 `lastname` VARCHAR(45) NOT NULL,  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `is\_stuff` TINYINT(1) NOT NULL

);

**Объект «Подкатегория»:**

TABLE `subcategory` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `name\_subcategory` VARCHAR(45) NULL,  
 `category\_id` INT NOT NULL

);

**Объект «Продукт»:**

TABLE `product` (

`id` INT NOT NULL AUTO\_INCREMENT,  
 `name` VARCHAR(45) NOT NULL,  
 `price` INT NOT NULL,  
 `description` TEXT(255) NOT NULL,  
 `shelf\_life` VARCHAR(45) NOT NULL,  
 `weight` DOUBLE NOT NULL,  
 `image` BLOB NULL,  
 `subcategory\_id` INT NOT NULL

);

**Объект «Заказ»:**

TABLE `transaction` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `number\_card` VARCHAR(45) NOT NULL,  
 `sum` DOUBLE NOT NULL,  
 `products\_list` LONGTEXT NOT NULL,  
 `user\_id` INT NOT NULL

);

**Объект «Администратор»:**

TABLE`admin` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `email` VARCHAR(45) NOT NULL,  
 `password` VARCHAR(45) NOT NULL,  
 `admin\_firstname` VARCHAR(45) NOT NULL,  
 `admin\_lastname` VARCHAR(45) NOT NULL

);

**Объект «Комментарий»:**

TABLE `comments` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `commentator\_firstname` VARCHAR(45) NULL,  
 `commentator\_lastname` VARCHAR(45) NULL,  
 `commentary` LONGTEXT NULL,  
 `user\_id` INT NOT NULL

);

**Объект «Отзыв»:**

TABLE `review` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `id\_user` VARCHAR(45) NULL,  
 `user\_firstname` VARCHAR(45) NULL,  
 `commentary` LONGTEXT NULL,  
 `is\_user` TINYINT NULL,  
 `user\_id` INT NOT NULL

);

**Объект «Категория»:**

TABLE `category` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `name\_category` VARCHAR(45) NULL

);

**Отношение «Продукт - заказ»:**

TABLE product\_has\_transaction

(

product\_id INT NOT NULL,

transaction\_id INT NOT NULL,

);

## **1.1.3 Описание доменов атрибутов**

Ниже будут описаны необходимые атрибуты сущностей и их домены для создания БД интернет магазина.

Отношение «user»

Таблица 1 – Описание доменов отношения «user»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`email`** | VARCHAR(255) NOT NULL |
| **`password`** | VARCHAR(32) NOT NULL |
| **`create\_time`** | TIMESTAMP NULL DEFAULT CURRENT\_TIMESTAMP |
| **`firstname`** | VARCHAR(45) NOT NULL |
| **`lastname`** | VARCHAR(45) NOT NULL |
| **`id`(прим.: код пользователя)** | INT NOT NULL AUTO\_INCREMENT |
| **`is\_stuff`** | TINYINT(1) NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

**create\_time –** дата регистрации пользователя (текущая дата и время в момент регистрации пользователя)

Отношение «category»

Таблица 2 – Описание доменов отношения «category»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`name\_category`** | VARCHAR(45) NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

Отношение «subcategory»

Таблица 3 – Описание доменов отношения «subcategory»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`name\_subcategory`** | VARCHAR(32) NOT NULL |
| **`category\_id`** | INT NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

**category\_id –** внешний ключ к отношению «category» поле «id»

Отношение «product»

Таблица 4 – Описание доменов отношения «product»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`name`** | VARCHAR(45) NOT NULL |
| **`price`** | INT NOT NULL |
| **`description`** | TEXT(255) NOT NULL |
| **`shelf\_life`** | VARCHAR(45) NOT NULL |
| **`weight`** | DOUBLE NOT NULL |
| **`image`** | BLOB NULL |
| **`subcategory\_id`** | INT NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

**subcategory\_id** – внешний ключ к отношению «subcategory» поле «id»

Отношение «transaction»

Таблица 5 – Описание доменов отношения «transaction»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`number\_card`** | VARCHAR(45) NOT NULL |
| **`sum`** | DOUBLE NOT NULL |
| **`products\_list`** | LONGTEXT NOT NULL |
| **`user\_id`** | INT NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

**user\_id** – внешний ключ к отношению «user» поле «id»

Отношение «admin»

Таблица 6 – Описание доменов отношения «admin»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`email`** | VARCHAR(45) NOT NULL |
| **`password`** | VARCHAR(45) NOT NULL |
| **`admin\_firstname`** | VARCHAR(45) NOT NULL |
| **`admin\_lastname`** | VARCHAR(45) NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

Отношение «comments»

Таблица 7 – Описание доменов отношения «comments»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`commentator\_firstname`** | VARCHAR(45) NOT NULL |
| **`commentator\_lastname`** | VARCHAR(45) NOT NULL |
| **`commentary`** | LONGTEXT NOT NULL |
| **`user\_id`** | INT NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

**user\_id** – первичный ключ (значение - автоинкримент с 1)

Отношение «review»

Таблица 8 – Описание доменов отношения «review»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`id`** | INT NOT NULL AUTO\_INCREMENT |
| **`id\_user`** | VARCHAR(45) NULL |
| **`user\_firstname`** | VARCHAR(45) NULL |
| **`commentary`** | LONGTEXT NULL |
| **`is\_user`** | TINYINT NULL |
| **`user\_id`** | INT NOT NULL |

Ограничения:

**id** – первичный ключ (значение - автоинкримент с 1)

**user\_id** – внешний ключ к отношению «user» поле «id»

Отношение «product\_has\_transaction»

Таблица 9 – Описание доменов отношения «product\_has\_transaction»

|  |  |
| --- | --- |
| **Имя атрибута** | **Домен атрибута** |
| **`product\_id`** | INT NOT NULL |
| **`transaction\_id`** | INT NOT NULL |

Ограничения:

**product\_id** – первичный ключ (значение — id в таблице Product)

**transaction\_id** – первичный ключ (значение — id в таблице Transaction)

## **1.1.4 Создание схемы в виде ER-диаграммы**

Создание логической схемы является следующим этапом в процессе разработки и проектирования базы данных магазина.

**Логическое (даталогическое) проектирование** — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи [2].

В процессе разработки логическая модель данных постоянно тестируется и проверяется на соответствие требованиям пользователей.

Созданная логическая модель данных является источником информации для этапа физического проектирования и обеспечивает разработчика физической базы данных средствами поиска компромиссов, необходимых для достижения поставленных целей, что очень важно для эффективного проектирования. При правильно организованном сопровождении поддерживаемая в актуальном состоянии модель данных позволяет точно и наглядно представить любые вносимые в базу данных изменения, а также оценить их влияние на прикладные программы и использование данных, уже имеющихся в базе.В настоящий момент фактическим стандартом в моделировании баз данных стала модель «сущность-связь» [2].

Рассмотрим основные понятия, используемые для построения ER-диаграммы.

**Сущность** (entity) – это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна. В диаграммах сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа. Каждый экземпляр сущности (объект) должен быть отличим от любого другого экземпляра той же сущности [3].

**Связь** (relationship) – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь). Возможны связи на основе отношений:

– один-к-одному, т.е., каждый экземпляр одного множества сущностей допускает соединение не более чем с одним экземпляром другого множества сущностей.

– один-ко-многим означает, что каждой записи первой сущности могут соответствовать несколько записей из второй сущности. Однако каждой записи второй сущности соответствует только одна запись из первой сущности. Например, один клиент может сделать несколько заказов.

– многие-ко-многим означает, что каждой записи первой сущности могут соответствовать несколько записей из второй сущности. Однако и каждой записи второй сущности может соответствовать несколько записей из первой сущности [3]. Например, один заказ может содержать несколько продуктов и один продукт может находиться в нескольких заказах.

Таким образом, на основе представлений конечных пользователей БД интернет магазина хлебной продукции можно составить обобщенное представление предметной области, выделив основные объекты будущей базы данных и отбросив второстепенные. В качестве концептуальной модели была выбрана ER-диаграмма. ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является цикличным. ER-модель использует графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями.

В базе данных «Интернет магазин» можно выделить следующие основные сущности: «Пользователь», «Администратор», «Продукт», «Комментарий», «Категория», «Подкатегория», «Обзор», «Заказ». Рассмотрим атрибуты каждой сущности, необходимые конечным пользователям для работы.

«Пользователь»: код пользователя, имя, фамилия, е-мэйл, пароль.

«Продукт»: код продукта, описание, цена, вес.

«Категория»: код категории, коды подкатегорий, которые находятся в категориях.

«Комментарий»: код комментария, текст, имя автора, фамилия автора.

«Отзыв»: код отзыва, имя автора, фамилия автора, текст отзыва.

«Подкатегория»: код подкатегории, коды продуктов, которые находятся в подкатегории.

«Администратор»: код администратора, е-мэйл, пароль, имя, фамилия.

«Заказ»: код заказа, номер карты для оплаты, список товаров, сумма заказа.

Основные связи в проектируемой ER-диаграмме: «Категория» имеет название и у каждой «Категории» имеется «Подкатегории». «Подкатегория» имеет название и в нем содержатся «Продукты». У продуктов есть «Комментарии», которые создает «Пользователь». У продуктов есть «Обзоры», имеющие автора «Пользователя». «Заказы» имеет несколько «Продуктов».

Таким образом, на основе приведенных представлений пользователей, выделенных основных объектов, описанных атрибутов и связей была спроектирована семантическая модель предметной области «Онлайн магазин» (Приложение А: ER-диаграмма).

## **1.1.5 Описание связей между объектами**

1. У каждой категории может быть несколько подкатегории. Отсюда возникает связь: Category–Subcategory (1: N);
2. У каждой подкатегории может быть несколько продуктов. Отсюда возникает связь: Subcategory – Product (1: N);
3. Один товар может быть в нескольких заказах и один заказ может включать в себя несколько товаров. Отсюда возникает связь: Transaction–Product (N: N); Данная связь порождает новую сущность Product\_has\_transaction со связями Transaction (1: N) Product (N: 1) соответственно.
4. У каждого продукта может быть несколько комментариев. Отсюда возникает связь: User–Comments (1: N);
5. У каждого продукта может быть несколько обзоров. Отсюда возникает связь: Product–Review (1: N);
6. У каждого пользователя может быть несколько заказов. Отсюда возникает связь: User–Transaction (1: N);

## **1.2 Построение концептуального уровня БД интернет магазина хлебной продукции**

## **1.2.1 Описание функциональных зависимостей, имеющих место в предметной области**

Отношение **User** = {id, firstname, lastname, email, password,is\_stuff}

От имени, фамилии, пароля и является ли пользователь работником или нет зависит его уникальный номер и электронная почта, т.к. два разных человека не могут быть зарегистрированными под одной электронной почтой.

{ id, email}  { firstname, lastname, password, is\_stuff,}

Отношение **Product** = { id, name, price, description, shelf\_life, weight, image, модель Subcategory}

От названия, цены, описания, срока годности, веса зависит уникальный номер продукта и номер подкатегории в которой он находится, т.к. не может существовать два одинаковых продукта в одной подкатегории.

{ id, модель Subcategory }  { name, price, description, shelf\_life, weight }

Отношение **Admin** = { id, email, password, admin\_firstname, admin\_lastname}

От имени, фамилии, пароля администратора зависит его уникальный номер и его электронная почта, т.к. не может быть зарегистрировано два администратора под одной электронной почтой.

{id, email}  {password, admin\_firstname, admin\_lastname}

Отношение **Category** = { id, name\_subcategory}

От названия категории зависит её уникальный номер, т.к. не может существовать две категории с одним названием.

{id} → {name\_category}

Отношение **Subcategory** = { id, name\_category, модель Category }

От название подкатегории зависит её уникальный номер и уникальный номер категории в которой он находится, т.к. не может существовать две подкатегории с одинаковым названием в одной категории.

{id, category\_id} → {name\_subcategory}

Отношение **Comments** = { id, product\_id, user\_id, commentary}

От текста комментария зависит уникальный номер комментария, номер продукта, уникальный номер пользователя, т.к. два одинаковых комментария не могут быть написаны одним пользователем под одним продуктом.

{id,user\_id, product\_id} → {commentary}

Отношение **Review** = { id, product\_id, user\_id, commentary}

От текста отзыва зависит уникальный номер отзыва, номер продукта, уникальный номер пользователя, т.к. два одинаковых отзыва не могут быть написаны одним пользователем под одним продуктом.

{id,user\_id, product\_id} → {commentary}

## **1.2.2 Нормализация отношений**

## **1.2.2.1 Нормализация до первой нормальной формы (1НФ)**

Отношение находится в первой нормальной форме (1НФ), если каждый его атрибут атомарен, т.е. содержит только одно значение, а не список значений.

Рассмотрим базовые отношения БД «Интернет магазин хлебной продукции»:

Отношение **User** = {id, firstname, lastname, email, password, create\_time, is\_stuff}

Отношение **Transaction** = { id, number\_card, user\_id, sum}

Отношение **Review** = { id, commentary, product\_id, user\_id}

Отношение **Comment** = { id,commentary,user\_id}

Отношение **Product** = { id, name, price, description, shelf\_life, weight, image, subcategory\_id}

Отношение **Subcategory** = { id, name\_subcategory }

Отношение **Admin** = { id, email, password, admin\_firstname, admin\_lastname }

Отношение **Category** = { id, name\_category }

Отношение **Product\_has\_transaction** = { product\_id, transaction\_id }

Все базовые отношения находятся в 1НФ; Производное отношение «Product\_has\_transaction» введено для того, чтобы нормализовать отношения «Product», «Transaction». Т.е. были устранены поля, которые представляют собой списки значения.

## **1.2.2.2 Обоснование выбора первичных ключей и приведение ко второй нормальной форме (2НФ)**

Отношение находится во второй нормальной форме, если все атрибуты, за исключением ключа, зависят от ключа полностью, а не частично. Иными словами, если отношение находится в первой нормальной форме, и при этом любой её атрибут, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа.

Рассмотрим выбранные первичные ключи отношений БД «Интернет магазин хлебной продукции».

Отношение User содержит один потенциальный ключ: {id}; он же и является первичным.

Отношение Review содержит три потенциальных ключа: {id, product\_id, user\_id}; ключ {id} - является первичным.

Отношение Comments содержит три потенциальных ключа: {id, product\_id, user\_id}; ключ {id} - является первичным.

Отношение Transaction содержит один потенциальный ключ: {id}; он же и является первичным.

Отношение Product содержит два потенциальный ключ: {id, subcategory\_id}; ключ {id} является первичным.

Отношение Admin содержит один потенциальный ключ: {id}; он же и является первичным.

Отношение Subcategory содержит два потенциальный ключ: {id, category\_id}; ключ {id} является первичным.

Отношение Category содержит один потенциальный ключ: {id}; он же и является первичным.

Отношение Transaction содержит два потенциальный ключ: {id, user\_id}; ключ {id} является первичным.

Отношение Product\_has\_transaction содержит два потенциальный ключ: {product\_id, transaction\_id}; они же и являются составным первичным ключом.

## **1.2.2.3 Нормализация до третьей нормальной формы (3НФ). Выводы**

Отношение находится в третьей нормальной форме (3НФ), если оно находится во второй нормальной форме, и не имеет транзитивной зависимости атрибута от ключа. Иными словами, так как все атрибуты зависят от ключа, то отношение, чтобы считаться приведенным к третьей нормальной форме, не должно иметь зависимостей между атрибутами.

Все базовые отношения БД «Онлайн магазин» находятся в 3НФ, т.к. все не ключевые атрибуты этих отношений не транзитивно зависят от ключей.

Отношение **User** = {id, firstname, lastname, email, password, create\_time, is\_stuff}

Отношение **Transaction** = { id, number\_card, user\_id, sum}

Отношение **Review** = { id, commentary, product\_id, user\_id}

Отношение **Comment** = { id,commentary,user\_id}

Отношение **Product** = { id, name, price, description, shelf\_life, weight, image, subcategory\_id}

Отношение **Subcategory** = { id, name\_subcategory }

Отношение **Admin** = { id, email, password, admin\_firstname, admin\_lastname }

Отношение **Category** = { id, name\_category}

Между отношениями «Product»и «Transaction» существует связь многие ко многим, т.к. множество продуктов может быть в множество заказов. Таким образом проявляется транзитивная зависимость между эти отношениями и для её устранения мы проводим декомпозицию этих отношений с внедрением вспомогательной таблицы «Product\_has\_transaction»

Отношение **Product\_has\_transaction** = { product\_id, transaction\_id }

Все не ключевые атрибуты этих отношений не транзитивно зависят от ключей. Таким образом все базовые отношения БД находятся в 3НФ.

Итак, была произведена нормализация отношений: структура БД «Онлайн магазин» была приведена к виду, обеспечивающему минимальную логическую избыточность. Была сокращена потенциальная противоречивость хранимой в базе данных информации.

## **1.2.3 Логическое (даталогическое) проектирование базы данных**

На этапе логического проектирования разрабатывается логическаяструктура будущей базы данных интернет магазина.

База данных создаётся на основании схемы базы данных. Инфологическую модель данных, построенную в виде ER-диаграммы, необходимо преобразовать в схему БД. Преобразование ER-диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, в таблицы-отношения [2].

Для реляционной модели данных даталогическая модель — это набор схем отношений, обычно с указанием первичных ключей и связей между отношениями, представляющих собой внешние ключи.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика СУБД.

Ранее были описаны связи между отношениями, обоснован выбор первичных ключей. Результат логического проектирования отображен на даталогической схеме БД «Интернет магазина хлебной продукции» (Приложение Б – Даталогическая схема).

## **2. Физическая реализация модели БД онлайн магазина**

## **2.1 Обоснование выбора целевой системы управления базами данных (СУБД)**

На данный момент существует множество СУБД, например:

Oracle Database – это мощный программный комплекс, позволяющий создавать приложения любой степени сложности. Ядром этого комплекса является база данных, хранящая информацию, количество которой за счет предоставляемых средств масштабирования практически безгранично. C высокой эффективностью работать с этой информацией одновременно может практически любое количество пользователей (при наличии достаточных аппаратных ресурсов), не проявляя тенденции к снижению производительности системы при резком увеличении их числа. Еще одной составляющей успеха СУБД Oracle является многоплатформенность. Но очень высокая стоимость сервера и поддержки [3].

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

PostgreSQL – свободная объектно-реляционная СУБД. PostgreSQL базируется на языке SQL. То, что PostgreSQL не просто реляционная, а объектно-реляционная СУБД, делает ее более гибкой и надежной в сравнении с аналогами. PostgreSQL поддерживает целостность данных. Открытость кодов PostgreSQL означает их абсолютную доступность для любого, а либеральная BSD лицензия не накладывает никаких ограничений на использование кода [5].

Для реализации физической модели была выбрана СУБД MySQL, так как она соответствует всем требованиям и поддерживает инструменты бизнес-аналитики с поддержкой самообслуживания. Так же она обладает высокой производительностью и защищенностью данных, поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных.

## **2.2 Создание таблиц данных в среде целевой СУБД**

SQL (язык структурированных запросов) — декларативный язык программирования, применяемый для создания, модификации и

управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных [1].

В среде администрирования и разработки MySQL с помощью SQL-запросов была создана база данных «onlineshop»:

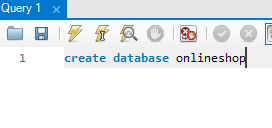


Рисунок 1 – Создание БД «onlineshop»

А также все необходимые нам таблицы: «user», «admin», «product» и т.д. Создание таблиц представлено на примере таблицы «user»:

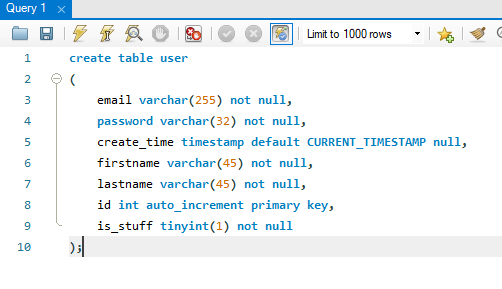


Рисунок 2 – Создание таблицы «user»

В среде администрирования и разработки MySQL с помощью SQL-запросов было создано представление «view\_user»:

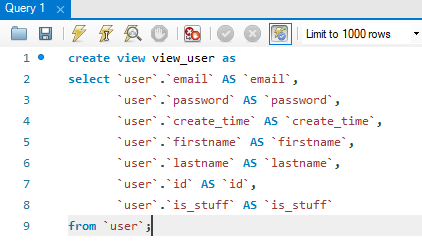


Рисунок 3 – Создание представления «view\_user»

А также все необходимые нам представления: «view\_product», «view\_category», «view\_subcategory», «view\_admin», «view\_comments», «view\_review», «view\_transaction», «view\_product\_hast\_transaction».

В данной базе данных находится множество таблиц для реализации темпоральности. Так, триггер newUser срабатывает при добавлении данных в таблицу «user»:

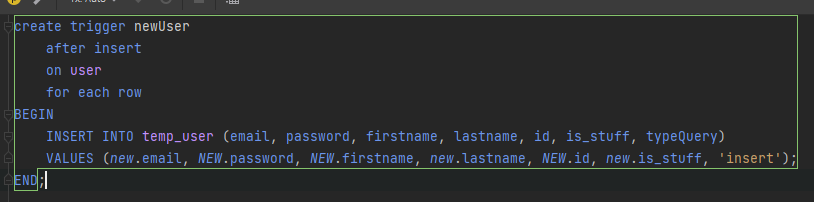


Рисунок 4 – Триггер newUser

В данной базе данных присутствует несколько триггеров для обеспечения функциональности. Далее описаны все эти триггеры.

Триггеры newUser, deleteUser, updateUser используются для записи действий, производимых над таблицей «user», в зависимости от произведённого действия происходит запись в «temp\_user» старого значения таблицы «user».

Триггеры newProduct, deleteProduct, updateProduct используются для записи действий, производимых над таблицей «product», в зависимости от произведённого действия происходит запись в таблицу «temp\_product» старого значения таблицы «temp\_product».

Триггеры newCategory, deleteCategory, updateCategory используются для записи действий, производимых над таблицей «category», в зависимости от произведённого действия происходит запись в таблицу «story», а так же запись в «temp\_category» старого значения таблицы «category».

Триггеры newSubCategory, deleteSubCategory, updateSubCategory используются для записи действий, производимых над таблицей «subcategory», в зависимости от произведённого действия происходит запись в таблицу «story», а так же запись в «temp\_subcategory» старого значения таблицы «subcategory».

Триггеры newTransaction, deleteTransaction, updateTransaction используются для записи действий, производимых над таблицей «transaction», в зависимости от произведённого действия происходит запись в таблицу «story», а так же запись в «temp\_transaction» старого значения таблицы «transaction».

Триггеры newAdmin, deleteAdmin, updateAdmin используются для записи действий, производимых над таблицей «admin», в зависимости от произведённого действия происходит запись в таблицу «story», а так же запись в «temp\_admin» старого значения таблицы «admin».

Триггеры newComment deleteComment, updateComment используются для записи действий, производимых над таблицей «comment», в зависимости от произведённого действия происходит запись в таблицу «story», а так же запись в «temp\_comment» старого значения таблицы «comment».

Триггеры newReview deleteReview, updateReview используются для записи действий, производимых над таблицей «review», в зависимости от произведённого действия происходит запись в таблицу «story», а так же запись в «temp\_review» старого значения таблицы «review».

Триггер undo() выполняет следующую функцию. Он срабатывает при удалении значения из таблицы «story». Осуществляет тем самым основную функцию в реализации темпоральности базы данных библиотеки. При его выполнении триггер обращается к удаляемому значению и производит обратные действия, предшествующие добавлению в эту таблицу, т. е. если в таблице было указано, что было произведено добавление, в некоторую таблицу нашей базы данных, происходит удаление соответствующей записи. Если же в «story» указанно удаление или изменение, то триггер, наоборот, восстанавливает эту запись, её старое значение, получая при этом из соответствующей temp\_[Название\_Таблицы]таблицы.

Итак, при помощи СУБД в MySQL были созданы все необходимые таблицы для реализации БД онлайн магазина, определены первичные ключи, проведены связи между таблицами, а также созданы таблицы и триггеры для осуществления темпоральности базы данных. Все таблицы были заполнены необходимыми данными.

## **3. Разработка клиентского интерфейса для работы с базой данных**

Необходимо провести обоснование выбора ПО для взаимодействия с БД, основываясь на требованиях к БД. Из нескольких вариантов нужно выбрать один.

Интерфейс для взаимодействия пользователей с базой данных онлайн магазина должен соответствовать некоторым требованиям, а именно: быть дружественным, интуитивно понятным и позволять каждому пользователю работать с базой в соответствии с его полномочиями. То есть, должны быть предусмотрены различные роли входа, для каждой роли должны быть реализованы свои функции. Так, например, у администратора должен быть простой и удобный способ добавления, изменения и удаления информации о сотрудниках онлайн магазина, о его продуктах. Сотрудник же должен иметь возможность удаления, и изменения комментариев или отзывов, получать информацию о заказах покупателя, а также должен иметь доступ просмотра информации о покупателе. Покупатель должен иметь доступ к просмотру товаров, своих заказах, своих комментариев или отзывов, возможность совершить заказ. Таким образом, ни один из пользователей не имеет доступа к возможностям другого, если их функции не пересекаются.

Помимо различных представлений, для пользователей базы данных интерфейс должен предусматривать возможность отката базы данных, на количество указанных транзакций или до указанной даты, удаление записей, редактирование любой из записей, создание любых записей, интерфейс для создания резервной копии и восстановления по ней базы данных, словом, всё необходимое для поддержания работоспособности базы данных.

## **3.1 Выбор среды для разработки приложения**

Практически все современные языки программирования имеют возможность подключения и работы с СУБД MySQL. Поэтому следует определиться с выбором языка, рассмотри несколько из них, а именно: С++, Python и С#.

C++ — компилируемый, статически типизированный язык программирования общего назначения. Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Для удобной реализации перечисленных требований к интерфейсу БД «Онлайн магазин» был выбран язык С#. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML [4].

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов) [4].

Существует множество сред разработки подходящих для решения поставленной задачи, например, SharpDevelop, Rider, MS VisualStudio, Code::Blocks.

Code::Blocks достаточно прост в изучении, легковесен, бесплатен, однако не подходит для создания комплексных приложений в силу недостаточной функциональности и нестабильности – порой приходится сталкиваться с невнятными ошибками в отладке, а при работе с большим приложением это неудобно.

SharpDevelop используется, как более простая альтернатива MS VisualStudio. Его плюсом является производительность, однако данная среда имеет меньший функционал, нежели VisualStudio.

MS VisualStudio - Данная среда является бесплатной, официальной, обладает большим функционалом, имеет широкие возможности по разработке приложений под Windows.

Rider — IDE от компании JetBrains, поддерживает полный цикл разработки и ReSharper, прост в освоении. Поддерживает точно такие же функции, что и известная MS VisualStudio.

Поэтому интерфейс для взаимодействия пользователей с базой данных был реализован на языке С# в программной среде Rider.

Наиболее простой и удобной платформой для разработки приложений до сих пор остается Windows Presentation Foundation.

Windows Presentation Foundation —  аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) (начиная с версии [3.0](https://ru.wikipedia.org/wiki/.NET_Framework_3.0)), использующая язык [XAML](https://ru.wikipedia.org/wiki/XAML).

Windows Presentation Foundation включает широкий набор элементов управления, которые можно добавлять на формы: текстовые поля, кнопки, раскрывающиеся списки, переключатели и даже веб-страницы.

## **3.2 Описание основных классов и методов интерфейса**

Для реализации графического интерфейса и взаимодействия с ним, был разлизано восемь классов: MainWindow, Profile, UserMenu, DataBaseConnect, CreateTransaction, Authorize, AdminMenu, AddComment, AddReview. MainWindow имеет методы CreateMD5 – хэширование пароля, ButtonBase\_OnClick - метод считывающий информацию о пользователе, которую он ввел в окне регистрации, Login – метод переводящий пользователя в пользовательское окно, после удачно авторизации.

Класс Profile служит для хранения и вывода информации о пользователе. В нем содержится метод конструктор для инициализации пользователя, в котором происходит поиск о пользователе по его id.

Класс UserMenu реализует класс Windows и является окном для работы для пользователя. В нем содержится поле DataTable tb для хранение текущей информации, которую просматривает пользователь. Так же есть метод ListViewMenu\_SelectionChanged, который служит для выбора и открытия новых таблиц данных для пользователя. Метод CreateTransaction служит для создания заказа, AddComment и AddReview нужны для того, чтобы пользователь смог оставлять комментарии или отзывы к продукту. Метод Submit и Delete созданы для того, чтобы пользователь смог работать со своими комментариями и отзывами.

Класс Authorize служит для авторизации пользователя и открытия ему окна UserMenu.

Класс CreateTransaction представляет собой окно для ввода информации о заказе покупателя.

Класс AddComment и AddReview представляют окна для ввода комментария или отзыва о продукте.

AdminMenu является родителем класса UserMenu. При помощи этого класса администратору базы данных предоставляется полный доступ для работы с ней.

UML-диаграмма классов находится в Приложении В.

Исходный код приложения находится в Приложении Д.

## **3.3 Пример работы графического интерфейса**

После запуска программы на экране появляется окно для регистрации нового пользователя, в котором пользователь должен ввести правильные логин и пароль (рисунок 5).

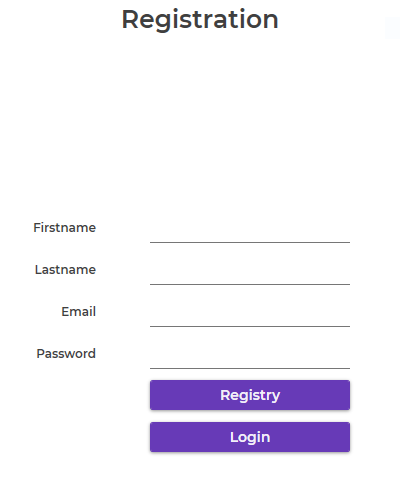


Рисунок 5 – Окно регистрации пользователя

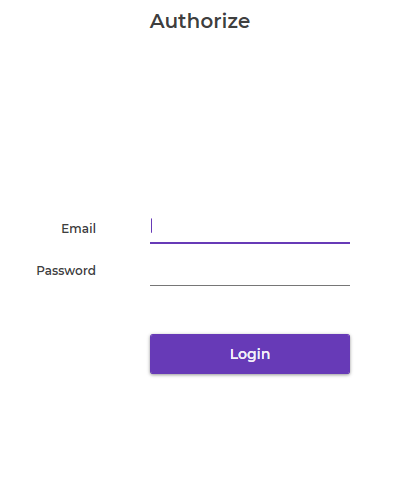
Если пользователь уже зарегистрирован в приложении, то он нажимает кнопку Login и ему открывается новое окно авторизации (рисунок 6).

Рисунок 6 – Окно авторизации пользователя

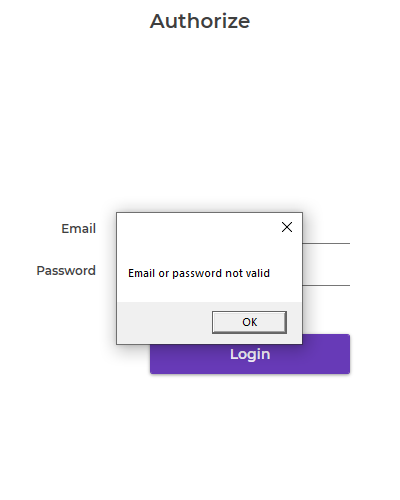
Если пользователем были введены неверные данные, программа сообщит ему об ошибке ввода данных (рисунок 7).

Рисунок 7 – Сообщение об ошибке авторизации.

После регистрации или авторизации пользователя открывается главное окно управления приложением. Рассмотрим на его примере администратора БД (рисунок 8).

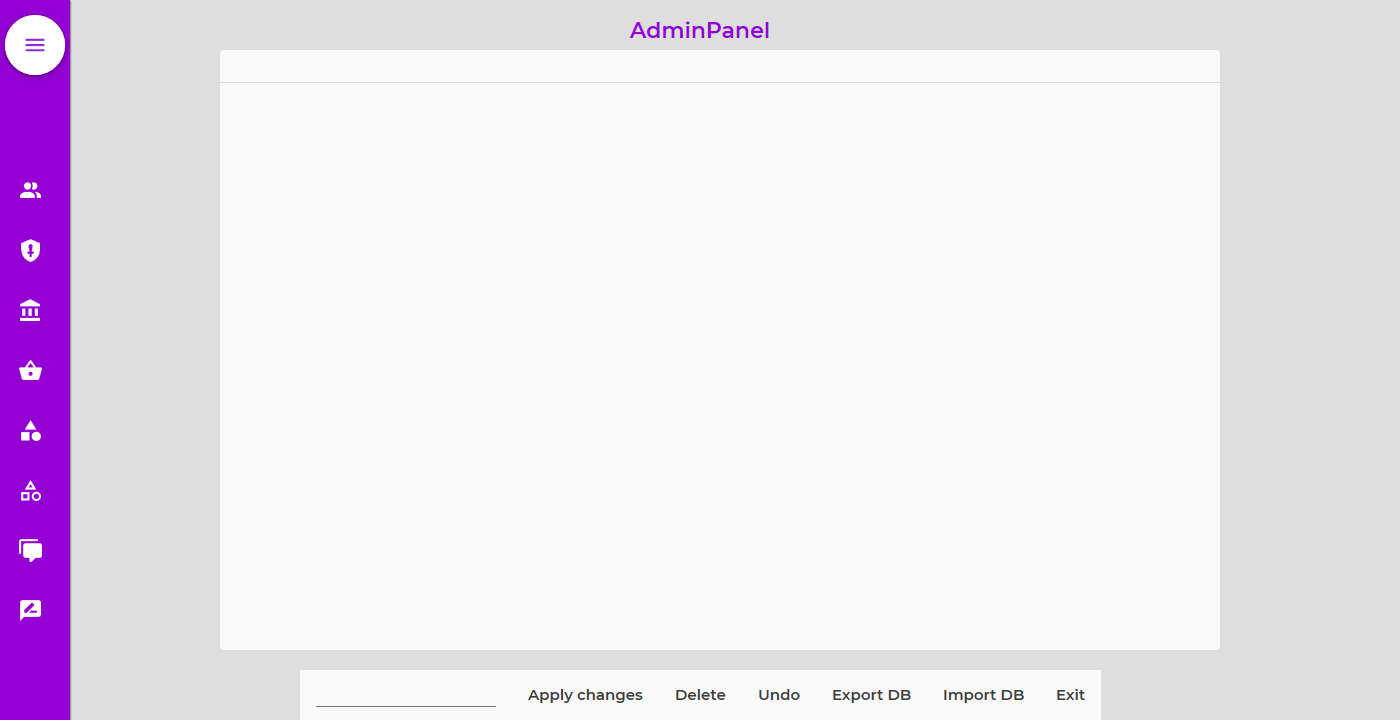


Рисунок 8 – Основное окно для работы с таблицами

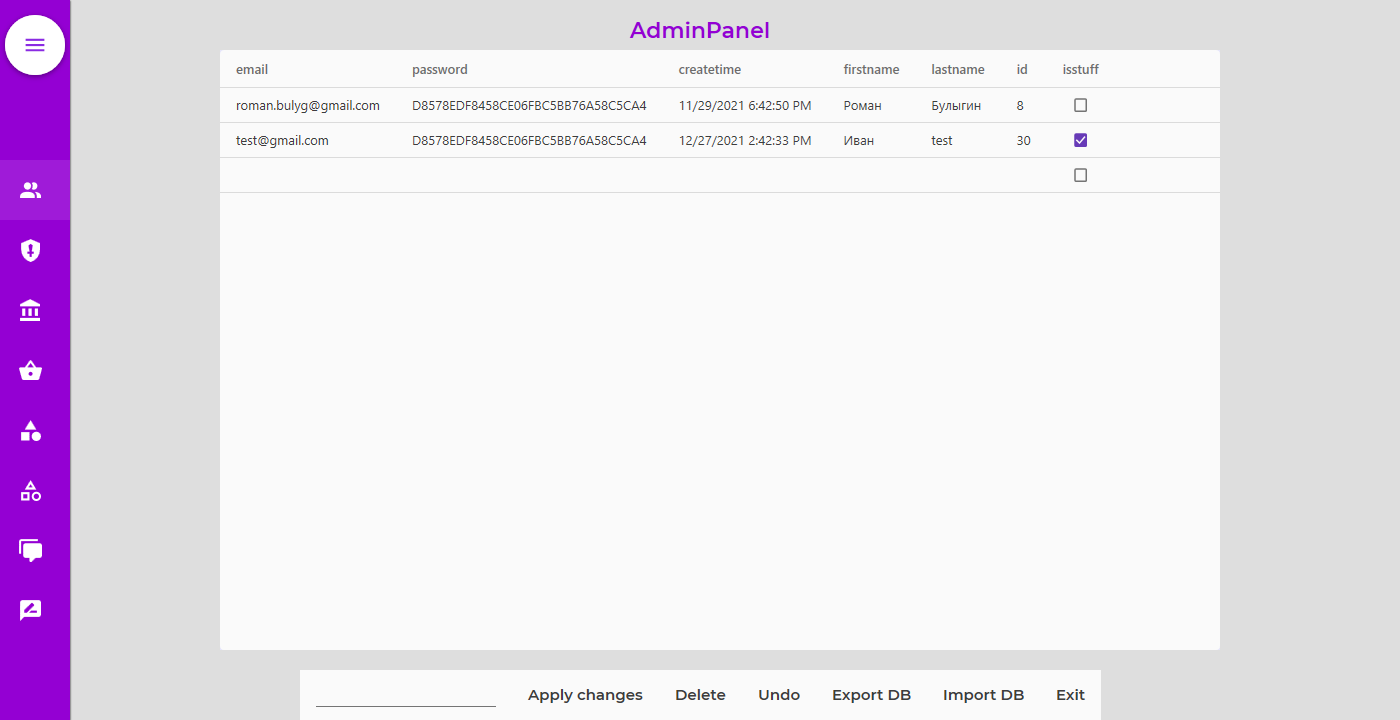
В левой области окна приложение администратор может выбирать одну из таблиц для отображения (рисунок 9).

Рисунок 9 – Выбор таблицы базы данных

После загрузки таблицы в области отображения, администратор может взаимодействовать с ними. Выберем таблицу продуктов для демонстрирования основных функий. (рисунок 10).

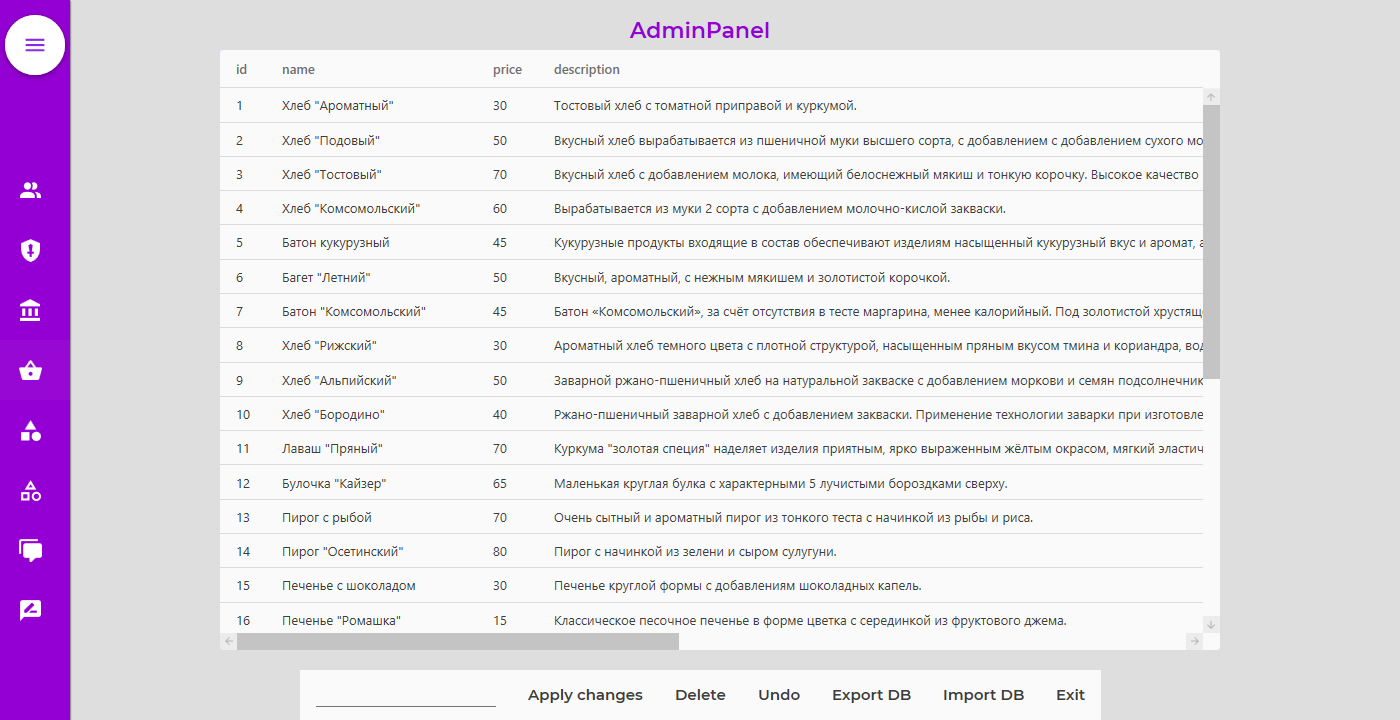


Рисунок 10 – Данные таблица Product

После выбора таблицы и её загрузки администратор может напрямую взаимодействовать с таблицей (рисунок 11).

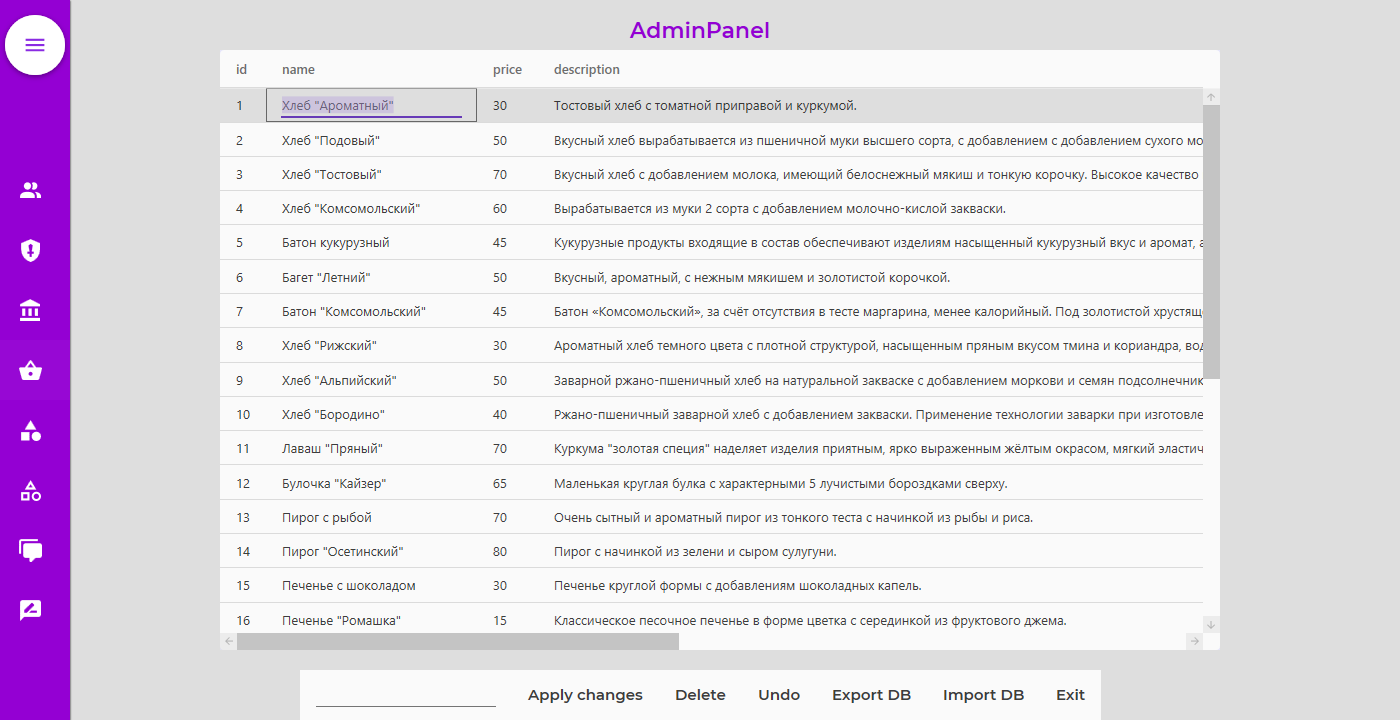


Рисунок 11 – Взаимодействие с таблицей

Для изменения какой либо строчки администратору достаточно просто выбрать её, ввести новое значение и нажать на кнопку Apply changes (рисунок 12).

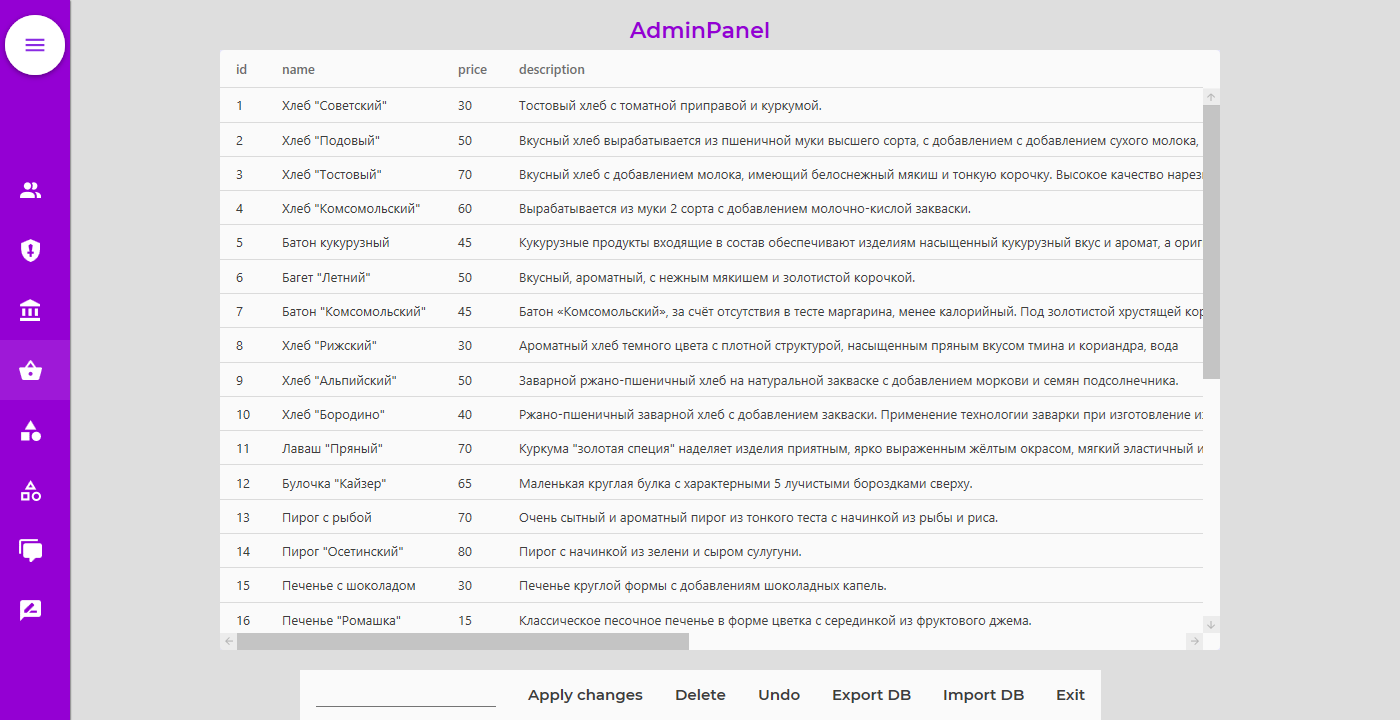
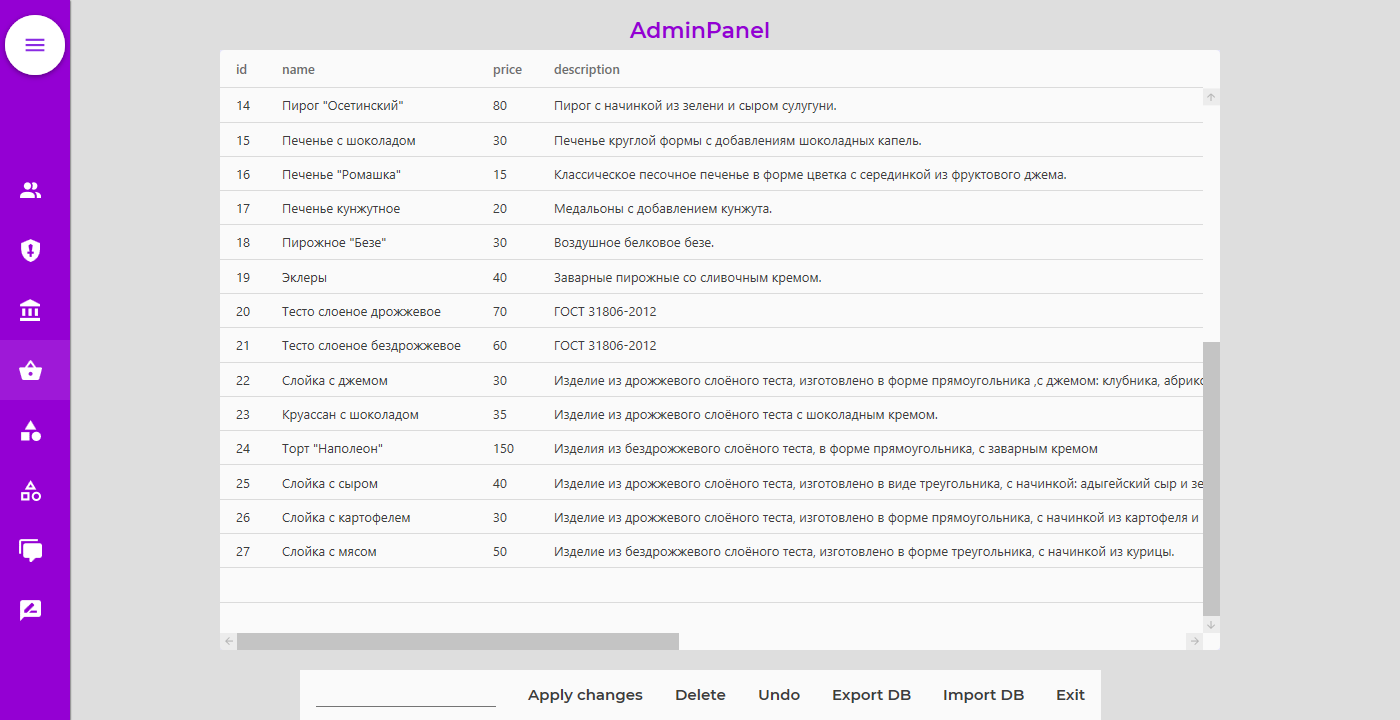
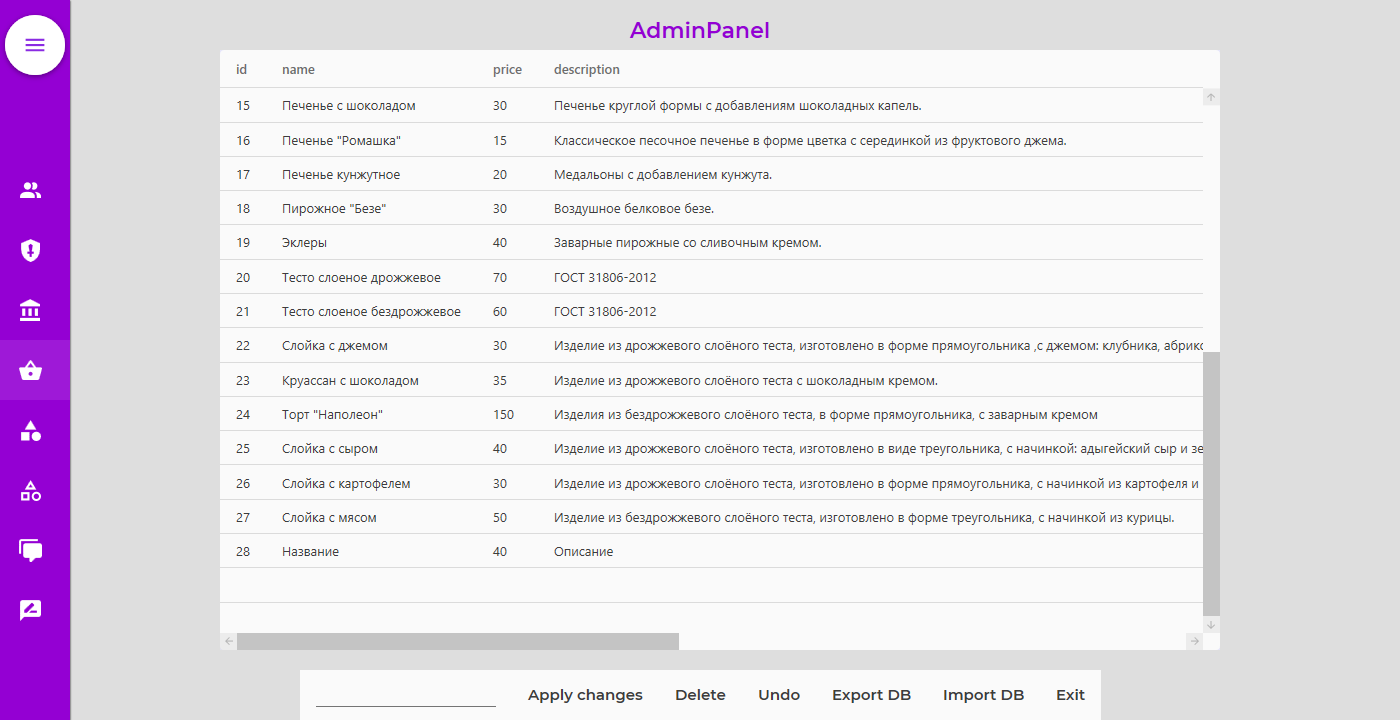


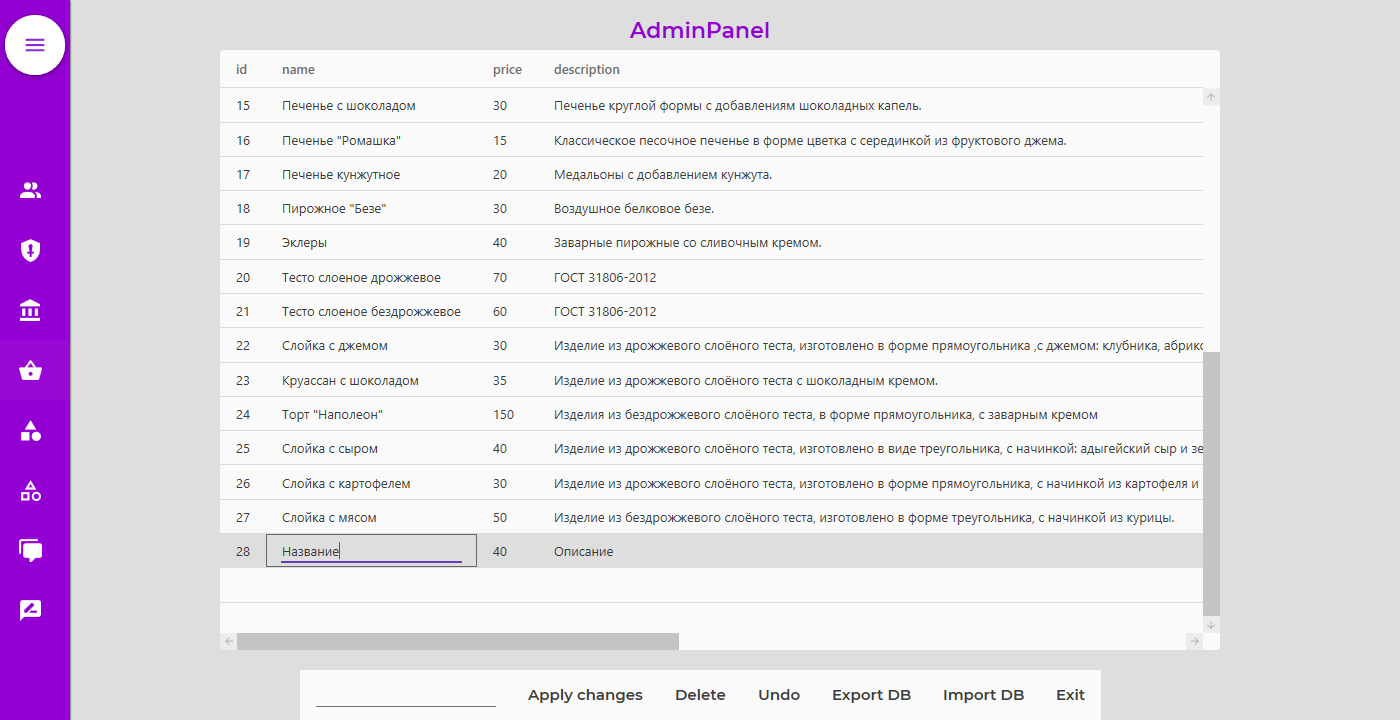
Рисунок 12 – Изменение значение поля записи в таблице

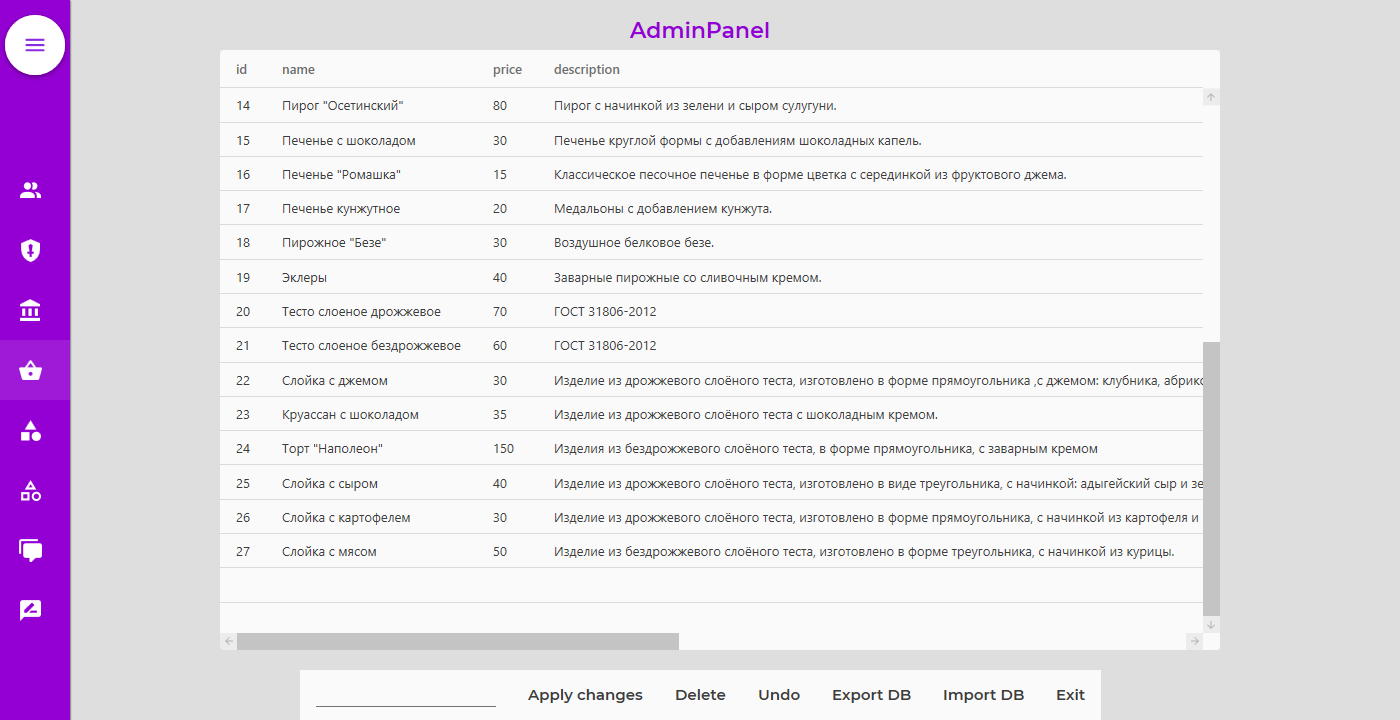
Для добавление новой записи администратору достаточно просто ввести значения в пустой строчке и нажать на кнопку Apple changes (рисунки 13,14).



Рисунки 13 и 14 – Добавление новой записи в таблицу

Для удаления записи из таблицы достаточно выбрать одно из полей записи, нажать кнопку Delete, следом подтвердить свой действие нажатием на кнопку Apply changes (рисунок 15). Из таблицы удалилась запись (рисунок 16).





Рисунки 15 и 16 – Удаление записи

Так же возможен поиск записи по всем полям таблицы. Для примера найдём запись о печенье «Ромашка» по его названию (рисунок 17).

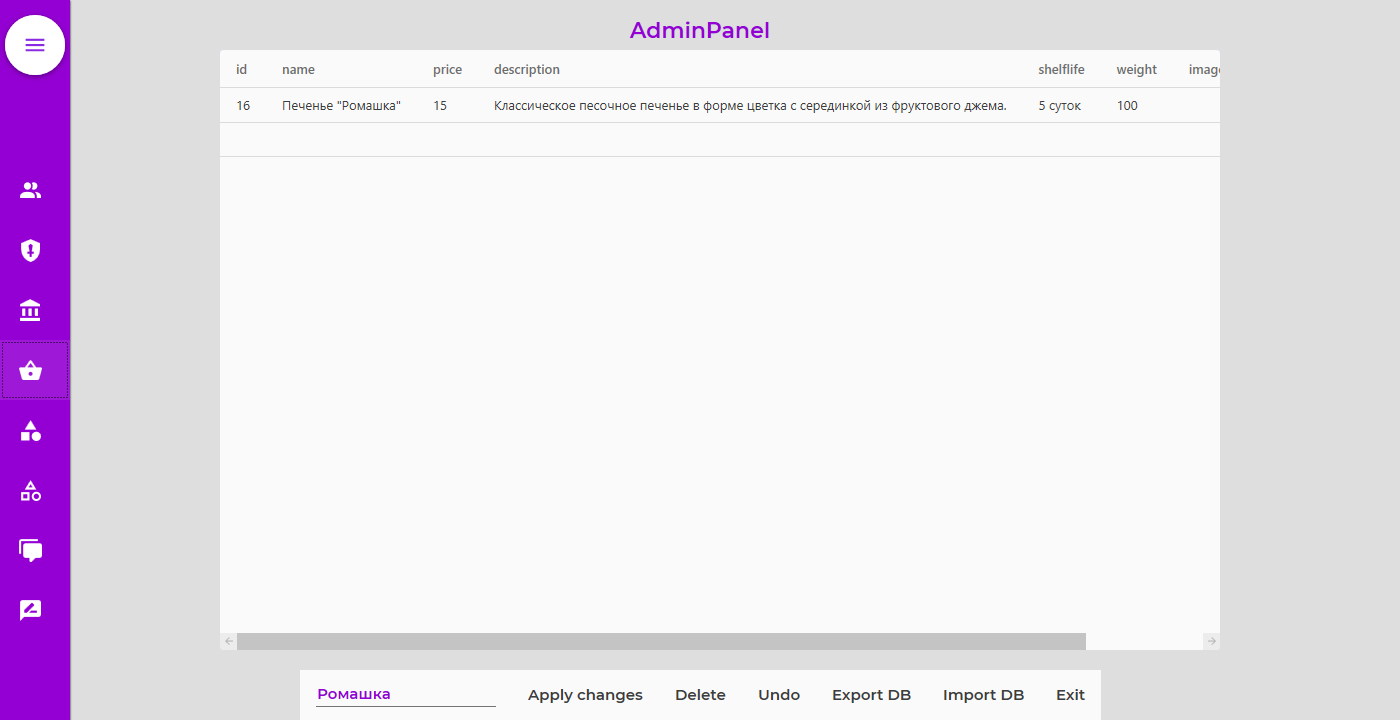
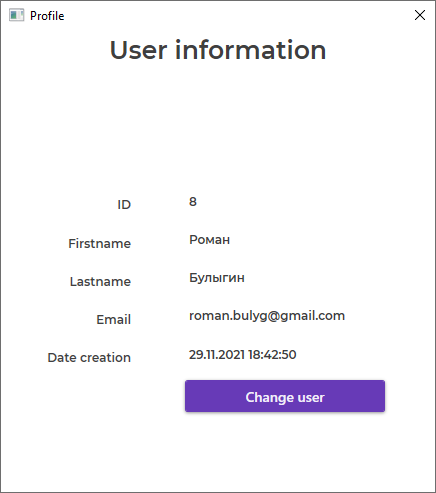


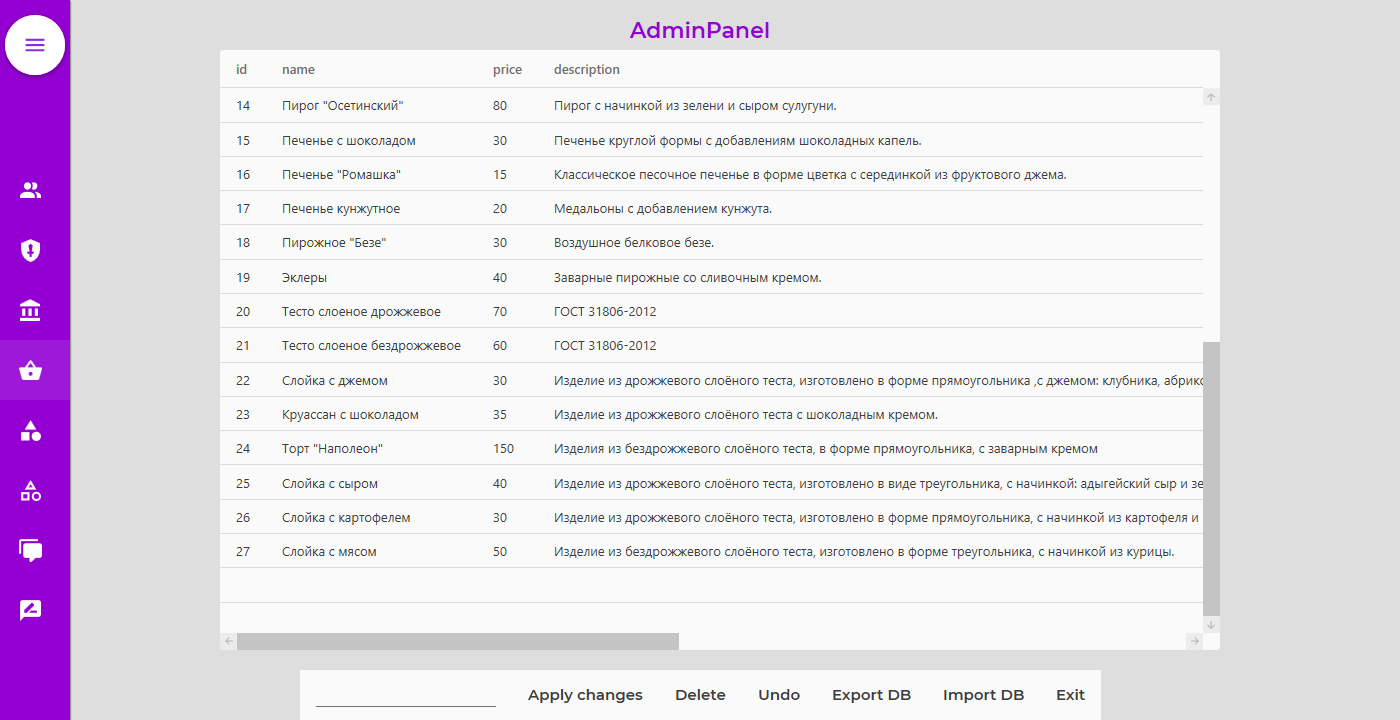
Рисунок 17 – Поиск по названию

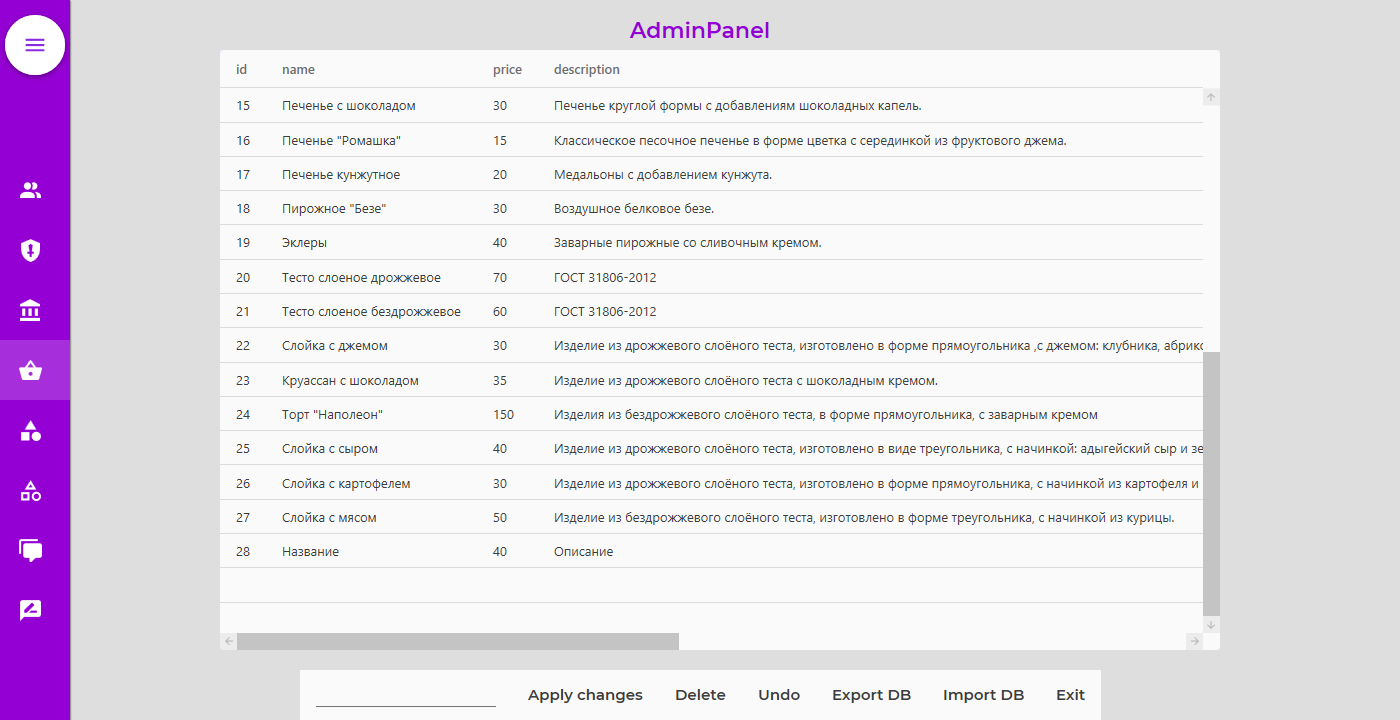
Пользователь можно сменить, если мы прошли авторизацию как пользователь. Выберем пункт с данными о нашем аккаунте. В открывшемся окне нажимаем кнопку для смены пользователя. (рисунок 18). На экране появляется окно для авторизации.



Рисунки 18 – Окно смены пользователя

Выберем таблицу Product, для восстановления ранее удаленной записи (рисунки 19 и 20).





Рисунки 19 и 20 – Восстановление удаленной записи

Так же была реализована функция создание резервной копии БД, для загрузки предыдущей её версии. Для создания резервной копии нужно нажать на кнопку Export DB (рисунок 21).

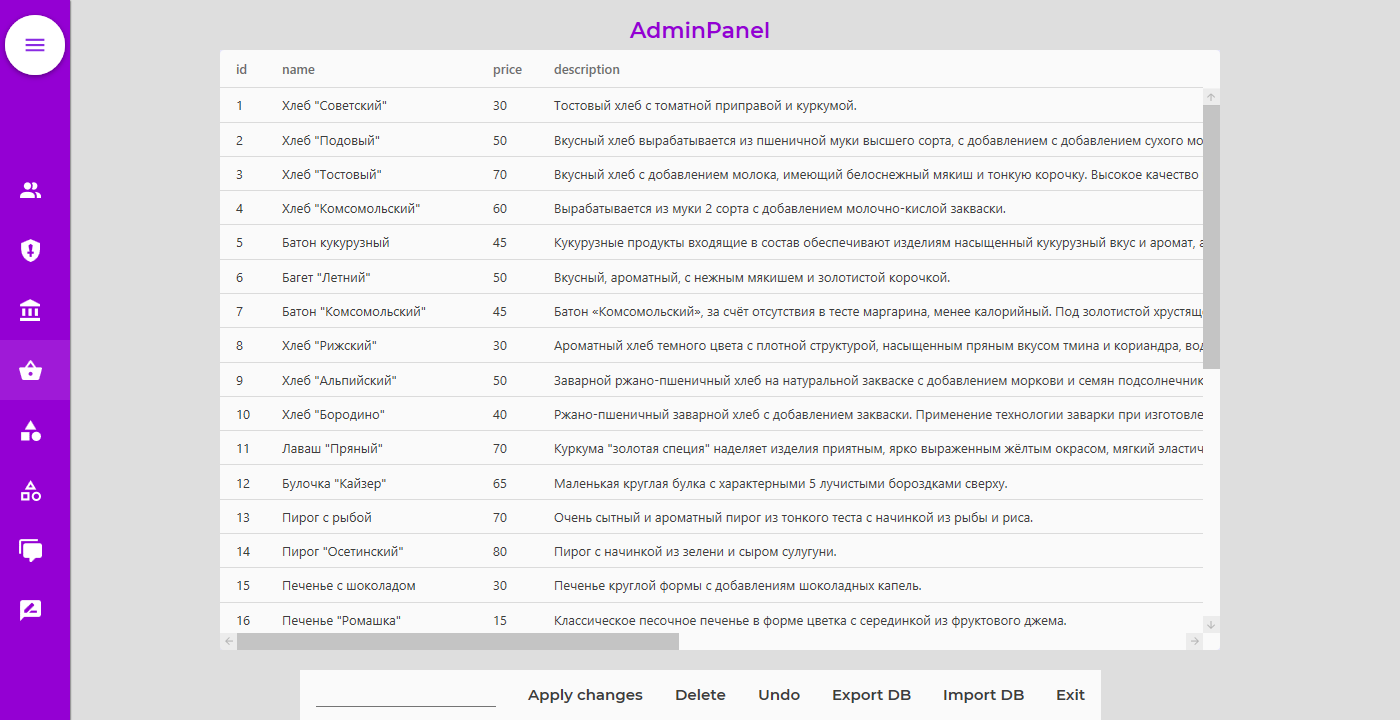
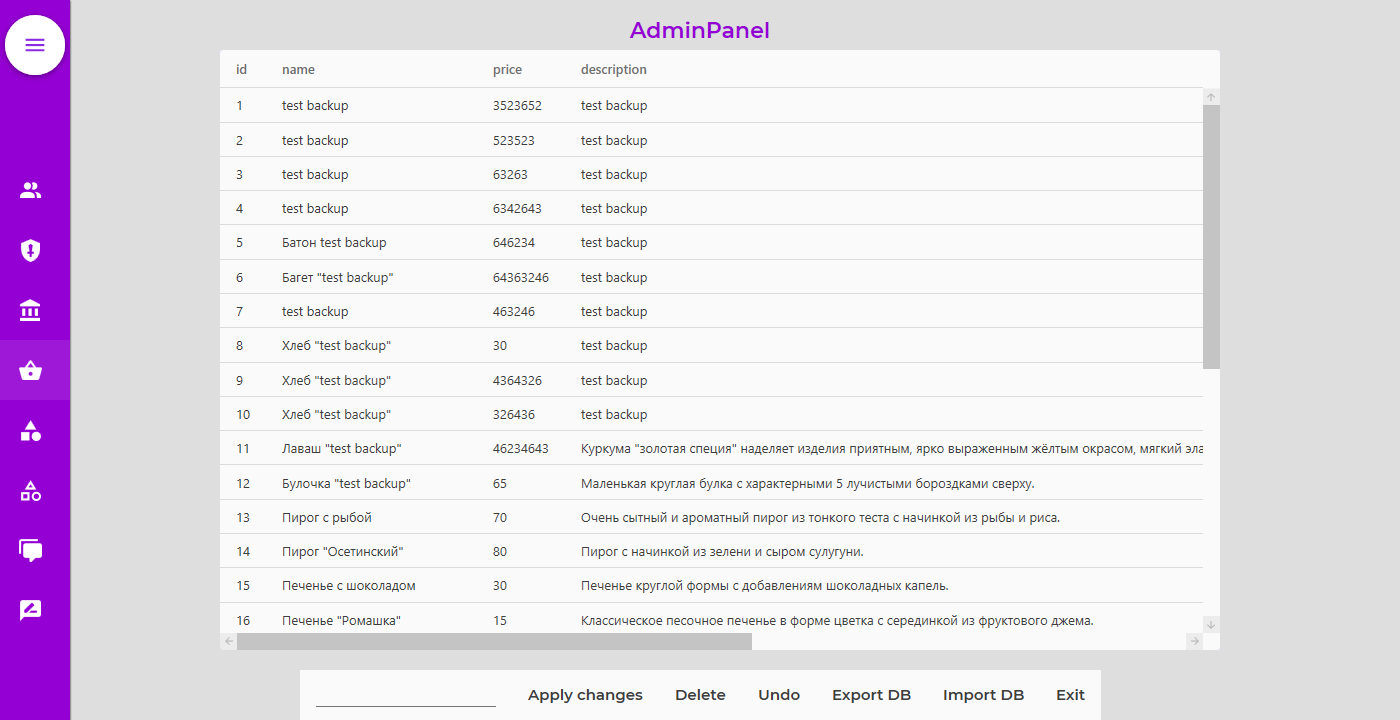
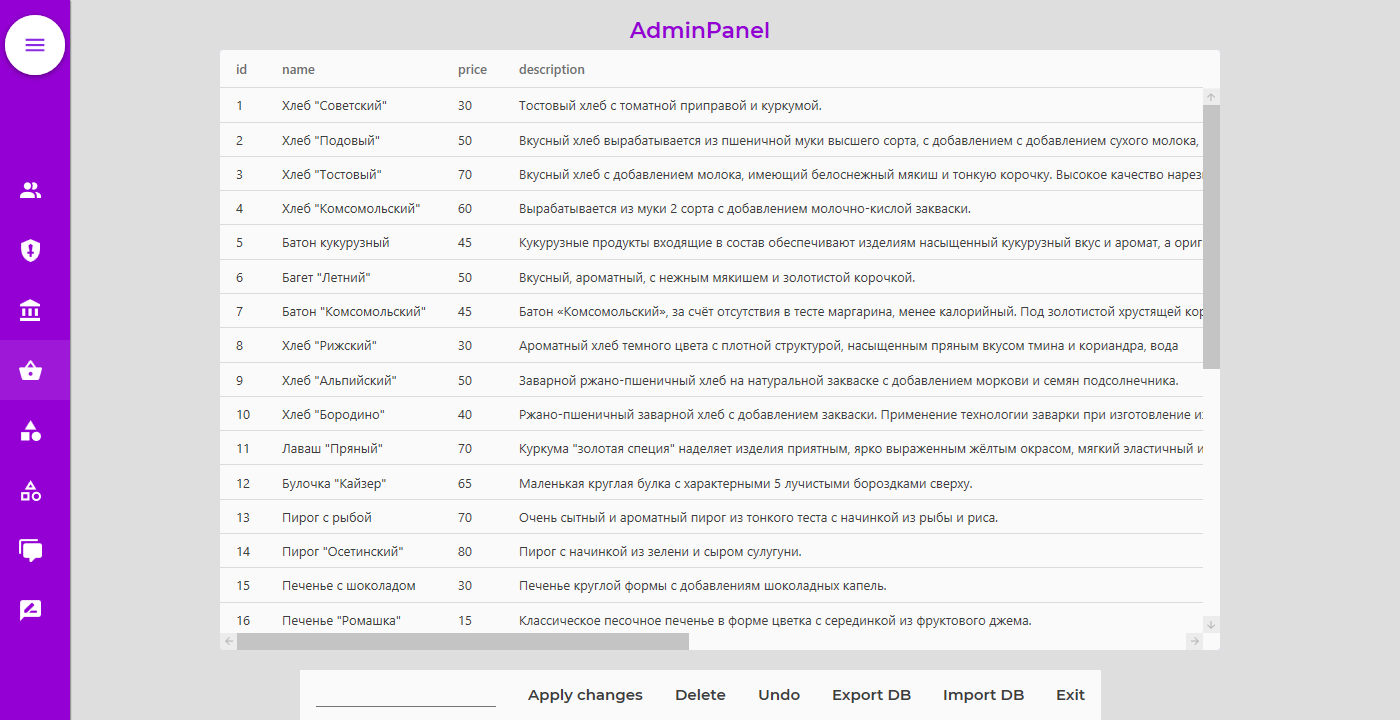


Рисунок 21 – Вид таблицы до изменения значений

Изменим несколько записей в таблице, чтобы убедиться в работе функции создания резервной копии (рисунок 22).

Рисунок 22 – Вид таблицы после изменения значений

Для загрузки созданной резервной копии нужно нажать на кнопку Import DB (рисунок 23).

Рисунок 23 – Вид таблицы после загрузки резервной копии БД

Таким образом было реализовано приложение, удовлетворяющее условиям поставленной задачи. Был разработан графический интерфейс для удобного взаимодействия с базой данных «Интернет магазин хлебной продукции».

# **ЗАКЛЮЧЕНИЕ**

В рамках данной курсовой работы была поставлена задача проектирования базы данных для онлайн магазина хлебной продукции и разработки приложения для работы с ней.

Для решения поставленной задачи была изучена предметная область, определены конечные пользователи, описаны объекты и их атрибуты. Также была разработана концептуальная модель в виде ER-диаграммы. Составлена даталогическая схема отношений. В ходе выполнения данной работы была изучена выбранная СУБД MySQL, ее возможности и функционал.

В результате была спроектирована и реализована база данных интернет магазина хлебной продукции и реализован набор интерфейсов для работы с ней. Так же были проведены необходимые тестирование и отладка разработанного программного обеспечения.

## **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Дейт К.Дж. Введение в системы баз данных, 6-е издание: Пер. с англ. – К.; М.; СПб.: Издательский дом «Вильямс», 2000. – 848 с.

2. Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2000. – 1120 с.

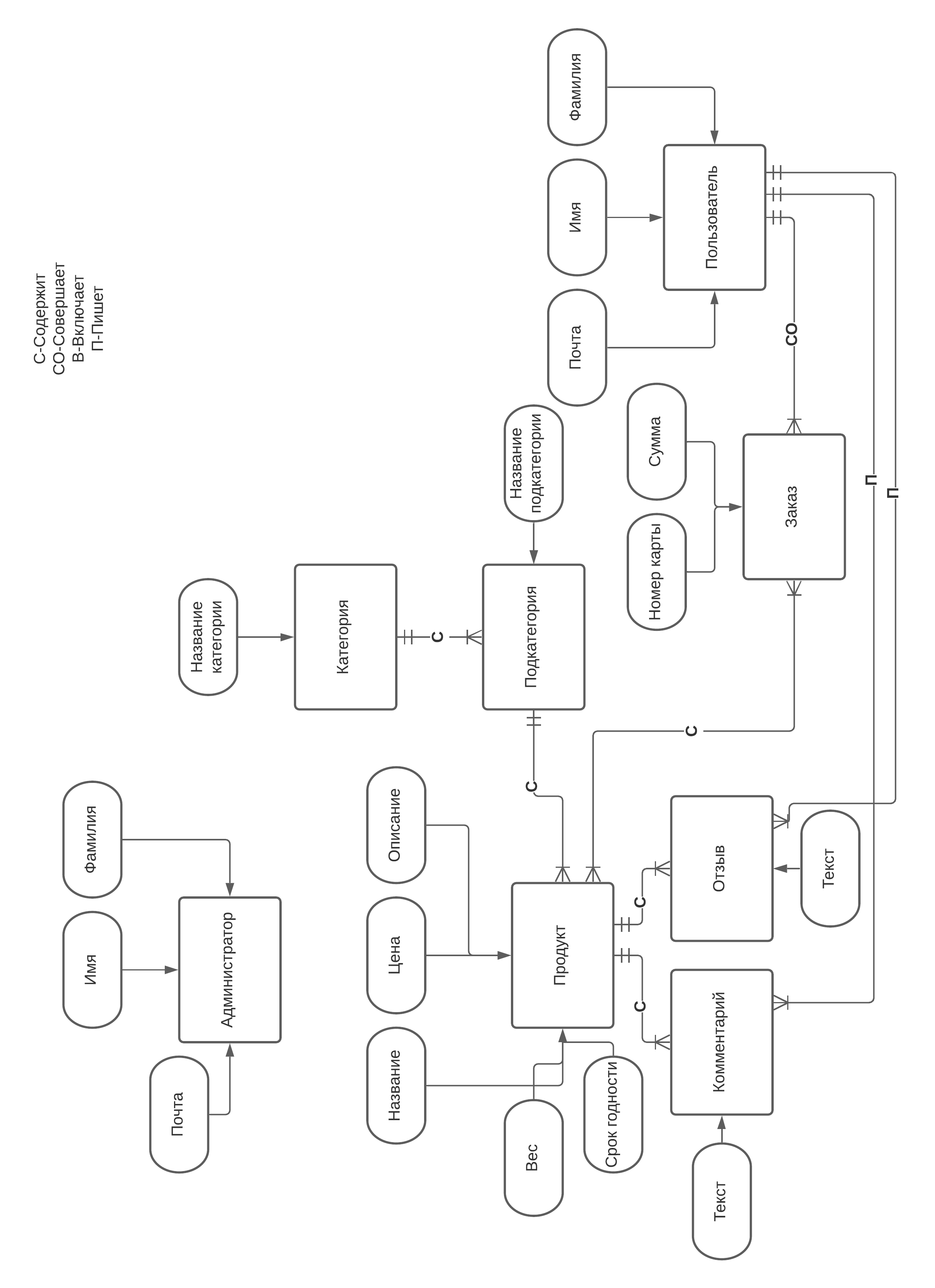
3. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.

4. А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. Язык программирования C#. Классика Computers Science. 4-е изд.: – СПб.: Питер, 2012. – 784 с.

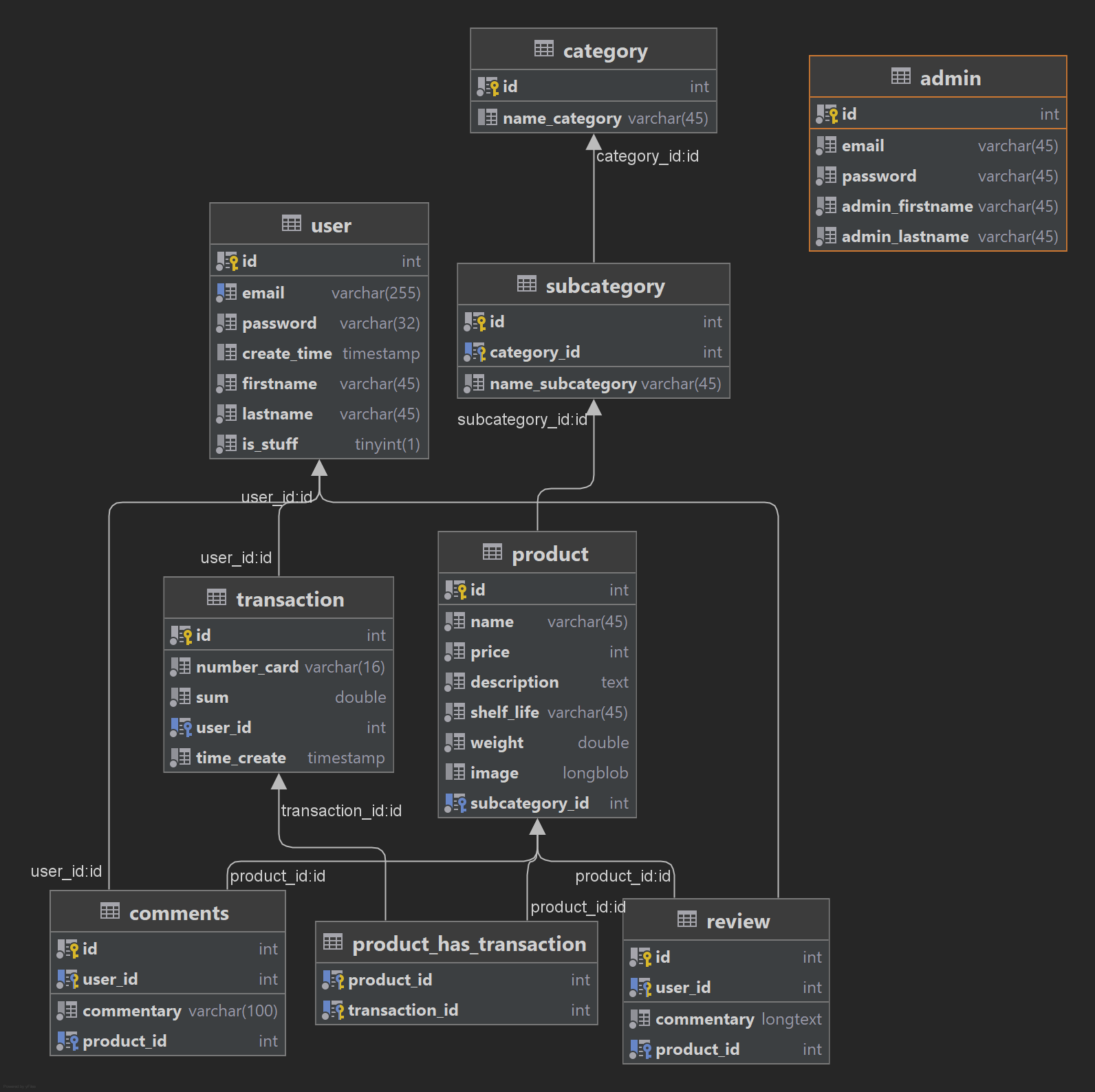
5. Э. Стиллмен, Дж. Грин. Изучаем С#. 2-е изд.: – СПб.: Питер, 2012. – 704 с.

6. Паттон Джефф. Пользовательские истории. Искусство гибкой разработки ПО. – СПб.: Питер, 2019. – 288 с.

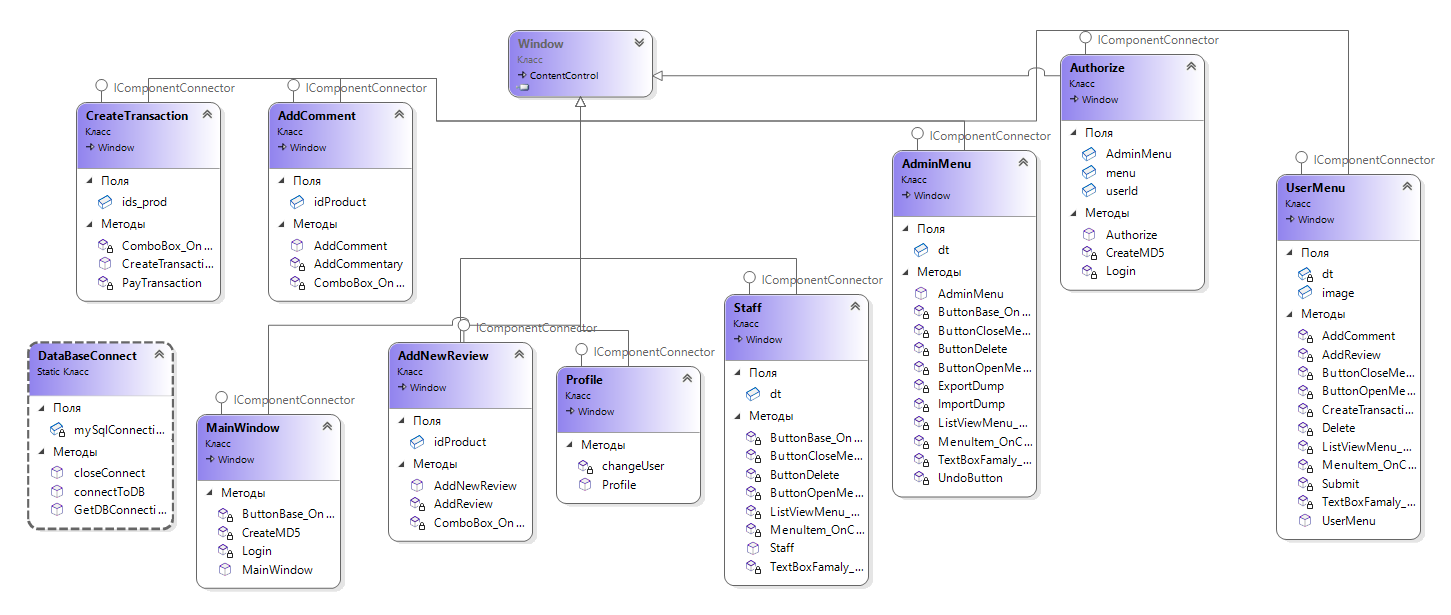
**ПРИЛОЖЕНИЕ А – ER-диаграмма**



# **ПРИЛОЖЕНИЕ Б – Даталогическая схема**



# **ПРИЛОЖЕНИЕ В – UML-диаграмма**

****

**ПРИЛОЖЕНИЕ Г – Код базы данных в MySQL**

CREATE TABLE IF NOT EXISTS `mydb`.`admin` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `email` VARCHAR(45) NOT NULL,  
 `password` VARCHAR(45) NOT NULL,  
 `admin\_firstname` VARCHAR(45) NOT NULL,  
 `admin\_lastname` VARCHAR(45) NOT NULL,  
 PRIMARY KEY (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`category` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `name\_category` VARCHAR(45) NULL DEFAULT NULL,  
 PRIMARY KEY (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`subcategory` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `name\_subcategory` VARCHAR(45) NOT NULL,  
 `category\_id` INT NOT NULL,  
 PRIMARY KEY (`id`, `category\_id`),  
 INDEX `fk\_subcategory\_category1\_idx` (`category\_id` ASC) VISIBLE,  
 CONSTRAINT `fk\_subcategory\_category1`  
 FOREIGN KEY (`category\_id`)  
 REFERENCES `mydb`.`category` (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`product` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `name` VARCHAR(45) NOT NULL,  
 `price` INT NOT NULL,  
 `description` TEXT NOT NULL,  
 `shelf\_life` VARCHAR(45) NOT NULL,  
 `weight` DOUBLE NOT NULL,  
 `image` LONGBLOB NULL DEFAULT NULL,  
 `subcategory\_id` INT NOT NULL,  
 PRIMARY KEY (`id`),  
 INDEX `fk\_product\_subcategory1\_idx` (`subcategory\_id` ASC) VISIBLE,  
 CONSTRAINT `fk\_product\_subcategory1`  
 FOREIGN KEY (`subcategory\_id`)  
 REFERENCES `mydb`.`subcategory` (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`user` (  
 `email` VARCHAR(255) NOT NULL,  
 `password` VARCHAR(32) NOT NULL,  
 `create\_time` TIMESTAMP NULL DEFAULT CURRENT\_TIMESTAMP,  
 `firstname` VARCHAR(45) NOT NULL,  
 `lastname` VARCHAR(45) NOT NULL,  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `is\_stuff` TINYINT(1) NOT NULL,  
 PRIMARY KEY (`id`),  
 UNIQUE INDEX `user\_email\_uindex` (`email` ASC) VISIBLE)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`comments` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `commentary` VARCHAR(100) NOT NULL,  
 `product\_id` INT NOT NULL,  
 `user\_id` INT NOT NULL,  
 PRIMARY KEY (`id`, `user\_id`),  
 INDEX `fk\_comments\_product1\_idx` (`product\_id` ASC) VISIBLE,  
 INDEX `fk\_comments\_user1\_idx` (`user\_id` ASC) VISIBLE,  
 CONSTRAINT `fk\_comments\_product1`  
 FOREIGN KEY (`product\_id`)  
 REFERENCES `mydb`.`product` (`id`),  
 CONSTRAINT `fk\_comments\_user1`  
 FOREIGN KEY (`user\_id`)  
 REFERENCES `mydb`.`user` (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`transaction` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `number\_card` VARCHAR(16) NOT NULL,  
 `sum` DOUBLE NOT NULL,  
 `user\_id` INT NOT NULL,  
 `time\_create` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 PRIMARY KEY (`id`),  
 INDEX `fk\_transaction\_user1\_idx` (`user\_id` ASC) VISIBLE,  
 CONSTRAINT `fk\_transaction\_user1`  
 FOREIGN KEY (`user\_id`)  
 REFERENCES `mydb`.`user` (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`product\_has\_transaction` (  
 `product\_id` INT NOT NULL,  
 `transaction\_id` INT NOT NULL,  
 PRIMARY KEY (`product\_id`, `transaction\_id`),  
 INDEX `fk\_product\_has\_transaction\_transaction1\_idx` (`transaction\_id` ASC) VISIBLE,  
 INDEX `fk\_product\_has\_transaction\_product1\_idx` (`product\_id` ASC) VISIBLE,  
 CONSTRAINT `fk\_product\_has\_transaction\_product1`  
 FOREIGN KEY (`product\_id`)  
 REFERENCES `mydb`.`product` (`id`),  
 CONSTRAINT `fk\_product\_has\_transaction\_transaction1`  
 FOREIGN KEY (`transaction\_id`)  
 REFERENCES `mydb`.`transaction` (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`review` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `commentary` LONGTEXT NOT NULL,  
 `product\_id` INT NOT NULL,  
 `user\_id` INT NOT NULL,  
 PRIMARY KEY (`id`, `user\_id`),  
 INDEX `fk\_review\_product1\_idx` (`product\_id` ASC) VISIBLE,  
 INDEX `fk\_review\_user1\_idx` (`user\_id` ASC) VISIBLE,  
 CONSTRAINT `fk\_review\_product1`  
 FOREIGN KEY (`product\_id`)  
 REFERENCES `mydb`.`product` (`id`),  
 CONSTRAINT `fk\_review\_user1`  
 FOREIGN KEY (`user\_id`)  
 REFERENCES `mydb`.`user` (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`story` (  
 `id` INT NOT NULL AUTO\_INCREMENT,  
 `typeQuery` VARCHAR(255) NOT NULL,  
 `time\_query` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,  
 `table\_name` VARCHAR(25) NOT NULL,  
 `id\_in\_table` INT NOT NULL,  
 PRIMARY KEY (`id`))  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_admin` (  
 `id` INT NOT NULL,  
 `email` VARCHAR(45) NOT NULL,  
 `password` VARCHAR(45) NOT NULL,  
 `admin\_firstname` VARCHAR(45) NOT NULL,  
 `admin\_lastname` VARCHAR(45) NOT NULL,  
 `typeQuery` VARCHAR(20) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_category` (  
 `id` INT NOT NULL,  
 `name\_category` VARCHAR(45) NOT NULL,  
 `typeQuery` VARCHAR(20) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_comments` (  
 `id` INT NOT NULL,  
 `commentary` LONGTEXT NOT NULL,  
 `product\_id` INT NOT NULL,  
 `user\_id` INT NOT NULL,  
 `typeQuery` VARCHAR(20) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_product` (  
 `id` INT NOT NULL,  
 `name` VARCHAR(45) NOT NULL,  
 `price` INT NOT NULL,  
 `description` TEXT NOT NULL,  
 `shelf\_life` VARCHAR(45) NOT NULL,  
 `weight` DOUBLE NOT NULL,  
 `image` LONGBLOB NULL DEFAULT NULL,  
 `subcategory\_id` INT NOT NULL,  
 `typeQuery` VARCHAR(20) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_product\_has\_transaction` (  
 `product\_id` INT NOT NULL,  
 `transaction\_id` INT NOT NULL,  
 `typeQuery` VARCHAR(25) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_review` (  
 `id` INT NOT NULL,  
 `commentary` LONGTEXT NOT NULL,  
 `product\_id` INT NOT NULL,  
 `user\_id` INT NOT NULL,  
 `typeQuery` VARCHAR(20) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_subcategory` (  
 `id` INT NOT NULL,  
 `name\_subcategory` VARCHAR(45) NOT NULL,  
 `category\_id` INT NOT NULL,  
 `typeQuery` VARCHAR(20) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_transaction` (  
 `id` INT NOT NULL,  
 `number\_card` VARCHAR(16) NOT NULL,  
 `sum` DOUBLE NOT NULL,  
 `user\_id` INT NOT NULL,  
 `typeQuery` VARCHAR(25) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
CREATE TABLE IF NOT EXISTS `mydb`.`temp\_user` (  
 `email` VARCHAR(255) NOT NULL,  
 `password` VARCHAR(32) NOT NULL,  
 `firstname` VARCHAR(45) NOT NULL,  
 `lastname` VARCHAR(45) NOT NULL,  
 `id` INT NOT NULL,  
 `is\_stuff` TINYINT(1) NOT NULL,  
 `typeQuery` VARCHAR(25) NOT NULL,  
 `time` TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP)  
  
USE `mydb`;  
  
DELIMITER $$  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteAdmin`  
AFTER DELETE ON `mydb`.`admin`  
FOR EACH ROW  
BEGIN

insert into temp\_admin (id,email, password, admin\_firstname, admin\_lastname, typeQuery)

VALUES (OLD.id,OLD.email, old.password, old.admin\_firstname, old.admin\_lastname, 'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','admin',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newAdmin`  
AFTER INSERT ON `mydb`.`admin`  
FOR EACH ROW  
BEGIN

INSERT INTO temp\_admin (id,email, password, admin\_firstname, admin\_lastname, typeQuery)

VALUES (NEW.id,new.email, NEW.password, NEW.admin\_firstname, new.admin\_lastname, 'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','admin',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateAdmin`  
BEFORE UPDATE ON `mydb`.`admin`  
FOR EACH ROW  
BEGIN

IF OLD.email <> new.email or OLD.password <> NEW.password or

OLD.admin\_firstname <> NEW.admin\_firstname or old.admin\_lastname <> new.admin\_lastname THEN

insert into temp\_admin (id,email, password, admin\_firstname, admin\_lastname, typeQuery)

VALUES (OLD.id,OLD.email, OLD.password, OLD.admin\_firstname, OLD.admin\_lastname,'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','admin',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteCategory`  
AFTER DELETE ON `mydb`.`category`  
FOR EACH ROW  
BEGIN

insert into temp\_category (id,name\_category, typeQuery)

VALUES (OLD.id,OLD.name\_category, 'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','category',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newCategoryToStory`  
AFTER INSERT ON `mydb`.`category`  
FOR EACH ROW  
BEGIN

INSERT INTO temp\_category (id, name\_category, typeQuery)

VALUES (NEW.id, new.name\_category, 'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','category',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateCategory`  
BEFORE UPDATE ON `mydb`.`category`  
FOR EACH ROW  
BEGIN

IF OLD.id <> new.id or OLD.name\_category <> NEW.name\_category THEN

insert into temp\_category (id, name\_category, typeQuery)

VALUES (OLD.id, OLD.name\_category, 'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','category',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteSubCategory`  
AFTER DELETE ON `mydb`.`subcategory`  
FOR EACH ROW  
BEGIN

insert into temp\_subcategory (id, name\_subcategory,category\_id, typeQuery)

VALUES (OLD.id, OLD.name\_subcategory,old.category\_id, 'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','subcategory',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newSubCategory`  
AFTER INSERT ON `mydb`.`subcategory`  
FOR EACH ROW  
BEGIN

insert into temp\_subcategory (id, name\_subcategory,category\_id, typeQuery)

VALUES (new.id, new.name\_subcategory,new.category\_id, 'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','subcategory',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateSubCategory`  
BEFORE UPDATE ON `mydb`.`subcategory`  
FOR EACH ROW  
BEGIN

IF OLD.id <> new.id or OLD.name\_subcategory <> NEW.name\_subcategory or OLD.category\_id<>new.category\_id THEN

insert into temp\_subcategory (id, name\_subcategory,category\_id, typeQuery)

VALUES (OLD.id, OLD.name\_subcategory,OLD.category\_id, 'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','subcategory',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteProduct`  
AFTER DELETE ON `mydb`.`product`  
FOR EACH ROW  
BEGIN

insert into temp\_product (id, name, price, description, shelf\_life, weight,image,subcategory\_id,typeQuery)

VALUES (old.id, old.name, old.price, old.description, old.shelf\_life, old.weight,old.image,old.subcategory\_id,'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','product',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newProduct`  
AFTER INSERT ON `mydb`.`product`  
FOR EACH ROW  
BEGIN

insert into temp\_product (id, name, price, description, shelf\_life, weight,image,subcategory\_id,typeQuery)

VALUES (new.id, new.name, new.price, new.description, new.shelf\_life, new.weight,new.image,new.subcategory\_id,'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','product',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateProduct`  
AFTER UPDATE ON `mydb`.`product`  
FOR EACH ROW  
BEGIN

IF OLD.id <> new.id or OLD.name <>new.name or OLD.price<>new.price or OLD.description<>new.description or old.shelf\_life<>new.shelf\_life or old.image<>new.image or old.subcategory\_id<>new.subcategory\_id

THEN

insert into temp\_product (id, name, price, description, shelf\_life, weight,image,subcategory\_id,typeQuery)

VALUES (OLD.id, OLD.name, OLD.price, OLD.description, OLD.shelf\_life, OLD.weight,OLD.image,OLD.subcategory\_id,'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','product',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteUser`  
AFTER DELETE ON `mydb`.`user`  
FOR EACH ROW  
BEGIN

insert into temp\_user (email, password, firstname, lastname, id, is\_stuff, typeQuery)

VALUES (OLD.email, old.password, old.firstname, old.lastname, old.id, old.is\_stuff, 'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','user',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newUser`  
AFTER INSERT ON `mydb`.`user`  
FOR EACH ROW  
BEGIN

INSERT INTO temp\_user (email, password, firstname, lastname, id, is\_stuff, typeQuery)

VALUES (new.email, NEW.password, NEW.firstname, new.lastname, NEW.id, new.is\_stuff, 'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','user',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateUser`  
BEFORE UPDATE ON `mydb`.`user`  
FOR EACH ROW  
BEGIN

IF OLD.is\_stuff <> new.is\_stuff or OLD.lastname <> new.lastname or OLD.firstname <> NEW.firstname or

OLD.password <> NEW.password or old.email <> new.email THEN

insert into temp\_user (email, password, firstname, lastname, id,is\_stuff, typeQuery)

VALUES (OLD.email, OLD.password, OLD.firstname, OLD.lastname,OLD.id, OLD.is\_stuff, 'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','user',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteComment`  
AFTER DELETE ON `mydb`.`comments`  
FOR EACH ROW  
BEGIN

insert into temp\_comments (id, commentary, product\_id, user\_id,

typeQuery)

VALUES (old.id, old.commentary, old.product\_id, old.user\_id,

'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','comments',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newComment`  
AFTER INSERT ON `mydb`.`comments`  
FOR EACH ROW  
BEGIN

insert into temp\_comments (id, commentary, product\_id, user\_id,

typeQuery)

VALUES (new.id, new.commentary, new.product\_id, new.user\_id,

'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','comments',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateComments`  
AFTER UPDATE ON `mydb`.`comments`  
FOR EACH ROW  
BEGIN

IF OLD.id <> new.id or old.commentary <> new.commentary or

old.user\_id <> new.user\_id or old.product\_id <> new.product\_id THEN

insert into temp\_comments (id, commentary, product\_id, user\_id,

typeQuery)

VALUES (OLD.id, OLD.commentary, OLD.product\_id,

OLD.user\_id, 'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','admin',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteTransaction`  
AFTER DELETE ON `mydb`.`transaction`  
FOR EACH ROW  
BEGIN

insert into temp\_transaction (id, number\_card, sum,user\_id, typeQuery)

VALUES (OLD.id, OLD.number\_card, old.sum,old.user\_id, 'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','transaction',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newTransaction`  
AFTER INSERT ON `mydb`.`transaction`  
FOR EACH ROW  
BEGIN

insert into temp\_transaction (id, number\_card, sum,user\_id, typeQuery)

VALUES (new.id, new.number\_card, new.sum,new.user\_id, 'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','transaction',new.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateTransaction`  
BEFORE UPDATE ON `mydb`.`transaction`  
FOR EACH ROW  
BEGIN

IF OLD.id <> new.id or OLD.number\_card <> NEW.number\_card or OLD.sum <> new.sum or old.user\_id<>new.user\_id THEN

insert into temp\_transaction (id, number\_card, sum,user\_id, typeQuery)

VALUES (OLD.id, OLD.number\_card, OLD.sum,OLD.user\_id, 'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','transaction',OLD.id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteProductHasTrans`  
AFTER DELETE ON `mydb`.`product\_has\_transaction`  
FOR EACH ROW  
BEGIN

insert into temp\_product\_has\_transaction (product\_id, transaction\_id, typeQuery)

VALUES (old.product\_id,OLD.transaction\_id,'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete', 'product\_has\_transaction', OLD.transaction\_id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newProductHasTrans`  
AFTER INSERT ON `mydb`.`product\_has\_transaction`  
FOR EACH ROW  
BEGIN

insert into temp\_product\_has\_transaction (product\_id, transaction\_id,typeQuery)

VALUES (new.product\_id,new.transaction\_id,'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert', 'product\_has\_transaction', NEW.product\_id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateProductHasTrans`  
AFTER UPDATE ON `mydb`.`product\_has\_transaction`  
FOR EACH ROW  
BEGIN

IF OLD.product\_id <> new.product\_id or OLD.transaction\_id <> new.transaction\_id

THEN

insert into temp\_product\_has\_transaction (product\_id, transaction\_id, typeQuery)

VALUES (new.product\_id, new.transaction\_id,'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update', 'product\_has\_transaction', NEW.transaction\_id);

END IF;

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`deleteReview`  
AFTER DELETE ON `mydb`.`review`  
FOR EACH ROW  
BEGIN

insert into temp\_review (id, commentary, product\_id, user\_id,typeQuery)

VALUES (old.id, OLD.commentary,old.product\_id, old.user\_id,'delete');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('delete','review',OLD.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`newReview`  
AFTER INSERT ON `mydb`.`review`  
FOR EACH ROW  
BEGIN

insert into temp\_review (id, commentary, product\_id, user\_id,

typeQuery)

VALUES (new.id, new.commentary, new.product\_id, new.user\_id,

'insert');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('insert','review',NEW.id);

END$$  
  
USE `mydb`$$  
CREATE  
DEFINER=`root`@`localhost`  
TRIGGER `mydb`.`updateReview`  
AFTER UPDATE ON `mydb`.`review`  
FOR EACH ROW  
BEGIN

IF OLD.id <> new.id or old.commentary <> new.commentary or

old.user\_id <> new.user\_id or old.product\_id <> new.product\_id THEN

insert into temp\_review (id, commentary, product\_id, user\_id,

typeQuery)

VALUES (OLD.id, OLD.commentary, OLD.product\_id, OLD.user\_id,

'update');

insert into story (typeQuery, table\_name, id\_in\_table) VALUES ('update','review',OLD.id);

END IF;

END$$

**ПРИЛОЖЕНИЕ Д – Код клиентского интерфейса**

using System;

using System.Data;

using System.Windows;

using System.Windows.Controls;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class AddComment : Window

{

public int idProduct;

public AddComment()

{

DataTable dt;

InitializeComponent();

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT name FROM product", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

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

{

ComboBox.Items.Add(dt.Rows[i]["name"]);

}

}

}

private void AddCommentary(object sender, RoutedEventArgs e)

{

string firstname = String.Empty;

string lastname = String.Empty;

using (var conn = DataBaseConnect.GetDBConnection())

{

conn.Open();

MySqlCommand commSel = new MySqlCommand("SELECT \* FROM user WHERE id=@ids", conn);

commSel.Parameters.AddWithValue("@ids", Authorize.userId.ToString());

using (MySqlDataReader reader = commSel.ExecuteReader())

{

if (reader.Read())

{

firstname = reader.GetString(3);

lastname = reader.GetString(4);

}

}

}

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

MySqlCommand comm = conn.CreateCommand();

comm.CommandText = "INSERT INTO comments(commentary,product\_id,user\_id) VALUES(@commentary,@product\_id,@user\_id)";

comm.Parameters.AddWithValue("@commentary", comment.Text);

comm.Parameters.AddWithValue("@product\_id", idProduct);

comm.Parameters.AddWithValue("@user\_id", Authorize.userId.ToString());

comm.ExecuteNonQuery();

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

private void ComboBox\_OnSelectionChanged(object sender, SelectionChangedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

conn.Open();

MySqlCommand commSel = new MySqlCommand("SELECT id FROM product WHERE name=@name", conn);

commSel.Parameters.AddWithValue("@name", (sender as ComboBox).SelectedItem as string);

using (MySqlDataReader reader = commSel.ExecuteReader())

{

if (reader.Read())

{

idProduct = reader.GetInt32(0);

}

}

}

}

}

}

using System;

using System.Data;

using System.Windows;

using System.Windows.Controls;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class AddNewReview : Window

{

public int idProduct;

public AddNewReview()

{

DataTable dt;

InitializeComponent();

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT name FROM product", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

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

{

ComboBox.Items.Add(dt.Rows[i]["name"]);

}

}

}

private void AddReview(object sender, RoutedEventArgs e)

{

string firstname = String.Empty;

string lastname = String.Empty;

using (var conn = DataBaseConnect.GetDBConnection())

{

conn.Open();

MySqlCommand commSel = new MySqlCommand("SELECT \* FROM user WHERE id=@ids", conn);

commSel.Parameters.AddWithValue("@ids", Authorize.userId.ToString());

using (MySqlDataReader reader = commSel.ExecuteReader())

{

if (reader.Read())

{

firstname = reader.GetString(3);

lastname = reader.GetString(4);

}

}

}

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

MySqlCommand comm = conn.CreateCommand();

comm.CommandText = "INSERT INTO review(commentary,product\_id,user\_id) VALUES(@commentary,@product\_id,@user\_id)";

comm.Parameters.AddWithValue("@commentary", review.Text);

comm.Parameters.AddWithValue("@product\_id", idProduct);

comm.Parameters.AddWithValue("@user\_id", Authorize.userId.ToString());

comm.ExecuteNonQuery();

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

private void ComboBox\_OnSelectionChanged(object sender, SelectionChangedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

conn.Open();

MySqlCommand commSel = new MySqlCommand("SELECT id FROM product WHERE name=@name", conn);

commSel.Parameters.AddWithValue("@name", (sender as ComboBox).SelectedItem as string);

using (MySqlDataReader reader = commSel.ExecuteReader())

{

if (reader.Read())

{

idProduct = reader.GetInt32(0);

}

}

}

}

}

}

using System;

using System.Data;

using System.Diagnostics;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class AdminMenu : Window

{

public DataTable dt = new DataTable();

public AdminMenu()

{

InitializeComponent();

}

private void ButtonOpenMenu\_Click(object sender, RoutedEventArgs e)

{

ButtonCloseMenu.Visibility = Visibility.Visible;

ButtonOpenMenu.Visibility = Visibility.Collapsed;

}

private void ButtonCloseMenu\_Click(object sender, RoutedEventArgs e)

{

ButtonCloseMenu.Visibility = Visibility.Collapsed;

ButtonOpenMenu.Visibility = Visibility.Visible;

}

private void ListViewMenu\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

UserControl usc = null;

GridMain.Children.Clear();

switch (((ListViewItem)((ListView)sender).SelectedItem).Name)

{

case "UsersGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM user", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "TransactionGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM transaction", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ProductGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM product", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "CommentsGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM comments", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "CategoryGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM category", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "SubcategoryGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM subcategory", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ReviewGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM review", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ModerateGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM user WHERE is\_stuff=1", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

default:

break;

}

}

private void ButtonBase\_OnClick(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

if (ListViewMenu.SelectedIndex == 0)

{

string sql = "select \* from user";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 1)

{

string sql = "select \* from user where is\_stuff=1";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 2)

{

string sql = "select \* from transaction";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 3)

{

string sql = "select \* from product";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 4)

{

string sql = "select \* from category";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 5)

{

string sql = "select \* from subcategory";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 6)

{

string sql = "select \* from comments";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 7)

{

string sql = "select \* from review";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

private void ButtonDelete(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

var row = grid.SelectedItem as DataRowView;

if (row == null)

return;

row.Delete();

}

}

private void ExportDump(object sender, RoutedEventArgs e)

{

string file = @"C:\Users\Roman Bulygin\RiderProjects\WpfApplication1\backup.sql";

using (var conn = DataBaseConnect.GetDBConnection())

{

using (MySqlCommand cmd = new MySqlCommand())

{

using (MySqlBackup mb = new MySqlBackup(cmd))

{

try

{

cmd.Connection = conn;

conn.Open();

mb.ExportToFile(file);

conn.Close();

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

}

}

private void ImportDump(object sender, RoutedEventArgs e)

{

string file = @"C:\Users\Roman Bulygin\RiderProjects\WpfApplication1\backup.sql";

using (var conn = DataBaseConnect.GetDBConnection())

{

using (MySqlCommand cmd = new MySqlCommand())

{

using (MySqlBackup mb = new MySqlBackup(cmd))

{

try

{

cmd.Connection = conn;

conn.Open();

mb.ImportFromFile(file);

conn.Close();

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

}

}

private void MenuItem\_OnClick(object sender, RoutedEventArgs e)

{

Authorize.AdminMenu.Close();

this.Close();

}

private void UndoButton(object sender, RoutedEventArgs e)

{

int maxid = 0;

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

MySqlCommand comm = new MySqlCommand("Select max(`id`) as `maxid` from story", conn);

using (MySqlDataReader reader = comm.ExecuteReader())

{

if (reader.Read())

{

maxid = reader.GetInt32(0);

}

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

try

{

string typeQuery;

string tableName;

int id\_in\_table;

MySqlCommand comm = new MySqlCommand("SELECT \* FROM story WHERE id=@ids", conn);

comm.Parameters.AddWithValue("@ids", maxid);

using (MySqlDataReader reader = comm.ExecuteReader())

{

if (reader.Read())

{

typeQuery = reader.GetString(1);

tableName = reader.GetString(3);

id\_in\_table = reader.GetInt32(4);

reader.Close();

if (typeQuery.Equals("insert"))

{

if (tableName.Equals("product\_has\_transaction"))

{

int maxtransid = 0;

comm = new MySqlCommand("SELECT id\_in\_table FROM story WHERE id=@ids", conn);

comm.Parameters.AddWithValue("@ids", maxid-1);

using (MySqlDataReader idread = comm.ExecuteReader())

{

if (idread.Read())

{

maxtransid = idread.GetInt32(0);

}

}

comm = new MySqlCommand("select \* from product\_has\_transaction inner join transaction t on product\_has\_transaction.transaction\_id = t.id; delete from product\_has\_transaction where transaction\_id = @maxid; delete from transaction where id = @maxid;", conn);

comm.Parameters.AddWithValue("@maxid", maxtransid);

comm.ExecuteNonQuery();

}

else

{

comm = new MySqlCommand($@"DELETE FROM {tableName} WHERE id={id\_in\_table}", conn);

comm.Parameters.AddWithValue("@id\_in\_table", id\_in\_table);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("category"))

{

string name\_category = String.Empty;

comm = new MySqlCommand($@"SELECT \* FROM temp\_category WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

name\_category = catrReader.GetString(1);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO category (id, name\_category) values (@idtab,@namecatr)",conn);

comm.Parameters.AddWithValue("@idtab", id\_in\_table);

comm.Parameters.AddWithValue("@namecatr", name\_category);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("subcategory"))

{

string name\_subcategory = String.Empty;

int categoryid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_subcategory WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

name\_subcategory = catrReader.GetString(1);

categoryid = catrReader.GetInt32(2);

}

catrReader.Close();

MessageBox.Show(id\_in\_table + ", " + name\_subcategory + ", " + categoryid);

comm = new MySqlCommand("INSERT INTO subcategory (id, name\_subcategory,category\_id) values (@idtab,@namesubcatr,@idcat)",conn);

comm.Parameters.AddWithValue("@idtab", id\_in\_table);

comm.Parameters.AddWithValue("@namesubcatr", name\_subcategory);

comm.Parameters.AddWithValue("@idcat", categoryid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("user"))

{

string email = String.Empty;

string pass = String.Empty;

string create\_time = String.Empty;

string firstname = String.Empty;

string lastname = String.Empty;

comm = new MySqlCommand($@"SELECT \* FROM temp\_user WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

email = catrReader.GetString(0);

pass = catrReader.GetString(1);

firstname = catrReader.GetString(3);

lastname = catrReader.GetString(4);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO user (email,password,firstname,lastname,id,is\_stuff) values (@email,@password,@firstname,@lastname,@id,0)",conn);

comm.Parameters.AddWithValue("@email", email);

comm.Parameters.AddWithValue("@password", pass);

comm.Parameters.AddWithValue("@firstname", firstname);

comm.Parameters.AddWithValue("@lastname", lastname);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("admin"))

{

string email = String.Empty;

string pass = String.Empty;

string firstname = String.Empty;

string lastname = String.Empty;

comm = new MySqlCommand($@"SELECT \* FROM temp\_admin WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

email = catrReader.GetString(1);

pass = catrReader.GetString(2);

firstname = catrReader.GetString(3);

lastname = catrReader.GetString(4);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO admin (id,email,password,admin\_firstname,admin\_lastname) values (@id,@email,@password,@firstname,@lastname)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@email", email);

comm.Parameters.AddWithValue("@password", pass);

comm.Parameters.AddWithValue("@firstname", firstname);

comm.Parameters.AddWithValue("@lastname", lastname);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("comments"))

{

string commentary = String.Empty;

int productid = 0;

int userid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_comments WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

commentary = catrReader.GetString(1);

productid = catrReader.GetInt32(2);

userid = catrReader.GetInt32(3);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO comments (id,commentary,product\_id,user\_id) values (@id,@commentary,@productid,@userid)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@commentary", commentary);

comm.Parameters.AddWithValue("@productid", productid);

comm.Parameters.AddWithValue("@userid", userid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("review"))

{

string commentary = String.Empty;

int productid = 0;

int userid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_review WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

commentary = catrReader.GetString(1);

productid = catrReader.GetInt32(2);

userid = catrReader.GetInt32(3);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO review (id,commentary,product\_id,user\_id) values (@id,@commentary,@productid,@userid)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@commentary", commentary);

comm.Parameters.AddWithValue("@productid", productid);

comm.Parameters.AddWithValue("@userid", userid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("product"))

{

string name = String.Empty;

int price = 0;

string desc = String.Empty;

string shelflife = String.Empty;

double weight = 0;

int subcategoryid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_product WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

name = catrReader.GetString(1);

price = catrReader.GetInt32(2);

desc = catrReader.GetString(3);

shelflife = catrReader.GetString(4);

weight = catrReader.GetDouble(5);

subcategoryid = catrReader.GetInt32(7);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO product (id,name,price,description,shelf\_life,weight,subcategory\_id) values (@id,@name,@price,@description,@shelf\_life,@weight,@subcategory\_id)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@name", name);

comm.Parameters.AddWithValue("@price", price);

comm.Parameters.AddWithValue("@description", desc);

comm.Parameters.AddWithValue("@shelf\_life", shelflife);

comm.Parameters.AddWithValue("@weight", weight);

comm.Parameters.AddWithValue("@subcategory\_id", subcategoryid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("delete") & tableName.Equals("transaction"))

{

string numbercard = String.Empty;

double sum = 0;

int userid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_transaction WHERE id={id\_in\_table}", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

numbercard = catrReader.GetString(1);

sum = catrReader.GetDouble(2);

userid = catrReader.GetInt32(3);

}

catrReader.Close();

comm = new MySqlCommand("INSERT INTO transaction (id,number\_card,sum,user\_id) values (@id,@number\_card,@sum,@userid)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@number\_card", numbercard);

comm.Parameters.AddWithValue("@sum", sum);

comm.Parameters.AddWithValue("@userid", userid);

comm.ExecuteNonQuery();

comm = new MySqlCommand($@"SELECT \* FROM temp\_product\_has\_transaction WHERE transaction\_id={id\_in\_table}", conn);

using (MySqlDataReader currentReader = comm.ExecuteReader())

{

int productid = 0;

int transactionid = 0;

if (currentReader.Read())

{

productid = currentReader.GetInt32(0);

transactionid = currentReader.GetInt32(1);

}

currentReader.Close();

comm = new MySqlCommand("INSERT INTO product\_has\_transaction (product\_id,transaction\_id) values (@productid,@transactionid)",conn);

comm.Parameters.AddWithValue("@productid", productid);

comm.Parameters.AddWithValue("@transactionid", transactionid);

comm.ExecuteNonQuery();

}

}

}

if (typeQuery.Equals("update") & tableName.Equals("category"))

{

string name\_category = String.Empty;

comm = new MySqlCommand($@"SELECT \* FROM temp\_category WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

name\_category = catrReader.GetString(1);

}

catrReader.Close();

comm = new MySqlCommand("update category set name\_category=@namecatr where id=@idtab",conn);

comm.Parameters.AddWithValue("@idtab", id\_in\_table);

comm.Parameters.AddWithValue("@namecatr", name\_category);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("update") & tableName.Equals("subcategory"))

{

string name\_subcategory = String.Empty;

int categoryid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_subcategory WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

name\_subcategory = catrReader.GetString(1);

categoryid = catrReader.GetInt32(2);

}

catrReader.Close();

comm = new MySqlCommand("UPDATE subcategory set name\_subcategory=@namesubcatr,category\_id=@idcat where id=@idtab",conn);

comm.Parameters.AddWithValue("@idtab", id\_in\_table);

comm.Parameters.AddWithValue("@namesubcatr", name\_subcategory);

comm.Parameters.AddWithValue("@idcat", categoryid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("update") & tableName.Equals("user"))

{

string email = String.Empty;

string pass = String.Empty;

string create\_time = String.Empty;

string firstname = String.Empty;

string lastname = String.Empty;

comm = new MySqlCommand($@"SELECT \* FROM temp\_user WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

email = catrReader.GetString(0);

pass = catrReader.GetString(1);

firstname = catrReader.GetString(3);

lastname = catrReader.GetString(4);

}

catrReader.Close();

comm = new MySqlCommand("UPDATE user set email=@email,password=@password,firstname=@firstname,lastname=@lastname,is\_stuff=0 where id=@id",conn);

comm.Parameters.AddWithValue("@email", email);

comm.Parameters.AddWithValue("@password", pass);

comm.Parameters.AddWithValue("@firstname", firstname);

comm.Parameters.AddWithValue("@lastname", lastname);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("update") & tableName.Equals("admin"))

{

string email = String.Empty;

string pass = String.Empty;

string firstname = String.Empty;

string lastname = String.Empty;

comm = new MySqlCommand($@"SELECT \* FROM temp\_admin WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

email = catrReader.GetString(1);

pass = catrReader.GetString(2);

firstname = catrReader.GetString(3);

lastname = catrReader.GetString(4);

}

catrReader.Close();

comm = new MySqlCommand("UPDATE admin set email=@email,password=@password,admin\_firstname=@firstname,admin\_lastname=@lastname where id=@id",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@email", email);

comm.Parameters.AddWithValue("@password", pass);

comm.Parameters.AddWithValue("@firstname", firstname);

comm.Parameters.AddWithValue("@lastname", lastname);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("update") & tableName.Equals("comments"))

{

string commentary = String.Empty;

int productid = 0;

int userid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_comments WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

commentary = catrReader.GetString(1);

productid = catrReader.GetInt32(2);

userid = catrReader.GetInt32(3);

}

catrReader.Close();

comm = new MySqlCommand("UPDATE comments set commentary=@commentary,product\_id=@productid,user\_id=@userid where id=@id",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@commentary", commentary);

comm.Parameters.AddWithValue("@productid", productid);

comm.Parameters.AddWithValue("@userid", userid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("update") & tableName.Equals("review"))

{

string commentary = String.Empty;

int productid = 0;

int userid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_review WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

commentary = catrReader.GetString(1);

productid = catrReader.GetInt32(2);

userid = catrReader.GetInt32(3);

}

catrReader.Close();

comm = new MySqlCommand("UPDATE review set commentary=@commentary,product\_id=@productid,user\_id=@userid where id=@id)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@commentary", commentary);

comm.Parameters.AddWithValue("@productid", productid);

comm.Parameters.AddWithValue("@userid", userid);

comm.ExecuteNonQuery();

}

}

if (typeQuery.Equals("update") & tableName.Equals("product"))

{

string name = String.Empty;

int price = 0;

string desc = String.Empty;

string shelflife = String.Empty;

double weight = 0;

int subcategoryid = 0;

comm = new MySqlCommand($@"SELECT \* FROM temp\_product WHERE id={id\_in\_table};", conn);

using (MySqlDataReader catrReader = comm.ExecuteReader())

{

if (catrReader.Read())

{

name = catrReader.GetString(1);

price = catrReader.GetInt32(2);

desc = catrReader.GetString(3);

shelflife = catrReader.GetString(4);

weight = catrReader.GetDouble(5);

subcategoryid = catrReader.GetInt32(7);

}

catrReader.Close();

comm = new MySqlCommand("UPDATE product set name=@name,price=@price,description=@description,shelf\_life=@shelf\_life,weight=@weight,subcategory\_id=@subcategory\_id where id=@id)",conn);

comm.Parameters.AddWithValue("@id", id\_in\_table);

comm.Parameters.AddWithValue("@name", name);

comm.Parameters.AddWithValue("@price", price);

comm.Parameters.AddWithValue("@description", desc);

comm.Parameters.AddWithValue("@shelf\_life", shelflife);

comm.Parameters.AddWithValue("@weight", weight);

comm.Parameters.AddWithValue("@subcategory\_id", subcategoryid);

comm.ExecuteNonQuery();

}

}

}

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

private void TextBoxFamaly\_OnTextChanged(object sender, TextChangedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

string textSearch = textBoxFamaly.Text;

if (ListViewMenu.SelectedIndex == 0)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM user WHERE concat(email,firstname,lastname,id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 1)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM user WHERE is\_stuff=1 and concat(email,firstname,lastname,id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 2)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM transaction WHERE concat(number\_card,sum,user\_id,id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 3)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM product WHERE concat(id,name,price,description,shelf\_life,weight,subcategory\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 4)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM category WHERE concat(id,name\_category) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 5)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM subcategory WHERE concat(id,name\_subcategory,category\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 6)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM comments WHERE concat(id,commentary,product\_id,user\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 7)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM review WHERE concat(id,commentary,product\_id,user\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

}

}

}

}

using System;

using System.Text;

using System.Text.RegularExpressions;

using System.Windows;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class Authorize : Window

{

public static int userId;

public static UserMenu menu;

public static AdminMenu AdminMenu;

public Authorize()

{

InitializeComponent();

}

private static string CreateMD5(string input)

{

using (System.Security.Cryptography.MD5 md5 = System.Security.Cryptography.MD5.Create())

{

byte[] inputBytes = Encoding.ASCII.GetBytes(input);

byte[] hashBytes = md5.ComputeHash(inputBytes);

StringBuilder sb = new StringBuilder();

for (int i = 0; i < hashBytes.Length; i++)

{

sb.Append(hashBytes[i].ToString("X2"));

}

return sb.ToString();

}

}

private void Login(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

string mail = email.Text;

Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");

Match match = regex.Match(mail);

if (match.Success)

{

MySqlCommand comm = new MySqlCommand("SELECT \* FROM user WHERE email=@mail", conn);

MySqlCommand commSearchAdmin = new MySqlCommand("SELECT \* FROM admin WHERE email=@mail", conn);

MySqlCommand commSearchStaff = new MySqlCommand("SELECT \* FROM user WHERE email=@mail and is\_stuff=1", conn);

comm.Parameters.AddWithValue("@mail", email.Text);

using (MySqlDataReader reader = comm.ExecuteReader())

{

String pass = CreateMD5(password.Password);

if (reader.Read())

{

if (email.Text.Equals(reader.GetValue(0)) & pass.Equals(reader.GetValue(1)))

{

userId = (int) reader.GetValue(5);

menu = new UserMenu();

menu.Show();

this.Close();

}

}

}

commSearchAdmin.Parameters.AddWithValue("@mail", email.Text);

using (MySqlDataReader readerAdmin = commSearchAdmin.ExecuteReader())

{

if (readerAdmin.Read())

{

if (email.Text.Equals(readerAdmin.GetValue(1)) & password.Password.Equals(readerAdmin.GetValue(2)))

{

AdminMenu = new AdminMenu();

AdminMenu.Show();

this.Close();

}

}

}

commSearchStaff.Parameters.AddWithValue("@mail", email.Text);

using (MySqlDataReader readerStaff = commSearchStaff.ExecuteReader())

{

String pass = CreateMD5(password.Password);

if (readerStaff.Read())

{

if (email.Text.Equals(readerStaff.GetString(0)) & pass.Equals(readerStaff.GetString(1)))

{

Staff staff = new Staff();

staff.Show();

this.Close();

}

}

}

}

else

{

MessageBox.Show("Email or password not valid");

}

}

catch (Exception exception)

{

MessageBox.Show("Email or password not valid");

throw;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Text.RegularExpressions;

using System.Windows;

using System.Windows.Controls;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class CreateTransaction : Window

{

public CreateTransaction()

{

DataTable dt;

InitializeComponent();

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT name FROM product", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

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

{

ComboBox.Items.Add(dt.Rows[i]["name"]);

}

}

}

public static List<int> ids\_prod = new List<int>();

private void ComboBox\_OnSelectionChanged(object sender, SelectionChangedEventArgs e)

{

list\_product.Content=(sender as ComboBox).SelectedItem as string+", "+list\_product.Content;

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

MySqlCommand comm = conn.CreateCommand();

comm.CommandText = "SELECT price,id FROM product WHERE name=@pname";

comm.Parameters.AddWithValue("@pname",(sender as ComboBox).SelectedItem as string);

using (MySqlDataReader reader = comm.ExecuteReader())

{

if (reader.Read())

{

FinalSum.Content = Int32.Parse(FinalSum.Content.ToString())+reader.GetInt32(0);

ids\_prod.Add(reader.GetInt32(1));

}

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

}

private void PayTransaction(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

Regex regex = new Regex(@"^[0-9]+$");

if (regex.IsMatch(number\_card.Text) & number\_card.Text.Length==16)

{

try

{

conn.Open();

MySqlCommand comm = conn.CreateCommand();

comm.CommandText = "INSERT INTO transaction(number\_card,sum,user\_id) VALUES(@number\_card,@sum,@user\_id)";

comm.Parameters.AddWithValue("@number\_card", number\_card.Text);

comm.Parameters.AddWithValue("@sum", Double.Parse(FinalSum.Content.ToString()));

comm.Parameters.AddWithValue("@user\_id", Authorize.userId);

comm.ExecuteNonQuery();

int id\_trans = 0;

comm.CommandText = "SELECT MAX(id) as maxid FROM transaction";

using (MySqlDataReader reader = comm.ExecuteReader())

{

if (reader.Read())

{

id\_trans = reader.GetInt32(0);

}

}

comm.CommandText = "INSERT INTO product\_has\_transaction(product\_id,transaction\_id) VALUES(@product\_id,@transaction\_id)";

for (int i = 0; i < ids\_prod.Count; i++)

{

comm.Parameters.AddWithValue("@product\_id",ids\_prod.ToArray()[i]);

comm.Parameters.AddWithValue("@transaction\_id", id\_trans);

comm.ExecuteNonQuery();

comm.Parameters.Clear();

}

MessageBox.Show("Transaction success");

this.Close();

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

}

}

else

{

MessageBox.Show("Number card not found");

}

}

}

}

}

using System;

using System.Windows;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public static class DataBaseConnect

{

private static MySqlConnection mySqlConnection;

public static MySqlConnection GetDBConnection()

{

String connString = "Server=localhost;Database=mydb;port=3306;User Id=root;password=1337;";

MySqlConnection conn = new MySqlConnection(connString);

return conn;

}

public static void connectToDB()

{

string \_connectStr = "Server=localhost;Database=mydb;port=3306;User Id=root;password=1337;";

try

{

mySqlConnection = new MySqlConnection(\_connectStr);

mySqlConnection.Open();

MessageBox.Show("Success");

}

catch (Exception e)

{

MessageBox.Show("Error: not connect to database");

throw;

}

}

public static void closeConnect()

{

try

{

mySqlConnection.Close();

MessageBox.Show("Close");

}

catch (Exception e)

{

MessageBox.Show("Error: connect not close");

throw;

}

}

}

}

using System;

using System.Text.RegularExpressions;

using System.Windows;

using MySql.Data.MySqlClient;

using System.Text;

namespace WpfApplication1

{

public partial class MainWindow

{

public MainWindow()

{

InitializeComponent();

}

private static string CreateMD5(string input)

{

using (System.Security.Cryptography.MD5 md5 = System.Security.Cryptography.MD5.Create())

{

byte[] inputBytes = Encoding.ASCII.GetBytes(input);

byte[] hashBytes = md5.ComputeHash(inputBytes);

StringBuilder sb = new StringBuilder();

for (int i = 0; i < hashBytes.Length; i++)

{

sb.Append(hashBytes[i].ToString("X2"));

}

return sb.ToString();

}

}

private void ButtonBase\_OnClick(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

string mail = email.Text;

Regex regex = new Regex(@"^([\w\.\-]+)@([\w\-]+)((\.(\w){2,3})+)$");

Match match = regex.Match(mail);

if (match.Success)

{

try

{

MySqlCommand comm = conn.CreateCommand();

comm.CommandText = "INSERT INTO user(firstname,lastname,email,password,is\_stuff) VALUES(@firstname, @lastname,@email,@password,0)";

comm.Parameters.AddWithValue("@firstname", firstname.Text);

comm.Parameters.AddWithValue("@lastname", lastname.Text);

comm.Parameters.AddWithValue("@email", email.Text);

comm.Parameters.AddWithValue("@password", CreateMD5(password.Password));

comm.ExecuteNonQuery();

MessageBox.Show("Registry success");

}

catch (MySqlException exception)

{

if (exception.Code == 0)

{

MessageBox.Show("Error: this mail is already in use");

}

throw;

}

}

else

{

MessageBox.Show("Email not valid");

}

}

catch (Exception exception)

{

throw;

}

}

}

private void Login(object sender, RoutedEventArgs e)

{

Authorize authorize = new Authorize();

authorize.Show();

this.Close();

}

}

}

using System;

using System.Windows;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class Profile : Window

{

public Profile()

{

InitializeComponent();

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

MySqlCommand comm = new MySqlCommand("SELECT \* FROM user WHERE id=@ids", conn);

comm.Parameters.AddWithValue("@ids", Authorize.userId.ToString());

using (MySqlDataReader reader = comm.ExecuteReader())

{

if (reader.Read())

{

id.Content = reader.GetValue(5).ToString();

firstname.Content = reader.GetValue(3).ToString();

email.Content = reader.GetValue(0).ToString();

date.Content = reader.GetValue(2).ToString();

lastname.Content = reader.GetValue(4).ToString();

}

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

private void changeUser(object sender, RoutedEventArgs e)

{

Authorize.menu.Close();

Authorize authorize = new Authorize();

authorize.Show();

this.Close();

}

}

}

using System;

using System.Data;

using System.Diagnostics;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class Staff : Window

{

public DataTable dt = new DataTable();

public Staff()

{

InitializeComponent();

}

private void ButtonOpenMenu\_Click(object sender, RoutedEventArgs e)

{

ButtonCloseMenu.Visibility = Visibility.Visible;

ButtonOpenMenu.Visibility = Visibility.Collapsed;

}

private void ButtonCloseMenu\_Click(object sender, RoutedEventArgs e)

{

ButtonCloseMenu.Visibility = Visibility.Collapsed;

ButtonOpenMenu.Visibility = Visibility.Visible;

}

private void ListViewMenu\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

UserControl usc = null;

GridMain.Children.Clear();

switch (((ListViewItem)((ListView)sender).SelectedItem).Name)

{

case "TransactionGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM transaction", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ProductGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM product", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "CommentsGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM comments", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "CategoryGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM category", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "SubcategoryGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM subcategory", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ReviewGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("SELECT \* FROM review", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

default:

break;

}

}

private void ButtonBase\_OnClick(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

if (ListViewMenu.SelectedIndex == 0)

{

string sql = "select \* from transaction";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 1)

{

string sql = "select \* from product";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 2)

{

string sql = "select \* from category";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 3)

{

string sql = "select \* from subcategory";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 4)

{

string sql = "select \* from comments";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 5)

{

string sql = "select \* from review";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

}

private void ButtonDelete(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

var row = grid.SelectedItem as DataRowView;

if (row == null)

return;

row.Delete();

}

}

private void MenuItem\_OnClick(object sender, RoutedEventArgs e)

{

Authorize.AdminMenu.Close();

this.Close();

}

private void TextBoxFamaly\_OnTextChanged(object sender, TextChangedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

string textSearch = textBoxFamaly.Text;

if (ListViewMenu.SelectedIndex == 0)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM transaction WHERE concat(number\_card,sum,user\_id,id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 1)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM product WHERE concat(id,name,price,description,shelf\_life,weight,subcategory\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 2)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM category WHERE concat(id,name\_category) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 3)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM subcategory WHERE concat(id,name\_subcategory,category\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 4)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM comments WHERE concat(id,commentary,product\_id,user\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 5)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM review WHERE concat(id,commentary,product\_id,user\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

}

}

}

}

using System;

using System.Data;

using System.Data.Common;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using MySql.Data.MySqlClient;

namespace WpfApplication1

{

public partial class UserMenu : Window

{

public ImageSource image = new BitmapImage();

private DataTable dt = new DataTable();

public UserMenu()

{

InitializeComponent();

}

private void ListViewMenu\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

UserControl usc = null;

GridMain.Children.Clear();

switch (((ListViewItem) ((ListView) sender).SelectedItem).Name)

{

case "Profile":

{

Profile profile = new Profile();

profile.Focus();

profile.Show();

break;

}

case "TransactionGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("select number\_card, sum, time\_create, transaction\_id, product.name from transaction inner join product\_has\_transaction on transaction.id = product\_has\_transaction.transaction\_id inner join product on product\_has\_transaction.product\_id = product.id WHERE user\_id=@id", conn))

{

cmdSel.Parameters.AddWithValue("@id", Authorize.userId.ToString());

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ProductGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("select name,price,description,shelf\_life,weight,subcategory.name\_subcategory,category.name\_category from product inner JOIN subcategory on product.subcategory\_id = subcategory.id inner join category on subcategory.category\_id = category.id", conn))

{

MySqlCommand commSel = new MySqlCommand("SELECT image FROM product", conn);

commSel.Parameters.AddWithValue("@ids", Authorize.userId.ToString());

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "CommentGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("select commentary,user.firstname,user.lastname,product.name from comments inner JOIN user on comments.user\_id = user.id inner join product on comments.product\_id = product.id WHERE user\_id=@id", conn))

{

cmdSel.Parameters.AddWithValue("@id", Authorize.userId.ToString());

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

case "ReviewGroup":

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

using (MySqlCommand cmdSel = new MySqlCommand("select commentary,user.firstname,user.lastname,product.name from review inner JOIN user on review.user\_id = user.id inner join product on review.product\_id = product.id WHERE user\_id=@id", conn))

{

cmdSel.Parameters.AddWithValue("@id", Authorize.userId.ToString());

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

catch (Exception exception)

{

MessageBox.Show(exception.Message);

throw;

}

}

break;

}

}

}

private void ButtonOpenMenu\_Click(object sender, RoutedEventArgs e)

{

ButtonCloseMenu.Visibility = Visibility.Visible;

ButtonOpenMenu.Visibility = Visibility.Collapsed;

}

private void ButtonCloseMenu\_Click(object sender, RoutedEventArgs e)

{

ButtonCloseMenu.Visibility = Visibility.Collapsed;

ButtonOpenMenu.Visibility = Visibility.Visible;

}

private void CreateTransaction(object sender, RoutedEventArgs e)

{

WpfApplication1.CreateTransaction transaction = new CreateTransaction();

transaction.Show();

}

private void AddComment(object sender, RoutedEventArgs e)

{

AddComment addComment = new AddComment();

addComment.Show();

}

private void Submit(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

try

{

conn.Open();

if (ListViewMenu.SelectedIndex == 3)

{

string sql = "select \* from comments";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

if (ListViewMenu.SelectedIndex == 4)

{

string sql = "select \* from review";

MySqlDataAdapter mySqlAdap = new MySqlDataAdapter(sql, conn);

MySqlCommandBuilder cb = new MySqlCommandBuilder(mySqlAdap);

cb.GetUpdateCommand();

mySqlAdap.Update(dt);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

private void Delete(object sender, RoutedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

var row = grid.SelectedItem as DataRowView;

if (row == null)

return;

row.Delete();

}

}

private void AddReview(object sender, RoutedEventArgs e)

{

AddNewReview addNewReview = new AddNewReview();

addNewReview.Show();

}

private void MenuItem\_OnClick(object sender, RoutedEventArgs e)

{

Authorize.menu.Close();

this.Close();

}

private void TextBoxFamaly\_OnTextChanged(object sender, TextChangedEventArgs e)

{

using (var conn = DataBaseConnect.GetDBConnection())

{

string textSearch = textBoxFamaly.Text;

if (ListViewMenu.SelectedIndex == 1)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"SELECT \* FROM transaction WHERE concat(number\_card,sum,user\_id,id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 2)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"select name,price,description,shelf\_life,weight,subcategory.name\_subcategory,category.name\_category from product inner JOIN subcategory on product.subcategory\_id = subcategory.id inner join category on subcategory.category\_id = category.id WHERE concat(name,price,description,shelf\_life,weight,subcategory\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 3)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"select commentary,user.firstname,user.lastname,product.name from comments inner JOIN user on comments.user\_id = user.id inner join product on comments.product\_id = product.id WHERE user\_id={Authorize.userId} and concat(commentary,product\_id,user\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

if (ListViewMenu.SelectedIndex == 4)

{

using (MySqlCommand cmdSel = new MySqlCommand($@"select commentary,user.firstname,user.lastname,product.name from review inner JOIN user on review.user\_id = user.id inner join product on review.product\_id = product.id WHERE user\_id={Authorize.userId} and concat(commentary,product\_id,user\_id) like '%{textSearch}%'", conn))

{

dt = new DataTable();

MySqlDataAdapter da = new MySqlDataAdapter(cmdSel);

da.Fill(dt);

grid.ItemsSource = dt.DefaultView;

}

}

}

}

}

}

**ПРИЛОЖЕНИЕ Е** **– Отзыв руководителя**



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

**ОТЗЫВ**

**руководителя на курсовую работу**

Булыгина Романа Дмитриевича (Ф.И.О. студента)

Проектирование и реализация базы данных интернет магазина хлебной продукции

(наименование темы КР)

представленный к защите по направлению/специальности

09.03.04 Программная инженерия

(код и наименование направления специальности подготовки)

Программное обеспечение вычислительной техники и автоматизированных систем (наименования профиля/специализация)

/ Новиков С. П. /

(должность) (подпись) (Ф.И.О)

« » 20\_\_ г.