

МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное  
учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Институт компьютерных технологий и информационной безопасности

Направление подготовки (шифр, название) 09.03.04 «Программная инженерия»

## Отчет по практике

обучающегося 2 курса

Фамилия Антюхин  
Имя Иван  
Отчество Геннадьевич

Обучающийся

Антюхин И. Г.

подпись

расшифровка подписи

Место практики Научно-исследовательская лаборатория «Лаборатория мобильной и веб-разработки» ООО НПИЦИТ «Иносфера» кафедры МОП ЭВМ

наименование профильной организации

Вид практики Учебная практика

Тип практики Практика по получению первичных профессиональных умений и навыков

Способ проведения практики Стационарная

Форма проведения практики Дискретная (по видам практик)

Сроки прохождения практики с 09.02.2019 по 03.05.2019

Руководитель практики  
от структурного подразделения ЮФУ

Родзина Ольга Николаевна

ФИО, подпись

## СОДЕРЖАНИЕ

Введение .....	3
1 Техническое задание .....	4
2 Проектирование АИС .....	5
2.1 Функциональные возможности системы .....	5
2.2 Функциональные возможности пользователей.....	5
2.3 Разработка требований к спецификации данных, связям таблиц, связям между данными.....	6
2.4 Разработка перечня форм, связей между ними, перечня элементов управления форм .....	6
3 Описание программной реализации .....	7
3.1 Реализация базы данных.....	8
3.2 Реализация главной и приветственной форм .....	12
3.3 Реализация регистрации и авторизации.....	14
3.4 Реализация формы с фильтрами отображения товаров.....	16
3.5 Реализация форм для редактирования списка товаров .....	17
3.6 Реализация формы с подробным отображением информации о товаре .....	19
3.7 Реализация формы управления заказами .....	20
3.8 Реализация «корзины» пользователя.....	21
3.9 Реализация личного кабинета пользователей.....	22
3.10 Реализация добавления и редактирования отзывов о товаре .....	23
3.11 Реализация экспорта информации о товарах.....	24
3.12 Реализация печати табличной информации .....	24
Заключение.....	25
Список использованных источников .....	26
Приложение 1. Листинг .....	27

## ВВЕДЕНИЕ

Целями учебной практики являются углубление и закрепление теоретических знаний полученных при изучении институтских дисциплин, а также развитие начальных практических умений и навыков по анализу, проектированию, написанию программного кода.

Основными задачами практики являются разработка требований к программной системе, проектирование архитектуры программной системы, реализация программной системы в соответствии с заданием, описание по применению программной системы, получение навыков создания приложений с пользовательским интерфейсом используя Windows Forms, получение навыков работы с табличными данными.

В результате выполнения практики была разработана АИС «Магазин электронной цифровой техники», обеспечивающая хранение, обработку, обновление информации о товарах, просмотр информации о товаре, оформление заказов и отслеживание статуса их выполнения в личном кабинете покупателя.

## 1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Необходимо разработать и реализовать автоматизированную информационную систему (АИС) «Магазин цифровой электронной техники».

Она должна обеспечивать доступ к данным, хранящимся в структурированном виде в базе данных (далее – БД). Для взаимодействия с базой данных должен использоваться язык запросов SQL.

АИС должна обеспечивать работу трём видам пользователей: гость, зарегистрированный пользователь (покупатель), администратор (работник магазина).

Взаимодействие с пользователями должно осуществляться через интерфейс, реализованный на Windows Forms.

В АИС должны быть реализованы формы: регистрационная, авторизационная, основная, формы для работы с товарами, «корзина» заказа, оформление заказа, личный кабинет.

Все экранные формы приложения должны соответствовать единому стилю и иметь логотип. На формах должны присутствовать надписи, текстовые поля, кнопки, списки, выпадающие списки, таблицы и т. д.

Приложение должно содержать не менее 12 экранных форм, имеющих связь между собой. Одна из форм должна работать с отображением графических данных (файлов изображений).

Разработанное приложение должно выполнять обработку данных по запросу пользователя (в т. ч. используя одновременно данные из нескольких таблиц).

В приложении должны обрабатываться ошибки неправильного ввода.

## 2 ПРОЕКТИРОВАНИЕ АИС

Автоматизированная информационная система (АИС) - совокупность программно-аппаратных средств, предназначенных для автоматизации деятельности, связанной с хранением, передачей и обработкой информации.

Проектирование систем – это процесс определения архитектуры, компонентов, интерфейсов и других характеристик системы или её части.

### 2.1 Функциональные возможности системы

Для системы определены следующие функциональные возможности:

- Вывод списка товаров, фильтрация и сортировка.
- Добавление товаров в корзину, с последующим оформлением заказа.
- Печати таблицы, а также возможность экспорта таблицы в Excel или CSV-файл.
- Редактирование товаров: добавление, изменение, удаление.
- Просмотр заказов и управление ими.
- Регистрация новых пользователей, в т. ч. администраторов.
- Личный кабинет покупателя с его заказами.

### 2.2 Функциональные возможности пользователей

В системе определены 3 вида пользователей - гость, зарегистрированный пользователь, администратор.

Гость – может просматривать список товаров, просматривать подробную информацию о товаре, добавлять/удалять товары из корзины, печатать таблицу товаров, экспортировать её в книгу Excel или в файл CSV.

Зарегистрированный пользователь (покупатель) – может, помимо вышеперечисленного, оформлять заказы, отслеживать их статус в личном кабинете, добавлять отзывы о товаре, редактировать свои отзывы.

Администратор (работник магазина) – может, помимо вышеперечисленного, редактировать базу данных с товарами, редактировать заказы (изменять их статус), регистрировать новых администраторов.

## 2.3 Разработка требований к спецификации данных, связям таблиц, связям между данными

Данные АИС хранятся в базе данных в виде таблиц. Для хранения данных текстового формата используются типы данных:

- NVARCHAR(50), NTEXT – для текстовых данных;
- INT – для целых чисел;
- DATETIME – для даты;
- IMAGE – для изображений.

Табличные данные связаны между собой по уникальным ключам.

## 2.4 Разработка перечня форм, связей между ними, перечня элементов управления форм

Был создан перечень из 15 форм:

- Приветственная форма
- Главная (основная) форма
- Форма с фильтрами товаров
- Форма входа
- Форма регистрации
- Личный кабинет пользователя
- Личный кабинет администратора
- Форма с подробным отображением товаров
- Форма для добавления/редактирования отзывов о товаре
- Форма «корзины» покупателя
- Форма управления товаром
- Форма добавления товара в БД
- Форма изменения информации о товаре
- Форма управления заказами
- Форма «О программе».

Проработано взаимодействие между ними, последовательность их вызова.

### 3 ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

АИС реализована на языке программирования C# [2], с использованием Windows Forms [3]. Реализовано 15 форм, взаимодействующих между собой, в зависимости от пользователя в системе.

В программе существует статический класс Common, в котором содержатся строка подключения к базе данных и список из id товаров, находящихся в корзине. Строка подключения считывается из конфигурационного файла «App.config».

При написании приложения использовался шаблон проектирования Singleton, для реализации единственного пользователя в системе. Класс User – реализация этого паттерна. В нем хранится информация о пользователе, находящемся в системе. При выходе из системы происходит сброс данных о пользователе. Таким образом в системе всегда есть пользователь, но, если не было авторизации – пользователь является гостем.

В классе User описаны 3 статических метода:

- Login – выполняет вход пользователя в систему, внося данные о пользователе в экземпляр класса.
- Logout – выполняет выход из системы, «обнуляя» информацию о пользователе и устанавливая статус пользователя – гость (значение Guest из перечисления UserStatus) в экземпляре класса.
- GetUser – возвращает пользователя (экземпляр класса) находящегося в системе.

Для взаимодействия с БД реализовано 2 класса для работы с файлами (изображениями):

- FileTools – содержит следующие методы: GetFileFromDB, PutBytesToDB, FileInBytes.
- ImageTools - содержит следующие методы: GetImageFromDB, PutImageInDB.

В программе определены следующие перечисления:

- «RequiredFilter», с возможными значениями NotRequired, CheckedList, FromTo.
- «UserStatus», с возможными значениями Guest, Customer, Admin.

- «OrderStatus», с возможными значениями Registered, Perfomed, Completed.

Для всех форм задана одна иконка (красная буква «D»), один шрифт – Cambria, 10 pt. Установлена одна двухцветная схема: цвет 1 – Info, цвет 2 – BurlyWood.

Выполнены ограничение размера некоторых форм и реализация корректной масштабируемости форм.

### 3.1 Реализация базы данных

Данные в программе хранятся в виде таблиц в локальной базе данных MS-SQL. Она содержит 4 таблицы: Orders, Products, Reviews, Users.

Строка подключения к базе данных выглядит следующим образом:

«Data Source=(LocalDB)\MSSQLLocalDB; AttachDbFilename=|DataDirectory|\DataBaseDET.mdf; Integrated Security=True».

Таблица Orders содержит поля:

- Id – Первичный ключ, целое автоинкрементирующееся число;
- Customer\_id – Id покупателя, целое число;
- Product\_id – Id товара, целое число;
- Date – Дата заказа
- Amount – Сумма заказа, целое число;
- Status – Сумма заказа, целое число;

Запрос для создания таблицы:

```
CREATE TABLE [dbo].[Orders] (
  [Id]          INT          IDENTITY (1, 1) NOT NULL,
  [Customer_id] INT          NOT NULL,
  [Product_id]  INT          NOT NULL,
  [Date]        DATETIME    NOT NULL,
  [Amount]      INT          NOT NULL,
  [Status]      INT          NOT NULL,
  PRIMARY KEY CLUSTERED ([Id] ASC));
```

Таблица предназначена для хранения информации о заказах. Содержит два поля, в которых содержатся первичные ключи двух других таблиц – Users и Products. Все поля обязательны для заполнения.



Таблица Products содержит поля:

- Id – Первичный ключ, целое автоинкрементирующееся число;
- Тип ПК – строка длиной не более 50 символов;
- Производитель – строка длиной не более 50 символов;
- Модель – строка длиной не более 50 символов;
- CPU – строка длиной не более 50 символов;
- Кол-во ядер – целое число;
- GPU – строка длиной не более 50 символов;
- Объем RAM – целое число;
- Тип RAM – строка длиной не более 50 символов;
- HDD – целое число;
- SSD – целое число;
- Операционная система – строка длиной не более 50 символов;
- Блок питания – строка длиной не более 50 символов;
- Склад – целое число, кол-во данного товара на складе;
- Цена – целое число;
- Описание – текст;
- Изображение – изображение товара;

Запрос для создания таблицы:

```
CREATE TABLE [dbo].[Products] (  
    [Id] INT IDENTITY (1, 1) NOT NULL,  
    [Тип ПК] NVARCHAR (50) NOT NULL,  
    [Производитель] NVARCHAR (50) NOT NULL,  
    [Модель] NVARCHAR (50) NOT NULL,  
    [CPU] NVARCHAR (50) NOT NULL,  
    [Кол-во ядер] INT NOT NULL,  
    [GPU] NVARCHAR (50) NOT NULL,  
    [Объем RAM] INT NOT NULL,  
    [Тип RAM] NVARCHAR (50) NOT NULL,  
    [HDD] INT NULL,  
    [SSD] INT NULL,  
    [Операционная система] NVARCHAR (50) NULL,  
    [Блок питания] NVARCHAR (50) NULL,  
    [Склад] INT NOT NULL,  
    [Цена] INT NOT NULL,  
    [Описание] NTEXT NULL,  
    [Изображение] IMAGE NULL,  
    PRIMARY KEY CLUSTERED ([Id] ASC)  
);
```

В таблице Products хранится информация о товарах. Все поля кроме, HDD, SSD, Операционная система, Блок питания, Описание и Изображение, обязательны для заполнения, т. к. не могут содержать NULL.

Таблица Reviews содержит поля:

- Id – Первичный ключ, целое автоинкрементирующееся число;
- Product\_id – Id покупателя, целое число;
- User\_id – Id пользователя, целое число;
- Mark – Оценка пользователя, целое число от 1 до 5;
- Advantages – Достоинства товара, текст;
- Disadvantages – Недостатки товара, текст;
- Comment – Комментарий;

Запрос для создания таблицы:

```
CREATE TABLE [dbo].[Reviews] (  
    [Id] INT IDENTITY (1, 1) NOT NULL,  
    [Product_id] INT NOT NULL,  
    [User_id] INT NOT NULL,  
    [Mark] INT NOT NULL,  
    [Advantages] NTEXT NULL,  
    [Disadvantages] NTEXT NULL,  
    [Comment] NTEXT NULL,  
    PRIMARY KEY CLUSTERED ([Id] ASC)  
);
```

Таблица Reviews предназначена для хранения отзывов и оценок пользователей о товарах

Таблица Users имеет поля:

- Id – Первичный ключ, целое автоинкрементирующееся число;
- Surname – Фамилия пользователя, строка длиной не более 50 символов;
- Name – Имя пользователя, строка длиной не более 50 символов;
- Patronymic – Отчество пользователя, строка длиной не более 50 символов;
- E-mail – Электронный адрес, строка длиной не более 50 символов;
- Nick – Никнейм пользователя, строка длиной не более 50 символов;
- Password – Пароль пользователя, строка длиной не более 50 символов;
- Status – Уровень прав в системе, целое число от 1 до 2;

Picture – Изображение пользователя, изображение;

Запрос для создания таблицы:

```
CREATE TABLE [dbo].[Users] (  
    [Id] INT IDENTITY (1, 1) NOT NULL,  
    [Surname] NVARCHAR (50) NOT NULL,  
    [Name] NVARCHAR (50) NOT NULL,  
    [Patronymic] NVARCHAR (50) NULL,  
    [E-mail] NVARCHAR (50) NOT NULL,  
    [Nick] NVARCHAR (50) NOT NULL,  
    [Password] NVARCHAR (50) NOT NULL,  
    [Status] INT NOT NULL,  
    [Picture] IMAGE NULL,  
    PRIMARY KEY CLUSTERED ([Id] ASC),  
    UNIQUE NONCLUSTERED ([E-mail] ASC),  
    UNIQUE NONCLUSTERED ([Nick] ASC)  
);
```

Таблица предназначена для хранения информации о пользователях. Поля «Patronymic» и «Picture» не обязательны для заполнения. Поля «Nick» и «E-mail» уникальны для каждой записи.

Связь с базой данных осуществляется посредством класса SqlConnection из библиотеки System.Data.SqlClient.dll. Получение результатов запросов выполняется одним из двух вариантов:

- Используя класс SqlDataReader;
- Используя класс SqlDataAdapter;

В программе применяются два варианта использования SqlDataAdapter:

- Формирование запроса «вручную»;
- Формирование запроса с помощью SqlDataBuilder (реализовано в форме управления заказами).

Для каждого метода, которому требуется подключение к базе данных, создается своё подключение, чтобы не возникало конфликтов подключения, и по завершению метода подключение закрывается. Если подключение не используется методом, вызвавшим данный метод, используется подключение из пула подключений – особенность реализации класса SqlConnection.

### 3.2 Реализация главной и приветственной форм

При запуске программы запускается главное окно (Рис. 1), и вместе с ним приветственное (Рис. 1), на котором пользователю предлагается войти/зарегистрироваться.

На главной форме слева над таблицей товаров присутствуют кнопки «Печать», «Экспорт в CSV», «Экспорт в Excel», «Корзина», «Обновить» и «Фильтры». Слева – фильтр по наличию товаров на складе.

В MenuStrip во вкладке «Файл» содержатся 2 пункта:

- «О программе» - вызов соответствующей формы;
- «Выход» - закрытие программы;

Во вкладке «Личный кабинет» содержатся 4 пункта:

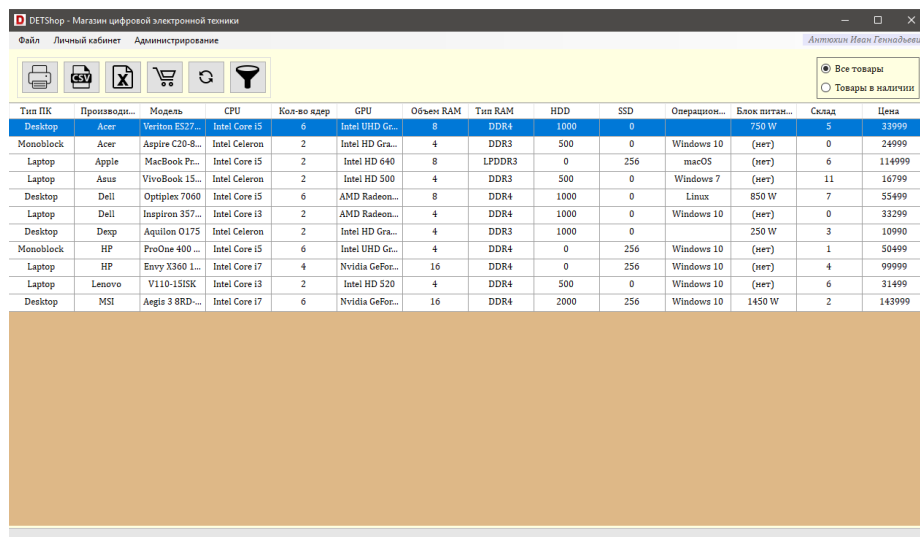
- «Перейти в личный кабинет»;
- «Выйти из учетной записи»;
- «Вход»;
- «Регистрация»;

Причем в зависимости от того выполнен вход или нет – видимы только либо пункты 1,2 либо пункты 3,4.

Вкладка «Администрирование» - видима (доступна) только когда в системе администратор. В ней 3 пункта:

- «Управление товарами»;
- «Управление заказами»;
- «Зарегистрировать нового пользователя» - для регистрации нового администратора;

После каждого вызова формы регистрации или авторизации вызывается метод UpdateControls. Он является «переключателем» элементов управления – показывает и прячет элементы управления в зависимости от типа пользователя в системе.



DETShop - магазин цифровой электронной техники

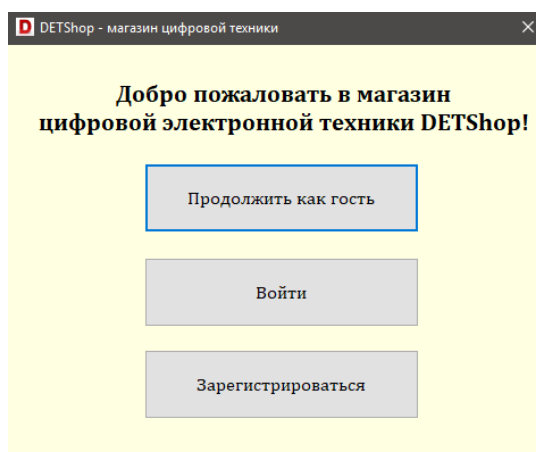
Файл Личный кабинет Администрирование

Антонин Иван Геннадьевич

☒ Все товары  
☐ Товары в наличии

Тип ПК	Производи...	Модель	CPU	Кол-во ядер	GPU	Объем RAM	Тип RAM	HDD	SSD	Операцион...	Блок питан...	Склад	Цена
Desktop	Acer	Veriton ES27...	Intel Core i5	6	Intel UHD Gr...	8	DDR4	1000	0	Windows 10	750 W	5	33999
Monoblock	Acer	Aspire C20-8...	Intel Celeron	2	Intel HD Gra...	4	DDR3	500	0	Windows 10	(нет)	0	24999
Laptop	Apple	MacBook Pr...	Intel Core i5	2	Intel HD 640	8	LPDDR3	0	256	macOS	(нет)	6	114999
Laptop	Asus	VivoBook 15...	Intel Celeron	2	Intel HD 500	4	DDR3	500	0	Windows 7	(нет)	11	16799
Desktop	Dell	Optiplex 7060	Intel Core i5	6	AMD Radeon...	8	DDR4	1000	0	Linux	850 W	7	55499
Laptop	Dell	Inspiron 357...	Intel Core i3	2	AMD Radeon...	4	DDR4	1000	0	Windows 10	(нет)	0	33299
Desktop	Dexp	Aquillon O175	Intel Celeron	2	Intel HD Gra...	4	DDR3	1000	0	Windows 10	250 W	3	10990
Monoblock	HP	ProOne 400 ...	Intel Core i5	6	Intel UHD Gr...	4	DDR4	0	256	Windows 10	(нет)	1	50490
Laptop	HP	Envy X360 1...	Intel Core i7	4	Nvidia Gefor...	16	DDR4	0	256	Windows 10	(нет)	4	99999
Laptop	Lenovo	V110-15ISK	Intel Core i3	2	Intel HD 520	4	DDR4	500	0	Windows 10	(нет)	6	31499
Desktop	MSI	Aegis 3 8RD...	Intel Core i7	6	Nvidia Gefor...	16	DDR4	2000	256	Windows 10	1450 W	2	143999

Рисунок 1. Главная форма



DETShop - магазин цифровой техники

**Добро пожаловать в магазин  
цифровой электронной техники DETShop!**

Продолжить как гость

Войти

Зарегистрироваться

Рисунок 2. Приветственная форма

Если выполнен вход в систему, в правом верхнем углу отображаются фамилия, имя, отчество пользователя.

Данные из БД загружаются функцией UpdateDataGridView. Этот метод учитывает отметку одной из 2-х радиокнопок (показывать все товары или только те, что в наличии) и формирует запрос в зависимости от них. Пока данные не загружены, кнопка «Фильтры» неактивна. После загрузки кнопка

становится активной. Функция UpdateDataGridView по мере загрузки данных «заполняет» ProgressBar под таблицей.

Корзина активна и открывается только если не пуста и, если пользователь в системе – не администратор.

При выборе из таблицы главной формы двойным щелчком товар открывается форма с подробной информацией о товаре (Рис. 6). Любой пользователь, кроме администратора, может добавить товары в корзину, но оформить заказ может только зарегистрированный пользователь. Зарегистрированный пользователь может оставить отзыв или отредактировать, если уже оставлял. Переключая вкладки можно просмотреть описание, характеристики товара и отзывы.

При нажатии кнопки «Добавить в корзину» происходит добавление товара в корзину. Добавить товар можно только один раз.

При нажатии кнопки «Добавить отзыв» открывается редактор отзывов (Рис. 7). Для сохранения отзыва следует нажать «Применить изменения».

### 3.3 Реализация регистрации и авторизации

В системе присутствует возможность регистрации новых пользователей.

Если в систему выполнен вход как администратор, то отображается CheckBox «Зарегистрировать как администратора».

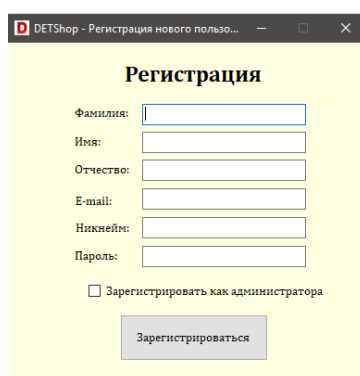


Рисунок 3. Форма регистрации

При нажатии кнопки «Зарегистрироваться» сначала производится проверка введенных полей на корректность. Для проверки корректности введенных полей используются регулярные выражения. Выглядят они следующим образом:

– « $\wedge\d+\$$ » – для проверки целых чисел.

- «<sup>^</sup>([a-z0-9]\_) {4,} \$» – для проверки никнейма и пароля. Допустимы латинские буквы, цифры и символ «\_». Кол-во введенных символов – не менее четырех.

Для проверки корректности e-mail используется функция IsValidEmail [4], которая использует регулярное выражение:

«<sup>^</sup>(? ("") ("".+?(?<!\\"""@)|(( [0-9a-z](\.(?!\\.))|[-!#\$%&'\*\+/=\?\\^\{\}\|~\w])\*)?(?<=[0-9a-z](?(\d)(\[(\d{1,3}\.){3}\d{1,3}\]|)|(( [0-9a-z] [-0-9a-z]\* [0-9a-z]\* \. )+[a-z0-9] [\ -a-z0-9]{0,22} [a-z0-9]))\$»

Выполняется проверка введенных никнейма и email на наличие только пробельных символов, проверяется на совпадения с уже имеющимися в базе. Для этого выполняется запросы «SELECT COUNT(Id) FROM Users WHERE [E-mail] = <введенный e-mail>» и «SELECT COUNT(Id) FROM Users WHERE [Nick] = <введенный никнейм>». При этом регистр символов не имеет значения, как в случае с авторизацией.

Рисунок 4. Форма входа пользователя

При загрузке формы авторизации вызывается метод LoadUserData, который загружает данные из таблицы пользователей (Users) используя следующий запрос:

"SELECT Id, Nick, [E-mail], Password FROM Users".

Это необходимо для корректного сравнения введенного никнейма, e-mail и пароля с имеющейся базой, т. к. SQL – регистронезависимый язык. Затем проверяется корректность введенных данных и, если проверки успешно пройдены, загружается полная информация о пользователе. Выполняется вход

в систему, используя статический метод Login класса User. В него передается информация о пользователе, загруженная из БД.

### 3.4 Реализация формы с фильтрами отображения товаров

В программе предусмотрена фильтрация списка товаров по критериям. Реализовано 2 типа фильтров:

- Фильтр с возможными вариантами выбора (FilterChecked) – для текстовых полей;
- Фильтр по диапазону (FilterFromTo) – для целочисленных полей;

Фильтры добавляются динамически, во время загрузки формы в элемент управления Panel. Для этого в программе реализованы классы FilterChecked и FilterFromTo. Каждый фильтр находится в своем GroupBox.

Рисунок 5. Фильтры товаров

Фильтр FilterChecked состоит из CheckedListBox. В конструкторе класса задается размер данного фильтра в пикселях, устанавливается имя фильтра.

Фильтр FilterFromTo состоит из 2-х TextBox, 2-х Label. Размеры и их положение в GroupBox задается в конструкторе класса.

Для реализации формы был использован класс Field, имеющий следующие поля:

- name – строка с именем столбца;



- filter – требуемый фильтр, одно из значений перечисления RequiredFilter;
- sqlCommand – строка, содержащая sql-запрос получения всех возможных значений данного столбца (необходимо для фильтра CheckedList). Запрос имеет вид «SELECT DISTINCT [ "<название столбца>" ] FROM [Products]».

При запуске формы с фильтрами изначально все фильтры отмечены (рис. 5), для фильтрации необходимо снять ненужные чекбоксы и заполнить требуемые поля. Каждый раз после закрытия формы, фильтры сбрасываются. Но при нажатии кнопки «Обновить» предыдущий запрос, заполнивший таблицу, выполняется ещё раз. Поэтому чтобы показать все товары достаточно открыть фильтры и нажать кнопку применить.

После нажатия на кнопку применения изменений, выполняется формирование запроса для обновления данных в главной форме. Для учета фильтров используются функции \_TakeAccountOfFiltersChecked и \_TakeAccountOfFiltersFromTo. Если они отработали без ошибок, то запрос обновления записывается в открытую статическую переменную QueryToUpdate формы MainForm (главной формы). После этого форма закрывается и данные в главной форме обновляются.

### 3.5 Реализация форм для редактирования списка товаров

Форма ProductManagement (рис. 6) предназначена для просмотра и выбора товара для удаления/изменения.

**Управление товаром**

Выберите товар из списка:

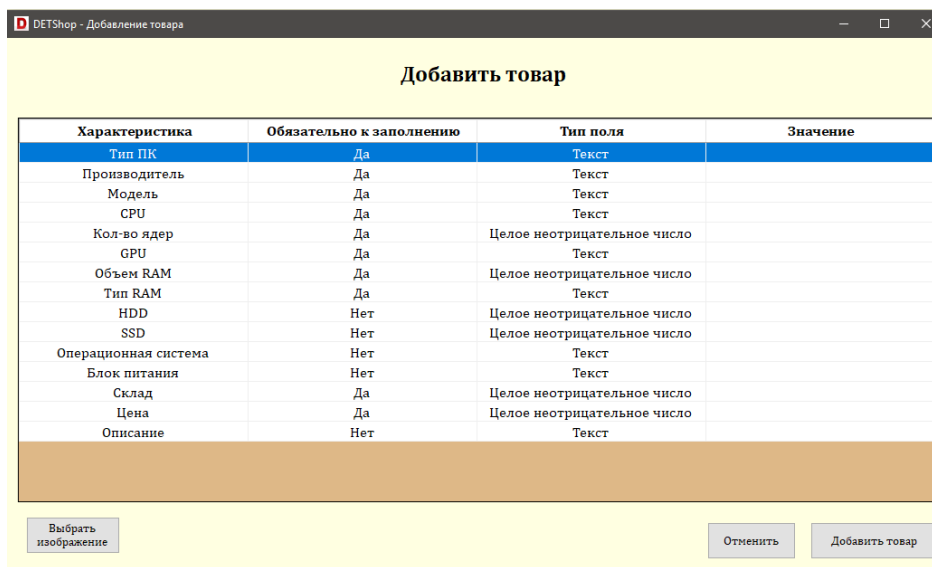
Тип...	Про...	Мо...	CPU	Кол...	GPU	Объ...	Тип...	HDD	SSD	Опе...	Бло...	Скл...	Цена	Опи...
Desktop	Desp	Aquilon...	Intel	4	NVIDIA	8	DDR4	(нет)	256	Windo...	(нет)	3	41990	ПК Asu...
MINI PC	Acer	m-12	Intel	4	AMD	4	DDR3	0	256	Ubuntu	CoolerL...	5	35990	Некото...
Laptop	Dell	Inspiro...	Intel	2	AMD	4	DDR4	1024	0	Ubuntu		6	24990	Описан...
Desktop	DEXP	Mars 19...	AMD	4	AMD	16	DDR4	2048	0	Windo...	PowerC...	0	46990	Описан...
Ноутбук	Asus	GoR 4-8...	Intel	8	NVIDIA	32	DDR4	1024	0	Windo...	AeroCo...	4	67990	Описан...
Desktop	Acer	Homer ...	AMD	4	AMD	4	DDR3	512	0	Windo...	Whistle...	3	72990	

Добавить новый товар    Изменить товар    Удалить товар

Рисунок 6. Форма управления товаром

Загружается при щелчке по пункту «Управление товарами» контекстного меню «Администрирование» в главной форме. Может запустить только администратор.

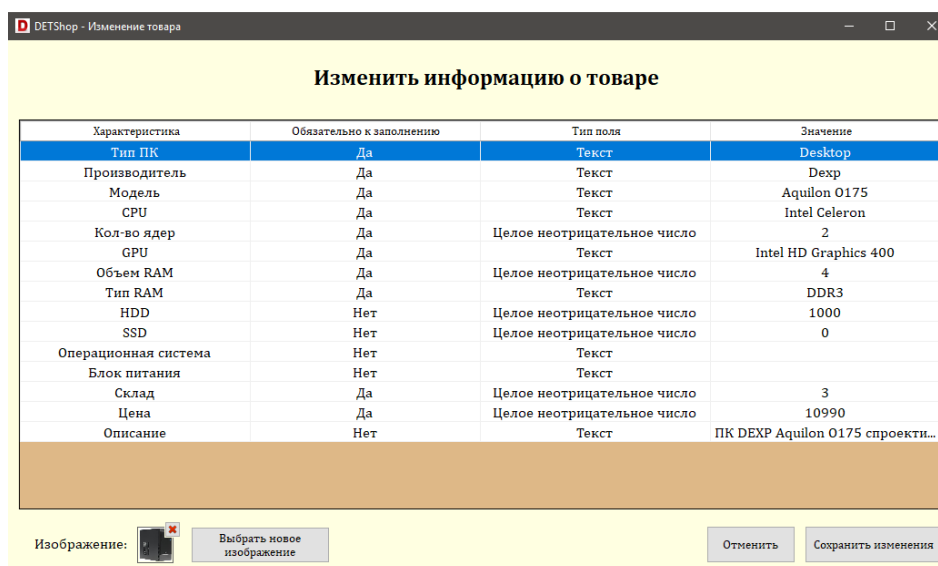
При нажатии кнопки удаления у пользователя спрашивается подтверждение удаления записи с выводом первых 4-х ячеек записи. В случае подтверждения, в БД отправляется запрос об удалении «DELETE FROM Products WHERE Id=<id товара>».



Характеристика	Обязательно к заполнению	Тип поля	Значение
Тип ПК	Да	Текст	
Производитель	Да	Текст	
Модель	Да	Текст	
CPU	Да	Текст	
Кол-во ядер	Да	Целое неотрицательное число	
GPU	Да	Текст	
Объем RAM	Да	Целое неотрицательное число	
Тип RAM	Да	Текст	
HDD	Нет	Целое неотрицательное число	
SSD	Нет	Целое неотрицательное число	
Операционная система	Нет	Текст	
Блок питания	Нет	Текст	
Склад	Да	Целое неотрицательное число	
Цена	Да	Целое неотрицательное число	
Описание	Нет	Текст	

Рисунок 7. Форма добавления товара

Форма AddProduct (рис. 7) запускается при нажатии кнопки «Добавить товар» в предыдущей форме (ProductManagement). Во время запуска заполняется таблица DataGridView в которой первые 3 поля – информативные, указывающие как заполнять поле, название которого указано в 1-м столбце, обязательность заполнения во 2-м и тип данных в 3-м. В 4-й столбец



Характеристика	Обязательно к заполнению	Тип поля	Значение
Тип ПК	Да	Текст	Desktop
Производитель	Да	Текст	Dexp
Модель	Да	Текст	Aquilon 0175
CPU	Да	Текст	Intel Celeron
Кол-во ядер	Да	Целое неотрицательное число	2
GPU	Да	Текст	Intel HD Graphics 400
Объем RAM	Да	Целое неотрицательное число	4
Тип RAM	Да	Текст	DDR3
HDD	Нет	Целое неотрицательное число	1000
SSD	Нет	Целое неотрицательное число	0
Операционная система	Нет	Текст	
Блок питания	Нет	Текст	
Склад	Да	Целое неотрицательное число	3
Цена	Да	Целое неотрицательное число	10990
Описание	Нет	Текст	ПК DEXP Aquilon 0175 спроекти...

Рисунок 8. Форма изменения информации о товаре

пользователь вносит значение поля. После нажатия кнопки «Добавить товар» выполняется проверка данных. При этом используется регулярное выражение « $\wedge\d+\$$ » для проверки корректности ввода целых чисел. Введенные пользователем данные хранятся в контейнере List. Если все поля заполнены корректно формируется запрос к БД.

Форма ChangeProduct (рис. 8) применяется для изменения информации о продукте. Запускается при нажатии кнопки «Изменить товар» в форме ProductManagement, либо при двойном нажатии левой кнопки мыши по строке с товаром. Отличается от предыдущей формы AddProduct оформлением и реализацией проверки корректности данных.

### 3.6 Реализация формы с подробным отображением информации о товаре

Форма Product (рис. 9) вызывается двойным кликом мыши по товару из списка в главной форме. При её создании в конструктор передается строка DataGridView, из которой форма получает id товара для загрузки полной информации о нём.

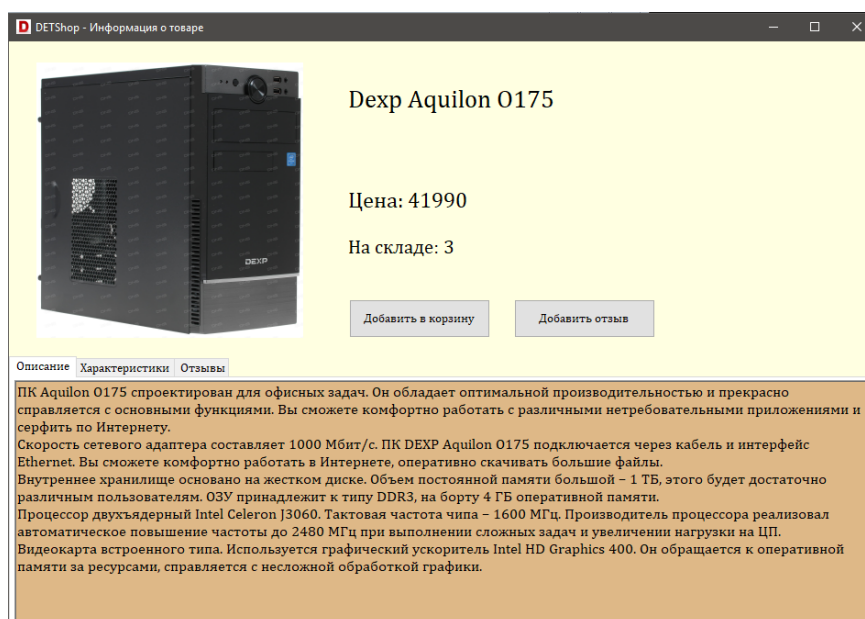


Рисунок 9. Форма с подробным отображением информации о товаре

При загрузке формы загружаются название продукта, цена, кол-во товара на складе. Затем загружаются описание, характеристики товара, и отзывы.

При нажатии кнопки «Добавить в корзину» id товара добавляется в контейнер ProductsInCarts (если его ещё там нет) содержащийся в статическом классе Common. Если товара нет на складе - кнопка «Добавить в корзину» деактивируется.

По нажатию кнопки добавить отзыв — модальным окном вызывается редактор отзывов (ReviewEditor) и после его закрытия происходит обновление отзывов.

Если в систему вошел администратор, то кнопки «Добавить в корзину» и «Добавить отзыв» скрываются.

### 3.7 Реализация формы управления заказами

Форма управления заказами (рис. 10) запускается при выборе пункта «Управление заказами» во вкладке «Администрирование» из контекстного меню MenuStrip на главной форме. Форма может быть открыта только администратором.

Номер заказа	Id покупателя	Id товара	Дата заказа	Сумма заказа	Статус заказа
6	1	7	05.04.2019 0:01	24990.0000	2
7	1	6	05.04.2019 0:05	35990.0000	1
8	1	12	05.04.2019 17:22	72990.0000	0

Рисунок 10. Форма управления заказами

В таблице с заказами, все столбцы, кроме «Статуса заказа», устанавливаются только для чтения. Администратор изменяет статус заказа по мере его выполнения.

Загрузка всех заказов из БД осуществляется в зависимости от выбранного в ComboBox режима. При первой загрузке будут загружены все заказы. Но при необходимости можно показать только не завершенные, выбрав соответствующий вариант в ComboBox.

Статус заказа – число от 0 до 2, где 0 – «заказ оформлен», 1 – «заказ обрабатывается», 2 - «заказ завершен». В программе соответствует одному из значений перечисления OrderStatus.

Администратор может изменять статус заказа (рис. 10). Чтобы изменить статус заказа необходимо кликнуть на ячейку «Статус заказа» и ввести новое значение.

На форме есть фильтр позволяющий скрыть все завершенные заказы – ComboBox.

Администратор может удалять заказы – по выделению строки нажать клавишу на клавиатуре “Delete”.

Для применения изменений следует нажать кнопку, соответствующую кнопу.

Для обновления данных в БД используются классы SqlCommandBuilder и SqlDataAdapter.

### 3.8 Реализация «корзины» пользователя

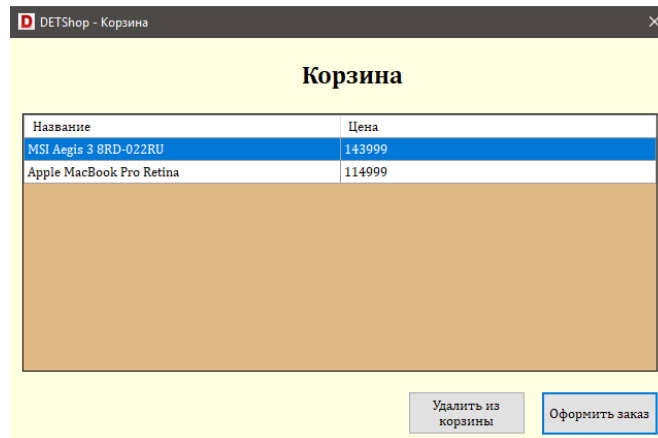


Рисунок 11. Корзина пользователя

Форма «корзины» (рис. 11) вызывается при нажатии на кнопке с изображением корзины в главной форме. Если корзина пуста или выполнен вход в систему как администратор - форма не запустится. При загрузке формы происходит загрузка всех товаров, id которых есть в Common.ProductsInCart.

По нажатию на кнопку «Оформить заказ» после подтверждения отправляется запрос на вставку данных в таблицу Orders.

Запрос имеет вид «INSERT INTO Orders (Customer\_id, Product\_id, Date, Amount, Status) VALUES (... , ... , ... , ... , ...)».

Для оформления заказа необходимо авторизоваться.

По нажатию кнопки «Удалить из корзины» происходит удаление элемента из DataGridView и из Common.ProductsInCart.

### 3.9 Реализация личного кабинета пользователей

В системе реализован личный кабинет (далее - ЛК) для 2-х видов пользователей: для покупателя и администратора.

В обоих ЛК выполняется загрузка фамилии, имени, отчества, e-mail, никнейма и изображение о пользователе в форму. Но, помимо этого, в форме ЛК покупателя (рис. 12) выполняется загрузка заказов пользователя по запросу:

```
SELECT CONCAT(pr.Производитель, ' ', pr.Модель), ord.Date, ord.Amount, ord.Status  
FROM Products AS pr, Orders AS ord WHERE(pr.Id=ord.Product_id) AND  
(ord.Customer_id={User.GetUser().Id})
```

При нажатии на linkLabel «Новое изображение» открывается OpenFileDialog чтобы выбрать новое изображение.

Товар	Дата	Сумма	Статус заказа
Apple MacBook Pro Reti...	05.07.2019 22:53:36	114999	0
Dexr Aquilon 0175	05.08.2019 14:42:18	10990	0

Рисунок 12. Личный кабинет покупателя

При закрытии формы у пользователя спрашивается, сохранять ли изменения. Если ответ утвердительный, то происходит обновление информации в БД.

### 3.10 Реализация добавления и редактирования отзывов о товаре

Форма ReviewEditor (рис. 13) запускается из формы Product (форма с подробностями о товаре), если выполнен вход как покупатель. Таким образом оставлять отзывы могут только покупатели.

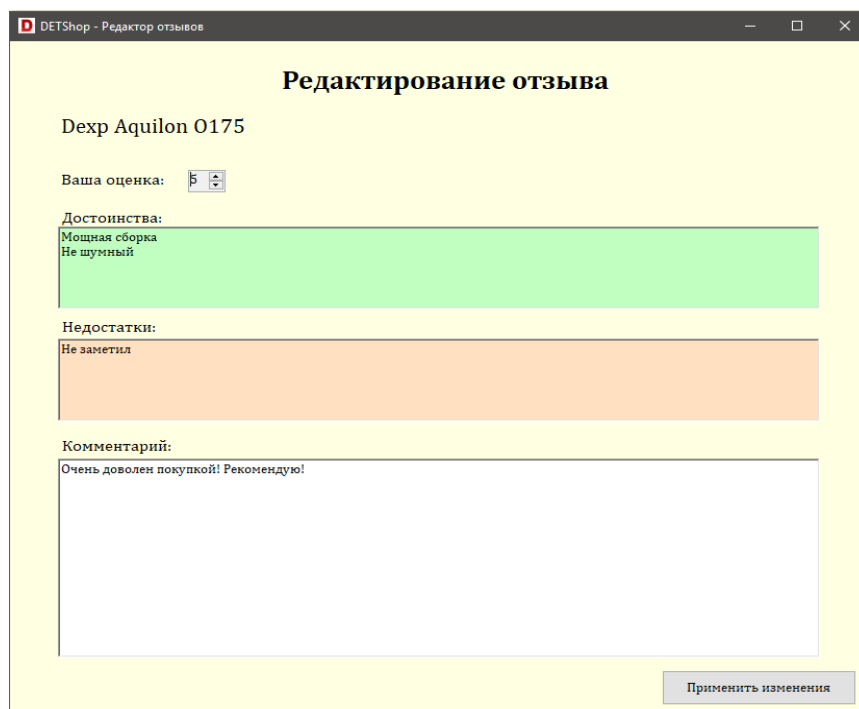


Рисунок 13. Редактор отзывов

Во время загрузки формы загружается название товара, а затем выполняется попытка загрузить данные об отзыве (оценка, достоинства, недостатки и комментарий к данному товару, от данного пользователя) и, если данные не получены (т. е. если пользователь ещё не оставлял отзыв) - устанавливается флаг создания нового отзыва. При этом используется команда вставки (INSERT INTO) новой записи.

Если же отзыв найден в БД, то он загружается в форму. И тогда используется запрос обновления (UPDATE), после нажатия на кнопку «Применить изменения» и подтверждения этого действия.

Отзыв о товаре состоит из 4-х элементов: оценка «от 1 до 5» и три текстовых поля «Достоинства», «Недостатки», «Комментарий».

### 3.11 Реализация экспорта информации о товарах

В программе реализован экспорт табличной информации в Excel-файлы и файлы формата CSV.

Для экспорта табличных данных в таблицы Excel используется библиотека Microsoft.Office.Interop.Excel.dll.

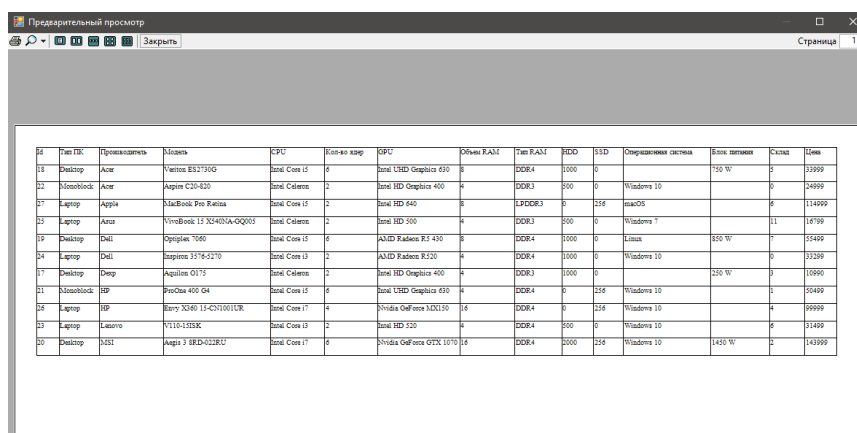
По нажатию на кнопку «ExportToExcel» (кнопка с изображением листа с буквой «X» на главной форме), открывается SaveFileDialog и пользователь выбирает имя файла и каталог для сохранения. При экспорте происходит копирование всех ячеек таблицы DataGridView в таблицу Excel.

Для того, чтобы экспортировать в CSV, следует нажать на кнопку с надписью CSV. При экспорте происходит конвертация всех ячеек таблицы DataGridView в одну строку с разделителями (разделитель – точка с запятой), затем экспорт этой строки в csv-файл.

### 3.12 Реализация печати табличной информации

По нажатию кнопки «Печать» создается новый документ для печати, производится подписка на событие printDocument\_PrintPage. Создается и показывается printDialog, пользователь выбирает параметры печати. Затем открывается предпросмотр previewPrintDialog (рис. 14). Печать происходит после нажатия соответствующей кнопки в предпросмотре.

printDocument\_PrintPage() – обработчик события печати - «рисует» документ для печати из DataGridView.



The screenshot shows a window titled 'Предварительный просмотр' (Preview) with a table of computer specifications. The table has 14 columns: ID, Тип ПК (PC Type), Производитель (Manufacturer), Модель (Model), CPU, Кол-во ядер (Number of cores), GPU, Объем RAM (RAM size), Тип RAM (RAM type), HDD, SSD, Операционная система (Operating system), Цена поставки (Delivery price), Скидка (Discount), and Цена (Price). The table contains 10 rows of data for various laptops and desktops.

ID	Тип ПК	Производитель	Модель	CPU	Кол-во ядер	GPU	Объем RAM	Тип RAM	HDD	SSD	Операционная система	Цена поставки	Скидка	Цена
18	Десктоп	Asus	Vaio ES1700G	Intel Core i5	4	Intel UHD Graphics 610	8	DDR4	1000	0	Windows 10	750 W	5	33999
22	Моноблок	Asus	Pro C20-E20	Intel Celeron	2	Intel HD Graphics 400	4	DDR3	500	0	Windows 10		0	24999
27	Лaptop	Apple	MacBook Pro Retina	Intel Core i5	2	Intel HD 440	8	LPDDR3	0	256	macOS		0	114999
28	Лaptop	Asus	VivoBook 15 X402NA-QQ001	Intel Celeron	2	Intel HD 500	4	DDR3	500	0	Windows 7		11	18799
39	Десктоп	Dell	Optiplex 7080	Intel Core i5	4	AMD Radeon R7 430	8	DDR4	1000	0	Linux	850 W	7	51499
24	Лaptop	Dell	Inspiron 15 7520	Intel Core i3	2	AMD Radeon R320	4	DDR4	1000	0	Windows 10		0	33299
17	Десктоп	Dell	Optiplex 7070	Intel Celeron	2	Intel HD Graphics 400	4	DDR3	1000	0	Windows 10	220 W	3	10999
01	Моноблок	HP	ProOne 400 G4	Intel Core i5	4	Intel UHD Graphics 610	8	DDR4	0	256	Windows 10		1	20499
04	Лaptop	HP	Butterfly 15-CN1001UR	Intel Core i7	4	Nvidia GeForce MX250	8	DDR4	0	256	Windows 10		4	99999
09	Лaptop	Samsung	NP150-1500K	Intel Core i3	2	Intel HD 520	4	DDR4	500	0	Windows 10		0	31499
10	Десктоп	ASUS	Asus Z 97D-4128RU	Intel Core i7	4	Nvidia GeForce GTX 1070	8	DDR4	1000	256	Windows 10	1450 W	2	145999

Рисунок 14. Предпросмотр документа перед печатью



## ЗАКЛЮЧЕНИЕ

В результате прохождения учебной практики разработана АИС «Магазин цифровой электронной техники», которая способна выполнять следующие задачи:

- Выводить список товаров;
- Осуществлять фильтрацию и сортировку списка товаров;
- Обеспечивать вход в систему и регистрацию новых пользователей;
- Просматривать подробную информацию о товаре;
- Добавлять отзывы о товарах;
- Добавлять товары в корзину;
- Оформлять заказы;
- Отслеживать статус выполнения заказов ч/з личный кабинет;
- Редактировать таблицу товаров, удалять товары, добавлять новые;

Для успешного выполнения задания практики пройден курс SQL Fundamentals Course от портала SoloLearn.

Были освоены средства разработки языка C#, платформы .NET, языка запросов SQL и базы данных MS-SQL, получен опыт проектирования, написания приложений на Windows Forms.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство по ADO.NET и работе с базами данных [metanit.com] // URL: <https://metanit.com/sharp/adonet/> (дата обращения 20.04.2019)
2. Язык программирования C# [ru.wikipedia.org] // URL: [https://ru.wikipedia.org/wiki/C\\_Sharp](https://ru.wikipedia.org/wiki/C_Sharp) (дата обращения 18.03.2019)
3. Руководство по программированию в Windows Forms [metanit.com] // URL: <https://metanit.com/sharp/windowsforms/> (дата обращения 23.04.2017)
4. Как выполнить проверку строк на соответствие формату электронной почты. [docs.microsoft.com] // URL: <https://docs.microsoft.com/ru-ru/dotnet/standard/base-types/how-to-verify-that-strings-are-in-valid-email-format/> (дата обращения 14.03.2019)

## ПРИЛОЖЕНИЕ 1. ЛИСТИНГ

### Файл AboutProgram.cs

```
namespace AIS_shop
{
    partial class AboutProgram : Form
    {
        public AboutProgram()
        {
            InitializeComponent();
            this.Text = "DETSnop - O программе";
            this.labelProductName.Text = "Автоматизированная информационная система
\\Магазин цифровой электронной техники\\";
            this.labelVersion.Text = String.Format("Версия {0}", AssemblyVersion);
            this.labelCopyright.Text = "© 2019, Антюхин И.Г.";
            this.labelCompanyName.Text = AssemblyCompany;
            this.textBoxDescription.Text = "Данный программный продукт предназначен
для автоматизации, ускорения, упрощения работы с базой данных магазина.";
        }

        #region Методы доступа к атрибутам сборки

        public string AssemblyTitle
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute),
false);
                if (attributes.Length > 0)
                {
                    AssemblyTitleAttribute titleAttribute =
(AssemblyTitleAttribute)attributes[0];
                    if (titleAttribute.Title != "")
                    {
                        return titleAttribute.Title;
                    }
                }
                return
System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);
            }
        }

        public string AssemblyVersion
        {
            get
            {
                return Assembly.GetExecutingAssembly().GetName().Version.ToString();
            }
        }

        public string AssemblyDescription
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute),
false);
                if (attributes.Length == 0)
                {
                    return "";
                }
            }
        }
    }
}
```

```

        return ((AssemblyDescriptionAttribute)attributes[0]).Description;
    }
}

public string AssemblyProduct
{
    get
    {
        object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute),
false);

        if (attributes.Length == 0)
        {
            return "";
        }
        return ((AssemblyProductAttribute)attributes[0]).Product;
    }
}

public string AssemblyCopyright
{
    get
    {
        object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute),
false);

        if (attributes.Length == 0)
        {
            return "";
        }
        return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;
    }
}

public string AssemblyCompany
{
    get
    {
        object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute),
false);

        if (attributes.Length == 0)
        {
            return "";
        }
        return ((AssemblyCompanyAttribute)attributes[0]).Company;
    }
}
#endregion

private void okButton_Click(object sender, EventArgs e)
{
    Close();
}
}
}

```

## Файл AddProduct.cs

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;
using System.Text.RegularExpressions;

namespace AIS_shop
{
    public partial class AddProduct : Form
    {
        List<_strToGridView> fields = new List<_strToGridView>(15);
        // изображение
        byte[] dataImage = null;
        // команда добавления в бд
        string commandText = null;

        public AddProduct()
        {
            InitializeComponent();
        }

        private void AddNewProduct_Load(object sender, EventArgs e)
        {
            foreach (DataGridViewColumn col in dgv.Columns)
                col.SortMode = DataGridViewColumnSortMode.NotSortable;

            fields.Add(new _strToGridView("Тип ПК", "Да", "Текст"));
            fields.Add(new _strToGridView("Производитель", "Да", "Текст"));
            fields.Add(new _strToGridView("Модель", "Да", "Текст"));
            fields.Add(new _strToGridView("CPU", "Да", "Текст"));
            fields.Add(new _strToGridView("Кол-во ядер", "Да", "Целое неотрицательное
число"));
            fields.Add(new _strToGridView("GPU", "Да", "Текст"));
            fields.Add(new _strToGridView("Объем RAM", "Да", "Целое неотрицательное
число"));
            fields.Add(new _strToGridView("Тип RAM", "Да", "Текст"));
            fields.Add(new _strToGridView("HDD", "Нет", "Целое неотрицательное
число"));
            fields.Add(new _strToGridView("SSD", "Нет", "Целое неотрицательное
число"));
            fields.Add(new _strToGridView("Операционная система", "Нет", "Текст"));
            fields.Add(new _strToGridView("Блок питания", "Нет", "Текст"));
            fields.Add(new _strToGridView("Склад", "Да", "Целое неотрицательное
число"));
            fields.Add(new _strToGridView("Цена", "Да", "Целое неотрицательное
число"));
            fields.Add(new _strToGridView("Описание", "Нет", "Текст"));

            labelFileName.Visible = false;
            pictureBox.Visible = false;
            buttonDelImage.Visible = false;
            dgv.RowHeadersVisible = false;

            // вывод в DataGridView
            for (int i = 0; i < fields.Count; i++)
                dgv.Rows.Add(fields[i].name, fields[i].obligation, fields[i].type);
            commandText = @"INSERT INTO Products (";
            // добавление к запросу полей, в которые будет осуществляться вставка

```

```

        foreach (var f in fields)
        {
            if (f != fields[0]) commandText += @", ";
            commandText += $"{f.name}";
        }
        commandText += @", [Изображение]";

        commandText +=
            @" VALUES (@type, @brand, @model, @cpu, @cores, @gpu, @ram, @typeram, @hdd, @ssd,
            @os, @psu, @stock, @cost, @descripton, @image)";
    }

    private void AddNewProduct_FormClosing(object sender, FormClosingEventArgs e)
    {

    }

    private async void buttonSave_Click(object sender, EventArgs e)
    {
        if (valid())
        {
            if (MessageBox.Show("Вы уверены, что хотите добавить этот товар?",
"Добавить товар",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                DialogResult.Yes)
            {
                // добавление введенных значений столбцов в команду
                SqlConnection connection = new
                SqlConnection(Common.StrSqlConnection);
                SqlCommand query = new SqlCommand(commandText, connection);

                query.Parameters.AddWithValue("@type", fields[0].value);
                query.Parameters.AddWithValue("@brand", fields[1].value);
                query.Parameters.AddWithValue("@model", fields[2].value);
                query.Parameters.AddWithValue("@cpu", fields[3].value);
                query.Parameters.AddWithValue("@cores", fields[4].value);
                query.Parameters.AddWithValue("@gpu", fields[5].value);
                query.Parameters.AddWithValue("@ram", fields[6].value);
                query.Parameters.AddWithValue("@typeram", fields[7].value);
                query.Parameters.AddWithValue("@hdd", fields[8].value);
                query.Parameters.AddWithValue("@ssd", fields[9].value);
                query.Parameters.AddWithValue("@os", fields[10].value);
                query.Parameters.AddWithValue("@psu", fields[11].value);
                query.Parameters.AddWithValue("@stock", fields[12].value);
                query.Parameters.AddWithValue("@cost", fields[13].value);
                query.Parameters.AddWithValue("@descripton", fields[14].value);

                if (dataImage != null) query.Parameters.AddWithValue("@image",
dataImage);
                else query.Parameters.AddWithValue("@image", DBNull.Value);
                // выполнение команды
                try
                {
                    connection.Open();
                    if (await query.ExecuteNonQueryAsync() == 1)
                    {
                        MessageBox.Show("Запись была успешно добавлена в
таблицу", "Сообщение",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);
                        ProductManagement.updateFlag = true;
                        Close();
                    }
                }
                catch (Exception ex)
                {

```

```

        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (connection != null && connection.State !=
ConnectionState.Closed)
            connection.Close();
    }
}

private bool valid()
{
    string intPattern = @"^\d+$";

    foreach (DataGridViewRow row in dgv.Rows)
    {
        string sValue = null;
        string sChar = row.Cells[0]?.Value?.ToString();
        string sObligation = row.Cells[1]?.Value?.ToString();
        string sType = row.Cells[2]?.Value?.ToString();
        if (row.Cells[3].Value == null)
            sValue = "";
        else sValue = row.Cells[3]?.Value?.ToString();

        if (string.IsNullOrEmpty(sValue))
        { // если строка пустая или с одними пробелами
            if (sObligation == "Да")
            { // если обязательное поле
                MessageBox.Show($"Поле {sChar} должно быть заполнено",
"Некорректный ввод!",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
            fields.Find(f => f.name == sChar).value = DBNull.Value;
        }
        else
        {
            switch (sType)
            {
                case "Целое неотрицательное число":
                    Regex regex = new Regex(intPattern);
                    if (regex.IsMatch(sValue))
                    {
                        int val = int.Parse(sValue);
                        fields.Find(f => f.name == sChar).value = val;
                    }
                    else
                    {
                        MessageBox.Show($"Поле \"{sChar}\" заполнено не
корректно", "Некорректный ввод!",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return false;
                    }
                    break;
                case "Текст":
                    fields.Find(f => f.name == sChar).value = sValue;
                    break;
                default:
                    MessageBox.Show("Произошла ошибка при распознавании типа
значения. См. код \'AddNewProduct.valid()\'", "Ошибка!",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return false;
            }
        }
    }
}

```

```

    }
    }
    }
    return true;
}

private void buttonCancel_Click(object sender, EventArgs e)
{
    Close();
}

private void buttonAddImage_Click(object sender, EventArgs e)
{
    labelFileName.Text = "";
    openImage.Filter = "Изображения
(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG";
    if (openImage.ShowDialog() == DialogResult.Cancel) return;

    // получаем файл в виде байтов
    dataImage = FileTools.FileInBytes(openImage.FileName);
    // выводим его в pictureBox
    pictureBox.Image = Image.FromStream(new MemoryStream(dataImage));
    pictureBox.Visible = true;
    string filename = Path.GetFileName(openImage.FileName);
    if (filename.Length <= 50)
    {
        labelFileName.Text = filename;
        labelFileName.Visible = true;
    }
    buttonDelImage.Visible = true;
}

private void buttonDelImage_Click(object sender, EventArgs e)
{
    dataImage = null;
    pictureBox.Image = null;
    pictureBox.Visible = false;
    labelFileName.Text = "";
    labelFileName.Visible = false;
    buttonDelImage.Visible = false;
}
}
}

```

## Файл AdminProfile.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class AdminProfile : Form
    {
        byte[] dataImage { get; set; } = null;
        bool Unsaved = false;

        public AdminProfile()
        {
            InitializeComponent();
        }
    }
}

```



```

private void AdminProfile_Load(object sender, EventArgs e)
{
    LoadInfo();
}

private void AdminProfile_FormClosing(object sender, FormClosingEventArgs e)
{
    if (Unsaved)
    {
        DialogResult result =
            MessageBox.Show("Сохранить изменение изображения?",
                "Подтверждение действия",
                MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
            if (dataImage != null)
                if (FileTools.PutBytesToDB(dataImage,
                    Common.StrSQLConnection, @"Users", @"Picture", User.GetUser().Id))
                    MessageBox.Show("Файл успешно загружен в базу данных.",
                        "Сообщение",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
                else MessageBox.Show("Файл не был загружен в базу данных.",
                    "Сообщение",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            else deleteImage();

        if (result == DialogResult.Cancel) e.Cancel = true;
    }
}

private void LoadInfo()
{
    User user = User.GetUser();
    if (user.Status == UserStatus.Guest)
    {
        MessageBox.Show("Вы не авторизованы", "Вход не выполнен",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    textBoxSurname.Text = user.Surname;
    textBoxName.Text = user.Name;
    if (!string.IsNullOrEmpty(user.Patronymic))
        textBoxPatronymic.Text = user.Patronymic;
    else textBoxPatronymic.Text = "(не указано)";
    textBoxEmail.Text = user.Email;
    textBoxNick.Text = user.Nick;
    dataImage = FileTools.GetFileFromDB(Common.StrSQLConnection, "Users",
        "Picture", user.Id);
    if (dataImage == null)
    {
        pictureBox.Image = Properties.Resources.nofoto;
        buttonDelImage.Visible = false;
    }
    else pictureBox.Image = Image.FromStream(new MemoryStream(dataImage));
}

private void linkNewImage_LinkClicked(object sender,
    LinkLabelLinkClickedEventArgs e)
{
    openNewImage.Filter = "Изображения (*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG";
    if (openNewImage.ShowDialog() == DialogResult.Cancel) return;
    // получаем файл в виде байтов
    dataImage = FileTools.FileInBytes(openNewImage.FileName);
}

```

```

        // выводим его в pictureBox
        pictureBox.Image = Image.FromStream(new MemoryStream(dataImage));
        if (dataImage != null)
        {
            Unsaved = true;
            buttonDelImage.Visible = true;
        }
    }

    private void buttonDelImage_Click(object sender, EventArgs e)
    {
        dataImage = null;
        pictureBox.Image = Properties.Resources.nofoto;
        Unsaved = true;
        buttonDelImage.Visible = false;
    }

    private async void deleteImage()
    {
        SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
        SqlCommand sqlCommand = new SqlCommand($"UPDATE Users SET Picture=NULL
WHERE Id={User.GetUser().Id}", connection);
        try
        {
            await connection.OpenAsync();
            if (await sqlCommand.ExecuteNonQuery() == 1)
                MessageBox.Show("Изображение удалено", "Сообщение",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (connection != null && connection.State != ConnectionState.Closed)
                connection.Close();
        }
    }
}

```

## Файл Authorization.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class Authorization : Form
    {
        DataSet usersData = null;
        bool UsersDataLoaded = false;

        public Authorization()
        {
            InitializeComponent();
        }

        private void Authorization_Load(object sender, EventArgs e)
        {

```

```

        LoadUsersData();
    }

    private void Authorization_FormClosing(object sender, FormClosingEventArgs e)
    {
        Cursor = Cursors.Default;
    }

    private void bToRegistration_Click(object sender, EventArgs e)
    {
        Hide();
        Registration reg = new Registration();
        reg.ShowDialog();
        Close();
    }

    private async void bEnter_Click(object sender, EventArgs e)
    {
        string nick = maskedTextBox1.Text, password = maskedTextBox2.Text;
        if (!string.IsNullOrEmpty(nick) &&
!string.IsNullOrEmpty(password))
        {
            if (!UsersDataLoaded)
            {
                MessageBox.Show("Данные о пользователях не загружены. Попробуйте повторить попытку позже.", "Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                return;
            }
            int id = 0;
            for (int it = 0; it < usersData.Tables[0].Rows.Count; it++)
            {
                string currentNick =
usersData.Tables[0].Rows[it].ItemArray[1].ToString();
                string currentEmail =
usersData.Tables[0].Rows[it].ItemArray[2].ToString();
                string currentPassword =
usersData.Tables[0].Rows[it].ItemArray[3].ToString();

                if ((nick == currentNick || nick == currentNick) && password ==
currentPassword)
                {
                    id = (int)usersData.Tables[0].Rows[it].ItemArray[0];
                    break;
                }
            }
            if (id == 0)
            {
                MessageBox.Show("Некорректные данные для входа. Проверьте правильность введенных данных", "Пользователь не найден", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }
            // далее, выполняем вход
            SqlConnection connection = new
SqlConnection(Common.StrSQLConnection);
            SqlCommand query = new SqlCommand(@"SELECT Surname, Name, Patronymic,
[E-mail], Nick, Status FROM Users WHERE Id=" + id, connection);
            try
            {
                connection.Open();
                SqlDataReader reader = await query.ExecuteReaderAsync();
                if (reader.HasRows)
                {

```

```

        if (await reader.ReadAsync())
        {
            UserStatus status = UserStatus.Guest;
            switch ((int)reader.GetValue(5))
            {
                case 1:
                    status = UserStatus.Customer;
                    break;
                case 2:
                    status = UserStatus.Admin;
                    break;
                default:
                    MessageBox.Show("Ошибка чтения данных о
пользователе из БД.\n" +
                                "Вход будет выполненен как гость", "Ошибка",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
                    break;
            }
            // авторизация пользователя
            if (status == UserStatus.Guest)
                User.Logout();
            else User.Login(
                id,
                reader.GetValue(0)?.ToString(),
                reader.GetValue(1)?.ToString(),
                reader.GetValue(2)?.ToString(),
                reader.GetValue(3)?.ToString(),
                reader.GetValue(4)?.ToString(),
                status
            );
        }
    }
    if (!reader.IsClosed)
        reader.Close();
    Close();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (connection != null && connection.State !=
ConnectionState.Closed)
        connection.Close();
}

}
else MessageBox.Show("Некорректные входные данные!", "Ошибка!",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
// загрузка всех данных для входа, т.к. sql - регистронезависимый
private void LoadUsersData()
{
    Cursor = Cursors.WaitCursor;
    usersData = new DataSet();
    SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
    try
    {
        connection.Open();
        SqlDataAdapter adapter = new SqlDataAdapter(@"SELECT Id, Nick, [E-
mail], Password FROM Users", connection);
        adapter.Fill(usersData);
    }
}

```



```

        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
    }
}

private async void buttonCheckout_Click(object sender, EventArgs e)
{
    if (Common.ProductsInCart.Count == 0)
    {
        MessageBox.Show("В корзине нет товаров", "Сообщение",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }
    if (dgv.SelectedRows.Count == 0)
    {
        MessageBox.Show("Выберите товар!", "Товар не выбран",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }
    if (User.GetUser().Status == UserStatus.Guest)
    {
        MessageBox.Show("Вы не вошли в систему. Оформлять заказы могут только
зарегистрированные пользователи.", "Некорректное действие",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }
    if (MessageBox.Show("Вы уверены что хотите оформить этот заказ?",
        "Подтверждение действия",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) !=
        DialogResult.Yes) return;

    int id = (int)dgv.SelectedCells[0].Value;
    int cost = (int)dgv.SelectedCells[2].Value;

    string text = $"INSERT INTO Orders (Customer_id, Product_id, Date,
Amount, Status)
                VALUES ({User.GetUser().Id},{id},{date},{cost},0)";
    var connection = new SqlConnection(Common.StrSQLConnection);
    var query = new SqlCommand(text, connection);
    query.Parameters.AddWithValue("@date", DateTime.Now.ToString());
    try
    {
        await connection.OpenAsync();
        if (await query.ExecuteNonQueryAsync() != 0)
        {
            MessageBox.Show("Заказ успешно оформлен.", "Оформление заказа",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
            id = (int)dgv.SelectedCells[0].Value;

            if (Common.ProductsInCart.Count != 0)
            {
                Common.ProductsInCart.Remove(
                    Common.ProductsInCart.Find(f => f == id));
                dgv.Rows.Remove(dgv.SelectedRows[0]);
            }
        }
        else
        {
            MessageBox.Show("Заказ не был оформлен.", "Оформление заказа",
                MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
    catch (Exception ex)

```

```

        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (connection != null && connection.State != ConnectionState.Closed)
                connection.Close();
        }
    }

    private void buttonDeleteFromCart_Click(object sender, EventArgs e)
    {
        if (Common.ProductsInCart.Count == 0)
        {
            MessageBox.Show("В корзине нет товаров", "Сообщение",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
            return;
        }
        if (dgv.SelectedRows.Count == 0) return;
        if (MessageBox.Show("Вы уверены что хотите удалить товар из корзины?",
            "Подтверждение действия",
            MessageBoxButtons.YesNo, MessageBoxIcon.Question) !=
            DialogResult.Yes) return;

        int id = (int)dgv.SelectedCells[0].Value;

        if (Common.ProductsInCart.Count != 0)
        {
            Common.ProductsInCart.Remove(
                Common.ProductsInCart.Find(f => f == id));
            MessageBox.Show("Товар удален из корзины", "Сообщение",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
            dgv.Rows.Remove(dgv.SelectedRows[0]);
        }
    }
}

```

## Файл ChangeProduct.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Text.RegularExpressions;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class ChangeProduct : Form
    {
        DataGridViewRow Row { set; get; } = null;
        // столбцы в dataGridView
        List<_strToGridView> fields = null;
        // флаги изменения
        // словарь "поле таблицы"->"значение"
        Dictionary<string, object> FieldValue = null;
        // "старое" изображение
        // изображение
        byte[] dataImage = null;
        // команда добавления в бд
    }
}

```

```

string commandText = @"UPDATE Products SET ";

public ChangeProduct()
{
    InitializeComponent();
}

public ChangeProduct(DataGridViewRow row)
{
    InitializeComponent();
    Row = row;
}

private void ChangeProduct_Load(object sender, EventArgs e)
{
    foreach (DataGridViewColumn col in dgv.Columns)
        col.SortMode = DataGridViewColumnSortMode.NotSortable;

    FieldValue = new Dictionary<string, object>(15);
    fields = new List<_strToGridView>(15);

    fields.Add(new _strToGridView("Тип ПК", "Да", "Текст"));
    fields.Add(new _strToGridView("Производитель", "Да", "Текст"));
    fields.Add(new _strToGridView("Модель", "Да", "Текст"));
    fields.Add(new _strToGridView("CPU", "Да", "Текст"));
    fields.Add(new _strToGridView("Кол-во ядер", "Да", "Целое неотрицательное
число"));
    fields.Add(new _strToGridView("GPU", "Да", "Текст"));
    fields.Add(new _strToGridView("Объем RAM", "Да", "Целое неотрицательное
число"));
    fields.Add(new _strToGridView("Тип RAM", "Да", "Текст"));
    fields.Add(new _strToGridView("HDD", "Нет", "Целое неотрицательное
число"));
    fields.Add(new _strToGridView("SSD", "Нет", "Целое неотрицательное
число"));
    fields.Add(new _strToGridView("Операционная система", "Нет", "Текст"));
    fields.Add(new _strToGridView("Блок питания", "Нет", "Текст"));
    fields.Add(new _strToGridView("Склад", "Да", "Целое неотрицательное
число"));
    fields.Add(new _strToGridView("Цена", "Да", "Целое неотрицательное
число"));
    fields.Add(new _strToGridView("Описание", "Нет", "Текст"));

    foreach (var f in fields)
    {
        f.value = Row.Cells[f.name].Value.ToString();
        FieldValue.Add(f.name, f.value);
    }

    // вывод в DataGridView
    foreach (var f in fields)
        dgv.Rows.Add(f.name, f.obligation, f.type, f.value);

    // загрузка изображения из БД
    dataImage = FileTools.GetFileFromDB(Common.StrSQLConnection, @"Products",
@"Изображение", (int)Row.Cells[0].Value);
    if (dataImage != null) pictureBox.Image = Image.FromStream(new
MemoryStream(dataImage));
    if (pictureBox.Image == null) buttonDelImage.Visible = false;
    else buttonDelImage.Visible = true;
}

private void pictureBox_Click(object sender, EventArgs e)
{

```



```

    }

    private void buttonAddImage_Click(object sender, EventArgs e)
    {
        openImage.Filter = "Изображения (*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG";
        if (openImage.ShowDialog() == DialogResult.Cancel) return;
        // получаем файл в виде байтов
        dataImage = FileTools.FileInBytes(openImage.FileName);
        // выводим его в pictureBox
        pictureBox.Image = Image.FromStream(new MemoryStream(dataImage));
        if (pictureBox.Image == null) buttonDelImage.Visible = false;
        else buttonDelImage.Visible = true;
    }

    private async void buttonSave_Click(object sender, EventArgs e)
    {
        if (valid())
        {
            if (MessageBox.Show("Вы уверены, что хотите изменить товар?",
                "Изменение товара",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
                DialogResult.Yes)
            {
                commandText = $"UPDATE Products SET [Тип ПК]=@type,
                [Производитель]=@brand, [Модель]=@model, [CPU]=@cpu, [Кол-во ядер]=@cores, [GPU]=@gpu,
                [Объем RAM]=@ram, [Тип RAM]=@typeram, [HDD]=@hdd, [SSD]=@ssd, [Операционная система]=@os,
                [Блок питания]=@psu, [Склад]=@stock, [Цена]=@cost, [Описание]=@description,
                [Изображение]=@image WHERE Id={(int)Row.Cells[0].Value}";

                // выполнение команды
                SqlConnection connection = new
                SqlConnection(Common.StrSqlConnection);
                SqlCommand query = new SqlCommand(commandText, connection);
                query.Parameters.AddWithValue("@type",
                dgv.Rows[0].Cells[3].Value);
                query.Parameters.AddWithValue("@brand",
                dgv.Rows[1].Cells[3].Value);
                query.Parameters.AddWithValue("@model",
                dgv.Rows[2].Cells[3].Value);
                query.Parameters.AddWithValue("@cpu",
                dgv.Rows[3].Cells[3].Value);
                query.Parameters.AddWithValue("@cores",
                dgv.Rows[4].Cells[3].Value);
                query.Parameters.AddWithValue("@gpu",
                dgv.Rows[5].Cells[3].Value);
                query.Parameters.AddWithValue("@ram",
                dgv.Rows[6].Cells[3].Value);
                query.Parameters.AddWithValue("@typeram",
                dgv.Rows[7].Cells[3].Value);
                if (dgv.Rows[8].Cells[3].Value != null)
                    query.Parameters.AddWithValue("@hdd",
                dgv.Rows[8].Cells[3].Value);
                else query.Parameters.AddWithValue("@hdd", DBNull.Value);
                if (dgv.Rows[9].Cells[3].Value != null)
                    query.Parameters.AddWithValue("@ssd",
                dgv.Rows[9].Cells[3].Value);
                else query.Parameters.AddWithValue("@ssd", DBNull.Value);
                if (dgv.Rows[10].Cells[3].Value != null)
                    query.Parameters.AddWithValue("@os",
                dgv.Rows[10].Cells[3].Value);
                else query.Parameters.AddWithValue("@os", DBNull.Value);
                if (dgv.Rows[11].Cells[3].Value != null)
                    query.Parameters.AddWithValue("@psu",
                dgv.Rows[11].Cells[3].Value);
            }
        }
    }

```

```

        else query.Parameters.AddWithValue("@psu", DBNull.Value);
        query.Parameters.AddWithValue("@stock",
dgv.Rows[12].Cells[3].Value);
        query.Parameters.AddWithValue("@cost",
dgv.Rows[13].Cells[3].Value);
        if (dgv.Rows[14].Cells[3].Value != null)
            query.Parameters.AddWithValue("@descripton",
dgv.Rows[14].Cells[3].Value);
        else query.Parameters.AddWithValue("@descripton", DBNull.Value);
        if (dataImage != null) query.Parameters.AddWithValue("@image",
dataImage);
        else query.Parameters.AddWithValue("@image", DBNull.Value);
        try
        {
            connection.Open();
            if (await query.ExecuteNonQueryAsync() == 1)
            {
                MessageBox.Show("Информация успешно обновлена",
"Сообщение",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
                ProductManagement.updateFlag = true;
                Close();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (connection != null && connection.State !=
ConnectionState.Closed)
                connection.Close();
        }
    }
    Close();
}

private bool valid()
{
    string intPattern = @"^\d+$";

    foreach (DataGridViewRow row in dgv.Rows)
    {
        string sChar = row.Cells[0]?.Value?.ToString();
        string sObligation = row.Cells[1]?.Value?.ToString();
        string sType = row.Cells[2]?.Value?.ToString();
        string sValue = null;
        if (row.Cells[3].Value == null)
            sValue = "";
        else sValue = row.Cells[3]?.Value?.ToString();

        if (string.IsNullOrEmpty(sValue))
        { // если строка пустая или с одними пробелами - пропускаем
            if (sObligation == "Да")
            { // если обязательное поле
                MessageBox.Show($"Поле {sChar} должно быть заполнено",
"Некорректный ввод!",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }
            continue;
        }
    }
}

```

```

        switch (sType)
        {
            case "Целое неотрицательное число":
                Regex regex = new Regex(intPattern);
                if (!regex.IsMatch(sValue))
                {
                    MessageBox.Show($"Поле \"{sChar}\" заполнено не
корректно", "Некорректный ввод!",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return false;
                }
                break;
            case "Текст":
                break;
            default:
                MessageBox.Show("Произошла ошибка при распознавании типа
значения. См. код 'AddNewProduct.valid()', "Ошибка!",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
        }
    }
    return true;
}

private void buttonDelImage_Click(object sender, EventArgs e)
{
    pictureBox.Image = null;
    dataImage = null;
    buttonDelImage.Visible = false;
}

private void buttonCancel_Click_1(object sender, EventArgs e)
{
    Close();
}
}
}

```

## Файл Filters.cs

```

using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class Filters : Form
    {
        List <FilterChecked> filtersChecked = null;
        List <FilterFromTo> filtersFromTo = null;
        List <Field> fields = new List<Field>(13);
        private string SQLCommandToUpdate = null;

        public Filters()
        {
            InitializeComponent();
        }

        private void Filters_Load(object sender, EventArgs e)
        {

```

```

fields.Add(new Field("Тип ПК", RequiredFilter.CheckedList));
fields.Add(new Field("Производитель", RequiredFilter.CheckedList));
fields.Add(new Field("Модель", RequiredFilter.CheckedList));
fields.Add(new Field("CPU", RequiredFilter.CheckedList));
fields.Add(new Field("Кол-во ядер", RequiredFilter.CheckedList));
fields.Add(new Field("GPU", RequiredFilter.CheckedList));
fields.Add(new Field("Объем RAM", RequiredFilter.FromTo));
fields.Add(new Field("Тип RAM", RequiredFilter.CheckedList));
fields.Add(new Field("HDD", RequiredFilter.FromTo));
fields.Add(new Field("SSD", RequiredFilter.FromTo));
fields.Add(new Field("Операционная система",
RequiredFilter.CheckedList));
fields.Add(new Field("Блок питания", RequiredFilter.CheckedList));
fields.Add(new Field("Цена", RequiredFilter.FromTo));

foreach (var field in fields)
    if (field.filter == RequiredFilter.CheckedList)
        field.sqlCommand = @"SELECT DISTINCT [" + field.name + "] FROM
[Products]";

SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
try
{
    connection.Open();
    // для каждого поля делаем фильтр, в зависимости от параметра
    "field.num"
    foreach (Field field in fields)
    {
        SqlCommand command = new SqlCommand(field.sqlCommand,
connection);
        SqlDataAdapter adapter = new SqlDataAdapter(command);

        if (field.filter == RequiredFilter.CheckedList)
        { // если для поля требуется чекбоксы
            // загружаем
            List<string> variants = new List<string>();
            DataSet ds = new DataSet();
            adapter.Fill(ds);
            foreach (DataRow row in ds.Tables[0].Rows)
            {
                if
(string.IsNullOrEmpty(row.ItemArray[0].ToString())) continue;
                else variants.Add(row.ItemArray[0].ToString());
            }
            // создаем фильтры-чекбоксы
            if (filtersChecked == null)
                filtersChecked = new List<FilterChecked>();
            filtersChecked.Add(new FilterChecked(field.name));

            foreach (string variant in variants)
                filtersChecked[filtersChecked.Count -
1].checkedList.Items.Add(variant, true);

            variants.Clear();
        } // иначе - если требуется - фильтр "от и до"
        else if (field.filter == RequiredFilter.FromTo)
        {
            if (filtersFromTo == null)
                filtersFromTo = new List<FilterFromTo>();
            filtersFromTo.Add(new FilterFromTo(field.name));
        }
    }
    // добавление эл-тов на форму
    foreach (var filter in filtersChecked)
        flowLayoutPanel1.Controls.Add(filter.groupBox);
}

```

```

        foreach (var filter in filtersFromTo)
            flowLayoutPanel1.Controls.Add(filter.groupBox);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
    }
}

// применение изменений - создание sql-запроса
private int ApplyChanges()
{
    SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
    try
    {
        SQLCommandToUpdate = @"SELECT * FROM Products";
        // добавление в команду для обновления запрос по фильтрам
        if (_TakeAccountOfFiltersChecked() != -1 &&
            _TakeAccountOfFiltersFromTo() != -1)
        { // если все ок - записываем команду в главной форме
            MainForm.QueryToUpdate = SQLCommandToUpdate;
            return 1;
        }
        else return 0;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
        return -1;
    }
}

private void button2_Click(object sender, EventArgs e)
{
    Close();
}

private void button1_Click(object sender, EventArgs e)
{
    // применяем фильтры
    if (ApplyChanges() == 1)
        Close();
}

private int _TakeAccountOfFiltersChecked()
{
    if (filtersChecked != null)
    {
        // цикл по фильтрам
        foreach (var filter in filtersChecked)
        {
            // если в каком-либо группбоксе не выбран ни один вариант
            if (filter.checkedList.CheckedItems.Count == 0)
            {
                MessageBox.Show("Необходимо выбрать как минимум один вариант
из списка",

```

```

        filter.groupBox.Text, MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
        return -1;
    }
    // пропускаем фильтр, если отмечены все чекбоксы
    if (filter.checkedList.CheckedItems.Count ==
filter.checkedList.Items.Count)
        continue;
    // если команда не была модифицирована на более ранних итерациях
    if (SQLCommandToUpdate.Contains("WHERE"))
        SQLCommandToUpdate += " AND";
    else SQLCommandToUpdate += " WHERE";

    // если отмечено более 1 чекбокса - используем команду IN
    if (filter.checkedList.CheckedItems.Count > 1)
    {
        int i = 0;
        // добавляем название столбца
        SQLCommandToUpdate += " [" + filter.groupBox.Text + "] IN (";

        // добавляем все возможные значения этого столбца
        foreach (var ch in filter.checkedList.CheckedItems)
        {
            SQLCommandToUpdate += "\"" +
filter.checkedList.GetItemText(ch) + "\"";

            if (i++ != filter.checkedList.CheckedItems.Count - 1)
                SQLCommandToUpdate += ", ";
        }
        SQLCommandToUpdate += ")";
    }
    else // иначе - если отмечен только один вариант - используем
команду AND
    {
        // добавляем название столбца
        SQLCommandToUpdate += " [" + filter.groupBox.Text + "]";
        // добавляем значение этого столбца
        SQLCommandToUpdate += "=" + "\" +
filter.checkedList.GetItemText(filter.checkedList.CheckedItems[0]) + "\"";
    }
    }
    return 1;
}
else return 0;
}
private int _TakeAccountOfFiltersFromTo()
{
    if (filtersFromTo != null)
    {
        foreach (var filter in filtersFromTo)
        {
            int f = 0, // левая граница диапазона
            t = 0; // правая граница диапазона

            // пропуск фильтров с пустыми textBox
            if (string.IsNullOrEmpty(filter.from.Text) &&
string.IsNullOrEmpty(filter.to.Text))
                continue;
            // если команда не была модифицирована на более ранних стадиях
            if (SQLCommandToUpdate.Contains("WHERE"))
                SQLCommandToUpdate += " AND";
            else SQLCommandToUpdate += " WHERE";

            if (!string.IsNullOrEmpty(filter.from.Text))

```

```

        {
            if (filter.from.Text.All(char.IsDigit))
            {
                f = int.Parse(filter.from.Text);
            }
            else
            {
                MessageBox.Show("Некорректный ввод в поле \'' +
filter.groupBox.Text + "\'. Допустимы только натуральные числа.", "Ошибка ввода!",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return -1;
            }
        }

        if (!string.IsNullOrEmpty(filter.to.Text))
        {
            if (filter.to.Text.All(char.IsDigit))
            {
                t = int.Parse(filter.to.Text);
            }
            else
            {
                MessageBox.Show("Некорректный ввод в поле \'' +
filter.groupBox.Text + "\'. Допустимы только натуральные числа.", "Ошибка ввода!",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return -1;
            }
        }

        SQLCommandToUpdate += " [" + filter.groupBox.Text + "]";
        if (f > t)
        {
            if (t != 0)
            {
                MessageBox.Show("Некорректный ввод",
filter.groupBox.Text,
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return -1;
            }
            SQLCommandToUpdate += ">=" + f.ToString();
        }
        else if (f == t)
            SQLCommandToUpdate += "=" + f.ToString();
        else if (f < t)
            if (f == 0)
                SQLCommandToUpdate += "<=" + t.ToString();
            else SQLCommandToUpdate += " BETWEEN " + f.ToString() + " AND
" + t.ToString();
    }
    return 1;
}
return 0;
}
}
}

```

## Файл MainForm.cs

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Drawing.Printing;
using System.IO;
using System.Text;
using System.Windows.Forms;
using Excel = Microsoft.Office.Interop.Excel;
using ExcelApplication = Microsoft.Office.Interop.Excel.Application;

namespace AIS_shop
{
    public partial class MainForm : Form
    {
        // Первая инициализация пользователя - как гость
        User user = User.GetUser();
        // SQL-запрос для обновления данных в dataGridView
        public static string QueryToUpdate { set; get; } = null;

        public MainForm()
        {
            InitializeComponent();
        }

        private void MainForm_Load(object sender, EventArgs e)
        {
            administrationToolStripMenuItem.Visible = false;
            buttonFilters.Enabled = false;
            Show();
            Welcome welcome = new Welcome();
            welcome.ShowDialog();
            UpdateControls();
            QueryToUpdate = @"SELECT * FROM Products";
            UpdateDataGridView();
        }

        private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (MessageBox.Show("Закреть программу?", "Выход из программы",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.No)
            {
                e.Cancel = true;
            }
        }

        private async void UpdateDataGridView()
        {
            Cursor = Cursors.WaitCursor;
            progressBar.Value = 0;
            if (string.IsNullOrEmpty(QueryToUpdate))
                QueryToUpdate = @"SELECT * FROM Products";

            string WhereInStock = @" WHERE Склад>0";
            string AndInStock = @" AND Склад>0";
            string Sort = @" ORDER BY Производитель";
            progressBar.Value = 10;
            if (QueryToUpdate.Contains(Sort)) QueryToUpdate =
QueryToUpdate.Replace(Sort, "");
            if (radioButtonStock.Checked)
            {
                // показать товары в наличии
            }
        }
    }
}
```



```

        if (!QueryToUpdate.Contains("WHERE"))
            QueryToUpdate += WhereInStock;
        else
            if (!QueryToUpdate.Contains("Склад>0"))
                QueryToUpdate += AndInStock;
    }
    else
    if (radioButtonAllProduct.Checked)
    {
        // показать все товары
        if (QueryToUpdate.Contains(WhereInStock))
            QueryToUpdate = QueryToUpdate.Replace(WhereInStock, "");
        if (QueryToUpdate.Contains(AndInStock))
            QueryToUpdate = QueryToUpdate.Replace(AndInStock, "");
    }

    // сортировка
    if (!QueryToUpdate.Contains("ORDER BY")) QueryToUpdate += Sort;
    progressBar.Value = 30;

    SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
    try
    {
        progressBar.Value = 35;
        await connection.OpenAsync();
        progressBar.Value = 60;
        SqlDataAdapter sqlAdapter = new SqlDataAdapter(QueryToUpdate,
connection);

        DataSet dataSet = new DataSet();
        sqlAdapter.Fill(dataSet);
        progressBar.Value = 95;
        dataGridView.DataSource = dataSet.Tables[0];
        dataGridView.Columns[0].Visible = false;
        dataGridView.Columns[15].Visible = false;
        dataGridView.Columns[16].Visible = false;
        progressBar.Value = 100;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
        if (dataGridView.DataSource != null) buttonFilters.Enabled = true;
        Cursor = Cursors.Default;
        progressBar.Value = 0;
    }
}

private void UpdateControls()
{
    if (user.Status == UserStatus.Guest)
    {
        loginToolStripMenuItem.Visible = true;
        registerToolStripMenuItem.Visible = true;
        goToPersonalAreaToolStripMenuItem.Visible = false;
        logoutToolStripMenuItem.Visible = false;
        administrationToolStripMenuItem.Visible = false;
        textBoxUser.Visible = false;
    }
    else
    {

```

```

        textBoxUser.Text = $"{user.Surname} {user.Name} {user.Patronymic}";
        textBoxUser.Visible = true;
        if (user.Status == UserStatus.Admin)
            administrationToolStripMenuItem.Visible = true;
        else administrationToolStripMenuItem.Visible = false;
        loginToolStripMenuItem.Visible = false;
        registerToolStripMenuItem.Visible = false;
        goToPersonalAreaToolStripMenuItem.Visible = true;
        logoutToolStripMenuItem.Visible = true;
    }
}

private void dataGridView_CellMouseDoubleClick(object sender,
DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0 && e.ColumnIndex >= 0 && e.Button ==
MouseButtons.Left)
    {
        Cursor = Cursors.WaitCursor;
        Product product = new Product(dataGridView.SelectedRows[0]);
        product.ShowDialog();
        Cursor = Cursors.Default;
    }
}

private void bRefresh_Click(object sender, EventArgs e)
{
    // очистка dataGridView
    dataGridView.DataSource = null;
    // загрузка данных из БД
    if (!string.IsNullOrEmpty(QueryToUpdate))
        UpdateDataGridView();
}

private void buttonFilters_Click(object sender, EventArgs e)
{
    QueryToUpdate = null;
    Filters filters = new Filters();
    filters.ShowDialog();
    UpdateDataGridView();
}

private void buttonCart_Click(object sender, EventArgs e)
{
    if (Common.ProductsInCart.Count == 0)
    {
        MessageBox.Show("Корзина пуста.", "Сообщение",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        Cart cart = new Cart();
        cart.ShowDialog();
    }
}

private void aboutProgramToolStripMenuItem_Click(object sender, EventArgs e)
{
    AboutProgram about = new AboutProgram();
    about.ShowDialog();
}

private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}

```

```

    }

    private void goToPersonalAreaToolStripMenuItem_Click(object sender, EventArgs
e)
    {
        switch (user.Status)
        {
            case UserStatus.Admin:
                AdminProfile adminProfile = new AdminProfile();
                adminProfile.ShowDialog();
                break;
            case UserStatus.Customer:
                Profile profile = new Profile();
                profile.ShowDialog();
                break;
            case UserStatus.Guest:
                MessageBox.Show("Сначала авторизуйтесь.", "Переход в личный
кабинет",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                break;
        }
    }

    private void logoutToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (user.Status != UserStatus.Guest)
        {
            User.Logout();
            Common.ProductsInCart.Clear();
            UpdateControls();
        }
    }

    private void loginToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Authorization auth = new Authorization();
        auth.ShowDialog();
        UpdateControls();
    }

    private void registerToolStripMenuItem_Click(object sender, EventArgs e)
    {
        Registration reg = new Registration();
        reg.ShowDialog();
        UpdateControls();
    }

    private void productManageToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (user.Status == UserStatus.Admin)
        {
            ProductManagement change = new ProductManagement();
            change.ShowDialog();
            UpdateDataGridView();
        }
        else MessageBox.Show("Недостаточно полномочий для завершения этого
действия.",
            "Некорректное действие!",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }

    private void registerNewUserToolStripMenuItem_Click(object sender, EventArgs
e)
    {

```

```

        Registration reg = new Registration();
        reg.ShowDialog();
        UpdateControls();
    }

    private void orderManagerToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (user.Status == UserStatus.Admin)
        {
            Orders orders = new Orders();
            orders.ShowDialog();
        }
        else MessageBox.Show("Недостаточно полномочий для завершения этого
действия.",
            "Некорректное действие!",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
    }

    private void buttonPrint_Click(object sender, EventArgs e)
    {
        if (dataGridView.RowCount == 0)
        {
            MessageBox.Show("Список товаров пуст!", "Ошибка!",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        Document = new PrintDocument();
        Document.PrintPage += new PrintPageEventHandler(printDocument_PrintPage);
        printPreviewDialog = new PrintPreviewDialog
        {
            Width = 1200,
            Height = 1200,
            Document = Document
        };
        if (printDialog.ShowDialog() == DialogResult.OK)
        {
            Document.PrinterSettings = printDialog.PrinterSettings;
            Document.DefaultPageSettings.Landscape = true;
            printPreviewDialog.ShowDialog();
        }
    }

    private void printDocument_PrintPage(object sender, PrintPageEventArgs e)
    {
        Graphics g = e.Graphics;
        int x = 30;
        int y = 30;
        int cell_height = 0;

        int colCount = dataGridView.ColumnCount-2;
        int rowCount = dataGridView.RowCount;

        Font font = new Font("Times New Roman", 8, FontStyle.Regular,
        GraphicsUnit.Point);

        int[] widthC = new int[colCount];

        int current_col = 0;
        int current_row = 0;

        while (current_col < colCount)
        {
            if
(g.MeasureString(dataGridView.Columns[current_col].HeaderText.ToString(), font).Width >
widthC[current_col])

```

```

        {
            widthC[current_col] =
(int)g.MeasureString(dataGridView.Columns[current_col].HeaderText.ToString(), font).Width
+ 20;
        }
        current_col++;
    }

    while (current_row < rowCount)
    {
        while (current_col < colCount)
        {
            if (g.MeasureString(dataGridView[current_col,
current_row].Value.ToString(), font).Width > widthC[current_col])
            {
                widthC[current_col] =
(int)g.MeasureString(dataGridView[current_col, current_row].Value.ToString(), font).Width
+ 20;
            }
            current_col++;
        }
        current_col = 0;
        current_row++;
    }

    current_col = 0;
    current_row = 0;

    string value = "";

    int width = widthC[current_col];
    int height = dataGridView[current_col, current_row].Size.Height;

    Rectangle cell_border;
    SolidBrush brush = new SolidBrush(Color.Black);

    while (current_col < colCount)
    {
        width = widthC[current_col];
        cell_height = dataGridView[current_col, current_row].Size.Height;
        cell_border = new Rectangle(x, y, width, height);
        value = dataGridView.Columns[current_col].HeaderText.ToString();
        g.DrawRectangle(new Pen(Color.Black), cell_border);
        g.DrawString(value, font, brush, x, y);
        x += widthC[current_col];
        current_col++;
    }
    while (current_row < rowCount + 1)
    {
        while (current_col < colCount)
        {
            width = widthC[current_col];
            cell_height = dataGridView[current_col, current_row -
1].Size.Height;

            cell_border = new Rectangle(x, y, width, height);
            value = dataGridView[current_col, current_row -
1].Value.ToString();

            g.DrawRectangle(new Pen(Color.Black), cell_border);
            g.DrawString(value, font, brush, x, y);
            x += widthC[current_col];
            current_col++;
        }
        current_col = 0;
    }

```

```

        current_row++;
        x = 30;
        y += cell_height;
    }
}

private void buttonExportToCSV_Click(object sender, EventArgs e)
{
    if (dataGridView.RowCount == 0)
    {
        MessageBox.Show("Список товаров пуст!", "Ошибка!",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (saveFile.ShowDialog() == DialogResult.OK)
    {
        string fileInStr = null;

        for (int i = 0; i < dataGridView.RowCount; i++)
        {
            for (int j = 0; j < dataGridView.Columns.Count; j++)
                fileInStr += dataGridView[j, i].Value.ToString() + ";";
            fileInStr += "\n";
            progressBar.Value += 90 / dataGridView.RowCount;
        }
        progressBar.Value = 100;
        StreamWriter streamWriter = new StreamWriter(saveFile.FileName,
false,
            Encoding.GetEncoding("Windows-1251"));
        streamWriter.Write(fileInStr);
        streamWriter.Close();
        progressBar.Value = 0;
    }
}

private void buttonExportToExcel_Click(object sender, EventArgs e)
{
    if (dataGridView.RowCount == 0)
    {
        MessageBox.Show("Список товаров пуст!", "Ошибка!",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (saveFile.ShowDialog() == DialogResult.OK)
    {
        ExcelApplication ExcelApp = new ExcelApplication();
        progressBar.Value = 0;
        Excel.Workbook workbook = ExcelApp.Workbooks.Add();
        Excel.Worksheet worksheet = workbook.ActiveSheet;
        ExcelApp.Columns.ColumnWidth = 20;
        ExcelApp.Cells[1, 1] = "Id товара";
        ExcelApp.Cells[1, 2] = "Тип ПК";
        ExcelApp.Cells[1, 3] = "Производитель";
        ExcelApp.Cells[1, 4] = "Модель";
        ExcelApp.Cells[1, 5] = "Процессор";
        ExcelApp.Cells[1, 6] = "Кол-во ядер";
        ExcelApp.Cells[1, 7] = "Видеокарта";
        ExcelApp.Cells[1, 8] = "Объем RAM";
        ExcelApp.Cells[1, 9] = "Тип RAM";
        ExcelApp.Cells[1, 10] = "Объем HDD";
        ExcelApp.Cells[1, 11] = "Объем SSD";
        ExcelApp.Cells[1, 12] = "Операционная система";
        ExcelApp.Cells[1, 13] = "Блок питания";
    }
}

```

```

ExcelApp.Cells[1, 14] = "Кол-во на складе";
ExcelApp.Cells[1, 15] = "Цена";
progressBar.Value += 30;
for (int i = 0; i < dataGridView.Columns.Count - 2; i++)
{
    for (int j = 0; j < dataGridView.Rows.Count; j++)
    {
        ExcelApp.Cells[j + 2, i + 1] = dataGridView[i,
j].Value.ToString();
    }
    progressBar.Value += 66 / dataGridView.Columns.Count - 2;
}
progressBar.Value = 99;
ExcelApp.AlertBeforeOverwriting = false;
ExcelApp.DisplayAlerts = false;
workbook.SaveAs(saveFile.FileName);
ExcelApp.Quit();
progressBar.Value = 100;
saveFile.FileName = "Products_DET_Shop.xlsx";
progressBar.Value = 0;
    }
}
}
}
}

```

## Файл Orders.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class Orders : Form
    {
        private SqlConnection Connection { set; get; } = null;
        private SqlCommand Query { set; get; } = null;
        private SqlDataAdapter Adapter { set; get; } = null;
        private DataSet Data { set; get; } = null;

        public Orders()
        {
            InitializeComponent();
            comboBox.SelectedIndex = 0;
        }

        private void Orders_Load(object sender, EventArgs e)
        {
            UpdateDataGridView(@"SELECT * FROM Orders");

            dataGridView.Columns["Id"].HeaderText = "Номер заказа";
            dataGridView.Columns["Id"].ReadOnly = true;
            dataGridView.Columns["Customer_id"].HeaderText = "Id покупателя";
            dataGridView.Columns["Customer_id"].ReadOnly = true;
            dataGridView.Columns["Product_id"].HeaderText = "Id товара";
            dataGridView.Columns["Product_id"].ReadOnly = true;
            dataGridView.Columns["Date"].HeaderText = "Дата заказа";
            dataGridView.Columns["Date"].ReadOnly = true;
            dataGridView.Columns["Amount"].HeaderText = "Сумма заказа";
            dataGridView.Columns["Amount"].ReadOnly = true;
            dataGridView.Columns["Status"].HeaderText = "Статус заказа";
        }
    }
}

```

```

private void comboBox_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (comboBox.SelectedIndex)
    {
        case 0:
            UpdateDataGridView(@"SELECT * FROM Orders");
            break;
        case 1:
            UpdateDataGridView(@"SELECT * FROM Orders WHERE Status<2");
            break;
    }
}

private async void UpdateDataGridView(string command_text)
{
    Connection = new SqlConnection(Common.StrSQLConnection);
    Query = new SqlCommand(command_text, Connection);
    Adapter = new SqlDataAdapter(Query);
    Data = new DataSet();
    try
    {
        await Connection.OpenAsync();
        Data.Clear();
        Adapter.Fill(Data);
        dataGridView.DataSource = Data.Tables[0];
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (Connection != null && Connection.State != ConnectionState.Closed)
            Connection.Close();
    }
}

private void buttonApplyChanges_Click(object sender, EventArgs e)
{
    if (Adapter == null) return;
    if (MessageBox.Show("Вы уверены, что хотите применить изменения?",
        "Подтверждение действия", MessageBoxButtons.YesNo, MessageBoxIcon.Question) !=
        DialogResult.Yes) return;
    Connection = new SqlConnection(Common.StrSQLConnection);
    try
    {
        SqlCommandBuilder commandBuilder = new SqlCommandBuilder(Adapter);
        Adapter.Update(Data.Tables[0]);
        Data.Clear();
        Adapter.Fill(Data);
        dataGridView.DataSource = Data.Tables[0];
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (Connection != null && Connection.State != ConnectionState.Closed)
            Connection.Close();
    }
}

```



```

        private void dataGridView_DataError(object sender,
DataGridViewDataErrorEventArgs e)
        {
            MessageBox.Show("Проверьте корректность введенных данных. Все поля, кроме
даты, имеют числовой формат. Дата записывается в виде чисел с разделителями.",
"Некорректный ввод", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

```

## Файл Product.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class Product : Form
    {
        DataGridViewRow Row { set; get; } = null;
        User user = User.GetUser();
        int product_id { set; get; }

        public Product(DataGridViewRow row)
        {
            InitializeComponent();
            Row = row ?? throw new ArgumentNullException(nameof(row));
            product_id = (int)Row.Cells[0].Value;
        }

        private void Product_Load(object sender, EventArgs e)
        {
            labelProductName.Text = Row.Cells["Производитель"].Value.ToString() + " "
+ Row.Cells["Модель"].Value.ToString();
            labelCost.Text = "Цена: " + Row.Cells["Цена"].Value.ToString();
            labelInStock.Text = "На складе: " + Row.Cells["Склад"].Value.ToString();
            //
            richTextBoxDescription.Text = "*** Описание не было загружено ***";
            LoadDescription();
            //
            listBoxChars.Items.Add("*** Характеристики товара не загружены ***");
            LoadCharacteristics();
            //
            richTextBoxReviews.Text = "*** Отзывы не были загружены ***";
            LoadReviews();
            //
            LoadPicture("Products", product_id);
            //
            if ((int)Row.Cells["Склад"].Value == 0)
                buttonAddToCart.Enabled = false;

            if (user.Status == UserStatus.Admin)
            {
                buttonAddToCart.Visible = false;
                buttonAddReview.Visible = false;
            }
        }

        private async void LoadReviews()
        {
            SqlConnection connection = new SqlConnection(Common.StrSQLConnection);

```

```

SqlConnection connection2 = new SqlConnection(Common.StrSqlConnection);
SqlDataReader reader = null;
try
{
    string commandText = @"SELECT User_id, Mark, Advantages,
Disadvantages, Comment FROM Reviews WHERE Product_id=" + product_id;
    connection.Open();
    SqlCommand query = new SqlCommand(commandText, connection);
    reader = await query.ExecuteReaderAsync();
    if (reader.HasRows)
    {
        richTextBoxReviews.Clear();
        while (await reader.ReadAsync())
        {
            SqlCommand query2 = new SqlCommand(@"SELECT [Nick] FROM
[Users] WHERE [Id]=" + (int)reader.GetValue(0), connection2);
            connection2.Open();
            string result = query2.ExecuteScalar()?.ToString();
            if (result != null)
                richTextBoxReviews.Text += "=== Отзыв от пользователя " +
result.ToString();
            else
                richTextBoxReviews.Text += "=== Отзыв от неизвестного
пользователя";
            richTextBoxReviews.Text += "\nОценка по 5-бальной шкале: " +
reader.GetValue(1).ToString();
            //
            richTextBoxReviews.Text += "\n---Достоинства:\n";
            if
(!string.IsNullOrWhiteSpace(reader.GetValue(2).ToString()))
                richTextBoxReviews.Text += reader.GetValue(2).ToString();
            else richTextBoxReviews.Text += " *Не указано*";
            //
            richTextBoxReviews.Text += "\n---Недостатки:\n";
            if
(!string.IsNullOrWhiteSpace(reader.GetValue(3).ToString()))
                richTextBoxReviews.Text += reader.GetValue(3).ToString();
            else richTextBoxReviews.Text += " *Не указано*";
            //
            richTextBoxReviews.Text += "\n---Комментарий:\n";
            if
(!string.IsNullOrWhiteSpace(reader.GetValue(4).ToString()))
                richTextBoxReviews.Text += reader.GetValue(4).ToString();
            else richTextBoxReviews.Text += " *Не указано*";
            richTextBoxReviews.Text += "\n-----
-----\n";

        }
    }
    else
    {
        richTextBoxReviews.Text = "*** Отзывов нет ***";
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (connection != null && connection.State != ConnectionState.Closed)
        connection.Close();
}

```

```

        if (connection2 != null && connection2.State !=
        ConnectionState.Closed)
            connection2.Close();
        if (reader != null && !reader.IsClosed) reader.Close();
    }

    private async void LoadCharacteristics()
    {
        SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
        SqlDataReader reader = null;
        try
        {
            string commandText = string.Format(@"SELECT * FROM Products WHERE
            Id={product_id}");
            await connection.OpenAsync();
            SqlCommand query = new SqlCommand(commandText, connection);
            reader = await query.ExecuteReaderAsync();
            if (reader.HasRows)
            {
                if (await reader.ReadAsync())
                {
                    listBoxChars.Items.Clear();
                    for (int i = 1; i < reader.FieldCount-4; i++)
                    {
                        if
                        (string.IsNullOrEmpty(reader.GetValue(i).ToString())) continue;
                        string text = reader.GetName(i).ToString() + ": " +
                        reader.GetValue(i).ToString();
                        listBoxChars.Items.Add(text);
                    }
                }
                else
                {
                    MessageBox.Show("Не удалось загрузить характеристики
                    товара.\n Не удалось прочитать данные из \"SqlDataReader\" ", "Ошибка загрузки",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                    listBoxChars.Items.Clear();
                    listBoxChars.Items.Add("*** Характеристики товара не
                    загружены ***");
                }
            }
            else
            {
                MessageBox.Show("Не удалось загрузить характеристики товара.",
                "Ошибка загрузки",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
                listBoxChars.Items.Clear();
                listBoxChars.Items.Add("*** Характеристики товара не загружены
                ***");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (connection != null && connection.State != ConnectionState.Closed)
                connection.Close();
            if (reader != null && !reader.IsClosed) reader.Close();
        }
    }

```

```

private async void LoadDescription()
{
    SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
    try
    {
        await connection.OpenAsync();
        SqlCommand command = new SqlCommand();
        command.CommandText = string.Format($"SELECT Описание FROM Products
WHERE [Id]={(int)Row.Cells[0].Value}");
        command.Connection = connection;
        object result = await command.ExecuteScalarAsync();
        if (result != null)
        {
            richTextBoxDescription.Text = result.ToString();
        }
        else
        {
            richTextBoxDescription.Text = "*** Описание отсутствует ***";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
    }
}

private void LoadPicture(string tableName, int id)
{
    // загрузка из БД
    Image image = ImageTools.GetImageFromDB(Common.StrSQLConnection,
tableName, "Изображение", id);
    // загрузка изображения в pictureBox
    if (image != null)
        pictureBox.Image = image;
    else pictureBox.Image = Properties.Resources.nofoto;
}

private void buttonAddReview_Click(object sender, EventArgs e)
{
    if (user.Status == UserStatus.Guest)
    {
        MessageBox.Show("Оставлять отзывы могут только авторизованные
пользователи. Войдите или зарегистрируйтесь.", "Некорректное действие",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        return;
    }
    var connection = new SqlConnection(Common.StrSQLConnection);
    var query = new SqlCommand($"SELECT COUNT(Id) FROM Reviews WHERE
Product_id={product_id} AND User_id={user.Id}", connection);
    try
    {
        connection.Open();
        int result = (int)query.ExecuteScalar();
        if (result > 0)
        {
            if (MessageBox.Show("Вы уже добавляли отзыв об этом товаре.
Хотите отредактировать его?", "Сообщение",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)

```

```

        {
            ReviewEditor editReview = new ReviewEditor(product_id);
            editReview.ShowDialog();
        }
    }
    else
    {
        ReviewEditor createReview = new ReviewEditor(product_id);
        createReview.ShowDialog();
    }
    LoadReviews();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}
finally
{
    if (connection != null && connection.State != ConnectionState.Closed)
        connection.Close();
}
}

private void buttonAddToCart_Click(object sender, EventArgs e)
{
    if (Common.ProductsInCart.Find(x => x == product_id) > 0)
    {
        MessageBox.Show("Товар уже в корзине", "Сообщение",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }
    Common.ProductsInCart.Add(product_id);
    MessageBox.Show("Товар успешно добавлен в корзину", "Сообщение",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
}
}
}

```

## Файл ProductManagement.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class ProductManagement : Form
    {
        public static bool updateFlag = false;
        public ProductManagement()
        {
            InitializeComponent();
        }

        private void ProductManager_Load(object sender, EventArgs e)
        {
            UpdateData();
        }

        private void UpdateData()

```

```

        {
            SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
            SqlDataAdapter adapter = new SqlDataAdapter(@"SELECT * FROM Products",
connection);
            DataSet data = new DataSet();
            try
            {
                connection.Open();
                adapter.Fill(data);
                dataGridView.DataSource = data.Tables[0];
                dataGridView.Columns[0].Visible = false;
                dataGridView.Columns[16].Visible = false;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (connection != null && connection.State != ConnectionState.Closed)
                    connection.Close();
            }
        }

        private void buttonAdd_Click(object sender, EventArgs e)
        {
            AddProduct newProduct = new AddProduct();
            newProduct.ShowDialog();
            if (updateFlag)
            {
                UpdateData();
                updateFlag = false;
            }
        }

        private void buttonUpdate_Click(object sender, EventArgs e)
        {
            if (dataGridView.SelectedRows.Count == 1)
            {
                ChangeProduct change = new
ChangeProduct(dataGridView.SelectedRows[0]);
                change.ShowDialog();
                if (updateFlag)
                {
                    UpdateData();
                    updateFlag = false;
                }
            }
            else MessageBox.Show("Вы не выбрали товар для изменения", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

        private void buttonDel_Click(object sender, EventArgs e)
        {
            if (dataGridView.SelectedRows.Count != 1)
            {
                MessageBox.Show("Не выбран товар для удаления", "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            string title = "Подтвердите действие";

```

```

        string qst = "Вы уверены, что хотите удалить выбранный товар из базы  

данных?";
        string recordToDelete = null;
        qst += "\nТовар \n";
        for (int i = 0; i < 4; i++)
        {
            if (i != 0) recordToDelete += " | ";
            recordToDelete += dataGridView.SelectedCells[i].Value.ToString();
        }
        recordToDelete += "\n";
        if (recordToDelete != null) qst += recordToDelete;
        if (MessageBox.Show(qst, title, MessageBoxButtons.YesNo,
        MessageBoxIcon.Question) == DialogResult.Yes)
        {
            SqlConnection connection = new
        SqlConnection(Common.StrSQLConnection); ;
            try
            {
                connection.Open();
                SqlCommand query = new SqlCommand();
                query.CommandText = string.Format($"DELETE FROM Products WHERE  

Id={dataGridView.SelectedCells[0].Value}");
                query.Connection = connection;
                if (query.ExecuteNonQuery() == 1)
                {
                    MessageBox.Show($"Удаление записи \"{recordToDelete}\"  

успешно выполнено", "Операция выполнена",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
                    UpdateData();
                }
                else MessageBox.Show($"Удаление записи \"{recordToDelete}\" не  

выполнено", "Операция невыполнена",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (connection != null && connection.State !=
        ConnectionState.Closed)
                    connection.Close();
            }
        }
        else dataGridView.ClearSelection();
    }
}

```

### Файл Profile.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class Profile : Form
    {

```

```

byte[] dataImage { get; set; } = null;
bool Unsaved = false;

public Profile()
{
    InitializeComponent();
}

private void Profile_Load(object sender, EventArgs e)
{
    LoadInfo();
    LoadOrders();
}

private void Profile_FormClosing(object sender, FormClosingEventArgs e)
{
    if (Unsaved)
    {
        DialogResult result =
            MessageBox.Show("Сохранить изменение изображения?",
"Подтверждение действия",
            MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);
        if (result == DialogResult.Yes)
            if (dataImage != null)
                if (FileTools.PutBytesToDB(dataImage,
Common.StrSQLConnection, @"Users", @"Picture", User.GetUser().Id))
                    MessageBox.Show("Файл успешно загружен в базу данных.",
"Сообщение",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
                else MessageBox.Show("Файл не был загружен в базу данных.",
"Сообщение",
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            else DeleteImage();

        if (result == DialogResult.Cancel) e.Cancel = true;
    }
}

private void LoadInfo()
{
    User user = User.GetUser();
    if (user.Status == UserStatus.Guest)
    {
        MessageBox.Show("Вы не аторизованы", "Вход не выполнен",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        Close();
    }
    textBoxSurname.Text = user.Surname;
    textBoxName.Text = user.Name;
    if (!string.IsNullOrEmpty(user.Patronymic))
        textBoxPatronymic.Text = user.Patronymic;
    else textBoxPatronymic.Text = "(не указано)";
    textBoxEmail.Text = user.Email;
    textBoxNick.Text = user.Nick;
    dataImage = FileTools.GetFileFromDB(Common.StrSQLConnection, "Users",
"Picture", user.Id);
    if (dataImage == null)
    {
        pictureBox.Image = Properties.Resources.nofoto;
        buttonDelImage.Visible = false;
    }
    else pictureBox.Image = Image.FromStream(new MemoryStream(dataImage));
}

private async void LoadOrders()

```



```

    {
        string text = $"
            SELECT CONCAT(pr.Производитель, ' ', pr.Модель), ord.Date,
ord.Amount, ord.Status
            FROM Products AS pr, Orders AS ord
            WHERE (pr.Id=ord.Product_id) AND
(ord.Customer_id={User.GetUser().Id})";
        SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
        SqlDataAdapter adapter = new SqlDataAdapter(text, connection);
        DataSet ds = new DataSet();
        try
        {
            await connection.OpenAsync();
            adapter.Fill(ds);
            foreach (DataRow i in ds.Tables[0].Rows)
                dgv.Rows.Add(i.ItemArray);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (connection != null && connection.State != ConnectionState.Closed)
                connection.Close();
        }
    }

    private void linkNewImage_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
    {
        openNewImage.Filter = "Изображения
(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG";
        if (openNewImage.ShowDialog() == DialogResult.Cancel) return;
        // получаем файл в виде байтов
        dataImage = FileTools.FileInBytes(openNewImage.FileName);
        // выводим его в pictureBox
        pictureBox.Image = Image.FromStream(new MemoryStream(dataImage));
        if (dataImage != null)
        {
            Unsaved = true;
            buttonDelImage.Visible = true;
        }
    }

    private void buttonDelImage_Click(object sender, EventArgs e)
    {
        dataImage = null;
        pictureBox.Image = Properties.Resources.nofoto;
        Unsaved = true;
        buttonDelImage.Visible = false;
    }

    private void DeleteImage()
    {
        SqlConnection connection = new SqlConnection(Common.StrSQLConnection);
        SqlCommand sqlCommand = new SqlCommand($"UPDATE Users SET Picture=NULL
WHERE Id={User.GetUser().Id}", connection);
        try
        {
            connection.Open();
            if (sqlCommand.ExecuteNonQuery() == 1)
                MessageBox.Show("Изображение удалено", "Сообщение",

```



```

command.CommandText = $"INSERT INTO
    Users (Surname, Name, Patronymic, [E-mail], Nick, Password,
Status)
    VALUES (@surname, @name, @patronymic, @email, @nick,
@password, @status)";

command.Parameters.AddWithValue("@surname", textBoxSurname.Text);
command.Parameters.AddWithValue("@name", textBoxName.Text);
command.Parameters.AddWithValue("@patronymic",
textBoxPatronymic.Text);
command.Parameters.AddWithValue("@email", textBoxEmail.Text);
command.Parameters.AddWithValue("@nick", textBoxNick.Text);
command.Parameters.AddWithValue("@password",
textBoxPassword.Text);
command.Parameters.AddWithValue("@status", status);

if (await command.ExecuteNonQueryAsync() != 1)
{
    MessageBox.Show("Пользователь не был зарегистрирован",
"Ошибка при регистрации",
    MessageBoxButtons.OK, MessageBoxIcon.Information);
    return;
}
else
    MessageBox.Show("Пользователь зарегистрирован", "Сообщение",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
SqlCommand query = new SqlCommand($"SELECT Id, Surname, Name,
Patronymic, [E-mail], Nick, Status FROM Users WHERE Nick='{textBoxNick.Text}';",
connection);

SqlDataReader reader = await query.ExecuteReaderAsync();
if (reader.HasRows)
{
    if (await reader.ReadAsync())
    {
        UserStatus userStatus = UserStatus.Guest;
        switch ((int)reader.GetValue(6))
        {
            case 1:
                userStatus = UserStatus.Customer;
                break;
            case 2:
                userStatus = UserStatus.Admin;
                break;
            default:
                MessageBox.Show("Ошибка чтения данных о
пользователе из БД.\nВход будет выполнен как гость", "Ошибка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
                break;
        }
        // авторизация пользователя
        User user = User.Login(
            (int)reader.GetValue(0),
            reader.GetValue(1)?.ToString(),
            reader.GetValue(2)?.ToString(),
            reader.GetValue(3)?.ToString(),
            reader.GetValue(4)?.ToString(),
            reader.GetValue(5)?.ToString(),
            userStatus
        );
    }
}
if (!reader.IsClosed)
    reader.Close();
Close();
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            if (connection != null && connection.State !=
ConnectionState.Closed)
                connection.Close();
        }
        Close();
    }

    private bool validation()
    {
        try
        {
            connection = new SqlConnection(Common.StrSQLConnection);
            connection.Open();
            Regex regex = null;
            // если не введены необходимые данные
            if (string.IsNullOrEmpty(textBoxName.Text) ||
                string.IsNullOrEmpty(textBoxSurname.Text) ||
                string.IsNullOrEmpty(textBoxEmail.Text) ||
                string.IsNullOrEmpty(textBoxNick.Text) ||
                string.IsNullOrEmpty(textBoxPassword.Text))
            {
                MessageBox.Show("Все поля, кроме отчества, обязательны для
заполнения", "Некорректный ввод",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (!textBoxName.Text.All(char.IsLetter))
            {
                MessageBox.Show("Имя может состоять только из букв",
"Некорректный ввод",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (!textBoxSurname.Text.All(char.IsLetter))
            {
                MessageBox.Show("Фамилия может состоять только из букв",
"Некорректный ввод",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (!textBoxPatronymic.Text.All(char.IsLetter))
            {
                MessageBox.Show("Отчество может состоять только из букв",
"Некорректный ввод",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            regex = new Regex(@"^([a-z0-9]|_){4,}$");
            if (!regex.IsMatch(textBoxNick.Text))
            {
                MessageBox.Show("Никнейм может состоять только из латинских букв,
цифр и нижнего подчеркивания и при этом иметь не менее 4 символов", "Некорректный ввод",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }
            if (!IsValidEmail(textBoxEmail.Text))
            {

```

```

        MessageBox.Show("Некорректный e-mail адрес", "Некорректный ввод",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }
    regex = new Regex(@"^([a-z0-9]|_){4,}$");
    if (!regex.IsMatch(textBoxPassword.Text))
    {
        MessageBox.Show("Пароль может состоять только латинских из букв,
            цифр и нижнего подчеркивания и при этом иметь не менее 4 символов", "Некорректный ввод",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }
    // проверка на совпадение введенного e-mail с e-mail других
пользователей
    SqlCommand query = new SqlCommand($"SELECT COUNT(Id) FROM Users
WHERE [E-mail]='{textBoxEmail.Text}'", connection);
    int answer = -1;
    answer = Convert.ToInt32(query.ExecuteScalar());
    if (answer > 0 || answer == -1)
    {
        if (answer == -1)
            MessageBox.Show("Произошла ошибка при сверке e-mail с
            базой.", "Ошибка!",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        else MessageBox.Show("Пользователь с таким e-mail уже
            зарегистрирован!", "Ошибка!",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }
    // проверка на совпадение введенного ника с никами другими
пользователями
    query = new SqlCommand($"SELECT COUNT(Id) FROM Users WHERE
[Nick]='{textBoxNick.Text}'", connection);
    answer = -1;
    answer = Convert.ToInt32(query.ExecuteScalar());
    if (answer > 0 || answer == -1)
    {
        if (answer == -1)
            MessageBox.Show("Произошла ошибка при сверке ника с базой.",
            "Ошибка!",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        else MessageBox.Show("Пользователь с таким ником уже
            зарегистрирован!", "Ошибка!",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }
    return true;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    return false;
}
finally
{
    if (connection != null && connection.State != ConnectionState.Closed)
        connection.Close();
}
}

private bool IsValidEmail(string email)
{
    if (string.IsNullOrEmpty(email))

```

```

        return false;

    try
    {
        // Normalize the domain
        email = Regex.Replace(email, @"(@)(.+)$", DomainMapper,
                               RegexOptions.None,
TimeSpan.FromMilliseconds(200));

        // Examines the domain part of the email and normalizes it.
        string DomainMapper(Match match)
        {
            // Use IdnMapping class to convert Unicode domain names.
            var idn = new IdnMapping();

            // Pull out and process domain name (throws ArgumentException on
invalid)
            var domainName = idn.GetAscii(match.Groups[2].Value);

            return match.Groups[1].Value + domainName;
        }
    }
    catch (RegexMatchTimeoutException)
    {
        return false;
    }
    catch (ArgumentException)
    {
        return false;
    }

    try
    {
        return Regex.IsMatch(email,
            @"^(?("")("".+?(?!\\"""")@"|((([0-9a-z]([\.?!\\.)|[-
!#\$\%&'\*\+\/=\?\^\`\{\}\|\~\w])*)(?<=[0-9a-z])@))" +
            @"(?:\([\]\[\d{1,3}\.]{3}\d{1,3}\)|((([0-9a-z](-[0-9a-
z]*\.)+[a-z0-9](-a-z0-9){0,22}[a-z0-9]))))$",
            RegexOptions.IgnoreCase, TimeSpan.FromMilliseconds(250));
    }
    catch (RegexMatchTimeoutException)
    {
        return false;
    }
}
}
}

```

## Файл ReviewEditor.cs

```

using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class ReviewEditor : Form
    {
        User user = User.GetUser();
        int Product_id { set; get; } = 0;
        int Mark { set; get; } = 0;
        string Advantages { set; get; } = null;
    }
}

```

```

string Disadvantages { set; get; } = null;
string Comment { set; get; } = null;
bool fCreateReview = false;

public ReviewEditor(int product_id)
{
    InitializeComponent();
    Product_id = product_id;
}

private void ReviewEditor_Load(object sender, EventArgs e)
{
    LoadData();
}

private async void LoadData()
{
    var connection = new SqlConnection(Common.StrSqlConnection);
    string text = $"SELECT CONCAT(Производитель, ' ', Модель) AS Name FROM
Products WHERE Id={Product_id}";
    SqlCommand query = new SqlCommand(text, connection);
    SqlDataReader reader = null;
    try
    {
        await connection.OpenAsync();
        reader = await query.ExecuteReaderAsync();
        if (reader.HasRows)
        {
            if (await reader.ReadAsync())
                labelProductName.Text = reader.GetValue(0).ToString();
        }
        reader.Close();
        reader = null;

        query.CommandText = $"SELECT Mark, Advantages, Disadvantages,
Comment FROM Reviews WHERE Product_id={Product_id} AND User_id={user.Id}";
        reader = await query.ExecuteReaderAsync();
        if (reader.HasRows)
        {
            if (await reader.ReadAsync())
            {
                // загрузка из БД
                if ((int)reader.GetValue(0) > 5)
                    Mark = 5;
                else
                if ((int)reader.GetValue(0) < 1)
                    Mark = 1;
                else Mark = (int)reader.GetValue(0);
                Advantages = reader.GetValue(1).ToString();
                Disadvantages = reader.GetValue(2).ToString();
                Comment = reader.GetValue(3).ToString();
                // загрузка в форму
                numericUpDownMark.Value = Mark;
                richTextBoxAdvantages.Text = Advantages;
                richTextBoxDisadvantages.Text = Disadvantages;
                richTextBoxComment.Text = Comment;
            }
        }
        else fCreateReview = true;
        reader.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

    }
    finally
    {
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
        if (reader != null && !reader.IsClosed) reader.Close();
    }
}

private async void buttonApplyChanges_Click(object sender, EventArgs e)
{
    var connection = new SqlConnection(Common.StrSQLConnection);
    string text = null;

    if (fCreateReview)
        text = $"INSERT INTO Reviews (Product_id, User_id, Mark, Advantages,
Disadvantages, Comment)
                VALUES ({Product_id}, {user.Id},
{numericUpDownMark.Value}, @adv, @disadv, @comm)";
    else
        text = $"UPDATE Reviews SET Mark={numericUpDownMark.Value},
Advantages=@adv, Disadvantages=@disadv, Comment=@comm WHERE Product_id={Product_id} AND
User_id={user.Id}";

    SqlCommand query = new SqlCommand(text, connection);
    query.Parameters.AddWithValue("@adv",
richTextBoxAdvantages.Text.Replace("'", "''"));
    query.Parameters.AddWithValue("@disadv",
richTextBoxDisadvantages.Text.Replace("'", "''"));
    query.Parameters.AddWithValue("@comm",
richTextBoxComment.Text.Replace("'", "''"));
    try
    {
        await connection.OpenAsync();
        if (await query.ExecuteNonQueryAsync() == 1)
        {
            MessageBox.Show("Операция успешно выполнена.", "Сообщение",
MessageBoxButtons.OK, MessageBoxIcon.Information);
            Close();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), ex.Source.ToString(),
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    finally
    {
        if (connection != null && connection.State != ConnectionState.Closed)
            connection.Close();
    }
}
}
}
}

```



## Файл Welcome.cs

```
using System;
using System.Windows.Forms;

namespace AIS_shop
{
    public partial class Welcome : Form
    {
        public Welcome()
        {
            InitializeComponent();

            // Кнопка "Перейти в каталог"
            private void button1_Click(object sender, EventArgs e)
            {
                Close();
            }
            // Кнопка "Войти"
            private void button2_Click(object sender, EventArgs e)
            {
                Authorization auth = new Authorization();
                auth.ShowDialog();
                Close();
            }
            // Кнопка "Регистрация"
            private void button3_Click(object sender, EventArgs e)
            {
                Registration reg = new Registration();
                reg.ShowDialog();
                Close();
            }

            private void Welcome_Load(object sender, EventArgs e)
            {
            }
        }
    }
}
```