Федеральное государственное бюджетное образовательное учреждение высшего образования

«Армавирский государственный педагогический университет»

Институт прикладной информатики, математики и физики

Кафедра информатики и ИТО

**КУРСОВАЯ РАБОТА**

направление подготовки 09.03.01 «Информатика и вычислительная техника»

направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем»

по дисциплине «**Базы данных**»

**Тема «Магазин компьютерной техники»**

Выполнил:

студент группы ВМ-ИВТ-3-1,

ИПИМиФ, очной формы обучения

**ЗИНЧЕНКО Кирилл Алексеевич**

Руководитель:

старший преподаватель кафедры информатики и ИТО

**ЛАПШИН Николай Александрович**

АРМАВИР, 2021

Содержание

[Введение 1](#_Toc73528382)

[1. Аналитическая часть 2](#_Toc73528383)

[1.1. Обзор предметной области 2](#_Toc73528384)

[1.2. Выбор средств и среды разработки 4](#_Toc73528385)

[2. Проектная часть. 6](#_Toc73528386)

[2.1. Разработка базы данных. 6](#_Toc73528387)

[2.2. Разработка программы 8](#_Toc73528388)

[Заключение 35](#_Toc73528389)

[Список литературы 36](#_Toc73528390)

### Введение

**Актуальность**

Приложение для магазина компьютерной техники является очень актуальным, так как в наше время люди привыкли совершать покупки онлайн. И производить доставку товаров на дом в период пандемии это оказалось особенно актуальным. Разработка приложения для магазина компьютерной техники позволяет поработать над многими функциями и возможностями. Так же учит разработчика не перезагружать приложение, оставляя только самое необходимое. Позволяет поработать на всех фронтах, таких как фронт-энд разработка при создании дизайна, кнопок и общего стиля приложения. И так же позволяет поработать над бэк-энд разработкой. Так же разработка приложения для магазина компьютерной техники не может быть реализована без разработки базы данных для хранения и работы с данными магазина.

**Цели**

Целью данного курсового проекта является проверка навыков полученных за время обучения и создание приложения и базы данных и последующее улучшение на всем этапе обучения, а именно создание магазина компьютерной техники. Данное приложение можно использовать в магазинах для поисков необходимой продукции.

**Задачи**

* Проверка теоретических навыков полученных за время обучения и применение их на практике
* Написание приложения
* Тестирование
* Отладка
* Обновление
* Подведение общего итога

**Объект**

Объектом данной курсовой работы является разработка база данных и приложение магазин компьютерной техники.

**Предмет**

Предметом данной курсовой работы является база данных и приложение магазин компьютерной техники.

**Практическая значимость.**

Практическая значимость данной курсовой работы заключается в освоении и закреплении практических навыков на примере создания базы данных и приложения связанного с ней.

### Аналитическая часть

### Обзор предметной области

В данном разделе будут рассмотрены отдельные элементы.

Элемент Button, а в нашем случае кнопка «Готовые ПК» отвечает за сортировку товаров по категории «Готовые ПК».

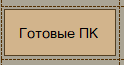


Рисунок 1.1.1 «Готовые ПК»

Реализация:

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

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products where Category = 'PC'", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

Таблица DataGridView «Все пользователи» отвечает за отображение списка покупателей.

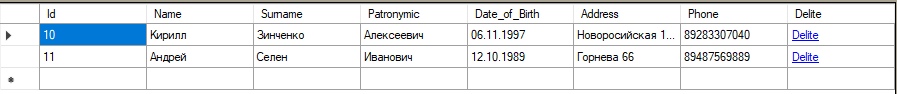
Реализация:

Рисунок 1.1.2 «Все пользователи»

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Users", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView3.DataSource = set1.Tables[0];

LoadUsers();

Контекстное меню MenuStrip

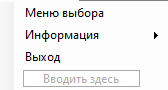


Рисунок 1.1.3 «Контекстное меню»

Реализация:   
private void выходToolStripMenuItem\_Click(object sender, EventArgs e) // закрытие формы

{

Application.Exit();

}

### Выбор средств и среды разработки

Выбор среды программирования пал на VisualStudio а именно на VisualStudio2019.

Во первых VisualStudio2019 имеет бесплатную версию которая и используется для обучения студентов на начальном этапе.

VisualStudio - это комбинация нескольких важнейших технологий:

* Высокопроизводительный компилятор в машинный код;
* Объектно-ориентированная модель компонент;
* Визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;
* Масштабируемые средства для построения баз данных;

Компилятор, встроенный в VisualStudio, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре “клиент-сервер”. Этот компилятор в настоящее время является самым быстрым в мире, его скорость компиляции составляет свыше 120 тысяч строк в минуту на компьютере 486DX33. Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения (4GL) и в то же время обеспечивает качество кода, характерного для компилятора 3GL. Кроме того, VisualStudio обеспечивает быструю разработку.

В последнее время компьютерные технологии начали свое стремительное развитие, появление мощнейших графических ускорителей и центральных процессоров способствовало не менее бурному развитию индустрии компьютерных игр. Выдающиеся разработки этой отрасли – это сложнейшие программы, как правило, с очень высокими требованиями к аппаратной части компьютера. Но, и существует великое множество небольших компьютерных игр, которые также популярны. Для возможности отдохнуть в перерыве от выполнения какой-либо работы оператору компьютера не всегда требуется новейшая компьютерная игра, а зачастую использовать её не позволяет маломощное оборудование офисного компьютера. Именно этой цели – отдыху от монотонной работы служит разработанная в рамках данного курсового проекта программа «Математический бой », которая и будет являться объектом моего исследования.

VisualStudio — это прекрасная система визуального объектно-ориентированного программирования, одинаково радующая и новичков, и опытных пользователей. Новичку она позволяет с небольшими затратами сил и времени создавать прикладные программы, которые внешне неотличимы от программ, созданных профи, и удовлетворяют всем требованиям Операционной системы. Впрочем, теперь в VisualStudio имеется возможность создавать кросс-платформенные программы, которые могут работать и в среде Windows, и в среде Linux. Ну а для опытного программиста VisualStudio открывает неограниченные возможности создавать сколь угодно сложные, приложения, работающие с любыми данными и файлами.

VisualStudio2019 позволяющих создавать как отдельные прикладные программы, так и разветвленные комплексы, предназначенные для работы в корпоративных сетях и в сети «Интернет». Система объектно-ориентированного программирования в VisualStudio2019 достаточна, понятна и удобна в использовании.

ООП - в наше время совершенно естественный подход к построению программных систем любого уровня. Когда вы открываете любую программу на вашем компьютере, вы видите окно с множеством кнопок, разделов меню, окон редактирования, списков и т.п. Все это объекты. Причем сами по себе они ничего не делают. Они ждут каких-то событий — нажатия пользователем клавиш или кнопок мыши, перемещения курсора и т.д. Когда происходит подобное событие, объект получает сообщение об этом и как-то на него реагирует: выполняет некоторые вычисления, разворачивает список, заносит символ в окно редактирования. Вот такая программа Windows и есть объектно-ориентированная программа.

Приложение, построенное по принципам объектной ориентации — это не последовательность каких-то операторов, не некий жесткий алгоритм. Объектно-ориентрованная программа — это совокупность объектов и способов их взаимодействия. Отдельным (и главным) объектом при таком подходе во многих случаях можно считать пользователя программы. Он же служит и основным, но не единственным, источником событий, управляющих приложением.

### Проектная часть.

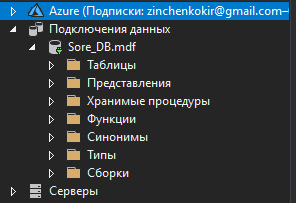
### Разработка базы данных.

База данных, основанная на службах или SQLite это локальная база данных. SQLite — компактная [встраиваемая СУБД](https://ru.wikipedia.org/wiki/%D0%92%D1%81%D1%82%D1%80%D0%B0%D0%B8%D0%B2%D0%B0%D0%B5%D0%BC%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94). [Исходный код](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%85%D0%BE%D0%B4%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) библиотеки передан в [общественное достояние](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B4%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5). В [2005 году](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4).

SQLite поддерживает [динамическое типизированние](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D1%82%D0%B8%D0%BF%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) данных. Возможные типы значений: INTEGER, REAL, TEXT и [BLOB](https://ru.wikipedia.org/wiki/BLOB). Также поддерживается специальное значение [NULL](https://ru.wikipedia.org/wiki/NULL_(SQL)).

Размеры значений типа TEXT и BLOB не ограничены ничем, кроме константы SQLITE\_MAX\_LENGTH в исходниках sqlite, равной миллиарду (109).

Каждое значение в любом поле любой записи может быть любого из этих типов, независимо от типа, указанного при объявлении полей таблицы. Указанный при объявлении поля тип хранится для справки в его исходном написании, и используется в качестве основы для выбора предпочтений (так называемое «type affinity»: это подход, редко встречающийся в других СУБД) при выполнении неявных преобразований типов на основании похожести этого названия типа на что-либо, знакомое SQLite. В этот алгоритм зашит обширный перечень практикуемых в других СУБД вариантов названий типов данных. Если безопасного преобразования записываемого значения в предпочитаемый тип не получается, SQLite записывает значение в его исходном виде. Для получения значений из базы есть ряд функций для каждого из типов, и если тип хранимого значения не соответствует запрашиваемому, оно тоже, по возможности, преобразуется. Разработка базы данных производится через обозреватель серверов.

Рисунок 2.1.1 «Обозреватель серверов»

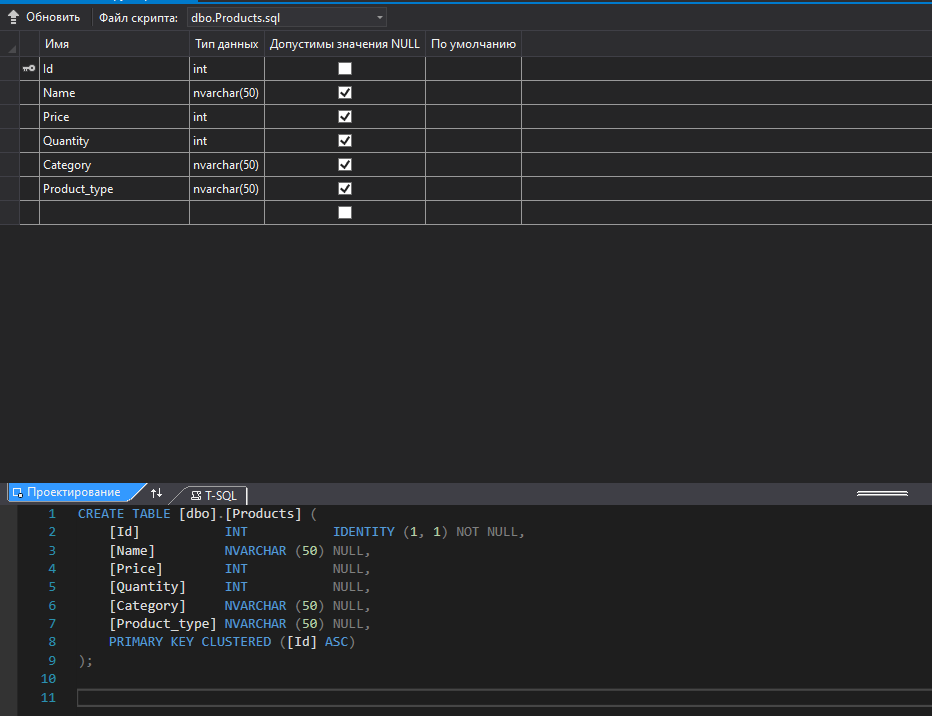
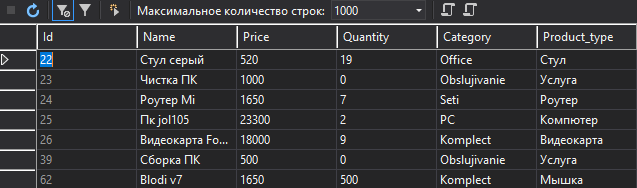
Для создания таблиц необходимо перейти во вкладку таблицы и создать новую. Далее необходимо ввести столбцы и указать тип данных. После создания столбов необходимо ввести название таблицы.

Рисунок 2.1.2 «Создание таблицы»

После создание таблицы ее можно заполнять или использовать для заполнения в приложении.

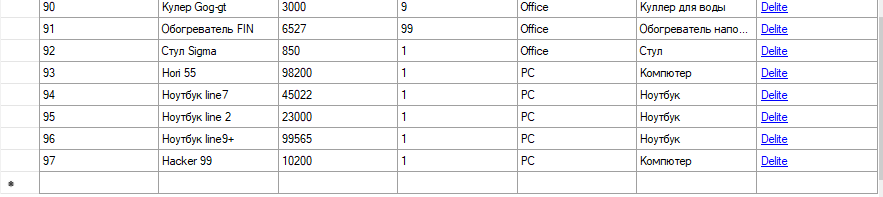
Рисунок 2.1.3 «Заполнение таблицы в VisualSudio»

Рисунок 2.1.4 «Заполнение таблицы в приложении»

* 1. Разработка программы

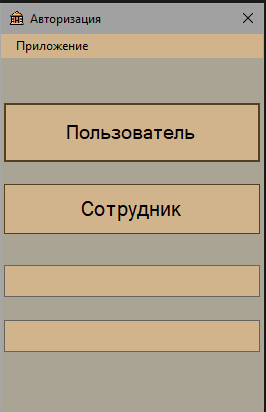
При запуске приложения пользователь видит две кнопки и два поля для ввода дынных. Сверху слева было реализовано Main Menu.

Рисунок 2.2.1 «Поля авторизации»

При нажатии на «Приложение» пользователь увидит три вкладки:

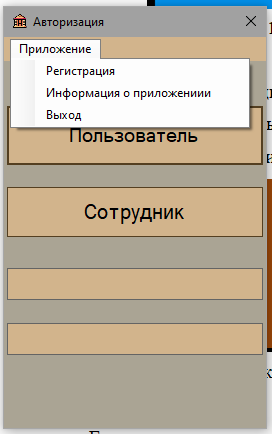
* Регистрация
* Информация о приложении
* Выход

Рисунок 2.2.2 «Приложение»

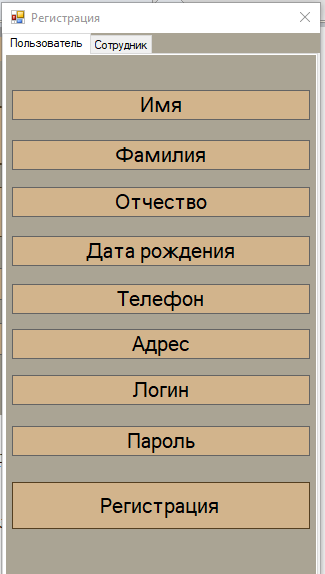
При нажатии на вкладку «Регистрация» откроется новое окно, в котором пользователь сможет произвести регистрацию.

Рисунок 2.2.3 «Вкладка регистрация»

В данном окне есть две вкладки:

* Пользователь
* Сотрудник

Обе вкладки имеют поля для ввода данных пользователя и кнопку «Регистрация» после нажатия, на которую окно регистрации буде закрыто и пользователь вернется на первое окно, которое загружается при запуске приложения.

После регистрации пользователь может выбрать одну из вкладок. Для начала посмотрим на вкладку «Сотрудник».

После нажатия мы увидим окно, в котором будут некоторые объекты:

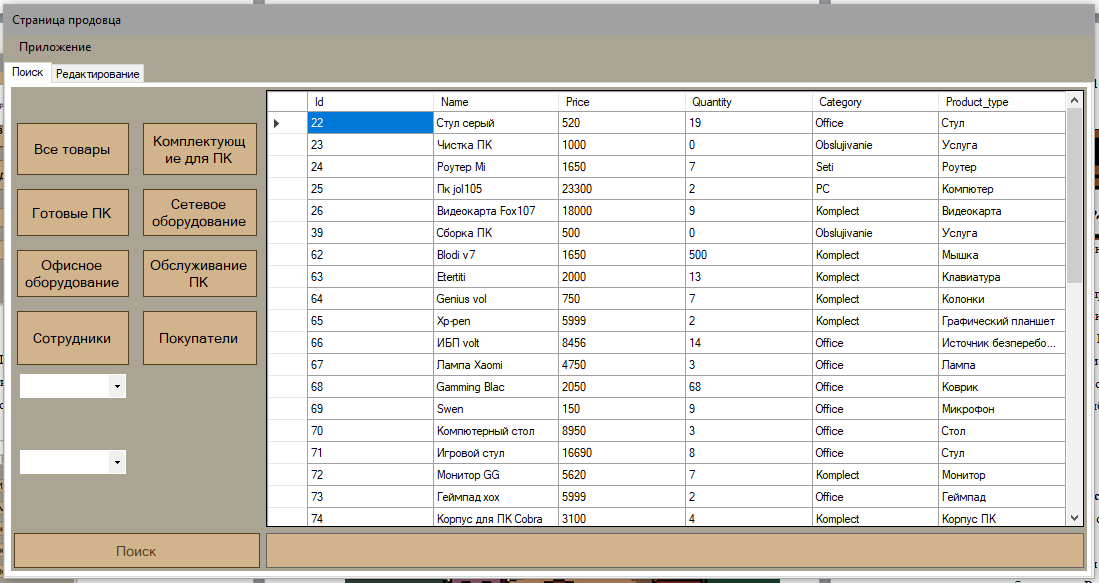
* Вкладка «Приложение»
* Панели «Поиск» и «Редактирование»
* Кнопки для работы и сортировки данных
* Панели для ввода данных
* Таблица с данными

Рисунок 2.2.4 «Окно сотрудника»

В панели поиск можно производить поиск и сортировку данных таких как:

* Товары, имеющиеся в магазине
* Данные покупателей
* Данные сотрудников магазина

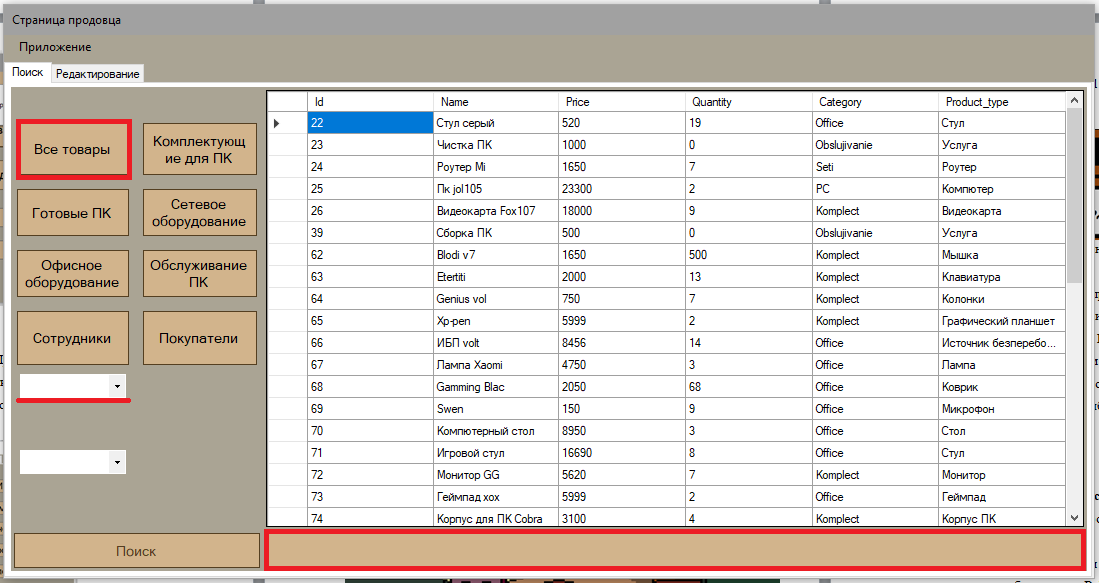
Сортировка данных производится несколькими способами по использованию категорий товаров, цене, количеству и названию товара.

Рисунок 2.2.5 «Виды сортировок и поиска товаров»

При открытии окна мы видим список всех товаров. При использовании поисковой строки мы можем найти товары по их названию.

С помощью кнопок «Сотрудники» и «Покупатели» можно выбрать соответствующие данные и произвести несколько видов сортировок.

Помимо этого в окне «Сотрудник» как и во всех основных окнах, есть вкладка «Приложение». Она содержит пункты:

* Меню выбора
* Информация
* Выход

При нажатии на «Меню выбора» мы вернемся на первое окно. На вкладке выход приложение закрывается. Вкладка «Информация» имеет дополнительные подпункты:

* О приложении
* Версия

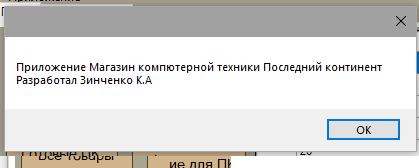
Вкладка «О приложении» выводит сообщение с описанием приложения. Вкладка версия показывает текущую версию приложения.

Рисунок 2.2.6 «Вкладка “О приложении”»

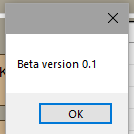


Рисунок 2.2.7 «Вкладка “Версия”»

Сортировка в приложении производится несколькими один из более простых и быстрых это выбор по самым популярным категориям товаров:

* Все товары
* Комплектующие для ПК
* Готовые ПК
* Сетевое оборудование
* Офисное оборудование
* Обслуживание ПК
* Сотрудники
* Покупатели

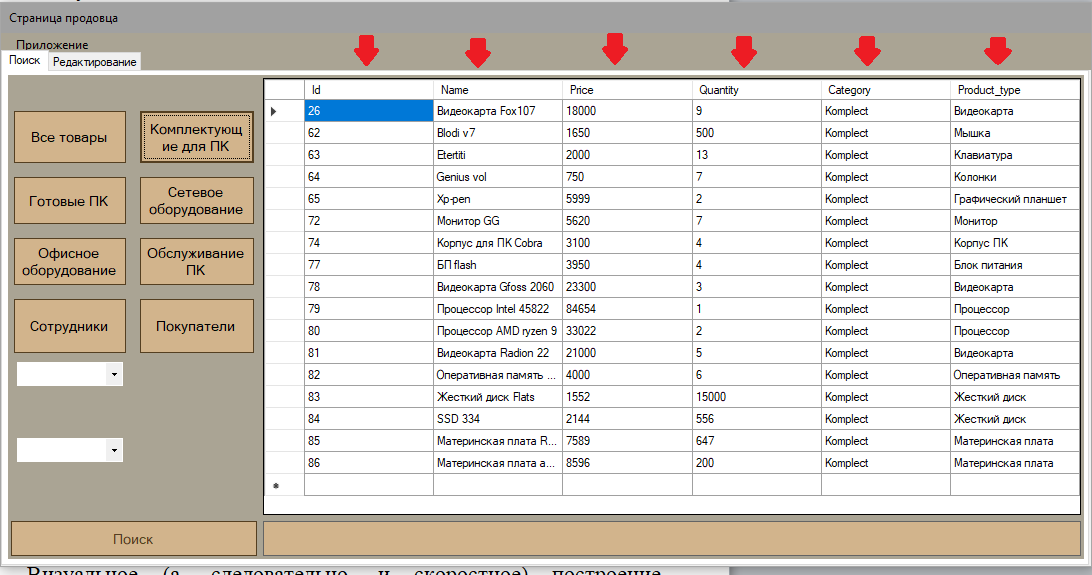
Для примера сделаем выбор вкладки «Комплектующие для ПК». При выборе мы видим список товаров, которые имеют категорию «Komplekt». Сортировку можно произвести по возрастанию и убыванию по каждой вкладке.

Рисунок 2.2.8 «Сортировка по категориям»

Также можно производить фильтрацию товаров по количеству. Это выполняется под действием вкладки, которая находится под популярными категориями.

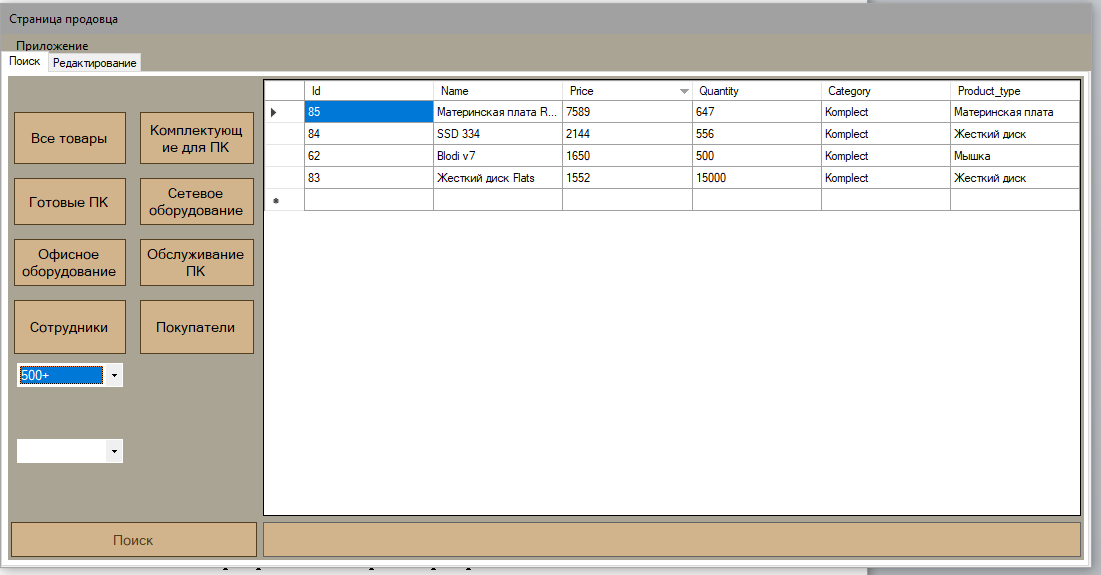


Рисунок 2.2.9 «Фильтрация по количеству»

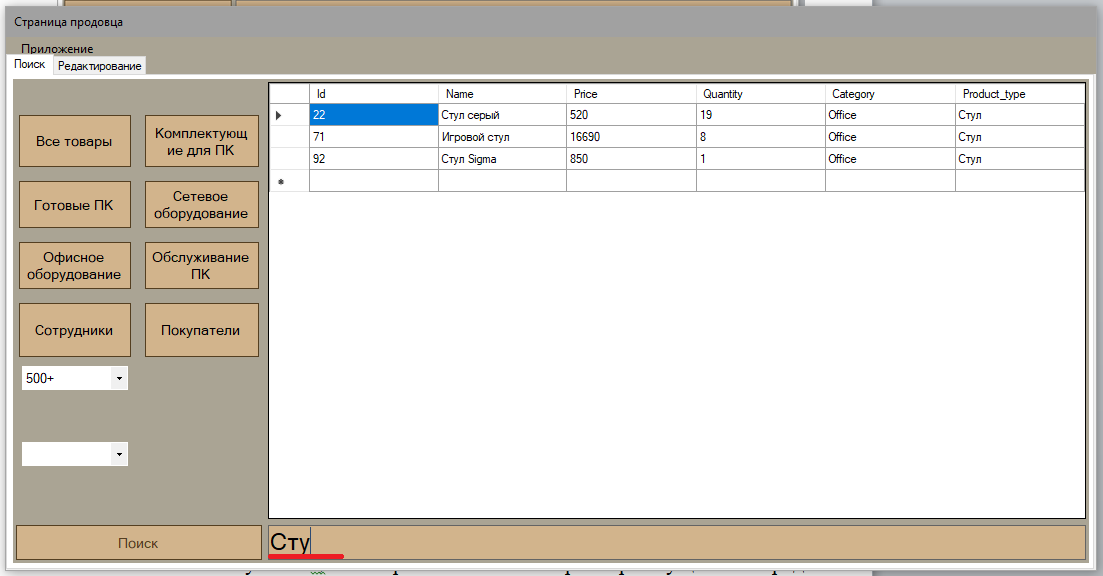
Также сортировка товаров может производиться по названию товара. Это происходит в динамическом режиме, как только пользователь начинает вводить буквы в строку.

Рисунок 2.2.10 «Сортировка по названию»

Ниже вкладки количество товаров находится вкладка, которая активирует и деактивирует кнопку поиск. Она необходима для программного ввода поиска товаров.

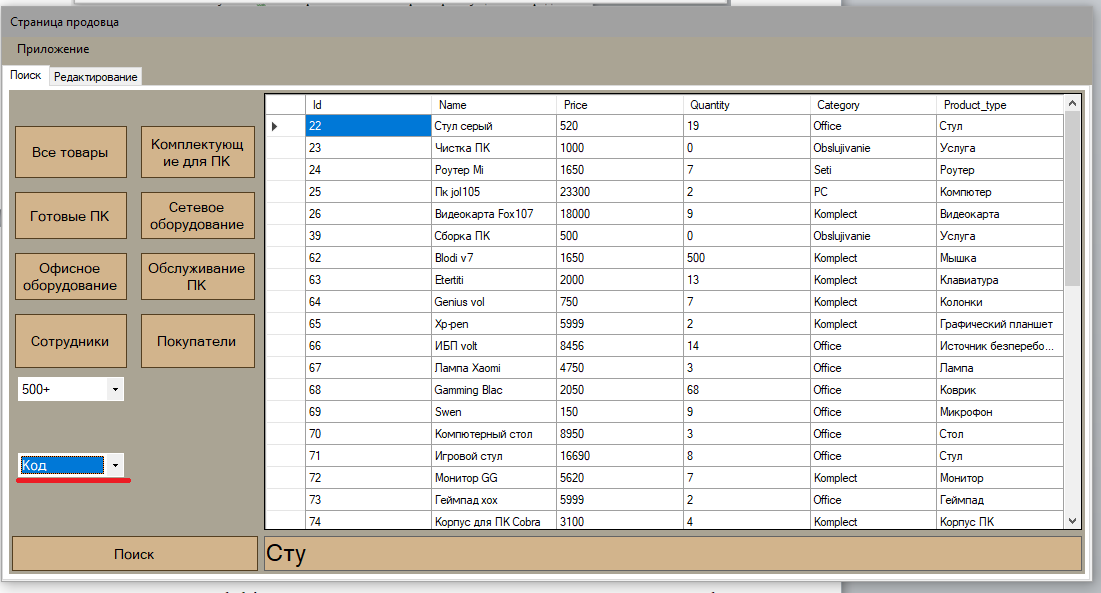


Рисунок 2.2.11 «Сортировка по коду»

Далее мы переходим на вкладку редактирование она имеет под панели:

* Товары
* Пользователи
* Сотрудники

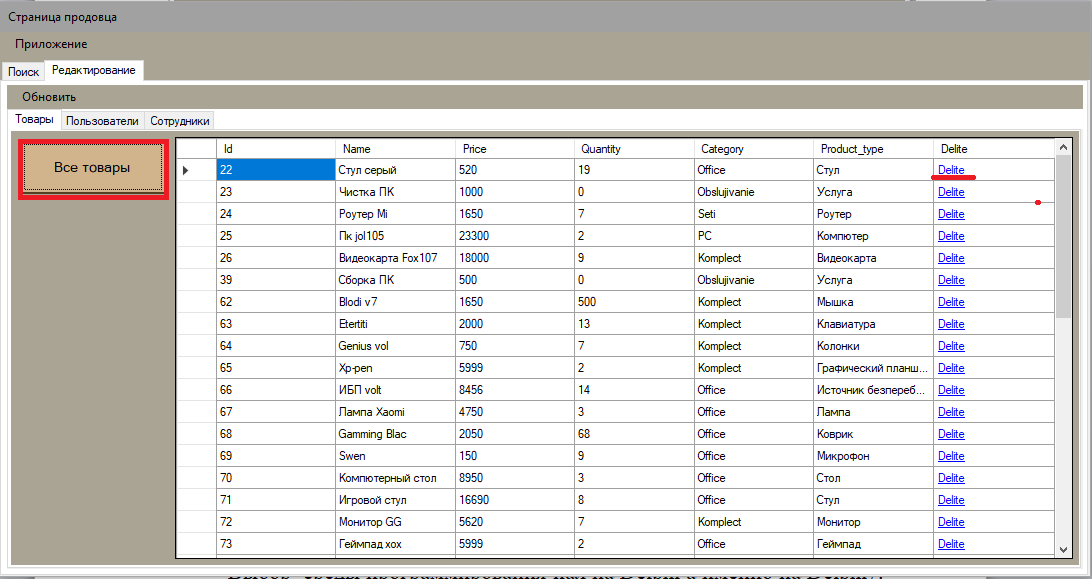
При нажатии, кнопки все товары. У нас отображается список всех товаров. В окне таблицы можно производить ввод редактирование и удаление дынных.

Рисунок 2.2.12 «Редактирование товаров»

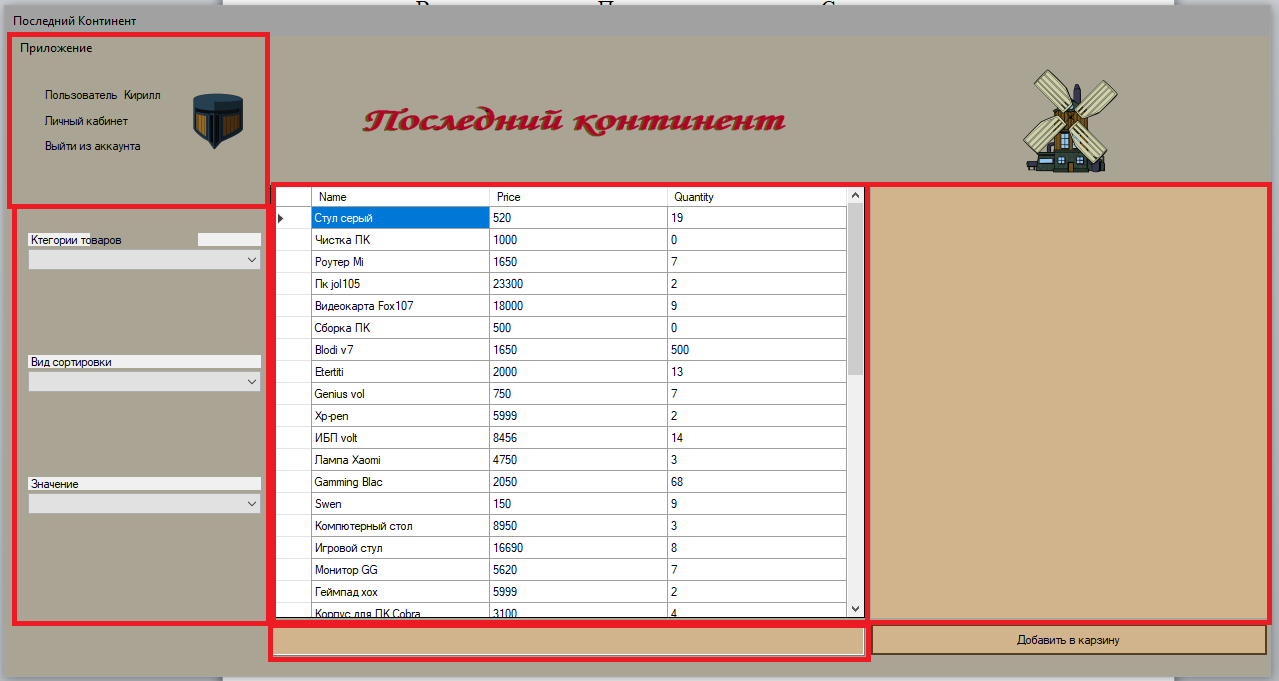
Во вкладках «Пользователи» и «Сотрудники» производится редактирование данных пользователей и сотрудников. Далее мы переходим обратно в главное меню и переходим во вкладку «Пользователь». Окно разбито на зоны, которые имеют свои функции данная часть приложения ещё не до конца реализована.

Рисунок 2.2.13 «Окно пользователя»

Панель с данными пользователя предполагает вход и выход из аккаунта и отображение данных пользователя и переход в его личный кабинет.

В окне можно производить фильтрации и сортировки. Также в последующих версиях будет реализована корзина товаров и отображение выбранного товара.

**Компоненты Visual Studio используемые в программе**

**menuStrip** – позволяет поместить главное меню в программу. При помещении menuStrip на форму это выглядит, как просто иконка Иконки данного типа называют невидимым (невизуальным) компонентом, поскольку они невидимы во время выполнения программы.



Рисунок 2.2.14 «menuStrip до запуска»

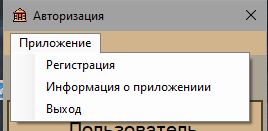


Рисунок 2.2.15 «menuStrip после запуска»

Создание меню включает три шага:

1. Помещение menuStrip на форму;
2. Вызов Menu Designer (Рис.2);
3. Определение пунктов меню в Menu Designer.

** Label –**служит для отображения текста на экране. Можно изменить шрифт и цвет метки, если дважды щелкнуть на свойство Font в Инспекторе Объектов. Видно, что это легко сделать и во время выполнения программы, написав всего одну строчку кода.

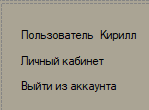


Рисунок 2.2.16 «Компонент Lable»

Компонент представляет собой статический текст. С помощью этого компо­нента на рабочей поверхности формы можно отобразить информацию, сделать пояснения и показать названия других компонентов. Но он имеет и другую важную функцию – если в составе текста Label есть символы-акселераторы, информация об их нажатии может передаваться от Label другому элементу управления.

Компонент PictureBox предназначен для вставки изображений в приложение. Изображение можно использовать в нескольких форматах увеличенное, по размеру объекта, по размеру изображения.

Рисунок 2.2.17 «Компонент PictureBox»

 Button Представляет элемент управления "Кнопка Windows", который будет реагировать на событие [Click](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.primitives.buttonbase.click?view=net-5.0). Чтобы применить одни и те же параметры свойств к нескольким [Button](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.button?view=net-5.0) элементам управления, используйте [Style](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.frameworkelement.style?view=net-5.0) свойство. Можно изменить значение по умолчанию, [ControlTemplate](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.controltemplate?view=net-5.0) чтобы обеспечить уникальность внешнего вида элемента управления.

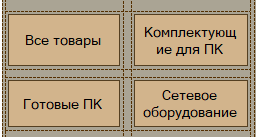


Рисунок 2.2.18 «Компонент Button»

Combo box — элемент графического интерфейса пользователя. Сочетание выпадающего списка и однострочного текстового поля, которое позволяет пользователю ввести значение вручную или выбрать из списка.

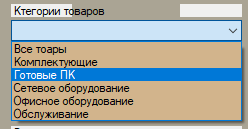


Рисунок 2.2.19 «Компонент Combo box»

**DataGridView** — очень мощный инструмент для работы с данными. Он очень гибок и позволяет автоматизировать вывод информации.

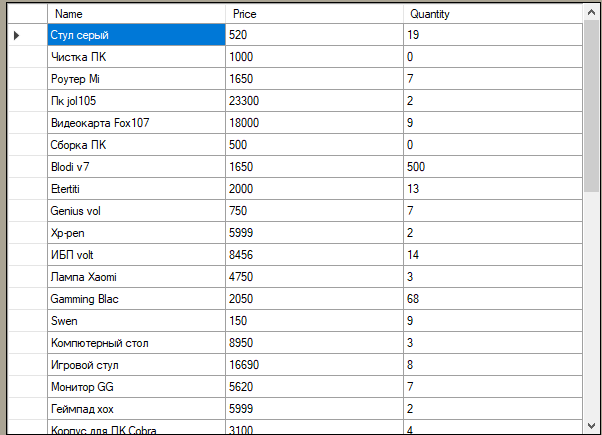


Рисунок 2.2.20 «Компонент Combo box»

**Листинг базы данных**

CREATE TABLE [dbo].[Products] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[Name] NVARCHAR (50) NULL,

[Price] INT NULL,

[Quantity] INT NULL,

[Category] NVARCHAR (50) NULL,

[Product\_type] NVARCHAR (50) NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);

**Листинг программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using System.Configuration;

namespace Store

{

public partial class Staff : Form

{

Form1 con1 = new Form1();

public int znach; // переменная значения с другой формы

private SqlCommandBuilder sqlBilder = null;

private SqlDataAdapter adapter = null;

private DataSet set1 = null;

private bool newRowAdding = false;

public Staff()

{

InitializeComponent();

}

private void LoadData() // загрузка занных бд

{

try

{

adapter = new SqlDataAdapter("SELECT \*, 'Delite' AS [Delite] FROM Products", con1.Connect\_SroteDB);

sqlBilder = new SqlCommandBuilder(adapter);

sqlBilder.GetDeleteCommand();

sqlBilder.GetUpdateCommand();

sqlBilder.GetInsertCommand();

set1 = new DataSet();

adapter.Fill(set1, "Products");

dataGridView2.DataSource = set1.Tables["Products"];

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView2[6, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadData() // обновление данных в бд

{

try

{

set1.Tables["Products"].Clear();

adapter.Fill(set1, "Products");

dataGridView2.DataSource = set1.Tables["Products"];

for (int i = 0; i < dataGridView2.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView2[6, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button9\_Click(object sender, EventArgs e) // открытие списка всех пользователей

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Users", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = false;

}

private void button2\_Click(object sender, EventArgs e) // Открытие списка всех товаров

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

}

private void Staff\_Load(object sender, EventArgs e) //запуск базы данных

{

con1.Connect\_SroteDB = new SqlConnection(ConfigurationManager.ConnectionStrings["Store\_db"].ConnectionString);

con1.Connect\_SroteDB.Open();

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

}

private void button8\_Click(object sender, EventArgs e) // открытие списка всех сотрудников

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Staff", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = false;

}

private void button1\_Click(object sender, EventArgs e) // поиск по командной строке

{

SqlDataAdapter adapter = new SqlDataAdapter(textBox1.Text, con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

}

private void button11\_Click(object sender, EventArgs e) // список всех товаров на вкладке редактирование товары

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView2.DataSource = set1.Tables[0];

LoadData();

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e) // закрытие формы

{

Application.Exit();

}

private void обновитьToolStripMenuItem\_Click(object sender, EventArgs e) // выбор функции для обновления на разных вкладках панели

{

if (tableLayoutPanel2.TabIndex == 0)

{

ReloadData();

}

else if (tableLayoutPanel2.TabIndex == 1)

{

ReloadUsers();

}

else if (tableLayoutPanel2.TabIndex == 2)

{

ReloadStaff();

}

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e) // бействия удаления обновления и редактирования

{

try

{

if (e.ColumnIndex == 6)

{

string task = dataGridView2.Rows[e.RowIndex].Cells[6].Value.ToString();

if (task == "Delite") // удаление данных из таблицы и бд

{

if (MessageBox.Show("Удалить эту строку?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dataGridView2.Rows.RemoveAt(rowIndex);

set1.Tables["Products"].Rows[rowIndex].Delete();

adapter.Update(set1, "Products");

}

}

else if (task == "Insert") // ввод даных в бд

{

int rowIndex = dataGridView2.Rows.Count - 2;

DataRow row = set1.Tables["Products"].NewRow();

row["Name"] = dataGridView2.Rows[rowIndex].Cells["Name"].Value;

row["Price"] = dataGridView2.Rows[rowIndex].Cells["Price"].Value;

row["Quantity"] = dataGridView2.Rows[rowIndex].Cells["Quantity"].Value;

kat ifrm = new kat();

ifrm.Owner = this;

ifrm.ShowDialog();

MessageBox.Show(Convert.ToString(znach)," staff");

if (znach == 0)

{

row["Category"] = "Komplect";

}

else if (znach == 1)

{

row["Category"] = "PC";

}

else if (znach == 2)

{

row["Category"] = "Seti";

}

else if (znach == 3)

{

row["Category"] = "Office";

}

else if (znach == 4)

{

row["Category"] = "Obslujivanie";

}

row["Product\_type"] = dataGridView2.Rows[rowIndex].Cells["Product\_type"].Value;

set1.Tables["Products"].Rows.Add(row);

set1.Tables["Products"].Rows.RemoveAt(set1.Tables["Products"].Rows.Count - 1);

dataGridView2.Rows.RemoveAt(dataGridView2.Rows.Count - 2);

dataGridView2.Rows[e.RowIndex].Cells[6].Value = "Delite";

adapter.Update(set1, "Products");

newRowAdding = false;

}

else if (task == "Update") // изменение данных в бд

{

int r = e.RowIndex;

set1.Tables["Products"].Rows[r]["Name"] = dataGridView2.Rows[r].Cells["Name"].Value;

set1.Tables["Products"].Rows[r]["Price"] = dataGridView2.Rows[r].Cells["Price"].Value;

set1.Tables["Products"].Rows[r]["Quantity"] = dataGridView2.Rows[r].Cells["Quantity"].Value;

set1.Tables["Products"].Rows[r]["Category"] = dataGridView2.Rows[r].Cells["Category"].Value;

set1.Tables["Products"].Rows[r]["Product\_type"] = dataGridView2.Rows[r].Cells["Product\_type"].Value;

adapter.Update(set1, "Products");

dataGridView2.Rows[e.RowIndex].Cells[6].Value = "Delite";

}

ReloadData();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView2\_UserAddedRow(object sender, DataGridViewRowEventArgs e) // добавление сторки в таблицу

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dataGridView2.Rows.Count - 2;

DataGridViewRow row = dataGridView2.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView2[6, lastRow] = linkCell;

row.Cells["Delite"].Value = "Insert";

}

}

catch (Exception)

{

}

}

private void dataGridView2\_CellValueChanged\_1(object sender, DataGridViewCellEventArgs e) // проверка на ввод

{

try

{

if (newRowAdding == false)

{

int rowIndex = dataGridView2.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dataGridView2.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView2[6, rowIndex] = linkCell;

editingRow.Cells["Delite"].Value = "Update";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView2\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e) // ввод только цифр в цифровых полях

{

e.Control.KeyPress -= new KeyPressEventHandler(columCeyPress);

if (dataGridView2.CurrentCell.ColumnIndex == 2 || dataGridView2.CurrentCell.ColumnIndex == 3)

{

TextBox textBox = e.Control as TextBox;

if (textBox != null)

{

textBox.KeyPress += new KeyPressEventHandler(columCeyPress);

}

}

}

private void columCeyPress(object Sender, KeyPressEventArgs e) // проверка строки новая или нет

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

// ------------------------------------------------------------------------Вкладка пользователи----------------------------------------------------------------

private void LoadUsers() // загрузка бызы всех поьзователей

{

try

{

adapter = new SqlDataAdapter("SELECT \*, 'Delite' AS [Delite] FROM Users", con1.Connect\_SroteDB);

sqlBilder = new SqlCommandBuilder(adapter);

sqlBilder.GetDeleteCommand();

sqlBilder.GetUpdateCommand();

sqlBilder.GetInsertCommand();

set1 = new DataSet();

adapter.Fill(set1, "Users");

dataGridView3.DataSource = set1.Tables["Users"];

for (int i = 0; i < dataGridView3.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView3[7, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadUsers()

{

try

{

set1.Tables["Users"].Clear();

adapter.Fill(set1, "Users");

dataGridView3.DataSource = set1.Tables["Users"];

for (int i = 0; i < dataGridView3.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView3[7, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void button10\_Click(object sender, EventArgs e) /// открытие списка всех пользователей

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Users", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView3.DataSource = set1.Tables[0];

LoadUsers();

}

private void dataGridView3\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == 7)

{

string task = dataGridView3.Rows[e.RowIndex].Cells[7].Value.ToString();

if (task == "Delite")

{

if (MessageBox.Show("Удалить эту строку?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dataGridView3.Rows.RemoveAt(rowIndex);

set1.Tables["Users"].Rows[rowIndex].Delete();

adapter.Update(set1, "Users");

}

}

else if (task == "Insert")

{

int rowIndex = dataGridView3.Rows.Count - 2;

DataRow row = set1.Tables["Users"].NewRow();

row["Name"] = dataGridView3.Rows[rowIndex].Cells["Name"].Value;

row["Surname"] = dataGridView3.Rows[rowIndex].Cells["Surname"].Value;

row["Patronymic"] = dataGridView3.Rows[rowIndex].Cells["Patronymic"].Value;

row["Date\_of\_Birth"] = dataGridView3.Rows[rowIndex].Cells["Date\_of\_Birth"].Value;

row["Address"] = dataGridView3.Rows[rowIndex].Cells["Address"].Value;

row["Phone"] = dataGridView3.Rows[rowIndex].Cells["Phone"].Value;

set1.Tables["Users"].Rows.Add(row);

set1.Tables["Users"].Rows.RemoveAt(set1.Tables["Users"].Rows.Count - 1);

dataGridView3.Rows.RemoveAt(dataGridView3.Rows.Count - 2);

dataGridView3.Rows[e.RowIndex].Cells[7].Value = "Delite";

adapter.Update(set1, "Users");

newRowAdding = false;

}

else if (task == "Update")

{

int r = e.RowIndex;

set1.Tables["Users"].Rows[r]["Name"] = dataGridView3.Rows[r].Cells["Name"].Value;

set1.Tables["Users"].Rows[r]["Surname"] = dataGridView3.Rows[r].Cells["Surname"].Value;

set1.Tables["Users"].Rows[r]["Patronymic"] = dataGridView3.Rows[r].Cells["Patronymic"].Value;

set1.Tables["Users"].Rows[r]["Date\_of\_Birth"] = dataGridView3.Rows[r].Cells["Date\_of\_Birth"].Value;

set1.Tables["Users"].Rows[r]["Address"] = dataGridView3.Rows[r].Cells["Address"].Value;

set1.Tables["Users"].Rows[r]["Phone"] = dataGridView3.Rows[r].Cells["Phone"].Value;

adapter.Update(set1, "Users");

dataGridView3.Rows[e.RowIndex].Cells[7].Value = "Delite";

}

ReloadUsers();

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView3\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dataGridView3.Rows.Count - 2;

DataGridViewRow row = dataGridView3.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView3[7, lastRow] = linkCell;

row.Cells["Delite"].Value = "Insert";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void dataGridView3\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dataGridView3.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dataGridView3.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView3[7, rowIndex] = linkCell;

editingRow.Cells["Delite"].Value = "Update";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//----------------------------------------------------------------------------------Сотрудники---------------------------------------------------------------------------------------

private void LoadStaff() // загрузка бызы всех сотрудников

{

try

{

adapter = new SqlDataAdapter("SELECT \*, 'Delite' AS [Delite] FROM Staff", con1.Connect\_SroteDB);

sqlBilder = new SqlCommandBuilder(adapter);

sqlBilder.GetDeleteCommand();

sqlBilder.GetUpdateCommand();

sqlBilder.GetInsertCommand();

set1 = new DataSet();

adapter.Fill(set1, "Staff");

dataGridView4.DataSource = set1.Tables["Staff"];

for (int i = 0; i < dataGridView4.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView4[5, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ReloadStaff()

{

try

{

set1.Tables["Staff"].Clear();

adapter.Fill(set1, "Staff");

dataGridView4.DataSource = set1.Tables["Staff"];

for (int i = 0; i < dataGridView4.Rows.Count; i++)

{

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView4[5, i] = linkCell;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

private void button12\_Click(object sender, EventArgs e)

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Staff", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView4.DataSource = set1.Tables[0];

LoadStaff();

}

private void dataGridView4\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

try

{

if (e.ColumnIndex == 5)

{

string task = dataGridView4.Rows[e.RowIndex].Cells[5].Value.ToString();

if (task == "Delite")

{

if (MessageBox.Show("Удалить эту строку?", "Удаление", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

int rowIndex = e.RowIndex;

dataGridView4.Rows.RemoveAt(rowIndex);

set1.Tables["Staff"].Rows[rowIndex].Delete();

adapter.Update(set1, "Staff");

}

}

else if (task == "Insert")

{

int rowIndex = dataGridView4.Rows.Count - 2;

DataRow row = set1.Tables["Staff"].NewRow();

row["Name"] = dataGridView4.Rows[rowIndex].Cells["Name"].Value;

row["Surname"] = dataGridView4.Rows[rowIndex].Cells["Surname"].Value;

row["Position"] = dataGridView4.Rows[rowIndex].Cells["Position"].Value;

row["Phone"] = dataGridView4.Rows[rowIndex].Cells["Phone"].Value;

set1.Tables["Staff"].Rows.Add(row);

set1.Tables["Staff"].Rows.RemoveAt(set1.Tables["Staff"].Rows.Count - 1);

dataGridView4.Rows.RemoveAt(dataGridView4.Rows.Count - 2);

dataGridView4.Rows[e.RowIndex].Cells[5].Value = "Delite";

adapter.Update(set1, "Staff");

newRowAdding = false;

}

else if (task == "Update")

{

int r = e.RowIndex;

set1.Tables["Staff"].Rows[r]["Name"] = dataGridView4.Rows[r].Cells["Name"].Value;

set1.Tables["Staff"].Rows[r]["Surname"] = dataGridView4.Rows[r].Cells["Surname"].Value;

set1.Tables["Staff"].Rows[r]["Position"] = dataGridView4.Rows[r].Cells["Position"].Value;

set1.Tables["Staff"].Rows[r]["Phone"] = dataGridView4.Rows[r].Cells["Phone"].Value;

adapter.Update(set1, "Staff");

dataGridView4.Rows[e.RowIndex].Cells[5].Value = "Delite";

}

ReloadStaff();

}

}

catch (Exception ex)

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

private void dataGridView4\_UserAddedRow(object sender, DataGridViewRowEventArgs e)

{

try

{

if (newRowAdding == false)

{

newRowAdding = true;

int lastRow = dataGridView4.Rows.Count - 2;

DataGridViewRow row = dataGridView4.Rows[lastRow];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView4[5, lastRow] = linkCell;

row.Cells["Delite"].Value = "Insert";

}

}

catch (Exception)

{

}

}

private void dataGridView4\_CellValueChanged(object sender, DataGridViewCellEventArgs e)

{

try

{

if (newRowAdding == false)

{

int rowIndex = dataGridView4.SelectedCells[0].RowIndex;

DataGridViewRow editingRow = dataGridView4.Rows[rowIndex];

DataGridViewLinkCell linkCell = new DataGridViewLinkCell();

dataGridView4[5, rowIndex] = linkCell;

editingRow.Cells["Delite"].Value = "Update";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

//-------------------------------------------------------------------------------------Фильтры по категориям------------------------------------------------------

private void button3\_Click(object sender, EventArgs e)//- комплектующие для ПК

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products where Category = 'Komplect'", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

}

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

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products where Category = 'PC'", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

}

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

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products where Category = 'Seti'", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

}

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

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products where Category = 'Office'", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

}

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

{

SqlDataAdapter adapter = new SqlDataAdapter("Select \* FROM Products where Category = 'Obslujivanie'", con1.Connect\_SroteDB);

DataSet set1 = new DataSet();

adapter.Fill(set1);

dataGridView1.DataSource = set1.Tables[0];

comboBox1.Enabled = true;

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

{

MessageBox.Show("Beta version 0.1");

}

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

{

MessageBox.Show("" +

"Приложение Магазин компютерной техники Последний континент"+

" Разработал Зинченко К.А");

private void textBox1\_TextChanged(object sender, EventArgs e)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = $"Name LIKE '%{textBox1.Text}%'";

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedIndex == 0)

{

button1.Enabled = true;

}

else if (comboBox2.SelectedIndex == 1)

{

button1.Enabled = false;

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = $"Quantity <= 10";

}

else if (comboBox1.SelectedIndex == 1)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = $"Quantity >= 11 AND Quantity <= 499

}

else if (comboBox1.SelectedIndex == 2)

{

(dataGridView1.DataSource as DataTable).DefaultView.RowFilter = $"Quantity >= 500";

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

{

Form ifrm = new Form1();

ifrm.Show(); // отображаем Form2

this.Hide(); // скрываем Form1 (this - текущая форма)

### Заключение

При создании базы данных и приложения было поставлено и реализовано несколько задач:

* Создание базы данных
* Создание таблиц
* Заполнение таблиц для работы данными
* Написание приложения
* Создание визуальной части
* Написание кода приложения
* Подключение базы данных к приложению

На многих этапах были ошибки и множество проблем с которыми помогли справится методические материалы советы преподавателей и однокурсников. В заключении хотелось бы сказать написание базы данных и приложения «Магазин компьютерной техники» очень трудоемкий, но и в тоже время интересный процесс. Конечно, не все будет гладко и так как задумывалось всегда будет, что то хорошее, но и без отрицательного результата нельзя ведь это тоже результат.

Но все же это очень кропотливый процесс, который требует терпения и усидчивости. Но в итоге это приводит к результату.

**Достигнута ли цель?**

Полностью понять это можно будет только со временем, но на данный момент я считаю, что выполнил поставленные перед собой задачи и продолжаю работать над приложением «Магазин компьютерной техники». Использовать приложение как материал для тестирования все новых функций и задач, поставленных преподавателями и лично мной.

### Список литературы

1. Алексеев, А.А. Основы параллельного программирования с использованием Visual Studio 2010 : учебное пособие / Алексеев А.А.. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 330 c. — ISBN 978-5-4497-0341-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/89456.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
2. Разработка приложений на C# в среде Visual Studio : учебное пособие / А.М. Нужный, [и др.].. — Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2019. — 89 c. — ISBN 978-5-7731-0776-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/93286.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
3. Сафонов, В.О. Возможности Visual Studio 2013 и их использование для облачных вычислений : учебное пособие / Сафонов В.О.. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 371 c. — ISBN 978-5-4497-0870-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/102010.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
4. Тассов, К.Л. Консольные приложения в среде Microsoft Visual C++ 7.0. NET 2005 : методические указания к выполнению лабораторных работ по курсу «Программирование на С/С++» / Тассов К.Л., Шикуть А.В.. — Москва : Московский государственный технический университет имени Н.Э. Баумана, 2010. — 36 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/31025.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
5. Разработка приложений на C# в среде Visual Studio : учебное пособие / А.М. Нужный, [и др.].. — Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2019. — 89 c. — ISBN 978-5-7731-0776-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/93286.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
6. Столбовский, Д.Н. Разработка Web-приложений ASP.NET с использованием Visual Studio .NET : учебное пособие / Столбовский Д.Н.. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2020. — 375 c. — ISBN 978-5-4497-0370-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/89469.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
7. Сафонов, В.О. Возможности Visual Studio 2013 и их использование для облачных вычислений : учебное пособие / Сафонов В.О.. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 371 c. — ISBN 978-5-4497-0870-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/102010.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
8. Катунин, Г.П. Обработка цифровых изображений в программе ACDSee Photo Studio Ultimate 2019 : учебное пособие / Катунин Г.П.. — Саратов : Ай Пи Эр Медиа, 2019. — 551 c. — ISBN 978-5-4486-0715-8. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/80922.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
9. Смоленцев, Н.К. MATLAB: программирование на Visual C#, Borland C#, JBuilder, VBA : учебный курс / Смоленцев Н.К.. — Саратов : Профобразование, 2019. — 456 c. — ISBN 978-5-4488-0066-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/89868.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
10. Кодификация российского частного права 2019 / В.В. Витрянский, [и др.].. — Москва : Статут, 2019. — 492 c. — ISBN 978-5-8354-1547-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/94609.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
11. Алексеев, Д.В. Компьютерное моделирование физических задач в Microsoft Visual Basic / Алексеев Д.В.. — Москва : СОЛОН-ПРЕСС, 2009. — 518 c. — ISBN 5-98003-092-1. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/8649.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
12. Нахман, А.Д. Моделирование процессов средствами дифференциальных уравнений : учебное пособие / Нахман А.Д., Родионов Ю.В.. — Тамбов : Тамбовский государственный технический университет, ЭБС АСВ, 2019. — 112 c. — ISBN 978-5-8265-2019-2. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/99773.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
13. Кариев, Ч.А. Разработка Windows-приложений на основе Visual C# : учебное пособие / Кариев Ч.А.. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 978 c. — ISBN 978-5-4497-0909-7. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/102057.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
14. Смоленцев, Н.К. MATLAB: программирование на Visual C#, Borland C#, JBuilder, VBA : учебный курс / Смоленцев Н.К.. — Саратов : Профобразование, 2019. — 456 c. — ISBN 978-5-4488-0066-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/89868.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. Пользователей
15. Ефремова, М.В. Моделирование СВЧ приборов с помощью программы CST Particle Studio / Ефремова М.В., Иванов И.М., Курушин А.А.. — Москва : СОЛОН-Пресс, 2019. — 332 c. — ISBN 978-5-91359-331-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/94942.html (дата обращения: 28.05.2021). — Режим доступа: для авторизир. пользователей