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

[Реферат 3](#_Toc8003242)

[Обозначения и сокращения 4](#_Toc8003243)

[Введение 4](#_Toc8003244)

[Основная часть 4](#_Toc8003245)

[Заключение 4](#_Toc8003246)

[Список использованных источников 4](#_Toc8003247)

[Приложение 4](#_Toc8003248)

# Реферат

реферат

# Обозначения и сокращения

# Введение

Цели практики:

* Углубление и закрепление теоретических знаний полученных при изучении институтских дисциплин.
* Развитие и накопление практических умений и навыков по анализу, проектированию, написанию программного кода.

Задачи практики:

* Разработка требований к программной системе.
* Проектирование архитектуры программной системы.
* Разработка программной системы в соответствии с заданием.
* Описание по применению программной системы.
* Получение навыков создания интерфейсных приложений используя Windows Forms.
* Получение навыков работы с табличными данными.

При выполнении практики были выполнены следующие задания:

* Авторизация пользователей в системе.
* Выполнение запросов к базе данных и обработка полученных результатов.
* Фильтрация и сортировка данных.

# 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, 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.

# Проектирование системы

Автоматизированная информационная система (далее - АИС) «Магазин цифровой электронной техники» обеспечивает удобный способ хранения, редактирования, обновления информации о товарах магазина.

## 2.1. Функциональные возможности программы

В АИС предусмотрены следующие функции, призванных упростить работу с товарами магазина:

* Вывод списка товаров, фильтрация и сортировка.
* Добавление товаров в корзину, с последующим оформлением заказа.
* Печати таблицы или её части, 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 пользователя 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. База данных

В решении используется локальная база данных MS-SQL. Она содержит 4 таблицы:

* Orders – таблица заказов (Рисунок 1)
* Products – таблица товаров (Рисунок 2)
* Reviews – таблица с отзывами (Рисунок 3)
* Users – таблица с пользователями (Рисунок 4)

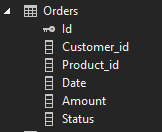
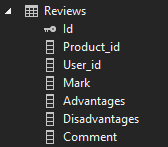


Рисунок 2. Таблица с отзывами

Рисунок 1. Таблица заказов

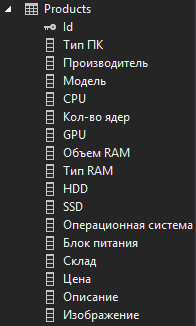
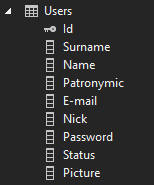


Рисунок 4. Таблица с пользователями

Рисунок 3. Таблица с товарами

Строка подключения к базе данных выглядит следующим образом:

«Data Source=(LocalDB)\MSSQLLocalDB; AttachDbFilename=<Путь к файлу AIS\_shop.exe>\DataBaseDET.mdf; Integrated Security=True»

В каждой таблице есть первичный ключ – id записи. По этим ключам осуществляется связь между таблицами.

Также в АИС формируются запросы получения данных из нескольких таблиц – при загрузке заказов пользователя в профиль.

# 3. Описание программной реализации

АИС реализована на языке программирования C#, с использованием Windows Forms.

В программе существует статический класс Common, в котором содержатся строка подключения к базе данных, получаемая из конфигурационного “.config”-файла и список из id товаров, находящихся в корзине.

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

В АИС при проверке корректности введенных полей используются регулярные выражения. Они выглядят следующим образом:

* "^\d+$" – для проверки целых чисел.
* "^([a-z0-9]|\_){4,}$" – для проверки ника и пароля. Допустимы латинские буквы, цифры и символ «\_». Кол-во введенных символов – не менее четырех.

Для проверки корректности e-mail используется функция IsValidEmail(), которая также использует регулярное выражение:

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

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.

Реализовано 2 класса для работы с файлами/изображениями и БД:

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

# 4. Описание интерфейса

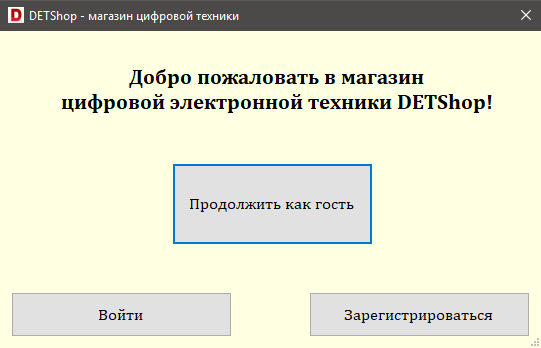
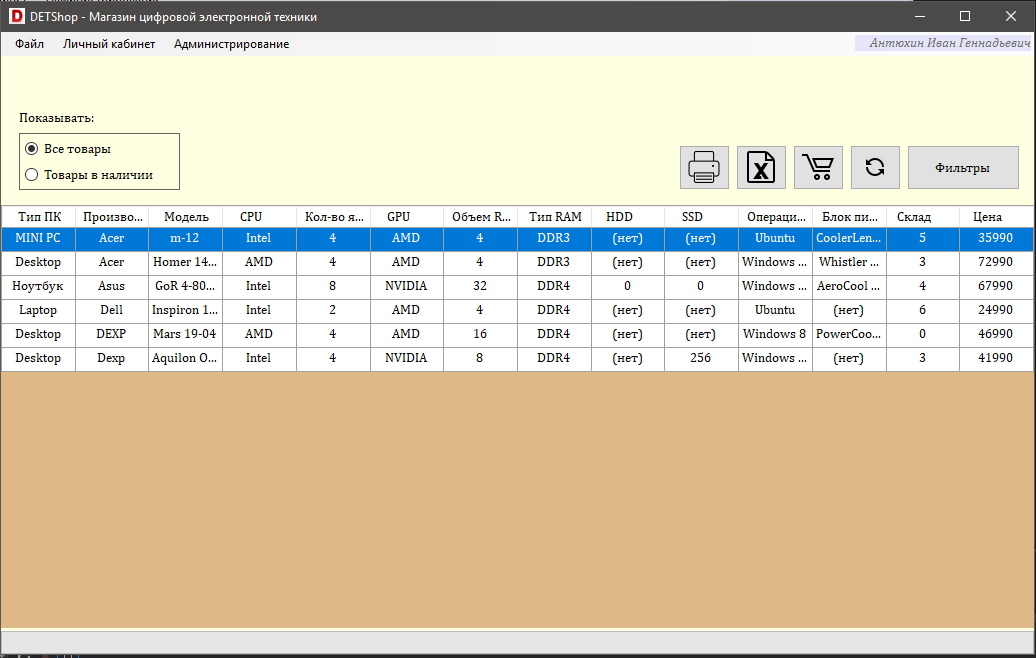
АИС имеет интерфейс пользователя. При запуске программы запускается главное окно (Рис. 6), и вместе с ним приветственное (Рис. 5), на котором пользователю предлагается войти/зарегистрироваться (Рис. 11-12).

Рисунок 6. Главная форма

Рисунок 5. Приветственная форма

На главной форме (рис. 6), справа над таблицей товаров присутствуют кнопки «Печать», «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» - видна (доступна) только когда в системе администратор. В ней 3 пункта:

* «Управление товарами»;
* «Управление заказами»;
* «Зарегистрировать нового пользователя» - для регистрации нового администратора (Рис. 12);

При запуске формы с фильтрами изначально все фильтры отмечены (рис. 7), 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.

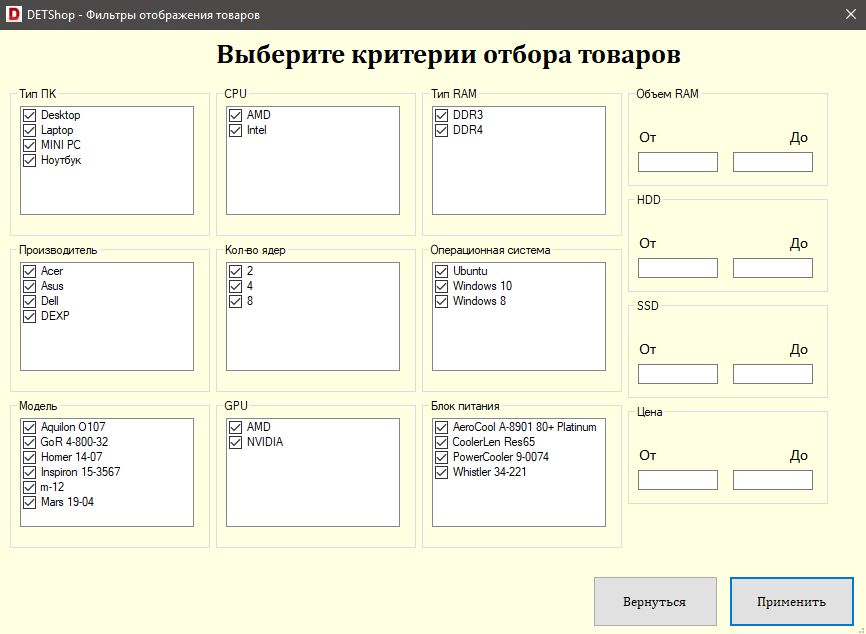
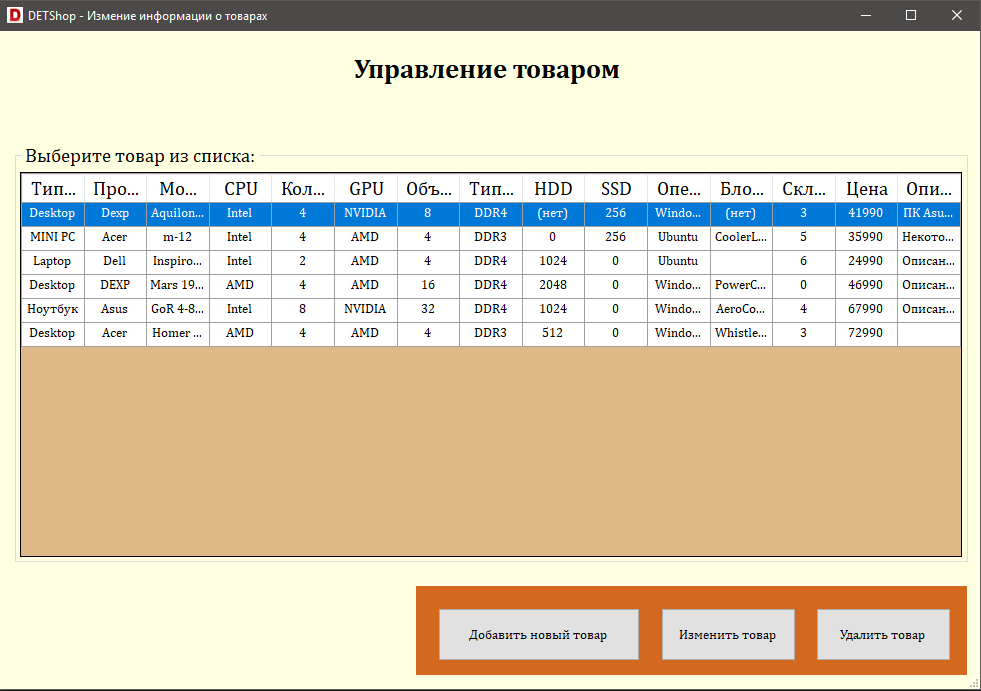


Рисунок 7. Форма с фильтрами

Изменять список товаров может только администратор на форме ProductManagement (Рис. 8). Чтобы удалить товар, нужно выделить его и нажать на кнопку удалить. Добавление и изменение товара – эти формы внешне мало отличаются друг от друга (Рис. 14).

Администратор меняет статус заказа через соответствующую форму (Рис . 9). Чтобы изменить статус заказа необходимо кликнуть на ячейку и ввести новое значение. Статус заказа – число от 0 до 2, где 0 – «заказ оформлен», 1 – «заказ обрабатывается», 2 - «заказ завершен». Также на форме есть фильтр позволяющий скрыть все завершенные заказы – ComboBox. Также администратор может удалять заказы – по выделению строки нажать клавишу на клавиатуре “Delete”.



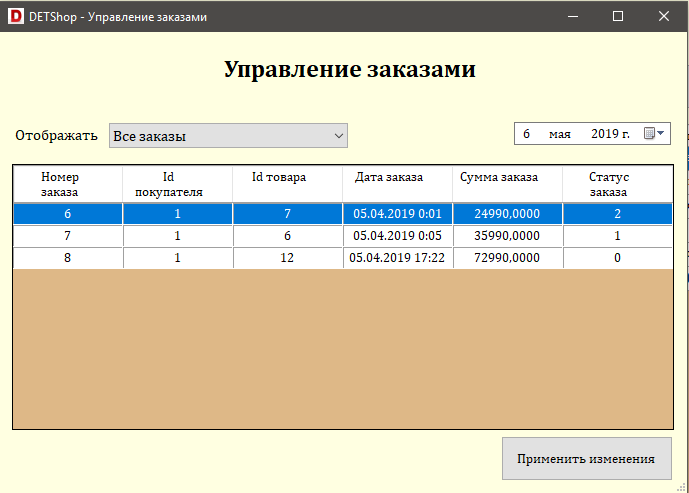
Рисунок 8. Управление товаром

Рисунок 9. Форма управления заказами

При выборе из таблицы главной формы двойным щелчком товар открывается форма с подробной информацией о товаре (Рис. 10). Любой пользователь 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.

Рисунок 10. Форма с подробной информацией о товаре

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. 13). С помощью NumeicUpDown происходит установка оценки. Для созранения отзыва следует нажать «Применить изменения».

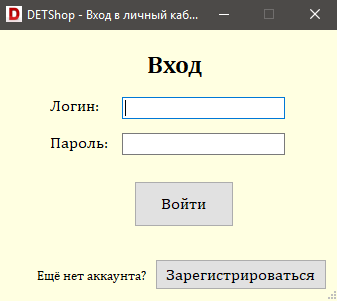


Рисунок 11. Форма входа

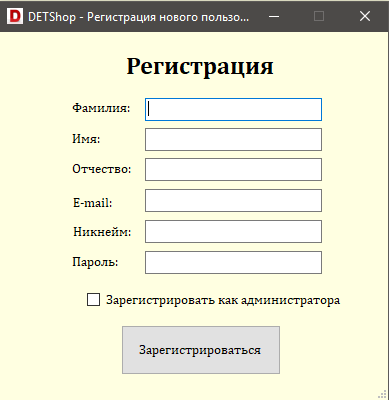


Рисунок 12. Форма регистрации

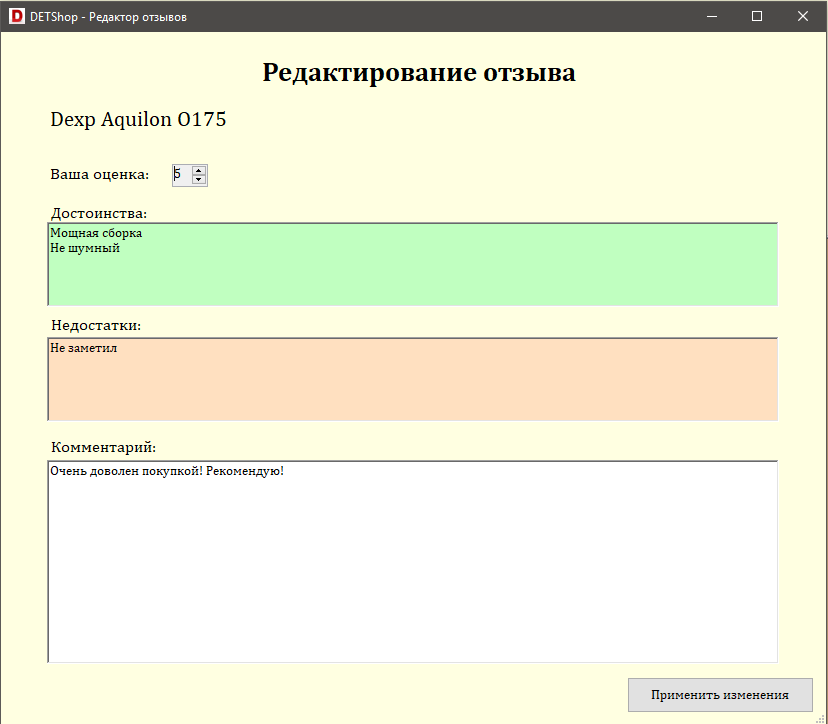
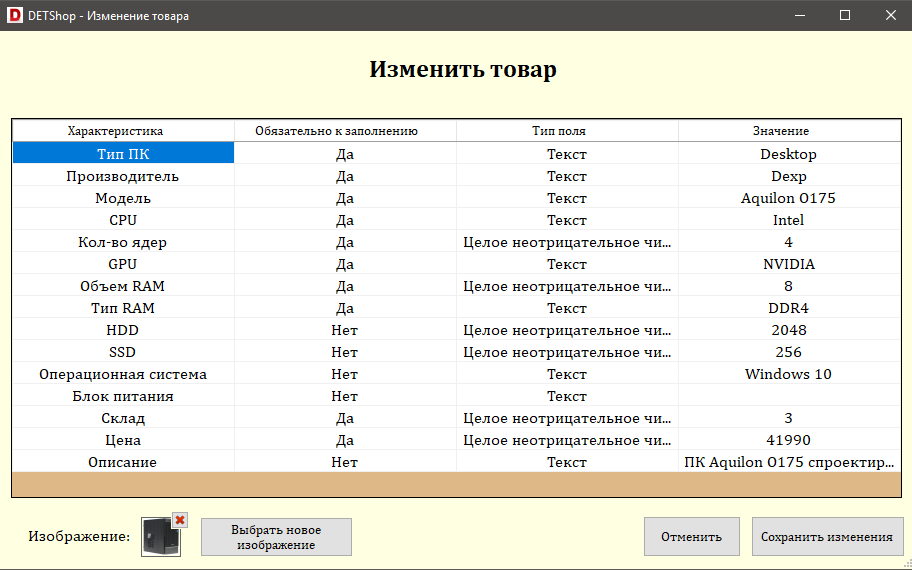


Рисунок 14. Форма изменения товара

Рисунок 13. Форма добавления/редактирования отзывов

# 5. Теоретическая подготовка. Изучение SQL и основ баз данных

Было решено использовать в качестве таблиц с данными таблицы СУБД, поэтому в рамках подготовки к практике был пройден курс SQL Fundamentals Course от портала SoloLearn.

# 6. Разработка требований к спецификации данных, связям таблиц, связям между данными

После некоторых исследований, для хранения данных текстового формата было решено использовать типы данных – nvarchar(50), ntext;

для целых чисел – int;

для даты – DateTime;

для изображений – Image.

Решено было осуществлять связь между данными таблиц по первичным ключам.

# 7. Разработка перечня форм, названий и связей между ними для каждого вида пользователей, последовательность вызова форм.

Был создан перечень необходимых форм и проработано взаимодействие между ними. Придуманы названия форм.

Создание форм с минимальной функциональностью – переходами между формами. Обработка и реализация требований перехода между формами.

# 8. Приведение элементов графического интерфейса к единому стилю

Для всех форм задана одна иконка, один шрифт – Cambria, 10 pt.

Установлена одна цветовая схема:

* Цвет фона 1 - Info;
* Цвет фона 2 - BurlyWood;

Ограничение размера некоторых форм и реализация корректной масштабируемости форм.

# 9. Реализация главной формы

Программирование главной формы для отображения товаров, возможности перехода в личный кабинет, управления товарами, фильтрацию товаров. Добавление базы данных в проект. Обеспечение загрузки данных из неё. Программирование приветственной формы.

При загрузке формы запускается приветственное окно как модальное. Затем загружаются данные из БД, с помощью метода UpdateDataGridView. Эта функция учитывает отметку в radioButtons и формирует запрос в зависимости от них.

Пока данные не загружены, кнопка «Фильтры» неактивна. После загрузки можно фильтровать товары.

После каждого вызова формы регистрации/авторизации вызывается метод UserStateChange. Он является «переключателем» эл-тов управления - показывает/прячет эл-ты в зависимости от пользователя в системе.

# 10. Реализация формы с фильтрами отображения товаров

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

* Фильтр с возможными вариантами выбора (FilterChecked);
* Фильтр по диапазону (FilterFromTo);

Фильтры добавляются динамически. Для этого в программе реализованы классы FilterChecked и FilterFromTo. Каждый фильтр находятся в своем GroupBox.

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.

# 11. Реализация формы редактирования товара

Ввод данных через таблицу DataGridView в 4 ячейку (Рис. 14).

Проверка данных выполняется функцией valid. Если все поля корректно заполнены формируется запрос к БД.

Изображения загружаются в БД в виде потока байтов (byte[]).

# 12. Реализация формы с подробным отображением информации о товаре

Для того чтобы вызвать эту форму, необходимо в её конструктор передать строку DataGridViewRow, из которой она берет id товара.

При загрузке формы происходят загрузка названия, цены, наличия на складе и т. д. Затем асинхронно загружается описание, затем – характеристики товара, затем – отзывы.

Если товара нет на складе – деактивировать кнопку «Добавить в корзину».

По нажатию кнопки добавить отзыв – вызвать модальным окном Редактор отзывов и после его завершения обновить отзывы.

# 13. Реализация формы управления заказами

Загрузка всех заказов из БД, в зависимости от выбранного в ComboBox режима.

Все столбцы, кроме «Статуса заказа», устанавливаются только для чтения.

Для обновления измененного статуса используется SqlCommandBuilder и SqlDataAdapter.

# 14. Реализация «корзины»

При загрузке формы загрузка всех товаров, id которых есть в

Common.ProductsInCart.

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

По нажатию кнопки «Удалить из корзины» удалить элемент из DataGridView и из Common.ProductsInCart

# 15. Реализация личного кабинета

Загрузка информации о пользователе в форму. Загрузка изображения в pictureBox.

Загрузка заказов пользователя. Запрос:

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})

Смена изображения по нажатию на linkLabel.

При закрытии формы спрашивать, сохранять ли изменения.

# 16. Реализация возможности добавления и редактирования отзывов

Во время загрузки формы загрузить название товара, а затем загрузить оценку, достоинства, недостатки и комментарий к данному товару, от данного пользователя. Если данные не получены – использовать команду вставки (INSERT INTO) новой записи. Иначе использовать команду обновления (UPDATE).

# 17. Реализация экспорта информации о товарах в таблицу Excel

Используется библиотека Microsoft.Office.Interop.Excel.dll, Используется SaveFileDialog.

Если было выбрано расширение ‘.csv’ – конвертация всех ячеек в одну строку, затем экспорт таблицы в виде csv-файла. Иначе вывод данных в ячейки Excel.

# 18. Реализация возможности печати информации о товаре

По нажатию кнопки «Печать» создание нового документа для печати, подписка на событие printDocument\_PrintPage. Создание printDialog, установка параметров печати. Вызов previewPrintDialog.

printDocument\_PrintPage() – рисует документ для печати.

# Заключение

В результате прохождения учебной практики было разработано приложение, которое способно выполнять следующие функции:

* Выводить список товаров;
* Осуществлять фильтрацию и сортировку списка товаров;
* Вход и регистрация 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, добавлять новые, удалять;

Были освоены средства разработки языка C#, платформы .NET, языка запросов SQL и базы данных MS-SQL. Был 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 25.03.2019)

1. Язык программирования C# [ru.wikipedia.org] // URL: <https://ru.wikipedia.org/wiki/C_Sharp> (дата обращения 18.03.2019)
2. Руководство по программированию в Windows Forms [metanit.com] // URL: https://metanit.com/sharp/windowsforms/ (дата обращения 21.04.2017)
3. 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 27.03.2019)

# Приложение а

Файл AboutProgram.cs

namespace AIS\_shop

{

partial class AboutProgram : Form

{

public AboutProgram()

{

InitializeComponent();

this.Text = "DETShop - О 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)

{

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.CommandText = query.CommandText.Replace("@image", "'NULL'");

// 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.IsNullOrWhiteSpace(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 распозновании 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)

{

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;

labelFileName.Text = Path.GetFileName(openImage.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 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;

public Authorization()

{

InitializeComponent();

}

private void Authorization\_Load(object sender, EventArgs e)

{

Cursor = Cursors.WaitCursor;

if (!loadUsersData())

{

MessageBox.Show("Данные о пользователях не были загружены из базы данных. Попробуйте перезапустить форму.", "Сообщение",

MessageBoxButtons.OK, MessageBoxIcon.Error);

Cursor = Cursors.Default;

Close();

}

Cursor = Cursors.Default;

}

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.IsNullOrWhiteSpace(nick) && !string.IsNullOrWhiteSpace(password))

{

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.Normal;

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

User user = 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("Введите данные!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

// загрузка всех данных для входа, т.к. sql - регистронезависимый

private bool loadUsersData()

{

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);

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();

}

}

}

}

Файл Cart.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace AIS\_shop

{

public partial class Cart : Form

{

public Cart()

{

InitializeComponent();

}

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) return;

if (MessageBox.Show("Вы уверены что хотите оформить этот заказ?", "Подтверждение действия",

MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes) 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 (dgv.SelectedRows.Count != 1)

{

MessageBox.Show("Выберите товар", "Некорректное действие",

MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

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 buttonCancel\_Click(object sender, EventArgs e)

{

Close();

}

private void Cart\_Load(object sender, EventArgs e)

{

if (Common.ProductsInCart.Count != 0) loadData();

else Close();

}

private async void loadData()

{

SqlConnection connection = new SqlConnection(Common.StrSQLConnection);

SqlCommand query = new SqlCommand(@"SELECT Id, CONCAT(Производитель, ' ', Модель) AS Название, Цена FROM Products WHERE", connection);

SqlDataAdapter adapter = new SqlDataAdapter(query);

DataSet ds = new DataSet();

try

{

await connection.OpenAsync();

int count = 0;

foreach (var it in Common.ProductsInCart)

{

if (count != 0) query.CommandText += @" OR";

query.CommandText += $@" Id={it}";

count++;

}

adapter.Fill(ds);

dgv.DataSource = ds.Tables[0];

dgv.Columns[0].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 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;

// стобцы 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)

{

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]=@descripton, [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);

query.Parameters.AddWithValue("@hdd", dgv.Rows[8].Cells[3].Value);

query.Parameters.AddWithValue("@ssd", dgv.Rows[9].Cells[3].Value);

query.Parameters.AddWithValue("@os", dgv.Rows[10].Cells[3].Value);

query.Parameters.AddWithValue("@psu", dgv.Rows[11].Cells[3].Value);

query.Parameters.AddWithValue("@stock", dgv.Rows[12].Cells[3].Value);

query.Parameters.AddWithValue("@cost", dgv.Rows[13].Cells[3].Value);

query.Parameters.AddWithValue("@descripton", dgv.Rows[14].Cells[3].Value);

if (dataImage != null) query.Parameters.AddWithValue("@image", dataImage);

else query.CommandText = query.CommandText.Replace("@image", "NULL");

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.IsNullOrWhiteSpace(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 распозновании 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.IsNullOrWhiteSpace(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

{

bool ok = false;

SQLCommandToUpdate = @"SELECT \* FROM Products";

// добавление в команду для обновления запрос по фильтрам

if (\_TakeAccountOfFiltersChecked() != -1)

if (\_TakeAccountOfFiltersFromTo() != -1)

ok = true;

if (ok)

{ // 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 Filters\_FormClosing(object sender, FormClosingEventArgs e)

{

}

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.IsNullOrWhiteSpace(filter.from.Text) && string.IsNullOrWhiteSpace(filter.to.Text))

continue;

// если команда не была модифицирована на более ранних стадиях

if (SQLCommandToUpdate.Contains("WHERE"))

SQLCommandToUpdate += " AND";

else SQLCommandToUpdate += " WHERE";

if (!string.IsNullOrWhiteSpace(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.IsNullOrWhiteSpace(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();

// для взаимодействия других форм с главной

//--- SQL-запрос для обовления 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();

UserStateChange();

QueryToUpdate = @"SELECT \* FROM Products";

UpdateDataGridView();

}

private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if (MessageBox.Show("Закрыть 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.IsNullOrWhiteSpace(QueryToUpdate))

{

MessageBox.Show("Неверная команда! Будет выполнен запрос по-умолчанию.", "Ошибка!",

MessageBoxButtons.OK, MessageBoxIcon.Error);

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 dataGridView\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs 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.IsNullOrWhiteSpace(QueryToUpdate))

UpdateDataGridView();

}

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

{

QueryToUpdate = null;

Filters filters = new Filters();

filters.ShowDialog();

if (!string.IsNullOrWhiteSpace(QueryToUpdate))

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 UserStateChange()

{

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 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)

{

Profile profile = new Profile();

profile.ShowDialog();

}

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

{

if (user.Status != UserStatus.Guest)

{

User.Logout();

Common.ProductsInCart.Clear();

UserStateChange();

}

}

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

{

Authorization auth = new Authorization();

auth.ShowDialog();

UserStateChange();

}

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

{

Registration reg = new Registration();

reg.ShowDialog();

UserStateChange();

}

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();

UserStateChange();

}

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)

{

if (saveFile.Filter == "CSV files(\*.csv)|\*.csv")

{

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() + "\t";

fileInStr += "\r\n";

}

StreamWriter streamWriter = new StreamWriter(saveFile.FileName, false, Encoding.GetEncoding("Windows-1251"));

streamWriter.Write(fileInStr);

streamWriter.Close();

}

else

{

ExcelApplication ExcelApp = new ExcelApplication();

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";

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();

}

}

ExcelApp.AlertBeforeOverwriting = false;

ExcelApp.DisplayAlerts = false;

workbook.SaveAs(saveFile.FileName);

ExcelApp.Quit();

}

saveFile.FileName = "Products\_DET\_Shop.xlsx";

}

}

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 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);

}

}

}

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)

{

lableProductName.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)

bAddToCart.Enabled = false;

}

private void \_printUserNickToReview(int id)

{

if (id <= 0) return;

SqlConnection connection = new SqlConnection(Common.StrSQLConnection);

try

{

string commandText = @"SELECT [Nick] FROM [Users] WHERE [Id]=" + id;

connection.Open();

SqlCommand query = new SqlCommand(commandText, connection);

object result = query.ExecuteScalar();

if (result != null)

richTextBoxReviews.Text += "=== Отзыв от пользователя " + result.ToString();

else

richTextBoxReviews.Text += "=== Отзыв от неизвестного пользователя";

}

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 async void loadReviews()

{

SqlConnection connection = 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())

{

\_printUserNickToReview((int)reader.GetValue(0));

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 (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.IsNullOrWhiteSpace(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 async 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;

}

SqlConnection connection = new SqlConnection(Common.StrSQLConnection);;

SqlCommand query = new SqlCommand($@"SELECT \* FROM Products WHERE Id={product\_id}", connection);

try

{

await connection.OpenAsync();

SqlDataReader reader = await query.ExecuteReaderAsync();

if (reader.HasRows)

if (await reader.ReadAsync())

{

Common.ProductsInCart.Add(product\_id);

MessageBox.Show("Товар успешно добавлен в корзину", "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 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)

{

dataGridView.RowHeadersVisible = false;

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 bAdd\_Click(object sender, EventArgs e)

{

AddProduct newProduct = new AddProduct();

newProduct.ShowDialog();

if (updateFlag)

{

updateData();

updateFlag = false;

}

}

private void bUpdate\_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 bDel\_Click(object sender, EventArgs e)

{

if (dataGridView.SelectedRows.Count == 1)

{

string title = "Подтвердите действие";

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

string recordToDelete = null;

qst += "\nЗапись \"";

for (int i = 0; i < 4; i++)

{

if (i != 0) recordToDelete += " | ";

recordToDelete += dataGridView.SelectedCells[i].Value.ToString();

}

recordToDelete += "\"";

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();

}

else MessageBox.Show("Вы не выбрали товар для удаления", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

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 иземенения?", "Подтверждение сохранения",

MessageBoxButtons.YesNoCancel, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

if (FileTools.PutBytesToDB(dataImage, Common.StrSQLConnection, @"Users", @"Picture", User.GetUser().Id))

MessageBox.Show("Файл загружен в базу данных.", "Сообщение");

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.IsNullOrWhiteSpace(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 = Image.FromStream(new MemoryStream(dataImage));

else pictureBox.Image = Properties.Resources.nofoto;

}

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 (pictureBox.Image != null) Unsaved = true;

}

}

}

Файл Registration.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Globalization;

using System.Linq;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace AIS\_shop

{

public partial class Registration : Form

{

SqlConnection connection;

public Registration()

{

InitializeComponent();

}

private void Registration\_Load(object sender, EventArgs e)

{

if (User.GetUser().Status == UserStatus.Admin)

regAsAdmin.Visible = true;

}

private void Registration\_FormClosing(object sender, FormClosingEventArgs e)

{

if (connection != null && connection.State != ConnectionState.Closed)

connection.Close();

}

private async void bRegistration\_Click(object sender, EventArgs e)

{

if (validation())

{

int status;

if (regAsAdmin.Checked)

status = 2;

else

status = 1;

try

{

connection = new SqlConnection(Common.StrSQLConnection);

connection.Open();

SqlCommand command = new SqlCommand();

command.Connection = connection;

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.Normal;

break;

case 2:

userStatus = UserStatus.Admin;

break;

default:

MessageBox.Show("Ошибка чтения данных о пользователе из БД.\nВход будет выполненен 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.IsNullOrWhiteSpace(textBoxName.Text) ||

string.IsNullOrWhiteSpace(textBoxSurname.Text) ||

string.IsNullOrWhiteSpace(textBoxEmail.Text) ||

string.IsNullOrWhiteSpace(textBoxNick.Text) ||

string.IsNullOrWhiteSpace(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 зарегистриррован!", "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.IsNullOrWhiteSpace(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]\*[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)

{

}

}

}