|  |
| --- |
| 09.02.07 ПР-419  **ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**  **ПМ.11 Разработка, администрирование и защита баз данных** |
| **УП.11.02 Разработка приложения** |

|  |  |  |  |
| --- | --- | --- | --- |
| Практикант |  |  | Пустовой А.В.  Сумаренко В.О. |
| Руководитель практики |  |  | **Пьяненкова А.В.** |
|  |  |  |  |

Оглавление

[УП.11.02 Разработка приложения 3](#_Toc116632133)

[**База практики** 3](#_Toc116632134)

[**Период прохождения практики** 3](#_Toc116632135)

[**Основная задача практики** 3](#_Toc116632136)

[**Виды работ** 3](#_Toc116632137)

[**Задание на разработку** 4](#_Toc116632138)

[Описание организации данных 5](#_Toc116632139)

[**Логическая модель данных** 5](#_Toc116632140)

[**Схема данных** 5](#_Toc116632141)

[**Описание таблиц** 6](#_Toc116632142)

[**Описание интерфейса** 7](#_Toc116632143)

[**Описание отчетов** 16](#_Toc116632144)

[**Заключение** 19](#_Toc116632145)

[Используемые источники информации 20](#_Toc116632146)

**УП.11.02 Разработка приложения**

**База практики**

Практика проводилась на базе полигона вычислительной техники БПОУ ОО «Омский авиационный колледж имени Н.Е. Жуковского»

**Период прохождения практики**

Практика Разработка приложения по разработке программных продуктов проходила с **31.10.2022 г.** по **05.11.2022 г.** на полигоне вычислительной техники Омавиат.

**Основная задача практики**

Основная задача практики - формирование практических навыков по разработке кода программного продукта на основе готовой спецификации на уровне модуля, проведения тестирования программного модуля по определенному сценарию, использования основных методологий процессов разработки программного обеспечения, работы с объектами базы данных в конкретной СУБД.

**Виды работ**

* создание базы данных;
* разграничение данных пользователей;
* разработка приложения;
* документирование и презентация решений.

**Задание на разработку**

**Описание**

Требуется разработать программное решение для учета платежей физических лиц. Используя полученный программный продукт, физические лица могут вести учет своих платежей. Кроме того, появится возможность производить несложный анализ затрат в разрезе периодов или категорий. Также можно получать результаты анализа в печатной форме.

**Требования**

**Разграничение данных пользователей**

1. Программное решение должно быть многопользовательским;
2. Пароли должны обязательно храниться в зашифрованном виде;
3. Логин можно выбирать из выпадающего списка;

**Окно с основными данными**

1. В окне основных данных выводятся только данные текущего пользователя;
2. Данные должны помещаться на один экран по ширине;
3. Данные можно отображать за выбранный период (от даты до даты);
4. Данные можно отображать по одной из категорий;
5. Данные можно добавлять;
6. Данные можно удалять;
7. По отображаемым данным можно получить отчет;

**Добавление платежей**

1. При добавлении данных можно указывать категорию платежа;
2. При добавлении данных обязательно указывается назначение платежа;
3. Назначение платежа указывается на русском языке;
4. Назначение платежа состоит как минимум из 3 букв;
5. Едино разово можно совершить несколько одинаковых платежей;
6. Количество обязательно должно быть целым положительным числом;
7. Для платежа обязательно указывается цена в рублях;
8. Цена не может быть отрицательной;
9. Стоимость платежа рассчитывается как количество \* цена;
10. Дата платежа указывается как текущее время на компьютере;

**Удаление платежей**

1. При удалении данных система должна запрашивать подтверждение;
2. В подтверждении нужно выводить назначение платежа и другую значимую информацию;
3. Удаление должно сопровождаться звуковым или визуальным оформлением;

**Отчет**

1. Отчет может выводиться встроенными средствами, либо в распространенных программных продуктах (Microsoft Office Word, Microsoft Office Excel, Internet-браузер);
2. У отчета обязательно должен быть заголовок на первой странице;
3. В заголовке может быть указан период и категория платежей;
4. На каждой странице указывается ФИО пользователя и номер страницы;
5. Платежи группируются по категориям;
6. Внутри каждой категории платежи сортируются по дате;
7. В конце отчета выводится суммарная стоимость всех показанных платежей.

Описание организации данных

**Логическая модель данных**

Описание основных объектов разрабатываемой системы

|  |  |
| --- | --- |
| Объект | Описание |
| Пользователь | Информация о пользователях системы. |
| Продукт | Информация о продуктах пользователей, зарегистрированных в системе |
| Категория | Информация о категориях продуктов пользователей, зарегистрированных в системе |
| Платежи | Информация о платежах пользователей, зарегистрированных в системе |

На основе предоставленных данных была создала ER-диаграмма в Visual Paradigm

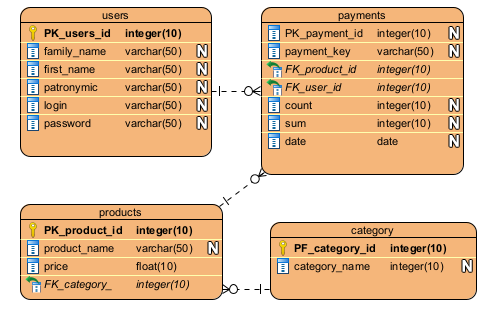
**

Рисунок 1. ER-диаграмма.

**Схема данных**

Схема в БД SQL, перенесённая из Visual Paradigm

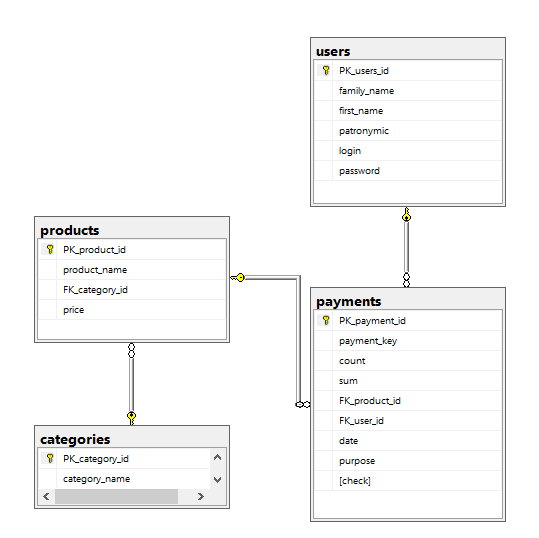
****

Рисунок 2. Диаграмма данных созданная в SQL

**Описание таблиц**

Таблица 1. Категории продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_category\_id | Уникальный идентификатор категории | int | 10 |
| category\_name | Название категории | nvarchar | 50 |

Таблица 2. Платежи

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_payment\_id | Уникальный идентификатор платежа | int | 10 |
| payment\_key | Уникальное имя платежа | nvarchar | 50 |
| count | Кол-во продуктов | int | 10 |
| sum | Стоимость продуктов | money |  |
| FK\_product\_id | Внешний ключ продукта | int | 10 |
| FK\_user\_id | Внешний ключ пользователя | int | 10 |
| date | Дата платежа | date | 20 |

Таблица 3. Продукты

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_product\_id | Уникальный идентификатор продукта | int | 10 |
| product\_name | Имя продукта | nvarchar | 100 |
| FK\_category\_id | Внешний ключ категории продукта | int | 10 |
| Price | Стоимость продукта | money |  |

Таблица 4. Пользователь

|  |  |  |  |
| --- | --- | --- | --- |
| **Название поля** | **Описание** | **Тип значения/поля** | **Кол-во знаков** |
| PK\_users\_id | Уникальный идентификатор пользователя | int | 10 |
| family\_name | Фамилия пользователя | nvarchar | 50 |
| first\_name | Имя пользователя | nvarchar | 50 |
| patronymic | Отчество пользователя | nvarchar | 50 |
| login | Логин пользователя | nvarchar | 50 |
| password | Пароль пользователя | nvarchar | 50 |

**Описание интерфейса**

**Форма авторизации**

После загрузки приложения первоначально появляется окно авторизации, на котором расположены поля для ввода данных пользователя.

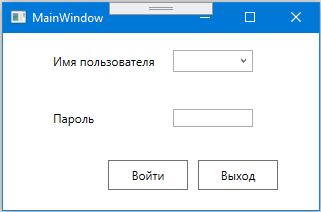


Рисунок 3. Окно авторизации

После ввода логина и пароля, при нажатии на кнопку входа, если введенные данных совпадет с теми, что находятся в базе, открывается главное окно.

Листинг 1. Авторизация

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

{

login = loginText.Text;

pass = ToSha256(loginText.Text);

ChouseData chouseData;

//Поиск пользователя в базе данных

try {

var user = Instances.db.users.Where(p => p.login == login && p.password == pass).FirstOrDefault();

if (user != null && loginText.Text == passwordText.Text)

{

chouseData = new ChouseData(user);

chouseData.Show();

this.Close();

}

else

{

//Обработка количества попыток

if (tryes < 3)

{

tryes++;

MessageBox.Show("Неверный логин или пароль \n Попыток: "+(3-tryes));

Capcha capchaWindow = new Capcha();

capchaWindow.Show();

this.Visibility=Visibility.Collapsed;

}

else

{

//Таймер при максимальном количестве неудачных попыток

currentTime = 0;

MessageBox.Show("Недопустимое количество попыток. \n Необходимо подождать 30 секунд");

this.IsEnabled = false;

textLock.Visibility=Visibility.Visible;

timer.Tick += new EventHandler(timerTick);

timer.Interval = new TimeSpan(0, 0, 1000);

timer.Start();

tryes = 0;

}

}

}

catch

{

MessageBox.Show("Такого пользователя не существует");

}

}

// Обработка тика таймера

private void timerTick(object sender, EventArgs e)

{

if (currentTime <= 30)

{

currentTime++;

textLock.Text = "Необходимо подождать: " + currentTime;

}

else

{

textLock.Visibility = Visibility.Collapsed;

this.IsEnabled = true;

timer.Stop();

}

}

//Кодировка в Sha256

private string ToSha256(string x)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] source = Encoding.UTF8.GetBytes(x);

byte[] hash = sha256Hash.ComputeHash(source);

x = BitConverter.ToString(hash).Replace("-", String.Empty);

}

return x;

}

}

Если данные не совпадают с теми, что находятся в базе, то появляется окно с уведомлением о том, что мы ввели неверный логин или пароль и открывается окно с капчей.

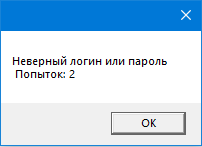


Рисунок 4. Окно уведомления об ошибке



Рисунок 5. Окно капчи

Листинг 2. Капча

public Capcha()

{

InitializeComponent();

GenerateCapcha();

}

private void EnterCapcha\_TextChanged(object sender, TextChangedEventArgs e)

{

if (enterCapcha.Text.Count() == 10)

{

if (enterCapcha.Text == textCapcha.Text)

{

enterCapcha.IsEnabled = false;

MainWindow main = new MainWindow();

main.Visibility=Visibility.Visible;

this.Close();

}

else

{

textCapcha.Text = "";

enterCapcha.Text = "";

GenerateCapcha();

}

}

}

//генерация капчи

private void GenerateCapcha()

{

enterCapcha.Text = "";

textCapcha.Text = "";

string allowchar = "";

allowchar = "A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z";

allowchar += "a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,y,z";

allowchar += "1,2,3,4,5,6,7,8,9,0";

char[] a = { ',' };

string[] ar = allowchar.Split(a);

string pwd = "";

string temp = "";

Random r = new Random();

for (int i = 0; i < 10; i++)

{

temp = ar[(r.Next(0, ar.Length))];

pwd += temp;

}

textCapcha.Text = pwd;

}

}

**Форма с основными данными**

После успешной авторизации открывается главное окно с данными пользователя

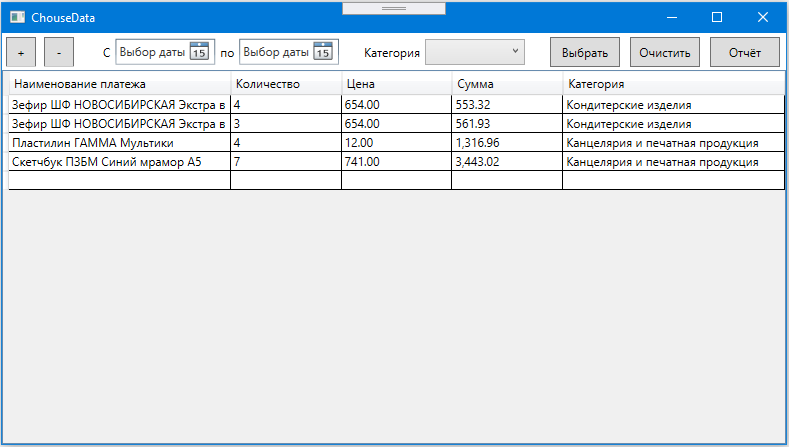


Рисунок 6. Главное окно

При вводе определенного периода в полях с датами платежи отображаются в соответствие с этим периодом времени.

При выборе категории из выпадающего списка отображаются платежи, которые ей соответствуют.

При нажатии на кнопку очистки, список очищается.

Листинг 3. Окно с платежами пользователей

public partial class ChouseData : Window

{

users user;

public ChouseData(users user)

{

InitializeComponent();

if (user != null) this.user = user;

datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id).ToList();

}

//добавление записи

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

{

AddData addData = new AddData(user);

bool result = (bool)addData.ShowDialog();

if (!result) datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id).ToList();

}

//удаление записей

private void RemData(object sender, RoutedEventArgs e)

{

var dataFR = datagride.SelectedItems.Cast<payments>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие {dataFR.Count()} элементов?", "Внимение",

MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes)

{

try

{

Instances.db.payments.RemoveRange(dataFR);

Instances.db.SaveChanges();

MessageBox.Show("Данные удалены.");

datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id).ToList();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message.ToString());

}

}

}

//выбор начальной даты

private void DateFrom\_CalendarClosed(object sender, RoutedEventArgs e)

{

if (dateTo.SelectedDate != null)

datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id && p.date >= dateFrom.SelectedDate && p.date <= dateTo.SelectedDate).ToList();

else datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id && p.date >= dateFrom.SelectedDate).ToList();

}

//выбор конечной даты

private void DateTo\_CalendarClosed(object sender, RoutedEventArgs e)

{

if (dateFrom.SelectedDate != null)

datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id && p.date >= dateFrom.SelectedDate && p.date <= dateTo.SelectedDate).ToList();

else datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id && p.date >= dateTo.SelectedDate).ToList();

}

//выбор категории

private void ComboBox\_DropDownClosed(object sender, EventArgs e)

{

if (comboBoxCategory.SelectedItem != null)

{

datagride.ItemsSource = Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id&&p.products.FK\_category\_id==comboBoxCategory.SelectedIndex).ToList();

}

}

private void MarkColors()

{

}

//очистка списка

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

{

datagride.Items.Clear();

}

При выборе платежей и нажатии на кнопку со знаком “-” появляется окно с запросом на подтверждение удаления данных.

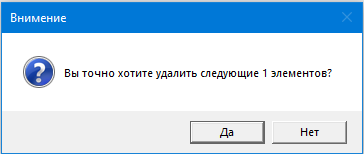


Рисунок 9.Окно с запросом на подтверждение удаления данных

При нажатии на кнопку “Отмена” окно добавления платежа закрывается, и мы возвращаемся на главный экран, введенный данный не сохраняются.

При нажатии на кнопку со знаком “+” появляется окно добавления платежей.

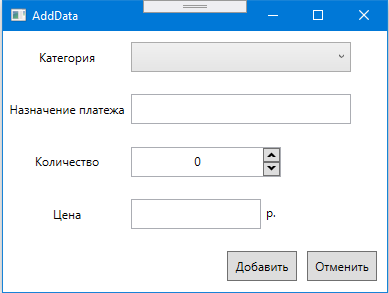


Рисунок 7. Окно добавления платежей

Из выпадающего списка можно выбрать категорию платежа. Далее необходимо ввести в поле ввода ниже назначение платежа, а затем мы можем выбрать количество платежей. В поле с ценой вводим стоимость платежа.

После заполнения всех данных и нажатия на кнопку добавления, если данные введены корректно, то появляется окно с уведомлением об успешном добавление платежа, данное окно закрывается, и мы переходим на главный экран, в списке отображается добавленный платёж.

При условии, если данные не удовлетворяют требованиям, на экран выводится уведомление с описанием ошибки.

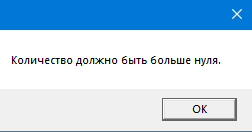


Рисунок 8. Окно уведомления с описанием ошибки

Листинг 4. Окно добавления платежей

public AddData(users user)

{

InitializeComponent();

this.user = user;

DataContext = \_payments;

comboCategor.ItemsSource = Instances.db.categories.ToList();

}

private void Cancle(object sender, RoutedEventArgs e)

{

this.Close();

}

private void addData(object sender, RoutedEventArgs e)

{

StringBuilder errors = new StringBuilder();

if (comboCategor.SelectedItem == null)

errors.AppendLine("Выберите категорию.");

if (string.IsNullOrWhiteSpace(purOfPay.Text))

errors.AppendLine("Введите назначение платежа.");

if (qual.Text.Length < 0)

errors.AppendLine("Укажите количество.");

if (Convert.ToInt32(qual.Text) <= 0)

errors.AppendLine("Количество должно быть больше нуля.");

if (price.Text.Length < 0)

errors.AppendLine("Укажите цену.");

if (errors.Length > 0)

{

MessageBox.Show(errors.ToString());

}

else

{

products pr = new products { product\_name = purOfPay.Text, FK\_category\_id = comboCategor.SelectedIndex, price = Convert.ToDecimal(price.Text) };

Instances.db.products.Add(pr);

Instances.db.SaveChanges();

payments pd = new payments

{

payment\_key = comboCategor.SelectedItem.ToString().Substring(0, 1) + (Instances.db.payments.Where(p => p.FK\_user\_id == user.PK\_users\_id).Count() + 1).ToString() + DateTime.Now.Date.ToString(),

count = Convert.ToInt32(qual.Text),

sum = Convert.ToInt32(qual.Text) \* Convert.ToInt32(price.Text),

FK\_product\_id = Instances.db.products.Count()+1,

FK\_user\_id = user.PK\_users\_id ,

date = DateTime.Now.Date,

};

Instances.db.payments.Add(pd);

}

Instances.db.SaveChanges();

MessageBox.Show("Платёж добавлен.");

}

}

При нажатии на кнопку отчёта, появляется окно сохранения.

**Описание отчетов**

Листинг 5. Отчёт

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

{

if (datagride.Items.Count > 0)

{

SaveFileDialog sfd = new SaveFileDialog();

sfd.Filter = "PDF (\*.pdf)|\*.pdf";

sfd.FileName = "Output.pdf";

bool fileError = false;

if (sfd.ShowDialog() == DialogResult)

{

if (File.Exists(sfd.FileName))

{

try

{

File.Delete(sfd.FileName);

}

catch (IOException ex)

{

fileError = true;

MessageBox.Show("Невозможно сохранить документ: " + ex.Message);

}

}

if (!fileError)

{

try

{

PdfPTable pdfTable = new PdfPTable(datagride.Columns.Count);

pdfTable.DefaultCell.Padding = 3;

pdfTable.WidthPercentage = 100;

pdfTable.HorizontalAlignment = Element.ALIGN\_LEFT;

foreach (DataGridColumn column in datagride.Columns)

{

PdfPCell cell = new PdfPCell(new Phrase(column.Header.ToString()));

pdfTable.AddCell(cell);

}

foreach (DataGridRow row in datagride.Items)

{

foreach (DataGridViewCell cell in row.Cells)

{

pdfTable.AddCell(cell.Value.ToString());

}

}

using (FileStream stream = new FileStream(sfd.FileName, FileMode.Create))

{

Document pdfDoc = new Document(PageSize.A4, 10f, 20f, 20f, 10f);

PdfWriter.GetInstance(pdfDoc, stream);

pdfDoc.Open();

pdfDoc.Add(pdfTable);

pdfDoc.Close();

stream.Close();

}

MessageBox.Show("Data Exported Successfully !!!", "Info");

}

catch (Exception ex)

{

MessageBox.Show("Error :" + ex.Message);

}

}

}

}

else

{

MessageBox.Show("No Record To Export !!!", "Info");

}

}

}

}

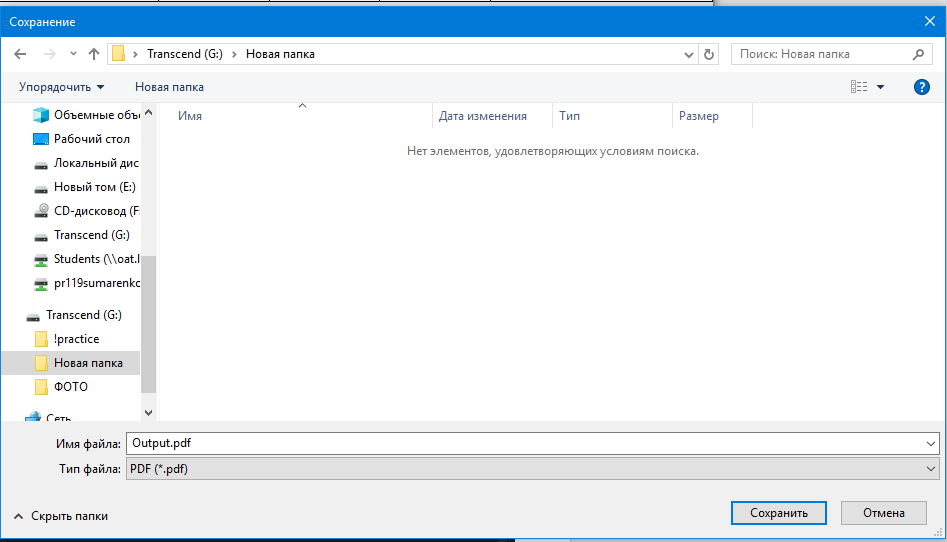


Рисунок 10. Окно сохранения

**Заключение**

В результате практики «УП.11.02 Разработка приложения» было проведено формирование практических навыков по разработке кода программного продукта на основе готовой спецификации на уровне модуля, проведения тестирования программного модуля по определенному сценарию, использования основных методологий процессов разработки программного обеспечения, работы с объектами базы данных в конкретной СУБД.

В период практики было разработано программное решение для учета платежей физических лиц. Используя полученный программный продукт, физические лица могут вести учет своих платежей. Кроме того, появится возможность производить несложный анализ затрат в разрезе периодов или категорий. Также можно получать результаты анализа в печатной форме.

Основные виды работ, которые были выполнены: создание форм приложения, подключение базы данных к приложению, авторизация пользователя, создание капчи, шифрование паролей, вывод данных пользователя, сортировка данных по дате и категориям, удаление платежей, добавление платежей, создание отчёта.

# Используемые источники информации

1. Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5, Matthew MacDonald (русский перевод: WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов, Мэтью Макдональд). Разбор XAML'а, лаконичные, но полезные примеры. Пристальное, но не навязчивое внимание к деталям.
2. METANIT.COM Сайт о программировании - https://metanit.com/sharp/entityframework/
3. Microsoft Learn - <https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022>
4. Основа WPF - Windows Presentation Foundation - <https://professorweb.ru/my/WPF/base_WPF/level1/base_WPF_index.php>
5. Habr.com - https://habr.com/ru/post/427325/