**министерство науки и высшего образования российской федерации**

**«Ростовский государственный экономический университет (РИНХ)»**

Факультет «Компьютерных технологий и информационной безопасности»

Кафедра «Информационных систем и прикладной информатики»

**ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

На тему **«**Информационная подсистема учета продаж сервисов»

Группа ПИZS-331

Направление 09.03.03 «Прикладная информатика»

Выполнил студент Чаленко Владимир Владимирович

Преподаватель Долженко Алексей Иванович

г. Ростов-на-Дону

2021

СОДЕРЖАНИЕ

[Введение 3](#_Toc87210319)

[1 Описание предметной области 4](#_Toc87210320)

[1.1 Описание задачи необходимых для формирования данных учета продаж 4](#_Toc87210321)

[2 Разработка моделей данных 5](#_Toc87210322)

[2.1 Создание модели данных 5](#_Toc87210323)

[3 Разработка представления данных по моделям 8](#_Toc87210324)

[3.1 Создание представления моделей и отображение данных 8](#_Toc87210325)

[4 Создание, редактирование и удаление данных форм 9](#_Toc87210326)

[4.1 Добавление данных 9](#_Toc87210327)

[4.2 Редактирование данных 10](#_Toc87210328)

[4.3 Удаление данных 11](#_Toc87210329)

[Заключение 12](#_Toc87210330)

[Список использованных источников 13](#_Toc87210331)

[Приложения](#_Toc87210332)

# **ВВЕДЕНИЕ**

Продажа изготовленной продукции представляет собой важнейший показатель деятельности производственной организации. Ведь именно продажей продукции завершается оборот средств, затраченных на ее изготовление. В результате продажи продукции организация – изготовитель получает оборотные средства, необходимые для возобновления нового цикла производственного процесса.

В рамках курсовой работы был проведен анализ существующих подсистем учета продаж сервисов. Было разработано приложение, которое помогает управлять учетом продаж, благодаря чему мы можем все время получать актуальную информацию.

Объектом исследования является информационная подсистема учета продаж сервисов.

Цель курсовой работы разработка приложения для учета продаж сервисов с помощью языка программирования C#.

# **Описание предметной области**

Назначение подсистемы поддержание в актуальном состоянии информации по продажам сервисов. *(Рисунок 1)*.



Рисунок 1 – Структура классов подсистемы учета продаж

* 1. **Описание задачи необходимых для формирования данных учета продаж**
* Разработка и сопровождение программных систем. Технологии Microsoft.NET для разработки приложений;
* Разработка моделей данных;
* Разработка представления данных по моделям;
* Создание, редактирование и удаление данных клиента;
* Создание, редактирование и удаление данных услуг;
* Создание, удаление данных покупки.

# **Разработка моделей данных**

* 1. **Создание модели данных**

Общие сведения:

Создается WPF-приложение для обработки данных учета продаж *Рисунок 1*. Приложение должно отображать данные учета продаж, добавлять, редактировать и удалять данные.

Создаем проект приложения на основе шаблона классического приложения на C#. В созданный проект добавим три папки *(Рисунок 2):*

1. Model;
2. View;
3. ViewModel.

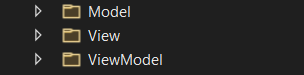


Рисунок 2 – Папки для проекта

В папке Model создаем классы, которые описывают предметную область *(Рисунок 3).*

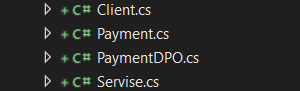


Рисунок 3 – Классы папки Model

В папке ViewModel создадим классы, в которых будут сформированы коллекции данных *(Рисунок 4).*

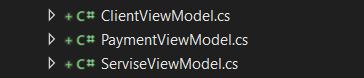


Рисунок 4 – Классы папки ViewModel

В папке View создадим окна для представления данных *(Рисунок 5).*

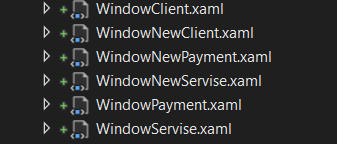


Рисунок 5 – Представления View

Для представления данных учета продаж, шаблон экранной формы будет иметь вид, представленный на *(Рисунок 6).*

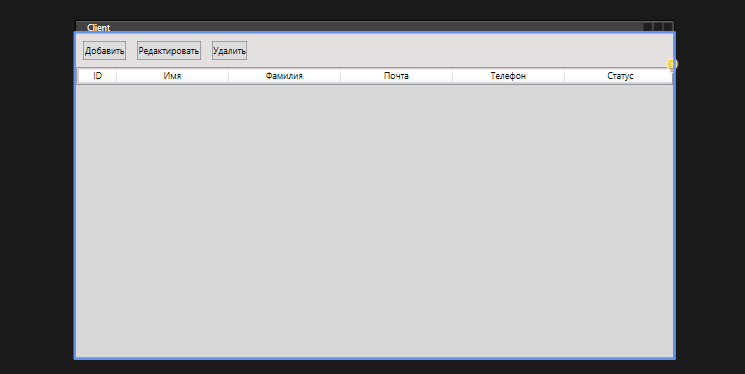


Рисунок 6 – Готовый шаблон для формы клиента

На главном окне приложения создадим меню для вывода и обработки данных *(Рисунок 7)*.

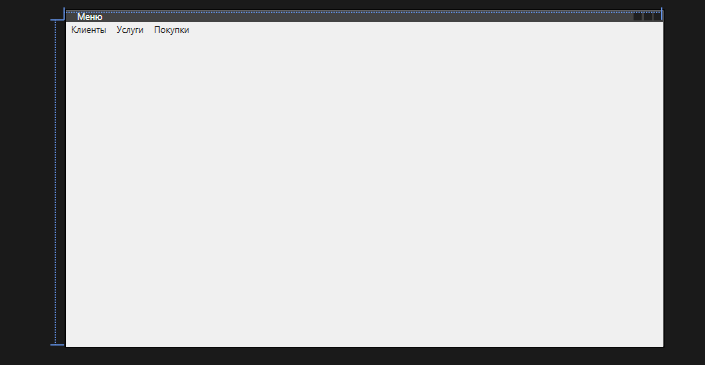


Рисунок 7 – Главное меню программы

XAML-разметка главной формы.

<Window x:Class="WpfApp2.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApp2"

mc:Ignorable="d"

Title="Меню" Height="450" Width="800">

<Grid>

<Menu>

<MenuItem x:Name="WNPClient" Header="Клиенты" Click="WNPClient\_Click"></MenuItem>

<MenuItem x:Name="WNPServise" Header="Услуги" Click="WNPServise\_Click"></MenuItem>

<MenuItem x:Name="WNPPayment" Header="Покупки" Click="WNPPayment\_Click"></MenuItem>

</Menu>

</Grid>

</Window>

# **Разработка представления данных по моделям**

* 1. **Создание представления моделей и отображение данных**

Для отображения списка будем использовать класс ListView, который предназначен для отображения данных спискового типа, состоящих из множества столбцов. Для класса ListView используется свойство View (ListView.View), которое обеспечивает стили и форматирование данных. Для создания в отображении списка столбцов будем использовать класс GridView, который предоставляет списковое представление со множеством столбцов. Столбцы определяются в коллекции GridView.Columns путем задания объектов GridViewColumn. На данном этапе проектирования приложения для объектов GridViewColumn зададим только заголовок столбца Header.

После сборки и запуска программы при вызове окна клиент экранная форма будет иметь вид *(Рисунок 8).*

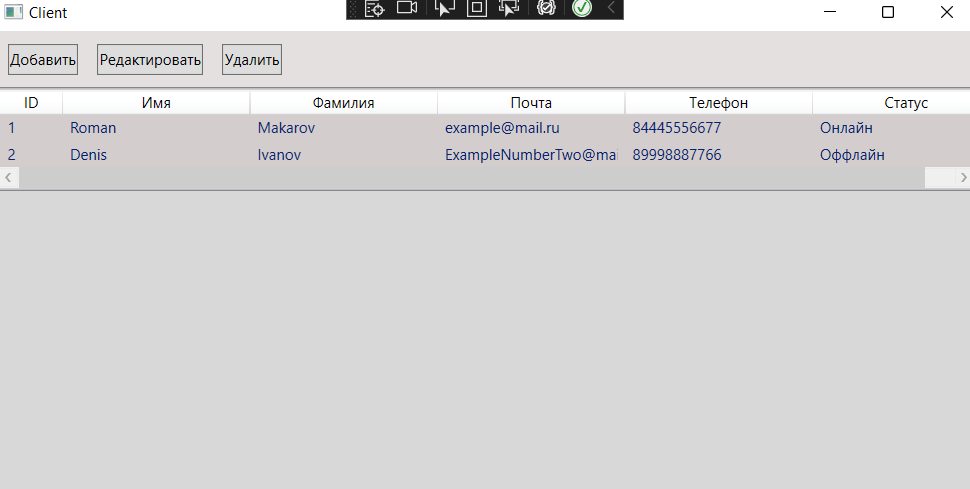


Рисунок 8 – Форма клиента

# **Создание, редактирование и удаление данных форм**

* 1. **Добавление данных**

Для обеспечения функциональности обработки данных добавим в окно формы кнопки добавить, удалить и редактировать *(Рисунок 9)*.

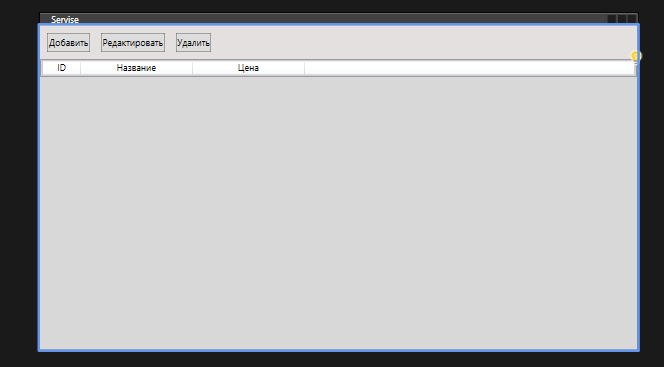


Рисунок 9 – Пример кнопок на форме услуг

При создании новой услуги её код должен вычисляться как максимальный код услуги в имеющихся данных, т.е. в списке услуг, увеличенный на единицу. Для определения максимального значения кода в списке добавим в класс ServiseViewModel метод MaxId().

public int MaxId()

{

int max = 0;

foreach (var p in this.ListServise)

{

if (max < p.ID)

{

max = p.ID;

};

}

return max;

}

* 1. **Редактирование данных**

При редактировании данных создадим для кнопки редактировать окна WindowsRole код обработчика btnEdit\_Click.

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

{

WindowNewServise ser = new WindowNewServise

{

Title = "Редактирование",

Owner = this

};

Servise srv = WNPServise.SelectedItem as Servise;

if (srv != null)

{

Servise tempsrv = srv.ShallowCopy();

ser.DataContext = tempsrv;

if (ser.ShowDialog() == true)

{

srv.Name = tempsrv.Name;

srv.Price = tempsrv.Price;

WNPServise.ItemsSource = null;

WNPServise.ItemsSource = servm.ListServise;

}

}

else

{

MessageBox.Show("Необходимо выбрать поле для редактированния",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

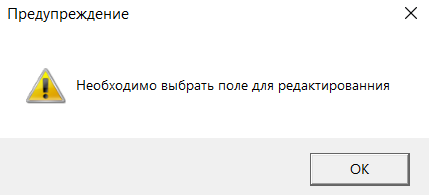


Рисунок 10 – Предупреждение редактирования

* 1. **Удаление данных**

При удалении создадим для кнопки удалить окна WindowsRole код обработчика btnDelete\_Click.

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

{

Servise srv = (Servise)WNPServise.SelectedItem;

if (srv != null)

{

MessageBoxResult result = MessageBox.Show("Удалить " + srv.ID +

srv.Name, "Предупреждение", MessageBoxButton.OKCancel,

MessageBoxImage.Warning);

if (result == MessageBoxResult.OK)

{

servm.ListServise.Remove(srv);

}

}

else

{

MessageBox.Show("Необходимо выбрать поле для удаления",

"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

Если выбор для удаления произведен, то выводим предупреждение *(Рисунок 11)*.

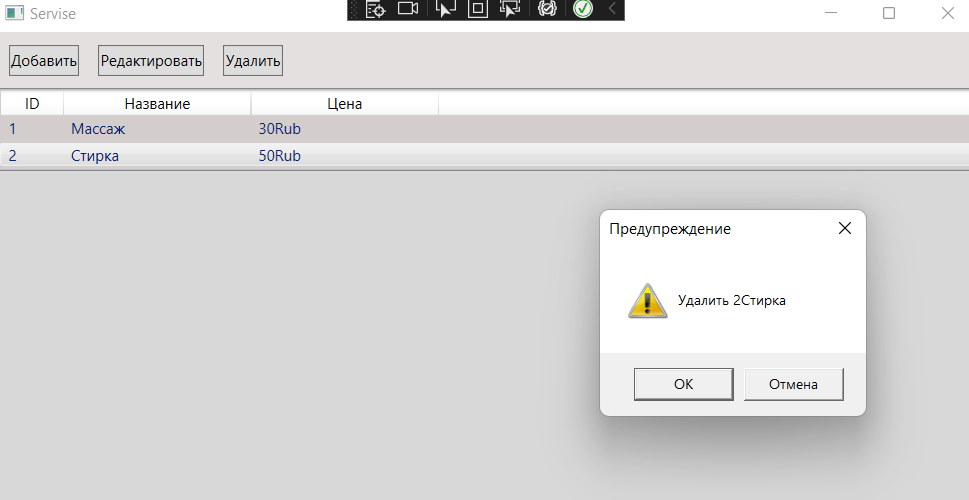


Рисунок 11 – Предупреждение об удалении данных

# **ЗАКЛЮЧЕНИЕ**

В ходе данной курсовой работы было создано приложение для управления учета продаж, которое позволяет мониторить информацию в реальном моменте времени. Приложение было разработано на платформе С#, WPF. Данная платформа предназначена для создания приложений на базе .NET. Была использована актуальная на сегодняшний день платформа компании Microsoft Visual Studio.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Основы WPF // <http://msdn.microsoft.com/ruru/library/ms746927.aspx>.
2. Мак-Дональд, М. WPF:Windows Presentation Foundation в .NET 3.0 для профессионалов.: Пер. с англ. –М.: ООО “И.Д. Вильямс”, 2008.
3. Нейгел, К. C# 4.0 и платформа .NET 4 для профессионалов. / К. Нейгел, Б. Ивьен, Дж. Глинн, К. Уотсон.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2011.
4. Петцольд Ч. Microsoft Windows Presentation Foundation. Базовый курс.: - М.: Изд. «Русская Редакция» СПб.: Питер, 2008.
5. Троелсен, Э. Язык программирования C# 2010 и платформа .NET 4.0, 5-е изд.: Пер. с англ. –М.: ООО “И.Д. Вильямс”, 2011.

# **ПРИЛОЖЕНИЕ A**

**Код программы С#**

1. Payment

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WpfApp2.ViewModel;

namespace WpfApp2.Model

{

//Класс, содержит поля необходимые по заданию

class Payment

{

public int ID { get; set; }

public int ClientID { get; set; }

public int ServiceID { get; set; }

public DateTime Date { get; set; }

public int Quantity { get; set; }

public string Amount { get; set; }

public Payment() { }

//Конструктор чтобы воспользоваться полями

public Payment(int id, int clientID, int service, DateTime date, int quantity, string amount)

{

this.ID = id;

this.ClientID = clientID;

this.ServiceID = service;

this.Date = date;

this.Quantity = quantity;

this.Amount = amount;

}

//нужно для правильного вывода полей, в которых содержится ID другого класса(в данном случае таких 2: Client и Servise)

public Payment CopyFromCompanyDPO(PaymentDPO pay)

{

ClientViewModel client = new ClientViewModel();

int clientid = 0;

foreach (var cl in client.ListClient)

{

if (cl.LastName == pay.ClientID)

{

clientid = cl.ID;

break;

}

}

ServiseViewModel servise = new ServiseViewModel();

int serviseid = 0;

foreach (var srv in servise.ListServise)

{

if (srv.Name == pay.ServiceID)

{

serviseid = srv.ID;

break;

}

}

if ((clientid != 0) && (serviseid != 0))

{

this.ID = pay.ID;

this.ClientID = clientid;

this.ServiceID = serviseid;

this.Date = pay.Date;

this.Quantity = pay.Quantity;

this.Amount = pay.Amount;

}

return this;

}

}

}

1. PaymentDPO

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using WpfApp2.ViewModel;

namespace WpfApp2.Model

{

class PaymentDPO

{

// Отличие от Payment только в том чтобы ID в строковом формате(Нужно для правильного отображения на форме)

public int ID { get; set; }

public string ClientID { get; set; }

public string ServiceID { get; set; }

public DateTime Date { get; set; }

public int Quantity { get; set; }

public string Amount { get; set; }

public PaymentDPO() { }

public PaymentDPO(int id, string clientID, string serviceID, DateTime date, int quantity, string amount)

{

this.ID = id;

this.ClientID = clientID;

this.ServiceID = serviceID;

this.Date = date;

this.Quantity = quantity;

this.Amount = amount;

}

public PaymentDPO CopyFromPerson(Payment pay)

{

PaymentDPO payDPO = new PaymentDPO();

ClientViewModel client = new ClientViewModel();

string statusString = string.Empty;

foreach (var cl in client.ListClient)

{

if (cl.ID == pay.ClientID)

{

statusString = cl.FirstName;

break;

}

}

ServiseViewModel serv = new ServiseViewModel();

string serString = string.Empty;

foreach (var srv in serv.ListServise)

{

if (srv.ID == pay.ServiceID)

{

serString = srv.Name;

break;

}

}

if ((statusString != string.Empty) && (serString != string.Empty))

{

payDPO.ID = pay.ID;

payDPO.ClientID = statusString;

payDPO.ServiceID = serString;

payDPO.Date = pay.Date;

payDPO.Quantity = pay.Quantity;

payDPO.Amount = pay.Amount;

}

return payDPO;

}

}

}