Описание

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

ТРЕБОВАНИЯ

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 (кнопка «Отчет»);
8. По отображаемым данным можно получить диаграмму (кнопка «Выбрать»);

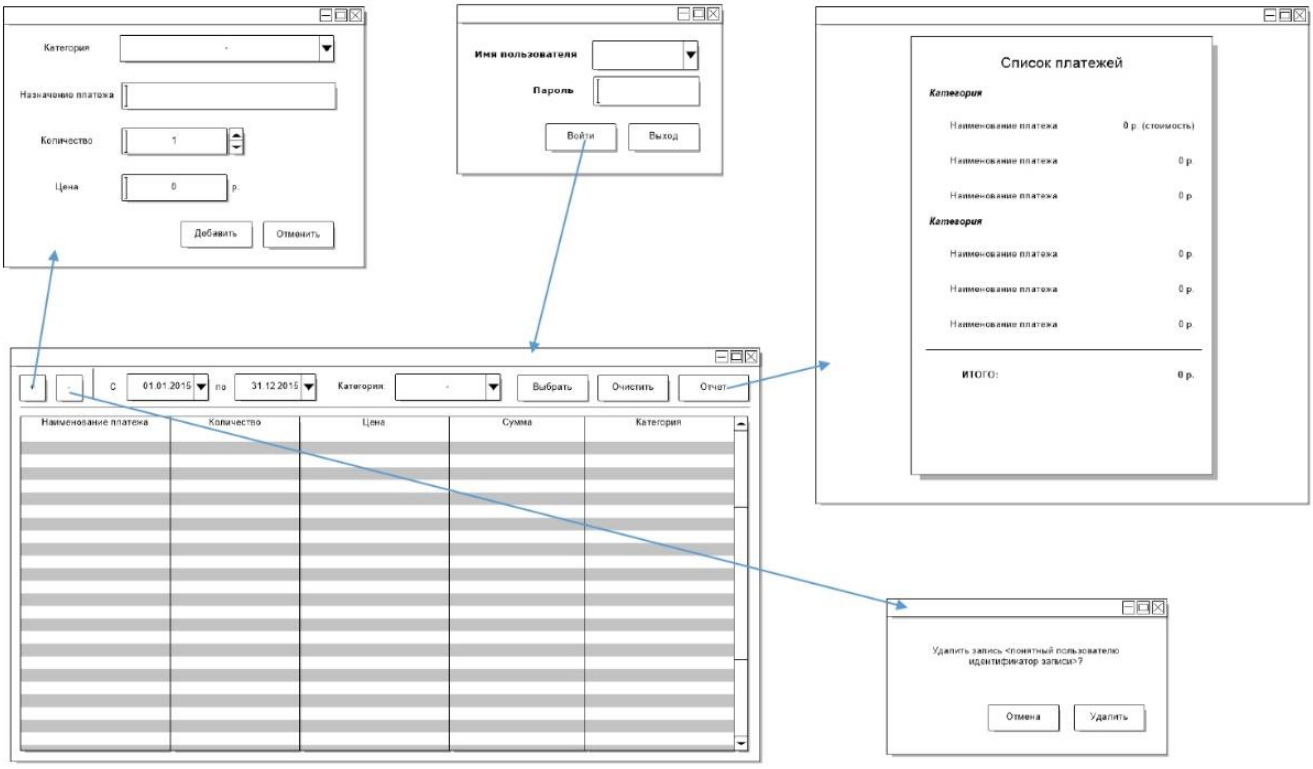


Рис.1 Переходы между экранными формами

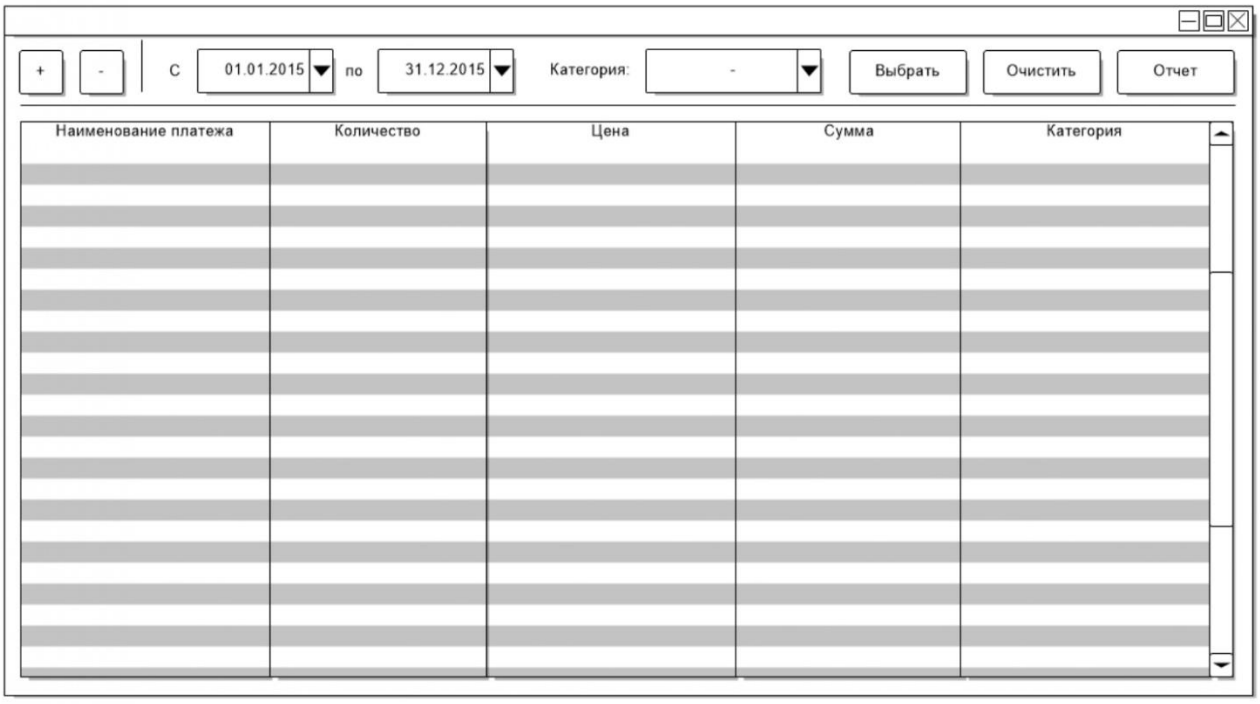
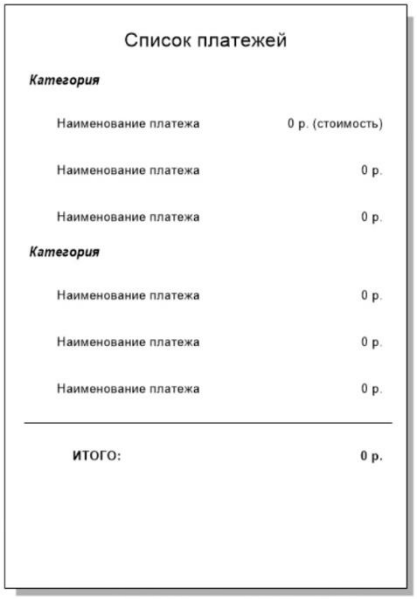
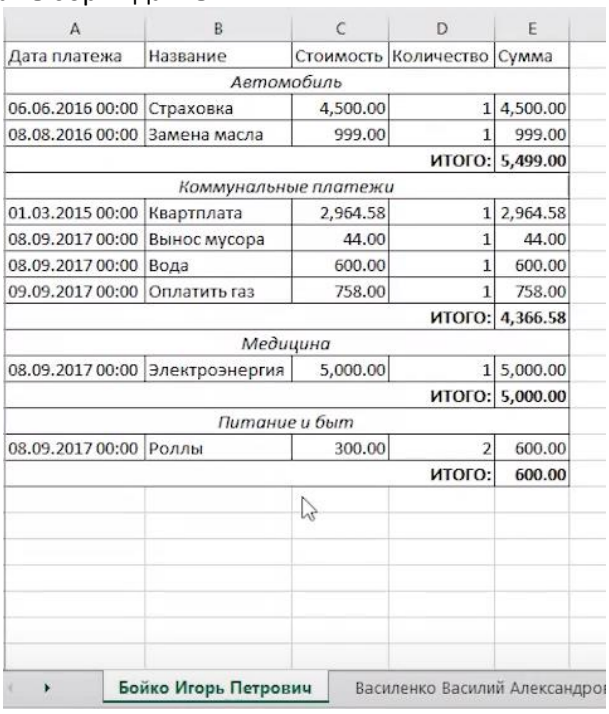


Рис.2 Форма выборки данных





Дата платежа	Название	Стоимость	Количество	Сумма
Автомобиль				
06.06.2016 00:00	Страховка	4,500.00	1	4,500.00
08.08.2016 00:00	Замена масла	999.00	1	999.00
			ИТОГО:	**5,499.00**
Коммунальные платежи				
01.03.2015 00:00	Квартплата	2,964.58	1	2,964.58
08.09.2017 00:00	Вынос мусора	44.00	1	44.00
08.09.2017 00:00	Вода	600.00	1	600.00
09.09.2017 00:00	Оплатить газ	758.00	1	758.00
			ИТОГО:	**4,366.58**
Медицина				
08.09.2017 00:00	Электроэнергия	5,000.00	1	5,000.00
			ИТОГО:	**5,000.00**
Питание и быт				
08.09.2017 00:00	Роллы	300.00	2	600.00
			ИТОГО:	**600.00**

Рис.3 Форма представления отчета

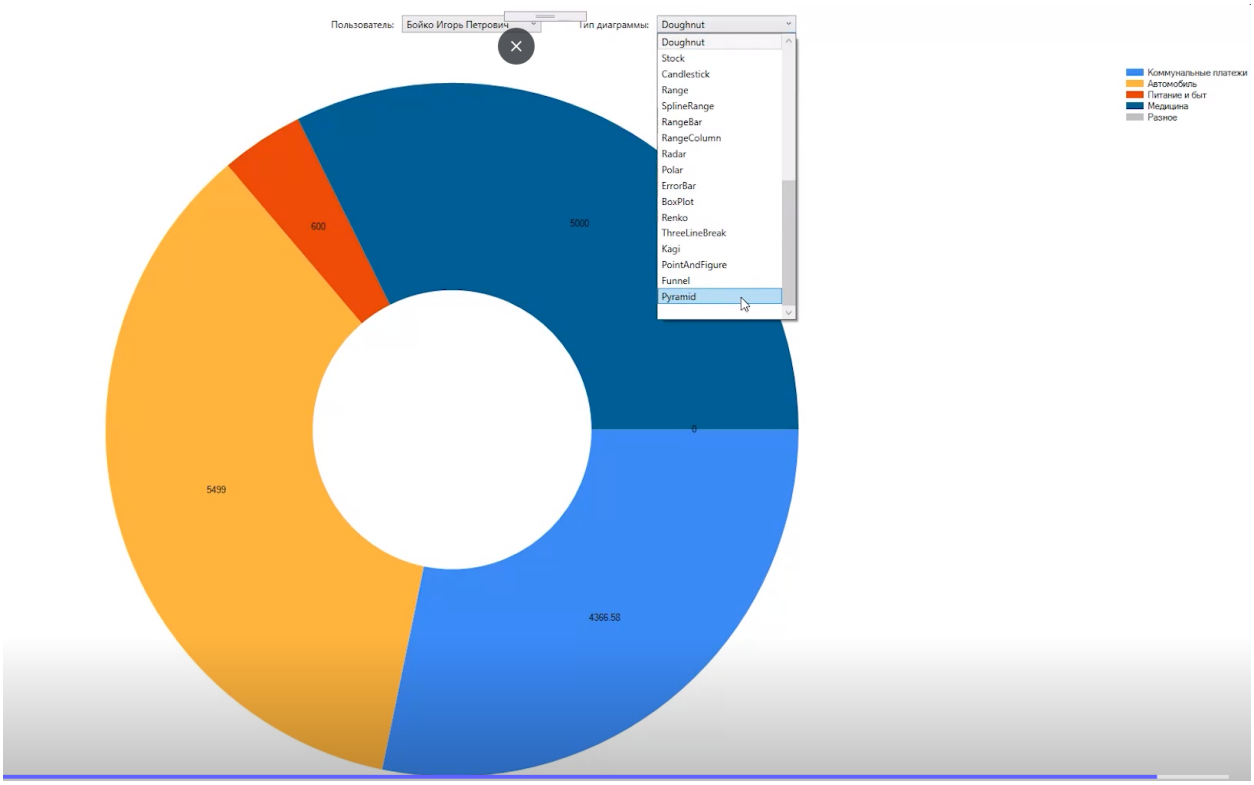


Рис.4 Форма представления диаграммы

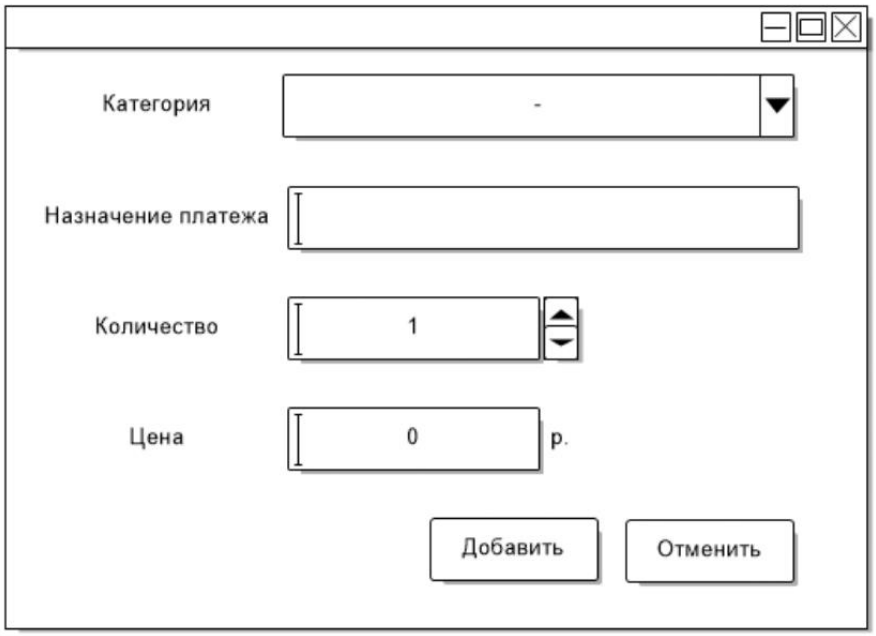


Рис.5 Форма добавления данных

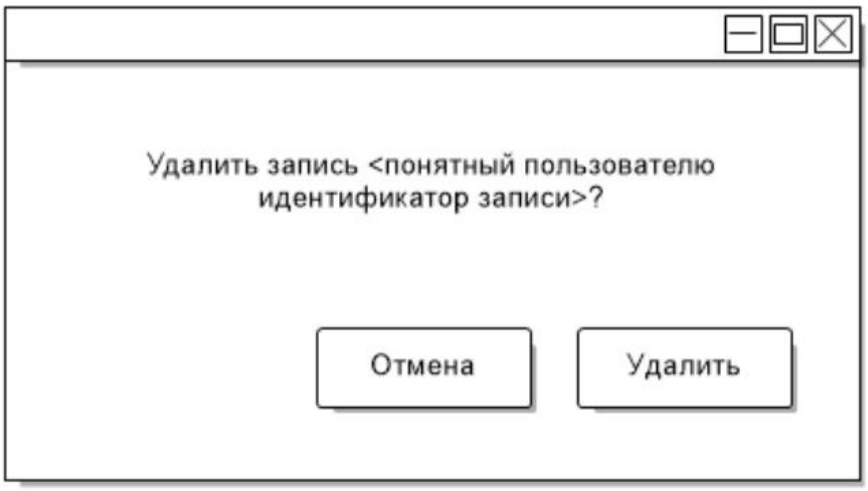


Рис.6 Форма удаления данных

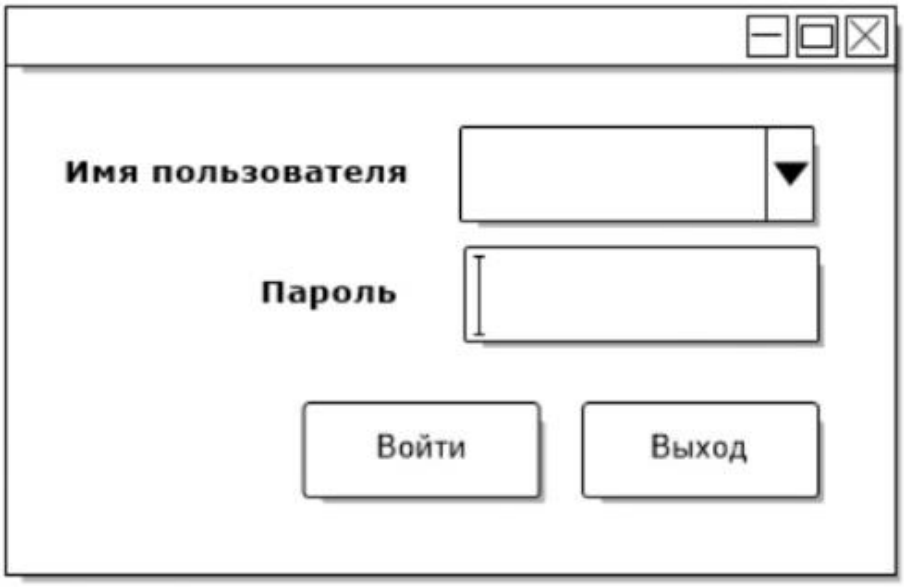


Рис.7 Форма идентификации пользователей

Создание приложения

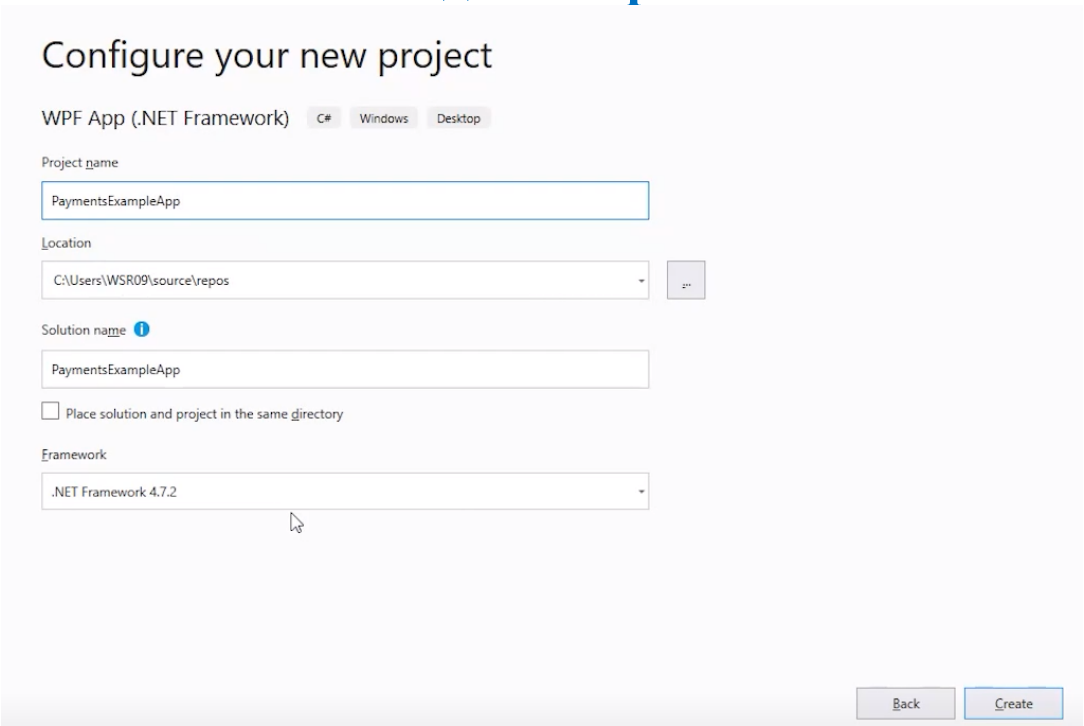


Рис.8 Создание проекта

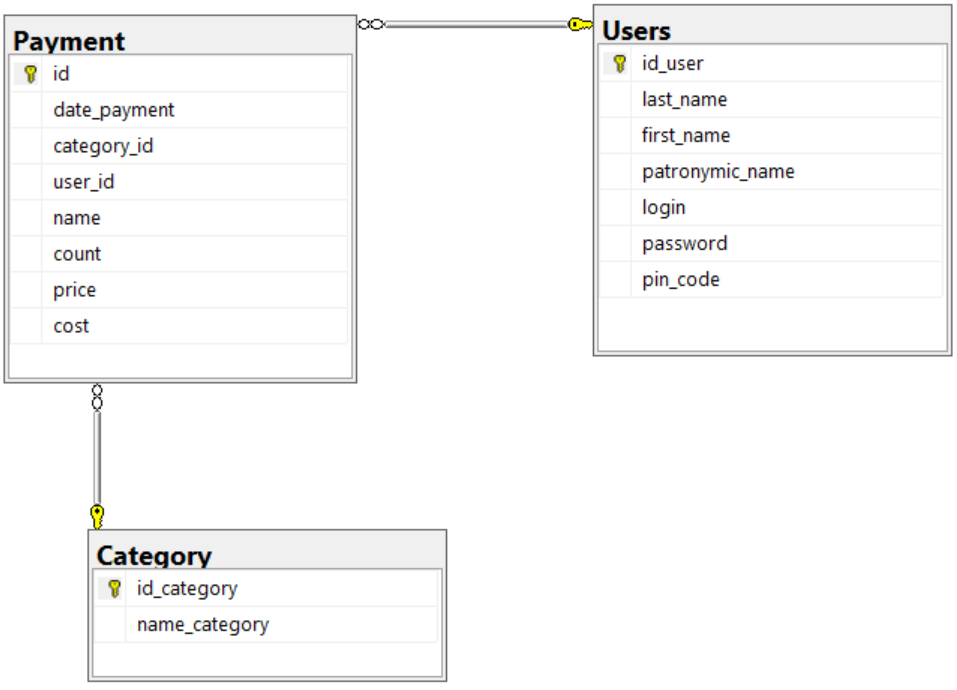


Рис.9 ERD-диаграмма

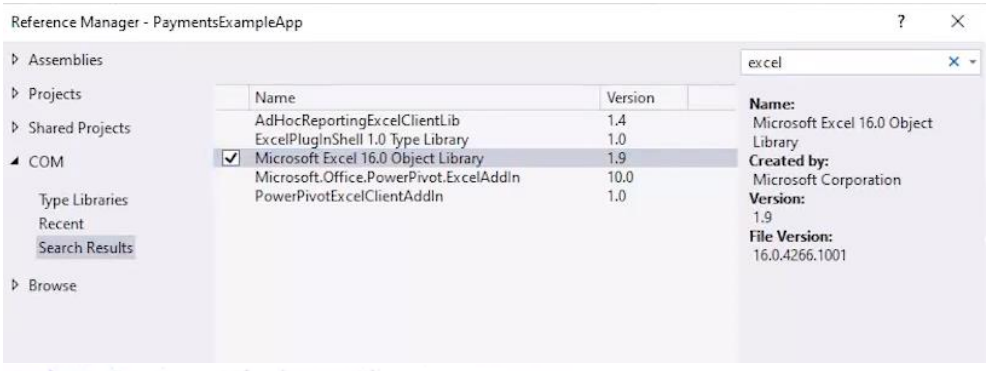
Вывод в Excel

Расходы каждого пользователя будут экспортироваться в отдельный лист, названием которого будет ФИО пользователя. Используем единую строку заголовков для вывода результатов по категориям: «Дата платежа», «Название», «Стоимость», «Количество», «Сумма»
Расходы будут распределены по категориям, причем по каждой категории будут указываться общие затраты.

Дата платежа	Название	Стоимость	Количество	Сумма
	Автомобиль			
06.06.2016 00:00	Страховка	4,500.00	1	4,500.00
08.08.2016 00:00	Замена масла	999.00	1	999.00
			ИТОГО:	**5,499.00**
	Коммунальные платежи			
01.03.2015 00:00	Квартплата	2,964.58	1	2,964.58
08.09.2017 00:00	Вынос мусора	44.00	1	44.00
08.09.2017 00:00	Вода	600.00	1	600.00
09.09.2017 00:00	Оплатить газ	758.00	1	758.00
			ИТОГО:	**4,366.58**
	Медицина			
08.09.2017 00:00	Электроэнергия	5,000.00	1	5,000.00
			ИТОГО:	**5,000.00**
	Питание и быт			
08.09.2017 00:00	Роллы	300.00	2	600.00
			ИТОГО:	**600.00**

Бойко Игорь Петрович | Василенко Василий Александрович | Дорофеева Анна Геннадьевна

Подключаем библиотеку для работы с Excel. Для экспорта данных в Excel используется библиотека Interop. Excel (Object library), расположенная во вкладке COM. Выполним подключение библиотеки.




```
using System.Windows.Shapes;
using Excel = Microsoft.Office.Interop.Excel;
```

```
0 references
private void BtnExportToExcel_Click(object sender, RoutedEventArgs e)
{
    var allUsers = _context.Users.ToList().OrderBy(p => p.FIO).ToList();

    var application = new Excel.Application();
    application.SheetsInNewWorkbook = allUsers.Count();

    Excel.Workbook workbook = application.Workbooks.Add(Type.Missing);
```

```
        for (int i = 0; i < allUsers.Count(); i++)
        {
            int startRowIndex = 1;

            Excel.Worksheet worksheet = application.Worksheets.Item[i + 1];
            worksheet.Name = allUsers[i].FIO;

            worksheet.Cells[1][startRowIndex] = "Дата платежа";
            worksheet.Cells[2][startRowIndex] = "Название";
            worksheet.Cells[3][startRowIndex] = "Стоимость";
            worksheet.Cells[4][startRowIndex] = "Количество";
            worksheet.Cells[5][startRowIndex] = "Сумма";

            startRowIndex++;

            var usersCategories = allUsers[i].Payments.OrderBy(p => p.Date).GroupBy(p => p.Category).OrderBy(p => p.Key.Name);

        foreach (var groupCategory in usersCategories)
        {
            Excel.Range headerRange = worksheet.Range[worksheet.Cells[1][startRowIndex], worksheet.Cells[5][startRowIndex]];
            headerRange.Merge();
            headerRange.Value = groupCategory.Key.Name;
            headerRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;
            headerRange.Font.Italic = true;

            startRowIndex++;

            foreach (var payment in groupCategory)
            {
                worksheet.Cells[1][startRowIndex] = payment.Date.ToString("dd.MM.yyyy HH:mm");
                worksheet.Cells[2][startRowIndex] = payment.Name;
                worksheet.Cells[3][startRowIndex] = payment.Price;
                worksheet.Cells[4][startRowIndex] = payment.Num;

                worksheet.Cells[5][startRowIndex].Formula = $"=C{startRowIndex}*D{startRowIndex}";

                worksheet.Cells[3][startRowIndex].NumberFormat =
                    worksheet.Cells[3][startRowIndex].NumberFormat = "#,###.00";

                startRowIndex++;
            }

            Excel.Range sumRange = worksheet.Range[worksheet.Cells[1][startRowIndex], worksheet.Cells[4][startRowIndex]];
            sumRange.Merge();
            sumRange.Value = "ИТОГО:";
            sumRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignRight;
            worksheet.Cells[5][startRowIndex].Formula = $"=SUM(E{startRowIndex - groupCategory.Count()}:" +
                $"E{startRowIndex - 1})";

            sumRange.Font.Bold = worksheet.Cells[5][startRowIndex].Font.Bold = true;
            worksheet.Cells[5][startRowIndex].NumberFormat = "#,###.00";

            startRowIndex++;

            Excel.Range rangeBorders = worksheet.Range[worksheet.Cells[1][1], worksheet.Cells[5][startRowIndex - 1]];
            rangeBorders.Borders[Excel.XlBordersIndex.xlEdgeBottom].LineStyle =
            rangeBorders.Borders[Excel.XlBordersIndex.xlEdgeLeft].LineStyle =
            rangeBorders.Borders[Excel.XlBordersIndex.xlEdgeRight].LineStyle =
            rangeBorders.Borders[Excel.XlBordersIndex.xlEdgeTop].LineStyle =
            rangeBorders.Borders[Excel.XlBordersIndex.xlInsideHorizontal].LineStyle =
            rangeBorders.Borders[Excel.XlBordersIndex.xlInsideVertical].LineStyle = Excel.XlLineStyle.xlContinuous;

            worksheet.Columns.AutoFit();
        }
    }
    application.Visible = true;
}
```

Создание диаграмм различных видов

Создадим приложение, которое позволит построить диаграммы различных типов для визуализации расходов пользователей по категориям.

Для работы с диаграммами необходимо подключить библиотеку

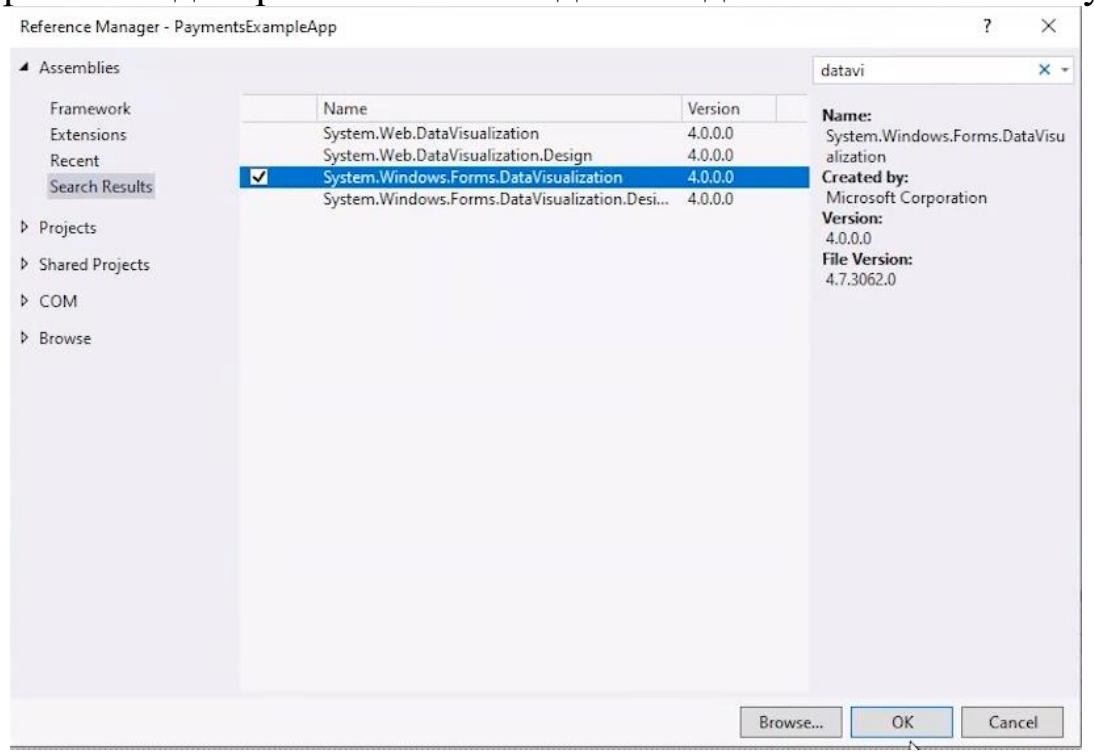


Рис.8 Подключение библиотеки для создания графиков

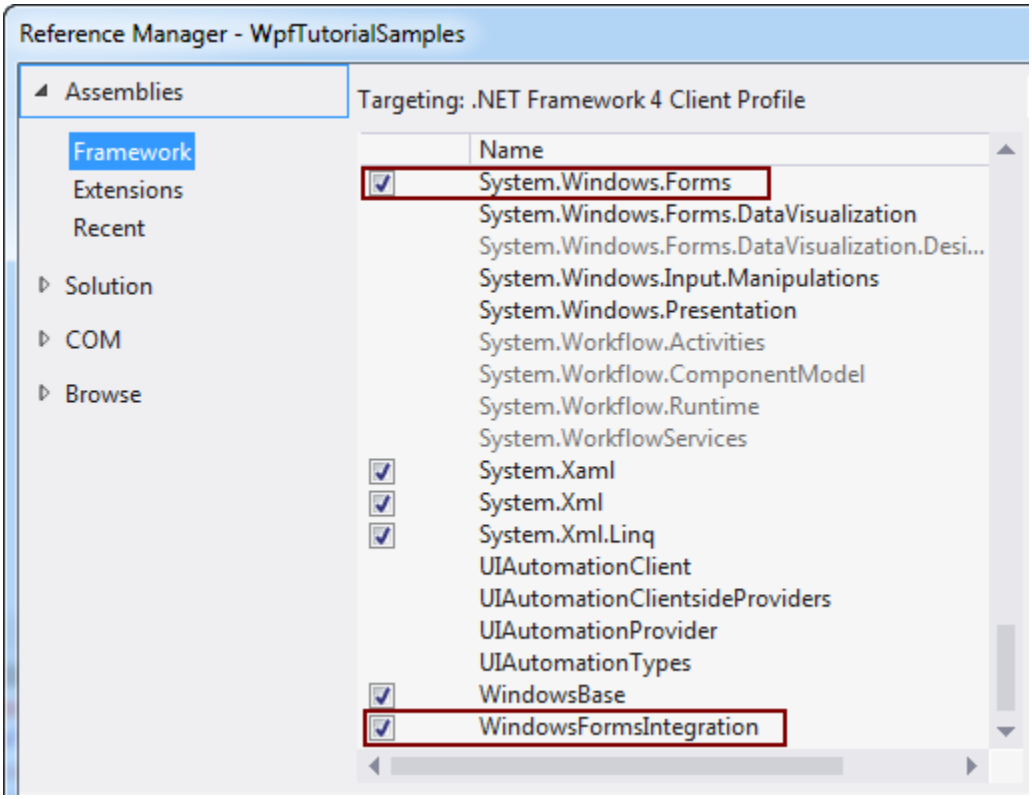


Рис.9 Подключение библиотеки для использования компонента <WindowsFormsHost>

Особое внимание следует уделить здесь строке кода, где мы добавляем пространство имен WinForms в данном окне, чтобы мы имели возможность ссылаться на нужные элементы:

```
xmlns:wf="clr-namespace:System.Windows.Forms;assembly=System.Windows.Forms"
xmlns:charts="clr-
namespace:System.Windows.Forms.DataVisualization.Charting;assembly=System.Windows.Forms.D
ataVisualization"
```

Создадим интерфейс, содержащий выпадающие списки, которые позволяют выбрать пользователя и тип диаграммы

```
x:Class="PaymentsExampleApp.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:PaymentsExampleApp"
xmlns:charts="clr-namespace:System.Windows.Forms.DataVisualization.Charting;assembly=System.Windows.Forms.DataVisualization"
mc:Ignorable="d"
Title="MainWindow" Height="450" Width="800">
 <Grid>
     <Grid.RowDefinitions>
         <RowDefinition Height="auto"></RowDefinition>
         <RowDefinition Height="*"></RowDefinition>
     </Grid.RowDefinitions>
<StackPanel Orientation="Vertical" HorizontalAlignment="Center">
   <StackPanel Orientation="Horizontal">
      <TextBlock Text="Пользователь:" Width="125" Margin="5" VerticalAlignment="Center"
           TextAlignment="Right"></TextBlock>
      <ComboBox Name="ComboUsers" SelectionChanged="UpdateChart" SelectedIndex="0" Width="175" Margin="5" DisplayMemberPath="FIO"></ComboBox>
      <TextBlock Text="Тип диаграммы:" Width="125" Margin="5" VerticalAlignment="Center"
           TextAlignment="Right"></TextBlock>
      <ComboBox Name="ComboChartTypes" SelectionChanged="UpdateChart" SelectedIndex="0" Width="175" Margin="5" ></ComboBox>
   </StackPanel>
   <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
      <Button Content="Экспорт в Excel" VerticalAlignment="Center"
           Width="175" Margin="5" Name="BtnExportToExcel" Click="BtnExportToExcel_Click"></Button>
   </StackPanel>
</StackPanel>
     <WindowsFormsHost Grid.Row="1" Margin="5">
         <charts:Chart x:Name="ChartPayments">
             <charts:Chart.Legends>
                 <charts:Legend>

                 </charts:Legend>
             </charts:Chart.Legends>
         </charts:Chart>
     </WindowsFormsHost>
 </Grid>
```

Выполним построение графиков

ChartAreas создает область построения диаграммы.

Для каждого набора данных (например, для вывода столбца на графике) необходимо добавлять коллекцию Series.

Для нашего задания одна серия данных будет отображать сумму платежей заданного пользователя по категориям. Объект Series создается с указанием названия

В выпадающий список ComboUsers добавим данные о пользователях (из таблицы Users).

В выпадающий список ComboChartTypes загрузим типы диаграмм (из перечисления SeriesChartType)

При выборе значений из выпадающих списков будет вызываться метод UpdateChart()

```
0 references
public MainWindow()
{
    InitializeComponent();
    ChartPayments.ChartAreas.Add(new ChartArea("Main"));

    var currentSeries = new Series("Payments")
    {
        IsValueShownAsLabel = true
    };
    ChartPayments.Series.Add(currentSeries);

    ComboUsers.ItemsSource = _context.Users.ToList();
    ComboChartTypes.ItemsSource = Enum.GetValues(typeof(SeriesChartType));
}
```

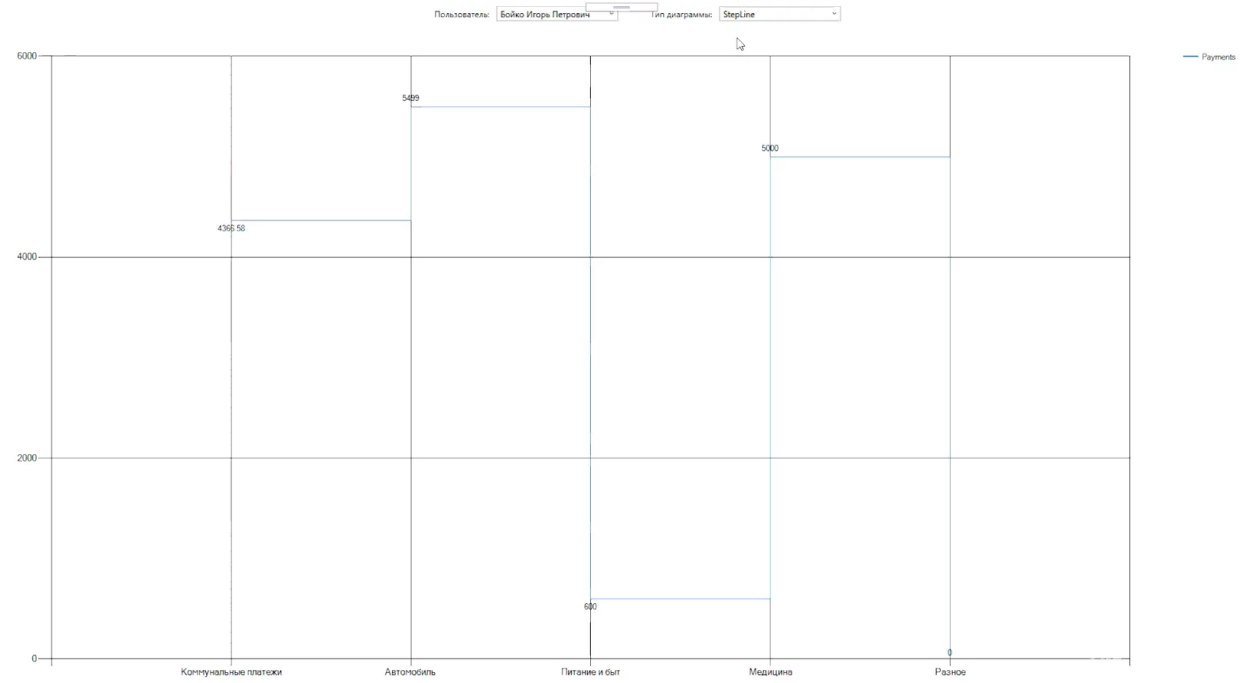
```
0 references
private void UpdateChart(object sender, SelectionChangedEventArgs e)
{
    if (ComboUsers.SelectedItem is User currentUser &&
            ComboChartTypes.SelectedItem is SeriesChartType currentType)
    {
        Series currentSeries = ChartPayments.Series.FirstOrDefault();
        currentSeries.ChartType = currentType;
        currentSeries.Points.Clear();

        var categoriesList = _context.Categories.ToList();
        foreach (var category in categoriesList)
        {
            currentSeries.Points.AddXY(category.Name,
                _context.Payments.ToList().Where(p => p.User == currentUser
                && p.Category == category).Sum(p => p.Price * p.Num));
        }
    }
}
```

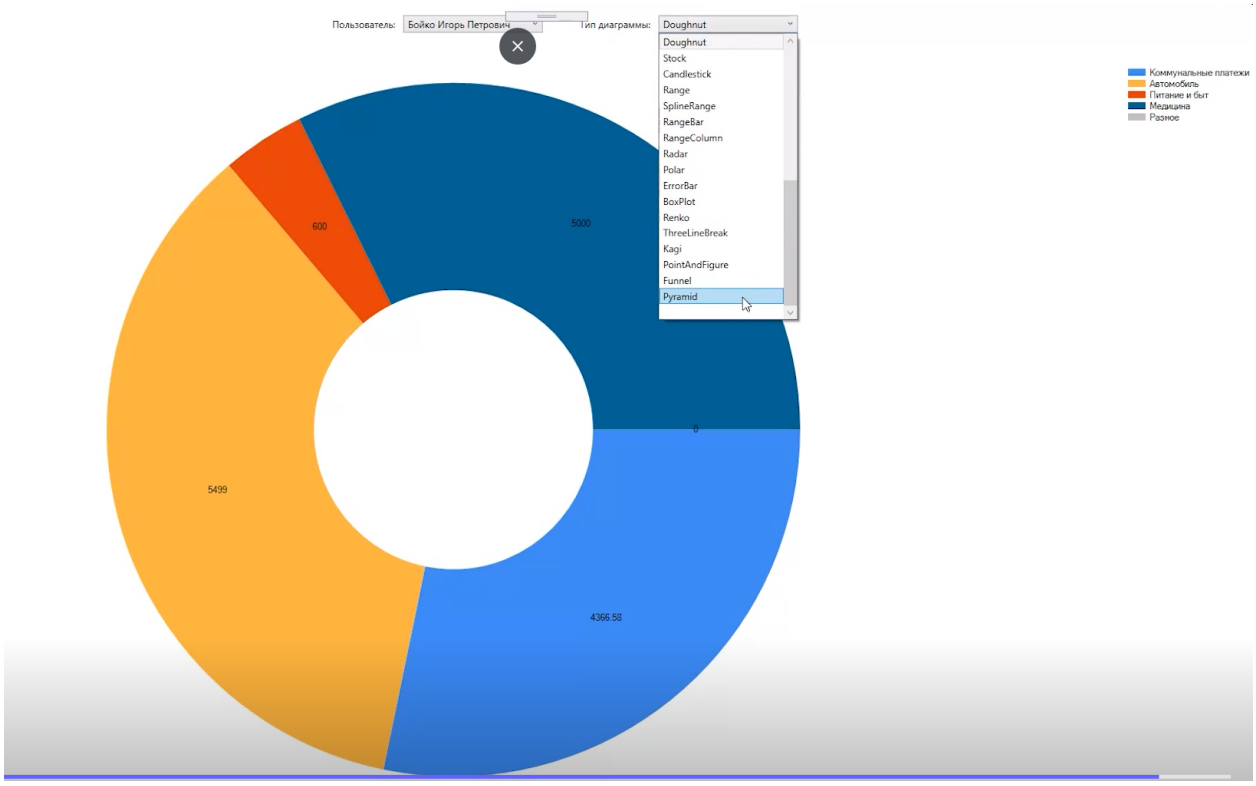
Приведенный код описывает получение серии данных диаграммы из соответствующей коллекции Series, установку типа диаграммы и очистку предыдущих данных.

Список категорий выдается из базы данных.

В цикле foreach для каждой категории значение точки диаграммы добавляется в Points.

Координата X будет названием категории, а координата Y будет суммой платежа для выбранного пользователя в текущей категории


Пользователь:
Бойко Игорь Петрович
Тип диаграммы:
StepLine
Payments
6000
4000
2000
0
5499
5000
4366.58
600
0
Коммунальные платежи
Автомобиль
Питание и быт
Медицина
Разное


Пользователь:
Бойко Игорь Петрович
Тип диаграммы:
Doughnut
Doughnut
Stock
Candlestick
Range
SplineRange
RangeBar
RangeColumn
Radar
Polar
ErrorBar
BoxPlot
Renko
ThreeLineBreak
Kagi
PointAndFigure
Funnel
Pyramid
Коммунальные платежи
Автомобиль
Питание и быт
Медицина
Разное
600
5000
0
5499
4366.58