1. **Назначение и условия применения программы**

Программа предназначена для формирования семейного бюджета, учётом ведения расходов.

|  |  |
| --- | --- |
| Процессор | [Intel Core i5-6400, CPU](https://www.citilink.ru/product/processor-intel-s-core-i5-12400f-soc-1700-2-5ghz-oem-1782240/) |
| Оперативная Память | 2 ГБ |
| Разрешение экрана | 1920x1080 |
| Размер экрана | От 15 дюймов |
| Устройства ввода | Мышь, клавиатура |
| Дисковое пространство | 0.5Гб |
| Подключаемые ус-ва | Принтер |
| Операционная система | Windows 10 |

Требования к программным средствам:

Для корректной работы программы необходимо: ОС Windows 10, SQL Server, Microsoft Excel 2010 (и выше).

1. **Характеристика программы**

Программа должна обеспечивать следующие функции:

* Добавление, удаление и редактирование данных о расходах и категорий
* Поиск по комментарию и по названию категории
* Сортировка расходов: «От Мин. до Макс.» и «От Макс. до Мин.»
* Фильтр по категории расходов
* Экспорт отчета о расходах в Excel

1. **Обращение к программе**

С# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270

Код страницы «Расходы»:

public partial class PageExpenses : Page

{

public PageExpenses()

{

InitializeComponent();

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.ToList();

var listcat = AccountingEntities.GetContext().CostCategory.Select(x => x.Name).Distinct().ToList();

CMBFilterForm.Items.Add("Все категории");

foreach (string Categori in listcat)

{

CMBFilterForm.Items.Add(Categori);

}

System.Int32 customerCount = AccountingEntities.GetContext().Expenses.Count();

Txtname.Text = customerCount.ToString();

}

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

{

Classes.ClassFrame.frmObj.Navigate(new PageCategor());

}

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

{

var AftoForRemoving = DtgSQL.SelectedItems.Cast<Expenses>().ToList();

if (MessageBox.Show($"Вы точно хотите удалить следующие {AftoForRemoving.Count()} записи?", "Внимание",

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

{

try

{

AccountingEntities.GetContext().Expenses.RemoveRange(AftoForRemoving);

AccountingEntities.GetContext().SaveChanges();

MessageBox.Show("Данные удалены!");

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.ToList();

}

catch (Exception ex)

{

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

}

}

int count = AccountingEntities.GetContext().Expenses.Count();

Txtname.Text = count.ToString();

}

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

{

Classes.ClassFrame.frmObj.Navigate(new PageAddExpenses((Expenses)DtgSQL.SelectedItem));

}

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

{

Excel.Application excelApp = new Excel.Application();

Excel.Workbook wb = excelApp.Workbooks.Open($"{Directory.GetCurrentDirectory()}\\Shablon.xlsx");

Excel.Worksheet ws = (Excel.Worksheet)wb.Worksheets[1];

ws.Cells[7, 1] = "Статистика";

ws.Cells[8, 1] = "Дата";

DateTime thisDay = DateTime.Today;

ws.Cells[8, 2] = thisDay.ToShortDateString();

int indexRows = 9;

//ячейка

ws.Cells[1][indexRows] = "Дата";

ws.Cells[2][indexRows] = "Категория";

ws.Cells[3][indexRows] = "Цена";

var printItems = DtgSQL.Items;

//цикл по данным из списка для печати

foreach (Expenses item in printItems)

{

ws.Cells[1][indexRows + 1] = item.Date;

ws.Cells[2][indexRows + 1] = item.CostCategory.Name;

ws.Cells[3][indexRows + 1] = item.Price;

indexRows++;

}

int ch = (int)AccountingEntities.GetContext().Expenses.Sum(x => x.Price);

ws.Cells[indexRows + 2, 1] = "Сумма:";

ws.Cells[indexRows + 2, 2] = ch.ToString();

excelApp.Visible = true;

}

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

{

ClassFrame.frmObj.Navigate(new PageAddExpenses(null));

}

private void CMBFilterForm\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string form = CMBFilterForm.SelectedValue.ToString();

if (form != "Все категории")

{

int cod = AccountingEntities.GetContext().CostCategory.First(x => x.Name == form).Id\_category;

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.Where(x => x.Id\_category == cod).ToList();

int count = AccountingEntities.GetContext().Expenses.Where(x => x.Id\_category == cod).Count();

Txtname.Text = count.ToString();

}

else

{

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.ToList();

int count1 = AccountingEntities.GetContext().Expenses.Count();

Txtname.Text = count1.ToString();

}

}

private void RbUp\_Checked(object sender, RoutedEventArgs e)

{

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.OrderBy(x => x.Price).ToList();

int count = AccountingEntities.GetContext().Expenses.OrderBy(x => x.Price).Count();

Txtname.Text = count.ToString();

}

private void RbDown\_Checked(object sender, RoutedEventArgs e)

{

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.OrderByDescending(x => x.Price).ToList();

int count = AccountingEntities.GetContext().Expenses.OrderByDescending(x => x.Price).Count();

Txtname.Text = count.ToString();

}

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

{

DtgSQL.ItemsSource = AccountingEntities.GetContext().Expenses.Where(x => x.Comment.ToLower().Contains(Findcomen.Text.ToLower())).ToList();

int count = AccountingEntities.GetContext().Expenses.Where(x => x.Comment.ToLower().Contains(Findcomen.Text.ToLower())).ToList().Count();

Txtname.Text = count.ToString();

}

# }Входные и выходные данные

Входные данные:

|  |  |
| --- | --- |
| **Поля** | **Тип данных** |
|  |  |
| Название категории | Текстовый |
| Дата | Текстовый |
| Цена | Дата |
| Комментарий | Текстовый |

**Выходная информация:**

Выходной информацией является экспорт таблицы в виде нового Excel-файла.

# Сообщения

Попытка добавления нового расхода с неуказанными данными

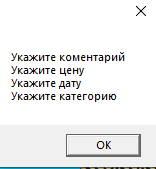


Рис.1 «Попытка добавления расхода»

Подтверждение удаления расхода

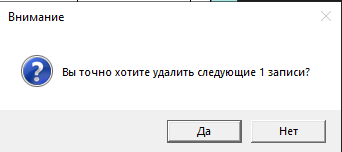


Рис.2 «Подтверждение удаления расхода»

Информация после удаления расхода

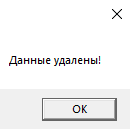


Рис.3 «Информация после удаления»

Информация после изменения расхода

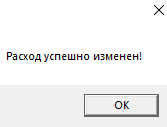


Рис.4 «Информация при изменении расхода»

Информация после добавления расхода

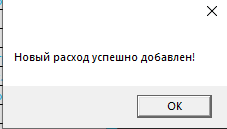


Рис.5 «Информация при добавлении расхода»

Попытка добавить повторяющуюся запись расхода

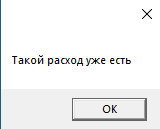


Рис. 6 «Попытка добавить повторяющуюся запись»