## 4 РЕАЛИЗАЦИЯ

**4.1 Разработка классов информационной системы**

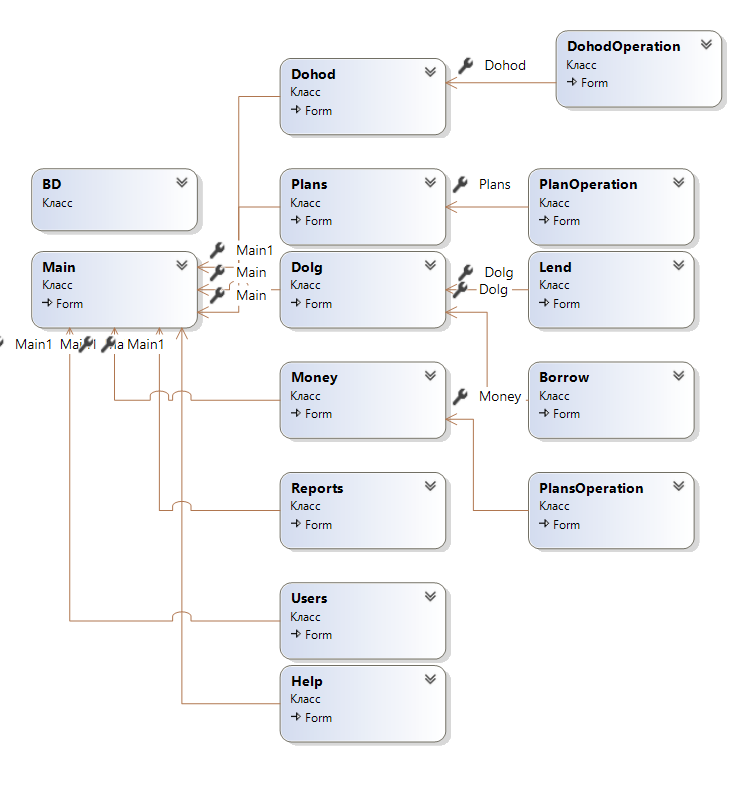


Рисунок 4.1 – Диаграмма классов

Главное окно программы представлено классом Main. Оно содержит экземпляры классов каждого окна для быстрого доступа. При нажатии кнопки расходы, пользователю показывается окно расходов, реализуемого классом Money. При добавлении/изменении какого-либо расхода, открывается окно экземпляра PlanOperation, в котором подробно описана одна из записей расходов.

При выборе планирования, пользователю открывается окно планирования, реализуемое классом Plans. Для добавления/редактирования какого-либо пункта плана, как и с расходами, будет вызвано окно редактирования, реализуемого классом PlansOperation.

Окно пользователя и категорий товаров реализовано классами Users и Help соответственно. Окно отчетов реализовано классомReports.

## 4.2 Разработка интерфейса программного продукта

Главное окно программы представлено на рисунке ниже. Оно в себя включает пункты меню, для доступа к соответствующим формам.

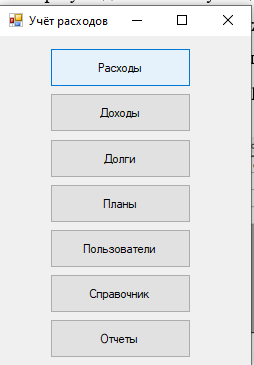


Рисунок 4.2 – Главное окно программы

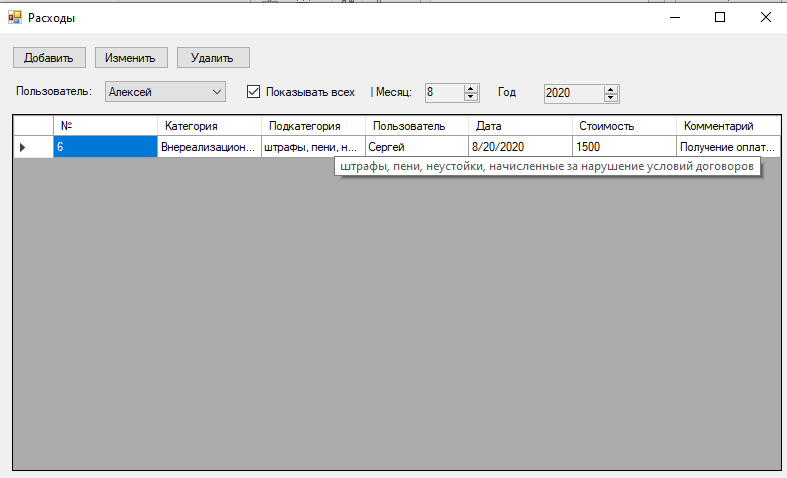
Окно расходов показывает, какой пользователь и на что потратил деньги. Данные в таблице можно сортировать по категории, подкатегорию пользователям и другим столбцам. За разпоказываются расходы пользователя на выбранный месяц. Можно поставить значок на флажок «Показывать всех», так в таблице будут отображены расходы всех пользователей за указанный месяц

Рисунок 4.3 – форма расходов

Для добавления/редактирования расходов используется форма ниже. Она позволяет выбирать категорию и подкатегорию товара, на который будет потрачена та или иная сумма. Также тут выбирается время. Оно автоматически ставит сегодняшнее число, но если что-то забыли написать – можно указать задним числом, изменив дату.

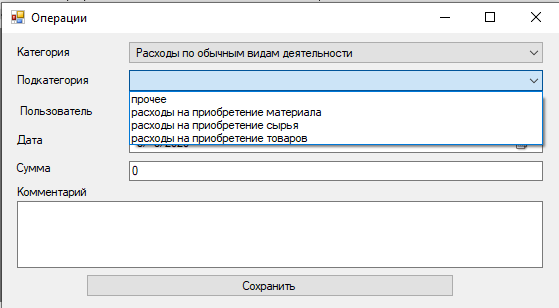


Рисунок 4.4 – операции с расходом

При необходимости удалить ту или иную операцию, достаточно выбрать ее и нажать кнопку удалить. Для убережения пользователей от случайного нажатия, удаление требует подтверждения.

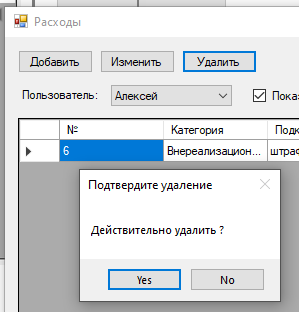


Рисунок 4.5 – подтверждение удаления операции

Ниже показана форма дохода

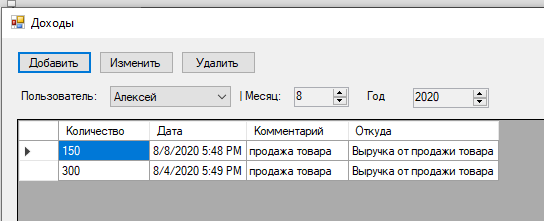


Рисунок 4.6 – доход

Для работы с доходом используется форма ниже

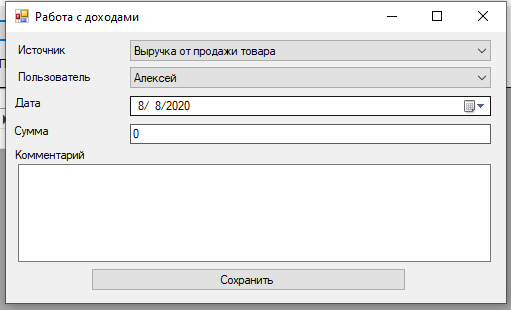


Рисунок 4.7 – форма работы с доходом

Если организация заимствует или дает деньги в долг – предусмотрена форма работы с данными операциями.

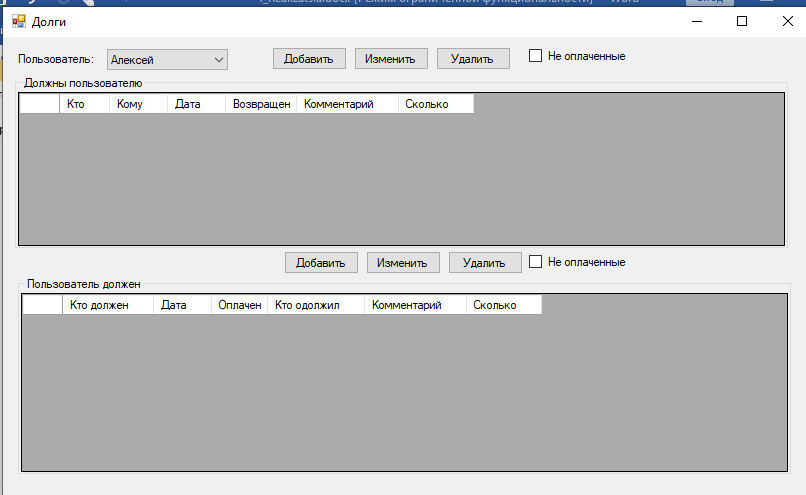


Рисунок 4.8 – долги

Для работы с операциями по долгам используется форма ниже

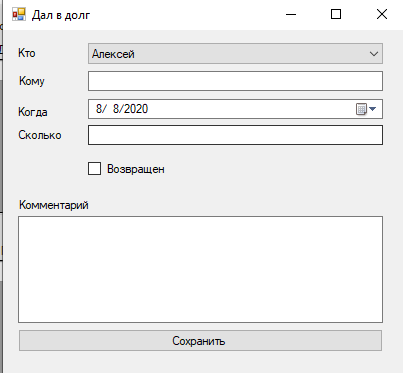


Рисунок 4.9 – Форма работы с долгами

Для предварительного планирования используется форма ниже. На ней можно писать примерно ожидаемые траты.

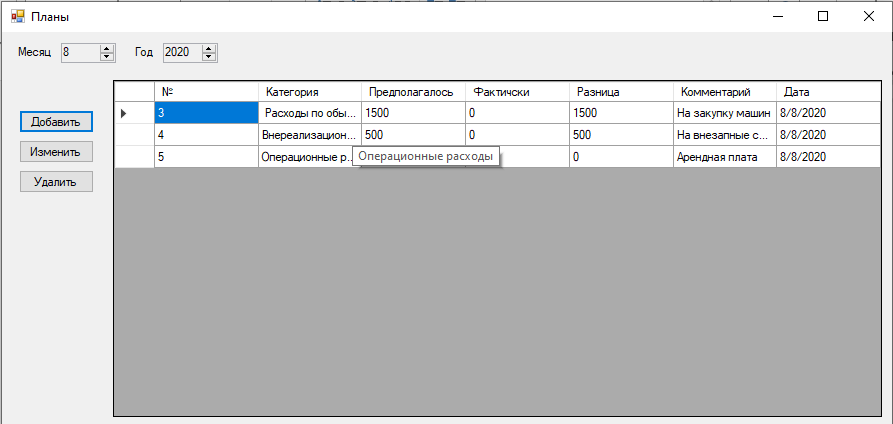


Рисунок 4.10 – планы

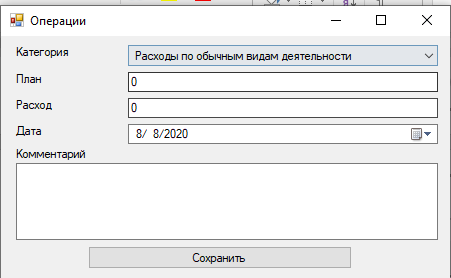


Рисунок 4.11 – Форма работы с планами

Список пользователей представлен на отдельной форме

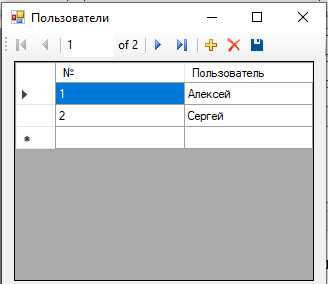


Рисунок 4.12 – список пользователей

На случай необходимости изменения данных в выпадающих списках используется справочник

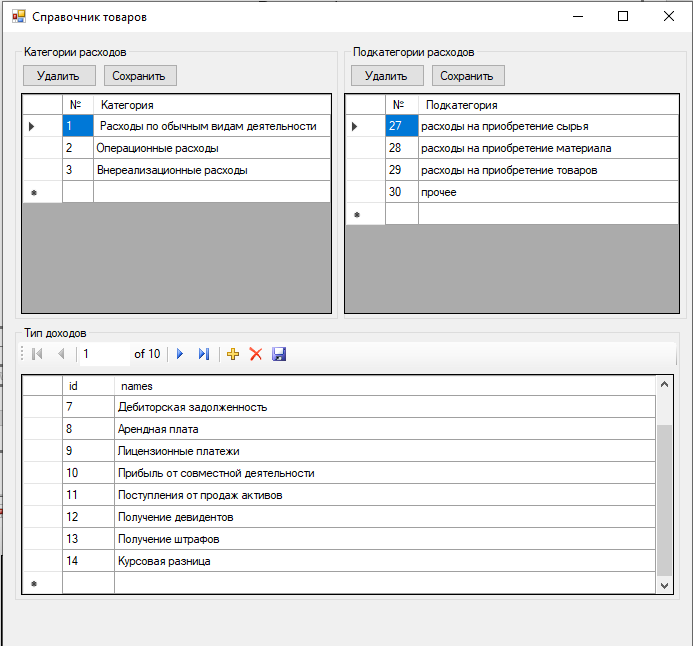


Рисунок 4.13 – справочник

Для создания отчета используется форма отчета. На ней выбирается требуемый тип и период расчета, пользователя, по которому идет учет и идет само его создание

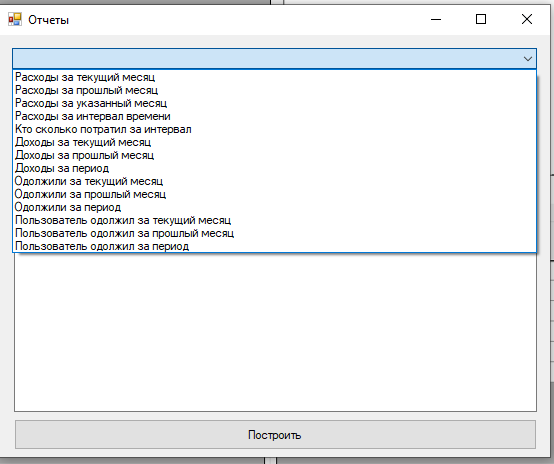


Рисунок 4.14 – отчеты. Список вариантов отчета

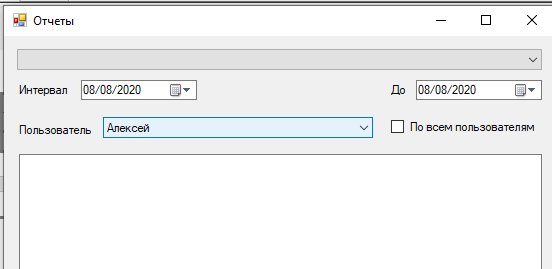


Рисунок 4.15 – отчеты

## 4.3 Разработка алгоритмов реализации вариантов использования.

При решении общей задачи возникали более мелкие, для решения которых необходимо было разрабатывать алгоритмы. Несколько примеров можно увидеть ниже:

1. Вывод данных из базы в dataGridView

public void SelectGridPlus(string com, DataGridView dg)

{

try

{

//Открываем подключение, передаем комманду

connection.Open();

OleDbCommand mes = new OleDbCommand();

mes.Connection = connection;

mes.CommandText = com;

// резервируем место и создаем новый адаптер, в который считываются значения из таблицы в зависимости от команды

OleDbDataAdapter da = new OleDbDataAdapter(mes.CommandText, connection);

// создаем новую таблицу, в которую мы запишем все, что принял адаптер

DataTable dt = new DataTable();

// сейчас записываем это

da.Fill(dt);

// выводим данные из таблиц в грил

dg.DataSource = dt;

connection.Close();

}

catch (Exception ex)

{

MessageBox.Show("FAIL" + ex);

connection.Close();

}

}

1. Запрет ввода букв в поля ввода денег

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!(Char.IsDigit(e.KeyChar)) && !((e.KeyChar == '.') &&

(((TextBox)sender).Text.IndexOf(".") == -1) &&

(((TextBox)sender).Text.Length != 0)))

{

if (e.KeyChar != (char)Keys.Back)

{

e.Handled = true;

}

}

}

1. Получение данных с баззы данных в виде Dictionary

public Dictionary<int, string> getDictionary(string com, int collumnCount)

{

Dictionary<int, string> dict= new Dictionary<int, string>();

try

{

connection.Open();

OleDbCommand command = new OleDbCommand();

command.Connection = connection;

command.CommandText = com;

OleDbDataReader reader = command.ExecuteReader();

while (reader.Read())

{

for (int i = 0; i < collumnCount; i++) {

dict.Add(i, reader[i].ToString());

}

}

reader.Close();

connection.Close();

}

catch (Exception ex)

{

MessageBox.Show("FAIL" + ex);

connection.Close();

}

return dict;

}

1. Формирование отчета за указанный пользователем период по потраченным другими пользователями деньгам

//Кто сколько потратил за интервал

private void report4()

{

string month = "", user = "";

month = $" (DateTime BETWEEN #{dateTimePicker1.Text}# AND #{dateTimePicker2.Text}#) ";

report = "<===== Отчет ====>\r\n" +

"Тип отчета: Кто сколько потратил за интервал\r\n";

Dictionary<string, float> dict = new Dictionary<string, float>();

if (!checkBox1.Checked)

{

user += " AND User=" + comboBox2.SelectedValue;

}

else

report += "По всем пользователям\r\n";

command = $"SELECT Users.name as [Пользователь], Money.DateTime as [Дата], Money.Cost as [Стоимость] FROM Users INNER JOIN((Category INNER JOIN [Money] ON Category.Id = Money.Category) AND([Users].[id] =[Money].[User]) AND ({month} {user}) );";

Console.WriteLine(command);

Main.DataBase.SelectGridPlus(command, dataGridView1);

string categ;

float sum, all = 0;

for (int i = 0; i < dataGridView1.RowCount - 1; i++)

{

categ = dataGridView1[0, i].Value.ToString();

sum = float.Parse(dataGridView1[1, i].Value.ToString());

if (dict.ContainsKey(categ)) dict[categ] = dict[categ] + sum;

else

dict.Add(categ, sum);

}

report += $"Отчет за {currentMonth} {DateTime.Now.Year}\r\n" +

$"Дата/время создания отчета: {DateTime.Now}\r\n\r\n";

foreach (string cat in dict.Keys)

{

report += $"{cat}: {dict[cat]}\r\n";

all += dict[cat];

}

report += "\r\nИТОГО: " + all;

textBox1.Text = report;

}

## Модульное тестирование алгоритмов реализации вариантов использования

Модульное тестирование, или юнит-тестирование - это процесс в программировании, который направлен на проверку отдельных небольших частей приложения, также называемых атомарными, которые можно исследовать изолированно от других подобных частей. При выполнении данного тестирования могут проверяться как отдельные функции или методы классов, так и сами классы, взаимодействие классов, небольшие библиотеки и отдельные части приложения. Довольно часто данный вид тестирования реализуется с использованием специальных технологий и инструментальных средств автоматизации тестирования, значительно упрощающих и ускоряющих разработку соответствующих тест-кейсов [7]. Таким образом, оценивая каждый элемент отдельно и подтверждая правильность его работы, установить проблему значительно проще чем, если бы элемент был частью системы.

Примеры тестирования алгоритмов реализации вариантов использования:

1. Тестирование вывода расходов (таблица 4.1).

Таблица 4.1 - Тестирование вывода расходов

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование вывода расходов | 1. Запустить программу | 1. Нажать на кнопку расходы. 2. Нажать на кнопку Показать всех | 1. После нажатия на кнопку будет показаны расходы текущего пользователя за сегодняшний месяц. После нажатия на кнопку показать всех, будут отображены расходы всех пользователей за текущий месяц |

2. Тестирование добавления расхода(таблица 4.2).

Таблица 4.2 - Тестирование добавления расхода

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование добавления расхода | 1. Запустить программу | 1. Нажать на кнопку расходы 2. Нажать кнопку добавить 3. Заполнить все поля 4. Нажать сохранить | После сохранения, пользователю покажется диалоговое окно с надписью «успешно». После закрытия окна и формы добавления, в списке расходов будет показан добавленный расход |

3. Тестирование изменение расхода (таблица 4.3).

Таблица 4.3 - Тестирование изменения расхода

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование изменение расхода | 1. Запустить программу | 1. Нажать на кнопку расходы 2. Выбрать расход 3. Нажать на кнопку изменить 4. Изменить любое поле 5. Нажать кнопку сохранить | После сохранения, пользователю покажется диалоговое окно с надписью «успешно». После закрытия окна и формы изменения, в списке расходов будет показан измененный расход |