

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Ярославский государственный технический университет»  
Кафедра «Информационные системы и технологии»

Работа защищена  
с оценкой \_\_\_\_\_  
Преподаватель  
\_\_\_\_\_ А.Н. Прозоров  
«\_\_» \_\_\_\_\_ 2025

## **КНИГА УЧЕТА**

Отчет по лабораторной работе №1  
по курсу «Технологии программирования»

ЯГТУ 09.03.02-001 ЛР

Работу выполнил  
студент группы ЗЦИС-26  
\_\_\_\_\_ О.А. Жолудев  
«\_\_» \_\_\_\_\_ 2025

**Цель работы:**

Создание Книги учёта

**Задачи:**

- Разработать приложение, выполняющее следующие требования
- Создание\изменение\удаление категории операций
- Создание\изменение\удаление операции с финансами (доход\расход)
- Подсчет баланса и суммы по категориям
- сохранение в JSON и открытие из него
- (\*) построение графика (Chart)

В соответствии с поставленной целью и задачами был определен алгоритм выполнения данной лабораторной работы, в программной среде Visual Studio была реализована форма Windows Forms, на которой расположили основные элементы, необходимые для нашего проекта

На рисунке 1 представлен вариант будущей формы с некоторыми компонентами проекта.

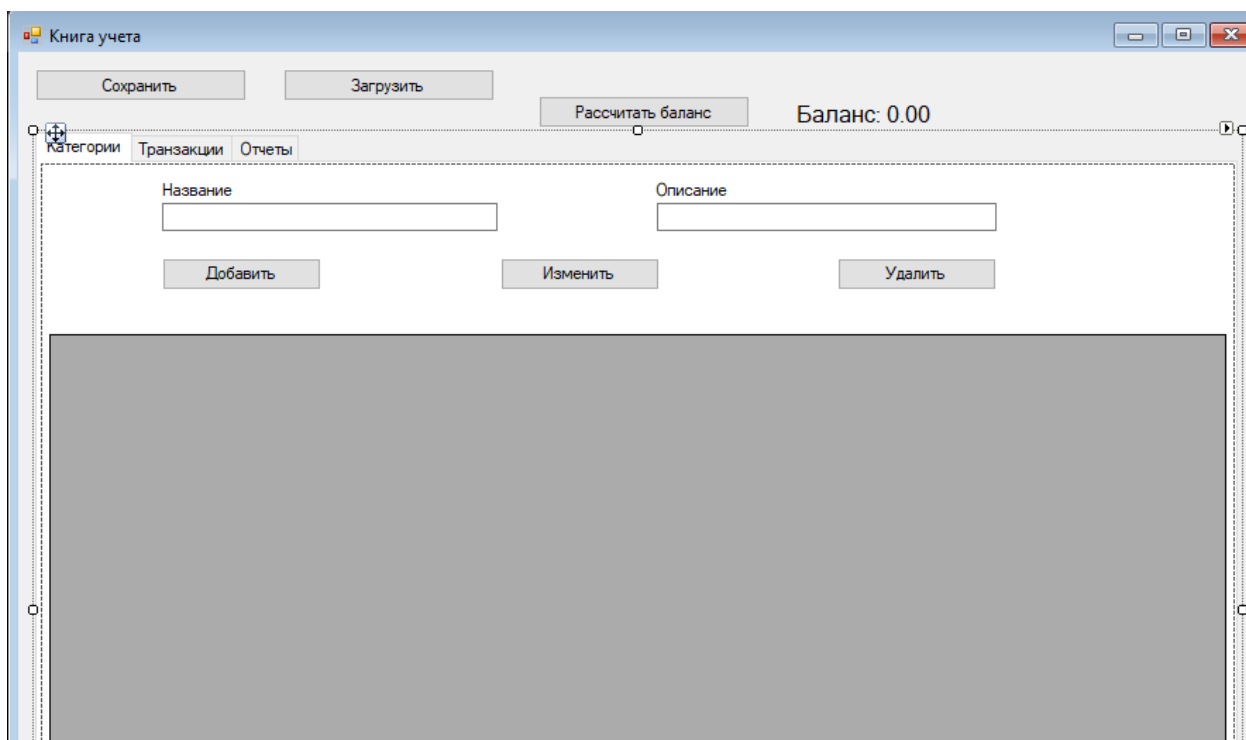


Рисунок 1. Начальный этап создания формы

Архитектура приложения построена на основе трех основных классов: Transaction, Category и AccountingData.

Класс Transaction: Представляет собой запись о финансовой транзакции. Содержит следующие свойства:

Date (DateTime): Дата транзакции (без указания времени).

Description (string): Описание транзакции.

Amount (десятичное число): сумма транзакции (положительная для доходов, отрицательная для расходов).

Category (Категория): объект Category, к которому относится транзакция.

Type (TransactionType): перечисление TransactionType, указывающее тип транзакции (Income или Expense).

Класс Category: представляет собой категорию транзакций. Содержит свойства:

Name (string): Название категории.

Description (строка): Описание категории (необязательно). Метод Equals переопределен для сравнения объектов категорий по свойству Name, что необходимо для корректной работы с коллекциями и ComboBox.

Класс `AccountingData`: Центральный класс, управляющий данными приложения. Содержит:

`Transactions (BindingList)`: коллекция объектов `Transaction`, используемая для привязки к `DataGridView`. `BindingList` обеспечивает автоматическое обновление `DataGridView` при изменении списка.

`Categories (BindingList)`: коллекция объектов `Category`, также используемая для привязки к `DataGridView` и `ComboBox`.

`AddTransaction(Transaction transaction)`: Метод для добавления новой транзакции в список `Transactions`.

`DeleteTransaction(DateTime date, string description, decimal amount)`: Метод для удаления транзакции из списка `Transactions`.

`UpdateTransaction(DateTime date, string description, decimal amount, Transaction newTransaction)`: Метод для обновления существующей транзакции. Для поиска транзакции используется комбинация `Date`, `Description` и `Amount` (ключевые поля).

`AddCategory(Category category)`: Метод для добавления новой категории в список `Categories`.

`DeleteCategory(string name)`: Метод для удаления категории из списка `Categories`. При удалении категории также происходит обновление транзакций, связанных с этой категорией (замена на `null` или категорию по умолчанию).

`UpdateCategory(string oldName, Category newCategory)`: Метод для обновления существующей категории. Также обновляет связанные транзакции.

`CalculateBalance()`: Метод для расчёта текущего баланса на основе всех транзакций.

`CalculateCategoryTotals()`: Метод для расчёта сумм расходов и доходов по каждой категории. Возвращает словарь (`Dictionary`), где ключ — название категории, а значение — общая сумма.

`GetDataForChart()`: Метод, подготавливающий данные для диаграммы. Возвращает словарь, содержащий общие суммы по каждой категории, для отображения структуры расходов/доходов.

Интерфейс пользователя:

Интерфейс приложения реализован с использованием `Windows Forms` и включает в себя следующие элементы:

`DataGridView (dgvTransactions)`: для отображения списка транзакций. Каждый столбец `DataGridView` привязан к соответствующему свойству класса `Transaction` через свойство `DataPropertyName`. Столбцы «Категория» и «Тип» отображаются как текстовые столбцы (`DataGridViewTextBoxColumn`), связанные со свойствами `Category.Name` и `Transaction.Type.ToString()` соответственно.

`DataGridView (dgvCategories)`: для отображения списка категорий. Столбцы привязаны к свойствам `Name` и `Description` класса `Category`.

`DateTimePicker (dtpTransactionDate)`: для выбора даты транзакции.

Текстовое поле (txtTransactionDescription): для ввода описания транзакции.

NumericUpDown (nudTransactionAmount): Для ввода суммы транзакции.

Комбо-поле (cmbTransactionCategory): для выбора категории транзакции. Список категорий привязан к коллекции Categories класса AccountingData. Свойства DisplayMember и ValueMember установлены в "Name" для отображения и выбора названий категорий.

Переключатели (rdbTransactionIncome, rdbTransactionExpense): для выбора типа транзакции (доход или расход).

Кнопки («Добавить», «Изменить», «Удалить» для транзакций и категорий): обработчики событий для этих кнопок выполняют соответствующие действия с данными (добавление, редактирование, удаление объектов Transaction и Category из коллекций Transactions и Categories класса AccountingData). После каждого изменения данных вызывается метод ResetBindings(false) для transactionsBindingSource и categoriesBindingSource, чтобы обновить отображение данных в DataGridView.

Метка (lblBalance): для отображения текущего баланса, который вычисляется методом CalculateBalance() класса AccountingData.

ListBox (lstCategoryTotals): для отображения общей суммы доходов и расходов по каждой категории, полученной с помощью метода CalculateCategoryTotals() класса AccountingData.

Диаграмма (chtCategoryTotals): для графического представления структуры расходов и доходов по категориям. Данные для диаграммы формируются методом GetDataForChart() класса AccountingData и передаются в объект Series диаграммы.

OpenFileDialog (openFileDialog1) и SaveFileDialog (saveFileDialog1): для выбора файла JSON для загрузки и сохранения данных.

Кнопки “Сохранить” и “Загрузить”: обработчики событий для этих кнопок используют библиотеку Newtonsoft.Json для сериализации и десериализации данных в формате JSON и сохранения/загрузки в файл.

Привязка данных:

Для обеспечения двусторонней связи между данными и элементами управления пользовательского интерфейса используются компоненты BindingSource:

categoriesBindingSource: Привязан к коллекции Categories класса AccountingData и DataGridView для отображения категорий.

transactionsBindingSource: Привязан к коллекции Transactions класса AccountingData и DataGridView для отображения транзакций.

Метод ResetBindings(false) вызывается после каждого изменения данных для обновления отображения в DataGridView.

Сохранение и загрузка данных:

Для сохранения и загрузки данных используется библиотека Newtonsoft.Json. Методы SaveDataToFile() и LoadDataFromFile() выполняют сериализацию и десериализацию объекта AccountingData в формате JSON и

сохранение/загрузку в файл, выбранный пользователем в диалоговых окнах OpenFileDialog и SaveFileDialog.

При загрузке данных выполняется проверка наличия категорий, указанных в транзакциях, и удаление транзакций с несуществующими категориями.

#### Обработка ошибок:

Для обработки ошибок используется обработчик событий DataError для DataGridView (dgvTransactions\_DataError). Этот обработчик отображает сообщение об ошибке, если при вводе данных в DataGridView возникает исключение, например при вводе некорректного формата даты или числа.

#### Технологии и инструменты:

- C#
- Формы Windows
- Визуальная студия
- Newtonsoft.Json (для сериализации/десериализации JSON)
- Git (для контроля версий)

Конечные наглядные представления Windows Forms, созданной для проекта «Книга учета», изложены в рисунках ниже. На рисунках 2 и 3 представлены варианты формы со всеми необходимыми рабочими элементами и графиком для наглядного представления информации, согласно требованиям указанным к заданию.

The screenshot shows a Windows Forms application titled "Книга учета". The interface includes a menu bar with "Категории", "Транзакции", and "Отчеты". Below the menu bar, there are buttons for "Сохранить", "Загрузить", and "Рассчитать баланс", along with a balance display showing "Баланс: 0.00". The main form area contains a date picker set to "21 мая 2025 г.", a text field for "Описание", a numeric field for "0.00", and radio buttons for "Доход" (selected) and "Расход". At the bottom of the form are buttons for "Добавить", "Изменить", and "Удалить". A large gray rectangle at the bottom of the window represents a data grid.

Рисунок 2. Наглядное представление необходимых компонентов формы

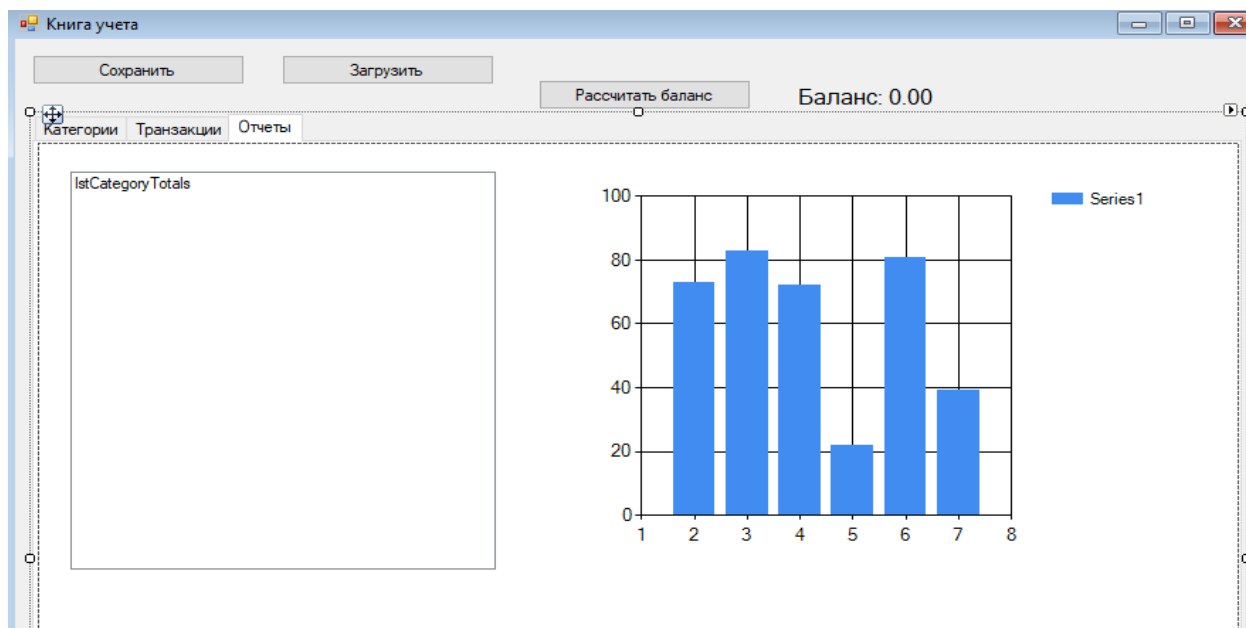


Рисунок 3. Наглядное представление необходимых компонентов формы

На рисунках 4,5,6 , представленных ниже изображена работоспособная версия нашего проекта с вводом некоторой информации и наглядным представлением вводимой пользователем информации в виде графика.

Name	Description
иваиави	чс спавафф
ьроен	ымфвья
*	

Рисунок 4. Форма с вводом некоторой информации

Книга учета

Сохранить Загрузить Рассчитать баланс Баланс: 25 660,00 Р

Категории Транзакции Отчеты

21 мая 2025 г. Описание ьроен

0,00 Доход Расход

Добавить Изменить Удалить

	Дата	Описание	Сумма	Категория	Тип
▶	2025-05-21	фавымся сав	25580,00	иваизави	Income
	2025-05-21	препнавп	80,00	ьроен	Income
*					

Рисунок 5. Форма с вводом некоторой информации

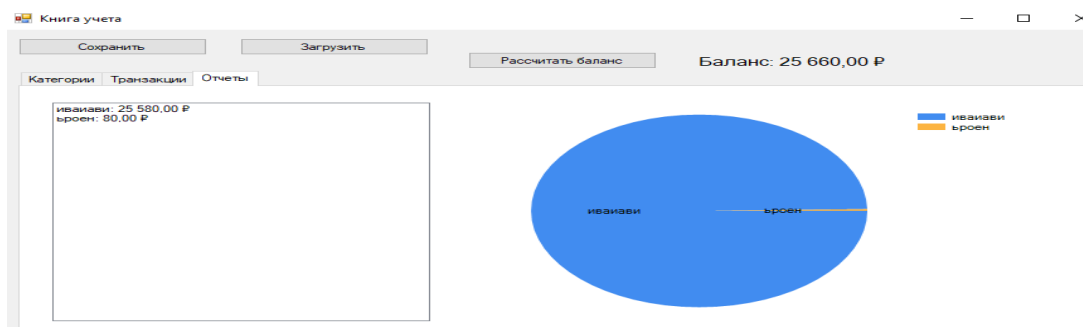


Рисунок 6. Графическое отображение вводимой информации.



**Вывод:**

В ходе выполнения лабораторной работы было разработано приложение Windows Forms на C# для ведения учета личных финансов, успешно реализующее добавление, редактирование и удаление транзакций с классификацией по категориям, расчетом текущего баланса и отображением структуры расходов, а также обеспечивающее сохранение и загрузку данных из JSON-файла; приобретены навыки работы с Windows Forms, привязкой данных, библиотекой Newtonsoft.Json и системой контроля версий Git, что позволяет сделать вывод о достижении поставленной цели — создании работающего приложения, демонстрирующего понимание принципов разработки приложений с графическим интерфейсом и управления данными.

## Содержимое файла Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using Книга_учета;
using Newtonsoft.Json;
using System.IO;
using System.Windows.Forms.DataVisualization.Charting;

namespace Книга_учета
{
    public partial class Form1 : Form
    {
        private AccountingData accountingData = new AccountingData();
        private BindingSource categoriesBindingSource = new BindingSource();
        private BindingSource transactionsBindingSource = new BindingSource();

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Инициализация BindingSource и DataGridView
            categoriesBindingSource.DataSource = accountingData.Categories;
            dgvCategories.DataSource = categoriesBindingSource;
            dgvCategories.AutoGenerateColumns = false;

            // Запрещаем редактирование ячеек напрямую
            foreach (DataGridViewColumn column in dgvCategories.Columns)
            {
                column.ReadOnly = true;
            }
            dgvCategories.AllowUserToAddRows = false; //Отключаем добавление
            dgvCategories.AllowUserToDeleteRows = false; //Отключаем удаление
        }
    }
}
```

```

transactionsBindingSource.DataSource = accountingData.Transactions;
dgvTransactions.DataSource = transactionsBindingSource;
dgvTransactions.AutoGenerateColumns = false;
dgvTransactions.DataError += dgvTransactions_DataError;
dgvTransactions.EditingControlShowing                                     +=
dgvTransactions_EditingControlShowing; // Подписываемся на событие

// Настройка колонок для DataGridView (транзакции)
dgvTransactions.Columns.Clear();
DataGridViewTextBoxColumn dateColumn = new
DataGridViewTextBoxColumn { DataPropertyName = "Date", HeaderText =
"Дата", Name = "Date" };
dgvTransactions.Columns.Add(dateColumn);
if (dgvTransactions.Columns.Contains("Date"))
{
    dgvTransactions.Columns["Date"].DefaultCellStyle.Format = "yyyy-
MM-dd";
}

DataGridViewTextBoxColumn descriptionColumn = new
DataGridViewTextBoxColumn { DataPropertyName = "Description", HeaderText
= "Описание", Name = "Description" };
dgvTransactions.Columns.Add(descriptionColumn);

DataGridViewTextBoxColumn amountColumn = new
DataGridViewTextBoxColumn { DataPropertyName = "Amount", HeaderText =
"Сумма", Name = "Amount" };
dgvTransactions.Columns.Add(amountColumn);

DataGridViewTextBoxColumn categoryColumn = new
DataGridViewTextBoxColumn
{
    DataPropertyName = "Category",
    HeaderText = "Категория", // Заголовок на русском
    Name = "Category"
};
dgvTransactions.Columns.Add(categoryColumn);

DataGridViewTextBoxColumn typeColumn = new
DataGridViewTextBoxColumn
{
    DataPropertyName = "Type",
    HeaderText = "Тип", // Заголовок на русском
    Name = "Type"
};

```

```

dgvTransactions.Columns.Add(typeColumn);

// Запрещаем редактирование ячеек напрямую
foreach (DataGridViewColumn column in dgvTransactions.Columns)
{
    column.ReadOnly = true;
}

dgvTransactions.AllowUserToAddRows = false; //Отключаем
добавление строк
dgvTransactions.AllowUserToDeleteRows = false; //Отключаем удаление
строк

openFileDialog1.Filter = "JSON files (*.json)|*.json|All files (*.*)|*.*";
saveFileDialog1.Filter = "JSON files (*.json)|*.json|All files (*.*)|*.*";

// LoadDataFromFile(); // Используем LoadDataFromFile при загрузке
формы - УДАЛЕНО!!!
UpdateBalance();
UpdateCategoryTotals();
UpdateChart();
}

private void dgvTransactions_DataError(object sender,
DataGridViewDataErrorEventArgs e)
{
    string columnName = "";
    if (dgvTransactions.Columns.Count > 0 && e.ColumnIndex >= 0 &&
e.ColumnIndex < dgvTransactions.Columns.Count)
    {
        columnName = dgvTransactions.Columns[e.ColumnIndex].Name; //
Получаем имя колонки
    }
    if (e.Exception is FormatException)
    {
        MessageBox.Show($"Некорректный формат данных в колонке
'{columnName}'. Пожалуйста, выберите допустимое значение из списка.",
"Ошибка формата", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        e.ThrowException = false;
        e.Cancel = true;
    }
    else
    {

```

```

        MessageBox.Show($"Произошла ошибка в колонке '{columnName}':
{e.Exception.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        e.ThrowException = false;
    }
}

```

```

private void dgvTransactions_EditingControlShowing(object sender,
DataGridViewEditingControlShowingEventArgs e)
{
    //Больше не нужно
}

```

```

private void LoadDataFromFile()
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string filePath = openFileDialog1.FileName;
        try
        {
            string jsonData = File.ReadAllText(filePath);
            accountingData =
JsonConvert.DeserializeObject<AccountingData>(jsonData);

```

```

        // Проверка категорий в транзакциях
        foreach (var transaction in accountingData.Transactions.ToList()) //
        ToList() чтобы избежать изменения коллекции во время итерации
        {
            if (transaction.Category != null &&
!accountingData.Categories.Any(c => c.Name == transaction.Category.Name))
//Проверяем, есть ли категория с таким именем
            {
                // Категория не найдена
                MessageBox.Show($"Категория '{transaction.Category.Name}'
не найдена. Транзакция будет удалена.", "Предупреждение",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                accountingData.Transactions.Remove(transaction); // Удаляем
транзакцию
                //transaction.Category =
accountingData.Categories.FirstOrDefault(); // Или заменяем на категорию по
умолчанию
            }
        }
    }
}

```

```

        // Обновление BindingSource и UI
        categoriesBindingSource.DataSource = accountingData.Categories;
        transactionsBindingSource.DataSource = accountingData.Transactions;

        categoriesBindingSource.ResetBindings(false);
        transactionsBindingSource.ResetBindings(false);
        UpdateCategoryComboBox();
        UpdateBalance();
        UpdateCategoryTotals();
        UpdateChart();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки данных: {ex.Message}",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void SaveDataToFile()
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        string filePath = saveFileDialog1.FileName;
        try
        {
            string jsonData = JsonConvert.SerializeObject(accountingData,
                Newtonsoft.Json.Formatting.Indented);
            File.WriteAllText(filePath, jsonData);
            MessageBox.Show("Данные успешно сохранены!", "Успех",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка сохранения данных: {ex.Message}",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

// Методы для категорий
private void btnAddCategory_Click(object sender, EventArgs e)
{
    string name = txtNameCategory.Text.Trim();

```

```

string description = txtDescriptionCategory.Text.Trim();

if (!string.IsNullOrEmpty(name))
{
    accountingData.AddCategory(new Category(name, description));
    categoriesBindingSource.ResetBindings(false);
    UpdateCategoryComboBox();
    txtNameCategory.Clear();
    txtDescriptionCategory.Clear();
    UpdateCategoryTotals();
    UpdateChart();
}
else
{
    MessageBox.Show("Введите название категории.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}

private void btnEditCategory_Click(object sender, EventArgs e)
{
    if (dgvCategories.SelectedRows.Count > 0)
    {
        Category selectedCategory =
(Category)dgvCategories.SelectedRows[0].DataBoundItem;
        if (selectedCategory != null)
        {
            string oldName = selectedCategory.Name;
            string newName = txtNameCategory.Text.Trim();
            string newDescription = txtDescriptionCategory.Text.Trim();

            if (!string.IsNullOrEmpty(newName))
            {
                Category newCategory = new Category(newName,
newDescription);
                accountingData.UpdateCategory(oldName, newCategory);
                categoriesBindingSource.ResetBindings(false);
                transactionsBindingSource.ResetBindings(false);
                UpdateCategoryComboBox();
                UpdateCategoryTotals();
                UpdateChart();
                txtNameCategory.Clear();
                txtDescriptionCategory.Clear();
            }
            else

```

```

        {
            MessageBox.Show("Введите новое название категории.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
}
else
{
    MessageBox.Show("Выберите категорию для редактирования.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}

private void btnDeleteCategory_Click(object sender, EventArgs e)
{
    if (dgvCategories.SelectedRows.Count > 0)
    {
        Category selectedCategory =
(Category)dgvCategories.SelectedRows[0].DataBoundItem;
        if (selectedCategory != null)
        {
            DialogResult result = MessageBox.Show($"Вы уверены, что хотите
удалить категорию '{selectedCategory.Name}'?", "Подтверждение удаления",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {
                accountingData.DeleteCategory(selectedCategory.Name);
                categoriesBindingSource.ResetBindings(false);
                transactionsBindingSource.ResetBindings(false);
                UpdateCategoryComboBox();
                UpdateCategoryTotals();
                UpdateChart();
            }
        }
    }
    else
    {
        MessageBox.Show("Выберите категорию для удаления.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

private void UpdateCategoryComboBox()
{
    cmbTransactionCategory.DataSource = null;

```



```

cmbTransactionCategory.DataSource = accountingData.Categories;
cmbTransactionCategory.DisplayMember = "Name";
cmbTransactionCategory.ValueMember = "Name";
}

// Методы для транзакций
private void btnAddTransaction_Click(object sender, EventArgs e)
{
    if (cmbTransactionCategory.SelectedItem == null)
    {
        MessageBox.Show("Пожалуйста, выберите категорию.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    Category selectedCategory =
(Category)cmbTransactionCategory.SelectedItem;

    // Проверяем, существует ли выбранная категория в списке категорий
    if (!accountingData.Categories.Contains(selectedCategory))
    {
        MessageBox.Show("Выбранная категория не существует.
Пожалуйста, выберите другую категорию.", "Предупреждение",
MessageButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    DateTime date = dtpTransactionDate.Value.Date; // Сохраняем только
дату
    string description = txtTransactionDescription.Text.Trim();
    decimal amount = nudTransactionAmount.Value;

    TransactionType type = rdbTransactionExpense.Checked ?
TransactionType.Expense : TransactionType.Income; // Получаем тип из
RadioButton

    if (!string.IsNullOrEmpty(description))
    {
        Transaction transaction = new Transaction(date, description, amount,
selectedCategory, type);

        accountingData.AddTransaction(transaction);
        transactionsBindingSource.ResetBindings(false);
        UpdateBalance();
        UpdateCategoryTotals();
    }
}

```

```

        UpdateChart();
        ClearTransactionFields();
    }
    else
    {
        MessageBox.Show("Пожалуйста,      введите      описание.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}

private void btnEditTransaction_Click(object sender, EventArgs e)
{
    if (dgvTransactions.SelectedRows.Count > 0)
    {
        Transaction          selectedTransaction          =
(Transaction)dgvTransactions.SelectedRows[0].DataBoundItem;
        if (selectedTransaction != null)
        {
            DateTime newDate = dtpTransactionDate.Value.Date; // Сохраняем
только дату
            string newDescription = txtTransactionDescription.Text.Trim();
            decimal amount = nudTransactionAmount.Value;

            TransactionType  type  =  rdbTransactionExpense.Checked  ?
TransactionType.Expense  :  TransactionType.Income; // Получаем тип из
RadioButton

            Category          selectedCategory          =
(Category)cmbTransactionCategory.SelectedItem;

            if      (selectedCategory      !=      null      &&
!string.IsNullOrEmpty(newDescription))
            {
                selectedTransaction.Date = newDate;
                selectedTransaction.Description = newDescription;
                selectedTransaction.Amount = amount;
                selectedTransaction.Category = selectedCategory;
                selectedTransaction.Type = type;

                dgvTransactions.Refresh();
                transactionsBindingSource.ResetBindings(false);

                UpdateBalance();
                UpdateCategoryTotals();
                UpdateChart();
            }
        }
    }
}

```

```

        ClearTransactionFields();
    }
    else
    {
        MessageBox.Show("Пожалуйста, выберите категорию и введите
описание.", "Предупреждение", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
    }
}
}
else
{
    MessageBox.Show("Выберите транзакцию для редактирования.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
}

private void btnDeleteTransaction_Click(object sender, EventArgs e)
{
    if (dgvTransactions.SelectedRows.Count > 0)
    {
        Transaction selectedTransaction =
(Transaction)dgvTransactions.SelectedRows[0].DataBoundItem;
        if (selectedTransaction != null)
        {
            DialogResult result = MessageBox.Show($"Вы уверены, что хотите
удалить транзакцию '{selectedTransaction.Description}'?", "Подтверждение
удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (result == DialogResult.Yes)
            {
                accountingData.DeleteTransaction(selectedTransaction.Date,
selectedTransaction.Description, selectedTransaction.Amount);
                transactionsBindingSource.ResetBindings(false);
                UpdateBalance();
                UpdateCategoryTotals();
                UpdateChart();
            }
        }
    }
    else
    {
        MessageBox.Show("Выберите транзакцию для удаления.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
}
}

```

```

private void ClearTransactionFields()
{
    txtTransactionDescription.Clear();
    nudTransactionAmount.Value = 0;
    rdbTransactionIncome.Checked = true;
}

private void UpdateBalance()
{
    decimal balance = accountingData.CalculateBalance();
    lblBalance.Text = $"Баланс: {balance:C}";
}

private void UpdateCategoryTotals()
{
    Dictionary<string, decimal> categoryTotals =
accountingData.CalculateCategoryTotals();

    // ВЫВОД в ListBox
    lstCategoryTotals.Items.Clear();
    foreach (var kvp in categoryTotals)
    {
        lstCategoryTotals.Items.Add($" {kvp.Key}: {kvp.Value:C}");
    }
    UpdateChart();
}

private void UpdateChart()
{
    chtCategoryTotals.Series.Clear();

    Series series = new Series("Category Totals");
    series.ChartType = SeriesChartType.Pie;

    Dictionary<string, decimal> chartData =
accountingData.GetDataForChart();

    foreach (var kvp in chartData)
    {
        series.Points.AddXY(kvp.Key, kvp.Value);
    }

    chtCategoryTotals.Series.Add(series);
}

```

```

private void btnSaveTransactions_Click(object sender, EventArgs e)
{
    SaveDataToFile();
}

private void btnLoadTransactions_Click(object sender, EventArgs e)
{
    LoadDataFromFile();
}

// Обработчики событий для текстовых полей (можно добавлять
// валидацию)
private void txtNameCategory_TextChanged(object sender, EventArgs e) { }
private void txtDescriptionCategory_TextChanged(object sender, EventArgs
e) { }
private void dtpTransactionDate_ValueChanged(object sender, EventArgs e)
{ }
private void cmbTransactionCategory_SelectedIndexChanged(object sender,
EventArgs e) { }
private void txtTransactionDescription_TextChanged(object sender,
EventArgs e) { }
private void nudTransactionAmount_ValueChanged(object sender,
EventArgs e) { }
private void rdbTransactionIncome_CheckedChanged(object sender,
EventArgs e) { }
private void rdbTransactionExpense_CheckedChanged(object sender,
EventArgs e) { }

private void btnCalculateBalance_Click(object sender, EventArgs e)
{
    UpdateBalance();
    UpdateCategoryTotals();
    UpdateChart();
}
}

```

## Содержимое файла Transaction.cs

Transaction.cs

```
using System;
```

```
namespace Книга_учета
```

```
{
```

```
    // Класс для финансовой операции
```

```
    public class Transaction
```

```
    {
```

```
        public DateTime Date { get; set; }
```

```
        public string Description { get; set; }
```

```
        public decimal Amount { get; set; }
```

```
        public Category Category { get; set; } // Ссылка на категорию
```

```
        public TransactionType Type { get; set; } // Доход или расход
```

```
        public Transaction() { } // Обязательный конструктор без параметров для десериализации JSON
```

```
        public Transaction(DateTime date, string description, decimal amount, Category category, TransactionType type)
```

```
        {
```

```
            Date = date;
```

```
            Description = description;
```

```
            Amount = amount;
```

```
            Category = category;
```

```
            Type = type;
```

```
        }
```

```
        public override string ToString()
```

```
    {  
        return $"{Date.ToShortDateString()} - {Description} - {Amount:C}  
({Category})";  
    }  
}
```

```
// Тип операции  
public enum TransactionType  
{  
    Income,  
    Expense  
}  
}
```

Содержимое файла AccountingData.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using Newtonsoft.Json;

namespace Книга_учета
{
    // Класс для хранения всех данных
    public class AccountingData
    {
        public List<Category> Categories { get; set; } = new List<Category>();
        public List<Transaction> Transactions { get; set; } = new
List<Transaction>();

        // Конструктор
        public AccountingData() { }

        // Методы CRUD для категорий
        public void AddCategory(Category category)
        {
            if (!Categories.Any(c => c.Name == category.Name))
            {
                Categories.Add(category);
            }
            else
            {
                Console.WriteLine("Категория с таким именем уже существует.");
            }
        }

        public void UpdateCategory(string oldName, Category newCategory)
        {
            Category existingCategory = Categories.FirstOrDefault(c => c.Name ==
oldName);
            if (existingCategory != null)
            {
                existingCategory.Name = newCategory.Name;
                existingCategory.Description = newCategory.Description;
            }
            else
            {
                Console.WriteLine("Категория не найдена.");
            }
        }
    }
}
```



```

    }
}

public void DeleteCategory(string categoryName)
{
    Category categoryToRemove = Categories.FirstOrDefault(c => c.Name ==
categoryName);
    if (categoryToRemove != null)
    {
        Categories.Remove(categoryToRemove);
        // Удалить все транзакции, связанные с этой категорией. Важно!
        Transactions.RemoveAll(t => t.Category.Name == categoryName);
    }
    else
    {
        Console.WriteLine("Категория не найдена.");
    }
}

// Методы CRUD для операций
public void AddTransaction(Transaction transaction)
{
    Transactions.Add(transaction);
}

public void UpdateTransaction(DateTime oldDate, string oldDescription,
decimal oldAmount, Transaction newTransaction)
{
    Transaction existingTransaction = Transactions.FirstOrDefault(t => t.Date
== oldDate && t.Description == oldDescription && t.Amount == oldAmount);
    if (existingTransaction != null)
    {
        existingTransaction.Date = newTransaction.Date;
        existingTransaction.Description = newTransaction.Description;
        existingTransaction.Amount = newTransaction.Amount;
        existingTransaction.Category = newTransaction.Category;
        existingTransaction.Type = newTransaction.Type;
    }
    else
    {
        Console.WriteLine("Транзакция не найдена.");
    }
}

```

```

    public void DeleteTransaction(DateTime date, string description, decimal
amount)
    {
        Transaction transactionToRemove = Transactions.FirstOrDefault(t =>
t.Date == date && t.Description == description && t.Amount == amount);
        if (transactionToRemove != null)
        {
            Transactions.Remove(transactionToRemove);
        }
        else
        {
            Console.WriteLine("Транзакция не найдена.");
        }
    }

    // Подсчет баланса
    public decimal CalculateBalance()
    {
        decimal income = Transactions.Where(t => t.Type ==
TransactionType.Income).Sum(t => t.Amount);
        decimal expense = Transactions.Where(t => t.Type ==
TransactionType.Expense).Sum(t => t.Amount);
        return income - expense;
    }

    // Подсчет суммы по категориям
    public Dictionary<string, decimal> CalculateCategoryTotals()
    {
        return Transactions.GroupBy(t => t.Category.Name)
            .ToDictionary(g => g.Key, g => g.Sum(t => (t.Type ==
TransactionType.Income ? t.Amount : -t.Amount)));
    }

    // Сохранение в JSON
    public void SaveToJson(string filePath)
    {
        string json = JsonConvert.SerializeObject(this, Formatting.Indented);
        System.IO.File.WriteAllText(filePath, json);
    }

    // Загрузка из JSON
    public static AccountingData LoadFromJson(string filePath)
    {
        if (System.IO.File.Exists(filePath))
        {

```

```

        string json = System.IO.File.ReadAllText(filePath);
        return JsonConvert.DeserializeObject<AccountingData>(json);
    }
    else
    {
        Console.WriteLine("Файл не найден. Создан новый экземпляр
AccountingData.");
        return new AccountingData(); // Возвращаем новый экземпляр, если
файл не найден
    }
}

// Подготовка данных для графика (пример)
public Dictionary<string, decimal> GetDataForChart()
{
    return CalculateCategoryTotals(); // Используем уже посчитанные
суммы по категориям
}
}
}

```