МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство для учёта финансов «Finance Control»

Исполнитель

студент 2 курса группы 6 Гвоздовский Кирилл Владимирович

(Ф.И.О.)

Руководитель работы ассистент Чистякова Ю.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2023

Содержание

[Введение 3](#_Toc135220825)

[1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству 4](#_Toc135220826)

[2. Анализ требований к программному средству и разработка функциональных требований 6](#_Toc135220827)

[2.1 Описание инструментов для разработки программного средства 6](#_Toc135220828)

[2.2 Описание функциональности программного средства 7](#_Toc135220829)

[3. Проектирование программного средства 8](#_Toc135220830)

[3.1 Модель базы данных 8](#_Toc135220831)

[3.2 Структура проекта 9](#_Toc135220832)

[3.3 Структура классов программного средства 10](#_Toc135220833)

[3.4 Работа с базой данных 11](#_Toc135220834)

[3.5 Аутентификация пользователей в системе 12](#_Toc135220835)

[3.6 Окна приложения 12](#_Toc135220836)

[4. Реализация программного средства 13](#_Toc135220837)

[4.1 Реализация MVVM и других шаблонов 13](#_Toc135220838)

[4.2 Добавление кошелька 16](#_Toc135220839)

[4.3 Добавление и редактирование транзакции 17](#_Toc135220840)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 18](#_Toc135220841)

[6. Методика использования программного средства 22](#_Toc135220842)

[6.1 Регистрация и авторизация 22](#_Toc135220843)

[6.2 Домашнее представление 23](#_Toc135220844)

[6.3 Смена пользователя 26](#_Toc135220845)

[Заключение 27](#_Toc135220846)

[Список литературы 28](#_Toc135220847)

[Приложение А 29](#_Toc135220848)

[Приложение Б 30](#_Toc135220849)

[Приложение В 31](#_Toc135220850)

[Приложение Г 32](#_Toc135220851)

[Приложение Д 33](#_Toc135220852)

[Приложение Е 34](#_Toc135220853)

# Введение

В современном мире управление финансами является важной частью жизни каждого человека. Однако, не всегда удается контролировать свои расходы и вести учет своих финансов. Для решения этой проблемы было разработано приложение для контроля расходов.

В настоящее время многие люди сталкиваются с проблемой нехватки денежных средств, неспособностью правильно распределять свои доходы и контролировать свои расходы. Хорошо организованный учет финансов и контроль над своими тратами позволяют избежать подобных проблем и сделать свою жизнь более комфортной и уверенной.

Цель данного курсового проекта - разработать приложение, которое позволит пользователям легко и удобно контролировать свои расходы и доходы, а также анализировать свои финансовые показатели.

В рамках проекта будет произведен анализ существующих приложений для управления финансами, определены требования к разрабатываемому приложению, разработана архитектура приложения и реализованы основные функции.

Результатом данного проекта будет готовое приложение, которое позволит пользователям вести учет своих финансов, контролировать свои расходы и доходы, а также получать аналитическую информацию о своих финансовых показателях.

В настоящее время многие люди сталкиваются с проблемой нехватки денежных средств, неспособностью правильно распределять свои доходы и контролировать свои расходы. Хорошо организованный учет финансов и контроль над своими тратами позволяют избежать подобных проблем и сделать свою жизнь более комфортной и уверенной.

Цель данного курсового проекта - разработать приложение, которое позволит пользователям легко и удобно контролировать свои расходы и доходы, а также анализировать свои финансовые показатели. Язык разработки– C#. При выполнении курсового проекта будут использованы принципы и приемы ООП, база данных SQLite, и технология Windows Presentation Foundation (WPF).

# Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству

В настоящее время на рынке существует множество приложений, предназначенных для учёта расходов и доходов финансов, которые помогают пользователям эффективно управлять своими финансами.

В качестве одного из аналогов возьмем программу «Mint». Данная программа разработана для устройств на базе операционных систем Android и iOS. Данное приложение имеет следующие возможности:

– автоматический импорт транзакций;

– категоризация расходов, что позволяет видеть, куда уходят ваши деньги;

– управление бюджетом с помощью установки целей и отслеживание прогресса;

– графики и отчеты для анализа расходов и бюджетирования.

Интерфейс программы представлен на рисунке 1.1.

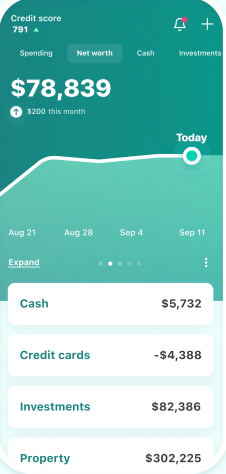


Рисунок 1.1 – приложение «Mint»

Недостатки «Mint»:

* отсутствие возможности для индивидуальной настройки категорий расходо.

Ещё один аналог для анализа –кросс-платформенное приложение «You Need a Budget (YNAB)». Имеет следующие возможности:

* фокус на бюджетировании и планировании расходов, помогающий управлять финансами более осознанно;
* ручной ввод транзакций, что обеспечивает более активное взаимодействие с финансовыми данными;
* Синхронизация данных между устройствами, что позволяет легко получать доступ к своим финансам везде, где есть интернет.

Интерфейс программы представлен на рисунке 1.2.

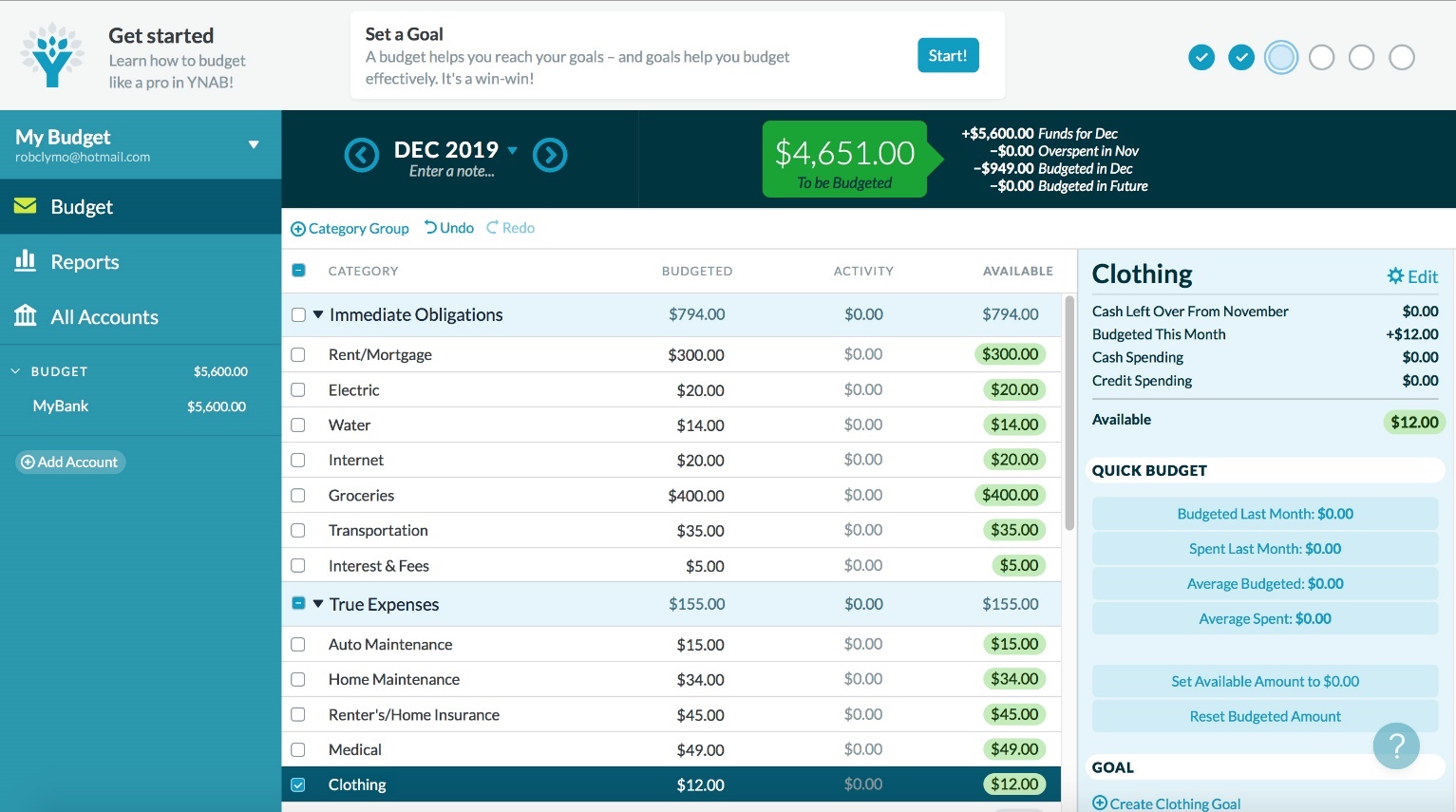


Рисунок 1.2 – You Need A Budget (YNAB)

# 2. Анализ требований к программному средству и разработка функциональных требований

2.1 Описание инструментов для разработки программного средства

В ходе разработки данного программного средства используются следующие инструменты:

* объектно-ориентированный язык программирования C#;
* платформа для кроссплатформенной разработки с открытым исходным

кодом .NET Framework Core;

* расширяемый язык разметки XAML;
* система управления базами данных SQLite;
* шаблон проектирования MVVM;
* фреймворк для работы с базой данных Entity Framework Core 6;
* интегрированная среда разработки Microsoft Visual Studio2022.

C# — это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework. Visual C#— это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и другим полезным, правильным классам, что существенно ускоряет цикл разработки.

XAML— это декларативный язык разметки. С точки зрения модели программирования .NET Core язык XAML упрощает создание пользовательского интерфейса для приложения .NET Core. Можно создать видимые элементы пользовательского интерфейса в декларативной XAML-разметке, а затем отделить определение пользовательского интерфейса от логики времени выполнения, используя файлы кода программной части, присоединенные к разметке с помощью определений разделяемых классов. Язык XAML напрямую представляет создание экземпляров объектов в конкретном наборе резервных типов, определенных в сборках.

SQLite (Structured Query Language Lite)— это компактная встраиваемая реляционная система управления базами данных (СУБД). Как СУБД, SQLite предоставляет механизмы для создания, организации, хранения и манипулирования данными в структурированном формате. SQLite предоставляет удобный и простой способ управления данными в небольших и средних проектах, особенно когда требуется локальное хранение данных или работа в офлайн-режиме. Однако, для крупных и масштабируемых проектов, требующих многопользовательской работы, более мощные СУБД, такие как Microsoft SQL Server, могут быть предпочтительными.

MVVM— шаблон проектирования архитектуры приложения, который позволяет отделить логику приложения от визуальной части, что упрощает тестирование и поддержку приложения.

2.2 Описание функциональности программного средства

На основе анализа были составлены следующие функциональные требования для программного средства для пользователя:

* авторизация и регистрация пользователя;
* добавление и удаление кошельков;
* возможность добавления кошельков с разными валютами;
* визуализация категорий расходов и доходов в виде диаграмм.

В графическом материале приведена схема Use-case, описывающая функциональность программного средства для пользователя.

Из данной схемы видно, что клиент может авторизироваться или создать аккаунт, добавить доход или расход и выбрать для них категорию, также создать несколько кошельков с разными валютами.

# 3. Проектирование программного средства

3.1 Модель базы данных

Для создания программного средства была разработана база данных FinanceManager, состоящая из 5 таблиц. Описания таблиц базы данных представлены в таблицах 3.1 – 3.5:

Таблица 3.1 – Таблица Accounts

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| AccountId | integer | Содержит идентификатор кошелька |
| Name | text | Содержит название кошелька |
| Balance | real | Содержит состояние баланса кошелька |
| CurrencyId | integer | Содержит идентификатор валюты |
| Image | text | Содержит путь до файла с фото иконки для кошелька |
| TakeIntoBalance | integer | Проверяет учитывается ли баланс кошелька в общую сумму аккаунта |

Таблица предназначена для хранения информации и баланса кошелька.Таблица

3.2 – Таблица Categories

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| Name | text | Название категории |
| Image | text | Строка с полным путём до файла с фото иконки для категории |
| DefaultSum | real | Содержит сумму для категории по умолчанию |
| Type | Integer | Указывает на тип категории: расходы или доходы |
| CategoryId | integer | Содержит идентификатор категории |

В таблице Categotries хранится информация о каждой категории.

Таблица 3.3 – Таблица Users

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| Username | text | Имя пользователя |
| Password | text | Пароль пользователя |
| Id | integer | Идентификатор пользователя |

Таблица Users предназначена для хранения логинов и паролей пользователей.

Таблица 3.4 – Таблица Transactions

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| AccountId | integer | Идентификатор кошелька |
| CategoryId | integer | Идентификатор категории |
| Comment | text | Содержит дополнительное описание для транзакции |
| Date | text | Содержит строку с датой транзакции |
| Money | real | Сумма транзакции |
| Type | integer | Тип транзакции: доход или расход |
| TransactionId | integer | Идентификатор транзакции |

Таблица Transactions содержит информацию о транзакции.

Таблица 3.5 – Таблица Currency

|  |  |  |
| --- | --- | --- |
| Имя столбца | Тип данных | Описание |
| Name | text | Название валюты |
| Designation | text | Сокращённое название валюты |
| CurrencyId | integer | Идентификатор валюты |

Таблица Currency хранит название валюты и её сокращённое название.

Диаграмма базы данных, отображающая связи между таблицами, представлена в графическом материале.

3.2 Структура проекта

Программное средство выполнено одним проектом, структура которого показана на рисунке 3.2

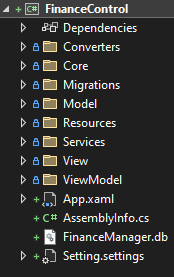


Рисунок 3.2 – Структура проекта

Описание хранимых в папках проекта элементах представлено в таблице 3.9.

Таблица 3.9 – Описание структуры проекта

|  |  |
| --- | --- |
| Элемент | Описание |
| Папка «Converters» | Содержит конвертеры |
| Папка «Models» | Содержит в себе модели для хранения данных |
| Папка «Resources» | Содержит стили для элементов интерфейса |
| Папка «ViewModels» | Содержит ViewModels для MVVM паттерна |
| Папка «Views» | Содержит представления, которые отвечают за визуальную часть для MVVM паттерна |
| Папка «Core» | Содержит контекст базы данных |
| Папка «Migrations» | Содержит миграции базы данных |
| Папка «Services» | Содержит классы для отрисовки диаграмм и получения даты |
| FinanceManager.db | Содержит базу данных приложения |
| App.xaml | Содержит стартовую точку, с которой начинается выполнение приложения |

3.3 Структура классов программного средства

Данное программное средство имеет структуру, представленную в виде диаграммы классов.

Диаграмма классов определяет различные типы классов в системе и статические связи между ними. На диаграммах классов также отображаются атрибуты классов, операции классов и ограничения, которые определяют взаимосвязи между классами. Важно отметить, что интерпретация диаграммы классов может зависеть от точки зрения, то есть от уровня абстракции. В процессе анализа диаграмма классов может представлять сущности предметной области, а в процессах проектирования и реализации— элементы программной системы.

Основными элементами диаграммы классов являются классы и связи между ними. Классы описываются при помощи атрибутов и операций.

Атрибуты определяют свойства объектов класса. Каждый объект класса получает свою уникальность благодаря различиям в их атрибутах и связях с другими объектами. Однако возможны случаи, когда объекты имеют одинаковые значения атрибутов и связей. В таких случаях уникальность объектов определяется самим фактом их существования, а не различиями в их свойствах. Каждому атрибуту должно быть присвоено уникальное имя в пределах класса. Имя атрибута может быть сопровождено его типом и значением по умолчанию.

В данном пункте будут рассматриваться только классы модели приложения без структуры модели представления.

Модель приложения представлена 6 основными классами.

Класс Account определят кошелёк, который имеет свой идентификатор, иконку, название, баланс и в какой валюте он считается.

Класс Category определяет категорию транзакции, иконку этой категории, и тип категории: расход или доход.

Класс Currency определяет валюту. Имеет название, идентификатор валюты и её аббревиатуру.

Класс Transaction определяет тип транзакции: расход или доход, сумму, категорию, дату транзакции и дополнительную информацию, указанную пользователем.

Класс User определяет пользователя, имеет его логин, пароль и идентификатор.

Класс Week определяет неделю, содержит в себе некоторые методы для работы с неделей.

На основании данных классов была создана база данных и ее таблицы. База данных с таблицами создавалась с помощью Entity Framework 6.

Диаграмма классов представлена в графической части.

3.4 Работа с базой данных

Вся работа с базой данных целиком возложена на WCF сервис. При разработке приложения использовался Entity Framework 6 с подходом Code First. Клиент взаимодействует с сервисом посредством вызова методов сервиса. При создании базы данных были заполнены несколько таблиц.

3.5 Аутентификация пользователей в системе

Программное средство подразумевает использование несколькими пользователями. Для этого необходима регистрация и аутентификация пользователей в системе. В системе может зарегистрироваться каждый. Данные, введенные пользователями, должны проходить валидацию на пустую строку.

Схема работы алгоритма регистрации приведена в приложении А, а схема работы алгоритма авторизации – в приложении Б.

3.6 Окна приложения

Набор окон приложения будет доступен каждому пользователю, Запуск ПО будет начинаться с окна логина, откуда пользователь может перейти к регистрации, или войти и перейти к основному меню. Далее он может добавить транзакцию (доход или расход), просматривать диаграммы доходов или расходов, а также создавать или удалять кошельки. Схемы пользователя представлена на рисунке 3.3.

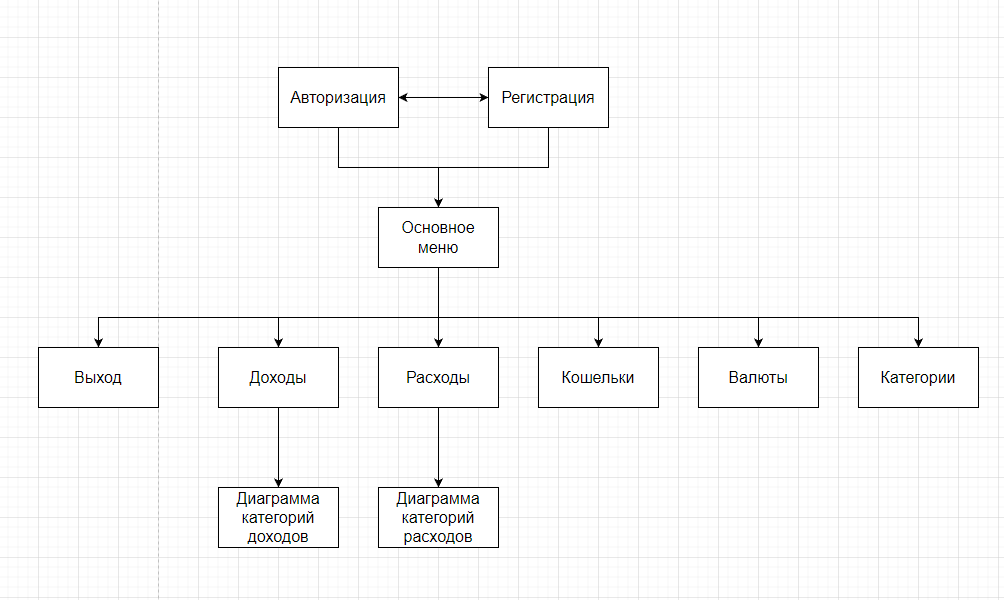


Рисунок 3.3 – Схема окон пользователя

# 4. Реализация программного средства

4.1 Реализация MVVM и других шаблонов

Базовый класс BaseVM (листинг 4.1) используется для реализации паттерна MVVM (Model-View-ViewModel). Он наследует интерфейс INotifyPropertyChanged, который служит для уведомления представления о изменениях свойств объекта. При реализации этого интерфейса классом, генерируется событие PropertyChanged каждый раз, когда происходит изменение значения свойства объекта. Это позволяет привязкам данных в пользовательском интерфейсе отслеживать состояние объекта и автоматически обновлять данные при изменении связанных свойств. В данном проекте все ViewModel основаны на классе BaseVM.

|  |
| --- |
| abstract class BaseVM : INotifyPropertyChanged  {  public event PropertyChangedEventHandler PropertyChanged;  protected void OnPropertyChanged([CallerMemberName] string name = null)  {  PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(name));  }  } |

Листинг 4.1 – структура класса BaseVM

Примером реализации потомка данного класса может служить класс AccountsViewModel (листинг 4.2).

|  |
| --- |
| class AccountsViewModel:BaseVM  {  Service service = Service.GetInstance();  public RelayCommand DeleteAccount { set; get; }  public RelayCommand AddAccount { set; get; }  public RelayCommand EditAccount { set; get; }  public ObservableCollection<AccountViewModel> Accounts  {  get  {  return new ObservableCollection<AccountViewModel>(ServiceConverter.ConvertToAccountVM(service.Accounts));  }  }  private AccountViewModel \_selectedAccount;  private object \_currentVM;  public AccountsViewModel()  {  #region Commands  DeleteAccount = new RelayCommand(obj =>  {  AccountViewModel newAccount = obj as AccountViewModel;  if (newAccount != null)  {  Service service = Service.GetInstance();  service.DeleteAccount(newAccount.Account);  OnPropertyChanged(nameof(Accounts));  OnPropertyChanged(nameof(TotalBalance));  }  CurrentVM = null;  });  AddAccount = new RelayCommand(obj =>  {  Service service = Service.GetInstance();  AccountViewModel \_newAccount = new AccountViewModel(new Account(service.DefaultCurrency),true);  CurrentVM = \_newAccount;  \_newAccount.SaveObject += (object sender, SaveObjectChangesEventArgs e) =>  {  AccountViewModel newAccount = e.Object as AccountViewModel;  service.AddAccount(newAccount.Account);  OnPropertyChanged(nameof(Accounts));  OnPropertyChanged(nameof(TotalBalance));  CurrentVM = null;  };  });  EditAccount = new RelayCommand(obj =>  {  CurrentVM = null;  if (SelectedAccount!=null)  {  CurrentVM = new AccountViewModel(new Account(SelectedAccount.Name, SelectedAccount.Balance, SelectedAccount.Currency, SelectedAccount.Image, SelectedAccount.TakeIntoBalance),false);  (CurrentVM as AccountViewModel).SaveObject += (object sender, SaveObjectChangesEventArgs e) =>  {  Account newAccount = (e.Object as AccountViewModel).Account;  Service service = Service.GetInstance();  service.EditAccount(SelectedAccount.Account, newAccount);  OnPropertyChanged(nameof(Accounts));  CurrentVM = null;  OnPropertyChanged(nameof(TotalBalance));  };  }  });  #endregion  }  #region Properties  public object CurrentVM  {  get => \_currentVM;  set  {  \_currentVM = value;  OnPropertyChanged();  }  }  public AccountViewModel SelectedAccount  {  get => \_selectedAccount;  set  {  \_selectedAccount = value;  OnPropertyChanged();  }  }  public float TotalBalance  {  get  {  float balance = 0;  foreach (var account in Accounts)  {  if (account.TakeIntoBalance)  if (account.Currency.Equals(DefaultCurrency))  balance += account.Balance;  }  return balance;  }  }  public Currency DefaultCurrency  {  get  {  return service.DefaultCurrency;  }  }  #endregion  public void UpDate()  {  CurrentVM = null;  }  } |

Листинг 4.2 – структура класса AccountsViewModel

4.2 Добавление кошелька

Добавление кошелька происходит следующим образом: авторизованный пользователь во вкладке «Кошельки» нажимает на кнопку «+», записывает имя кошелька, сумма, которая там изначально будет хранится, указывает, в какой валюте она будет хранится и выбирает иконку для кошелька. Все данные сохраняются в таблице Accounts.

На листинге 4.3 находится код, реализующий добавление кошелька.

|  |
| --- |
| AddAccount = new RelayCommand(obj =>{  Service service = Service.GetInstance();  AccountViewModel \_newAccount = new AccountViewModel(new Account(service.DefaultCurrency),true);  CurrentVM = \_newAccount;  \_newAccount.SaveObject += (object sender, SaveObjectChangesEventArgs e) =>{  AccountViewModel newAccount = e.Object as AccountViewModel;  service.AddAccount(newAccount.Account);  OnPropertyChanged(nameof(Accounts));  OnPropertyChanged(nameof(TotalBalance));  CurrentVM = null;};}); |

Листинг 4.3 – Команда, реализующая добавление кошелька

4.3 Добавление и редактирование транзакции

Чтобы добавить транзакцию, авторизованный пользователь переходит на вкладку «Расходы» или «Доходы» и нажимает на кнопку «+». Затем пользователь вводит сумму транзакции, выбирает соответствующий кошелек и категорию. При необходимости, пользователь может также добавить дополнительную информацию о транзакции. Наконец, пользователь выбирает дату и добавляет транзакцию. Все данные записываются в таблицу Transactions.

Отредактировать транзакцию можно нажав на её карточку. Можно изменить любую информацию, которую пользователь вводил при добавлении.

На листинге 4.4 находится код реализации записи на занятие.

|  |
| --- |
| public TransactionsViewModel(List<DateTime> periodDate, Period period,OperationType Type, Currency currency)  {  this.PeriodDate = periodDate;  this.Period = period;  this.Type= Type;  this.\_currency = currency;  SetText();  #region Commands  EditTransaction = new RelayCommand(obj =>  {  CurrentVM = null;  if (SelectedTransaction!= null)  {  CurrentVM = new TransactionViewModel(new Transaction(SelectedTransaction.Date, service.GetAccount(SelectedTransaction.Account), SelectedTransaction.Category, SelectedTransaction.Comment, SelectedTransaction.Money, SelectedTransaction.Type));  (CurrentVM as TransactionViewModel).SaveObject += Edittransaction;  (CurrentVM as TransactionViewModel).DeleteObject += DeleteTransaction;  }  });  AddTransaction = new RelayCommand(obj =>  {  CurrentVM = new TransactionViewModel(Type,currency);  (CurrentVM as TransactionViewModel).SaveObject += Addtransaction;  }); |

Листинг 4.4 – Команда, реализующая запись на занятие

# 5. Тестирование, проверка работоспособности и анализ полученных результатов

В курсовом проекте задействуется обработка ошибок, таким образом, что пользователь будет уведомлен о неудачном выполнении запроса к базе данных, или недоступности данных в формах.

При запуске приложения открывается представление с формой для авторизации, куда следует ввести свой логин и пароль. Если логин или пароль введены неверно, то в MessageBox будет выведена информация о произошедшей ошибке (рис. 5.1).

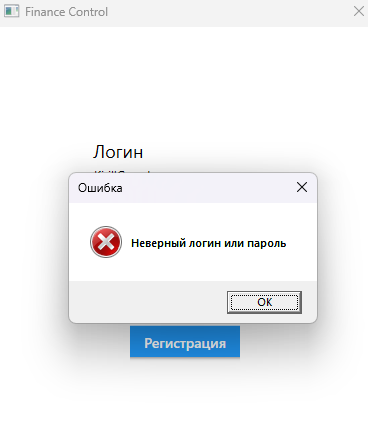


Рисунок 5.1 – При авторизации введены ошибочные данные

Далее рассмотрим представление для регистрации. Оно имеет 2 поля для ввода: логин и пароль. В приложении предусмотрена обработка следующих ошибок:

Если пользователь оставил поле незаполненным, он увидит сообщение с рисунка 5.2.

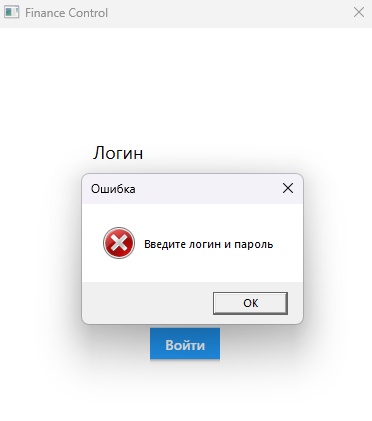


Рисунок 5.2 – При регистрации есть пустые поля

Если пользователь попробует занять логин, который уже существует в базе данных, он увидит сообщение с рисунка 5.3.

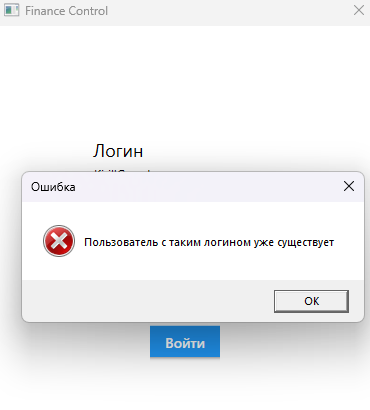


Рисунок 5.3 – При регистрации выбран уже занятый логин

Нельзя добавлять транзакцию с суммой 0, пользователь увидит сообщение, представленное на рисунке 5.4.

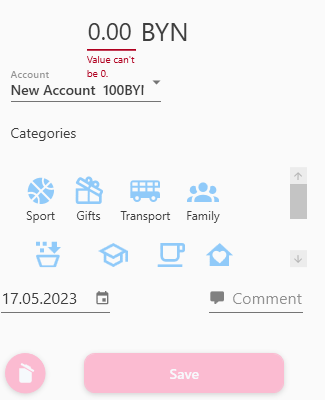


Рисунок 5.4

Так же были проведены другие проверки, которые прошли успешно.

# 6. Методика использования программного средства

6.1 Регистрация и авторизация

При открытии приложения открывается главное окно, которое содержит форму для входа. Если пользователь не имеет учетной записи, он может нажать кнопку «Регистрация», которая перенаправит его на форму регистрации. Форма для входа представлена на рисунке 6.1.

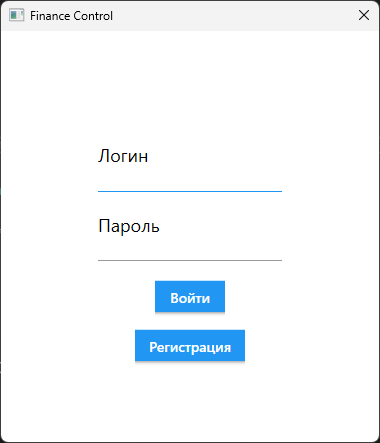


Рисунок 6.1 – Представление авторизации

Если была нажата кнопка «Регистрация», в открывшемся представлении следует ввести все данные о регистрируемом пользователе. В случае, если кнопка была нажата ошибочно, можно нажать на кнопку «Войти» и вернуться к представлению авторизации. Представление регистрации изображено на рисунке 6.2.

В случае, если кнопка была нажата ошибочно, можно нажать на кнопку «Войти» и вернуться к представлению авторизации. Представление регистрации изображено на рисунке 6.2.

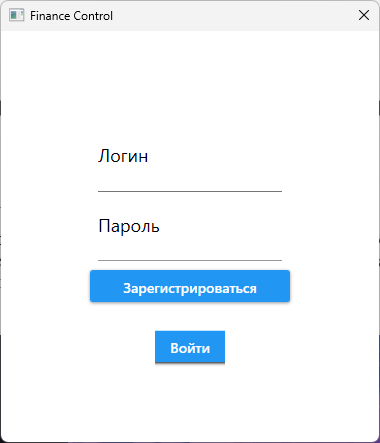


Рисунок 6.2 ‒ Представление регистрации

## **6.2** Домашнее представление

После успешной авторизации пользователя, приложение переключится на домашнее представление. На рисунке 6.3 изображено, что новый пользователь увидит после первой авторизации.

На рисунке 6.3 изображено, что новый пользователь увидит после первой авторизации.

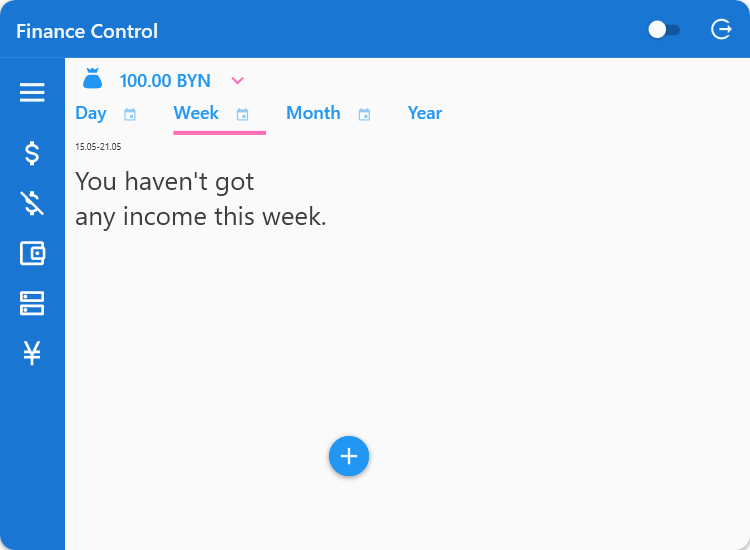


Рисунок 6.3 ‒ Домашнее представление после авторизации

Если пользователь желает просмотреть список валют, ему следует нажать в меню на кнопку «Валюта», ему откроется окно списка валют, как можно увидеть на рисунке 6.4.

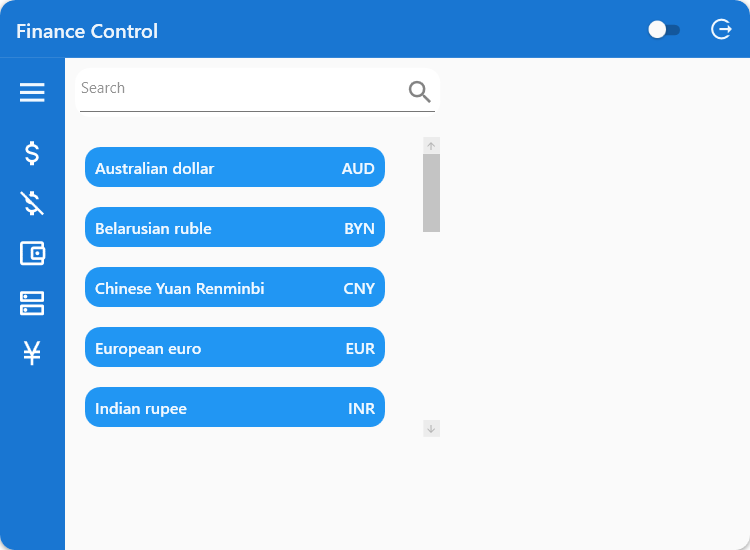


Рисунок 6.4 ‒ Список валют

Чтобы увидеть все ваши кошельки на аккаунте в меню нажмите «Кошельки», откроется список всех ваших кошельков и их баланс. Если нужно удалить кошелёк, то при нажатии на него появится кнопка «Delete account», при нажатии на неё удалится кошелёк. Это можно увидеть на рисунке 6.5.

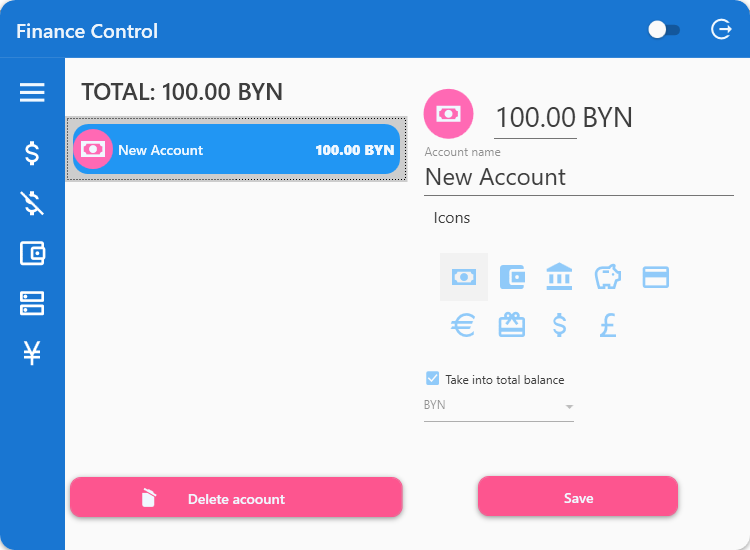


Рисунок 6.5— Удаление кошелька

Чтобы просмотреть все категории расходов и доходов, в меню выбрать «Категории». Это изображено на рисунке 6.6.

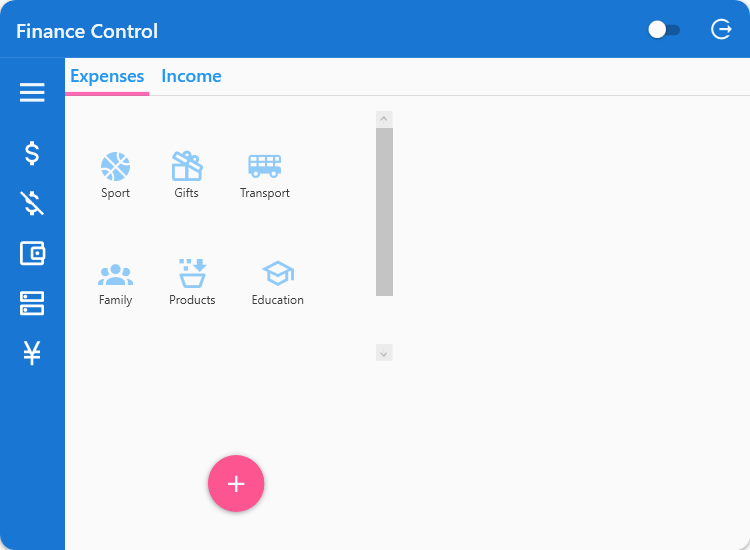


Рисунок 6.6 – Категории транзакций

## **6.3 Смена пользователя**

При желании сменить пользователя или выйти из аккаунта, пользователю стоит нажать на кнопку «Выход» в меню.

# Заключение

В рамках данного курсового проекта было разработано приложение для учета расходов и доходов, предназначенное для удобного и эффективного управления финансами пользователей. Приложение позволяет пользователям записывать свои расходы и доходы, а также проводить анализ своих финансовых операций.

В процессе разработки приложения был проведен анализ существующих аналогов, выявлены их преимущества и недостатки. Были определены требования к программному обеспечению, включая удобный интерфейс.

Приложение успешно прошло тестирование и проверку на соответствие заданным требованиям. Ошибок на стадии разработки не было выявлено, что говорит о его стабильной работе.

Разработанное приложение удовлетворяет всем требованиям, предъявленным в задании. В дальнейшем возможно оставление отзывов о инструкторе. Ошибок на стадии разработки выявлено не было, так как приложение работает без ошибок.

В результате выполнения данного курсового проекта были закреплены навыки программирования на языке C#, работа с платформой .NET Framework Core, создание десктопных приложений с использованием WPF, работа с базой данных с помощью Entity Framework Core 6, применение паттернов MVVM, а также проектирование базы данных и ее реализация в SQLite.

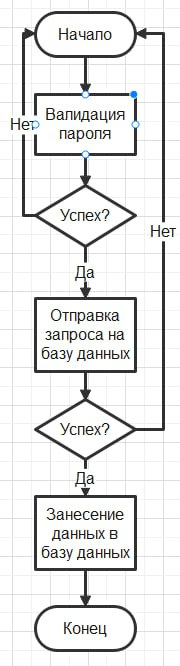
# Список литературы

1. docs.microsoft.com Сайт о программировании [Электронный ресурс] / Режим доступа: <https://docs.microsoft.com>.

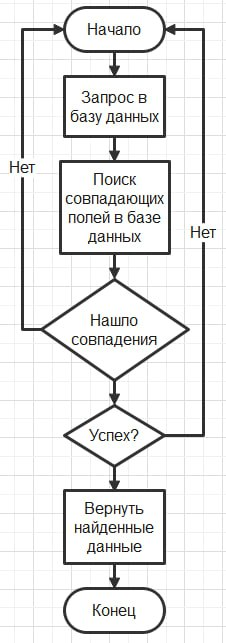
2. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.

3. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com>.

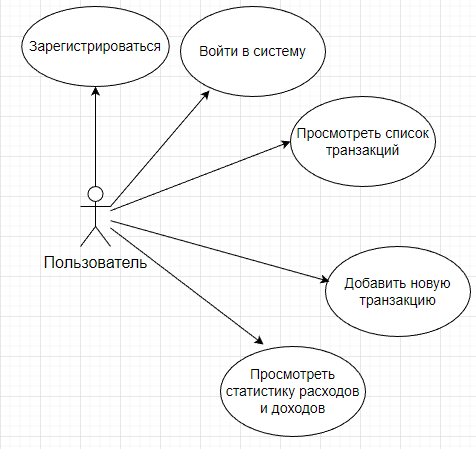
# Приложение А



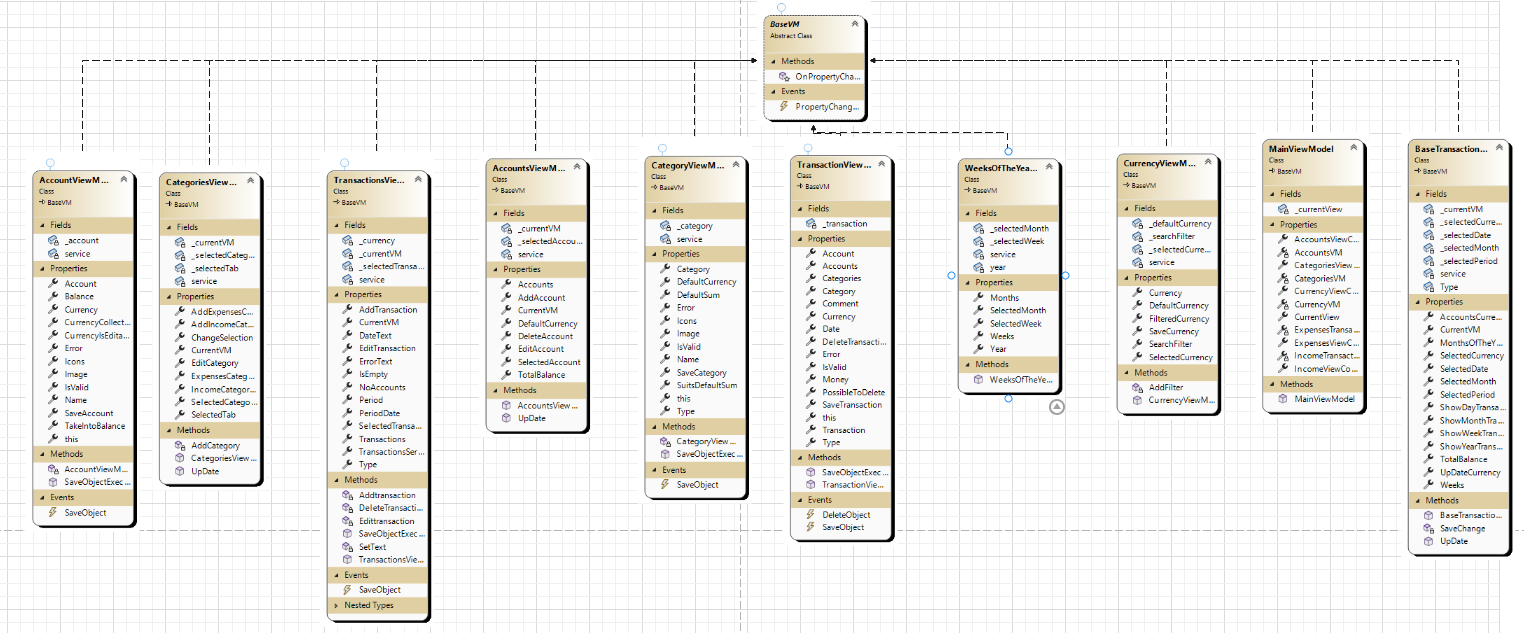
# Приложение Б

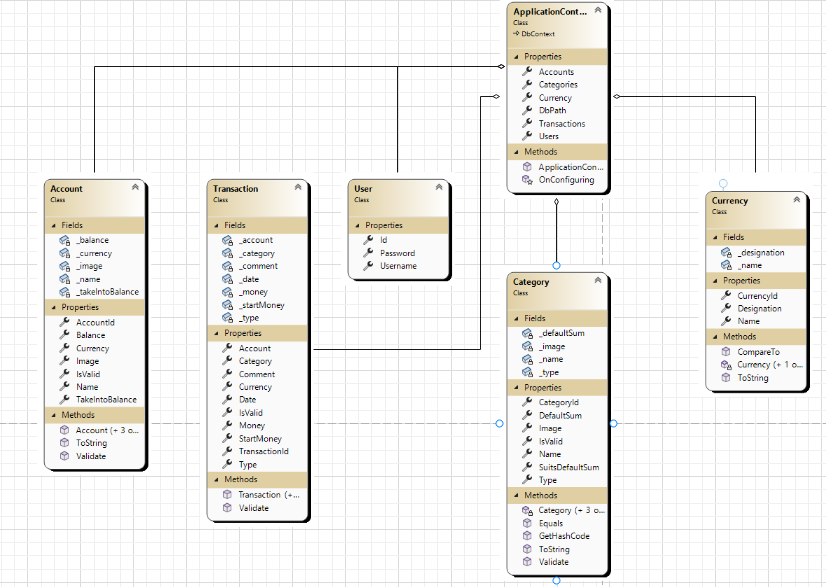


# Приложение В

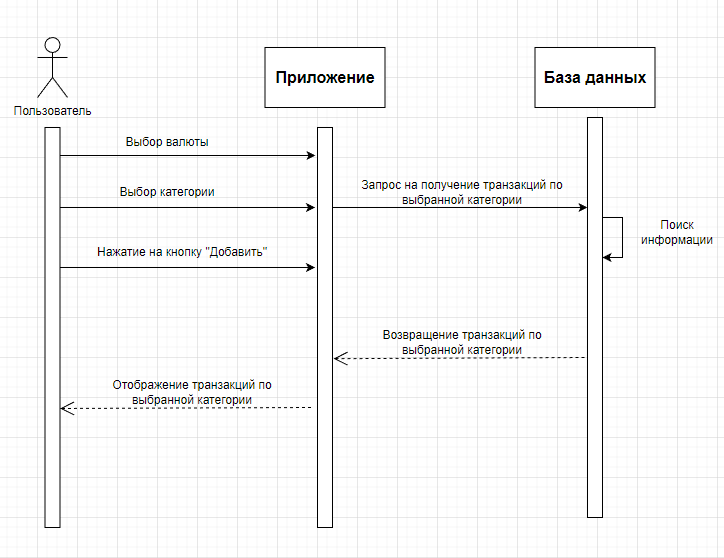


# Приложение Г





# Приложение Д



# Приложение Е

