МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

**Кафедра современных технологий программирования**

ЛАТОШ ЕВГЕНИЙ АЛЕКСАНДРОВИЧ

**Приложение учета домашних финансов**

Курсовая работа

по дисциплине «Языки программирования»

студента 2 курса специальности

1-40 01 01 «Программное обеспечение информационных технологий»

дневной формы получения образования

|  |  |
| --- | --- |
|  | Научный руководитель  Урбан Ольга Ивановна,  старший преподователь  кафедры современных технологий программирования |
|  |  |

Гродно, 2021

#### РЕЗЮМЕ

Тема курсовой работы

**«Приложение учета домашних финансов»**

Работа содержит 35 страницы, 24 иллюстраций, 4 использованных источников литературы.

Ключевые слова: архитектура приложения, аккаунт, статьи расходов и доходов, отчет, окно приложения.

Цель курсовой работы – разработка приложения для учета домашних финансов.

Объектом исследования выступают приложения для учета домашних финансов.

Предметом исследования настоящей работы является архитектура приложений.

В работе были использованы следующие методы: сравнительный анализ, моделирование, абстрагирование.

**Авторская характеристика работы.**

Программа написана на языке высокого уровня С#. В ней реализованы общие алгоритмы функционирования приложения. Основной задачей данной работы является создание законченного продукта, готового к использованию. Так же реализованные функции могут найти применение в создании новых приложений.

#### SUMMARY

Course work topic

**"Home Finance Accounting Application"**

The work contains 35 pages, 24 illustrations, 4 used literature sources.

Keywords: application architecture, account, expense and income items, report, application window.

The purpose of the course work is to develop an application for accounting for household finances.

The object of the research is applications for accounting for household finances.

The subject of research in this work is application architecture.

The following methods were used in the work: comparative analysis, modeling, abstraction.

**Author's characteristics of the work.**

The program is written in the high-level language C #. It implements the general algorithms for the operation of the application. The main task of this work is to create a finished product, ready to use. Also, the implemented functions can be used in the creation of new applications.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 6](#_Toc90381638)

[ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 8](#_Toc90381639)

[1.1 Основные аспекты 8](#_Toc90381640)

[1.2 Обзор существующих решений 9](#_Toc90381641)

[1.2.1 Приложение Quicken 9](#_Toc90381642)

[1.2.2 Приложение Moneydance 10](#_Toc90381643)

[1.2.3 Приложение Mint 10](#_Toc90381644)

[1.3 Вывод по главе 1 11](#_Toc90381645)

[ГЛАВА 2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА ДОМАШНИХ ФИНАНСОВ 12](#_Toc90381646)

[2.1 Проектирование архитектуры приложения 12](#_Toc90381647)

[2.2 Сохранение и чтение пользовательской информации 13](#_Toc90381648)

[2.3 Добавление и вывод на экран записей расходов и доходов 14](#_Toc90381649)

[2.4 Работа с записями расходов и доходов 15](#_Toc90381650)

[2.4.1 Просмотр истории записей по дате 15](#_Toc90381651)

[2.4.2 Подсчет статистики за месяц 15](#_Toc90381652)

[2.5 Проектирование авторизации и регистрации 16](#_Toc90381653)

[2.5.1 Проектирование авторизации 16](#_Toc90381654)

[2.5.2 Проектирование регистрации 17](#_Toc90381655)

[2.6 Вывод по главе 2 18](#_Toc90381656)

[ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА ДОМАШНИХ ФИНАНСОВ 19](#_Toc90381657)

[3.1 Сохранение и чтение пользовательской информации 19](#_Toc90381658)

[3.2 Добавление и вывод на экран записей расходов и доходов 20](#_Toc90381659)

[3.3 Работа с записями расходов и доходов 22](#_Toc90381660)

[3.3.1 Просмотр истории записей по дате 22](#_Toc90381661)

[3.3.2 Подсчет статистики за месяц 23](#_Toc90381662)

[3.4 Программная реализация авторизации и регистрации 26](#_Toc90381663)

[3.4.1 Реализация авторизации 26](#_Toc90381664)

[3.4.2 Реализация регистрации 27](#_Toc90381665)

[3.5 Демонстрация работы реализованного приложения 29](#_Toc90381666)

[3.6 Вывод по главе 3 29](#_Toc90381667)

[ЗАКЛЮЧЕНИЕ 30](#_Toc90381668)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc90381669)

# **ВВЕДЕНИЕ**

Если вы считаете, что управление домашними финансами бесполезно или неэффективно, то в таком случае вы сильно ошибаетесь. Учет финансов - одна из важнейших дисциплин в повседневной жизни. Способность управлять вашими финансами сильно помогает улучшить ваше финансовое состояние.

В случае если вы хотите оптимизировать свои расходы, а также провести наглядную аналитику вашего финансового баланса, то вы непременно столкнетесь с необходимостью завести определенную систему для учета домашними финансами. Так как возникает проблема: человеку очень трудно вести учет финансов самостоятельно. Данная проблема очень актуальна, особенно для студентов, так как очень сложно запоминать, когда и где были произведены затраты, как лучше сэкономить. Не так давно для решения данной проблемы люди использовали бумажные чековые книжки для отслеживания своих расходов и доходов, но данный подход не удобен для нашего времени. Я предлагаю решить данную проблему при помощи создания приложения. В данном приложении будут содержаться записи расходов и доходов, отчет о финансах, которые будут помогать человеку контролировать свои расходы.

Цель курсовой работы – разработать приложение для учета домашних финансов, основанную на учете статей доходов и расходов. Статьи делятся на два типа: доходы и расходы. Каждая статья представляет собой запись, содержащую данные: дата, сумма в руб., комментарий. Каждой записи можно присвоить свою категорию (например, платежи, еда, развлечения, спорт). Реализовать просмотр истории записей по дате, с возможностью сортировки (по времени или по сумме) и фильтрации по типу записи. Реализовать подсчет статистики и баланса за месяц: статистика по категориям в процентном и суммарном отношении, суммарные объемы доходов, расходов, прибыли. Реализация аккаунтов с возможностями регистрации, авторизации и удаления аккаунта.

Предусмотрено решение следующих задач:

1) Разработать архитектуру приложения

2) Организовать сохранение и чтение пользовательской информации

3) Реализовать добавление и вывод на экран записей расходов и доходов

4) Реализовать просмотр истории записей

5) Реализовать подсчет статистики за месяц

6) Реализовать авторизацию и регистрацию

# **ГЛАВА 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Основные аспекты**

Важность учета финансов в нашей повседневной жизни проявляется повсюду, от отслеживания расходов до покупки продуктов и оплаты счетов. Есть популярное высказывание, что «самый простой способ стать банкротом - не обращать внимания на свои расходы». Это высказывание отражает важность отслеживания ваших расходов.

Осведомленность о своих привычках к расходам - лучший способ использовать свои деньги. Когда вы знаете, сколько денег вы тратите, легко сбалансировать ваш доход с вашими расходами и даже откладывать на будущее.

Если вы оперируете бюджетом, лучший способ убедиться, что ваши расходы находятся в рамках бюджета, - это отслеживать свои расходы. Когда вы отслеживаете свои расходы, вы знаете, куда уходят ваши деньги, и можете быть уверены, что ваши деньги используются с умом.

Отслеживание своих расходов также позволяет понять, почему вы в долгах и как вы туда попали. Это поможет вам разработать подходящую стратегию выхода из долгов.

Одним из средств, позволяющих упростить отслеживание своих расходов, является приложение для учета домашних финансов. Суть таких приложений в том, что они сводят воедино внесенные вручную статьи доходов/расходов, сортируют их по категориям, строят наглядные графики. Такие приложения берут половину работы на себя и позволяют пользователю получить отчет о его бюджете, без нужды в рутинных расчетах.

Преимущества программного обеспечения для учета финансов включают повышенную точность за счет уменьшения или устранения человеческих ошибок в расчетах. Процессы ручного финансового учета включают в себя выполнение большого количества математических расчетов вручную. Неправильный расчет на ранней стадии процесса может сильно повлиять на конечный баланс.

Кроме того, использование такого программного обеспечения позволяет пользователям обрабатывать статьи расходов и доходов с большой скоростью. Частично это увеличение скорости связано с использованием компьютеров, которые могут обрабатывать цифры намного быстрее, чем человеческий мозг [2].

## **1.2 Обзор существующих решений**

Чтобы выявить основные аспекты, связанные с реализацией приложения учета домашних финансов, необходимо ознакомиться с рынком и установить на какие приложения существует спрос на данный момент и каким требованиям они должны отвечать. Рассмотрим разные приложения, которые могут использоваться чаще всего и имеют богатый функционал, позволяющий достичь нужного результата.

### **1.2.1 Приложение Quicken**

Quicken - это давно зарекомендовавший себя инструмент для управления личными учетными записями, и, хотя его репутация была основана на настольной версии, теперь он доступен для запуска в качестве приложения на ваших мобильных устройствах.

Quicken предлагает хороший набор инструментов для финансовой отчетности. Они устанавливаются для нескольких различных областей, а именно для составления бюджета, счетов и даже инвестиций. Для составления бюджета он предлагает вам возможность ввести свои покупки и доход, чтобы вы могли сравнить их вместе, чтобы лучше понять, сколько вы тратите по сравнению с тем, сколько вы зарабатываете.

Что касается счетов, вы также можете видеть, за какие коммунальные услуги и тому подобное вы постоянно платите, и видеть, как суммы, которые должны быть оплачены, так и то, сколько денег у вас осталось. Для целей бухгалтерского учета вы даже можете объединить свои банковские счета и счета по кредитной карте в одном месте, чтобы иметь очень четкое представление о том, сколько вы платите. Это особенно удобно, так как люди легко недооценивают, насколько обычные небольшие покупки могут добавить к затратам.

### **1.2.2 Приложение Moneydance**

Moneydance, созданный в первую очередь для пользователей Mac (но также и для Windows и Linux), представляет собой настольный пакет управления деньгами с очень аккуратным однооконным интерфейсом. Загрузите его, и вы получите мгновенное представление о своих финансах, предстоящих счетах, последних расходах и многом другом. Щелкните элемент на левой боковой панели, и основное содержимое изменится, чтобы отразить его.

Его функции отчетности довольно сильные, если не впечатляющие, а одним из самых полезных разделов Moneydance является реестр учетных записей. Также есть приложение iOS для регистрации транзакций на ходу, которое позже синхронизируется с программным обеспечением на вашем рабочем столе [3].

### **1.2.3 Приложение Mint**

Mint - это исключительная служба личных финансов, получившая несколько наград благодаря своей простоте, удобству использования и умным финансовым инструментам. Он позволяет вам настраивать подключения к своим финансовым счетам в Интернете, проверять свой кредитный рейтинг и, среди прочего, получить хорошую оценку вашего собственного капитала.

Разработчики Mint продолжают использовать технологию машинного обучения (ИИ) для обеспечения более интеллектуального, персонализированного и автоматизированного взаимодействия с более глубоким пониманием и интеллектом. В сервисе представлены улучшенные инструменты навигации, и он заранее побуждает вас улучшать методы управления капиталом вместо пассивного отслеживания счетов.

## **1.3 Вывод по главе 1**

Как можно понять, ведение личной бухгалтерии, планирование доходов и расходов не является обязательным для каждого из нас, но позволяет более рационально использовать имеющиеся ресурсы. Приложения для учёта домашних финансов значительно облегчают ведение домашней бухгалтерии, помогают экономить на мелочах, а значит и накапливать сбережения.

Также приложения, рассмотренные в данной главе, позволили выявить основные аспекты, связанные с реализацией приложения. А именно:

1. Реализовать добавление записей расходов и доходов
2. Реализовать подсчет статистики за месяц
3. Реализовать вывод графиков
4. Реализовать просмотр истории записей

# **ГЛАВА 2 ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА ДОМАШНИХ ФИНАНСОВ**

## **Проектирование архитектуры приложения**

При проектировании архитектуры приложения будет использоваться паттерн MVP.

Model View Presenter (MVP) - это шаблон проектирования, который особенно полезен для реализации пользовательских интерфейсов таким образом, чтобы разделять программное обеспечение на отдельные задачи, например, предназначенные для обработки и хранения данных (модель), бизнес-логики, маршрутизации пользовательских команд и т. д., тем самым делая большую часть вашего кода доступной для модульного тестирования.

Шаблон проектирования MVP разделяет следующие части:

**Model**: Хранит данные, которые будут отображаться или действовать в пользовательском интерфейсе.

**View**: Пассивный пользовательский интерфейс, который отображает данные модели и направляет инициируемые пользователем события, такие как команды щелчка мыши, представлению, чтобы он действовал в соответствии с этими данными.

**Presenter**: Действует на Model и View. Он извлекает данные из Model и отображает их в View.

Когда пользователь отправляет запрос, такой как щелчок мышью по кнопке управления, View принимает запрос и делегирует запрос объекту Presenter, который вызовет выбранный собственный метод.

Presenter, который может получить как состояние текущего View, так и текущие данные Model, выполнит любые требуемые для него вычисления или бизнес-логику и обновит View с результатами, и при необходимости работать с Model [4].

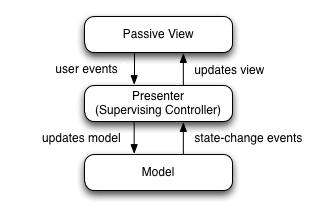


Рис. 2.1 (Схема паттерна MVP)

## **Сохранение и чтение пользовательской информации**

В данном курсовом проекте для сохранения или чтения пользовательской информации будет использоваться сериализация и десериализация.

Сериализация в C # - это процесс приведения объекта в форму, которая может быть записана в потоке. Это процесс преобразования объекта в форму, чтобы его можно было сохранить в файле, базе данных или памяти; или он может быть передан по сети. Его основная цель - сохранить состояние объекта, чтобы его можно было воссоздать при необходимости.

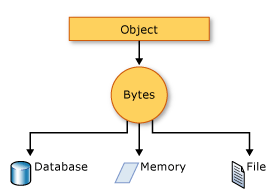


Рис. 2.2 (Схема сериализации)

Как следует из названия, десериализация в C # - это процесс, обратный сериализации. Это процесс возврата сериализованного объекта, чтобы его можно было загрузить в память. Он воскрешает состояние объекта, задавая свойства, поля и т. Д.

Существует несколько типов сериализации:

1. Двоичная сериализация
2. XML-сериализация
3. Сериализация JSON

Поскольку в этом приложении данные сохраняются в XML файлах, мной будет использоваться тип XML-сериализации.

## **Добавление и вывод на экран записей расходов и доходов**

В данном приложении записи расходов и доходов будут выводится в виде таблиц. В разных разделах приложения будут находится разные таблицы, которые будут принимать записи по определенному признаку. Для того чтобы реализовать такой способ вывода записей, мной будет использован объект DataGridView.

В Microsoft Visual Studio элемент управления dataGridView разработан для использования в приложениях, созданных по шаблону Windows Forms Application. Данный элемент управления позволяет организовывать данные в виде таблицы. Данные могут быть получены из базы данных, коллекции, внутренних переменных — массивов или других объектов программы.

Для того чтобы добавить запись о расходах или доходах в приложение, требуется заполнить соответствующие строки и кликнуть на кнопку “Добавить”. После того как данные проверяться на корректность, они добавятся в один из объектов List.



Entry

2

New Entry

Entry

4

Entry

3

Entry

1

Рис. 2.3 (Схема добавления записи в List)

После того как запись была добавлена в общий список, она добавляется в соответствующий объект DataGridView, который выводит запись на экран.

## **Работа с записями расходов и доходов**

### **2.4.1 Просмотр истории записей по дате**

В приложении предусмотрен просмотр истории записей по дате. Для реализации этой возможности требуется получить из общего списка записей список с записями по определенной дате. Так как в записях о расходах и доходах хранится дата, то для получения такого списка достаточно воспользоваться методом FindAll() из объекта List. После чего записи из этого списка добавляются в соответствующий DataGridView, для вывода на экран.

### **2.4.2 Подсчет статистики за месяц**

Для подсчета статистики за месяц, в программе существуют списки категорий. Каждая категория хранит: имя категории, сумма и процентное соотношение.

При добавлении новой записи расходов или доходов, определяется вид записи и его категория. Если записей о данной категории не было в течении последнего месяца, она добавляется, в противном случае пересчитываются значения для существующей категории. Также при каждом добавлении записи, пересчитывается процентное соотношение категорий для данного месяца, после чего они сортируются по убыванию.

В конечном итоге данные добавляются в соответствующие DataGridView, а также на их основе строятся графики, которые выводятся на экран.

## **Проектирование авторизации и регистрации**

### **2.5.1 Проектирование авторизации**

При запуске программы, пользователя встречает окно авторизации. В данном окне есть два текстовых поля (для логина и пароля), кнопка, для попытки авторизации, а также две подсказки, которые можно свернуть. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран регистрации.

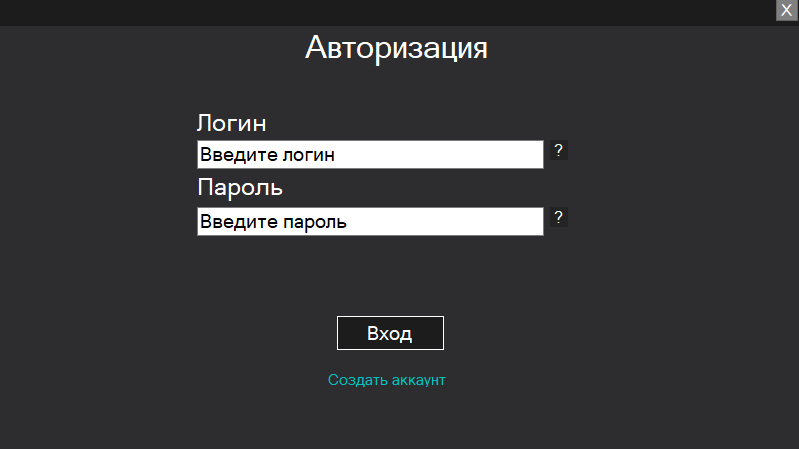


Рис. 2.4 (Графическое представление окна авторизации)

При нажатии на кнопку «Вход», происходит поиск аккаунта из списка по введенным логину и паролю, в случае если такой аккаунт был найден, производится вход в приложение при помощи данного аккаунта, в противном случае на экран выводится соответствующая ошибка.

### **2.5.2 Проектирование регистрации**

В окне регистрации есть четыре текстовых поля (для логина, пароля, фамилии и имени), кнопка, для попытки регистрации, а также две подсказки, которые можно свернуть. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран авторизации.

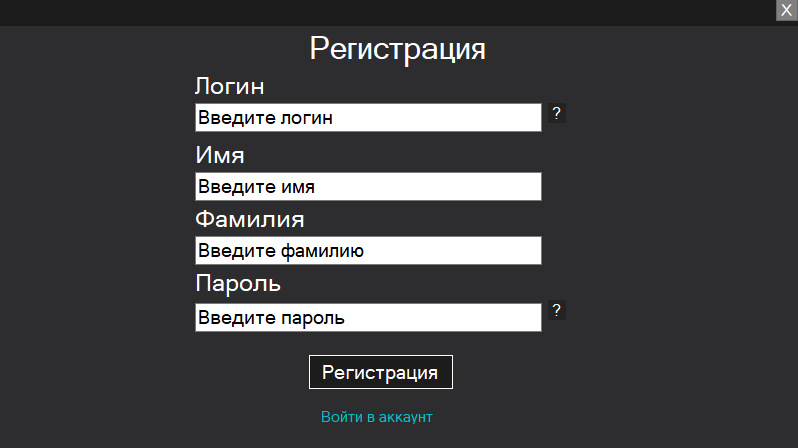


Рис. 2.5 (Графическое представление окна регистрации)

При нажатии на кнопку «Регистрация», происходит поиск аккаунта из списка по введенным логину, в случае если такой аккаунт не был найден, создается новый аккаунт, а также производится вход в приложение при помощи данного аккаунта, в противном случае на экран выводится соответствующая ошибка.

## **2.6 Вывод по главе 2**

Во второй главе был выбран и разобран шаблон проектирования MVP. На основе которого будет построена архитектура приложения. Также были спроектированы основные возможности приложения и выбраны способы их реализации. После того, как были спроектированы основные возможности приложения можно приступать к их реализации программным кодом.

В качестве среды разработки приложения будет использована Visual Studio 2019 Community. Используемый язык программирования – С#.

# **ГЛАВА 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ ДЛЯ УЧЕТА ДОМАШНИХ ФИНАНСОВ**

## **Сохранение и чтение пользовательской информации**

Для сохранения в XML файл или чтения из XML файла будет использоваться сериализация и десериализация.

Сериализация реализована в обобщенном методе public void Save<T>(String file, T item) класса Database.

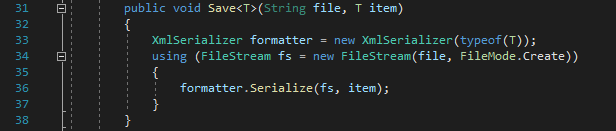


Рис. 3.1 (Метод реализующий сериализацию)

Данный метод принимает два входных параметра (Первым параметром идет путь к файлу, вторым – элемент, который надо сохранить), также при вызове метода нужно указать тип сохраняемого объекта.

В строке 33 создается объект XmlSerializer, который позволяет сериализовать объект в XML файл. Кроме того, конструктору XmlSerializer передается тип сериализуемого объекта.

В строке 34 создается поток, связанный с XML файлом, путь к которому передавался первым параметром метода. Кроме того, этот поток будет создавать новый файл с данным путем, вне зависимости существует уже такой файл или нет.

В строке 36 происходит сериализация объекта, который был передан в метод, используя поток, созданный в строке 34.

Десериализация реализована в обобщенном методе public T Read<T>(String file) класса Database.

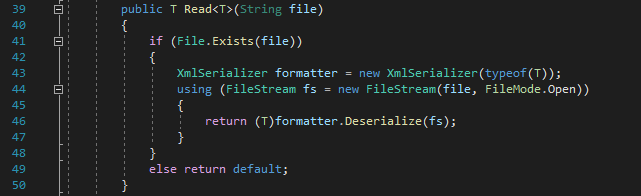


Рис. 3.2 (Метод реализующий десериализацию)

Данный метод принимает один входной параметр (путь к файлу), также при вызове метода нужно указать тип десериализуемого объекта.

В строке 41 проверяется, существует ли файл по пути, который был передан в метод. Если такого файла не существует, то метод возвращает значение по умолчанию.

В строке 43 создается объект XmlSerializer, который позволяет десериализовать объект из XML файл. Кроме того, конструктору XmlSerializer передается тип десериализуемого объекта.

В строке 44 создается поток, связанный с XML файлом, путь к которому передавался первым параметром метода. Кроме того, этот поток будет открывать файл, находящийся по данному пути.

В строке 46 возвращается объект, который был десериализован используя поток, созданный в строке 44.

## **Добавление и вывод на экран записей расходов и доходов**

В данном проекте записи расходов и доходов хранятся в объекте List. Записи выводятся на экран при помощи объекта DataGridView.

Добавление новой записи реализовано в методе public void UpdateElements() класса MainPresenter.

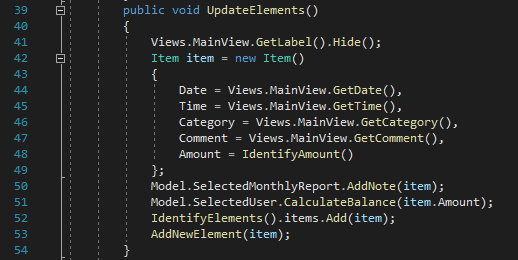


Рис. 3.3 (Метод реализующий добавление записи)

В строке 41 скрывается запись, которая сигнализирует пользователю, что список пуст.

В строках 42-49 создается новый объект записи, который заполняется данными, что ввел пользователь.

В строке 50 обновляется статистика за месяц с учетом новой записи. А в строке 51 вычисляется текущий баланс с учетом новой записи.

В строке 52 определяется вид записи (расходов или доходов) и данная запись добавляется в соответствующий список.

В строке 53 вызывается метод, который выводит новую запись на экран.

Вывод записи на экран реализовано в методе private void AddNewElement(Item item) класса MainPresenter.

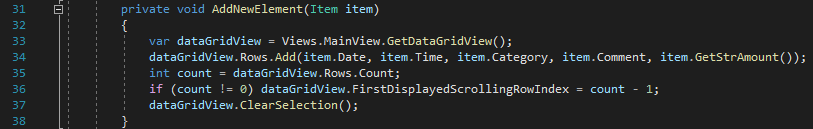


Рис. 3.4 (Метод реализующий вывод записи на экран)

Данный метод принимает один входной параметр (запись которая будет выведена).

В строке 33 определяется в какую таблицу добавлять запись. Далее в строке 34 в нужную таблицу добавляется новая строка и данные из записи распределяются по соответствующим столбцам.

В строке 35 определяется количество строк в таблице. Далее в строке 36 таблица прокручивается на последнюю запись.

## **Работа с записями расходов и доходов**

### **3.3.1 Просмотр истории записей по дате**

Просмотр истории записей по дате реализовано в методе public void UpdateHistory() класса MainPresenter.

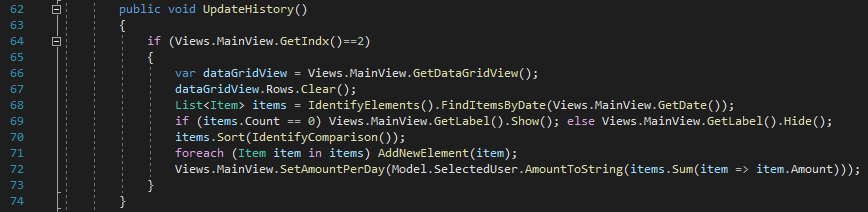


Рис. 3.5 (Метод реализующий просмотр записей по дате)

В строке 64 проверяется, что метод был вызван в соответствующем разделе приложения.

В строке 66 определяется в какую таблицу выводить записи. Далее в строке 67 эта таблица очищается.

В строке 68 определяется тип записи (расходы или доходы) и формируется список из записей соответствующих искомой дате.

В строке 69 в зависимости есть ли искомые записи, выводится или скрывается текстовое предупреждение пользователю.

В строке 70 список сортируется по выбранному типу сортировки. И в строке 71 данные записи добавляются в таблицу.

В строке 72 высчитывается и выводится на экран общая сумма выбранных записей.

Поиск записей по дате реализован в методе public List<Item> FindItemsByDate(String date) класса Elements.

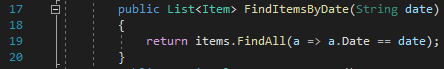


Рис. 3.6 (Метод реализующий поиск записей по дате)

Данный метод принимает один входной параметр (дата, по которой будут выбираться записи). Так как каждая запись хранит дату, то для выбора всех записей по дате достаточно использовать метод FindAll() из класса List. Поэтому в строке 19 вызывается метод FindAll() с соответствующим лямбда выражением.

### **3.3.2 Подсчет статистики за месяц**

Подсчет статистики за месяц происходит в методе public void AddNote(Item item) класса MonthlyReport. Данный метод вызывается в методе public void UpdateElements() в строке 50.

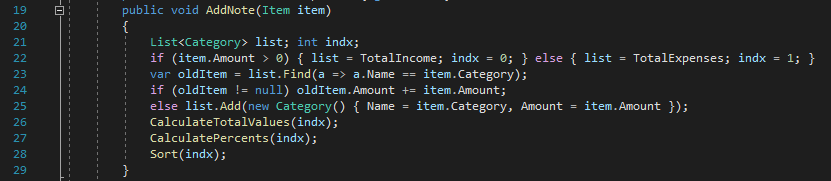


Рис. 3.7 (Метод реализующий подсчет статистики)

Данный метод принимает один входной параметр (Новая запись, которая будет изменять статистику).

В строке 22 определяется тип записи (расходы или доходы) и выбирается нужный список записей.

В строке 23 производится проверка, были ли добавлены записи с такой категорией до этого. Если записи с такой категорией уже проходили, то в строке 24 пересчитывается общая сумма этой категории. В противном случае в строке 25 создается новая категория, которая добавляется в общий список.

В страх 26 – 28 пересчитываются значения категорий выбранного типа записей. А именно: в строке 26 пересчитывается общая сумма всех категорий, в строке 27 пересчитываются процентные соотношения для каждой категории, в строке 28 категории сортируются.

Вывод на экран графиков и таблиц об отчете реализован в методе public void UpdateCharts() класса MainPresenter.

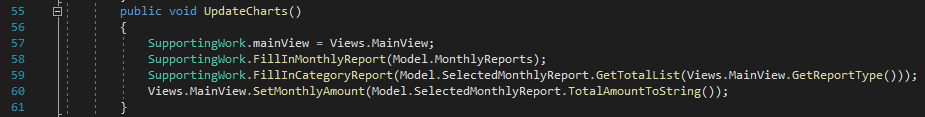


Рис. 3.8 (Метод обновляющий графики и таблицы об отчете)

В строке 57 классу SupportingWork передается ссылка на главное окно приложения. Далее в строках 58 - 59 производится обновление графиков и таблиц. После чего в строке 60 выводится на экран общая сумма за месяц.

Обновление отчета о категориях реализовано в методе public static void FillInCategoryReport(List<Category> list) класса SupportingWork.

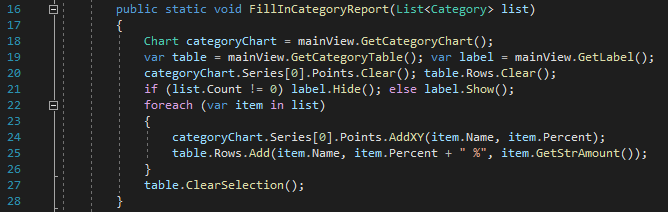


Рис. 3.9 (Метод обновляющий отчет о категориях)

Данный метод принимает один входной параметр (Список категорий текущего месяца, которые нужно вывести).

В строках 18 - 19 определяются ссылки на график, таблицу и текстовое предупреждение. Далее в строке 20 график и таблица очищаются. После чего в строке 21 проверяется список на пустоту и в зависимости от результата скрывается или выводится предупреждение.

В строках 22 – 26 проходим по каждой категории. Далее в строке 24 добавляем категорию в график, а в строке 25 добавляем значения категории в таблицу.

Обновление статистики за несколько месяцев реализовано в методе public static void FillInMonthlyReport(List<MonthlyReport> monthlyReports) класса SupportingWork.

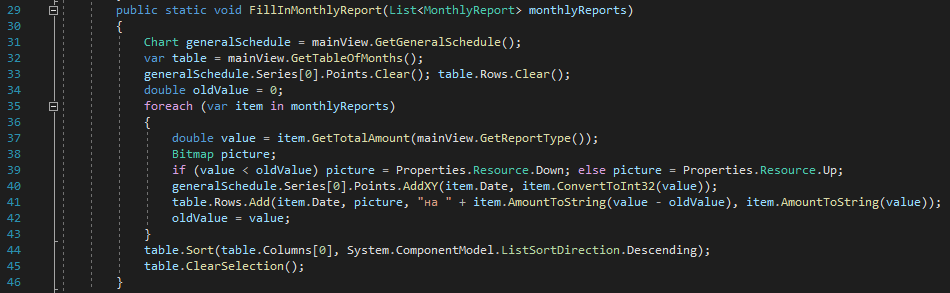


Рис. 3.10 (Метод обновляющий статистику за несколько месяцев)

Данный метод принимает один входной параметр (Список отчетов по месяцам).

В строках 31 - 32 определяются ссылки на график и таблицу. Далее в строке 33 график и таблица очищаются.

В строках 35 – 43 проходим по каждому отчету за месяц. В строке 37 получаем нужную сумму просматриваемого месяца. В строках 38 – 37 сравниваем два месяца и в зависимости от результата присваиваем нужную картинку. Далее в строке 41 добавляем месяц в график, а в строке 41 добавляем значения месяца в таблицу. После чего в строке 42 сохраняем сумму данного месяца.

В строке 44 сортируем таблицу по дате.

Хотелось бы отметить, так как я строю графики при помощи объекта Chart, мне достаточно сообщить объекту о новой точке, а построение графика проходит автоматически.

## **Программная реализация авторизации и регистрации**

### **3.4.1 Реализация авторизации**

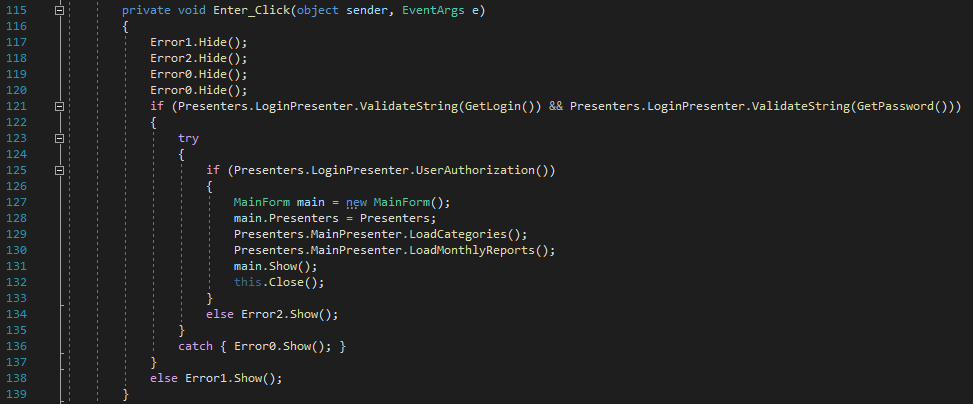
Авторизация вызывается в событии private void Enter\_Click(object sender, EventArgs e) класса Form1. Данное событие отвечает за нажатие кнопки “Вход”. 

Рис. 3.11 (Событие вызывающее авторизацию)

В строках 117 – 120 происходит скрытие всех предупреждений об ошибке. Далее в строке 121 строки проверяются на корректность, в случае если они не корректны в строке 138 выводится предупреждение о данной ошибке.

В строке 125 производится попытка авторизации, если не получается произвести авторизацию, то в строке 134 выводится предупреждение о данной ошибке.

В строках 127 – 128 создается главное окно приложения и ему передаются нужные для работы ссылки.

В строке 129 загружаются записи данного пользователя, а в строке 130 загружается его статистика за несколько месяцев. Если при загрузке данных произойдет ошибка, то в строке 136 выводится предупреждение о данной ошибке.

Далее в троке 131 главное окно выводится на экран, а в строке 132 уничтожается окно авторизации.

Авторизация пользователя реализована в методе public bool UserAuthorization() класса LoginPresenter.

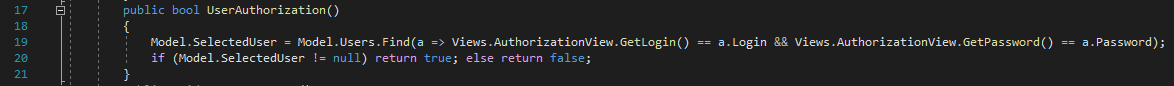


Рис. 3.12 (Метод реализующий авторизацию)

В строке 19 происходит поиск аккаунта по логину и паролю. Далее в строке 20 если такой пользователь существует возвращается значение true, в противном случае – false.

### **3.4.2 Реализация регистрации**

Регистрация вызывается в событии private void Enter\_Click(object sender, EventArgs e) класса Form2. Данное событие отвечает за нажатие кнопки “Регистрация”.

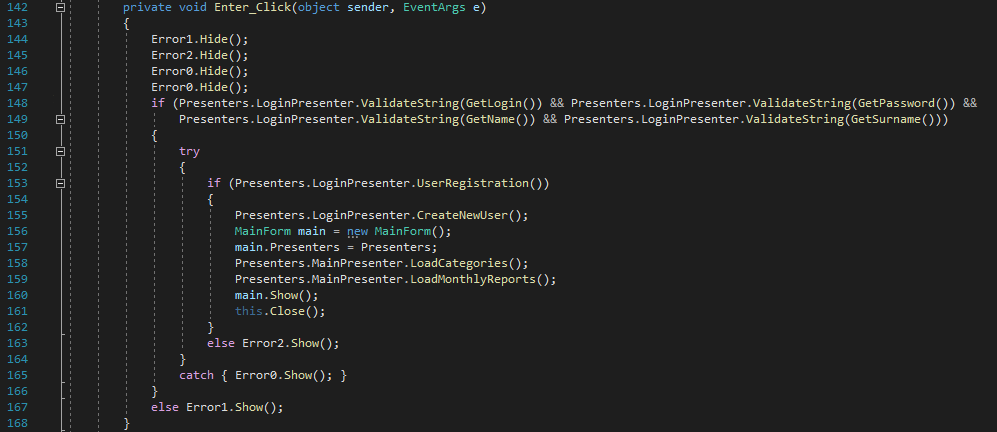


Рис. 3.13 (Событие вызывающее регистрацию)

В строках 144 – 147 происходит скрытие всех предупреждений об ошибке. Далее в строках 148 - 149 строки проверяются на корректность, в случае если они не корректны в строке 167 выводится предупреждение о данной ошибке.

В строке 153 производится попытка регистрации, если не получается произвести регистрацию, то в строке 163 выводится предупреждение о данной ошибке.

В строке 155 создается новый аккаунт по данным которые ввел пользователь.

В строках 156 – 157 создается главное окно приложения и ему передаются нужные для работы ссылки.

В строке 158 загружаются записи данного пользователя, а в строке 159 загружается его статистика за несколько месяцев. Если при загрузке данных произойдет ошибка, то в строке 165 выводится предупреждение о данной ошибке.

Далее в троке 160 главное окно выводится на экран, а в строке 161 уничтожается окно регистрации.

Регистрация пользователя реализована в методе public bool UserRegistration() класса LoginPresenter.

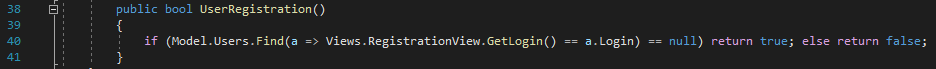


Рис. 3.14 (Метод реализующий регистрацию)

В строке 40 происходит поиск аккаунта по логину. Далее если такой аккаунт не был найден, возвращается значение true, в противном случае возвращается значение false.

## **Демонстрация работы реализованного приложения**

При запуске программы, пользователя встречает окно авторизации. В данном окне есть два текстовых поля (для логина и пароля), кнопка (для входа в приложение), а также две подсказки, которые можно свернуть. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран регистрации.

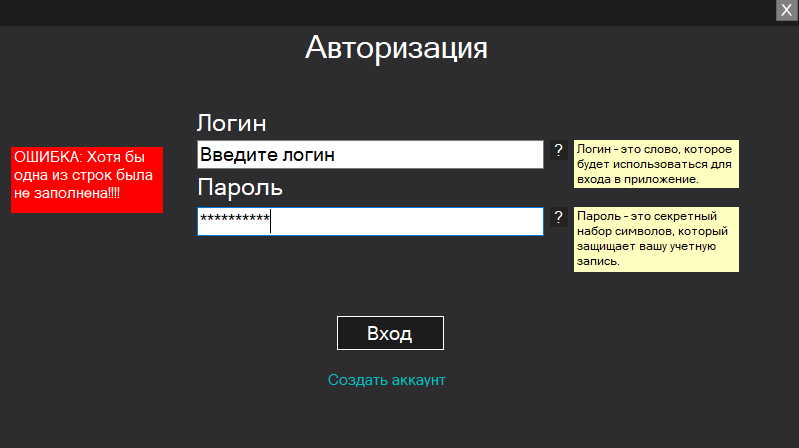


Рис. 3.15 (Реализованное окно авторизации)

После перехода на экран регистрации, на нем мы можем увидеть четыре текстовых поля (для логина, пароля, фамилии и имени), кнопка (для создания нового аккаунта и входа в приложение), а также две подсказки, которые можно свернуть. Кроме того, при введении неверных данных предусмотрен вывод на экран соответствующих ошибок. Также существует кнопка для перехода на экран авторизации.

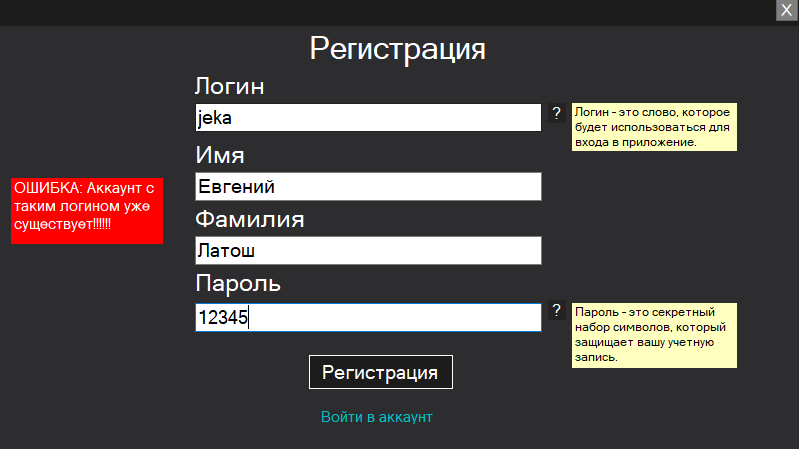


Рис. 3.16 (Реализованное окно регистрации)

После того как пользователь успешно зашел в приложение под своим аккаунтом, его встречает главное окно приложения, открытое на разделе добавления статей расходов. Данное окно разделено на три части (левая часть, правая часть, и часть по середине). Левая и правая части остаются постоянными, а часть по середине меняется от выбранного раздела. В левой части расположены кнопки навигации по разделам приложения, а в правой части отображается информация об аккаунте пользователя, кроме того в ней находятся кнопки (для выхода из аккаунта и удаления аккаунта).

В данном окне существует четыре раздела: “расходы”, “доходы”, “история”, “отчет”. В разделе “расходы” пользователь может посмотреть последние записи расходов, добавить новую категорию для расходов, а также добавить новую запись расходов, предварительно заполнив текстовые поля, находящиеся в нижней части раздела.

Раздел “доходы” идентичен разделу “расходы”, за исключением того, что работа ведется с записями о доходах.

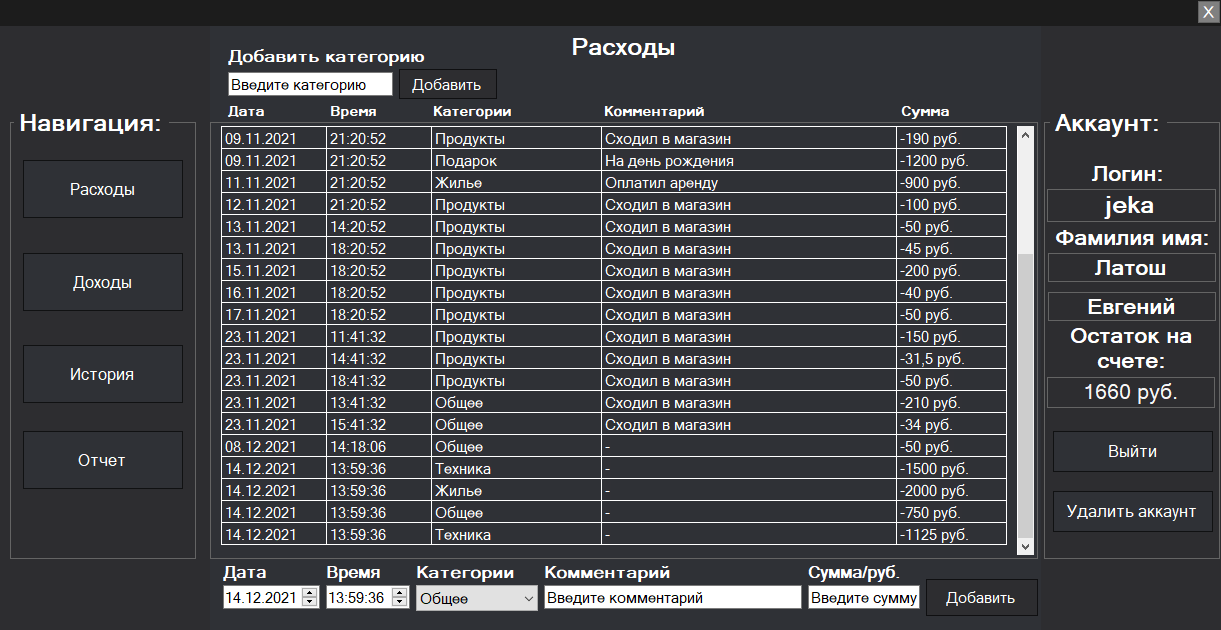


Рис. 3.17 (Главное окно приложения с открытым разделом “расходы”)

В разделах “расходы” и “доходы” отображаются только последние записи, добавленные пользователем. Поэтому возникает проблема, так как пользователь не может посмотреть все записи. Данную проблему решает раздел “история”.

В разделе “история” пользователь может просмотреть записи выбрав определенную дату. Кроме того, пользователь может отфильтровать записи по типу (расходы, доходы), а также отсортировать их по времени или сумме. Также пользователю отображается общая сумма, заработанная/потраченная за день. Данная сумма может меняться в зависимости от выбранного типа записи, например, если выбран тип “только расходы”, то будет отображаться только потраченная сумма.

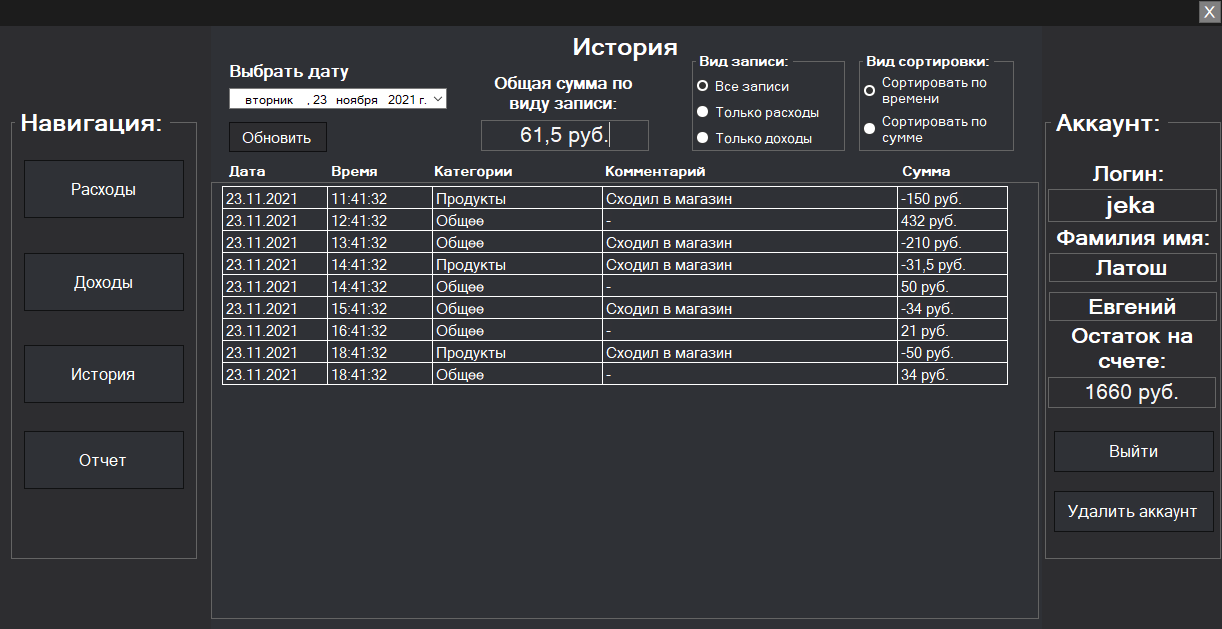


Рис. 3.18 (Главное окно приложения с открытым разделом “история”)

Также реализованное приложение позволяет просматривать отчет за текущий месяц и сравнивать его с предыдущими. Данная функция реализована в разделе “отчет”. В данном разделе можно просмотреть статистику по разным типам записей (расходы, доходы) как в виде таблиц, так и в графическом виде. Также отображается общая сумма, заработанная за месяц.

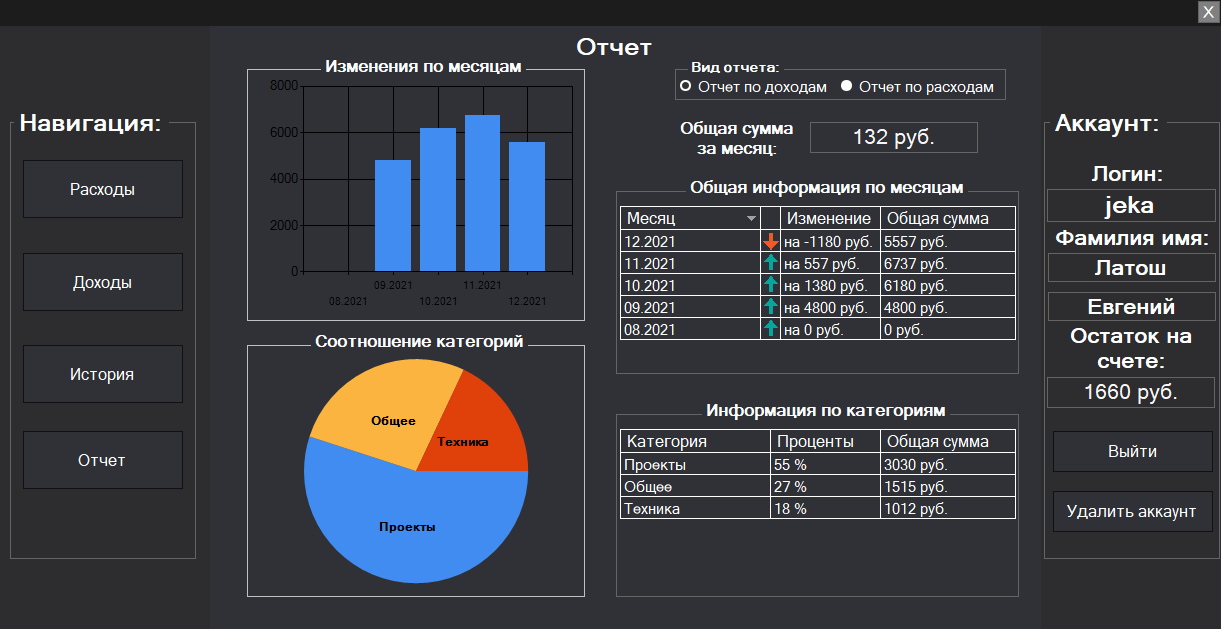


Рис. 3.19 (Главное окно приложения с открытым разделом “отчет”)

## **Вывод по главе 3**

Таким образом мы разобрались с функционалом приложения и его реализацией. В данной главе была описана реализация основных методов данного проекта.

В будущем, проект может представлять, как практическую, так и академическую пользу. Поскольку в случае его удачной реализации, его можно будет развивать дальше, а конечная версия, написанная в максимально простом виде, может в дальнейшем использоваться как шаблон, которым смогут воспользоваться все желающие для того, чтобы ознакомиться с архитектурой приложений на конкретном примере.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы были изучены различные шаблоны проектирования и в результате был выбран подходящий для данного приложения шаблон. На этапе разработки программы были получены практические навыки создания приложения. Благодаря использованию дополнительной учебной литературы, особых трудностей при написании работы не возникло.

Дальнейшую модификацию программы вижу в возможности дополнения ее новыми функциями и возможностями.

В первой главе курсовой работы была доказана актуальность выбранной темы. Также были рассмотрены существующие решения и были выявлены основные аспекты, связанные с реализацией приложения.

Во второй главе была спроектирована архитектура приложения. А также были спроектированы основные методы, использованные в курсовой работе.

В третьей главе была описана программная реализация основных методов, использованных в курсовой работе.

В заключении хочется отметить, что данное приложение можно будет развивать дальше, добавляя новы возможности. Кроме того, реализованные методы могут быть применены не только в данном проекте, их можно использовать в различных приложениях. Также данный проект может использоваться в качестве примера для ознакомиться с архитектурой приложений.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Павловская Т. А. C# программирование на языке высокого уровня / Т. А. Павловская – СПб.: Питер Пресс, 2009. — 432 с.
2. smallbusiness.chron.com [Электронный ресурс]: Важность бухгалтерского программного обеспечения. -URL: <https://smallbusiness.chron.com/importance-accounting-software-71524.html> (дата обращения: 05.11.2021)
3. techradar.com [Электронный ресурс]: /best/best-personal-finance-software. -URL: <https://www.techradar.com/best/best-personal-finance-software> (дата обращения: 10.11.2021)
4. Habr.com [Электронный ресурс]: Паттерны для новичков: MVC vs MVP vs MVVM. -URL: <https://habr.com/ru/post/215605/> (дата обращения: 15.11.2021)