МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет \_\_\_\_\_\_\_\_\_\_\_\_\_Информационных технологий**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_**

Кафедра \_\_\_\_\_\_\_\_\_\_\_\_\_ Программной инженерии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_

Специальность**\_\_\_**1-40 05 01-03 Программное обеспечение информационной безопасности мобильных систем\_\_ \_

Направление специальности**\_\_\_**1-40 05 01-03 Программное обеспечение информационной безопасности мобильных систем\_ \_

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине **«**Объектно-ориентированное программирование»

тема **«**Программное средство самоорганизации по методу agile Катерины Ленгольд и джедайским техникам Макса Дорофеева»

Исполнитель  
студент 2 курса группы 8\_\_\_\_\_\_Пилик Семён Игоревич\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (Ф.И.О.)

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_ст. препод. Чевжик Е.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Курсовой проект защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Председатель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пацей Н.В.\_\_\_\_\_\_\_\_\_\_\_ \_  
 (подпись)

Минск 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет заочный   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В Пацей\_

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование"

|  |  |
| --- | --- |
| Специальность: 1-40 01 05 03 Программное обеспечение и безопасность мобильных систем  Студент:\_\_\_\_Пилик С.И. | Группа:\_\_8\_\_ |
| **Тема:** Программное средство «Финансовый менеджер» | |

**1. Срок сдачи студентом законченной работы**: "24 май 2019 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС должно:

* Составить базу, содержащую статьи расходов пользователя;

**2.2.** Программное средство должно быть выполнено с использованием ООП, взаимодействовать с базой данных. Отображение, бизнес логика и хранилище данных должны быть максимально независимы друг от друга для возможности масштабирования. Язык разработки С#. Управление программой должно быть интуитивно понятным и удобным. К листингам проекта должны быть комментарии.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи (алгоритмы, обзор литературы)
* Разработка архитектуры (концепция, модель, структура, выполняемые функции и взаимосвязь всех компонентов).
* Разработка функциональной модели и модели данных ПС;
* Руководство пользователя;
* Тестирование;
* Заключение;
* Список используемых источников;
* Приложения.

**4. Форма представления выполненной курсовой работы:**

* Теоретическая часть курсового проекта должны быть представлены в формате MS Word. Оформление записки должно быть согласно выданным правилам.
* Необходимые схемы, диаграммы и рисунки допускается делать в MS Office Visio, Rational Rose, VS или копии экрана (интерфейс).
* Листинги программы представляются частично в приложении.
* К записке необходимо приложить CD (DVD), который должен содержать: пояснительную записку, листинги и инсталляцию проекта.

**Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2019 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2019 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2019 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2019 |  |
| 5 | Кодирование программного средства | 23.04.2019 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2019 |  |
| 7 | Оформление пояснительной записки | 7.05.2019 |  |
| 8 | Сдача проекта | 27.05.2019 |  |

**5. Дата выдачи задания \_\_\_\_**21.01.2019**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Е.А. Чевжик*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

# **СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 5](#_Toc9549875)

[**1.** **АНАЛИТИЧЕСКАЯ ЧАСТЬ** 6](#_Toc9549876)

[1.1 Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования. 6](#_Toc9549877)

[1.2 Обзор аналогов 6](#_Toc9549878)

[1.2.1 Mastersoft Money v6.02 7](#_Toc9549879)

[1.2.2 Money Manager 7](#_Toc9549880)

[1.2.3 Домашняя бухгалтерия 8](#_Toc9549881)

[**2.** **АНАЛИЗ И ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ** 9](#_Toc9549882)

[2.1. Проектирование приложения 9](#_Toc9549883)

[2.2. Проектирование структуры базы данных 9](#_Toc9549884)

[2.2.1. Таблицы 9](#_Toc9549885)

[2.2.2. Хранимые Процедуры 12](#_Toc9549886)

[2.2.3. Тестирование БД 13](#_Toc9549887)

[2.3. Структура Классов 14](#_Toc9549888)

[**3.** **РЕАЛИЗАЦИЯ ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ** 15](#_Toc9549889)

[3.1. Окно регистрации 16](#_Toc9549890)

[3.2. Окно входа 16](#_Toc9549891)

[3.3. Главное окно пользователя 16](#_Toc9549892)

[3.3.1. Страница дохода 17](#_Toc9549893)

[3.3.2. Страница расхода 17](#_Toc9549894)

[3.4. Окно истории операций 18](#_Toc9549895)

[3.5. Окно запланированных операций 19](#_Toc9549896)

[3.6. Окно сообщений 19](#_Toc9549897)

[3.7. Информационное окно Creator 20](#_Toc9549898)

[**4.** **ТЕСТИРОВАНИЕ** 20](#_Toc9549899)

[**5.** **ЗАКЛЮЧЕНИЕ** 25](#_Toc9549900)

[**6.** **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 26](#_Toc9549901)

[**7.** **ПРИЛОЖЕНИЕ** 27](#_Toc9549902)

[**Приложение А** 27](#_Toc9549903)

[**Приложение Б** 28](#_Toc9549904)

[**Приложение В** 30](#_Toc9549905)

[**Приложение Д** 31](#_Toc9549906)

# **ВВЕДЕНИЕ**

Среди трендов последних лет очень четко выделяется направление самоорганизации, саморазвития. Приложения для повышения продуктивности занимают все более высокие позиции в рейтингах по количеству скачиваний в PlayMarket и AppStore. Тренд очевиден: современный человек сегодня – это человек, стремящийся контролировать и анализировать свою жизнь. И, безусловно, такой человек не откажется от удобного приложения, которое позволит управлять собственным бюджетом обдуманно.

Моя разработка – программное воплощение метода шести кувшинов. Суть метода следующая: все возможные траты бюджета разделяются по шести категориям, условно называемых кувшинами. Кувшин 1: расходы на самое необходимое (55% от бюджета). В эту сумму включены траты, от которых невозможно отказаться. Это траты на еду, коммунальные услуги, оплату аренды жилья, транспортные расходы. Кувшин 2: расходы на развлечения (10% от бюджета). Эти деньги можно расходовать эмоционально, без глубокого анализа. Важно то, средства должны быть потрачены на товар или услугу, который принесет приятную эмоцию, ощущение, поможет расслабиться и отдохнуть. Кувшин 3: сбережения (10% от бюджета). Это «сокровенный запас», который не должен тратиться, но может использоваться для инвестирования. Иными словами, эта сумма должна стать основой пассивного дохода в будущем. Кувшин 4: расходы на образование (10% от бюджета). Эти деньги могут быть потрачены на оплату учебы в университете, оплаты курсов повышения квалификации, саморазвития, самоанализа. Кувшин 5: резервный фонд (10% от бюджета). Эти сбережения предполагается накапливать на крупную покупку. Автомобили или квартиру, например. Такой запас – гарант уверенности в завтрашнем дне. Кувшин 6: расходы на подарки и благотворительность (5% от бюджета). Эта сумма может быть потрачена на подарок близкому человеку, либо передана в помощь социальному проекту, пожертвована на благотворительность.

Часто планирование бюджета – достаточно личная тема, поэтому в данной разработке особо важно не предоставление данных третьим лицам. В связи с этим выбрана форма программного средства десктопное приложение. Доступ к приложению можно будет получить только с устройства (персональный компьютер, ноутбук), на котором приложение установлено. С этой же целью сервер базы данных разрабатывается локальный.

При разработке будет учитываться тот факт, что пользователи, которые будут взаимодействовать с данным программным средством, не имеют технического образования и владеют компьютером на уровне пользователя. Поэтому важно разработать программное средство с user-friendly интерфейсом.

Совокупность всех приведенных выше положений приведет к созданию качественного и полезного продукта, который будет отвечать требованиям безопасности.

Цели проекта:

* Совершенствование умений написания программного кода;
* Совершенствование умений создания адаптивного дизайна;
* Формирование навыков проектирования баз данных;
* Формирование навыков проектирования и разработки пользовательского интерфейса;
* Формирование умения строить архитектуру проекта, оставляя для него возможность масштабирования.

Задача проекта:

Разработать архитектуру приложения с учетом интересов потенциального потребителя и создать программное средство с user-friendly интерфейсом и максимально сегментированной базой данных. Использовать в разработке объектно-ориентированный язык программирования. Также настроить SQL сервер и создать базу данных, обеспечив простой доступ к ней из приложения.

# **АНАЛИТИЧЕСКАЯ ЧАСТЬ**

## Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования.

При написании данной работы будут использованы научная и учебно-методическая литература, а также статьи популярных информационных веб-сайтов. Основным источником, раскрывающими теоретические и практические основы языка C#, является работа Джеффри Рихтера «CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#». Данный источник подробно описывает внутреннее устройство и функционирование общеязыковой исполняющей среды (CLR) Microsoft .NET Framework версии 4.5.

Основными пользователями могут быть люди возрастной группы 16+ лет.

Международный опыт реализации «десктопных» приложений рассмотрен на основе многочисленных статей различных сайтов. Хочется отметить такие, как: <https://habrahabr.ru/>, <http://codeproject.com/>, а также <https://professorweb.ru/>.

## Обзор аналогов

В мире не существует финансовых менеджеров, основанных на методе 6 кувшинов. Однако существуют финансовые менеджеры основанные на других методах, такие как:

### Mastersoft Money v6.02



Рисунок 1.1 Mastersoft Money v6.02

Это приложение позволяет создавать подробные статистические обзоры ваших расходов и доходов, а также общего баланса за определенный промежуток времени. Вы можете вести информацию по кредитной карточке, счете в банке и других финансовых источниках. Также предусмотрена защита паролем и встроенный конвертор валют.

### Money Manager

Ещё одно функциональное приложение для учёта финансов

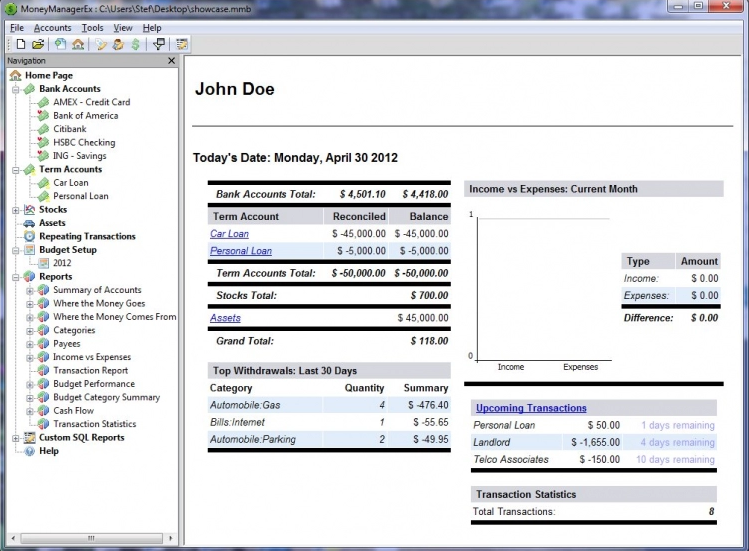


Рисунок 1.2. Money Manager

Оно не только позволяет контролировать доходную и расходную часть, но и предоставляет очень гибкий функционал для работы с внутренней статистикой за любой период времени. Помимо стандартного набора, отличается и такими особенностями, как: возможность работы с данными приложения на ПК; дублирование записей для обеспечения максимальной сохранности данных; планирование бюджета по шаблонам и категориям. Кроме того, программу можно настроить на ведение календаря выплат по кредитным картам, что для многих, безусловно, будет очень полезной функцией.

### Домашняя бухгалтерия

Программа поддерживает многопользовательский режим. Поэтому в одной программе можно вести бюджет отдельных членов семьи. Можно даже вести учет "заначек", поскольку вход каждого пользователя защищается паролем. не настраиваемый, но зато имеет множество уже заданных фильтров: по видам и

Внешний вид программы вы можете увидеть на рисунке 1.3

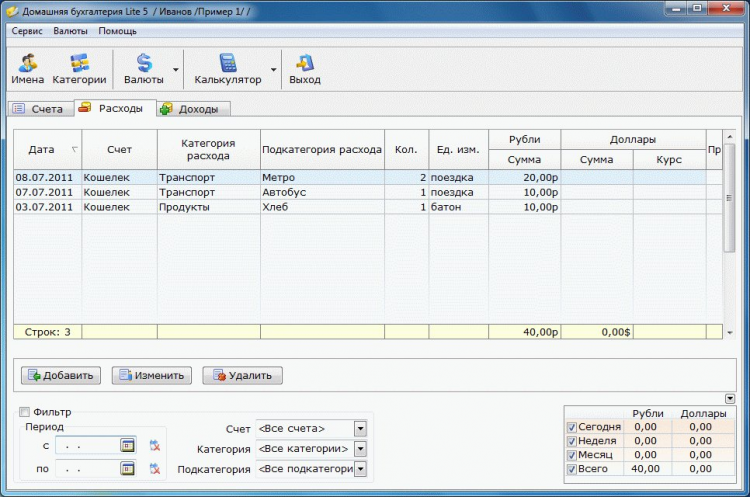


Рисунок 1.3. Money Manager

Ввод новых данных очень удобен: для вставки даты имеется всплывающий календарь, а для ввода суммы - калькулятор. Также калькулятор можно вызвать из панели инструментов для проведения промежуточных расчетов. Заслуживает отдельного внимания генератор отчетов. Хотя в бесплатной версии программы он не настраиваемый, но зато имеет множество уже заданных фильтров: по видам и названиям расходов, видам валюты, периоду отчета (все время, месяц, неделю, день и произвольный период).

"Домашняя бухгалтерия" будет хорошим выбором для тех, кому не надо печатать подробные отчеты о состоянии вашего семейного бюджета, а просто быть в курсе, как обстоят дела с вашими деньгами.

# **АНАЛИЗ И ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ ПРИЛОЖЕНИЯ**

## Проектирование приложения

Данную работу функционально можно разделить на 2 части. Потому что приложение «Save My Money» по сути представляет собой интерфейс для работы пользователя. Интерфейс будет содержать набор элементов управления, как кнопки, текстовые поля и др. Диаграмма последовательности отображена в Приложении В. Блок-схема в Приложении Д.

Варианты использования показаны на рисунке 2.1.

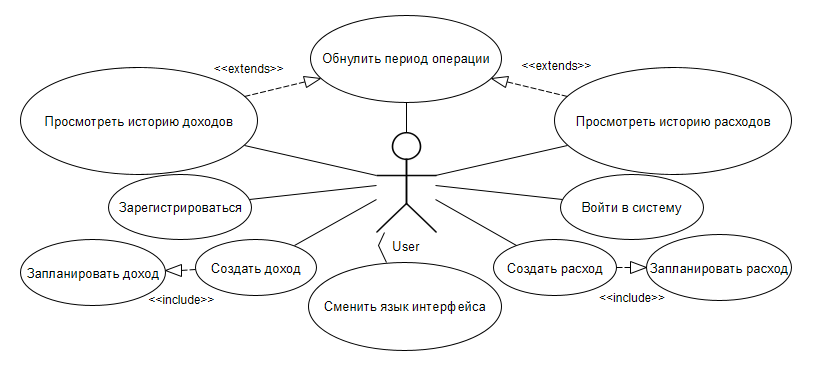


Рисунок 2.1 – UML диаграмма

## Проектирование структуры базы данных

База данных – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ.

### Таблицы

Таблицы являются основой любой базы данных, именно в них хранится вся информация. Для хранения и предоставления данных пользователю была разработана архитектура базы данных, которая содержит в себе 5 таблиц. Диаграмма базы данных представлена на рисунке 2.2.

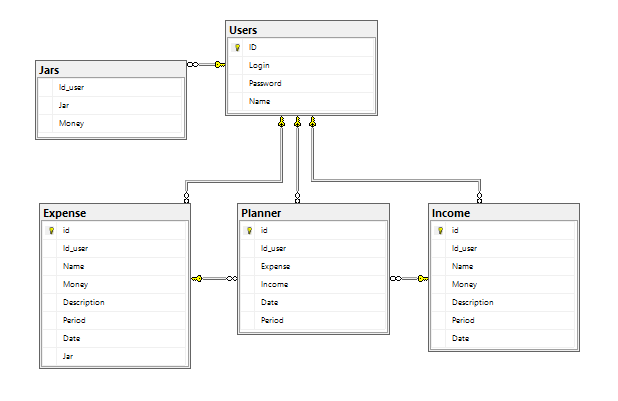


Рисунок 2.2 – Диаграмма базы данных

Таблица *Users –* хранит данные о пользователях.

Таблица *Jars –* хранит данные о кувшинах.

Таблица *Expense–* хранит информацию о расходах.

Таблица *Planner–* хранит информацию о запланированных операциях.

Таблица *Income ­­–* хранит информацию о доходах.

Ниже представлено описание таблиц базы данных с определением типа данных столбцов. Скрипт для создания таблиц, а также зависимостей между ними представлен в *приложении А*.

Таблица *Users* (рисунок 2.3)

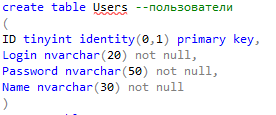


Рисунок 2.3 – Таблица *Users*

Таблица *Jars* (рисунок 2.4)

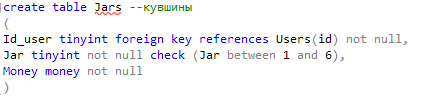


Рисунок 2.4 – Таблица *Jars*

Таблица *Expenses* (рисунок 2.5)

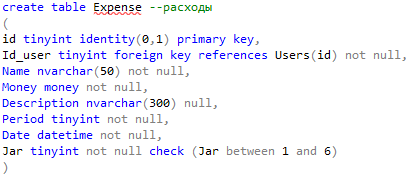


Рисунок 2.5 – Таблица *Expenses*

Таблица *Income* (рис.2.6)

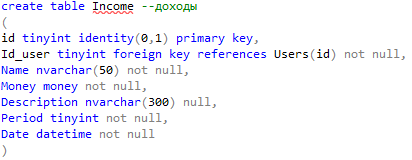


Рисунок 2.6 – Таблица *Income*

Таблица *Planner* (рис.2.7)

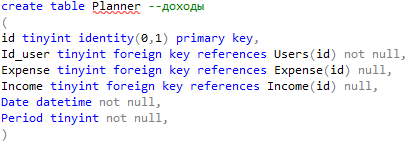


Рисунок 2.7 – Таблица *Planner*

### Хранимые Процедуры

Использование хранимых процедур позволяет ограничить или вообще исключить непосредственный доступ пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным. Кроме того, некоторые СУБД поддерживают шифрование текста (wrapping) хранимой процедуры.

В рамках данного курсового проекта было разработано 68 процедур. Данные процедуры реализуют возможность взаимодействия с базой данной. Большинство операций с данными так или иначе представляют собой CRUD операции.

**CRUD** (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») — акроним, обозначающий четыре базовые функции, используемые при работе с хранилищами данных:

* создание;
* чтение;
* редактирование;
* удаление.

В качестве примера рассмотрим рисунок 2.8

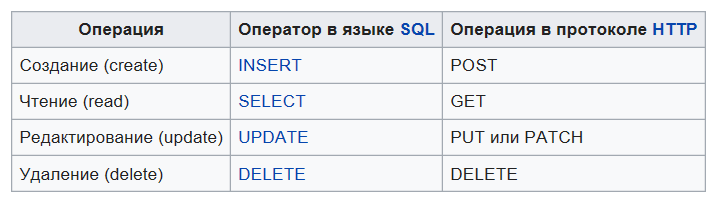


Рисунок 2.8 – CRUD

Скрипт для создания всех процедур предоставлен в *приложении В*.

В рамках данного курсового проекта было разработано 14 процедур.

**Процедуры добавления данных в базу:**

*addExpense* – процедура добавления расхода;

*addIncome* – процедура добавления дохода;

*addJars* – процедура добавления кувшинов пользователя для последующего обновления;

*addUser* – процедура добавления пользователя;

**Процедуры получения данных из базы:**

*GetPlanner* – процедура получения запланированных доходов и рассходов;

*GetIncome* – процедура получения дохода;

*GetExpense* – процедура получения расхода;

*GetSelectedJar* – процедура получения суммы в кувшине;

*GetLastId* – процедура получения последнего элемента в расходах;

*GetUser* – процедура получения пользователя;

*GetId –* процедура получения id по логину;

**Процедуры обновления данных в базе:**

*UpdatePeriod –* процедура обновления операция в зависимости от периода;

*UpdateJarsExpense –* процедура обновления суммы расход;

*UpdateJarsIncome –* процедура обновления суммы доход;

### Тестирование БД

Тестирование происходило с помощью утилиты с открытым исходным кодом – *SQLQueryStress.* Данная утилита позволяет задавать количество повторов запросов, а также количество потоков (клиентов). Пример работы программы представлен на рисунке 2.9

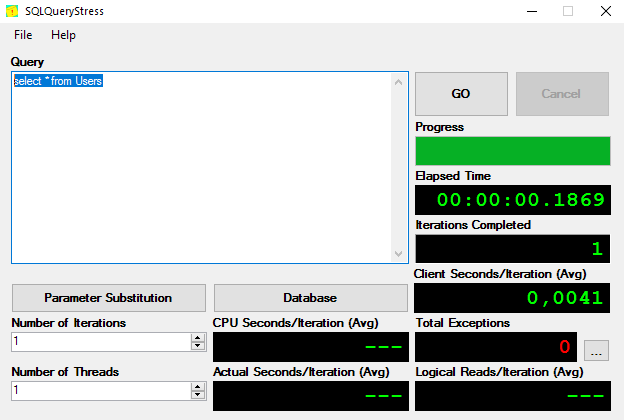


Рисунок 2.9 – Пример работы утилиты *SQLQueryStres*

Наиболее ресурсозатратной оказалась процедура вывода всех регистраций животных - UpdatePeriod, поскольку она обращается к 4 связанным таблицам, а также производит расчёт данных. Процесс её тестирования показан на рисунке 2.10.

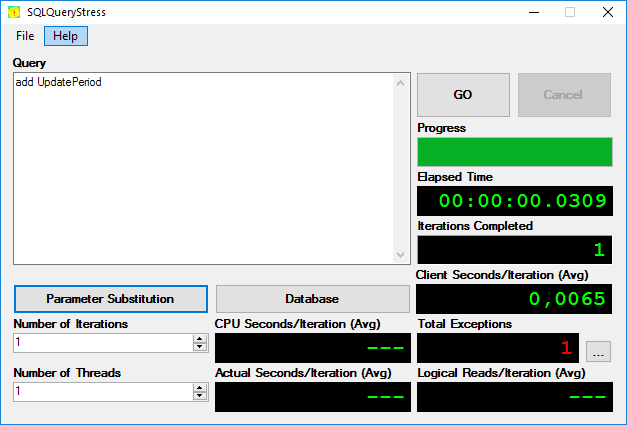


Рисунок 2.10 – Тестирование БД

В ходе тестирования не было выявлено каких-либо замедлений в обработке данных, даже не смотря на большом количестве потоков и итераций. Вследствие чего необходимость в индексе отпала сама собой. Результаты можно улучшить путем улучшения характеристик самого сервера, т.к. данное тестирование проводилось на обычной пользовательской машине, где производительность ниже, чем у сервера.

## Структура Классов

Структура проекта – “скелет” любой программы. Он состоит из различных элементов. Это могут быть ресурсы, статические классы, формы и прочее. Диаграмма классов отображена на рисунке 2.11

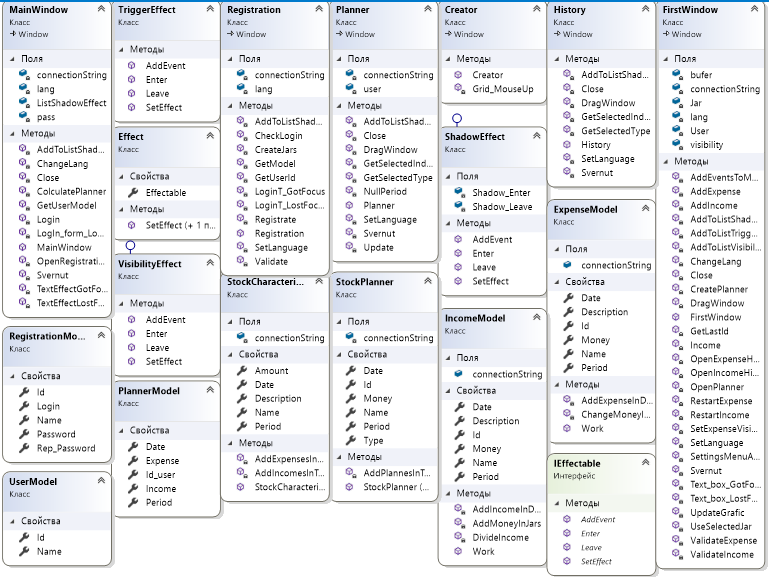


Рисунок 2.11 – Диаграмма классов

# **РЕАЛИЗАЦИЯ ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ**

Приложение состоит из 7 окон:

* Окно регистрации;
* Окно входа;
* Главное окно пользователя;
* Окно истории Операций;
* Окно запланированных операций;
* Окно сообщений;
* Информационное окно Creator.

Все окна в приложении адаптируются на все возможные размеры экранов. Дизайн окон и элементов управления разработан мной. Пользователю предоставлена возможность выбрать 2 языка интерфейса (Русский, Английский).

## Окно регистрации

При заполнении окна регистрации мы вводим информацию о пользователе и пароль. Данные проходят валидацию и заносятся в базу данных.

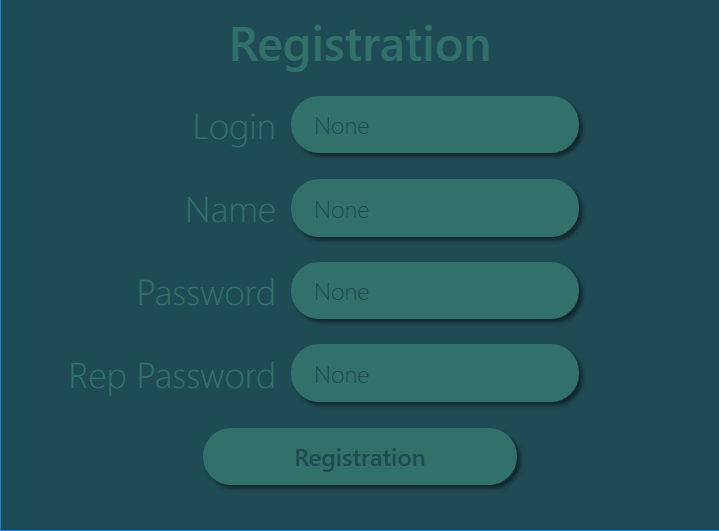


Рисунок 3.1 – Окно Регистрации

## Окно входа

Данная форма предоставляет возможность выбрать пользователя. Также предоставляет возможность переключить язык (нажатие на кнопку “Language”) и перейти на форму регистрации осмотренную ранее. Для открытия окна Мастера требуется ввести «master» в графы логина и пароля.

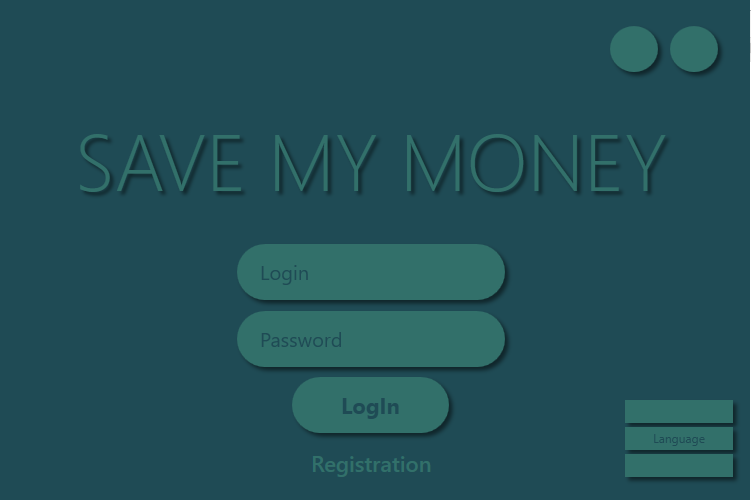


Рисунок 3.2 – Окно Входа

## Главное окно пользователя

Главное окно состоит из 2 страниц:

* Страница дохода;
* Страница расхода.

Также присутствует меню перехода по этим страницам (в левой части экрана), и кнопки для выхода из приложения, сворачивания окна, открытия подменю настроек (Рисунок 3.3), выход в окно входа.

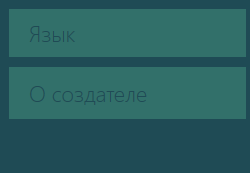


Рисунок 3.3 – подменю настроек

Далее рассмотрим каждую страницу по отдельности.

### Страница дохода

Страница дохода состоит из четырёх полей для ввода и двух кнопок. При нажатии на кнопку «Add» или «Добавить» в зависимости от выбранного пользователем языка производится валидация введённых данных в поля и занесение их в БД. При нажатии на кнопку «History» отрывается окно Истории доходов пользователя.

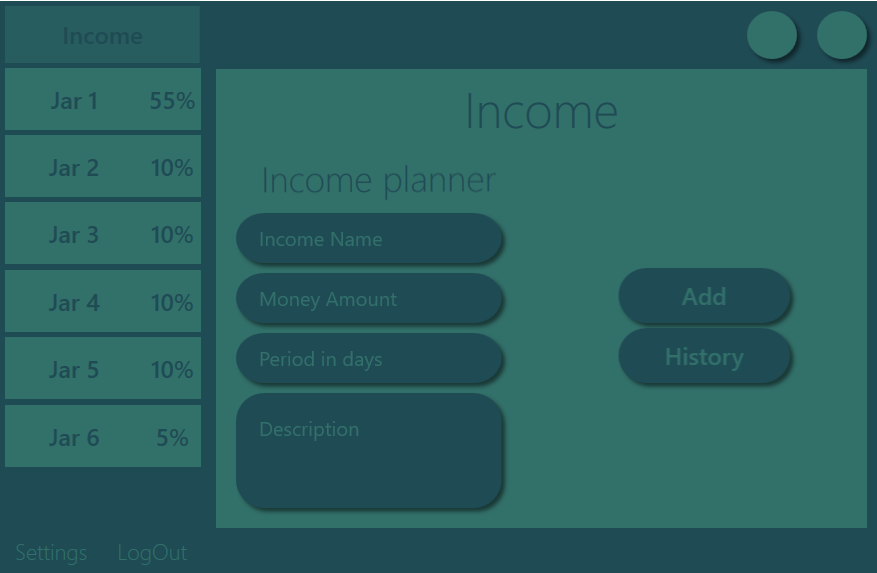


Рисунок 3.4– Главное окно пользователя. Страница дохода

### Страница расхода

Страница расхода состоит из четырёх полей для ввода, трёх кнопок, и графика. При нажатии на кнопку «добавить» происходит валидация данных в полях и занесение их в БД. При нажатии на кнопку «планы» открывается окно запланированных расходов и доходов. Кнопка «история» открывает окно истории расходов. На рисунке 3.5 можно увидеть, как работает график. Число сверху – количество денег на счету текущего кувшина. При вводе суммы расхода, она мгновенно отображается на графике (значение снизу), при превышении суммой расхода, суммы в кувшине, значение расхода подсвечивается красным, тем самым информируя пользователя о превышении бюджета (Рисунок 3.6)

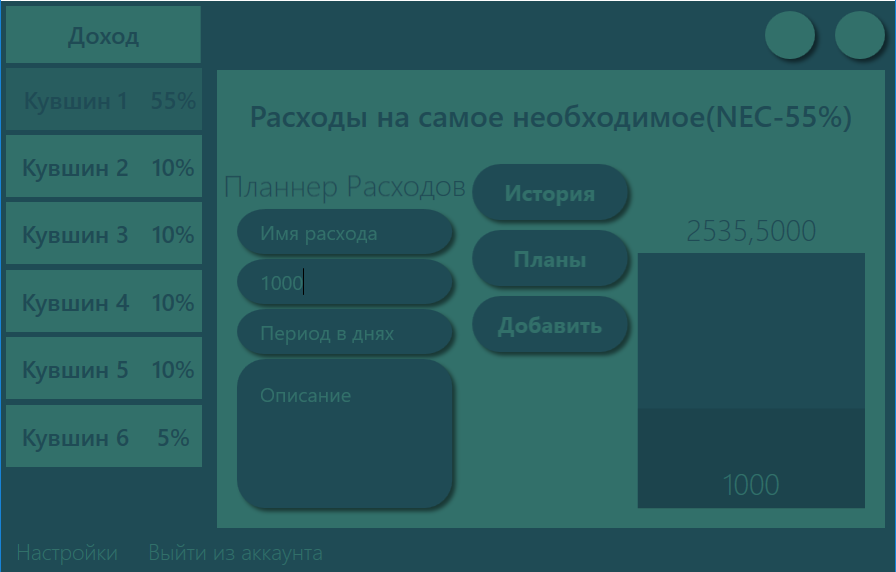


Рисунок 3.5 – Главное окно пользователя. Страница расхода

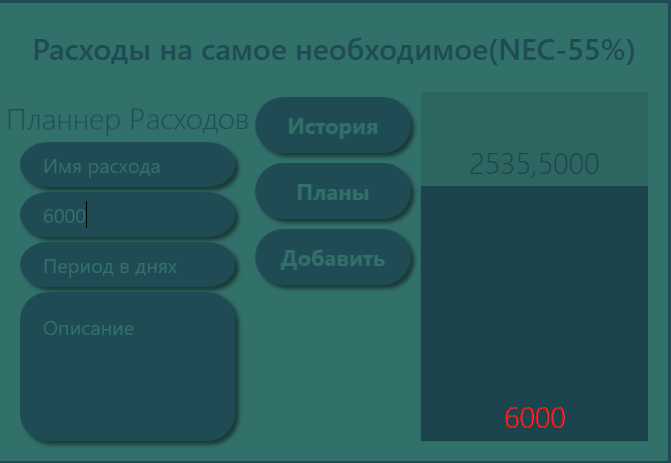


Рисунок 3.6 – Срабатывание графика

## Окно истории операций

Контент в окне истории операций зависит от страницы запуска. Если пользователь запустит окно через страницу расходов, то выведется история расходов, если через страницу доходов, то доходов. Окно состоит из таблицы и кнопки. Строки в таблице можно выделять, и при нажатии на кнопку, удалять из БД.

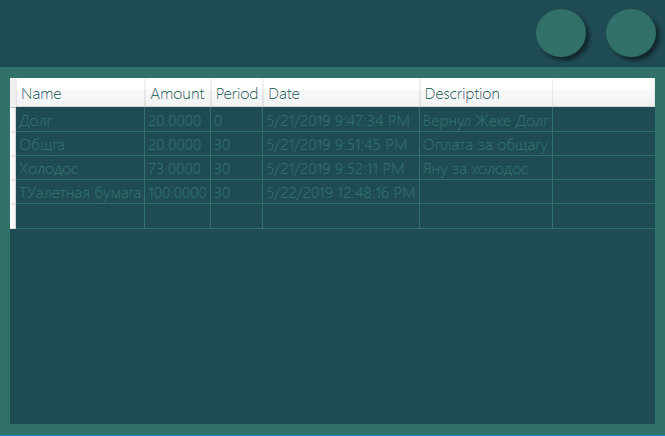


Рисунок 3.4 – Окно истории операций

## Окно запланированных операций

Окно запланированных операция включает в себя таблицу и две кнопки. В нём вы можете просмотреть все операции, которые имеют не нулевой период. И обнулить его, если потребуется.

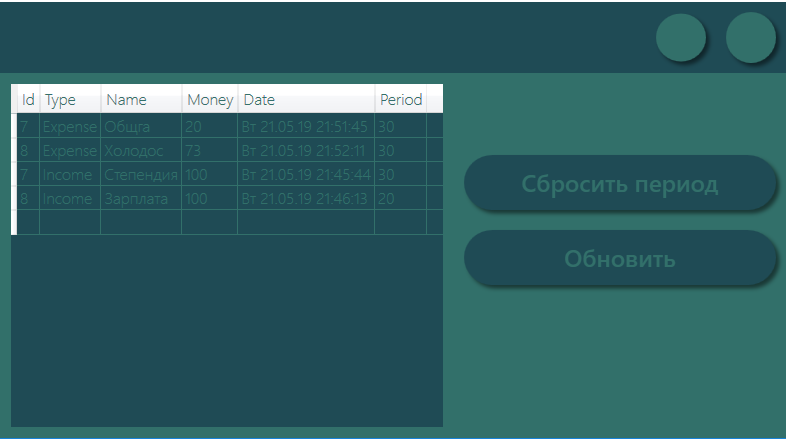


Рисунок 3.5 – Окно запланированных операций

## Окно сообщений

Это стилизованный Message Box, который выводит подсказки пользователю. И используется для вывода ошибок, совершенных пользователем. И используется для валидации введённых данных.

Окно состоит из одного текстблока и закрывается при нажатии на любое место в окне.

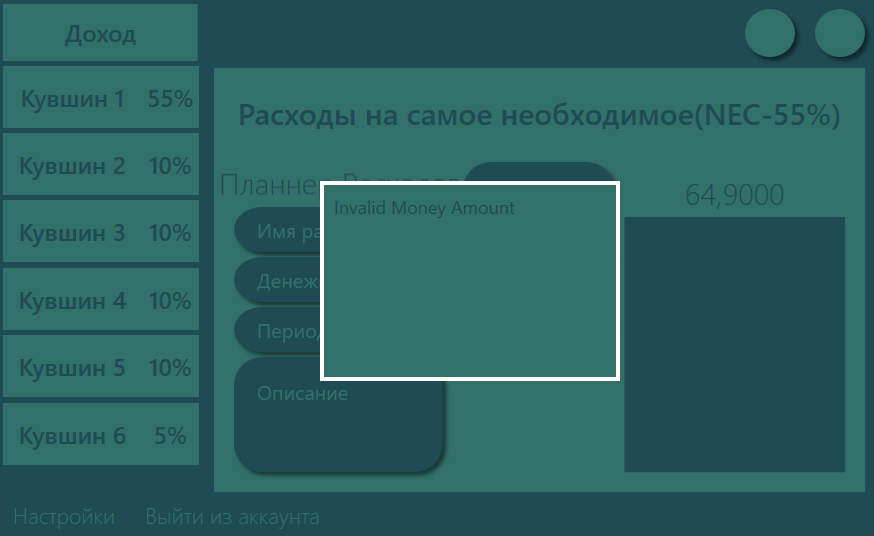


Рисунок 3.6 – Окно сообщений

## Информационное окно Creator

Это окно с информацией о создателе приложения.



Рисунок 3.7 – Окно Creator

# **ТЕСТИРОВАНИЕ**

Для корректной работы программы необходимо обеспечить защиту работы пользователя от ошибок и сбоев. Для этого используются валидаторы. Они служат для “отлавливания” исключений с посредствующей их обработкой. Это необходимо, чтобы при вводе некорректной информации, удаление несуществующий данных, изменение данных на непредполагаемые типы данных, программа не получала сбоев.

Предусмотрена структурированная обработка следующих исключительных ситуаций:

* При регистрации поле «Логин» не должно совпадать с значениями «Логин» в БД (Рисунок 4.1);
* В пароле должно быть от 5 символов (Рисунок 4.2);
* Пароли должны совпадать (Рисунок 4.3);
* Поля должны быть заполнены (Рисунок 4.4).

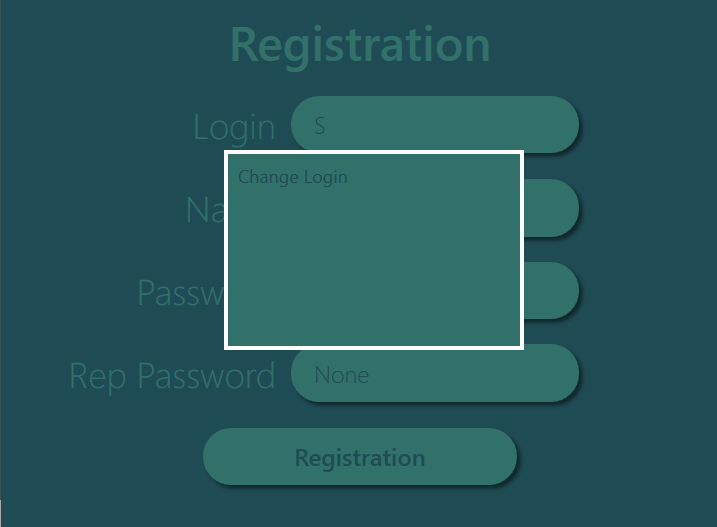


Рисунок 4.1 – Ошибка 1

Сообщение о совершенной пользователем ошибке появляется в центре экрана в окне сообщений, описанном ранее.

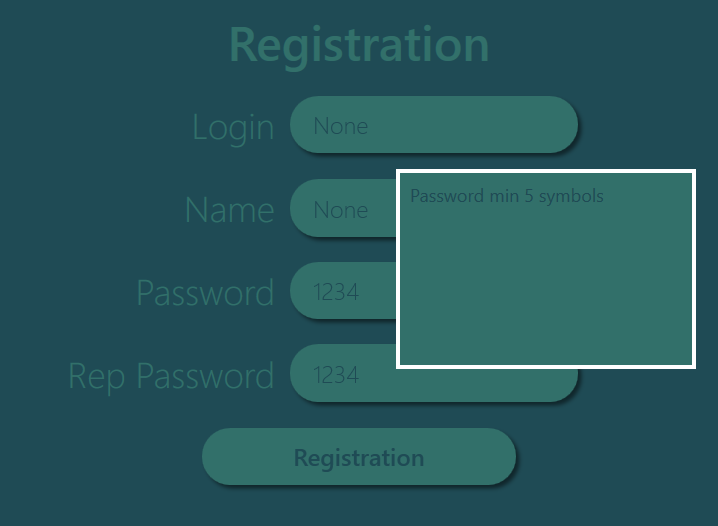


Рисунок 4.2 – Ошибка 2

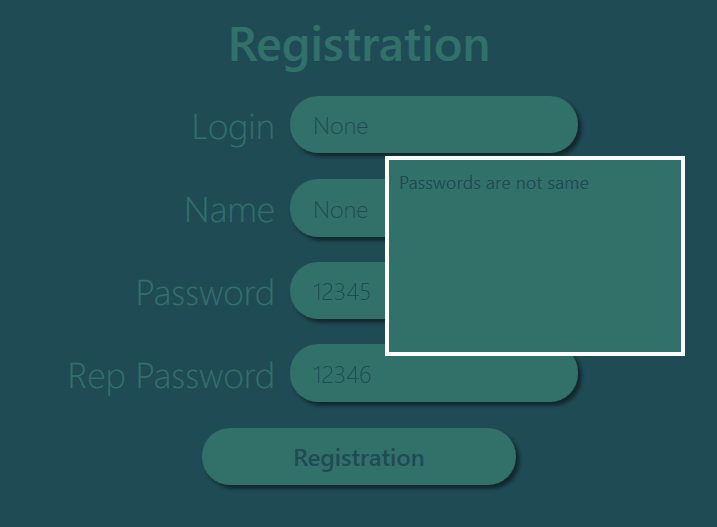


Рисунок 4.3 – Ошибка 3

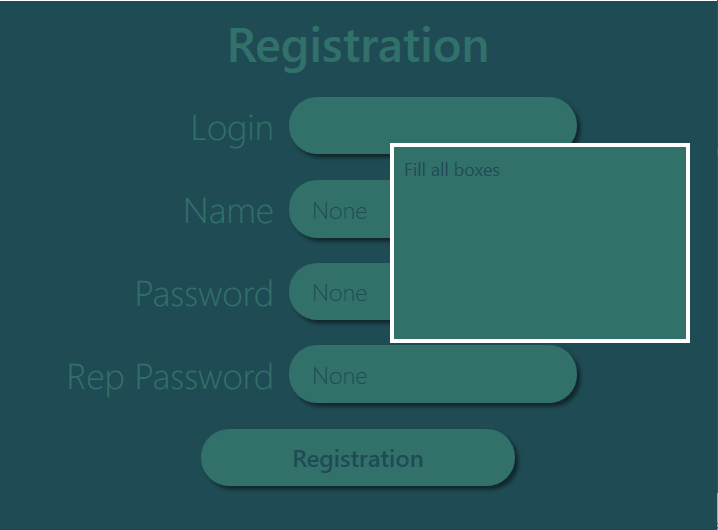


Рисунок 4.4 – Ошибка 4

В окне авторизации сравниваются значения, введённые в поля с базой данных. Если они не совпадают, то выдаётся сообщение об ошибке, представленное на рисунке 4.5.



Рисунок 4.5 – Ошибка Входа

В главном окне при вводе в графу «Денежная сумма» символов алфавита и знаков препинания (кроме «.» и «,») выскакивает сообщение, которое вы можете увидеть на рисунке 4.6

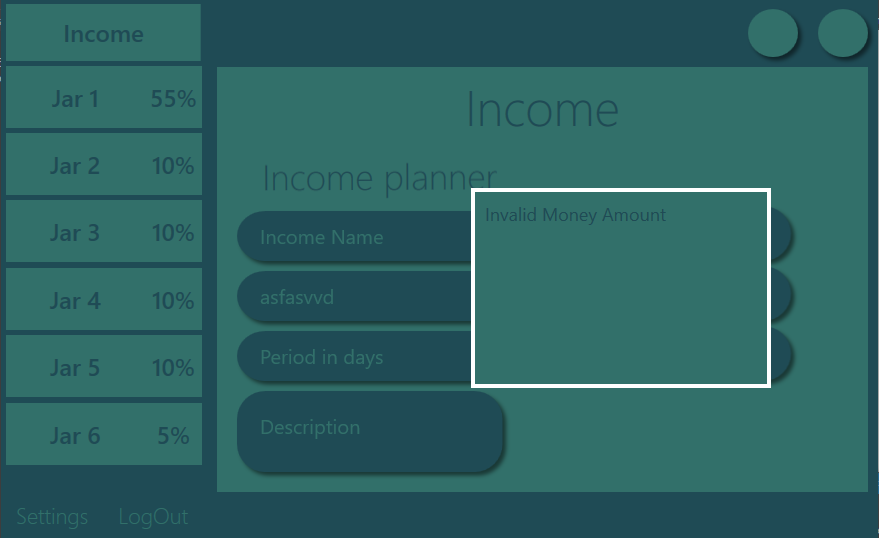


Рисунок 4.6 – Ошибка ввода суммы

При открытии окон истории или планера выскакивает ошибка (Рисунок 4.7) если в БД отсутствуют строки.

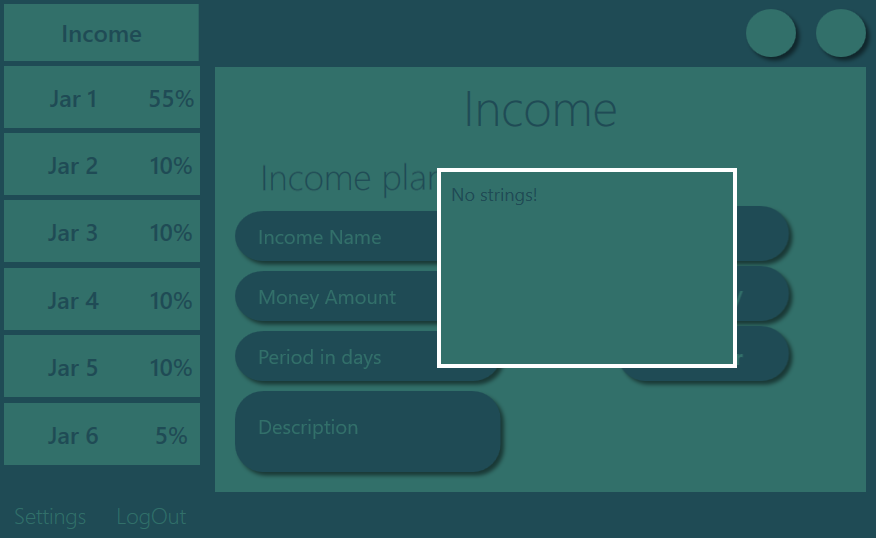


Рисунок 4.7 – Ошибка запуска истории операций

# **ЗАКЛЮЧЕНИЕ**

Данный проект был разработан для многочисленной аудитории. Главной задачей было сделать простое и удобное приложение с дружелюбным интерфейсом. Также был проработан ряд исключительных ситуаций с выводом сообщений, в случае некорректной работы пользователя.

Это приложение – первый опыт в написании такого рода программ, если не считать лабораторные работы. Я считаю, что я справился с поставленной задачей. Конечно, функционала в аналогах больше, но в данном проекте я выступал в роли дизайнера, программиста, тестировщика и т.д. Поэтому я считаю нормальным выполненный объём работы за данный промежуток времени.

Я подтянул свои знания как в программировании, так и в дизайне. Много сил было потрачено на адаптацию интерфейса, правильное конструирование кода и базы данных.

Приложение создано с возможностью масштабирования.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Э– Режим доступа: http://msdn.microsoft.com/library/rus/ -- Дата доступа:14.03.2015.

2. Петцольд Ч Программирование для Microsoft Windows на С#. В 2 томах

М.: Русская редакция – 2002

3. Разработка Windows-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# .NET. Учебный курс СПб.: Русская Редакция. – 512 с

4. Рихтер Д Программирование на платформе Microsoft .NET Framework

СПб.: Русская редакция – 2005. – 512 с

5. Шилдт Г Полный справочник по C# Вильямс – 2004 – 752 с

6. Фролов А., Фролов Г Язык C#. Самоучитель – 2002 – 560 с.

7. Троелсен, Э.А. Язык программирования C# 2010 и платформа .NET 4. /

Э.А. Троелсен. – М,2010. – 300 с.

8. Хейлсберг, А.Б. Язык программирования C#. / А.Б. Хейлсберг. – 216 с.

# **ПРИЛОЖЕНИЕ**

## **Приложение А**

**create table Users --пользователи**

**(**

**ID tinyint identity(0,1) primary key,**

**Login nvarchar(20) not null,**

**Password nvarchar(50) not null,**

**Name nvarchar(30) not null**

**)**

**create table Jars --кувшины**

(

Id\_user tinyint foreign key references Users(id) not null,

Jar tinyint not null check (Jar between 1 and 6),

Money money not null

)

create table Expense --расходы

(

id tinyint identity(0,1) primary key,

Id\_user tinyint foreign key references Users(id) not null,

Name nvarchar(50) not null,

Money money not null,

Description nvarchar(300) null,

Period tinyint not null,

Date datetime not null,

Jar tinyint not null check (Jar between 1 and 6)

)

create table Income --доходы

(

id tinyint identity(0,1) primary key,

Id\_user tinyint foreign key references Users(id) not null,

Name nvarchar(50) not null,

Money money not null,

Description nvarchar(300) null,

Period tinyint not null,

Date datetime not null

)

create table Planner --доходы

(id tinyint identity(0,1) primary key,

Id\_user tinyint foreign key references Users(id) not null,

Expense tinyint foreign key references Expense(id) null,

Income tinyint foreign key references Income(id) null,

Date datetime not null,

Period tinyint not null)

## **Приложение Б**

CREATE PROCEDURE UpdatePeriod as

begin

Declare @b int = 0, @c, @d, @e, @g int,

@a date,

@f money,

while @b<=(Select Max(id) from Planner)

begin

if (Select Period from Planner where id=@b) !=0

begin

set @a = (Select Date from Planner where id=@b)

set @c = (DATEDIFF(day,@a,GETDATE())/(Select Period from Planner where id=@b))

end;

if (Select Expense from Planner where id=@b) is not null

begin

set @d = (Select Expense from Planner where id=@b)

set @e = (Select Jar from Expense where id = @d)

set @f = (Select Money from Expense where id = @d)

set @g = (Select Id\_user from Expense where id = @d)

UPDATE JARS set Money = money - @f\*@c where Id\_user=@g and Jar = @e;

UPDATE Planner set Date = GETDATE() where id = @b

end;

else

begin

set @d = (Select Income from Planner where id= @b)

set @e = (Select Money from Income where id = @d)

set @g = (Select Id\_user from Income where id = @d)

Update Jars set Money = money + ((@e\*@c)/100)\*55 where Id\_user = @g and Jar = 1

Update Jars set Money = money + ((@e\*@c)/100)\*10 where Id\_user = @g and Jar = 2

Update Jars set Money = money + ((@e\*@c)/100)\*10 where Id\_user = @g and Jar = 3

Update Jars set Money = money + ((@e\*@c)/100)\*10 where Id\_user = @g and Jar = 4

Update Jars set Money = money + ((@e\*@c)/100)\*10 where Id\_user = @g and Jar = 5

Update Jars set Money = money + ((@e\*@c)/100)\*5 where Id\_user = @g and Jar = 6

UPDATE Planner set Date = GETDATE() where id = @b

end

set @b = @b+1;

end

end;

go

Create Procedure GetPlanner @id tinyint as

begin

Select id,'Expense'as Type,Name,Money,Date,Period from Expense where Id\_user = @id and Period !=0

union

Select id,'Income'as Type,Name,Money,Date,Period from Income where Id\_user = @id and Period !=0

end

go

Create Procedure AddExpense @id tinyint,@name nvarchar(50),@money money,@desc nvarchar(50),@Period tinyint,@Jar tinyint as

INSERT INTO Expense (Id\_user,Name,Money,Description,Period,Date,Jar) values(@id,@name,@money,@desc,@Period,GETDATE(),@Jar);

go

Create Procedure AddIncome @id tinyint,@name nvarchar(50),@money money,@desc nvarchar(50),@Period tinyint as

INSERT INTO Income(Id\_user,Name,Money,Description,Period,Date) values(@id,@name,@money,@desc,@Period,GETDATE());

go

Create Procedure UpdateJarsExpense @id tinyint,@Money money,@Jar tinyint as

Update jars set Money = Money-@Money where Id\_User = @id and Jar = @Jar

go

Create Procedure UpdateJarsIncome @id tinyint,@Money money,@Jar tinyint as

Update jars set Money = Money+@Money where Id\_User = @id and Jar = @Jar

go

Create Procedure GetIncome @id tinyint as

SELECT \* FROM Income where Id\_user= @id

go

Create Procedure GetExpense @id tinyint as

SELECT \* FROM Expense where Id\_user= @id

go

Create Procedure GetSelectedJar @id tinyint, @jar int as

Select \* from Jars where Id\_user=@id and jar=@jar

go

Create Procedure GetLastId as

Select Max(Id) from Expense

go

Create Procedure GetUser @login nvarchar(50), @password nvarchar(50) as

Select id,name from Users where Login = @login and Password = @password

go

Create Procedure CheckLogin @login nvarchar(50) as

Select Login from Users where Login = @login

go

Create Procedure GetId @login nvarchar(50) as

Select ID from Users where Login = @login

go

Create Procedure AddJars @id tinyint, @jar int as

INSERT INTO Jars (Id\_user,Jar,Money) VALUES (@id,@jar,'0')

## **Приложение В**

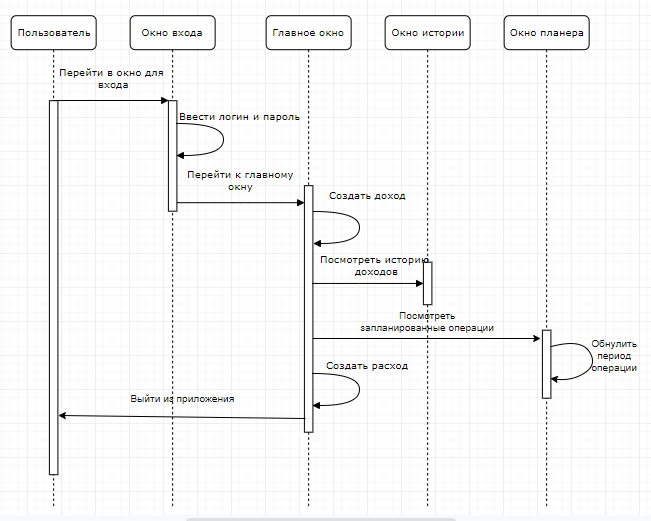
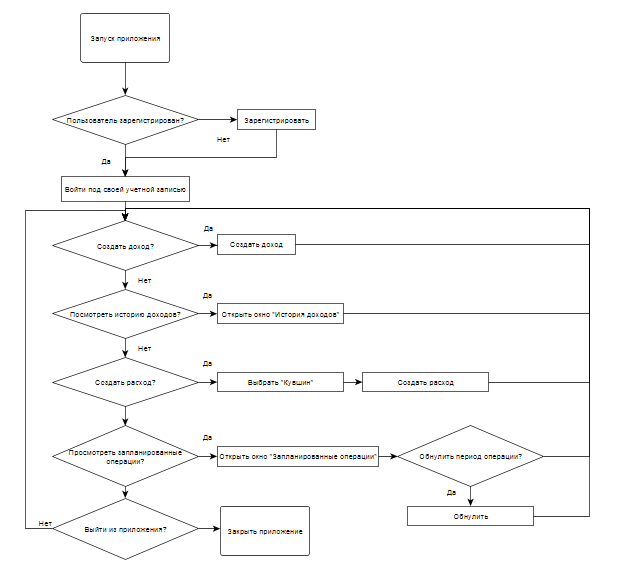


Диаграмма последовательности

## **Приложение Д**



Блок-схема приложения