МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Кафедра информационные технологии управления

Курсовая работа по курсу  
«Технологии программирования»

Мобильное приложение «Мои расходы»

Выполнили: студенты 3 курса, группы 3.2

Добрынина Е.А., Немчанинова Ю.Е.

Руководитель: Полещук Х. А.

Воронеж 2019

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc10128324)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc10128325)

[2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc10128326)

[2.1. Глоссарий 5](#_Toc10128327)

[2.2. Анализ существующих решений 5](#_Toc10128328)

[2.2.1. Expense Manager 5](#_Toc10128329)

[2.2.2. CoinKeeper 6](#_Toc10128330)

[2.2.3. EasyMoney 6](#_Toc10128331)

[2.3. Анализ средств реализации 7](#_Toc10128332)

[2.4. UML диаграммы 8](#_Toc10128333)

[2.4.1. Диаграмма вариантов использования 8](#_Toc10128334)

[2.4.2. Диаграмма последовательности 9](#_Toc10128335)

[2.4.3. Диаграмма коммуникаций 11](#_Toc10128336)

[2.4.4. Диаграмма состояний 13](#_Toc10128337)

[2.4.5. Диаграмма классов 14](#_Toc10128338)

[2.4.6. Диаграмма развертывания 16](#_Toc10128339)

[2.4.7. Диаграмма объектов 17](#_Toc10128340)

[2.4.8. Диаграмма деятельности 17](#_Toc10128341)

[2.5. Схема базы данных 20](#_Toc10128342)

[3. ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ СИСТЕМЕ 21](#_Toc10128343)

[4. КРИТЕРИИ ПРИЕМКИ 22](#_Toc10128344)

[5. ПЛАНИРОВАНИЕ РАБОТ 25](#_Toc10128345)

[5.1. Выбор методологии проектирования 25](#_Toc10128346)

[5.2. Виды работ, которые необходимо выполнить в процессе разработки программного средства 25](#_Toc10128347)

[5.3. Состав команды, распределение задач по участникам и по времени 26](#_Toc10128348)

[6. РЕАЛИЗАЦИЯ 28](#_Toc10128349)

[6.1. Использованные средства реализации 28](#_Toc10128350)

[7. ТЕСТИРОВАНИЕ 28](#_Toc10128351)

[8. ИТОГИ РАБОТЫ 28](#_Toc10128352)

[ЗАКЛЮЧЕНИЕ 29](#_Toc10128353)

[СПИСОК ЛИТЕРАТУРЫ 30](#_Toc10128354)

# ВВЕДЕНИЕ

В настоящее время каждому человеку приходится осуществлять большое количество финансовых операций. Поэтому становится очевидной необходимость ведения личной бухгалтерии. С ее помощью возможно увидеть уязвимости в личном бюджете, которые до этого на глаза не попадались. Но главное — благодаря личной бухгалтерии будет возможность найти простые способы для оптимизации вашего бюджета.

Существует множество способов ведения личной бухгалтерии, к примеру, записи в блокнот или онлайн-сервисы. Тем не менее, наиболее предпочтительным является мобильное приложение. В современном мире смартфон находится рядом с владельцем в течение всего дня, поэтому создавать новые записи о расходах, просматривать отчеты и анализ финансов становится очень быстро и удобно, все записи находятся «под рукой», нет привязки к компьютеру.

Но большинство существующих мобильных приложений не обладает интуитивно понятным интерфейсом, необходимыми пользователю функциями, не являются легковесными и удобными, а поэтому не предпочтительны. Кроме того, к сожалению, эргономичные, удобные, простые приложения, предоставляющие пользователю широкий спектр возможностей, являются платными, наполнены рекламой или предоставляют бесплатно весьма ограниченный функционал.

В данной курсовой работе рассматривается проблема создания простого, легковесного и не нагруженного излишней функциональностью, но обладающего всеми необходимыми функциями мобильного приложения «Мои расходы».

# ПОСТАНОВКА ЗАДАЧИ

Цель курсовой работы - реализовать клиент-серверное Android-приложение для отслеживания личных расходов.

Приложение должно позволять пользователю:

1. Создавать, изменять, удалять категории расходов

2. Определять порог расходов для каждой категории

3. Создавать, изменять, удалять расходы по категориям

4. Получать уведомления о превышении порога по категории

5. Получать отчет о расходах в виде круговой диаграммы

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 2.1. Глоссарий

*Расходы* — денежные средства, потраченные пользователем

на какую-либо группу товаров или услуг.

*Категория* — группа товаров или услуг.

*Порог* — максимальная сумма денег, которую намерен потратить пользователь по категории за месяц.

## 2.2. Анализ существующих решений

Рассмотрим существующие решения.

### 2.2.1. Expense Manager

Приложений с подобным названием в данный момент в Google Play несколько, но именно приложение разработанное компанией Bishinews пользуется популярностью. Об этом можно судить по счетчику скачиваний в Google Play, который совсем недавно достиг отметки в 5 000 000 скачиваний.

Функционал приложения огромен, но самым главным является планировка своих финансов. Однако разработчики не наделили программу русским интерфейсом. При первом входе в приложение будет показан шаблон с несколькими пунктами расходов, которые при желании можно изменить, перейдя в настройки. Так же можно очень легко просматривать необходимые платежи или же взглянуть на подробный и наглядный график расходов.

Достоинства:

* много полезных функций;
* просмотр отчетов в виде графиков;

Недостатки:

* ограниченный функционал бесплатной версии;
* отсутствие русского языка.

### 2.2.2. CoinKeeper

Одно из самых популярных приложений для ведения учета расходов и доходов прямо на смартфоне или планшете на Android. После первого запуска приложение предложит выбрать наиболее подходящий способ планирования доходов: автоматический (в этом случае необходимо указать сумму ежемесячного дохода) или же ручной (более тонкая настройка каждого из параметров). Приложение распространяется в Google Play на условно-бесплатной основе, что означает бесплатное использование всего функционала в течение 15 дней, а в дальнейшем при желании продолжать работу с приложением необходимо приобрести полную версию.

Достоинства:

* любые операции можно провести перетягиванием одной иконки на другую;
* подсказки;
* автоматическое планирование бюджета;
* синхронизация с другими устройствами;
* возможность установки пароля.

Недостатки:

* невозможность представления отчета в виде графика;
* медленная анимация, которая тормозит даже на современных устройствах.

### 2.2.3. EasyMoney

Дорогое и самое функциональное приложение из данной категории приложений. Но, судя по количеству скачиваний в Google Play, пользователей цена приложения вовсе не останавливает, тем более что есть бесплатная версия программы и можно протестировать начальный функционал на ней, а уже потом решить стоит ли тратиться на данную программу или же нет.

Достоинства:

* огромный функционал;
* наглядные отчеты в виде интерактивных графиков;
* наличие удобного виджета.

Недостатки:

* не совсем понятный интерфейс;
* отсутствие русского языка;
* высокая стоимость за полную версию программы.

## 2.3. Анализ средств реализации

Ниже представлены технологии, которые используются для реализации клиент-серверного мобильного приложения для отслеживания личных расходов:

* Android SDK - набор компонентов и средств, необходимых для разработки приложений на Android, обладающий широкими функциональными возможностями, позволяющими запускать тестирование и отладку исходных кодов, оценивать работу приложения в режиме совместимости с различными версиями ОС Android и наблюдать результат в реальном времени.
* Язык реализации клиентской части приложения - Java.
* Серверная часть приложения реализована на языке Python с использованием инструмента FastAPI - фреймворк, имеющий встроенный веб-сервер.
* В качестве СУБД выбрана SQLite.
* ORM для взаимодействия с базой данных – Peewee.

## 2.4. UML диаграммы

### 2.4.1. Диаграмма вариантов использования



Рис. 1. Диаграмма вариантов использования.

Рассмотрим диаграмму вариантов использования для приложения на рис. 1.

Актер – пользователь приложения.

Отношения ассоциации - действия, которые может осуществлять пользователь:

* добавить расход;
* редактировать расход;
* удалить расход;
* добавить категорию;
* редактировать категорию;
* редактировать порог;
* удалить категорию;
* получить отчет.

Отношение включения: установить порог категории при ее добавлении.

### 2.4.2. Диаграмма последовательности

* Диаграмма последовательности для запуска приложения, рис. 2:

Рис. 2. Диаграмма последовательности для запуска приложения.

При запуске приложение делает запрос на сервер, который в свою очередь делает запрос в БД на данные по расходам. Когда БД возвращает список всех расходов, сервер посылает ответ в формате json. Затем на начальном экране отображается список всех расходов, содержащий дату, категорию, сумму и комментарий для каждого расхода.

* Диаграмма последовательности для добавления категории, рис. 3:

Рис. 3. Диаграмма последовательности для добавления категории.

Для добавления новой категории пользователю необходимо на панели управления перейти на вкладку «Категории» и нажать кнопку «+». После этого отображается окно для ввода названия категории и ежемесячного порога. Пользователю необходимо ввести данные и нажать кнопку «Сохранить». В случае успешно введенных данных, приложение делает запрос на сервер, который обращается в БД с запросом на сохранение. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список категорий.

* Диаграмма последовательности для добавления расхода, рис. 4:

Рис. 4. Диаграмма последовательности для добавления расхода.

Для добавления новой категории пользователю необходимо на главном экране нажать кнопку «+». После этого отображается окно для ввода данных о расходе. Пользователю необходимо ввести данные и нажать кнопку «Сохранить». В случае успешно введенных данных, приложение делает запрос на сервер, который обращается в БД с запросом на сохранение. Когда данные сохранятся, сервер посылает ответ приложению. Затем пользователю отображается обновленный список расходов.

### 2.4.3. Диаграмма коммуникаций

* Диаграмма коммуникации для запуска приложения, рис. 5:

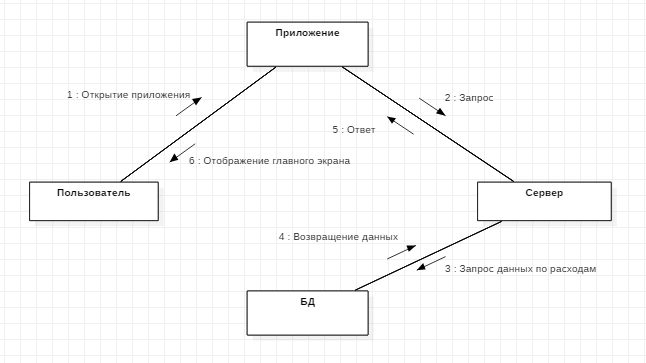


Рис. 5. Диаграмма коммуникации для запуска приложения.

* Диаграмма коммуникации для добавления категории, рис. 6:

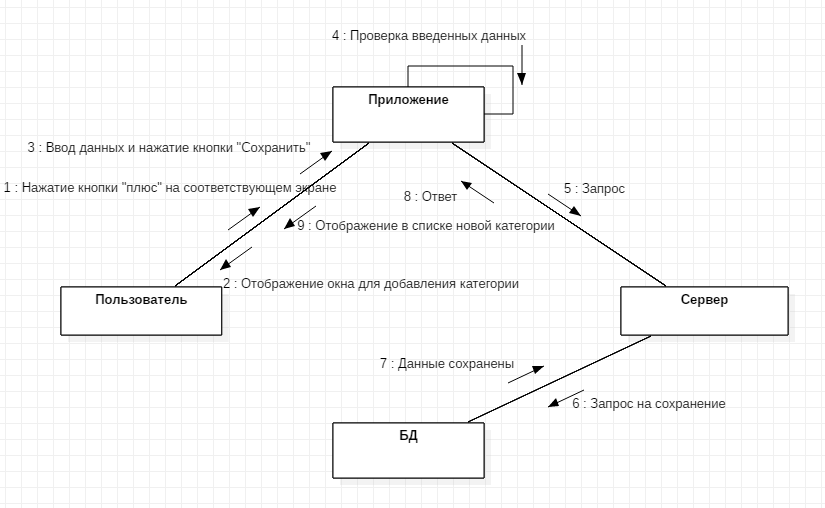


Рис. 6. Диаграмма коммуникации для добавления категории.

* Диаграмма коммуникации для добавления расхода, рис. 7:

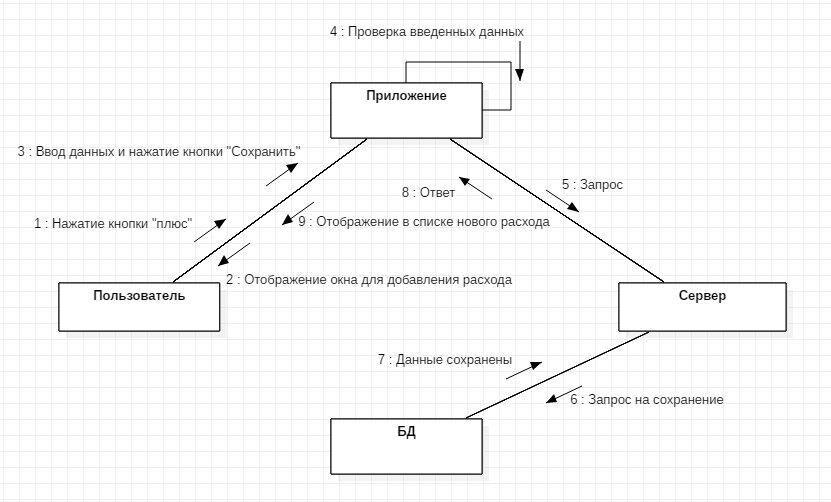


Рис. 7. Диаграмма коммуникации для добавления расхода.

Сообщения в диаграмме коммуникаций такие же, что и в диаграмме последовательности, поэтому их повторное описание не приводится.

### 2.4.4. Диаграмма состояний

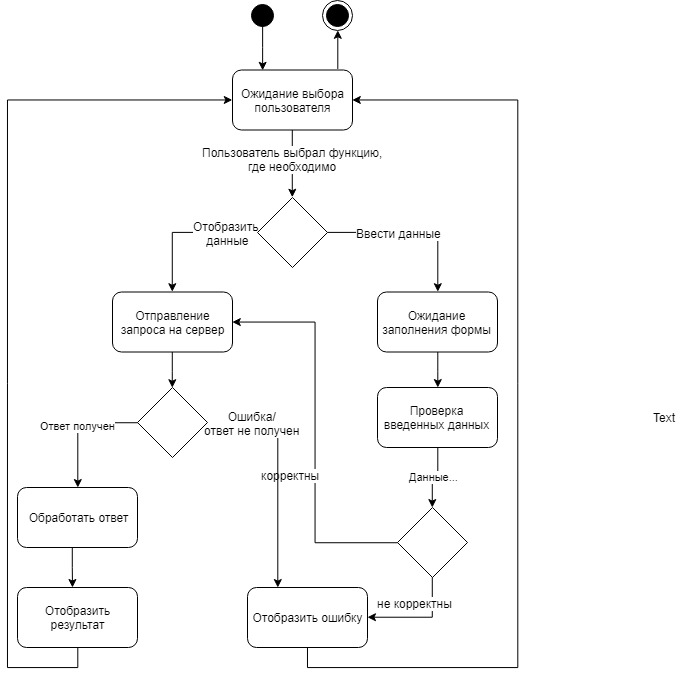


Рис. 8. Диаграмма состояний.

Диаграмма состояний отражает возможные состояния системы. При входе в приложение система находится в ожидании выбора пользователя. В зависимости от выбора пользователя возможны 2 основные цепочки состояний, связанных с отображение данных, необходимых пользователю, и с вводом данных.

Если пользователь выбрал функцию, где необходимо отобразить данные, система отправляет соответствующий запрос на сервер, далее если ответ получен, система обрабатывает ответ и отображает результат пользователю и переходит в состояние ожидания выбора пользователя. Если запрос выполнился с ошибкой или ответ не получен система выводит ошибку и переходит в состояние ожидания.

Если пользователь выбрал функцию, где необходимо вводить данные, система сначала ожидает заполнения формы, затем проверяет данные. Если данные корректны, отправляет запрос на сервер, далее действует аналогично предыдущей функции. Если данные, введенные пользователей, некорректны, система выводит ошибку.

### 2.4.5. Диаграмма классов

### 2.4.6. Диаграмма развертывания

На рис. 12 представлена диаграмма развертывания, чтобы определить какие аппаратные компоненты («узлы») существуют, какие программные компоненты работают на каждом узле и как различные части этого комплекса соединяются друг с другом.

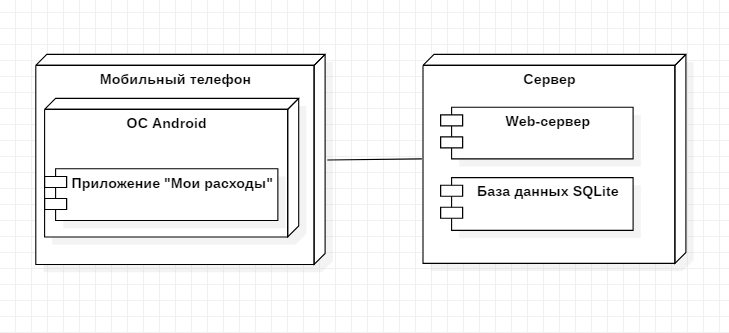


Рис. 12. Диаграмма развертывания.

### 2.4.7. Диаграмма объектов

### 2.4.8. Диаграмма деятельности

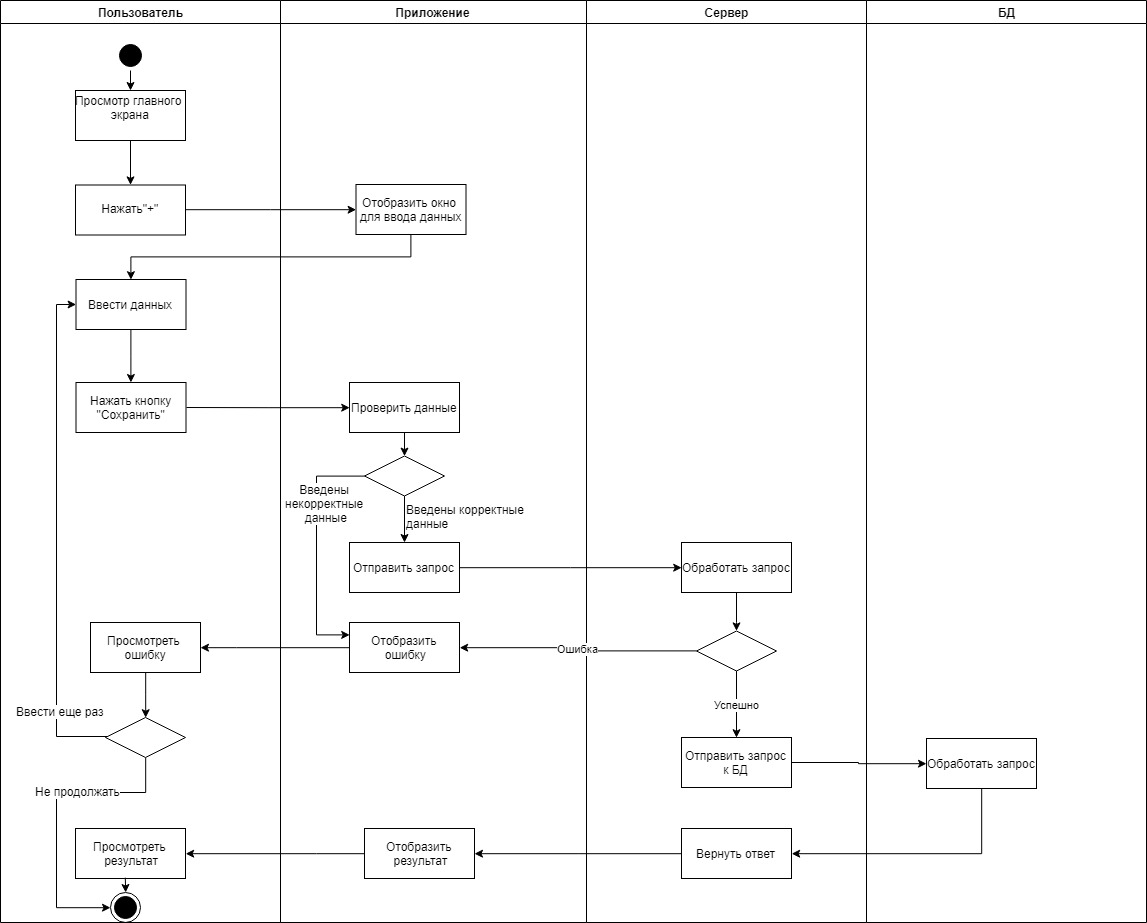


Рис. \*. Диаграмма активности для добавления расхода

Пользователь может добавить новые расход с главного экрана, нажав кнопку «+». Приложение отобразить окно для ввода данных. Пользователю необходимо ввести данные о новым расходе и нажать кнопку «Сохранить». Далее приложение проверяет введенные данные, если данные корректны, приложение оправляет запрос на сервер. Если пользователь ввел не верные данные, приложение выведет пользователю ошибку. После чего пользователь может заново ввести данные или закончить действие. После отправки запроса на сервер, сервер обрабатывает запрос и отправлять запрос к базе данных. После чего получает данные от БД и возвращает ответ. Приложение возвращает результат пользователю, и пользователь просматривает результат.

Если при обработке на сервере произошла ошибка или ответ от сервера не получен, приложение отображает пользователю ошибку, пользователь просматривает ошибку, после чего может попробовать еще раз или завершить действие.

## 2.5. Схема базы данных



Рис. 15. Схема базы данных

В базе данных содержится 3 таблицы:

Таблица «Категории» хранит данные о существующих категориях расходах, это id, название и порог категории. Id является первичным ключом. Поле «Название» является обязательным, поле «Порог» - не обязательным.

Таблица «Расходы» хранит данные расходах. Id является первичным ключом. Поле «Категория» показывает к какой категории относится расход, является внешним ключом на таблицу «Категории» и обязательным, поле «Сумма» показывает потраченную сумму денег, является обязательным. Поле «Дата» хранит дату совершения расхода, является обязательным.

Таблица «Категория\_сумма» хранит информацию об общей сумме денег, потраченной по категории за текущий месяц. Поле «Категория» является первичным ключом и внешним ключом на таблицу «Категории». Поле «Сумма» хранит общую сумму потраченных денег по категории, является обязательным.

# ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ СИСТЕМЕ

1. Приложение должно иметь клиент-серверную архитектуру.
2. Приложение должно быть разработано для операционной системы Android.
3. Данные пользователя хранятся в базе данных на сервере.
4. При запросе данных, клиентское приложение должно отправлять соответствующий запрос на сервер, сервер должен обрабатывать запрос и делать запрос к БД. После чего отправлять ответ.
5. При добавлении, изменении или удалении данных клиентское приложение должно отправлять соответствующий запрос на сервер, сервер должен обрабатывать запрос и делать запрос к БД. После чего отправлять ответ.
6. Запросы на сервер должны отправляться по HTTP протоколу, сервер отправляет ответы в формате JSON.
7. Система должна выполнять обозначенные выше задачи, а именно:
   1. Пользователь может просматривать все записи о расходах.
   2. Пользователь может добавлять запись о расходах с помощью выбора категории, ввода суммы расхода, даты и комментария (по желанию).
   3. Если после пересчета суммы по категории эта сумма превышает заданный пользователем порог, система выводит уведомление о превышении порога.
   4. Пользователь может изменять существующие записи о расходах, выбрав необходимую запись в общем списке записей. Он может изменить категорию расхода, сумму, дату или комментарий
   5. Пользователь может удалить существующую запись. Для этого в общем списке записей необходимо нажать кнопку удаления на нужной записи.
   6. Пользователь может добавлять новые категории расходов. Ему нужно указать название категории и по желанию порог для этой категории.
   7. Пользователь может редактировать категории. Можно изменить название категории, добавить порог или изменить порог.
   8. Пользователь может удалить категорию. При этом все записи расходов для этой категории удаляются.
   9. Пользователь может получать отчет в виде круговой диаграммы с данными об общей сумме расходов по каждой категории за текущий месяц.

# 5. ПЛАНИРОВАНИЕ РАБОТ

## 5.1. Выбор методологии проектирования

Для проекта выбрана методология Agile. Главная причина отказа от классического Waterfall является то, что в этой методологии предполагает последовательный переход к каждому этапу разработки и невозможностью вернуться на шаг назад. Внести какие-либо изменения будет возможно только после релиза проекта.

Преимущества методологии Agile:

* 1. Внесение необходимых изменений и внедрение нового функционала может происходить независимо от цикла разработки продукта.
  2. Проект состоит из коротких и понятных циклов, по окончании которых создается рабочая версия продукта. Каждый цикл представляет собой завершенный мини-проект, в котором есть этапы анализа, планирования, тестирования и реализации. В итоге клиент получает продукт, который, при необходимости, корректируется.
  3. Гибкий процесс корректировок в любой итерации позволяет снизить производственные риски. - Довольно быстрый релиз пробной версии для дальнейших корректировок и тестирования.

## 5.2. Виды работ, которые необходимо выполнить в процессе разработки программного средства

Разработка приложения должна быть проведена в два этапа:

* 1. разработка документации;
  2. разработка приложения.

Рассмотрим подробнее каждый из них.

На этапе разработки документации должны быть выполнены следующие работы:

* написание введения;
* определение актуальности задачи;
* составление анализа аналогов;
* выбор методологии разработки;
* определение требований;
* создание диаграммы вариантов использования;
* создание диаграммы коммуникаций;
* создание диаграммы состояний;
* создание диаграммы последовательности;
* создание диаграммы развертывания;
* создание диаграммы деятельности;
* создание диаграммы объектов;
* написание глоссария проекта;
* написание ТЗ;
* создание функциональной схемы;
* создание прототипов экранов;
* создание диаграммы классов.

На этапе разработки приложения должна быть выполнены следующие виды работ:

● создание репозитория в системе контроля версия GitHub;

● установка и настройка необходимых инструментальных средств;

● проектирование и создание БД;

● реализация классов приложения;

● создание интерфейсной части приложения;

● создание логической части приложения.

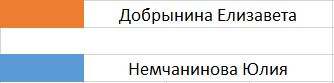
## 5.3. Состав команды, распределение задач по участникам и по времени

В состав команды разработчиков вошли:

1. Добрынина Елизавета;

2. Немчанинова Юлия.

Ниже представлена диаграмма Ганта для выполненных работ на два этапа, описанных в пункте 5.2.



# 6. РЕАЛИЗАЦИЯ

## 6.1. Использованные средства реализации

# 7. ТЕСТИРОВАНИЕ

# 8. ИТОГИ РАБОТЫ

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРЫ