МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *Информационных технологий управления*

*Мобильное приложение по учету личных расходов “Мои расходы”*

*Курсовой проект*

*09.03.02 Информационные системы и технологии*

Допущен к защите

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Добрынина Е.А., 3 курс, д/о*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Немчанинова Ю.Е., 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*Х.А. Полещук, преподаватель*

Воронеж 2019

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc10485110)

[1. ПОСТАНОВКА ЗАДАЧИ 5](#_Toc10485111)

[2. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 6](#_Toc10485112)

[2.1. Глоссарий 6](#_Toc10485113)

[2.2. Анализ существующих решений 6](#_Toc10485114)

[2.2.1. Expense Manager 6](#_Toc10485115)

[2.2.2. CoinKeeper 7](#_Toc10485116)

[2.2.3. EasyMoney 7](#_Toc10485117)

[2.3. UML диаграммы 8](#_Toc10485118)

[2.3.1. Диаграмма вариантов использования 8](#_Toc10485119)

[2.3.2. Диаграмма последовательности 9](#_Toc10485120)

[2.3.3. Диаграмма коммуникаций 11](#_Toc10485121)

[2.3.4. Диаграмма состояний 13](#_Toc10485122)

[2.3.5. Диаграмма развертывания 15](#_Toc10485123)

[2.3.6. Диаграмма объектов 16](#_Toc10485124)

[2.3.7. Диаграмма деятельности 16](#_Toc10485125)

[2.4. Схема базы данных 17](#_Toc10485126)

[3. ТРЕБОВАНИЯ К РАЗРАБАТЫВАЕМОЙ СИСТЕМЕ 19](#_Toc10485127)

[4. ПЛАНИРОВАНИЕ РАБОТ 21](#_Toc10485128)

[4.1. Выбор методологии проектирования 21](#_Toc10485129)

[4.2. Виды работ, которые необходимо выполнить в процессе разработки программного средства 21](#_Toc10485130)

[4.3. Состав команды, распределение задач по участникам 23](#_Toc10485131)

[5. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 24](#_Toc10485132)

[5.1. Анализ средств реализации 24](#_Toc10485133)

[5.2. Реализация 25](#_Toc10485134)

[5.2.1. Основные сущности приложения и их реализация 25](#_Toc10485135)

[5.2.2. Реализация работы с базой данных 25](#_Toc10485136)

[5.2.3. Реализация серверной части 25](#_Toc10485137)

[5.3. Интерфейс 25](#_Toc10485138)

[6. ТЕСТИРОВАНИЕ 30](#_Toc10485139)

[ЗАКЛЮЧЕНИЕ 31](#_Toc10485140)

[СПИСОК ЛИТЕРАТУРЫ 32](#_Toc10485141)

[ПРИЛОЖЕНИЕ 1 33](#_Toc10485142)

# 

# 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, обладающее простым и интуитивно-понятным интерфейсом.

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. UML диаграммы

### 2.3.1. Диаграмма вариантов использования



*Рисунок 1. Диаграмма вариантов использования*

Рассмотрим диаграмму вариантов использования для приложения на рисунке 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.

### 2.3.2. Диаграмма последовательности

Диаграмма последовательности для запуска приложения, рисунок 2

*Рисунок 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, рисунок 3

*Рисунок 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.

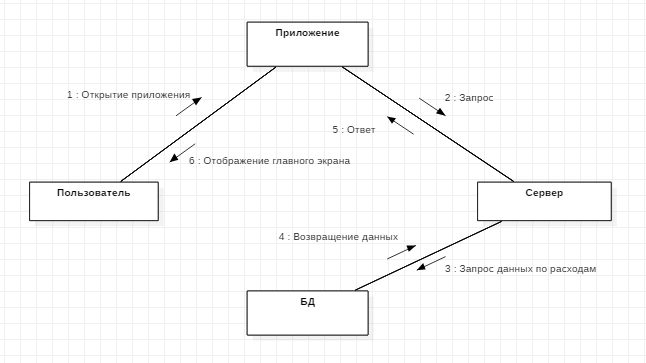
Диаграмма последовательности для добавления расхода, рисунок 4

*Рисунок 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.

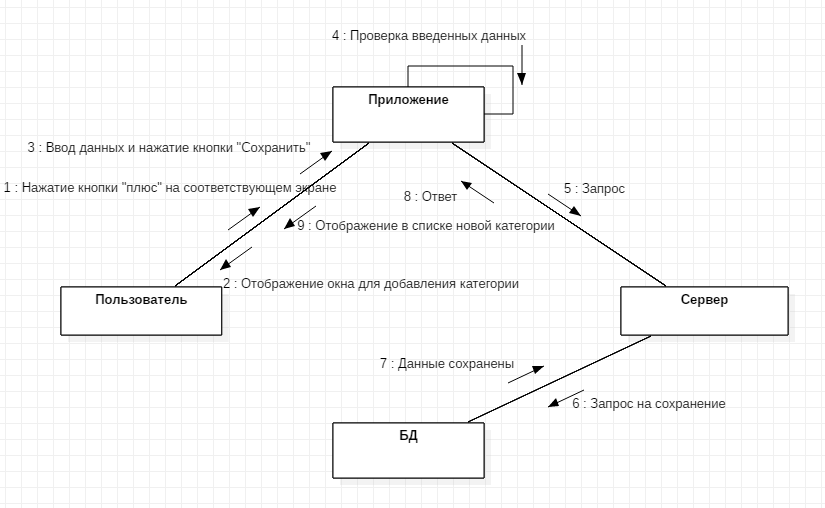
### 2.3.3. Диаграмма коммуникаций

Диаграмма коммуникации для запуска приложения, рисунок 5



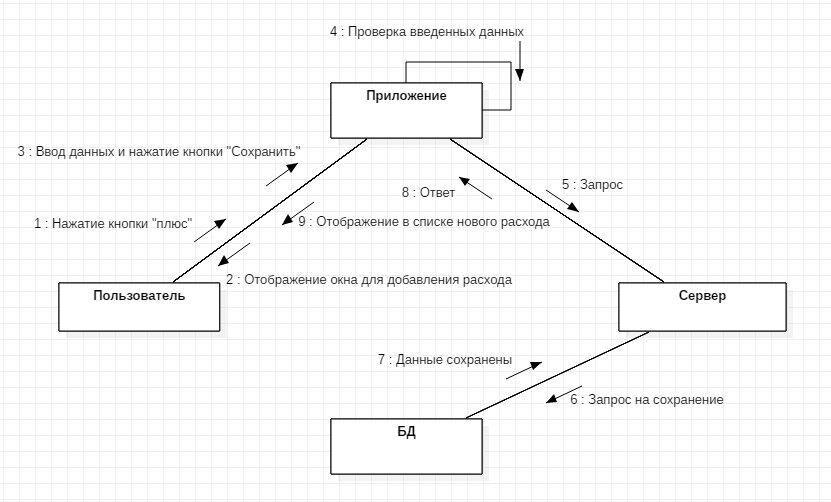
*Рисунок 5. Диаграмма коммуникации для запуска приложения*

Диаграмма коммуникации для добавления категории, рисунок 6



*Рисунок 6. Диаграмма коммуникации для добавления категории*

Диаграмма коммуникации для добавления расхода, рисунок 7

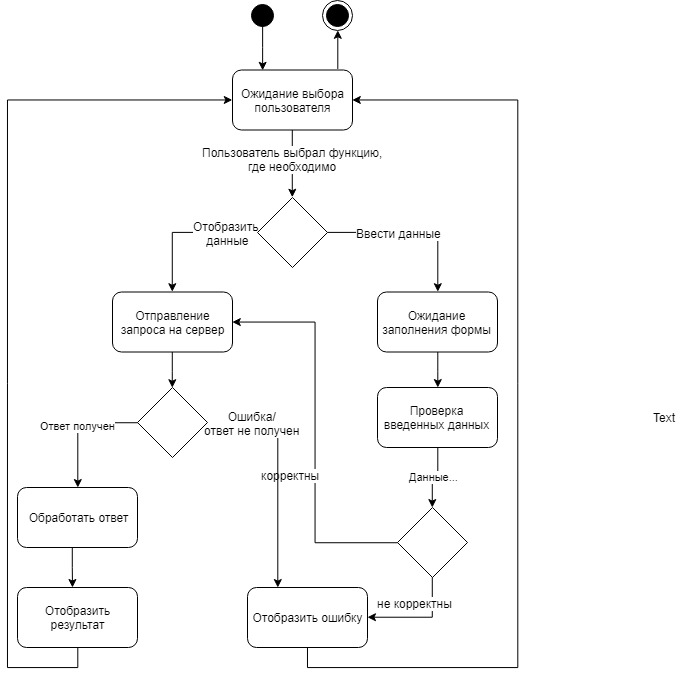


*Рисунок 7. Диаграмма коммуникации для добавления расхода*

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

### 2.3.4. Диаграмма состояний

На рисунке 8 изображена диаграмма состояний.



*Рисунок 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.

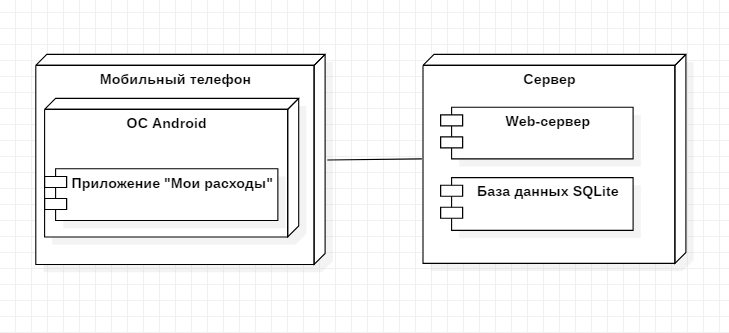
Если пользователь выбрал функцию, где необходимо отобразить данные, система отправляет соответствующий запрос на сервер, далее если ответ получен, система обрабатывает ответ и отображает результат пользователю и переходит в состояние ожидания выбора пользователя. Если запрос выполнился с ошибкой или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

### 2.3.5. Диаграмма развертывания

На рисунке 9 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.

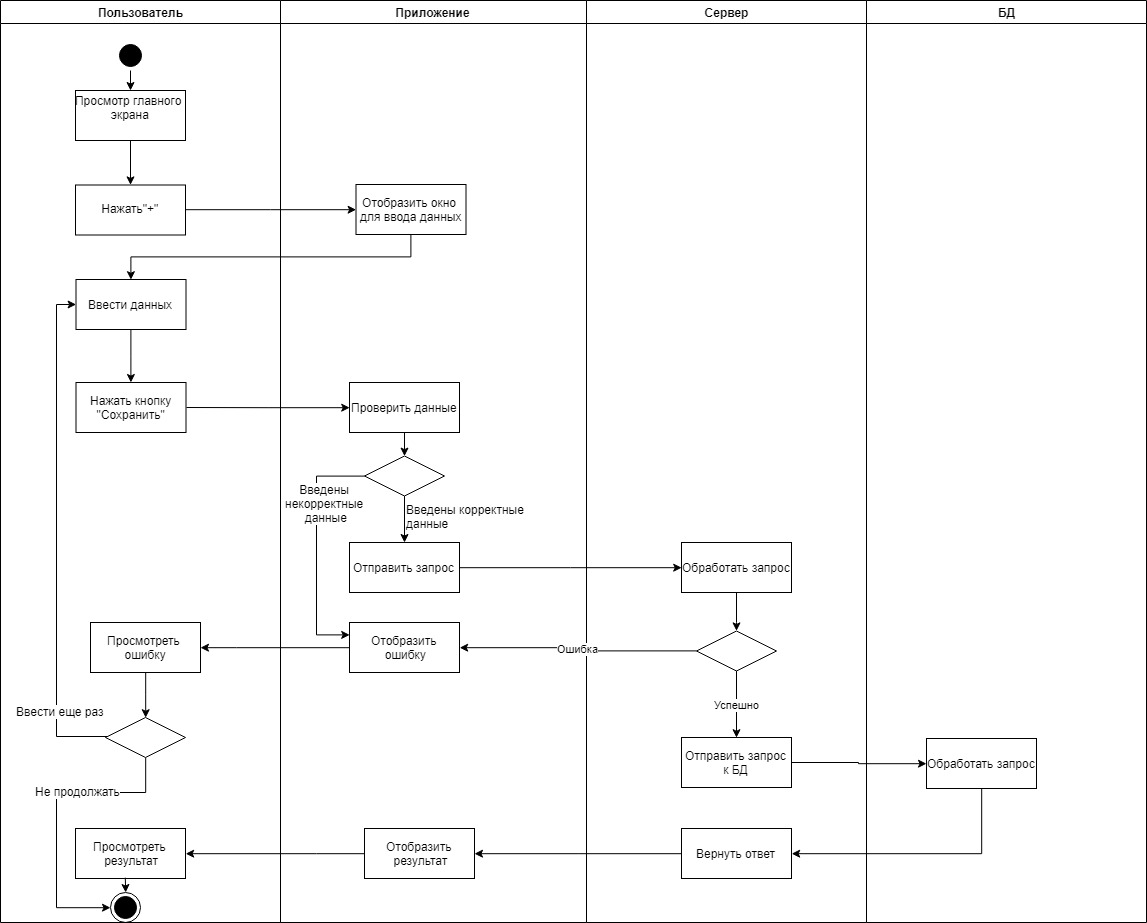
Для данного приложения узлами являются мобильный телефон и сервер. Средой для мобильного приложения является операционная система Android, где находится клиентское приложения. А на узле сервера расположены веб-сервер и база данных.



*Рисунок 9. Диаграмма развертывания*

### 2.3.6. Диаграмма деятельности

На рисунке 10 изображена диаграмма активности.



*Рисунок 10. Диаграмма активности для добавления расхода*

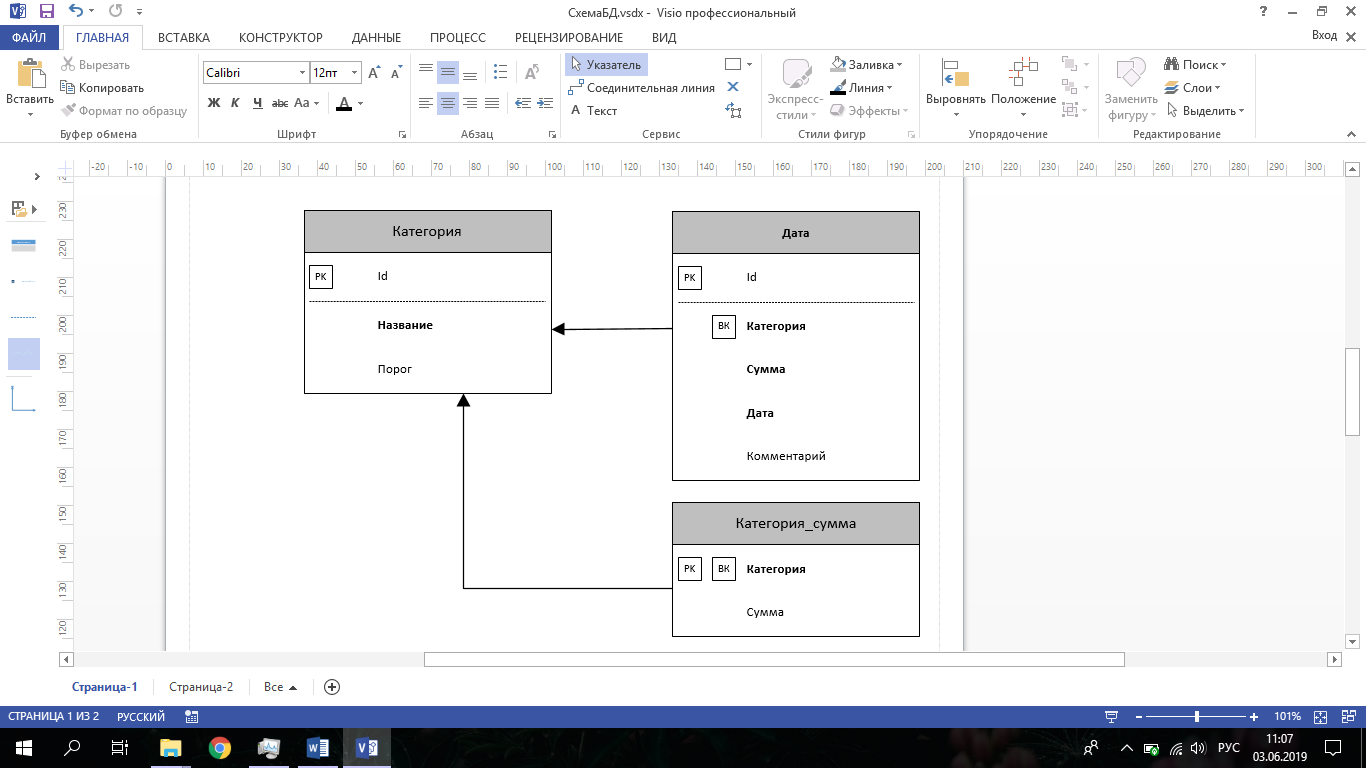
Пользователь может добавить новые расход с главного экрана, нажав кнопку «+». Приложение отобразить окно для ввода данных. Пользователю необходимо ввести данные о новым расходе и нажать кнопку «Сохранить».

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.

## 2.4. Схема базы данных

На рисунке 11 изображена схема базы данных.



*Рисунок 11. 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. Общение клиента и сервера производится с помощью REST API запросов по HTTP протоколу с телом запроса в формате JSON.
    7. Система должна выполнять обозначенные выше задачи, а именно:

7.1. Пользователь может просматривать все записи о расходах.

7.2. Пользователь может добавлять запись о расходах с помощью выбора категории, ввода суммы расхода, даты и комментария (по желанию).

7.3. Если после пересчета суммы по категории эта сумма превышает заданный пользователем порог, система выводит уведомление о превышении порога.

7.4. Пользователь может изменять существующие записи о расходах, выбрав необходимую запись в общем списке записей. Он может изменить категорию расхода, сумму, дату или комментарий

7.5. Пользователь может удалить существующую запись. Для этого в общем списке записей необходимо нажать кнопку удаления на нужной записи.

7.6. Пользователь может добавлять новые категории расходов. Ему нужно указать название категории и порог для этой категории.

7.7. Пользователь может редактировать категории. Можно изменить название категории или порог.

7.8. Пользователь может удалить категорию. При этом все записи расходов для этой категории удаляются.

7.9. Пользователь может получать отчет в виде круговой диаграммы с данными об общей сумме расходов по каждой категории за текущий месяц.

# 4. ПЛАНИРОВАНИЕ РАБОТ

## 4.1. 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.

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

Разработка приложения должна быть проведена в три этапа:

* 1. разработка документации;
  2. разработка приложения;
  3. тестирование готового продукта.

Рассмотрим подробнее каждый из них.

На этапе разработки документации должны быть выполнены следующие работы:

* + 1. написание технического задания согласно ГОСТ 34;
    2. написание анализа предметной области, включающее:

2.1. актуальность выбранной задачи;

2.2 анализ существующих решений;

2.3. uml-диаграммы (вариантов использования, последовательности, коммуникаций, состояний, деятельности, объектов, развертывания, классов (front-end и back-end));

2.4. схему базы данных.

* + 1. составление модульной схемы приложения.

На этапе разработки приложения должны быть выполнены следующие работы:

* + 1. написание клиентского приложения, включающее:

1.1. создание экранов, требуемых для осуществления всех функциональных требований:

1.2.1. экран для просмотра всех категорий;

1.2.2. экран для просмотра всех расходов;

1.2.3. экран для просмотра отчета.

1.2. создание панели управления, осуществляющей возможность перехода между экранами в п. 1.1;

1.3. добавление расхода;

1.4. добавление категории;

1.5. редактирование расхода;

1.6. редактирования категории;

1.7. удаление расхода;

1.8. удаление категории;

1.9. формирование отчета;

1.10. вывод уведомления о превышении порога категории.

* + 1. Написание серверной части, включающее:

2.1. создание базы данных на сервере;

2.2. написание rest-запросов к базе данных для осуществления всех функциональных требований из п. 1;

2.3. размещение серверной части приложения на облачном хостинге;

2.4. осуществление взаимодействия между клиентским приложением и сервером.

На этапе тестирования готового продукта должны быть выполнены следующие работы:

* + 1. написание тест-плана;
    2. составление тест-кейсов;
    3. проведение тестирования;
    4. документирование результатов тестирования.

## 4.3. Состав команды, распределение задач по участникам

В состав команды разработчиков вошли:

1. Добрынина Елизавета;

2. Немчанинова Юлия.

С таблицами по распределению задач по участникам можно ознакомиться в Приложении 1.

# 5. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

## 5.1. Анализ средств реализации

В качестве средств реализации для Android-приложения «Мои расходы» были выбраны:

1. Android SDK – средство разработки мобильных приложений для операционной системы Android.

Чертой, отличающей от других средств разработки, является наличие широких функциональных возможностей, позволяющих запускать тестирование и отладку исходных кодов, оценивать работу приложения в режиме совместимости с различными версиями ОС Android.

2. Клиентское приложение выполнено на языке Java.

На данный момент самыми распространенными технологиями для создания Android-приложений являются Java и Kotlin. Kotlin – новый язык, стремительно набирающий популярность среди разработчиков, однако предпочтение было отдано Java по следующим причинам:

* Поскольку Android разработан на Java, этот язык имеет уже готовые библиотеки и SDK для облегчения процесса разработки.
* С помощью виртуальной машины Java программы на этом языке могут запускаться практически в любой системе. В результате Java получает лидерство среди Android-приложений.
* По данным tiobe.com на 2019 год Java является самым популярным языком в мире, благодаря чему в сообществах на GitHub и Stack Overflow разработчики могут получить помощь практически по любой проблеме.

3. В качестве СУБД была выбрана SQLite. Ее преимуществом перед другими СУБД является:

* Высокая скорость операций выборки данных.
* Отсутствие необходимости настройки сервера СУБД. Следовательно, база данных вместе с сервером будут находиться на облачном хостинге, и они смогу общаться напрямую, без осуществления дополнительных запросов сервер-БД.

4. Сервер приложения написан на языке Python c использованием веб-фреймворка FastAPI.

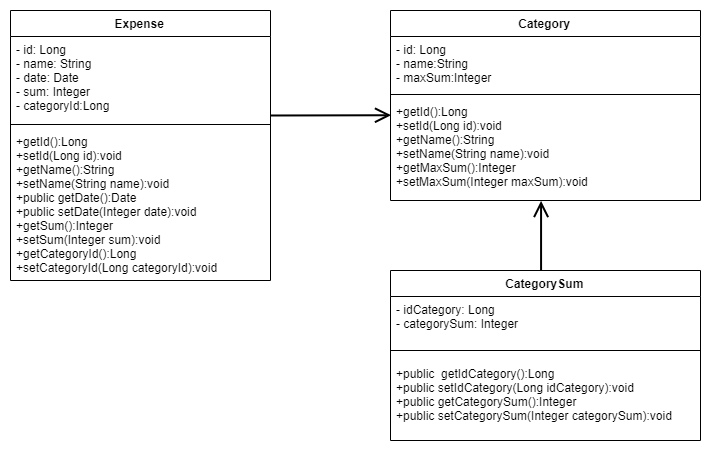
5. Общение с сервером происходит посредством REST API по протоколу HTTP запросами и ответами в формате JSON.

6. На стороне клиентского приложения используется библиотека Android Volley, которая автоматически составляет и принимает на себя сетевые запросы, а также выполняет их для извлечения ответа.

## 5.2. Реализация

### 5.2.1. Основные сущности приложения и их реализация

На рисунке 12 представлена диаграмма классов основных объектов – сущностей, соответствующих таблицам и базе данных.



*Рисунок 12. Диаграмма классов основных сущностей приложения*

Класс Category соответствует сущности Категория, который имеет следующие свойства:

* «id» - уникальный идентификатор;
* «name» - имя категории;
* «maxSum» - порог расходов по категории.

Класс Expense соответствует сущности Расход, который имеет следующие свойства:

* «id» - уникальный идентификатор;
* «name» - имя категории;
* «date» - дата;
* «sum» - сумма расхода;
* «categoryId» - индекс категории, которой принадлежит этот расход.

Класс CategorySum соответствует сущности Категория\_сумма, который имеет следующие свойства:

* «idCategory» - уникальный идентификатор категории;
* «currentSum» - текущую сумму расходов по категории.

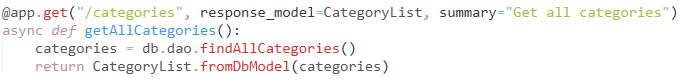
### 5.2.2. Сервер приложения

Класс dao.py содержит функции, с помощью которых сервер взаимодействует с базой данных:

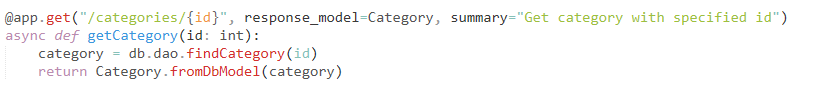
* findAllCategories() возвращает список всех категорий из базы данных;
* findCategory(id) возвращает экземпляр класса Category с указанным id;
* createCategory(name, maxSum) создает новый экземпляр класса Category;
* deleteCategory(id) удаляет категорию с указанным id;
* updateCategory(id, name, maxSum) обновляет экземпляр класса Category с указанным id;
* findAllExpenses()возвращает список всех расходов из базы данных;
* createExpense(name, date, sum, categoryId) создает новый экземпляр класса Expense;
* deleteExpense(id) удаляет расход с указанным id;
* updateExpense(id, name, date, sum, categoryId) обновляет экземпляр класса Expense с указанным id.

Класс app.py содержит REST запросы сервера к базе данных, используя функции класса dao.py:

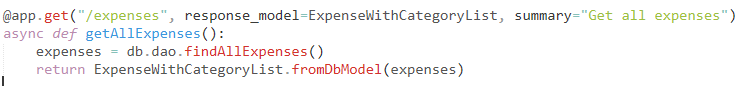
* GET – запросы:
  + Запрос на получение списка всех категорий:



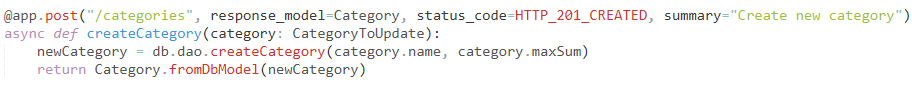
* + Запрос на получение категории с указанным id:



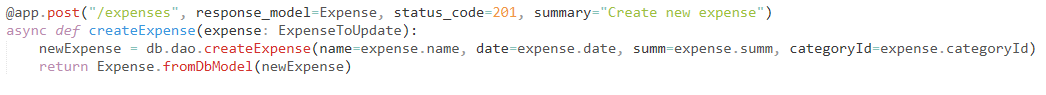
* + Запрос на получение списка всех расходов:



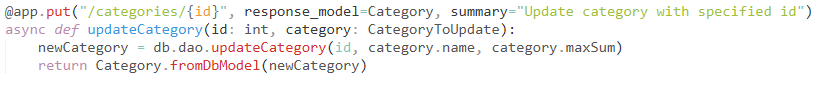
* POST – запросы:
  + Запрос на создание новой категории:



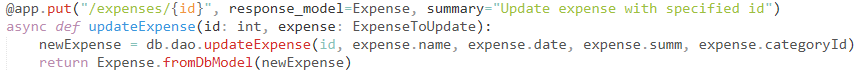
* + Запрос на создание нового расхода:



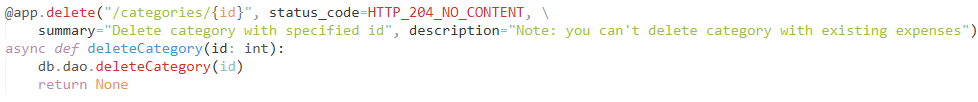
* PUT – запросы:
  + Запрос на обновление категории:



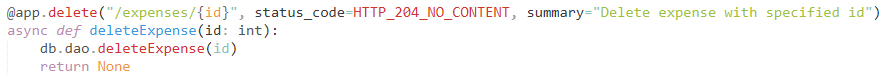
* + Запрос на обновление расхода:



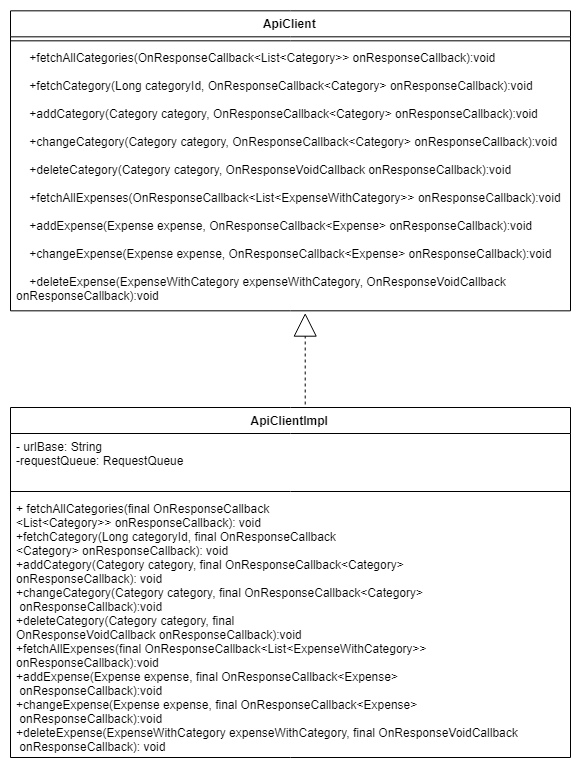
* DELETE – запросы:
  + Запрос на удаление категории:



* + Запрос на удаление расхода:



## 5.2.3. Запросы на стороне клиентского приложения



*Рисунок 13. Диаграмма классов модуля request*

Класс ApiClientImpl наследуется от базового класса ApiClient.

Методы класса ApiClientImpl обеспечивают обработку ответов сервера.

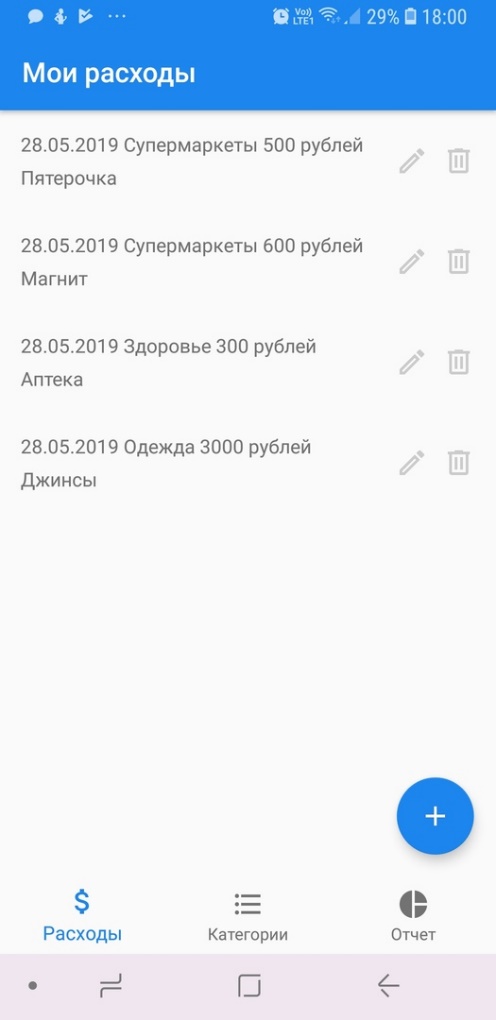
## 5.3. Интерфейс

На рисунках 14-21 представлен интерфейс итогового продукта.

Главный экран приложения (рисунок 14) содержит список всех расходов, куда входит информация о дате, категории, сумме и комментарии для расхода. Напротив каждого расхода есть визуальные элементы «ручка» и «ведро», соответствующие кнопкам для редактирования и удаления расхода.

В нижней части экрана расположена панель управления, позволяющая перемещаться между основными экранами приложения, и кнопка «+» для добавления расхода.

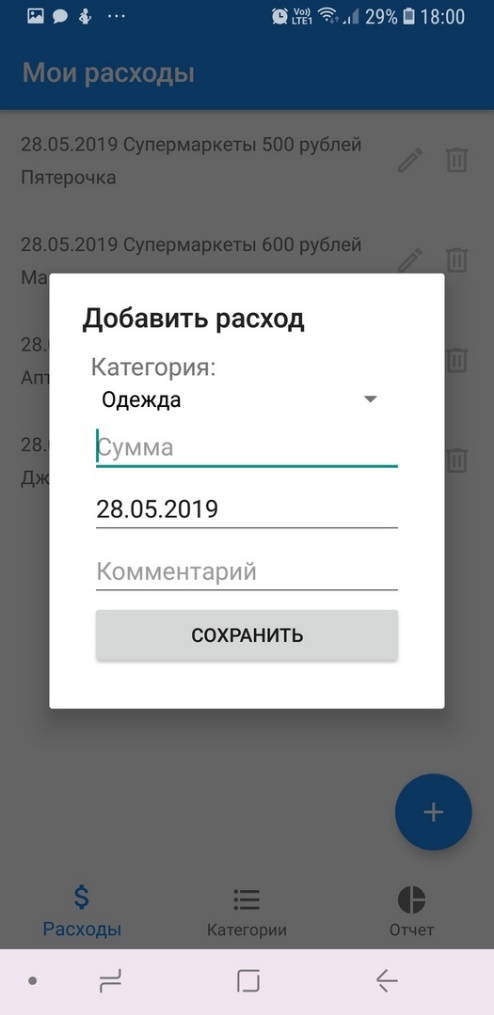
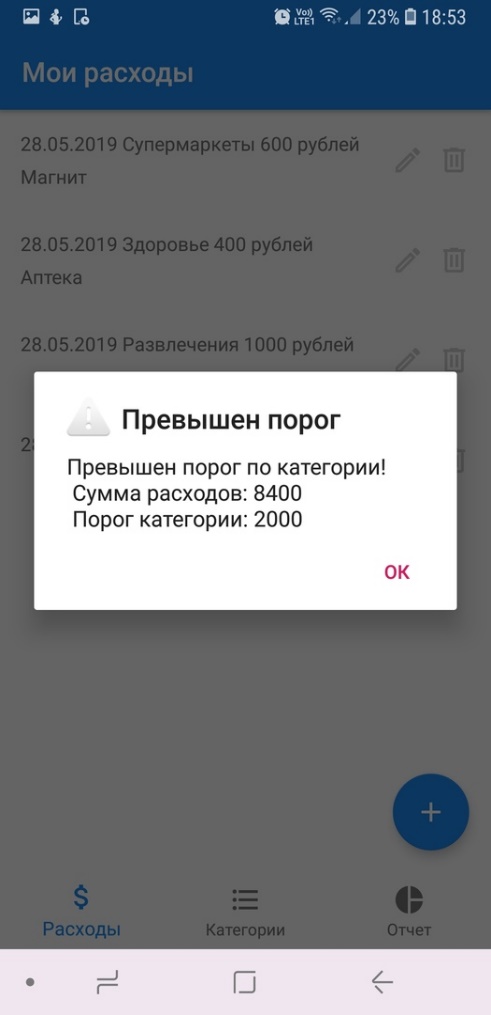
Данные элементы и их расположение являются очень удобными для пользователя, т.к. они не перегружают интерфейс лишней визуализацией и позволяют легко управлять приложением одной рукой.



*Рисунок 14. Главный экран*

При нажатии на кнопку «+», открывается окно для ввода данных о расходе (рисунок 15). Оно содержит поле для выбора категории из уже существующих, а также поля для ввода суммы, даты и комментария.

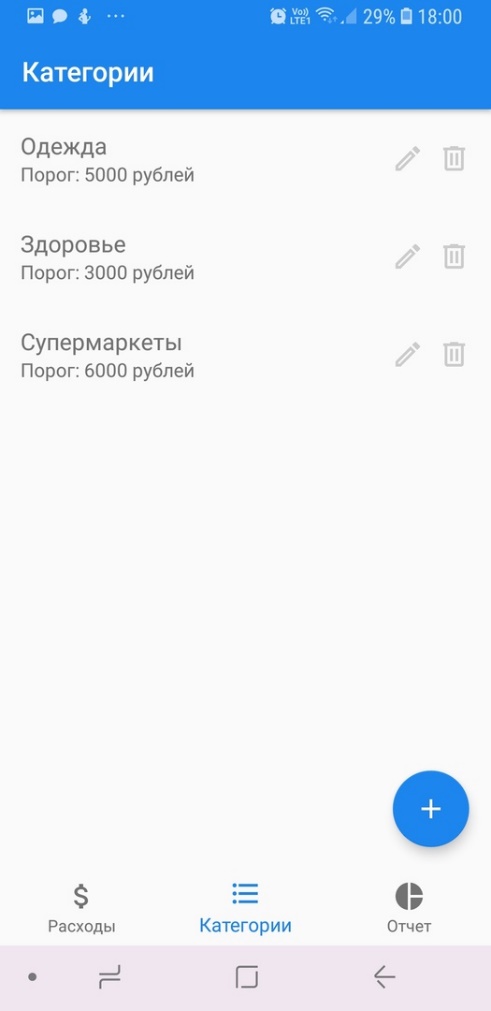
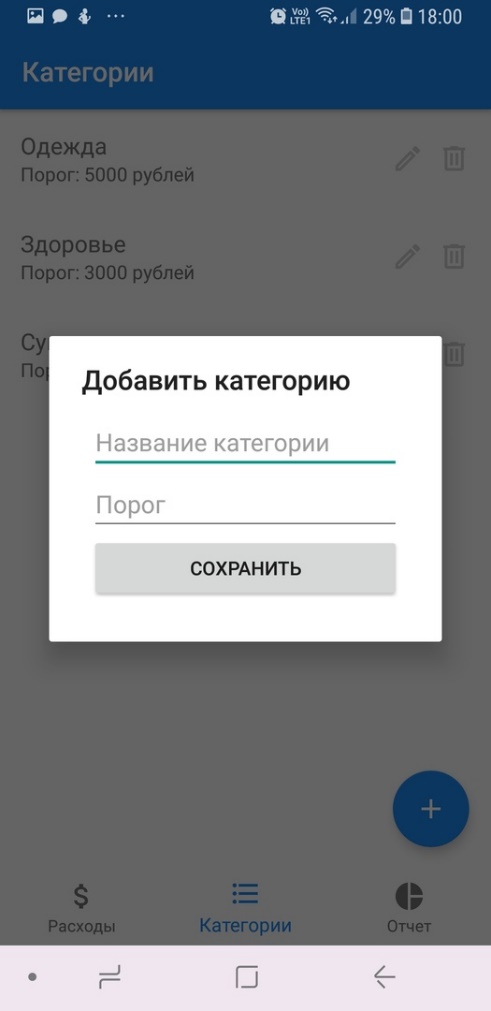
Если при добавлении нового расхода сумма всех расходов в этой категории превысит порог, пользователю высвечивается предупреждение (рисунок 16).

*Рисунок 15. Добавление расхода Рисунок 16. Превышение порога*

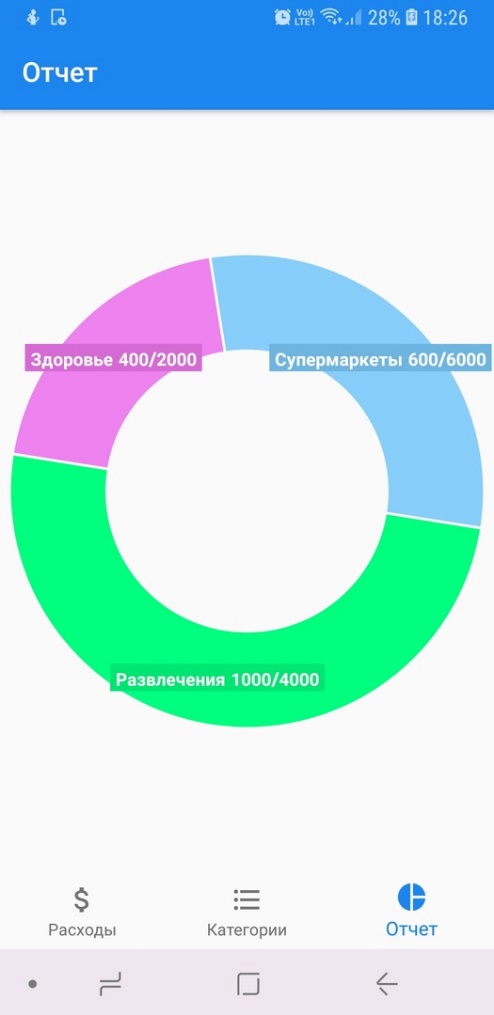
При выборе на панели управления вкладки «Категории», открывается экран, содержащий список всех категорий с названием и порогом (рисунок 17). Здесь пользователю также предоставляется возможность удалять и редактировать категории с помощью тех же элементов, что в расходах.

При нажатии на кнопку «+», открывается окно для ввода данных о категории (рисунок 18). Оно содержит поля для ввода названия категории и ее ежемесячного порога.

*Рисунок 17. Экран категорий Рисунок 18. Добавление категории*

Что просмотреть отчет, пользователю необходимо на панели управления выбрать вкладку «Отчет» (рисунок 19). Отчет включает в себя круговую диаграмму, на которой содержится информация о том, какая сумма расходов для каждой категории существует на данный момент, и как расходы по категориям соотносятся друг с другом.



*Рисунок 19. Экран отчета*

# 6. ТЕСТИРОВАНИЕ

Тестирование было проведено в соответствии с документом «План тестирования», который прилагается к данной курсовой работе.

Основные виды тестирования, которые были проведены:

1. Smoke testing – поверхностное тестирование системы на предмет работоспособности. В рамках этого тестирования были проверены возможности запуска клиентского приложения, отображение пользовательского интерфейса, соединение и получение ответа от сервера .

2. Sanity testing – тестирование работы конкретных функций системы для доказательства того, что система работает согласно требованиям в Техническом Задании. В рамках данного теста были проверены функциональные возможности системы : добавление, изменение, удаление записей о расходах, добавление, изменение, удаление категорий и получение отчета.

3. Usability testing – тестирование того, насколько легко конечный пользователь сможет освоить систему. Здесь проверялось наличие сообщений-подсказок для пользователя, при неправильных действиях с его стороны.

4. Negative testing – тестирование поведении системы, при попытке ввода некорректных данных. Было протестировано поведение системы, при попытке ввода некорректных данных в различных функциях.

Тест - кейсы представлены отдельным документом в качестве приложения. В результате тестирования оказалось более 90 % пройденных тест-кейсов. Не пройденные тест-кейсы не являются критическими и могут быть исправлены в случае необходимости доработки или расширения системы.

С результатами тестирования можно ознакомиться в Приложении 2.

# ЗАКЛЮЧЕНИЕ

В ходе курсового проекта было реализовано Android-приложение по отслеживаю личных расходов, которое удовлетворяет следующим требованиям:

* имеет удобный и интуитивно-понятный интерфейс;
* позволяет создавать, изменять, удалять категории расходов;
* позволяет определять порог расходов для каждой категории;
* позволяет создавать, изменять, удалять расходы по категориям;
* позволяет получать уведомления о превышении порога по категории;
* позволяет получать отчет о расходах в виде круговой диаграммы.

Были выполнены следующие задачи:

* разработано клиентское приложение;
* разработан сервер приложения, расположенный на облачном хостинге;
* осуществлено хранение данных в базе данных, расположенной с сервером на облачном хостинге;
* осуществлено взаимодействие клиентского приложения и сервера с помощью REST API по HTTP протоколу.

# СПИСОК ЛИТЕРАТУРЫ

# ПРИЛОЖЕНИЕ 1

**Распределение задач по участникам команды**

Таблица 1 – Распределение задач для участника Добрыниной Елизаветы

|  |  |  |
| --- | --- | --- |
| *№ задачи* | *Наименование* | *Ссылка на GitHub* |
| 1 | Введение |  |
| 2 | Анализ существующих решений |  |
| 3 | Анализ средств реализации |  |
| 4 | Диаграмма вариантов использования |  |
| 5 | Диаграмма последовательности |  |
| 6 | Диаграмма коммуникаций |  |
| 7 | Диаграмма развертывания |  |
| 8 | Модульная схема |  |
| 9 | Панель управления в нижней части экрана | <https://github.com/liza-dobrynina/MyCosts/commit/0ebdf70c24e0e51d858a4902ed5b33d0718f6049> |
| 10 | Добавление, редактирование и удаление категорий | <https://github.com/liza-dobrynina/MyCosts/commit/ddded447654c90075865c07a21e21c217e11c4dd> |
| 11 | Добавление, редактирование и удаление расходов | <https://github.com/liza-dobrynina/MyCosts/commit/405ef446b8a6f79bfb2823059761dc034e2670e5> |
| 12 | Предупреждение о превышении порога категории | <https://github.com/liza-dobrynina/MyCosts/commit/1886d7894eb0bee31545acce6f21475a25f4931c> |
| 13 | Отчет | <https://github.com/liza-dobrynina/MyCosts/commit/358a0a9d6354b39ae7a497f150ed034cdb9d5158> |
| 14 | Back-end | 1. <https://github.com/liza-dobrynina/MyCosts/commit/f27304347848215e4a2bc42f18b2e554d14ef28f> 2. <https://github.com/liza-dobrynina/MyCosts/commit/d0cee8200d9485414ae06e2653f0dcf23225033a> 3. <https://github.com/liza-dobrynina/MyCosts/commit/358a0a9d6354b39ae7a497f150ed034cdb9d5158> |
| 15 | Swagger |  |
| 16 | Видео |  |

Таблица 2 – Распределение задач для участника Немчаниновой Юлии

|  |  |  |
| --- | --- | --- |
| *№ задачи* | *Наименование* | *Ссылка на GitHub* |
| 1 | Постановка задачи |  |
| 2 | Выбор и анализ методологий проектирования |  |
| 3 | Глоссарий |  |
| 4 | Диаграмма состояний |  |
| 5 | Диаграмма деятельности |  |
| 6 | Диаграмма классов front-end |  |
| 7 | Диаграмма классов back-end |  |
| 8 | Диаграмма объектов |  |
| 9 | Схема и описание базы данных |  |
| 10 | Техническое задание по ГОСТу 34 |  |
| 11 | Создание экранов для расходов, категорий, отчета, добавления расхода и добавления категории | <https://github.com/liza-dobrynina/MyCosts/commit/d4cad7f01ef5517a7de7a0b9ed8764070ad4b3da> |
| 12 | Осуществление работы с базой данных | <https://github.com/liza-dobrynina/MyCosts/commit/ce4977ff9df51c089d81f6ee7d9e149bac90aa55> |
| 13 | Написание тест-кейсов |  |
| 14 | Проведение тестирования |  |
| 15 | Написание главы о тестировании в отчете |  |
| 16 | Презентация |  |

# ПРИЛОЖЕНИЕ 2

**Результаты тестирования**

