МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

Приложение по учету расходов “MoneyKeeper”

Курсовой проект

09.03.02 Информационные системы и технологии
Программная инженерия в информационных системах

Допущен к защите

Обучающийся _____*А.Д. Борисов, 3 курс, д/о*

Обучающийся _____*И.Е. Никонов, 3 курс, д/о*

Руководитель _____*Х.А. Полещук, преподаватель*

Воронеж 2019

Оглавление

Введение

Грамотное управление личными финансами в современном мире – это задача, которая лежит на каждом человеке. Сегодня все сложнее держать в голове все многочисленные расходы, а также сопоставлять их с доходами. Можно управляться с этой задачей, делая записи в блокнот, что не всегда удобно, или пользуясь онлайн-сервисами, что так же имеет свои недостатки, такие как наличие доступа к интернету и неуверенность в безопасности данных. В итоге наиболее удобным инструментом является мобильное приложение. Смартфон всегда под рукой и добавление новых записей о доходах и расходах – минутное дело.

Желаемое приложение должно облегчать жизнь пользователю, а не усложнять, это, к сожалению, могут далеко не все приложения подобного плана. Приложение должно быть легковесным и предоставлять только необходимую функциональность:

- Учет расходов и доходов;
- Просмотр истории финансовой деятельности;
- Ненагруженный, интуитивно понятный интерфейс также является необходимой особенностью хорошего инструмента.

Данный курсовой проект посвящен разработке именно такого, простого в освоении, но в то же время выполняющего самые необходимые функции, приложения, способного облегчить финансовые сложности человека, не нагружая его сложным интерфейсом, а его телефон жесткими системными требованиями.

1. Постановка задачи

Цель курсовой работы: реализовать Android приложение, которое позволяет осуществлять учет доходов и расходов, а также имеет визуально информативный интерфейс, который позволяет получать всю необходимую информацию в виде диаграммы с главного экрана.

Для достижения данной цели были выделены следующие задачи:

1. Разработка Front-end части приложения, находящиеся на телефоне пользователя;
2. Разработка Back-end части приложения, развернутой на удаленном сервере приложений;
3. Создание связи между Front-end и Back-end частями приложения;
4. Разработка базы данных, расположенной на удаленном сервере.

2. Анализ предметной области

2.1 Глоссарий

Доход — денежные средства, полученные пользователем, в результате какой-либо деятельности за определённый период времени.

Расход — денежные средства, потраченные пользователем на какую-либо категорию товаров или услуг.

Категория расходов — группа товаров или услуг.

Порог — максимальная сумма, которую пользователя намерен потратить в выбранной категории.

Акция — мероприятие компании по предоставлении скидочной цены на тот или иной продукт.

Валидность — соответствие требованиям, допустимость, правильность.

2.2 Анализ существующих решений

2.2.1 CoinKeeper

Является одним из ключевых игроков на рынке мобильных приложений по учёту расходов. При первом запуске сразу можно заметить большое количество настойчивой рекламы и навязывание акций компании, пример которых изображен на рис.1.

Приложение распространяется в GooglePlay на условно-бесплатной основе, что означает бесплатное использование всех функций в течение 15 дней, а в дальнейшем, при желании продолжать работу с приложением, необходимо приобрести полную версию.

Достоинства:

- Наличие основных функций;
- Автоматическая распланировка бюджета;
- Резервное копирование данных;
- Синхронизация с другими устройствами;
- Возможность установки пароля.

Недостатки:

- Медленная анимация, которая тормозит даже на современных устройствах.
- Отсутствие возможности получить визуальное представления о состоянии расходов\доходов в обобщенном плане в виде диаграммы на главном экране.

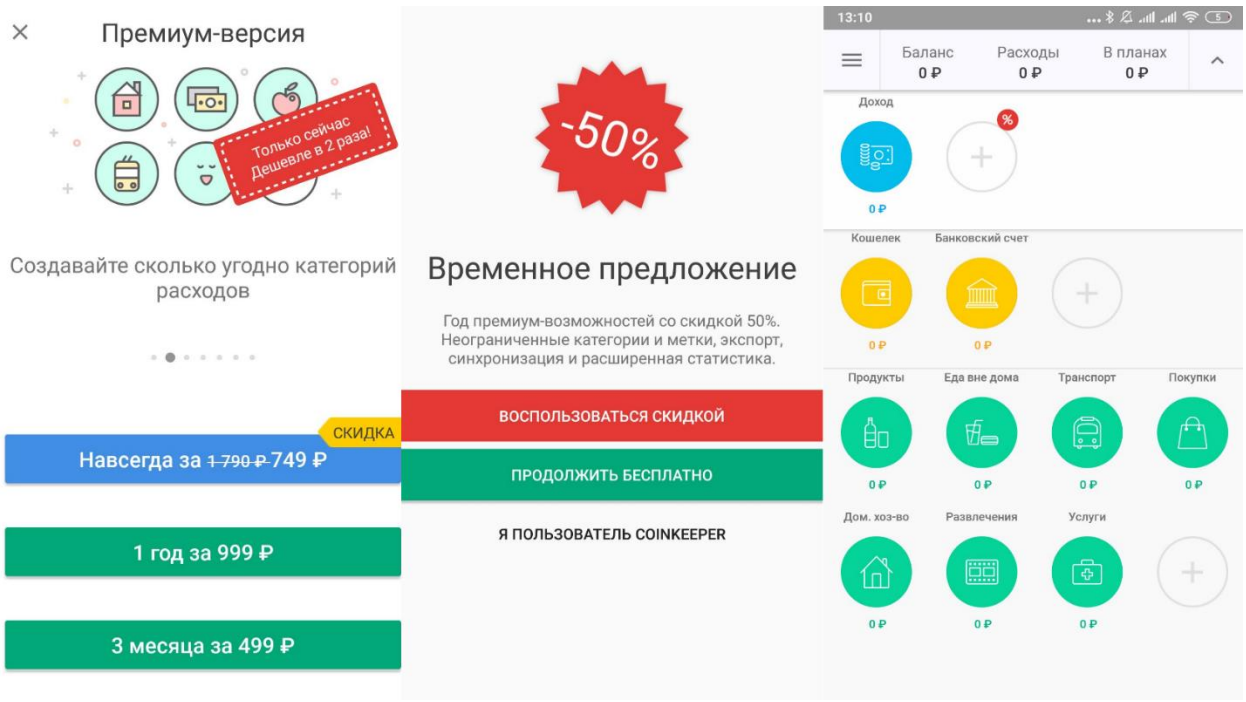

Рис. 1. Акции CoinKeeper и главный экран приложения

2.2.2 Monefy

Является самым визуально информативным из рассматриваемых приложений. При первом запуске сразу можно увидеть всю необходимую информацию в визуальном представлении (рис 2). Приложение не навязывает акции и дополнительные сервисы, а также распространяется по бесплатной модели с возможностью расширения имеющихся функций за дополнительную плату.

Достоинства:

- Наличие основных функций;
- Отсутствие необходимости в информационной справке;
- Синхронизация с другими устройствами;
- Возможность установки пароля.

Недостатки:

- Невозможность установить порог на расходы;

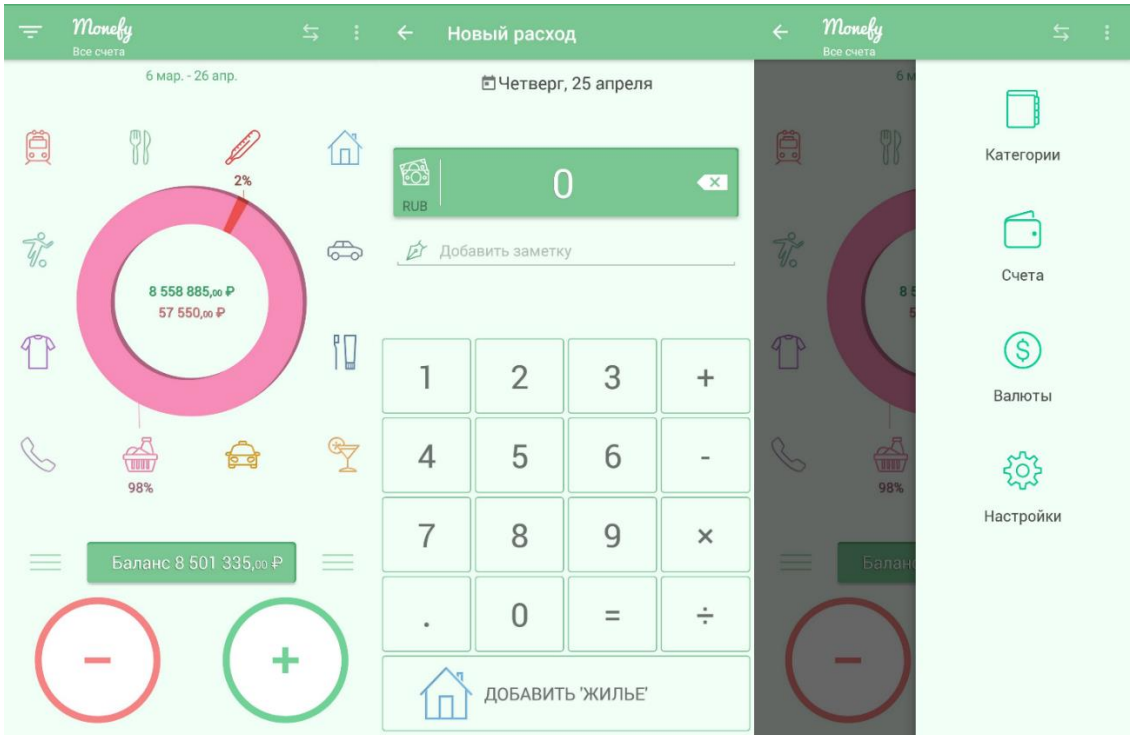

Рис.2. Интерфейс приложения Monefy

2.2.3 Fingen

Является приложением с самым большим количеством функций из рассматриваемых. Оно может сканировать и анализировать чеки, проводить автоматический учет расходов и доходов. Имеет перегруженный интерфейс. На данные момент распространяется по бесплатной модели с возможностью платного расширения в виде отчетов.

Достоинства:

- Максимально-возможный функционал;
- Возможность установки пароля.
- Сканирование чеков;
- Автоматический учет доходов/расходов;

Недостатки:

- Отсутствие отображения информации в виде диаграмм или графиков;
- Отчеты только в платной версии;
- Сложный интерфейс.

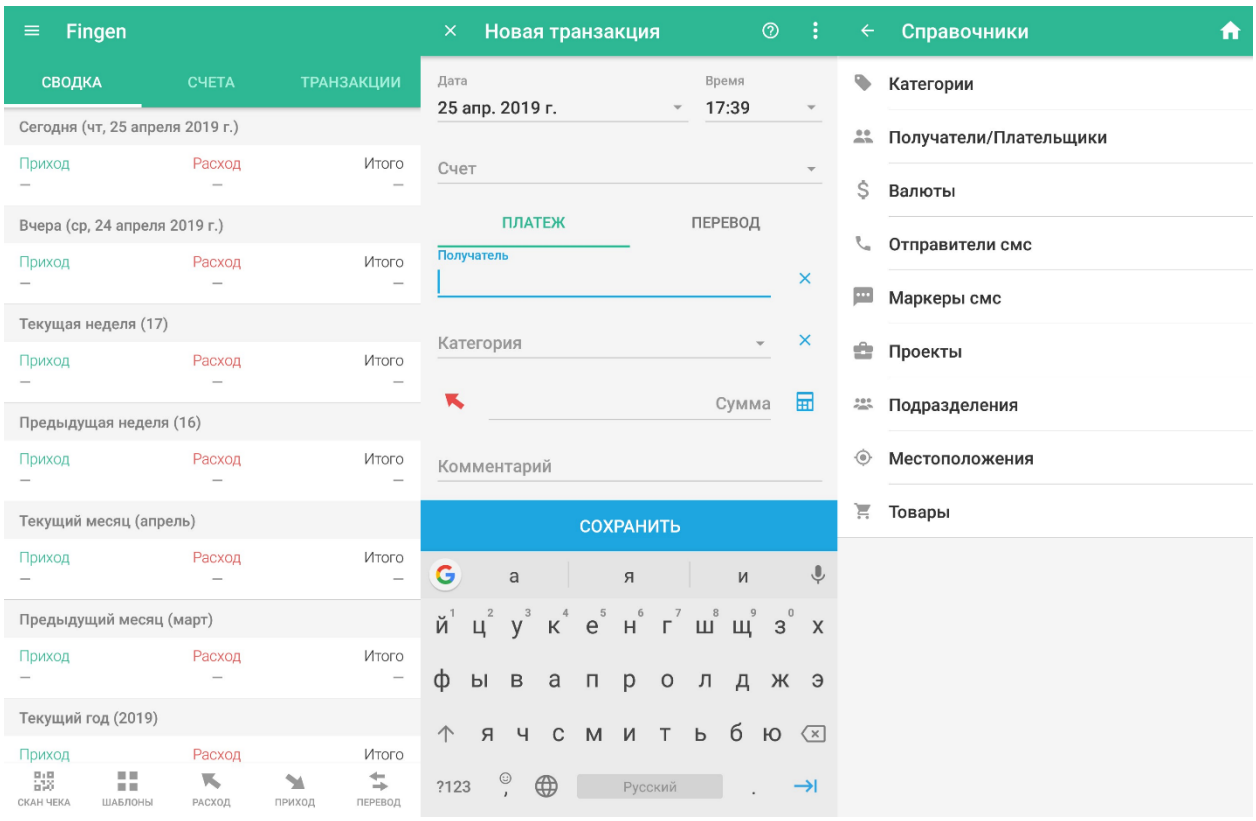

Рис.3. Интерфейс приложения Fingen.

2.3 Анализ задачи

2.3.1 Варианты использования приложения

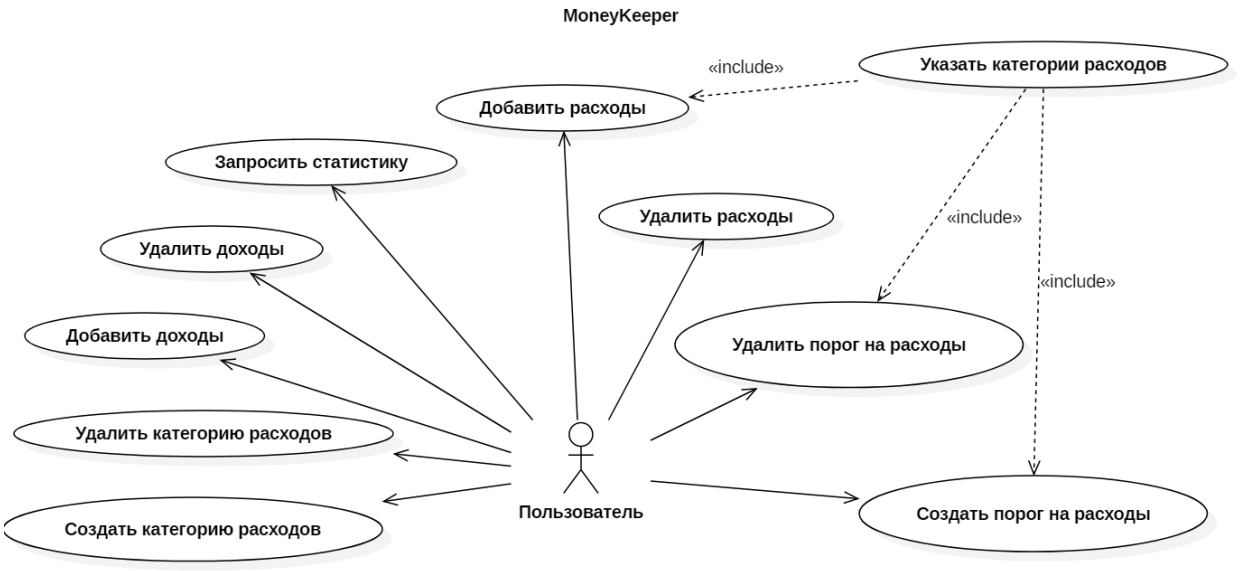

Рисунок 4. Диаграмма прецедентов.

При взаимодействии с приложением у пользователя есть определенный список возможностей, который более наглядно изображен на рисунке 1:

- Добавление Доходов
- Удаление Доходов
- Добавление Расходов
    - При добавлении расходов пользователь должен указать категорию
- Удаление Расходов
- Добавление Категорий Расходов
- Удаление Категорий Расходов
- Добавление порогов расходов
    - При добавлении порога на расход пользователь должен указать конкретную категорию расходов

- Удаление порогов расходов
    - При удалении порога на расход пользователь должен выбрать конкретную категорию расходов
- Просмотр статистики

2.3.2 Взаимодействие компонентов системы

Рисунок 5. Диаграмма последовательности для просмотра статистики.

Рисунок 6. Диаграмма коммуникации для просмотра статистики.

На рисунке 6, показана диаграмма коммуникации, на которой явно указываются отношения между объектами при просмотре статистики в приложении, а на рисунке 5 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

Для просмотра статистики пользователь обращается к графическому интерфейсу приложения, который в свою очередь запрашивает статистику у менеджера статистики. Менеджер статистики делает запрос к базе данных, она же в свою очередь возвращает данные менеджеру статистики, который отправляет эти данные на графический интерфейс, который в свою очередь показывает пользователю запрашиваемую статистику.

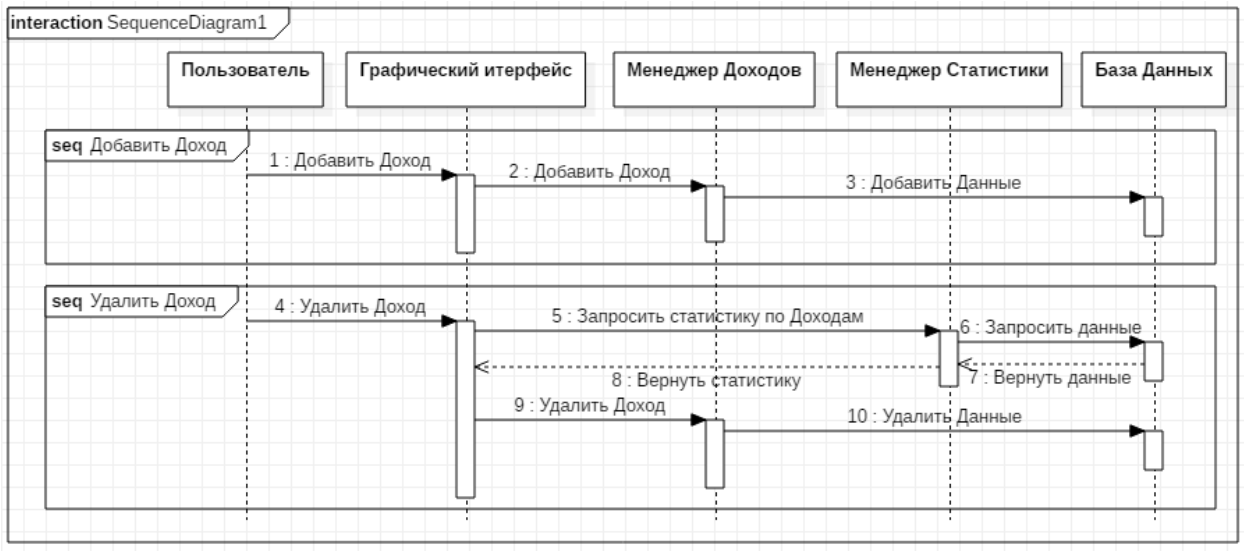

Рисунок 7. Диаграмма последовательности для добавления/удаления дохода.

Рисунок 8. Диаграмма коммуникации для добавления/удаления дохода.

На рисунке 8, показана диаграмма коммуникации, на которой явно указываются отношения между объектами при добавлении/удалении дохода в приложении, а на рисунке 7 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

Для добавления нового дохода пользователь обращается к графическому интерфейсу приложения, который в свою очередь просит менеджер доходов добавить соответствующий доход. Менеджер доходов осуществляет добавление данных в базу данных.

Для удаления существующего дохода пользователь обращается к графическому интерфейсу приложения, который в свою очередь делает запрос по доходам к менеджеру статистики. Менеджер статистики делает запрос по поводу данных к базе данных, которая возвращает данные менеджеру статистики. Менеджер посылает статистику по доходам обратно графическому интерфейсу, который выводит пользователю его, после чего просит менеджер доходов удалить выбранный пользователем доход. Менеджер дохода в свою очередь удаляет данные соответствующего дохода из базы данных.

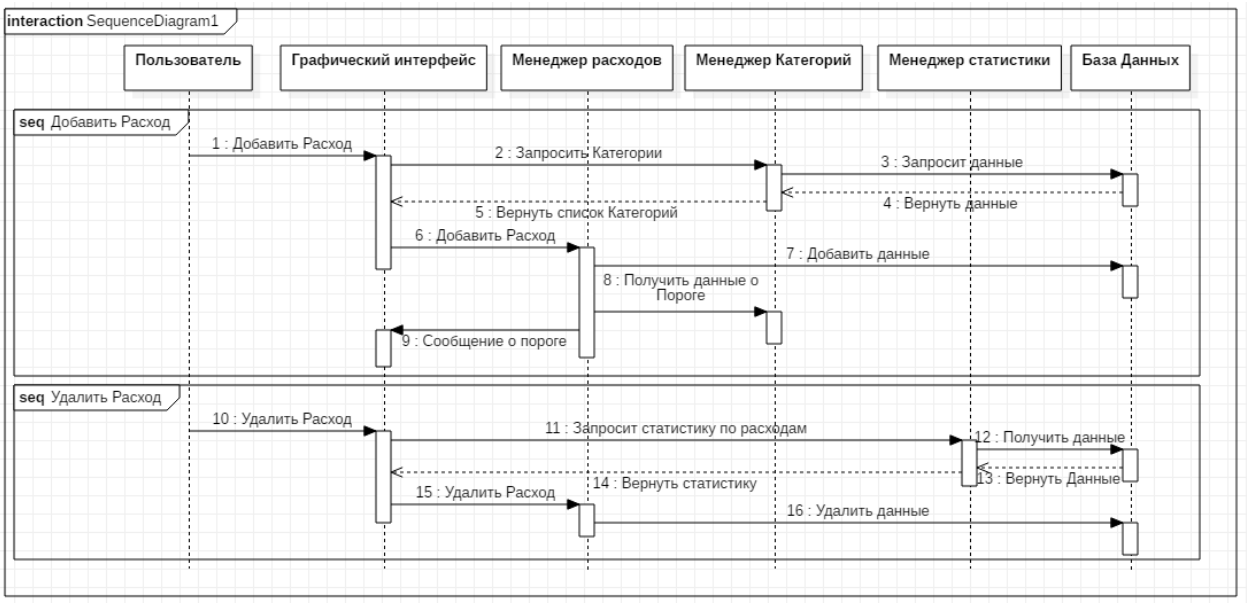

Рисунок 9. Диаграмма последовательности для добавления/удаления расхода.

Рисунок 10. Диаграмма коммуникации для добавления/удаления расхода.

На рисунке 10, показана диаграмма коммуникации, на которой явно указываются отношения между объектами при добавлении/удалении расхода в приложении, а на рисунке 9 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

Пользователь обращается к графическому интерфейсу с целью добавить расход, интерфейс запрашивает у менеджера категории, который в свою очередь запрашивает их у базы данных, после чего база данных возвращает их менеджеру категорий, а он графическому интерфейсу. После чего графический интерфейс просит менеджер расходов добавить расход пользователя в базу данных. После чего менеджер расходов сообщает менеджеру категорий информацию для порога расходов для данной категории и выводит сообщение о пороге в графический интерфейс приложения.

Если пользователь обращается к графическому интерфейсу с целью удалить расход, интерфейс в свою очередь делает запрос по расходам к менеджеру статистики. Менеджер статистики делает запрос по поводу данных к базе данных, которая возвращает данные менеджеру статистики. Менеджер посылает статистику по расходам обратно графическому интерфейсу, который выводит пользователю его, после чего просит менеджер расходов удалить

выбранный пользователем расход. Менеджер расходов в свою очередь удаляет данные соответствующего расхода из базы данных.

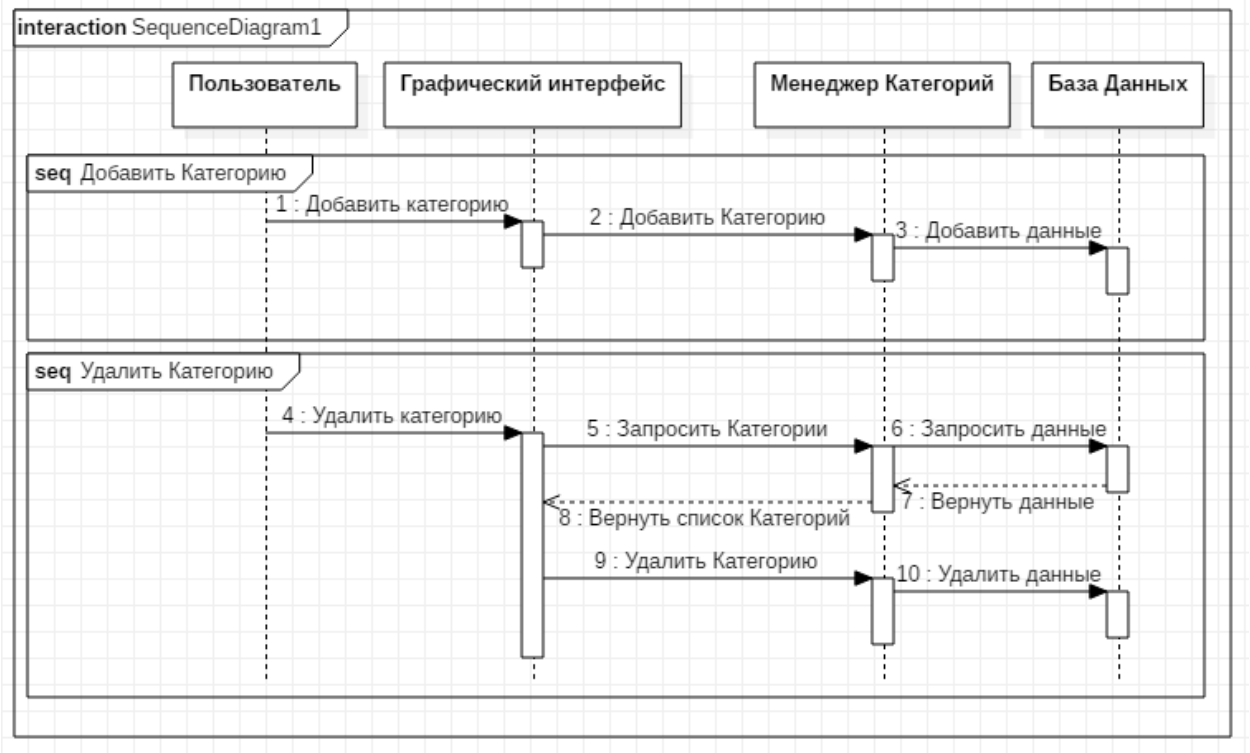

Рисунок 11. Диаграмма последовательности для добавления/удаления категории.

Рисунок 12. Диаграмма коммуникации для добавления/удаления категории.

На рисунке 12, показана диаграмма коммуникации, на которой явно указываются отношения между объектами при добавлении/удалении категории в приложении, а на рисунке 11 изображена диаграмма

последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

Для добавления новой категории пользователь обращается к графическому интерфейсу приложения, который в свою очередь просит менеджер категорий добавить соответствующую категорию. Менеджер категорий осуществляет добавление данных в базу данных.

Для удаления существующей категории пользователь обращается к графическому интерфейсу приложения, который в свою очередь делает запрос по категориям к менеджеру категорий. Менеджер категорий делает запрос по поводу данных к базе данных, которая возвращает данные менеджеру категорий. Менеджер посылает категории обратно графическому интерфейсу, который выводит пользователю его, после чего просит менеджер удалить выбранный пользователем категорию. Менеджер категорий в свою очередь удаляет данные из базы данных.

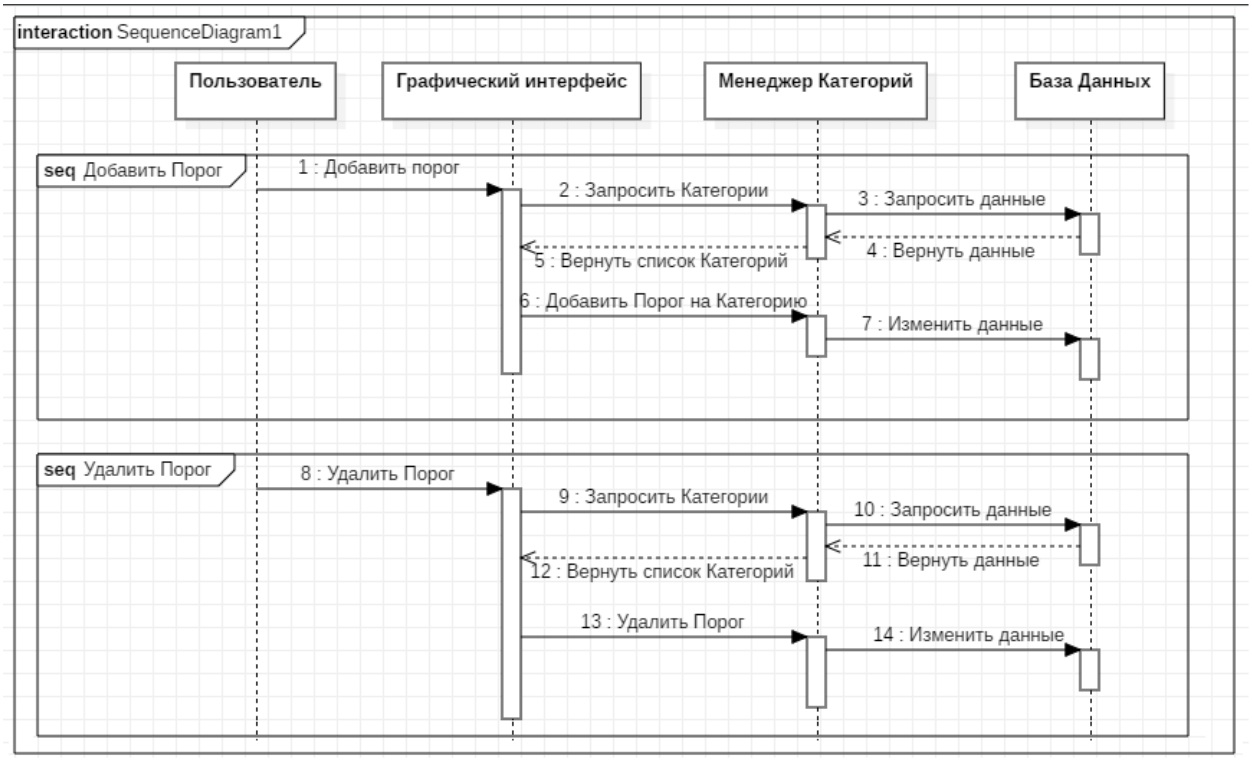

Рисунок 13. Диаграмма последовательности для добавления/удаления порога.

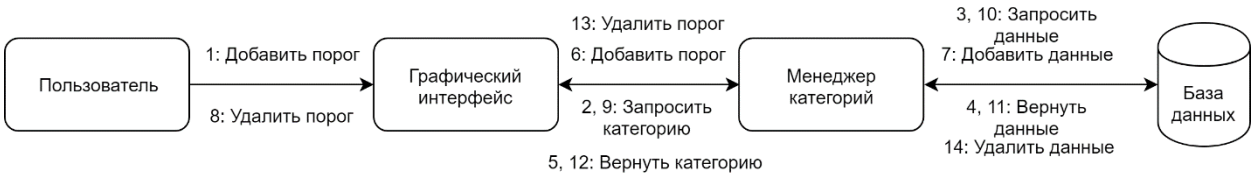

Рисунок 14. Диаграмма коммуникации для добавления/удаления порога.

На рисунке 14, показана диаграмма коммуникации, на которой явно указываются отношения между объектами при добавлении/удалении порога в приложении, а на рисунке 13 изображена диаграмма последовательности, на которой изображено упорядоченное во времени взаимодействие объектов.

Для добавления нового порога пользователь обращается к графическому интерфейсу приложения, который в свою очередь просит менеджер категорий добавить соответствующую категорию. Менеджер категорий делает запрос по поводу данных к базе данных, которая возвращает данные менеджеру категорий. Менеджер посылает категории обратно графическому интерфейсу, который выводит пользователю его, после чего просит менеджер добавить выбранный пользователем порог на расходы для данной категории. Менеджер в свою очередь добавляет данные в базу данных.

Для удаления существующего порога пользователь обращается к графическому интерфейсу приложения, который в свою очередь делает запрос по категориям к менеджеру категорий. Менеджер категорий делает запрос по поводу данных к базе данных, которая возвращает данные менеджеру категорий. Менеджер посылает категории обратно графическому интерфейсу, который выводит пользователю его, после чего просит менеджер удалить выбранный пользователем порог для категории. Менеджер категорий в свою очередь удаляет данные из базы данных.

2.3.3 Варианты состояния системы

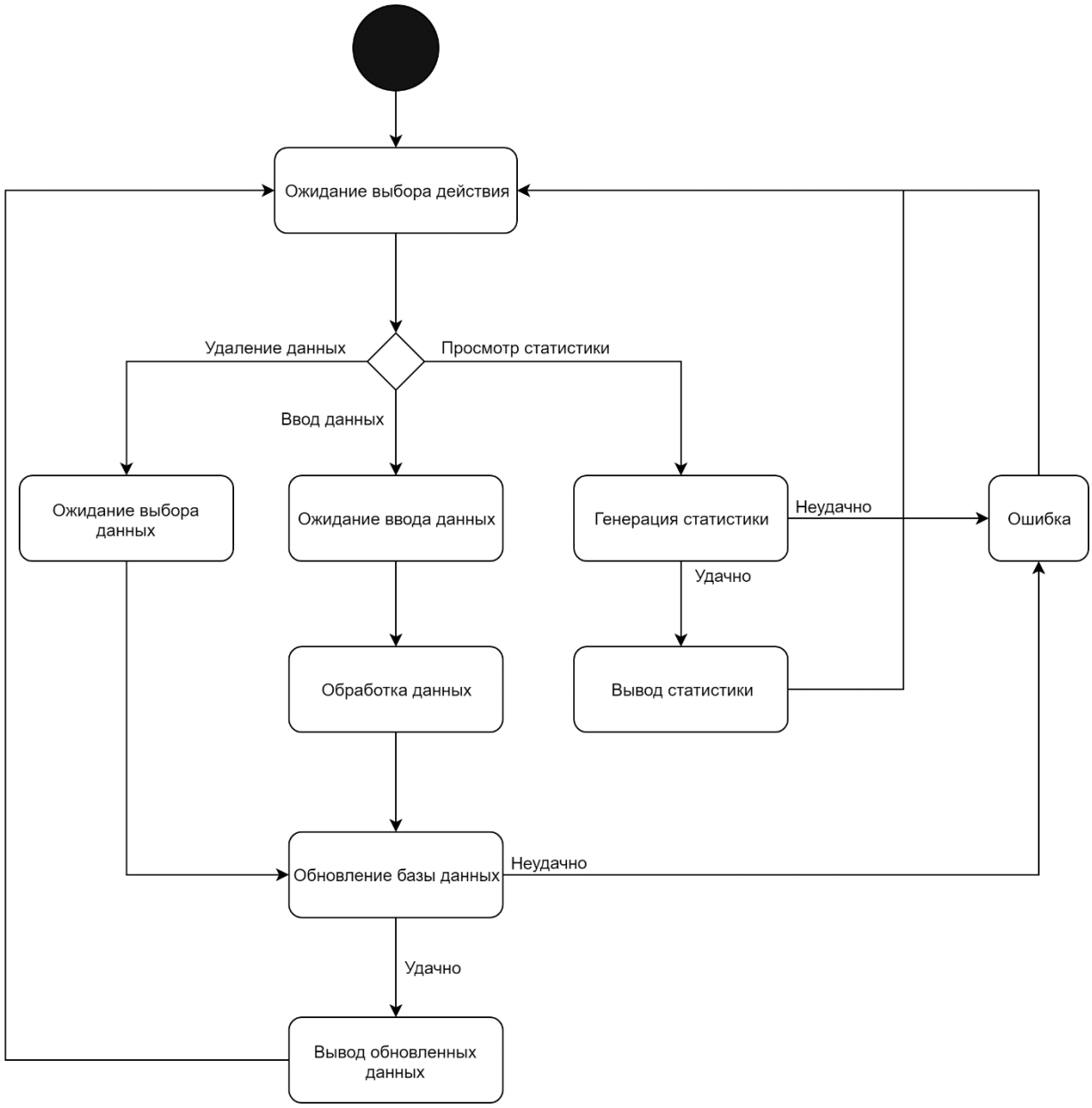

Рисунок 15. Диаграмма состояния.

Диаграмма состояний, изображенная на Рисунке 15, отражает возможные состояния системы. При запуске приложения система находится в ожидании выбора действия. В зависимости от выбора пользователя возможны 3 основные цепочки состояний:

- Ввод данных
- Удаление данных

- Просмотр статистики

Если пользователь выбирает опцию ввода данных, система переходит в состоянии ожидания ввода данных, после ввода данных пользователем система переходит в состояния обработки введенных данных, затем переходит в состояние обновления базы данных. Если все данные прошли проверки на валидность успешно, то система переходит в состояние вывода обновленных данных и переходит в состояния ожидания выбора новой опции от пользователя. Если же данные не прошли проверки, то система переходит в состояние вывода ошибки после чего переходит в ожидание выбора новой опции от пользователя.

Если пользователь выбирает опцию удаления данных, то система переходит в состояние ожидание выбора данных для удаления, после чего переходит в состояние обновления базы данных, в случае не успешного обновления переходит в состояние отображения ошибки и после переходит в состояние ожидания выбора действия. Если же обновление базы данных прошло успешно, то переходит в состояние отображения обновленных данных, после чего переходит в состояние ожидания выбора действия.

Если пользователь выбирает опцию просмотра статистики, то система переходит в состояние генерации статистики, в случае не успешной генерации переходит в состояние отображения ошибки и затем ожидания выбора действия. Если же статистика была сгенерирована без ошибок, то переходит в состояние вывода статистики и переходит в состояние ожидания выбора действия пользователя.

2.3.4 Варианты действия в системе

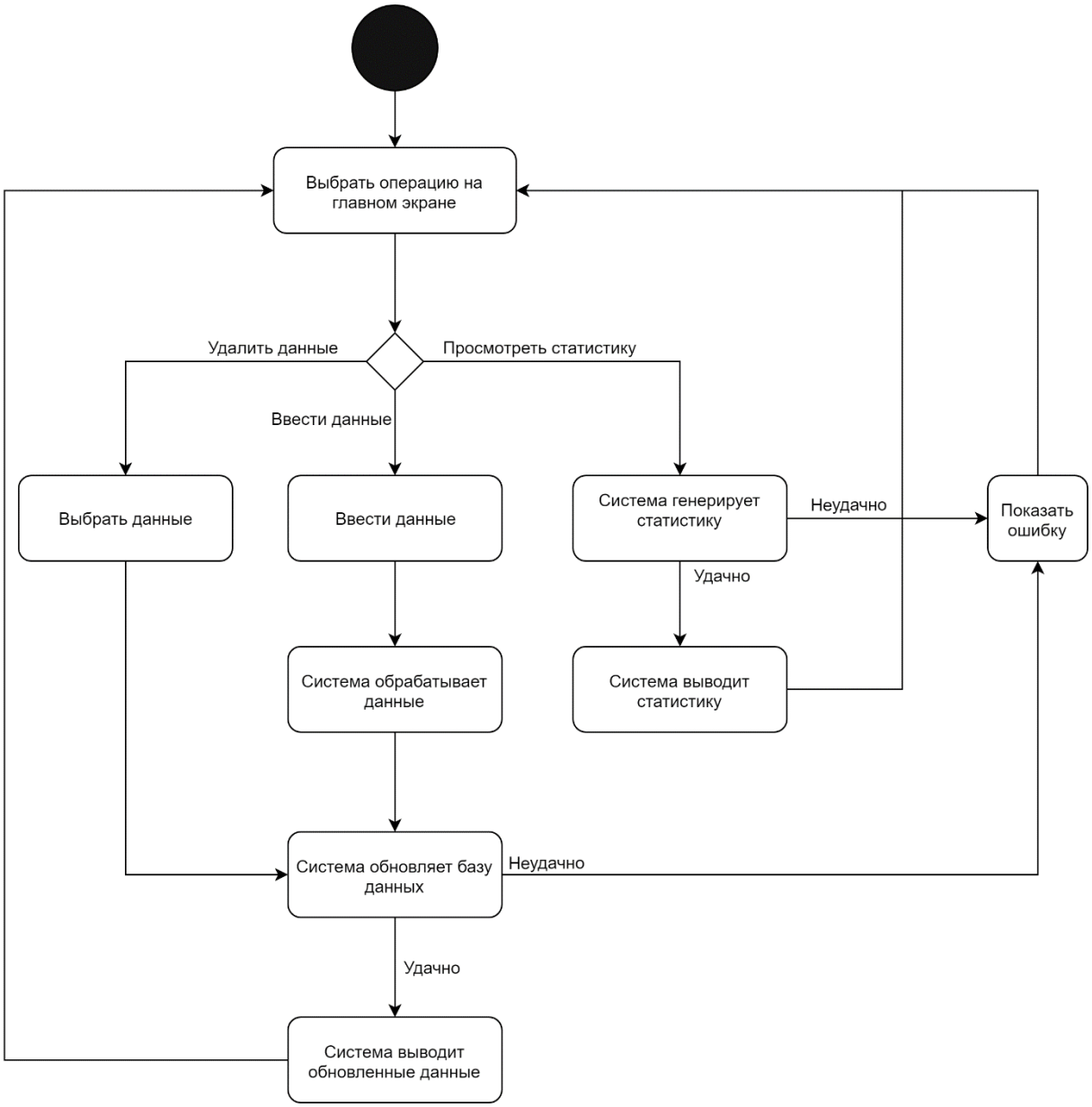

Рисунок 16. Диаграмма активности.

Диаграмма активности, изображенная на Рисунке 16, отражает возможные действия, состояния которых описаны на диаграмме состояния (Рисунок 15). При запуске приложения пользователь должен выбрать действие

над системой. В зависимости от выбора пользователя возможны 3 основные цепочки действий:

- Ввод данных
- Удаление данных
- Просмотр статистики

Если пользователь выбирает опцию ввода данных, то ему необходимо ввести данные для добавления, после чего система занимается обработкой введенных пользователем данных и в последствие осуществляет обновление базы данных. В случае ошибки, система показывает пользователю окно ошибки и пользователь переходит на начальный экран выбора операции. Если же обновление базы данных прошло успешно, то система выводит пользователю обновленные данные и пользователь переходит на выбор операции на главном окне приложения.

Если пользователь выбирает удаление данных на главном экране, то ему необходимо выбрать данные, которые он хочет удалить из системы, после чего система обновляет базу данные, в случае возникновения ошибки система выведет окно ошибки пользователю с информацией, после чего пользователь переходит на главный экран, для выбора действия. Если обновление базы данных прошло успешно, то система выведет пользователю обновленные данные и пользователь перейдет на главный экран для выбора следующего действия.

Если пользователь выберет операцию просмотра статистики, то система сгенерирует ему статистику, в случае возникновении ошибки система покажет пользователю окно с ошибкой после чего пользователь перейдет на главный экран для выбора следующего действия. В случае успешной генерации статистики система выведет пользователю статистики, после чего пользователь переходит на главный экран для дальнейшего выбора действия.

2.3.5 Модель базы данных

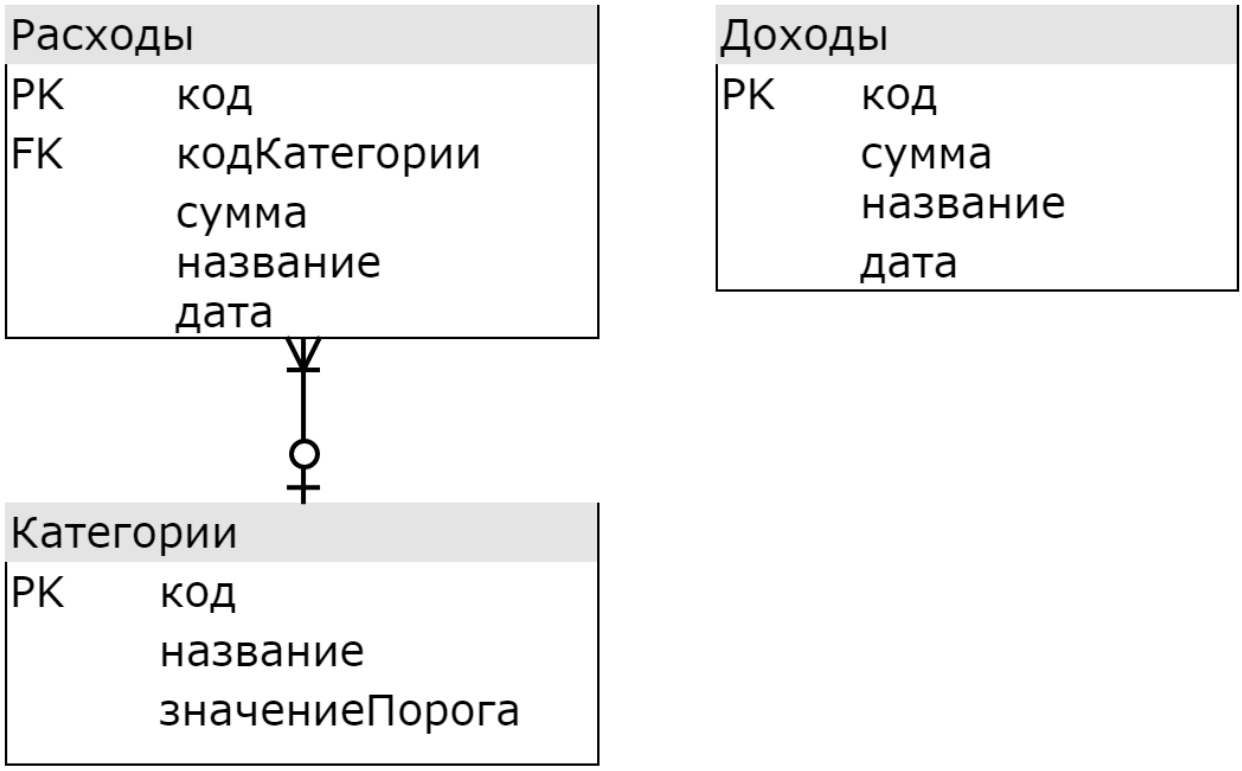

Рисунок 17. Схема базы данных

На рисунке 17 представлена схема базы данных для разрабатываемого приложения по контролю за расходами.

Данная база данных состоит из 3 таблиц:

- Расходы
- Доходы
- Категории

Таблица “Расходы” содержит следующие атрибуты:

- Код, который является первичным ключом
- Код категории, который является вторичным ключом, ссылающимся на таблицу “Категории”
- Сумма, которая определят размер расхода

- Название, которое является неким описанием расхода
- Дата, определяющие время записи расхода

Таблица “Категории” содержит следующие атрибуты:

- Код, который является первичным ключом
- Название самой категории
- Значение порога, которое определяет максимальное значение суммарного расхода по данной категории

Между таблицами “Расходы” и таблицей “Категории” имеется связь многие ко многим. Каждый расход обязательно относится к какой-то одной категории и к каждой категории может относится один и более расход, а может не одного не быть.

Таблица “Доходы” содержит следующие атрибуты:

- Код, который является первичным ключом
- Сумма, которая определят размер дохода
- Название, которое является неким описанием дохода
- Дата, определяющие время записи дохода

2.3.6 Развертывание приложения

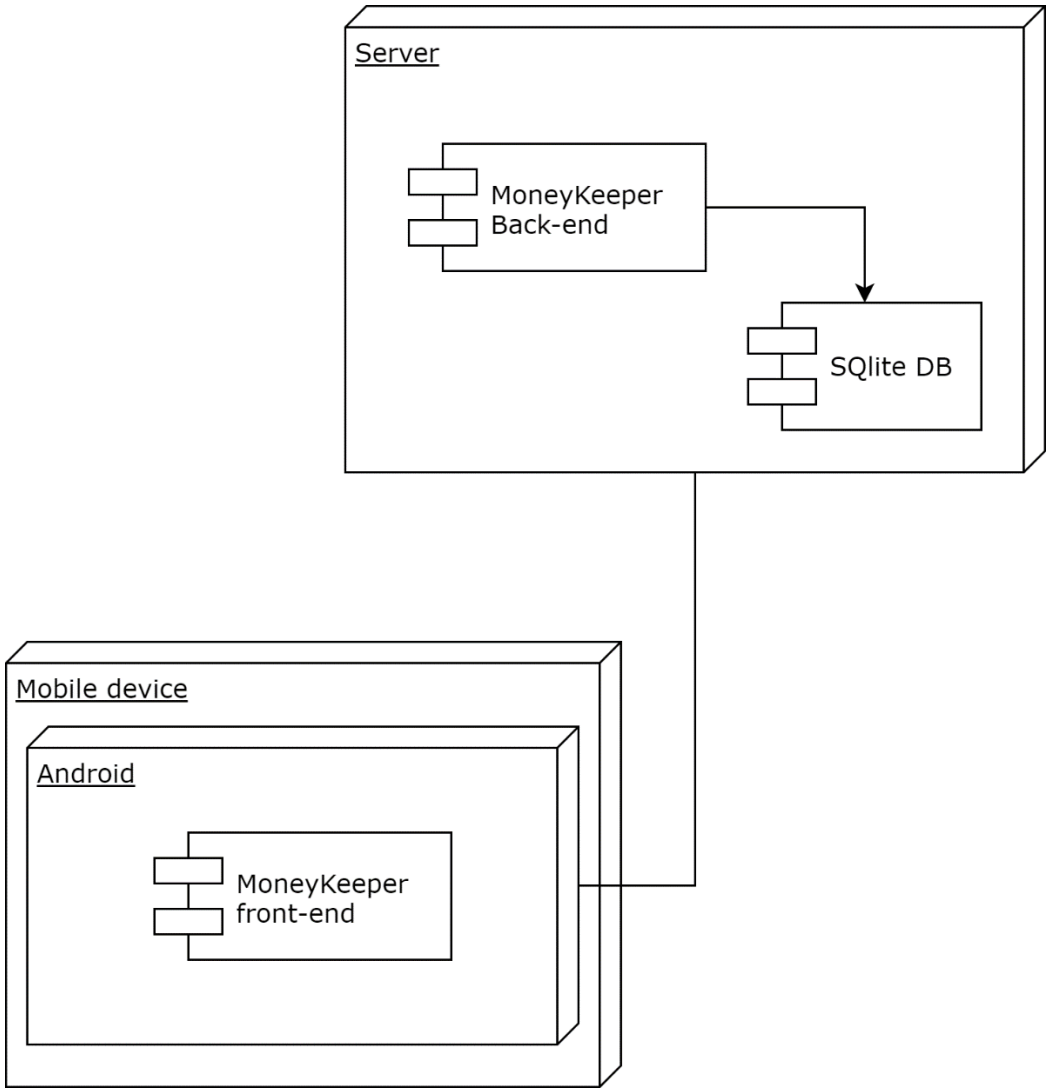

Рисунок 17. Диаграмма развертывания.

На Рисунке 17. представлена диаграмма развертывания, чтобы определить какие аппаратные компоненты («узлы») существуют, какие программные компоненты(«артефакты») работают на каждом узле и как различные части этого комплекса соединяются друг с другом.

Для разрабатываемого мобильного приложения узлом устройства является мобильное устройство и сервер, а в качестве узла среды выполнения

выступает операционная система Android. В операционной системе развернут front-end приложения, а на серверной части back-end и база данных.

3. Анализ средств реализации

В качестве средств реализации приложения были выбраны следующие технологии:

- 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.
- В качестве СУБД была выбрана SQLite, так Отличная при разработке и тестировании - в процессе разработки приложений часто появляется необходимость масштабирования. SQLite предлагает всё что необходимо для этих целей, так как состоит всего из одного файла и библиотеки, написанной на языке C.
- В качестве основного языка разработки использовался язык java, так как имеет весь необходимый для работы набор инструментов.

Приложение разрабатывается под версии Android 4.1 и выше, по причине того, что больше 99% пользователей работают на устройствах под этой системой, а оставшийся 1% на устройствах с уже не поддерживаемой версией Android.

Общение между Front-end и back-end происходит по средствам REST API, а обмен информации происходит с помощью передачи JSON файлов.

4. Реализация

4.1 Сущности

В приложении присутствуют три сущности: «Доход», «Расход» и «Категория расходов». Диаграмма классов, отражающих их отношения изображена на рисунке 18.

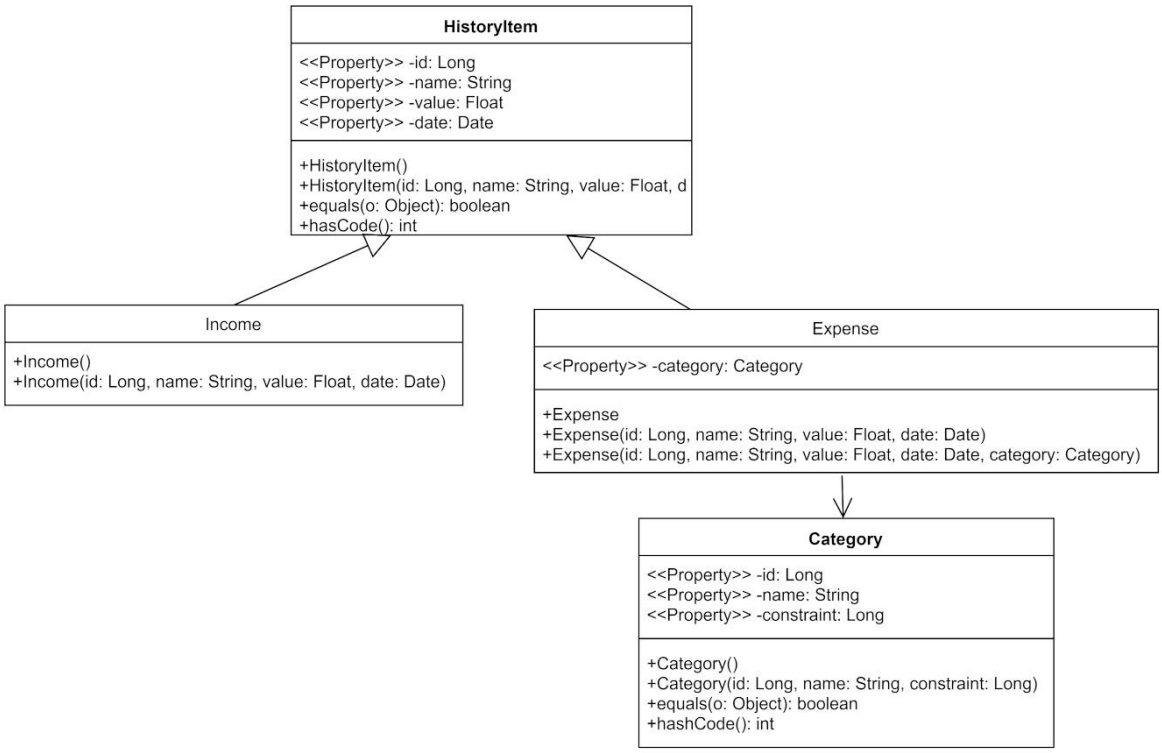

Рисунок 18. Диаграмма классов сущностей.

1. Класс «HistoryItem» – представляет собой базовый класс для классов «Income» и «Expense». Класс имеет следующие свойства:
   - «id» – уникальные идентификатор.
   - «name» – название дохода или расхода.
   - «value» – сумма дохода или расхода.
   - «date» – дата добавления дохода или расхода.
2. Класс «Income» – представляет собой отражение сущности «Доход» в программном коде;

3. Класс «Expense» – представляет собой отражение сущности «Расход» в программном коде. Помимо унаследованных свойств имеет свойство «category» – категория расхода;
4. Класс «Category» – представляет собой отражения сущности «Категория расходов». Класс имеет следующие свойства:
    - «id» – уникальные идентификатор.
    - «name» – название категории.
    - «constraint» – Порог для данной категории расходов.

4.2 Серверная часть приложения

Серверная часть приложения состоит из следующих классов бизнес логики:

1. «HistoryService» – Предоставляет возможность получить упорядоченный по времени добавления список доходов и расходов
2. «IncomeService» – Предоставляет возможность получить список доходов, добавлять и удалять доходы
3. «ExpenseServise» – Предоставляет возможность получить список расходов, добавлять и удалять расходы
4. «CategoryServise» – Предоставляет возможность получить список категорий, добавлять и удалять категории, а также изменять порог для категорий.

Серверная часть приложения состоит из следующих классов для работы с базой данных:

1. «DAO» – Базовый класс для классов работы с базой данных
2. «IncomeDAO» – Класс для работы с таблицей доходов базы данных
3. «ExpenseDAO» – Класс для работы с таблицей расходов базы данных
4. «CategoryDAO» – Класс для работы с таблицей категорий базы данных
   Диаграмма классов, отражающих их отношения изображена на рисунке 19.

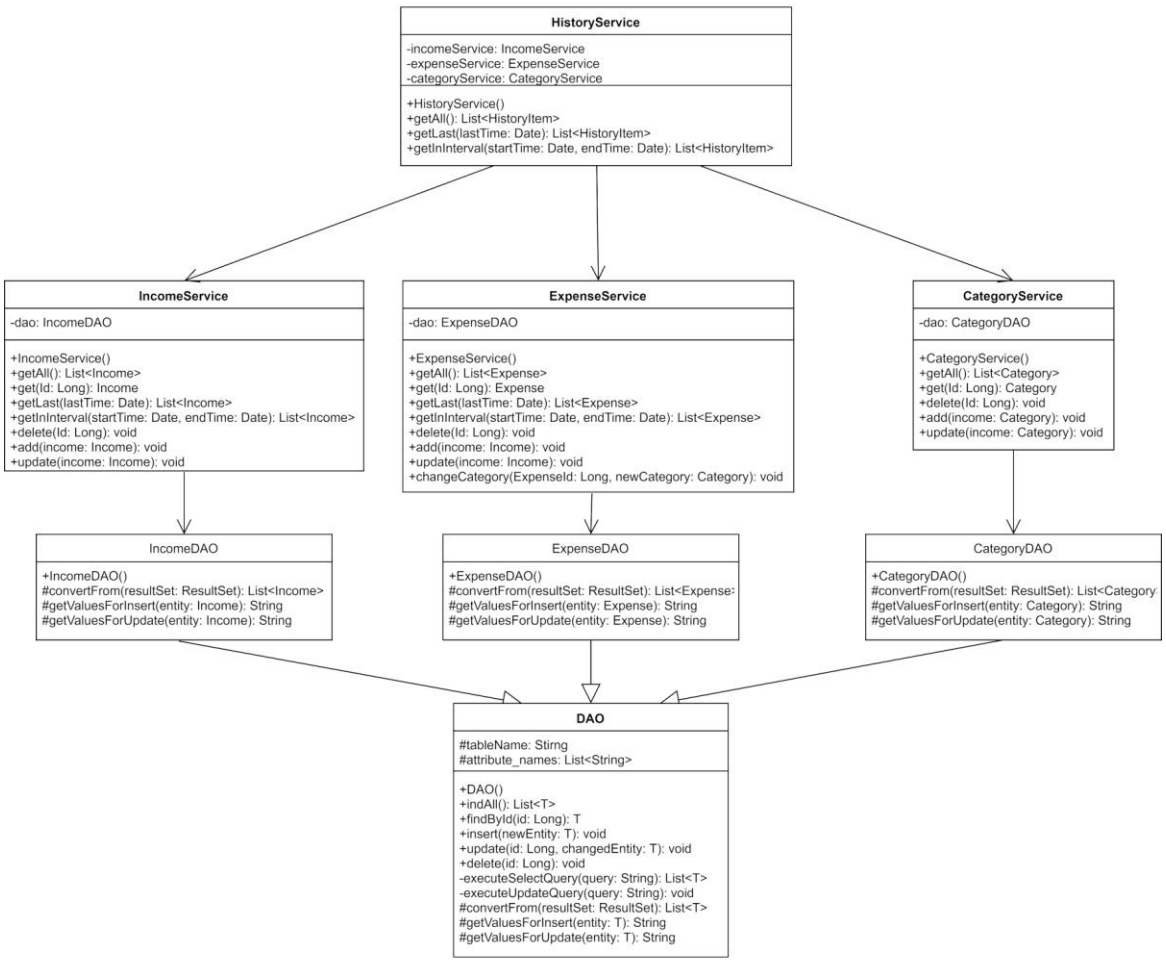

Рисунок 18. Диаграмма классов бизнес логики и работой с базой данных.

За реализацию REST API для сообщения с пользовательской частью приложения отвечают четыре REST контроллера.

«HistoryController» – контроллер, интерфейс которого представлен на рисунке 19, предоставляющий следующие возможность:

1. Получения списка доходов и расходов по средству обращения с помощью метода GET

```java
@RestController
public class HistoryController {
    private HistoryService service;
    private HistoryMapper mapper;

    @Autowired
    public HistoryController(HistoryService service, HistoryMapper mapper) {
        this.service = service;
        this.mapper = mapper;
    }

    @RequestMapping(value = "/history", method = RequestMethod.GET)
    public List<HistoryResponseDTO> getAll() {
        return mapper.historyItemListToHistoryItemResponseDtoList(service.getAll());
    }
}
```

Рисунок 19. Интерфейс контроллера «HistoryController»

«IncomeController» – контроллер, интерфейс которого представлен на рисунке 20, предоставляющий следующие возможность:

1. Получения доходов по средству обращения с помощью метода GET
2. Добавления доходов по средствам обращения с помощью метода POST, при этом в качестве параметра предается JSON файл, описывающий добавляемый доход
3. Удаления доходов по средствам метода DELETE, при этом в качестве параметра передается ID удаляемого дохода

```java
@RestController
public class IncomeController {
    private IncomeService service;
    private IncomeMapper mapper;

    @Autowired
    public IncomeController(IncomeService service, IncomeMapper mapper) {...}

    @RequestMapping(value = "/incs", method = RequestMethod.POST)
    public ResponseEntity<Long> create(@RequestBody IncomeRequestDTO requestDTO) {...}

    @RequestMapping(value = "/incs", method = RequestMethod.GET)
    public List<IncomeResponseDTO> getAll() {...}

    @RequestMapping(value = "/incs/{id}", method = RequestMethod.DELETE)
    public ResponseEntity delete(@PathVariable Long id) {...}
}
```

Рисунок 20. Интерфейс контроллера «IncomeController»

«ExpenseController» – контроллер, интерфейс которого представлен на рисунке 21, предоставляющий следующие возможность:

1. Получения расходов по средству обращения с помощью метода GET
2. Добавления расходов по средствам обращения с помощью метода POST, при этом в качестве параметра предается JSON файл, описывающий добавляемый расход
3. Удаления расходов по средствам метода DELETE, при этом в качестве параметра передается ID удаляемого расхода

```
@RestController
public class ExpenseController {
    private ExpenseService service;
    private ExpenseMapper mapper;

    @Autowired
    public ExpenseController(ExpenseService service, ExpenseMapper mapper) {...}

    @RequestMapping(value = "/exps", method = RequestMethod.POST)
    public ResponseEntity<Long> create(@RequestBody ExpenseRequestDTO requestDTO) {...}

    @RequestMapping(value = "/exps", method = RequestMethod.GET)
    public List<ExpenseResponseDTO> getAll() {...}

    @RequestMapping(value = "/exps/{id}", method = RequestMethod.DELETE)
    public ResponseEntity delete(@PathVariable Long id) {...}
}
```

Рисунок 21. Интерфейс контроллера «ExpenseController»

«CategoryController» – контроллер, интерфейс которого представлен на рисунке 22, предоставляющий следующие возможность:

1. Получения категорий по средству обращения с помощью метода GET
2. Добавления категорий по средствам обращения с помощью метода POST, при этом в качестве параметра предается JSON файл, описывающий добавляемую категорию

3. Удаления доходов по средствам метода DELETE, при этом в качестве параметра передается ID удаляемой категории
4. Изменение для категории порога по средствам метода PUT, при этом в качестве параметра передаются ID категории и новое значение порога.

```
@RestController
public class CategoryController {
    private CategoryService categoryService;
    private CategoryMapper categoryMapper;

    @Autowired
    public CategoryController(CategoryService categoryService, CategoryMapper categoryMapper) {...}

    @RequestMapping(value = "/categ", method = RequestMethod.POST)
    public ResponseEntity<Long> create(@RequestBody CategoryRequestDTO categoryRequestDTO) {...}

    @RequestMapping(value = "/categ", method = RequestMethod.GET)
    public List<CategoryResponseDTO> getAll() {...}

    @RequestMapping(value = "/categ/{id}/{cons}", method = RequestMethod.PUT)
    public ResponseEntity update(@PathVariable Long id, @PathVariable Float cons) {...}

    @RequestMapping(value = "/categ/{id}", method = RequestMethod.DELETE)
    public ResponseEntity delete(@PathVariable Long id) {...}

}
```

Рисунок 22. Интерфейс контроллера «CategoryController»

4.3 Графический интерфейс

4.3.1 Главный экран

Выводится при старте приложения и предоставляет следующие возможности для пользователя:

1. Узнать суммарное количество доходов и расходов
2. Узнать процентное распределение расходов по категориям
3. Перейти на экран историй с помощью нажатия на круговую диаграмму
4. Перейти на экран добавления доходов
5. Перейти на экран добавления расходов
6. Перейти на экран работы с категориями
7. Перейти на экран работы с порогами

Главный экран изображен на рисунке 23.

Рисунок 23. Главный экран

4.3.2 Экран истории

Представляет собой список с доходами и расходами, упорядоченный по времени добавления. Предоставляет следующие возможности для пользователя:

1. Посмотреть историю добавления расходов и доходов
2. Удалить доход или расход

Экран истории изображен на рисунке 24.

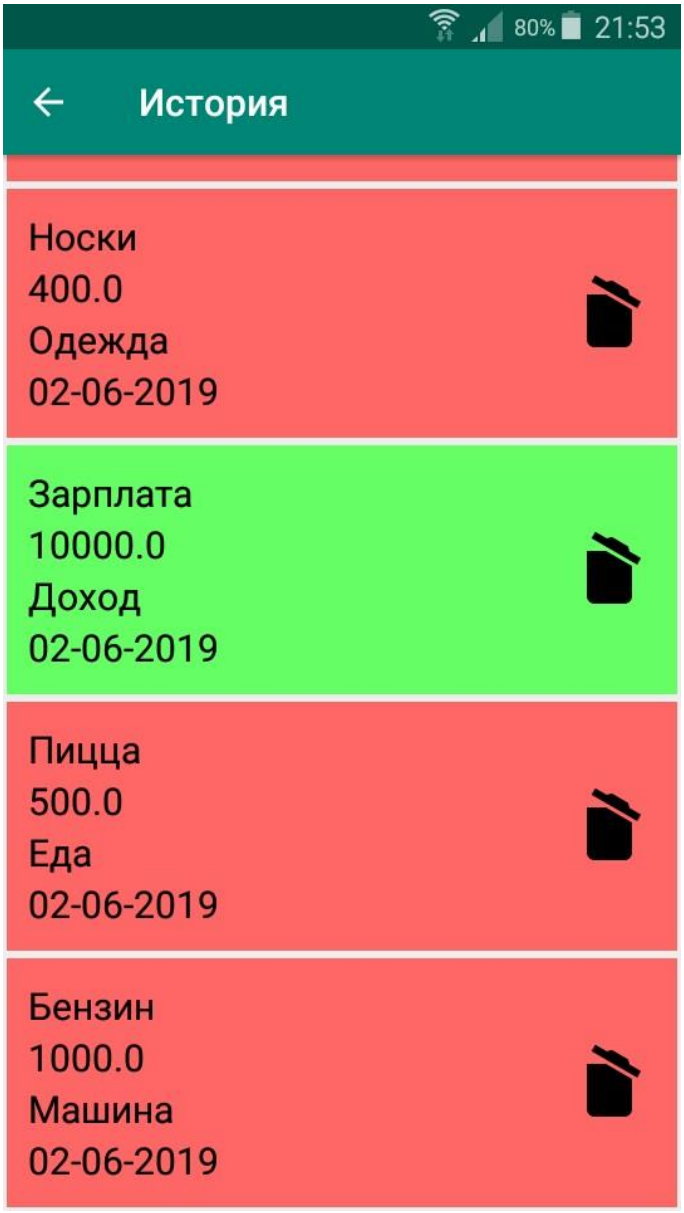

Рисунок 24. Экран истории

4.3.3 Экран добавления доходов

Предоставляет пользователю возможность добавления доходов.

Экран добавления доходов изображен на рисунке 25.

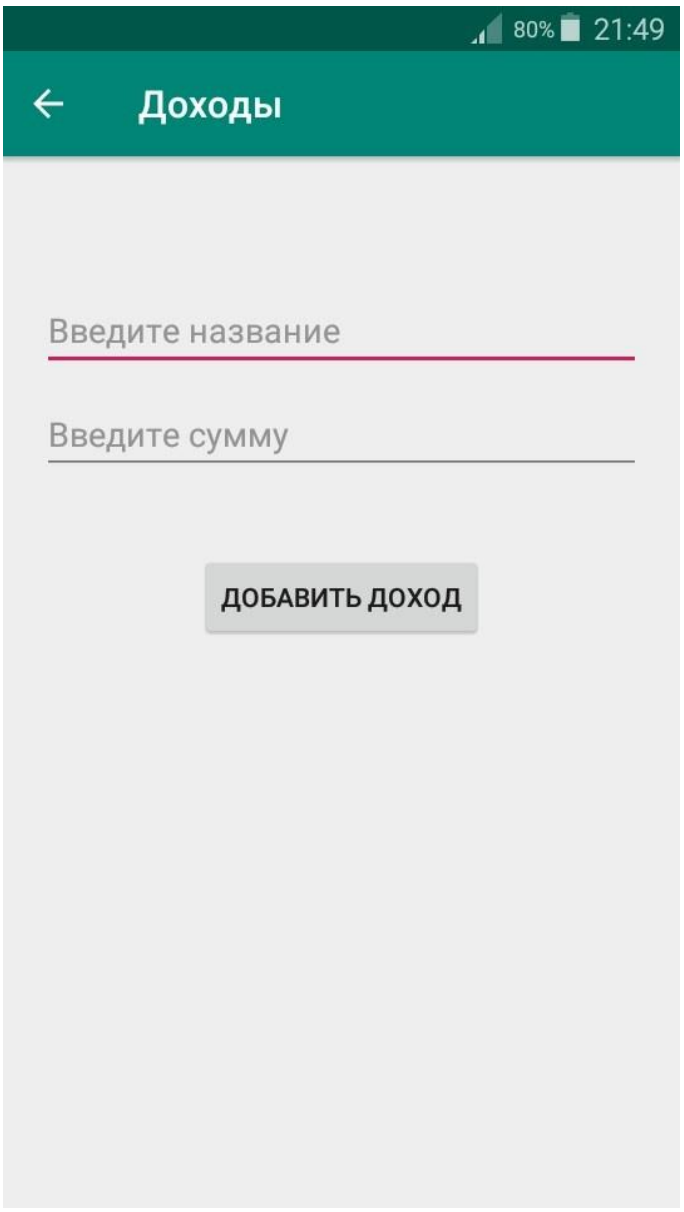

Рисунок 25. Экран добавления доходов

4.3.4 Экран добавления расходов

Предоставляет пользователю возможность добавления доходов.

Экран добавления доходов изображен на рисунке 26.

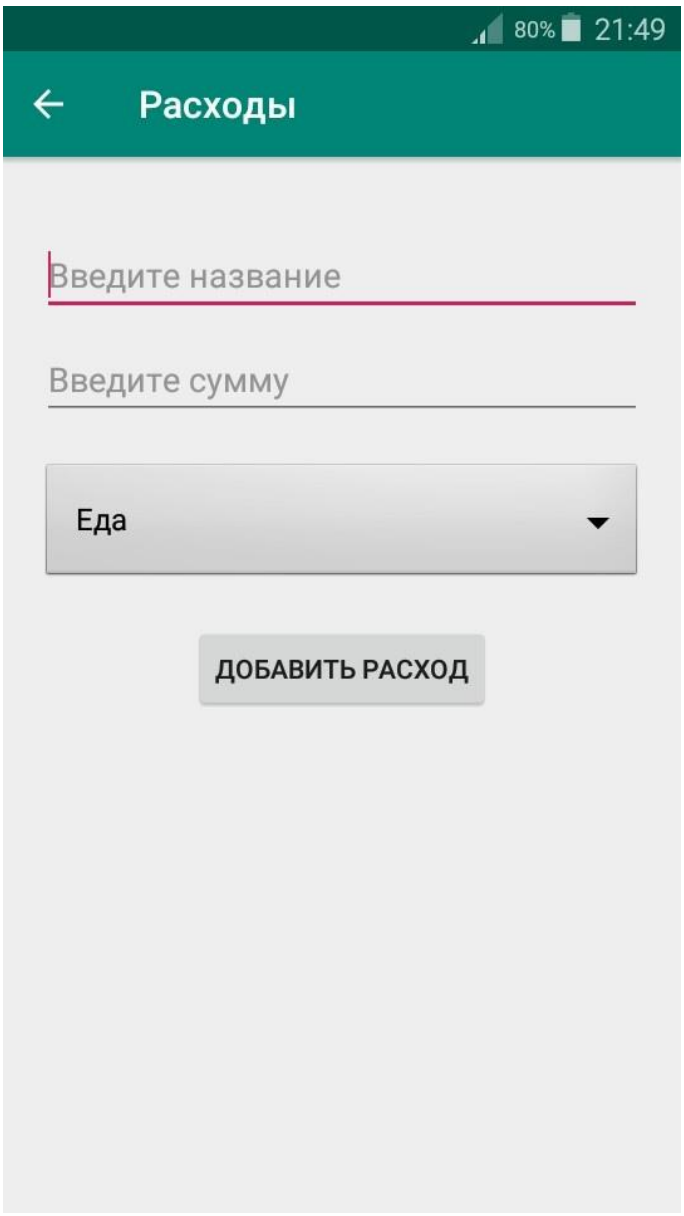

Рисунок 26. Экран добавления расходов

4.3.5 Экран работы с категориями

Предоставляет пользователю возможность добавлять и удалять категории.

Экран работы с категориями изображен на рисунке 27.

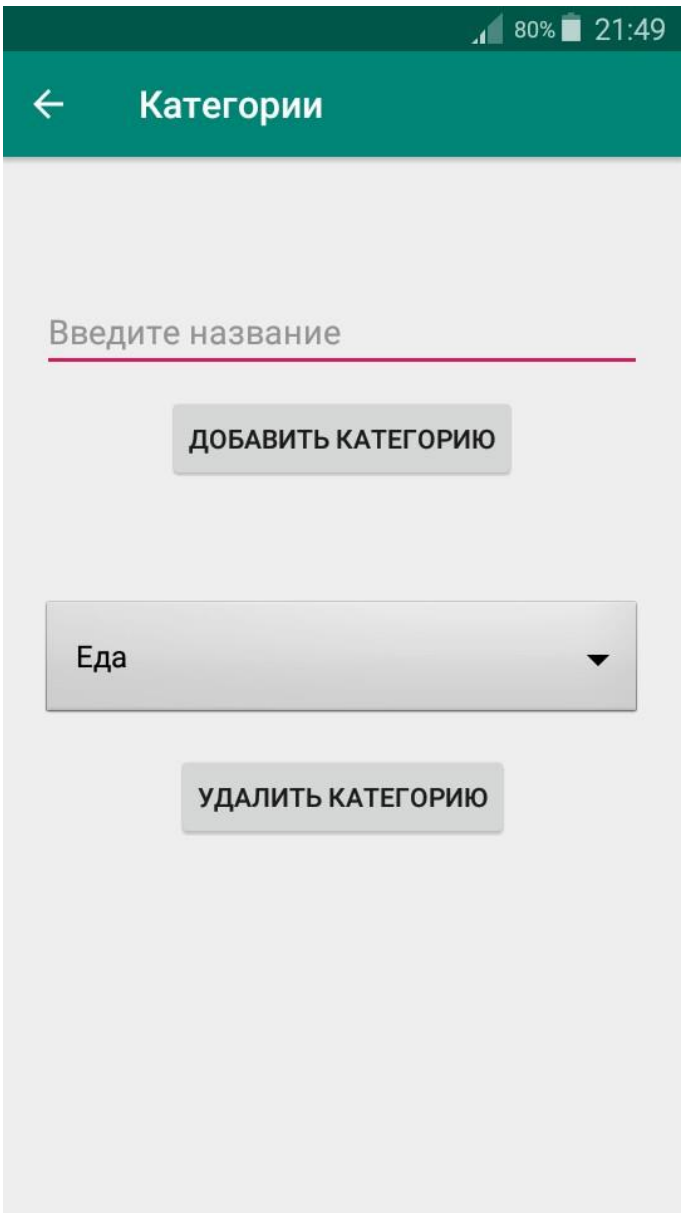

Рисунок 27. Экран работы с категориями

4.3.6 Экран работы с порогами

Предоставляет пользователю возможность добавлять и удалять пороги для категорий.

Экран работы с категориями изображен на рисунке 28.

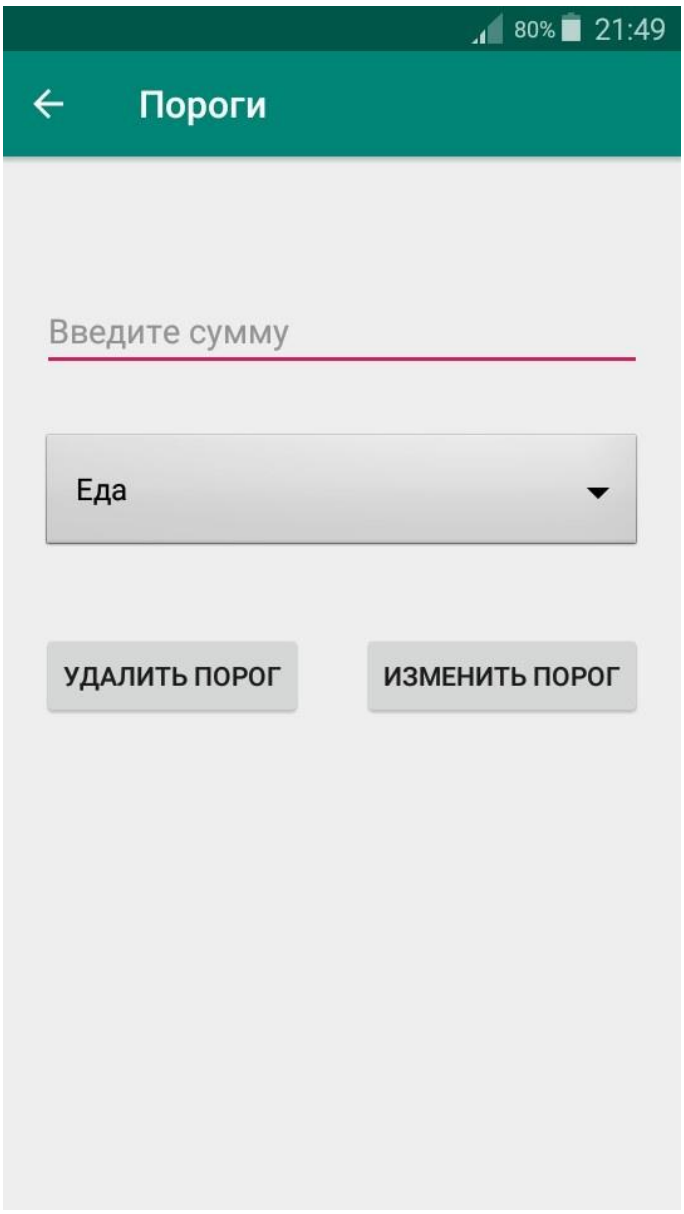

Рисунок 28. Экран работы с порогами

5. Тестирование

После реализации всех задач, был проведен запланированный набор тестов. Он включает 3 вида тестирования:

- Дымовое тестирование
- UI тесты
- Юзабилити тесты

5.1 Дымовое тестирование

Для данного тестирования необходимо было проверить работоспособность приложение на следующих основных сценариях:

- Добавление доходов;
- Удаление доходов;
- Добавление расходов;
- Удаление расходов;
- Добавление порога;
- Удаление порога;
- Отображение статистики;
- Добавление категории;
- Удаление категории;

Дымовое тестирование проводилось ручным способом, на заранее установленном приложении в Android 4.1 и 8.1, с включенным мобильным 4G интернетом для связи с back-end частью приложения. Результаты, полученные в ходе тестирования представлены в таблице 1.

Таблица 1 – Результаты дымового тестирования

Сценарий	**Результат**
Добавление доходов	Пройден
Удаление доходов	Пройден
Добавление расходов	Пройден
Удаление расходов	Пройден
Добавление порога	Пройден

Продолжение таблицы 1

Удаление порога	Пройден
Отображение статистики	Пройден
Добавление категории	Пройден
Удаление категории	Пройден

По итогу дымового тестирования было установлено, что приложение проходит все основные утвержденные сценарии.

5.2 UI тесты

В результате UI тестирования было выполнено шестнадцать тестов, охватывающие почти все возможности приложения.

Таблица 2 – Результаты UI тестирования

Шаги теста	Ожидаемый результат	Статус
1. Нажимается кнопка "Категории" 2. Нажимается кнопка возврата	Открылась вкладка "Категории". Возврат на главный экран	Пройден
1. Нажимается кнопка "Пороги" 2. Нажимается кнопка возврата	Открылась вкладка "Пороги". Возврат на главный экран	Пройден
1. Нажимается кнопка "Расходы" 2. Нажимается кнопка возврата	Открылась вкладка 'Расходы". Возврат на главный экран	Пройден
1. Нажимается кнопка "Доходы" 2. Переход на главный экран	Открылась вкладка Доходы. Возврат на главный экран	Пройден
1. В поле "Введите название" вводится "Еда" 2. Нажимается кнопка "Добавить категорию"	Появилось сообщение о том, что категория добавлена	Пройден
1. В поле "Введите название" вводится "" 2. Нажимается кнопка "Добавить категорию"	Появилось сообщение о том, что необходимо ввести название	Пройден
1. Нажимается кнопка "Удалить категорию"	Появилось сообщение о том, что категория удалена	Пройден
1. В поле "Введите сумму" вводится "100" 2. Нажимается кнопка "Изменить порог"	Появилось сообщение о том, что порог именен	Пройден
1. В поле "Введите сумму" вводится "" 2. Нажимается кнопка "Изменить порог"	Появилось сообщение о том, что необходимо ввести значение	Пройден

Продолжение таблицы 2

1. Нажимается кнопка "Удалить порог"	Появилось сообщение о том, что порог удален	Пройден
1. В поле "Введите название" вводится "Зарплата" 2. В поле "Введите сумму" вводится "5000" 3. Нажимается кнопка "Добавить доход"	Появилось сообщение о том, что доход добавлен	Пройден
1. В поле "Введите название" вводится "Зарплата" 2. В поле "Введите сумму" вводится "" 3. Нажимается кнопка "Добавить доход"	Появилось сообщение о том, что необходимо ввести значение	Пройден
1. В поле "Введите название" вводится "" 2. В поле "Введите сумму" вводится "5000" 3. Нажимается кнопка "Добавить доход"	Появилось сообщение о том, что необходимо ввести название	Пройден
1. В поле "Введите название" вводится "Пепси" 2. В поле "Введите сумму" вводится "75" 3. Нажимается кнопка "Добавить расход"	Появилось сообщение о том, что доход добавлен	Пройден
1. В поле "Введите название" вводится "Пепси" 2. В поле "Введите сумму" вводится "75" 3. Нажимается кнопка "Добавить расход"	Появилось сообщение о том, что необходимо ввести значение	Пройден

Продолжение таблицы 2

1. В поле "Введите название" вводится "Пепси" 2. В поле "Введите сумму" вводится "75" 3. Нажимается кнопка "Добавить расход"	Появилось сообщение о том, что необходимо ввести название	Пройден

5.3 Юзабилити тесты

Для проведения юзабилити тестирования было случайно отобрано 3 человека, не пользовавшиеся заранее приложением. Для данного тестирования необходимо проверить следующие основные сценарии взаимодействия пользователя с приложением:

- Добавление доходов;
- Удаление доходов;
- Добавление расходов;
- Удаление расходов;
- Добавление порога;
- Удаление порога;
- Проверка статистики;
- Проверка работы порога;
- Добавление категории;
- Удаление категории.

Результаты тестирования отображены в следующей таблице 3.

Таблица 3 – Результаты юзабилити тестирования

Сценарий	**Пользователь 1**	**Пользователь 2**	**Пользователь 3**
Добавление доходов	Пройден	Пройден	Пройден
Удаление доходов	Пройден	Пройден	Пройден
Добавление расходов	Пройден	Пройден	Пройден

Продолжение таблицы 3

Удаление расходов	Пройден	Пройден	Пройден
Добавление порога	Пройден	Пройден	Пройден
Удаление порога	Пройден	Пройден	Пройден
Проверка статистики	Не пройден	Не пройден	Пройден
Проверка работы порога	Пройден	Пройден	Пройден
Добавление категории	Пройден	Пройден	Пройден
Удаление категории	Пройден	Пройден	Пройден

В результате тестирования два пользователя не смогли разобраться как посмотреть статистику, в дальнейшем планируется разработать более интуитивный интерфейс для этого сценария.

Заключение

В результате работы было реализовано Android приложение, которое позволяет осуществлять учет доходов и расходов, а также имеет визуально информативный интерфейс, который позволяет получать всю необходимую информацию в виде диаграммы с главного экрана.

Были выполнены следующие задачи:

1. Разработана Front-end часть приложения, находящиеся на телефоне пользователя;
2. Разработана Back-end часть приложения, развернутая на удаленном сервере приложений;
3. Была создана связь между Front-end и Back-end частями приложения с помощью REST API;
4. Разработана базы данных, расположенная на удаленном сервере;