Минобрнауки россии

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных *наук*

Кафедра программирования и информационных технологий

Ведение группового и индивидуального бюджета

Курсовой проект

*09.03.02 Информационные системы и технологии*

*Программная инженерия в информационных системах*

Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_С. Д. Махортов, д.ф.- м.н.,доцент

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.А. Змаев, 3 курс, д/о

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.В. Мамонов, 3 курс, д/о

Обучающийся\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_П.А. Смирнов, 3 курс, д/о

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Тарасов, ст. преподаватель\_\_.\_.20\_\_

Воронеж 2023

**Содержание**

[Введение 4](#_Toc136879008)

[1 Постановка задачи 5](#_Toc136879009)

[1.1 Цели создания системы 5](#_Toc136879010)

[1.2 Требования к разрабатываемой системе 5](#_Toc136879011)

[1.3 Задачи проекта 5](#_Toc136879012)

[1.3.1 Для авторизованного пользователя 5](#_Toc136879013)

[1.3.2 Для неавторизованного пользователя 5](#_Toc136879014)

[1.3.3 Для создателя группы 5](#_Toc136879015)

[2 Анализ предметной области 6](#_Toc136879016)

[2.1 Терминология 6](#_Toc136879017)

[2.2 Обзор аналогов 7](#_Toc136879018)

[2.2.1 CoinKeeper 7](#_Toc136879019)

[2.2.2 Финансы 8](#_Toc136879021)

[2.2.3 Дзен-мани 9](#_Toc136879022)

[2.3 Моделирование системы 10](#_Toc136879024)

[2.3.1 Диаграмма в стиле методологии IDEF0 10](#_Toc136879025)

[2.3.2 Диаграмма прецедентов 11](#_Toc136879026)

[2.3.3 Диаграммы классов 12](#_Toc136879027)

[2.3.4 Диаграммы последовательности 13](#_Toc136879028)

[2.3.5 Диаграмма развертывания 17](#_Toc136879029)

[2.3.6 Диаграммы состояния 18](#_Toc136879030)

[2.3.7 Диаграмма объектов 19](#_Toc136879031)

[3 Реализация 20](#_Toc136879032)

[3.1 Средства реализации 20](#_Toc136879033)

[3.2 Реализация базы данных 20](#_Toc136879034)

[3.2.1 ER-диаграмма 21](#_Toc136879035)

[3.3 Реализация клиентской части 22](#_Toc136879036)

[3.3.1 Экраны счетов 23](#_Toc136879037)

[3.3.2 Экраны шаблонов 25](#_Toc136879038)

[3.3.3 Экраны финансовых операций 29](#_Toc136879039)

[3.3.4 Экран категорий 32](#_Toc136879040)

[3.3.5 Экран отчета 35](#_Toc136879041)

[3.3.6 Экраны кредитов 36](#_Toc136879043)

[3.3.7 Экраны группы 39](#_Toc136879044)

[3.3.8 Экран аккаунта пользователя 45](#_Toc136879045)

[3.3.9 Экран регистрации 48](#_Toc136879046)

[3.3.10 Экран входа 49](#_Toc136879047)

[3.3.11 Экран восстановления пароля 50](#_Toc136879048)

[3.3.12 Onboarding screen 51](#_Toc136879049)

[3.4 Серверная часть 52](#_Toc136879050)

[3.4.1 Структура модуля-приложения 53](#_Toc136879051)

[3.4.2 Модуль — приложение users 54](#_Toc136879052)

[3.4.3 Модуль — приложение operations 55](#_Toc136879053)

[3.4.4 Модуль — приложение groups 57](#_Toc136879054)

[3.4.5 Модуль — приложение onboard 58](#_Toc136879055)

[3.4.6 Аутентификация и авторизация 58](#_Toc136879056)

[3.4.7 Панель администратора 58](#_Toc136879057)

[4 Тестирование 62](#_Toc136879058)

[4.1.1 Интеграционное тестирование 62](#_Toc136879059)

[4.1.2 Тестирование репозиториев 62](#_Toc136879060)

[4.1.3 Тестирование ViewModel 62](#_Toc136879061)

[5 Аналитика 64](#_Toc136879062)

[Заключение 66](#_Toc136879063)

Введение

Существует множество вариантов ведения бюджета, как личного, так и нескольких людей (например, семейного): от обычной тетради до современного программного обеспечения. Тем не менее, одними из стандартных средств для ведения семейного бюджета является таблица Excel или ведения бюджета от руки в тетради. Такие способы занимают много времени, а иногда и достаточно сложны для человека. Затрагивая тему технического прогресса, надо отметить, что технологии заняли важное место в жизни человечества. Сейчас сложно обходиться в жизни без телефонов, планшетов, компьютеров, ноутбуков и т.д. Мобильное приложение для ведения семейного бюджета всегда возможно иметь под рукой и заполнять в любой момент времени, а также оно упрощает процессы подсчета и записи расходов и доходов. Контролировать доходы и расходы становится удобнее, проще, быстрее, программные средства приложения помогут наглядно увидеть отчеты о своем бюджете и выполненных финансовых операциях. К тому же, в ведение бюджета входит такая важная операция, как определенные выплаты по кредиту, которые тоже тяжело рассчитывать в тетрадке, поэтому приложение — помощник должно иметь несложный калькулятор для расчета кредитных платежей. Для достижения поставленных задач сначала надо рассмотреть аналогичные сторонние решения, а затем сформулировать полный список требований к данному приложению

1. Постановка задачи
   1. Цели создания системы

Целью курсовой работы является создание мобильного приложения для ведения личного и группового бюджета.

* 1. Требования к разрабатываемой системе
* Обеспечение учета доходов и расходов
* Обеспечение группировки индивидуальных финансовых операций
* Использование протокола передачи данных HTTP
* Построение приложения на трехуровневой архитектуре
  1. Задачи проекта
     1. Для авторизованного пользователя
* Ведение учета доходов и расходов
* Управление шаблонами для частых операций
* Расчет ежемесячного кредитного платежа
* Управление категориями финансовых операций
* Создание и сохранение отчета в CSV формат
* Формирование групп и приглашение в них пользователей для совместного отслеживания доходов и расходов
  + 1. Для неавторизованного пользователя
* Возможность просмотра экранов приложения для ознакомления
* Расчет ежемесячного кредитного платежа
  + 1. Для создателя группы
* Возможность создания, удаления группы
* Приглашение участников в группу
* Исключение участников из группы

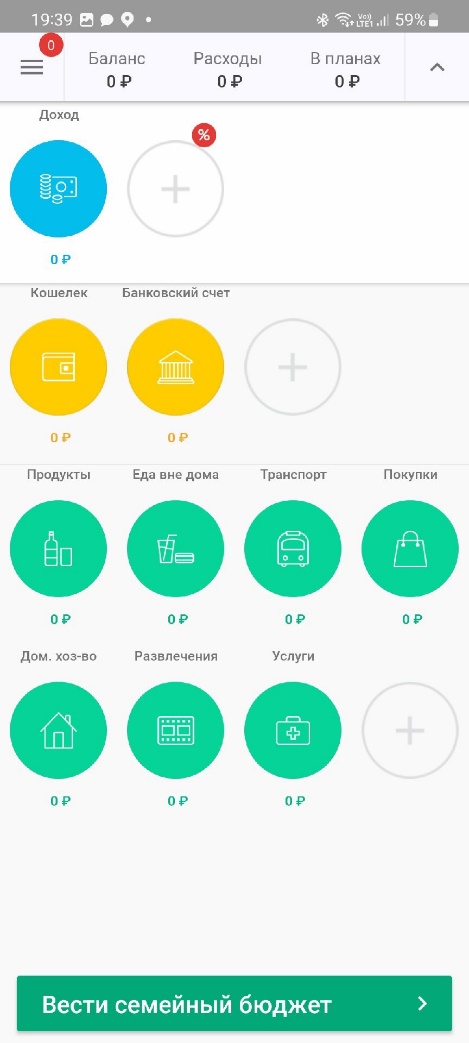
1. Анализ предметной области
   1. Терминология

* Мобильное приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для конкретной платформы (iOS, Android, WindowsPhone и т. д.).
* Android-приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для платформы Android.
* Клиент — это аппаратный или программный компонент вычислительной системы, посылающий запросы серверу.
* Сервер — выделенный или специализированный компьютер для выполнения сервисного программного обеспечения.
* База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД).
* HTTP — это протокол, позволяющий получать различные ресурсы, например HTML-документы. Протокол HTTP лежит в основе обмена данными в Интернете.
* SQL-запросы — это наборы команд для работы с реляционными базами данных.
* Дизайн-макет — это схематичное изображение финальной идеи с указанием всех деталей. В нем указываются концепция, шрифты, тексты, изображения, расположение всех элементов и общая картина продукта.
* Аутентификация — процедура проверки подлинности, например, проверка подлинности пользователя путем сравнения введенного им пароля с паролем, сохраненным в базе данных.
* Авторизация — предоставление определенному лицу или группе лиц прав на выполнение определенных действий.
* Android — это операционная система с открытым исходным кодом, созданная для мобильных устройств на основе модифицированного ядра Linux.
* Фреймворк — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.
* SQL-инъекция — внедрении в запрос произвольного SQL-кода, который может повредить данные, хранящиеся в БД или предоставить доступ к ним.
* HTTPS — расширение протокола HTTP для поддержки шифрования в целях повышения безопасности.
* Пользователь — человек, который использует приложение.
* Аккаунт или учетная запись — это персональная страница пользователя или личный кабинет, который создается после регистрации на сайте.
* Frontend — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.
* Backend — программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней части.
* REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети.
* API — описание взаимодействия одной компьютерной программы с другой.
  1. Обзор аналогов
     1. CoinKeeper

CoinKeeper – это приложение для ведения бюджета для Android и iOS, позволяет распределять доходы и расходы по категориям, создавать графики финансовых операций.

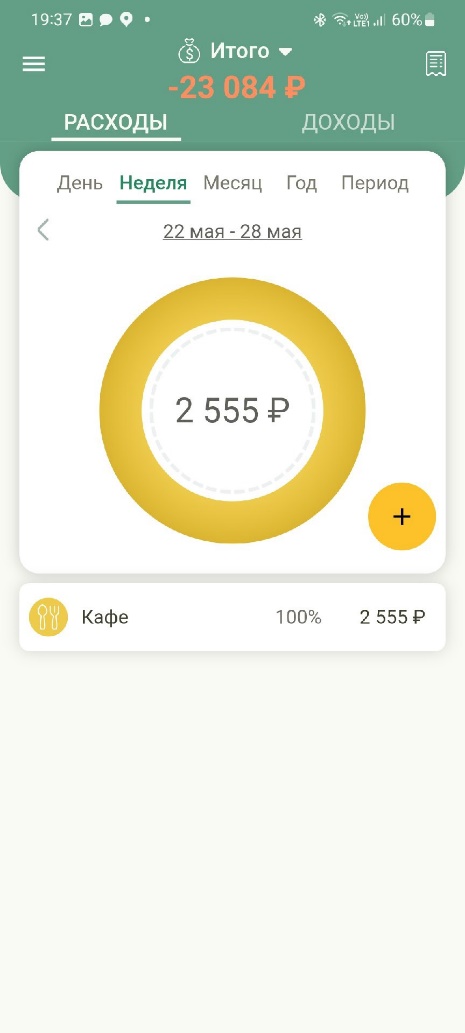
Недостатки:

* ограниченная функциональность бесплатного режима;
* отсутствие кредитного калькулятора;
* данные сохраняются локально;
* отсутствие экспорта финансовых операций в CSV формат.



1. Интерфейс приложения CoinKeeper
   * 1. Финансы

Финансы — это приложение, которое позволяет отслеживать расходы и доходы, управлять финансами и планировать бюджет.

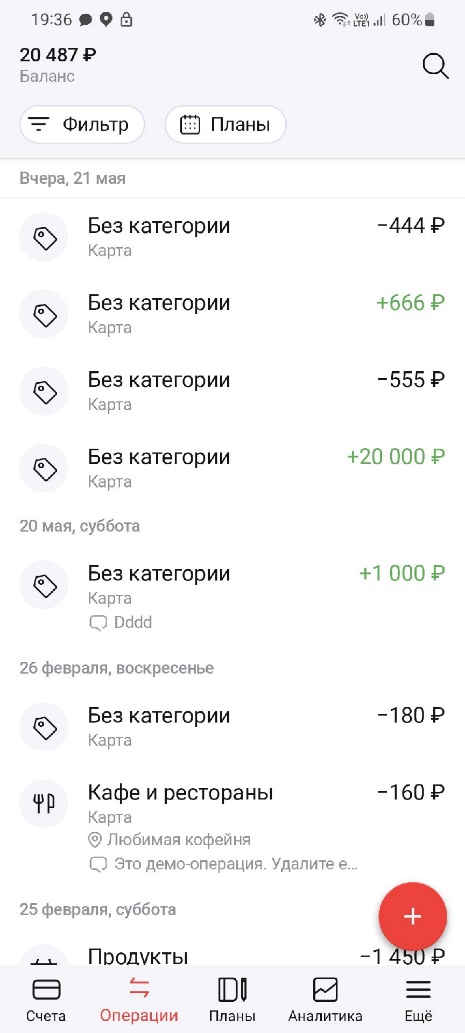


1. Интерфейс приложения Финансы

Недостатки:

* отсутствие экспорта финансовых операций в CSV формат;
* отсутствие кредитного калькулятора.
  + 1. Дзен-мани

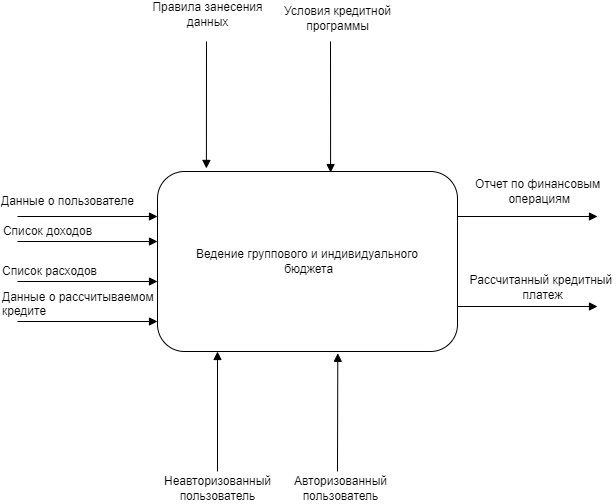
Дзен-мани — это веб-сервис, с помощью которого можно вести учёт личных финансов и планировать свой личный бюджет.



1. Интерфейс приложения Дзен-мани

Недостатки:

* ограниченная функциональность бесплатного режима;
* отсутствие кредитного калькулятора;
* отсутствие экспорта финансовых операций в CSV формат в бесплатном режиме.
  1. Моделирование системы
     1. Диаграмма в стиле методологии IDEF0



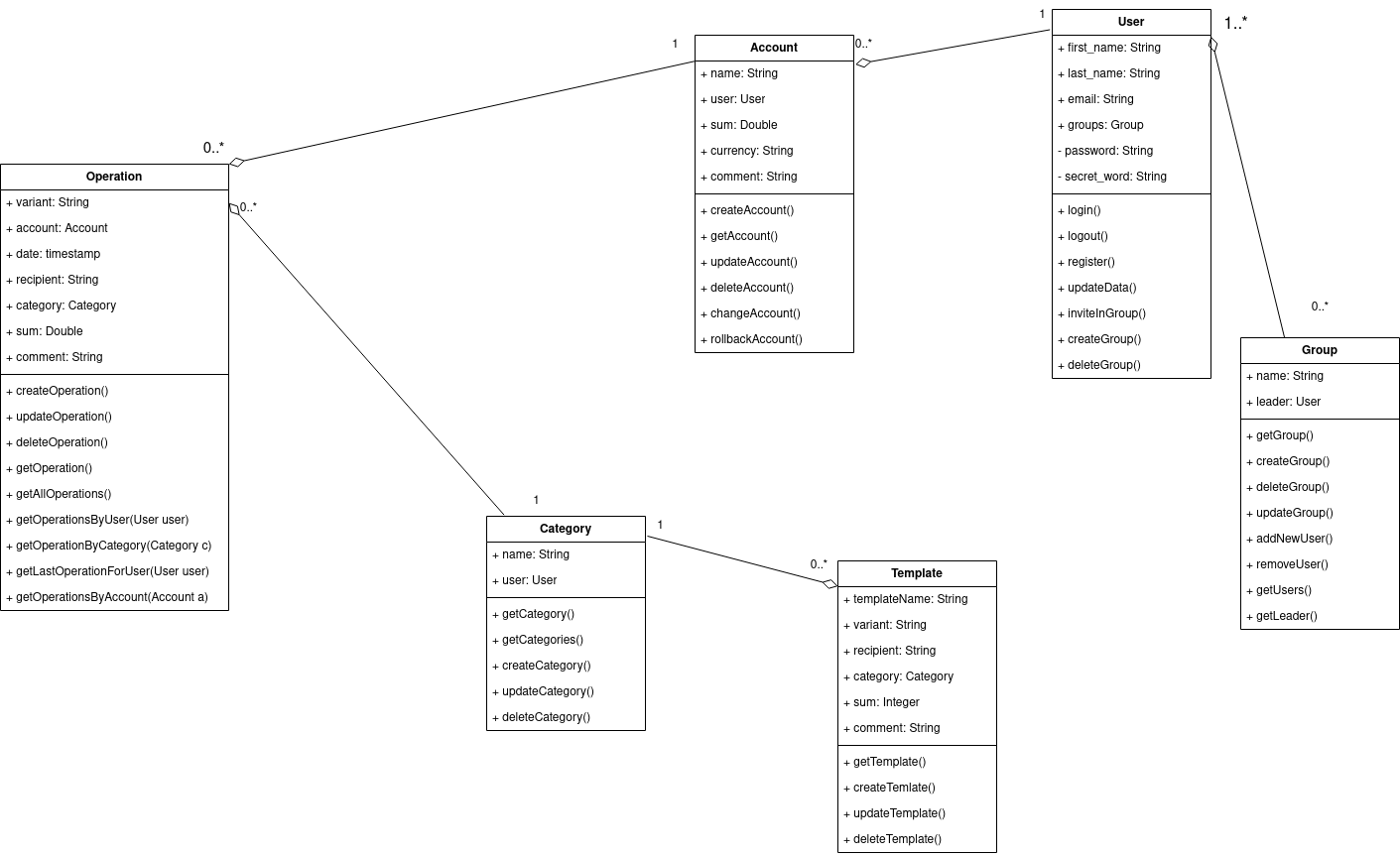
1. Диаграмма в стиле методологии IDEF0
   * 1. Диаграмма прецедентов

Рассмотрим полную диаграмму для использования приложения разными типами пользователей (неавторизованный, авторизованный, исполнитель). В данном случае необходимость составления диаграммы прецедентов продиктована прежде всего тем, что use-case диаграмма — это инструмент для моделирования системы и понимания ее функциональности и потребностей пользователей. Они помогают в определении основных действий, которые пользователь должен совершить в системе, чтобы достичь определенных целей. Они также позволяют определить возможные риски и проблемы, которые могут возникнуть в ходе использования системы.



1. Use-Case диаграмма пользования приложением
   * 1. Диаграммы классов

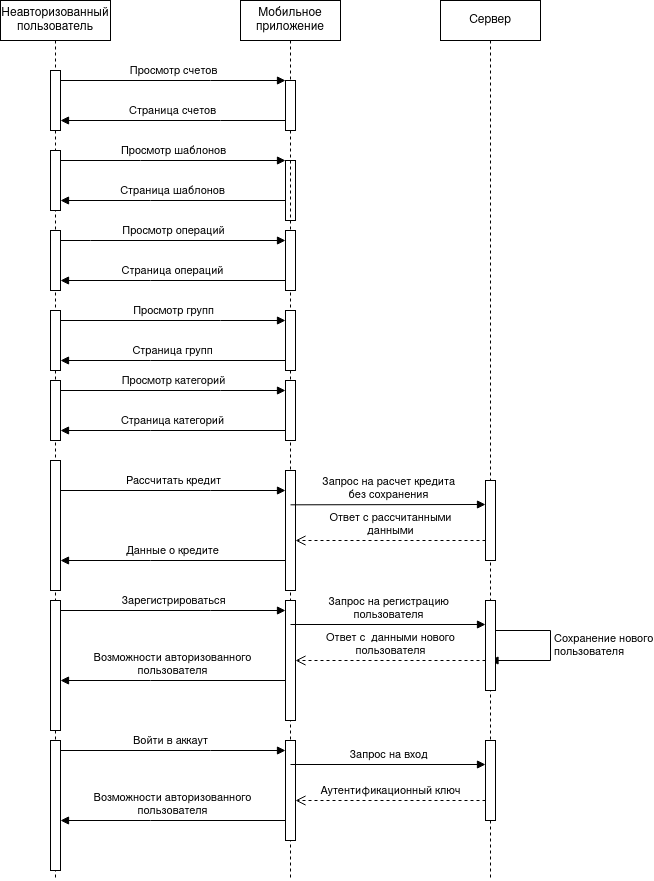
В данном пункте будут рассмотрены диаграммы классов-сущностей серверной части приложения. Диаграмма классов позволяет легко понять структуру проекта и отношения между классами. Это позволяет всем участникам проекта быть на одной волне, что позволяет сократить количество ошибок и повторных разъяснений в процессе разработки, помимо этого классы, которые созданы на диаграмме, обычно хорошо описываются, что позволяет легче отлаживать код и тестировать приложение в целом.



1. Диаграмма классов-сущностей серверной части приложения.
   * 1. Диаграммы последовательности

Диаграмма последовательности является важным инструментом для проекта, который помогает более глубоко понимать процесс, улучшать его эффективность и упрощать взаимодействие.

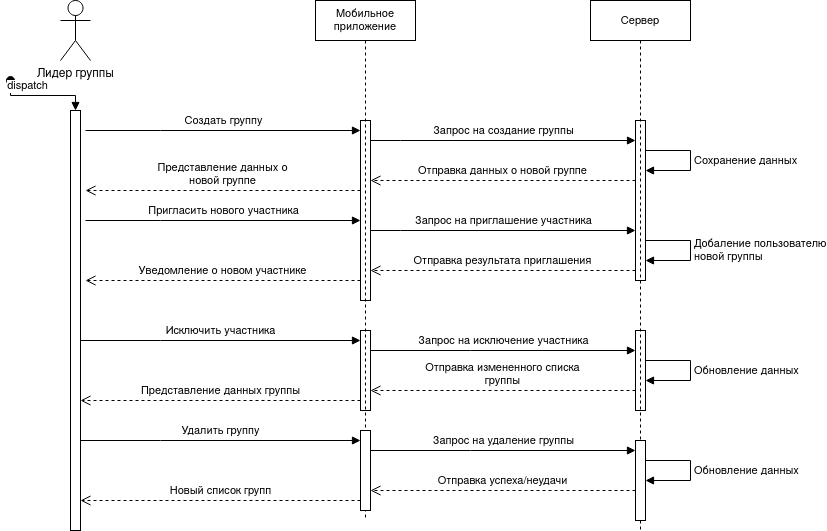
Рассмотрим диаграмму последовательности для пользователей разных типов:



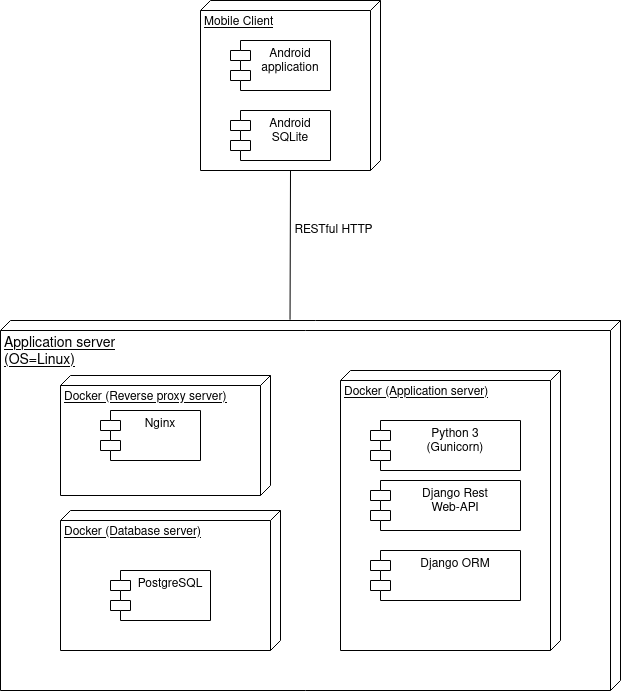
1. Диаграмма последовательности неавторизованного пользователя



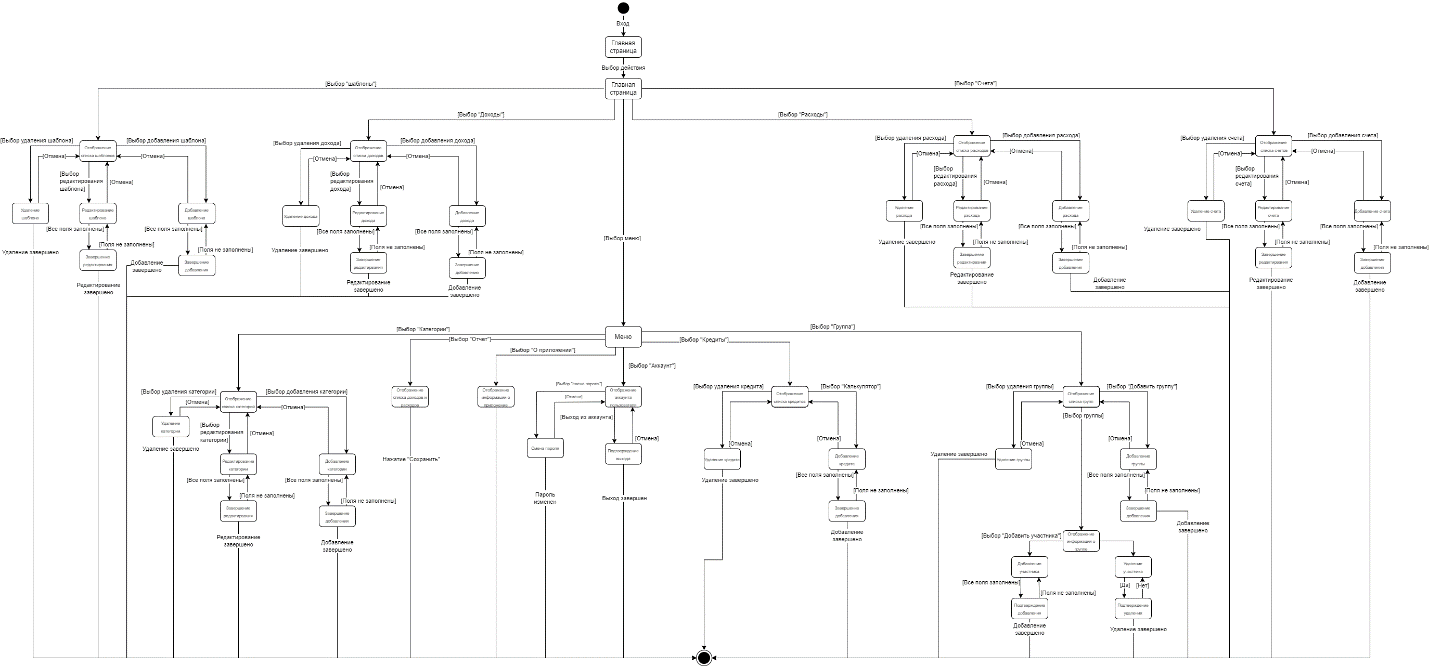
1. Диаграмма последовательности авторизированного пользователя



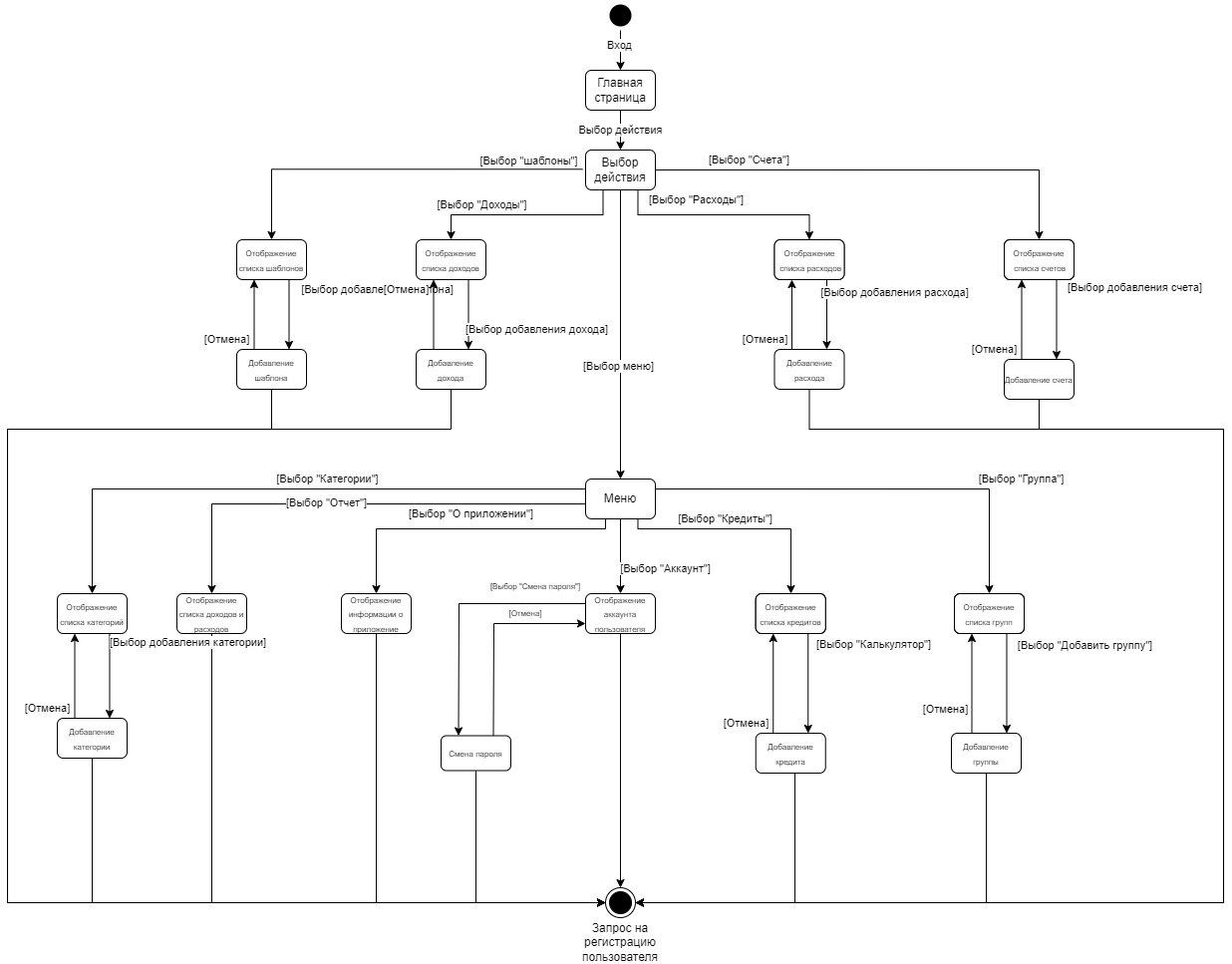
1. Диаграмма последовательности создателя группы
   * 1. Диаграмма развертывания



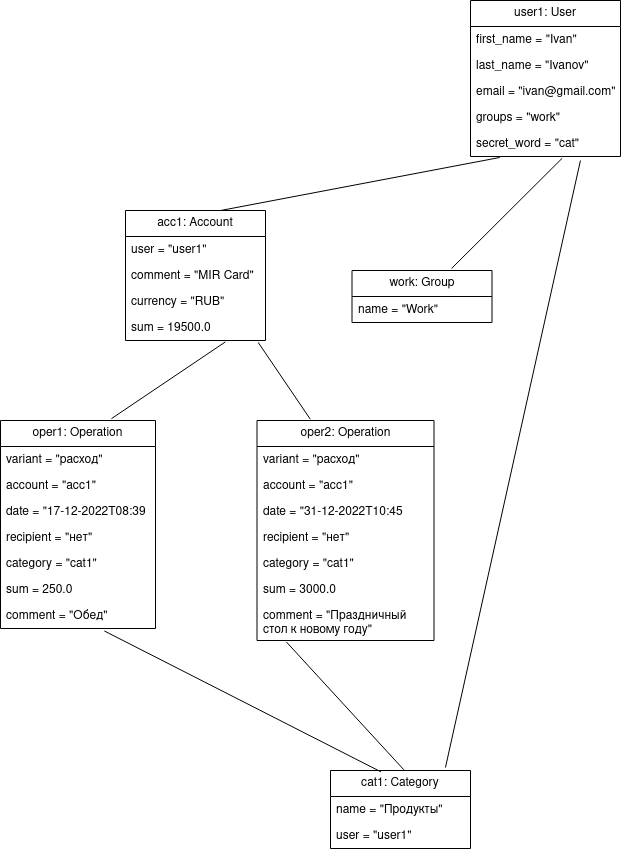
1. Диаграмма развертывания приложения
   * 1. Диаграммы состояния



1. Диаграмма состояния авторизованного пользователя



1. Диаграмма состояния неавторизованного пользователя
   * 1. Диаграмма объектов



1. Диаграмма объектов
2. Реализация
   1. Средства реализации

Ниже приведен перечень используемых технологий, который в ходе разработки может расширяться.

Backend

* Python;
* Django;
* PostgreSQL;
* Docker.

Frontend:

* Android SDK;
* Kotlin.

Инструменты для ведения документации:

* Miro;
* Swagger;
* Draw.io;
* Ramus;
* Figma.

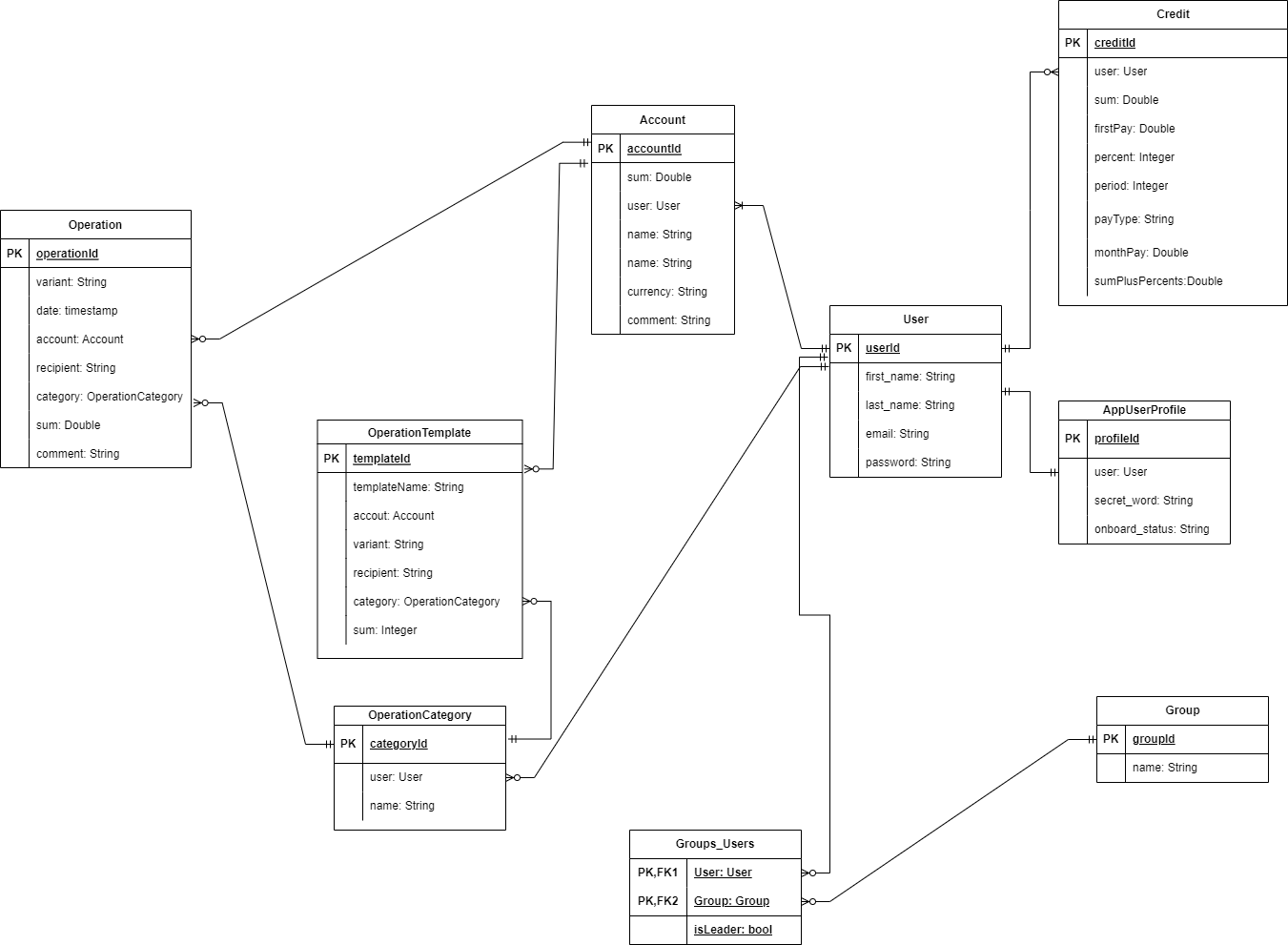
Дополнительный инструментарий:

* Git;
* GitHub;
* Trello.
  1. Реализация базы данных

Для хранения данных была выбрана база данных PostgreSQL. Она является продуктом с открытым исходным кодом, который поддерживается многими серверами. Присутствует поддержка различных типов данных. PostgreSQL поддерживает множественные типы данных, такие как числа разной точности, тексты с различными кодировками, изображения, звуки, видео, XML-документы, JSON-объекты и многие другие.

В приложении присутствуют следующие сущности:

* AppUserProfile – профиль пользователя приложения, который хранит в себе зашифрованное секретное слово, по которому происходит восстановление пароля, и статус показа для него onboard экранов;
* User – сущность пользователя, которая используется в качестве внешнего ключа в других сущностях и в системе аутентификации/авторизации в приложении;
* Group – сущность группы пользователей;
* Account – счет с денежными средствами пользователя;
* Operation – финансовая операция, совершенная пользователем (доход или расход);
* OperationCategory – категория финансовых операций;
* OperationTemplate – шаблон для создания финансовых операций с заранее установленными данными;
* CreditPay – рассчитанный кредитный платеж пользователя;
* OnboardScreen – показываемый пользователю экран с информацией во время первого входа в приложение или при обновлении приложения.
  + 1. ER-диаграмма



1. ER-диаграмма базы данных
   1. Реализация клиентской части

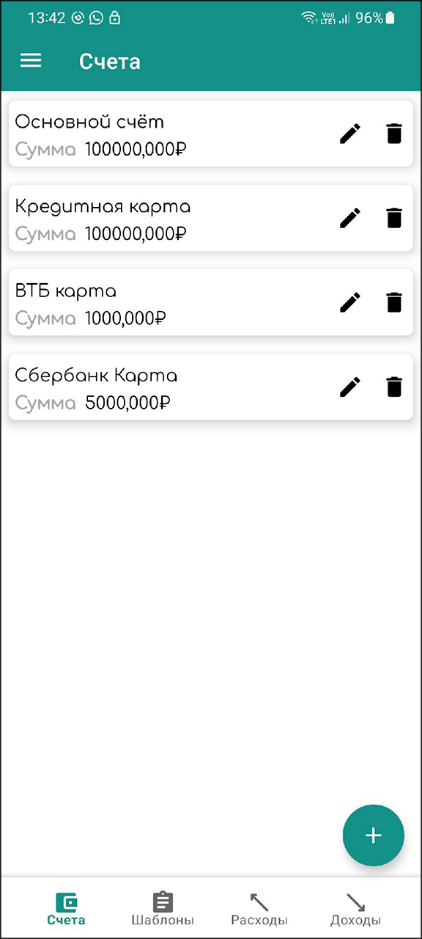
Мобильное приложение построено на архитектурном паттерне MVVM (Model-View-ViewModel), т.к. он позволяет разделить ответственность между компонентами, улучшить тестирование, управлять жизненным циклом компонентов, обеспечить масштабируемость и повысить эффективность разработки мобильного приложения.

Приложение разбито по пакетам:

* api – хранит в себе еще два пакета: service и model. Service отвечает за описание интерфейсов для работы REST API, которая включает в себя методы для взаимодействия с сервером, а также включает вспомогательный объект для создания экземпляра класса. В пакете model хранятся структуры данных, которые отвечают за сериализацию и десериализацию данных из HTTP-запросов и ответов сервера;
* repository - используются для абстрагирования доступа к сетевым ресурсам, которые могут быть использованы для получения, отправки, обновления и удаления данных на сервере;
* view - отвечает за отображение данных на экране устройства и взаимодействие с пользователем;
* viewmodel - отвечает за управление данными и бизнес-логикой приложения.
  + 1. Экраны счетов

На экране «Счета» авторизованный пользователь имеет возможность просматривать уже добавленные счета, редактировать и удалять их, а также создать новый при нажатии на кнопку.

При первой регистрации пользователя автоматически создается счет «Первоначальный счет».



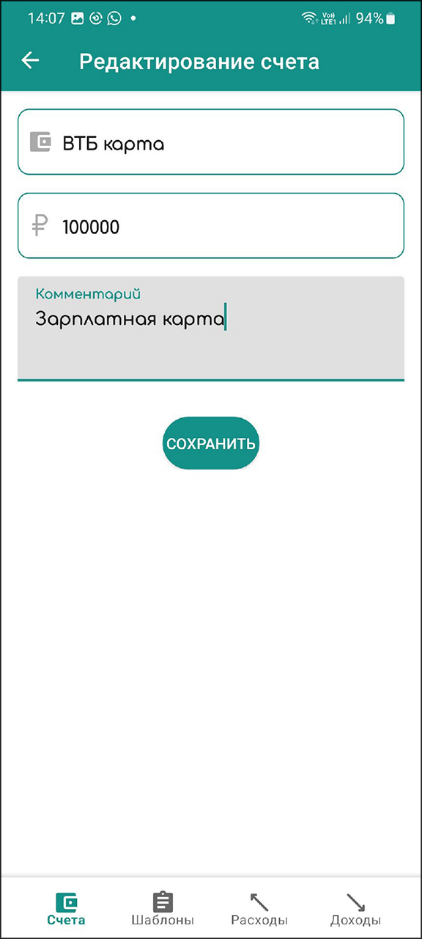
1. Экран счетов

На экране «Счет», необходимого для создания нового счета, отображены поля с название счета, изначальной суммой и комментарием, ниже расположена кнопка для сохранения.



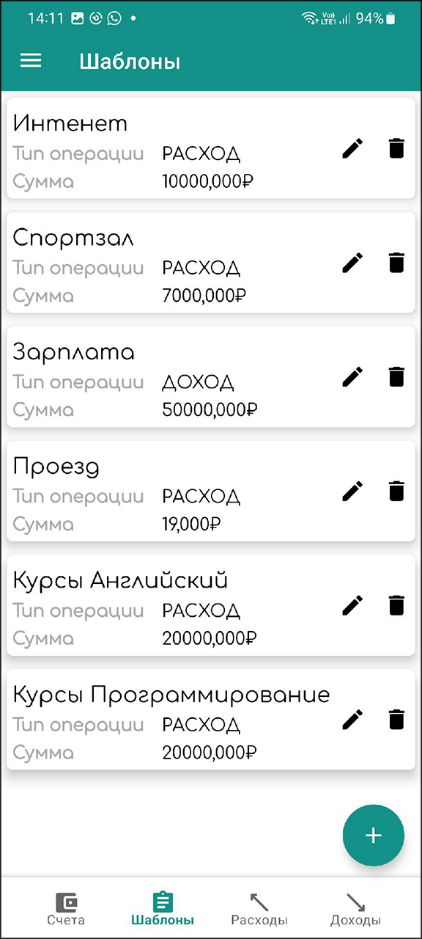
1. Экран добавления счета

На экране «Редактирование счета» отображены поля с название счета, изначальной суммой и комментарием, ниже расположена кнопка для сохранения. Данный экран отображается после нажатия кнопки с изображением ручки на экране «Счета».



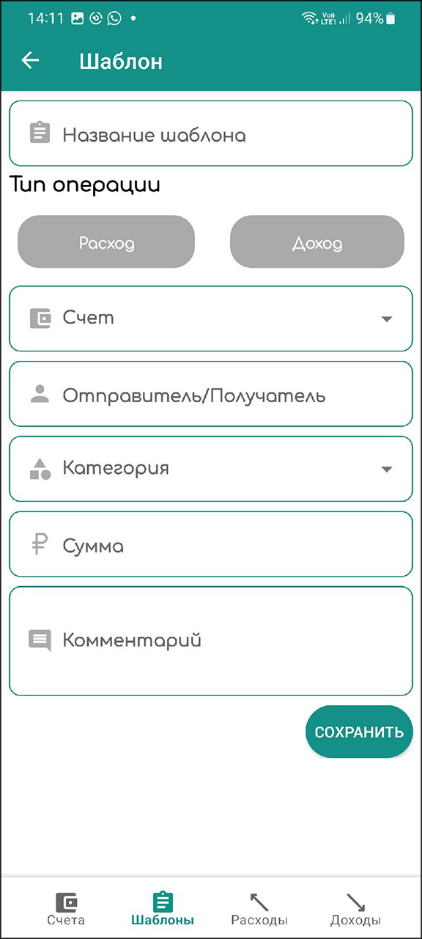
1. Экран редактирования счета
   * 1. Экраны шаблонов

На экране «Шаблоны» авторизованный пользователь имеет возможность просматривать уже добавленные шаблоны, редактировать и удалять их, а также создать новый при нажатии на кнопку.



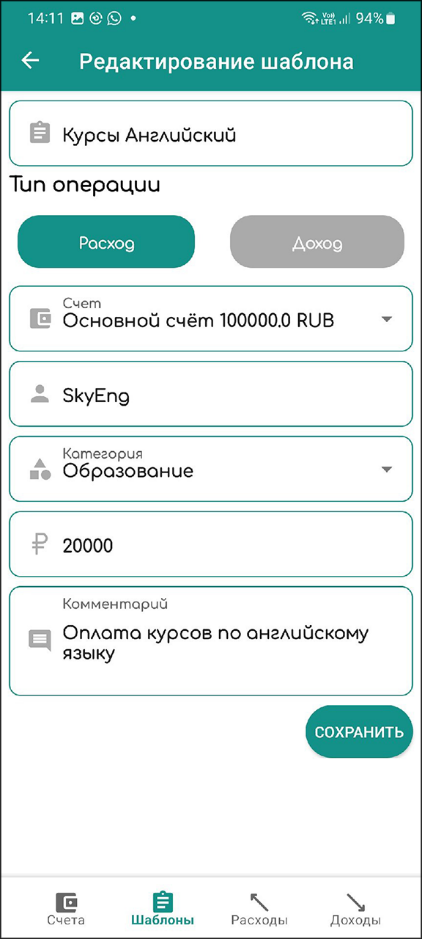
1. Экран шаблонов

Экране «Шаблон» необходим, для добавления нового шаблона. На нем отображены поля с названием шаблона, выбором счета, типом операции, отправителем или получателем, выбором категории, суммы и комментария. Ниже расположена кнопка для сохранения.



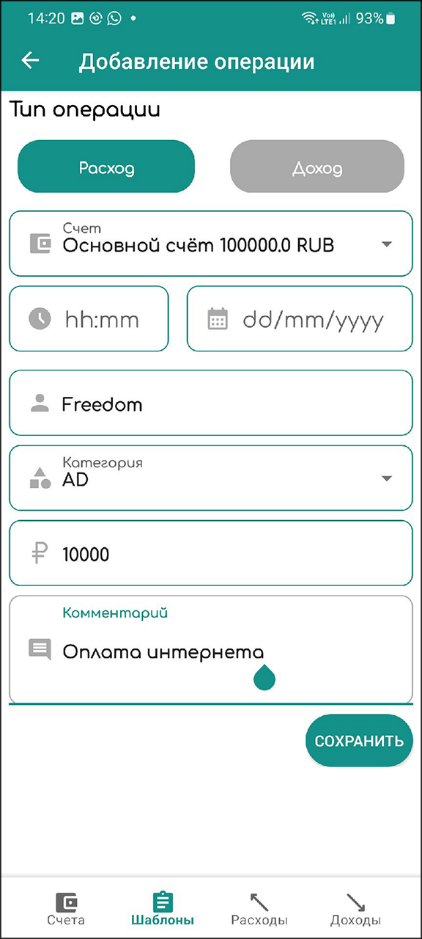
1. Экран создания шаблона

На экране «Редактирование шаблона» отображены поля с названием шаблона, выбором счета, типом операции, отправителем или получателем, выбором категории, суммы и комментария. Ниже расположена кнопка для сохранения. Данный экран отображается после нажатия кнопки с изображением ручки на экране «Шаблоны».



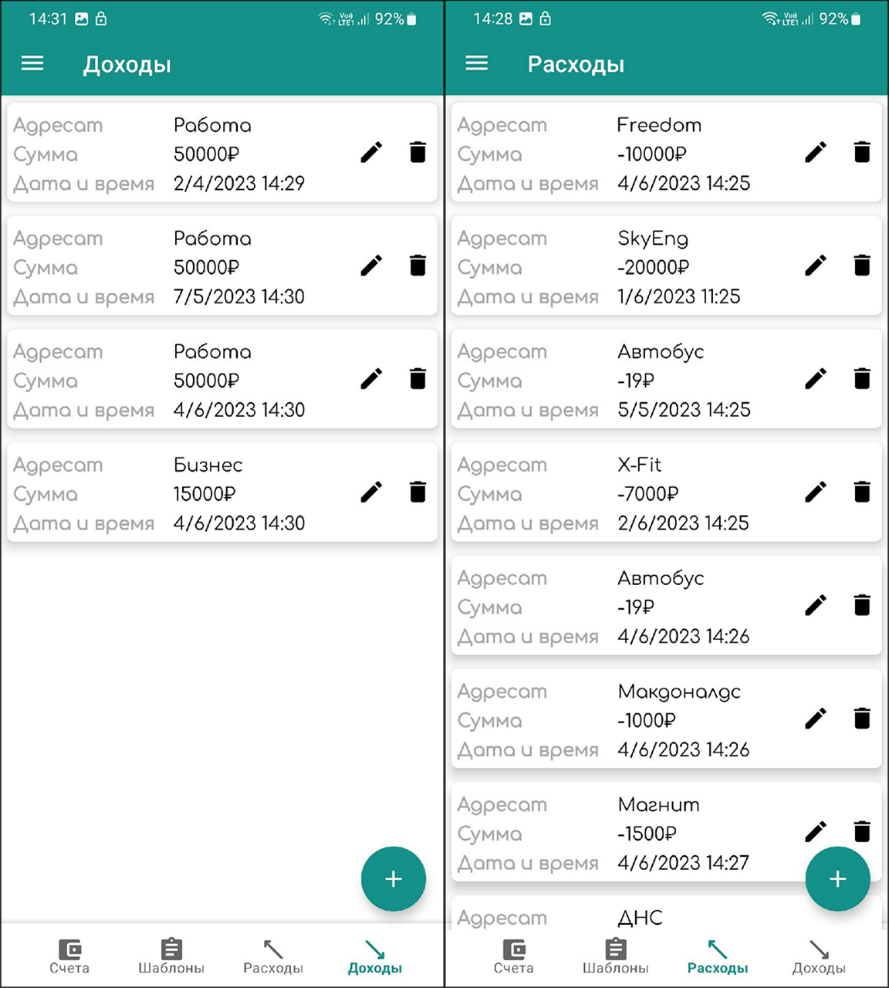
1. Экран редактирования шаблона

При нажатии на шаблон, отображается экран «Добавление операции», в котором следующие поля предзаполнены: тип операции, счет, отправитель или получатель, категория, сумма, комментарий.



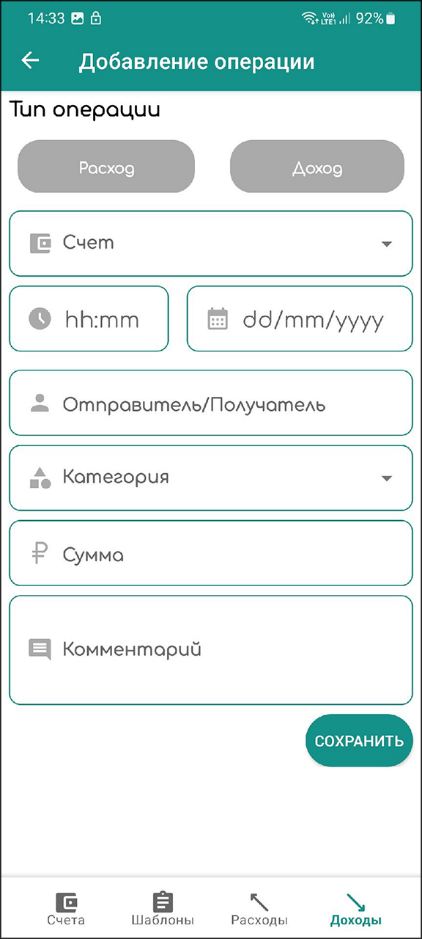
1. Экран добавления операции при нажатии на шаблон
   * 1. Экраны финансовых операций

На экране «Доходы» и «Расходы» авторизованный пользователь имеет возможность просматривать уже добавленные финансовые операции, редактировать, удалять их, а также создать новый при нажатии на кнопку.



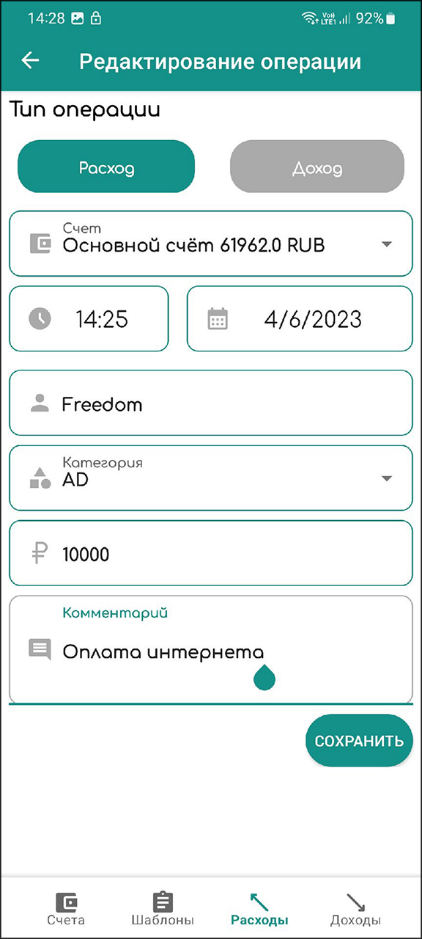
1. Экраны доходов и расходов

На экране «Добавление операции» отображены поля с типом операции, выбором счета, времени и даты, отправителем или получателем, выбором категории, суммы и комментария. Ниже расположена кнопка для сохранения.



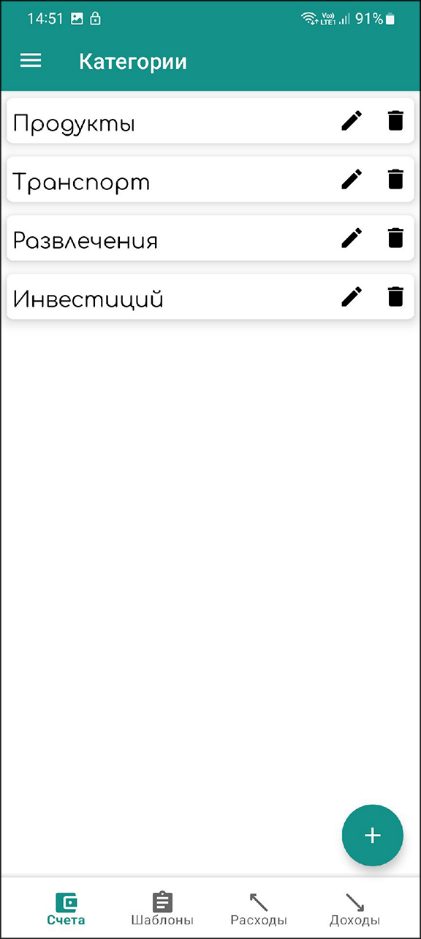
1. Экран добавления операции

На экране «Редактирование операции» отображены поля с типом операции, выбором счета, времени и даты, отправителем или получателем, выбором категории, суммы и комментария. Ниже расположена кнопка для сохранения.



1. Экран редактирование операции
   * 1. Экран категорий

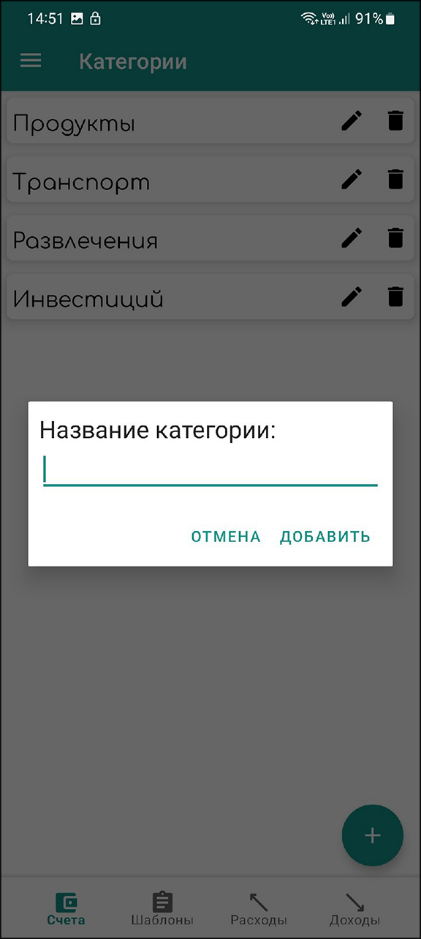
На экране «Категории» авторизованный пользователь имеет возможность просматривать уже добавленные категории, редактировать и удалять их, а также создать новый при нажатии на кнопку. При первой регистрации пользователя автоматический создается три категории: «Продукты», «Транспорт» и «Развлечения».



1. Экран категорий

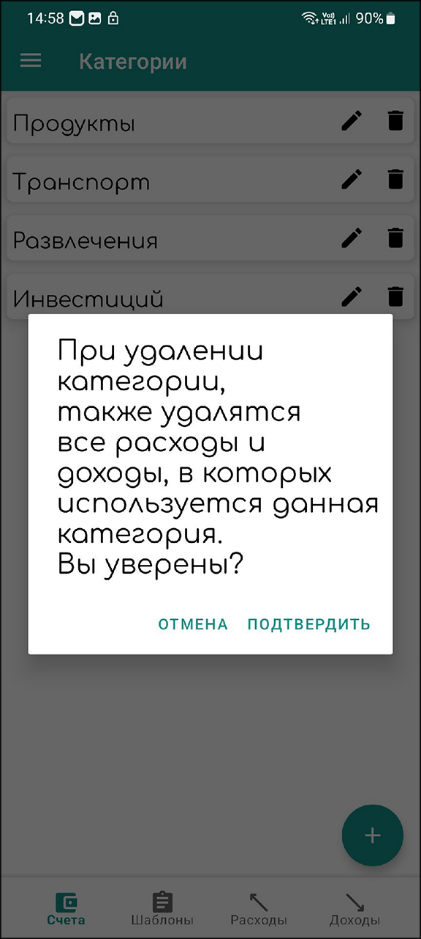
При попытке добавления новой категории открывается диалоговое окно с полем для названия категории и двумя кнопками: «Добавить» и «Отмена».

При нажатии на кнопку с изображением карандаша, откроется диалоговое окно, схожее с диалоговым окном, открывающимся при добавлении новой категории.



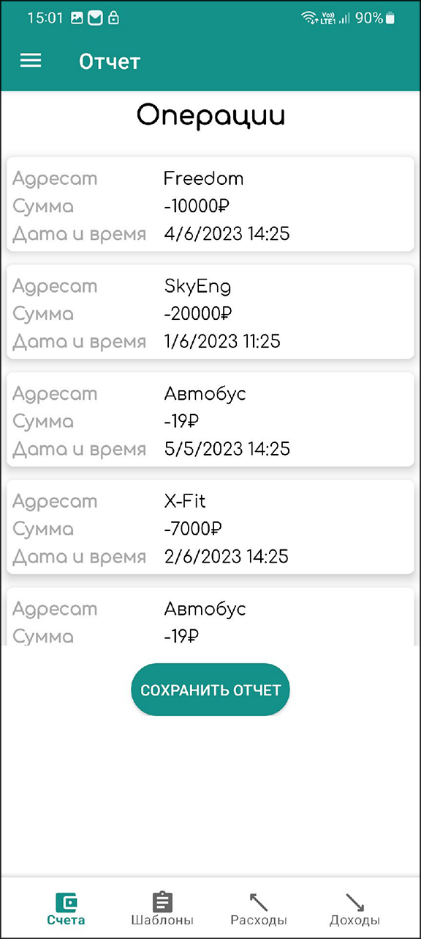
1. Экран создания и редактирования категории

При попытке удаления категории, путем нажатия на кнопку с изображением мусорной корзины, появится диалоговое окно с подтверждением удаления категории.



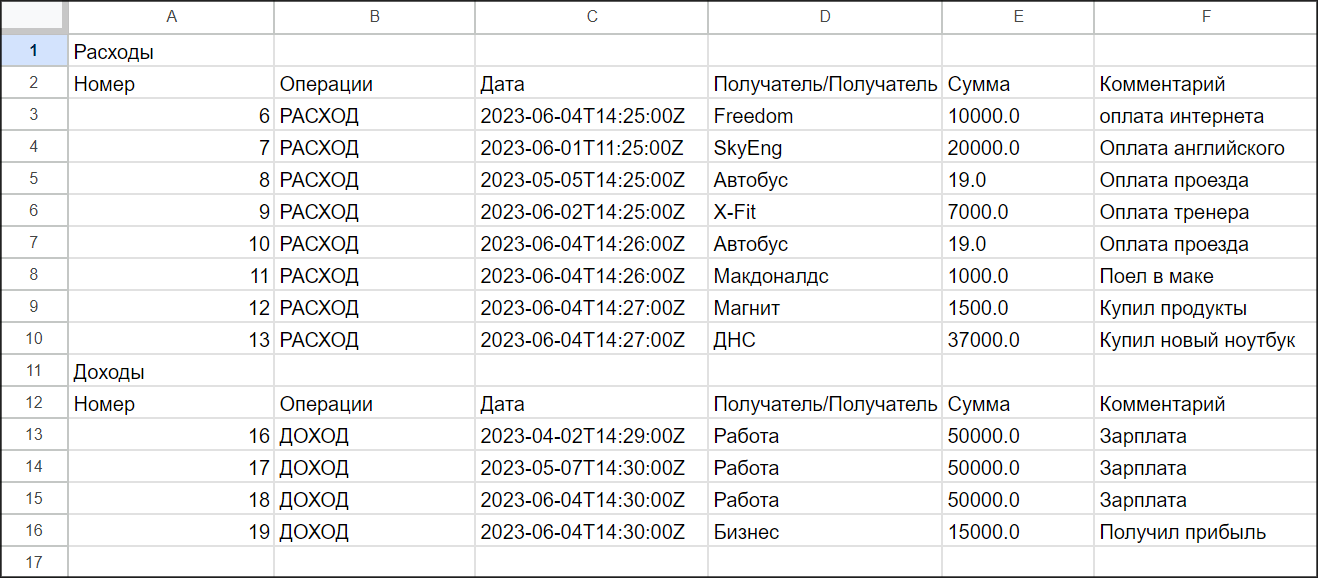
1. Диалоговое окно удаления категории
   * 1. Экран отчета

На данном экране отображены все финансовые операции, добавленные авторизованным пользователем. Под списком операций имеется кнопка для сохранения отчета, при нажатии на нее составляется таблица в формате «CSV».



1. Экран отчета

Таблица сохраняется в локальное хранилище устройства в директории «Документы».

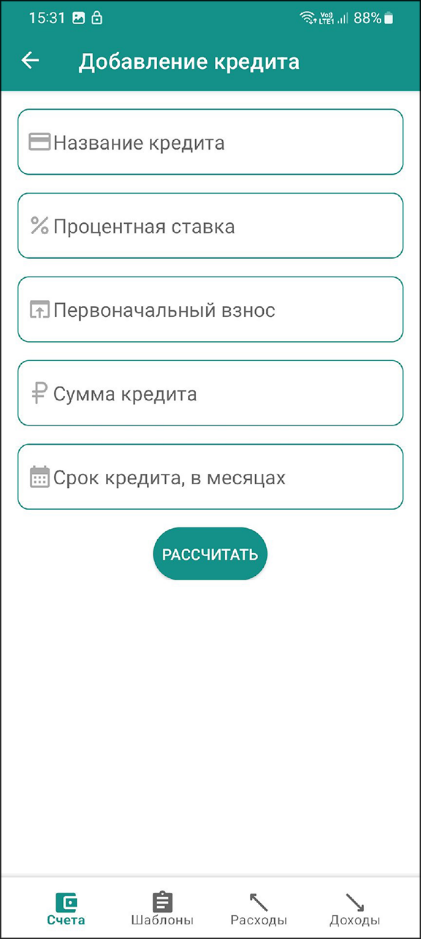


1. Результат сохранения отчета
   * 1. Экраны кредитов

На экране «Ваши кредиты» авторизованный пользователь имеет возможность просмотреть ранее добавленные кредиты, а также добавить новый при нажатии на кнопку, расположенной в правом нижнем углу экрана.

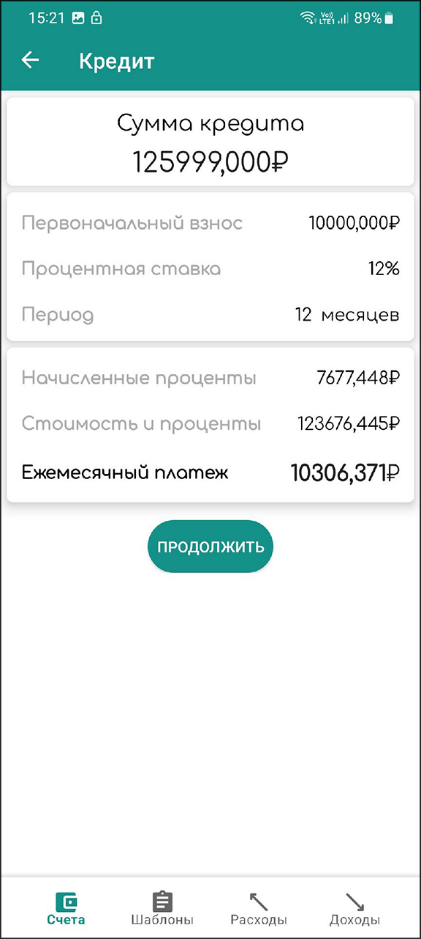


1. Экран кредитов



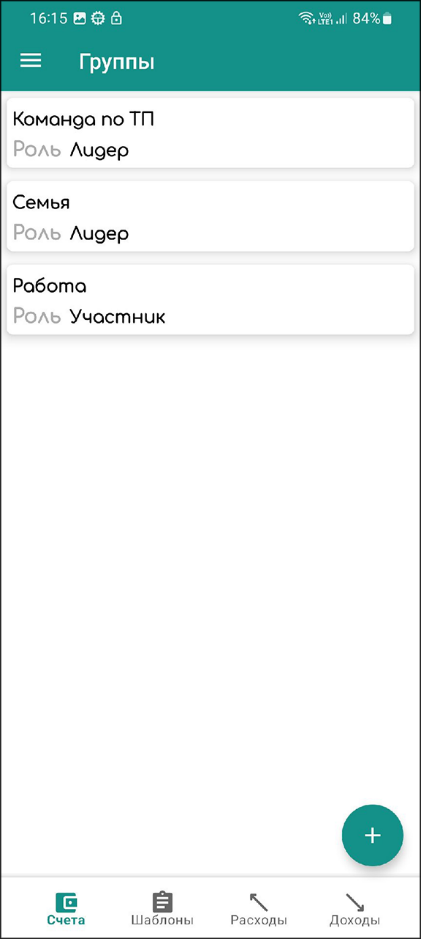
1. Экран добавления кредита

После корректного ввода всех полей и нажатие на кнопку «Рассчитать», а также при нажатии на кредит из экрана «Ваши кредиты», отобразится экран «Кредит», в котором показана информация о добавленном кредите: сумма кредита, первоначальный взнос, процентная ставка, период, начисленные проценты, стоимость и проценты и ежемесячный платеж. Также под информацией о кредите расположена кнопка «Продолжить», при нажатии на которую, отобразится экран «Ваши кредиты».



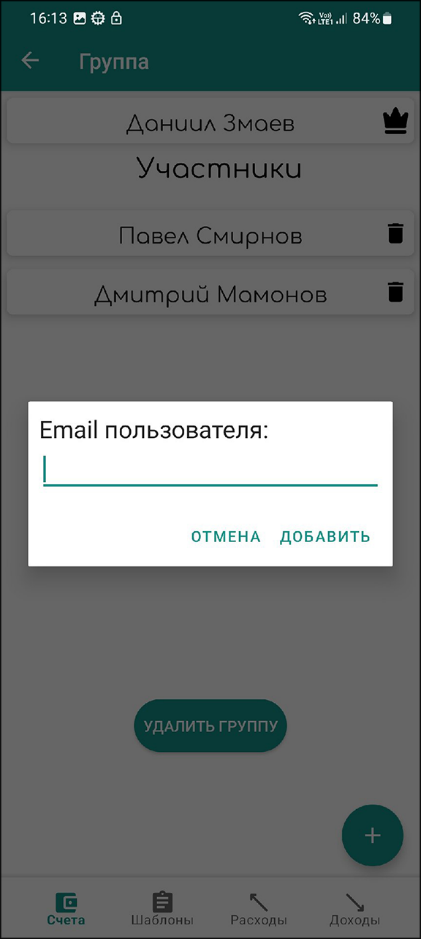
1. Экран просмотра информации о кредите
   * 1. Экраны группы

На данном экране расположены список групп, в которых состоит пользователь, а также кнопка, при нажатии на которую, всплывает диалоговое окно.



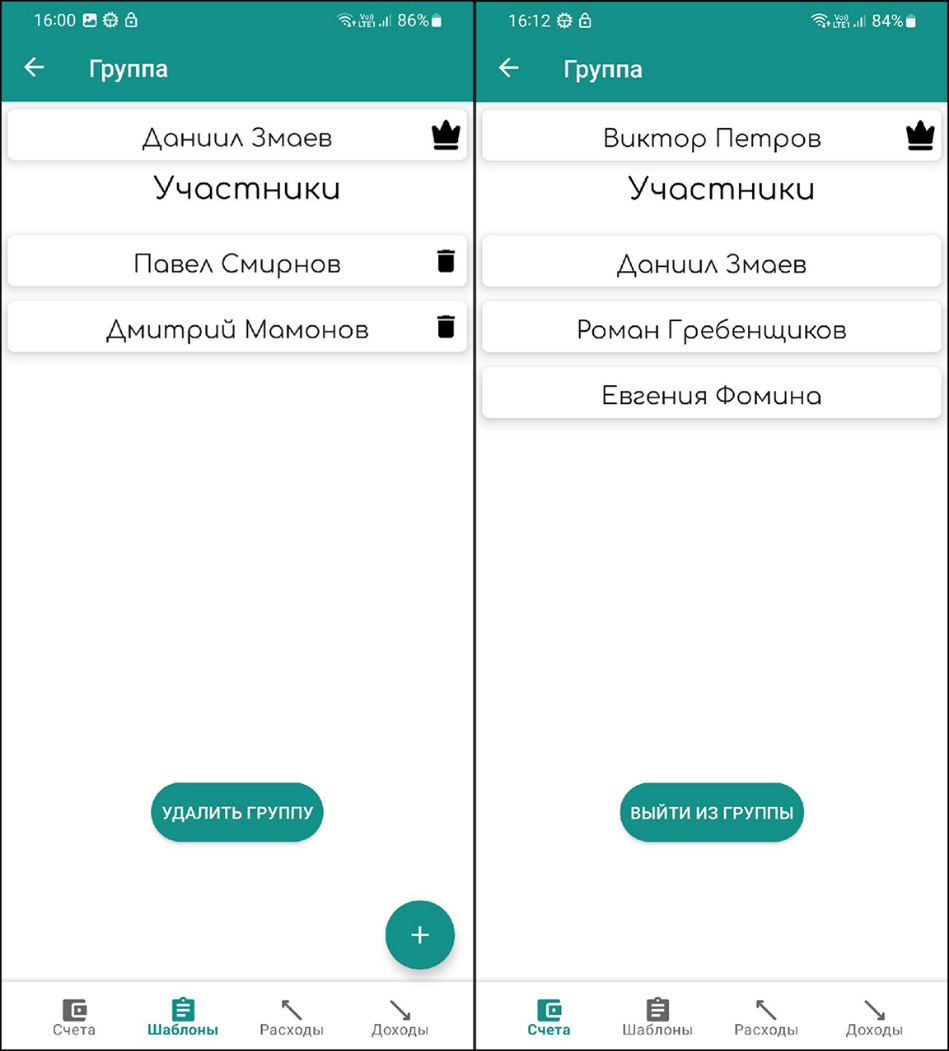
1. Экран просмотра списка групп

Диалоговое окно содержит поле для ввода почты пользователя, а также две кнопки «Отмена» и «Добавить».



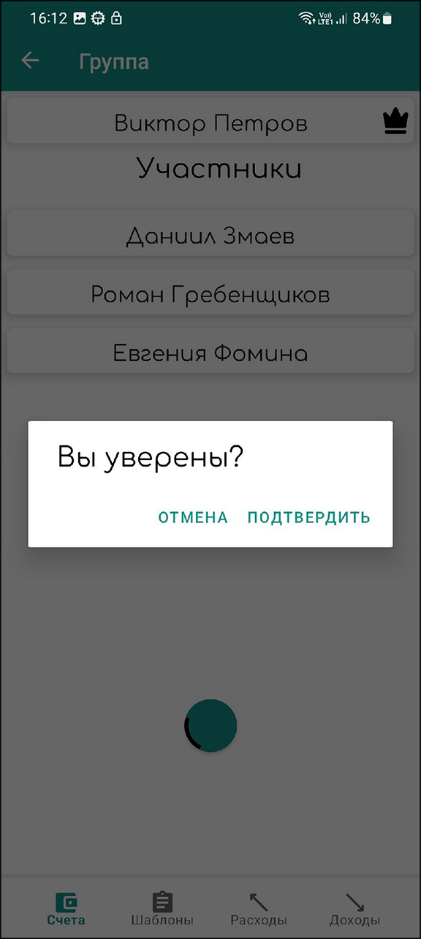
1. Диалоговое окно добавления пользователя

При нажатии на группу, открывается экран «Группа», в котором отображены все участники, состоящие в группе.



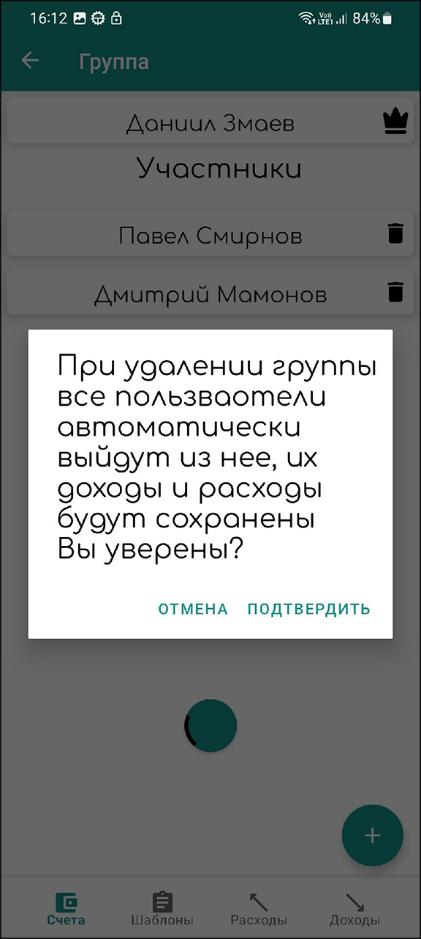
1. Экран группы для создателя и участника

При нажатии на кнопку с изображением мусорной корзины, для удаления пользователя из группы, или при нажатии на кнопку «Выйти из группы» всплывает диалоговое окно.



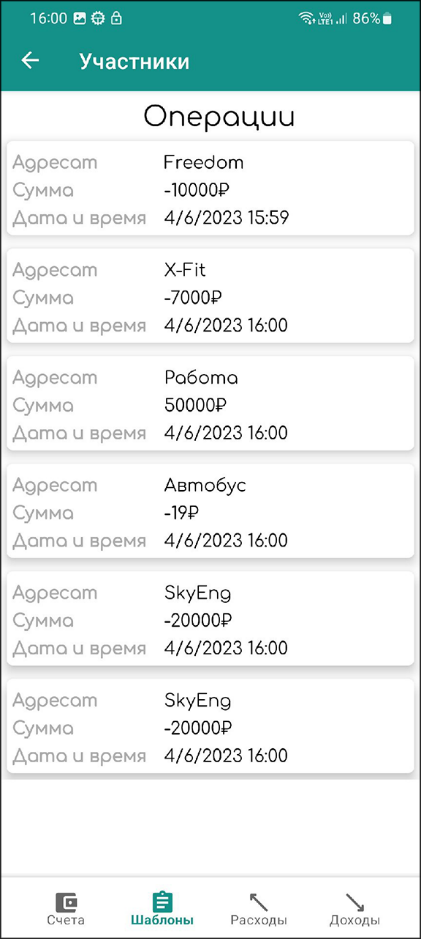
1. Диалоговое окно выхода или исключения пользователя из группы

При нажатии на кнопку «Удалить группы» создателем группы, отображается диалоговое окно с подтверждением действия.



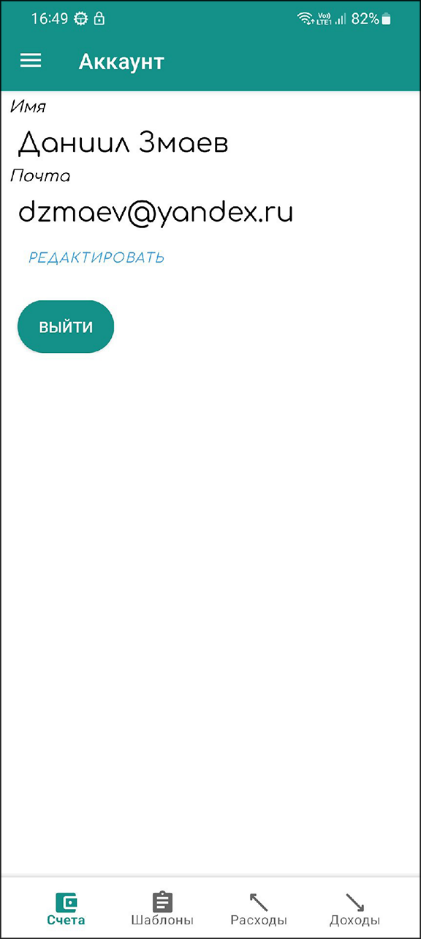
1. Диалоговое окно удаления группы

При нажатии на участника группы отображается экран финансовых операций выбранного участника.



1. Экран операций участника группы
   * 1. Экран аккаунта пользователя

На данном экране расположена информация о пользователе: имя и фамилия и почта. Также имеются две кнопку: «Редактировать» и «Выйти»



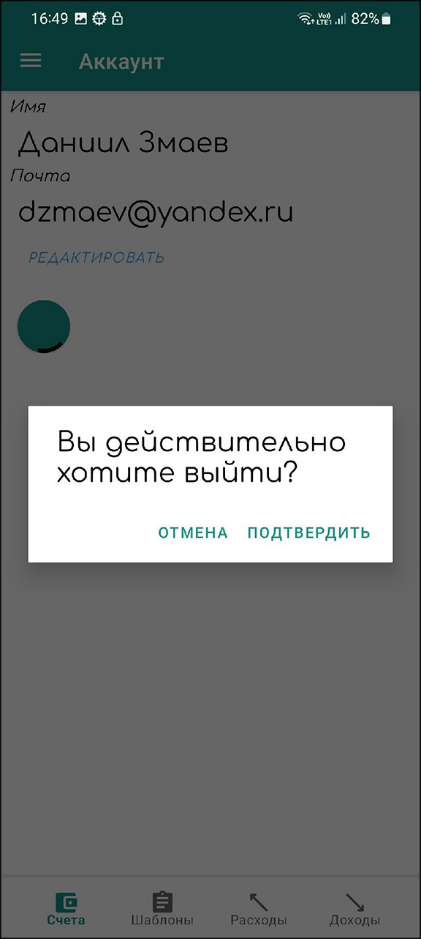
1. Экран аккаунта

При нажатии на кнопку «Редактировать», отображается экран редактирования аккаунта пользователя, в котором расположены следующие поля: имя, фамилия. Под полем фамилия имеется кнопка «Подтвердить», необходимая для сохранения отредактированных данных.



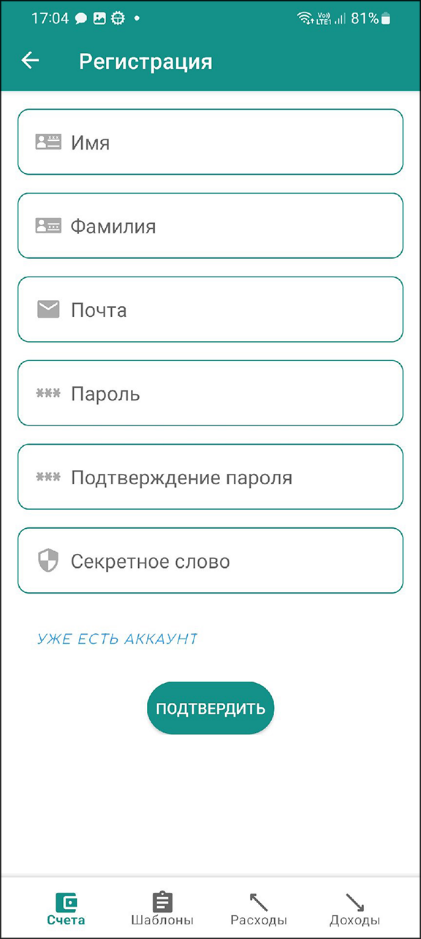
1. Экран редактирования аккаунта

При нажатии на кнопку «Выйти», появляется диалоговое окно, необходимое для подтверждения действия.



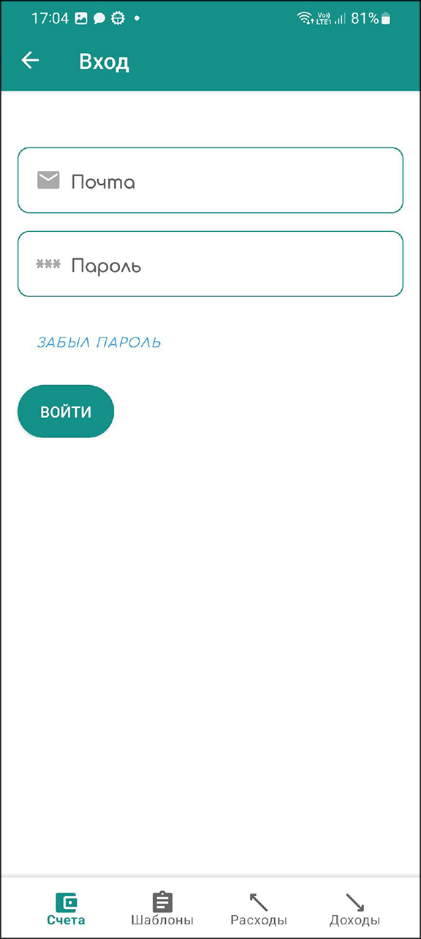
1. Диалоговое окно выхода из аккаунта
   * 1. Экран регистрации

На данной странице отображается форма для регистрации пользователя: имя, фамилия, почта, пароль, подтверждение пароля и секретное слово. Также имеются две кнопки, «Уже есть аккаунт» и «Подтвердить». Данный экран отображается при попытке пользователя добавить счет, финансовую операцию, группу.



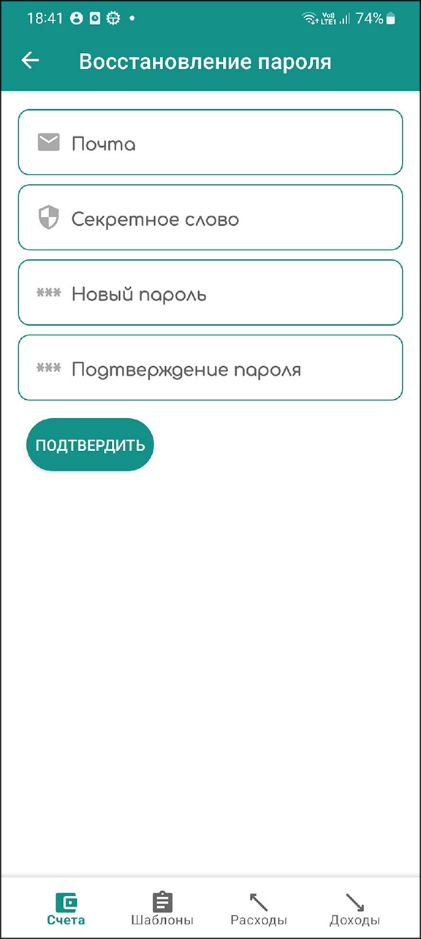
1. Экран регистрации
   * 1. Экран входа

На данном экране расположены поля для выполнения входа в аккаунт: почта и пароль, также имеются две кнопки «Забыл пароль» и «Войти». Данный экран отображается при нажатии на кнопку «Уже есть аккаунт» на экране «Регистрация».



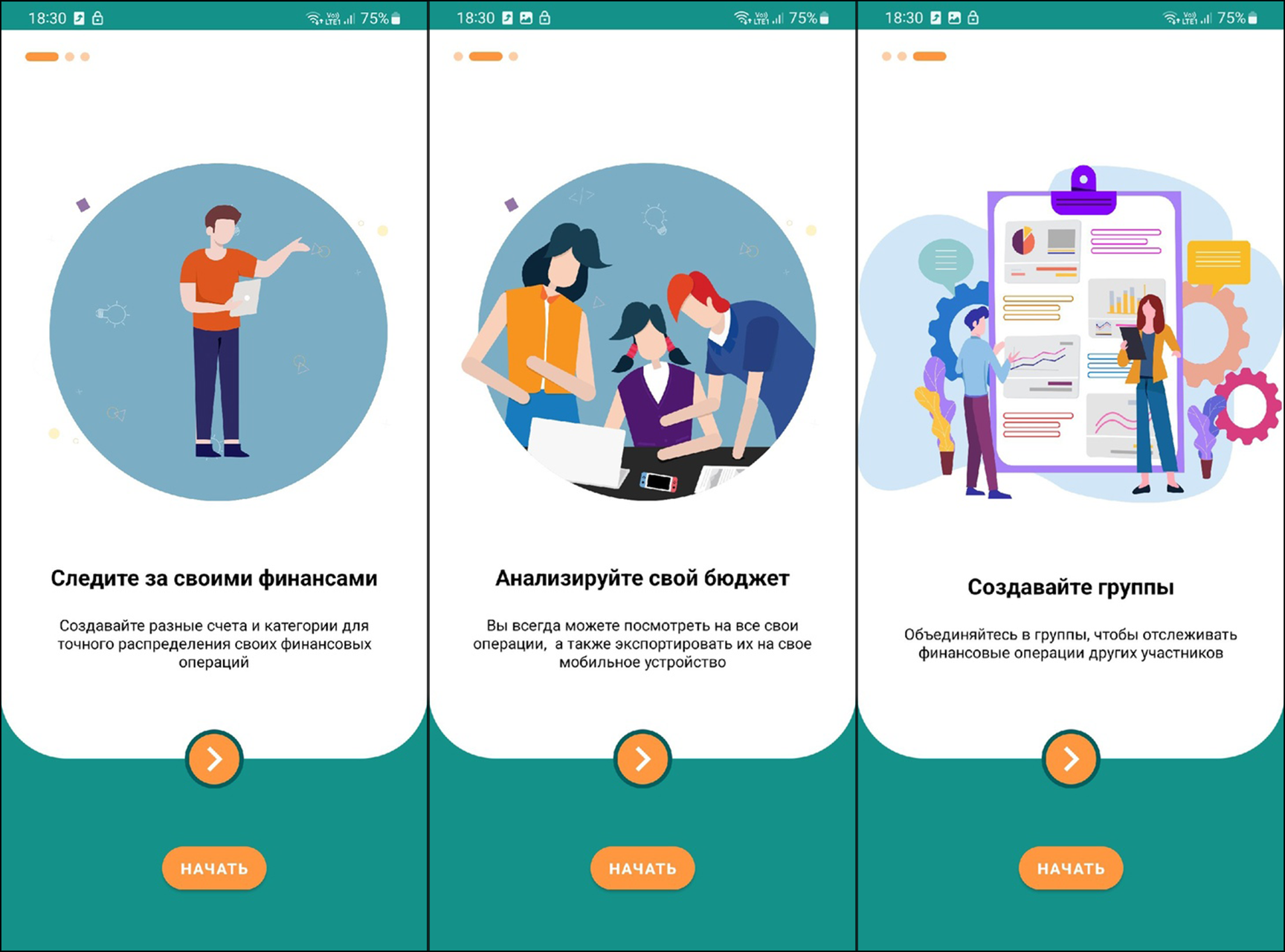
1. Экран входа
   * 1. Экран восстановления пароля

На данном экране расположены поля для выполнения процедуры восстановления пароля: почта и секретное слово, новый пароль и его подтверждение, также имеется и «Подтвердить». Данный экран отображается при нажатии на кнопку «Забыл пароль» на экране «Вход».



1. Экрана восстановления пароля
   * 1. Onboarding screen

При первом запуске приложения отображаются экраны с основной информацией о приложении. На данном экране имеется кнопка в виде стрелки, указывающей влево, при нажатии на нее отображается следующий экран с информацией. При нажатии на кнопку «Начать» отображается главный экран приложения.



1. Onboarding screens
   1. Серверная часть

Серверное приложение построено на шаблоне проектирования MVC (Model-View-Controller), так как он является одним из самых распространенных для построения приложений шаблонов и предоставляет разделение ответственностей в приложении, отделение бизнес-логики от реализации интерфейса и масштабируемость в будущем. Данный шаблон веб-фреймворк Django предлагает по умолчанию. В качестве архитектурного стиля создания API был выбран REST.

Все веб-приложение Django состоит из отдельных модулей-пакетов, которые являются полноценными приложениями, построенными на шаблоне MVC. Такой пакет может представлять определенную функциональность или группу функциональностей для удобства организации кода и его повторного использования. В данном серверном приложении выделяются следующие модули:

* users — приложение, отвечающее за функциональность для пользователей;
* operations — пакет, отвечающий за функциональность, связанную с финансовыми операциями, счетами, категориями операций, шаблонами и кредитами;
* groups — приложение, отвечающее за функциональность для работы с группами пользователей;
* onboard – модуль, предоставляющий возможности работы с onboard экранами для мобильного приложения.
  + 1. Структура модуля-приложения

Каждое внутреннее приложение проекта имеет похожую файловую и классовую структуру:

* директория migrations - в ней хранятся файлы всех миграций для текущего приложения;
* admin.py - данный файл предназначен для регистрации моделей приложения во встроенной в фреймворк панели администратора с целью управления экземплярами этих моделей в базе данных;
* apps.py - здесь определена базовая конфигурация модуля, тип первичного ключа для моделей этого приложения и название самого модуля для его регистрации в главной конфигурации приложения;
* models.py - предоставляет слой моделей в MVC, в нем определяются классы моделей - сущностей в базе данных, а также может определяться логика обработки сигналов изменения модели данных;
* serializers.py - в этом файле определяются специальные классы для сериализации структур данных в JSON формат и десериализации.

Данные классы используются как для моделей данных приложения, так и для тела HTTP-запросов/ответов;

* services.py - здесь определена необходимая бизнес - логика взаимодействия с моделями данных в виде процедур или отдельных классов;
* tests.py - для каждого приложения изначально определяется свой файл, где хранятся тесты, связанные только с логикой или слоями абстракции данного приложения;
* urls.py - данный файл создает блок с маршрутами для контроллера приложения, после чего добавляет его в глобальный список маршрутов urlpatterns. Если в приложении несколько контроллеров, то удобнее всего сделать отдельную директорию urls и в ней создать файлы блоков маршрутизации для каждого из контроллеров приложения. После того, как все блоки были созданы, они все включаются в файл urls, определенный в конфигурационной директории всего проекта, туда же и подключаются маршруты из внешних библиотек;
* views.py - представляет слой контроллеров в MVC, в нем определяются классы представлений, которые в контексте Django являются контроллерами. В нашем проекте все контроллеры приложений унаследованы от класса ModelViewSet, который предоставляет для указанной в его поле queryset модели базовые действия получения из базы данных всех записей этой модели, получения одной записи по ее первичному ключу, создания, редактирования и удаления записей модели. Эти базовые действия можно переопределить непосредственно в классе представления. Также здесь можно определить собственные дополнительные действия с помощью специального декоратора action;
  + 1. Модуль — приложение users

В этом пакете слой моделей представлен двумя моделями, профилем пользователя AppUserProfile и специальной моделью GroupUser, отображающей связь many-to-many между этими сущностями. Бизнес – логика определена тремя процедурами: создание нового пользователя в системе, изменение личных данных текущего пользователя и восстановление пароля по переданному секретному слову.

Слой контроллеров состоит из одного представления (контроллера) UserProfileViewSet, для которого переопределены базовые действия создания и обновления для использования процедур из слоя сервисов.

Дополнительные функции-представления в контроллере модуля добавляют следующие возможности:

* получить информацию о зарегистрированном в текущий момент в системе пользователе (функция me, тип запроса GET);
* получить список счетов пользователя по его первичному ключу в базе данных (функция accounts, тип запроса GET);
* получить список всех финансовых операций пользователя по его первичному ключу, также если пользователь состоит в группе, то можно получить список операций, начиная с той, которую он добавил первой после вступления в группу (функция operations, тип запроса GET);
* получить список кредитов зарегистрированного в текущий момент пользователя (функция credits, тип запроса GET);
* получить список групп, в которых состоит зарегистрированный в текущий момент пользователь (функция groups, тип запроса GET);
* сменить свой пароль для входа в приложение (функция reset\_password, тип запроса POST).
  + 1. Модуль — приложение operations

Данное приложение определяет логику работы с финансами приложения. Слой моделей включает в себя следующие модели данных:

* Account - счет пользователя;
* OperationCategory - категория финансовых операция;
* OperationTemplate - шаблон создания операции;
* Operation - финансовая операция (доход или расход);
* CreditPay - рассчитанный кредитный платеж для авторизованного пользователя.

Сервисы бизнес-логики хранят в себе процедуры для работы с моделями данных приложения в базе данных, а также функции генерации отчета по всем финансовым операциям, сохранения отчета в файл формата CSV на сервере и открытия его для отправки пользователю на устройство по HTTP. Еще в этом слое стоит отметить процедуру расчета кредитного платежа по определенной формуле.

Файл представлений включает в себя контроллеры для каждой модели из этого приложения. Каждый контроллер имеет встроенные функции-представления для получения всех записей модели из базы данных, получения записи по ее первичному ключу, редактирование и удаление записи по ее первичному ключу. При этом они реализуют дополнительные возможности:

* получить операции зарегистрированного пользователя по первичному ключу категории операций (контроллер OperationCategoryViewSet, функция operations, тип запроса GET);
* сгенерировать отчет по всем операциям зарегистрированного пользователя без сохранения на сервере и получить его в JSON формате (контроллер OperationViewSet, функция report, тип запроса GET);
* сохранить сгенерированный отчет на сервере в формат CSV (контроллер OperationViewSet, функция save\_report, тип запроса POST);
* отправить сохраненный файл с отчетом по HTTP (контроллер OperationViewSet, функция send\_report, тип запроса GET);
* получить операции зарегистрированного пользователя по первичному ключу счета (контроллер AccountViewSet, функция operations, тип запроса GET);
* рассчитать кредит для неавторизованного пользователя без сохранения результата в базе данных (контроллер CreditPayViewSet, функция calc\_credit, тип запроса GET).
  + 1. Модуль — приложение groups

В приложении groups определена одна модель групп пользователя Group, соответствующая бизнес-логика описана следующими процедурами:

* создать группу;
* исключить пользователя из группы;
* выбрать все финансовые операции пользователя;
* добавить нового участника;
* получить список участников группы;
* получить экземпляр лидера группы;
* расформировать группу.

Слой контроллеров представляет собой один контроллер, определенный для модели Group, с переопределенными действиями создания и удаления группы, а также дополнительными действиями для управления составом группы:

* получить список пользователей группы по ее первичному ключу в базе данных (функция users, тип запроса GET);
* проверить, является ли зарегистрированный пользователь лидером группы с определенным первичным ключом в базе данных (функция is\_leader, тип запроса GET);
* добавить нового пользователя в группу по ее определенному первичному ключу, при этом в теле запроса указывается логин-email пользователя в JSON формате (функция add\_user, тип запроса POST);
* исключить пользователя из группы по ее определенному первичному ключу, при этом в теле запроса указывается id пользователя в базе данных в JSON формате (функция remove\_user, тип запроса POST);
* выйти зарегистрированному пользователю из группы по ее первичному ключу (функция exit\_from\_group, тип запроса POST).
  + 1. Модуль — приложение onboard

В приложении onboard определена одна модель экрана с данными OnboardScreen, бизнес-логика содержит одну процедуру для получения экранов по их типу в базе данных (приветственные экраны, экраны обновления приложения и т.д.).

Слой контроллеров включает в себя один контроллер, который не переопределяет базовые действия для модели и имеет дополнительное действие для выборки экранов по их типу через параметры запроса (функция get\_onboards\_by\_type, тип запроса GET). Для всех действия установлено специальное ограничение, смысл которого в том, что получать информацию об экранах с помощью запроса GET могут все типы пользователей, а создавать и редактировать их может только пользователь-администратор.

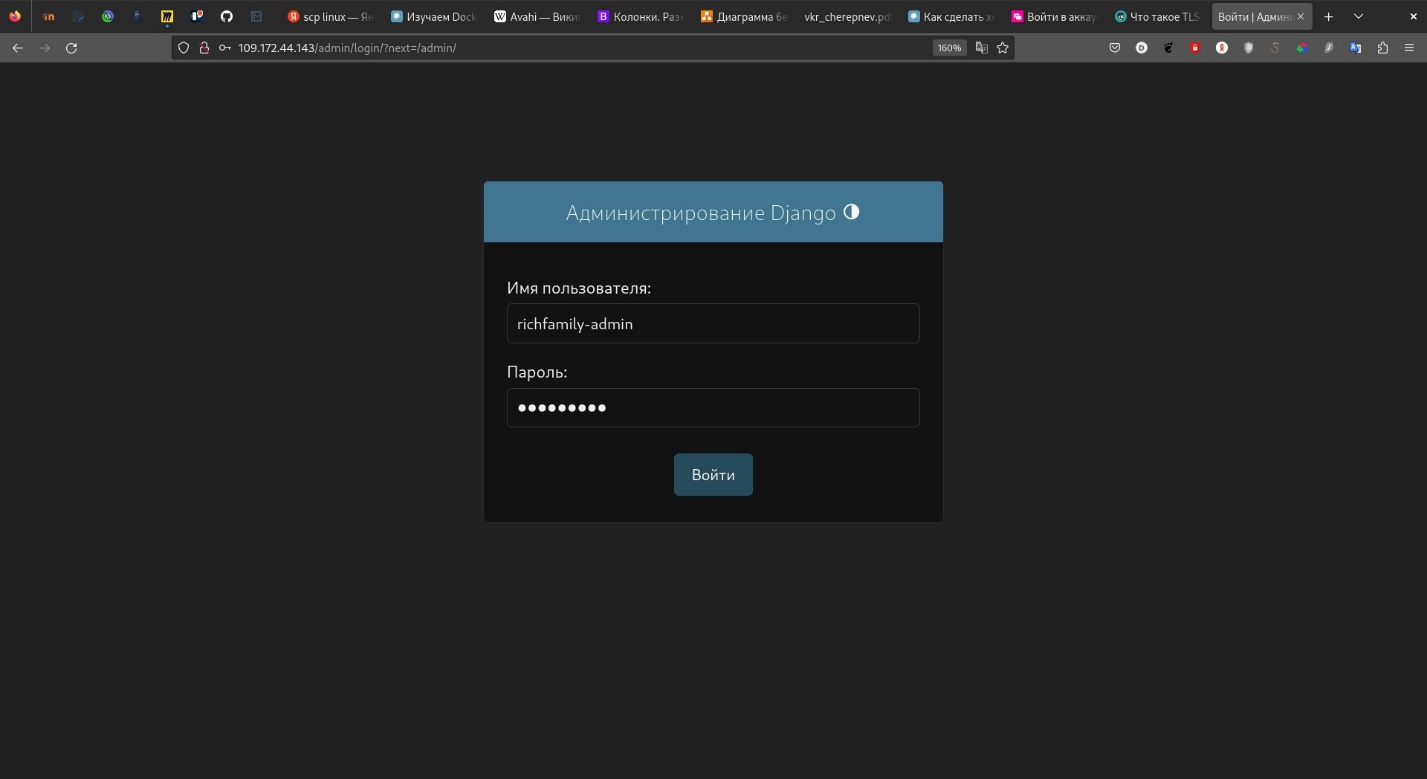
* + 1. Аутентификация и авторизация

В качестве системы аутентификации в приложении используется внутренняя реализация в веб-фрейворке и дополнительный пакет Djoser, предоставляющий интерфейс для аутентификации с помощью JWT-токенов. При запросе на вход пользователя в приложение ему в теле ответа отправляется токен, который он должен указывать в заголовках к запросам на сервер. Авторизация по токену присутствует во всех контроллерах кроме контроллера взаимодействия с кредитными платежами, так как рассчитать платеж имеет возможность и неавторизованный пользователь.

* + 1. Панель администратора

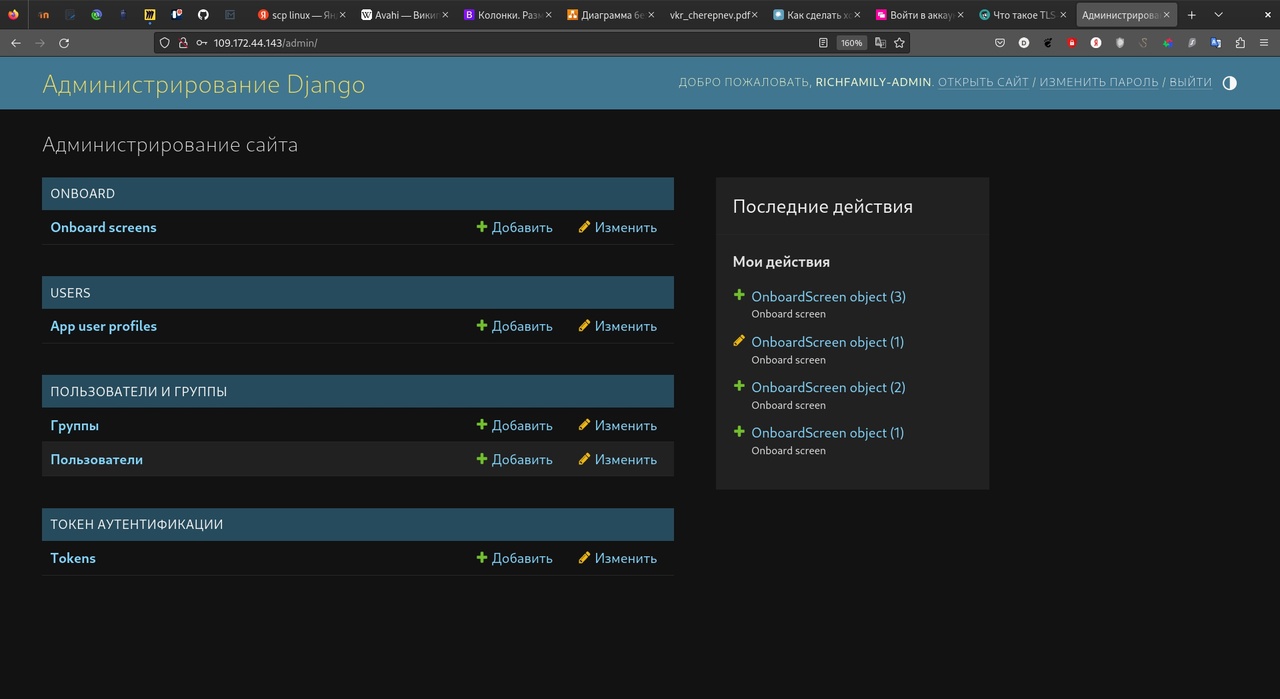
Для управления onboard экранами, а также статусом их отображения у пользователей было принято решение задействовать встроенную в веб-фреймворк Django панель администратора. Чтобы можно было ее использовать, для нее регистрируется специальный пользователь – администратор с помощью утилит фреймворка и все необходимые модели для управления регистрируются в admin-файлах своих приложений. После этого

по маршруту /admin/ можно зайти на страницу входа в панель управления.

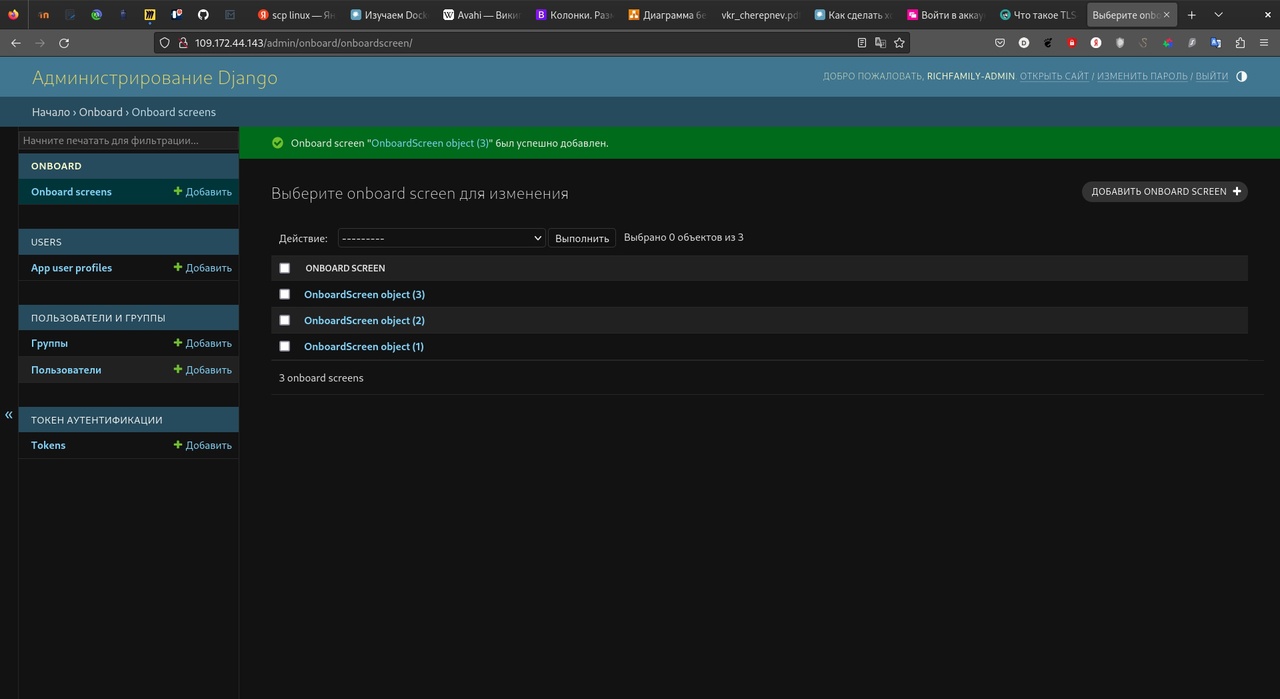


1. Вход администратора в панель

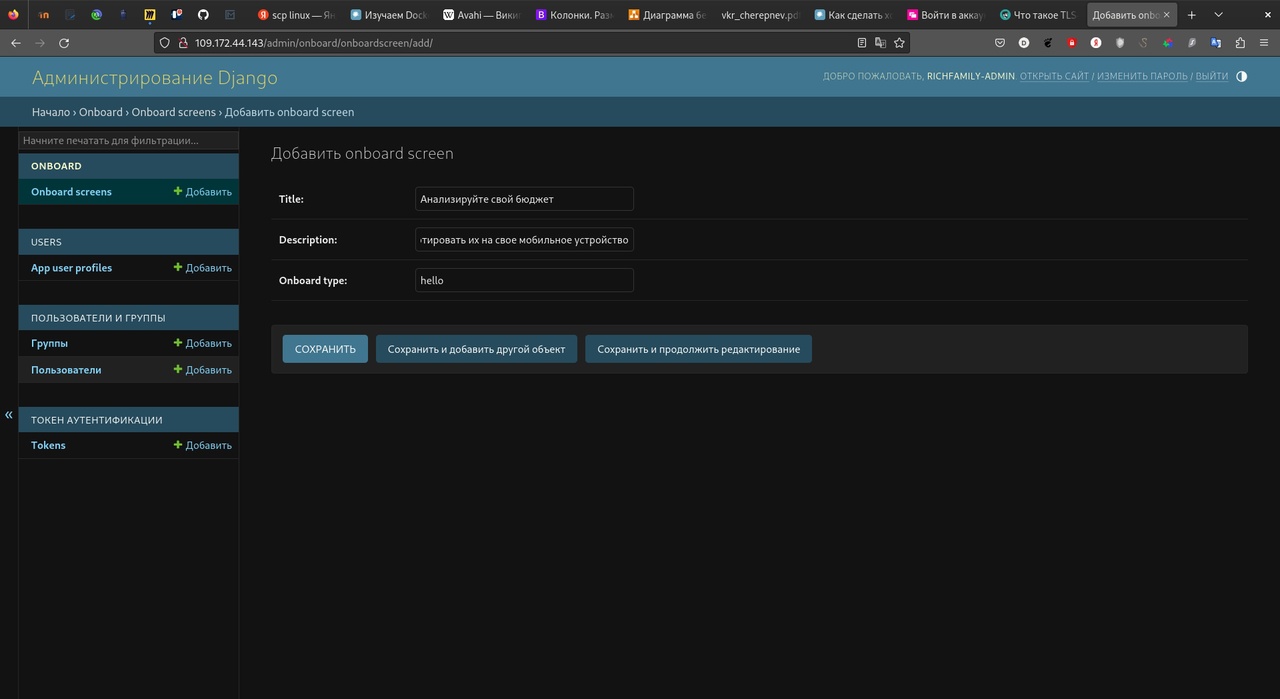
Как только пользователь произвел вход, для него открывается страница управления, где будут расположен список моделей, доступных для создания и редактирования. При нажатии на модель открывается страница со списком записей в базе данных, и можно просматривать все записи по отдельности, редактировать их и создавать новые.



1. Главная страница панели администратора

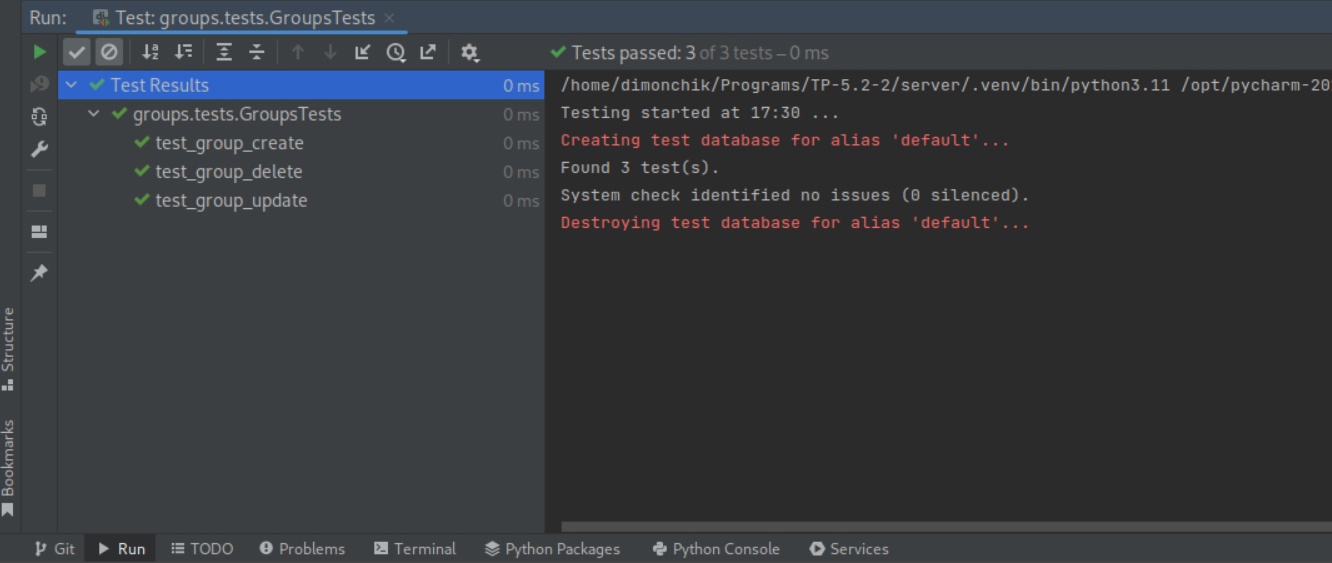


1. Просмотр записей модели OnboardingScreen



1. Страница редактирования записи модели OnboardingScreen
2. Тестирование
   * 1. Интеграционное тестирование

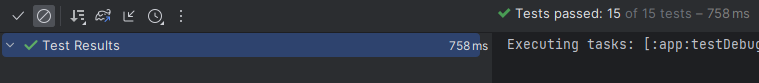
Интеграционное тестирование — это процесс проверки совместной работы разных компонентов или модулей программного обеспечения после их интеграции в одну систему.



1. Результат интеграционных тестов
   * 1. Тестирование репозиториев

Тестирование репозиториев — это процесс проверки корректности работы репозиториев в приложении при помощи написания специальных тестовых сценариев.

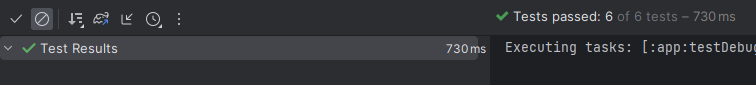
Для тестирования репозиториев использовались фреймворки, такие как JUnit и Mockito.



1. Результат тестирования репозиториев
   * 1. Тестирование ViewModel

Тестирование ViewModel — это процесс проверки корректности работы ViewModel в приложении при помощи написания специальных тестовых сценариев.

Для тестирования ViewModel использовались фреймворки, такие как JUnit и Mockito.



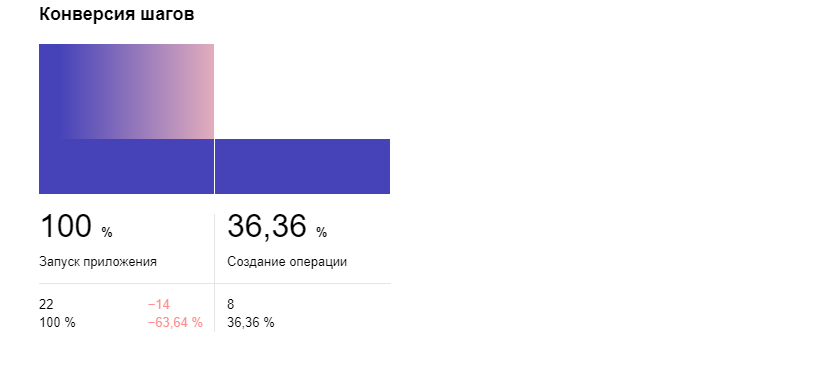
1. Результат тестирования ViewModel
2. Аналитика

Для сбора метрик приложения используется библиотека AppMetrica от Яндекса.

Были составлены четыре воронки конверсии:

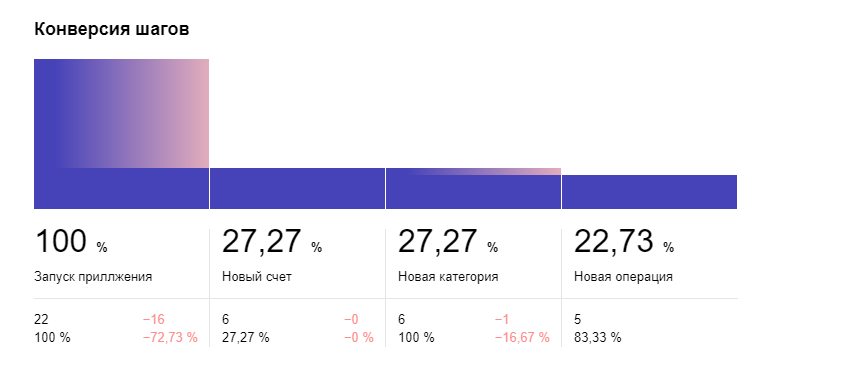
* создание операции;
* создание операции после создания счета и категории;
* группы.

Воронка конверсии «Создание операции» необходима для того, чтобы просмотреть статистику по количеству пользователей, создавших финансовую операцию.



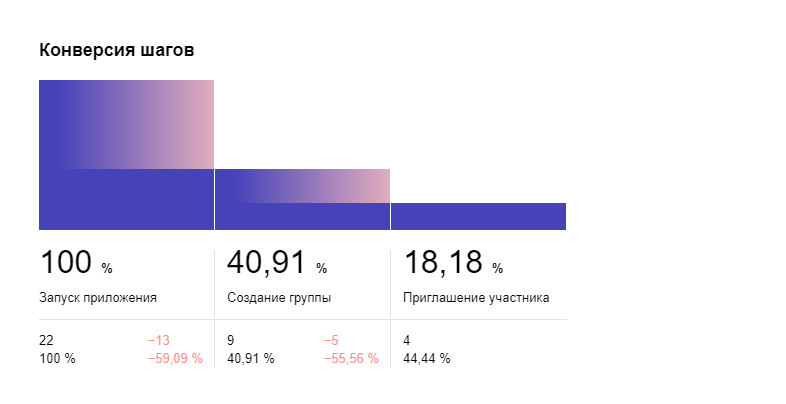
1. Воронка конверсии «Создание операции»

Воронка конверсии «Создание операции после создания счета и категории» необходима для того, чтобы просмотреть статистику по количеству пользователей, создавших финансовую операцию после создания счета и категории.



1. Воронка конверсии «Создание операции после создания счета и категории»

Воронка конверсии «Группы» необходима для того, чтобы просмотреть статистику по количеству пользователей, создавших группу, а также пригласивших в нее участников.



1. Воронка конверсии «Группы»

Заключение

В ходе выполнения курсового проекта командой было разработано мобильное приложение учета финансов с возможностью формирования отчетов и расчета кредита, соответствующее поставленным перед проектом задачам.

В начале разработки был проведен анализ предметной области, определены основные требования к разрабатываемой системе, определены основные сценарии приложения.

По результатам разработки были проведены тесты с целью выявления ошибок в работе приложения.

В процессе работы были реализованы следующие задачи проекта:

* обеспечение учета доходов и расходов;
* обеспечение группировки индивидуальных финансовых операций;
* обеспечение создания шаблонов для частых транзакций;
* обеспечение расчета кредита;
* обеспечение создания категорий;
* обеспечение создания отчета;
* обеспечение сохранения отчета в CSV формат;
* обеспечение создания групп и приглашения в них пользователей для совместного отслеживания доходов и расходов.