

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 __. __.20__
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 _____ И.В. Клейменов, ассистент
Руководитель _____ В.С. Тарасов, ст. преподаватель

Воронеж 2023

Содержание

Введение	5
1 Постановка задачи.....	6
1.1 Цели создания системы.....	6
1.2 Требования к разрабатываемой системе	6
1.3 Задачи проекта.....	6
1.3.1 Для авторизованного пользователя	6
1.3.2 Для неавторизованного пользователя.....	6
1.3.3 Для создателя группы	6
2 Анализ предметной области	7
2.1 Терминология	7
2.2 Обзор аналогов	8
2.2.1 CoinKeeper.....	8
2.2.2 Финансы	9
2.2.3 Дзен-мани	10
2.3 Моделирование системы.....	11
2.3.1 Диаграмма в стиле методологии IDEF0	11
2.3.2 Диаграмма прецедентов	12
2.3.3 Диаграммы классов.....	13
2.3.4 Диаграммы последовательности.....	14
2.3.5 Диаграмма развертывания.....	18
2.3.6 Диаграммы состояния.....	19
2.3.7 Диаграмма объектов	21
3 Реализация	22
3.1 Средства реализации	22

3.2 Реализация базы данных	22
3.2.1 ER-диаграмма.....	23
3.3 Реализация клиентской части	24
3.3.1 Экраны счетов.....	25
3.3.2 Экраны шаблонов	27
3.3.3 Экраны финансовых операций.....	31
3.3.4 Экран категорий.....	34
3.3.5 Экран отчета.....	37
3.3.6 Экраны кредитов.....	38
3.3.7 Экраны группы.....	41
3.3.8 Экран аккаунта пользователя.....	47
3.3.9 Экран регистрации.....	50
3.3.10 Экран входа	51
3.3.11 Экран восстановления пароля	52
3.3.12 Onboarding screen	53
3.4 Серверная часть	54
3.4.1 Структура модуля-приложения.....	55
3.4.2 Модуль — приложение users	56
3.4.3 Модуль — приложение operations	57
3.4.4 Модуль — приложение groups	58
3.4.5 Модуль — приложение onboard.....	59
3.4.6 Аутентификация и авторизация.....	60
3.4.7 Панель администратора.....	60
3.4.8 Шифрование данных	62
4 Тестирование	64

4.1.1 Тестирование сервера	64
4.1.2 Тестирование репозитория	64
4.1.3 Тестирование ViewModel	65
5 Аналитика	67
Заключение	69
Список использованных источников	70

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

1.1 Цели создания системы

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

1.2 Требования к разрабатываемой системе

- обеспечение учета доходов и расходов;
- обеспечение группировки индивидуальных финансовых операций;
- построение приложения на трехуровневой архитектуре.

1.3 Задачи проекта

1.3.1 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.

1.3.2 Для неавторизованного пользователя

- возможность просмотра экранов приложения для ознакомления;
- расчет ежемесячного кредитного платежа.

1.3.3 Для создателя группы

- возможность создания, удаления группы;
- приглашение участников в группу;
- исключение участников из группы.

2 Анализ предметной области

2.1 Терминология

- Мобильное приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для конкретной платформы (iOS, Android, Windows Phone 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).
- 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.
- 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.

2.2 Обзор аналогов

2.2.1 CoinKeeper

CoinKeeper – это приложение для ведения бюджета для Android и iOS, позволяет распределять доходы и расходы по категориям, создавать графики финансовых операций[3]. Интерфейс приложения представлен на рисунке 1.

Недостатки:

- ограниченная функциональность бесплатного режима;
- отсутствие кредитного калькулятора;
- данные сохраняются локально;

— отсутствие экспорта финансовых операций в CSV формат.

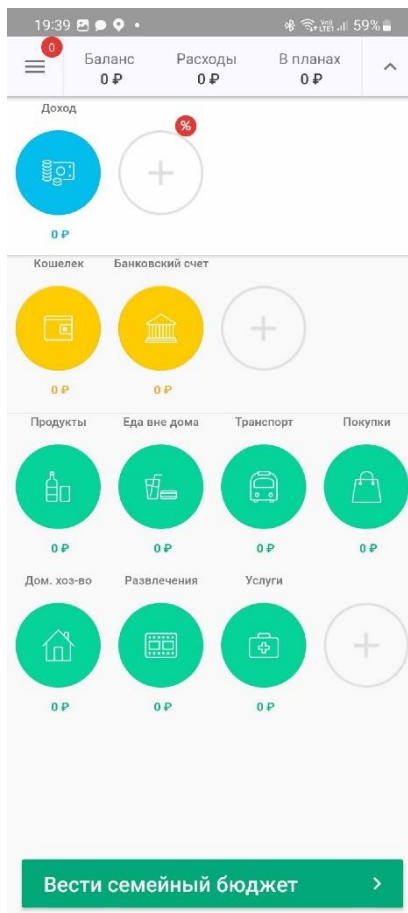


Рисунок 1 - Интерфейс приложения CoinKeeper

2.2.2 Финансы

Финансы — это приложение, которое позволяет отслеживать расходы и доходы, управлять финансами и планировать бюджет. Интерфейс приложения представлен на рисунке 2.

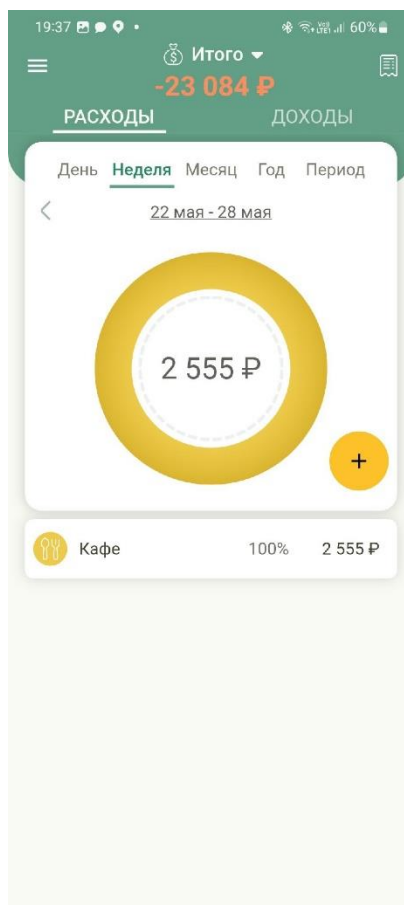


Рисунок 2 - Интерфейс приложения Финансы

Недостатки:

- отсутствие экспорта финансовых операций в CSV формат;
- отсутствие кредитного калькулятора.

2.2.3 Дзен-мани

Дзен-мани — это веб-сервис, с помощью которого можно вести учёт личных финансов и планировать свой личный бюджет[4]. Интерфейс приложения представлен на рисунке 3.

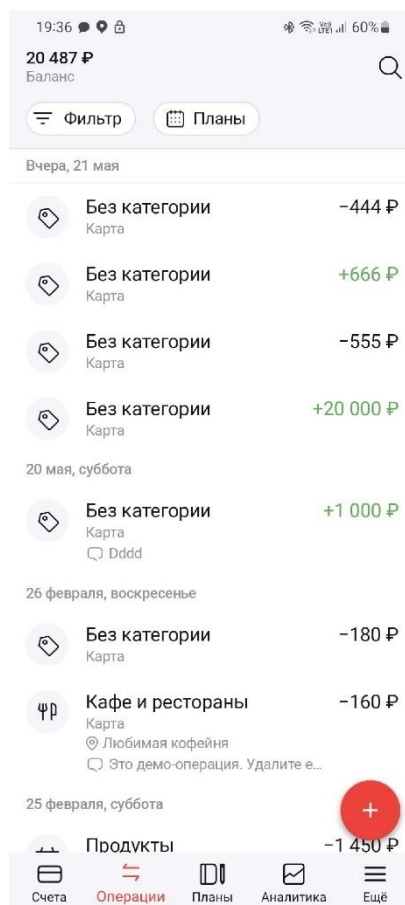


Рисунок 3 - Интерфейс приложения Дзен-мани

Недостатки:

- ограниченная функциональность бесплатного режима;
- отсутствие кредитного калькулятора;
- отсутствие экспорта финансовых операций в CSV формат в бесплатном режиме.

2.3 Моделирование системы

2.3.1 Диаграмма в стиле методологии IDEF0

Диаграмма в стиле методологии IDEF0 представлена на рисунке 4.

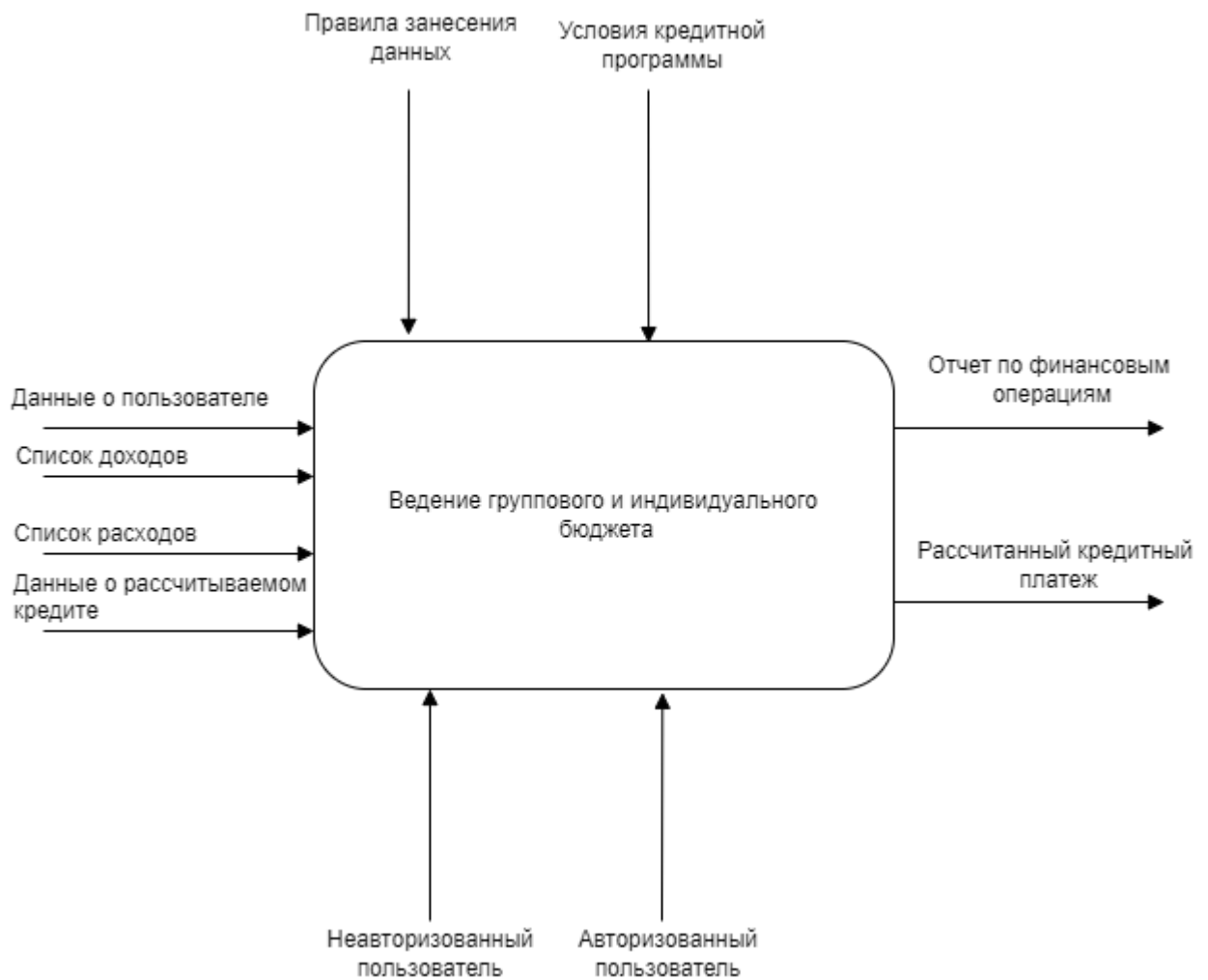


Рисунок 4 - Диаграмма в стиле методологии IDEF0

2.3.2 Диаграмма прецедентов

Рассмотрим полную диаграмму (рисунок 5) для использования приложения разными типами пользователей (неавторизованный, авторизованный, создатель группы).



Рисунок 5 - Use-Case диаграмма пользования приложением

2.3.3 Диаграммы классов

В данном пункте будут рассмотрены диаграммы классов-сущностей (рисунок 6) серверной части приложения.

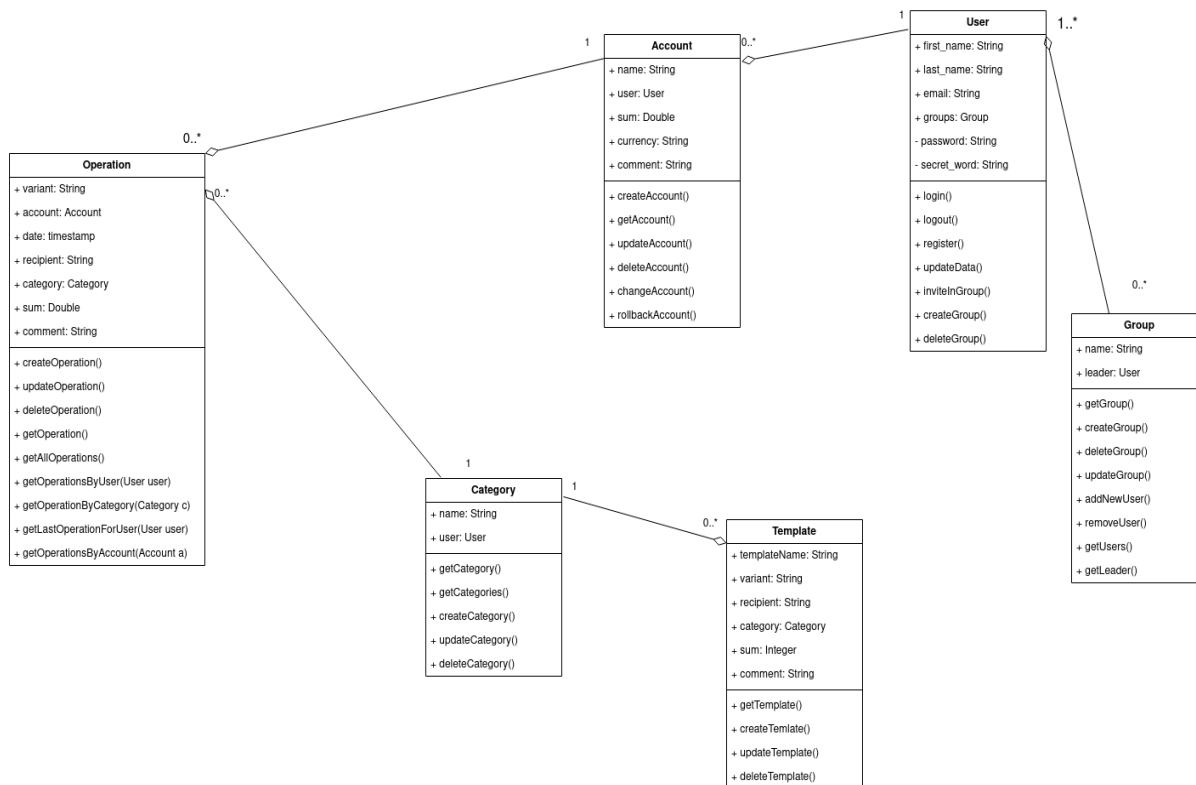


Рисунок 6 - Диаграмма классов-сущностей серверной части приложения.

2.3.4 Диаграммы последовательности

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

Рассмотрим диаграмму последовательности (рисунки 7-9) для пользователей разных типов:

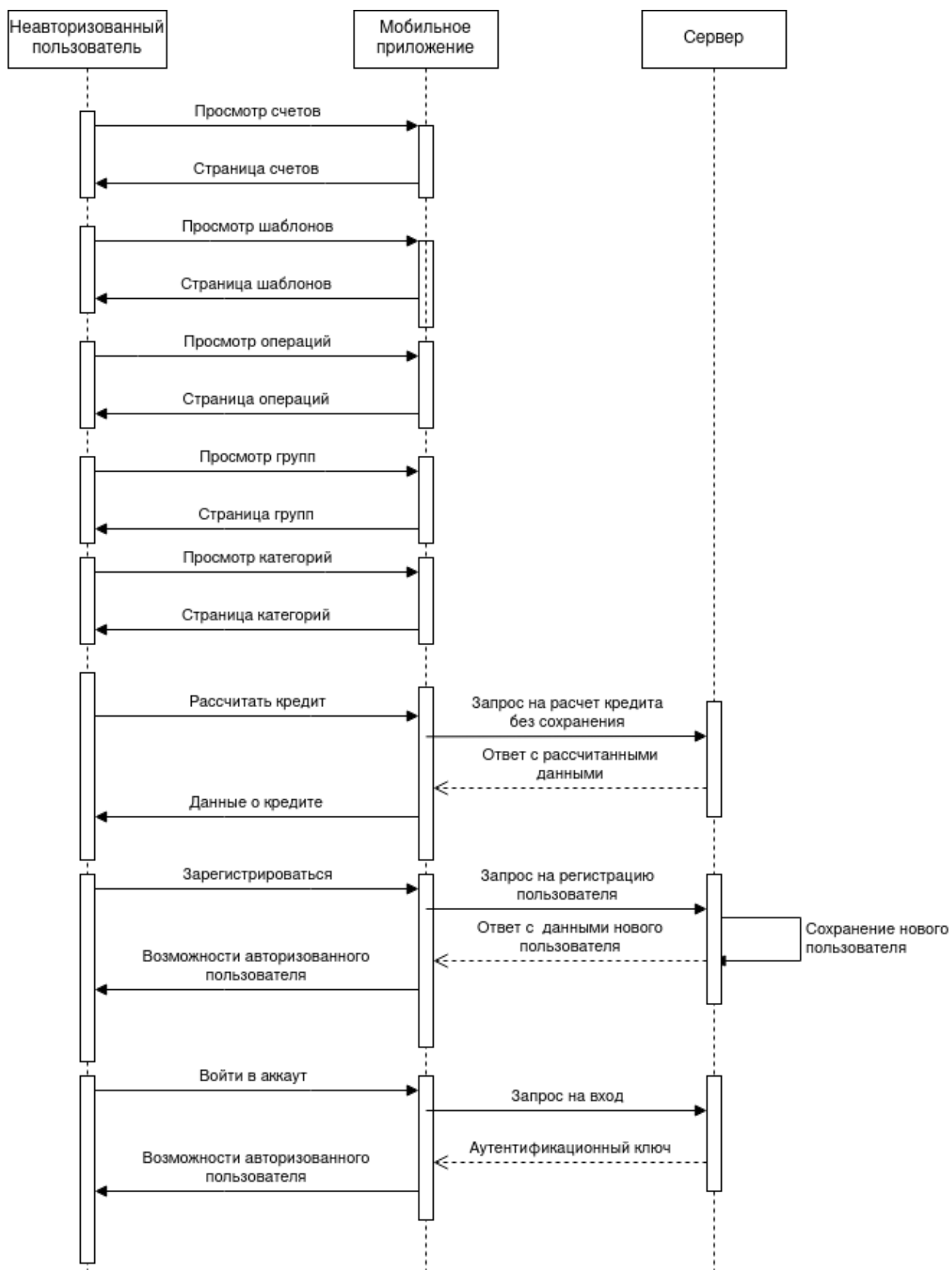


Рисунок 7 - Диаграмма последовательности неавторизованного пользователя

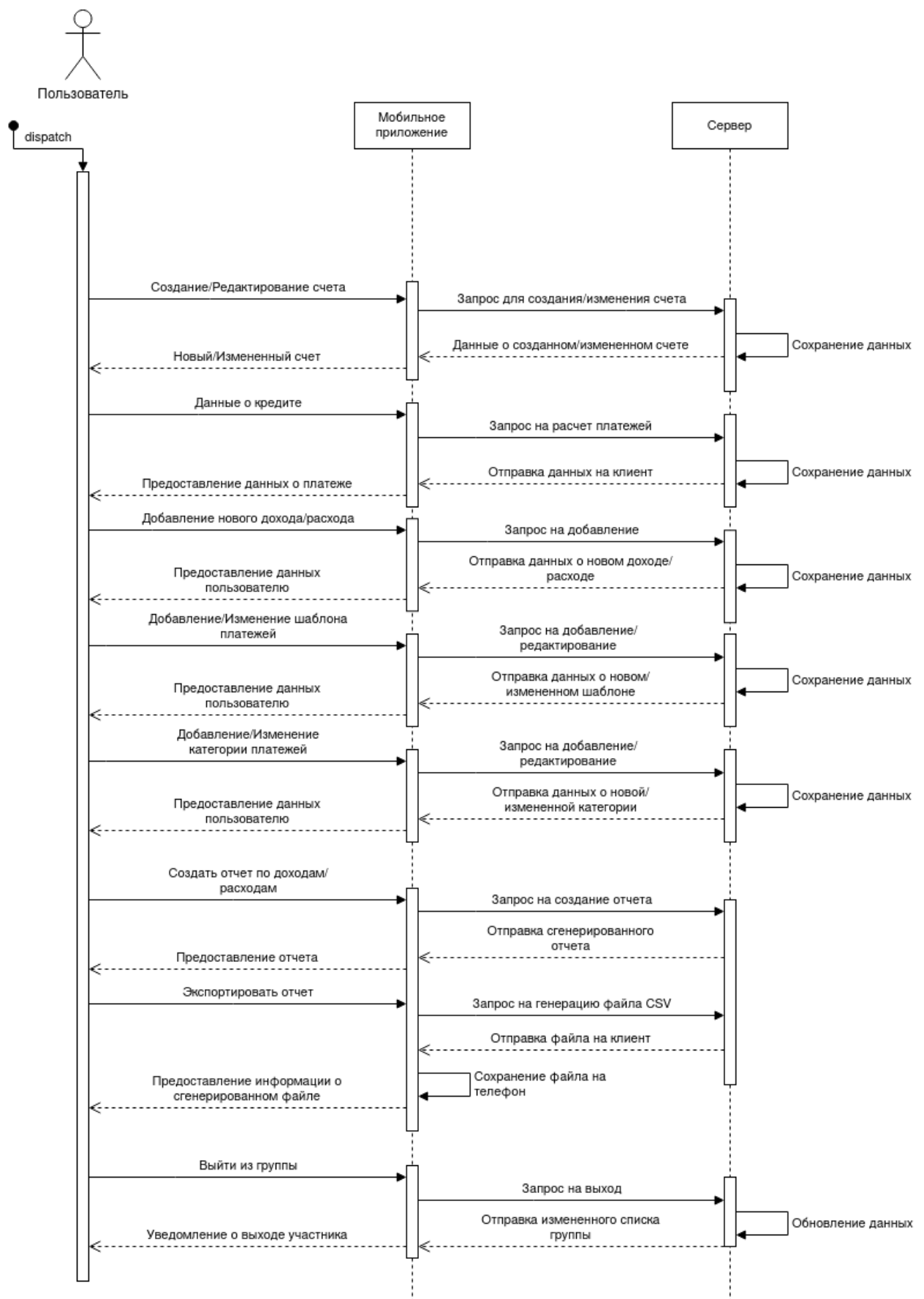


Рисунок 8 - Диаграмма последовательности авторизованного пользователя

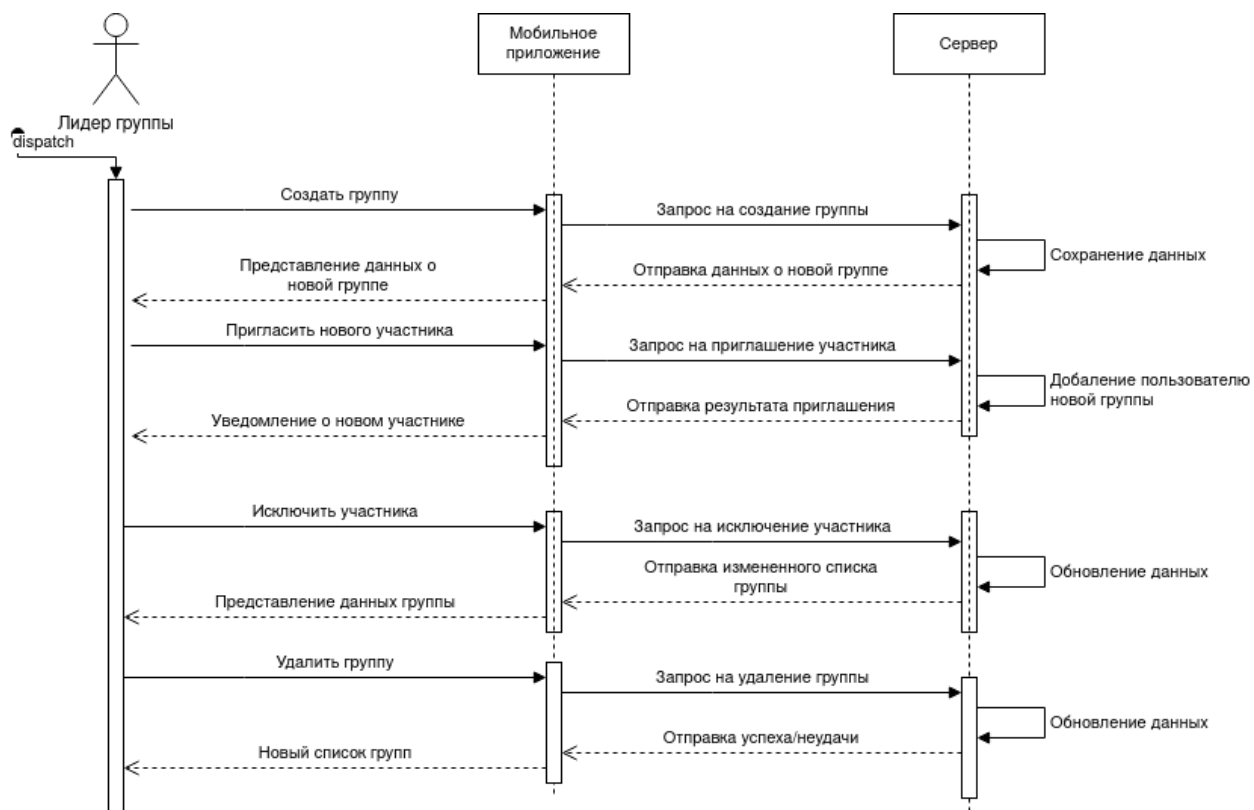


Рисунок 9 - Диаграмма последовательности создателя группы

2.3.5 Диаграмма развертывания

Диаграмма развертывания представлена на рисунке 10.

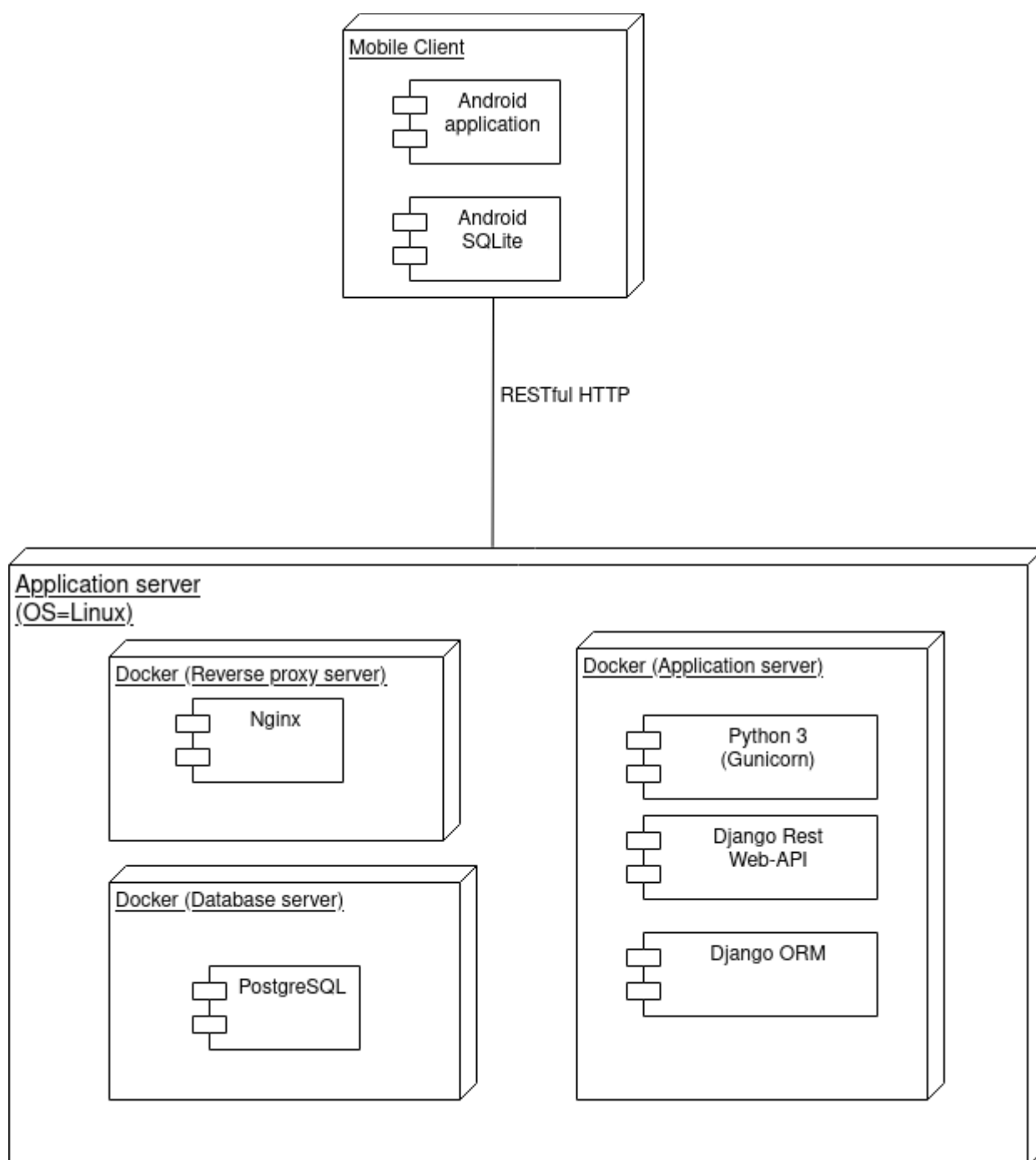


Рисунок 10 - Диаграмма развертывания приложения

2.3.6 Диаграммы состояния

Диаграмма состояния для авторизованного пользователя представлена на рисунке 11, для неавторизованного на рисунке 12.

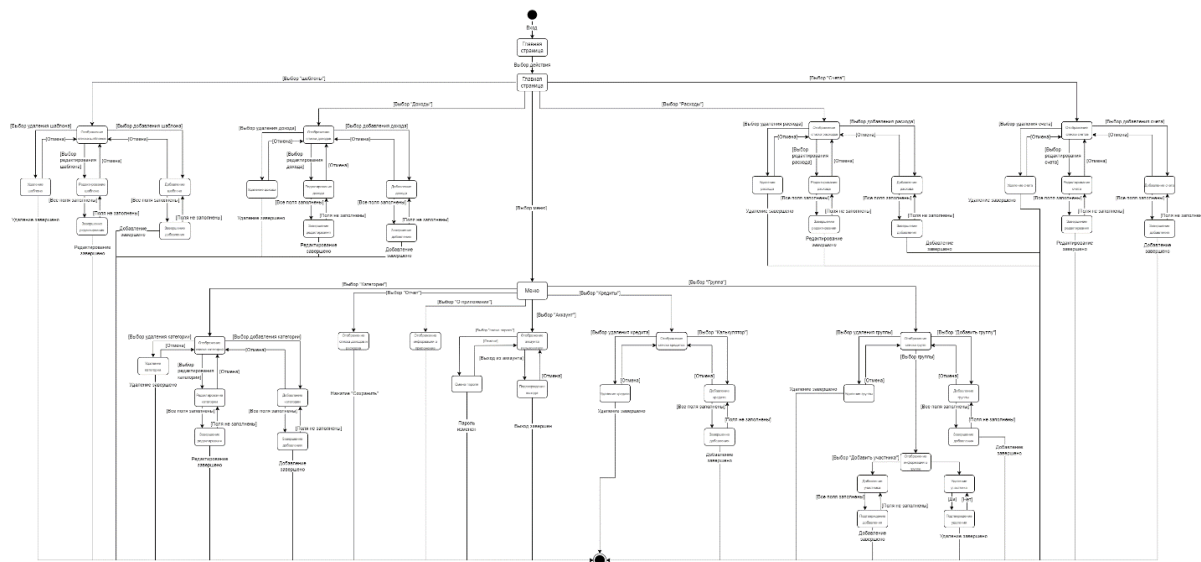


Рисунок 11 - Диаграмма состояния авторизованного пользователя

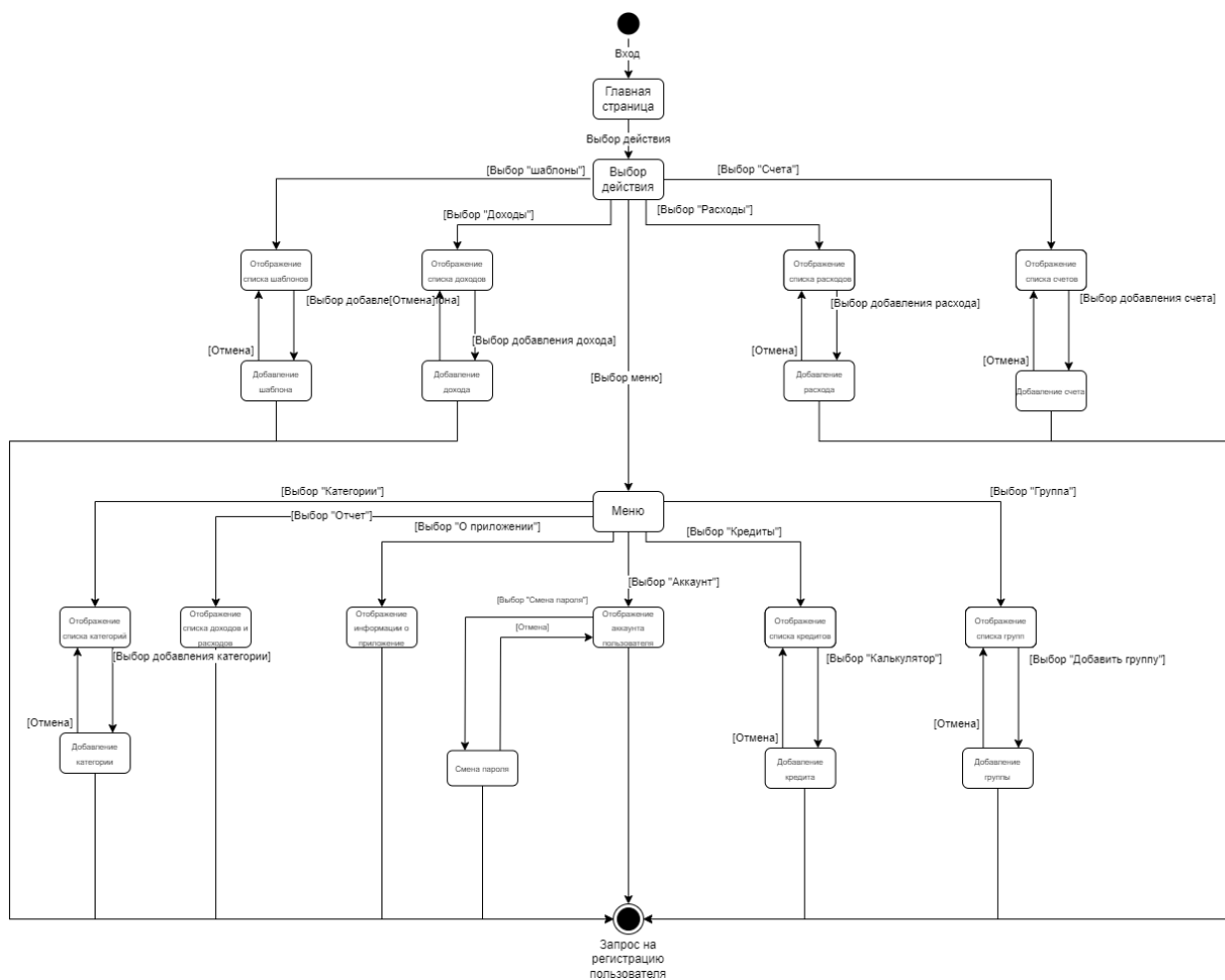


Рисунок 12 - Диаграмма состояния неавторизованного пользователя

2.3.7 Диаграмма объектов

Диаграмма объектов представлена на рисунке 13.

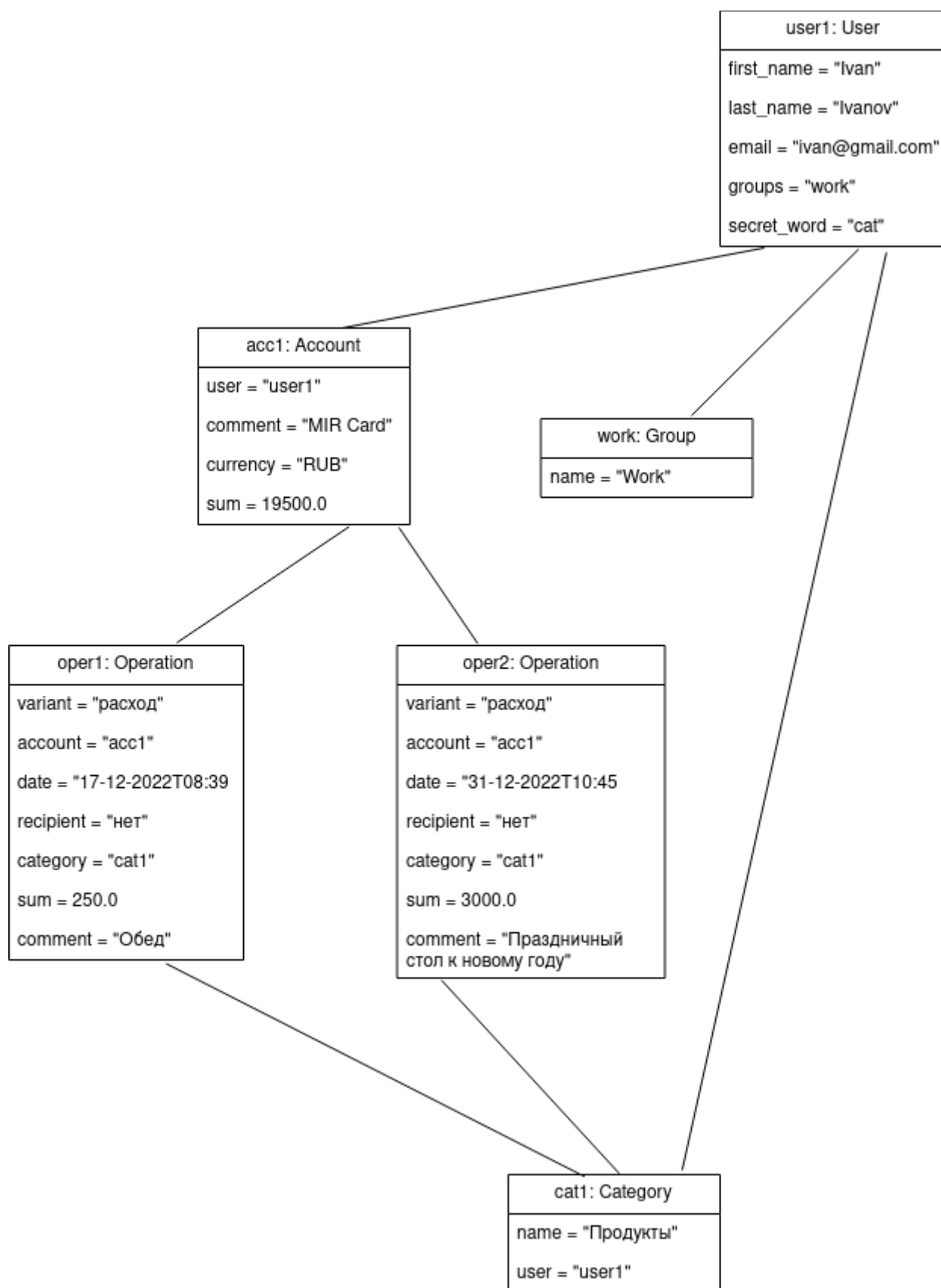


Рисунок 13 - Диаграмма объектов

3 Реализация

3.1 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.

Дополнительный инструментарий:

- Git;
- GitHub;
- Trello.

3.2 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.

3.2.1 ER-диаграмма

ER-диаграмма базы данных представлена на рисунке 14.

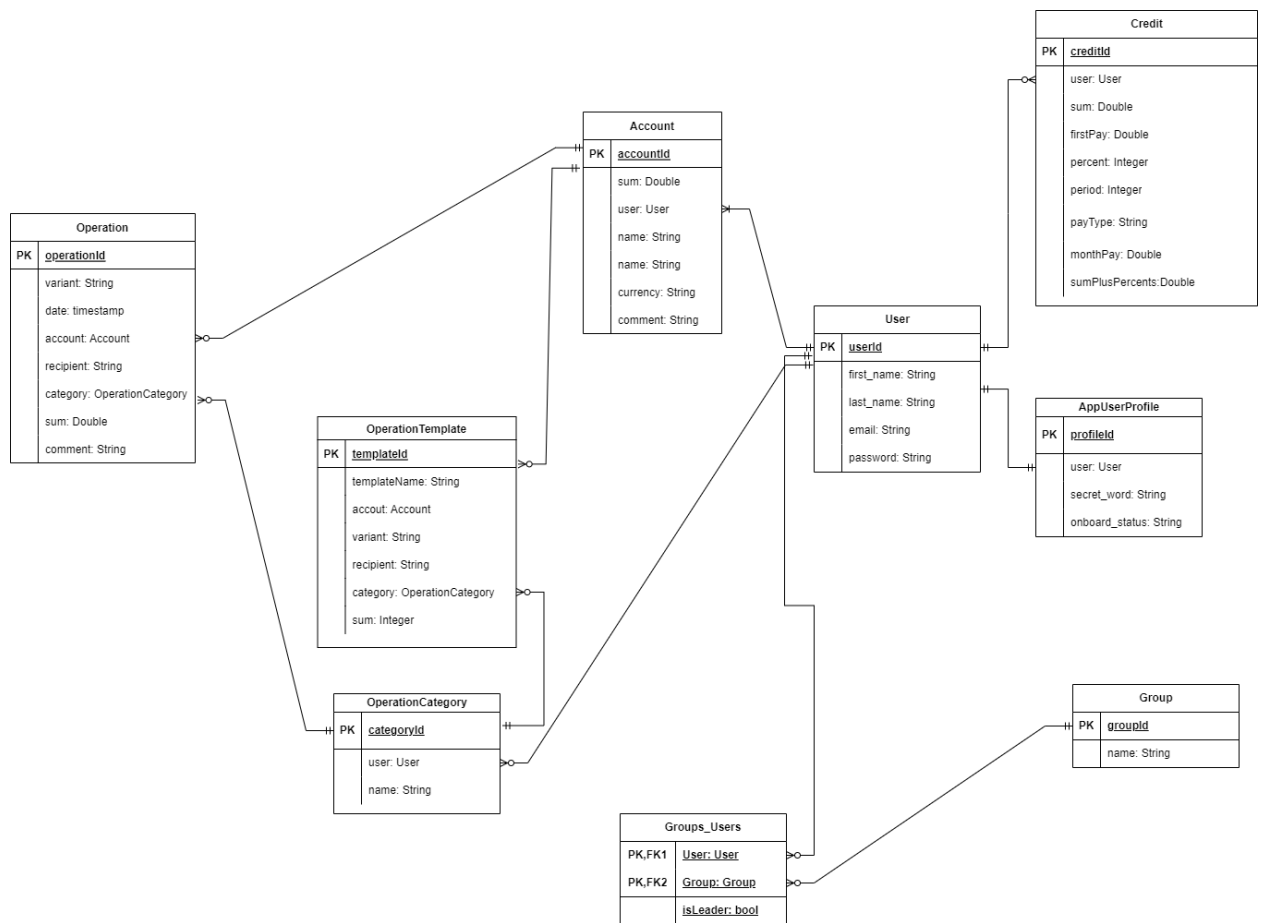


Рисунок 14 - ER-диаграмма базы данных

3.3 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.

3.3.1 Экраны счетов

На экране «Счета» (рисунок 15) 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».

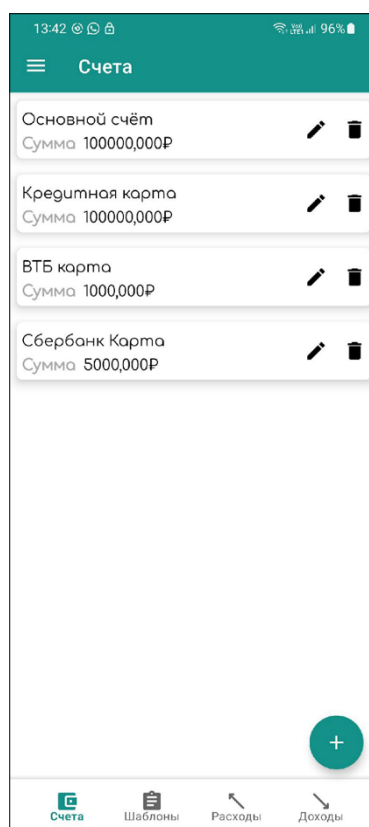


Рисунок 15 - Экран счетов

На экране «Счет» (рисунок 16), необходимого для создания нового счета, отображены поля с название счета, изначальной суммой и комментарием, ниже расположена кнопка для сохранения.

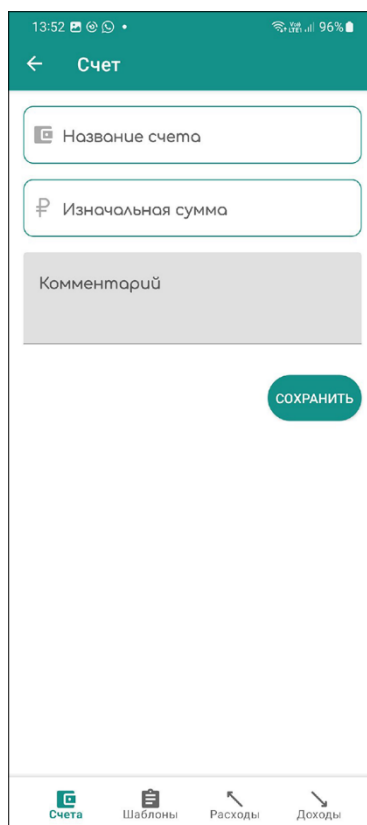


Рисунок 16 - Экран добавления счета

На экране «Редактирование счета» (рисунок 17) 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».

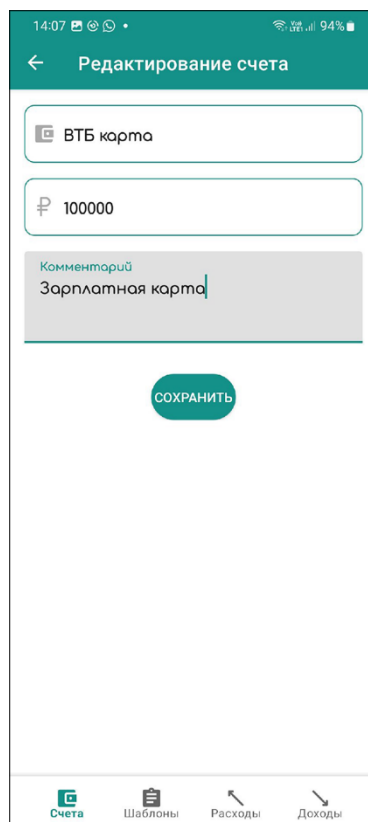


Рисунок 17 - Экран редактирования счета

3.3.2 Экраны шаблонов

На экране «Шаблоны» (рисунок 18) авторизованный пользователь имеет возможность просматривать уже добавленные шаблоны, редактировать и удалять их, а также создать новый при нажатии на кнопку.

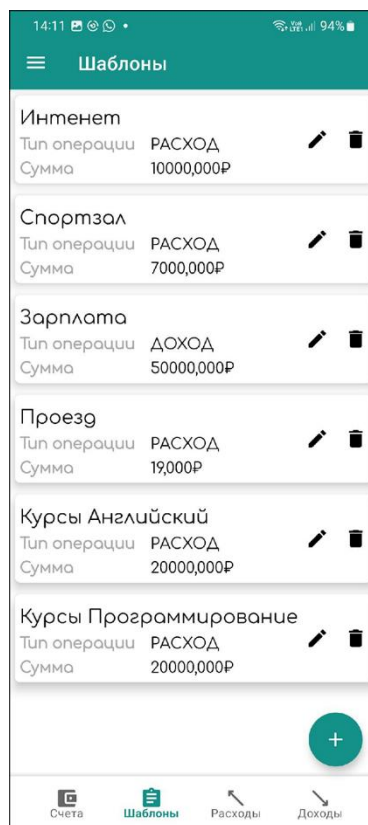


Рисунок 18 - Экран шаблонов

Экране «Шаблон» (рисунок 19) 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.

Рисунок 19 - Экран создания шаблона

На экране «Редактирование шаблона» (рисунок 20) 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».

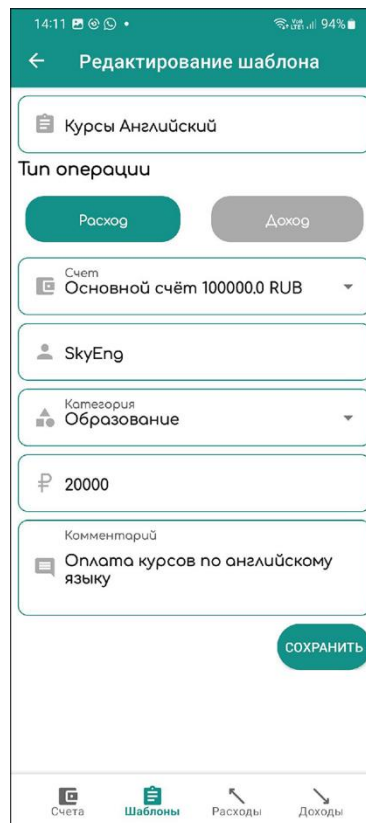


Рисунок 20 - Экран редактирования шаблона

При нажатии на шаблон, отображается экран «Добавление операции» (рисунок 21), в котором следующие поля предзаполнены: тип операции, счет, отправитель или получатель, категория, сумма, комментарий.

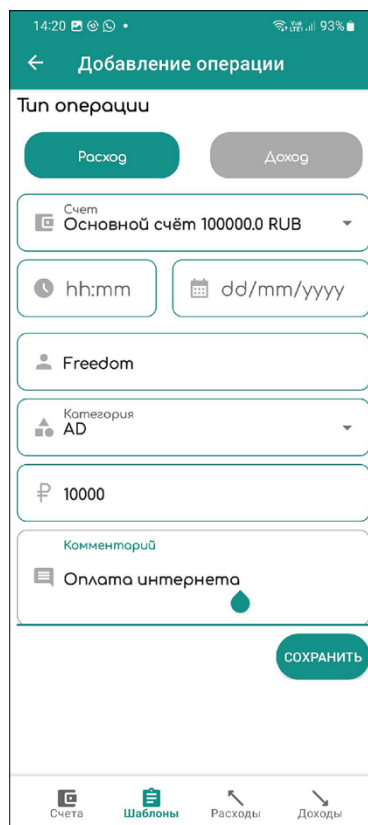


Рисунок 21 - Экран добавления операции при нажатии на шаблон

3.3.3 Экраны финансовых операций

На экране «Доходы» и «Расходы» (рисунок 22) авторизованный пользователь имеет возможность просматривать уже добавленные финансовые операции, редактировать, удалять их, а также создать новый при нажатии на кнопку.

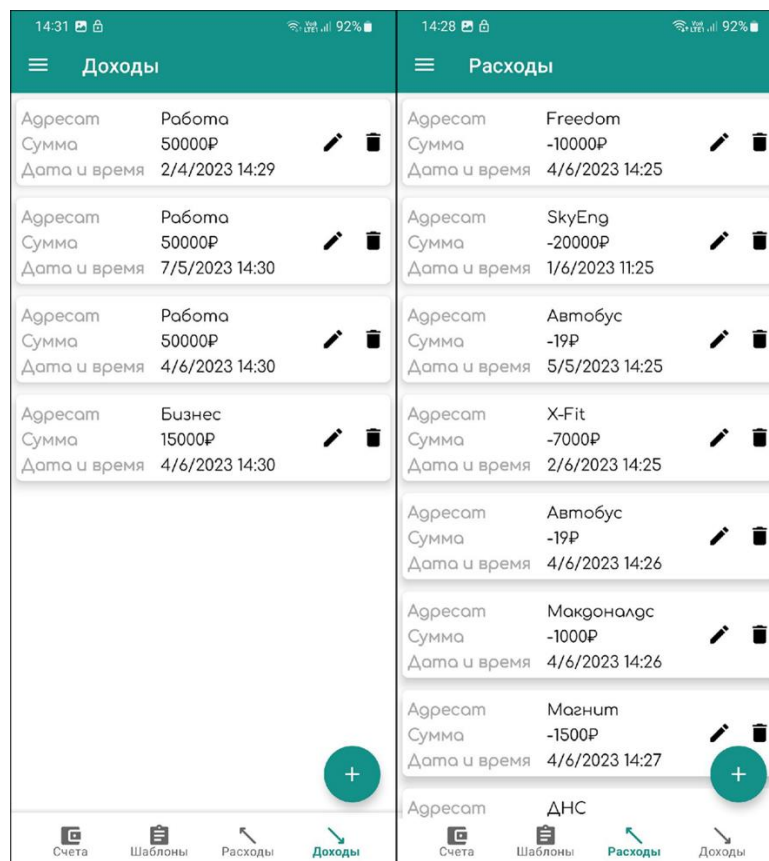


Рисунок 22 - Экраны доходов и расходов

На экране «Добавление операции» (рисунок 23) отображены поля с типом операции, выбором счета, времени и даты, отправителем или получателем, выбором категории, суммы и комментария. Ниже расположена кнопка для сохранения.

Рисунок 23 - Экран добавления операции

На экране «Редактирование операции» (рисунок 24) отображены поля с типом операции, выбором счета, времени и даты, отправителем или получателем, выбором категории, суммы и комментария. Ниже расположена кнопка для сохранения.

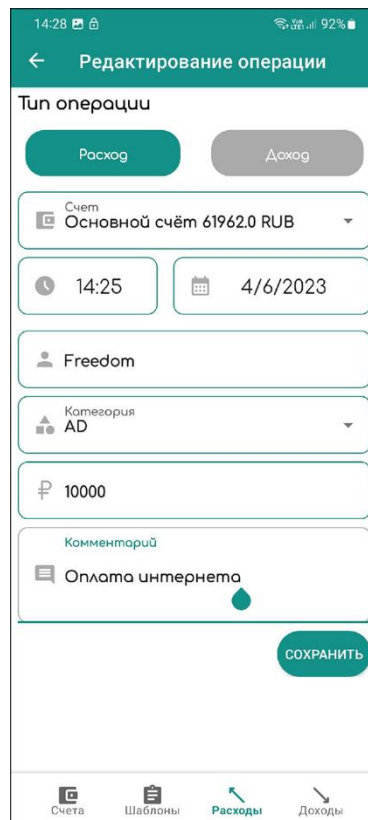


Рисунок 24 - Экран редактирование операции

3.3.4 Экран категорий

На экране «Категории» (рисунок 25) авторизованный пользователь имеет возможность просматривать уже добавленные категории, редактировать и удалять их, а также создать новый при нажатии на кнопку. При первой регистрации пользователя автоматически создается три категории: «Продукты», «Транспорт» и «Развлечения».

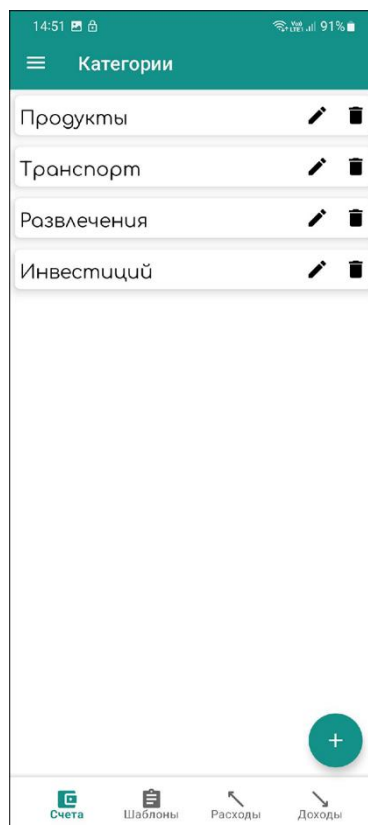


Рисунок 25 - Экран категорий

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 (рисунок 26), схожее с диалоговым окном, открывающимся при добавлении новой категории.

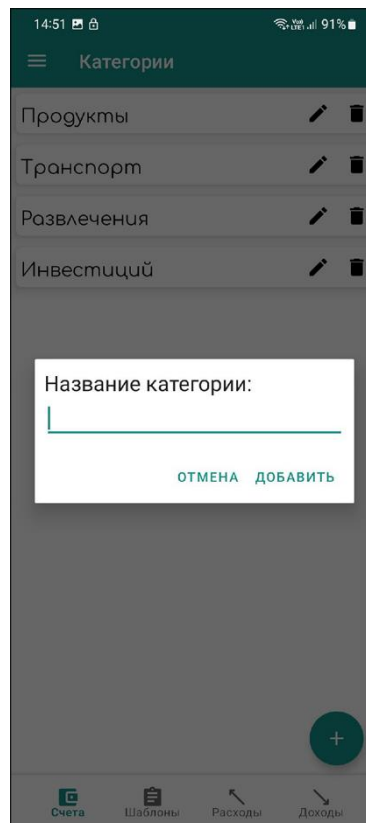


Рисунок 26 - Диалоговое окно создания и редактирования категории

При попытке удаления категории, путем нажатия на кнопку с изображением мусорной корзины, появится диалоговое окно (рисунок 27) с подтверждением удаления категории.

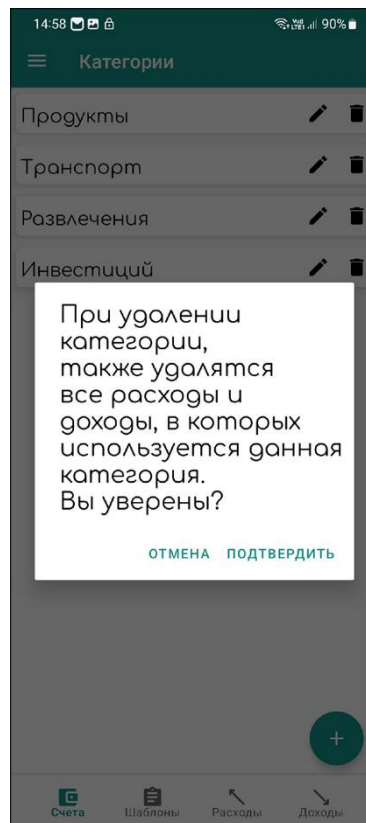


Рисунок 27 - Диалоговое окно удаления категории

3.3.5 Экран отчета

На данном экране (рисунок 28) отображены все финансовые операции, добавленные авторизованным пользователем. Под списком операций имеется кнопка для сохранения отчета, при нажатии на нее составляется таблица в формате «CSV».

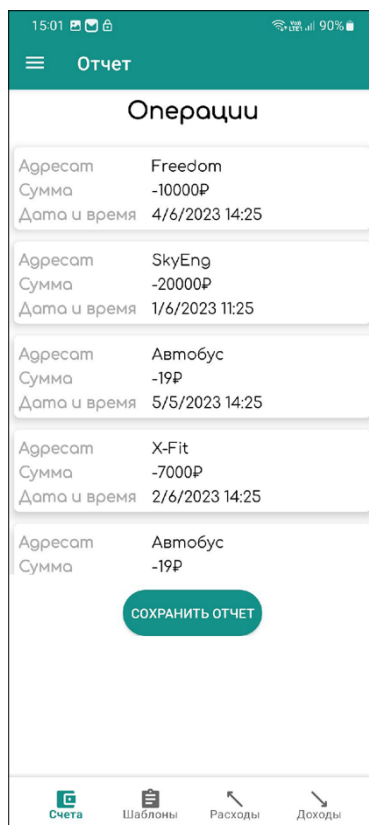


Рисунок 28 - Экран отчета

Таблица (рисунок 29) сохраняется в локальное хранилище устройства в директории «Документы».

	A	B	C	D	E	F
1	Расходы					
2	Номер	Операции	Дата	Получатель/Получатель	Сумма	Комментарий
3		6 РАСХОД	2023-06-04T14:25:00Z	Freedom	10000.0	оплата интернета
4		7 РАСХОД	2023-06-01T11:25:00Z	SkyEng	20000.0	Оплата английского
5		8 РАСХОД	2023-05-05T14:25:00Z	Автобус	19.0	Оплата проезда
6		9 РАСХОД	2023-06-02T14:25:00Z	X-Fit	7000.0	Оплата тренера
7		10 РАСХОД	2023-06-04T14:26:00Z	Автобус	19.0	Оплата проезда
8		11 РАСХОД	2023-06-04T14:26:00Z	Макдоналдс	1000.0	Поел в маке
9		12 РАСХОД	2023-06-04T14:27:00Z	Магнит	1500.0	Купил продукты
10		13 РАСХОД	2023-06-04T14:27:00Z	ДНС	37000.0	Купил новый ноутбук
11	Доходы					
12	Номер	Операции	Дата	Получатель/Получатель	Сумма	Комментарий
13		16 ДОХОД	2023-04-02T14:29:00Z	Работа	50000.0	Зарплата
14		17 ДОХОД	2023-05-07T14:30:00Z	Работа	50000.0	Зарплата
15		18 ДОХОД	2023-06-04T14:30:00Z	Работа	50000.0	Зарплата
16		19 ДОХОД	2023-06-04T14:30:00Z	Бизнес	15000.0	Получил прибыль
17						

Рисунок 29 - Результат сохранения отчета

3.3.6 Экраны кредитов

На экране «Ваши кредиты» (рисунок 30) авторизованный пользователь имеет возможность просмотреть ранее добавленные кредиты, а также

добавить новый при нажатии на кнопку, расположенной в правом нижнем углу экрана.

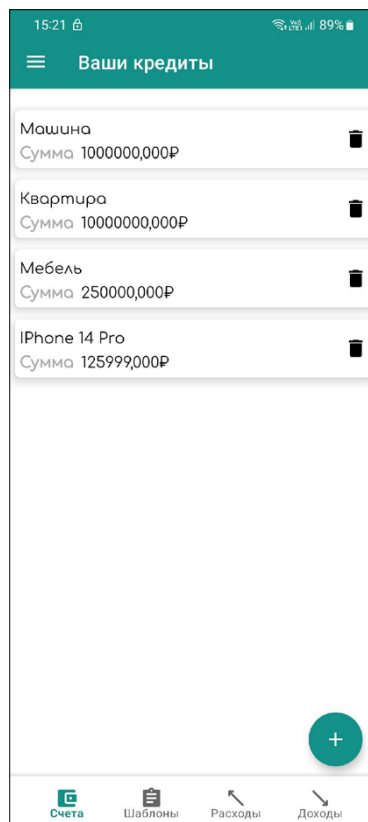


Рисунок 30 - Экран кредитов

Рисунок 31 - Экран добавления кредита

После корректного ввода всех полей и нажатие на кнопку «Рассчитать», а также при нажатии на кредит из экрана «Ваши кредиты» (рисунок 31), отобразится экран «Кредит» (рисунок 32), 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».

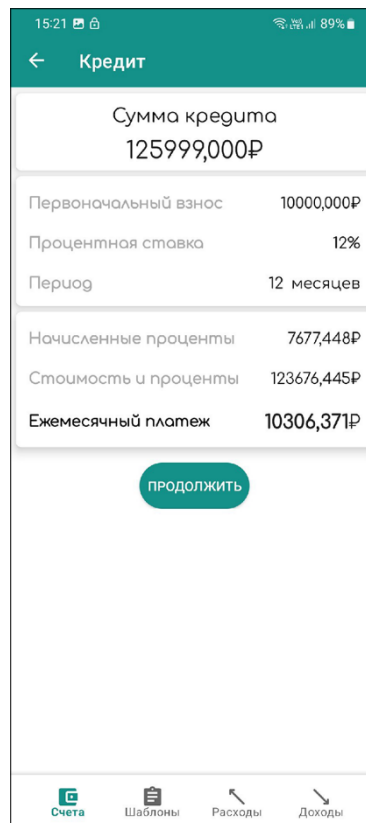


Рисунок 32 - Экран просмотра информации о кредите

3.3.7 Экраны группы

На данном экране (рисунок 33) расположены список групп, в которых состоит пользователь, а также кнопка, при нажатии на которую, всплывает диалоговое окно.

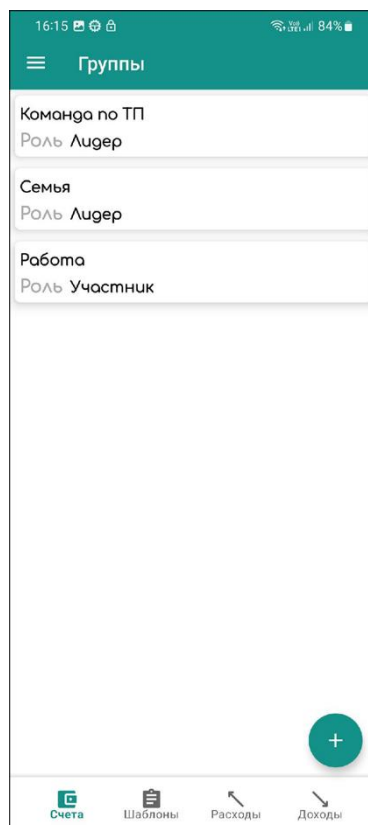


Рисунок 33 - Экран просмотра списка групп

Диалоговое окно (рисунок 34) содержит поле для ввода почты пользователя, а также две кнопки «Отмена» и «Добавить».

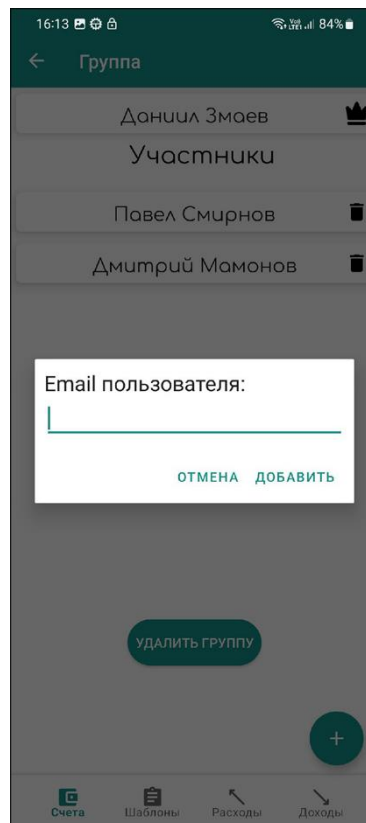


Рисунок 34 - Диалоговое окно добавления пользователя

При нажатии на группу, открывается экран «Группа» (рисунок 35), в котором отображены все участники, состоящие в группе.

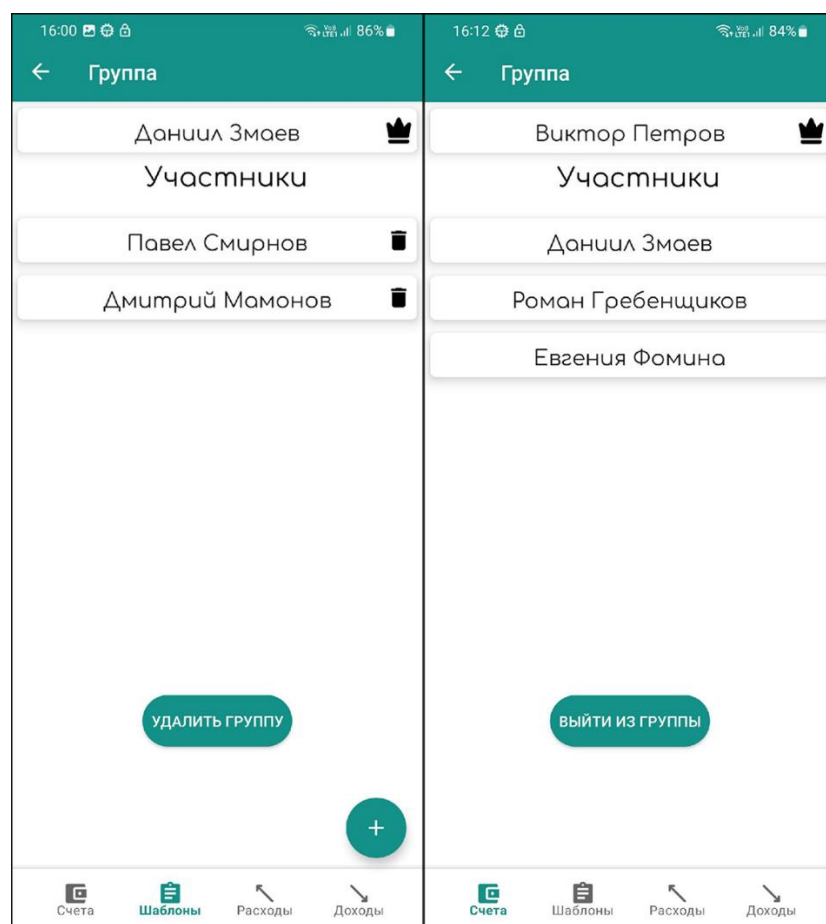


Рисунок 35 - Экран группы для создателя и участника

При нажатии на кнопку с изображением мусорной корзины, для удаления пользователя из группы, или при нажатии на кнопку «Выйти из группы» всплывает диалоговое окно (рисунок 36).

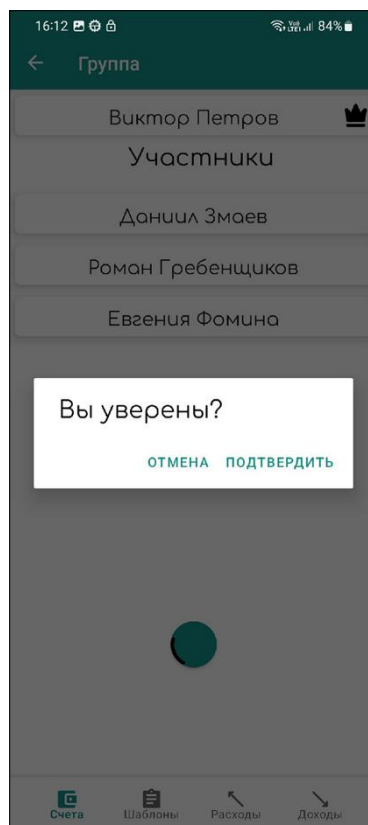


Рисунок 36 - Диалоговое окно выхода или исключения пользователя из группы

При нажатии на кнопку «Удалить группы» создателем группы, отображается диалоговое окно с подтверждением действия (рисунок 37).

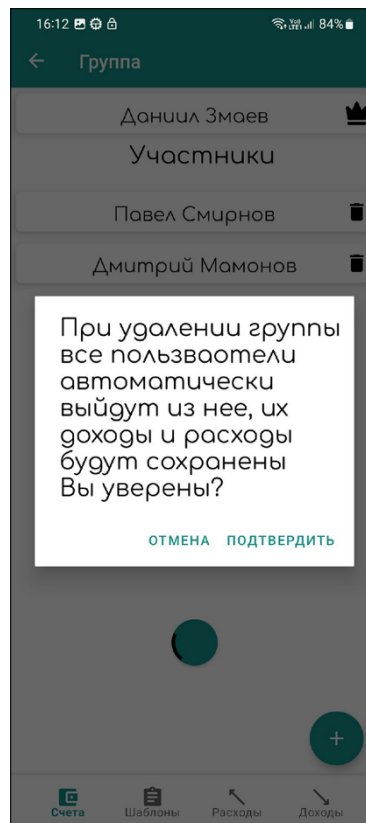


Рисунок 37 - Диалоговое окно удаления группы

При нажатии на участника группы отображается экран финансовых операций выбранного участника (рисунок 38).

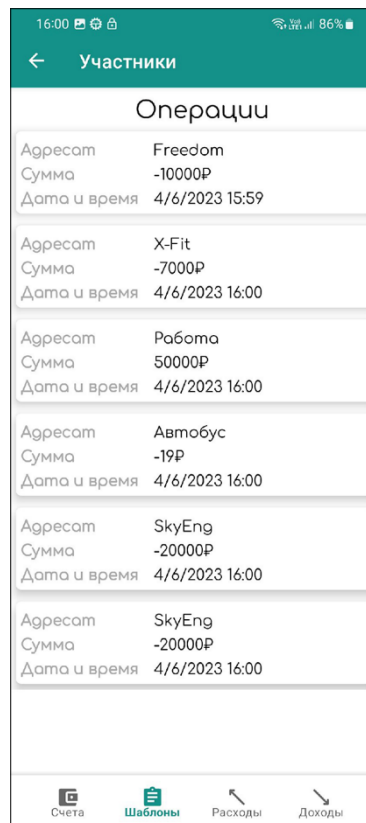


Рисунок 38 - Экран операций участника группы

3.3.8 Экран аккаунта пользователя

На данном экране (рисунок 39) расположена информация о пользователе: имя и фамилия и почта. Также имеются две кнопки: «Редактировать» и «Выйти»

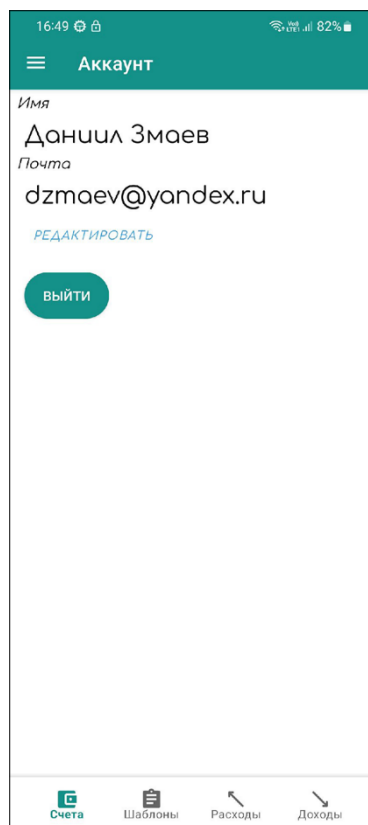


Рисунок 39 - Экран аккаунта

При нажатии на кнопку «Редактировать», отображается экран редактирования аккаунта пользователя (рисунок 40), в котором расположены следующие поля: имя, фамилия. Под полем фамилия имеется кнопка «Подтвердить», необходимая для сохранения отредактированных данных.

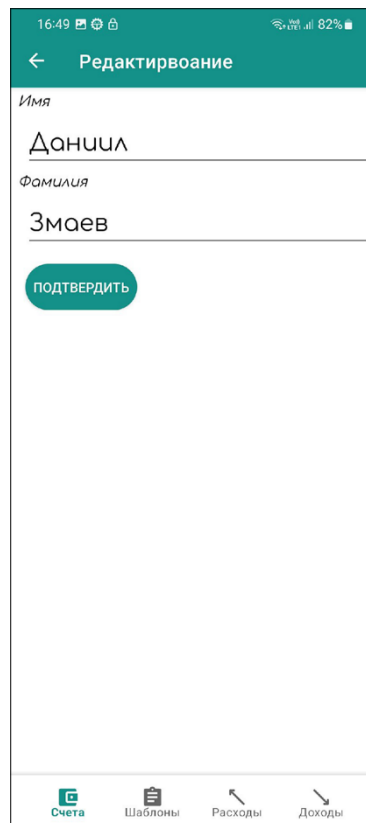


Рисунок 40 - Экран редактирования аккаунта

При нажатии на кнопку «Выйти», появляется диалоговое окно (рисунок 41), необходимое для подтверждения действия.



Рисунок 41 - Диалоговое окно выхода из аккаунта

3.3.9 Экран регистрации

На данном экране (рисунок 42) 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.

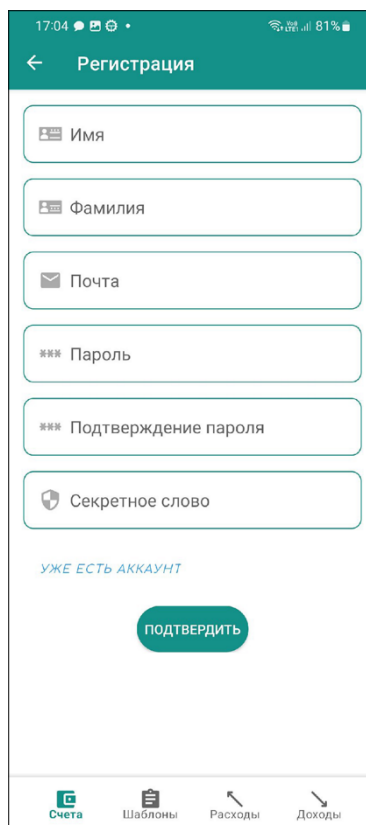


Рисунок 42 - Экран регистрации

3.3.10 Экран входа

На данном экране (рисунок 43) 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».

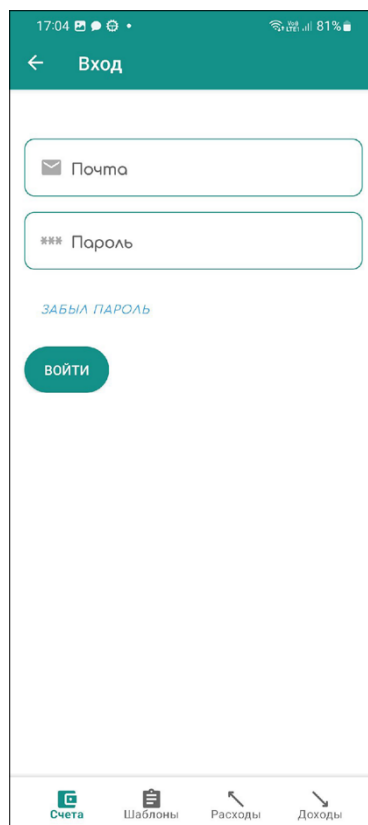


Рисунок 43 - Экран входа

3.3.11 Экран восстановления пароля

На данном экране (рисунок 44) 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».

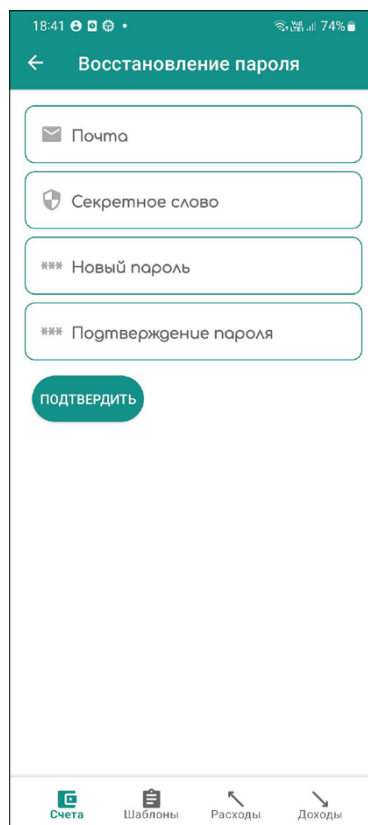


Рисунок 44 - Экрана восстановления пароля

3.3.12 Onboarding screen

При первом запуске приложения отображаются экраны (рисунок 45) 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.

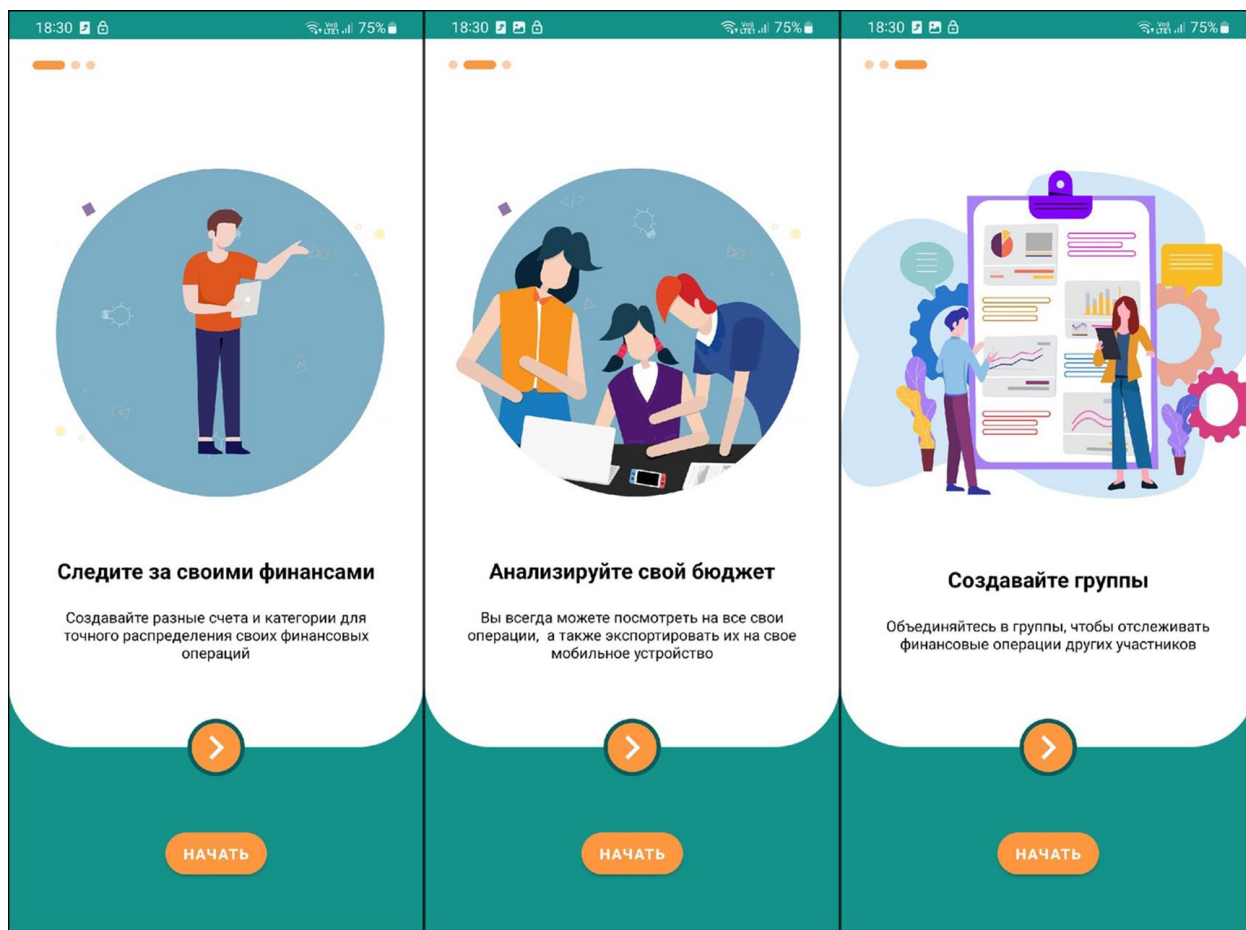


Рисунок 45 - Onboarding screens

3.4 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.

3.4.1 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`;

3.4.2 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`).

3.4.3 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 операций;
- `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).

3.4.4 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`).

3.4.5 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.

3.4.6 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.

3.4.7 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 (рисунок 46) в панель управления (рисунок 47).

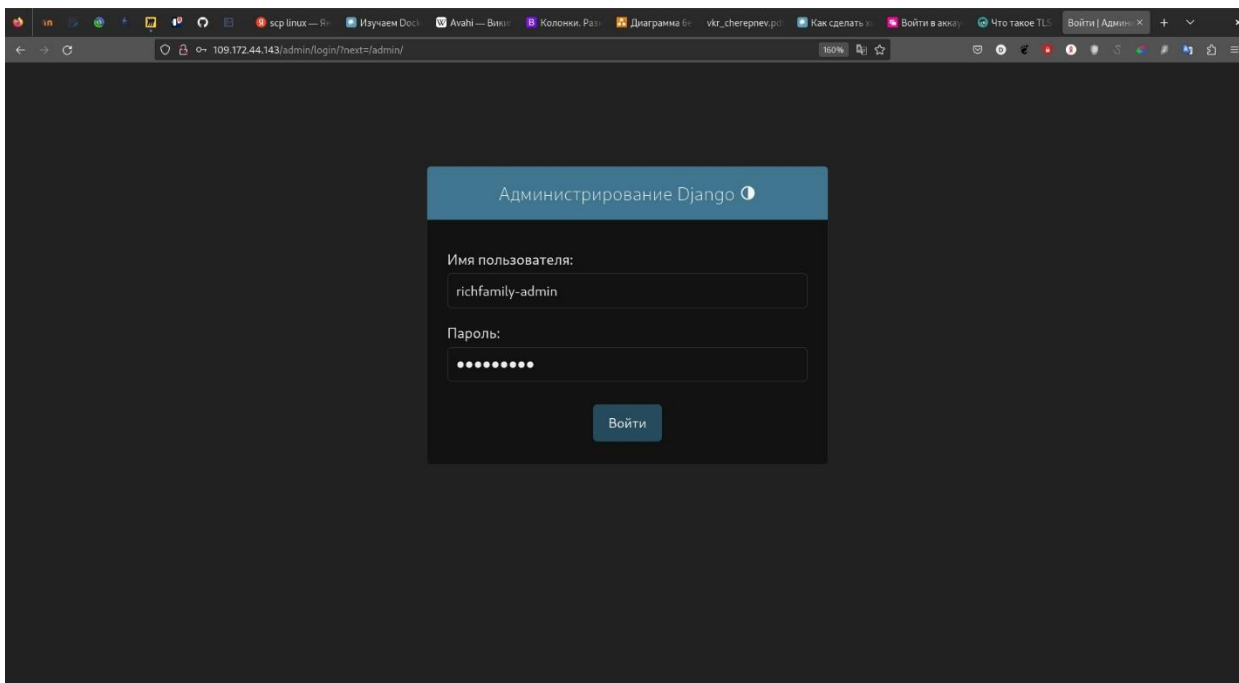


Рисунок 46 - Вход администратора в панель

Как только пользователь произвел вход, для него открывается страница управления (рисунок 47), где будут расположен список моделей, доступных для создания и редактирования. При нажатии на модель открывается страница со списком записей в базе данных (рисунок 48), и можно просматривать все записи по отдельности, редактировать (рисунок 49) их и создавать новые.

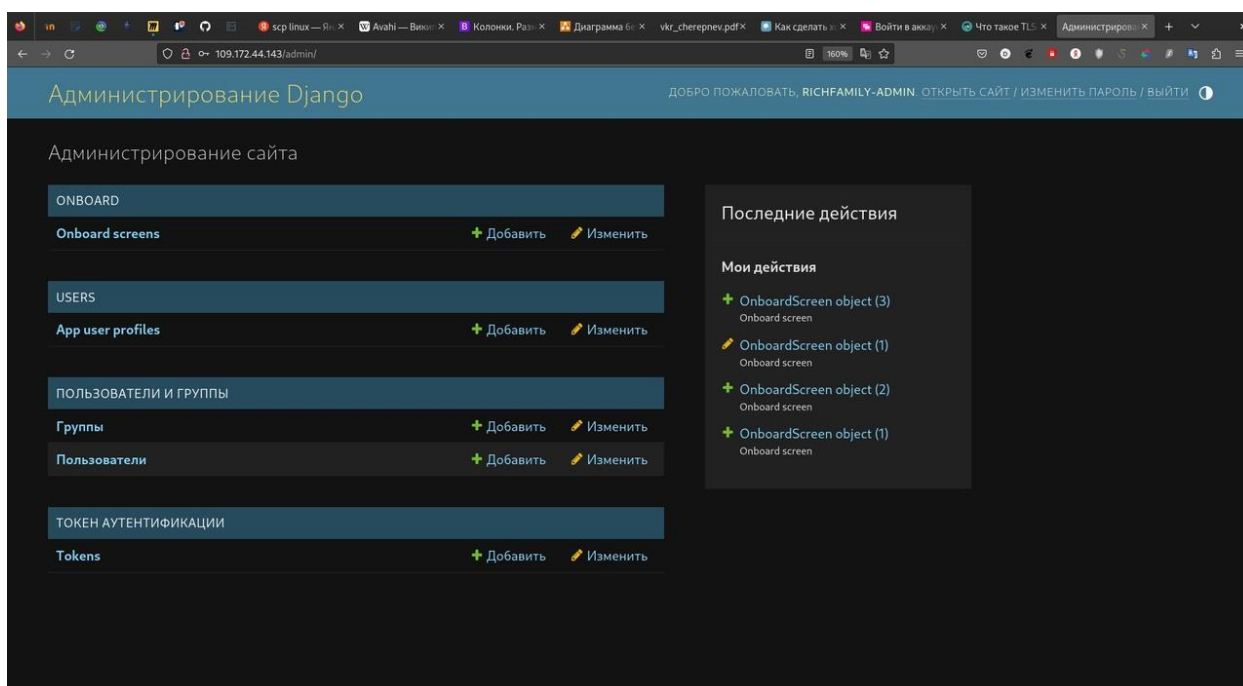


Рисунок 47 - Главная страница панели администратора

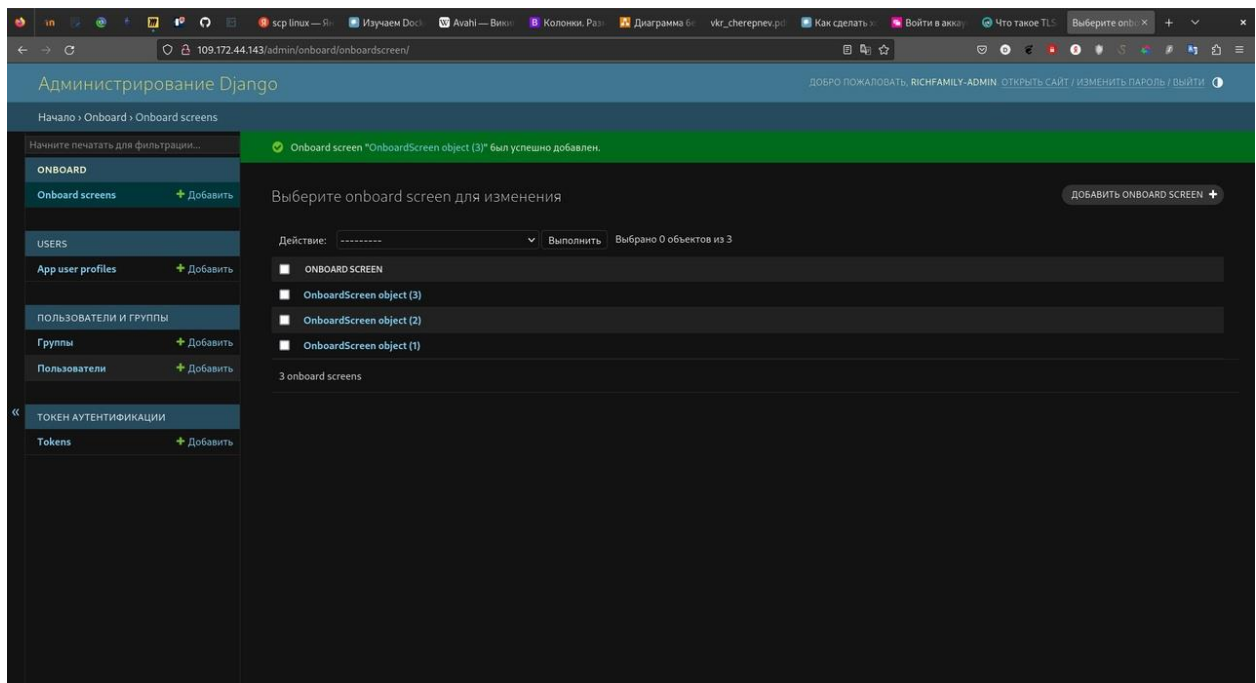


Рисунок 48 - Просмотр записей модели OnboardingScreen

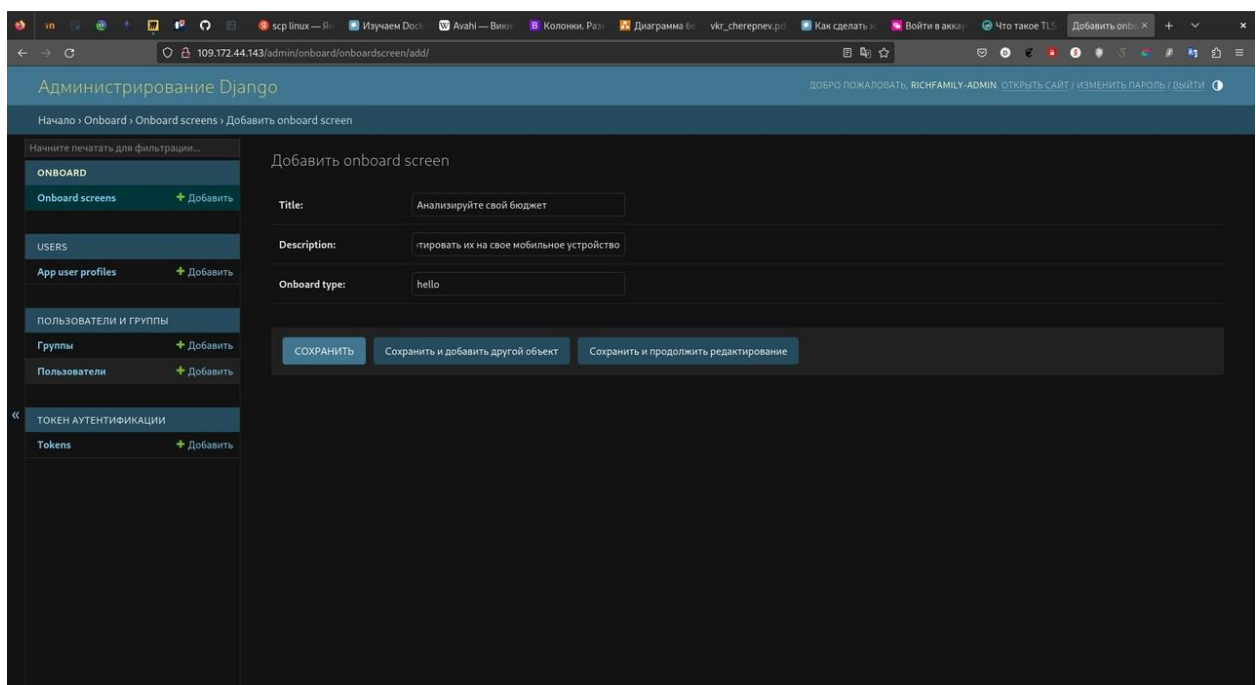


Рисунок 49 - Страница редактирования записи модели OnboardingScreen

3.4.8 Шифрование данных

Для обеспечения безопасности при передаче трафика между клиентским приложением и сервером в качестве протокола передачи данных был выбран HTTPS. Для удаленного хостинга, на котором развернуто серверное приложение, был зарегистрирован домен. После чего для хостинга был

сгенерирован SSL-сертификат, подтвержденный центром сертификации Let's Encrypt[11] через специальную утилиту Certbot[12] именно для этого домена. Затем в конфигурации веб-сервера nginx была создана новая запись для сгенерированного сертификата, и он был добавлен через volume в контейнер веб-сервера. Также для обеспечения HTTPS запросов на веб-сервер для него был открыт на прослушивание порт 443 и установлено перенаправление всех незащищенных запросов на этот порт для автоматического шифрования данных. Так как сертификат подписан уже определенным центром сертификации, то новых сертификатов для определения корректности корневого создавать не приходится.

4 Тестирование

4.1.1 Тестирование сервера

Результат тестирования сервера изображен на рисунках 50-52.

: 15 total, 15 passed			1.84 s
			Collapse Expand
operations			1.84 s
tests			1.84 s
Operations Tests			1.84 s
test_account_create	passed		123 ms
test_account_delete	passed		140 ms
test_account_update	passed		119 ms
test_credit_pay_create	passed		123 ms
test_credit_pay_delete	passed		124 ms
test_credit_pay_update	passed		123 ms
test_operation_category_create	passed		120 ms
test_operation_category_delete	passed		119 ms
test_operation_category_update	passed		121 ms
test_operation_create	passed		119 ms
test_operation_delete	passed		124 ms
test_operation_template_create	passed		121 ms
test_operation_template_delete	passed		122 ms
test_operation_template_update	passed		119 ms
test_operation_update	passed		120 ms

Рисунок 50 - Результат тестирования operations

: 3 total, 3 passed			6 ms
			Collapse Expand
tests			6 ms
GroupsTests			6 ms
test_group_create	passed		3 ms
test_group_delete	passed		2 ms
test_group_update	passed		1 ms

Рисунок 51 - Результат тестирования groups

: 3 total, 3 passed			371 ms
			Collapse Expand
users			371 ms
tests			371 ms
UsersTests			371 ms
test_app_user_profile_create	passed		124 ms
test_app_user_profile_delete	passed		124 ms
test_app_user_profile_update	passed		123 ms

Рисунок 52 - Результат тестирования users

4.1.2 Тестирование репозиториев

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

Для тестирования репозитория использовались фреймворки, такие как JUnit и Mockito. Результат тестирования репозитория изображен на рисунке 53.

ru.vsu.cs.tp.richfamily.CategoryRepositoryTest and 2 more: 15 total, 15 passed			758 ms
			Collapse Expand
ru.vsu.cs.tp.richfamily.CategoryRepositoryTest			692 ms
should edit category and return true	passed		683 ms
should add category and return true	passed		1 ms
should delete category test and return true	passed		1 ms
should not delete category test and throws NullPointerException	passed		5 ms
should get categories test and return true	passed		2 ms
ru.vsu.cs.tp.richfamily.TemplateRepositoryTest			27 ms
should get template and return true	passed		23 ms
should edit template and return true	passed		1 ms
should add template and return true	passed		1 ms
should delete template and return true	passed		1 ms
should not delete template and throws NullPointerException	passed		1 ms
ru.vsu.cs.tp.richfamily.WalletRepositoryTest			39 ms
should not delete wallet and throws NullPointerException	passed		36 ms
should edit wallet and return true	passed		1 ms
should add wallet and return true	passed		1 ms
should get wallet and return true	passed		1 ms
should delete wallet and return true	passed		0 ms

Рисунок 53 - Результат тестирования репозитория

4.1.3 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. Результат тестирования ViewModel изображен на рисунке 54.

ru.vsu.cs.tp.richfamily.WalletViewModelTest and 2 more: 6 total, 6 passed			751 ms
			Collapse Expand
ru.vsu.cs.tp.richfamily.CategoryViewModelTest			701 ms
get all categories test	passed		699 ms
empty category list test	passed		2 ms
ru.vsu.cs.tp.richfamily.TemplateViewModelTest			27 ms
get all template test	passed		26 ms
empty template list test	passed		1 ms
ru.vsu.cs.tp.richfamily.WalletViewModelTest			23 ms
empty wallet list test	passed		22 ms
get all wallet test	passed		1 ms

Рисунок 54 - Результат тестирования ViewModel

5 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» (рисунок 55) необходима для того, чтобы просмотреть статистику по количеству пользователей, создавших финансовую операцию.



Рисунок 55 - Воронка конверсии «Создание операции»

Воронка конверсии «Создание операции после создания счета и категории» (рисунок 56) необходима для того, чтобы просмотреть статистику по количеству пользователей, создавших финансовую операцию после создания счета и категории.

Конверсия шагов

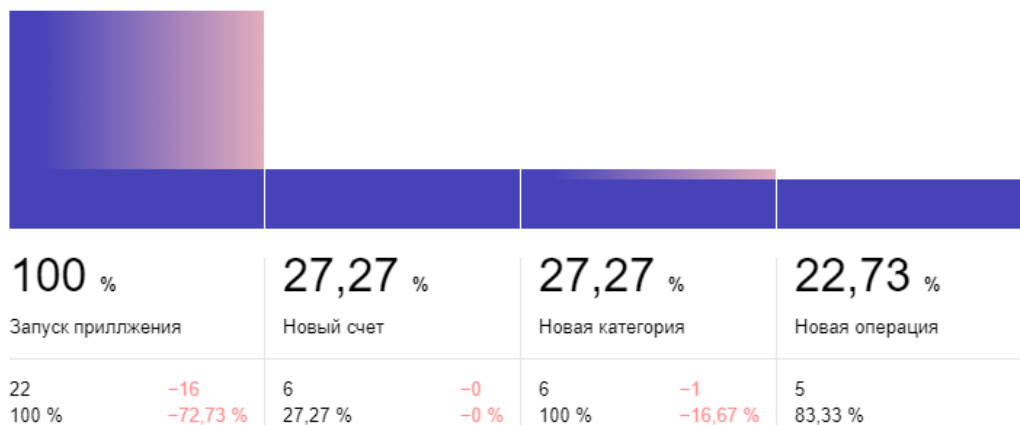


Рисунок 56 - Воронка конверсии «Создание операции после создания счета и категории»

Воронка конверсии «Группы» (рисунок 57) необходима для того, чтобы просмотреть статистику по количеству пользователей, создавших группу, а также пригласивших в нее участников.

Конверсия шагов

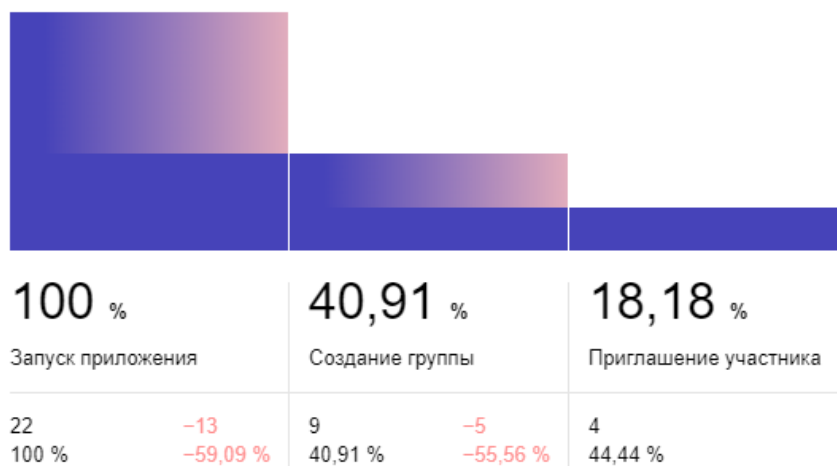


Рисунок 57 - Воронка конверсии «Группы»

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.

Список использованных источников

1. Android Programming: The Big Nerd Ranch Guide (Big Nerd Ranch Guides) / K. Marsicano, B. Gardner, B. Phillips, C Stewart. – New York: Big Nerd Ranch Guides, 2019. – 1036 с.
2. Android Application Development Cookbook: 93 Recipes for Building Winning Apps / L. Wei-Meng. – New York: Wrox, 2013. – 408 с.
3. CoinKeeper [Электронный ресурс]. – Режим доступа: <https://about.coinkeeper.me/>. – Заглавие с экрана. – (Дата обращения: 20.05.2023).
4. Дзен-мани [Электронный ресурс]. – Режим доступа: <https://zenmoney.ru/>. – Заглавие с экрана. – (Дата обращения: 20.05.2023).
5. Python 3.11.3 documentation [Электронный ресурс]. – Режим доступа: <https://docs.python.org/3/>. – Заглавие с экрана. – (Дата обращения: 19.05.2023).
6. Django: The web framework for perfectionists with deadlines [Электронный ресурс]. – Режим доступа: <https://docs.djangoproject.com/en/4.2/>. – Заглавие с экрана. – (Дата обращения: 20.05.2023).
7. PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.
8. The Docker Book: Containerization is the new virtualization Kindle Edition / James Turnbull. - James Turnbull, 2014. – 388 с.
9. Pro Git: Everything you need to know about Git / Scott Chacon, Ben Straub. – 2-е изд. – изд. Apress, 2014. – 458 с.
10. MDN Web Docs Glossary: MVC [Электронный ресурс]. - Режим доступа: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>. – Заглавие с экрана. - (Дата обращения: 20.05.2023).

11. Let`s Encrypt [Электронный ресурс]. – Режим доступа:
<https://letsencrypt.org/ru/> – Заглавие с экрана. – (Дата обращения:
20.05.2023).
12. Certbot [Электронный ресурс]. – Режим доступа:
<https://certbot.eff.org/> – Заглавие с экрана. – (Дата обращения:
20.05.2023).