МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Техническое задание
на разработку мобильного приложения
«Приложение, которое отслеживает траты, составляет бюджеты и предлагает персонализированные стратегии экономии»

Исполнители

_____ Д. С. Огнев

_____ Д. И. Чихирев

_____ И. А. Серванс

_____ В. А. Мартьянов

_____ С. С. Бадиров

Заказчик

_____ В. С. Тарасов

Воронеж 2025

Оглавление

1 Термины, используемые в техническом задании

Термин	Определение термина
Авторизация	Процесс аутентификации пользователя через email/пароль или биометрические данные для доступа к финансовым данным
ИИ-аналитика	Алгоритмы на базе искусственного интеллекта для выявления паттернов расходов и генерации рекомендаций
Пользователь	Лицо, которое использует действующую систему для выполнения конкретной функции
Профиль (в android-приложении)	Персональная запись пользователя, где хранится информация, необходимая для взаимодействия с ресурсом.
Транзакция	Запись о доходе/расходе с атрибутами: сумма, категория, дата, описание (ISO 20022-стандарт)
СУБД	PostgreSQL - объектно-реляционная система управления данными с поддержкой JSON-полей для хранения финансовых операций
Финансовая цель	SMART-объект с параметрами: целевая сумма, срок, текущий прогресс, приоритет
Оффлайн-режим	Функционирование приложения без сети с синхронизацией при восстановлении соединения (CRDT-алгоритмы)
Чат	Средство обмена различной информацией по компьютерной сети в режиме реального времени, а также программное обеспечение, позволяющее организовывать такое общение

Термин	Определение термина
OpenAPI	Спецификация REST API для интеграции с внешними сервисами (версия 3.1.0)
Front-end	Презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты
DI-контейнер	Система внедрения зависимостей (Riverpod) для управления состоянием Flutter-приложения
PSD2	Директива ЕС о платежных услугах (используется как эталон безопасности)
Горизонт планирования	Период для прогнозирования денежного потока (30/90/365 дней)
Back-end	Внутренняя часть сайта или приложения, которая находится на сервере и отвечает за бизнес-логику, обработку данных, и взаимодействие с базами данных или другими внешними системами
Экспорт данных	Генерация отчетов в форматах: PDF (через pdfkit), CSV (RFC 4180), XLSX (ECMA-376)
Финансовая цель	SMART-объект с параметрами: целевая сумма, срок, текущий прогресс, приоритет

2 Общие сведения

2.1 Наименование системы

Полное наименование приложения: «Приложение, которое отслеживает траты, составляет бюджеты и предлагает персонализированные стратегии экономии».

Краткое наименование: «MoneyGuard».

2.2 Наименование исполнителя и заказчика приложения

2.2.1 Наименование заказчика

Заказчик: Старший преподаватель Тарасов Вячеслав Сергеевич. Воронежский Государственный Университет, Факультет компьютерных наук, кафедра Программирования и Информационных Технологий.

2.2.2 Наименование исполнителя

Разработчик: команда «5» группы «10».

Состав команды разработчика:

— Огнев Дмитрий Сергеевич (team lead, backend-разработчик, DevOps инженер);

— Чихирев Даниил Игоревич (тестировщик, pm);

— Серванс Иман Абдмариам Абдалла (frontend-разработчик);

— Мартьянов Владислав Александрович (дизайнер);

— Бадиров Самур Сабриевич (аналитик).

2.3 Перечень документов, на основании которых создается приложение

Данное мобильное приложение будет создаваться на основании следующих документов:

- закона РФ от 07.02.1992 N 2300–1 (ред. от 11.06.2021) "О защите прав потребителей";
- федерального закона "О персональных данных" от 27.07.2006 N 152-ФЗ.

2.4 Состав и содержание работ по созданию приложения

Состав и содержание работ по созданию приложения включают в себя следующие этапы:

- Сбор необходимой информации, постановка целей, задач системы, которые в будущем должны быть реализованы 16.02.25 – 01.03.25;
- Анализ предметной области, анализ конкурентов и построение структуры требований, ведущих к решению поставленных задач и целей 01.03.25 – 16.03.25;
- Построение модели программы, описание спецификаций данных, определение связей между сущностями, разработка модели БД 16.03.25 – 01.04.25;
- Разработка рабочего проекта, состоящего из написания программного кода, отладки и корректировки кода программы 01.04.25 – 16.05.25;
- Проведение тестирования программного обеспечения 16.05.25 – 01.06.25.

2.5 Порядок оформления и предъявления заказчику результатов работ по созданию приложения

Предварительные отчёты по работе будет проводиться во время рубежных аттестаций:

- 1 аттестация (конец марта 2025) – создан репозиторий проекта на GitHub, распределены задачи проекта в таск-трекер Weeek, создан проект с общей логикой системы, предоставлены промежуточные результаты по курсовому проекту и готовое техническое задание;
- 2 аттестация (конец апреля 2025) – написана основополагающая часть программного кода приложения, которая содержит большинство требуемого функционала, реализована БД и ее взаимодействие с сервером, проведена отладка и доработка кода, проведено тестирование по работе системы;
- 3 аттестация (конец мая 2025) – разработан курсовой проект, выполнены завершающие работы по доработке приложения, предоставлена готовая система.

2.6 Цели и назначение создания android-приложения

Цель проекта заключается в создании android-приложения, которое поможет пользователям эффективно управлять личными финансами, контролировать расходы, планировать бюджет и получать персонализированные рекомендации с использованием технологий искусственного интеллекта. Приложение предназначено для упрощения финансового учета, повышения финансовой грамотности и достижения финансовой стабильности как для индивидуальных пользователей, так и для семей.

Android-приложение позволяет решать следующие задачи:

— контроль расходов: автоматический учет и категоризация транзакций для наглядного отображения финансовых потоков;

— Планирование бюджета: установка лимитов по категориям расходов и отслеживание их выполнения;

— ИИ-аналитика: генерация персонализированных рекомендаций по оптимизации бюджета на основе анализа финансовых привычек;

— безопасность: защита финансовых данных пользователей с использованием современных технологий шифрования;

— импорт данных: удобный импорт транзакций из банковских выписок (Excel/CSV) без необходимости ввода учетных данных;

— постановка целей: возможность установки финансовых целей (например, накопление) и отслеживания их выполнения;

— доступность: кроссплатформенность (благодаря Flutter) для использования на различных устройствах.

3 Анализ конкурентов

Финансовые технологии в современном мире помогают пользователям управлять личными финансами более эффективно. Это связано с растущей потребностью в контроле расходов, планировании бюджета и достижении финансовых целей. На рынке мобильных приложений для учета финансов представлено множество решений, среди которых выделяются CoinKeeper и ZenMoney. Оба сервиса предлагают инструменты для анализа месячных трат, но различаются подходом к организации финансового учета.

3.1 CoinKeeper

CoinKeeper — это приложение для учета личных финансов с визуальным интерфейсом в виде кошельков, упрощающим отслеживание доходов и расходов по категориям. Оно ориентировано на удобство и наглядность, предлагая быстрый ввод транзакций и детализированные отчеты.

Преимущества:

— Интуитивно понятный интерфейс с системой кошельков;

— Синхронизация данных между устройствами;

— Автоматическое распознавание трат по банковским выпискам;

— Функция планирования бюджета с контролем лимитов;

— Напоминания о предстоящих платежах и финансовых целях.

— Поддержка мультивалютности.

Недостатки:

— Не все банки поддерживают автоматическую загрузку транзакций;

— Отсутствие прогнозирования расходов на основе истории;

— Ограниченный функционал в бесплатной версии (требуется подписка).

3.2 ZenMoney

ZenMoney — это продвинутое приложение для ведения личного и семейного бюджета, ориентированное на автоматизацию учета финансов. Ключевая особенность — интеграция с банками и автоматическая загрузка транзакций.

Преимущества:

— Автоматическая синхронизация с банковскими картами и счетами (минимальный ручной ввод);

— Гибкая система категорий для детализации расходов;

— Напоминания о платежах и задолженностях;

— Поддержка совместного бюджета для семьи.

Недостатки:

— Многие функции доступны только в платной версии;

— Возможны задержки синхронизации с банками;

— Сложность первоначальной настройки категорий и счетов;

— Интерфейс может показаться перегруженным новым пользователям;

— Ограниченный список поддерживаемых банков.

4 Требования к приложению и программному обеспечению

4.1 Требования к базе данных

База данных будет реализована с помощью PostgreSQL, кэширование будет реализовано с помощью Redis, Shared Preferences и Hive (для оффлайн режима).

4.2 Требования к архитектуре

Приложение должно быть реализовано с применением клиент-серверной архитектуры на основе REST API, обеспечивающей надежное взаимодействие между мобильным клиентом (Flutter) и серверной частью (Java/Spring Boot).

4.2.1 Требования к средствам реализации

Для реализации серверной части сайта будут использоваться следующие средства:

— Язык программирования: Java (высокая производительность, надежность, богатая экосистема);

— Spring Boot (ускоренная разработка, встроенные модули для безопасности, работы с БД и API);

— Spring Security (аутентификация, авторизация, защита от CSRF, JWT-токены);

— ORM Hibernate (для работы с PostgreSQL);

— СУБД PostgreSQL (надежное хранение финансовых данных);

— Docker (позволяет ускорить разработку, тестирование и развертывание приложения).

— Кэширование Redis (ускорение доступа к часто запрашиваемым данным, например, аналитике).

— В качестве AI-модели будет использоваться Qwen 2.5

Для реализации клиентской части приложения будут использоваться:

— Flutter (кроссплатформенность для iOS и Android);

— Управление состоянием Riverpod (гибкость и производительность);

— SharedPreferences для настроек и Hive для офлайн-доступа к транзакциям и целям;

— Графики flutter_charts (визуализация расходов/доходов);

— Уведомления flutter_local_notifications (напоминания о платежах);

— Безопасность flutter_secure_storage (хранение токенов в Keychain/Keystore).

Для ведения документации:

— OpenAPI

4.3 Требования к защите информации

— Аутентификация: OAuth 2.0 + JWT;

— Шифрование: передача данных HTTPS (TLS 1.2+), хранение паролей bcrypt;

— Защита от атак: SQL-инъекции — параметризованные запросы (Hibernate), CSRF — токены в Spring Security;

— кэширование с помощью Redis, Shared Preferences и Hive.

4.4 Требования по патентной части

Приложение должно быть разработано в соответствии с законодательством об авторских правах и лицензиях (MIT для Flutter, Apache 2.0 для Spring). Использование сторонних API (банки, ИИ) требует соблюдения их условий. Ответственность за нарушения несет исполнитель.

4.5 Требования к интеграциям

Приложение должно поддерживать интеграцию с:

— Банками: Импорт транзакций через CSV/Excel (парсинг на стороне сервера).

— Уведомления: Firebase Cloud Messaging (FCM) для push-сообщений.

5 Функциональные требования

Приложение должно поддерживать функционал для различных пользователей:

— неавторизованный пользователь;

— авторизованный пользователь;

5.1 Функциональные требования неавторизованного пользователя

Перечень функций:

— авторизация в системе;

— регистрация в системе;

— просмотр ознакомительной информации.

5.2 Функциональные требования авторизованного пользователя

Данный функционал доступен авторизованным ролям.

Перечень функций:

— Выход из системы на всех устройствах;

— просмотр страницы профиля;

— редактирование данных профиля;

— просмотр страницы, содержащей информацию о подписке и тарифах;

— оформление/отмена подписки;

— Добавление/редактирование/удаление транзакций (вручную или импорт из CSV/Excel);

— Создание категорий расходов и доходов;

— Установка лимитов по категориям;

— Просмотр отчетов;

— Получение ИИ-рекомендаций по оптимизации бюджета;

— Создание финансовых целей;

— Отслеживание прогресса.

6 Нефункциональные требования

Android-приложение должно выполнять следующие нефункциональные требования:

- время отклика API: Основные операции (добавление транзакции, просмотр баланса) — ≤ 300 мс, сложные запросы (аналитика за месяц) — ≤ 1 сек;
- обработка пиковых нагрузок: Поддержка ≥ 10 000 одновременных пользователей (с горизонтальным масштабированием серверов);
- оптимизация для слабых сетей: Работа в офлайн-режиме (кэширование данных на устройстве через Hive/SQLite), минимальный трафик для синхронизации (только дельты изменений);
- защита данных: шифрование всех передаваемых данных (HTTPS/TLS 1.3), хранение паролей в виде хешей (bcrypt). Использование JWT-токенов с коротким временем жизни (30 мин);
- защита от атак: SQL-инъекции — параметризованные запросы (Hibernate), DDoS — ограничение запросов (Rate Limiting через Spring Cloud Gateway);
- надежность: аварийное восстановление — резервное копирование БД каждые 24 часа + snapshots;
- android: версии 10+ (API 29);
- адаптивный интерфейс: корректное отображение при масштабировании шрифтов (от 90% до 125%);
- интуитивность: добавление транзакции — ≤ 3 тапов;
- документация и поддержка: для пользователей — Видео-инструкции для сложных функций (например, импорт из банка), FAQ. Для разработчиков: API-документация (OpenAPI 3.0).

6.1 Макет приложения

6.1.1 Макет стартового экрана

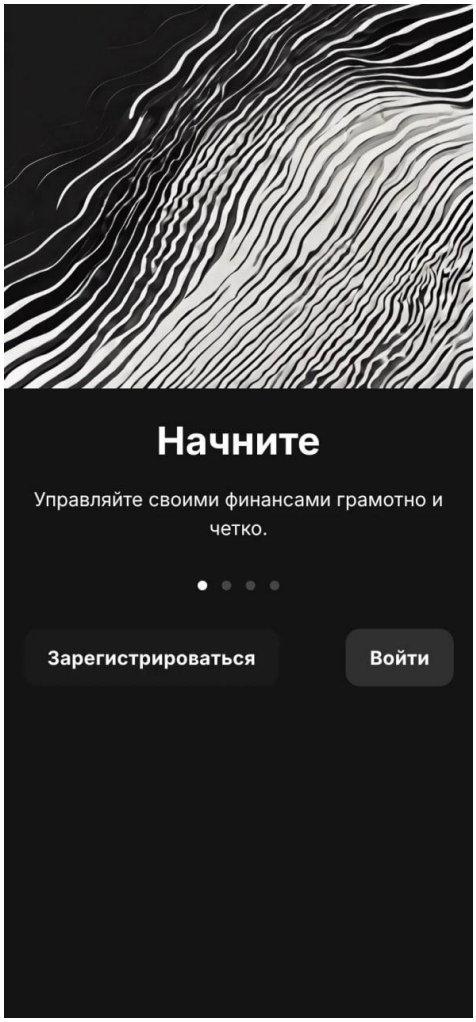

Рисунок 1 - Макет стартового экрана

На изображении представлен стартовый экран приложения MoneyGuard с минималистичным дизайном. Экран выполнен в тёмной цветовой гамме с контрастными текстовыми элементами. Отсутствуют поля для ввода данных - они появятся после нажатия одной из кнопок. Дизайн ориентирован на быстрое принятие решения о входе или регистрации, и блок с иконками/превью основных функций (4 элемента).

6.1.2 Макет информационного экрана

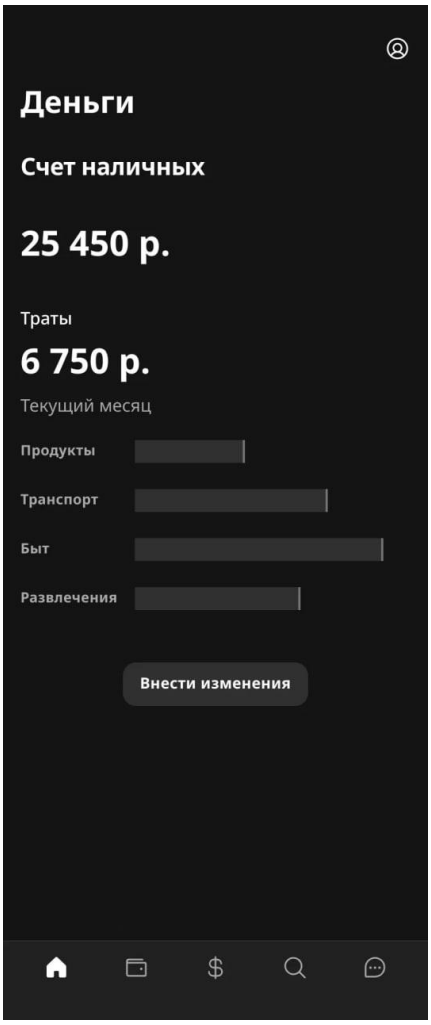

Рисунок 2 - Макет информационного экрана

На изображении представлен главный финансовый дашборд приложения с детализированной информацией о состоянии счета и расходах.

6.1.3 Макет экрана добавления транзакций

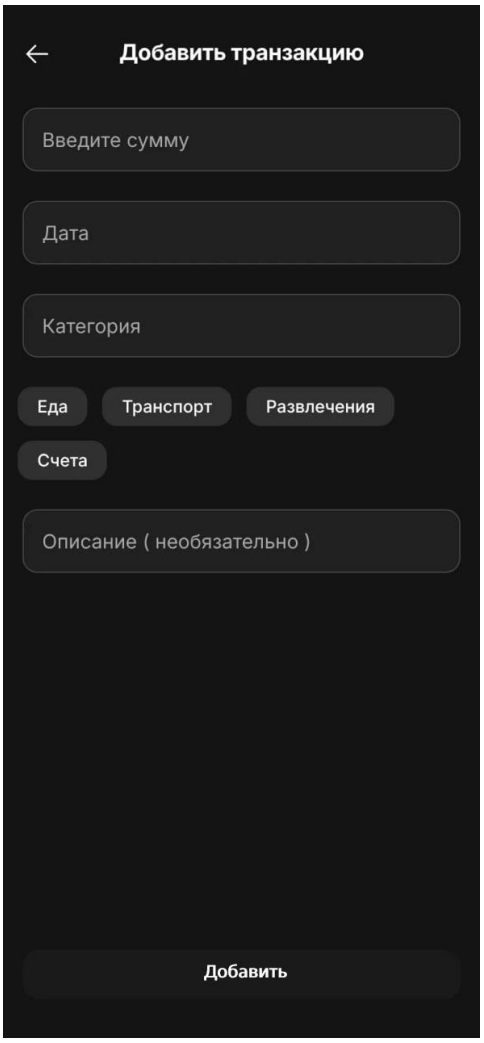

Рисунок 3 - Макет экрана добавления транзакций

Экран добавления транзакции с полями для ввода суммы, даты, категории (Еда, Транспорт, Развлечения, Счета) и необязательного описания. Без обязательных заполненных полей добавление транзакции невозможно.

6.1.4 Макет экрана чата с ИИ

Рисунок 4 - Макет экрана чата с ИИ

На данном экране пользователь может получить персонализированные рекомендации от ИИ, приложить фото чека. Обычное общение на другие темы кроме финансов не представляется возможным.

7 Планы на дальнейшее развитие проекта

После запуска MoneyGuard планируется расширение функционала за счёт внедрения семейных аккаунтов, кастомных категорий и интеграции с Open Banking API для автоматического импорта транзакций. ИИ-ассистент

получит предиктивную аналитику. В планах локализация на английский язык, поддержка мультивалютности, партнёрства с банками, а также оптимизация для офлайн-режима. Ключевые этапы: Q2 2025 - семейные аккаунты и чат-бот, Q4 2025 - Open Banking, 2026 - локализация и предиктивная аналитика, с целью создания комплексной финансовой экосистемы.

8 Источники разработки

1. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы;
2. CoinKeeper (https://about.coinkeeper.me/eng);
3. ZenMoney (https://zenmoney.app).

Приложение

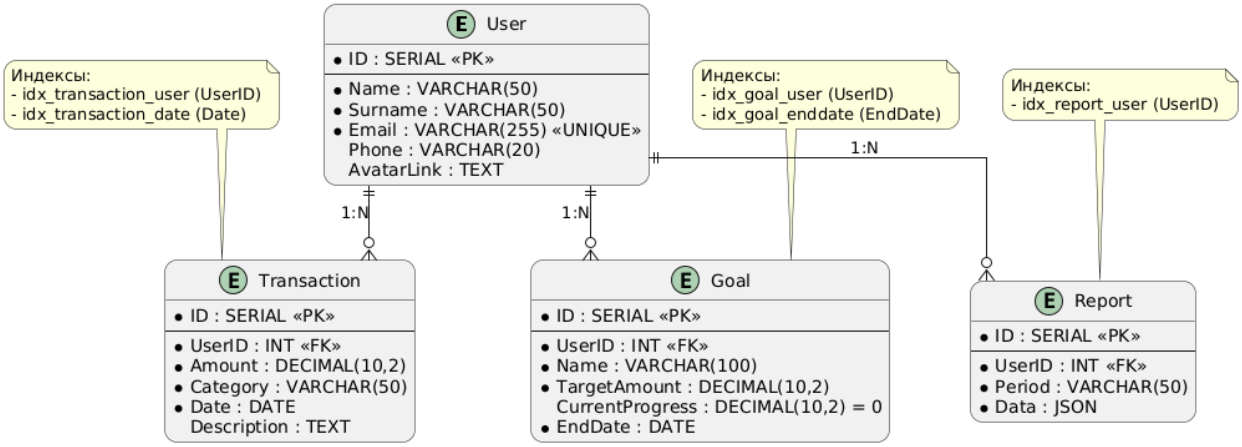

Рисунок 5 - ER-диаграмма

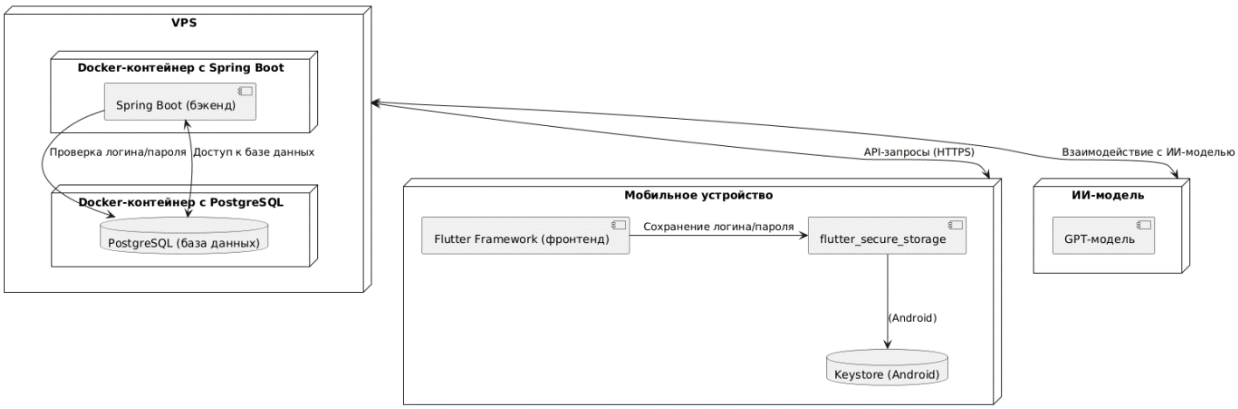

Рисунок 6 - Диаграмма развертывания

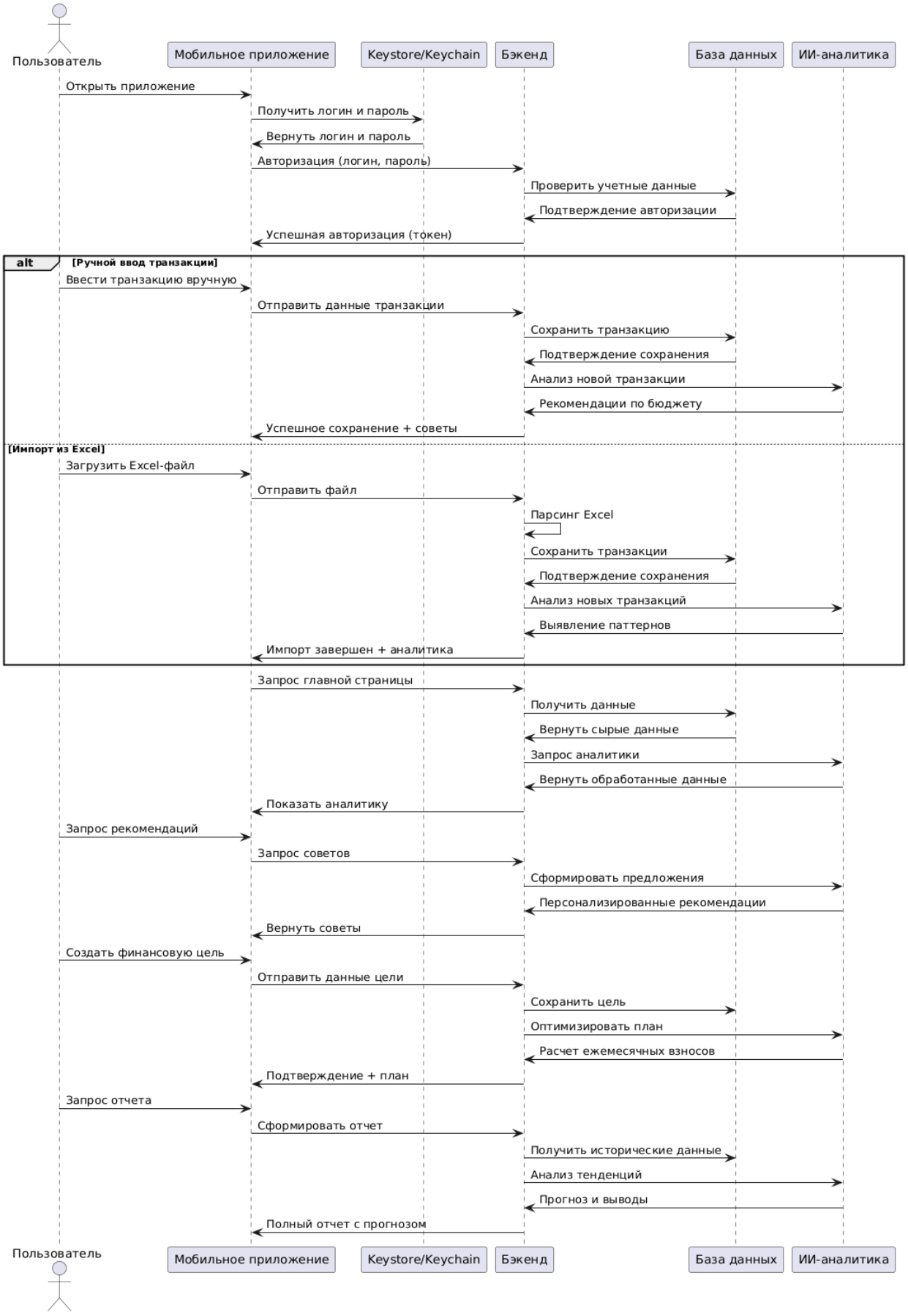

Рисунок 7 - Диаграмма последовательности

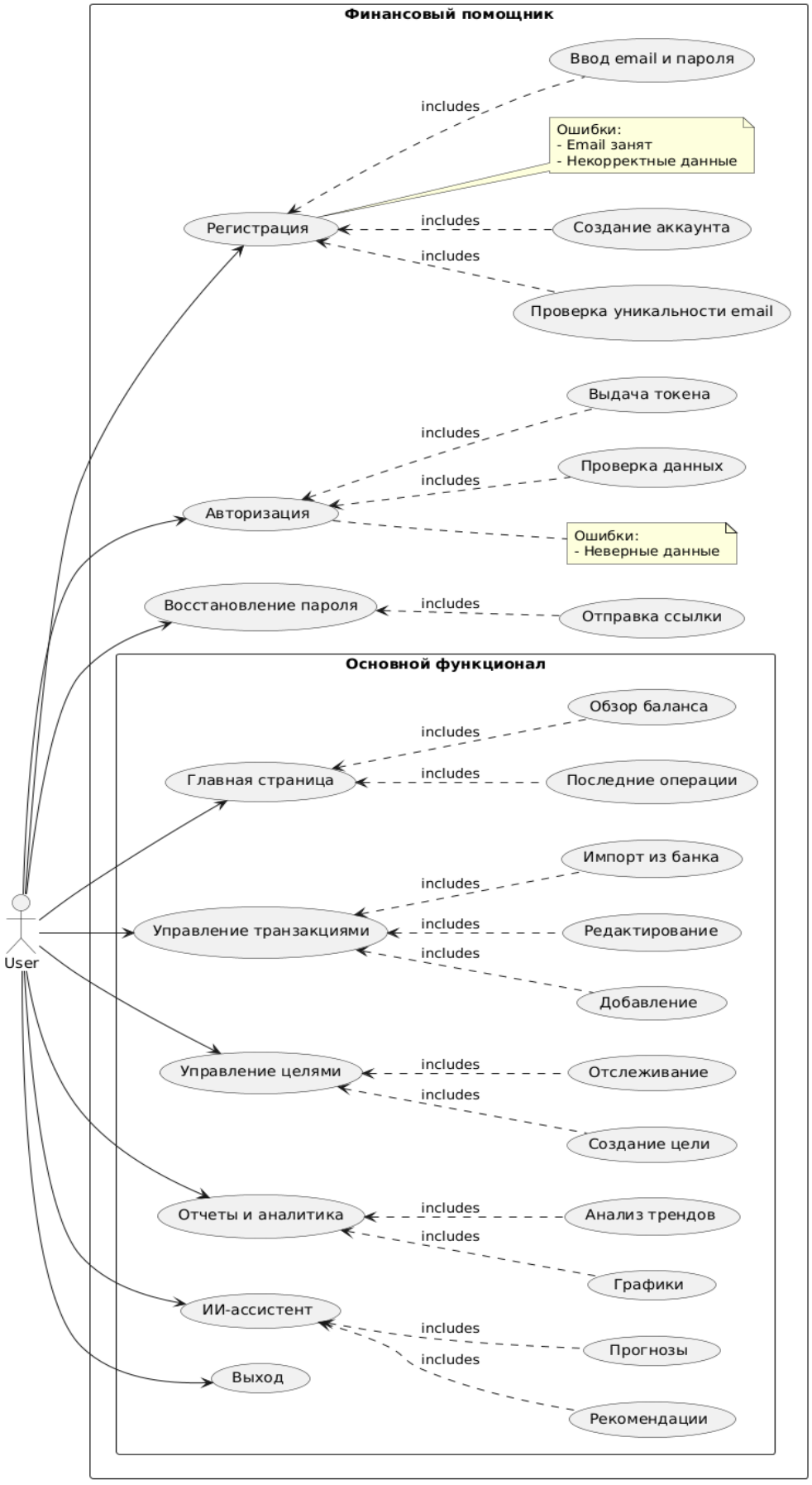

Рисунок 8 - Диаграмма прецендентов

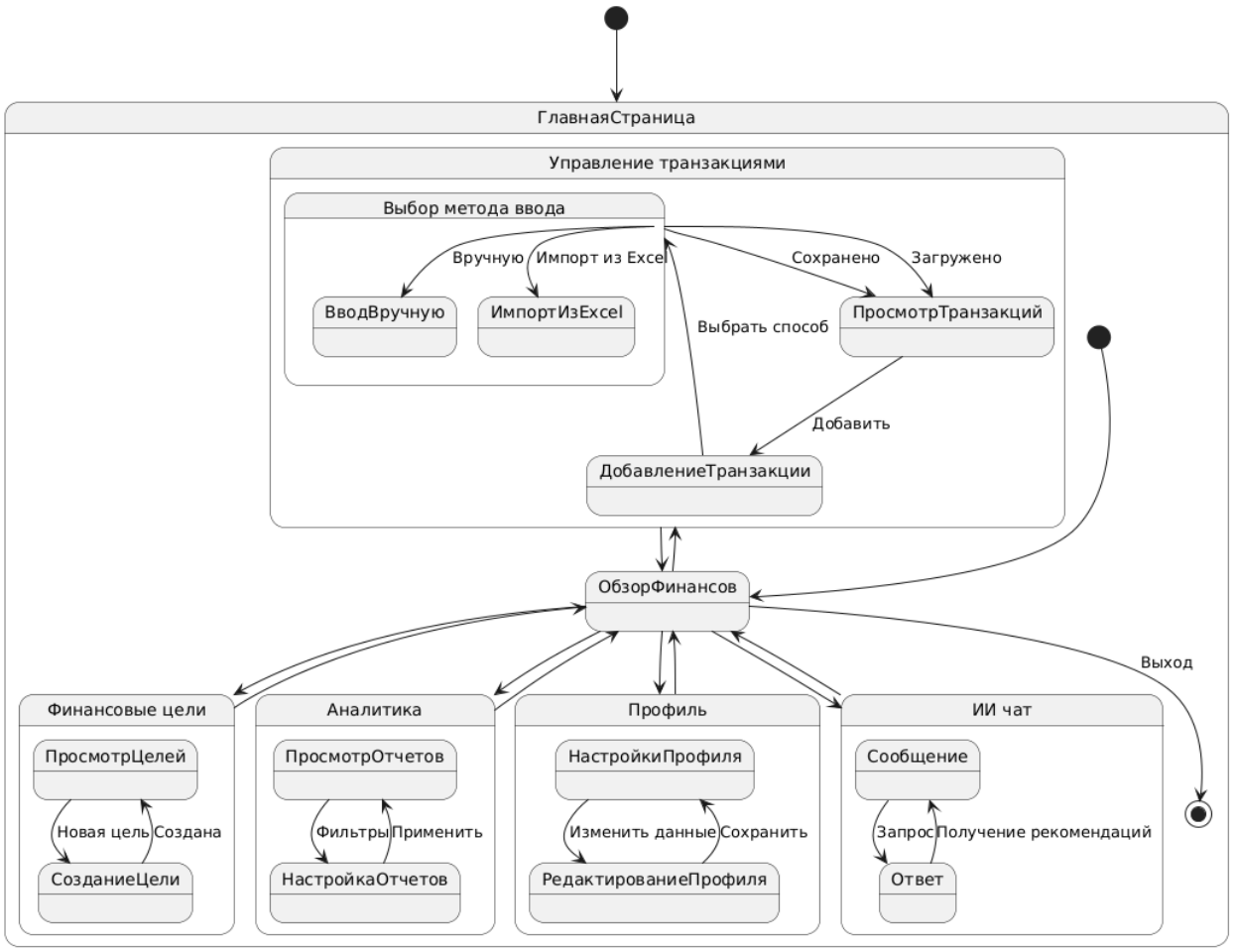

Рисунок 9 - Диаграмма активности