

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет Компьютерных наук

Кафедра программирования и информационных технологий

Мобильное приложение
«Оценка качества продукции “QWality”»

Курсовая работа

Направление: 09.03.02. Информационные системы и технологии

Зав. Кафедрой _____ д. ф.-м. н, доцент С.Д. Махортов

Руководитель _____ ст. преподаватель В.С. Тарасов

Руководитель практики _____ ассистент В.А. Ушаков

Обучающийся _____ Р.Ю. Перцев, 3 курс, д/о

Обучающийся _____ Д.С. Сушкова, 3 курс, д/о

Обучающийся _____ В.А. Баранов, 3 курс, д/о

Обучающийся _____ В.В. Лихачев, 3 курс, д/о

Обучающийся _____ Д.В. Фролов, 3 курс, д/о

Обучающийся _____ М.В. Бен Амор, 3 курс, д/о

Воронеж 2025

Содержание

Определения, обозначения, сокращения	3
1. Общие положения	4
1.1. Название приложения	4
1.2. Разработчики и заказчик	4
1.3. Перечень документов, на основании которых создаётся приложение .	5
1.4. Порядок оформления и предъявления заказчику результатов работ по созданию приложения	5
2. Постановка задачи.	6
2.1. Цели.....	6
2.2. Задачи.....	7
2.3. Функциональные требования:.....	7
2.4. Нефункциональные требования:.....	7
2.5. Требования к архитектуре	8
3. Анализ предметной области и конкурентов	9
3.1. Анализ предметной области.....	10
3.2. Задачи, решаемые в процессе разработки.....	10
3.3. Анализ предметной области	11
4. Реализация	13
4.1. Frontend Web	13
4.2. Frontend Mobile	18
4.3. Machine Learning	20
4.4. Backend	22
5. Диаграммы, иллюстрирующие работу системы.....	28
6. Дизайн приложения	46
Приложение 1. Примечания	61

Определения, обозначения, сокращения

В настоящем техническом задании применяют следующие термины с соответствующими определениями:

Электролюминесцентный спектр – метод визуализации, применяемый для выявления дефектов в структуре солнечных панелей.

Журнал логов (логирование) – автоматическая запись действий пользователей и работы системы для аудита и отладки.

CI/CD (Continuous Integration/Continuous Deployment) – методология автоматизированной сборки, тестирования и развертывания кода (реализована через GitHub Actions и Docker).

API (Application Programming Interface) – интерфейс программирования приложений, позволяющий взаимодействовать между клиентской и серверной частями системы.

UI Kit – набор дизайн-компонентов интерфейса в Figma.

GPU (Graphics Processing Unit) – графический процессор, используемый для ускорения обработки изображений.

CUDA – технология NVIDIA для параллельных вычислений на GPU.

RAM (Random Access Memory) – оперативная память сервера.

VRAM (Video RAM) – видеопамять графической карты.

FAQ (Frequently Asked Questions) – раздел с ответами на часто задаваемые вопросы.

1. Общие положения

1.1. Название приложения

Полное наименование приложения: «Оценка качества продукции “QWality”».

Условное обозначение приложения: «QWality».

1.2. Разработчики и заказчик

Разработчик: «2» команда группы «4»

Состав команды разработчика:

- студент Перцев Роман Юрьевич, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Технологий Обработки и Защиты Информации;
- студент Сушкова Дарья Сергеевна, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Технологий Обработки и Защиты Информации;
- студент Баранов Виталий Алексеевич, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Технологий Обработки и Защиты Информации;
- студент Лихачев Валерий Валерьевич, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Технологий Обработки и Защиты Информации;
- студент Фролов Данила Валерьевич, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Технологий Обработки и Защиты Информации;
- студент Бен Амор Мохамед Вассим, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Технологий Обработки и Защиты Информации;

Заказчик: Старший Преподаватель Тарасов Вячеслав Сергеевич, Воронежский Государственный Университет, Факультет Компьютерных Наук, кафедра Программирования и Информационных Технологий.

1.3. Перечень документов, на основании которых создаётся приложение

Приложение создается на основе:

- Федерального закона «Об информации, информационных технологиях и о защите информации» от 27.07.2006 г. № 149-ФЗ;
- Федерального закона "О персональных данных" от 27.07.2006 N 152-ФЗ;
- Пользовательского соглашения, регулирующего отношения между пользователями и разработчиками приложения (см. Приложение 1);
- Политики конфиденциальности, определяющей порядок сбора, обработки и хранения персональных данных пользователей (см. Приложение 1);
- Технического задания, составленного в соответствии с ГОСТ 34.602-89.

1.4. Порядок оформления и предъявления заказчику результатов работ по созданию приложения

Предварительные отчёты по работе будут проводиться во время рубежных аттестаций:

1 аттестация (конец марта 2025) – предоставлены ссылки на репозиторий проекта на GitHub и проект в таск-менеджере Jira. Созданы промежуточные результаты курсового проекта, включающие готовое техническое задание (ТЗ), определённые функциональные и нефункциональные требования, пользовательские сценарии (User Stories) и список ключевых функциональных блоков системы. Разработана предварительная архитектура, включающая UML-диаграммы, ER-диаграмму базы данных, схему API и предварительно

выбранный технологический стек. Подготовлены начальные дизайн-макеты в Figma и настроена структура проекта в Git-репозитории.

2 аттестация (конец апреля 2025) – разработан программный код, охватывающий большую часть запланированной функциональности, включая бэкенд с основными API-методами (CRUD-операции), настроенную базу данных с тестовыми данными и её интеграцию с сервером. Создана и протестирована модель машинного обучения, выполнена предобработка данных. Проведены отладка и оптимизация кода, реализованы функции авторизации пользователей. Разработан минимальный пользовательский интерфейс для работы с системой. Настроен процесс CI/CD с автоматическим деплоем. Подготовлена тестовая документация с предварительными результатами тестирования.

3 аттестация (конец мая 2025) – представлен курсовой проект, включающий полностью функциональное приложение с интегрированными бэкендом, фронтендом и моделью машинного обучения. Завершена разработка всех основных функций, включая ролевую авторизацию и взаимодействие через API. Полностью задокументирован API с использованием Swagger/OpenAPI. Проведены функциональное и интеграционное тестирование, подготовлены отчёты с результатами тестирования. Реализована система сбора аналитики.

Результаты работы должны быть представлены в электронном виде в формате PDF и размещены на GitHub.

2. Постановка задачи.

2.1. Цели

- Автоматизация контроля качества – разработка приложения для мониторинга и анализа качества продукции.
- Повышение прозрачности – хранение и анализ данных о проверках качества.

— Полный доступ к информации

2.2. Задачи

- Разработка мобильного приложения, которое будет предоставлять доступ к информации о браке (процентное соотношение брака к нормальному товару, отображение участков производства с наивысшим количеством брака, доступ к фотографиям с бракованным товаром).
- Создание системы отчетности для мониторинга качества продукции.
- Обеспечение ролевого доступа для разных категорий пользователей.
- Разработка искусственного интеллекта, который будет проводить оценку качества товара.
- Разработка веб приложения.

2.3. Функциональные требования:

- Распознавание дефектов по классам.
- Генерация отчетов по запросу.
- Доступ к разным функциям в зависимости от роли.
- Интеграция с базами данных для хранения результатов анализа.
- Экспорт отчетов в форматы PDF, CSV.
- Поддержка форматов изображений JPEG, PNG
- Удаление истории логов.
- Логирование всех решений ИИ.

2.4. Нефункциональные требования:

- Производительность – высокая скорость обработки изображений за счёт эффективного алгоритма.
- Доступность системы - 99,5% это означает, что система не может простаивать более 3,65 часов в месяц).

- Работа в условиях производственного потока: не менее 2 панелей в минуту при условии, что панели состоят из 60 ячеек.
- Поддержка пиковых нагрузок: до 5 панелей в минуту, масштабируемая через балансировку нагрузки, при условии, что панели состоят из 60 ячеек.
- Совместимость - поддерживает работу в популярных браузерах: Google Chrome, Mozilla Firefox, Yandex.
- Энергоэффективность - обработка изображений выполняется на сервере, а не на мобильном устройстве.
- Поддержка развертывания в облаке и на локальных серверах.
- Обеспечение безопасности данных и контроль доступа.
- Дизайн, соответствующий заявленному набору для пользовательского интерфейса.

2.5. Требования к архитектуре

- Бекенд - python, Flask
- ИИ - python, PyTorch
- Android - React Native
- CI/CD - github actions
- Контейнеризация – Docker
- Объектное хранилище - MinIO
- СУБД – postgresSQL

3. Анализ предметной области и конкурентов

Цель: Изучить особенности предметной области (контроль качества солнечных панелей) и существующие аналоги для определения уникальных особенностей приложения «QWality».

Подзадачи:

— Сбор информации о предметной области:

Изучение технологии производства кристаллических кремниевых солнечных панелей (монокристаллических и поликристаллических).

Анализ типов дефектов (микротрещины, загрязнения, нарушения структуры элементов) и их влияния на эффективность панелей.

Исследование электролюминесцентного метода визуализации, используемого для получения снимков панелей, включая требования к качеству изображений (разрешение 200x200–1920x1080, форматы JPEG/PNG, отсутствие размытия и засветки).

Изучение производственных процессов, включая скорость конвейера (не менее 2 панелей в минуту, до 5 при пиковых нагрузках) и требования к освещению.

— Анализ конкурентов:

Исследование существующих решений, таких как SolarEye, PanelCheck и SunInspect, с акцентом на их функциональность, интерфейс, производительность и ограничения.

Сравнение аналогов по критериям: поддержка ИИ, ролевой доступ, форматы изображений, генерация отчетов, наличие мобильной версии, логирование.

Формирование списка преимуществ «QWality» (например, ролевой доступ, высокая производительность, автоматизация анализа).

— Определение потребностей пользователей:

Выявление целевых групп пользователей: инженеры по качеству (рядовые пользователи), модераторы (управление камерами), администраторы (управление системой), владельцы (полный доступ).

Составление user stories для каждого типа пользователей (например, «Как инженер, я хочу видеть статистику дефектов, чтобы выявить проблемные участки производства»).

Формирование требований к интерфейсу (единый стиль, адаптивность, поддержка популярных браузеров) и функционалу (анализ изображений, отчеты, настройки).

3.1. Анализ предметной области

Солнечные панели, особенно кристаллические кремниевые (моно- и поликристаллические), требуют строгого контроля качества из-за возможных дефектов (микротрещины, загрязнения, нарушения структуры).

Электролюминесцентные снимки позволяют выявлять такие дефекты с высокой точностью. Приложение «QWality» использует ИИ для автоматизации анализа, что сокращает время проверки и повышает точность.

3.2. Задачи, решаемые в процессе разработки

- Анализ предметной области и конкурентов.
- Проектирование базы данных (ER-диаграмма).
- Разработка серверной части (Flask, PyTorch).
- Разработка клиентской части (React Native).
- Реализация ИИ для анализа изображений.

- Интеграция API и тестирование.
- Развертывание на локальных серверах.

3.3. Анализ предметной области

- SolarEye:

Описание: Приложение для анализа солнечных панелей с использованием ИИ.

Плюсы: Поддержка электролюминесцентных снимков, интеграция с облаком.

Минусы: Нет ролевого доступа, ограниченная аналитика, высокая стоимость подписки.

- PanelCheck:

Описание: Система для ручного анализа изображений панелей.

Плюсы: Простота использования, поддержка JPEG/PNG.

Минусы: Отсутствие автоматизации, нет мобильной версии.

- SunInspect:

Описание: Программа для мониторинга качества панелей.

Плюсы: Генерация отчетов в PDF.

Минусы: Нет реального времени, ограниченная поддержка форматов.

Характеристика	SolarEye	PanelCheck	SunInspect	QWality
ИИ для анализа	Да	Нет	Частично	Да
Ролевой доступ	Нет	Нет	Нет	Да
Форматы JPEG/PNG	Да	Да	Нет	Да

Генерация отчетов	Нет	Нет	Да	Да
Мобильная версия	Да	Нет	Да	Да
Логирование действий	Нет	Нет	Нет	Да

4. Реализация

4.1. Frontend Web

Общая структура веб-приложения:

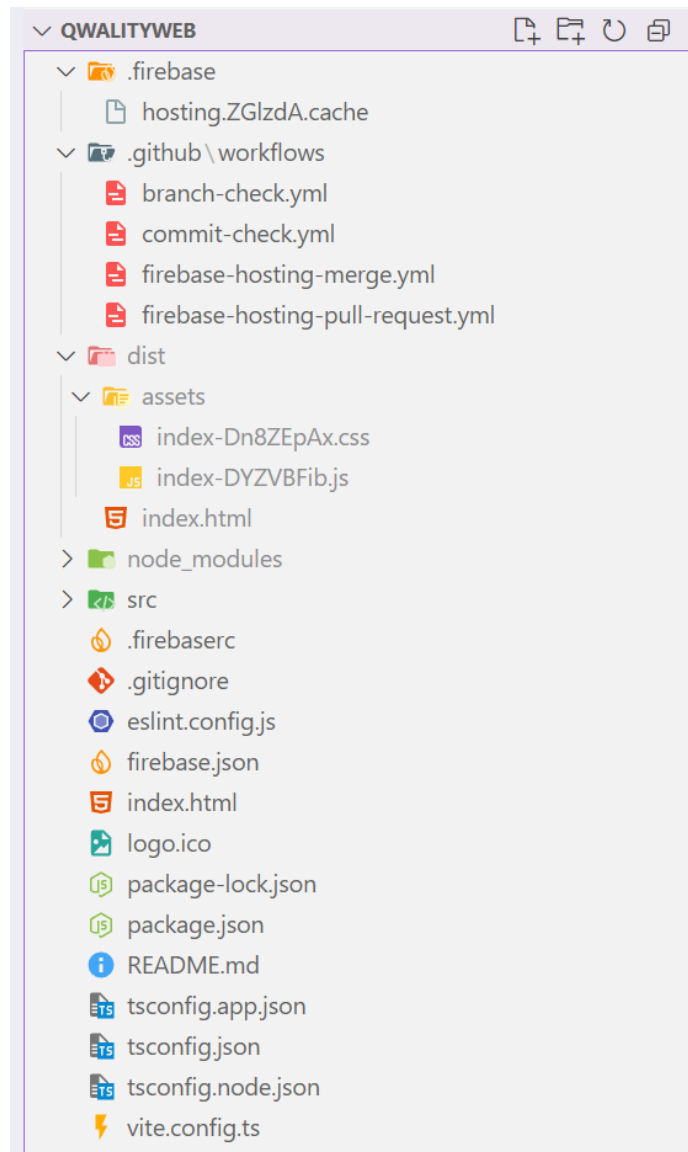


Рисунок 1 - Общая структура веб-приложения

Приложение QWalityWeb представляет собой веб-приложение, разработанное с использованием сборщика Vite и Firebase для хостинга и бэкенда.

Папка .firebase нужна для хранения конфигурации и кэша Firebase, чтобы связать приложение с облачными сервисами (например, хостингом) и ускорить деплой.

`.github/workflows` – папка с рабочими процессами GitHub Actions, используемыми для автоматизации задач, таких как проверка названий коммитов и веток или деплой приложения на Firebase Hosting при слиянии в основную ветку разработки и создании пул-реквестов.

Папка `dist` содержит собранные файлы приложения, готовые для деплоя. Она содержит HTML-файл, служащий точкой входа в приложение после сборки.

`node_modules` – папка с зависимостями проекта, необходимыми для его работы.

Файл `firebase.json` – локальная конфигурация Firebase для разработки.

`.gitignore` – файл, указывающий Git, какие файлы и папки игнорировать при коммите.

`eslint.config.js` – конфигурация ESLint для проверки и поддержания качества кода на TypeScript.

`firebase.json` – основной файл конфигурации Firebase, определяющий настройки хостинга.

`package-lock.json` и `package.json` – файлы управления зависимостями, содержащие список библиотек и скрипты для установки, сборки и запуска.

`tsconfig.app.json`, `tsconfig.json`, `tsconfig.node.json` – файлы конфигурации TypeScript.

`vite.config.ts` – основной файл конфигурации Vite, задающий настройки сборки.

Папка `src` – это основная папка с исходным кодом приложения, структура этой папки:

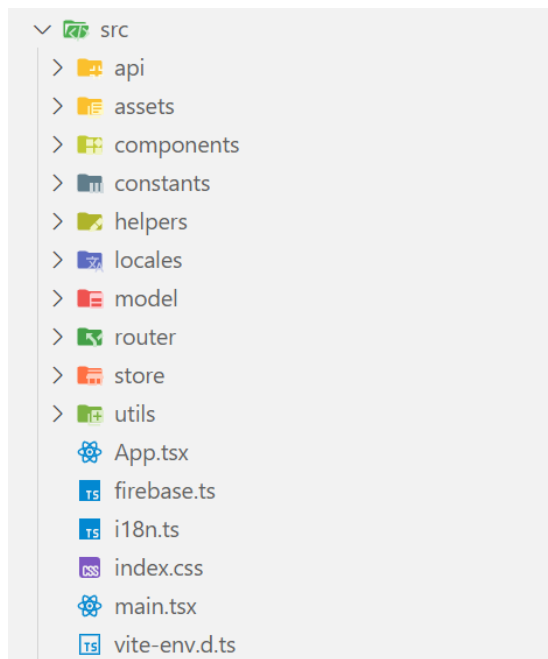


Рисунок 2 - Структура папки src

Основные папки и файлы в папке src:

— Папка `api` — содержит функции, которые отвечают за взаимодействие с внешними API. Например, здесь могут реализованы запросы для получения списка камер клиента, обновления JWT-токенов и т.д.

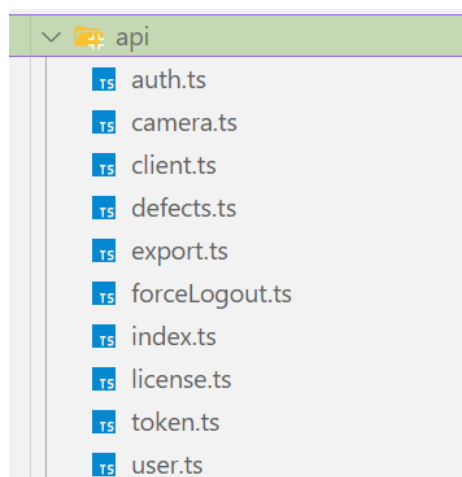


Рисунок 3 - Структура папки api

- Assets – эта папка предназначена для хранения статических ресурсов, таких как svg иконки приложения.
- Папка components содержит все основные React-компоненты приложения. Так как в проекте используется Atomic паттерн, то компоненты разделены на атомы (самые мелкие компоненты приложения, например, кнопки, инпуты), молекулы (компоненты состоящие из нескольких атомов, например дефект или камера на главной странице), организмы (еще более составные и крупные компоненты, например формы, модальные окна), страницы и шаблоны страниц для предотвращения дублирования (например, шаблон страницы с боковым меню).

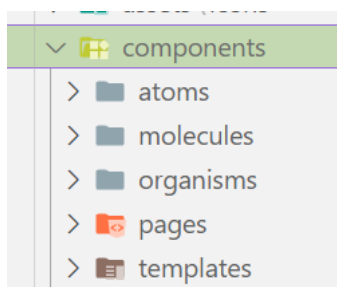


Рисунок 4 - Структура папки components

- В папке constants хранятся константы, которые используются в разных частях проекта, например названия шрифтов, тем, ролей и т.п.

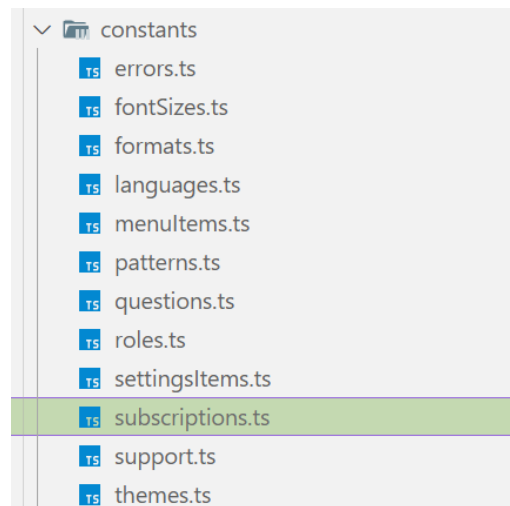


Рисунок 5 - Структура папки constants

- В папке `helpers` находятся вспомогательные функции, которые помогают в работе приложения. Например, функции форматирования дат, отображения попап-элементов (`toast`) и др.
- Папка `locales` отвечает за локализацию приложения. В ней хранятся JSON-файлы переводов на разные языки (русский, английский и французский).
- В папке `model` хранится описание данных (основные поля и их типы), с которыми работает приложение, в виде интерфейсов.
- В папке `router` находится конфигурация маршрутизации приложения – файлы, определяющие маршруты (роуты) приложения, их параметры и соответствующие компоненты страниц.
- `Store` – эта папка отвечает за управление состоянием приложения. Здесь находятся файлы с определением хранилищ (`store`), где создаются состояния и методы их изменения с помощью `Zustand`. Например, `useCamerasStore` отвечает за хранение и обновление списка камер.

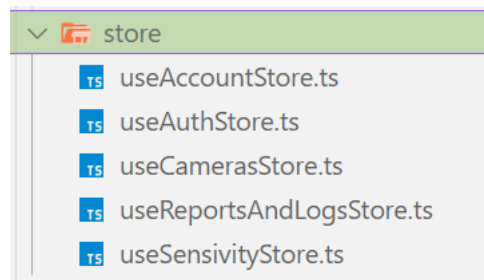


Рисунок 6 - Структура папки store

- В папке `utils` содержатся файлы-конвертеры для приведения данных, полученных путем запроса на бэкенд, к заданным в приложении интерфейсам для удобства дальнейшей работы с ними.
- Файл `App.tsx` – это главный компонент приложения, который выступает как корневой. Здесь задается основная структура приложения, подключаются глобальные провайдеры, а также рендерятся основные компоненты или маршруты.
- `i18n.ts` отвечает за настройку языка в приложении. Здесь задаются поддерживаемые языки, подключаются файлы переводов из папки `locales` и настраивается язык по умолчанию.

4.2. Frontend Mobile

Структура мобильного приложения, разработанного с использованием React Native и Expo (набор инструментов для удобной разработки мобильных приложений с использованием React Native), достаточно похожа на структуру веб-приложения, так как имеет папки и файлы с такими же названиями и назначением:

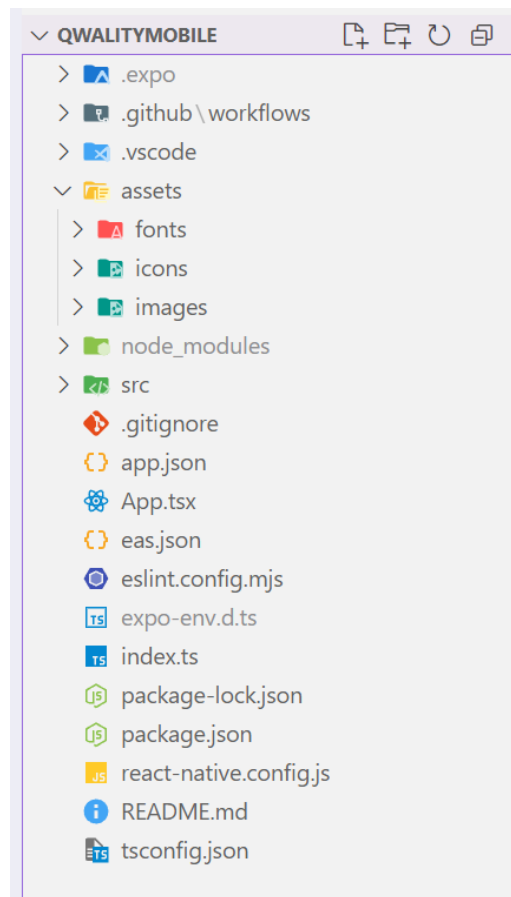


Рисунок 7 - Структура мобильного приложения

Здесь также добавляются:

- Папка `.expo`, которая автоматически создается Ехро для хранения временных файлов и конфигураций, связанных с процессом разработки.
- В папке `assets` со статическими данными добавлена папка `fonts`, содержащая пользовательские шрифты приложения.
- Файл `app.json` – основной конфигурационный файл для Ехро. Здесь задаются метаданные приложения, такие как название, иконка, версия, поддерживаемые ориентации экрана и другие параметры, необходимые для сборки приложения.
- Файл `eas.json` – конфигурационный файл для Ехро Application Services (EAS). Здесь настраиваются процессы сборки и обновлений

приложения через EAS Build и EAS Update. Здесь можно указать разные профили сборки (development, production, preview).
— react-native.config.js – файл конфигурации для React Native.

В папке src так же, как и в веб-приложении, есть папка api для работы с запросами, components, содержащая atomic-компоненты (атомы, молекулы и др.), constants для хранения неизменяемых данных приложения (названия тем, роли и т.д.), helpers – вспомогательные функции приложения, hooks – папка с основными хранилищами и кастомными хуками, locales для хранения словарей переводов, model для описания основных интерфейсов объектов, navigation для настройки роутинга приложения и utils для преобразования данных.

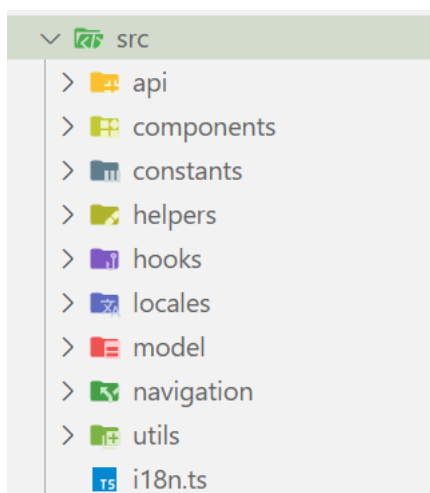


Рисунок 8 - Структура папки src в мобильном приложении

4.3. Machine Learning

Искусственный интеллект для приложения был реализован в данной структуре:

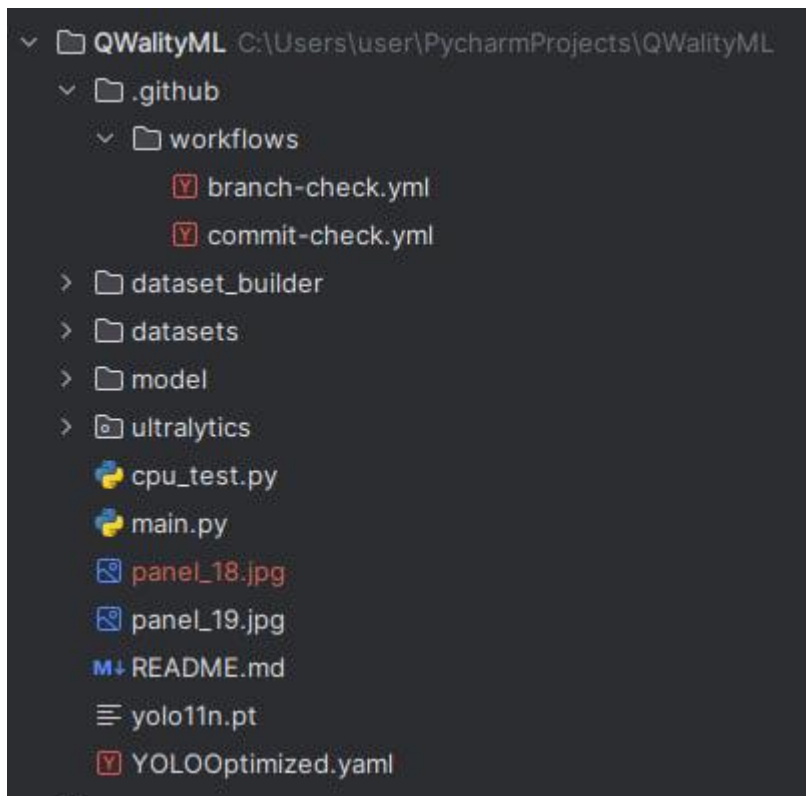


Рисунок 9 - Структура ML-репозитория

- dataset_builder - был взят датасет ячеек, из которых были собраны полноценные солнечные панели размером 60 ячеек в различных конфигурациях и с небольшими искажениями для повышения точности модели
- datasets - содержит размеченный датасет на котором производилось обучение модели, датасет содержит в себе 10 000 изображений частично дефектных солнечных панелей
- ultralytics - оригинальная библиотека для модели YOLO, в которой была заменена backbone архитектура, оптимизировав процесс обучения и инференса конкретно под нашу задачу
- main.py - скрипт обучения получившейся модели
- cpu_test.py - скрипт для сбора метрик при инференсе модели

4.4. Backend

Общая структура Backend-приложения:

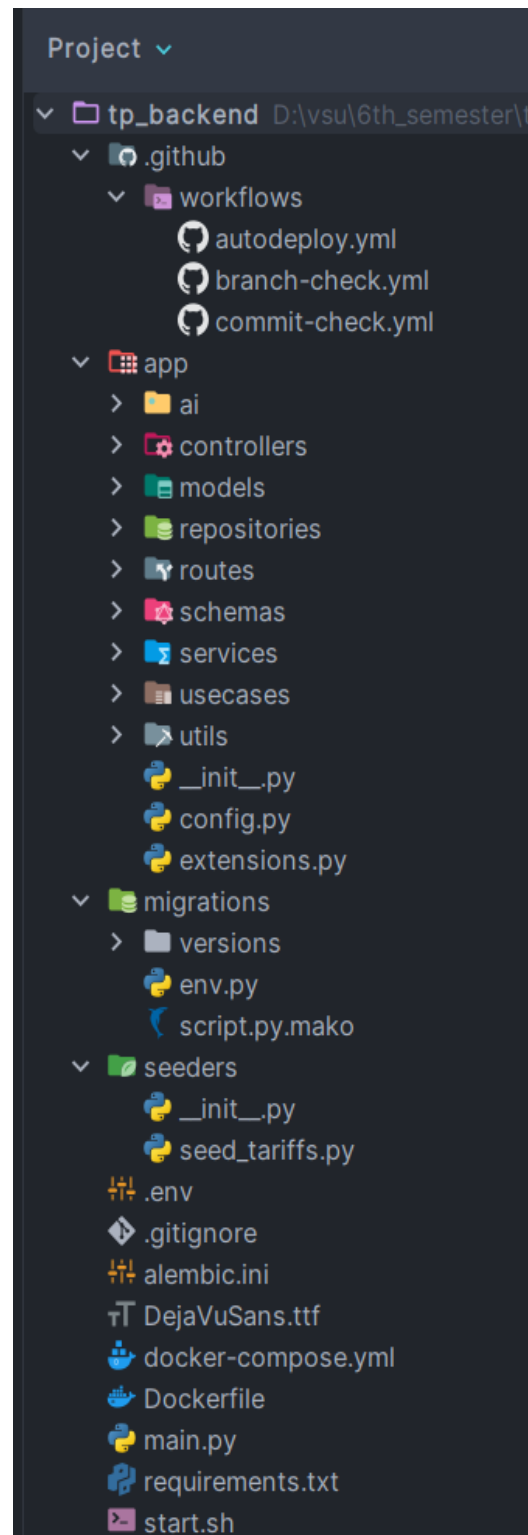


Рисунок 10 - Общая структура backend приложения

QwalityBackend представляет собой серверное приложение, разработанное с использованием фреймворка Flask, PostgreSQL + SQLAlchemy.

Директория `.github/workflows` – папка с рабочими процессами GitHub Actions, используемыми для автоматизации задач, таких как проверка названий коммитов и веток или деплой приложения на удалённый сервер при слиянии в основную ветку разработки и создании пул-реквестов.

Пакет `app` содержит в себе множество других подпакетов, а также конфигурацию python-приложения (`config.py`) и файл с объектами, расширяющими стандартный функционал Flask-приложения.

В файле `.env` задаются переменные окружения, например, `url` для базы данных, данные о конфигурации почтового менеджера или платёжной системы.

`alembic.ini` определяет конфигурацию утилиты для выполнения миграций базы данных, например, путь к папке с миграциями.

`DejaVuSans.ttf` – файл с шрифтом, используемым при создании pdf-файла.

`docker-compose.yml` – файл определяющий поведение множества Docker-контейнеров и общие настройки Docker-окружения.

`Dockerfile` содержит последовательность команд, подлежащих к выполнению при запуске приложения.

`main.py` – входная точка, запускает всё приложение

`requirements.txt` – файл, содержащий набор версий и требований - python-пакетов, необходимых для корректной работы приложения.

`start.sh` – файл, содержащий набор команд, выполняемых при запуске основного Docker-контейнера.

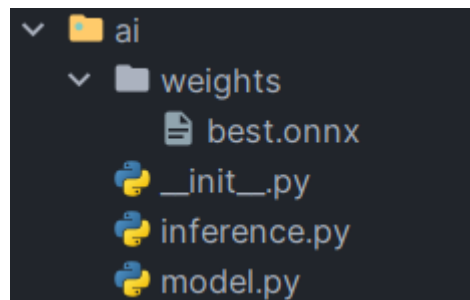


Рисунок 11 - Структура папки ai

Пакет ai содержит в себе веса для модели машинного обучения, а также функции, предназначенные для запуска в работу yolov-модели.

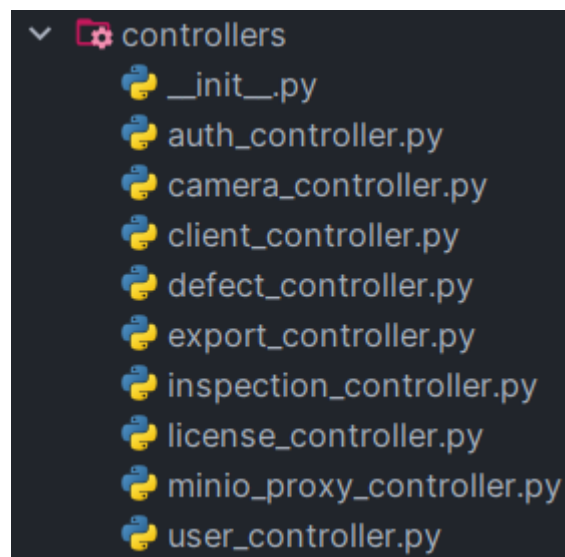


Рисунок 12 - Структура папки controller

Пакет controllers хранит множество контроллеров, обрабатывающих внешние запросы.

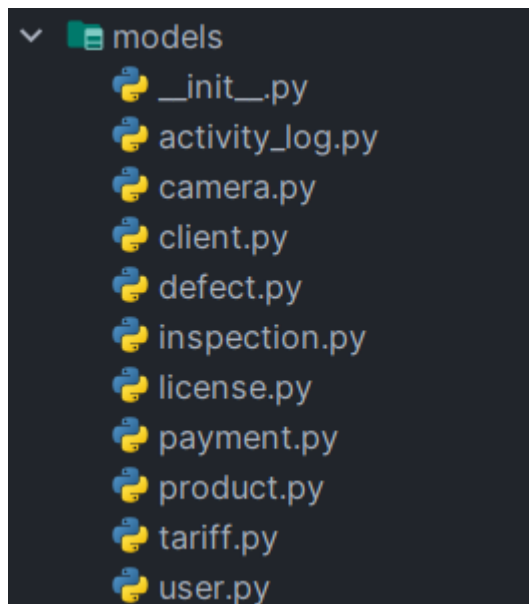


Рисунок 13 - Структура папки models

Пакет models содержит множество моделей – классов, описывающих структуру сущностей из базы данных и предоставляющих их отображение.

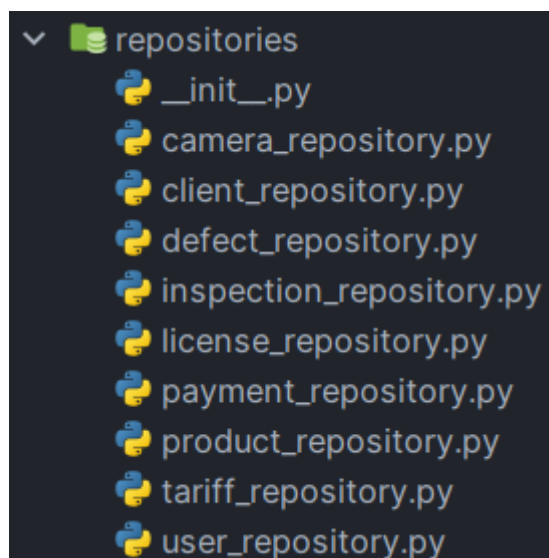


Рисунок 14 - Структура папки repositories

Пакет repositories – директория, содержащая набор классов, предназначенных для работы с базой данных.

В пакете routes лежит единственный файл, регистрирующий все контроллеры из пакета controllers.

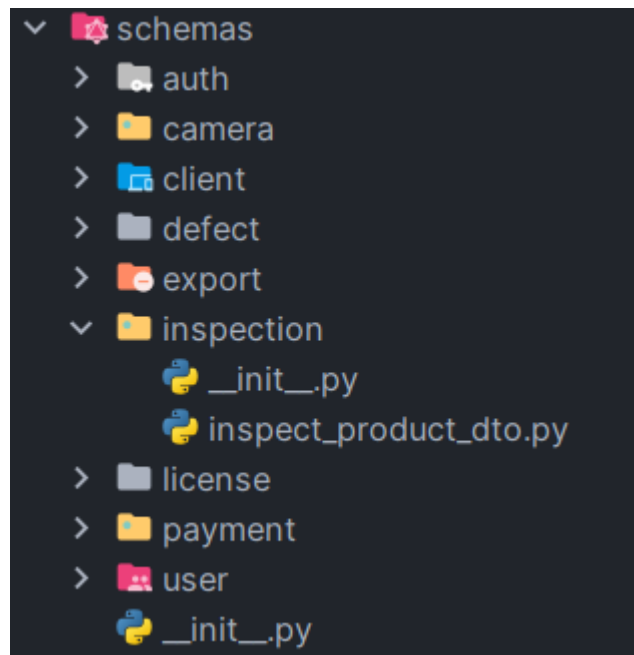


Рисунок 15 - Структура папки schemas

Пакет schemas содержит множество DTO, распределенных по соответствующим подпакетам.

В пакете services лежат файлы, содержащие классы, предназначенные для работы с внешними API MinIO и FreeKassa.

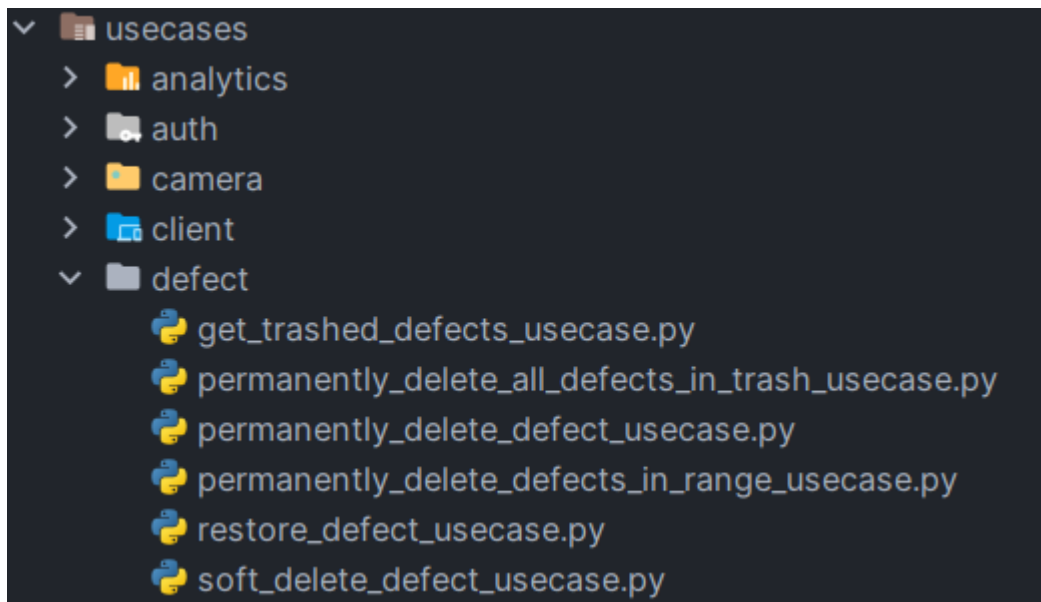


Рисунок 16 - Структура папки usecases

Пакет usecases содержит набор классов, отвечающих за исполнение логики пользовательских сценариев.

Пакет utils содержит множество прочих вспомогательных утилит и декораторов, не относящихся к другим пакетам.

В пакете migrations лежат файлы env.py и script.py.mako, отвечающие за конфигурацию связки утилиты alembic для миграций баз данных и приложения на flask и подробную структуру для файлов миграций соответственно. Директория versions содержит множество файлов, содержащих набор команд для выполнения соответствующей миграции.

В пакете seeders лежит только один исполняемый файл, отвечающий за заполнение базы данных при первом её запуске данными о предоставляемых тарифах.

5. Диаграммы, иллюстрирующие работу системы

Диаграмма прецедентов (Рис. 17-20): Описывает действия для пользователей, модераторов, администраторов и владельцев.

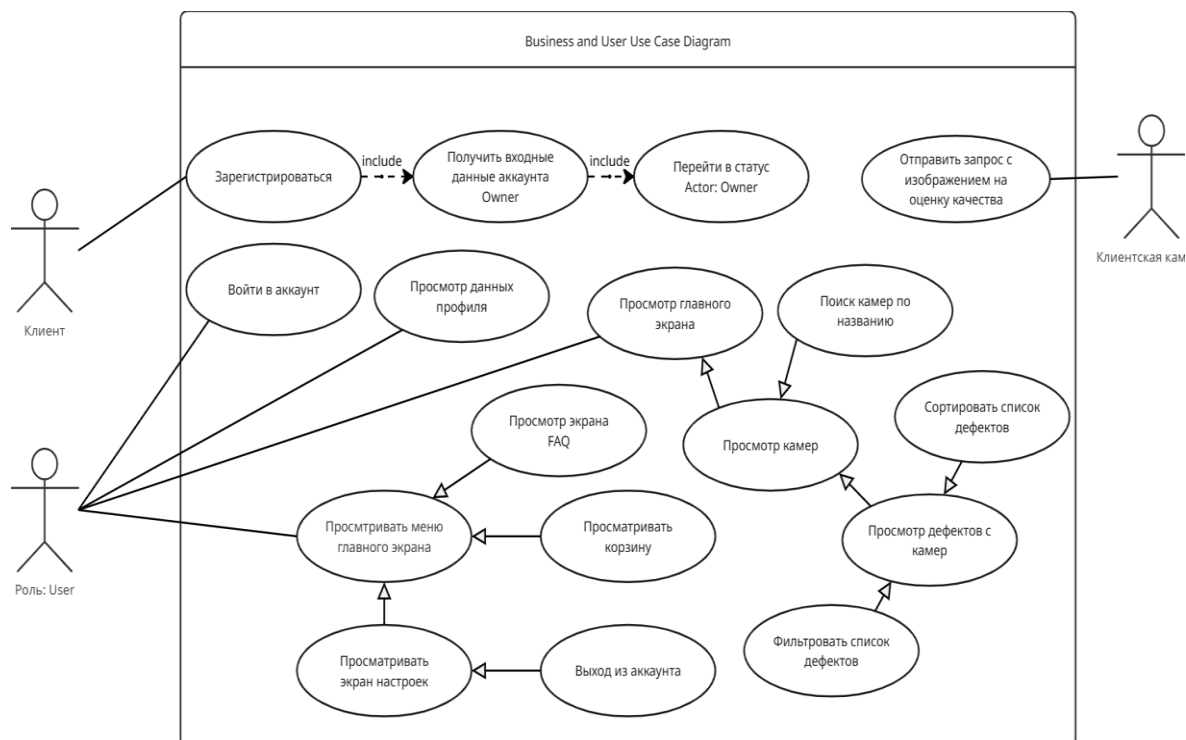


Рисунок 17 - Юзер-клиент диаграмма

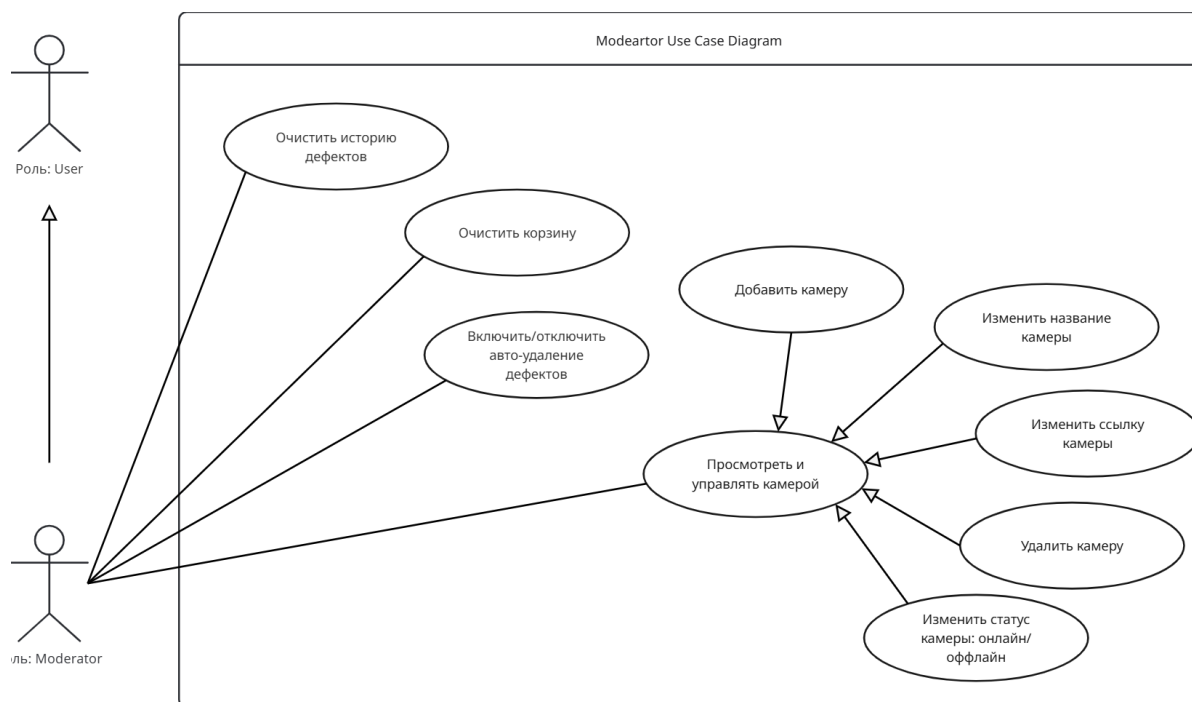


Рисунок 18 - Диаграмма роли «Модератор»

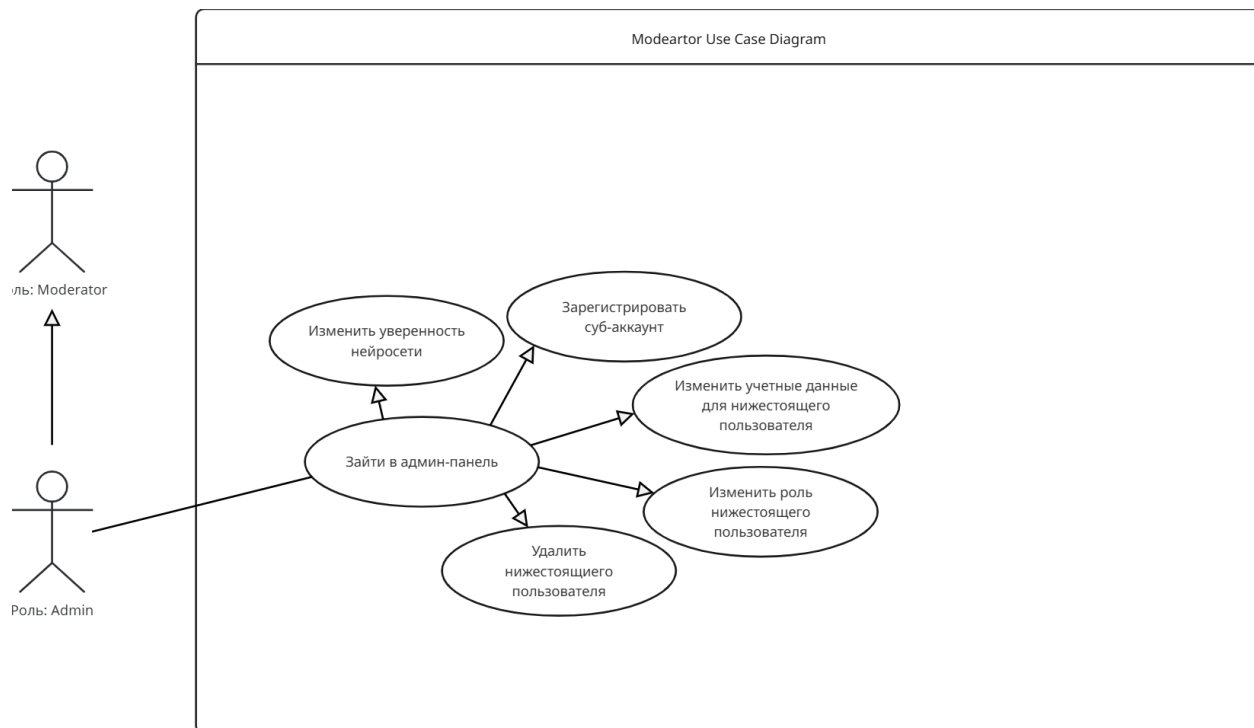


Рисунок 19 - Диаграмма роли «Администратор»



Рисунок 20 - Диаграмма роли «Владелец»

ER-диаграмма (Рис. 21): Показывает структуру базы данных (сущности: пользователи, камеры, отчеты, логи).

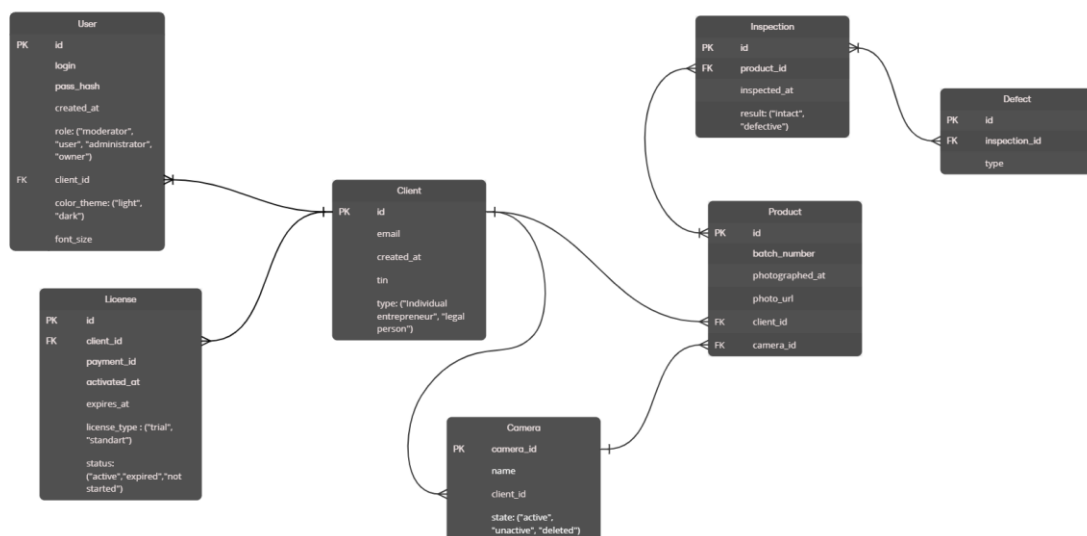


Рисунок 21 - ER-диаграмма

Диаграммы последовательности (Рис. 22-31): Иллюстрируют процессы входа, регистрации, покупки лицензии, добавления камеры и т.д.

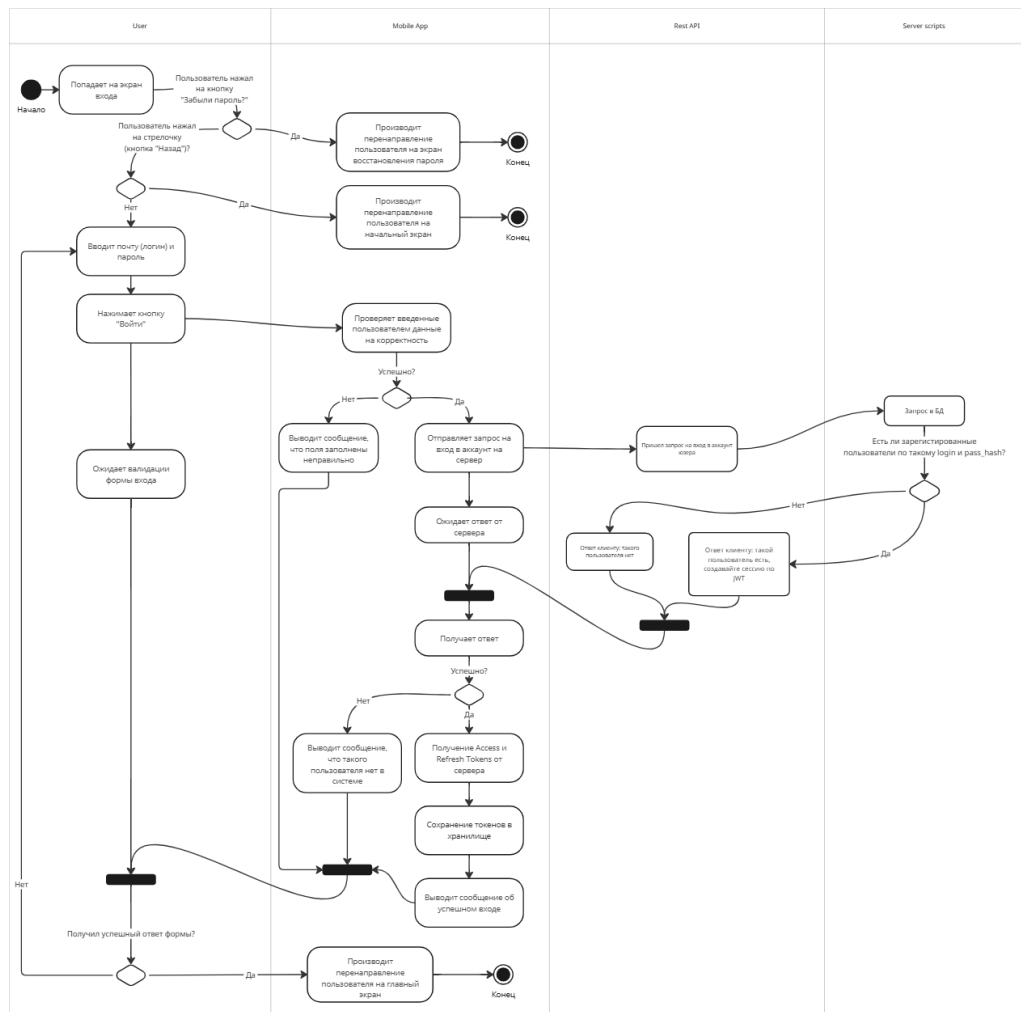


Рисунок 22 - Диаграмма активностей «Логин»

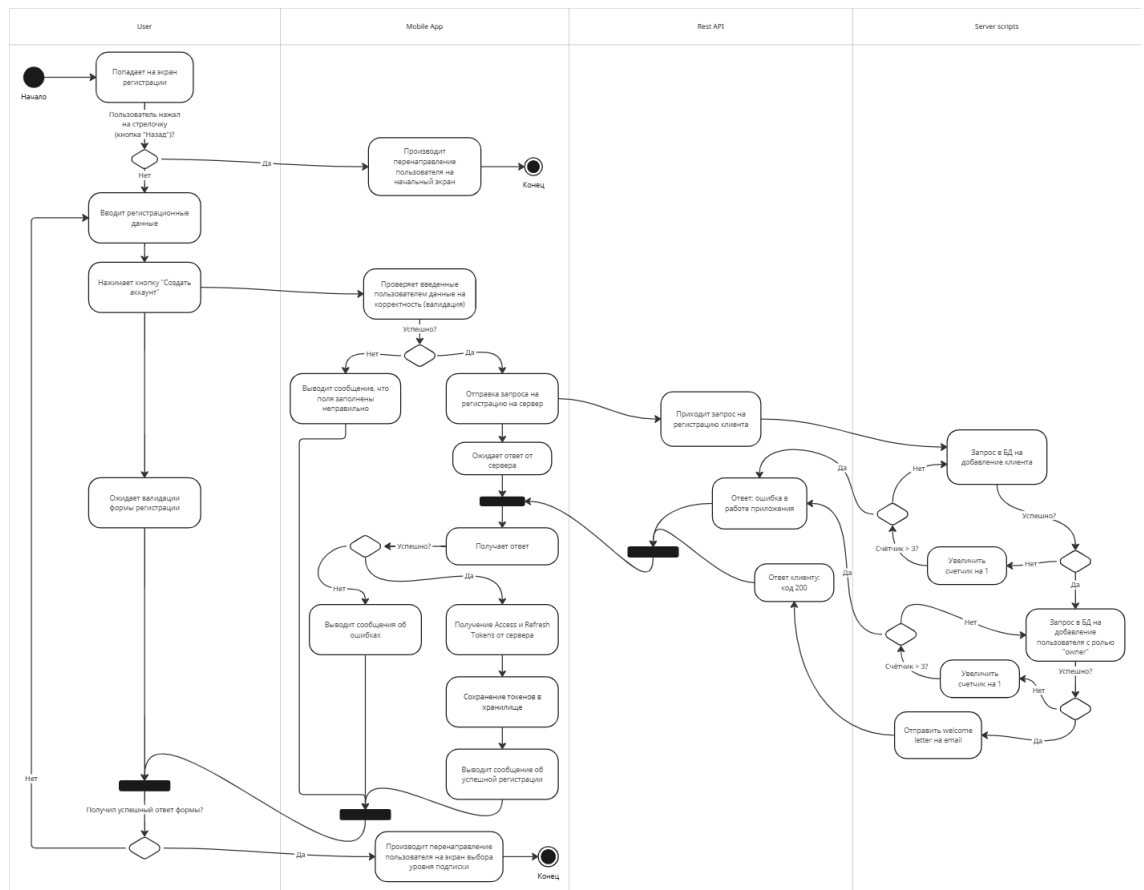


Рисунок 23 - Диаграмма активностей «Регистрация клиента»

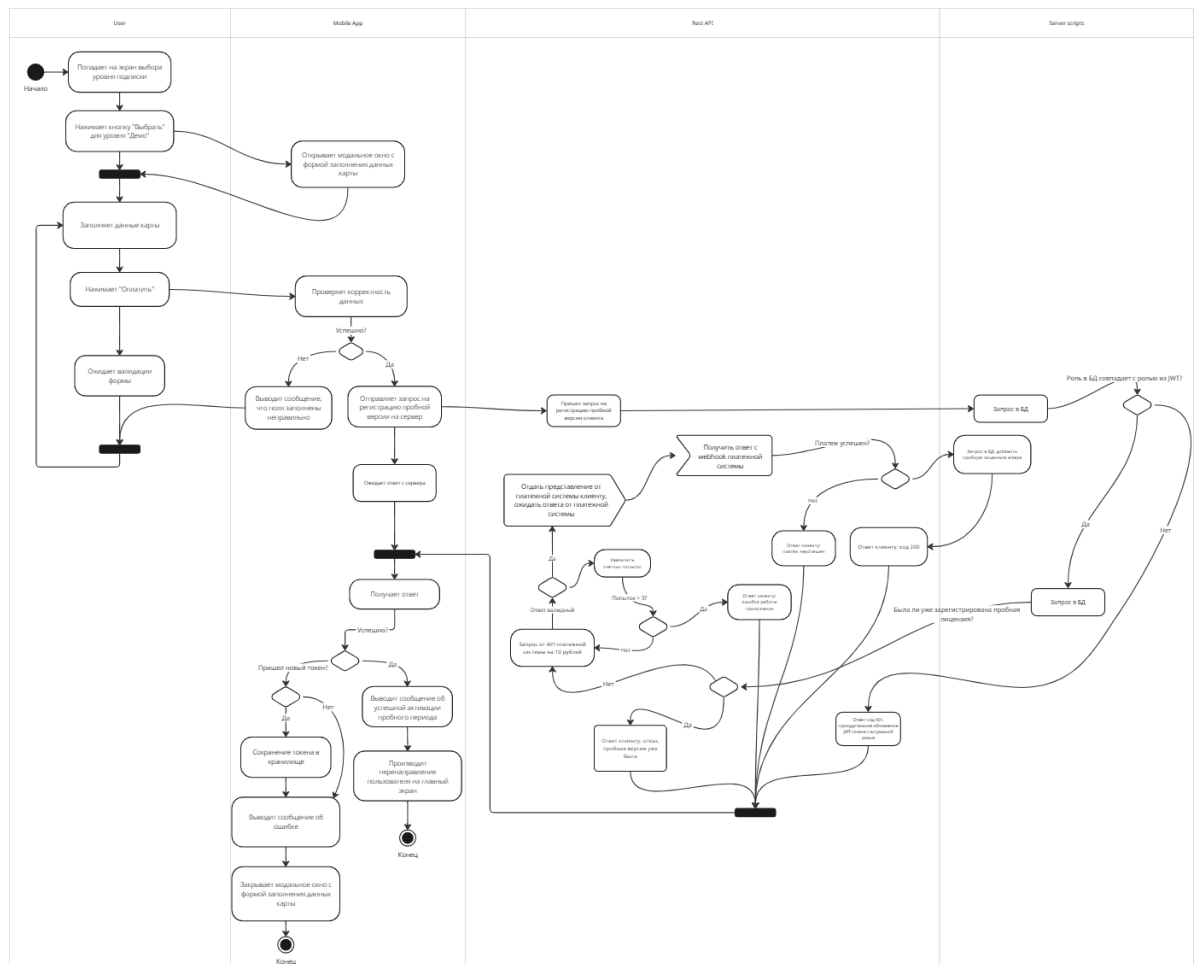


Рисунок 24 - Диаграмма активностей «Регистрация демо-версии»

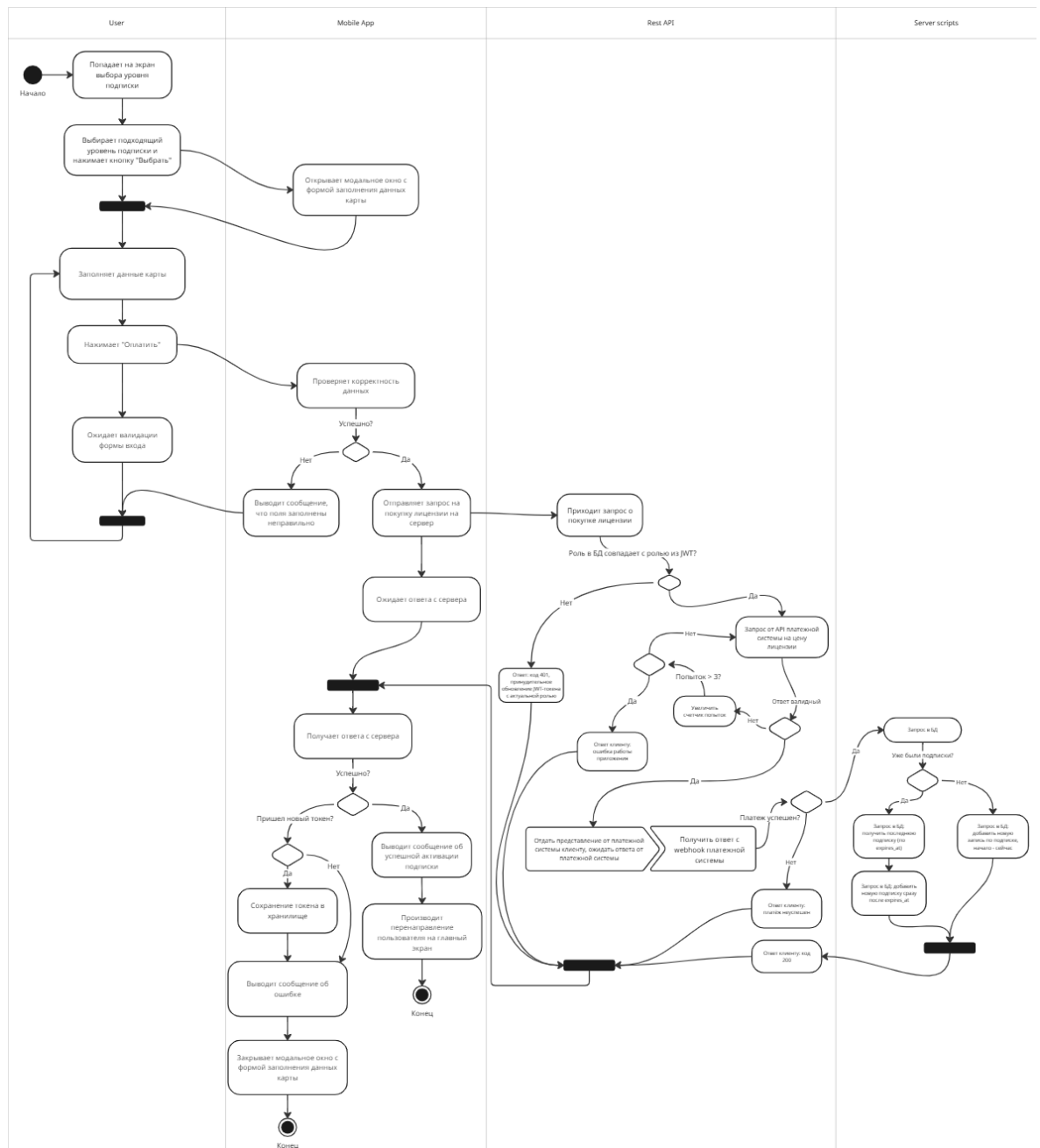


Рисунок 25 - Диаграмма активностей «Приобретение лицензии»

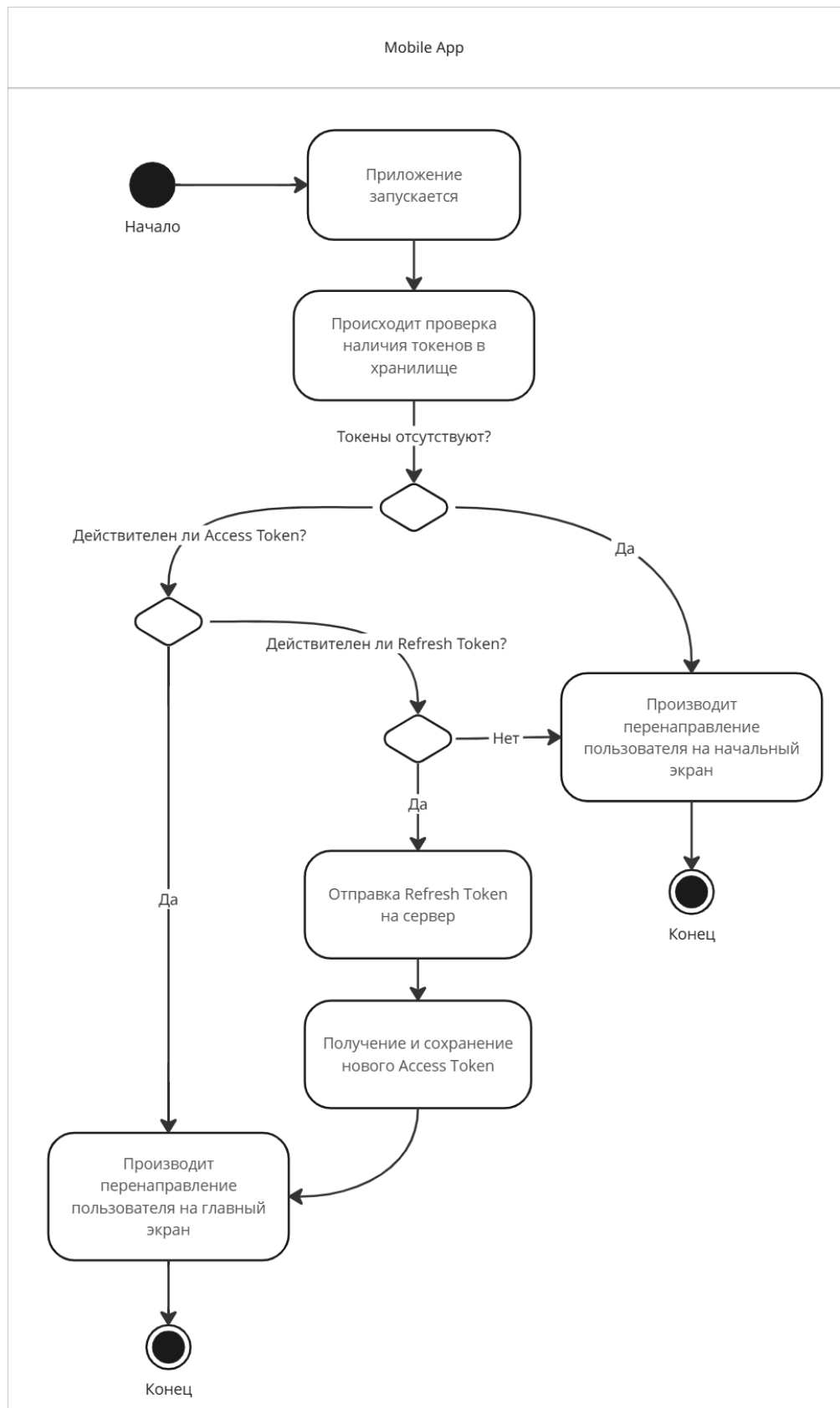


Рисунок 26 - Диаграмма активностей «JWT Token»

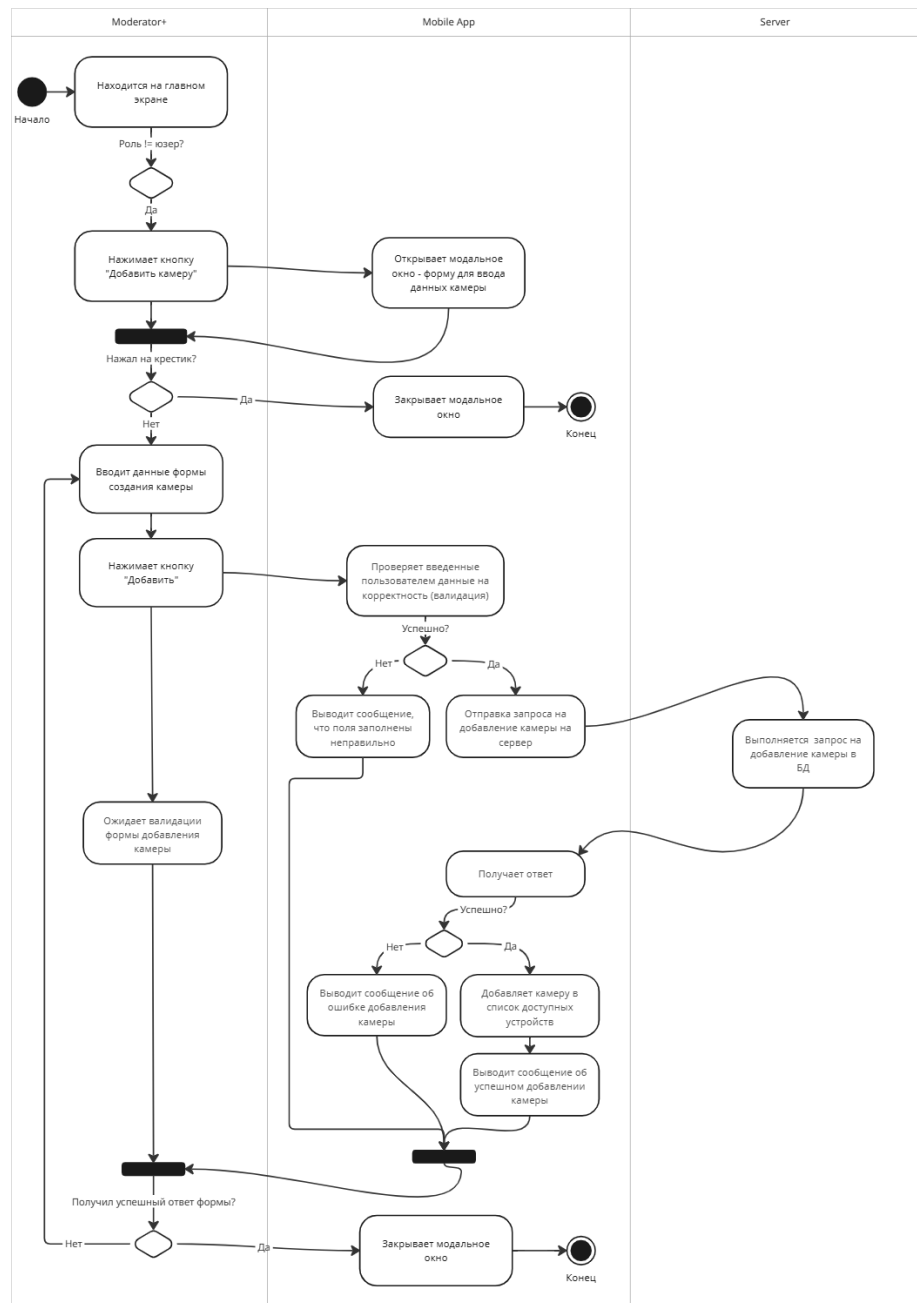


Рисунок 27 - Диаграмма активностей «Добавить камеру»

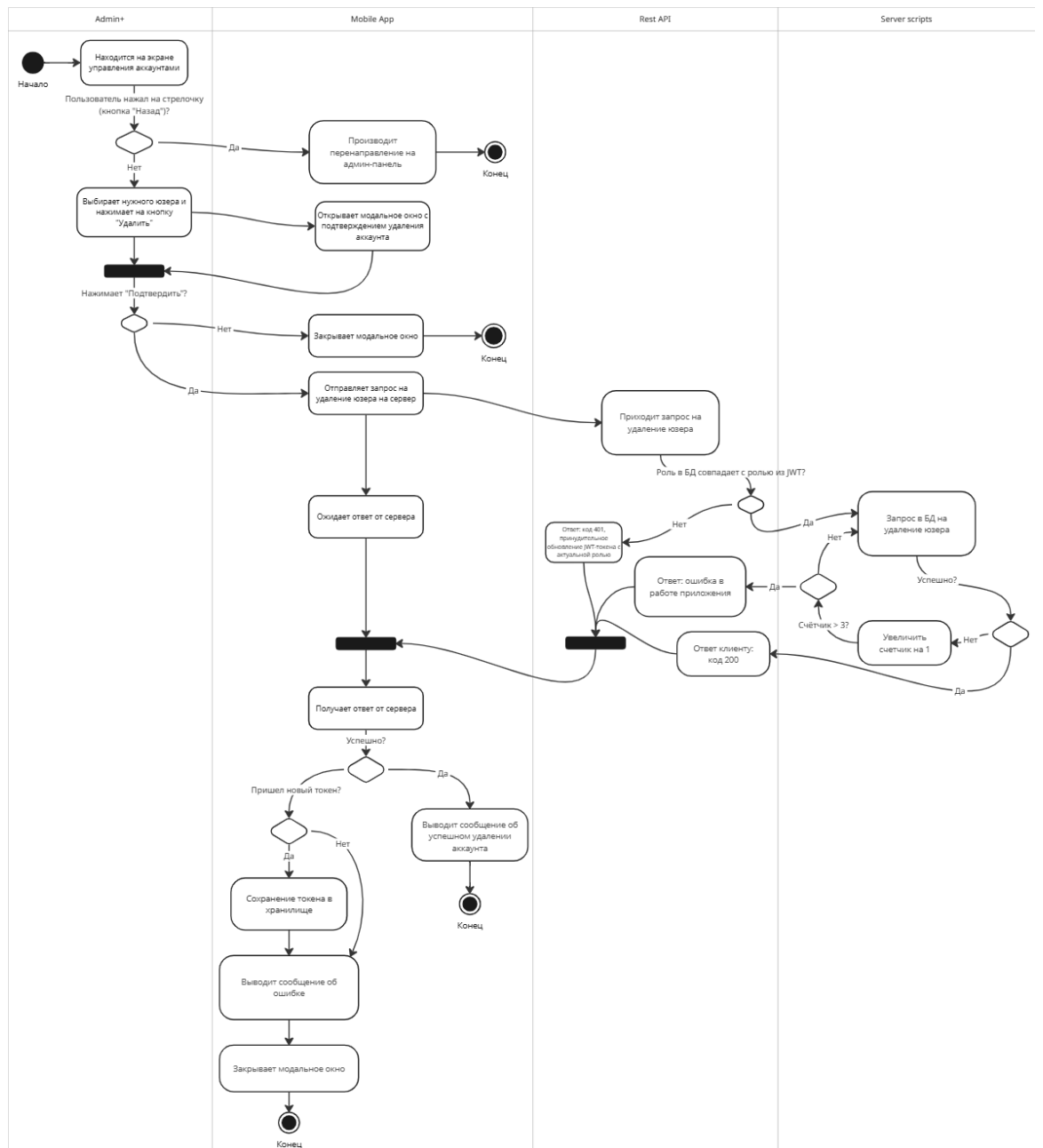


Рисунок 28 - Диаграмма активностей «Изменение роли»

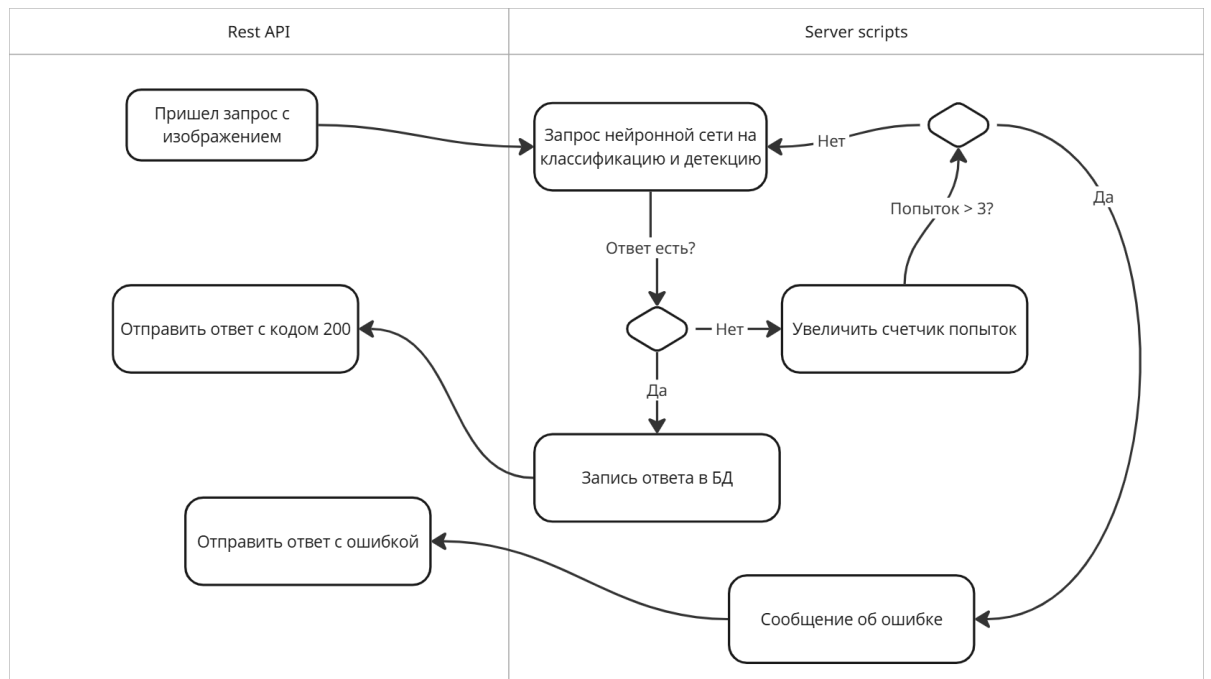


Рисунок 29 - Диаграмма активностей «Проверка качества»

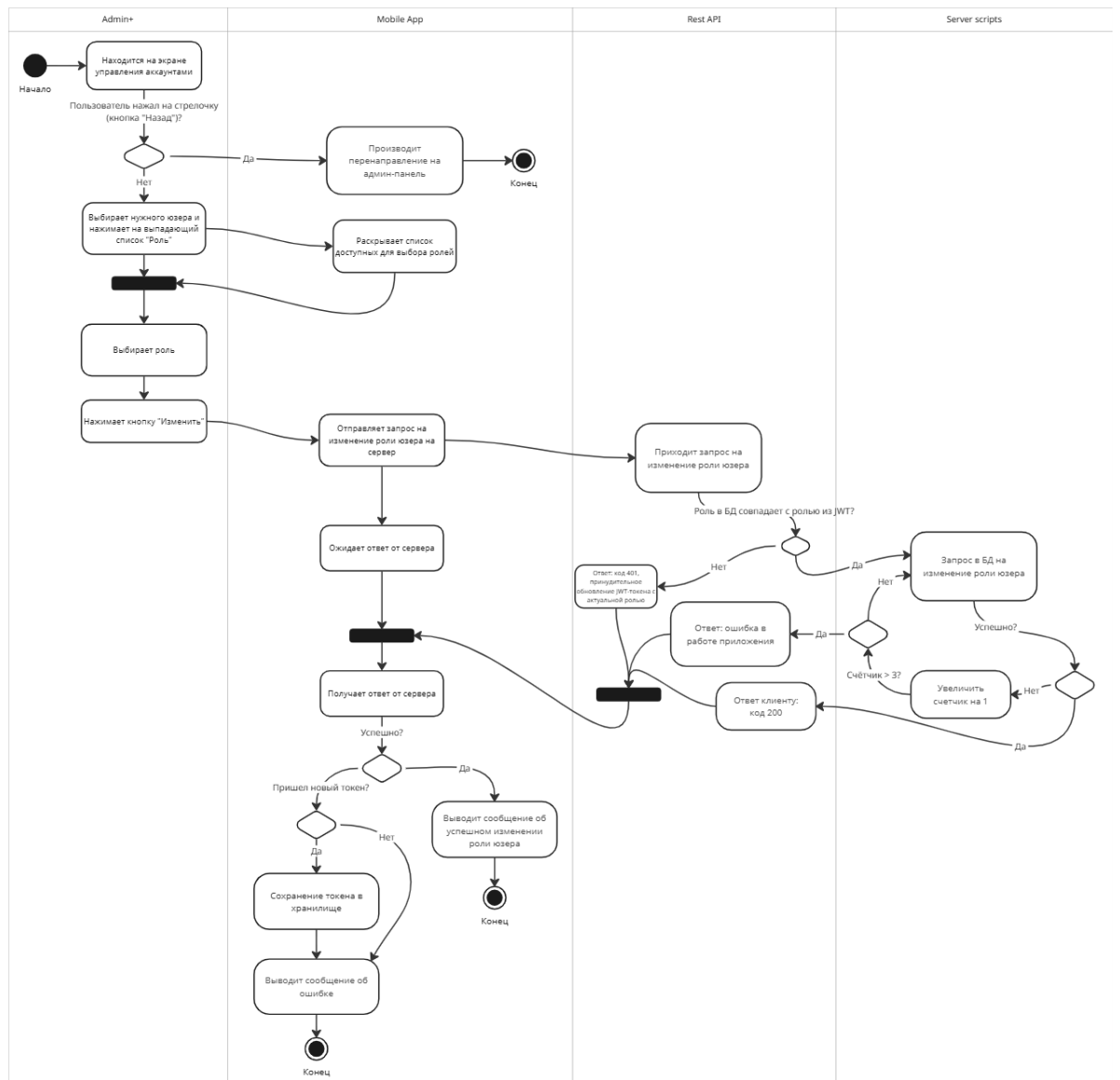


Рисунок 30 - Диаграмма активностей «Управление аккаунтами»

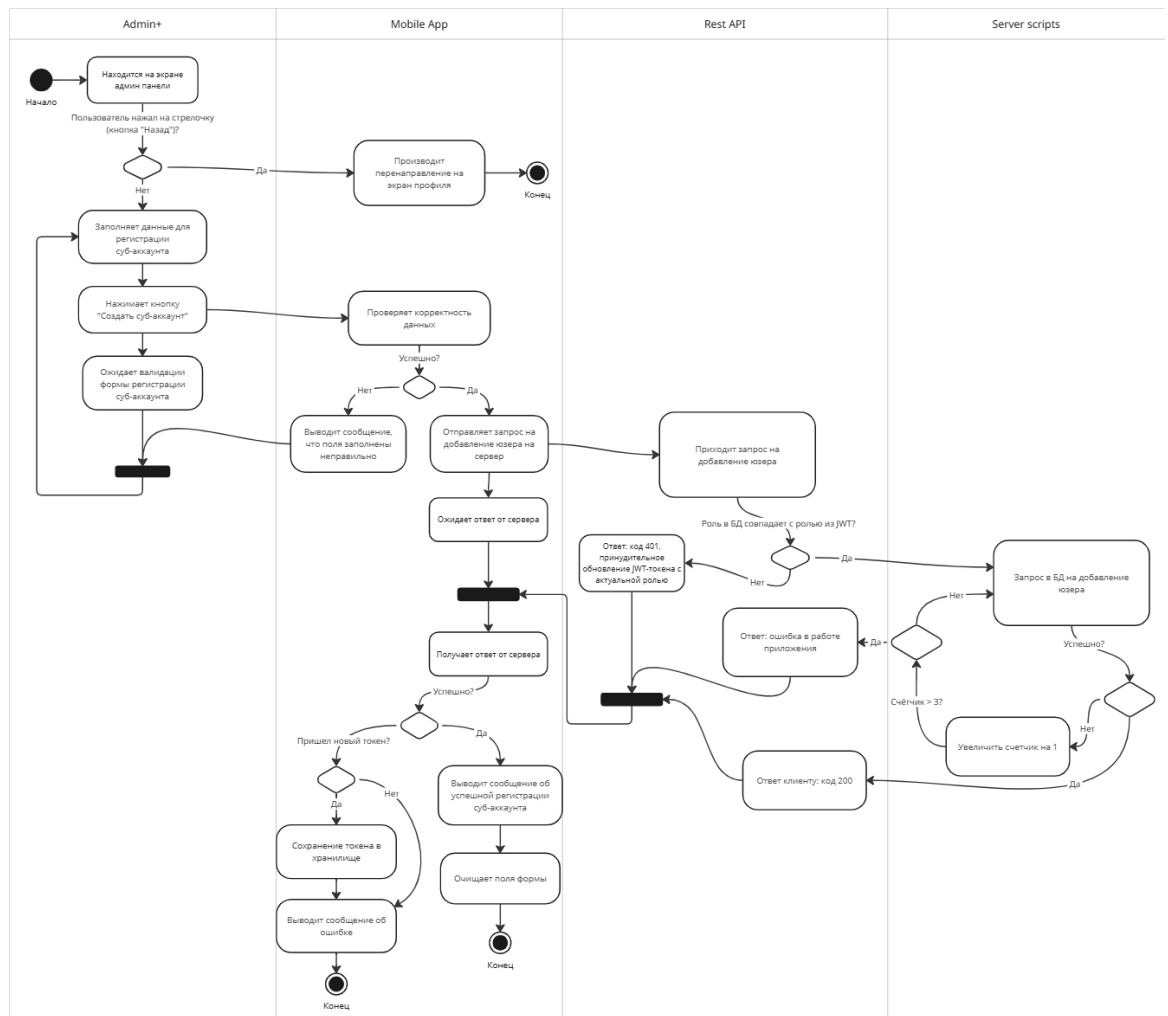


Рисунок 31 - Диаграмма активностей «Регистрация суб-аккаунта»

Диаграммы состояний (Рис. 32-40): Описывают жизненный цикл объектов (вход, регистрация, управление камерами).

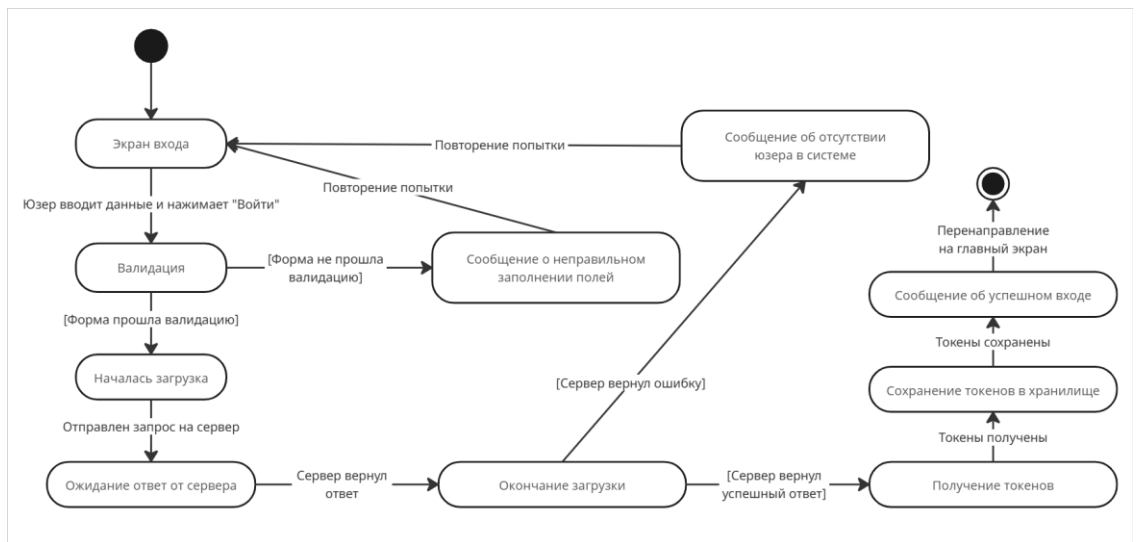


Рисунок 32 - Диаграмма состояний «Экран входа»

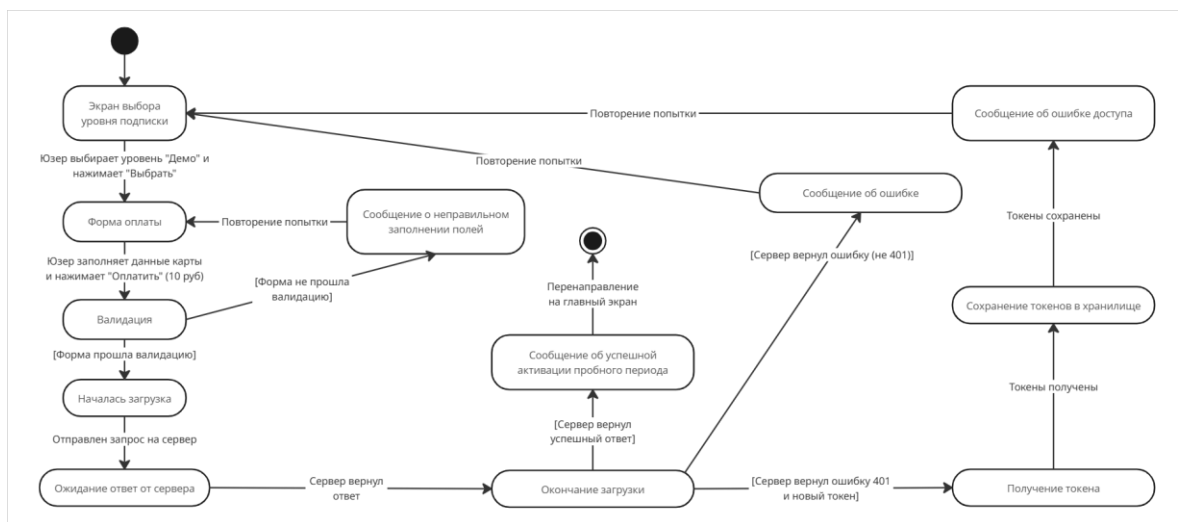


Рисунок 33 - Диаграмма состояний «Активация демо-версии»

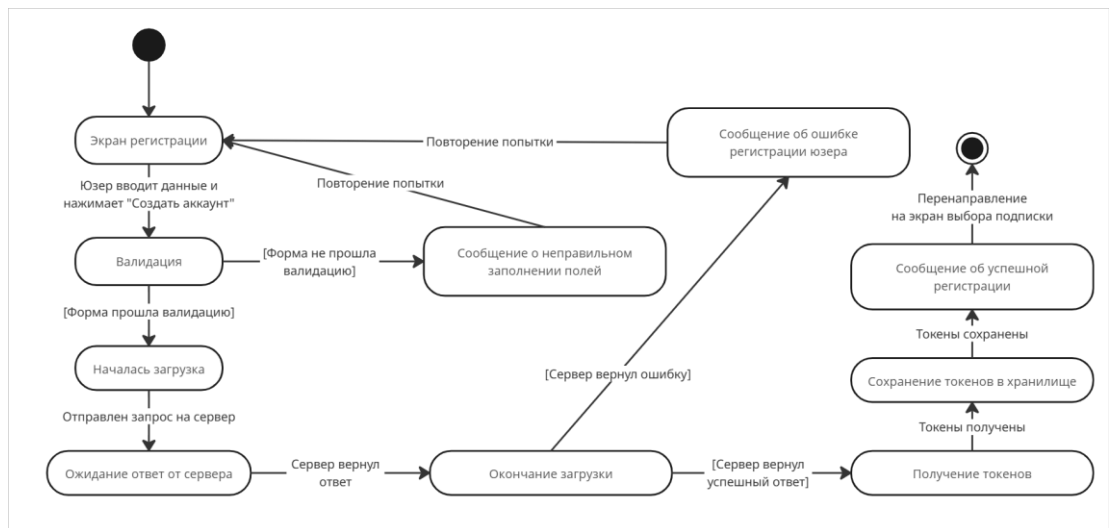


Рисунок 34 - Диаграмма состояний «Регистрация»

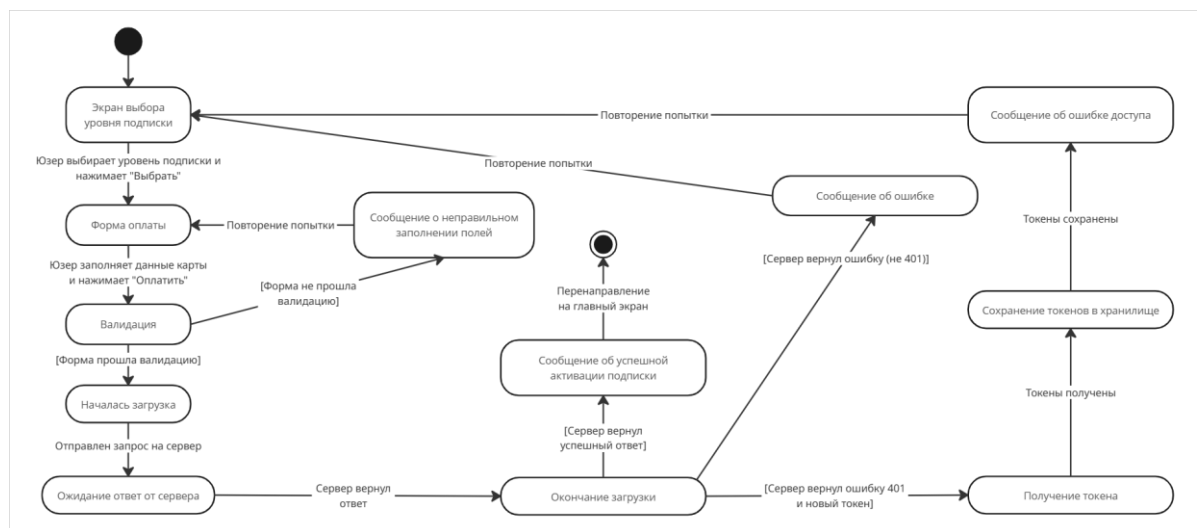


Рисунок 35 - Диаграмма состояний «Приобретение подписки»

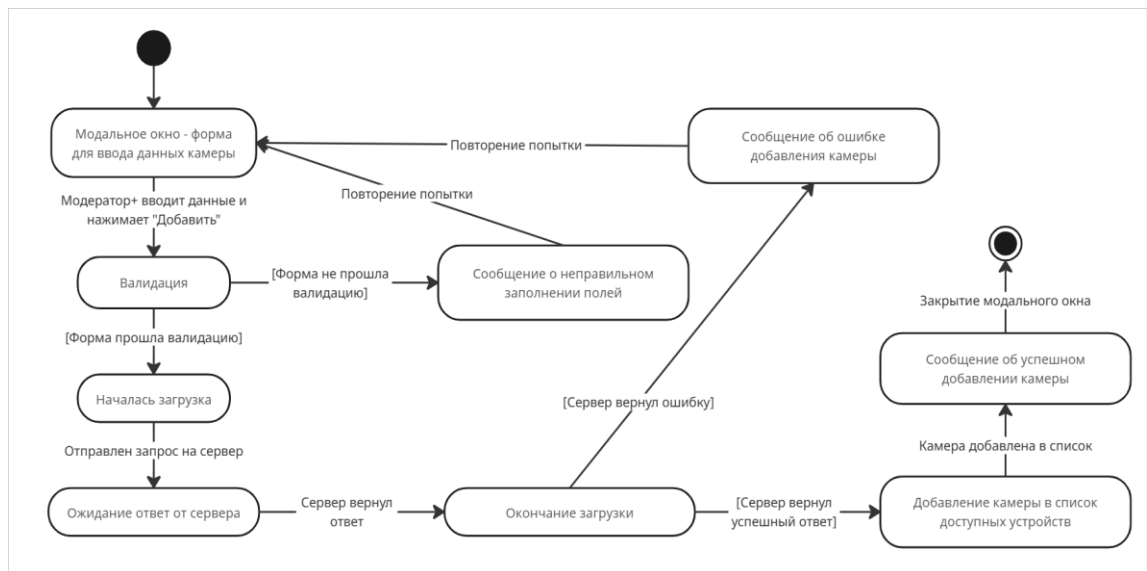


Рисунок 36 - Диаграмма состояний «Добавление камеры»

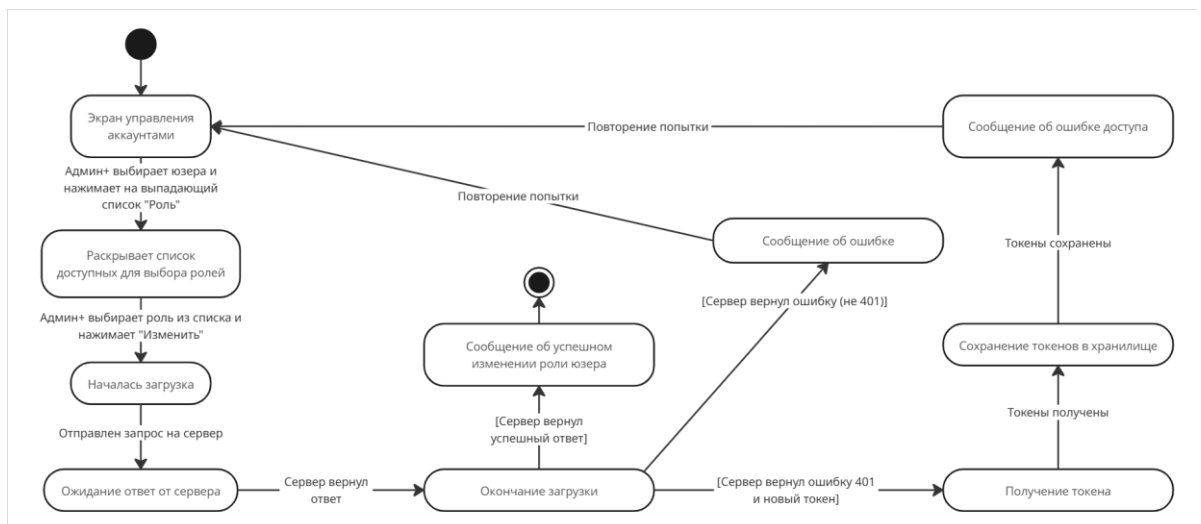


Рисунок 37 - Диаграмма состояний «Изменение роли»

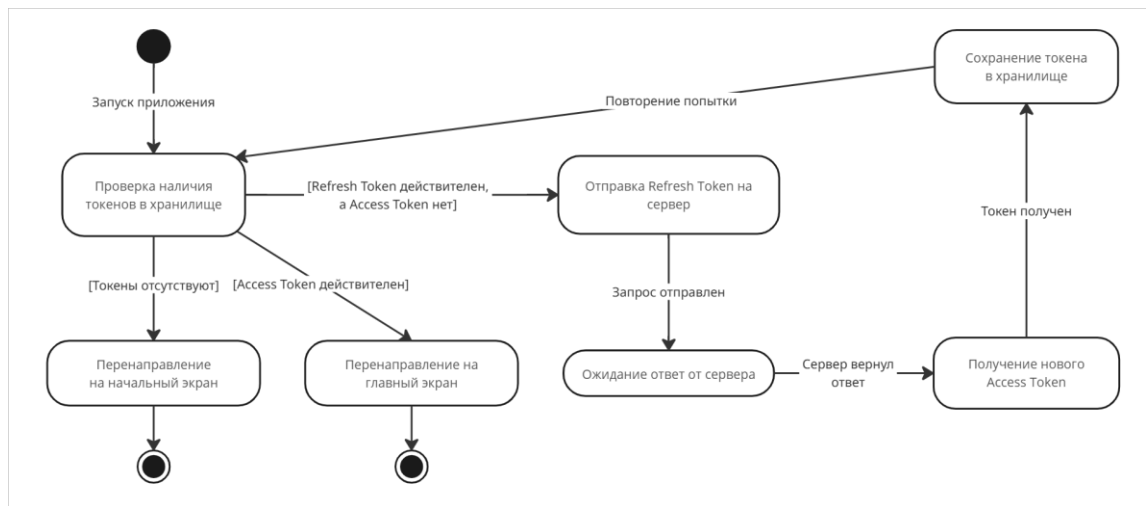


Рисунок 38 - Диаграмма состояний «JWT Token»

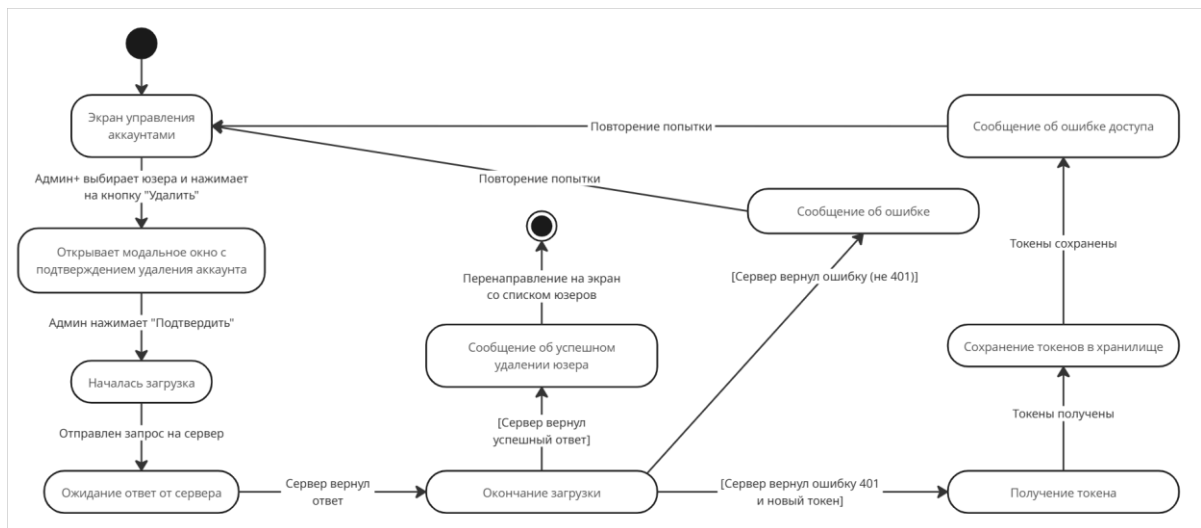


Рисунок 39 - Диаграмма состояний «Удаление суб-аккаунта»

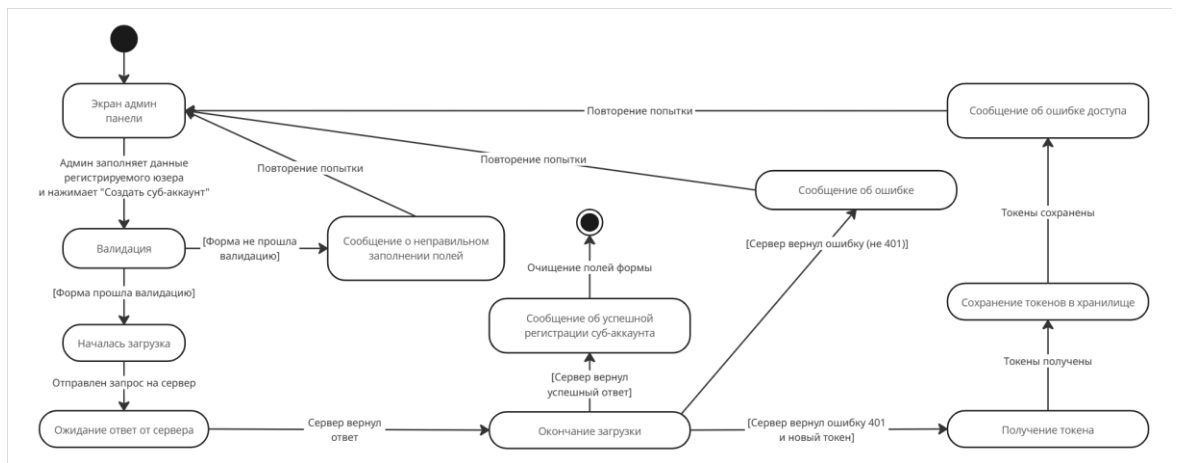


Рисунок 40 - Диаграмма состояний «Регистрация суб-аккаунта»

Диаграмма развертывания: Показывает взаимодействие сервера (Flask, PostgreSQL, MinIO), клиентской части (React Native) и облака (Yandex Cloud).

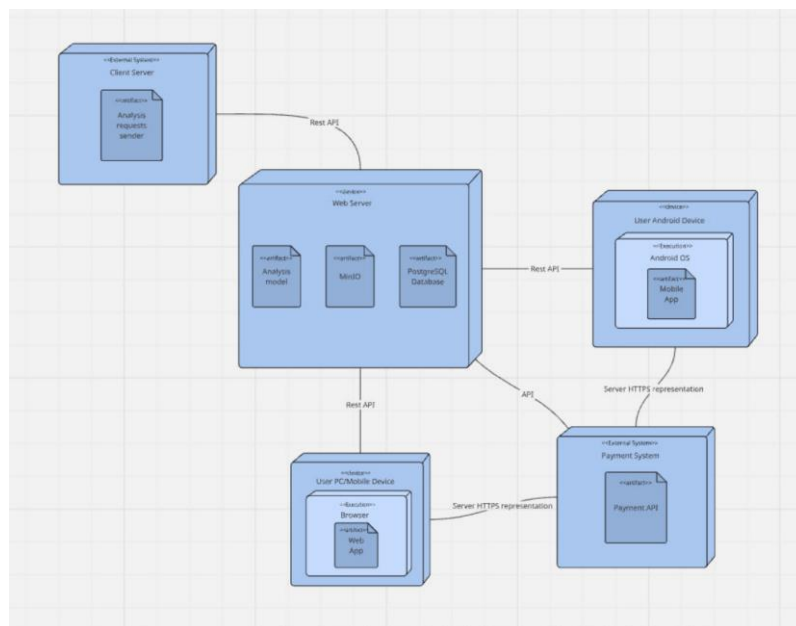


Рисунок 41 - Диаграмма развертывания

6. Дизайн приложения

При первом запуске приложения появляется экран с информацией про приложение и компанию. На экране есть 2 функциональные кнопки «Зарегистрироваться» и «Уже есть аккаунт? Войти».

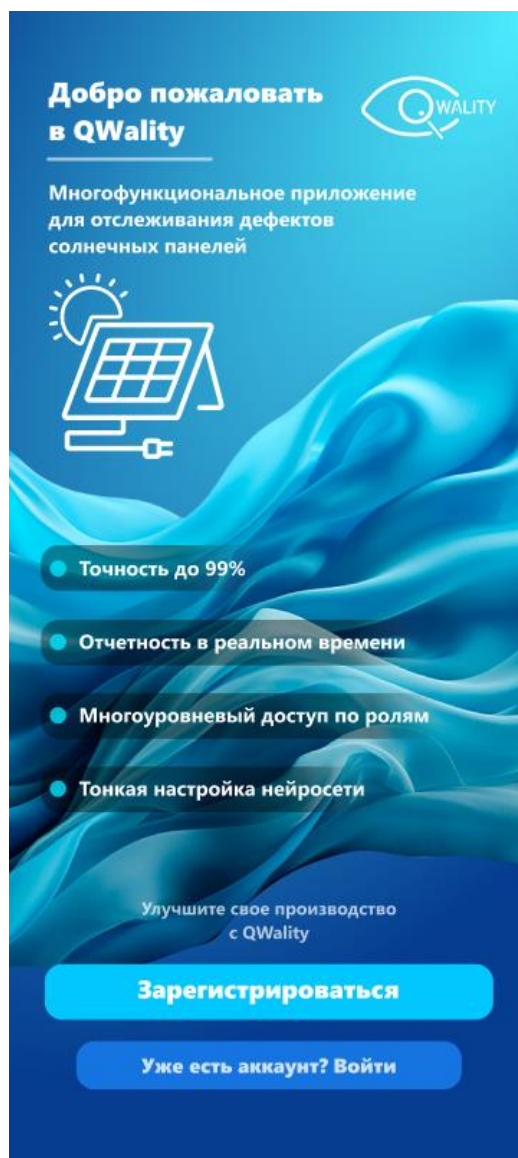


Рисунок 42 - Приветственный экран

При нажатии на кнопку регистрации появляется экран регистрации.

The image shows a mobile application registration screen with a blue gradient background. At the top left is a white back arrow, and at the top center is the title 'Регистрация' in white. Below the title are four white input fields with labels in Russian: 'ИНН', 'Почта', 'Код подтверждения', and 'Пароль'. To the right of the 'Код подтверждения' field is a blue button with white text 'Отправить код'. Below these fields is a large blue button with white text 'Создать аккаунт'. At the bottom, there is a white checkbox followed by the text 'Я принимаю условия пользовательского соглашения', where 'условия пользовательского соглашения' is underlined.

Рисунок 43 - Экран регистрации

При нажатии на кнопку входа появляется экран входа.

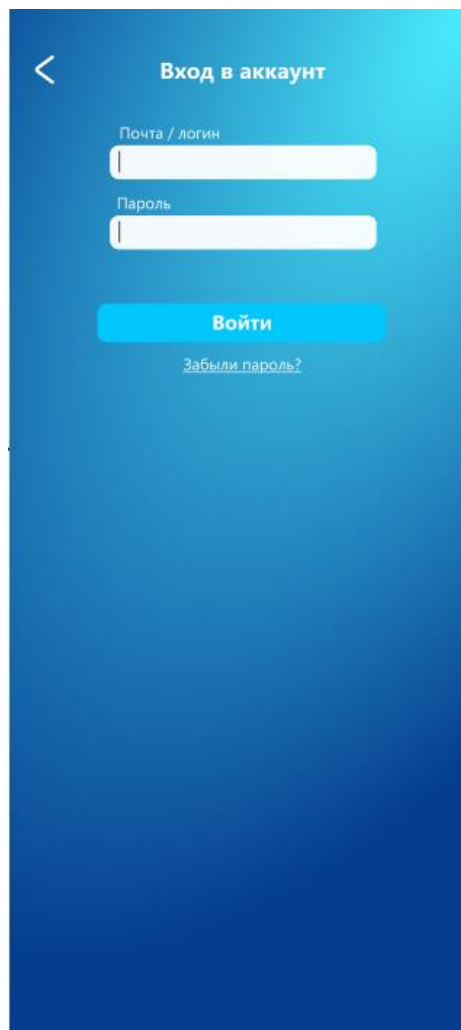


Рисунок 44 - Экран входа

На экране входа есть функциональная кнопка «Восстановить пароль» при нажатии на неё открывается экран восстановления пароля.

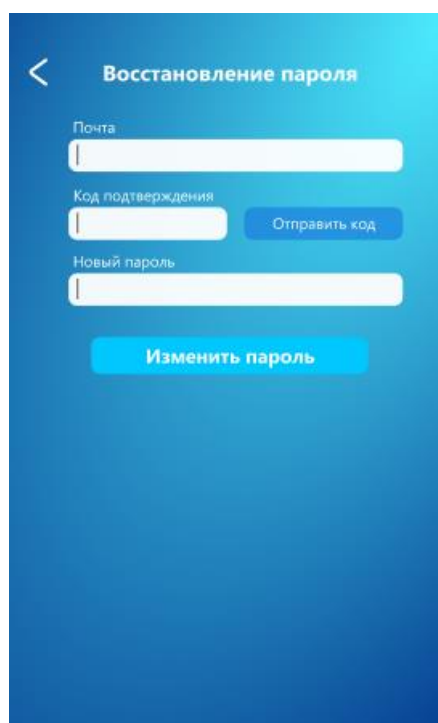
The image shows a mobile application screen for password recovery. The background is a blue gradient. At the top left is a white back arrow icon. To its right is the title 'Восстановление пароля' in white. Below the title are three white input fields with light blue borders. The first is labeled 'Почта', the second 'Код подтверждения', and the third 'Новый пароль'. To the right of the 'Код подтверждения' field is a blue button with white text 'Отправить код'. Below the 'Новый пароль' field is a large blue button with white text 'Изменить пароль'.

Рисунок 45 - Экран восстановления пароля

После регистрации, пользователь может выбрать демо режим, либо сразу купить подписку.

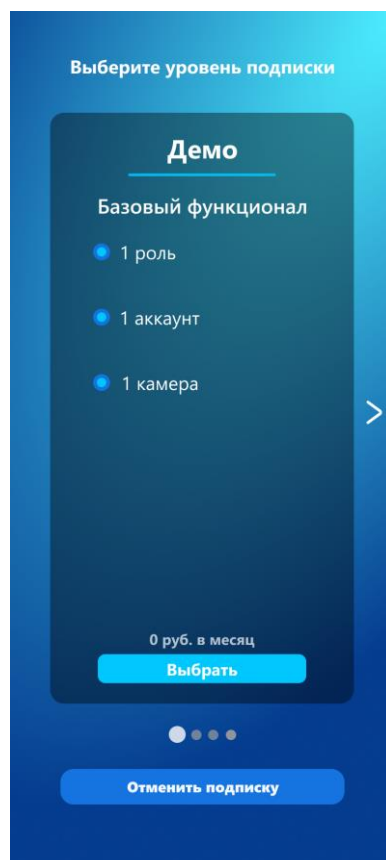


Рисунок 46 - Экран выбора тарифа

После заполнения реквизитов и оплаты подписки или выбора демо режима, создается аккаунт, которым можно управлять на экране профиля.

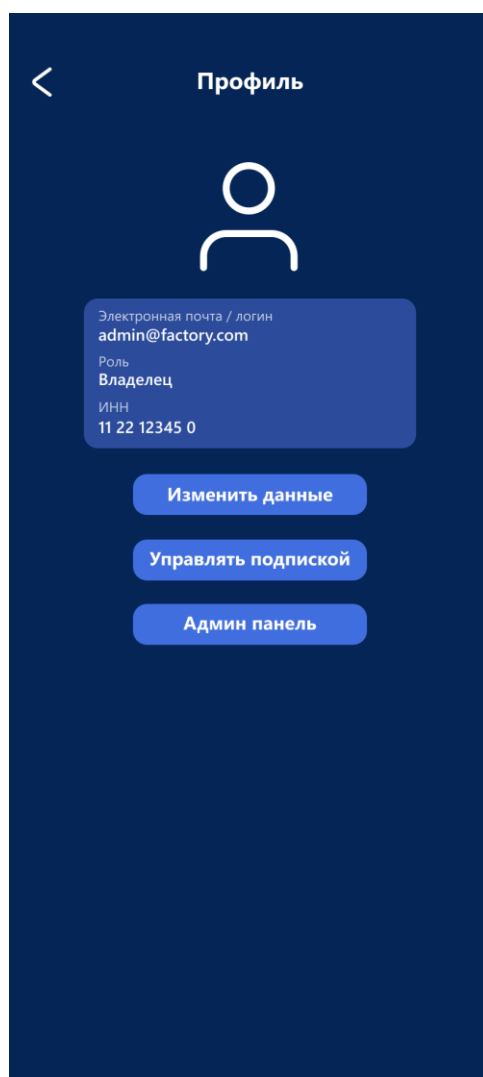


Рисунок 47 - Экран профиля

На экране профиля есть 2 функциональные кнопки «Изменить данные» и «Админ панель» при нажатии на кнопку изменить данные осуществляется переход на экран редактирования данных пользователя.

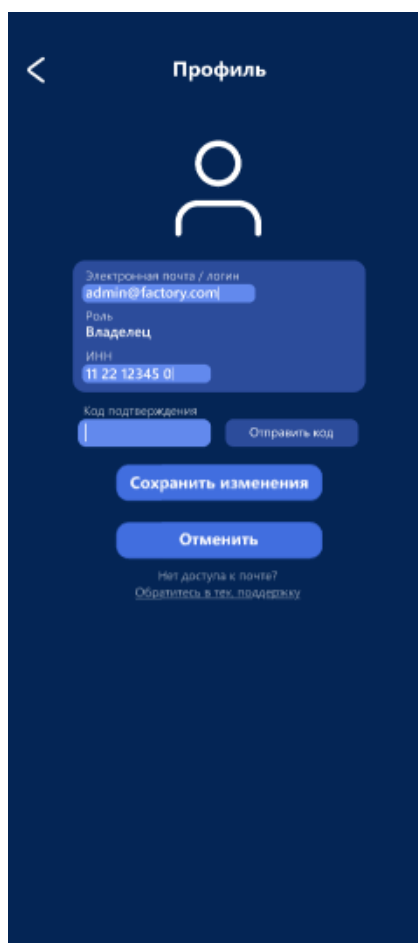


Рисунок 48 - Экран изменения данных профиля

При нажатии на кнопку «Админ панель» осуществляется переход на экран админ панели.

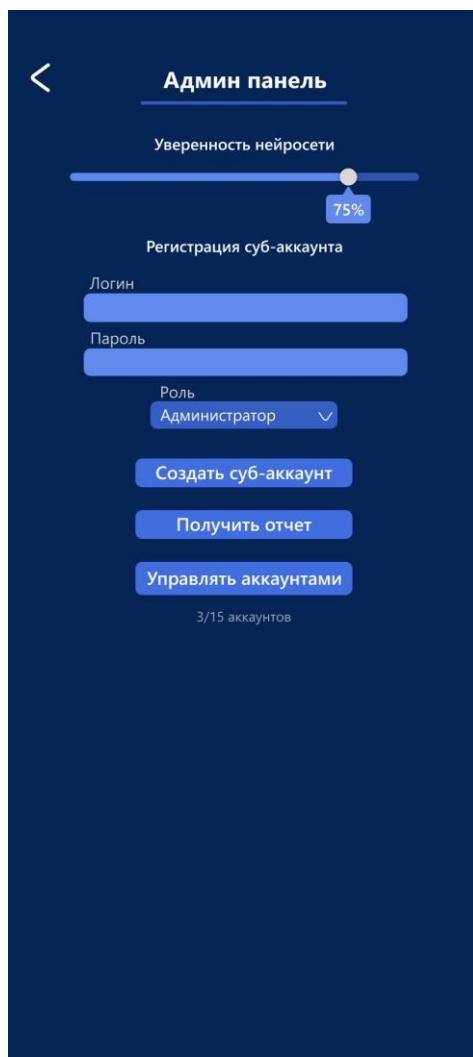


Рисунок 49 - Экран админ-панели

На админ панели есть функциональная кнопка «Управлять аккаунтами», при нажатии на неё осуществляется переход на экран управления аккаунтами.

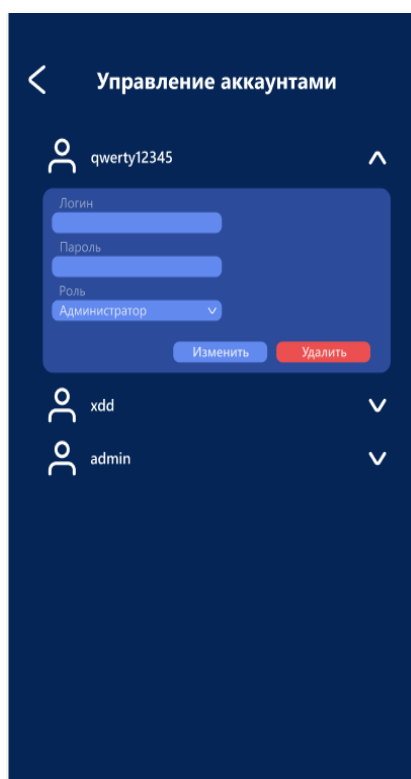


Рисунок 50 - Экран управления аккаунтами

Из профиля пользователь может перейти на главный экран.

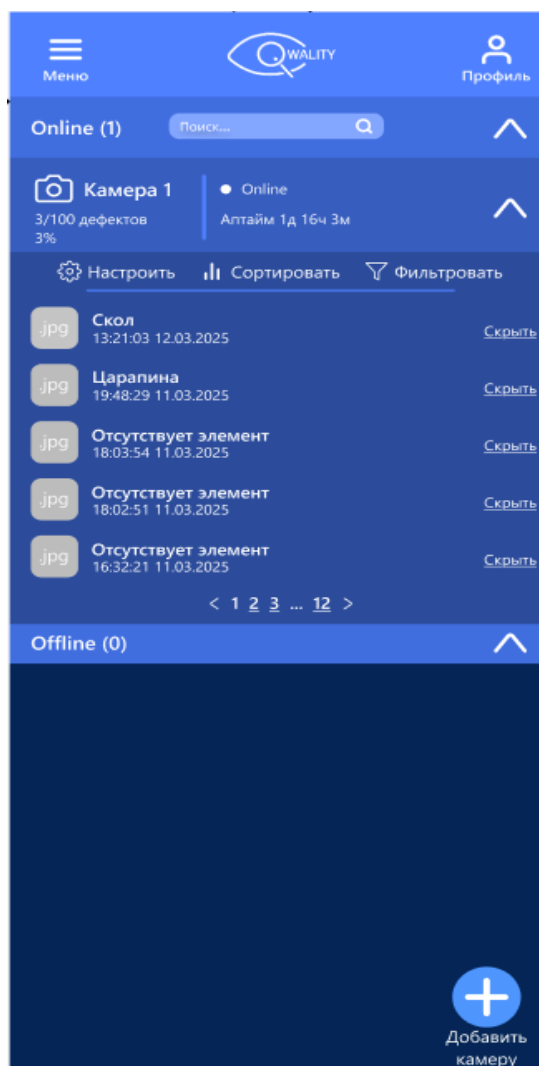


Рисунок 51 - Основной экран

С главного экрана пользователь может перейти на экран добавления камеры.

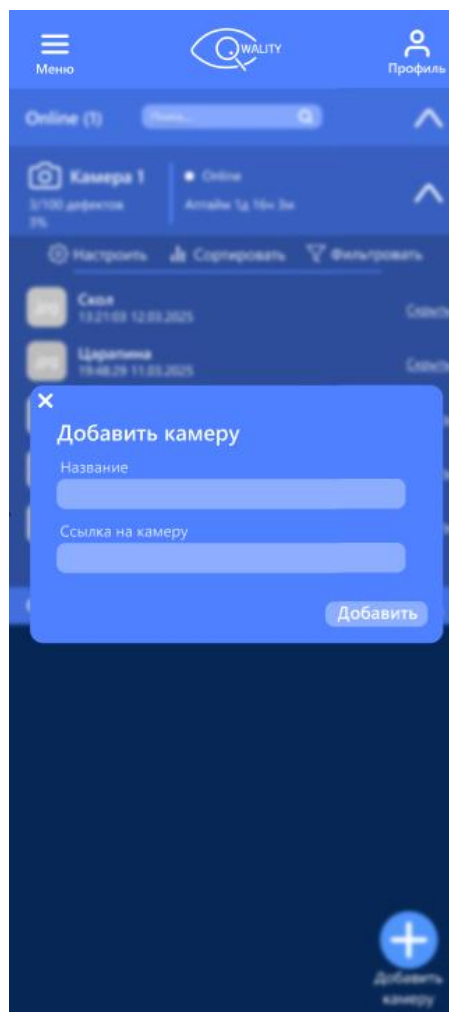


Рисунок 52 - Экран добавления камеры

Также с главного экрана пользователь может через меню перейти на экран корзины.



Рисунок 53 - Экран корзины

С экрана корзины пользователь может перейти на экран очистки.

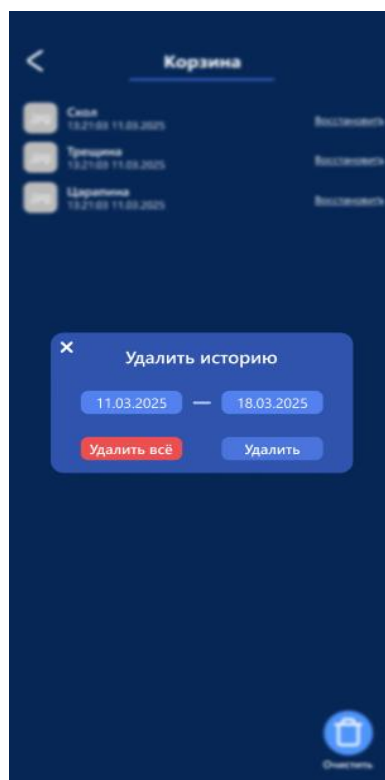


Рисунок 54 - Экран очистки корзины

Пользователь с главного экрана может перейти на экран настроек.

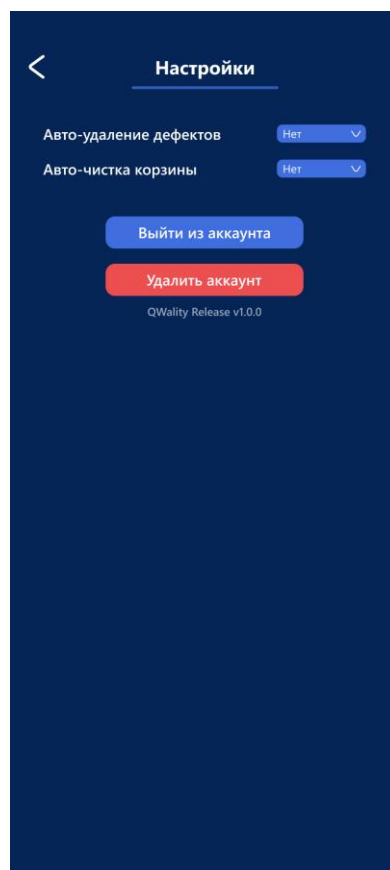


Рисунок 55 - Экран настроек

На экране настроек есть одна функциональная кнопка для выхода из аккаунта, при нажатии на которую осуществляется переход на экран выхода из аккаунта.

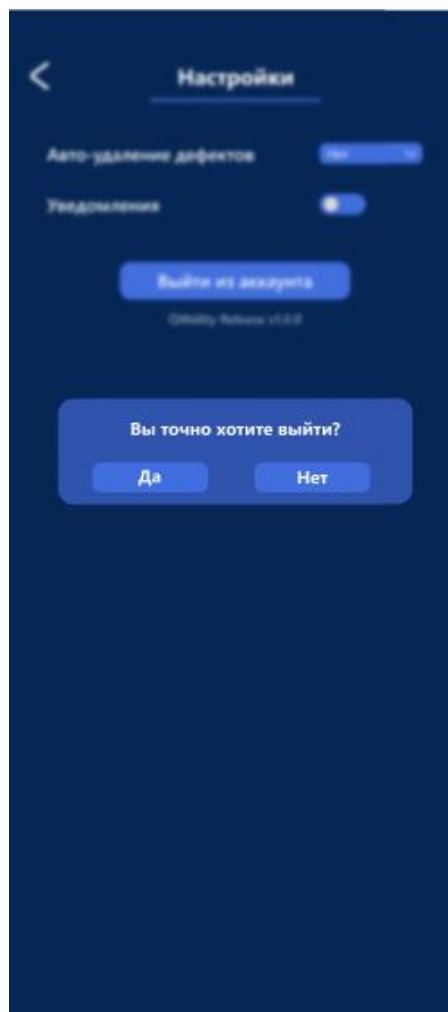


Рисунок 57 - Экран выхода из аккаунта

Приложение 1. Примечания

Тексты Политики конфиденциальности и Пользовательского соглашения прилагаются отдельными файлами к настоящему ТЗ.

Схема API прилагается отдельным файлом к настоящему ТЗ.

Правила коммитов прилагаются отдельным файлом к настоящему ТЗ.