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

Факультет Компьютерных наук
Кафедра программирования и информационных технологий

Курсовая работа
по разработке мобильного приложения
«MeAndFlora»

Зав. Кафедрой_____С.Д. Махортов, д.ф.-м.н., доцент

Обучающийся _____ Д.С. Котов

Обучающийся _____ А.С. Телегина

Обучающийся _____ В.В Шепляков

Руководитель_____ В.А. Ушаков, преподаватель

Руководитель_____ В.С. Тарасов, ст. преподаватель

Воронеж 2024

Содержание

Определения, обозначения и сокращения

В настоящем отчете о ВКР применяют следующие термины с соответствующими определениями.

Мобильное приложение – Программное изделие, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных (портативных, переносных, карманных) устройствах

Frontend – Презентационная часть информационной или программной системы, ее пользовательский интерфейс и связанные с ним компоненты

Backend – Логика работы сайта, внутренняя часть продукта, которая находится на сервере и скрыта от пользователя

Клиент (клиентская сторона) – Приложение, которое предоставляет пользователю возможность взаимодействовать со всей системой

Сервер (серверная часть) – Компьютер, обслуживающий другие устройства (клиентов) и предоставляющий им свои ресурсы для выполнения определенных задач

Микросервис – Веб-сервис, отвечающий за один элемент логики в определенной предметной области

GitHub – Веб-сервис для хостинга IT-проектов и их совместной разработки

PostgreSQL – Реляционная база данных с открытым кодом

Фреймворк – Программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта

Flutter – Комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, веб-приложений, а также настольных приложений под Windows, macOS и Linux с использованием языка программирования Dart

Python – Высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью

Pytorch – Фреймворк для глубокого обучения на языке программирования Python

Java – Строго типизированный объектно-ориентированный язык программирования общего назначения

API – Набор способов и правил, по которым различные программы общаются между собой и обмениваются данными

Spring – Универсальный фреймворк с открытым исходным кодом для Java-платформы

Kafka – Распределённый программный брокер сообщений с открытым исходным кодом, разрабатываемый в рамках фонда Apache на языках Java и Scala

Docker – Программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений

JSON Web Token – Открытый стандарт для создания токенов доступа, основанный на формате JSON

Введение

Идентификация растений имеет решающее значение для различных областей, включая ботанику, экологию, сельское хозяйство и медицину. Традиционные методы идентификации растений часто требуют обширных знаний и специализированного оборудования, что делает их недоступными для многих людей.

В связи с ростом доступности мобильных устройств и развитием технологий искусственного интеллекта возникла возможность создания мобильных приложений, которые могут распознавать растения по фотографиям. Такие приложения имеют огромный потенциал для облегчения идентификации растений и предоставления ценных знаний тем, у кого их нет. Приложение может помочь людям узнать больше о растениях и их значении в природе, способствуя охране окружающей среды. Кроме того, оно может служить инструментом для сбора данных о распространении и экологии растений, а также образовательным ресурсом для учащихся и любителей природы.

Разрабатываемое приложение не только облегчит жизнь садоводам и ботаникам, позволяя быстро определять растения, но также может быть полезным инструментом для образовательных целей и научных исследований. Таким образом, данный проект направлен на улучшение пользовательского опыта и расширение возможностей в области распознавания растений с помощью современных технологий.

В целом, мобильные приложения для распознавания фотографий являются быстро развивающейся областью с большим потенциалом для влияния на различные отрасли. Продолжающиеся исследования и инновации в этой области обещают сделать идентификацию растений более легкой, быстрой и точной, чем когда-либо прежде.

1 Постановка задачи

1.1 Цели создания системы

Целями данной курсовой работы являются:

— упрощение поиска информации о растениях по названиям или фото;

— получение прибыли с интегрированной сторонней рекламы;

— получение актуальной информации о местоположении растений с целью обновления информации об их расселении;

— сбор актуальной базы фотографий растений в исследовательских целях.

1.2 Функциональные требования к разрабатываемой системе

— Функциональными требованиями системы являются:

— получение описания и фото растения по названию;

— получение названия и описания растения по сделанной или выбранной из галереи фотографии;

— просмотр подробной информации о растении;

— просмотр информации о ранее распознанных растениях авторизированным пользователям и ботаникам;

— добавление понравившихся растений в список отслеживаемых авторизированным пользователям;

— редактирование данных своего аккаунта после авторизации или регистрации в системе;

— просмотр списка нераспознанных растений ботаником;

— идентификация ботаником растения из списка неправильно распознанного нейронной сеть;

— создать/удалить пользователей;

— просмотр статистики распознавания;

— просмотр статистики показа рекламы.

1.3 Задачи системы

Задачами работы являются:

— анализ рынка мобильных приложений по идентификации растений по фотографиям для определения сильных и слабых сторон конкурентов;

— определение требований к приложению;

— разработка архитектуры приложения и базы данных для хранения информации о пользователях и растениях;

— реализация функционала приложения.

— разработка эффективной схемы взаимодействия пользователя с интерфейсом мобильного приложения;

— тестирование.

2 Анализ предметной области

Идентификация растений по фотографиям является относительно новой областью исследований, которая возникла с развитием технологий искусственного интеллекта и машинного обучения. Первые попытки создания таких приложений были предприняты в начале 2010-х годов, и с тех пор эта область быстро развивается.

Мобильные приложения для распознавания растений можно разделить на два основных типа:

Приложения на основе искусственного интеллекта. Эти приложения используют алгоритмы искусственного интеллекта для анализа изображений и распознавания растений. Они обычно имеют высокую точность и могут идентифицировать широкий спектр растений.

Приложения на основе базы данных. Эти приложения сравнивают изображения растений с базой данных известных растений. Они обычно менее точны, чем приложения на основе искусственного интеллекта, но могут быть полезны для идентификации растений, которые не представлены в базе данных искусственного интеллекта.

2.1 Обзор аналогов

Для выявления сильных и слабых сторон приложений для поиска экскурсий выведем общие критерии для сравнения:

— существование бесплатного доступа к идентификации по фотографии;

— возможность поиска по названию;

— сохранение истории;

— возможность отслеживания растений;

— повторная проверка идентификации.

Результат, проведенного сравнения аналогов, представлен в Таблица 1.

Таблица 1 - обзор аналогов

	Бесплатный доступ	Поиск по названию	Сохранение истории	Отслеживание растений	Повторная проверка
PlantNet	+	+	+	-	-
INaturalist	+	+	+	-	-
PlantSnap	-	+	+	-	-

2.1.1 PlantNet

Pl@ntNet - это гражданская научная платформа, использующая искусственный интеллект (ИИ) для облегчения идентификации и инвентаризации видов растений. Это одна из крупнейших в мире обсерваторий биоразнообразия, в которой участвуют несколько миллионов человек из более чем 200 стран.

Приложение Pl@ntNet, доступное в веб-версиях и для смартфонов (Android, iOS), позволяет бесплатно идентифицировать десятки тысяч видов растений, просто сфотографировав их. Также доступен поиск растения по названию. Иллюстрация страницы выбора фотографии представлена на Рисунок 1- страница выбора фотографии в приложении PlantNet.

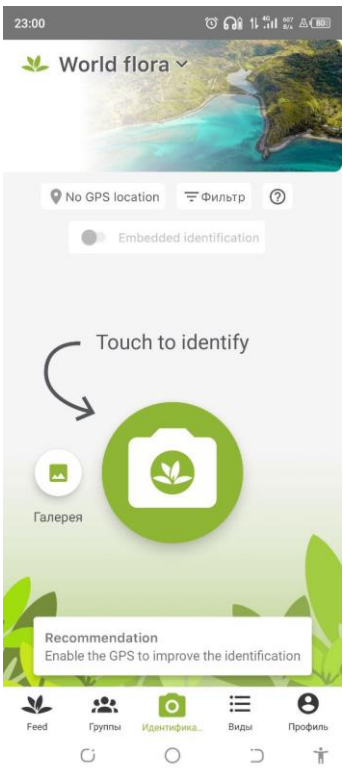

Рисунок 1- страница выбора фотографии в приложении PlantNet

В основе Pl@ntNet лежит принцип совместного обучения. Пользователи, создавшие учетную запись, могут хранить историю идентификаций, а также делиться своими наблюдениями, которые затем могут быть рассмотрены сообществом и использованы ИИ для обучения распознаванию растений. Иллюстрация страницы поиска по названию приложения представлена на Рисунок 2 - страница поиска по названию в приложении PlantNet.

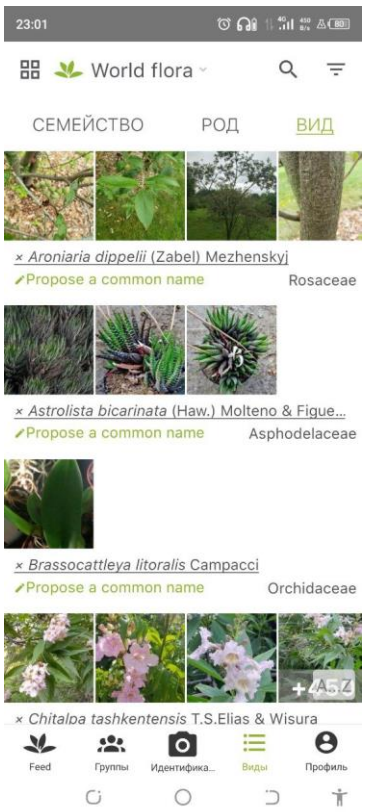

Рисунок 2 - страница поиска по названию в приложении PlantNet

Pl@ntNet собрала более миллиарда изображений растений. Однако лишь небольшая часть из них доступна исследователям по всему миру (через открытые порталы данных о биоразнообразии, такие как GBIF). Важным элементом является наличие GPS-координат. Эта информация имеет решающее значение для составления карт видов. Существуют также фильтры качества изображений, которые отбраковывают слишком размытые, перегруженные или содержащие недостаточно информации для идентификации вида.

2.1.2 INaturalist

Приложение iNaturalist помогает идентифицировать растения и животных, одновременно собирая данные для научных целей и охраны природы. Иллюстрация страницы выбора фотографии представлена на Рисунок 3 - страница выбора фотографии в приложении INaturalist.

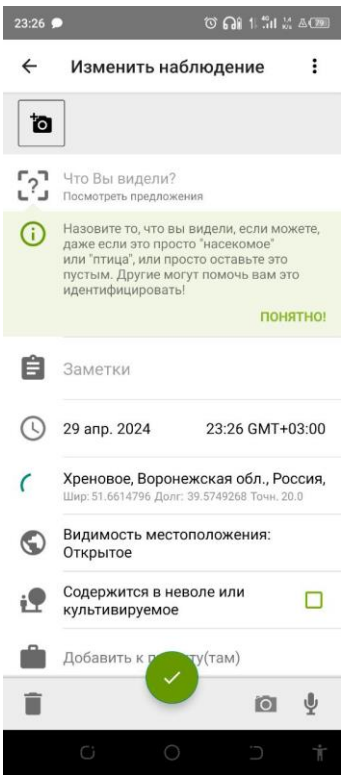

Рисунок 3 - страница выбора фотографии в приложении INaturalist

Это также краудсорсинговая система идентификации видов и инструмент для регистрации встречаемости организмов. С ее помощью вы можете записывать свои собственные наблюдения, получать помощь в идентификации, сотрудничать с другими для сбора такого рода информации для общей цели или получать доступ к данным наблюдений, собранным пользователями. Иллюстрация страницы поиска по названию приложения представлена на Рисунок 4 - страница поиска по названию в приложеии INaturalist.

Рисунок 4 - страница поиска по названию в приложеии INaturalist

2.1.3 PlantSnap

PlantSnap – приложение с платной подпиской, предполагающее 5 бесплатных распознаваний. Иллюстрация страницы выбора фотографии представлена на Рисунок 5 - страница камеры в приложении PlantSnap.

Рисунок 5 - страница камеры в приложении PlantSnap

В базе данных с возможностью поиска PlantSnap содержится более 650 000 растений, и они переведены на 37 языков.

В PlantSnap удалось создать систему, которая позволяет загружать фотографию и получать подробную информацию о сфотографированном растении без участия человека. Кроме того, в iOS-версии PlantSnap используется новая технология, называемая автоопределением и дополненной реальностью. Функция автоматического распознавания на самом деле подсказывает вам, когда нужно сделать снимок, чтобы каждый раз получать идеальное изображение.

Рисунок 6 - страница поиска по названию в приложении PlantSnap

2.2 Моделирование системы

2.2.1 Диаграмма прецедентов

Рассмотрим полную диаграмму для использования приложения разными типами пользователей. В данном случае необходимость составления диаграммы прецедентов продиктована прежде всего тем, что use-case диаграмма — это инструмент для моделирования системы и понимания ее

функциональности и потребностей пользователей. Они помогают в определении основных действий, которые пользователь должен совершить в системе, чтобы достичь определенных целей. Они также позволяют определить возможные риски и проблемы, которые могут возникнуть в ходе использования системы. Данная диаграмма представлена на Рисунок 7.

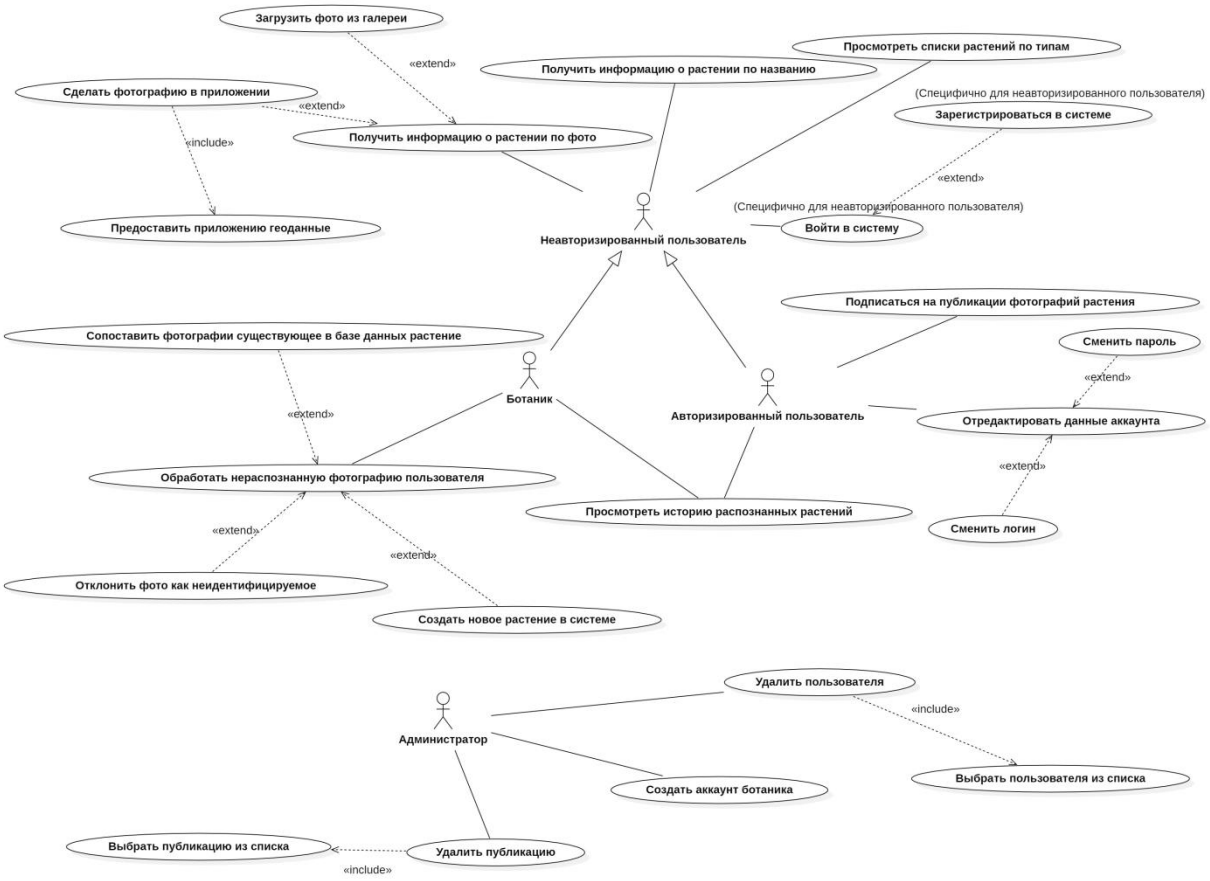

Рисунок 7 - диаграмма прецедентов

2.2.2 Диаграммы последовательности

Диаграмма последовательности является важным инструментом для проекта, который помогает более глубоко понимать процесс, улучшать его эффективность и упрощать взаимодействие. Данная диаграмма представлена на Рисунок 8 и Рисунок 9.

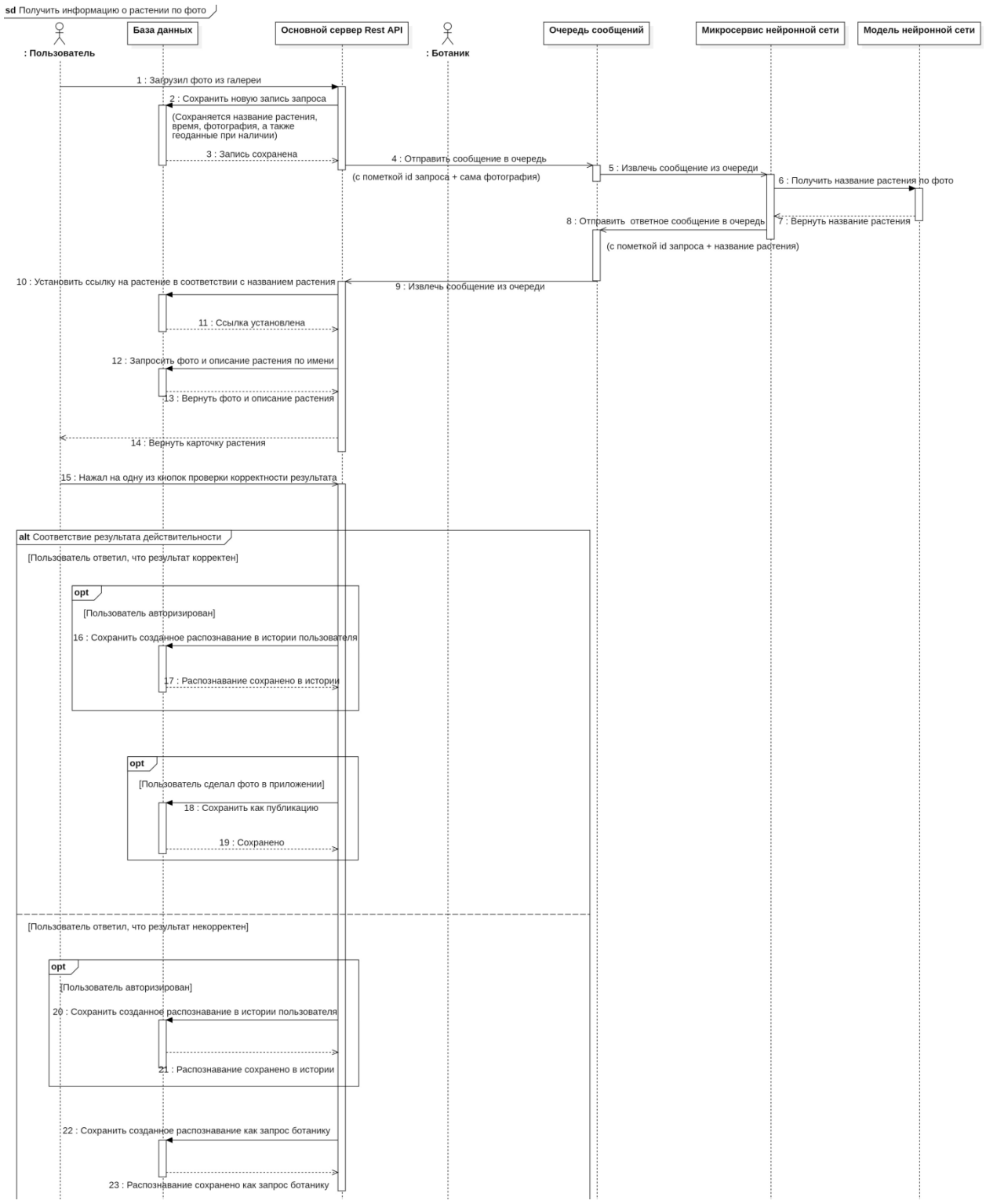

Рисунок 8 - диаграмма последовательности для процесса распознавания растения по фотографии

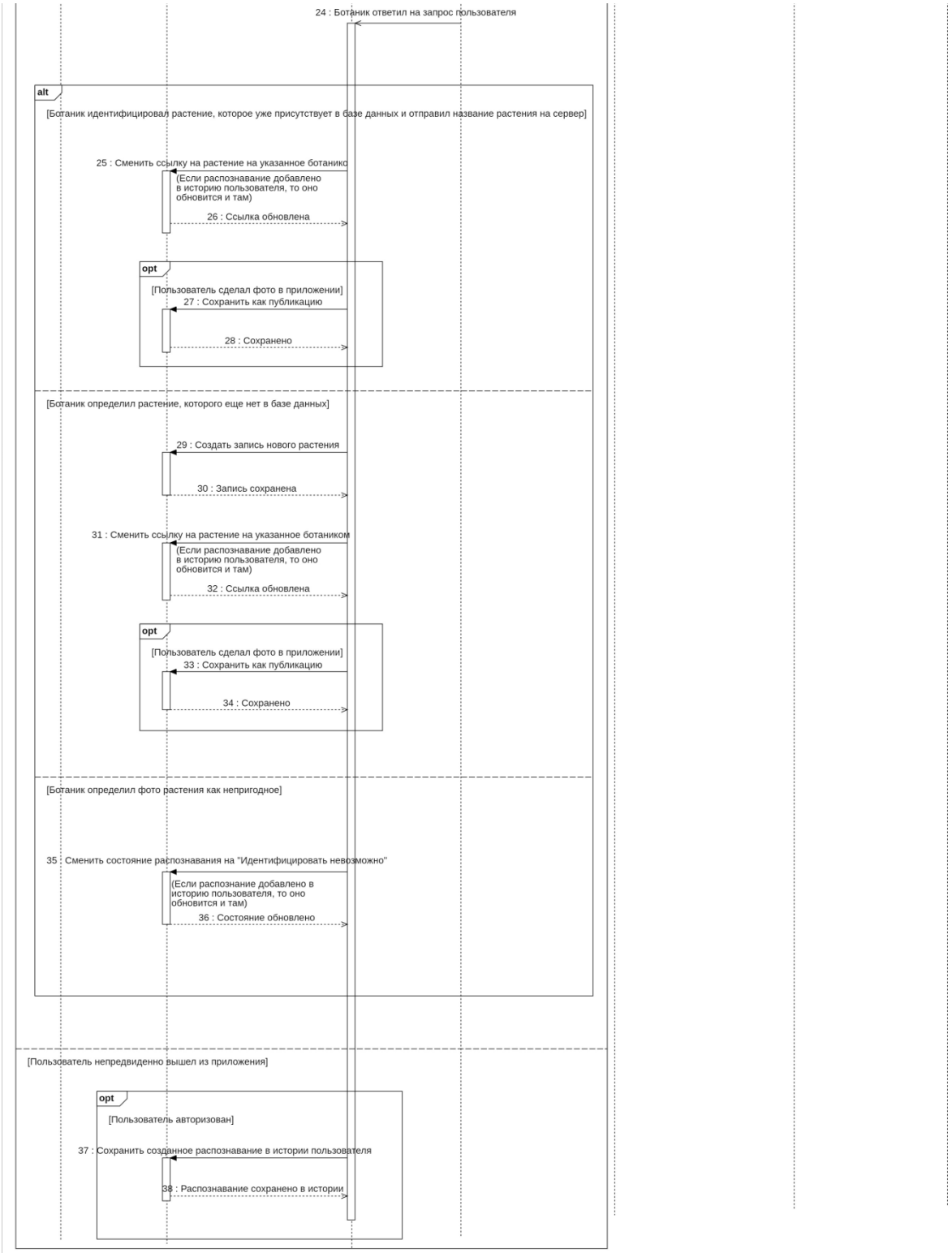

Рисунок 9 - продолжение диаграммы последовательности для процесса распознавания растения по фотографии

2.2.3 Диаграмма развертывания

Диаграмма развертывания позволяет определить требования к аппаратному обеспечению, планировать установку и настройку компонентов системы, а также оценивать ее производительность и масштабируемость. Данная диаграмма представлена на Рисунок 10.

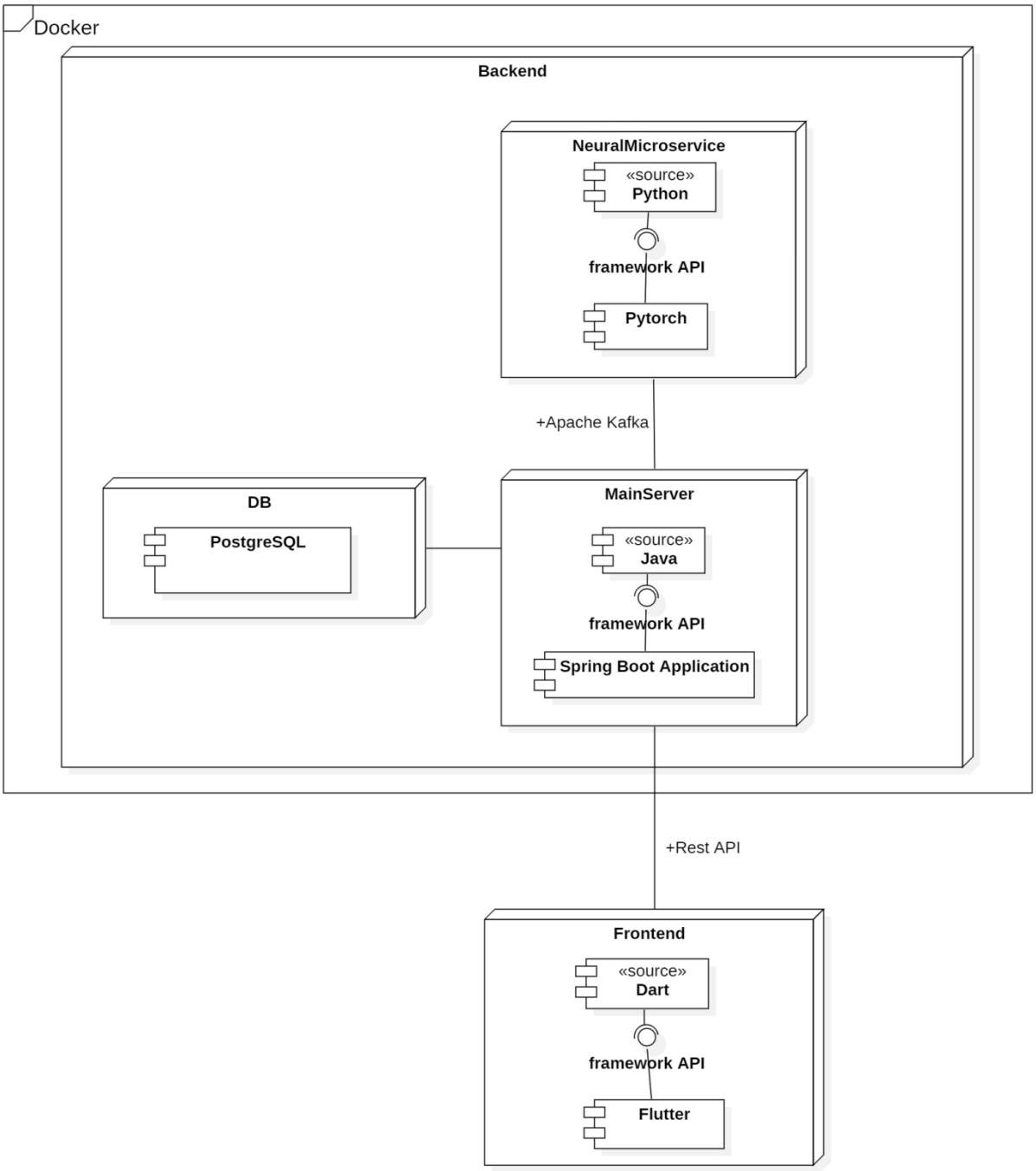

Рисунок 10 - диаграмма развертывания

2.2.4 Диаграмма состояния

Диаграмма состояния позволяет определить возможные сценарии поведения системы, выделить ключевые состояния и переходы между ними, а также оценить ее надежность и устойчивость к ошибкам. Для данного проекта были спроектированы 2 диаграммы, представленные на Рисунок 11 и Рисунок 12.

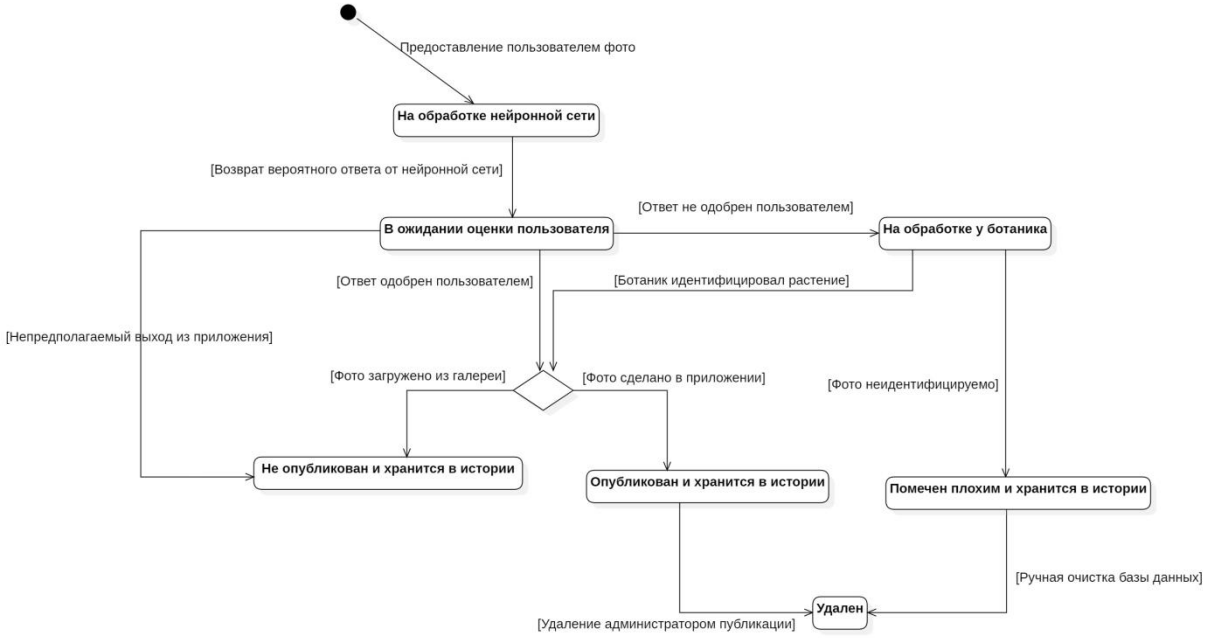

Рисунок 11 - диаграмма состояния запроса обработки фотографии

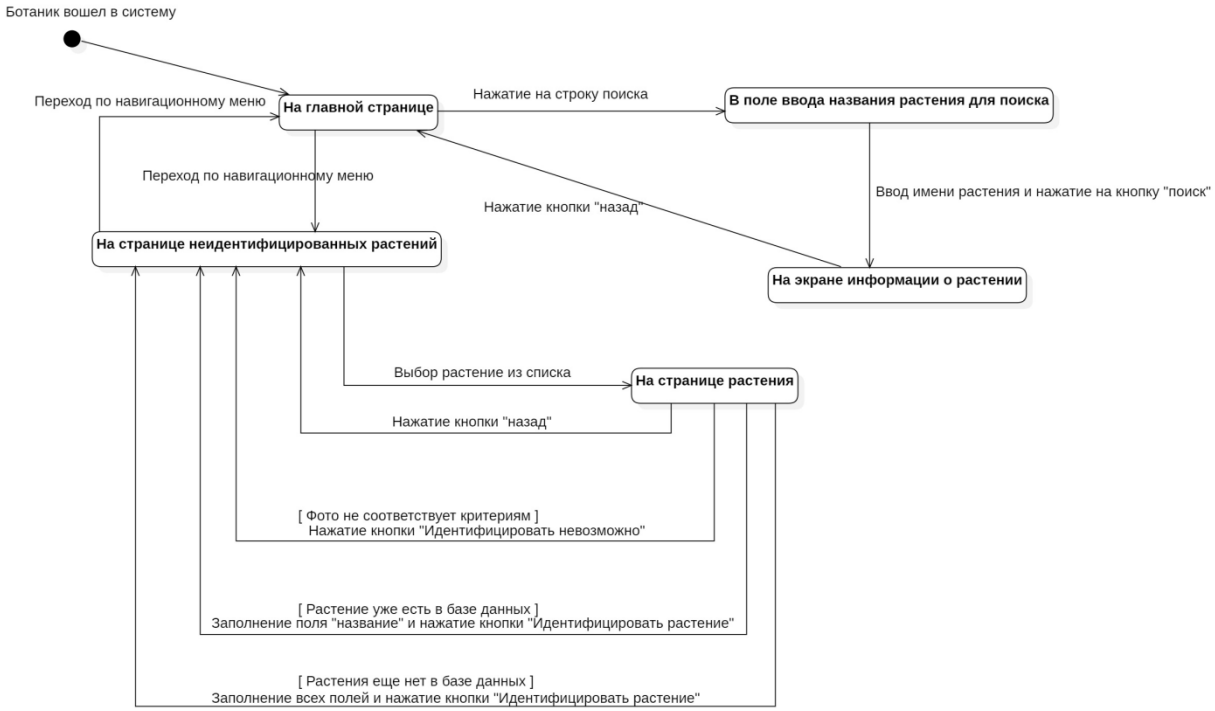

Рисунок 12 - диаграмма состояния процесса идентификации растения ботаником

2.2.5 Er-диаграмма

Er-диаграмма демонстрирует отношения сущностей в базе данных. Данная диаграмма представлена на Рисунок 13**.**

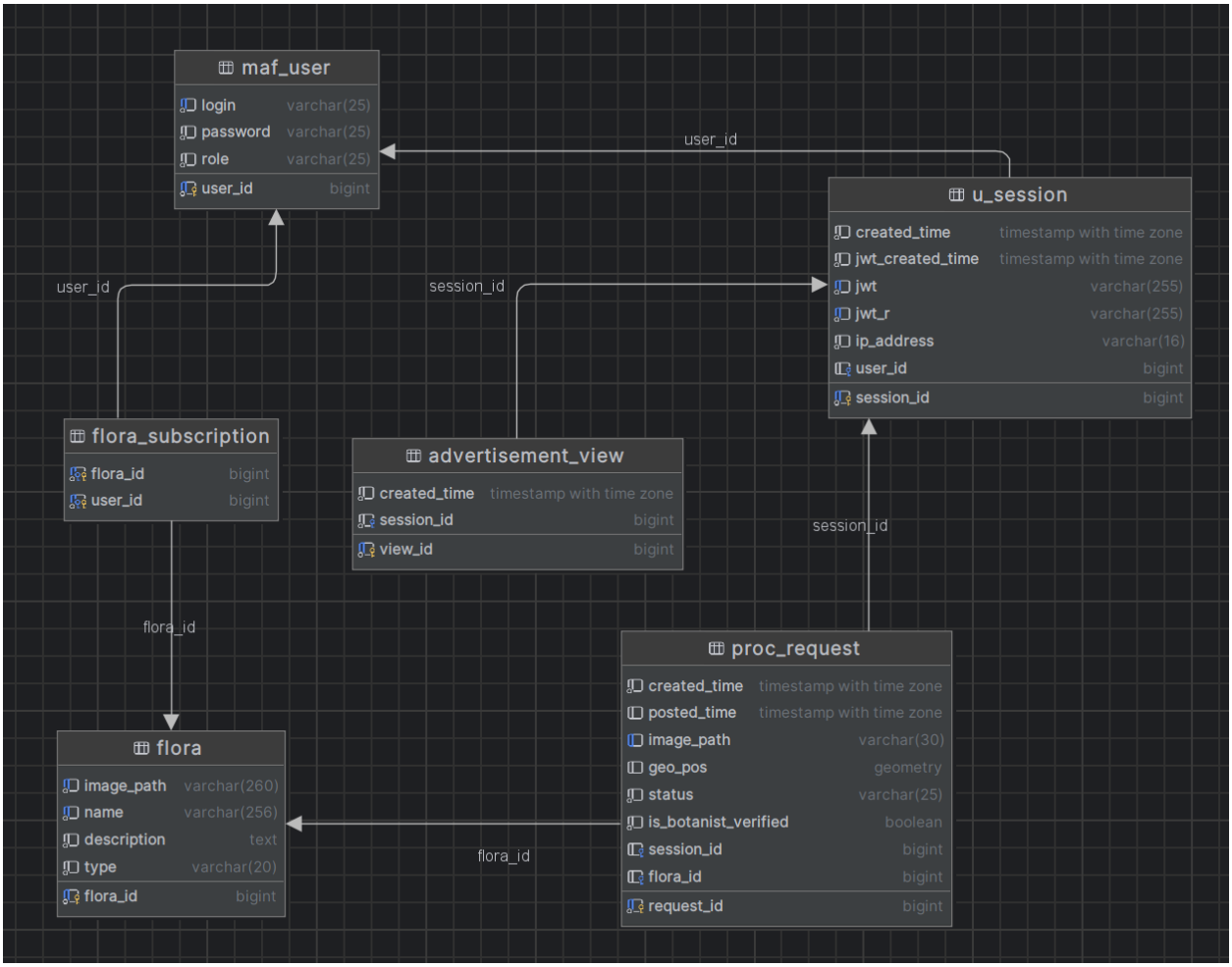

Рисунок 13 - er-диаграмма

2.2.6 Функциональная схема

Функциональная схема демонстрирует возможные сценарии использования на различных страницах. Данная диаграмма представлена на Рисунок 14.

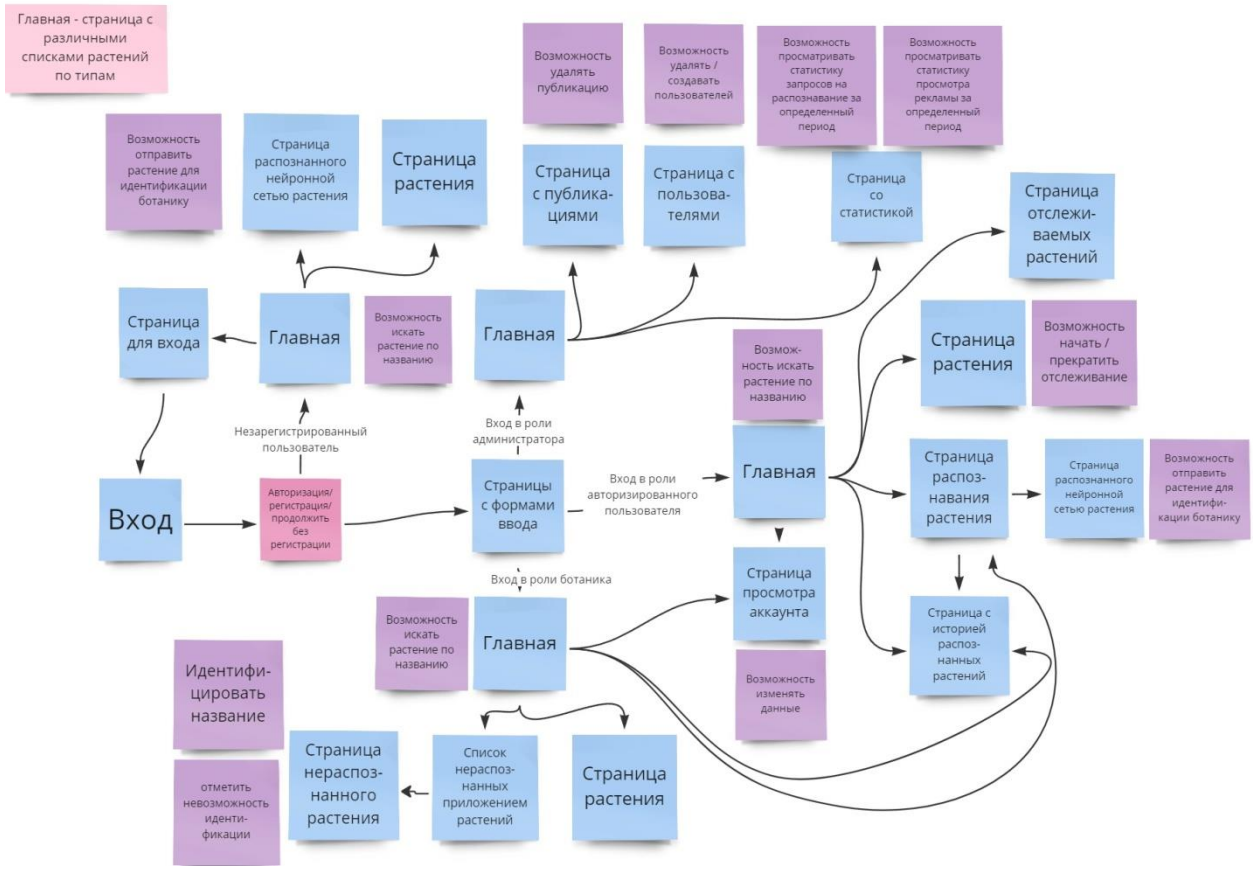

Рисунок 14 - функциональная схема приложения

3 Реализация

3.1 Средства реализации

Мобильное приложение имеет архитектуру, соответствующую смешанной модели Клиент - Серверного взаимодействия на основе REST API и взаимодействия между сервером и микросервисом по классификации растений на основе очереди сообщений.

Для реализации серверной части приложения будут использоваться следующие средства:

— фреймворк Spring с модулем Spring Boot

Выбор такого решения основан на наличии большого количества модулей, предоставляющих простой интерфейс для разработчика и позволяющих существенно сократить время разработки, а также возможностями фреймворка по работе с различными моделями взаимодействия элементов системы.

— язык программирования Java

Важным преимуществом в рамках системы языка Java является его высокая надежность вследствие строгой статической типизации, которая позволит наиболее корректно работать со сложной структурой базы данных. Также Java является кроссплатформенным и производительным языком.

— СУБД PostgreSQL

Данная СУБД является свободно распространяемой и предоставляет функционал аналогичный платным конкурентам [1]. Также PostgreSQL имеет в своей функциональности расширение PostGIS, предоставляющее возможность индексации геометрических объектов, что является важным в рамках разрабатываемого приложения.

— Docker

Контейниризатор позволит быстрее и надежнее масштабировать приложения в рамках системы, упаковывая их в отдельные блоки.

Для реализации микросервиса по классификации растений будут

использоваться следующие средства:

— язык программирования Python

Простой и понятный синтаксис этого языка, а также наличие множества библиотек для машинного обучения и анализа данных делает его предпочтительным выбором для разработки и обучения модели нейронной сети.

— фреймворк Pytorch.

Был выбран благодаря простоте использования, а также возможности обучения моделей на различных устройствах, таких как CPU и GPU. PyTorch включает в себя готовые модели, что облегчает и ускоряет процесс создания и настройки сложных архитектур глубокого обучения [2].

В качестве очереди сообщений между нейросетевым микросервисом и сервером будет использовано следующее средство:

— Apache Kafka

Данная технология была выбрана, так как является масштабируемой, отказоустойчивой и гибкой системой, которая позволяет обрабатывать большой поток данных и обеспечивать сохранность информации.

Для реализации клиентской части приложения будут использоваться следующие средства:

— язык программирования Dart;

— фреймворк Flutter.

Данный стек технологий был выбран, так как Flutter имеет одинаковый пользовательский интерфейс и бизнес-логику для всех платформ, позволяет сократить время разработки кода, а также есть возможность использовать плагин от Google для получения координат GPS, обработки разрешений и др.

3.2 Логика приложения

3.2.1 Разработка микросервиса по классификации растений

В данной работе разработан сервис для обработки фотографий растений, основанный на применении модели нейронной сети. Для реализации сервиса был выбран язык программирования Python и библиотека PyTorch, широко известная своими мощными возможностями для создания и обучения нейронных сетей.

Среди множества моделей, представленных в PyTorch, предпочтение было отдано ResNet-50. Эта модель славится своей высокой точностью и сравнительно небольшим размером, что делает её особенно привлекательной для приложений, не требующих значительных вычислительных ресурсов [3]. Кроме того, ResNet-50 обладает превосходной репутацией благодаря своей способности обеспечивать высокую производительность при обработке изображений, которая обеспечивается за счет того, что в отличии от традиционной нейронной сети, которая для входа x и желаемого выхода $H(x)$ пытается аппроксимировать функцию $H(x)$, аппроксимирует функцию $F(x) = H(x) - x$, и итоговое преобразование записывается как:

$$H(x) = F(x) + x$$

Эта простая модификация позволяет легче обучать очень глубокие сети, поскольку градиенты могут легче проходить через сеть, улучшая распространение ошибок в обратном направлении.

Остаточный блок обычно включает два или три сверточных слоя с batch normalization и ReLU активацией. Представлен на Рисунок 15 Пример двухслойного остаточного блока:

$$y = ReLU(x + F(x, \{W_i\}))$$

где $F(x, \{W_i\}$ – это последовательность операций (свертка, нормализация, активация) над входом x с параметрами W_i.

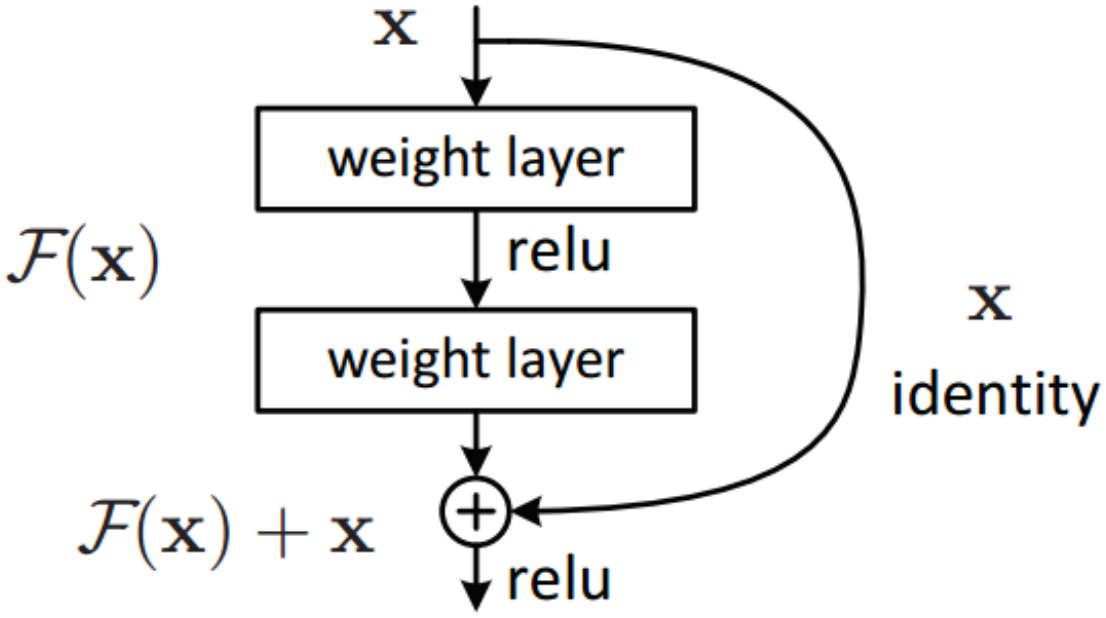

Рисунок 15 - представление Residual block

ResNet-50 состоит из 50 слоев, разделенных на 5 блоков, каждый из которых содержит набор остаточных блоков. Остаточные блоки позволяют сохранить информацию из более ранних слоев, что помогает сети лучше изучить представление входных данных. Архитектура этой нейронной сети представлена на Рисунок 16. Она основывается на следующих компонентах:

1 Сверточные слои. Несколько начальных сверточных слоев обрабатывают входной тензор для выделения начальных признаков. Эти слои включают свертку (Convolution) и нормализацию (Batch Normalization):

$$X_1 = ReLU\big(Conv\big(X_{0,}W_1\big) + b_1\big);$$

$$X_2 = BatchNorm(X_1);$$

где $Conv$ – операция свертки, W_1 и b_1 — параметры слоя (веса и смещения), а $ReLU$ — функция активации ReLU.

Этот процесс может повторяться для нескольких сверточных слоев, чтобы выделить начальные признаки из входного тензора.

2 Остаточные блоки. Глубокие остаточные блоки обрабатывают данные, извлекая сложные иерархические признаки. Каждый остаточный блок состоит из двух или трех сверточных слоев с batch normalization и ReLU активацией.

Для двухслойного остаточного блока:

$$F(X_l) = ReLU(BatchNorm(Conv(X_l, W_{l1}) + b_{l1}));$$

$$F(X_l) = BatchNorm(Conv(F(X_l), W_{l2}) + b_{l2});$$

$$X_{l+1} = ReLU\big(X_l + F(X_l)\big)$$

где X_l — входной тензор на уровне l, , W_{l1} и W_{l2}— веса сверточных слоев, b_{l1} и b_{l2}— смещения.

3 Полносвязные блоки. Они играют ключевую роль в преобразовании извлеченных признаков в окончательные классы .Сверточный слой, за которым следует softmax активация, чтобы получить вероятности действий.

$$P = softmax\big(Conv\big(X_l, W_p\big) + b_p\big)$$

где X_l — входной тензор после последнего остаточного блока, W_p и b_p— параметры сверточного слоя.

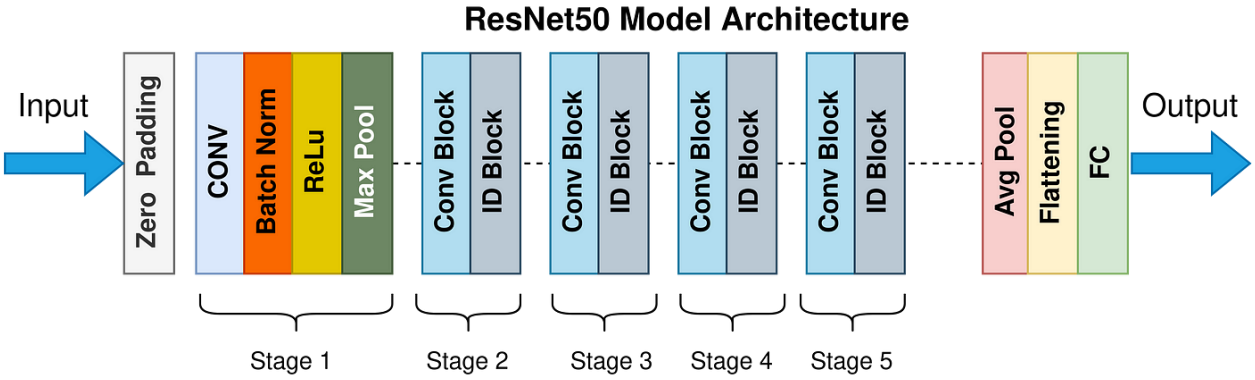

Рисунок 16 - архитектура модели ResNet-50

Для обучения модели потребовалось собрать обширный и разнообразный датасет, содержащий изображения различных видов растений. Используя ресурсы таких платформ, как Kaggle и GitHub, удалось создать набор данных, включающий почти 60 тысяч фотографий, разделённых на 81 класс растений. Для улучшения качества модели и увеличения объёма данных перед обучением была применена аугментация данных. Этот процесс включал такие техники, как вращение, изменение масштаба, сдвиг, искажений и другие преобразования изображений, что позволило существенно увеличить разнообразие обучающей выборки.

Процесс обучения модели проводился с использованием метода обучения с учителем. Первые 10 эпох модель обучалась с замороженными параметрами, что позволило ей изучить базовые признаки и установить стабильность. На вторых 10 эпохах параметры модели размораживались, и происходила более глубокая настройка всех слоев модели для достижения более высокой точности и общей производительности. Этот подход позволил модели эффективно изучать данные и адаптироваться к сложным паттернам в данных, обеспечивая при этом оптимальное качество результатов. После размораживания всех параметров точность предсказаний резко возросла, что отражено на Рисунок 17, а доля ошибки почти прямо пропорционально достигла минимума, что представлено на Рисунок 18.

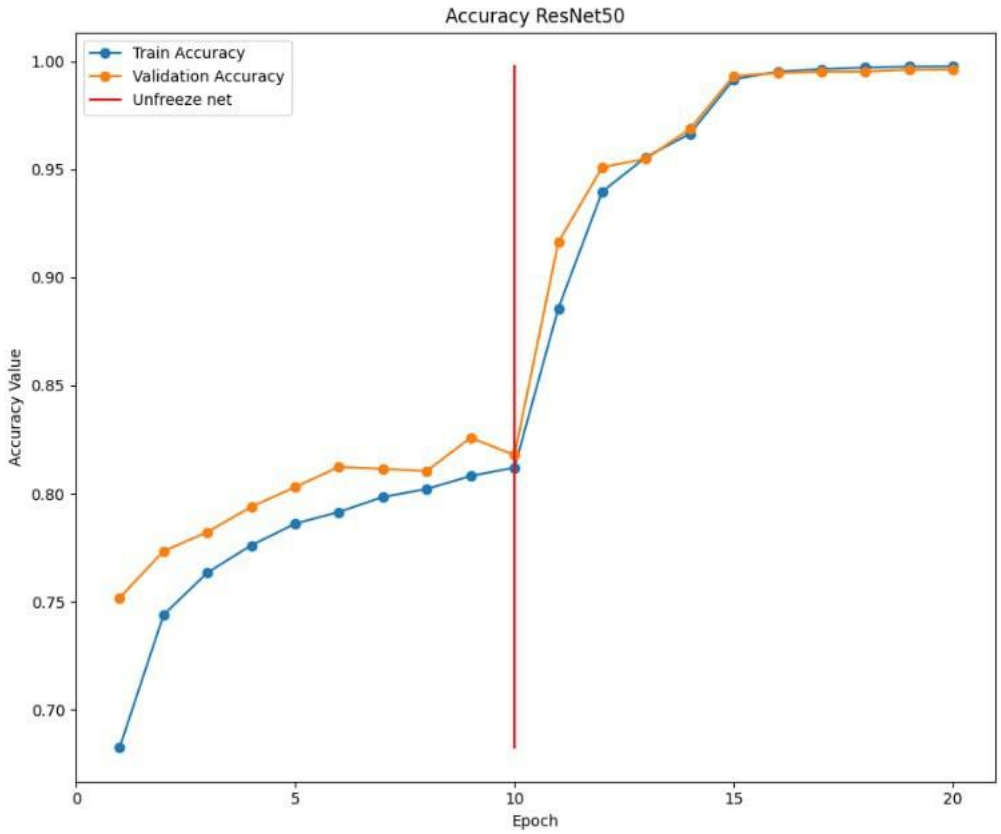

Рисунок 17 - увеличение точности в процессе обучения

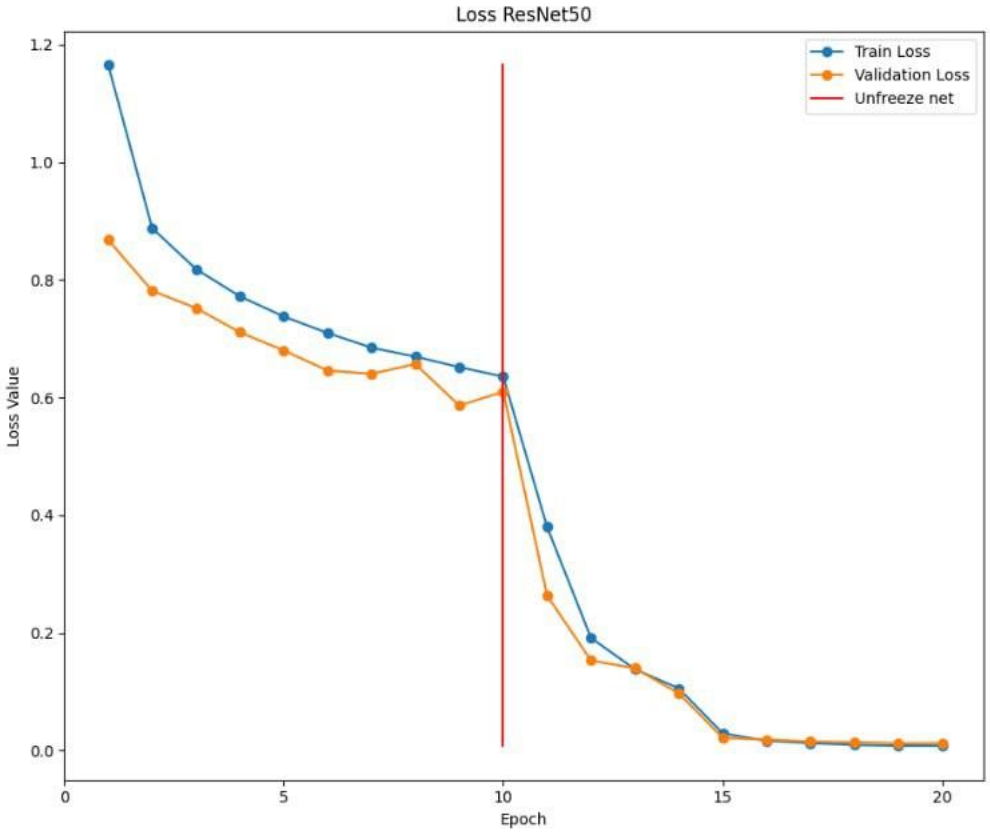

Рисунок 18 - уменьшение доли ошибки в процессе обучения

После успешного обучения нейронной сети для классификации растений был проведен анализ результатов с использованием матрицы ошибок (Confusion Matrix), представленной на Рисунок 19, используя тестовый набор данных растений, которые не использовались при обучении. В результате анализа выявлено, что модель иногда ошибочно относит Клюкву и Барбарис к Бруснике; также наблюдаются ошибки при различении Гвоздики и Пиона, Георгина и Хризантемы, а также Нектарина и Манго.

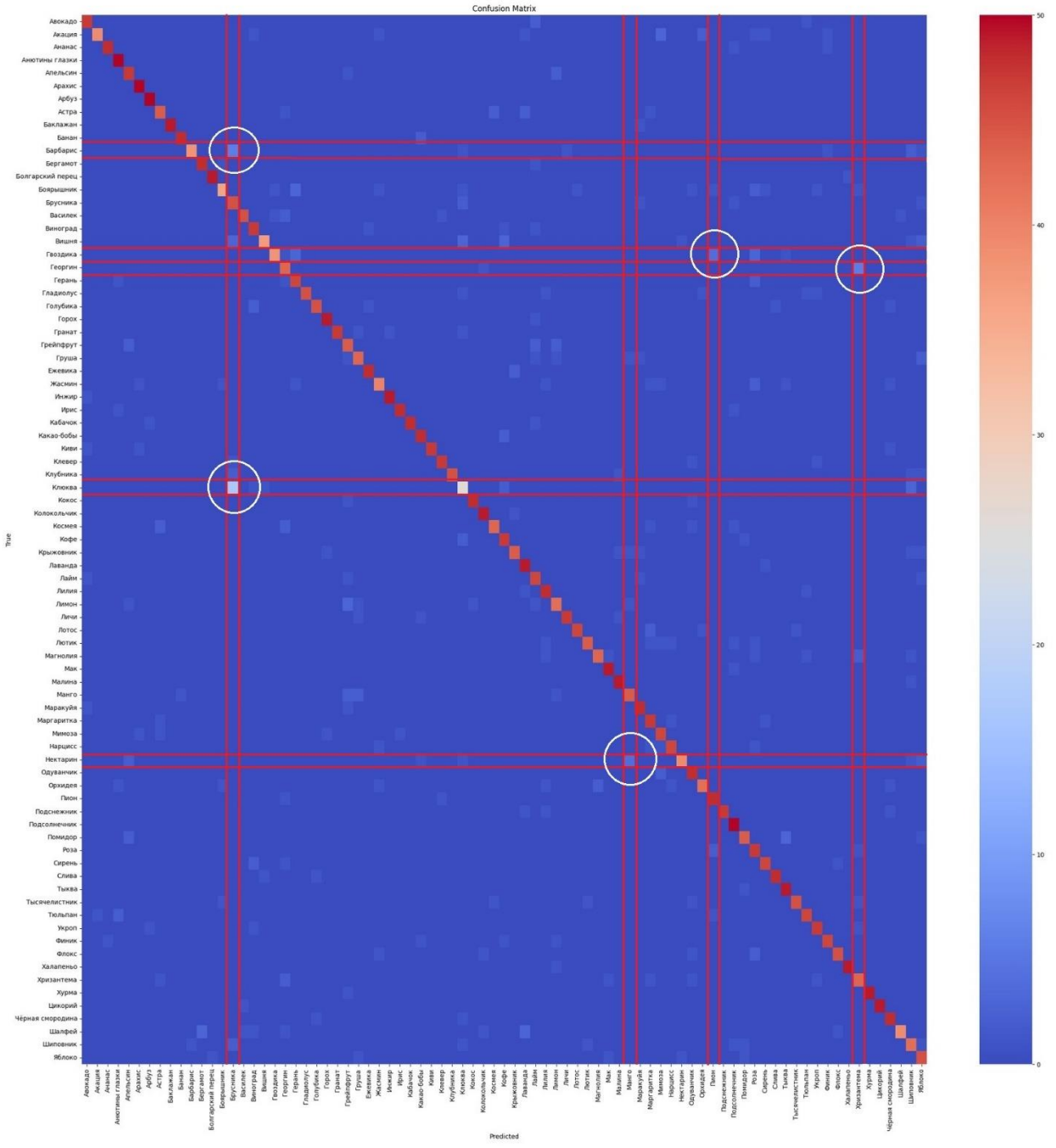

Рисунок 19 - матрица ошибок

Приём и передача сообщений в сервисе были реализованы с использованием Apache Kafka. Выбор Kafka обусловлен её преимуществами в обработке потоковых данных: высокой производительностью, надёжностью, масштабируемостью и гибкостью в настройке. Kafka позволяет эффективно обрабатывать большие объёмы данных в реальном времени, что является критически важным для сервиса, работающего с большим количеством изображений.

После завершения обучения модель была протестирована на отдельной тестовой выборке, что позволило оценить её эффективность. Средний уровень точности модели на тестовых данных составил 90%, что свидетельствует о высокой способности нейронной сети к правильной идентификации различных видов растений.

3.2.2 Backend разработка

Приложение построено на основе классического многоуровневого архитектурного паттерна, использующего jpa и состоящего из следующих слоев:

— Controller – слой контроллеров отвечает за прием Rest запросов клиента, передачу информации на обработку слою сервисов приложения, обработка исключений, а также возврат соответствующего ответа пользователю

— Service – сервисы представляют собой всю бизнес логику приложения. Здесь происходит валидация данных, их обработка, использование уровня репозиториев для доступа к базе данных, а также порождение исключений в процессе работы приложения.

— Repository – слой jpa репозиториев позволяет на очень высоком уровне абстракции работать с базой данных. С помощью него происходит работа с сущностями PostgreSQL [4].

Для работы со слоем контроллеров клиенту сначала необходимо авторизоваться в системе. Для этого используется авторизация на основе

JWT. Веб-токен JSON (JWT) - это открытый стандарт, который определяет компактный и автономный способ безопасной передачи информации между сторонами в виде объекта JSON [5]. Эта информация может быть проверена и ей можно доверять, поскольку она имеет цифровую подпись. Любой пользователь приложения для начала работы должен запросить случайно сгенерированный токен. Сделать это можно либо с помощью авторизации/регистрации в приложении, либо с помощью функции “Войти без регистрации”, которая позволяет работать пользователю, основываясь только на сессии в приложении.

JWT сохраняется в базе данных и возвращается пользователю, а после пользователь обязан прикладывать этот токен в header запроса, чтобы использовать функции приложения. Срок жизни токена ограничен, чтобы уменьшить вероятность кражи токена. В связи с этим применяется два вида токена: access и refresh. Если при получении запроса указан JWT, срок жизни которого уже истек, возвращается код ошибки 403.Данный код говорит о том, что доступ к запрашиваемой странице запрещен или у пользователя не прав на просмотр контента. После чего, при наличии refresh JWT, клиент отправляет запрос на обновление токенов, указывая в header запроса данный refresh JWT. Получив новую пару токенов, клиент отправляет изначальный запрос снова с новым access токеном.

3.2.3 Frontend разработка

При разработке клиентской части приложения применялась BLoC-архитектура. Блок-архитектура (BLoC) во Flutter приложениях является паттерном проектирования, который разделяет бизнес-логику приложения от пользовательского интерфейса [6]. Это помогает улучшить тестируемость, управляемость и разделение ответственности в приложении.

Основные компоненты BLoC-архитектуры:

— BLoCs - неизменяемые объекты, которые содержат состояние (данные) и логику для обновления этого состояния. Действия, такие как

нажатия кнопок или сетевые запросы, отправляются в блоки, которые затем обновляют свое состояние и уведомляют слушателей об изменениях.

— Events - неизменяемые сообщения, которые отправляются в блоки для инициирования изменений состояния.

— States - неизменяемые объекты, которые представляют текущее состояние приложения.

— Listeners компоненты пользовательского интерфейса, которые подписываются на изменения состояния и обновляют свой внешний вид в соответствии с изменениями.

Преимущества BLoC-архитектуры:

— Тестируемость. Блоки можно легко протестировать отдельно от пользовательского интерфейса, что упрощает обнаружение и исправление ошибок.

— Управляемость. Разделение бизнес-логики и пользовательского интерфейса улучшает управляемость кода и делает его проще для понимания и сопровождения.

— Разделение ответственности. BLoC-архитектура четко разделяет ответственность между различными компонентами приложения, что повышает его согласованность и надежность.

— Предсказуемость. Поскольку BLoC-архитектура управляет состоянием централизованным образом, изменения состояния приложения становятся более предсказуемыми и контролируемыми.

3.3 Реализация интерфейса

3.3.1 Общие экраны для всех пользователей приложения

3.3.1.1 Экран авторизации

Имеются следующие элементы экрана:

— форма для заполнения полей личными данными:

а) логин (не менее 6 символов);

b) пароль (не менее 6 символов);

— кнопка «Авторизоваться»;

— кликабельная ссылка «Зарегистрироваться»;

— кликабельная ссылка «Войти без регистрации».

Поля, которые могут появиться при взаимодействии с экраном:

— поле для вывода информации об отсутствии в базе данных пользователя с введенными данными;

— поле для вывода информции об ограниченном функционале приложения для незарегистрированных пользователей.

Демонстрация внешнего вида экрана представлена на Рисунок 20 – экран авторизации

Компоновка и логика заключается в том, что этот экран необходим для осуществления входа пользователя в систему.

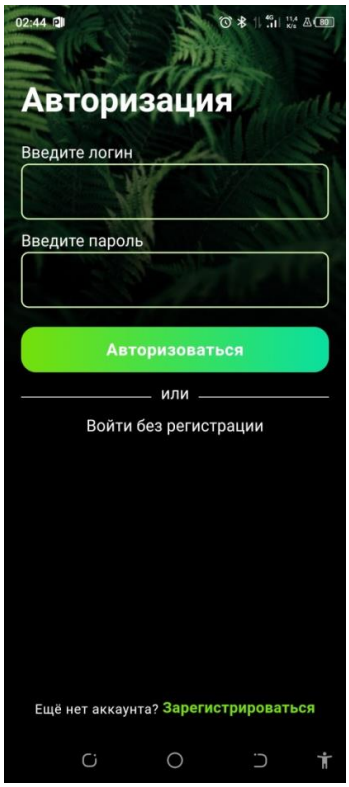

Рисунок 20 – экран авторизации

3.3.1.2 Экран регистрации

Имеются следующие элементы экрана:

— форма для заполнения полей личными данными:

a) логин (не менее 6 символов, не должен уже находиться в базе данных);

b) пароль (не менее 6 символов);

c) повторите пароль (должен совпадать с полем Пароль);

— кликабельная ссылка «Пользовательское соглашение»;

— чек-бокс для проверки обязательного ознакомления с «Пользовательским соглашением»;

— кнопка «Зарегистрироваться»;

— кликабельная ссылка «Авторизоваться»;

— кликабельна ссылка «Войти без регистрации».

Поля, которые могут появиться при взаимодействии с экраном:

— поле с текстом «Пользовательского соглашения»;

— кнопка «Согласен» в поле с текстом «Пользовательского соглашения»;

— поле для вывода информации об ограниченном функционале приложения для незарегистрированных пользователей.

Демонстрация внешнего вида экрана представлена на Рисунок 21 - экран регистрации

Компоновка и логика заключается в том, что этот экран необходим для осуществления регистрации пользователя в системе.

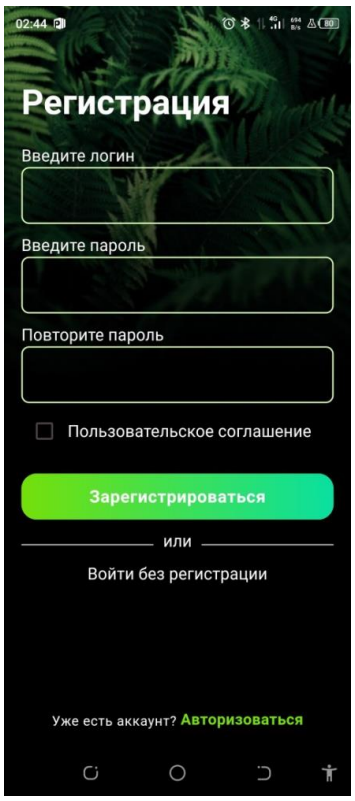

Рисунок 21 - экран регистрации

3.3.1.3 Главный экран

Имеются следующие элементы экрана:

— навигационная панель в нижней части экрана;

— поле для текстового поиска по названию растения;

— несколько горизонтальных списков, состоящих из карточек растений, сгруппированных по их типу:

a) цветок;

b) дерево;

c) трава;

d) мох;

— карточка растения:

a) фотография растения;

b) название.

Поля, которые могут появиться при взаимодействии с экраном:

— поле для вывода информации об отсутствии в базе данных

растения, с указанным в поиске названием.

Демонстрация внешнего вида экрана представлена на Рисунок 22 - главный экран

Компоновка и логика заключается в том, что этот экран предоставляет пользователю доступ к информации о различных растениях, а также предоставляет возможность поиска по названию.

Рисунок 22 - главный экран

3.3.1.4 Экран информации о растении

Имеются следующие элементы экрана:

— фото растения;

— текстовое поле с названием растения;

— текстовое поле с описанием растения;

— кнопка «Назад» в левом верхнем углу экрана;

— кнопка «Отслеживать» в правом верхнем углу экрана;

— навигационная панель внизу экрана.

Поля, которые могут появиться при взаимодействии с экраном:

— поле для вывода информации об ограниченном функционале приложения для незарегистрированных пользователей;

— поле для ввода информации об успешности работы нейронной сети:

a) текстовое поле: «Согласны ли вы с работой нейронной сети?»;

b) кнопка «Согласен»;

c) кнопка «Не согласен».

Демонстрация внешнего вида экрана представлена на Рисунок 23 – Экран информации о растении

Компоновка и логика заключается в том, что этот экран предоставляет пользователю информацию о запрошенном растении.

Рисунок 23 – Экран информации о растении

3.3.1.5 Экран камеры

Имеются следующие элементы экрана:

— экран съёмки камеры;

— кнопка «Назад» в левом верхнем углу экрана;

— текст: «Фото, сделанное через приложение, будет видно другим пользователям» вверху экрана;

— кнопка «Сфотографировать» внизу экрана посередине;

— кнопка «Выбрать из галереи» внизу экрана справа от кнопки «Сфотографировать»;

Поля, которые могут появиться при взаимодействии с экраном:

— поле с запросом на доступ к камере;

— поле с запросом на доступ к определению геолокации (координат);

— поле с предложением посмотреть рекламу.

Компоновка и логика заключается в том, что пользователь выбирает способ предоставления фотографии для распознавания растения.

3.3.2 Общие экраны для зарегистрированных пользователей

3.3.2.1 Экран профиля пользователя

Имеются следующие элементы экрана:

— навигационная панель внизу экрана;

— форма с информацией о пользователе;

— форма с информацией о приложении;

— кнопка «Редактировать профиль»;

— кнопка «Выйти.

Поля, которые могут появиться при взаимодействии с экраном:

— кнопка «Сохранить изменения»;

— кнопка «Удалить изменения»;

— поле с информацией об успешном изменении личных данных;

— поле с информацией о неудачном изменении личных данных.

Демонстрация внешнего вида экрана представлена на Рисунок 24 - экран профиля пользователя

Компоновка и логика заключается в том, что этот экран содержит

информацию о пользователе и его правах, полезные ссылки, а также предоставляет пользователю возможность выйти из аккаунта.

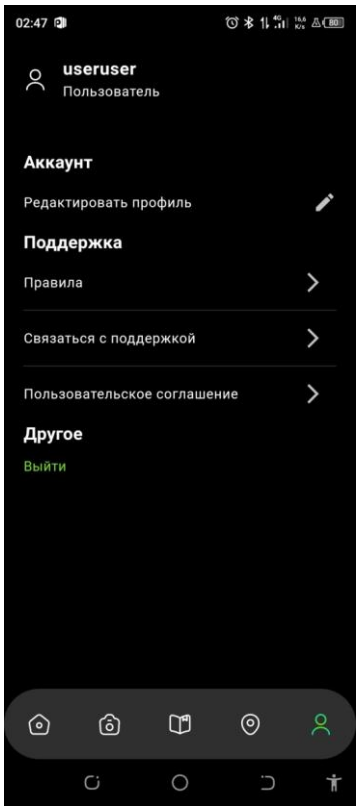

Рисунок 24 - экран профиля пользователя

3.3.2.2 Экран истории загруженных растений

Имеются следующие элементы экрана:

— название экрана «История»;

— навигационная панель внизу экрана;

— вертикальный список из карточек загруженных растений;

— карточка загруженного пользователем растения:

a) фотография растения, предоставленная пользователем для распознавания;

b) текстовое поле с названием для распознанного растения или текста «Название неизвестно» для нераспознанного;

c) геолокация (координаты), если фото было сделано через приложение;

d) дата предоставления фотографии.

Демонстрация внешнего вида экрана представлена на Рисунок 25 - экран истории загруженных растений

Компоновка и логика заключается в том, что этот экран предоставляет информацию пользователю о распознанных им ранее растениях.

Рисунок 25 - экран истории загруженных растений

3.3.2.3 Экран отслеживаемых растений

Имеются следующие элементы экрана:

— название экрана «Отслеживаемые растения»;

— навигационная панель внизу экрана;

— вертикальный список карточек отслеживаемых растений;

— карточка отслеживаемого пользователем растения:

a) фотография растения;

b) название;

c) геолокация (координаты) сделанной фотографии;

d) дата предоставления фотографии.

Демонстрация внешнего вида экрана представлена на Рисунок 26 - экран отслеживаемых растений

Компоновка и логика заключается в том, что этот экран предоставляет информацию пользователю об отслеживаемых им растениях.

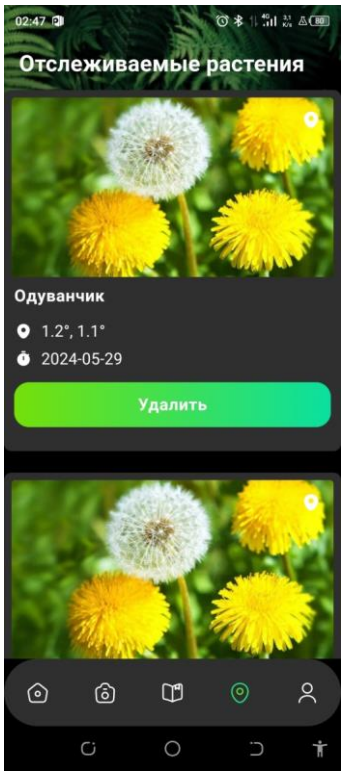

Рисунок 26 - экран отслеживаемых растений

3.3.3 Экраны для ботаников

3.3.3.1 Экран со списком нераспознанных растений

Имеются следующие элементы экрана:

— название экрана «Неидентифицированные растения»;

— навигационная панель внизу экрана;

— вертикальный список карточек нераспознанных растений;

— карточка нераспознанного приложением растения:

a) фотография растения;

b) текст «Неизвестно»;

c) геолокация (координаты);

d) дата предоставления фотографии.

Демонстрация внешнего вида экрана представлена на Рисунок 27 - экран со списком нераспознанных растений

Компоновка и логика заключается в том, что этот экран предоставляет ботанику список нераспознанных приложением растений.

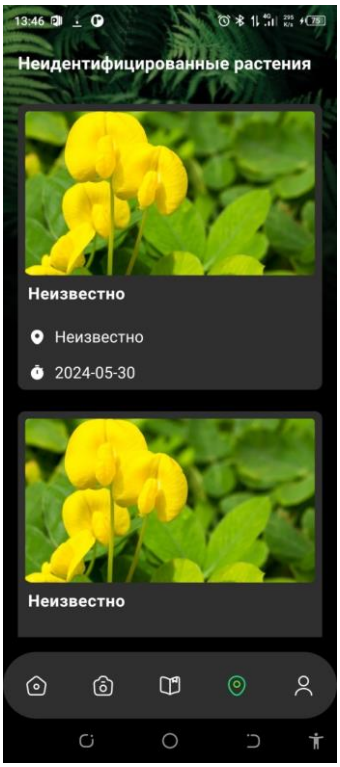

Рисунок 27 - экран со списком нераспознанных растений

3.3.3.2 Экран нераспознанного растения для ботаника

Имеются следующие элементы экрана:

— кнопка «Назад» в левом верхнем углу экрана;

— текстовое поле для ввода названия растения;

— текстовое поле для ввода названия типа растения;

— текстовое поле для ввода описания растения;

— кнопка «Идентифицировать растение»;

— кнопка «Идентифицировать невозможно»;

— навигационная панель внизу экрана;

— карточка нераспознанного приложением растения:

а) фотография растения;

b) текст «Неизвестно»;

c) геолокация (координаты);

d) дата предоставления фотографии.

Демонстрация внешнего вида экрана представлена на Рисунок 28 - экран нераспознанного растения

Компоновка и логика заключается в том, что этот экран предоставляет ботанику данные о нераспознанном растении, а также возможность идентифицировать растение.

Рисунок 28 - экран нераспознанного растения

3.3.4 Экраны для администратора

3.3.4.1 Экран со списком пользователей

Имеются следующие элементы экрана:

— название раздела «Пользователи»;

— кнопка «Добавить» в правом низу экрана;

— навигационная панель внизу экрана;

— карточки пользователей, включающие:

a) логин;

b) роль;

c) кнопка «Удалить».

Демонстрация внешнего вида экрана представлена на Рисунок 29 - экран со списком пользователей

Компоновка и логика заключается в том, что этот экран предоставляет администратору список пользователей и их ролей.

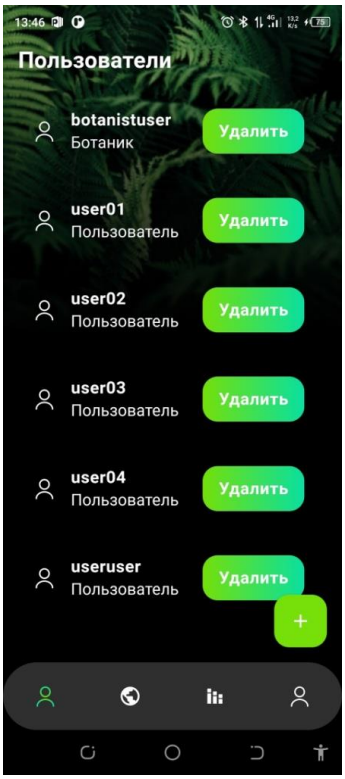

Рисунок 29 - экран со списком пользователей

3.3.4.2 Экран со списком публикаций

Имеются следующие элементы экрана:

— название раздела «Публикации фотографий»;

— навигационная панель внизу экрана;

— вертикальный список карточек опубликованных пользовательских фотографий растений;

— карточка опубликованной пользовательской фотографии

растения:

a) фотография растения;

b) название;

c) геолокация (координаты);

d) дата предоставления фотографии;

e) кнопка «Удалить».

Демонстрация внешнего вида экрана представлена на Рисунок 30 - экран со списком публикаций

Компоновка и логика заключается в том, что этот экран предоставляет администратору список публикаций пользовательских фотографий.

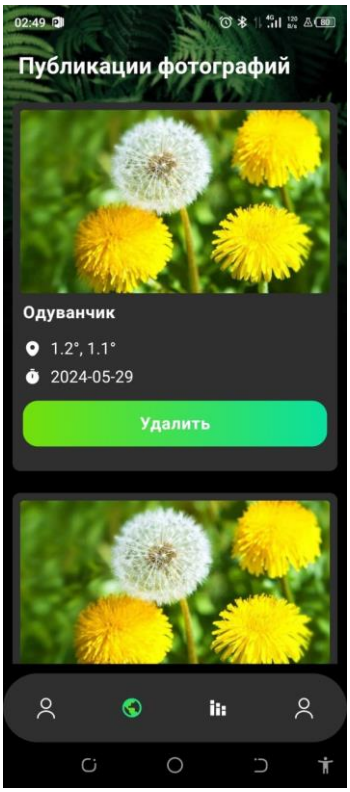

Рисунок 30 - экран со списком публикаций

3.3.4.3 Экран со статистическими данными приложения

Имеются следующие элементы экрана:

— название раздела «Статистика приложения»;

— календарь для выбора даты начала отсчета статистики;

— календарь для выбора даты окончания отсчета статистики;

— кнопка «Показать статистику распознавания»;

— кнопка «Показать статистику просмотра рекламы»;

— навигационная панель внизу экрана.

Поля, которые могут появиться при взаимодействии с экраном:

— Поле с информацией о некорректном вводе данных;

— таблица со столбцами:

a) «Дата»;

b) «Кол-во операций».

Демонстрация внешнего вида экрана представлена на Рисунок 31 - экран со статистическими данными приложения

Компоновка и логика заключается в том, что этот экран предоставляет администратору статистические данные.

Рисунок 31 - экран со статистическими данными приложения

Заключение

В ходе выполнения курсового проекта был проведен всесторонний анализ предметной области и изучены существующие аналогичные разработки. На основе полученной информации были сформулированы функциональные и нефункциональные требования к приложению, которые позволили заложить основу для его дальнейшей разработки.

Для визуализации будущего приложения были разработаны макеты интерфейса, которые отражают основные элементы дизайна и взаимодействия с пользователем. Был выбран подходящий стек технологий и платформа для разработки приложения, обеспечивающие наилучшую производительность и масштабируемость.

Для эффективного управления проектом и контроля версий был создан репозиторий GitHub, а также построены UML диаграммы, отражающие структуру и взаимосвязи элементов приложения.

В ходе разработки были реализованы основные функции приложения, которые позволяют пользователям фотографировать растения и получать информацию о них.

Разработанное мобильное приложение для распознавания растений имеет большой потенциал для различных применений, включая любительскую ботанику, исследования, сельское хозяйство, садоводство и образование. Приложение может помочь людям идентифицировать растения, узнать больше об их характеристиках и использовании, а также способствовать охране окружающей среды и научным исследованиям.

Список использованных источников

1. Что такое база данных | Oracle СНГ: [электронный ресурс] – URL: https://goo.su/sea4 (дата обращения: 15.05.2024). – Текст. : электронный.

2. PyTorch: [электронный ресурс] – URL: https://en.wikipedia.org/wiki/PyTorch (дата обращения: 25.05.2024). – Текст. : электронный.

3. ResNet(34, 50, 101) «остаточные» CNN для классификации изображений: [электронный ресурс] – URL: https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/ (дата обращения: 25.05.2024). – Текст. : электронный.

4. Многоуровневая архитектура в проекте на Java (Часть 1): [электронный ресурс] – URL: https://alexkosarev.name/2018/07/27/n-tier-java-part1/ (дата обращения: 25.05.2024). – Текст. : электронный.

5. Introduction to JSON WebToken: [электронный ресурс] – URL: https://jwt.io/introduction (дата обращения: 25.05.2024). – Текст. : электронный.

6. Разделение бизнес-логики и UI во Flutter с помощью BLoC-архитектуры: [электронный ресурс] - URL: https://inostudio.com/blog/articles-develop/razdelenie-biznes-logiki-i-ui-vo-flutter-s-pomoshchyu-bloc-arkhitektury/ (дата обращения 25.05.2024). – Текст. : электронный.