Московский авиационный институт (национальный исследовательский университет)
Институт №8 «Компьютерные науки и прикладная математика»

Проект на тему:

«Web-приложение для определения и описания болезней растений по изображению»

Исполнители:
Павловский Алексей Валерьевич
Катин Иван Вячеславович
Желанов Даниил Вячеславович

Москва - 2025

Цели проекта

- **ML цель:** разработать модель классификации, которая определяет здорова ли культура или какой тип заболевания присутствует. Достичь высокой точности (более 95%) на тестовом наборе данных, обеспечить при этом быструю и качественную обработку запросов пользователей.

- **Бизнес цель:**
  - снизить расходы на диагностику заболеваний;
  - увеличить урожайность за счет своевременного выявления проблем с растениями;
  - автоматизировать процесс мониторинга состояния растений.

Описание проекта

- Описание проекта: мы разработали систему для автоматического выявления заболеваний растений по изображениям листьев, для этого мы использовали модель основанную на модифицированной архитектуре ResNet9 с остаточными блоками для повышения производительности и эффективности. Решение помогает фермерам и агрономам оперативно определять болезни и принимать меры для их устранения, благодаря полезной информации, которая предоставляется пользователям через ответ от YandexGPT.
- Исполнители:

ФИО	Группа	Обязанности
Павловский Алексей Валерьевич	М8О-410Б-21	Анализ литературы, обучение модели классификации, настройка взаимодействия сервера и модели, оформление документации, создание презентации
Катин Иван Вячеславович	М8О-410Б-21	Разработка backend части web-приложения, реализация внутренней логики сервера, настройка работы инфраструктуры cloud.ru, развертывание виртуальной машины, оформление документации
Желанов Даниил Вячеславович	М8О-410Б-21	Разработка frontend части web-приложения, проработка дизайна, развертывание виртуальной машины, реализация API, дебаггинг кода, оформление документации.

ML обучение

- Классификация - это задача машинного обучения, при которой модель предсказывает категорию (класс) для входных данных. Цель состоит в том, чтобы обучить алгоритм, который сможет разделить данные на группы, основываясь на их характеристиках. В нашем случае классификация направлена на:
  - Определение здоровья растения (здоровое/больное).
  - Распознавание конкретного заболевания на основе изображений листьев растений.
- Модель обучается на наборе данных изображений, где каждая картинка имеет метку (название заболевания или "здоровое"). После обучения модель способна анализировать новые изображения и предсказывать, к какому классу они принадлежат.
- В нашей задаче мы использовали Cross-Entropy Loss (кросс-энтропию) - это основная функция потерь, используемая для многоклассовой классификации. Она измеряет разницу между распределением вероятностей, предсказанным моделью, и истинными метками классов.
- Формула:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log(p_{y_i}), \text{где}$$

L - значение функции потерь;

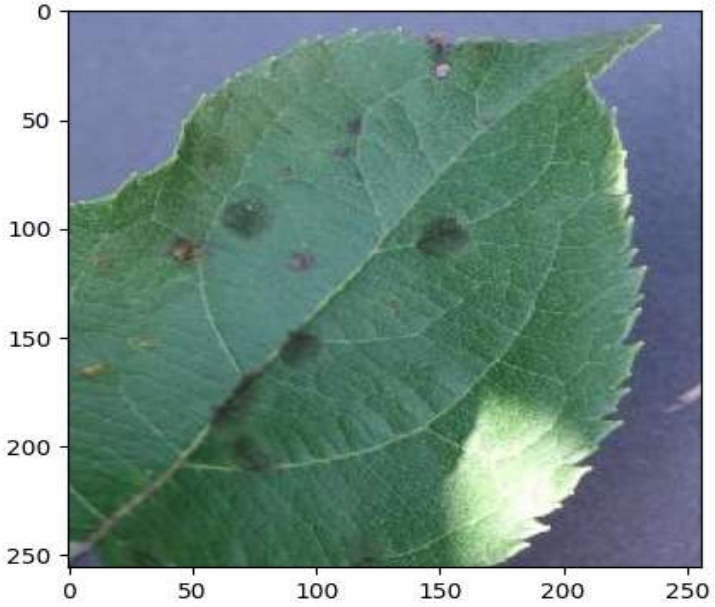
N - количество примеров в одном батче;

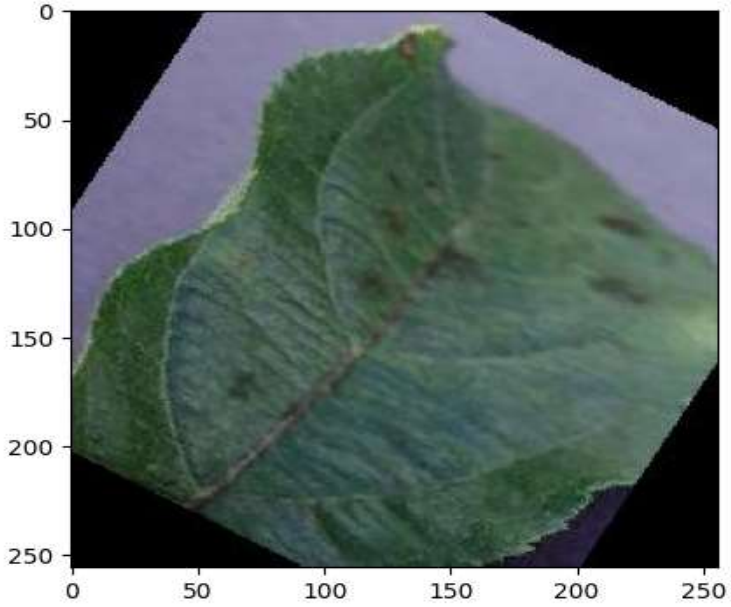
p_{y_i} - предсказанная моделью вероятность для истинного класса y_i для i-ого примера.

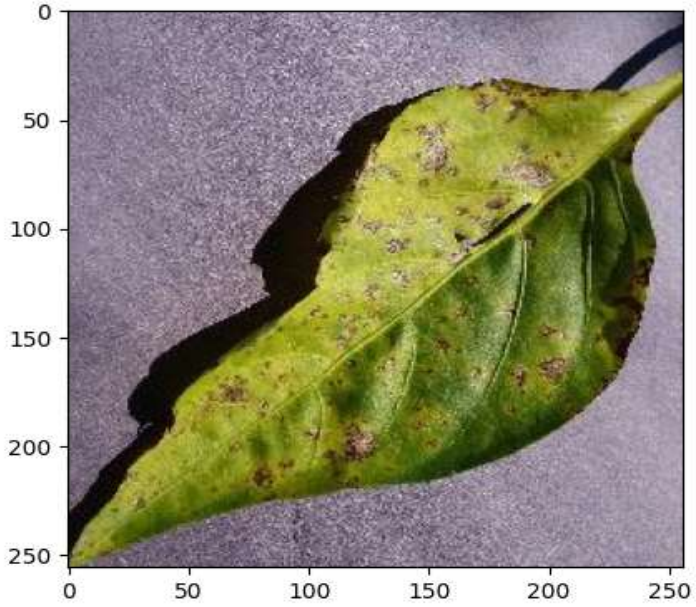
y_i - истринный класс для i-ого примера (индекс метки)

ML обучение. Dataset

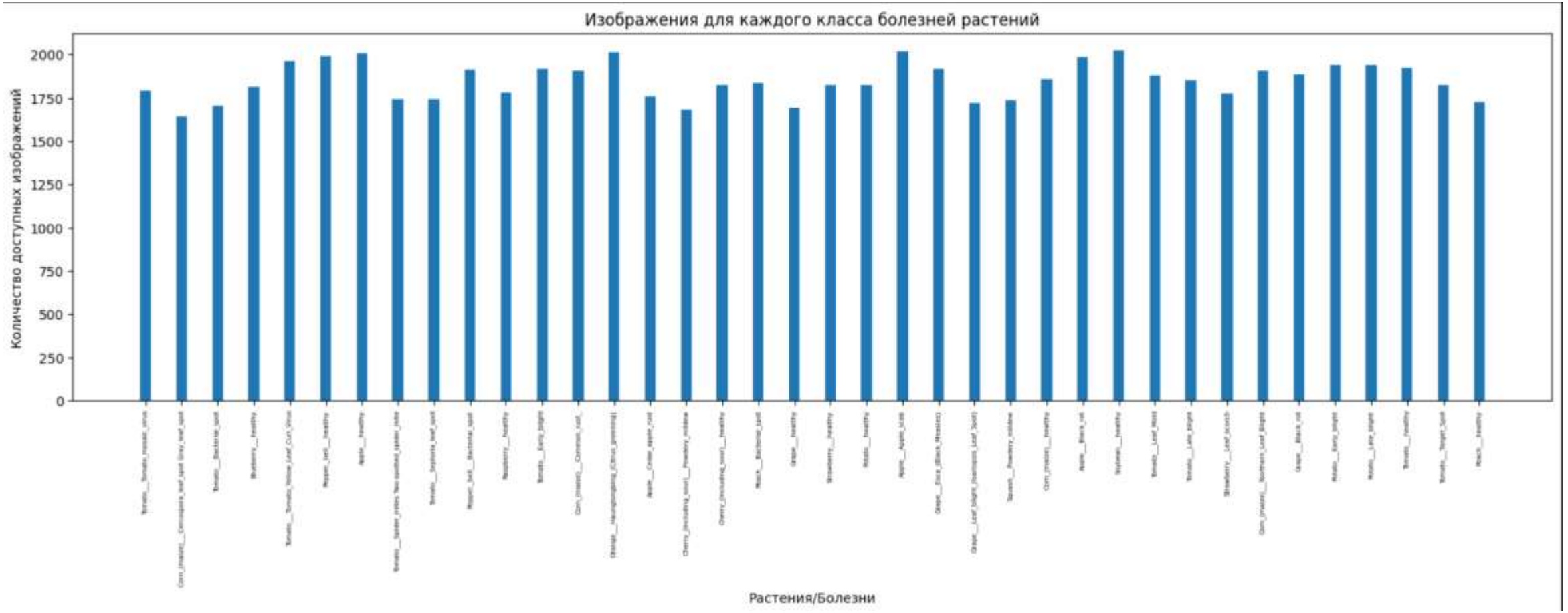
- **Dataset:**
  - мы использовали New Plant Diseases Dataset (Augmented) датасет для обучения модели. Он состоит включает в себя 38 классов, предсатвленные как комбинация названия растения и состояния (например, "Apple___Scab" — яблоко с паршой).
  - всего 14 вида растений, 26 вида болезней.
  - общее количество изображений:
    - около 87,000 изображений для обучения.
    - около 21,000 изображений для валидации.
  - количество изображений на класс:
    - В среднем: около 2300 изображений на класс.
    - Распределение почти равномерное, что делает датасет сбалансированным.
    - Ниже представлены примеры исходных данных







ML обучение. Dataset.



ML обучение. Архитектура.

- Для решения задачи классификации изображений листьев растений и определения их состояния, была выбрана архитектура ResNet9. Эта архитектура представляет собой сверточную нейронную сеть, которая благодаря остаточным связям (Residual Connections) позволяет решать проблему затухания градиентов и обеспечивает эффективное обучение даже при увеличении глубины сети. ResNet9, будучи сравнительно компактной моделью, демонстрирует высокую производительность при обработке изображений.

- Архитектура ResNet9 строится на основе свёрточных блоков, каждый из которых включает в себя слои свёртки, нормализации и активации ReLU. Остаточные блоки модели добавляют результаты предыдущих слоёв к текущим, что позволяет модели лучше сохранять важные признаки изображения на различных уровнях глубины. Эта особенность делает её особенно подходящей для анализа изображений листьев, где признаки заболевания могут быть едва заметными.

- Выбор ResNet9 обусловлен также её сравнительно небольшой сложностью и высокой эффективностью. Она легче и быстрее обучается по сравнению с более глубокими сетями, такими как ResNet50 или ResNet101, что важно при работе с большим количеством изображений (около 87,000 в тренировочном наборе). Однако архитектура ResNet9 имеет и свои ограничения: она менее точна для задач, где требуется очень высокая степень детализации, по сравнению с более глубокими сетями.

ML обучение. Архитектура.

- Для обучения модели классификации заболеваний растений из изображений используются признаки, извлекаемые из архитектуры ResNet9, которая включает несколько типов слоев, организованных в блоки для обработки и анализа изображений.

- Первые свёрточные блоки (ConvBlock1 и ConvBlock2) отвечают за извлечение низкоуровневых признаков, таких как края, текстуры и цветовые паттерны. Эти слои включают свёрточные операции, нормализацию (BatchNorm2d) и функцию активации ReLU, что позволяет выявлять основные визуальные элементы листьев растений.

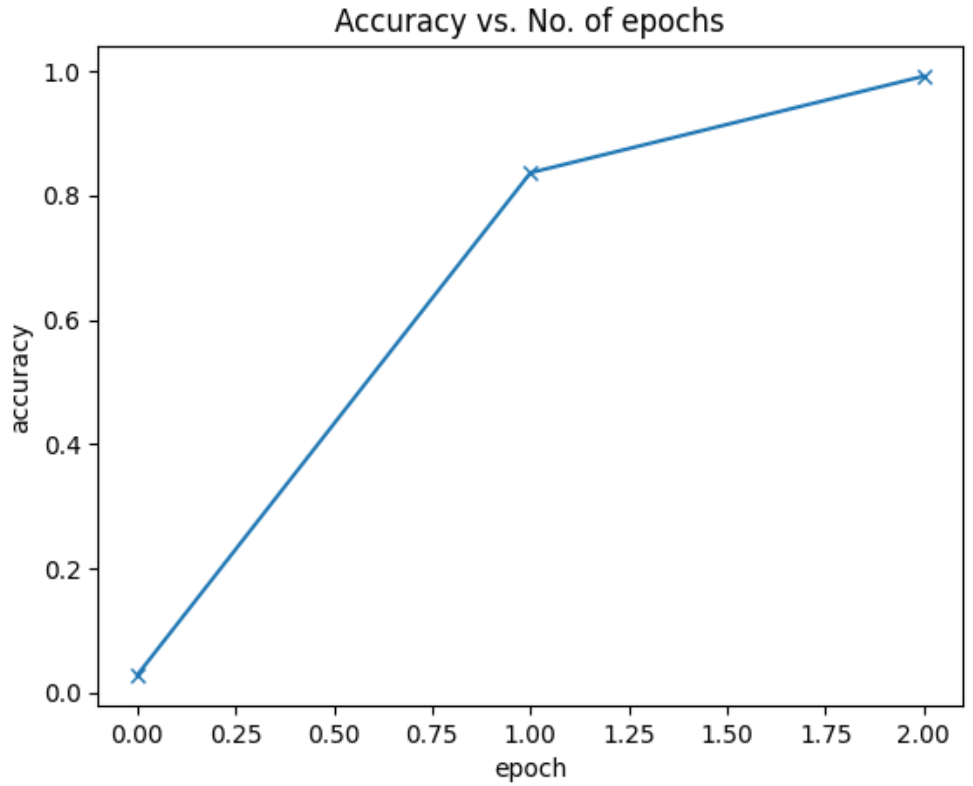
- Средние слои, такие как ConvBlock3, ConvBlock4 и первый остаточный блок (Res1), начинают кодировать более сложные текстуры и формы, характерные для конкретных заболеваний. Остаточные блоки в архитектуре позволяют сохранять и комбинировать признаки из предыдущих слоев, улучшая способность модели улавливать как локальные, так и глобальные зависимости в изображениях.

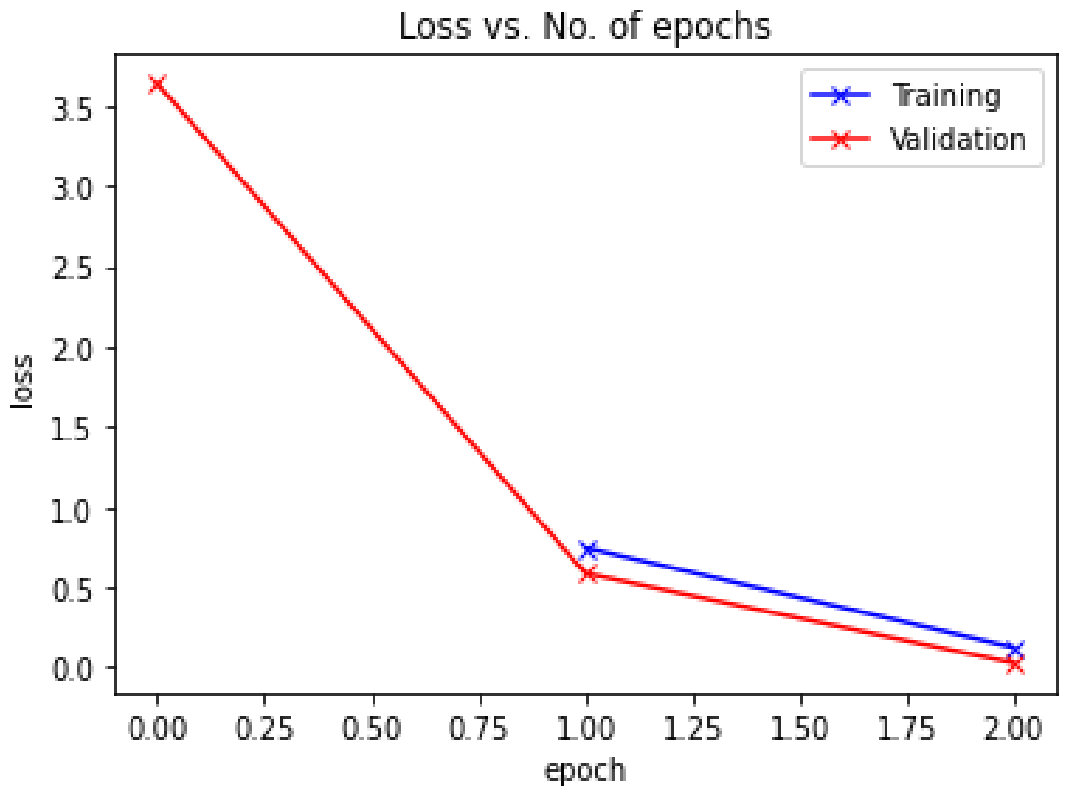
- Последние слои, включая второй остаточный блок (Res2) и классификатор, обрабатывают высокоуровневые признаки, такие как общая структура и семантический контекст изображения. Классификатор включает MaxPooling, Flatten и полносвязный слой, который преобразует выходные данные в вероятности для каждого из 38 классов заболеваний.

- Благодаря такой организации, ResNet9 эффективно выделяет признаки на разных уровнях абстракции, что позволяет модели точно распознавать состояния растений, даже если визуальные различия между классами минимальны.

ML обучение. Точность.

- Достигнутая точность 99% демонстрирует, что цель была успешно реализована. Модель эффективно решает задачу многоклассовой классификации, обеспечивая высокую точность на валидационном наборе данных. Более того, значение функции потерь (loss) также является крайне низким, что свидетельствует о том, что модель не только точно предсказывает классы, но и уверенно оценивает свои предсказания.

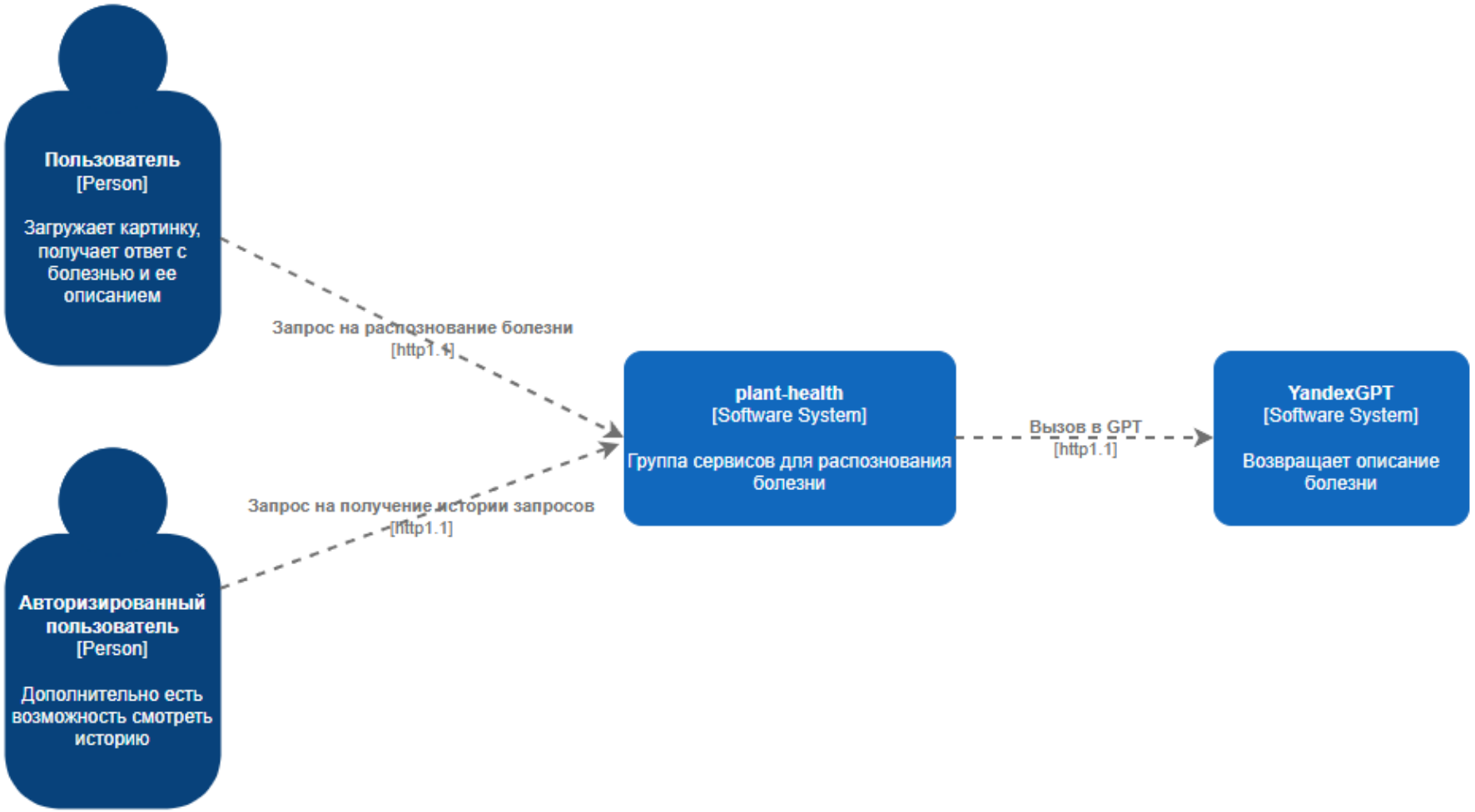




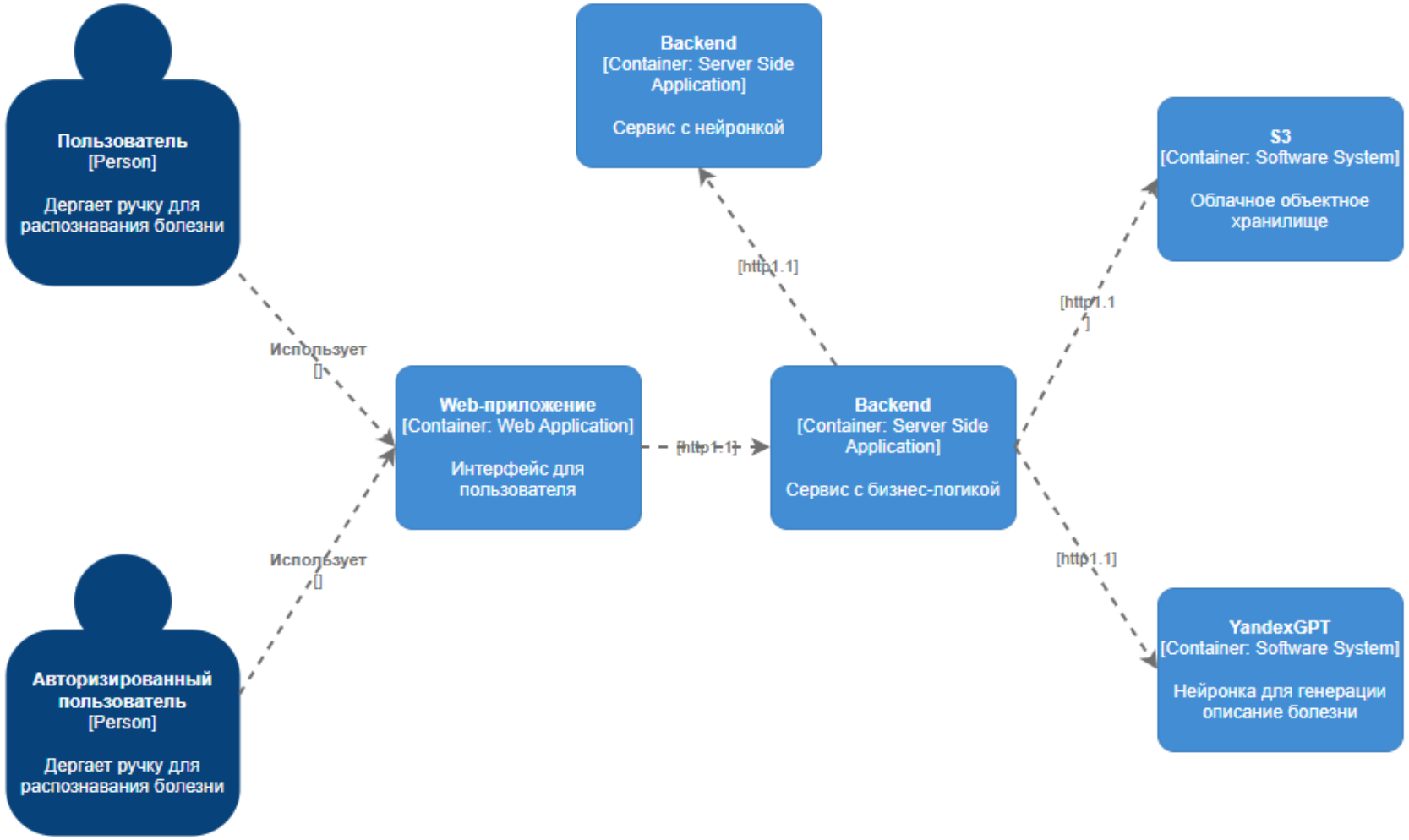
Стек технологий

Компонент	Технология	Обоснование
Язык программирования	Java	Высокая производительность, надежность, масштабируемость для серверной части.
ML-фреймворк	PyTorch	Гибкость, простота настройки и высокая производительность при работе с нейронными сетями.
Фреймворк для сервера	Spring Boot	Удобная разработка REST API, интеграция с базой данных и встроенные механизмы безопасности.
Оркестрация	Docker	Для изоляции компонентов, удобной развертки и масштабирования приложения.
База данных	PostgreSQL	Реляционная база данных с поддержкой сложных запросов и надежным хранением.
Объектное хранилище	S3 Evolution Object Storage	Надежное хранение изображений с быстрым доступом и масштабируемостью. Быстрая интеграция с Cloud.ru
Генерация текста	YandexGPT	Генерация полезных текстов на основе результатов анализа изображений.
Облачная инфраструктура	Cloud.ru	Высокая доступность и производительность для размещения серверной части.

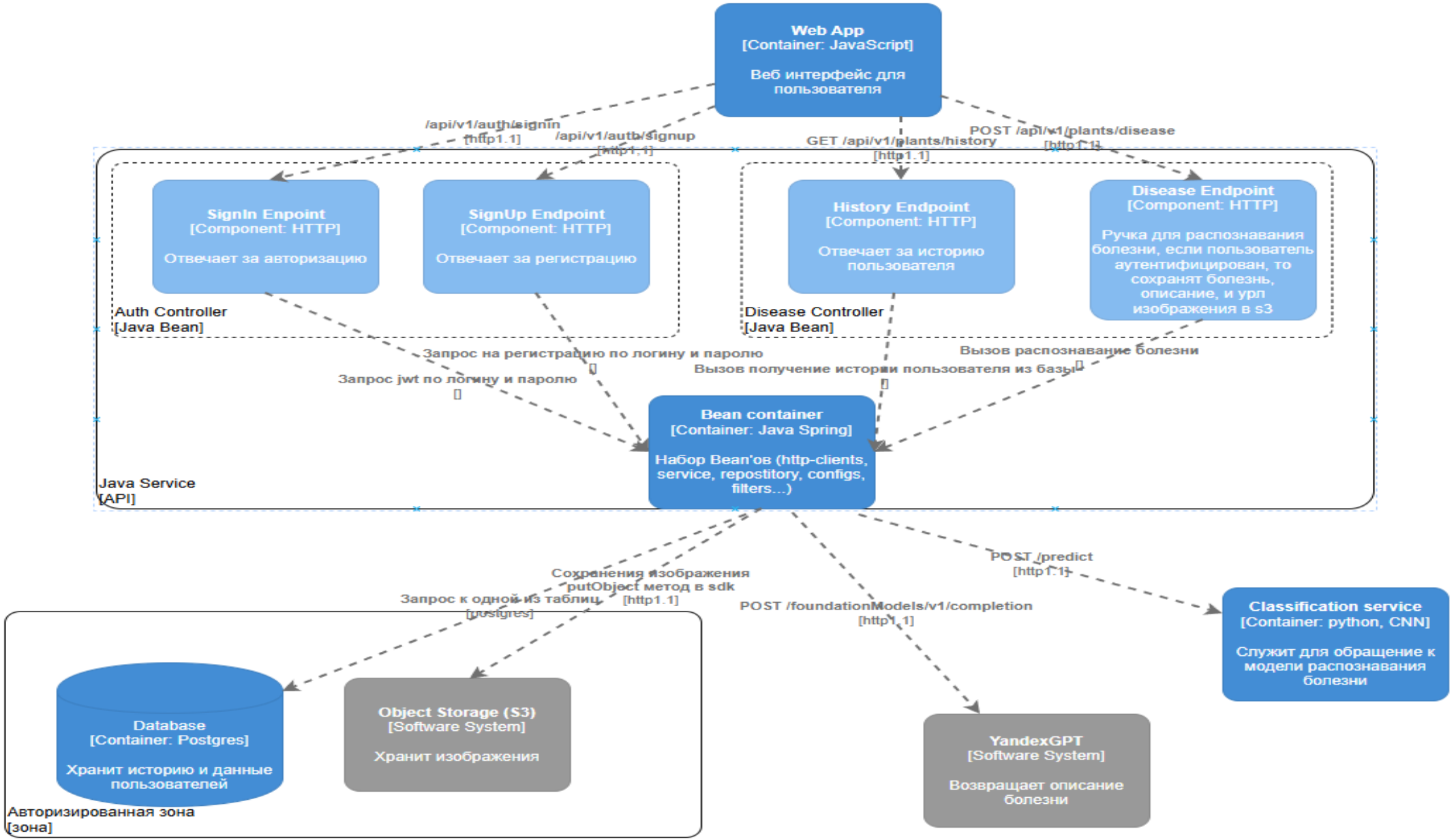
Архитектура (C4 схема). Контекст



Архитектура (C4 схема). Контейнеры



Архитектура (C4 схема). Компоненты



Архитектура (C4 схема). Код

```
/root
├── frontend/                    # Каталог с фронтендом
│   ├── css/                     # Стили для пользовательского интерфейса
│   │   └── style.css            # Основной файл стилей
│   ├── js/                      # Скрипты для взаимодействия
│   │   └── script.js            # Основной файл скрипта
│   └── index.html               # Главная HTML-страница
├── backend/                     # Каталог с серверной частью
│   ├── aop/                     # Каталог для обработки исключений
│   ├── config/                  # Конфигурации приложения
│   ├── controller/              # Контроллеры API
│   ├── dto/                     # Объекты передачи данных
│   ├── model/                   # Модели данных
│   ├── repository/              # Репозитории для работы с базой данных
│   ├── service/                 # Логика приложения
│   ├── util/                    # Вспомогательные утилиты
│   ├── BackendApplication.java  # Точка входа в приложение
│   ├── build.gradle             # Файл конфигурации Gradle
│   └── settings.gradle          # Настройки Gradle
├── ml/                          # Каталог с ML моделью
│   ├── Dockerfile               # Сборка контейнера для ML
│   ├── handler.py               # Основной файл для запуска ML модели
│   ├── plant-disease-model.pth  # Модель классификации
│   ├── plant-disease-model-complete.pth  # Полная модель
│   └── requirements.txt         # Зависимости для ML
└── docker-compose.yml           # Оркестрация контейнеров
```

Обоснование архитектуры

Архитектура проекта построена с учетом модульности, асинхронности и необходимости высокой производительности. Это обеспечивает надежность системы, легкость масштабирования и возможность интеграции новых функций. Выбор технологий основан на функциональных требованиях проекта и их преимуществ в реализации задач классификации заболеваний растений. Ниже приводится обоснование архитектурных решений и выбора технологий.

Язык программирования: Java

Java был выбран для реализации серверной части благодаря своей производительности, надежности и масштабируемости. Язык отлично подходит для создания высоконагруженных приложений.

Почему не Python? Python удобен для разработки ML-моделей, но для серверной части он менее эффективен из-за своей однопоточности и меньшей производительности в условиях высокой нагрузки.

ML-фреймворк: PyTorch

PyTorch был выбран для разработки и обучения модели классификации заболеваний растений (ResNet9) благодаря своей гибкости и производительности. Этот фреймворк предоставляет богатый инструментарий для работы с глубокими нейронными сетями.

Почему не TensorFlow? Хотя TensorFlow также является мощным инструментом, PyTorch предлагает более простой и гибкий подход к обучению моделей. Это особенно важно при необходимости быстро тестировать и модифицировать архитектуры.

Архитектура сервера: Spring Boot

Для серверной части использован Spring Boot, поскольку он предоставляет эффективные инструменты для реализации REST API, работы с базой данных и обеспечения безопасности.

Обоснование архитектуры

Обработка изображений: ResNet9

Архитектура ResNet9 выбрана для классификации изображений растений. Она эффективно обрабатывает визуальные данные благодаря остаточным связям, позволяющим сохранять важные признаки на разных уровнях абстракции.

Почему не VGG16? VGG16 имеет большую сложность и требует больше времени на обучение. ResNet9 предоставляет сопоставимые результаты с меньшими вычислительными затратами.

Облачная инфраструктура: Cloud.ru (Evolution)

Cloud.ru предоставляет удобную и производительную облачную инфраструктуру, которая позволяет масштабировать проект в зависимости от нагрузки. Высокая доступность сервиса обеспечивает бесперебойную работу бота.

Почему не Google Cloud или подобные инфраструктуры? Cloud.ru предлагает более доступные тарифы и соответствует требованиям локального рынка.

Объектное хранилище: S3 Evolution Object Storage

Объектное хранилище используется для долговременного хранения изображений и данных. S3 Evolution поддерживает управление большими объемами данных, предоставляя быстрый доступ и возможность масштабирования.

Асинхронность

Асинхронные процессы: Изображения загружаются в облачное хранилище S3 и обрабатываются параллельно с запросами к модели и YandexGPT.

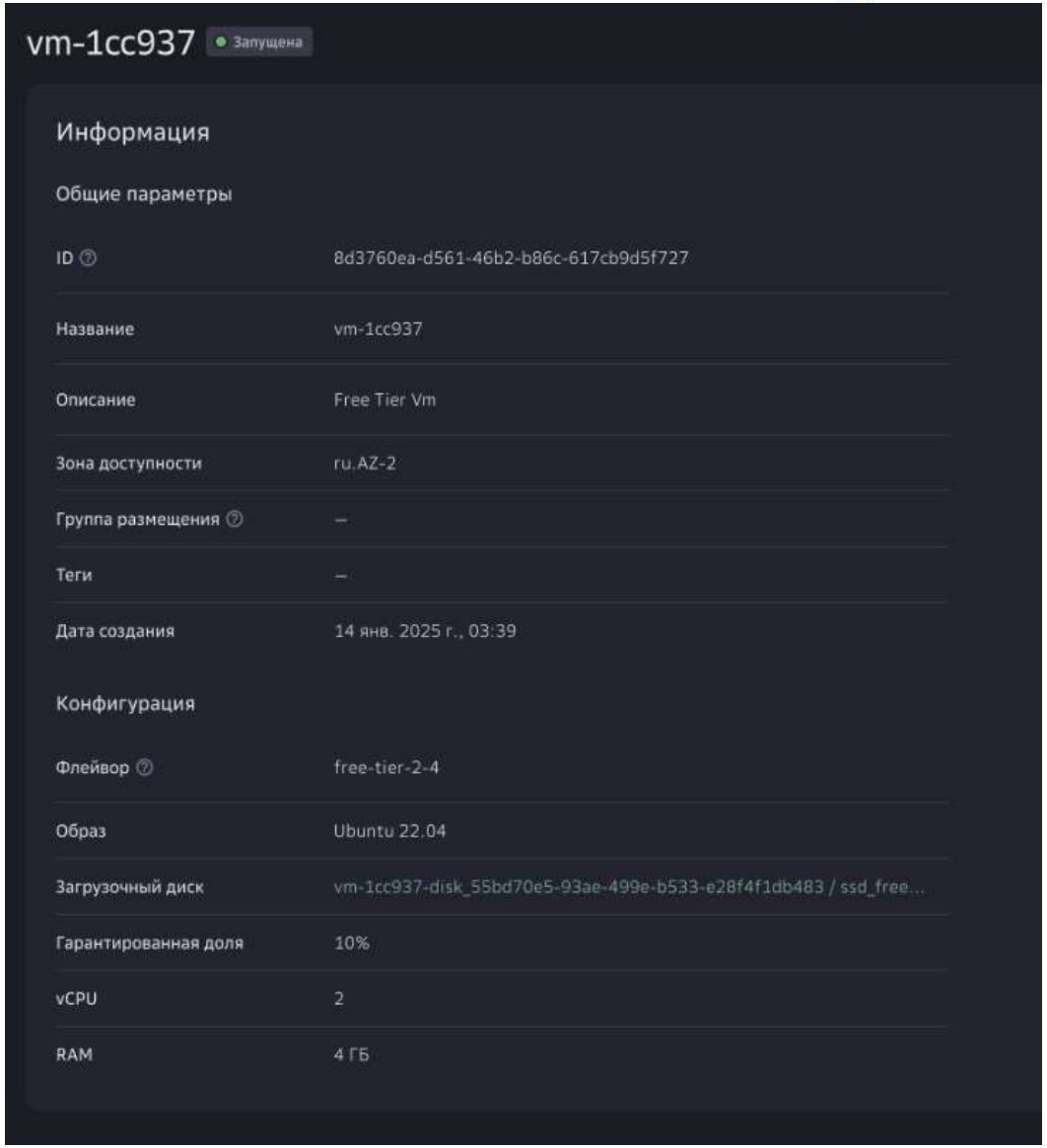
Обоснование дополнительных технологий

- YandexGPT: Внешний API для генерации текстовых описаний заболеваний растений, интеграция позволяет повысить ценность предоставляемых данных.
- PostgreSQL: Реляционная база данных для хранения пользователей, изображений и результатов обработки.

Виртуальная машина

Виртуальная машина на cloud.ru была выбрана для этого проекта, так как она:

- Предоставляет оптимальную производительность для работы моделей и сервисной части приложения.
- Обеспечивает централизованное управление и доступность.
- Позволяет эффективно использовать ресурсы и адаптироваться к растущей нагрузке.
- Обеспечивает высокий уровень безопасности, отказоустойчивости и интеграции с другими сервисами.



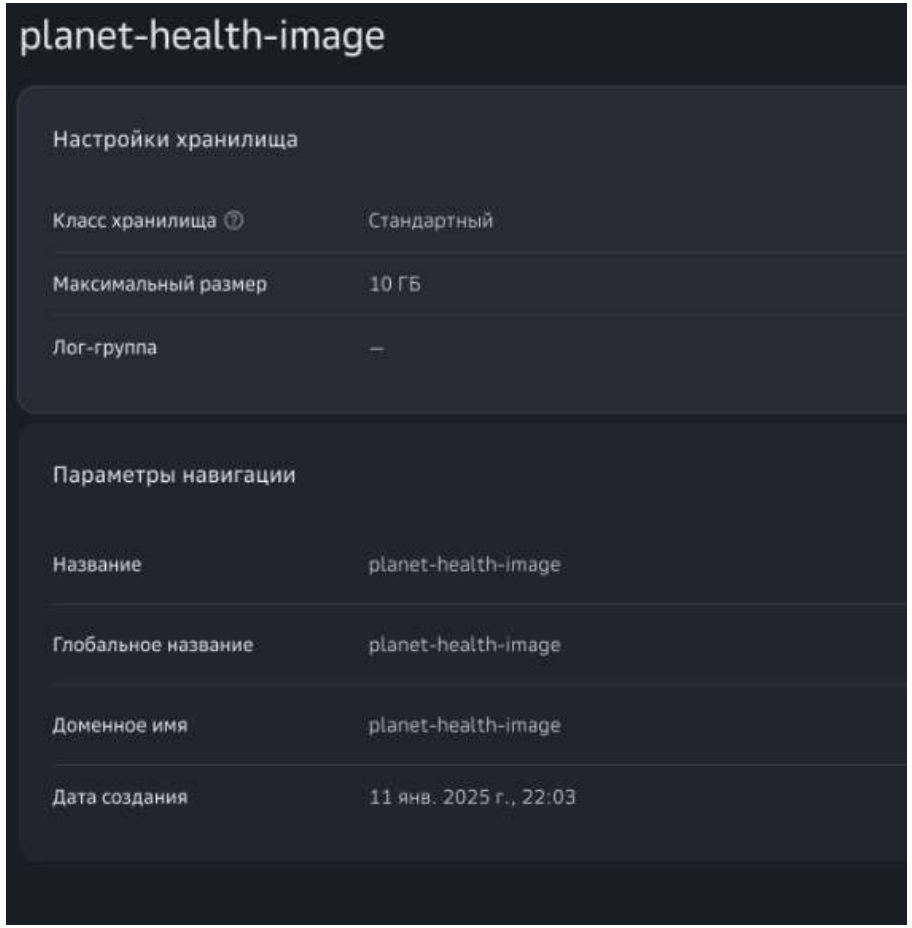
Объектное хранилище

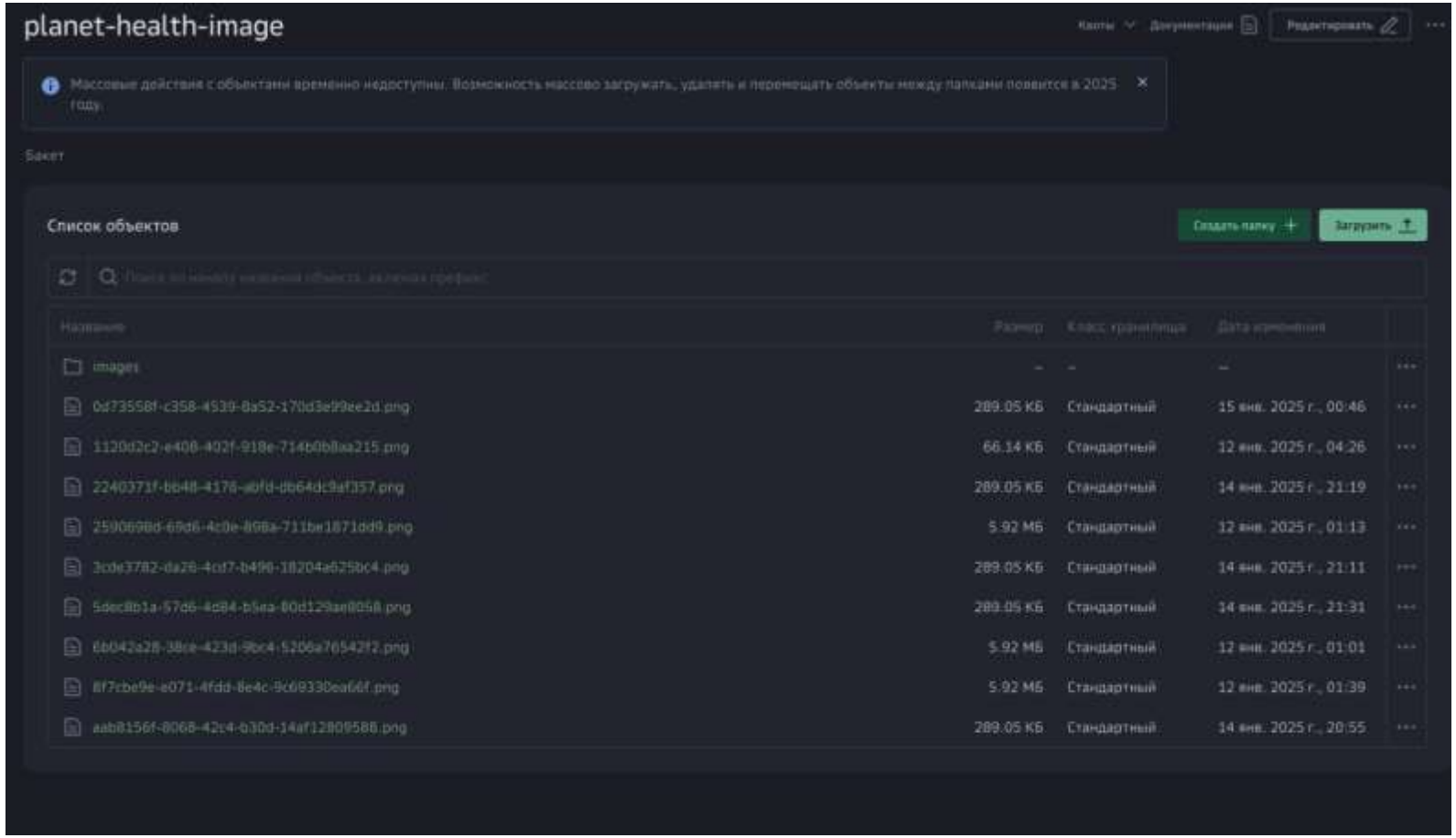
S3-хранилище на cloud.ru используется в проекте для:

- Хранения входных изображений и результатов работы модели.
- Масштабируемого и отказоустойчивого хранения данных.
- Разгрузки локальных ресурсов виртуальной машины.
- Удобной интеграции с компонентами проекта и обеспечения безопасности данных.

Это решение даёт гибкость и надёжность, необходимые для эффективной работы проекта с большим количеством пользователей

Объектное хранилище





Frontend web-приложения

Основной стек технологий:

- HTML: Разметка интерфейса страниц.
- CSS:
  - Стилизация элементов: кнопки, формы.
  - Фоновые изображения.
- JavaScript:
  - Управление DOM (отображение страниц, переключение вкладок).
  - Обработка событий (клики, ввод данных, загрузка изображений).
  - Взаимодействие с API через fetch.
- Основные компоненты:
  - Авторизация и регистрация:
  - Формы ввода для логина и пароля.
  - Управление состоянием сессии через localStorage.
- Загрузка изображения:
  - Динамическое создание <input type="file">.
- Отображение результата:
  - Интерактивное представление названия и описания болезни.
  - Использование библиотеки marked.js для форматирования текста.
- История болезней:
- Карточки с изображениями и краткой информацией.

Пример работы. Загрузка изображения

Пример работы. Ответ сервера

Potato Early blight

Причина болезни:

Картофельная гниль раннего периода (Potato Early blight) вызывается оомицетом *Alternaria solani*. Это грибковое заболевание, которое может поражать картофель и другие растения семейства паслёновых. Болезнь обычно проявляется в виде тёмных пятен на листьях, стеблях и клубнях растений.

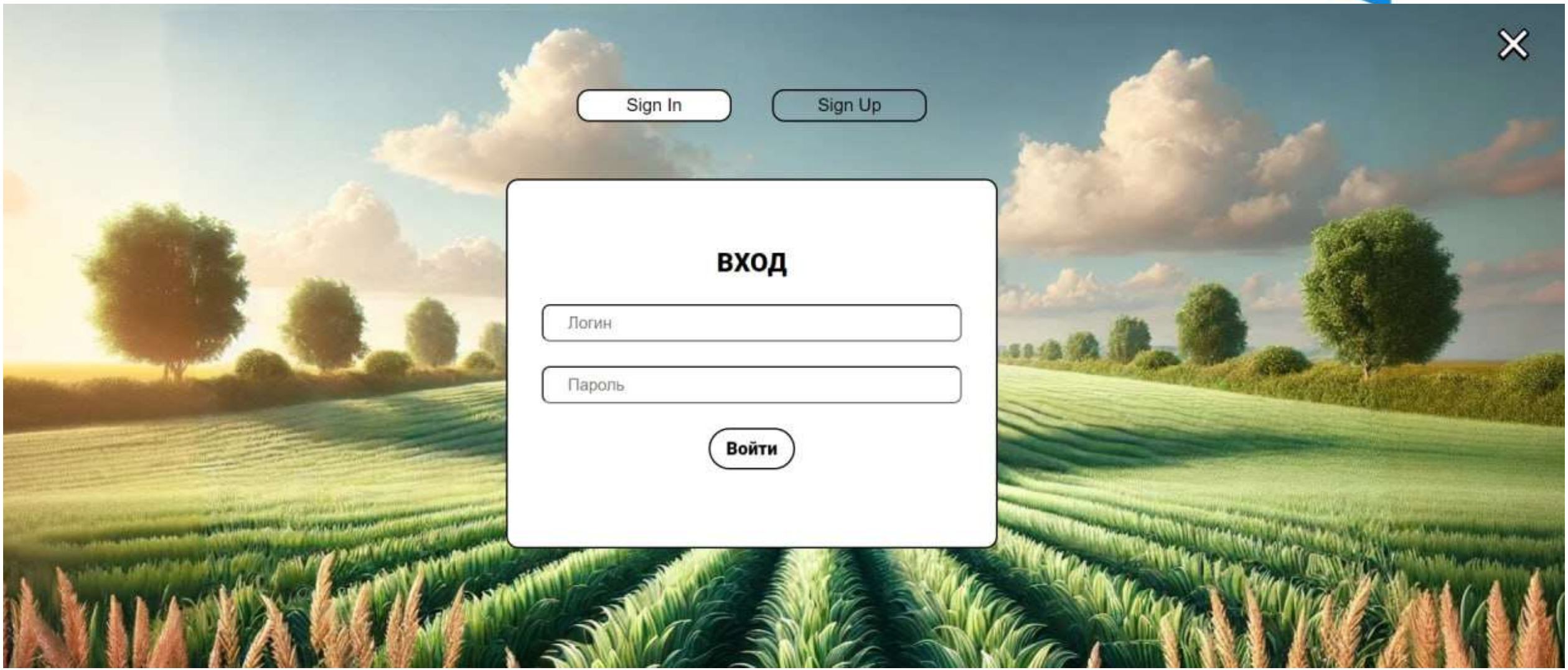
Факторы, способствующие развитию болезни:

- высокая влажность воздуха;
- температура воздуха от 15 до 22 °C;
- наличие спор гриба в почве или на растительных остатках.

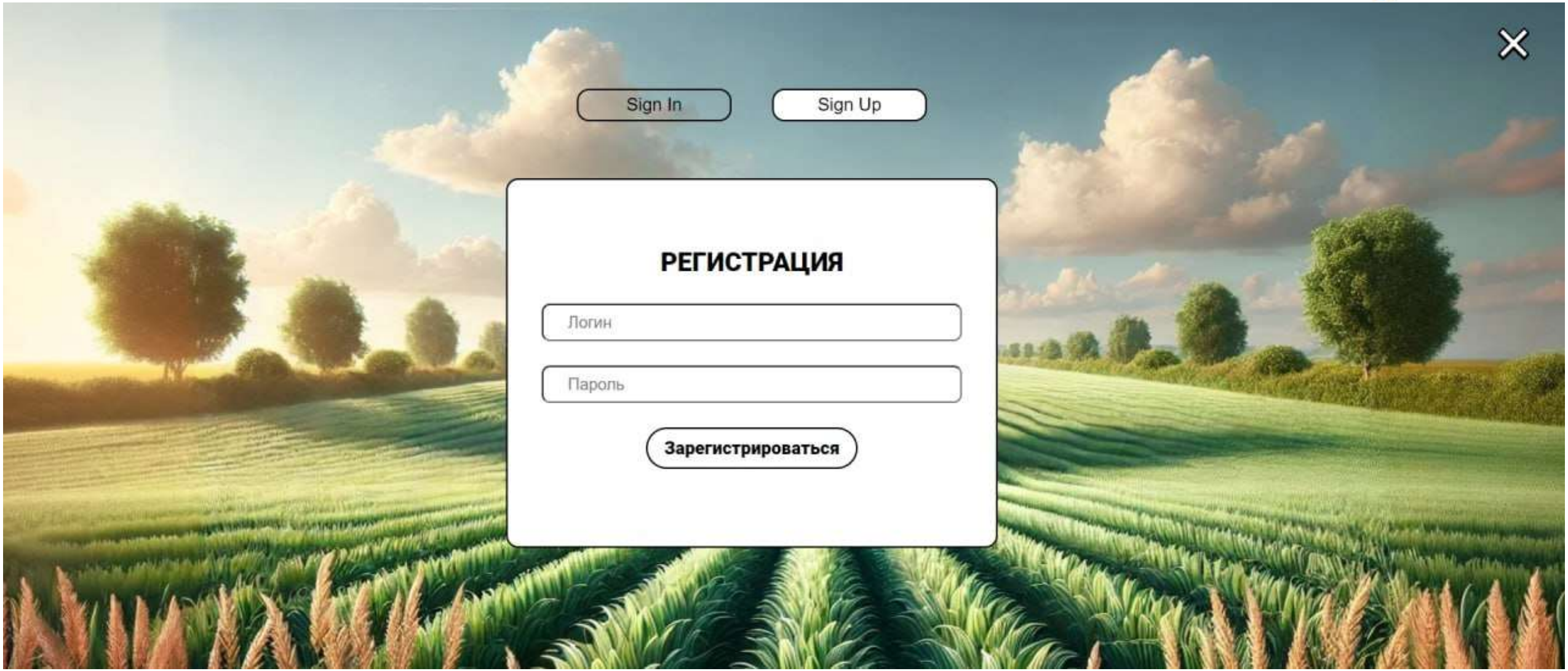
Болезнь может быстро распространяться при наличии благоприятных условий. Споры гриба легко переносятся ветром, водой и насекомыми, что способствует быстрому распространению заболевания.

Лечение и профилактика:

Пример работы. Вход



Пример работы. Регистрация



Пример работы. История запросов