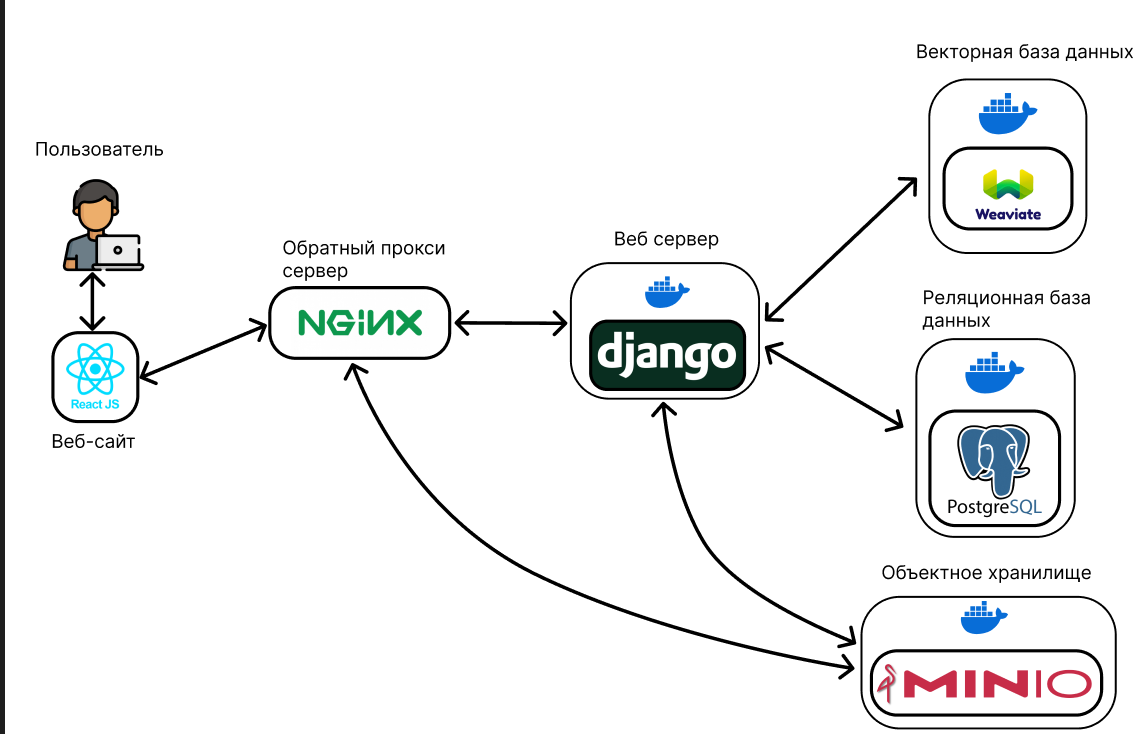
Архитектура сервиса поиска схожих по смыслу изображений



# Frontend

Клиентская часть приложения реализована в виде [веб-страницы](https://app.meiiiok.ru/).

Приложение разработано при помощи web-фреймворка React.

A screenshot of a screenshot of a zebra

Description automatically generated

## Функционал

Приложение позволяет пользователю загрузить в сервис любую фотографию, после чего отображает пользователю 20 наиболее релевантных фотографий, найденных в базе данных в порядке убывания релевантности. Также при выборе изображения у пользователя есть возможность выбрать «Режим сохранения» который помимо поиска схожих картинок также сохранит загруженное изображение в базе данных для последующего поиска.

Взаимодействие с базой данных происходит через backend сервер посредством REST API запросов.

# Backend

## Django

Серверная часть приложения, отвечающая за обеспечение корректной работы всех компонентов сервиса, разработана на языке программирования Python с использованием фреймворка Django.

Backend-сервер предоставляет одну точку доступа /api/search/ для взаимодействия с клиентской частью приложения.

Точка доступа принимает POST запрос, принимающий следующие параметры:

* image – загруженная пользователем картинка
* save – булевое значение, отвечающее за сохранение загруженного файла в базу данных

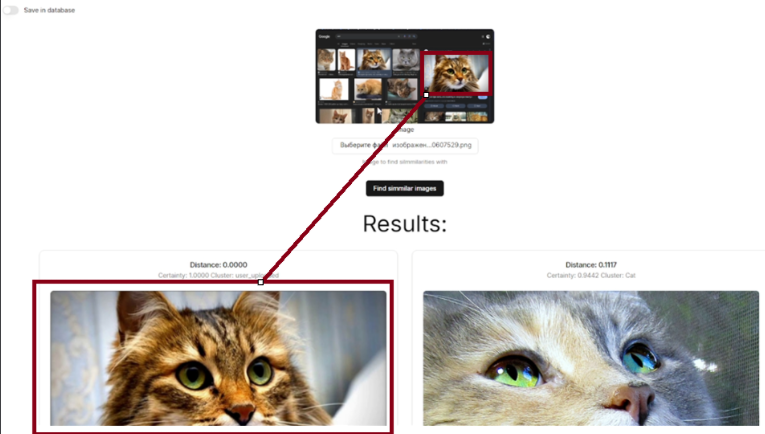
После получения запроса сервер формирует запрос в векторную базу данных weaviate, включающий в себя переданную пользователем картинку. Сервис определяет, является ли поданное на вход изображение скриншотом, и автоматически вырезает главное изображение композиции. Пользователю выводится исходное изображение, а на вход в базу данных или поиск поступает уже обрезанное изображение.

В примере ниже:

- Изображение поступило на вход (пользователю показывается исходное)

- Обрезалось

- Сохранилось в базу данных (save in database = True)



При повторном вызове этого же скриншота мы видим, что в базу добавился вырезанный котик из скриншота.

Алгоритм по вырезанию изображения основан на opencv.

После обрезки изображения сервер обрабатывает полученный ответ от векторной базы данных, состоящий из 20 наиболее близких по смыслу изображений. После получения ответа сервер запрашивает из реляционной базы данных Postgres метаданные для всех изображений:

* Название изображения
* Категория изображения
* Расположение файла в объектном хранилище minio

Далее сервер генерирует пред-подписанные ссылки (Pre-signed URL’s) на изображения в объектном хранилище.

Как итог – сервер возвращает список из 20 объектов ответ со следующей структурой:

{

“name” – название файла

“cluster” – категория к которой относится изображение

“image” – ссылка на изображение в объектном хранилище

“distance” – близость изображения к загруженному

“certainty” – метрика уверенности точности определения расстояния

A zebra grazing on grass

Description automatically generated}

В случае если пользователь выбрал режим сохранения изображения в базу – помимо формирования описанного выше ответа сервер загружает изображение в векторную базу данных weaviate, после чего получает уникальный идентификатор загруженного изображения, вносит в реляционную базу данных все присущие изображению метаданные, а также сохраняет файл в объектное хранилище minio.

## Weaviate

Weaviate - это open-source векторный поисковый движок, который предназначен для хранения, поиска и управления векторными данными. Он позволяет пользователям эффективно выполнять поиск и анализ данных, используя методы машинного обучения и векторизации.

Выбранная нами база данных разработана с учетом масштабируемости (поддержка нескольких нод, шардинг), что позволит обрабатывать большие объемы данных и поддерживать высокую производительность при выполнении запросов.

Большим плюсом weaviate является разделение базы данных, которая оптимизирована под работу на процессоре и модулей для векторизации изображений, которые оптимизированы под работу на графических процессорах.

# Minio

Minio - это высокопроизводительное объектное хранилище, которое совместимо с API Amazon S3. Оно предназначено для хранения больших объемов неструктурированных данных, таких как изображения, видео, резервные копии, журналы и другие файлы.

Мы выбрали minio в качестве объектного хранилища изображений исходя из следующих соображений:

* Open-source модель распространения позволяет развернуть объектное хранилище на своих серверах и иметь полный контроль над данными.
* Пред-подписанные ссылки (Pre-signed URL’s) позволяют генерировать ссылки на файлы с ограниченным временем действия, что предотвращает неавторизованный доступ ко всем имеющимся фавлам.
* Простота интеграции с Django

Таким образом разработанное нами решение позволяет быстро осуществлять поиск схожих по смыслу изображений, а также предоставляет возможность горизонтального масштабирования всей системы в целом благодаря контейнеризации каждого отдельного сервиса и грамотному целевому использованию современных решений.