Этапы работа над проектом

Анализ требований:
- Определение функциональных требований (обработка изображений с распознаванием лиц).
- Выбор технологии для распознавания лиц (например, OpenCV или Dlib).
- Разработка API для получения изображений от пользователей.

Разработка backend:
- Создание FastAPI сервиса для приема и обработки изображений.
- Реализация функционала для распознавания лиц с использованием библиотеки OpenCV.
- Применение паттерна Singleton для управления моделью распознавания лиц.

Работа с базой данных:
- Проектирование структуры базы данных для хранения изображений и результатов распознавания.
- Реализация CRUD операций для сохранения и извлечения данных.

Тестирование и валидация:
- Тестирование работы сервиса с различными изображениями.
- Валидация корректности данных, передаваемых через API

Документация и развертывание:
- Документирование API и сервиса.

1. Микросервис для распознавания лиц

Разработать микросервис для распознавания лиц в изображениях с использованием библиотеки OpenCV и модели машинного обучения. Микросервис будет принимать изображения через API, обрабатывать фих и возвращать координаты лиц, найденных на изображении.

Задачи:

- Создание API с использованием FastAPI.
- Разработка backend-логики для обработки изображений с использованием OpenCV.
- Хранение изображений и результатов в базе данных (например, PostgreSQL).
- Применение паттерна Singleton для управления моделью распознавания лиц.

2. Веб-приложение для классификации изображений

Создать веб-приложение, которое использует предобученную модель для классификации изображений (например, CIFAR-10). Пользователь загружает изображение, и приложение возвращает прогноз о классе изображения.

Задачи:

- Разработка frontend с использованием HTML/CSS/JS для загрузки изображений.
- Разработка backend для обработки изображений с использованием Flask и TensorFlow.
- Взаимодействие с базой данных (например, SQLite) для хранения истории запросов и прогнозов.
- Применение паттерна MVC для разделения логики обработки запросов и отображения данных.

3. Микросервис для анализа текста на изображениях (OCR)

Разработать микросервис, который использует технологию распознавания текста (OCR) для извлечения текста из изображений. Микросервис должен принимать изображение через API и возвращать распознанный текст.

Задачи:

- Использование библиотеки Tesseract для OCR.
- Создание API с использованием FastAPI.
- Валидация данных (например, проверка, что изображение содержит текст).
- Хранение результатов в базе данных и отображение их через frontend.

4. Веб-приложение для фильтрации и обработки изображений

Создать веб-приложение, которое позволяет пользователю загружать изображения, применять различные фильтры (например, изменение яркости, контраста), и сохранять результат.

Задачи:

- Разработка frontend с использованием JavaScript и React.
- Создание backend с Flask для обработки изображений с использованием Pillow.
- Применение паттерна Factory для динамического создания фильтров.
- Хранение обработанных изображений в базе данных (например, MongoDB).

5. Микросервис для обнаружения объектов на видео

Разработать микросервис, который будет анализировать видеопотоки и обнаруживать объекты в реальном времени с использованием моделей машинного обучения (например, YOLO).

Задачи:

- Реализация сервиса для анализа видеопотока с использованием OpenCV и модели YOLO.
- Разработка backend-логики для получения и обработки видеофайлов.
- Применение паттерна Strategy для выбора разных алгоритмов детекции объектов в зависимости от типа задачи.
- Хранение метаданных о видео и обнаруженных объектах в базе данных.

6. Веб-приложение для мониторинга движения объектов в реальном времени

Разработать веб-приложение для отслеживания движения объектов на видеопотоке (например, для системы безопасности).

Задачи:

- Разработка frontend на React для отображения видеопотока и отметки движущихся объектов.
- Создание backend с использованием Flask и OpenCV для анализа видеопотока.
- Валидация данных (например, проверка на наличие объектов в кадре).
- Применение паттерна Observer для уведомлений пользователей о движении объектов.

7. Микросервис для сравнения изображений (Image Similarity)

Создать микросервис, который будет сравнивать два изображения и определять их схожесть, используя алгоритмы сравнения изображений (например, структурное сравнение).

Задачи:

- Использование алгоритмов сравнения изображений (например, SSIM или Histogram Comparison).
- Разработка API с использованием FastAPI для приема и обработки изображений.
- Хранение результатов в базе данных (например, Redis) для быстрого доступа.
- Применение паттерна Adapter для интеграции с другими сервисами для получения изображений.