Документация WebPage Analyzer API

Общее описание

WebPage Analyzer - это RESTful API для анализа и сравнения веб-страниц на основе их скриншотов. Приложение использует компьютерное зрение и OCR (оптическое распознавание символов) для извлечения структуры веб-страниц и выявления различий между ними.

Технологический стек

- **Backend**: Python, Flask, Flask-RESTX
- **Обработка изображений**: OpenCV, Tesseract OCR
- **Хранение данных**: Redis (для кэширования)
- **Контейнеризация**: Docker, Docker Compose

Установка и запуск

Требования

- Docker и Docker Compose
- Python 3.9+

Запуск через Docker

```
docker-compose up -d --build
```

Приложение будет доступно по адресу: `http://localhost:5000`

Документация API

Swagger UI доступен по адресу: `http://localhost:5000/api/v1`

API Endpoints

1. Анализ изображения веб-страницы

Endpoint: `POST /analyzer/analyze`

Параметры:

- `image` : Файл изображения (PNG/JPG)

Пример запроса:

```
curl -X POST -F "image=@screenshot.png" http://localhost:5000/analyzer/analyze
```

Пример ответа:

```
{
  "elements": [
    {
      "type": "text",
      "text": "Пример текста",
      "position": {
        "x": 100,
        "y": 200,
        "width": 300,
        "height": 50
      }
    },
    {
      "type": "button",
      "text": "Кнопка",
      "position": {
        "x": 400,
        "y": 300,
        "width": 100,
        "height": 40
      }
    }
  ],
  "width": 1920,
  "height": 1080
}
```

2. Сравнение двух версий веб-страницы

Endpoint: `POST /analyzer/compare`

Параметры:

- `image1` : Первая версия страницы (PNG/JPG)
- `image2` : Вторая версия страницы (PNG/JPG)

Пример запроса:

```
curl -X POST -F "image1=@old.png" -F "image2=@new.png" http://localhost:5000/analyzer/compare
```

Пример ответа:

```
{
  "differences": [
    {
      "change_type": "added",
      "new_position": {
        "x": 500,
        "y": 200,
        "width": 100,
        "height": 30
      },
      "new_text": "Новый элемент",
      "similarity_score": 0.0
    }
  ],
  "added_count": 1,
  "removed_count": 0,
  "modified_count": 0,
  "moved_count": 0
}
```

Модели данных

WebPageStructure

```
{
  "elements": [
    {
      "type": "string (text|button|image|input)",
      "position": {
        "x": "number",
        "y": "number",
        "width": "number",
        "height": "number"
      },
      "text": "string (optional)",
      "image_embedding": "string (base64, optional)"
    }
  ],
  "width": "number",
  "height": "number"
}
```

PageComparison

```
{
  "differences": [
    {
      "change_type": "string (added|removed|modified|moved)",
      "old_position": "object (optional)",
      "new_position": "object (optional)",
      "old_text": "string (optional)",
      "new_text": "string (optional)",
      "similarity_score": "number (0-1)"
    }
  ],
  "added_count": "number",
  "removed_count": "number",
  "modified_count": "number",
  "moved_count": "number"
}
```

Настройки окружения

Переменная	Значение по умолчанию	Описание
FLASK_APP	app/**init**.py	Точка входа Flask
FLASK_ENV	development	Режим работы (development/production)
TESSDATA_PREFIX	/usr/share/tesseract-ocr/4.00/tessdata	Путь к данным Tesseract OCR

Локализация

API поддерживает распознавание текста на русском и английском языках. Для других языков необходимо установить соответствующие языковые пакеты Tesseract.

Обработка ошибок

Приложение возвращает следующие HTTP-коды ошибок:

- 400: Некорректный формат изображения
- 422: Ошибка обработки изображения
- 500: Внутренняя ошибка сервера

Разработка

Для локальной разработки:

1. Клонируйте репозиторий
2. Создайте виртуальное окружение:

```
python -m venv venv
source venv/bin/activate  # Linux/Mac
venv\Scripts\activate  # Windows
```

3. Установите зависимости:

```
pip install -r requirements.txt
```

4. Запустите Redis:

```
docker-compose up redis -d
```

5. Запустите приложение:

```
flask run
```