OCR ve Metin Analizi Projesi Raporu

Proje Tanımı: Bu rapor, Optical Character Recognition (OCR) ve metin analizi üzerine odaklanan bir projeyi detaylandırmaktadır. Proje, görsel verileri işleyerek içerisinde bulunan hassas verileri çıkarmayı amaçlamaktadır. Görsel veriler, API aracılığıyla alınmakta, metin içeriği çıkarılmakta ve bu metin içeriği üzerinde belirli hassas veri tipleri tespit edilmektedir. Proje, aynı zamanda daha önce işlenmiş görsellerin sonuçlarını önbellekte saklayarak hızlı yanıtlar sağlamaktadır.

Kullanılan Teknolojiler: Proje geliştirilirken aşağıdaki teknolojiler kullanılmıştır:

- Python: 3.11.4 (https://www.python.org/)
- Dynaconf: 3.2.1 (https://github.com/rochacbruno/dynaconf)
- FastAPI: 0.100.1 (https://github.com/tiangolo/fastapi)
- Uvicorn: 0.23.2 (https://github.com/encode/uvicorn)
- Jinja2: 3.1.2 (https://github.com/pallets/jinja)
- Httpx: 0.24.1 (https://github.com/encode/httpx)
- PyTesseract: 0.3.10 (https://github.com/madmaze/pytesseract)
- Python Multipart: 0.0.6 (https://github.com/andrew-d/python-multipart)
- Pillow: 10.0.0 (https://github.com/python-pillow/Pillow)
- Redis: 5.0.0 (https://github.com/redis/redis-py)
- Async Timeout: 4.0.3 (https://github.com/aio-libs/async-timeout)
- BeautifulSoup4: 4.12.2 (https://pypi.python.org/pypi/beautifulsoup4)
- Dateparser: 1.1.8 (https://github.com/scrapinghub/dateparser)
- Validators: 0.21.2 (https://validators.readthedocs.io/en/latest/)

OCR ve Metin Analizi Proje Hazırlık Süreci

Bu bölümde, projenin geliştirilme aşamasındaki hazırlık süreci ayrıntılı bir şekilde açıklanmıştır.

1. **Araştırma ve Teknoloji Seçimi:** Projede kullanılacak olan Optical Character Recognition (OCR) ve metin analizi teknikleri için başlangıçta geniş bir araştırma yapıldı. Bu araştırmada, PyTesseract ve Pillow gibi kütüphaneler projenin amacına en uygun çözüm olarak belirlendi.
2. **PyTesseract ve Pillow Kütüphanelerinin Seçimi ve Araştırması:** OCR ve metin analizi için PyTesseract ve Pillow kütüphanelerinin projeye entegre edilmesi kararı alındı. PyTesseract, görüntülerdeki metni algılama ve çıkarma yeteneği sunarken, Pillow görüntü işleme işlevselliği sağlamaktadır. Her iki kütüphane de projenin gereksinimlerini karşılayacak şekilde araştırıldı.
3. **Poetry ve Dynaconf Entegrasyonu:** Projenin geliştirilmesi ve yönetimi için bağımlılıkların yönetimi için Poetry ve konfigürasyon ayarlarının düzenlenmesi için Dynaconf entegre edildi. Poetry, proje bağımlılıklarını yönetmek için kullanılırken Dynaconf, proje ayarlarının dinamik olarak yönetilmesine yardımcı oldu.

4. **FastAPI ile Endpoint Oluşturma:** Proje, kullanıcıların görsel dosyaları yükleyebileceği bir API endpointi oluşturmak amacıyla FastAPI kullanıldı. FastAPI, hızlı ve etkili bir web API çözümü sağlamaktadır.
5. **OCR Analizi Kodunun Yazılması:** PyTesseract kütüphanesi kullanılarak görsel verilerin metin içeriğine dönüştürülmesi işlemi için gerekli olan kod yazıldı. Bu adım, projenin ana fonksiyonelliğini temsil eder.Bu kod verdiğimiz resimdeki yazıları alır ve string olarak dönderir.
6. **Metin İçeriğinin MD5 Hash Kontrolü ve Redis Cacheleme:** OCR analizi sonucu dönen string değeri md5 ile hashledim.Eğer bu değerde rediste cache te değer yoksa gerekli analizlerden sonra redise kaydetim. Eğer rediste bu cachete bir değer varsa önceden analiz edilmiş demektir. Cacheten diek analiz sonucunu aldım.
7. **Hassas Veri Türlerinin Kontrolü:** Belirlenen hassas veri tipleri (PHONE_NUMBER, ID_NUMBER, vb.) için ayrı ayrı kontrol işlemleri gerçekleştirildi. Doğrulama işlemleri, verinin geçerli ve güvenilir olduğundan emin olmak amacıyla yapıldı.
8. **Ek Doğrulama Mekanizmaları:** Hassas veri türlerinin doğrulanması için özgün mekanizmalar geliştirildi. Örneğin, kredi kartı numaralarının Luhn algoritması kullanılarak doğrulanması ve DNS lookup işlemi ile alan adlarının tespiti gibi aşamalar gerçekleştirildi.
9. **Gizli Bilgilerin Korunması:** Projenin gizli bilgileri (örneğin API anahtarları) git-secret kullanılarak korundu. Bu sayede hassas verilerin sızdırılması riski minimize edildi.
10. **Docker ve Docker Compose:** Proje, Dockerfile ve docker-compose.yml dosyaları yardımıyla konteynerlara taşındı. Bu sayede projenin taşınabilirliği ve dağıtımı kolaylaştırıldı.
11. **Konfigürasyon Dosyası:** Proje ayarlarını yönetmek için .pre-config.yml dosyası oluşturuldu. Bu dosya projenin farklı ortamlarda (örneğin geliştirme ve üretim) nasıl davranması gerektiğini belirledi.
12. **Gitignore Dosyası:**Geliştirme sürecinde kullanılan geçici dosyaların ve gereksiz verilerin sürüme eklenmesini engellemek için .gitignore dosyası oluşturuldu.

Bu süreçte belirtilen adımlar, projenin temelini oluşturdu ve projenin ilerleyen aşamalarına geçiş için güçlü bir temel sağladı. Teknoloji seçimi, entegrasyon süreci ve farklı aşamaların oluşturulması sayesinde projenin amacına ulaşması sağlandı.

Beklenen HTTP Durum Kodları:

- HTTP 200 - Başarılı: İşlem başarıyla tamamlandı.
- HTTP 204 - İçerik Yok: Görsel okundu, ancak içerik bulunamadı.
- HTTP 400 - Hatalı İstek: Görsel okunamadı veya hatalı formatta.

Hassas Veri Tipleri:

- PHONE_NUMBER
- ID_NUMBER
- CREDIT_CARD_NUMBER
- PLATE
- DATE
- EMAIL
- DOMAIN

- URL
- HASH
- COMBOLIST

Proje Sonucu: Proje, belirtilen hassas veri tiplerini tespit etme yeteneğine sahip bir OCR ve metin analizi çözümü sunmaktadır. Projenin sağladığı önbellekleme mekanizması, daha hızlı ve verimli sonuçlar elde etmeyi mümkün kılmaktadır.

Proje Dağıtımı: Proje, Docker Compose kullanılarak uygulama ve Redis önbelleği dahil bir bütün olarak dağıtılmaktadır. Docker Compose, projenin farklı bileşenlerini tek bir yapıda çalıştırmamızı ve konfigürasyonunu yönetmemizi sağlar. Bu yaklaşım, projenin kolayca bir sunucuda veya tunnelling mekanizması ile internete açılmasını sağlar. Docker konteynerleri, projenin herhangi bir ortamda çalışmasını ve dağıtılmasını kolaylaştırır. Redis önbelleği sayesinde daha önce işlenmiş veriler hızlıca alınabilir ve gereksiz işlem maliyeti azaltılır. Bu dağıtım yöntemi, projenin etkili bir şekilde yönetilmesini ve hızlı bir dağıtım sürecini destekler.