22.10.2025

Case Study: Belge ve Kimlik Doğrulama Sistemi Tasarımı ve Geliştirilmesi

Emir Yorgun

Amaç

Kullanıcılar tarafından yüklenen iki farklı belgenin, kimlik ve imzalı form, içindeki temel bilgileri ve imzaları karşılaştırarak tutarlılığını doğrulayan bir sitemin tasarlanması ve prototipinin geliştirilmesi.

1. Teknik Tasarım ve Mimari

Bu bölümde temel teknik mimarisi, sistem bileşenleri ve bu bileşenlerin birlikte çalışma prensipleri ele alınmıştır.

1.1. Mimari Seçim

Sistem, temelde birbirinden farklı sorumluluklara ve teknik gereksinimlere sahip servisleri (kullanıcıdan belge alımı, optik karakter tanıma (OCR) ile metin okuma, imza karşılaştırma vb.) barındırmaktadır. Sistemin bu çok katmanlı yapısı, mimari seçimi üzerinde belirleyici bir rol oynamaktadır. Özellikle "Metin Tanıma" ve "İmza Karşılaştırma" gibi görüntü işleme adımları yoğun hesaplama gücü (CPU/GPU) gerektirirken, kullanıcı isteklerini karşılayan "API Servisi" daha çok ağ trafiğini yönetmeye odaklıdır.

Bu kaynak ihtiyacı farklılığı, sistemin monolith bir yapıda olması durumunda, yalnızca tek bir bölümdeki yoğunluk nedeniyle tüm uygulamanın ölçeklendirilmesini gerektirerek verimsizliğe yol açacaktır. Mikroservis yaklaşımı ise, her işlevin kendi kaynak ihtiyacına göre bağımsız olarak ölçeklenmesini sağlar. Böylece yalnızca yoğunluk yaşanan servis güçlendirilir, sistemin geneli etkilenmeden kaynaklar verimli kullanılır. Örneğin, sadece OCR servisi yoğun yük altındayken, diğer servisleri etkilemeden yalnızca bu servisin işlem kapasitesi artırılabilir. Ayrıca, bu servislerin birbirinden izole edilmesi, sistemin hatalara karşı dayanıklılığını artırır. Örneğin, "İmza Karşılaştırma Servisi"nde yaşanacak yoğunluğa bağlı bir yavaşlama veya hata, "API Servisi" üzerinden yeni kullanıcıların belge yükleme işlemini kesintiye uğratmaz.

1.2. Sistemin Ana Bileşenleri ve Haberleşme Yapısı

Sistem mimarisi, belirli sorumluluklara sahip mantıksal bileşenlerin tanımlanması ve bu bileşenler arasındaki etkileşim modelinin oluşturulması esasına dayanır. Sistemin bütününde, işlem bütünlüğünü ve esnekliği sağlamak amacıyla servisler arası iletişim için asenkron bir haberleşme modeli benimsenmiştir. Bu model, zaman alıcı operasyonların sistemin genel performansını etkilemesini önler.

API Gateway: Kullanıcıdan gelen belge yükleme taleplerini karşılayan ilk bileşendir. Bu talepleri alır ve sistemin iç servislerine ileterek doğrulama sürecini başlatır.

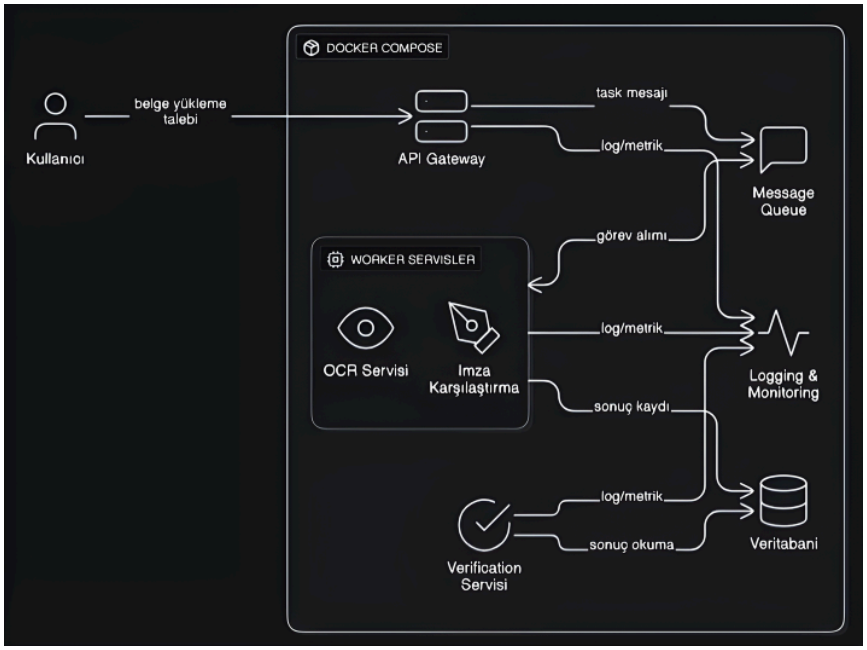
Message Queue: Servisler arası iletişimi asenkron şekilde yöneten yapıdır. API Gateway tarafından tetiklenen işlemleri görev olarak tanımlar ve ilgili servislerin erişebileceği şekilde sıraya alır. Bu sayede servisler birbirinden bağımsız çalışabilir.

Worker Servisler: Belge üzerindeki teknik işlemleri gerçekleştiren servislerdir. Örneğin OCR servisi metin tanıma işlemini yürütürken, imza karşılaştırma servisi görsel benzerlik analizini yapar. Her servis, Message Queue üzerinden aldığı görevleri işler ve sonuçları veri tabanına kaydeder.

Verification Servisi: Worker servislerden gelen çıktıları değerlendirerek doğrulama kararını verir. Veritabanındaki işlenmiş verileri analiz eder ve sistemin nihai çıktısını üretir.

Veritabanı: Sistemin tüm işlem geçmişini, belge içeriklerini ve doğrulama sonuçlarını kalıcı olarak saklayan bileşendir.

Bu bileşenler arasındaki haberleşme, Message Queue üzerinden yönetilen event-driven bir model ile sağlanır. Süreç, API Gateway'in bir talep almasıyla başlar; bu talep, Message Queue'ya bir task mesajı olarak gönderilir. Worker servisler, bu kuyruktan görevleri alır, bağımsız olarak işler ve sonuçlarını merkezi Veritabanı'na yazar. Son olarak Verification Servisi, sistemin son kararını verir.



Figür 1 - Sistemin Akış Diyagramı

1.3. Docker

Her bir mikroservis (API Gateway, Worker Servisler vb.), kendi bağımlılıkları ile birlikte bir Dockerfile kullanılarak container haline getirilir. Bu yaklaşım, servisleri birbirinden ayrıştırarak her birinin farklı ortamlarda tutarlı bir şekilde çalışmasını

sağlar. Ayrıca her servis kendi imajı içerisinde çalıştığı için dependency conflict durumu engellenir.

Tüm bu container'ların ve sistemin ihtiyaç duyduğu diğer bileşenlerin (Veritabanı, Message Queue) birlikte çalışması için docker-compose kullanılır. Proje ana dizininde yer alan docker-compose.yml dosyası, tüm sistemi tek bir komut (docker-compose up) ile çalışır hale getirir. Bu yapı sayesinde servisler ortak bir ağ üzerinde haberleşir, veritabanı gibi kalıcı bileşenler için volume kullanılarak veri kaybı önlenir. Böylece geliştirme ve üretim ortamları arasındaki farklar ortadan kalkar, yeni bir geliştiricinin projeyi kurma süresi önemli ölçüde kısalır.

Son olarak, bu yaklaşım yalnızca geliştirme sürecini kolaylaştırmakla kalmaz, aynı zamanda ileride Kubernetes gibi daha gelişmiş orkestrasyon çözümlerine geçiş için de sağlam bir temel oluşturur. Docker imajlarının CI/CD süreçlerine entegre edilmesiyle birlikte, her yeni sürüm güvenilir şekilde paketlenebilir ve dağıtılabilir hale gelir.

2. Teknoloji ve Kütüphane Seçimi

2.1. Metin Tanıma (OCR)

Belgeler üzerinden Ad, Soyad ve TCKN gibi metin tabanlı verilerin çıkarılması için temel teknoloji olarak açık kaynaklı Tesseract OCR motorunun ve Python dilindeki pytesseract kütüphanesinin kullanılması planlanmaktadır.

Tesseract, güçlü bir Türkçe desteğine sahiptir, ücretsizdir ve geniş bir topluluk tarafından desteklenmektedir. Temel metin tanıma işlemleri için yeterli bir performans sunmasına rağmen düşük çözünürlüklü, eğik, gölgeli veya üzerinde parlama olan görsellerde düşük performans göstermeye yatkındır.

RAG ve LLM, Tesseract'in bu dezavantajını telafi etmek ve sistemin esnekliğini artırmak için kullanılacaktır.Tesseract, belgeden metni düz metin formatında çıkarır. Bu ham metinden doğru bilgileri (Ad, Soyad vb.) ayıklamak için RegEx kullanmak, belge formatlarındaki çeşitlilik nedeniyle kırılgandır ve farklı yapılar karşısında başarısız olabilir.

Tesseract tarafından çıkarılan ham metin, bir LLM'e girdi olarak verilecektir. LLM, bu metni bağlamıyla birlikte analiz ederek istenen bilgileri yapılandırılmış (JSON) formatta ayıklayacaktır. Bu yöntem, belge tasarımlarındaki çeşitliliğe karşı çok daha esnek ve dayanıklı bir çözüm sunar.

2.2. İmza Karşılaştırma

İki imzanın benzerliğini ölçmek, yalnızca piksel düzeyinde karşılaştırma yapmaktan öte; imzanın stilini, eğimini ve karakteristik özelliklerini anlamayı gerektiren karmaşık bir görevdir.

Geleneksel görüntü işleme yöntemleri (örneğin OpenCV ile yapılan özellik eşleştirme), bu tür ince yapısal farkları yakalamakta yetersiz kalabilir. Bu nedenle, bu problem için derin öğrenme tabanlı modern bir yaklaşım tercih edilmiştir.

Spesifik olarak, Siamese Network (Siyam Ağı) mimarisi kullanılacaktır. Bu mimari, iki görüntüyü aynı anda işleyerek her birinden birer özellik vektörü (embedding) çıkarır. Ardından bu vektörler arasındaki mesafe ölçülerek bir benzerlik skoru hesaplanır. Bu skor, önceden belirlenmiş bir eşik değeriyle karşılaştırılarak imzaların "uyumlu" veya "uyumsuz" olduğu belirlenir.

2.3. API Geliştirme

Sistemin API Gateway bileşeni için Python tabanlı FastAPI framework'ü tercih edilmiştir. Bu tercih, hem teknik gereksinimlere hem de geliştirici deneyimine yönelik avantajlar sunar. FastAPI, ASGI mimarisi sayesinde asenkron programlamayı destekleyerek belge yükleme gibi I/O ağırlıklı işlemlerde yüksek performans sağlar. Ayrıca Pydantic tabanlı veri doğrulama sistemi, gelen isteklerin otomatik olarak kontrol edilmesini mümkün kılar; bu da kod tekrarını azaltırken hata riskini düşürür. Kod yazımı sırasında kendiliğinden oluşan OpenAPI (Swagger) dokümantasyonu, mikroservisler arası iletişimi açık ve sürdürülebilir hale getirir. OCR ve imza karşılaştırma servislerinin Python ekosisteminde geliştirilecek olması da, API katmanının aynı dilde olmasıyla birlikte geliştirme sürecini sadeleştirir ve entegrasyonu kolaylaştırır.

3. Uygulamanın Çalıştırılması

Projeyi çeşitli şekillerde çalıştırabilirsiniz.

3.1. Docker Hub

Docker kurulu bir sistemde, aşağıdaki komutla doğrudan çalıştırabilirsiniz:

- docker run -p 8000:8000 hootbu/doc-auth-system

Ardından tarayıcıdan https://localhost:8000 adresine giderek uygulamayı kullanabilirsiniz.

3.2 Yerel Ortamda (Local)

GitHub reposundaki LOCAL_prototip_web_app klasörünü klonlayarak aşağıdaki komutlarla çalıştırabilirsiniz.

- git clone https://github.com/hootbu/doc-auth-system
- cd doc-auth-system/kodlar/LOCAL_prototip_web_app
- uvicorn main:app --reload

3.3. GitHub'daki Dockerize Edilmiş Versiyonu Build Ederek

GitHub reposundaki prototip_web_app klasörünü klonlayıp aşağıdaki adımları izleyerek kendi imajınızı oluşturabilir ve çalıştırabilirsiniz.

- git clone https://github.com/hootbu/doc-auth-system
- cd doc-auth-system/kodlar/prototip_web_app

- docker build -t doc-auth-system .
- docker run -p 8000:8000 doc-auth-system

Projeyi çalıştırdığınızda test olarak kullanabileceğiniz çeşitli görselleri repo içerisindeki test_imgs klasöründe bulabilirsiniz.

4. Potansiyel İyileştirmeler ve Riskler

4.1. Sistemin Zayıf Yönleri ve Potansiyel Riskler

Geliştirilen prototip sistem, temel belge doğrulama işlevselliğini sergilemekle birlikte, gerçek dünya uygulamalarında karşılaşılabilecek çeşitli riskler ve sınırlamalar içermektedir. Sistemin doğruluğu, büyük ölçüde optik karakter tanıma (OCR) teknolojisinin performansına dayanmaktadır. Tesseract OCR motoru gibi araçların başarısı, girdi olarak alınan görsellerin kalitesine – çözünürlük, ışık koşulları, eğiklik ve netlik gibi faktörlere – son derece duyarlıdır. Prototip geliştirme sürecinde de gözlemlendiği üzere, OCR çıktısındaki küçük farklılıklar veya metin dışı gürültüler dahi, veriyi ayıklamak için kullanılan RegEx (Düzenli İfadeler) kurallarının kırılganlığına neden olmaktadır. Gerek kimlik kartındaki gerekse formlardaki bilgi alanlarının göreceli konumlarının veya Tesseract tarafından okunma biçimlerinin her görselde farklılık gösterebilmesi, tüm olası durumları kapsayan tek bir RegEx kural seti oluşturmayı pratik olarak zorlaştırmaktadır. Bu durum, farklı belge formatları veya beklenmedik karakterler (örneğin el yazısı) söz konusu olduğunda daha da belirginleşir. Bununla birlikte, mevcut prototip, sunulan belgelerin sahte olup olmadığını veya üzerinde dijital oynama yapılıp yapılmadığını kontrol etme yeteneğine sahip değildir. Ayrıca, prototipte yer almayan imza karşılaştırma işlevi de tam bir doğrulama süreci için kritik öneme sahiptir ve farklı belge türleri (pasaport, ehliyet vb.) için de destek bulunmamaktadır.

4.2. Risk Azaltma Yöntemleri

Belirtilen riskleri azaltmak ve sistemin güvenilirliğini artırmak için çeşitli iyileştirmeler planlanmalıdır. Öncelikle, sisteme yüklenen görsellerin kalitesini artırmaya yönelik adımlar atılmalıdır. Kullanıcı tarafında minimum çözünürlük gibi kriterler zorlanabilirken, sunucu tarafında eğiklik düzeltme, perspektif düzeltme, parlama giderme gibi gelişmiş görüntü ön işleme teknikleri uygulanarak OCR doğruluğu artırılabilir. Ancak, özellikle RegEx ile veri ayıklamanın karşılaştığı temel zorluklar düşünüldüğünde, daha köklü bir çözüm gerekmektedir. Bu noktada, daha yüksek doğruluk sunan ticari OCR servisleri veya yapay zeka tabanlı yaklaşımlar devreye girmelidir. Özellikle Büyük Dil Modelleri (LLM), yapılandırılmamış OCR çıktısını anlamlandırarak, farklı formatlardaki belgelerden bile istenen veriyi yapısal bir formatta çıkarma potansiyeli sunar. Sistemin gelecekte daha fazla belge türünü destekleyebilmesi ve yüksek doğrulukla çalışabilmesi için LLM tabanlı bir veri çıkarma katmanına geçiş yapılması stratejik bir gereklilik olarak öne çıkmaktadır. Sahtecilik riskini azaltmak için ise, kullanıcıdan kısa bir video kaydı (canlılık kontrolü - liveness check) istenebilir veya kimlik kartı üzerindeki hologram, MRZ alanı gibi fiziksel güvenlik öğelerinin varlığı görüntü işleme teknikleriyle analiz edilebilir.

İmzaların doğrulanması için Siamese Networks gibi derin öğrenme modelleri veya uzmanlaşmış çözümler kullanılmalıdır. Son olarak, sisteme yüklenen belgenin türünü otomatik olarak tanıyan ve buna uygun işleme modelini tetikleyen bir sınıflandırma mekanizması eklemek, sistemin esnekliğini artıracaktır.

4.3. Sistemin Gelecekte Daha Akıllı ve Güvenilir Hale Getirilmesi

Prototip aşamasında elde edilen tecrübeler, geleneksel OCR ve kural tabanlı (RegEx) veri ayıklama yöntemlerinin, geniş ölçekli ve çeşitli kullanım senaryolarına uygun, güvenilir bir sistem sunmak için gereken doğruluk, esneklik ve dayanıklılık seviyesini sağlamakta zorlandığını göstermektedir. Bu nedenle, sistemin geleceği yapay zeka (AI), özellikle de Makine Öğrenmesi (ML) ve Derin Nöral Ağlar (DNN) üzerine inşa edilmelidir. Bu teknolojiler, farklı kalitelerdeki görsellerden yüksek doğrulukla metin okuma, belge düzenini anlama ve yapılandırılmış veri çıkarma gibi temel görevlerde üstün performans sergilemektedir. Özellikle Konvolüsyonel Nöral Ağlar (CNN), Tekrarlayan Nöral Ağlar (RNN/LSTM) ve Transformer tabanlı modeller (LayoutLM, LLM'ler), geleneksel yöntemlerin kırılganlığını ortadan kaldırarak çok çeşitli belge formatlarına adaptasyonu mümkün kılar.

Bu AI yeteneklerini sisteme entegre etmek için temelde iki stratejik yol bulunmaktadır: hazır ticari/bulut API'lerini kullanmak veya özel modeller geliştirmek. Google (Document AI), Microsoft (Azure AI Vision) gibi platformların sunduğu hazır API'ler, geliştirme sürecini hızlandırma ve başlangıçta daha düşük Ar-Ge maliyeti avantajları sunar. Ancak, bu servislerin kullanımına bağlı maliyetler ve veri gizliliği veya özelleştirme konularındaki olası kısıtlamalar dikkatle değerlendirilmelidir. Diğer yandan, kendi özel ML modellerimizi geliştirmek, daha yüksek bir başlangıç yatırımı gerektirse de, uzun vadede operasyonel verimlilik, tam veri kontrolü, belirli ihtiyaçlara göre optimize edilmiş performans ve alana özgü yetenekler geliştirme potansiyeli sunar. Bu bağlamda hibrit bir yaklaşım mantıklı olabilir: Temel fonksiyonlar için ticari API'lerle başlayıp, eş zamanlı olarak en kritik veya yüksek hacimli işlemler için optimize edilmiş özel modeller geliştirmeye yatırım yapmak.

Sistemin başarısı için sadece doğruluk yeterli değildir; güvenilirlik ve sürekli iyileşme de kritik öneme sahiptir. Sahtecilik tespiti (anomali detection, liveness check, güvenlik öğesi analizi) ve imza doğrulama gibi katma değerli AI yetenekleri, sistemin yetkinliğini artıracaktır. Yetkili kullanıcıların bildirimlerinden öğrenen mekanizmalar ve modellerin düzenli olarak güncellenmesi, sistemin zamanla daha da akıllanmasını ve yeni zorluklara karşı dirençli kalmasını sağlayacaktır. Yasal uyumluluk çerçevesinde resmi veri kaynaklarıyla entegrasyonlar, doğrulama süreçlerine ek bir güven katmanı ekleyebilir. Son olarak, kurumsal düzeyde bir çözüm için, tüm bu yeteneklerin sağlam bir güvenlik altyapısı (şifrelenmesi, erişim kontrolü, denetim logları) ile korunması ve diğer iş sistemleriyle kolay entegrasyon sağlayan esnek API'ler aracılığıyla sunulması, platformun ölçeklenebilir ve sürdürülebilir olmasının temelini atacaktır.

Kaynakça

[1] Amazon. (n.d.). *Monolithic vs microservices - difference between software development architectures- AWS*. Amazon Web Services. https://aws.amazon.com/compare/the-difference-between-monolithic-and-microservices-architecture/

[2] Nag, R. (2022, November 19). *A comprehensive guide to siamese neural networks*. Medium. https://medium.com/@rinkinag24/a-comprehensive-guide-to-siamese-neural-networks-3358658c0513

[3] Google. (n.d.). *Document AI | google cloud*. Google. https://cloud.google.com/document-ai

[4] *Azure ai vision with OCR and AI: Microsoft Azure*. with OCR and AI | Microsoft Azure. (n.d.). https://azure.microsoft.com/en-us/products/ai-services/ai-vision

[5] Yorgun, E. (n.d.). *Project's Docker Hub Page*. Docker. https://hub.docker.com/r/hootbu/doc-auth-system

[6] Yorgun, E. (n.d.-a). *Hootbu/Doc-AUTH-System*. GitHub. https://github.com/hootbu/doc-auth-system