MucizeOl - Backend Geliştirme Raporu (Mikroservis Mimarisi - Paylaşımlı Veritabanı Modeli)

Proje: MucizeOl - Hayvan Sahiplenme Platformu

Hazırlayan: Ahmet Bera ÇELİK

Tarih: 4 Kasım 2025

Amaç: Bu belge, "MucizeOl" projesinin "Paylaşımlı Veritabanı" (Shared Database) modelini kullanan mikroservis altyapısını, servis sorumluluklarını, API sözleşme stratejisini ve lokal geliştirme (DevOps) süreçlerini tanımlar.

1. Mimariye Genel Bakış

Proje, 4 ana servisten oluşan bir Mikroservis mimarisi benimseyecektir. Tüm servisler **tek bir merkezi veritabanını (mucizeol_db)** paylaşacaktır.

1.1. Teknoloji Stack'i (Backend)

- **Dil/Framework:** Java , Spring Boot
- **Veritabanı:** MySQL
- **Migration Yönetimi:** Flyway
- **Dosya Depolama:** DigitalOcean Spaces (S3 Uyumlu)
- **Proje Yönetimi:** Maven
- **API Sözleşmesi:** OpenAPI 3 (Swagger)
- **Lokal Orkestrasyon:** Docker Compose
- **CI/CD (Prod):** GitHub, Kubernetes, ArgoCD

1.2. Mimari Bileşenleri (Servisler)

1. **gateway-service (API Gateway):** Güvenlik (JWT) ve yönlendirme merkezi.
2. **auth-service (Kullanıcı & Yetkilendirme Servisi):** Kullanıcı, rol ve token yönetiminden sorumlu.
3. **listing-service (İlan Servisi):** İlan ve ilan meta verilerinin (şehir, tür) yönetiminden sorumlu.
4. **request-service (Talep Servisi):** Sahiplenme taleplerinin yönetiminden sorumlu.

2. Merkezi Veritabanı Mimarisi (mucizeol_db)

Mimari, "Database-per-Service" modeli yerine "Shared Database" (Paylaşımlı Veritabanı) modelini kullanır. 4 servis de aynı mucizeol_db veritabanına bağlanacaktır.

2.1. Flyway ve Veritabanı Yönetimi

- Tüm veritabanı şeması (mucizeol_db) **Flyway** ile yönetilecektir.
- Tüm SQL migration script'leri (V1__init_tables.sql vb.) tek bir yerde (muhtemelen auth-service projesinin resources/db/migration klasöründe) tutulacaktır.
- **Strateji:** Veritabanı kilitlenmelerini (race conditions) önlemek için, **sadece bir servis** (örn: auth-service) migration'ları çalıştırmaktan (spring.flyway.enabled=true) sorumlu olacaktır.
- Diğer servisler (listing-service, request-service) ya migration'ı pas geçecek (spring.flyway.enabled=false) ya da sadece mevcut şemayı doğrulayacaktır (spring.flyway.validate=true).

2.2. Paylaşımlı Tablolar (mucizeol_db)

auth-service'in Ana Sorumluluğundaki Tablolar:

- users: (user_id, first_name, last_name, email, password_hash, phone_number, role_id, refresh_token_hash, refresh_token_expires_at, created_at)
- roles: (role_id, role_name)

listing-service'in Ana Sorumluluğundaki Tablolar:

- listings: (listing_id, **user_id** (FK -> users), title, description, **image_url** (String - DigitalOcean Spaces'teki dosya URL'si), animal_type_id, animal_breed_id, city_id, age, gender, status, created_at, updated_at)
- cities: (city_id, city_name)
- animal_types: (type_id, type_name)
- animal_breeds: (breed_id, type_id, breed_name)

request-service'in Ana Sorumluluğundaki Tablolar:

- adoption_requests: (request_id, **user_id** (FK -> users), **listing_id** (FK -> listings), status, request_message, created_at)

3. Servis Sorumlulukları ve API Sözleşmeleri (OpenAPI)

API Gateway (:8080) hala tek giriş noktasıdır ve güvenlikten sorumludur. İç servisler (8081, 8082, 8083) tüm iş mantığını içerir.

3.1. gateway-service (Spring Cloud Gateway)

- **Port:** :8080
- **Veritabanı:** Yok.
- **Ana Sorumlulukları:** Yönlendirme (Routing), Güvenlik (Global Filter - JWT), CORS Yönetimi.

3.2. auth-service

- **Port:** :8081
- **Veritabanı:** mucizeol_db (Sorumlu: users, roles)
- **Ana Endpoint'ler:**
    - POST /api/v1/auth/register: (Public) Yeni kullanıcı hesabı oluşturur.
    - POST /api/v1/auth/login: (Public) Kullanıcı girişi yapar, Access ve Refresh token döndürür.
    - POST /api/v1/auth/refresh: (Public) Süresi dolmamış Refresh Token'ı kullanarak yeni bir Access Token alır.
    - POST /api/v1/auth/logout: (Protected) Kullanıcının mevcut Refresh Token'ını geçersiz kılarak (DB'den silerek) güvenli çıkış yapar.
    - GET /api/v1/users/me: (Protected) Giriş yapmış kullanıcının kendi profil bilgilerini döndürür.

3.3. listing-service

- **Port:** :8082
- **Veritabanı:** mucizeol_db (Sorumlu: listings, cities, animal_types, animal_breeds)
- **Dosya Depolama:** DigitalOcean Spaces'e yükleme yapmaktan sorumlu.
- **Ana Endpoint'ler:**
    - GET /api/v1/listings: (Public) Filtrelenmiş ve "Mevcut" durumdaki ilanları listeler (Paginasyonlu).
    - GET /api/v1/listings/{id}: (Public) Tek bir ilanın detaylı bilgilerini döndürür.

- POST /api/v1/listings: (Protected) Yeni bir sahiplendirme ilanı oluşturur (Gelen multipart/form-data içindeki resmi **Bölüm 5**'teki stratejiye göre yükler).
- PUT /api/v1/listings/{id}: (Protected) Sadece ilan sahibinin kendi ilanını güncellemesini sağlar.
- DELETE /api/v1/listings/{id}: (Protected) Sadece ilan sahibinin (veya Admin'in) ilanı silmesini sağlar.
- GET /api/v1/meta/cities: (Public) Filtreleme için kullanılabilir tüm şehirleri listeler.
- GET /api/v1/meta/animal-types: (Public) Filtreleme için kullanılabilir tüm hayvan türlerini listeler.
- GET /api/v1/meta/animal-types/{typeId}/breeds: (Public) Bir türe ait cinsleri listeler.

3.4. request-service

- **Port:** :8083
- **Veritabanı:** mucizeol_db (Sorumlu: adoption_requests)
- **Ana Endpoint'ler:**
  - POST /api/v1/requests: (Protected) Bir ilana sahiplenme talebi göndermeyi sağlar.
  - GET /api/v1/requests/my-requests: (Protected) Kullanıcının gönderdiği taleplerin listesini döndürür.
  - GET /api/v1/requests/my-listings-requests: (Protected) Kullanıcının ilanlarına gelen taleplerin listesini döndürür.
  - POST /api/v1/requests/{id}/approve: (Protected) İlan sahibinin, ilanına gelen bir talebi onaylamasını sağlar.
  - POST /api/v1/requests/{id}/reject: (Protected) İlan sahibinin, ilanına gelen bir talebi reddetmesini sağlar.

4. Servisler Arası İletişim Akışı (Veritabanı Transaction Modeli)

Senaryo: İlan Sahibi Talebi Onayladığında (POST /api/v1/requests/{id}/approve)

1. **Frontend** -> POST :8080/api/v1/requests/123/approve (Token ile)

2. **gateway-service** (:8080): Token'ı doğrular. X-User-ID: 5 header'ını ekler. İsteği request-service'e yönlendirir.
3. request-service (:8083): İsteği alır ve bir Veritabanı Transaction'ı (@Transactional) başlatır.

a. Gerekli verileri almak için sorgu atar (örn: adoption_requests ve listings tablolarına JOIN yaparak).

b. X-User-ID (5) ile ilanın sahibinin ID'sinin (5) eşleştiğini doğrular. (Yetki kontrolü).

c. Kendi sorumlu olduğu tabloyu günceller: UPDATE adoption_requests SET status = 'Onaylandı' WHERE request_id = 123.

d. listing-service'in sorumlu olduğu tabloyu günceller: UPDATE listings SET status = 'Sahiplendirildi' WHERE listing_id = 77.

4. **request-service**, Transaction'ı **Commit** eder. Her iki UPDATE işlemi de başarılı olursa 200 OK yanıtı döner.

5. Statik Dosya (Görsel) Depolama Stratejisi

Uygulama, Docker ve Kubernetes üzerinde "stateless" (durumsuz) çalışacak şekilde tasarlanmıştır. Bu nedenle, hayvan resimleri gibi kalıcı dosyalar servislerin kendi container'larında (geçici dosya sisteminde) **saklanamaz.**

- **Çözüm:** Nesne Depolama (Object Storage) kullanılacaktır.
- **Servis Sağlayıcı: DigitalOcean Spaces** (S3-Uyumlu API).
- **Sorumlu Servis:** listing-service, resim yükleme (upload) mantığından sorumlu tek servistir.
- **İş Akışı (POST /api/v1/listings):**
    1. listing-service, kullanıcıdan gelen multipart/form-data içindeki resim dosyasını alır.
    2. Standart **AWS S3 SDK**'sını kullanarak (DigitalOcean Spaces endpoint'ine bağlanarak) dosyayı mucizeol-images adlı "bucket"a yükler.
    3. DigitalOcean Spaces, yükleme başarılı olduğunda kalıcı bir URL (örn: https://mucizeol-images.fra1.digitaloceanspaces.com/kedi-123.jpg) döndürür.
    4. listing-service, bu URL'yi listings tablosundaki image_url sütununa kaydeder.

6. Lokal Geliştirme (Docker Compose) (Önceki 5. Bölüm)

docker-compose.yml dosyası şunları içerecektir:

1. **MySQL Servisi:** mucizeol_db adında tek bir veritabanı şeması oluşturan MySQL container'ı.
2. **auth-service:** 8081 portunda. (Flyway migration'larını bu servis çalıştırabilir).
3. **listing-service:** 8082 portunda.
4. **request-service:** 8083 portunda.
5. gateway-service: 8080 portunda.

Not: Lokal geliştirme sırasında DigitalOcean Spaces yerine, docker-compose.yml'e MinIO (S3 uyumlu self-hosted servis) eklenebilir.

7. Git ve DevOps Stratejisi (Önceki 6. Bölüm)

- **mucizeOl Reposu:** backend ve frontend ana klasörlerini içerir.
- **mucizeOlManifests Reposu:** Kubernetes ve ArgoCD manifestlerini içerecektir.