MucizeOl - Veritabanı Şeması ve İşleyiş Raporu

Proje: MucizeOl - Hayvan Sahiplenme Platformu

Hazırlayan: Ahmet Bera ÇELİK

Amaç: Bu belge, projenin backend mimarisini (Java Spring Boot / Spring Data JPA) destekleyecek olan ilişkisel veritabanı şemasını (MySQL) detaylandırmak, her bir tablonun ve attribute'ün işlevsel amacını açıklamak ve tablolar arası ilişkileri tanımlamaktır.

1. Veritabanı Genel Bakış

Şema, 8 tablodan oluşmaktadır. Bu yapı, üç ana işlevsel gruba ayrılmıştır:

1. **Kullanıcı ve Güvenlik Mimarisi:** (users, roles, user_refresh_tokens) - Kimlik doğrulama, yetkilendirme ve güvenli oturum yönetiminden sorumludur.
2. **Temel İş Mantığı Mimarisi:** (listings, adoption_requests) - Uygulamanın ana amacı olan ilan verme ve talep toplama işlevlerini yönetir.
3. **Normalizasyon ve Filtreleme Mimarisi:** (cities, animal_types, animal_breeds) - Veri bütünlüğünü sağlayan, depolama alanını optimize eden ve arayüzdeki filtreleme işlemlerini besleyen "Lookup" tablolarıdır.

2. Tablo Detayları ve Attribute İşlevleri

2.1. Kullanıcı ve Güvenlik Mimarisi

users (Kullanıcılar)

Uygulamaya kayıt olan her bir bireyi temsil eder. İlan sahibi veya sahiplenmek isteyen kişi olabilir.

- user_id (PK): **Birincil Anahtar.** Kullanıcıyı sistemde benzersiz kılan kimlik numarasıdır. Tüm ilişkisel tablolarda (listings, adoption_requests vb.) bu ID referans alınır.
- first_name: (String) **İşlev:** Kullanıcının adını tutar. Arayüzde ("Hoş geldin Ahmet"), ilan detaylarında ("İlan Sahibi: Ahmet Bera") ve talep listelerinde gösterilir.
- last_name: (String) **İşlev:** Kullanıcının soyadını tutar. (Bkz: first_name).
- email: (String, Unique) **İşlev:** Kullanıcının sisteme giriş (login) yaparken kullanacağı kullanıcı adıdır. UNIQUE kısıtlaması, aynı e-posta ile birden fazla hesap açılmasını veritabanı seviyesinde engeller.

- password_hash: (String) **İşlev:** Kullanıcının şifresinin **asla** düz metin (plain-text) olarak saklanmaması içindir. Spring Security BCrypt gibi algoritmalarla hash'lenmiş (şifrelenmiş) halini tutar.
- phone_number: (String) **İşlev:** Kullanıcının profilindeki iletişim bilgisidir. Sahiplenme talebi onaylandığında ilan sahibi ile iletişime geçilmesi için kullanılabilir.
- role_id: (FK -> roles) **İşlev:** Kullanıcının yetkisini belirler. Bu ID, roles tablosundaki "ROLE_USER" veya "ROLE_ADMIN" girdisine işaret eder. Spring Security, kullanıcının hangi endpoint'lere (örn: /admin) erişebileceğine bu alana bakarak karar verir.
- refresh_token_hash: (String, Nullable) **İşlev**: Kullanıcının en son aktif olan Refresh Token'ının hash'lenmiş halini tutar. Yeni bir giriş yapıldığında, bu alandaki eski token hash'inin üzerine yenisi yazılır, böylece eski oturum geçersiz kılınır ("Tek Oturum" garantisi).
- refresh_token_expires_at: (Timestamp, Nullable) **İşlev**: refresh_token_hash'in son kullanma tarihini tutar. Backend, bu tarihi geçmiş token'ları reddeder.
- created_at: (Timestamp) **İşlev:** Denetim (audit) amaçlıdır. Kullanıcının platforma ne zaman kayıt olduğunu gösterir.

roles (Roller)

Kullanıcı yetki seviyelerini tanımlayan bir "Lookup" tablosudur.

- role_id (PK): **Birincil Anahtar.** Rolü benzersiz kılan kimliktir. (users.role_id tarafından referans alınır).
- role_name: (String, Unique) **İşlev:** Yetkinin metinsel karşılığıdır (örn: "ROLE_USER", "ROLE_ADMIN"). Spring Security bu metinleri okuyarak yetki kontrolü yapar. UNIQUE olması veri tutarlılığını garantiler.

2.2. Temel İş Mantığı Mimarisi

listings (İlanlar)

Uygulamanın ana içeriğidir. Sahiplendirilmeyi bekleyen hayvanların tüm detaylarını tutar.

- listing_id (PK): **Birincil Anahtar.** İlanı benzersiz kılan kimliktir. İlan detay sayfasına giderken (/ilan/{listing_id}) ve talep oluştururken kullanılır.
- user_id: (FK -> users) **İşlev:** İlanın sahibini (ilanı kimin oluşturduğunu) belirtir. "İlanlarım" sayfası bu ID'ye göre filtrelenir.

- title: (String) **İşlev:** İlanın başlığıdır. Ana sayfadaki ilan kartlarında ve ilan detay sayfasının en üstünde gösterilir.
- description: (Text) **İşlev:** İlanın uzun açıklamasıdır. Sadece ilan detay sayfasında gösterilir.
- image_url: (String) **İşlev:** (MVP sadeleştirmesi) İlanın **tek** fotoğrafının adresini (URL) tutar. Ana sayfadaki ilan kartlarında ve detay sayfasında bu fotoğraf gösterilir.
- animal_type_id: (FK -> animal_types) **İşlev:** Filtreleme ve veri kategorizasyonu içindir. İlanın "Kedi", "Köpek" vb. hangi türe ait olduğunu belirtir.
- animal_breed_id: (FK -> animal_breeds) **İşlev:** Filtreleme ve veri kategorizasyonu içindir. İlanın "Tekir", "Golden" vb. hangi cinse ait olduğunu belirtir.
- city_id: (FK -> cities) **İşlev:** Hayvanın bulunduğu lokasyonu belirtir. Ana sayfadaki **en önemli filtreleme** kriteridir (örn: "İstanbul'daki ilanlar").
- age: (Integer) **İşlev:** Hayvanın yaşını tutar. İlan detayında gösterilir.
- gender: (String) **İşlev:** Hayvanın cinsiyetini ("Erkek", "Dişi") tutar. İlan detayında gösterilir.
- status: (String) **İşlev: Kritik iş kuralı attribute'ü.** İlanın mevcut durumunu ("Mevcut", "Sahiplendirildi", "Askıda") tutar. Ana sayfada sadece "Mevcut" olanlar listelenir. Bir talep onaylandığında ilan durumu "Sahiplendirildi" olarak güncellenir ve yeni talep alması engellenir.
- created_at: (Timestamp) **İşlev:** İlanın oluşturulma tarihidir. Ana sayfada ilanları "En Yeniye Göre" sıralamak için bu sütun kullanılır (ORDER BY created_at DESC).
- updated_at: (Timestamp) **İşlev:** İlanın son güncellenme tarihini tutar.

adoption_requests (Sahiplenme Talepleri)

Kullanıcıların ilanlara yaptığı "sahiplenmek istiyorum" başvurularını yönetir.

- request_id (PK): **Birincil Anahtar.** Her bir talebi benzersiz kılar. İlan sahibinin bir talebi onaylaması/reddetmesi işlemi bu ID üzerinden yapılır (örn: .../api/requests/{request_id}/approve).
- user_id: (FK -> users) **İşlev:** Talebi **yapan** kullanıcıyı belirtir.
- listing_id: (FK -> listings) **İşlev:** Talebin **hangi ilana** yapıldığını belirtir.

- status: (String) **İşlev:** Talep sürecinin (workflow) mevcut durumunu tutar ("Beklemede", "Onaylandı", "Reddedildi"). İlan sahibi bu durumu günceller.
- request_message: (Text) **İşlev:** Talepte bulunan kullanıcının ilan sahibine gönderdiği ilk mesajdır ("Neden sahiplenmek istiyorum?"). İlan sahibinin talebi değerlendirmesi için kritik öneme sahiptir.
- created_at: (Timestamp) **İşlev:** Talebin gönderilme tarihidir. İlan sahibinin "Gelen Talepler" kutusunu "en yeniye göre" sıralaması için kullanılır.
- **Kısıtlama:** (user_id, listing_id) üzerinde UNIQUE bir kısıtlama (constraint) olmalıdır. **İşlev:** Bir kullanıcının aynı ilana birden fazla mükerrer talep göndermesini veritabanı seviyesinde engeller.

2.3. Normalizasyon ve Filtreleme Mimarisi (Lookup Tables)

Bu tabloların genel amacı; veri tekrarını önlemek (örn: "İstanbul" kelimesini listings tablosunda binlerce kez yazmak yerine sadece ID'sini yazmak), veri bütünlüğünü garantilemek (yazım hatalarını engellemek) ve filtreleme sorgularını hızlandırmaktır.

cities (Şehirler)

- city_id (PK): **Birincil Anahtar.** Şehrin benzersiz ID'sidir.
- city_name: (String) **İşlev:** Arayüzdeki "Şehir Seçiniz" dropdown menüsünü doldurmak ve ilan detayında (city_id yerine) "İstanbul" gibi okunabilir metin göstermek için kullanılır.

animal_types (Hayvan Türleri)

- type_id (PK): **Birincil Anahtar.** Türün benzersiz ID'sidir.
- type_name: (String) **İşlev:** Arayüzdeki "Tür Seçiniz" (Kedi, Köpek) filtresini doldurmak için kullanılır.

animal_breeds (Hayvan Cinsleri)

- breed_id (PK): **Birincil Anahtar.** Cinsin benzersiz ID'sidir.
- type_id: (FK -> animal_types) **İşlev:** Cinsin hangi türe ait olduğunu belirtir (örn: "Tekir" -> "Kedi"). Arayüzde kullanıcı "Kedi" türünü seçtiğinde, "Cins Seçiniz" menüsüne sadece type_id'si Kedi'ye ait olan cinslerin (Tekir, Siyam vb.) gelmesini (dinamik/kademeli filtreleme) sağlar.
- breed_name: (String) **İşlev:** Arayüzdeki "Cins Seçiniz" filtresini doldurmak için kullanılır.

3. Tablolar Arası İlişkiler (Relationships)

Bu şema, Spring Data JPA'de @OneToMany, @ManyToOne ve @OneToOne annotasyonları ile yönetilecek olan standart ilişkisel bağlantıları kullanır.

- **roles <-> users (One-to-Many)**
    - Bir Rol (örn: ROLE_USER) birden çok Kullanıcıya sahip olabilir.
    - Bir Kullanıcı sadece bir Role sahip olabilir.
- **users <-> user_refresh_tokens (One-to-Many)**
    - Bir Kullanıcı birden çok RefreshToken'a (aktif oturuma) sahip olabilir. Bu, çoklu cihaz desteğini sağlar.
    - Bir RefreshToken sadece bir Kullanıcıya aittir.
- **users <-> listings (One-to-Many)**
    - Bir Kullanıcı (ilan sahibi) birden çok İlan oluşturabilir.
    - Bir İlan sadece bir Kullanıcıya (sahibe) aittir.
- **users <-> adoption_requests (One-to-Many)**
    - Bir Kullanıcı (sahiplenmek isteyen) birden çok Talep gönderebilir.
    - Bir Talep sadece bir Kullanıcı tarafından gönderilebilir.
- **listings <-> adoption_requests (One-to-Many)**
    - Bir İlana birden çok Talep gelebilir.
    - Bir Talep sadece bir İlan için yapılabilir.
- **animal_types <-> animal_breeds (One-to-Many)**
    - Bir Hayvan Türü (Kedi) birden çok Hayvan Cinsine (Tekir, Siyam, Sarman) sahip olabilir.
    - Bir Hayvan Cinsi (Tekir) sadece bir Hayvan Türüne (Kedi) aittir.
- **cities <-> listings (One-to-Many)**
    - Bir Şehirde birden çok İlan olabilir.
- **animal_types <-> listings (One-to-Many)**
    - Bir Hayvan Türü (Kedi) kategorisinde birden çok İlan olabilir.

- **animal_breeds <-> listings (One-to-Many)**
    - Bir Hayvan Cinsi (Tekir) kategorisinde birden çok İlan olabilir.