Emlak İşletmeleri Project SDD

VERSION 1.0

10.10.25

EDA KOCAMAN

1. Giriş

1.1. Amaç

Bu doküman, Emlak İşletmeleri Web Uygulamasının yazılım mimarisi, veri modeli, katman yapısı ve arayüz tasarımını tanımlar.

1.2. Version Tarihi

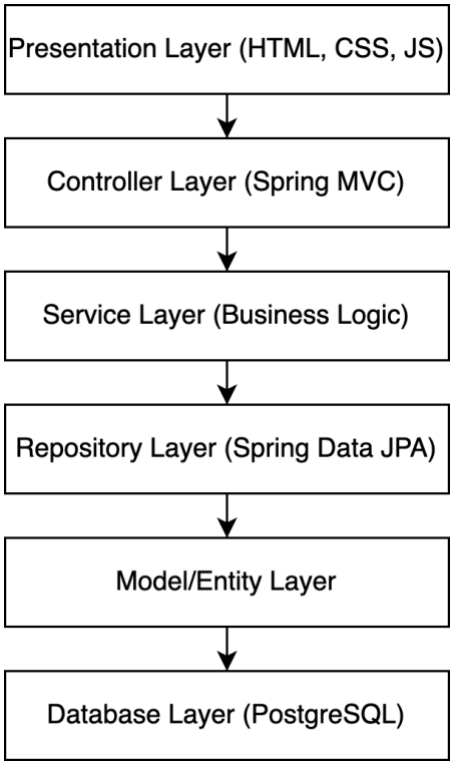
Version No	Değişiklik Sebebi	Tarih
1.0	İlk Yayın	10.10.25

1.3. Terimler Sözlüğü

- **Company:** Emlak işletmesinin kayıtlı olduğu işyeri bilgileri (ad, adres, telefon vb.)
- **Employee:** Şirkette çalışan ve sistemi kullanan personel (admin veya staff rolünde)
- **Customer:** Sistemdeki kullanıcı; alıcı, satıcı veya her ikisi olabilir
- **Property:** Müşteriler tarafından ilan verilen emlak kayıtları (konut, işyeri vb.)
- **PropertyRequest:** Alıcı müşterinin belirlediği kriterlere göre aradığı emlak talepleri
- **Role:** Employee'in yetkisini belirten enum değer; admin veya staff
- **Customer_type:** Müşteri türünü belirten enum değer; buyer, seller veya both

2. System Architecture

Uygulama çok katmanlı mimari kullanılarak geliştirilecektir:



Şekil 1

Uygulama, katmanlı bir mimari ile tasarlanmıştır.

Presentation Layer (UI):

- Kullanıcı ile sistem arasındaki etkileşimi sağlar.
- React, HTML5, CSS3 ve JavaScript kullanılarak web tabanlı arayüzler geliştirilir.
- Kullanıcı girişleri toplanır ve Controller katmanına iletilir.

Controller Layer:

- HTTP isteklerini alır ve doğrulama yapar.
- İş mantığını uygulayan Service katmanına yönlendirir.
- RESTful endpointler aracılığıyla veri transferi sağlar.
- Hataları ve istisnaları kullanıcıya uygun biçimde iletir.

Service Layer:

- İş mantığını uygular ve kuralları denetler.
- Transaction yönetimi ve exception handling burada gerçekleştirilir.
- Repository katmanı ile veri erişimi yapar ve Controller'a yanıt döner.

Repository Layer:

- Veritabanı işlemlerini soyutlar.
- Spring Data JPA veya ORM kullanılarak veri erişimi sağlanır.
- Entity sınıfları aracılığıyla veriler model katmanına aktarılır.

Model/Entity Layer:

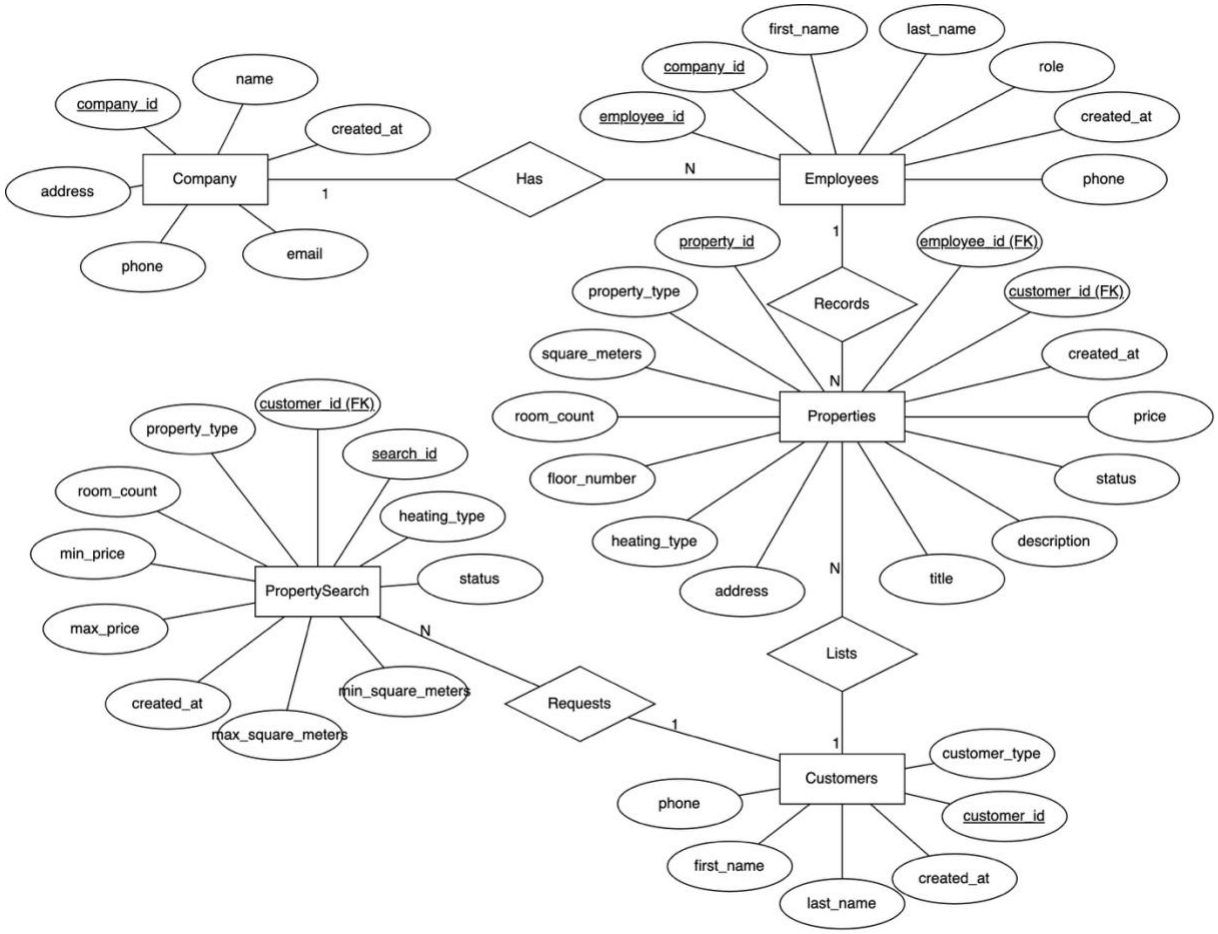
- Veritabanı tablolarının Java sınıfları ile temsilidir.
- Veri bütünlüğü ve ilişki kuralları burada tanımlanır.

Database Layer: PostgreSQL / tercih edilen veritabanı

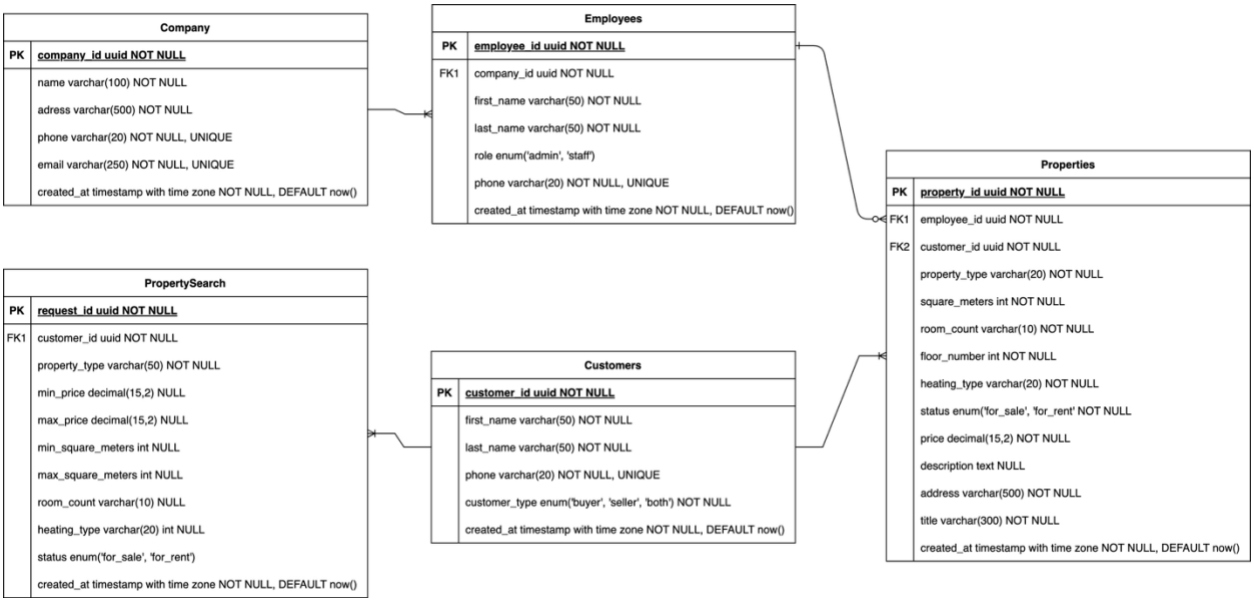
- PostgreSQL gibi ilişkisel veritabanı kullanılır.
- Tablolar ve ilişkiler, Entity sınıfları üzerinden katmanlar arası veri aktarımını destekler.

Katmanlı mimari, bakım kolaylığı, test edilebilirlik ve genişletilebilirlik sağlar. Kullanıcı arayüzünden veritabanına kadar her katman sorumlulukları net bir şekilde ayrılmıştır.

3. Database Design



Şekil 2



Şekil 3

3.1. Tables (Tablolar) ve Attributes (Alanlar)

- Company
  - company_id (PK, UUID, NOT NULL)
  - name (VARCHAR, NOT NULL)
  - address (VARCHAR, NOT NULL)
  - phone (VARCHAR, NOT NULL)
  - email (VARCHAR, NUT NULL)
  - created_at (TIMESTAMP WITH TIME ZONE, NOT NULL, DEFAULT now())
- Employees
  - employee_id (PK, UUID, NOT NULL)
  - company_id (FK → Company.company_id, NOT NULL)
  - first_name (VARCHAR, NOT NULL)
  - last_name (VARCHAR, NOT NULL)
  - role (ENUM('admin','staff'), NOT NULL)
  - created_at (TIMESTAMP WITH TIME ZONE, NOT NULL, DEFAULT now())
- Customers
  - customer_id (PK, UUID, NOT NULL)
  - first_name (VARCHAR, NOT NULL)
  - last_name (VARCHAR, NOT NULL)
  - customer_type (ENUM('buyer','seller','both'), NOT NULL)
  - phone (VARCHAR, NOT NULL)
  - created_at (TIMESTAMP WITH TIME ZONE, NOT NULL, DEFAULT now())
- Properties
  - property_id (PK, UUID, NOT NULL)
  - customer_id (FK → Customers.customer_id, NOT NULL)
  - employee_id (FK → Employees.employee_id, NOT NULL)
  - property_type (VARCHAR, NOT NULL)
  - square_meters (INT, NOT NULL)
  - room_count (VARCHAR, NOT NULL)
  - floor_number (INT, NOT NULL)
  - heating_type (VARCHAR, NULL)
  - status (ENUM('for_sale','for_rent', ‘sold’, ‘rented’), NOT NULL)
  - price (DECIMAL, NOT NULL)
  - description (TEXT, NULL)
  - adress (VARCHAR, NOT NULL)
  - title (VARCHAR, NOT NULL)
  - created_at (TIMESTAMP WITH TIME ZONE, NOT NULL, DEFAULT now())

- PropertySearch
  - request_id (PK, UUID, NOT NULL)
  - customer_id (FK → Customers.customer_id, NOT NULL)
  - property_type (VARCHAR, NOT NULL)
  - min_price (DECIMAL, NULL)
  - max_price (DECIMAL, NULL)
  - min_square_meters (INT, NULL)
  - max_square_meters (INT, NULL)
  - room_count (VARCHAR, NULL)
  - heating_type (VARCHAR, NULL)
  - stasus (ENUM ('for_sale', 'for_rent')
  - created_at (TIMESTAMP WITH TIME ZONE, NOT NULL, DEFAULT now())

3.2. ER Diagram Relationships (İlişkiler)

Kaynak Entity	İlişki Adı	Hedef Entity	Kardinalite	Açıklama
Company	has	Employees	1 ↔ N	Bir şirket birden çok çalışana sahiptir
Employees	records	Properties	1 ↔ N	Bir çalışan, birden çok emlak kaydı girebilir
Customers	lists	Properties	1 ↔ N	Satıcı olan müşteri birden çok emlak ilanı açabilir
Customers	requests	PropertySearch	1 ↔ N	Alıcı olan müşteri birden çok arama talebi oluşturabilir

4. Layered Design

4.1. Controller Layer:

Amaç: HTTP isteklerini almak, doğrulamak ve uygun servis metoduna yönlendirmek.

İçerik:

- CompanyController, EmployeeController, CustomerController, PropertyController, PropertySearchController
- CRUD işlemleri için RESTful endpointler (GET, POST, PUT, DELETE)
- Request/Response DTO'ları ile veri transferi

4.2. Service Layer:

Amaç: İş mantığını ve uygulama kurallarını yönetmek.

Örnek:

- Property eklerken müşteri tipinin seller olup olmadığını kontrol eder
- Employee kaydı eklenirken şirket ID'sinin geçerliliğini doğrular
- Transaction yönetimi

4.3. Repository Layer:

Amaç: Veritabanı ile etkileşim sağlamak.

İçerik:

- Spring Data JPA veya ORM kullanımı
- Entity sınıflarıyla CRUD sorguları
- Custom sorgular için JPA query veya native query desteği

4.4. Model / Entity Layer:

Amaç: Veritabanındaki tabloların yazılım tarafındaki karşılıklarını tutmak.
İçerik:

- Company, Employee, Customer, Property, PropertyRequest Entity sınıfları
- Entity ilişkileri (OneToMany, ManyToOne)
- Validation ve constraint bilgileri

4.5. DTO Layer:

Amaç: Frontend ile backend arasındaki veri transferini yönetmek

İçerik:

- Create/Update DTO'ları (örn. EmployeeCreateDTO)
- Response DTO'ları (örn. EmployeeDTO)
- Güvenlik ve gereksiz veri sızıntısını önleme

5. Functional Modules

- **Company Management:** Şirket bilgilerini ekleme, güncelleme
- **Employee Management:** Çalışan kaydı ve yönetimi
- **Customer Management:** Alıcı ve satıcı müşteri kayıtları
- **Property Management:** Emlak ekleme, listeleme, güncelleme
- **Property Request Management:** Alıcı taleplerini kaydetme ve sorgulama

6. User Interface Design

İşyeri Ekleme:

- Sisteme işyeri ekleme.

Company Screen:

- İşyeri Bilgisi: İşyeri bilgilerini görüntüleme.
- Çalışan Ekle:Sisteme yeni çalışan ekleme.
- Çalışan Listesi: Sistemdeki tüm çalışanların listesini görüntüleme.
- Çalışan Giriş: Çalışan login.
- İlan Listesi: Sistemdeki tüm ilanların listesini görüntüleme.
- Müşteri Listesi: Sistemdeki tüm müşterilerin listesini görüntüleme.

Employee Screen:

- İşyeri Bilgisi: İşyeri bilgilerini görüntüleme.
- İlan Ekle: Sisteme yeni bir emlak ilani ekleme.
- Müşteri Ekle: Sisteme yeni bir müşteri ekleme.
- İlan Ara: Sistemde kayıtlı ilanları arama.
- İlan Listesi: Sistemdeki tüm ilanların listesini görüntüleme.
- Müşteri Listesi: Sistemdeki tüm müşterilerin listesinigörüntüleme.

7. Work Breakdown Structre