ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4531 Ağ Tabanlı Teknolojiler ve Uygulamaları Proje Raporu

Özel Etkinlikler için Pazar Yeri Uygulaması

Hüseyin TINAZTEPE

21290360

https://github.com/hsyntinaztepe/mekanbudur-event-venue-finder

https://youtu.be/yOrqCT6bWN4

ÖZET

Bu proje, etkinlik düzenleyiciler (düğün, nikah, nişan, doğum günü vb.) ile hizmet sağlayıcılarını (mekan, pastane, fotoğrafçı vb.) buluşturan, mikroservis mimarisinde geliştirilmiş özel etkinlikler için pazar yeri platformudur. Amacı tek platform üzerinde hem kullanıcıların ihtiyaçlarını ilan açarak giderebilmesi hem de hizmet sağlayıcılarının bu ilanlara başvurabilmesi; aynı zamanda hizmet sağlayıcılarının kendi mekanlarının da tanıtımını yapabilmesini sağlamaktır. Kullanıcılar kategorize edilmiş etkinlik ilanları oluşturur, hizmet sağlayıcılar bu ilanlara teklif verir ve platform üzerinde anlaşma sağlanır. Proje, Docker kullanılarak ayağa kaldırılmıştır böylece farklı sistemlerde yaşanacak uyumsuzlukların önüne geçmek hedeflenmiştir. Projenin Backendinde mikroservis yapısı kullanılmıştır. Bunun kullanılma sebebi projenin mikroservise ihtiyaç duyması değil mikroservis kavramının ve servisler arası iletişimin pekiştirilmesi amacıyla uygulanmıştır. Ve bunların sonucunda hem kullanıcı hem de hizmet sağlayıcı tarafından kullanılabilecek ihtiyaçları karşılayan sosyal bir uygulama ortaya konmuştur.

1. GİRİŞ

Bu rapor, platformun tasarım, geliştirme ve uygulama süreçlerini kapsamlı bir şekilde ele almaktadır. Proje, üç ana servisten oluşmaktadır: Ana API servisi (main backend), Geo servisi (mikroservis) ve Web frontend (kullanıcı arayüzü). Her bir servisin sorumluluğu, teknolojik altyapısı ve diğer servislerle olan iletişim protokolleri detaylı olarak incelenecektir.

Raporun devamında, öncelikle projenin teknolojik altyapısı ve kullanılan teknolojiler detaylı olarak açıklanacaktır. Ardından, sistem mimarisi ve mikroservis yapısı, veritabanı tasarımı ve ilişkileri, API endpoint'leri ve iş mantığı katmanları incelenecektir. Uygulama aşamasında karşılaşılan zorluklar, alınan kararlar ve çözüm yaklaşımları tartışılacaktır. Son olarak, projenin mevcut durumu değerlendirilecek ve gelecekte yapılabilecek iyileştirmeler önerilecektir.

Bu çalışmada, mikroservis mimarisinin tüm karmaşıklığı değil, temel kavramlar ve servisler arası iletişim mekanizmaları üzerine odaklanılmıştır. Ödeme entegrasyonu, gelişmiş raporlama ve analitik özellikleri de projenin mevcut versiyonunda yer almamaktadır; bunlar gelecek geliştirmeler için potansiyel alanlar olarak değerlendirilmektedir.

2. SİSTEM MİMARİSİ VE MİKROSERVİS YAPISI

Proje, monolitik bir yapı yerine sorumlulukların ayrıldığı mikroservis benzeri modüler bir mimariyi benimsemiştir. İletişim, senkron HTTP çağrıları üzerinden yürütülmektedir.

2.1. Main API Service (Ana Backend):

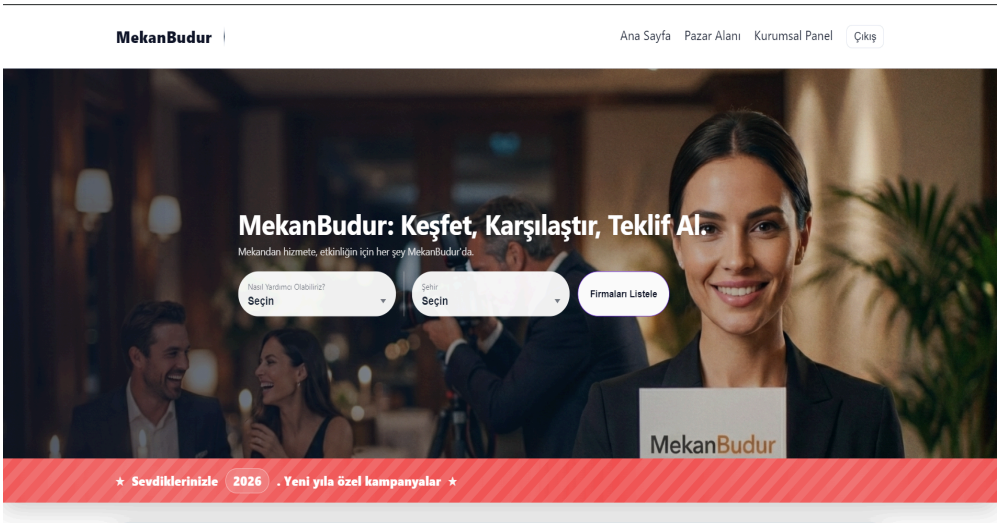
Sistemin kalbidir. Kullanıcı kimlik doğrulama, ilan yönetimi, teklif verme süreçleri ve veritabanı işlemlerini yönetir.

2.2. Geo Service (Konum Servisi):

Bu servis, ana servisten bağımsız çalışarak konumsal hesaplamaları ve dış kaynaklı harita verilerini (Google Places API) yönetir.

2.3. Client (Web UI):

Kullanıcıların sistemle etkileşime girdiği arayüz katmanıdır. Backend ile RESTful API prensiplerine sadık kalarak `fetch` API aracılığıyla haberleşir.



Şekil 2.1 WebUI homepage görüntüsü

3. VERİTABANI TASARIMI VE VARLIK İLİŞKİLERİ

Veritabanı şeması, pazar yeri mantığını destekleyecek şekilde ilişkisel (Relational) olarak tasarlanmıştır. `AppDbContext` yapısı incelendiğinde temel varlıklar şunlardır:

3.1. Users & VendorProfiles:

Sistemde temel `User` tablosu kimlik bilgilerini tutarken, `VendorProfile` tablosu hizmet sağlayıcılara özgü (Firma adı, hizmet kategorileri, kapasite vb.) detayları tutar. Bu, bire-bir ilişki mantığıyla kurgulanmıştır.

3.2. EventListings & Items:

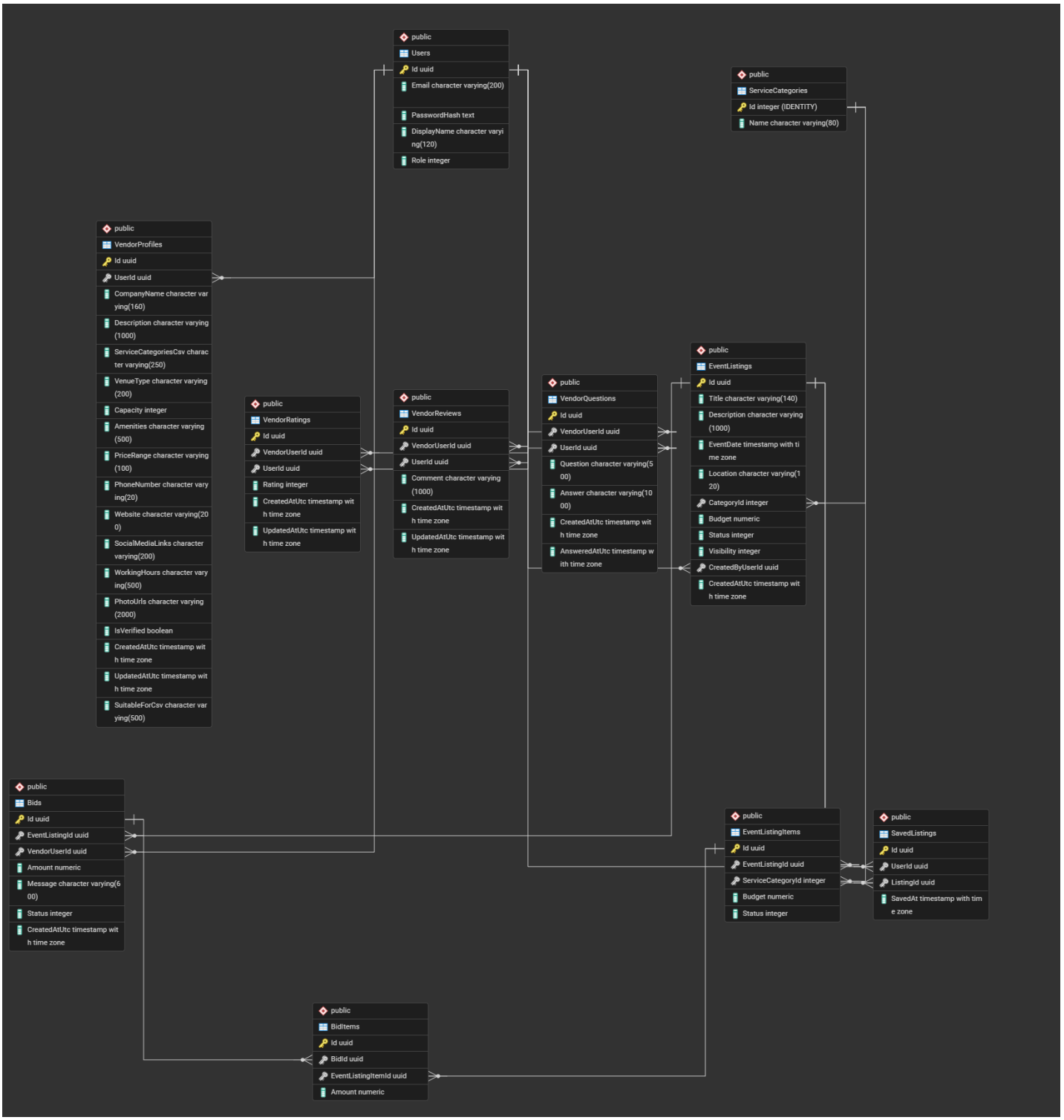
Bir etkinlik ilanı (`EventListing`), birden fazla ihtiyaç kaleminden (`EventListingItem`) oluşur. Örneğin, bir düğün ilanı altında hem "Fotoğrafçı" hem de "Pastane" kalemleri ayrı satırlar olarak tutulur.

3.3. Bids (Teklifler):

Tedarikçiler (`Vendor`), ilanlara (`EventListing`) teklif verirken, teklif detayları `BidItem` tablosunda saklanır. Bu yapı, tedarikçinin ilandaki sadece belirli kalemlere (örn: sadece fotoğrafçılık) fiyat verebilmesine olanak tanır.

3.4. Reviews & Ratings:

Güven mekanizmasını oluşturmak adına `VendorReviews` ve `VendorRatings` tabloları ile kullanıcıların hizmet sağlayıcıları puanlaması sağlanmıştır.



Şekil 3.1 Main Backend'e ait ilişkisel veri tabanı görüntüsü

4. API ENDPOINT'LERİ VE İŞ MANTIĞI

4.1. Kimlik Doğrulama ve Rol Yönetimi (Auth & Security)

Sistemin giriş kapısı olan bu katman, kullanıcıların sisteme güvenli bir şekilde dahil olmasını ve sadece yetkili oldukları işlemleri yapabilmesini sağlar.

4.1.1. Kayıt ve Giriş (Register & Login):

Kullanıcı veya Hizmet Sağlayıcı, `/api/auth/register` endpoint'ine e-posta, şifre ve rol bilgisi gönderir.

Backend, şifreyi veritabanına kaydetmeden önce `PasswordHasher` kullanarak güvenli bir şekilde hash'ler (şifreler).

`/api/auth/login` endpoint'i, gönderilen şifrenin hash'ini veritabanındaki ile doğrular ve başarılıysa kullanıcıya bir JWT (JSON Web Token) döner.

4.1.2. Token ve Rol Taşıma:

Üretilen JWT içerisinde kullanıcının ID'si (`NameIdentifier`), e-postası ve en önemlisi Rolü (User, Vendor veya Admin) şifrelenmiş olarak bulunur.

Frontend, bu token'ı alır ve saklar. `setAuthUI()` fonksiyonu, token içindeki veya login yanıtındaki rol bilgisini okuyarak arayüzü şekillendirir.

Örneğin; rolü "Vendor" olan bir kullanıcı giriş yaptığında, sadece hizmet sağlayıcılara özel olan "Teklif Ver" butonlarını görürken, "User" olanlar "İlan Oluştur" butonlarını görür.

4.2. İlan ve Teklif Döngüsü

Bu süreç, projenin temel işlevidir. Kullanıcı bir talep oluşturur (ilan), tedarikçiler bu talebe fiyat verir (teklif) ve kullanıcı en uygununu seçer.

4.2.1. İlan Oluşturma (User)

İşlem: Kullanıcı; düğün, nişan gibi bir etkinlik için tarih, konum ve bütçe kalemlerini belirler.

Teknik Detay:

Frontend tarafında kullanıcı, harita üzerinden (`initListingCreateMap`) konum seçer ve ihtiyaç kalemlerini (örn: Fotoğrafçı - 5000 TL, Pastane - 2000 TL) ekler.

Bu veriler `POST /api/listings` endpoint'ine gönderilir. Backend, bu isteği `EventListing` (İlan Başlığı) ve `EventListingItems` (Alt Kalemler) olarak ilişkisel veritabanına kaydeder.

Bu aşamada ilan durumu `ListingStatus.Open` (Açık) olarak işaretlenir.

4.2.2. İlanları Listeleme ve Görüntüleme (Vendor)

İşlem: Hizmet sağlayıcılar, potansiyel iş fırsatlarını görmek için sisteme girer.

Teknik Detay:

Vendor, `GET /api/listings` endpoint'i üzerinden sadece durumu "Aktif" ve "Açık" olan ilanları çeker.

Backend, veritabanı sorgusunda `Include(l => l.Items)` kullanarak ilanın alt kalemlerini (hangi hizmete ne kadar bütçe ayrıldığını) de listeye dahil eder.

4.2.3. Teklif Verme

İşlem: Vendor, ilanın tamamına veya sadece kendi hizmet verdiği kalemlere fiyat teklifi verir.

Teknik Detay:

Vendor, frontend üzerindeki "Teklif Ver" butonuna bastığında `openPazarBidModal` fonksiyonu çalışır ve bir form açılır.

Vendor, ilandaki kalemlerden hangilerini karşılayabileceğini seçer ve fiyat girer.

`POST /api/bids` endpoint'i çağrılır. Backend, `Authorize(Roles = "Vendor")` etiketiyle sadece vendorların teklif verebildiğini doğrular.

Ayrıca sistem, bir kullanıcının kendi açtığı ilana teklif vermesini engelleyen bir kontrol mekanizmasına sahiptir (`listing.CreatedByUserId == myId` kontrolü).

4.2.4. Teklif Onaylama ve Anlaşma (User)

İşlem: İlan sahibi gelen teklifleri inceler ve uygun gördüğünü onaylar.

Teknik Detay:

Kullanıcı, `GET /api/listings/{id}/bids` ile ilanına gelen teklifleri listeler.

Kullanıcı bir teklifi kabul ettiğinde POST `/api/bids/{id}/accept` endpoint'i tetiklenir.

Bu işlem veritabanında iki kritik güncelleme yapar:

Teklifin durumu `BidStatus.Accepted` (Kabul Edildi) olarak güncellenir.

İlgili ilan kalemlerinin durumu `ListingStatus.Awarded` (Verildi/Kazanıldı) olarak değiştirilir.

Eğer ilandaki tüm kalemler bir teklifle eşleşirse, ilanın genel durumu da `Awarded` olarak güncellenerek ilan kapanır.

4.3. Yönetim ve Denetim (Admin Kontrolü)

Platformun kalitesini korumak için yönetici yetkilerine sahip kullanıcılar için özel bir arka kapı (back-office) mantığı kurulmuştur.

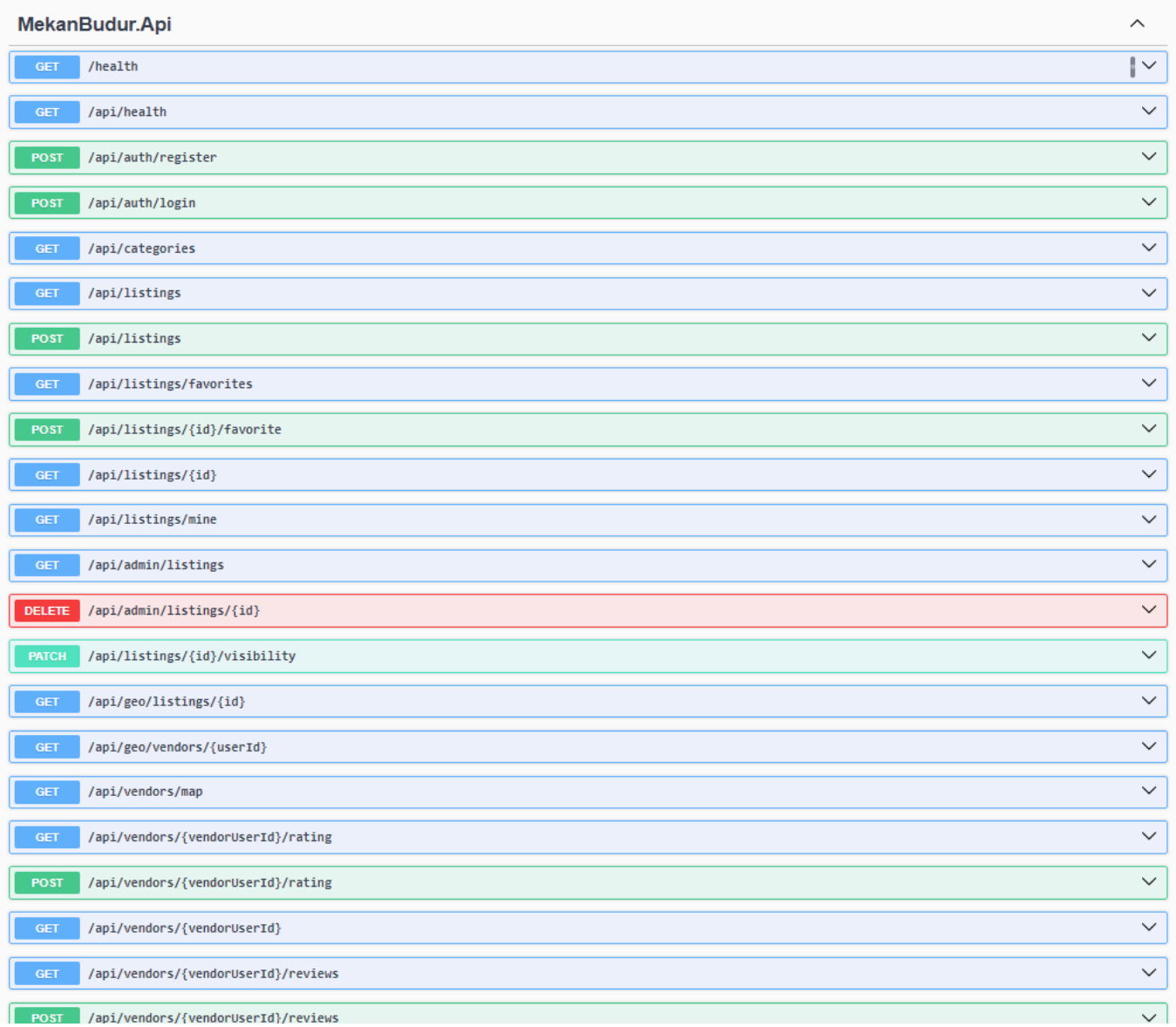
4.3.1. Genişletilmiş Görünüm:

Normal kullanıcılar sadece "Aktif" ilanları görebilirken, Adminler `/api/admin/listings` endpoint'i ile gizlenmiş veya silinmiş tüm ilanlara erişebilir.

4.3.2. Müdahale Yetkisi:

Admin, sahte veya uygunsuz bir ilan tespit ettiğinde `DELETE /api/admin/listings/{id}` ile ilanı silebilir. Bu işlem, "Cascade Delete" mantığıyla ilana bağlı tüm teklifleri ve alt kalemleri de veritabanından temizler.

Aynı şekilde, `/api/admin/vendors` üzerinden onay bekleyen mekanları görüntüleyebilir ve `/api/admin/vendors/{userId}` ile kural ihlali yapan bir işletmenin hesabını, profiliyle birlikte sistemden kaldırabilir.



Şekil 4.1 Swagger üzerinde API'ların gösterimi

5. UYGULAMA AŞAMASI VE KARŞILAŞILAN ZORLUKLAR

Geliştirme sürecinde karşılaşılan temel zorluklar ve bunlara üretilen çözümler şunlardır:

5.1. Harita Verilerinin Yönetimi:

Google Places API gibi dış servislerin maliyetli olması ve kota sınırları zorluk yaratmıştır.

Çözüm: `app.js` içerisinde görüldüğü üzere, Gölbaşı gibi belirli bölgeler için sabit (hardcoded) mekan verileri ve görselleri yerel olarak tanımlanmış, hibrit bir yapı kurularak API bağımlılığı azaltılmıştır. Ayrıca GeoService bir proxy (vekil sunucu) görevi görerek frontend'in doğrudan API anahtarlarına erişmesi engellenmiştir.

5.2. Frontend State Yönetimi:

React gibi bir kütüphane kullanılmadığı için sayfa yenilemeleri ve veri tutarlılığı zorluğu yaşanmıştır.

Çözüm: `app.js` içerisinde global değişkenler (`window.__pazarListings` vb.) ve olay dinleyicileri (Event Listeners) ile manuel bir state yönetim yapısı kurulmuş; `localStorage` kullanılarak oturumun sürekliliği sağlanmıştır.

5.3. Mikroservis İletişimi:

Servislerin birbirini bulması ve hataya dayanıklılık.

Çözüm: Docker Compose ağ yapısı ile servislerin isimleri üzerinden haberleşmesi sağlanmış ve basit HTTP Client yapısı ile iletişim standardize edilmiştir.

6. SONUÇ

Bu proje ile etkinlik sahipleri ve hizmet sağlayıcılarını bir araya getiren, konum tabanlı ve interaktif bir pazar yeri platformu başarıyla belli bir aşamaya getirilmiştir. Mikroservis mimarisinin temel prensipleri (servis ayrımı, konteynerizasyon) uygulanarak modüler bir yapı elde edilmiştir.

Projenin gelecekteki sürümleri için şu geliştirmeler önerilmektedir:

6.1. Ödeme Sistemi:

Teklif kabul edildikten sonra ödemenin platform üzerinden alınması (Iyzico, Stripe vb.).

6.2. Gerçek Zamanlı Bildirimler:

WebSocket kullanılarak, yeni bir teklif geldiğinde kullanıcıya anlık bildirim gönderilmesi.

6.3. Gelişmiş Geo-Sorgular:

PostGIS eklentisi kullanılarak "bana en yakın 5km içindeki fotoğrafçılar" gibi daha karmaşık uzamsal sorguların veritabanı seviyesine indirilmesi.

6.4. Harita İçeriğinin Arttırılması:

Aktif olarak tek bölgede veri bulunan bu sistemi her ilde aktif hale getirilecek.