Backend Case: Redis Tabanlı Company Detail Cache & Servis Entegrasyonu

Amaç

Şirket detay bilgilerini Redis üzerinde önbelleğe alarak, veritabanı veya başka bir mikrositeye yalnızca cache-miss olduğunda başvuran, hızlı ve ölçeklenebilir bir CompanyService inşa etmek.
GET çağrılarında Redis birincil kaynak gibi davranır; yazma-güncelleme operasyonlarında ilgili anahtar invalidasyonu ile tutarlılık sağlanır.
Ayrıca sistem; canlı veri akışı için WebSocket üzerinden bildirimler sağlar ve harici bir açık API ile entegre çalışır.

İş Gereksinimleri

1. Teknoloji Seçimi

• .NET Runtime / SDK: **.NET 8**
• Proje docker-compose ile ayağa kalkmalı (**API + Redis + SQL**)

2. Veri Modeli

• Örnek alanlar: id, name, sector, city, employeeCount, lastUpdated, ...
• Basit bir SQL (MSSQL) şeması ve örnek veriler içermeli

3. API Uçları

• GET /companies/{id} → Detay döner
• POST /companies → Yeni şirket ekler
• PUT /companies/{id} → Günceller, cache invalidasyonu yapar
• DELETE /companies/{id} → Siler, cache temizler

4. Cache Katmanı

• Redis key kalıbı: company:{id}
• TTL: Varsayılan **30 dakika** (konfigürasyonla değiştirilebilir)
• Connection pooling, retry, timeout gibi Redis bağlantı ayarları yapılandırılmalı

5. Cache Stratejisi

• **Read-Through + TTL:** GET çağrısında önce Redis kontrol edilir → yoksa SQL sorgulanır → sonuç Redis'e yazılır → cevap dönülür
• **Write/Update:** DB güncellenir → Redis'teki ilgili anahtar DEL ile silinir (veya overwrite yapılır)

6. Gerçek Zamanlı Bildirimler (WebSocket)

• Şirket güncelleme/silme olaylarında, WebSocket üzerinden istemcilere bildirim gönderilmelidir
• Örnek WebSocket endpoint: ws://localhost:5000/ws/companies/updates
• Güncellenen şirket bilgisi bu kanal üzerinden abonelere iletilmeli

7. Harici API Entegrasyonu

• Uygulama, internetten alınan ücretsiz bir public API ile entegre olmalıdır
• Örneğin: https://randomuser.me/api/ üzerinden örnek şirket benzeri veri çekilebilir
• En az bir uç nokta bu API'yi kullanarak veri döndürmelidir (örn. GET /external/companies/random)

8. Güvenlik ve Gözlemlenebilirlik

• Basit JWT veya API-Key doğrulama uygulanmalı
• Her istekte "cache-hit / cache-miss" bilgisi loglanmalı
• Sağlık kontrolü (/health) veya Prometheus endpoint'i ile metrikler sağlanmalı

9. Teslimat Paketi

• Çalışır kaynak kodu
• docker-compose.yml
• Açıklayıcı README.md:

- Yapılandırma açıklamaları

Kategori	Açıklama	Ağırlık
Fonksiyonellik	Doğru cache akışı, TTL, invalidasyon, WebSocket ve harici API çalışmalı	**30%**
Kod Kalitesi	Katmanlı mimari, SOLID, tip güvenliği, temiz logging	20%
Performans	Cache-hit'te milisaniyelik cevap, eşzamanlılıkta tutarlılık	25%
EKSTRA | Dokümantasyon	Net README	5%
WebSocket & API	WebSocket yayını ve harici API'nin doğru şekilde çalışması	**25%**