



Olaylar Gerçek Bir Hikayeden Alınmıştır

.Net Applications With ModernTools

BORA KAŞMER



borakasmer.com



@coderbora

- 25 YILDIR KOD YAZIYOR.

Ben Kimim

-  VBT'DE HEAD OF SOFTWARE ARCHITECT POZİSYONUNDA ÇALIŞIYOR

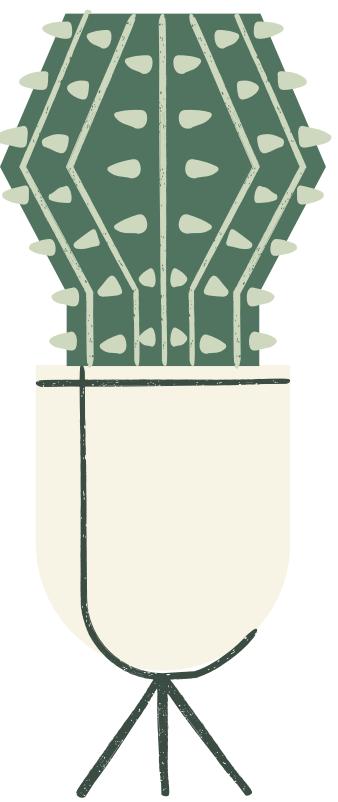
- MICROSOFT MVP (DEVELOPER TECHNOLOGIES)

- 2 KIZ BABASI, OYUNCU VE MOTORCU



github.com/borakasmer

Senaryo



Firmanın özel isteği üzerine, yabancı para birimi ile bir ürün satın alınması yapıldığında, HER ALIM için ilgili kur doviz.com'dan pars edilerek alınacak ve ₺ karşılığına çevrilecek. Kur bilgisi alındığı zaman, ilgili client bilgilendirilecek. Kaydetme işlemi sırasında, önemli alanlar, encrypted olarak saklanacak. Güncelleme anında Audit Log Tutulacak. Kur bilgisi belirlenmiş kayıtlar, DB yerine belli bir süre memory bir cacheden getirilecek.

.Net Core 5.0 Back-End

Angular 11 Front-End

Go WebParser
Microservice

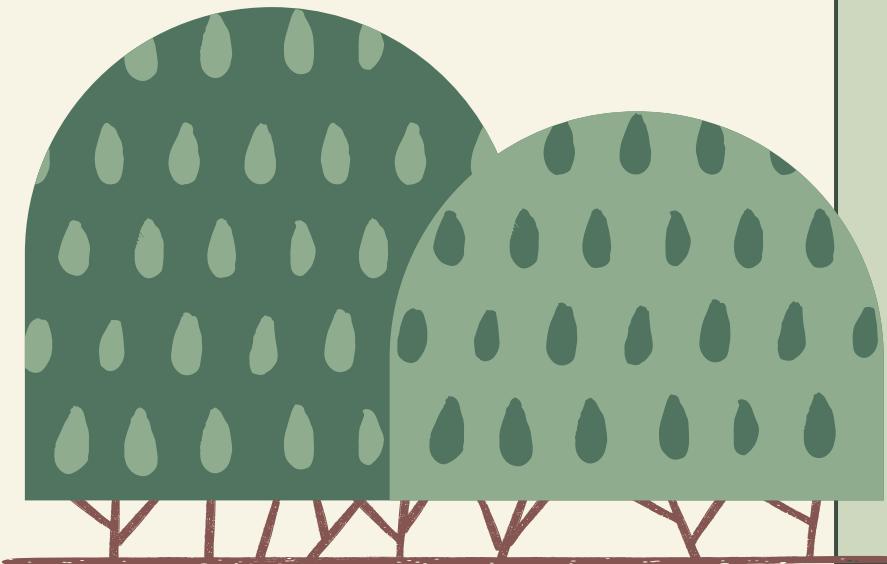
RabbitMQ Queue

ElasticSearch Audit Log

SignalR Socket

Redis Distributed Cache

Her Alım İçin Parse İşlemi



Kaydetme işlemi en hızlı şekilde yapılmalı ve parse işlemi yapılırken, kullanıcı sayfa üzerinde diğer işelevlere devam edebilmelidir. Kur bilgisi çekilip kaydedildikten sonra, performans amaçlı belli bir süre ilgili ürün için DB'ye gidilmemelidir.

Parse işlemi asenkron yürütülecektir. Her kaydetme işlemi kur değeri bulunmadan yapılacak ve ilgili ürün, RabbitMQ'ya atılacaktır. Go Microservice, Queue'dan aldığı ürünün kur bilgisini, döviz.com'dan parse edip, .NET Core servise Post edecktir. Kur bilgisi bulunan kayıt, Redis'e de atılacaktır.



DB, Core, Controller, Service, Repository

Öncelikle proje katmanlarına ayrılır.

DB First ile DAL katmanı oluşturulur. Endpointler Controller katmanında istekleri karşılar. Service katmanında tüm bussines işlemler yapılır. DB ile alakalı tüm işler Repository katmanında halledilir. Ayrıca Entity işaretlenmiş ise, Loglama ve DataEncryption Global olarak bu katmanda Service Injection ile yapılır. Ortak kullanılan araç ve modeller Core katmanında saklanır.

Giriş - Gelişme - Sonuç

Kod okunaklığının artması, Test ve Debug İşlemlerinin hızlanması, birden fazla kişinin aynı projede çalışması için, Katmanlı Mimari büyük kolaylıktır.

Katman 1: Güvenlik amaçlı, mobile ve web servisleri burada karşılanır. SignalR tetiklenir. RabbitMQ'ya işlenmesi amacı ile, kayıt atılır. Yani bir çeşit Adaptordür.

Controller

```
[HttpPost("InsertProduct")]
public IEnumerable<ViewProduct> InsertProduct([FromBody] ViewProduct model)
{
    if (model != null)
    {
        if (model.ID > 0)
            return _productService.UpdateProduct(model);
        else
            return _productService.InsertProduct(model);
    }
    return new List<ViewProduct>();
}
```

Katman 2: Tüm sorguların ve hesaplamaların yapıldığı yerdir. İlgili kaydın, Redisde olup olmadığına burada bakılır.

Service

```
public List<ViewProduct> InsertProduct(ViewProduct product)
{
    var data = _mapper.Map<Product>(product);
    _productRepository.Insert(data, true);

    //Insert Redis
    var cacheKey = string.Format(ProductDetail, data.Id);
    var model = _mapper.Map<ViewProduct>(data);
    model.SeriNo = product.SeriNo; //SeriNo Decrypt olarak atanır.
    _redisCacheManager.Set(cacheKey, model);
}
```

Katman 3: DB'ye yazma ve okuma işlemleri, Encryption ve Elastic Search'e Loglama, global olarak burada yapılır.

Repository

```
public void Insert(T entity, bool isEncrypt = false)
{
    if (entity == null)
        throw new ArgumentNullException(nameof(entity));

    if (isEncrypt)
    {
        entity = EncryptEntityFields(entity, _context);
    }
    entity.UsedTime = DateTime.Now;
    //-----

    _entities.Add(entity);
    _context.SaveChanges();
}
```

Mikroservis İşlemler

HER KAYIT İŞLEMİNDE "DOVİZ.COM"'UN PARSE İŞLEMİ UZUN SÜRECEĞİNDEN, GO MİCROSERVİS YARDIMI İLE, ARKADA PARS İŞLEMİ ASENKRON OLARAK YAPILIR VE SONUÇ .NET CORE SERVİSİNE POST EDİLİR.

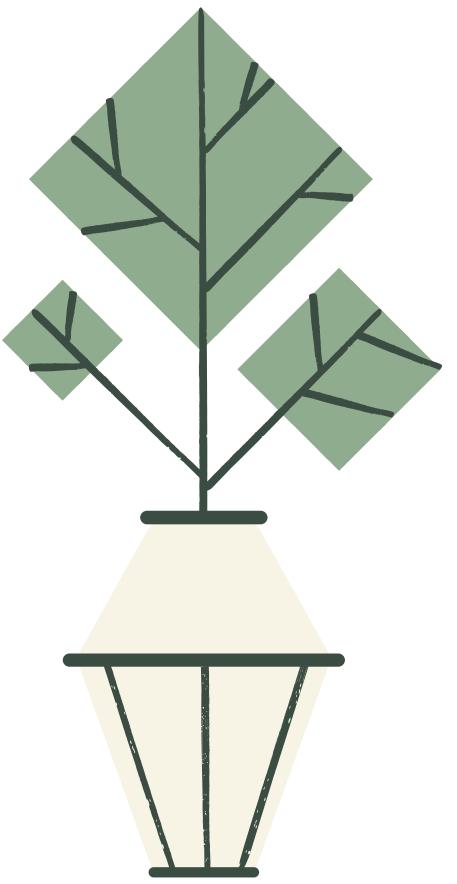
Insert olan product kaydı, Db'ye ve Redis'e kaydedildikten sonra, vakit alan pars işleminin asenkron yapılması için, RabbitMQ'ya atılır. Entity'e ait kritik kolonlar, şifrelenerek atılır.

Go servisinde "Product" channel'ından alınan kayıt, "döviz.com" pars edilerek güncellenir. Ve .Net Core servisine ürünün son güncel ₺ fiyatı hali, Post edilir.

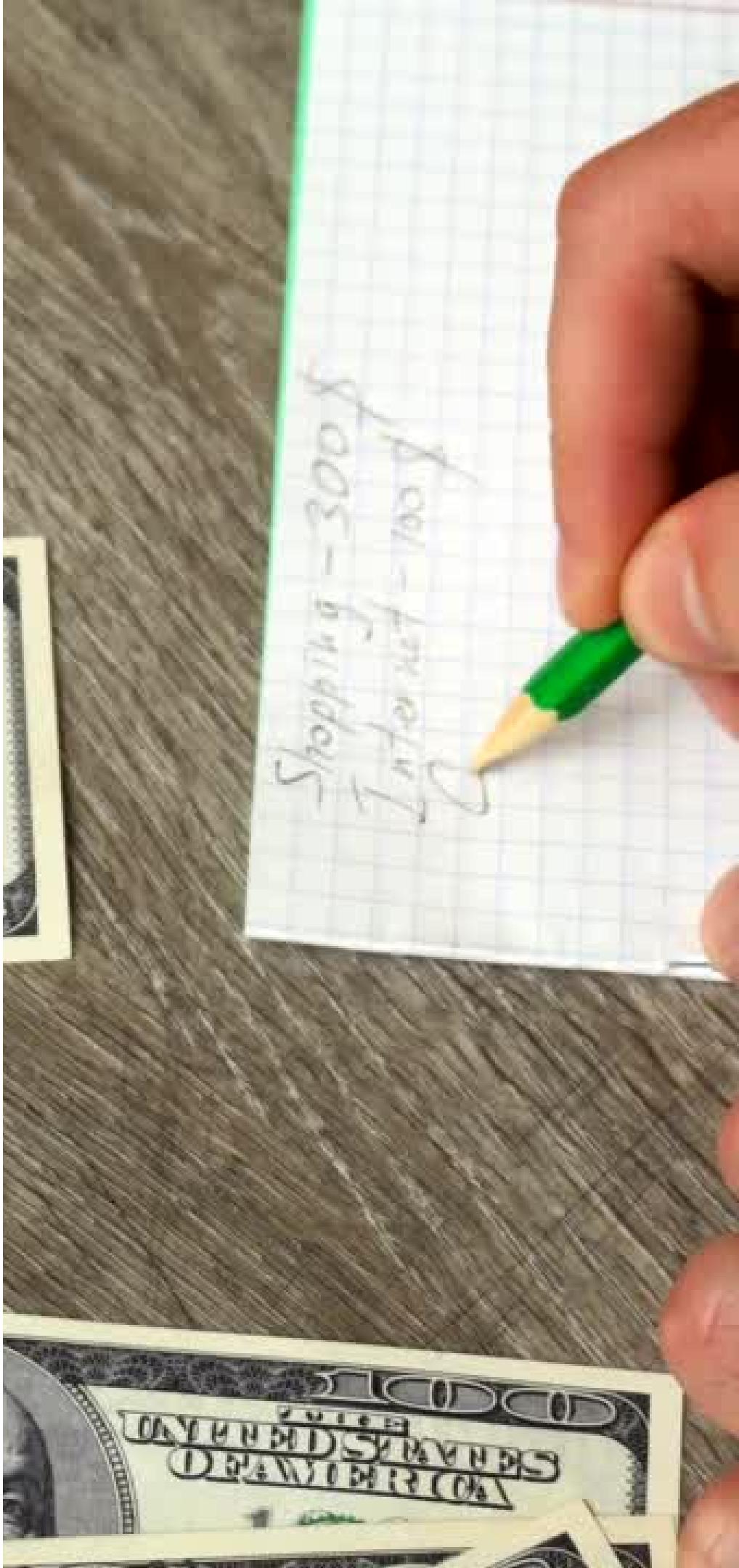


Ürünün Güncellenmesi

Go Microservice'den gelen güncel ₺ fiyatlı ürün bilgisi, Controller katmanından servis katmanına gönderilirken, aynı zamanda tüm clientlara signalR kullanılarak Real Time post edilir.



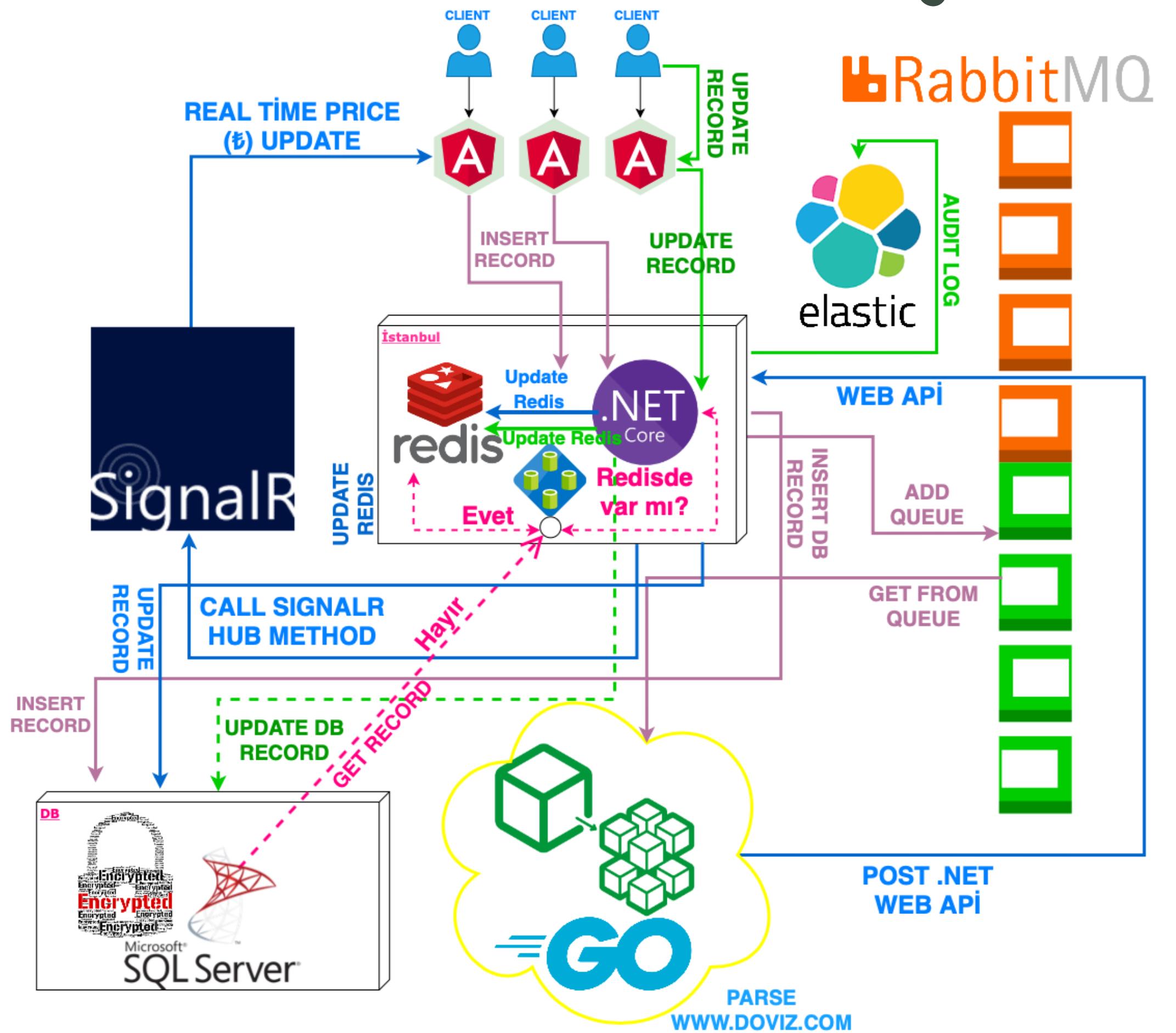
- Güncellenen ürün, signalR ile tüm clientlara Real Time gönderilir.
- Servis katmanında ürünün son toplam ₺ fiyatı hesaplanıp, Repository katmanında DB'ye kaydedilir. Ayrıca hasas datalar, değişmiş ise şifrelenerek atılır.
- Servis katmanında, DB'de güncelenen ürün bilgisi, Redis'de de güncellenir.
- Repository katmanında, güncelene ürünün Audit Log kaydı, ElasticSearch'e indexlenir.



Globalization

Repository katmanında, güncellenen Entityler içinden sadece, "IAuditable" interface'inden türeyenler, Elastic Search'e indexlenir. Ve Entity üzerinde sadece "CryptoData" ile işaretli kolonlar, şifrelenerek DB'ye atılır.

Temel Mimari Şema



kibana



Kullanılan Teknolojiler

Bu seminerde ihtiyaçlara yönelik kullanılan tüm teknolojiler aşağıdadır.



Akla Gelebilecek Sorular

Yavaş yavaş ilerlemek, sabit durmaktan iidir.

BU KADAR TEKNOLOJİ GEREKLİ Mİ?

Her işi ustasına bırakmak lazım.Belki hepsi, bir anda öğrenilemiyebilinir. Ama zamanla, ihtiyaç duyuldukça, kütüphaneye katılmasında hiçbir zarar yoktur.

KAYIT SONRADAN GÜNCELLENİR İSE NE OLUR?

Servis katmanında, Redis güncellenir. Repository katmanında DB güncellenir ve ElasticSearch'e Audit Index atılır. Socket ile diğer clientlara push notify yapılmaz. O işlem sadece Kur bilgisi parse edilince, anlık yapılır.

NEDEN MİCROSERVİS OLARAK GO KULLANILDI?

Projenin başta çok hızlı ayağa kalkması, zamanla harcadığı bellek miktarının çok az olması, bir sitenin Pars edilebilmesi için çok pratik bir kütüphanesinin olması ve stabilizasyonu için tercih edilmiştir.

