Seçilen CVE Veritabanı

CVE veritabanı olarak https://nvd.nist.gov/ adresinden hizmet veren “National Vulnerability Database” seçilmiştir. Bu seçimdeki en büyük etken diğer pek çok veritabanı verileri json feedler hâlinde verirken NVD’nin oldukça zengin bir sorgulama API’ı ve bu API’ı açıklayan iyi bir dökümantasyonu olmasıdır. Yaptığım kıyaslamalara göre diğer pek çok sistem CVE verileri deklare edildikten sonra üzerinde çalışma yapmadan kaydı açan kişi/kurumun sunduğu verileri olduğu gibi sunuyor ve bu da pek çok CVE kaydının CPE eşleşmesi olmaması, CPE URIlerinin hatalı/eksik girilmesi gibi sorunlar doğuruyor. NVD ise bünyesinde çalıştırdığı analistler sayesinde kopuk deklare edilen CVE kayıtlarındaki CPE URIlerinin hatalarını gidermek, CPE eşleşmelerini sağlamak ve bu projede faydalanılmasa da etki skorlandırması yapmak gibi görevler üstleniyor. 2000 yılından beri Amerika Birleşik Devletleri bünyesinde geliştirilen bir sistem olduğu için de güvenilirlik ve süreklilik konusunda bir sorun yaşanma ihtimalinin minimum olacağını düşünerek veri kaynağı olarak NVD’yi seçtim.

Kullanılan Ürünler

Genel Kriterler:

Aşağıdaki kriterler veritabanı sağlayıcısı hariç tüm ürün seçimlerinde ilk kriterler olarak gözetilmiştir. Bu kriterleri sağlamayan hiçbir ürün dikkate alınmamıştır.

1) Açık kaynak kodlu olması.
2) Ürün lisansının doğrudan ilgili ürünü servis olarak sunmak dışında ticari kullanıma izin vermesi.

Framework Seçimi:

Uygulamanın ana framework’ü olarak LTS (Life Time Support) sürümü olmamasına rağmen .NET 5.0 kullanılmıştır. Bunun en büyük sebebi Microsoft ve bağımsız pek çok araştırmacı tarafından yapılan benchmark testlerinde .NET 5.0’in en güncel .NET LTS sürümü olan Core 3.1’den pek çok alanda açık ara farkla performanslı olmasıdır. .NET 6.0 projenin mimarisine karar verildiği zamanda henüz Release Candidate aşamasına dahi gelemediği için değerlendirmeye alınmamıştır.

Veritabanı Seçimi:

Uygulamadaki anlık transaction ve toplam kayıt sayısının RDBMS çözümlerinin karşılayamayacağı boyutlarda olması beklenmediği için RDBMS bir veritabanı seçilmiştir. Veritabanı sağlayıcısı olarak .NET ekosistemi ve seçilen ORM çözümü Entity Framework ile out-of-the-box olarak en az özel müdahaleye gerek duyarak çalışan SQL Server seçilmiştir. Mimariye karar verildiği esnadaki en güncel stabil sürüm olan SQL Server 2019 kullanılmıştır.

ORM Seçimi:

Uygulamayı seçilen veritabanından ve veri depolama işlemlerinden tamamen bağımsız hâle getirebilmek adına Dapper gibi sadece object-relation ilişkisi kuran bir Micro-ORM değil LINQ2SQL gibi çözümlerle veritabanını tamamen soyutlayan bir ORM kullanılma kararı alınmıştır. Bu tipteki ORMler arasından da performans, dökümantasyon desteği, kullanım sıklığı ve out-of-the-box kullanım kolaylığı metrikleri gözetilerek Entity Framework seçilmiştir. Mimariye karar verildiği esnadaki en güncel stabil sürüm olan EF Core 5 kullanılmıştır.

Gateway Seçimi:

Hem iç mikroservislerin dış dünyaya kapalı hâle getirilmesi hem de BFF çözümü olarak hizmet verebilmesi için uygulamada bir Gateway kullanılmasına ve clientların uygulamaya bu Gateway üzerinden erişmesine karar verilmiştir. Gateway çözümleri arasından merkezi auth desteği, request aggregation ve koda dokunmadan configlerle routing/auth ayarlamaları yapılamayan çözümler değerlendirmeye alınmamıştır. Bu kriterleri sağlayan çözümler arasından dökümantasyon desteği, kullanım sıklığı ve out-of-the-box kullanım kolaylığı metrikleri gözetilerek Ocelot seçilmiştir. Mimariye karar verildiği esnadaki en güncel stabil sürüm olan Ocelot 17 kullanılmıştır.

Message Broker Seçimi:

Message Broker seçiminde ilk kriter olarak haberleşme protokolünde AMQP gibi standart bir protokol kullanılmasını değerlendirmeye aldım. Kullanılan Broker değiştirilmek istendiğinde standart dışı özel protokol kullanan bir ürünü değiştirmek için uygulamanın Broker ile olan haberleşme yöntemlerini de değiştirmek gerekeceği için Kafka gibi özel protokoller kullanan Brokerlar değerlendirmeye alınmamıştır. AMQP protokolüne destek veren Brokerlar arasından uygulamada beklenen mesaj yoğunluğu, performans, dökümantasyon desteği, kullanım sıklığı ve out-of-the-box kullanım kolaylığı metrikleri gözetilerek RabbitMQ seçilmiştir. Mimariye karar verildiği esnadaki en güncel stabil sürüm olan RabbitMQ 3 kullanılmıştır.

Loglama Seçimi:

Tüm mikroservis ve background workerların loglarının merkezi tek noktadan görüntülenebilmesi için merkezi bir loglama sunucusu kullanılmasına karar verilmiştir. Ürün değişiminin uygulamayı etkilememesi adına UDP protokolünü desteklemeyen ve .NET 5 içerisindeki standart ILogger interfacei dışında kod kullanımı gerektiren ürünler dikkate alınmamıştır. Bu kriterleri sağlayan merkezi loglama ürünleri arasından uygulamanın üreteceği log yoğunluğu, performans, dahili ya da üçüncü parti monitoring uygulamalarının kullanım kolaylığı, dökümantasyon desteği, kullanım sıklığı ve out-of-the-box kullanım kolaylığı metrikleri gözetilerek Graylog seçilmiştir. Mimariye karar verildiği esnadaki en güncel stabil sürüm olan Graylog 4.1 kullanılmıştır. Graylog'un bağımlılıkları olmasından dolayı projeye MongoDB 4.2 ve ElasticSearch 7.10.2 dahil edilmiştir.

LDAP İmplementasyonu Seçimi:

Dökümantasyon desteği, kullanım sıklığı ve kullanım kolaylığı metrikleri gözetilerek OpenLDAP isimli implementasyon seçilmiştir.

OpenLDAP'ın kendisi bir docker image sağlamadığı için üçüncü parti bir docker image seçilmiştir. Dökümantason desteği ve kullanım sıklığı metrikleri gözetilerek osixia/openldap (https://github.com/osixia/docker-openldap) projesi seçilmiştir.

Directory yönetim önyüzü olarak 3. parti bir çözüm kullanılmasına karar verilmiştir. Kullanım kolaylığı, dökümantasyon desteği, kullanım sıklığı ve kurulum kolaylığı metrikleri göze alındığında seçilen osixia docker image'ının da destek verdiği phpLDAPadmin ürünü kullanılmıştır.

Cache Seçimi:

Scaling kapsamında load balancer + N adet container yoluna gidilebileceği için cache çözümü olarak distributed cache kullanılmasına karar verilmiştir. Uygulamanın ihtiyacı olan veri tiplerine destek, performans, dökümantasyon desteği, kullanım sıklığı ve out-of-the-box kullanım kolaylığı metrikleri gözetilerek Redis seçilmiştir. Mimariye karar verildiği esnadaki en güncel stabil sürüm olan Redis 6.2 kullanılmıştır.

Cache yönetim önyüzü olarak 3. parti bir çözüm kullanılmasına karar verilmiştir. Kullanım kolaylığı, dökümantasyon desteği, kullanım sıklığı ve kurulum kolaylığı metirkleri göze alınarak Redis Commander ürünü seçilmiştir.

Birim Test Frameworkü Seçimi:

Visual Studio 2019 Test Explorer ekranına ve Fixture yapısına destek vermeyen frameworkler değerlendirmeye alınmamıştır. Kriterleri sağlayan frameworkler arasından dökümantasyon desteği, kullanım sıklığı ve kullanım kolaylığı metrikleri gözetilerek NUnit seçilmiştir. Mimariye karar verildiği esnadaki en güncel sürüm olan NUnit 3.13.1 kullanılmıştır.

.NET Standart Kütüphanesi Dışı Kullanılan 3. Parti Kütüphaneler

AsyncEnumerator:

.NET 5 içerisindeki Parallel.ForEach metodu lambda fonksiyonun async olmasına izin vermediği için bu desteği sağlayan AsyncEnumerator kütüphanesi kullanılmıştır.

AutoMapper:

Basit obje dönüşümleri için kod yazmamak adına kullanım sıklığı ve dökümantasyon desteği metrikleri gözetilerek AutoMapper kütüphanesi kullanılmıştır.

Gelf.Extensions.Logging:

Graylog bağlantısını kurmak adına dökümantasyon desteği ve kullanım sıklığı metrikleri gözetilerek GELF APIları içeren Gelf.Extensions.Logging kütüphanesi kullanılmıştır.

Newtonsoft.Json:

Standart JSON Serializera kıyasla daha performanslı yapısı ve konfigürasyon kolaylığı sayesinde NewtonSoft.Json kütüphanesi kullanılmıştır.

Novell.Directory.Ldap.NETStandard:

LDAP entegrasyonunu sağlamak adına kullanım sıklığı ve dökümantasyon desteği metrikleri gözetilerek Novell.Directory.Ldap.NETStandard kütüphanesi kullanılmıştır.

StackExchange.Redis:

Redis entegrasyonunu yönetmek adına performans, kullanım sıklığı, dökümantasyon desteği ve kullanım kolaylığı metrikleri gözetilerek StackExchange.Redis kütüphanesi kullanılmıştır.

Swashbuckle.AspNetCore

OpenAPI dökümantasyonlarının otomatik üretimi ve doğrudan önyüzden sunulmasının sağlanması adına bir ürün kullanımına karar verilmiştir. Özellik zenginliği, kullanım kolaylığı, dökümantasyon desteği ve kullanım sıklığı metrikleri gözetilerek Swashbuckle.AspNetCore kütüphanesi seçilmiştir.