**Micro Servis Mimarisi Proje**

**Tools**

* **RabbitMQ :**

( ilk olarak Gelişmiş Mesaj Kuyruklama Protokolünü uygulayan ve o zamandan beri Akışlı Metin Yönelimli Mesajlaşma Protokolü, MQ Telemetri Aktarımı ve diğer protokolleri desteklemek için bir eklenti mimarisiyle genişletilen açık kaynaklı bir mesaj aracı yazılımıdır. )

* **Azure Service Bus :**

(Kurumsal hizmet veri yolu, hizmet odaklı bir mimaride karşılıklı etkileşen yazılım uygulamaları arasında bir iletişim sistemi uygular. )

* **Redis:**

( Redis bir veri yapısı sunucusudur. Açık kaynak, bellek kullanımlı, anahtar-değer deposudur. Redis "Uzak Sözlük Sunucusu" anlamına gelmektedir. Çeşitli kaynaklara göre en çok kullanılan anahtar-değer veritabanıdır.)

* **Sql Server:**

(veri tabanı)

* **Blazor WASM:**

(Blazor; browser üzerinde WebAssembly teknolojisini kullanılarak, ASP.NET Core üzerinde C#, Razor ve HTML kodları ile client-side uygulamalar yapabileceğimiz yeni bir .NET web framework’üdür. Single page application (SPA) oluşturma sürecini basitleştirerek .NET üzerinde full-stack olarak geliştirme yapmamıza yardımcı olmaktadır)

* **Net5:**

(tüm önemli masaüstü geliştirme işlevleri ve kitaplıkları ile birlikte gelecektir. WPF, Windows Forms ve UWP (Evrensel Windows Platformu) 3 temel masaüstü platformudur. . Net 5 ayrıca Entity Framework, GDI +, LINQ ve ADO.Net gibi masaüstü geliştirmeler için bazı önemli alt framework'leride destekler.)

* **Ef5 :**

(NET Core platformu için geliştirilen ORM aracıdır.)

* **Graylog:**

(Graylog, sunucular, ağ cihazları, uygulamalar ve kendi yazdığınız kodlarla üreteceğiniz log (kayıt) dosyalarını toplamanıza ve işlemenize olanak sağlayan bir araçlar paketi.)

* **Docker:**

(Docker, "konteynerleştirme" olarak da bilinen işletim sistemi seviyesinde sanallaştırma sağlayan bir bilgisayar programıdır. )

**Patterns**

* **Domain Driven Design (DDD):**

( DDD domaini kendine ait ortak bir dil barındıran, sınırları belirli, bağımsız bileşenlere ayıran bir yaklaşımı belirler. Oluşan bu ortak dile **ubiquitous language,** bağımsız birimlere ise **bounded context** denir. )

* **CQRS:**

(CQRS, ana odağı write (yazma) ve read (okuma) sorumluluklarının ayrıştırılmasına dayanan bir mimari tasarım modelidir. CQRS mimarisi, CQS ilkesi baz alınarak kurulmuştur. CQS’in ana fikrinden bahsetmek gerekirse; bir metot objenin durumunu değiştirmelidir ya da geriye bir sonuç dönmelidir,)

* **Mediator:**

( nesneler arasındaki kaotik bağımlılıkları azaltmayı sağlayan davranışsal bir tasarım desenidir. Bu desen nesneler arasındaki doğrudan iletişimi sınırlar ve sadece mediator nesnesi üzerinden haberleşmeye zorlar.)

* **Http Aggregation:**

(Aggregation bir class’ın hayatanın diğer bir class’ın hayatını etkilememesi olayıdır.)

* **JWT(JSON Web Tokens) :**

(JSON Web Token, tarafların birbirleri arasındaki veri alışverişini ve bunun doğrulamasını sağlayan JSON tabanlı RFC 7519'de tanımlanmış açık bir standarttır. Örneğin bir sunucu, kullanıcının yönetici ayrıcalıklarına sahip olduğunu belirten bir anahtar oluşturabilir ve bunu kullanıcıya gönderebilir. )

* **Healthcheck:**

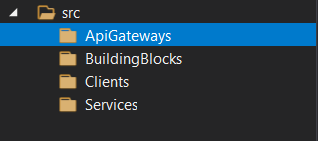
(Health Check servisleri; uygulamanın sağlık durumunun kendisi tarafından ölçülebilmesini ve bunu dışarıyla paylaşmasını sağlayan, uygulamanın iç metriklerini kontrol eden bir servistir.)

https://www.youtube.com/watch?v=ClHWjomsnUQ&list=PLRp4oRsit1bzd6v\_1zwNjdBOnGNuvHjWy&index=3

**Proje Oluşturma Aşamaları**

ilk olarak projeye oluşturuyoruz.

blank solution seçip projenin ismini oluşturuyoruz.

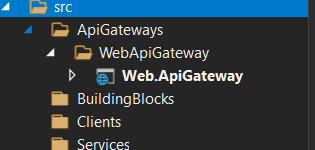
src dosyanı oluşturup altına bu klasörleri oluşturuyoruz

**ApiGateways:**  Eğer ki clientların servislerimize ellerini kollarını sallayarak girmelerini istemiyorsak, öncesinde bir entry point olmasını ve buradan geçmelerini istiyorsak API gateway kullanabiliriz. API gatewayin birçok işlevi olmasının yanı sıra temel işlevi clienttan requestleri alıp onları uygun servislere yönlendirmektir. Bir requestle arkada birden çok servis çağırıp sonuçları toplayıp dönebilir.

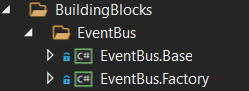
**BuildingBlocks:** İnşa sırasında ortak blokları koyacağımız yer

**Clients:**servisleri tüketecek olan clientler için

**Services:** Microservislerimiz için oluşturulan klasör

ApiGateways’in altına WebApiGateway isimli bir klasör oluşturuyoruz ve o klasörün içine bir web api projesi oluşturuyoruz oluştururken location yolunu elimizle yazıyoruz (örn: src/ApiGateways/webApiGateway)

Oluşan projede ki gerekli temizlikleri yapıyoruz.

daha sonra BuildingBlocks altına EventBus klasörü açıyoruz onun altına EventBus.Base adında bir class library oluşturuyoruz. Location biligisini girmeyi unutma(orn:src/dfdff/fgg)

EventBus.Base projesi bizim birden fazla servisbus’ımız olabilecek hem rabbitMq hemde azure servis kullanabileceğiz bu bizim ikisinin base classı olacak.

Yine aynı şekilde EventBus.Factory oluşturuyoruz.

EventBus.Factory dışardan servislerimizin yani microservislerimizin ortak kullanacağı sadece bir tane parametre ile direk rabbitmq ya da azure base bağlanacağı bir factory librarymiz olacak.