Order servisimizle async haberleşeceğimiz için RabbitMQ kullanacağız. RabbitMQ'yu direkt kullanmak yerine MassTransit ile birlikte kullanacağız.

**MassTransit, distribute sistemler için tasarlanmış bir framework'tür.**

Distribute sistemlerde mutlaka bir mesaj kuyruk sistemi olacağından veya distribute transaction olacağından dolayı bu işlemleri kolayca yönetebileceğimiz bir   
frameworktür.

**Hata yönetimi, yeniden deneme, bekleme ,transaction'u yönetme gibi işlemleri yapabilir.  
MassTransit bir çok mesaj kuyruk sistemiyle çalışır ; Amazon, azure, service bus, rabbitmq.**  
Entegrasyon yapıldıktan sonra retry mekanizmalarıyla uğraşmak yerine (rabbitmq ayakta olmadığı durumlar tekrar bağlanma vs senaryolar) masstransit bu işlemleri yönetiyor.  
**Transaction yönetimi için Saga Pattern uygulanması gibi bir çok işlemi yapmamızı sağlar.**

A screenshot of a computer screen

Description automatically generated

MassTransit, RabbitMQ ya bir mesaj gönderirken bu mesajlar Command ve Event tipleri olarak ikiye ayrılır.

Mass Transit;

RabbitMQ’ya gönderilecek mesaj Tek bir servis tarafından işlenecekse Command gönder,

RabbitMQ’ya gönderilecek mesaj birden fazla farklı servis işleyecekse Event gönder der.

Bizim örneğimizde(ödeme alındıktan sonra rabbitmq’ya sipariş detaylarını göndermek istiyoruz) mesajı sadece order servisi işleyecek.Bu sebepten Command göndermeliyiz.

Event kullanımına örnek :  
User Created isimli bir event oluştuğunda birden farklı servis işin içine girecekse (kullanıcıya email gönderebilir, kullanıcıya indirim kuponu tanımlar, profil fotoğrafını işleme her biri ayrı servis) event oluşturur. Event’ler geçmiş ifadeyle kullanılır.

Command’de bir servisin birden fazla instance’ı olabilir. Tek bir servis vardır. Order servisinden 5 tane container ayağa kaldırılabilir. Kuyrukta da 5 mesaj olduğunu düşünürsek hepsine birer birer dağıtır. İlgili mesaj sadece bir kere işlenmiş olur.

A screenshot of a computer

Description automatically generated

Mesaj iletim tipleri (Message Transport Types):

Send Command için, Publish ise event için kullanılır.

Send ifadesi genellikle ISend endpoint interface’i üzerinden Command’lerimizi gönderebiliriz.

IPublish endpointleri üzerinden eventlerimizi gönderebiliriz.

Publisher (Mesajı yayınlayan/gönderen) ve Receiver(Mesajı alacak olan) aynı namespace’de olmalıdır.

Bu durumda command ve eventleri SharedLibrary’de oluşturacağız ve Publisher ve Receiver SharedLibrary’i referans alacak.

A screenshot of a computer

Description automatically generated