Transaction-Service

Transaction servis, kullanıcının işlem için gönderdiği isteği internal db’ye kaydeder (bu işlem için H2 In Memory db seçilmiştir) ve işlem bilgilerinin işlenmesi için RabbitMQ’daki kuyruğa (transaction-queue) yazar. Bunun sebebi, gönderilen ister dokümanında işlemin 2 dakika sürebileceği belirtilmişti. 2 dakika sürecek olan işlem çok uzun zaman harcayan bir işlemdir ve bu işlemlerin asenkron ve hatta multi-thread işlenmesi gerekir. Bu sebepten transaction servisi için kurgulanan yapı, gelen isteğin anında kaydedilmesi ve müşteriye yanıt dönülmesi yönündedir. İşlemin internal db’ye kaydedilmesi ve kuyruğa atılması aynı servis içinde ve Transactional olarak geliştirildi. Herhangi bir sebepten kuyruğa yazma işlemi gerçekleşmez ise o zaman internal db’ye atılan kayıt dahi rollback olacaktır.

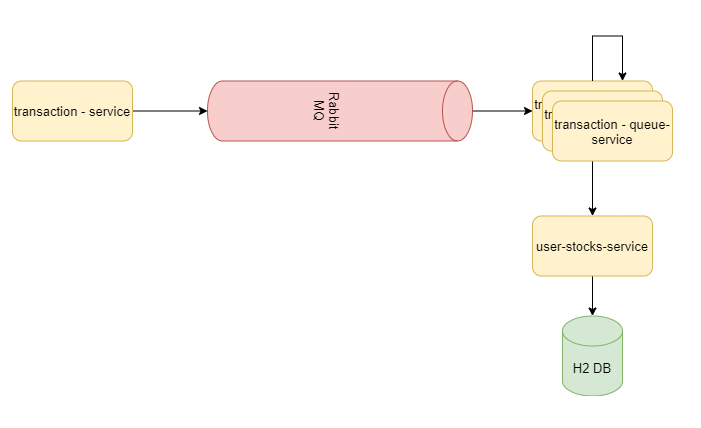
Kuyruğa atılan işlem kaydı, kuyruğu dinleyen transaction-queue-service modülünde dinlenip işlenecektir.

Transaction-Queue-Service

Bu servisin görevi, transaction-service ile kullanıcı için yapılan alım/satım işlemini external servisi çağırarak gerçekleştirmek ve sonra olarak kullanıcının hisse senedi bilgilerini güncellemektir. Eğer alım yapıldıysa kullanıcının hisse bilgileri artar, satım işlemi yapıldıysa kullanıcının hisse bilgilerini azaltır. Alım – Satım ile ilgili olan logic servis içinde implement edilmiştir.

Servisin işleyişi şu şekildedir: **transaction-service** tarafından RabbitMQ’ya bırakılan mesajı dinleyip external bir servisi call eder gibi Thread sleep edilir. Sleep işlemi sonrası kullanıcının hisse(stocks) bilgisini güncelleyen servisi FeignClient ile çağırır ve işlemini tamamlar. Aşağıdaki resimde bu sürecin akışı gösterilmiştir.

Transcation-Queue-Service bir cloud teknolojisi üzerinde çalıştırıldığında birden fazla instance ayağa kaldırılıp, kuyruktaki kayıtların multi-thread bir şekilde işlenmesi sağlanabilir ve müşteriye hızlı yanıt dönülebilmesi için bu işlem yapılmalıdır.



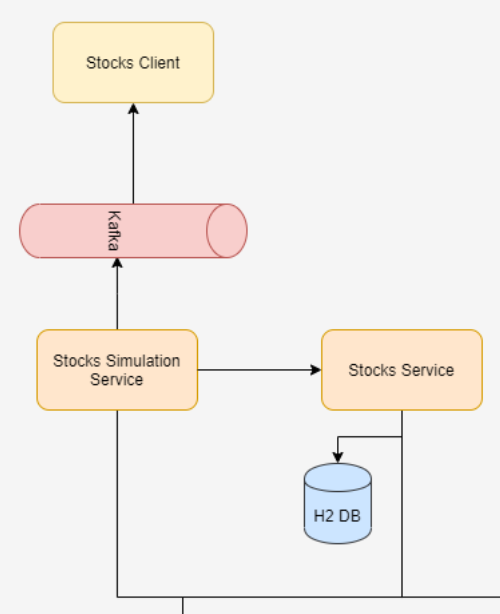
User-Stocks-Service

Kullanıcının satın aldığı veya sattığı hisse bilgilerini tutan servistir. Bu servis Transaction-Queue-Servis tarafından kullanıcının alım veya satım(buy/sell) işlemleri için kullanılır. Servis, H2 in-memory db’ye sahiptir ve kullanıcı + hisse bilgilerini bu db’de saklar. Saklanan bu verileri örnek için geliştirdiğim front-end uygulaması olan Stocks-Client servisi Gateway-Server üzerinden call ederek giriş yapan kullanıcın sahip olduğu hisse bilgilerini gösterir.

Stocks-Simulation-Service

User story’lerde belirtilen ve 3 borsadan hisse bilgilerinin çekilmesini simüle eden servistir. Bu servis içinde dummy olarak verilmiş 2 tane hisse senedi için Random olarak fiyat belirler ve yeni fiyatları Stocks-Service’i Feignt client ile call ederek günceller. Bunun sebebi hisselerin son fiyat bilgilerini kendi veritabanımda tutma isteğidir. Bu işlemin hemen ardından stream edilmesi için Kafka’ya publish eder. Burada Kafka kullanılmasının sebebi hisse bilgilerinin anlık olarak iletilmesi gerekliliğidir. Kafka Stream ile bu işlem olması gerektiği kadar hızlı olacaktır.

Kafka’ya gönderilen hisse bilgileri, Stocks-Client projesi içinde bulunan listener tarafından dinlenir ve web socket ile bağlanan tüm client’lara publish edilir. Bu sayede simulation servisinin generate ettiği random hisse fiyatlarını çok hızlı bir şekilde client’a iletmiş oluruz. Stocks-Client projesi kendi başlığında detaylı anlatılacaktır. Genel akış aşağıdaki gibidir.



Stocks-Service

Bu servisin görevi bankada hisse senetleri bilgilerini tutmak ve istendiği/gerektiği taktirde güncel veriyi iletmektir. Simulation servisinin generate ettiği random hisse bilgilerini eğer banka sisteminde kayıt yoksa insert eder, eğer kayıt varsa da günceller. DB’ye ihtiyaç duyan tüm microservisler gibi bu servis de kendi db’sine sahip ve in-memory H2 database’dir.

Stocks-Client

Örnek olarak geliştirilmiş Spring MVC uygulamasıdır. Bu uygulama basic authentication sağlar, içinde sadece **Username = user1, user2 / Password = 123,123** olan kullanıcı bilgileri vardır. Docker compose up işlemi sonrası oluşturulan container’a [**http://localhost:8080**](http://localhost:8080)adresinden ulaşılabilir. Spring security entegre edildiği için login olmadan giriş yapılamayacaktır. Giriş sonrası müşterinin Payday-Bank üzerinden alım-satım yapabileceği hisse senetleri listelenir, müşteri buradan bir işlem yaptığında Stock-Client uygulaması üzerinden Gateway-Server’a istek atılır. Stocks-Client uygulaması gerçekte sadece gateway-server üzerinden işlem yapmalıdır(auth işlemleri dahil). Ancak geliştirme için yeterli zamanım olmadığı için simülasyon amacı ile auth servislerini geliştirmedim, onun yerine spring security’yi configure ederek dummy user’lar ayarladım.