T.C.
SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BSM 401 BİLGİSAYAR MÜHENDİSLİĞİ TASARIMI

1. İLERLEME RAPORU

MİKROSERVİS MİMARİSİ İLE UYGULAMA GELİŞTİRME

B181210091-Furkan ERGÜN

Bölüm : BİLGİSAYAR MÜHENDİSLİĞİ
Danışman : Prof. Dr. Celal ÇEKEN

2021-2022 Güz Dönemi

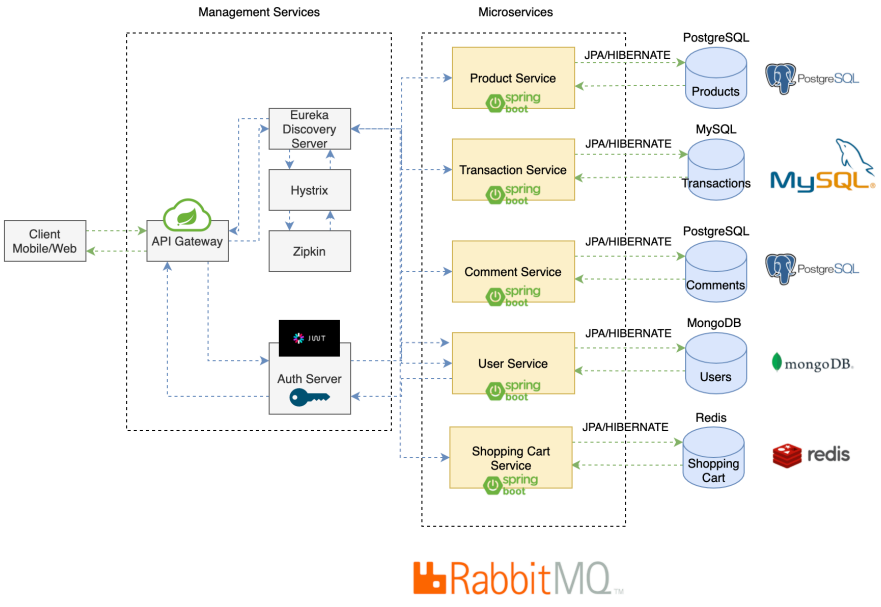
Proje Özeti

Projenin daha iyi anlaşılabilmesi adına öncelikle monolitik mimarinin ve mikroservis mimarinin kısa bir tanımı yapılmalıdır. Monolitik mimari veya monolitik bir uygulama, birden fazla modül içeren tek bir kod tabanına ve tek bir derleme sistemine sahiptir. Bir uygulamanın tüm servislerinin bir çatı altında toplanmış halidir. Buna karşılık mikroservis mimarisi, tek bir parçadan oluşan sistemin, her biri birbirinden bağımsız olarak çalışan ve birbirleriyle haberleşen küçük servislere ayrılarak ortaya çıkarılmış bir mimari çeşididir.

BSM401 Bilgisayar Mühendisliği Tasarımı dersinin projesi kapsamında geliştirilecek somut proje için mikroservis mimarisi ile e-ticaret uygulaması gerçekleme konusu seçildi. Mikroservis mimarisinin çalışma prensibi, nerelerde kullanıldığı, avantajları ve dezavantajları araştırıldı. Mimariyi daha iyi anlamak ve mikroservislerin hangi programlama dilinde geliştirileceğine karar vermek adına farklı programlama dillerinde yazılmış olan örnek mikroservis projeleri incelendi. Yapılan incelemelere göre Java programlama dili tercih edildi. Akabinde bir e-ticaret uygulamasındaki fonksiyonlar, geliştirilecek olan mikroservislerin şeması, mikroservislerin kullanacağı veri tabanı teknolojileri ve mikroservislerin birbirleri ile iletişim şekilleri planlandı. İncelenen örneklere göre 5 adet mikroservis oluşturma planı yapıldı. Birbirleri ile haberleşmeleri için *RabbitMQ ve Eureka Server*, mikroservislerin güvenliği için token sistemi olan *JWT* teknolojisi seçildi. Verileri depolamak için *MongoDB, PostgreSQL, MySQL ve Redis* teknolojileri tercih edildi. Az sayıda bulunan mikroservislerde hata bulmak ve onları çözmek kolay olsa da mikroservislerin sayısı arttıkça çıkan hataların nereden kaynaklandığını bulmak zorlaşmaktadır. Hataları daha kolay şekilde yönetmek adına hata toleransı kütüphanesi olan *Spring Cloud Hystrix* kütüphanesi ile hataları bulma stratejisi izlenilecektir. *Eureka Discovery Server,* mikroservislerimizin birbirleri ile olan iletişimlerini bir noktadan sağlamakla beraber Yük Dengeleme (Load Balancer) görevini de üstlenmektedir. Uygulamayı konteyner hale getirmek için Docker, otomatik deploy etmek ve sayılarını arttırıp azaltmak için de Kubernetes teknolojilerinin kullanılması planlanmaktadır.

Projenin Github Linki: https://github.com/fukuli053/e-commerce-microservices

Sistemin planlanan yapısı **Şekil 1** ‘de bulunan görseldeki gibidir.



Şekil 1

Tamamlananlar

- Literatür araştırması yapıldı. Mikroservis mimarisinin gereksinimleri ve kullanıldığı yerler araştırıldı.
- Farklı programlama dillerindeki projeler incelendi. Yapılacak e-ticaret projesi için Java dili ve Spring'in sağlamış olduğu teknolojiler tercih edildi.
- E-Ticaret sitelerinin veri tabanı modelleri ve mikroservis ile yapılmış proje örnekleri incelendi. İnceleme sonucunda proje sistem şeması ve kullanılacak olan veri tabanlarının diyagramları oluşturuldu.
- Spring Boot ile ilk mikroservis olan Product Service mikroservisi oluşturuldu. Veri tabanı şeması çıkarıldı ve mikroserviste bulunan ürün modeli, veri tabanı şemasına göre yenilendi. PostgreSQL bağlantısı yapıldı.
- Eureka Discovery Server oluşturuldu. Eureka Discovery Server, Sistemde Yük Dengeleyici (Load Balancer) görevini görmekle beraber mikroservislerin tek noktadan iletişim kurmasına imkân sağlamaktadır. Mikroservislerin birbirleri ile kuracakları iletişim konfigürasyonları ve sistemde bulunacak tüm servislerin Eureka Server Client bağlantıları yapıldı.
- Sistemdeki kullanıcıları yönetecek olan User Service oluşturuldu. Veri tabanı şeması ve mikroserviste bulunan kullanıcı modeli oluşturuldu. MongoDB ile bağlantısı yapıldı.
- Mikroservislere istemci tarafından tek adres ve porttan ulaşmamızı sağlayacak olan API Gateway, Spring Cloud Gateway teknolojisi kullanılarak oluşturuldu ve URL konfigürasyon ayarları yapıldı.
- API Gateway'in ve diğer mikroservislerin güvenliğini sağlayacak olan Authentication Server mikroservisi oluşturuldu. User servisi ve API Gateway ile bağlantısı yapıldı. Bu servis sayesinde geçerli JWT (Json Web Token) olmadan, belirlenen mikroservislerin belirlenen adresleri hariç diğer adreslerine ulaşılması engelletildi.
- Üye kayıt işlemi yapıldıktan hemen sonra cevap olarak Access token ve refrest token geri döndürüldü.
- Şu ana kadar oluşturulan tüm mikroservisler gerekli konfigürasyonlar yapılarak Docker konteyneri haline getirildi.

Yapılacaklar

- Uygulamadaki birden fazla servisin loglama ihtiyacını karşılamak için Zipkin teknolojisi projeye dahil edilecek.
- Mikroservislerdeki hataları daha kolay analiz etme amacıyla Hystrix servisi projeye dahil edilecek.
- Transaction servisi oluşturulacak ve diğer servislerle bağlantısı yapılacak.
- Product servisinin modeli yenilenecek.
- Comment servisi oluşturulacak ve Product servisi ile bağlantısı yapılacak.
- RabbitMQ araştırılacak ve Spring Boot ile bağlantıları incelenecek.
- Shopping Cart servisi oluşturacak ve Redis ile bağlantısı yapılacak.
- Auth Server tarafında iyileştirilmeler yapılacak
- Rol ve Yetkilendirme (Authorization) işlemleri eklenecek.
- Yeni oluşturulan servislerin Eureka Server ve Gateway bağlantıları yapılacak.
- Load Balancer için Spring Cloud Gateway iyileştirmesi yapılacak.
- Postman programı ile servisler test edilecek.

- ReactJS Framework’u kullanılarak uygulamanın Front-End kısmı geliştirilecek.
- Front-end kısmı ile API Gateway bağlantısı yapılacak.
- Detaylıca Kubernetes araştırılacak ve uygulama Kubernetes ile deploy edilecek.