JAVA CHALLENGE ANSWERS - ARDA ALTINÖRS

1- SAP Commerce (Hybris) nedir? Hangi amaçlarla kullanılır? Kullandığı teknolojiler nelerdir? Kısaca açıklayınız.

Büyük ölçekli işletmeler için online satışları kolaylaştırmak üzere tasarlanmış kapsamlı bir e-ticaret platformudur. Çok kanallı dijital ticareti yönetmek için sağlam bir altyapı sunar ve işletmelerin kişiselleştirilmiş müşteri deneyimleri sunmasını ve operasyonlarını verimli hale getirilmesini sağlar.

Kullanım Amaçları:

- **E-Ticaret Çözümleri:** Online mağazaların oluşturulması ve yönetilmesi.
- **Müşteri Deneyimi Yönetimi:** Kişiselleştirilmiş ve tutarlı müşteri deneyimleri sunmak.
- **Ürün İçerik Yönetimi:** Ürün kataloglarını ve bilgilerini merkezi bir sistemde yönetmek.
- **Sipariş Yönetimi:** Siparişlerin alınması, işlenmesi ve teslimat süreçlerinin yönetimi.
- **Çok Kanallı Satış:** Farklı satış kanallarında (web, mobil, mağaza içi) entegre bir ticaret deneyimi sağlamak.

Teknolojiler:

- Spring: Java'nın Spring framework'ü ile geliştirilmiş.
- Apache Solr, büyük veriler arasında hızlı arama yapmak için kullanılmış.
- Data Hub: Envanter ve siparişler hakkında gerçek zamanlı güncellemeler sağlayarak ticaret platformu ve ERP sistemleri arasında asenkron veri transferi yapar.
- SAP Cloud Platform Entegrasyonu: SAP'nin diğer bulut hizmetleriyle entegrasyon için.
- Entegrasyon için API'ler: Açık API'ler, üçüncü taraf hizmetler ve mevcut iş sistemleriyle hızlı entegrasyonu kolaylaştırır.

2- Birbirinden bağımsız iki platformun birbiriyle haberleşmesi nasıl sağlanabilir? Örneğin, X platformu Java ile yazılmış olsun, Y platform u C# ile. Bu iki platformun bir biri ile iletişim halinde request-response ilişkisi kurması gerekiyor. Bu yapıyı nasıl sağlarız? Bu iletişim sırasında güvenlik nasıl sağlanır?

İki platformun birbiriyle haberleşmesi için platformdan bağımsız olarak bir haberleşme protokolü seçilir. Aşağıda seçilebilecek protokolleri açıkladım ve kendi projelerimle örnekledim:

➔ **RESTful API ve GraphQL Kullanmak:**
- ◆ **Rest API**, HTTP protokolünü kullanarak iki servisin haberleşmesini sağlar. JSON ve XML formatında verileri döndürür.
- ◆ X platformu bir API sunucusu oluşturur ve belirli işlemler için endpoint'ler oluşturur. Y platformu ise bu endpoint'lere istekler göndererek veriyi alır veya işlemler gerçekleştirir.
- ◆ **GraphQL**, daha esnek ve özelleştirilebilir veri sorguları sunar. İstemciler yalnızca ihtiyaç duydukları alanları sorgulayabilir, bu da veri aktarımını optimize eder.
- ◆ Tyche isimli projemin backend kısmında NodeJS kullanarak böyle bir yapı oluşturmuştum.

➔ **Veri Tabanı Üzerinden Haberleşme**
- ◆ Haberleşme amacıyla pek kullanılmasa da mümkün. İki platform aynı veritabanını kullanarak dolaylı yoldan haberleşebilir.

➔ **WebSocket kullanmak**
- ◆ Anlık haberleşme gerektiren durumlarda kullanılır. Gerçek zamanlı ve çift yönlü haberleşme sağlar.
- ◆ Bu protokolü Full-Stack Facebook Klon projemde kullanmıştım. Django ile geliştirdiğim server bir websocket sunucusu kurup, gönderi beğenileri ve yeni yorumlar gibi eventleri sunucuya gönderiyordu. React ile geliştirdiğim client ise bu sunucuya bağlanıp mesajları dinliyor ve anlık olarak kullanıcı arayüzünde gösteriyordu.

➔ **Message Queue Kullanımı**
- ◆ **RabbitMQ**, **Kafka** gibi platformlar arasında asenkron iletişim sağlamak için uygundur. Yüksek hacimli veriler ve gecikmelere toleranslı uygulamalar için uygundur.
- ◆ Bu protokolü, Borda Teknoloji'de gerçekleştirdiğim staj sırasında kullanmıştık. IoT cihazlarından gelen verileri işlemek için RabbitMQ bir mesaj kuyruğu olarak kullanıldı. IoT cihazlarından gelen veri, önce bir RabbitMQ kuyruğuna gönderiliyor, ardından Python ile geliştirdiğimiz consumer uygulaması bu mesajları kuyruğu dinleyerek alıyor ve işliyordu.

➔ **gRPC Kullanımı**
- ◆ Özellikle yüksek performans ve düşük gecikme gereksinimleri olan sistemler için uygundur. Protobuf kullanarak veri serileştirir ve REST'ten daha hızlıdır.
- ◆ Henüz gRPC kullanarak bir proje geliştirmemiş olsam da, gRPC'nin protokol-bağımsızlığı ve performans avantajları sayesinde IoT veya mikroservis tabanlı projelerde tercih edilebileceğini düşünüyorum.

Güvenlik Önlemleri

- **JWT/OAuth2:** Kullanıcı doğrulaması ve yetkilendirme. Tyche projem hem JWT hem de OAuth2 kullanıyor.
- **TLS/SSL:** Veri transferinde şifreleme
- **API Rate Limiting:** API Gateway veya middleware kullanarak çağrı sınırlarını yönetme.

- **Veri Şifreleme:** Hassas veriler için AES veya RSA gibi şifreleme algoritmaları.

3- SOLR Nedir? Kullanım alanlarını araştırınız. Kurumsal bir projede kullanılabilecek iki farklı kullanım alanı örneği veriniz.

Apache Solr, büyük ölçekli verilerin hızlı ve etkili bir şekilde aranmasını ve indekslenmesini sağlayan açık kaynaklı bir arama platformudur. Java tabanlı olan Solr, Apache Lucene üzerine inşa edilmiş, tam metin arama, faceted arama, gerçek zamanlı indeksleme ve coğrafi konum tabanlı arama gibi güçlü özellikler sunar. Web siteleri, e-ticaret uygulamaları, büyük veri analitiği ve kurumsal içerik yönetimi gibi çeşitli alanlarda yaygın olarak kullanılmaktadır. Solr'ın sağladığı yüksek performans ve esneklik, onu farklı sektörlerdeki projeler için ideal bir çözüm haline getirir.

Kurumsal projelerde Solr'ın kullanım alanları oldukça çeşitlidir. Örneğin, bir şirket içi bilgi portalında Solr kullanılarak çalışanların şirket politikaları, prosedürler ve eğitim materyallerine hızlıca erişmeleri sağlanabilir. Bu sayede bilgiye erişim süresi kısalır ve çalışan verimliliği artar. Bir diğer örnek olarak, e-ticaret platformlarında Solr entegrasyonu sayesinde ürün arama ve filtreleme işlemleri büyük ölçüde iyileştirilebilir. Müşteriler, marka, fiyat, kategori gibi çeşitli kriterlere göre ürünleri kolayca bulabilir ve bu da satışların artmasına katkıda bulunur.

Ek olarak, Solr'ın SAP Commerce (Hybris) gibi büyük kurumsal e-ticaret çözümleriyle entegrasyonu, arama ve kişiselleştirme süreçlerini daha da güçlendirir. Örneğin, bir perakende şirketi, Solr sayesinde müşterilere ilgi alanlarına göre özelleştirilmiş ürün önerileri sunabilir ve çok dilli içerik aramalarını optimize edebilir. Bu tür entegrasyonlar, müşteri deneyimini iyileştirirken işletmelere rekabet avantajı sağlar. Sonuç olarak, Apache Solr, kurumsal projelerde arama ve veri yönetimi ihtiyaçlarını karşılamak için esnek ve güçlü bir araç olarak öne çıkmaktadır.

4- Aşağıdaki algoritma için uygun çözümü üretin.

- Java'da 100 adet random sayıya sahip bir liste oluşturun.
- Daha sonra bu listenin bir kopyasını oluşturun.
- 0 ile 100 arasında rastgele bir sayı üretin.
- Kopya listedeki bu random sayının olduğu indisteski değeri silin.
- Şimdi elinizde iki adet liste var ve kopya listede orjinal listeye göre bir eleman eksik.
- Hangi elemanın eksik olduğunu bulan bir metot oluşturun.

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

public class Main {
    public static void main(String[] args) {
        // 100 elemanlı rastgele sayıya sahip bir liste oluştur
        List<Integer> orjinalListe = createRandomList(100, 100);

        // Listeyi kopyala
        List<Integer> kopyaListe = new ArrayList<>(orjinalListe);

        // 0 ile 100 arasında rastgele bir sayı üretir
        Random rand = new Random();
        int rastgeleIndis = rand.nextInt(100);

        // Kopya listedeki bu rastgele sayının olduğu indeksteki değeri siler
        kopyaListe.remove(rastgeleIndis);

        // eksik elemanı bulur
        int eksikEleman = findMissingElement(orjinalListe, kopyaListe);
        System.out.println("Eksik olan eleman: " + eksikEleman);
    }

    // Rastgele elemanlarla dolu bir liste oluşturan metot
    public static List<Integer> createRandomList(int n, int bound) {
        List<Integer> liste = new ArrayList<>();
        Random rand = new Random();
        for (int i = 0; i < n; i++) {
            liste.add(rand.nextInt(bound));
        }
        return liste;
    }

    // İki liste arasındaki eksik elemanı bulan metot
```

```
   public static int findMissingElement(List<Integer>
orjinal, List<Integer> kopya) {
       Collections.sort(orjinal);
       Collections.sort(kopya);
       for (int i = 0; i < kopya.size(); i++) {
           if (!orjinal.get(i).equals(kopya.get(i))) {
               return orjinal.get(i);
           }
       }
       // Son eleman eksikse
       return orjinal.get(orjinal.size() - 1);
   }
}
```

Açıklama: findMissingElement metodu, orjinal ve kopya listeleri sıralayarak her iki listede bulunan elemanları sırasıyla karşılaştırır. İlk farklılık bulunduğunda, orijinal listedeki bu eleman eksik olan olarak belirlenir. Ancak, sıralama işlemi performansı olumsuz etkileyebilir.

Daha performanslı bir metodun nasıl olabileceğini araştırdığımda, HashSet kullanımı, toplam farkı ve XOR yöntemleri kullanmanın daha optimize sonuçlar verebileceğini buldum.

5- Product, Customer, Cart ve Order tablolarının ve bu tablolarının miras aldığı bir Base Entity'nin bulunduğu
Spring Boot ile geliştirilmiş bir proje oluşturun.

Bir müşterinin bir sepeti (cart) ve birden fazla siparişi (order) olabilecek şekilde ilişkilendirme işlemini gerçekleştirin.
Sepetin ve siparişin toplam fiyat bilgisi her işlemde (sepete ekleme, çıkarma, miktar arttırıp azaltma) hesaplansın ve sepete kaydedilsin.
Bir müşteri sipariş geçtikten sonra, sipariş içerisindeki ürünlerin fiyatı daha sonradan güncellendiğinde müşteri satın aldığı anki fiyatı geçmişe yönelik olarak görebilsin. Bunun için farklı bir tablo tutabilirsiniz. Bu tablo üzerinde ürün, fiyat, miktar gibi bilgiler tutulabilir.
Ürün üzerinde stok takibi yapılsın, bir ürünün stoğu bittikten sonra o üründen daha fazla sipariş verilemesin.

Yazılması beklenen servisler:

AddCustomer

GetProduct

CreateProduct

UpdateProduct

DeleteProduct

GetCart

UpdateCart

EmptyCart

PlaceOrder

GetOrderForCode

GetAllOrdersForCustomer

AddProductToCart

RemoveProductFromCart

Proje repository’si: https://github.com/ardaaltinors/enoca-task

Proje ile ilgili tüm detaylı açıklama README içerisinde bulunuyor.

Challenge Notu

Birinci, ikinci ve üçüncü soruları araştırmalarınız ve kendi yorumlarınız ile açıklayınız. Dördüncü sorudaki isteğe ait kaynak kodların ekran görüntülerini (ve varsa açıklamalarınızı) ilk üç soruyla aynı dokümanda yeni bir sayfaya ekleyiniz. Beşinci soruda projenizi tamamladıktan sonra kaynak kodunuzu bir repository'e (Github, Bitbucket vs.) ekleyin. Proje için kısa bir doküman hazırlayın. İlk 4 soru için dokümanınızı, 5. soru için proje dokümanını ve kaynak kodunuzun olduğu repository linkini maile ekleyiniz. Süre (haftasonu dahil) 5 gündür.