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

SAP Commerce (Hybris), SAP ekosistemi içerisinde müşteri deneyimini yönetmek ve dijital ticaret platformları oluşturmak için kullanılan bir ticaret platformudur. SAP Commerce'u genellikle çevrimiçi satış, müşteri ilişkileri yönetimi (SAP CX), ürün yönetimi, sipariş yönetimi, pazarlama otomasyonu ve dijital ticaret operasyonlarını desteklemek için kullanırlar. SAP Commerce, geniş ölçekli işletmeler için tasarlanmıştır. SAP Commerce'u çeşitli ürünleri ve hizmetleri çevrimiçi olarak satmak için kullanabilirler. Platform, müşteri siparişlerini yönetmek, envanteri izlemek, fiyatlandırmayı kontrol etmek ve ödeme işlemlerini işlemek için gereken özellikleri sunar. Ayrıca SAP Commerce, diğer kurumsal uygulamalarıyla entegrasyon sağlar ve özelleştirilmiş ticaret çözümleri oluşturmak için esnek bir yapı sunar. Böylece, işletmelere, mevcut sistemlerini ve süreçlerini SAP Commerce'a entegre edebilme ve genişletebilme sağlar. Kısacası E-Ticaret çözümlerinde arasında SAP ekosisteminin sunduğu platformdur.

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?

Verileri tüm cihazlara veya başka platformlara göndermek istediğimizde devreye Web Service kavramı girer. Web Service ile platform bağımsız tüm platformlara veri aktarımı gerçekleştirilir. Web servisler platform bağımsız olmak üzere bir çok uygulama, cihaz ya da nodeun birbiri ile iletişim kurmalarını sağlayan yapılardır.Ortak olarak Json ve Xml formatı tüm platformlarda ortak olarak kullanılabiliyor olması bize aynı dilde konuşma imkanı sağlar. Web Servislerde aslında bunu kullanırlar. En yaygın olarak Rest, Soap ve SOA mimarisi kullanılmaktadı. Rest, client-server arasındaki iletişimin HTTP protokolü sayesinde kolay ve hafif bir şekilde yapılmasını sağlayan bir mimaridir. Rest mimarisinde SOAP'da bulunan GetProduct , GetCategory gibi metotlar üzerinden iletişim kurulması yerine tüm bilgiler, URI'ler üzerinden sunulur.

Örn : http://myserver/api/v1/products/1 gibi web linkleri düşünebiliriz.

Restful web servisleri ise , REST mimarisi temel alınarak geliştirilmiş oldukça hafif, genişletilebilir ve basit servislerdir. Restful servislerin amacı client-server arasındaki veri akışını platform bağımsız olarak gerçekleştirebilmek ve veri akışını en az yükle sağlayabilmektir. Restful servisleri response tipi olarak JSON, HTML, XML gibi bir çok formatta çalışabilirler. Yapısının az yer kaplaması ve başka platformda kullanışlı olmasından dolayı response tipi olarak JSON kullanılmaktadır. Restful servisleri esnek ve kolay geliştiriebilir bunun yanında dil ve

platform bağımsızdırlar. SOAP servislerinin aksine ekstra bir kütüphaneye ihtiyaç duymadan çalıştırılabilir. SOAP en temel anlamda, internet üzerinden küçük miktarda bilgileri yada mesajları aktarma protokolüdür. SOAP mesajları XML formatındadırlar ve genellikle HTTP(Hyper Text Transfer Protocol) protokolu(bazende TCP/IP) kullanılarak gönderilirler. SOAP ,XML tabanlı kullanıma mecbur bırakır. Bu konuda esnek değildir. SOAP isteklerini oluşturmak için de bir standart gerekmektedir. İşte burada WSDL devreye girmektedir. WSDL gerçekleştirilebilecek bütün SOAP isteklerinin kayıtarını tutan, XML tabanlı web servisleri tanımlamak ve yerini belirtmek için tanımlanmış dildir. Bu iletişim sırasında güvenliği sağlayabilemk için Kimlik Doğrulama ve Yetkilendirme kullanılabilir. İletişimde yer alan platformlar, kimlik doğrulama ve yetkilendirme mekanizmalarıyla birbirini tanıyabilir ve sadece yetkilendirilmiş kullanıcıların erişimine izin verebilir.

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

SOLR, açık kaynak kodludur. SOLR, büyük miktarda metin tabanlı veriyi hızlı bir şekilde aramak, indekslemek ve analiz etmek için kullanılan bir arama motoru ve metin tabanlı veri deposudur. Büyük metin veri kümelerini hızlı ve kolay bir şekilde arama ve analiz etmeye olanak sağlar. Özellikle kullanım örneklerine bakıldığında daha çok arama işlemlerinde kullanımı yaygındır. Solr, web siteleri, e-ticaret siteleri ve içerik yönetim sistemleri gibi çeşitli platformlarda kullanılan arama motorları için ideal bir çözümdür. Büyük miktarda içeriği hızlı bir şekilde indeksleyebilir ve kullanıcılara hızlı ve etkili arama sonuçları sağlayabilir. Bir e-ticaret platformunda, müşterilerin ürünleri hızlı bir şekilde arayıp bulmalarını sağlamak için Solr kullanılabilir. Solr, ürün katalogunu indeksleyebilir ve müşterilere hızlı, doğru ve özelleştirilmiş arama sonuçları sunabilir.

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 iki metot oluşturun.
- İlk metotta size göre eksik sayıyı bulan EN YAVAŞ metodu yazın.
- İkinci metotta size göre eksik sayıyı bulan EN HIZLI metodu yazın.
- Algoritmanız içerisinde Java kütüphanelerinden gelen hazır metotları (contains,

CollectionUtils metotları vs.) kullanmayın, bunun yerine kendi çözümlerinizi üretin.

NOT: Lütfen internette gördüğünüz veya yapay zekadan oluşturduğunuz metotları kullanmayınız.

En hızlı çözümü üretememiş olsanız bile kendi oluşturduğunuz eşsiz ve mantıklı bir algoritma kullanmayı tercih ediniz.

Açıklama: findMissingNumberSlow metodu, daha yavaş çalışacaktır. Çünkü iç içe döngüler kullanarak her bir sayının orijinal listeye ve kopya listeye kaç kez eklendiğini sayıyor. Bu nedenle, orijinal listedeki her bir eleman için, her iki listenin tamamını tarıyor. Bu durumda, her bir eleman için bir döngü oluşturulduğunda, toplam karmaşıklık $O(n2)$ olur. findMissingNumberFast metodun karmaşıklığı $O(n)$ olduğu için, büyük veri setleriyle daha iyi performans sağlar.

```java
public class Main {
    new *
    public static void main(String[] args) {
        // Orjinal listeyi oluştur
        List<Integer> originalList = createRandomList( size: 100);

        // Kopya listeyi oluştur
        List<Integer> copyList = new ArrayList<>(originalList);

        // Random bir sayı seç ve kopya listeden çıkar
        Random rand = new Random();
        int randomIndex = rand.nextInt( bound: 100);
        int removedNumber = copyList.remove(randomIndex);

        // Eksik sayıları bulan metotları çağır
        int slowMissingNumber = findMissingNumberSlow(originalList, copyList);
        int fastMissingNumber = findMissingNumberFast(originalList, copyList, removedNumber);

        System.out.println("Eksik sayı (EN YAVAŞ): " + slowMissingNumber);
        System.out.println("Eksik sayı (EN HIZLI): " + fastMissingNumber);
    }

    // 100 adet random sayıya sahip bir liste oluştur
    1 usage  new *
    public static List<Integer> createRandomList(int size) {
        List<Integer> list = new ArrayList<>();
        Random rand = new Random();
        for (int i = 0; i < size; i++) {
            list.add(rand.nextInt( bound: 101)); // 0 ile 100 arasında random sayılar
        }
        return list;
    }
```

```java
// Eksik sayıyı bulan EN YAVAŞ metot
1 usage  new *
public static int findMissingNumberSlow(List<Integer> originalList, List<Integer> copyList) {
    for (int num : originalList) {
        int countOriginal = 0;
        int countCopy = 0;

        for (int originalNum : originalList) {
            if (originalNum == num) {
                countOriginal++;
            }
        }

        for (int copyNum : copyList) {
            if (copyNum == num) {
                countCopy++;
            }
        }

        if (countOriginal != countCopy) {
            return num;
        }
    }
    return -1; // Eğer eksik sayı bulunamazsa -1 döndür
}
```

```java
// Eksik sayıyı bulan EN HIZLI metot
1 usage  new *
public static int findMissingNumberFast(List<Integer> originalList, List<Integer> copyList, int removedNumber) {
    int sumOriginal = 0;
    int sumCopy = 0;

    for (int num : originalList) {
        sumOriginal += num;
    }

    for (int num : copyList) {
        sumCopy += num;
    }

    // Orjinal listenin toplamından kopya listenin toplamını çıkararak eksik sayıyı bul
    return sumOriginal - sumCopy;
}
}
```

5. Proje Linki : https://github.com/FatihSengul/enoca-challenge/tree/main/commerce

   Proje Dökümanı: https://github.com/FatihSengul/enoca-challenge/tree/main