Homework 1 - Ahmet Dolma

Q1. Java dünyasındaki framework'ler ve çözdükleri problemler nedir? Kod Örneklendirini de içermelidir

Framework, Türkçe karşılığı çerçeve olan, yazılım geliştiriciler tarafından geliştirilen ve daha öncesinde oluşturulan kütüphanelerin içerisinde bulumduğu kod dökümanlarıdır.

Yazılımcıları sık sık kullanılan kodları mükerrer bir şekilde yazmaktan, mükerrer işlemlerden uzaklaşmaktan ve daha efektif, hızlı ve zamandan tasarruf etmemize yardımcı olurlar.

Örnek Backend Framework'leri:

- Spring: Java tabanlı uygulamalar için kapsamlı bir programlama ve yapılandırma modeli sunar. Spring, güvenlik, veri erişimi, işlemler ve daha fazlasını içeren geniş bir yelpazede hizmetler sağlar.

```
@RestController
public class HelloController {
    @GetMapping("/hello")
    public String hello() {
        return "Hello, World!";
    }
}
```

- **Django:** yüksek seviyeli bir Python web framework'üdür

```
from django.http import HttpResponse

def hello(request):
        return HttpResponse("Hello, World!")
```

- Spring Boot: Spring Boot, Spring framework'ünün üzerine inşa edilmiş ve onun sunduğu özellikleri kolayca kullanabilmek için otomatik yapılandırma, başlangıç bağımlılıkları ve daha fazlasını sunan bir araçtır. Spring Boot, Spring'in karmaşıklığını azaltır ve standart bir uygulamanın hızla geliştirilip dağıtılabilmesi için gereken süreyi kısaltır.

```
@RestController
public class HelloWorldController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, World!";
    }
}
```

Örnek Frontend Framework'leri: Frontend framework'leri (ön uç), bir web sitesinin veya web uygulamasının ön ucunu oluşturmak için HTML, CSS ve JavaScript'in temel şablonlarını içerir.

- React
- Vue.js
- Angular.js

Q2. SOA- Web Service- Restful Service- HTTP methods kavramlarını örneklerle açıklayınız. (15 Puan)

SOA

Açılımı Servıce Orıented Archıteture (Servis Yönelimi Mimarisi)dir. SOA, yazılım bileşenlerinin ağ üzerinden birbirleriyle iletişim kurabilmesini sağlayan bir mimari yapıdır. Bu bileşenler, işlevselliği hizmetler (servisler) olarak sunar ve bu hizmetler, farklı uygulama bileşenleri veya iş işlemleri arasında yeniden kullanılabilir.

Örnek: Bir e-ticaret platformunda, ödeme işlemleri, envanter yönetimi ve müşteri hizmetleri gibi farklı işlevler, bağımsız servisler olarak tasarlanabilir. Bu servisler, SOA mimarisi kullanılarak birbirleriyle iletişim kurabilir. Örneğin, bir siparişin tamamlanması için, ödeme servisi ödemenin yapılmasından sorumlu olurken, envanter servisi ürünün stoktan düşülmesini ve müşteri hizmetleri servisi müşteriye sipariş onayı gönderilmesini sağlayabilir.

WEB SERVICE

Tanım: Web servisleri, farklı ağlar veya platformlar üzerindeki uygulamaların birbiriyle iletişim kurmasını sağlayan yazılım sistemleridir. SOAP (Simple Object Access Protocol) ve REST (Representational State Transfer) gibi protokoller aracılığıyla çalışırlar.

Örnek: Bir hava durumu bilgisi servisi, SOAP protokolünü kullanarak, belirli bir şehir için hava durumu tahminlerini sağlayan bir web servisi olabilir. Kullanıcılar, bu servise SOAP mesajları göndererek hava durumu verilerini sorgulayabilir ve XML formatında yanıt alabilirler.

RESTful Service

Tanım: RESTful servisler, REST mimari prensiplerine dayanan web servisleridir. HTTP protokolünü kullanarak kaynak tabanlı bir yaklaşım sunarlar. Kaynaklar (genellikle veri veya iş nesneleri), URL'ler ile tanımlanır ve standart HTTP metodları (GET, POST, PUT, DELETE) aracılığıyla erişilir ve manipüle edilir.

Örnek:

- Bir blog uygulamasındaki RESTful yorum servisi, bir blog yazısına yorum eklemek için **POST /posts/{postId}/comments** endpoint'ine bir HTTP POST isteği,
- Bir yorumu güncellemek için **PUT /comments/{commentId}** endpoint'ine HTTP PUT,
- bir yorumu silmek için **DELETE /comments/{commentId}** endpoint'ine HTTP DELETE isteği gönderilir.

HTTP Methods

HTTP protokolü, web servisleri ve RESTful API'lerde yaygın olarak kullanılan standart metodlar setini tanımlar:

- **GET:** Bir kaynağın temsilini almak için kullanılır. Örneğin, **GET /users/1** isteği, ID'si 1 olan kullanıcının bilgilerini getirir.
- **POST:** Yeni bir kaynak oluşturmak için kullanılır. Örneğin, **POST /users** ile gönderilen bir istek, yeni bir kullanıcı oluşturabilir.
- **PUT:** Bir kaynağı tamamen güncellemek için kullanılır. Örneğin, **PUT /users/1** isteği, ID'si 1 olan kullanıcının tüm bilgilerini günceller.
- **DELETE:** Bir kaynağı silmek için kullanılır. Örneğin, **DELETE /users/1** isteği, ID'si 1 olan kullanıcıyı siler.
- **PATCH:** Bir kaynağın bir kısmını güncellemek için kullanılır. Örneğin, **PATCH /users/1** isteği, ID'si 1 olan kullanıcının sadece belirli bir alanını güncelleyebilir.

Bu protokoller uygulamanın nasıl çalıştığını görmek, test etmek için kullanabiliriz. (Stress test, Load Test, Peak Test, etc.) (Örnek test uygulamaları: Postman,Locust,Jmeter)