MODUL 4 RESTful API

4.1 Deskripsi Singkat

Representational State Transfer (REST) adalah gaya arsitektur yang digunakan dalam pengembangan layanan web yang berfokus pada prinsip-prinsip yang sederhana dan skalabel untuk mengatur komunikasi antara aplikasi yang berbeda. REST didasarkan pada protokol dan metode yang ada dalam World Wide Web, sehingga memanfaatkan konsep-konsep seperti URI (Uniform Resource Identifier) untuk mengidentifikasi sumber daya dan metode HTTP untuk berinteraksi dengan sumber daya tersebut.

REST mengandalkan beberapa prinsip penting, termasuk:

- Stateless (Tanpa Kondisi): Setiap permintaan klien ke server harus mencakup semua informasi yang diperlukan untuk memahami dan memproses permintaan tersebut. Server tidak menyimpan informasi tentang klien di antara permintaan.
- Interaksi Berdasarkan Sumber Daya: REST memperlakukan segala sesuatu sebagai sumber daya yang dapat diakses melalui URI. Klien berinteraksi dengan sumber daya dengan metode HTTP yang tepat (GET, POST, PUT, DELETE).
- Pemisahan Antar Sumber Daya dan Representasi: REST memisahkan sumber daya (data) dari cara sumber daya tersebut direpresentasikan (misalnya, format JSON atau XML).
- Operasi Tanpa Kondisi (Idempotent): Metode seperti GET dan DELETE adalah operasi tanpa kondisi, yang berarti hasil dari operasi tersebut akan selalu sama jika dipanggil berulang kali.
- Penggunaan Caching: REST mendukung penggunaan caching untuk meningkatkan kinerja. Respons dapat disimpan di sisi klien atau server, mengurangi lalu lintas jaringan.

REST terkenal karena kesederhanaan dan fleksibilitasnya. Ini dapat digunakan dalam berbagai kasus, termasuk pembangunan aplikasi web, layanan web API, dan sistem mikroservis. Keuntungan utamanya

termasuk efisiensi komunikasi karena format data yang ringan seperti JSON, kemudahan dalam pengembangan, serta kemampuan untuk diimplementasikan di berbagai bahasa pemrograman.

Konsep Utama REST:

- URI (Uniform Resource Identifier): Mengidentifikasi sumber daya dengan alamat unik.
- Metode HTTP: Menggunakan metode seperti GET, POST, PUT, dan DELETE untuk berinteraksi dengan sumber daya.
- Representasi: Menggunakan format data seperti JSON atau XML untuk merepresentasikan sumber daya.
- Stateless: Tidak menyimpan informasi antara permintaan.

Kritik terhadap REST meliputi beberapa keterbatasan dalam mengatasi kompleksitas bisnis yang tinggi atau dalam situasi di mana perlu banyak metadata. REST juga dapat menjadi kurang optimal dalam hal kinerja jika digunakan dalam skenario yang membutuhkan lalu lintas jaringan yang besar.

REST telah menjadi pendekatan populer dalam pembangunan layanan web yang berfokus pada kesederhanaan, skalabilitas, dan fleksibilitas. Dengan memanfaatkan prinsip-prinsip seperti statelessness dan interaksi berdasarkan sumber daya, REST memberikan cara yang efisien dan terstandarisasi untuk menghubungkan aplikasi dan berbagai sistem di dunia yang semakin terhubung secara digital.

4.2 Tujuan Praktikum

Tujuan praktikum ini adalah membuat suatu layanan menggunakan RESTful API yang diimplementasikan dengan Spring Framework.

Setelah praktikum ini kompetensi yang akan dicapai adalah mahasiswa mampu memahami konsep teknologi web service dan memahami implementasi web service dengan RESTful API.

4.3 Alokasi Waktu

Alokasi waktu pada praktikum ini adalah sebagai berikut :

1. Tatap muka di kelas : 50 menit

2. Kerja mandiri (mengerjakan penugasan) : 120 menit

4.4 Material Praktikum

Praktikum ini memerlukan beberapa material sebagai berikut:

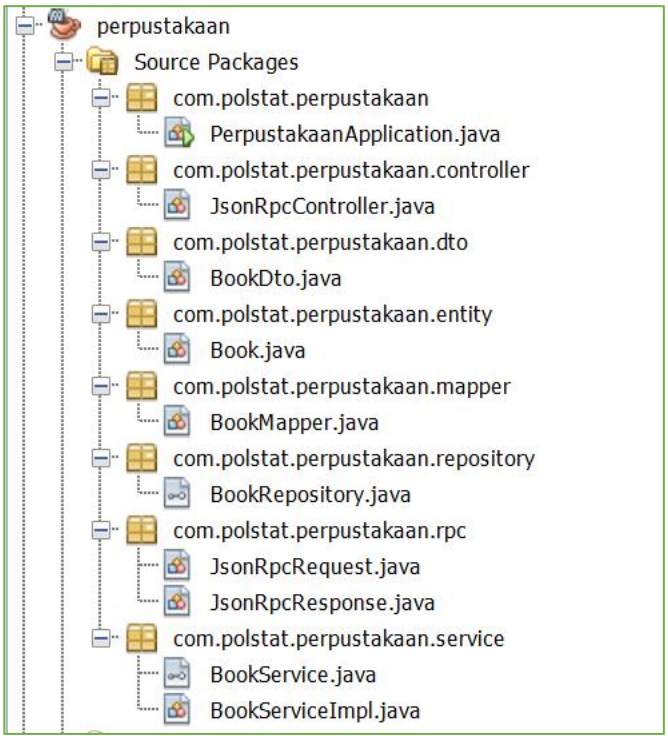
1) Java Development Kit (JDK) versi 17,
2) Java IDE: Netbeans, IntelliJ IDEA, atau Spring Tool Suite (STS),
3) Maven 3.5+,
4) MySQL,
5) Postman https://www.postman.com/downloads/.

4.5 Kegiatan Praktikum

Pada praktikum ini kita akan memodifikasi aplikasi perpustakaan yang telah dibuat pada praktikum 2 dengan menambahkan operasi CRUD anggota (member) perpustakaan. Ikuti langkah-langkahnya sebagai berikut.

Langkah 1: Menyiapkan Proyek

1) Proyek pada modul praktikum 4 ini menggunakan proyek perpustakaan pada modul praktikum 2.
2) Struktur folder proyek perpustakaan seperti gambar berikut ini.



3) Tambahkan dependency Rest Repository pada file pom.xml.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

Langkah 2: Membuat Kode Program

1) Membuat Kelas Entity Member

Kelas Member adalah kelas entity yang merepresentasikan tabel member dalam database. Atribut kelas ini antara lain: memberID, name, address, dan phoneNumber.

Buatlah kelas Member.java pada package *com.polstat.perpustakaan.entity*. Berikut kode program selengkapnya.

```
package com.polstat.perpustakaan.entity;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
```

```
import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Setter
@Getter
@AllArgsConstructor
@NoArgsConstructor
@Builder
@Entity
@Table(name = "members")
public class Member {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @Column(nullable = false)
   private String memberID;
   @Column(nullable = false)
   private String name;
   @Column(nullable = false)
   private String address;
   @Column(nullable = true)
   private String phoneNumber;
}
```

2) Membuat Kelas Repository Member

Kelas ini selain untuk menangani operasi database (CRUD) juga menangani *request* RESTful API yang ditandai dengan anotasi @RepositoryRestResource. Spring akan otomatis membuat kelas ini dapat menangani *request* dengan *endpoint* sebagai berikut:

- GET /member: medapatkan data semua member
- POST /member: membuat member baru
- PUT /member/{:id}: mengubah data member berdasarkan id
- PATCH /member/{:id}: mengubah beberapa atribut member berdasarkan id.
- DELETE /member/{:id}: menghapus data member berdasarkan member

- GET /member/search/findByName?name={:name}: mencari data member berdasarkan nama.
- GET /member/search/findByMemberID?member_id={:member_id}: mencari data member berdasarkan memberID.

Buatlah interface *MemberRepository.java* pada package *com.polstat.perpustakaan.repository*.

```
package com.polstat.perpustakaan.repository;

import com.polstat.perpustakaan.entity.Member;
import java.util.List;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource(collectionResourceRel = "member", path = "member")
public interface MemberRepository extends PagingAndSortingRepository<Member,
Long>, CrudRepository<Member,Long> {
   List<Member> findByName(@Param("name") String name);
   List<Member> findByMemberID(@Param("member_id") String memberID);
}
```

Di dalam MemberRepository, terdapat dua metode yang didefinisikan. Kedua metode ini digunakan untuk melakukan pencarian data anggota berdasarkan nama dan ID anggota. Anotasi @Param digunakan untuk menunjukkan bahwa parameter metode tersebut akan diikuti oleh parameter dengan nama yang sesuai dalam permintaan HTTP.

Langkah 3: Menguji Proyek

1) Mejalan Proyek

   Jalankan proyek dengan klik menu Run File pada file *PerpustakaanApplication.java*.

2) Menguji RESTful API dengan Postman

   Lakukan pengujian API melalui Postman untuk beberapa operasi sebagai berikut:

a) Menambahkan Member

- URI: `http://localhost:8080/member`
- Method: `POST`
- Data:

```
{"memberID": "111", "name": "Ahmad","address": "Jalan Merdeka No. 1","phoneNumber": "082193223xxx"}
```

- Headers: `Content-Type:application/json`

b) Menampilkan semua Member

- URI: `http://localhost:8080/member`
- Method: `GET`

c) Mendapatkan data Member berdasarkan ID

- URI: `http://localhost:8080/member/1`
- Method: `GET`

d) Mengubah data Member

- URI: `http://localhost:8080/member/1`
- Method: `PUT`
- Data:

```
{"memberID": "111", "name": "Ahmad Anas","address": "Jalan Merdeka No. 2","phoneNumber": "082193223xxx"}
```

- Headers: `Content-Type:application/json`

e) Mengubah nomor telepon Member

- URI: `http://localhost:8080/member/1`
- Method: `PATCH`
- Data: `{"phoneNumber": "082193223111"}`
- Headers: `Content-Type:application/json`

f) Mencari data Member berdasarkan memberID

- URI:
  `http://localhost:8080/member/search/findByMemberID?member_id=111`
- Method: `GET`

g) Mencari data Member berdasarkan nama

- URI:
  `http://localhost:8080/member/search/findByName?name=Ahmad%20Anas`
- Method: `GET`

h) Menghapus Member

- URI: `http://localhost:8080/member/1`
- Method: `DELETE`

Output respon dari masing-masing endpoint berformat HATEAOS.

4.6 Penugasan

Modifikasi aplikasi perpustakaan dengan menambahkan RESTful API untuk layanan peminjaman dan pengembalian buku oleh member. Atribut peminjaman buku antara lain: id member yang meminjam, id buku yang dipinjam, tanggal pinjam, tanggal kembali, status peminjaman (sedang dipinjam, sudah dikembalikan), dan jumlah hari telat dikembalikan.

Buat laporan pekerjaan Anda dengan penjelasan yang meliputi penjelasan kode program yang dibuat, pengujian semua enpoint beserta *output* responnya. Buat laporan dalam format pdf.