### Laporan Praktikum 5.5: Implementasi GraphQL untuk Operasi CRUD Entity Member

#### 1. **Tujuan**

Praktikum ini bertujuan untuk memodifikasi aplikasi perpustakaan yang sebelumnya telah dibuat dengan menambahkan layanan GraphQL yang mendukung operasi CRUD pada entity **Member**. Dengan implementasi GraphQL ini, diharapkan aplikasi dapat menyediakan API yang lebih fleksibel dan efisien dalam mengelola data.

#### 2. **Langkah-Langkah Pengerjaan**

##### 2.1 **Persiapan Proyek**

* Gunakan proyek yang telah dibuat pada praktikum 4 sebagai dasar pengembangan.
* Tambahkan dependency **Spring Boot GraphQL** pada file pom.xml:
* <dependency>  
   <groupId>org.springframework.boot</groupId>  
   <artifactId>spring-boot-starter-graphql</artifactId>  
  </dependency>

##### 2.2 **Membuat Kode Program**

* **Schema GraphQL untuk Entity Member**: Membuat schema GraphQL untuk mendefinisikan tipe data dan operasi CRUD pada entity **Member**. File schema.graphqls yang terletak di folder src/main/resources/graphql diisi dengan kode berikut:

type Book {  
 id: ID  
 title: String  
 author: String  
 description: String  
}  
type Member {  
 id: ID  
 memberID: String  
 name: String  
 address: String  
 phoneNumber: String  
}  
  
type Query {  
 books:[Book]  
 bookById(id: ID): Book  
  
 members: [Member]  
 memberById(id: ID): Member  
}  
type Mutation {  
 createBook(title: String!, description: String, author: String!) : Book!  
 updateBook(id:String!, title: String!, description: String, author: String!) : Book!  
 deleteBook(id:String!): Book  
  
 createMember(memberID: String!, name: String!, address: String!, phoneNumber: String): Member!  
 updateMember(id: ID!, memberID: String!, name: String!, address: String!, phoneNumber: String): Member!  
 deleteMember(id: ID!): Member  
}

* **Service Layer untuk Entity Member**: Membuat/mengubah service layer dengan menambahkan method CRUD untuk entity **Member** pada interface MemberService:

package com.polstat.perpustakaan.service;  
  
import com.polstat.perpustakaan.dto.MemberDto;  
import java.util.List;  
  
public interface MemberService {  
 MemberDto createMember(MemberDto memberDto);  
 List<MemberDto> getMembers();  
 MemberDto getMember(Long id);  
 MemberDto updateMember(MemberDto memberDto);  
 void deleteMember(Long id);  
}

* **Implementasi Service Layer**: Implementasi service CRUD untuk entity **Member** pada kelas MemberServiceImpl:

package com.polstat.perpustakaan.service;  
  
import com.polstat.perpustakaan.dto.MemberDto;  
import com.polstat.perpustakaan.entity.Member;  
import com.polstat.perpustakaan.mapper.MemberMapper;  
import com.polstat.perpustakaan.repository.MemberRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@Service  
public class MemberServiceImpl implements MemberService {  
  
 @Autowired  
 private MemberRepository memberRepository;  
  
 @Override  
 public MemberDto createMember(MemberDto memberDto) {  
 Member member = memberRepository.save(MemberMapper.mapToMember(memberDto));  
 return MemberMapper.mapToMemberDto(member);  
 }  
  
 @Override  
 public List<MemberDto> getMembers() {  
 List<Member> members = (List<Member>) memberRepository.findAll();  
 return members.stream().map(MemberMapper::mapToMemberDto).collect(Collectors.toList());  
 }  
  
 @Override  
 public MemberDto getMember(Long id) {  
 Member member = memberRepository.findById(id).orElseThrow(() -> new RuntimeException("Member not found"));  
 return MemberMapper.mapToMemberDto(member);  
 }  
  
 @Override  
 public MemberDto updateMember(MemberDto memberDto) {  
 Member member = memberRepository.save(MemberMapper.mapToMember(memberDto));  
 return MemberMapper.mapToMemberDto(member);  
 }  
  
 @Override  
 public void deleteMember(Long id) {  
 memberRepository.deleteById(id);  
 }  
}

* **GraphQL Controller untuk Entity Member**: Membuat MemberGraphqlController untuk menangani request GraphQL:

package com.polstat.perpustakaan.controller;  
  
import com.polstat.perpustakaan.dto.MemberDto;  
import com.polstat.perpustakaan.service.MemberService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.graphql.data.method.annotation.Argument;  
import org.springframework.graphql.data.method.annotation.MutationMapping;  
import org.springframework.graphql.data.method.annotation.QueryMapping;  
import org.springframework.stereotype.Controller;  
  
import java.util.List;  
  
@Controller  
public class MemberGraphqlController {  
  
 @Autowired  
 private MemberService memberService;  
  
 @QueryMapping  
 public List<MemberDto> members() {  
 return memberService.getMembers();  
 }  
  
 @QueryMapping  
 public MemberDto memberById(@Argument Long id) {  
 return memberService.getMember(id);  
 }  
  
 @MutationMapping  
 public MemberDto createMember(@Argument String memberID, @Argument String name, @Argument String address, @Argument String phoneNumber) {  
 MemberDto memberDto = MemberDto.builder()  
 .memberID(memberID)  
 .name(name)  
 .address(address)  
 .phoneNumber(phoneNumber)  
 .build();  
 return memberService.createMember(memberDto);  
 }  
  
 @MutationMapping  
 public MemberDto updateMember(@Argument Long id, @Argument String memberID, @Argument String name, @Argument String address, @Argument String phoneNumber) {  
 MemberDto memberDto = memberService.getMember(id);  
 memberDto.setMemberID(memberID);  
 memberDto.setName(name);  
 memberDto.setAddress(address);  
 memberDto.setPhoneNumber(phoneNumber);  
 return memberService.updateMember(memberDto);  
 }  
  
 @MutationMapping  
 public void deleteMember(@Argument Long id) {  
 memberService.deleteMember(id);  
 }  
}

* **Pengujian GraphQL**: Untuk menguji GraphQL, kita menggunakan **GraphiQL Playground**. Aktifkan fitur ini dengan menambahkan properti di file application.properties:
* spring.graphql.graphiql.enabled=true

#### 3. **Pengujian Proyek**

Pengujian dilakukan menggunakan **GraphiQL Playground** dengan mengakses URL http://localhost:8080/graphiql?path=/graphql.

Berikut beberapa pengujian yang dilakukan:

1. **Menambah Member**

* mutation {  
   createMember(memberID: "M001", name: "John Doe", address: "123 Main St", phoneNumber: "08123456789") {  
   id  
   memberID  
   name  
   address  
   phoneNumber  
   }  
  }
* **Hasil**:
* {  
   "data": {  
   "createMember": {  
   "id": "1",  
   "memberID": "M001",  
   "name": "John Doe",  
   "address": "123 Main St",  
   "phoneNumber": "08123456789"  
   }  
   }  
  }

1. **Menampilkan Semua Member**

* query {  
   members {  
   id  
   memberID  
   name  
   address  
   phoneNumber  
   }  
  }
* **Hasil**:
* {  
   "data": {  
   "members": [  
   {  
   "id": "1",  
   "memberID": "M001",  
   "name": "John Doe",  
   "address": "123 Main St",  
   "phoneNumber": "08123456789"  
   }  
   ]  
   }  
  }

1. **Mengubah Member**

* mutation {  
   updateMember(id: 1, memberID: "M002", name: "John Updated", address: "456 Updated St", phoneNumber: "08987654321") {  
   id  
   memberID  
   name  
   address  
   phoneNumber  
   }  
  }

1. **Menghapus Member**

* mutation {  
   deleteMember(id: 1)  
  }

#### 4. **Kesimpulan**

Dengan implementasi GraphQL ini, aplikasi perpustakaan dapat menangani operasi CRUD untuk entity **Member** secara lebih fleksibel dan efisien. Fitur ini memudahkan pengguna dalam mengelola data, terutama ketika hanya ingin mengakses bagian data tertentu tanpa harus mengirim permintaan besar seperti dalam RESTful API.

package com.polstat.perpustakaan.dto;  
  
import lombok.Builder;  
import lombok.Data;  
  
@Data  
@Builder  
public class MemberDto {  
 private Long id;  
 private String memberID;  
 private String name;  
 private String address;  
 private String phoneNumber;  
}

package com.polstat.perpustakaan.mapper;  
  
import com.polstat.perpustakaan.dto.MemberDto;  
import com.polstat.perpustakaan.entity.Member;  
  
public class MemberMapper {  
  
 // Mengubah Member menjadi MemberDto  
 public static MemberDto mapToMemberDto(Member member) {  
 return MemberDto.builder()  
 .id(member.getId())  
 .memberID(member.getMemberID())  
 .name(member.getName())  
 .address(member.getAddress())  
 .phoneNumber(member.getPhoneNumber())  
 .build();  
 }  
  
 // Mengubah MemberDto menjadi Member  
 public static Member mapToMember(MemberDto memberDto) {  
 Member member = new Member();  
 member.setId(memberDto.getId());  
 member.setMemberID(memberDto.getMemberID());  
 member.setName(memberDto.getName());  
 member.setAddress(memberDto.getAddress());  
 member.setPhoneNumber(memberDto.getPhoneNumber());  
 return member;  
 }  
}

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);  
}