5.6 Penugasan

Member.java

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

import jakarta.persistence.*;
import lombok.*;

@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;
}
```

Kode ini bertanggung jawab untuk mendefinisikan dan merepresentasikan tabel members dalam database, dimana berisikan 5 atribut diantaranya id, memberId, name, address dan phoneNumber.

MemberGraphController.java

```java
package com.polstat.perpustakaan.controller;

import com.polstat.perpustakaan.dto.MemberDto;
import com.polstat.perpustakaan.service.MemberService;
import java.util.List;
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;

@Controller
public class MemberGraphqlController {
    @Autowired
    private MemberService memberService;

    @QueryMapping
    public List<MemberDto> members() {
        return memberService.getMembers(null);
    }

    @QueryMapping
    public MemberDto memberById(@Argument String memberID) {
        List<MemberDto> members = memberService.getMembers(memberID);
        if (members.isEmpty()) {
            return null;
        }
        return members.get(0);
    }

    @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.getMemberById(id);
        if (memberDto == null) {
            throw new RuntimeException("Member not found with ID: " + id);
        }

        memberDto.setMemberID(memberID); // Optional: if you want to update 
memberID too
        memberDto.setName(name);
        memberDto.setAddress(address);
        memberDto.setPhoneNumber(phoneNumber);
        return memberService.updateMember(memberDto);
    }

    @MutationMapping
    public String deleteMember(@Argument Long id) {
        try {
            memberService.deleteMember(id);
            return "Member with ID " + id + " deleted successfully.";
        } catch (RuntimeException e) {
            throw new RuntimeException("Failed to delete member: " + 
e.getMessage());
        }
    }
}
```

Kode ini bertanggung jawab untuk melakukan pengelolaan operasi GraphQL untuk entitas Member, dimana terdapat 2 query dan 3 mutation yang masing – masing berguna untuk menampilkan semua member, menampilkan member berdasarkan id, membuat member, mengupdate member dan menghapus member.

MemberDto.java

```
package com.polstat.perpustakaan.dto;

import jakarta.validation.constraints.NotEmpty;
import jakarta.validation.constraints.NotNull;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
```

```java
public class MemberDto {
    private Long id;
    @NotEmpty(message = "ID member wajib diisi")
    private String memberID;
    private String name;
    @NotNull(message = "Nama member wajib diisi")
    private String address;
    private String phoneNumber;
}
```

Kode ini berfungsi sebagai wadah untuk menyimpan dan mentransfer data terkait member.

MemberMapper.java

```java
package com.polstat.perpustakaan.mapper;

import com.polstat.perpustakaan.dto.MemberDto;
import com.polstat.perpustakaan.entity.Member;

public class MemberMapper {

    public static Member mapToMember(MemberDto memberDto) {
        return Member.builder()
                .id(memberDto.getId())
                .memberID(memberDto.getMemberID())
                .name(memberDto.getName())
                .address(memberDto.getAddress())
                .phoneNumber(memberDto.getPhoneNumber())
                .build();
    }

    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();
    }
}
```

Kode ini berfungsi untuk memudahkan konversi antara dua jenis objek yang berhubungan dengan member. Ini membantu memisahkan logika antara penyimpanan data dan pengiriman data, yang penting untuk menjaga kebersihan dan keteraturan kode dalam aplikasi.

MemberRepository.java

```java
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);
}
```

Kode ini berfungsi untuk melakukan operasi database pada entitas member.

MemberService.java

```java
package com.polstat.perpustakaan.service;

import com.polstat.perpustakaan.dto.MemberDto;
import java.util.List;

public interface MemberService {
    MemberDto createMember(MemberDto memberDto);
    MemberDto updateMember(MemberDto memberDto);
    void deleteMember(Long id);
    List<MemberDto> getMembers(String memberID);
    MemberDto getMemberById(Long id);
}
```

Kode ini mendefinisikan metode yang terkait dengan member, diantaranya yaitu untuk membuat member, mengupdate member, menghapus member, serta mendapatkan member baik berdasarkan ID yang otomatis ter*generate* maupun berdasarkan memberId.

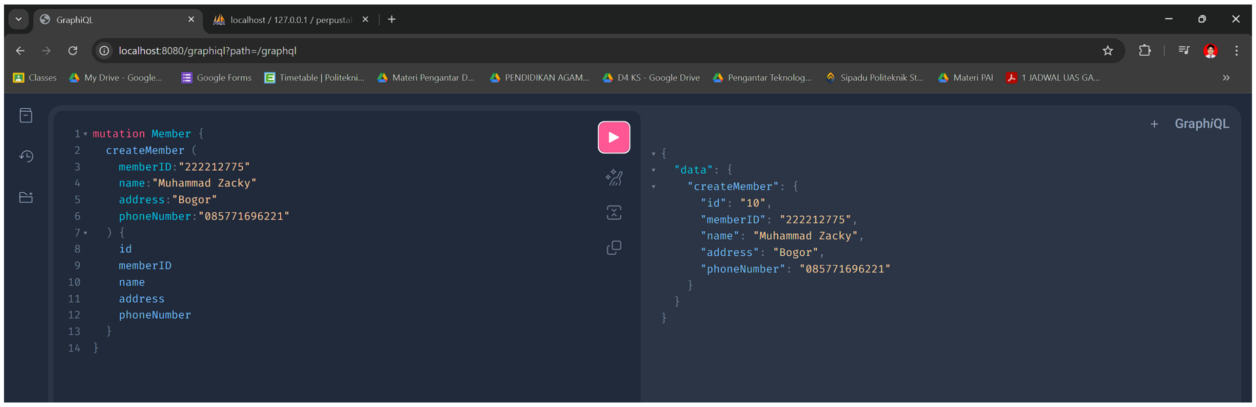
MemberServiceImpl.java

```java
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 java.util.List;
import java.util.Optional;
import java.util.stream.Collectors;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@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 MemberDto updateMember(MemberDto memberDto) {
        Member member =
memberRepository.save(MemberMapper.mapToMember(memberDto));
        return MemberMapper.mapToMemberDto(member);
    }

    @Override
    public void deleteMember(Long id) {
        Member member = memberRepository.findById(id)
                .orElseThrow(() -> new RuntimeException("Member not found with
ID: " + id));
        memberRepository.delete(member);
    }


    @Override
    public List<MemberDto> getMembers(String memberID) {
        if (memberID == null || memberID.isEmpty()) {
            List<Member> members = (List<Member>) memberRepository.findAll();
            return members.stream()
                    .map(MemberMapper::mapToMemberDto)
                    .collect(Collectors.toList());
        } else {
            List<Member> members = memberRepository.findByMemberID(memberID);
            return members.stream()
                    .map(MemberMapper::mapToMemberDto)
                    .collect(Collectors.toList());
        }
    }

    public MemberDto getMemberById(Long id) {
        Optional<Member> memberOptional = memberRepository.findById(id);
        return memberOptional.map(MemberMapper::mapToMemberDto)
                             .orElse(null); // Return null if not found
    }
}
```

Kode ini mengimpelemtasikan logika bisnis untuk mengelola pencatatan member. Ini mencakup pembuatan, pengupdatean, penghapusan serta pemanggilan list member berdasarkan ID.

Uji Coba GraphQL

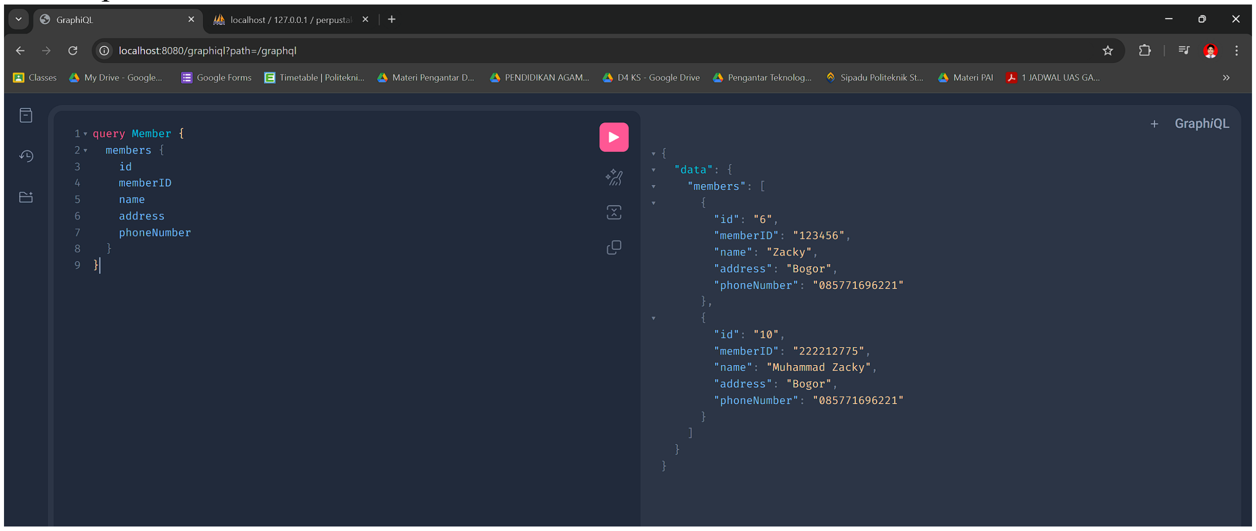
- Menambahkan Member



```
mutation Member {
  createMember (
    memberID:"222212775"
    name:"Muhammad Zacky"
    address:"Bogor"
    phoneNumber:"085771696221"
  ) {
    id
    memberID
    name
    address
    phoneNumber
  }
}
```

```
{
  "data": {
    "createMember": {
      "id": "10",
      "memberID": "222212775",
      "name": "Muhammad Zacky",
      "address": "Bogor",
      "phoneNumber": "085771696221"
    }
  }
}
```

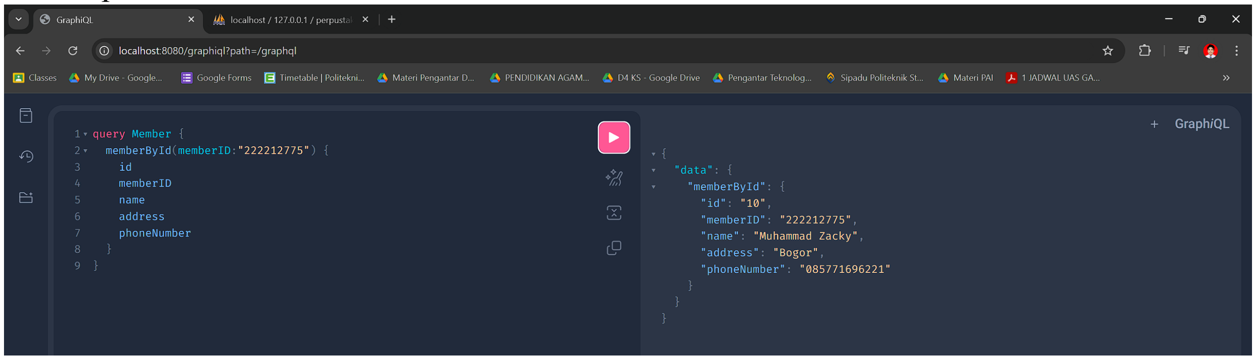
- Menampilkan List Member



```
query Member {
  members {
    id
    memberID
    name
    address
    phoneNumber
  }
}
```

```
{
  "data": {
    "members": [
      {
        "id": "6",
        "memberID": "123456",
        "name": "Zacky",
        "address": "Bogor",
        "phoneNumber": "085771696221"
      },
      {
        "id": "10",
        "memberID": "222212775",
        "name": "Muhammad Zacky",
        "address": "Bogor",
        "phoneNumber": "085771696221"
      }
    ]
  }
}
```

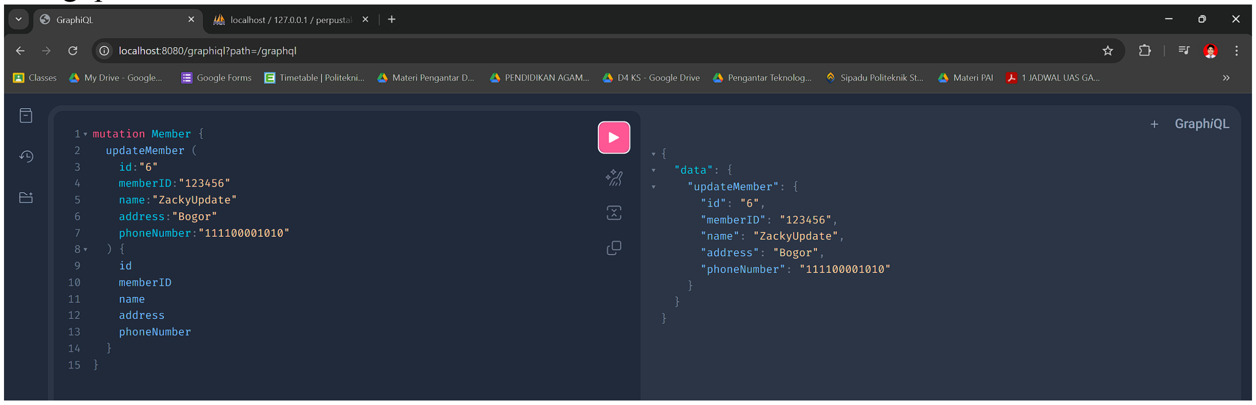
- Menampilkan List Member berdasarkan memberID



```
query Member {
  memberById(memberID:"222212775") {
    id
    memberID
    name
    address
    phoneNumber
  }
}
```

```
{
  "data": {
    "memberById": {
      "id": "10",
      "memberID": "222212775",
      "name": "Muhammad Zacky",
      "address": "Bogor",
      "phoneNumber": "085771696221"
    }
  }
}
```

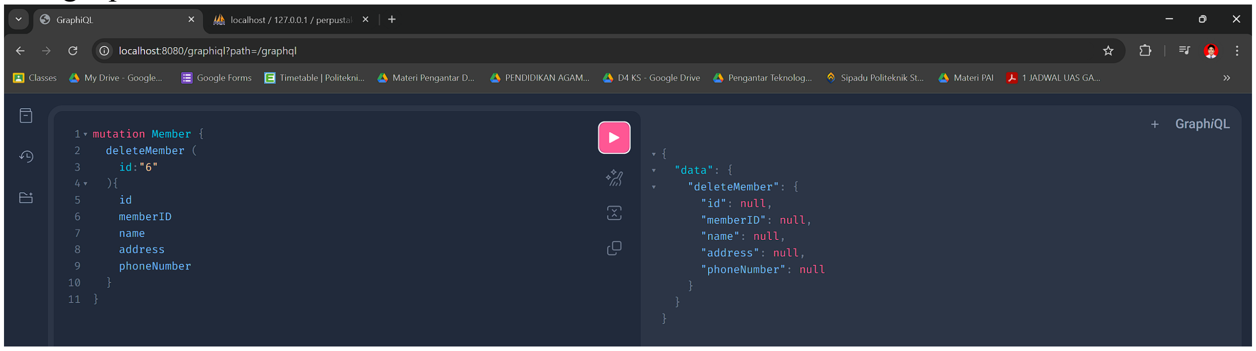
- Mengupdate Member



```
mutation Member {
  updateMember (
    id:"6"
    memberID:"123456"
    name:"ZackyUpdate"
    address:"Bogor"
    phoneNumber:"111100001010"
  ) {
    id
    memberID
    name
    address
    phoneNumber
  }
}
```

```
{
  "data": {
    "updateMember": {
      "id": "6",
      "memberID": "123456",
      "name": "ZackyUpdate",
      "address": "Bogor",
      "phoneNumber": "111100001010"
    }
  }
}
```

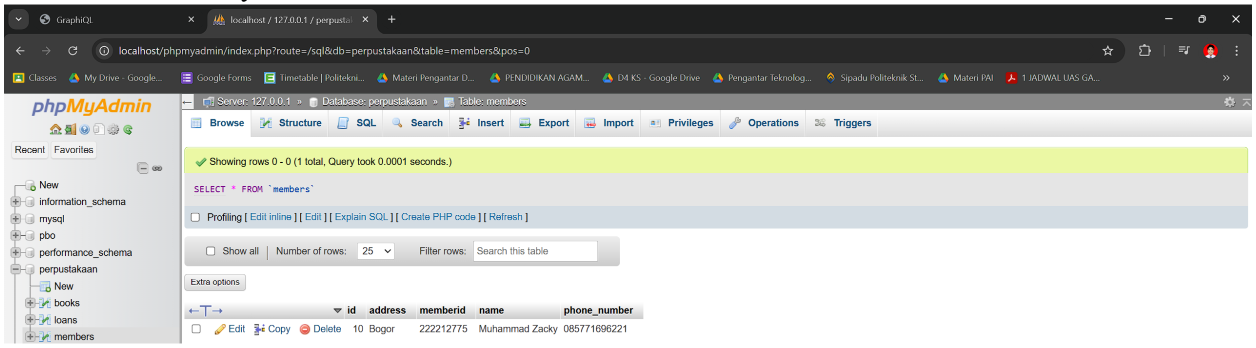
- Menghapus List Member



```
mutation Member {
  deleteMember (
    id:"6"
  ){
    id
    memberID
    name
    address
    phoneNumber
  }
}
```

```
{
  "data": {
    "deleteMember": {
      "id": null,
      "memberID": null,
      "name": null,
      "address": null,
      "phoneNumber": null
    }
  }
}
```

Dan berikut hasilnya dalam database



localhost/phpmyadmin/index.php?route=/sql&db=perpustakaan&table=members&pos=0

Server: 127.0.0.1 » Database: perpustakaan » Table: members

Showing rows 0 - 0 (1 total, Query took 0.0001 seconds.)

```
SELECT * FROM `members`
```

	id	address	memberid	name	phone_number
Edit Copy Delete	10	Bogor	222212775	Muhammad Zacky	085771696221