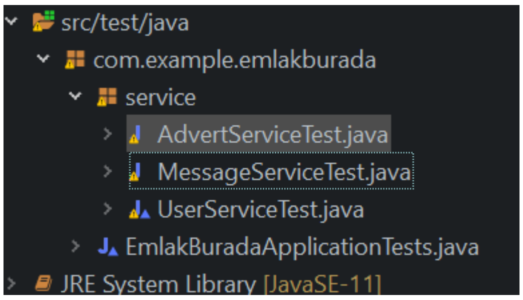
Projede sadece belirli işlemleri yapıldı. Projenin tüm aşamalarını çeşitli nedenlerden yetiştiremedim. Eksik olan aşamalar; Loglama,UML diagram, NoSQL, Projenin Paket servisi gibi Projenin belli bölümlerini yetiştiremedim. Yetiştirebildiğim alanlar projenin Unit Test, ExceptionHandle , ORM, Hibarnate, RabbitMq, Map leme gibi temel işelmleri yetiştirebildim ve yetiştirebildiğim bu alanları ekledim. Eklenilen bölümlerle ilgili olarak;

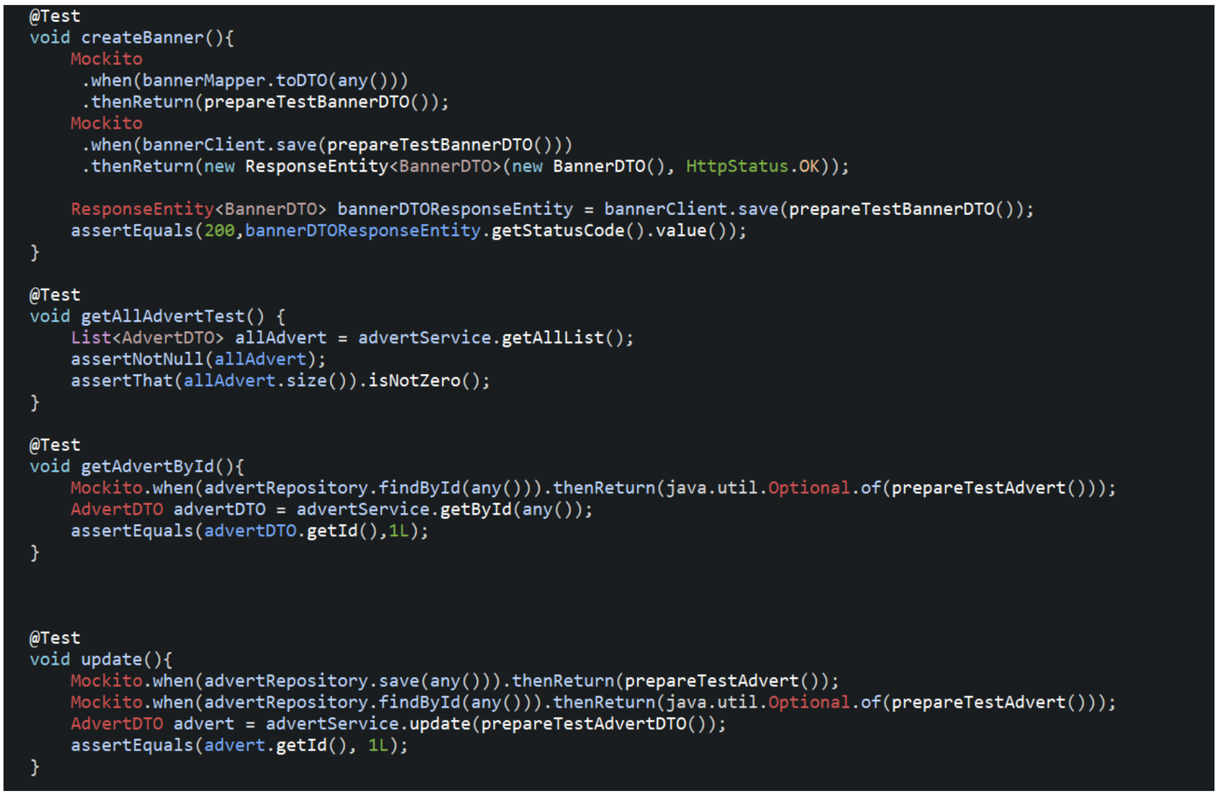
Unit Test;

EmlakBurada Unit Test;



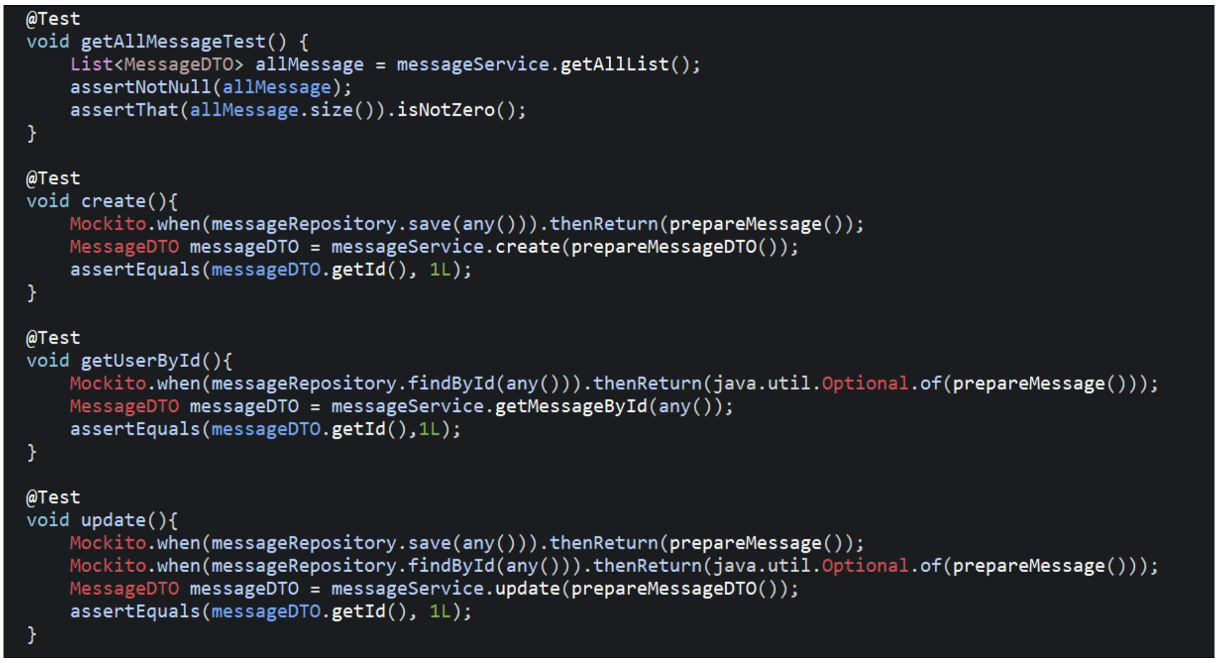
```
src/test/java
  com.example.emlakburada
    service
      AdvertServiceTest.java
      MessageServiceTest.java
      UserServiceTest.java
    EmlakBuradaApplicationTests.java
JRE System Library [JavaSE-11]
```

AdvertServiceTest;



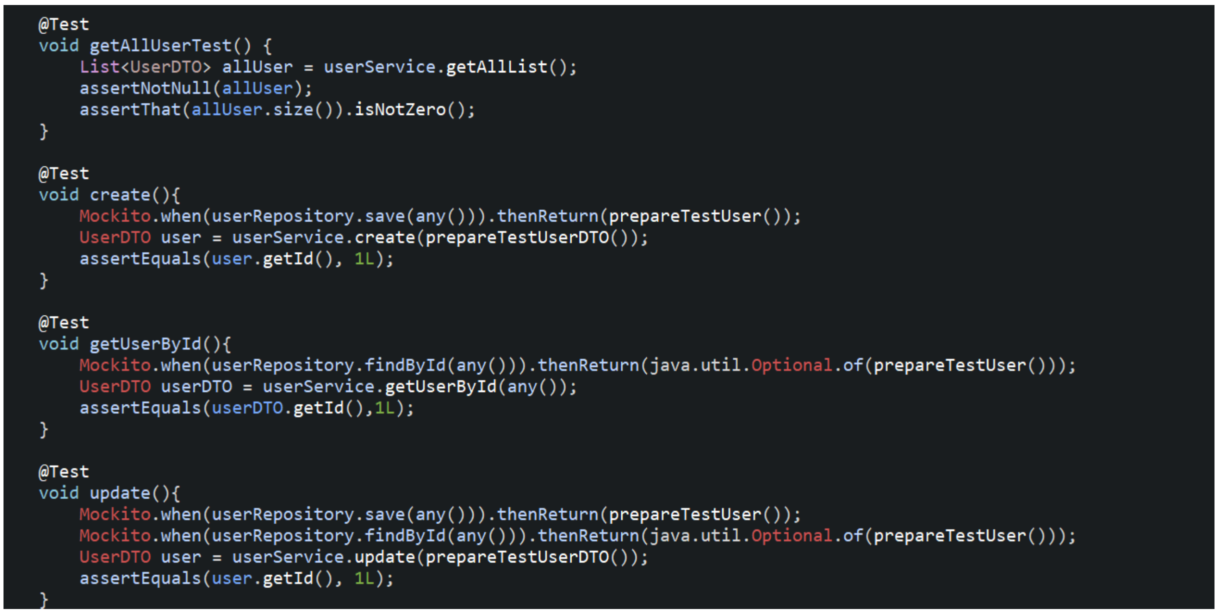
```
@Test
void createBanner(){
    Mockito
     .when(bannerMapper.toDTO(any()))
     .thenReturn(prepareTestBannerDTO());
    Mockito
     .when(bannerClient.save(prepareTestBannerDTO()))
     .thenReturn(new ResponseEntity<BannerDTO>(new BannerDTO(), HttpStatus.OK));

    ResponseEntity<BannerDTO> bannerDTOResponseEntity = bannerClient.save(prepareTestBannerDTO());
    assertEquals(200,bannerDTOResponseEntity.getStatusCode().value());
}

@Test
void getAllAdvertTest() {
    List<AdvertDTO> allAdvert = advertService.getAllList();
    assertNotNull(allAdvert);
    assertThat(allAdvert.size()).isNotZero();
}

@Test
void getAdvertById(){
    Mockito.when(advertRepository.findById(any())).thenReturn(java.util.Optional.of(prepareTestAdvert()));
    AdvertDTO advertDTO = advertService.getById(any());
    assertEquals(advertDTO.getId(),1L);
}



@Test
void update(){
    Mockito.when(advertRepository.save(any())).thenReturn(prepareTestAdvert());
    Mockito.when(advertRepository.findById(any())).thenReturn(java.util.Optional.of(prepareTestAdvert()));
    AdvertDTO advert = advertService.update(prepareTestAdvertDTO());
    assertEquals(advert.getId(), 1L);
}
```

MessageServiceTest;



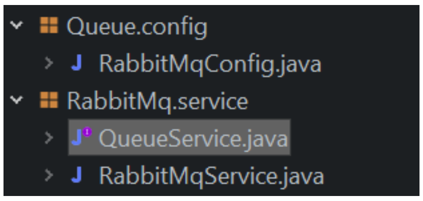
```java
@Test
void getAllMessageTest() {
    List<MessageDTO> allMessage = messageService.getAllList();
    assertNotNull(allMessage);
    assertThat(allMessage.size()).isNotZero();
}

@Test
void create(){
    Mockito.when(messageRepository.save(any())).thenReturn(prepareMessage());
    MessageDTO messageDTO = messageService.create(prepareMessageDTO());
    assertEquals(messageDTO.getId(), 1L);
}

@Test
void getUserById(){
    Mockito.when(messageRepository.findById(any())).thenReturn(java.util.Optional.of(prepareMessage()));
    MessageDTO messageDTO = messageService.getMessageById(any());
    assertEquals(messageDTO.getId(),1L);
}

@Test
void update(){
    Mockito.when(messageRepository.save(any())).thenReturn(prepareMessage());
    Mockito.when(messageRepository.findById(any())).thenReturn(java.util.Optional.of(prepareMessage()));
    MessageDTO messageDTO = messageService.update(prepareMessageDTO());
    assertEquals(messageDTO.getId(), 1L);
}
```

UserServiceTest;

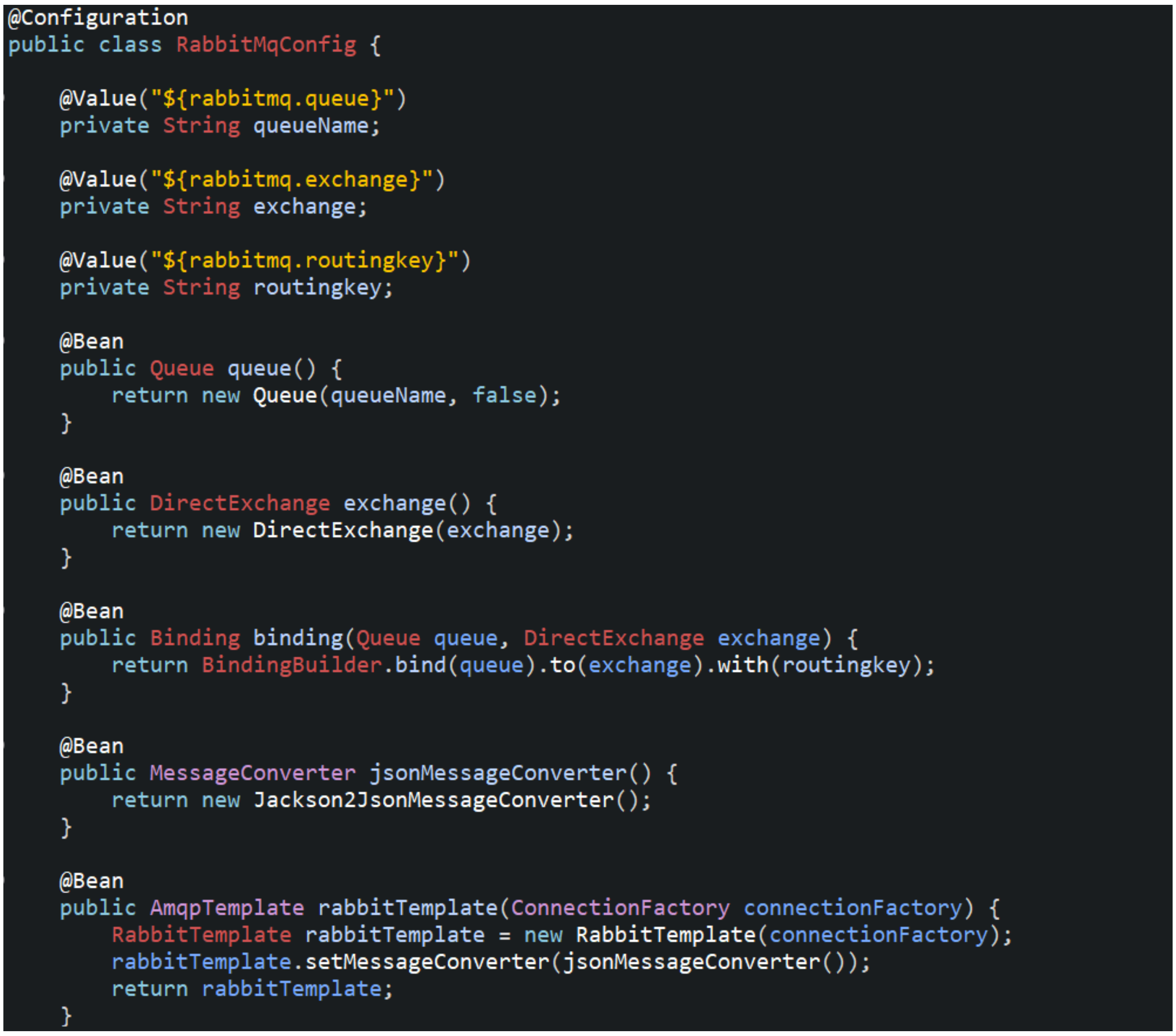


```java
@Test
void getAllUserTest() {
    List<UserDTO> allUser = userService.getAllList();
    assertNotNull(allUser);
    assertThat(allUser.size()).isNotZero();
}

@Test
void create(){
    Mockito.when(userRepository.save(any())).thenReturn(prepareTestUser());
    UserDTO user = userService.create(prepareTestUserDTO());
    assertEquals(user.getId(), 1L);
}

@Test
void getUserById(){
    Mockito.when(userRepository.findById(any())).thenReturn(java.util.Optional.of(prepareTestUser()));
    UserDTO userDTO = userService.getUserById(any());
    assertEquals(userDTO.getId(),1L);
}

@Test
void update(){
    Mockito.when(userRepository.save(any())).thenReturn(prepareTestUser());
    Mockito.when(userRepository.findById(any())).thenReturn(java.util.Optional.of(prepareTestUser()));
    UserDTO user = userService.update(prepareTestUserDTO());
    assertEquals(user.getId(), 1L);
}
```

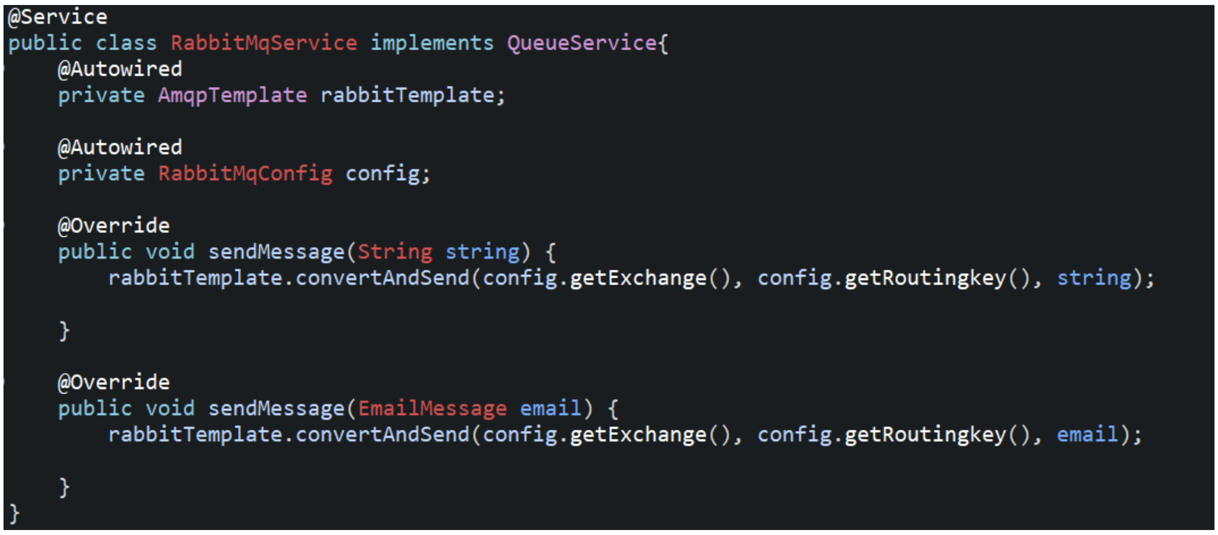
RabbitMq;



```
Queue.config
    RabbitMqConfig.java
RabbitMq.service
    QueueService.java
    RabbitMqService.java
```



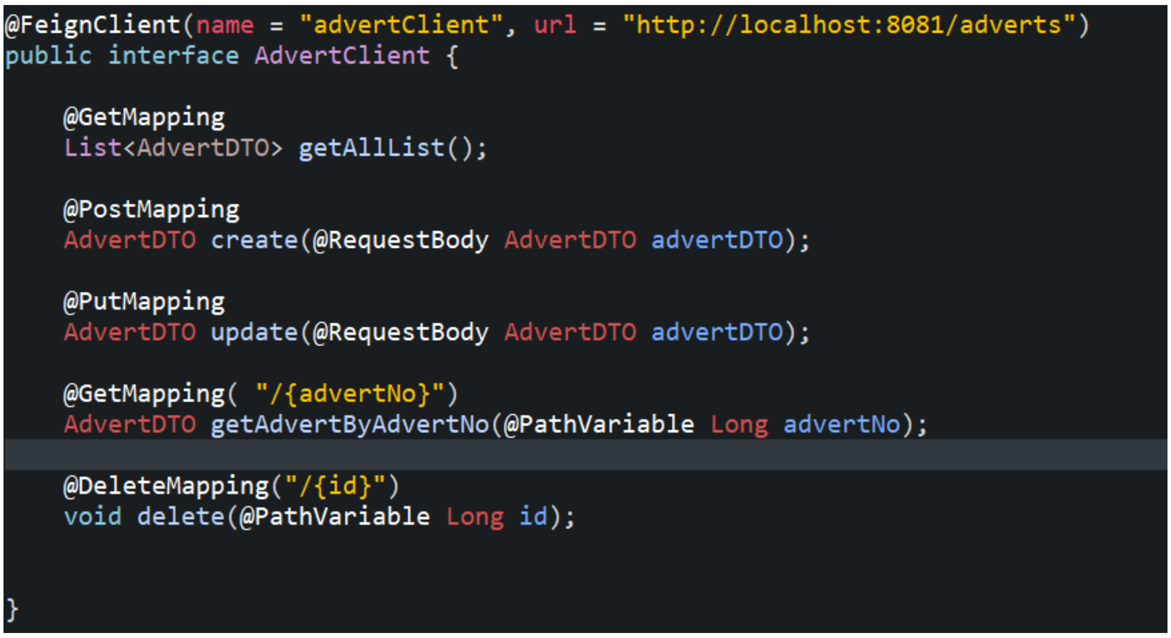
```java
@Configuration
public class RabbitMqConfig {

    @Value("${rabbitmq.queue}")
    private String queueName;

    @Value("${rabbitmq.exchange}")
    private String exchange;

    @Value("${rabbitmq.routingkey}")
    private String routingkey;

    @Bean
    public Queue queue() {
        return new Queue(queueName, false);
    }

    @Bean
    public DirectExchange exchange() {
        return new DirectExchange(exchange);
    }

    @Bean
    public Binding binding(Queue queue, DirectExchange exchange) {
        return BindingBuilder.bind(queue).to(exchange).with(routingkey);
    }

    @Bean
    public MessageConverter jsonMessageConverter() {
        return new Jackson2JsonMessageConverter();
    }

    @Bean
    public AmqpTemplate rabbitTemplate(ConnectionFactory connectionFactory) {
        RabbitTemplate rabbitTemplate = new RabbitTemplate(connectionFactory);
        rabbitTemplate.setMessageConverter(jsonMessageConverter());
        return rabbitTemplate;
    }
```



```java
@Service
public class RabbitMqService implements QueueService{
    @Autowired
    private AmqpTemplate rabbitTemplate;

    @Autowired
    private RabbitMqConfig config;

    @Override
    public void sendMessage(String string) {
        rabbitTemplate.convertAndSend(config.getExchange(), config.getRoutingkey(), string);

    }

    @Override
    public void sendMessage(EmailMessage email) {
        rabbitTemplate.convertAndSend(config.getExchange(), config.getRoutingkey(), email);

    }
}
```

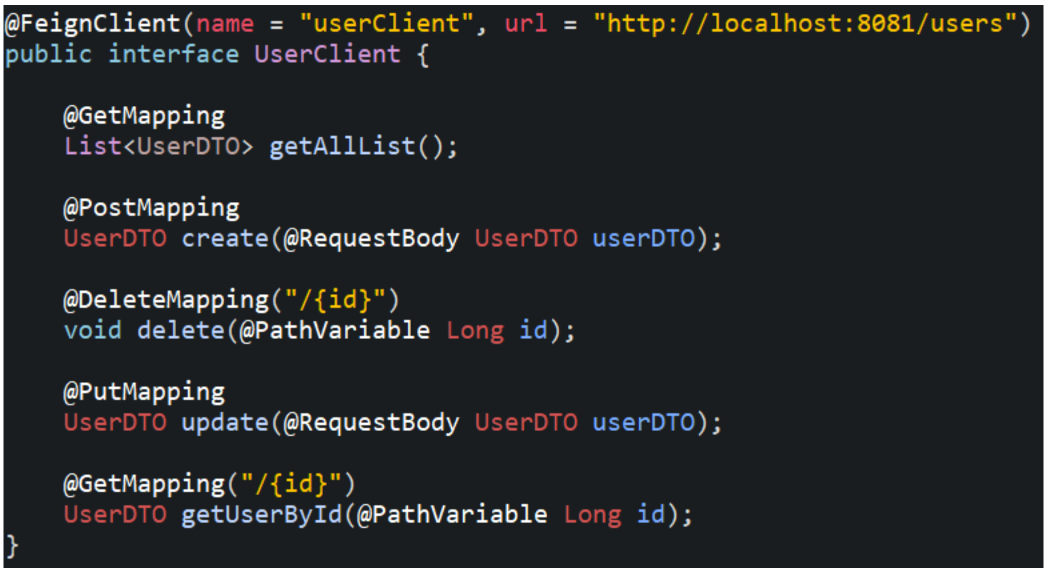
Gateway;

AdvertClient;



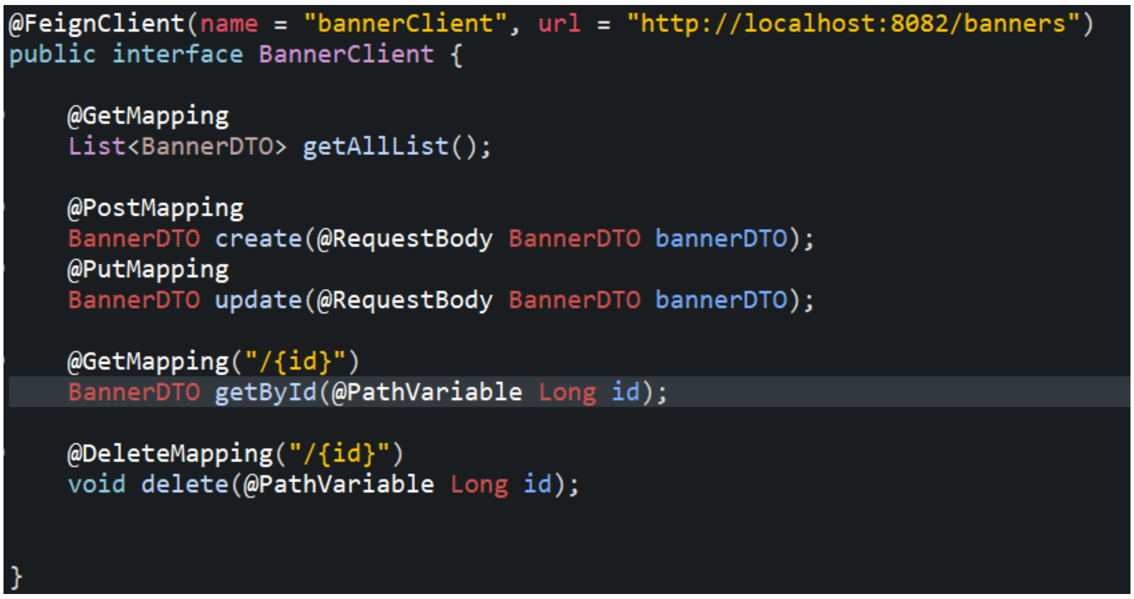
```java
@FeignClient(name = "advertClient", url = "http://localhost:8081/adverts")
public interface AdvertClient {

    @GetMapping
    List<AdvertDTO> getAllList();

    @PostMapping
    AdvertDTO create(@RequestBody AdvertDTO advertDTO);

    @PutMapping
    AdvertDTO update(@RequestBody AdvertDTO advertDTO);

    @GetMapping( "/{advertNo}")
    AdvertDTO getAdvertByAdvertNo(@PathVariable Long advertNo);

    @DeleteMapping("/{id}")
    void delete(@PathVariable Long id);


}
```

UserClient;



```java
@FeignClient(name = "userClient", url = "http://localhost:8081/users")
public interface UserClient {

    @GetMapping
    List<UserDTO> getAllList();

    @PostMapping
    UserDTO create(@RequestBody UserDTO userDTO);

    @DeleteMapping("/{id}")
    void delete(@PathVariable Long id);

    @PutMapping
    UserDTO update(@RequestBody UserDTO userDTO);

    @GetMapping("/{id}")
    UserDTO getUserById(@PathVariable Long id);
}
```

BannerClient;



```java
@FeignClient(name = "bannerClient", url = "http://localhost:8082/banners")
public interface BannerClient {

    @GetMapping
    List<BannerDTO> getAllList();

    @PostMapping
    BannerDTO create(@RequestBody BannerDTO bannerDTO);
    @PutMapping
    BannerDTO update(@RequestBody BannerDTO bannerDTO);

    @GetMapping("/{id}")
    BannerDTO getById(@PathVariable Long id);

    @DeleteMapping("/{id}")
    void delete(@PathVariable Long id);


}
```