Elaboro	Documento	Versión	Descripción	Fecha
Luis Sánchez Martínez	Práctica integración FeignClient.	1.0	Creación de documento	10/04/2024
Luis Sánchez Martínez	Práctica integración FeignClient.	1.1	Se agrega ErrorDecoder y CircuitBreaker en el FeignClient.	13/04/2024

Práctica integración FeignClient

Tiempo estimado: 3 horas.

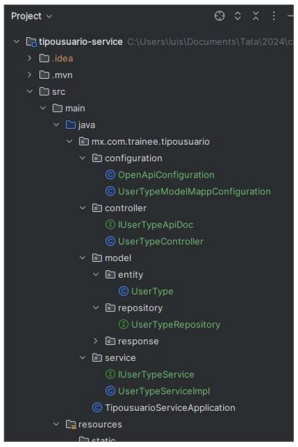
Uso de:

- JDK 17
- Maven
- IntelliJ
- Postman
- Git
- Spring Cloud
    - Resilience4J
    - Actuator
    - FeignClient

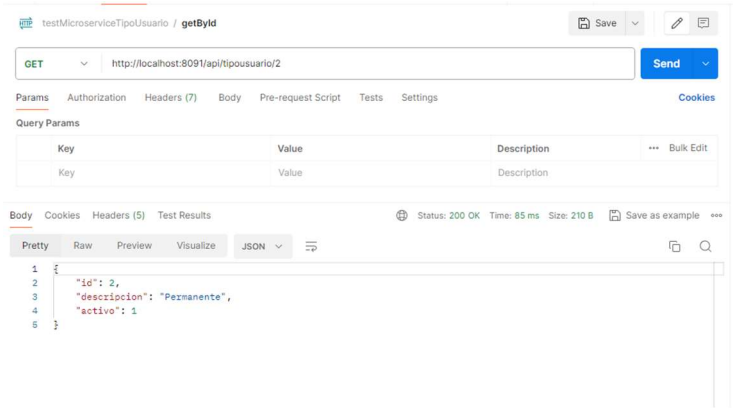
Crear Api y Microservicio realizara el CRUD a la tabla TipoUsuario.

1. Se crea el API para el CRUD de *tipousuario-service*.

2. Se crea el proyecto en *Spring Boot* donde se implementará el microservicio *tipousuario-service*.



3. Levantar el servicio *tipousuario-service* y ejecutar la petición del endpoint GET *getById*.

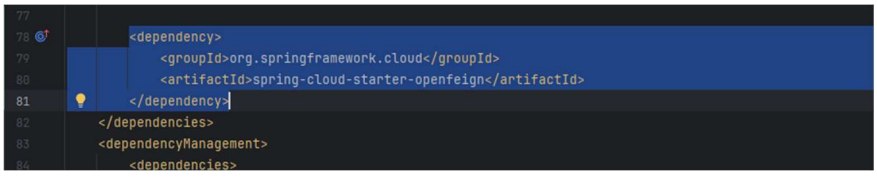


Nota: bajar el proyecto de la ruta en git https://github.com/luis-s-mtz/tipousuario-service.git, bajar de la rama *development*.

Integrar FeignClien en el proyecto de Spring Boot permiso-service.

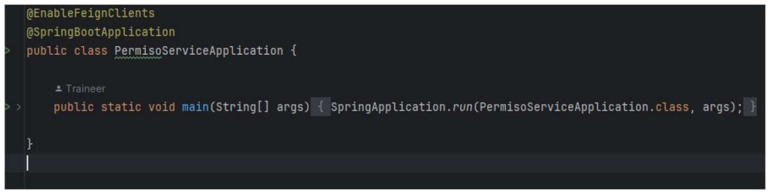
1. Agregar la dependencia *spring-cloud-starter-openfeign* en el pom del proyecto *permiso-service*.

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```



2. Colocar la notación *@EnableFeignClients* en la clase *PermisoServiceApplication*.

```java
@EnableFeignClients
@SpringBootApplication
public class PermisoServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(PermisoServiceApplication.class, args);
    }

}
```



3. Definir una interface, agregarle la notación *@FeignClient* y definir el método para consumir el servicio que mandara a llamar *tipousuario-service*.

```java
@FeignClient(name = "tipousuario-service", path = "/api/tipousuario"
        , url = "localhost:8091")
public interface UserTypeFeignClient {

    @GetMapping("/{id}")
    ResponseEntity<UserTypeDTO> getById(@PathVariable(value = "id")Integer id);
}
```

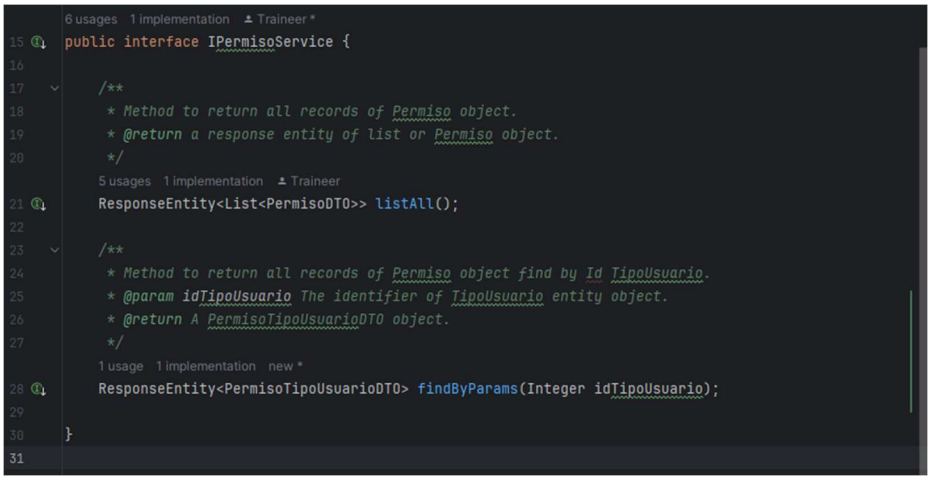
4. Modificar los scripts de creación de esquema e inserción de registros para integrar una tabla relacional que integre los datos del identificador del permiso relacionado con el identificador del tipo de usuario.

```
DROP TABLE IF EXISTS Permiso;
CREATE TABLE Permiso (
id INT AUTO_INCREMENT  PRIMARY KEY,
nombre VARCHAR(25) NOT NULL,
descripcion VARCHAR(50) NOT NULL,
activo INT NOT NULL,
id_Padre INT NOT NULL,
icono VARCHAR(50) NOT NULL
);
DROP TABLE IF EXISTS Tipo_Usuario_Permiso;
CREATE TABLE Tipo_Usuario_Permiso (
id INT AUTO_INCREMENT  PRIMARY KEY,
id_Tipo_Usuario INT NOT NULL,
id_Permiso INT NOT NULL,
activo INT NOT NULL
);
```

```
INSERT INTO Permiso VALUES (1,'Academico', 'Categoria de permisos
academicos', 1, 0, '/images/icon_school.gif');
INSERT INTO Permiso VALUES (2,'Enfermedad', 'Categoria de permisos por
enfermedad', 1, 0, '/images/icon_disease.gif');
INSERT INTO Permiso VALUES (3,'Permiso academ por examen', 'Presentacion de
examenes', 1,  1, '/images/icon_school_quiz.gif');
INSERT INTO Permiso VALUES (4,'Permiso por enfermedad', 'Consulta en
Clinica', 1,  2, '/images/icon_check_health.gif');
INSERT INTO Tipo_Usuario_Permiso VALUES (1,1,4,1);
INSERT INTO Tipo_Usuario_Permiso VALUES (2,2,3,1);
INSERT INTO Tipo_Usuario_Permiso VALUES (3,2,4,1);
```

5. Agregar la lógica en la capa de servicio para mandar a llamar el cliente que se utilizara para el servicio tipousuario-service.

```
/**
 * Method to return all records of Permiso object find by Id TipoUsuario.
 * @param idTipoUsuario The identifier of TipoUsuario entity object.
 * @return A PermisoTipoUsuarioDTO object.
 */
ResponseEntity<PermisoTipoUsuarioDTO> findByParams(Integer idTipoUsuario);
```



```
6 usages  1 implementation  Trainee *
public interface IPermisoService {

    /**
     * Method to return all records of Permiso object.
     * @return a response entity of list or Permiso object.
     */
    5 usages  1 implementation  Trainee
    ResponseEntity<List<PermisoDTO>> listAll();

    /**
     * Method to return all records of Permiso object find by Id TipoUsuario.
     * @param idTipoUsuario The identifier of TipoUsuario entity object.
     * @return A PermisoTipoUsuarioDTO object.
     */
    1 usage  1 implementation  new *
    ResponseEntity<PermisoTipoUsuarioDTO> findByParams(Integer idTipoUsuario);

}
```

```
@Override
public ResponseEntity<PermisoTipoUsuarioDTO> findByParams(Integer idTipoUsuario) {

    PermisoTipoUsuarioDTO permisoTipoUsuarioDTO;
    ResponseEntity<UserTypeDTO> respUserTypeDTO = userTypeFeignClient.getById(idTipoUsuario);

    String tipoUsuario = respUserTypeDTO.getStatusCode().equals(HttpStatus.OK) ?
            respUserTypeDTO.getBody().getDescription():
            null;

    if (tipoUsuario == null){
        log.error("Error in [{}] findByParams(): {}",
                this.getClass().getName(),"Error al consultar informacion de feign client.");
        throw new PermisoSrvInternalServErrorException("Error al consultar informacion de
feign client.");
    } else {
        permisoTipoUsuarioDTO = new PermisoTipoUsuarioDTO();
        permisoTipoUsuarioDTO.setTipoUsuario(tipoUsuario);
        List<TipoUsuarioPermiso> relationList = findPermisoByIdTipoUsuario(idTipoUsuario);

        List<Integer> ids = new ArrayList<>();
        relationList.stream().
                map(tipUsrPerm -> ids.add(tipUsrPerm.getIdPermiso())).
                toList();

        permisoTipoUsuarioDTO.setPermisos(getPermisosByIds(ids));
    }
    return ResponseEntity.ok(permisoTipoUsuarioDTO);
}
```

6. Agregar el endpoint GET *findByParams* en la clase *PermisoController* que permitirá obtener la lista de permisos relacionados con el tipo de usuario.

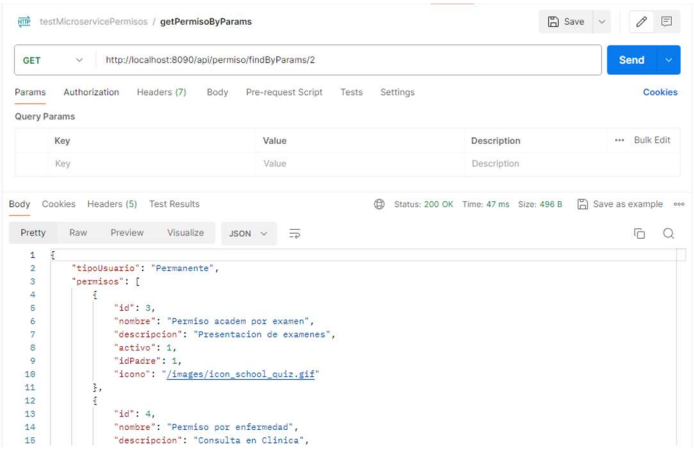
```java
/**
 * Endpoint used to get all records in Permiso entity.
 * @return a ResponseEntity of the list of Permiso object.
 */
1 usage  Traineer
@Override
public ResponseEntity<List<PermisoDTO>> getAll() { return permisoService.listAll(); }

/**
 * Endpoint used to get all records in Permiso entity find by idTipoUsuario.
 *
 * @param idTipoUsuario The identifier of the UserType entity.
 * @return A ResponseEntity of the PermisoTipoUsuarioDTO object.
 */
no usages  new *
@Override
public ResponseEntity<PermisoTipoUsuarioDTO> findByParams(Integer idTipoUsuario) {
    return permisoService.findByParams(idTipoUsuario);
}
}
```

7. Agregar la configuración al Slf4J en el archivo de propiedades.

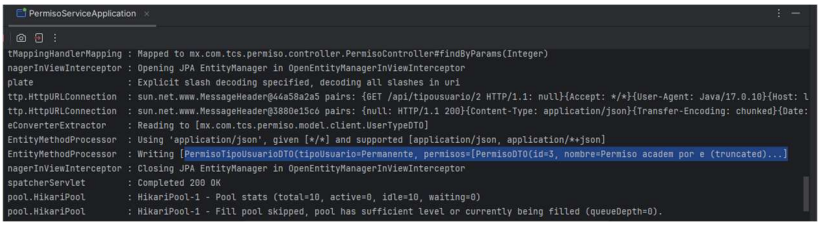
```
logging.level.root=warn
logging.level.org.springframework.web=debug
logging.level.org.hibernate=error
```

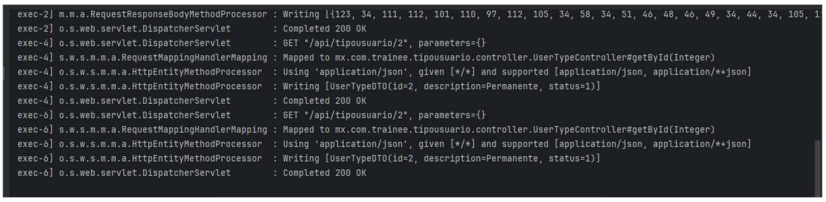
```
management.endpoint.metrics.enabled=true
management.endpoints.web.exposure.include=metrics,info,health
management.endpoint.health.show-components=always
management.endpoint.health.show-details=always

spring.cloud.circuitbreaker.resilience4j.enabled=true

logging.level.root=warn
logging.level.org.springframework.web=debug
logging.level.org.hibernate=error
```

8. Levantar el servicio de *permiso-service* y realizar la prueba ejecutando el endpoint de consulta de lista de la tabla permiso usando el id de TipoUsuario.



9. Revisar los logs para identificar la llamada al servicio tipousuario-service utilizando FeignClient.





10. Integrar la clase *CustomErrorDecoder* y *FeignConfiguration* para recuperar las respuestas desde la solicitud desde *FeignClient* y manejar las excepciones lanzadas 404 y 500 desde el servicio tipousuario-service.

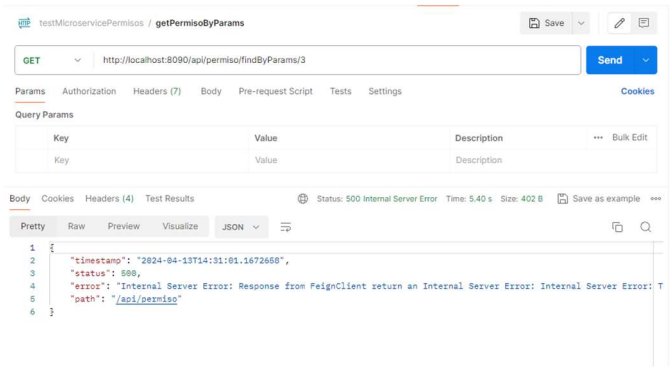
```
public class CustomErrorDecoder implements ErrorDecoder {
    /**
     * Method to manage the errors generated in FeignClient.
     *
     * @param s         The methodKey.
     * @param response The response of the FeignClient.
     * @return The exception throw.
     */
    @Override
    public Exception decode(String s, Response response) {
        ErrorMessageDTO feignClienErrorDTO = deserializeResponse(response);
        String errorMessage = feignClienErrorDTO.getError();

        switch (response.status()) {
            case 400:
                return new BadRequestException();
            case 404:
                return new ItemNotFoundException("Not found in request to FeignClient: " + errorMessage + ".");
            case 500:
                return new PermisoSrvInternalServErrorException(
                        "Response from FeignClient return an Internal Server Error: "+ errorMessage +".");
            default:
                return new PermisoSrvInternalServErrorException("Exception in request from FeignClient: "+
errorMessage +".");
        }
    }

    private ErrorMessageDTO deserializeResponse(Response response) {
        ErrorMessageDTO errorDTO = null;
        try (InputStream bodyIs = response.body().asInputStream()) {
            ObjectMapper mapper = new ObjectMapper();
            errorDTO = mapper.readValue(bodyIs, ErrorMessageDTO.class);
        } catch (IOException e) {
            throw new PermisoSrvInternalServErrorException("Error when deserialize response from FeignClient: "+
e.getMessage());
        }
        return errorDTO;
    }
}
```

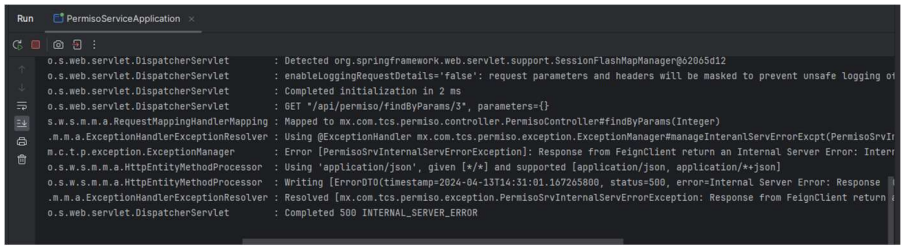
```java
@Slf4j
public class CustomErrorDecoder implements ErrorDecoder {
    /**
     * Method to manage the errors generated in FeignClient.
     *
     * @param s        The methodKey.
     * @param response The response of the FeignClient.
     * @return The exception throw.
     */
    Traineer
    @Override
    public Exception decode(String s, Response response) {
        ErrorMessageDTO feignClienErrorDTO = deserializeResponse(response);
        String errorMessage = feignClienErrorDTO.getError();

        switch (response.status()) {
            case 400:
                return new BadRequestException();
            case 404:
                return new ItemNotFoundException("Not found in request to FeignClient: " + errorMessage + ".");
            case 500:
                return new PermisoSrvInternalServErrorException(
                        "Response from FeignClient return an Internal Server Error: "+ errorMessage +".");
            default:
                return new PermisoSrvInternalServErrorException("Exception in request from FeignClient: "+ errorMessage +".");
        }
```

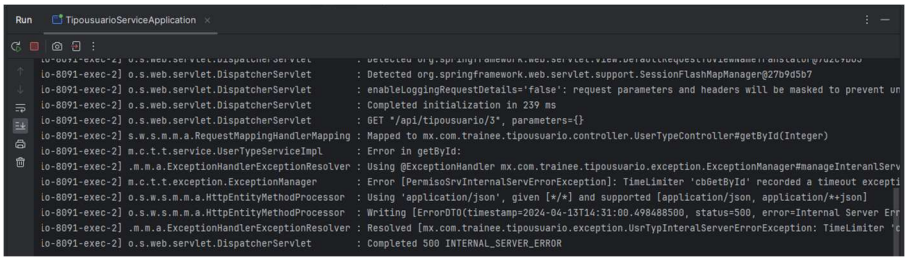
```java
public class FeignConfiguration {

    /**
     * Method to create the Bean of ErrorDecoder.
     * @return An ErrorDecoder Bean.
     */
    @Bean
    ErrorDecoder errorDecoder() {
        return new CustomErrorDecoder();
    }

}
```

```java
/**
 * @author luis
 * @since 1.0
 * Configuration class to create an ErrorDecoder Bean using the CustomerErrorDecoder.
 */
2 usages  Traineer
@Configuration
public class FeignConfiguration {

    /**
     * Method to create the Bean of ErrorDecoder.
     * @return An ErrorDecoder Bean.
     */
    no usages  Traineer
    @Bean
    ErrorDecoder errorDecoder() { return new CustomErrorDecoder(); }

}
```

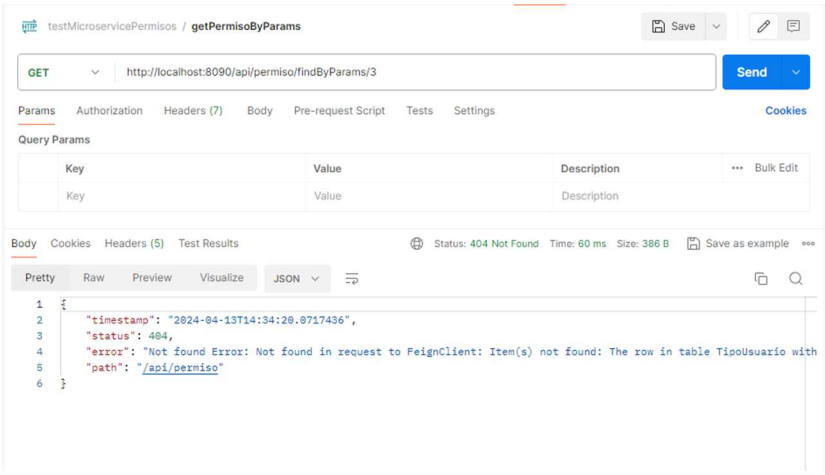
11. Ejecutar la validación del error 500 lanzado desde el cliente tipousuario-service.

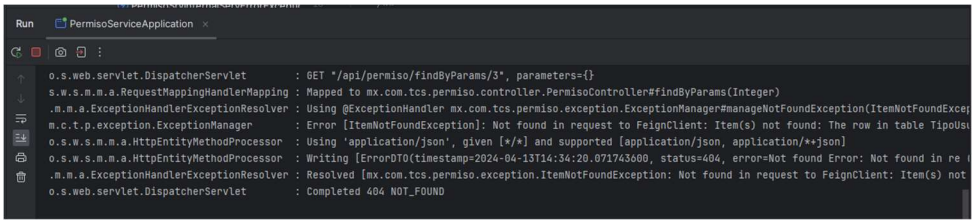






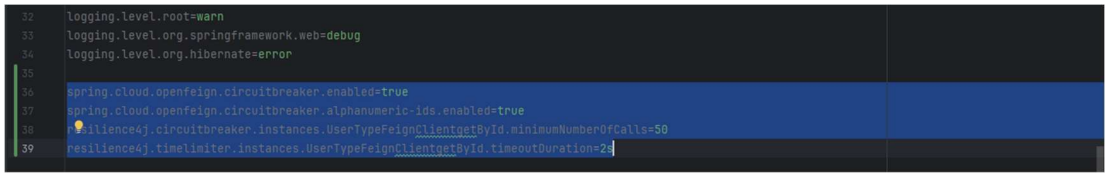
12. Realizar pruebas para validar el manejo de ErrorDecoder con un error 404 recuperado desde el FeignClient.





13. Integrar la configuración en el archivo de propiedades para el uso de CircuitBreaker en el cliente de UserTypeFeignClient.

```
spring.cloud.openfeign.circuitbreaker.enabled=true
spring.cloud.openfeign.circuitbreaker.alphanumeric-ids.enabled=true
resilience4j.circuitbreaker.instances.UserTypeFeignClientgetById.minimumNumberOfCalls=50
resilience4j.timelimiter.instances.UserTypeFeignClientgetById.timeoutDuration=2s
```

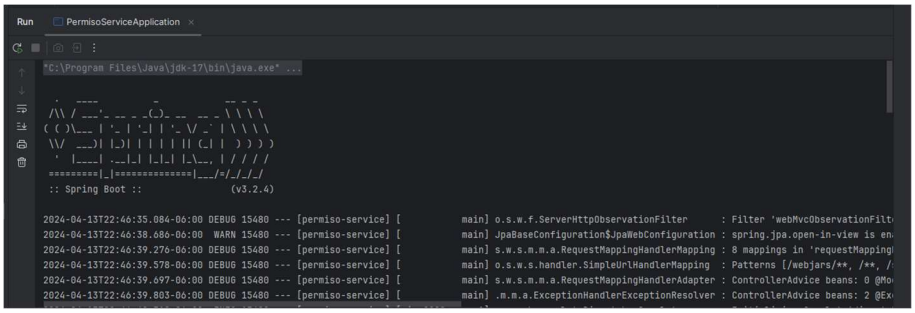


14. Agregar la clase UserTypeFallback para el manejo de la petición en caso de que el circuito sea abierto y no se obtenga respuesta del cliente.

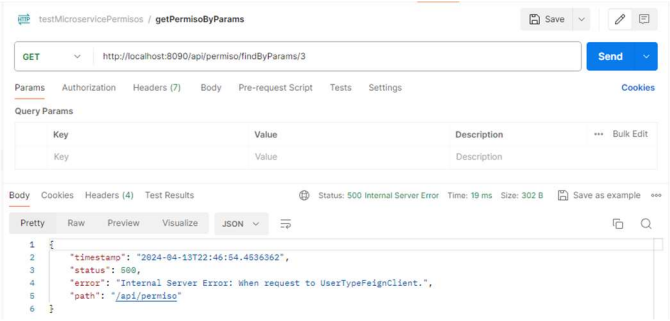
```
@Component
public class UserTypeFallback implements UserTypeFeignClient{
    /**
     * Method used to get UserType record find by Id.
     * @param id Identifier of UserType objet to find.
     * @return An ResponseEntity object of UserTypeDTO.
     */
    @Override
    public ResponseEntity<UserTypeDTO> getById(Integer id) {
        throw new PermisoSrvInternalServErrorException("When request to UserTypeFeignClient.");
    }
}
```

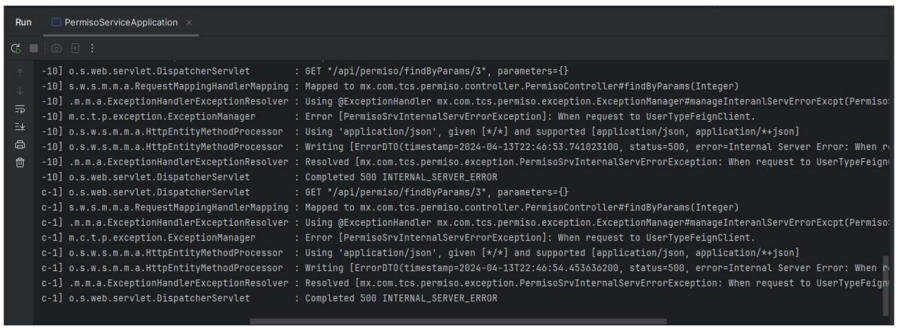
```java
/**
 * @author luis
 * @since 1.0
 *
 * Component class to implements fallback when UserTypeFeignClient use the CircuitBreaker
 * pattern.
 */
1 usage  new *
@Component
public class UserTypeFallback implements UserTypeFeignClient{
    /**
     * Method used to get UserType record find by Id.
     * @param id Identifier of UserType objet to find.
     * @return An ResponseEntity object of UserTypeDTO.
     */
    1 usage  new *
    @Override
    public ResponseEntity<UserTypeDTO> getById(Integer id) {
        throw new PermisoSrvInternalServErrorException("When request to UserTypeFeignClient.");
    }
}
```

15. Levantar el servicio permiso-service para validar la implementación de CircuitBreaker en el cliente de UserTypeFeignClient.



16. Ejecutar peticiones a la función GET /api/permiso/findByParam/{id} con el servicio tipousuario-service apagado para validar el funcionamiento del circuito abierto y la ejecución de la clase Fallback.





Referencias

https://docs.spring.io/spring-cloud-openfeign/docs/current/reference/html/

https://barcochrist.medium.com/comunicaci%C3%B3n-entre-dos-micro-servicios-con-feign-y-eureka-ab4f3daf70c6

https://cloud.spring.io/spring-cloud-netflix/multi/multi_spring-cloud-feign.html

https://docs.spring.io/spring-boot/docs/2.1.13.RELEASE/reference/html/boot-features-logging.html

https://www.baeldung.com/spring-boot-logging

https://www.baeldung.com/java-feign-client-exception-handling

https://docs.spring.io/spring-cloud-openfeign/docs/current/reference/html/#spring-cloud-feign-overriding-defaults