***CHURRASCO***

**Código de recuperación de Sendgrid (Lucas):** 5G9BSC5GBLHG85QWHMLDGMB8

# **1 ENTORNO DEL NAVEGADOR Y ESPECIFICACIONES**

**API KEY (Sendgrid Lucas):** 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

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzM4NCJ9.eyJzdWIiOiI4NmNjN2M5My04NWEzLTQzZTgtOGI1Ny0yYmQ2ZWQ4MGI4ZDEiLCJpYXQiOjE3NDAzMzQzMjksImV4cCI6MTc0MDMzNDM4OX0.ELoUFE1eK-fRfvB-LYfZKohgBjwTfNM6c5YoiDQJY8cH98Vg4Aayvr\_bUJ8\_vM5Q

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzM4NCJ9.eyJzdWIiOiJmOTJjOWM0Ni02MGIxLTQ2YTItYTZjOS0xMGZlMmM5MDE1MGYiLCJpYXQiOjE3NDAzMzU1NDUsImV4cCI6MTc0MDMzNTYwNX0.5Ir90N-eEQAlN7zR4AwJQkrXRZLDHnF1IsM98CWnUXVz7UjpO26AVn8d17lIgBnu

{

"username": "Khin07",

"password": "K",

"email": "falla.urluc24@triana.salesianos.edu",

"nombre": "Khindasvinto",

"apellidos": "Khin Khin",

"sexo": "HOMBRE",

"numTelefono": "636889910",

"fechaNacimiento": "2005-03-27T00:00:00",

"lugarNacimiento": "TARRAGONA",

"lugarResidencia": "TARRAGONA"

}

[**http://localhost:8080/tag/?page=0&size=5**](http://localhost:8080/tag/?page=0&size=5)

**http://localhost:8080/tag/?sort=id**

public GetUserDTO(UUID id, String username, String email, String nombre, String apellidos, Sexo sexo,

String numTelefono, String imagenPerfil, LocalDate fechaNacimiento, Provincia lugarNacimiento,

Provincia lugarResidencia, Long puntosPopularidad, String idiomaNativo, String otrosIdiomas,

String conocimientos){

this(id, username, email, nombre, apellidos, sexo, numTelefono, imagenPerfil, fechaNacimiento,

lugarNacimiento, lugarResidencia, puntosPopularidad, idiomaNativo, otrosIdiomas,

conocimientos, null);

}

Para manejar todas las excepciones como `NotFoundException`, puedes crear una jerarquía de excepciones personalizadas y un \*\*manejador global de excepciones\*\* usando `@ControllerAdvice`.

---

## \*\*1️⃣ Crear la Excepción Base (`NotFoundException`)\*\*

Todas las excepciones específicas heredarán de esta.

```java

package com.example.Recysell.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT\_FOUND)

public class NotFoundException extends RuntimeException {

public NotFoundException(String message) {

super(message);

}

}

```

---

## \*\*2️⃣ Crear Excepciones Específicas (Opcional)\*\*

Si quieres personalizar, puedes hacer excepciones que extiendan `NotFoundException`:

```java

package com.example.Recysell.exception;

public class TrabajadorNotFoundException extends NotFoundException {

public TrabajadorNotFoundException(Long id) {

super("Trabajador con ID " + id + " no encontrado.");

}

}

```

```java

package com.example.Recysell.exception;

public class UserNotFoundException extends NotFoundException {

public UserNotFoundException(String username) {

super("Usuario con username '" + username + "' no encontrado.");

}

}

```

---

## \*\*3️⃣ Crear un Manejador Global con `@ControllerAdvice`\*\*

Este método maneja \*\*todas las excepciones de tipo `NotFoundException`\*\*, y permite personalizar con `instanceof` y pattern matching.

```java

package com.example.Recysell.exception.handler;

import com.example.Recysell.exception.NotFoundException;

import com.example.Recysell.exception.TrabajadorNotFoundException;

import com.example.Recysell.exception.UserNotFoundException;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.Map;

@RestControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(NotFoundException.class)

public ResponseEntity<Map<String, Object>> handleNotFoundException(NotFoundException ex) {

HttpStatus status = HttpStatus.NOT\_FOUND;

String errorType = "Recurso no encontrado";

// Personalizar mensajes usando pattern matching

if (ex instanceof TrabajadorNotFoundException trabajadorEx) {

errorType = "Trabajador no encontrado";

} else if (ex instanceof UserNotFoundException userEx) {

errorType = "Usuario no encontrado";

}

Map<String, Object> errorBody = Map.of(

"error", errorType,

"message", ex.getMessage(),

"status", status.value()

);

return ResponseEntity.status(status).body(errorBody);

}

}

```

---

## \*\*4️⃣ Lanzar Excepciones en el Servicio\*\*

Ejemplo en el servicio de trabajadores:

```java

public Trabajador getTrabajadorById(Long id) {

return trabajadorRepository.findById(id)

.orElseThrow(() -> new TrabajadorNotFoundException(id));

}

```

---

### ✅ \*\*Ventajas de este Enfoque\*\*

✔ \*\*Código más limpio\*\*: Se maneja todo en un solo lugar.

✔ \*\*Personalización opcional\*\*: Con `instanceof`, puedes diferenciar excepciones si es necesario.

✔ \*\*Menos código repetitivo\*\*: No necesitas manejar `NotFoundException` en cada controlador.

Este patrón mejora la mantenibilidad del código y es ideal para aplicaciones grandes. 🚀