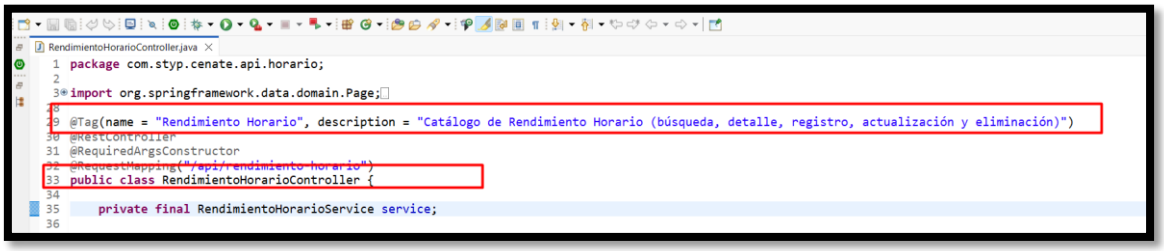
Documentación SWAGGER

Enlace

http://localhost:8080/swagger-ui/index.html#/

Documentación de Controladores

- **Anotación @Tag**



A nivel de clase controladora se utiliza la anotación @Tag, la cual permite agrupar y describir de manera semántica los endpoints relacionados dentro de la documentación Swagger.

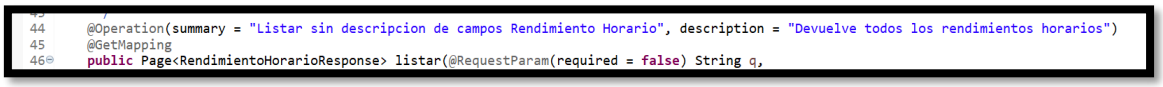
Esta anotación cuenta con los siguientes atributos:

- name: Define el nombre del grupo o categoría del controlador.
- description: Proporciona una descripción funcional del conjunto de endpoints asociados.

El uso de @Tag mejora la legibilidad, organización y navegabilidad de la documentación, especialmente en APIs con múltiples controladores.

Documentación de métodos (END_POINT)

- **Anotación @Operation**



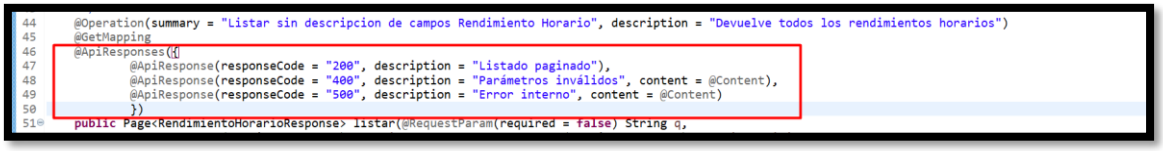
A nivel de método se utiliza la anotación @Operation, la cual describe el comportamiento específico de cada endpoint expuesto por el controlador.

Esta anotación incluye los siguientes atributos:

- summary: Resume de forma breve la funcionalidad del endpoint.
- description: Describe de manera detallada el propósito, alcance y comportamiento del servicio.

El uso adecuado de @Operation facilita la comprensión funcional del endpoint tanto para desarrolladores como para consumidores de la API.

- **Anotación @ApiResponses**



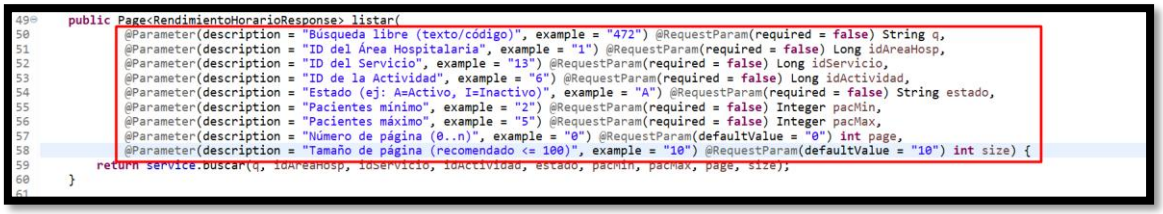
La anotación @ApiResponses permite documentar los posibles códigos de respuesta HTTP que puede retornar un endpoint, junto con su respectiva descripción.

Cada respuesta se define mediante la anotación @ApiResponse, la cual incluye:

- responseCode: Código HTTP devuelto por el servicio (por ejemplo, 200, 400, 404, 500).
- description: Descripción del significado de la respuesta.
- content (opcional): Tipo de contenido y estructura del objeto retornado.

El uso de @ApiResponses garantiza una documentación clara de los escenarios de éxito y error, facilitando la integración y el manejo adecuado de respuestas por parte del cliente.

- **Anotación @Parameter**



La anotación @Parameter se utiliza para documentar los parámetros de entrada que recibe un endpoint, tales como parámetros de ruta (@PathVariable), parámetros de consulta (@RequestParam) o encabezados HTTP (@RequestHeader).

Esta anotación permite describir de manera explícita el propósito y las características de cada parámetro, mejorando la claridad y precisión de la documentación generada en Swagger UI.

Los principales atributos de la anotación @Parameter son:

- name: Nombre del parámetro tal como es expuesto por el endpoint.
- description: Descripción funcional del parámetro y su finalidad dentro del servicio.
- required: Indica si el parámetro es obligatorio (true) u opcional (false).
- example (opcional): Proporciona un valor de ejemplo que facilita la comprensión de su uso.

El uso adecuado de @Parameter contribuye a una mejor experiencia para los consumidores de la API, ya que permite conocer claramente qué información debe ser enviada, en qué formato y en qué condiciones.

- **Anotación @RequestBody**

```
@PostMapping
public ResponseEntity<RendimientoHorarioResponse> crear(
        @io.swagger.v3.oas.annotations.parameters.RequestBody(
                description = "Datos para crear rendimiento horario",
                required = true,
                content = @Content(schema = @Schema(implementation = RendimientoHorarioRequest.class)))
        @RequestBody RendimientoHorarioRequest req) {
    RendimientoHorarioResponse created = service.crear(req);
    return ResponseEntity.status(HttpStatus.CREATED).body(created);
}
```

La anotación @RequestBody se utiliza para documentar los datos que el cliente envía en el cuerpo de la solicitud HTTP, comúnmente en formato JSON, cuando se realizan operaciones como creación o actualización de recursos.

En la documentación Swagger, esta anotación permite describir la estructura del objeto recibido, así como el tipo de contenido y ejemplos de uso, facilitando la correcta comprensión del contrato del servicio.

Los principales atributos de la anotación @RequestBody son:

- description: Describe el propósito del objeto enviado en el cuerpo de la solicitud.
- required: Indica si el cuerpo de la petición es obligatorio.
- content: Define el tipo de contenido (mediaType) y el esquema del objeto esperado.

El uso de @RequestBody permite documentar de forma clara la estructura de los datos de entrada, garantizando una correcta validación y consumo del endpoint por parte de las aplicaciones cliente.

```
}

@Operation(summary = "Actualizar por id de rendimiento de horario",
           description = """
                   Actualiza el registro existente. Debe existir el ID y los catálogos relacionados.
                   """)
@PutMapping("/{id}")
public RendimientoHorarioResponse actualizar(@PathVariable Long id, @RequestBody RendimientoHorarioRequest req) {
    return service.actualizar(id, req);
}

@Operation(summary = "Eliminar rendimiento horario por ID", description = """
        Elimina el registro por ID.
        Retorna **204 No Content** si se eliminó correctamente.
        """)
@ApiResponses({ @ApiResponse(responseCode = "204", description = "Eliminado correctamente"),
        @ApiResponse(responseCode = "404", description = "No existe el ID solicitado", content = @Content),
        @ApiResponse(responseCode = "409", description = "Conflicto por integridad referencial (está en uso)", content = @Content),
        @ApiResponse(responseCode = "500", description = "Error interno", content = @Content) })
@DeleteMapping("/{id}")
public ResponseEntity<Void> eliminar(
        @Parameter(description = "ID del rendimiento horario", example = "10") @PathVariable Long id) {
    service.eliminar(id);
    return ResponseEntity.noContent().build();
}
```

En caso consideres agregar más texto a la description considera el uso de “”” (3 comillas dobles) al inicio y al final.

Documentación de DTOs

- **Anotación @Schema**

```
@Schema(name = "RendimientoHorarioRequest", description = "Request para crear un rendimiento horario")
public record RendimientoHorarioRequest(
        @Schema(description = "ID Área Hospitalaria", example = "1", requiredMode = Schema.RequiredMode.REQUIRED) Long idAreaHosp,
        @Schema(description = "ID Servicio", example = "13", requiredMode = Schema.RequiredMode.REQUIRED) Long idServicio,
        @Schema(description = "ID Actividad", example = "6", requiredMode = Schema.RequiredMode.REQUIRED) Long idActividad,
        @Schema(description = "ID de la Subactividad (opcional)", example = "3", nullable = true) Long idSubactividad,
        @Schema(description = "Pacientes por hora", example = "4", minimum = "0") Integer pacientesPorHora,
        @Schema(description = "Cantidad adicional de pacientes", example = "1", minimum = "0") Integer adicional,
        @Schema(description = "Estado", example = "A", allowableValues = {
                "A", "I" }) String estado){
}
```

La anotación @Schema se utiliza para documentar los modelos de datos (DTO) que forman parte de las solicitudes y respuestas de los endpoints expuestos por la API. Esta anotación permite describir de manera estructurada cada atributo del objeto, incluyendo su finalidad, restricciones y ejemplos de uso.

En el sistema, el DTO RendimientoHorarioRequest representa el objeto utilizado para la creación de un registro de rendimiento horario, y se encuentra documentado mediante anotaciones @Schema tanto a nivel de clase como de atributo.

Documentación a nivel de clase

A nivel de clase, la anotación @Schema permite definir un nombre representativo y una descripción general del propósito del objeto dentro del sistema.

- name: Nombre del esquema que se mostrará en Swagger UI.
- description: Describe la finalidad del objeto de solicitud.

Esto facilita la identificación del modelo dentro de la sección *Schemas* de Swagger UI.

Documentación a nivel de atributos

Cada atributo del DTO se documenta individualmente utilizando la anotación @Schema, especificando:

- description: Explica la finalidad del campo.
- example: Proporciona un valor de ejemplo para facilitar su comprensión.
- requiredMode: Indica si el campo es obligatorio u opcional.
- nullable: Define si el campo acepta valores nulos.
- minimum: Establece restricciones numéricas mínimas.
- allowableValues: Define los valores permitidos para campos con dominio controlado.