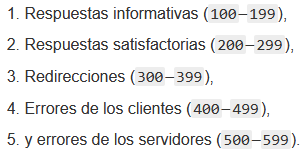
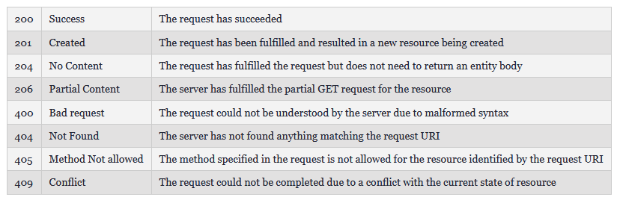
MANEJO DE ERRORES Y NEGOCIACION DE CONTENIDO

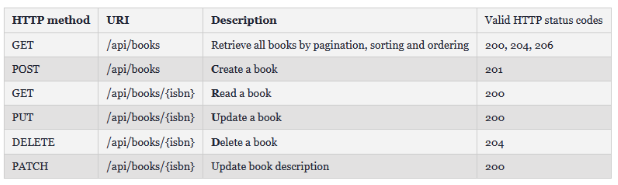
En el protocolo HTTP al recibirse una respuesta, esta siempre viene acompañada de un código. Este código sirve para saber que la petición ha terminado y para conocer el estado de esta, ya que el hecho de recibir una respuesta no quiere decir que se haya procesado de forma satisfactoria. Los códigos de respuesta HTTP se agrupan en 5 categorías, dentro de las cuales existe un número específico para cada tipo de respuesta.

Mas información sobre los códigos: https://www.restapitutorial.com/httpstatuscodes.html





También, dependiendo del código del método HTTP utilizado, hay algunos códigos de respuesta que normalmente se usan para indicar que la petición se procesó de forma correcta; por ejemplo, si estuviéramos creando un API o servicios para el manejo de un catálogo de libros usamos los siguientes verbos y códigos de respuesta HTTP:



En Spring, se tiene una enumeración que contiene una constante por cada uno de los códigos HTTP, [HttpStatus](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/http/HttpStatus.html). Además, cuenta con la anotación @ResponseStatus que permite indicar qué código se regresará cuando el método se ejecute correctamente, incluyendo los manejadores de errores. Podemos utilizar la anotación @ResponseStatus para establecer el estado y la razón en nuestra respuesta HTTP.

A partir de su versión 5, Spring proporciona la clase ResponseStatusException, que permite regresar diferentes códigos de respuesta, en un mismo método, dependiendo del error que haya ocurrido. De esta forma podemos indicar en tiempo de ejecución el estatus de nuestra petición.

Spring proporciona 3 constructores para generar ResponseStatusException:

* ResponseStatusException(HttpStatus status)
* ResponseStatusException(HttpStatus status, java.lang.String reason)
* ResponseStatusException(HttpStatus status, java.lang.String reason, java.lang.Throwable cause)

ResponseStatusException, argumentos del constructor:

* status - un estado HTTP establecido para la respuesta HTTP
* reason - un mensaje que explica la excepción establecida en la respuesta HTTP
* cause - una causa Throwable de la ResponseStatusException

Nota: en Spring, HandlerExceptionResolver intercepta y procesa cualquier excepción lanzada y no manejada por un Controlador.

También proporciona la clase base ResponseEntityExceptionHandler que maneja muchos tipos comunes de excepciones, por lo que podemos usarla como una buena base para nuestros manejadores globales de excepciones (las clases anotadas con @RestControllerAdvice).

Además de realizar un buen manejo de errores en nuestros controladores, otra buena práctica consiste en permitir que el usuario de nuestra API decida el tipo de contenido en el que desea recibir la respuesta (usualmente seleccionando entre JSON y XML).

Para ello implementamos la negociación de contenidos con:

* (Deprecated) Using URL suffixes (extensions) in the request (eg .xml/.json)
* Using URL parameter in the request (eg ?format=json)
* Using Accept header in the request