**[APIRest](https://www.redhat.com/es/topics/api/what-is-a-rest-api)**

Las API son conjuntos de definiciones y protocolos que se utilizan para diseñar e integrar el software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta)

Las API permiten interactuar con un sistema para obtener datos o ejecutar una función, de manera que el sistema comprenda la solicitud y la cumpla.

REST no es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura. Los desarrolladores de las API pueden implementarlo de distintas maneras.

Cuando el cliente envía una solicitud a través de una API de RESTful, esta transfiere una representación del estado del recurso requerido a quien lo haya solicitado o al extremo. La información se entrega por medio de HTTP en uno de estos formatos: JSON (JavaScript Object Notation), HTML, XLT, Python, PHP o texto sin formato.

La comunicación se suele realizar a través de [HTTP](https://developer.mozilla.org/es/docs/Web/HTTP/Methods)

**SpringBoot**

**Al desarrollar fundamentalmente existen tres pasos a realizar**. El primero es crear un proyecto Maven/Gradle y descargar las dependencias necesarias. En segundo lugar, desarrollamos la aplicación y en tercer lugar la desplegamos en un servidor. Si nos ponemos a pensar un poco a detalle en el tema, **únicamente el paso dos es una tarea de desarrollo**. Los otros pasos están más orientados a infraestructura. No deberíamos tener que estar eligiendo continuamente las dependencias y el servidor de despliegue.

SpringBoot nace con la intención de simplificar **los pasos 1 y 3** y que nos podamos centrar en el desarrollo de nuestra aplicación.

A la hora de utilizar SpringBoot trabajaremos con anotaciones que nos simplificaran mucho el trabajo:

@SpringBootApplication

Utilizamos esta anotación en nuestra Aplicación o clase Main para permitir la carga de configuración y el escaneo de componentes de spring

@Autowired

Es una de las anotaciones más habituales cuando trabajamos con Spring Framework ya que se trata de la anotación que permite inyectar unas dependencias con otras dentro de Spring.

@RestController

Se utiliza para convertir una clase en un controlador RESTFul

@GetMapping

@PostMapping

@PutMapping

@DeleteMapping

Utilizamos estas anotaciones sobre los métodos de la clase REST para que se encarguen de la petición correspondiente.

Ej: @GetMapping(“/documentos”) controlara las peticiones de tipo GET a /documentos

@RequestBody

Indicamos esta anotación en los parámetros del método para indicar que el atributo que la siga será el body de la petición.

@PathVariable(“variable”)

Indicamos esta anotación en los parámetros del método para indicar que el atributo que la siga se corresponderá con la variable URI que indiquemos

Ej:

@GetMapping(“documentos/{id}”)

public Documento getDocumentoPorId(@PathVariable(“id”) int id)

Ahora que hemos visto que es springBoot y las anotaciones que necesitamos vamos a crear un nuevo proyecto utilizando el wizard de spring <https://start.spring.io/>

[**Patrón Controlador-Servicio-Repositorio**](https://tom-collings.medium.com/controller-service-repository-16e29a4684e5)

La capa **Controlador** es la capa exterior y es responsable de exponer la funcionalidad para que pueda ser consumida por entidades externas (incluyendo si los hubiera los componentes de UI).

La capa de **Repositorio** es la capa más profunda y se encarga de almacenar, modificar y recuperar los datos.

La capa de **Servicio** es en la que se encontrará toda la lógica de negocio. Si la lógica de negocio requiere buscar/almacenar datos, utilizara entonces la capa de repositorio.

Cuando alguien quiere acceder a está lógica de negocio lo hará a través del Controlador.