

AMS Bootcamp Fullstack Developer.

**REST y SpringREST**



Confidencial

Índice

[1. Personal Notebook 4](#_Toc129343696)

[1.1 Datos alumno 4](#_Toc129343697)

[1.2 Personal Notebook 4](#_Toc129343698)

[2. REST, Spring REST ( y SpringBoot) 5](#_Toc129343699)

[2.1 Objetivos de aprendizaje 5](#_Toc129343700)

[2.2 ¿Qué es MVC? ¿Cómo funciona (teórico)? 6](#_Toc129343701)

[2.2.1 Objetivos 6](#_Toc129343702)

[2.2.2 Recursos formativos 6](#_Toc129343703)

[2.2.3 Posibles preguntas 6](#_Toc129343704)

[2.2.4 Tareas 6](#_Toc129343705)

[2.2.5 Dudas 6](#_Toc129343706)

[2.3 REST 7](#_Toc129343707)

[2.3.1 Objetivos 7](#_Toc129343708)

[2.3.2 Recursos formativos 7](#_Toc129343709)

[2.3.3 Posibles preguntas 7](#_Toc129343710)

[2.3.4 Tareas 8](#_Toc129343711)

[2.3.5 Dudas 8](#_Toc129343712)

[2.4 Spring Boot 9](#_Toc129343713)

[2.4.1 Objetivos 9](#_Toc129343714)

[2.4.2 Recursos formativos 9](#_Toc129343715)

[2.4.3 Posibles preguntas 9](#_Toc129343716)

[2.4.4 Tareas 9](#_Toc129343717)

[2.4.5 Dudas 9](#_Toc129343718)

[2.5 Spring REST 10](#_Toc129343719)

[2.5.1 Objetivos 10](#_Toc129343720)

[2.5.2 Recursos formativos 10](#_Toc129343721)

[2.5.3 Posibles preguntas 10](#_Toc129343722)

[2.5.4 Tareas 11](#_Toc129343723)

[2.5.5 Dudas 11](#_Toc129343724)

# Personal Notebook

## Datos alumno

* DAS
* Nombre Apellidos
* email
* URL de GitHub corporativo

## Personal Notebook

Este es tu cuaderno de trabajo. En él tendrás que registrar tu aprendizaje, tus avances y tus dudas. ¡Vas a generar tu propio material!

Puede ser una gran herramienta si la utilizas correctamente.

El cuaderno estará compartido con los tutores y el resto de colaboradores para ver tu progreso y ayudarte.

Al final del Bootcamp tendrás que entregarlo como evidencia de tu aprendizaje.



Instrucciones de uso:

El fichero está ubicado en tu carpeta del espacio que tenemos en Teams.

Siempre deberá ser editado en modo online y/o con guardado automático en la nube.

Responde a las preguntas con la información que necesites para dar una respuesta cualificada al formador o a tus compañeros. No es necesario copiar texto solo por “rellenar”. Cita las Fuentes en caso de utilizar otros recursos.

# REST, Spring REST ( y SpringBoot)

## Objetivos de aprendizaje

* Objetivo:
  + **Comprender el funcionamiento de las aplicaciones basadas en REST, las diferencias con arquitecturas web clásicas, los principales métodos y herramientas para trabajar con servicios REST en Spring.**
  + **Conocer las ventajas de usar Spring Boot y su configuración inicial**
* Conceptos principales a adquirir:
  + Arquitectura MCV y REST
  + Spring Boot
  + Spring REST

Lecciones:

1. MVC (teoría)
2. REST (teoría)
3. Sprint Boot
4. Spring REST

## ¿Qué es MVC? ¿Cómo funciona (teórico)?

### Objetivos

1. **Comprender el flujo de las peticiones, desde que sale de un navegador hasta la bbdd.**
2. **Explicar el modelo de capas**

### Recursos formativos

<https://www.youtube.com/watch?v=T_DOdKInZNk&list=PLU8oAlHdN5Blq85GIxtKjIXdfHPksV_Hm&index=26> (26-30)

<https://www.youtube.com/watch?v=z6WppAQ3LUg>

<https://www.youtube.com/watch?v=-0b6VtyU5S0>

### Posibles preguntas

¿Cómo funciona una petición http?

http es un protocolo que nos Permite transmitir informacion por internet entre el cliente y el servidor utilizando un sistema de peticiones y respuestas. http define la sintaxis y la semantica que utilizan los elementos de software de la arquitectura web para comunicarse.

Una peticion http esta compuesta de las siguientes secciones

Url: es la direccion hacia donde se dirige la peticion ejemplo: htt`//google.com

Metodo: es la accion segun se vaya a realizar en la peticion. Estos metodo son llamados verbos y son GET, UP, POST Y DELETE.

Header: le sirven al cliente para enviar infomracion al servidor . Es un espacio destinado para enviar metadata en la peticion http.

Status: nos dice si la peticion se realizo correctamente 202/404/500

Cuerpo: contiene la informacion que queremos enviar al servidor

Response: informacion de respuesta a esta peticion.

¿Qué es MVC?

Es un patron de diseño que separa conceptos y asi conseguir una alta cohesion de clases. El modelo, vista y controlador son estos conceptos. Lo mas importante del diseño modelo vista controlador es:

* Diseñar la logica del negocio(Modelo)
* Diseñar la Vista para mostrar el modelo
* Conectarlos a través del controlador

¿Cuáles son los componentes por los que pasa la petición?

La peticion pasa por el Controlador este lo procesa y consulta con el Modelo

El Modelo recibe la consulta y le proporciona la informacion. El controlador recibe la informacion y se la envia a la Vista.

La Vista recibe la informacion y la muestra. La vista es una pagina web.

¿Para que sirven las 3 capas?

Sirven para modularizar el proyecto. Mantener un orden y dividir la responsabilidades de la creacion de la aplicaion.

¿Ventajas?

Favorece el desarrollo

Favorece al mantenimiento

Favorece la depuracion

Favorece la escalabilidad

¿Hay otros modelos y/o arquitecturas? (MVVM, MVW, MVP...) ¿Cuándo usar uno u otro? ¿Cuales?

¿Qué es el modelo por capas?

### Tareas

NA

### Dudas

¿si tengo muchos archivos controlador que orden sigue de ejecucion?

¿?

¿?

¿?

¿?

¿?

## REST

### Objetivos

1. **Comprender el funcionamiento de las aplicaciones basadas en REST,**
2. **Comprender las diferencias con arquitecturas web clásicas,**
3. **Conocer los principales métodos y herramientas para trabajar con servicios REST**

### Recursos formativos

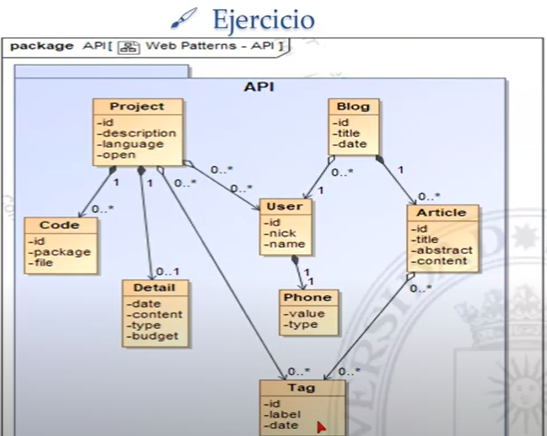
* Píldoras video
  + Intro a REST: <https://www.youtube.com/watch?v=aoOKxUOhg18> (8 min)
  + 6 consejos para que diseñes bien tu API REST <https://www.youtube.com/watch?v=bUmy7Nvsh4s&t=413s> (16 min)
  + Aprende A Documentar Tu API Con Swagger
* Long videos
  + Diseño de APIs REST <https://www.youtube.com/watch?v=3EceTpFV4Ys> (70 min)
  + API Rest. Introducción. Buenas prácticas <https://www.youtube.com/watch?v=t_L88F3lTdQ> (42 min)
* Lecturas
  + <https://www.arquitecturajava.com/servicios-rest/>
  + <https://bravedeveloper.com/2022/06/06/estilos-de-arquitectura-para-construir-apis/>
  + <https://desarrolloweb.com/articulos/que-es-rest-caracteristicas-sistemas.html>
  + <https://antoniofernandez.com/documentar-api-swagger/>

### Posibles preguntas

* ¿Qué significa REST? ¿Para qué se usa? ¿En qué tipo de aplicaciones?
* REST (REPRESENTANTIONAL STATE TRANSFER) es una interfaz par conectar varios sistemas basados en el protocolo HTTP . Se usa para obtener y generar datos y operaciones, devolviendonos ciertos datos en unos formatos muy especificos, como XML y JSON.
* REST se utiliza comunmente en aplicaciones web y moviles, especialmente en aplicaciones que utlizan una arquitectura de cliente – servidor.
* Explicar la arquitectura de una aplicación Front - Back
* ¿Cómo funcionan las peticiones?
* ¿Qué formatos de datos puede manejar?
* Los formatos usados son xml, json, formato binario o texto plano.
* ¿Qué es API Rest?
* Api es un abstraccion de funciones y procedimientos
* Rest es una logica de restricciones y recomendaciones bajo la cual se puede construir un api, es un estilo de arquitectura.
* API REST es una api ya implementada que esta contruida utilizando la logica de rest
* ¿Cuál es la diferencia con SOAP?
* La diferencia es que rest esta orientados a recursos y soap a operaciones y transacciones
* ¿Qué es el concepto Stateless?
* Debido a que el servidor necesita que cada requerimiento tenga toda la informacion necesaria para poder ejecutar un procedimiento se dice que Rest es stateless ya que no se necesita guardar informacion o el estado de peticiones anteriores para poder satisfacer peticiones nuevas. Cada peticion es independiente de otras he ahi el significado de sus siglas.
* ¿Cómo puedo trabajar con WebService con Java?
* ¿Qué es el principio HATEOAS?
* Rest no es un conjuntos de cosas que viven cada uno aislado sino que estan interconectadas entre si y que incluso el propio servirdor te puede modelar flujos o procesos de negocio en base a las respuestas que te va devolviendo para que un cliente las interprete.
* Nos dice ademas que un API apartir de un recurso raiz puede ser recorrido sin necesidad de ninguna documentacion.
* En el contexto de Rest: ¿Qué es un Resource?
* Los recursos en REST pueden ser cualquier cosa, desde documentos HTML, imágenes, videos, archivos de audio, hasta cualquier tipo de datos que pueda ser representado en un formato estándar como JSON, XML, entre otros
* ¿cuáles son los niveles de madurez de REST?
* Nivel 0: http como capa de transporte
* Nivel 1: orientamos el diseño a recursos
* Nivel 2: uso correcto de verbos http
* Nivel 3: añadimos controles hypermedia

### Tareas

* Plantear distintos Endpoints para el siguiente modelo de datos
* Documentarlo con OpenAPI (Swagger)
* Generar el código tanto de servidor como de cliente de manera automática mediante Swagger (probarlo)



### Dudas

¿?

¿?

VIP: registra las dudas que tengas para la sesión con el formador!!

¿?

¿?

¿?

¿?

## Spring Boot

### Objetivos

* **Conocer que es Spring boot, que ventajas aporta y sus configuraciones básicas**

### Recursos formativos

* Píldoras videos
  + <https://www.youtube.com/watch?v=rnyObUvNEdU> (7 min)
* Lecturas:
  + https://spring.io/guides/gs/spring-boot/
  + <https://www.arquitecturajava.com/que-es-spring-boot/>
  + <https://www.arquitecturajava.com/spring-boot-starter-un-concepto-fundamental/>
  + <https://www.arquitecturajava.com/spring-boot-devtools-y-recarga-de-aplicaciones/>
  + <https://www.baeldung.com/spring-boot>
  + https://www.baeldung.com/spring-vs-spring-boot
* Ejemplos
  + <https://github.com/spring-projects/spring-petclinic> (<https://www.baeldung.com/spring-boot-package-structure>)

### Posibles preguntas

* ¿Qué es Spring Boot? ¿Cómo se usa/configura? ¿Qué ventajas aporta?
* ¿Es cierto que solo sirve para crear aplicaciones web?
* ¿Cuál es la diferencia con Spring? ¿Si uso Spring estoy usando Spring Boot?
* ¿Para qué sirve Spring initializer?
* ¿Por qué se asocia tanto Spring Boot a los Microservicios?

### Tareas

### Dudas

## Spring REST

### Objetivos

* **Conocer las opciones que proporciona Spring para crear servicios REST.**
* **Adquirir conocimiento sobre buenas prácticas en diseño de servicios REST**

### Recursos formativos

Lecturas

* + - <https://www.arquitecturajava.com/responseentity-spring-y-rest/>
  + https://www.arquitecturajava.com/swagger-documentando-nuestro-api-rest/

Curso

* + (pildorasInformáticas) <https://www.youtube.com/watch?v=cI0F-9eLn2g> (videos 108 -111)
  + CÓMO CREAR UN API REST EN SPRING BOOT (<https://www.youtube.com/playlist?list=PLf8XMtbjh0dUIrBN0OEx_PJLQJRmV6tp1>)
  + Seguridad en Spring boot [Protege tu API usando JWT y Spring Security (2022) - YouTube](https://www.youtube.com/watch?v=_p-Odh3MZJc)
  + Curso práctico de APIs REST en Spring Boot con JWT , Spring Security , MySQL y Spring Data JPA <https://www.youtube.com/watch?v=dJaY43Butm8> (complete 5hrs(

Píldoras video

* + - * CÓMO CONFIGURAR SWAGGER PARA TU API REST CON SPRING BOOT <https://www.youtube.com/watch?v=ECTRRXJalh0> 8min

### Posibles preguntas

* ¿Cuáles son los componentes de Spring Rest?
* ¿Qué ocurre cuando uso la anotación @RestController? ¿Es distinto a @Controler?
* ¿Cómo podemos gestionar la seguridad en el backend?
* ¿Para qué sirve RestTemplate?
* ¿para qué nos sirve un Header?
* ¿Qué podemos hacer con el ResponseEntity?
* ¿Qué es JWT?

### Tareas

### Dudas

¿?