DOCUMENTO CON INFORMACION PRESENTACION DE PROCESEOS Y SERVICIOS

Contenido

[1. INTRODUCCION 3](#_Toc159180340)

[2. QUÉ SON LOS SERVICIOS WEB 3](#_Toc159180341)

[3. QUE ES UN SERVICIO WEB REST 3](#_Toc159180342)

[3.1 Principios Fundamentales de REST 3](#_Toc159180343)

[3.2 Características clave de los Servicios Web REST 4](#_Toc159180344)

[4. Diferencias entre SOAP Y REST 5](#_Toc159180345)

[5. Servicio Web Rest en JAVA 5](#_Toc159180346)

[6. Despliegue de aplicaciones 5](#_Toc159180347)

[PREGUNTAS KAHOOT 6](#_Toc159180348)

# INTRODUCCION

* Importancia de los servicios web: Son fundamentales en el desarrollo de aplicaciones modernas, permitiendo la comunicación entre diferentes componentes y sistemas de manera eficiente.
* Servicios web REST: Enfocaremos nuestra atención en los servicios web REST debido a su relevancia y popularidad en la actualidad.

# QUÉ SON LOS SERVICIOS WEB

* Definición: Tecnologías que permiten la comunicación entre aplicaciones a través de la web usando estándares como HTTP.
* Función: Facilitan la integración de sistemas distribuidos al proveer una interfaz estandarizada para la comunicación y el intercambio de datos.
* Permite la comunicación remota: Aplicaciones pueden interactuar entre sí de forma eficiente y segura a través de internet.

# QUE ES UN SERVICIO WEB REST

* Definición de REST (Representational State Transfer): REST es un estilo arquitectónico para diseñar sistemas distribuidos basados en la web. Se basa en principios simples y estándares de la web, como HTTP y URI, para permitir la comunicación entre sistemas de manera eficiente y escalable.
* Principios fundamentales de REST:
  + **Recursos:**
  + **URI (Identificadores de Recursos Uniformes):**
  + **Métodos HTTP (GET, POST, PUT, DELETE):**
  + **Representaciones de recursos:**
* Características clave de los servicios web REST:
  + **Stateless (sin estado):**
  + **Cacheable (almacenable en caché):**
  + **Interfaz uniforme:**
  + **Cliente-servidor:**
  + **Sistema en capas:**

# Principios Fundamentales de REST

* **Recursos**: En REST, todo es considerado como un recurso, que puede ser cualquier cosa que se pueda nombrar o identificar. Por ejemplo, un usuario, una publicación en un blog, etc.
* **URI (Identificadores de Recursos Uniformes):** Cada recurso en un sistema REST se identifica de manera única a través de un URI, que es una cadena de caracteres que proporciona una ubicación única para acceder al recurso.
* **Métodos HTTP (GET, POST, PUT, DELETE):** REST utiliza los métodos estándar de HTTP para realizar operaciones sobre los recursos. Por ejemplo, GET para obtener recursos, POST para crear nuevos recursos, PUT para actualizar recursos existentes y DELETE para eliminar recursos.
* **Representaciones de recursos:** Los recursos en un sistema REST pueden tener múltiples representaciones, como JSON o XML, lo que permite a las aplicaciones comunicarse de manera flexible y eficiente.

# Características clave de los Servicios Web REST

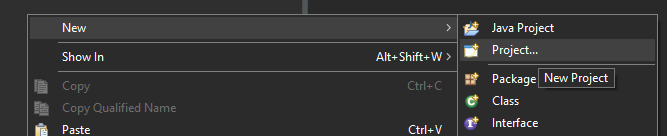
* **Stateless (sin estado):** Cada solicitud de cliente al servidor en un servicio web REST debe contener toda la información necesaria para comprenderla, lo que significa que el servidor no guarda información sobre el estado del cliente entre solicitudes. Esto mejora la escalabilidad y la confiabilidad del sistema.
* **Cacheable (almacenable en caché):** Los recursos en un servicio web REST pueden ser marcados como cacheables, lo que permite a los clientes almacenar en caché las respuestas y mejorar el rendimiento del sistema.
* **Interfaz uniforme:** Los servicios web REST tienen una interfaz uniforme que simplifica la arquitectura del sistema y promueve la interoperabilidad entre diferentes aplicaciones y sistemas.
* **Cliente-servidor:** REST sigue el modelo cliente-servidor, donde el cliente y el servidor son entidades independientes que se comunican a través de una interfaz uniforme.
* **Sistema en capas:** Los sistemas REST pueden ser construidos en capas, lo que permite una arquitectura escalable y flexible al separar las preocupaciones entre los diferentes componentes del sistema.

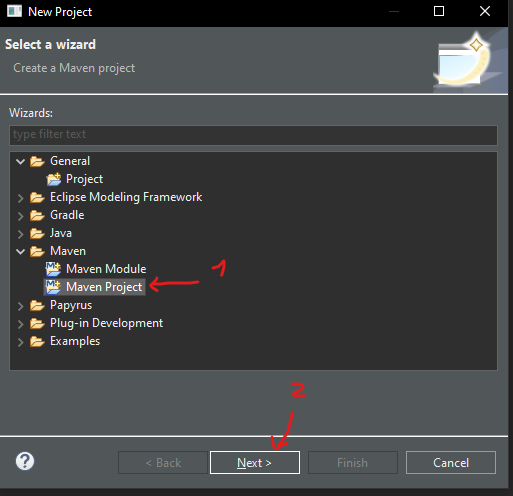
# Diferencias entre SOAP Y REST

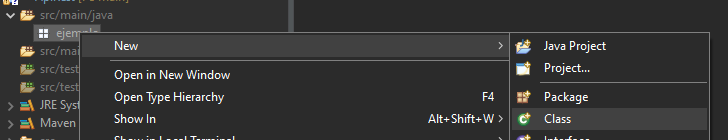
* SOAP es un protocolo de comunicación basado en XML utilizado para intercambiar información estructurada en la web.
* Diferencias clave entre SOAP y REST:
  + **Protocolo de comunicación:**
    - SOAP: Se adapta a diversos protocolos de transporte como HTTP, SMTP o TCP.
    - REST: Se basa principalmente en HTTP, simplificando la implementación y mejorando la interoperabilidad.
  + **Formato de mensaje:**
    - SOAP: Utiliza XML, lo que lo hace más pesado y complejo.
    - REST: Utiliza formatos ligeros como JSON o XML, reduciendo la sobrecarga de datos y mejorando el rendimiento.
  + **Acciones permitidas:**
    - SOAP: Define operaciones estándar (CRUD) a través de WSDL.
    - REST: Utiliza métodos HTTP estándar (GET, POST, PUT, DELETE), simplificando la interfaz y la implementación.
  + **Manejo de errores:**
    - SOAP: Maneja errores mediante códigos de estado y mensajes definidos en el estándar.
    - REST: Utiliza códigos de estado HTTP, simplificando el manejo de errores.

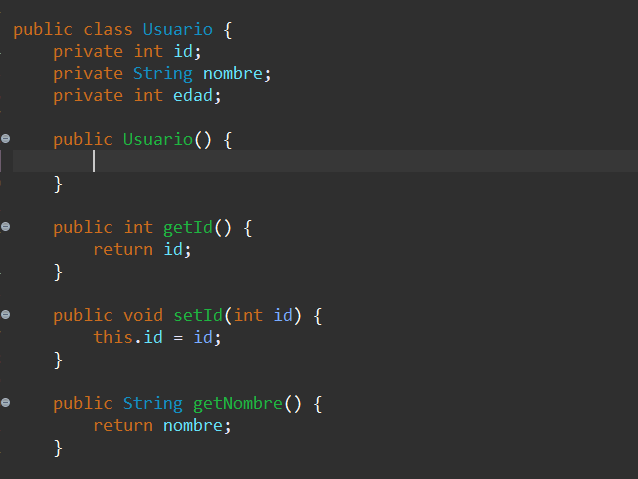
# Servicio Web Rest en JAVA

Creamos un Maven Project

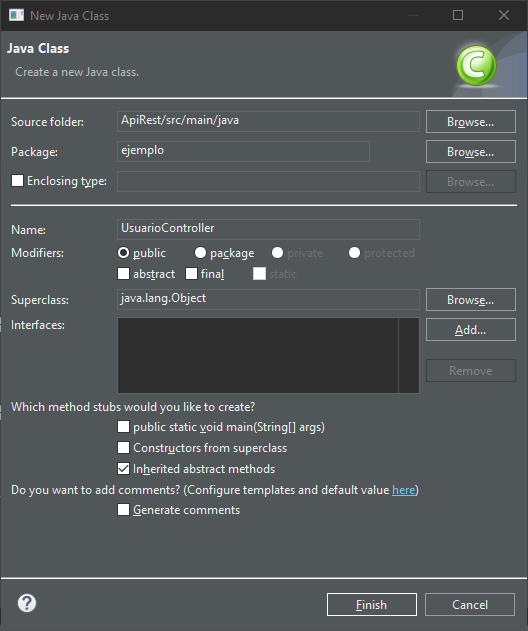


  
Creamos una clase Usuario

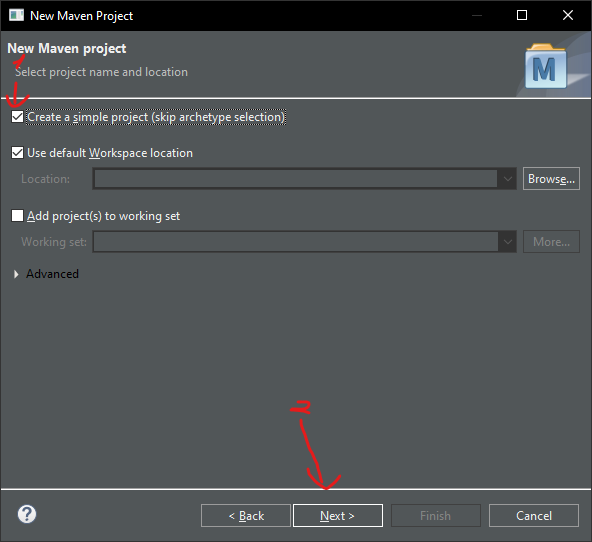


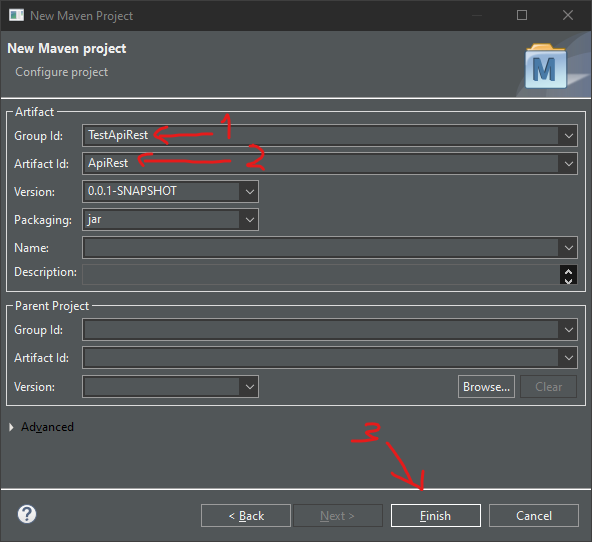


Añadimos un constructor vacío y getters y setters

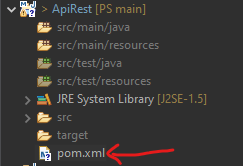


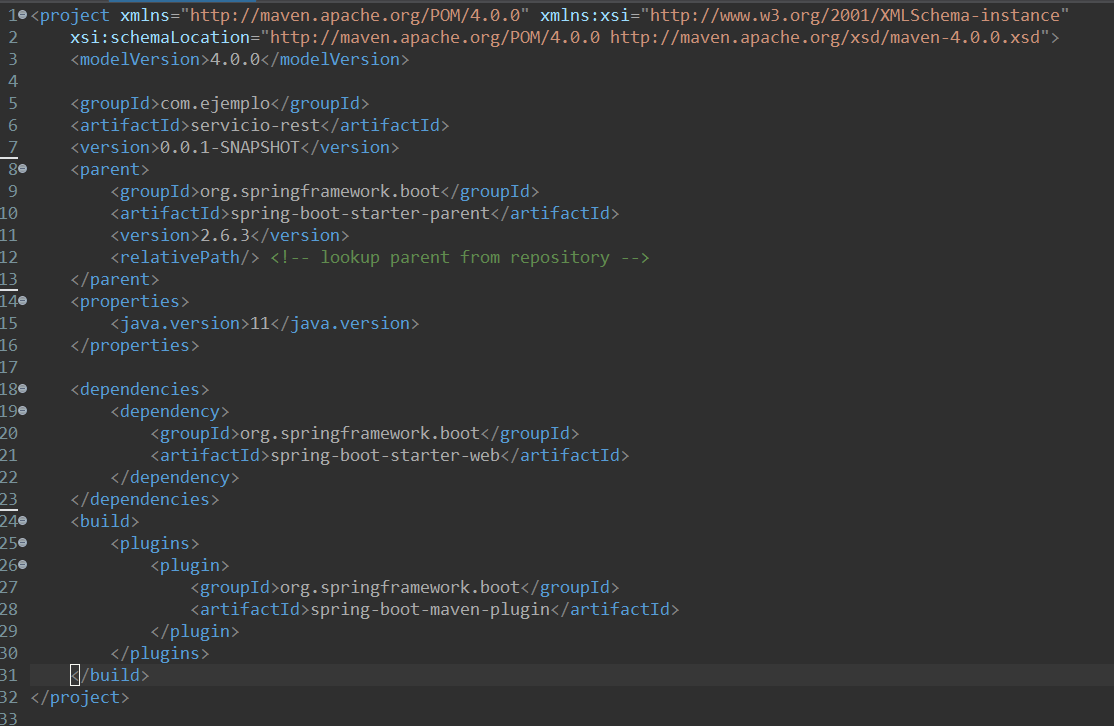
Creamos otra clase denominada UsuarioController



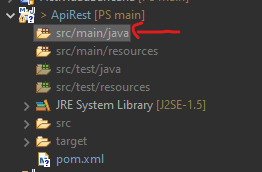


Añadimos las dependencias en el archivo pom.xml

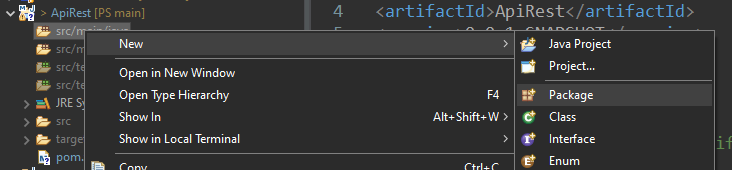


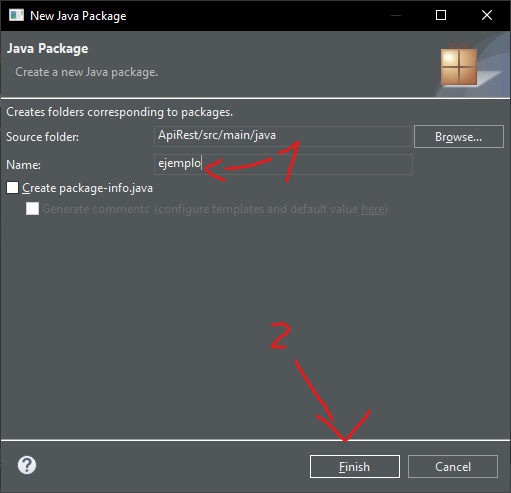


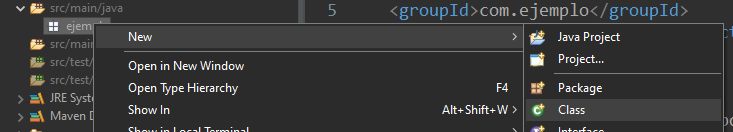
Añadimos las dependencias necesarias para poder operar con el framework de Spring



Creamos un paquete en este directorio del proyecto Maven.

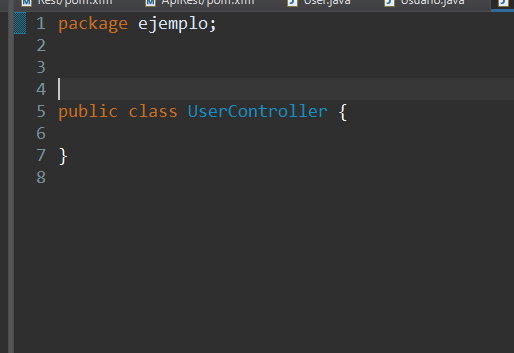








Creamos una clase POJO de User



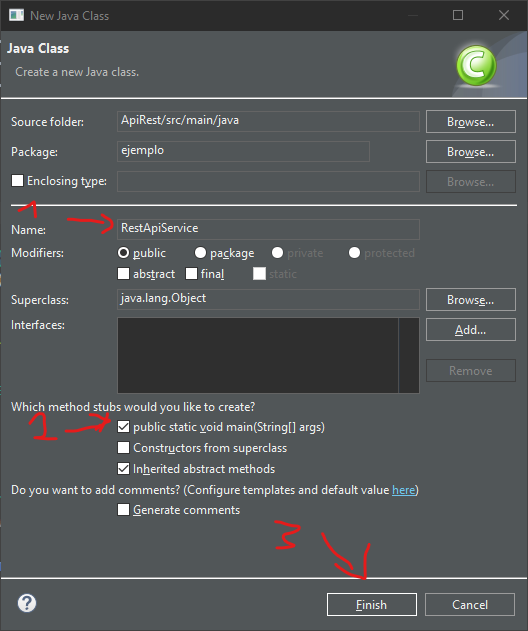
Creamos una clase denominada UserController



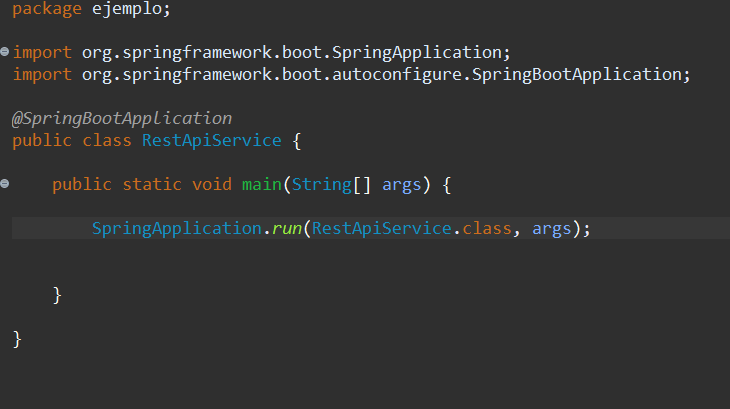
Añadimos notificaciones en la clase para indicar que esta clase va a ser la designada para controlar todas las peticiones a una fuente de datos (en este caso devolverá un usuario que creamos al momento de ejecutar el método, pero podría ser una consulta a una BBDD en la que devolvería un usuario por el id)

La anotación @GetMapping sirve para indicar la entrada que se debe introducir en la uri para poder acceder a un recurso

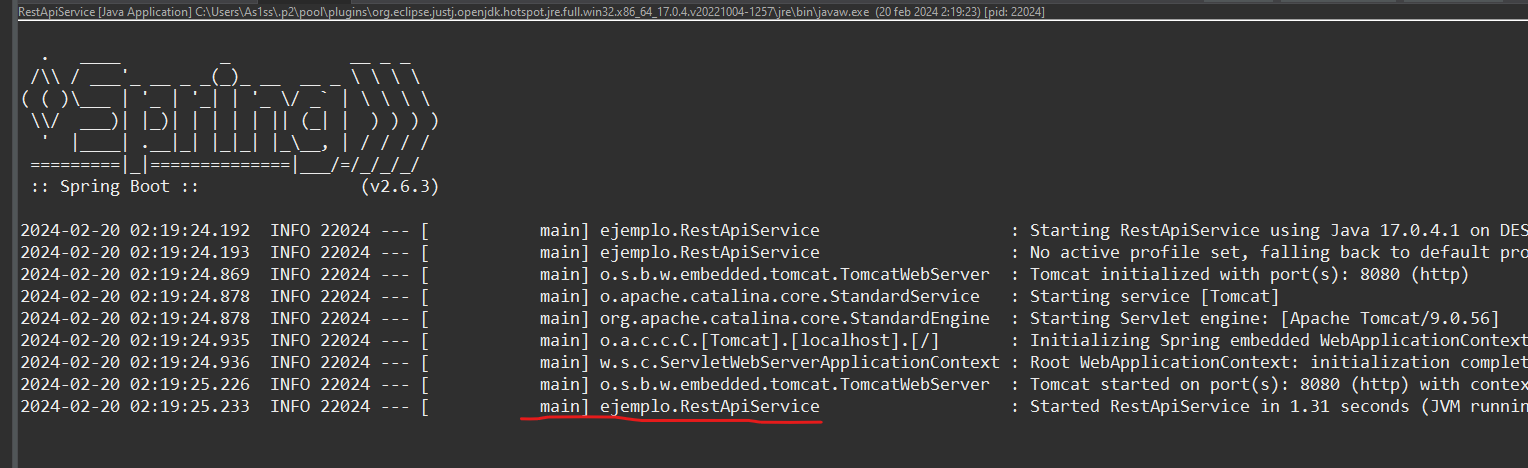
La anotación @PathVariable indica que esa variable va a ser la entrada introducida por el usuario en la url



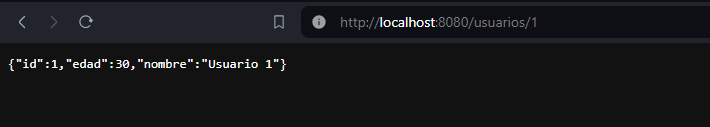
Por ultimo creamos una clase main para poder iniciar el servicio Rest



En esta clase agregaremos la anotación @SpringBootAplication para indicar que esta clase será la clase ejecutable del servicio rest y el método StringApplication.run() con la clase ejecutora, es decir la propia clase

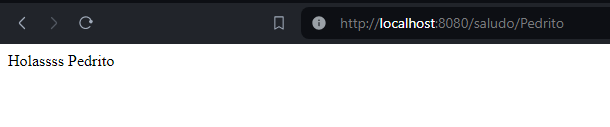


Si todo ha ido bien debería aparecer tu clase main seguido con el mensaje Started <nombreClase> in x seconds



Nos dirigimos al navegador e introducimos la siguiente dirección URI <http://localhost:8080/usuarios/1>

Con el primer método de UserController podemos visualizar los usuarios que en este caso se crean en tiempo real, es decir si introducimos 1 nos devolverá un usuario con el id 1, si escribiésemos 2 pues nos devolvería el Usuario 2 etc.



Si introducimos la dirección URI <http://localhost:8080/saludo/nombre>

Nos devolverá un saludo con el nombre que le indiquemos

# Despliegue de aplicaciones

* Los despliegues de aplicaciones se refieren al proceso de implementar y ejecutar una aplicación en un entorno operativo específico, como un servidor web o un servidor de aplicaciones, para que esté disponible para su uso.
* Implica configuración de recursos, dependencias y entorno de ejecución.
* Crucial para llevar la aplicación desde el desarrollo hasta la producción.
* Parte esencial del ciclo de vida del desarrollo de software.

# PREGUNTAS KAHOOT

**¿Qué es SOAP?**

a) Un protocolo de comunicación basado en XML.

b) Un estilo arquitectónico para sistemas distribuidos.

c) Un formato de mensaje ligero como JSON.

d) Un método HTTP utilizado en servicios web REST.

**¿Qué es REST?**

a) Un protocolo de comunicación basado en XML.

b) Un estilo arquitectónico para sistemas distribuidos.

c) Un formato de mensaje ligero como JSON.

d) Un método HTTP utilizado en servicios web SOAP.

**¿Cuál es una diferencia clave entre SOAP y REST en cuanto al formato de mensaje?**

a) SOAP utiliza XML, mientras que REST utiliza JSON.

b) SOAP utiliza JSON, mientras que REST utiliza XML.

c) Ambos utilizan exclusivamente XML.

d) Ambos utilizan exclusivamente JSON.

**¿Cuál de los siguientes es un principio fundamental de REST?**

a) Definición de operaciones estándar a través de WSDL.

b) Utilización de métodos HTTP estándar como GET, POST, PUT y DELETE.

c) Basado en un protocolo de transporte extensible como HTTP, SMTP o TCP.

d) Utilización de un protocolo de comunicación específico como SOAP.

**¿Cuál de las siguientes es una característica de SOAP pero no de REST?**

a) Utilización de métodos HTTP estándar.

b) Uso de XML como formato de mensaje.

c) Basado en un protocolo de transporte extensible.

d) Utilización de formatos ligeros como JSON.

**¿Qué tipo de formato de mensaje es más ligero y menos complejo en comparación con XML?**

a) JSON

b) XML

c) SOAP

d) HTTP

**¿Cuál de las siguientes afirmaciones es verdadera sobre el manejo de errores en SOAP y REST?**

a) SOAP utiliza códigos de estado HTTP para manejar errores.

b) REST tiene un mecanismo integrado para manejar errores a través de códigos de estado y mensajes definidos en el estándar.

c) REST utiliza códigos de estado HTTP para manejar errores.

d) SOAP no tiene un mecanismo específico para manejar errores.

¿Qué se entiende por "despliegue de aplicaciones"?

a) El proceso de desarrollar una aplicación desde cero.

b) La implementación y ejecución de una aplicación en un entorno específico.

c) La fase de prueba de una aplicación antes de su lanzamiento.

d) El proceso de desinstalación de una aplicación en un dispositivo.

¿Por qué son importantes los despliegues de aplicaciones en el desarrollo de software?

a) Porque permiten la creación de aplicaciones móviles.

b) Porque garantizan que la aplicación esté lista para ser descargada desde una tienda de aplicaciones.

c) Porque llevan la aplicación desde el desarrollo hasta el entorno de producción, donde los usuarios finales pueden acceder a ella.

d) Porque permiten la integración de nuevas características en una aplicación existente.