Servicios Web

En la Entrega 2 del proyecto se tiene que implementar un servicio web basado en REST, así como la posibilidad de reproducir los vídeos registrados. En la Figura 1 se puede ver una propuesta de arquitectura que muestra cómo conectar la aplicación web de la Entrega 1 con el servicio web de la Entrega 2 (que tiene que ser una nueva aplicación web). En esta entrega se sigue utilizando el servidor Glassfish. La funcionalidad a implementar se detalla a continuación.

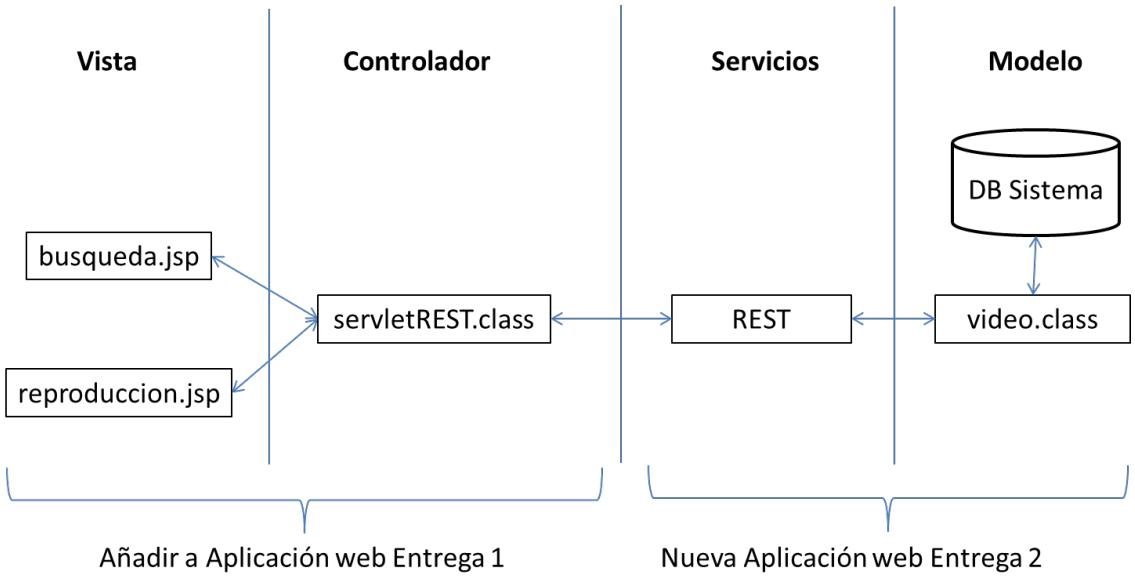

Figura 1. Propuesta arquitectura

Entrega 2 – Reproducción de vídeos y servicio Web REST

En la Entrega 2 se va a implementar la reproducción de los vídeos registrados en la base de datos. Además, hay que desarrollar un servicio web basado en REST que actualice el número de reproducciones de dicho vídeo y permita hacer búsquedas. Estas funcionalidades se deben **integrar con la aplicación web desarrollada en la Entrega 1**.

El trabajo se divide en dos partes, una para la reproducción de vídeos y otra para el servicio REST.

Parte 1 – Reproducción de vídeos

Hay que añadir al listado de vídeos de la aplicación web de la Entrega 1 la posibilidad de reproducirlos.

Existen distintas maneras de incrustar reproductores de audio y vídeo en una aplicación web. Un ejemplo de reproductor es VideoJs (`https://videojs.com/`), librería JavaScript de código abierto que permite reproducir audio y vídeo, y que ha sido desarrollada para funcionar con HTML5. No es obligatorio utilizar este reproductor concreto.

Parte 2 - RESTful Web Services

Implementar un servicio web basado en REST que actualice el número de reproducciones de los vídeos de vuestra aplicación cada vez que se pide visualizar un vídeo. Este servicio se debe implementar en una **nueva aplicación web**.

Además, este servicio debe realizar búsquedas sobre los vídeos almacenados en la base de datos. Deberá implementar, como mínimo, las siguientes operaciones:

- búsqueda por título
- búsqueda por autor
- búsqueda por fecha de creación (cualquier combinación de fecha es posible, sólo año, sólo mes/año o día/mes/año)

Como respuesta devolverá la información correspondiente al/los vídeo(s) (título, autor, fecha, duración, reproducciones y descripción) almacenada en la base de datos.

En el informe, indica las características tanto de la operación de actualización como de las de búsqueda. En concreto, indica el método HTTP que has utilizado, el formato y los parámetros de entrada (y cómo se pasan dichos parámetros, por la URL, como parámetros de un formulario u otro mecanismo) y el formato de salida (texto, HTML, XML, JSON, otro). Revisa la documentación de teoría referente a servicios web basados en REST para completar esta información.

Consultad el Anexo I para ver los detalles de creación de un RESTful Web service con Apache Netbeans.

Forma de entrega

Se tiene que entregar el código de las dos aplicaciones web, la de la Entrega 1 actualizada y la que implementa el servicio REST (sin librerías extra, sólo el código que hayáis implementado). Además, es necesario redactar un informe que describa las decisiones de diseño y la arquitectura final de vuestra solución para el servicio web REST. En este informe también se debe describir cualquier elemento extra que se haya desarrollado en la práctica. Utilizad la plantilla publicada en el Racó para dicho informe.

Anexo I: Creación de servicios web RESTful Apache Netbeans

En esta práctica vamos a implementar servicios web RESTful utilizando Apache Netbeans. Lo primero, es crear una aplicación web con Java with Maven → Web Application, tal y como ya se hizo en la Entrega 1. Una vez creada la aplicación, vemos que hay unos packages que contienen las clases necesarias para un RESTful Web Services. En concreto, en la clase `JakartaEE941Resource` ya viene implementado el método `ping()` (Figura 2) que responde al método GET de HTTP.

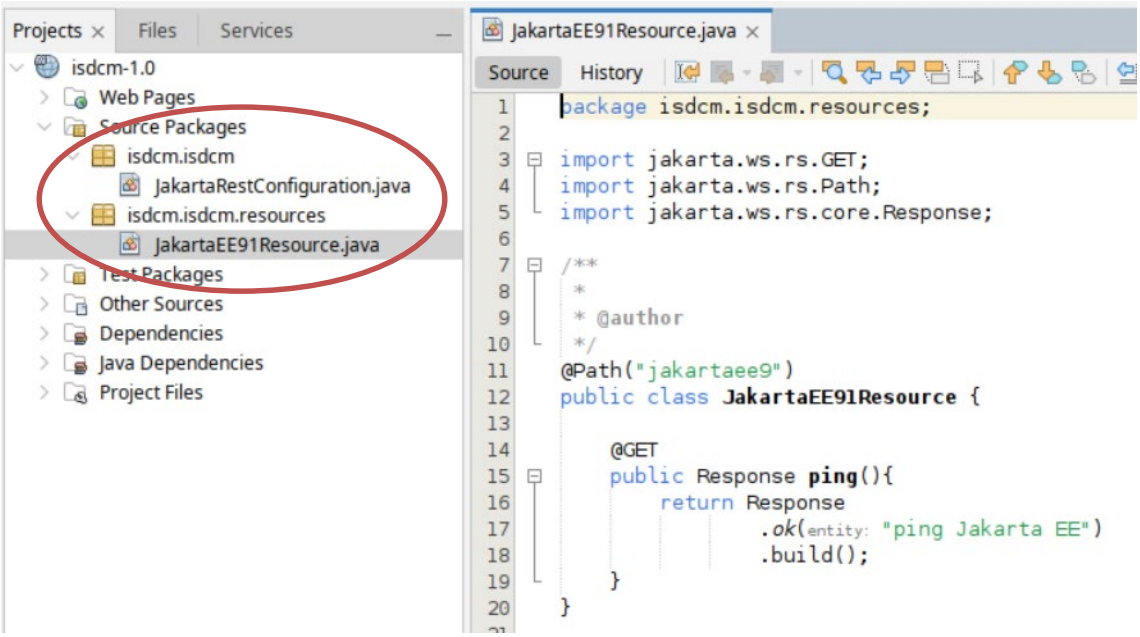

Figura 2. RESTful Web Service ya creado

Para acceder a este recurso, se tiene que ejecutar la aplicación y acceder a la siguiente URL: `http://localhost:8080/isdcm/resources/jakartaee9`, tal y como se ve en la Figura 3. Se puede llamar al servicio desde la barra de direcciones del navegador porque está configurado para que responda al método GET de HTTP (Anotación `Java @GET`).

Ver los ficheros `index.html` y `JakartaEE91Resource.java` que hay en el Racó para tener un ejemplo completo de servicio RESTful que llama a los métodos GET y POST. Puedes utilizar el código de estos ficheros en tu proyecto. ***Nota:*** En el caso del fichero Java, tendréis que comprobar el package Java para que funcione correctamente.

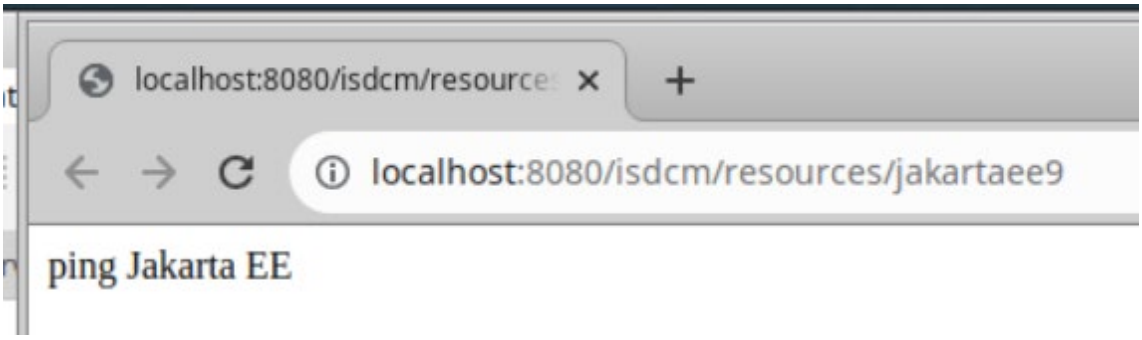

Figura 3. Acceso a la operación ping del servicio REST