Provisión de un API REST

OBJETIVO: Crear un API REST con el Micro-framework usado en la práctica 8 "Slim" para exponer la lógica de negocio de las operaciones previamente programadas como servicios SOAP (PHP y WCF), pero ahora cumpliendo con las características básicas de un API RESTful. Además, como parte del proceso de desarrollo y testeo de un servicio Web, se debe comprobar con POSTMAN el correcto funcionamiento del servicio Web resultante. Una vez que se ha comprobado que todas las operaciones trabajan correctamente, implementar una aplicación *cliente* con .Net, Java o JavaScript (HTML y CSS) que consuma el API REST.

I. Diseño del Servicio

Con base en la descripción del siguiente apartado y la experiencia que has adquirido en Prácticas previas, crea los diagramas UML de Casos de Uso y de Clases. También proporciona la especificación de los casos de uso (proporciona detalles textuales de cada caso de uso).

II. Implementación del API REST en PHP usando Slim.

A continuación la lista de especificaciones técnicas para cumplir con los requerimientos de un servicio Web tipo REST:

#	Operación	Ruta	Método (verbo)	*Headers* de la solicitud	*Body* de la solicitud
1	Obtener lista de *Productos* de determinada categoría.	*/productos/<categoria>*	GET	• user • pass	No Aplica
2	Obtener lista de *Detalles* de un producto en particular usando su clave (isbn)	*/detalles/<clave>*	GET	• user • pass	No Aplica
3	Insertar un nuevo *Producto* junto con sus *Detalles*	*/producto*	POST	• user • pass	• categoria • producto
4	Actualizar los *Detalles* de un *Producto*	*/producto/detalles*	PUT ó PATCH	• user • pass	• clave (isbn) • detalles
5	Eliminar *Producto* junto con sus *Detalles*	*/producto*	DELETE	• user • pass	• clave (isbn)

Ver: https://www.slimframework.com/docs/v4/objects/routing.html#how-to-create-routes

NOTA: Te puedes guiar del código adjunto a este material para implementar un proceso de *autenticación* a través de *headers* (user y pass), o puedes implementar tu propio mecanismo de autenticación revisando la documentación de Slim.

A continuación la lista anterior de operaciones, pero especificando los tipos de respuesta. Estos tipos de respuesta, son las estructuras JSON que se han especificado ya en las Prácticas de este curso, así que este sólo es un recordatorio:

#	Operación	Ruta	Respuesta	"data"
1	Obtener lista de *Productos* de determinada categoría.	*/productos/<categoria>*	{ "code": "", "message": "", "data": "", "status": "" }	{ "<isbn>": "<nombre>", "<isbn>": "<nombre>", ... "<isbn>": "<nombre>", }
2	Obtener lista de *Detalles* de un producto en particular usando su clave (isbn).	*/detalles/<clave>*	{ "code": "", "message": "", "data": "", "status": "", "oferta": "" }	{ "ISBN": "", "Autor": "", "Nombre": "", "Editorial": "", "Fecha": "", "Precio": "", "Descuento": "", }
3	Insertar un nuevo *Producto* junto con sus *Detalles.*	*/producto*	{ "code": "", "message": "", "data": "", "status": "" }	"AAAA-MM-DDTHH:MM:SS"
4	Actualizar los *Detalles* de un *Producto.*	*/producto/detalles*	{ "code": "", "message": "", "data": "", "status": "" }	"AAAA-MM-DDTHH:MM:SS"
6	Eliminar *Producto* junto con sus *Detalles.*	*/producto*	{ "code": "", "message": "", "data": "", "status": "" }	"AAAA-MM-DDTHH:MM:SS"

II. Comprobación del correcto funcionamiento con POSTMAN.

Como evidencia de la actividad de testeo, previo a consumir el servicio con una aplicación cliente programada en un lenguaje de programación de alto nivel, se deben guardar los *request* y los *response* de tus pruebas en un POSTMAN-Collection.

NOTA: se adjunta POSTMAN-*Collection* correspondiente al tutorial de la Práctica 8.

III. Aplicación con Interfaz Gráfica de Usuario (GUI).

Desarrolla una aplicación Web o de Escritorio con interfaz gráfica de usuario (GUI), que simule un sistema de inventario utilizado por dos roles diferentes "*Ventas*" y "*Almacén*". Donde el departamento de "*Ventas*" tendrá acceso a las operaciones de consulta (GET) y el departamento de "*Almacén*" tendrá acceso a las operaciones de consulta (GET), de inserción (POST), actualización (PUT/PATCH) y eliminación (DELETE) de productos.

Las características de la interfaz gráfica son las siguientes:

- Cada departamento tendrá su propio apartado, ventana o pestaña.
- Deben utilizarse un logotipo, colores y metáforas apropiadas.
- La aplicación debe mostrar los datos de respuesta en forma de campos, de una lista o en forma de tabla. Es decir, el usuario no debe visualizar el JSON de respuesta.
- La aplicación debe ser la responsable de crear el JSON a enviar para insertar o actualizar un producto. Es decir, debe existir un formulario para especificar los datos del producto.
- Para el departamento de "Almacén", la aplicación deberá mostrar una lista o tabla de productos, misma que dispondrá de un mecanismo (link, botón, etc.) para elegir el producto a editar (actualizar). Una vez seleccionado el producto a editar, los datos deberán ser mapeados al formulario de registro para no tener que capturar todos los datos del producto que se va a actualizar.

III. Documentación.

Como parte de la evidencia de este trabajo, se debe entregar un documento con las siguientes partes:

1. Portada, con el nombre de los integrantes del equipo, iniciando por el representante.
2. Nombre del proyecto.
3. Resumen o descripción breve.
4. Evidencias de análisis y diseño (diagramas UML de Casos de Uso y de Clases).
5. Evidencias del proceso de pruebas con POSTMAN (capturas de pantalla).
6. Evidencias del consumo desde una aplicación cliente implementada con .Net, Java, o JavaScript (capturas de pantalla).