Proyecto SW2

Gestión de una agenda con XML, ahora con servicios RESTful

Enunciado

En este proyecto se busca poner en práctica lo aprendido en clase de REST. Para ello se va a diseñar un pequeño programa en Java que gestione una agenda utilizando un servicio web REST. Recordad que el enunciado mencionará Agenda y Persona, pero cada uno debe realizarlo sobre las clases que tiene asignadas.

En este caso además se pide que haya autentificación de usuarios y que cada uno tenga su propia Agenda almacenada en una base de datos.

Además, se pedirá que se sigan lo más fielmente posible los principios de la filosofía REST.

Estructura XML

Debéis definir una estructura XML de la Agenda (según corresponda) con XSD. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="emailAddress">
    <xs:restriction base="xs:token">
        <xs:maxLength value="254"/>
        <xs:pattern value="[_\-a-zA-Z0-9\.\+]+@[a-zA-Z0-
9](\.?[\-a-zA-Z0-9]*[a-zA-Z0-9])*"/>
    </xs:restriction>
</xs:simpleType>

<xs:complexType name="Persona">
        <xs:sequence>
         <xs:element name="name" type="xs:string"/>
    <xs:element name="email" type="emailAddress"/>
         <xs:element name="telephone" type="xs:string"/>
         <xs:any namespace="##any" minOccurs="0"
maxOccurs="unbounded" processContents="lax"/>
        </xs:sequence>
</xs:complexType>

<xs:element name="Agenda">
 <xs:complexType>
    <xs:sequence>
      <xs:element name="Persona" type="Persona" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="Persona" type="Persona"/>
</xs:schema>
```

Servicios a desarrollar:

1. Permitir validar objetos Personas con la XSD (sin autentificación)
2. Permitir validar objetos Agenda con la XSD (sin autentificación)
3. (Opcional) Validar la Agenda utilizando una DTD (sin autentificación)
4. Permitir obtener la Agenda de un usuario (con autentificación)
5. Permitir obtener una Persona en concreto de un usuario a partir del nombre (con autentificación)
6. (Opcional) Permitir obtener para un usuario el HTML de su Agenda directamente, convertido con XSLT (con autentificación)
7. Todas las operaciones CRUD sobre una Persona de la Agenda del usuario (Crear/Leer/Modificar/Borrar) (con autentificación)
8. Gestionar la autentificación del usuario.
9. (Opcional) Permitir que un mismo usuario tenga varias Agendas, por lo tanto, desarrollar las operaciones CRUD sobre Agenda también.
10. (Opcional) Permitir que existieran Agendas compartidas entre varios usuarios
11. Cualquier otro servicio que se considere necesario/útil.

Adicionalmente se pueden desarrollar los servicios de tal forma que admitan búsquedas.

También se debe intentar seguir la filosofía REST al completo al realizar estos servicios, incluyendo HATEOAS.

Cliente Java

1. Desarrollar un cliente Java que interactúe con los servicios creados.

Cliente Web

1. Desarrollar una aplicación web que consuma los servicios creados.

Interfaz

La interfaz de los clientes debe permitir acceder a toda la funcionalidad antes descrita:

1. Opción para crear una nueva agenda
2. Opción para obtener la Agenda contenida en el servicio web y mostrarla
3. Opción para obtener una Persona del servicio y mostrarla por pantalla
4. Opción para crear una Persona y añadirla al servicio web de la agenda
5. Opción para borrar una Persona del servicio web
6. Opción para actualizar una Persona del servicio web
7. Opción para enviar una Agenda para su validación
8. Opción para enviar una Persona para su validación
9. Autentificarse
10. Etc…

Estructura y diseño

1. Seguid siempre que sea posible las convenciones de nomenclatura de Java y sed coherentes en todo el proyecto
2. Separad claramente las distintas responsabilidades dentro de vuestro proyecto. Usad clases y paquetes para estructurar el código de forma adecuada, usando herencia y interfaces cuando sea conveniente.
3. La interacción entre los distintos componentes del sistema debe realizarse mediante objetos.
4. No deben aparecer rutas absolutas para los ficheros.
5. Debéis entregar tres proyectos distintos, uno que genera los servicios y los dos clientes.

Final

1. Revisar que se responde a lo que se pide.
2. Revisar que el código que enviáis funciona y que no hay errores obvios.
3. El código debe funcionar “out-of-the box”
4. Revisar que el proyecto que vais a enviar funciona sin la necesidad de ningún archivo adicional o que está incluido en el proyecto. Añadid la documentación necesaria para hacerlo funcionar. El proyecto debe ser autocontenido y será evaluado por otro grupo.