Prueba técnica Desarrollador

Contexto

Se busca automatizar el comportamiento de una biblioteca cuando un usuario desea prestar un libro, un préstamo es representado por los siguientes atributos:

isbn: identificador único de un libro (campo alfanumérico de máximo 10 dígitos)

identificacionUsuario: número de la identificación del usuario (campo alfanumérico de maximo 10 digitos)

tipoUsuario: determina la relación que tiene el usuario con la biblioteca (campo que puede tener solo un dígito numérico)

1: usuario afilado
2: usuario empleado de la biblioteca
3: usuario invitado

Objetivo

Su objetivo es crear una API tipo REST la cual permita llevar a cabo las siguientes funcionalidades

1. **Crear un servicio REST con el Path /prestamo y el método HTTP tipo POST: permite crear un préstamo con las siguientes validaciones**

Un usuario invitado solo puede tener un libro prestado en la biblioteca, si un usuario invitado intenta prestar más de un libro debería retornar un error HTTP 400 con el siguiente json.

```
{
  "mensaje" : "El usuario con identificación xxxxxx ya tiene un libro prestado por lo cual no se le puede realizar otro préstamo"
}
```

Para verificar si un usuario ya tiene un préstamo se debe usar el campo identificaciónUsuario

Al momento de realizar el préstamo se debe hacer el cálculo de la fecha máxima de devolución del libro, con las siguientes reglas de negocio:

- Si el préstamo lo hace un usuario tipo afiliado la fecha de devolución debería ser la fecha actual más 10 días sin contar sábados y domingos
- Si el préstamo lo hace un usuario tipo empleado la fecha de devolución debería ser la fecha actual más 8 días sin contar sábados y domingos • Si el préstamo lo hace un usuario tipo invitado la fecha de devolución debería ser la

fecha actual más 7 días sin contar sábados y domingos

Esta fecha deberá ser almacenada en la base de datos junto con toda la información del préstamo

Si en el campo tipoUsuario llega un valor diferente a los permitidos, deberá retornar un un error HTTP 400 con el siguiente JSON

```
{
  "mensaje" : "Tipo de usuario no permitido en la biblioteca"
}
```

El siguiente es un ejemplo de petición y un ejemplo de cómo debería ser la respuesta en un caso exitoso

Petición path: /prestamo método: POST

```
{
   "isbn":"DASD154212",
   "identificaciónUsuario":"154515485",
   "tipoUsuario":1
}
```

Respuesta exitosa: HTTP Status Code 200

```
{
    "id": 1,
   "fechaMaximaDevolucion" : "15/02/2021"
}
```

El id en la respuesta, corresponde al identificador con el que es almacenado en la base de datos, con el cual posteriormente se podrá consultar

Para fechaMaximaDevolucion se debe respetar el formato dd/MM/yyyy

2. **Crear un servicio REST con el Path /prestamo/{id-prestamo} y el método HTTP tipo GET, donde la variable {id-prestamo} corresponde al identificador con el cual se almacenó el préstamo en la base de datos, explicado en el primer punto.**

El siguiente es un ejemplo de petición y un ejemplo de cómo debería ser la respuesta en un caso exitoso

Petición path: /prestamo/1 método: GET

Respuesta exitosa HTTP Status Code 200

```
{
   "id": 1,

   "isbn":"DASD154212",

   "identificaciónUsuario":"154515485",

   "tipoUsuario":1,
   "fechaMaximaDevolucion" : "15/02/2021"
}
```

Detalles técnicos

Los WS los debe implementar en Spring Boot

La base de datos a utilizar es el motor postgresql en la versión 12 en adelante, no utilizar la BD interna H2 del Java.

3. **Crear un formulario en el framework Angular para el Frontend de la biblioteca, donde se puedan consumir los servicios creados previamente para (prestar y consultar), adicional otro formulario para listar los usuarios que tienen prestamos activos, para esto debe tener en cuenta la comprensión de sus características y de las buenas prácticas.**

• Crear Loguin
• Validar datos de entrada
• Manejar mensajes de error y confirmación cuando se crea, modifica o elimina un registro.
• Controlar errores en tiempo de ejecución.
• Debe utilizar programación orientada a objetos (Clases).

Conceptos para evaluar

✓ Cumplimiento de todos los requerimientos

✓ Asegúrese que al momento de enviar el código este se ejecute correctamente, esta es la manera de evaluar la prueba.

✓ Valoramos que su código sea mantenible y con principios de código limpio.

✓ Arquitectura: valoramos que la arquitectura propuesta demuestre una

correcta separación de responsabilidades.

✓ Te recomendamos hacer uso del principio de responsabilidad única

✓ Te recomendamos usar un patrón de arquitectura, como arquitectura hexagonal, arquitectura limpia o MVC.

✓ Trata de no poner la lógica de negocio en los controladores, separa tu lógica de acuerdo a las restricciones del patrón de arquitectura seleccionado.

✓ Pruebas unitarias y de integración (deseable): valoramos si logra construir pruebas unitarias y de integración de su lógica de negocio (Postman).

✓ Cuando termines la prueba favor enviarla al correo del cual fue recibida teniendo en cuenta el tiempo máximo de indicado, si se identifica que la prueba no ha sido desarrollada por usted inmediatamente se cancela el proceso de selección, posterior a esto se define una sesión virtual para la sustentación de esta.