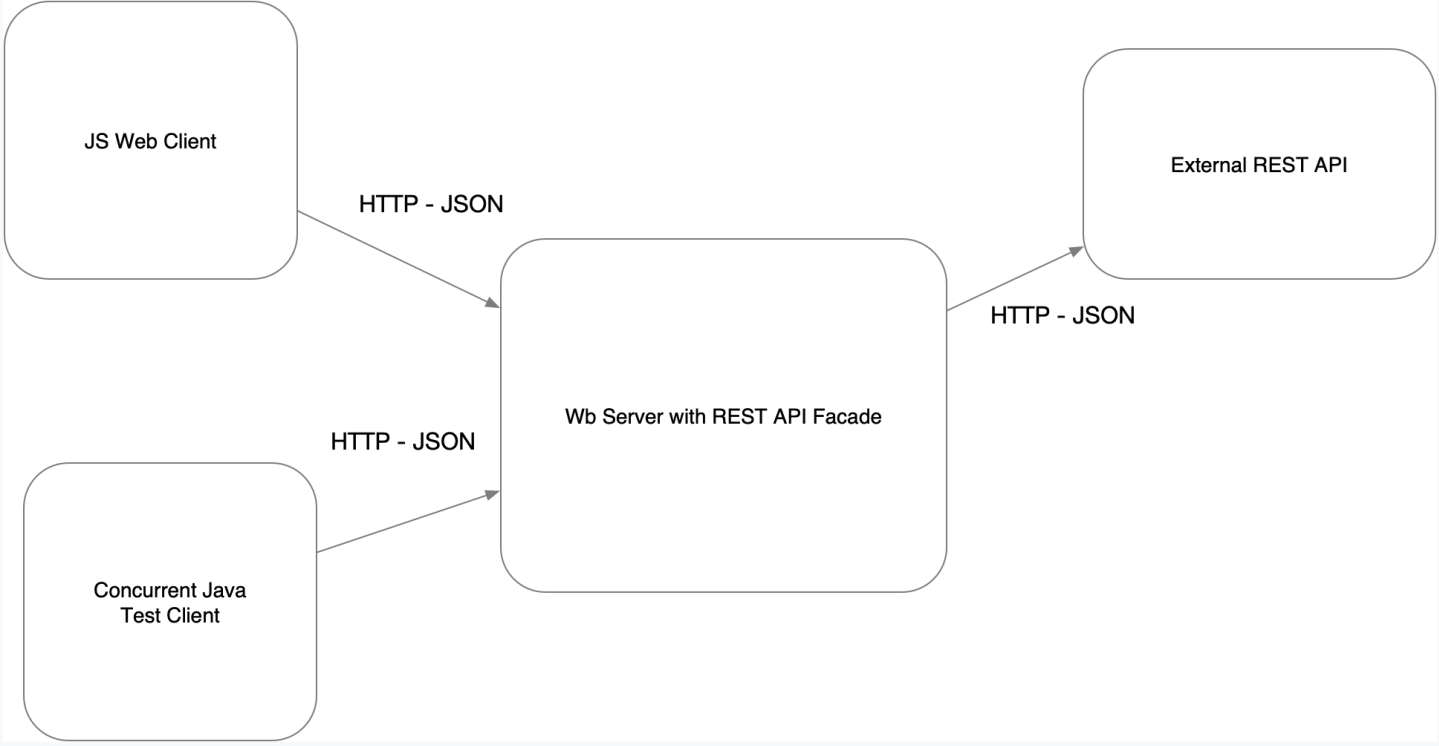
Apertura: lunes, 22 de enero de 2024, 21:23
Cierre: domingo, 28 de enero de 2024, 23:59

Debe construir una aplicación para consultar la información de películas de cine. La aplicación recibirá una frase de búsqueda del título, por ejemplo "Guardians of the galaxy" y deberá mostrar el los datos de la película correspondiente. Para esto utilice el API gratuito de https://www.omdbapi.com/ (Puede crear obtener una llave gratuita para realizar consultas). Se le pide que su implementación sea eficiente en cuanto a recursos así que debe implementar un Caché que permita evitar hacer consultas repetidas al API externo.

La arquitectura debe tener las siguientes características.

1. El cliente Web debe ser un cliente asíncrono que corra en el browser y use Json como formato para los mensajes.
2. El servidor de servirá como un gateway para encapsular llamadas a otros servicios Web externos.
3. La aplicación debe ser multiusuario.
4. Todos los protocolos de comunicación serán sobre HTTP.
5. Los formatos de los mensajes de intercambio serán siempre JSON.
6. La interfaz gráfica del cliente debe ser los más limpia y agradableolo HTML y JS (Evite usar librerías complejas). Para invocar métodos REST desde el cliente usted puede utilizar la tecnología que desee.
7. La fachada de servicios tendrá un caché que permitirá que llamados que ya se han realizado a las implementaciones concretas con parámetros específicos no se realicen nuevamente. Puede almacenar el llamado como un String con su respectiva respuesta, y comparar el string respectivo. Recuerde que el caché es una simple estructura de datos.
8. Se debe poder extender fácilmente, por ejemplo, es fácil agregar nuevas funcionalidades, o es fácil cambiar el proveedor de una funcionalidad.
9. Debe utilizar maven para gestionar el ciclo de vida, git y github para almacenar al código fuente y heroku como plataforma de producción.
10. En el backend debe utilizar solo Java. No puede utilizar frameworks como SPRING.



TODOS SUS ENTREGABLES DEBEN SER DE LA MAYOR CALIDAD POSIBLE. TRABAJE CON LA MAYOR CLARIDAD POSIBLE Y MANEJE UN ENTORNO DE DESARROLLO Y UNA ARQUITECTURA SIMPLE. SEA RIGUROSO CON SUS ESTÁNDARES DE PROGRAMACIÓN

Para el ejercicio debe entregar:

1. La aplicación funciona y hays instrucciones para ejecutarla correctamente al bajarla desde un proyecto GitHub.
2. Los fuentes deben estar documentados y bien estructurado para generar el Javadoc.
3. El Readme.md debe describir el diseño, explicar cómo se puede extender y cómo podría, por ejemplo, hacer que una función específica la implementara un proveedor de servicios diferentes.
4. Indique la urls de Github
5. Guarde una copia de su proyecto.

Criterios de evaluación:

1. Cliente escrito en JS invocando servicios REST (10%)
2. Servidor fachada exponiendo servicios REST (10%)
3. Conexión a servicios externos (20%)
4. Aplicación respecta arquitectura (10%)
5. Cache (10%)
6. Incorpora TDD. (10%)
7. Diseño y descripción del diseño son de alta calidad (30%)
    1. Extensible
    2. Usa patrones
    3. Modular
    4. Organizado
    5. Identifica la función de componentes individuales demuestra conocimiento del funcionamiento general de la arquitectura.

1.

Ayuda

1. Para invocar un servicios get desde java puede hacerlo de manera fácil con:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;

public class HttpConnectionExample {

    private static final String USER_AGENT = "Mozilla/5.0";
    private static final String GET_URL = "https://www.alphavantage.co/query?
function=TIME_SERIES_DAILY&symbol=fb&apikey=Q1QZFVJQ21K7C6XM";

    public static void main(String[] args) throws IOException {

        URL obj = new URL(GET_URL);
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", USER_AGENT);

        //The following invocation perform the connection implicitly before getting the code
        int responseCode = con.getResponseCode();
        System.out.println("GET Response Code :: " + responseCode);

        if (responseCode == HttpURLConnection.HTTP_OK) { // success
            BufferedReader in = new BufferedReader(new InputStreamReader(
                    con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            // print result
            System.out.println(response.toString());
        } else {
            System.out.println("GET request not worked");
        }
        System.out.println("GET DONE");
    }

}
```

2. Para invocar servicios rest de forma asíncrona desde un cliente JS

```
<!DOCTYPE html>
<html>
    <head>
        <title>Form Example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>Form with GET</h1>
        <form action="/hello">
            <label for="name">Name:</label><br>
            <input type="text" id="name" name="name" value="John"><br><br>
            <input type="button" value="Submit" onclick="loadGetMsg()">
        </form>
        <div id="getrespmsg"></div>

        <script>
            function loadGetMsg() {
                let nameVar = document.getElementById("name").value;
                const xhttp = new XMLHttpRequest();
                xhttp.onload = function() {
                    document.getElementById("getrespmsg").innerHTML =
                    this.responseText;
                }
                xhttp.open("GET", "/hello?name="+nameVar);
                xhttp.send();
            }
        </script>

        <h1>Form with POST</h1>
        <form action="/hellopost">
            <label for="postname">Name:</label><br>
            <input type="text" id="postname" name="name" value="John"><br><br>
            <input type="button" value="Submit" onclick="loadPostMsg(postname)">
        </form>

        <div id="postrespmsg"></div>

        <script>
            function loadPostMsg(name){
                let url = "/hellopost?name=" + name.value;

                fetch (url, {method: 'POST'})
                    .then(x => x.text())
                    .then(y => document.getElementById("postrespmsg").innerHTML = y);
            }
        </script>
    </body>
</html>
```

Editar entrega | Borrar entrega

ESTADO DE LA ENTREGA

Estado de la entrega	Enviado para calificar
Estado de la calificación	Calificado
Tiempo restante	La tarea fue enviada 21 horas 47 minutos después
Última modificación	lunes, 29 de enero de 2024, 21:46
Texto en línea	https://github.com/ELS4NTA/AREP-LAB-01.git
Comentarios de la entrega	Comentarios (0)

COMENTARIO

Calificación	48,00 / 50,00
Calificado sobre	sábado, 10 de febrero de 2024, 21:38
Calificado por	 JULIAN DAVID CASTILLO SOTO

Comentarios de retroalimentación

	Referncia	Evaluación
Entregables	7	7
Desplegado en github	1	1
Tiene .gitignore completo	1	1
Tiene README.md	1	1
No contiene archivos o carpetas basura	1	1
Tiene POM.xml	1	1
Respeta estructura de maven	1	1
No contiene la carpeta target	1	1
Diseño y Arquitectura	28	27
El diseño del sistema parece razonable para el problema	3	3
El diseño está bien documentado en el README.md	3	3
El README contiene instrucciones de instalación y uso	3	3
El README evidencia pruebas	3	3
Tiene pruebas automáticas	3	3
El repositorio se puede clonar y ejecutar	3	3
El desarrollo es completo y de alta calidad	10	9
Total	35	34
Nota	5	4.85714286

bien!!!

ENLACES INSTITUCIONALES

Biblioteca

Investigación e innovación

Enlace - Académico

ENLACES DE INTERÉS

Ministerio de Educación Nacional

Colombia Aprende

Red Latinoamericana de Portales Educativos

Red Universitarias Metropolitana de Bogotá

CONTACTE CON NOSOTROS

AK.45 No.205-59 (Autopista Norte).

Teléfono: +57(1) 668 3600

Email: contactocc@escuelaing.edu.co