Taller: Diseño y Estructuración de Aplicaciones Distribuidas en Internet

Requisitos de finalización

En este taller usted explorará la arquitectura de las aplicaciones distribuidas. Concretamente, exploraremos la arquitectura de los **servidores web** y el **protocolo HTTP** sobre el que están soportados.

Reto

Escriba un **servidor web** que soporte múltiples solicitudes seguidas **no concurrentes**. El servidor debe:

- Leer archivos del disco local.
- Retornar todos los archivos solicitados:
    - HTML
    - JavaScript
    - CSS
    - Imágenes

Además, debe construir una aplicación web utilizando **JavaScript, CSS e imágenes** para probar su servidor.

La aplicación debe incluir **comunicación asíncrona** con unos servicios **REST** en el backend.

> **Importante:** No use frameworks web como *Spark* o *Spring*. Use únicamente **Java** y las librerías de manejo de red.

Entregables

- Proyecto con los retos implementados y funcionando.
- Repositorio público en **GitHub** con los más altos estándares de calidad.
- **README** con:
    - Instalación
    - Cómo ejecutarlo
    - Arquitectura del prototipo
    - Evaluación (pruebas realizadas)

Ayudas

1. Invocar servicios REST de forma asíncrona desde JavaScript

```
<!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>
```

---

2. Invocar un servicio REST desde Java

```
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 performs 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");
    }

}
```