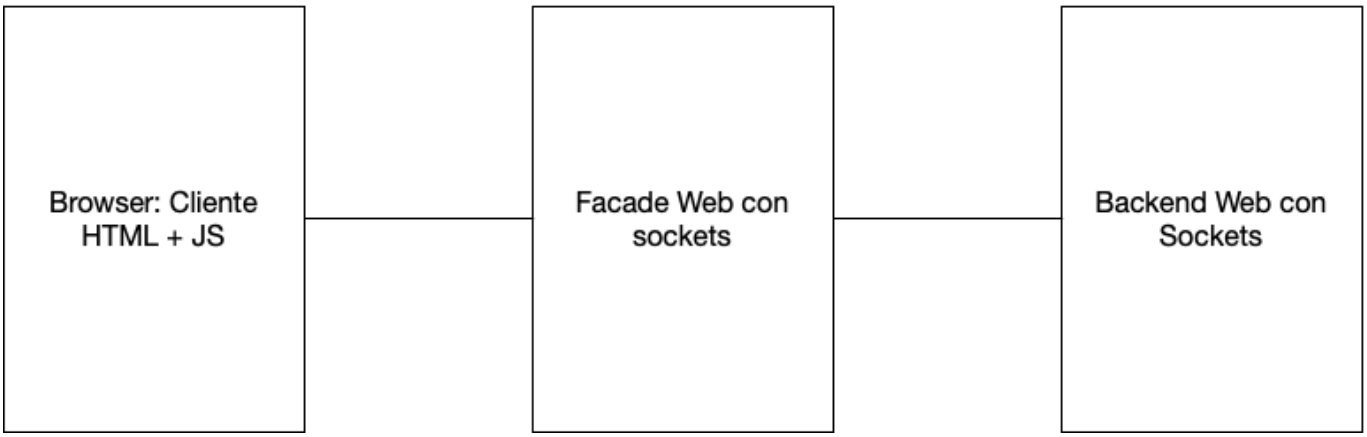
Taller de verificación de conocimientos técnicos 3

Requisitos de finalización

Requerimientos

- **No PUEDE** revisar código en internet ni código antiguo en su computador.
- No saque copias del enunciado.

Diagrama de Arquitectura



Descripción

Debe construir un **almacenamiento llave–valor distribuido**. Consta de:

- Un **servidor backend** que responde solicitudes HTTP GET desde la Fachada.
- Un **servidor Fachada** que responde GET desde el cliente.
- Un **cliente HTML+JS** que envía los comandos y muestra respuestas.

La API permite almacenar tuplas **(key, value)** tipo *string* en el backend, y recuperarlas por llave.

API mínimo (solo GET) para Fachada y Backend

1) Cliente → Fachada

1.1) GET `/setkv?key={key}&value={value}`

Crea o reemplaza el valor asociado a una llave.

Respuestas:

- `200 OK` – reemplazado o creado
- `400 Bad Request` – faltan datos o no son string

Ejemplo JSON

```
{ "key": "mi_llave", "value": "mi_valor", "status": "created" }
```

1.2) GET `/getkv?key={key}`

Obtiene el valor de una llave.

Respuestas:

- `200 OK`

```
{ "key": "mi_llave", "value": "mi_valor" }
```

- `404 Not Found`

```
{ "error": "key_not_found", "key": "mi_llave" }
```

---

2) Fachada → Backend

La Fachada reenvía el **mismo contrato**, propagando códigos y cuerpo.

2.1) GET `/setkv?key={key}&value={value}`

Respuestas iguales a la sección anterior.

2.2) GET `/getkv?key={key}`

Respuestas iguales a la sección anterior.

---

3) Consideraciones mínimas

- Usar `Content-Type: application/json; charset=utf-8`.
- Validaciones en Fachada:
  - tamaño máximo de key/value
  - recortar espacios
  - rechazar `key == ""`

- Errores estándar: incluir `"error"` y opcional `"message"`.

Arquitectura

- Tres componentes distribuidos:
  - Fachada
  - Backend
  - Cliente HTML+JS
- Fachada y Backend se ejecutan en **máquinas virtuales diferentes**.
- Cliente HTML+JS se entrega desde la fachada (puede ser un string en hardcode).
- Comunicación usando HTTP.
- Respuestas en **JSON**.
- Llamados del cliente → fachada deben ser **asíncronos con JS mínimo**.
- No recargar página en cada llamado.
- Buen diseño OO.
- Contrato mínimo GET/POST.

Entregables

- Repositorio GitHub
- README con pruebas
- Video corto mostrando funcionamiento

Referencias útiles

- https://www.w3schools.com/js/js_json_syntax.asp
- https://docs.oracle.com/javase/tutorial/reflect/index.html
- https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/package-summary.html
- https://docs.oracle.com/en/java/javase/11/docs/api/index.html

---

Ayudas

1. Cliente JS asincrónico

```
<!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>

</body>
</html>
```

---

2. Servidor HTTP mínimo que responde múltiples solicitudes

```
import java.net.*;
import java.io.*;

public class HttpServer {
  public static void main(String[] args) throws IOException {
   ServerSocket serverSocket = null;
   try {
      serverSocket = new ServerSocket(36000);
   } catch (IOException e) {
      System.err.println("Could not listen on port: 35000.");
      System.exit(1);
   }

   Socket clientSocket = null;
```

```
    try {
        System.out.println("Listo para recibir ...");
        clientSocket = serverSocket.accept();
    } catch (IOException e) {
        System.err.println("Accept failed.");
        System.exit(1);
    }
    PrintWriter out = new PrintWriter(
                          clientSocket.getOutputStream(), true);
    BufferedReader in = new BufferedReader(
                          new InputStreamReader(clientSocket.getInputStream()));
    String inputLine, outputLine;
    while ((inputLine = in.readLine()) != null) {
       System.out.println("Recibí: " + inputLine);
       if (!in.ready()) {break; }
    }
    outputLine = "HTTP/1.1 200 OK\r\n"
         + "Content-Type: text/html\r\n"
          + "\r\n"
          + "<!DOCTYPE html>\n"
          + "<html>\n"
          + "<head>\n"
          + "<meta charset=\"UTF-8\">\n"
          + "<title>Title of the document</title>\n"
          + "</head>\n"
          + "<body>\n"
          + "<h1>Mi propio mensaje</h1>\n"
          + "</body>\n"
          + "</html>\n";
     out.println(outputLine);
     out.close();
     in.close();
     clientSocket.close();
     serverSocket.close();
  }
}
```

---

3. 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);

        int responseCode = con.getResponseCode();
        System.out.println("GET Response Code :: " + responseCode);

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

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

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