

**Apertura:** jueves, 22 de febrero de 2024, 13:15

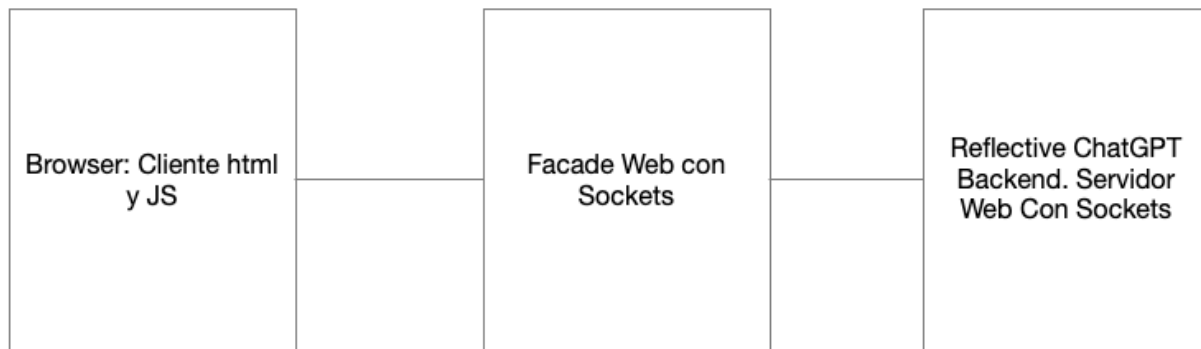
**Cierre:** jueves, 22 de febrero de 2024, 16:10

## REQUERIMIENTOS

No PUEDE revisar código en internet ni código antiguo que tenga en el computador. Use solo el código especificado en este enunciado.

No saque copias del enunciado.

Diagrama de Arquitectura:



## DESCRIPCIÓN:

Usted debe construir un "**Reflective ChatGPT**". La solución consta de un servidor backend que responde a solicitudes HTTP POST y/o GET de la Facade, un Servidor Facade que responde a solicitudes HTTP POST y/o GET del cliente y un cliente Html+JS que envía los comandos y muestra las respuestas. El api permite explorar clases del API de java. Cuando el usuario solicita información de una clase el chat le responde con el nombre de la clase, la lista de los campos declarados en la clase y la lista de los métodos declarados en la clase. Además el API debe permitir invocar y mostrar la salida de métodos estáticos con 0, 1 o 2 parámetros. Los parámetros de entrada pueden ser numéricos o Strings.

## LOS COMANDOS QUE SOPORTA EL CHAT SON LOS SIGUIENTES:

1. `Class([class name])`: Retorna una lista de campos declarados y métodos declarados
2. `invoke([class name],[method name])`: retorna el resultado de la invocación del método. Ejemplo: `invoke(java.lang.System, getenv)`.
3. `unaryInvoke([class name],[method name],[paramtype],[param value])`: retorna el resultado de la invocación del método. `paramtype = int | double | String`.

Ejemplos:

- `unaryInvoke(java.lang.Math, abs, int, 3)`

- `unaryInvoke(java.lang.Integer, valueOf, String, "3")`

3. `binaryInvoke([class name],[method name],[paramtype 1],[param value], [paramtype 1],[param value],):` retorna el resultado de la invocación del método. `paramtype = int | double | String`. Ejemplos:

- `binaryInvoke(java.lang.Math, max, double, 4.5, double, -3.7)`

- `binaryInvoke(java.lang.Integer, add, int, 6, int, -3)`

4. El chat solo soporta invocación de métodos estáticos.

Debe usar sockets solamente no puede usar ni spark ni spring.

## ARQUITECTURA:

- La aplicación tendrá tres componentes distribuidos: Una fachada de servicios, un servicio de backend, y un cliente web (html +js).
- Los servicios de la fachada y del backend deben estar desplegados en máquinas virtuales diferentes.
- El cliente es un cliente web que usa html y js. Se descarga desde un servicio en la fachada (Puede entregar el cliente directamente desde un método no es necesario que lo lea desde el disco).
- La comunicación se hace usando http y las respuestas de los servicios son en formato JSON.
- Los llamados al servicio de fachada desde el cliente deben ser asíncronos usando el mínimo JS posible. No actualice la página en cada llamado, solo el resultado.
- Los retornos deben estar en formato JSON o TEXTO.
- El diseño de los servicios WEB debe tener en cuenta buenas prácticas de diseño OO.
- Despliegue los servicios en máquinas virtuales separadas.
- El API de la fachada será
  - [url de la app]/cliente : Este servicio entrega el cliente web en formato html + js.
  - [url de la app]/consulta?comando=[comando con parámetros separados por coma entre paréntesis] : retorna el valor solicitado en formato JSON
  - Ejemplo: `http://localhost:35000/consulta?comando=binaryInvoke(java.lang.Math, max, double, 4.5, double, -3.7)`
- El API de la calculadora será
  - [url del backend]/comprefflex?comando =[comando con parámetros separados por coma entre paréntesis] :retorna el valor solicitado en formato JSON
  - Ejemplo: `http://localhost:45000/comprefflex?comando=binaryInvoke(java.lang.Math, max, double, 4.5, double, -3.7)`
- Asegúrese de retornar los encabezados correctos en HTTP y de responder mensajes válidos de HTTP ante solicitudes inesperadas

**Sugerencia realice la implementación de manera incremental. Haga commits regulares.**

Entregue todo en GIT HUB

## PÁGINAS QUE PUEDE ABRIR:

[https://www.w3schools.com/js/js\\_json\\_syntax.asp](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. 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>

```

2. Cómo hacer un servidor web mínimo que funcione correctamente:

- a. Hacer que el servidor responda múltiples solicitudes
- b. Asegurarse que la salida sea una salida http válida

```
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";
```

d. Código del servidor http



```

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 =
            "<!DOCTYPE html>" +
            "<html>" +
            "<head>" +
            "<meta charset=\"UTF-8\">" +
            "<title>Title of the document</title>\n" +
            "</head>" +
            "<body>" +
            "<h1>Mi propio mensaje</h1>" +
            "</body>" +
            "</html>";
        out.println(outputLine);
        out.close();
        in.close();
        clientSocket.close();
        serverSocket.close();
    }
}

```

3. Para 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=Q1QZFYJQ21K7C6XM";

    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");
}
```

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 23 minutos 39 segundos antes
Última modificación	jueves, 22 de febrero de 2024, 15:46
Texto en línea	<div><div></div><div><a href="https://github.com/ELS4NTA/AREP-PARCIAL-T1.git">https://github.com/ELS4NTA/AREP-PARCIAL-T1.git</a></div></div>
Comentarios de la entrega	<div><div></div><div>Comentarios (0)</div></div>

## CRITERIOS DE CALIFICACIÓN

Entrega en GitHub de manera ordenada y clara	No cumple <i>0 puntos</i>	Deficiente <i>1 puntos</i>	Aceptable <i>2 puntos</i>	Bueno <i>4 puntos</i>	
Infraestructura básica (Chat) (Fachada y Cliente)	No cumple <i>0 puntos</i>	Deficiente <i>1 puntos</i>	Aceptable <i>2 puntos</i>	Bueno <i>3 puntos</i>	Excelente <i>4 puntos</i>
Class	No cumple <i>0 puntos</i>	Parece implementarlo <i>1 puntos</i>	Funciona <i>2 puntos</i>	Diseño robusto <i>4 puntos</i>	
Invoke	No cumple <i>0 puntos</i>	Parece implementarlo <i>1 puntos</i>	Funciona <i>2 puntos</i>	Diseño robusto <i>4 puntos</i>	
Invoke unary or y/obinary	No cumple <i>0 puntos</i>	Parece implementarlo <i>1 puntos</i>	Funciona <i>2 puntos</i>	Diseño robusto <i>4 puntos</i>	

## COMENTARIO

Calificación	40,00 / 50,00
Calificado sobre	domingo, 3 de marzo de 2024, 19:49
Calificado por	 LUIS DANIEL BENAVIDES NAVARRO

## DESGLOSE DE LA CALIFICACIÓN

Entrega en GitHub de manera ordenada y clara	No cumple <i>0 puntos</i>	Deficiente <i>1 puntos</i>	Aceptable <i>2 puntos</i>	Bueno <i>4 puntos</i>	
Infraestructura básica (Chat) (Fachada y Cliente)	No cumple <i>0 puntos</i>	Deficiente <i>1 puntos</i>	Aceptable <i>2 puntos</i>	Bueno <i>3 puntos</i>	Excelente <i>4 puntos</i>
Class	No cumple <i>0 puntos</i>	Parece implementarlo <i>1 puntos</i>	Funciona <i>2 puntos</i>	Diseño robusto <i>4 puntos</i>	
Invoke	No cumple <i>0 puntos</i>	Parece implementarlo <i>1 puntos</i>	Funciona <i>2 puntos</i>	Diseño robusto <i>4 puntos</i>	
Invoke unary or y/obinary	No cumple <i>0 puntos</i>	Parece implementarlo <i>1 puntos</i>	Funciona <i>2 puntos</i>	Diseño robusto <i>4 puntos</i>	



## 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](mailto:contactocc@escuelaing.edu.co)

Escuela Colombiana de Ingeniería Julio Garavito