Enunciado Parcial AREP 2022-2

Enunciado

Diseñé, construya y despliegue un aplicación web para identificar si una cadena es una palíndromo o no. El programa debe estar desplegado en tres máquinas virtuales de EC2 de AWS como se describe abajo. Las tecnologías usadas en la solución deben ser maven, git, github, maven, sparkjava, html5, y js. No use liberías adicionales.

No copie código existente ni suyo ni de otras fuentes. todo el entregable debe ser original creado en la clase por usted.

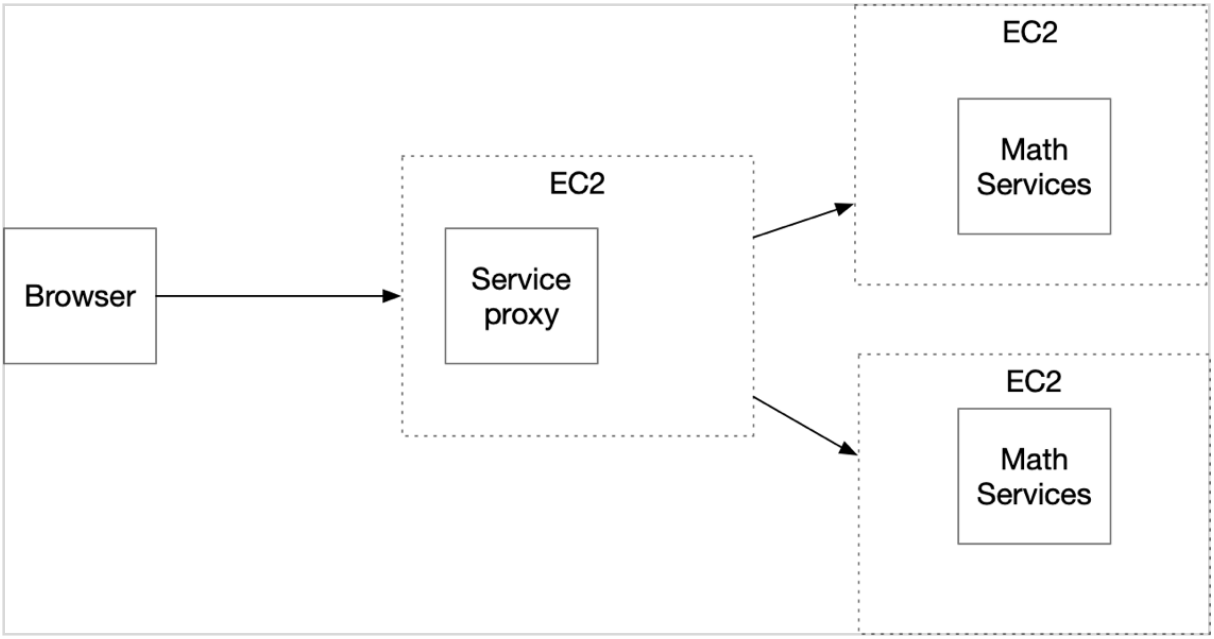
Solo use la documentación autorizada por el profesor.

Documentación autorizada:

- https://docs.oracle.com/javase/8/docs/api/
- https://sparkjava.com/
- https://docs.aws.amazon.com/corretto/latest/corretto-8-ug/amazon-linux-install.html
- AWS Academy
- AWS

Problema:

Diseñe un prototipo de sistema de microservicios que tenga un servicio (En la figura se representa con el nombre Math Services) para detectar si una cadena es un palíndromo o no. El servicio de palíndromos debe estar desplegado en almenos dos instancias virtuales de EC2. Adicionalmente, debe implementar un service proxy que reciba las solicitudes de llamdo desde los clientes y se las delega a las dos instancias del servicio de palíndromo usando un algoritmo de round-robin. El proxy deberá estar desplegado en otra máquina EC2. Asegúrese que se pueden configurar las direcciones y puertos de las instancias del servicio en el proxy usando variables de entorno del sistema operativo. Finalmente, construya un cliente Web mínimo con un formulario que reciba la cadena y de manera asíncrona invoke el servicio único en el PROXY. El cliente debe ser escrito en HTML y JS.



Palíndromo:

Palíndromo Es una cadena que se lee igual si se lee de izquierda a derecha, que si se lee de derecha a izquierda.

Por ejemplo, “adbaeea” no es un palíndromo, pero en cambio, “aadbaeeabdaa” si es una palíndormo.

Otro ejemplo, “12785” no es una palíndromo, pero “127858721” si es una palíndromo.

Detalles adicionales de la arquitectura y del API

Implemente los servicios para responder al método de solicitud HTTP GET or POST. Deben usar el nombre de la función especificado y el parámetro debe ser pasado en la variable de query con nombre “value”.

El proxy debe delegar el llamado a los servicios de backend. El proxy y los servicios se deben implementar en Java usando Spark.

Ejemplo de un llamado:

EC2

https://amazonxxx.x.xxx.x.xxx:{port}/espalindromo?value=12344321

Salida. El formato de la salida y la respuesta debe ser un JSON con el siguiente formato

{

“operation”: “palíndromo”,

"input": "12344321",

"output": "Si es palíndromo"

}

Entregable:

1. Proyecto actualizado en github (un solo repositorio)
2. Descripción del proyecto enel README con pantallazos que muestren el funcionamiento.
3. Descripción de como correrlo en EC2.
4. Video de menos de un minuto del funcionamiento (lo puede tomar con el celular una vez funcione)

En el campo de texto escriba la direción de su repositorio GITHUB.

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>
```

3. Aplicación Spark mínima

```
public class SparkWebServer {

  public static void main(String… args){
      port(getPort());
      get("hello", (req,res) → "Hello Docker!");
  }

  private static int getPort() {
    if (System.getenv("PORT") != null) {
      return Integer.parseInt(System.getenv("PORT"));
    }
    return 4567;
  }

}
```

4. Build con dependencias

```
<!– build configuration –>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-dependency-plugin</artifactId>
        <version>3.0.1</version>
        <executions>
          <execution>
            <id>copy-dependencies</id>
            <phase>prepare-package</phase>
            <goals>
```

```
                <goal>copy-dependencies</goal>
              </goals>
            </execution>
          </executions>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-assembly-plugin</artifactId>
          <executions>
            <execution>
              <phase>package</phase>
              <goals>
                <goal>single</goal>
              </goals>
              <configuration>
                <archive>
                  <manifest>
                    <mainClass>
                      co.edu.escuelaing.securityprimerliveg2.HelloService
                    </mainClass>
                  </manifest>
                </archive>
                <descriptorRefs>
                  <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
              </configuration>
            </execution>
          </executions>
        </plugin>
      </plugins>
    </build>
```