

# Despliegue del Api biblioteca en Oracle

## Cambios del código original

Modifico ClienteAPI.java para que funcione en el servidor de Oracle con OkHttpClient y ServerSocket.

```
package cliente;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.Response;

public class ClienteAPI {

    private static final String API_KEY = "3b5faf6c57b79d39e76351663a5e0cbe";
    private static final String BASE_URL = "https://api.themoviedb.org/3";
    private static final OkHttpClient client = new OkHttpClient();

    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(8080)) {
            System.out.println("Servidor iniciado en el puerto 8080 ...");

            while (true) {
                Socket clientSocket = serverSocket.accept();
                new Thread(() -> handleRequest(clientSocket)).start();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private static void handleRequest(Socket clientSocket) {
        try (
            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            OutputStream outputStream = clientSocket.getOutputStream()
        ) {
            String requestLine = in.readLine();
            System.out.println("Solicitud recibida: " + requestLine);

            if (requestLine != null) {
                String[] parts = requestLine.split(" ");
                String method = parts[0];
                String path = parts[1];

                if (method.equals("OPTIONS")) {
                    sendResponse(outputStream, "", 200);
                } else if (method.equals("GET")) {
                    if (path.startsWith("/search")) {
                        String query = path.split("\\?")[1].split("=")[1];
                        String response = searchMovieByTitle(query);
                        sendResponse(outputStream, response, 200);
                    } else if (path.startsWith("/popular")) {

```

```

        String response = listPopularMovies();
        sendResponse(outputStream, response, 200);
    } else if (path.startsWith("/by-genre")) {
        String genreId = path.split("\\?")[1].split("=")[1];
        String response = listMoviesByGenre(genreId);
        sendResponse(outputStream, response, 200);
    } else {
        sendResponse(outputStream, "Ruta no encontrada", 404);
    }
} else {
    sendResponse(outputStream, "Método no permitido", 405);
}
}
clientSocket.close();
} catch (Exception e) {
    e.printStackTrace();
}
}

// Buscar películas por título
private static String searchMovieByTitle(String title) throws Exception {
    String encodedTitle = URLEncoder.encode(title, StandardCharsets.UTF_8.toString());
    String url = BASE_URL + "/search/movie?api_key=" + API_KEY + "&query=" + encodedTitle;
    return makeRequest(url);
}

// Obtener películas populares
private static String listPopularMovies() throws Exception {
    String url = BASE_URL + "/movie/popular?api_key=" + API_KEY;
    return makeRequest(url);
}

// Buscar por género
private static String listMoviesByGenre(String genreId) throws Exception {
    String url = BASE_URL + "/discover/movie?api_key=" + API_KEY + "&with_genres=" + genreId;
    return makeRequest(url);
}

// Realizar la solicitud HTTP
private static String makeRequest(String url) throws Exception {
    Request request = new Request.Builder().url(url).build();
    try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            return "Error en la solicitud: " + response;
        }
        return response.body().string();
    }
}

private static void sendResponse(OutputStream outputStream, String body, int statusCode) {
    try {
        String headers = "HTTP/1.1 " + statusCode + " OK\r\n" +
            "Content-Type: application/json\r\n" +
            "Access-Control-Allow-Origin: *\r\n" +
            "Access-Control-Allow-Methods: GET, POST, OPTIONS\r\n" +
            "Access-Control-Allow-Headers: Content-Type\r\n" +
            "Content-Length: " + body.getBytes().length + "\r\n" +
            "\r\n";

        outputStream.write(headers.getBytes(StandardCharsets.UTF_8));

        outputStream.write(body.getBytes(StandardCharsets.UTF_8));
        outputStream.flush();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

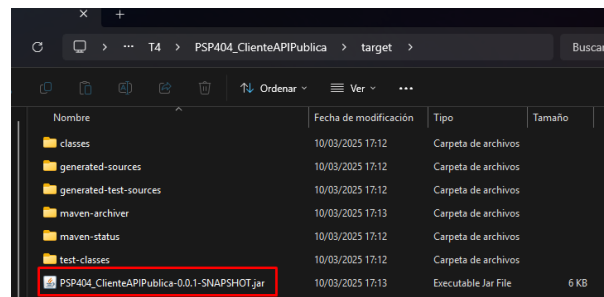
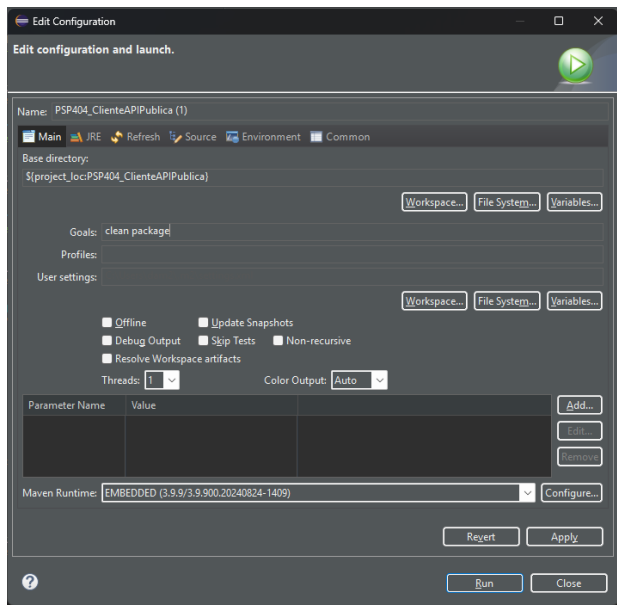
```
}  
}  
}
```

Y también el pom.xml para que se importen los pluguins al compilar:

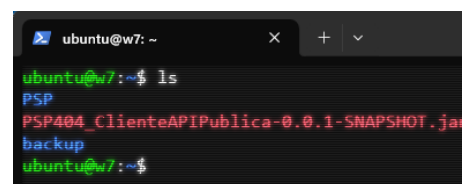
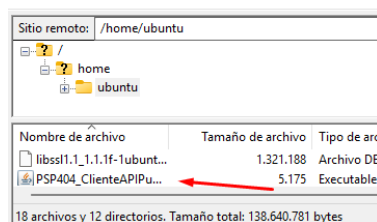
```
<project xmlns="http://maven.apache.org/POM/4.0.0"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
  <modelVersion>4.0.0</modelVersion>  
  <groupId>0</groupId>  
  <artifactId>PSP404_ClienteAPIPublica</artifactId>  
  <version>0.0.1-SNAPSHOT</version>  
  
  <dependencies>  
    <dependency>  
      <groupId>com.squareup.okhttp3</groupId>  
      <artifactId>okhttp</artifactId>  
      <version>4.10.0</version>  
    </dependency>  
  
    <dependency>  
      <groupId>com.google.code.gson</groupId>  
      <artifactId>gson</artifactId>  
      <version>2.10.1</version>  
    </dependency>  
  </dependencies>  
  
  <build>  
    <plugins>  
      <plugin>  
        <groupId>org.apache.maven.plugins</groupId>  
        <artifactId>maven-shade-plugin</artifactId>  
        <version>3.2.4</version>  
        <executions>  
          <execution>  
            <phase>package</phase>  
            <goals>  
              <goal>shade</goal>  
            </goals>  
            <configuration>  
              <transformers>  
                <transformer  
  
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">  
  
                </transformer>  
              </transformers>  
            </configuration>  
          </execution>  
        </executions>  
      </plugin>  
    </plugins>  
  </build>  
</project>
```

## Creación del servicio

Comienzo compilando para obtener el .jar



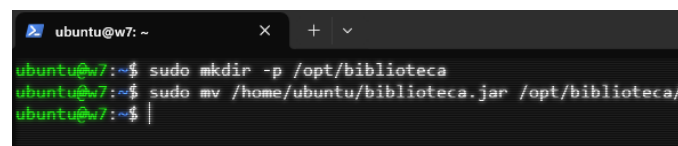
Lo subo a Oracle con Filezilla



Lo renombro a biblioteca.jar para usarlo más comodamente.



Lo movemos a un directorio dedicado a servicios, como /opt/.



Creamos el archivo del servicio para systemd:

`sudo vim /etc/systemd/system/biblioteca.service`

Con el siguiente código:

[Unit]

Description=Servicio API Biblioteca  
After=network.target

[Service]

User=ubuntu

ExecStart=/usr/bin/java -jar /opt/biblioteca/biblioteca.jar --server.port=8080  
--server.address=0.0.0.0

SuccessExitStatus=143

Restart=on-failure

RestartSec=10

WorkingDirectory=/opt/biblioteca

[Install]

WantedBy=multi-user.target

Y activamos el servicio:

```
ubuntu@w7:~$ sudo systemctl daemon-reload
ubuntu@w7:~$ sudo systemctl enable biblioteca.service
Created symlink /etc/systemd/system/multi-user.target.wants/biblioteca.service → /etc/systemd/system/biblioteca.service.
ubuntu@w7:~$ sudo systemctl start biblioteca.service
ubuntu@w7:~$ sudo systemctl status biblioteca.service
● biblioteca.service - Servicio API Biblioteca
   Loaded: loaded (/etc/systemd/system/biblioteca.service; enabled; preset: enabled)
   Active: activating (auto-restart) (Result: exit-code) since Mon 2025-03-10 16:32:59 UTC; 4s ago
     Process: 914713 ExecStart=/usr/bin/java -jar /opt/biblioteca/biblioteca.jar --server.port=8080 --server.address=0.0.0.0
    Main PID: 914713 (code=exited, status=217/USER)
      CPU: 0
lines 1 6/6 (FND)
```


Y ahora nos aseguramos que el puerto elegido, 8080, esté abierto tanto en Oracle como en iptables.

<input type="checkbox"/>	Yes	0.0.0.0/0	TCP	All	2017	TCP traffic for ports: 2017
<input type="checkbox"/>	Yes	0.0.0.0/0	TCP	All	5000	TCP traffic for ports: 5000
<input type="checkbox"/>	Yes	0.0.0.0/0	TCP	All	3000	TCP traffic for ports: 3000
<input type="checkbox"/>	Yes	0.0.0.0/0	TCP	All	8080	TCP traffic for ports: 8080
<input type="checkbox"/>	No	0.0.0.0/0	TCP	All	22	TCP traffic for ports: 22 SSH Remote Login Protocol

```
ubuntu@w7:~$ sudo iptables -I INPUT 5 -p tcp --dport 8080 -j ACCEPT
ubuntu@w7:~$ sudo iptables-save | sudo tee /etc/iptables/rules.v4
```

Y compruebo en el port checker que el puerto está abierto:

Online service Port check

 **Port check** – Tests if TCP port is opened on specified IP

IP address or host name:  Port number:

79.72.63.217:8080 port is **open**

Oscar Montero Hinojosa  
Programación de Servicios y Procesos

## Código del cliente

Creo un index.html con javascript con el siguiente código para acceder a los endpoints:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Cliente API Biblioteca</title>
</head>
<body>
  <h1>Cliente API Biblioteca</h1>

  <!-- Buscar película por título -->
  <div>
    <h2>Buscar película por título</h2>
    <input type="text" id="title" placeholder="Título de la película">
    <button onclick="searchMovie()">Buscar</button>
    <pre id="searchResult"></pre>
  </div>

  <!-- Obtener películas populares -->
  <div>
    <h2>Obtener películas populares</h2>
    <button onclick="getPopular()">Ver populares</button>
    <pre id="popularResult"></pre>
  </div>

  <!-- Obtener detalles por género -->
  <div>
    <h2>Buscar por género</h2>
    <input type="number" id="genreId" placeholder="ID del género">
    <button onclick="getByGenre()">Buscar</button>
    <pre id="genreResult"></pre>
  </div>

  <script>
    const BASE_URL = 'http://79.72.63.217:8080';

    // Buscar película por título
    async function searchMovie() {
      const title = document.getElementById('title').value;
      if (!title) {
        alert('Introduce el título de la película');
        return;
      }
      try {
        const response = await fetch(`${BASE_URL}/search?title=${
          encodeURIComponent(title)
        }`);
        if (!response.ok) throw new Error(`Error HTTP: ${response.status}`);

        const text = await response.text();
        if (!text) throw new Error('Respuesta vacía del servidor');

        const data = JSON.parse(text);
        document.getElementById('searchResult').innerText = JSON.stringify(data,
          null, 2);
      } catch (error) {
        console.error('Error:', error);
        alert(`Error al buscar película: ${error.message}`);
      }
    }
  </script>
</body>
</html>
```

```

    }
}

// Obtener películas populares
async function getPopular() {
    try {
        const response = await fetch(`${BASE_URL}/popular`);
        if (!response.ok) throw new Error(`Error HTTP: ${response.status}`);

        const text = await response.text();
        if (!text) throw new Error('Respuesta vacía del servidor');

        const data = JSON.parse(text);
        document.getElementById('popularResult').innerText = JSON.stringify(data,
null, 2);
    } catch (error) {
        console.error('Error:', error);
        alert(`Error al obtener películas populares: ${error.message}`);
    }
}

// Buscar por género
async function getByGenre() {
    const genreId = document.getElementById('genreId').value;
    if (!genreId) {
        alert('Introduce el ID del género');
        return;
    }
    try {
        const response = await fetch(`${BASE_URL}/by-genre?genreId=${genreId}`);
        if (!response.ok) throw new Error(`Error HTTP: ${response.status}`);

        const text = await response.text();
        if (!text) throw new Error('Respuesta vacía del servidor');

        const data = JSON.parse(text);
        document.getElementById('genreResult').innerText = JSON.stringify(data, null,
2);
    } catch (error) {
        console.error('Error:', error);
        alert(`Error al buscar por género: ${error.message}`);
    }
}

</script>
</body>
</html>

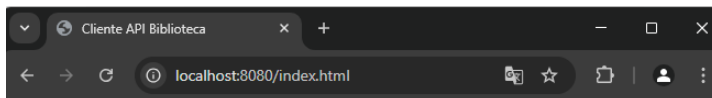
```

Y ahora con el siguiente comando en la carpeta del proyecto donde se encuentra el index.html abrimos un servidor local para simular que estamos en una web:

```
python3 -m http.server 8080
```

Y accedemos a <http://localhost:8080/index.html> y podremos probar la conexión.

# Resultado



## Cliente API Biblioteca

### Buscar película por título

### Obtener películas populares

### Buscar por género