**Por :** JUAN CAMILO ORTIZ MEDINA

**PARTE 1**

* Error : 301 Moved Permanently

Esta y todas las solicitudes futuras deben dirigirse al URI dado.

* Los otros codigos de error que existen son:

- 302 Found (Previously "Moved temporarily")

- 303 See Other

- 304 Not Modified

- 305 Use Proxy

- 306 Switch Proxy

- 307 Temporary Redirect

- 308 Permanent Redirect

* ¿Cuál es la diferencia entre los verbos GET y POST? ¿Qué otros tipos de peticiones existen?

|  |  |  |
| --- | --- | --- |
| **MÉTODO** | **CONCEPTO** | **OBSERVACIONES** |
| **GET** | GET lleva los datos de forma "visible" al cliente (navegador web). El medio de envío es la URL. Los datos los puede ver cualquiera. | Los datos son visibles por la URL, por ejemplo:  www.aprenderaprogramar.com/  action.php?nombre=pedro&apellidos1= gomez |
| **POST** | POST consiste en datos "ocultos" (porque el cliente no los ve) enviados por un formulario cuyo método de envío es post. Es adecuado para formularios. Los datos no son visibles. | La ventaja de usar POST es que estos datos no son visibles al usuario de la web. En el caso de usar get, el propio usuario podría modificar la URL escribiendo diferentes parámetros a los reales en su navegador, dando lugar a que la información tratada no sea la prevista. |
| PUT | Utilizado normalmente para actualizar contenidos, pero también pueden crearlos. | En caso de éxito devuelve 201 (created, en caso de que la acción haya creado un elemento) o 204 (no response, si el servidor no devuelve ningún contenido). |
| DELETE | Simplemente elimina un resource identificado en la URI | Si se elimina correctamente devuelve 200 junto con un body response, o 204 sin body. DELETE, al igual que PUT y GET, también es idempotente. |
| HEAD | Es idéntido a GET, pero el servidor no devuelve el contenido en el HTTP response. Cuando se envía un HEAD request, significa que sólo se está interesado en el código de respuesta y los headers HTTP, no en el propio documento. | Con este método el navegador puede comprobar si un documento se ha modificado, por razones de caching. Puede comprobar también directamente si el archivo existe. |

* Que es CURL?

El comando curl linux es una herramienta útil para hacer simulaciones de las acciones de los usuarios desde navegadores.

Soporta los protocolos FTP, FTPS, HTTP, HTTPS, TFTP, SCP, SFTP, Telnet, DICT, FILE y LDAP, entre otros.

El principal propósito y uso para cURL es automatizar transferencias de archivos o secuencias de operaciones no supervisadas. Es por ejemplo, una herramienta válida para simular las acciones de usuarios en un navegador web.

* Utilice ahora el parámetro -v y con el parámetro -I ¿Cuáles son las diferencias con los diferentes parámetros?

</html>%

**\* Rebuilt URL to: www.httpbin.org/**

**\* Trying 52.202.2.199...**

**\* TCP\_NODELAY set**

**\* Connected to www.httpbin.org (52.202.2.199) port 80 (#0)**

**> GET / HTTP/1.1**

**> Host: www.httpbin.org**

**> User-Agent: curl/7.58.0**

**> Accept: \*/\***

>

< HTTP/1.1 200 OK

< Date: Thu, 20 Feb 2020 13:44:31 GMT

< Content-Type: text/html; charset=utf-8

< Content-Length: 9593

< Connection: keep-alive

< Server: gunicorn/19.9.0

< Access-Control-Allow-Origin: \*

< Access-Control-Allow-Credentials: true

<

<!DOCTYPE html>

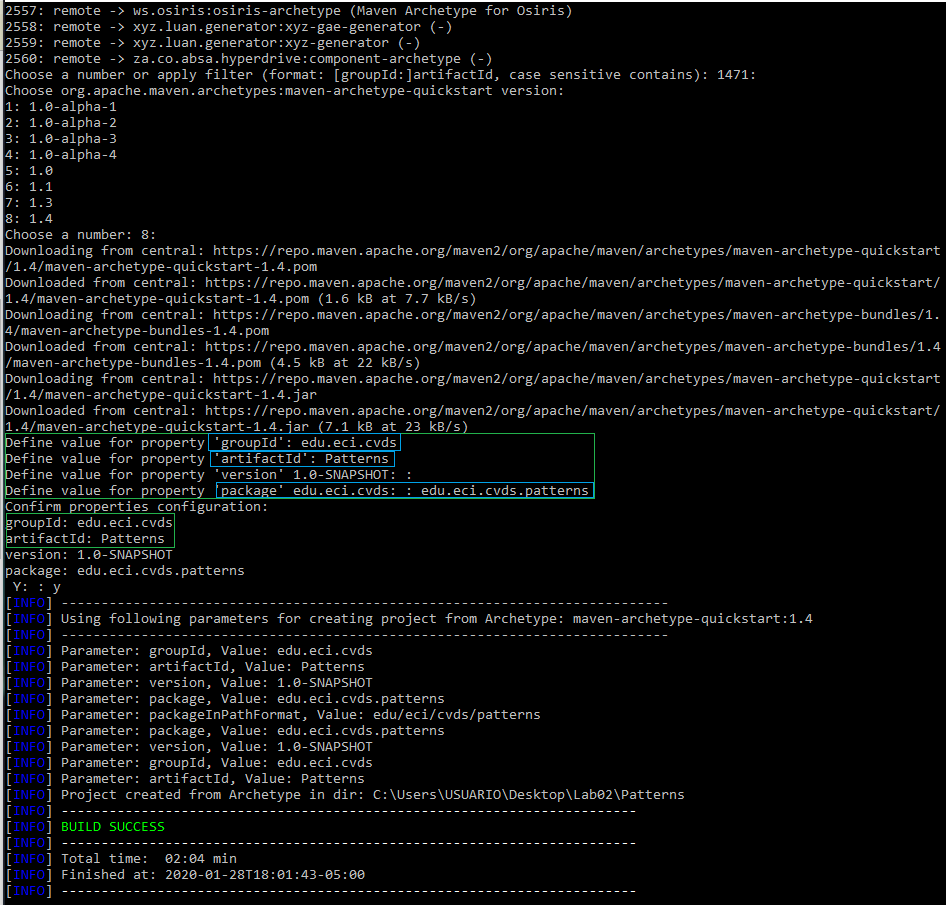
<html lang="en">

**NOTA :** El comando curl -v arroja más información (La parte en negrilla) que el comando curl -i.

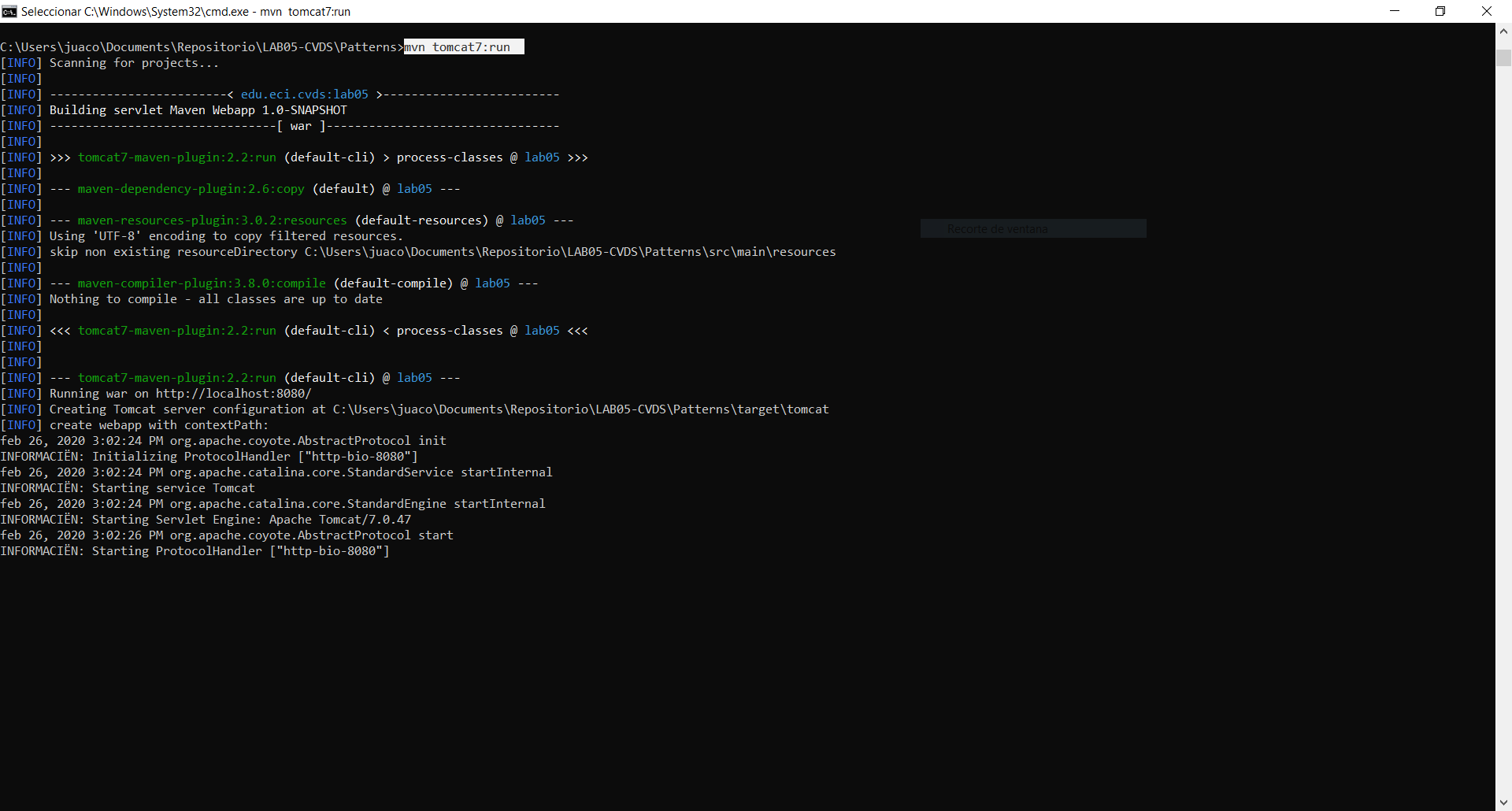
**PARTE 2**

* Cree un proyecto maven nuevo usando el arquetipo de aplicación Web estándar maven-archetype-webapp

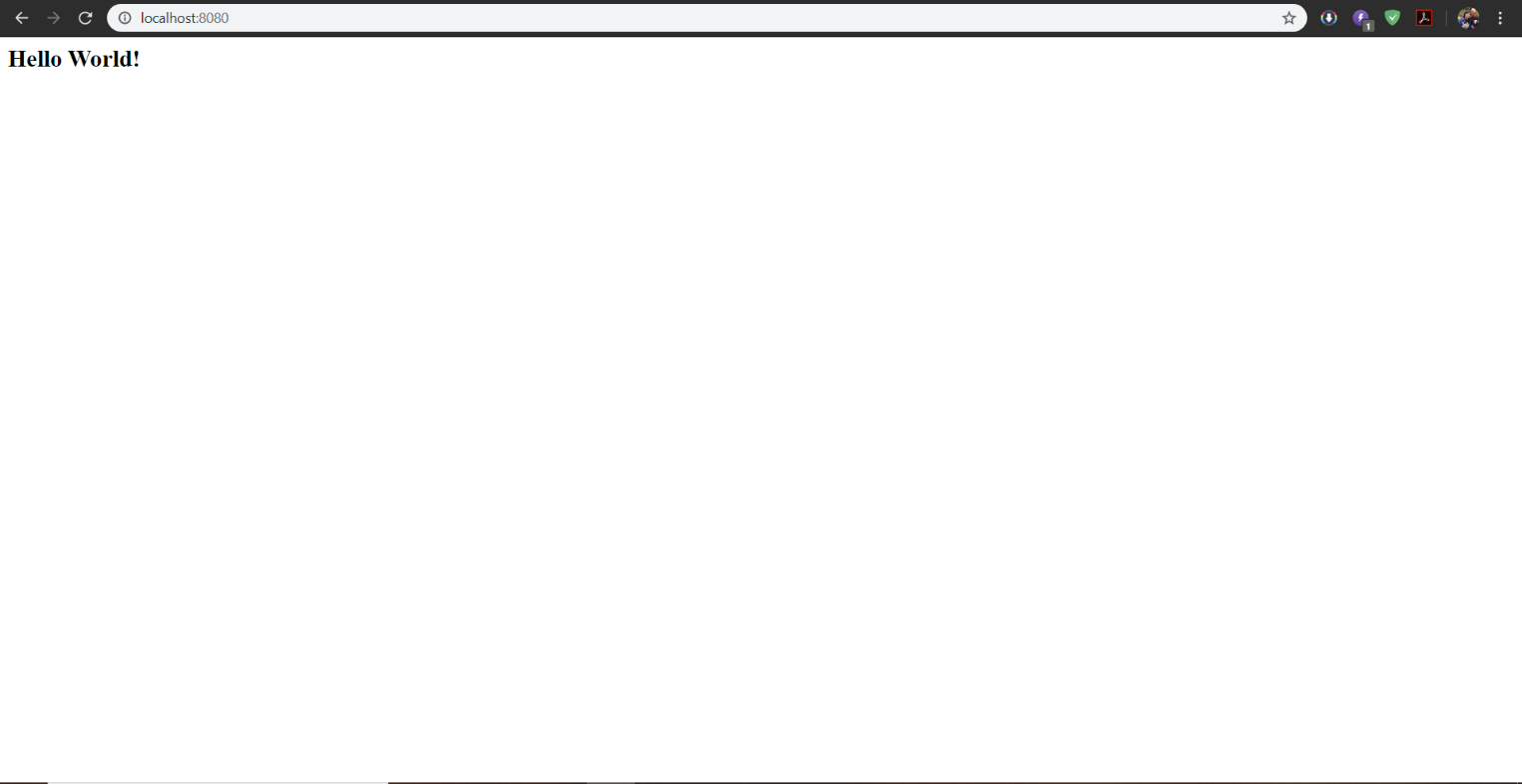
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DarchetypeArtifactId=maven-archetype-webapp -DarchetypeVersion=1.4



Activamos el Tomcat7:

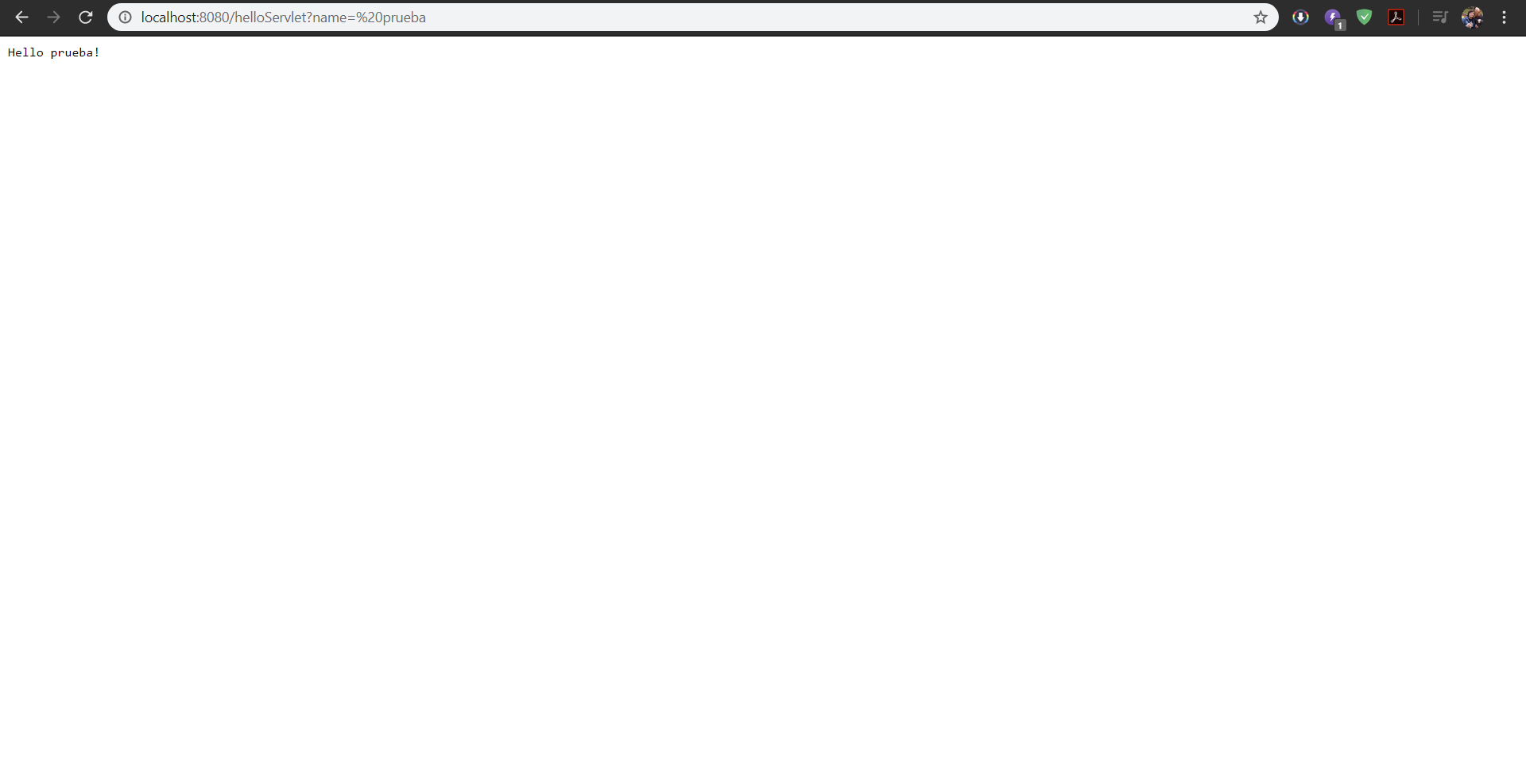


Obtenemos el mensaje de saludo.



* Observe que el Servlet ‘SampleServlet’ acepta peticiones GET, y opcionalmente, lee el parámetro ‘name’. Ingrese la misma URL, pero ahora agregando un parámetro GET (si no sabe como hacerlo, revise la documentación en http://www.w3schools.com/tags/ref\_httpmethods.asp).

<http://localhost:8080/helloServlet?name=prueba>



* En el navegador revise la dirección https://jsonplaceholder.typicode.com/todos/1. Intente cambiando diferentes números al final del path de la url.

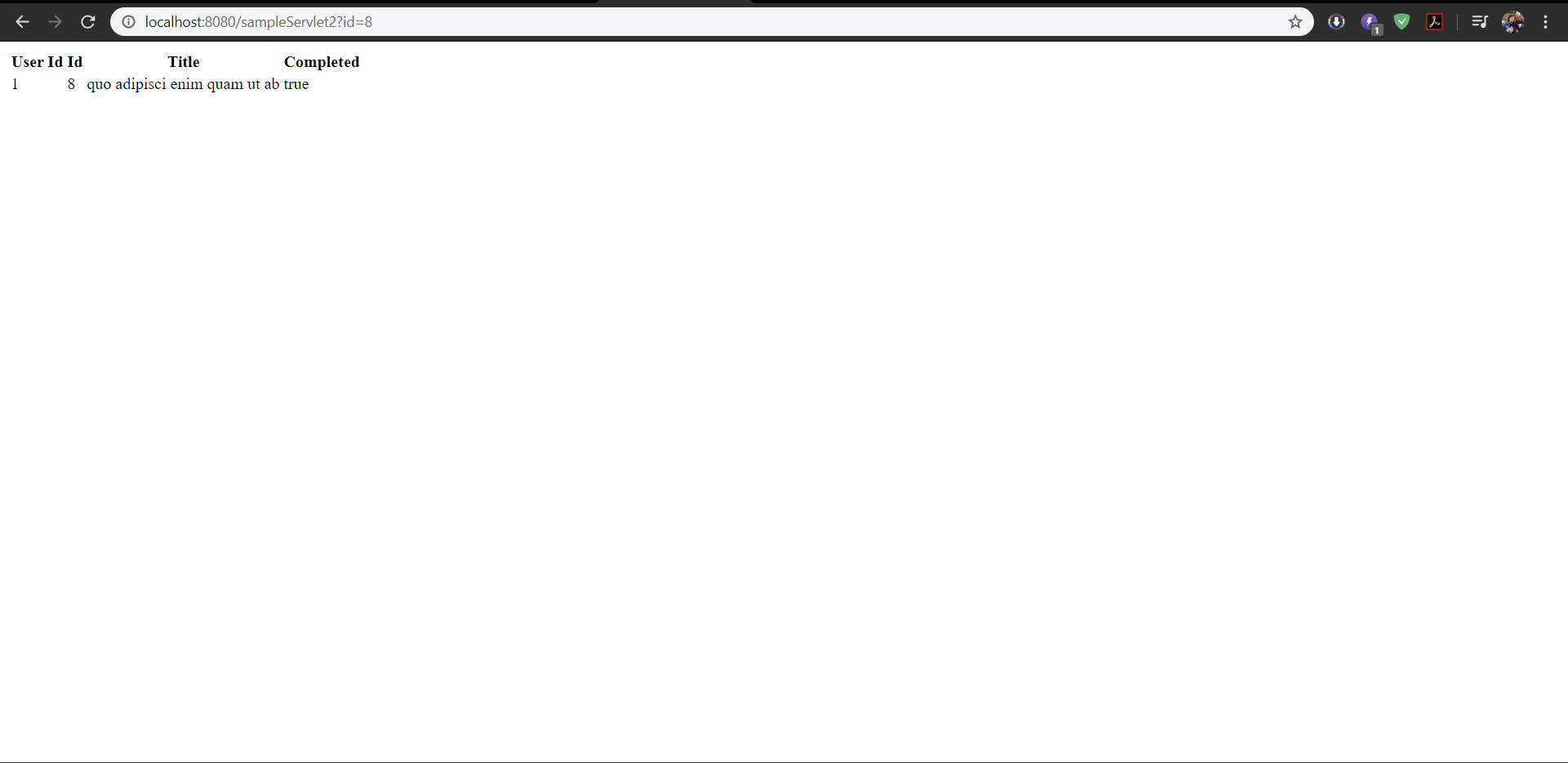
Caso 1 :



Caso 2 :



Caso 3:

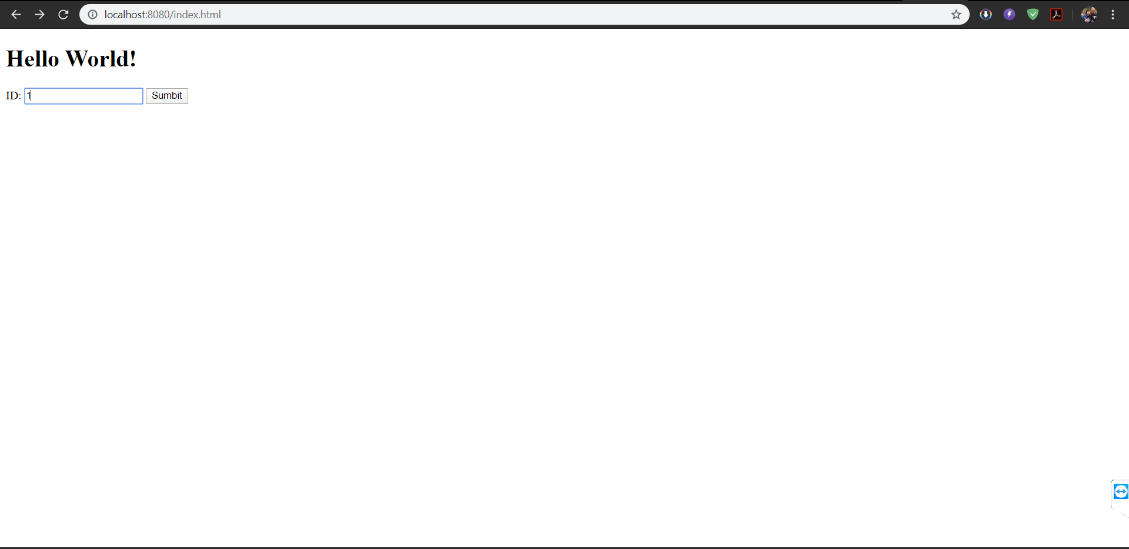


**PARTE 3**

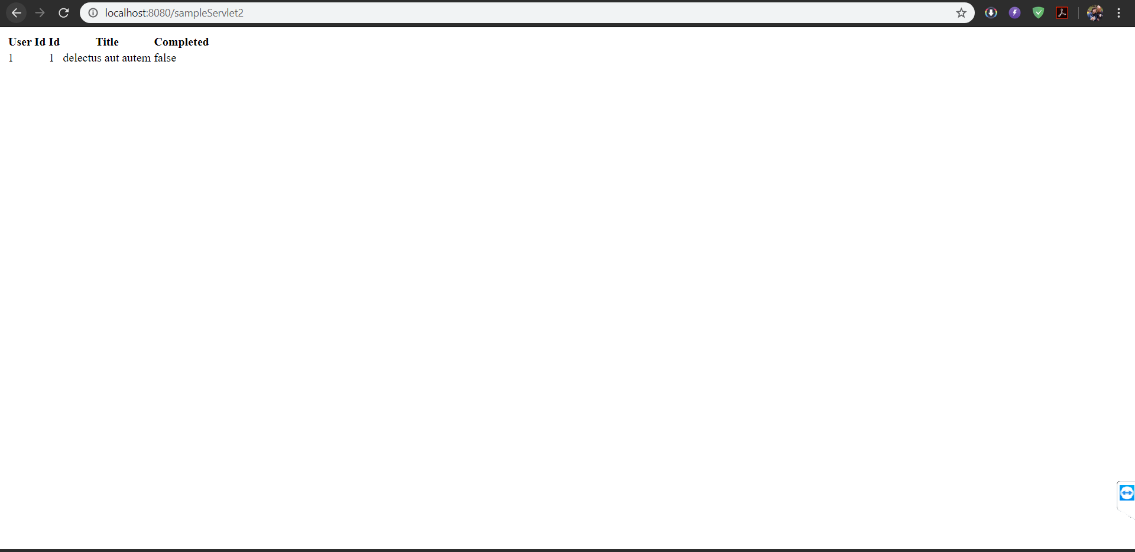
13.

* Si el item existe:
  + Responder con el código HTTP que equivale a ‘OK’ ([ver referencia anterior](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)), y como contenido de dicha respuesta, el código html correspondiente a una página con una tabla que tenga los detalles del item, usando la clase "Service" creada en el punto 10 par crear la tabla.

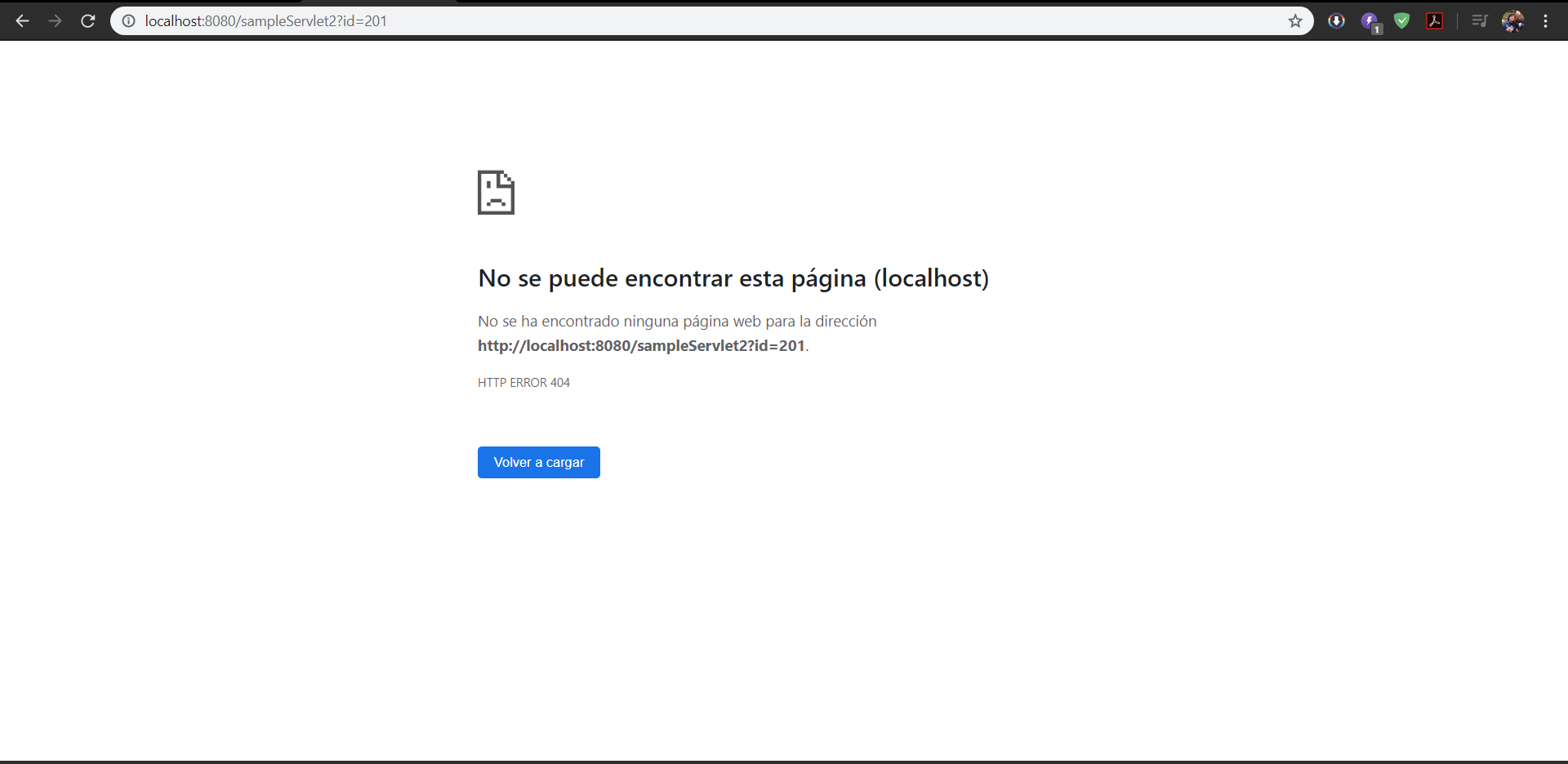
Paso 1:



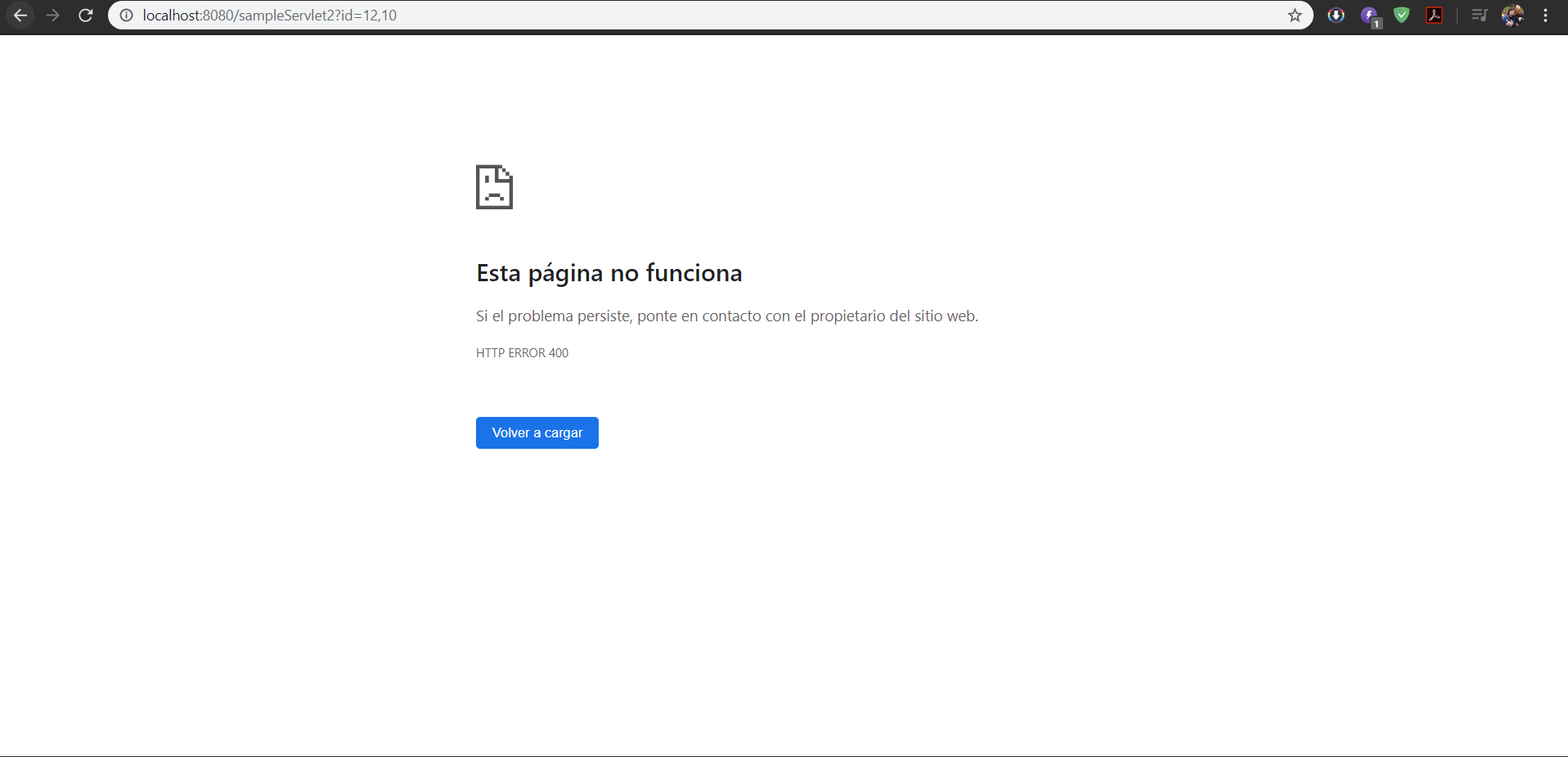
Paso 2:



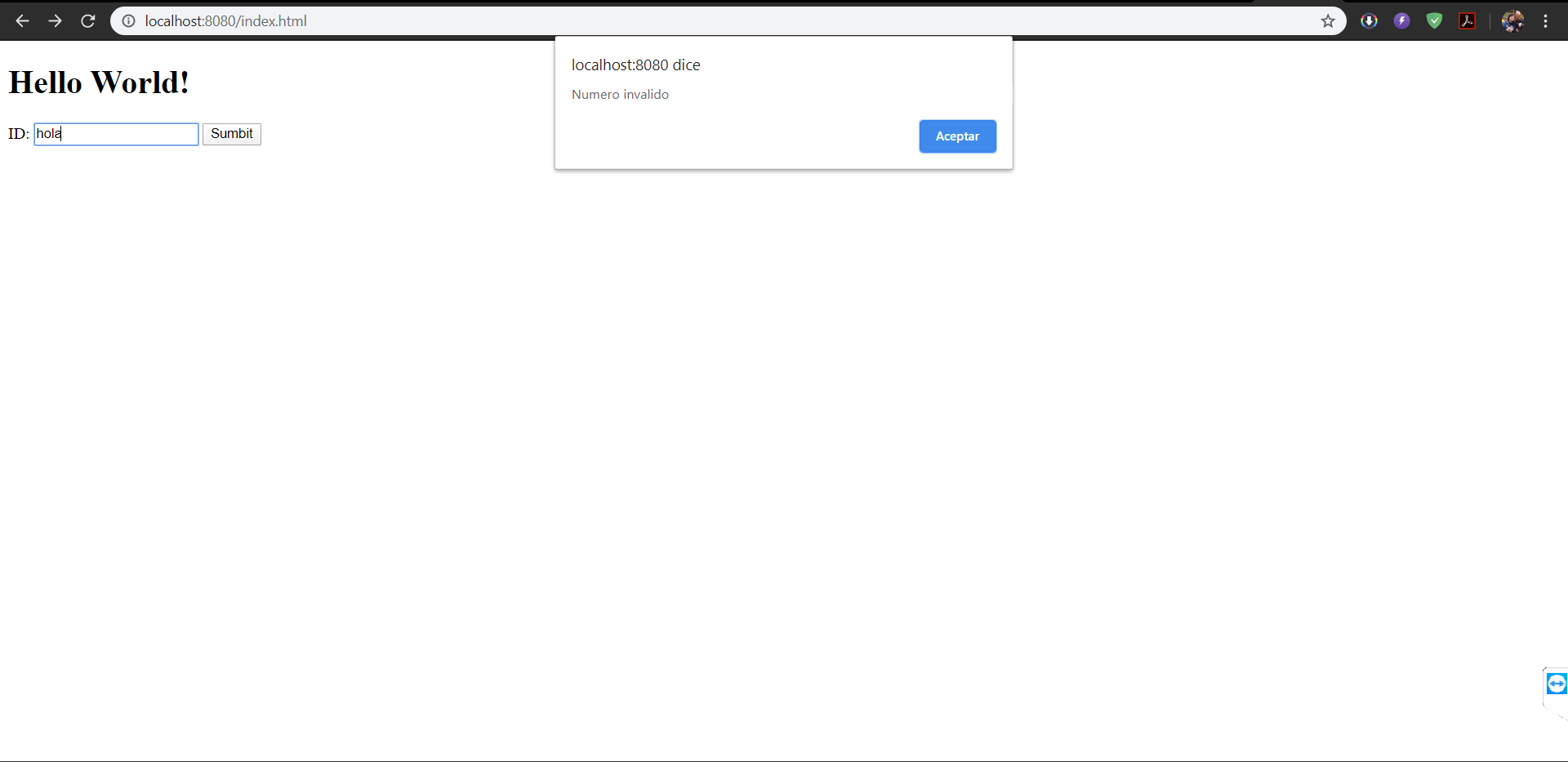
* Si el item no existe:
  + Responder con el código correspondiente a ‘no encontrado’, y con el código de una página html que indique que no existe un item con el identificador dado.



* + Si no se paso parámetro opcional, o si el parámetro no contiene un número entero, devolver el código equivalente a requerimiento inválido.



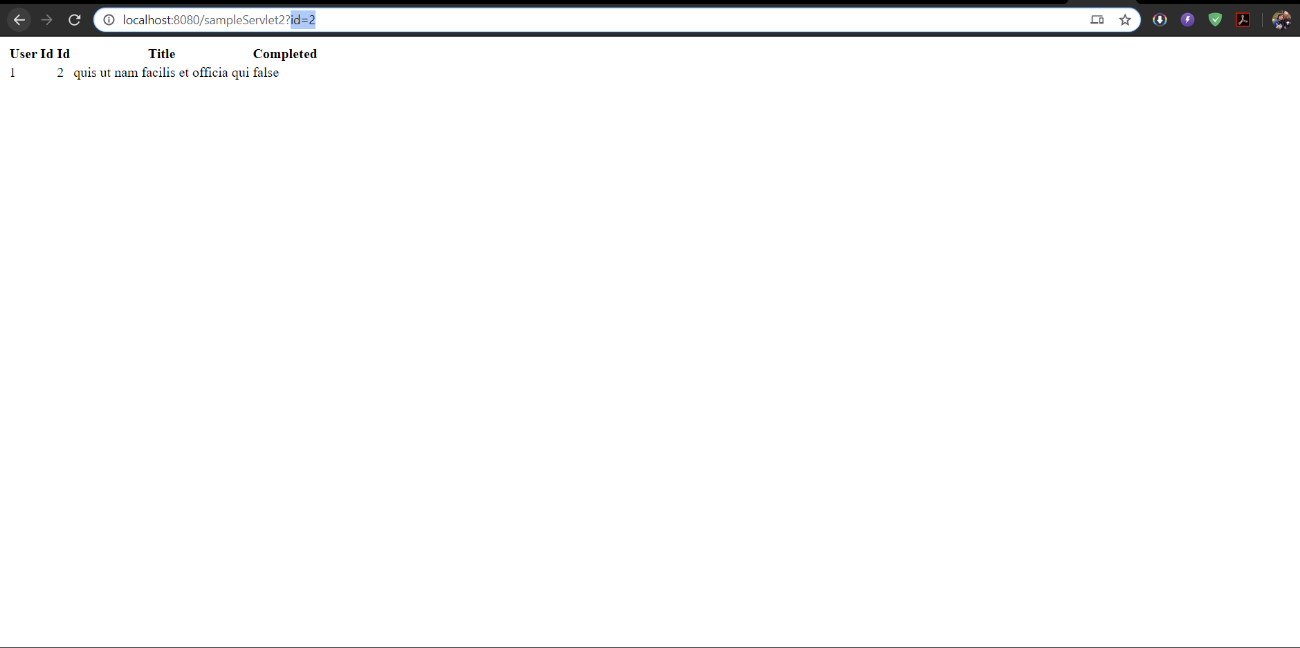
* + Si se genera la excepcion MalformedURLException devolver el código de error interno en el servidor
  + Para cualquier otra excepcion, devolver el código equivalente a requerimiento inválido.
* Revise [este ejemplo de validación de formularios con javascript](http://www.w3schools.com/js/js_validation.asp) y agruéguelo a su formulario, de manera que -al momento de hacer ‘submit’- desde el browser se valide que el valor ingresado es un valor numérico.



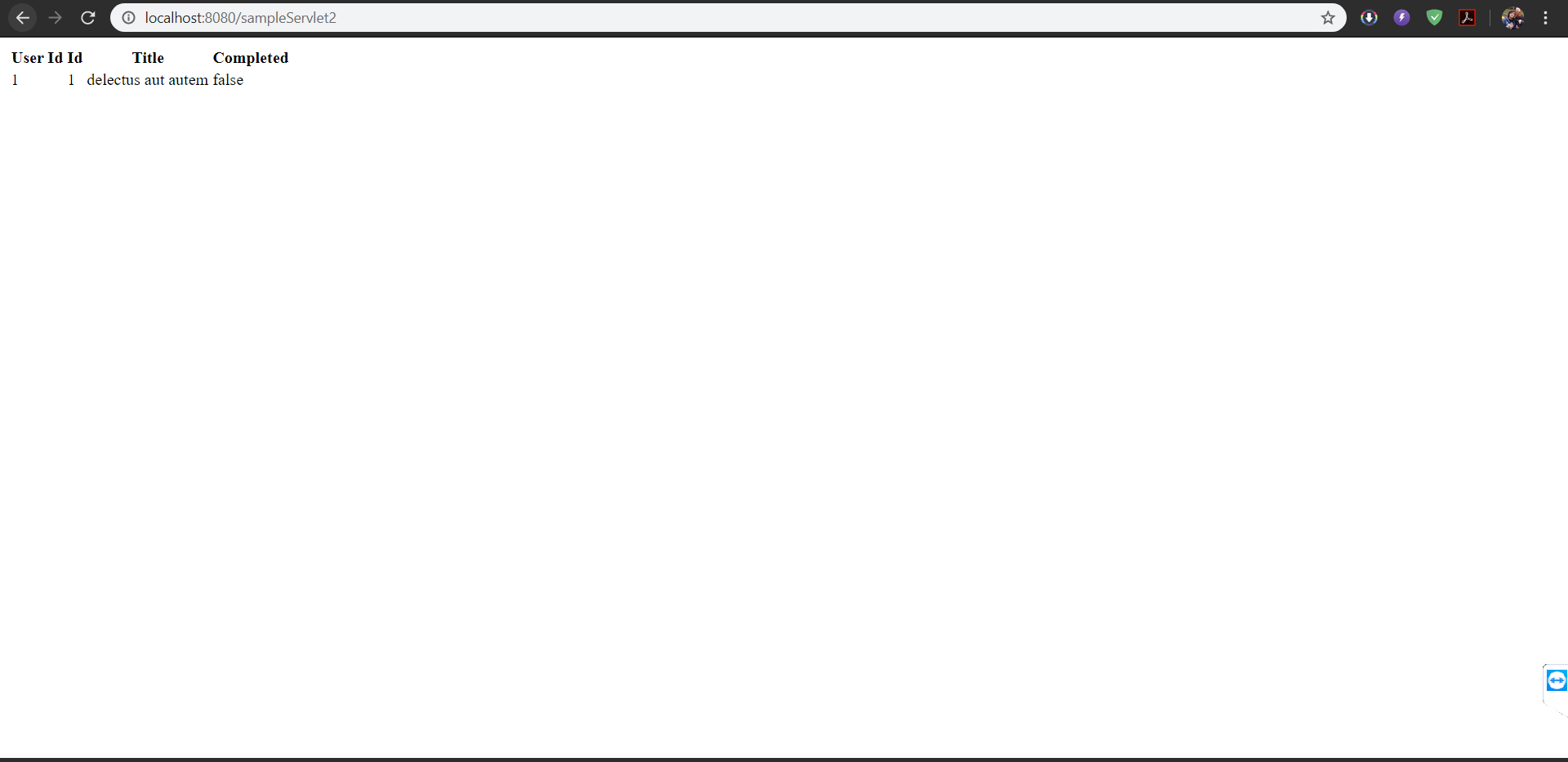
Cambie el formulario para que ahora en lugar de POST, use el método GET . Qué diferencia observa?

En el URL del Get se ve el valor de entrada y en el Post no.

**Get:**



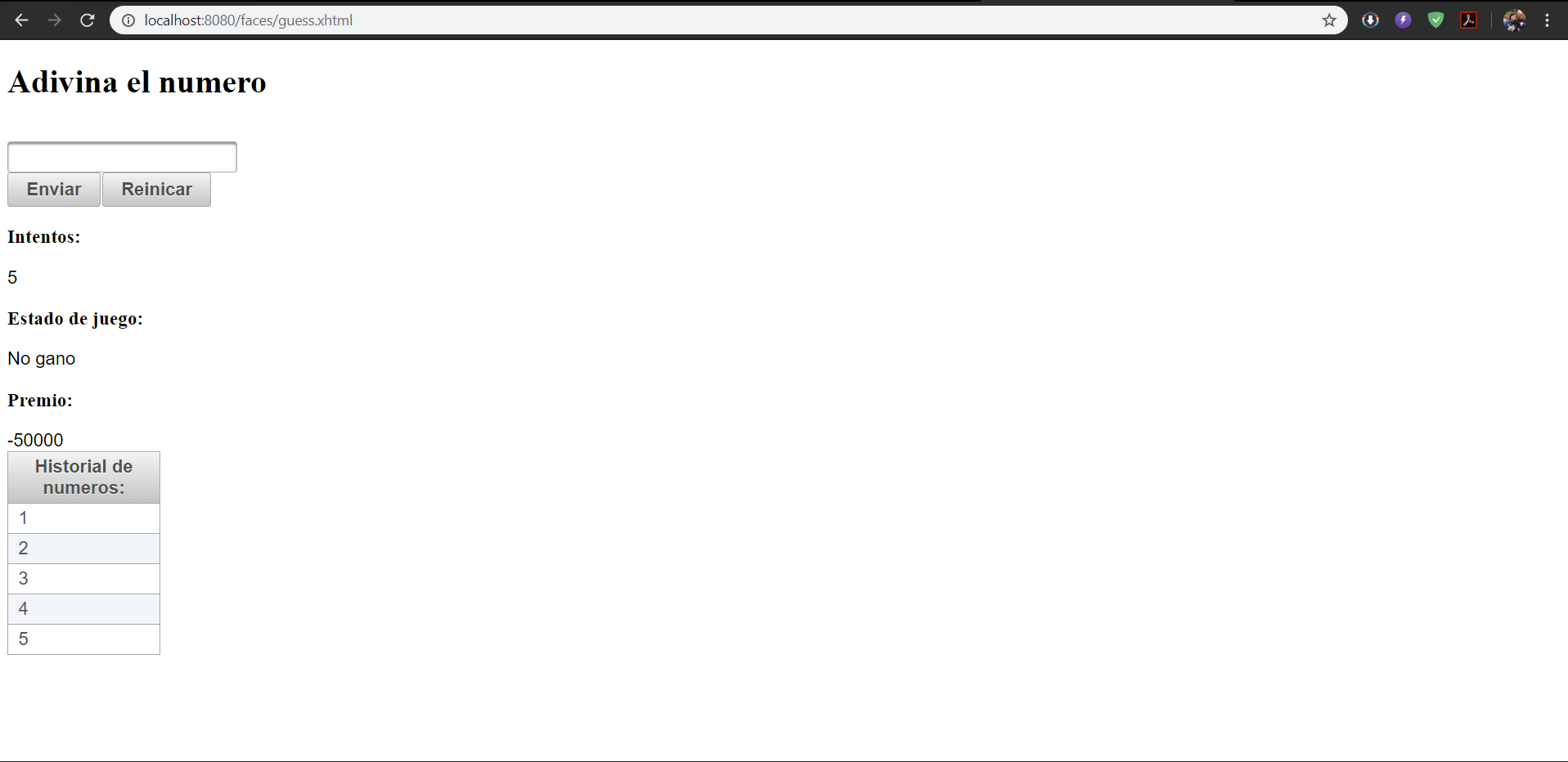
**Set:**



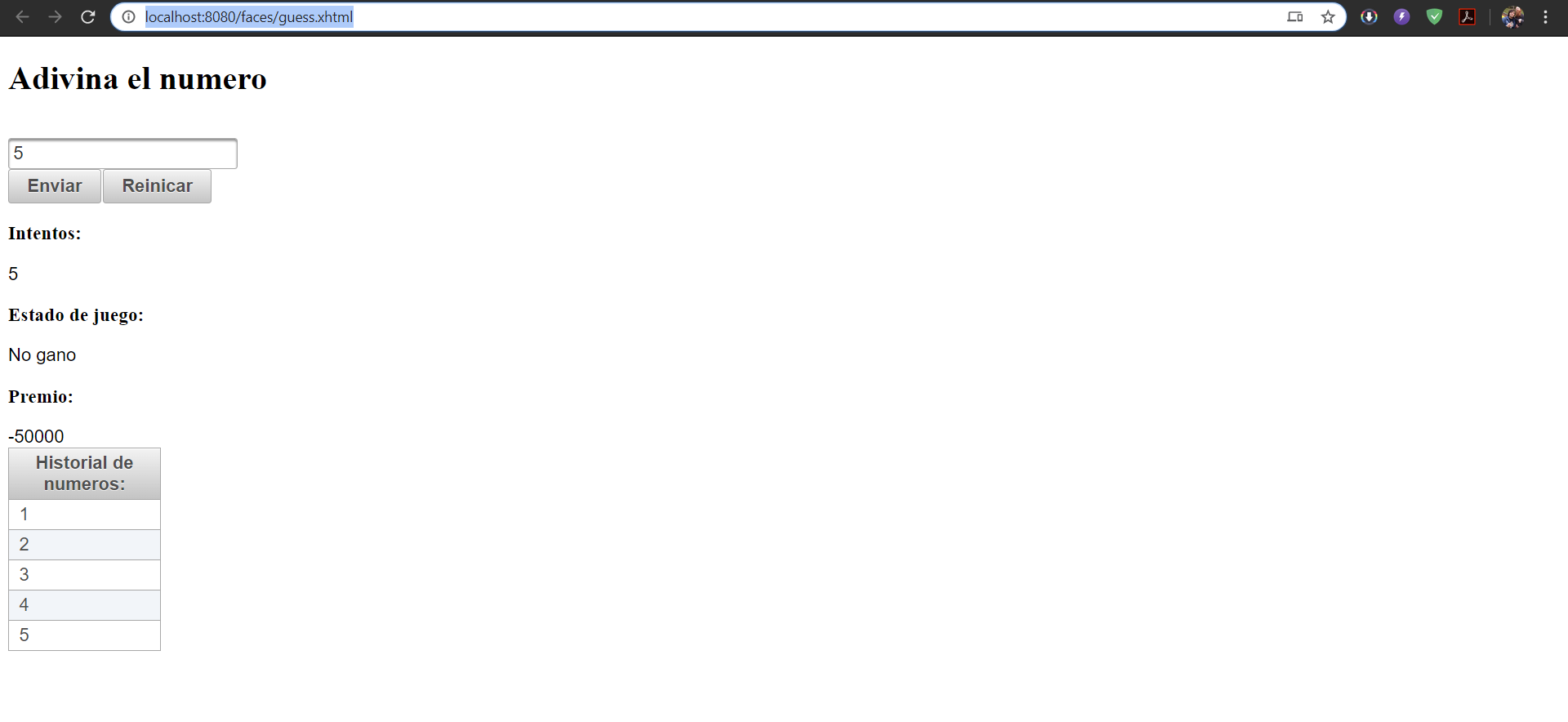
**PARTE 4**

Si todo funcionó correctamente, realice las siguientes pruebas:

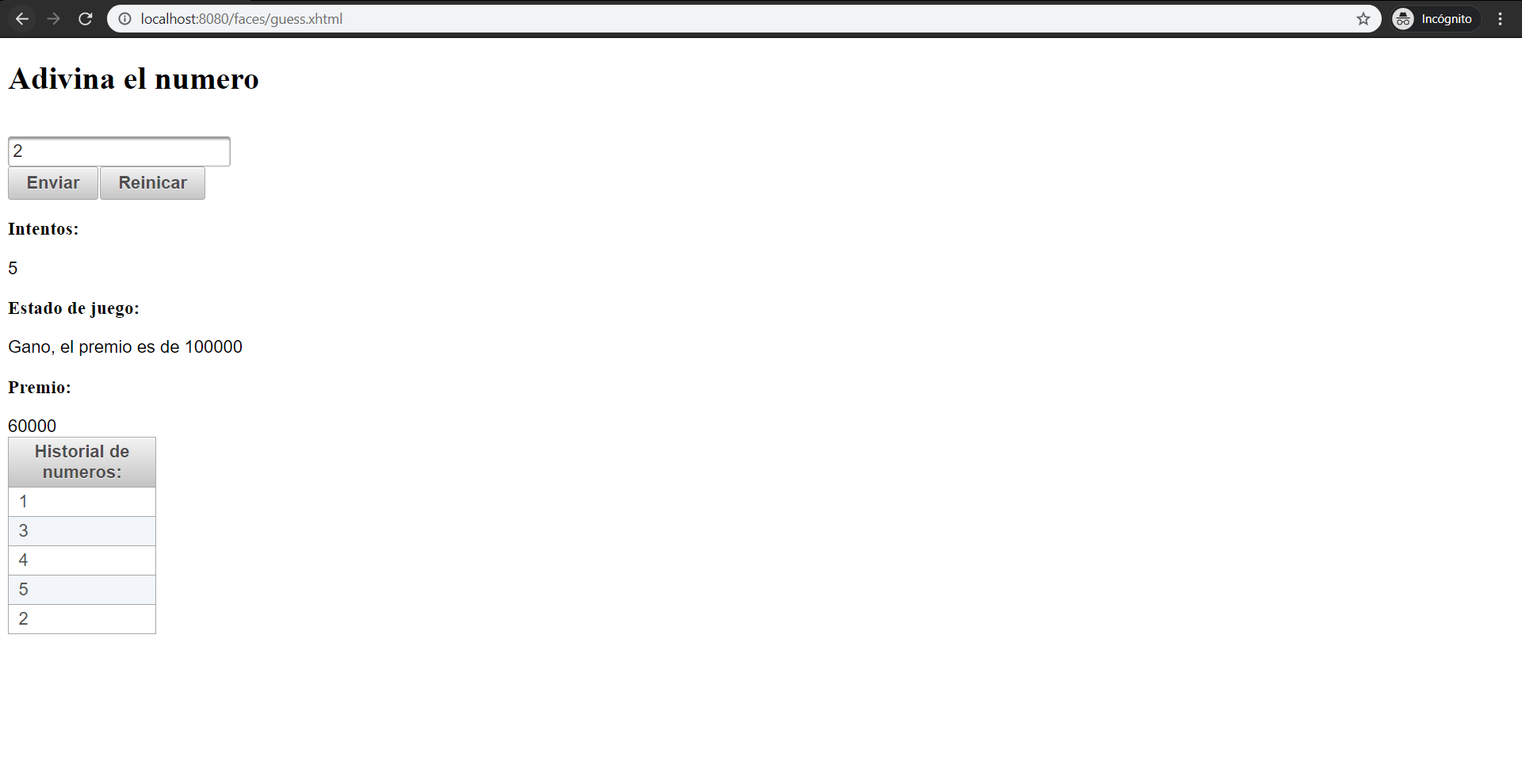
* Abra la aplicación en un explorador. Realice algunas pruebas con el juego e intente adivinar el número
* Abra la aplicación en dos computadores diferentes. Si no dispone de uno, hágalo en dos navegadores diferentes (por ejemplo Chrome y Firefox; incluso se puede en un único navegador usando una ventana normal y una ventana de incógnito / privada). Haga cinco intentos en uno, y luego un intento en el otro. ¿Qué valor tiene cada uno?



**Chrome normal :**



**Chrome incognito:**



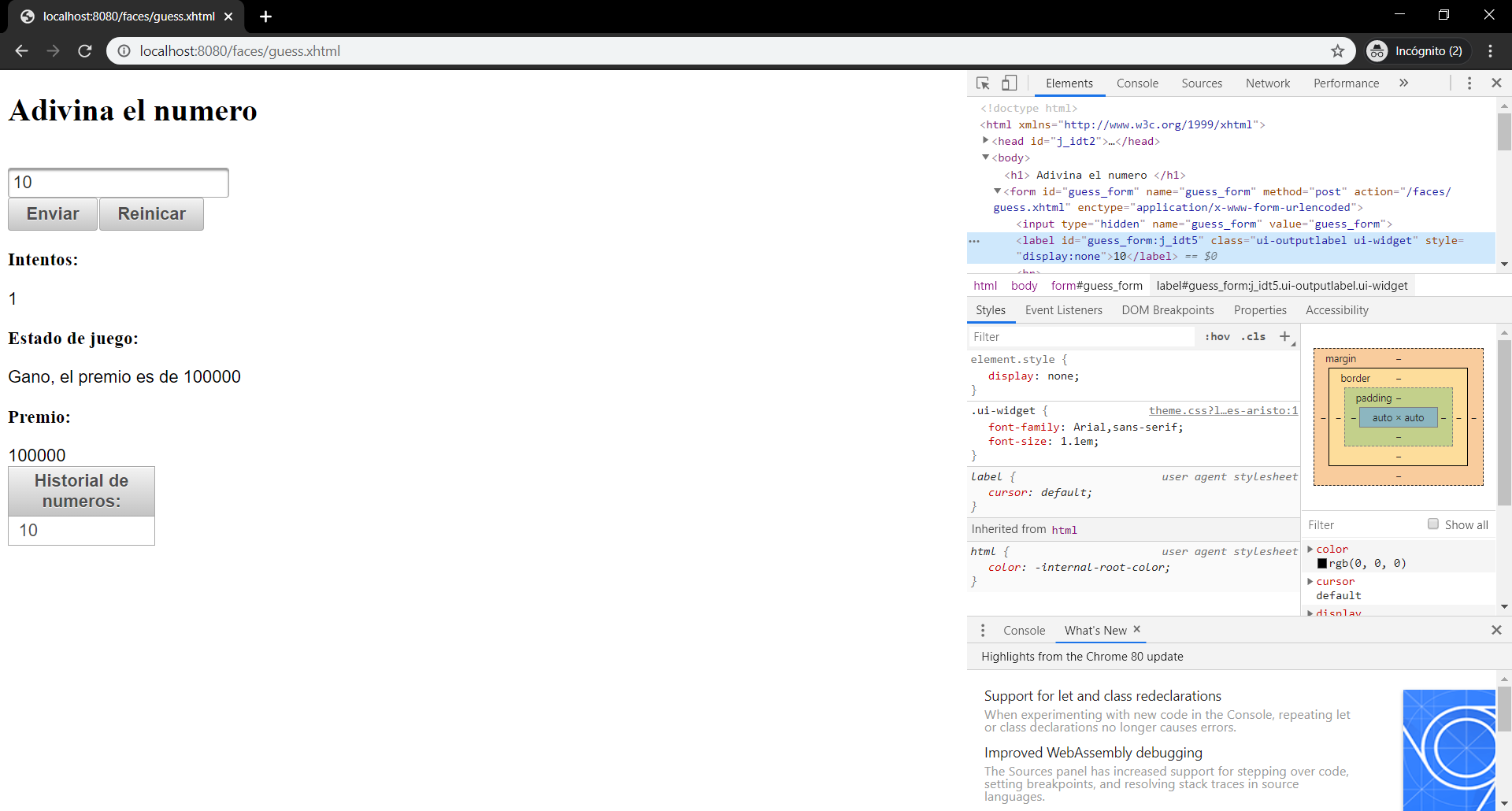
* Aborte el proceso de Tomcat-runner haciendo Ctrl+C en la consola, y modifique el código del backing-bean de manera que use la anotación @SessionScoped en lugar de @ApplicationScoped. Reinicie la aplicación y repita el ejercicio anterior.

Dado la anterior, ¿Cuál es la diferencia entre los backing-beans de sesión y los de aplicación?

Un [@SessionScoped](https://docs.oracle.com/javaee/7/api/javax/enterprise/context/SessionScoped.html) vive tanto como la sesión HTTP establecida.

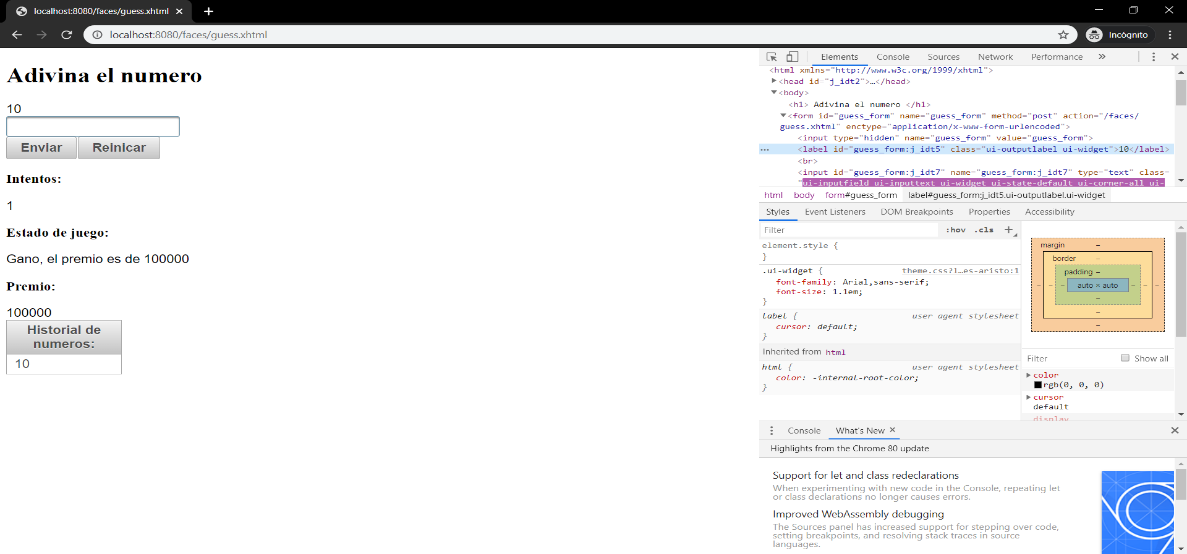
Un [@ApplicationScoped](https://docs.oracle.com/javaee/7/api/javax/enterprise/context/ApplicationScoped.html) vive mientras se ejecute la aplicación web.

* Ubique el código HTML generado por el servidor.
* Busque el elemento oculto, que contiene el número generado aleatoriamente.

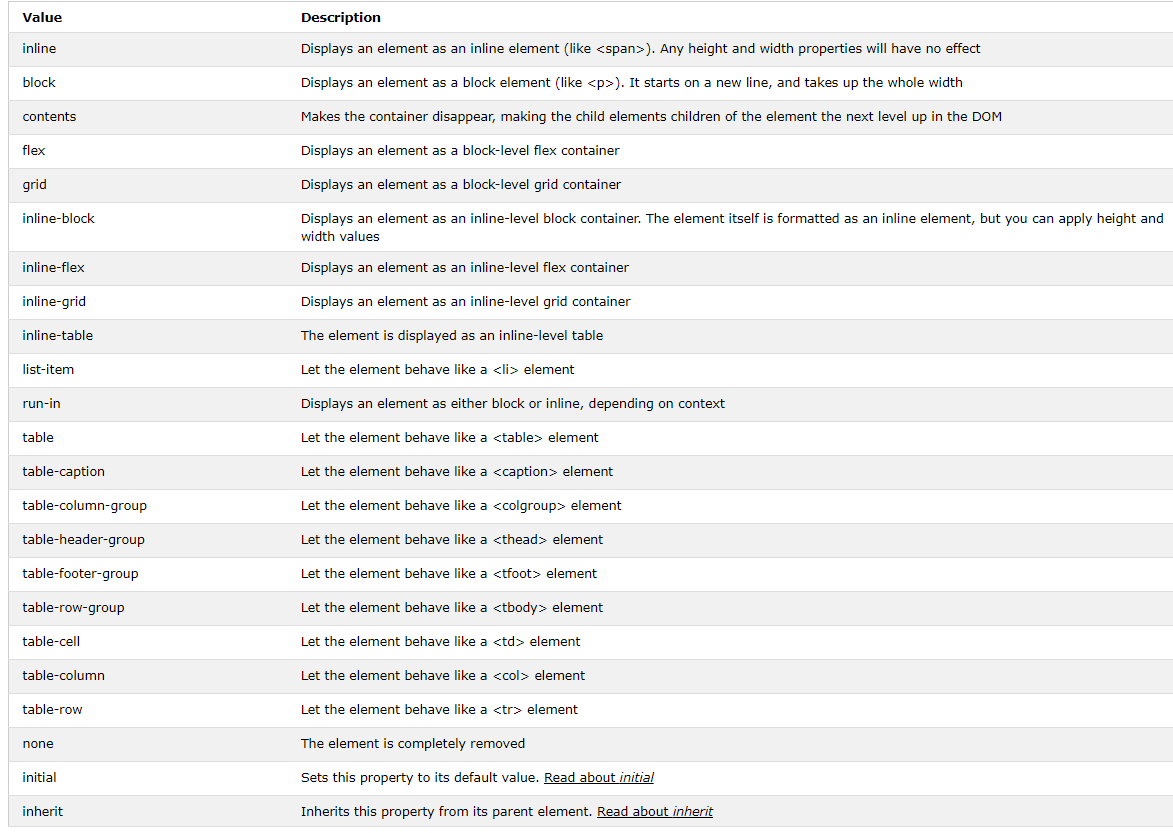


* En la sección de estilos, deshabilite el estilo que oculta el elemento para que sea visible.
* Observe el cambio en la página, cada vez que se realiza un cambio en el estilo.

Quitamos el style="display:none"



* Revise qué otros estilos se pueden agregar a los diferentes elementos y qué efecto tienen en la visualización de la página.



* Revise qué otros cambios se pueden realizar y qué otra información se puede obtener de las herramientas de desarrollador.

