UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

TAREA HTTP - URI

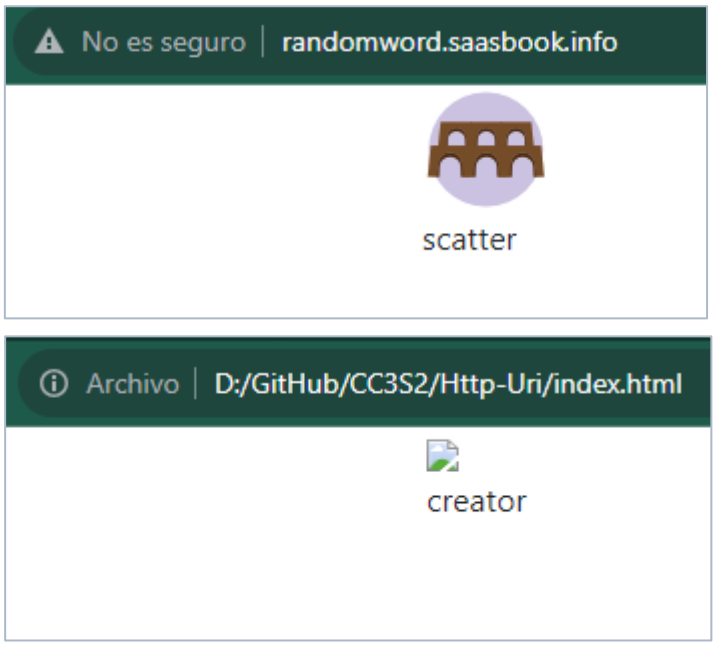
- SEMESTRE: 2023-2
- PROFESOR: Cesar Jesús Lara Ávila

Integrantes:

- Bustos Ttito, José Fabricio
- Pérez Cueva Chandler Steven
- Poma Navarro Christian Daniel

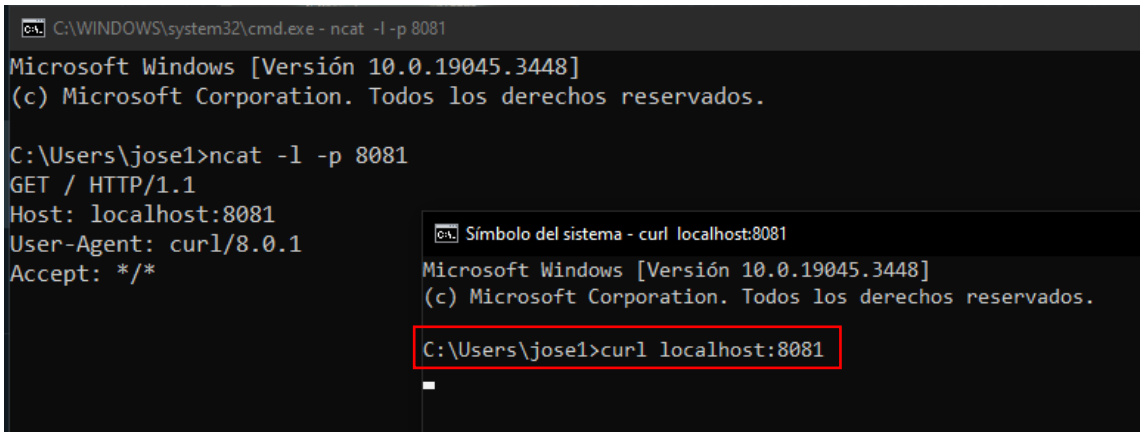
Pregunta:
¿Cuáles son las dos diferencias principales que has visto anteriormente y lo que ves en un navegador web 'normal'? ¿Qué explica estas diferencias?

En la web normal cada vez que actualizo la página me da una palabra aleatoria, en cambio en el archivo que guarde el código que me dio el comando curl solo sale texto plano, que en este caso sale la palabra creator.



La página web puede incluir elementos adicionales que no se obtienen el comando curl. Estos elementos pueden incluir referencias a archivos CSS, JavaScript, imágenes, etc.

Pregunta:
Suponiendo que estás ejecutando curl desde otro shell ¿qué URL tendrás que pasarle a curl para intentar acceder a tu servidor falso y por qué?



Una vez mandado el comado del ncat, que permite iniciar un servidor que escucha el puerto 8081, la URL que le enviaría seria localhost (127.0.0.1), ya que le mandamos la ip local de mi computadora.

Pregunta:
La primera línea de la solicitud identifica qué URL desea recuperar el cliente. ¿Por qué no ves http://localhost:8081 en ninguna parte de esa línea?

```
GET / HTTP/1.1
```

Es una convención en las solicitudes HTTP que la URL completa no se repita cuando se solicita la página de inicio en el puerto predeterminado. En su lugar, se utiliza la ruta "/" para representar la página de inicio.

Pregunta:
Según los encabezados del servidor, ¿cuál es el código de respuesta HTTP del servidor que indica el estado de la solicitud del cliente y qué versión del protocolo HTTP utilizó el servidor para responder al cliente?

El código de respuesta es el 200, y la versión del protocolo es HTTP/1.1

```
C:\Users\jose1>curl -i "http://randomword.saasbook.info"
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/html;charset=utf-8
Content-Length: 483
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.4.2 (Ruby/2.6.6/2020-03-31)
Date: Sun, 24 Sep 2023 22:11:08 GMT
Via: 1.1 vegur

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" r
    <title>Random Word Generator</title>
  <body class="container">
    <div id="image">
      <img src="esaas.png">
    </div>
    <div id="word">
      secretive
    </div>
  </body>
</html>
```

Pregunta:
Cualquier solicitud web determinada puede devolver una página HTML, una imagen u otros tipos de entidades. ¿Hay algo en los encabezados que crea que le dice al cliente cómo interpretar el resultado?

Si el Content-type, que me dice el tipo de formato que me devuelve que en este caso es HTML, y también indica la codificación de caracteres utilizada para el contenido, aqui, se especifica utf-8.

Pregunta:
¿Cuál sería el código de respuesta del servidor si intentaras buscar una URL inexistente en el sitio generador de palabras aleatorias?

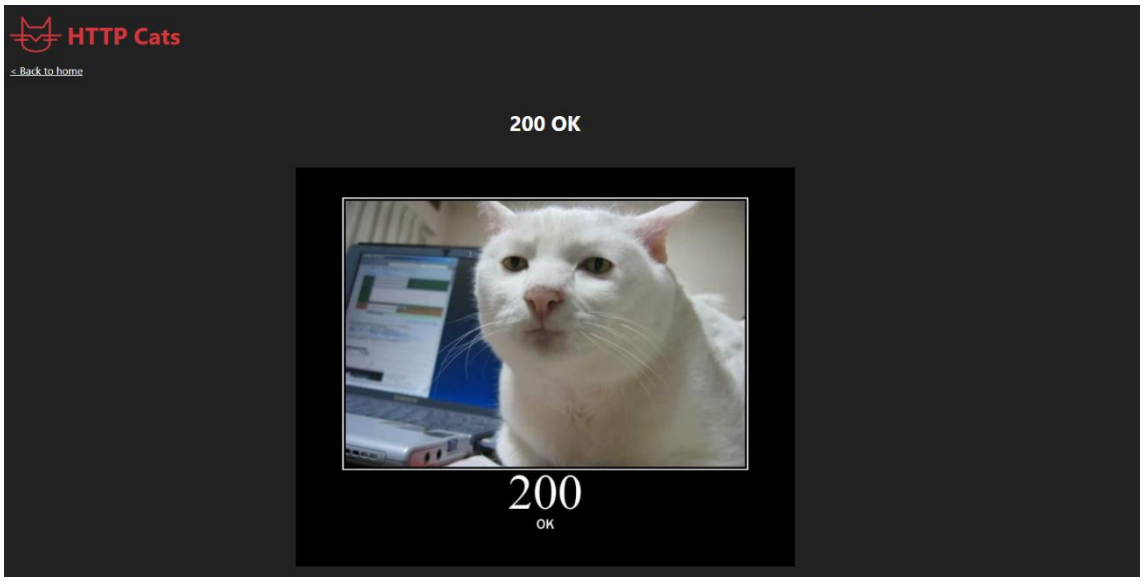
```
C:\Users\jose1>curl -i "http://randomword.saasbook.info/hola"
HTTP/1.1 404 Not Found
Connection: keep-alive
X-Cascade: pass
Content-Type: text/html;charset=utf-8
Content-Length: 9
X-Xss-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Server: WEBrick/1.4.2 (Ruby/2.6.6/2020-03-31)
Date: Sun, 24 Sep 2023 22:32:18 GMT
Via: 1.1 vegur
```

404, es el código de la respuesta del servidor el cual me indica que no encuentra el recurso solicitado.

Pregunta:
¿Qué otros códigos de error HTTP existen? Utiliza Wikipedia u otro recurso para conocer los significados de algunos de los más comunes: 200, 301, 302, 400, 404, 500. Ten en cuenta que estas son familias de estados: todos los estados 2xx significan funcionó, todos los 3xx son redireccionar etc.

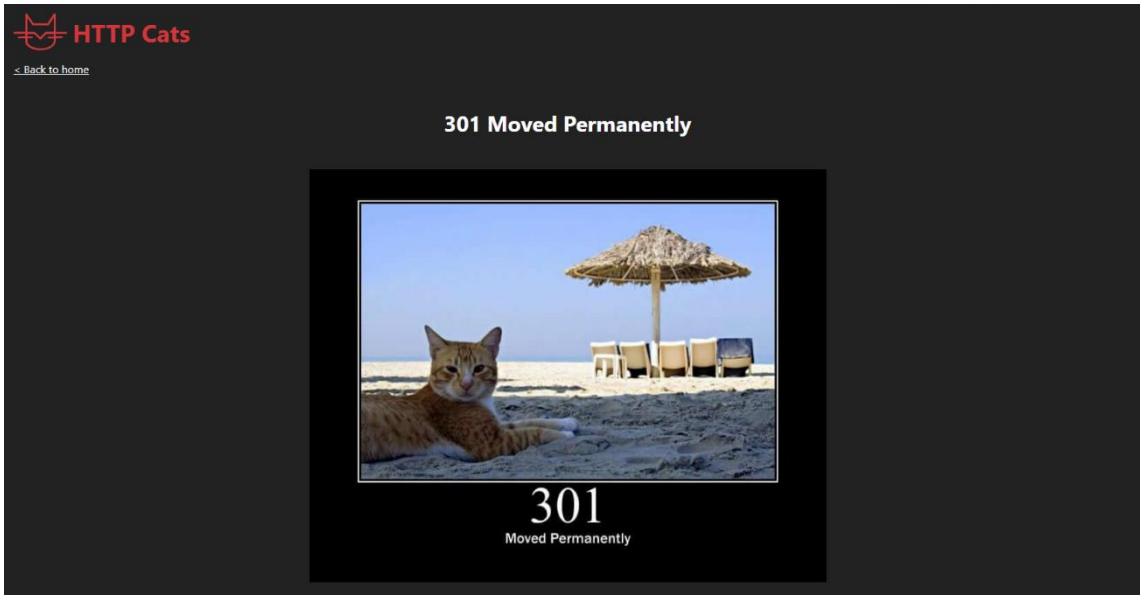
En la página https://http.cat/ nos da todo los tipos de códigos de respuesta a las solicitudes HTTP

200:



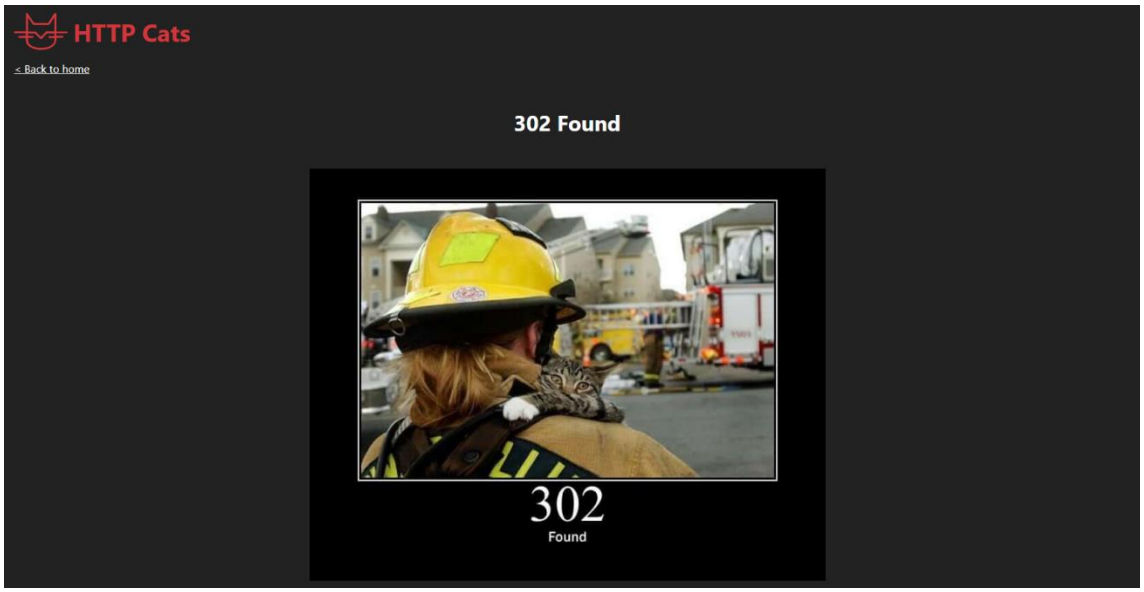
El código de estado HTTP 200 OK, que denota éxito, señala que la solicitud fue exitosa.

301:



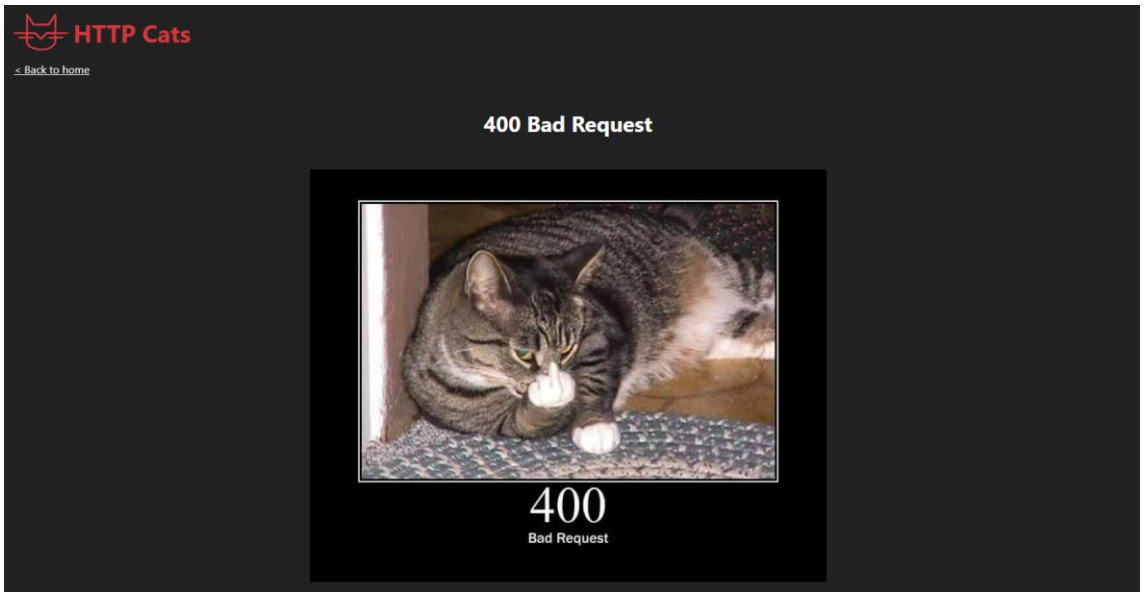
Indica que el recurso solicitado se ha movido definitivamente a la URL proporcionada por los encabezados de Ubicación.

302:



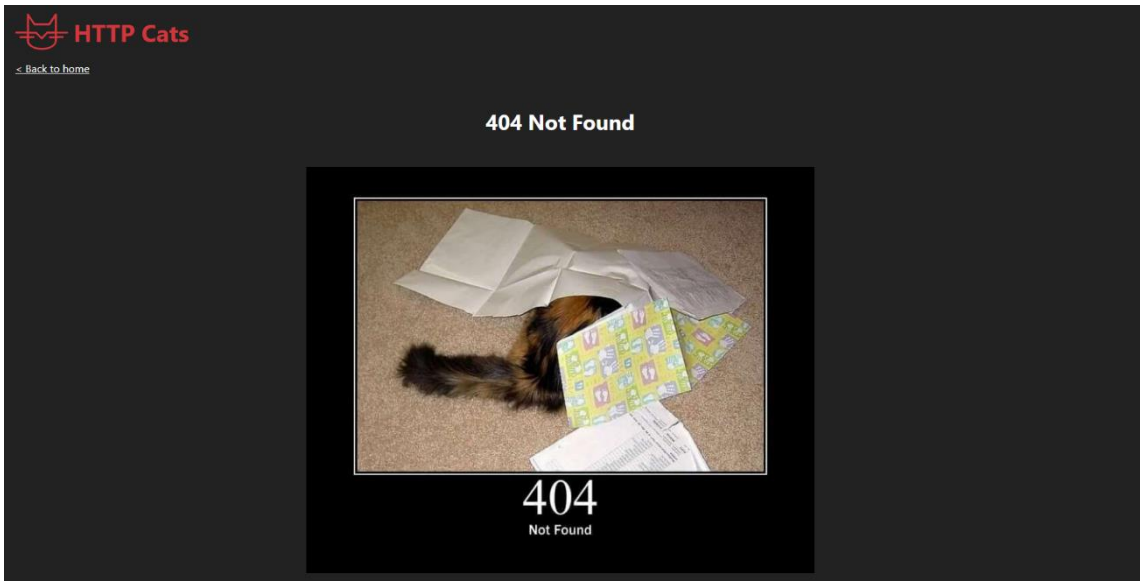
Indica que el recurso solicitado se ha movido temporalmente a la URL proporcionada por el encabezado Ubicación.

400:



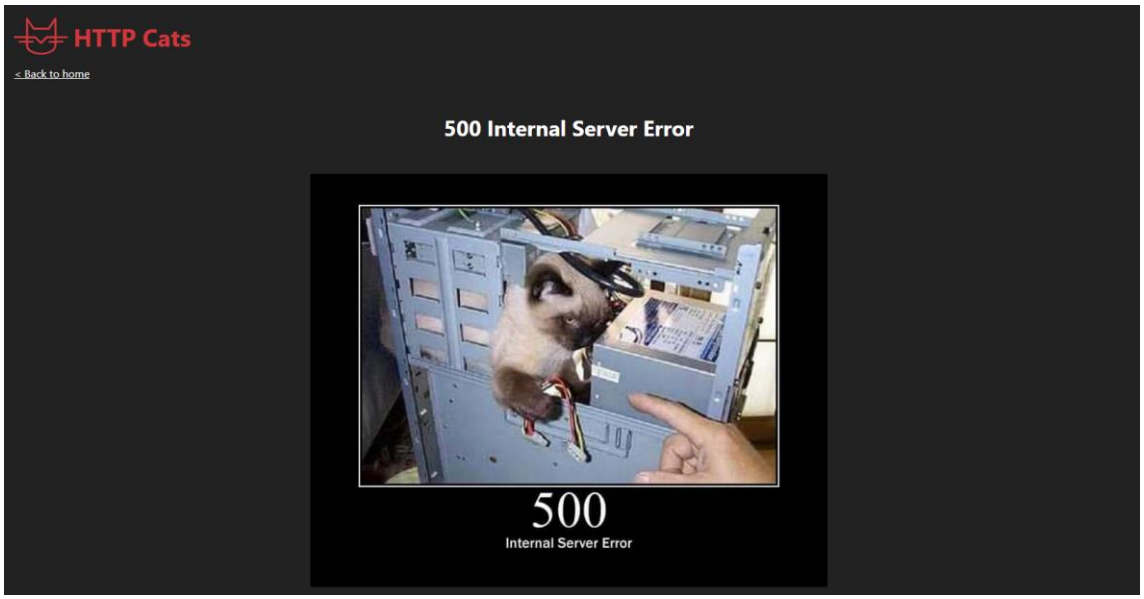
Indica que el servidor no puede o no procesará la solicitud debido a algo que se percibe como un error del cliente

404:



Indica que el servidor no puede encontrar el recurso solicitado.

500:

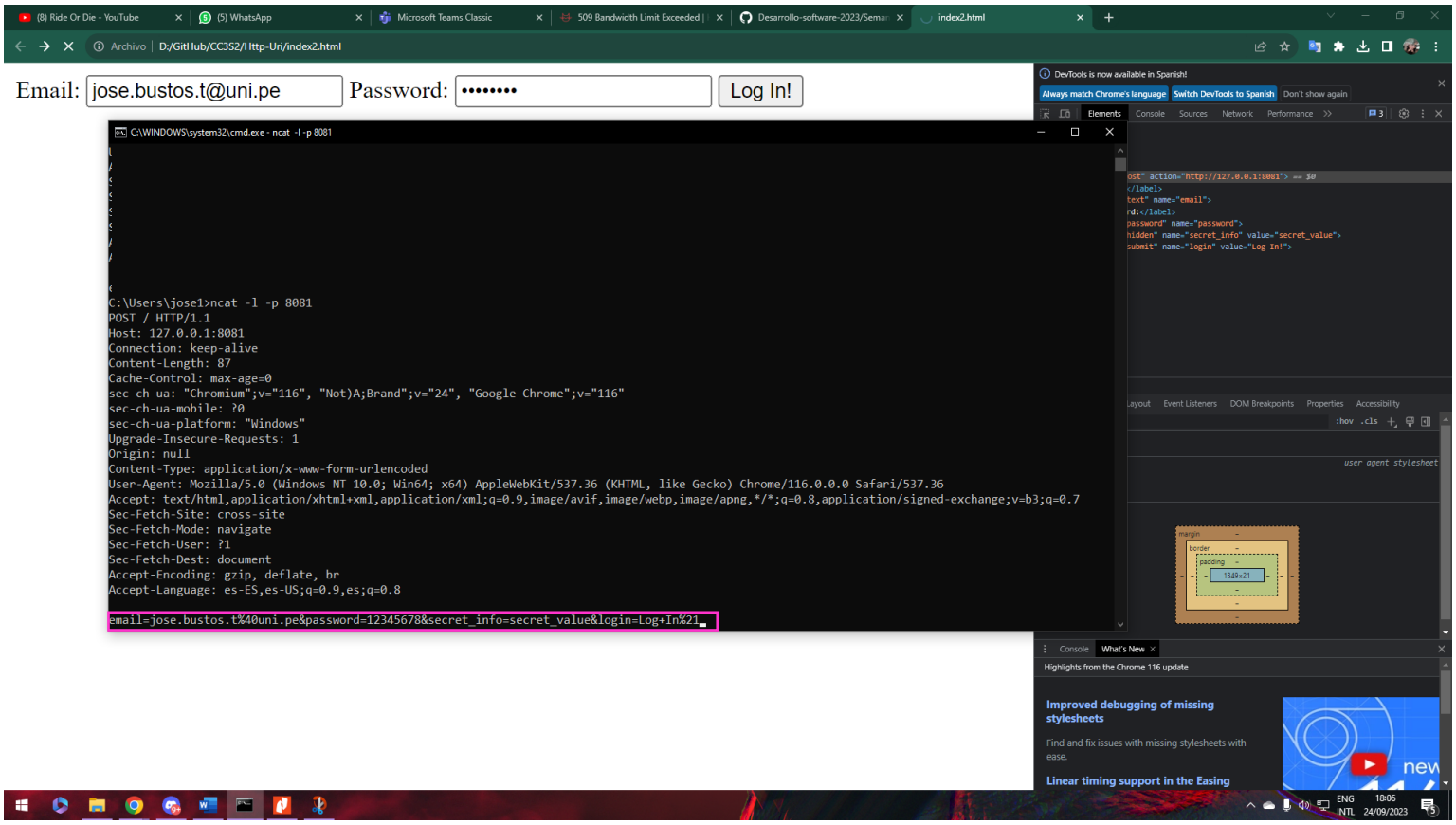


Internal Server Error indica que el servidor encontró una condición inesperada que le impidió cumplir con la solicitud.

Pregunta:
Tanto el encabezado 4xx como el 5xx indican condiciones de error. ¿Cuál es la principal diferencia entre 4xx y 5xx?

Los 4xx son errores por parte del cliente, y los 5xx por parte del servidor.

Pregunta:
Cuando se envía un formulario HTML, se genera una solicitud HTTP POST desde el navegador. Para llegar a tu servidor falso, ¿con qué URL deberías reemplazar Url-servidor-falso en el archivo anterior?



Cambiamos a la dirección IP local de nuestro sistema que es `http://127.0.0.1:8081`

Pregunta:
¿Cómo se presenta al servidor la información que ingresó en el formulario? ¿Qué tareas necesitaría realizar un framework SaaS como Sinatra o Rails para presentar esta información en un formato conveniente a una aplicación SaaS escrita, por ejemplo, en Ruby?

Se presenta al servidor como una solicitud HTTP POST que contiene los datos del formulario. La solicitud POST incluirá los datos del formulario como parte de su cuerpo (body), y estos datos estarán codificados en un formato específico.

1. ***Recibir la Solicitud POST:*** El framework debe estar configurado para manejar las solicitudes HTTP POST en la URL especificada en el atributo action
2. ***Parsar los Datos del Formulario:*** El framework debe analizar los datos enviados en la solicitud POST para extraer la información ingresada en el formulario, como el correo electrónico, la contraseña y cualquier otro campo.
3. ***Validación y Procesamiento:*** El framework debe realizar la validación de los datos del formulario.
4. ***Generar Respuesta HTML o JSON:*** Una vez que los datos del formulario se hayan procesado y validado, el framework puede generar una respuesta HTML o JSON para enviarla al cliente.

Preguntas:

¿Cuál es el efecto de agregar parámetros URI adicionales como parte de la ruta POST?
No tiene ningún efecto directo en el contenido del formulario que se envía a través de la solicitud POST.

```
C:\Users\jose1>ncat -l -p 8081
POST /buscar?query=ejemplo&pagina=1 HTTP/1.1
Host: 127.0.0.1:8081
Connection: keep-alive
Content-Length: 92
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="116", "Not)A;Brand";v="24", "Google Chrome";v="116"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: null
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es-US;q=0.9,es;q=0.8

email=dsad&password=dasasdasd&secret_info=secret_value&colorpicker=%23a03131&login=Log+In%21
```

¿Cuál es el efecto de cambiar las propiedades de nombre de los campos del formulario?
Solo afectará cómo se identifican esos campos en el lado del servidor cuando se procesa la solicitud POST.

```
email1=hola&password=carro&secret_info=secret_value&colorpicker=%2369b31e&login=Log+In%21
```

¿Puedes tener más de un botón Submit? Si es así, ¿cómo sabe el servidor en cuál se hizo clic?
(Sugerencia: experimenta con los atributos de la etiqueta <submit>).
Si, en la solicitud POST te indica en que botón se hizo el clic como muestra la imagen

```
C:\Users\jose1>ncat -l -p 8081
POST /buscar?query=ejemplo&pagina=1 HTTP/1.1
Host: 127.0.0.1:8081
Connection: keep-alive
Content-Length: 72
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="116", "Not)A;Brand";v="24", "Google Chrome";v="116"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: null
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es-US;q=0.9,es;q=0.8

email1=hola&password=comoestas&secret_info=secret_value caballo=caballo1
```

¿Se puede enviar el formulario mediante GET en lugar de POST? En caso afirmativo, ¿cuál es la diferencia en cómo el servidor ve esas solicitudes?

Si, los datos se muestran dentro de la URL como muestra en la imagen.

```
GET /buscar?email1=get&password=get1&secret_info=secret_value&login=Log+In%21 HTTP/1.1
Host: 127.0.0.1:8081
Connection: keep-alive
sec-ch-ua: "Chromium";v="116", "Not)A;Brand";v="24", "Google Chrome";v="116"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: es-ES,es-US;q=0.9,es;q=0.8
```

¿Qué otros verbos HTTP son posibles en la ruta de envío del formulario? ¿Puedes hacer que el navegador web genere una ruta que utilice PUT, PATCH o DELETE?

Los verbos HTTP más comunes utilizados en formularios HTML son GET y POST. Los navegadores web generan solicitudes GET y POST. Para utilizar PUT, PATCH o DELETE, se necesita un lenguaje de programador.

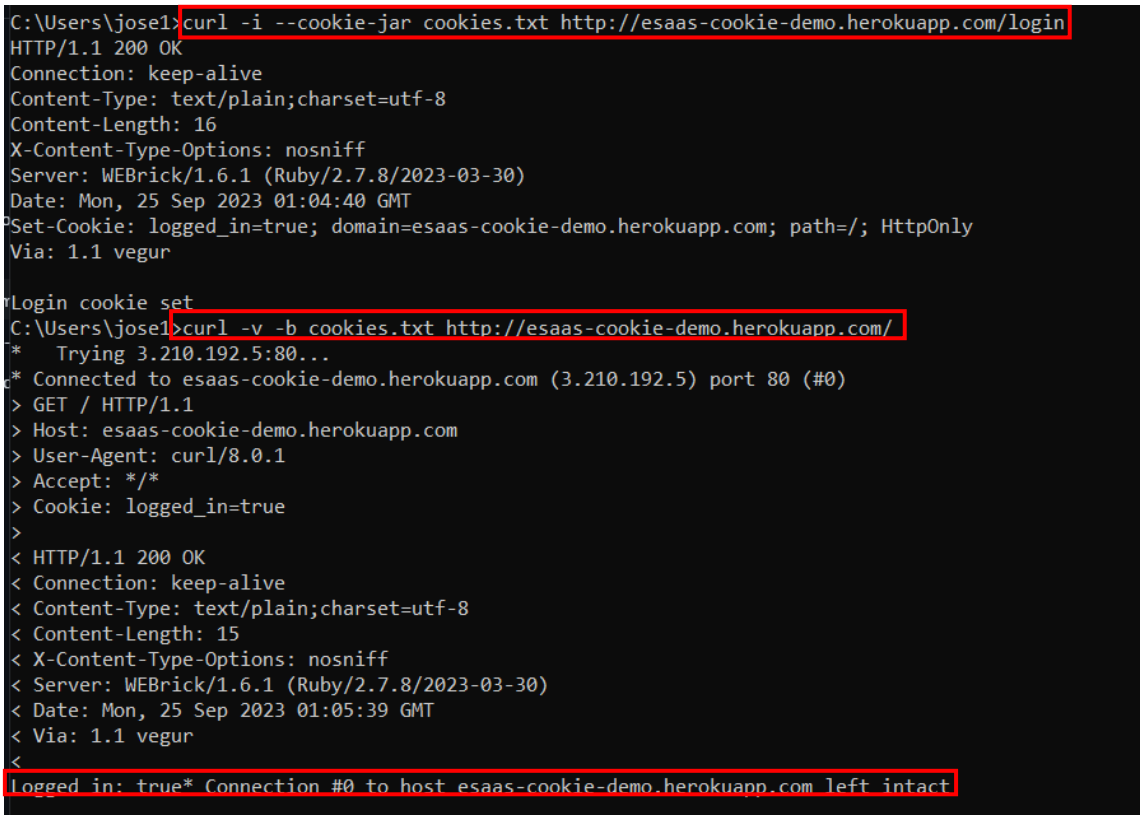
Pregunta:
Prueba las dos primeras operaciones GET anteriores. El cuerpo de la respuesta para la primera debe ser "Logged in: false" y para la segunda "Login cookie set". ¿Cuáles son las diferencias en los encabezados de respuesta que indican que la segunda operación está configurando una cookie? (Sugerencia: usa curl -v, que mostrará tanto los encabezados de solicitud como los encabezados y el cuerpo de la respuesta, junto con otra información de depuración. curl --help imprimirá una ayuda voluminosa para usar cURL y man curl mostrará la página del manual de Unix para cURL en la mayoría de los sistemas.)

```
C:\Users\jose1>curl -v "https://esaas-cookie-demo.herokuapp.com/"
*   Trying 54.83.6.65:443...
* Connected to esaas-cookie-demo.herokuapp.com (54.83.6.65) port 443 (#0)
* schannel: disabled automatic use of client certificate
* ALPN: offers http/1.1
* ALPN: server did not agree on a protocol. Uses default.
* using HTTP/1.x
> GET / HTTP/1.1
> Host: esaas-cookie-demo.herokuapp.com
> User-Agent: curl/8.0.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Connection: keep-alive
< Content-Type: text/plain;charset=utf-8
< Content-Length: 16
< X-Content-Type-Options: nosniff
< Server: WEBrick/1.6.1 (Ruby/2.7.8/2023-03-30)
< Date: Mon, 25 Sep 2023 00:48:03 GMT
< Via: 1.1 vegur
<
Logged in: false* Connection #0 to host esaas-cookie-demo.herokuapp.com left intact
```

```
C:\Users\jose1>curl -v "https://esaas-cookie-demo.herokuapp.com/login"
*   Trying 3.210.192.5:443...
* Connected to esaas-cookie-demo.herokuapp.com (3.210.192.5) port 443 (#0)
* schannel: disabled automatic use of client certificate
* ALPN: offers http/1.1
* ALPN: server did not agree on a protocol. Uses default.
* using HTTP/1.x
> GET /login HTTP/1.1
> Host: esaas-cookie-demo.herokuapp.com
> User-Agent: curl/8.0.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Connection: keep-alive
< Content-Type: text/plain;charset=utf-8
< Content-Length: 16
< X-Content-Type-Options: nosniff
< Server: WEBrick/1.6.1 (Ruby/2.7.8/2023-03-30)
< Date: Mon, 25 Sep 2023 00:48:20 GMT
< Set-Cookie: logged_in=true; domain=esaas-cookie-demo.herokuapp.com; path=/; secure; HttpOnly
< Via: 1.1 vegur
<
Login cookie set* Connection #0 to host esaas-cookie-demo.herokuapp.com left intact
```

En el encabezado de la pagina sin el login, manda la operación get a la raíz, mientras que el que si tiene login manda la operación get /login.

Ejecutamos los comandos dados:



Pregunta:

Al observar el encabezado Set-Cookie o el contenido del archivo cookies.txt, parece que podría haber creado fácilmente esta cookie y simplemente obligar al servidor a creer que ha iniciado sesión. En la práctica, ¿cómo evitan los servidores esta inseguridad?

- Las cookies pueden configurarse como "seguras", lo que significa que solo se transmiten a través de conexiones HTTPS seguras.
- Las cookies suelen tener una fecha de vencimiento después de la cual ya no son válidas.