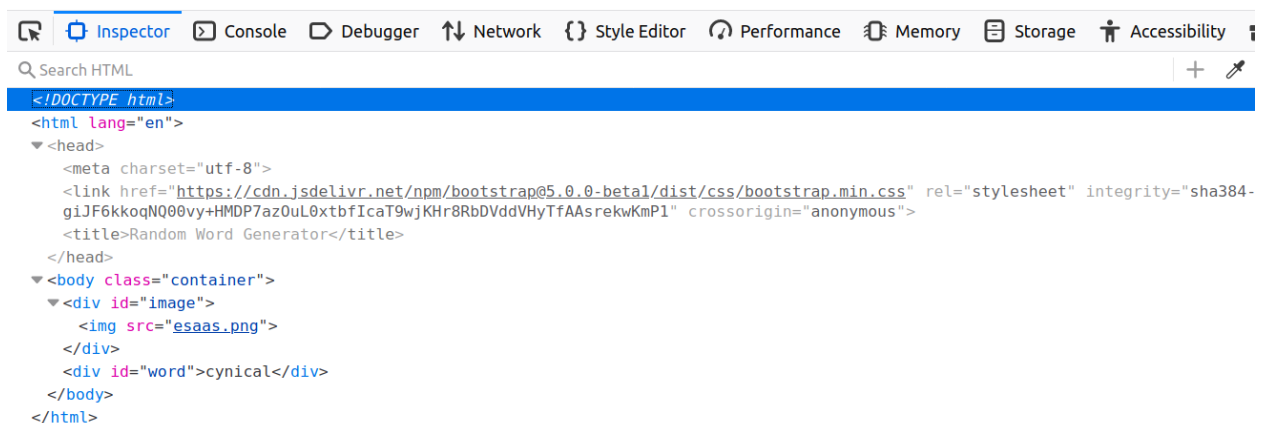


Introduccion HTTP y URL

Comprendiendo Request y Response

```
devasc@labvm:~$ curl 'http://randomword.saasbook.info'
<!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" rel="stylesheet" integrity="sha384-giJF6kkoqN00vy+HMDP7az0uL0xtbficaT9wjKhr8RbDVddVHyTfAAsrekWkmP1" crossorigin="anonymous">
    <title>Random Word Generator</title>
  <body class="container">
    <div id="image">
      
    </div>
    <div id="word">
      stupid
    </div>
  </body>
</html>
devasc@labvm:~$
```



```
<!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" rel="stylesheet" integrity="sha384-giJF6kkoqN00vy+HMDP7az0uL0xtbficaT9wjKhr8RbDVddVHyTfAAsrekWkmP1" crossorigin="anonymous">
    <title>Random Word Generator</title>
  </head>
  <body class="container">
    <div id="image">
      
    </div>
    <div id="word">cynical</div>
  </body>
</html>
```

Guardamos la salida en el archivo rw.html

```
devasc@labvm:~$ curl 'http://randomword.saasbook.info' > rw.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  484    100  484    0    0   1222    0 --:--:-- --:--:-- --:--:--  1222
```

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

- Respuesta: En la pagina web de randomword se puede ver la interfaz, es decir, en la pagina web ya esta renderizada y en la terminal solo se puede ver codigo html.

Cómo cree el servidor que se ve una solicitud

Nos haremos pasar por un servidor Web escuchando el puerto 8081: `nc -l 8081`.

```
devasc@labvm:~$ nc -l 8081
```

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?

- Respuesta : Tenemos que pasarle la URL `http://localhost:8081` porque estamos ejecutando el “servidor” desde nuestra propia maquina y estamos escuchando desde el puerto 8081

Abrimos otra terminal y mandamos una solicitud http al servidor falso

```
devasc@labvm:~$ curl http://localhost:8081
```

Retornamos al terminal donde teniamos a nuestro servidor falso escuchando y notamos lo siguiente

```
devasc@labvm:~$ nc -l 8081
GET / HTTP/1.1
Host: localhost:8081
User-Agent: curl/7.68.0
Accept: */*
```

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?

- Respuesta: Porque esa informacion se encuentra en el comando curl

Veamos cómo se ve la respuesta desde el cliente.

Probaremos `curl --help` para verificar que la línea de comando

`curl -i 'http://randomword.saasbook.info'` mostrará ambos el encabezado de respuesta del servidor y el cuerpo de la respuesta.

```
--http2          Use HTTP 2
--http2-prior-knowledge Use HTTP 2 without HTTP/1.1 Upgrade
--http3          Use HTTP v3
--ignore-content-length Ignore the size of the remote resource
-i, --include     Include protocol response headers in the output
-k, --insecure    Allow insecure server connections when using SSL
--interface <name> Use network INTERFACE (or address)
-4, --ipv4        Resolve names to IPv4 addresses
-6, --ipv6        Resolve names to IPv6 addresses
```



```
devasc@labvm:~$ 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 20:58:07 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" rel="stylesheet" integrity="sha384-giJf6kkoqNQ00vy+
HMDP7az0uL0xtbFIcAT9wjKhr8RbDdVdVHyTfAAsrekwKmP1" crossorigin="anonymous">
    <title>Random Word Generator</title>
  <body class="container">
    <div id="image">
      
    </div>
    <div id="word">
      treatment
    </div>
  </body>
</html>
```

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?

- Respuesta : En la primera línea del encabezado se puede ver : HTTP/1.1 200 Ok , esto indica que la versión es 1.1 y el estado “200 ok “ indica que la solicitud se ha procesado correctamente

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?

- Como habíamos visto anteriormente <http://randomword.saasbook.info> te devolvía una respuesta html y en la tercera línea podemos ver “Content-type : text/html /

charset=utf-8"

por lo que podemos considerar que esto indica que debe interpretarse como un código html

¿Qué sucede cuando falla un HTTP request?

Pregunta: ¿Cuál sería el código de respuesta del servidor si intentaras buscar una URL inexistente en el sitio generador de palabras aleatorias? Pruéba esto utilizando el procedimiento anterior.

```
devasc@labvm:~$ curl -i 'http://randomword.saasbook.info/cualquiercosa'
HTTP/1.1 404 Not Found
Connection: keep-alive
X-Cascade: pass
Content-Type: text/html; charset=utf-8
Content-Length: 18
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 21:06:57 GMT
Via: 1.1 vegur
```

¿Qué otros códigos de error HTTP existen?

- **200** OK ⇒ La solicitud se ha procesado correctamente
- **301** Moved Permanently ⇒ El recurso solicitado a cambiado de ubicacion permanentemente
- **302** Found ⇒ El recurso solicitado a cambiado de ubicacion temporalmente.
- **400** Bad Request ⇒ No se comprende la solicitud del cliente debido a una sintaxis incorrecta
- **404** Not Found ⇒ El recurso solicitado no se enucentra en el servidor
- **500** Internal Server Error ⇒ Error interno en el servidor que impide que la solicitud se procese

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

- Respuesta : La principal diferencia esta en que 4xx se da cuando la solicitud del cliente es incorrecta, es decir, el error esta en el cliente y el 5xx indica un error interno en el servidor.

¿Qué es un cuerpo de Request?

Formulario HTML

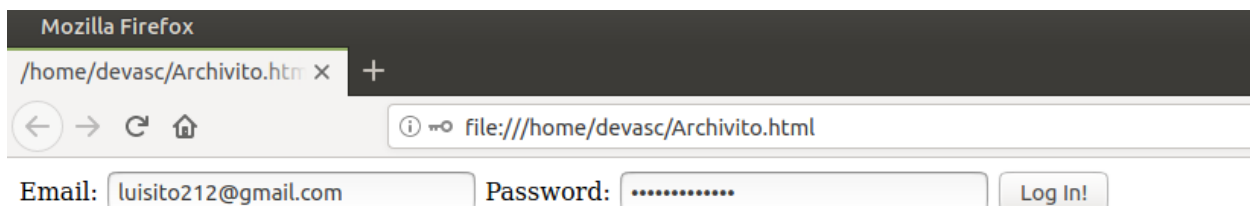
Creamos el archivo Archivito.html y guardamos el formulario html

```
*Archivito.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4 </head>
5 <body> <form method="post" action="http://localhost:8081">
6 <label>Email:</label>
7 <input type="text" name="email">
8 <label>Password:</label>
9 <input type="password" name="password">
10 <input type="hidden" name="secret_info" value="secret_value">
11 <input type="submit" name="login" value="Log In!">
12 </form>
13 </body>
14 </html>
```

Pregunta : ¿con qué URL deberías reemplazar `Url-servidor-falso` en el archivo anterior?

- Respuesta : Debemos cambiar la url por la de "http://localhost:8081" ya que tenemos un servidor falso en nuestra pc a la espera de una solicitud en el puerto 8081.

Si lo abrimos en el navegador y hacemos login



Mozilla Firefox

/home/devasc/Archivito.html x +

file:///home/devasc/Archivito.html

Email: Password:

Notamos que le llega esto a la terminal donde nc esta escuchando

```
devasc@labvm:~  
File Edit View Search Terminal Help  
devasc@labvm:~$ nc -l 8081  
POST / HTTP/1.1  
Host: localhost:8081  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 92  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
  
email=luisito212%40gmail.com&password=contrasena123&secret_info=secret_value&login=Log+In%21
```

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?

Repite el experimento varias veces para responder las siguientes preguntas observando las diferencias en el resultado impreso por `nc`:

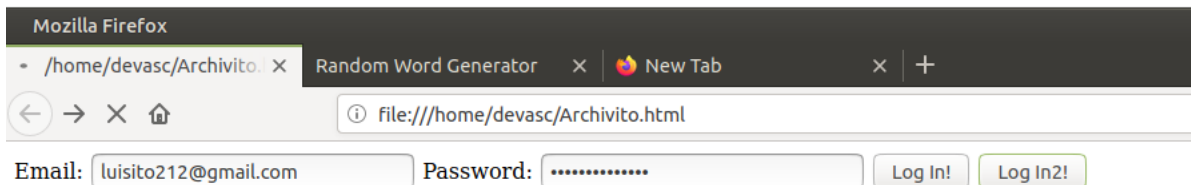
- ¿Cuál es el efecto de cambiar las propiedades de nombre de los campos del formulario?

```
devasc@labvm:~  
File Edit View Search Terminal Help  
devasc@labvm:~$ nc -l 8081  
POST / HTTP/1.1  
Host: localhost:8081  
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 92  
Connection: keep-alive  
Upgrade-Insecure-Requests: 1  
  
email=luisito212%40gmail.com&password=contrasena123&secret_info=secret_value&login=Log+In%21
```

Los campos del formulario se identificarían de otra forma

- ¿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>`).

```
Archivito.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4 </head>
5 <body> <form method="post" action="http://localhost:8081">
6 <label>Email:</label>
7 <input type="text" name="email">
8 <label>Password:</label>
9 <input type="password" name="password">
10 <input type="hidden" name="secret_info" value="secret_value">
11 <input type="submit" name="login" value="Log In!">
12 <input type="submit" name="nuevo_login" value="Log In2!">
13 </form>
14 </body>
15 </html>
16
```



```
devasc@labvm:~$ nc -l 8081
POST / HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 100
Connection: keep-alive
Upgrade-Insecure-Requests: 1

email=luisito212%40gmail.com&password=contrasena1234&secret_info=secret_value&nuevo_login=Log+In2%21
```

Sabe quien hizo click por el atributo “name” que existe en cada campo.

- ¿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?

Con GET, los datos del formulario son visibles en la URL como podemos ver

```
File Edit View Search Terminal Help
devasc@labvm:~$ nc -l 8081
GET /?email=luisito212%40gmail.com&password=contrasena123&secret_info=secret_value&login=Log+In%21 HTTP/1.1
Host: localhost:8081
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20100101 Firefox/77.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

- ¿Qué otros verbos **HTTP** son posibles en la ruta de envío del formulario?
 - **PUT**: Se utiliza para actualizar un recurso específico en el servidor.
 - **PATCH**: Similar a PUT, se utiliza para actualizar un recurso en el servidor, pero en lugar de enviar una representación completa, el cliente envía solo los cambios o modificaciones al recurso.