Práctica 1

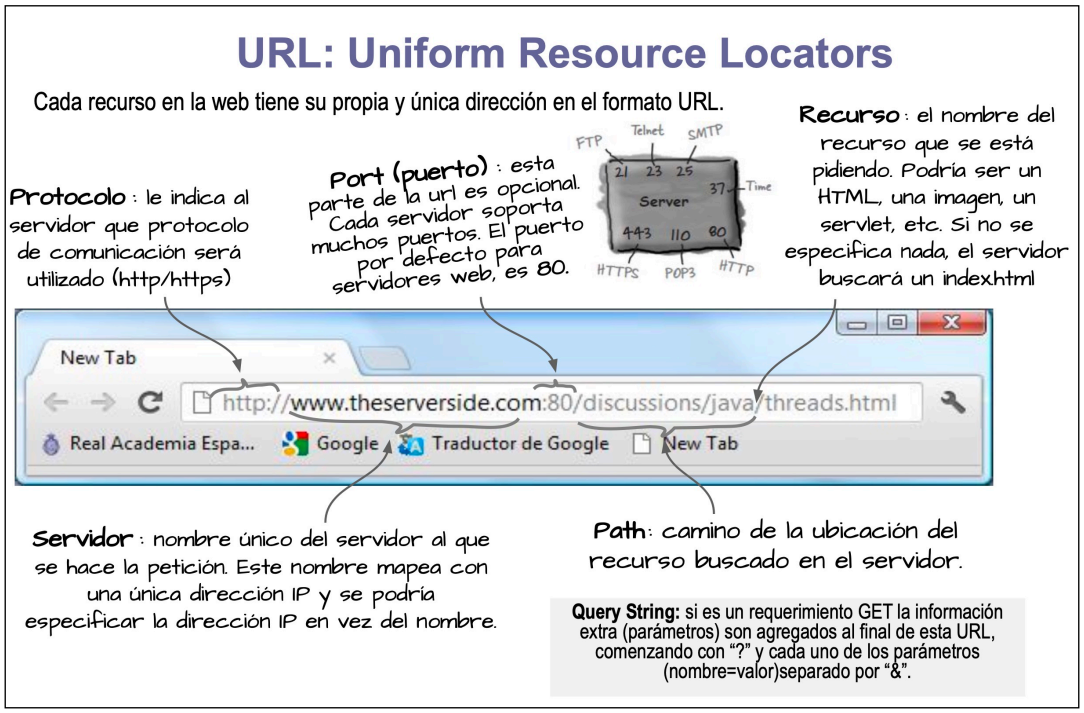
1. Conceptos básicos

a. ¿Qué es una URL y cómo se compone?

Una URL (Uniform Resource Locator) es la dirección única que identifica a cada recurso disponible en la web (como una página, imagen, video, archivo, etc.). Gracias a la URL, el navegador sabe dónde encontrar ese recurso y cómo acceder a él.

Una URL se **compone** de varias partes:

- **Protocolo:** indica el método de comunicación que se usará, como http o https.
- **Servidor (host):** es el nombre del servidor al que se hace la petición. Normalmente es un dominio (www.ejemplo.com), pero también puede ser una dirección IP.
- **Puerto (port):** es opcional. Identifica el canal de comunicación dentro del servidor. El puerto por defecto para la web es el 80 (HTTP) y el 443 (HTTPS).
- **Path (ruta):** es el camino dentro del servidor hacia el recurso solicitado, por ejemplo / discussions/java/threads.html.
- **Recurso:** el archivo específico al que se accede, como index.html, una imagen o un documento. Si no se especifica, el servidor normalmente carga index.html.
- **Query string (cadena de consulta):** son parámetros adicionales que se envían al servidor, generalmente después de un ?, por ejemplo: ?id=5&orden=asc. Se usa mucho en peticiones tipo GET.



En conclusión, una URL es la dirección única que permite localizar y acceder a recursos en la web, formada por protocolo, servidor, puerto (opcional), ruta, recurso y parámetros de consulta.

b. ¿Qué componentes de la URL son opcionales?

En una URL, los componentes opcionales son:
- **Puerto:** si no se indica, se usa el 80 (HTTP) o 443 (HTTPS).
- **Recurso:** si no se especifica, el servidor carga por defecto index.html.
- **Query string:** solo aparece cuando se envían parámetros (ej. ?id=5).

Lo obligatorio siempre es el protocolo y el servidor (host).

c. ¿Cómo se especifica en el encabezado HTTP el recurso al que se quiere acceder?

En el encabezado HTTP, el recurso al que se quiere acceder se especifica en la línea de petición (request line).

Por ejemplo, si el usuario escribe la URL:

```
http://www.theserverside.com:80/discussions/java/threads.html
```

La petición HTTP enviada sería algo así:

```
GET /discussions/java/threads.html HTTP/1.1
Host: www.theserverside.com
```

- **GET** → es el método HTTP.
- **/discussions/java/threads.html** → es la parte de la URL que representa el recurso (path + recurso).
- **HTTP/1.1** → versión del protocolo.
- **Host: www.theserverside.com** → encabezado obligatorio para indicar el servidor al que se hace la petición.

d. ¿Cómo se diferencian los mensajes de requerimiento HTTP de los mensajes de respuesta?

Los mensajes de requerimiento HTTP y de respuesta HTTP se diferencian en su contenido:

Requerimiento HTTP (Request) -> indica qué se pide.
- Tiene un método (GET, POST, etc.).
- Indica el recurso a acceder (URL o path).
- Puede incluir parámetros de la petición.

Respuesta HTTP (Response) -> indica qué devuelve el servidor.
- Contiene un código de estado (200, 404, 500, etc.).
- Incluye un header con información, como el Content-Type que indica el tipo de dato devuelto.
- Aporta el contenido solicitado (HTML, imagen, etc.), que el navegador usa para mostrar la respuesta.

e. ¿Cómo sabe el servidor qué navegador está utilizando el visitante? (Revise la información provista por la herramienta para el desarrollador (F12) respecto de los Headers del Request)

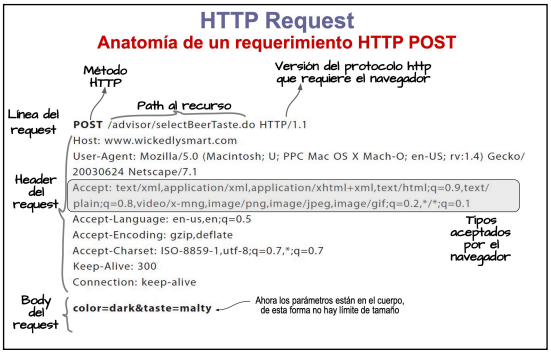
El servidor sabe qué navegador utiliza el visitante gracias al header User-Agent que el navegador envía en cada HTTP Request.

Ese campo dentro de los headers del request incluye información como:
- El nombre del navegador (Chrome, Firefox, Safari, etc.).
- La versión del navegador.
- El sistema operativo y su versión.

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/117.0.0.0 Safari/537.36
```

f. ¿Cómo se envían los parámetros en los mensajes con método POST?

En un requerimiento HTTP POST, los parámetros se envían en el cuerpo (body) del request. De esta forma no hay límite de tamaño para los datos, a diferencia de GET donde aparecen en la URL.



g. ¿Qué son las Cookies? ¿Cómo se envían las Cookies a través de HTTP?

Las cookies son pequeños archivos de texto que un servidor web guarda en la computadora del cliente a través del navegador.
Sirven para mantener información de estado entre diferentes peticiones HTTP, ya que el protocolo HTTP es sin estado (stateless).
Ejemplos de uso: recordar la sesión de un usuario, guardar preferencias, llevar el carrito de compras, etc.

El servidor envía una cookie al cliente mediante el header:	`Set-Cookie: nombre=valor; atributos`
Luego, en cada nueva petición al mismo servidor, el navegador devuelve la cookie automáticamente con el header:	`Cookie: nombre=valor`

De esta forma, el servidor puede identificar al usuario y mantener la continuidad de la sesión.

h. ¿Cuál es la diferencia entre HTTP y HTTPS? ¿Cuáles son los puertos por default (por defecto) de los mismos?

Diferencia entre HTTP y HTTPS

- **HTTP (HyperText Transfer Protocol):**
  Es el protocolo estándar de comunicación en la web. La información viaja en texto plano, lo que significa que puede ser interceptada fácilmente si alguien escucha la conexión. -> usa el **puerto 80** por defecto.
- **HTTPS (HyperText Transfer Protocol Secure):**
  Es la versión segura de HTTP. Funciona sobre SSL/TLS, lo que cifra la comunicación entre cliente y servidor. Esto asegura confidencialidad, integridad y autenticidad de los datos transmitidos. -> usa el **puerto 443** por defecto.

2.

a. La clasificación de los códigos de estado HTTP es la siguiente:

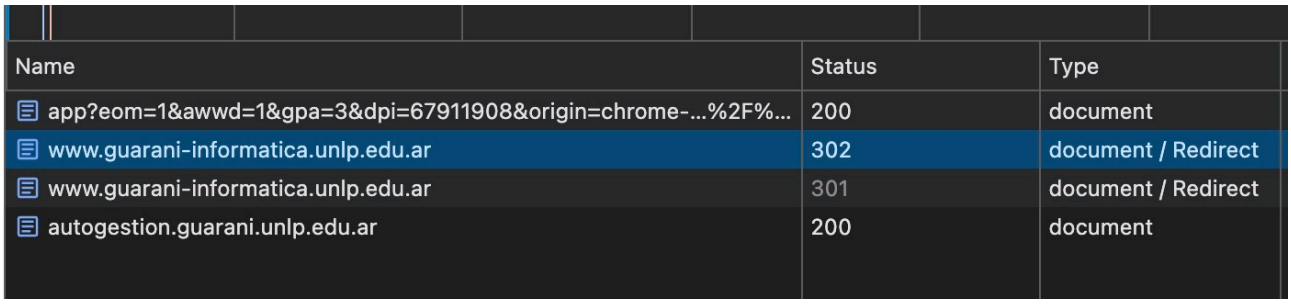
- **1xx – Respuestas informativas (Informational):** El servidor recibió la solicitud y el proceso continúa.
- **2xx – Respuestas satisfactorias (Successful):** La acción solicitada fue recibida, entendida y aceptada correctamente.
- **3xx – Redirecciones (Redirection):** Se requiere realizar más acciones por parte del cliente para completar la solicitud.
- **4xx – Errores del cliente (Client Error):** La solicitud contiene errores de sintaxis o no puede ser procesada por el servidor.
- **5xx – Errores del servidor (Server Error):** El servidor no pudo cumplir una solicitud aparentemente válida.

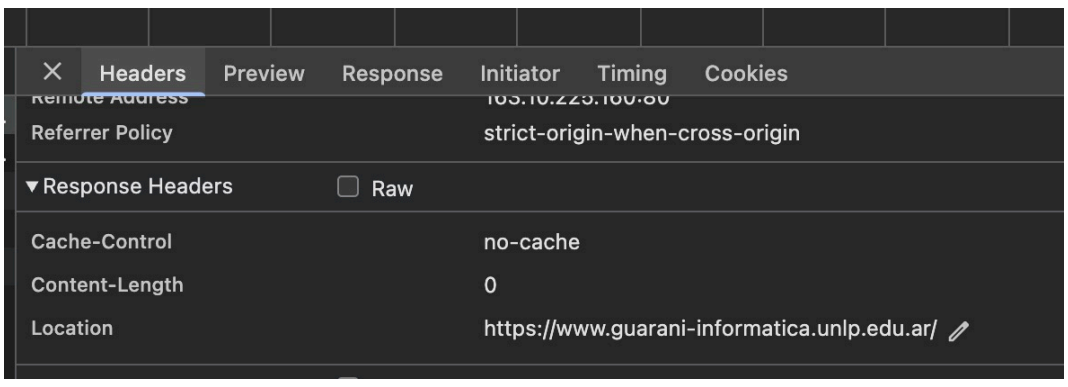
b.

Código	Descripción
200	**OK** – La solicitud ha tenido éxito. La información devuelta depende del método utilizado (por ejemplo, con GET se devuelve el recurso solicitado).
201	**Created** – La solicitud ha sido cumplida y ha resultado en la creación de un nuevo recurso.
204	**No Content** – El servidor ha cumplido la solicitud pero no devuelve contenido en el cuerpo de la respuesta.
206	**Partial Content** – El servidor devuelve solo una parte del recurso, como respuesta a una petición con cabecera Range.
301	**Moved Permanently** – El recurso solicitado se ha movido permanentemente a una nueva URL. El cliente debe usar la nueva dirección en el futuro.
302	**Found** – El recurso solicitado reside temporalmente en otra URL. El cliente debe usar la dirección original en futuras solicitudes.
304	**Not Modified** – Indica que el recurso no ha sido modificado desde la última vez que fue solicitado. Se utiliza con mecanismos de caché.
401	**Unauthorized** – La solicitud requiere autenticación. El cliente debe enviar credenciales válidas.
403	**Forbidden** – El servidor entendió la solicitud, pero se niega a cumplirla (no es un problema de autenticación).
404	**Not Found** – El servidor no encontró el recurso solicitado.
409	**Conflict** – La solicitud no pudo completarse debido a un conflicto con el estado actual del recurso.
500	**Internal Server Error** – Error interno del servidor. El servidor encontró una condición inesperada que le impidió cumplir la solicitud.

c. Al ingresar la URL **http://www.eldia.com** en el navegador, finalmente se muestra el código **200** (OK), ya que la página carga correctamente.
Sin embargo, antes de llegar a esa respuesta, el navegador recibe un **307** (Temporary Redirect) que lo redirige de manera automática hacia la versión segura **https://www.eldia.com**.
Por eso, aunque yo vea el **200** como resultado final, internamente ocurre primero el 307 que indica la redirección.

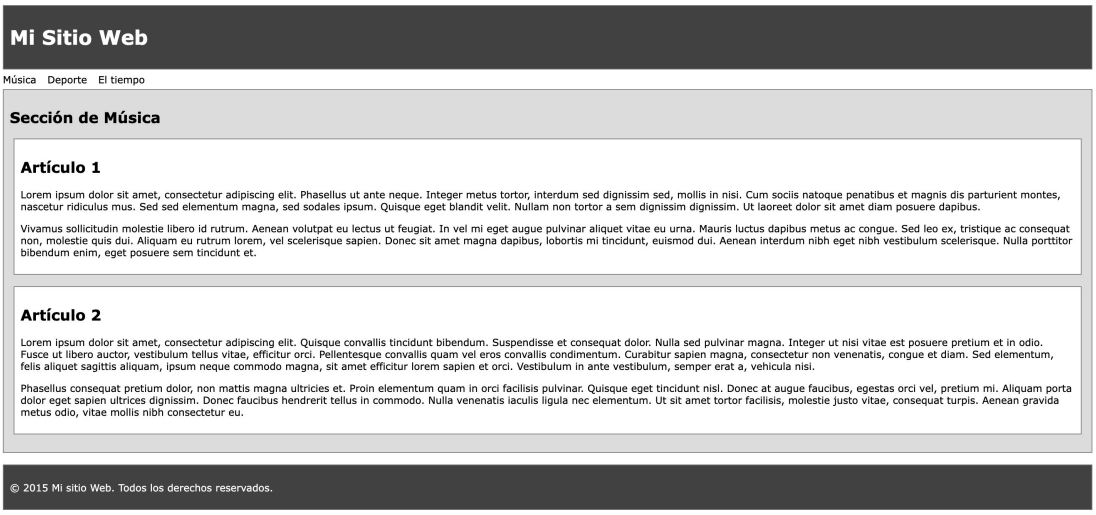
d. Cuando se ingresa la URL http://www.guarani-informatica.unlp.edu.ar, el servidor no entrega directamente la página, sino que responde primero con un código de estado de redirección (**301** o **302**).
En esa respuesta HTTP, el campo que indica la nueva dirección es el header (encabezado) **Location**.
Este campo Location contiene la URL de destino, que en este caso es la versión segura con **https://.**
El navegador lee ese encabezado y, de manera automática, redirige hacia la nueva dirección sin que el usuario tenga que hacer nada.





3. HTML 4

4. HTML 5



e. ¿En qué puerto está corriendo el servidor HTTP donde publicó (hosteó) su página?

Al visualizar mi página en Chrome, el URL que aparece es el siguiente:
http://localhost:63342/sitioestatico/paginav4.html?_ijt=vrkqoqeoofm9u63a3r3gefmvn3&_ij_reload=RELOAD_ON_SAVE

El servidor HTTP está corriendo en el puerto 63342, que aparece después de localhost: en la URL.

5.

Para ver el último visitante, usá el método GET con el parámetro formato=json que programaste.
http://localhost:8080/visitas50_war_exploded/Premio?formato=json

i. ¿Qué content-type debería utilizar para dar esta respuesta?

(i) Content-Type para JSON: application/json; charset=UTF-8 (ya seteado en doGet).

ii. **j. Para la declaración de un servlet se puede utilizar el archivo web.xml o anotaciones. ¿Cúal de los métodos utilizó en su proyecto'?. Duplique el proyecto y realice las modificaciones necesarias para tener ambas soluciones funcionando. ¿Qué método tiene prioridad sobre el otro si hubiera declarado el servlet de ambas maneras en un mismo proyecto?**

- Acá estás usando anotaciones (@WebServlet, @WebInitParam).
- Si duplicás el proyecto y usás web.xml, funciona igual.
- Prioridad: si en el mismo proyecto declararas el servlet por ambos métodos, web.xml tiene prioridad; y si pones metadata-complete="true" en <web-app>, el contenedor ignora las anotaciones.

Faltaría duplicar el proyecto (último inciso) pero en vez de usar anotaciones usar el archivo web.xml