👤 Datos del estudiante

Nombre: Julian Bastidas
Grupo: Solano
Fecha: 11/08/2025

1) Preguntas de comprensión

Rol del cliente y del servidor (con ejemplo):
Respuesta:

¿Qué ocurre desde que el cliente ingresa una URL hasta que ve la página? Menciona DNS, TCP/TLS y HTTP:
Respuesta:
El cliente es la aplicación o dispositivo que solicita un recurso o servicio, por ejemplo, un navegador web que pide una página HTML. El servidor es la máquina que procesa esa solicitud y envía la respuesta, por ejemplo, un servidor web que devuelve el contenido de index.html al navegador.

¿Por qué HTTP es sin estado (stateless)? ¿Cómo se gestionan las sesiones usualmente?
Respuesta:
HTTP es stateless porque cada petición se procesa de forma independiente, sin recordar interacciones anteriores. Las sesiones se gestionan mediante cookies, tokens o identificadores de sesión que el cliente envía en cada petición para mantener el contexto.

Diferencias clave entre HTTP y HTTPS. ¿Cuándo usar cada uno?
Respuesta:

- **HTTP:** Transmite datos en texto plano, puerto 80, inseguro ante interceptaciones.
- **HTTPS:** Transmite datos cifrados usando SSL/TLS, puerto 443, seguro.

¿Qué información crítica suelen contener las cabeceras de petición y respuesta?
Respuesta:

2) Diagrama de secuencia: `GET /productos`

Diagrama de secuencia

El diagrama de secuencia incluye: Navegador, DNS, Servidor Web, Base de Datos.
Señala método, ruta, código de estado y tiempos aproximados.

Descripción del diagrama:

1. El navegador envía petición DNS para resolver el dominio.
2. DNS responde con la IP del servidor.
3. El navegador establece conexión TCP y, si es HTTPS, realiza el handshake TLS.
4. Envía `GET /productos` al Servidor Web.
5. El servidor procesa y, si necesita datos, consulta a la Base de Datos.
6. La Base de Datos responde con la información solicitada.

7. El servidor genera la respuesta HTTP con código 200 y la envía al navegador.
8. El navegador renderiza el contenido.

3) Traza con DevTools

Sitio: https://jsonplaceholder.typicode.com/posts/1

URL	Método	Status	Tamaño	Content-Type	Tiempo total	Redirecciones
https://jsonplaceholder.typicode.com/posts/1	GET	200	292 B	application/json; charset=utf-8	~150 ms	0

4) Laboratorio con curl

Comandos ejecutados y salida resumida:

```
curl -i https://jsonplaceholder.typicode.com/posts/1
```

Salida (resumen):

```
HTTP/2 200
date: Tue, 12 Aug 2025 02:19:47 GMT
content-type: application/json; charset=utf-8
content-length: 292
access-control-allow-credentials: true
cache-control: max-age=43200
etag: W/"124-yiKdLzqO5gfBrJFrcdJ8Yq0LGnU"
expires: -1
nel: {"report_to":"heroku-
nel","response_headers":["Via"],"max_age":3600,"success_fraction":0.01,"failu
re_fraction":0.1}
pragma: no-cache
report-to: {"group":"heroku-
nel","endpoints":[{"url":"https://nel.heroku.com/reports?s=xek%2B%2FtHi6ATgM4
DdgK4dQaqZ8uaBr5Ucj13rpST3uLg%3D\u0026sid=e11707d5-02a7-43ef-b45e-
2cf4d2036f7d\u0026ts=1754422360"}],"max_age":3600}
reporting-endpoints: heroku-
nel="https://nel.heroku.com/reports?s=xek%2B%2FtHi6ATgM4DdgK4dQaqZ8uaBr5Ucj13
rpST3uLg%3D&sid=e11707d5-02a7-43ef-b45e-2cf4d2036f7d&ts=1754422360"
server: cloudflare
vary: Origin, Accept-Encoding
via: 2.0 heroku-router
x-content-type-options: nosniff
x-powered-by: Express
x-ratelimit-limit: 1000
x-ratelimit-remaining: 999
x-ratelimit-reset: 1754422389
age: 23676
cf-cache-status: HIT
cf-ray: 96dc7d62ccc23283-MIA
alt-svc: h3=":443"; ma=86400
```

```
{
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident occaecati excepturi optio
reprehenderit",
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et
cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt
rem eveniet architecto"
}
```

Análisis (método, status, cabeceras clave):

- **Método:** GET
- **Status:** 200
- **Cabeceras clave:** Content-Type, Content-Length, Date

```
curl -X POST \
  -H "Content-Type: application/json" \
  -d '{"title":"hola","body":"mundo","userId":1}' \
  https://jsonplaceholder.typicode.com/posts
```

Salida (resumen):

```
{
  "title": "hola",
  "body": "mundo",
  "userId": 1,
  "id": 101
}
```

Análisis (parámetros -i, -X, -H, -d y relación cuerpo/Content-Type):

- `-i` → Muestra cabeceras y cuerpo.
- `-X POST` → Define método POST.
- `-H` → Cabecera personalizada (Content-Type).
- `-d` → Cuerpo de la petición (JSON).
  El cuerpo coincide con el Content-Type: application/json

5) Mini-diseño de API – Recurso tareas

5.1 `GET /tareas`

- **Propósito:** Listar todas las tareas.
- **Parámetros:** `?completada=true|false`, `?page=1`, `?limit=10`
- **Respuesta:**

```
{
  "data": [
    { "id": 1, "titulo": "Comprar pan", "completada": false }
  ],
```

```
  "total": 1
}
```

- **Códigos:** 200, 500

5.2 `GET /tareas/:id`

- **Propósito:** Obtener una tarea específica.
- **Parámetros:** id (entero).
- **Respuesta:**

```
{ "id": 1, "titulo": "Comprar pan", "completada": false }
```

- **Códigos:** 200, 404, 500

5.3 `POST /tareas`

- **Propósito:** Crear una nueva tarea.
- **Body:**

```
{ "titulo": "Comprar pan", "completada": false }
```

- **Respuesta:**

```
{ "id": 1, "titulo": "Comprar pan", "completada": false }
```

- **Códigos:** 201, 400, 500

5.4 `PUT /tareas/:id`

- **Propósito:** Actualizar una tarea (idempotente: varias solicitudes con los mismos datos no cambian el resultado).
- **Body:**

```
{ "titulo": "Comprar pan", "completada": true }
```

- **Respuesta:**

```
{ "id": 1, "titulo": "Comprar pan", "completada": true }
```

- **Códigos:** 200, 400, 404, 500

5.5 `DELETE /tareas/:id`

- **Propósito:** Eliminar una tarea.
- **Parámetros:** id (entero).
- **Respuesta:** `{ "mensaje": "Tarea eliminada" }`
- **Códigos:** 200, 404, 500