

Proceso de selección - ProContacto

Gomez Wusinowski Jimena Rocío, jimenagomezwusi@hotmail.com

Preguntas generales sobre HTTP/HTTPS:

- ¿Qué es HTTP y cuál es su función principal?

Es un protocolo que permite la comunicación entre el cliente y el servidor a través de la web. Su función principal es la transferencia de información.

- ¿Cuál es la diferencia entre HTTP y HTTPS?

La diferencia está en que HTTPS protege la información, mientras que HTTP no.

- ¿Cómo funciona el proceso de cifrado en HTTPS?

Los datos se encriptan para que solo el receptor que esté autorizado pueda leerlos.

- ¿Qué es un certificado SSL/TLS y cuál es su importancia en HTTPS?

Es un documento digital que permite que el navegador y los usuarios verifiquen que el sitio al que entraron efectivamente es el que dice ser. Además, con este se pueden cifrar los datos de dicha web. Su importancia está en la seguridad que transmite (los usuarios pueden verlo fácilmente cuando hay un candado en la URL de la página que están visitando).

- ¿Qué es un método HTTP? ¿Podrías enumerar algunos de los más utilizados?

Es una instrucción que el cliente le manda al servidor para determinar qué acción queremos llevar a cabo sobre los datos. Considero que los más conocidos son los métodos GET y POST

- Explica las diferencias entre los métodos HTTP GET y POST.

Mientras GET se usa para obtener datos del servidor (los trae en la url), POST se utiliza para enviarle datos al mismo. Este último método es más seguro ya que los datos no viajan a simple vista.

- ¿Qué es un código de estado HTTP? ¿Podrías mencionar algunos de los más comunes y lo que significan?

Son números que indican el resultado de una solicitud. Entre los más comunes están el 404 (no encontrado), 503 (servicio no disponible), 200 (ok).

- ¿Qué es una cabecera HTTP? Da ejemplos de cabeceras comunes.

Es una parte del protocolo HTTP que se encarga de enviar información adicional en las solicitudes. Ejemplos: Content-Type, User-Agent.

- En qué consiste el concepto de "idempotencia" en los métodos HTTP? ¿Qué métodos cumplen con esta característica?

El concepto hace referencia a que por más que repitas varias veces el mismo método siempre va a dar el mismo resultado. Los métodos que cumplen con esto son GET, DELETE y PUT.

- ¿Qué es un redirect (redirección) HTTP y cuándo es utilizado?

Es cuando una URL lleva al usuario a otra URL, se puede utilizar cuando el contenido se mueve a una nueva ubicación

Preguntas técnicas y de seguridad en HTTP/HTTPS:

- ¿Cómo se asegura la integridad de los datos en una conexión HTTPS?

Se usan firmas y algoritmos que detectan si los datos fueron alterados.

- ¿Qué diferencia hay entre un ataque de "man-in-the-middle" y un ataque de "replay" en un contexto HTTPS?

NS / NC

- Concepto de "handshake" en HTTPS.

Es un proceso en el que el cliente y el servidor determinan una clave para cifrar la conexión.

- ¿Qué es HSTS (HTTP Strict Transport Security) y cómo mejora la seguridad de una aplicación web?

Es un estándar que obliga al navegador a acceder SOLO a páginas HTTPS, para prevenir el acceso a webs inseguras.

- ¿Qué es un ataque "downgrade" y cómo HTTPS lo previene?

Es cuando el atacante fuerza una conexión insegura utilizando HTTP o versiones viejas de HTTPS. Este lo previene bloqueando las mismas.

- ¿Qué es el CORS (Cross-Origin Resource Sharing) y cómo se implementa en una aplicación web?

Es un mecanismo que permite o restringe que una página pueda pedir datos de otro dominio. Se implementa configurando cabeceras HTTP en el servidor para permitir solicitudes de dominios específicos.

- ¿Qué diferencia hay entre una cabecera Authorization y una cabecera Cookie?

Mientras Authorization autentica al usuario, Cookie se encarga de mantener la sesión abierta en el navegador.

- ¿Qué son las cabeceras de seguridad como Content-Security-Policy o X-Frame-Options? ¿Cómo ayudan a mitigar ataques comunes?

Son configuraciones que protegen contra ataques restringiendo el contenido que puede cargarse en el navegador.

- ¿Cuáles son las diferencias entre HTTP/1.1, HTTP/2 y HTTP/3?

HTTP/1.1 maneja una solicitud a la vez

HTTP/2 permite hacer varias a la vez

HTTP/3 es aún más rápido.

- ¿Qué es un "keep-alive" en HTTP y cómo mejora el rendimiento de las aplicaciones?

Es una técnica que permite que la conexión entre el cliente y el servidor permanezca abierta incluso después de completar la solicitud.

Preguntas de implementación práctica:

- ¿Cómo manejarías la autenticación en una API basada en HTTP/HTTPS?
¿Qué métodos conoces (Basic, OAuth, JWT, etc.)?

NS/NC

- ¿Qué es un proxy inverso (reverse proxy) y cómo se utiliza en entornos HTTP/HTTPS?

Es un servidor que recibe solicitudes de los clientes y las redirige a otros servidores internos.

- ¿Cómo implementarías una redirección automática de HTTP a HTTPS en un servidor?

Configurando una regla en el servidor para que envíe a los usuarios de HTTP a HTTPS.

- ¿Cómo mitigarías un ataque de denegación de servicio (DDoS) en un servidor HTTP?

La implementación de firewalls me parece una buena opción.

- ¿Qué problemas podrías enfrentar al trabajar con APIs que dependen de HTTP, y cómo los resolverías?

Se pueden dar problemas de latencia y problemas con el CORS. Se puede resolver optimizando el servidor, y configurando bien las cabeceras.

- ¿Qué es un cliente HTTP? ¿Mencionar la diferencia entre los clientes POSTMAN y CURL?

Es una herramienta que permite hacer peticiones a los servidores. POSTMAN es un programa con interfaz gráfica, CURL solo usa línea de comandos.

Preguntas de GIT:

- ¿Qué es GIT y para qué se utiliza en desarrollo de software?

GIT es una herramienta para guardar versiones de nuestro código.

- ¿Cuál es la diferencia entre un repositorio local y un repositorio remoto en GIT?

El local se encuentra dentro de nuestra computadora, mientras que el remoto está en un servidor compartido, en "la nube".

- ¿Cómo se crea un nuevo repositorio en GIT y cuál es el comando para inicializarlo? Explica la diferencia entre los comandos `git commit` y `git push`.

Para crear un repo debes posicionarte en una carpeta, abrir el GIT Bash e ingresar el comando `git init`.

`git commit`: Guarda los cambios localmente.

`git push`: Sube los cambios al repositorio remoto.

- ¿Qué es un "branch" en GIT y para qué se utilizan las ramas en el desarrollo de software?

Son "copias" del proyecto donde podemos trabajar para no afectar a la rama principal.

- ¿Qué significa hacer un "merge" en GIT y cuáles son los posibles conflictos que pueden surgir durante un merge?

Hacer merge es unir el contenido de dos ramas distintas. Entre los conflictos que suelen aparecer, puede haber contenido superpuesto debido a que dos personas cambiaron la misma línea.

- Describe el concepto de "branching model" en GIT y menciona algunos modelos comunes (por ejemplo, Git Flow, GitHub Flow).

El "branching model" es una estrategia para gestionar ramas en un proyecto, para así organizar el trabajo en equipo. Además de esos dos modelos nombrados conozco GitLab Flow.

- ¿Cómo se deshace un cambio en GIT después de hacer un commit pero antes de hacer push?

Con el comando `git reset`.

- ¿Qué es un "pull request" y cómo contribuye a la revisión de código en un equipo?

Es una solicitud para revisar y aprobar cambios antes de mergearlos al proyecto.

- ¿Cómo puedes clonar un repositorio de GIT y cuál es la diferencia entre `git clone` y `git pull`?

Para clonar un repositorio a nuestra computadora abrimos el GIT Bash donde deseemos guardar el mismo, y utilizamos el comando `git clone <url repositorio remoto>`. Mientras `git clone` trae el proyecto completo desde el repositorio remoto a la PC, `git pull` trae solo las actualizaciones.

Preguntas de Node.js:

- ¿Qué es Node.js y por qué es una opción popular para el desarrollo backend?

Es un entorno de ejecución, y es una opción popular debido a que permite ejecutar JavaScript en el servidor, y no necesariamente en la web.

- ¿Cómo funciona el modelo de I/O no bloqueante en Node.js y cómo beneficia el rendimiento de una aplicación backend?

Este modelo permite que el servidor maneje varias solicitudes en paralelo sin esperar a que terminen, lo cual mejora el rendimiento.

- ¿Qué es el Event Loop en Node.js y cuál es su papel en la ejecución de código asíncrono?

Es el encargado de gestionar el código asíncrono sin detener el programa, permitiendo que se realicen varias tareas a la vez sin esperar a que terminen las anteriores.

- ¿Cuál es la diferencia entre `require()` y `import` en Node.js?

`require()` es el método de importación en CommonJS, `import` es el estándar de ES6.

- ¿Qué es npm y cuál es su función en el ecosistema de Node.js?

npm es el gestor de paquetes de Node.js, y permite instalar dependencias.

- ¿Cómo se inicializa un proyecto de Node.js usando npm y cuál es el propósito del archivo `package.json`?

Para inicializar un proyecto usando npm, se ejecuta `npm init` en la terminal, y el archivo `package.json` guarda la información del proyecto, las dependencias y los scripts necesarios para gestionarlo.

- ¿Qué son las dependencias en npm y cómo se instalan? Explica la diferencia entre dependencias y dependencias de desarrollo.

Son paquetes que se necesitan para que el proyecto funcione correctamente. Se instalan con `npm install <nombre_paquete>`. Las dependencias normales se necesitan para usar la aplicación desarrollada, y las de desarrollo solo mientras se escribe el código.

- ¿Cómo puedes gestionar versiones específicas de paquetes en npm y para qué sirve el archivo `package-lock.json`?

Podes indicar la versión que querés en el `package.json`. El archivo `package-lock.json` sirve para asegurar que se usen exactamente las mismas versiones de las dependencias en cada instalación, así no hay problemas cuando se desarrolla en dos computadoras o entornos diferentes.

- ¿Qué es `nest.js` y cómo se usa en Node.js para construir aplicaciones backend?

Es un framework de Node.js que se usa organizando el backend en controladores (para rutas), servicios (para la lógica), y módulos (para agrupar todo), lo que lo hace ideal para crear APIs.

- ¿Cómo se manejan errores en Node.js y cuál es la diferencia entre callbacks, promesas y `async/await` para manejar código asíncronico?

Los errores se manejan con `callbacks` pasando el error como primer argumento, con `promesas` usando `.catch()`, y con `async/await` dentro de un bloque `try/catch`.

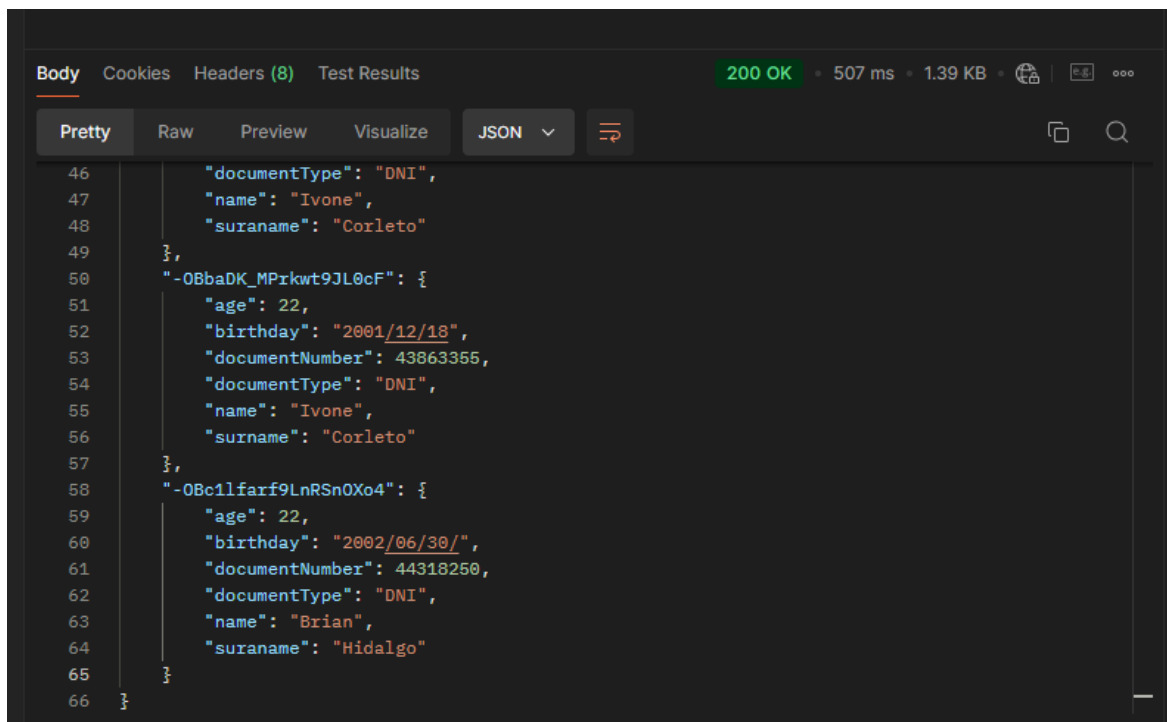
Callbacks: son más complicados y difíciles de manejar cuando hay muchos.

Promesas: organizan mejor el código y hacen más fácil manejar errores.

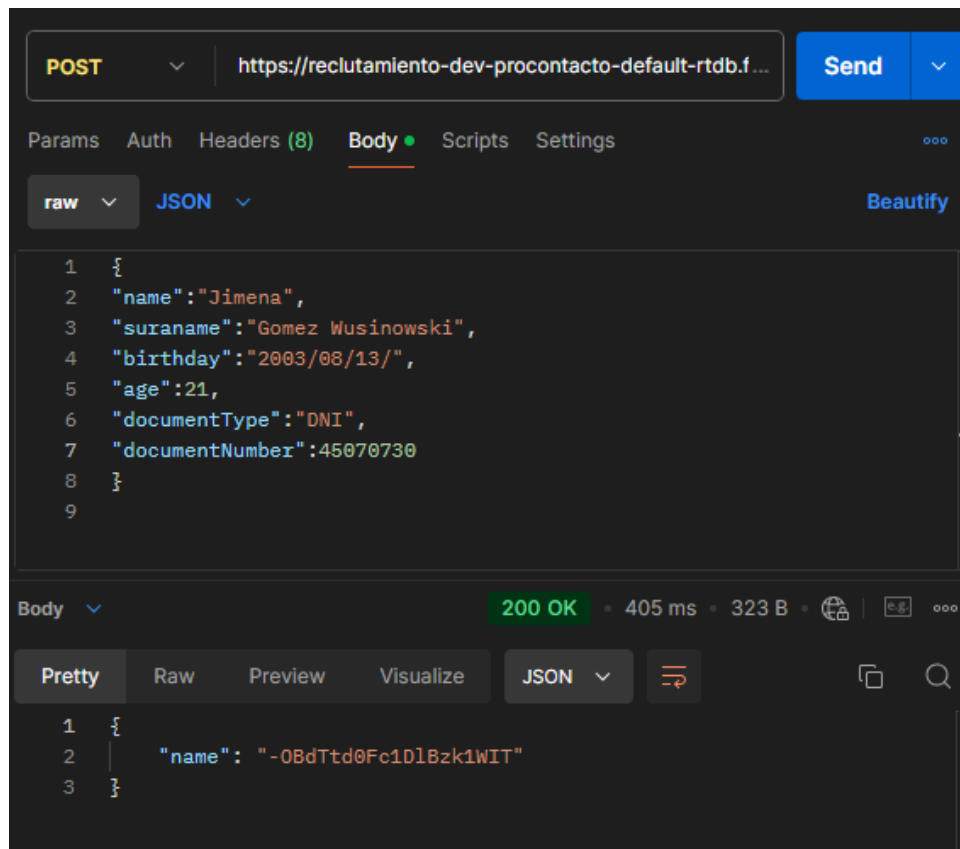
Async/await: la opción más limpia y fácil de leer para manejar código asíncronico.

Actividad práctica número 1

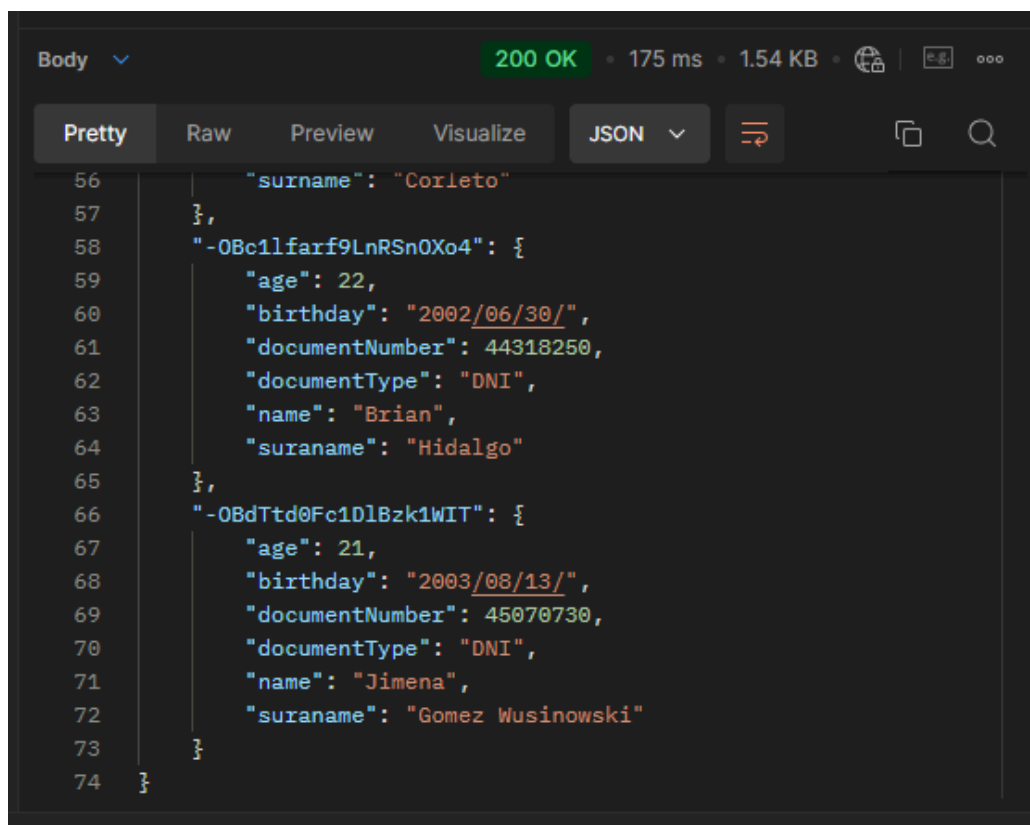
1.



```
46     "documentType": "DNI",
47     "name": "Ivone",
48     "surname": "Corleto"
49   },
50   "-0BbaDK_MPrkwt9JL0cF": {
51     "age": 22,
52     "birthday": "2001/12/18",
53     "documentNumber": 43863355,
54     "documentType": "DNI",
55     "name": "Ivone",
56     "surname": "Corleto"
57   },
58   "-0Bc1lfarf9LnRSn0Xo4": {
59     "age": 22,
60     "birthday": "2002/06/30",
61     "documentNumber": 44318250,
62     "documentType": "DNI",
63     "name": "Brian",
64     "surname": "Hidalgo"
65   }
66 }
```



2. Al realizar el GET por segunda vez, se agrega el body en formato JSON con mis datos.



Actividad práctica número 2

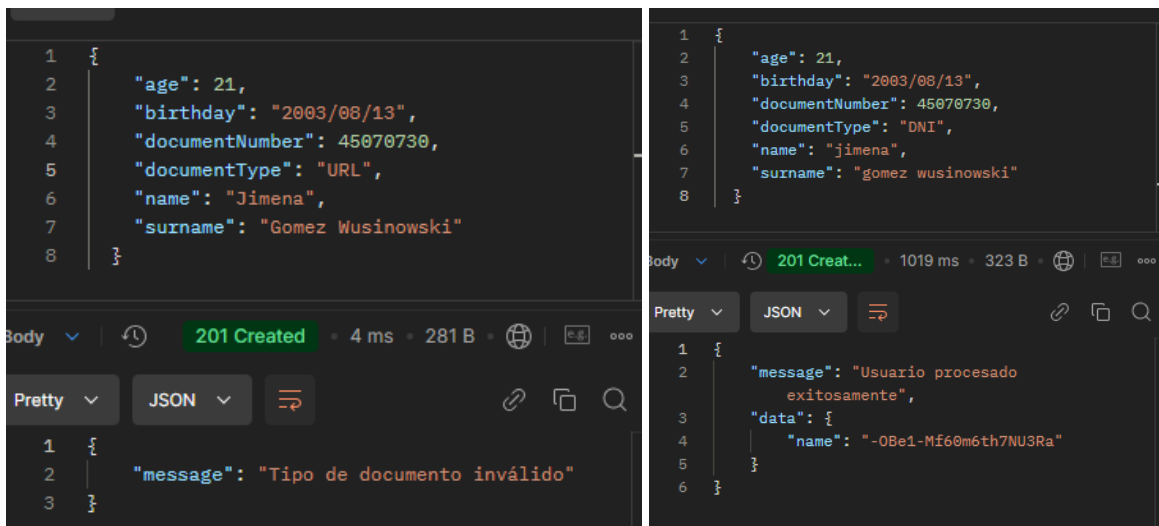
```
JS ejercicioDos.js > ...
1  const https = require('https');
2  const url = 'https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json';
3
4  https.get(url, (response) => {
5    let datosRespuesta = '';
6
7    response.on('data', (chunk) => {
8      datosRespuesta += chunk; // Cada vez que llega un pedazo de datos, lo concatenamos a la variable
9    });
10
11   response.on('end', () => {
12     console.log(JSON.parse(datosRespuesta));
13   });
14
15 })
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
documentNumber: 43863355,
documentType: 'DNI',
name: 'Ivone',
surname: 'Corleto'
},
'-0Bc1lfarf9LnRSnOXo4': {
  age: 22,
  birthday: '2002/06/30/',
  documentNumber: 44318250,
  documentType: 'DNI',
  name: 'Brian',
  suraname: 'Hidalgo'
},
'-0BdTtd0Fc1D1Bzk1WIT': {
  age: 21,
  birthday: '2003/08/13/',
  documentNumber: 45070730,
  documentType: 'DNI',
  name: 'Jimena',
  suraname: 'Gomez Wusinowski'
}
}
}
PS C:\Users\pc\Desktop\Postulacion-ProContacto>
```

Actividad práctica número 3

<pre>1 { 2 "age": 21, 3 "birthday": "1899/08/13", 4 "documentNumber": 45070730, 5 "documentType": "DNI", 6 "name": "Jimena", 7 "surname": "Gomez Wusinowski" 8 }</pre>	<pre>1 { 2 "age": "Veintiuno", 3 "birthday": "2003/08/13", 4 "documentNumber": 45070730, 5 "documentType": "DNI", 6 "name": "Jimena", 7 "surname": "Gomez Wusinowski" 8 }</pre>
<p>Body 201 Created 4 ms 299 B</p> <p>Pretty JSON</p> <pre>1 { 2 "message": "Formato de fecha inválido o 3 fecha incorrecta" 3 }</pre>	<p>Body 201 Created 5 ms 286 B</p> <p>Pretty JSON</p> <pre>1 { 2 "message": "Edad deberia ser de tipo 3 Integer" 3 }</pre>



```
84 },
85 "-0Be1-Mf60m6th7NU3Ra": {
86   "age": 21,
87   "birthday": "2003/08/13",
88   "documentNumber": 45070730,
89   "documentType": "DNI",
90   "name": "Jimena",
91   "surname": "Gomez Wusinowski"
92 }
93 }
```

Tanto nombre como apellido se cargaron con mayúsculas pese a no tenerlas originalmente.