Preguntas generales sobre HTTP/HTTPS

¿Qué es HTTP y cuál es su función principal? Un HTTP es un comunicador con el cliente y su función es básicamente comunicarse el cliente y el servicio web.

¿Cuál es la diferencia entre HTTP y HTTPS? La seguridad.

¿Cómo funciona el proceso de cifrado en HTTPS? Es un proceso en el cual el cliente se pone de acuerdo con el servidor usando un protocolo llamado SSL/TLS que está diseñado para proteger la comunicación. Es como un idioma secreto que nadie entiende salvo ellos, si alguien del exterior quisiera leer va a ver solo símbolos sin sentido.

¿Qué es un certificado SSL/TLS y cuál es su importancia en HTTPS? Es una especie de documento electrónico que se instala en el servidor para verificar que el sitio con el que estás hablando es real y para cifrar la información, te asegura que la conexión sea cifrada y verificada.

¿Qué es un método HTTP? ¿Podrías enumerar algunos de los más utilizados? Es la forma en la que el cliente le dice al servidor que acción quiere realizar. Los más comunes son: get, post, put, delete, etc.

Explica las diferencias entre los métodos HTTP GET y POST. El Get es para obtener datos y el Post para enviar datos o crear algo nuevo.

¿Qué es un código de estado HTTP? ¿Podrías mencionar algunos de los más comunes y lo que significan? Es un código de 3 dígitos que indican el estado de la solicitud. Los más comunes son: 200 OK, 201 Created, 400 Bad Request, 500 Internal Server Error, etc

¿Qué es una cabecera HTTP? Da ejemplos de cabeceras comunes. Es una información adicional, los más comunes son: content type, authorization, user-agent, etc.

¿En qué consiste el concepto de "idempotencia" en los métodos HTTP? ¿Qué métodos cumplen con esta característica? Un método es idempotente si al hacer la misma acción varias veces, al servidor no le causa nada. No es idempotente si te tira error o causa problemas. Cumplen con la idempotencia el GET, PUT, DELETE pero no simple el POST (podes crear veces el mismo elemento).

¿Qué es un redirect (redirección) HTTP y cuándo es utilizado? Es una especie de redireccionamiento del servidor, se utiliza para reubicar recursos o guiar al usuario de una URL a la otra.

Preguntas técnicas y de seguridad en HTTP/HTTPS:

¿Cómo se asegura la integridad de los datos en una conexión HTTPS? Se asegura utilizando el protocolo TLS (Transport Layer Security). No es suficiente que los datos están encriptados sino también te asegura que nadie los manipule en el camino:

¿Qué diferencia hay entre un ataque de "man-in-the-middle" y un ataque de "replay" en un contexto HTTPs? El primero es como un espia que se mete entre las dos partes y altera la comunicación, y el segundo repite algo para obtener un beneficio (pagas algo y te vuelve a mandar para intentar que se pague de vuelta)

Explica el concepto de "handshake" en HTTPS. Es el proceso inicial entre el cliente y el servidor donde se acuerdan las reglas para poder confiar mutuamente.

¿Qué es HSTS (HTTP Strict Transport Security) y cómo mejora la seguridad de una aplicación web? Es una política de seguridad que forza al cliente a usar el HTTPS, lo mejora drásticamente ya que te asegura una conexión confiable, te asegura que nunca se acceda de forma insegura como por ejemplo al banco.

¿Qué es un ataque "downgrade" y cómo HTTPS lo previene? El atacante fuerza a que una conexión use versiones antiguas o inseguras para poder interceptar y manipular todo fácilmente. Se previene con el TLS que no permite usar versiones antiguas y si el servidor implementa HSTL nunca jamás vuelve a usar HTTP.

¿Qué es el CORS (Cross-Origin Resource Sharing) y cómo se implementa en una aplicación web? Es un mecanismo de seguridad que controla qué recursos pueden ser accedidos de otro origen. Se implementa cuando el servidor incluye cabeceras HTTP en su respuesta para indicar que permite que otros dominios accedan a ellos.

¿Qué diferencia hay entre una cabecera Authorization y una cabecera Cookie? La diferencia es que Authorization se usa para apis modernas que usan tokens y la cookie es como una notita que el navegador guarda y le muestra al servidor, ideal para sesiones.

¿Qué son las cabeceras de seguridad como Content-Security-Policy o X-Frame-Options? - ¿Cómo ayudan a mitigar ataques comunes? Son cabeceras de seguridad que ayudan a prevenir ataques comunes, se previene ya que el content le dice al navegador donde puede cargar sus cosas y el X-Frame evita que otro sitio cargue el tuyo en secreto.

¿Cuáles son las diferencias entre HTTP/1.1, HTTP/2 y HTTP/3? Son versiones del protocolo HTTP, la HTTP/1.1 fue el estándar por mucho tiempo pero quedó lento, HTTP/2 mejoró mucho la velocidad usando una sola conexión para todo, y HTTP/3 directamente cambia la forma de conectarse usando otra tecnología más rápida.

¿Qué es un "keep-alive" en HTTP y cómo mejora el rendimiento de las aplicaciones? Una función del protocolo HTTP/1.1, es como abrir una nueva conversación con el servidor sin cortar, te ahorra tiempo y el sitio carga más rápido.

Preguntas de implementación práctica:

¿Cómo manejarías la autenticación en una API basada en HTTP/HTTPS? ¿Qué métodos conoces (Basic, OAuth, JWT, etc.)? Hay varios métodos para autenticar en una API, como enviar usuario y contraseña (Basic), usar tokens (como JWT), o usar claves especiales (API Key). JWT es el más común hoy en día porque es seguro, liviano y fácil de usar en APIs REST.

¿Qué es un proxy inverso (reverse proxy) y cómo se utiliza en entornos HTTP/HTTPS? Un proxy inverso es como un intermediario entre el usuario y los servidores. Recibe los pedidos, decide a qué servidor enviarlos, y después le pasa la respuesta al usuario. Sirve para repartir la carga, proteger los servidores, y hacer que todo funcione más rápido y seguro.

¿Cómo implementarías una redirección automática de HTTP a HTTPS en un servidor? Cuando alguien entra a un sitio usando HTTP, el servidor lo redirige automáticamente a la versión segura (HTTPS). Esto se hace con configuraciones en el servidor como Nginx o Apache.

¿Cómo mitigarías un ataque de denegación de servicio (DDoS) en un servidor HTTP? Un DDoS es como si muchas personas entraran a la vez a una tienda para que nadie más pueda usarla. Para evitarlo, se pueden poner límites a cuántas veces alguien puede entrar, usar filtros automáticos, o protegerse con servicios como Cloudflare que bloquean el tráfico raro antes de que llegue.

¿Qué problemas podrías enfrentar al trabajar con APIs que dependen de HTTP, y cómo los resolverías? Al usar APIs puede haber problemas como errores por muchas peticiones, demoras, seguridad si no usan HTTPS, o fallos por CORS. La solución depende del problema, pero muchas veces se resuelven optimizando el código, configurando bien el servidor, o manejando los errores de forma inteligente.

¿Qué es un cliente HTTP? ¿Mencionar la diferencia entre los clientes POSTMAN y CURL? Un cliente HTTP es una herramienta para enviar pedidos a una API. Postman es más visual y cómodo para probar cosas rápido, y CURL se usa mediante una línea de comandos, ideal para automatizar o trabajar desde código.

Preguntas de GIT

¿Qué es GIT y para qué se utiliza en desarrollo de software? Git es una herramienta que usamos para guardar y controlar todos los cambios que hacemos en el código de un proyecto. Sirve para que podamos trabajar en equipo sin pisarnos el trabajo, y también para volver atrás si algo no funciona

¿Cuál es la diferencia entre un repositorio local y un repositorio remoto en GIT? El repositorio local es el que tengo en mi computadora, y el remoto es el que está en internet,

como en GitHub. Yo trabajo en un local, y cuando quiero compartir o guardar mi trabajo con el equipo, lo subo al remoto.

¿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 empezar un proyecto con Git uso git init, que crea el repositorio. Cuando hago cambios y los quiero guardar, uso git commit, y si quiero subirlos a GitHub, uso git push.

¿Qué es un "branch" en GIT y para qué se utilizan las ramas en el desarrollo de software? Un branch es como una copia del proyecto donde puedo trabajar en una parte del código sin tocar lo que ya está funcionando. Así puedo hacer cambios, probar cosas nuevas o arreglar errores sin romper nada.

¿Qué significa hacer un "merge" en GIT y cuáles son los posibles conflictos que pueden surgir durante un merge? Hacer un merge es juntar los cambios que hice en una rama con la rama principal. A veces hay conflictos si dos personas cambian lo mismo, y ahí tengo que decidir qué parte del código queda.

Describe el concepto de "branching model" en GIT y menciona algunos modelos comunes (por ejemplo, Git Flow, GitHub Flow). Un modelo de branching es una forma de organizar cómo usamos las ramas en el equipo. Por ejemplo, Git Flow tiene ramas para cada tipo de tarea (como arreglos o nuevas funciones), y GitHub Flow es más simple y directo.

¿Cómo se deshace un cambio en GIT después de hacer un commit pero antes de hacer push? Si ya hice un commit pero todavía no lo subí a GitHub, puedo usar git reset para deshacerlo. Dependiendo del caso, puedo conservar los cambios o borrarlos completamente. También puedo modificar el commit con git commit --amend

¿Qué es un "pull request" y cómo contribuye a la revisión de código en un equipo? Un pull request es pedir que revisen mi código antes de juntarlo con el proyecto principal. Así otros pueden dar su opinión, corregir errores o mejorar lo que hice.

¿Cómo puedes clonar un repositorio de GIT y cuál es la diferencia entre git clone y git pull? Con git clone bajo una copia del proyecto desde GitHub a mi compu. Después, cuando ya tengo ese proyecto y quiero actualizarlo con los cambios nuevos, uso git pull.

Preguntas de NODE.JS

¿Qué es Node.js y por qué es una opción popular para el desarrollo backend? Node.js me deja usar JavaScript del lado del servidor. Es muy rápido porque usa el motor de Chrome, y además me permite manejar muchas peticiones al mismo tiempo sin que se trabe. Por eso se usa mucho en backend.

¿Cómo funciona el modelo de I/O no bloqueante en Node.js y cómo beneficia el rendimiento de una aplicación backend? Node no se queda esperando cuando hace tareas lentas cómo leer archivos o consultar una base de datos. Mientras tanto, sigue trabajando.

Eso hace que sea muy rápido y pueda atender muchas cosas a la vez sin bloquearse.

¿Qué es el Event Loop en Node.js y cuál es su papel en la ejecución de código asincrónico? ¿Cuál es la diferencia entre require() y import en Node.js? El Event Loop es como un encargado que se fija qué tareas se pueden ejecutar. Mientras espera que terminen las cosas lentas, sigue trabajando. Así Node puede manejar muchas tareas sin bloquearse. La diferencies que require() es la forma vieja de importar cosas en Node, y import es la forma nueva. Las dos sirven para lo mismo, pero import necesita que configure el proyecto como módulo en el package.json.

¿Qué es npm y cuál es su función en el ecosistema de Node.js? npm (Node Package Manager) es el sistema que uso para instalar librerías o herramientas que necesita mi proyecto en Node. También sirve para compartir código y manejar todas las dependencias desde un solo lugar.

¿Cómo se inicializa un proyecto de Node.js usando npm y cuál es el propósito del archivo package.json? Cuando arranco un proyecto en Node, uso npm init, que me crea un archivo llamado package.json. Este archivo guarda toda la información del proyecto y las librerías que usa, así otras personas pueden instalarlo fácilmente.

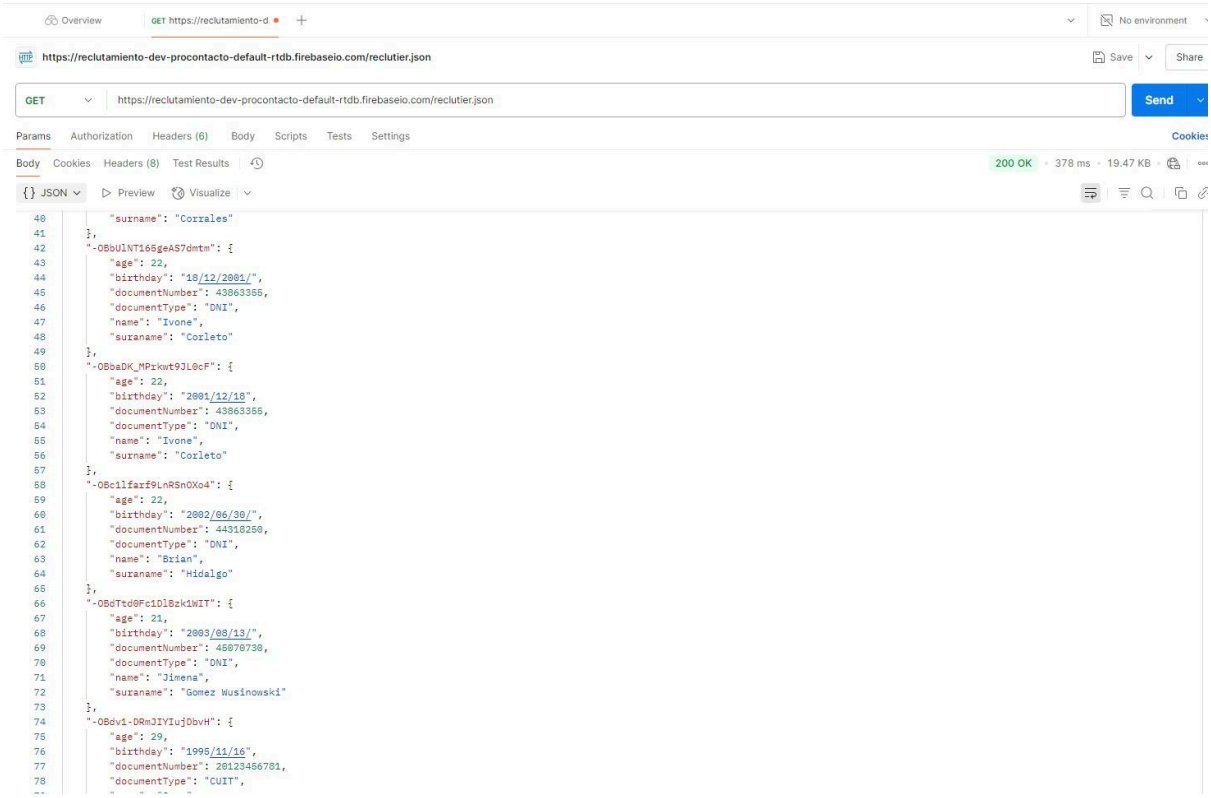
¿Qué son las dependencias en npm y cómo se instalan? Explica la diferencia entre dependencias y dependencias de desarrollo. Las dependencias son las librerías que necesita mi proyecto para funcionar. Y las dependencias de desarrollo son las que solo uso mientras programo, como para testear o revisar el código.

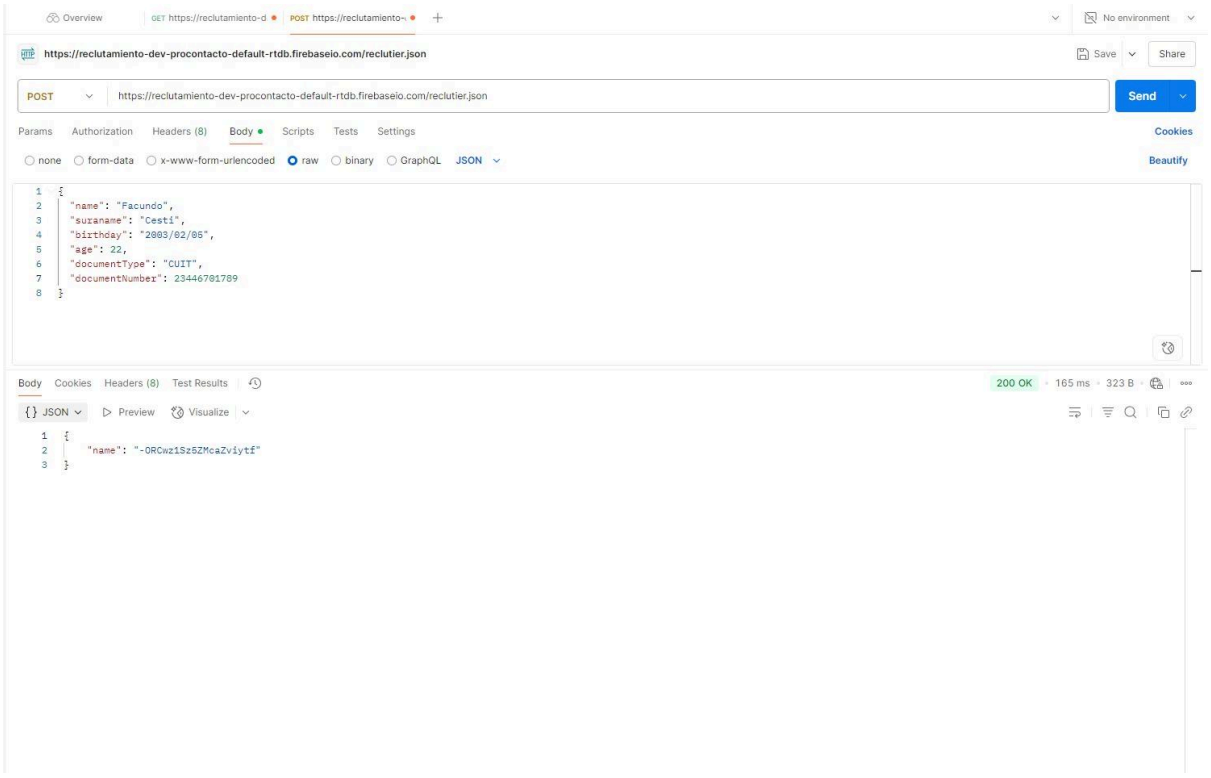
¿Cómo puedes gestionar versiones específicas de paquetes en npm y para qué sirve el archivo package-lock.json? Con npm puedo elegir una versión específica de una librería para asegurarme que funcione bien. Y el package-lock.json guarda todas las versiones exactas, así el proyecto siempre se instala igual en cualquier compu.

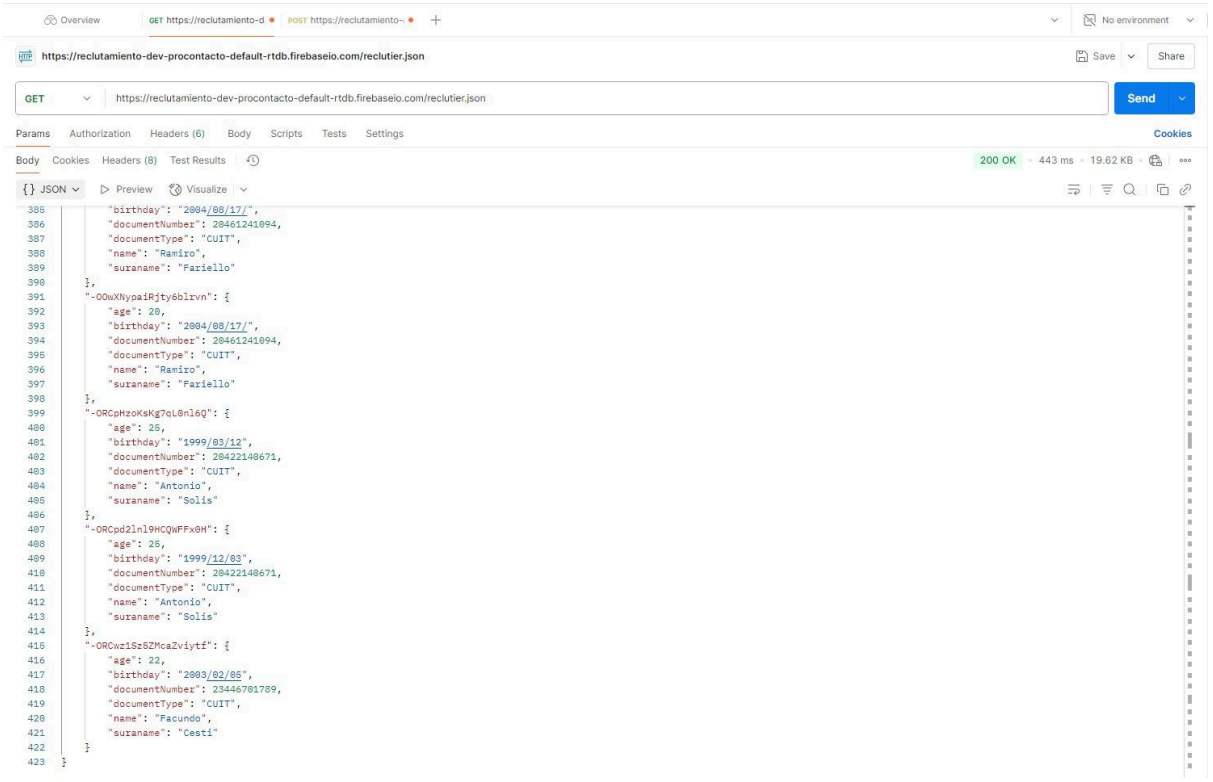
¿Qué es nest.js cómo se usa en Node.js para construir aplicaciones backend? NestJS es un framework para crear APIs en Node.js, pero de forma más ordenada y profesional. Me deja separar el código en partes como controladores y servicios, y es ideal para proyectos grandes, porque ya trae muchas herramientas listas para usar.

¿Cómo se manejan errores en Node.js y cuál es la diferencia entre callbacks, promesas y async/await para manejar código asincrónico? Node tiene varias formas de manejar tareas que tardan, como leer archivos o consultar una base de datos. Antes se usaban callbacks, pero eran complicados. Ahora se usan promesas o async/await, que hacen el código más claro. Para manejar errores, uso try/catch con async/await.

Preguntas/Ejercicios:
1. Adjuntar imagenes del response de un GET y de un POST de cada punto
2. ¿Qué sucede cuando hacemos el GET por segunda vez, luego de haber ejecutado el POST?


Overview
GET https://reclutamiento-d
No environment
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json
Save
Share
GET
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json
Send
Params
Authorization
Headers (6)
Body
Scripts
Tests
Settings
Cookies
Body
Cookies
Headers (8)
Test Results
200 OK
378 ms
19.47 KB
JSON
Preview
Visualize
"surname": "Corrales"
},
"-OBbUlNT165geAS7dmtm": {
"age": 22,
"birthday": "18/12/2001/",
"documentNumber": 43863355,
"documentType": "DNI",
"name": "Ivone",
"suraname": "Corleto"
},
"-OBbaDK_MPrkwt9JL0cF": {
"age": 22,
"birthday": "2001/12/18",
"documentNumber": 43863355,
"documentType": "DNI",
"name": "Ivone",
"surname": "Corleto"
},
"-OBc1lfarf9LnRSnOXo4": {
"age": 22,
"birthday": "2002/06/30/",
"documentNumber": 44318280,
"documentType": "DNI",
"name": "Brian",
"suraname": "Hidalgo"
},
"-OBdTtd0Fc1DlBzk1WIT": {
"age": 21,
"birthday": "2003/08/13/",
"documentNumber": 45070730,
"documentType": "DNI",
"name": "Jimena",
"suraname": "Gomez Wusinowski"
},
"-OBdv1-DRmJIYIujDbvH": {
"age": 29,
"birthday": "1995/11/16",
"documentNumber": 20123456781,
"documentType": "CUIT",


Overview
GET https://reclutamiento-d
POST https://reclutamiento-
No environment
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json
Save
Share
POST
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json
Send
Params
Authorization
Headers (8)
Body
Scripts
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Facundo",
"suraname": "Cesti",
"birthday": "2003/02/05",
"age": 22,
"documentType": "CUIT",
"documentNumber": 23446701789
}
Body
Cookies
Headers (8)
Test Results
200 OK
165 ms
323 B
JSON
Preview
Visualize
{
"name": "-ORCwz1Sz5ZMcaZviytf"
}



Después de hacer el POST, cuando hago el GET otra vez, aparece el dato nuevo que mandé. Eso muestra que el sistema guardó lo que envié correctamente.