Introducción a las preguntas teóricas:

Como sabemos, en la actualidad existen grandes modelos de inteligencia artificial con la capacidad de responder la mayoría de preguntas que podamos realizar como reclutadores. Si bien las respuestas van a ser evaluadas, la finalidad de la parte teórica es ser una guía para que el postulante pueda entender la orientación de la posición y afianzarse con los conocimientos necesarios para su desarrollo laboral.

Posteriormente a la evaluación del trabajo práctico, podrá existir una instancia de conversación donde validemos los conocimientos y el entendimiento de los conceptos.

Teoría:

Preguntas generales sobre HTTP/HTTPS:

¿Qué es HTTP y cuál es su función principal?

¿Cuál es la diferencia entre HTTP y HTTPS?

¿Cómo funciona el proceso de cifrado en HTTPS?

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

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

Explica las diferencias entre los métodos HTTP GET y POST.

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

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

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

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

Preguntas técnicas y de seguridad en HTTP/HTTPS:

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

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

Explica el concepto de "handshake" en HTTPS.

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

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

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

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

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

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

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

Preguntas de implementación práctica:

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

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

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

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

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

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

Preguntas de GIT

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

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

¿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.

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

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

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

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

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

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

Preguntas de GIT

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

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

¿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?

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

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

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

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

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

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

Práctica

Para realizar los ejercicios prácticos deberás contar en tu ambiente de trabajo con las siguientes herramientas:

- Postman
- GIT
- Node.js instalado

La entrega deberá realizarse en un repositorio de código público que se deberá compartir al equipo de reclutamiento. El repositorio deberá contener tanto la parte teórica como la práctica

Actividad práctica número 1
Pasos:

1) Realizar una petición GET a la siguiente URL a través de Postman:
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json

Ejemplo con CURL:

curl --location --request GET
'https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json' \
--header 'Content-Type: application/json'

2) Realizar una petición POST a la siguiente URL a través de Postman:
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json
y con el siguiente body:

```
{
   "name":"TuNombre",
   "suraname":"TuApellido",
   "birthday":"1995/11/16/",
   "age":29,
   "documentType":"CUIT",
   "documentNumber":20123456781
}
```

Reemplazar los campos por los valores personales tuyos.

3) Volver a realizar el GET del punto número 1.

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?

Actividad práctica número 2

Realizar un script en Node.JS que realice un GET a la URL:
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json y muestre por pantalla todos los registros.

Actividad práctica número 3:

Realizar un Servicio Web en nestjs que permita recibir una petición post con el formato:

```
{
   "name":"TuNombre",
   "suraname":"TuApellido",
   "birthday":"1995/11/16/",
   "age":29,
   "documentType":"CUIT",
   "documentNumber":20123456781
}
```

una vez recibida la petición deberá ejecutar otra petición hacia el servicio web de reclutamiento:
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json

Adicionalmente el servicio web de nestjs deberá tener las siguientes características:

1. El campo Name y Surename deberán empezar siempre con la primer letra de cada palabra en mayúscula, si se recibe una petición con otro formato deberá normalizarse al formato esperado
2. Validar que el campo birthday tenga un formato válido en el formato YYYY/MM/DD. La fecha proporcionada no podrá ser posterior al día de hoy ni anterior al 1900/01/01. En caso que no se cumpla alguna petición se deberá rechazar la petición
3. El campo age deberá ser un número entero. En caso que no se cumpla alguna petición se deberá rechazar la petición
4. Los valores posibles de documentType son: “CUIT” o “DNI” si se envía otros valores se deberá rechazar la petición