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?
HTTP (Hypertext Transfer Protocol) es un protocolo de la capa de aplicación ysu funcion es definir las reglas de comunicación para la transferencia de datos entre un navegador y un servidor web.

¿Cuál es la diferencia entre HTTP y HTTPS?
La diferencia que presentas es que en HTTPS, el navegador y el servidor establecen una conexión segura y cifrada antes de transferir datos.

¿Cómo funciona el proceso de cifrado en HTTPS?
El proceso de cifrado en HTTPS funciona de la siguiente manera:

- El navegador web se conecta a un sitio web que utiliza HTTPS.
- El sitio web envía al navegador su certificado SSL o TLS, que contiene la clave pública necesaria para iniciar la sesión segura.
- El navegador utiliza el certificado para autenticarse en el sitio web.
- El protocolo de cifrado Transport Layer Security (TLS) cifra las comunicaciones entre el navegador y el servido

¿Qué es un certificado SSL/TLS y cuál es su importancia en HTTPS?
Un certificado SSL/TLS son certificados digitales que permiten a los navegadores web identificar y establecer conexiones cifradas con sitios web. Son importantes porque protegen los datos personales y distintos tipos de información confidencial, como números de tarjetas de crédito, direcciones, contraseñas, etc.

¿Qué es un método HTTP? ¿Podrías enumerar algunos de los más utilizados?
Los métodos HTTP permiten comunicar al servidor lo que se quiere realizar con un resource bajo una URL. Los métodos más importantes de HTTP son POST, GET, PUT, DELETE y HEAD.

Explica las diferencias entre los métodos HTTP GET y POST.
El get se utiliza para solicitar un url, mientras que el post se utiliza para actualizar un dato en el archivo.

¿Qué es un código de estado HTTP? ¿Podrías mencionar algunos de los más comunes y lo que significan?
Un código de estado nos dice si la petición se ha resuelto bien o no. Algunos códigos son el 404 not found, el 200 respuestas satisfactorias

¿Qué es una cabecera HTTP? Da ejemplos de cabeceras comunes.
Se trata de un número de tres dígitos que se incluye en el encabezado HTTP de una página web.
Ejemplos:

- **100–199**: Respuestas informativas
- **200–299**: Respuestas satisfactorias
- **300–399**: Redirecciones
- **400–499**: Errores de los clientes
- **500–599**: Errores de los servidore

¿En qué consiste el concepto de "idempotencia" en los métodos HTTP? ¿Qué métodos cumplen con esta característica?
La idempotencia es una propiedad de las operaciones o solicitudes de API que garantiza que repetir la operación varias veces produzca el mismo resultado que ejecutarla una vez. Los métodos que cumplen son GET ,PUT y DELETE . mientras que los POST Y PATH no lo son.

¿Qué es un redirect (redirección) HTTP y cuándo es utilizado?
Un redirect o redirección HTTP es una instrucción que se envía a los navegadores y rastreadores para informarles que una URL se ha movido a otra. Se utiliza para redirigir a los usuarios y a los motores de búsqueda a un nuevo contenido cuando el original ya no está disponible.

Preguntas técnicas y de seguridad en HTTP/HTTPS:

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

asegura la integridad de los datos en una conexión a través de encriptación y verificación con lso protocolos mencionados anteriormente SSL/TLS

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

"man-in-the-middle" este tipo de ataque ocurre cuando un atacante intercepta y posiblemente altera la comunicación entre dos partes (por ejemplo, un usuario y un servidor), sin que estas se den cuenta de la intervención. Mientras que "replay" es una técnica donde el atacante captura datos legítimos en tránsito (como una solicitud HTTPS

autenticada) y luego los retransmite más tarde para intentar repetir alguna acción en el nombre del usuario original.

Explica el concepto de "handshake" en HTTPS.
Es el proceso incial que establece una conexión segura entre un cliente y un servidor. Este proceso es fundamental para garantizar que la comunicación posterior este cifrada y autenticada, protegiendo los datos que se intercambian entre ambas partes.
El hansshake busca garantizar 3 aspectos claves:
Autenticación: confirma la identidad del servidor y, en algunos casos, también del cliente.
Confidencialidad: establece una clave de sesión única que cifra la comunicación
Integridad: asegura que los datos no puedan ser alterados o interceptados sin que las partes se den cuenta.

¿Qué es HSTS (HTTP Strict Transport Security) y cómo mejora la seguridad de una aplicación web?
es un mecanismo de seguridad que permite a un servidor web indicar a los navegadores que solo deben interactuar con él a través de conexiones seguras HTTPS, nunca mediante HTTP. Este protocolo está diseñado para mitigar ataques de "downgrade" (donde un atacante fuerza la conexión a utilizar HTTP) y para evitar ataques de tipo "man-in-the-middle" (MITM) cuando un sitio web es accesible tanto por HTTP como por HTTPS.

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

Es una técnica en la cual un atacante intenta debilitar la seguridad de una conexión entre un cliente (como un navegador) y un servidor, forzando a que esta utilice un protocolo o cifrado menos seguro. El objetivo es que el atacante pueda aprovechar vulnerabilidades conocidas en protocolos o cifrados obsoletos para interceptar o modificar los datos en tránsito. HTTPS utiliza el protocolo TLS (Transport Layer Security) para establecer conexiones seguras .

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

es un mecanismo de seguridad en aplicaciones web que permite o restringe el acceso a recursos entre diferentes orígenes. En el contexto de la web, un "origen" se define como la combinación de protocolo, dominio y puerto de una URL. Por defecto, los navegadores bloquean solicitudes HTTP que intentan acceder a recursos de un origen diferente (cross-origin) para proteger al usuario de ataques como el Cross-Site Scripting (XSS) o el Cross-Site Request Forgery (CSRF).

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

La cabecera Authorization se utiliza para enviar tokens de autenticación con cada solicitud HTTp al servidor, con el objetivo de autenticar al cliente. una cabecera Cookie se utiliza para almacenar y enviar datos de sesión o preferencias del usuario al servidor con cada solicitud.

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

Las cabeceras de seguridad como Content-Security-Policy (CSP) y X-Frame-Options son configuraciones que ayudan a proteger aplicaciones web contra ataques comunes, como el Cross-Site Scripting (XSS) y el Clickjacking, entre otros. Estas cabeceras permiten a los desarrolladores definir políticas de seguridad que los navegadores deben respetar, duciendo así la superficie de ataque de las aplicaciones web.

La cabecera Content-Security-Policy es especialmente útil para prevenir ataques de Cross-Site Scripting (XSS) e inyecciones de contenido. Estos ataques ocurren cuando un atacante logra insertar y ejecutar código malicioso (como scripts) dentro de una página web confiable, lo que le permite robar datos de usuarios, secuestrar sesiones o realizar otras actividades maliciosas.

Cómo mitiga ataques:Restringiendo orígenes de scripts y contenido: CSP permite especificar de dónde se pueden cargar scripts, imágenes, estilos y otros recursos. Por ejemplo, al restringir la carga de scripts solo a fuentes confiables, el navegador bloquea la ejecución de cualquier script inyectado por un atacante.

La cabecera X-Frame-Options protege contra ataques de Clickjacking . En un ataque de Clickjacking, un atacante incrusta el sitio web objetivo en un <iframe>invisible sobre su propio sitio, engañando a los usuarios para que interactúen con el sitio incrustado sin saberlo (por ejemplo, haciendo clic en botones sin su consentimiento).

Cómo mitigar los ataques :Bloqueando la incrustación de la página : Con X-Frame-Options, el desarrollador puede evitar que su página sea cargada en un <iframe>en otros sitios web. Esto previene que el sitio sea embebido en sitios de terceros con intenciones maliciosas.

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

- **Velocidad**: HTTP/3 es más rápido que HTTP/1.1 y HTTP/2.
- **Seguridad**: HTTP/3 es más seguro que HTTP/1.1 y HTTP/2.
- **Soporte**: HTTP/3 tiene menos soporte que HTTP/1.1 y HTTP/2.
- **Protocolo de transporte**: HTTP/3 utiliza QUIC (Quick UDP Internet Connections) en lugar del TCP (Protocolo de Control de Transmisión) que utilizan HTTP/1 y HTTP/2.
- **Multiplexación**: HTTP/2 permite enviar múltiples solicitudes y respuestas simultáneamente, mientras que HTTP/1.1 carga los recursos uno después del otro.
- **Compresión**: HTTP/2 utiliza la compresión HPACK para reducir el tamaño de los encabezados, mientras que HTTP/1.1 envía encabezados de texto no compactados.

- **Codificación**: HTTP/2 utiliza codificación binaria en lugar de codificación de texto.

¿Qué es un "keep-alive" en HTTP y cómo mejora el rendimiento de las aplicaciones?
Un "keep-alive" en HTTP, también conocido como conexión persistente HTTP, es una instrucción que permite a un servidor web y un cliente mantener abierta una conexión TCP para enviar y recibir múltiples solicitudes y respuestas HTTP. El "keep-alive" mejora el rendimiento de las aplicaciones porque: Reduce la cantidad de solicitudes HTTP, Acelera la página web, hace más eficiente la transmisión de datos.

Preguntas de implementación práctica:

¿Cómo manejarías la autenticación en una API basada en HTTP/HTTPS? ¿Qué métodos conoces (Basic, OAuth, JWT, etc.)?
Lo manejaría con autenticación basada en token. Conozco el método JWT.

¿Qué es un proxy inverso (reverse proxy) y cómo se utiliza en entornos HTTP/HTTPS?
Un proxy inverso (o reverse proxy) es un servidor que se sitúa entre los usuarios finales y los servidores backend, redirigiendo las solicitudes de los usuarios a los servidores internos adecuados.
Solicitudes HTTP/HTTPS del Cliente: Los clientes (usuarios finales) envían solicitudes HTTP o HTTPS hacia el servidor proxy inverso en lugar de hacerlo directamente al servidor de destino
El Proxy Inverso:

- En el caso de HTTP: El proxy inverso toma la solicitud HTTP y la redirige al servidor backend correspondiente para su procesamiento.
- En el caso de HTTPS: Si el proxy inverso maneja terminación SSL/TLS, cifra y descifra la comunicación. Luego, puede reenviar las solicitudes a los servidores backend a través de HTTP (después de haber descifrado el tráfico HTTPS) o también puede cifrar la conexión con los servidores backend utilizando HTTPS.

Respuesta del Servidor Backend: Los servidores backend procesan las solicitudes, y el contenido resultante (como una página web) se devuelve al proxy inverso.
El Proxy Inverso Responde al Cliente

¿Cómo implementarías una redirección automática de HTTP a HTTPS en un servidor?
Para implementar una redirección automática de HTTP a HTTPS en un servidor web, se debe asegurar de que las solicitudes que llegan al servidor a través del protocolo no seguro HTTP sean automáticamente redirigidas al protocolo seguro HTTPS. Este proceso puede realizarse de diferentes maneras, dependiendo del servidor web que se este utilizando.
¿Cómo mitigarías un ataque de denegación de servicio (DDoS) en un servidor HTTP?
Algunas opciones son

1. Configurar Firewalls y filtros de IP.
2. Usar Captchas para bloquear bots.
3. Monitorear tráfico en tiempo real.
4. Redundancia de servidores y failover.
5. Aplicar TCP SYN Cookies y anti-spoofing.

¿Qué problemas podrías enfrentar al trabajar con APIs que dependen de HTTP, y cómo los resolverías?
Algunos de los problemas que podrían a llegar a surgir son que la API que estamos consultando este caido el servidor, por lo tanto habría que esperar. Otra podría ser que url este mal, ósea que lo tendríamos que corregir.

¿Qué es un cliente HTTP? ¿Mencionar la diferencia entre los clientes POSTMAN y CURL?
Un cliente HTTP es un software o herramienta que se utiliza para realizar solicitudes a un servidor a través del protocolo HTTP (o HTTPS en su versión segura). Estos clientes envían solicitudes de distintos tipos (como GET, POST, PUT, DELETE, entre otros) a una URL y reciben respuestas de los servidores, que pueden contener datos o información sobre el estado de la solicitud. La diferencia entre postman y curl son que:
Posman utiliza una interfaz grafica mientras que curl es una herramienta de línea de comandos entre otros aspectos.

Preguntas de GIT

¿Qué es GIT y para qué se utiliza en desarrollo de software?
Git es un sistema de control de versiones distribuidoosea que cada colaborador tiene una copia en su disco local. Sirve para clonar proyectos y que participen varios colaboradores y asi tener un historial de versiones.

¿Cuál es la diferencia entre un repositorio local y un repositorio remoto en GIT?
La diferencia esta en donde se almacenan, el repositorio local se aloja en el equipo del usuario mientras que el remoto se aloja en el servidor.

¿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.
Un repositorio se crea con git init o git clone si ya existe.
Un git commit es un registro de los cambios, representa algo especifico.
Un git push se utiliza para comunicar con otro repositorio

¿Qué es un "branch" en GIT y para qué se utilizan las ramas en el desarrollo de software?
Las ramas son ramificaciones, nuevos caminos que toma el proyecto. Rama principal es master donde esta proyecto al publico y la dev es la que vamos a trabajar. Sirven para no romper el proyecto.

¿Qué significa hacer un "merge" en GIT y cuáles son los posibles conflictos que pueden surgir durante un merge?
Un merge es fusionar tu rama con la rama principal del proyecto, los problemas que podrían haber son que se actualicen dos archivos iguales, entonces git no sabría que versión utilizar.

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

un Branching Model o modelo de ramificación es un enfoque estratégico para gestionar el control de versiones de un proyectoun ejemplo es el flujo de github.

¿Cómo se deshace un cambio en GIT después de hacer un commit pero antes de hacer push?
Con el comado git reset

¿Qué es un "pull request" y cómo contribuye a la revisión de código en un equipo?
solicitar que otro desarrollador incorpore (o haga un pull) una rama de tu repositorio al suyo.

¿Cómo puedes clonar un repositorio de GIT y cuál es la diferencia entre git clone y git pull?
Git clone clona el proyecto para un nuevo integrante por ejemplo,
Git pull extrae y descarga contenido desde un repositorio remoto y actualiza el repositorio local.

Preguntas de GIT

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

Es un entorno de ejecución para javascript, es popular porque facilita el desarrollo de funciones de backend o API (interfaz de programación de aplicaciones) sólidas y escalables.

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

permite que la aplicación continúe ejecutándose mientras espera los resultados de las operaciones de entrada/salida (I/O). Esto se logra gracias a los métodos asíncronos, que permiten ejecutar múltiples operaciones simultáneamente sin esperar a que una se complete

¿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 funciona en un ciclo constante, revisando la cola de tareas y ejecutando cada una de ellas en orden. Permite que tu programa siga corriendo sin la obligación de esperar a que un proceso termine su ejecución. la principal diferencia entre require() e import() es que require() es sincrónico y se puede llamar desde cualquier parte del programa, mientras que import() se ejecuta siempre al inicio del archivo

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

Npm es un gestor de paqutes. Su función es gestionar las dependencias, distribuir el código.

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

Npm init

El propósito de package.json es que nos proporciona información sobre nuestra aplicación.

¿Qué son las dependencias en npm y cómo se instalan? Explica la diferencia entre dependencias y dependencias de desarrollo.
Dependencias: Son necesarias para que tu proyecto funcione en producción y se instalan con npm install.
Dependencias de desarrollo: Son necesarias solo durante el desarrollo y se instalan con npm install --save-dev.
Las dependencias de desarrollo no se incluyen en el entorno de producción, lo que ayuda a reducir el tamaño del proyecto en producción

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

De forma exacta sin actualzaciones "express": "4.17.1" entre otras opciones.
El archivo package-lock.json es un archivo que npm crea automáticamente cuando se instala o actualiza cualquier paquete. Su propósito es bloquear las versiones exactas de todas las dependencias de un proyecto, incluyendo dependencias de las dependencias (subdependencias), para asegurar que el proyecto se instale con las mismas versiones en cualquier máquina o entorno.

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

NestJS es un framework para construir aplicaciones backend escalables y de alto rendimiento en Node.js, utilizando TypeScript como lenguaje principal. Utiliza conceptos avanzados como inyección de dependencias y decoradores, lo que facilita la creación de aplicaciones backend estructuradas, de fácil mantenimiento y prueba.

¿Cómo se manejan errores en Node.js y cuál es la diferencia entre callbacks, promesas y async/await para manejar código asincrónico?
El manejo de errores en Node.js se puede hacer de diferentes maneras, dependiendo del tipo de operación (sincrónica o asincrónica). En operaciones sincrónicas, se usa try-catch; en operaciones asincrónicas, puedes usar callbacks, promesas o async/await.

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:

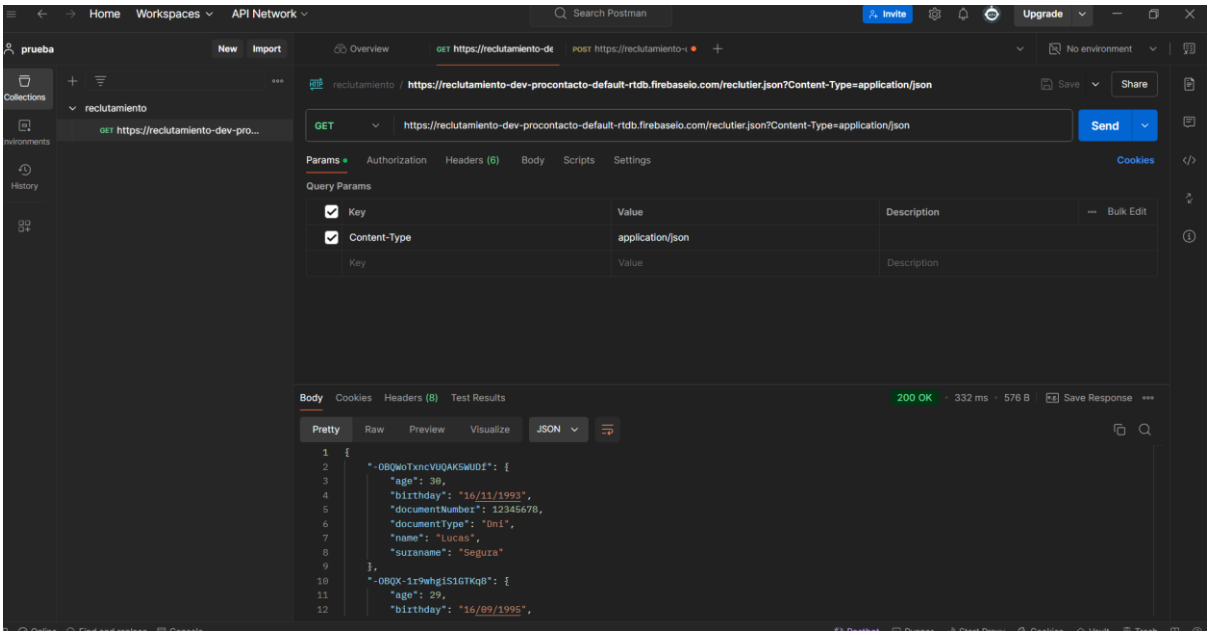
```
{
    "name":"TuNombre",
    "suraname":"TuApellido",
    "birthday":"1995/11/16/",
    "age":29,
    "documentType":"CUIT",
    "documentNumber":20123456781
}
```

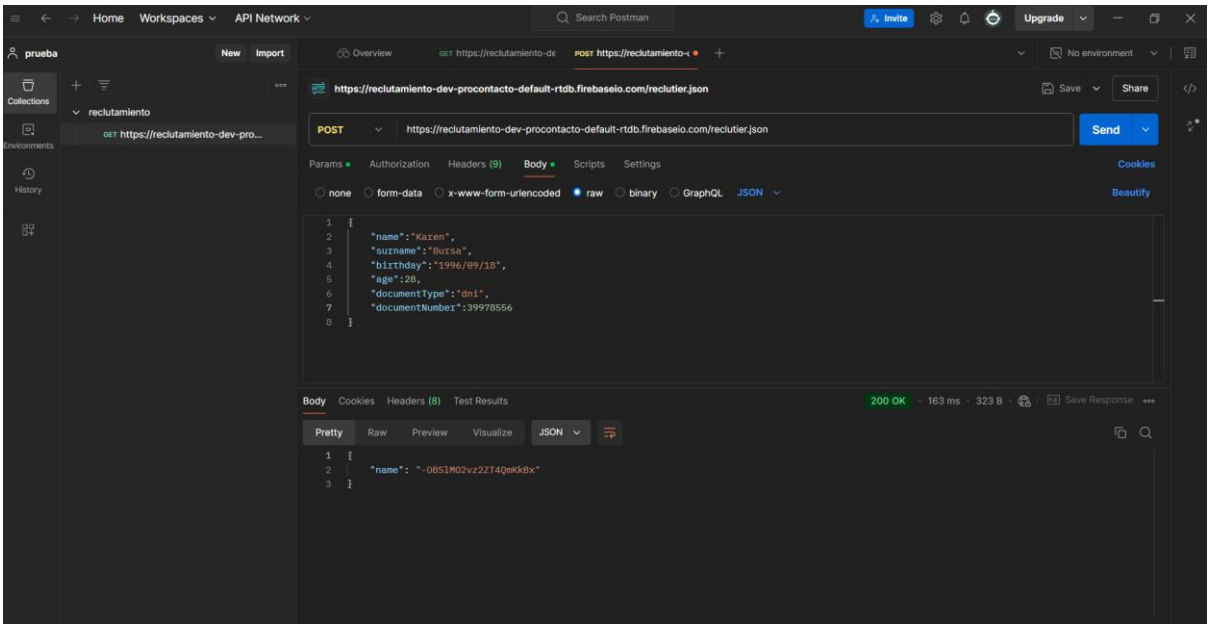
Reemplazar los campos por los valores personales tuyos.

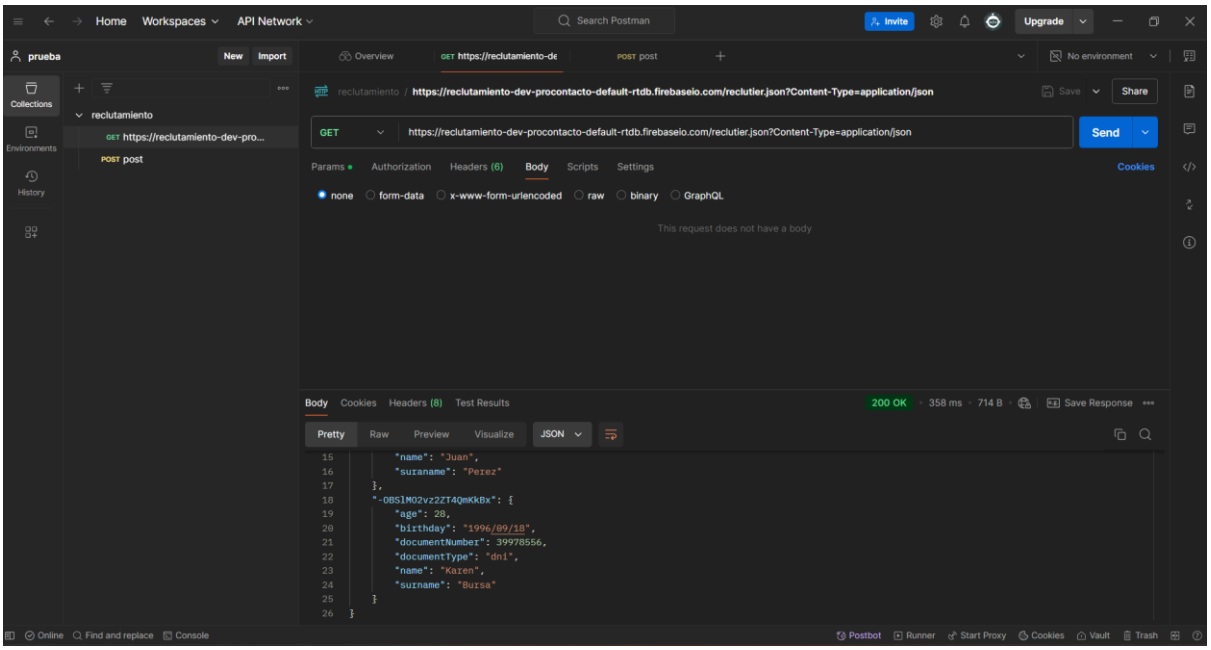
3) Volver a realizar el GET del punto número 1.

Preguntas/Ejercicios:

1. Adjuntar imagenes del response de un GET y de un POST de cada punto


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
prueba
New
Import
Collections
Environments
History
reclutamiento
GET https://reclutamiento-dev-pro...
Overview
GET https://reclutamiento-de
POST https://reclutamiento-
No environment
reclutamiento / https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json?Content-Type=application/json
Save
Share
GET
https://reclutamiento-dev-procontacto-default-rtdb.firebaseio.com/reclutier.json?Content-Type=application/json
Send
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Content-Type
application/json
Body
Cookies
Headers (8)
Test Results
200 OK
332 ms
576 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"-OBQWoTxncVUQAK5WUDf": {
"age": 30,
"birthday": "16/11/1993",
"documentNumber": 12345678,
"documentType": "Dni",
"name": "Lucas",
"suraname": "Segura"
},
"-OBQX-1r9whgiS1GTKq8": {
"age": 29,
"birthday": "16/09/1995",


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
prueba
New
Import
Collections
Environments
History
reclutamiento
GET https://reclutamiento-dev-pro...
Overview
GET https://reclutamiento-de
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 (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"Karen",
"surname":"Bursa",
"birthday":"1996/09/18",
"age":28,
"documentType":"dni",
"documentNumber":39978556
}
Body
Cookies
Headers (8)
Test Results
200 OK
163 ms
323 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"name": "-OBSlMO2vz2ZT4QmKkBx"
}



2. ¿Qué sucede cuando hacemos el GET por segunda vez, luego de haber ejecutado el POST?

   Se actualizan los datos en el body como lo indica en la tercera imagen. El post agrega un nuevo registro en la base de datos y el get actualiza permitendo visualizar el nuevo registro insertado.

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