**1. Considera la función de cada tecnología**

* + **Body-parser**: manejar solicitudes HTTP.
  + **CORS**: gestionar permisos entre dominios.
  + **Express**: manejar rutas y middleware.
  + **GraphQL**: realizar consultas de datos específicas.
  + **Lodash**: manipular arreglos y objetos.
  + **Mongoose**: modelar datos en MongoDB.
  + **Yarn**: gestionar dependencias.

----------------------------------------------------------------------------------------------------------------------

**1.Express**

* **Palabras clave: API, RESTful, servidor web, rutas, HTTP.**
* **Propósito:** Crear servidores web y APIs de manera sencilla.
* **Usado para:** Manejar rutas, solicitudes HTTP, y middlewares en aplicaciones Node.js.
* **Ejemplo:** Si te preguntan sobre un **framework para crear servidores web y APIs** y ves opciones como **CORS** o **Axios**, elimínalas, ya que la correcta es **Express**.

**2. CORS (Cross-Origin Resource Sharing)**

* **Palabras clave: permisos, diferentes dominios, solicitudes cruzadas, Cross-Origin.**
* **Propósito:** Manejar políticas de seguridad para compartir recursos entre diferentes dominios.
* **Usado para:** Permitir o restringir solicitudes entre el frontend y el backend cuando están en diferentes dominios.
* **Ejemplo:** Si te preguntan sobre **permitir solicitudes HTTP entre diferentes dominios** y ves opciones como **JWT** o **Mongoose**, elimínalas, ya que la respuesta correcta es **CORS**.

**3. ApolloServer**

* **Palabras clave: consultas, escalabilidad, GraphQL, petición específica de datos.**
* **Propósito:** Facilitar consultas y orquestación en GraphQL.
* **Usado para:** Definir esquemas y resolvers que permiten a los clientes consultar y manipular datos de manera eficiente en GraphQL.
* **Ejemplo:** Si te preguntan sobre un **servidor para gestionar consultas en GraphQL** y ves opciones como **Mongoose** o **REST**, elimínalas, ya que la correcta es **ApolloServer**.

**4. Mongoose**

* **Palabras clave: ODM, MongoDB, modelo, esquema, consultas, validación.**
* **Propósito:** Modelar y operar datos en MongoDB usando un ODM (Object Data Modeling).
* **Usado para:** Crear esquemas y realizar consultas, validación, y control de datos en MongoDB.
* **Ejemplo:** Si te preguntan sobre una biblioteca para **modelar datos y realizar operaciones en MongoDB** y ves opciones como **Lodash** o **Nodemon**, elimínalas, ya que la correcta es **Mongoose**.

**5. Body-parser**

* **Palabras clave: analizar, cuerpo, solicitudes POST, datos JSON.**
* **Propósito:** Analizar el cuerpo de las solicitudes HTTP entrantes (especialmente en métodos POST).
* **Usado para:** Extraer datos enviados en JSON o formularios en las solicitudes HTTP.
* **Ejemplo:** Si te preguntan sobre un middleware que **analiza el cuerpo de las solicitudes HTTP entrantes** y ves opciones como **CORS** o **Express**, elimínalas, ya que la correcta es **Body-parser**.

**6. GraphQL**

* **Palabras clave: lenguaje de consulta, API, consultas específicas, resolver, eficiencia de datos.**
* **Propósito:** Proporcionar un lenguaje de consulta para las APIs, permitiendo pedir solo los datos necesarios.
* **Usado para:** Definir qué datos necesita el frontend, optimizando la consulta y reduciendo el tráfico innecesario entre cliente y servidor.
* **Ejemplo:** Si te preguntan sobre un **lenguaje de consulta que permite solicitar solo los datos necesarios** y ves opciones como **REST** o **npm**, elimínalas, porque la respuesta correcta es **GraphQL**.

**7. Yarn**

* **Palabras clave: dependencias, instalar paquetes, bibliotecas, gestión de versiones.**
* **Propósito:** Administrar dependencias y paquetes en aplicaciones de JavaScript.
* **Usado para:** Instalar y gestionar bibliotecas de terceros, garantizando versiones estables y consistentes.
* **Ejemplo:** Si te preguntan sobre **instalar y gestionar dependencias** y ves opciones como **GraphQL** o **Mongoose**, elimínalas, ya que **Yarn** o **npm** son las respuestas correctas.

**8. Lodash**

* **Palabras clave: arreglos, objetos, mapear, filtrar, manipulación de datos.**
* **Propósito:** Proporcionar utilidades para manipulación de datos, especialmente arreglos y objetos.
* **Usado para:** Realizar operaciones complejas como filtrado, mapeo, o mezcla de objetos y arreglos.
* **Ejemplo:** Si te preguntan sobre **manipulación de arreglos y objetos en JavaScript** y ves opciones como **Nodemon** o **Body-parser**, elimínalas, porque la correcta es **Lodash**.

**9. JWT (JSON Web Token)**

* **Palabras clave: autenticación, token, sesiones, JSON Web Token (JWT), firmado.**
* **Propósito:** Autenticar y autorizar usuarios en aplicaciones web.
* **Usado para:** Enviar información de usuario firmada entre cliente y servidor, generalmente usada para sesiones y autenticación.
* **Ejemplo:** Si te preguntan sobre **autenticación y manejo de sesiones mediante tokens** y ves opciones como **CORS** o **GraphQL**, elimínalas, ya que la correcta es **JWT**.

**10. REST (Representational State Transfer)**

* **Palabras clave: arquitectura, APIs, HTTP, GET, POST, PUT, DELETE, estado.**
* **Propósito:** Estandarizar la manera en que los sistemas se comunican a través de la web.
* **Usado para:** Crear APIs que utilizan operaciones HTTP estándar como GET, POST, PUT, y DELETE.
* **Ejemplo:** Si te preguntan sobre la **arquitectura de APIs que utiliza GET, POST, PUT, DELETE**, y ves opciones como **GraphQL** o **ApolloServer**, elimínalas, ya que **REST** es la respuesta correcta.

**11. Axios**

* **Palabras clave: solicitudes HTTP, peticiones, promesas, API, cliente HTTP**
* **Propósito:** Realizar solicitudes HTTP desde el frontend o backend.
* **Usado para:** Enviar solicitudes a servidores y recibir respuestas, manejando peticiones AJAX o REST.
* **Ejemplo:** Si te preguntan sobre **realizar solicitudes HTTP desde el frontend o backend** y ves opciones como **Mongoose** o **Nodemon**, elimínalas, ya que la correcta es **Axios**.

**12. Nodemon**

* **Palabras clave: reinicio automático, monitor de cambios, servidor, desarrollo, Node.js.**
* **Propósito:** Reiniciar automáticamente el servidor de Node.js cuando los archivos se modifican.
* **Usado para:** Mejorar el flujo de desarrollo al evitar reinicios manuales en cada cambio de código.
* **Ejemplo:** Si te preguntan sobre una herramienta que **reinicia automáticamente un servidor Node.js al detectar cambios en el código**, y ves opciones como **Axios** o **REST**, elimínalas, porque **Nodemon** es la respuesta correcta.

**13. Socket.io**

* **Palabras clave: tiempo real, eventos, comunicación bidireccional, mensajes instantáneos.**
* **Propósito:** Facilitar la comunicación en tiempo real entre el cliente y el servidor.
* **Usado para:** Implementar características como chats en vivo, actualizaciones en tiempo real y notificaciones instantáneas.
* **Ejemplo:** Si te preguntan sobre **comunicación en tiempo real entre el servidor y el cliente** y ves opciones como **REST** o **Express**, elimínalas, ya que **Socket.io** es la respuesta correcta.

**14. Middleware**

* **Palabras clave: interceptar, procesar solicitudes, intermediario, función entre el cliente y el servidor.**
* **Propósito:** Interceptar y modificar solicitudes o respuestas entre el cliente y el servidor en una aplicación web.
* **Usado para:** Autenticación, validación de datos, manejo de errores, entre otras funciones.
* **Ejemplo:** Si te preguntan sobre el **procesamiento de solicitudes HTTP antes de llegar al servidor** y ves opciones como **npm** o **MongoDB**, elimínalas, porque el término correcto es **Middleware**.

**15. PM2**

* **Palabras clave: producción, procesos, administración, monitoreo, reinicio automático.**
* **Propósito:** Administrar procesos de Node.js en producción.
* **Usado para:** Mantener aplicaciones Node.js en ejecución continua, reiniciarlas automáticamente y monitorear su desempeño.
* **Ejemplo:** Si te preguntan sobre una herramienta que **gestiona procesos de Node.js en producción** y ves opciones como **Body-parser** o **Axios**, elimínalas, porque la respuesta correcta es **PM2**.

**16. MongoDB**

* **Palabras clave: base de datos, NoSQL, documentos, JSON, escabilidad.**
* **Propósito:** Base de datos NoSQL orientada a documentos.
* **Usado para:** Almacenar datos en formato JSON-like, permitiendo flexibilidad en la estructura de los datos.
* **Ejemplo:** Si te preguntan sobre una **base de datos NoSQL que almacena documentos en formato JSON** y ves opciones como **Axios** o **Nodemon**, elimínalas, ya que la correcta es **MongoDB**.

**17. npm (Node Package Manager)**

* **Palabras clave: gestión de paquetes, instalar, librerías, dependencias, JavaScript.**
* **Propósito:** Administrar y compartir paquetes y dependencias de JavaScript.
* **Usado para:** Instalar, actualizar y mantener paquetes y bibliotecas en proyectos de Node.js.
* **Ejemplo:** Si te preguntan sobre **instalación de dependencias** y ves opciones como **GraphQL** o **Express**, elimínalas, porque **npm** o **Yarn** son las correctas.

**Body-parser** es para ver desde el sanbox la data

**cors** para ver el servidor sin tener scl

**express** es el servidor web

**Graphql** es el mediador entre mongodb y express

**Lodash** sirve para unir arreglos

**Mongose** para operar con mongoDB

**Yarn** sirve para que instale las dependencias en su mejor version y este todo preparado

**el @** sirve para instalar una version en particular usando yarn

**la !** es utilizada para un parametro que sea necesario estilo ID!, es decir que lo hace obligatorio

operaciones que realiza el template string que son las query y los mutation

Query llama y retorna nada mas

La mutation es como un commit para realizar transacciones de datos por lo que puede borrar y actualizar datos

const app = express();

app.use(cors());

app.listen(8090, function(){

console.log("Servidor Iniciado");

})

esto es para sincronizar express y apollo

2^cantidad de referencias + 1  
  
INPUT  
necesita todos los parámetros para una llamada

POPULATE

El método populate en Mongoose (una biblioteca de MongoDB para Node.js) se utiliza para realizar consultas de referencia entre documentos de diferentes colecciones. Esto permite cargar y mostrar datos relacionados de otras colecciones al realizar una consulta.

Por ejemplo, si tienes dos colecciones en MongoDB, Users y Posts, donde cada publicación (post) tiene un campo que hace referencia a un usuario (user), puedes usar populate para obtener no solo los datos del post, sino también la información del usuario asociado.























