Pregunta 1
GraphQL

GraphQL es una especificación para comunicaciones remotas entre cliente y servidor. Es un lenguaje de consulta y manipulación de datos para APIs, y un entorno de ejecución para realizar consultas con datos existentes. Este lenguaje no va a remplaza a REST, pero si puede ser una buena y novedosa alternativa para realizar consultas y manejar APIS.
Es factible emplear tanto GraphQL como REST en una misma aplicación, e incluso se puede ocultar y consolidar las APIs REST dentro de un servidor GraphQL. Esto se logra al encapsular el punto final REST dentro de un punto final GraphQL a través del uso de resolvers raíz.
Existen muchas cuestiones sobre GraphQL y SQL , aun que estas son totalmente distintas e independientes.
La diferencia principal con SQL, es que GraphQL es independiente de las fuentes de datos y utiliza resolvers para acceder y manipular los datos.
Estos resolvers se delegan en una capa de lógica de negocio que se comunica con diversas fuentes de datos, como API, bases de datos y caché.
GraphQL no solo esta ligado a proyectos que se realicen JavaScript como es un lenguaje para backend, el backend que se crea con GraphQL puede ser implementado en cualquier proyecto que se esté realizando.
Algo muy llamativo de este lenguaje es que ofrece muchos beneficios como ser

- transferencia eficiente de datos,
- consultas flexibles,
- productividad del desarrollador mejorada,
- adaptabilidad a cambios y un
- ecosistema de herramientas sólido

Lo que convierte a GraphQL como una opción poderosa para el desarrollo de APIS.

Clientes en GraphQL

Un cliente de GraphQL es una biblioteca o herramienta que permite interactuar con una API basada en GraphQL desde el lado del cliente. Su propósito principal es facilitar la realización de consultas, mutaciones y suscripciones a la API GraphQL, así como el manejo de la respuesta recibida.
Ejemplo de clientes en GraphQL

1. Apollo Client
2. Relay
3. GraphQL.js

4. Strawberry

Escalabilidad

GraphQL está diseñado para ser escalable y se utiliza en producción en muchas empresas bajo cargas de trabajo intensas.
GraphQL ofrece mejoras de rendimiento integradas, pero es responsabilidad del desarrollador escalarlo adecuadamente en todas las instancias y monitorear el rendimiento en producción.
Además de que existen clientes de GraphQL que permiten trabajar en modo desconectado utilizando funciones específicas para operaciones de datos sin conexión, como almacenamiento en caché y trabajadores de servicios.

Microservicios

Integrar GraphQL en una arquitectura de microservicios puede resultar beneficioso, proporcionando flexibilidad y control sobre las consultas, permitiendo que el cliente solicite solo los datos necesarios y reduciendo la cantidad de solicitudes realizadas. Una práctica recomendada es utilizar un esquema de GraphQL como una puerta de enlace API en lugar de permitir que el cliente se comunique directamente con varios servicios GraphQL. De esta manera, se puede dividir el backend en microservicios, pero consolidar todos los datos necesarios desde una única API en el frontend. Esta estrategia Además, GraphQL ofrece herramientas y características que facilitan la gestión y el rendimiento de la puerta de enlace API en un entorno de microservicios.

Resumen

GraphQL es un lengauje para comunicaciones remotas entre cliente y servidor.
Se puede utilizar en conjunto con REST y ofrece beneficios como consultas flexibles, eficiencia en la transferencia de datos y un sólido ecosistema de herramientas.
GraphQL no reemplaza a SQL, pero proporciona una alternativa novedosa para consultar y manipular datos.
Los clientes de GraphQL, como Apollo Client y Relay, permiten interactuar con una API GraphQL desde el lado del cliente.
GraphQL es escalable y se puede integrar en arquitecturas de microservicios, actuando como una puerta de enlace API para consolidar y controlar las consultas a múltiples servicios GraphQL.

Pregunta 2

La principal similitud entre autorización y autenticación es que ambas se utilizan para la seguridad de un sistema, ya sea de manera local o remota (en la red). Estas dos definiciones si bien cumplen un mismo objetivo, la seguridad, en la práctica son totalmente diferentes.
En que se diferencian?
La diferencia principal es que la autenticación verifica la identidad del usuario, mientas que la autorización es el proceso por el cual se determinan ciertas acciones de los usuarios. Estas dos funciones no pueden existir sin una de la otra, ya que hacen un proceso y dan una capa de seguridad de manera integral.
Una función verifica (autentificación) y la otra restringe (autorización) las funciones y acciones que pueden hacer los usuarios autenticados previamente dentro del sistema en el que se está trabajando.

Flujo de todo este proceso
Autenticación -> usuario existe? -> contraseña correcta? -> autenticado -> autorización? -> tipo de autorización del usuario (root, vendedor, usuario común, administrador) -> autorización

Ejemplos de autenticación: incluyen contraseñas, biometría (huellas dactilares, reconocimiento facial), tarjetas inteligentes, etc.
Ejemplos de autorización: permisos de lectura, escritura o ejecución en un sistema o acceso a recursos específicos.

Pregunta 3

JSON Web Tokens in Detail

1. Introducción:

Los JWT tienen una estructura muy particular dividida en 3 partes:

- El encabezado (header)
- La carga útil (payload)
- Los datos de firma

Se utiliza una variante de codificación Base64 que es segura para URL, sustituyendo ciertos caracteres remplazándolos por otros para mayor seguridad y cifrado en el envío de los datos.
Acá un ejemplo:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
Se traduce como:

Header:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload:

```
{
  "sub": "1234567890",
  "name": "Heber Tito",
  "docente": "El mejor",
  "iat": 1516239022
}
```


2. Header

El header dentro de cualquier JWT son super importantes. En este apartado se especifican los datos más importantes dentro de esta estructura (aparte de la data que se encuentra almacenada en el JWT.)

Dentro del header hay información crucial para descifrar con que algoritmo fue cifrado el JSON (alg), siendo este el único campo obligatorio que debe tener cualquier JWT.
Dentro del header también se encuentra información typ, que quiere decir que tipo de estructura es, por defecto este campo tiende a ser redundante ya que siempre se coloca typ: jwt/jwe/jose
y cómo último campo que existe en el header es el cty, que especifica el tipo de contenido de la carga útil (payload) del JWT.
En la mayoría de los JWT, la carga útil consiste en afirmaciones específicas más datos arbitrarios.
Algunos ejemplos:

```
{
 "alg": "RS256",
 "typ": "JWT",
 "cty": "application/json"
}
```



```
{
 "alg": "RS256",
 "typ": "JWT",
 "cty": "JWT"
```

}
En algunos casos también se encuentra el apartado "enc" que hace la referencia a la encriptación del JWT, que es distinto del cifrado, es una capa más de seguridad.

3. The payload

En el payload es donde viene toda la información en sí, toda la data que se este transportando, que se quiera enviar dentro del formato JSON. En este apartado también se pueden especificar los famosos reclamos:

5. iss: Indica quien emitió el JWT
6. sub: Se utiliza para identificar de manera única al sujeto sobre el cual se hacen afirmaciones en el JWT.
7. aud: Identifica de manera única a los destinatarios previstos del JWT.
8. exp: Indica el tiempo de expiración del JWT
9. nbf: Especifica la fecha desde cuando está válido el JWT. (no antes de)
10. iat: Específica la fecha de cuanod fue emitido el JWT.
11. jti: Representa el ID del JWT

Algunos de estos reclamos pueden ser públicos y algunos privados, en su mayorïa los nbf, iat y jti son privados.

4. Seguridad

En los JWT la seguridad no es una norma, pueden haber JWT que no tengan cifrado, es decir la firma . Creado por desarrolladores que no tengan buenas prácticas.

Un JWT es un estándar abierto (RFC 7519) que proporciona un método compacto y seguro para transmitir información entre partes como un objeto JSON. Sin embargo, la seguridad de un JWT depende de cómo se implemente y se utilice en una aplicación.

5. Conclusión

Los JSON Web Tokens (JWT) son estructuras de datos que constan de un encabezado, una carga útil y una firma. Utilizan una codificación segura y se envían como cadenas de caracteres. El encabezado contiene información sobre el algoritmo de cifrado utilizado, el tipo de estructura y el tipo de contenido de la carga útil.

La carga útil es donde se agrega la información relevante del usuario y puede contener reclamos registrados, como el emisor, el sujeto, los destinatarios, la fecha de expiración, la fecha de emisión y el identificador del JWT. Algunos de estos reclamos pueden ser públicos, mientras que otros son privados y dependen de la implementación y la aplicación específica.

Es importante destacar que la seguridad de un JWT depende de cómo se implemente y se utilice.

Aunque los JWT pueden ser utilizados sin cifrado ni firma, esto puede comprometer la seguridad de la información. Es fundamental seguir buenas prácticas de seguridad al utilizar JWT y garantizar que se implementen medidas adecuadas de protección de datos.
En conclusión, los JSON Web Tokens son una forma eficiente y segura de transmitir información entre partes, pero su seguridad esta en función de la correcta implementación y uso responsable en las aplicaciones.
Es esencial comprender los diferentes componentes de un JWT y aplicar las medidas de seguridad adecuadas para proteger la integridad y confidencialidad de los datos transmitidos.

JSON Web Algorithms

1. Introducción

En este ensayo, exploraremos los algoritmos fundamentales utilizados en los JSON Web Tokens (JWT). Estos algoritmos desempeñan un papel crucial en la magia detrás de los JWTs, permitiendo una variedad de usos interesantes. Aunque comprenderlos a fondo no es necesario para utilizar los JWTs de manera efectiva, conocer estos algoritmos nos brinda una comprensión más profunda de cómo funcionan y cómo pueden aplicarse en diferentes contextos. A lo largo de este ensayo, descubriremos cómo estos algoritmos contribuyen a la seguridad y versatilidad de los JWTs

2. Algoritmos Generales

2.1 Base 64

Base64 es un algoritmo de codificación binario a texto que convierte una secuencia de octetos en caracteres imprimibles. Permite representar números en base 256 como caracteres en base 64. Aunque no es utilizado directamente por la especificación JWT, se utiliza una variante llamada Base64-URL. Este algoritmo es ampliamente utilizado y su estándar de facto es el RFC 4648.
El RFC 4648 es un estándar que define la codificación Base64 y Base32. Proporciona especificaciones detalladas sobre cómo representar datos binarios en forma de caracteres ASCII legibles y viceversa

2.2 SHA

El algoritmo de hash seguro (SHA) utilizado en JWT está definido en FIPS-180-2. Pertenece a la familia SHA-2, que incluye SHA-224, SHA-256, SHA-384 y SHA-512. En JWT, se utilizan principalmente SHA-256 y SHA-512.
Estos algoritmos procesan la entrada en fragmentos de tamaño fijo, aplicando operaciones matemáticas para producir un resumen. Son considerados seguros porque evitan colisiones y es computacionalmente difícil encontrar la entrada original a partir del resumen calculado.

3. Algoritmos de firma

3.1 HMAC

HMAC utiliza una función hash criptográfica y una clave para crear un código de autenticación para un mensaje específico. Esto garantiza la integridad del mensaje y proporciona autenticación utilizando una clave secreta.

3.2 RSA

RSA es un criptosistema ampliamente utilizado que se basa en claves asimétricas. Fue desarrollado en 1977 por Rivest, Shamir y Adleman. Utiliza la factorización de enteros como herramienta matemática principal. RSA se utiliza para el cifrado y la firma digital, donde se utiliza una clave pública para cifrar y una clave privada para descifrar o firmar. El algoritmo tiene diversas variaciones, y su seguridad se basa en la dificultad de factorizar grandes números enteros.

4. Conlusión

En conclusión, los algoritmos fundamentales utilizados en los JSON Web Tokens (JWT) desempeñan un papel esencial en la seguridad y funcionalidad de los JWTs. Aunque comprenderlos a fondo no es necesario para utilizar los JWTs de manera efectiva, tener un conocimiento sólido de estos algoritmos nos brinda una comprensión más profunda de su funcionamiento. Pero en estos momentos, con entender los conceptos es más que suficiente para sacar todo el provecho de estos algoritmos.