Principios REST

- Stateless (Sin estado): Las interacciones entre el cliente y el servidor en una API REST deben ser independientes y no depender del contexto almacenado en el servidor. Cada petición del cliente debe contener toda la información necesaria para que el servidor la procese y comprenda. Esto mejora la escalabilidad y la simplicidad del servidor al no tener que manejar y almacenar el estado del cliente entre peticiones.
- Client-Server (Cliente-Servidor): REST sigue el modelo cliente-servidor, donde el cliente es responsable de la interfaz de usuario y el servidor es responsable de procesar las peticiones y manejar la lógica empresarial. Esto permite la separación de responsabilidades y facilita la evolución independiente de ambas partes.
- Cacheable (Almacenable en caché): Las respuestas del servidor en una API REST pueden ser marcadas como cacheables o no cacheables. Si una respuesta es cacheable, el cliente puede almacenarla localmente y reutilizarla para peticiones futuras similares, lo que mejora la eficiencia, reduce la latencia y disminuye la carga en el servidor.
- Layered System (Sistema en capas): Una arquitectura REST puede organizarse en capas, con cada capa realizando una función específica. Esto permite la modularidad, la separación de responsabilidades y la mejora de la escalabilidad y la seguridad. Los componentes en una capa no necesitan conocer detalles sobre componentes en otras capas más allá de su interfaz de comunicación.
- Uniform Interface (Interfaz uniforme): REST enfatiza en la consistencia y simplicidad en la comunicación entre el cliente y el servidor a través de una interfaz uniforme. Esto incluye la identificación de recursos a través de URIs (Uniform Resource Identifiers), la manipulación de recursos a través de representaciones (por ejemplo, JSON o XML) y el uso de métodos estandarizados de HTTP (GET, POST, PUT, DELETE, PATCH) para realizar acciones en los recursos.
- Code on Demand (Código bajo demanda) - Opcional: Este principio establece que el servidor puede enviar código ejecutable al cliente, como scripts de JavaScript, para ampliar la funcionalidad del cliente según sea necesario. Aunque este principio es opcional en una arquitectura REST, puede ser útil en ciertas situaciones.