|  |  |
| --- | --- |
| **Web service dibutuhkan ketika** ada kebutuhan untuk komunikasi dan pertukaran data antara aplikasi yang berbeda atau komponen dalam suatu sistem yang terdistribusi  **Meliputi :**Interoperabilitas Antar Sistem,Pembagian Fungsi Layanan,Arsitektur Terdistribusi  **Arsitektur Microservice**  Peran API: Komunikasi Antar Layanan,Kebergantungan yang Rendah (Loose Coupling),Standarisasi dan Interoperabilitas,Keamanan    **Arsitektur Monolith**  Peran API: Modularisasi Kode,Ekspansi Fungsi,Refactoring dan Migrasi    Contoh Penggunaan  1. Injection Attacks  Serangan ini terjadi ketika penyerang memasukkan kode berbahaya ke dalam input yang diteruskan ke server.  2. Denial of Service (DoS) Attacks  Deskripsi:  Serangan DoS bertujuan untuk membuat layanan API tidak tersedia bagi pengguna yang sah dengan membanjiri server dengan permintaan yang sangat banyak atau kompleks.  3. Broken Authentication and Authorization  Deskripsi:  Serangan ini terjadi ketika mekanisme autentikasi dan otorisasi pada API tidak diterapkan dengan benar, memungkinkan penyerang untuk mengakses data atau fungsi yang seharusnya tidak mereka miliki.  **Database per Service**  Memastikan bahwa antar service tidak ketergantungan  Tiap service bisa menggunakan aplikasi database sesuai dengan kebutuhan  Service tidak perlu tahu kompleksitas internal database service lain  **Shared Database**  Ketika melakukan transisi dari aplikasi Monolith ke Microservices  Ketika bingung memecahkan data antar Service  Ketika dikejar waktu, sehingga tidak ada waktu untuk bikin API    **Stateless Microservices**  Biasanya tidak memiliki database,Digunakan untuk melakukan tugas sederhana,Biasa digunakan juga sebagai utility untuk microservice lain,Tidak bergantung dengan microservice lain  **Aggregation Microservices**  Tergantung dengan microservice lain,Biasa digunakan sebagai pusat business logic aplikasi,Boleh memiliki database ataupun tidak,Tidak bisa berdiri sendiri  **Keuntungan Service Orchestration**  Mudah dibuat, karena kode business logic akan terpusat di Aggregation Microservices  Mudah dimengerti, karena kode business logic akan terpusat di Aggregation Microservices  **Kekurangan Service Orchestration**  Aggregation Microservices terlalu ketergantungan dengan Microservices lain  Aggregation Microservices akan lebih lambat karena harus terkoneksi dengan Microservices lain  Aggregation Microservices akan lebih mudah error jika di Microservices lain terdapat masalah  Jika perlu Microservices baru, perlu dilakukan perubahan di Aggregation Microservices  **Keuntungan Service Choreography**  Aggregation Microservices tidak tergantung dengan Microservices lainnya  Aggregation Microservice akan lebih cepat, karena tidak perlu berkomunikasi dengan Microservices lainnya  Jika ada Microservice baru, Aggregation Microservice tidak perlu melakukan perubahan lagi  **Kekurangan Service Choreography**  Lebih sulit di-debug ketika terjadi masalah  Business logic akan terdistribusi di semua Microservices, sehingga sulit untuk dimengerti secara keseluruhan | **1.Consume API**  **Inisialisasi cURL**  Fungsi ini mengembalikan resource cURL yang akan digunakan dalam permintaan berikutnya.  Code : $ch = curl\_init();  **Mengatur opsi cURL**  mengatur opsi-opsi yang diperlukan untuk permintaan. Opsi-opsi ini diatur menggunakan fungsi curl\_setopt().  Code : $url = "https://api.example.com/data";  curl\_setopt($ch, CURLOPT\_URL, $url); // URL tujuan  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true); // Mengembalikan hasil sebagai string  curl\_setopt($ch, CURLOPT\_HTTPHEADER, array(  'Content-Type: application/json',  'Authorization: Bearer YOUR\_ACCESS\_TOKEN'  )); // Mengatur header  **Melakukan permintaan**  melakukan permintaan dengan fungsi curl\_exec(). Fungsi ini akan mengembalikan hasil dari permintaan API.  $response = curl\_exec($ch);  **Menutup Sesi cURL**  Langkah terakhir adalah menutup sesi cURL untuk membersihkan resource yang digunakan. Ini dilakukan dengan fungsi curl\_close().  Code : curl\_close($ch);  **NoSQL**  NoSQL bukanlah NO (TIDAK/BUKAN) SQL  NoSQL singkatan dari Not Only SQL  Kenapa??  Agar bisa disesuaikan dengan kebutuhan  Bisa mencari alternatif cara mengolah data  Mempercepat dalam proses penulisan atau pencarian  **Jenis-jenis :** Document Oriented Database,Key-Value Database,Column Families Database,Graph Database,Search Database,Time Series Database  Contoh : MongoDB : Document Oriented Database  Elasticsearch : Search Database  Redis : Key-Value Database  Apache Cassandra : Column Families Database  Neo4J : Graph Database  InfluxDB : Time Series Database  **Contoh Remote Procedure Invocation**  RESTful API (HTTP),gRPC,Apache Thrift ,SOAP,Java RMI,Corba  **Kemudahan menggunakan RPI**  Sederhana dan Mudah  Biasanya digunakan untuk komunikasi Request - Reply  Biasanya digunakan untuk proses Sync (yang butuh menunggu jawaban)  **Komunikasi dengan Cara Messaging**  Messaging biasanya digunakan untuk komunikasi Async  Async artinya komunikasi dilakukan tanpa harus menunggu selesai di proses  Dalam async, kadang tidak perlu peduli balasan dari service yang dituju  Biasanya komunikasi Messaging membutuhkan Message Channel sebagai jembatan untuk mengirim dan menerima data  Direkomendasikan menggunakan aplikasi Message Broker untuk melakukan management Message Channel  Contoh : Redis (PubSub),Apache Kafka,RabbitMQ,NSQ,Google PubSub,Amazon Web Service SQS  **Api gateway**  API Gateway adalah aplikasi yang bertugas sebagai gerbang dari luar ke dalam  Luar adalah akses dari internet, dan Dalam adalah aplikasi microservices  API Gateway bertugas sebagai proxy server ke semua aplikasi microservices  Aplikasi microservices hanya bisa diakses dari luar melalui API Gateway |