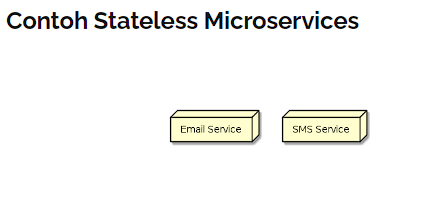
**Type of Microservices**

* Stateless Microservice
* Persistence Microservice
* Aggregation Microservices

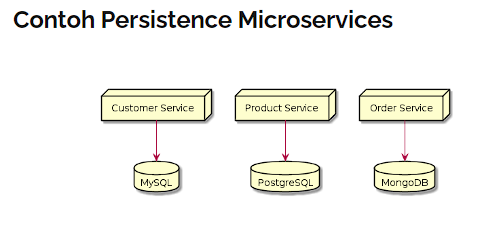
**Stateless Microservices**

* Biasanya tidak memiliki database
* Digunakan untuk melakukan tugas sederhana
* Biasa digunakan juga sebagai utility untuk microservice lain
* Tidak bergantung dengan microservice lain



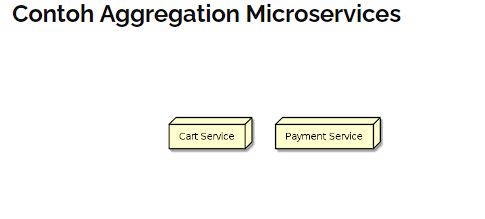
**Persistence Microservices**

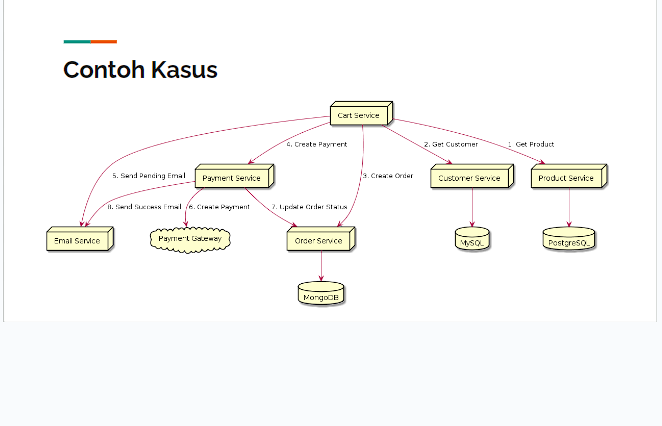
* Biasanya memiliki database
* Bisa juga disebut sebagai Master Data Microservice
* Biasa digunakan untuk mengolah data di database (CRUD)



**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

* Service orchestration memudahkan pembuatan aplikasi karena logika bisnis terpusat di Aggregation Microservices. Ini membuat kode menjadi lebih mudah dimengerti karena fokus pada satu tempat.
* Dengan menggunakan service orchestration, pengembang dapat dengan mudah mengelola dan memahami logika bisnis aplikasi mereka.

Keuntungan Menggunakan RPI

* RPI (Remote Procedure Invocation) adalah pendekatan sederhana dan mudah digunakan untuk komunikasi antara layanan, biasanya untuk tipe komunikasi Request-Reply dan proses yang bersifat synchronous.
* Penggunaan RPI mempermudah interaksi antar layanan dalam aplikasi yang memerlukan respons cepat dan sinkron.

Kelebihan Arsitektur Microservices

* Arsitektur microservices memiliki kelebihan dalam kemudahan pemahaman, pengembangan, skalabilitas, dan fleksibilitas teknologi. Layanan yang relatif kecil memudahkan pengembangan, pemeliharaan, pengujian, dan penyebaran aplikasi.
* Tim pengembang dapat bekerja lebih efisien, mengadopsi teknologi baru dengan mudah, dan menyesuaikan skala aplikasi sesuai kebutuhan dengan arsitektur microservices.

Masalah dalam Komunikasi RPI

* Implementasi komunikasi RPI dapat menghadapi masalah seperti proses yang memakan waktu, pengiriman data berulang, dan kompleksitas dalam membuat proses paralel.
* Pengembang perlu memperhatikan masalah-masalah ini untuk memastikan kinerja aplikasi tetap optimal.

Permasalahan dengan Banyak Jenis Frontend

* Berbagai masalah kompleksitas dalam pengembangan frontend aplikasi dapat muncul.
* Pengembang perlu mempertimbangkan faktor-faktor ini untuk menghasilkan aplikasi yang optimal dan sesuai dengan kebutuhan pengguna.

Keuntungan Service Choreography

* Service choreography memberikan keuntungan dalam interaksi yang lebih dinamis antara layanan, proses bisnis yang lebih fleksibel, dan kolaborasi yang lebih efektif.
* Dengan menggunakan service choreography, aplikasi dapat dirancang untuk beradaptasi dengan perubahan bisnis dan lingkungan dengan lebih baik.

Kenapa Harus Database per Service?

* Penggunaan database per layanan memastikan bahwa setiap layanan tidak saling tergantung satu sama lain, memungkinkan setiap layanan menggunakan database sesuai kebutuhan, dan mengurangi kompleksitas dalam interaksi antar layanan.
* Dengan pendekatan ini, pengembang dapat meningkatkan isolasi antar layanan, fleksibilitas, dan mengurangi risiko kesalahan dalam pengelolaan data.

Contoh Kasus

* Slide ini menyajikan contoh kasus tanpa penjelasan lebih lanjut, mungkin digunakan untuk mengilustrasikan penerapan konsep-konsep sebelumnya dalam situasi nyata.
* Contoh kasus membantu pengembang memahami bagaimana konsep-konsep yang dipelajari dapat diterapkan dalam konteks yang lebih konkret.

Pembagian Aplikasi Microservices

* Pembagian aplikasi menjadi layanan-layanan kecil (microservices) seperti Merchant, Product, dan Shipping.
* Dengan membagi aplikasi menjadi layanan-layanan kecil, pengembang dapat meningkatkan modularitas, skalabilitas, dan fleksibilitas aplikasi mereka.