BAB III. LANDASAN TEORI

3.1. Aplikasi Web

Aplikasi web adalah aplikasi yang dipanggil dengan peramban web melewati internet. Ada dua langkah dalam eksekusi aplikasi web, yaitu pemuatan atau *loading* dan komputasi yang disebabkan oleh suatu peristiwa. Selama pemuatan, peramban (*browser)* akan melakukan *parsing* dokumen HTML *(Hypertext Markup Language)* dan membuat pohon objek dokumen, yang kemudian ditampilkan di antarmuka pengguna berdasarkan CSS *(Cascading Style Sheet)* oleh mesin rendering [5]. Dalam satu dekade, web berevolusi dari tempat penyimpanan halaman yang utamanya digunakan untuk mengakses informasi statis yang mayoritas bersifat ilmiah, menjadi peron yang kuat untuk pengembangan aplikasi. Teknologi, bahasa dan metodologi baru membuka kemungkinan pengembangan aplikasi yang dinamis yang memungkinkan kolaborasi di antara banyak pengguna. Perkembangan aplikasi web kedepannya akan banyak dipengaruhi oleh perkembangan teknologi peramban, infrastruktur internet, protokol standar, metodologi rekayasa perangkat lunak, dan tren aplikasi [6].

3.2. REST API

API atau *Application Programming Interface* mengintegrasikan antara dua aplikasi yang berbeda dengan tujuan untuk berbagi data sebagai bentuk komunikasi. *Representational State Transfer* (REST) merupakan salah satu gaya arsitektur dalam mengembangkan API. REST API digunakan sebagai mekanisme integrasi aplikasi utama melalui Internet [7]. *Client* melakukan akses pada data di server yang dibedakan oleh id global atau *Universal Resource Identifiers* (URIs) dan data tersebut kemudian diproses dan dikembalikan oleh server dalam format berupa JSON ataupun XML.

3.3. *Microservices*

Munculnya ide pengembangan *micro frontend* berawal dari naiknya kepopuleran *microservices* sebagai solusi untuk kemacetan pada pengembangan *backend* dengan arsitektur monolitik. Arsitektur *microservice* memungkinkan pengembang untuk mengembangkan dan *deploy* aplikasi atau suatu proyek secara independen. Pada saat suatu layanan memerlukan perubahan fungsionalitas ataupun penambahan fitur, hanya layanan terkait saja yang dapat diubah dan di-*deploy* kembali tanpa harus melakukan perubahan dan *redeployment* pada seluruh aplikasi. Terlebih lagi, karena kesederhanaan arsitektur ini, *designers* dapat dengan mudah memahami dan melakukan perubahan, membuat member tim yang baru dapat lebih cepat menjadi produktif [8]. *Microservices* "adalah layanan kecil dan otonom yang bekerja sama'' [9].

3.4. Micro *Frontend*

Naiknya kepopuleran dari *microservices* membuat teknik *frontend* monolitik mengalami kemacetan untuk proyek aplikasi *full-stack,* dikarenakan perkembangan kerangka kerja monolitik tergolong lambat dibandingkan *microservices* atau kerangka kerja modular. Disisi ini *micro frontend* dapat mengatasi masalah tersebut karena mengikuti prinsip yang mirip dengan *microservices,* dan dapat dibangun diatas sebuah *microservices* atau *micro frontend* yang lainnya. *Micro-frontends* didefinisikan sebagai ″gaya arsitektur di mana aplikasi *frontend* yang dapat dikirimkan secara independen disusun menjadi satu kesatuan yang lebih besar″ [10].