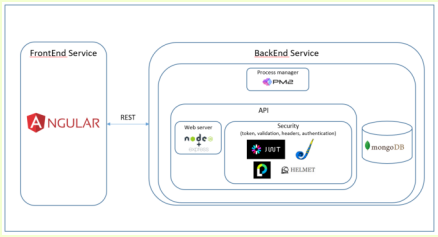



**Gambar 4**. Rancangan basis data menggunakan arsitektur *microservice*

*back end* pada sistem informasi DSS Pusat Sains Antariksa. Antara layanan *front end* dan *back end* dihubungkan oleh REST sebagai arsitektur yang digunakan dalam API, dengan data yang diberikan oleh server REST berupa format JSON, dan pengolahan basis datanya menggunakan MongoDB.

**Pustaka**


[1] Newman, S. (2015). Building Microservices. O'Reilly Media, Inc.

[2] microservices.io/i/Decomposing-Applications.011.jpg

[3] medium.com/kugen/software-architecture-monolithic-and-microservice-a9ed178ed954

[4] docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

[5] datacommcloud.co.id/monolithic-atau-microservices/


■

TEKNOLOGI INFORMASI

# Pengantar Kerangka Pemrograman Flutter untuk Pengembangan Aplikasi *Smartphone*

*Oleh*
**A.Z. Utama** | Pussainsa OR-PA BRIN

Dalam beberapa dekade terakhir banyak negara yang memiliki lembaga keantariksaan yang bertanggung jawab untuk melakukan kegiatan observasi, monitoring, analisis, pemodelan, dan prediksi cuaca antariksa serta memberikan mitigasi dini kepada masyarakat ataupun pemangku kepentingan yang terdampak langsung. Pusat Riset Antariksa LAPAN BRIN merupakan lembaga penelitian pemerintah yang bertugas memberikan pelayanan informasi cuaca antariksa di Indonesia. SWIFtS (*Space Weather Information and Forecast Services*) merupakan layanan informasi dan prediksi cuaca antariksa yang dimiliki oleh Pussainsa. Pussainsa memberikan informasi mengenai hasil prediksi kondisi cuaca antariksa dan aplikasinya pada web swifts.sains.lapan.go.id.

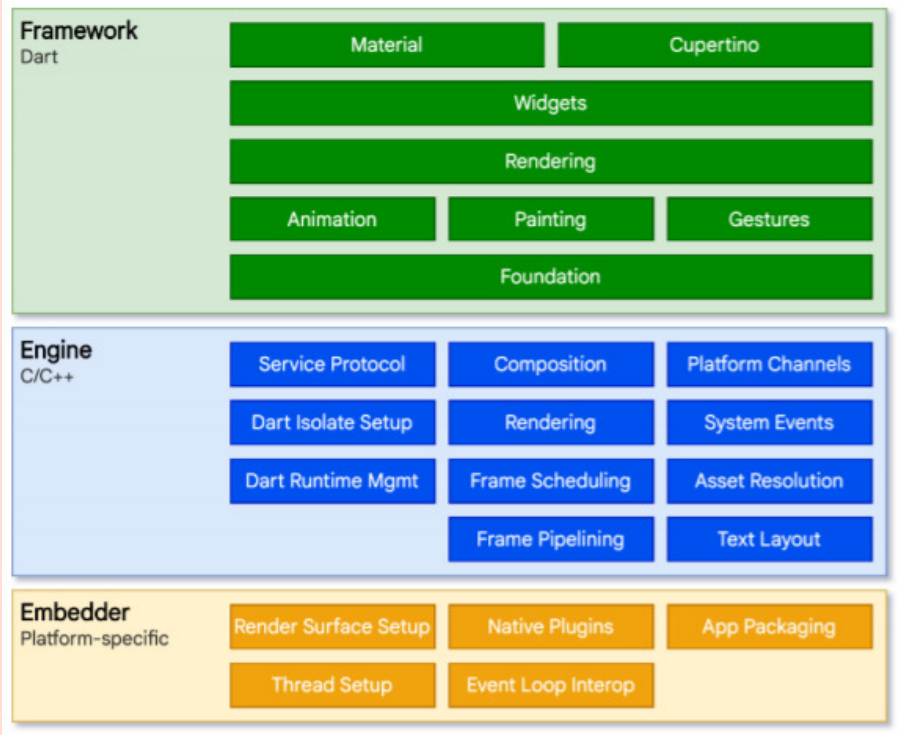
Efek dari cuaca antariksa dapat berakibat fatal pada kehidupan modern. Pada tahun 2012 terjadi peristiwa CME yang besar dan hampir mengenai planet Bumi. Para peneliti mengestimasi efek dari peristiwa CME tersebut apabila mengenai planet Bumi akan berakibat pada kegagalan sistem listrik dan akan berdampak pula pada sistem pengairan, akses internet, dan sistem perbankan. Radiasi yang dihasilkan oleh aktivitas Matahari aktif membutuhkan jam bahkan hari untuk sampai ke Bumi. Untuk meminimalisir dampak dari cuaca antariksa tersebut diperlukan sistem peringatan dini yang handal agar pemerintah dan perusahaan yang terkena dampak dapat segera mengambil keputusan untuk melakukan mitigasi secara tepat dan akurat agar infrastuktur dan teknologinya dapat terlindungi.

Pada masa kini kehidupan manusia tidak dapat dipisahkan dengan perangkat seluler pintar (*smartphone*). Pemanfaatan teknologi *smartphone* dapat membantu peneliti melakukan kegiatan desiminasi untuk masyarakat mengenai hasil penelitian ataupun informasi penting secara cepat. Aplikasi *smartphone* merupakan suatu wadah yang efektif untuk menyebarkan informasi tersebut, karena hampir sebagian masyarakat dunia sudah menggunakan perangkat tersebut. Aplikasi *mobile* dapat membantu dan mempermudah para *stakeholder* untuk mendapatkan informasi mengenai cuaca antariksa dan aplikasinya. Flutter dikeluarkan oleh perusahaan raksasa Google untuk membangun aplikasi dalam satu *codebase* yang dapat dikompilasi ke dalam perangkat *mobile*, web, dan *desktop*. Flutter merupakan kerangka pemrograman aplikasi *smartphone* yang bersifat *open-source*, yang artinya pengembang aplikasi *smartphone*

diizinkan untuk membuat, mengubah, mengembangkan, dan menyebarkan aplikasi yang dibangun di atas kerangka Flutter.

Flutter adalah alat (kit) perancangan perangkat lunak yang dikembangkan oleh Google. Flutter dapat digunakan untuk membuat aplikasi pada *platform* yang berbeda seperti Android, IOS, Linux, Mac, Windows, dan web dalam satu kode utama. Pengembang dapat membuat aplikasi yang memiliki performa tinggi dibeberapa *platform* secara sekaligus, sehingga dapat mempersingkat waktu pengembangan, mengurangi biaya, dan menghilangkan kompleksitas pengembangan produk pada *platform* yang berbeda. Tidak hanya itu, Flutter juga menyediakan komponen antar muka, dan animasi yang menarik bagi pengguna sehingga pengembang tidak akan mengalami kesulitan dalam membuat dan merancang antar muka. Flutter menggunakan konsep *Object-Oriented Programming* (*Class, Method, Variable*, dan lainnya) dan logika imperatif pemrograman (perulangan, kondisi, dan lainnya). Gambar 1 merupakan lapisan komponen utama yang membentuk sebuah Flutter *kit*.

Aplikasi Flutter dibangun di atas bahasa pemrograman Dart. Bahasa pemrograman yang dirancang khusus untuk rancangan antar muka dan logikanya secara optimal, sehingga dapat mempercepat pengembangan perangkat lunak khususnya perangkat *mobile*. Flutter beroperasi di lingkup *Dart Virtual Machine* pada sistem operasi Windows, MacOS, dan Linux. Dart *Platform* memungkinkan pengembang untuk melihat hasil kode secara




**Gambar 1**. Lapisan arsitektur pembentuk Flutter yang terdiri dari beberapa bagian fungsi yang independen. (Sumber: `https://flutter.dev/web`)

langsung tanpa harus menunggu waktu kompilasi atau disebut dengan istilah "*hot reload*". Flutter memanfaatkan fitur tersebut sehingga hanya membutuhkan satu kali waktu kompilasi.

Flutter *Engine* merupakan sebuah *portable-runtime* untuk menyediakan layanan untuk aplikasi Flutter yang dibangun menggunakan bahasa C++. Flutter *Engine* mengimplementasikan pustaka kode Flutter di antaranya grafik, animasi, *file* dan network I/O, fitur berbagi kode dan versi, arsitektur pemrograman, dan Dart *runtime*. Semua fitur tersebut dapat diakses melalui Flutter *Framework* yang didalamnya termasuk *Reactive Framework*, *service protocol*, Dart *isolate setup*, Dart *runtime management*, *rendering*, *system event*, *cross-platform*, dan fondasi *widgets*.

Fondasi pemrograman dibangun menggunakan Dart dan menyediakan berbagai *class* dan fungsi yang dapat digunakan oleh setiap pengembangan. Flutter menyediakan berbagai komponen yang telah dilengkapi dengan fungsionalitas atau dikenal dengan istilah *widgets*. *Widgets* inilah komponen pembentuk dari sebuah aplikasi. Setiap *widgets* mendefinisikan struktur dari sebuah elemen (contohya seperti tombol), atribut elemen, tata letak, dan lainnya. Flutter *Widgets* mengikuti pola rancangan *Material Design* yang memiliki kualitas yang baik dan UI/UX yang sudah teruji. Satu komponen dapat terbentuk dari beberapa *widgets* berdasarkan struktur pohon (model hirarki). Setiap *widget* yang berada dibawah *widget* lainnya, dapat menerima atribut atau konteks dari induknya tersebut.

■