BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dodo Kids Browser (Dodo) merupakan aplikasi *multi-platform* yang berfungsi sebagai *parental control*. Aplikasi ini membantu orangtua dalam mengontrol dan mengawasi aktivitas anaknya di internet. Dodo memiliki 3 fitur utama yaitu: Surfior, Notifior, Reportior [1]. Saat ini, Dodo berjalan dalam platform Windows Phone dan Ekstensi Browser. Dalam perkembangannya, Dodo akan dibangun dalam platform Desktop, Android, dan iOS. Untuk memenuhi kebutuhan tersebut, telah dibangun *Application Programming Interface* (API) menggunakan teknologi REST.

Sejauh ini, REST merupakan teknologi API yang paling populer karena memiliki banyak kelebihan yang beberapa diantaranya; terdapat beberapa pilihan tipe response; pengimplementasian yang dinilai relatif mudah; memiliki pemisahan yang jelas antara *client* dan *server* [2]. Namun demikian, REST memiliki kekurangan yang cukup serius yaitu *underfetching* dan *overfetching* [3]. *Underfetching* terjadi ketika *server* menerima satu request dari *client* namun tidak dapat sepenuhnya memberikan data yang diperlukan sehingga dibutuhkan *request* lain sebagai pelengkap. Hal ini dapat menyebabkan meningkatnya latensi sehingga pengguna perlu menunggu lebih lama sampai halaman berhasil dimuat, dan tingginya kompleksitas program karena dari sisi *client* perlu melakukan lebih dari satu *request* hanya untuk satu kebutuhan data.

Masalah *underfetching* pada REST ini dapat diatasi dengan menambahkan *endpoint* yang sesuai dengan kebutuhan *client*, namun hal ini justru akan menimbulkan masalah lain. Jika terdapat kode yang memiliki fungsi yang mirip ditulis lebih dari satu kali, seiring dengan berkembangnya aplikasi, kompleksitas kode akan semakin meningkat. Selain itu, proses *development* di sisi *client* juga akan melambat karena tertunda sampai API selesai diperbarui.

Kekurangan selanjutnya adalah *overfetching* yang merupakan keadaan di mana *server* memberikan *response* melebihi dari yang *client* butuhkan. Hal ini tentu tidak efisien karena pengguna mengunduh data yang tidak diperlukan, Ukuran *payload* yang besar akan menjadi masalah terutama pada platform mobile yang mengharapkan sistem yang responsif dan memiliki latensi rendah.

Pada tahun 2015, Facebook mempublikasikan teknologi API bernama GraphQL [3]. GraphQL merupakan bahasa *query* yang memungkinkan *client* menentukan secara spesifik data yang akan di-*request* ke *server*. Fleksibilitas inilah yang diharapkan dapat mengatasi masalah *underfetching* dan *over-fething* yang terdapat pada REST. Setelah dilakukan pengujian awal terhadap aplikasi yang sedang berjalan, pada penelitian ini akan dilakukan implementasi GraphQL untuk mengatasi *underfetching* dan *overfetching* pada aplikasi Dodo. Dengan menerapkan GraphQL, diharapkan aplikasi Dodo dapat memiliki performa yang baik.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, dapat dirumuskan suatu permasalahan yaitu apakah dengan mengimplementasikan GraphQL pada aplikasi Dodo dapat mengatasi masalah *underfetching* dan *overfetching*?

1.3 Maksud dan Tujuan

Penelitian ini bermaksud untuk mengimplementasikan GraphQL pada aplikasi Dodo, dengan tujuan mengatasi masalah *underfetching* dan *overfetching* demi meningkatkan performa aplikasi.

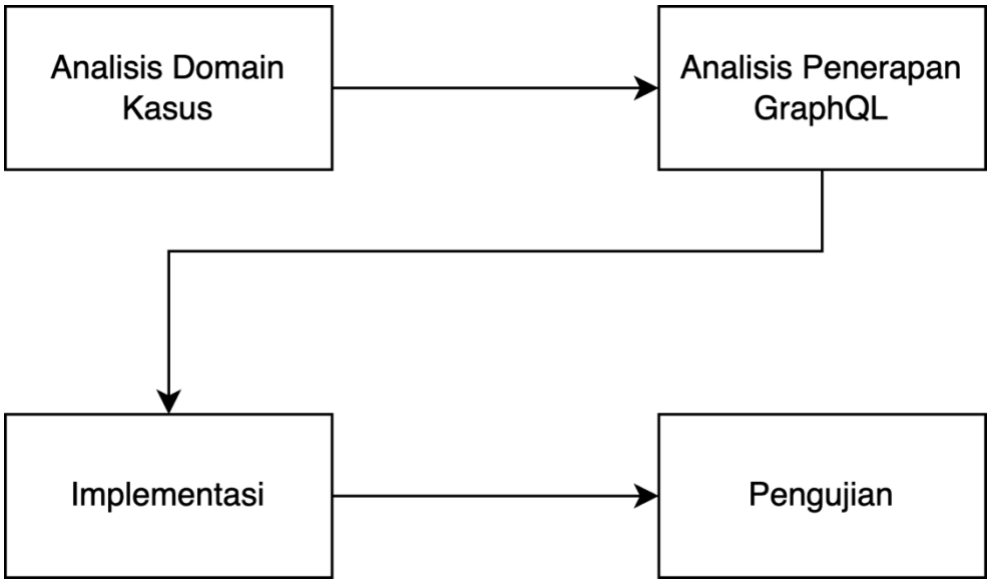
1.4 Batasan Masalah

Agar penelitian tidak menyimpang dari tujuan, telah ditentukan batasan masalah sebagai berikut:

1. Penelitian ini hanya berfokus pada sisi *server*.
2. Format data API yang digunakan adalah JSON.

1.5 Metodologi Penelitian

Metodologi penelitian komparatif akan digunakan pada penelitian ini, yang bertujuan membandingkan antara dua objek atau lebih dengan variabel tertentu[4]. Pada kasus ini, objek yang dibandingkan merupakan tingkat performa API pada aplikasi Dodo sebelum dan setelah diterapkannya GraphQL. Tahapan-tahapan dalam penelitian ini adalah sebagai berikut:



Gambar 1-1. Metodologi Penelitian

1. Analisis Domain Kasus

   Pada tahapan ini akan dianalisis fungsionalitas dari aplikasi beserta deskripsinya, kemudian dilakukan pemetaan terhadap API dari sistem yang berjalan. Selanjutnya dilakukan analisis underfetching dan overfetching terhadap API tersebut untuk kemudian diukur performansinya.
2. Analisis Penerapan GraphQL

   Tahapan ini akan dilakukan anlisis terhadap data pada sistem berjalan lalu dilakukan perancangan GraphQL berdasarkan data tersebut.
3. Implementasi GraphQL API

   Tahapan ini berisi penerapan GraphQL API ke dalam kode berdasarkan perancangan yang telah dilakukan.
4. Pengujian

   Pada tahapan ini akan dilakukan pengujian terhadap GraphQL yang telah diimplementasikan apakah dia dapat menyelesaikan masalah *underfetching* dan *overfetching*.

1.6 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini dibagi menjadi 5 bab yang tersusun secara sistematis untuk memberikan gambaran umum mengenai penelitian yang dikerjakan.

1.6.1 BAB 1 – Pendahuluan

Pada Bab 1 akan dijelaskan mengenai latar belakang masalah, perumusan masalah, maksud dan tujuan, batasan masalah, metodologi penilitan dan juga sistematika penulisan.

1.6.2 BAB 2 – Landasan Teori

Pada Bab 2 akan diuraikan mengenai teori – teori dasar yang akan digunakan pada penelitian. Selain itu juga akan ditinjau perihal serupa dari penelitian-penelitian sebelumnya sebagai acuan dalam pemecahan masalah.

1.6.3 BAB 3 – Analisis dan Perancangan

Pada bab 3 akan dianalisis masalah yang ditemukan dalam domain kasus dan berupaya merancang solusi untuk memecahkan masalah tersebut.

1.6.4 BAB 4 – Implementasi dan Pengujian

Pada bab 4 akan dijelaskan bagaimana implementasi yang dilakukan berdasarkan perancangan solusi yang telah dilakukan, kemudian akan dijelaskan bagiamana solusi tersebut diuji.

1.6.5 BAB 5 – Kesimpulan dan Saran

Pada bab 5 akan dijelaskan kesimpulan yang diperolah dari keseluruhan penelitian dan juga saran untuk penelitian ini ke depannya.