**IMPLEMENTASI GRAPHQL UNTUK MENGATASI UNDER-FETCHING PADA PENGEMBANGAN SISTEM INFORMASI PELACAKAN ALUMNI POLITEKNIK NEGERI MALANG**

**SKRIPSI**

Digunakan Sebagai Syarat Maju Ujian Diploma IV

Politeknik Negeri Malang

**Oleh:**

**FANY ERVANSYAH. NIM. 1641720080**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**JULI 2020**

HALAMAN PENGESAHAN

**IMPLEMENTASI GRAPHQL UNTUK MENGATASI UNDER-FETCHING PADA PENGEMBANGAN SISTEM INFORMASI PELACAKAN ALUMNI POLITEKNIK NEGERI MALANG**

Disusun oleh:

FANY ERVANSYAH. NIM. 1641720080

Laporan Akhir ini telah diuji pada tanggal .... ........ 2020

Disetujui oleh:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. | Penguji I | : | ................................................  NIP. | .......................... |
|  |  |  |  |  |
| 2. | Penguji II | : | ................................................  NIP. | .......................... |
|  |  |  |  |  |
| 3. | Pembimbing I | : | Putra Prima Arhandi, S.T., M.Kom.  NIP. 198611032014041001 | .......................... |
|  |  |  |  |  |
| 4. | Pembimbing II | : | Dhebys Suryani H, S.Kom., MT.  NIP. 198311092014042001 | .......................... |

Mengetahui,

|  |  |
| --- | --- |
| Ketua Jurusan  Teknologi Informasi | Ketua Program Studi  Teknik Informatika |
| Rudy Ariyanto, S.T., M.Cs. | Imam Fahrur Rozi, S.T., M.T. |
| NIP. 19711110 199903 1 002 | |  | | --- | | NIP. 19840610 200812 1 004 | |

**PERNYATAAN**

Dengan ini saya menyatakan bahwa pada Skripsi ini tidak terdapat karya, baik seluruh maupun sebagian, yang sudah pernah diajukan untuk memperoleh gelar akademik di Perguruan Tinggi manapun, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini serta disebutkan dalam daftar sitasi/pustaka.

|  |  |
| --- | --- |
|  | Malang, .... ......... 2020  Fany Ervansyah |

**ABSTRAK**

**Ervansyah, Fany**. “Implementasi Graphql Untuk Mengatasi Under-Fetching Pada Pengembangan Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang”. **Pembimbing: (1) Putra Prima Arhandi, S.T., M.Kom., (2) Dhebys Suryani H, S.Kom., MT.**

**Skripsi, Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang, 2020.**

Pada proses bisnis UMKM di Indonesia, umumnya masih belum menerapkan manajemen stok berdasarkan prediksi data penjualan. Artinya dari segi pencatatan masih bersifat Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi.

Oleh karena itu, perlu adanya Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi.

**Kata Kunci :** Sistem Informasi, GraphQL, under-fetching

***ABSTRACT***

***Ervansyah, Fany****. “Graphql Implementation To Solve Under-Fetching In Development Of State Polytechnic of Malang Alumni Searcher Information System”.* ***Counseling Lecturer: (1) Putra Prima Arhandi, S.T., M.Kom., (2) Dhebys Suryani H, S.Kom., MT.***

***Thesis, Informatics Management Study Program, Department of Information Technology, State Polytechnic of Malang, 2020.***

*As we might already know, the business process of small business in Indonesia, in general, have not implement selling data prediction based stock management. It means that the data recording process still having Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi.*

*Due to the given fact, it undoubtedly needed a kind of Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi. Lorem ipsum dolor sit amet. A quick brown fox jumps over a lazy frog. Kerjakan segera apa yang bisa dikerjakan hari ini, termasuk skripsi.*

***Keywords:*** *Information System, GraphQL, Under-fetching*

**KATA PENGANTAR**

Puji Syukur atas kehadirat Allah SWT/Tuhan YME karena atas segala rahmat dan hidayah-Nya, penulis dapat menyelesaikan skripsi dengan judul “IMPLEMENTASI GRAPHQL UNTUK MENGATASI UNDER-FETCHING PADA PENGEMBANGAN SISTEM INFORMASI PELACAKAN ALUMNI POLITEKNIK NEGERI MALANG”. Skripsi ini penulis susun sebagai persyaratan untuk menyelesaikan studi program Diploma IV Program Studi Teknik Informatika, Jurusan Teknologi Informasi, Politeknik Negeri Malang.

Penulis menyadari bahwasannya tanpa adanya dukungan dan kerja sama dari berbagai pihak, kegiatan laporan akhir ini tidak akan dapat berjalan baik. Untuk itu, penulis ingin menyampaikan rasa terima kasih kepada:

1. Bapak Rudy Ariyanto, ST., M.Cs., selaku ketua jurusan Teknologi Informasi.
2. Bapak Imam Fahrur Rozi, ST., MT., selaku ketua program studi Manajemen Informatika.
3. Putra Prima Arhandi, S.T., M.Kom., selaku pembimbing 1.
4. Dhebys Suryani H, S.Kom., MT., selaku pembimbing 2.
5. Dan seluruh pihak yang telah membantu lancarnya pembuatan Laporan Akhir dari awal hingga akhir yang tidak dapat kami sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan laporan akhir ini, masih banyak terdapat kekurangan dan kelemahan yang dimiliki penulis baik itu sistematika penulisan maupun penggunaan bahasa. Untuk itu penulis mengharapkan saran dan kritik dari berbagai pihak yang bersifat membangun demi penyempurnaan laporan ini. Semoga laporan ini berguna bagi pembaca secara umum dan penulis secara khusus. Akhir kata, penulis ucapkan banyak terima kasih.

Malang, .... ........... 2020

Penulis

**DAFTAR ISI**

Halaman

SAMPUL DEPAN i

HALAMAN JUDUL ii

HALAMAN PENGESAHAN iii

HALAMAN PERNYATAAN iv

ABSTRAK vi

*ABSTRACT* vii

KATA PENGANTAR viii

DAFTAR ISI ix

DAFTAR GAMBAR xi

DAFTAR TABEL xii

DAFTAR LAMPIRAN xiii

BAB I. PENDAHULUAN 1

* 1. Latar Belakang 1
  2. Rumusan Masalah 2
  3. Tujuan 3
  4. Batasan Masalah 3
  5. Sistematika Penulisan 4

BAB II. LANDASAN TEORI 5

* 1. 5
  2. 5

BAB III. METODOLOGI PENELITIAN 6

* 1. 8
  2. 10

BAB IV. ANALISIS DAN PERANCANGAN SISTEM 13

* 1. Analisis Kebutuhan Fungsional 20
  2. Analisis Data 23
  3. Perancangan Antarmuka Pengguna 24
  4. Perancangan Basisdata 25

BAB V. IMPLEMENTASI DAN PENGUJIAN 26

* 1. Implementasi Basisdata 28
  2. Implementasi Antarmuka Pengguna 30
  3. Implementasi Sistem 34
  4. Pengujian Fungsional 35
  5. Pengujian Akurasi 40

BAB VI. HASIL DAN PEMBAHASAN 42

* 1. 42
  2. 49

BAB VII. KESIMPULAN DAN SARAN 50

* 1. Kesimpulan 58
  2. Saran \* 58

DAFTAR PUSTAKA 63

LAMPIRAN-LAMPIRAN 35

**DAFTAR GAMBAR**

Halaman

Gambar 2.1 Statistik Institusi 16

Gambar 2.2 Struktur Organisasi 18

Gambar 3.1 Waterfall Model 19

Gambar 3.2 Desain Input 22

Gambar 3.3 Desain Output 28

Gambar 3.4 Antarmuka Utama 36

Gambar 3.5 Laporan Data Pegawai 46

**DAFTAR TABEL**

Halaman

Tabel 2.1 Data Penjualan Barang Selama Setahun 24

Tabel 2.2 Tugas dan Wewenang Tiap Unit Kerja 25

Tabel 3.3 Fungsi-fungsi/ Unit-unit Pada Delphi Yang Digunakan 35

Tabel 4.1 Peralatan Yang Diperlukan 49

**DAFTAR LAMPIRAN**

Lampiran 1 Menu Utama Aplikasi

Lampiran 2 Keluaran Sistem

Lampiran 3 Kuesioner Uji Pengguna

Lampiran 4 *Listing Program*

**BAB I. PENDAHULUAN**

* 1. **Latar Belakang**

Distribusi data merupakan hal yang sangat penting dalam pengembangan sebuah sistem informasi. Dalam pendistribusian data, beberapa website menerapkan metode yang berbeda-beda. Mulai dari menggabungkan antara bagian yang bertugas menampilkan informasi ke pengguna (Frontend) dan bagian yang mengatur bagaimana data - data diolah (Backend) hingga menyediakan layanan distribusi data seperti REST API, untuk sistem dengan struktur frontend dan backend yang terpisah.

Pada tahun 2017, metode yang paling banyak digunakan dalam pendistribusian data adalah dengan menggunakan metode REST API (Motroc, 2017). Namun, dalam metode REST API, terdapat suatu masalah yang disebut under-fetching, dimana bagian frontend perlu untuk melakukan permintaan data lebih dari 1 kali ke bagian backend (Porcello, Banks, 2018). Hal itu akan meningkatkan latensi, yang membuat pengakses website harus menunggu lebih lama sebelum data dikirim pada bagian frontend dengan sempurna. Selain itu, kompleksitas program juga semakin bertambah karena bagian frontend harus menambahkan 1 permintaan data lagi ke bagian backend.

Salah satu cara mengatasi under-fetching pada REST API adalah dengan membuat endpoint baru yang melakukan pengambilan data sesuai dengan apa yang diminta oleh frontend. Namun, jika terdapat banyak data yang mirip dan memiliki endpoint masing-masing, maka bentuk kode pada bagian backend akan menjadi kompleks dan kurang baik diakibatkan oleh adanya kode yang memiliki fungsi mirip namun ditulis lebih dari 1 kali.

Pada tahun 2015, secara publik, Facebook meluncurkan sebuah query language yang menjadi metode baru dalam mengatur pendistribusian data. Nama query language tersebut adalah GraphQL. Salah satu masalah yang dapat diatasi oleh GraphQL adalah masalah under-fetching.

Oleh karena itu, kali ini penulis mencoba mengimplementasikan GraphQL untuk mengatasi under-fetching pada Pengembangan Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang yang akan penulis kembangkan. Dengan adanya GraphQL, diharapkan dapat meningkatkan performa website Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang.

* 1. **Rumusan Masalah**

Berdasarkan latar belakang diatas, maka rumusan masalah yang dapat diambil adalah sebagai berikut :

Contoh untuk penyajian rincian ke bawah yang memiliki derajat urutan:

1. Bagaimana cara mengimplementasikan GraphQL pada pengembangan Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang?
2. Bagaimana GraphQL dapat mengatasi masalah under-fetching pada sistem informasi yang dikembangkan?
   1. **Tujuan**

Tujuan dari pengimplementasian GraphQL pada Sistem Informasi Pelacakan Alumni adalah:

1. Mengimplementasikan GraphQL pada Sistem Informasi Pelacakan Alumni Politeknik Negeri Malang.
2. Mengatasi masalah *under-fetching* pada sistem yang dikembangkan.
   1. **Batasan Masalah**

Agar skripsi penulis yang berjudul “Implementasi GraphQL Untuk Mengatasi Under-fetching pada Pengembangan Sistem Informasi Pelacakan Alumni” dapat berjalan sesuai rencana dan tujuan awal, maka penulis memberikan batasan - batasan masalah sebagai berikut:

1. Sistem Informasi Pelacakan Alumni menampilkan data alumni hasil Input Manual dan Scraping dari LinkedIn.
2. Implementasi GraphQL berfokus pada bagaimana mengatasi masalah Under-fetching pada pendistribusian data.
3. Teknologi yang digunakan adalah React.js untuk bagian frontend, Node.js untuk bagian backend, Selenium untuk scraping, dan Typescript sebagai bahasa pemrograman.
   1. **Sistematika Penulisan**

....

**BAB II. LANDASAN TEORI**

Sejauh ini, penulis menemukan 5 jurnal yang membahas mengenai graphQL. Jurnal - jurnal tersebut adalah:

1. “*An Initial Analysis of Facebook’s GraphQL Language*” bertujuan untuk memahami bahasa Graph milik GraphQL dan menunjukkan bahwa bahasa tersebut memiliki kompleksitas yang rendah (Hartig, Pérez, 2017).
2. “*API Design in Distributed Systems: A Comparison between GraphQL and REST*” bertujuan untuk membandingkan REST API dan GraphQL (Eizinger, 2017).
3. “*Improving the OEEU’s data-driven technological ecosystem’s interoperability with GraphQL*” yaitu menerapkan GraphQL pada Observatory of Employment and Employability, sebuah grup riset untuk lulusan universitas di Spanyol. Hasil dari penerapan GraphQL menunjukkan peningkatan performa, *flexibility*, dan *maintainability* (Vazquez, Cruz, García, 2017).
4. “*Performance of frameworks for declarative data fetching: An evaluation of Falcor and Relay+GraphQL*” membahas tentang perbandingan performa pengambilan data menggunakan Falcor, dan Relay+GraphQL. Falcor merupakan produk buatan Netflix yang juga dapat membantu pengambilan data dari *backend* ke *frontend*. Sedangkan Relay adalah sebuah *framework* untuk mempermudah pengambilan data dari sisi *client* (Cederlund, 2016).
5. “*Using GraphQL for Content Delivery in Kentico Cloud*” bertujuan untuk meriset GraphQL sebagai alternatif untuk mengantar konten pada Kentico Cloud, selain menggunakan API yang sudah disediakan oleh Kentico Cloud, yaitu sebuah CMS online yang menyediakan konten sebagai *service*. Kentico menyediakan REST API untuk melakukan pengambilan data, namun di sini, dicoba diterapkan GraphQL (Čechák, 2017).

**BAB III. METODOLOGI PENELITIAN**

* 1. **Metode Pengambilan Data**

Pengambilan data dilakukan dengan cara *scraping* pada situs Linkedin dan input secara manual melalui sistem informasi yang akan dibuat.

Pada pengambilan data di Linkedin, data yang diambil adalah data mahasiswa yang sudah mendaftarkan Politeknik Negeri Malang sebagai riwayat pendidikan mereka. Mekanisme pengambilan datanya adalah dengan menggunakan bot untuk melakukan login terotomatisasi, kemudian masuk ke halaman Politeknik Negeri Malang, lalu membuka halaman detail setiap card dari mahasiswa yang muncul. Dari halaman detail yang sudah dibuka, akan diambil data-data yang diperlukan.

Pada pengambilan data dari input manual, pengambilan data dilakukan dengan mengisikan data alumni melalui *form* yang sudah disediakan di sistem. *Form* tersebut memiliki *field* diantaranya berupa nama, tahun masuk dan lulus dari Politeknik Negeri Malang, jurusan yang diambil, pekerjaan saat ini, jabatan/posisi yang dipegang pada pekerjaan saat ini, dan email yang bisa dihubungi.

* 1. **Metode Pengembangan Sistem**

Metode pengembangan aplikasi yang penulis gunakan adalah metode Agile dengan framework Kanban. Alasan penulis memilih metode Kanban karena metode ini termasuk pada metode Agile yang memberikan kebebasan pada pengembang untuk melakukan penambahan fitur (incremental). Selain itu, kanban memiliki WIP (*Work-In-Progress*) yang memberi batasan berapa banyak tugas yang bisa diselesaikan. Sehingga, pengembang dapat fokus pada beberapa tugas terlebih dahulu.

* 1. **Fase pengembangan sistem**

Fase pengembangan sistem menggunakan metode Agile dengan *framework* Kanban memiliki fase sebagai berikut:

1. User Story, merupakan suatu kasus berupa cerita ketika pengguna sedang mengakses produk yang kita buat. User story berisi tentang *role* pengguna, kegiatan apa yang dilakukan, dan alasan pengguna melakukan kegiatan tersebut. Contoh dari user story adalah sebagai berikut:

“Sebagai Administrator, saya ingin mencari data alumni. Sehingga, saya bisa menemukan data alumni yang saya inginkan.”

Dari contoh user story di atas, dapat diketahui bahwa pengguna dengan *role* sebagai administrator memerlukan fitur untuk mencari data alumni. Dari sini, pengembang dapat menentukan bahwa sistem yang akan dikembangkan memerlukan suatu fitur pencarian alumni, yang dapat diakses oleh pengguna dengan role administrator, yang nantinya akan menampilkan data alumni sesuai yang dicari.

1. Menentukan prioritas user story. Dengan menentukan prioritas dari suatu user story, pengembang dapat mengetahui fitur mana yang harus dikerjakan terlebih dahulu. Sehingga, pengembang dapat mengetahui tujuannya dengan jelas dan pengerjaan produk menjadi lebih terstruktur.
2. Menentukan WIP (*Work In Progress*). Dengan adanya WIP, pengembang dapat membatasi fitur yang harus dikerjakan dan fokus untuk mengerjakan fitur-fitur yang dipilih terlebih dahulu. Jika salahsatu fitur sudah selesai, maka pengembang dapat memilih fitur lainnya yang ingin dikerjakan.
   1. **Proses Pengujian**

Metode pengujian akan dilakukan dengan cara membandingkan performa akses website sistem informasi pelacakan alumni Politeknik Negeri Malang menggunakan GraphQL, dengan sistem informasi pelacakan alumni Politeknik Negeri Malang dengan menggunakan metode pendistribusian data REST API.

Pengujian akan dilakukan dengan cara memuat suatu halaman yang sama dan memiliki masalah under-fetching, pada Sistem Informasi yang menggunakan GraphQL maupun yang menggunakan REST sebagai metode pendistribusian datanya. Kemudian, halaman akan di-*refresh* sebanyak 20 kali. Pada masing-masing sesi *refresh,* akan ada beberapa data yang dicatat pada masing-masing sistem informasi. Diantaranya adalah jumlah *request* ke *backend*, waktu muat *request* hingga selesai, dan waktu total memuat halaman.

Dari data-data tersebut, akan dirata-rata pada masing-masing sistem informasi dan akan dibandingkan. Sehingga, dapat diketahui kelebihan/kekurangan dari graphQL dan REST jika digunakan dalam pendistribusian data.

GraphQL dikatakan berhasil menyelesaikan masalah under-fetching jika sistem yang dikembangkan dengan mengimplementasikan GraphQL dapat melakukan pengaksesan endpoint lebih sedikit daripada sistem dengan metode REST API dalam memenuhi kebutuhan datanya, dan dapat meningkatkan performa dari website.

**BAB IV. ANALISIS DAN PERANCANGAN**

* 1. **Kebutuhan Fungsional**

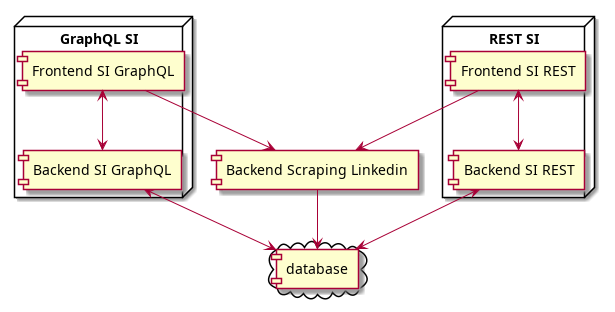
Kebutuhan fungsional dari sistem informasi pencarian alumni dengan graphQL dan sistem informasi pencarian alumni REST yang akan penulis kembangkan adalah:

* + 1. **Admin**

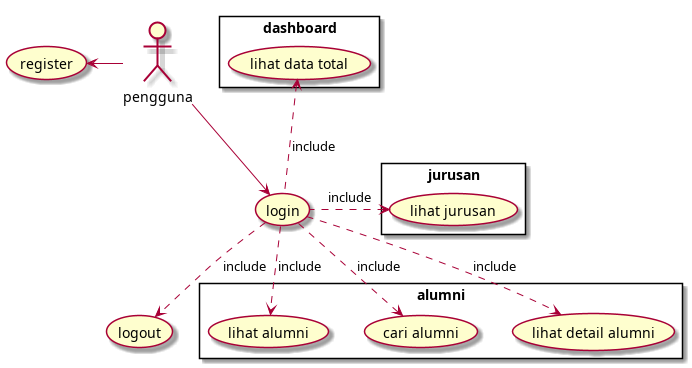
1. Mengelola jurusan, diantaranya melihat daftar jurusan, menambah jurusan baru, mengubah data jurusan, dan menghapus jurusan.
2. Autentikasi pengguna, diantaranya *login* dan *logout*.
3. Mengelola data alumni, diantaranya melakukan *scraping* dari Linkedin, pencarian alumni, mengubah data alumni, menghapus data alumni, melihat daftar alumni, dan melihat detail data dari alumni.
4. Mengelola pengguna, diantaranya melihat daftar pengguna, mengubah data pengguna, menambah pengguna, dan menghapus pengguna.
   * 1. **Pengguna**
5. Mengakses data jurusan, diantaranya melihat daftar jurusan.
6. Autentikasi pengguna, diantaranya *login*, *register*, dan *logout*.
7. Mengakses data alumni, diantaranya melakukan pencarian alumni, melihat daftar alumni, dan melihat detail data dari alumni.
   1. **Kebutuhan Non-Fungsional**

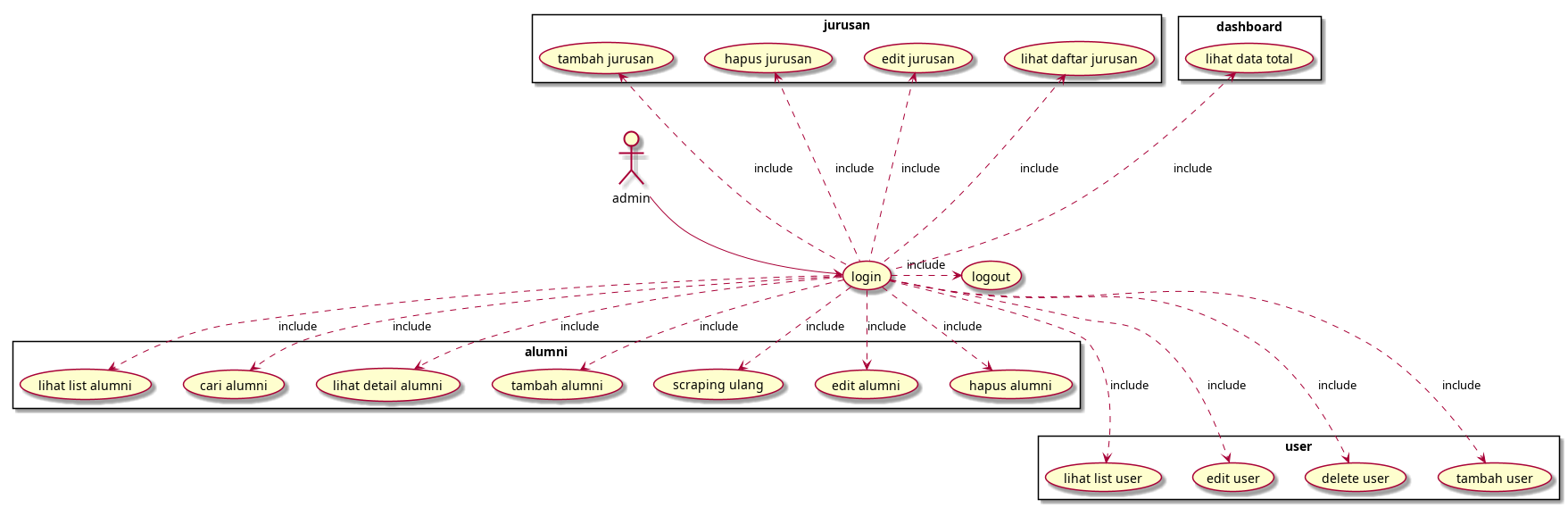
Kebutuhan non-fungsional dari sistem informasi pencarian alumni adalah:

1. Sistem informasi berjalan pada *web browser*.
2. *Frontend* dikembangkan dengan menggunakan React.js, baik sistem informasi yang menggunakan GraphQL maupun REST.
3. *Backend* dikembangkan dengan menggunakan *Node.js*, baik sistem informasi yang menggunakan GraphQL maupun REST.
4. Bahasa pemrograman yang digunakan untuk mengembangkan kedua sistem informasi adalah Typescript.
5. *Scraping* data alumni dari Linkedin dilakukan dengan menggunakan selenium.js.
6. Database yang digunakan adalah mongodb dan tersimpan pada *cloud service* milik mongodb. Sehingga, dapat diakses oleh backend GraphQL maupun REST.
   1. **Rancangan Sistem**
      1. **Arsitektur Sistem**

****

* + 1. **Use Case**

****

****

* + 1. **Struktur Data**

Struktur data dari sistem informasi yang akan penulis kembangkan adalah:

1. *Collection* alumni\_inputs, untuk menyimpan data alumni hasil input dari sistem.

|  |  |
| --- | --- |
| **Column Name** | **Tipe Data** |
| name | String |
| entry\_year | Number |
| graduate\_year | Number |
| major | String |
| work\_at | String |
| work\_position | String |
| email | String |
| data\_source | String |

1. *Collection* alumni\_linkedins, untuk menyimpan data alumni hasil *scraping* dari Linkedin. Meski bentuk datanya hampir sama dengan data dari alumni\_inputs, data dari hasil *scraping* Linkedin sengaja diletakkan pada *collection* berbeda untuk mempermudah pengelolaan data dari sumber yang berbeda seperti melihat data hasil dari input sistem saja, dan sebaliknya.

|  |  |
| --- | --- |
| **Column Name** | **Tipe Data** |
| name | String |
| entry\_year | String |
| graduate\_year | String |
| major | String |
| work\_at | String |
| work\_position | String |
| email | String |
| data\_source | String |

1. *Collection* major, berisi nama dari jurusan.

|  |  |
| --- | --- |
| **Column Name** | **Tipe Data** |
| name | String |

1. *Collection* user, berisi data dari pengguna. Baik administrator maupun pengguna biasa.

|  |  |
| --- | --- |
| **Column Name** | **Tipe Data** |
| name | String |
| email | String |
| password | String |
| level | Number |

* + 1. **Epic**

*Epic* adalah suatu *user story* yang memiliki cangkupan cukup luas. Sehingga, memiliki kemungkinan besar untuk dapat di-*breakdown*menjadi beberapa *user story* yang lebih detail.

*Epic* yang penulis buat pada pengembangan sistem informasi kali ini adalah:

1. Admin: Mengelola Jurusan. Sebagai administrator, saya ingin mengelola jurusan. Sehingga, saya dapat mengelola jurusan yang ada di sistem informasi alumni.
2. Admin: Autentikasi. Sebagai administrator, saya ingin melakukan autentikasi pada sistem. Sehingga, saya dapat mengakses fitur-fitur yang ada di website sesuai dengan wewenang saya.
3. Admin: Mengelola Pengguna. Sebagai administrator, saya ingin mengelola user. Sehingga, saya dapat mengelola data user yang dapat login ke sistem.
4. Admin: Mengelola data alumni. Sebagai administrator, saya ingin dapat mengelola data alumni. Sehingga, saya bisa tahu data alumni yang sudah terdaftar pada sistem.
5. User: Autentikasi. Sebagai user, saya ingin melakukan autentikasi pada sistem. Sehingga, saya dapat mengakses fitur-fitur yang ada di website sesuai dengan wewenang saya.
6. User: Melacak data alumni. Sebagai user, saya ingin melacak data alumni. Sehingga, saya bisa mengetahui data dari alumni yang saya cari.
7. User: Melihat data jurusan. Sebagai user, saya ingin dapat melihat data jurusan. Sehingga, saya tahu jurusan apa saja yang ada pada kampus Politeknik Negeri Malang.
   * 1. **User Story**

User Story di sini adalah hasil *breakdown* dari Epic yang sudah dibuat diatas. Karena penulis akan mengembangkan 2 sistem informasi yang sama, maka *user story* untuk kedua sistem informasi adalah sama. Berikut adalah:

1. Admin: Melihat list jurusan. Sebagai admin, saya ingin dapat melihat daftar jurusan yang ada. Sehingga, ketika menambah alumni baru, saya bisa memastikan bahwa jurusan dari alumni tersebut sudah tersedia.
2. Admin: Menambah jurusan baru. Sebagai admin, saya ingin dapat menambahkan jurusan baru. Sehingga, saya dapat menambahkan jurusan yang belum tersedia di sistem.
3. Admin: Mengubah data jurusan. Sebagai admin, saya ingin dapat mengubah data jurusan. Sehingga, apabila ada salah input, maka data bisa diperbarui.
4. Admin: Menghapus jurusan. Sebagai admin, saya ingin dapat menghapus data jurusan yang sudah ada. Sehingga, nama jurusan yang sudah tidak ada, tidak akan muncul ketika memilih jurusan untuk menambahkan alumni.
5. Admin: login. Sebagai admin, saya ingin dapat melakukan login. Sehingga, saya dapat mengakses fitur-fitur admin pada website.
6. Admin: logout. Sebagai admin, saya ingin dapat melakukan logout. Sehingga, saya dapat mengakhiri sesi saya dari sistem.
7. Admin: melihat daftar user. Sebagai admin, saya ingin dapat melihat daftar user. Sehingga, saya dapat melihat daftar beberapa user sekaligus.
8. Admin: mengubah data user. Sebagai admin, saya ingin dapat memperbarui data user yang sudah ada. Sehingga, saya bisa menambahkan admin baru dari user yang sudah ada, atau mereset data user yang lupa password.
9. Admin: Menghapus user. Sebagai admin, saya ingin dapat menghapus data user. Sehingga, saya dapat menghapus data user tertentu.
10. Admin: mencari data user. Sebagai admin, saya ingin dapat mencari user. Sehingga, saya dapat menemukan user yang saya inginkan.
11. Admin: melakukan *scraping* ulang dari linkedin. Sebagai admin, saya ingin melakukan scraping data alumni dari LinkedIn. Sehingga, saya bisa memperbarui data alumni dari Linkedin.
12. Admin: pencarian alumni. Sebagai admin, saya ingin mencari data alumni. Sehingga, saya bisa menemukan data alumni yang saya inginkan.
13. Admin: mengubah data alumni. Sebagai admin, saya ingin meng-*update* data alumni yang diinputkan manual. Sehingga, saya bisa mengubah data alumni jika ada kesalahan input.
14. Admin: menghapus data alumni. Sebagai admin, saya ingin menghapus data alumni. Sehingga, saya bisa menghapus data alumni yang tidak valid.
15. Admin: menambahkan data alumni. Sebagai admin, saya ingin menambahkan data alumni. Sehingga, saya bisa menambahkan data alumni selain melalui *scraping*.
16. Admin: melihat daftar alumni. Sebagai Admin, saya ingin melihat daftar data alumni. Sehingga, saya bisa melihat beberapa data alumni sekaligus.
17. Admin: melihat detail alumni. Sebagai Admin, saya ingin melihat detail data alumni. Sehingga, saya bisa melihat detail data setiap alumni.
18. User: login. Sebagai user, saya ingin dapat melakukan login. Sehingga, saya dapat mengakses fitur-fitur user pada website.
19. User: register. Sebagai user, saya ingin dapat melakukan register. Sehingga, saya dapat melakukan login pada website.
20. User: logout. Sebagai User, saya ingin dapat melakukan logout. Sehingga, saya dapat mengakhiri sesi pada website.
21. User: mencari data alumni. Sebagai User, saya ingin dapat melakukan pencarian alumni. Sehingga, saya dapat mengetahui data dari alumni yang saya inginkan.
22. User: melihat daftar alumni. Sebagai User, saya ingin dapat melihat daftar alumni. Sehingga, saya dapat melihat beberapa data alumni sekaligus.
23. User: melihat detail alumni. Sebagai User, saya ingin dapat melihat detail dari suatu alumni. Sehingga, saya dapat melihat detail data dari seorang alumni.
24. User: melihat daftar jurusan. Sebagai user, saya ingin dapat melihat data jurusan. Sehingga, saya tahu jurusan apa saja yang ada pada kampus Politeknik Negeri Malang.
    * 1. **A**

**BAB V. IMPLEMENTASI DAN PENGUJIAN**

* 1. **Implementasi**
     1. **Pembuatan Scraper Linkedin**

Scraper dibuat menggunakan Selenium.js dengan backend Node.js. Scraper dibuat pada *backend* tersendiri. Sehingga, dapat diakses oleh sistem informasi dengan GraphQL maupun sistem informasi dengan REST.

Langkah pertama yang penulis lakukan adalah membuat akun linkedin dan menambah koneksi dengan dengan akun linkedin yang sudah memasukkan politeknik negeri malang pada riwayat pendidikan dan memiliki banyak relasi. Kemudian memastikan bahwa tab alumni pada halaman Politeknik Negeri Malang dapat diakses dan memunculkan data alumni. Namun, ketika pertama kali akun dibuat, akun tersebut tidak bisa melihat data akun lain pada alumni Politeknik Negeri Malang. Sehingga, penulis mendiamkan akun tersebut selama 3 hari. Setelah itu, beberapa data alumni muncul pada bagian tab alumni di halaman Politeknik Negeri Malang, namun masih ada beberapa data alumni yang belum muncul (hanya muncul tulisan “Linkedin Member” tanpa bisa diklik). Setelah memastikan data berhasil dimuat, barulah pengembangan *backend* untuk *scraping* dimulai.

Pada pengembangan *backend* ini, pengembangan dilakukan dengan menggunakan bahasa typescript untuk memberikan batasan pada variabel agar tidak bisa menerima sembarang tipedata seperti pada javascript. Dengan begitu, variabel memiliki tipedata yang jelas dan tidak menimbulkan variabel yang tidak diketahui tipe datanya. Karena node.js tidak memiliki bawaan untuk menggunakan typescript, maka penulis melakukan pengaturan pada node.js dengan menambahkan modul typescript pada node.js dan menambahkan file tsconfig.json yang menjadi pengaturan untuk modul typescript, bagaimana typescript bertindak untuk mengatur kode yang ditulis. Isi dari tsconfig.json adalah:

{

"compilerOptions": {

"module": "commonjs",

"esModuleInterop": true,

"target": "es6",

"noImplicitAny": true,

"moduleResolution": "node",

"sourceMap": true,

"outDir": "dist",

"baseUrl": "src"

}

}

Setelah typescript sudah dipasang, pengembangan dilanjutkan dengan memasang modul selenium untuk javascript, modul express, dan modul mongoose sebagai database untuk menyimpan data ke database setelah scraping selesai. Setelah semua modul yang dibutuhkan terpasang, pengembangan scraper untuk linkedin dimulai.

Pengembangan *scraper* dimulai dengan membuka browser chrome webdriver dan melakukan login menggunakan akun yang sudah dibuat. Dimana setelah login, linkedin akan langsung mengarahkan ke halaman alumni dari Politeknik Negeri Malang. Jika login berhasil, *scraper* akan menunggu hingga data alumni sudah dimuat dengan batas waktu tunggu 10 detik. Jika lebih dari itu, maka *scraper* akan berhenti dan menganggap data alumni tidak ditemukan. Setelah data alumni berhasil dimuat, maka *scraper* akan memulai untuk men-*scroll* halaman untuk memuat semua data alumni yang ada hingga halaman tidak bisa lagi di-*scroll.* Setelah semua data alumni sudah muncul, maka *scraper* mengambil masing-masing url alumni yang sudah ada dan menyimpannya pada suatu *array*. *Array* tersebut nantinya akan di-*looping* dan pada tiap iterasi, akan membuka tab baru dengan alamat sesuai dengan array. Setelah membuka halaman alumni yang dimaksud, *scraper* melakukan *scroll down* agar data *experience* dari alumni dapat dibaca dan diambil datanya. Beberapa alumni terkadang tidak memberikan data yang diperlukan. Sehingga, menyebabkan data yang tersimpan adalah berupa *string* kosong (“”). Setelah data alumni berhasil diambil, maka data alumni tersebut disimpan dalam suatu objek dan dimasukkan pada *array* untuk alumni. Proses ini diulangi hingga semua alamat pada *array* berhasil dibuka dan diambil datanya. Setelah selesai pada proses memuat data alumni, sistem akan mengosongkan *collection* dengan nama “alumni\_linkedins” dan menyimpan data alumni yang baru. Pengosongan *collection* dilakukan untuk menghindari adanya duplikasi data atau adanya 2 data alumni yang sama.

Setelah berhasil menyimpan data pada *database*, maka *scraper* akan menutup browser dan mengakhiri proses *scraping* linkedin.

* + 1. **Pengembangan Backend REST**

aaaaaaa

* + 1. **Pengembangan Frontend REST**

aaaaaaaa

* + 1. **Pengembangan Backend GraphQL**

Aaaaa

* + 1. **Pengembangan Frontend GraphQL**

aaaaaa

* 1. **Pengujian**
  2. aaaa