SKRIPSI

IMPLEMENTASI DAN ANALISIS KINERJA SISTEM PENCARIAN BERBASIS *FUZZY SEARCH*. STUDI KASUS: SITUS *DIGITAL LIBRARY* FAKULTAS TEKNIK UNIVERSITAS HASANUDDIN

Disusun dan diajukan oleh

GILBERT HYMAN GOES

D121171306

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2022

LEMBAR PENGESAHAN SKRIPSI

IMPLEMENTASI DAN ANALISIS KINERJA SISTEM PENCARIAN BERBASIS *FUZZY SEARCH*. STUDI KASUS: SITUS *DIGITAL LIBRARY* FAKULTAS TEKNIK UNIVERSITAS HASANUDDIN

Disusun dan diajukan oleh:

GILBERT HYMAN GOES
D121171306

Telah dipertahankan dihadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika Fakultas Teknik Universitas Hasanuddin pada tanggal 30 November 2022 dan dinyatakan telah memenuhi syarat kelulusan.

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping,
Dr. Amil Ahmad Ilham, S.T., M.IT.	Iqra' Aswad, S.T., M.T
NIP. 19731010 199802 1 001	NIP. 19901128 201904 3 001

Ketua Departemen,

Dr. Indrabayu, S.T., M.T., M.Bus.Sys.
NIP. 19750716 200212 1 004

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Gilbert Hyman Goes
NIM	: D121171306
Departemen	: Teknik Informatika
Jenjang	: S1

Menyatakan dengan ini karya tulisan saya berjudul:

IMPLEMENTASI DAN ANALISIS KINERJA SISTEM PENCARIAN BERBASIS FUZZY SEARCH. STUDI KASUS: SITUS *DIGITAL LIBRARY* FAKULTAS TEKNIK UNIVERSITAS HASANUDDIN

Adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 30 November 2022

Yang Menyatakan,

Gilbert Hyman Goes

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus yang senantiasa menjadi penopang dan teman terbaik penulis sepanjang hidupnya. Hanya oleh kasih karunia dan kemurahan-Nya akhirnya penulis dapat menyelesaikan penulisan skripsi yang berjudul **"Implementasi dan Analisis Kinerja Sistem Pencarian Berbasis *Fuzzy Search*. Studi Kasus: Situs *Digital Library* Fakultas Teknik Universitas Hasanuddin"** guna memenuhi salah satu persyaratan dalam menyelesaikan jenjang Strata-1 di Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari kesempurnaan karena menyadari segala keterbatasan yang ada. Dalam penulisan skripsi ini penulis menghadapi berbagai kendala dan masalah, namun karena usaha yang maksimal dan kemampuan yang Tuhan berikan kepada penulis serta bantuan dan dukungan dari berbagai pihak, maka penulisan skripsi ini dapat selesai. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Tuhan Yesus Kristus, penolong yang selalu ada di setiap langkah hidup penulis.
2. Kedua orang tua penulis, Bapak Ismail Gus dan Ibu Lianney Yanti Bunadi yang selalu memberikan kasih sayang, nasehat, motivasi, dukungan, dan doa kepada penulis.
3. Bapak Dr. Amil Ahmad Ilham, S.T., M.IT., selaku pembimbing utama dan Bapak Iqra, S.T., M.T., selaku pembimbing pendamping yang senantiasa menyediakan waktu, tenaga, pikiran, dan perhatian yang luar biasa dalam mengarahkan penulis dalam penyusunan tugas akhir ini.
4. Bapak Robert, Bapak Zainuddin dan Ibu Yuanita serta segenap staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran penyelesaian tugas akhir penulis.

5. Segenap Dosen dan Staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah banyak membantu semasa perkuliahan hingga penyelesaian tugas akhir penulis.
6. Kakak-kakak Adhyaksa khususnya kak Wawan, kak Alam, kak Yakip, dan kak Rafi yang mengajarkan banyak hal dan menjadi inspirasi penulis dalam mengarungi kehidupan sebagai mahasiswa di Departemen Informatika Fakultas Teknik Universitas Hasanuddin.
7. Rekan-rekan kerja Fairtech Pte. Ltd. yang selalu memberikan semangat kepada penulis untuk segera menyelesaikan jenjang Strata-1.
8. Saudara seperjuangan penulis RECOGN17ER yang telah menemani dan mendukung perjalanan penulis sekaligus tempat berbagi keluh kesah selama menjadi mahasiswa teknik di Departemen Informatika Fakultas Teknik Universitas Hasanuddin.
9. Marwah yang telah menemani dan menyemangati penulis saat pengerjaan skripsi.
10. Adik Agil yang telah menemani dan mendukung penulis selama proses pengerjaan skripsi.
11. Seluruh pihak yang tidak sempat disebutkan satu persatu yang telah banyak meluangkan tenaga, waktu, dan pikiran selama penyusunan tugas akhir ini.

ABSTRAK

Fitur pencarian saat ini merupakan bagian terpenting dalam sistem yang memiliki data dalam jumlah besar. Salah satu jenis sistem yang memiliki data dalam jumlah besar adalah *digital library*. *Digital library* adalah sebuah sistem untuk menyimpan buku atau karya ilmiah dalam format digital yang dapat diakses melalui perangkat elektronik yang terhubung ke internet. Dibutuhkan sebuah sistem yang andal untuk menjalankan sebuah sistem *digital library* bukan hanya sekedar data dapat disimpan dan diakses, melainkan juga proses dari pencarian data dari sistem, dengan utilitas pencarian yang andal akan memberikan pengalaman yang baik bagi pengguna situs. Fakultas Teknik Universitas Hasanuddin telah memiliki sistem *digital library* namun, masih terdapat kekurangan pada bagian utilitas pencarian yang masih berbasis *full-text search*. Penelitian ini mengimplementasikan sistem pencarian berbasis *fuzzy search* dimana pencarian dengan algoritma *fuzzy search* ini memiliki *indexing time*, *search response time*, dan hasil relevansi pencarian yang lebih baik dari *full-text search*. Selain itu, dilaksanakan uji coba ketiga sistem arsitektur perangkat lunak untuk membandingkan sistem pencarian elasticsearch dan typesense yang berbasis *fuzzy search* dan mencari arsitektur perangkat lunak terbaik untuk diimplementasikan ke Situs *Digital Library* Fakultas Teknik Universitas Hasanuddin. Hasil analisis menunjukkan elasticsearch unggul dalam segi *indexing time* dengan rata-rata hasil waktu 146,92 ms sedangkan, typesense unggul dalam segi *search response time* dengan memberikan rata-rata hasil waktu untuk percobaan fitur *basic search* 5705,12 ms. Arsitektur perangkat lunak dengan sistem pencarian gabungan merupakan arsitektur terbaik karena implementasi dua sistem pencarian dapat menutupi kekurangan dari masing-masing sistem pencarian.

Kata Kunci: *Search Engine,* Elasticsearch, Typesense, *Fuzzy Search, Digital Library, Management Data*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Fitur pencarian saat ini merupakan bagian terpenting dalam sistem yang memiliki data dalam jumlah besar. Dengan fitur pencarian pengguna dimudahkan dalam memproses pencarian data yang diinginkan. Fitur pencarian yang akurat dan memiliki relevansi yang tinggi terhadap masukan kata kunci pengguna akan sangat disukai oleh pengguna. Sebaliknya, fitur pencarian yang kurang akurat tidak terlalu disukai oleh pengguna. Hal ini berhubungan dengan User Experience (UX) yang dirasakan oleh pengguna, sistem pencarian yang akurat dapat memberikan pengalaman yang baik bagi pengguna untuk mendapatkan informasi. Keuntungan lain yang bisa didapatkan dengan sistem yang memiliki pencarian yang akurat dengan relevansi tinggi dapat meningkatkan kredibilitas dari sistem tersebut, meningkatkan *traffic* pengguna, dan memenangkan persaingan terhadap sistem sejenis.

Sistem pencarian berbasis fuzzy search umumnya diintegrasikan dengan sistem yang memiliki sejumlah data besar untuk mempermudah proses pencarian. Adanya algoritma *damerau-levenshtein distance* yang digunakan pada sistem pencarian berbasis *fuzzy search*, dapat memberikan beberapa fitur tambahan ke dalam mesin pencari, yaitu: kemampuan untuk memprediksi kata kunci *typo* yang biasanya dilakukan pengguna saat melakukan proses pencarian dan memberikan data yang relevan berdasarkan kata kunci yang dimasukkan, dapat memberikan saran kata kunci yang benar jika pengguna memasukkan kata kunci *typo*, dan memberikan fitur *autocomplete suggest* saat pengguna sedang mengetikkan kata kunci. Namun sayangnya hal ini tidak pernah ditemukan pada sistem yang dimiliki oleh universitas yang berada di Indonesia. Pengembangan hanya berfokus sampai sistem berjalan namun,

mengabaikan bagian terpenting untuk sistem dengan jumlah data besar yaitu harus didukung dengan arsitektur sistem yang memberikan kinerja yang baik. Hal ini menyebabkan pencarian literatur pada situs *digital library* menjadi sulit dan hanya dijadikan sebagai pilihan kedua dalam pencarian literatur oleh *civitas academica*. Hal yang sama terjadi pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin yang berjalan saat ini masih memiliki beberapa kelemahan, yaitu: hanya menyediakan fitur *basic search* untuk memproses pencarian yang belum mendukung fitur toleransi *typo*, pemberian saran kata kunci, dan *autocomplete suggest*. Selain itu, pada saat proses pencarian masih terdapat kegagalan ketika melakukan proses indexing pada dokumen sistem pencarian, serta atribut pencarian pada indeks yang masih kurang. Beberapa kekurangan tersebut yang menjadikan *user experience* dari pengguna berkurang dan lebih memilih situs lain untuk mencari literatur. Berdasarkan hal tersebut, penulis akan mengembangkan situs *Digital Library* Fakultas Teknik Universitas Hasanuddin dengan mesin pencarian berbasis *fuzzy search* yang mendukung fungsi toleransi *typo*, pemberian saran kata kunci, *autocomplete suggest*, serta melakukan pembaruan fitur pengindeksan pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin.

Berdasarkan penelitian terdahulu yang dilakukan oleh Rahmat Firman berjudul *Perancangan Sistem Pencarian Berbasis Full-Text Search Engine Dengan Elasticsearch Studi Kasus: Situs Digital Library Fakultas Teknik Universitas Hasanuddin*, telah berhasil melakukan implementasi salah satu *inbuilt system search engine* menggunakan elasticsearch pada *Digital Library* Fakultas Teknik Universitas Hasanuddin menggunakan metode *full-text search*. Berdasarkan penelitian tersebut didapatkan hasil *f-measure* pemberian hasil kata kunci yang relevan di angka 83 % dengan rata-rata waktu pencarian terbaik 263 *milisecond*. Dari hasil penelitian tersebut, masih dapat dilakukan

peningkatan dalam segi performa kecepatan, relevansi pencarian, dan kemampuan mengoreksi kata *typo* pada sistem *Digital Library* Fakultas Teknik Universitas Hasanuddin hingga mencapai tingkat yang optimal. *Fuzzy search* juga memiliki tingkat akurasi yang cukup tinggi pada penelitian yang dilakukan oleh Ichsan Taufik, dkk: Implementasi *Fuzzy Search* Untuk Pendeteksi Kata Asing Pada Dokumen Microsoft Word (2017) dengan tingkat akurasi sistem pendeteksian kata asing sebesar 89,6 %.

Sistem pencarian merupakan komponen vital yang terdapat pada sistem *Digital Library* Fakultas Teknik Universitas Hasanuddin yang saat ini berjalan menggunakan sistem pencarian elasticsearch berbasis *full-text search*. Pada penelitian ini penulis akan melakukan optimasi menggunakan elasticsearch dan implementasi sistem pencarian typesense berbasis *fuzzy search* dan melakukan analisis terhadap kedua sistem pencarian. Penulis akan melakukan analisis performa kedua sistem berdasarkan parameter-parameter: *indexing time*, *indexing cpu usage percentage*, *search response time,* relevansi hasil pencarian, dan analisis metode *ranking*. Terdapat dua jenis pencarian, yaitu *basic search* dan *advanced search*. Kedua jenis sistem pencarian ini akan dilakukan uji coba tiga arsitektur perangkat lunak: Sistem pencarian menggunakan elasticsearch, typesense, dan hibrid, yakni gabungan antara elasticsearch dan typesense). Setelah dilakukan analisis dan membandingkan dua sistem pencarian, akan ditentukan sistem pencarian yang akan digunakan pada tahap produksi elasticsearch, typesense, ataupun hibrid.

Berdasarkan latar belakang yang telah diuraikan, penulis akan melakukan penelitian berupa optimasi elasticsearch dan implementasi sistem pencarian typesense pada *Digital Library* Fakultas Teknik Universitas Hasanuddin serta, mengukur performansi kedua sistem

pencarian. Maka penulis menjadikan penelitian tersebut sebagai tugas akhir dengan judul **"Implementasi dan Analisis Kinerja Sistem Pencarian Berbasis *Fuzzy Search.* Studi Kasus: Web Digital Library Fakultas Teknik Universitas Hasanuddin"**.

1.2. Rumusan Masalah

Berdasarkan latar belakang di atas, penulis merumuskan permasalahan, yaitu:

1. Bagaimana cara melakukan optimasi sistem pencarian elasticsearch dari *full-text search* menjadi *fuzzy search* pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin ?
2. Bagaimana cara melakukan implementasi sistem pencarian typesense berbasis *fuzzy search* pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin ?
3. Bagaimana cara menemukan implementasi arsitektur perangkat lunak terbaik untuk diaplikasikan pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin ?
4. Bagaimana cara mengukur kinerja dari sistem pencarian typesense berbasis *fuzzy search* pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin dan membandingkannya dengan sistem pencarian elasticsearch ?

1.3. Tujuan Penelitian

Tujuan akhir dari penelitian ini:

1. Melakukan optimasi sistem pencarian elasticsearch dari metode *full-text search* menjadi *fuzzy search* pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin.

2. Melakukan implementasi sistem pencarian typesense berbasis *fuzzy search* pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin.
3. Menemukan implementasi arsitektur perangkat lunak terbaik untuk diaplikasikan pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin.
4. Mengukur kinerja dari sistem pencarian typesense berbasis *fuzzy search* pada situs *Digital Library* Fakultas Teknik Universitas Hasanuddin dan membandingkannya dengan sistem pencarian elasticsearch.

1.4. Manfaat Penelitian

Adapun manfaat dari penelitian ini:

1. Mengembangkan situs *Digital Library* Fakultas Teknik Universitas Hasanuddin.
2. Meningkatkan *user experience* (UX) pengguna situs *Digital Library* Fakultas Teknik Universitas Hasanuddin.
3. Memberikan sumbangan ilmiah berupa data hasil kinerja sistem pencarian *elasticsearch* dan *typesense* yang berbasis *fuzzy search*.

1.5. Batasan Masalah

Batasan masalah dari penelitian ini, yaitu atribut data jurnal yang digunakan untuk proses pencarian adalah judul, penulis, abstrak, dan *related keyword (tag)* dari jurnal penelitian.

1.6. Sistematika Penulisan

Laporan tugas akhir ini dibagi dalam lima bab yang tersusun secara sistematis, yaitu:

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang pengambilan topik sistem pencarian berbasis elasticsearch dan typesense berbasis *fuzzy search* pada sistem *Digital Library* Fakultas Teknik Universitas Hasanuddin, perumusan masalah, tujuan penelitian, batasan masalah, manfaat penulisan, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini menjelaskan konsep dasar dan landasan teori dari komponen-komponen yang digunakan pada penelitian ini. Teori-teori yang akan dijelaskan yaitu konsep dasar sistem informasi, *digital library*, *fuzzy search*, *search engine*, *elasticsearch*, *typesense*, docker, grails, dan metode analisis.

BAB III METODOLOGI PENELITIAN

Bab ini menjelaskan mengenai metode yang digunakan dalam penelitian, mencakup tempat dan waktu penelitian, instrumen penelitian, prosedur penelitian, gambaran umum sistem, dan metode analisis sistem.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil penerapan sistem pencarian berbasis *fuzzy search* menggunakan elasticsearch dan typesense, serta hasil pengujian sistem dan pembahasan dari hasil pengujian yang dilakukan.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan dan saran dari penelitian yang dilakukan dan saran-saran yang dapat dilakukan pada penelitian selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1. Konsep Dasar Sistem Informasi

Menurut Tata Sutabri (2012) dalam bukunya yang berjudul *Analisis Sistem Informasi*, terdapat dua pendekatan dalam pendefinisian sistem, yaitu kelompok yang menekankan pada prosedur dan kelompok yang menekankan pada elemen atau komponennya. Pendekatan yang menekankan pada prosedur mendefinisikan sistem sebagai suatu jaringan kerja prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Sedangkan pendekatan sistem yang lebih menekankan pada elemen atau komponen mendefinisikan sistem sebagai kumpulan elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Kedua pendefinisian tersebut benar dan tidak bertentangan yang berbeda adalah cara pendekatannya.

Informasi merupakan proses lebih lanjut dari data yang sudah memiliki nilai tambah informasi, informasi adalah data yang telah diklasifikasikan, diolah, atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan. Fungsi utama informasi adalah menambah pengetahuan atau mengurangi ketidakpastian pemakai informasi. Informasi yang disampaikan kepada pengguna informasi merupakan hasil dari data yang dimasukkan ke dalam pengolahan. Dalam kebanyakan pengambilan keputusan yang kompleks, informasi hanya dapat menambah kemungkinan kepastian atau mengurangi bermacam-macam pilihan. Informasi yang disediakan bagi pengambil keputusan memberikan suatu kemungkinan faktor risiko pada tingkat-tingkat pendapat yang berbeda.

Berdasarkan definisi sistem dan informasi yang telah dijabarkan, sistem informasi dapat didefinisikan sebagai sebuah jaringan kerja yang

berisi kumpulan data informasi yang dapat menambah wawasan ataupun menyelesaikan masalah bagi penggunanya. Berdasarkan penelitian Yaser Hasan Salem Al-Mamary (2014) dalam jurnal *The Role of Different Types of Information Systems In Business Organizations: A Review*, terdapat 14 jenis sistem informasi:

1. *Transaction Processing Systems* (TPS)

   Transaction processing systems (TPS) adalah jenis sistem digital dasar yang biasanya digunakan pada level operasional dari suatu instansi atau perusahaan. Sistem TPS digunakan untuk menangani hal-hal yang bersangkutan dengan proses perekaman transaksi harian dalam instansi atau perusahaan, seperti perekaman nota pembelian harian.

2. *Process Control Systems*

   Process control systems adalah jenis sistem digital yang digunakan untuk melakukan pemantauan terhadap proses kerja sistem fisik yang berjalan. Misalnya sistem informasi pemantauan kelembaban tanaman padi.

3. *Management Information Systems* (MIS)

   Management information systems adalah jenis sistem digital yang dapat menyimpan dan menampilkan informasi yang dapat digunakan oleh pengguna untuk mengambil keputusan atau menyelesaikan suatu masalah.

4. *Decision Support Systems* (DSS)

   Decision support systems adalah jenis sistem digital yang bertujuan untuk menerima, memproses, dan menganalisis data yang akan mempengaruhi seseorang dalam pengambilan keputusan. Jenis

sistem tersebut biasa digunakan untuk membantu dalam pengambilan keputusan yang kompleks serta, DSS tidak digunakan untuk mengambil keputusan secara independen melainkan hanya membantu penggunanya.

5. *Executive Information Systems* (EIS)

Executive information systems adalah jenis sistem digital yang dikembangkan untuk memantau informasi penting berdasarkan sumber yang beragam baik dari internal maupun eksternal, misalnya dari MIS, DSS, ataupun sumber lainnya yang dibutuhkan. EIS merupakan sistem yang dirancang oleh instansi untuk melakukan pemantauan data atau informasi yang dimana sistem tersebut memiliki kemampuan untuk mengolah data yang abstrak dan menyajikannya menjadi suatu informasi yang sederhana dan mudah dibaca.

6. *Expert Systems*

Expert systems adalah jenis sistem digital yang menjalankan *artificial intelligence* (AI) pada sistemnya dengan tujuan menyelesaikan permasalahan yang rumit dan kompleks. Berbeda dengan DSS, *expert systems* menghasilkan keputusan akhir bagi pembuat keputusan.

7. *Knowledge Management Systems*

Knowledge management systems adalah sebuah sistem informasi adalah jenis sistem digital yang bertujuan untuk menyajikan dan memberikan informasi bagi pengguna dalam menyelesaikan suatu tugas atau masalah dengan lebih terstruktur. Contoh dari sistem informasi ini: *FAQ content, forum or community feature,* dan lain-lain.

8. *Strategic Information Systems*

Strategic information systems adalah sistem digital yang mengumpulkan semua sumber informasi esensial yang digunakan untuk mencapai tujuan tertentu. Pada perusahaan, sistem informasi tersebut digunakan dengan tujuan mengembangkan pasar dari bisnis yang dilakukan.

9. *Functional Business Systems (Information Systems for Functional Perspective)*

Functional business systems adalah sistem digital yang berfokus pada operasional dan manajemen untuk mendukung kegiatan dari suatu instansi. Contoh dari sistem informasi tersebut adalah sistem informasi akuntansi, keuangan, pemasaran, atau manajemen operasional.

10. *Sales and Marketing Information Systems*

Sales and marketing information systems adalah sistem digital yang digunakan untuk memasarkan produk atau jasa yang dijual oleh suatu instansi atau perusahaan.

11. *Manufacturing and Production Information Systems*

Manufacturing and production information systems adalah sistem digital yang dimanfaatkan oleh instansi atau perusahaan untuk memantau kualitas produksi dari awal produksi produk hingga produk tersebut sampai ke tangan konsumen.

12. *Finance and Accounting Information Systems*

Finance and accounting information systems adalah sistem digital yang digunakan untuk mengelola dan menyediakan informasi terkait semua kegiatan finansial yang ada pada suatu instansi.

13. *Human Resource Information Systems*

Human resource information systems adalah sistem digital yang menyediakan informasi terkait pengelolaan sumber daya manusia yang tersedia dengan tujuan untuk mengambil keputusan tertentu dalam suatu instansi atau perusahaan.

2.2. Digital Library

Dunia perpustakaan semakin hari semakin berkembang dan bergerak ke depan. Perkembangan dunia perpustakaan ini didukung oleh perkembangan teknologi informasi dan pemanfaatannya yang telah merambah ke berbagai bidang. Hingga saat ini tercatat beberapa masalah di dunia perpustakaan yang coba diperbaiki dengan menggunakan pendekatan teknologi informasi. Berdasarkan data dan dokumen yang disimpan di perpustakaan, awalnya dimulai dari perpustakaan tradisional yang hanya terdiri dari kumpulan koleksi buku tanpa katalog, kemudian muncul perpustakaan semi modern yang menggunakan katalog (indeks). Katalog mengalami metamorfosa menjadi katalog elektronik yang lebih mudah dan cepat dalam temu kembali informasi koleksi yang disimpan di perpustakaan, yaitu dengan munculnya *digital library* yang memiliki keunggulan dalam kecepatan pengaksesan karena berorientasi ke data digital dengan akses melalui jaringan internet. Menurut Ummi Rodliyah (2012) dalam artikelnya *Perpustakaan Digital dan Prospeknya Menuju Resource Sharing*, *digital library*, terdapat beberapa alasan pentingnya *digital library*, yaitu:

1. *Digital library* dapat diakses di mana saja.

2. *Digital library* menawarkan berbagai macam cara penelusuran dan temu kembali yang canggih dengan menyediakan basis data secara elektronik sehingga memudahkan kepada pengguna untuk mengakses informasi.
3. *Digital library* memberikan fasilitas untuk memudahkan penyebaran literatur.
4. *Digital library* dapat membantu pengguna mendapatkan informasi yang mutakhir. *Digital library* memungkinkan untuk dapat mengakses informasi-informasi berseri (*periodical collection*) dengan *digital publishing*.
5. *Digital library* tidak lagi dibatasi oleh ruang, waktu, bahasa, dan budaya, sehingga lebih memudahkan penggunaan informasi. Informasi yang beragam dari berbagai belahan dunia dengan beragam bahasa dan berbagai budaya memudahkan penelusuran.
6. *Digital library* meningkatkan kolaborasi antarpengguna untuk meningkatkan proses penyebaran dan penggunaan informasi.
7. *Digital library* menurunkan kesenjangan literasi di berbagai tempat karena mudahnya mengakses informasi yang terdapat di dalamnya.

Digital library adalah suatu perpustakaan dalam jaringan yang menyimpan data berupa karya tulis, gambar, atau suara dalam bentuk berkas elektronik dan mendistribusikannya menggunakan protokol elektronik melalui jaringan internet. Berdasarkan definisi *digital library* dan jenis-jenis sistem informasi yang telah dijelaskan di atas, *digital library* termasuk ke dalam jenis *management information system* yaitu situs yang dimanfaatkan untuk menyimpan, mengkategorikan sekaligus menampilkan karya tulis, gambar atau suara. Menurut Ummi Rodliyah (2012) dalam artikelnya yang berjudul *Perpustakaan Digital dan Prospeknya Menuju Resource Sharing*, *digital library* memiliki tiga karakteristik utama:

1. Menggunakan teknologi yang mengintegrasikan kemampuan menciptakan, mencari, dan menggunakan informasi dalam berbagai bentuk dalam sebuah jaringan yang tersebar luas.
2. Memiliki koleksi yang mencakup data dan metadata yang saling mengaitkan berbagai data, baik di lingkungan internal maupun eksternal.
3. Merupakan kegiatan mengoleksi dan mengatur sumber daya digital yang dikembangkan bersama-sama komunitas pemakai jasa untuk memenuhi kebutuhan informasi mereka. Untuk itu, perpustakaan digital merupakan integrasi berbagai institusi yang memilih, mengoleksi, mengolah, merawat, dan menyediakan informasi secara meluas ke berbagai komunitas.

Berdasarkan hasil pengamatan dari beberapa *digital library* di Indonesia, terdapat beberapa spesifikasi fitur dari sebuah situs *digital library* (Winarko, 2009), sebagai berikut:

1. Fitur keanggotaan. Fitur ini dimaksudkan untuk membedakan hak akses antara pengguna terdaftar dengan pengguna lainnya. Pengguna yang terdaftar memiliki keuntungan berupa hak akses terhadap keseluruhan informasi yang tersedia. Salah satu perpustakaan yang memiliki fitur keanggotaan adalah ITB *central library*. Sedangkan perpustakaan digital yang belum menerapkan fitur ini adalah Perpustakaan Institut Bisnis dan Informatika Indonesia.
2. Fitur pencarian. Fitur ini dapat digunakan oleh pengguna untuk mengakses informasi secara cepat dengan menggunakan *search engine* yang tersedia. Fitur ini dapat digunakan untuk mencari informasi baik dalam lingkup perpustakaan itu sendiri maupun lingkup global. Hampir semua perpustakaan memiliki fitur ini terkecuali perpustakaan islam.com.
3. Fitur *link* atau pranala. Fitur ini memungkinkan untuk pengunjung perpustakaan digital untuk mengakses perpustakaan digital lainnya

dengan cara menyediakan *link* menuju perpustakaan digital tersebut. *Link* memberikan keuntungan kepada pengunjung karena tidak perlu mencari *link* perpustakaan lainnya secara mandiri. Salah satu perpustakaan digital yang menerapkan fitur ini adalah ITB *Central Library*.

4. Fitur dwi bahasa. Fitur ini memberikan kemudahan bagi pengunjung perpustakaan digital yang tidak memiliki kemampuan berbahasa Indonesia yang cukup. Menyediakan alternatif bahasa pada perpustakaan digital dapat meningkatkan jangkauan pengguna domestik maupun internasional. Salah satu perpustakaan digital yang memiliki fitur ini adalah Perpustakaan Digital UIN Sunan Kalijaga Yogyakarta.
5. Fitur artikel. Fitur ini bervariasi antara satu perpustakaan digital dengan perpustakaan digital lainnya, mulai dari artikel yang bersifat ilmiah seperti penelitian ataupun artikel yang bersifat populer seperti warta. Biasanya fitur ini digunakan untuk menampilkan artikel yang dihasilkan oleh lembaga yang bersangkutan. Salah satu perpustakaan digital yang memiliki fitur ini adalah ITB *Central Library*.
6. *Folder* dan arsip. Fitur *folder* dan arsip dimaksudkan untuk menyimpan *file* atau artikel yang biasanya bukan *file* terkini. Salah satu perpustakaan digital yang memiliki fitur *folder* dan arsip adalah Perpustakaan Digital UIN Sunan Kalijaga Yogyakarta.

2.3. *Fuzzy Search*

Definisi *fuzzy search* menurut Ichsan Taufik, *et al.* (2017) dalam Jurnal *Implementasi Fuzzy Search Untuk Pendeteksi Kata Asing Pada Dokumen Microsoft Word* adalah suatu proses yang menempatkan halaman situs atau dokumen yang mungkin yang mungkin relevan dengan argumen pencarian bahkan ketika argumen tidak sesuai dengan informasi yang diinginkan. Sebuah *fuzzy search* dijalankan dengan pencocokan *fuzzy*,

yang mengembalikan daftar hasil berdasarkan kemungkinan relevansi meskipun pencarian kata argumen dan ejaan mungkin tidak tepat. *Fuzzy search* berfungsi untuk menemukan semua kata yang memiliki kemiripan dengan *query* yang diberikan pada dokumen yang ada.

Fuzzy search dapat mengkompresi kesalahan pengetikan *input* yang umum serta kesalahan yang diperkenalkan oleh pengenalan karakter *optic scanning* dokumen yang dicetak. Program pencocokan *fuzzy* biasanya menampilkan istilah yang tidak relevan dan relevan.. Hasil yang berlebihan mungkin juga untuk mengkombinasikan dengan beberapa arti, hanya dari salah satu makna yang dimaksud oleh pengguna. *Fuzzy search* adalah teknik yang sangat berguna untuk melakukan pencarian data dari suatu dokumen atau basis data yang didasarkan pada informasi lengkap ataupun sebagian guna memperoleh data yang akurat, dan dapat pula dijadikan pendeteksi bahasa-bahasa asing atau istilah-istilah yang menggunakan ejaan yang tepat dan tidak diketahui secara luas. *Fuzzy search* dikembangkan menjadi suatu teknik pencarian dengan menggunakan algoritma *levenshtein distance* ataupun algoritma *damerau-levenshtein distance*.

Levenshtein distance adalah algoritma yang dibuat oleh Vladimir Levenshtein pada tahun 1965. Perhitungan *edit distance* didapatkan dari matriks yang digunakan untuk menghitung jumlah perbedaan string antara dua *string*. Perhitungan jarak antara dua *string* ini ditentukan dari jumlah minimum operasi perubahan untuk membuat *string* A menjadi *string* B. Algoritma ini berjalan mulai dari pojok kiri atas sebuah *array* dua dimensi yang telah diisi sejumlah karakter *string* awal dan *string* target dan diberikan nilai *cost*. Nilai *cost* pada ujung kanan bawah menjadi nilai *edit distance* yang memberikan jumlah perbedaan dua *string*. Pada algoritma

levenshtein distance terdapat tiga operasi: *insertion, deletion,* dan *substitution*. Berikut ini adalah rumus dari algoritma *levenshtein distance*:

(1)

$$lev(a,b) = \begin{cases} |a| & jika\ |b| = 0 \\ |b| & jika\ |a| = 0 \\ lev(tail(a), tail(b)) & jika\ [0] = b[0] \\ 1 + min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & lainnya. \end{cases}$$

Damerau-levenshtein distance adalah algoritma yang dikembangkan oleh Frederick J. Damerau berdasarkan algoritma *levenshtein distance* yang dikembangkan oleh Vladimir Levenshtein. Definisi *Damerau-levenshtein distance* menurut Nur Hamidah, *et al.* (2020) dalam jurnal ilmiah berjudul *Spelling Checker using Algorithm Damerau Levenshtein Distance and Cosine Similarity* adalah algoritma yang digunakan untuk mencari operasi minimal untuk melakukan penyuntingan dua *string*, agar menjadi sama. Dalam algoritma *damerau-levenshtein distance* terdapat 4 operasi yaitu *insertion, deletion, substitution,* dan *transposition*. Dari hal tersebut, algoritma *damerau-levenshtein distance* dapat dirumuskan sebagai berikut:

$$DL_{(a,b)}[i,j] = min \begin{cases} 0 & jika\ i = j = 0 \\ DL_{(a,b)}(i-1, j) + 1 & jika\ i > 0 \\ DL_{(a,b)}(i,j-1) + 1 & jika\ j > 0 \\ DL_{(a,b)}(i-1, j-1) + 1_{(ai\ !=bj)} & jika\ i,j > 0 \\ DL_{(a,b)}(i-2, j-2) + 1 & jika\ i,j > 1\ dan\ a[i] = b[j-1]\ dan\ a[i-1] = b[j] \end{cases} \quad (2)$$

Dengan a dan b merupakan dua buah *string* yang akan dibandingkan. Masing-masing kasus di atas berkorespondensi sebagai berikut:

DL(a,b) (i − 1,j) + 1 , deletions

DL(a,b) (i,j − 1) + 1, insertions

DL(a,b) (i − 1,j − 1) + 1(ai ≠bj) , subtitutions

DL(a,b) (i − 2,j − 2) + 1, transpositions

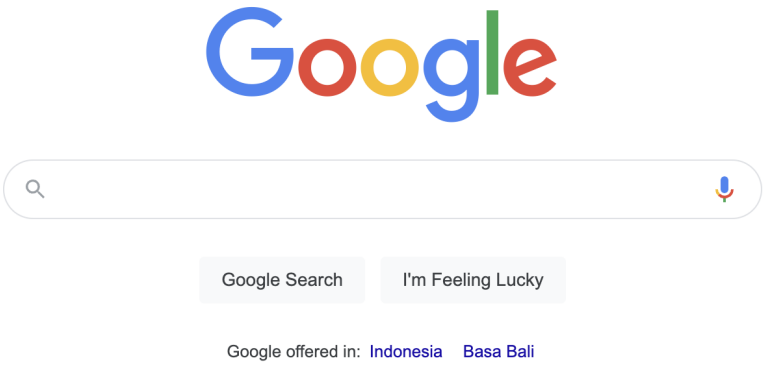
Hasil dari algoritma *damerau-levenshtein distance* yaitu *minimum edit distance*. *Minimum edit distance* adalah representasi jumlah *edit* yang diperlukan untuk mentransformasi *string*. Pada sistem pencarian berbasis *fuzzy search* kata kunci yang dimasukkan oleh pengguna nantinya akan diproses oleh sistem dan akan menghasilkan pencarian yang berelasi dengan kata kunci berdasarkan operasi *minimum edit distance*.

2.4. *Search Engine*

Definisi *search engine* menurut Jerri L. Ledford (2016) adalah bagian dari sistem perangkat lunak yang menggunakan suatu algoritma mencari dan menemukan informasi yang terdapat pada halaman situs. Informasi yang diindeks adalah frasa atau kata kunci yang merupakan indikator terkait isi dari halaman situs. Cara kerja *search engine* pada antarmuka situs ketika pengguna memasukkan kata kunci saat mencoba mencari informasi, kemudian pengguna mengklik *search button*, algoritma sistem kemudian memproses informasi yang ada di basis data dan menyajikan kata kunci yang relevan yang dicari oleh pengguna.

Search engine memiliki beberapa anatomi, yaitu:

1. *Query interface* adalah antarmuka dari *search engine*. Ketika pengguna mengakses *search engine* maka fitur ini yang akan terlihat pertama kali. Acuan dari kualitas antarmuka *search engine* yaitu fleksibilitas pengguna dalam mengaksesnya di berbagai gawai, konten halaman, struktur halaman, nama domain, dan *meta tags*.



Gambar 2.1 Google *Search Engine*

2. *Search algorithms* adalah pondasi dari keseluruhan komponen sistem pencarian. Keseluruhan alur kerja dari *search engine* ditopang oleh algoritma pencarian yang berhubungan langsung dengan data yang dicari oleh pengguna. Fungsi dasar dari algoritma pencarian yaitu memproses kata kunci, mengevaluasi kata kunci yang mungkin relevan, dan memberikan hasil berdasarkan kata kunci yang dimasukkan. Algoritma pencarian secara umum terbagi menjadi tiga kategori, yaitu:
    a. *On-page algorithms* adalah algoritma pencarian yang mengukur berdasarkan faktor konten penyusun dari suatu halaman. Kata kunci merupakan hal yang sangat berperan pada proses algoritma dari pencarian ini karena hasil pencarian yang diberikan bergantung pada banyaknya frasa atau kata kunci yang berhubungan dengan situs terkait. Algoritma ini melihat keterkaitan hasil pencarian berdasarkan kedekatan dari kata kunci.
    b. *Whole-site algorithms* adalah algoritma pencarian yang memberikan hasil berdasarkan hubungan keselarasan antar halaman pada suatu situs.

c. *Off-site algorithms* adalah algoritma pencarian yang memberikan hasil berdasarkan keselarasan isi konten situs dengan *external link* yang terdapat pada situs tersebut.

2.5. Elasticsearch

Gambar 2.2 Elasticsearch

Elasticsearch adalah sistem pencarian terbuka, gratis, terdistribusi, dan *analytics engine* untuk seluruh tipe data, yaitu: tekstual, numerikal, geospasial, data terstruktur, dan data tidak terstruktur. Elasticsearch dibangun di atas sistem *apache lucene* dan pertama kali dikeluarkan pada tahun 2010. Cara kerja elasticsearch yaitu data mentah diindeks ke dalam basis data elasticsearch dari sumber sistem utama. Data yang akan diindeks ke dalam elasticsearch sebelumnya penguraian dan normalisasi. Setelah data diindeks, pengguna dapat menjalankan query yang kompleks dan dapat melakukan agregasi serta pengelompokan data.

Elasticsearch indeks adalah koleksi dari dokumen-dokumen yang saling terhubung satu sama lain. Elasticsearch menyimpan data dalam bentuk dokumen JSON. Tiap dokumen berkorelasi pada sekumpulan *property keys* serta value dari properti tersebut. Elasticsearch

menggunakan struktur data yang disebut inverted index, yaitu struktur data yang dirancang untuk melakukan pencarian dengan cepat. Sebuah *inverted index* terdiri dari sekumpulan kata yang unik yang muncul pada dokumen dan mengidentifikasi seluruh dokumen tempat setiap kata yang muncul. Berdasarkan penelitian yang dilaksanakan oleh Rahmat Firman (2021) berjudul *Perancangan Sistem Pencarian Berbasis Full-Text Search Engine dengan Elasticsearch Studi Kasus: Web Digital Library Fakultas Teknik Universitas Hasanuddin*, terdapat beberapa fitur dalam sistem elasticsearch, yaitu:

1. Indeks

   Dalam elasticsearch, indeks dapat dianggap sebagai basis data. Indeks diatur dengan mengirim *query* ke elasticsearch melalui akses API. Pada indeks ini akan diatur jumlah *shard* yang akan digunakan dan jumlah *replica* yang akan dibuat.

2. *Mapping*

   Mapping adalah bagian dimana pengguna mengatur bagaimana data akan disimpan dalam elasticsearch. Di bagian inilah kolom-kolom dokumen diatur beserta tipe datanya. Pengguna juga dapat mengatur bagaimana teks-teks yang akan diindeks ditokenisasi, misalnya *tags* HTML akan dibersihkan sebelum diinputkan ke dalam indeks. Dalam satu indeks pengguna dapat menambahkan beberapa *mapping*. *Mapping* dapat dianggap sebagai tabel elasticsearch.

3. Dokumen

   Dokumen dalam elasticsearch merupakan *entity*, terdiri dari beberapa kolom dan setiap kolomnya memiliki penamaan dan dapat menyimpan satu atau lebih nilai dalam bentuk *array*. Dokumen dalam elasticsearch akan mempunyai kolom yang berbeda dan dokumen yang

disimpan akan memiliki format JSON. Dokumen dapat dianggap sebagai kolom dalam elasticsearch.

4. *Type*

Setiap dokumen yang ada dalam elasticsearch memiliki *type*. *Type* dalam dokumen dikelompokkan ke dalam beberapa grup logika untuk mempermudah proses pencarian.

5. *Node*

Node adalah *instance* dari sebuah elasticsearch. Elasticsearch memerlukan minimal satu *node* untuk dapat bekerja. Jika lebih dari satu *node*, maka penamaan *node* biasanya menggunakan urutan angka. *Node-node* ini akan saling berbagi data dan beban menyelesaikan suatu tugas yang diberikan. Satu *node* akan bertindak sebagai *node master* yang akan mengatur *node* yang lain seperti penambahan atau penghapusan *node*.

6. *Cluster*

Kumpulan dari beberapa *node* dapat disebut *cluster*. Konsep penggunaan *cluster* adalah untuk mengurangi beban dari sistem dengan membagi tugas-tugas menjadi beberapa bagian. Kelebihan dari sistem *cluster* adalah ketika sistem mengalami kegagalan pada *node* tertentu maka sistem akan tetap beroperasi dengan mengalihkan kinerja *node* yang gagal ke *node* yang masih aktif, sementara *node* yang gagal akan melakukan *restart* agar dapat beroperasi lagi.

7. *Shard*

Data yang ada dalam elasticsearch akan dibagi menjadi beberapa bagian atau *chunks* kemudian disimpan ke dalam *shard*.

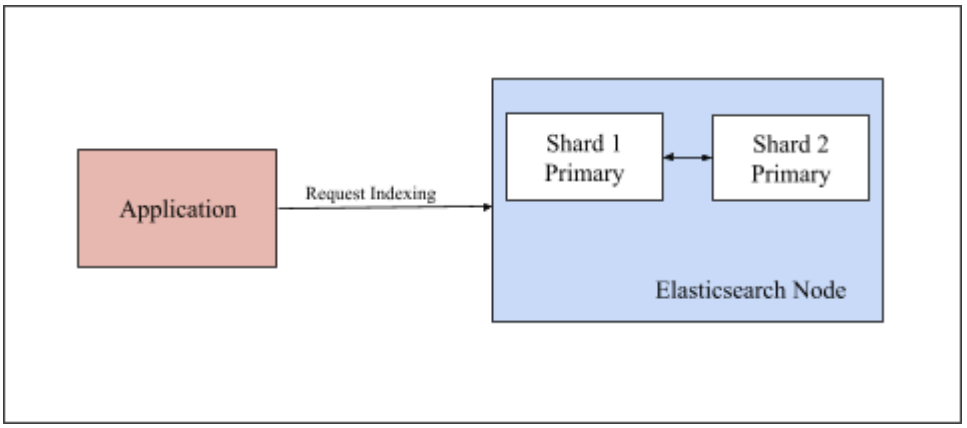
Shard merupakan partisi dari suatu indeks. Ketika indeks pertama kali dibuat, pengguna dapat mengatur jumlah *shard* yang akan digunakan. Ketika pengguna memasukkan data ke dalam elasticsearch *(indexing)*, maka elasticsearch akan secara otomatis melakukan *sharding* ke dalam *shard*. Pengguna tidak dapat mengetahui data yang diambil dari elasticsearch berasal dari *shard* mana.

2.5.1. *Indexing* elasticsearch

Menurut Rafał Kuć, *et al.* dalam bukunya yang berjudul *Mastering Elasticsearch*, cara untuk melakukan proses *indexing* yaitu dengan menggunakan *index* API yang telah disediakan elasticsearch. Dimana dokumen yang akan di indeks dikirim menggunakan *curl tool*, berikut adalah contoh perintah untuk melakukan *indexing*:

curl -XPUT http://localhost:9200/blog/article/1 -d '{"title": "New version of Elastic Search released!", "content": "...", "tags": ["announce", "elasticsearch", "release"] }'

Berikut adalah alur proses *indexing* pada sistem pencarian elasticsearch:



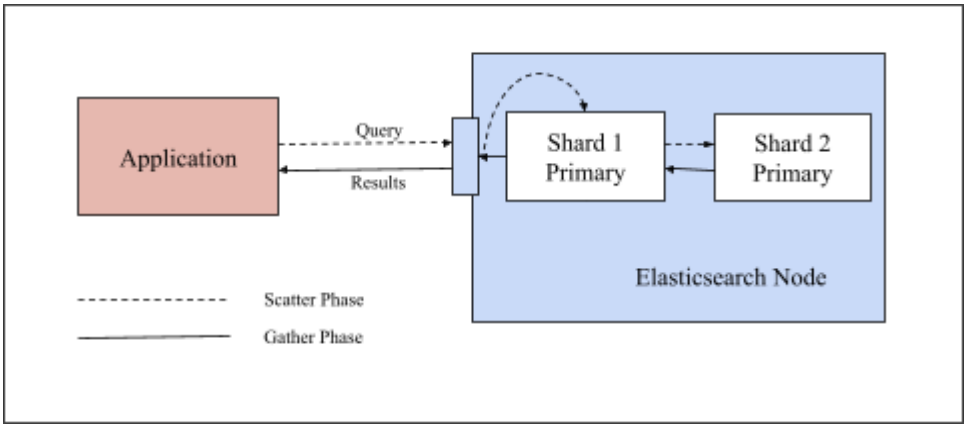
Gambar 2.3 Proses *Indexing Single Node*

(Sumber: *Mastering Elasticsearch*, 2013)

Proses *indexing* terjadi saat pengguna mengirimkan *request* untuk melakukan pemasukan data ke dalam sistem pencarian *elasticsearch*. Sebelum melakukan *indexing* data, nama *node*, *cluster*, dan *replica* telah dilakukan konfigurasi terlebih dahulu untuk menghindari tumpah tindih saat proses *indexing*. Data yang akan diindeks dilakukan pemisahan agar dapat disimpan ke dalam *shard* yang sesuai secara otomatis. Pada Gambar 2.3, hanya dilakukan instansiasi 1 node saja maka dapat dipastikan request akan dikirim ke *node master*. Pada kasus yang khusus, terdapat lebih dari satu instansiasi *node* jika terjadi kesalahan indeks aplikasi mengirimkan *request* ke *node* yang memiliki *shard replica*, maka data yang diindeks akan diteruskan ke *node* yang memiliki *shard primer*.

2.5.2. Querying Data

Menurut Rafał Kuć, *et al.* dalam bukunya yang berjudul *Mastering Elasticsearch*, *query* API dalam sistem pencarian elasticsearch menggunakan *dsl query* (berbasiskan JSON untuk mendefinisikan *query* yang kompleks). Berikut adalah alur proses yang *query* dalam elasticsearch:



Gambar 2.4 Proses *Query Single Node*

(Sumber: *Mastering Elasticsearch*, 2013)

Fase scatter terjadi saat pengguna mengirimkan *request* ke dalam sistem yang akan dikirimkan ke seluruh *shard* dalam *node*. *Shard* tersebut akan mengeluarkan data-data yang relevan berdasarkan *query dsl* yang dikirim melalui request. Setelah fase *scatter* selesai, maka dilanjutkan pada fase *gather*. Pada fase ini, terjadi proses penggabungan, penyortiran, dan pengolahan data yang telah dikeluarkan untuk dikirim kembali dalam bentuk response pada aplikasi utama.

Elasticsearch merupakan sistem pencarian yang sangat fleksibel karena terdapat banyak fitur yang dapat diimplementasikan ke dalam proses pencarian salah satunya adalah pencarian berbasis *fuzzy search*. Berdasarkan artikel internet yang ditulis oleh Andrew Cholakian dalam situs resmi elasticsearch (https://www.elastic.co/blog/found-fuzzy-search) pencarian berbasis *fuzzy search* dapat diimplementasikan pada sistem pencarian elasticsearch dengan menambahkan properti *fuzziness* dengan *value* berupa jumlah maksimum *edit distance*. Jumlah maksimum *edit distance* dapat mempengaruhi kecepatan proses pencarian karena data hasil pencarian yang diambil akan semakin banyak dan kemungkinan relevansi dari hasil pencarian akan semakin kecil karena prediksi perbaikan kata kunci yang salah.

2.6. Typesense

Gambar 2.5 Typesense

Typesense adalah sistem pencarian *open-source* yang toleran terhadap kata kunci *typo* dan mengklaim memiliki kecepatan proses pencarian berlatensi rendah diantara 50 ms. Beberapa fitur yang terdapat pada mesin pencari typesense, yaitu :

1. Node

   Node adalah *instance* dari typesense. Ketika menginisialisasi *node* dari typesense diperlukan --*api-key* untuk menjalankannya, tujuan dari --*api-key* tersebut untuk mengamankan *instance* typesense agar *endpoint* typesense tidak dapat diakses oleh semua orang. Secara umum, terdapat dua --*api-key* bagi pengguna, yaitu *search api key* dan *admin key*. *Search api key* merupakan *key* yang digunakan oleh pengguna umum untuk melakukan proses pencarian dengan mengakses *endpoint* pencarian saja. *Admin key* merupakan *key* yang disediakan bagi administrator untuk mengakses *endpoint*, membuat dokumen, *collection*, dan melakukan *indexing* data pencarian pada *instance* typesense.

2. Dokumen

Dokumen dalam typesense layaknya adalah basis data yang terdapat dalam basis data relasional yang bertindak menampung seluruh data yang diindeks pada suatu *instance* typesense. Di dalam dokumen typesense terdapat kumpulan *collection*.

3. *Collection*

Collection dalam typesense layaknya adalah tabel yang terdapat dalam basis data relasional, bertindak menyimpan seluruh entitas yang saling berkaitan ke dalam suatu *collection*. Sebelum melakukan *indexing data* pada suatu *collection* perlu dilakukan pembuatan *collection schema* yang digunakan *collection* nantinya untuk mengenali data dan tipe data yang diindeks ke dokumen tersebut.

2.6.1. *Indexing* Typesense

Berdasarkan dokumentasi resmi typesense, proses *indexing* typesense dapat dilakukan dengan menggunakan *index* API yang disediakan typesense. Dokumen yang akan di indeks dikirim menggunakan perintah *curl*. Berikut adalah contoh *indexing* pada sistem pencarian typesense:

```
curl “http://localhost:8108/collections/companies/documents” -H
“Content-Type: application/json” \
-H “X-TYPESENSE-API-KEY: ${TYPESENSE_API_KEY}” \
-d ‘{
        “id”: “124”,
        “company_name”: “Stark Industries”,
        “num_employees”: 5215,
        “country”: “USA”
}’
```

Perintah *curl* yang telah dijalankan akan mengirimkan *request http* ke sistem pencarian typesense untuk selanjutnya dimasukkan ke dalam *collection*.

2.6.2. *Querying Data*

Berdasarkan dokumentasi resmi typesense, proses pencarian typesense menggunakan *query* API yang disediakan oleh typesense. Proses *query data* dapat dilakukan dengan mengirimkan *http request* ke *query* API melalui *curl command*, kemudian data akan diambil berdasarkan *collection* yang dipilih dan kriteria yang dimasukkan, selanjutnya memberikan *response* untuk dikirimkan kembali ke aplikasi utama.

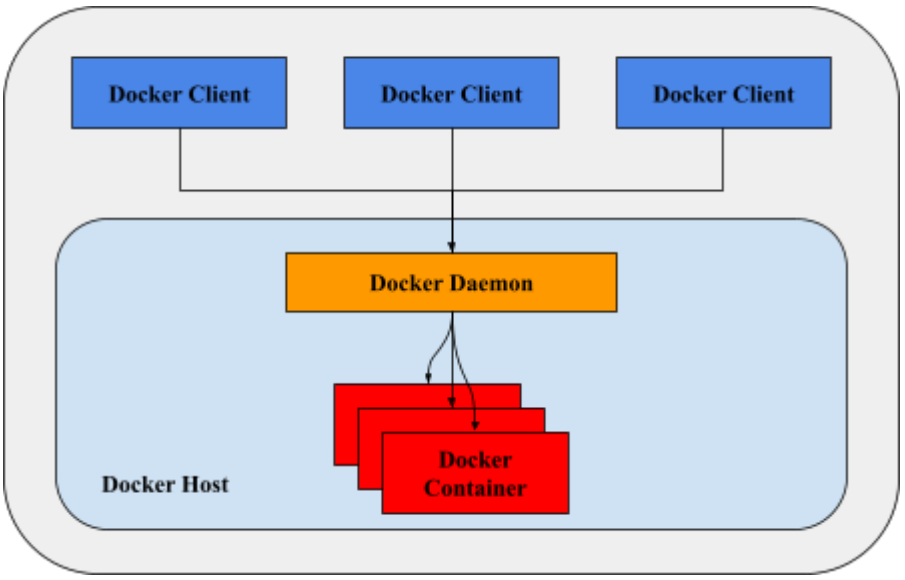
Secara bawaan typesense merupakan sistem pencarian yang berbasis *fuzzy search* yang dapat melakukan koreksi / *edit distance* sebanyak 2 huruf dari kata kunci yang dimasukkan dan memberikan hasil yang mungkin relevan dengan kata kunci yang dimasukkan. Maksimum *edit distance* ini dapat disesuaikan sebanyak yang pengguna inginkan namun semakin banyak jumlah *edit distance* maka kecepatan pemberian *response data* akan semakin berkurang karena proses kalkulasi dan jumlah dokumen yang diberikan akan semakin banyak, serta dapat kemungkinan untuk memberikan hasil yang kurang relevan akan semakin besar.

2.7. Docker

Docker adalah *open source platform* yang dapat menjalankan aplikasi dan membuat proses pengembangan dan distribusi menjadi lebih mudah. Aplikasi yang dibangun dalam docker mengemas semua *dependencies* pendukung aplikasi di dalam sebuah *container*. *Container*

berjalan dalam sebuah lingkungan yang terisolasi di atas sistem operasi. Terdapat fasilitas yang disediakan oleh docker, yaitu untuk membuat dan melakukan kontrol dengan *container*. Dengan melakukan virtualisasi aplikasi, pengembang dengan mudah menjalankan aplikasi di mana saja tanpa perlu melakukan banyak konfigurasi sistem. Terlebih docker dapat dengan mudah di *deploy* ke lingkungan *cloud computing*. Terdapat 4 komponen internal, yaitu:

1. *Docker Client and Server*



Gambar 2.6 Arsitektur Docker

Docker dapat dijelaskan sebagai aplikasi yang berbasiskan klien dan server seperti pada Gambar 2.4. Docker server mendapatkan *request* dari docker *client* kemudian memprosesnya. Docker server dan docker klien dapat dijalankan pada server yang sama atau docker klien dapat dihubungkan dengan sebuah *remote* server yang berjalan pada server yang berbeda.

2. *Docker Images*

Docker images adalah *tools* atau kumpulan *file* yang menunjang sebuah aplikasi untuk membangun sebuah *container*.

3. *Docker Registries*

   Docker images disimpan di dalam docker registries. Bertugas menyimpan *source code repository* dimana pengguna dapat melakukan *pull* dan *push image* dari sebuah sumber tunggal. Terdapat dua tipe *registry*, yaitu *public* dan *private*. *Docker hub* disebut sebagai *public registry*, yaitu tempat bagi pengguna untuk melakukan *pull* pada *image* yang tersedia dan melakukan *push* terhadap image yang dibuat. *Images* dapat didistribusikan baik secara publik dan privat melalui fitur *docker hub*.

4. *Docker Containers*

   Docker containers adalah wadah untuk mengemas dan menjalankan aplikasi. Wadah ini mencakup kode, *runtime*, *system tools*, dan pengaturan. *Container* hanya bisa mengakses *resource* yang telah ditentukan dalam Docker *image*.

Docker memiliki banyak kelebihan dibandingkan *virtual machine* yaitu:

1. Docker dapat menjalankan sebuah sistem dalam waktu yang sangat cepat dibandingkan *virtual machine*.
2. *Container* lebih ringan dibandingkan *virtual machine* karena *container* hanya menjalankan library dan aplikasi yang dijalankan saja. Sedangkan *virtual machine* harus menjalankan seluruh komponen seperti perangkat keras, kernel, sistem operasi, dan aplikasi.
3. Aplikasi yang dijalankan dengan docker sangat mudah dipindahkan dari satu server ke server lain karena hanya perlu menjalankan satu *script* konfigurasi saja.
4. Alokasi *resource hard disk* dan RAM pada *container* dibagi oleh *host server*, *container* akan mengambil alokasi *resource* yang ada di *hardware* sesuai dengan yang *container* butuhkan.

2.8. Grails

Gambar 2.7 *Grails*

Grails merupakan kerangka kerja *website* yang dikembangkan oleh Graeme Rocher menggunakan bahasa pemrograman groovy. Grails didukung oleh beberapa teknologi yang sangat populer yaitu:

1. *Hibernate* merupakan standar *object relational mapping* dalam pemrograman java untuk melakukan pemetaan basis data relasional menjadi sebuah *object oriented programming (oop)* yang dimana *hibernate* menyediakan fungsi-fungsi untuk melakukan operasi-operasi data seperti *insert*, *update*, *delete*, dan *select*.
2. *Spring* adalah kerangka kerja *open source* untuk *website* menggunakan bahasa pemrograman java. Dikembangkan dengan menggunakan prinsip pemrograman *Inversion of Control (IoC)* untuk meningkatkan modularitas dari program yang dibuat.
3. *Sitemesh* adalah *layout-rendering framework* yang digunakan dalam proses menampilkan tampilan situs pada pengguna.

Keuntungan menggunakan grails karena telah menghimpun kerangka-kerangka kerja *hibernate*, *spring*, dan *sitemesh* menjadi satu *layer* abstraksi menggunakan bahasa pemrograman *groovy*. Pengembang bahkan tidak menyadari bahwa ia membangun sebuah aplikasi berbasis

spring dan *hibernate* karena tidak perlu bersentuhan langsung dengan kerangka kerja tersebut.

2.9. Metode Analisis

Confusion matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya, *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya (Eko Prasetyo, 2012).

Terdapat 4 nilai yang dihasilkan di dalam tabel *confusion matrix*, yaitu *true positive* (TP), *false positive* (FP), *false negative* (FN), dan *true negative* (TN). Berikut adalah ilustrasi tabel *confusion matrix*:

Tabel 2.1 Ilustrasi Tabel *Confusion Matrix*

		Nilai Aktual	
		Positive	**Negative**
Nilai Prediksi	**Positive**	TP	FP
	Negative	FN	TN

Keterangan:

1. *True Positive* (TP) : Jumlah data yang bernilai positif dan diprediksi benar sebagai positif.
2. *False Positive* (FP) : Jumlah data yang bernilai negatif tetapi diprediksi sebagai positif.
3. *False Negative* (FN) : Jumlah data yang bernilai positif tetapi diprediksi sebagai negatif.
4. *True Negative* (TN) : Jumlah data yang bernilai negatif dan diprediksi benar sebagai negatif.

Nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) dapat digunakan untuk menghitung *precision*, *recall*, dan *F-Measure*.

Precision merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif, *precision* dapat dirumuskan dengan persamaan:

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

Recall merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif, *recall* dapat dirumuskan dengan persamaan:

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

F-Measure merupakan acuan yang baik digunakan untuk mengukur kinerja dari suatu algoritma pemrograman dibandingkan akurasi karena *F-Measure* merupakan *harmonic mean* dari *recall* dan *precision* sehingga, memberikan hasil yang lebih akurat karena memperhatikan *False Negative* (FN) dan *False Positive* (FP) dari data yang diuji. *F-Measure* dapat dirumuskan dengan persamaan:

$$F\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (5)$$