BAB II
TINJAUAN PUSTAKA

2.1 Rantai Pasok

Rantai pasok atau *supply chain* merupakan segala kegiatan atau usaha yang melibatkan setiap pihak yang terlibat dalam proses produksi barang atau jasa, mulai dari produsen hingga ke konsumen [11].

Gambar 2. 1 Ilustrasi Rantai Pasok [19]

Rantai pasok berbentuk jaringan saling terkait yang mencakup seluruh rangkaian aktivitas mulai dari pengadaan bahan baku, produksi, penyimpanan, distribusi, hingga penjualan. Tujuan dari rantai pasok adalah untuk memastikan bahwa produk dapat disediakan secara efektif dan efisien serta sampai tepat waktu kepada konsumen dengan biaya yang optimal.

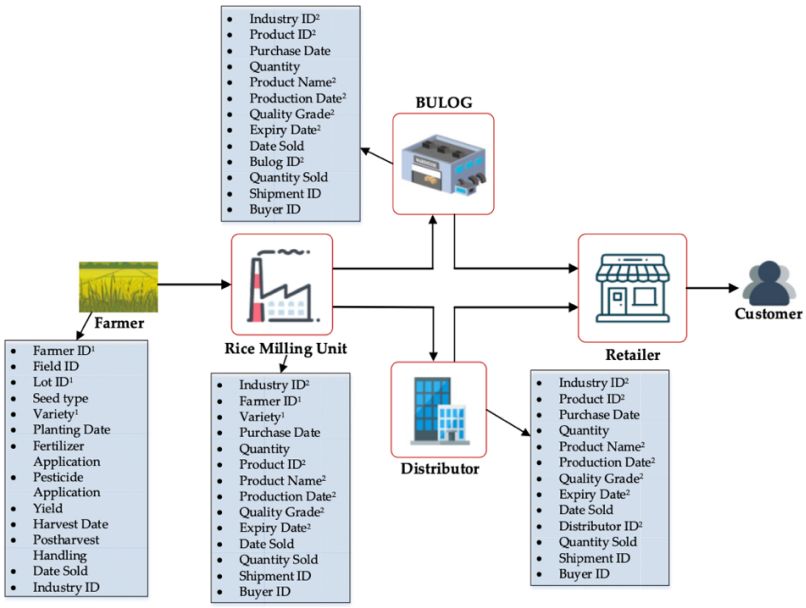
2.1.1 Manajemen Rantai Pasok

Manajemen rantai pasok atau *supply chain management* merupakan sistem yang melibatkan proses produksi, pengiriman, penyimpanan, distribusi, dan penjualan produk untuk memenuhi permintaan termasuk segala proses dan kegiatan yang terlibat di dalamnya [13]. Kegiatan atau usaha yang terjadi di dalam suatu rantai pasok harus dikelola melalui manajemen rantai pasok yang baik. Hal itu karena permasalahan yang muncul dalam ekosistem rantai pasok adalah tidak adanya data dan informasi yang dapat dialirkan dalam lingkungan rantai pasok sehingga sering menyebabkan terjadinya kesalahan pendataan produk, ketidaktransparanan pengelolaan produk, dan kualitas produk yang tidak sesuai harapan [23].

2.2 Ketertelusuran

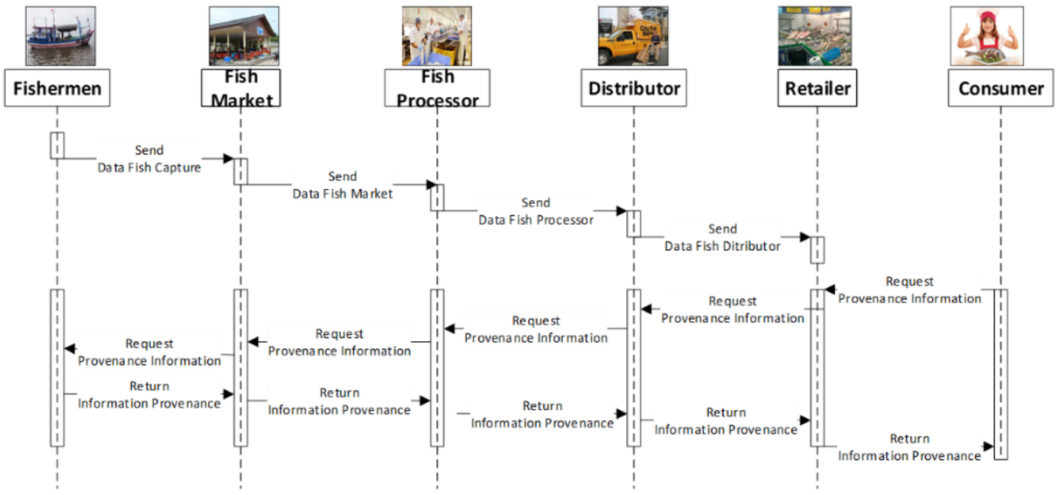
Ketertelusuran atau *traceability* merupakan suatu kemampuan untuk menyajikan informasi terkait riwayat dan perpindahan produk yang terjadi melalui setiap proses produksi dan distribusinya [14]. Kemampuan ini bermanfaat untuk proses pelacakan atau penelusuran pergerakan sebuah produk pada suatu rantai pasok. Sistem ketertelusuran ini mampu mendukung manajemen rantai pasok karena dapat menghubungkan area produksi ke dalam rantai pasok dan mengidentifikasi aktor yang terlibat di dalamnya seperti produsen, pedagang, distributor, dan konsumen sehingga lebih menghemat biaya dan terkoordinasi [24].

Namun, mengembangkan sistem ketertelusuran dalam rantai pasok membutuhkan integritas dan koordinasi semua pihak yang terlibat di dalamnya. Proses penelusuran suatu produk akan memakan waktu lama jika ekosistem rantai pasok masih berdiri sendiri dan belum terintegrasi [23]. Pertukaran informasi menjadi dasar integrasi dan koordinasi antar pihak karena dapat meningkatkan transparansi dalam proses manajemen risiko serta memengaruhi mutu, kualitas, dan keamanan produk [25]. Gambar 2. 2 menunjukkan contoh informasi yang harus disimpan dan dibagikan di dalam suatu rantai pasok.



Gambar 2. 2 Informasi pada Sistem Ketertelusuran [25]

Ketertelusuran memungkinkan setiap pihak yang terlibat di dalam rantai pasok untuk memverifikasi keaslian dan kualitas produk. Informasi tersebut mencakup sumber asal produk hingga berada pada konsumen akhir termasuk seluruh proses dalam rantai pasok seperti produksi, pengolahan, dan distribusi. Gambar 2. 3 menunjukkan aliran data dan permintaan informasi pada sistem ketertelusuran.



Gambar 2. 3 Alur Ketertelusuran pada Rantai Pasok [19]

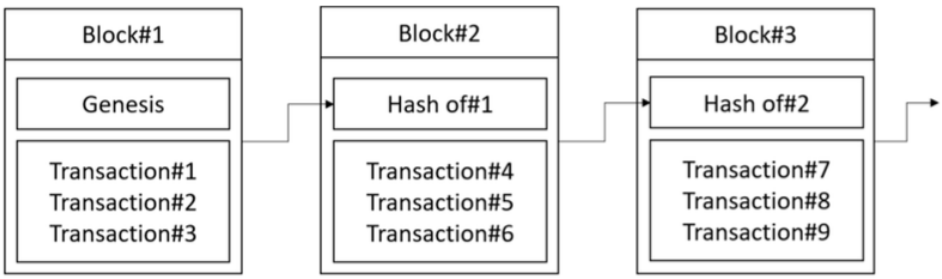
Mekanisme ketertelusuran seperti yang ditunjukkan pada Gambar 2. 3 dapat dianggap sebagai aliran proses dari hulu ke hilir, sedangkan permintaan informasi mengalir dari hilir ke hulu [19]. Dengan demikian, ketertelusuran bermanfaat untuk meningkatkan aspek visibilitas dan transparansi pada suatu rantai pasok karena dapat menyediakan informasi aktivitas suatu produk sehingga dapat mendukung sistem manajemen rantai pasok menjadi lebih baik.

2.3 Blockchain

Blockchain merupakan basis data yang berisi sejumlah catatan yang terdistribusi atau buku besar yang memuat segala transaksi atau peristiwa digital yang terjadi untuk kemudian dibagikan kepada setiap pihak yang berpartisipasi di dalamnya [15]. Teknologi *blockchain* mirip seperti *spreadsheet* yang berisi sejumlah transaksi, lalu disalin sebanyak jumlah *node* atau peserta yang ada pada jaringan *blockchain* [21]. Berdasarkan hal tersebut, setiap peserta dapat mencatat

transaksi secara mandiri tanpa melalui perantara. Namun, dalam proses pengesahannya, memerlukan persetujuan mayoritas peserta atau biasa disebut sebagai konsensus.

Blockchain menyimpan data secara permanen di dalam *record* atau *block* data yang kemudian dikomunikasikan secara *peer-to-peer* ke setiap *node* yang ada di dalam jaringan [16] sehingga transaksi yang telah tersimpan tidak dapat diubah kembali. Hal yang dapat dilakukan untuk mengubah data yang telah tersimpan yaitu dengan membuat blok baru sebagai perubahan atas blok yang dipilih yang memerlukan persetujuan mayoritas peserta lain yang ada di dalam jaringan *blockchain* [26].



Gambar 2. 4 Ilustrasi *Blockchain*

Terdapat tiga jenis *blockchain* yaitu *public*, *permissoned or private*, dan *federated or consortium* [27]. *Public blockchain* merupakan jenis jaringan *blockchain* yang mengizinkan siapapun untuk berpartisipasi di dalamnya tanpa dibatasi. Sebaliknya, dalam jaringan *blockchain* bertipe *permissoned or private*, hanya pengguna yang diberi izin yang dapat berpartipasi untuk dapat mengakses data tertentu. Sedangkan *blockchain* bertipe *federated or consortium* mengontrol secara ketat proses konsensus oleh sekumpulan *node* yang telah dipilih sebelumnya.

Adapun berikut merupakan tiga komponen utama pembentuk *blockchain* [28]:

1. *Block*

   Block atau blok merupakan informasi transaksi yang disimpan pada sebuah *ledger*. Informasi transaksi tersebut memiliki ukuran, periode, dan *trigger* yang dapat berbeda-beda sesuai dengan implementasi *blockchain*.

2. *Chain*

   Chain atau rantai merupakan media yang berfungsi untuk menyambungkan setiap blok yang berisi informasi transaksi. Rantai tersebut berupa *hash* yang terbentuk dari informasi transaksi pada blok sebelumnya. Dengan begitu, jika suatu blok diubah, maka akan mudah terdeteksi perubahannya sehingga hal inilah yang dapat meningkatkan aspek kepercayaan.

3. *Network*

   Network atau jaringan pada *blockchain* terdiri dari sejumlah peserta yang disebut sebagai *node*. Setiap *node* tersebut menyimpan catatan lengkap dari seluruh transaksi yang disimpan dalam suatu jaringan *blockchain*.

2.3.1 Konsensus

Konsensus merupakan proses yang berfungsi untuk memastikan keabsahan suatu transaksi yang akan dicatat melalui proses validasi transaksi yang dilakukan oleh setiap *node* di dalam jaringan [29]. Konsensus dilakukan agar semua *node* dapat mencapai kesepakatan tentang keadaan terbaru dari *ledger* atau *database* pada *blockchain*. Hal tersebut penting untuk dicapai pada suatu sistem terdistribusi [30]. Proses ini juga bertujuan untuk memastikan bahwa seluruh *node* memiliki salinan yang identik dan akurat.

Dalam konsensus *blockchain*, setiap transaksi yang diajukan dijaga oleh protokol atau algoritma konsensus tertentu di mana setiap *node* memverifikasi transaksi tersebut sebelum mengajukannya ke jaringan. Setelah sebuah blok

transaksi selesai diverifikasi, setiap *node* di jaringan *blockchain* harus mencapai kesepakatan tentang validitas blok tersebut sebelum memasukkannya ke dalam *blockchain*. Dengan demikian, penggunaan konsensus dapat menjamin integritas data dan menghindari kecurangan atau manipulasi data [31]. Adapun berikut merupakan empat jenis konsensus yang umum digunakan di dalam *blockchain*:

1. *Proof of Work*

   Proof of Work (PoW) merupakan algoritma konsensus di mana setiap validator berpartisipasi untuk memvalidasi transaksi pada jaringan *blockchain* dengan imbalan berupa *criptocurrency* [32] sehingga prosesnya disebut sebagai penambangan (*mining*). Validator atau penambang (*miner*) harus memecahkan suatu teka-teki matematis yang kompleks sebagai bukti telah memvalidasi blok transaksi baru. Oleh sebab itu, proses konsensus pada algoritma PoW ini memerlukan daya komputasi dan energi yang besar karena setiap validator berlomba untuk secepat mungkin menyelesaikan proses validasi. Algoritma PoW dipakai pada platform Bitcoin.

2. *Proof of Stake*

   Proof of Stake (PoS) merupakan algoritma konsensus di mana bukan daya komputasi yang menjadi faktor penentu dalam memvalidasi transaksi baru. Pemilihan validator didasarkan pada jumlah *stake* (*criptocurrency*) yang dimiliki oleh seorang validator [33]. Dalam algoritma PoS, validator mengunci sejumlah *criptocurrency* sebagai jaminan untuk menjadi validator transaksi. Validator kemudian dipilih secara acak untuk memvalidasi transaksi dan jika berhasil, akan diberi imbalan berupa *criptocurrency*. Platform *blockchain* yang menggunakan algoritma ini adalah Ethereum.

3. *Practical Byzantine Fault Tolerance*

   Practical Byzantine Fault Tolerance (PBFT) merupakan algoritma konsensus untuk memastikan keamanan dan konsistensi data. Algoritma ini cocok untuk

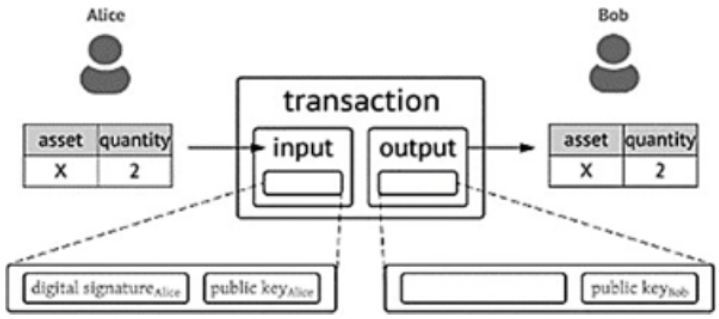
jaringan *blockchain* yang memprioritaskan integritas data [34], seperti pada platform *blockchain* keuangan.

4. *Delegated Proof of Stake*

   Delegated Proof of Stake (DPoS) merupakan algortima konsensus mirip seperti PoS. Namun yang membedakan ialah pada PoS pemilihan validator melalui demokrasi langsung sedangkan pada DPoS pemilihan validator melalui demokrasi perwakilan [35]. DPoS menggunakan pemilihan delegasi untuk memilih *node* yang berhak memvalidasi transaksi dan membuat blok baru di dalam jaringan. Setiap pemilik *stake* dalam jaringan dapat memilih delegasi yang akan mewakilinya dan memilih *node* yang memiliki reputasi baik dan dapat dipercaya sebagai validator. DPoS menawarkan kinerja jaringan yang cepat dan efisien dengan biaya transaksi yang rendah.

2.3.2 *Smart Contract*

Smart contract adalah program komputer atau *script* yang disimpan di dalam *blockchain* [36]. *Smart contract* dirancang untuk mengeksekusi kontrak secara otomatis tanpa melibatkan pihak ketiga.



Gambar 2. 5 Konsep *Smart Contract*

Isi dari *smart contract* merupakan serangkaian kondisi dan aturan yang dijalankan ketika dipicu oleh waktu atau peristiwa tertentu seperti terjadinya sebuah transaksi [37]. Contoh penggunaan *smart contract* seperti melakukan pembayaran otomatis setelah barang diterima atau asuransi otomatis yang dibayarkan ketika

terjadi sesuatu. *Smart contract* dapat membantu memastikan transparansi, keamanan, dan efisiensi dalam proses bisnis serta menghilangkan keterlibatan pihak ketiga.

2.4 OpenStreetMap

OpenStreetMap (OSM) merupakan proyek peta berbasis *web* dengan tujuan untuk membuat peta seluruh dunia secara gratis dan terbuka yang dibangun oleh sukarelawan dengan melakukan survei menggunakan GPS dan pengolahan citra dari satelit [38]. OpenStreetMap merupakan proyek kolaboratif berbasis komunitas yang bertujuan untuk membuat peta dunia yang bebas, terbuka, dan terperinci. OpenStreetMap memungkinkan siapa pun untuk ikut berkontribusi dengan informasi geografis, termasuk data tentang jalan, bangunan, perbatasan, dan fitur geografis lainnya. Hal ini menjadi alternatif terbuka dan gratis terhadap layanan peta komersial seperti Google Maps.

Gambar 2. 6 Logo OpenStreetMap

Peta yang dihasilkan oleh OpenStreetMap dibuat oleh komunitas sukarelawan yang berkontribusi dengan cara mengumpulkan, mengedit, dan memvalidasi data geografis. Komunitas sukarelawan ini menggunakan berbagai alat dan aplikasi pihak ketiga untuk mengumpulkan data lapangan. Kontribusi dari ribuan orang di seluruh dunia telah membentuk peta yang kaya dengan informasi dari berbagai jenis wilayah, baik perkotaan maupun pedesaan.

2.5 Basis Data

Basis data atau *database* merupakan kumpulan informasi yang terorganisir secara sistematis dan tersimpan di dalam sebuah komputer yang dapat diakses menggunakan suatu program komputer [39]. Basis data dapat mencakup berbagai jenis data seperti teks, angka, gambar, dan suara yang terkait dengan suatu topik atau tujuan tertentu. Data dalam basis data dapat dikelola menggunakan perangkat lunak basis data. Pengelolaan tersebut memudahkan pengguna untuk membuat, memperbarui, menghapus, dan mencari data secara efisien [40]. Manfaat basis data yaitu memudahkan pengaksesan dan pembaruan data, menghemat waktu dan biaya pengolahan data, serta menghindari duplikasi data dan kesalahan pengolahan data.

2.5.1 *Relational Database*

Relational database merupakan jenis basis data yang menggunakan model relasional di mana data tersimpan dalam tabel yang terdiri dari kolom sebagai atribut dan baris sebagai catatan [41]. Tabel mewakili entitas atau objek tertentu. Kolom di dalam tabel mewakili atribut atau sifat sedangkan baris data mewakili nilai data dari suatu entitas.

Hubungan antara tabel ditentukan oleh kunci (*key*) yang didefinisikan di dalam tabel seperti kunci utama (*primary key*) dan kunci asing (*foreign key*). *Primary key* digunakan sebagai penanda unik setiap baris di dalam tabel, sedangkan *foreign key* digunakan untuk menghubungkan tabel yang berbeda yang memiliki hubungan logis satu sama lain. Bahasa pemrograman SQL (*Structured Query Language*) digunakan dalam *relational database* untuk melakukan *query* data ke *database*.

2.5.2 *Non-Relational Database*

Non-relational database juga dikenal sebagai basis data NoSQL (Not only SQL). Jenis basis data ini tidak menggunakan skema tabular baris dan kolom melainkan menggunakan model penyimpanan yang dioptimalkan untuk kebutuhan

spesifik dari jenis data yang disimpan [42]. *Non-relational database* menggunakan struktur data yang berbeda, seperti dokumen, grafik, *key-value*, atau grup kolom.

Setiap struktur data dirancang untuk mengakomodasi jenis data yang berbeda dan memungkinkan fleksibilitas dalam mengelola data yang tidak terstruktur atau semi-struktur. Dengan demikian, *non-relational database* memiliki kemampuan untuk mengakomodasi data yang berubah-ubah dan tidak terstruktur serta kemampuannya untuk mengintegrasikan data dari berbagai sumber.

2.6 REST API

REST API yang merupakan singkatan dari Representational State Transfer Application Programming Interface merupakan layanan yang memungkinkan berbagai sistem untuk berkomunikasi dan berbagi data dengan cara yang sederhana [43]. Proses pertukaran data antara REST API dengan aplikasi atau layanan dilakukan melalui protokol HTTP (Hypertext Transfer Protocol). REST API menyediakan berbagai metode atau *endpoint* untuk berinteraksi dengan aplikasi atau layanan tersebut. Data yang diambil atau dikirimkan oleh REST API umumnya berbentuk representasi sederhana dari suatu objek atau sumber daya tertentu seperti JavaScript Object Notation (JSON).

2.7 Golang

Golang atau Go adalah bahasa pemrograman *open source* yang dikembangkan di Google oleh Robert Griesemer, Rob Pike, dan Ken Thompson pada tahun 2007 dan dirilis secara resmi pada tahun 2009 [44]. Golang dirancang dengan tujuan untuk memudahkan pengembangan aplikasi yang bersifat *scalable*, *reliable*, dan efisien terutama dalam lingkungan yang terdistribusi atau berbasis jaringan.

Gambar 2. 7 Logo Golang

Bahasa pemrograman ini mendukung pengaplikasian *concurrency* dengan cukup mudah, mendukung *parallel processing* pada waktu bersamaan, dan memiliki *garbage collection* dengan proses kompilasi yang sangat cepat [45] . Golang memiliki berbagai fitur dan *library* yang berfokus pada pengembangan aplikasi web, pemrosesan data, dan komputasi terdistribusi. Karena fleksibilitas dan kinerjanya yang tinggi, Golang digunakan oleh banyak perusahaan dan organisasi untuk mengembangkan berbagai jenis aplikasi dan layanan.

2.8 JavaScript

JavaScript merupakan bahasa pemrograman untuk *client-side* yang mudah dipelajari dan digunakan untuk memperkaya berbagai fitur pada *website* seperti menampilkan dan menghilangkan elemen serta memanggil kembali elemen tersebut menggunakan fungsi [46]. JavaScript digunakan untuk membuat interaksi pada halaman *web* menjadi lebih dinamis dan interaktif.

Gambar 2. 8 Logo JavaScript

Bahasa ini pertama kali dikembangkan oleh Brendan Eich dari Netscape pada tahun 1995. Sejak saat itu, JavaScript menjadi salah satu bahasa pemrograman yang

paling populer dan paling banyak digunakan di dunia. JavaScript biasanya digunakan untuk membuat efek visual seperti animasi, validasi form, dan efek *hover*. Selain itu, bahasa ini juga digunakan untuk membuat aplikasi *web* modern dengan menggunakan *framework* seperti React. JavaScript mendukung konsep pemrograman seperti *Object-Oriented Programming* (OOP) dan *Functional Programming* (FP) sehingga memungkinkan untuk membuat kode yang lebih modular, *scalable*, dan *reusable*.

2.9 Hyperledger Fabric

Hyperledger Fabric merupakan sebuah platform *blockchain open source* yang dikembangkan oleh The Linux Foundation dan dirancang untuk digunakan dalam lingkungan bisnis. Berbagai industri dan kasus penggunaan telah mengadopsi Hyperledger Fabric dengan lebih dari 400 prototipe, *proof-of-concept*, dan sistem pencatatan terdistribusi lainnya [47].

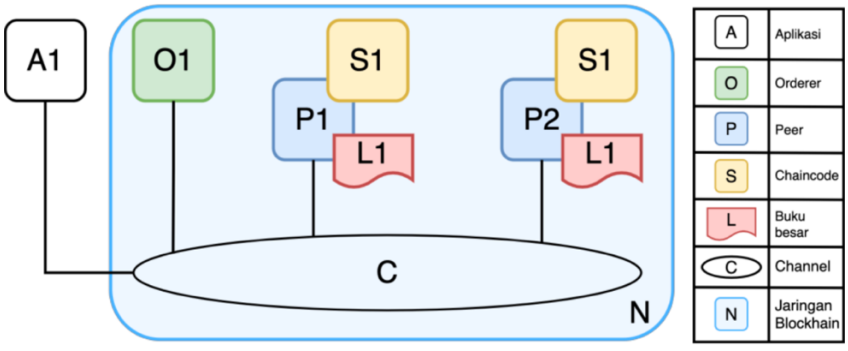
Gambar 2. 9 Logo Hyperledger Fabric

Setiap anggota dalam jaringan Hyperledger Fabric dapat saling berinteraksi melalui sebuah *channel* di mana data disimpan di dalam *ledger* yang terdiri dari *blockchain* dan *state database* untuk mengelola *state* terbaru dari suatu transaksi [48]. Hyperledger Fabric merupakan *framework blockchain* bertipe privat yang mana setiap anggotanya dapat melacak, bertukar, dan berinteraksi dengan aset digital melalui transaksi yang diatur oleh *chaincode* untuk memfasilitasi interaksi dengan *ledger* di dalam jaringan [17]. *Chaincode* merupakan sebutan untuk *smart contract* pada Hyperledger Fabric yang dapat diprogram menggunakan bahasa pemrograman Golang [29].

Hyperledger Fabric terdiri atas tiga komponen yakni aplikasi *client*, *peer node*, dan *ordering service node*. Ketiganya memerlukan identitas pengenal yang dibuat oleh *Certificate Authority* (CA), sedangkan hak akses dari masing-masing identitas tersebut diatur oleh *Membership Service Provider* (MSP) [49]. Berikut merupakan rincian masing-masing komponen tersebut [48]:

1. Aplikasi *client* bertugas mengirimkan proposal transaksi untuk dieksekusi, mengatur fase eksekusi, dan menyebarkan transaksi kepada *ordering service node* [47].
2. *Peer node* dapat memiliki dua peran. Peran *peer* pertama disebut *endorsing peer* yang bertugas untuk menyimulasikan transaksi melalui pengeksekusian *chaincode* dan mengesahkan hasilnya (*endorsing*). Sedangkan peran *peer* kedua disebut *committing peer* yang bertugas untuk memvalidasi transaksi dan mengelola *ledger* [17].
3. *Ordering service node* atau *orderer* bertugas untuk menentukan urutan transaksi di dalam blok yang akan disimpan ke dalam *ledger*. *Orderer* tidak mengetahui bagaimana keadaan aplikasi serta tidak berpartisipasi dalam eksekusi dan validasi transaksi.

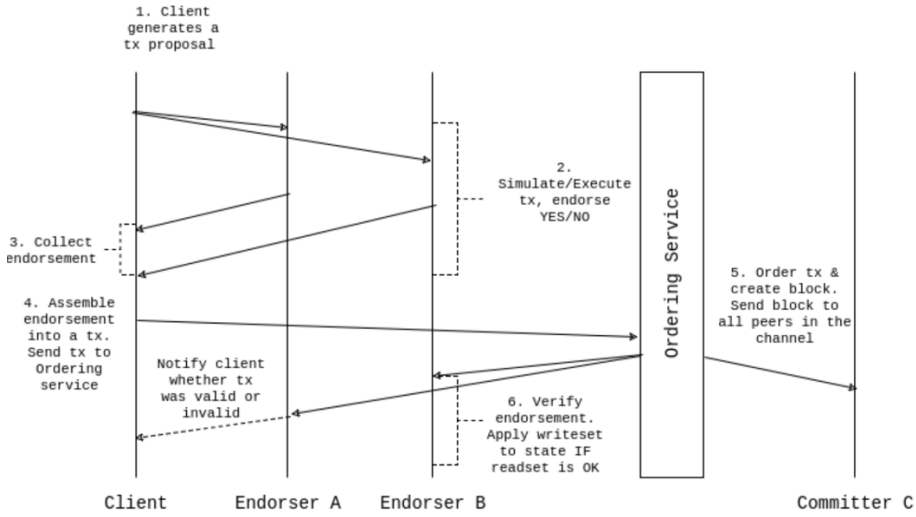
Berikut merupakan ilustrasi arsitektur Hyperledger Fabric lengkap dengan komponen pembentuknya yang ditunjukkan pada Gambar 2. 10.



Gambar 2. 10 Arsitektur Hyperledger Fabric [48]

Transaksi pada Hyperledger Fabric dimulai apabila terdapat dua atau lebih partisipan yang bergabung ke dalam sebuah *channel*. Setelah itu, partisipan tersebut

harus menyetujui dan mengimplementasikan *chaincode* pada *channel* tempatnya berada. Proses transaksi berikutnya ditunjukkan pada Gambar 2. 11 [17].



Gambar 2. 11 Alur Transaksi Hyper Ledger Fabric [17]

Berikut merupakan penjelasan dari Gambar 2. 11:

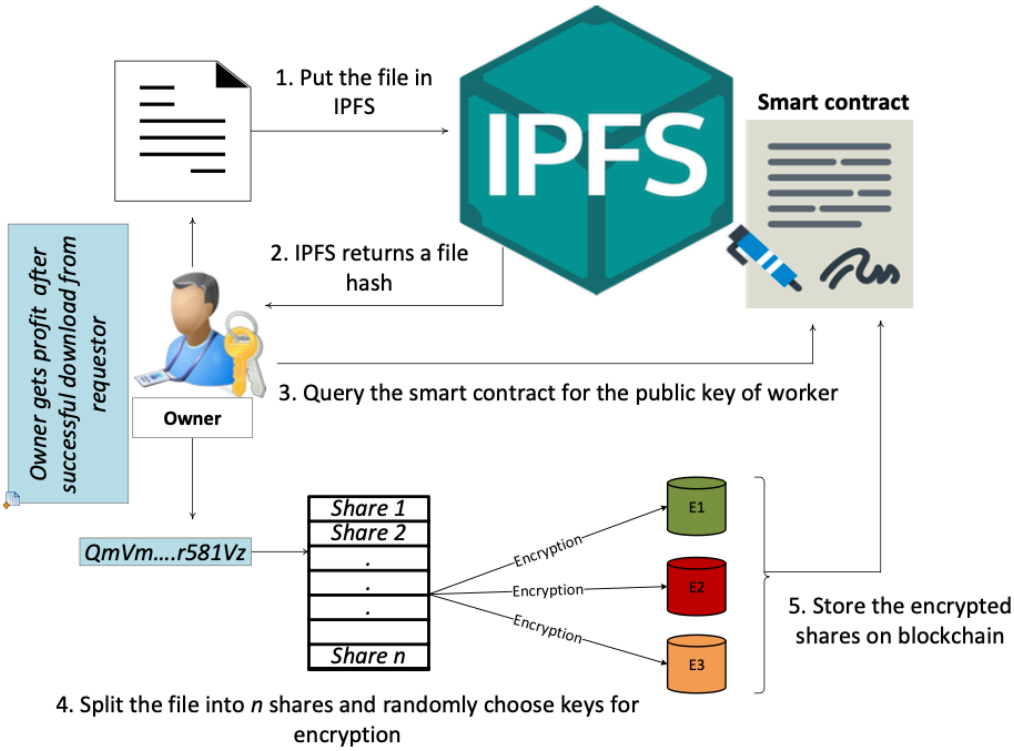
1. Seorang klien mengirimkan proposal transaksi kepada *endorsing peer*.
2. *Endorsing peer* akan melakukan proses verifikasi tanda tangan (*signature*) proposal dan menentukan apakah klien tersebut boleh melakukan operasi yang diminta atau tidak. Jika diperbolehkan, *endorsing peer* akan menggunakan proposal transaksi sebagai data masukan dan mengeksekusi transaksi yang diminta sehingga menghasilkan hasil transaksi.
3. Hasil transaksi beserta keputusan disahkan atau tidaknya dikirimkan kembali kepada klien.
4. Klien memastikan bahwa hasil transaksi konsisten dan telah ditandatangani, dilanjutkan dengan mengirimkan kembali transaksi beserta hasilnya kepada *orderer*.
5. *Orderer* akan mengurutkan transaksi secara FIFO (*First In First Out*) ke dalam suatu blok. Lalu blok tersebut dikirim ke semua *peer* dan dimasukkan ke dalam *ledger*.
6. Setiap transaksi di dalam blok ditandai dengan *valid* atau *invalid*. Hanya transaksi yang valid saja yang dimasukkan ke dalam basis data *state*.

2.10 IPFS

IPFS (InterPlanetary File System) merupakan suatu protokol atau sistem file terdistribusi berbasis jaringan *peer-to-peer* (P2P) yang menghubungkan setiap perangkat komputasi yang memiliki sistem file yang sama pada sebuah jaringan [50]. IPFS merupakan media penyimpanan file pada sistem komputasi secara terdesentralisasi dan terdistribusi. Oleh karena itu, IPFS dapat dijadikan sebagai solusi untuk meningkatkan keamanan dan kepercayaan pengguna terhadap keutuhan, ketersediaan, dan kerahasiaan informasi sehingga pengguna dapat berbagi dan memanfaatkan informasi dengan aman [51]. Kelebihan lain dari IPFS yaitu kemampuannya untuk mencegah serangan DDoS (*Distributed Denial of Service*) dan proses pengiriman konten terdistribusi di dalamnya akan menghemat *bandwidth* jaringan [52].

Gambar 2. 12 Logo IPFS

IPFS memberikan *hash* file dari file yang diunggah oleh pengguna ke dalam sistem IPFS, yang mana *hash* file tersebut dapat disimpan ke dalam jaringan *blockchain*. Setelah *hash file* dihasilkan, pasangan kunci publik dan kunci privat kemudian disimpan melalui *smart contract*. Adapun rincian alur kerja dari proses pengunggahan file pada sistem IPFS ditunjukkan pada Gambar 2. 13.



Gambar 2. 13 Alur Pengunggahan File pada IPFS [53]

Ketika *hash* dari IPFS diterima oleh penerima, kemudian dilakukan verifikasi pengirim yang terdapat pada *smart contract*. Pengirim tersebut bertanggung jawab dalam melakukan dekripsi file untuk penerima. Algoritma SSS juga digunakan dalam proses pembagian file *hash* ke sejumlah *share* IPFS. Setiap pembagian file *hash* tersebut disimpan ke dalam jaringan *blockchain* berikut informasi penting lainnya seperti otorisasi file penerima.

2.11 MySQL

MySQL merupakan sistem manajemen basis data yang termasuk ke dalam jenis *relational database* atau biasa disebut sebagai *Relational Database Management System* (RDBMS) sehingga menerapkan konsep tabel, kolom, dan baris serta menggunakan perintah berbasis *Structured Query Language* (SQL) [54].

Gambar 2. 14 Logo MySQL

MySQL bersifat *open source* dengan lisensi GNU General Public License (GPL) dan dapat dioperasikan pada berbagai sistem operasi. MySQL menyediakan basis data relasional yang memungkinkan pengguna untuk dapat mengakses, memasukkan, memperbarui, dan menghapus data. Selain itu, terdapat juga fitur-fitur seperti keamanan, transaksi, *indexing*, dan replikasi data untuk memastikan keamanan dan integritas data. MySQL mendukung integrasi dengan berbagai bahasa pemrograman populer sehingga memungkinkan pengguna untuk membuat aplikasi *web* maupun *desktop* dengan mudah.

2.12 CouchDB

CouchDB (Apache CouchDB) merupakan sistem manajemen basis data yang termasuk ke dalam jenis *non-relational database* atau NoSQL sehingga data yang disimpan menggunakan model data bebas skema dalam format dokumen berbasis JSON yang dapat menyederhanakan pengelolaan *record* di berbagai perangkat komputasi [55].

Gambar 2. 15 Logo CouchDB

CouchDB adalah sistem basis data *open source* yang dikembangkan oleh Apache Software Foundation. CouchDB menyediakan antarmuka RESTful API yang memungkinkan pengguna untuk mengakses dan memanipulasi data melalui protokol HTTP dengan menggunakan metode seperti GET, POST, PUT, dan DELETE. Selain itu, CouchDB juga memiliki fitur-fitur seperti replikasi data yang didesain untuk penggunaan yang *mobile*, sinkronisasi *offline*, dan kemampuan untuk menangani banyak pengguna secara bersamaan. CouchDB juga mendukung pengembangan aplikasi dengan bahasa pemrograman seperti JavaScript, Python, dan Ruby.

2.13 Node.js

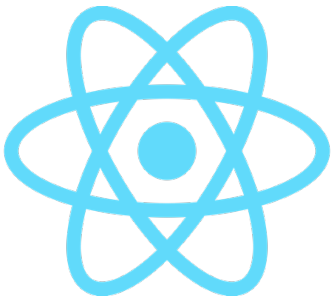
Node.js merupakan sebuah platform perangkat lunak *open source* yang digunakan untuk membangun aplikasi berbasis *server-side* menggunakan bahasa pemrograman JavaScript [56]. Node.js berjalan pada mesin JavaScript bernama V8 yang juga digunakan oleh *browser* Google Chrome.

Gambar 2. 16 Logo Node.js

Node.js menyediakan *runtime environtment* untuk JavaScript yang asinkronus sehingga memungkinkan aplikasi untuk mengoperasikan I/O (input/output) dan jaringan dengan cepat dan efisien. Dengan menggunakan Node.js, pengembang dapat dengan mudah membuat aplikasi *web* karena *backend* dan *frontend* aplikasi menggunakan bahasa pemrograman yang sama yaitu JavaScript. Node.js dilengkapi dengan fitur-fitur seperti modul yang terpisah, manajemen paket dengan NPM (Node Package Manager), serta dukungan protokol HTTP dan WebSocket.

2.14 React.js

React.js atau React merupakan sebuah *library open source* berbasis JavaScript yang berguna dalam membangun *user interface* (UI) atau antarmuka pengguna dengan menyediakan berbagai kode JavaScript yang sudah tertulis (*pre-written*) sehingga dapat meningkatkan efisiensi dalam proses pengembangan aplikasi *website* [57]. React menggunakan pendekatan komponen, di mana UI dibangun sebagai koleksi dari komponen-komponen yang dapat digunakan ulang dan terpisah. Komponen tersebut dapat diatur dalam sebuah hierarki agar pengembangan aplikasi yang kompleks menjadi lebih terstruktur.



Gambar 2. 17 Logo React.js

Selain itu, React juga menggunakan pendekatan *unidirectional data flow*, di mana data diatur dari komponen induk ke komponen anak, sehingga memudahkan proses pengembangan dan pemeliharaan aplikasi yang besar dan kompleks. React dilengkapi dengan fitur-fitur seperti JSX yang memungkinkan penulisan kode JavaScript yang lebih dekat dengan HTML serta dukungan untuk virtual DOM (*Document Object Model*) yang mempercepat proses *rendering* UI.

2.15 Express.js

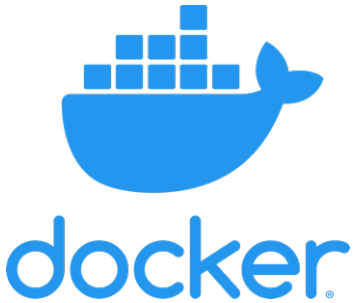
Express.js atau Express adalah sebuah *framework open source* Node.js yang digunakan untuk membangun aplikasi *web* berbasis *server-side* yang betanggungjawab mengatur fungsionalitas *website* seperti pengelolaan *routing* dan *session*, HTTP *request*, penanganan *error*, serta pertukaran data di *server* [58].

Gambar 2. 18 Logo Express.js

Express juga menyediakan fitur seperti *parsing body* permintaan, manajemen sesi, penggunaan *template*. Selain itu, Express juga memiliki *middleware* yang bermanfaat dalam proses autentikasi atau validasi data. Dengan demikian, Express mendukung dan memudahkan proses pengembangan REST API.

2.16 Docker

Docker merupakan platform *open source* yang menyediakan layanan untuk mengemas dan menjalankan aplikasi dalam *container* yang terisolasi dengan aman sehingga memungkinkan pengguna untuk menjalankan banyak *container* di waktu yang bersamaan pada *host* yang telah ditentukan [59].



Gambar 2. 19 Logo Docker

Dalam sebuah *container*, aplikasi beserta dependensinya dibungkus dalam sebuah unit terpisah dan dapat dipindahkan antara lingkungan pengembangan, pengujian, dan produksi tanpa perlu memodifikasi konfigurasi atau kode aplikasi.

2.17 Web3.Storage

Web3.Storage merupakan layanan penyimpanan terdesentralisasi yang memungkinkan pengguna untuk mengunggah dan berbagi konten secara aman dan efisien pada media yang terdesentralisasi. Web3.Storage terdiri dari sekumpulan API dan layanan yang memudahkan pengguna untuk mengunggah dan menyimpan data agar tetap tersedia secara berkelanjutan [60]. Layanan Web3.Storage memanfaatkan teknologi IPFS (InterPlanetary File System) untuk menyimpan dan mendistribusikan konten di seluruh jaringan yang didistribusikan secara global.

Gambar 2. 20 Logo Web3.Storage

Layanan ini merepresentasikan bagaimana teknologi *blockchain* dan sistem yang terdesentralisasi dapat diaplikasikan dalam penyimpanan dan distribusi konten digital. Layanan ini juga memungkinkan pengguna untuk berpartisipasi dalam ekosistem *web* terdesentralisasi dengan keuntungan pada aspek keamanan dan penghematan biaya penyimpanan yang mungkin tidak dapat ditemukan dalam layanan penyimpanan tradisional. Web3.Storage akan memberikan Content Identifier (CID) setiap pengguna mengunggah file. CID merupakan versi yang lebih generik dari *hash* file dalam IPFS dengan kode identifikasi unik. CID digunakan untuk merujuk pada konten dalam sistem IPFS yang mencakup informasi lebih lanjut selain *hash* kriptografis biasa, seperti algoritma *hash* yang digunakan, tipe konten, dan versi *encoding*.

2.18 Leaflet.js

Leaflet.js merupakan *library* JavaScript sumber terbuka yang berfungsi untuk membuat peta digital interaktif yang memiliki keunggulan responsivitas terhadap sejumlah ukuran layar perangkat pengguna, ukuran file yang kecil, dan dapat dilakukan kostumisasi [61]. Leaflet.js dapat digunakan untuk menampilkan peta dengan berbagai macam fitur seperti *panning*, *zoom*, *location marker*, dan poligon.

Gambar 2. 21 Lofo Leaflet.js

Leaflet.js dapat diintegrasikan dengan berbagai sumber data peta, seperti *tile layers* (lapisan tegel) dari berbagai penyedia peta online. Dengan menggunakan Leaflet.js, pengembang dapat dengan mudah membuat peta interaktif yang responsif dan menarik. Hal ini menjadikan Leaflet.js populer untuk membangun tampilan peta pada berbagai jenis proyek *web*.

2.19 React Leaflet

React Leaflet merupakan integrasi Leaflet.js untuk *library* React.js dengan mengabstraksi Leaflet.js ke dalam komponen di dalam React [62]. React Leaflet menyediakan komponen React yang dapat memudahkan penggunaan dan manajemen peta Leaflet.js. Hal tersebut juga memudahkan pembangunan antarmuka yang dinamis dan fleksibel pada React dengan tetap menggunakan fitur-fitur peta yang disediakan oleh Leaflet.js.

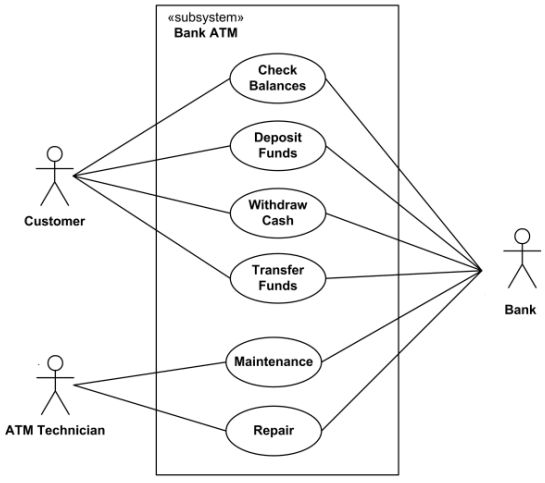
Gambar 2. 22 Logo React Leaflet

2.20 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) merupakan pemodelan berbentuk visual yang digunakan sebagai sarana perancangan sistem berorientasi objek [63]. Hal ini dilakukan untuk memahami, merancang, menelusuri, mengonfigurasi, memelihara, dan mengontrol informasi mengenai sistem yang dirancang [64]. UML banyak digunakan untuk memudahkan komunikasi antara anggota tim, mengidentifikasi kebutuhan sistem, dan merancang solusi yang efektif dan efisien. UML memberikan standar penulisan bagi sistem secara *blue print* meliputi proses bisnis, penulisan *class* dalam bahasa pemrograman yang spesifik, skema *database*, dan komponen- komponen lain yang diperlukan sistem [65]. Selain itu, UML juga memungkinkan agar kendala suatu sistem dapat diketahui sejak tahap perancangan [66]. UML terdiri dari berbagai diagram seperti *use case diagram*, *activity diagram*, *class diagram*, dan *sequence*.

2.20.1 *Use Case Diagram*

Use case diagram adalah diagram yang menggambarkan interaksi antara sistem dengan pihak-pihak yang terlibat dalam penggunaan sistem, baik pengguna maupun pihak lain.

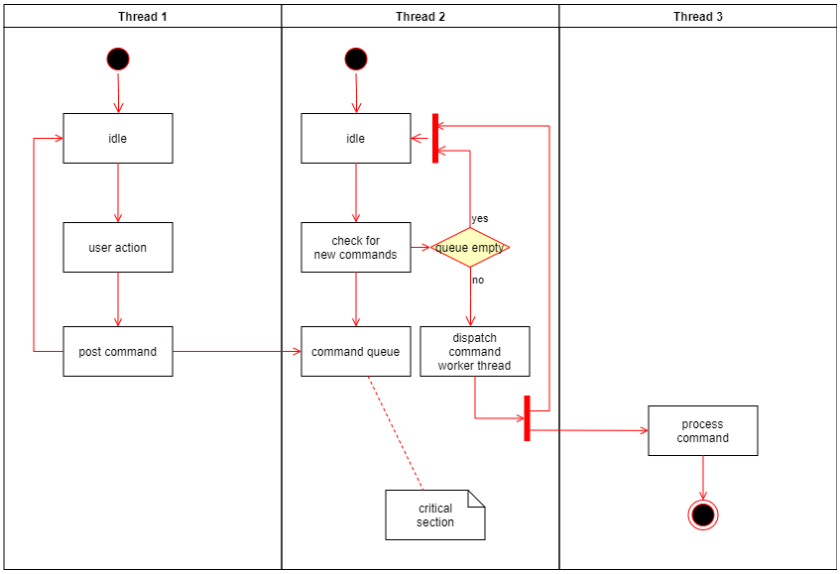


Gambar 2. 23 Diagram *Use Case* [63]

Diagram ini menunjukkan fungsi-fungsi atau fitur-fitur dari sistem serta mendeskripsikan tipe interaksi antara pengguna dengan sistem [63].

2.20.2 *Activity Diagram*

Activity diagram digunakan untuk menggambarkan aktivitas atau tindakan dalam proses bisnis, sistem, atau aplikasi. Diagram ini menunjukkan urutan tindakan dalam bentuk aktivitas atau tugas dan hubungan antara aktivitas tersebut seperti aliran kerja, keputusan, dan percabangan.

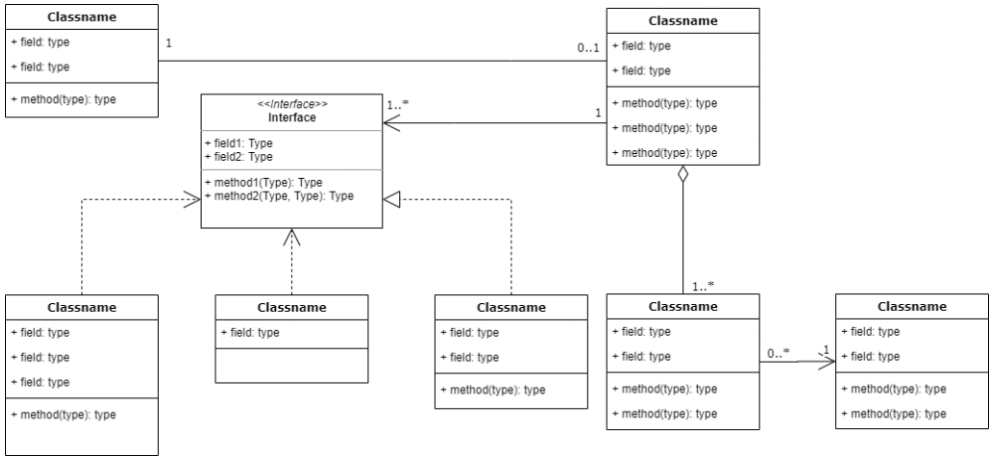


Gambar 2. 24 Diagram *Activity* [63]

Activity diagram menunjukkan aliran suatu aktivitas ke aktivitas lain pada sistem dan lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum [65].

2.20.3 *Class Diagram*

Class diagram digunakan untuk menggambarkan struktur dan deskripsi yang terkait dengan sistem dalam bentuk *class*, *interface*, *object*, relasi antar *class*, dan pewarisan antar kelas. Diagram ini menunjukkan keadaan (atribut/properti) suatu sistem dan cara untuk memanipulasi keadaan tersebut (metode/fungsi) [65].

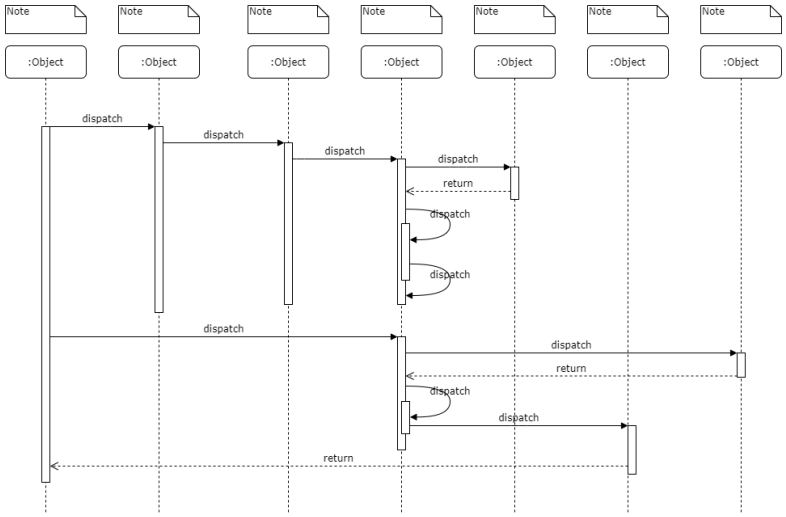


Gambar 2. 25 Diagram *Class* [63]

Desain model pada diagram ini dibagi menjadi dua, yaitu domain model yang merupakan abstraksi dari basis data dan modul program MVC (Model View Controller) yang memiliki *class boundary* sebagai *class interface*, *class contro*l sebagai tempat menyimpan algoritma, serta *class entity* sebagai tabel dalam basis data dan *query* program [63].

2.20.4 *Sequence Diagram*

Sequence diagram adalah diagram yang menunjukkan objek-objek yang terlibat dalam sebuah skenario dan pesan-pesan atau pemanggilan metode yang dikirim antara objek-objek tersebut dalam urutan waktu tertentu.



Gambar 2. 26 Diagram *Sequence* [63]

Diagram ini umumnya digunakan untuk menggambarkan skenario atau langkah-langkah yang dilakukan saat suatu *event* terjadi sehingga dapat menghasilkan *output* tertentu [65]. *Sequence diagram* berguna untuk memvisualisasikan bagaimana objek-objek dalam sistem saling berinteraksi dan berkomunikasi dalam konteks tertentu.

2.21 Metode Pengujian

Metode pengujian merupakan teknik atau pendekatan sistematis yang digunakan untuk mengevaluasi dan memverifikasi kualitas sistem yang dibangun. Tujuannya adalah untuk mengidentifikasi kesalahan atau cacat dalam sistem serta memastikan bahwa sistem berfungsi sesuai dengan ketentuan yang telah ditetapkan.

2.21.1 Pengujian *Black Box*

Pengujian *black box* (*black box testing*) atau *behavioral testing* merupakan metode pengujian perangkat lunak dengan memeriksa hasil *input* dan *output* yang dilakukan tanpa mengetahui bagaimana kode program diimplementasikan [67]. Dalam pengujian ini, *input* diberikan ke sistem dan *output* yang dihasilkan diamati tanpa perlu mengetahui bagaimana sistem mengolah *input* tersebut. Pengujian *black box* dapat dilakukan pada pengujian fungsional, unit, integrasi, dan sistem.