BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

Bagian ini akan menjelaskan teori-teori pokok yang menjadi dasar acuan penulis dalam penelitian. Adapun teori tersebut menggambarkan prinsip dasar yang menjadi pondasi dalam penelitian ini.

2.1.1 Akuntansi

Akuntansi merupakan suatu seni yang melibatkan identifikasi, proses pengumpulan, klasifikasi, dan pencatatan peristiwa transaksi dan keuangan yang terkait. Tujuan dari kegiatan ini adalah untuk menciptakan informasi yang bermanfaat bagi pihak yang berkepentingan, yang kemudian disajikan dalam bentuk laporan keuangan (Sumarsan 2018).

Laporan keuangan memegang peranan krusial dalam dunia bisnis dan kehidupan usaha. Kejelasan laporan keuangan dalam perusahaan sangat penting agar dapat dimengerti dengan mudah oleh para manajer dan pengelola. Dengan begitu, perusahaan akan memiliki analisa dan implementasi kebijakan yang tepat dan akurat (Situmorang 2020).

Dalam praktiknya, akuntansi melibatkan proses pencatatan setiap transaksi dalam perusahaan, seperti penerimaan pendapatan, pengeluaran, investasi, dan pinjaman. Data ini selanjutnya dimanfaatkan untuk menyusun berbagai laporan keuangan seperti laporan laba rugi. Pemahaman yang baik tentang akuntansi sangat penting dalam menjalankan

bisnis atau entitas keuangan lainnya dengan baik dan efisien. Selain itu, akuntansi juga memainkan peran penting dalam menjaga transparansi, akuntabilitas, dan kepatuhan terhadap standar akuntansi yang berlaku.

Semua transaksi keuangan dicatat dalam jurnal umum (*general journal*) dalam sistem akuntansi perusahaan. Jurnal umum (*general journal*) adalah jurnal tempat dicatatnya transaksi yang tidak termasuk dalam jurnal khusus, seperti penyesuaian dan koreksi (Lestari dan Amri 2020). Dalam jurnal umum, setiap transaksi dicatat dengan rincian seperti tanggal transaksi, deskripsi transaksi, akun yang terlibat, jumlah debit, dan jumlah kredit. Setelah dicatat dalam jurnal umum, data transaksi kemudian ditransfer atau diposting ke buku besar dalam akun yang relevan.

Buku besar adalah kumpulan semua akun milik perusahaan dan saldo-saldonya. Semua akun milik perusahaan terhubung dan membentuk satu kesatuan. Siklusnya dimulai dengan meninjau dan mengurutkan dokumen berdasarkan jenis transaksinya kemudian dicatat dalam jurnal. Kemudian setiap periode tertentu seperti seminggu sekali, ringkasan dalam jurnal tersebut di-*posting* ke buku besar sesuai dengan jenis akunnya (Muda dkk. 2017).

Buku besar memungkinkan perusahaan untuk memantau dan mengelola saldo dari setiap akun secara terperinci. Ini mencakup akun-akun seperti aset, liabilitas, ekuitas, pendapatan, dan biaya. Dengan buku besar, perusahaan dapat mengetahui saldo akun pada suatu titik waktu tertentu dan melacak perubahan saldo akun seiring berjalannya waktu. Buku besar juga memiliki peran sentral dalam penyusunan laporan keuangan karena data dari buku besar digunakan dalam

penyusunan laporan-laporan ini untuk memberikan gambaran yang akurat tentang keuangan perusahaan.

Sistem akuntansi sangat krusial bagi perusahaan karena menghasilkan laporan keuangan yang akurat dan relevan, yang nantinya dapat dimanfaatkan oleh berbagai pihak dalam proses pengambilan keputusan. Adapun infomasi keuangan atau laporan keuangan yang akan diadopsi oleh sistem akuntansi pada PT Segara Catur Pekasa adalah laporan laba rugi, yaitu laporan yang meliputi pendapatan perusahaan, biaya, serta laba atau rugi bersih yang dialami perusahaan dalam rentang waktu tertentu. Laporan ini dapat memberikan gambaran tentang kinerja finansial perusahaan dalam periode tersebut.

2.1.2 Website

Website merupakan singkatan dari "*World Wide Web Site*" merupakan suatu rangkaian halaman *web* di *internet*. Halaman-halaman ini dapat mengandung berbagai jenis informasi atau berbentuk aplikasi *web* yang disusun untuk tujuan tertentu.

Menurut (Vermaat dkk. 2018) *website* merupakan sekelompok halaman yang saling terkait, dimana terdapat berbagai elemen seperti gambar dan dokumen yang disimpan di dalam *web server*. Selain itu, *web server* juga dapat memuat aplikasi *web* yang dapat diakses oleh pengguna melalui peramban web (*web browser*).

Sedangkan menurut (Nurmi, 2017) dalam jurnal (Izzah 2020) *website* juga dapat didefinisikan sebagai sekumpulan halaman yang mampu menyajikan

berbagai informasi dalam bentuk gambar, suara, animasi, teks, atau kombinasi dari berbagai bentuk tersebut.

Sebuah *website* dapat diakses menggunakan *Uniform Resource Locator* (URL) yaitu alamat unik yang berfungsi untuk mengarahkan *user* pada *website* tujuan . *Website* juga dapat berfungsi sebagai sarana komunikasi, hiburan, bisnis, atau pendidikan. *Website* telah menjadi salah satu sarana untuk berinteraksi dengan dunia maya dan berbagi informasi karena kemudahan dan kecepatannya. Selain itu *website* juga bermanfaat untuk meningkatkan kredibilitas bisnis atau perusahaan, serta dapat menjangkau audiens secara luas.

2.2 Teori Khusus

Bagian ini menjelaskan teori-teori yang menjelaskan atau mendukung konsep teoritis yang berkaitan langsung dengan perancangan *REST API* sistem akuntansi berbasis web.

2.2.1 *REST API*

Menurut (Muri, Utomo, dan Sayyidati 2019) *Application Programming Interface* (*API*) merupakan sebuah konsep dimana fungsi dapat dipanggil oleh program lain dengan menjadi perantara atau jembatan dari berbagai aplikasi yang berbeda *platform.*

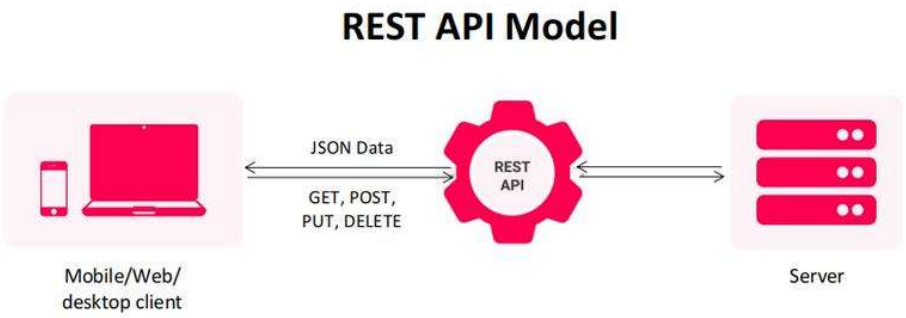
Sementara itu arsitektur *REST* (*Representational State Transfer*) adalah standar arsitektur komunikasi berbasis *web* yang juga banyak diterapkan untuk pengembangan sistem terdistribusi atau layanan berbasis *web* (*web services*)

(Danandjaya dan Aji 2018). Konsep ini, yang awalnya diperkenalkan oleh Roy Fielding dalam disertasinya pada tahun 2000, telah menjadi salah satu pendekatan yang paling umum digunakan dalam pengembangan sistem berbasis *web*. Arsitektur *REST* berlandaskan pada beberapa prinsip inti, antara lain mengidentifikasi sumber daya dengan menggunakan *Uniform Resource Identifier* (*URI*), menerapkan metode *HTTP* seperti *DELETE*, *GET*, *POST*, dan *PUT* untuk berinteraksi dengan sumber daya, serta merepresentasikan data dalam berbagai format seperti JSON atau XML.

Salah satu karakteristik utama dari arsitektur *REST* adalah kemampuannya untuk berkomunikasi dengan berbagai bahasa pemrograman dan *platform* sehingga arsitektur *REST* menjadi pilihan populer dalam mengembangkan *API* (*Application Programming Interface*) karena sistem-sistem berbeda menjadi bisa berkomunikasi dan berbagi data melalui protokol *HTTP*. Selain itu, arsitektur *REST* juga mendukung arsitektur *client-server* yang memungkinkan sistem dapat dikembangkan dengan fleksibel dan memiliki skalabilitas yang baik.

Jadi secara sederhana *REST API* (*Representational State Transfer Application Programming Interface*) merupakan *API* yang didasarkan pada prinsip-prinsip arsitektur *REST*. *API* ini sering dipakai dalam proses pengembangan aplikasi untuk *web* dan *mobile.* karena dapat mengintegrasikan layanan dan berbagi data antara sistem yang berbeda. *API* ini didesain untuk memungkinkan komunikasi antara perangkat lunak atau sistem yang berbeda melalui jaringan.

Pada dasarnya, arsitektur *REST* merupakan suatu arsitektur *client-server* yang didesain untuk menggunakan protokol *stateless*, seperti *HTTP*. Secara umum arsitektur *REST API* dapat digambarkan sebagai berikut:



Gambar 2.1 *REST API* Model

Sumber : https://www.mindinventory.com

Pada gambar di atas dapat dilihat bahwa *API* dapat digunakan pada *platform* yang berbeda yaitu *mobile*, *web* dan *desktop client. Client* berfungsi sebagai tempat atau antarmuka pengguna yang digunakan untuk mengirimkan *request* dengan metode HTTP. *Request* tersebut kemudian diteruskan ke *server* oleh *API*, kemudian *server* menerima *request* dari *client* dan memprosesnya sesuai dengan metode HTTP yang digunakan, *server* akan mengidentifikasi sumber daya yang diminta dan mengambil atau mengolahnya terlebih dahulu sesuai dengan operasi yang diperlukan, setelah itu *server* akan mengirimkan respon dan status *HTTP* atas *request* yang diterima, respon dikirimkan ke *client* melalui *API* dengan format seperti JSON atau XML.

REST API memiliki keunggulan dan kekurangan yaitu:

1. Kelebihan
   a. Dapat digunakan pada beberapa platform yang berbeda.
   b. Dapat digunakan oleh beberapa bahasa pemrograman yang berbeda.
   c. *Client* dan *server* yang terpisah, sehingga memudahkan pengembangan karena komponen pengembangan dapat dikembangkan secara independen.
   d. Memungkinkan sistem yang dibuat untuk diintegrasikan dengan sistem lain.
2. Kekurangan
   a. Proses pengembangan cenderung lebih lama karena memerlukan tim.

2.2.2 *Unified Modeling Language* (UML)

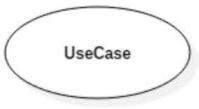
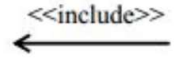
Unified Modeling Language (UML) salah satu bahasa yang kegunaannya adalah mengilustrasikan arsitektur, membuat analisis, merincikan kebutuhan, dan mendeskripsikan rancangan dalam konteks *object-oriented programming* (Putra dan Andriani 2019). UML membantu dalam pemahaman, komunikasi, dan analisis perangkat lunak dengan menyediakan serangkaian diagram yang memungkinkan para pihak yang terlibat selama pengembangan dalam mengekspresikan aspek-aspek berbeda dari sistem, seperti struktur, perilaku, dan interaksi antara komponen-komponennya.

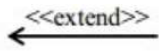
Dalam penelitian ini, UML digunakan sebagai sarana analisis dalam merancang *REST API* pada sistem akuntansi berbasis *web.* Berikut adalah diagram yang digunakan dalam penelitian ini:

1. *Use case diagram*

Diagram *use case* adalah representasi grafis yang mengilustrasikan interaksi antara aktor dan sistem, di mana aktor merupakan representasi dari entitas, baik itu individu maupun objek lain seperti perangkat atau sistem. (Purnasari, Hartiwi, dan Nurhayati 2022). *Use case* dibuat agar pengguna dapat lebih mudah memahami gambaran sistem yang dibuat.

Tabel 2.1 Simbol *Use case diagram*

Simbol	Deskripsi
	*Use case* merupakan representasi abstrak dari interaksi antara sistem dan aktor.
	*Actor* merupakan pihak sistem atau pengguna yang mengarahkan dan mengontrol sistem dari luar.
	*Association* merupakan abstraksi dari keterhubungan antara *use case* dengan *actor.*
	*Generalization* merupakan indikasi bahwa suatu aktor telah berspesialisasi untuk dapat terlibat dalam suatu *use case*
	Menerangkan bahwa suatu *use case* sepenuhnya merupakan bagian dari fungsionalitas *use case* lainnya.

	Menerangkan bahwa suatu *use case* menjadi tambahan fungsional dari *use case* lainnya apabila kondisi tertentu terpenuhi.

Sumber : (Sukamto dan Salahuddin 2018)

2. *Activity diagram*

Activity diagram adalah gambaran visual dari urutan aktivitas yang terjadi dalam suatu operasi atau proses. Diagram ini menyerupai *flowchart* karena menggambarkan alur kerja (*workflow*) dari aplikasi atau sistem. *Activity diagram* dapat membantu *stakeholders* dan *developer* dalam memahami bagaimana keseluruhan proses berjalan, selain itu *activity diagram* juga dimanfaatkan untuk mengilustrasikan interaksi di antara beberapa *use case*. (Isa dan Hartawan 2017).

Tabel 2.2 Simbol *Activity diagram*

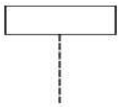
Simbol	Deskripsi
	Menerangkan simbol awal atau mulai pada sistem.
	Menerangkan aktivitas yang dilakukan didalam operasi.
	Menerangkan percabangan yang ada pada sistem,
	Menerangkan gabungan pada sistem.
	Menerangkan status Akhir pada sistem.

Sumber : (Sukamto dan Salahuddin 2018)

3. *Sequence* diagram

Sequence diagram merupakan representasi visual dari perilaku objek dalam suatu *use case*. Diagram ini menyajikan informasi tentang waktu hidup objek beserta pesan-pesan yang dikirimkan dan diterima oleh objek tersebut. Jumlah diagram *sequence* yang dibuat bergantung pada banyaknya pedefinisian *use case* yang memiliki proses sendiri (Putra dan Andriani 2019).

Tabel 2.3 Simbol *sequence* diagram

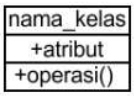
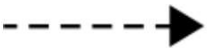
Simbol	Deskripsi
	*Life line*: Representasi dari entitas objek atau *interface* yang saling berhubungan dalam diagram *sequence*.
	*Message*: Deskripsi interaksi antar objek yang memuat informasi mengenai aktivitas yang terjadi.
	*Message*: Penjelasan mengenai interaksi antar objek berupa respon yang mencakup informasi tentang aktivitas yang terjadi.

Sumber : (Mufid 2023)

4. *Class diagram*

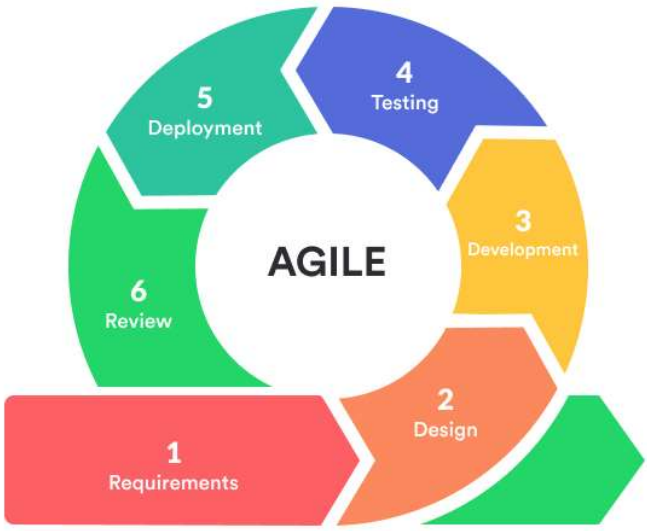
Class diagram menguraikan struktur sistem dengan merinci kelas-kelas yang terlibat dalam sistem yang dikembangkan. Diagram ini terdiri dari atribut dan operasi yang bertujuan untuk menyesuaikan hubungan antara perangkat lunak dan dokumentasi perancangan (Putra dan Andriani 2019).

Tabel 2.4 Simbol *class diagram*

Simbol	Deskripsi
	*Class* : Representasi simbolis yang menghubungkan informasi dari para pelaku.
	Asosiasi berarah : Hubungan yang diterapkan oleh satu kelas ke kelas lain, seringkali melibatkan konsep *multiplicity.*
	Generalisasi: Keterkaitan antara kelas yang mencerminkan hubungan generalisasi-spesialisasi.
	Simbol i*nterface* atau antarmuka mirip dengan kelas, tetapi mencakup metode yang dideklarasikan tanpa konten implementasi.
	*Dependency* / kebergantungan : Simbol yang merepresentasikan hubungan ketergantungan antar kelas..

Sumber : (Ansori 2022)

2.2.3 *Agile Software Development*



Gambar 2.2 *Agile Software Development Life Cycle*

Sumber : https://mlsdev.com/blog/agile-sdlc

Agile Software Development ialah suatu metodologi pengembangan perangkat lunak (*software*) yang ringan yang diciptakan untuk mengatasi keterbatasan metode pengembangan yang kompleks. dengan tujuan mengurangi masalah dan biaya serta memberikan fleksibilitas untuk mengakomodasi perubahan kebutuhan kapan pun diperlukan. Pendekatan ini mengelola tugas-tugas dan koordinasinya melalui seperangkat nilai dan prinsip tertentu (Al-Saqqa, Sawalha, dan Abdelnabi 2020).

Agile merupakan suatu pendekatan yang berfokus pada fleksibilitas, kolaborasi, dan kebutuhan pelanggan. Dalam metode ini, proyek perangkat lunak dibagi menjadi iterasi singkat yang disebut "*sprint*" yang biasanya berlangsung selama beberapa minggu. Pada setiap sprint, tim pengembang akan mengerjakan

sejumlah fitur atau fungsi yang telah tentukan sebelumnya, kemudian hasil dari setiap tim akan diperiksa dan dievaluasi oleh *stakeholders*.

Dalam proses pengembangan proyek menggunakan metode *agile* perubahan dan penyesuaian dapat dilakukan dengan lebih cepat karena *agile* mengutamakan komunikasi terbuka antara tim dalam pengembang. Hal ini menjadikan *agile* sebagai metode yang sangat adaptif terhadap perubahan kebutuhan yang mungkin muncul selama proyek pengembangan berlangsung.

Pendekatan *agile* juga mengutamakan produk yang dapat digunakan secara berkala, sehingga para *stakeholders* dapat melihat perkembangan proyek lebih awal dan lebih sering, tentu saja hal ini berguna untuk mengurangi risiko dan memastikan bahwa hasilnya lebih sesuai dengan ekspektasi. Menurut (Lubis 2023) tahapan-tahapan dari metode *agile* adalah:

1. *Requirement* (*Planning*)

   Tahap pertama dalam metode ini adalah pengidentifikasian dan pengumpulan kebutuhan kemudian merencanakan langkah-langkah untuk mencapai hasil yang diinginkan. Hal ini melibatkan komunikasi antara tim pengembangan dan pemangku kepentingan (*stakeholders*) untuk memahami apa yang harus dicapai oleh *software* atau produk yang akan dikembangkan.

2. *Design* (Desain)

   Setelah mengidentifikasi kebutuhan, tim pengembangan membuat rancangan *software* yang mencakup perancangan arsitektur sistem, *user interface* (UI/UX), dan desain *database*. Pembuatan desain harus mempertimbangkan kebutuhan *stakeholders* dan bagaimana interaksi antar komponen-komponen di dalamnya.

3. *Development* (Pengembangan)

   Tahap berikutnya adalah membangun *software* berdasarkan spesifikasi dan desain yang telah dibuat sebelumnya. Tahapan ini adalah tahap mengimplementasikan fitur-fitur dan fungsionalitas yang dibutuhkan dalam *source code* perangkat lunak (*software*).

4. *Testing* (Pengujian)

   Setelah pengembangan selesai maka tahap berikutnya adalah pengujian. Tahap ini sangat penting karena memverifikasi kinerja perangkat lunak, memastikan kesesuaian dengan desain yang telah ditetapkan, dan memenuhi kebutuhan pelanggan atau *stakeholders*. Tim pengujian akan melaksanakan sejumlah pengujian, termasuk pengujian unit, integrasi, fungsional, dan penerimaan. Tujuan pengujian adalah memastikan bahwa perangkat lunak berfungsi dengan baik, memenuhi kebutuhan, dan tidak memiliki *bug* atau masalah.

5. *Deployment* (Implementasi)

   Setelah perangkat lunak diuji dan dianggap siap, tahap implementasi dimulai. Perangkat lunak di-*deploy* ke *production environtment* (lingkungan produksi) atau *platform* yang sesuai. Pengguna dapat mulai menggunakan perangkat lunak.

6. *Review* (Evaluasi)

   Setelah penggunaan awal perangkat lunak, ada tahap *review* atau evaluasi. Para *stakeholders* dan pengguna memberikan *feed back* (umpan balik) tentang kinerja

perangkat lunak, kebutuhan tambahan, atau perubahan yang diperlukan. *Review* ini dapat menjadi perbaikan atau perubahan dalam iterasi berikutnya.

2.2.4 Sekilas PT Segara Catur Perkasa

PT Segara Catur Perkasa adalah Badan Usaha Pelabuhan (BUP) milik swasta yang bergerak dibidang penyedia jasa Pemanduan dan Penundaan kapal di Batam. Berdasarkan Peraturan Pemerintah No.31 Tahun 2021 Tentang Penyelenggaraan Bidang Pelayaran "Badan Usaha Pelabuhan adalah badan usaha yang kegiatan usahanya khusus di bidang pengusahaan Terminal dan Fasilitas Pelabuhan lainnya."

Berdasarkan Peraturan Menteri Perhubungan Republik Indonesia No.57 Tahun 2015 tentang Pemanduan dan Penundaan kapal "Pemanduan adalah Kegiatan Pandu dalam membantu, memberi saran dan informasi kepada Nakhoda tentang keadaan perairan setempat yang penting agar navigasi-pelayaran dapat dilksanakan dengan selamat, tertib, dan lancer demi keselamatan kapal dan lingkungan" sedangkan "Penundaan Kapal adalah bagian dari pemanduan yang meliputi kegiatan mendorong, menarik, menggandeng, mengawal (*escort*), dan membantu (*assist*) kapal yang berolah-gerak di alur-pelayaran, daerah labuh jangkar maupun kolam pelabuhan, baik untuk bertambat kea tau untuk melepas dari dermaga, *jetty, trestle, pier,* pelampung, *dolphin,* kapal, dan fasilitas tambat lainnya dengan mempergunakan kapal tunda sesuai dengan ketentuan yang dipersyaratkan."

PT Segara Catur Perkasa sudah berdiri sejak tahun 2021 dan mulai mengungguli pasar di awal tahun 2023, seiring dengan perkembangan itu maka kebutuhan perusahaan akan perangkat pendukung juga ikut bertambah agar dapat menjaga performanya.

2.3 Software yang digunakan

Bagian ini akan menguraikan alat (*tools*) atau perangkat lunak (*software*) pendukung yang digunakan untuk membangun sistem pada penelitian ini, berikut *tools* / *software* pendukung yang digunakan:

2.3.1 Node JS

Gambar 2.3 Node.js

Sumber : https://wikipedia.org/

Node.js adalah lingkungan *runtime* JavaScript yang memungkinkan pengembang menjalankan kode JavaScript di sisi *server*. Node.js cepat dan andal dalam mengelola *file* yang berat dan aplikasi dengan muatan jaringan yang berat karena berbasis *event-driven*, *non-blocking*, dan pendekatan *asynchronous*, Node.js juga memungkinkan pengembang untuk memelihara (*maintain)* aplikasi dalam satu

halaman utuh (*Single Page Applications*) dan juga bisa digunakan untuk proyek berbasis IOT (*Internet of Things*) (Shah dan Soomro 2017).

Node.js juga digunakan untuk membangun aplikasi *real-time* yang membutuhkan performa dan skalabilitas tinggi. Node.js memberikan kemampuan untuk mengelola banyak koneksi secara efisien dan merespon permintaan dengan cepat.

Node.js memiliki manajer paket yang bernama NPM (*Node.js Package Manager*) yang dibuat pada tahun 2009 sebagai proyek sumber terbuka untuk membantu pengembang JavaScript dengan mudah berbagi modul paket kode (Anon t.t.). Dalam ekosistem Node.js peran NPM sangat penting karena NPM memungkinkan pengembang yang menggunakan bahasa pemrograman JavaScript untuk mencari, menginstal, mengelola, atau mendistribusikan paket dan modul JavaScript yang dibutuhkan. NPM menyediakan repositori publik yang berisi ribuan paket dan modul yang dapat digunakan secara gratis, pengembang dapat dengan mudah mengimpor dan menggunakan paket-paket tersebut dalam proyek mereka, menjadikan pengembangan perangkat lunak lebih efisien.

Berikut adalah paket (*package*), modul, *framework* dan *library* yang digunakan dalam penelitian ini:

1. Express.js

   Express.js adalah kerangka kerja (*framework*) aplikasi *web* yang minimal dan fleksibel untuk Node.js. *Framework* ini menyediakan serangkaian fitur yang dapat membantu proses pengembangan aplikasi *web*. Express.js juga memudahkan pengembang dalam membuat *API*, karena banyaknya metode

utilitas *HTTP* dan *middleware* (Dalbard dan Isacson 2021). Salah satu keunggulan Express adalah kemampuannya untuk mengelola rute (*routes*) atau *endpoint HTTP* dengan mudah. Express.js mendukung *middleware*, yang merupakan fungsi-fungsi yang dapat digunakan untuk melakukan berbagai tugas seperti otentikasi, otorisasi, validasi input, dan lainnya. *Middleware* memungkinkan pengembang untuk memisahkan logika aplikasi menjadi modul-modul yang lebih kecil dan dapat digunakan kembali.

2. React.js

   React.js ialah sebuah pustaka (*library*) JavaScript yang dimanfaatkan untuk menciptakan antarmuka pengguna atau biasa disebut dengan UI (*user interface*), salah satu keunggulan React.js adalah konsep komponen dimana antarmuka pengguna dibangun sebagai komponen independen yang dapat digunakan kembali. Setiap komponen dapat memiliki logika sendiri dan dapat digunakan untuk menampilkan bagian tertentu dari antarmuka pengguna.

3. Sequelize

   Sequelize adalah sebuah *library* atau lebih tepatnya *ORM* (*Object-Relational Mapping*) yang digunakan dalam pengembangan aplikasi berbasis Node.js yang berinteraksi dengan *relational database*, seperti MySQL, PostgreSQL, SQLite, dan lainnya. Sequelize memberi cara untuk pengembang dapat berinteraksi dengan basis data menggunakan model dan objek JavaScript, mengabstraksi operasi SQL yang kompleks, dan memfasilitasi proses pengembangan aplikasi yang lebih efisien. Dengan Sequelize, pengguna dapat dengan mudah mendefinisikan skema *database*, menjalankan kueri *database*, dan melakukan

operasi CRUD (*Create, Read, Update, Delete*) tanpa harus menulis SQL secara langsung. Hal ini sangat bermanfaat dalam pembuatan aplikasi berbasis *server* yang membutuhkan akses dan manipulasi data dalam basis data relasional, karena mengurangi kompleksitas dalam mengelola kueri SQL dan memungkinkan pengembang untuk fokus pada logika bisnis aplikasi mereka.

2.3.2 *Visual Studio Code*

Gambar 2.4 *Visual Studio Code*

Sumber : https://www.stickpng.com/

Visual Studio Code (*VS Code*) adalah lingkungan pengembangan terintegrasi atau *Integrated Development Environtment* (IDE) yang dikembangkan oleh Microsoft yang dirancang khusus untuk pengembangan perangkat lunak dan mendukung banyak bahasa pemrograman populer, termasuk JavaScript, Python, C++, dan banyak lagi (VS Code, 2023).

VS Code memiliki berbagai fitur yang memudahkan pengembangan perangkat lunak seperti *auto-complete code* (penylesaian kode otomatis), *integrated version management* (manajemen versi yang terintegrasi), dan banyak ekstensi yang dapat diinstal untuk mendukung fungsionalitasnya.

Salah satu fitur unggulan VS Code adalah kemampuan untuk mengintegrasikan dengan berbagai alat pengembangan, seperti Github untuk manajemen versi, serta alat *debugging* untuk berbagai bahasa pemrograman. Selain itu, VS Code dapat dioperasikan di berbagai *platform*, seperti Linux, macOS, dan Windows, membuatnya menjadi salah satu editor kode yang banyak digunakan oleh *software developer*. VS Code memiliki beberapa keunggulan, yaitu:

1. Ringan dan Cepat
2. Memiliki banyak Ekstensi
3. Integrasi dengan Git
4. Mendukung berbagai macam bahasa pemrograman
5. *Debugging* berbagai macam bahasa pemrograman
6. Integrasi dengan layanan *Cloud*
7. *Intellisense* (prediksi dan saran kode selama penulisan kode)

2.3.3 Postman

Gambar 2.5 Postman
Sumber : https://logowik.com/

Postman adalah *software* yang digunakan untuk menguji, mendokumentasikan, dan berinteraksi dengan *API*. Postman memudahkan *developer* dalam mengirim *HTTP request* ke *API* yang berbeda, mengamati dan menganalisis respon dari *API*, serta menyusun permintaan dan respon dalam bentuk yang terstruktur.

Postman juga memiliki *interface* yang mudah digunakan, antarmuka ini membuat proses *endpoint testing* atau *request* ke *API* menjadi lebih mudah. Postman memungkinkan eksplorasi dan pengujian *API web*, serta membantu penguji dan pengembang untuk mengetahui cara kerja *API*. Dengan Postman, pengguna dapat membuat otomatisasi pengujian yang efektif untuk jenis *API* apa pun (Westerveld 2021).

Postman memiliki fitur *collection* yang berfungsi untuk mengorganisir *request API* kedalam kelompok yang terstruktur, hal ini memudahkan pengembang dalam mengelola *request* yang berkaitan. Postman juga memiliki fitur kolaborasi dimana sebuah *workspace* atau ruang kerja dapat dikelola atau digunakan oleh tim, fitur ini memfasilitasi kerja sama dalam menguji dan mengembangkan *API*.

Selain itu Postman juga digunakan untuk membuat dokumentasi dari *API* yang telah dibuat, Postman memiliki fitur dokumentasi *API* yang mana dengan fitur tersebut pengguna dapat membuat dokumentasi secara otomatis dari *API collection* yang telah dibuat.

Berikut adalah beberapa fitur lain yang dapat membantu meningkatkan efisiensi kerja pengembang dalam membangun *API*:

1. Pengujian otomatis, Postman mendukung pengujian otomatis (*automated testing*) dengan menyediakan alat untuk menulis skrip pengujian.
2. *Environment Management*, pengguna dapat mengelola berbagai lingkungan seperti lingkungan pengembangan (*development*), uji(*testing*), dan produksi (*production*). Fitur ini memungkinkan pengguna mengganti nilai-nilai variabel dengan cepat.
3. *Integration with Services*, Postman dapat terintegrasi dengan layanan-layanan seperti Git, Jenkins, dan layanan *cloud storage*.

Fitur-fitur di atas dapat meningkatkan efisiensi dalam seluruh siklus pengembangan perangkat lunak.

2.3.4 Git Hub

Gambar 2.6 GitHub

Sumber : https://png.monster/github-logo-png/

Github merupakan *System Control* yang berfungsi sebagai pengendali kode bahasa pemrograman yang memberikan fasilitas untuk mengembangkan project secara berkolaborasi (Sari dan Ekohariadi 2021).

GitHub dikenal sebagai *platform* sistem pengontrol versi (*version control system*) yang memungkinkan pengguna untuk menyimpan berbagai versi kode, sehingga memudahkan pengembang untuk berkolaborasi, melacak perubahan, dan mengatasi konflik yang mungkin terjadi dalam pengembangan perangkat lunak.

Pengguna dapat membuat *repository* yang berisi *source code* proyek, dan pengembang dapat menyimpan setiap perubahan dengan melakukan "*commit*", di dalam sebuah *commit* terdapat kode yang berubah, siapa yang melakukan perubahan, dan kapan perubahaan kode tersebut dilakukan. GitHub juga mendukung kolaborasi tim dengan fitur *pull requests* yang memungkinkan pengembang untuk memberikan masukan, meninjau perubahan, dan menggabungkan (*merging*) *source code* dengan lebih mudah. GitHub menjadi alat yang penting dalam pengembangan perangkat lunak karena dapat membantu tim pengembang untuk bekerja bersama dengan lebih efisien.

2.3.5 Vercel

Gambar 2.7 Vercel

Sumber : https://logowik.com/vercel-vector-logo-9512.html

Vercel adalah sebuah *platform cloud* yang mengkhususkan diri dalam *hosting* dan distribusi aplikasi dan situs berbasis web. Yang membedakan Vercel dengan layanan *hosting* lain adalah fokusnya pada *deploy* yang otomatis dan cepat dari aplikasi berbasis JavaScript, termasuk aplikasi React, Angular, Vue, dan banyak lagi.

Salah satu keuntungan menggunakan Vercel adalah infrastruktur yang dapat disesuaikan dengan kebutuhan pengguna. Vercel mendukung beberapa *runtime*

yang mana *runtime* tersebut juga memiliki kumpulan pustaka, *API*, dan fungsionalitasnya sendiri. hal tersebut yang memberikan keuntungan tersendiri bagi penggunanya (Vercel 2023).

Vercel memanfaatkan teknologi *serverless* untuk menghadirkan aplikasi dengan kecepatan tinggi dan skalabilitas yang baik, sehingga memungkinkan pengembang untuk fokus pada pengembangan aplikasi tanpa perlu khawatir tentang infrastruktur *server*. Selain itu, Vercel memiliki integrasi yang kuat dengan berbagai alat pengembangan, seperti Git dan GitHub, sehingga memudahkan proses *deploy* dan kolaborasi tim.

2.3.6 Draw.io

Gambar 2.8 Draw.io

Sumber : https://vedcraft.com/

Draw.io adalah sebuah *platform* yang berfungsi untuk membuat visualisasi desain rancangan dan diagram seperti diagram *usecase, activity, sequence dan class* serta *mockup interface* (Satyahadewi dan Mutiah 2019). Draw.io memungkinkan pengguna untuk membuat diagram, bagan, dan grafik secara mudah.

Draw.io merupakan sebuah aplikasi berbasis *web* yang tidak memerlukan instalasi, sehingga pengguna dapat membuat dan mengedit diagram secara langsung dari *web browser*. Selain itu, Draw.io juga mendukung berbagai format

penyimpanan, sehingga pengguna dapat mengimpor dan mengekspor diagram dalam berbagai format *file*, termasuk XML, PNG, JPG, dan lainnya. *Tools* ini sangat berguna dalam pengembangan perangkat lunak, manajemen proyek, dan berbagai bidang lain yang memerlukan visualisasi dan diagram.

2.4 Penelitian Terdahulu

Peneliti merujuk pada penelitian terdahulu sebagai sumber inspirasi untuk membangun landasan teoritis dan konseptual penelitian ini. Selain itu, penelitian sebelumnya juga dijadikan sebagai pembanding yang relevan dengan fokus penelitian ini. Adapun penelitian terdahulu yang menjadi rujukan meliputi:

1. Dalam penelitian (Choirudin & Adil, 2019) yang berjudul "**IMPLEMENTASI *REST API WEB SERVICE* DALAM MEMBANGUN APLIKASI *MULTIPLATFORM* UNTUK USAHA JASA**" ISSN : 1858-4144. Dijelaskan bahwa perkembangan kegiatan pariwisata di pulau Lombok membuat banyak profesi di bidang jasa menjadi ditekuni masyarakat dan salah satunya adalah jasa tukang, hal tersebut menciptakan kebutuhan akan sarana untuk menunjang kemudahan menjalankan usaha dan pencarian jasa tersebut sehingga dibuatlah aplikasi *multiplatform* dengan arsitektur *REST API web service* yang memudahkan pengguna untuk memilih *platform* yang digunakan karena dapat diakses melalui jaringan internet, aplikasi dibangun dengan metode pengembangan *waterfall* kemudian diujikan kepada beberapa tukang. Hasilnya adalah 51,4% responden menjawab sangat setuju sehingga disimpulkan bahwa aplikasi memenuhi kebutuhan tukang. Perbedaan utama antara penelitian ini

dengan penelitian yang dilakukan oleh peneliti sebelumnya terletak pada bahasa pemrograman yang digunakan. Peneliti sebelumnya menggunakan bahasa pemrograman PHP dengan *framework* Laravel untuk mengembangkan REST API dan aplikasi yang digunakan, sementara penelitian ini menggunakan bahasa pemrograman Javascript dan Node.js sebagai runtime, selain itu metode pengembangan yang digunakan oleh peneliti sebelumnya adalah *waterfall* sementara penelitian ini menggunakan metode pengembangan *agile* yang lebih cepat dalam koordinasi antar tim dan adaptif terhadap kebutuhan pengguna.

2. Dalam penelitian yang dilakukan oleh (Prayogi dkk. 2020) dengan berjudul "***Design and Implementation of REST API for Academic Information System***" Scopus : 2-s2.0-85089075166. Dijelaskan bahwa seiring meningkatnya jumlah sistem yang digunakan oleh organisasi mengakibatkan pertukaran data di dalamnya menjadi hal yang perlu diperhatikan, tak terkecuali kinerja dari *API* yang digunakan oleh sistem tersebut, sehingga dilakukan sebuah uji coba terhadap *API* dari 2 teknologi *server* yang berbeda yaitu Node.js dan PHP pada sistem informasi akademik menggunakan metode *prototype*, hasilnya adalah *REST API* yang dibangun menggunakan Node.js memiliki kinerja yang lebih baik yaitu 100% keluaran untuk *request* secara bersamaan hingga 1000 permintaan, sementara PHP 48,70% dengan *request* bersamaan dengan jumlah yang sama. Perbedaan dari penelitian ini dengan penelitian yang dilakukan oleh peneliti sebelumnya adalah metode pengembangan yang digunakan dimana penelitian sebelumnya menggunakan metode *prototype* untuk membangun *REST API* sementara penelitian ini menggunakan metode *Agile Software Development*.

3. Dalam penelitian (Ariesta dkk. 2021) yang berjudul "**PENERAPAN METODE *AGILE* DALAM PENGEMBANGAN *APPLICATION PROGRAMMING INTERFACE SYSTEM* PADA PT XYZ**" ISSN : 2599-3321. Dijelaskan bagaimana perusahaan yang berfokus pada Solusi Teknologi Informasi dapat menerapkan metode pengembangan *Agile Development* dalam proses pembuatan *API*. Dengan metode *agile* diharapkan pengembangan sebuah *API* dapat dilakukan lebih cepat karena karena dapat dilakukan pengembangan secara terpisah oleh beberapa tim. Pada implementasinya, ada alat bantu atau *software* yang digunakan yaitu Jira yang merupakan aplikasi *Project Management* yang membantu menangani proyek pengembangan *API* ini hingga selesai. Hasilnya adalah implementasi metode *Agile scrum* berjalan dengan baik dalam proyek pengembangan *API*, serta memiliki urutan kerja dan fungsi dari sistem dapat lebih mudah dipahami. Perbedaan antara penelitian ini dan penelitian sebelumnya adalah penelitian sebelumnya lebih berfokus pada bagiamana implementasi metode *Agile* dalam proses pengembangan *API* sementara penelitian ini berfokus pada pengembangan *API* untuk suatu sistem dengan menggunakan metode pengembangan *Agile*.
4. Dalam penelitian (Sabila, Praptono, dan Arini 2021) yang berjudul "**PERANCANGAN APLIKASI PENCATATAN LAPORAN KEUANGAN DENGAN MENGGUNAKAN METODE *AGILE DEVELOPMENT SCRUM***" ISSN: 2527-3116. Dapat disimpulkan bahwa pengelolaan keuangan pada kedai Intikopi masih memiliki kekurangan yaitu laporan yang dihasilkan dari metode pencatatan manual masih tidak sama dengan hasil yang diperoleh

karena tidak terstruktur sehingga dibuatlah aplikasi pencatatan keuangan yang dapat mengurangi kesalahan pencatatan keuangan serta dapat membantu memahami kondisi keuangan perusahaan dengan menggunakan PHP dan MySQL dengan metode *Agile Development Scrum,* hasilnya adalah aplikasi dapat menginput penghasilan dan pengeluaran perusahaan yang juga memudahkan perusahaan untuk mengetahui laporan keuangan secara langsung tanpa harus dibuat terlebih dahulu secara manual. Perbedaan antara penelitian ini dan penelitian sebelumnya terletak pada aplikasi keuangan yang dibuat untuk kedai IntiKopi tidak menggunakan *API* yang mana hal tersebut membuat pengembangan aplikasi menjadi lebih rumit jika suatu saat memerlukan fitur tambahan atau laporan keuangan yang lain. Selain itu aplikasi pencatatan laporan keuangan yang dibangun pada penelitian sebelumnya menggunakan bahasa pemrograman yang berbeda yaitu PHP. Sementara itu, penelitian ini menggunakan bahasa pemrograman Javascript dengan *runtime* Node.js.

5. Dalam penelitian (Kusumaningrum dkk. 2022) yang berjudul "**Pemanfaatan *Restful API* Pada *Mobile Based Test* Untuk Sertifikasi Karyawan**" ISSN : 2580-2291. Dijelaskan bahwa *test* merupakan sesuatu yang digunakan untuk menguji mutu kecerdasaan, hasil belajar, atau kemampuan seseorang. Tes sering kali dikaitkan dengan evaluasi tentang penguasaan materi seseorang yang hasilnya digunakan untuk membuat keputusan kelulusan seseorang dalam materi tersebut, namun tes berbasis kertas dinilai tidak efektif dan efisien karena memakan dalam mengoreksi hasilnya sehingga dibuatlah aplikasi *mobile test* yang menggunakan *REST API* dengan metode pengembangan *waterfall*,

hasilnya menunjukkan pemanfaatan *REST API* pada aplikasi *Mobile Test* yang dibuat berjalan dengan baik yang dapat dilihat dari hasil pengujian fungsionalitas yang didapatkan yaitu 100%. Perbedaan antara penelitian ini dan penelitan sebelumnya adalah metode pengembangan yang digunakan dalam membangun *REST API* dan aplikasi dimana penelitian sebelumnya menggunakan metode *waterfall* yang memakan waktu lebih lama dibandingkan dengan metode *agile* yang digunakan pada penelitian ini, selain itu dengan metode *agile* pengembangan dapat lebih sesuai dengan kebutuhan pengguna karena dalam pengembangan *REST API* banyak melibatkan pemangku kepentingan.

6. Dalam penlitian (Azkarin, Guntara, dan Herdiana 2023) yang berjudul "***Development of a REST API for Human Resource Information System for Employee Referral Management Domain Using the Express JS Framework and Node.js***" ISSN : 2962-6110. Dijelaskan bahwa persaingan di pasar kerja yang sangat kompetitif membuat berinovasi dan membuat strategi untuk mendapatkan kandidat potensial yang memiliki keterampilan yang tepat dan cocok dengan budaya perusahaan, cara yang digunakan adalah dengan pemanfaatan rujukan karyawan (*employee referral*). Atas dasar hal tersebut maka dibuatlah *REST API* untuk sistem informasi sumber daya manusia (HRIS) menggunakan Express.js dan Node.js yang dibangun menggunakan metode *waterfall.* Hasilnya adalah *REST API* dibangun menggunakan Express.js dan Node.js menunjukan hasil yang valid untuk setiap *request* dan *response* yang diujikan di setiap *endpoint API*. Pembangunan *REST API* pada penelitian ini dan penelitian sebelumnya sama-sama menggunakan *framework* Express.js dan

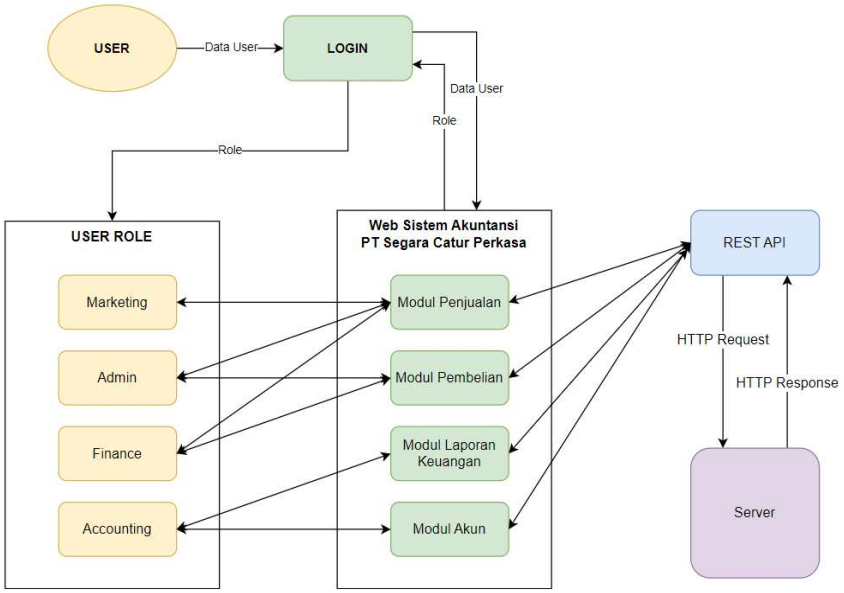
Node.js akan tetapi penelitian sebelumnya menggunakan metode *waterfall* yang kurang adaptif terhadap kebutuhan pemangku kepentingan, metode ini juga memerlukan waktu yang cenderung lama dalam pengembangannya, selain itu metode ini juga memiliki jangka waktu yang ketat dalam implementasinya, tidak seperti metode *agile* yang lebih singkat namun dapat selalu disesuaikan dengan kebutuhan pemangku kepentingan.

7. Dalam penelitian (Efuntade dan Efuntade 2023) yang berjudul "***Application Programming Interface (API) And Management of Web-Based Accounting Information System (AIS): Security of Transaction Processing System, General Ledger and Financial Reporting System*"** ISSN: 2695-2211. Dijelaskan bahwa *API* memungkinkan bisnis untuk membuat data dan fungsionalitas aplikasi dapat digunakan oleh pihak ketiga, mitra komersial, maupun departemen internal organisasi, *API* juga memungkinkan interaksi antara layanan dan produk beserta informasi dan fitur satu sama lain. Hal tersebut mendukung pengembangan aplikasi *web* terutama jika digabungankan dengan AIS (*Accounting Information System*) sehingga dibuatlah *API* untuk sistem informasi akuntansi sesuai dengan persyaratan IFRS (International Financial Reporting), hasilnya aplikasi dapat digunakan untuk melakukan pencatatan, pengelompokan, mengolah, dan menyajikan data transaksi keuangan yang kemudian menjadi bahan pengambilan keputusan aplikasi ini membutuhkan seorang *Admin* untuk mengelola setiap data penting dan *API* yang dibangun digunakan sebagai perantara antara aplikasi yang dibuat dengan pengembang eksternal. Dapat simpulkan bahwa perbedaan penelitian sebelumnya dengan

penelitian ini terletak pada metode pengembangan yang digunakan, pada penelitian sebelumnya tidak dijelaskan secara spesifik metode pengembangan apa yang digunakan namun lebih berfokus pada penjelasan tentang aspek-aspek penting dalam *API* yang dibangun dan beberapa aspek laporan akuntansi yang sesuai persyaratan IFRS.

2.5 Kerangka Pemikiran

Kerangka pikir adalah kerangka konseptual atau struktur dasar yang digunakan untuk merumuskan suatu masalah atau riset. Kerangka ini menjadi referensi yang membantu peneliti untuk memahami, merancang, dan mengorganisasi informasi serta ide-ide terkait dengan masalah atau topik dalam penelitian ini, berikut adalah visualisasi dari kerangka berpikir pada penelitian ini:



Gambar 2.9 Kerangka Pikir

Sumber : Data olahan peneliti (2023)

Dari kerangka pikir di atas dapat dilihat bahwa setelah *login* berhasil, *user* dapat mengakses modul yang tersedia berdasarkan *role* yang dimiliki, *role user* adalah peran atau tanggung jawab yang diemban oleh pengguna di dalam suatu sistem. Begitu pula hak aksesnya, hak akses *user* disesuaikan dengan perannya. Modul akan menampilkan menu-menu berdasarkan *role user* yang mengaksesnya, ketika *user* mengakses salah satu menu, maka menu tersebut akan melakukan *request* ke *API*. *API* akan meneruskan *request* ke *server* kemudian *server* akan memberikan respon atas permintaan dari *user* yang diteruskan kembali melalui *API* dan kemudian ditampilkan pada menu yang diakses. Setiap *role* diberi hak akses yang berbeda, adapun *role user* pada sistem yang dirancang adalah sebagai berikut:

1. *Role Marketing*

   Role Marketing adalah *role* yang hanya memiliki hak akses untuk modul Penjualan dan hanya dapat melihat, merubah dan membuat penawaran.

2. *Role Admin*

   Role Admin adalah *role* yang dapat mengakses modul Penjualan dan Pembelian, *Admin* berperan sebagai *user* yang dapat melihat *quotation* sebagai acuan untuk kegiatan operasional di lapangan. *Admin* juga dapat mengakses menu Tagihan dan Kas Kecil untuk membuat dan mencatat pembelian.

3. Role *Finance*

   Role Finance adalah *role* yang dapat melakukan *approval* (persetujuan) atas penawaran dan pengajuan isi ulang kas kecil oleh *Admin*, *user* dengan *role Finance* juga dapat melakukan pembelian pada menu Tagihan.

4. Role *Accounting*

   Role Accounting adalah *role* yang bertugas melakukan *monitoring* keuangan perusahaan melalui modul Laporan Keuangan, *user* dengan *role Accounting* juga berhak mengelola daftar akun perusahaan melalui modul Akun.

Modul adalah kumpulan menu yang dapat diakses oleh *user* dengan *role* tertentu. Menu yang ada pada modul adalah antarmuka yang menghubungkan *user* dengan *server* dan *database* yang komunikasinya dilakukan dengan *API*. Adapun rincian dari modul adalah sebagai berikut:

1. Modul Penjualan

   Modul ini berisi menu yang berkaitan dengan pendapatan perusahaan baik secara langsung maupun tak langsung seperti Penawaran, Faktur dan Pendapatan Kerjasama.

2. Modul Pembelian

   Modul ini berisi menu yang berkaitan dengan pengeluaran dan utang perusahaan seperti: Tagihan dan Kas Kecil.

3. Modul Laporan Keuangan

   Modul ini berisi laporan keuangan.

4. Modul Akun

   Modul ini berisi menu manajemen akun yang digunakan untuk mengolah daftar akun (*Chart of Account / COA*) perusahaan.