BAB II

TINJAUAN PUSTAKA

2.1 Perancangan

Soetam Rizky (2011) Mendefinisikan bahwa “Perancangan adalah sebuah proses untuk mendefinisikan sesuatu yang akan dikerjakan dengan mengunakanteknik yang bervariasi serta di dalam nya melibatkan deskripsi mengenai arsitektur serta detail mengenai komponen dan juga keterbatasan yang akan dialami dalam proses pengerjaan nya” (Andrian 2021).

Menurut Nataniel Dengen dan Heliza Rahmania Hatta (2009), perancangan didefinisikan sebagai proses aplikasi berbagai teknik dan prinsip bagi tujuan pendefinisian suatu perangkat, suatu proses atau *sistem* dalam detail yang memadai untuk memungkinkan realisasi fisiknya. Untuk mengendalikan proses desain, A. Davis mengusulkan serangkaian prinsip-prinsip dasar dalam perancangan sebagai berikut (Alfia 2020):

a. Desain tidak boleh menderita karena tunnel vision (visi terowongan).
b. Desain tidak boleh berulang.
c. Desain harus terstruktur untuk mengakomodasi perubahan.
d. Desain harus terstruktur untuk berdegradasi dengan baik, bahkan pada saat data dan event-event (kejadian-kejadian) menyimpang atau menghadapi kondisi operasi.
e. Desain bukan pengkodean dan pengkodean bukanlah desain.
f. Desain harus dinilai kualitasnya pada saat desain dibuat, bahkan setelah jadi.

g. Desain harus dikaji untuk meminimalkan kesalahan-kesalahan konseptual (semantik).

2.2 Aplikasi

Menurut Jogiyanto HM (dalam suhartini (2017), aplikasi merupakan penerapan, menyimpan sesuatu hal, data, permasalahan, pekerjaan ke dalam suatu sarana atau media yang dapat digunakan untuk diterapkan menjadi sebuah bentuk yang baru (Firdaus and Bakti 2024). Aplikasi adalah program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan. Pengertian aplikasi secara umum adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya, aplikasi merupakan suatu perangkat komputer yang siap pakai bagi user (Faradika, Zulfahmi, and Astri 2020).

Menurut Asropudin (2013:6), Aplikasi adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Ms.World*, *Ms.Excel* (Maiyana 2018). Aplikasi merupakan penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output* (Suharyanto 2022).

Aplikasi *software* yang dirancang untuk suatu tugas khusus dapat dibedakan menjadi dua jenis, yaitu (Suharyanto 2022):

a. Aplikasi software spesialis, program dengan dokumentasi tergabung yang dijalankan untuk menjalankan tugas tertentu.
b. Aplikasi software paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

2.3 Unified Modeling Language

Unified Modeling Language atau yang biasa dikenal dengan singkatan UML, adalah bahasa standar yang digunakan untuk menggambarkan, menjelaskan, dan membangun perangkat lunak. Ini merupakan metode dalam pengembangan *sistem* berbasis objek yang juga berfungsi sebagai alat bantu dalam proses pengembangan *sistem*. Beberapa alat bantu yang digunakan dalam perancangan berbasis objek dengan menggunakan UML meliputi diagram use case, activity diagram, sequence diagram, dan class diagram (Binangkit, Voutama, and Heryana 2023). Unified Model Language (UML) adalah bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks pendukung (Maidiansyah, Sularno, and Faradika n.d.). UML juga dapat didefinisikan sebagai sebuah standarisasi bahasa pemodelan untuk membangun perangkat lunak yang dibangun menggunakan teknik pemrograman berorientasi objek (Firdaus and Bakti 2024).

2.3.1 Pengenalan Unified Modeling Language (UML)

UML (*Unified Modelling Language*) adalah suatu metode dalam pemodelan secara visual yang digunakan sebagai sarana perancangan *sistem* berorientasi objek.

Diagram UML yang sering digunakan adalah *Use Case Diagram, Activity Diagram, Sequence Diagram, Class Diagram, Statemachine Diagram dan Component Diagram*(Marthiawati et al. 2024).

Unified Modeling Language (UML) pada saat analisis kebutuhan digunakan untuk visualisasi, menentukan ruang lingkup, dan mendokumentasikan artefak *sistem* secara efektif yang bermanfaat untuk berbagai pemangku kepentingan suatu aplikasi. Saat ini, sebagian besar para perancang *sistem* informasi dalam menggambarkan informasinya, memanfaatkan UML diagram dengan tujuan utama untuk membantu tim proyek berkomunikasi, mengeksplorasi potensi desain, dan melakukan validasi desain arsitektur perangkat lunak atau pembuatan program aplikasi (Marthiawati et al. 2024).

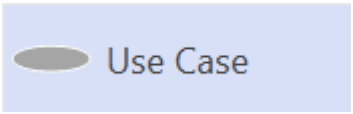
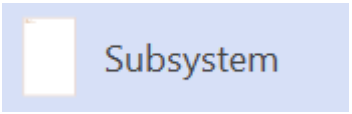
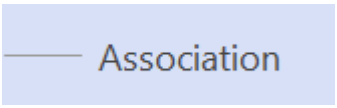
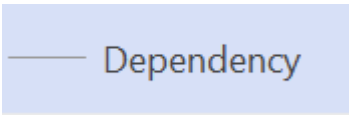
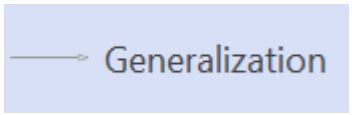
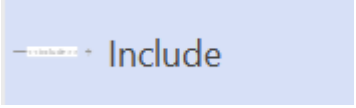
2.3.2 Jenis Jenis Diagram *Unified Modeling Language* (UML)

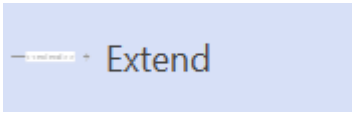
Kategori-kategori diagram UML mencakup berbagai aspek yang berbeda dari perancangan dan pengembangan perangkat lunak. Di bawah ini adalah beberapa jenis diagram UML yang umum digunakan untuk memodelkan *sistem* perangkat lunak:

a. *Use Case* Diagram

Diagram *Use Case* adalah pemodelan untuk perilaku *sistem* informasi yang akan dibuat. *Use case* bekerja dengan mendeskripsikan interaksi tipikal antara pengguna *sistem* dan *sistem* itu sendiri melalui sebuah cerita tentang bagaimana *sistem* tersebut digunakan. *Use case* atau diagram *use case* adalah pemodelan untuk perilaku (*behavior*) *sistem* informasi yang akan dikembangkan (Mahardika, Merani, and Suseno 2024).

Tabel 2.1 ***Use Case Diagram***

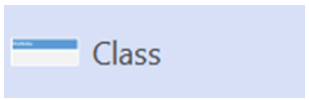
No	**Nama**	**Keterangan**
1		Mewakili pengguna atau entitas eksternal yang berinteraksi dengan *sistem*.
2		Sebuah unit fungsional yang koheren disediakan oleh sebuah *sistem* atau kelas.
3		Komponen *sistem* yang mungkin berisi beberapa kasus penggunaan.
4		Representasi partisipasi seorang aktor dalam sebuah kasus penggunaan.
5		Menunjukkan bahwa satu use case mempunyai ketergantungan pada use case lainnya.
6		Menunjukkan bahwa use case adalah cara spesifik untuk mencapai tujuan use case umum.
7		Menunjukkan bagaimana use case dipecah menjadi langkah-langkah

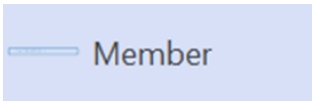
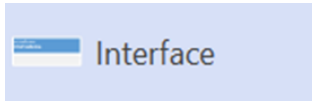
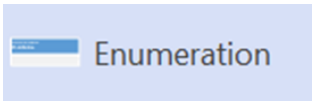
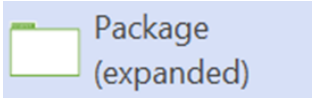
No	Nama	Keterangan
		yang lebih kecil.
8		Menunjukkan bahwa satu use case menambahkan fungsionalitas ke use case lainnya.

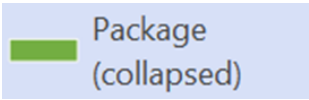
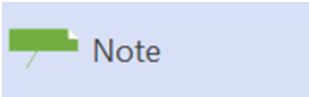
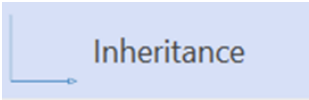
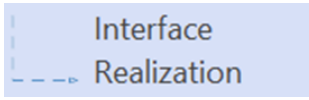
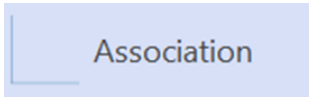
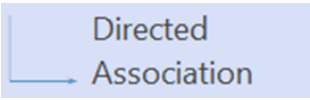
b. *Class* Diagram

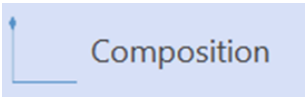
Class diagram adalah sebuah diagram yang dibuat dengan tujuan menyinkronkan antara dokumentasi perancangan dan perangkat lunak (Mahardika, Merani, and Suseno 2024). Agar *engineer* dapat membuat kelas-kelas sesuai rancangan diagram, maka kelas *rule* pada struktur *sistem* harus mampu melakukan fungsi sesuai kebutuhan *sistem*. Diagram kelas atau class diagram menggambarkan struktur *sistem* dari segini pendefinisian kelas-kelas yang akan dibuat untuk membangun *sistem* (Niqotaini et al. 2023).

Tabel 2.2 *Use Case Diagram*

No	Nama	Keterangan
1		Menjelaskan sekumpulan objek dengan struktur, perilaku, dan hubungan yang serupa. Nama suatu kelas harus unik dalam

No	Nama	Keterangan
		paketnya. Bentuk kelas dimulai sebagai persegi panjang dengan tiga baris. Nama kelas ada di baris paling atas. Dua baris lainnya adalah untuk metode atau operasi yang mungkin digunakan oleh kelas.
2		Menjelaskan atribut atau operasi.
3		Digunakan dalam bentuk kelas untuk memisahkan *operasi* dari *atribut* .
4		Menentukan operasi kelas, komponen, paket, atau elemen lain yang terlihat secara eksternal tanpa menentukan struktur internal.
5		Menjelaskan tipe data yang terdiri dari sekumpulan nilai bernama.
6		Mewakili elemen pengorganisasian dasar model UML. Ini menyediakan namespace untuk elemen yang dikelompokkan. Setiap elemen hanya dimiliki oleh satu paket, dan satu paket dapat disarangkan ke paket

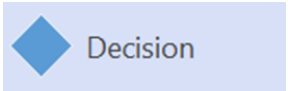
No	Nama	Keterangan
		lainnya.
7		Mewakili sebuah paket dalam suatu proses.
8		Digunakan sebagai komentar diagram yang tidak memiliki pengaruh semantik pada elemen model.
9		Menunjukkan bahwa tipe sumber mewarisi dari tipe target.
10		Menunjukkan bahwa tipe sumber merealisasikan antarmuka target.
11		Mewakili hubungan umum antara instance kelas.
12		Mewakili hubungan yang mengalir hanya dalam satu arah antar instance kelas.
13		Menandakan bahwa objek di ujung yang berbentuk berlian mengandung referensi ke objek di ujung lainnya. Jika berisi objek tersebut secara eksklusif, gunakan

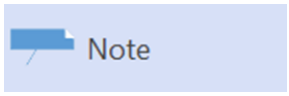
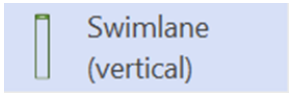
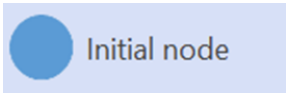
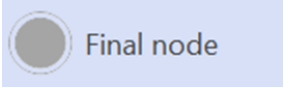
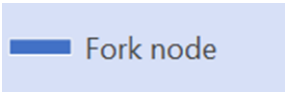
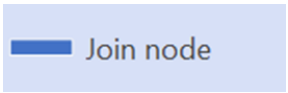
No	Nama	Keterangan
		bentuk Komposisi sebagai gantinya.
14		Menunjukkan bahwa tipe sumber bergantung pada tipe target.
15		Menunjukkan bahwa tipe sumber memiliki bagian dari tipe target.

c. *Activity* Diagram

Menggambarkan alur logika dari suatu proses atau aktivitas di dalam sistem. Diagram aktivitas menggunakan berbagai notasi, seperti tindakan (action), keputusan (decision), garis aliran (flow), dan banyak lagi, untuk menggambarkan langkah-langkah yang diperlukan dalam sebuah aktivitas. *Activity diagram* merupakan diagram aliran kerja atau juga dapat disebut dengan *work flow* yang merupakan aktivitas dari sebuah proses bisnis yang terdapat pada perangkat lunak (Sudipa et al. 2023).

Tabel 2.3 *Activity Diagram*

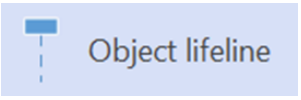
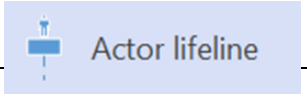
No	Nama	Keterangan
1		Mewakili sebuah keputusan. Harus memiliki setidaknya dua jalur yang bercabang darinya.

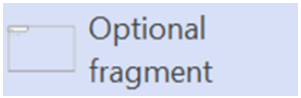
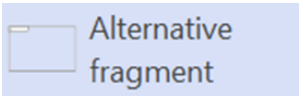
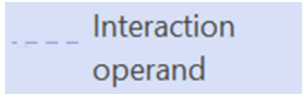
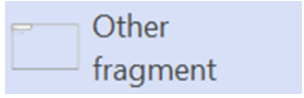
No	Nama	Keterangan
2		Digunakan sebagai komentar diagram yang tidak memiliki pengaruh semantik pada elemen model.
3		Membedakan tanggung jawab untuk sub-proses dalam diagram.
4		Merupakan langkah dalam suatu proses.
5		Menunjukkan akhir dari perilaku kondisional yang dimulai pada simpul keputusan sebelumnya. Ini menyebabkan aliran menyatu.
6		Mewakili *node* kontrol di mana aliran dimulai ketika aktivitas dimulai.
7		Mewakili *node* kontrol yang mengakhiri aliran.
8		Merupakan awal dari aktivitas paralel.
9		Merupakan akhir dari aktivitas paralel.

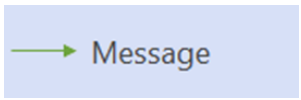
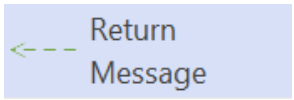
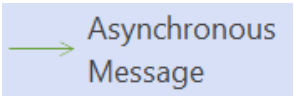
d. *Sequence* Diagram

Sequence diagram menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Gambaran sequence diagram dibuat minimal sebnayak pendefinisan use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada sequence diagram sehingga semakin banyak use case yang didefinisikan, maka sequence diagram yang harus dibuat juga semakin banyak (Maesaroh et al. 2024). Sequence diagram merupakan UML yang menggambarkan interaksi antar objek di dalam dan disekitar sistem, termasuk pengguna, display, dan sebagainya berupa message yang digambarkan terhadap waktu (Maesaroh et al. 2024).

Tabel 2.4 *Sequence Diagram*

No	Nama	Keterangan
1		Mewakili periode ketika peserta melakukan operasi.
2		Mewakili suatu objek atau komponen. Garis vertikal mewakili urutan peristiwa yang terjadi selama interaksi, seiring berjalannya waktu.
3		Mewakili peserta yang berada di luar sistem. Garis vertikal mewakili urutan peristiwa

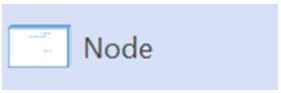
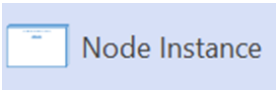
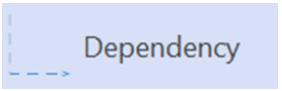
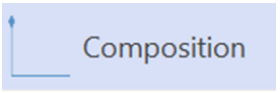
No	Nama	Keterangan
		yang terjadi selama interaksi, seiring berjalannya waktu.
4		Menunjukkan sebuah lingkaran. Anda dapat menentukan kondisi di mana hal itu harus diulang.
5		Meliputi urutan yang mungkin atau mungkin tidak terjadi. Anda dapat menentukan kondisi terjadinya.
6		Memodelkan konstruksi if/then/else. Setelah menambahkannya ke halaman, Anda dapat mengklik kanan untuk memasukkan operan interaksi tambahan.
7		Memodelkan satu kondisi dalam konstruksi if/then/else. Tambahkan bentuk ini ke fragmen Alternatif .
8		Digunakan untuk mewakili tipe fragmen lain, seperti Break, Par, Critical, Seq, atau Strict.

No	Nama	Keterangan
9		Merupakan komunikasi antar objek yang menyampaikan informasi dan menghasilkan suatu tindakan.
10	  	Pesan meluas dari garis hidup suatu objek ke garis hidup objek lainnya, kecuali dalam kasus Self Message (Pesan Diri) , dalam hal ini pesan dimulai dan diakhiri pada garis hidup yang sama.

e. *Deployment* Diagram

Diagram *deployment* atau *deployment* diagram menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Sistem terdistrubusi murni dan rekayasa ulang aplikasi (Rachmad et al. 2023). *Deployment* diagram menentukan komponen-komponen yang akan di deploy (install) pada *node-node* termasuk menentukan perangkat lunak dan perangkat keras yang dibutuhkan dalam *node-node* tersebut. Gunakan diagram penerapan UML untuk menunjukkan arsitektur penerapan artefak perangkat lunak ke *node* (Rachmad et al. 2023).

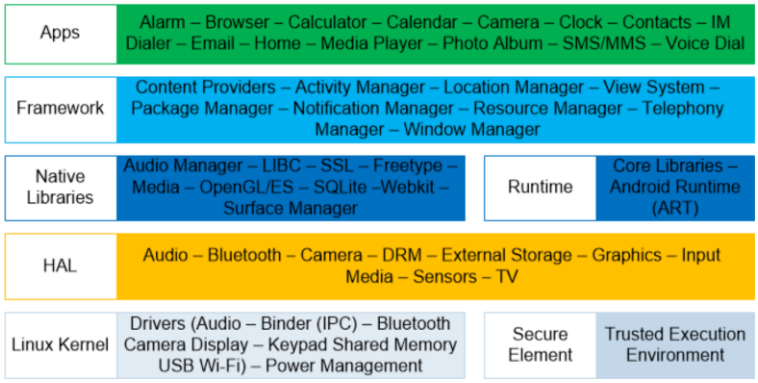
Tabel 2.5 *Deployment Diagram*

No	Nama	Keterangan
1		Menentukan perangkat komputasi atau fisik run-time.
2		Menentukan instance perangkat komputasi atau fisik run-time.
3		Menentukan artefak.
4		Menunjukkan bahwa tipe sumber bergantung pada tipe target. Panah menunjuk pada jenis target (atau item yang bergantung pada item lainnya).
5		Menandakan bahwa objek di ujung yang berbentuk berlian mengandung referensi ke objek di ujung lainnya. Jika berisi objek tersebut secara eksklusif, gunakan bentuk Komposisi sebagai gantinya.
6		Menunjukkan bahwa tipe sumber memiliki bagian dari tipe target.

2.4 Android

Aplikasi *Android* adalah sebuah aplikasi yang berbasis *Android* yang disusun dan dirancang dengan menggunakan bahasa pemrograman. Bahasa pemrogramannya diantaraanya *Kotlin, Java, C++* Dan lain sebagainnya (Wicaksono, Putra, and Hikmahwan 2022). Aplikasi berbasis *Android* dapat di install melalui Playstore, APKPure, dan lain sebagainya. Android adalah aplikasi sistem operasi untuk telepon seluler yang berbasis linux. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak (Faradika, Zulfahmi, and Astri 2020).

Android merupakan sistem operasi untuk perangkat *mobile* seperti *tablet* dan *smartphone* yang dikembangkan oleh *Google* dan *Open Handset Alliance* berbasis pada *kernel Linux* yang telah dimodifikasi dan dikembangkan terutama untuk peralatan dengan antarmuka (*interface*) layar sentuh (*touch screen*) (Prasetio and Wellem 2022). Saat ini versi *Android* terbaru adalah *Android* 12 dengan kernel versi 5 dan API Level 32. Terdapat lima lapisan (*layer*) pada sistem operasi *Android*, yaitu: 1) *Applications*, 2) *Framework*, 3) *Native Libraries* dan *Runtime* (berada pada layer yang sama), 4) *Hardware Abstraction Layer* (HAL), dan 5) *Linux Kernel*. Gambar 2.1 menunjukkan lima layer pada *Android* OS (Prasetio and Wellem 2022).



Gambar 2.1 *Android OS stack*

2.4.1 Android Studio

Merupakan IDE yang digunakan untuk mengembangkan aplikasi dengan basis android. Android Studio memiliki fitur yang ditawarkan diantaranya menggunakan Gradle yang fleksibel untuk sistem build, emulator yang dapat dijalankan dengan cepat dan banyak jenis, framework serta alat untuk uji yang lengkap, didukung dari C++, NDK dan masih banyak lagi (Wicaksono, Putra, and Hikmahwan 2022).

truktur projek di Android Studio memiliki satu atau beberapa modul. Untuk modulnya meliputi Modul aplikasi Android, Modul Google App Engine dan Modul library. Pada satu aplikasi berisikan folder yang isinya seperti manifest, java dan res. Folder Manifest di dalamnya terdapat file AndroidManifest.xml. Folder Java berisikan tentang file file .java, termasuk kode pengujian JUnit. Folder Res berisikan tentang semua resource (Wicaksono, Putra, and Hikmahwan 2022).

2.5 Bahasa Pemrograman

Bahasa pemrograman merupakan sebuah alat komunikasi antara manusia dengan perangkat komputer. Pembelajaran bahasa pemrograman dapat dilakukan melalui pendidikan formal maupun informal seperti lembaga kursus atau pembelajaran secara online layaknya bahasa manusia, bahasa itu juga memiliki tata tulis (syntax) dan aturan tertentu, bahasa pemrograman memfasilitasi cara dan aturan yang dilakukan oleh programmer untuk menuliskan perintah menyimpan, mengkompilasi dan melihat hasil secara benar. (Pebriyanti, Saptarini, and Suardani 2023).

2.6 Java

Java adalah sebuah bahasa pemrograman *scripting* yang sering digunakan dalam pembuatan aplikasi berbasis *handphone* dan juga dapat digunakan untuk menyediakan akses sebuah objek yang akan disisipkan di aplikasi. *Java* juga berfungsi sebagai penambah tingkah laku agar widget dapat tampil lebih atraktif (Dagha 2021).

Java adalah bahasa pemrograman umum, seperti *Python* atau *JavaScript* bahasa itu sendiri secara khusus merupakan bahasa pemrograman berorientasi objek, sehingga memiliki kemiripan dengan *C++, C#. Java* juga merupakan *platform*, yang berarti bahwa kode *Java* dapat berjalan di mesin apa pun yang memiliki *Java Virtual Machine* (JVM) di dalamnya. Awalnya kedua hal itu dapat dipertukarkan, dan satu-satunya yang berjalan di JVM adalah *Java*. Namun sejak saat itu, beberapa bahasa telah ditulis yang dapat berjalan di *platform Java*, seperti *Scala*, *Groovy*,

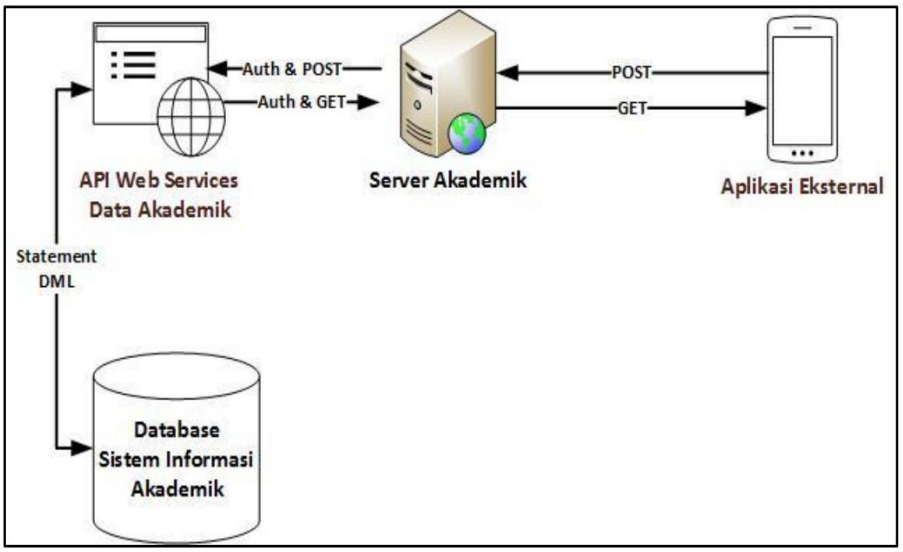
implementasi *Ruby* yang disebut *jRuby*, dan implementasi *Python* yang disebut *Jython* (Wicaksono, Putra, and Hikmahwan 2022).

2.7 Application Programming Interface (API)

API adalah antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program (Matalarens and Setyowatie 2023). Api memungkinkan pengembang untuk memakai fungsi yang sudah ada dari aplikasi lain . Dengan menggunakan teknik API, maka pengembang aplikasi lain dapat menggunakan *Database* tanpa harus terkoneksi secara langsung (Matalarens and Setyowatie 2023).

Dalam aplikasi berbasis web, pemanggilan api menggunakan protokol *Hyper Text Transfer Protocol* (HTTP) melalui alamat tertentu dan akan memberikan respon data yang dapat berupa *Hyper text markup language* (html), *Javascript Object Notation* (JSON) ataupun *Extensible Markup Language* XML (Tedyyana et al. 2022).

Dalam penelitian ini penulis menggunakan arsitektur REpresentational State Transfer (REST). Arsitektur REST merupakan model desain web servis yang dominan digunakan saat ini yang biasa disebut dengan RESTful (Tedyyana et al. 2022).



Gambar 2.2 ***Arsitektur Web Services***

2.8 Database

Database digunakan untuk menyimpan informasi dalam berbagai format, seperti teks, angka, gambar, dan lainnya. Informasi dalam *Database* disimpan dalam tabel yang terdiri dari baris dan kolom, mirip dengan *spreadsheet* (Indarta et al. 2021). Basis data atau *database* merupakan beberapa file yang dikumpulkan dan saling berelasi dengan ditunjukan kunci dari tiap file yang terkait. Sebuah *Database* menggambarkan kumpulan data yang digunakan dalam sebuah lingkup informasi (HERLINA and ASSIDIQ 2021).

2.8.1 Konsep Database

Dalam satu file terdapat record-record yang sejenis dan merupakan kumpulan entitas yang seragam. Pada basis data dapat didefinisikan beberapa atribut yang sangat terkait dengan database, diantaranya (Dirgantara et al. 2023).

a. Entity : Merupakan suatu objek baik itu orang, tempat atau konsep yang informasinya direkam pada suatu database. Misalnya entity kemacetan, kecelakaan, dan lainya.
b. Database : Merupakan kumpulan dari field-field yang mempunyai hubungan dengan field lain sehingga menghasilkan bangunan data untuk informasi yang dihasilkan.
c. File : Merupakan kumpulan record sejenis dengan panjang elemen dan atribut yang sama namun berbeda data.
d. Record : Kumpulan elemen yang saling terkait yang sepenuhnya terkait dengan informasi entitas, di mana satu catatan mewakili seluruh data.

2.8.2 Kegunaan Database

Ada beberapa kegunaan dari basis data, yang dapat di lihat sebagai berikut (Indarta et al. 2021):

a. Redundansi dan *inkonsistensi* data : Jika file –file dan program aplikasi yang diciptakan beberapa programmer dalam waktu yang berselang cukup panjang, maka ada beberapabagian data yang mengalami panggandaan pada file-file yang berbeda. Penyimpanan data yang berulang ulang dibeberapa file juga dapat mengakibatkan inkonsistensi (tidak konsisten).
b. Sulit mengakses data : Meskipun belum menulis program untuk mengekstrak data, Anda masih perlu mencetak data siapa pun sekaligus, tetapi kesulitan ini muncul. Untuk tujuan ini, masalah dapat diselesaikan dengan memimpin

sistem manajemen basis data yang mengambil data dalam bahasa yang akrab dan mudah digunakan secara terus menerus.

c. Isolasi data untuk standarisasi : Jika data tersebar di beberapa file dalam format yang berbeda, maka kesulitan menulis aplikasi untuk mengambil dan menyimpan data akan muncul. Kemudian data dalam database harus dibuat dalam format agar lebih mudah dalam pembuatan aplikasi.
d. Masalah keamanan atau security : Tidak semua pengguna sistem database dapat mengakses semua data. Misal, data tentang gaji pegawai hanya bisa diakses oleh bagian keuangan perusahaan. Keamanan ini dapat diatur oleh program yang dibuat oleh programmer atau fasilitas keamanan sistem operasi.
e. Masalah integrasi (kesatuan) : Basis data berisi file terkait. Oleh karena itu, ada beberapa masalah besar, seperti bagaimana tautan di antara file-file ini terjadi. Meskipun diketahui bahwa file A terkait dengan file B. Namun, secara teknis, ada file kunci yang menghubungkan kedua file tersebut.
f. Masalah data *independence* (Kebebasan Data) : Program aplikasi ditulis dalam bahasa yang dibuat oleh sistem manajemen database, setiap kali Anda ingin melihat data, apa yang terjadi pada struktur file, cukup menggunakan utilitas USE, anda hanya perlu menggunakan APPEND untuk menambahkan data, yang mana artinya database manajemen berasal dari database yang Dirilis. Setiap perubahan dalam database, semua pesanan akan tetap stabil, tidak perlu diubah.

2.9 Booking

Menurut Sulaeman, Dkk (2020), *Booking* (Pemesanan) berasal dari bahasa Inggris yaitu *to reserve* yang dapat di artikan proses perjanjian berupa pemesanan produk barang ataupun jasa namun belum ditutup oleh suatu pembelian. Dalam kamus besar Bahasa Indonesia *reservasi* atau pemesanan adalah proses, pembuatan, cara memesan (tempat, barang dan sebagainya) kepada orang lain (Diantara, Siswanto, and Yupianti 2022).