# LANDASAN TEORI

Teori yang digunakan sebagai landasan dalam penelitian ini meliputi pengertian absensi, *flutter*, location based service, *Application Programming Interface (API)*, *database*, *sql*, *unified modeling language (UML)* dan *metode waterfall*.

## Absensi

Pengertian absensi adalah ketidakhadiran karyawan saat yang bersangkutan dijadwalkan bekerja. Jumlah absesn kerja dalam perusahaan menggambarkan pertukaran benefit antara karyawan dan perusahaan yang menggaji. Tentu saja, perusahaan tidak ingin rugi akibat tingginya absen karyawan.

Ada tiga ukuran absen, yaitu hilangnya waktu kerja, frekuensi atau tingkat keseringan absen, dan jumlah dalam jangka pendek. Hilangnya waktu kerja adalah jumlah total jam atau hari kerja yang hilang selama setahun. Frekuensi adalah jumlah total terjadnya absen dalam setahun, tanpa memandang lama jangka waktu. Absen dalam jangka pendek adalah jumlah absen 1 – 2 hari kerja selama setahun.[1]

## *Flutter*

Flutter adalah SDK untuk pengembangan aplikasi mobile dengan kinerja tinggi, aplikasi untuk iOS dan Android, dari satu codebase (basis kode) yang di buat oleh Google dengan lisensi open source. Tujuannya adalah memungkinkan pengembang untuk menghadirkan aplikasi berkinerja tinggi yang terasa alami pada platform yang berbeda.

Untuk membuat aplikasi Flutter, diperlukan untuk mengerti bahasa Dart. Dart merupakan bahasa pemrograman yang dibuat oleh Google untuk menggantikan Javascript. Dart menggunakan static typing yang berarti sebelum memakai variabel, variabel perlu didefinisikan terlebih dahulu. Dart bisa berjalan pada semua perangkat juga, pada web Dart memakai dart2js yang artinya Dart diubah ke Javascript agar bisa dimengerti browser. Pada perangkat desktop Dart memakai dart2aot yang mengubah Dart menjadi bahasa mesin. Pada perangkat mobile Dart memakai Flutter. Syntax pada bahasa Dart sangat mudah dipelajari, karena Dart memiliki kemiripan dengan syntax- syntax pemrograman lain, seperti Javascript dan Java.

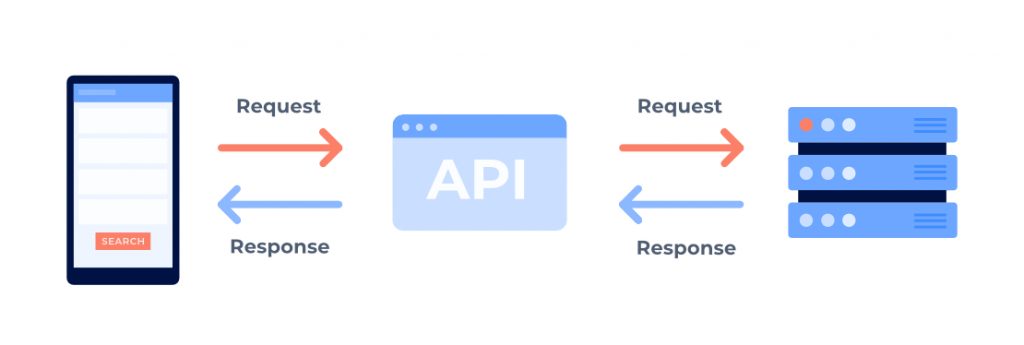
Dart bisa melakukan pemrograman Asynchronous, yaitu fungsi yang non-blocking yaitu memungkinkan program untuk menjalankan kode lain sambil menunggu fungsi asynchronous selesai. Contoh penggunaan sehari-hari adalah pada saat aplikasi sedang mengambil data dari web, aplikasi juga menampilkan halaman loading. Dart juga bisa melakukan pemrograman Reactive, yaitu pemrograman asynchronous dengan stream data. Stream data artinya datanya bisa di terima lebih dari sekali, kebalikan dengan asynchronous yang datanya hanya sekali terima saja. Pada Dart asynchronous menggunakan kata Future, reactive menggunakan Stream.[2]

## Location Based Service

LBS (Location Based Service) salah satu bentuk layanan dan didasarkan pada posisi user berada disaat ini. Kadang kala user tidak mengetahui dimana dia berada. Oleh karena itu sistem yang akan bekerja untuk membantu user menemukan posisinya saat ini. Selanjutnya setelah posisi tersebut diketahui, data tersebut dapat digunakan untuk memenuhi kebutuhan user dan memungkinkan user untuk mengakses segala informasi yang terkait dengan posisinya saat ini melalui GPS. Tingkat akurasi pada GPS lebih sering dipengaruhi oleh faktor lingkungan disekitarnya. Ketika alat GPS berada di sebuah lembah, maka tingkat akurasinya akan jauh lebih rendah daripada di puncak gunung. Koordinat lokasi bumi dengan GPS diawali dengan 2 koordinat letak bumi dari angka depan Latitude (ketinggian) dan angka belakang Longitude (panjang). Tujuannya Untuk membantu pimpinan bagian tenaga kerja mengontrol keberadaan dan posisi karyawannya dan memudahkan pimpinan mengambil sebuah keputusan dan mendisiplinkan karyawan dalam pekerjaan.[3]

## API

API atau *Application Programming Interface* adalah sebuah *interface* yang dapat menghubungkan aplikasi satu dengan aplikasi lainnya. API berperan sebagai perantara antar berbagai aplikasi berbeda, baik dalam satu platform yang sama atau lintas platform. API sendiri bisa digunakan untuk komunikasi dengan berbagai [bahasa pemrograman](https://www.niagahoster.co.id/blog/bahasa-pemrograman/" \t "_blank) yang berbeda. Hal ini memudahkan *developer*, dimana *developer* tidak perlu menyediakan semua data sendiri karena cukup mengambil data yang dibutuhkan dari platform lain melalui API.



(https://www.niagahoster.co.id/blog/api-adalah/)

## Database

NoSQL merupakan makna dari “*Not Only SQL*” merupakan *database* jenis non-relasional yang menjadi alternatif dari *database* SQL yang merupakan *database* relasional. Perbedaan dari kedua jenis database tersebut terdapat di bentuk skema, SQL memiliki skema yang kaku sedangkan *database* NoSQL memiliki bentuk skema yang lebih fleksibel dan mudah diubah tanpa menggangu sistem yang sedang berjalan.

## SQL

NoSQL merupakan makna dari “*Not Only SQL*” merupakan *database* jenis non-relasional yang menjadi alternatif dari *database* SQL yang merupakan *database* relasional. Perbedaan dari kedua jenis database tersebut terdapat di bentuk skema, SQL memiliki skema yang kaku sedangkan *database* NoSQL memiliki bentuk skema yang lebih fleksibel dan mudah diubah tanpa menggangu sistem yang sedang berjalan.

## *Unified Modeling Language* (UML)

*Unified Modelling Language* adalah sebuah teknik pengembangan sistem yang menggunakan bahasa grafis sebagai alat untuk pendokumentasian dan melakukan spesifikasi pada sistem. UML pertama kali di populerkan oleh Grady Booch dan James Rumbaugh pada tahun 1994 untuk mengkombinasikan dua metodologi terkenal yaitu Booch dan OMT, kemudian Ivar Jacobson yang menciptakan *Object Oriented Software Engineering* (OOSE) ikut bergabung. Standar UML dikelolah oleh *Object Management Group* (OMG).

1. *Use Case* Diagram

*Use Case* Diagrammerupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interkasi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Berikut adalah simbol-simbol yang ada pada diagram *Use Cas*.

Tabel 2.2 Simbol *Use Case* Diagram

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Pengertian** | **Keterangan** |
|  | *Use case* | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antara unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case* |
|  | Aktor | Orang, proses, atau lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang lain, tapi aktor belum tentu merupaka orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor |
|  | Asosiasi | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor |
| <<extend>> | Ekstensi | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemprograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan. |

Sumber: (Rosa A.S dan M. Shalahuddin, 2016)

1. *Activity* Diagram

*Activity* Diagram merupakan diagram yang menggambarkan *workflow* (alirankerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada padaperangkat lunak. Berikut adalah simbol-simbol yang ada pada *activity* diagram [4].

Tabel 2.3 Simbol *Activity* Diagram

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Pengertian** | **Keterangan** |
|  | Status awal | Status awal aktivitas sistem, sebuah diagram aktivitas  memiliki sebuah status awal |
| Aktivitas | Aktivitas | Aktivitas yang dilakukan sistem, aktivitas biasanya di awali dengan kata kerja |
|  | Percabangan | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
|  | Penggabungan | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu |
|  | *Swimlane* | Memisahkan organsasi bisnis yang bertanggung |
| State |  | jawab terhadap aktivitas yang terjadi sistem pada waktu tertentu. State dapat berubah jika ada event tertentu yang memicu perubahan tersebut |

Sumber: (Rosa A.S dan M. Shalahuddin, 2016)

1. *Sequence* Diagram

*Sequence* Diagram merupakan diagram yang menggambarkan kelakuan objek pada *Use Case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Berikut adalah simbol-simbol yang ada pada *sequence* diagram.

Tabel 2.4 Simbol *Sequence Diagram*

|  |  |
| --- | --- |
| **Simbol** | **Deskripsi** |
| Aktor  Nama actor   |  | | --- | | Nama aktor |   Atau  Tanpa waktu aktif | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi actor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor |
| Garis hidup / lifeline | Menyatakan kehidupan suatu objek |
| Objek  Nama objek: nama kelas | Menyatakan objek yang berinteraksi pesan |
| Waktu aktif | Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  1: *login*() 2: cek status *login*()    3: *open*()    Maka cek status *login*() dan *open*() dilakukan di dalam metode *login*() Aktor tidak memiliki waktu aktif |
| Pesan tipe create  <<create>> | Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat |
| Pesan tipe call  1: nama\_metode() | Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  1: nama\_metode() |

Sumber: (Rosa A.S dan M. Shalahuddin, 2016)

### *Class Diagram*

*Class* Diagram merupakan diagram yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi

Tabel 2.5 Simbol *Class* Diagram

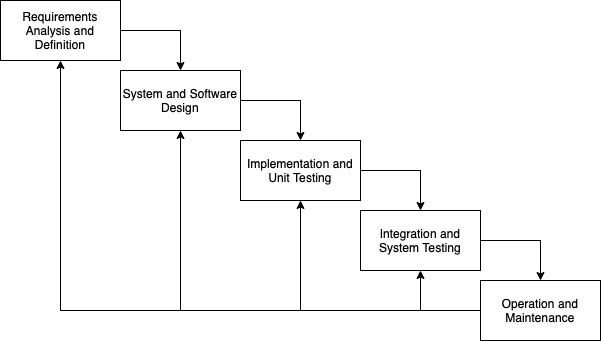
|  |  |  |
| --- | --- | --- |
| **Simbol** | **Pengertian** | **Keterangan** |
| |  | | --- | | Nama\_kelas | | +atribut | | +operasi() | | Kelas | Kelas pada struktur sistem |
|  | Antarmuka | Sama dengan konsep *interface* dalam pemprograman  berorientasi objek |
|  | Asosiasi | Relasi antarmuka dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity* |
|  | Asosiasi berarah | Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity* |
|  | Generalisasi | Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus) |
|  | Kebergantungan | Kebergantungan antar kelas |
|  | Agregasi | Relasi antarmuka dengan makna semua bagian *(whole-part)* |

Sumber: (Rosa A.S dan M. Shalahuddin, 2016)

## Metode *Waterfall*

Metode *Prototyping* adalah salah satu metode siklus hidup sistem yang didasarkan pada konsep model bekerja (working model). Tujuannya adalah mengembangkan model menjadi sistem final. Artinya sistem akan dikembangkan lebih cepat dari pada metode tradisional dan biayanya menjadi lebih rendah.

### Tahapan *Waterfall*



Gambar 2.2 Tahapan *waterfall*

(Sumber : https://medium.com/dot-intern/sdlc-metode-prototype-8f50322b14bf)

Dalam metode pengembangan *waterfall* terdapat tahapan-tahapan yang dilakukan yaitu sebagai berikut.[5]

1. Pengumpulan kebutuhan

Pelanggan dan pengembang bersama-sama mendefinisikan format dan kebutuhan keseluruhan perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

1. Membangun *prototyping*

Membangun *prototyping* dengan membuat perancangan sementara yang berpusat pada penyajian kepada pelanggan (misalnya dengan membuat *input* dan contoh *output*-nya).

1. Evaluasi *protoptyping*

Evaluasi ini dilakukan oleh pelanggan apakah *prototyping* yang sudah dibangun sudah sesuai dengan keinginan pelanggan. Jika sudah sesuai maka langkah keempat akan diambil. Jika tidak, maka *prototyping* diperbaiki dengan mengulang langkah 1, 2 , dan 3.

1. Pengkodean sistem

Dalam tahap ini *prototyping* yang sudah disepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

1. Menguji system

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan. Pengujian ini dilakukan *dengan White Box, Black Box, Basis Path*, pengujian arsitektur dan lain-lain.

1. Evaluasi Sistem

Pelanggan mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan . Jika sudah, maka langkah ketujuh dilakukan, jika belum maka mengulangi langkah 4 dan 5.

1. Menggunakan system

Perangkat lunak yang telah diuji dan diterima pelanggan siap untuk digunakan.

Reference

[1] T. Henny, F. Harumy, J. Sitorus, and M. Lubis, “SISTEM INFORMASI ABSENSI PADA PT. COSPAR SENTOSA JAYA MENGGUNAKAN BAHASA PEMPROGRAMAN JAVA,” 2018.

[2] S. Tjandra and G. S. Chandra, “Pemanfaatan Flutter dan Electron Framework pada Aplikasi Inventori dan Pengaturan Pengiriman Barang,” *Journal of Information System,Graphics, Hospitality and Technology*, vol. 2, no. 02, pp. 76–81, Dec. 2020, doi: 10.37823/insight.v2i02.109.

[3] L. N. Hayati, “SISTEM MONITORING KARYAWAN DENGAN METODE LBS (LOCATION BASED SERVICE) BERBASIS ANDROID,” *Jurnal RESISTOR (Rekayasa Sistem Komputer)*, vol. 2, no. 1, pp. 61–66, Apr. 2019, doi: 10.31598/jurnalresistor.v2i1.347.

[4] F. B. Zulmi, “Sistem Pendukung Keputusan Pemilihan Pegawai Berprestasi Pada Badan Kepegawaian Daerah (BKD) Binjai Menggunakan Metode Profile Matching Berbasis Android.” 2019.

[5] D. Rizky, “Mengenal Prototyping,” *DOT Intern*, 2019. https://medium.com/dot-intern/sdlc-metode-prototype-8f50322b14bf