BAB II
KAJIAN LITERATUR

2.1 Konsep Sistem Informasi

2.1.1 Sistem

Sistem adalah suatu kesatuan yang terdiri dari bagian-bagian yang saling terhubung dan berinteraksi guna mencapai sasaran tertentu. Sistem diartikan sebagai kumpulan elemen yang bekerja sama membentuk satu kesatuan yang terorganisasi untuk mencapai sasaran yang telah ditentukan. Setiap elemen dalam sistem memiliki peran dan fungsi yang berbeda, namun saling bergantung satu sama lain agar tujuan sistem dapat tercapai secara optimal [4].

Adapun karakteristik utama dari suatu sistem secara umum meliputi [5]:

1. Komponen Sistem (*Component*)

   Suatu sistem terdiri dari berbagai bagian yang bekerja sama untuk membentuk satu kesatuan. Subsistem dapat terdiri dari komponen sistem tersebut. Masing-masing subsistem memiliki karakteristik yang berbeda dari sistem secara keseluruhan dan dapat berdampak pada proses sistem secara keseluruhan.
2. Batasan Sistem (*Boundary*)

   Batasan sistem merupakan garis pemisah yang mendefinisikan ruang lingkup sistem dan membedakannya dari lingkungan luarnya. Batas ini penting untuk memfokuskan analisis dan perancangan.
3. Lingkungan Luar Sistem (*Environment*)

   Lingkungan luar sistem adalah segala sesuatu di luar batas sistem yang dapat mempengaruhi operasi sistem. Lingkungan luar sistem dapat menguntungkan dan dapat juga merugikan sistem.
4. Penghubung Sistem (*Interface*)

   Penghubung sistem adalah media yang memungkinkan aliran sumber daya atau data antara satu komponen dengan komponen lain atau antara satu subsistem dengan subsistem yang lain.
5. Masukan Sistem (Input)

   Masukan merupakan segala sesuatu yang dimasukkan ke dalam sistem untuk diproses agar menghasilkan keluaran yang diinginkan. Masukan dapat berupa data mentah, energi, atau sumber daya lainnya. Sebagai contoh, dalam sistem pengajuan cuti

karyawan, data seperti nama, jenis cuti, tanggal, dan alasan pengajuan berfungsi sebagai input yang akan diolah untuk menentukan validitas dan persetujuan cuti.

6. Pengolahan Sistem (*Process*)

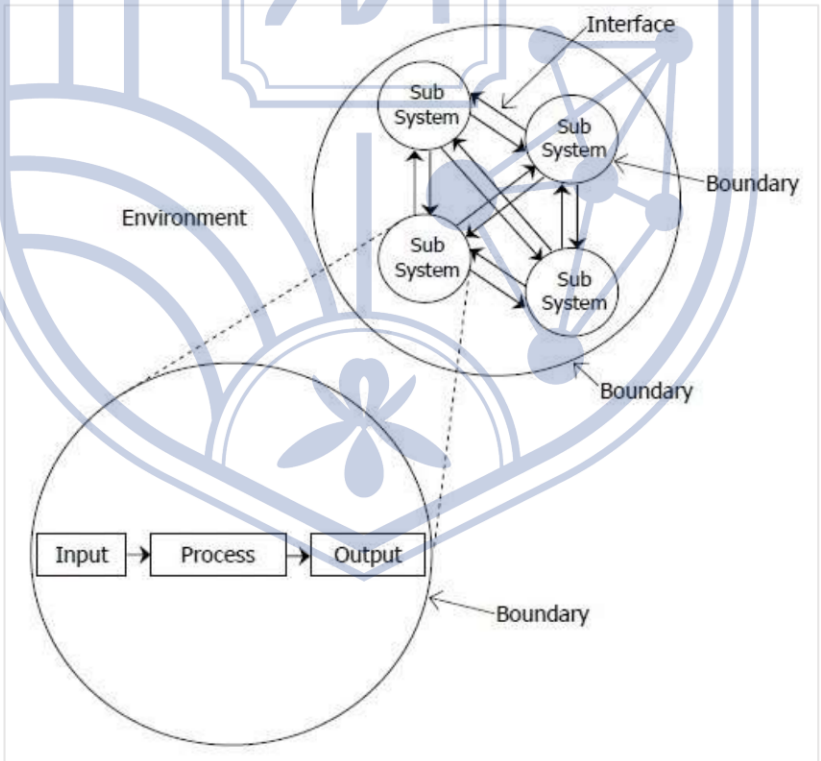
   Pengolahan sistem merupakan bagian penting yang berfungsi untuk mengubah masukan menjadi keluaran yang memiliki nilai guna. Proses ini menjadi inti dari berjalannya sistem karena menentukan hasil akhir yang dihasilkan.

7. Keluaran Sistem (*Output*)

   Keluaran merupakan hasil dari pemrosesan yang berguna bagi pengguna atau sistem lain. Keluaran harus selaras dengan tujuan sistem. Sebagai contoh, dalam sistem pengajuan cuti karyawan, keluaran yang dihasilkan dapat berupa notifikasi persetujuan atau penolakan cuti, serta pembaruan terhadap saldo cuti karyawan.

8. Sasaran Sistem (*Objective*)

   Sasaran sistem merupakan tujuan utama yang ingin dicapai oleh suatu sistem. Setiap sistem harus memiliki sasaran yang jelas agar seluruh proses yang dijalankan dapat terarah dan memberikan hasil yang bermanfaat. Tanpa adanya sasaran yang pasti, sistem tidak akan memiliki arah kerja yang jelas sehingga tidak efektif.



Gambar 2.1 Karakteristik Sistem

2.1.2 Informasi

Informasi adalah data yang telah diproses sehingga memiliki makna, nilai, dan kegunaan bagi penerimanya dalam konteks tertentu. Informasi merupakan hasil transformasi data sehingga mampu membantu pengambilan keputusan dalam sistem [5].

Informasi yang baik bisa memperhatikan sifat-sifat di bawah ini [4]:

1. Tepat Waktu (*Timeliness*)

   Informasi yang baik harus tersedia pada saat diperlukan. Ketepatan waktu ini memastikan bahwa informasi tersebut masih relevan dan dapat digunakan secara efektif dalam proses pengambilan keputusan. Sistem yang baik akan mampu merekam dan menampilkan informasi berdasarkan waktu kejadian secara akurat.

2. Akuntabilitas (*Accountability*)

   Akuntabilitas menunjukkan sejauh mana suatu informasi dapat dipertanggungjawabkan, terutama dalam penyajian data yang bersifat numerik. Semakin tinggi akuntabilitas, semakin kecil kemungkinan terjadi kesalahan interpretasi atau penyalahgunaan data.

3. Akurasi (*Accuracy*)

   Akurasi berkaitan dengan ketepatan dan keandalan informasi dalam menggambarkan kondisi sebenarnya. Informasi yang akurat bebas dari kesalahan, bias, atau distorsi, serta benar-benar mengukur hal yang seharusnya diukur.

4. Kepadatan (*Density*)

   Informasi yang padat berarti disajikan secara ringkas namun tetap mencakup inti permasalahan. Kepadatan membantu pengguna untuk memahami pokok persoalan tanpa harus membaca data yang terlalu detail dan berlebihan.

5. Relevansi (*Relevance*)

   Relevansi menunjukkan sejauh mana informasi tersebut memiliki keterkaitan dengan kebutuhan pengguna atau topik tertentu. Informasi yang relevan akan memberikan kontribusi nyata terhadap pemecahan masalah atau pengambilan keputusan yang dihadapi.

2.1.3 Sistem Informasi

Sistem informasi merupakan sekumpulan prosedur dan subsistem yang saling berinteraksi serta tersusun secara terstruktur, melibatkan kombinasi antara pengguna, perangkat keras, perangkat lunak, dan basis data. Seluruh komponen tersebut berfungsi untuk menambahkan, mengolah, memperbarui, serta menyebarkan informasi melalui sistem

telekomunikasi guna mendukung pelaksanaan transaksi harian dan pencapaian tujuan bersama dalam suatu organisasi atau perusahaan [4].

Sistem informasi terdiri dari komponen-komponen yang disebut dengan blok bangunan (*building block*). Komponen-komponen yang dimaksud terdiri dari 6 komponen berikut [6]:

1. Blok Masukan

   Blok masukan berfungsi untuk menerima dan mengumpulkan data yang berasal dari berbagai sumber, baik internal maupun eksternal organisasi. Data tersebut kemudian digunakan sebagai bahan dasar dalam proses pengelolaan informasi agar menghasilkan keluaran yang bermanfaat.

2. Blok Model

   Blok model berisi prosedur, logika, dan model matematis yang digunakan untuk mengolah data input menjadi informasi yang bermakna. Melalui blok ini, sistem dapat melakukan analisis dan perhitungan yang mendukung proses pengambilan keputusan.

3. Blok Keluaran

   Blok keluaran menghasilkan informasi yang menjadi hasil akhir dari proses sistem dan digunakan oleh berbagai tingkatan manajemen. Informasi yang dihasilkan dapat berupa laporan, dokumen, atau tampilan digital yang membantu pengawasan dan pengambilan keputusan organisasi.

4. Blok Teknologi

   Blok teknologi berperan sebagai alat utama yang memungkinkan sistem informasi berjalan dengan baik, mencakup perangkat keras, perangkat lunak, dan manusia sebagai pengguna. Teknologi ini digunakan untuk memproses input, menyimpan data, menjalankan model, dan menyalurkan keluaran informasi.

5. Blok Basis Data

   Blok basis data menyimpan kumpulan data yang saling berhubungan dan dikelola menggunakan sistem manajemen basis data. Melalui blok ini, sistem dapat mengakses, memanipulasi, dan memperbarui data secara efisien sesuai kebutuhan pengguna.

6. Blok Kendali

   Blok kendali ini berfungsi menjaga agar sistem informasi tetap berjalan dengan aman, stabil, dan terhindar dari gangguan seperti kesalahan manusia, virus, atau sabotase. Kontrol ini dilakukan melalui penerapan kebijakan keamanan, prosedur pengawasan, dan mekanisme pemulihan data.

2.1.4 Sistem Informasi Sumber Daya Manusia

Sumber daya manusia merupakan unsur terpenting dalam suatu organisasi karena menjadi penggerak utama yang menentukan keberhasilan pencapaian tujuan organisasi. SDM tidak hanya dipandang sebagai tenaga kerja, tetapi juga sebagai aset strategis yang memiliki potensi, kreativitas, dan kemampuan berpikir yang dapat dikembangkan untuk menghasilkan nilai tambah bagi organisasi [7]. Dengan demikian, manusia menjadi faktor penggerak yang mengatur dan memanfaatkan sumber daya lain seperti modal, material, dan teknologi agar tujuan organisasi dapat tercapai secara efektif dan efisien. Dalam konteks modern, organisasi dituntut tidak hanya mampu merekrut SDM yang kompeten, tetapi juga harus mampu mengembangkan potensi karyawan agar terus relevan dengan kebutuhan zaman yang dinamis.

Perkembangan teknologi informasi telah membawa perubahan besar dalam pengelolaan SDM. Saat ini, berbagai organisasi mengadopsi Sistem Informasi Sumber Daya Manusia untuk mendukung efektivitas manajemen dan pengambilan keputusan. Sistem Informasi Sumber Daya Manusia merupakan suatu sistem terintegrasi yang dirancang secara sistematis untuk mengumpulkan, menyimpan, mengelola, menganalisis, dan mendistribusikan informasi yang relevan mengenai sumber daya manusia dalam sebuah organisasi [8]. Sistem ini tidak hanya berfungsi sebagai media penyimpanan data, tetapi juga sebagai alat bantu strategis dalam pengambilan keputusan manajerial.

Sistem informasi manajemen sumber daya manusia memiliki berbagai fungsi penting dalam mendukung efektivitas manajemen organisasi. Melalui penerapan sistem ini, berbagai aktivitas yang berkaitan dengan pengelolaan tenaga kerja dapat dilakukan secara lebih sistematis, efisien, dan akurat. Adapun fungsi utama dalam konteks organisasi meliputi beberapa aspek berikut [9], [10], [11]:

1. Pengelolaan Data Karyawan

   Berperan dalam mengelola data personal dan profesional setiap karyawan, termasuk informasi pribadi, riwayat pekerjaan, serta kualifikasi yang dimiliki. Dengan adanya sistem berbasis data yang terorganisir, perusahaan dapat melakukan pemantauan terhadap karyawan dengan lebih mudah, mempercepat proses pencarian data, dan mendukung pengambilan keputusan yang berkaitan dengan ketenagakerjaan.

2. Layanan Mandiri Karyawan

   Sistem ini menyediakan platform mandiri bagi karyawan untuk mengelola berbagai kebutuhan administratif secara langsung. Melalui fitur ini, karyawan dapat mengakses data pribadi, mengajukan pengajuan, serta melihat informasi terkait pekerjaan mereka

tanpa perlu melalui proses manual. Penerapan layanan mandiri ini mendorong ketertiban karyawan dalam pengelolaan data mereka sendiri serta membantu mengurangi beban kerja administrasi pada bagian HRD.

3. Rekrutmen dan Seleksi

   Dalam proses perekrutan tenaga kerja baru, sistem ini dapat membantu organisasi menyediakan sarana digital untuk mengelola lamaran, memantau proses seleksi, dan menilai calon karyawan berdasarkan data yang terintegrasi. Sistem ini meningkatkan efisiensi proses seleksi dan memastikan kandidat yang terpilih sesuai dengan kebutuhan perusahaan, karena keputusan yang diambil didasarkan pada informasi yang valid dan terukur.

4. Pengembangan dan Pelatihan Karyawan

   Sistem ini juga digunakan untuk mengelola program pengembangan kompetensi dan pelatihan karyawan. Sistem ini memudahkan perusahaan dalam merencanakan jadwal pelatihan, memantau tingkat partisipasi karyawan, serta mengevaluasi efektivitas pelaksanaan pelatihan tersebut. Dengan begitu, perusahaan dapat memastikan peningkatan keterampilan dan performa kerja karyawan secara berkelanjutan.

5. Manajemen Penggajian dan Kompensasi

   Sistem ini juga berfungsi dalam mengelola sistem penggajian dan kompensasi secara otomatis. Melalui sistem ini, perhitungan gaji, tunjangan, serta insentif dapat dilakukan secara akurat dan transparan. Selain itu, sistem mampu menyesuaikan kompensasi dengan kebijakan perusahaan yang berlaku tanpa harus dilakukan secara manual, sehingga mengurangi potensi kesalahan administratif.

6. Manajemen Absensi

   Sistem ini berfungsi untuk mencatat dan memantau kehadiran karyawan secara otomatis melalui integrasi teknologi absensi digital. Fitur ini memungkinkan pencatatan waktu hadir secara *real-time* menggunakan berbagai metode, seperti menggunakan autentikasi biometrik dan geolokasi. Dengan adanya sistem ini, proses absensi bisa lebih menjadi transparan.

7. Keselamatan dan Kesehatan Kerja (K3)

   Penerapan K3 memiliki kaitan langsung dengan karyawan dalam organisasi, di mana kondisi kerja yang aman dan sehat terbukti dapat meningkatkan kinerja karyawan. Karyawan yang terlindungi dari risiko kecelakaan atau penyakit akibat kerja cenderung lebih produktif dan memiliki kehadiran yang lebih baik.

8. Pengelolaan Jaminan Sosial Ketenagakerjaan dan Kesehatan (BPJS)

   Sistem Informasi SDM juga berperan dalam mengelola administrasi jaminan sosial yang wajib dimiliki pekerja di Indonesia. BPJS Kesehatan adalah badan penyelenggara jaminan kesehatan nasional yang memberikan perlindungan layanan kesehatan bagi seluruh masyarakat Indonesia. Program ini memastikan pekerja mendapatkan akses pelayanan medis yang memadai mulai dari fasilitas kesehatan tingkat pertama hingga rujukan, sehingga risiko finansial akibat sakit dapat diminimalkan. BPJS Ketenagakerjaan merupakan lembaga yang memberikan perlindungan sosial bagi pekerja terkait risiko kerja, seperti kecelakaan kerja, kematian, hari tua, dan pensiun. Program-program seperti Jaminan Kecelakaan Kerja (JKK), Jaminan Kematian, Jaminan Hari Tua, dan program *Return to Work* memastikan karyawan mendapatkan perlindungan saat terjadi risiko di tempat kerja, termasuk pemulihan hingga dapat kembali bekerja.

2.2 *Rapid Application Development* (RAD)

Rapid Application Development (RAD) adalah metodologi pengembangan perangkat lunak yang berfokus pada penyelesaian sistem dalam waktu singkat melalui proses pengembangan yang bersifat iteratif, dengan melibatkan pengguna secara aktif pada setiap tahapan untuk memastikan kesesuaian kebutuhan [12]. RAD dikategorikan sebagai model proses pembangunan perangkat lunak yang bersifat inkremental, yang dirancang secara spesifik untuk menghasilkan sistem berkualitas tinggi dalam siklus pengembangan yang sangat singkat. Esensi dari RAD terletak pada kemampuannya untuk mengembangkan aplikasi secara cepat dan responsif terhadap perubahan, menjadikannya pilihan yang tepat untuk proyek-proyek dengan target waktu penyelesaian yang singkat [13].

Metode *Rapid Application Development* (RAD) memiliki empat tahapan utama yang menjadi dasar proses pengembangannya. Keempat tahap ini dilakukan secara iteratif dan melibatkan pengguna secara aktif agar sistem yang dihasilkan benar-benar sesuai dengan kebutuhan bisnis. Berikut adalah keempat tahapan tersebut [14]:

1. Perencanaan Kebutuhan (*Requirement Planning*)

   Tahap ini merupakan proses awal yang berfungsi untuk mengidentifikasi dan menganalisis kebutuhan sistem. Pengembang dan pengguna melakukan diskusi intensif guna mendefinisikan ruang lingkup, tujuan, batasan, serta sumber daya yang dibutuhkan. Proses ini juga mencakup identifikasi masalah utama, alur bisnis, serta kebutuhan data yang harus dipenuhi sistem. Fase ini dinyatakan selesai ketika kedua

belah pihak, yaitu pengembang dan pengguna telah menyepakati permasalahan inti serta arah pengembangan sistem.

2. Desain Pengguna (*User Design*)

   Pada tahap ini, pengguna dilibatkan langsung dalam proses perancangan sistem melalui pembuatan prototipe interaktif. Tujuannya adalah agar pengguna dapat memberikan masukan secara cepat terhadap rancangan yang sedang dikembangkan. Teknik yang sering digunakan dalam fase ini adalah *Joint Application Design* (JAD) dan *Computer-Aided System Engineering* (CASE) *tools*. Dengan alat bantu tersebut, keinginan pengguna dapat diterjemahkan ke dalam rancangan antarmuka dan alur sistem secara visual. Proses ini dilakukan secara berulang, sehingga setiap revisi desain didasarkan pada umpan balik pengguna hingga diperoleh bentuk rancangan yang sesuai.

3. Konstruksi (*Construction*)

   Tahap konstruksi merupakan fase di mana rancangan sistem diubah menjadi aplikasi nyata melalui proses pengodean atau yang biasa disebut *coding*. Pengembang fokus membangun modul dan fungsi sistem berdasarkan prototipe yang telah disetujui sebelumnya. Pada tahap ini dilakukan pula pengujian internal testing untuk memastikan fungsi sistem berjalan sesuai desain. Penggunaan kembali komponen perangkat lunak (*reusable components*) serta alat bantu otomatisasi pemrograman (*automated tools*) menjadi ciri khas metode RAD di fase ini, karena mampu mempercepat waktu pengerjaan dan mengurangi biaya produksi.

4. Implementasi (*Implementation*)

   Fase terakhir ini merupakan tahap pelaksanaan dari proses pengembangan ke penggunaan nyata. Aktivitas utama meliputi pengujian akhir, konversi data, pelatihan pengguna baru, serta uji coba dan pengenalan sistem ke lingkungan operasional. Selain itu, dilakukan evaluasi performa sistem serta perbaikan kecil jika ditemukan kekeliruan pasca-penerapan. Tujuan utama fase implementasi adalah memastikan sistem dapat berjalan stabil, mudah digunakan, dan siap dioperasikan sepenuhnya oleh pengguna.

Gambar 2.2 *Rapid Application Development*

Metode RAD memiliki berbagai kelebihan yang menjadikannya efektif dalam pengembangan sistem yang menuntut hasil cepat dan fleksibel. Keunggulan RAD terletak pada kolaborasi intens antara pengembang dan pengguna, serta efisiensi waktu yang dicapai melalui iterasi prototipe dan penggunaan kembali komponen perangkat lunak. Secara ringkas, kelebihan metode ini meliputi [14]:

1. Proses pengembangan lebih cepat karena menggunakan siklus iteratif dan prototipe yang segera diuji.
2. Dapat dimodifikasi dengan mudah sesuai kebutuhan pengguna tanpa mengubah keseluruhan sistem.
3. Menghemat biaya dan waktu, karena memanfaatkan komponen dan *tools* yang dapat digunakan kembali.
4. Tidak memerlukan tim yang terlalu besar, namun tetap efektif menghasilkan sistem yang berfungsi dengan baik.
5. Meningkatkan keterlibatan pengguna, karena mereka turut berpartisipasi dalam setiap tahap rancangan dan pengujian sistem.

Meskipun memiliki banyak keunggulan, metode RAD juga mempunyai sejumlah kelemahan yang perlu diperhatikan, terutama terkait ketergantungan tinggi pada pengguna dan profesionalitas tim pengembang. Jika koordinasi antar pihak tidak berjalan baik, maka hasil akhir sistem dapat menyimpang dari kebutuhan yang sebenarnya.

Adapun kelemahan metode RAD antara lain [14]:

1. Memerlukan manajemen proyek yang kuat, agar iterasi cepat terarah dan tidak saling tumpang tindih.
2. Ketergantungan yang tinggi terhadap partisipasi pengguna, sehingga jika pengguna tidak aktif, validasi rancangan menjadi terhambat.
3. Kurang cocok untuk proyek berskala besar atau kompleks, karena sulit untuk mengatur integrasi antar modul dalam waktu singkat.
4. Membutuhkan pengembang yang berpengalaman, sebab iterasi cepat dapat menurunkan kualitas sistem jika dilakukan secara terburu-buru.
5. Risiko dokumentasi dan pengujian kurang mendalam, karena fokus utama terletak pada kecepatan penyelesaian sistem, bukan dokumentasi formal.

2.3 *Unified Modelling Language* (UML)

Unified Modelling Language (UML) merupakan bahasa pemodelan standar yang digunakan untuk menggambarkan sistem perangkat lunak secara visual melalui bentuk diagram atau grafik yang bertujuan memberikan pemahaman menyeluruh terhadap struktur dan perilaku sistem. UML membantu pengembang dalam proses analisis, perancangan, serta dokumentasi sistem dengan menyediakan panduan visual yang mudah dipahami. Dalam pengembangan sistem berorientasi objek, UML berperan penting sebagai *blueprint* yang memuat model proses bisnis, rancangan basis data, serta struktur *class* yang dibutuhkan untuk membangun sistem. Selain itu, UML juga berfungsi sebagai sarana komunikasi yang efektif antar anggota tim pengembang karena mampu menampilkan hubungan dan interaksi antar komponen sistem secara terstruktur dan konsisten [15].

2.4 *Activity Diagram*

Activity Diagram merupakan salah satu jenis diagram dalam UML yang berfungsi untuk menggambarkan alur aktivitas atau proses yang terjadi di dalam suatu sistem secara berurutan. Diagram ini membantu menggambarkan urutan tindakan yang dieksekusi serta interaksi antar aktivitas yang membentuk suatu proses bisnis atau algoritma tertentu. Selain itu, *activity diagram* berfokus pada aliran kendali (*control flow*) yang menghubungkan satu aktivitas dengan aktivitas lainnya, baik dalam bentuk proses utama maupun sub-aktivitas. Diagram ini juga dapat menunjukkan bagaimana suatu aktivitas dimulai, dijalankan, dan berakhir, serta bagaimana keputusan dan percabangan logika terjadi di dalam sistem. Karena bersifat hierarkis, *activity diagram* mampu menjelaskan hubungan antara aktivitas utama

dan aktivitas pendukung secara terstruktur, sehingga memudahkan dalam memahami proses internal sistem maupun komunikasi data yang terjadi [16].

Berikut merupakan penjelasan mengenai simbol-simbol yang digunakan dalam *activity diagram* untuk menggambarkan alur kerja dan aktivitas dalam suatu sistem [17].

1. Titik Awal (*Initial Node*)

   Simbol ini berbentuk lingkaran penuh berwarna hitam yang menandakan titik awal dimulainya suatu aliran aktivitas (*workflow*). Setiap *activity diagram* hanya memiliki satu *initial node* yang menjadi titik mulai proses sistem.

2. Titik Akhir (*Final Node*)

   Simbol ini berupa lingkaran dengan lingkaran hitam di dalamnya. Simbol ini menandakan akhir dari aliran aktivitas, yaitu titik di mana proses atau alur kerja berhenti sepenuhnya.

3. Tindakan atau Aktivitas (*Action*)

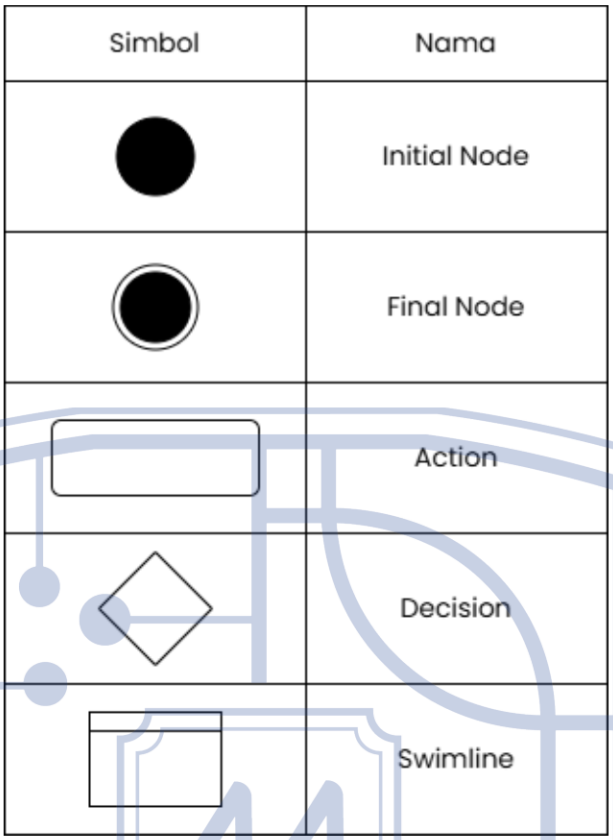
   Simbol *action* berbentuk persegi panjang. Setiap simbol *action* menggambarkan satu langkah kerja atau kegiatan yang dilakukan dalam proses sistem. Biasanya berisi deskripsi aktivitas seperti “Memverifikasi Data” atau “Menyimpan Informasi”.

4. Keputusan (*Decision*)

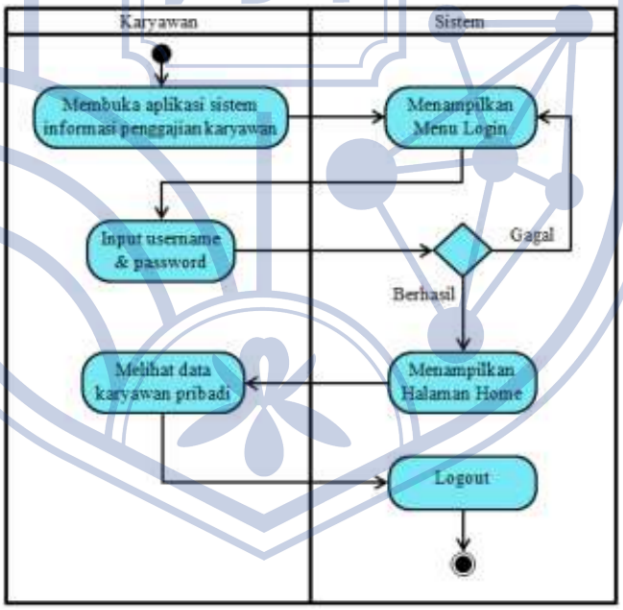
   Simbol *decision* berbentuk belah ketupat dan digunakan untuk menunjukkan adanya percabangan logika atau keputusan dalam alur aktivitas. Dari simbol ini akan keluar dua atau lebih panah alur dengan kondisi yang berbeda, misalnya “Ya” atau “Tidak”.

5. Kolom Aktor atau Bagian (*Swimlane*)

   Simbol *swimlane* berupa garis pemisah horizontal atau vertikal yang berfungsi untuk mengelompokkan aktivitas berdasarkan aktor, bagian, atau entitas yang bertanggung jawab melaksanakan aktivitas tersebut. Dengan *swimlane*, pembaca dapat dengan mudah mengetahui siapa yang melakukan setiap aktivitas dalam sistem.



Gambar 2.3 Simbol *Activity Diagram*



Gambar 2.4 Contoh *Activity Diagram*

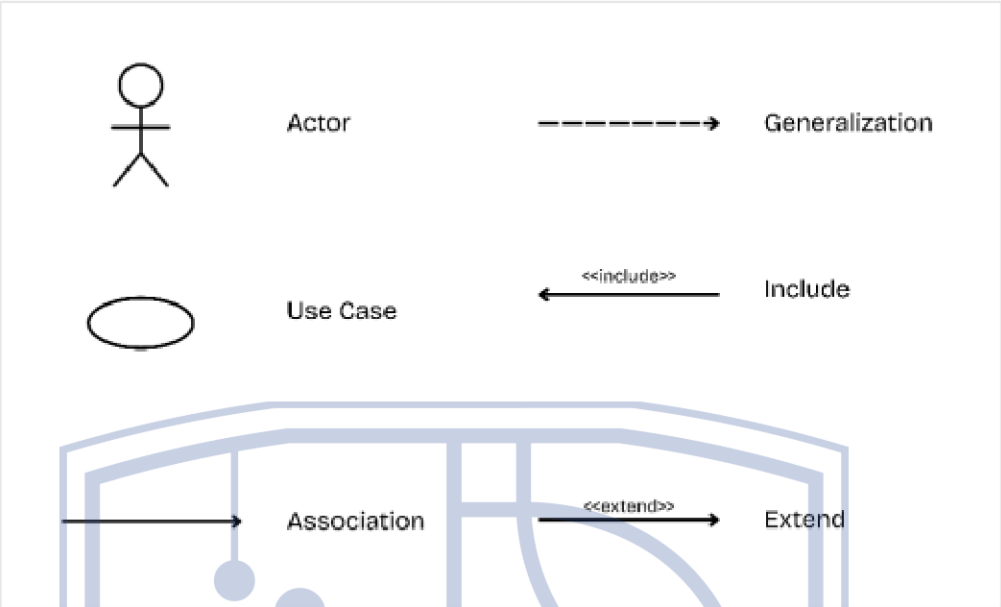
2.5 *Use Case Diagram*

Use case diagram adalah diagram pemodelan *Unified Modeling Language* (UML) yang mendefinisikan fungsi sistem dari perspektif penggunanya. Diagram ini menggambarkan fungsionalitas sistem serta interaksi antar aktor (pengguna atau sistem

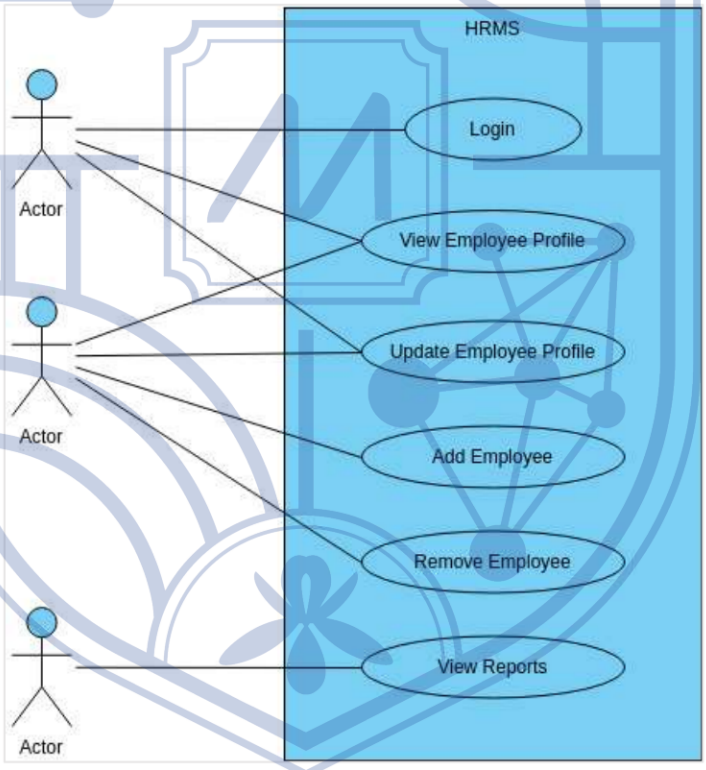
eksternal) dengan sistem tersebut [18]. Ada dua elemen penting yang harus digambarkan, yaitu aktor dan *use case*. Aktor adalah entitas eksternal, bisa berupa manusia, proses, atau sistem lain yang berinteraksi langsung dengan sistem yang dirancang. Sedangkan *use case* adalah fungsi atau layanan sistem yang menggambarkan apa yang sistem tawarkan kepada aktor, yaitu unit fungsional yang saling tukar interaksi antara sistem dan aktor untuk memenuhi kebutuhan pengguna.

Dalam *use case* diagram yang dikembangkan oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson (Three Amigos), terdapat enam simbol dasar yang sering dipakai untuk memodelkan interaksi sistem dengan aktor dan hubungan antar *use case*. Berikut adalah penjelasan setiap simbol [19]:

1. Aktor di *use case diagram* adalah entitas eksternal yang berinteraksi dengan sistem. Simbolnya digambarkan sebagai figur manusia sederhana (*stick figure*) atau kotak bernama aktor. Aktor menunjukkan siapa yang menggunakan atau terlibat dalam *use case* tertentu.
2. *Use case* mempresentasikan fungsi atau layanan yang disediakan sistem kepada aktor. Simbolnya berbentuk oval yang di dalamnya diberi nama *use case*. *Use case* menunjukkan apa yang dilakukan sistem dari sudut pandang pengguna.
3. Asosiasi adalah garis penghubung antara aktor dan *use case* yang menunjukkan interaksi atau komunikasi antara mereka. Simbolnya berupa garis lurus tanpa arah panah yang menghubungkan aktor ke *use case*.
4. Generalisasi menunjukkan bahwa satu aktor atau *use case* mewarisi sifat dari aktor atau *use case* yang lain yang lebih umum. Simbolnya berupa panah yang menunjuk ke entitas umum.
5. *Include* menunjukkan bahwa satu *use case* mewajibkan pelaksaan *use case* lain sebagai bagian dari alur kerjanya. Simbol relasi *include* berupa panah dengan label <<include>> yang menunjuk dari *use case* utama ke *use case* yang disertakan.
6. *Extend* menggambarkan kemungkinan tambahan opsional yang berfungsi untuk memperluas alur dari *use case* utama berdasarkan kondisi tertentu. Simbolnya juga panah dengan label <<extend>> yang dari *use case* ekstensi menuju *use case* dasar, menunjukkan bahwa ekstensi terjadi jika kondisi tertentu terpenuhi.



Gambar 2.5 Simbol *Use Case Diagram*



Gambar 2.6 Contoh *Use Case Diagram*

2.6 *Class Diagram*

Class diagram merupakan salah satu bentuk pemodelan yang memiliki peran penting dalam *Unified Modeling Language* (UML). Diagram ini berfungsi untuk menggambarkan model logis dari suatu sistem yang sedang dikembangkan. Melalui *class diagram*, perancang sistem dapat memperlihatkan struktur dan arsitektur sistem secara jelas, termasuk hubungan

antar bagian di dalamnya. Diagram ini digambarkan menggunakan *class* yang berisi atribut serta *method*, dan setiap *class* dihubungkan satu sama lain melalui garis yang disebut asosiasi. Dengan demikian, *class diagram* membantu dalam memahami bagaimana komponen sistem saling berinteraksi serta bagaimana data dan fungsi saling terhubung dalam suatu rancangan perangkat lunak [20].

Dalam *class* diagram, terdapat beberapa simbol yang digunakan untuk menggambarkan elemen dan hubungan antar *class* dalam sistem, sebagai berikut [19]:

1. *Class*

   Class merupakan komponen utama dalam pemodelan berorientasi objek yang berfungsi sebagai blok penyusun sistem. Setiap *class* digambarkan dalam bentuk persegi panjang yang terbagi menjadi tiga bagian, yaitu nama *class*, atribut, dan metode (*method*).

2. *Association*

   Association menggambarkan adanya hubungan atau interaksi antara dua *class*. Hubungan ini biasanya digambarkan dengan garis lurus yang memiliki ujung panah terbuka untuk menunjukkan arah aliran pesan dari satu *class* ke *class* lainnya.

3. *Aggregation*

   Aggregation menunjukkan hubungan keseluruhan bagian (*whole–part relationship*) antara *class*. Relasi ini menggambarkan bahwa satu *class* dapat terdiri dari beberapa *class* lain, tetapi *class* bagian masih dapat berdiri sendiri tanpa bergantung pada *class* utama.

4. *Composition*

   Composition merupakan bentuk hubungan yang menunjukkan ketergantungan penuh antara dua *class*. Apabila *class* bagian tidak dapat berdiri sendiri dan hanya dapat ada sebagai bagian dari *class* lain, maka hubungan tersebut disebut *composition*. Hubungan ini digambarkan dengan garis yang ujungnya memiliki simbol jajaran genjang berisi (*solid diamond*).

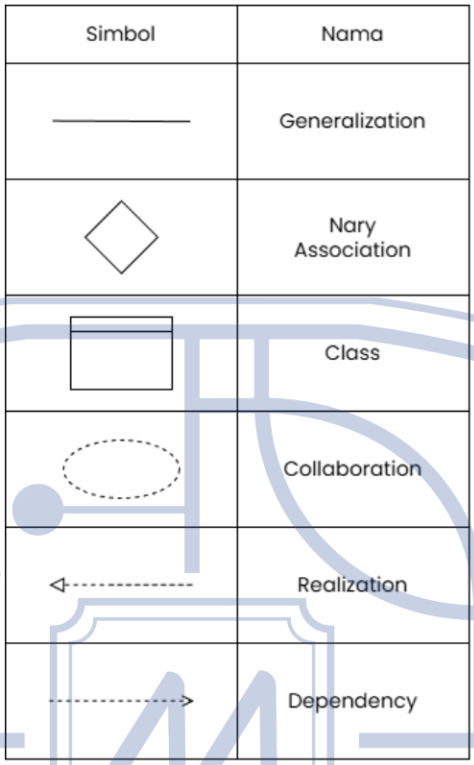
5. *Generalization*

   Generalization menggambarkan hubungan hierarkis antar *class* dari yang bersifat umum (*superclass*) menuju yang lebih spesifik (*subclass*). Simbol ini digunakan untuk menunjukkan pewarisan sifat atau karakteristik antar *class* dalam sistem.

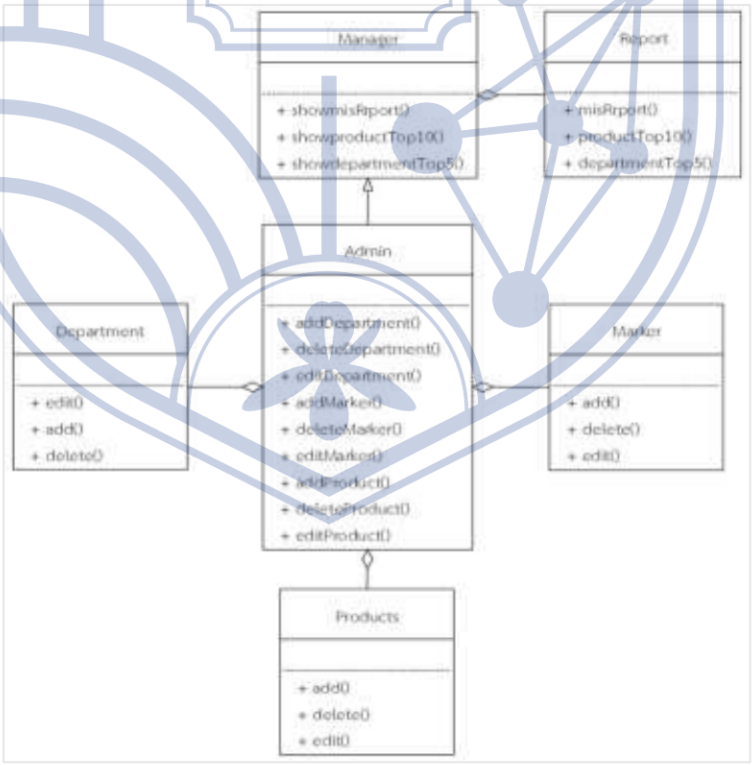
6. *Dependency*

   Dependency menunjukkan hubungan di mana satu *class* bergantung pada *class* lain, biasanya karena penggunaan atribut atau metode dari *class* tersebut. Hubungan

dependency dilambangkan dengan garis panah putus-putus (*dotted arrow*) untuk menunjukkan arah ketergantungan antar *class*.



Gambar 2.7 Simbol *Class Diagram*



Gambar 2.8 Contoh *Class Diagram*

2.7 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) merupakan diagram yang berfungsi untuk membantu proses perancangan basis data dengan menggambarkan hubungan antar entitas beserta atribut yang dimilikinya. ERD berperan penting dalam memvisualisasikan struktur data secara sistematis, sehingga hubungan antar data dapat dipahami dengan lebih mudah. Dengan demikian, ERD dapat digunakan sebagai model konseptual yang menjelaskan keterkaitan antara objek-objek utama dalam suatu sistem basis data yang saling berelasi dan saling mempengaruhi satu sama lain [21].

Dalam *Entity Relationship Diagram* (ERD), terdapat beberapa komponen utama yang berperan penting dalam menggambarkan struktur dan hubungan data di dalam suatu basis data. Komponen-komponen ini membantu perancang sistem untuk memahami bagaimana data saling berhubungan dan bagaimana informasi disimpan serta diorganisasikan. Adapun elemen utama yang terdapat dalam ERD meliputi [22]:

1. Entitas

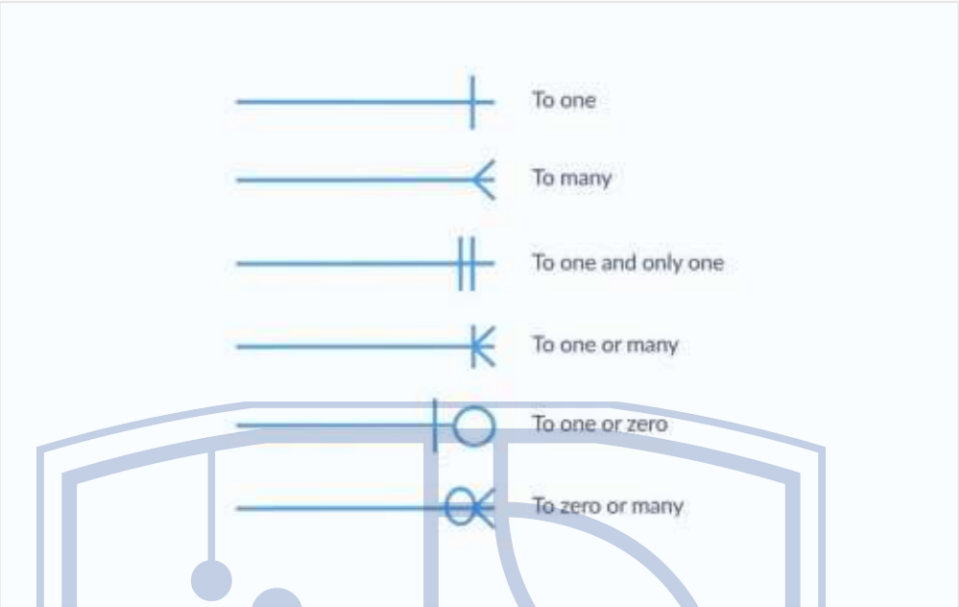
   Merupakan elemen utama yang menjadi fokus dalam suatu basis data. Entitas dapat berupa manusia, tempat, benda, maupun kondisi tertentu yang berkaitan dengan data yang diperlukan. Dalam diagram ERD, entitas biasanya digambarkan menggunakan simbol persegi panjang.

2. Atribut

   Merupakan data atau informasi yang melekat pada suatu entitas. Setiap entitas memiliki *primary key* sebagai identitas unik, serta atribut deskriptif yang memberikan perincian tambahan. Atribut dapat ditempatkan di dalam tabel entitas atau dipisahkan ke dalam tabel tersendiri, dan umumnya digambarkan dengan bentuk elips.

3. Relasi

   Merupakan penghubung antara dua atau lebih entitas dalam suatu basis data. Dalam diagram ERD, relasi digambarkan menggunakan simbol berbentuk belah ketupat.



Gambar 2.9 Simbol *Entity Relationship Diagram* (ERD)

Gambar 2.10 Contoh *Entity Relationship Diagram* (ERD)

2.8 Metode PIECES

Metode PIECES adalah salah satu pendekatan analisis sistem yang digunakan untuk mengidentifikasi permasalahan dan mengevaluasi kinerja suatu sistem dari berbagai aspek. PIECES merupakan akronim dari *Performance, Information, Economy, Control, Efficiency*, dan *Service*. Keenam aspek ini membantu menganalisis sistem dalam memahami kekurangan sistem yang berjalan serta menentukan prioritas perbaikan yang tepat. Melalui metode ini, analisis sistem dilakukan secara menyeluruh dan objektif terhadap seluruh komponen sistem informasi.

Analisis ini mencakup enam dimensi penting, yaitu [23]:

1. Kinerja (*Performance*)

   Aspek kinerja digunakan untuk menilai sejauh mana sistem mampu memberikan hasil sesuai dengan waktu yang diharapkan, termasuk kecepatan proses, waktu respons, dan kapasitas sistem dalam menangani beban kerja. Analisis kinerja dilakukan untuk mengidentifikasi bagian dari sistem yang mengalami penurunan performa dan perlu dioptimalkan.

2. Informasi (*Information*)

   Aspek ini berfokus pada kualitas dan keakuratan informasi yang dihasilkan oleh sistem. Data yang baik harus relevan, tepat waktu, lengkap dan akurat agar dapat mendukung pengambilan keputusan. Analisis informasi dilakukan untuk memastikan konten atau data yang disajikan mudah dipahami oleh pengguna dan sesuai dengan kebutuhan mereka.

3. Ekonomi (*Economy*)

   Analisis ekonomi ini bertujuan untuk membandingkan antara biaya yang dikeluarkan dan manfaat yang diperoleh dari sistem. Evaluasi dilakukan untuk memastikan bahwa investasi dalam pembangunan sistem sepadan dengan nilai manfaatnya. Aspek ini juga membantu dalam menentukan prioritas pengembangan berdasarkan efisiensi biaya.

4. Kontrol (*Control*)

   Aspek kontrol berkaitan dengan mekanisme pengawasan terhadap keamanan sistem dan validitas data. Sistem harus mampu mencegah penyalahgunaan, menjaga kerahasiaan data, serta membatasi hak akses sesuai dengan peran pengguna.

5. Efisiensi (*Efficiency*)

   Efisiensi mencerminkan seberapa optimal sumber daya sistem digunakan untuk menghasilkan keluaran. Sumber daya yang dimaksud mencakup waktu, tenaga, perangkat keras, serta perangkat lunak. Dalam analisis ini, dilakukan penilaian terhadap penggunaan sumber daya agar tidak terjadi pemborosan, seperti optimalisasi kode program, *query* basis data, atau kapasitas server.

6. Layanan (*Service*)

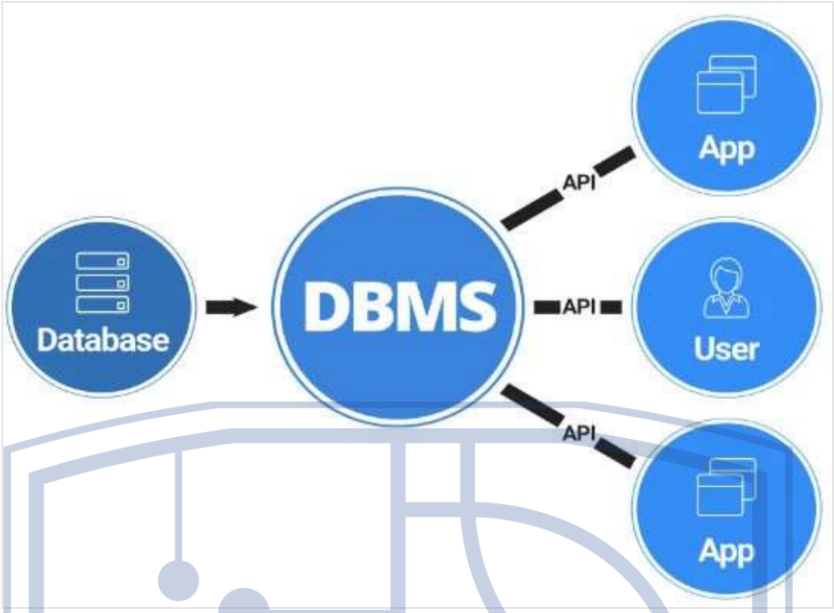
   Aspek layanan menilai sejauh mana sistem mampu memberikan pelayanan terbaik kepada pengguna. Hal ini meliputi kemudahan pengguna (*usability*), tampilan antarmuka yang menarik, dan kecepatan dalam memberikan respons terhadap kebutuhan pengguna.

Tabel 2.1 Contoh Analisis Menggunakan Metode PIECES

Parameter	Penjelasan
Performance	Proses administrasi karyawan masih manual dan memakan waktu lama karena harus melalui banyak tahapan persetujuan.
Information	Data kehadiran dan pengajuan belum dapat diakses secara *real-time*, sehingga menurunkan akurasi dan transparansi informasi.
Economy	Proses manual menyebabkan pemborosan waktu, biaya, dan tenaga akibat penggunaan formulir kertas serta rekap data berulang.
Control	Sistem manual kurang memiliki kontrol dan validasi data yang baik, sehingga rawan kesalahan dan manipulasi.
Efficiency	Tidak adanya integrasi sistem menyebabkan proses administrasi berulang dan memperlambat alur kerja.
Service	Penyampaian informasi HRD masih lambat dan karyawan belum memiliki akses digital terhadap data pribadi mereka.

2.9 *Database Management System* (DBMS)

Database Management System (DBMS) adalah sebuah sistem perangkat lunak yang dirancang khusus untuk mengelola, mengorganisasi, dan menampilkan data yang tersimpan dalam basis data. DBMS berfungsi sebagai perantara antara program aplikasi dan data fisik yang tersimpan, memungkinkan pengguna untuk menambah, menghapus, memodifikasi, dan mengambil informasi dengan cara yang terkontrol. Peran fundamental DBMS adalah menyediakan abstraksi data, di mana pengguna dan pengembang aplikasi tidak perlu mengetahui detail teknis tentang bagaimana data disimpan secara fisik. Dengan menyembunyikan kompleksitas ini, DBMS secara signifikan mempercepat pengembangan aplikasi dan memastikan data dapat diakses secara praktis dan efisien [24].



Gambar 2.11 *Database Management System*

2.9.1 Basis Data

Basis data adalah sekumpulan informasi yang terstruktur dan disimpan secara elektronik untuk mendukung kegiatan operasional maupun pengambilan keputusan pada organisasi [25]. Penggunaan basis data berfungsi untuk menjaga integritas data, menghindari redundansi, serta mempercepat proses pencarian informasi [26].

Dalam pengelolaan basis data dikenal beberapa istilah penting, yaitu [27]:

1. Entitas (*Entity*)

   Entitas merupakan objek atau konsep yang informasinya direkam di dalam sistem basis data. Dalam konteks pengelolaan SDM, entitas dapat berupa karyawan, jabatan, departemen, atau divisi. Setiap entitas berfungsi untuk merepresentasikan unit informasi penting yang menjadi dasar dalam kegiatan administrasi karyawan.

2. Atribut (*Field*)

   Field adalah atribut atau kolom yang menggambarkan karakteristik dari suatu entitas. Setiap *field* menyimpan nilai tertentu yang menjelaskan detail informasi entitas tersebut. Sebagai contoh, entitas karyawan dapat memiliki *field* seperti *employee_name*, *job_position*, dan *join_date*.

3. Rekaman Data (*Record*)

   Record merupakan kumpulan *field* yang saling berhubungan dan mewakili satu unit data utuh dari suatu entitas. Dalam sistem informasi SDM, satu *record* dapat

merepresentasikan data lengkap mengenai seorang karyawan, mulai dari identitas pribadi hingga riwayat pekerjaannya.

4. Nilai Data (*Data Value*)

   Nilai data adalah isi aktual dari setiap atribut atau *field* yang menyimpan informasi spesifik. Misalnya, pada *field employee_name*, nilai datanya bisa berupa "Budi". Nilai data inilah yang nantinya digunakan untuk proses analisis, pelaporan, atau pengambilan keputusan oleh pihak manajemen SDM.

5. Elemen Kunci Data (*Key Data Element*)

   Elemen kunci data merupakan atribut yang digunakan untuk mengidentifikasi setiap entitas secara unik di dalam basis data. Dalam sistem SDM, elemen kunci data dapat berupa *employee_id*, yang berfungsi membedakan satu karyawan dengan karyawan lainnya.

2.9.2 *Data Definition Language* (DDL)

Data Definition Language (DDL) merupakan bahasa dalam basis data yang berfungsi untuk mendefinisikan dan mengelola struktur dari suatu basis data, seperti pembuatan, perubahan, serta penghapusan objek-objek di dalamnya, termasuk tabel, indeks, dan tampilan. DDL memungkinkan pengguna menentukan rancangan data seperti nama tabel, kolom, tipe data, serta aturan integritas. Beberapa perintah utama dalam DDL antara lain [28]:

1. *CREATE TABLE* digunakan untuk membuat tabel baru beserta kolom dan kunci utamanya.
2. *ALTER TABLE* berfungsi untuk mengubah struktur tabel, misalnya menambah atau menghapus kolom.
3. *DROP TABLE* digunakan untuk menghapus tabel beserta seluruh isinya.

2.9.3 *Data Manipulation Language* (DML)

Data Manipulation Language (DML) merupakan bahasa dalam basis data yang digunakan untuk mengelola isi atau data di dalam tabel, seperti menambahkan, menampilkan, memperbarui, dan menghapus data. DML memungkinkan pengguna melakukan operasi CRUD (*Create, Read, Update, Delete*) terhadap data dalam basis data. Beberapa perintah utama dalam DML antara lain [28]:

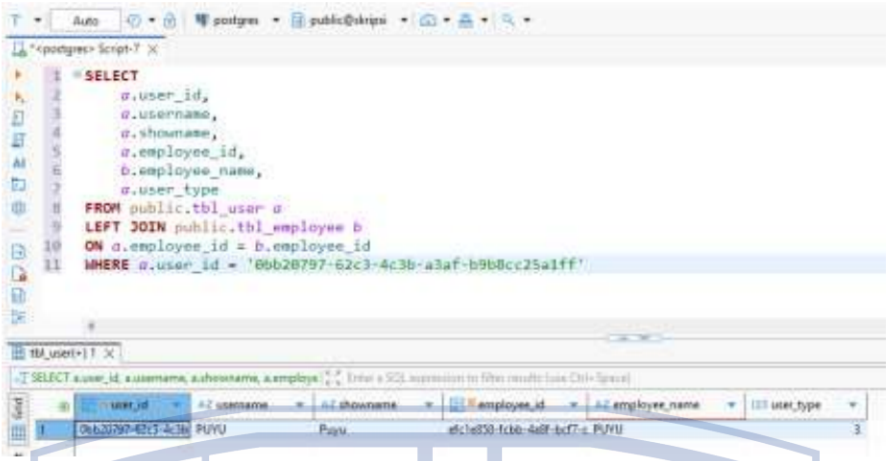
1. *SELECT* digunakan untuk menampilkan atau mengambil data dari tabel dalam basis data.

2. *INSERT* berfungsi untuk menambahkan data baru ke dalam tabel yang telah ada.
3. *UPDATE* digunakan untuk memperbarui atau mengubah data yang sudah tersimpan dalam tabel.
4. *DELETE* berfungsi untuk menghapus data tertentu dari tabel sesuai dengan kondisi yang ditentukan.

2.10 PostgreSQL

PostgreSQL merupakan sistem manajemen basis data objek-relasional *open-source* yang kuat dan fleksibel. Sistem ini memperluas kemampuan bahasa SQL dengan berbagai fitur canggih untuk penyimpanan dan pengelolaan data berskala besar secara aman dan efisien. PostgreSQL dikenal karena arsitekturnya yang andal, integritas data yang tinggi, serta ekstensibilitas yang luas, menjadikannya pilihan utama bagi banyak individu dan organisasi di seluruh dunia. Dengan hampir 40 tahun pengembangan aktif, PostgreSQL telah menjadi basis data relasional *open-source* terdepan yang terus berinovasi dan memberikan performa tinggi bagi berbagai kebutuhan aplikasi.

PostgreSQL menawarkan beragam fitur yang membantu pengembang, *administrator*, dan pengguna dalam membangun sistem yang tangguh serta menjaga integritas data. Sebagai perangkat lunak yang gratis dan sangat dapat diperluas, PostgreSQL memungkinkan pengguna membuat tipe data dan fungsi kustomisasi, serta menulis kode dalam berbagai bahasa pemrograman tanpa perlu melakukan kompilasi ulang basis data. Sistem ini mendukung beragam tipe data (seperti JSON, XML, UUID, dan geometri), indeks lanjutan, partisi tabel, replikasi sinkron dan asinkron, serta keamanan tingkat tinggi melalui autentikasi berlapis dan kontrol akses yang ketat. Dengan skalabilitas tinggi dan kemampuan menangani beban data dalam jumlah besar maupun banyak pengguna secara bersamaan, PostgreSQL menjadi solusi yang efisien, stabil, dan terpercaya untuk pengelolaan data modern [29].



Gambar 2.12 Contoh *Query* Pada PostgreSQL

2.11 Aplikasi *Mobile*

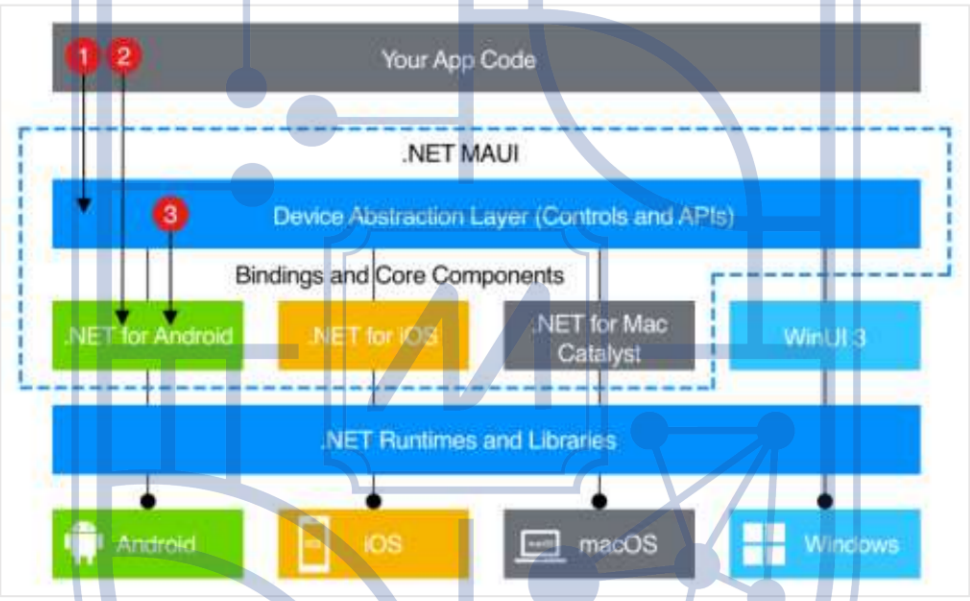
Aplikasi *mobile*, atau yang dikenal dengan aplikasi seluler, merupakan perangkat lunak yang dirancang untuk dijalankan pada perangkat portabel seperti ponsel pintar (*smartphone*) maupun tablet. Aplikasi jenis ini berfungsi untuk memenuhi kebutuhan pengguna melalui layanan yang mirip dengan yang tersedia di komputer pribadi. Meskipun keterbatasan perangkat keras menyebabkan aplikasi seluler menghindari aktivitas *multitasking* yang berat, perkembangan teknologi telah menjadikan aplikasi ini semakin fleksibel dan sesuai dengan kebutuhan pengguna, karena memungkinkan mereka memilih serta menyesuaikan fungsi sesuai keinginan melalui perangkat masing-masing [30].

2.12 *Framework* .NET MAUI

Teknologi .NET MAUI adalah kerangka kerja (*framework*) modern untuk membangun aplikasi *native* yang dapat dijalankan di berbagai sistem operasi seperti Android, iOS, macOS, dan Windows dengan menggunakan satu basis kode (*single codebase*) berbasis bahasa pemrograman C# dan XAML. Dengan pendekatan ini, pengembang dapat menciptakan antarmuka pengguna dan logika bisnis yang konsisten tanpa perlu menulis ulang kode untuk setiap platform. .NET MAUI merupakan evolusi dari Xamarin.Forms yang dioptimalkan agar lebih sederhana. terintegrasi, dan produktif dalam ekosistem .NET 6 ke atas.

.NET MAUI menyediakan satu kerangka kerja terpadu untuk membangun antarmuka pengguna (*User Interface*) pada aplikasi *mobile* dan *desktop*. Arsitektur .NET MAUI dirancang secara modular agar setiap lapisan memiliki peran spesifik dalam proses pengembangan dan eksekusi aplikasi lintas platform. Berdasarkan dokumentasi resmi

Microsoft, arsitektur kerja .NET MAUI memiliki arsitektur berlapis yang memungkinkan pengembangan aplikasi lintas platform dari satu basis kode. Lapisan teratas adalah *Your App Code*, tempat pengembang menulis logika dan antarmuka menggunakan C# dan XAML. Di bawahnya, lapisan .NET MAUI bertindak sebagai abstraksi perangkat, menyediakan kontrol dan API umum. Lapisan ini terhubung ke implementasi spesifik platform seperti .NET for Android, iOS, Mac Catalyst, dan WinUI 3, yang menjembatani ke sistem operasi masing-masing. Di bagian bawah, .NET Runtimes and Libraries mendukung eksekusi aplikasi secara konsisten di semua platform. Arsitektur ini memungkinkan aplikasi tampil dan berfungsi secara *native* tanpa perlu menulis ulang kode untuk tiap perangkat [31].



Gambar 2.13 .NET MAUI *Architecture Diagram*

2.13 *Black Box Testing*

Black Box Testing merupakan jenis pengujian perangkat lunak yang bertujuan untuk menemukan kesalahan atau kekeliruan pada sistem aplikasi, seperti kesalahan fungsi maupun menu yang tidak berfungsi atau hilang. Dengan kata lain, metode ini berfokus pada pengujian fungsionalitas sistem aplikasi, tanpa memperhatikan bagaimana kode atau struktur internalnya dibuat. Dalam proses pengujian ini, penguji memberikan input data secara acak untuk memastikan bagaimana sistem merespons. Jika data yang dimasukkan tidak sesuai dengan format atau aturan yang ditetapkan, maka sistem akan menolak input tersebut, dan data tidak akan tersimpan di basis data. Sebaliknya, bila data yang dimasukkan benar dan sesuai dengan format yang diharapkan, maka sistem akan menerima dan menyimpan data tersebut dalam basis data. Dengan demikian, pengujian *Black Box* menilai keandalan sistem berdasarkan hasil keluaran (*output*) yang muncul dari berbagai kondisi masukan [32].

Equivalence Class Partitioning (ECP) merupakan salah satu teknik pengujian dalam metode *Black Box* yang digunakan untuk memverifikasi kesesuaian data masukan terhadap aturan validasi sistem. Teknik ini dilakukan dengan cara membagi data input ke dalam beberapa kelas atau partisi ekuivalen yang memiliki karakteristik serupa, di mana satu data dari setiap partisi dianggap mewakili keseluruhan kelas tersebut. Pengujian ini diterapkan untuk memastikan bahwa sistem dapat mengenali data valid dan menolak data yang tidak sesuai dengan ketentuan [33].