BAB II

LANDASAN TEORI

2.1 Tinjauan Teori

2.1.1 *Database*

Berbagai organisasi maupun personal sekalipun membutuhkan informasi tertentu. Informasi-informasi seperti nama, tanggal, nomor telepon, hingga departemen dan gaji merupakan bagian-bagian dari informasi yang disebut dengan data. *Database* merupakan salah satu media yang dapat digunakan untuk menyimpan kumpulan data maupun *file* yang dapat diakses oleh pemilik maupun pihak dengan *privilege* atau otoritas atas akses dan/atau modifikasi berbagai data tersebut. Untuk memenuhi kebutuhan pengguna, saat ini semakin berkembang jenis dari *database* yang mampu menyimpan jenis dan format data yang lebih kompleks dan beragam untuk fungsi-fungsi tertentu.

Untuk mengakses, mengatur, dan memodifikasi *database*, pengguna membutuhkan sebuah sistem, maka sebuah sistem bernama *Database Management System* (DBMS) muncul untuk mengani kebutuhan tersebut. Selain mengelola data itu sendiri, DBMS mampu mengatur kebutuhan dalam akses, keamanan, dan penyimpanan serta dapat menjadi *host* untuk fungsi lain pada sistem *database* yang digunakan. DBMS memberikan keuntungan dalam ketersediaan penyimpanan, akses, dan *update* informasi dalam jumlah besar secara efisien dan efektif [7].

2.1.1.1 *Relational Database*

Relational Database merupakan jenis *database* yang merepresentasikan informasi di dalamnya menggunakan tabel-tabel (*tables*) berisi baris (*row*) dan kolom (*column*). Sebuah tabel terhubung dengan sebuah relasi (*relation*) yang mengelompokkan menjadi sebuah koleksi atau kumpulan objek dengan tipe yang sama (*rows*). Data dalam sebuah tabel dapat terhubung dengan sebuah kunci atau konsep umum, serta memiliki kemampuan untuk mengambil data yang terhubung

atau berelasi dengannya dalam sebuah tabel. Hal ini merupakan pengertian dari sistem *relational database* [8].

Sebuah sistem bernama *Relational Database Management System* (RDBMS) digunakan untuk mengelola *relational database*. Beberapa contoh diantaranya yaitu MySQL, PostgreSQL, Oracle Database, dan Microsoft SQL Server. RDBMS mempunyai *property* krusial untuk mendefinisikan relational *database yang* disebut dengan ACID *properties* [9]. *Property* ini dapat memastikan bahwa seluruh data transaksi pengguna dalam *database* lengkap dan tersimpan baik pada RDBMS termasuk seluruh perubahan data transaksi yang ada.

- *Atomicity*: Seluruh elemen yang menjadikan transaksi *database* lengkap.
- *Consistency*: Aturan untuk memelihara *data point* dalam *state* yang benar setelah setiap transaksi.
- *Isolation*: Menahan efek dari setiap transaksi agar tidak terlihat ke pihak lain sampai selesai *commit*, untuk menghindari kerancuan (*confusion*).
- *Durability*: Memastikan perubahan data menjadi permanen setelah transaksi di-*commit*.

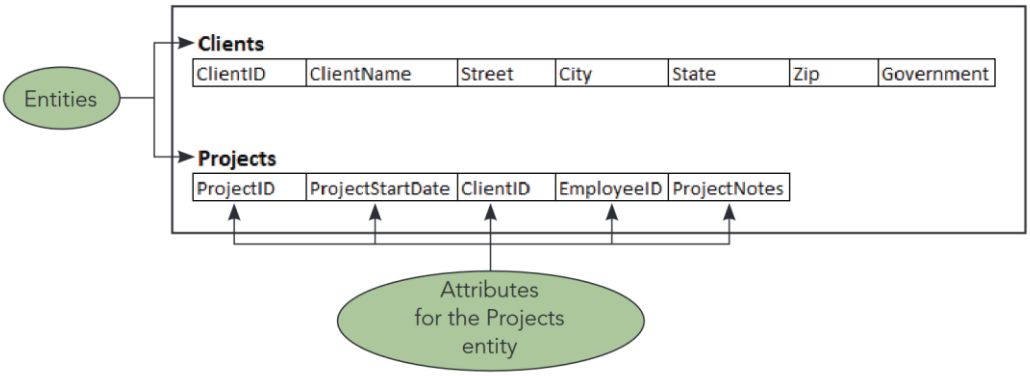
2.1.1.2 *Data Model*

Data model merupakan model *database* yang memberikan makna atau bertujuan untuk mencapai abstraksi data (*data abstraction*) dan menggambarkan skema *database*. Data model merupakan definisi yang abstrak, lengkap, dan tidak bergantung pada implementasi yang terdiri dari sekumpulan 4 *tuple* (*T*, *S*, *O*, *C*) yang menjadi basis pengembangan mesin abstrak yang berinteraksi dengan pengguna *database*. *T* yaitu kumpulan tipe data (*data type*), *S* yaitu kumpulan tipe struktur data (*data structure type*), *O* yaitu kumpulan tipe operasi data (*data operation type*), serta *C* yaitu kumpulan tipe batasan integritas (*integrity constraint type*) [10]. Secara sederhana, data model memberikan gambaran dari konsep abstrak dari bentuk atau skema *database* yang akan dibuat. *Data model* berisi elemen-elemen data dalam desain *database* sehingga sesuai dengan tipe-tipe, batasan, dan syarat data yang akan digunakan dalam *database*. *Data model* juga

dapat memberikan gambaran mengenai pemahaman jenis, arti, dan relasi data dalam skenarionya di dunia nyata.

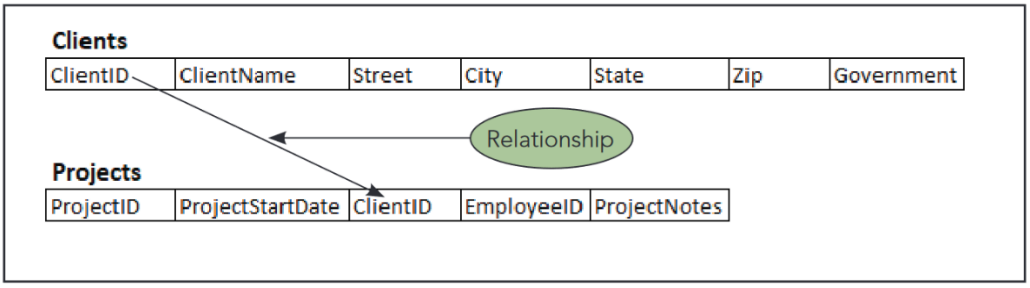
Terdapat banyak jenis dari *data model*, dua diantaranya yaitu *Entity-Relation* (ER) dan XML [10]. *Entity Relationship Model* (ERM) adalah sebuah model konseptual yang merepresentasikan struktur dari informasi sebuah domain masalah dalam konteks entitas dan relasi. Hasil dari ERM merupakan sebuah grafik yang menggambarkan sebuah *Entity Relationship Diagram* (ERD). Maka, ERD merepresentasikan struktur konseptual domain masalah yang dibuat modelnya. ERD banyak digunakan dalam perancangan *database* dan analisis sistem untuk menemukan *requirement* dari sebuah sistem atau domain masalah. Secara khusus, ketika digunakan untuk pemodelan data, ERD mendukung *database designer* untuk mengidentifikasikan baik data maupun aturan-aturan yang direpresentasikan dan digunakan dalam sebuah *database*. ERD dapat langsung diubah menjadi skema *relational database* [11]. Komponen-komponen dalam model ER antara lain:

1) *Entity* atau entitas adalah satu orang, tempat, kejadian, benda, atau transaksi lain yang akan disimpan dan diproses datanya. Contohnya yaitu karyawan, klien, estimasi proyek, dan tugas. *Entity* direpresentasikan oleh sebuah tabel data dalam sistem *relational database* [12].
2) *Attribute* atau atribut merupakan karakteristik atau *property* dari sebuah *entity*. Misalnya *entity* klien, atributnya dapat terdiri dari nama klien, jalan, kota, negara, kode pos, dan apakah klien tersebut merupakan badan pemerintahan. *Attribute* juga disebut sebagai *field* atau *column* dalam berbagai sistem *database* [12].



Gambar 2.1 *Entity* dan *attribute*
Sumber: [12]

3) *Relationship* atau relasi merupakan asosiasi antar *entity*. Misalnya, terdapat sebuah asosiasi antara klien dan proyek. Satu klien terhubung dengan seluruh proyek miliknya, dan sebuah proyek terhubung ke kliennya [12]. Jenis-jenis relasi antara lain *one-to-one*, *one-to-many*, dan *many-to-many*.



Gambar 2.2 Contoh relasi *one-to-many*
Sumber: [12]

2.1.1.3 SQL

Structured Query Language atau SQL merupakan bahasa yang digunakan untuk mengelola objek dan data dalam aplikasi *database* atau DBMS. IBM pertama kali mengembangkan versi asli SQL yang disebut *sequel* pada awal dekade 1970. Jenis-jenis SQL yaitu *data-definition language* (DDL), *data-manipulation language* (DML), *integrity*, *view definition*, *transaction control*, *embedded* SQL dan *dynamic* SQL, serta *authorization* [13]. Dengan menggunakan jenis SQL yang tepat, dapat memberikan hasil dan manipulasi sesuai tujuan dan kebutuhan dari pengguna *database*. Format SQL pada berbagai DBMS pun beragam dengan dialek masing-masing DBMS yang digunakan oleh pengguna.

2.1.2 Visualisasi Data

Visualisasi sendiri merupakan sebuah proses yang mengubah representasi data mengenai suatu hal dari kehidupan nyata menjadi representasi visual. Visualisasi adalah proses yang bertujuan untuk memberikan makna pada abstraksi visual. Perancang proses visualisasi harus memahami komponen dan tugas *point-of-view* untuk mencapai makna yang dimaksud [14]. Visualisasi data adalah representasi grafis dari informasi dan data dengan menggunakan elemen visual seperti bagan, grafik, dan peta yang memberikan akses untuk melihat dan memahami tren, *outlier*, dan pola dalam data [15].

Chen et al. [14] menyatakan beberapa tujuan dari visualisasi sebagai *"isms" in visualization*, yaitu:

1) *Insightism*: Tujuan utama visualisasi adalah mendapatkan *insight* atau wawasan dari data. Wawasan tersebut dapat berupa pemahaman dalam mengenai permasalahan yang kompleks atau momen pemahaman dalam situasi yang kompleks. Arti lain dari wawasan dalam visualisasi yaitu konklusi tepat yang didapatkan dari melihat visualisasi.
2) *Cognitivism*: Sejumlah visualisasi dapat membuat manusia "melihat yang tidak terlihat", "memaksimalkan pemahaman manusia", "memperkuat kognisi", dan "membantu berpikir".
3) *Communicationism*: Memberikan bantuan yang efektif untuk komunikasi informasi dan penyebaran pengetahuan.
4) *Economism*: Memberikan manfaat yang lebih nyata atau berwujud. Energi sebagai alat ukur dasar mendekati waktu dan biaya finansial dari jumlah informasi yang didapat. Jumlah tersebut didapatkan dari informasi akhir yang dipahami manusia karena dalam gambar visualisasi dan tidak ada bias kognitif yang menyebabkan hilangnya informasi.
5) *Pragmatism*: Memberikan jawaban dari pertanyaan yang tidak dipertanyakan dengan menjawab pertanyaan baru yang diselidiki dan ditemukan untuk mendukung keputusan sehingga dapat memberikan informasi, meningkatkan efisiensi, dan "menginspirasi".
6) *Essentialism*: Tidak menghadirkan hiasan visual (*visual embellishment*) yang tidak diperlukan untuk memahami data yang digambarkan.

7) *Decorationism*: *Visual embellishment* dapat digunakan dalam visualisasi dan mampu memberikan manfaat.
8) *Isomorphism*: Tidak menyebabkan distorsi yang tidak konsisten dalam sumber data.
9) *Polymorphism*: Distori dapat diperlihatkan dalam visualisasi serta memberikan manfaat.
10) *Mechanism*: Sebagian besar, jika tidak semua, proses *data intelligence* dapat diotomatisasi dengan *data mining* dan teknik *machine learning*. Jumlah data yang tersedia di era "*big data*" menjadikan otomatisasi penting dan *feasible* (dapat dilakukan).
11) *Anti-mechanism*: Berbagai *workflow* pada *data intelligence* yang cukup kompleks harus selalu melibatkan manusia, dimana kemampuan analitis manusia dapat ditingkatkan dengan teknik visualisasi interaktif.

2.1.3 PHP

PHP (*Hypertext Preprocessor*) merupakan bahasa pemrograman komputer yang digunakan untuk membuat *website*, halaman *web*, maupun aplikasi di dalam internet, beberapa diantaranya untuk Facebook, Wikipedia, dan WordPress. PHP diciptakan sebagai bahasa *script* untuk menyediakan *rich dynamic content* (yaitu konten yang berasal dari halaman PHP lain atau dinamis dari sumber eksternal seperti *database*). PHP merupakan *interpreted language*, yaitu bahasa yang tidak perlu di-*compile* dan dibuat menjadi *executable file*, tetapi PHP diinterpretasikan tiap barisnya oleh *web server* yang menjalankan PHP. Maka dari itu, PHP dapat langsung dimuat ulang untuk melihat perubahan dari kode yang sudah dibuat. PHP merupakan bahasa *script server-side* yang dapat direspons oleh *web server* melalui HTTP pada permintaan (*request*) *client*. Sebuah *client* (*browser*) meminta sebuah URL, yang kemudian akan dikirim oleh *web server* ke sebuah *script*. *Script* yang ada akan membaca permintaan dan mengembalikan konten yang diminta berdasarkan kode *script* [16].

2.1.4 Laravel

Laravel merupakan *framework* PHP yang diciptakan oleh Taylor Otwell pada tahun 2011 [17]. Laravel menyediakan *framework* pengembangan web yang terdiri

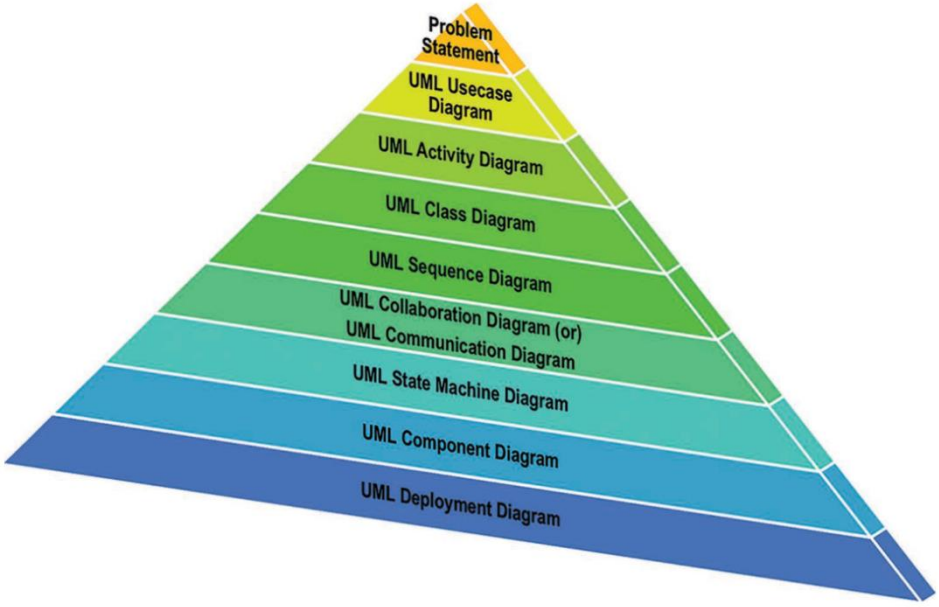
dari komponen *database*, *queue*, *web socket*, dan *authentication* yang diperlukan dalam pengembangan aplikasi web [18]. Laravel dibangun dengan arsitektur MVC (*Model-View-Controller*), yaitu sebuah arsitektur aplikasi yang memisahkan logika (data dan *controller*) dan presentasi (*user interface*) aplikasi. *Model* mewakili struktur data atau *database* dari aplikasi, *view* merupakan bagian yang mengatur tampilan halaman web ke *user*, dan *controller* merupakan *bridge* antara *model* dan *view* sehingga pengelolaan *database* dapat berjalan melalui *view* atau halaman web [19].

2.1.5 *Responsive Web Design*

Website design merupakan bagian penting dalam proses pengembangan *website* [20]. Hal ini termasuk dengan pengaturan konten menjadi model grafis yang dapat digunakan sebagai dasar kode situs. Perkembangan jumlah pengguna internet dan perangkat *mobile*, seperti *smartphone* dan tablet, menghadirkan sebuah kebutuhan untuk menyesuaikan konten yang ditampilkan pada masing-masing perangkat. Ide perancangan berbagai versi dari sebuah situs yang sama untuk mengisi setiap ukuran layar dan resolusi menghadirkan *responsive design* sebagai solusi teknis dimana *website* beradaptasi secara dinamis terhadap lebar perangkat. *Responsive web design* terdiri dari tiga komponen teknis yaitu: *fluid grid*, *flexible image*, dan *media query*. *Fluid grid* menetapkan unit yang bergantung atau unit relatif pada elemen-elemen halaman web; *flexible image* yang juga ditetapkan dengan ukuran dalam unit relatif; dan *media query* yang dapat mengalihkan ke berbagai CSS berbeda berdasarkan fitur dari perangkat [21].

2.1.6 *Unified Modeling Language* (UML)

UML merupakan sebuah alat diagram standar untuk perancangan. Ini merupakan sebuah bahasa grafis untuk menentukan, menggambarkan, mengonstruksikan, dan mendokumentasikan artefak sistem *software*. UML membantu memahami produk atau sistem *software* dengan lebih baik agar dapat dikembangkan antar *developer* dan konsumen. Diagram UML terdiri dari diagram-diagram penting utama seperti *usecase diagram*, *activity diagram*, *class diagram*, *sequence diagram*, *collaboration* atau *communication diagram*, *state machine diagram*, *component diagram*, dan *deployment diagram* [22].

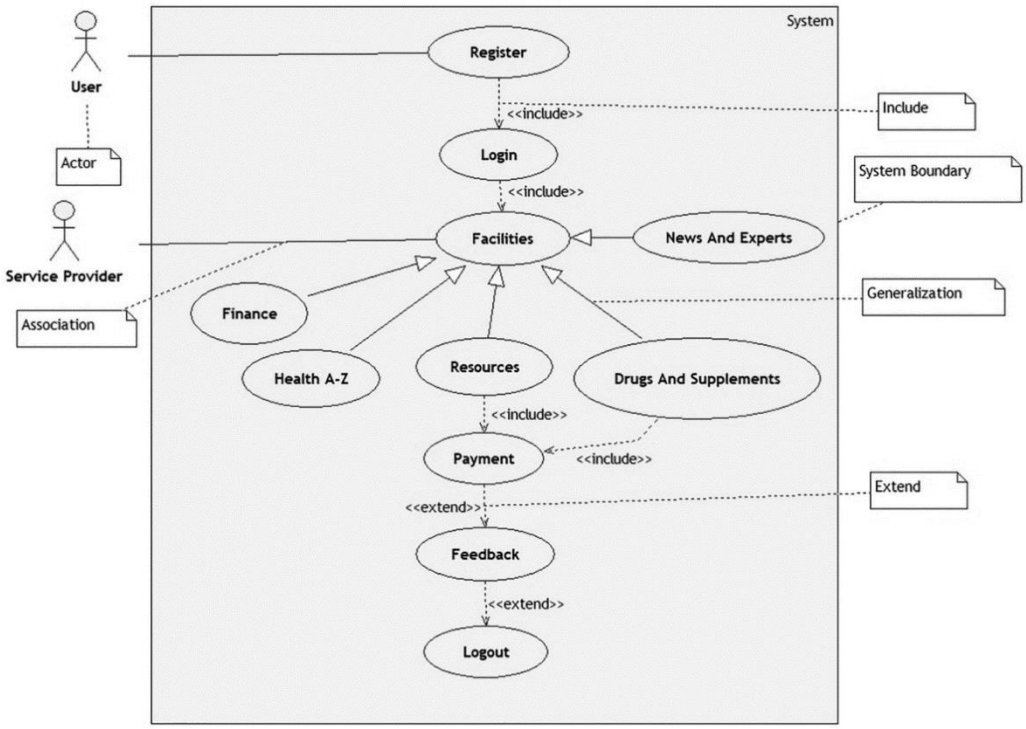


Gambar 2.3 Piramida diagram UML
Sumber: [22]

2.1.6.1 *Usecase Diagram*

Usecase diagram berfokus pada identifikasi dari kebutuhan-kebutuhan fungsional dari sistem yang akan dipertimbangkan. Komponen dari *usecase diagram* yaitu: *system boundary* yang menggambarkan *scope* sistem dan mengenkapsulasikan bagian komplit dan fungsionalitas sistem; *actor* yaitu pengguna yang berinteraksi dengan sistem dan sebuah *usecase* yang dapat berupa manusia, organisasi, atau sistem dan dapat memberikan pengaruh pada fungsionalitas sistem; dan *usecase* yang merupakan gambaran visual dari fungsionalitas yang berbeda-beda dalam sistem. Sementara relasi pada *usecase* yaitu: *association* yang menggambarkan relasi antara sebuah *actor* dan *usecase*; *directed association* yang menggambarkan relasi *one-way* dimana *actor* bertanggungjawab dalam mempengaruhi sebuah *usecase*; *include* menggambarkan sebuah *usecase* termasuk dalam fungsionalitas dari sebuah *usecase* lain; *extend* menggambarkan relasi antar dua *usecase* mengartikan relasi penting yang memperluas *usecase* dengan menambahkan *behavior* tambahan dari fungsionalitas yang sudah ada dari *usecase* dasar; *generalization* yang menggambarkan relasi antar dua *parent usecase* dengan satu atau lebih *child usecase*; dan *dependency* yang

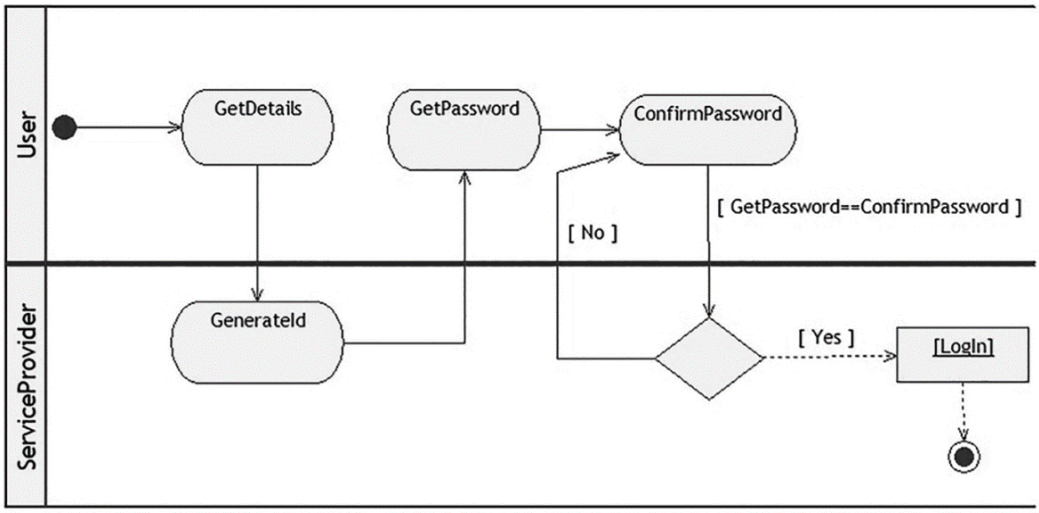
merupakan relasi dimana kehadiran sebuah *usecase* tergantung pada kehadiran *usecase* lain [22].



Gambar 2.4 Contoh *usecase diagram*
Sumber: [22]

2.1.6.2 *Activity Diagram*

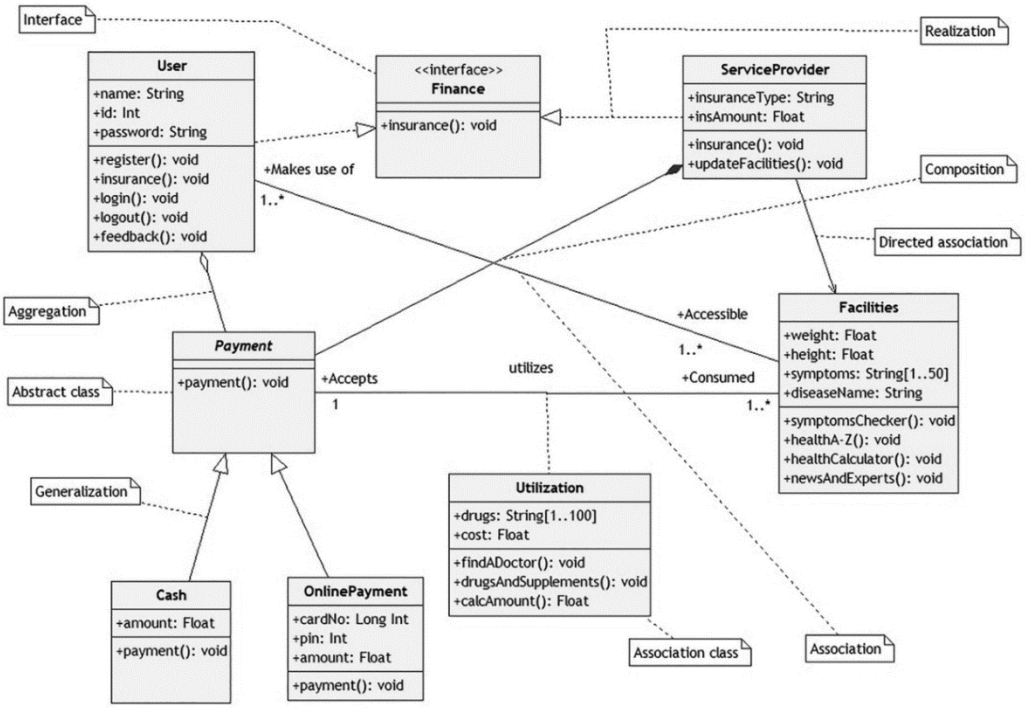
Activity diagram menggambarkan aktivitas-aktivitas yang saling berurutan atau paralel yang termasuk dalam bagian masing-masing kebutuhan fungsional dari sistem. Komponen *activity diagram* antara lain: *initial state* yang menggambarkan kondisi awal sistem yang dipertimbangkan; *final state* yaitu kondisi yang mengakhiri sistem yang dipertimbangkan; *swimlanes* yaitu partisi yang terdiri dari *entity* dan kumpulan aktivitas yang terkait; *action state* yaitu sebuah operasi, aktivitas bisnis, atau proses; *object* yaitu *entity* yang membawa data antar dua *action state*; *decision* merupakan *node* dengan 1 atau 2 atau *input* dan banyak *output* yang bergantung pada kondisi yang dirancang; dan *transition* yaitu panah yang menggambarkan pergerakan dari *activity state* sumber ke *activity state* target yang dipicu oleh penyelesaian aktivitas dari *activity state* sumber [22].



Gambar 2.5 Contoh *activity diagram*
Sumber: [22]

2.1.6.3 *Class Diagram*

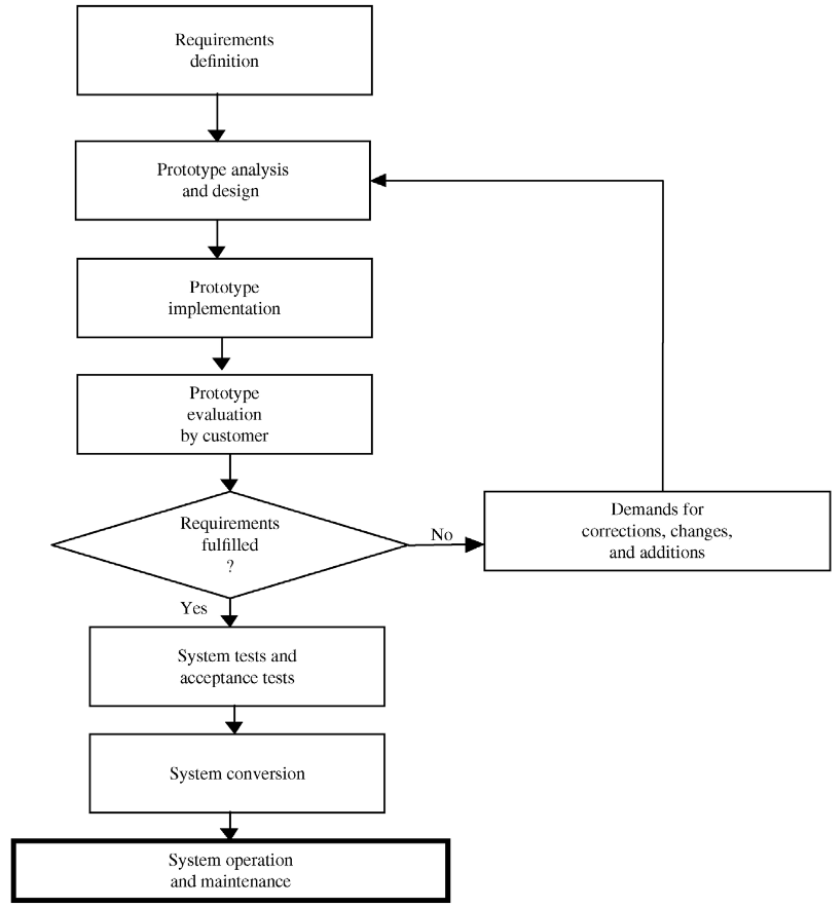
Class diagram menggambarkan struktur dari sistem yang dinyatakan dengan berbagai *class* dan *object*. Komponennya yaitu: *class* yang menggambarkan nama, struktur atau atribut, dan *behavior* atau operasi; dan *class relationship* yang menggambarkan relasi antar *class* yaitu *association*, *directed association*, *dependency*, *aggregation*, *composition*, *generalization*, dan *realization* [22].



Gambar 2.6 Contoh *class diagram*
Sumber: [22]

2.1.7 *Prototyping*

Prototyping merupakan salah satu dari model klasik untuk pengembangan *software*. Model *prototyping* merupakan sebuah metode *iterative* dimana setiap iterasinya akan dikembangkan sebuah *software prototype*. Pada iterasi pertama, hanya bagian yang ada dalam *requirement* yang diimplementasikan, dan pada setiap iterasi berikutnya terdapat implementasi *prototype* baru sebagai tambahan dari *requirement*. Setiap *prototype* diperiksa dan dievaluasi oleh *customer* dan tim *user*. Permintaan dari dua pihak tersebut digunakan dalam perbaikan, perubahan, dan penambahan yang berkaitan dengan *prototype* yang ada dan dapat dipertimbangkan untuk integrasi *prototype* selanjutnya oleh developer. Seluruh iterasi tersebut akan berlanjut hingga seluruh *requirement* terpenuhi [23].



Gambar 2.7 Pengaplikasian umum dari model *prototyping*
Sumber: [23]

Prototype merupakan model yang dapat digunakan, namun belum selesai. *Prototype* hanya terdiri dari sebagian fungsionalitas dan dimaksudkan untuk

direvisi, dikembangkan, dan ditingkatkan saat setiap siklus atau iterasi proses *development*. Terdapat dua kategori *prototype*. *Horizontal prototype* merupakan model keseluruhan *software* yang sedang dikembangkan yang berfokus pada *user interface*, menawarkan gambaran umum dari keseluruhan sistem dengan fungsi internal yang terbatas. Sementara itu, *vertical prototype* menggambarkan bagian-bagian tersendiri dari sistem secara mendalam yang memberikan gambaran dari cara kerja lebih rinci masing-masing komponen, seperti fungsi atau subsistem tertentu [24].

Boehm dalam Galin [23] bahwa secara statistik melalui penelitian komparasi antara pengembangan proyek dengan metode SDLC dan *prototyping* menunjukkan bahwa proyek *prototyping* 40% lebih sedikit menggunakan *resource*, 40% lebih sedikit kode, sedikit lebih rendah tingkat fungsionalitas dan *robustness* pada *software* yang dihasilkan, dan lebih tinggi tingkat kemudahan penggunaan dan pembelajarannya dibandingkan dengan proyek yang menggunakan SDLC.

2.2 Penelitian Terdahulu

No.	Penelitian	Fokus, Metode, dan Hasil Penelitian	Bagian dari Referensi yang Digunakan
1	Judul: *Design and realization of rock salt gas storage database management system based on SQL Server*. Penulis:	Fokus: Pengembangan *database management* untuk proyek konstruksi gua bawah tanah untuk penyimpanan batu garam menggunakan Microsoft SQL Server dalam bahasa C#. Sistem digunakan untuk manajemen data penampungan gas dengan fungsi untuk pendefinisian, penyimpanan, pengambilan, modifikasi,	Rancangan *database* dan pendefinisian data.

No.	Penelitian	Fokus, Metode, dan Hasil Penelitian	Bagian dari Referensi yang Digunakan
	Yingjie Wang, Jianjun Liu, Xiang He, Bing Wang. Jurnal: Petroleum. Volume 4. Issue 4. Tahun penerbitan: 2018.	penghapusan, operasi, dan *maintenance* data. Metode: Menggunakan 4 tahap perancangan arsitektur *database* meliputi fase *requirement analysis, conceptual design, logical structure design*, dan *physical design*. Hasil: Aplikasi *database* dan DBMS untuk proyek penyimpanan gas dengan sistem yang komprehensif dan kompleks. DBMS yang dibangun mampu memenuhi kebutuhan aktual proyek dengan meningkatkan operasi dan modifikasi data secara signifikan serta menyediakan dukungan teknis kuat untuk proyek.	
2	Judul: *Model Sistem Informasi untuk Asesmen Risiko*	Fokus: Membuat sistem informasi berbasis web menggunakan model prediksi Denver HIV	Rancangan *database* yang sudah dibangun.

No.	Penelitian	Fokus, Metode, dan Hasil Penelitian	Bagian dari Referensi yang Digunakan
	HIV Menggunakan Data Perilaku. Penulis: Rohana Uly Pradita Siregar, Kemal Nazaruddin Siregar. Jurnal: Preventia: The Indonesian Journal of Public Health. Volume 6. Nomor 1. Tahun penerbitan: 2021.	Risk Score untuk *assesment* risiko HIV. Sistem mengumpulkan data sosial-demografis dengan 8 variabel (3 variabel demografis dan 5 variabel perilaku berisiko). Metode: Menggunakan metode SDLC. Hasil: Sistem masih berupa rancangan sistem berdasarkan studi literatur. Belum terdapat wawancara dan observasi untuk menghasilkan aplikasi yang lebih memenuhi kebutuhan.	
3	Judul: *Analysis and Design of Web-Based Information System for Church Congregrations Case Study: Church BNKP Pewarta*. 	Fokus: Membuat sistem aplikasi berbasis *website* untuk memproses data lebih mudah untuk gereja serta dapat digunakan untuk memperoleh informasi secara cepat, tepat, dan akurat. Metode:	Penggunaan UML untuk merancang sistem *website*.

No.	Penelitian	Fokus, Metode, dan Hasil Penelitian	Bagian dari Referensi yang Digunakan
	Penulis: Jansen Wiratama, Ririn Ikana Desanti. Jurnal: Ultima Infosys: Jurnal Ilmu Sistem Informasi. Volume 12. No. 2. Tahun penerbitan: 2021	Pengembangan *website* dengan metode WDLC dan UML untuk memahami hubungan antar data. Hasil: *Website* dapat diakses dan mampu menjalankan fitur *create*, *read*, *update*, dan *delete* pada data kongregasi gereja serta dapat membuat dan menampilkan informasi dan berita dalam *website*.	
4	Judul: *Development of Mobile Cloud Application using UML*. Penulis: Dong Kwan Kim. Jurnal: International Journal of Electrical and Computer Engineering	Fokus: Pembuatan rangkaian proses dan prosedur untuk pengembangan *mobile cloud application* yang terdiri dari konfigurasi sistem *multi-tier*. Konfigurasi tersebut biasanya menghalangi pengembangan aplikasi. Penelitian dilakukan untuk meningkatkan efektivitas komunikasi *developer*. Metode: Proses pengembangan aplikasi menggunakan *Unified*	Penggunaan UML untuk merancang sistem *website*.

No.	Penelitian	Fokus, Metode, dan Hasil Penelitian	Bagian dari Referensi yang Digunakan
	(IJECE). Volume 8. No. 1. Tahun penerbitan: 2018.	*Modeling Language* (UML) untuk Android melalui Amazon Web Service. Hasil: Pendekatan untuk pengembangan aplikasi yang menyediakan pedoman sistematik untuk penerapan *profile extension*, *class diagram*, dan *deployment diagram* dalam UML. Hasil ini mampu meningkatkan produktivitas, skalabilitas, dan *maintainability* dari model rancangan *software*.	
5	Judul: *Perancangan Website E-commerce Produk Kopi Menggunakan Metode Prototyping (Studi Kasus: Kedai Kopi Kontekstual)*. Penulis:	Fokus: Perancangan *website* untuk memperluas pemasaran, *brand awareness*, dan pengetahuan calon pelanggan untuk Kedai Kopi Kontekstual. Metode: Metode *prototyping*. Hasil:	Metode pembangunan *website* dengan *prototyping*.

No.	Penelitian	Fokus, Metode, dan Hasil Penelitian	Bagian dari Referensi yang Digunakan
	Nyokro Hidayat Purba Wijayakusuma, Yudha Saintika, Irwan Susanto. Jurnal: Journal of Information Systems and Informatics. Volume 3. No. 3. Tahun penerbitan: 2021.	*Website* telah dibuat dan diuji dan sudah sesuai dengan kebutuhan pengguna untuk memperluas pemasaran Kedai Kopi Kontekstual.	