BAB II
TINJAUAN PUSTAKA

2.1 Sistem Informasi

Menurut Anggraeni & Irviani (2017) Sistem adalah sekelompok unsur yang hubungan satu dengan yang lainnya erat, dan berfungsi bersama-sama untuk mencapai tujuan tertentu. Sedangkan informasi adalah hasil dari pengolahan data dalam bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan kejadian-kejadian yang nyata yang digunakan untuk mengambil keputusan (Anggraeni & Irviani, 2017).

William S. Davis dan David C. Yen (1998) menyatakan bahwa sistem informasi merupakan seperangkat perangkat keras, perangkat lunak, data, manusia, dan komponen prosedural yang ditujukan untuk menyediakan data dan informasi yang benar kepada pengguna yang sesuai di waktu yang tepat.

2.2 PHP

PHP atau kependekan dari *Hypertext Preprocessor* adalah bahasa pemrograman web bersifat *server side*, yang bertujuan untuk menghasilkan *script* yang akan di-*generate* dalam kode HTML yang merupakan bahasa standar web (Edy Winarno ST, Zaki, & Community, 2013). HTML sendiri merupakan bahasa yang menjelaskan format dan konten dokumen, yang pada dasarnya terdiri atas teks dan Gambar statis (Darie, Brinzarea, Cherecheş-Toşa, & Bucica, 2006). HTML sendiri merupakan kependekan dari *Hypertext Markup Language*, yang menurut Edy Winarno dkk (2013) merupakan bahasa markup untuk memformat konten halaman web atau bisa dikatakan sebagai bahasa untuk mengatur bagaimana penampilan dan pemformatan konten di web.

2.3 MySQL

Untuk melakukan penyimpanan data atau informasi yang terintegrasi, diperlukan yang dinamakan *database*. Pengelolaan *database* tersebut dibutuhkan sebuah perangkat lunak atau *software* yang dinamakan *DBMS* atau *Database Management System*. *MySQL* adalah perangkat lunak atau *software* dari *DBMS* yang *multithread* dan *multi-user* (Solichin, 2010). *MySQL* menjadi salah satu aplikasi *database* yang paling populer terutama di web *programming* (Pratama,

2017). Menurut Achmad Solichin (2010), *SQL* atau *Structured Query Language* sendiri merupakan suatu bahasa (*language*) yang digunakan untuk mengakses *database* atau bisa disebut juga dengan istilah *query*.

2.4 XAMPP

Proses integrasi antara *database* dengan perangkat lunak dipermudah dengan adanya pemanfaatan *web server*. *XAMPP* difungsikan sebagai server yang berdiri sendiri dan merupakan *software* kompilasi dari beberapa program seperti Apache, HTTP Server, MySQL *Database* dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl, yang mendukung banyak sistem operasi (Palit, Yaulie D.Y. Rindengan, & Arie S.M. Lumenta, 2015). *Tool* ini sangat membantu pengembang perangkat lunak dalam melakukan pengembangan *software*.

2.5 *Framework* CodeIgniter

Berdasarkan deskripsi yang dituliskan dalam *website* CodeIgniter sendiri, dijelaskan bahwa CodeIgniter merupakan *application development framework* atau kerangka pengembangan aplikasi yang bertujuan untuk membantu pengembangan proyek aplikasi berjalan lebih cepat dibandingkan harus dikembangkan dari awal (*scratch*) (CodeIgniter Foundation, 2021). Arti dari *framework* sendiri adalah sebuah kerangka program yang digunakan untuk membantu *developer* atau pengembang untuk dapat mengembangkan kode secara konsisten (Kurniawan, 2020).

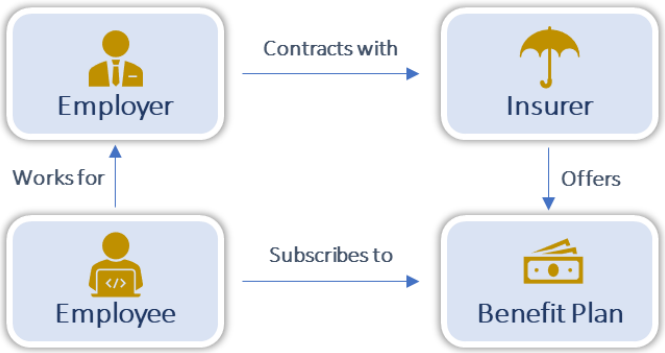
2.6 Visual Studio Code

Visual Studio Code merupakan sebuah teks editor buatan Microsoft yang ringan dan handal untuk sistem operasi multiplatform yang tersedia juga untuk versi Linux, Mac, dan Windows serta teks editor yang bersifat *open source* (Ummy Gusti Salamah, 2021). *Open source* disini, menurut buku dari Ummy Gusti Salamah (2021), memiliki arti bahwa kode dari sumbernya dapat dilihat dan dapat berkontribusi dalam pengembangannya, jadi para pengembang aplikasi dapat ikut serta dalam proses pengembangan VS Code kedepannya.

2.7 *Conceptual Data Model* (CDM)

Conceptual Data Model atau bisa disingkat CDM mendefinisikan hubungan antara entitas dunia nyata dalam suatu domain tertentu, dimana entitas tersebut

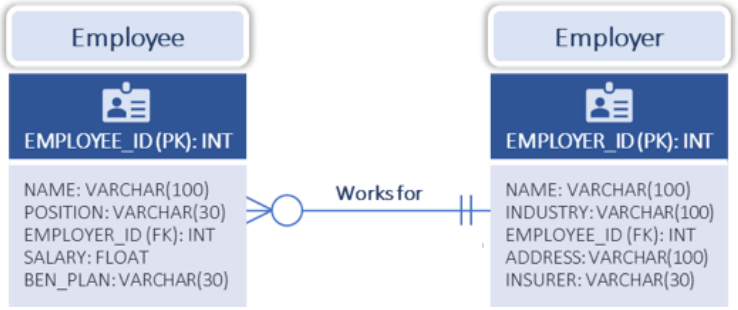
digambarkan dalam bentuk kotak, sementara hubungan antar entitas digambarkan dalam bentuk garis atau panah (CMS.gov, 2021). Entitas mewakili sesuatu yang nyata dan dapat dibedakan antara entitas satu dengan entitas lain, serta tiap entitas memiliki atribut yang mendefinisikan karakteristik dari entitas tersebut. Contoh penggambaran *Conceptual Data Model* (CDM) dapat dilihat pada Gambar 2.1.



Gambar 2. 1 *Conceptual Data Model* (CDM)

2.8 *Physical Data Model* (PDM)

Implementasi *data model* dalam *database* digambarkan *dalam Physical Data Model* atau PDM. Entitas digambarkan sebagai tabel, atribut sebagai kolom pada tabel, dan ditentukan juga tipe data dari setiap kolom. *Physical Data Model* (PDM) berisikan penggambaran struktur basis data aplikasi secara detail (Pradana, Hariadi, & Shintawati, 2017). Contoh penggambaran *Physical Data Model* (PDM) dapat dilihat pada Gambar 2.2.



Gambar 2. 2 *Physical Data Model* (PDM)

2.9 ICONIX Process

Berdasarkan pemaparan oleh Yulianta & Mursanto (2008), *ICONIX Process* memiliki tujuan utama untuk mewujudkan *use case* yang telah disusun menjadi kode, yaitu dari apa yang harus dilakukan sistem dengan menggambarkannya dalam

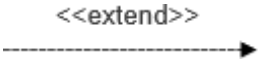
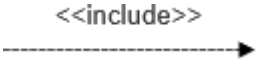
bentuk *use case* menjadi potongan-potongan kode yang komplit, telah diuji, dan bisa mengerjakan apa yang dituliskan pada *use case*. Qingxiong Ma (2019) menjelaskan bahwa *ICONIX* Process hanya memerlukan empat diagram dasar dari UML dalam empat tahapan proses yang dapat menjadikan *use case text* menjadi kode yang dapat bekerja. Oleh karena itu, Rosenberg & Stephens (2007) dalam Qingxiong Ma (2019) menyatakan bahwa *ICONIX Process* menjadi pendekatan yang lebih ringan daripada UML.

2.10 Use Case Diagram

Use Case menjelaskan bagaimana pengguna, yang digambarkan sebagai Aktor, akan berinteraksi dengan sistem, serta bagaimana sistem tersebut akan merespon kepada penggunanya (Ma, 2019). Menurut Yulianta & Mursanto (2008) *Use Case Modeling* dibutuhkan untuk mengetahui "Apa yang akan dilakukan oleh pengguna dari sistem?" dengan memberikan gambaran detail dari seluruh skenario yang akan dilakukan penggunanya terhadap sistem serta tanggapan yang akan diberikan oleh sistem tersebut. Bentuk penggambaran dari *use case modeling* tersebut adalah sebuah *use case diagram*. *Use case* sendiri memiliki beberapa komponen, yang dijelaskan dalam Tabel 2.1.

Tabel 2. 1 Komponen *Use Case Diagram*

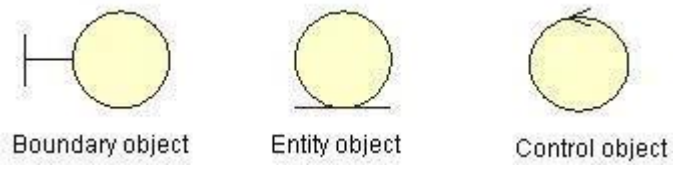
Simbol	Nama	Keterangan
	*Use Case*	Menggambarkan fungsionalitas yang disediakan oleh sistem sebagai unit yang saling bertukar pesan antar unit yang dinyatakan dengan menggunakan kata kerja
	*Actor*	Menggambarkan orang atau sistem yang lain. *Actor* berinteraksi dengan *use case*.

	Asosiasi / *Association*	Menggambarkan asosiasi atau hubungan antara *actor* dengan *use case*. Digambarkan dalam bentuh garis tanpa panah, yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung, dan bukan mengindikasikan data.
	*Extend*	Menggambarkan perluasan dari *use case* lain jika sebuah kondisi atau syarat terpenuhi. Menggambarkan relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri tanpa *use case* tambahan tersebut.
	*Include*	Menggambarkan pemanggilan *use case* oleh *use case* lain, seperti pemanggilan sebuah fungsi program. Menggambarkan relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan

		membutuhkan *use case* ini untuk dapat menjalankan fungsinya.
	Generalisasi / *Generalization*	Menggambarkan relasi antara *actor* dengan *use case* yang menggunakan panah terbuka untuk mengindikasikan apabila *actor* berinteraksi secara pasif dengan sistem.

2.11 Robustness Diagram

Robustness Analysis digunakan untuk memperkecil celah antara analisa dan desain, dengan beberapa peranan penting yaitu: uji kelayakan, uji kelengkapan, menemukan objek-objek baru, dan sebagai desain awal (Yulianta & Mursanto, 2008). Penggambaran dari *Robustness Analysis* yaitu dalam bentuk *Robustness Diagram*. Beberapa simbol dalam *Robustness Diagram* dijelaskan pada Gambar 2.3.

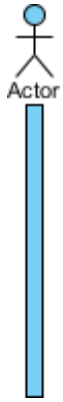
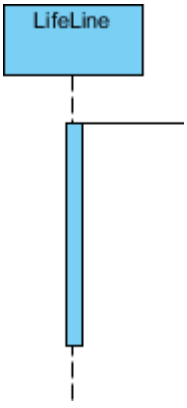
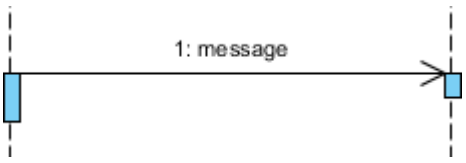


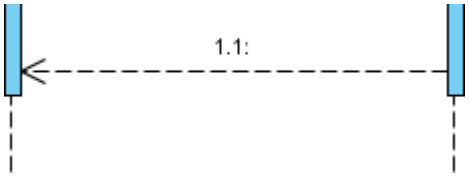
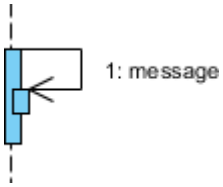
Gambar 2. 3 Komponen *Use Case Diagram*

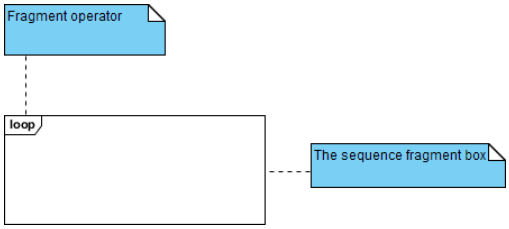
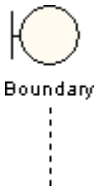
2.12 Sequence Diagram

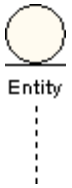
Perilaku dari objek pada *use case* digambarkan dalam *Sequence Diagram* dengan mendeskripsikan waktu hidup objek serta pesan apa yang dikirim dan diterima antar objeknya (Hendini, 2016). Komponen/simbol yang terdapat dalam *Sequence Diagram* dapat dilihat pada Tabel 2.2.

Tabel 2. 2 Komponen *Sequence Diagram*

Simbol	Nama	Keterangan
	*Actor*	Menggambarkan orang atau sistem yang lain. *Actor* berinteraksi dengan subjek, seperti contohnya bertukar data.
	*Lifeline*	Garis titik-titik yang terhubung dengan objek dan merepresentasikan partisipasi individual dalam interaksi.
	*Activations*	Menggambarkan durasi eksekusi operasi dari objek.
	*Call Message*	Menggambarkan komunikasi antar *Lifelines* dari sebuah interaksi yang erupakan jenis pesan yang mengirimkan

		permintaan pengoperasian terhadap *Lifeline* yang dituju.
	*Return Message*	Menggambarkan komunikasi antar *Lifelines* dari sebuah interaksi yang merupakan jenis pesan yang mengirimkan penyampaian informasi kembali ke pengirim pesan sebelumnya.
	*Self Message*	Menggambarkan komunikasi antar *Lifelines* dari sebuah interaksi yang merupakan jenis pesan yang menggambarkan pengiriman pesan ke *Lifeline* yang sama.
	*Recursive Message*	Menggambarkan komunikasi antar *Lifelines* dari sebuah interaksi yang merupakan jenis pesan yang menggambarkan

		pengiriman pesan ke *Lifeline* yang sama, namun pesan yang dikirim ditargetkan kepada dirinya sendiri.
	*Sequence Fragments*	Membuat *sequence diagram* lebih terjaga keakuratannya. Digambarkan dalam bentuk boks dan memilki beberapa jenis operator *fragment* yaitu: *ref*, *assert*, *loop*, *break*, *alt*, *opt*, dan *neg*.
	*Boundary Class*	Berisikan kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih *actor* dengan sistem.
	*Control Class*	Sebuah objek yang berisikan logika aplikasi yang tidak memiliki tanggung jawab kepada entitas.

	*Entity Class*	Bagian dari sistem yang berisikan kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.