

BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Beberapa penelitian terdahulu yang telah dilakukan mengenai rancang bangun sistem informasi pengelolaan data publikasi penelitian yaitu penelitian yang dilakukan oleh Setemen dan kawan-kawan pada tahun 2012 berupa pengembangan aplikasi pengelolaan karya ilmiah untuk mahasiswa dan dosen. Penelitian ini melakukan penelitian berupa pengembangan aplikasi pengelolaan karya ilmiah untuk mahasiswa dan dosen. Penelitian ini dikembangkan menggunakan CMS (*Content Management System*) *Ganesha Digital Library Versi 4.2* (GDL 4.2) yang dibuat oleh KMRG ITB serta rancangan penelitian dan pengembangan (R&D). Dalam penelitian ini menghasilkan sebuah sistem yang mampu menangani masalah pengelolaan karya akademik mahasiswa dan dosen meliputi laporan PKM, laporan PKL, laporan tugas akhir, laporan skripsi, tesis, disertasi, makalah dalam prosiding, artikel dalam jurnal, proposal penelitian dan laporan penelitian (Setemen, Dewi, & Mart, 2012) [2].

Penelitian kedua mengenai perancangan dan pembuatan aplikasi manajemen pengelolaan data publikasi penelitian juga dilakukan oleh Simaremare dan kawan-kawan pada tahun 2013. Perancangan aplikasi dalam penelitian ini dibangun dengan metode *Unified Software Development Process* dan perancangan aplikasi menggunakan *Unified Modeling Language* (UML). Dalam penelitian ini menghasilkan dokumen perancangan dan aplikasi manajemen publikasi ilmiah yang mampu meningkatkan kualitas dari proses pengelolaan jurnal ilmiah di Jurusan Sistem Informasi ITS menjadi lebih sistematis dan terorganisir (Simaremare, S, & Wibowo, 2013) [3].

Kemudian, Joni dan kawan-kawan pada tahun 2016 juga melakukan penelitian serupa mengenai sistem informasi manajemen sebagai alat pengelolaan penelitian dosen. Implementasi sistem informasi ini diterapkan pada STMIK STIKOM Indonesia (STIKI) yang merupakan salah satu perguruan tinggi swasta di Bali. Penelitian ini merupakan sistem yang dirancang dalam bentuk perangkat lunak untuk mengurangi tingkat kesalahan yang disebabkan *human error*.

Penelitian ini dirancang menggunakan model perancangan terstruktur dimulai dari *document flow diagram*, *system flow diagram*, *data flow diagram* sampai dengan *entity relationship diagram*. Dalam penelitian ini berhasil menunjukkan bahwa tingkat kesalahan yang disebabkan *human error* menjadi kecil (Joni & Sandika, 2016) [4].

Selanjutnya pada tahun 2018, Indri Handayani dan kawan-kawan juga melakukan penelitian mengenai pemanfaatan sistem *International Journal of Biosciences* (iJB) berbasis *Open Journal Systems* (OJS) sebagai media e-jurnal. Perancangan sistem ini menggunakan suatu metode pengumpulan data berupa observasi, wawancara dan studi pustaka serta menggunakan UML (*Unified Modeling Language*). Dalam penelitian ini menghasilkan sebuah sistem *e-journal* pada STISIP YUPPENTEK yang membantu pengguna dalam mengunggah jurnalnya dengan lebih mudah, cepat, dan hemat baik dari segi waktu, tenaga, serta biaya. Selain itu, penggunaan iJC (*iLearning Journal Center*) menjadikan pengelolaan *e-journal* meliputi proses penyerahan naskah, *review*, *edit*, dan mempublikasi dapat dilakukan dengan mudah sehingga menghasilkan sebuah jurnal *online* atau *e-journal* yang berkualitas (Handayani, Aini, & Sari, 2018) [5].

Rangkuman yang telah diambil dari penelitian-penelitian sebelumnya menjadi referensi dalam pembuatan Tugas Akhir untuk melakukan penelitian serupa namun dengan beberapa fitur, model, dan perancangan sistem yang berbeda. Penelitian ini akan dikembangkan di dalam kerangka kerja CodeIgniter dimana penelitian sebelumnya yang dilakukan oleh Hustinawati, Albert Kurnia Himawan, dan Latifah pada 2014 menunjukan bahwa kerangka kerja Codeigniter memiliki banyak keunggulan dibidang kemampuan pemisahan stuktur yang baik, sehingga mudah untuk melakukan pencarian dan perbaikan serta kerangka kerja ini menggunakan arsitektur MVC (Hustinawati, Himawan, & Latifah, 2014) [6].

Metode analisis dan perancangan perangkat lunak yang diterapkan dalam penelitian ini yaitu menggunakan metode RAD (*Rapid Application Development*). RAD ditandai dengan keterlibatan pengguna dalam jumlah besar, prototipe tambahan, dan manajemen proyek berbasis produk. Metode RAD juga dapat

menghemat waktu perancangan dan pembuatan sistem (Paul Beynon-Davies & Steve Holmes, 2002) [7].

Dari penjelasan sebelumnya Sistem Informasi Pengelolaan Data Penelitian Departemen Teknik Komputer Universitas Diponegoro akan dibuat dalam bentuk sistem informasi menggunakan kerangka kerja CodeIgniter, bahasa pemrograman PHP dan basis data MySQL serta pengembangannya menggunakan metode *Rapid Application Development* (RAD) dengan 4 *level* pengguna yaitu Admin, Operator, PAK/Reviewer, dan Dosen. Bagian *User Interface* (UI) secara keseluruhan akan berisikan halaman *login*, halaman beranda, halaman menambah *user* untuk level Admin, halaman membuat akun dan *update* akun dosen untuk *level* Operator, halaman *upload* jurnal untuk *level* dosen, halaman *review* dan penilaian jurnal untuk *level* PAK/Reviewer, dan *logout*.

2.2 Sistem Informasi

Sistem merupakan suatu kumpulan atau himpunan dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan kegiatan atau tujuan tertentu (Sutabari, 2012) [8]. Suatu organisasi atau bahkan bagian dari sistem komputer dapat digambarkan sebagai sebuah sistem. Sebuah sistem komputer terdiri dari komponen *hardware* yang bekerja sama dengan komponen *software* dan dijalankan dalam sebuah perangkat komputer (Rosso, 2017) [9].

Sedangkan informasi merupakan data mentah dari suatu organisasi atau perusahaan yang mana telah melalui proses pengolahan sehingga menjadi sesuatu yang berguna bagi manusia. Informasi ibarat darah yang mengalir dalam tubuh suatu organisasi sehingga informasi ini sangat penting di dalam suatu organisasi. Suatu sistem yang kurang mendapatkan informasi akan menjadi luruh, kerdil, dan akhirnya mati [8].

Maka sistem informasi merupakan suatu gabungan dari berbagai macam komponen yang saling bekerja sama untuk melakukan pencatatan dan pengolahan data dan menyajikannya menjadi suatu informasi agar pengguna dapat membuat suatu keputusan dengan baik (Hendini, 2016) [10].

2.3 Framework

Kerangka kerja atau *framework* adalah kumpulan perintah atau fungsi dasar yang membentuk aturan tertentu dan saling berinteraksi satu sama lain. Penggunaan *framework* bertujuan agar waktu pembuatan *website* menjadi lebih singkat dan pembuatan alur kode program lebih terarah (Somya, 2018) [11]. Kerangka adalah seperangkat perpustakaan yang disusun dalam desain arsitektur untuk memberikan kecepatan, akurasi, kenyamanan dan konsistensi dalam pengembangan aplikasi tersebut [5].

Belakangan, penggunaan kerangka kerja menjadi pilihan bagi para pengembang perangkat lunak untuk mengembangkan aplikasinya dikarenakan banyak kemudahan yang ditawarkan. Pada kerangka kerja sudah tersedia struktur aplikasi yang baik seperti *standard coding*, *best practice*, *design pattern* dan *common function*. Sehingga para pengembang aplikasi dapat lebih fokus kepada proses pembangunan sistem tanpa harus berpikir pada masalah struktur aplikasi, standar *coding*, dan sebagainya.

Dalam menggunakan *framework* untuk membuat sebuah program maka harus melakukan *coding* atau penulisan kode-kode program pada lingkungan *framework* tersebut. Maka para pengembang aplikasi perangkat lunak perlu menyadari dan memahami terlebih dahulu tentang alur atau lingkungan *framework* yang akan digunakan (Badiyanto & Murya, 2018) [12]. Dalam pengembangan aplikasi web, para pengembang perangkat lunak memiliki struktur penulisan kode yang berbeda misalnya menggunakan *web framework* yang disediakan oleh komunitas seperti CodeIgniter dengan arsitektur *framework* berbasis MVC (*Model View Controller*) (Widodo, 2013) [13].

Untuk menghasilkan sebuah *website* yang menarik tentu diperlukan sifat *website* yang dinamis. Bahasa pemrograman yang paling banyak digunakan dalam pembuatan *website* dinamis adalah PHP (Wardana, 2010) [14]. PHP singkatan dari *Hypertext Preprocessor* adalah bahasa *scripting* terkenal yang sering dikaitkan dengan pengembangan web meskipun memiliki area penggunaan lainnya. Menurut w3techs.com, PHP adalah bahasa *scripting* yang paling umum digunakan di internet dengan cakupan 82%. (Das & Saikia, 2016) [15].

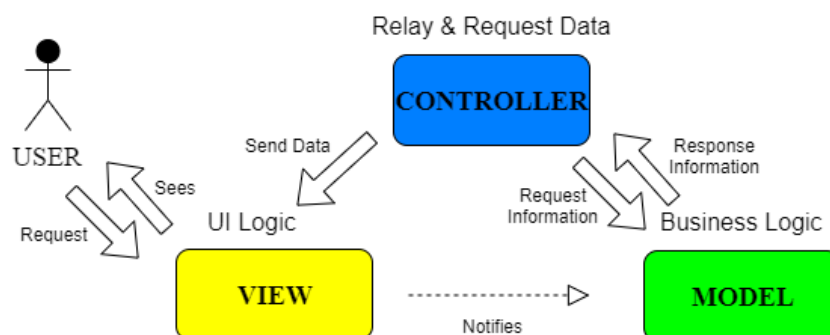
2.4 CodeIgniter

CodeIgniter dapat disebut sebagai *framework* pengembangan aplikasi (*Application Development Framework*) dengan menggunakan PHP sehingga menjadi sistematis karena pengembang tidak lagi membuat *website* dari awal (*from scrath*). CodeIgniter telah menyediakan *library* yang dapat dimanfaatkan untuk menyelesaikan pekerjaan yang umum, dengan menggunakan antarmuka dan struktur logika yang lebih sederhana untuk mengakses *library*.

Saat ini CodeIgniter dikembangkan oleh komunitas dan disebarluaskan ke seluruh dunia dengan lisensi bebas dengan ukuran kecil dan cukup mudah dipahami dan dikuasai. Penggunaan *framework* CodeIgniter yang mudah disesuaikan dan diintegrasikan dengan *library* menjadi keunggulannya. Selain itu, CodeIgniter merupakan *framework* yang bersifat gratis dengan lisensi Apache sehingga fapat dikembangkan dan digunakan sesuai dengan kebutuhan.

CodeIgniter sendiri dibangun menggunakan konsep MVC (*Model View Controller*) yang merupakan sebuah *pattern* atau teknik pemrograman yang memisahkan antara logika bisnis, penyimpanan data, dan antarmuka aplikasi. Secara sederhana dapat dikatakan bahwa antara desain dan proses data berada pada tempat yang terpisah [12]. Pemisahan dilakukan agar setiap perubahan dalam logika presentasi atau logika bisnis tidak saling memberi efek yang kompleks. Solusi pemisahan MVC diharapkan dapat meningkatkan fleksibilitas dan usabilitas aplikasi [5].

Arsitektur MVC memisahkan aplikasi menjadi tiga bagian, yaitu *Model*, *View* dan *Controller* seperti yang ditunjukkan pada Gambar 2.1.



Gambar 2.1 Arsitektur MVC

1. *Model*

Fungsi utama *model* adalah untuk menangani data, mengambil data dari basis data, memasukkan data ke dalam basis data, memanipulasi data melalui validasi data [5]. *Model* juga mempresentasikan struktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentuk berkas teks, XML, maupun *webservice*. Biasanya di dalam *model* berisi *class* tau fungsi untuk melakukan manipulasi data seperti *insert*, *update*, *delete*, dan *search* namun tidak dapat berhubungan dengan bagian *view* secara langsung. Sebuah aplikasi *website* biasanya menggunakan basis data dalam menyimpan data, oleh karena itu *model* biasanya akan berhubungan dengan perintah-perintah query SQL [12].

2. *View*

View berhubungan dengan segala sesuatu yang akan ditempatkan ke *end-user*. Bagian *view* dikhususkan hanya untuk menampilkan data-data hasil dari *model* dan *controller* [12]. Pengelompokkan semua tampilan dan kode presentasi di satu tempat ditujukan untuk memudahkan dalam mengubah tampilan tanpa memengaruhi logika bisnis dan data [5].

3. *Controller*

Controller merupakan penghubung antara *model* dan *view* dimana di dalamnya terdapat fungsi-fungsi yang akan memproses permintaan dari *view* menuju struktur data di *model*. Tugas dari *controller* adalah menyediakan variable yang akan ditampilkan di *view*, memanggil *model* untuk melakukan akses ke basis data, menyediakan penanganan *error*, mengerjakan proses logika dari aplikasi, serta melakukan validasi atau pengecekan terhadap *input* [12].

2.5 *Database*

Database merupakan representasi kumpulan fakta yang saling berhubungan dan disimpan dalam komputer secara sistematis untuk memenuhi berbagai kebutuhan. Database dapat dibuat dan diolah dengan menggunakan suatu program komputer, yaitu *software* (perangkat lunak). *Software* yang digunakan untuk mengelola dan memanggil *query database* disebut *database management system*

(DBMS) atau sistem manajemen basis data (Dr. H. A. Rusdiana & Moch. Irfan, 2014) [16]. Beberapa contoh dari DBMS antara lain MySQL, Microsoft SQL Server, Oracle, dan lain-lain.

Kehandalan suatu sistem basis data DBMS dapat diketahui dari cara kerja pengoptimasinya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. SQL (*Structured Query Language*) adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Turunan dari SQL adalah MySQL yang merupakan sebuah implementasi dari *Relational Database Management System* (RDBMS) (Februariyanti & Zuliarso, 2012) [17]. Selain itu, MySQL merupakan suatu aplikasi yang sifatnya *open source* serta *server* basis data MySQL memiliki kinerja sangat cepat, *reliable*, dan mudah untuk digunakan serta bekerja dengan arsitektur *client server* atau *embedded systems* (MySQL, 2020) [18]. Salah satu perangkat lunak *web server* yang didalamnya sudah tersedia basis data *server* MySQL dan dapat mendukung pemrograman PHP ialah XAMPP. XAMPP merupakan perangkat lunak yang mudah digunakan, gratis, dan mendukung instalasi di sistem operasi Linux dan Windows [17].

Pada jurnal penelitian yang dilakukan oleh Ramos Somya yang berjudul “Aplikasi Manajemen Proyek Berbasis *Framework* CodeIgniter dan Bootstrap di PT. Pura Barutama” menyimpulkan bahwa basis data MySQL merupakan infrastruktur DBMS yang cocok digunakan untuk menyimpan data karena lebih sederhana dalam penggunaannya dan dapat digunakan oleh beberapa *user* dalam waktu bersamaan [11].

Dari pemaparan diatas dapat disimpulkan bahwa pada sistem informasi yang akan dibuat memerlukan DBMS MySQL yang mampu mengolah data menjadi suatu informasi dengan cepat.

2.6 *Unified Modeling Language* (UML)

UML atau *Unified Modeling Language* merupakan kumpulan notasi grafis yang membantu dalam menggambarkan dan merancang sistem perangkat lunak,

khususnya sistem perangkat lunak yang dibangun menggunakan gaya berorientasi objek (*Object Oriented*). UML dapat digunakan dengan berbagai cara sesuai kebutuhan para pengembang perangkat lunak. Menurut Fowler, terdapat karakterisasi dari tiga mode di mana orang menggunakan UML, yaitu sebagai sketsa untuk membantu menampilkan beberapa aspek dari sistem, *blueprint* atau kerangka kerja yang terperinci, dan bahasa pemrograman untuk memodelkan logika perilaku dari sistem yang akan dibangun (Fowler, 2004) [19]. Menurut tiga pengembang model analisis berorientasi objek (Booch, Rumbaugh, dan Jacobson) UML menjadi standar yang efektif untuk pemodelan objek [16].

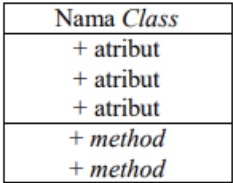
Di dalam UML, terdapat 13 tipe diagram yang diklasifikasikan menjadi 2 tipe diagram yaitu diagram struktur (*structure diagram*) dan diagram perilaku (*behavior diagram*) [19]. Pada penelitian ini menggunakan *class diagram* dan *deployment diagram* dari diagram struktur (*structure diagram*) sedangkan dari diagram perilaku (*behavior diagram*) menggunakan *use case diagram* dan *sequence diagram*.

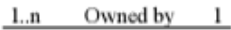



2.6.1 Class Diagram

Diagram kelas menggambarkan jenis objek dan berbagai macam hubungan antar kelas di dalam model desain dari suatu sistem. Diagram kelas juga menunjukkan atribut dan operasi-operasi dari sebuah kelas dan kendala yang berlaku pada cara objek terhubung [19]. Komponen atau simbol pembentuk diagram kelas dapat dilihat pada Tabel 2.1 dan Tabel 2.2.

Tabel 2.1 Simbol diagram kelas

Tabel 2.2 Simbol diagram kelas (lanjutan)

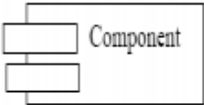
Simbol	Keterangan
	Kelas adalah atribut tunggal yang merupakan blok-blok pembangun pada pemrograman berorientasi objek. Sebuah <i>class</i> digambarkan sebagai sebuah kotak yang terbagi atas 3 bagian.



	<p><i>Association</i> merupakan relasi antara 2 kelas yang bermakna umum. Garis ini bisa melambangkan tipe-tipe relasi dan juga dapat menampilkan <i>multiplicity</i>. <i>Multiplicity</i> dari properti merupakan indikasi berapa banyak objek dapat mengisi properti.</p>
	<p><i>Dependency</i> digunakan untuk menunjukkan operasi pada suatu kelas yang menggunakan kelas yang lain.</p>
	<p><i>Composition</i> digunakan untuk <i>class</i> yang tidak bisa berdiri sendiri atau bergantung terhadap <i>class</i> lain</p>
	<p><i>Aggregation</i> merupakan relasi antar kelas yang memiliki makna semua-bagian (<i>whole-part</i>) dalam kelas.</p>

2.6.2 Deployment Diagram

Deployment Diagram menunjukkan tata letak fisik suatu sistem. *Deployment Diagram* digunakan untuk menggambarkan detail mengenai perangkat lunak berjalan pada perangkat keras tertentu [19]. Komponen atau simbol pembentuk *deployment diagram* dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol *deployment diagram*





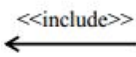
Simbol	Keterangan
	<p>Komponen digambarkan sebagai kotak dengan dua tab yang menunjukkan elemen perangkat</p>

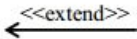
	lunak.
	<i>Node</i> merupakan objek perangkat lunak atau perangkat keras yang terhubung ke suatu sistem. Notasi untuk node digambarkan sebagai sebuah kubus 3 dimensi.
	Sebuah <i>association</i> digambarkan sebagai sebuah garis yang menghubungkan dua <i>node</i> yang mengindikasikan jalur komunikasi antara elemen-elemen <i>hardware</i> .

2.6.3 Use Case Diagram

Use Case Diagram merupakan teknik untuk membantu memahami perilaku fungsional suatu sistem. *Use case* bekerja dengan menggambarkan interaksi khusus antara pengguna sistem dan sistem itu sendiri serta memberikan narasi tentang bagaimana suatu sistem digunakan. Komponen atau simbol pembentuk *use case* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Simbol *use case diagram*




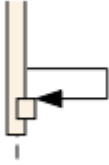

Simbol	Keterangan
	Disebut aktor yang merupakan peran yang dimainkan pengguna.
	Disebut <i>use case</i> yang merupakan abstraksi dan interaksi antara sistem dengan actor.
	Disebut <i>association</i> yang merupakan penghubung antara actor dengan suatu <i>use case</i> .
	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan <i>use case</i> .
	Menunjukkan bahwa suatu <i>use case</i> seluruhnya merupakan fungsionalitas dari

	<i>use case</i> lainnya.
	Menunjukkan bahwa suatu <i>use case</i> merupakan tambahan fungsional dari <i>use case</i> lainnya jika suatu kondisi terpenuhi.

2.6.4 Sequence Diagram

Sequence Diagram menggambarkan perilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek [19]. Komponen atau simbol pembentuk *use case* dapat dilihat pada Tabel 2.5.

Tabel 2.5 Simbol *sequence diagram*

Simbol	Keterangan
	Disebut <i>entity class</i> yang merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem.
	Disebut <i>boundary class</i> yang merupakan kumpulan kelas yang menjadi <i>interfaces</i> atau interaksi antara satu atau lebih aktor dengan sistem.
	Disebut <i>activation</i> yang merupakan perwakilan sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.
	Disebut <i>recursive</i> yang menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	Disebut <i>lifeline</i> yang merupakan garis titik-titik yang terhubung dengan objek, sepanjang <i>lifeline</i> terdapat <i>activation</i> .

2.7 Metode *Rapid Application Development* (RAD)

2.8 Usability Testing