

BAB II

LANDASAN TEORI

2.1 *Agile Development*

Agile Development adalah metode pengembangan perangkat lunak secara cepat dengan kondisi perubahan kebutuhan yang terjadi dalam waktu relatif singkat. Konsep utama *Agile Development* ialah pengerjaan aplikasi dan kerjasama tim. Fokus pengerjaan aplikasi dengan meminimalisir dokumentasi. Kerjasama tim berupa 2 orang programmer atau lebih yang bekerja dalam satu fitur serta komunikasi intensif programmer-klien. *Agile Development* berupa iterasi atau perulangan, tujuannya untuk merespon dan mengatasi setiap perubahan secara fleksibel, sehingga mengurangi waktu pengerjaan proyek dan mencapai kepuasan klien. Praktik *agile development* cocok digunakan pada proyek skala kecil dan dikerjakan oleh tim kecil pula [4].

Semua metodologi yang tergabung dalam *agile development* berlandaskan prinsip pada *agile manifesto* [14]. *Extreme Programming* (XP) dan SCRUM adalah contoh metodologi dalam *agile development*. Praktik SCRUM menitik beratkan manajemen tim pada iterasi (disebut sprint) proyek SCRUM. Faktor lain yang membedakan dengan XP adalah pada proyek SCRUM perubahan kebutuhan hanya boleh dilakukan pada akhir iterasi. XP disebut sebagai metodologi dengan siklus pengembangan pendek dimana iterasi dengan rentang waktu 1 hingga 2 minggu. Praktik XP didesain secara sederhana dengan fokus pada pengerjaan program aplikasi dalam waktu relatif singkat [4].

2.2 *Extreme Programming*

Extreme Programming (XP) adalah metodologi dikembangkan untuk proyek dengan skala tim kecil berjumlah 2 hingga 10 orang. XP diimplementasikan pada proyek yang memiliki kebutuhan samar dan mengalami banyak perubahan [8]. XP mengutamakan komunikasi dan *feedback*, sehingga tidak cocok untuk digunakan pada proyek skala besar dan sulit mendapatkan *feedback* [9]. Konsep dasar XP adalah menulis kode program dan pengujian. Terdapat 4 nilai utama yang mendasari XP sesuai pada [8].

- a. *Communication*, Praktik XP menggunakan komunikasi sebagai nilai penting demi kesuksesan proyek. *Pair programming*, *unit testing*, dan *estimating* adalah contoh praktik XP yang memungkinkan programmer, klien, dan manajer berkomunikasi selama proyek untuk mendapat hasil sesuai kebutuhan.
- b. *Simplicity*, setiap tahap XP dibuat sederhana. Membuat setiap bagian fungsi aplikasi sesuai kebutuhan saat ini, sehingga aplikasi selesai dalam waktu cepat.
- c. *Feedback*, XP membagi pengerjaan setiap fungsi dari aplikasi dan melakukan jadwal rilis jangka pendek untuk mendapat *feedback* dari klien lebih cepat. Jadwal rilis jangka pendek membuat klien dapat menguji program lebih cepat dan memberikan *feedback* dari hasil tersebut.
- d. *Courage*, Keberanian menjadi nilai penting XP dimana anggota tim berani dalam perbaikan kesalahan kode program dan *defect*. Termasuk dalam menghapus sebagian besar kode dan bekerja melebihi jam kerja untuk memperbaiki *defect* tersebut.

2.3 User Story

User Story adalah deskripsi singkat fungsionalitas, menyatakan kebutuhan klien, dan menggantikan *requirements* (*IEEE 830* dan *use case*). Perbedaan *requirements* pada waterfall dengan *user story* terletak dimana *requirements* menyatakan bagaimana sistem harus berfungsi, sebaliknya, *user story* menyatakan apa yang ingin dilakukan oleh pengguna. Selain itu dokumen *requirements* dibuat dan digunakan selama aplikasi berjalan. Sementara, *user story* akan dibuang setelah tahap pengembangan aplikasi selesai.

User story memiliki kelebihan yakni membantu komunikasi antara pengembang-klien, mudah dipahami kedua belah pihak karena tidak mengandung unsur teknis yang rumit, dan cocok untuk metodologi bersifat iterasi dimana kebutuhan banyak mengalami perubahan. Pada proyek XP, sebuah *user story* dapat ditulis, dilakukan implementasi kode, dan dilakukan pengujian secara langsung, sehingga tidak perlu menulis daftar secara keseluruhan seperti pada dokumen *requirements*. *User story* ditulis diatas kertas oleh klien dan untuk media diskusi antara pengembang-klien selama proses pengembangan aplikasi. Terdapat beberapa kriteria penulisan *user story* yaitu *negotiable*, *estimatable*, dan *testable* [15].

- a. *Negotiable*, sebuah *user story* dapat didiskusikan lebih lanjut apakah layak untuk dapat diimplementasikan, sehingga bukan sebuah kontrak yang harus dipenuhi dan diimplementasikan.
- b. *Estimatable*, sebuah *user story* harus dapat diperkirakan waktu yang dibutuhkan untuk proses implementasi.
- c. *Testable*, sebuah *user story* harus bisa diuji setelah diimplementasikan.

2.4 MoSCoW Method

MoSCoW *method* adalah penentuan prioritas kebutuhan dari sistem yang diimplementasikan. MoSCoW digunakan dalam metodologi pengembangan perangkat lunak DSDM. MoSCoW mudah dan sederhana dalam praktik, sehingga dapat membantu pihak klien untuk menentukan prioritas. MoSCoW sendiri terdiri atas beberapa sub kata yang menggambarkan kategori penentuan prioritas berdasarkan [12][13], yaitu

- a. *Must have*, adalah kebutuhan yang harus ada dan diimplementasikan dalam sistem. Jika kebutuhan pada kategori ini tidak ada, maka sistem dianggap gagal.
- b. *Should have*, kebutuhan pada kategori ini adalah dianggap wajib yang memiliki unsur nilai bisnis. Tidak seperti *must have*, sistem tetap dianggap sukses meski tidak diimplementasikan.
- c. *Could have*, adalah kebutuhan tambahan, jika sistem memiliki fitur ini maka akan menjadi pelengkap sistem.
- d. *Won't have*, semua kebutuhan yang dianggap belum dibutuhkan atau tidak berhubungan dengan sistem.

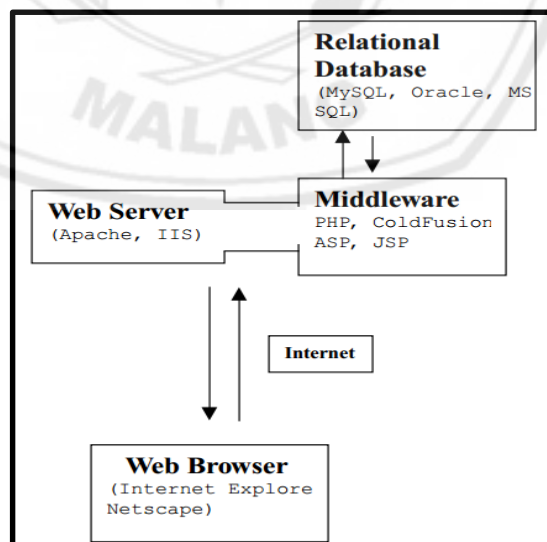
2.5 Library Information Management System

Library Information Management System adalah sistem yang memanfaatkan teknologi komputer sebagai tempat menyimpan data informasi buku dalam jumlah besar secara elektronik. Sistem ini terdiri dari tiga bagian yaitu, pembaca, pustakawan, dan admin sistem. Pembaca mampu melihat informasi keanggotaan, informasi buku, untuk tujuan dipinjam dan dikembalikan. Pustakawan sebagai pihak pengguna utama yang bertugas dalam manajemen buku dan pelayanan perpustakaan dalam sistem [16].

Beberapa tugas pustakawan dalam manajemen buku dan layanan diantaranya, pemesanan buku, penyusunan, menverifikasi buku koleksi baru, mendaftar bibliografi, register buku, register peminjaman dan pengembalian, membuat salinan register. Admin sistem bertugas untuk manajemen data pengguna dan buku, termasuk diantaranya meliputi manajemen informasi keanggotaan pembaca, dan pemeliharaan sistem. Pemeliharaan sistem menjadi bagian penting karena mencakup sistem log, hak akses pengguna, manajemen keanggotaan seperti menambah dan menghapus [16].

2.6 Aplikasi Web

Aplikasi web adalah aplikasi yang diakses menggunakan web browser melalui jaringan internet atau intranet yang memiliki dukungan HTML, JavaScript, AJAX, Java, dll. Pada dasarnya aplikasi web terdiri dari klien dan server. Dalam kinerja aplikasi web, web browser harus mendukung dan mampu memproses bahasa pemrograman utama *Hypertext Markup Language* (HTML), JavaScript, RealPlayer, Flash dan Shockwave. Server adalah tempat sebagian besar proses aplikasi web bekerja. Salah satu kerja server adalah komunikasi dengan browser dan *database* server menggunakan bahasa pemrograman tertentu. *Database* server adalah tempat menyimpan segala informasi yang dibutuhkan aplikasi web [17]. Arsitektur aplikasi web ditunjukkan pada gambar 2.1.



Gambar 2.1. Arsitektur Aplikasi Web [17]

Gambar 2.1 menunjukkan proses komunikasi yang terjadi di dalam aplikasi web. Web server sebagai bagian aplikasi web untuk menerima *request* yang dikirim

pengguna, lalu memproses dan mengirim respon *request*, kemudian menampilkan hasil *request* pengguna. Aplikasi web disarankan menggunakan PHP dan MySQL karena bersifat *open source*. Hal ini bertujuan agar setiap pengembang yang tertarik pada suatu proyek yang sedang dibangun dapat ikut berpartisipasi dalam proyek aplikasi web tersebut [17].

2.7 PHP

PHP adalah bahasa pemrograman untuk *server-side* sering digunakan untuk pembuatan Web. Dalam hal ini PHP membutuhkan dukungan dari web server, web browser, dan PHP parser (CGI) untuk menjalankan PHP dan menampilkan hasilnya dalam web browser. PHP dapat di sisipkan kedalam HTML dan berjalan bersamaan saat halaman web tersebut diakses. Penggunaan PHP parser membuat PHP dapat dijalankan meski tanpa browser atau server. Kelebihan PHP adalah fleksibel, artinya PHP saat ini dapat digunakan pada banyak macam web server (Apache, IIS dll) dan sistem operasi (Linux, Unix, Microsoft Windows, Mac OS X, dll). Selain itu PHP dapat berupa prosedural, *Object Oriented Programming* (OOP) atau gabungan keduanya [17].

PHP mendukung penyimpanan *database* dalam skala luas. Terdapat banyak jenis luaran PHP diantaranya, gambar, file PDF, dan video yang terintegrasi pada sistem *server-side*. Terlebih lagi, PHP mampu terhubung dengan servis lain melalui protokol HTTP, COM, dan POP3 serta pertukaran data dengan bahasa pemrograman web berbeda [17].

Proses kerja PHP dengan web server bermula pada browser mengirim *request* halaman web menuju web server. *Request* berisi informasi tentang browser, sebagai contoh nama browser, versi, atau sistem operasi yang digunakan serta informasi tambahan yang diketik oleh pengguna. Jika *request* ditujukan untuk file HTML, web server dapat dengan cepat menemukan file, mengembalikan respon ke browser, lalu mengirim konten file itu ke browser. Browser menangkap respon dan mulai membuka halaman web berdasarkan kode HTML di dalam konten file itu [18].

2.8 CodeIgniter

Sebuah proyek pengembangan aplikasi menggunakan framework untuk memudahkan programmer dalam menyusun fungsi dan desain kode program. Salah satu contoh framework berbasis PHP yang cepat dan mudah untuk

diimplementasikan. CodeIgniter digunakan karena statusnya open-source memiliki kemampuan separasi dan struktur kode menjadi tersusun rapi. Kelebihan lain adalah pemisahan kode fungsi program dengan kode tampilan sehingga meminimalisir kode dalam halaman web. Pemisahan itu didasari oleh konsep *Model-View-Controller* (MVC) [19].

2.8.1 Model

Model adalah bagian khusus menangani akses dan penyediaan data dari suatu sumber sebagai contoh adalah *database*. *Model* berisi baris perintah untuk digunakan proses manipulasi data. Struktur *model* dibuat sederhana, sekadar untuk memenuhi cakupan kebutuhan pengolahan data tahap awal. Sebaliknya, *controller* memegang peran penuh pada proses pengolahan data tahap lanjut yang dibutuhkan aplikasi.

2.8.2 View

Peran *view* adalah semua yang terkait dengan pengelolaan tampilan aplikasi. *View* juga disebut sebagai desain web atau *template*, digunakan untuk mengontrol tampilan hasil data dan antarmuka pengguna untuk berinteraksi dengan aplikasi. Penggunaan *view* dapat mengurangi beban programmer karena pemisahan kode *logic* program dan tampilan. Teknologi yang digunakan dalam *view* adalah HTML, CSS, JS, dan AJAX.

2.8.3 Controller

Controller adalah bagian untuk mengatur kinerja antara *model* dan *view*. Struktur *controller* berisi fungsi yang digunakan untuk mengatur aliran data dari *model* dan fungsi untuk menampilkannya ke *view*. Proses kerja *controller* bermula dengan menerima *request* dari pengguna melalui *view*, kemudian mengambil dan mengolah data yang diperlukan, lalu mengirimkan pada *view* dalam bentuk *response* kepada URL tempat *request* berasal.

2.9 Perbandingan Penelitian Sebelumnya

Tabel 2.1. Penelitian Sebelumnya.

| No | Nama | Tahun | Judul | Keterangan |
|----|--------------------|-------|---|--|
| 1 | Rotislav Fojtik | 2010 | Extreme Programming in development of specific software | Penelitian ini berupa pembuatan aplikasi interaksi untuk murid autis. Tahap XP untuk membangun aplikasi adalah <i>Planning and Managing, Designing Code generation, Testing</i> . Tahap <i>code generation</i> menggunakan TDD dan MS Visual Studio dimana kode ditulis 30% untuk dapat diuji. Metode pengujian menggunakan <i>Usability Testing</i> . |
| 2 | Hidayat Rizal, dkk | 2013 | Perancangan dan Pembuatan Mobile Learning Berbasis Android dengan Metode Personal Extreme Programming | Penelitian ini menerapkan PXP pada aplikasi android mulai dari menulis <i>user story</i> , estimasi, dan membagi <i>user story</i> ke dalam 4 iterasi. Terjadi satu perubahan <i>user story</i> yang kemudian ditampilkan dalam bentuk grafik <i>Iteration Burndown Chart</i> . |
| 3 | Kholid Haryono | 2015 | The Extreme Programming Approach for Financial Management System on Local Government | Penelitian ini menerapkan setiap tahap XP, dimulai dari membagi 2 tim antara klien dan pengembang hingga pengujian <i>acceptance test</i> dan kuisisioner untuk mendapat nilai kepuasan pelanggan. |

2.10 Kejaksaan Negeri Batu

Kejaksaan Republik Indonesia (RI) adalah salah satu lembaga pemerintah milik negara Indonesia. Undang-undang terkait kedudukan dan pelaksanaan

Kejaksaan diatur di dalam [11] memiliki kewenangan dan kekuasaan negara termasuk secara khusus terkait tugas penuntutan dan kehakiman. Kejaksaan RI dipimpin oleh Jaksa Agung secara langsung bertanggung jawab terhadap Presiden. Hakikat dasar Kejaksaan RI dalam susunan tata negara adalah yudikatif bertugas pada bagian kehakiman. Kejaksaan RI tersusun atas 3 bagian utama diantaranya Kejaksaan Agung, Kejaksaan Tinggi, dan Kejaksaan Negeri.

Kejaksaan Tinggi dan Kejaksaan Negeri sesuai [11] melaksanakan tugas penuntutan dan fungsi kehakiman di daerah wilayah yang telah ditetapkan. Kejaksaan Negeri Batu adalah salah satu contoh Kejaksaan Negeri yang berkedudukan di daerah Batu Jawa Timur. Kejaksaan Negeri Batu dipimpin oleh Kepala Kejaksaan Negeri mencakup bagian-bagian penting di dalam Kejaksaan Negeri. Bagian-bagian dalam Kejaksaan Negeri Batu meliputi Kasubbag Pembinaan, Kasi Intelijen, Kasi Pidum, Kasi Pidsus Kasi Datun, Kaur Kepegawaian, Kasi Keuangan, Kaur Perlengkapan, Kaur TU dan Perpustakaan.

Perpustakaan Kejaksaan Negeri Batu memiliki fungsi seperti perpustakaan pada umumnya. Fungsi Perpustakaan menyimpan dan menyediakan informasi-informasi seperti buku, dokumen, data-data untuk mendukung Kejaksaan Negeri Batu dalam pelaksanaan tugas hukum. Perbedaan Perpustakaan Kejaksaan Negeri Batu dengan perpustakaan umum adalah penyediaan dan akses hanya untuk pegawai saja.