BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Sudah banyak penelitian sebelumnya telah dilakukan untuk keamanan *RESTful API* dengan menggunakan autentikasi OAuth 2.0. Sejumlah penelitian tersebut telah berhasil memberikan hasil yang efektif dalam melindungi pertukaran data. Hasil-hasil penelitian sebelumnya juga mengindikasikan bahwa pengamanan dapat memberikan dampak positif dalam menjaga keamanan informasi. Berikut rincian dari beberapa penelitian terdahulu yang relevan dengan penelitian yang akan dilakukan.

Pertama, penelitian yang berjudul "Penerapan *RESTful API* untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi Oauth 2.0" dilakukan oleh Nina Wulandari , Argo Wibowo , Budi Susanto pada tahun 2021 [4]. Penelitian dilakukan untuk menerapkan otentikasi Oauth 2.0 pada program Pembayaran Piutang. Penelitian ini mengungkapkan bahwa penerapan autentikasi Oauth 2.0 lebih menguntungkan dibandingkan dengan *Basic Auth* dan memberikan dampak positif terhadap kinerja server dalam menangani permintaan. Dari hasil pengujian, meskipun diberikan jumlah permintaan yang cukup banyak yaitu 100 pengguna, *server* tetap merespons dengan cepat dengan rata-rata waktu respon 1,03 detik.

Kedua, penelitian yang berjudul "Implementasi *Restful Web Services* Dengan Otorisasi Oauth 2.0 Pada Sistem Pembayaran Parkir" dilakukan oleh Indra Kusuma, Ajib Susanto, Ibnu Utomo Wahyu Mulyono pada tahun 2019 [9]. Penelitian ini menghadirkan hasil implementasi yang dalam pengembangan aplikasi *web service* berbasis Android dengan fokus pada otoritasi *Oauth 2.0*. Optimasi dilakukan untuk meningkatkan keamanan transaksi data. Penelitian ini mengungkapkan bahwa penggunaan autentikasi

Oauth 2.0 lebih menguntungkan dibandingkan dengan otorisasi *API* standar seperti *Basic Auth* atau *API Key*. Penelitian ini mencakup pengujian sistem yang telah terintegrasi dengan baik menggunakan metode *black-box testing* dan *white-box testing*. Selain itu, berbagai tahap validasi, termasuk *client credentials, access token, dan scope*, telah dilakukan untuk memastikan akses yang sesuai terhadap fungsi yang disediakan oleh *API* aplikasi tersebut sesuai dengan protokol *OAuth 2.0*. Setelah dilakukannya penelitian tersebut, dapat disimpulkan bahwa penggunaan OAuth 2.0 lebih aman dibandingkan dengan metode otorisasi *API* standar seperti *basic-auth API* atau *API Key*.

Ketiga, penelitian yang berjudul "Perancangan dan Implementasi Sistem Otentikasi Oauth 2.0 dan *PKCE* Berbasis *Extreme Programming* (XP)" dilakukan oleh Rahmat Kurniawan pada tahun 2022 [10]. Penelitian bertujuan untuk merancang dan menerapkan autenikasi dengan menggunakan *Oauth 2.0* dan *PKCE* pada aplikasi HMS *multitenant*, melibatkan antara server dan juga client untuk melakukan sebuah proses autentikasi agar mempermudah sebuah proses autentikasi pada setiap tenant. Penelitian ini menggunakan metode pengembangan *Extreme Programming*. Dengan hasil dari penelitian adalah dengan penggunaan autentikasi *Oauth 2.0* peneliti dapat dengan gampang untuk menghapus hak untuk mengakses sebuah *client* tanpa harus mengubah *source code* pada sisi *client*. Pada penelitian ini juga dibuktikan bahwa penerapan *Oauth 2.0* jauh lebih aman apabila dibandingkan dengan menggunakan *Basic Auth* ataupun *API Key*. Dengan menggunakan *Oauth 2.0*, client tidak perlu membuat sistem *login* sendiri untuk dapat memvalidasi akun *user*.

Keempat, penelitian yang berjudul "Implementasi *Single Sign-On* Menggunakan *Google Identity, REST* dan Oauth *2.0* Berbasis *Scrum*" dilakukan oleh I Kadek Dendy Senapartha pada tahun 2021 [11]. Penelitian ini bertujuan untuk mengimplementasikan *Single Sign-On* dengan menggunakan *Google Identity*, *REST* dan OAuth 2.0 pada sistem *Student Relationship Management (SRM)* dengan menggunakan metode berbasis *Scrum* agar dapat meningkatkan kenyamanan dan keamanan akses pengguna pada sistemnya.

Hasil dari penelitian ini adalah Implementasi *SSO* telah berhasil diimplementasikan dalam sistem SRM menggunakan protokol *OAuth 2.0* dengan *Google Identity* dan arsitektur *REST*. Implementasi ini memudahkan pengguna aplikasi Android dan web mengakses sistem tanpa perlu login berulang-ulang. Mekanisme *refresh token* dalam *OAuth 2.0* memastikan keabsahan akses pengguna dengan memeriksa otentikasi, otorisasi, dan validitas *token* secara berkala, sehingga menjaga keamanan dan kontinuitas akses. Penggunaan layanan *Google Identity* mempercepat dan menyederhanakan proses implementasi *SSO* dalam sistem, memungkinkan pengembang untuk fokus pada keamanan transmisi data *SSO*. Selain itu, arsitektur *REST* yang menggunakan format data *JSON* mempercepat komunikasi antara *client* dan *server*. *JSON* menyediakan mekanisme komunikasi yang sederhana, sehingga meminimalkan waktu yang diperlukan untuk mengirim dan menerima data, serta memudahkan penguraian dan pemrosesan data oleh kedua belah pihak.

Kelima, penelitian yang berjudul "Implementasi *Single Sign On* Dengan *Oauth 2.0* Untuk Efektifitas *Login*" dilakukan oleh Mukrodin, Rohmat Taufiq, Deskal Dwi Rayananda pada tahun 2023 [12]. Penelitian ini bertujuan untuk meningkatkan efektivitas *login* pengguna serta mempermudah proses *login* agar tidak perlu khawatir tentang lupa akun atau *password* dengan menggunakan metode *Single Sign On (SSO)* dari *OAuth 2.0*. Dengan *SSO*, pengguna hanya memerlukan satu akun untuk mengakses berbagai sistem yang berbeda, sehingga mengurangi kerumitan dalam pengelolaan banyak kredensial. Penelitian ini menggunakan metode *Rapid Application Development* (RAD), yang memungkinkan pengembangan sistem dilakukan dengan cepat dan efisien melalui iterasi yang singkat. Hasil penelitian menunjukkan bahwa penerapan OAuth 2.0 berhasil dilakukan, memudahkan pengguna untuk *login* ke berbagai sistem hanya dengan satu akun. Mekanisme OAuth 2.0 memastikan keabsahan akses dengan memeriksa otorisasi dan autentikasi secara terpusat dan memvalidasi *access token* secara berkala.

Tabel 2. 1 Penelitian Terdahulu

No	Judul	Penulis	Tahun	Penerbit	Metode / Algoritma	Hasil	Perbedaan dengan penelitian yang dilakukan
1	Penerapan *RESTful API* untuk Membangun Program Pembayaran Piutang Menggunakan Otentikasi *Oauth 2.0*	Nina Wulandari , Argo Wibowo , Budi Susanto	2021	JUTEI Edisi Volume.5 No.1 April 2021 ISSN 2579-3675, e-ISSN 2579-5538 DOI 10.21460/jutei .2021.51.230	Metode penelitian pengumpulan data, perancangan sistem, kebutuhan fungsi, rancangan masukan, rancangan hasil.	Penelitian ini menunjukkan penggunaan autentikasi *Oauth 2.0* lebih menguntungkan daripada menggunakan *Basic Auth* dan berpengaruh positif terhadap kinerja server dalam menangani *request*. Dapat dilihat pada	Penelitian sebelumnya bertujuan untuk menguji dan membandingkan apakah penggunaan *OAuth 2.0* lebih menguntungkan dibandingkan dengan menggunakan Basic Auth saja. Lalu penelitian ini *Oauth 2.0* digunakan sebagai autentikasi

Tabel 2. 2 Lanjutan tabel Penelitian Terdahulu

No	Judul	Penulis	Tahun	Penerbit	Metode / Algoritma	Hasil	Perbedaan dengan penelitian yang dilakukan
						pengujian bahwa meski diberi request yang dapat dibilang banyak yaitu 100 user, respon dari *server*nya terhitung cepat dengan rata-rata 1,03 Detik	*web*, sedangkan pada penelitian ini digunakan sebagai autentikasi pada mobile.
2	Implementasi *Restful Web Services* Dengan Otorisasi *Oauth 2.0* Pada Sistem Pembayaran Parkir	Indra Kusuma, Ajib Susanto, Ibnu Utomo Wahyu Mulyono	2019	Jurnal SIMETRIS, Vol. 10 No. 1 April 2019 P-ISSN: 2252-	Metode *Extreme Programming*	Setelah penelitian tersebut dilakukan, dapat disimpulkan bahwa penggunaan *OAuth 2.0* lebih aman dibandingkan	Pada penelitian sebelumnya membandingkan antara *OAuth 2.0* dan *basic-auth*, sedangkan pada

Tabel 2. 3 Lanjutan tabel Penelitian Terdahulu

No	Judul	Penulis	Tahun	Penerbit	Metode / Algoritma	Hasil	Perbedaan dengan penelitian yang dilakukan
				4983, E-ISSN: 2549-3108		dengan metode otorisasi *API* standar seperti *Basic Auth* atau *API Key*.	penelitian ini pengujian akan dilakukan dengan cara penetrasi *REST API* yang telah dibuat
3	Perancangan dan Implementasi Sistem Otentikasi Oauth 2.0 dan PKCE Berbasis Extreme Programming (XP)	Rahmat Kurniawan	2022	Jurnal Pendidikan dan Teknologi Indonesia (JPTI), Vol. 2, No. 2, Februari 2022, Hal. 81-91	Metode *Extreme Programming*	Penelitian penggunaan otentikasi *OAuth 2.0*, hak akses untuk sebuah klien dapat dengan mudah dicabut tanpa perlu mengubah kode sumber di sisi klien.	Penelitian sebelumnya menggunakan OAuth 2.0, PKCE, dan Extreme Programming. Penelitian ini hanya menggunakan OAuth 2.0.

Tabel 2. 4 Lanjutan tabel Penelitian Terdahulu

No	Judul	Penulis	Tahun	Penerbit	Metode / Algoritma	Hasil	Perbedaan dengan penelitian yang dilakukan
						Implementasi *OAuth 2.0* menawarkan keamanan yang lebih tinggi dibandingkan dengan metode otorisasi *API* standar seperti *basic-auth API* atau *API Key*.	*2.0* pada aplikasi *mobile*
4	Implementasi *Single Sign-On* Menggunakan *Google Identity, REST* dan *Oauth 2.0* Berbasis *Scrum*	I Kadek Dendy Senapartha	2021	JuTISI, Volume 7 Nomor 2 Agustus 2021	Metode *Scrum*	Implementasi SSO diimplementasikan dalam sistem SRM menggunakan *OAuth 2.0* dengan *Google Identity* dan	Sebelumnya peneliti menggunakan metode *Scrum* untuk pengembangannya dan *Oauth 2.0* digunakan untuk

Tabel 2. 5 Lanjutan tabel Penelitian Terdahulu

No	Judul	Penulis	Tahun	Penerbit	Metode / Algoritma	Hasil	Perbedaan dengan penelitian yang dilakukan
						arsitektur *REST*. Implementasi ini memudahkan akses sistem tanpa login berulang, dengan mekanisme refresh token OAuth 2.0 memastikan akses tetap valid.	autentikasi pada *web* dan *mobile*, Penelitian ini menggunakan metode XP untuk pengembangan dan OAuth 2.0 untuk autentikasi pada mobile.
5	Implementasi *Single Sign On* Dengan Oauth 2.0	Mukrodin, Rohmat Taufiq,	2023	JT : Jurnal Teknik, Vol. 12 No. 02 Th. 2023, P-ISSN:	Metode *RAD*	Hasil dari penelitian ini menunjukkan bahwa Sistem *SSO* berhasil diterapkan	Penelitian sebelumnya menguji penggunaan *Oauth 2.0* sebagai

Tabel 2. 6 Lanjutan tabel Penelitian Terdahulu

No	Judul	Penulis	Tahun	Penerbit	Metode / Algoritma	Hasil	Perbedaan dengan penelitian yang dilakukan
	Untuk Efektifitas *Login*	Deskal Dwi Rayananda		2302-8734 E-ISSN: 2581-0006		Dengan menggunakan OAuth 2.0, pengguna dapat dengan mudah login ke beberapa sistem hanya dengan satu akun. Ini mempermudah autentikasi terpusat dan meningkatkan keamanan dengan mengurangi jumlah kredensial yang dikelola pengguna.	autentikasi dapat lebih meningkatkan efektifitas *login,* sedangkan pada penelitian ini *OAuth 2.0* digunakan untuk autentikasi *android* dan akan diuji dengan penetrasi

2.2 Landasan Teori

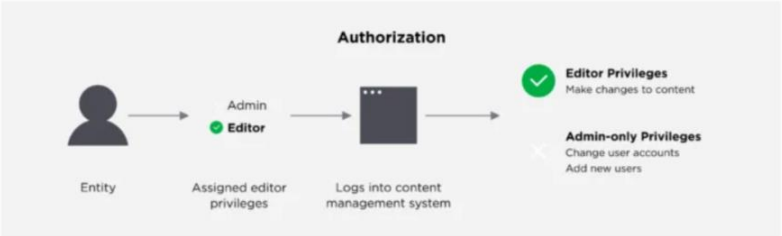
2.2.1 *Supply Chain Management*

Supply Chain Management adalah sistem atau rangkaian pasokan yang terdiri dari serangkaian aktivitas, informasi, organisasi, dan sumber daya yang bekerja bersama secara terkoordinasi untuk mengalirkan produk dari pemasok ke pelanggan [13]. *Supply Chain Management* (SCM) sendiri dapat dikatakan sebuah strategi perusahaan yang terfokus pada pengelolaan dan pengaturan semua aspek bisnis yang terkait dengan aliran barang dari penyedia hingga pelanggan akhir. SCM adalah konsep yang melibatkan integrasi dan koordinasi sistematis dalam perencanaan, desain, dan pengendalian aliran informasi dan material, dengan tujuan agar produk dapat disampaikan kepada konsumen dengan efisien dan akurat dalam waktu yang sesuai. Ini melibatkan pengelolaan seluruh rantai pasokan dengan hati-hati untuk memastikan bahwa barang-barang mencapai tangan pelanggan dengan kecepatan dan ketepatan yang optimal [14].

2.2.2 Otentikasi dan Otorisasi

Otentikasi adalah proses identifikasi oleh setiap pihak yang berkomunikasi untuk membangun kepercayaan penuh antara pengirim dan penerima informasi [15]. Sedangkan Otorisasi adalah proses yang menentukan apakah seorang pengguna memiliki hak untuk mengakses sistem atau sumber daya tertentu. Proses ini memastikan bahwa hanya pengguna yang memiliki izin yang sesuai dapat mengakses data, aplikasi, atau fungsi tertentu dalam sistem, sehingga menjaga keamanan dan integritas sistem tersebut [16].

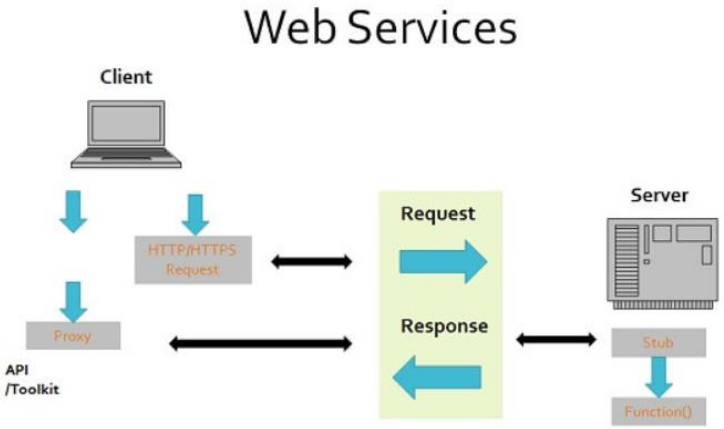
Gambar 2. 1 Otentikasi [17]



Gambar 2. 2 Otorisasi [17]

2.2.3 RESTful API

REST atau *RESTful* (*Representational State Transfer*) adalah salah satu varian dari *web service*. REST memungkinkan sistem pengirim permintaan untuk mengakses dan memanipulasi data yang direpresentasikan dalam bentuk teks melalui *web service*. *Web service API* yang mengadopsi pendekatan *REST* dikenal sebagai RESTful API. Berbeda dengan jenis *web service* lainnya, *RESTful API* tidak memiliki notasi resmi yang standar karena *REST* sendiri adalah suatu arsitektur[18]. REST API memfasilitasi komunikasi antara berbagai sistem dengan cara yang sangat sederhana, memungkinkan pertukaran data. Setiap panggilan *API REST* memiliki keterkaitan antara metode *HTTP* dan *URL* yang digunakan [19].



Gambar 2. 3 Alur *Web Service* [17]

2.2.3.1 OAuth 2.0

Oauth 2.0 adalah sebuah protokol yang memungkinkan aplikasi pihak ketiga untuk mendapatkan akses terbatas ke aplikasi lain dengan

mengizinkan aplikasi pihak ketiga tersebut untuk mengakses aplikasi tersebut atas nama pengguna yang terkait [20]. Dengan menggunakan metode *Single Sign-On* (SSO) dan memanfaatkan autentikasi *OAuth 2.0* dalam framework Laravel, pengguna memiliki kemampuan untuk masuk ke sistem yang dimiliki dengan satu akun tunggal [12]. *Oauth 2.0* menggunakan format data *JSON*, yang membuat integrasi dengan aplikasi modern menjadi lebih sederhana. Salah satu fitur krusial dari *Oauth 2.0* adalah kemampuannya untuk beradaptasi dengan berbagai lingkungan *runtime* dengan fleksibilitas yang tinggi [21]. *OAuth 2.0* memiliki beberapa konteks yaitu *Access Token, Authorization Code, Authorization Server, Client, Grant, Resource Server, Resource Owner*.

2.2.3.2 Framework

Framework adalah komponen pemrograman atau kerangka kerja yang dapat digunakan kembali kapan saja, guna menghindari kebutuhan *programmer* untuk membuat ulang skrip yang sama untuk tugas yang serupa. Misalnya, jika seorang *programmer* ingin membuat halaman *web* yang menampilkan data dengan paginasi, *framework* sudah menyediakan fungsi untuk paginasi tersebut. Programmer hanya perlu mengimplementasikan fungsi ini dalam kode mereka, dengan mematuhi pedoman-pedoman yang telah ditetapkan oleh *framework* yang digunakan [22].

2.2.3.3 Laravel

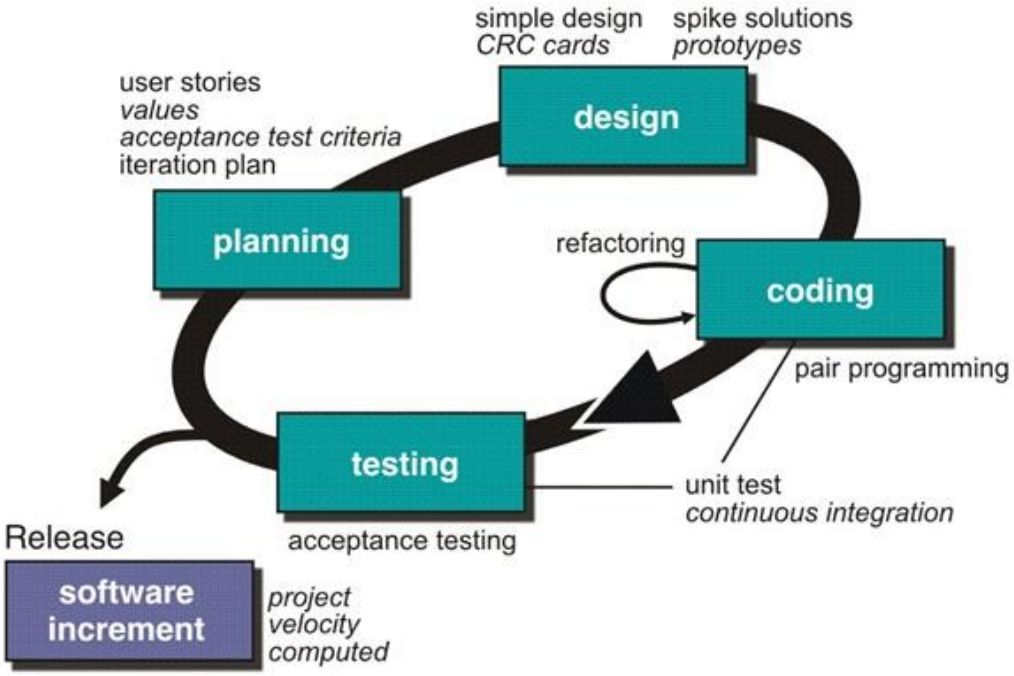
Laravel adalah sebuah *framework* yang dipakai untuk membangun aplikasi *web* menggunakan bahasa pemrograman *PHP*. *Laravel* termasuk salah satu *framework PHP* yang sangat populer dalam pengembangan aplikasi sisi *server* dengan *PHP*. *Framework Laravel* menyediakan berbagai fitur yang kuat yang dapat mempercepat proses pembuatan aplikasi *web* atau sistem informasi, dan dapat digunakan baik sebagai bagian *backend* maupun *frontend* dari suatu aplikasi, atau bahkan hanya sebagai *backend* saja [23].

2.2.3.4 PHP

PHP, atau kepanjangan dari *Hypertext Preprocessor*, adalah bahasa pemrograman yang terbuka yang amat sangat sesuai dan rancangannya dikhususkan pada pengembangan web. *PHP* dapat disisipkan dalam dokumen *HTML*. Bahasa *PHP* menggabungkan fitur-fitur dari berbagai bahasa pemrograman yang mencakup seperti *C, Java,* dan *Perl*, dan relatif mudah untuk mempelajarinya. *PHP* merupakan sebuah bahasa pemrograman *server-side scripting*, di mana pemrosesan data dilakukan di sisi server [24].

2.2.4 Extreme Programming (XP)

Extreme Programming (XP) adalah suatu pendekatan dalam rekayasa perangkat lunak melibatkan proses dengan pendekatan yang berorientasi dengan objek, dirancang khusus untuk tim yang berukuran kecil hingga tim yang berukuran menengah. Metode ini sangat sesuai digunakan dalam situasi di mana persyaratan proyek tidak dapat dengan jelas ditentukan atau sering mengalami perubahan yang cepat. Dalam kondisi tersebut, metode ini memungkinkan tim pengembang untuk tetap fleksibel dan adaptif terhadap perubahan yang terjadi. Proses iteratif dan inkremental yang diterapkan membantu tim untuk terus mengembangkan dan memperbaiki sistem sesuai dengan kebutuhan yang berkembang, tanpa terikat pada rencana yang kaku. Selain itu, metode ini juga memungkinkan untuk menerima umpan balik dari pengguna atau pemangku kepentingan secara lebih cepat, sehingga perbaikan dan penyesuaian dapat dilakukan dengan segera sesuai dengan kebutuhan proyek yang dinamis. XP menekankan fleksibilitas dan adaptabilitas dalam menghadapi dinamika proyek, menjadikannya solusi efektif untuk kondisi di mana kejelasan persyaratan proyek sulit dicapai atau dapat berubah dengan cepat. Dengan pendekatan iteratif dan inkremental, XP memungkinkan pengembang menyesuaikan dan mengembangkan sistem sesuai kebutuhan yang muncul, serta mendukung kolaborasi erat dengan pemangku kepentingan untuk memastikan sistem tetap relevan dengan perubahan kebutuhan [25]. Berikut adalah gambar dari metode *Extreme Programming* [5]:



Gambar 2. 4 **Extreme Programming** [26]**.**

Extreme Programming memiliki 4 tahapan yaitu :

1. *Planning* (Perencanaan)

   Pada tahap ini, peneliti bersama dengan pemangku kepentingan melakukan perencanaan proyek. Peneliti menetapkan fitur atau fungsi apa yang akan dikembangkan selama iterasi, menetapkan prioritas, dan membuat perkiraan waktu yang dibutuhkan.

2. *Design* (Perancangan)

   Tahap desain melibatkan pengembangan struktur dan arsitektur perangkat lunak. Fokus pada kejelasan dan desain yang sederhana, yang dapat disesuaikan dengan perubahan kebutuhan proyek.

3. *Coding* (Pengkodean)

   Pada tahap ini, peneliti mulai menulis kode untuk implementasi fitur atau fungsi yang telah direncanakan. XP menekankan pada pengkodean bersama *(pair programming*) dan pengujian terus-menerus untuk memastikan kualitas kode.

4. *Testing* (Pengujian)

   Pengujian dalam XP dilakukan secara kontinu sepanjang siklus pengembangan. Pengujian untuk memastikan bahwa setiap fitur berfungsi seperti yang diinginkan dan tidak menghasilkan regresi pada bagian lain dari sistem.

5. *Release* (Peluncuran)

   Pada tahap ini, dilakukan penelitian untuk memperlihatkan aplikasi yang telah dibuat kepada perusahaan yang akan menggunakan fitur tersebut.

2.2.5 Unit Testing

*Unit testin*g adalah metode untuk mengevaluasi keefektifan dan keandalan kode program perangkat lunak. Dalam banyak tahap pengembangan perangkat lunak, seringkali terjadi bahwa kode yang dibuat tidak memenuhi standar yang diinginkan atau bahkan tidak pernah digunakan sama sekali. Jenis pengujian ini bertujuan untuk mendeteksi dan mengidentifikasi potensi ketidakefektifan dalam kode *program*, serta mengukur sejauh mana kode tersebut berkontribusi pada pengembangan perangkat lunak secara keseluruhan. Selain itu, pengujian unit ini menjadi alat yang sangat berguna bagi para pengembang untuk menemukan dan memperbaiki kesalahan dan kelemahan yang mungkin tidak terlihat saat program dijalankan [27].

2.2.5.1 Black Box

Black Box Testing adalah salah satu metode pengujian yang fokus pada spesifikasi fungsionalitas dari perangkat lunak [28]. Pengujian *Black Box* difokuskan pada memastikan bahwa setiap proses dalam sistem beroperasi sesuai dengan kebutuhan yang diharapkan. Dalam metode ini, penguji mengidentifikasi berbagai kondisi masukan yang mungkin terjadi dan menjalankan pengujian terhadap fungsionalitas spesifik dalam sistem. Dengan demikian, pengujian berperan sebagai metode pelaksanaan program yang dirancang untuk mendeteksi kesalahan atau kesalahan potensial, dan kemudian memperbaikinya sehingga sistem menjadi layak dan andal untuk digunakan [29].

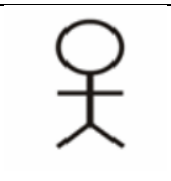
2.2.6 Unified Modeling Language (UML)

Unified Modeling Language (UML) merupakan sebuah alat pemodelan yang sangat kuat dan umum digunakan dalam dunia pengembangan sistem berorientasi objek [30]. *UML* membantu memudahkan komunikasi antar pengembang dan *client*. *UML* banyak digunakan di berbagai industri untuk mendefinisikan persyaratan (*requirements*), melakukan analisis, perancangan, serta menggambarkan arsitektur dengan konteks pemrograman berorientasi objek [31].

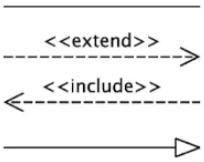
2.2.6.1 Use Case Diagram

Use Case Diagram adalah representasi visual dari interaksi antara aktor (pemain atau entitas yang berinteraksi dengan sistem) dan skenario penggunaan sistem (*use cases*). Fungsinya terletak pada analisis mendalam dan perancangan yang komprehensif dari suatu sistem dengan tujuan menggambarkan secara detail bagaimana aktor berinteraksi dengan berbagai kasus penggunaan yang terjadi. Dengan menggunakan diagram ini, informasi yang diperoleh tentang hubungan dan interaksi antara elemen-elemen sistem memungkinkan untuk memahami serta merancang fungsionalitas sistem secara lebih efektif. [32].

Tabel 2. 7 Komponen *Use Case Diagram* [33]**.**

Simbol	Nama
	Aktor
	*Use Case*

Tabel 2. 8 Lanjutan tabel Komponen Use Case Diagram

Simbol	Nama
	Relasi

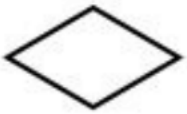
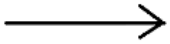
2.2.6.2 Activity Diagram

Activity Diagram adalah suatu representasi visual yang memodelkan serangkaian proses yang terjadi dalam sebuah sistem. Dalam diagram ini, urutan langkah-langkah proses dari suatu sistem digambarkan secara vertikal, memberikan gambaran alur kerja sistem dengan jelas.. *Activity diagram* sendiri merupakan perluasan dari konsep *Use Case*, yang menitikberatkan pada penciptaan representasi visual dari alur aktivitas yang terjadi dalam suatu proses atau skenario khusus dalam sistem tersebut [34].

Tabel 2. 9 Komponen *Activity Diagram* [33]

Simbol	Nama	Keterangan
	*Initial State*	Status Awal dari sebuah diagram
	*Activity*	Aktivitas yang dilakukan dalam suatu sistem

Tabel 2. 10 Lanjutan tabel Komponen Activity Diagram

Simbol	Nama	Keterangan
	Percabangan	Percabangan saat pilihan aktivitas lebih dari satu
	Penggabungan	Penggabungan lebih dari satu aktivitas menjadi satu
	*Final State*	Status akhir dari sebuah diagram aktivitas
	*Control Flow*	Arus Aktivitas

2.2.6.3 Sequence Diagram

Sequence Diagram adalah alat visual yang digunakan untuk menggambarkan dan menunjukkan interaksi antara objek-objek dalam suatu sistem secara mendetail. Diagram ini juga memperlihatkan pesan atau perintah yang dikirimkan antara objek-objek tersebut, beserta dengan waktu pelaksanaannya. Dengan menggunakan diagram ini, dapat dipahami bagaimana objek-objek berkomunikasi dan berinteraksi satu sama lain dalam konteks yang sangat terperinci[34].

Tabel 2. 11 Komponen *Sequence Diagram* [33]

Nama	Simbol	Fungsi
Object		Merepresentasikan sebuah *class* atau *object*

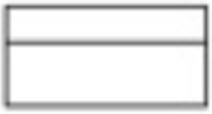
Tabel 2. 12 Lanjutan tabel Komponen Sequence Diagram

Nama	Simbol	Fungsi
Activity Boxes		Panjang waktu yang dibutuhkan *object*
Actors		Pengguna yang berinteraksi dengan sistem
Lifeline		Kehidupan suatu objek
Message		Interaksi antar *object*
Message Call		Memanggil operasi yang ada pada objek lain atau dirinya sendiri

2.2.6.4 Class Diagram

Class Diagram merupakan salah satu jenis diagram struktur dalam *UML* yang secara rinci menggambarkan struktur dan deskripsi dari setiap kelas, termasuk atribut, metode, dan hubungan antar objek. Diagram ini bersifat statis, yang berarti fokusnya lebih pada mendeskripsikan hubungan dan struktur kelas daripada menjelaskan peristiwa atau interaksi yang terjadi jika kelas-kelas tersebut berhubungan satu sama lain. Diagram ini memberikan pemahaman tentang bagaimana kelas-kelas disusun dan bagaimana mereka saling terkait dalam sistem, tanpa menunjukkan dinamika interaksi di antara mereka. Dengan demikian, diagram kelas memberikan pandangan yang komprehensif terhadap komponen-komponen statis dalam sistem [34].

Tabel 2. 13 Komponen *Class Diagram* [33]

Simbol	Nama	Keterangan
	Generalization	Hubungan antar objek anak berbagi perilaku dan struktur data.
	*Nary Association*	Upaya untuk asosiasi apabila lebih dari 2 objek.
	*Class*	Himpunan objek-objek yang berbagi atribut dan operasi yang sama
	*Realization*	Operasi yang dilakukan oleh suatu objek
	*Dependency*	Relasi perubahan yang terjadi pada suatu elemen yang mandiri akan berpengaruh dengan elemen yang bergantung pada elemen yang tidak mandiri
	Association	Menghubungkan antar objek