BAB II
LANDASAN TEORI

2.1. Tinjauan Pustaka

Penelitian terdahulu yang digunakan dalam penelitian ini adalah :

Penelitian yang dilakukan Rahmah Wati (2022) tentang Perancangan E-Katalog Berbasis Web Pada RR Collection Sampit Sebagai Media Branding Menggunakan Aplikasi Figma. E-Katalog sebagai media sarana promosi penjualan produk untuk RR Collection. Dengan adanya E-Katalog akan dapat memudahkan RR Collection mengelola proses transaksi penjualan. Dengan mempunyai E-Katalog, RR Collection akan terlihat lebih profesional dan terpercaya, disamping itu keamanan dalam mengelola bisnis juga akan lebih terjamin daripada membuka dan membangun etalase produk pada sebuah marketplace gratis yang belum tentu akan bertahan lama waktu aktif atau tidaknya marketplace tersebut.

Penelitian yang dilakukan Wahyu Sindu Prasetya Wahyu (2022) tentang Perancangan E-Katalog Usaha Mikro Kecil Menengah Menggunakan *Framework Bootstrap.* Tujuan dari penelitian ini adalah membangun sebuah e-katalog untuk menambah media promosi dan penjualan agar dapat menjangkau pasar yang lebih luas. Pengembangan e-katalog menggunakan *framework bootstrap*, metode penelitiannya DSRM (*Design Science Research Methodology*) dan metode pengembangan perangkat lunaknya RAD (*Rapid Application Development*). Website e-katalog yang dihasilkan dapat digunakan sebagai media promosi dan pengelolaan produk dengan menampilkan informasi yang lebih detil dan lengkap.

Penelitian yang dilakukan Virda Farkhatul Fuadiah,Trihastuti Yuniati,Cepi Ramdani (2022) tentang sistem informasi pengadaan logistik bencana berbasis web. Tujuan untuk menyajikan rancang bangun sistem e-katalog berbasis website pada CV. Kreasindo Karya Media untuk membantu penyebaran informasi produk agar cepat dan tepat sasaran. Metode pengembangan sistem menggunakan rapid application development dengan melibatkan pengguna sistem dari tahap awal hingga akhir dan waktu pengembangan lebih cepat. Tahapan pengembangan sistem terdiri dari pemodelan bisnis, pemodelan data, pemodelan proses, pembuatan aplikasi, pengujian dan turnover. Sistem didesain menggunakan Unified Modelling Language dan dikembangkan menggunakan bahasa pemrogramman PHP dan database MySQL. Validasi fungsi sistem dilakukan dengan menggunakan Black Box testing. Hasil uji fungsional sistem menunjukan fitur pada sistem yang telah dibangun telah memenuhi kebutuhan dalam penyebaran informasi produk dan 100% berjalan secara valid.

2.2. Landasan Teori

2.2.1. Sistem

Menurut Romney dan Steinbart (2019:3): Sistem adalah rangkaian dari dua atau lebih komponen-komponen yang saling berhubungan, yang berinteraksi untuk mencapai suatu tujuan. Sebagian besar sistem terdiri dari subsistem yang lebih kecil yang mendukung sistem yang lebih besar.

Menurut Mulyadi (2019:5), Sistem adalah “suatu jaringan prosedur yang dibuat menurut pola yang terpadu untuk melaksanakan kegiatan pokok perusahaan”.

Berdasarkan pengertian diatas dapat disimpulkan sistem adalah kumpulan dari komponen-komponen yang saling berkaitan satu dengan yang lain untuk mencapai tujuan dalam melaksanakan suatu kegiatan pokok perusahaan.

2.2.2. Infromasi

Menurut Krismaji (2019:14), Informasi adalah "data yang telah diorganisasi dan telah memiliki kegunaan dan manfaat".

Menurut Romney dan Steinbart (2019:4) Informasi adalah data yang telah dikelola dan diproses untuk memberikan arti dan memperbaiki proses pengambilan keputusan. Sebagaimana perannya, pengguna membuat keputusan yang lebih baik sebagai kuantitas dan kualitas dari peningkatan informasi.

Berdasarkan pengertian diatas dapat disimpulkan bahwa pengertian informasi adalah data yang diolah agar bermanfaat dalam pengambilan keputusan bagi penggunanya.

2.2.3. Penjualan

Menurut Hall (2011) Penjualan serangkaian aktivitas bisnis yang mengubah barang atau jasa untuk pelanggan menjadi kas.

Menurut Swastha (2014) penjualan adalah suatu proses pertukaran barang atau jasa antara penjual dan pembeli.

Dari definisi para ahli diatas dapat disimpulkan bahwa penjualan adalah suatu kegiatan bertemunya seorang pembeli dan penjual yang melakukan transaksi, saling mempengaruhi dan mempertimbangkan pertukaran antara barang atau jasa dengan uang.

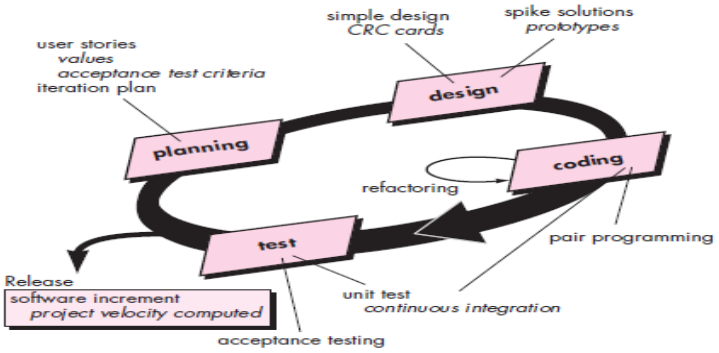
2.3. Metode Pengembangan Sistem

Agile software engineering merupakan salah satu model proses iteratif yang memberikan suatu alternatif yang layak dari berbagai macam metode konvensional untuk membangun berbagai jenis perangkat lunak dan berbagai macam tipe proyek pengembangan perangkat lunak (Pressman, 2010). *Agile software development* adalah Metode dari beberapa kumpulan prinsip untuk pengembangan *software* di mana persyaratan dan solusi melalui upaya kolaboratif dari antar tim fungsional dan klien sebagai pendukung perencanaan adaptif, perkembangan evolusi, awal pengiriman, dan perbaikan terus-menerus, dan itu mendorong respon yang cepat dan fleksibel untuk dirubah. Prinsip-prinsip ini mendukung definisi dan evolusi dari banyak metode pengembangan perangkat lunak.

Agile software development interaksi dan personel lebih penting dari pada proses dan alat, *software* yang berfungsi lebih penting daripada dokumentasi yang lengkap, kolaborasi dengan *client* lebih penting daripada negosiasi kontrak, dan sikap tanggap terhadap perubahan lebih penting daripada mengikuti rencana. Namun demikian, sama seperti model proses yang lain, *agile software development* memiliki kelebihan dan tidak cocok untuk semua jenis proyek, produk, orang dan situasi. *Agile software development* memungkinkan model proses yang toleransi terhadap perubahan kebutuhan sehingga perubahan dapat cepat ditanggapi.

Extreme Programming (XP) adalah metode pengembangan *software* yang cepat, efisien, beresiko rendah, *fleksibel*, terprediksi, *scientific*, dan menyenangkan. *Extreme Programming* (XP) merupakan suatu pendekatan yang paling banyak digunakan untuk pengembangan perangkat lunak cepat (Pressman, 2010). Alasan menggunakan metode XP karena sifat dari aplikasi yang di kembangkan dengan

cepat melalui tahapan-tahapan yang ada meliputi : *Planning* (perencanaan), *Design* (perancangan), *Coding* (Pengkodean), dan *Test* (pengujian).



Gambar 2.1. *Extreme Programming* (Pressman, 2010)

Tahapan dalam metode pengembangan sistem *Extreme Programming* yaitu :

1. ***Planning***

   Pada tahap perencanaan ini dimulai dari pengumpulan kebutuhan yang membantu tim teknikal untuk memahami konteks bisnis dari sebuah aplikasi. Selain itu pada tahap ini juga mendefinisikan *output* yang akan dihasilkan, fitur yang dimiliki oleh aplikasi dan fungsi dari aplikasi yang dikembangkan.

2. ***Design***

   Metode ini menekankan desain aplikasi yang sederhana, untuk mendesain aplikasi dapat menggunakan *Class-Responsibility-Collaborator* (CRC) *cards* yang mengidentifikasi dan mengatur *class* pada *object-oriented.*

3. ***Coding***

   Konsep utama dari tahapan pengkodean pada *extreme programming* adalah *pair programming*, melibatkan lebih dari satu orang untuk menyusun kode.

4. ***Test***

   Pada tahapan ini lebih fokus pada pengujian fitur dan fungsionalitas dari aplikasi.

2.4. UML

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. *UML* merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. *UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

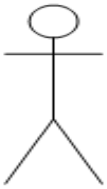
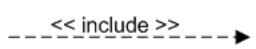
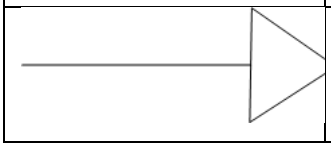
Unified Modeling Language (UML) adalah satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek(Sari and Utami, 2021).

2.4.1. *Use Case Diagram*

Menurut Sari (2021), *use case diagram* adalah diagram yang menggambarkan interaksi antara sistem, eksternal sistem dan *user*. Dengan kata lain, diagram ini menjelaskan sistem tersebut dan bagaimana cara *user* berinteraksi dengan sistem. *Use case diagram* menunjukkan hubungan statis antara *actor* dan *use case* dalam sistem. Mereka menyediakan pandangan awal dari struktur sistem. *Use case* berguna dalam membangun dan mengkomunikasikan pandangan umum sistem. Mereka menyediakan satu titik awal untuk desain, khususnya objek identifikasi dan diagram urutan. Elemen-elemen diagram *use case* adalah *use case*, aktor, kegunaan, dan tanda panah. Aktor diwakili oleh segala sesuatu yang berada diluar sistem, seperti jenis pengguna atau sistem eksternal, yang berinteraksi dengan sistem.

Simbol-simbol yang digunakan dalam *use case diagram* :

Tabel 2.1. Simbol-simbol *Use Case Diagram*

Simbol	Nama Simbol	Keterangan
	*Use-Case*	Gambaran fungsional sistem yang akan di buat, agar pengguna lebih mengerti penggunaan *system*
	*Actor*	*Actor* adalah pengguna sistem. *Actor* tidak terbatas hanya manusia saja, jika sebuah sistem berkomunikasi dengan aplikasi lain dan membutuhkan *input* atau memberikan *output*, maka aplikasi tersebut juga dianggap sebagai *actor*.
	Association	Asosiasi digunakan untuk menghubungkan *actor* dengan *use case*. Asosiasi digambarkan dengan sebuah garis yang menghubungkan antara *actor* dengan *use case.*
	*Extend*	Metode yang hanya berjalan di bawah kondisi tertentu.
	*Include*	Metode yang harus terpenuhi agar sebuah *event* dapat terjadi.
	*Generalization*	Sebuah elemen yang menjadi spesialisasi dari elemen yang lain.

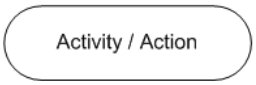
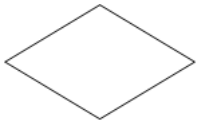
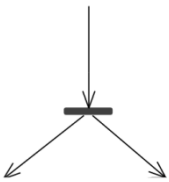
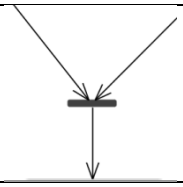
2.4.2. *Activity Diagram*

Sari (2021), *activity diagram* digunakan untuk menggambarkan alur dari proses bisnis atau langkah-langkah *use case* secara berurutan. Diagram ini juga digunakan untuk menggambar *action* (tindakan) yang akan dieksekusikan ketika suatu proses sedang berjalan dan berserta hasil dari proses eksekusi tersebut.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.2. Simbol-simbol *Activity Diagram*

Simbol	Nama Simbol	Keterangan
	*Initial node*	Awal sebuah proses.

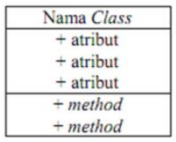
	*Action*	Urutan tindakan membentuk total aktivitas yang ditunjukkan oleh diagram.
	*Flow*	Kebanyakan aliran tidak membutuhkan kata-kata untuk mengidentifikasi mereka kecuali keluar dari keputusan.
	*Decision*	Aliran yang keluar ditandai untuk menunjukkan kondisi.
	*Fork*	bar hitam dengan satu alur masuk dan dua atau lebih alur keluar, aksi di bawah percabangan dapat terjadi dalam urutan apapun atau bahkan secara bersamaan.
	*Join*	bar hitam dengan dua atau lebih alur masuk dan satu alur keluar untuk menyatukan lagi alur aksi yang dipisahkan oleh *fork*.
	*Activity Final*	mewakili akhir proses atau aktivitas.
	*Swimlane*	pembagian *activity diagram* untuk menunjukkan siapa melakukan apa

2.4.3. Class Diagram

Class diagram merupakan gambaran grafis dari struktur objek statis dari sebuah sistem yang menunjukkan kelas objek yang tersusun dari hubungan antara kelas- kelas objek yang lain. *Class diagram* digunakan untuk menggambarkan tampilan desain statis sebuah sistem(Sari and Utami, 2021).

Tabel 2.3. Simbol-simbol *Class Diagram*

Simbol	Nama Simbol	Keterangan

	*Class*	*Class* terdiri atas 3 bagian yaitu bagian atas, bagian tengah, dan bagian bawah. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan *property*/atribut *class*. Bagian akhir mendefinisikan *method-method* dari sebuah *class*.
	Association	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class* dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship*.
	*Composition*	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *composition* terhadap *class* tempat dia bergantung tersebut.
	*Dependency*	Kadang kala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.
	*Aggregation*	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi

2.5. ISO 25010

Model ini merupakan bagian dari *Software product Quality Requirements and Evaluation* (SQuaRE), dimana model ini berkaitan dengan model kualitas perangkat lunak yang merupakan pengembangan dari model sebelumnya. Pada model ini terdapat beberapa sub-karakteristik tambahan dan beberapa sub-

karakteristik yang dipindahkan ke karakteristik lain. Berikut ini merupakan karakteristik atau faktor kualitas internal dan eksternal yang terdapat pada model ISO-25010(Requirements, 2011).

Tabel 2.4. ISO 25010

Faktor	Sub Faktor
Functional Suitability	1. *Appropriateness* 2. *Accuracy* 3. *Functional Suitability Compliance*
Reliability	1. *Availability* 2. *Fault Tolerance* 3. *Recoverability* 4. *Reliability Compliance*
Performance Efficiency	1. *Time-behaviour* 2. *Resource-utilisation* 3. *Performance Efficiency Compliance*
Operability	1. *Appropriateness Recognisability* 2. *Learnability* 3. *Ease of Use* 4. *Helpfulness* 5. *Attractiveness* 6. *Technical Accessibility* 7. *Operability Compliance*
Security	1. *Confidentiality* 2. *Integrity* 3. *Non-repudiation* 4. *Accountability* 5. *Authenticity* 6. *Security Compliance*
Compatibility	1. *Replaceability* 2. *Co-existence* 3. *Interoperability* 4. *Compatibility Compliance*
Maintainability	1. *Modularity* 2. *Reusability*
Functional Suitability	3. *Appropriateness* 4. *Accuracy* 5. *Functional Suitability Compliance*
Reliability	1. *Availability* 2. *Fault Tolerance* 3. *Recoverability* 4. *Reliability Compliance*
Maintainability	1. *Modularity* 2. *Reusability* 3. *Analyzability*

	4. *Changeability* 5. *ModificationStability* 6. *Testability* 7. *Maintainability Complianc*
Transferability	1. *Portability* 2. *Adaptability* 3. *Installability* 4. *Transferability Compliance*

Berikut ini merupakan pengertian dari masing-masing faktor dan sub-faktor yang terdapat pada model ISO-25010, antara lain :

1. *Functional Suitability*

   Functional suitability merupakan tingkat dimana perangkat lunak dapat menyediakan fungsionalitas yang dibutuhkan ketika perangkat lunak digunakan pada kondisi yang spesifik.

2. *Reliability*

   Reliability merupakan tingkatan dimana perangkat lunak dapat bertahan pada tingkatan tertentu ketika digunakan oleh pengguna pada kondisi yang spesifik.

3. *Performance Efficiency*

   Performance efficiency merupakan tingkat dimana perangkat lunak dapat memberikan kinerja yang tepat terhadap sejumlah sumber daya yang digunakan pada kondisi tertentu.

4. *Operability*

   Operability merupakan tingkat dimana perangkat lunak dapat dimengerti, dipelajari, digunakan, dan menarik perhatian pengguna ketika digunakan pada kondisi tertentu.

5. *Security*

Security merupakan perlindungan terhadap perangkat lunak dari berbagai ancaman, akses atau penggunaan dari pengguna yang tidak dikenal.

6. *Compatibility*

   Faktor ini merupakan kemampuan dari dua atau lebih komponen perangkat lunak untuk dapat melakukan pertukaran informasi dan melakukan fungsi yang dibutuhkan ketika digunakan pada hardwareatau lingkungan perangkat lunak yang sama.

7. *Maintainability*

   Maintainability merupakan tingkat dimana sebuah perangkat lunak dapat dimodifikasi. Modifikasi ini termasuk perbaikan, perubahan atau penyesuaian perangkat lunak untuk dapat berubah pada lingkungan, kebutuhan, dan fungsionalitas yang spesifik.

8. *Transferability*

   Transferability merupakan tingkat dimana perangkat lunak dapat berpindah dari lingkungan yang satu ke lingkungan yang lain.

2.6. Skala *Likert*

Skala *Likert* merupakan skala yang didesain untuk menilai sejauh mana responden setuju atau tidak dengan pernyataan pada skala dengan susunan sebagai berikut ini :

Tabel 2.5. Skala *Likert*

Jawaban	Skor
Sangat Setuju	5
Setuju	4
Ragu-Ragu	3

Tidak Setuju	2
Sangat Tidak Setuju	1

Dengan skala *Likert*, maka variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan titik tolak untuk menyusun instrument-instrumen berupa pernyataan.

Pada tahap pengolahan data dalam penelitian ini menggunakan jenis analisis deskriptif yang merupakan jenis penelitian yang menggambarkan fakta-fakta yang ada untuk selanjutnya diolah menjadi data. Data tersebut kemudian dianalisis untuk memperoleh suatu kesimpulan. Analisis statistik deskriptif digunakan untuk menggambarkan bagaimana tingkat kualitas *prototype* yang dibuat.

Langkah-langkah yang dilakukan dalam analisis statistik deskriptif tersebut adalah sebagai berikut (Narimawati, 2007):

1. Setiap indikator yang dinilai oleh responden, diklasifikasikan dalam lima alternatif jawaban dengan menggunakan skala ordinal yang menggambarkan peringkat jawaban.
2. Dihitung total skor setiap variabel/subvariabel = jumlah skor dari seluruh indikator variabel untuk semua responden.
3. Dihitung skor setiap variabel/subvariabel = rata-rata dari total skor.
4. Untuk mendeskripsikan jawaban responden, juga digunakan statistik deskriptif seperti distribusi frekuensi dan tampilan dalam bentuk tabel atau grafik.
5. Untuk menjawab deskripsi tentang variabel penelitian ini, digunakan rentang kriteria penelitian sebagai berikut :

$$SkorTotal = \frac{skorAktual}{skorIdeal} x\ 100\ \%$$

Skor aktual adalah jawaban seluruh responden atas kuesioner yang telah diajukan. Skor ideal adalah skor atau bobot tertinggi atau semua responden diasumsikan memilih jawaban dengan skor tertinggi. Penjelasan bobot nilai skor aktual dapat dilihat pada tabel berikut :

Tabel 2.6. Kriteria Presentase Tanggapan Responden

% Jumlah Skor	Kriteria
20,00 % - 36,00 %	Tidak Baik
36,01 % - 52,00 %	Kurang Baik
52,01 % - 68,00 %	Cukup
68,01 % - 84,00 %	Baik
84,01 % - 100 %	Sangat Baik