BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Penelitian sejenis yang dilakukan oleh M. Amdi Rizal, dkk(2022) berjudul "Aplikasi *Inventory* Persediaan Barang Berbasis *Web* Menggunakan Metode *Extreme Programming* (Studi Kasus : Esha 2 Cell)" dengan model pengembangan sistem XP (*Extreme Programming*) dengan Bahasa pemrograman PHP, DBMS MySQL yang memudahkan admin dalam proses pengelolaan barang masuk dan barang keluar, mempermudah dalam pencarian data barang, serta mempermudah admin dan *owner* dalam proses laporan persediaan dan transaksi penjualan[6].

Penelitian sejenis yang dilakukan oleh Jonathan Setiawan, dkk(2022) berjudul "Aplikasi *Inventory* Barang Berbasis *Web* Pada PT Tetap Panah Mas" dengan model pengembangan sistem *Waterfall*(air terjun) menggunakan Bahasa pemrograman PHP, DBMS MySQL yang memudahkan perusahaan dalam pemantauan dan pencatatan stok barang, barang masuk, dan barang keluar sehingga kinerja karyawan dapat lebih maksimal[7].

Penelitian sejenis yang dilakukan oleh Stephani Calista, dkk(2023) berjudul "Perancangan Sistem Informasi *Inventory* Barang Berbasis *Web* pada Toko Laris *Furniture* Jambi" dengan model pengembangan sistem UML (Unified Modelling Language), dengan Bahasa pemrograman PHP, DBMS

MySQL yang memudahkan proses bisnis secara lebih terkomputerisasi dan terintegrasi dengan menggunakan Bahasa pemrograman PHP dan DBMS MySQL[8].

Penelitian sejenis yang dilakukan oleh Hani Handayani, dkk(2023) berjudul "Perancangan Sistem Informasi *Inventory* Barang Berbasis *Web* Menggunakan Metode *Agile Software Development*" dengan model pengembangan *Agile Software Development* yang memudahkan pengelolaan barang masuk dan keluar di Toko Azura Pekanbaru dan dapat sesuai dengan kebutuhan pengguna sistem tersebut digunakan[9].

Penelitian sejenis yang dilakukan oleh Maulana Ardiansyah (2024) berjudul "Perancangan Aplikasi *Inventory* Barang Berbasis *Web* Dengan Metode *Economic Order Quantity* (Studi Kasus : Elfiza Cell)" dengan model pengembangan sistem XP (*Extrem Programming*) yang memudahkan *user* dalam mengelola data-data toko seperti : data barang masuk, data barang keluar, permintaan barang, *return* barang, data *supplier* data laporan stok barang, serta bisa menentukan jumlah order barang yang efisien[10].

2.2 Landasan Teori

2.2.1 Persediaan (*Inventory)*

Persediaan barang yang juga dikenal sebagai stok barang atau inventaris dengan kata lain *inventory*, mengacu pada jumlah barang yang dimiliki oleh sebuah perusahaan atau entitas bisnis pada suatu waktu tertentu. Persediaan barang mencakup semua barang yang

masih berada dalam proses produksi, barang yang telah selesai diproduksi tetapi belum terjual, serta bahan baku yang akan digunakan dalam proses produksi di masa mendatang.

Menurut Resista Vikaliana et al., (2020) persediaan merupakan sebagai suatu aktiva yang meliputi barang atau produk milik perusahaan yakni untuk dijual dalam suatu periode usaha yang normal, atau persediaan barang atau produk masih dalam pengerjaan/proses produksi, atau hanya persediaan barang baku yang menunggu penggunaanya dalam suatu proses produksi.

Menurut Mufida et al., (2019) *inventory* atau biasa juga disebut dengan persediaan adalah simpanan barang atau produk mentah, material atau barang jadi yang disimpan untuk digunakan dalam masa mendatang atau dalam kurun waktu tertentu.

Sistem pengendalian yang baik menciptakan hasil yang baik juga bagi pelaku usaha itu sendiri, sehingga kontrol persediaan terjaga dengan baik pula.

2.2.2 Metode Pengembangan Sistem *Waterfall*

Metode *Waterfall* pertama kali diperkenalkan oleh Winston W. Royce pada tahun 1970 dalam makalahnya "*Managing the Development of Large Software Systems*". Metode pengembangan sistem *waterfall* adalah salah satu model yang paling awal dan paling sederhana dalam pengembangan perangkat lunak. Metode ini mengikuti pendekatan yang linier dan berurutan, dimana setiap fase

harus diselesaikan sepenuhnya sebelum fase berikutnya dimulai. Proses ini dimulai dari tingkat sistem dan berkembang melalui tahapan analisis, desain, pengkodean, serta pengujian[11].

Metode *waterfall* terdiri dari beberapa tahapan utama:

1. *Requirement Analysis* (Analisis Kebutuhan)

   Pada tahap ini, kebutuhan dari sistem dikumpulkan dan didokumentasikan. Dokumen kebutuhan ini akan menjadi dasar untuk semua tahapan berikutnya.

2. *System Design* (Desain Sistem)

   Berdasarkan dokumen kebutuhan, desain sistem dikembangkan untuk mengarahkan bagaimana sistem akan dibangun.

3. *Implemetation* (Implementasi)

   Desain sistem diterjemahkan ke dalam kode program.

4. *Integration and Testing* (Integrasi dan Pengujian)

   Komponen yang telah dikembangkan diintegrasikan dan diuji untuk memastikan bahwa sistem bekerja sesuai dengan spesifikasi.

5. *Deployment* (Penerapan)

   Sistem yang sudah diuji diterapkan ke lingkungan produksi.

6. *Maintanance* (Pemeliharaan)

   Sistem yang sudah diterapkan akan dirawat dan diperbaiki bila ada kesalahan atau kebutuhan baru.

2.2.3 Pengujian Sistem

Pengujian perangkat lunak merupakan proses untuk menemukan kesalahan pada setiap item perangkat lunak, mencatat hasilnya, mengevaluasi setiap aspek dari setiap komponen sistem, dan menilai fasilitas-fasilitas yang ada pada perangkat lunak yang akan dikembangkan. Selain itu, menjaga kualitas perangkat lunak yang dibangun agar dapat bertahan, serta untuk mengoptimalkan biaya produksi sehingga aplikasi yang dibuat tidak terbuang sia-sia akibat kegagalan pemasaran atau produksi perangkat lunak[12].

Black Box testing adalah pengujian yang bertujuan untuk mengevaluasi fungsionalitas perangkat lunak dengan memberikan input dan kemudian memeriksa apakah output yang dihasilkan sesuai dengan yang diharapkan atau tidak[13].

White Box testing adalah metode pengujian yang didasarkan pada kode program. Untuk menggunakan metode ini, penguji harus memiliki pengetahuan tentang kode serta mampu menulis kasus uji dengan parameter yang sesuai. Dalam *White Box testing*, terdapat beberapa jenis pengujian yaitu data *flow testing*, *control flow testing*, *basis path testing*, dan *loop testing*[14].

2.2.4 *Website*

Menurut Elgamar(2020) *website* adalah suatu media yang terdiri dari beberapa halaman yang saling berkaitan satu sama lain, dan berfungsi sebagai media untuk menampilkan suatu informasi,

baik berbentuk gambar, video, teks, suara, ataupun gabungan dari semuanya. *Website* bersifat multi plat*form* yang artinya dapat dibuka dari segala perangkat atau *device* yang terhubung dengan jaringan internet [15].

2.2.5 *Database*

Database merupakan salah satu komponen penting dalam teknologi informasi yang mendukung penyimpanan data. Dengan perkembangan teknologi informasi yang semakin maju, penggunaan *database* memungkinkan informasi untuk disimpan secara terkomputerisasi dengan ukuran yang semakin kecil dan akses yang lebih cepat. Hal ini memungkinkan pengambilan keputusan yang lebih efektif dan efisien dalam berbagai bidang kehidupan, karena data dapat diakses dan dianalisis dengan lebih mudah dan cepat [16].

Perangkat lunak dalam kategori Sistem Manajemen Basis Data (DBMS) meliputi beragam program seperti dbase III+, FoxBase, MS-Access, dan Borland-Paradox untuk keperluan dasar, serta Borland Interbase, MySQL, SQLServer, Oracle, In*form*ix, dan Sybase untuk tingkat kompleksitas yang lebih tinggi [17].

2.2.6 MySQL

MySQL adalah sebuah *database* yang sering digunakan untuk mengembangkan aplikasi web dinamis, termasuk dalam kategori Sistem Manajemen Basis Data Relasional (RDBMS). MySQL kompatibel dengan Bahasa pemrograman PHP dan menyediakan

query SQL (*Structured Query Language*) yang sederhana, menggunakan karakter *escape* yang sama dengan PHP [18].

2.2.7 *Xampp*

XAMPP adalah singkatan dari X (tempat sistem operasi apa pun), *Apache, MySQL, PHP, Perl*. *XAMPP* adalah perangkat lunak bebas yang mendukung berbagai sistem operasi dan merupakan kumpulan dari beberapa program. *XAMPP* adalah sebuah tool yang menyediakan paket perangkat lunak secara terintegrasi dalam satu paket, memudahkan pengguna untuk menginstal dan menjalankan *server* web, *database*, dan Bahasa pemrograman seperti *Apache,* MySQL, PHP, dan Perl dalam lingkungan yang terpadu [19].

2.2.8 *Visual Studio Code*

Visual Studio Code adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung Bahasa pemrograman Javascript, Typescript, dan Node. Js, serta Bahasa pemrograman lainnya dengan bantuan *plugin* yang dapat dipasang via *marketplace Visual Studio Code* seperti : C++, C#, Python, Go, Java, PHP, dst [20].

2.2.9 *Bootstrap*

Bootstrap adalah sebuah kerangka kerja untuk CSS yang merupakan produk *open source* yang dikembangkan oleh Mark Otto

dan Jacob Thornton. Pada awalnya, *Bootstrap* diciptakan untuk menetapkan standar tampilan depan (*front end*) bagi semua pengembang di perusahaan mereka. Seiring waktu, *Bootstrap* telah berkembang dari proyek yang berbasis CSS menjadi sebuah *platform* yang menyediakan beragam *plugin* JavaScript dan ikon yang dapat dengan mudah diintegrasikan dalam formulir dan tombol [18].

2.2.10 *Unified Modeling Language (UML*)

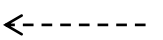
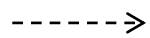
UML (*Unified Modeling Language*) adalah sebuah Bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan artifact (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya (Nugroho, 2010).

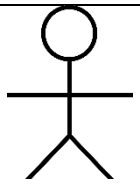
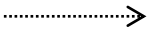
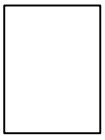
1. *Use Case* Diagram

   Use case diagram adalah alat yang digunakan untuk menggambarkan persyaratan sistem dengan jelas. Komponen utamanya meliputi actor, *use case*, dan subjek sistem. Sistem merupakan entitas utama yang dianalisis dalam diagram tersebut, yang mencakup semua subjek *use case* yang terlibat dalam implementasi sistem. *Use case* diagram menjelaskan bagaimana perilaku yang terjadi sesuai dengan scenario yang

dijelaskan dalam setiap *use case* [21]. Simbol-simbol yang digunakan dalam *Use Case* Diagram yaitu:

Tabel 2.1 *Use Case* Diagram menurut (Destriana, 2021)

No	Simbol	Pengertian	Keterangan
1.		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
2.		Asosiasi/ *Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
3.		Generalisasi/ *generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4.	 <<*extend*>>	Ekstensi/ *Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
5.	 <<*include*>>	*Include/uses*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
6.		*Actor*/Aktor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika

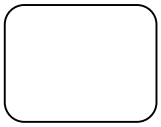
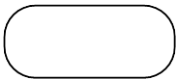
No	Simbol	Pengertian	Keterangan
			berinteraksi dengan *use case*.
7.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
8.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
9.		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10.		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

2. *Activity* Diagram

Activity Diagram adalah representasi visual dari alur kerja yang mencakup aktivitas dan tindakan, serta kemungkinan pilihan, pengulangan, dan konsekuensi. Dalam UML (*Unified Modeling Language*), diagram aktivitas digunakan untuk menggambarkan aktivitas komputer atau alur aktivitas dalam sebuah organisasi. Diagram aktivitas menggambarkan garis besar tentang alur control dalam suatu proses atau sistem. Komponen-komponen dalam diagram aktivitas dari awal hingga akhir [21].

Simbol-simbol yang digunakan dalam *Activity Diagram* yaitu:

Tabel 2.2 *Activity Diagram* menurut (Destriana, 2021)

No	Simbol	Pengertian	Keterangan
1.		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain.
2.		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3.		*Activity Final Mode*	Bagaimana objek dibentuk dan dihancurkan.

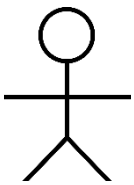
4.		Initial Node	Bagaimana objek dibentuk atau diawali.
5.		Fork Node	Satu aliran yang tahap tertentu berubah menjadi beberapa aliran.
6.		*Join Node*	Banyak aliran yang pada tahap tertentu berubah menjadi beberapa satu aliran

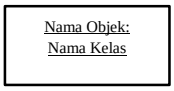
3. *Sequence Diagram*

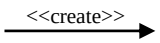
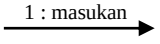
Sequence Diagram menggambarkan interaksi antara objek-objek di dalam dan di sekitar sistem (termasuk pengguna, tampilan, dan sebagainya) dalam bentuk pesan yang diperlihatkan seiring berjalannya waktu. Diagram ini mencakup dimensi vertikal (waktu) dan horizontal (objek-objek yang saling berhubungan)[22].

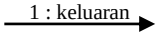
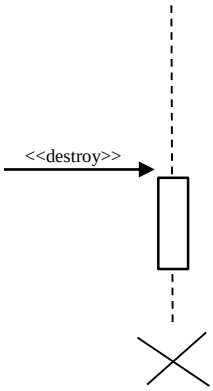
Berikut ini adalah simbol-simbol *sequence diagram*.

Tabel 2.3 *Sequence Diagram* menurut (Irfan, 2023)

No	Simbol	Pengertian	Keterangan
1.		Aktor	Proses, orang atau sistem yang berbeda yang terhubung dengan sistem data yang sedang dikembangkan berada di laur sistem

No	Simbol	Pengertian	Keterangan
			informasi yang sedang dibuat, sehingga kator tidak selalu berupa manusia meskipun simbol aktor adalah gambar seseorang. Biasanya, kata benda digunakan di depan frasa nama aktor untuk mengungkapkannya.
2.		*Lifeline*	Mewakili siklus hidup suatu objek.
3.		Objek	Mengidentifikasi objek yang berinteraksi melalui pesan.
4.		Waktu Aktif	Mengatur bahwa objek tersebut aktif dan interaktif, serta semua hal yang terkait dengan jangka waktu aktivitasnya merupakan langkah yang

No	Simbol	Pengertian	Keterangan
			diambil di dalamnya.
5.		Pesan tipe *Create*	Menunjukan bahwa sebuah objek menciptakan objek lain, dengan panah yang mengarah ke objek yang dibuat. Panah ini menunjuk ke objek yang memiliki operasi/metode tersebut. Operasi/metode yang dipanggil harus ada dalam diagram kelas sesuai dengan kelas dari objek yang berinteraksi.
6.		Pesan tipe *Send*	Merupakan objek yang mengirimkan *input* data, atau informasi ke objek lain, dengan panah yang mengarah ke objek penerima.

No	Simbol	Pengertian	Keterangan
7.		Pesan tipe *return*	Mengindikasikan bahwa suatu objek kembali ke objek tertentu setelah menjalankan operasi atau metode tertentu, dengan panah yang mengarah ke objek yang menerima pengembalian.
		Pesan tipe *destroy*	Menetapkan bahwa sebuah objek mengakhiri keberadaan objek lain, dengan panah yang mengarah ke objek yang diakhiri. Idealnya, jika ada *create*, harus ada juga *destroy*.

4. *Class Diagram*

Class Diagram adalah cara untuk memvisualisasikan struktur sistem dengan mendefinisikan kelas-kelas yang menyusunnya[23].