BAB II

TINJAUAN PUSTAKA

2.1 Penilitian Terkait

Dalam pengembangan sistem informasi berbasis *web*, khususnya pada sektor penjualan *furniture*, telah banyak dilakukan penelitian yang bertujuan untuk meningkatkan efektivitas promosi, memperluas jangkauan pasar, serta mempermudah proses transaksi secara daring. Hal ini menunjukkan adanya kebutuhan yang nyata terhadap digitalisasi proses penjualan di bidang mebel yang sebelumnya masih mengandalkan metode konvensional. Beberapa penelitian sebelumnya dapat dijadikan referensi dan pembanding dalam pengembangan sistem informasi pemesanan dan pengiriman barang berbasis *web* pada toko mebel azwaely putra yang menjadi fokus dalam penelitian ini.

Penelitian oleh Abdur Rochman et al. (2021) telah merancang sebuah aplikasi penjualan berbasis *web* untuk mengatasi persaingan dalam bisnis *furniture*. Aplikasi tersebut dikembangkan untuk mengatasi kurangnya promosi dan berhasil meningkatkan penjualan produk serta memudahkan transaksi pembelian secara online [3].

Penelitian lainnya dilakukan oleh Mohammad Jon Tasrif (2019) dengan judul Pemesanan *Furniture* Berbasis *Web* dengan Menggunakan PHP dan MySQL pada Toko Triana *Furniture*. Penelitian ini membahas perancangan sistem pemesanan *furniture* berbasis *web* untuk mempermudah proses penjangkauan konsumen dan pemesanan barang *furniture* yang sebelumnya

dilakukan secara manual [4].

Selain itu, Muchammad Tirozul Achyar dan Fandy Indra Pratama (2021) dalam jurnalnya berjudul Sistem *Informasi E-Commerce Furniture* Berbasis *Web* pada Toko Mebel Ubaidillah Kamal Jepara, menghasilkan sistem informasi *e-commerce furniture* berbasis *web* dengan metode *Waterfall* untuk memudahkan pembeli dalam melakukan pemesanan [5].

Meskipun ketiga penelitian tersebut telah mengembangkan sistem pemesanan berbasis *web*, namun belum banyak yang membahas integrasi fitur pengiriman secara otomatis dan sistem pembayaran yang lebih terstruktur, sehingga menjadi celah yang akan dikembangkan dalam penelitian ini.

2.2 Landasan Teori

Landasan teori merupakan konsep, teori, prinsip, dan pendapat yang mendukung proses perancangan sistem serta sumber daya yang digunakan dalam pengembangan sistem yang dibuat. Berikut dijelaskan teori-teori yang menjadi dasar atau pedoman dalam Rancang Bangun Sistem Pemesanan dan Pengiriman Barang pada Toko Azwaely Putra untuk meningkatkan efektivitas pengelolaan transaksi dan distribusi produk berbasis *website*.

2.2.1 Laravel

Laravel adalah sebuah *framework web* berbasis PHP yang *open-source* dan tidak berbayar, diciptakan oleh Taylor Otwell dan diperuntukkan untuk pengembangan aplikasi *web* yang menggunakan

pola MVC. Struktrur pola MVC pada laravel sedikit berbeda pada struktur pola MVC pada umumnya[6].

2.2.2 MySQL

MySQL adalah program *database server* berbasis *multi-user* yang menggunakan perintah standar SQL (*Structured Query Language*) [7]. Fungsinya yaitu mengatur dan menyimpan data dalam tabel yang saling terhubung, sehingga memudahkan pengguna dalam menyimpan, mengakses, serta mengelola data secara efisien.

2.2.3 *Payment Gateway* Midtrans

Midtrans merupakan *payment gatewa*y yang mempermudah proses pembayaran online serta menyediakan mode *sandbox* untuk pengujian transaksi. Fitur ini memungkinkan pengembang mensimulasikan berbagai metode pembayaran guna memastikan sistem berjalan aman sebelum digunakan secara resmi [8].

2.2.4 *Database*

Database adalah kumpulan data terkait yang disimpan bersama dengan redundansi terkontrol untuk melayani satu atau lebih aplikasi secara optimal. Data disimpan sedemikian rupa sehingga terlepas dari program yang digunakan orang untuk mengakses data. Akses untuk menambahkan data baru, mengedit, dan mengambil data[9].

2.2.5 UML

Unified Modeling Language (UML) adalah sebuah bahasa pemodelan perangkat lunak yang telah distandardisasi sebagai

media penulisan cetak biru *(blueprints)* perangkat lunak *(Pressman)*[10].

Di dalam UML terdapat 3 teknik perancangan sistem, model-model tersebebut yaitu : *Use Case Diagram, Sequence Diagram,* dan *Class Diagram.*

2.2.6 *Use Case Diagram*

Use case Diagaram adalah pemodelan yang mempunyai kemampuan untuk menggambarkan interaksi diantara aktor dan sistem[11]. Simbol *usecase diagram* bisa dilihat pada tabel 2.1.

Tabel 2.1 *Usecase Diagram*

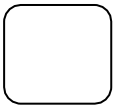
No	Gambar	Nama	Keterangan
1.		*Actor*	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
3.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada diatasnya objek induk (*ancertor*).
4.		*Include*	Menspesifikasikan *bahwa usecase* sumber secara eksplisit.
5.		*Extend*	Menspesifikasikan bahwa *usecase* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

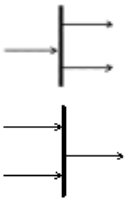
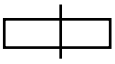
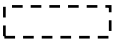
No	Gambar	Nama	Keterangan
6.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7.		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8.		*Use Case*	Deskripsi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
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.

2.2.7 *Activity Diagram*

Activity Diagram digunakan untuk menggambarkan alur aktivitas dalam sistem yang berjalan [12]. Simbol-simbol yang digunakan dapat dilihat pada Tabel 2.2

Tabel 2.2 *Activity Diagram*

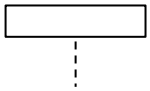
No	Gambar	Nama	Keterangan
1.		*Activity*	Memperlihatkan bagaimana masing - masing kelas antarmuka saling berinteraksi
2.		*Action*	State dari sistem yang mencerminkan eksekusi suatu aksi.
3.		*Initial Node*	Bagaimana objek dibentuk atau diawali.

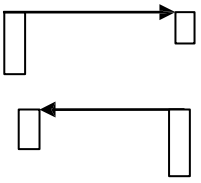
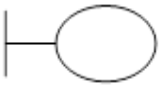
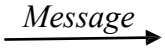
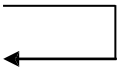
No	Gambar	Nama	Keterangan
4.		*Final Node*	Bagaimana objek dibentuk dan dihancurkan.
5.		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.
6.		*Decision*	Pilihan untuk mengambil keputusan
7.		*Fork/Join*	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
8.		*Rake*	Menunjukkan adanya dekomposisi
9.		*Time*	Tanda waktu
10.		*Send*	Tanda pengiriman

2.2.8 *Squence Diagram*

Sequence diagram adalah gambaran interaksi antar objek, yang digunakan untuk menunjukkan komunikasi atau pesan yang ada di antara objek tersebut[13]. komponen simbol *Sequence* bisa dilihat pada tabel 2.3.

Tabel 2.3 *Squence Diagram*

No	Gambar	Nama	Keterangan
1.		*LifeLine*	Objek *entity*, antar muka yang saling berinteraksi.

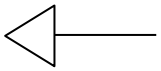
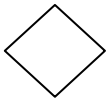
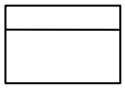
2.		*Message*	Spesifikasi dari komunikasi antar objek yang memuat informasi – informasi tentang aktifitas yang terjadi.
3		*Actor*	Menggambarkan orang yang sedang berinteraksi dengan sistem
4		*Boundary Class*	Menggambarkan penggambaran dari *form*
5		*Entity Class*	Mengambarkan hubungan kegiatan yang akan dilakukan
6.		*Control Class*	Menggambarkan penghubung antara Boundary dengan tabel
7		*Activation*	Sebagai sebuah objek yang akan melakukan sebuah aksi
8		*Message*	Mengindikasikan komunikasi antara objek dengan objek
9		*Self Message*	Mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri

2.2.9 *Class Diagram*

Class Diagram merupakan struktur sistem yang menggambarkan rancangan dari sistem perangkat lunak yang akan dibuat[14]. Diagram ini memodelkan *class, package*, dan objek, serta hubungan antar elemen-elemen tersebut, seperti *containment* (kepemilikan), *inheritance* (pewarisan), dan *associations* (asosiasi).

Untuk simbol dalam *class diagram* bisa dilihat pada tabel 2.4.

Tabel 2.4 *Class Diagram*

No	Gambar	Nama	Keterangan
1.		*Generalization*	Hubungan dimana objek anak(*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*)
2.		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		*Class*	Himpunan dari objek – objek yang berbagi atribut serta operasi yang sama.
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
5.		*Dependency*	Operasi yang benar - benar dilakukan oleh suatu objek.
6.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri(*independent*) akanmempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7.		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.