

BAB II

TINJAUAN PUSTAKA

2.1. Penelitian Terkait

Penelitian sebelumnya tentang peternakan telah berfokus pada pengembangan sistem *monitoring* lingkungan berbasis Internet of Things (*IoT*). Studi oleh Made Sukafona, Kadek Dwi Gandika Supartha menunjukkan bahwa sensor DHT11 dapat mengukur suhu dan kelembapan dengan tepat untuk penggunaan pemantauan kandang unggas. Menggunakan mikrokontroler *WiFi* untuk mengirimkan data ke server, sistem tersebut belum memantau kualitas udara, terutama gas amonia, yang berbahaya bagi kesehatan unggas[3].

Dalam penelitian tambahan yang dilakukan oleh Marco Alfiano Laoh, Andi Ikhtiar Bakti, sensor MQ-135 digunakan untuk memantau kualitas udara dan menemukan gas amonia di lingkungan peternakan. Hasil penelitian menunjukkan bahwa peningkatan kadar amonia dapat menyebabkan stres pada unggas, yang berarti ada kebutuhan akan sistem pemantauan otomatis. Meskipun demikian, penelitian tersebut hanya menampilkan data lokal, bukan *dashboard* yang dapat diakses dari jarak jauh melalui web[4].

Studi sebelumnya menunjukkan bahwa sistem pemantauan *IoT* yang mengukur suhu, kelembapan, dan amonia sekaligus menampilkan data secara real time melalui *dashboard* web masih jarang digunakan secara terpadu. Akibatnya, penelitian ini mengembangkan sistem pemantauan lingkungan

kandang puyuh berbasis *IoT* yang memungkinkan peternak memantau kondisi lingkungan kandang mereka dengan lebih baik dan lebih efisien[5].

2.2. Landasan Teori

2.2.1. Visual Studio Code

Teks editor Visual Studio Code dibuat oleh Microsoft. Teks editor ini menawarkan berbagai macam kemampuan, termasuk code completion, syntax highlighting, debugging, dan git control. Menurut Dermawan et al., (2021), teks editor ini kompatibel dengan semua sistem operasi, termasuk Windows, Linux, dan Macintosh. Typescript, HTML, CSS, Node.js, dan JavaScript semuanya didukung oleh Visual Studio Code[6].



Gambar 2. 1 Logo Visual Studio Code

Sumber: <https://tinyurl.com/mre43dvs>

2.2.2. Laravel

Laravel adalah salah satu framework php yang dibuat MIT, laravel berbasis MVC (Model-Controller-View), beberapa kelebihan dari laravel ialah ekspresif karena sintaks laravel dapat mudah diketahui oleh programmer, simple dalam pengelolaan basis data

dengan kode sederhana, accessible bagi programmer dengan mengurangi beberapa tugas dalam membuat *Website* seperti routing, membuat sesi, maupun cache[7].



Gambar 2. 2 *Laravel*

Sumber: <https://tinyurl.com/3yykrbhv>

2.2.3. **Arduino IDE**

Arduino IDE adalah software yang digunakan untuk membuat sketch pemrograman atau dengan kata lain arduino IDE sebagai media untuk pemrograman pada board yang ingin diprogram. Arduino IDE ini berguna untuk mengedit, membuat, meng-upload ke board yang ditentukan, dan meng-coding program tertentu. Arduino IDE dibuat dari bahasa pemrograman JAVA, yang dilengkapi dengan library C/C++(wiring), yang membuat operasi input/output lebih mudah[8].



Gambar 2. 3 *Arduino IDE*

Sumber: <https://tinyurl.com/2e79zv44>

2.2.4. Laragon

Laragon adalah perangkat lunak gratis yang mendukung banyak sistem operasi, berfungsi sebagai server diri sendiri atau *localhost*. Laragon menyediakan banyak *services*, *tools*, dan fitur mulai dari *Apache*, *MySQL*, *PHP Server*, *Memcached*, *Redis*, *Composer*, *Xdebug*, *PhpMyAdmin*, *Cmder* dan *Laravel*[9].



Gambar 2. 4 Laragon

Sumber: <https://tinyurl.com/msbmn79n>

2.2.5. Mysql

MySQL adalah salah satu jenis *database* yang paling umum digunakan untuk membuat aplikasi web yang dinamis. Karena sifatnya yang open source dan kapasitasnya yang besar, *MySQL* menjadi *database* manajemen sistem (DBMS) multiuser[10].



Gambar 2. 5 Mysql

Sumber: <https://tinyurl.com/vw2rs59e>

2.2.6. PHP

PhpMyAdmin adalah aplikasi atau program open source PHP yang memungkinkan mengelola *database MySQL* melalui jaringan lokal atau internet. *PhpMyAdmin* mendukung berbagai operasi *MySQL*, termasuk mengelola *database*, tabel-tabel, bidang, relasi, indeks, pengguna, perijinan, dan banyak lagi[11].

2.2.7. Tailwind Css

Tailwind CSS adalah framework CSS utility-first yang mempercepat pengembangan antarmuka dengan kelas-kelas utilitas siap pakai untuk tata letak, warna, dan tipografi. Framework ini menawarkan fleksibilitas desain penuh, konsistensi visual, serta optimalisasi performa melalui penghapusan kode CSS yang tidak terpakai, menjadikannya populer dalam pengembangan frontend modern.

2.2.8. Hypertext Transfer Protocol Secure (HTTPS)

HTTPS merupakan pengembangan dari *HTTP*, protokol ini menyediakan keamanan kriptografi, dengan membawa pesan *HTTP* melalui Transport Layer Security (TLS). Sebelum dapat berkomunikasi dengan *HTTPS*, langkah awal yang perlu dilakukan yaitu *TLS Handshake*. *TLS Handshake* merupakan proses komunikasi antara client dengan server untuk memperoleh *Session Key* yaitu *key* yang nantinya akan digunakan untuk melakukan enkripsi terhadap data yang akan dikirimkan[12]. Representational State Transfer atau juga dikenal sebagai *RESTful* web *API*, merupakan sebuah arsitektur perangkat lunak yang digunakan pada aplikasi berbasis web. *REST API* menggunakan protokol komunikasi *HTTP/HTTPS*. Dalam *REST API* dikenal istilah client dan server, client merupakan pihak pengguna akhir yang bertujuan untuk mengakses data yang disediakan oleh *API*, sedangkan server merupakan pihak yang menyediakan *API*[12].

2.2.9. UML (Unified Modeling Language)

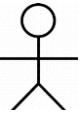
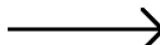
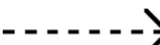
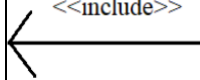
Unified Modeling Language (UML) merupakan bahasa pemodelan perangkat lunak yang telah distandardisasi dan digunakan sebagai sarana untuk menyusun cetak biru (blueprint) dalam pengembangan perangkat lunak (Pressman). UML dapat dimanfaatkan untuk melakukan visualisasi, spesifikasi, konstruksi, serta dokumentasi berbagai komponen atau bagian dari suatu sistem

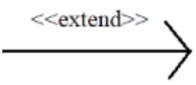
perangkat lunak. Penggunaan UML membantu pengembang dan pemangku kepentingan dalam memahami struktur serta perilaku sistem secara lebih jelas sebelum tahap implementasi dilakukan. Beberapa diagram UML yang umum digunakan antara lain:

1. *Use Case Diagram*

Diagram ini menggambarkan kumpulan *use case* beserta aktor yang terlibat di dalamnya. Diagram tersebut berperan penting dalam mengorganisasi serta memodelkan perilaku sistem yang diharapkan dan dibutuhkan oleh pengguna atau pihak manusia yang berinteraksi dengan sistem.

Tabel 2. 1 *UseCase Diagram*

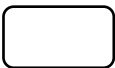
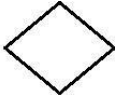
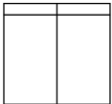
Simbol	Nama	Keterangan
	<i>Actor</i>	Mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan <i>use case</i> .
	<i>Use case</i>	Abstraksi dan interaksi antara sistem dan aktor.
	<i>Association</i>	Abstraksi dari penghubung antara aktor dan <i>use case</i> .
	Generalisasi	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan <i>use case</i> .
	<i>include</i>	Menunjukkan bahwa suatu <i>use case</i> seluruhnya merupakan fungsionalitas dari <i>use case</i> lainnya.

	<i>extend</i>	Menunjukkan bahwa suatu <i>use case</i> merupakan tambahan fungsional dari <i>use case</i> lainnya jika suatu kondisi terpenuhi.
---	---------------	--

2. Activity Diagram

Activity Diagram merupakan salah satu diagram dalam *UML* yang digunakan untuk menggambarkan perilaku dinamis suatu sistem atau bagian dari sistem. Diagram ini menunjukkan alur kontrol aktivitas melalui rangkaian aksi atau kegiatan yang dilakukan oleh sistem.

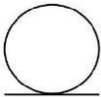
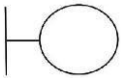
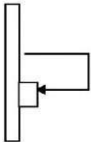
Tabel 2. 2 *Activity Diagram*

Simbol	Nama	Keterangan
	<i>Initial Node</i>	Sebuah diagram aktivitas memiliki sebuah status awal.
	<i>Action</i>	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	<i>Activity</i>	Memperlihatkan bagaimana masing masing kelas antarmuka saling berinteraksi satu sama lain.
	<i>Decision</i>	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Penggabungan atau <i>Join</i>	Penggabungan dimana yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.
	<i>Activity Final Node</i>	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
	<i>Vertical Swimlane</i>	membagi aktivitas berdasarkan pihak yang bertanggung jawab, sehingga alur proses lebih jelas, terstruktur, dan mudah dipahami.

3. Sequence Diagram

Sequence Diagram digunakan untuk menggambarkan interaksi antar objek di dalam maupun di sekitar sistem, termasuk pengguna dan komponen antarmuka, dalam bentuk pertukaran pesan (*message*) yang tersusun berdasarkan urutan waktu. Diagram ini umumnya dimanfaatkan untuk menjelaskan skenario atau tahapan proses yang terjadi sebagai respons terhadap suatu peristiwa (*event*) tertentu hingga menghasilkan keluaran yang diharapkan.

Tabel 2. 3 *Sequence Diagram*

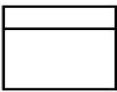
Simbol	Nama	Keterangan
	<i>Entity Class</i>	Gambaran sistem sebagai landasan dalam menyusun basis data.
	<i>Boundary Class</i>	Menangani komunikasi antar lingkungan sistem.
	<i>Control Class</i>	Bertanggung jawab terhadap kelas-kelas terhadap objek yang berisi logika.
	<i>Recursive</i>	Pesan untuk dirinya.
	<i>Activation</i>	Mewakili proses durasi aktivasi sebuah operasi.

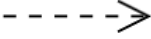
Simbol	Nama	Keterangan
	<i>Life Line</i>	Komponen yang digambarkan garis putus terhubung dengan objek.

4. *Class Diagram*

Class Diagram merupakan diagram yang digunakan untuk menggambarkan struktur sistem, termasuk deskripsi kelas (*class*), paket (*package*), serta objek-objek beserta hubungan yang terjadi di antaranya, seperti pewarisan (*inheritance*), asosiasi, dan hubungan lainnya.

Tabel 2. 4 *Class Diagram*

Simbol	Nama	Keterangan
	<i>Generalization</i>	Hubungan dimana objek anak berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk.
	<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih 2 objek.
	<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
	<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
	<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan

Simbol	Nama	Keterangan
		mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.