

BAB II

TINJAUAN PUSTAKA

2.1 Teori Terkait

Penelitian mengenai modernisasi sektor pertanian melalui teknologi *Internet of Things* (IoT) telah berkembang pesat dalam beberapa tahun terakhir, memberikan landasan kuat bagi perancangan sistem pemantauan tanaman yang presisi. Salah satu penelitian mendasar dilakukan oleh Abdiansyah, Rahman, dan Maulidda (2024) dalam Jurnal Teliska Polsri, yang merancang sistem monitoring kelembapan tanah untuk menjawab tantangan efisiensi penggunaan air. Melalui integrasi perangkat keras berbasis IoT, penelitian tersebut membuktikan bahwa mekanisme kontrol otomatis pada pompa dapat bekerja secara *real-time* untuk menjaga kestabilan parameter lingkungan[2]. Hal ini sejalan dengan temuan Adib Nugroho dan Kurniawan Irianto (2025) dalam *Smart Comp Journal*, yang mengembangkan sensor *soil* berbasis IoT untuk mengukur kelembapan tanah secara akurat. Mereka menekankan bahwa kelembapan adalah parameter vital bagi pertumbuhan tanaman, di mana penggunaan aplikasi berbasis internet memungkinkan petani melakukan pemantauan jarak jauh tanpa harus berada di lokasi lahan secara fisik[3].

Khusus pada komoditas melon, tantangan budidaya yang dihadapi adalah kerentanan terhadap kondisi air yang tidak terukur. Fathurrohman et al. (2024) dalam Jurnal JATI (Jurnal Mahasiswa Teknik Informatika)

membedah masalah ketidakefektifan penyiraman *manual* pada tanaman melon yang seringkali menurunkan kualitas buah. Dengan menggunakan metode sensor *Soil Moisture* sebagai pemicu (*trigger*) otomatisasi, penelitian tersebut berhasil membuktikan bahwa sistem IoT mampu menjaga kelembapan tanah pada fase pertumbuhan melon secara konsisten[4].

Penelitian oleh Ayasha Ninda Maharani (2021) yang berjudul “Rancang Bangun Aplikasi Pengontrol Sistem Penyiram Tanaman Berbasis *Arduino* dan *Android* ” memberikan referensi teknis mengenai integrasi perangkat keras ESP32. Dalam studinya, penggunaan platform *Firebase* sebagai *database* memungkinkan sinkronisasi antara sensor kelembapan dan relay dengan tingkat keberhasilan sistem mencapai 94.4%. Hasil penelitian ini menjadi acuan utama dalam menentukan klasifikasi ambang batas kelembapan tanah (kering, lembap, dan basah) yang diterapkan pada logika sistem[6]. Selain itu, Novita Kurnia Ningrum et al. (2023) dalam prosiding SENATIB menunjukkan bahwa penerapan analisis cerdas pada data sensor dapat meningkatkan akurasi pengambilan keputusan penyiraman secara otomatis dibandingkan dengan metode ambang batas statis[7].

Sebagai upaya pengembangan dari sistem yang sudah ada, Alfian Fathur Rohman (2024) melalui studinya di Universitas Muhammadiyah Surakarta berhasil mengintegrasikan aplikasi *Android* dengan *Node.js* untuk kendali jarak jauh. Penelitian tersebut membuktikan efektivitas aplikasi *mobile* sebagai antarmuka utama dalam ekosistem pertanian digital. Berdasarkan riwayat penelitian tersebut, pengembangan sistem pada Tugas

Akhir ini diarahkan pada penggunaan bahasa pemrograman Kotlin dan *backend* Supabase.[8]

2.2 Landasan Teori

2.2.1 Aplikasi

Aplikasi merupakan perangkat lunak yang dirancang untuk membantu pengguna dalam menjalankan tugas atau pekerjaan tertentu melalui sistem komputer atau perangkat digital. Dalam perkembangan teknologi informasi *modern*, aplikasi menjadi bagian penting dalam berbagai bidang karena mampu mempermudah proses pengolahan data, penyampaian informasi, serta pengelolaan sistem secara lebih efektif dan efisien. Selain itu, dalam buku *Software Engineering* oleh Ian Sommerville (2020) dijelaskan bahwa perangkat lunak aplikasi (*application software*) merupakan program yang dirancang untuk membantu pengguna menyelesaikan pekerjaan tertentu melalui sistem komputer, seperti pengolahan data, komunikasi, dan pengelolaan informasi.[9]

2.2.2 Android

Android adalah sistem operasi berbasis kernel Linux yang dirancang khusus untuk perangkat bergerak dengan layar sentuh. Sistem ini menyediakan lingkungan yang terbuka bagi pengembang untuk menciptakan aplikasi *mobile* dengan fleksibilitas tinggi. Menurut Zahid, Android sebagai platform terbuka telah mendorong

dominasi pangsa pasar smartphone di Indonesia karena kemudahan pengembangan dan distribusi aplikasinya[10].



Gambar 2. 1 Android

2.2.3 Android Studio

Android Studio merupakan lingkungan pengembangan terpadu atau *Integrated Development Environment* (IDE) resmi yang digunakan untuk merancang serta membangun aplikasi pada *platform Android* dengan dukungan fitur yang komprehensif. Menurut Aditya dkk. (2022), *platform* ini memfasilitasi pengembang dengan fitur emulator dan integrasi sistem yang memungkinkan pembuatan aplikasi berbasis *Android* secara lebih terstruktur dan efisien sesuai dengan kebutuhan pengguna saat ini [11].



Gambar 2. 2 Android Studio

2.2.4 Kotlin

Kotlin adalah bahasa pemrograman modern yang dikembangkan oleh JetBrains dan diresmikan Google sebagai bahasa utama pengembangan Android sejak 2017. Menurut Wicaksono & Leong (2021) dalam Proxies Journal, Universitas Katolik Soegijapranata, Kotlin memiliki sintaksis sederhana, mendukung paradigma berorientasi objek dan fungsional, serta lebih efisien dibanding *Java* dalam mengurangi bug dan error pada aplikasi Android[12].



Gambar 2. 3 Kotlin

2.2.5 Supabase

Supabase merupakan platform *Backend-as-a-Service* (BaaS) berbasis kode sumber terbuka (*open-source*) yang menggunakan PostgreSQL sebagai sistem manajemen *database* utamanya. Menurut Fauzi dkk. (2023), platform ini hadir sebagai solusi bagi pengembang yang membutuhkan integrasi database secara *real-time*, layanan autentikasi, dan penyimpanan file yang efisien tanpa harus mengelola infrastruktur server secara mandiri. Selain itu, penggunaan fitur *Row Level Security* (RLS) pada Supabase memastikan bahwa akses data

dapat dikontrol dengan ketat, sehingga meningkatkan aspek keamanan pada aplikasi *mobile* maupun web yang dibangun[13].



Gambar 2. 4 Supabase

2.2.6 *Application Programming Interface (API)*

Application Programming Interface atau API merupakan sekumpulan aturan dan protokol yang memungkinkan integrasi serta pertukaran data antar sistem yang berbeda secara efisien. Berdasarkan penelitian Rochman dkk. (2023), implementasi REST API berfungsi sebagai arsitektur jembatan yang menghubungkan bagian *frontend* dengan *backend* melalui metode HTTP, sehingga memungkinkan pengiriman dan manipulasi data secara terstruktur tanpa mengganggu kompleksitas internal sistem. Dalam pengembangannya, API berperan vital dalam menjamin sinkronisasi data yang responsif antara antarmuka aplikasi dengan layanan *database* terpusat secara *real-time* guna memastikan kelancaran komunikasi informasi serta fungsionalitas program yang optimal[14].

2.2.7 UML (*Unified Modeling Language*)

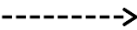
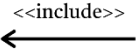
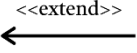
Unified Modeling Language (UML) merupakan bahasa pemodelan standar yang digunakan dalam pengembangan perangkat lunak untuk menggambarkan sistem secara visual. UML membantu pengembang dalam merancang, menjelaskan, serta mendokumentasikan sistem dengan cara yang terstruktur dan mudah dipahami. Ibarat seorang arsitek yang membuat gambar rancangan bangunan sebelum proses pembangunan dimulai, pengembang perangkat lunak juga menggunakan diagram UML sebagai pedoman dalam membangun sistem agar sesuai dengan perencanaan yang telah dibuat. Dengan memahami simbol dan aturan dalam UML, proses komunikasi antaranggota tim menjadi lebih efektif karena alur, struktur, dan desain sistem dapat dijelaskan secara lebih jelas.

Langkah ini mencakup pengumpulan dan penganalisaan data hingga dibutuhkan untuk menghasilkan produk. Analisis difokuskan pada identifikasi kebutuhan pengguna melalui hasil observasi di Pancafarm dan wawancara dengan Dinas Pertanian. Hasil analisis ini merumuskan bahwa aplikasi harus memiliki fitur *dashboard monitoring*, grafik riwayat sensor, kontrol penyiraman otomatis dan bisa mengatur penjadwalan, serta sistem yang ringan bagi petani guna meningkatkan efisiensi kerja mereka dalam mengelola lahan.

1. *Use Case Diagram*

Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem dan merepresentasikan sebuah interaksi antara aktor dan sistem. Di dalam *Use Case Diagram* terdapat (6 enam aktor) yang merupakan sebuah gambaran entitas dari manusia atau sebuah sistem yang melakukan pekerjaan di sistem.

Tabel 2. 1 *Use Case Diagram*

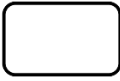
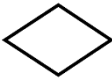
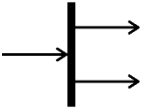
No	Gambar	Simbol	Keterangan
1.		<i>Actor</i>	Mewakili peran seorang <i>system</i> yang lain, atau alat ketika berkomunikasi dengan <i>Use Case</i> .
2.		<i>Use Case</i>	Mewakili Abstraksi dan interaksi antara sebuah <i>System</i> dan <i>Actor</i> .
3.		<i>Association</i>	Abstraksi penghubung antar <i>Actor</i> dengan <i>Use Case</i> .
4.		<i>Generalisasi</i>	Menunjukkan spesialisasi <i>actor</i> untuk dapat berpartisipasi dengan <i>Use Case</i> .
5.		<i>Include</i>	Menunjukkan bahwa suatu <i>Use Case</i> seluruhnya merupakan fungsionalitas dari <i>Use Case</i> lainnya.
6.		<i>Extend</i>	Menunjukkan bahwa suatu <i>Use Case</i> tambahan

No	Gambar	Simbol	Keterangan
			fungsi dari <i>Use Case</i> lainnya.

2. Activity Diagram

Menggambarkan alur kerja dari suatu proses dan urutan aktivitas, yang dimodelkan sebagai alur kerja dari satu aktivitas ke aktivitas berikutnya atau dari satu aktivitas ke aktivitas lainnya. Di dalam *Activity Diagram* terdapat beberapa simbol yang digunakan untuk menggambarkan alur aktivitas dan proses kerja dalam sistem.

Tabel 2. 2 *Activity Diagram*

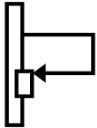
No	Gambar	Simbol	Keterangan
1		Initial node	Merupakan awal aktivitas yang akan dibuat
2		Activity	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi
3		Decicion	Menggambarkan suatu keputusan atau tindakan yang harus diambil pada kondisi tertentu.
4		Fork Join node	Digunakan untuk menunjukan kegiatan yang dilakukan secara paralel untuk dari dua kegiatan paralel jadi satu.

No	Gambar	Simbol	Keterangan
5		Fork node	Satu aliran pada tahap yang berubah menjadi beberapa aliran.
6		Activity Final node	Merupakan akhir aktivitas yang telah dibuat.

3. *Sequence Diagram*

Menggambarkan interaksi antar objek didalam dan disekitar sistem yang berupa *message* yang digambarkan terhadap waktu. Di dalam *Sequence Diagram* terdapat beberapa simbol yang digunakan untuk alur komunikasi pesan antar objek dalam sistem.

Tabel 2. 3 *Sequence Diagram*

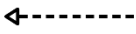
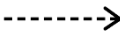
No	Gambar	Simbol	Keterangan
1.		<i>Entity Class</i>	Menggambarkan awal dari sebuah sistem, menjadi dasar penyusunan basisdata.
2.		<i>Boundary Class</i>	Menggambarkan komunikasi antara sistem seperti <i>Form</i> .
3.		<i>Control Class</i>	Penghubung antara <i>boundary</i> dengan <i>table</i> .
4.		<i>Recursive</i>	Mengirimkan pesan yang akan dikirim ke dirinya sendiri
5.		<i>Activation</i>	Menghubungkan proses durasi aktivasi sebuah proses

No	Gambar	Simbol	Keterangan
6.		<i>Life Line</i>	Menghubungkan tiap-tiap objek yang mana terdapat <i>activation</i>
7.		<i>Message</i>	Sebagai pengirim pesan ke tiap-tiap <i>Class</i>

4. *Class Diagram*

Mengambaran struktur dan deskripsi dari *class*, *package* dan objek yang saling berhubungan seperti diantaranya pewarisan, asosiasi dan lainnya di dalam *Class Diagram* terdapat beberapa simbol yang digunakan untuk hubungan dan struktur data dalam sistem.

Tabel 2. 4 *Class Diagram*

No	Gambar	Simbol	Keterangan
1		<i>Class</i>	Menggambarkan kumpulan objek-objek dengan atribut dan operasi yang sama
2		<i>Nary Association</i>	Sebagai upaya menghindari asosiasi dengan lebih dari 2 objek
3		<i>Collaboration</i>	Urutan aksi-aksi yang ditampilkan oleh sistem yang menghasilkan suatu hasil yang tertukar bagi suatu aktor
4		<i>Generalization</i>	Menggambarkan dimana hubungan objek descendent berperilaku dan struktur data objek
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek
6		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri akan mempengaruhi elemen yang tidak mandiri.