BAB II
TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Penelitian tugas akhir ini memerlukan studi literatur dalam bentuk beberapa karya ilmiah seperti, jurnal dan skripsi sebagai bahan pertimbangan untuk memperkuat bahwa penelitian ini merupakan penelitian asli serta dapat membedakan penelitian ini dengan penelitian terdahulu. Jurnal dan skripsi yang dijadikan referensi berkaitan dengan tema penelitian yaitu perancangan *Backend* dengan metode *Scrum*. Penjelasan mengenai pustaka rujukan dapat dilihat sebagai berikut.

Penelitian penelitian pertama mengimplementasikan layanan Pengaduan Aspirasi Masyarakat berbasis *web* menggunakan arsitektur *Microservices Springboot* [14]. Pendekatan *microservices* memecah aplikasi menjadi komponen kecil yang saling terhubung, membentuk proses bisnis yang terpadu. Penelitian sebelumnya mengungkapkan tantangan kompleks dalam penerapan *E-government* di Indonesia, termasuk kendala ekonomi, isu teknis, dan kekurangan sumber daya manusia [14]. Penelitian ini bertujuan untuk mengembangkan sebuah platform web untuk layanan Pengaduan Aspirasi Masyarakat yang dapat diakses dengan biaya investasi yang rendah. Metode pengembangan perangkat lunak yang digunakan adalah *Waterfall*, dengan mengumpulkan data dan informasi melalui studi literatur. Aplikasi SIAP dirancang untuk memungkinkan masyarakat untuk menyampaikan aspirasi, aduan, dan keluhan mereka kepada pemerintah daerah. Tujuannya adalah menciptakan komunikasi dua arah antara pemerintah dan masyarakat serta memperbaiki fasilitas dan layanan masyarakat.

Penelitian kedua mengembangkan *Backend* sistem manajemen *gymship* berbasis *website* dengan menggunakan *ERD (Entity Relationship Diagram)* dan *SwaggerHub*. *SwaggerHub* adalah platform terintegrasi untuk desain dan dokumentasi *API (Application Programming Interface)* yang meningkatkan konsistensi dan disiplin dalam pengembangan *API* [15].

Peneliti menemukan masalah seperti kesulitan pengelolaan data kedatangan wisatawan dan proses registrasi anggota yang tidak efisien. Metode analisis kebutuhan informasi organisasi digunakan dengan *Entity Relationship Diagram* untuk memodelkan kebutuhan informasi dan *SwaggerHub* untuk mendokumentasikan *API*. Pengujian dilakukan dengan menggunakan *Test Case* untuk memverifikasi persyaratan dan memastikan fitur aplikasi berfungsi dengan benar. Hasilnya adalah *Backend* sistem manajemen *gymship* yang memudahkan pendaftaran anggota, pemilihan instruktur, dan jadwal latihan, serta mengelola data pelanggan dan fasilitas *gymship*.

Pada penelitian ketiga, peneliti mengembangkan sistem *Backend* untuk aplikasi SECONDHAND, sebuah platform online yang memfasilitasi jual beli barang bekas, dengan menggunakan *Framework Spring Boot*. Penjualan barang bekas saat ini banyak dilakukan dengan cara mempromosikan kepada orang-orang terdekat. Beberapa penjual juga memanfaatkan media sosial sebagai media promosi. Interaksi antara penjual dan para pembeli biasanya menggunakan *SMS* atau melalui komentar di media sosial. Setelah terjadi kesepakatan, uang ditransfer oleh pembeli ke rekening penjual, dan barang diantar dengan menggunakan layanan kurir. Namun, penelitian menemukan bahwa transaksi melalui media sosial mudah terjadi penipuan [16]. Untuk itu, peneliti membuat aplikasi jual beli yang bertujuan untuk membina, mengatur, dan mengembangkan hubungan dagang secara *online* [16] Dalam penelitian ini, digunakan *Framework Spring Boot* yang populer dalam pengembangan *Backend* aplikasi *Microservices* berbasis *Java*. Peniliti menggunakan Arsitektur *Microservices* yang digunakan untuk memecah fungsi aplikasi menjadi beberapa layanan mikro terhubung yang membentuk satu aplikasi dengan proses bisnis yang utuh [16]. Penelitian ini menggunakan metode *Scrum* untuk mengembangkan perangkat lunak, dan melakukan pengujian dengan bantuan perangkat lunak *Postman*.

Penelitian keempat bertujuan untuk mengembangkan aplikasi E-Surat Dinas Komunikasi dan Statistik Provinsi Bali yang dapat membantu proses pembuatan surat, mengurangi kesalahan, dan mempercepat proses pengarsipan surat-surat dinas [17]. Dalam penelitian ini, ditemukan beberapa permasalahan terkait

pembuatan Surat Tugas dan pembuatan Surat Perjalanan Dinas. Saat ini, proses pembuatan surat masih menggunakan *Microsoft Word* dengan pengaturan manual, hal ini menimbulkan format surat yang tidak sama dan kesalahan dalam menulis surat. Pengarsipan surat juga masih dilakukan secara manual, meningkatkan risiko kehilangan atau kerusakan. Penelitian sebelumnya membahas tentang sistem informasi pembuatan surat perjalanan dinas di Kantor Wilayah Direktorat Jenderal Perbendaharaan menjadi acuan dalam menyelesaikan permasalahan ini. Tujuan penelitian ini adalah menciptakan aplikasi E-Surat Dinas Komunikasi dan Statistik Provinsi Bali untuk mempermudah pembuatan surat tugas dan SPPD, serta mengelola pengarsipan surat. Aplikasi ini memiliki tiga level pengguna: *super admin*, *admin*, dan *User*. Pengembangan aplikasi ini menggunakan metode *SDLC (Software Development Lifecycle)* dengan hosting baliprov.go.id dan *Database MySQL*. Pengujian *BlackBox* dilakukan untuk memastikan *output* aplikasi sesuai dengan desain yang telah ditentukan. Hasil penelitian ini adalah analisis dan pengembangan aplikasi E-Surat Dinas Komunikasi dan Statistik Provinsi Bali yang memudahkan pembuatan surat, mengurangi kesalahan, dan meningkatkan pengarsipan surat. Diharapkan aplikasi ini dapat meningkatkan kinerja pegawai Dinas Komunikasi dan Statistik Provinsi Bali.

Penelitian kelima ini membahas tentang pengembangan *Backend* untuk sistem pencarian pekerjaan bernama *Jobbie*. Aplikasi ini dirancang untuk membantu mengatasi permasalahan pengangguran dengan menyediakan platform bagi masyarakat untuk mencari pekerjaan atau mencari pekerja yang sesuai [18]. Fitur utama yang ditawarkan oleh aplikasi *Jobbie* adalah kemampuan untuk membuat lowongan pekerjaan dan melamar pekerjaan yang dipublikasikan di dalam aplikasi. Pengguna yang membutuhkan bantuan pekerjaan dapat menemukan seseorang yang sesuai dan memberikan imbalan (uang) setelah pekerjaan selesai dilakukan. *Jobbie* menyediakan berbagai jenis pekerjaan, baik yang membutuhkan keahlian profesional maupun yang non-profesional. Dengan hadirnya aplikasi *Jobbie*, diharapkan dapat mengurangi tingkat pengangguran di Indonesia, meningkatkan taraf hidup masyarakat, serta berkontribusi dalam kemajuan ekonomi negara. Untuk mengembangkan aplikasi *Jobbie*, peneliti memanfaatkan

Framework Laravel yang dapat mempersingkat waktu pembuatan. Penggunaan *API* pada *Backend* aplikasi *Jobbie* juga digunakan untuk mempermudah pertukaran data antara *server* dan berbagai *client*. Untuk menjamin aplikasi berjalan dengan lancar, dilakukan dua jenis pengujian yaitu *Unit Testing* dan *Blackbox Testing*. Penelitian ini menghasilkan *Backend* aplikasi *Jobbie* yang memberikan fitur-fitur yang dibutuhkan untuk mengoperasikan aplikasi *Jobbie*.

Pada Tabel 2.1 menjelaskan ringkasan dari penelitian sebelumnya mengenai metode *Scrum* dan metode *Blackbox Testing*.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Judul	Metode	Hasil
1.	Dedy Puji Jayanto	Rancang Bangun *Backend* SIAP: Sistem Informasi Aspirasi dan Pengaduan Masyarakat berbasis *Web* dengan menggunakan Metode *Microservice Springboot* [15].	*Waterfall*	Penelitian ini mengembangkan *Backend* aplikasi SIAP yang memenuhi kebutuhan dalam pelaporan pengaduan dan aspirasi masyarakat. Proses operasional aplikasi SIAP dimulai dengan registrasi daerah. Pemerintah daerah melakukan pendaftaran terlebih dahulu dan mengisi data yang diperlukan. Salah satu fitur yang ditawarkan oleh aplikasi SIAP adalah kemudahan dalam menyampaikan keluhan. Warga hanya perlu mengakses menu "Buat Aduan" yang berada di halaman depan aplikasi untuk mengirimkan laporan mereka. Pada menu pembuatan aduan, warga memasukkan informasi yang diperlukan

No	Peneliti	Judul	Metode	Hasil
				melalui formulir “pembuatan aduan”. Untuk melanjutkan proses penanganan aduan yang valid, silakan tekan tombol "Terima". Aduan yang tidak memenuhi syarat bisa dibatalkan dan tidak akan ditindaklanjuti. Aduan yang lolos verifikasi akan ditampilkan di menu "Aduan Telah Diterima".
2	Muhammad Azril Wijaya, Muhammad Faqih Dzulqarnain & Safri Adam	Rancang Bangun *Backend* Sistem Manajemen *Gymship* berbasis *Website* pada *Alterra Academy* [19].	*Software Development Lifecycle*	Penelitian ini mengembangkan sebuah *Backend* untuk *Gymship*, yang membantu para anggota untuk melakukan pendaftaran, memilih instruktur, dan menjadwalkan latihan dengan mudah. Sistem manajemen *Gymship* ini memungkinkan pengelolaan data pelanggan dan fasilitas *Gymship* yang lebih efisien. Selain itu, sistem ini juga

No	Peneliti	Judul	Metode	Hasil
				mempercepat dan mempermudah pembuatan laporan keuangan mengenai pengeluaran dan pemasukan bulanan.
3.	Tania Prilsafira, Yesi Novaria Kunang & Muhammad Hatta Putra	*REST API Backend* Aplikasi *E-Commerce* SECONDHAND Menggunakan *Framework Spring Boot* [16].	*Scrum*	Penelitian ini menghasilkan sebuah sistem *Backend* untuk aplikasi SECONDHAND yang merupakan aplikasi untuk menjual barang-barang *second* yang dibangun dengan *Framework Spring Boot*, dan menggunakan bahasa pemrograman *Java*. Format pertukaran data yang digunakan menggunakan *JSON*, data menggunakan *Database PostgreSQL* dan menggunakan metode pengembangan perangkat lunak *Scrum*.

No	Peneliti	Judul	Metode	Hasil
4.	Gede Indra Rudyarta & Made Widiartha	Pengembangan Aplikasi *Backend* Pembuatan Surat (E-Surat) Dinas Komunikasi, Informatika, dan Statistik Provinsi Bali [17].	*Software Development Lifecycle*	Penelitian ini menghasilkan analisis dan pengembangan aplikasi E-Surat untuk Dinas Komunikasi, Informatika, dan Statistik Provinsi Bali. Aplikasi ini dirancang untuk memudahkan pembuatan surat dengan cepat dan akurat, serta meningkatkan pengarsipan surat. Penggunaan aplikasi ini diharapkan dapat meningkatkan kinerja pegawai Dinas Komunikasi, Informatika, dan Statistik Provinsi Bali secara keseluruhan. Proses pengembangan aplikasi E-Surat ini mengikuti metode *SDLC* dan menggunakan *hosting* baliprov.go.id serta *Database MySQL*. Hasil pengujian *Blackbox Testing* menunjukkan bahwa aplikasi berjalan dengan lancar dan

No	Peneliti	Judul	Metode	Hasil
				sesuai dengan skenario uji yang telah ditentukan.
5.	Bima Syaftiaan, Nadia Fitri Safira & Febrian Rizky	Rancang Bangun *Backend* Aplikasi *Jobbie*: Pencarian dan Penyedia Jasa Lapangan Kerja [18].	*Extreme Programming*	Hasil dari penelitian ini adalah pengembangan *Backend* untuk aplikasi *Jobbie*, sebuah platform pencarian dan penyediaan jasa lapangan pekerjaan. Dengan adanya aplikasi ini, masyarakat dapat dengan mudah mencari pekerjaan atau mencari pekerja yang sesuai dengan kebutuhan mereka, sehingga dapat membantu mengatasi masalah pengangguran. Pembuatan aplikasi *Jobbie* menggunakan metode *Extreme Programming (XP)* dalam tahap pengembangannya. *Framework Laravel* digunakan untuk mempercepat proses pembuatan aplikasi *Jobbie*. Untuk menjamin

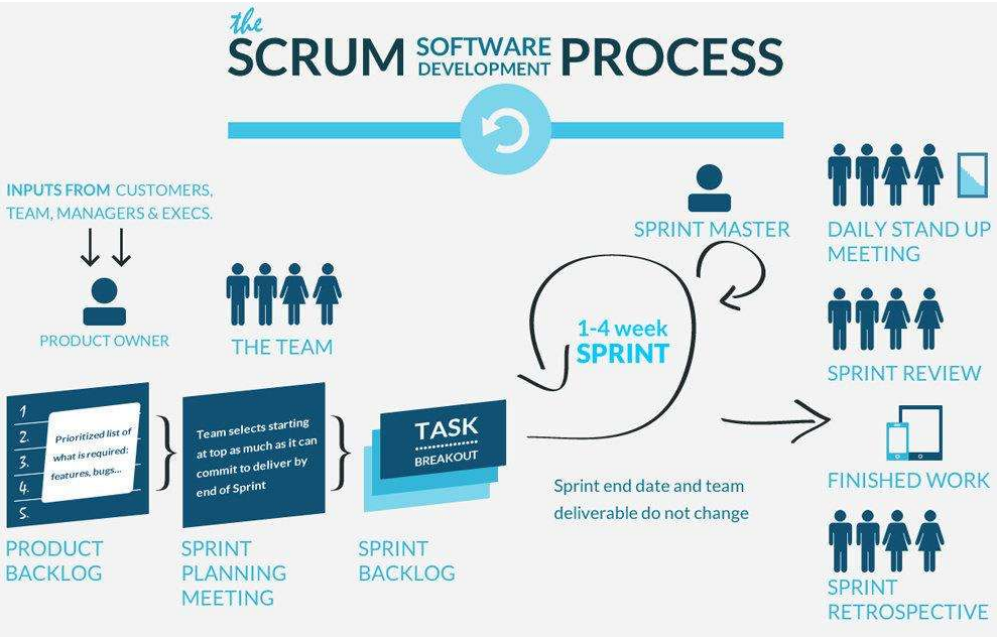
No	Peneliti	Judul	Metode	Hasil
				aplikasi berjalan dengan lancar, dilakukan dua jenis pengujian yaitu Unit Testing dan *BlackBox Testing*. Penelitian ini menghasilkan *Backend* aplikasi *Jobbie* yang memberikan fitur-fitur yang dibutuhkan untuk mengoperasikan aplikasi *Jobbie*.

2.2 Landasan Teori

Berikut merupakan kajian tentang beberapa teori yang diterapakan pada penelitian ini:

2.2.1 *Scrum*

Scrum merupakan kerangka kerja *agile* yang memungkinkan pengembangan perangkat lunak yang fleksibel dan sesuai dengan kebutuhan pengguna. Dalam *Scrum*, pengembangan dibagi menjadi bagian-bagian kecil dan berulang [8]. Terdapat tiga stakeholder penting dalam *Scrum*: *Product Owner*, *Scrum Master*, dan *Development Team*. *Product owner* menentukan kebutuhan pengguna dan mengatur prioritas fitur, *Scrum Master* memastikan penerapan *Scrum* yang efektif, dan *Development Team* bertanggung jawab dalam mengerjakan pengembangan perangkat lunak. Dengan melibatkan ketiga stakeholder ini, *Scrum* memungkinkan pengembangan perangkat lunak yang lebih adaptif, kolaboratif, dan sesuai kebutuhan pengguna. Secara umum proses *Scrum* ditunjukkan pada Gambar 2.1:



Gambar 2.1 Scrum Software Development Process [17]

Selain konsep tersebut, metode *Scrum* juga melibatkan tiga *stakeholder* penting [8]:

1. *Product Owner*: *Product Owner* adalah perwakilan dari pemangku kepentingan dan bertanggung jawab untuk menentukan kebutuhan pengguna serta mengatur prioritas fitur-fitur yang akan dikembangkan.
2. *Scrum Master*: *Scrum Master* adalah pemimpin tim yang bertanggung jawab untuk memastikan bahwa proses pengembangan menggunakan metode *Scrum* secara efektif. Tugas utama *Scrum Master* adalah menghilangkan hambatan yang dihadapi tim pengembangan, memfasilitasi komunikasi, dan memastikan bahwa prinsip-prinsip dan praktik *Scrum* diterapkan dengan baik.
3. *Development Team*: *Development Team* terdiri dari anggota-anggota yang berkolaborasi dalam mengerjakan pekerjaan pengembangan perangkat lunak. Tim ini biasanya multi-disiplin dan mandiri, memiliki keahlian yang diperlukan untuk menghasilkan increment pada setiap *Sprint*.

Berikut ini adalah langkah-langkah dari Metode *Scrum* yang harus dilakukan [8]:

1. *Product Backlog*: Tahapan pertama dalam *Scrum* adalah membuat *Product Backlog*. Ini adalah daftar prioritas yang berisi semua fitur, fungsionalitas, perbaikan, dan perubahan yang diinginkan untuk produk yang sedang dikembangkan. *Product Backlog* dapat diperbarui dan disesuaikan secara teratur sepanjang proyek.
2. *Sprint Planning*: Setelah memiliki *Product Backlog*, tahap berikutnya adalah *Sprint Planning*. Dalam tahap ini, tim *Scrum*, yang terdiri dari *Product Owner* dan anggota tim pengembangan, berkumpul untuk merencanakan *Sprint* berikutnya. Mereka berkolaborasi untuk memilih *item* dari *Product Backlog* yang akan dimasukkan ke dalam *Sprint Backlog*. Mereka juga menentukan tujuan *Sprint* dan merencanakan pekerjaan yang diperlukan untuk mencapainya.

3. *Sprint Backlog*: Setelah *Sprint Planning*, tahap berikutnya adalah membuat *Sprint Backlog*. *Sprint Backlog* adalah *subset* dari *Product Backlog* yang dipilih oleh tim *Scrum* untuk dikerjakan selama *Sprint* tertentu. Ini berisi item pekerjaan yang diperinci lebih lanjut, dengan estimasi waktu yang lebih spesifik untuk setiap *item*. *Sprint Backlog* berfungsi sebagai panduan bagi tim *Scrum* selama *Sprint* untuk menyelesaikan pekerjaan yang telah dipilih.
4. *Sprint*: *Sprint* adalah periode waktu terbatas di mana tim *Scrum* bekerja untuk mencapai tujuan *Sprint* yang telah ditetapkan. Durasi *Sprint* biasanya antara 1 hingga 4 minggu. Selama *Sprint*, tim *Scrum* melakukan pekerjaan yang telah dipilih dalam *Sprint Backlog*. Mereka melakukan pertemuan harian yang disebut *Daily Scrum* untuk melakukan sinkronisasi, memantau kemajuan, mengidentifikasi hambatan, dan membuat penyesuaian jika diperlukan.
5. *Daily Scrum*: *Daily Scrum* adalah pertemuan harian yang singkat di mana semua anggota tim *Scrum* berbagi perkembangan pekerjaan mereka. Dalam pertemuan ini, setiap anggota tim menjawab tiga pertanyaan: "Apa yang telah saya kerjakan sejak pertemuan terakhir?", "Apa yang akan saya kerjakan hari ini?", dan "Apakah ada hambatan yang menghalangi kemajuan saya?". Tujuannya adalah memastikan keterbukaan komunikasi dan koordinasi yang efektif di antara anggota tim.
6. *Sprint Review*: Setelah selesai *Sprint*, tim *Scrum* melakukan *Sprint Review*. Ini adalah pertemuan di mana mereka memperlihatkan hasil kerja mereka kepada *stakeholder*, termasuk *Product Owner*. Mereka mengevaluasi apa yang telah dicapai selama *Sprint*, memperoleh umpan balik, dan berdiskusi tentang langkah selanjutnya berdasarkan umpan balik tersebut. *Sprint Review* membantu dalam memvalidasi dan mengarahkan perkembangan produk.
7. *Sprint Retrospective:* Setelah *Sprint Review*, tim *Scrum* melakukan *Sprint Retrospective*. Ini adalah pertemuan refleksi di mana tim *Scrum* mengevaluasi cara mereka bekerja dan memperbaiki proses pengembangan mereka. Mereka mengidentifikasi apa yang berjalan dengan baik dan apa yang bisa

ditingkatkan dalam *Sprint* berikutnya. Tujuannya adalah untuk terus melakukan perbaikan secara berkelanjutan.

Dengan melalui tahap-tahap ini, Metode *Scrum* memungkinkan pengembangan perangkat lunak yang adaptif dan sesuai dengan kebutuhan pengguna.

2.2.2 *Social Networking*

Social Networking mengacu pada praktik atau proses berinteraksi dan berkomunikasi antara individu atau kelompok melalui platform online. Hal ini memungkinkan pengguna untuk terhubung, berbagi informasi, dan membangun hubungan dengan orang-orang yang memiliki minat, latar belakang, atau tujuan yang serupa [20]. Platform *Social Networking* umumnya menyediakan fitur-fitur seperti pembuatan profil pengguna, pengiriman pesan pribadi, berbagi konten seperti gambar dan video, serta kemampuan untuk mengikuti atau berinteraksi dengan pengguna lain melalui *like*, komentar, dan berbagi. Beberapa contoh populer platform *Social Networking* termasuk *Facebook, Twitter, Instagram, LinkedIn,* dan banyak lagi.

Melalui *Social Networking*, individu dapat memperluas jaringan sosial mereka, menjaga hubungan dengan teman, keluarga, atau rekan kerja, berpartisipasi dalam komunitas dengan minat yang sama, mendapatkan informasi terkini, atau mempromosikan produk dan layanan. Selain itu, *Social Networking* juga dapat digunakan untuk tujuan pendidikan, kampanye sosial, dan kolaborasi dalam berbagai bidang. *Social Networking* telah mengubah cara kita berinteraksi dan berkomunikasi, memungkinkan konektivitas global dan pertukaran informasi yang lebih mudah. Namun, penting untuk menyadari privasi dan keamanan dalam menggunakan platform *Social Networking* serta memahami bagaimana memanfaatkannya dengan bijak [20].

2.2.3 *Web Service*

Web service adalah sistem perangkat lunak yang memungkinkan komunikasi antara aplikasi yang berbeda melalui jaringan. *Web service* menggunakan standar

dan protokol yang umum, seperti *Hypertext Transfer Protocol (HTTP)*, *Extensible Markup Language (XML)*, *Simple Object Access Protocol (SOAP)*, dan *Web Services Description Language (WSDL)*, untuk menyediakan layanan yang dapat diakses oleh klien yang beragam [21]. *Web service* memudahkan integrasi dan interoperabilitas antara sistem yang berbeda, karena mereka tidak bergantung pada platform, bahasa pemrograman, atau arsitektur tertentu. *Web service* dapat menyediakan fungsi-fungsi seperti otentikasi, enkripsi, pencatatan, dan penemuan layanan. Dengan menggunakan *Web service*, pengguna dapat mengatasi tantangan seperti interoperabilitas dan mengintegrasikan sistem yang berbeda. *Web service* memfasilitasi interaksi dan pertukaran data antara aplikasi yang berbeda melalui jaringan komputer [22].

2.2.4 *Application Programming Interface (API)*

Application Programming Interface (API) adalah sebuah *interface* yang berfungsi sebagai perantara antara aplikasi yang satu dengan yang lainnya, baik yang berada di platform yang sama maupun yang berbeda. Dengan adanya kemajuan ini, *Developer* dapat membuat aplikasi yang berdasarkan pada aplikasi yang sudah ada sebelumnya. Dalam aplikasi *web*, *API* dipanggil menggunakan protokol *HTTP* dengan menggunakan alamat tertentu, dan akan menghasilkan respon data dalam format seperti *Hyper Text Markup Language (HTML)*, *Javascript Object Notation (JSON)*, atau *Extensible Markup Language (XML)* [23].

2.2.5 *Representational State Transfer (REST)*

Representational State Transfer (REST) adalah salah satu jenis arsitektur yang digunakan dalam *web* untuk menyediakan layanan. *REST* digunakan untuk mengatur interaksi antara *server* dan *client* dalam pertukaran informasi melalui media yang sama dalam suatu jaringan. Untuk mengakses sumber daya (*resource*), identifikasi dan manipulasi diperlukan. *URI (Uniform Resource Identifier)* dapat digunakan untuk mengidentifikasi sumber daya yang ada dalam jaringan, dan ini memungkinkan sumber daya menjadi "*addressable*" yang berarti sumber daya tersebut dapat ditemukan dan dimanipulasi menggunakan aplikasi tertentu [24].

REST didasarkan pada sejumlah prinsip dan batasan yang memungkinkan pengembangan sistem yang terdistribusi, skalabel, dan interoperabel. Beberapa konsep penting dalam *REST* meliputi:

1. Representasi

   Data dalam *REST* direpresentasikan dalam bentuk sumber daya (*resources*), yang dapat berupa dokumen *teks*, gambar, video, atau jenis data lainnya. Setiap sumber daya memiliki identifikasi unik *(URI)* dan dapat diakses melalui *HTTP*.

2. *Statelessness*

   REST didesain agar klien dan *server* tidak menyimpan status (*state*) satu sama lain. Setiap permintaan dari klien ke *server* harus mencakup semua informasi yang diperlukan untuk memahami dan memproses permintaan tersebut. Tidak ada penyimpanan status sesi di *server*, sehingga memungkinkan skalabilitas dan replikasi yang lebih mudah.

3. Metode

   REST menggunakan metode *HTTP* seperti *GET*, *POST*, *PUT*, dan *DELETE* untuk berinteraksi dengan sumber daya. Setiap metode memiliki semantik yang berbeda, seperti *GET* untuk mendapatkan data, *POST* untuk membuat data baru, *PUT* untuk mengubah data, dan *DELETE* untuk menghapus data.

4. *Hypermedia* sebagai Mesin Negosiasi *(HATEOAS)*

   REST mendorong penggunaan hypermedia untuk menyediakan tautan (*links*) yang dinamis antara sumber daya terkait. Ini memungkinkan klien untuk menavigasi melalui aplikasi secara dinamis, mengikuti tautan yang disediakan oleh server.

5. *Javascript Object Notation (JSON)*

   JSON adalah singkatan dari *JavaScript Object Notation*. Format ini sering digunakan untuk mentransfer ataupun menyimpan suatu data. *JSON* sangat berbeda jika dibandingkan dengan *XML* atau format lain dengan fungsi yang sama. *JSON* menggunakan sintaksis yang lebih ringkas dan mudah dipahami daripada *XML* [25].

2.2.6 *MongoDB*

MongoDB adalah basis data *Not Only SQL (NoSQL)* yang bersifat *document based* [26]. *MongoDB* bersifat *document based* artinya *MongoDB* tidak memilki Tabel, kolom ataupun baris. *MongoDB* hanya memilki koleksi dan dokumen. Data yang disimpan dalam basis data *MongoDB* berupa file *JSON* yang disebut dengan istilah *Binary JSON BSON)*. Sistem basis data *MongoDB* menggunakan *key-value*, artinya setiap dokumen dalam *MongoDB* dipastikan memilki *key.*

2.2.7 *Kotlin*

Kotlin adalah sebuah bahasa pemrograman yang berjalan di atas *Java Virtual Machine (JVM)*. Bahasa ini dirancang secara pragmatis untuk pengembangan aplikasi *Android* dan menggabungkan paradigma *Object-Oriented Programming (OOP)* dan *Functional Programming (FP)* [27]. Selain itu, *Kotlin* memiliki interoperabilitas yang memungkinkannya digunakan bersama dengan kode bahasa *Java* dalam satu proyek. Bahasa ini memiliki banyak keunggulan, seperti kemampuan untuk membuat aplikasi *Desktop, Web,* dan *Backend*. *Kotlin* diciptakan oleh *JetBrains*, perusahaan yang membuat *IntelliJ IDEA*, salah satu *Integrated Development Environment (IDE)* terbaik di dunia. Setelah melalui banyak pengembangan, *Kotlin* kemudian dirilis secara *open source* dan terus mengalami perkembangan yang pesat.

Kotlin menawarkan beberapa fitur yang berguna, seperti kemampuan untuk menulis kode dengan lebih ringkas dan mengurangi jumlah kode *boilerplate* [28]. Selain itu, *Kotlin* juga memberikan keamanan tambahan dengan menghindari kesalahan umum seperti *Null Pointer Exceptions* pada seluruh kelas. Karena dapat memanfaatkan library yang tersedia untuk *JVM*, bahasa ini memiliki kemudahan dalam penggunaannya. *Kotlin* merupakan bahasa Pemrograman terbuka dan mendapatkan dukungan yang terus meningkat di lingkungan pengembangan *Android Studio*.

2.2.8 *Ktor*

Ktor merupakan sebuah *framework Backend* yang digunakan untuk membangun aplikasi *Web* dan layanan *Web RESTful*. *Ktor* dikembangkan menggunakan bahasa pemrograman *Kotlin* dan menggunakan *Kotlin coroutines* untuk mendukung pemrograman secara *asynchronous*. *Framework* ini memberikan kemudahan dalam membangun aplikasi yang responsif dan memberikan pengalaman terbaik bagi pengguna [29].

Ktor merupakan *framework* yang ringan dan fleksibel, sehingga Anda dapat menggunakan hanya fitur-fitur yang diperlukan dan mengatur struktur aplikasi sesuai kebutuhan. *Framework* ini ditulis dan didesain menggunakan *Kotlin*, yang merupakan bahasa yang ringkas dan dapat digunakan di berbagai platform [29]. Dengan menggunakan *Ktor*, Anda dapat mengembangkan aplikasi *Backend* secara *asynchronous* dan membuat aplikasi *client* dan *server-side*. *Ktor* memberikan kemampuan untuk membuat rute-rute (*routes*) baru sesuai dengan kebutuhan Anda. Anda dapat menambahkan rute-rute baru di dalam *scope Routing* dengan menggunakan *extension functions* yang disediakan oleh *Ktor* [30].

2.2.9 *Unified Modeling Language (UML)*

Bahasa pemodelan *Unified Modeling Language (UML)* digunakan untuk membuat sistem perangkat lunak yang menggunakan objek-objek. [31]. *UML* digunakan untuk pemodelan dan komunikasi di sekitar sistem yang menggunakan skema dan skrip pendukung. *UML* bertujuan untuk menyediakan bahasa pemodelan yang independen dari berbagai bahasa pemrograman dan proses rekayasa. Selain itu, penggunaan *UML* memiliki penggunaan standarisasi praktik terbaik saat ini dalam pemodelan dan menyediakan model siap pakai [32].

2.2.10 *Unit Testing*

Unit testing merupakan tahap dasar dalam pengujian. Unit testing berfokus pada penguijian building block yang lebih kecil daripada program atau sistem yang diuji. Pengujian ini mengeksekusi setiap fitur atau fungsionalitas untuk memastikan

masing-masing fitur atau fungsionalitas tersebut berfungsi sesuai dengan yang diharapkan. Pada definisi lainnya, unit testing merupakan pengujian yang mencakup pengujian sepotong atau sebagian kode. Dengan demikian, pengujian ini dapat digunakan untuk menguji kelas atau modul tertentu [12].

2.2.11 *Blackbox Testing*

Pengujian *Blackbox Testing*, juga dikenal sebagai pengujian perilaku, melibatkan pengujian perangkat lunak tanpa pengetahuan tentang struktur internal atau logika yang diuji oleh penguji. Penguji mengacu pada spesifikasi kebutuhan dan tidak perlu menganalisis kode secara mendetail. Pendekatan pengujian *Blackbox* dilakukan dari perspektif pengguna akhir [33]. Setelah menulis kode program, program diuji untuk memverifikasi dan menjamin bahwa komponen-komponen berjalan sesuai ekspektasi. Uji ini penting untuk mengidentifikasi kesalahan atau kekurangan yang mungkin terdapat [34].