BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2. 1 Tinjauan Pustaka

Penelitian ini berlandaskan pada berbagai studi yang membahas implementasi teknologi berbasis *web* dan *web service* untuk mendukung pengelolaan data dan penyajian informasi secara efisien.

Herdiyatmoko (2022) mengembangkan sebuah aplikasi *back-end* berbasis *Restful API* menggunakan *Laravel Framework*. Penelitian ini bertujuan untuk menyediakan layanan data yang dapat diakses oleh klien multiplatform dalam format *JSON* melalui protokol *HTTP*. Hasilnya adalah *REST Server* yang mampu mendukung interoperabilitas melalui *endpoints JSON* dan status kode yang terintegrasi.

Putra dan Danuri (2018) merancang sistem pemetaan demografi berbasis digital untuk tingkat desa dengan integrasi *Google maps API*. Sistem ini memungkinkan pengguna, termasuk pejabat daerah, untuk mengetahui sebaran populasi serta data statistik seperti tingkat pendidikan, pekerjaan, usia, dan tanggungan keluarga. Hasil penelitian menghasilkan sistem berbasis *web* yang dapat menampilkan peta desa dengan batas-batas wilayah hingga tingkat RT/RW serta menyajikan data kependudukan dalam bentuk grafik.

Ikhwanuzaki dan Handayani (2024) mengimplementasikan teknologi *Restful API* untuk mendukung proses pemesanan sarung goyor di Perusahaan Suhutex. Sistem ini dirancang untuk mengintegrasikan aplikasi *web* dan *mobile*, memungkinkan pengolahan data produk seperti gambar, harga, dan kategori. Dengan metode *Service-Oriented Architecture (SOA)*, penelitian ini membuktikan bahwa teknologi *Restful API* mampu meningkatkan efisiensi pemasaran, rek*api*tulasi data, dan penyampaian informasi kepada pelanggan.

Ariandi dan Agustini (2018) mengembangkan Sistem Informasi Geografis (GIS) untuk menganalisis penyebaran penduduk di Kecamatan Rambutan. Sistem ini menampilkan data geografis terkait distribusi penduduk berdasarkan kategori seperti jenis kelamin, usia sekolah, dan usia manula. Dengan metode *action research*, penelitian ini menghasilkan sistem berbasis GIS yang memberikan visualisasi data penduduk secara geografis untuk mendukung pengambilan keputusan.

Gowell dan Suprihadi (2024) merancang sistem berbasis *REST API* untuk layanan perjalanan wisata di Tenta Tour Salatiga. Sistem ini menggunakan model prototyping dan diuji untuk memastikan kinerja yang cepat dan stabil. Penelitian ini menghasilkan *web service* yang menyediakan informasi tentang paket perjalanan, fitur reservasi, dan kustomisasi. Teknologi ini mendukung interoperabilitas dan transfer data yang efisien antara sistem *web* dan *mobile*.

Penelitian-penelitian di atas menjadi dasar yang kuat dalam mendukung pengembangan sistem berbasis *web* dan *web service* untuk pengelolaan data

demografi. Integrasi *REST API*, penggunaan GIS, dan pendekatan berbasis data menjadi elemen kunci yang relevan dalam penelitian ini.

Tabel Tinjauan Pustaka merupakan tabel yang dibuat untuk mendefinisikan penelitian yang sebelumnya hampir sama dilakukan dengan penelitian yang diajukan saat ini. Adapun perbandingan yang dijadikan tabel tinjauan pustaka yakni dijabarkan pada tabel 2.1.

Tabel 2. 1 Perbandingan Penelitian

Penulis	Objek	Masalah	Teknologi	Hasil Penelitian
Herdiyatmoko (2022)	*Back-end Sistem* berbasis *REST API*	Tidak adanya *REST Server* yang dapat diakses multiplatform	*Restful API, JSON, Laravel*	*REST Server* yang mampu menyediakan *endpoints* berbasis *JSON* yang mendukung interoperabilitas.
Putra dan Danuri (2018)	Peta digital demografi tingkat desa	Data demografi diolah manual, sulit diakses masyarakat umum	*Google maps API, PHP*	Sistem berbasis *web* yang menampilkan peta desa dengan batas wilayah dan statistik penduduk secara digital.
Ikhwanuzaki dan Handayani (2024)	*Web service REST API*	Proses pemasaran manual, rekap data rentan kesalahan	*Restful API, Web* Mobile	Sistem pemesanan sarung berbasis *REST API* yang mempermudah pemasaran, rekap data, dan penyampaian informasi.

Ariandi dan Agustini (2018)	Sistem Informasi Geografis (SIG)	Sulit mengetahui penyebaran penduduk berdasarkan lokasi geografis	GIS, Action Research	Sistem GIS yang menampilkan data penduduk berdasarkan jenis kelamin, usia, dan golongan secara geografis.
Gowell dan Suprihadi (2024)	*Web service* untuk wisata	Metode perencanaan wisata kurang efisien	*REST API*, Prototyping	Sistem *REST API* yang menyediakan informasi paket perjalanan serta mendukung fitur reservasi dan kustomisasi.
Syahrur Ro'uf (2025)	*Web service* untuk demografi penduduk	Tidak adanya sistem yang menampilkan data demografi tanpa info privat secara agregat dan efisien	*REST API*, *Laravel*, *JSON*, *Bootstrap*	Prototipe sistem yang menyajikan data demografi tingkat kabupaten hingga kelurahan berbasis *REST API* untuk akses publik.

2. 2 Dasar Teori

2. 2. 1 Demografi

Demografi adalah studi tentang penduduk yang dilihat dari ukuran (jumlah), struktur/komposisi, persebaran ke ruangan serta faktor-faktor yang mempengaruhi jumlah, struktur dan persebaran penduduk yaitu fertilitas, mortalitas dan migrasi di suatu wilayah tertentu. Dalam demografi terdapat aspek kependudukan yang statis dan dinamis sifatnya. Aspek statis ditunjukkan oleh komposisi penduduk misalnya. Komposisi penduduk merupakan gambaran kondisi penduduk pada suatu titik tertentu, yaitu pada saat dilaksanakan sensus atau survei. Sesudah tanggal atau hari tersebut, komposisi penduduk akan berubah. Perubahan komposisi ini terjadi karena

perubahan kelahiran, kematian dan migrasi. Jadi dalam demografi juga dipelajari aspek statis dan aspek dinamis, yang keduanya saling mempengaruhi. Contoh, jumlah kelahiran akan mempengaruhi jumlah penduduk muda di suatu wilayah tertentu (Sonny Harry B. Harmadi, 2019).

2. 2. 2 *Web service*

Web service adalah aplikasi sekumpulan data (database), perangkat lunak (*software*) atau bagian dari perangkat lunak yang dapat diakses secara remote oleh berbagai piranti dengan sebuah perantara tertentu. Penggunaan *web service* mampu mengatasi permasalahan *interoperability* dan mengintegrasikan sistem yang berbeda. Secara umum, *web service* dapat diidentifikasikan dengan menggunakan URL seperti hanya *web* pada umumnya (misal : www.nama*web*.com). Namun yang membedakan *web service* dengan *web* pada umumnya adalah interaksi yang diberikan oleh *web service*. Berbeda dengan URL *web* pada umumnya, URL *web service* hanya menggandung kumpulan informasi, perintah, konfigurasi atau sintaks yang berguna membangun sebuah fungsi-fungsi tertentu dari aplikasi.

Web service dapat diartikan juga sebuah metode pertukaran data, tanpa memperhatikan dimana sebuah database ditanamkan, dibuat dalam bahasa apa sebuah aplikasi yang mengkonsumsi data, dan di platform apa sebuah data itu dikonsumsi. *Web service* mampu menunjang interoperabilitas. Sehingga *web service* mampu menjadi sebuah jembatan penghubung antara berbagai sistem yang ada.

Web pada umumnya digunakan untuk melakukan respon dan *request* yang dilakukan antara *client* dan *server*. Sebagai contoh, seorang pengguna layanan *web*

tertentu mengetikan alamat url *web* untuk membentuk sebuah *request*. *Request* akan sampai pada *server*, diolah dan kemudian disajikan dalam bentuk sebuah respon. Dengan singkat kata terjadilah hubungan *client- server* secara sederhana. Sedangkan pada *web service* hubungan antara *client* dan *server* tidak terjadi secara langsung. Hubungan antara *client* dan *server* dijembatani oleh file *web service* dalam format tertentu. Sehingga akses terhadap database akan ditanggani tidak secara langsung oleh *server*, melainkan melalui perantara yang disebut sebagai *web service*. Peran dari *web service* ini akan mempermudah distribusi sekaligus integrasi database yang tersebar di beberapa *server* sekaligus (Fanny Aulia Fadilah, 2020).

2. 2. 3 Service-Oriented Architecture (SOA)

Service Oriented Architecture(SOA)adalah model pengembangan sistem informasi yang dikemas kedalam arsitektur aplikasi berbentukservice. SOA memberikan berbagai informasi dan platform yang berbeda-beda sehingga bisa dijadikan sebagai alur proses bisnis yang berbasis teknologi informasi

Metode Service Oriented Architecture (SOA)mampu menterjemahkan permasalahan secara logis kedalam unit yang lebih rinci serta saling berhubungan.Selain itu metode ini dapat digunakan pada skala besar dengan pendekatan komputasi yang terdtribusi sehingga memberikan efisiensi pada pengembangan sistem

Service oriented architecturememperkenalkan satu lapisan baru yang disebut lapisan service, yang berfungsi untuk mengenkapsulasi logik dari aplikasi terhadap logikadari bisnis. Hal ini terkait dengan salah satu karakteristik SOA,

yakni meningkatkan aspek loosely coupledantara logik bisnis dan logik aplikasi. Ketika lapisan servicedapat merepresentasikan logik dari bisnis dan aplikasi, maka ketergantungan langsung antara logik bisnis dan logik aplikasi rendah. (Muhammad Muslih dan Mutiara Islam Hasanah, 2019 (Purnama, 2018))

2. 2. 4 *Laravel*

Laravel adalah *Framework PHP* yang bersifat open source dibuat oleh Taylor Otwell dan salah satu *framework web PHP* yang sangat popular yang dirilis dibawah lisensi MIT. *Laravel* dirancang agar proses pengembangan aplikasi *web* menjadi lebih mudah dan cepat karena terdapat sejumlah fitur bawaan didalamnya. Pada umumnya *Laravel* digunakan untuk membangun aplikasi *web* disisi *server* atau back end. Dengan menggunakan *framework backend developer* dapat mengembangkan fitur-fitur yang dibutuhkan di sisi *server* seperti pengelolaan *user*, manajemen pesanan, dan lain-lain.

Sejak versi 3, *Laravel* telah berkembang menjadi salah satu *framework* yang paling popular, dan banyak digunakan pada Bahasa pemograman *PHP*. Sumber tambahan paket lainnya terdapat dalam repository *Laravel* di GitHub (Badiyanto, 2024).

2. 2. 5 *Faker*

Faker adalah pustaka *PHP* yang digunakan oleh *Laravel* Factory untuk menghasilkan data palsu dengan mudah dan realistis. *Laravel* sudah menyertakan *Faker* secara bawaan, sehingga anda tidak perlu menginstalnya secara terpisah. *Faker* menyediakan berbagai method untuk menghasilkan data acak, seperti nama

palsu, alamat palsu, teks palsu, dan sebagainya. Factory menggunakan *Faker* untuk mengisi nilai atribut-atribut model dengan data acak saat membuat data palsu (Badiyanto, 2024).

2. 2. 6 *API*

A. P. I. a.k.a. Antarmuka Pemrograman Aplikasi merupakan protokol yang terdiri atas kumpulan instruksi yang disimpan dalam bentuk *library* dan menjelaskan(mengatur) bagaimana agar suatu *software* dapat berinteraksi dengan *software* lain. Jadi, dengan adanya *API*, maka terdapat aturan bagaimana cara *software* dapat berinteraksi dengan *software* lain untuk mengakses resources(data) yang terdapat di dalam *software* tersebut melalui interface (fungsi,sintaks,protocol) yang telah tersedia tanpa perlu mengetahui bagaimana *software*/aplikasi itu dibuat.

API dapat digunakan dengan berbagai bahasa programming (pemrograman), ataupun hanya dengan menggunakan URL (Uniform Resource Locator) yang telah disediakan oleh suatu *web*site. Bahkan, ternyata kita bisa membuat / mendesain *API* kita sendiri contoh: membuat *Web API* dengan menggunakan teknologi yang berbeda seperti *PHP*, *Java, .NET,, Mysql, XML* dll. Hal ini telah dilakukan oleh Microsoft , dimana mereka mendesain *API* sendiri dengan (*Visual C++, C#, Visual Basic, F#,*dll.) yang tertanam di *System Visual Studio* mereka.

Ketika digunakan dalam konteks pengembangan *web*, sebuah *API* biasanya didefinisikan sebagai sekumpulan pesan permintaan *Hypertext Transfer Protocol* (*HTTP*), bersama dengan definisi struktur pesan respons, yang biasanya dalam *Extensible Markup Language* (*XML*) atau Format *JavaScript Object Notation*

(*JSON*). Tren terkini (disebut *Web* 2.0) telah bergerak menjauh dari layanan berbasis Access Protocol (SOAP) sederhana menuju komunikasi gaya *Representational State Transfer* (*REST*) langsung. *Web API* ini memungkinkan kombinasi beberapa layanan ke dalam satu aplikasi baru yang dikenal sebagai *mashup application* dan mengambil data / informasi / sumber daya fungsi yang disediakan aplikasi lainnya (Sinta Anjelina, 2019).

2. 2. 7 *Postman*

Postman adalah platform *API* yang digunakan untuk membangun dan menggunakan *API*. *Postman* menyederhanakan setiap langkah dalam siklus hidup *API* dan memperlancar kolaborasi, sehingga Anda dapat membuat *API* yang lebih baik dengan lebih cepat. Platform ini mencakup berbagai alat yang membantu mempercepat siklus hidup *API*, mulai dari desain, pengujian, dokumentasi, hingga berbagi dan menemukan *API*. *Postman* juga memungkinkan penyimpanan dan pengelolaan spesifikasi *API*, dokumentasi, resep alur kerja, kasus pengujian, dan hasilnya.

Postman menyediakan alat dasar yang memungkinkan pengguna untuk menjelajahi, mendebug, dan menguji *API* dengan mudah. Alat ini mendukung berbagai protokol seperti *HTTP*, *REST*, SOAP, GraphQL, dan *Web*Sockets. *Postman* juga memungkinkan pengguna untuk mengorganisir permintaan ke dalam Koleksi *Postman*, yang membantu dalam pengelolaan dan penggunaan kembali permintaan.

Pengujian *API* adalah proses yang memastikan bahwa *API* berfungsi sesuai dengan yang diharapkan. Ada beberapa jenis pengujian *API*, dan masing-masing

memiliki peran yang berbeda dalam memastikan bahwa fungsi, keamanan, dan kinerja *API* tetap dapat diandalkan. Pengembang dapat menjalankan pengujian *API* secara manual atau mengotomasinya dengan alat pengujian *API*.

Secara tradisional, pengujian *API* dilakukan pada akhir fase pengembangan, tet*api* semakin banyak tim yang melakukan pengujian lebih awal dalam siklus hidup *API*. Pendekatan pengujian *API* ini, yang dikenal sebagai "*shifting left*" mendukung iterasi cepat dengan memungkinkan tim mendeteksi dan memperbaiki masalah segera setelah masalah tersebut muncul (*Postman*).