

BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

Digunakan sebagai dasar ilmiah untuk menunjukkan perkembangan penelitian *REST API*, pengujian *endpoint*, dan teknologi *backend* yang relevan. Referensi dipilih terutama dari penelitian yang terbit dalam lima tahun terakhir agar posisi penelitian ini dapat dibandingkan dengan penelitian terdahulu. Penelitian terkait disajikan untuk menunjukkan persamaan, perbedaan, dan posisi penelitian pada implementasi serta pengujian *REST API backend* sistem penjualan UMKM.

Tabel 1 Penelitian Terkait

No	Nama	Judul
(1)	(2)	(3)
1	(Syahroni et al., 2024)	Judul: Uji Keamanan <i>Back End</i> Aplikasi Berbasis <i>Website</i> Menggunakan Metode <i>Black Box Testing</i> Metode: <i>Black Box Testing</i> . Software: <i>Postman</i> , <i>JWT (JSON Web Token)</i> . Hasil Penelitian: Hasil penelitian <i>authentication</i> berjalan baik, karena <i>API</i> tidak dapat diakses tanpa <i>JWT</i> dan akan memberikan 401 <i>unauthorized</i> .
2	(Jati et al., 2025)	Judul: Efisiensi Penggunaan <i>Raw Query</i> dan <i>ORM</i> pada Ekosistem <i>Node.js</i> Metode: <i>Eksperimen komparatif</i> Software: <i>Node.js</i> , <i>Express.js</i> , <i>PostgreSQL</i> , <i>Prisma ORM</i> , <i>node-pg</i> , <i>Grafana K6</i> Hasil Penelitian: Hasil penelitian menunjukkan bahwa terdapat perbedaan performa yang signifikan antara <i>Raw Query</i> dan <i>ORM</i> . Pada operasi <i>read</i> , <i>Raw Query</i> menghasilkan median <i>response time</i> sebesar 3,48 ms, sedangkan <i>ORM</i> 6,26 ms.

3	(Rangga, Gelar Guntara & Azkari, 2023)	Judul: Implementasi dan Pengujian <i>REST API</i> Sistem Reservasi Ruang Rapat dengan Metode <i>Black Box Testing</i> (Studi Kasus: PT Lizzie Parra Kreasi) Metode: <i>Software Development Life Cycle (SDLC)</i>. Software: <i>Node.js, Postman, REST API, Database Management System</i>. Hasil Penelitian: Hasil penelitian pengujian dengan metode <i>black box testing</i>, yang menunjukkan bahwa semua fungsi setiap <i>endpoint API</i> berjalan sesuai dengan output yang diharapkan. Melalui metode <i>HTTP</i>, seluruh <i>endpoint</i> dapat berjalan dengan baik, mulai dari perolehan data dengan metode <i>GET</i>, pembaharuan data dengan metode <i>PUT</i>, penambahan data dengan metode <i>POST</i>, dan penghapusan data dengan metode <i>DELETE</i>.
4	(Muhaimma & Paputangan, 2024)	Judul: Pengembangan <i>Backend Server</i> Berbasis Arsitektur <i>REST API</i> pada Sistem Transfer Dompot Digital Metode: <i>Extreme Programming (XP)</i> Software: <i>NestJS, Node.js, MySQL, Postman, Visual Studio Code, Swagger</i>. Hasil Penelitian: Hasil penelitian berhasil mengimplementasikan pemindahan saldo antar dompet digital. Semua <i>API</i> telah berhasil diimplementasi dan diuji secara berkala mulai kegunaan, <i>error handling</i>, dan lain sebagainya.
5	(Sulaiman et al., 2023)	Judul: Perancangan Aplikasi Todo Menggunakan <i>Node.js</i> dan <i>REST API</i> Metode: <i>Agile</i>. Software: <i>Node.js, Express.js, REST API, Postman</i>. Hasil Penelitian: Hasil penelitian menunjukkan bahwa penerapan teknologi modern seperti <i>node.js</i> dan metodologi <i>agile</i> dalam pengembangan aplikasi dapat menghasilkan solusi yang responsif dan efektif bagi pengguna.

6 (Arief et al., 2025)	Judul: Pengembangan Sistem Informasi Transaksi Penjualan UMKM menggunakan Metode <i>Rapid Application Development</i> Metode: <i>Rapid Application Development (RAD)</i> Software: <i>ExpressJS, EJS (Embedded JavaScript), SQLite, Model-View-Controller (MVC)</i> Hasil Penelitian: Hasil penelitian berhasil mengembangkan sistem informasi transaksi penjualan berbasis <i>web</i> yang sesuai dengan kebutuhan operasional UMKM Uwais Telur. Sistem menyediakan fitur pencatatan transaksi, pengelolaan data pelanggan, monitoring pembayaran, pencetakan nota, manajemen stok, serta pencadangan data ke format Excel. Hasil <i>Alpha Testing</i> menunjukkan seluruh fitur berjalan dengan baik dan sesuai kebutuhan sistem. Hasil <i>Beta Testing</i> juga menunjukkan bahwa pengguna merasa sistem lebih sederhana, ringan, mudah digunakan, serta lebih sesuai dibandingkan aplikasi yang di gunakan sebelumnya.
7 (Naufal et al., 2022)	Judul: Implementasi <i>REST API</i> untuk Fitur Rencana Strategis Program Pada SIMPEDIA Metode: Pengembangan Sistem. Software: <i>Frame Work Phalcon, PHP, Javascript, Postman.</i> Hasil Penelitian: Hasil implementasi <i>REST API</i> berhasil mempermudah pertukaran data antara <i>client</i> dan <i>server</i>, meningkatkan efisiensi sistem, serta menghasilkan seluruh <i>endpoint CRUD</i>

		yang diuji dengan status sukses. <i>Output</i> data menggunakan format <i>JSON</i> sehingga lebih mudah diproses oleh <i>client</i> .
8	(Hasanudin, 2022)	Judul: Rancangan Bangun <i>REST API</i> Aplikasi sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan Metode: <i>SCRUM</i> Software: <i>Node.js, ExpressJS, PostgreSQL, Sequelize (ORM), Postman</i>. Hasil Penelitian: Hasil pengujian merancang dan membangun sistem berbasis <i>REST API</i> dengan arsitektur <i>REST</i> pada sisi <i>backend</i> yang dapat diimplementasikan pada aplikasi <i>website</i> dan <i>mobile (Android)</i> untuk layanan donasi <i>online</i>. Sistem memudahkan proses transaksi donasi, pertukaran data menggunakan format <i>JSON</i>, serta mendukung pengelolaan dan verifikasi donasi secara lebih efisien.
9	(Listyano et al., 2024)	Judul: Optimasi Aplikasi Toko Bangunan Melalui Integrasi <i>Rest API</i> dan <i>WhatsApp Gateway</i> Metode: : Analisis Kebutuhan Sistem. Software: <i>REST API, WhatsApp Gateway, Database SQL, Aplikasi E-commerce</i>. Hasil Penelitian: Hasil Penelitian berhasil mengembangkan sistem berbasis <i>REST API</i> yang mengintegrasikan manajemen stok, pemesanan, pembayaran, dan pelacakan pengiriman. Hasil penelitian menunjukkan peningkatan efisiensi pengelolaan stok, peningkatan konektivitas antara sistem dan konsumen, serta kemudahan pemantauan pengiriman secara <i>real-time</i> sehingga operasional toko menjadi lebih efektif.

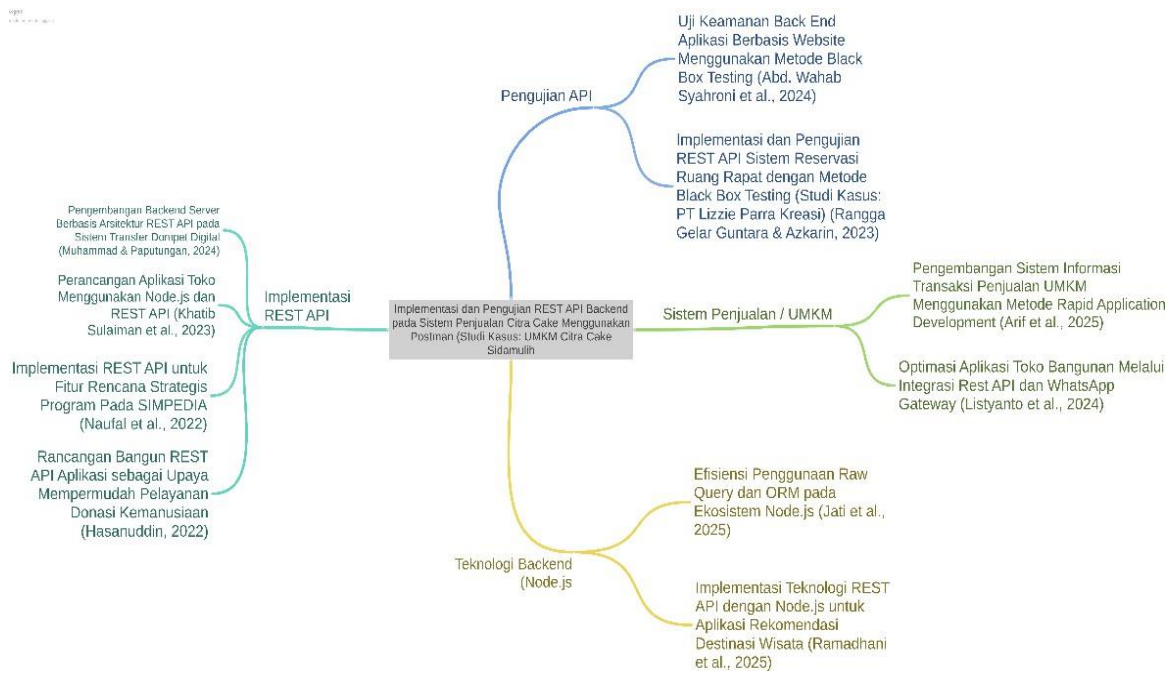
	Judul: Implementasi Teknologi <i>REST API</i> dengan <i>Node.js</i> untuk Aplikasi Rekomendasi Destinasi Wisata
	Metode: Observasi
	Software: <i>Node.js</i> , <i>REST API</i> , <i>Cloud Computing</i> , <i>Google Maps API</i> .
	Hasil Penelitian:
10 (Rama dhani et al., 2025)	Hasil penelitian <i>REST API</i> yang dibangun menggunakan <i>Node.js</i> berhasil menyediakan layanan data destinasi wisata secara cepat dan efektif. Implementasi ini menunjukkan bahwa <i>Node.js</i> memiliki performa yang baik untuk pengembangan <i>backend REST API</i> serta mendukung integrasi dengan berbagai sumber data dan peningkatan skalabilitas sistem.

Penelitian ini memiliki kebaruan dengan menggabungkan implementasi dan pengujian *REST API backend* pada sistem penjualan UMKM Citra Cake Sidamulih menggunakan teknologi modern. Secara objek penelitian, sistem yang dirancang berfokus pada pengelolaan produk, kategori, stok, keranjang belanja, *profile* pengguna, dan pemesanan kue yang mendukung kebutuhan operasional UMKM secara terintegrasi. Fitur tersebut membutuhkan pengelolaan data yang terstruktur dan komunikasi yang baik antara *frontend* dan *backend*.

Kebaruan penelitian juga terletak pada pengujian *endpoint REST API* menggunakan *Postman* untuk memverifikasi fungsi layanan *backend*. Pengujian dilakukan pada *endpoint* autentikasi, profil pengguna, produk, kategori, keranjang, dan pemesanan dengan memeriksa *request*, *response*, *status code*, *response time*, dan *success rate*. Pengujian *Single Track* dilakukan satu kali, sedangkan *Multi Track* dilakukan lima kali pada *endpoint* yang diuji. Hasil pengujian memberikan gambaran struktur mengenai keberhasilan fungsi layanan *backend* pada Sistem Penjualan Citra Cake Sidamulih.

Dengan pengukuran tersebut, penelitian dapat menunjukkan keberhasilan dan karakteristik repons layanan *REST API backend* berdasarkan hasil pengujian *Postman*.

Selain itu, penelitian ini tidak hanya berfokus pada implementasi *REST API backend*, tetapi juga pada evaluasi respons layanan yang dihasilkan melalui pengujian menggunakan *Postman*. Evaluasi dilakukan dengan mengukur *Response Time*, *Status Code*, dan tingkat keberhasilan *endpoint* pada setiap fitur yang tersedia. Hasil tersebut menjadi dasar untuk menilai apakah *endpoint* yang diuji telah memberikan respons sesuai dengan kebutuhan sistem penjualan UMKM Citra Cake Sidamulih. Dengan adanya pengukuran tersebut, penelitian ini dapat memberikan gambaran mengenai kualitas layanan *backend* yang diimplementasikan serta memastikan bahwa setiap *endpoint* mampu memberikan respons yang sesuai dengan kebutuhan sistem penjualan UMKM Citra Cake Sidamulih.



Gambar 1 Mind Map

B. Landasan Teori

1. Sistem Penjualan dan *Backend*

Sistem penjualan berbasis *web* merupakan sistem informasi yang mengelola data produk, pelanggan, stok, pemesanan, dan transaksi secara terintegrasi. Dalam penelitian ini, kebutuhan tersebut dipenuhi melalui pembagian fungsi antara *frontend* dan *backend*. *Backend* menjadi bagian *server-side* yang menangani logika bisnis, validasi, autentikasi, pengelolaan data, dan penyediaan layanan yang dapat diakses oleh *frontend* melalui *API*. Dengan demikian, *backend* menjadi pusat pemrosesan data yang menghubungkan proses bisnis sistem dengan basis data.

2. *REST API* sebagai Arsitektur Layanan *Backend*

REST API digunakan sebagai arsitektur layanan pada *backend* karena menyediakan mekanisme komunikasi berbasis *HTTP* antara *client* dan *server*. *Resource* sistem direpresentasikan melalui *URI/endpoint*, sedangkan operasi terhadap *resource* dilakukan menggunakan metode *HTTP* seperti *GET*, *POST*, *PUT/PATCH*, dan *DELETE*. *Response* dikembalikan dalam format yang dapat diproses *client*, seperti *JSON*, bersama *HTTP Status Code* sebagai penanda hasil pemrosesan. Penelitian empiris mengenai desain *REST* menunjukkan bahwa penggunaan aturan *REST* yang tepat dapat meningkatkan keterpahaman *Web API*[6]. Dalam penelitian ini, konsep tersebut menjadi dasar untuk merancang *endpoint* yang terpisah berdasarkan fungsi autentikasi, profil, produk, kategori, keranjang, dan pesanan. Lingkungan Implementasi *Backend*. Lingkungan implementasi *backend* terdiri atas *Node.js* dan *Express.js* sebagai lingkungan dan *framework server*, *PostgreSQL* sebagai basis data relasional, serta *Prisma ORM* sebagai penghubung antara aplikasi dan basis data. *Node.js* menjalankan kode *JavaScript* pada sisi *server*, *Express.js* mengelola *routing* dan *middleware HTTP*, *PostgreSQL* menyimpan data sistem, sedangkan *Prisma ORM* mempermudah

pengelolaan data melalui model dan skema basis data. Kombinasi tersebut digunakan agar implementasi *endpoint* dapat dilakukan secara terstruktur dan terhubung dengan basis data secara konsisten.

3. Pengujian *REST API* dengan *Postman*

Pengujian *REST API* dilakukan untuk memvalidasi perilaku *endpoint* dari sisi input dan *output* tanpa bergantung pada tampilan *frontend*. *Black Box Testing* sesuai dengan fokus penelitian karena pengujian memeriksa *request*, *response*, *Status Code*, dan keberhasilan fungsi *endpoint*. Penelitian *RESTest* menunjukkan bahwa pengujian *black-box* dapat digunakan untuk menguji *RESTful Web API* berdasarkan spesifikasi layanan[24], sedangkan *RapiTest* menunjukkan bahwa pengujian *RESTful API* dapat diarahkan pada *correctness* dan *reliability* layanan[8]. Dalam penelitian ini, *Postman* digunakan sebagai alat untuk mengirim *request* dan mencatat *Response Time*, *Status Code*, serta *success rate* pada setiap *endpoint*.

4. Lingkungan *Deployment*

Virtual Private Server (VPS) digunakan sebagai lingkungan *deployment backend* agar layanan *REST API* dapat diakses melalui jaringan. Setelah implementasi dan pengujian awal dilakukan, backend ditempatkan pada *VPS* berbasis *Ubuntu* sehingga *endpoint* dapat diuji pada lingkungan *server* yang lebih dekat dengan kondisi operasional sistem.

5. Metode *Agile Development*

Agile Development digunakan sebagai pendekatan pengembangan karena penelitian dilakukan secara bertahap dan memungkinkan perbaikan ketika ditemukan kesalahan pada proses pengujian. Tahapan yang digunakan adalah *plan* untuk menetapkan kebutuhan dan ruang lingkup, *design* untuk merancang arsitektur serta *endpoint*, *develop* untuk mengimplementasikan *backend*, *test* untuk menguji *endpoint* menggunakan *Postman*, *deploy* untuk menempatkan *backend* pada *VPS*, dan *review* untuk memvalidasi hasil. Jika hasil pengujian belum sesuai, proses kembali ke tahap perbaikan implementasi *backend* sebelum pengujian ulang. Pendekatan ini membuat prosedur penelitian dapat dipahami dan direplikasi pada penelitian selanjutnya.



Gambar 2 Metode *Agile Development*[25]