BAB 2

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian mengenai penggunaan REST API pada aplikasi mobile pernah dilakukan oleh Amras dkk. (2023) sebagai solusi percepatan kegiatan masuk dan keluar wilayah parkir kendaraan. Diimplementasikan sebagai aplikasi mobile yang dapat melakukan pembayaran dan pemesanan tiket masuk parkir. Aplikasi menghasilkan kode QR yang dapat discan ketika pengguna masuk dan keluar wilayah parkir.

Penelitian mengenai penggunaan REST API pada aplikasi Web juga pernah dilakukan oleh Serrano dan Oñate (2021) sebagai jawaban dari survey yang dilakukan pada anggota fakultas dan pengembang sistem informasi kampus *Camarines Sur Polytechnic Colleges* di Filipina. Dalam hasil survey tersebut dijelaskan bahwa sistem informasi mahasiswa atau *Student Information and Accounting System (SIAS)*, kurang memenuhi kebutuhan yang ada. Sistem buatan pihak ketiga tersebut hanya bisa melakukan ekspor data dalam format Excel dan CSV. Maka dibuatlah aplikasi dengan REST API untuk dapat mengakses dan mencari data mahasiswa menggunakan alamat surel *(email)* atau ID mahasiswa.

Penelitian mengenai penggunaan REST API pada aplikasi Web juga pernah dilakukan oleh dua dosen UTDI, Pramono dan Yana Javista (2021) untuk melakukan proses presensi kehadiran karyawan. Aplikasi android yang dibuat melakukan autentikasi pengguna dengan teknologi REST API dan format pesan berupa JWT (JSON Web Token) dari layanan Firebase Auth. Setelah pengguna

login ke aplikasi, data yang dibutuhkan untuk melakukan presensi mencakup Android Device ID dan SSID Wi-Fi dan menghasilkan kode QR sebagai bukti kehadiran.

Penelitian mengenai penggunaan REST API pada aplikasi Web juga pernah dilakukan oleh Ferdiansyah dkk. (2022) agar dapat mengakses data AIS (Automatic Identification System). Data berupa informasi kapal di wilayah perairan maritim Indonesia dari satelit LAPAN-A2 dan LAPAN-A3. Aplikasi web Laravel yang dibuat menampilkan data yang bisa diakses dengan REST API untuk menampilkan data nama, posisi, waktu dan tipe kapal yang dinginkan. Autentikasi dan autorisasi keamanan menggunakan teknologi OAuth 2.0 yang terintegrasi dengan framework Laravel.

Penelitian mengenai penggunaan REST API pada aplikasi mobile Android juga pernah dilakukan oleh Wijayanto dkk. (2019) untuk pengembangan Sistem Informasi Usaha Kecil Menengah (UKM) di Kabupaten Purbalingga, Jawa Tengah. Sistem tersebut menggunakan bahasa pemrograman Python sebagai server API dan menggunakan autentikasi layanan dari Google. Data yang ada ditampilkan dalam aplikasi mobile Android seperti data UKM terdaftar, produk yang ditawarkan masing-masing UKM dan lowongan pekerjaan di UKM.

Penelitian mengenai penggunaan REST API pada aplikasi Web juga pernah dilakukan oleh mahasiswa UTDI, Calvyano (2022) dengan membuat sistem peminjaman dan informasi pementasan barongsai di Sekata Barongsai, Madiun, Jawa Timur. Pengguna sistem dapat melakukan login, registrasi, dan melakukan proses peminjaman. Sistem ini dibangun dengan menggunakan framework Laravel dan Vue.js.

Penelitian oleh Kristanto dkk. (2020) yang mengembangkan sistem

Sandbox untuk aplikasi Dhanapala di PT. Semangat Gotong Royong. Penelitian ini bertujuan untuk mengatasi masalah ketergantungan pada fitur dari pihak ketiga dalam proses pengujian aplikasi. Sistem Sandbox yang dikembangkan ini menyerupai karakteristik pihak ketiga dan ditulis menggunakan bahasa pemrograman Go untuk implementasi REST API, serta didukung oleh New Simple Queue (NSQ) untuk mendukung komunikasi konkuren dan mencegah kegagalan transmisi data.

Melihat pustaka yang telah direferensi, penelitian ini mengangkat masalah data transformasi dan konversi dan menampilkannya menggunakan arsitektur REST API dengan framework antarmuka Mithril.js dan Go.

Berikut adalah tabel 2.1 yang berisi penelitian berhubungan dengan REST API yang digunakan oleh penulis sebagai tinjauan pustaka.

Tabel 2.1 Penelitian yang berhubungan dengan REST API.

Peneliti	**Teknologi Terkait**	**Interaksi User**
Amras dkk. (2023)	Mobile App	Pesan dan pembayaran tiket parkir
Serrano dan Oñate (2021)	CodeIgniter	Pencarian dengan email dan ID mahasiswa
Pramono dan Yana Javista (2021)	Android & Firebase	Presensi kehadiran
Ferdiansyah dkk. (2022)	Laravel & OAuth 2.0	Akses data *AIS*
Wijayanto dkk. (2019)	Python, Google Auth & Android	Melihat dan menghapus data
Calvyano (2022, Augustus 29)	Laravel dan VueJS	Registrasi dan peminjaman
Kristanto dkk. (2020)	Go	Reimplementasi Pihak ketiga dalam Sandbox

Tabel 2.1 (Lanjutan)

Peneliti	Teknologi Terkait	Interaksi User
Aldo (2024)	Mithril.js & Go	Transformasi dan Konversi data

2.2 Dasar Teori

2.2.1 HTTP

HTTP (*HyperText Transfer Protocol*) adalah protokol yang digunakan untuk mentransfer data antara klien dan server di World Wide Web (WWW). HTTP adalah protokol yang bersifat permintaan/tanggapan. Klien (biasanya *browser* atau peramban web) mengirimkan permintaan ke server, dan server merespons dengan status, metadata, dan data yang diminta.

HTTP dikembangkan oleh Tim Berners-Lee di CERN (European Organization for Nuclear Research) pada akhir 1980-an dan awal 1990-an. Pengembangan HTTP kemudian dilanjutkan oleh Internet Engineering Task Force (IETF), yang mengeluarkan standar dan RFC terkait untuk memastikan evolusi dan interoperabilitas protokol ini (Nielsen dkk., 1999). Informasi tahun rilis dan fitur utama yang diperkenalkan tiap versi HTTP bisa dilihat pada tabel 2.2.

Tabel 2.2 Versi HTTP.

Versi	Tahun Rilis	Fitur Utama
HTTP/0.9	1991	Versi pertama HTTP, sangat sederhana, hanya mendukung pengembalian halaman HTML. Tidak ada status kode atau header.

Tabel 2.2 (Lanjutan)

Versi	Tahun Rilis	Fitur Utama
HTTP/1.0	1992	Memperkenalkan transmisi berbagai jenis file seperti CSS, video, skrip, dan gambar. Memperkenalkan header HTTP untuk pertukaran informasi tambahan.
HTTP/1.1	1995	Menambahkan kemampuan untuk menggunakan kembali koneksi untuk permintaan selanjutnya, mendukung pipelining, chunked transfer encoding, dan peningkatan pada caching serta pengendalian aliran data.
Pembaruan HTTP/1.1	1999	Peningkatan efisiensi dan keandalan, memperkuat HTTP/1.1 sebagai versi yang paling banyak digunakan hingga saat ini.
HTTP/2	2015	Peningkatan kinerja dengan multiplexing yang memungkinkan banyak permintaan dan respons dalam satu koneksi, kompresi header, dan pengelolaan sumber daya yang lebih efisien.
HTTP/3	Dalam Pengembangan	Menggunakan protokol QUIC berbasis UDP untuk meningkatkan kecepatan dan keandalan koneksi internet, terutama dalam kondisi jaringan yang tidak stabil.

2.2.2 API

API (Application Programming Interface) memungkinkan aplikasi perangkat lunak untuk berinteraksi dan terintegrasi satu sama lain tanpa antarmuka pengguna. API berfungsi sebagai penghubung atau jembatan, memfasilitasi komunikasi antara berbagai sistem perangkat lunak, seringkali melalui teknologi web seperti HTTP. API biasanya dirancang untuk pengembang, memungkinkan mereka membuat aplikasi dengan memberikan akses yang mudah dan efisien ke layanan dan data. API memungkinkan komunikasi antar mesin dan sangat penting

untuk mengintegrasikan dan memperluas sistem perangkat lunak. (*What is an API?* n.d.)

2.2.3 RESTful API

Salah satu pionir web adalah Roy Fielding, yang mengeksplorasi faktor-faktor kesuksesan perangkat lunak internet dan kekurangannya. Dalam disertasi doktoralnya, Fielding merumuskan enam aturan yang disebutnya sebagai REpresentational State Transfer (REST). Dia berpendapat bahwa jika suatu arsitektur memenuhi enam aturan ini, maka arsitektur tersebut akan memiliki sifat-sifat yang diinginkan seperti skalabilitas, keterpisahan, dan kesederhanaan, yang sangat penting dalam sistem berukuran internet, di mana komputer perlu berkomunikasi lintas batas organisasi. (Fielding, 2000)

Aturan formal untuk REST adalah:

1. **Client Server:**

   Klien dan server dipisahkan melalui antarmuka yang jelas dan terdefinisi dengan baik.
2. **Stateless:**

   Setiap permintaan dari klien ke server harus bersifat stateless, artinya server tidak menyimpan informasi tentang status sesi atau konteks dari permintaan sebelumnya.
3. **Cache:**

   Respons dari server harus menyertakan informasi tentang *cacheability*.
4. **Uniform Interface:**

   REST API menerapkan antarmuka yang seragam untuk interaksi antara klien dan server.
5. **Layered System:**

Arsitektur REST memungkinkan penggunaan sistem berlapis.

6. **Code on Demand (opsional):**

   Server memiliki kemampuan untuk memperluas atau menyesuaikan fungsionalitas klien.

Fielding menggunakan aturan-aturan ini sebagai daftar periksa untuk mengevaluasi standar web baru, dengan tujuan mengidentifikasi desain yang buruk sebelum diterapkan pada jutaan server web. Dia berhasil menggunakan aturan-aturan ini untuk mengevaluasi standar web baru seperti HTTP 1.1 dan sintaks URI, di mana dia berperan sebagai penulis utama. Kedua standar ini telah bertahan dalam ujian waktu, meskipun menghadapi tekanan besar sebagai protokol penting di web yang digunakan oleh miliaran orang setiap hari.

Konsep sentral dalam REST adalah konsep sumber daya (resource), yang merupakan struktur data abstrak. REST API mengekspos dan memanipulasi sumber daya ini. Nama API umumnya tidak dapat dibedakan dari nama sumber daya yang dapat dimanipulasi oleh API tersebut. Selain itu, URL dari API adalah URL dari sumber daya.

Sumber daya dalam REST hampir seperti objek dalam paradigma pemrograman berorientasi objek. Namun, metode yang digunakan dalam REST terbatas pada sekumpulan metode HTTP yang disebut sebagai antarmuka sumber daya yang seragam (uniform resource interface). Metode HTTP ini mencakup operasi CRUD (create, read, update, delete) yang mudah dipetakan ke metode HTTP seperti POST, GET, PUT, dan DELETE. Penggunaan tiap metode dapat dilihat pada tabel 2.3.

Tabel 2.3 Penggunaan HTTP Method dalam REST API.

HTTP Method	Penggunaan
GET	Metode GET digunakan untuk mengambil informasi apa pun yang ditentukan di URI. Ini adalah metode yang sama yang digunakan saat membuka halaman web di browser. Server seharusnya tidak melakukan apa pun selain untuk mengambil URI yang diminta.
POST	Membuat sumber daya baru di bawah sumber daya yang ditentukan dalam URI. Metode ini tidak aman, yang artinya mempunyai efek samping, dan tidak idempoten, yang mana artinya jika membuat permintaan yang sama dua kali, akan mendapat dua hasilnya berbeda.
PUT	Metode HTTP PUT menggantikan sumber daya yang ditentukan dalam URI dengan representasi sumber daya baru di badan permintaan. PUT penuh ganti, dan bukan pembaruan sebagian. Artinya, mirip dengan menimpaa file lengkap, daripada memperbarui satu kolom database. PUT bersifat idempoten, artinya jika dijalankan dua kali maka hasilnya akan sama seperti jika menjalankannya sekali.
DELETE	Menghapus sumber daya yang ditentukan dalam URI.

Keuntungan menggunakan REST termasuk skalabilitas sistem yang baik karena sifat stateless dan independen dari permintaan, kinerja yang baik karena kemampuan caching bawaan dalam HTTP, dukungan untuk penanganan berbagai jenis konten, kesederhanaan dalam pembuatan API baru, dan kemampuan eksplorasi dan penemuan API yang baik.

2.2.4 JSON (JavaScript Object Notation)

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca oleh manusia, dan mudah diproses oleh mesin. JSON berasal dari JavaScript tetapi bersifat independen terhadap bahasa, sehingga dapat digunakan dengan berbagai bahasa pemrograman seperti C, C++, Java, dan

Python. JSON dibuat oleh Douglas Crockford didasarkan pada subset dari Standar Bahasa Pemrograman JavaScript ECMA-262 Edisi ke-3 - Desember 1999, dan juga diatur dalam ECMA-404 sebagai sintaksis pertukaran data JSON edisi kedua - Desember 2017. (Bray, 2014)

JSON dibangun berdasarkan dua struktur utama: pasangan nama/nilai (objek) dan daftar nilai yang terurut (array). Objek adalah kumpulan pasangan nama/nilai yang tidak terurut, sementara array adalah kumpulan nilai yang terurut. Nilai JSON dapat berupa string, angka, boolean, null, objek, atau array, dan struktur-struktur ini dapat disarangkan. File JSON menggunakan ekstensi .json dan dapat digunakan independen terhadap bahasa pemrograman yang digunakan. (Crockford, 2006)

2.2.5 Go

Go adalah bahasa pemrograman umum yang didesain oleh Google dan awalnya merupakan jawaban atas kebutuhan internal Google. Awalnya dibuat oleh tiga orang yaitu Robert Pike, Ken Thompson dan Robert Griesemer. Rilis ke publik pada tanggal 10 November 2009 dengan dibukanya situs https://golang.org. Go versi 1 rilis pada bulan maret tahun 2012 dengan jaminan kesesuaian dan masih menepati janji tersebut hingga versi 1.22 saat penulisan penelitian ini. (*Go 1.22 Release Notes - The Go Programming Language* 2024)

Bahasa pemrograman Go (Golang) memiliki berbagai karakteristik yang signifikan. Sebagai bahasa open-source, Go memungkinkan siapa pun untuk mengunduh, mengembangkan, dan memperbaiki kode secara bebas. Salah satu fitur utamanya adalah dukungan konkurensi yang sederhana dan mudah dilacak, yang memungkinkan pengembang untuk mengeksekusi permintaan lebih cepat dan efisien melalui penggunaan goroutine dan konkurensi berbasis saluran. Selain itu,

Go memiliki kemampuan pengujian yang kuat, memungkinkan pengembang untuk membuat unit tes, memahami cakupan kode, serta melakukan tes benchmark. Kemampuan *garbage collection* menghilangkan kebutuhan untuk mengelola memori secara manual, sehingga meningkatkan efisiensi pengembangan. Implementasi interface secara implisit dalam Go memungkinkan keamanan tipe dan pemisahan komponen sambil tetap mempertahankan fleksibilitas. Sebagai bahasa dengan pengetikan statis, Go memastikan kompilasi kode yang akurat dan penanganan konversi tipe yang baik. Penanganan kesalahan dalam Go dilakukan dengan mengembalikan nilai dari tipe error sebagai nilai akhir dari fungsi. Go menggunakan struct sebagai pengganti konsep kelas dan objek seperti bahasa berorientasi objek lainnya. Go juga memiliki l*standard library* yang besar dan serangkaian *tools* yang kuat, seperti Gofmt, Gorun, Goget, dan Godoc, yang memudahkan proses pengembangan tanpa ketergantungan pada paket pihak ketiga. (*Effective Go - The Go Programming Language* n.d.)

2.2.6 Mithril.js

Mithril.js adalah JavaScript *framework* ringan untuk membangun aplikasi satu halaman (SPA). Mithril.js memiliki fitur mesin DOM virtual yang mengoptimalkan manipulasi DOM dengan hanya memperbarui apa yang telah berubah, meminimalkan perenderan ulang yang tidak perlu, dan meningkatkan respons aplikasi. Pendekatan ini memastikan bahwa aplikasi satu halaman (SPA) yang dibangun dengan Mithril.js mempertahankan interaksi pengguna yang cepat dan pemanfaatan sumber daya yang efisien. (*Mithril.js* n.d.)

Mithril.js mengadopsi struktur reaktif berbasis komponen yang mirip dengan kerangka kerja seperti React dan Vue.js. Hal ini memungkinkan pengembang untuk membuat komponen UI modular dan dapat digunakan kembali

yang secara otomatis melakukan sinkronisasi dengan perubahan data atau status aplikasi. Dengan mempromosikan gaya pemrograman deklaratif, Mithril memfasilitasi pembuatan antarmuka pengguna canggih yang terdiri dari elemen-elemen yang saling berhubungan dan dapat diperbarui sendiri. Dengan ukuran sekitar 8 KB saat terkompres, Mithril.js sangat cocok untuk proyek yang membutuhkan basis kode yang relatif kecil dan waktu pemuatan yang cepat, sehingga menguntungkan untuk pembuatan dan pengembangan web modern.

2.2.7 Webpack

Webpack adalah bundler yang mengemas banyak file, biasanya JavaScript dan aset web lainnya seperti HTML, CSS, dan gambar, ke dalam satu atau beberapa file. Webpack ditulis dalam bahasa JavaScript dan dirancang untuk digunakan dengan Node.js. Program ini dikembangkan oleh Tobias Koppers dan pertama kali dirilis pada 10 Maret 2012. Tujuan utama dari Webpack adalah untuk menggabungkan file JavaScript agar dapat digunakan di browser. Namun, Webpack juga dapat menangani berbagai jenis file melalui sistem plugin dan loader yang ekstensif. Webpack mendukung fitur-fitur seperti pemisahan kode, penggantian modul panas (*hot reload*), dan *tree shaking*, menjadikannya program yang sangat serbaguna dan fleksibel untuk pengembangan web. (*Concepts* n.d.)

Beberapa alternatif kontemporer untuk Webpack yang muncul dengan fitur dan keunggulan yang unik antara lain Parcel; Rollup; Vite; Snowpack dan esbuild. Parcel, yang dirilis pada November 2017, dikenal dengan pendekatan tanpa konfigurasi dan waktu build yang lebih cepat. Rollup, yang dirilis pada 27 Oktober 2015, berfokus pada modul ES6 dan *tree shaking*. Vite, yang dikembangkan oleh Evan You (pembuat Vue.js) dan dirilis pada 20 April 2020, menyediakan alat pengembangan frontend generasi baru yang dioptimalkan untuk modul ES (Ecma

Script). Snowpack, yang diluncurkan pada 13 Januari 2020, menawarkan lingkungan pengembangan instan dan pendekatan pengembangan yang tidak terikat (*unbundled development*). Terakhir, esbuild, yang dirilis pada Maret 2020, adalah bundler dan minifier yang sangat cepat, ditulis dalam bahasa Go, dan tidak memerlukan konfigurasi apa pun.