

LAPORAN
DESAIN DAN PERANCANGAN REST API
PROGRAM PEMROGRAMAN BERBASIS PLATFORM



Disusun Oleh:

Nisya Lathifah	20240040087
Rizki Kurniawan	20240040113
Muhammad Luthfi Abdul Lathif	20240040088

PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS NUSA PUTRA
2026

1. PENDAHULUAN

Perkembangan teknologi informasi pada era digital saat ini mengalami peningkatan yang sangat pesat dan berpengaruh pada berbagai bidang kehidupan, termasuk dalam pengelolaan fasilitas umum. Salah satu fasilitas yang memerlukan sistem pengelolaan yang baik adalah area parkir kendaraan. Pengelolaan parkir yang masih dilakukan secara manual sering kali menimbulkan berbagai permasalahan, seperti kesalahan pencatatan, kehilangan data, kesulitan dalam memantau jumlah kendaraan yang sedang parkir, serta kurangnya transparansi dalam perhitungan biaya parkir.

Seiring dengan meningkatnya jumlah kendaraan bermotor, baik roda dua maupun roda empat, kebutuhan akan sistem parkir yang terkomputerisasi menjadi semakin penting. Sistem parkir yang terintegrasi dapat membantu petugas parkir dalam mencatat data kendaraan masuk dan keluar secara lebih akurat, serta memudahkan pihak pengelola dalam melakukan monitoring dan evaluasi operasional parkir.

Salah satu solusi yang dapat diterapkan adalah dengan membangun REST API (Representational State Transfer Application Programming Interface) sebagai backend sistem parkir. REST API memungkinkan pertukaran data antara sistem backend dan aplikasi frontend secara efisien, fleksibel, dan terstruktur. Dengan menggunakan REST API, data parkir dapat diakses oleh berbagai platform, seperti aplikasi web maupun aplikasi mobile.

Pada tugas ini, dikembangkan sebuah REST API Sistem Parkir Kendaraan menggunakan Node.js dan Express.js sebagai server-side framework serta MySQL sebagai basis data. Sistem ini dirancang untuk mengelola data pengguna, kendaraan, aktivitas parkir, serta pembayaran parkir. Selain itu, sistem ini juga dilengkapi dengan mekanisme keamanan menggunakan JSON Web Token (JWT) untuk memastikan bahwa hanya pengguna yang memiliki hak akses yang dapat menggunakan fitur tertentu.

Dengan adanya REST API ini, diharapkan proses pengelolaan parkir dapat menjadi lebih efektif, efisien, dan terorganisir. Sistem ini juga dapat dikembangkan lebih lanjut dengan mengintegrasikan API publik lainnya, seperti API pembayaran digital atau API notifikasi, sesuai dengan kebutuhan di masa depan.

2. STUDI KASUS

Studi kasus yang digunakan dalam perancangan dan implementasi REST API pada tugas ini adalah Sistem Parkir Kendaraan yang diterapkan pada sebuah area parkir umum, seperti gedung perkantoran, pusat perbelanjaan, atau kampus. Area parkir tersebut melayani berbagai jenis kendaraan, baik kendaraan roda dua maupun roda empat, dengan sistem pencatatan parkir yang membutuhkan keakuratan dan kecepatan dalam pengelolaannya.

Pada kondisi awal, sistem parkir masih dilakukan secara manual, yaitu petugas mencatat data kendaraan masuk dan keluar pada buku atau karcis parkir. Metode ini memiliki beberapa kelemahan, antara lain:

1. Data parkir mudah hilang atau rusak.
2. Sulit melakukan pencarian data parkir tertentu.
3. Tidak tersedia laporan parkir secara real-time.
4. Proses perhitungan biaya parkir kurang akurat.
5. Sulit memantau jumlah kendaraan yang sedang parkir.

Untuk mengatasi permasalahan tersebut, dirancang sebuah sistem parkir berbasis REST API yang mampu mengelola seluruh aktivitas parkir secara digital. Sistem ini memiliki beberapa aktor utama, yaitu petugas parkir sebagai pengguna sistem dan kendaraan sebagai objek yang dikelola. Petugas parkir bertanggung jawab untuk mencatat kendaraan yang masuk dan keluar area parkir serta melakukan pencatatan pembayaran.

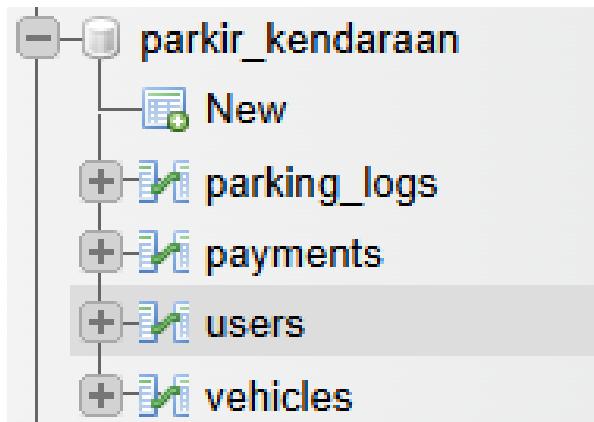
Sistem parkir yang dibangun memiliki beberapa fitur utama, antara lain:

- Pengelolaan data pengguna (petugas parkir)
- Pengelolaan data kendaraan
- Pencatatan kendaraan masuk dan keluar
- Pencatatan status parkir (aktif atau selesai)
- Pengelolaan pembayaran parkir
- Penyediaan data monitoring dan laporan parkir

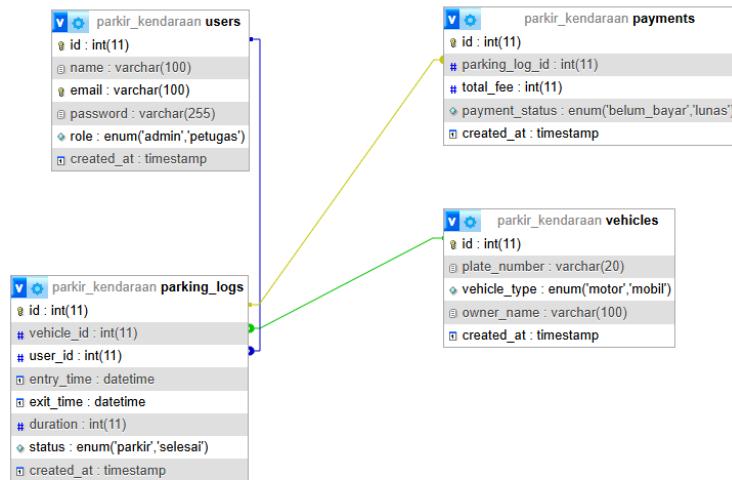
REST API digunakan sebagai penghubung antara sistem backend dengan aplikasi frontend. Setiap aktivitas parkir direpresentasikan dalam bentuk endpoint REST yang dapat diakses menggunakan metode HTTP seperti GET, POST, PUT, dan DELETE. Dengan pendekatan ini, sistem menjadi lebih modular, mudah diuji, dan mudah dikembangkan.

Melalui penerapan REST API Sistem Parkir Kendaraan ini, diharapkan proses pengelolaan parkir dapat dilakukan secara lebih profesional, transparan, dan terintegrasi. Sistem ini juga memberikan dasar yang kuat untuk pengembangan fitur lanjutan, seperti integrasi pembayaran digital, sistem notifikasi otomatis, serta analisis data parkir untuk pengambilan keputusan manajemen.

3. PERANCANGAN DATA BASE



4. RELASI ANTAR TABLE



Berdasarkan gambar Entity Relationship Diagram (ERD) di atas, sistem parkir kendaraan memiliki empat tabel utama yang saling berelasi, yaitu users, vehicles, parking_logs, dan payments. Relasi antar tabel dirancang untuk menjaga konsistensi data dan mendukung proses bisnis sistem parkir.

1. Relasi Tabel Users dengan Parking_logs

Tabel users berelasi dengan tabel parking_logs melalui kolom user_id pada tabel parking_logs yang mengacu pada id pada tabel users.

Jenis

relasi:

→ One to Many (1:N)

Penjelasan:

Satu pengguna (petugas parkir) dapat menangani banyak aktivitas parkir kendaraan. Namun, satu data parkir hanya dicatat oleh satu petugas parkir. Oleh karena itu, setiap data pada tabel parking_logs harus memiliki satu user_id yang valid dari tabel users.

Fungsi relasi:

- Mengetahui petugas yang mencatat kendaraan masuk dan keluar
- Memudahkan pelacakan tanggung jawab petugas parkir

2. Relasi Tabel Vehicles dengan Parking_logs

Tabel vehicles berelasi dengan tabel parking_logs melalui kolom vehicle_id pada tabel parking_logs yang mengacu pada id pada tabel vehicles.

Jenis relasi:



Penjelasan:

Satu kendaraan dapat memiliki beberapa riwayat parkir pada waktu yang berbeda. Namun, satu catatan parkir hanya terkait dengan satu kendaraan. Relasi ini memungkinkan sistem untuk menyimpan histori parkir kendaraan secara lengkap.

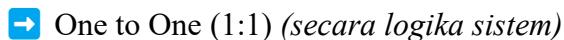
Fungsi relasi:

- Menyimpan riwayat parkir setiap kendaraan
- Memudahkan pencarian data parkir berdasarkan kendaraan

3. Relasi Tabel Parking_logs dengan Payments

Tabel parking_logs berelasi dengan tabel payments melalui kolom parking_log_id pada tabel payments yang mengacu pada id pada tabel parking_logs.

Jenis relasi:



Penjelasan:

Setiap aktivitas parkir yang telah selesai akan menghasilkan satu transaksi pembayaran. Oleh karena itu, satu data pada tabel parking_logs hanya memiliki satu data pembayaran pada tabel payments.

Fungsi relasi:

- Menghubungkan data parkir dengan data pembayaran
- Menjamin bahwa setiap pembayaran terkait dengan aktivitas parkir yang valid

4. Peran Tabel Parking_logs sebagai Tabel Utama

Tabel parking_logs berperan sebagai tabel pusat (tabel transaksi) yang menghubungkan:

- Data pengguna (users)
- Data kendaraan (vehicles)
- Data pembayaran (payments)

Tabel ini menyimpan informasi penting seperti waktu masuk, waktu keluar, durasi parkir, dan status parkir.

Kesimpulan Relasi Database

Relasi antar tabel pada sistem parkir kendaraan dirancang secara terstruktur untuk:

1. Menjaga integritas data antar tabel
2. Menghindari duplikasi data
3. Memudahkan proses pencatatan, pencarian, dan pelaporan data
4. Mendukung proses bisnis sistem parkir secara menyeluruh

Dengan desain relasi seperti ini, sistem parkir kendaraan dapat berjalan secara efisien, konsisten, dan mudah dikembangkan di masa mendatang.

```
C:\Users\LENOVO>mkdir parkir_kendaraan
C:\Users\LENOVO>cd parkir_kendaraan
C:\Users\LENOVO\parkir_kendaraan>npm init -y
Wrote to C:\Users\LENOVO\parkir_kendaraan\package.json:

{
  "name": "parkir_kendaraan",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

C:\Users\LENOVO\parkir_kendaraan>npm install express mysql2 sequelize jsonwebtoken bcryptjs body-parser dotenv
added 109 packages, and audited 110 packages in 51s
26 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
C:\Users\LENOVO\parkir_kendaraan>code .
```

Pada tahap awal pengembangan REST API Sistem Parkir Kendaraan, digunakan Command Prompt (CMD) untuk membuat struktur proyek dan menginstal kebutuhan awal aplikasi. Penggunaan CMD bertujuan untuk mempercepat proses pembuatan folder, pengelolaan dependensi, serta menjalankan server aplikasi.

1. Pembuatan Folder Proyek

Perintah berikut digunakan untuk membuat folder utama proyek:

```
mkdir parkir_kendaraan
```

Penjelasan:

Perintah `mkdir` (make directory) digunakan untuk membuat sebuah folder baru dengan nama `parkir_kendaraan`. Folder ini berfungsi sebagai direktori utama (root folder) dari proyek REST API Sistem Parkir Kendaraan.

2. Masuk ke Folder Proyek

```
cd parkir_kendaraan
```

Penjelasan:

Perintah cd (change directory) digunakan untuk berpindah ke folder parkir_kendaraan agar seluruh proses selanjutnya dilakukan di dalam folder proyek.

3. Inisialisasi Proyek Node.js

```
npm init -y
```

Penjelasan:

Perintah ini digunakan untuk membuat file package.json secara otomatis. File ini berisi informasi proyek serta daftar dependensi yang digunakan dalam pengembangan REST API.

4. Instalasi Library yang Dibutuhkan

```
npm install express mysql2 dotenv jsonwebtoken bcryptjs
```

Penjelasan:

Perintah ini digunakan untuk menginstal beberapa library utama, yaitu:

- express → framework backend
- mysql2 → koneksi ke database MySQL
- dotenv → mengelola environment variable
- jsonwebtoken → autentikasi JWT
- bcryptjs → enkripsi password

5. Instalasi Nodemon (Opsional)

```
npm install -D nodemon
```

Penjelasan:

Nodemon digunakan untuk menjalankan server secara otomatis setiap kali terjadi perubahan kode, sehingga memudahkan proses pengembangan.

6. Pembuatan Struktur Folder Proyek

```
mkdir config controllers models routes middleware
```

Penjelasan:

Perintah ini digunakan untuk membuat beberapa folder utama sesuai dengan arsitektur MVC, yaitu:

- config → konfigurasi database
- controllers → logika bisnis
- models → query database
- routes → endpoint REST API
- middleware → autentikasi dan keamanan

7. Pembuatan File Utama Server

type nul > server.js

Penjelasan:

Perintah ini digunakan untuk membuat file server.js yang berfungsi sebagai file utama untuk menjalankan server Express.

8. Pembuatan File Environment

type nul > .env

Penjelasan:

File .env digunakan untuk menyimpan variabel sensitif seperti port server dan secret key JWT agar tidak ditulis langsung di dalam kode.

9. Menjalankan Server

node server.js

atau jika menggunakan nodemon:

npx nodemon server.js

Penjelasan:

Perintah ini digunakan untuk menjalankan server REST API. Jika server berhasil dijalankan, maka akan muncul pesan bahwa server berjalan pada port tertentu.

5. KESIMPULAN PENGGUNAAN CMD

Penggunaan Command Line (CMD) sangat membantu dalam proses pengembangan REST API karena:

1. Mempercepat pembuatan struktur proyek
2. Memudahkan instalasi library
3. Mengelola dan menjalankan server aplikasi
4. Membuat proyek lebih terstruktur dan rapi

- Folder Struktur

```
✓ PARKIR_KENDARAAN
  ✓ config
    JS db.js
  ✓ controllers
    JS authController.js
    JS parkingController.js
    JS paymentController.js
    JS userController.js
    JS vehicleController.js
  > middleware
  ✓ models
    JS index.js
  > node_modules
  ✓ routes
    JS authRoutes.js
    JS parkingRoutes.js
    JS paymentRoutes.js
    JS userRoutes.js
    JS vehicleRoutes.js
  ⚙ .env
  { } package-lock.json
  { } package.json
  JS server.js
```

POST > login

HTTP <http://localhost:3000/auth/login>

POST [▼](#) http://localhost:3000/auth/login

Docs Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL [JSON](#) [▼](#)

```
1 {  
2   "email": "admin@mail.com",  
3   "password": "123456"  
4 }  
5
```

Body Cookies Headers (7) Test Results | [⌚](#)

{ } JSON [▼](#) [▶ Preview](#) [☒ Visualize](#) [▼](#)

```
1 {  
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJpZCI6Miwiexp9sZSI6ImFkbWluIiwiaWF0IjoxNzY5MzU2NTUwLCJleHAiOjE3NjkzNjAxNTB9.  
5Cisemvc_dhNxT8SU0mEM-VKF-KL8aPvIyNCwA2WBgg"  
3 }
```

GET > vehicles

GET [▼](#) http://localhost:3000/vehicles

Docs Params Authorization Headers (7) Body Scripts Settings

Headers [⌚](#) 6 hidden

	Key	Value
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9. eyJpZCI6Miwiexp9sZSI6ImFkbWluIiwiaWF0IjoxNzY5MzU2NTUwLCJleHAiOjE3NjkzNjAxNTB9. 5Cisemvc_dhNxT8SU0mEM-VKF-KL8aPvIyNCwA2WBgg"
	Key	Value

Body Cookies Headers (7) Test Results | [⌚](#)

{ } JSON [▼](#) [▶ Preview](#) [☒ Visualize](#) [▼](#)

```
1 [  
2   {  
3     "id": 1,  
4     "plate_number": "B 1234 ABC",  
5     "vehicle_type": "motor",  
6     "owner_name": "Andi",  
7     "created_at": "2026-01-25T15:09:31.000Z"  
8   }  
9 ]
```

POST

PARKING

ENTRY

POST http://localhost:3000/parking/entry

☰ Docs Params Auth Headers (9) Body • Scripts Settings

raw ▾ JSON ▾ S

```
1  {
2    "vehicle_id": 1,
3    "user_id": 1
4  }
5
```

Body ▾ ⏱

200 OK • 106 ms • ↗

{ } JSON ▾ ▷ Preview 🖥 Visualize ▾



```
1  {
2    "message": "Kendaraan masuk"
3  }
```

GET Headers (7) Body Scripts Settings

Headers (6 hidden)

Key	Value
Authorization	Bearer eyJhbGciOiJIUzI1...
Key	Value
Key	Description

Body 200 OK • 83 ms

{ } JSON ▾ ▶ Preview Visualize |

```
1 [  
2   {  
3     "id": 1,  
4     "vehicle_id": 1,  
5     "user_id": 2,  
6     "entry_time": "2026-01-25T16:13:33.000Z",  
7     "exit_time": null,  
8     "duration": null,  
9     "status": "parkir",  
10    "created_at": "2026-01-25T16:13:33.000Z"  
11  }  
12 ]
```

POST Headers (9) Body Scripts Settings

raw ▾ JSON ▾ S

```
1 {  
2   "vehicle_id": 1,  
3   "user_id": 1  
4 }  
5
```

Body 200 OK • 106 ms • 2

{ } JSON ▾ ▶ Preview Visualize |

```
1 {  
2   "message": "Kendaraan masuk"  
3 }
```

POST Send

Docs Params Auth Headers (8) Body Scripts Settings Cookies

raw Schema Beautify

```
1 {  
2   "email": "petugas@mail.com",  
3   "password": "123456"  
4 }  
5
```

Body 31 ms 408 B |

{ } JSON

```
1 {  
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJpZCI6NCwicm9sZSI6InBldHVnYXMiLCJpYXQiOjE3NjkzNTg1ODcsImV  
4cCI6MTc2OTM2MjE4N30.  
_krWKBZE4cdluAj86QNcGui_bbdUWSVvhRqd103FFMI"  
3 }
```

GET <http://localhost:3000/users>

☰ Docs Params Auth Headers (7) Body Scripts Settings

Headers (6 hidden)

	Key	Value

Body [🕒](#) 200 OK 2

{ } JSON ▾ ▶ Preview 🖼 Visualize | ▾

```
1 [  
2   {  
3     "id": 1,  
4     "name": "Admin Parkir",  
5     "email": "admin@parkir.com",  
6     "role": "admin"  
7   },  
8   {  
9     "id": 2,  
10    "name": "Admin Parkir",  
11    "email": "admin@mail.com",  
12    "role": "admin"  
13  },  
14  {  
15    "id": 3,  
16    "name": "Nisya lathifah",  
17    "email": "nisya@mail.com",
```

POST VEHICLES

The screenshot shows a POST request to `http://localhost:3000/vehicles`. The request body is a JSON object:

```
1 {  
2   "plate_number": "B 1234 CD",  
3   "type": "Mobil"  
4 }  
5
```

The response status is `200 OK` with a size of `114`. The response body is:

```
1 {  
2   "message": "Vehicle added"  
3 }
```