

**PEMROGRAMAN API
REST API MENGGUNAKAN NEXT.JS**

Dosen Pengampu:
Saiful Nur Budiman, M.Kom



Nama Kelompok:	
Rizqi Harisma Uchrowi	23104410001
Dimas Akbar Maulana	23104410056
Angga Novantara	24104410044
Andrecaesar Setiawan	23104410041
Intan Lusiana	23104410106
Enjing Dwi Retno Wulan	23104410049

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ISLAM BALITAR**

2025

Pendahuluan

Proyek ini merupakan pengembangan API backend menggunakan framework Next.js versi 16.0.5. Nama proyek adalah "project-api-nextjs" dengan versi 0.1.0. Tujuan proyek ini adalah membangun sebuah sistem API yang aman, dengan fitur autentikasi dan otorisasi, serta integrasi database menggunakan Prisma. Proyek ini dirancang untuk mendukung manajemen data seperti students dan users, dengan penekanan pada keamanan akses berdasarkan role pengguna (misalnya, role ADMIN).

Proyek ini dibuat sebagai bagian dari Ujian Akhir Semester (UAS) kelompok, yang menunjukkan penerapan konsep-konsep seperti server-side rendering, middleware untuk keamanan, dan konfigurasi lingkungan development. Semua file konfigurasi dan dependencies telah disesuaikan untuk mendukung pengembangan yang efisien dan skalabel.

Deskripsi Proyek

Proyek ini adalah aplikasi API berbasis Next.js yang fokus pada backend. Fitur utama meliputi:

1. Autentikasi dan Otorisasi: Menggunakan JWT (JSON Web Token) untuk verifikasi token pengguna.
2. Proteksi Route API: Middleware memastikan hanya pengguna terautentikasi yang dapat mengakses route tertentu, dan role ADMIN diperlukan untuk akses ke /api/users.
3. Integrasi Database: Menggunakan Prisma sebagai ORM untuk berinteraksi dengan database (misalnya PostgreSQL via adapter PG).
4. Rate Limiting: Untuk mencegah abuse API.
5. Validasi Data: Menggunakan Zod untuk schema validation.

Proyek ini private (tidak untuk publikasi langsung) dan dioptimalkan untuk environment development, build, dan production.

Teknologi dan Dependencies

Berdasarkan file package.json dan package-lock.json, berikut adalah daftar dependencies utama yang digunakan:

Dependencies Utama (Runtime):

Nama Package	Versi	Deskripsi
@prisma/adapter-pg	^7.1.0	Adapter Prisma untuk PostgreSQL.
@prisma/client	^7.0.1	Client untuk Prisma ORM, digunakan untuk query database.
bcryptjs	^3.0.3	Library untuk hashing password (keamanan autentikasi).
jose	^6.1.2	Library untuk JWT verification (menggantikan jsonwebtoken untuk performa lebih baik).
next	16.0.5	Framework utama untuk building API dan aplikasi web.
rate-limiter-flexible	^9.0.0	Untuk membatasi request API guna mencegah DDoS atau abuse.
react	19.2.0	Library UI (meskipun fokus backend, mungkin untuk komponen hybrid).
react-dom	19.2.0	DOM renderer untuk React.
zod	^4.1.13	Library untuk schema validation dan parsing data.

DevDependencies (Development):

Nama Package	Versi	Deskripsi
@tailwindcss/postcss	^4	Plugin PostCSS untuk Tailwind CSS.
@types/node	^20	Type definitions untuk Node.js.
@types/react	^19	Type definitions untuk React.
@types/react-dom	^19	Type definitions untuk React DOM.
baseline-browser-mapping	^2.8.32	Mapping untuk compatibility browser.
dotenv	^17.2.3	Untuk loading environment variables dari .env.
eslint	^9	Tool untuk linting code.
eslint-config-next	16.0.5	Konfigurasi ESLint khusus Next.js.
prisma	^7.0.1	CLI Prisma untuk migrasi dan generate schema.
tailwindcss	^4	Framework CSS utility-first.

typescript	^5	Bahasa superset JavaScript untuk type safety.
------------	----	---

Total packages terinstal: Lebih dari 100 (berdasarkan package-lock.json), dengan lockfile version 3 untuk konsistensi instalasi.

Struktur dan Konfigurasi Proyek

Proyek ini memiliki konfigurasi standar Next.js dengan penambahan custom untuk keamanan dan database. Berikut ringkasan file utama:

1. middleware.js:

1. Fungsi middleware untuk memverifikasi token JWT pada route API tertentu (/api/students/* dan /api/users/*).
2. Menggunakan jose untuk verifikasi token dengan secret dari environment variable JWT_SECRET.
3. Menambahkan header x-user-role dari payload token.
4. Memblokir akses jika token invalid atau role bukan ADMIN untuk /api/users.
5. Error handling: Mengembalikan response JSON dengan status 401 atau 403 jika gagal.

2. eslint.config.mjs:

1. Konfigurasi ESLint menggunakan eslint-config-next untuk core web vitals dan TypeScript.
2. Override ignores untuk file seperti .next/, out/, dll., agar linting fokus pada source code.

3. .env.example:

1. Contoh environment variables: DATABASE_URL untuk koneksi database dan JWT_SECRET untuk JWT.

4. next.config.ts:

1. File konfigurasi Next.js dasar (kosong, siap untuk custom seperti rewrites atau images).

5. .gitignore:

1. Mengabaikan file seperti node_modules, .env, .next, dll., untuk menjaga repo bersih dari file sensitif atau generated.

6. **README.md:**

1. Instruksi dasar untuk menjalankan proyek: npm run dev untuk development server di localhost:3000.
2. Referensi ke dokumentasi Next.js dan deployment di Vercel.

7. **postcss.config.mjs:**

1. Konfigurasi PostCSS dengan plugin Tailwind CSS untuk processing CSS.

8. **tsconfig.json:**

1. Konfigurasi TypeScript: Target ES2017, include JSX, paths alias (@/* untuk root), dan exclude node_modules.
2. Mendukung incremental compilation untuk performa.

9. **prisma.config.ts:**

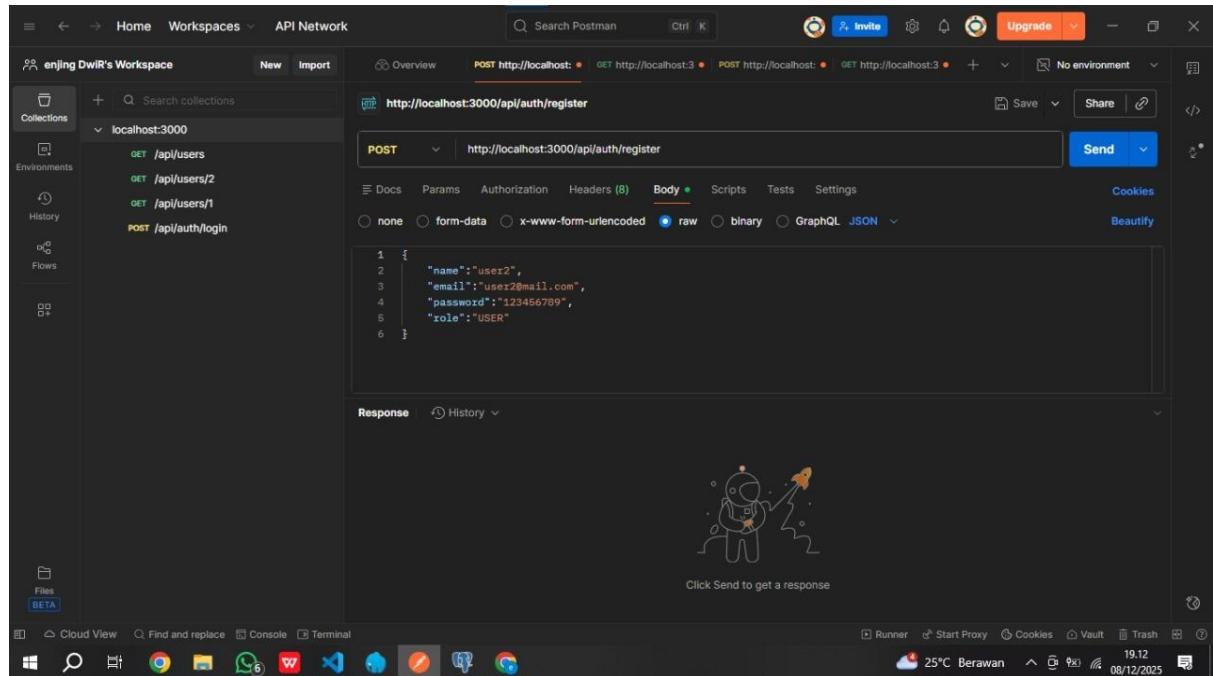
1. Konfigurasi Prisma: Schema di prisma/schema.prisma, migrations di prisma/migrations, dan datasource dari DATABASE_URL.

Cara Menjalankan Proyek

1. Install dependencies: npm install.
2. Setup environment: Copy .env.example ke .env dan isi variabel.
3. Jalankan Prisma: npx prisma generate dan npx prisma migrate dev.
4. Development: npm run dev (server di <http://localhost:3000>).

Screenshot Hasil Testing

1. Menambahkan / mendaftar sebagai admin dan user

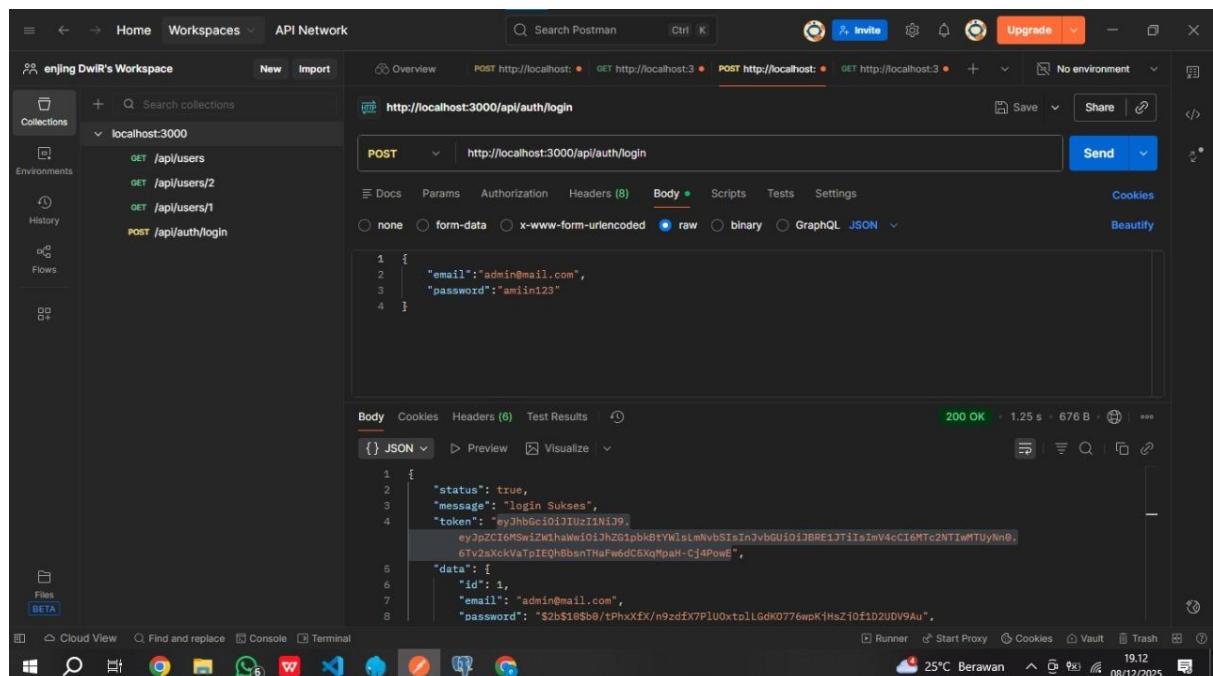


The screenshot shows the Postman application interface. On the left, the sidebar lists collections, environments, history, flows, and files. The main area shows a collection named "localhost:3000" with several API endpoints listed under it. A specific POST request to "http://localhost:3000/api/auth/register" is selected. The "Body" tab is active, showing a raw JSON payload:

```
1 {
2   "name": "user2",
3   "email": "user2@mail.com",
4   "password": "123456789",
5   "role": "USER"
6 }
```

At the bottom right of the main window, there is a cartoon illustration of an astronaut. Below the illustration, a button says "Click Send to get a response". The status bar at the bottom shows system icons and the date/time.

2. Login sebagai admin untuk mendapatkan token



The screenshot shows the Postman application interface. The sidebar and collection structure are identical to the previous screenshot. A new POST request to "http://localhost:3000/api/auth/login" is selected. The "Body" tab is active, showing a raw JSON payload:

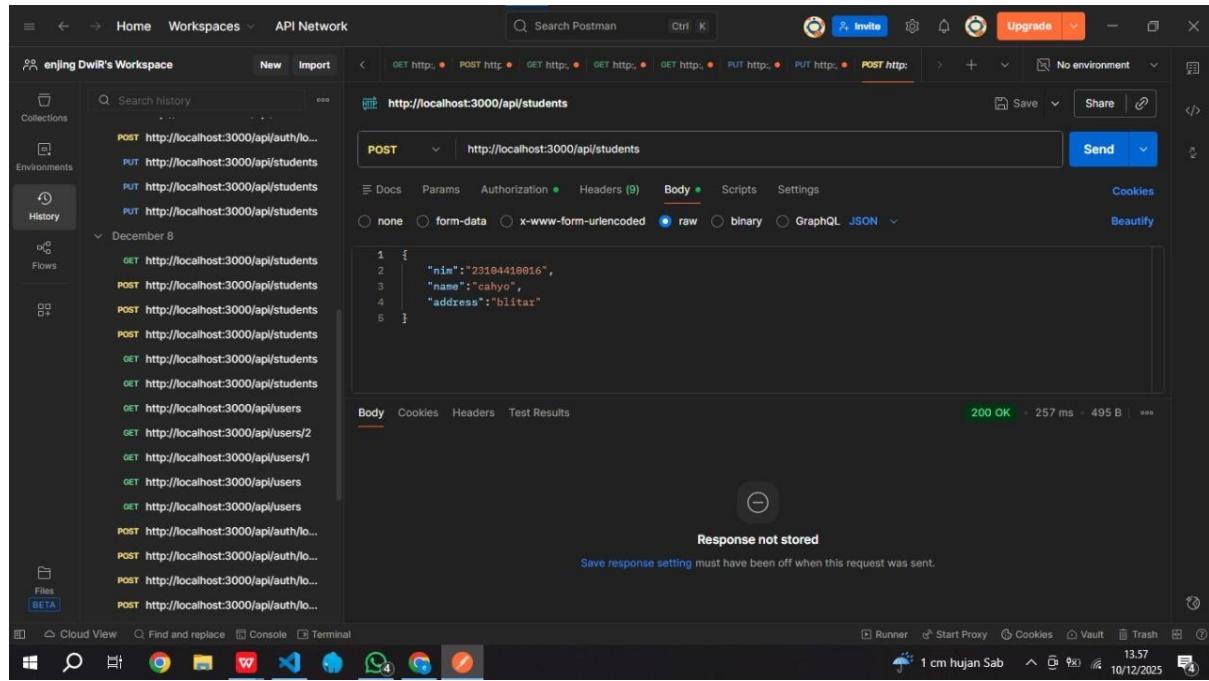
```
1 {
2   "email": "admin@mail.com",
3   "password": "amini123"
4 }
```

The "Test Results" section at the bottom shows a successful response with status code 200 OK, execution time of 1.25 s, and response size of 676 B. The response body is displayed in JSON format:

```
1 {
2   "status": true,
3   "message": "login Sukses",
4   "token": "eyJhbGciOiJIUzI1NiJ9.\n        eyJpZC16MwizWihawwIoiJhZG1pbkByYisLmNvbSisInJvbGUiOiBRE1JTIisImV4cCI6MTc2NTIwMTUyNn0.\n        G7v2axckvapIEQhBbsnThaFw6dC5XqMpAh-Cj4PwE",
5   "data": [
6     {
7       "id": 1,
8       "email": "admin@mail.com",
9       "password": "$2b$10$b@/tPhxXEx/n9zdFx7Plu0xtplLGdK0776wpKjHsZ10f1D2UDV9Au",
10      "role": "ADMIN"
11    }
12  ]}
```

The status bar at the bottom shows system icons and the date/time.

3. Menambahkan data student

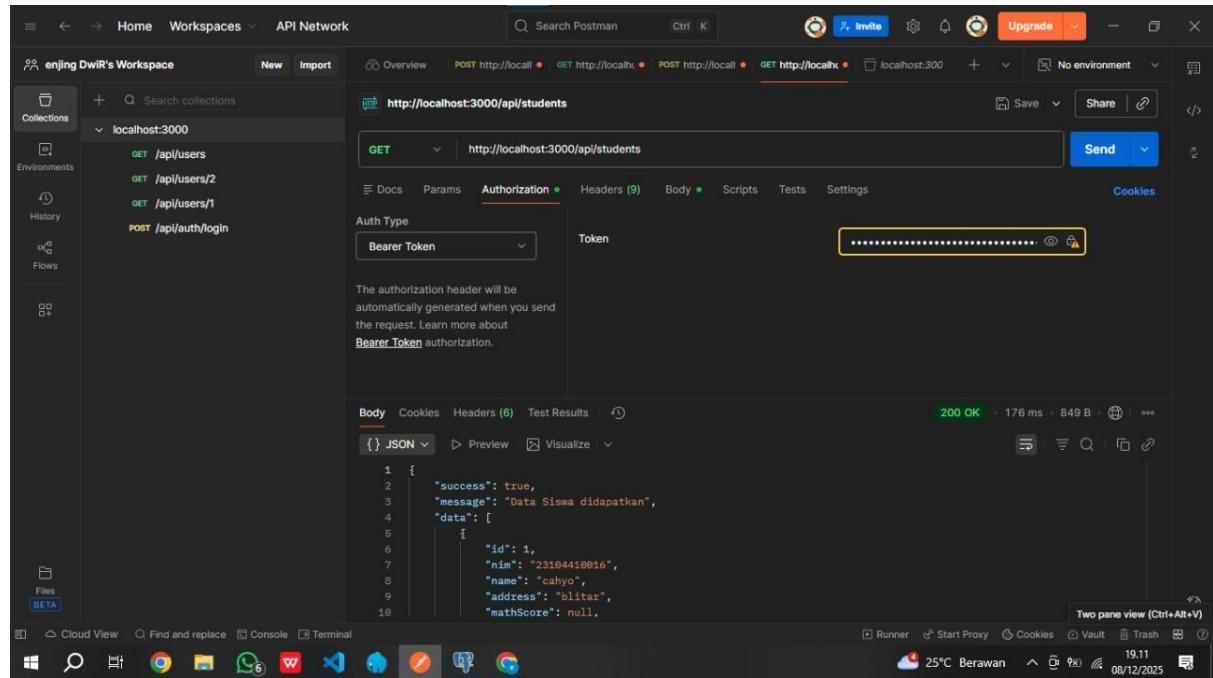


The screenshot shows the Postman application interface. On the left sidebar, there's a collection named "enjeng Dwir's Workspace" containing various API endpoints. The main workspace shows a POST request to `http://localhost:3000/api/students`. The "Body" tab is selected, displaying the following raw JSON:

```
1: {  
2:   "nim": "23104410016",  
3:   "name": "cahyo",  
4:   "address": "blitar"  
5: }
```

The response status is `200 OK` with a duration of `257 ms` and a size of `495 B`. A note at the bottom says "Response not stored" and "Save response setting must have been off when this request was sent".

4. Mendapatkan semua data students



The screenshot shows the Postman application interface. The left sidebar shows a collection named "localhost:3000" with endpoints like `/api/users`, `/api/users/2`, `/api/users/1`, and `POST /api/auth/login`. The main workspace shows a GET request to `http://localhost:3000/api/students`. The "Authorization" tab is selected, showing "Bearer Token" and a token input field containing a redacted string. The "Body" tab shows a JSON response:

```
1: {  
2:   "success": true,  
3:   "message": "Data Siswa didapatkan",  
4:   "data": [  
5:     {  
6:       "id": 1,  
7:       "nim": "23104410016",  
8:       "name": "cahyo",  
9:       "address": "blitar",  
10:      "mathScore": null,  
11:    }  
12:  ]  
13: }
```

The response status is `200 OK` with a duration of `176 ms` and a size of `849 B`. A note at the bottom says "Two pane view (Ctrl+Alt+V)".

5. Memperbarui (Update) data student

The screenshot shows the Postman application interface. In the left sidebar, under 'History', there is a list of API requests. The main area displays a 'PUT' request to 'http://localhost:3000/api/students/1'. The 'Body' tab is selected, showing a JSON payload:

```
1 "nim": "23104410016",
2 "name": "cahyo",
3 "address": "blitar",
4 "mathScore": 86,
5 "computerScore": 90
```

Below the body, the response is shown in a JSON object:

```
3 "message": "Data siswa berhasil diupdate",
4 "data": [
5   {
6     "id": 1,
7     "nim": "23104410016",
8     "name": "cahyo",
9     "address": "blitar",
10    "mathScore": 86,
11    "computerScore": 98,
12    "createdAt": "2025-12-09T11:55:38.338Z",
13    "updatedAt": "2025-12-09T17:35:57.474Z"
14  ]
15 ]
```

6. Delete data student

The screenshot shows the Postman application interface. In the left sidebar, under 'History', there is a list of API requests. The main area displays a 'DELETE' request to 'http://localhost:3000/api/students/3'. The 'Body' tab is selected, showing a single line of text: '1 Ctrl+Alt+P to Ask AI'. Below the body, the response is shown in a JSON object:

```
2 "status": true,
3 "message": "Student berhasil dihapus",
4 "data": [
5   {
6     "id": 3,
7     "nim": "23104410018",
8     "name": "ubah",
9     "address": "malang",
10    "mathScore": null,
11    "computerScore": null,
12    "createdAt": "2025-12-09T11:56:20.804Z",
13    "updatedAt": "2025-12-09T11:56:20.804Z"
14  ]
15 ]
```

Kesimpulan

Proyek "project-api-nextjs" berhasil mengimplementasikan API backend yang aman dan scalable menggunakan Next.js. Ini mencakup autentikasi, otorisasi, dan integrasi database, yang sesuai dengan tujuan UAS kelompok untuk mendemonstrasikan kemampuan pengembangan web modern. Demikian laporan ini kami buat dengan sebenar-benarnya.