

# Aplikasi CRUD dengan React, Express, dan PostgreSQL

---

Aplikasi ini merupakan proyek latihan untuk mengimplementasikan fitur CRUD menggunakan **React (Vite)** di frontend, **Express.js** di backend, dan **PostgreSQL** sebagai database.

Developed by:

- Raisha Alika Irwandira (10231077)
  - Yosan Pratiwi (10231091)
- 

Developed by:

- Raisha Alika Irwandira (10231077)
  - Yosan Pratiwi (10231091)
- 

## 💡 Fitur Aplikasi

- Menampilkan daftar produk dengan tampilan responsif
  - Menambahkan produk baru dengan validasi input
  - Mengedit produk yang sudah ada dengan modal form
  - Menghapus produk dengan konfirmasi sebelum eksekusi
  - Menerapkan notifikasi sukses/gagal pada operasi CRUD
  - Menghubungkan frontend dan backend menggunakan API
  - Menampilkan data dalam tabel dengan fitur pencarian dan pagination
  - Menggunakan state management dengan React Hooks (`useState, useEffect`)
- 

## 🚀 Cara Install dan Menjalankan Aplikasi

### 1. Clone Repository

```
git clone https://github.com/yosanpratiwi/ecommerce-kecil.git  
cd repository-name
```

### 2. Install Dependensi

#### Backend

```
cd backend  
npm install
```

## Frontend

```
cd frontend  
npm install
```

## 3. Konfigurasi Database

1. Pastikan PostgreSQL sudah terinstall.
2. Buat database dengan nama **ecommerce\_kecil**.
3. Sesuaikan konfigurasi di file **db.js**.
4. Jalankan migrasi database:

```
cd backend  
node migrate.js
```

## 4. Menjalankan Aplikasi

### Menjalankan Backend

```
cd backend  
node index.js
```

Backend akan berjalan di <http://localhost:3001>.

### Menjalankan Frontend

```
cd frontend  
npm run dev
```

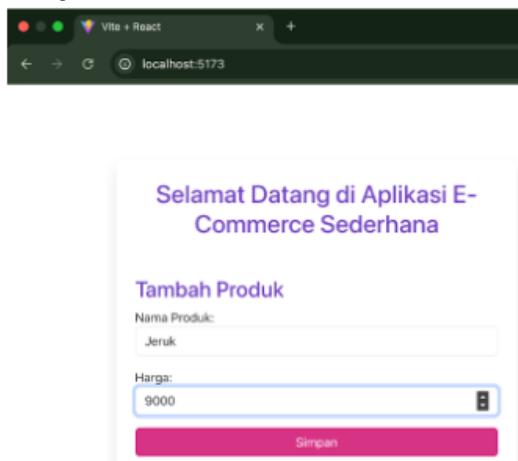
Frontend akan berjalan di <http://localhost:5173>.

## ⌚ Teknologi yang Digunakan

- **Frontend:** React (Vite), Tailwind CSS, Axios, React Hooks
- **Backend:** Express.js, Node.js, CORS
- **Database:** PostgreSQL, pgAdmin
- **Tools:** Postman, GitHub, Railway, Vercel
- **State Management:** useState, useEffect, Context API
- **Deployment:** GitHub Actions, Vercel, Railway

## 📸 Screenshot Aplikasi

### 1. Tampilan Tambah Produk



Selamat Datang di Aplikasi E-Commerce Sederhana

Tambah Produk

Nama Produk: Jeruk

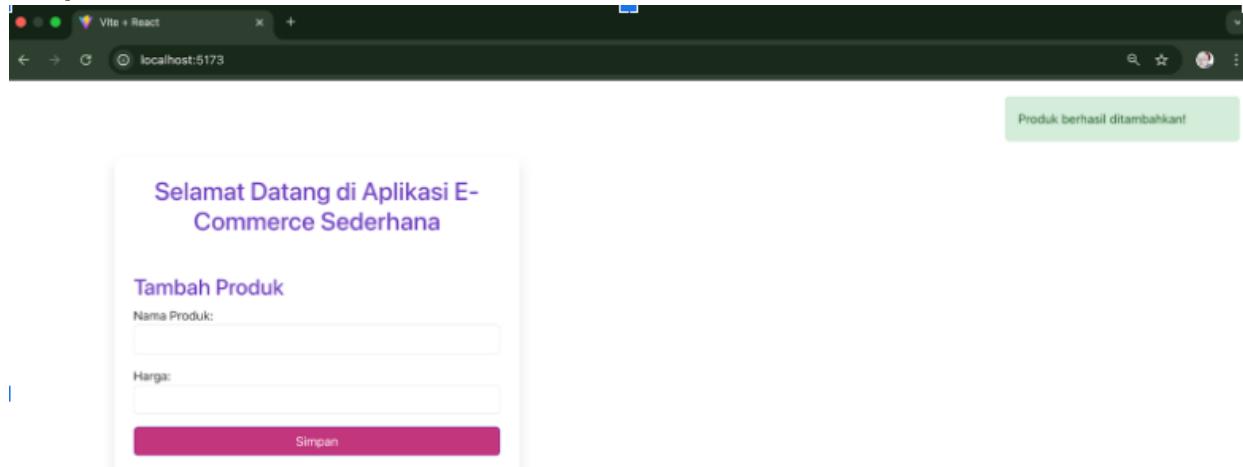
Harga: 9000

**Simpan**


Nama Produk	Harga	Aksi
Bawang Merah	Rp10000	<b>Edit</b> <b>Delete</b>
Alpukat	Rp35000	<b>Edit</b> <b>Delete</b>
Anggur	Rp25000	<b>Edit</b> <b>Delete</b>

### 2. Tampilan Read Data dalam Tabel



Produk berhasil ditambahkan!

Selamat Datang di Aplikasi E-Commerce Sederhana

Tambah Produk

Nama Produk:

Harga:

**Simpan**


Nama Produk	Harga	Aksi
Bawang Merah	Rp10000	<b>Edit</b> <b>Delete</b>
Alpukat	Rp35000	<b>Edit</b> <b>Delete</b>
Anggur	Rp25000	<b>Edit</b> <b>Delete</b>

### 3. Tampilan Edit Produk

Selamat Datang di Aplikasi E-Commerce Sederhana

Tambah Produk

Nama Produk:

Harga:

Simpan

Edit Produk

Nama Produk: Jeruk

Harga: 12000

Simpan Batal

Daftar Produk

Nama Produk	Harga	Aksi
Bawang Merah	Rp10000	<button>Edit</button> <button>Delete</button>
Alpukat	Rp35000	<button>Edit</button> <button>Delete</button>
Anggur	Rp25000	<button>Edit</button> <button>Delete</button>
Jeruk	Rp9000	<button>Edit</button> <button>Delete</button>

Produk berhasil diperbarui!

Selamat Datang di Aplikasi E-Commerce Sederhana

Tambah Produk

Nama Produk:

Harga:

Simpan

Daftar Produk

Nama Produk	Harga	Aksi
Bawang Merah	Rp10000	<button>Edit</button> <button>Delete</button>
Alpukat	Rp35000	<button>Edit</button> <button>Delete</button>
Anggur	Rp25000	<button>Edit</button> <button>Delete</button>
Jeruk	Rp12000	<button>Edit</button> <button>Delete</button>

#### 4. Tampilan Hapus Produk

The screenshot shows a browser window titled "Vite + React" at "localhost:5173". A green notification bar at the top right says "Produk berhasil dihapus!". Below it, a heading "Selamat Datang di Aplikasi E-Commerce Sederhana" is displayed. On the left, a form titled "Tambah Produk" contains fields for "Nama Produk" and "Harga", with a pink "Simpan" button. On the right, a table titled "Daftar Produk" lists three items: "Bawang Merah" (Rp10000), "Alpukat" (Rp35000), and "Jeruk" (Rp12000). Each row has "Edit" and "Delete" buttons.

## 🔧 Troubleshooting

### 1. Pesan "Cannot GET /" pada localhost:3001

A screenshot of a browser window showing a 404 error. The address bar says "localhost:3001". Below the address bar, there are links for "Gmail", "YouTube", "Maps", and "Adobe Acrobat". The main content area displays the text "Cannot GET /".

- Pastikan server Node.js atau Express berjalan dengan benar.
- Jika menggunakan Express, tambahkan rute di server.js atau app.js:

```
app.get('/', (req, res) => {
  res.send('Hello, World!');
});
```

- Pastikan ada script untuk menjalankan server dengan benar.
- Jika sudah ada rute yang benar, coba restart server:

```
ctrl + c  
npm start
```

## 2. Nama dan harga tidak tersimpan dengan benar di database

The screenshot shows the Postman interface. At the top, it says "POST" and the URL "http://localhost:3001/api/produk". Below the URL, there are tabs for "Params", "Authorization", "Headers (9)", "Body", "Pre-request Script", and "Tests". The "Body" tab is selected, with "raw" selected as the type. The JSON body is:

```
1 {  
2   "nama": "Produk 0",  
3   "harga": 50000  
4 }  
5
```

Below the request, there are tabs for "Body", "Cookies", "Headers (8)", and "Test Results". The "Body" tab is selected, showing the response status as "Status: 200 OK". The response body is:

```
1 {  
2   "id": 55,  
3   "nama": null,  
4   "harga": null  
5 }
```

- Jika menggunakan Express, pastikan middleware `express.json()` sudah ada di `server.js` atau `app.js`:

```
app.use(express.json());  
app.use(express.urlencoded({ extended: true }));
```

- Pastikan nama kolom di database cocok dengan yang dikirim dari frontend (nama, harga).
- Pastikan query INSERT ke database sudah benar. Contoh untuk PostgreSQL dengan pg:

```

const insertQuery = 'INSERT INTO produk (nama, harga) VALUES ($1, $2) RETURNING *';
const values = [req.body.nama, req.body.harga];

const result = await pool.query(insertQuery, values);
res.json(result.rows[0]);

```

- Periksa apakah request dikirim dalam format JSON (di Postman pilih raw JSON).
- Tambahkan log untuk memastikan data diterima dengan benar di backend:

```
console.log(req.body);
```

### 3. Data nama dan harga tersimpan sebagai null

The screenshot shows the pgAdmin 4 interface. In the top navigation bar, 'File', 'Object', 'Tools', 'Edit', 'View', 'Window', and 'Help' are visible. Below the navigation bar, there are two tabs: 'Welcome' and 'ecommerce-kecil/postgres@PostgreSQL 17\*'. The 'ecommerce-kecil/postgres@PostgreSQL 17\*' tab is active. In the main area, there is a toolbar with various icons. Below the toolbar, the 'Query' tab is selected, showing the following SQL code:

```

1  SELECT column_name, data_type FROM information_schema.columns WHERE table_name = 'produk';
2

```

Next to the code, there is a button labeled 'Execute script' with an F5 icon. At the bottom of the query window, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Data Output' tab is selected, displaying a table with three rows of data:

	column_name	data_type
1	id	integer
2	nama	character varying
3	harga	integer

- Pastikan Express bisa membaca request body JSON:

```
app.use(express.json());
```

- Pastikan query INSERT di backend benar dan menggunakan parameterized query:

```
const insertQuery = 'INSERT INTO produk (nama, harga) VALUES ($1, $2) RETURNING *';
const values = [req.body.nama, req.body.harga];

const result = await pool.query(insertQuery, values);
res.json(result.rows[0]);
```

- Kirim request dalam format JSON (di Postman: raw JSON).
- Cek apakah req.body berisi data yang benar:

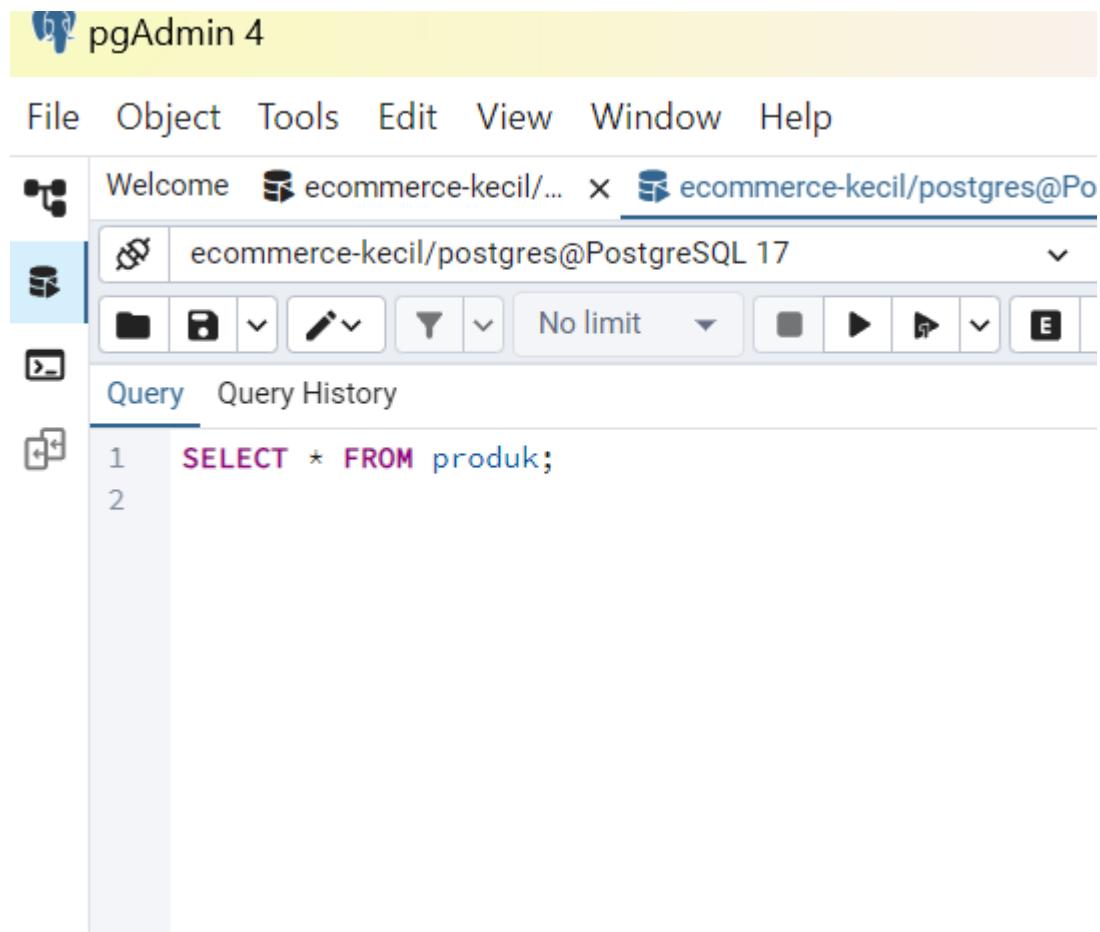
```
console.log(req.body);
```

- Jalankan SQL berikut di pgAdmin untuk memastikan tidak ada constraint yang mencegah insert:

```
INSERT INTO produk (nama, harga) VALUES ('Produk Test', 50000);
```

- Setelah perbaikan, restart server Node.js:

#### 4. Data tidak tersimpan dengan benar saat insert



Data Output    Messages    Notifications

**SQL**

	<b>id</b> [PK] integer	<b>nama</b> character varying (50)	<b>harga</b> integer
2	39	Produk D	200000
3	41	Produk d	5000000
4	42	Produk E	300000
5	43	hrugeugtu	-1
6	47	hrugeugtu	-1
7	44	Produk C	230000
8	48	Produk F	99999
9	49	[null]	[null]
10	50	[null]	[null]
11	51	[null]	[null]
12	52	[null]	[null]
13	53	[null]	[null]
14	54	[null]	[null]
15	55	[null]	[null]

Total rows: 15    Query complete 00:00:00.056

- Pastikan express.json() Aktif di Backend

```
app.use(express.json());
```

- Gunakan console.log(req.body); untuk memastikan data yang dikirim ke backend sudah sesuai.
- Validasi Data Sebelum Insert Tambahkan validasi sebelum menyimpan data:

```
if (!req.body.nama || req.body.harga < 0) {
    return res.status(400).json({ error: "Data tidak valid" });
}
```

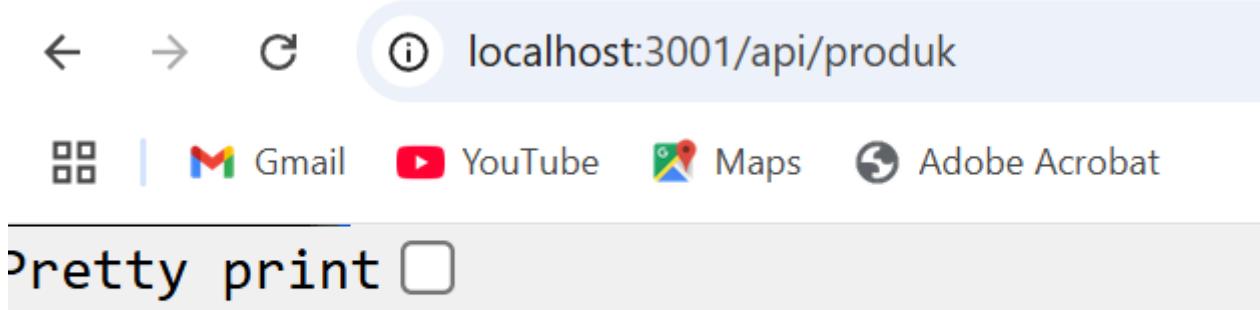
- Pastikan Query INSERT Menggunakan Parameter yang Benar

```
const insertQuery = 'INSERT INTO produk (nama, harga) VALUES ($1, $2) RETURNING *';
const values = [req.body.nama, req.body.harga];
const result = await pool.query(insertQuery, values);
res.json(result.rows[0]);
```

- Hapus Data Null untuk Membersihkan Database

```
DELETE FROM produk WHERE nama IS NULL;
```

## 5. Kesalahan di backend



- Tambahkan `console.error(err)`; di bagian error handler:

```
app.use((err, req, res, next) => {
  console.error(err);
  res.status(500).json({ error: "Server error" });
});
```

- Pastikan Database PostgreSQL Terhubung
- Tes Koneksi ke Database dengan menambahkan ini sebelum query:

```
pool.query('SELECT 1', (err, res) => {
  if (err) {
    console.error("Database connection error", err);
  } else {
```

```
        console.log("Database connected!");
    }
});
```

- Pastikan tidak ada kesalahan pada query SQL (INSERT, SELECT, dll.).

## 6. Database PostgreSQL Tidak Terhubung

- Pastikan PostgreSQL sudah berjalan dan user `postgres` memiliki akses.
- Cek konfigurasi di `db.js`.
- Jalankan perintah berikut di PostgreSQL untuk memastikan database sudah ada:

```
SELECT * FROM produk;
```

## 7. Masalah Fetch Data di Frontend

- Pastikan backend berjalan dengan benar di `http://localhost:3001`.
- Gunakan `npm install axios` jika Axios belum terinstall.
- Pastikan CORS sudah diaktifkan di backend.

## ❖ Best Practices dalam Pengembangan

- **Gunakan Environment Variables:** Simpan kredensial database di `.env`.
- **Gunakan GitHub Issues & Projects:** Kelola tugas dengan jelas.
- **Gunakan Middleware di Express:** Untuk menangani error dan autentikasi.
- **Optimasi Query di PostgreSQL:** Gunakan indeks jika perlu untuk performa tinggi.
- **Pastikan UI Responsif:** Gunakan media queries dengan Tailwind CSS.

## ❖ Commit dan Push ke GitHub

```
git add .
git commit -m "Menambahkan dokumentasi dan fitur baru"
git push origin main
```

## Inspirasi Hari Ini

"Mengetahui saja tidak cukup; kita harus menerapkannya. Berkeinginan saja tidak cukup; kita harus melakukannya." – Goethe