



POLITEKNIK NEGERI BANYUWANGI

Laporan Final Project MHS 1 (Vendor A): Banyuwangi Marketplace

Interopabilitas
Semester Ganjil 2024/2025

Disusun Oleh Kelompok 02:

Inka Nazwa Fitri	NIM 362458302047	<i>Mahasiswa 1 (Vendor A)</i>
Dany Darmawan	NIM 362458302046	<i>Mahasiswa 2 (vendor B)</i>
Moh. Syaifudin	NIM 362458302109	<i>Mahasiswa 3 (Vendor C)</i>
Nur Ifani Chairun Nisa	NIM 362458302048	<i>Mahasiswa 4</i>

Tahun 2025

BAB 1

BAB 1 Pendahuluan & Pembagian Kerja

1.1 Deskripsi Aplikasi

Aplikasi yang dikembangkan pada tugas ini merupakan layanan backend Vendor A yang berfungsi sebagai penyedia data produk UMKM dengan karakteristik sistem warung legacy. Aplikasi ini dibangun sebagai bagian dari studi kasus **Banyuwangi Marketplace** pada mata kuliah **Interoperabilitas**, di mana setiap vendor memiliki struktur data yang berbeda.

1.2 Simulasi Sistem Legacy

Saya sebagai Vendor A mensimulasikan sistem lama yang masih menggunakan format data sederhana. Seluruh atribut data produk, termasuk harga, direpresentasikan dalam bentuk **String**. Hal ini bertujuan untuk merefleksikan kondisi sistem legacy yang belum menerapkan **type safety** secara ketat.

1.3 Tabel Pembagian Tanggung Jawab

[Mahasiswa 1, Inka Nazwa Fitri (362459302047) — Backend Vendor A, API, PostgreSQL (Neon), Format legacy (string)]

[Mahasiswa 2, Dany Darmawan (362458302046) — Backend Vendor B]
--

[Mahasiswa 3, Moh. Syaifudin (362458302109) — Backend Vendor C]

[Mahasiswa 4, Nur Ifani Chairun Nisa (362458302048) — Integrasi data ke format JSON]
--

1.4 Tujuan

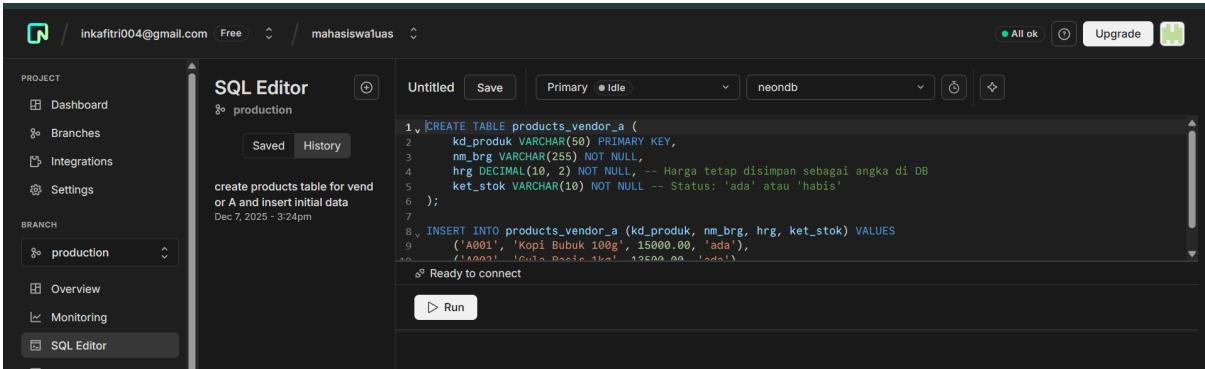
Tujuan saya sebagai Mahasiswa 1 dalam mengembangkan aplikasi Vendor A adalah:

1. Menyediakan layanan backend sebagai sumber data produk warung.
2. Mensimulasikan sistem legacy dengan tipe data **string**.
3. Menyediakan data yang siap digunakan pada proses **interoperabilitas**.

BAB 2

Deskripsi Sistem Vendor A

Sistem Vendor A yang saya kembangkan merupakan sebuah REST API backend yang dibangun menggunakan Node.js dan framework Express.js. Aplikasi ini tidak memiliki tampilan frontend karena fokus utama saya adalah menyediakan data melalui endpoint API. Data produk disimpan pada database PostgreSQL berbasis cloud menggunakan layanan Neon. Aplikasi ini menyediakan endpoint yang dapat diakses oleh sistem lain untuk mengambil data produk warung.



The screenshot shows the Neon web interface. On the left, there's a sidebar with 'PROJECT' and 'BRANCH' sections. Under 'PROJECT', 'Dashboard', 'Branches', 'Integrations', and 'Settings' are listed. Under 'BRANCH', 'production' is selected. The main area is titled 'SQL Editor' and shows an 'Untitled' tab. The code editor contains the following SQL script:

```
1 v CREATE TABLE products_vendor_a (
2   kd_produk VARCHAR(50) PRIMARY KEY,
3   nm_brg VARCHAR(255) NOT NULL,
4   hrg DECIMAL(10, 2) NOT NULL, -- Harga tetap disimpan sebagai angka di DB
5   ket_stok VARCHAR(10) NOT NULL -- Status: 'ada' atau 'habis'
6 );
7
8 v INSERT INTO products_vendor_a (kd_produk, nm_brg, hrg, ket_stok) VALUES
9   ('A001', 'Kopi Bubuk 100g', 15000.00, 'ada'),
10  ('A002', 'Kopi Bubuk 1kg', 12500.00, 'ada')
```

Below the code editor, it says 'Ready to connect' and has a 'Run' button. At the top right, there are status indicators: 'All ok', 'Upgrade', and a green circular icon.

Gambar 1: Konfigurasi di Neon

2.1 Karakteristik Data Vendor A Ciri Utama Data Vendor A:

1. Semua field bertipe String
2. Harga produk dikirim dalam bentuk String
3. Status stok menggunakan teks sederhana seperti “ada” atau “habis”

Field Utama: kd_produk, nm_brg, hrg, ket_stok

BAB 3

Arsitektur dan Alur Sistem

Arsitektur aplikasi vendor A yang saya rancang terdiri dari:

1. Integrator sebagai peminta data (mahasiswa 4)
2. Server Express sebagai pengelola request dan response
3. Database PostgreSQL (Neon) sebagai penyimpan data produk
4. Platform Vercel sebagai media deployment

3.1 Alur Kerja Sistem

Alur kerja sistem dimulai ketika mahasiswa 1 dan mahasiswa 4 mengakses endpoint /warung. Server kemudian mengambil data dari database, memformat data sesuai spesifikasi Vendor A, dan mengirimkan response dalam bentuk JSON.

BAB 4

Implementasi Sistem

4.1 Konfigurasi Environment (.env)

```
.env
1 PORT=3001
2 DATABASE_URL=postgresql://neondb_owner:npg_dJ0sKDGb3Lcy@ep-round-cake-a1mtwh0y-pooler.ap-southeast-1.aws.neon.tech/neondb?sslmode=require
```

Gambar 2: Konfigurasi environment

Saya menggunakan file .env untuk menyimpan konfigurasi environment agar informasi sensitif tidak ditulis langsung di dalam kode. Variabel PORT digunakan untuk menentukan port server, sedangkan DATABASE_URL digunakan untuk mengatur koneksi ke database PostgreSQL Neon.

4.2 Koneksi Database (db.js)

```
JS db.js > ...
1 const { Pool } = require('pg');
2
3 // Mengambil DATABASE_URL dari .env
4 const pool = new Pool({
5   connectionString: process.env.DATABASE_URL,
6   ssl: {
7     rejectUnauthorized: false
8   }
9 });
10
11 module.exports = {
12   query: (text, params) => pool.query(text, params),
13 };
```

Gambar 3: Konfigurasi Database

Saya menggunakan library pg sebagai penghubung Node.js dengan PostgreSQL. Objek Pool saya pilih karena mampu mengelola koneksi database secara efisien. Konfigurasi SSL diperlukan karena database berada pada layanan cloud. Fungsi query saya buat agar pemanggilan database menjadi lebih sederhana dan terstruktur.

4.3 Implementasi Server backend (Index.js)

Inisialisasi dan Middleware

```
JS index.js > ...
1 require("dotenv").config();
2 const express = require("express");
3 const cors = require("cors");
4 const db = require("./db");
5 const app = express();
6 const PORT = process.env.PORT || 3001;
7
8 // === MIDDLEWARE ===
9 app.use(cors());
10 app.use(express.json());
```

Gambar 4: Konfigurasi Database dan Server

Saya menggunakan dotenv untuk membaca konfigurasi environment, express sebagai framework backend, dan cors agar API dapat diakses dari domain lain. Middleware express.json() saya gunakan untuk membaca data JSON dari request.

Endpoint Status

```
12 app.get("/status", (req, res) => {
13   res.json({ ok: true, service: "vendor-a-api" });
14 });
15
```

Gambar 5: endpoint /status

Pada Endpoint ini saya sediakan untuk memastikan bahwa server Vendor A berjalan dengan baik dan siap menerima request dari client atau integrator.

Endpoint Utama /warung

```
16 app.get("/warung", async (req, res, next) => {
17   try {
18     const sql = `
19       SELECT kd_produk, nm_brg, hrg, ket_stok
20     FROM warung_vendor_a
21     ORDER BY kd_produk ASC
22   `;
23   const result = await db.query(sql);
24
25   const legacyData = result.rows.map(row => ({
26     kd_produk: String(row.kd_produk),
27     nm_brg: String(row.nm_brg),
28
29     hrg: String(row.hrg),
30
31     ket_stok: String(row.ket_stok),
32   }));
33
34   res.json(legacyData);
35 } catch (err) {
36   console.error("Error fetching data from Vendor A DB:", err.stack);
37   next(err);
38 }
39});
```

Gambar 6: endpoint /warung

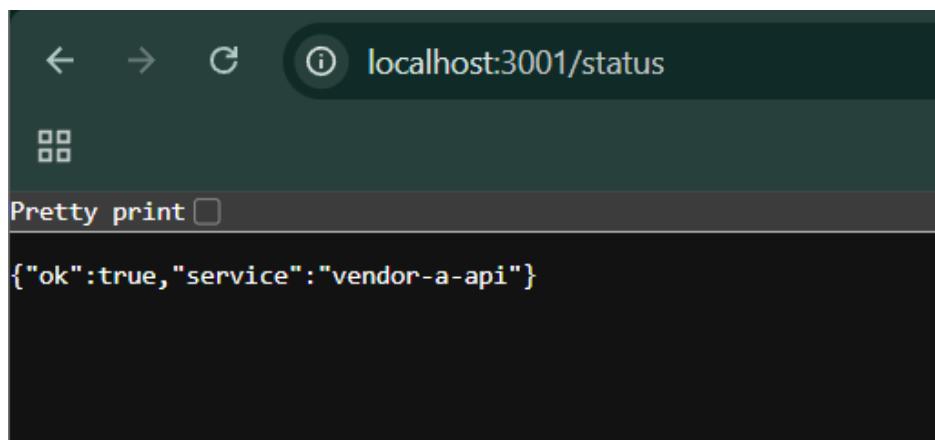
Pada endpoint ini, saya mengambil data produk dari database menggunakan query SQL. Data kemudian saya format ulang dengan memastikan seluruh field bertipe String. Langkah ini penting untuk mensimulasikan sistem legacy sesuai dengan spesifikasi Vendor A.

BAB 5

Pengujian Sistem

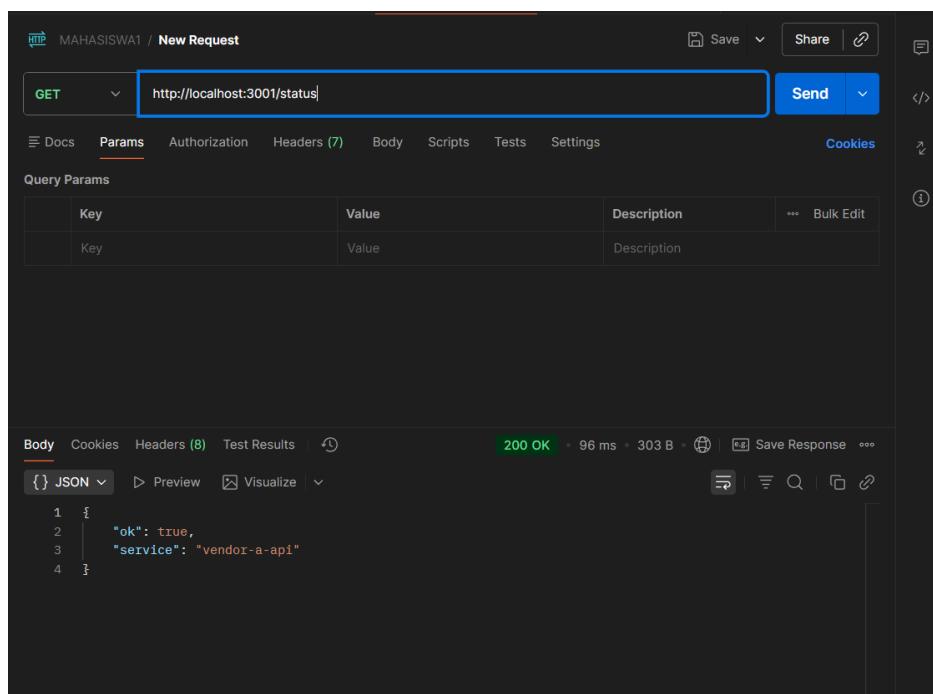
5.1 Metode Pengujian

Saya melakukan pengujian menggunakan browser dan Postman dengan mengakses endpoint /status dan /warung



```
localhost:3001/status
{"ok":true, "service":"vendor-a-api"}
```

Gambar 7: Pengujian /status



MAHASISWAT / New Request

GET http://localhost:3001/status Send

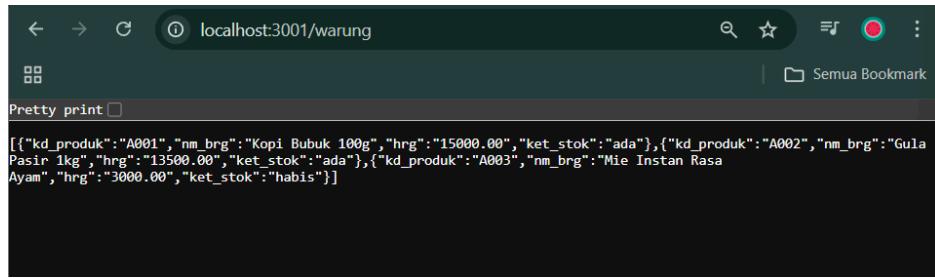
Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (8) Test Results 200 OK 96 ms 303 B Save Response

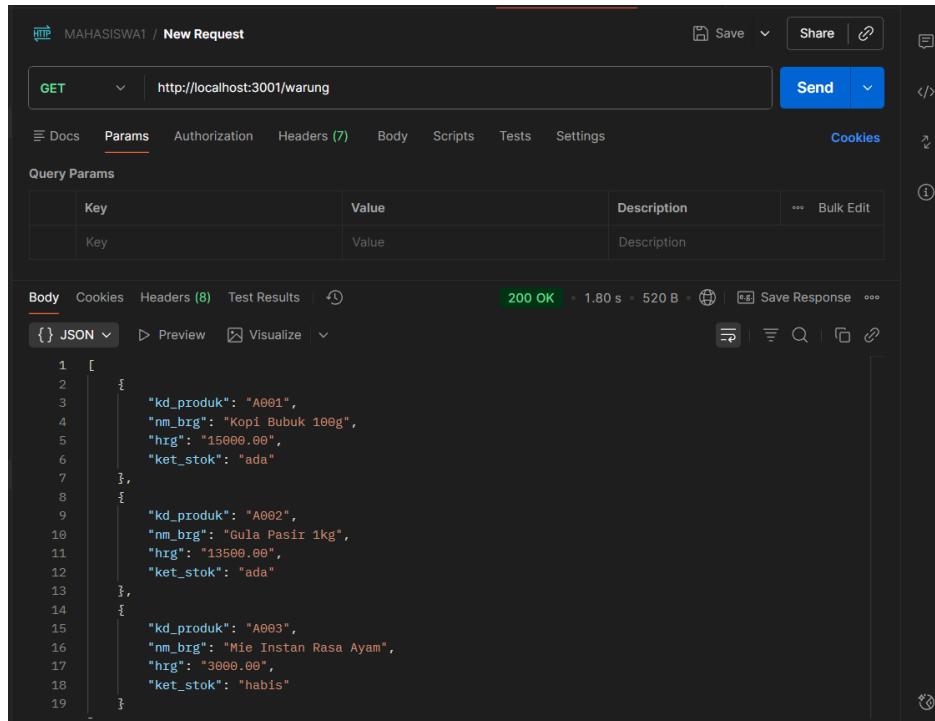
```
{ } JSON ▾ ▶ Preview ▾ Visualize ▾
1 {
2   "ok": true,
3   "service": "vendor-a-api"
4 }
```

Gambar 8: Pengujian Postman /status



```
[{"kd_produk": "A001", "nm_brg": "Kopi Bubuk 100g", "hrg": "15000.00", "ket_stok": "ada"}, {"kd_produk": "A002", "nm_brg": "Gula Pasir 1kg", "hrg": "13500.00", "ket_stok": "ada"}, {"kd_produk": "A003", "nm_brg": "Mie Instan Rasa Ayam", "hrg": "3000.00", "ket_stok": "habis"}]
```

Gambar 9: Pengujian /warung



```
HTTP MAHASISWA / New Request  
GET http://localhost:3001/warung Send </>  
Docs Params Authorization Headers (7) Body Scripts Tests Settings Cookies  
Query Params  


| Key | Value | Description | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | ... |           |

  
Body Cookies Headers (8) Test Results 200 OK 1.80 s 520 B Save Response <...>  
{} JSON Preview Visualize  
1 [  
2 {  
3 "kd_produk": "A001",  
4 "nm_brg": "Kopi Bubuk 100g",  
5 "hrg": "15000.00",  
6 "ket_stok": "ada"  
7 },  
8 {  
9 "kd_produk": "A002",  
10 "nm_brg": "Gula Pasir 1kg",  
11 "hrg": "13500.00",  
12 "ket_stok": "ada"  
13 },  
14 {  
15 "kd_produk": "A003",  
16 "nm_brg": "Mie Instan Rasa Ayam",  
17 "hrg": "3000.00",  
18 "ket_stok": "habis"  
19 }
```

Gambar 10: Pengujian Postman /warung

5.2 Hasil Pengujian

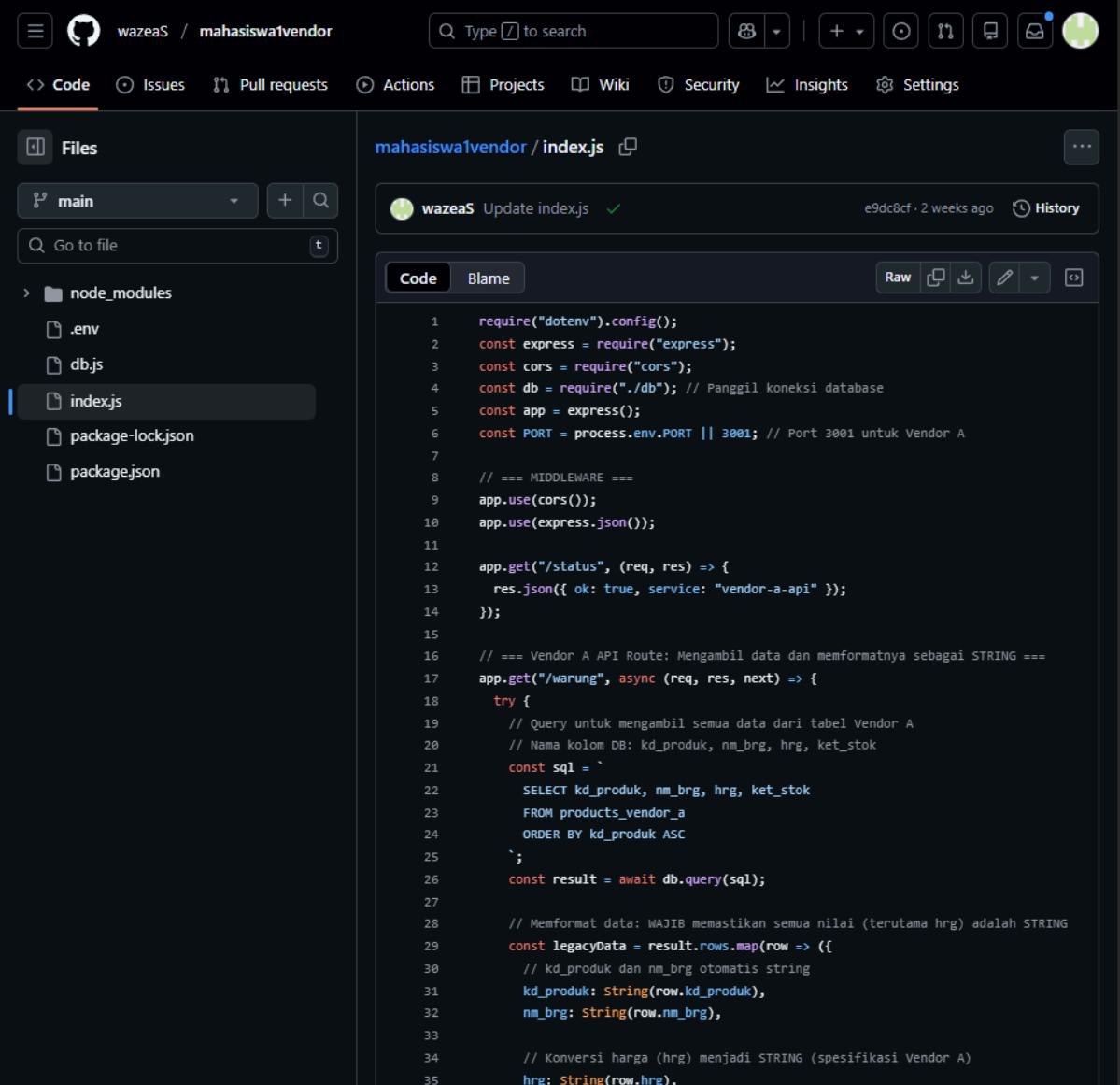
Hasil pengujian menunjukkan bahwa server dapat berjalan tanpa error dan endpoint /warung mengembalikan data JSON yang valid dan sesuai dengan format Vendor A.

BAB 6

Version Control dan Deployment

6.1 Version Control

Dalam pengembangan aplikasi ini, saya menggunakan Git sebagai alat version control dan menyimpan source code pada repository GitHub sebagai dokumentasi pengerjaan.



The screenshot shows a GitHub repository named 'mahasiswa1vendor'. The 'Code' tab is selected. On the left, the file tree shows 'main' (with 'index.js' selected), 'node_modules', '.env', 'db.js', 'package-lock.json', and 'package.json'. The main pane displays the content of 'index.js'. A commit by 'wazeaS' titled 'Update index.js' is shown, made 2 weeks ago with commit hash 'e9dc8cf'. The code in 'index.js' is as follows:

```
1  require("dotenv").config();
2  const express = require("express");
3  const cors = require("cors");
4  const db = require("./db"); // Panggil koneksi database
5  const app = express();
6  const PORT = process.env.PORT || 3001; // Port 3001 untuk Vendor A
7
8  // === MIDDLEWARE ===
9  app.use(cors());
10 app.use(express.json());
11
12 app.get("/status", (req, res) => {
13   res.json({ ok: true, service: "vendor-a-api" });
14 });
15
16 // === Vendor A API Route: Mengambil data dan memformatnya sebagai STRING ===
17 app.get("/warung", async (req, res, next) => {
18   try {
19     // Query untuk mengambil semua data dari tabel Vendor A
20     // Nama kolom DB: kd_produk, nm_brg, hrg, ket_stok
21     const sql = `
22       SELECT kd_produk, nm_brg, hrg, ket_stok
23       FROM products_vendor_a
24       ORDER BY kd_produk ASC
25     `;
26     const result = await db.query(sql);
27
28     // Memformat data: WAJIB memastikan semua nilai (terutama hrg) adalah STRING
29     const legacyData = result.rows.map(row => ({
30       // kd_produk dan nm_brg otomatis string
31       kd_produk: String(row.kd_produk),
32       nm_brg: String(row.nm_brg),
33
34       // Konversi harga (hrg) menjadi STRING (spesifikasi Vendor A)
35       hrg: String(row.hrg),
    
```

Gambar 11: Repository yang sudah dibuat pada github

6.2 Deployment

Aplikasi Vendor A saya deploy menggunakan Vercel sehingga API dapat diakses secara online tanpa perlu menjalankan server secara lokal.

The screenshot shows the 'Deployment Details' page for a Vercel application. At the top right are buttons for 'Share', 'Visit', and more. Below is a table with columns: 'Created' (wazeas Dec 7), 'Status' (Ready Latest), 'Duration' (11s 15d ago), and 'Environment' (Production Current). A code snippet in a box shows an error message: `{"error": "Rute tidak ditemukan"}`. Under 'Domains', there are three entries: 'mahasiswa1vendor.vercel.app' (with a plus icon), 'mahasiswa1vendor-git-main-wazeas-projects.vercel.app', and 'mahasiswa1vendor-hxa83uyt8-wazeas-projects.vercel.app'. Under 'Source', it shows 'main' with a commit hash 'e9dc8cf' and a note 'Update index.js'. At the bottom are links for 'Deployment Settings' and '4 Recommendations'.

Gambar 12: Vercel yang sudah dibuat

The screenshot shows a browser window displaying the deployed Vercel application at the URL 'mahasiswa1vendor-hxa83uyt8-wazeas-projects.vercel.app/warung'. The page content is a JSON array of product data:

```
[{"kd_produk": "A001", "nm_brg": "Kopi Bubuk 100g", "hrg": "15000.00", "ket_stok": "ada"}, {"kd_produk": "A002", "nm_brg": "Gula Pasir 1kg", "hrg": "13500.00", "ket_stok": "ada"}, {"kd_produk": "A003", "nm_brg": "Mie Instan Rasa Ayam", "hrg": "3000.00", "ket_stok": "habis"}]
```

Gambar 13: Link vercel saat dijalankan

BAB 7

Kesimpulan & Penutup

7.1 Kesimpulan

Berdasarkan hasil pengembangan dan pengujian yang telah saya lakukan, aplikasi Vendor A berhasil dibangun sesuai dengan peran saya sebagai Mahasiswa 1. Aplikasi ini mampu menyediakan data produk warung legacy dalam format JSON sederhana dengan seluruh tipe data berupa String dan siap digunakan sebagai sumber data dalam proses interoperabilitas.

7.2 Link Video Demo

Berikut adalah link video demo aplikasi yang diunggah ke Google Drive/YouTube:

- **Link:** [<https://youtu.be/Oc4kE1kf3bA?si=02lvswAD5YTYmjNi>]