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	*Mahasiswa 1 (Vendor A)*
Dany Darmawan	NIM 362458302046	*Mahasiswa 2 (vendor B)*
Moh. Syaifudin	NIM 362458302109	*Mahasiswa 3 (Vendor C)*
Nur Ifani Chairun Nisa	NIM 362458302048	*Mahasiswa 4*

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]	

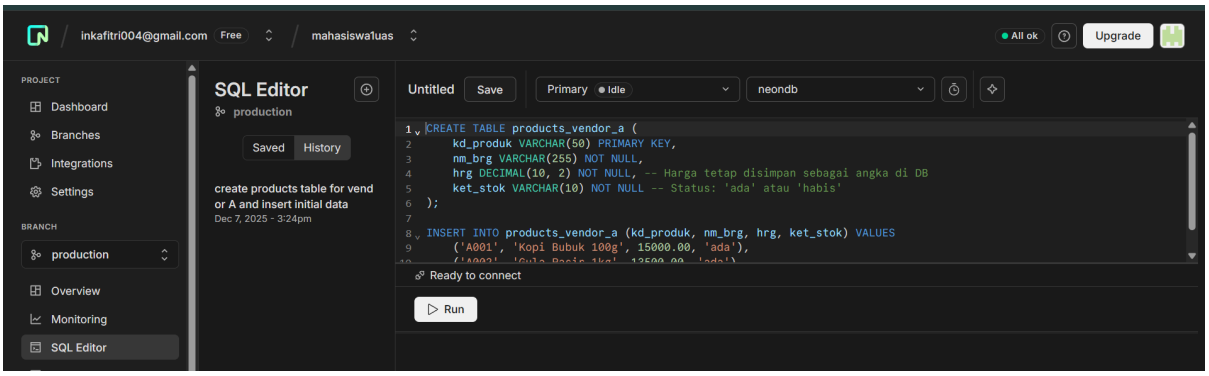
1.4 Tujuan

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.



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=requ
```

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)

```
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

```
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());
11
```

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  });
```

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  });
```

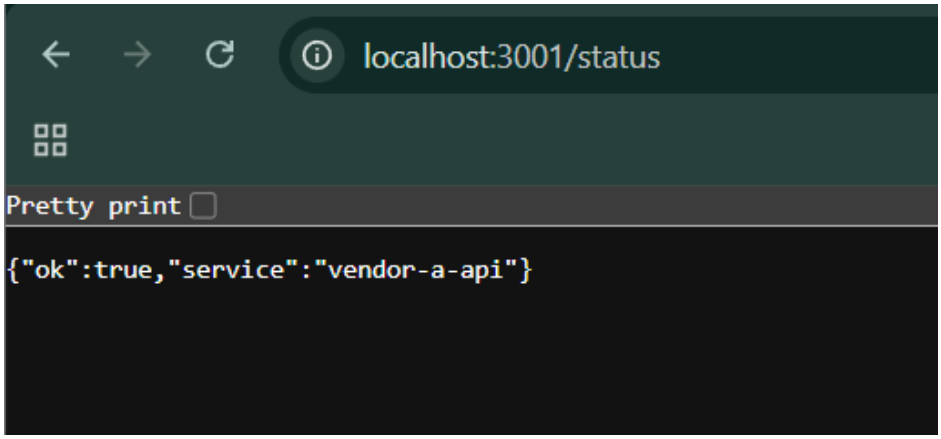
Gambar 6: endpoint /warung

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.

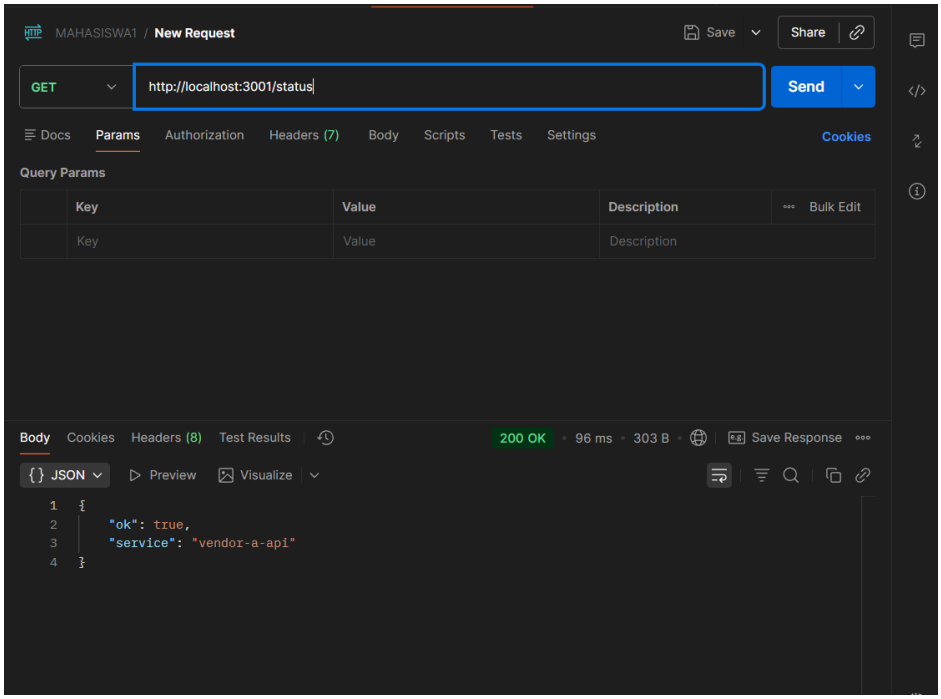
BAB 5
Pengujian Sistem

5.1 Metode Pengujian

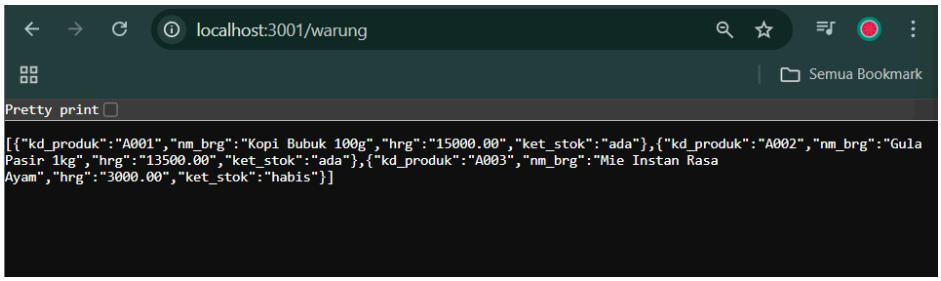
Saya melakukan pengujian menggunakan browser dan Postman dengan mengakses endpoint / status dan /warung



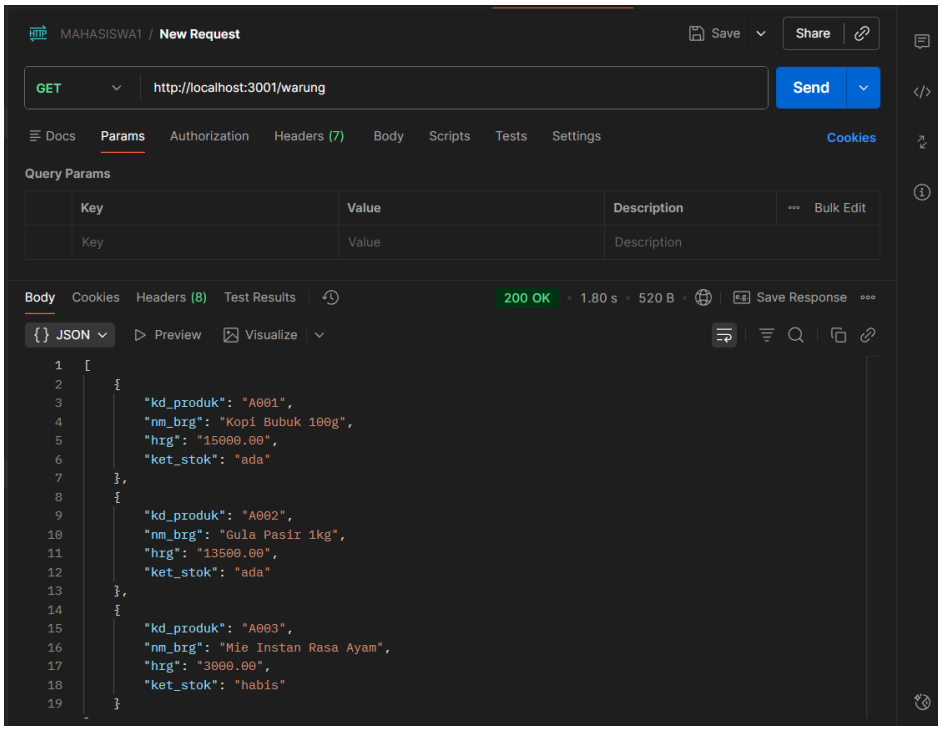
Gambar 7: Pengujian /status



Gambar 8: Pengujian Postman /status



Gambar 9: Pengujian /warung



Gambar 10: Pengujian Postman /warung

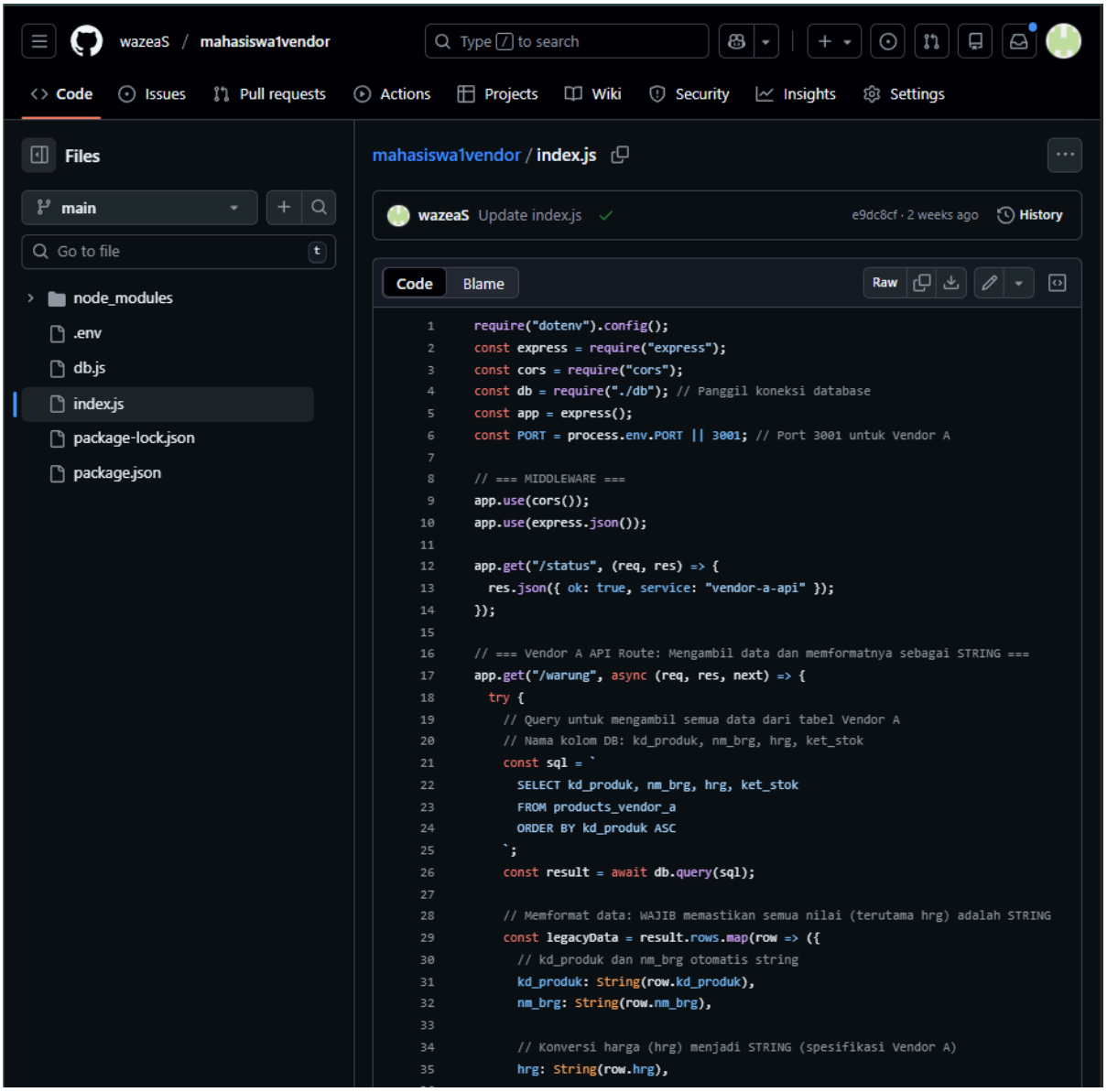
5.2 Hasil Pengujian

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

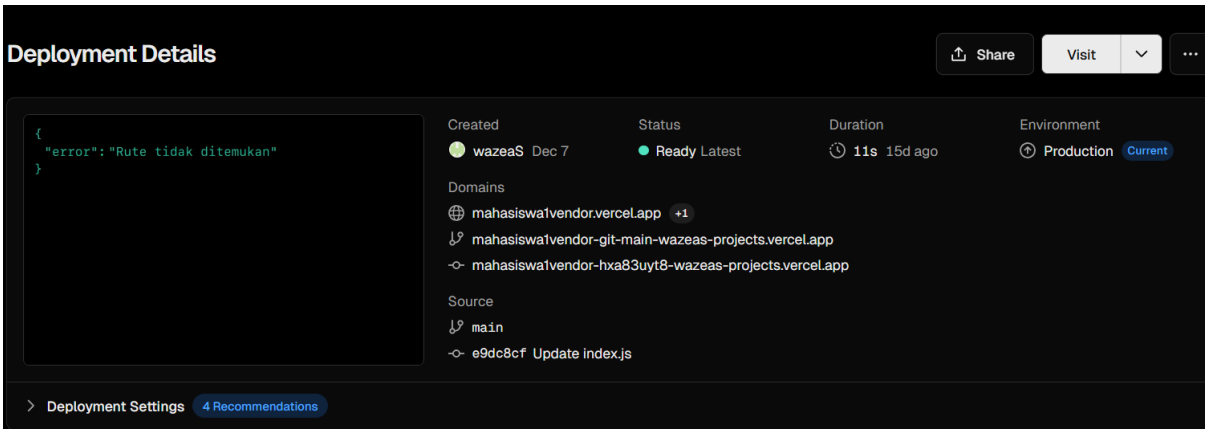
Dalam pengembangan aplikasi ini, saya menggunakan Git sebagai alat version control dan menyimpan source code pada repository GitHub sebagai dokumentasi pengerjaan.



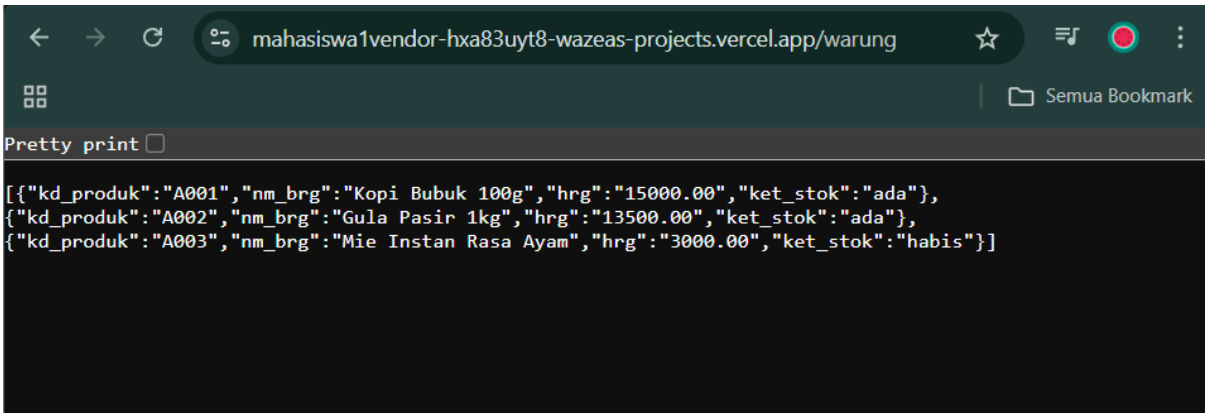
Gambar 11: Repository yang sudah dibuat pada github

6.2 Deployment

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



Gambar 12: Vercel yang sudah dibuat



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]