Implementasi Backend untuk Sistem Informasi Manajemen di Jaringan Restoran CV Balibul dengan Javascript dan Python

Evan Pradipta Hardinatha
School of Electrical Engineering
Telkom University
Bandung, Indonesia
kleopasevan@student.telkomuniversity.ac.id

Anggunmeka Luhur Prasasti
School of Electrical Engineering
Telkom University
Bandung, Indonesia
anggunmeka@telkomuniversity.ac.id

Astri Novianty
School of Electrical Engineering
Telkom University
Bandung, Indonesia
astrinovianty@telkomuniversity.ac.id

***Abstrak* — Solusi berbasis kertas tradisional dalam sistem informasi manajemen restoran tidak efisien dan tidak sesuai karena rentan terhadap kesalahan manusia dan kurangnya kemampuan analisis data. Sistem ini menghambat kemampuan manajer untuk membuat keputusan yang tepat dan berdasarkan data. Penelitian ini mengungkapkan masalah yang ada pada CV Balibul dalam penggunaan teknik manajemen manual yang tidak efisien dan tidak efektif untuk operasional manajemennya. Untuk mengatasi masalah ini, proyek ini memperkenalkan sistem manajemen restoran digital yang canggih yang khusus dirancang untuk Jaringan Restoran CV Balibul. Penelitian ini bertujuan untuk mengembangkan *backend* aplikasi yang terdiri dari *Application Programming Interface* (API) dan *database.* Metode yang digunakan meliputi desain sistem, pengembangan database, pengembangan api, serta pengujian dan evaluasi aplikasi. Hasil penelitian menunjukkan bahwa *backend* aplikasi efektif dan *reliable* dalam meningkatkan efisiensi sistem manajemen restoran digital dengan tingkat kecepatan respon yang mumpuni walaupun terhubung dengna banyak user. *Backend* apalikasi ini menawarkan Solusi yang efisien dalam sistem informasi manajemen restoran dengan performa yang mumpuni dan fitur yang sesuai dengan harapan CV Balibul. Rekomendasi pengembangan lebih lanjut *backend* ini adalah untuk mengoptimalkan performa machine learning API.**

Kata kunci— Backend, Application Programming Interface (API), database, Sistem Informasi Manajemen.

I. PENDAHULUAN

Dalam dunia bisnis yang semakin maju, manajemen yang merupakan seni dan ilmu perencanaan, pengorganisasian, penempatan karyawan, pemberian perintah, dan pengawasan terhadap sumber daya manusia untuk mencapai tujuan yang telah ditentukan [1] menjadi penting dalam keberlanjutan bisnis. Sistem manajemen yang baik juga memerlukan suatu sistem informasi manajemen yang baik. Sistem informasi manajemen akan membantu perusahaan dalam operasi perusahaan dan lingkungan sekitarnya [2], [3].

CV Balibul adalah sebuah perusahaan kuliner yang telah merentang jaringan restoran di wilayah Jawa Tengah dan Daerah Istimewa Yogyakarta. Salah satu masalah utama yang dihadapi oleh CV Balibul adalah sistem manajemen yang masih mengandalkan penggunaan kertas. Penggunaan kertas ini seringkali menghasilkan human error dalam penulisan dan perhitungan. Selain itu, data yang tersimpan pada kertas kurang efektif untuk dianalisis lebih lanjut, menghambat kemampuan perusahaan untuk membuat keputusan yang didasarkan pada data yang akurat dan relevan. Meskipun sebagian proses sudah bergerak menuju digitalisasi dengan adopsi WhatsApp dan aplikasi Qasir, masih ada kesenjangan dalam pengelolaan informasi yang menyulitkan manajemen yang efisien. Semua kelemahan ini telah memunculkan kebutuhan mendesak untuk mengintegrasikan solusi digital yang lebih canggih untuk mengatasi masalah-masalah ini.

II. KAJIAN TEORI

A. JavaScript untuk *backend* aplikasi

JavaScript adalah bahasa pemrograman yang awalnya dikembangkan untuk digunakan dalam web sebagai bahasa scripting yang dapat dengan mudah ditulis di dalam halaman HTML. Saat ini JavaScript digunakan secara luas untuk pengembangan *frontend*, *backend*, dan aplikasi desktop, sehingga dapat dianggap sebagai bahasa pemrograman umum. JavaScript memiliki *runtime environment* bernama Node JS. Node JS dapat digunakan untuk mengembangkan *backend* aplikasi dengan menggunakan *framework* seperti Express.js.

Express JS adalah *framework* yang ada dalam Node.JS yang memungkinkan kita untuk membangun sebuah server aplikasi atau API dalam JavaScript[4]. Express merupakan *framework* yang sederhana, fleksibel, dan cepat yang memungkinkan developer untuk dengan mudah membangun aplikasi berkinerja tinggi dengan menyesuaikan fungsionalitas sesuai kebutuhan[5].

B. Python untuk *backend* aplikasi

Python adalah bahasa pemrograman yang dikembangkan pada akhir 1980-an dan awal 1990-an oleh Guido van Russom di Belanda, merupakan bahasa pemrograman yang dinamis, mendukung pemrograman berorientasi objek dengan penggunaan kelas, dan populer sebagai bahasa pemrograman yang ramah pemula, dengan dukungan komunitas yang besar dan berbagai fitur yang mendukung pengembangan berbagai jenis program[4]. Python juga dapat digunakan untuk mengembangkan *backend* menggunakan *framework* seperti FastAPI.

FastAPI adalah kerangka kerja (*framework*) pada python untuk membuat aplikasi berbasis API web yang kuat. FastAPI dibangun diatas starlette, yang merupakan kerangka kerja web *asynchronous server gateway interface* (ASGI) yang dikenal karena ringan dan berkinerja tinggi. ASGI memungkinkan untuk melakukan asynchronous, yang dapat meningkatkan kinerja[6].

C. *Database* dengan PostgreSQL

PostgreSQL merupakan sebuah sistem manajemen *database*, atau juga dapat disebut sebagai DBMS (*Database Management System*). DBMS itu sendiri merupakan software yang bertujuan untuk mengendalikan pembuatan, pemeliharaan, pengolahan, dan penggunaan data yang berskala besar. PostgreSQL merupakan sistem manajemen basis data objek-relasional.

D. Visual Studio Code

Visual Studio Code (VSCode) adalah *code editor* yang dikembangkan Microsoft. Visual Studio Code adalah editor code yang mendukung lingkungan JavaScript dan Python. Visual Studio Code juga memiliki banyak fitur yang mendukung pengembangan *backend*.

E. Github untuk *Continous Integration/ Continous Deployment* (CI/CD)

Salah satu aspek penting dalam pengembangan aplikasi adalah penerapan strategi *version control* dan Continuous Integration/Continuous Deployment (CI/CD), dapat dilakukan menggunakan GitHub. Github memiliki fitur bernama Github Actions untuk melakukan CI/CD sesuai Langkah dan konfigurasi yang diinginkan pengguna.

III. METODE

Metode pengembangan aplikasi dimulai dengan melakukan perancangan fitur, perancangan arsitektur aplikasi, perancangan database, kemudian dilanjutkan dengan implementasi.

A. Perancangan Fitur Aplikasi

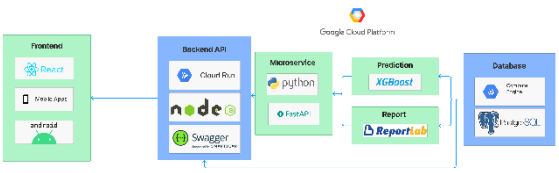
Fungsi dari aplikasi ini mengikuti hasil *interview* yang dilakukan dengan pihak manajemen, berikut adalah fungsi dari aplikasi.

Tabel 1 Fitur aplikasi

No	Fitur
1	Authentication
2	Pencatatan Transaksi
3	Laporan
4	Mengatur Manajemen
5	Analisis Data dan Prediksi

Fitur ini kemudian akan dikembangkan kedalam bentuk *backend* dengan menggunakan perancangan arwsitektur

B. Arsitektur Aplikasi

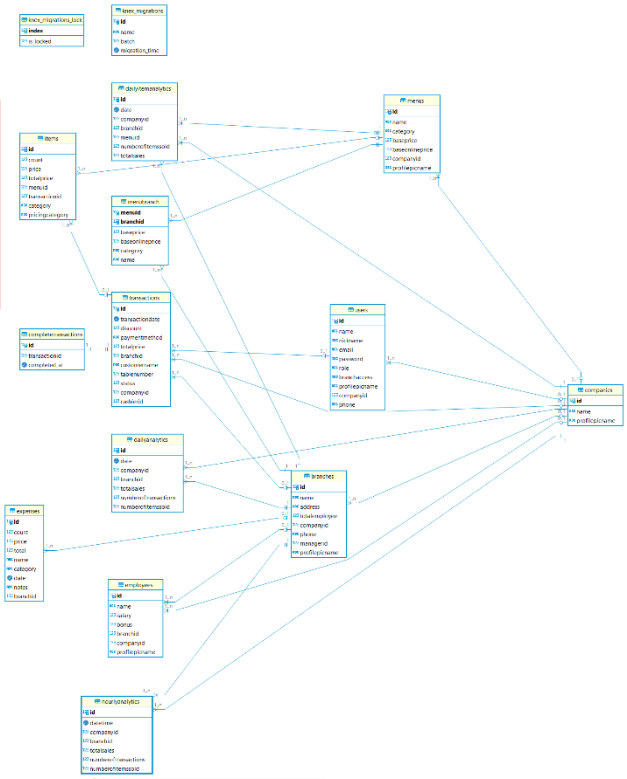


Gambar 1 Arsitektur aplikasi

Sistem Informasi ini menggunakan *front-end* dari *framework* react-native yang dikembangkan di android. Backend terdapat dua *Application Programming Interface* (API). API utama menggunakan node.js, sedangkan API *microservice* menggunakan python. *Database* menggunakan PostgreSQL. *Backend* dan *Database* akan di-*deploy* di Google Cloud Platform, menggunakan Compute Engine dan Cloud Run.

C. Perancangan Database

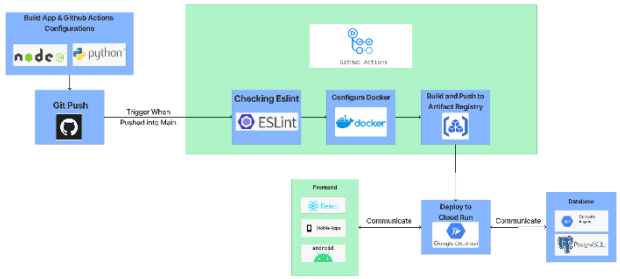
Perancangan Database aplikasi dilakukan dengan membuat *Entity Relationship Diagram* (ERD). Berikut adalah *Entity Relationship Diagram* yang dibuat.



Gambar 2 Entity Relationship Diagram

Database pada aplikasi ini memiliki 13 Tabel Utama dan 2 tabel pembantu untuk migrasi. Tabel Companies akan menyimpan setiap Perusahaan yang terdaftar. Tabel ini akan memiliki hubungan dengan tabel *Branches* yang akan menyimpan cabang. Tabel cabang akan memiliki hubungan dengan menu yang akan memiliki kemungkinan perbedaan pada cabang. Tabel ini juga akan memiliki hubungan dengan *Employees*. Terdapat juga tabel *Users* yang akan menyimpan seluruh pengguna dan akses yang mereka miliki. Transaksi akan disimpan pada tabel *Transactions* dan *Items*. Tabel *Expenses* digunakan untuk mencatat pengeluaran. Terdapat juga tabel yang berfungsi untuk *analytics* dan *trigger* yaitu pada *completetransactions*, *hourlyanalytics*, *dailyanalytics*, dan *dailyitemsanalytics*.

D. Deployment API



Gambar 3 Deployment API

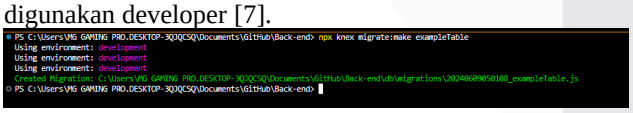
Proses deployment API untuk aplikasi "Elevate Eats" menggunakan strategi modern dan otomatisasi yang efektif melalui penerapan *Continuous Integration/Continuous Deployment* (CI/CD) dengan menggunakan GitHub Actions. Di awal, kode sumber yang dikembangkan menggunakan Node.js dan Python di-push ke repositori GitHub. Setiap kali ada perubahan pada branch utama, GitHub Actions akan memicu serangkaian operasi otomatis. Pertama, ESLint akan memeriksa kode untuk memastikan semua standar penulisan kode terpenuhi, mencegah potensi bug dan meningkatkan kualitas kode secara keseluruhan.

Setelah kode terverifikasi memenuhi standar *linting*, proses berlanjut ke pembuatan *container* menggunakan Docker. Container ini kemudian akan dibangun dan gambar yang dihasilkan disimpan dalam Artifact Registry. Langkah ini memastikan bahwa semua dependensi dan konfigurasi aplikasi terkemas dengan benar, memudahkan pengelolaan versi dan distribusi.

Tahap akhir dalam *pipeline* ini adalah deployment aplikasi yang sudah ter-containerisasi ke Google Cloud Run. Pendekatan ini tidak hanya meningkatkan keandalan dan kecepatan deployment tetapi juga meminimalisir kesalahan manusia, mendukung aplikasi "Elevate Eats" untuk beroperasi dengan lebih efisien dalam lingkungan produksi.

E. Implementasi *Database*

Pengembangan skema *database* pada aplikasi Elevate Eats menggunakan skema *database* yang dibentuk dalam skema migrasi di knex. Pembuatan skema migrasi dapat dilakukan dengan menggunakan *command line interface* (CLI) yang merupakan perintah berbasis teks yang banyak digunakan developer [7].



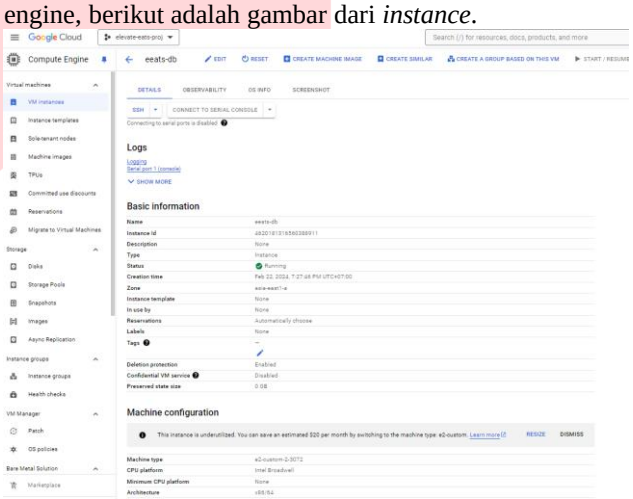
Gambar 4 proses migrasi skema melalui CLI

Gambar tersebut merupakan proses migrasi skema melalui CLI.

Kode 1 migrasi skema

```
/* eslint-disable func-names */
/**
 * @param { import("knex").Knex } knex
 * @returns { Promise<void> }
 */
exports.up = function (knex) {
  return
knex.schema.createTable('companies',
(table) => {
    table.increments('id').primary();
    table.string('name',
100).unique();
    table.string('profilepicname');
  });
};
/**
 * @param { import("knex").Knex } knex
 * @returns { Promise<void> }
 */
exports.down = function (knex) {
  return
knex.schema.dropTableIfExists('compani
es');
};
```

Kode tersebut adalah contoh pembuatan skema migrasi database menggunakan knex. Setelah skema sudah selesai dibuat, database akan di deploy pada *instance* compute engine, berikut adalah gambar dari *instance*.



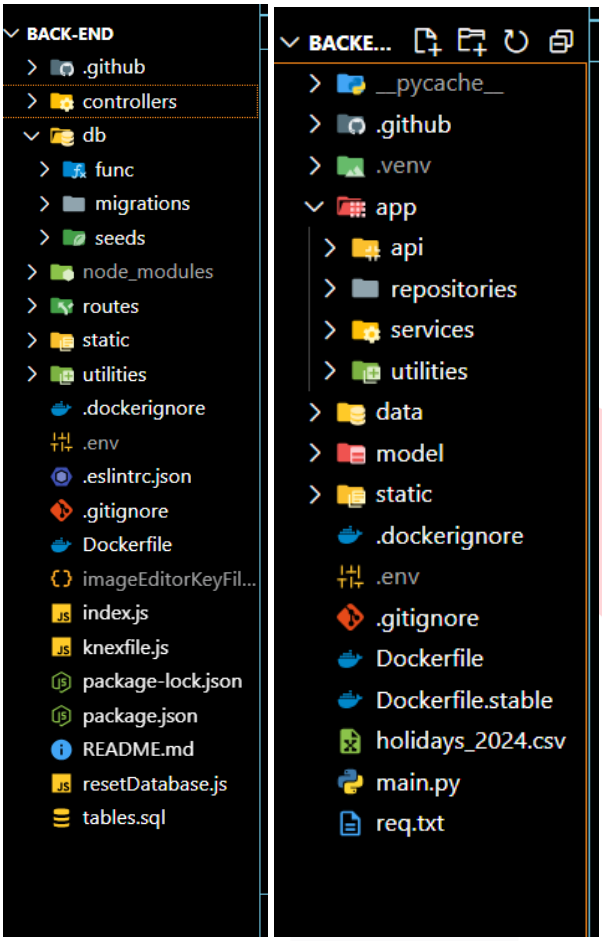
Gambar 5 instance

F. Implementasi API

Sub-sistem API menggunakan node.js dan python. *Framework* yang digunakan adalah express.js dan fastapi. *Deployment* akan dilakukan dengan Github Actions yang memungkinkan *deployment* dengan konsep *Continous Integration*/*Continous Development* (CI/CD). API akan menggunakan aturan linting dari ESLint untuk menjaga kerapian code. Setelah pengecekan linting, API akan dimuat dalam container menggunakan docker, yang akan disimpan kedalam artifact registry untuk di-*deploy* ke Google Cloud Run.

Implementasi API pada Elevate Eats terbagi menjadi dua, yaitu API utama yang menggunakan kerangka kerja Express.JS pada node.js. API utama merupakan api yang terhubung langsung dengan front-end. Selain API utama, terdapat juga Microservice yang dibangun menggunakan kerangka kerja FastAPI pada Python. *Microservice* pada aplikasi ini merupakan API yang menjalankan model Machine Learning yang melakukan prediksi jumlah transaksi dan total pendapatan restoran serta membuat laporan dalam bentuk pdf secara otomatis menggunakan data yang terdapat pada *database*. API juga menggunakan swagger untuk dokumentasi API. API juga menggunakan kontrol versi Github yang disimpan dalam repositori organisasi Elevate-Eats.

Berikut adalah hasil kode dari *backend*. Terdapat dua repositori, pertama untuk API utama, kedua untuk API *Microservice*.



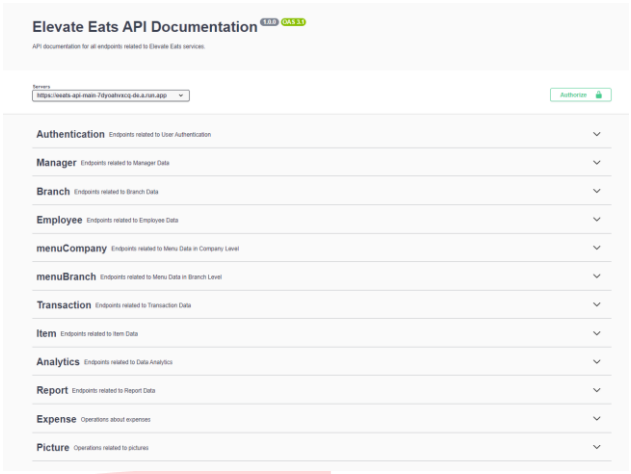
Gambar 6 hasil kode dari backend

Kode API utama terdiri atas *controller*, fungsi basis data yang memiliki fungsi migrasi, rute, dan fungsi pembantu. Titik awal kode terletak di index.js. Kode ini menggunakan node package manager (npm) sebagai *package manager* kode. Kode API Microservice terdiri atas api yang menyimpan rute, repositori untuk penghubung ke database, *service* untuk menyimpan servis yang ada, dan fungsi pembantu. *Package manager* yang digunakan adalah pip.

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengembangan API

Hasil pembuatan API akan terlihat melalui dokumentasi. Berikut adalah dokumentasi API aplikasi.



Gambar 7 documentation endpoint

API memiliki 12 kelompok *endpoint* dan 57 total *endpoint*. Kelompok ini terdiri dari *authentication*, *manager*, *branch*, *employee*, *menucompany*, *menubranch*, *transaction*, *item*, *analytics*, *report*, *expense*, *picture*.API fitur akan terkunci menggunakan JSON Web Token yang dibuat saat user melakukan *login*. API akan mengirimkan hasil JSON sesuai dengan status yang ada. Berikut adalah contoh JSON hasil API.

```
{
  "error": false,
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "credentials": {
    "id": "123"
  }
}
```

Gambar 8 contoh JSON hasil API.

JSON respon ini terdiri atas kondisi *error*, *message* atau pesan API, dan data yang akan dikirimkan sesuai dengan fitur.

Deployment Aplikasi menggunakan proses (CI/CD). Proses integrasi berkelanjutan dan penyebaran berkelanjutan (CI/CD) untuk aplikasi "Elevate Eats" dilakukan melalui penggunaan GitHub *Actions*, yang mengotomatisasi *workflow* dari push kode hingga *deployment* ke Google Cloud Run. Workflow ini dimulai dengan trigger pada setiap push ke cabang utama yang disebut 'main'. Dalam *job* yang bernama 'Setup, Build, Publish, and Deploy', yang berjalan pada sistem operasi terbaru Ubuntu, terdapat beberapa langkah penting yang dilakukan untuk menyiapkan dan melakukan *deploy* aplikasi. Berikut adalah potongan kode awal persiapan *deployment*:

Kode 2 potongan kode awal persiapan deployment

```
name: Deploy to Google Cloud Run with
Artifact Registry
on:
  push:
    branches:
      - main
jobs:
  setup-build-publish-deploy:
```

Pada CI/CD ini akan mencakup *deploy* ke Google Cloud Run dengan Artifact Registry seperti terlihat pada kode diatas.

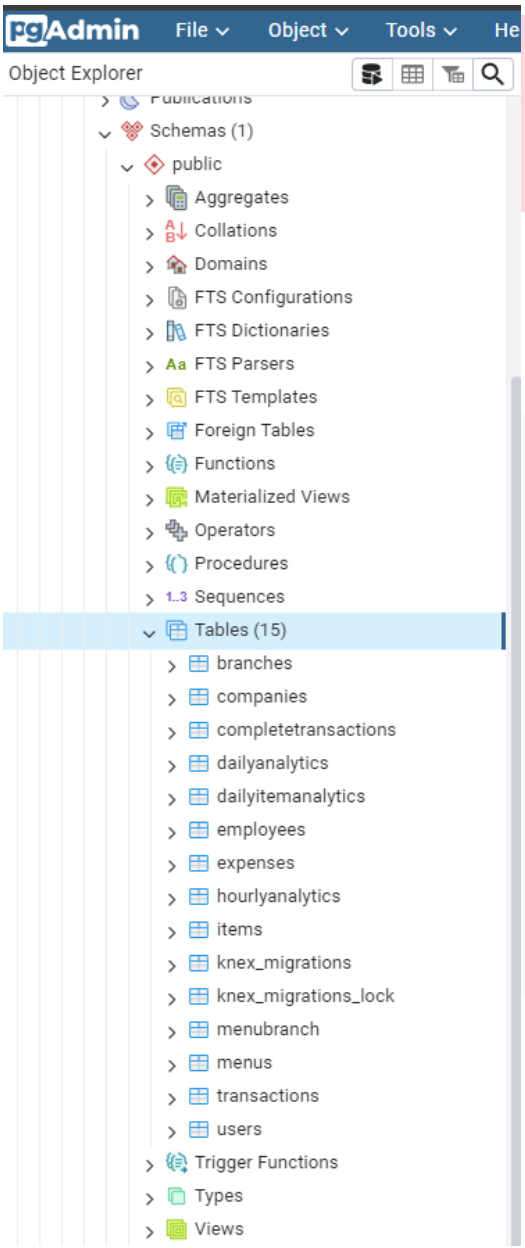
B. Hasil Pengembangan Database

Hasil pengembangan database melalui beberapa tahap pengembangan, pengembangan pertama adalah pembuatan server database, berikut adalah *container* server pada *virtual machine*.

```
eeats@eeats-db:~/DB-Config$ docker compose ps
NAME                IMAGE                                COMMAND                  SERVICE
Wireguard-EEats-DB  lscr.io/linuxserver/wireguard:latest  "/init"                  Wireguard
db-config-db-1      postgres:15                          "docker-entrypoint.s…"   db
db-config-pgadmin-1 dpage/pgadmin4                       "/entrypoint.sh"         pgadmin
```

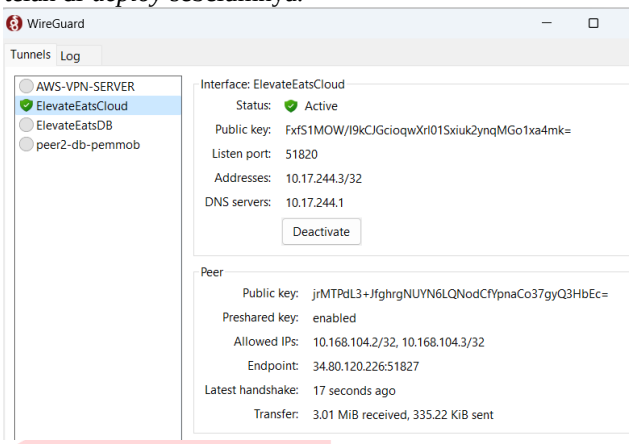
Gambar 9 container server pada virtual machine.

Diatas merupakan beberapa bagian dari pengembangan database, terdapat *server* postgresql, *virtual private network* (VPN) wireguard untuk keamanan pada pengembangan, dan PGAdmin untuk manajemen basis data. Hasil berikutnya adalah hasil pengembangan tabel *database* yang dapat dilihat pada *database management* PGAdmin.



Gambar 10 hasil database

Dari gambar diatas didapatkan seluruh tabel telah berhasil dibuat, 13 tabel utama dan 2 tabel untuk migrasi. Tabel ini dapat dilihat dan diatur pada halaman *webpage* PGAdmin. PGAdmin ini dapat diakses melalui VPN Wireguard yang telah di-*deploy* sebelumnya.

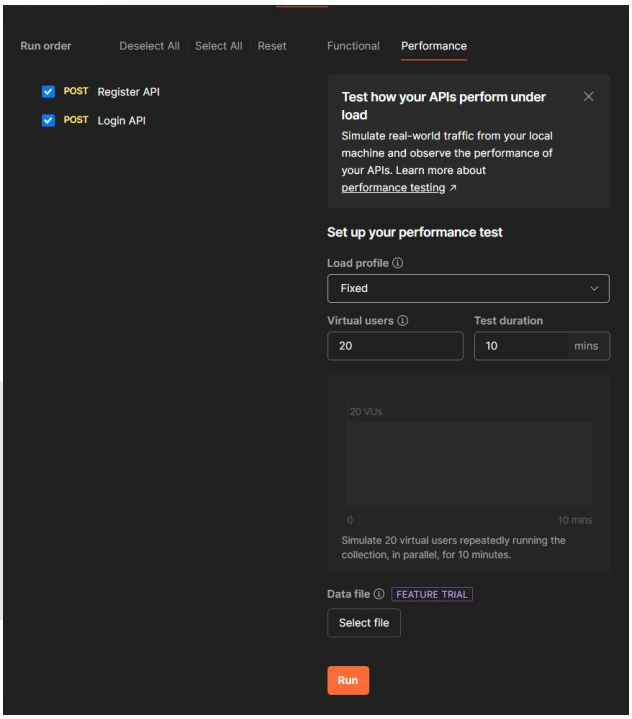


Gambar 11 tampilan wireguard

Apabila wireguard sudah dikonfigurasi sesuai dengan VPN *server*, maka *database* dan PGAdmin dapat diakses untuk keperluan development. Sedangkan, untuk akses database pada API, menggunakan IP Public yang tersedia.

C. Pengujian *Stress Test*

Pengujian API akan melalui pengujian *Stress Test* yang akan dilakukan di platform Postman. Pengujian akan dilakukan untuk 5, 20, hingga 50 pengguna. Berikut adalah konfigurasi postman untuk *Stress Test*



Gambar 12 konfigurasi postman untuk Stress Test

Pengujian *Stress Test* pada postman dapat mengatur jumlah user dan durasi, pada *Stress Test* kali ini, dapat diatur jumlah user dan durasi dari stress test, durasi stress test adalah 1 menit.

Tabel 2 Pengujian Stress Test

No	Feature	User	Developer	Selisih
1	Registration	110 detik	60 detik	+50 detik

2	Login	47 detik	20 detik	+27 detik
3	Manager	131 detik	86 detik	+45 detik
4	Branch	71 detik	56 detik	+15 detik
5	Employees	58 detik	43 detik	+15 detik
6	Transaction	34 detik	40 detik	-6 detik
7	Predictions	17 detik	18 detik	-1 detik
8	Analytics	21 detik	16 detik	+5 detik
9	Expenses	43 detik	30 detik	+13 detik
10	Daily Reports	16 detik	13 detik	+3 detik

V. KESIMPULAN

Penelitian ini berhasil mengembangkan sebuah sistem informasi manajemen restoran yang terintegrasi dan berbasis digital, yang secara signifikan meningkatkan efisiensi operasional dan kemampuan analisis data di CV Balibul. Dengan transisi dari sistem berbasis kertas yang rentan terhadap kesalahan manusia ke sistem digital yang canggih, manajemen CV Balibul kini dapat mengambil keputusan yang lebih informasi dan tepat berdasarkan data real-time dan analisis yang akurat. Backend aplikasi, yang dibangun menggunakan JavaScript dan Python, telah terbukti reliable dan efektif, memberikan performa tinggi meskipun terhubung dengan banyak pengguna secara simultan. Kesuksesan implementasi ini juga menunjukkan pentingnya integrasi teknologi modern seperti Node.js, Express.js, Python, dan FastAPI dalam pengembangan sistem informasi manajemen. Untuk pengembangan lebih lanjut, direkomendasikan untuk mengintegrasikan kapabilitas machine learning untuk mendukung prediksi dan analisis data yang lebih canggih, yang akan lebih memaksimalkan potensi sistem dalam mendukung pengambilan keputusan strategis di CV Balibul.

REFERENSI

[1] J. Suprihanto, *MANAJEMEN*, 1st ed. Yogyakarta: GADJAH MADA UNIVERSITY PRESS, 2018.

[2] H. A. Rusdiana, M. M. Moch, S. T. Irfan, M. Kom, and H. M. A. Ramdhadi, *Sistem Informasi Manajemen*. 2018.

[3] S. Arifin *et al.*, *Sistem Informasi Manajemen*, 1st ed. Padang: Global Eksekutif Teknologi, 2023.

[4] R. Horntvedt and T. Akesson, *Java, Python and Javascript, a comparison*, vol. 1. 2019.

[5] E. Brown, *Web Development with Node and Express*, vol. 2. 2019.

[6] J. Haro Peralta, *Microservice APIs*, 1st ed. New York: Manning Publication, 2023.

[7] S. bin Uzayr, N. Cloud, and T. Ambler, “Knex and Bookshelf,” in *JavaScript Frameworks for Modern Web Development*, Berkeley, CA: Apress, 2019, pp. 377–426. doi: 10.1007/978-1-4842-4995-6_10.