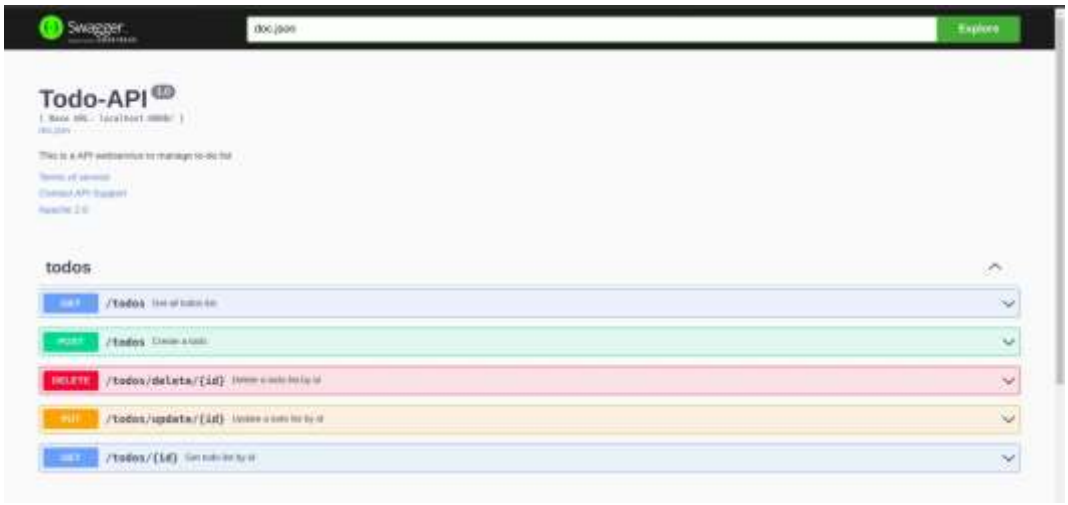
BAB V

IMPLEMENTASI DAN PENGUJIAN SISTEM

5.1 HASIL IMPLEMENTASI

1. Hasil Final Project 1 : To-Do List

Hasil dari Final Project 1 adalah sebuah prototype Todo list API tanpa database yang dilengkapi dengan dokumentasi, berikut adalah gambaran endpoint yang disediakan :



Gambar 5. 1 Dokumentasi Todo API menggunakan Swagger

Pada projek pertama ini sendiri hanya memiliki satu *endpoint* yaitu *todos* yang menggunakan empat *method* sesuai dengan tugas yang diberikan antara lain sebagai berikut:

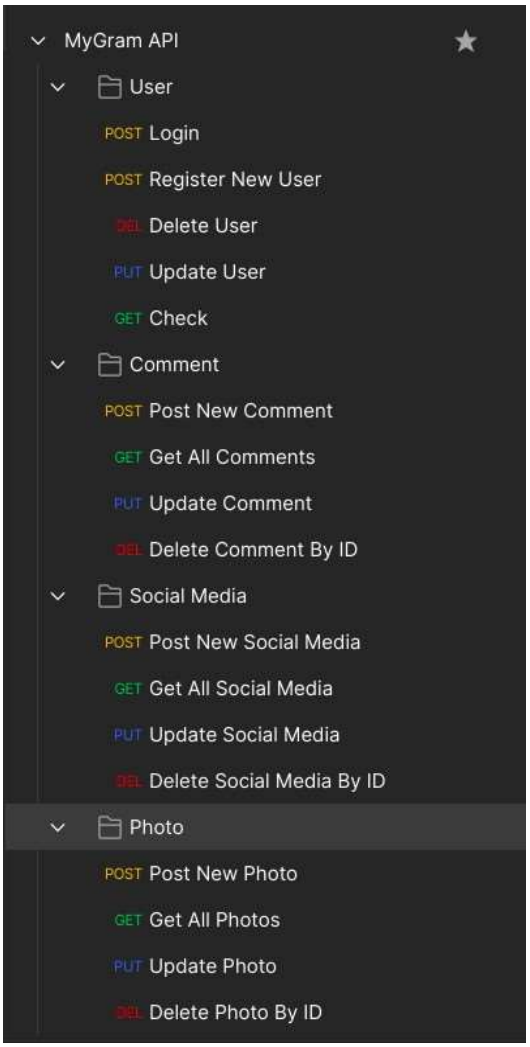
1. GET, pada *method* ini bertujuan untuk mengambil semua data yang telah disimpan dalam *array* sebelumnya yang dapat diolah datanya.
2. POST, pada *method* ini bertujuan untuk membuat data berdasarkan data yang dikirimkan yang akan disimpan dalam suatu *array*.

3. DELETE pada *path* delete, pada *method* ini bertujuan untuk menghapus data yang disimpan dalam *array* berdasarkan *id* yang disertakan dalam *path* tautan.
4. PUT pada *path* update, pada *method* ini bertujuan untuk meletakkan data yang baru kedalam *array* berdasarkan *id* yang disertakan dalam *path* tautan, dimana data yang lama akan diganti dengan data yang baru.
5. GET, pada *method* ini lebih dikhususkan dari pada *method* GET sebelumnya dimana data yang diambil dari *array* diambil berdasarkan *id* pada *path* tautan.

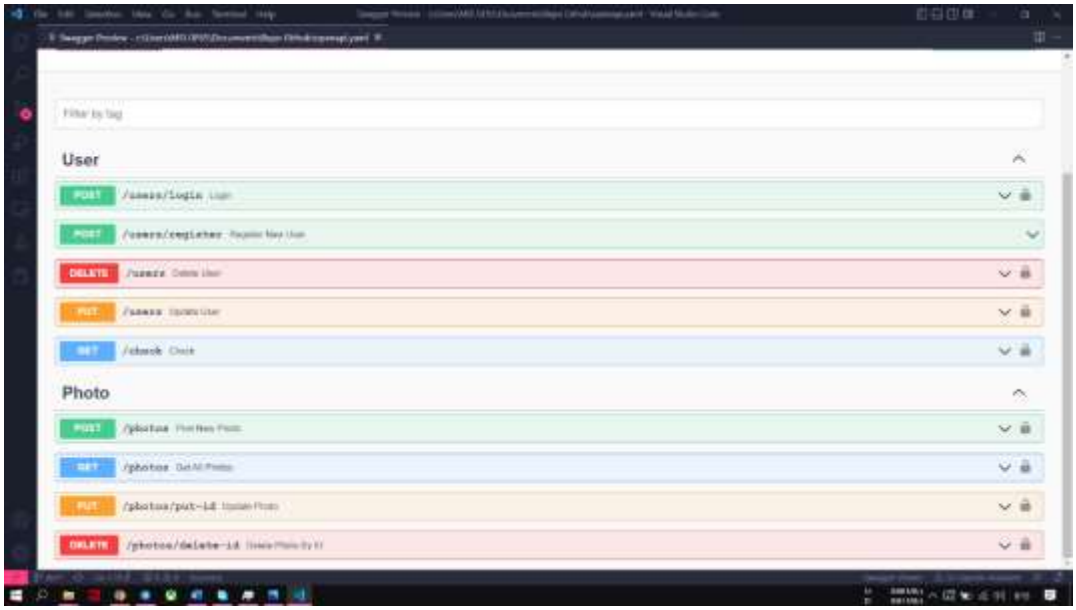
Semua *method* yang dilakukan akan mengirimkan kembali response dalam bentuk JSON, yang berupa pesan apakah method berhasil dilakukan. Sumber kode dari program dapat dilihat melalui situs Github berikut https://github.com/rifqoi/todos-api-go/

2. Hasil Final Project 2 : MyGram

Hasil dari final project 2 ini adalah sebuah API bernama MyGram yang bisa diakses secara publik, berikut adalah gambaran *endpoint* nya:



Gambar 5. 2 Kumpulan End Point Final Project 2 (Bagian 1)



Gambar 5. 3 Kumpulan End Point Final Project 2 (Bagian 2)

Pada projek kedua ini memiliki empat *endpoint* antara lain yaitu *users, photos, comments,* dan *socialmedia.* yang menggunakan beberapa *method* yang akan melewati *authorization* pada bagian *middleware*, proses autentikasi menggunakan JWT atau JSON Web Token dan semua data disimpan dalam database menggunakan PostgreSQL, kemudian sesuai dengan tugas yang diberikan *method* yang digunakan antara lain sebagai berikut:

a. Users

1. POST pada *path* register, pada *method* ini bertujuan untuk melakukan pendaftaran pengguna kedalam database dengan data yang digunakan antara lain *age, email, username,* dan *password*, *password* tersebut nantinya akan di*hash* sebelum disimpan kedalam database menggunakan *bcrypt*.
2. POST pada *path* login, pada *method* ini bertujuan untuk bergabung kedalam aplikasi menggunakan data yang digunakan

yaitu *email* dan *password*, jika data yang diberikan ada dalam database maka akan memberikan respon berupa token JWT.

3. PUT, pada *method* ini bertujuan untuk memperbarui data yang telah disimpan dalam database berdasarkan user id pada parameter, data berupa *email* dan *password*, serta *headers* yang memiliki token JWT yang telah dibuat sebelumnya untuk proses autentikasi.
4. DELETE, pada *method* ini bertujuan untuk menghapus user yang telah dibuat sebelumnya dari database dengan headers yang berisikan token JWT sebagai metode untuk autentikasi kedalam aplikasi.

b. Photos

1. POST, pada *method* ini bertujuan untuk mengirimkan data berupa *title, caption,* dan *photo_url* kedalam database dengan melewati autentikasi terlebih dahulu menggunakan token JWT.
2. GET, pada *method* ini bertujuan untuk mengambil semua data yang telah disimpan kedalam database dalam bentuk *array* sebagai respon yang dikembalikan dan diperlukan autentikasi terlebih dahulu menggunakan token JWT.
3. PUT, pada *method* ini bertujuan untuk meletakkan data yang baru kedalam database berdasarkan photo id yang ada pada *path* tautan dengan data baru berupa *title, caption,* dan *photo_url* dan diperlukan autentikasi terlebih dahulu menggunakan token JWT.

4. DELETE, pada *method* ini bertujuan untuk menghapus data dari database berdasarkan photo id pada *path* tautan dan diperlukan autentikasi terlebih dahulu menggunakan token JWT.

c. Comments

1. POST, pada *method* ini bertujuan untuk menyimpan suatu komentar berdasarkan data yang digunakan yaitu *message*, dan *photo_id* yang berasal dari *endpoint* Photos kedalam database melalui autentikasi terlebih dahulu menggunakan token JWT.
2. GET, pada *method* ini bertujuan untuk mengambil semua data yang disimpan dari database dalam bentuk *array* dan diperlukan autentikasi terlebih dahulu menggunakan token JWT.
3. PUT, pada *method* ini bertujuan untuk meletakkan komentar baru kedalam database berdasarkan comment id yang ada pada *path* tautan dengan autentikasi terlebih dahulu menggunakan token JWT.
4. DELETE, pada *method* ini bertujuan untuk menghapus salah satu komentar yang dibuat dalam database menggunakan comment id pada *path* tautan dengan autentikasi terlebih dahulu menggunakan token JWT.

d. Social Media

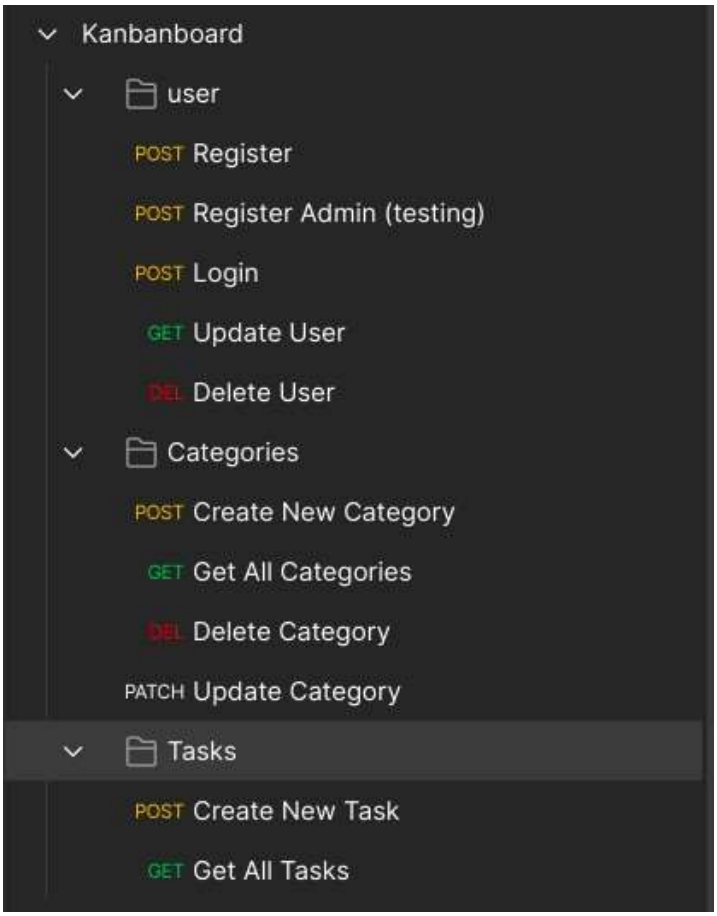
1. POST, pada *method* ini bertujuan untuk menyimpan data kedalam database berupa *name* dan *social_media_url* dan diperlukan autentikasi terlebih dahulu menggunakan token JWT.

2. GET, pada *method* ini bertujuan untuk mengambil semua data yang disimpan dalam database dalam bentuk array melalui autentikasi terlebih dahulu menggunakan token JWT.
3. PUT, pada method ini bertujuan untuk meletakkan data baru berupa kedalam database berdasarkan social media id yang ada pada path tautan dengan autentikasi terlebih dahulu menggunakan token JWT.
4. DELETE, pada *method* ini bertujuan untuk menghapus salah satu data yang dibuat dalam database menggunakan social media id pada *path* tautan dengan autentikasi terlebih dahulu menggunakan token JWT.

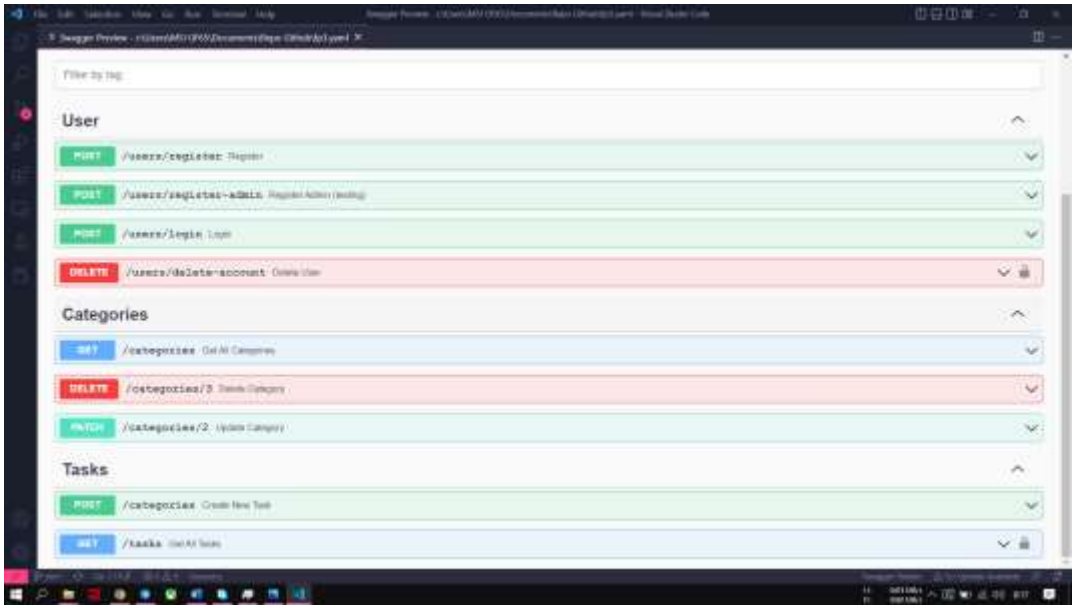
Sumber kode dari program dapat dilihat melalui situs Github berikut https://github.com/jusidama18/mygram-api-go. Situs aplikasi juga disebarkan melalui Railway melalui *link* berikut https://mygram-api-goproduction.up.railway.app/

3. Hasil Final Project 3 : Kanban Board

Hasil dari final project 3 ini adalah sebuah API bernama KanbanBoard yang bisa diakses secara publik, berikut adalah gambaran endpoint nya:



Gambar 5. 4 Kumpulan End Point Final Project 3 (Bagian 1)



Gambar 5. 5 Kumpulan End Point Final Project 3 (Bagian 2)

Pada projek ketiga ini memiliki tiga *endpoint* antara lain yaitu *users, categories,* dan *tasks.* menggunakan beberapa *method* yang akan melewati *authorization* pada bagian *middleware*, proses autentikasi menggunakan JWT atau JSON Web Token dan semua data disimpan dalam database menggunakan PostgreSQL, kemudian sesuai dengan tugas yang diberikan *method* yang digunakan antara lain sebagai berikut:

a. Users

1. POST pada *path* register, pada *method* ini bertujuan untuk melakukan pendaftaran pengguna yang akan diberikan role berupa *member*. Kemudian data akan disimpan kedalam database dengan data yang digunakan antara lain *full_name, email,* dan *password*. Sebelum disimpan kedalam database, data akan di*hash* menggunakan *bcrypt*.

2. POST pada *path* login, pada *method* ini bertujuan untuk bergabung kedalam aplikasi menggunakan data yang digunakan yaitu *email* dan *password*, jika data yang diberikan berhasil divalidasi dan data ditemukan dalam database maka akan memberikan respon berupa token JWT.
3. PUT pada *path* update-account, pada *method* ini bertujuan untuk memperbarui data yang telah disimpan dalam database, data berupa *email* dan *password*, serta *headers* yang memiliki token JWT yang telah dibuat sebelumnya untuk proses autentikasi.
4. DELETE pada *path* delete-account, pada *method* ini bertujuan untuk menghapus user yang telah dibuat sebelumnya dari database dengan headers yang berisikan token JWT sebagai metode untuk autentikasi kedalam aplikasi.

b. Categories

1. POST, pada *method* ini bertujuan untuk menyimpan data kedalam database berupa *type* melalui autentikasi menggunakan token JWT pada *headers* saat melakukan *request*. Metode ini bertujuan untuk membuat kategori dari tugas yang akan dibuat.
2. GET, pada *method* ini bertujuan untuk mengambil semua data dari database dalam bentuk *array* dimana dalam kategori terdapat *array* yang berisikan *task* yang telah dibuat, diperlukan autentikasi menggunakan token JWT saat proses *request*.

3. PATCH, pada *method* ini bertujuan untuk menempelkan atau memperbarui data yang telah ada dalam database melalui category id pada *path* tautan, proses autentikasi token JWT diperlukan saat melakukan *request* serta data yang diperbarui berupa *type* dari kategori.
4. DELETE, pada *method* ini bertujuan untuk menghapus data yang telah disimpan dalam database menggunakan category id pada *path* tautan dengan melalui proses autentikasi sebelumnya menggunakan token JWT.

c. Tasks

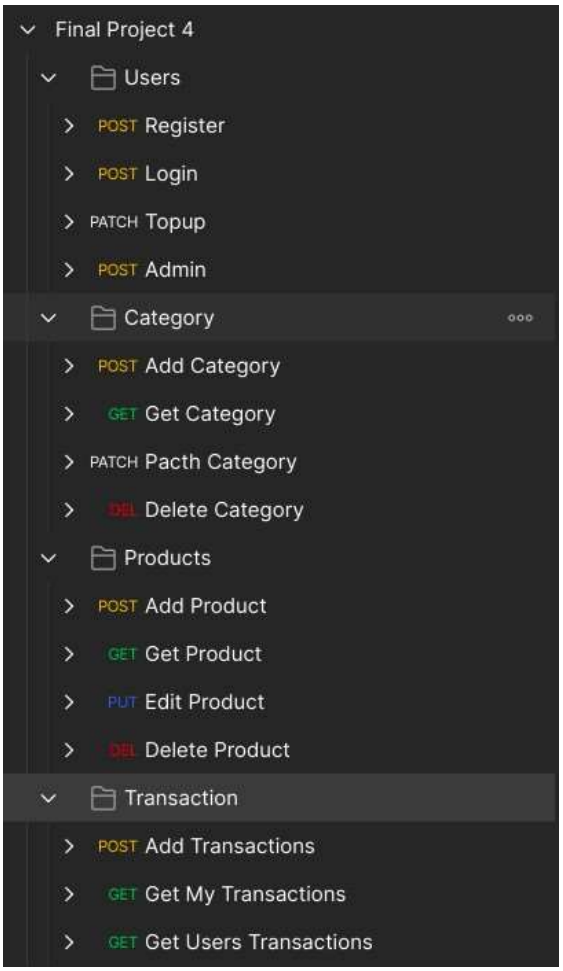
1. POST, pada *method* ini bertujuan untuk menyimpan data kedalam database berupa *title* dan *description* serta category id berdasarkan di kategori mana data akan dikelompokkan, dalam proses *request* diperlukan autentikasi melalui token JWT.
2. GET, pada *method* ini bertujuan untuk mengambil data dari database dalam bentuk *array* yang terdiri dari *task* yang ada pada semua kategori disertai informasi *user* yang memiliki *task* tersebut. Diperlukan autentikasi dalam proses *request* menggunakan token JWT.
3. PUT, pada *method* ini bertujuan untuk meletakkan data yang baru pada database berdasarkan task id yang ada pada *path* tautan yang akan mengubah *task* dari suatu kategori, dan diperlukan juga autentikasi dengan token JWT saat proses *request*.

4. PATCH pada *path* update-status, pada *method* ini bertujuan untuk memperbarui suatu status dalam *task* bernilai boolean yang disimpan dalam database berdasarkan task id yang ada pada *path* tautan, dalam proses *request* diperlukan autentikasi menggunakan token JWT.
5. PATCH pada *path* update-category, pada *method* ini bertujuan untuk memperbarui suatu kategori dalam *task* bernilai integer yang disimpan dalam database berdasarkan task id yang ada pada *path* tautan, dalam proses *request* diperlukan autentikasi menggunakan token JWT.
6. DELETE, pada *method* ini bertujuan untuk menghapus suatu *task* yang disimpan dalam database menggunakan task id pada *path* tautan dengan autentikasi token JWT saat melakukan *request*.

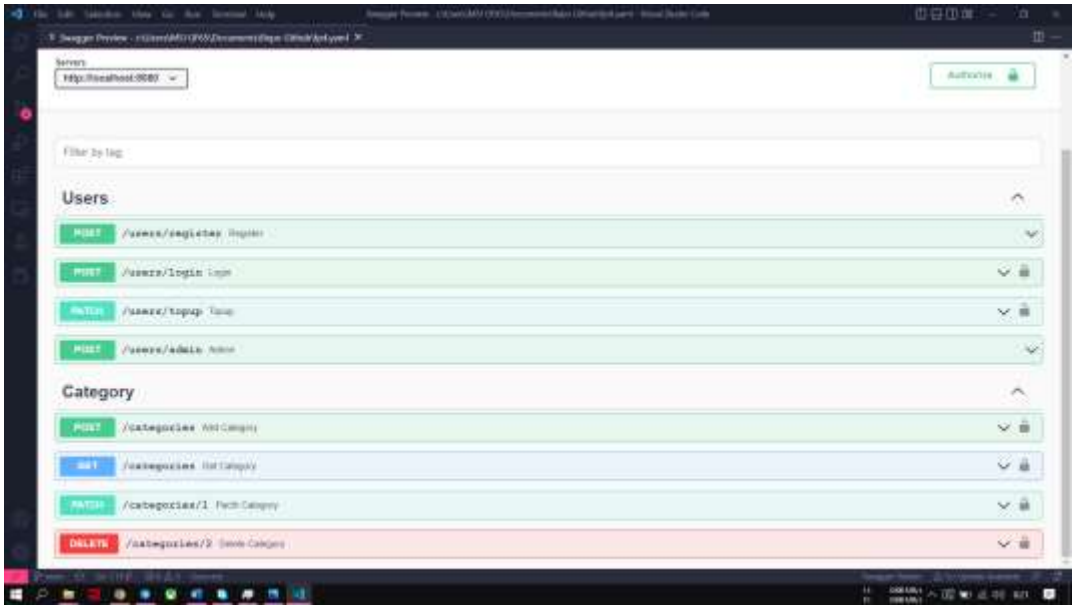
Sumber kode dari program dapat dilihat melalui situs Github berikut https://github.com/DionFauzi/fp-Kanbanboard. Situs aplikasi juga disebarkan melalui Railway melalui *link* berikut https://fp-kanbanboardproduction.up.railway

4. Hasil Final Project 4 : Toko Belanja

Hasil dari final project 4 ini adalah sebuah API bernama Toko-Belanja yang bisa diakses secara publik, berikut adalah gambaran endpoint nya:



Gambar 5. 6 Kumpulan End Point Final Project 4 (Bagian 1)



Gambar 5. 7 Kumpulan End Point Final Project 4 (Bagian 2)

Pada projek keempat ini memiliki empat *endpoint* antara lain yaitu *users, categories, products,* dan *transactions,* menggunakan beberapa *method* yang akan melewati *authorization* pada bagian *middleware*, proses autentikasi menggunakan JWT atau JSON Web Token dan semua data disimpan dalam database menggunakan PostgreSQL, kemudian sesuai dengan tugas yang diberikan *method* yang digunakan antara lain sebagai berikut:

a. Users

1. POST pada *path* register, pada *method* ini bertujuan untuk melakukan pendaftaran pengguna yang akan diberikan role berupa *member*. Kemudian data akan disimpan kedalam database dengan data yang digunakan antara lain *full_name, email,* dan *password*. Sebelum disimpan kedalam database, data akan di*hash* menggunakan *bcrypt*.

2. POST pada *path* login, pada *method* ini bertujuan untuk bergabung kedalam aplikasi menggunakan data yang digunakan yaitu *email* dan *password*, jika data yang diberikan berhasil divalidasi dan data ditemukan dalam database maka akan memberikan respon berupa token JWT.
3. PATCH pada *path* topup, pada *method* ini bertujuan untuk menambahkan atau mengubah data berupa *balance* dari *user* kemudian dilakukan pengecekkan *balance* yang akan diubah selanjutnya akan disimpan dalam database, saat proses *request* diperlukan autentikasi terlebih dahulu menggunakan token JWT.
4. POST pada *path* admin, pada *method* ini bertujuan untuk mendaftarkan suatu *role* admin kedalam database dengan autentikasi terlebih dahulu menggunakan token JWT.

b. Categories

1. POST, pada *method* ini bertujuan untuk membuat suatu kategori berdasarkan data yang diberikan yaitu *type* dimana kategori tersebut akan disimpan dalam database dan melewati proses autentikasi token JWT saat melakukan *request*.
2. GET, pada *method* ini bertujuan untuk mengambil data dari database berupa *array* kategori beserta *array* dari produk yang menggunakan kategori tersebut didalamnya. Diperlukan autentikasi terlebih dahulu menggunakan token JWT.

3. PATCH, pada *method* ini bertujuan untuk memperbarui data yang disimpan dalam database berdasarkan category id yang ada pada *path* tautan dengan melewati proses autentikasi token JWT terlebih dahulu.
4. DELETE, pada *method* ini bertujuan untuk menghapus data yang telah disimpan dalam database berdasarkan category id yang ada pada *path* tautan dan diperlukan autentikasi terlebih dahulu menggunakan token JWT.

c. Products

1. POST, pada *method* ini bertujuan untuk membuat suatu produk dengan data berupa *title, price, stock,* dan *category_id* yang mana category id digunakan untuk mengelompokkan kemana produk akan disimpan dalam database serta diperlukan autentikasi terlebih dahulu dengan token JWT.
2. GET, pada *method* ini bertujuan untuk mengambil semua data produk dari database dan mengembalikan *array* berisikan daftar produk yang telah disimpan sebelumnya, pada *method* ini tidak diperlukan autentikasi menggunakan token JWT.
3. PUT, pada method ini bertujuan untuk memperbarui data suatu produk berdasarkan product id yang ada pada *path* tautan kemudian data yang telah diubah disimpan dalam database, diperlukan proses autentikasi menggunakan token JWT saat proses *request*.

4. DELETE, pada *method* ini bertujuan untuk menghapus suatu produk yang disimpan dalam database menggunakan product id pada *path* tautan dengan melewati proses autentikasi terlebih dahulu menggunakan token JWT.

d. Transactions

1. POST, pada *method* ini bertujuan untuk menyimpan data berupa *quantity* dan *product_id* yang mana digunakan sebagai proses perhitungan total jumlah produk yang telah dipesan berdasarkan product id dengan melewati proses autentikasi menggunakan token JWT.
2. GET pada *path* my-transactions, pada *method* ini bertujuan untuk menampilkan semua data transaksi dari database berupa *array* yang terdiri dari jumlah harga, jumlah produk, dan daftar produk yang telah disimpan sebelumnya beserta data ketersediaan produk yang ada berdasarkan *user_id* milik pengguna, *method* ini hanya menampilkan data transaksi serta data produk milik pengguna, diperlukan autentikasi menggunakan token JWT saat proses *request.*
3. GET pada *path* pada user-transactions, pada *method* ini bertujuan untuk menampilkan seluruh transaksi dari user dalam bentuk *array* yang diambil dari database sebelumnya yang berisikan data produk, data user serta data dari masing-masing transaksi

yang telah disimpan sebelumnya, diperlukan autentikasi menggunakan token JWT saat proses *request.*

Sumber kode dari program dapat dilihat melalui situs Github berikut https://github.com/DionFauzi/Fp-TokoBelanja. Situs aplikasi juga disebarkan melalui Railway melalui *link* berikut https://fp-tokobelanja-productionb9b7.up.railway.app/

5.2 PENGUJIAN SISTEM

Pada tahap ini penulis mencoba menguji berbagai komponen dari game yang dibuat. Pengujian ini dijalankan dengan menggunakan metode black box, yaitu suatu cara untuk melakukan pengujian fungsional tanpa mempertimbangkan bagaimana struktur dibuat dalam fungsi tersebut [27]. Metode Black Box ini hanya menguji semua fungsi yang ada pada program. Misalnya, jika Anda memberikan input ke suatu program, itu akan diuji untuk melihat apakah outputnya seperti yang diharapkan. Berbeda dengan Metode White Box, metode ini berfokus pada pengujian sistem yang lebih dalam, seperti pada bagian bahasa pemrograman. Metode White Box membutuhkan pengetahuan mendalam tentang bahasa pemrograman untuk menjalankan pengujian, sedangkan Metode Black Box tidak memerlukan pengetahuan mendalam tentang bahasa pemrograman untuk menjalankan pengujian, tetapi metode ini membutuhkan pengetahuan tentang apa yang dilakukan sistem sudah sesuai yang diharapkan [28].

Modul yang diuji	**Cara Pengujian**	**Input**	**Output Yang Diharapkan**	**Output Yang Didapatkan**	**Kesimpulan**
Final Project 1 : To-Do List	Melakukan Request menggunakan setiap	Data berdasarkan dokumen	Menampilkan response sukses dengan status	Menerima response sukses dengan	Berhasil

	Endpoint dengan metode GET, POST, DELETE, Dll	tasi yang telah dibuat	code 200 / 201	output sesuai serta status code 200 / 201	
Final Project 2 : MyGram	Melakukan Request menggunakan setiap Endpoint dengan metode GET, POST, DELETE, Dll	Data berdasarkan dokumentasi yang telah dibuat	Menampilkan response sukses dengan status code 200 / 201	Menerima response sukses dengan output sesuai serta status code 200 / 201	Berhasil
Final Project 3 : Kanban Board	Melakukan Request menggunakan setiap Endpoint dengan metode GET, POST, DELETE, Dll	Data berdasarkan dokumentasi yang telah dibuat	Menampilkan response sukses dengan status code 200 / 201	Menerima response sukses dengan output sesuai serta status code 200 / 201	Berhasil

Final Project 4 : Toko Belanja	Melakukan Request menggunakan setiap Endpoint dengan metode GET, POST, DELETE, Dll	Data berdasarkan dokumentasi yang telah dibuat	Menampilkan response sukses dengan status code 200 / 201	Menerima response sukses dengan output sesuai serta status code 200 / 201	Berhasil

Tabel 5. 1 Tabel Hasil Pengujian Final Project

5.3 ANALISIS HASIL SISTEM

Di bagian ini penulis akan menjelaskan hasil yang telah dicapai dari penelitian ini. Hasil yang telah dicapai dipaparkan dalam bentuk kelebihan dan kekurangan dari keempat web service yang telah dibuat dan disajikan dalam bentuk daftar.

5.3.1 KELEBIHAN WEB SERVICE

Adapun kelebihan dari pembuatan keempat web service yang telah dibuat dalam *Final Project* dengan menghasilkan REST API menggunakan bahasa Go adalah sebagai berikut :

1. Web service ini memiliki dokumentasi yang mudah dibaca menggunakan Swagger dalam pembuatannya.
2. Web service ini telah memiliki keamanan yang memadai menggunakan JSON Web Token (JWT) sebagai metode autentikasi dalam proses *request*.

3. Web service ini mampu digunakan dalam pengembangan suatu website sebagai bagian *backend* program.
4. Web service ini telah menggunakan *database* yang membuat data tersimpan secara persisten.

5.3.2 KEKURANGAN WEB SERVICE

Adapun kelebihan dari pembuatan keempat web service yang telah dibuat dalam *Final Project* dengan menghasilkan REST API menggunakan bahasa Go adalah sebagai berikut :

1. Web service ini belum memiliki kemampuan untuk menerima pengiriman data dalam bentuk file atau dokumen.
2. Web service ini masih cukup sederhana dalam pembuatannya.
3. Web service ini belum menerapkan *Rate Limit* yang membatasi banyaknya jumlah *request* dalam satu waktu.
4. Web service ini masih banyak bagian yang perlu dikembangkan untuk kemudahan penggunaannya.