Pengembangan API Perusahaan HiColleagues pada Modul Master Data dengan Metode Monolitik

Kevin Laurence[#1], Risal[*2]

[#]*Program Studi Teknik Informatika Fakultas Teknologi Informasi, Universitas Kristen Maranatha*
Jalan Prof. Drg. Surya Sumantri No.65, Bandung 40164
[1]2072030@maranatha.ac.id
[2]laurentius.risal@it.maranatha.edu

Abstract — **The rapid development of information technology and business has transformed the interaction between companies and users. A user can access to product information through mobile applications, websites, and social media has significantly increased. HiColleagues is currently facing challenges in business development sector. In response to the challenges of business growth, HiColleagues has developed the API HiColleagues website. The primary focus of this development is on the master data module API. The goal of API development in the master data module is to automate the addition of new data and manage it efficiently. Additionally, the development of the API in the master data module aims to ensure that crucial data (master data) within HiColleagues is always up-to-date. The development of the API use Go programming language, with Echo framework, and PostgreSQL as the database. The API in the master data module includes several developed features, such as banner, career, class, FAQ, instructor, and voucher. This development can assist HiColleagues in addressing the challenges of business growth and meeting the company's needs.**

Keywords— ***API, HiColleagues, Master Data, Manage Data.***

I. PENDAHULUAN

Perkembangan teknologi informasi dan dunia bisnis yang semakin pesat telah mengubah interaksi antara perusahaan dengan pengguna secara signifikan. Perubahan ini mencakup peningkatan aksesibilitas pengguna terhadap informasi, produk, dan layanan perusahaan. Sebagai contoh, pengguna dapat mengakses berbagai informasi secara cepat tentang produk, harga, dan ulasan melalui aplikasi seluler, situs web, atau platform media sosial.

Saat ini, perusahaan HiColleagues sedang dihadapi tantangan modern terkait dengan pertumbuhan bisnis yang cepat. Pertumbuhan ini disebabkan oleh permintaan pengguna yang semakin banyak. Dengan pertumbuhan yang cepat, data-data yang akan diterima oleh perusahaan dapat menjadi banyak dan sulit untuk dikelola. Dalam upaya untuk menyelesaikan permasalahan tersebut, perusahaan HiColleagues mendapati bahwa perlunya sebuah pendekatan baru terhadap pengelolaan data ini agar dapat merespons perubahan yang cepat dalam permintaan pengguna dan mengembangkan produk baru secara efisien dan maksimal.

Tujuan pembahasan dari penelitian ini adalah mengembangkan API HiColleagues pada modul master data dengan metode monolitik yang bertujuan untuk mengotomatisasi penambahan data baru dan dapat mengelola data yang dibutuhkan perusahaan HiColleagues. Selain itu, pengembangan API pada modul master data bertujuan agar dapat memastikan bahwa data penting (master data) dalam perusahaan HiColleagues selalu *up-to-date*. Pengembangan ini menggunakan metode monolitik, yaitu sebuah metode yang digunakan dalam pengembangan perangkat lunak untuk menggambarkan pembangunan sistem perangkat lunak sebagai satu kesatuan, dalam artian tidak terpisah.

II. KAJIAN TEORI

A. Go

Bahasa Go atau Golang merupakan bahasa pemrograman yang dikembangkan oleh tiga orang yang bekerja di Google pada tahun 2007, yaitu Rob Pike, Robert Griesemer, dan Ken Thompson [1]. Tujuan bahasa pemrograman ini dibuat adalah menjadikan bahasa pemrograman yang efisien dalam melakukan kompilasi ataupun eksekusi, dan efektif dalam menulis program yang andal. Go merupakan turunan dari bahasa pemrograman C, yang membuat Go mempunyai keunggulan dari sisi kecepatan dan kesederhanaan.

Bahasa pemrograman Go memiliki fitur konkurensi yang efisien dan pendekatannya terhadap abstraksi data dan pemrograman berorientasi objek juga sangat fleksibel. Go sangat cocok digunakan untuk membangun infrastruktur seperti server yang terhubung jaringan, situs web, dan perangkat lunak lainnya. Program-program Go biasanya berjalan lebih cepat daripada program yang

ditulis dalam bahasa pemrograman yang dinamis, dan mengalami jauh lebih sedikit kegagalan karena kesalahan tipe yang tidak terduga [2].

B. Echo

Echo merupakan salah satu *framework* bahasa pemrograman Golang yang digunakan untuk pengembangan aplikasi web. *Framework* Echo memiliki performa yang tinggi, *extensible*, dan minimalis sehingga mempermudah pengembangan aplikasi web dengan cepat, efisien, dan sederhana. *Framework* Echo memiliki beberapa kelebihan seperti *router* yang sederhana, didukung dengan teknologi HTTP/2, *scalable*, *data binding*, *data rendering*, dan *middleware* [3].

C. PostgreSQL

PostgreSQL adalah sebuah basis data *Object Relational Database Management System* (ORDBMS) yang bersifat *open source*. PostgreSQL dapat dijalankan di sebagian besar sistem operasi modern, seperti Windows, Mac, dan berbagai varian Linux. PostgreSQL juga salah satu perangkat lunak basis data yang banyak digunakan dan dapat bersaing dengan beberapa basis data. Selain itu, ada beberapa *extension* untuk mengakses, mengelola, dan memonitor data pada PostgreSQL, seperti pgAdmin 4. Instalasi dan konfigurasi PostgreSQL cukup mudah karena didukung oleh sebagian besar *packaging tools*, seperti *yum* dan *apt* [4].

D. API

Application Programming Interface (API) adalah sebuah protokol atau mekanisme yang memungkinkan berbagai aplikasi yang berbeda itu saling terhubung atau berkomunikasi.. API berfungsi sebagai jembatan penghubung antar perangkat lunak yang memungkinkan mereka dapat berkomunikasi dan berbagi data. API dapat digunakan dalam pengembangan situs web, aplikasi *mobile*, dan integrasi perangkat lunak. API menggunakan pemanggilan fungsi melalui *Hyper Text Transfer Protocol* (HTTP) dan mendapatkan respons berupa *Extensible Markup Language* (XML) atau *JavaScript Object Notation* (JSON) [5].

E. REST API

Representational State Transfer (REST) adalah sebuah gaya pendekatan komunikasi yang digunakan dalam proses pengembangan aplikasi [6]. REST API banyak digunakan dalam pengembangan web dan aplikasi *mobile* karena sifatnya yang sederhana, fleksibel, dan dapat melakukan transmisi data melalui protokol HTTP. REST API bersifat *stateless*, yang artinya server tidak akan menyimpan informasi data *client* ketika melakukan permintaan. REST API sendiri memiliki empat metode utama yaitu GET untuk mengambil data, POST untuk membuat data baru, PUT untuk memperbarui data, dan DELETE untuk menghapus data dari server [7].

F. JSON

JavaScript Object Notation (JSON) adalah sebuah format berbasis teks yang digunakan untuk menyimpan dan mengirimkan data. JSON sendiri berfungsi sebagai cara yang sederhana dalam melakukan penyimpanan dan pengiriman data karena ringan dan mudah dibaca dengan menggunakan struktur *key-value pair* (pasangan kunci-nilai). JSON umumnya digunakan dalam pengembangan situs web dan API untuk mengirim dan menerima data yang terstruktur. JSON juga tidak bergantung pada bahasa pemrograman apa pun, menjadikan JSON sebagai format yang fleksibel karena dapat digunakan pada semua bahasa pemrograman [8].

G. Postman

Postman adalah sebuah aplikasi yang digunakan untuk mengembangkan, menguji, dan mendokumentasikan suatu API. Postman menyediakan fitur berupa permintaan HTTP (GET, POST, PUT, DELETE, dan lain-lain) [9]. Postman juga memiliki fitur pengujian API yang terotomatisasi, simulasi *endpoint* secara langsung, pemantauan performa dan waktu respons dari sebuah API. Postman dapat digunakan juga untuk membuat dokumentasi API yang lengkap, seperti contoh permintaan, deskripsi *endpoint*, dan informasi terkait lainnya [10].

H. Monolitik

Monolitik merupakan sebuah pendekatan arsitektur perangkat lunak di mana aplikasi yang dibangun dibungkus dan dijadikan sebagai satu kesatuan yang besar. Dalam arsitektur monolitik, semua komponen dan fungsi aplikasi terintegrasi menjadi satu *package* tunggal yang berjalan dalam satu proses. Kelebihan dari arsitektur monolitik ini adalah proses pengembangan dan *debugging code* lebih mudah. Karena semua komponen terkait dalam satu unit, proses *deployment* aplikasi dengan pendekatan monolitik jauh lebih sederhana [11].

I. JSON Web Token (JWT)

JSON Web Token (JWT) adalah sebuah standar terbuka (RFC 7519) yang mendefinisikan cara aman transfer informasi antar pihak dalam bentuk objek JSON. JWT digunakan untuk mengotentikasi dan mengotorisasi pengguna atau sistem secara

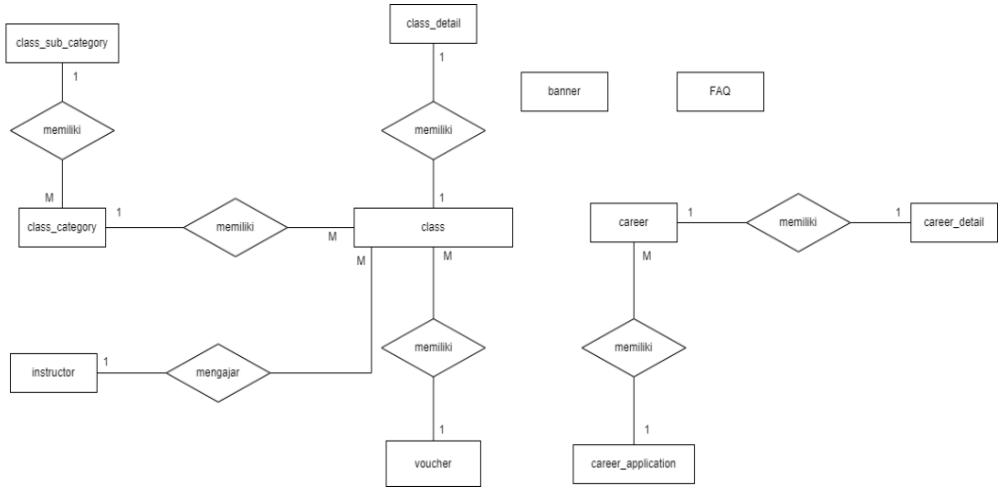
terdistribusi. JWT sangat berguna dalam skenario autentikasi di mana kredensial atau *token* pengguna perlu disertakan bersama dengan setiap permintaan. Kredensial atau *token* ini biasanya digunakan sebagai mekanisme autentikasi di aplikasi web dan layanan daring [12].

J. Clean Architecture

Clean Architecture adalah suatu pendekatan arsitektur dalam pengembangan perangkat lunak yang bertujuan untuk menciptakan struktur yang terorganisir, mudah dipahami, dan terpisah-pisah. *Clean Architecture* bertujuan agar membantu *developer* untuk melakukan pemeliharaan, pengujian, dan pengelolaan sebuah aplikasi. Selain itu, *developer* dapat menambahkan atau memperbarui fitur dalam aplikasi tanpa merusak bagian lainnya [13].

III. ANALISIS DAN RANCANGAN

A. Entity Relationship Diagram (ERD)



Gambar 1. Entity Relationship Diagram API

Gambar 1 merupakan *Entity Relationship Diagram* (ERD) dari API yang dibuat. Tabel *class* memiliki hubungan *many-to-one* dengan *class category*, *voucher*, dan *instructor*. Tabel *class category* memiliki hubungan *many-to-one* dengan *class sub category*. Tabel *class* juga memiliki hubungan *one-to-one* dengan tabel *class detail*. Selain itu, terdapat juga tabel *career* yang memiliki hubungan *one-to-one* dengan *career detail* dan *many-to-one* dengan *career application*. Untuk tabel *banner* dan FAQ tidak memiliki hubungan dengan tabel lain dan kedua tabel tersebut berdiri sendiri.

B. Kebutuhan Tabel Entitas

Pada bagian *requirement* ini, akan dijelaskan mengenai kebutuhan tabel entitas apa saja yang digunakan dalam pengembangan proyek ini. Terdapat beberapa tabel entitas yang akan dijelaskan berikut ini.

1) Tabel Banner

Tabel I menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas Banner.

TABEL I
TABEL ENTITAS BANNER

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
name	varchar	
img_link_url_web	varchar	
img_link_url_mobile	varchar	
link_url	varchar	
is_active	boolean	
description	varchar	
category	varchar	
created_at	timestamptz	not null

Nama Field	Tipe Data	Keterangan
updated_at	timestamptz	not null
deleted_at	integer	not null
is_deleted	boolean	not null

2) Tabel Career

Tabel II menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Career*.

TABEL II
TABEL ENTITAS CAREER

Nama Field	Tipe Data	Keterangan
id	uuid	primary key, not null
slug_name	varchar	
position	varchar	
company	varchar	
location	varchar	
contract_type	varchar	
work_type	varchar	
level	varchar	
started_at	date	
closed_at	date	
is_active	boolean	
created_at	timestamptz	not null
updated_at	timestamptz	not null
deleted_at	integer	not null
is_deleted	boolean	not null

3) Tabel Career Application

Tabel III menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Career Application*.

TABEL III
TABEL ENTITAS CAREER APPLICATION

Nama Field	Tipe Data	Keterangan
id	uuid	primary key, not null
career_id	uuid	foreign key career, not null
full_name	varchar	
email	varchar	
phone_number	varchar	
linkedin_url	varchar	
resume_url	varchar	
portfolio_url	varchar	
created_at	timestamptz	not null
updated_at	timestamptz	not null
deleted_at	integer	not null
is_deleted	boolean	not null

4) Tabel Career Detail

Tabel IV menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Career Detail*.

TABEL IV
TABEL ENTITAS CAREER DETAIL

Nama Field	Tipe Data	Keterangan
id	uuid	primary key, not null
career_id	uuid	foreign key career, not null
overview	text	
responsibilities	jsonb	
requirements	jsonb	
benefits	jsonb	

5) Tabel Class

Tabel V menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Class*.

TABEL V
TABEL ENTITAS CLASS

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
name	varchar	
image_url	varchar	
description	varchar	
level	varchar	
class_category_id	integer	foreign key class_category
class_sub_category_id	integer	foreign key class_sub_category
created_at	timestamptz	not null
updated_at	timestamptz	not null
deleted_at	integer	not null
is_deleted	boolean	not null
slug_name	varchar	
thumbnail_url	varchar	
small_image_url	varchar	
learning_link	varchar	
consultancy_link	varchar	
consultancy_schedule	varchar	
group_chat_link	varchar	
instructor_id	integer	foreign key instructor
material	varchar	
is_active	boolean	
method	varchar	
meta_description	varchar	
is_main	boolean	

6) Tabel Class Category

Tabel VI menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Class Category*.

TABEL VI
TABEL ENTITAS CLASS CATEGORY

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
name	varchar	
created_at	timestamptz	not null
updated_at	timestamptz	not null

7) Tabel Class Sub Category

Tabel VII menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Class Sub Category*.

TABEL VII
TABEL ENTITAS CLASS SUB CATEGORY

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
class_category_id	integer	foreign key class_category
name	varchar	
created_at	timestamptz	not null
updated_at	timestamptz	not null

8) Tabel Class Detail

Tabel VIII menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Class Detail*.

Tabel VIII
TABEL ENTITAS CLASS DETAIL

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
class_id	integer	foreign key class
price	integer	
discount_price	integer	
session_total	integer	
days	varchar	
hours	varchar	
learning_objective	varchar	
is_discount	varchar	
created_at	timestamptz	not null
updated_at	timestamptz	not null
deleted_at	integer	not null
is_deleted	boolean	not null
syllabus	varchar	
target_audience	jsonb	
profession	jsonb	
benefits	jsonb	
start_discount_date	timestamp	
end_discount_date	timestamp	
summary_syllabus	jsonb	

9) Tabel FAQ

Tabel IX menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas FAQ.

TABEL IX
TABEL ENTITAS FAQ

Nama Field	Tipe Data	Keterangan
id	uuid	primary key, not null
question	varchar	
answer	text[]	
category	varchar	
created_at	timestamptz	
updated_at	timestamptz	
deleted_at	integer	not null
is_deleted	boolean	not null

10) Tabel Instructor

Tabel X menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Instructor*.

TABEL X
TABEL ENTITAS INSTRUCTOR

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
name	varchar	
image_url	varchar	
description	varchar	
linkedin_url	varchar	
cv_url	varchar	
position	varchar	
company	varchar	
created_at	timestamptz	not null
updated_at	timestamptz	not null
deleted_at	integer	not null

11) Tabel Voucher

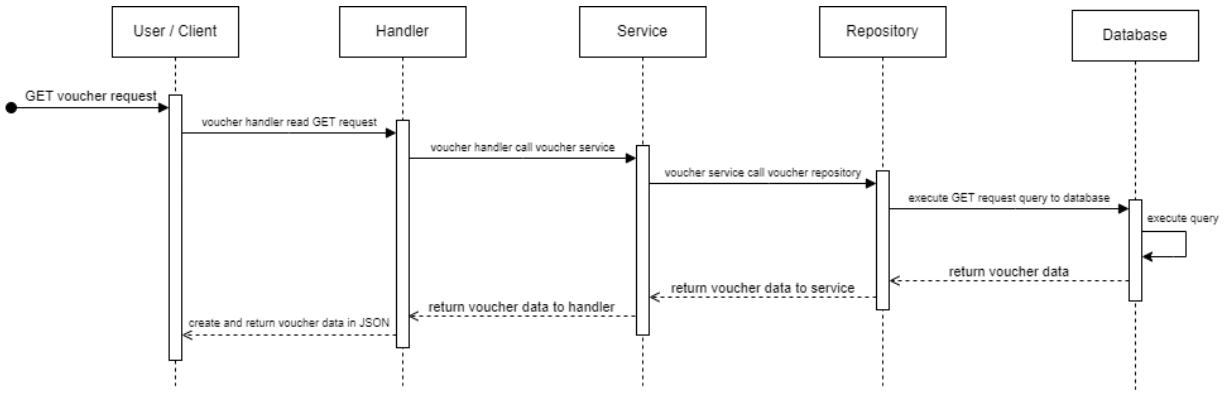
Tabel XI menjelaskan mengenai daftar dan deskripsi atribut yang terdapat dalam entitas *Voucher*.

TABEL XI
TABEL ENTITAS VOUCHER

Nama Field	Tipe Data	Keterangan
id	integer	primary key, not null
name	varchar	
image_url	varchar	
description	varchar	
voucher_code	varchar	
discount	integer	
stock	integer	
start_date	timestamptz	
end_date	timestamptz	
is_active	boolean	
created_at	timestamptz	not null
updated_at	timestamptz	not null
deleted_at	integer	not null
is_deleted	boolean	not null

IV. IMPLEMENTASI

A. Alur Pemanggilan API



Gambar 2. Sequence Diagram Call API GET Voucher

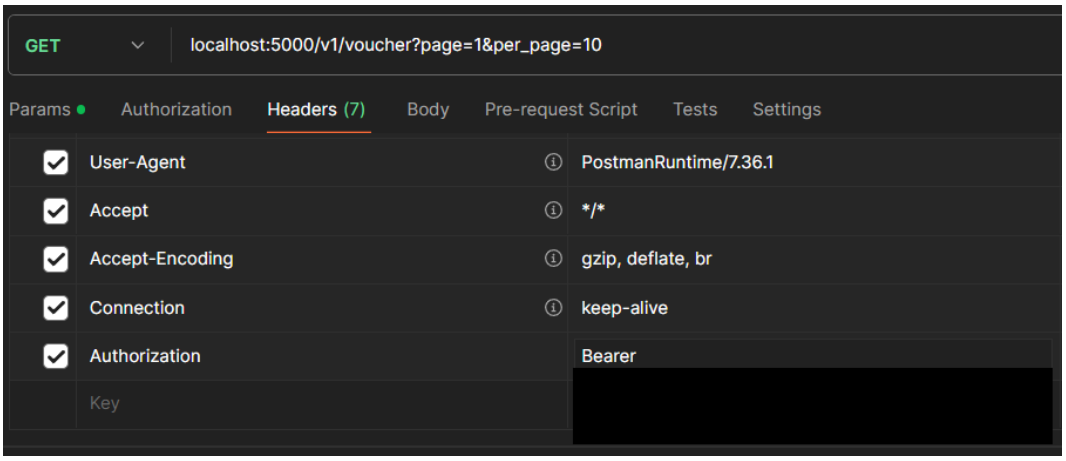
Gambar 2 mendefinisikan alur pemanggilan sebuah endpoint GET voucher pada API, dimulai dari pengguna melakukan GET *voucher request*. Setelah itu, permintaan GET *voucher* tersebut akan diterima oleh bagian *Handler*. Pada bagian ini, *handler* dapat melakukan validasi *input*, memanggil *voucher service*, dan menyiapkan respons yang sesuai dengan *request* yang diminta.

Setelah itu, *voucher handler* akan meneruskan GET *voucher request* dan memanggil *voucher service* melalui *voucher interface service*, yang di mana *interface service* ini berfungsi sebagai kontrak yang mendefinisikan operasi atau metode yang dapat diakses oleh *handler* dari *service*. *Service* bertugas untuk menjalankan logika bisnis aplikasi. *Service* ini juga dapat memanipulasi data, melakukan operasi terkait bisnis, dan mempersiapkan data yang diperlukan untuk respons.

Setelah itu, *voucher service* akan meneruskan *request* dan memanggil *voucher repository* melalui *voucher interface repository*, dengan tujuan yang sama seperti *voucher interface service*. *Voucher repository* akan berinteraksi dengan basis data atau penyimpanan data untuk melakukan operasi *query* sesuai fungsi yang diminta. *Query* yang diminta dalam contoh tersebut adalah pengambilan data voucher dari basis data. *Voucher repository* juga akan memetakan data ke dalam bentuk domain jika terdapat data yang dikembalikan dari basis data.

Setelah data *voucher* diolah pada bagian *voucher repository*, data tersebut akan dikembalikan ke bagian *voucher service*, dan *voucher service* juga akan meneruskan data *voucher* tersebut ke *voucher handler*. *Voucher Handler* akan menyiapkan sebuah respons apakah data *voucher* yang dikembalikan tersebut berhasil atau gagal. Jika gagal, *handler* akan membuat sebuah respons *error*, dan jika berhasil, *handler* akan membuat sebuah respons *success*. Respons tersebut yang akan ditampilkan pada pengguna yang memanggil sebuah API.

B. Autentikasi API



Gambar 3. Headers Authorization Postman

Setiap pemanggilan API dilakukan, sistem aplikasi akan melakukan pengecekan autentikasi apakah pengguna yang memanggil API tersebut dapat melakukan pemanggilan API dan mendapatkan data atau tidak. Setiap *endpoint* dalam API ini menggunakan autentikasi *JSON Web Token* (JWT). Tujuan dilakukan autentikasi agar tidak sembarang pengguna yang melakukan pemanggilan API dapat melihat atau memanipulasi data dalam basis data.

Pengguna harus *login* terlebih dahulu dan ketika *login* sudah berhasil, maka pengguna akan mendapatkan token JWT. Token tersebut akan dimasukkan ke dalam *Headers* dalam aplikasi Postman ketika ingin memanggil sebuah *endpoint* API. Pada *Headers*, masukkan "*Authorization*" sebagai *key* (di kiri), dan untuk tokennya menggunakan kata "*Bearer*" di depannya dan tambahkan token setelah kata tersebut.

C. Fitur Banner

Fitur *Banner* adalah fitur yang berfungsi untuk mengelola data *Banner* pada API HiColleagues. *Banner* adalah sebuah media promosi yang berisikan gambar untuk mengenalkan atau menawarkan sesuatu yang ada di HiColleagues. Fitur *Banner* yang dikembangkan memiliki fungsi *Create Banner, Get Banners, Get Banner By ID, Update Banner,* dan *Delete Banner.*

D. Fitur Career

Fitur *Career* adalah fitur yang berfungsi untuk mengelola data *Career* pada API HiColleagues. *Career* adalah sebuah program yang ditawarkan oleh HiColleagues bagi orang yang ingin masuk ke dalam dunia karier. Fitur *Career* yang dikembangkan memiliki fungsi *Create Career, Get Careers, Get Career By ID, Update Career, Delete Career,* dan *Get Career By Slug Name.*

E. Fitur Career Application

Fitur *Career Application* adalah fitur yang berfungsi untuk mengelola data *Career Application* pada API HiColleagues. *Career Application* adalah data pengguna yang mendaftar ke program *Career* di HiColleagues. Fitur *Career Application* yang

dikembangkan memiliki fungsi *Get Career Application, Get Career Application By ID, Delete Career Application,* dan *Update Career Application.*

F. Fitur Career Detail

Fitur *Career Detail* adalah fitur yang berfungsi untuk mengelola data *Career Detail* pada API HiColleagues. *Career Detail* adalah data yang lebih detail dari tabel *Career*. Fitur *Career Detail* yang dikembangkan hanya *Get Career Detail By ID*, dikarenakan untuk *Create, Update,* dan *Delete Career Detail* itu sudah bersamaan dengan fitur *Career*.

G. Fitur Class

Fitur *Class* adalah fitur yang berfungsi untuk mengelola data *Class* pada API HiColleagues. *Class* adalah sebuah program *bootcamp* yang ditawarkan oleh HiColleagues bagi orang yang ingin memiliki *skill* tertentu dalam dunia kerja. Fitur *Class* yang dikembangkan adalah *Create Class, Get Classes, Get Class By ID, Get Class By Category, Update Class,* dan *Delete Class.*

H. Fitur Class Category

Fitur *Class Category* adalah fitur yang berfungsi untuk mengelola data *Class Category* pada API HiColleagues. *Class Category* adalah data kategori untuk tabel *Class*. Fitur *Class Category* yang di kembangkan adalah *Create Class Category, Get Class Categories, Get Class Category By ID, Update Class Category,* dan *Delete Class Category*.

I. Fitur Class Sub Category

Fitur *Class Sub Category* adalah fitur yang berfungsi untuk mengelola data *Class Sub Category* pada API HiColleagues. Class *Sub Category* adalah data sub kategori yang merupakan turunan dari tabel *Class Category*. Fitur *Class Sub Category* yang dikembangkan adalah *Create Class Sub Category, Get Class Sub Categories, Get Class Sub Category By ID, Update Class Sub Category* dan *Delete Class Sub Category*.

J. Fitur Class Detail

Fitur *Class Detail* adalah fitur yang berfungsi untuk mengelola data *Class Detail* pada API HiColleagues. *Class Detail* adalah data yang lebih detail dari tabel *Class*. Fitur *Class Detail* yang dikembangkan hanya *Get Class Detail By ID*, dikarenakan untuk *Create, Update,* dan *Delete Class Detail* itu sudah bersamaan dengan fitur *Class*.

K. Fitur FAQ

Fitur FAQ adalah fitur yang berfungsi untuk mengelola data *Frequently Asked Question* (FAQ) pada API HiColleagues. FAQ adalah sebuah halaman daftar pertanyaan umum yang sering ditanyakan oleh banyak orang dan jawaban tentang topik tertentu. Fitur FAQ yang dikembangkan adalah *Get FAQ By Category, Get FAQ By ID, Create FAQ, Update FAQ,* dan *Delete FAQ.*

L. Fitur Instructor

Fitur *Instructor* adalah fitur yang berfungsi untuk mengelola data *Instructor* pada API HiColleagues. *Instructor* adalah kumpulan data pengajar yang mengajar pada program *Class* di HiColleagues. Fitur *Instructor* yang dikembangkan adalah *Get Instructors, Get Instructor By ID, Create Instructor, Update Instructor,* dan *Delete Instructor*.

M. Fitur Voucher

Fitur *Voucher* adalah fitur yang berfungsi untuk mengelola data *Voucher* pada API HiColleagues. *Voucher* adalah kumpulan data kupon yang memiliki kode tertentu untuk ditukarkan ketika pembelian sebuah program *Class* di HiColleagues agar mendapat potongan harga. Fitur *Voucher* yang dikembangkan adalah *Get Vouchers, Get Voucher By ID, Create Voucher, Update Voucher, Delete Voucher,* dan *Import Voucher*.

N. Fitur Scheduler

scheduler.go

```go
func RunScheduler(repo registry.RepositoryRegistry) {
    locate, _ := time.LoadLocation("Asia/Jakarta")

    s := gocron.NewScheduler(locate)

    s.Every(1).Day().At("01:00;02:00;03:00").Do(func() error {
        logrus.Info("Scheduler running at ", time.Now())
        // update voucher status
        repo.GetVoucherRepository().UpdateVoucherStatus(context.TODO())

        // update class discount status
        repo.GetClassDetailRepository().UpdateClassDiscountStatus(context.TODO())
        return nil
    })

    s.WaitForScheduleAll()
    s.StartAsync()
}
```

Gambar 4. File scheduler.go

Fitur *Scheduler* adalah fitur yang berfungsi untuk menjalankan tugas-tugas tertentu secara otomatis pada waktu tertentu atau dengan interval tertentu. Dalam proyek ini, fitur *Scheduler* dapat melakukan *Update Voucher Status* dan *Update Class Discount Status* ketika sudah melewati tanggal kedaluwarsa ataupun memasuki tanggal mulai berlakunya.

V. SIMPULAN DAN SARAN

A. Simpulan

Pengembangan API pada modul master data sudah berjalan dengan cukup baik. Fitur-fitur yang sudah dikembangkan dapat berjalan dengan baik dan bisa mengelola data sesuai kebutuhan bisnis perusahaan HiColleagues. Semua fitur yang dikembangkan sudah melewati tahap pengujian API melalui aplikasi Postman dan tidak terjadi kendala. Pengembangan API pada modul master data ini dapat membantu perusahaan dalam merespons tantangan pertumbuhan bisnis dan memenuhi kebutuhan perusahaan.

B. Saran

Saran yang dapat disampaikan ketika mengembangkan API pada modul master data adalah menggunakan *Object Relational Mapping* (ORM) untuk mengelola perubahan skema basis data dengan lebih mudah dan memudahkan proses pengembangan sebuah fitur karena tidak perlu menuliskan *query* secara manual. Selain itu, menggunakan teknologi *Caching* menggunakan Redis juga dapat memberikan kinerja baca dan tulis data yang cepat. Tujuan digunakan Redis agar ke depannya ketika data yang dimiliki oleh perusahaan sudah banyak, permintaan data ke basis data dapat berkurang dan menghasilkan latensi yang lebih rendah.

DAFTAR PUSTAKA

[1] J. Meyerson, “The Go Programming Language,” *IEEE*, pp. 101-104, 2014.

[2] A. A. A. Donovan and B. W. Kernighan, The Go Programming Language, Amerika: Addison-Wesley Professional Computing Series, 2016.

[3] “Echo,” Echo, LabStack LLC., 2023. [Online]. Available: https://echo.labstack.com/. [Accessed 5 November 2023].

[4] S. Juba, A. Vannahme and A. Volkov, Learning PostgreSQL, Birmingham: Packt Publishing, 2015.

[5] Hasanuddin, H. Asgar and B. Hartono, “Rancang Bangun REST API Aplikasi WeShare Sebagai Upaya Mempermudah Pelayanan Donasi Kemanusiaan,” *Jurnal Informatika Teknologi dan Sains*, vol. 4, no. 1, pp. 8-14, 2022.

[6] I. A. K. P. Paramitha, D. M. Wiharta and I. M. A. Suyadnya, “Perancangan dan Implementasi RESTful API Pada Sistem Informasi Manajemen Dosen Universitas Udayana,” *Jurnal SPEKTRUM*, vol. 9, no. 3, pp. 15-23, 2022.

[7] R. Patria, “Rest API adalah: Perbedaan Rest API dan Restful API,” DomaiNesia, 24 April 2023. [Online]. Available: https://www.domainesia.com/berita/rest-api-adalah/. [Accessed 21 November 2023].

[8] Y. Herdiana, “Aplikasi Rumus Matematika SMA Berbasis Mobile,” *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*, 2014.

[9] Postman, “What is Postman?,” Postman, 2023. [Online]. Available: https://www.postman.com/product/what-is-postman/. [Accessed 21 November 2023].

[10] W. G. Wardhana, I. Arwani and B. Rahayudi, “Implementasi Teknologi Restful Web Service Dalam Pengembangan Sistem Informasi Perekaman Prestasi Mahasiswa Berbasis Website,” *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 4, no. 2, pp. 680-689, 2020.

[11] S. Waruwu and I. K. D. Nuryana, “Implementasi Arsitektur Monolitik Pada Rancang Bangun Sistem Informasi,” *Journal of Informatics and Computer Science (JINACS)*, vol. 4, no. 4, 2023.

[12] S. Peyrott, JWT Handbook, Auth0 Inc., 2018.

[13] A. W. Subagio and F. Muttaqin, “Penerapan Clean Architecture pada Pengembangan,” *Jurnal Teknologi Elektro dan Kejuruan*, vol. 32, no. 2, pp. 324-333, 2022.