

Basic Golang untuk Back End Developer

Membuat MyGram Rest API / Final Project

Panduan Pengerjaan & Penilaian

Pada final project ini, kalian akan diminta untuk membuat suatu aplikasi bernama MyGram, yang dimana pada aplikasi ini kalian dapat menyimpan foto maupun membuat comment untuk foto orang lain. Aplikasi ini akan dilengkapi dengan proses CRUD dengan table dan alur yang akan dijelaskan berikut ini:

1. Project ini bebas dikerjakan dengan library apapun. Namun agar proses pengerjaannya lebih cepat dan mudah, disarankan untuk menggunakan framework Gin Gonic dan orm Gorm.

2. Berikut merupakan library/package yang wajib digunakan

- github.com/dgrijalva/jwt-go
- golang.org/x/crypto

3. Dalam project ini akan memerlukan 4 table. Berikut merupakan table-table dan field-field yang diperlukan dalam project ini.



User

- id (Primary key)
- username (string)
- email (string)
- password (string)
- age (integer)
- created_at (Date)
- updated_at (Date)

Photo

- id (Primary key)
- title (string)
- caption (string)
- photo_url (string)
- user_id (Foreign Key Of User Table)
- created_at (Date)
- updated_at (Date)

Comment

- id (Primary Key)
- user_id (Foreign Key Of User Table)
- photo_id (Foreign Key Of Photo Table)
- message (string)
- created_at (Date)
- updated_at (Date)

SocialMedia

- id (Primary Key)
- name (String/ varchar)
- social_media_url (String/ Text)
- UserId(Foreign Key Of User Table)

4. Keempat table tersebut harus mempunyai validasi-validasi pada tiap field-field nya. Validasi boleh dibuat sendiri ataupun menggunakan package seperti Go Validator. Berikut merupakan penjelasan validasinya:

A. Validasi untuk table User

1. Field email

- Validasi pengecekan format email yang valid
- Validasi agar dapat menjadi unique index
- Validasi agar field email tidak boleh kosong atau harus terisi

2. Field username

- Validasi agar dapat menjadi unique index
- Validasi agar field username tidak boleh kosong atau harus terisi

3. Field password

- Validasi agar field password tidak boleh kosong
- Validasi agar field password minimal memiliki panjang sebanyak 6 karakter

4. Field age

- Validasi agar field age tidak boleh kosong atau harus terisi
- Validasi agar field age minimal memiliki nilai diatas 8

B. Validasi Untuk Table Photo

1. Field title

- Validasi agar field title tidak boleh kosong

2. Field photo_url

-Validasi agar field photo_url tidak boleh kosong atau harus terisi

C. Validasi Untuk Table SocialMedia

1. Field name

-Validasi agar field name tidak boleh kosong atau haru terisi

2. Field social_media_url

-Validasi agar field social_media_url tidak boleh kosong atau harus terisi

D. Validasi untuk Table Comment

1.Field message

-Validasi agar field message tidak boleh kosong atau harus terisi

4. Berikut merupakan alur dari aplikasinya:

- Endpoint-endpoint untuk mengakses data pada table SocialMedia, Photo, dan Comment harus melalui proses autentikasi terlebih dahulu, dan proses autentitakasinya wajib menggunakan JsonWebToken.

- Untuk endpoint-endpoint yang berguna untuk memodifikasi data kepemilikan seperti Update atau delete maka harus melalui proses autorisasi.

5. Berikut merupakan endpoint-endpoint yang harus dibuat beserta request body dan response data yang harus diikuti (Wajib)

- Users

POST /users/register

Request

- body:

```
{
  "age": "integer",
  "email": "string",
  "password": "string",
  "username": "string"
}
```

Response:

- status 201
- data:

```
{
  "age": "integer",
  "email": "string",
  "id": "integer",
  "username": "string"
}
```

Notes: Password user harus di hash menggunakan package Bcrypt sebelum di simpan ke database.

POST /users/login

Request

- body:

```
{  
  "email": "string",  
  "password": "string"  
}
```

Response:

- status 200
- data:

```
{  
  "token": "jwt string"  
}
```

Notes: Pada endpoint ini, wajib melakukan logika user login yang dimana harus melakukan pengecekan email dan password user. Pengecekan password wajib dilakukan dengan bantuan library/package Bcrypt.

PUT /users

Request:

- headers: Authorization (Bearer token string)
- params: userId (integer)
- body:

```
{
  "email": "string",
  "username": "string"
}
```

Response:

- status 200
- data:

```
{
  "id": "integer",
  "email": "string",
  "username": "string",
  "age": "integer",
  "updated_at": "date"
}
```

Notes: Endpoint ini perlu melewati proses autentikasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

DELETE /users

Request

- headers: Authorization (Bearer token string)

Response:

- status 200
- data:

```
{
  "message": "Your account has been successfully deleted"
}
```

Notes: Endpoint ini perlu melewati proses autentikasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

- Photos

POST /photos

Request

- headers: Authorization (Bearer token string)
- body:

```
{
  "title": "string",
  "caption": "string",
  "photo_url": "string"
}
```

Response:

- status 201
- data:

```
{
  "id": "integer",
  "title": "string",
  "caption": "string",
  "photo_url": "string",
  "user_id": "integer",
  "created_at": "date"
}
```

Notes: Endpoint ini perlu melewati proses autentikasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

GET /photos

Request

- headers: Authorization (Bearer token string)

Response:

- status 200
- data:


```
[
  {
    "id": "integer",
    "title": "string",
    "caption": "string",
    "photo_url": "string",
    "user_id": "integer",
    "created_at": "date",
    "updated_at": "date",
    "User": {
      "email": "string",
      "username": "string"
    }
  },
  {
    "id": "integer",
    "title": "string",
    "caption": "string",
    "photo_url": "string",
    "user_id": "integer",
    "created_at": "date",
    "updated_at": "date",
    "User": {
      "email": "string",
      "username": "string"
    }
  }
]
```

Notes: Endpoint ini perlu melewati proses autentikasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

PUT /photos/:photold

Request

- headers: Authorization (Bearer token string)
- params: photold (string)
- body:

```
{
  "title": "string",
  "caption": "string",
  "photo_url": "string"
}
```

Response:

- status 200
- data:

```
{
  "id": "integer",
  "title": "string",
  "caption": "string",
  "photo_url": "string",
  "user_id": "integer",
  "updated_at": "date"
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken. Dan alur proses otorisasinya adalah user hanya boleh mengupdate data photo miliknya sendiri.

DELETE /photos/:photold

Request

- headers: Authorization (Bearer token string)
- params: photold (integer)

Response:

- status 200
- data:

```
{  
  "message": "Your photo has been successfully deleted"  
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken. Dan alur proses otorisasinya adalah user hanya boleh menghapus data photo miliknya sendiri.

- Comments

POST /comments

Request

- headers: Authorization (Bearer token string)
- body:

```
{  
  "message": "string",  
  "photo_id": "integer"  
}
```

Response:

- status 201
- data:

```
{  
  "id": "integer"  
  "message": "string",  
  "photo_id": "integer",  
  "user_id": "integer",  
  "created_at": "date"  
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

GET /comments

Request

- **headers:** Authorization (Bearer token string)

Response:

- **status** 200
- **data:**

```
[
  {
    "id": "integer",
    "message": "string",
    "photo_id": "integer",
    "user_id": "integer",
    "updated_at": "date",
    "created_at": "date",
    "User": {
      "id": "integer",
      "email": "string",
      "username": "string"
    },
    "Photo": {
      "id": "integer",
      "title": "string",
      "caption": "string",
      "photo_url": "string",
      "user_id": "integer",
    }
  }
]
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

PUT /comments/:commentId

Request

- headers: Authorization (Bearer token string)
- params: commentId (integer)
- body:

```
{  
  "message": "string"  
}
```

Response:

- status 200
- data:

```
{  
  "id": "integer",  
  "title": "string",  
  "caption": "string",  
  "photo_url": "string",  
  "user_id": "integer",  
  "updated_at": "date"  
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan autorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken. Dan alur proses autorisasinya adalah user hanya boleh mengupdate data comment miliknya sendiri.

DELETE /comments/:commentId

Request

- headers: Authorization (Bearer token string)
- params: commentId (integer)

Response:

- status 200
- data:

```
{  
  "message": "Your comment has been successfully deleted"  
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken. Dan alur proses otorisasinya adalah user hanya boleh menghapus data comment miliknya sendiri.

- SocialMedias

POST /socialmedias

Request

- headers: Authorization (Bearer token string)
- body:

```
{
  "name": "string",
  "social_media_url": "string"
}
```

Response:

- status: 201
- body:

```
{
  "id": "integer",
  "name": "string",
  "social_media_url": "string",
  "user_id": "integer",
  "created_at": "date"
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

GET /socialmedias

Request

- headers: Authorization (Bearer token string)

Response:

- status: 200
- body:

```
{
  "social_medias": [
    {
      "id": "integer",
      "name": "string",
      "social_media_url": "string",
      "UserId": "integer",
      "createdAt": "date",
      "updatedAt": "date",
      "User": {
        "id": "integer",
        "username": "string",
        "profile_image_url": "string"
      }
    }
  ]
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan autorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken.

PUT /socialmedias/:socialMediaId

Request

- headers: Authorization (Bearer token string)
- params: socialMediaId (integer)
- body:


```
{
  "name": "string",
  "social_media_url": "string"
}
```

Response:

- status: 200
- body:

```
{
  "id": "integer",
  "name": "string",
  "social_media_url": "string",
  "user_id": "integer",
  "updated_at": "date"
}
```

Notes: Endpoint ini perlu melewati proses autentikasi dan otorisasi terlebih dahulu. Proses autentikasi wajib dilakukan dengan bantuan package/library JsonWebToken. Dan alur proses otorisasinya adalah user hanya boleh mengupdate data social media miliknya sendiri.

DELETE /socialmedias/:socialMediaId

Request

- headers: Authorization (Bearer token string)
- params: socialMediaId (integer)

Response:

- status: 200
- body:

```
{  
  "message": "Your social media has been successfully deleted"  
}
```

Notes (Wajib)

- Seluruh routing endpoint diatas harus diikuti dengan betul
- Seluruh request body, headers maupun request params harus diikuti dengan betul.
- Seluruh response status, maupun response data nya harus diikuti dengan betul.
- Mohon untuk diperhatikan notes-notes yang telah diberikan diatas seperti endpoint-endpoint yang harus melewati proses autentikasi dan yang tidak. Begitu juga dengan proses autorisasinya.
- Perlu diingat disini bahwa proses autorisasi dilakukan setelah proses autentikasi, bukan sebaliknya.

Pengumpulan Tugas (Wajib)

- Push project ini ke akun github kalian. Setelah itu wajib di deploy ke heroku.
- Buat satu file readme.md di repo project github kalian, dan letakkan link herokunya disana.
- Cantumkan link repo project github kalian di google classroom
- Batas pengumpulan unjuk keterampilan adalah maksimal 3x24 jam setelah instruksi diberikan
- Penilaian atas unjuk keterampilan dilakukan maksimal 2x24 jam setelah tenggat waktu pengumpulan.

Project Rubric

Proyek kamu akan dinilai oleh Hacktiv8 PTP Program Code Reviewer berdasarkan rubrik. Pastikan untuk me-review proyek kamu terlebih dahulu sebelum submit. Semua kriteria harus terpenuhi untuk mendapatkan nilai.

Code Review

Criteria	Meet Expectations
Response Data	Dapat memberikan data response yang sesuai dengan yang sudah dijelaskan
Authentication	Dapat menerapkan proses autentikasi dengan betul menggunakan package JsonWebToken.
Authorization	Dapat menerapkan proses autorisasi yang benar dan sesuai tergantung dari kasus nya.
Validation	Dapat menerapkan validasi-validasi yang telah dijelaskan
Routing	Dapat menerapkan routing yang sesuai dengan yang diminta
Request	Dapat menerapkan request body, headers, maupun request params yang sesuai dengan yang diminta
Status Code	Dapat memberikan status code yang sesuai tergantung dari data response dan kasusnya.
Login	Dapat melakukan alur user login yang betul dan mampu membuat token menggunakan package JsonWebToken
Apakah Kode Berjalan Tanpa Ada Error?	Kode berjalan tanpa ada error. Seluruh kode berfungsi dan dibuat dengan benar.

FORM FINAL PROJECT

ASPEK	RUBRIK		BOBOT	SKOR
	SKALA	DESKRIPSI		
Validation	4	Membuat validasi dengan orm Sequelize tetapi kurang lengkap sesuai dengan yang diminta.	10	...
	3	fix nama kolom, fix missing value, fix column data type		
	2	fix nama kolom, handle missing value, sudah tidak ada missing value namun masih ada invalid value		
	1	fix nama kolom, handle missing value namun masih ada missing value		
Response Data	4	Dapat memberikan response data yang sesuai namun status code tidak sesuai dengan yang diminta	10	...
	3	Response kurang sesuai dengan yang diminta namun status code sesuai.		
	2	Response dan status code tidak sesuai		
	1	Hanya dapat memberikan response namun sama sekali tidak ada yang sesuai.		
Authentication	4	Sudah menerapkan proses autentikasi namun penempatan autentikasi pada routing masih salah.	20	...
	3	Sudah menerapkan proses autentikasi namun logika dari autentikasinya kurang tepat.		
	2	Sudah menerapkan proses autentikasi namun masih ada error.		
	1	Sudah menerapkan proses autentikasi namun tidak menggunakan package JsonWebToken.		

--

Authorization	4	Sudah menerapkan proses autorisasi namun logika dari proses autorisasi agak kurang tepat.	25	...
	3	Sudah menerapkan proses autorisasi namun penempatan pada routingnya kurang tepat.		
	2	Sudah menerapkan proses autorisasi namun masih ada error		
	1	Sudah menerapkan proses autorisasi		
Request	4	Dapat memberikan request body, headers, params namun params tidak sesuai	10	
	3	Dapat memberikan request body, params namun headers tidak sesuai		
	2	Dapat memberikan request headers, params namun body tidak sesuai		
	1	Sudah memberikan request body, params dan headers namun tidak ada yang sesuai		
Routing	4	Dari 16 routing yang diberikan, ada < 3 yang tidak sesuai	10	
	3	Dari 16 routing yang diberikan, ada < 6 yang tidak sesuai		
	2	Dari 16 routing yang diberikan, ada < 13 yang tidak sesuai		

	1	Routing masih ada error		
Deployment	4	Mendeploy ke heroku tapi masih ada error	15	...
	3	Memberikan requirement heroku tanpa mendeploy		
	2	nomor 1, menambahkan requirement heroku, tanpa deployment		
	1	dapat membuat flask tanpa deployment		
TOTAL SKOR			100	