Leonardho R Sitanggang

Assignment Guidance: Building API CRUD

Golang Bootcamp Batch 3

Deskripsi Assignment

Assignment ini merupakan kelanjutan dari tugas sebelumnya **"Web Server dan Golang Route"**. Pada tugas sebelumnya, peserta telah membangun sistem backend dasar dengan CRUD API menggunakan Golang dan framework Gin. Sekarang, tugas ini berfokus pada **implementasi file handling dan storage API**, yang akan memungkinkan pengguna untuk mengunggah dan mengunduh file gambar produk dengan aman dan efisien.

Peserta akan mengembangkan sistem backend dengan fitur:

- **Unggah dan unduh gambar produk** untuk memperkaya data inventaris.
- **Penyimpanan file di sistem lokal atau layanan cloud** seperti AWS S3 atau Firebase.
- **Validasi file, error handling, dan optimasi pengelolaan file** dalam API.

Dengan menyelesaikan tugas ini, peserta akan memahami bagaimana mengelola file dalam sistem backend serta menerapkan konsep RESTful API dalam file handling.

Detail Assignment

Tugas ini merupakan kelanjutan dari assignment sebelumnya **"Web Server dan Golang Route"**. Pada tugas ini, peserta akan memperluas fitur backend yang telah dibuat sebelumnya dengan menambahkan **file handling dan storage API** untuk mendukung unggah dan unduh file gambar produk.

Tugas ini berfokus pada **implementasi file handling dan storage API**. Peserta harus mengembangkan fitur berikut:

1. File Handling and Storage

Tambahan utama dalam tugas ini adalah **unggah dan unduh gambar produk**, dengan spesifikasi berikut:

- **Endpoint Upload:** Mengunggah gambar produk ke penyimpanan lokal
- **Endpoint Download:** Mengunduh gambar produk berdasarkan ID produk.
- **Penyimpanan:** Bisa menggunakan folder lokal
- **Validasi File:** Pastikan hanya format gambar tertentu yang diterima (misal: PNG, JPG, JPEG) dengan ukuran maksimum yang ditentukan.
- **Error Handling:** Tangani skenario kesalahan seperti file berukuran terlalu besar atau format yang tidak sesuai.

2. Web Server and Routing

- Gunakan framework **Gin** untuk menyusun route API.
- Pastikan endpoint menangani metode HTTP **POST dan GET** dengan benar.

3. API Integration and Testing

- Uji endpoint **unggah dan unduh file** menggunakan **Postman**.
- Pastikan validasi input dan error handling diterapkan dengan baik.

Jawab

1. **File Type Validation & Size Validation**

```go
// Rules
allowedTypes := []string{"image/jpg", "image/png", "image/jpeg"}
maxSize := 5
```

```go
// Produk Image Upload
var produkImage *string
file, err := ctx.FormFile("produk_image")
if err == nil {
    // Validate File Size
    if file.Size > int64(maxSize)*1024*1024 {
        ctx.JSON(http.StatusBadRequest, gin.H{"message": fmt.Sprintf("file size must be less than %dMB", maxSize)})
        return
    }

    // Open File & Validate File Type
    fileHeader, err := file.Open()
    if err != nil {
        ctx.JSON(http.StatusInternalServerError, gin.H{"message": "failed to open file"})
        return
    }
    defer fileHeader.Close()
    buffer := make([]byte, 512)
    _, err = fileHeader.Read(buffer)
    if err != nil {
        ctx.JSON(http.StatusInternalServerError, gin.H{"message": "failed to read file"})
        return
    }
    fileType := http.DetectContentType(buffer)
    valid := false
    for _, t := range allowedTypes {
        if fileType == t {
            valid = true
            break
        }
    }
    if !valid {
        ctx.JSON(http.StatusBadRequest, gin.H{"message": fmt.Sprintf("file type must be %s", strings.Join(allowedTypes, ", "))})
        return
    }
```

2. **File Storage**

```
// Save File
filename := fmt.Sprintf("%d_%s", time.Now().Unix(), file.Filename)
filePath := "storages/" + filename
if err := ctx.SaveUploadedFile(file, filePath); err != nil {
    ctx.JSON(http.StatusInternalServerError, gin.H{"message": "failed to save image"})
    return
}
url := fmt.Sprintf("http://%s/%s", ctx.Request.Host, filePath)
produkImage = &url
```

3. **Error Handling**

```
if err := ctx.SaveUploadedFile(file, filePath); err != nil {
    ctx.JSON(http.StatusInternalServerError, gin.H{"message": "failed to save image"})
    return
}
```

Gagal mengunggah file ke storage local

```
if !valid {
    ctx.JSON(http.StatusBadRequest, gin.H{"message": fmt.Sprintf("file type must be %s", strings.Join(allowedTypes, ", "))})
    return
}
```

Tipe file unggahan tidak valid

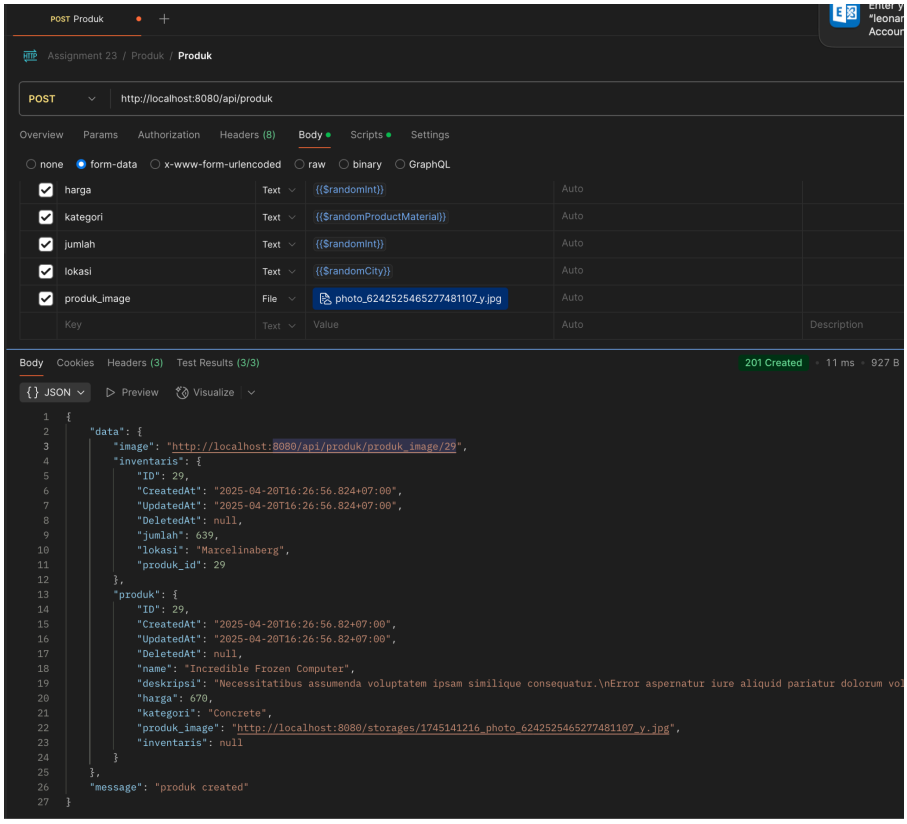
```
// Validate File Size
if file.Size > int64(maxSize)*1024*1024 {
    ctx.JSON(http.StatusBadRequest, gin.H{"message": fmt.Sprintf("file size must be less than %dMB", maxSize)})
    return
}
```

Ukuran file melebihi batas

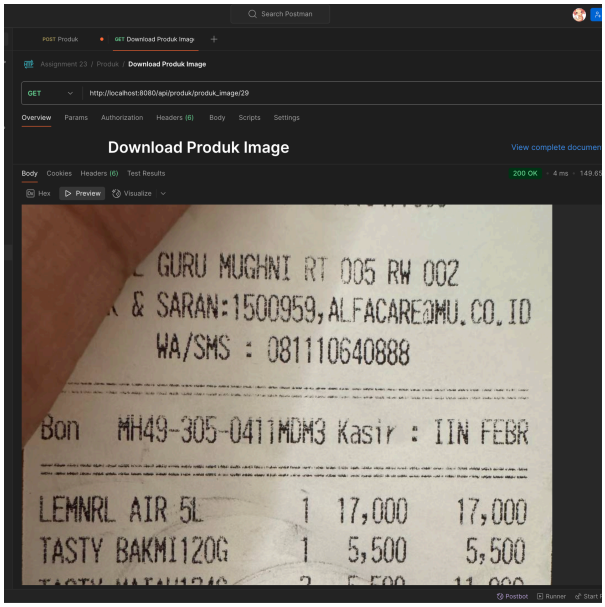
4. **Route**

```
api := r.Group("/api")
{
    produk := api.Group("/produk")
    {
        produk.GET("/", produkController.GetAllProduk)
        produk.GET("/:id", produkController.GetProdukById)
        produk.GET("/produk_image/:id", produkController.GetDownloadProdukImage)
        produk.GET("/kategori/:kategori", produkController.GetProdukByKategori)
        produk.POST("/", produkController.CreateProduk)
        produk.PUT("/:id", produkController.UpdateProdukById)
        produk.DELETE("/delete/:id", produkController.SoftDeleteProdukById)
        produk.DELETE("/destroy/:id", produkController.HardDeleteProdukById)
    }
```

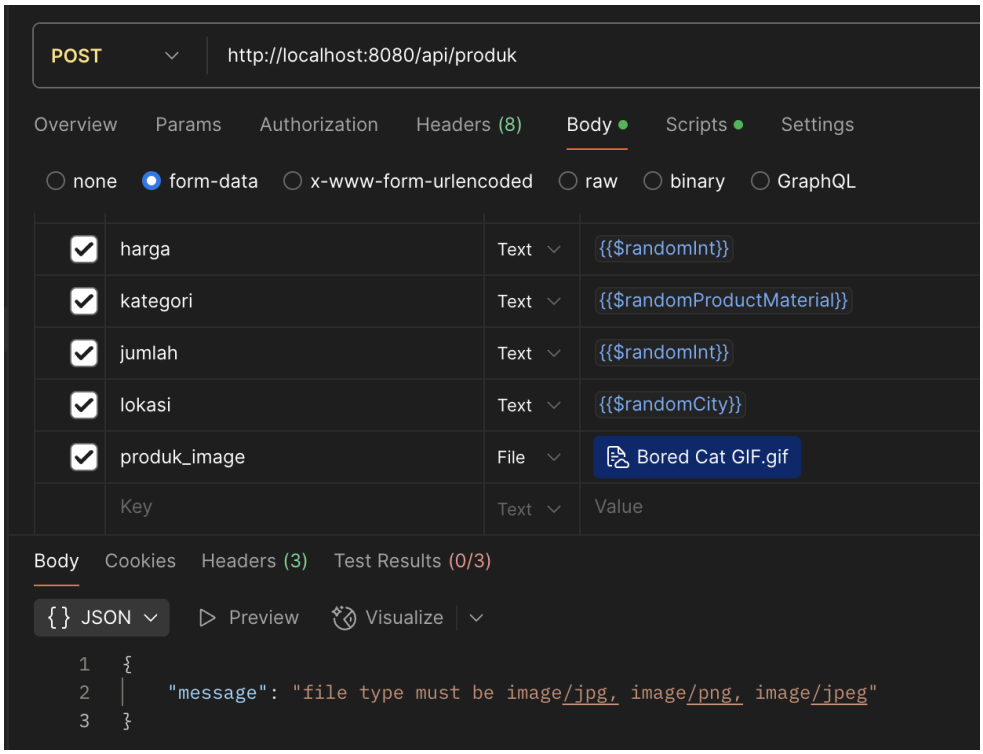
5. Testing Postman



POST : Create Produk & Upload Produk Image



GET : Download Produk Image



Error Handling

Postman Collection Export :
https://drive.google.com/file/d/19zkK_XVjHu536VWTQQhcexKCuRRkSgzu/view?usp=sharing
Dokumentasi Postman :
https://documenter.getpostman.com/view/18882129/2sB2cYdgA4
Github Repo :
https://github.com/FlazeFy/Assignment-Day-23-Bootcamp-Golang-BE