

Nama : Leonardho R Sitanggang

## Assignment Guidance: Case Study Running web server dan route in golang

Golang Backend Development Bootcamp

### Deskripsi Assignment

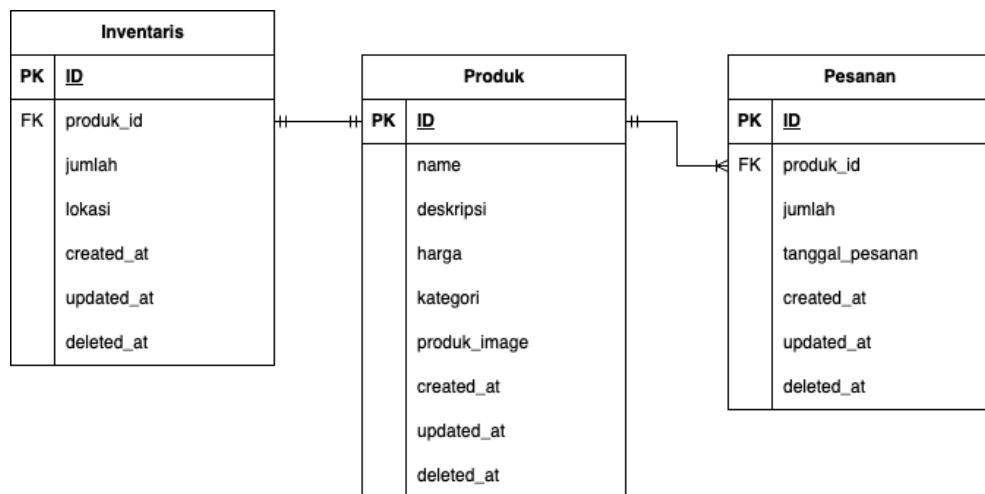
Dalam tugas ini, student akan membangun sistem backend untuk mengelola aplikasi manajemen inventaris. Proyek ini akan melibatkan perancangan dan query pada database relasional untuk mengelola produk, inventaris, dan pesanan. Student akan membuat dan mengintegrasikan RESTful API untuk berinteraksi dengan database, memungkinkan operasi CRUD untuk data inventaris menggunakan Golang.

Selain itu, student akan mengimplementasikan penanganan file untuk mengunggah dan mengunduh gambar produk, serta menyusun route menggunakan framework Gin untuk menyajikan API secara efektif. Dengan menyelesaikan tugas ini, student akan mendapatkan pengalaman langsung dalam pengembangan backend, integrasi API, dan penanganan penyimpanan file, serta menerapkan konsep pemrograman inti dalam skenario dunia nyata.

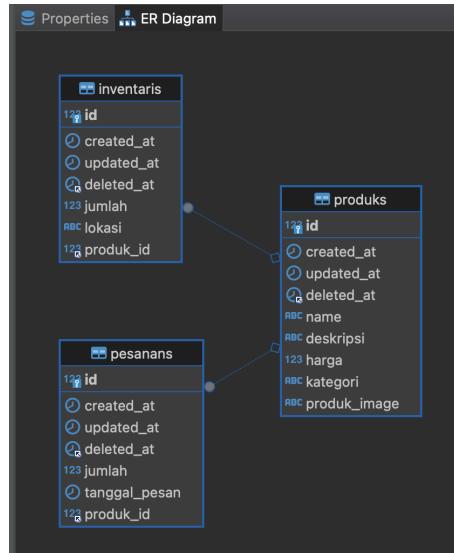
### Detail Assignment

#### 1. Database Design and Queries

- a. Desain sebuah database relasional dengan setidaknya tabel-tabel berikut::
- Produk: Menyimpan detail produk (ID, nama, deskripsi, harga, dan kategori).
  - Inventaris: Melacak tingkat stok dan lokasi produk (ID produk, jumlah, dan lokasi).
  - Pesanan: Mencatat pesanan pelanggan (ID pesanan, ID produk, jumlah, dan tanggal pesanan).



Rancangan ERD



Relasi Database Hasil Migration

- b. Write SQL scripts to:
- Memasukkan data sampel ke dalam tabel-tabel.

```

@ INSERT INTO pesanans
(id, created_at, updated_at, deleted_at, jumlah, tanggal_pesan, produk_id)
VALUES(0, NULL, NULL, NULL, 3, '2024-10-10',1);

Statistics 1 X
Name | Value
Updated Rows 1
Query      INSERT INTO pesanans
(id, created_at, updated_at, deleted_at, jumlah, tanggal_pesan, produk_id)
VALUES(0, NULL, NULL, NULL, 3, '2024-10-10',1)
Start time   Sat Apr 12 06:49:12 WIB 2025
Finish time  Sat Apr 12 06:49:12 WIB 2025

```

```

@ INSERT INTO inventaris
(id, created_at, updated_at, deleted_at, jumlah, lokasi, produk_id)
VALUES(0, NULL, NULL, NULL, 10, 'South Jakarta', 1);

Statistics 1 X
Name | Value
Updated Rows 1
Query      INSERT INTO inventaris
(id, created_at, updated_at, deleted_at, jumlah, lokasi, produk_id)
VALUES(0, NULL, NULL, NULL, 10, 'South Jakarta', 1)
Start time   Sat Apr 12 06:48:11 WIB 2025
Finish time  Sat Apr 12 06:48:11 WIB 2025

```

```

-- Nomor 1B. 1
INSERT INTO produk
(id, created_at, updated_at, deleted_at, name, deskripsi, harga, kategori, produk_image)
VALUES(0, NULL, NULL, NULL, 'Permen', 'Manisan', 10000, 'Snack', NULL);

```

**Statistics 1**

Name	Value
Updated Rows 1	

Query    INSERT INTO produk  
(id, created\_at, updated\_at, deleted\_at, name, deskripsi, harga, kategori, produk\_image)  
VALUES(0, NULL, NULL, NULL, 'Permen', 'Manisan', 10000, 'Snack', NULL)

Start time    Sat Apr 12 06:47:21 WIB 2025  
Finish time    Sat Apr 12 06:47:21 WIB 2025

- Melakukan query untuk produk, inventaris, dan detail pesanan.

select \* from inventaris i;

**inventaris 1**

*	from inventaris i   ↵ ↺ Enter a SQL expression to filter results (use Ctrl+Space)					
id	created_at	updated_at	deleted_at	jumlah	lokasi	produk_id
1	2025-04-12 01:08:53.666	2025-04-12 06:28:23.562	[NULL]	153	West Nehaburgh	1
2	2025-04-12 05:59:03.771	2025-04-12 06:27:55.233	[NULL]	5	North Roosevelt	2
3	2025-04-12 06:06:48.624	2025-04-12 06:06:48.624	[NULL]	489	Enidfurt	3
4	2025-04-12 06:18:10.867	2025-04-12 06:18:10.867	[NULL]	620	New Adelleside	4

select \* from produk p | ↵ ↺ Enter a SQL expression to filter results (use Ctrl+Space)

**produk 1**

*	from produk p   ↵ ↺ Enter a SQL expression to filter results (use Ctrl+Space)	id	created_at	updated_at	deleted_at	name	deskripsi	harga
Grid	1	2025-04-12 01:08:53.662	2025-04-12 01:21:39.883	2025-04-12 01:23:34.180	[NULL]	Handmade Cotton Chair	test update description	659
Text	2	2025-04-12 05:59:03.769	2025-04-12 05:59:03.769	[NULL]	Awesome Granite Car	Volutatem accusantium suscipit soluta nemo ipsa.¶Maxim	453	
Text	3	2025-04-12 06:06:48.623	2025-04-12 06:06:48.623	[NULL]	Ergonomic Rubber Car	Nemo officii in.¶Fugit nisi minus autem enim doloribus qui	114	
Text	4	2025-04-12 06:18:10.862	2025-04-12 06:18:10.862	[NULL]	Small Concrete Cheese	Laboriosam illo rerum voluptatum consectetur earum modi	489	

SELECT \* from pesanans p;

**pesanans 1**

*	from pesanans p   ↵ ↺ Enter a SQL expression to filter results (use Ctrl+Space)	id	created_at	updated_at	deleted_at	jumlah	tanggal_pesan	produk_id
Grid	1	2025-04-12 06:01:54.820	2025-04-12 06:01:54.820	[NULL]	5	2025-04-12	2	
Text	2	2025-04-12 06:27:55.230	2025-04-12 06:27:55.230	[NULL]	4	2025-04-12	2	
Text	3	2025-04-12 06:28:07.027	2025-04-12 06:28:07.027	[NULL]	5	2025-04-12	1	
Text	4	2025-04-12 06:28:12.124	2025-04-12 06:28:12.124	[NULL]	1	2025-04-12	1	
Text	5	2025-04-12 06:28:23.559	2025-04-12 06:28:23.559	[NULL]	4	2025-04-12	1	

<localhost> Script-31

```

SELECT p.name, p.deskripsi, p.harga, i.jumlah, i.lokasi,
count(p2.jumlah) as total_pesanan,
sum(p2.jumlah) as total_jumlah_pesanan,
avg(p2.jumlah) as average_jumlah_per_pesanan
FROM produk p
join inventaris i on i.produk_id = p.id
left join pesanans p2 on p2.produk_id = p.id
group by p.id;

```

**produk(+)**

*	from produk(+)	name	deskripsi	harga	jumlah	lokasi	total_pesana	total_jumlah_pe	average_jum
Grid	1	Handmade Cotton Chair	test update description	659	153	West Nehaburgh	3	10	3.3333
Text	2	Awesome Granite Car	Volutatem accusantium s	2	5	North Roosevelt	2	9	4.5
Text	3	Ergonomic Rubber Car	Nemo officii in.¶Fugit nis	114	489	Enidfurt	0	[NULL]	[NULL]
Text	4	Small Concrete Cheese	Laboriosam illo rerum vol	453	620	New Adelleside	0	[NULL]	[NULL]

```
• UPDATE pesanans
  SET jumlah=1
  WHERE id=5;|
```

Statistics 1 ×

	Value
Edited Rows 1	1
Time	Sat Apr 12 06:51:28 WIB 2025
Time	Sat Apr 12 06:51:28 WIB 2025

```
DELETE from pesanans WHERE id=5;|
```

Statistics 1 ×

	Value
Deleted Rows 1	1
Time	Sat Apr 12 06:52:23 WIB 2025
Time	Sat Apr 12 06:52:23 WIB 2025

- Melakukan agregasi seperti total pesanan untuk suatu produk atau tingkat stok di lokasi tertentu.

```
• -- Nomor 1B. 3
SELECT p.name, p.harga,
       count(p2.jumlah) as total_pesanan,
       sum(p2.jumlah) as total_jumlah_pesanan,
       avg(p2.jumlah) as average_jumlah_per_pesanan,
       sum(p.harga) as total_harga_penjualan
FROM produk p
join inventaris i on i.produk_id = p.id
left join pesanans p2 on p2.produk_id = p.id
where p.id = 1
group by p.id;|
```

Dukus 1 ×

	name	harga	total_pesanan	total_jumlah_pesanan	average_jumlah_per_pesanan	total_harga_penjualan
	Handmade Cotton Chair	659	3	10	3.3333	1,977

```
• SELECT p.name, i.jumlah as stok,
       count(p2.jumlah) as total_pesanan,
       sum(p2.jumlah) as total_jumlah_pesanan,
       avg(p2.jumlah) as average_jumlah_per_pesanan,
       i.jumlah / NULLIF(AVG(p2.jumlah), 0) AS perkiraan_habis_pada_pesanan_ke,
       max(p2.jumlah) as pesanan_jumlah_terbanyak
FROM produk p
join inventaris i on i.produk_id = p.id
left join pesanans p2 on p2.produk_id = p.id
where p.id = 1
group by p.id;|
```

Dukus(+) 1 ×

	name	stok	total_pesanan	total_jumlah_pesanan	average_jumlah	perkiraan_habis	pesanan_jumlah
	Handmade Cotton Chair	153	3	10	3.3333	45.9	5

## 2. RESTful API Development

Buat API untuk hal-hal berikut:

- Produk:**
  - Menambahkan, melihat, memperbarui, dan menghapus produk.

**POST** | http://localhost:8080/api/produk

Overview Params Authorization Headers (8) **Body** Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
<input checked="" type="checkbox"/> name	Text <input type="text" value="{{randomProductName}}"/>
<input checked="" type="checkbox"/> deskripsi	Text <input type="text" value="{{randomLoremLines}}"/>
<input checked="" type="checkbox"/> harga	Text <input type="text" value="{{randomInt}}"/>
<input checked="" type="checkbox"/> kategori	Text <input type="text" value="{{randomProductMaterial}}"/>
<input checked="" type="checkbox"/> jumlah	Text <input type="text" value="{{randomInt}}"/>
<input checked="" type="checkbox"/> lokasi	Text <input type="text" value="{{randomCity}}"/>

Body Cookies Headers (3) Test Results

{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

1 {
2   "data": {
3     "inventaris": {
4       "ID": 1,
5       "CreatedAt": "2025-04-12T01:08:53.666+07:00",
6       "UpdatedAt": "2025-04-12T01:08:53.666+07:00",
7       "DeletedAt": null,
8       "jumlah": 163,
9       "lokasi": "West Nehaburgh",
10      "produk_id": 1
11    },
12    "produk": {
13      "ID": 1,
14      "CreatedAt": "2025-04-12T01:08:53.662+07:00",
15      "UpdatedAt": "2025-04-12T01:08:53.662+07:00",
16      "DeletedAt": null,
17      "name": "Licensed Soft Soap",
18      "deskripsi": "Nesciunt vitae veritatis reprehenderit architecto",
19      "harga": 837,
20      "kategori": "Steel",
21      "produk_image": null,
22      "inventaris": null
23    }
24  },
25  "message": "produk created"
26 }
```

## Menambahkan Produk

**PUT** | http://localhost:8080/api/produk/1

Overview Params Authorization Headers (8) **Body** Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL

```

1 {
2   "name": "{{randomProductName}}",
3   "deskripsi": "test update description",
4   "harga": {{randomInt}}
5 }
```

Body Cookies Headers (3) Test Results

{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

1 {
2   "data": {
3     "ID": 1,
4     "CreatedAt": "2025-04-12T01:08:53.662+07:00",
5     "UpdatedAt": "2025-04-12T01:21:39.883+07:00",
6     "DeletedAt": null,
7     "name": "Handmade Cotton Chair",
8     "deskripsi": "test update description",
9     "harga": 659,
10    "kategori": "",
11    "produk_image": null,
12    "inventaris": null
13  },
14  "message": "produk updated"
15 }
```

## Mengubah Produk

**GET** | http://localhost:8080/api/produk

Overview Params Authorization Headers (6) **Body** Scripts Settings

None form-data x-www-form-urlencoded raw binary GraphQL

Key	Value
Key	Text <input type="text"/>

Body Cookies Headers (3) Test Results

{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

1 {
2   "data": [
3     {
4       "ID": 1,
5       "CreatedAt": "2025-04-12T01:08:53.662+07:00",
6       "UpdatedAt": "2025-04-12T01:08:53.662+07:00",
7       "DeletedAt": null,
8       "name": "Licensed Soft Soap",
9       "deskripsi": "Nesciunt vitae veritatis reprehenderit architecto",
10      "harga": 837,
11      "kategori": "Steel",
12      "produk_image": null,
13      "inventaris": null
14    }
15  ],
16  "message": "produk fetched"
17 }
```

## Melihat Produk

**DELETE** | http://localhost:8080/api/produk/delete/1

Overview Params Authorization Headers (6) Body

Body Cookies Headers (3) Test Results

{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

1 {
2   "message": "produk deleted"
3 }
```

**DELETE** | http://localhost:8080/api/produk/destroy/1

Overview Params Authorization Headers (6) Body Script

Body Cookies Headers (3) Test Results

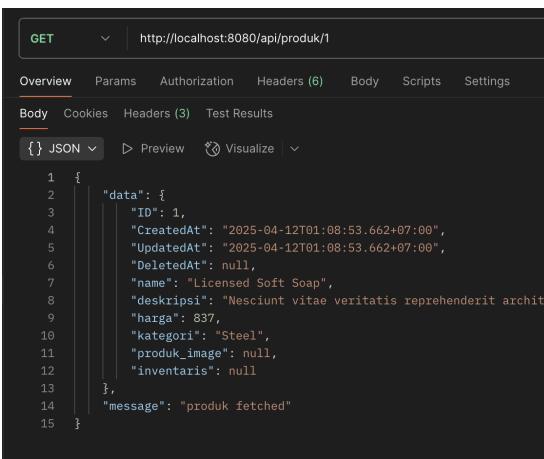
{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

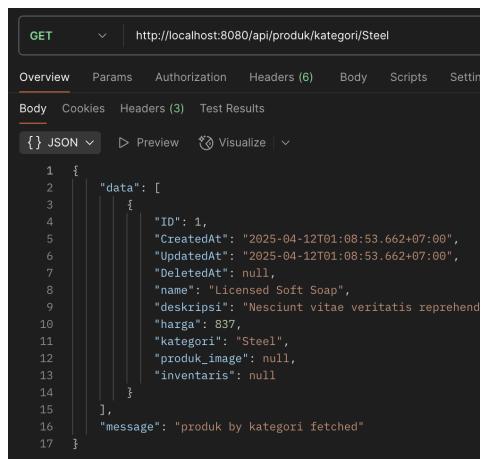
1 {
2   "message": "produk permanently deleted"
3 }
```

## Menghapus Produk

- Melihat detail produk berdasarkan ID atau kategori.



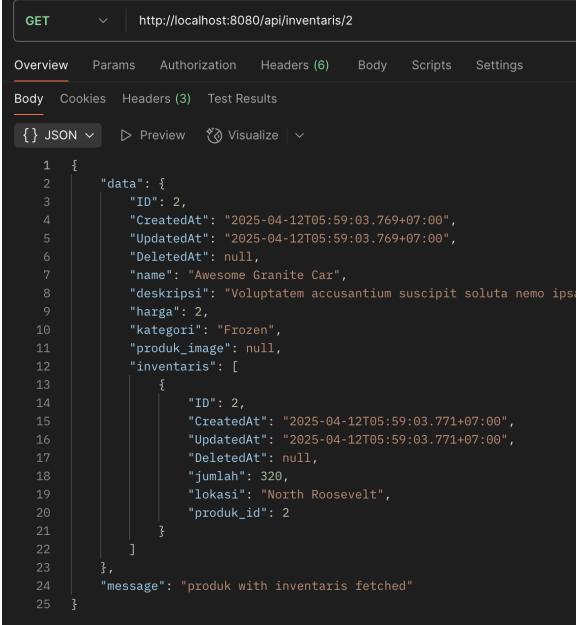
Berdasarkan ID



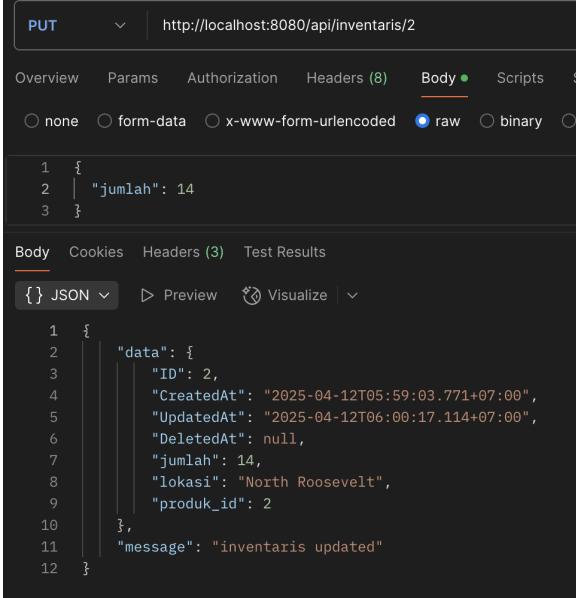
Berdasarkan Kategori

- **Inventaris:**

- Melihat tingkat stok untuk suatu produk.
- Memperbarui tingkat stok (menambah atau mengurangi stok).



Melihat tingkat stok suatu produk

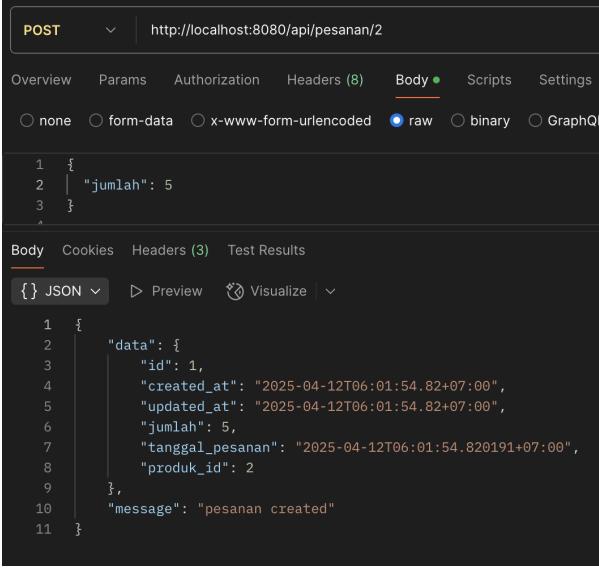


Memperbarui tingkat stok

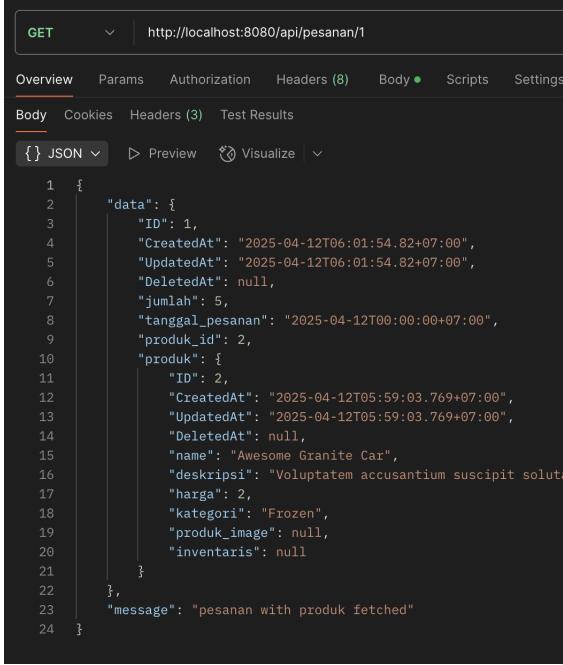
- **Pesanan:**

- Membuat pesanan baru.

- Mengambil detail pesanan berdasarkan ID.



**Membuat pesanan baru**



**Mengambil detail pesanan berdasarkan ID**

### 3. API Integration

- Uji API untuk memastikan integrasi dan fungsionalitas berjalan lancar.
- Tulis skrip atau gunakan alat (misalnya, Postman, Curl) untuk menguji semua endpoint.

Integration Testing API:

- **Produk:**
  - Menambahkan, melihat, memperbarui, dan menghapus produk.

**POST** http://localhost:8080/api/produk

```

1 pm.test("Response has data and message", function () {
2   const jsonData = pm.response.json()
3 
4   pm.expect(jsonData).to.have.property("data")
5   pm.expect(jsonData).to.have.property("message")
6   pm.expect(jsonData.data).to.be.an("object")
7 }
8 )
9 
10 pm.test("Validate each data type (Produk)", function () {
11   const jsonData = pm.response.json().data.produk
12   const stringCol = ["CreatedAt", "UpdatedAt", "name", "deskripsi", "kategori"]
13   stringCol.forEach(dt => {
14     pm.expect(jsonData).to.have.property(dt).that.is.a("string")
15   })
16 
17   const numberCol = ["ID", "harga"]
18   numberCol.forEach(dt => {
19     pm.expect(jsonData).to.have.property(dt).that.is.a("number")
20   })
21 
22   const stringNullableCol = ["DeletedAt", "produk_image"]
23   stringNullableCol.forEach(dt => {
24     pm.expect(jsonData).to.have.property(dt)
25     if (jsonData[dt] === null) {
26       pm.expect(jsonData[dt]).to.be.a("string")
27     } else {
28       pm.expect(jsonData[dt]).to.be.null
29     }
30   })
31 }
32 )
33 
34 pm.test("Validate each data type (Inventaris)", function () {
35   const jsonData = pm.response.json().data.inventaris
36   const stringCol = ["CreatedAt", "UpdatedAt", "lokasi"]
37   stringCol.forEach(dt => {
38     pm.expect(jsonData).to.have.property(dt).that.is.a("string")
39   })
40 
41   const numberCol = ["ID", "jumlah", "produk_id"]
42   numberCol.forEach(dt => {
43     pm.expect(jsonData).to.have.property(dt).that.is.a("number")
44   })
45 
46   const stringNullableCol = ["DeletedAt"]
47   stringNullableCol.forEach(dt => {
48     pm.expect(jsonData).to.have.property(dt)
49     if (jsonData[dt] === null) {
50       pm.expect(jsonData[dt]).to.be.a("string")
51     } else {
52       pm.expect(jsonData[dt]).to.be.null
53     }
54   })
55 }
56 )
57 
```

Body Cookies Headers (3) Test Results (3/3)

Filter Results ▾

PASSED Response has data and message

PASSED Validate each data type (Produk)

PASSED Validate each data type (Inventaris)

**PUT** http://localhost:8080/api/produk/6

```

1 pm.test("Response has data and message", function () {
2   pm.response.to.have.status(200)
3   const jsonData = pm.response.json()
4 
5   pm.expect(jsonData).to.have.property("data")
6   pm.expect(jsonData).to.have.property("message")
7   pm.expect(jsonData.data).to.be.an("object")
8 }
9 )
10 
11 pm.test("Validate each data type", function () {
12   const jsonData = pm.response.json().data.produk
13   const stringCol = ["CreatedAt", "UpdatedAt", "name", "deskripsi", "kategori"]
14   stringCol.forEach(dt => {
15     pm.expect(jsonData).to.have.property(dt).that.is.a("string")
16   })
17 
18   const numberCol = ["ID", "harga"]
19   numberCol.forEach(dt => {
20     pm.expect(jsonData).to.have.property(dt).that.is.a("number")
21   })
22 
23   const stringNullableCol = ["DeletedAt", "produk_image"]
24   stringNullableCol.forEach(dt => {
25     pm.expect(jsonData).to.have.property(dt)
26     if (jsonData[dt] === null) {
27       pm.expect(jsonData[dt]).to.be.a("string")
28     } else {
29       pm.expect(jsonData[dt]).to.be.null
30     }
31   })
32 }
33 )
34 
```

Body Cookies Headers (3) Test Results (2/2)

Filter Results ▾

PASSED Response has data and message

PASSED Validate each data type

## Mengubah Produk

## Menambahkan Produk

**Melihat Produk**

```

    pm.test("Response has data and message", function () {
        pm.response.to.have.status(200)
        const jsonData = pm.response.json()
        pm.expect(jsonData).to.have.property("data")
        pm.expect(jsonData).to.have.property("message")
        pm.expect(jsonData.data).to.be.an("array")
    })
    pm.test("Validate each data type", function () {
        const jsonData = pm.response.json()
        jsonData.data.forEach(item => {
            const stringCol = ["createdAt", "updatedAt", "name", "deskripsi", "kategori"]
            stringCol.forEach(dt => {
                pm.expect(item).to.have.property(dt).that.is.a("string")
            })
        })
        const numberCol = ["ID", "harga"]
        numberCol.forEach(dt => {
            pm.expect(item).to.have.property(dt).that.is.a("number")
        })
    })
    const stringNullableCol = ["DeletedAt", "produk_image"]
    stringNullableCol.forEach(dt => {
        pm.expect(item).to.have.property(dt)
        if (item[dt] === null) {
            pm.expect(item[dt]).to.be.a("string")
        } else {
            pm.expect(item[dt]).to.be.null
        }
    })
})
    
```

PASSSED Response has data and message  
PASSED Validate each data type

**Menghapus Produk**

```

    pm.test("Response has data and message", function () {
        pm.response.to.have.status(200)
        const jsonData = pm.response.json()
        pm.expect(jsonData).to.have.property("message")
        pm.expect(jsonData["message"]).that.is.a("string")
    })
    
```

PASSSED Response has data and message

- Melihat detail produk berdasarkan ID atau kategori.

```

    pm.test("Response has data and message", function () {
        pm.response.to.have.status(200)
        const jsonData = pm.response.json()
        pm.expect(jsonData).to.have.property("data")
        pm.expect(jsonData).to.have.property("message")
        pm.expect(jsonData.data).to.be.an("object")
    })
    pm.test("Validate each data type", function () {
        const jsonData = pm.response.json().data
        const stringCol = ["createdAt", "updatedAt", "name", "deskripsi", "kategori"]
        stringCol.forEach(dt => {
            pm.expect(jsonData).to.have.property(dt).that.is.a("string")
        })
        const numberCol = ["ID", "harga"]
        numberCol.forEach(dt => {
            pm.expect(jsonData).to.have.property(dt).that.is.a("number")
        })
    })
    const stringNullableCol = ["DeletedAt", "produk_image"]
    stringNullableCol.forEach(dt => {
        pm.expect(jsonData).to.have.property(dt)
        if (jsonData[dt] === null) {
            pm.expect(jsonData[dt]).to.be.a("string")
        } else {
            pm.expect(jsonData[dt]).to.be.null
        }
    })
})
    
```

PASSSED Response has data and message  
PASSED Validate each data type

```

    pm.test("Response has data and message", function () {
        pm.response.to.have.status(200)
        const jsonData = pm.response.json()
        pm.expect(jsonData).to.have.property("data")
        pm.expect(jsonData).to.have.property("message")
        pm.expect(jsonData.data).to.be.an("array")
    })
    pm.test("Validate each data type", function () {
        const jsonData = pm.response.json()
        jsonData.data.forEach(item => {
            const stringCol = ["CreatedAt", "UpdatedAt", "name", "deskripsi", "kategori"]
            stringCol.forEach(dt => {
                pm.expect(item).to.have.property(dt).that.is.a("string")
            })
        })
        const numberCol = ["ID", "harga"]
        numberCol.forEach(dt => {
            pm.expect(item).to.have.property(dt).that.is.a("number")
        })
    })
    const stringNullableCol = ["DeletedAt", "produk_image"]
    stringNullableCol.forEach(dt => {
        pm.expect(item).to.have.property(dt)
        if (item[dt] === null) {
            pm.expect(item[dt]).to.be.a("string")
        } else {
            pm.expect(item[dt]).to.be.null
        }
    })
})
    
```

PASSSED Response has data and message  
PASSED Validate each data type

- **Inventaris:**
- Melihat tingkat stok untuk suatu produk.

- Memperbarui tingkat stok (menambah atau mengurangi stok).

The screenshot shows two Postman requests side-by-side:

- GET /api/inventaris/2**: This request has its **Script** tab open, containing Jest-style test code to validate the response data type. It checks for properties like 'data' and 'message', and then iterates over 'data' to validate sub-properties like 'name', 'deskripsi', 'kategori', 'ID', 'harga', 'lokasi', and 'produk\_id'.
- PUT /api/inventaris/2**: This request also has its **Script** tab open, with similar validation logic for updating data. It checks for properties like 'data' and 'message', and then iterates over 'data' to validate sub-properties like 'DeletedAt', 'UpdatedAt', 'lokasi', 'ID', 'jumlah', and 'produk\_id'.

Both requests show a **Test Results** section at the bottom with three green **PASSED** status indicators:

- Response has data and message
- Validate each data type (Produk)
- Validate each data type (Inventaris)

Melihat tingkat stok suatu produk

## Memperbarui tingkat stok

- **Pesanan:**
  - Membuat pesanan baru.
  - Mengambil detail pesanan berdasarkan ID.

**POST** | http://localhost:8080/api/pesanan/2

Overview Params Authorization Headers (8) Body Scripts Settings

```

1 pm.test("Response has data and message", function () {
2   pm.response.to.have.status(201)
3   const jsonData = pm.response.json()
4 
5   pm.expect(jsonData).to.have.property("data")
6   pm.expect(jsonData).to.have.property("message")
7   pm.expect(jsonData.data).to.be.an("object")
8 })
9 
10 pm.test("Validate each data type", function () {
11   const jsonData = pm.response.json().data
12   const stringCol = ["CreatedAt", "UpdatedAt", "tanggal_pesanan"]
13   stringCol.forEach(dt => {
14     pm.expect(jsonData).to.have.property(dt).that.is.a("string")
15   })
16 
17   const numberCol = ["ID", "jumlah", "produk_id"]
18   numberCol.forEach(dt => {
19     pm.expect(jsonData).to.have.property(dt).that.is.a("number")
20   })
21 })

```

Body Cookies Headers (3) Test Results (2/2)

Filter Results ▾

PASSED Response has data and message  
PASSED Validate each data type

**GET** | http://localhost:8080/api/pesanan/1

Overview Params Authorization Headers (8) Body Scripts Settings

```

1 pm.test("Response has data and message", function () {
2   pm.response.to.have.status(200)
3   const jsonData = pm.response.json()
4 
5   pm.expect(jsonData).to.have.property("data")
6   pm.expect(jsonData).to.have.property("message")
7   pm.expect(jsonData.data).to.be.an("object")
8 })
9 
10 pm.test("Validate each data type (Pesanan)", function () {
11   const jsonData = pm.response.json().data
12   const stringCol = ["CreatedAt", "UpdatedAt", "tanggal_pesanan"]
13   stringCol.forEach(dt => {
14     pm.expect(jsonData).to.have.property(dt).that.is.a("string")
15   })
16 
17   const numberCol = ["ID", "jumlah", "produk_id"]
18   numberCol.forEach(dt => {
19     pm.expect(jsonData).to.have.property(dt).that.is.a("number")
20   })
21 })
22 
23 pm.test("Validate each data type (Produk)", function () {
24   const jsonData = pm.response.json().data.produk
25   const stringCol = ["CreatedAt", "UpdatedAt", "name", "deskripsi", "kategori"]
26   stringCol.forEach(dt => {
27     pm.expect(jsonData).to.have.property(dt).that.is.a("string")
28   })
29 
30   const numberCol = ["ID", "harga"]
31   numberCol.forEach(dt => {
32     pm.expect(jsonData).to.have.property(dt).that.is.a("number")
33   })
34 
35   const stringNullableCol = ["DeletedAt", "produk_image"]
36   stringNullableCol.forEach(dt => {
37     pm.expect(jsonData).to.have.property(dt)
38     if (jsonData[dt] === null) {
39       pm.expect(jsonData[dt]).to.be.a("string")
40     } else {
41       pm.expect(jsonData[dt]).to.be.null
42     }
43   })
44 })

```

Body Cookies Headers (3) Test Results (3/3)

Filter Results ▾

PASSED Response has data and message  
PASSED Validate each data type (Pesanan)  
PASSED Validate each data type (Produk)

Membuat pesanan baru

Mengambil detail pesanan berdasarkan ID

## 4. Web Server and Routing

Gunakan framework Gin di Golang untuk:

- Menyusun route untuk semua endpoint API (misalnya, /products, /inventory, /orders).
- Menangani metode HTTP seperti GET, POST, PUT, dan DELETE.

```

9
10 func SetUpRoutes(r *gin.Engine, db *gorm.DB) {
11     produkController := controllers.NewProdukController(db)
12     inventarisController := controllers.NewInventarisController(db)
13     pesananController := controllers.NewPesananController(db)
14
15     api := r.Group("/api")
16     {
17         produk := api.Group("/produk")
18         {
19             produk.GET("/", produkController.GetAllProduk)
20             produk.GET("/:id", produkController.GetProdukById)
21             produk.GET("/produk_image/:id", produkController.GetDownloadProdukImage)
22             produk.GET("/kategori/:kategori", produkController.GetProdukByKategori)
23             produk.POST("/", produkController.CreateProduk)
24             produk.PUT("/:id", produkController.UpdateProdukById)
25             produk.DELETE("/delete/:id", produkController.SoftDeleteProdukById)
26             produk.DELETE("/destroy/:id", produkController.HardDeleteProdukById)
27         }
28         inventaris := api.Group("/inventaris")
29         {
30             inventaris.GET("/:id", inventarisController.GetProdukInventarisByProdukId)
31             inventaris.PUT("/:id", inventarisController.UpdateInventarisById)
32             inventaris.DELETE("/destroy/:id", inventarisController.HardDeleteInventarisById)
33         }
34         pesanan := api.Group("/pesanan")
35         {
36             pesanan.GET("/:id", pesananController.GetDetailPesananById)
37             pesanan.POST("/:produk_id", pesananController.CreatePesanan)
38             pesanan.DELETE("/destroy/:id", pesananController.HardDeletePesananById)
39         }
40     }
41 }

```

## 5. Mengunggah dan Mengunduh File

The screenshot shows a POST request to `http://localhost:8080/api/produk`. The request body is set to `form-data` and contains several fields: `deskripsi`, `harga`, `kategori`, `jumlah`, `lokasi`, and `produk_image`. The `produk_image` field is currently selected, showing a file upload interface with the path `403019_avatar_male_man_person_user_i...` and a size of `25.4 KB`. The response on the right is a successful `200 OK` with the following details:

- Request URL:** `http://localhost:8080/api/produk/produk_image/3`
- Status Code:** `200 OK`
- Remote Address:** `[::]:8080`
- Referrer Policy:** `strict-origin-when-cross-origin`
- Response Headers:**
  - `Content-Type: application/json; charset=utf-8`
  - `Content-Length: 25858`
  - `Date: Fri, 11 Apr 2025 23:19:20 GMT`
  - `Last-Modified: Fri, 11 Apr 2025 23:06:48 GMT`
- Raw Response:**

```
{
    "data": {
        "inventaris": {
            "ID": 3,
            "CreatedAt": "2025-04-12T06:06:48.624+07:00",
            "UpdatedAt": "2025-04-12T06:06:48.624+07:00",
            "DeletedAt": null,
            "jumlah": 489,
            "lokasi": "Enidfuzt",
            "produk_id": 3
        },
        "produk": {
            "ID": 3,
            "CreatedAt": "2025-04-12T06:06:48.623+07:00",
            "UpdatedAt": "2025-04-12T06:06:48.623+07:00",
            "DeletedAt": null,
            "name": "Ergonomic Rubber Car",
            "deskripsi": "Nemo officiis in.\nFugit nisi minus autem enim doloribus qui",
            "harga": 114,
            "kategori": "Metal",
            "produk_image": "http://localhost:8080/storages/174412808_403019_avatar_male_man_person_user_i...",
            "inventaris": null
        }
    },
    "message": "produk created"
}
```

Mengunggah image pada produk

```
✓ ASSIGNMENT-DAY-23-BOOTCAMP-GOLANG-BE
  ✓ config
    -eo database.go
  ✓ controllers
    -eo inventaris_controller.go
    -eo pesanan_controller.go
    -eo produk_controller.go
  ✓ models
    -eo inventaris_model.go
    -eo pesanan_model.go
    -eo produk_model.go
  ✓ routes
    -eo routes.go
  ✓ storages
    📸 1744412808_403019_avatar_male_man_person_user...
  > utils
  ⚙ .env
  ⚡ .gitignore
  ⚡ go.mod
  ⚡ go.sum
  -eo main.go
  ⓘ README.md
```

File disimpan pada folder storages

## 6. Dokumentasi API (Postman) & Tautan Lainnya

Postman Collection Export :

<https://drive.google.com/file/d/1nyozUV-EN3Cjf6fMf-F5tQmTPS5BX4Dm/view?usp=sharing>

Dokumentasi Postman :

<https://documenter.getpostman.com/view/18882129/2sB2cYdgA4>

DB Dump :

<https://drive.google.com/file/d/1oi1RoJBq2nDdW1ozVXXVn5Sr-Hwj1ZbZ/view?usp=sharing>

Github Repo :

<https://github.com/FlazeFy/Assignment-Day-23-Bootcamp-Golang-BE>