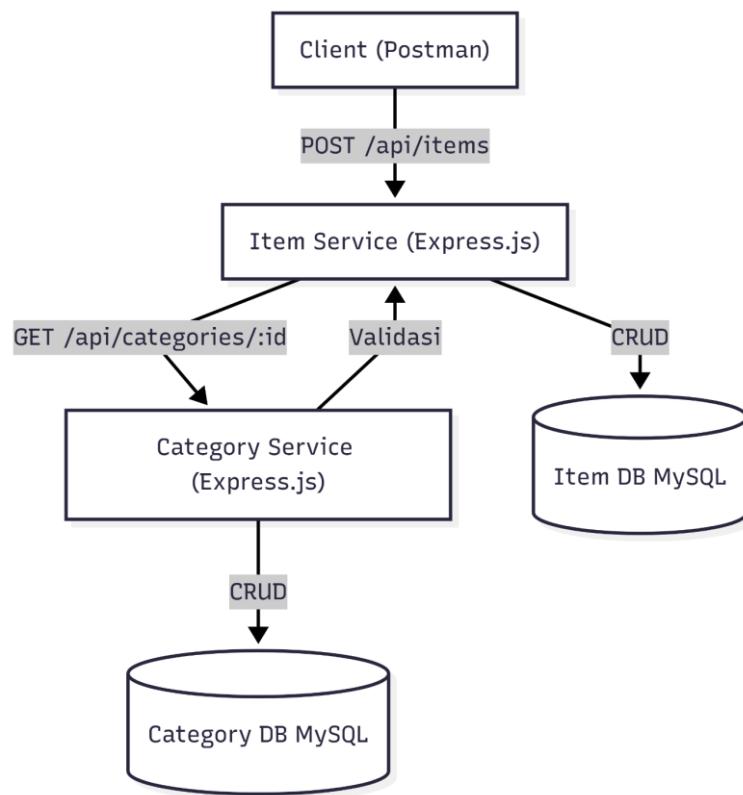


2. Diagram Arsitektur Microservice



1. Client → Category Service

Client dapat mengirimkan request langsung ke Category Service untuk melakukan operasi CRUD data kategori, seperti menambah, menampilkan, mengubah, dan menghapus kategori. Category Service akan memproses request tersebut dan berinteraksi langsung dengan Category Database.

2. Client → Item Service

Client juga dapat mengirimkan request ke Item Service untuk melakukan operasi CRUD data item. Item Service bertanggung jawab untuk mengelola data item dan menyimpannya ke Item Database.

3. Item Service → Category Service (Komunikasi Antar Service)

Pada saat proses pembuatan atau pembaruan data item, Item Service melakukan komunikasi ke Category Service melalui REST API untuk memvalidasi data kategori berdasarkan category_id. Hal ini memastikan bahwa item hanya dapat dibuat atau diperbarui apabila kategori yang digunakan benar-benar tersedia.

4. Service → Database

Setiap service memiliki database masing-masing dan hanya service tersebut yang memiliki akses langsung ke databasenya.

- Category Service ↔ Category Database
- Item Service ↔ Item Database

5. Response ke Client

Setelah proses selesai, masing-masing service akan mengembalikan response dalam format JSON kepada client, baik berupa data hasil proses maupun pesan status.

6. Screenshot hasil pengujian

Category Service

- Create

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** POST /api/categories
- Body:** Raw JSON (POST created)

```
1 [ {  
2   "name": "Dessert"  
3 }  
4 ]
```

- Response:** 201 Created

```
1 {  
2   "message": "Kategori berhasil ditambahkan",  
3   "data": [  
4     {  
5       "id": 4,  
6       "name": "Dessert"  
7     }  
8   ]  
9 }
```

- Read

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** GET /api/categories
- Query Params:** None
- Response:** 200 OK

```
1 {  
2   "message": "Berhasil mengambil data kategori",  
3   "data": [  
4     {  
5       "id": 2,  
6       "name": "Makanan",  
7       "created_at": "2026-01-08T16:05:06.000Z"  
8     },  
9     {  
10       "id": 3,  
11       "name": "Minuman",  
12       "created_at": "2026-01-08T16:05:16.000Z"  
13     },  
14     {  
15       "id": 4,  
16       "name": "Dessert",  
17       "created_at": "2026-01-08T18:27:50.000Z"  
18     }  
19   ]  
20 }
```

● Read by ID

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History, Flows, Files (BETA).
- Request URL:** http://localhost:3001/api/categories/4
- Method:** GET
- Headers:** (6)
- Body:** (JSON) Preview shows a JSON object with fields: id, name, and created_at.
- Response:** 200 OK, 5 ms, 316 B. Body content: { "id": 4, "name": "Makanan Ringan", "created_at": "2026-01-08T18:27:50.000Z"}

● Update

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History, Flows, Files (BETA).
- Request URL:** http://localhost:3001/api/categories/4
- Method:** PUT
- Headers:** (8)
- Body:** (raw) JSON content: { "name": "Makanan Ringan"}
- Response:** 200 OK, 14 ms, 317 B. Body content: { "message": "Kategori berhasil diupdate", "data": { "id": "4", "name": "Makanan Ringan"}}

• Delete

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History, Flows, Files.
- Request URL:** Category-Service / New Request, DELETE http://localhost:3001/api/categories/4
- Body:** raw JSON:

```
1 Ctrl+Alt+P to Ask AI
```
- Response:** 200 OK, 13 ms, 274 B, message: "Kategori berhasil dihapus"
- Bottom:** Cloud View, Find and replace, Console, Terminal, Runner, Start Proxy, Cookies, Vault, Trash.

Item Service

• Create

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History, Flows, Files.
- Request URL:** Item-Service / created, POST http://localhost:3002/api/items
- Body:** raw JSON:

```
1 {
2   "name": "Strawberry Milk",
3   "price": 30000,
4   "category_id": 3
5 }
```
- Response:** 201 Created, 66 ms, 352 B, message: "Item berhasil ditambahkan", data: { id: 15, name: "Strawberry Milk", price: 30000, category_id: 3 }
- Bottom:** Cloud View, Find and replace, Console, Terminal, Runner, Start Proxy, Cookies, Vault, Trash.

● Read

The screenshot shows the Postman interface with a collection named "Item-Service". A GET request is made to `http://localhost:3002/api/items`. The response is a JSON array of items:

```
[{"id": 69, "name": "Americano", "price": 10000, "category_id": 3, "created_at": "2026-01-08T16:14:15.000Z"}, {"id": 70, "name": "Espresso", "price": 10000, "category_id": 3, "created_at": "2026-01-08T16:39:22.000Z"}, {"id": 71, "name": "Strawberry Milk", "price": 30000, "category_id": 3, "created_at": "2026-01-08T18:50:04.000Z"}]
```

● Update

The screenshot shows the Postman interface with a collection named "Item-Service". A PUT request is made to `http://localhost:3002/api/items/15` with the following JSON body:

```
{ "name": "Strawberry Milk Iced", "price": 35000, "category_id": 3 }
```

The response is a JSON object:

```
{"message": "Item berhasil diupdate", "data": { "id": 4, "name": "Strawberry Milk Iced", "price": 35000, "category_id": 3 }}
```

• Delete

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History, Flows, Files (BETA).
- Middle Panel:** Item-Service / `/delete`.
 - Method: `DELETE`.
 - URL: `http://localhost:3002/api/items/15`.
 - Params tab is selected.
 - Body tab shows a JSON response:

```
1 {  
2   "message": "Item berhasil dihapus"  
3 }
```
- Bottom Panel:** Test Results: 200 OK, 11 ms, 270 B.

• Komunikasi antar service

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, History, Flows, Files (BETA).
- Middle Panel:** Item-Service / `/created`.
 - Method: `POST`.
 - URL: `http://localhost:3002/api/items`.
 - Body tab is selected, showing JSON input:

```
1 {  
2   "name": "Strawberry Milk",  
3   "price": 30000,  
4   "category_id": 3  
5 }  
6
```
 - Body tab shows a JSON response:

```
1 {  
2   "message": "Item berhasil ditambahkan",  
3   "data": {  
4     "id": 15,  
5     "name": "Strawberry Milk",  
6     "price": 30000,  
7     "category_id": 3  
8   }  
9 }
```
- Bottom Panel:** Test Results: 201 Created, 66 ms, 352 B.

Pada proses pembuatan item, Item Service melakukan request ke Category Service untuk memvalidasi kategori berdasarkan category_id.