

PDF EXPLANATION

Anggota Kelompok 06:

- | | |
|------------------------------|----------------------|
| 1) Kukuh Agus Hermawan | (24/533395/PA/22573) |
| 2) Aulia Fathus Tsani | (24/534388/PA/22661) |
| 3) Alifa Batrisyia Nariswari | (24/535256/PA/22707) |

CRUD Data Menggunakan Postman

Pada tugas week 3 ini, seluruh proses pengelolaan data (CRUD: *Create, Read, Update, Delete*) diuji menggunakan Postman. Postman merupakan salah satu tools standar industri yang digunakan untuk mengirim request HTTP ke server, sehingga memudahkan proses verifikasi API sebelum membangun antarmuka frontend. Penggunaan Postman pada tahap ini bertujuan untuk memastikan bahwa backend telah bekerja dengan benar, termasuk validasi input, manipulasi data pada database, serta pengembalian respons sesuai standar API.

Cara Menggunakan Postman untuk Mengakses API InfiniRead

Berikut langkah-langkah umum untuk setiap operasi CRUD:

1. Memilih Metode HTTP

Pada sisi kiri kolom URL, pilih method sesuai operasi:

The screenshot shows the Postman interface with the following details:

- HTTP tab is selected.
- Collection: My Collection / Post data
- Method: GET (highlighted with a blue border)
- URL: http://localhost:3000/api
- Request Type dropdown: GET
- Authorization dropdown: x-www-form-urlencoded
- Headers table:

Operasi	Method
Create	POST
Read	GET
Update	PUT / PATCH
Delete	DELETE
- Body table:

"": "Andi Adinat
"": "Alia Adelia
"": "Zen Gunawan
- Left sidebar: GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS
- Bottom: Type a new method

2. Mengisi URL Endpoint

Setiap request diarahkan ke server backend di:

[http://localhost:3000/api/...](http://localhost:3000/api/)

Contoh:

- Tambah staff:
- POST <http://localhost:3000/api/staff>

HTTP My Collection / InfiniRead

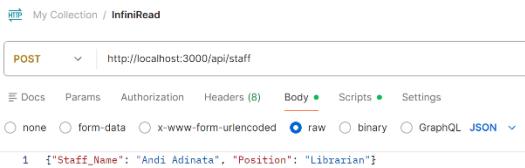
POST http://localhost:3000/api/staff

3. Menambahkan Body JSON (Untuk POST, PUT, PATCH, DELETE)

Masuk ke:

Body → raw → JSON

Lalu isi dengan JSON sesuai yang ditentukan di dokumentasi.



```
POST http://localhost:3000/api/staff
{
  "Staff_Name": "Andi Adinata",
  "Position": "Librarian"
}
```

Contoh POST (insert):

```
{"Staff_Name": "Andi Adinata", "Position": "Librarian"}
```

4. Mengirim Request

Klik tombol:

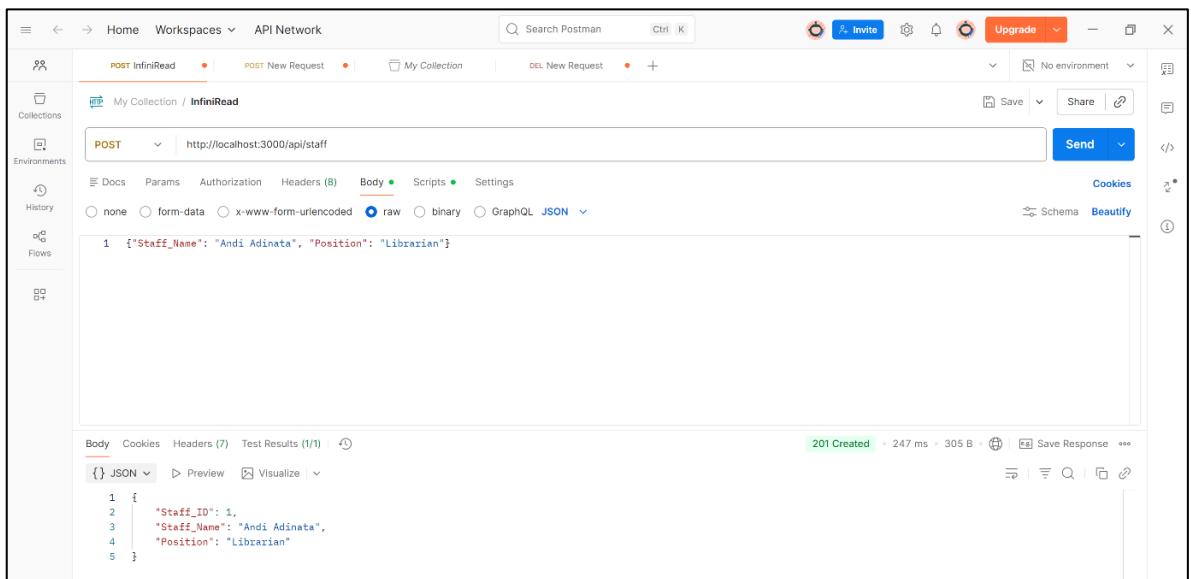
Send

Kemudian lihat hasil inspeksi pada bagian response:

- status code (200, 201, 400, 404)

Code	Arti	Makna
200	OK	Request berhasil
201	Created	Data baru berhasil dibuat
400	Bad Request	Request salah / format keliru
404	Not Found	Data / endpoint tidak ditemukan

- body JSON hasil server
- pesan error jika ada



```
201 Created
{
  "Staff_ID": 1,
  "Staff_Name": "Andi Adinata",
  "Position": "Librarian"
}
```

OPERASI CRUD PADA DATABASE

1. Create Data (POST)

Untuk menambah data baru, digunakan metode POST. Setelah memilih POST dan memasukkan URL endpoint, pengguna membuka *Body* → *raw* → *JSON* lalu menuliskan data sesuai format, misalnya data member atau staff baru. Ketika dikirim, server akan menampilkan respons seperti “*Created*” jika input valid.

A. Contoh Request Single Insert

- Staff

The screenshot shows the Postman interface with a collection named "My Collection". A POST request is made to the URL `http://localhost:3000/api/staff`. The request body is a JSON object:

```
1 {"Staff_Name": "Andi Adinata", "Position": "Librarian"}
```

The response status is `201 Created`, indicating success. The response body is:

```
1 {  
2   "Staff_ID": 1,  
3   "Staff_Name": "Andi Adinata",  
4   "Position": "Librarian"  
5 }
```

- Books

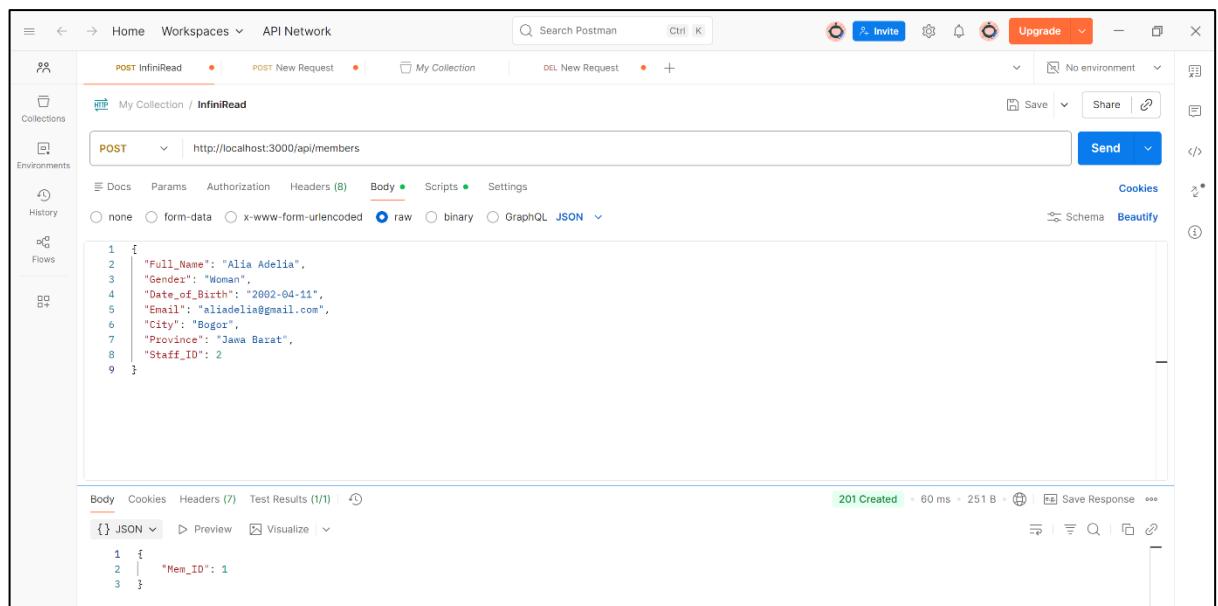
The screenshot shows the Postman interface with a collection named "My Collection". A POST request is made to the URL `http://localhost:3000/api/books`. The request body is a JSON object:

```
1 {  
2   "Book_ID": "SI-000001",  
3   "Category": "Self-Improvement",  
4   "Title": "Aku Punya Kendala, Allah Punya Kendali",  
5   "ISBN": "978-979-794-842-9",  
6   "Pages": 121,  
7   "Pub_Year": 2025,  
8   "Age_Rating": 0,  
9   "Stock": 10,  
10  "Staff_ID": 1  
11 }
```

The response status is `201 Created`, indicating success. The response body is:

```
1 {  
2   "Book_ID": "SI-000001"  
3 }
```

- **Members**



POST InfiniRead

POST New Request

My Collection

DEL New Request

Save Share

Send

Body

```

1 {
2   "Full_Name": "Alia Adelia",
3   "Gender": "Woman",
4   "Date_of_Birth": "2002-04-11",
5   "Email": "aliadelia@gmail.com",
6   "City": "Bogor",
7   "Province": "Java Barat",
8   "Staff_ID": 2
9 }

```

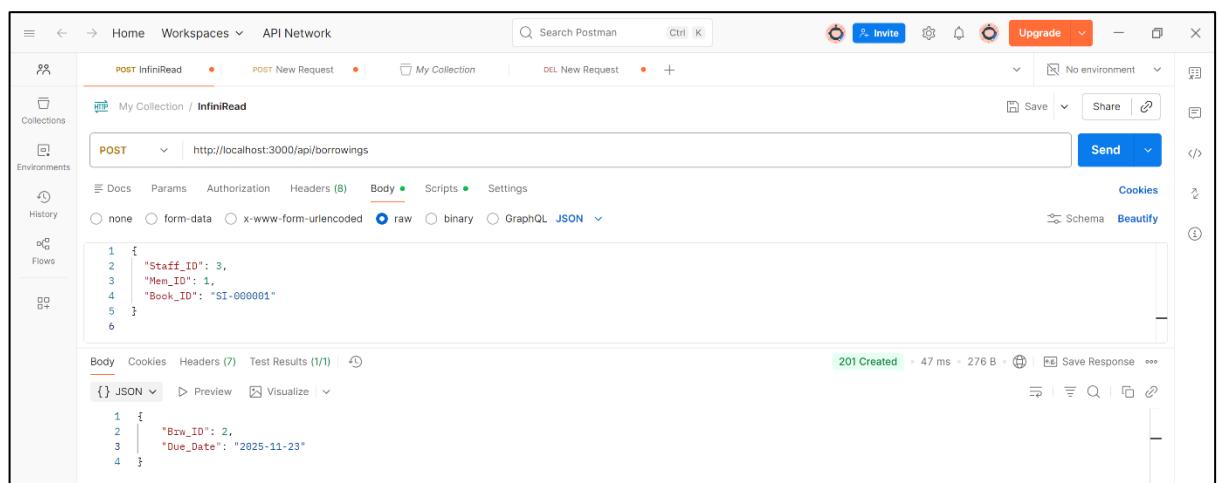
201 Created

Body Cookies Headers (7) Test Results (1/1)

{ } JSON Preview Visualize

1 {
2 "Mem_ID": 1
3 }

- **Borrowings**



POST InfiniRead

POST New Request

My Collection

DEL New Request

Save Share

Send

Body

```

1 {
2   "Staff_ID": 3,
3   "Mem_ID": 1,
4   "Book_ID": "SI-000001"
5 }
6

```

201 Created

Body Cookies Headers (7) Test Results (1/1)

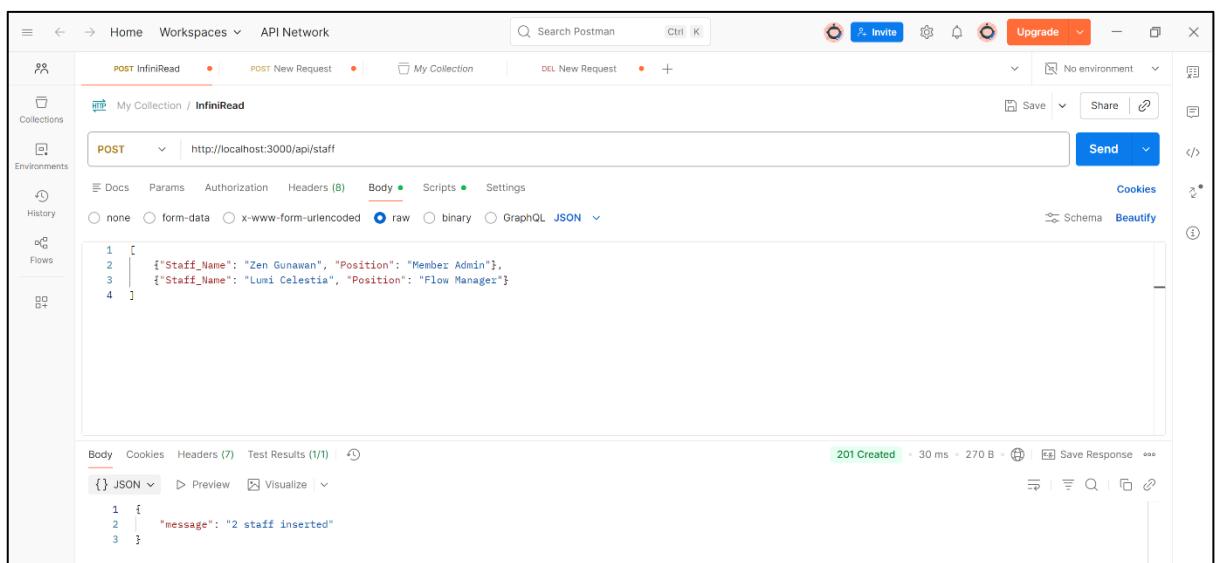
{ } JSON Preview Visualize

1 {
2 "Brw_ID": 2,
3 "Due_Date": "2025-11-23"
4 }

B. Contoh Request Multiple Insert

Sistem mendukung input banyak data dalam satu kali permintaan.

- **Staff**



POST InfiniRead

POST New Request

My Collection

DEL New Request

Save Share

Send

Body

```

1 [
2   {"Staff_Name": "Zen Gunawan", "Position": "Member Admin"},
3   {"Staff_Name": "Lumi Celestia", "Position": "Flow Manager"}
4 ]

```

201 Created

Body Cookies Headers (7) Test Results (1/1)

{ } JSON Preview Visualize

1 {
2 "message": "2 staff inserted"
3 }

● Books

POST /api/books

```

1 [
2   {
3     "Book_ID": "NV-000001", "Category": "Novel", "Title": "Keajaiban Toko Kelontong Namiya", "ISBN": "978-602-064-829-3",
4     "Pages": 400, "Pub_Year": 2020, "Age_Rating": 17, "Stock": 15, "Staff_ID": 1
5   },
6   {
7     "Book_ID": "CP-000001", "Category": "Computer", "Title": "Basis Data Non Relasional", "ISBN": "978-623-713-192-2",
8     "Pages": 188, "Pub_Year": 2025, "Age_Rating": 0, "Stock": 12, "Staff_ID": 1
9   }
10 ]

```

201 Created

```

1 {
2   "message": "2 books inserted"
3 }

```

● Members

POST /api/members

```

1 [
2   {
3     "Full_Name": "Alifa Batrisyia Naziswari", "Gender": "Woman", "Date_of_Birth": "2006-06-24",
4     "Email": "lifabatrisy@gmail.com", "City": "Yogakarta", "Province": "DIY", "Staff_ID": 2
5   },
6   {
7     "Full_Name": "Aulia Fathus Tsan", "Gender": "Woman", "Date_of_Birth": "2006-08-07",
8     "Email": "ullayselay@gmail.com", "City": "Boyalali", "Province": "Jawa Tengah", "Staff_ID": 2
9   },
10  {
11    "Full_Name": "Kukuh Agus Hermawan", "Gender": "Man", "Date_of_Birth": "2006-08-12",
12    "Email": "kukuhagusherawan12@gmail.com", "City": "Kebumen", "Province": "Jawa Tengah", "Staff_ID": 2
13  }
14 ]
15

```

201 Created

```

1 {
2   "message": "3 members inserted"
3 }

```

● Borrowings

POST /api/borrowings

```

1 [
2   {
3     "Staff_ID": 3, "Mem_ID": 3, "Book_ID": "CP-000001"
4   },
5   {
6     "Staff_ID": 3, "Mem_ID": 1, "Book_ID": "NV-000001"
7   }
8 ]
9

```

201 Created

```

1 {
2   "message": "2 borrowings inserted"
3 }

```

C. Request Insert Authors

- Satu author

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/books/SI-000001/authors`. The request body contains a JSON object with a single author:

```
1 {  
2   "authors": ["Salim Aljufri"]  
3 }  
4
```

. The response status is 201 Created, indicating successful insertion.

- Beberapa author

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/books/CP-000001/authors`. The request body contains a JSON object with five authors:

```
1 {  
2   "authors": ["Kemas Rahmat S. W.", "Shaufiah", "Danang Triantoro M.", "Erwin Budi S.", "Anisa Herdiani"]  
3 }  
4
```

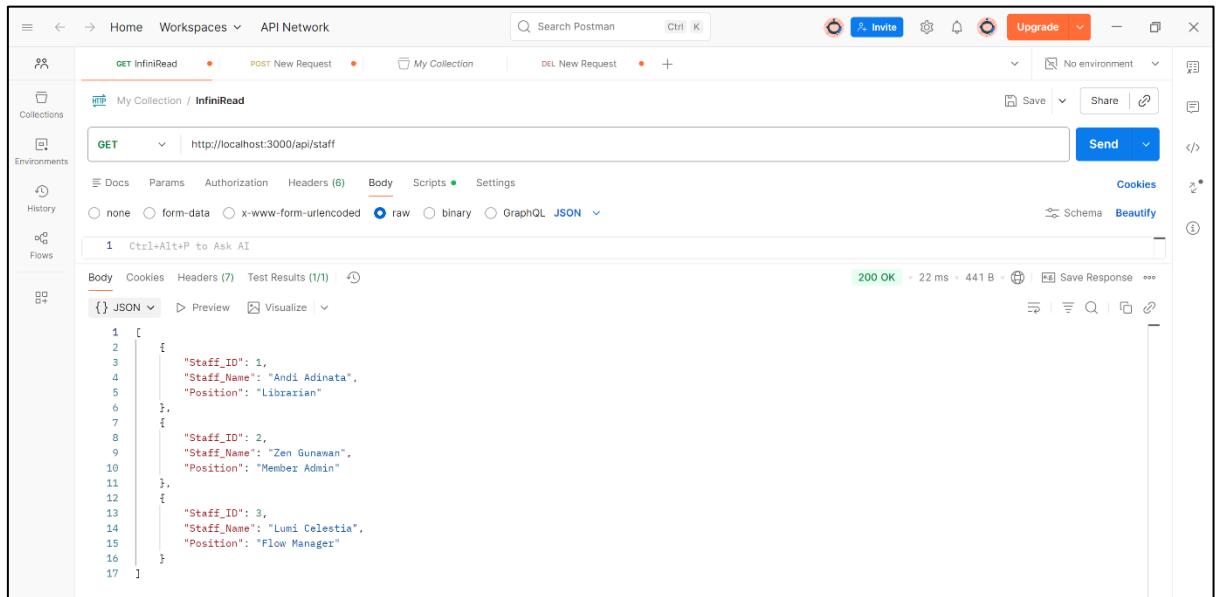
. The response status is 201 Created, indicating successful insertion.

2. Read Data (GET)

Operasi GET digunakan untuk menampilkan data dari server. Pada Postman, cukup memilih metode GET, memasukkan URL endpoint seperti `http://localhost:3000/api/books`, lalu menekan Send. Respons akan ditampilkan dalam bentuk JSON berisi seluruh data pada tabel tersebut.

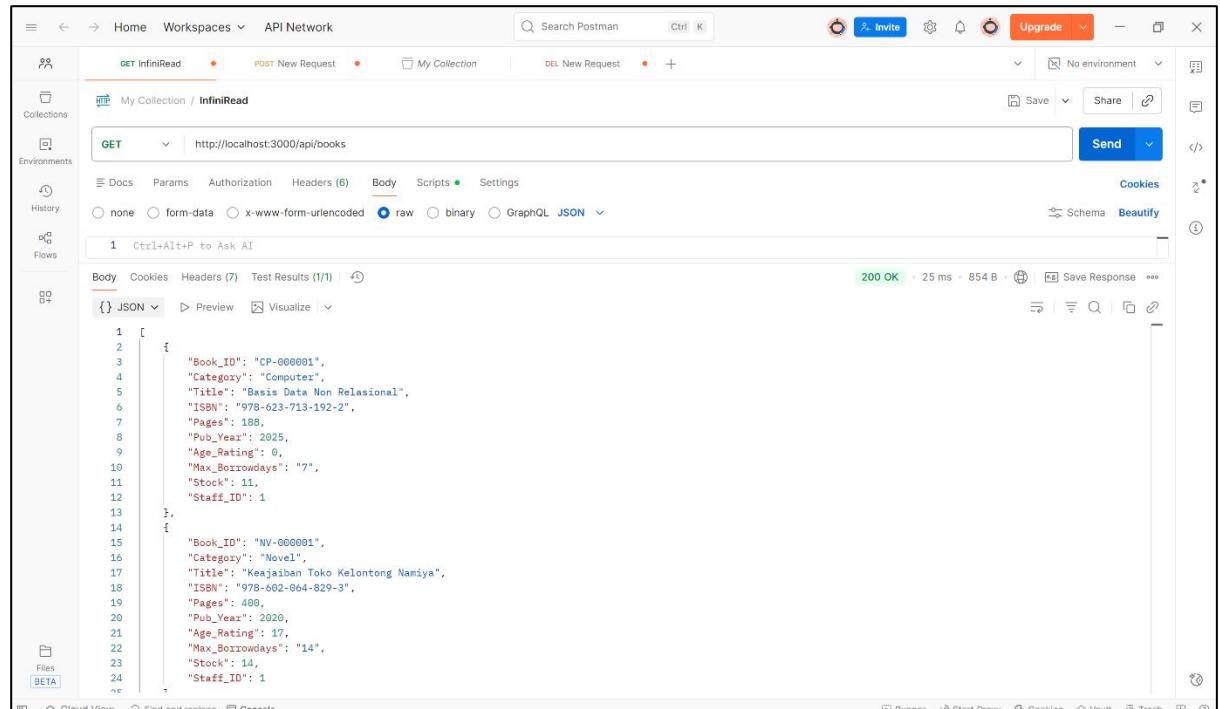
A. Read All

- Staff



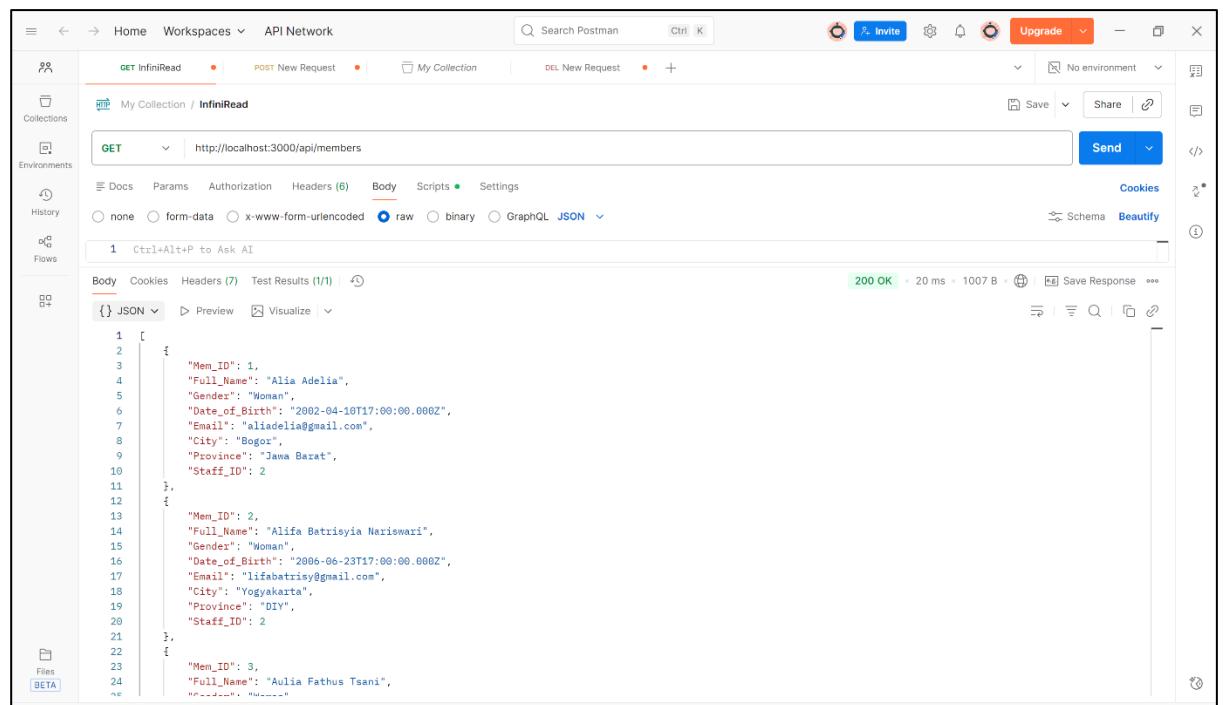
```
1 [  
2   {  
3     "Staff_ID": 1,  
4     "Staff_Name": "Andi Adinata",  
5     "Position": "Librarian"  
6   },  
7   {  
8     "Staff_ID": 2,  
9     "Staff_Name": "Zen Gunawan",  
10    "Position": "Member Admin"  
11  },  
12  {  
13    "Staff_ID": 3,  
14    "Staff_Name": "Lumi Celestia",  
15    "Position": "Flow Manager"  
16  }  
17 ]
```

- Books



```
1 [  
2   {  
3     "Book_ID": "CP-000001",  
4     "Category": "Computer",  
5     "Title": "Basis Data Non Relasional",  
6     "ISBN": "978-623-713-192-2",  
7     "Pages": 188,  
8     "Pub_Year": 2025,  
9     "Age_Rating": 0,  
10    "Max_Borrowdays": "7",  
11    "Stock": 11,  
12    "Staff_ID": 1  
13  },  
14  {  
15    "Book_ID": "NV-000001",  
16    "Category": "Novel",  
17    "Title": "Keajaiban Toko Kelontong Namiya",  
18    "ISBN": "978-602-864-829-3",  
19    "Pages": 400,  
20    "Pub_Year": 2020,  
21    "Age_Rating": 17,  
22    "Max_Borrowdays": "14",  
23    "Stock": 14,  
24    "Staff_ID": 1  
25  }]
```

• Members

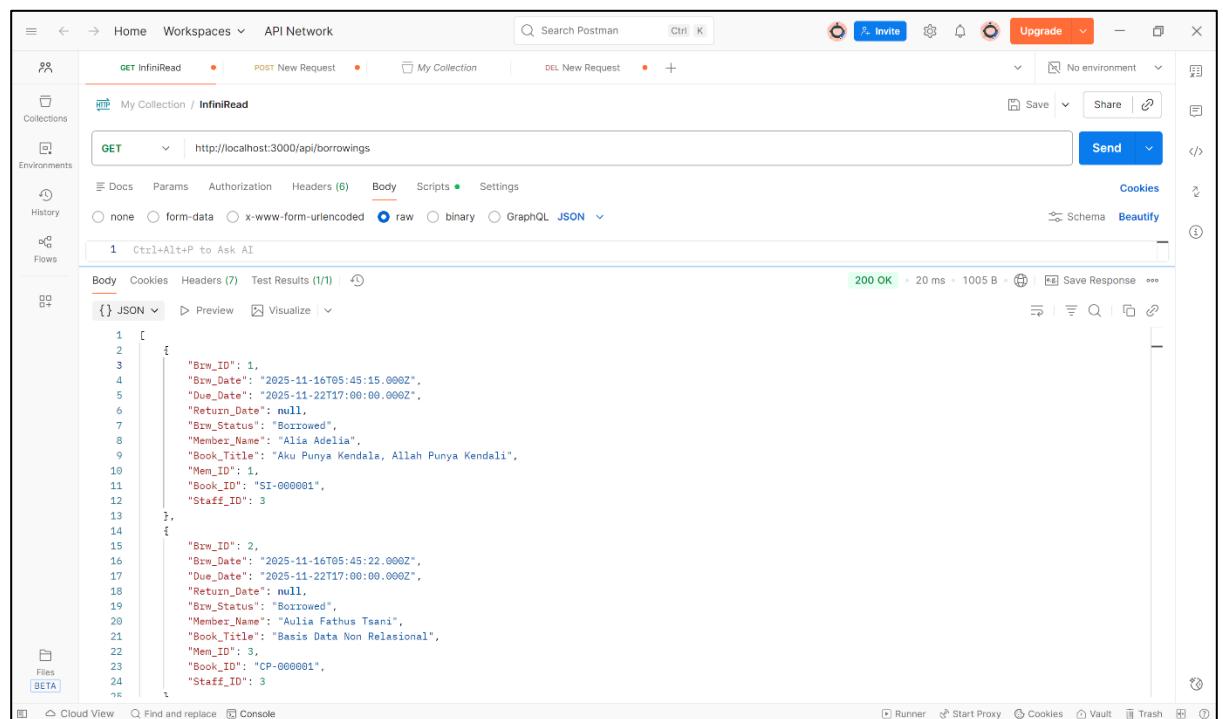


```

GET /api/members
HTTP/1.1
Host: localhost:3000
Content-Type: application/json

[{"Mem_ID": 1, "Full_Name": "Alia Adelia", "Gender": "Woman", "Date_of_Birth": "2002-04-10T17:00:00.000Z", "Email": "alialidia@gmail.com", "City": "BoGOR", "Province": "Jawa Barat", "Staff_ID": 2}, {"Mem_ID": 2, "Full_Name": "Alifa Batrisyia Nariswari", "Gender": "Woman", "Date_of_Birth": "2006-06-23T17:00:00.000Z", "Email": "lifabatrisy@gmail.com", "City": "Yogyakarta", "Province": "DIY", "Staff_ID": 2}, {"Mem_ID": 3, "Full_Name": "Aulia Fathus Tsani", "Gender": "Woman", "Date_of_Birth": "2004-01-01T17:00:00.000Z", "Email": "auliafathus@gmail.com", "City": "Surabaya", "Province": "Jawa Timur", "Staff_ID": 3}]
  
```

• Borrowings



```

GET /api/borrowings
HTTP/1.1
Host: localhost:3000
Content-Type: application/json

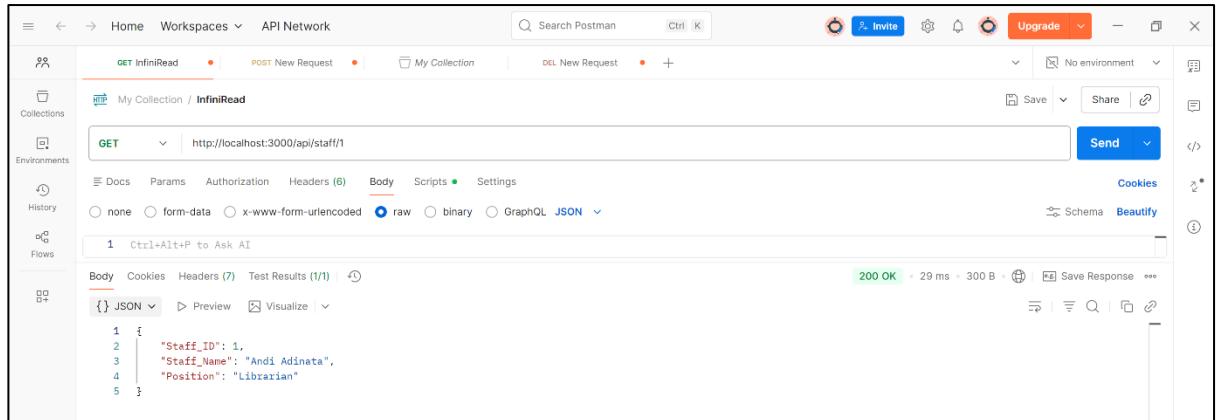
[{"Brw_ID": 1, "Brw_Date": "2025-11-15T05:45:15.000Z", "Due_Date": "2025-11-22T17:00:00.000Z", "Return_Date": null, "Brw_Status": "Borrowed", "Member_Name": "Alia Adelia", "Book_Title": "Aku Punya Kendala, Allah Punya Kendali", "Mem_ID": 1, "Book_ID": "SI-000001", "Staff_ID": 3}, {"Brw_ID": 2, "Brw_Date": "2025-11-16T05:45:22.000Z", "Due_Date": "2025-11-22T17:00:00.000Z", "Return_Date": null, "Brw_Status": "Borrowed", "Member_Name": "Aulia Fathus Tsani", "Book_Title": "Basis Data Non Relasional", "Mem_ID": 3, "Book_ID": "CP-000001", "Staff_ID": 3}]
  
```

Format waktu seperti `2025-11-15T17:00:00.000Z` muncul karena ketika data bertipe DATE diambil dari MySQL, Node.js secara otomatis mengubahnya menjadi objek Date JavaScript dan menampilkannya dalam format standar internasional ISO 8601, yang selalu menyertakan jam dan zona waktu meskipun di database hanya tersimpan

tanggalnya saja. Akibat konversi ini, data DATE yang sederhana berubah menjadi format lengkap dengan waktu bawaan (biasanya 00:00:00 diubah ke UTC).

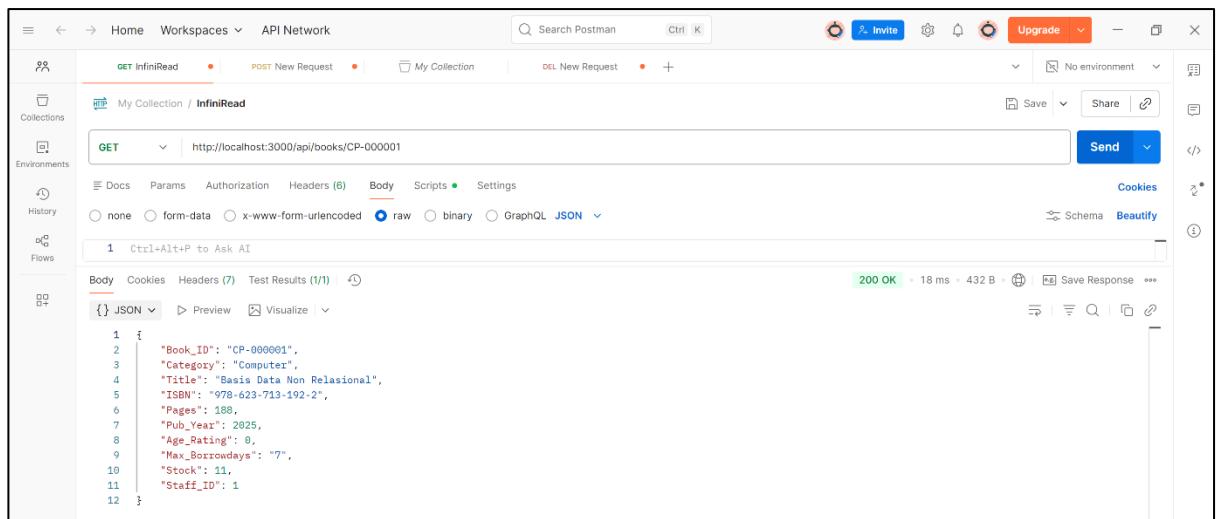
B. Read by ID

- Staff



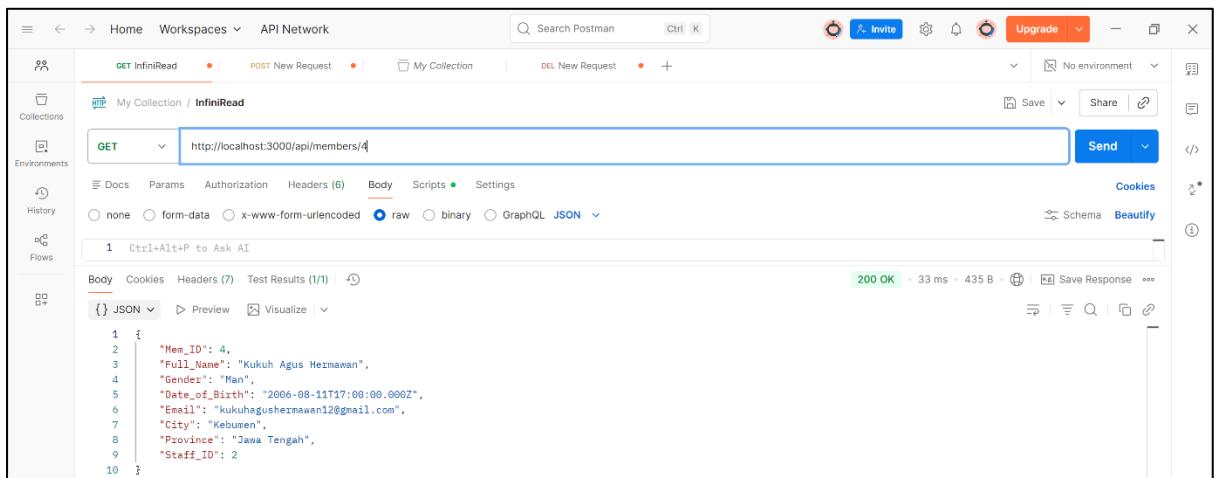
```
1 {
2   "Staff_ID": 1,
3   "Staff_Name": "Andi Adinata",
4   "Position": "Librarian"
5 }
```

- Books



```
1 {
2   "Book_ID": "CP-000001",
3   "Category": "Computer",
4   "Title": "Basis Data Non Relasional",
5   "ISBN": "978-623-713-192-2",
6   "Pages": 188,
7   "Pub_Year": 2025,
8   "Age_Rating": 0,
9   "Max_Borrowdays": "7",
10  "Stock": 11,
11  "Staff_ID": 1
12 }
```

- Members



```
1 {
2   "Mem_ID": 4,
3   "Full_Name": "Kukuh Agus Hermawan",
4   "Gender": "Man",
5   "Date_of_Birth": "2006-08-11T17:00:00.000Z",
6   "Email": "kukuhagushermawan12@gmail.com",
7   "City": "Kebumen",
8   "Province": "Jawa Tengah",
9   "Staff_ID": 2
10 }
```

- **Borrowings**

Postman screenshot showing a successful GET request to `http://localhost:3000/api/borrowings/2`. The response status is 200 OK, and the response body is a JSON object:

```

1 {
2   "Bw_ID": 2,
3   "Bw_Date": "2025-11-16T05:45:22.000Z",
4   "Due_Date": "2025-11-22T17:00:00.000Z",
5   "Return_Date": null,
6   "Bw_Status": "Borrowed",
7   "Member_Name": "Aulia Fathus Tsani",
8   "Book_Title": "Basis Data Non Relasional",
9   "Mem_ID": 3,
10  "Book_ID": "CP-0000001",
11  "Staff_ID": 3
12 }

```

3. Update Data (PUT/PATCH)

Operasi Update digunakan untuk mengubah data yang sudah ada. (menggunakan PUT [harus input keseluruhan data] atau PATCH [cukup data tertentu yang mau di edit]). Pengguna menuliskan URL endpoint yang sama, misalnya /api/books, kemudian mengisi *Body → JSON* dengan data yang ingin diperbarui, termasuk ID sebagai acuan. Setelah menekan Send, Postman akan menampilkan respons apakah data berhasil diperbarui atau terjadi kesalahan input.

- **Staff**

Postman screenshot showing a successful PUT request to `http://localhost:3000/api/staff/1`. The response status is 200 OK, and the response body is a JSON object:

```

1 {
2   "Staff_Name": "Arjuna Arkana", "Position": "Librarian"
3 }
4

```

Body:

```

1 {
2   "message": "Staff updated (PUT)"
3 }

```

- **Books**

Postman screenshot showing a successful PATCH request to `http://localhost:3000/api/books/SI-000001`. The response status is 200 OK, and the response body is a JSON object:

```

1 {
2   "Stock": 25
3 }
4

```

Body:

```

1 {
2   "success": true,
3   "updated": {
4     "Stock": 25
5   }
6 }

```

- **Members**

The screenshot shows the Postman interface with a PATCH request to `http://localhost:3000/api/members/1`. The request body contains the JSON `{"Full_Name": "Aya Aulya"}`. The response status is 200 OK, and the response body is `{"message": "Member updated (PATCH)", "updated": {"Full_Name": "Aya Aulya"}}`.

```

PATCH http://localhost:3000/api/members/1
{
  "Full_Name": "Aya Aulya"
}
{
  "message": "Member updated (PATCH)",
  "updated": {
    "Full_Name": "Aya Aulya"
  }
}
  
```

- **Borrowings**

The screenshot shows the Postman interface with a PATCH request to `http://localhost:3000/api/borrowings/2`. The request body contains the JSON `{"Brw_Status": "Returned"}`. The response status is 200 OK, and the response body is `{"success": true, "updated": {"Brw_Status": "Returned", "Return_Date": "2025-11-16 12:48:02"}}`.

```

PATCH http://localhost:3000/api/borrowings/2
{
  "Brw_Status": "Returned"
}
{
  "success": true,
  "updated": {
    "Brw_Status": "Returned",
    "Return_Date": "2025-11-16 12:48:02"
  }
}
  
```

Ubah data author

- Put: langsung mengedit keseluruhan

The screenshot shows the Postman interface with a PUT request to `http://localhost:3000/api/books/CP-000001/authors`. The request body contains the JSON `{"authors": ["Kemas Rahmat Saleh Wihaja", "Shaufiah", "Danang Triantoro M.", "Erwin Budi S.", "Anisa Herdiani"]}`. The response status is 200 OK, and the response body is `{"message": "Authors fully replaced (PUT)", "Book_ID": "CP-000001", "authors": ["Kemas Rahmat Saleh Wihaja", "Shaufiah", "Danang Triantoro M.", "Erwin Budi S.", "Anisa Herdiani"]}`.

```

PUT http://localhost:3000/api/books/CP-000001/authors
{
  "authors": [
    "Kemas Rahmat Saleh Wihaja",
    "Shaufiah",
    "Danang Triantoro M.",
    "Erwin Budi S.",
    "Anisa Herdiani"
  ]
}
{
  "message": "Authors fully replaced (PUT)",
  "Book_ID": "CP-000001",
  "authors": [
    "Kemas Rahmat Saleh Wihaja",
    "Shaufiah",
    "Danang Triantoro M.",
    "Erwin Budi S.",
    "Anisa Herdiani"
  ]
}
  
```

- Patch: mengedit author tertentu satu persatu

The screenshot shows the Postman interface with a PATCH request to `http://localhost:3000/api/books/CP-000001/authors`. The request body is raw JSON:

```

1 {
2   "author": "Danang Triantozo M.",
3   "new_author": "Danang Triantozo Murdiansyah"
4 }
5

```

The response status is 200 OK, with a message indicating the author was updated.

4. Delete Data (DELETE)

Operasi DELETE digunakan untuk menghapus data berdasarkan ID. Pada Postman, pilih metode DELETE, masukkan URL endpoint, lalu pada Body → raw → JSON isi format `{"ids": [ID]}` untuk menghapus satu atau beberapa data sekaligus. Setelah menekan Send, server akan menghapus ID yang valid dan menampilkan respons sukses, sementara ID yang tidak ditemukan atau format yang salah akan menghasilkan pesan error.

A. Delete by ID

- Staff

The screenshot shows the Postman interface with a DELETE request to `http://localhost:3000/api/staff/1`. The response status is 200 OK, with a success message.

- Books

The screenshot shows the Postman interface with a DELETE request to `http://localhost:3000/api/books/SI-000001`. The response status is 200 OK, with a success message.

- **Members**

Postman screenshot showing a successful DELETE request to `http://localhost:3000/api/members/1`. The response status is `200 OK` with a success message: "Member & related borrowings deleted".

```

1 {
2   "success": true,
3   "message": "Member & related borrowings deleted"
4 }
    
```

- **Borrowings**

Postman screenshot showing a successful DELETE request to `http://localhost:3000/api/borrowings/2`. The response status is `200 OK` with a success message: "success: true".

```

1 {
2   "success": true
3 }
    
```

B. Delete Multiple Books (Bulk Delete)

Sistem juga mendukung penghapusan banyak data sekaligus.

- **Staff**

Postman screenshot showing a successful DELETE request to `http://localhost:3000/api/staff` with a JSON body containing multiple IDs: `[{"ids": ["2", "3"]}]`. The response status is `200 OK` with a success message: "2 staff deleted".

```

1 [
2   "ids": ["2", "3"]
3 ]
4
    
```

```

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

- Books

DELETE http://localhost:3000/api/books

```
{
  "ids": ["NV-000001", "CP-000001"]
}
```

Body Cookies Headers (7) Test Results (1/1)

```
{
  "message": "2 books deleted"
}
```

200 OK 43 ms 264 B Save Response

- Members

DELETE http://localhost:3000/api/members

```
{
  "ids": [2, 3, 4]
}
```

Body Cookies Headers (7) Test Results (1/1)

```
{
  "message": "3 members deleted"
}
```

200 OK 45 ms 266 B Save Response

- Borrowings

DELETE http://localhost:3000/api/borrowings

```
{
  "ids": [1, 3]
}
```

Body Cookies Headers (7) Test Results (1/1)

```
{
  "message": "2 borrowings deleted"
}
```

200 OK 39 ms 269 B Save Response

Kesimpulan:

Pengujian fitur CRUD pada sistem *InfiniRead* melalui Postman menunjukkan bahwa seluruh operasi Create, Read, Update, dan Delete dapat dijalankan dengan baik pada semua tabel, termasuk Staff, Members, Books, Borrowings, dan Book_Author. Setiap request berhasil diproses oleh server, mulai dari penambahan data baru, pengambilan data, pembaruan data, hingga penghapusan data tunggal maupun batch. Seluruh respons API dikembalikan dalam

format JSON. Validasi input dan pesan kesalahan juga berjalan sesuai aturan sehingga menjaga integritas data. Secara keseluruhan, pengujian ini membuktikan bahwa backend InfiniRead telah berfungsi stabil dan siap untuk tahap integrasi dengan user interface.