



Assignment for ITS 4243 - Microservices and Cloud Computing

University of Sri Jayewardenepura – Faculty of Technology

Student Name: Daluwatta H.N.S.

Student ID: ICT/21/820

Smart Travel Booking Platform

Github Repository: <https://github.com/NadunDalu/smart-travel-platform>

Postman Collection: <https://github.com/NadunDalu/smart-travel-platform/tree/main/postman/collections>

Architecture Overview

Services

1. **User Service** (Port 8081) - Manages user information
2. **Flight Service** (Port 8082) - Manages flight details and availability
3. **Hotel Service** (Port 8083) - Manages hotel details and availability
4. **Booking Service** (Port 8084) - Main orchestrator for booking workflow
5. **Payment Service** (Port 8085) - Handles payment processing
6. **Notification Service** (Port 8086) - Sends notifications to users

Communication Flow

1. User sends booking request to Booking Service
2. Booking Service → User Service (WebClient)
 - Validates user exists and is active
3. Booking Service → Flight Service (Feign Client)
 - Checks flight availability
4. Booking Service → Hotel Service (Feign Client)
 - Checks hotel availability
5. Booking Service
 - Calculates total cost
 - Stores booking as PENDING
6. Booking Service → Notification Service (WebClient)
 - Sends booking created notification

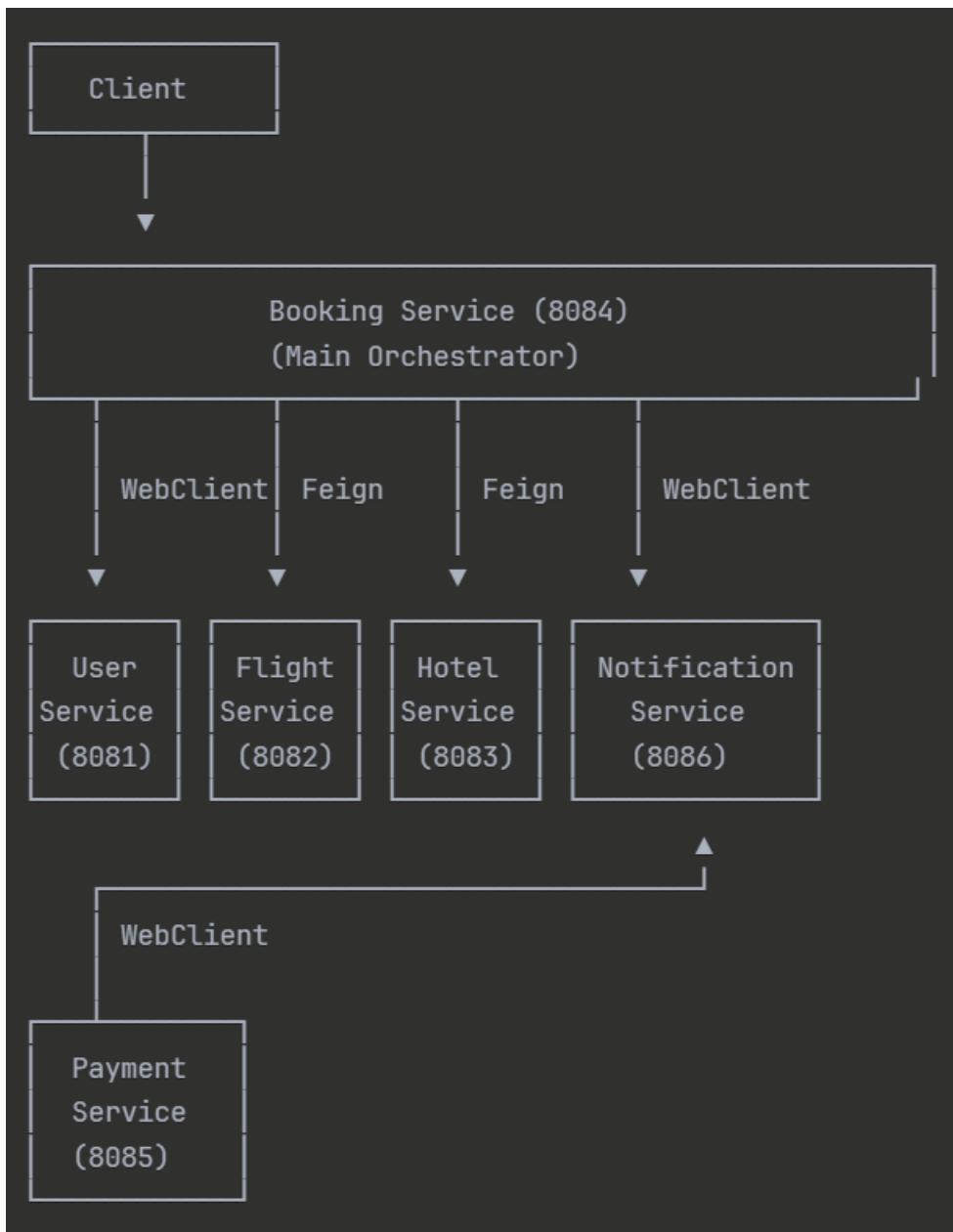
7. Payment Service → Booking Service (WebClient)

- Processes payment
- Updates booking status to CONFIRMED

8. Booking Service → Notification Service (WebClient)

- Sends booking confirmed notification

Architecture Diagram



Setup Instructions

Used tools

- Java 17
- Maven 3.6+
- MySQL 8.0+
- Postman (for testing)

Step 1: Database Setup

Create 6 MySQL databases

```
CREATE DATABASE user_db;  
CREATE DATABASE flight_db;  
CREATE DATABASE hotel_db;  
CREATE DATABASE booking_db;  
CREATE DATABASE payment_db;  
CREATE DATABASE notification_db;
```

Step 2: Clone and Setup

Create project directory

```
mkdir smart-travel-platform  
cd smart-travel-platform
```

Create 6 service directories

```
mkdir user-service flight-service hotel-service booking-service payment-service notification-service
```

Step 3: Configure Database Password

Update application.properties in each service

```
spring.datasource.password= mysql_password
```

Step 4: Build All Services

Build each service

```
cd user-service
```

```
mvn clean install
```

```
cd ..
```

```
cd flight-service
```

```
mvn clean install
```

```
cd ..
```

```
cd hotel-service
```

```
mvn clean install
```

```
cd ..
```

```
cd notification-service
```

```
mvn clean install
```

```
cd ..
```

```
cd payment-service
```

```
mvn clean install
```

```
cd ..
```

```
cd booking-service
```

```
mvn clean install
```

```
cd ..
```

Step 5: Run All Services

Important: Start services in this order:

Terminal 1 - User Service

```
cd user-service  
mvn spring-boot:run
```

Terminal 2 - Flight Service

```
cd flight-service  
mvn spring-boot:run
```

Terminal 3 - Hotel Service

```
cd hotel-service  
mvn spring-boot:run
```

Terminal 4 - Notification Service

```
cd notification-service  
mvn spring-boot:run
```

Terminal 5 - Payment Service

```
cd payment-service  
mvn spring-boot:run
```

Terminal 6 - Booking Service

```
cd booking-service  
mvn spring-boot:run
```

API Testing

1. Create a User

The screenshot shows the Postman application interface. In the top navigation bar, the workspace is set to "smart-travel-platform" and the request type is "POST New Request". The URL is "http://localhost:8081/api/users". The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   "name": "Hizuni Prabodhya",
3   "email": "hizuni@example.com",
4   "phone": "+94771234567",
5   "address": "123 Main St, Colombo"
6 }
```

Below the body, the response is displayed under the "Test Results" tab, showing a status of "201 Created" with a response time of 322 ms and a size of 302 B. The response JSON is:

```
1 {
2   "id": 1,
3   "name": "Hizuni Prabodhya",
4   "email": "hizuni@example.com",
5   "phone": "+94771234567",
6   "address": "123 Main St, Colombo",
7   "active": true
8 }
```

2. Create a Flight

The screenshot shows the Postman application interface. In the top navigation bar, the workspace is set to "smart-travel-platform" and the request type is "POST Create a User" (which has been renamed to "Create a Flight"). The URL is "http://localhost:8082/api/flights". The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   "flightNumber": "SL101",
3   "airline": "SriLankan Airlines",
4   "origin": "Colombo",
5   "destination": "Dubai",
6   "departureTime": "2026-01-10 08:00",
7   "arrivalTime": "2026-01-10 12:00",
8   "price": 45000.00,
9   "availableSeats": 50
10 }
```

Below the body, the response is displayed under the "Test Results" tab, showing a status of "201 Created" with a response time of 227 ms and a size of 394 B. The response JSON is:

```
1 {
2   "id": 1,
3   "flightNumber": "SL101",
4   "airline": "SriLankan Airlines",
5   "origin": "Colombo",
6   "destination": "Dubai",
7   "departureTime": "2026-01-10 08:00",
8   "arrivalTime": "2026-01-10 12:00",
9   "price": 45000.00,
10   "availableSeats": 50,
11   "available": true
12 }
```

3. Create a Hotel

The screenshot shows the Postman interface with a collection named "smart-travel-platform". A new request is being prepared for "Create a Hotel" at `http://localhost:8083/api/hotels`. The request method is set to POST. The body contains the following JSON payload:

```
1 {
2   "name": "Hilton Colombo",
3   "location": "Colombo",
4   "address": "2 Sir Chittampalam A Gardiner Mawatha",
5   "rating": 5,
6   "pricePerNight": 15000.00,
7   "availableRooms": 20,
8   "amenities": "WiFi, Pool, Gym, Spa"
9 }
```

The response status is 201 Created, indicating success. The response body is identical to the request body.

4. Create a Booking

The screenshot shows the Postman interface with a collection named "smart-travel-platform". A new request is being prepared for "Create a Booking (Main Flow)" at `http://localhost:8084/api/bookings`. The request method is set to POST. The body contains the following JSON payload:

```
1 {
2   "userId": 1,
3   "flightId": 1,
4   "hotelId": 1,
5   "travelDate": "2025-01-10"
6 }
```

The response status is 201 Created, indicating success. The response body is identical to the request body.

5. Process Payment

The screenshot shows the Postman application interface. A collection named "smart-travel-platform" is open, containing several requests. The current request is a POST to "http://localhost:8085/api/payments/process". The request body is set to raw JSON:

```
1 {  
2     "bookingId": 1,  
3     "amount": 66000.00,  
4     "paymentMethod": "CREDIT_CARD"  
5 }
```

The response status is 201 Created, with a response time of 3.16 s and a size of 359 B. The response body is:

```
1 {  
2     "id": 1,  
3     "bookingId": 1,  
4     "amount": 66000.00,  
5     "paymentMethod": "CREDIT_CARD",  
6     "status": "COMPLETED",  
7     "transactionId": "64524424-c464-48ed-8757-b9a1b63a8a45",  
8     "paymentDate": "2025-12-11T15:46:10Z"  
9 }
```

6. Verify Booking Status

The screenshot shows the Postman application interface. A collection named "smart-travel-platform" is open, containing several requests. The current request is a GET to "http://localhost:8084/api/bookings/1". The request includes a "Query Params" section with a single entry: "Key" and "Value".

The response status is 200 OK, with a response time of 25 ms and a size of 319 B. The response body is:

```
1 {  
2     "id": 1,  
3     "userId": 1,  
4     "flightId": 1,  
5     "hotelId": 1,  
6     "travelDate": "2025-01-10",  
7     "totalCost": 66000.00,  
8     "status": "CONFIRMED",  
9     "bookingDate": "2025-12-11T15:43:24.371174"  
10 }
```

7. Check Notifications

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, and API Network. The main workspace is titled "smart-travel-platform / Check Notifications". A search bar at the top right contains the placeholder "Search Postman". On the far right, there are buttons for "Invite", "Upgrade", and "No environment". Below the workspace title, there are several API endpoints listed: POST Create a User, POST Create a Flight, POST Create a Hotel, POST Create a Booking, POST Process Payment, GET Verify Booking Status, and GET Check Notification. The "Check Notification" endpoint is currently selected. The main area shows a GET request to "http://localhost:8086/api/notifications/user/1". The "Params" tab is selected, showing a table with one row and two columns: "Key" and "Value". The "Body" tab is also visible, showing a JSON response. The response body is a JSON array containing two objects, each representing a notification:

```
[{"id": 1, "userId": 1, "bookingId": 1, "message": "Your booking has been created and is pending payment confirmation.", "type": "EMAIL", "status": "SENT", "sentAt": "2025-12-11T15:43:25.445076"}, {"id": 2, "userId": 1, "bookingId": 1, "message": "Your booking has been confirmed! Payment received.", "type": "EMAIL", "status": "SENT", "sentAt": "2025-12-11T15:45:48.341562"}]
```

The status bar at the bottom indicates a 200 OK response with 609 ms and 513 B. There are also buttons for "Save Response", "Runner", "Start Proxy", "Cookies", "Vault", and "Trash".

Database

```
MySQL> use user_db
Database changed
MySQL> show tables;
+-----+
| Tables_in_user_db |
+-----+
| users             |
+-----+
1 row in set (0.02 sec)

MySQL> select*from users;
+----+-----+-----+-----+-----+
| id | active | address          | email           | name            | phone           |
+----+-----+-----+-----+-----+
| 1  | 0x01   | 123 Main St, Colombo | hiruni@example.com | Hiruni Prabodhya | +94771234567 |
+----+-----+-----+-----+-----+
1 row in set (0.01 sec)

MySQL> show databases;
+-----+
| Database |
+-----+
| booking_db |
| course_db |
| enrollment_db |
| flight_db |
| hotel_db |
| information_schema |
| mysql |
| notification_db |
| payment_db |
| performance_schema |
| result_db |
| student_db |
| student_management |
| sys |
| user_db |
+-----+
15 rows in set (0.00 sec)

MySQL> 
1 row in set (0.00 sec)

MySQL> use payment_db;
Database changed
MySQL> show tables;
+-----+
| Tables_in_payment_db |
+-----+
| payments             |
+-----+
1 row in set (0.00 sec)

MySQL> select*from payments;
+----+-----+-----+-----+-----+
| id | amount | booking_id | payment_date          | payment_method | status        | transaction_id |
+----+-----+-----+-----+-----+
| 1  | 60000.00 | 1          | 2025-12-11 15:45:46.191822 | CREDIT_CARD    | COMPLETED    | 04524424-c404-48ed-8757-b9a1b63a8a45 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

MySQL> use notification_db;
Database changed
MySQL> show tables;
+-----+
| Tables_in_notification_db |
+-----+
| notifications             |
+-----+
1 row in set (0.00 sec)

MySQL> select*from notifications;
+----+-----+-----+-----+-----+-----+
| id | booking_id | message                                | sent_at          | status      | type       | user_id |
+----+-----+-----+-----+-----+-----+
| 1  | 1          | Your booking has been created and is pending payment confirmation. | 2025-12-11 15:43:25.445076 | SENT        | EMAIL      | 1         |
| 2  | 1          | Your booking has been confirmed! Payment received.                | 2025-12-11 15:45:48.341562 | SENT        | EMAIL      | 1         |
+----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

MySQL> |
```

```

MySQL 8.0 Command Line Cli + ×

mysql> use hotel_db;
Database changed
mysql> show tables;
+-----+
| Tables_in_hotel_db |
+-----+
| hotels             |
+-----+
1 row in set (0.00 sec)

mysql> select*from hotels;
+-----+
| id | address           | amenities          | available    | available_rooms | location      | name        | price_per_night |
|----+-----+-----+-----+-----+-----+-----+-----+
| 1  | 2 Sir Chittampalam A Gardiner Mawatha | WiFi, Pool, Gym, Spa | 0x01          | 20            | Colombo     | Hilton Colombo | 15000.00      |
|----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> use booking_db;
Database changed
mysql> show tables;
+-----+
| Tables_in_booking_db |
+-----+
| bookings            |
+-----+
1 row in set (0.00 sec)

mysql> select*from bookings;
+-----+
| id | booking_date       | flight_id | hotel_id | status   | total_cost | travel_date | user_id |
|----+-----+-----+-----+-----+-----+-----+-----+
| 1  | 2025-12-11 15:43:24.371174 | 1         | 1         | CONFIRMED | 60000.00   | 2025-01-10 | 1         |
|----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

MySQL 8.0 Command Line Cli + ×

mysql> use flight_db;
Database changed
mysql> show tables;
+-----+
| Tables_in_flight_db |
+-----+
| flights             |
+-----+
1 row in set (0.00 sec)

mysql> select*from flights;
+-----+
| id | airline           | arrival_time | available    | available_seats | departure_time | destination | flight_number | origin    | price |
|----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SriLankan Airlines | 2026-01-10 12:00 | 0x01          | 50            | 2026-01-10 08:00 | Dubai       | SL101      | Colombo  | 45000.00 |
|----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> use hotel_db;
Database changed
mysql> show tables;
+-----+
| Tables_in_hotel_db |
+-----+
| hotels             |
+-----+
1 row in set (0.00 sec)

mysql> select*from hotels;
+-----+
| id | address           | amenities          | available    | available_rooms | location      | name        | price_per_night |
|----+-----+-----+-----+-----+-----+-----+-----+

```

Technology Stack

- **Framework:** Spring Boot 3.2.0
- **Language:** Java 17

- **Database:** MySQL 8.0
- **Communication:**
 - Spring Cloud OpenFeign (Feign Client)
 - Spring WebFlux (WebClient)
 - REST APIs
- **ORM:** Spring Data JPA
- **Build Tool:** Maven
- **Libraries:** Lombok

Communication Technologies Used

Feign Client

- Booking Service → Flight Service
- Booking Service → Hotel Service

WebClient

- Booking Service → User Service
- Booking Service → Notification Service
- Payment Service → Booking Service

Troubleshooting

Service Won't Start

- Check if the port is already in use
- Verify MySQL is running
- Check database credentials in application.properties

Feign Client Error

- Ensure Flight and Hotel services are running
- Check the URLs in booking-service application.properties

WebClient Timeout

- Increase timeout in application.properties
- Verify target service is responding

Database Connection Error

- Verify MySQL is running
- Check database exists
- Verify username/password

END.

Annexure 1 – Postman Collection

```
{  
  "info": {  
    "_postman_id": "21eec738-565e-474b-ba2c-727ee29be48d",  
    "name": "smart-travel-platform",  
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",  
    "_exporter_id": "49881238",  
    "_collection_link": "https://nadundaluwatta0426-4321055.postman.co/workspace/9433a909-f925-4013-839f-df80b41484e6/collection/49881238-21eec738-565e-474b-ba2c-727ee29be48d?action=share&source=collection_link&creator=49881238"  
  },  
  "item": [  
    {  
      "name": "Create a User",  
      "id": "453dcc1e-c550-4922-a1b4-848270a60876",  
      "request": {  
        "method": "POST",  
        "header": [  
          {  
            "key": "Content-Type",  
            "value": "application/json",  
            "type": "text"  
          }  
        ]  
      }  
    }  
  ]  
}
```

```
],
  "body": {
    "mode": "raw",
    "raw": "{\r\n  \"name\": \"Hiruni Prabodhya\",\r\n  \"email\": \"hiruni@example.com\",\r\n  \"phone\": "+94771234567",\r\n  \"address\": \"123 Main St,\nColombo\"\r\n}",
    "options": {
      "raw": {
        "language": "json"
      }
    }
  },
  "url": {
    "raw": "http://localhost:8081/api/users",
    "protocol": "http",
    "host": [
      "localhost"
    ],
    "port": "8081",
    "path": [
      "api",
      "users"
    ]
  }
},
  "response": []
},
{
  "name": "Create a Flight",
  "id": "c7e84e6c-a228-4381-9147-3b8f51e262bf",
  "request": {
    "method": "POST",
    "header": [
      {
        "key": "Content-Type",
        "value": "application/json",
        "type": "text"
      }
    ]
  }
}
```

```
],
  "body": {
    "mode": "raw",
    "raw": "{\r\n  \"flightNumber\": \"SL101\", \r\n  \"airline\": \"SriLankan Airlines\", \r\n  \"origin\": \"Colombo\", \r\n  \"destination\": \"Dubai\", \r\n  \"departureTime\": \"2026-01-10 08:00\", \r\n  \"arrivalTime\": \"2026-01-10 12:00\", \r\n  \"price\": 45000.00, \r\n  \"availableSeats\": 50\r\n}",
    "options": {
      "raw": {
        "language": "json"
      }
    },
    "url": {
      "raw": "http://localhost:8082/api/flights",
      "protocol": "http",
      "host": [
        "localhost"
      ],
      "port": "8082",
      "path": [
        "api",
        "flights"
      ]
    }
  },
  "response": []
},
{
  "name": "Create a Hotel",
  "id": "8149b2ae-bbda-4720-9901-00af1accd709",
  "request": {
    "method": "POST",
    "header": [
      {
        "key": "Content-Type",
        "value": "application/json",
        "type": "text"
      }
    ]
  }
}
```

```

        },
    ],
    "body": {
        "mode": "raw",
        "raw": "{\r\n    \"name\": \"Hilton Colombo\", \r\n    \"location\": \"Colombo\", \r\n    \"address\": \"2 Sir Chittampalam A Gardiner Mawatha\", \r\n    \"rating\": 5, \r\n    \"pricePerNight\": 15000.00, \r\n    \"availableRooms\": 20, \r\n    \"amenities\": \"WiFi, Pool, Gym, Spa\"\r\n},
        "options": {
            "raw": {
                "language": "json"
            }
        }
    },
    "url": {
        "raw": "http://localhost:8083/api/hotels",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "8083",
        "path": [
            "api",
            "hotels"
        ]
    }
},
"response": []
},
{
    "name": "Create a Booking (Main Flow)",
    "id": "24c944ea-c133-4358-9786-9a921bfcc343",
    "request": {
        "method": "POST",
        "header": [
            {
                "key": "Content-Type",
                "value": "application/json",
            }
        ]
    }
}

```

```
        "type": "text"
    }
],
"body": {
    "mode": "raw",
    "raw": "{\r\n    \"userId\": 1,\r\n    \"flightId\": 1,\r\n    \"hotelId\": 1,\r\n    \"travelDate\": \"2025-01-10\"\r\n}",
    "options": {
        "raw": {
            "language": "json"
        }
    }
},
"url": {
    "raw": "http://localhost:8084/api/bookings",
    "protocol": "http",
    "host": [
        "localhost"
    ],
    "port": "8084",
    "path": [
        "api",
        "bookings"
    ]
}
},
"response": []
},
{
    "name": "Process Payment",
    "id": "f2199e2c-aad2-4ed6-a0f6-8946827cd8f9",
    "request": {
        "method": "POST",
        "header": [
            {
                "key": "Content-Type",
                "value": "application/json",
                "type": "text"
            }
        ]
    }
}
```

```
        }
    ],
    "body": {
        "mode": "raw",
        "raw": "{\r\n    \"bookingId\": 1,\r\n    \"amount\": 60000.00,\r\n    \"paymentMethod\": \"CREDIT_CARD\"\r\n}",
        "options": {
            "raw": {
                "language": "json"
            }
        }
    },
    "url": {
        "raw": "http://localhost:8085/api/payments/process",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "8085",
        "path": [
            "api",
            "payments",
            "process"
        ]
    }
},
"response": []
},
{
    "name": "Verify Booking Status",
    "id": "29c510d3-a016-4c62-a874-220d1c06ba54",
    "request": {
        "method": "GET",
        "header": [],
        "url": {
            "raw": "http://localhost:8084/api/bookings/1",
            "protocol": "http",
            "host": [

```

```
        "localhost"
    ],
    "port": "8084",
    "path": [
        "api",
        "bookings",
        "1"
    ]
}
},
"response": []
},
{
"name": "Check Notifications",
"id": "dc452abe-c848-4a9f-97ff-9ab267c9c04b",
"request": {
    "method": "GET",
    "header": [],
    "url": {
        "raw": "http://localhost:8086/api/notifications/user/1",
        "protocol": "http",
        "host": [
            "localhost"
        ],
        "port": "8086",
        "path": [
            "api",
            "notifications",
            "user",
            "1"
        ]
    }
},
"response": []
}
]
```