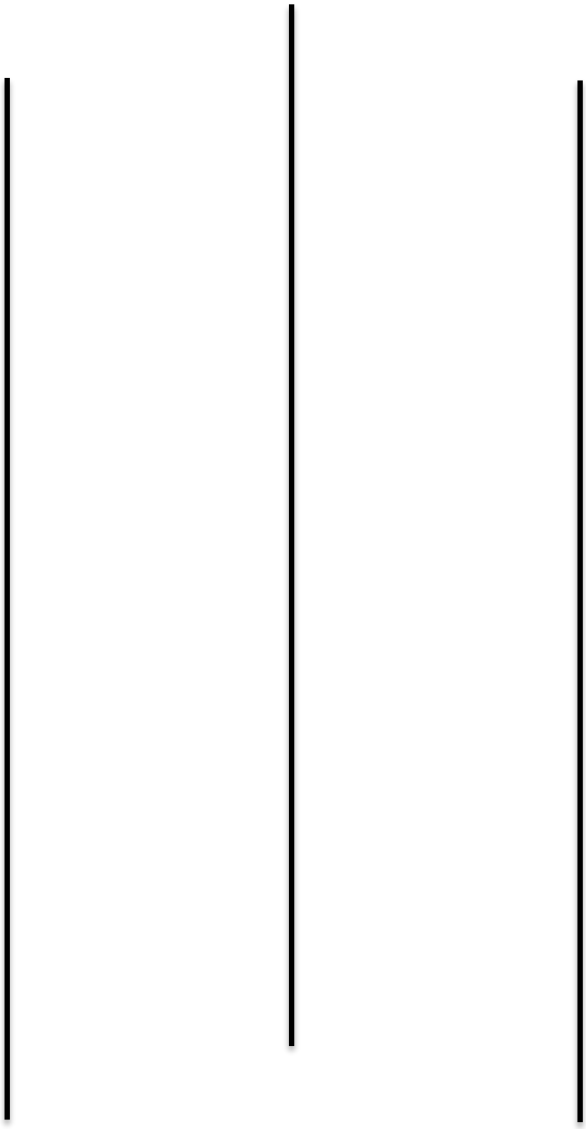
MAKALAH: APLIKASI TRANSAKSI CRUD MENGGUNAKAN REST API DALAM DOCKER DAN POSTMAN

Oleh: Tio Kukuh Ardiansyah
UNIVERSITAS Bhayangkara Surabaya



Surabaya 19 Juni 2025

1. Pendahuluan

Dalam pengembangan aplikasi modern, arsitektur REST API menjadi standar dalam membangun sistem backend yang fleksibel dan mudah diintegrasikan. Proyek ini merupakan implementasi aplikasi transaksi dengan operasi CRUD (Create, Read, Update, Delete) menggunakan PHP dan MySQL yang dibungkus dalam lingkungan Docker. Aplikasi ini mengadopsi metode HTTP seperti GET, POST, PUT, dan DELETE untuk mengakses layanan REST API.

2. Tujuan

Tujuan dari proyek ini adalah membangun layanan RESTful API sederhana yang dapat menangani transaksi menggunakan metode CRUD dan dijalankan secara terisolasi menggunakan Docker Container.

3. Teknologi yang Digunakan

- PHP 7.4
- MySQL 5.7
- Docker & Docker Compose
- Postman
- phpMyAdmin
- RESTful API

4. Struktur Proyek

Berikut adalah struktur direktori proyek:

```
transaksi_project/
├── Dockerfile
├── docker-compose.yml
├── config/
│   └── database.php
├── transaksi/
│   └── Transaksi.php
└── api/
    └── transaksi.php
```

5. Implementasi CRUD Menggunakan HTTP Method

Berikut adalah metode yang digunakan dalam REST API:

• GET: Menampilkan seluruh data transaksi.

• POST: Menambahkan data transaksi baru.

• PUT: Memperbarui data transaksi yang sudah ada berdasarkan ID.

• DELETE: Menghapus data transaksi berdasarkan ID.

Semua operasi ini diterapkan dalam satu endpoint `transaksi.php` dengan mengandalkan nilai dari `$_SERVER['REQUEST_METHOD']`.

6. Konfigurasi Database

File `database.php` berada dalam folder config dan digunakan untuk mengatur koneksi ke database MySQL di dalam container Docker. Parameter koneksi sesuai dengan environment pada docker-compose, seperti host = 'db', username = 'root', dan password = 'root'.

7. Pengujian API dengan Postman

Contoh request JSON yang digunakan dalam Postman:

• POST (Tambah):

```
{
  "nama": "Beli Kertas",
  "jumlah": 15000,
  "tanggal": "2025-06-17"
}
```

PUT (Ubah):

```
{
  "id": 1,
  "nama": "Beli Pulpen",
  "jumlah": 10000,
  "tanggal": "2025-06-18"
}
```

- GET (Melihat):

```
{
  “action”: “read”
}
```

• DELETE (Hapus):

```
{
  "id": 1
}
```

8.Source Code

\API

transaksi.php

```
<?php
include_once '../config/database.php';
include_once '../transaksi/Transaksi.php';

$db = (new Database())->getConnection();
$transaksi = new Transaksi($db);

// Mendapatkan method request
$method = $_SERVER['REQUEST_METHOD'];

// Ambil data JSON jika tersedia
$input = json_decode(file_get_contents("php://input"), true);

switch ($method) {
    case 'GET':
        $stmt = $transaksi->read();
        $data = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($data, JSON_PRETTY_PRINT);
        break;

    case 'POST':
        $transaksi->nama = $input['nama'] ?? '';
        $transaksi->jumlah = $input['jumlah'] ?? 0;
        $transaksi->tanggal = $input['tanggal'] ?? '';

        if ($transaksi->create()) {
            echo json_encode(["message" => "Transaksi berhasil ditambahkan"]);
        } else {
            echo json_encode(["message" => "Gagal menambahkan transaksi"]);
        }
        break;
```

```
    case 'PUT':
        $transaksi->id = $input['id'] ?? 0;
        $transaksi->nama = $input['nama'] ?? '';
        $transaksi->jumlah = $input['jumlah'] ?? 0;
        $transaksi->tanggal = $input['tanggal'] ?? '';

        if ($transaksi->update()) {
            echo json_encode(["message" => "Transaksi berhasil diperbarui"]);
        } else {
            echo json_encode(["message" => "Gagal memperbarui transaksi"]);
        }
        break;

    case 'DELETE':
        $transaksi->id = $input['id'] ?? 0;

        if ($transaksi->delete()) {
            echo json_encode(["message" => "Transaksi berhasil dihapus"]);
        } else {
            echo json_encode(["message" => "Gagal menghapus transaksi"]);
        }
        break;

    default:
        echo json_encode(["message" => "Metode tidak dikenali"]);
        break;
}
?>
```

\config

Database.php:

```
<?php
class Database {
    private $host = "db";
    private $db_name = "php_rest_api";
    private $username = "root";
    private $password = "root";
    public $conn;

    public function getConnection() {
        $this->conn = null;
        try {
```

```
            $this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name,
                                  $this->username, $this->password);
            $this->conn->exec("set names utf8");
        } catch (PDOException $exception) {
            echo "Connection error: " . $exception->getMessage();
        }
        return $this->conn;
    }
}
?>
```

\transaksi

Transaksi.php

```
<?php
class Transaksi {
    private $conn;
    private $table_name = "transaksi";

    public $id;
    public $nama;
    public $jumlah;
    public $tanggal;

    public function __construct($db) {
        $this->conn = $db;
    }

    public function create() {
        $query = "INSERT INTO " . $this->table_name . " (nama, jumlah, tanggal) VALUES (:nama, :jumlah, :tanggal)";
        $stmt = $this->conn->prepare($query);

        $stmt->bindParam(":nama", $this->nama);
        $stmt->bindParam(":jumlah", $this->jumlah);
        $stmt->bindParam(":tanggal", $this->tanggal);

        return $stmt->execute();
    }
```

```
    public function read() {
        $query = "SELECT * FROM " . $this->table_name . " ORDER BY id ASC";
        $stmt = $this->conn->prepare($query);
        $stmt->execute();
        return $stmt;
    }

    public function update() {
        $query = "UPDATE " . $this->table_name . " SET nama=:nama, jumlah=:jumlah,
tanggal=:tanggal WHERE id=:id";
        $stmt = $this->conn->prepare($query);

        $stmt->bindParam(":nama", $this->nama);
        $stmt->bindParam(":jumlah", $this->jumlah);
        $stmt->bindParam(":tanggal", $this->tanggal);
        $stmt->bindParam(":id", $this->id);

        return $stmt->execute();
    }

    public function delete() {
        $query = "DELETE FROM " . $this->table_name . " WHERE id=:id";
        $stmt = $this->conn->prepare($query);

        $stmt->bindParam(":id", $this->id);
        return $stmt->execute();
    }
}
?>
```

Docker-compose.yml

```
version: '3.8'
services:
  web:
    build: .
    ports:
      - "8000:80"
    volumes:
```

```
    - .:/var/www/html/
  depends_on:
    - db

 db:
  image: mysql:5.7
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_DATABASE: php_rest_api
  ports:
    - "3306:3306"

 phpmyadmin:
  image: phpmyadmin/phpmyadmin
  restart: always
  ports:
    - "8080:80"
  environment:
    PMA_HOST: db
    MYSQL_ROOT_PASSWORD: root
  depends_on:
    - db
```

Dockerfile

```
FROM php:7.4-apache
RUN docker-php-ext-install pdo pdo_mysql
COPY . /var/www/html/
```

9. Kesimpulan

Proyek ini memberikan gambaran bagaimana membangun sistem CRUD dengan REST API yang dihosting dalam Docker secara efisien dan terisolasi. Pendekatan ini sangat relevan dalam pengembangan perangkat lunak modern karena fleksibel, modular, dan mudah di-deploy ke berbagai lingkungan.