**TUGAS PRAKTIKUM DAN TUGAS INTEROPERABILITAS**



**Dosen Pengampu :**

Sepyan Purnama Kristanto, S.Kom., M.Kom.

**Disusun Oleh** :

NAMA : Moh. Naufal

NIM : 362358302146

KELAS : 2A

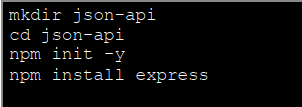
**JURUSAN BISNIS DAN INFORMATIKA**

**PRODI TEKNOLOGI REKAYASA PERANGKAT LUNAK**

**POLITEKNIK NEGERI BANYUWANGI 2024**

**Praktikum 1:**

1. Instalasi Node.js dan Express.js



1. Membuat Server Express.js

const express = require('express');

const app = express();

const port = 3000;

app.use(express.json());

let persons = [

    {

        'id': 1,

        'nama': 'John Doe',

        'umur': 30,

        'alamat': {

            'jalan': 'Jalan ABC',

            'kota': 'Jakarta'   },

            'hobi': ['Membaca', 'Bersepeda']

        }

];

app.get('/person', (req, res) => {

    res.json(persons);

});

app.post('/person', (req, res) => {

    const newPerson = req.body;

    newPerson.id = persons.length + 1;

    persons.push(newPerson);

    res.status(201).json(newPerson);

});

app.delete('/person/:id', (req, res) => {

    const id = parseInt(req.params.id);

    persons = persons.filter(person => person.id !== id);

    res.status(204).send();

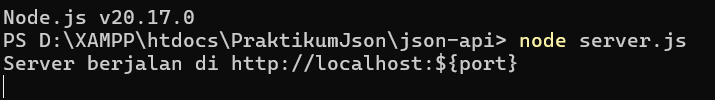
});

app.listen(port, () => {

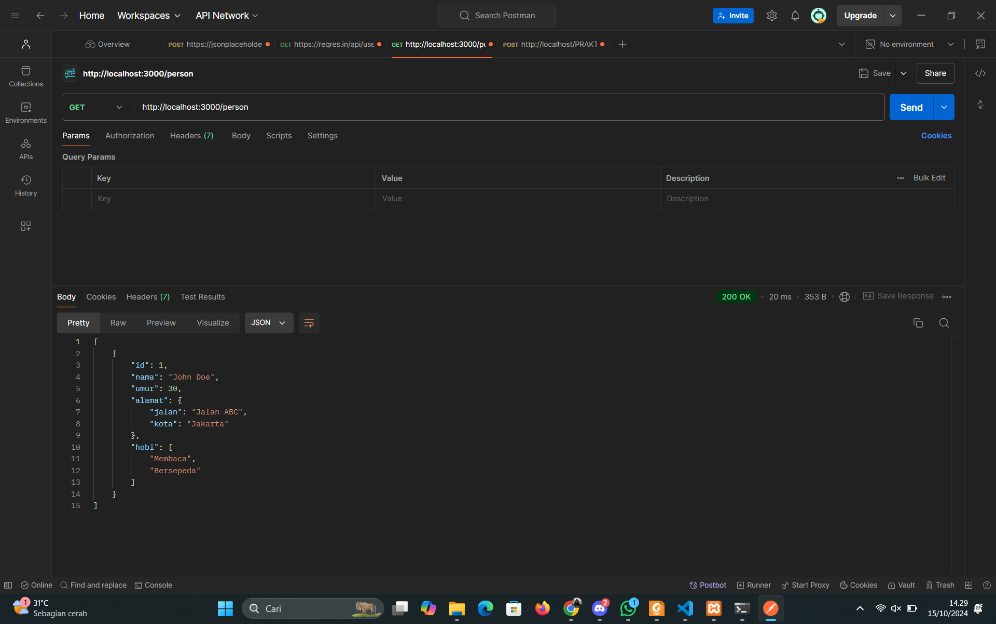
    console.log('Server berjalan di http://localhost:${port}');

});

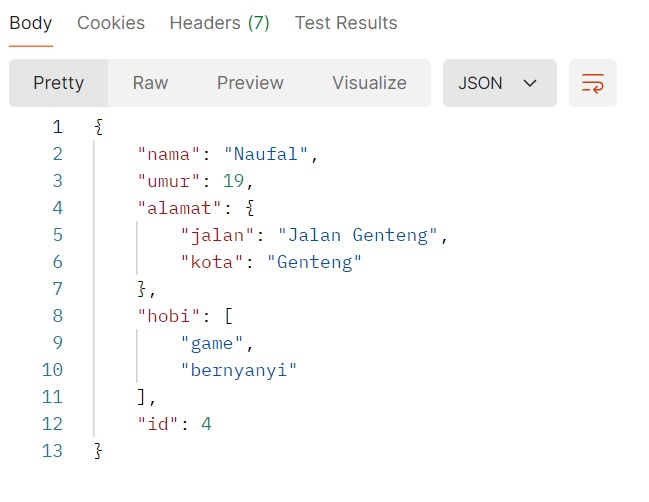
1. Menjalankan Server

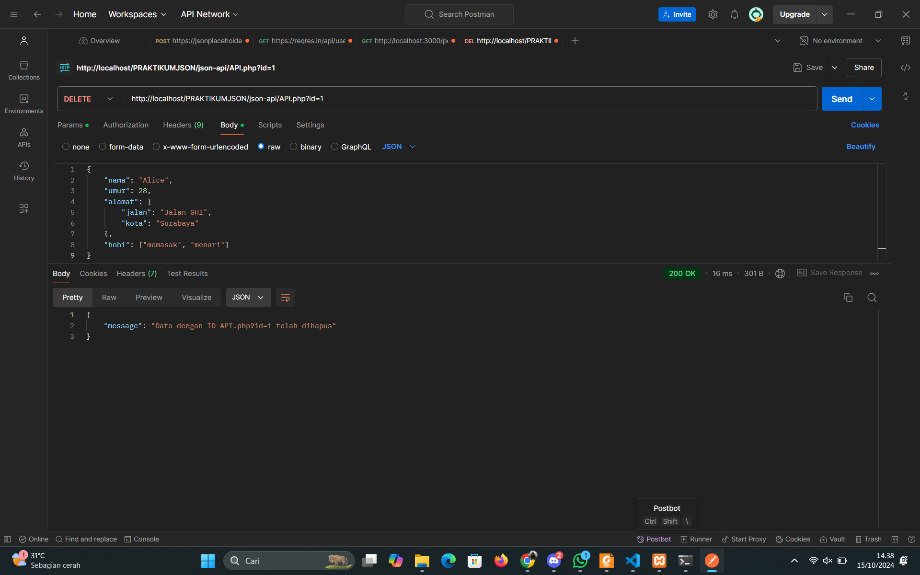


1. Mengakses API

Menggunakan postman

GET :

 Post :

 Delete:

**Praktikum 2:**

1. Instalasi PHP dan server Web
2. Membuat file API

<?php header('Content-Type: application/json');

// Data dummy

$persons = [

[

"id" => 1,

"nama" => "John Doe",

"umur" => 30,

"alamat" => [

"jalan" => "Jalan ABC",

"kota" => "Jakarta"

],

"hobi" => ["membaca", "bersepeda"]

],

[

"id" => 2,

"nama" => "Jane Doe",

"umur" => 25,

"alamat" => ["jalan" => "Jalan DEF",

"kota" => "Bandung"

],

"hobi" => ["menulis", "berenang"]

]

];

// Mendapatkan metode HTTP yang digunakan (GET, POST, DELETE)

$method = $\_SERVER['REQUEST\_METHOD'];

// Mengatur respon berdasarkan metode HTTP

switch ($method) {

case 'GET':

// Mengembalikan semua data persons

echo json\_encode($persons);

break;

case 'POST':

// Mendapatkan data dari body request

$input = json\_decode(file\_get\_contents('php://input'), true);

$input['id'] = end($persons)['id'] + 1; // Menambahkan ID baru

$persons[] = $input; // Menambahkan data baru ke array

echo json\_encode($input);

break;

case 'DELETE':

// Mendapatkan ID dari URL

$url\_parts = explode('/', $\_SERVER['REQUEST\_URI']);

$id = end($url\_parts);

// Menghapus data berdasarkan ID

$persons = array\_filter($persons, function ($person) use ($id) {

return $person['id'] != $id;

});

echo json\_encode(["message" => "Data dengan ID $id telah dihapus"]);

break;

default:

// Metode HTTP tidak didukung

http\_response\_code(405);

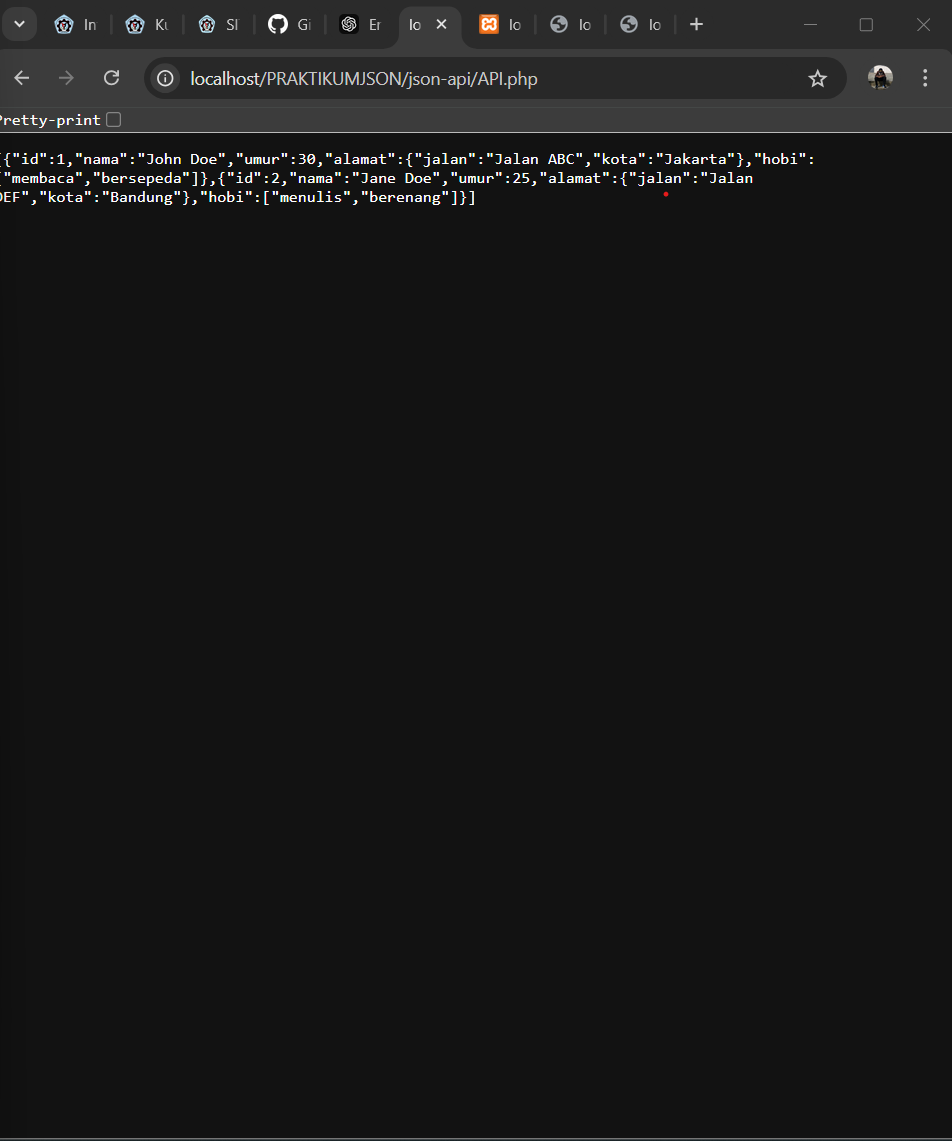
echo json\_encode(["message" => "Metode HTTP tidak didukung"]);

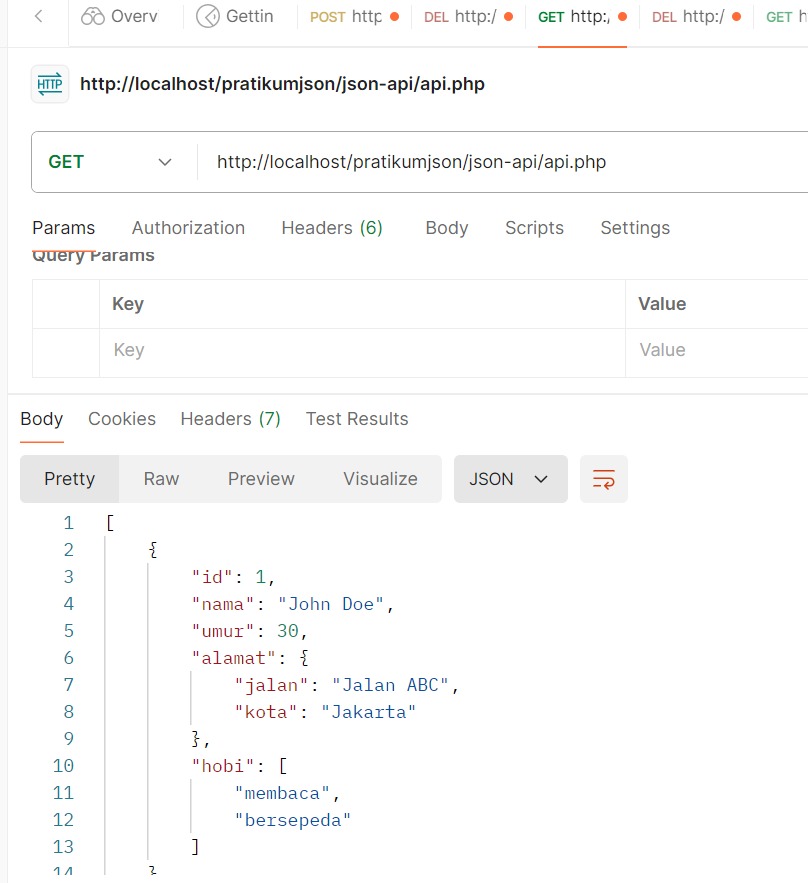
break;

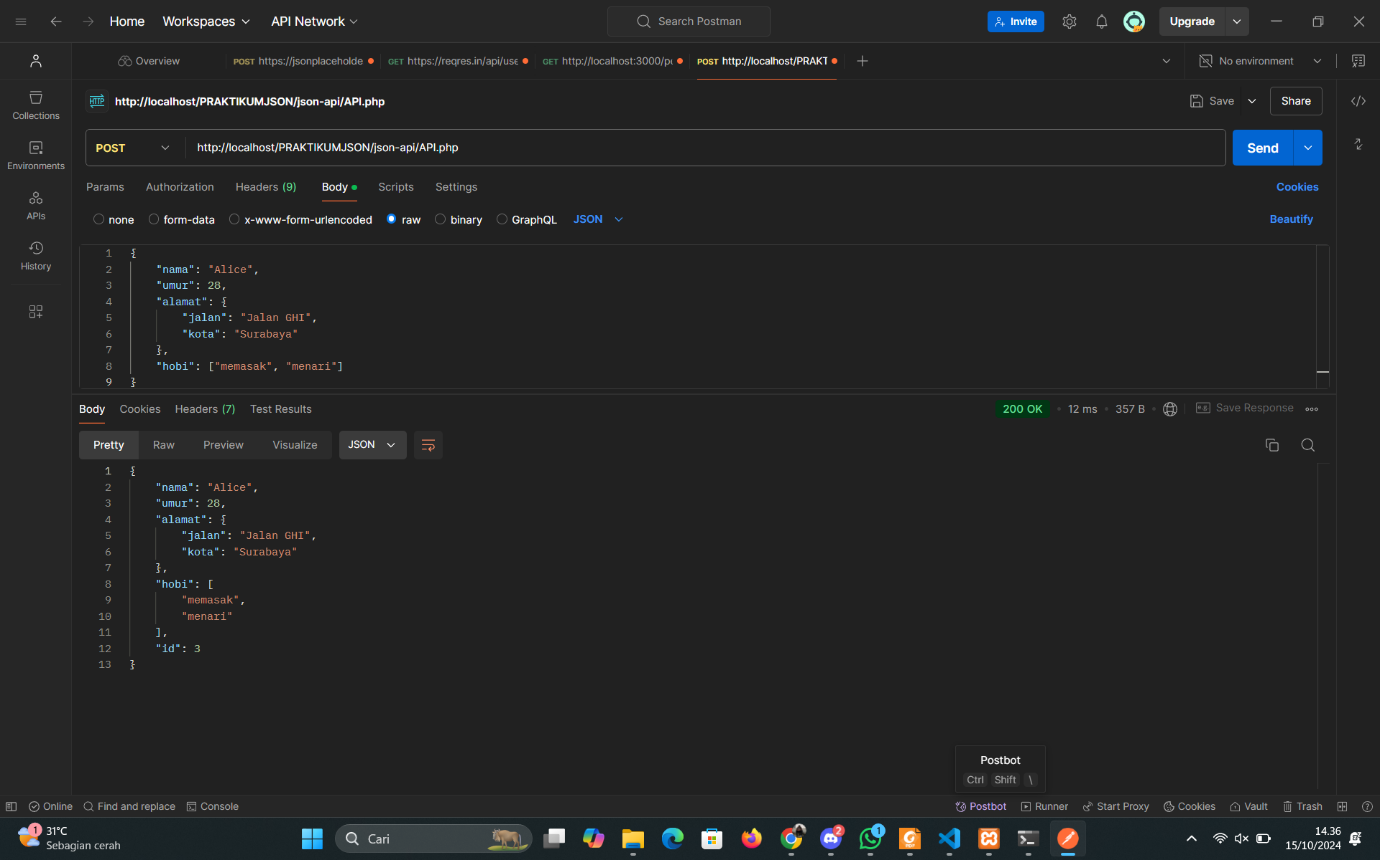
}

?>

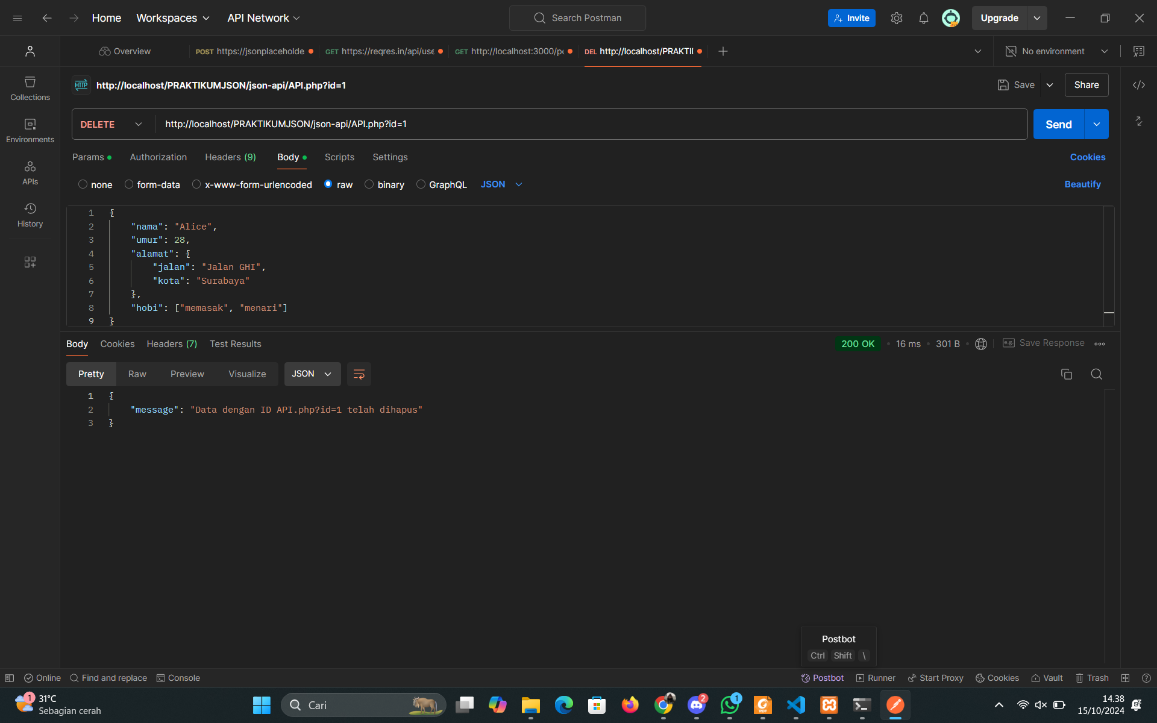
1. Menjalankan Server



1. Mengakses API :
2. Post:



1. Delete :



**Praktikum 3 :**

1. Membuat Endpoint XML

<?php header('Content-Type: application/xml');

$xml = new SimpleXMLElement('<persons/>');

$person = $xml->addChild('person');

$person->addChild('id', 1);

$person->addChild('name', 'John Doe');

$person->addChild('age', 30);

$address = $person->addChild('address');

$address->addChild('street', 'Jalan ABC');

$address->addChild('city', 'Jakarta');

$hobbies = $person->addChild('hobbies');

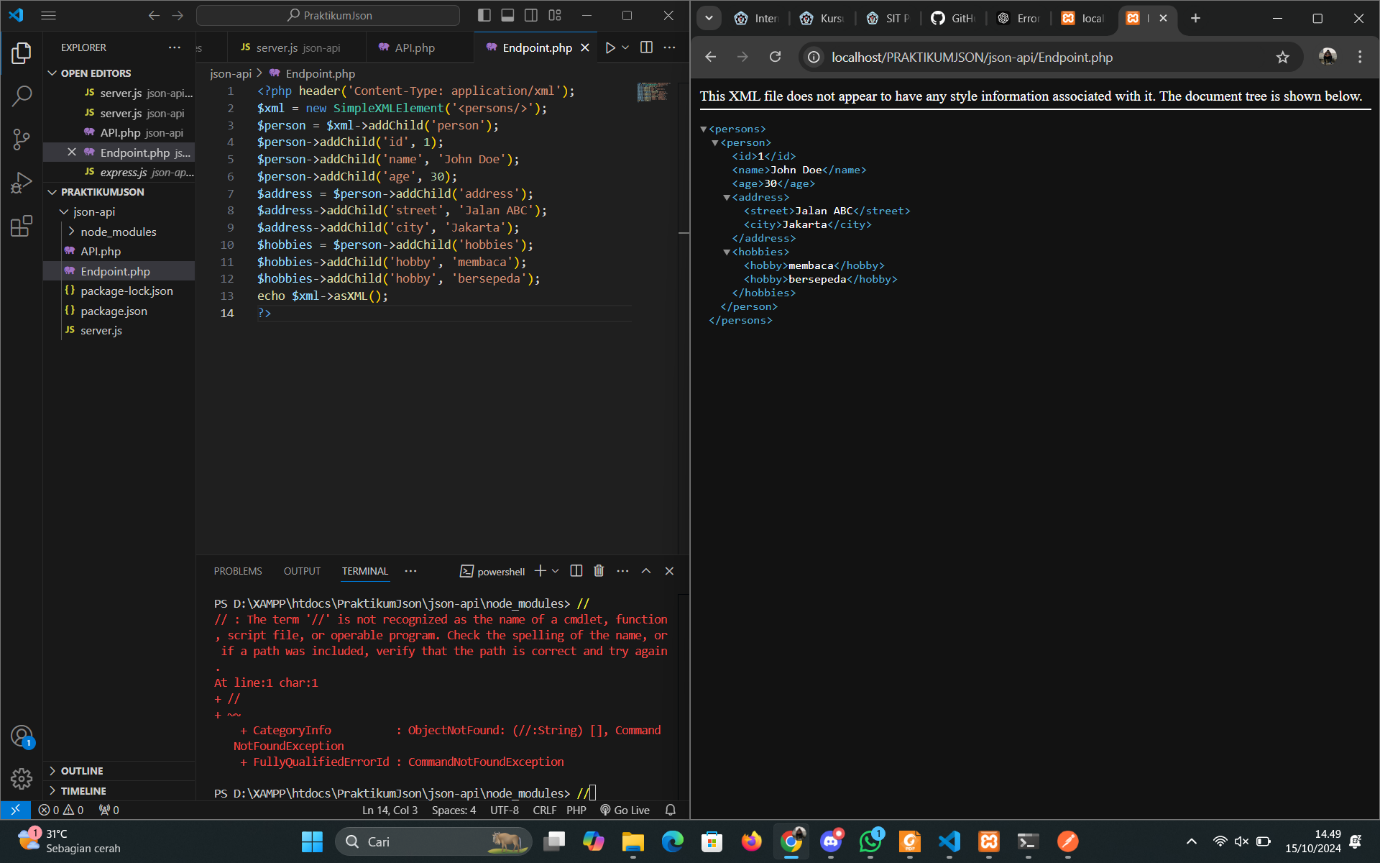
$hobbies->addChild('hobby', 'membaca');

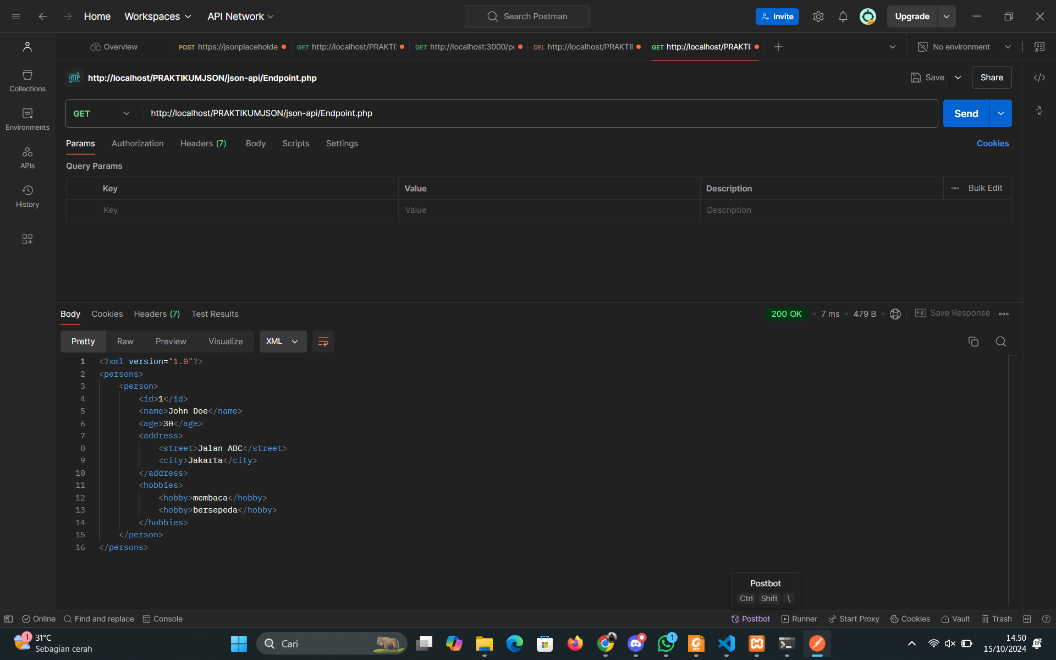
$hobbies->addChild('hobby', 'bersepeda');

echo $xml->asXML();

?>

Hasilnya :





1. Mengakses API

**TUGAS PRAKTIKUM**

1. Pengembangan API : Tambahkan endpoint baru untuk menambahkan data baru ke   
   dalam JSON dan XML API.  
   o Buat endpoint GET di API JSON dan XML yang mengembalikan daftar item   
   (misalnya, daftar buku atau daftar film).  
   o Buat endpoint POST di API JSON dan XML yang memungkinkan pengguna   
   menambahkan item baru ke daftar.

Kodenya:

const express = require('express');

const bodyParser = require('body-parser');

const xml2js = require('xml2js');

const app = express();

const port = 3001;

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: true }));

// Daftar buku sebagai contoh data

let books = [

    {

        id: 1,

        title: "Buku A",

        author: "Penulis A"

    },

    {

        id: 2,

        title: "Buku B",

        author: "Penulis B"

    },

    {

        id: 3,

        title: "Buku C",

        author: "Penulis B"

    }

];

// Endpoint GET untuk API JSON

app.get('/api/books/json', (req, res) => {

    res.json(books);

});

// Endpoint GET untuk API XML

app.get('/api/books/xml', (req, res) => {

    const builder = new xml2js.Builder();

    const xml = builder.buildObject({ books: { book: books } });

    res.set('Content-Type', 'application/xml');

    res.send(xml);

});

// Endpoint POST untuk menambahkan buku dalam format JSON

app.post('/api/books/json', (req, res) => {

    const newBook = req.body;

    newBook.id = books.length + 1; // Menentukan ID baru

    books.push(newBook);

    res.status(201).json(newBook);

});

// Endpoint POST untuk menambahkan buku dalam format XML

app.post('/api/books/xml', (req, res) => {

    const newBook = req.body;

    newBook.id = books.length + 1; // Menentukan ID baru

    books.push(newBook);

    const builder = new xml2js.Builder();

    const xml = builder.buildObject(newBook);

    res.set('Content-Type', 'application/xml');

    res.status(201).send(xml);

});

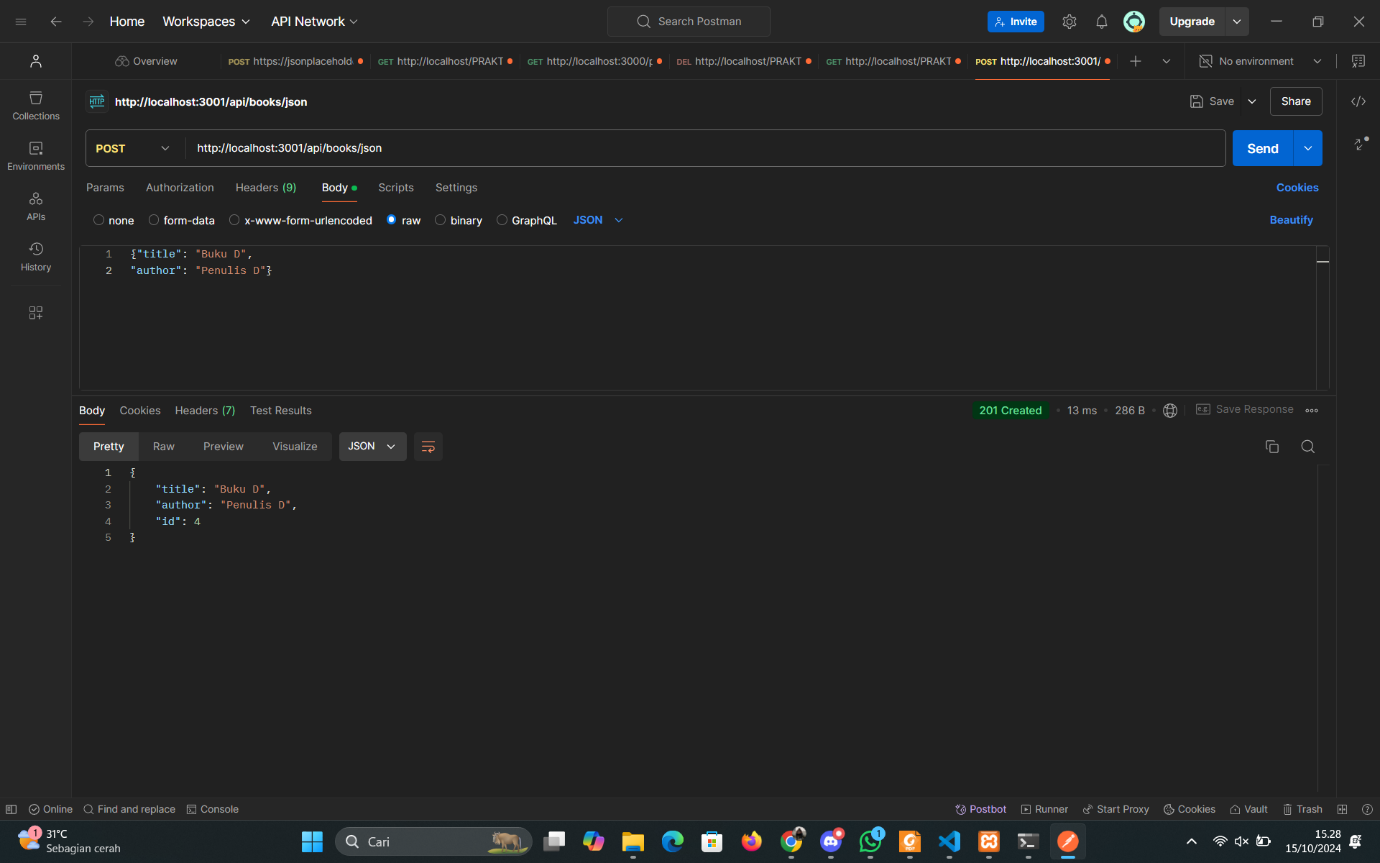
// Menjalankan server

app.listen(port, () => {

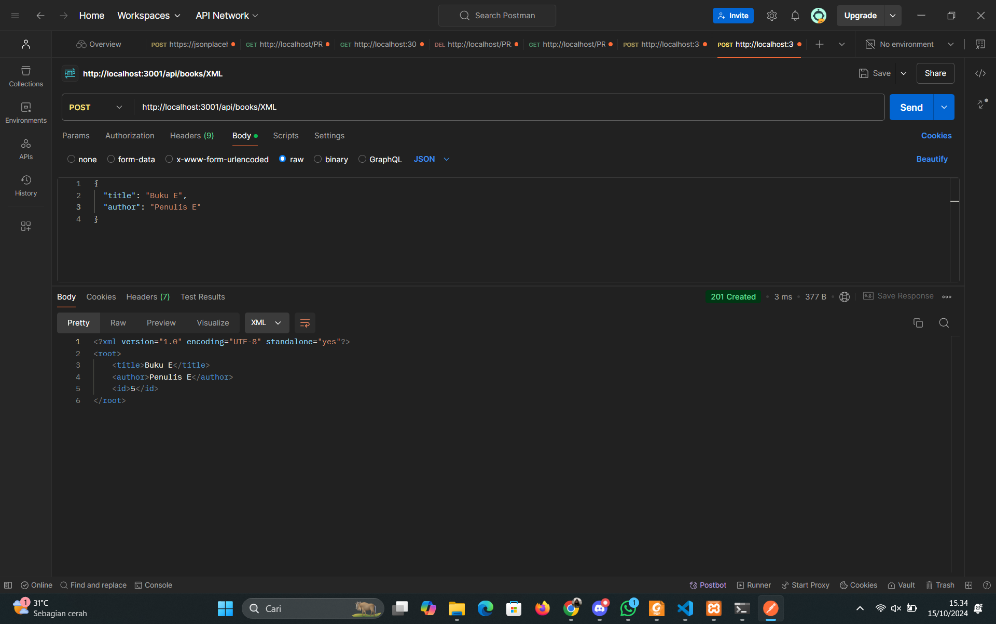
    console.log(`Server berjalan di http://localhost:${port}`);

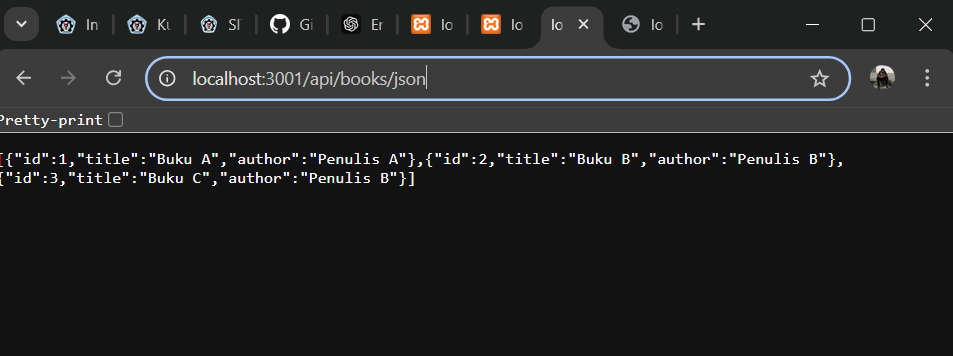
});

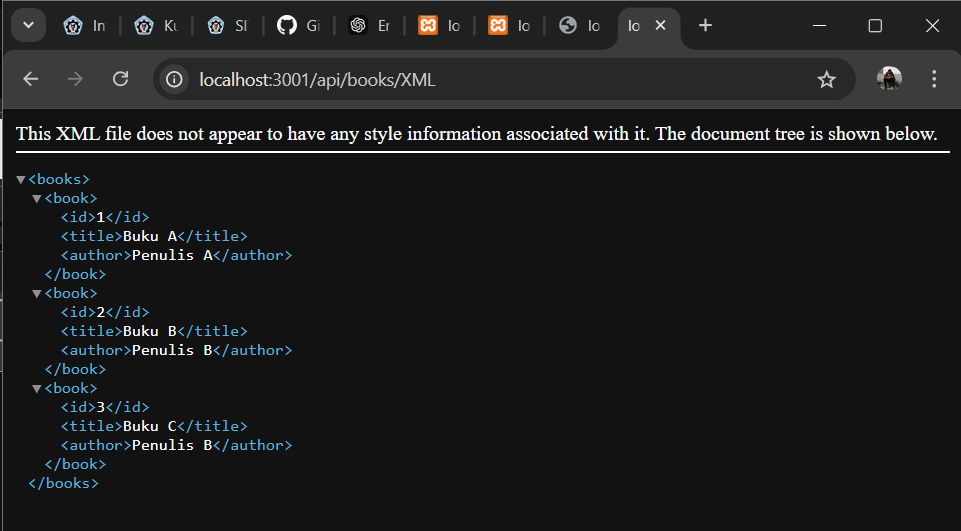
Hasilnya:



Bentuk XML:







**Kesimpulan :**

Dari praktikum tersebut berguna untuk mengenalkan bagaimana cara post get dan delete, dan memperdalam pemahaman tentang API JSON dan XML.