Department of Digital Business
Journal of Artificial Intelligence and Digital Business (RIGGS)
Homepage: https://journal.ilmudata.co.id/index.php/RIGGS
Vol. 4 No. 4 (2026) pp: 10457-10465
P-ISSN: 2963-9298, e-ISSN: 2963-914X

Implementasi Konsep Object-Oriented Programming dalam Pengembangan Rest Api Menggunakan Framework Laravel

Nur Alinuddin Kaharu[1], Agus Romadhona[2]
[1]Sistem Informasi, STMIK Adhi Guna
[2]Teknik Informatika, STMIK Adhi Guna
alinuddinkaharu@gmail.com

Abstrak

Pemrograman Berbasis Objek (Object-Oriented Programming / OOP) merupakan paradigma pemrograman yang banyak digunakan dalam pengembangan perangkat lunak modern karena kemampuannya dalam meningkatkan keterbacaan kode, kemudahan pemeliharaan, serta mendukung pengembangan sistem secara berkelanjutan. Dalam konteks sistem informasi akademik, pengelolaan data yang kompleks dan kebutuhan integrasi lintas platform menuntut adanya arsitektur perangkat lunak yang terstruktur dan fleksibel. Salah satu pendekatan yang banyak digunakan untuk memenuhi kebutuhan tersebut adalah pengembangan Representational State Transfer Application Programming Interface (REST API). Penelitian ini bertujuan untuk mengimplementasikan konsep OOP dalam pengembangan REST API menggunakan framework Laravel pada sistem informasi akademik. Metode penelitian yang digunakan adalah rekayasa perangkat lunak dengan pendekatan prototype, yang meliputi tahap analisis kebutuhan, perancangan sistem, implementasi, serta pengujian. Konsep OOP yang diterapkan meliputi enkapsulasi, abstraksi, pewarisan, dan polimorfisme. Implementasi konsep tersebut dilakukan melalui pemanfaatan komponen utama Laravel, seperti model, controller, service, dan repository pattern. Perancangan sistem dilakukan menggunakan Unified Modeling Language (UML) untuk memodelkan kebutuhan fungsional dan struktur kelas sistem, sedangkan pengujian REST API dilakukan menggunakan Postman untuk memastikan setiap endpoint berfungsi sesuai dengan spesifikasi yang telah ditetapkan. Hasil penelitian menunjukkan bahwa penerapan konsep OOP pada pengembangan REST API berbasis Laravel menghasilkan sistem yang modular, terstruktur, dan mudah dikembangkan. Selain itu, penerapan OOP juga meningkatkan tingkat maintainability serta mempermudah proses integrasi REST API dengan aplikasi klien. Dengan demikian, implementasi OOP dalam pengembangan REST API menggunakan framework Laravel terbukti mampu meningkatkan kualitas perangkat lunak dan efisiensi pengembangan sistem informasi akademik.

Kata kunci: Pemrograman Berbasis Objek, REST API, Laravel, Sistem Akademik, Rekayasa Perangkat Lunak

1. Latar Belakang

Perkembangan teknologi informasi telah mendorong peningkatan kebutuhan akan sistem informasi yang mampu menyediakan layanan data secara cepat, terintegrasi, dan mudah dikembangkan (Abdullah et al., 2025). Dalam konteks sistem akademik, pengelolaan data mahasiswa, mata kuliah, dan nilai menuntut adanya arsitektur perangkat lunak yang tidak hanya berfungsi secara teknis, tetapi juga memiliki kualitas dari sisi keteraturan struktur kode, kemudahan pemeliharaan, serta fleksibilitas terhadap perubahan kebutuhan sistem (Umar et al., 2025).

Seiring dengan meningkatnya penggunaan aplikasi berbasis web dan mobile, pendekatan arsitektur berbasis layanan (*service-oriented architecture*) menjadi semakin relevan. Salah satu bentuk implementasi arsitektur tersebut adalah REST API, yang berfungsi sebagai penghubung antara sistem backend dan berbagai aplikasi klien. REST API memungkinkan pertukaran data yang terstandarisasi serta mendukung integrasi lintas platform secara efisien (N. A. Kaharu, 2023). Namun, pengembangan REST API yang kompleks memerlukan pendekatan pemrograman yang sistematis agar perangkat lunak tetap terkelola dengan baik (N. K. Kaharu et al., 2023).

Object-Oriented Programming (OOP) merupakan paradigma pemrograman yang menekankan pada pengelompokan data dan fungsi ke dalam objek yang saling berinteraksi (Uzayr, 2022). Penerapan konsep OOP, seperti enkapsulasi, abstraksi, pewarisan, dan polimorfisme, bertujuan untuk meningkatkan kualitas perangkat lunak melalui modularitas, keterbacaan kode, dan kemudahan pengembangan (Abdurrahman et al., n.d.). Dalam

pengembangan REST API, penerapan OOP menjadi penting untuk menghindari struktur kode yang tidak terorganisasi serta sulit dipelihara seiring bertambahnya kompleksitas sistem.

Framework Laravel merupakan salah satu framework PHP yang mendukung penerapan konsep OOP secara optimal melalui arsitektur Model-View-Controller (MVC) serta berbagai fitur pendukung pengembangan REST API. Laravel menyediakan mekanisme yang memungkinkan penerapan prinsip OOP secara konsisten, seperti penggunaan model, controller, dan repository pattern (Maulani et al., 2025). Dengan dukungan tersebut, Laravel menjadi pilihan yang relevan dalam pengembangan REST API sistem akademik yang terstruktur dan berkelanjutan.

Berdasarkan uraian tersebut, penelitian ini bertujuan untuk mengkaji implementasi konsep Object-Oriented Programming dalam pengembangan REST API sistem akademik menggunakan framework Laravel. Penelitian ini diharapkan dapat memberikan gambaran mengenai bagaimana penerapan OOP berkontribusi terhadap kualitas sistem, khususnya dari aspek struktur kode, fleksibilitas pengembangan, dan kemudahan integrasi dengan aplikasi lain (Aulia et al., 2025).

2. Metode Penelitian

Penelitian ini menggunakan metode rekayasa perangkat lunak (software engineering) dengan pendekatan prototype. Pendekatan ini dipilih karena memungkinkan pengembangan sistem dilakukan secara iteratif, sehingga kebutuhan pengguna dapat dievaluasi dan disempurnakan secara bertahap (Jannah et al., 2025). Metode ini sesuai untuk pengembangan sistem informasi akademik yang membutuhkan fleksibilitas terhadap perubahan kebutuhan selama proses pengembangan.

Tahapan penelitian yang dilakukan meliputi analisis kebutuhan, perancangan sistem, implementasi, dan pengujian sistem. Setiap tahapan dilakukan secara sistematis untuk memastikan sistem REST API yang dikembangkan memenuhi kebutuhan fungsional serta menerapkan konsep Object-Oriented Programming (OOP) secara konsisten.

2.1. Analisis Kebutuhan

Tahap analisis kebutuhan bertujuan untuk mengidentifikasi kebutuhan fungsional dan nonfungsional dari sistem REST API akademik yang akan dikembangkan. Analisis dilakukan dengan mengkaji proses pengelolaan data akademik yang meliputi data mahasiswa, mata kuliah, dan nilai.

Berdasarkan hasil analisis, sistem REST API yang dikembangkan memiliki fitur utama sebagai berikut:

1. Manajemen data mahasiswa, meliputi proses penambahan, perubahan, penghapusan, dan penampilan data mahasiswa.
2. Manajemen data mata kuliah, yang mencakup pengelolaan informasi mata kuliah seperti kode mata kuliah, nama mata kuliah, dan jumlah satuan kredit semester (SKS).
3. Manajemen data nilai mahasiswa, yang meliputi pencatatan dan pengelolaan nilai akademik mahasiswa.

Kebutuhan nonfungsional sistem meliputi kemudahan integrasi dengan aplikasi klien, struktur kode yang terorganisasi, serta kemudahan pemeliharaan sistem melalui penerapan konsep OOP.

2.2. Perancangan Sistem

Tahap perancangan sistem dilakukan untuk memodelkan kebutuhan yang telah dianalisis ke dalam bentuk desain sistem yang terstruktur. Perancangan dilakukan menggunakan Unified Modeling Language (UML) sebagai alat bantu pemodelan sistem.

Diagram UML yang digunakan dalam penelitian ini meliputi:

1. Use Case Diagram, yang digunakan untuk menggambarkan interaksi antara aktor dan sistem REST API akademik serta mengidentifikasi kebutuhan fungsional sistem.
2. Class Diagram, yang digunakan untuk memodelkan struktur kelas, atribut, method, serta hubungan antar kelas berdasarkan prinsip Object-Oriented Programming.

DOI: https://doi.org/10.31004/riggs.v4i4.5165

Perancangan sistem mengacu pada konsep OOP untuk memastikan adanya pemisahan tanggung jawab (separation of concerns) pada setiap kelas, sehingga sistem yang dikembangkan bersifat modular dan mudah dikembangkan.

2.3. Implementasi

Tahap implementasi merupakan proses penerapan hasil perancangan sistem ke dalam bentuk perangkat lunak. Implementasi sistem dilakukan menggunakan framework Laravel yang mendukung penerapan konsep OOP secara optimal.

Penerapan konsep OOP dalam implementasi sistem dilakukan melalui pemanfaatan komponen utama Laravel, yaitu:

1. Model, yang berfungsi sebagai representasi data dan pengelola interaksi dengan basis data.
2. Controller, yang berfungsi untuk mengatur alur logika aplikasi dan menangani permintaan serta respons REST API.
3. Service, yang digunakan untuk menampung logika bisnis agar tidak bercampur dengan controller.
4. Repository, yang berfungsi sebagai lapisan abstraksi antara controller dan model untuk meningkatkan fleksibilitas serta kemudahan pemeliharaan sistem.

2.4. Pengujian

Tahap pengujian dilakukan untuk memastikan bahwa sistem REST API yang dikembangkan telah berjalan sesuai dengan kebutuhan fungsional yang telah ditetapkan. Pengujian dilakukan menggunakan aplikasi Postman dengan cara mengirimkan permintaan (request) ke setiap endpoint REST API.

Pengujian mencakup pengujian operasi Create, Read, Update, dan Delete (CRUD) pada setiap fitur utama sistem. Hasil pengujian menunjukkan bahwa seluruh endpoint REST API dapat berfungsi dengan baik dan menghasilkan respons yang sesuai dengan spesifikasi sistem. Dengan demikian, sistem REST API yang dikembangkan dinyatakan layak untuk digunakan dan diintegrasikan dengan aplikasi klien.

3. Hasil dan Diskusi

Implementasi Konsep OOP pada REST API Akademik

3.1 Enkapsulasi (*Encapsulation*)

Konsep enkapsulasi diterapkan dalam sistem REST API akademik dengan cara membungkus data dan perilaku ke dalam kelas yang sesuai dengan tanggung jawabnya. Pada penelitian ini, enkapsulasi diimplementasikan melalui penggunaan kelas Model pada Laravel yang merepresentasikan entitas akademik seperti mahasiswa, mata kuliah, dan nilai.

Setiap atribut data, seperti nim, nama, dan program_studi, didefinisikan sebagai bagian dari kelas Mahasiswa dan diakses melalui mekanisme yang telah disediakan oleh framework Laravel. Pendekatan ini bertujuan untuk melindungi data dari akses langsung yang tidak terkontrol serta memastikan bahwa manipulasi data dilakukan melalui mekanisme yang telah ditentukan (Romadhona et al., 2025). Dengan penerapan enkapsulasi, integritas data dapat terjaga dan risiko kesalahan pemrograman dapat diminimalkan.

Selain itu, penerapan enkapsulasi juga berkontribusi terhadap peningkatan keterbacaan dan keteraturan struktur kode (Rifa'ih & Susetyo, 2024). Dengan membatasi akses langsung terhadap atribut kelas, pengembang dapat memahami alur pengolahan data dengan lebih mudah karena setiap perubahan data harus melalui method yang telah didefinisikan. Hal ini sangat penting dalam pengembangan sistem akademik yang melibatkan banyak entitas dan proses bisnis yang saling berkaitan.

Dari sisi pemeliharaan sistem, enkapsulasi mempermudah proses pengembangan lanjutan karena perubahan pada struktur data internal suatu kelas tidak akan berdampak langsung pada komponen lain selama antarmuka kelas tersebut tetap dipertahankan. Dengan demikian, penerapan enkapsulasi mendukung prinsip maintainability dan

DOI: https://doi.org/10.31004/riggs.v4i4.5165

robustness dalam rekayasa perangkat lunak, khususnya pada pengembangan REST API berbasis Laravel yang memiliki tingkat kompleksitas data yang cukup tinggi.

Contoh Model Mahasiswa (Laravel Eloquent):

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Mahasiswa extends Model
{
    protected $table = 'mahasiswa';

    protected $fillable = [
        'nim',
        'nama',
        'program_studi',
        'angkatan'
    ];
}
```

Gambar 1. Model Mahasiswa

3.2 Abstraksi (*Abstraction*)

Abstraksi dalam sistem REST API akademik ini diterapkan dengan menggunakan interface dan repository pattern. Interface MahasiswaRepositoryInterface digunakan untuk mendefinisikan kontrak layanan pengelolaan data mahasiswa tanpa bergantung pada implementasi teknis tertentu. Pendekatan ini memungkinkan pemisahan yang jelas antara definisi layanan dan detail implementasinya.

Melalui penerapan abstraksi, pengembang dapat berfokus pada apa yang dilakukan sistem tanpa harus mengetahui secara detail bagaimana proses tersebut diimplementasikan. Dengan adanya lapisan abstraksi, perubahan pada implementasi akses data, seperti penggantian sumber data atau penambahan logika bisnis, dapat dilakukan tanpa memengaruhi komponen lain yang bergantung pada interface tersebut (Ilmiah & Grafis, 2021). Hal ini meningkatkan fleksibilitas sistem dan mendukung pengembangan perangkat lunak yang berkelanjutan (Wildan et al., 2024).

Selain meningkatkan fleksibilitas, penerapan abstraksi juga berkontribusi terhadap peningkatan kualitas arsitektur sistem. Dengan memisahkan logika bisnis dari akses data, struktur sistem menjadi lebih terorganisasi dan mudah dipahami. Pendekatan ini mempermudah proses pemeliharaan dan pengujian sistem, karena setiap lapisan dapat dikembangkan dan diuji secara terpisah tanpa menimbulkan ketergantungan yang kuat antar komponen.

Dalam konteks pengembangan REST API akademik, abstraksi berperan penting dalam mengantisipasi perubahan kebutuhan sistem di masa mendatang. Dengan arsitektur yang berbasis abstraksi, sistem dapat dikembangkan lebih lanjut untuk menangani kebutuhan akademik yang lebih kompleks tanpa memerlukan perubahan mendasar pada struktur kode yang telah ada. Dengan demikian, penerapan abstraksi mendukung prinsip scalability dan maintainability dalam pengembangan REST API berbasis Laravel.

Interface Repository Mahasiswa:

DOI: https://doi.org/10.31004/riggs.v4i4.5165

```php
<?php

namespace App\Repositories;

interface MahasiswaRepositoryInterface
{
    public function getAll();
    public function findById($id);
    public function create(array $data);
}
```

Gambar 2. Repository Mahasiswa

3.3 Pewarisan (*Inheritance*)

Konsep pewarisan diterapkan dalam pengembangan REST API akademik melalui penggunaan kelas BaseController yang berperan sebagai kelas induk bagi controller REST API lainnya. Kelas MahasiswaController mewarisi BaseController untuk memperoleh method standar dalam pengelolaan respons API, seperti format respons sukses, pesan kesalahan, dan penanganan status HTTP.

Penerapan pewarisan ini bertujuan untuk mengurangi duplikasi kode dan meningkatkan konsistensi dalam pengelolaan respons REST API. Dengan adanya kelas induk, setiap controller tidak perlu mendefinisikan ulang logika respons yang sama, sehingga struktur kode menjadi lebih ringkas dan mudah dipahami. Pendekatan ini, setiap controller tidak perlu mendefinisikan ulang logika respons yang sama, sehingga struktur kode menjadi lebih ringkas, konsisten, dan mudah dipelihara (Wisnawa et al., 2024).

Selain meningkatkan konsistensi, pewarisan juga berkontribusi terhadap kemudahan pemeliharaan sistem. Perubahan atau penambahan fitur pada mekanisme respons REST API dapat dilakukan pada kelas induk tanpa harus memodifikasi seluruh controller turunan. Hal ini menunjukkan bahwa penerapan pewarisan mendukung prinsip reusability dan maintainability dalam rekayasa perangkat lunak.

Dalam konteks pengembangan REST API berbasis Laravel, pewarisan memberikan kerangka kerja yang jelas dalam pengelolaan controller, sehingga pengembangan sistem dapat dilakukan secara lebih terstruktur dan efisien. Dengan demikian, konsep pewarisan berperan penting dalam mendukung pengembangan sistem akademik yang skalabel dan mudah dikembangkan di masa mendatang.

BaseController:

```php
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;

class BaseController extends Controller
{
    protected function successResponse($data, $message = 'Success')
    {
        return response()->json([
            'status' => true,
            'message' => $message,
            'data' => $data
        ], 200);
    }
}
```

Gambar 3. Base Controller

DOI: https://doi.org/10.31004/riggs.v4i4.5165

3.4 Polimorfisme (*Polymorphism*)

Polimorfisme dalam penelitian ini diterapkan melalui implementasi interface repository oleh kelas MahasiswaRepository. Kelas tersebut mengimplementasikan seluruh method yang didefinisikan dalam MahasiswaRepositoryInterface sesuai dengan kebutuhan sistem REST API akademik. Pendekatan ini memungkinkan penggunaan objek repository yang berbeda melalui satu antarmuka yang sama.

Dengan penerapan polimorfisme, sistem dapat menggunakan berbagai implementasi repository selama masih memenuhi kontrak interface yang telah ditentukan. Hal ini memberikan fleksibilitas dalam pengembangan sistem, khususnya ketika diperlukan perubahan atau pengembangan lanjutan pada mekanisme akses data. Implementasi repository dapat diganti, diperluas, atau dioptimalkan tanpa memerlukan perubahan pada kode controller yang bergantung pada interface tersebut (Romadhona et al., 2024).

Selain meningkatkan fleksibilitas, polimorfisme juga berperan penting dalam mendukung proses pengujian sistem. Dengan adanya interface, pengembang dapat menggunakan implementasi repository alternatif, seperti mock repository, untuk melakukan pengujian tanpa bergantung pada basis data secara langsung. Pendekatan ini meningkatkan efisiensi proses pengujian serta mendukung prinsip testability dalam rekayasa perangkat lunak.

Dalam konteks pengembangan REST API berbasis Laravel, penerapan polimorfisme membantu menjaga konsistensi arsitektur sistem serta mendukung pengembangan sistem yang skalabel. Dengan struktur yang fleksibel dan mudah dikembangkan, sistem REST API akademik dapat menyesuaikan diri terhadap perubahan kebutuhan bisnis dan teknologi di masa mendatang. Dengan demikian, polimorfisme memberikan kontribusi yang signifikan terhadap peningkatan kualitas dan keberlanjutan sistem.

Repository Mahasiswa:

```php
<?php

namespace App\Repositories;

use App\Models\Mahasiswa;

class MahasiswaRepository implements MahasiswaRepositoryInterface
{
    public function getAll()
    {
        return Mahasiswa::all();
    }

    public function findById($id)
    {
        return Mahasiswa::findOrFail($id);
    }

    public function create(array $data)
    {
        return Mahasiswa::create($data);
    }
}
```

Gambar 4. Repository Mahasiswa

3.5 Controller REST API

Integrasi konsep Object-Oriented Programming (OOP) dalam pengembangan REST API akademik ini didukung oleh arsitektur Model-View-Controller (MVC) yang disediakan oleh framework Laravel. Dalam arsitektur ini, Model berfungsi sebagai representasi data dan pengelola interaksi dengan basis data, Controller berperan dalam

DOI: https://doi.org/10.31004/riggs.v4i4.5165

mengatur alur logika aplikasi dan menangani permintaan serta respons REST API, sedangkan Repository bertindak sebagai penghubung antara controller dan model melalui mekanisme abstraksi.

Controller REST API berfungsi sebagai titik masuk utama dalam pengelolaan permintaan dari aplikasi klien. Controller menerima request, memvalidasi data, serta meneruskan proses bisnis ke lapisan service atau repository. Dengan pendekatan ini, controller tidak berisi logika bisnis yang kompleks, sehingga struktur kode tetap bersih dan mudah dipahami. Pemisahan tanggung jawab ini mencerminkan penerapan prinsip OOP yang menekankan keteraturan dan modularitas sistem.

Kombinasi penerapan enkapsulasi, abstraksi, pewarisan, dan polimorfisme menghasilkan arsitektur sistem yang modular dan terstruktur dengan baik. Setiap komponen memiliki tanggung jawab yang jelas, sehingga memudahkan proses pemeliharaan dan pengembangan sistem di masa depan. Perubahan atau penambahan fitur dapat dilakukan pada lapisan tertentu tanpa memengaruhi keseluruhan sistem.

Hasil implementasi menunjukkan bahwa penggunaan controller REST API yang terstruktur mampu meningkatkan kualitas perangkat lunak, khususnya dalam hal keterbacaan kode, fleksibilitas pengembangan, serta kemudahan integrasi REST API dengan berbagai aplikasi klien. Dengan demikian, peran controller sebagai pengelola alur logika aplikasi menjadi komponen penting dalam mendukung keberhasilan penerapan OOP pada pengembangan REST API sistem akademik berbasis Laravel.

MahasiswaController:

```php
<?php

namespace App\Http\Controllers\Api;

use App\Repositories\MahasiswaRepositoryInterface;
use Illuminate\Http\Request;

class MahasiswaController extends BaseController
{
    protected $mahasiswaRepo;

    public function __construct(MahasiswaRepositoryInterface $mahasiswaRepo)
    {
        $this->mahasiswaRepo = $mahasiswaRepo;
    }

    public function index()
    {
        $data = $this->mahasiswaRepo->getAll();
        return $this->successResponse($data);
    }

    public function store(Request $request)
    {
        $data = $this->mahasiswaRepo->create($request->all());
        return $this->successResponse($data, 'Data mahasiswa berhasil ditambahkan');
    }
}
```

Gambar 5. Controller Mahasiswa

DOI: https://doi.org/10.31004/riggs.v4i4.5165

3.6 Pembahasan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat diketahui bahwa penerapan konsep Object-Oriented Programming (OOP) pada pengembangan REST API sistem akademik menggunakan framework Laravel memberikan kontribusi yang signifikan terhadap kualitas perangkat lunak yang dihasilkan. Penerapan OOP memungkinkan sistem dikembangkan secara modular dengan pembagian tanggung jawab yang jelas pada setiap komponen, sehingga struktur kode menjadi lebih terorganisasi dan mudah dipahami.

Konsep enkapsulasi berperan penting dalam menjaga integritas data akademik dengan membatasi akses langsung terhadap atribut kelas. Data mahasiswa, mata kuliah, dan nilai dikelola melalui kelas model yang terstruktur, sehingga mengurangi potensi kesalahan dalam manipulasi data. Penerapan enkapsulasi ini sejalan dengan prinsip OOP yang menekankan keamanan data serta keteraturan dalam pengelolaan informasi akademik.

Penerapan abstraksi melalui penggunaan interface dan repository pattern memberikan fleksibilitas dalam pengembangan sistem. Dengan adanya abstraksi, perubahan pada lapisan akses data dapat dilakukan tanpa memengaruhi logika aplikasi pada controller. Hal ini menunjukkan bahwa sistem yang dikembangkan memiliki tingkat maintainability yang baik serta mampu mendukung pengembangan jangka panjang tanpa mengorbankan stabilitas sistem.

Konsep pewarisan dan polimorfisme turut mendukung efisiensi pengembangan sistem dengan mengurangi duplikasi kode dan meningkatkan konsistensi implementasi. Pewarisan pada controller memungkinkan penggunaan format respons REST API yang seragam, sedangkan polimorfisme memungkinkan penggunaan berbagai implementasi repository yang berbeda dengan tetap mempertahankan struktur sistem yang sama. Pendekatan ini mempermudah proses pengujian serta pengembangan lanjutan sistem.

Selain meningkatkan keteraturan struktur kode, penerapan OOP juga berdampak pada kemudahan pengembangan fitur baru. Dengan arsitektur yang modular, penambahan endpoint REST API maupun pengembangan layanan akademik lainnya dapat dilakukan tanpa memerlukan perubahan besar pada komponen yang telah ada. Kondisi ini menunjukkan bahwa sistem memiliki tingkat fleksibilitas yang tinggi terhadap perubahan kebutuhan pengguna.

Dari sisi pengujian, pemisahan logika bisnis ke dalam service dan repository memungkinkan proses pengujian dilakukan secara lebih terfokus pada masing-masing komponen. Hal ini memudahkan identifikasi kesalahan serta mempercepat proses perbaikan sistem. Dengan demikian, penerapan OOP berkontribusi terhadap peningkatan testability dan efisiensi pemeliharaan sistem.

Secara keseluruhan, integrasi konsep OOP dalam arsitektur Laravel menghasilkan REST API akademik yang terstruktur, mudah dipelihara, dan siap untuk diintegrasikan dengan berbagai aplikasi klien, seperti aplikasi web dan mobile. Hasil penelitian ini memperkuat temuan penelitian sebelumnya yang menyatakan bahwa penerapan OOP pada pengembangan perangkat lunak berbasis layanan mampu meningkatkan kualitas sistem dari sisi fleksibilitas, skalabilitas, dan keterbacaan kode, serta memberikan kontribusi nyata dalam mendukung pengembangan sistem informasi akademik yang berkelanjutan.

4. Kesimpulan

Berdasarkan hasil penelitian, dapat disimpulkan bahwa implementasi konsep Object-Oriented Programming dalam pengembangan REST API menggunakan framework Laravel pada sistem akademik memberikan manfaat signifikan. OOP mampu meningkatkan struktur kode, memudahkan pemeliharaan, serta mendukung pengembangan sistem secara berkelanjutan. Laravel sebagai framework modern sangat mendukung penerapan OOP melalui fitur MVC, Eloquent ORM, dan dependency injection. Oleh karena itu, penerapan OOP dalam pengembangan REST API sangat direkomendasikan untuk sistem informasi akademik.

Referensi

1. Abdullah, S., Irwanto, Budiman, M. A., Umar, B., Sadidan, I., Setiawan, H., Aldin, M., Rahutama, S., Yuhanef, A., Armin, E. U., Edra, A. P., Nastiti, T. I., Kaharu, N. A., & Mamase, S. (2025). *Teknologi Informasi dan Komunikasi*. Eureka Media Aksara.
2. Abdurrahman, U., Siregar, Y., & Siregar, R. (n.d.). IMPLEMENTASI OBJECT-ORIENTED PROGRAMMING (OOP) DALAM APLIKASI "ORDERWASH" BERBASIS DESKTOP MENGGUNAKAN VISUAL BASIC. In *Jurnal Mahasiswa Teknik Informatika)* (Vol. 9, Issue 4).

DOI: https://doi.org/10.31004/riggs.v4i4.5165

3. Aulia, U., Hernita, A., Pandu, M. J., Kurniawan, H., Mahyuni, E. T., Saleh, H., Musniati, N., Tita, F. S., Arief, M. H., Al-Reza, D. D., S., Muh. R., Pangestuti, A., & Kaharu, N. A. (2025). *Pengantar Sistem Informasi Geografis*.
4. Ilmiah, J., & Grafis, K. (2021). *Object Oriented Programming Untuk Menentukan Rasio Financial Statement Sebagai Upaya Meningkatkan Financial Perform Pada "Sae Kerupuk Bawang." 14*(2). http://journal.stekom.ac.id/index.php/pixel□page333
5. Jannah, U. M., Wildan, Zulfan, Hernita, A., Ulfiah, Setyantoro, D., Munawird, Kaharu, N. A., & Mardin, M. I. (2025). *Algoritma & Pemrograman*.
6. Kaharu, N. A. (2023). *Aplikasi Pengelolaan Data Penjualan Obat Pada Apotek Berbasis Mobile*.
7. Kaharu, N. K., Risaldi, M., & Wildan. (2023). *Implementasi Aplikasi Tambal Ban Berbasis Mobile di Kota P*.
8. Maulani, G., Kurniawan, Y. I., Munawir, Zulfan, Fauzi, W. M., Ikhsan, M., Jannah, U. M., Bachtiar, A., Kaharu, N. A., & Wildan. (2025). *Pemrograman Web*. Alifba Media.
9. Rifa'ih, R., & Susetyo, B. (2024). Analisis Rancangan Aplikasi Assessment Green Building dengan Metode Object Oriented Programming (OOP) Berbasis PHP. *Jurnal Teknik Sipil*, *27*(2), 157–168. https://doi.org/10.5614/jts.2020.27.2.6
10. Romadhona, A., Kaharu, N. A., Karim, N. A. R., & Reza, L. (2025). *Sistem Absensi Berbasis Web pada Bagian Bina Marga*. *1*.
11. Romadhona, A., Kaharu, N. A., & Silviani, J. A. (2024). *Sistem Peringatan Upload Berita pada Dinas Komunikasi dan Informatika Kota Palu*.
12. Umar, B., Kaharu, N. A., Budiman, A. A., Nugroho, C. W., Afifa, L. N., Syofian, S., Herianto, Mamase, S., Tamriesfatno, S., Fuad, M., Saleh, H., Rahayu, S. T., & Pratama, Muh. A. W. (2025). *Dasar Data Scientist*.
13. Uzayr, S. bin. (2022). Object-Oriented Programming (OOP) Concepts. In *Mastering Kotlin*. https://doi.org/10.1201/9781003311904-2
14. Wildan, Panyili, A. S., Warmayani, & Kaharu, N. A. (2024). *Penerapan Aplikasi Sistem Informasi Pengajuan Cuti Berbasis Web*.
15. Wisnawa, I. P. O., Saleh, H., Djaali, A. A., Yahya, N., Syamsiyah, Novianti, E., Wildan, I. M. D., Sulastra, Aziz, A., Umar, B., Budiman, A. A., Ashar, M. H., & Kaharu, N. A. (2024). *Pengenalan Jaringan Komputer*. www.medsan.co.id

DOI: https://doi.org/10.31004/riggs.v4i4.5165