JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 13 No. 3, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v13i3.6705

ANALISIS PENERAPAN DESIGN PATTERN SINGLETON DALAM PENGELOLAAN KONEKSI DATABASE UNTUK EFISIENSI MEMORI

Bhanu Azizi[1*], Azhar Adyatma Pratama[2], Gareth Mu'ammar Dysa[3], Carudin[4]

[1,2,3,4] Universitas Singaperbangsa Karawang; Jl. HS.Ronggo Waluyo, Puseurjaya, Telukjambe Timur, Karawang, Jawa Barat 41361; Telp. (0267) 641177

Keywords:
Design Pattern, Singleton, Database Connection, PHP, Performance Optimization.

Corespondent Email:
bhanuazizi0407@gmail.com

Abstrak. Pengelolaan koneksi database yang efisien merupakan aspek penting dalam pengembangan aplikasi berbasis PHP, terutama untuk mengurangi pemborosan sumber daya dan meningkatkan performa sistem. Penelitian ini mengkaji penerapan *design pattern Singleton* dalam pengelolaan koneksi *database* untuk menilai efektivitasnya dalam mengurangi waktu eksekusi dan penggunaan memori dibandingkan metode konvensional yang membuat koneksi baru pada setiap permintaan. Metode penelitian dilakukan dengan mengembangkan dua versi aplikasi PHP: versi konvensional dan versi menggunakan pola *Singleton*. Pengujian dilakukan dengan 1000 iterasi proses pembuatan koneksi, mengukur waktu eksekusi serta konsumsi memori menggunakan fungsi bawaan PHP. Hasil pengujian menunjukkan rata-rata waktu eksekusi Singleton sebesar 0,000022 detik, jauh lebih cepat dibandingkan metode konvensional yang mencapai 0,012842 detik. Penggunaan memori keduanya relatif serupa, yaitu sekitar 46 byte. Temuan ini menegaskan bahwa pola Singleton secara signifikan meningkatkan efisiensi waktu tanpa menambah beban memori dalam aplikasi PHP dengan akses database intensif. Oleh karena itu, penggunaan pola Singleton direkomendasikan untuk aplikasi yang memerlukan pengelolaan koneksi *database* yang efisien dan *scalable*.

Abstract. *Efficient database connection management is crucial in PHP-based application development to minimize resource wastage and enhance system performance. This study examines the implementation of the Singleton design pattern in managing database connections to evaluate its effectiveness in reducing execution time and memory usage compared to the conventional method that creates a new connection for each request. The research involved developing two PHP application versions: a conventional version and one implementing the Singleton pattern. Testing was conducted over 1000 iterations of connection creation, measuring execution time and memory consumption using built-in PHP functions. Results showed the Singleton method achieved an average execution time of 0.000022 seconds, significantly faster than the conventional method's 0.012842 seconds. Memory usage was similar for both methods, approximately 46 bytes. These findings confirm that the Singleton pattern substantially improves execution efficiency without increasing memory overhead in PHP applications with intensive database access. Therefore, the Singleton pattern is recommended for applications requiring efficient and scalable database connection management.*

1. PENDAHULUAN

Perkembangan teknologi informasi saat ini menuntut aplikasi perangkat lunak yang tidak hanya andal dan cepat, tetapi juga efisien dalam pemanfaatan sumber daya, khususnya dalam pengelolaan koneksi ke basis data *(database)*.

Pengelolaan koneksi *database* merupakan aspek krusial karena pembuatan koneksi merupakan proses yang memerlukan sumber daya yang cukup besar, terutama pada aplikasi dengan *trafict* tinggi dan transaksi data yang intensif. Tanpa pengelolaan yang tepat, pembuatan koneksi yang berulang dapat menyebabkan pemborosan memori, memperlambat respon aplikasi, dan meningkatkan risiko terjadinya *bottleneck* yang berujung pada penurunan performa sistem.

Berbagai penelitian sebelumnya telah mengidentifikasi masalah ini dan mengusulkan solusi menggunakan *design pattern*, khususnya pola *Singleton* yang termasuk dalam kategori *creational pattern*. *Singleton* menjamin bahwa sebuah kelas hanya memiliki satu *instance* selama siklus hidup aplikasi dan menyediakan akses global ke *instance* tersebut. Dalam konteks koneksi *database*, pola ini memungkinkan aplikasi untuk menggunakan satu koneksi yang sama secara bersama, sehingga mengurangi *overhead* akibat pembuatan koneksi berulang. Penerapan *Singleton* telah banyak digunakan di berbagai bahasa pemrograman termasuk PHP. Namun, karakteristik PHP yang *stateless* pada setiap *request* menuntut implementasi *Singleton* yang lebih eksplisit dan terintegrasi, seperti menggunakan *container* atau *dependency injection framework* untuk mempertahankan konsistensi koneksi selama *request* berlangsung.

Penelitian oleh [1] mengkaji penerapan metode *Singleton* dalam pengembangan *framework* PHP berbasis *Model-View-Controller (MVC)*. Hasil penelitian tersebut menunjukkan bahwa *framework* dengan penerapan *Singleton* mampu meningkatkan *throughput*, mengurangi waktu eksekusi, dan menekan penggunaan memori dibandingkan *framework* populer seperti *Yii* dan *CodeIgniter*. Hal ini mengindikasikan bahwa penggunaan pola *Singleton* dapat secara nyata meningkatkan efisiensi aplikasi PHP, khususnya dalam pengelolaan koneksi *database* yang berulang dan kritikal terhadap performa.

Meski demikian, masih terdapat kekurangan dalam kajian terdahulu, khususnya terkait pengaruh penerapan *Singleton* pada aplikasi PHP terhadap efisiensi memori dan performa koneksi *database* dalam skenario pengujian yang lebih komprehensif. Kesenjangan ini menjadikan perlunya penelitian yang secara spesifik menganalisis dampak penggunaan *Singleton pattern* pada aplikasi PHP dengan fokus pada penggunaan memori dan waktu eksekusi dalam lingkungan aplikasi yang disederhanakan namun realistis.

Oleh karena itu, penelitian ini bertujuan untuk mengisi celah tersebut dengan mengkaji secara empiris penerapan *Singleton pattern* dalam pengelolaan koneksi *database* menggunakan PHP. Penelitian ini akan membandingkan performa antara aplikasi dengan koneksi *database* biasa dan aplikasi dengan koneksi *database* yang menggunakan pola *Singleton*. Dengan demikian, diharapkan penelitian ini dapat memberikan kontribusi praktis yang berguna bagi pengembang aplikasi web dalam membangun sistem yang efisien, *scalable*, dan dapat diandalkan.

2. TINJAUAN PUSTAKA

2.1 Studi Literatur

Studi literatur merupakan metode penelitian yang memanfaatkan sumber-sumber tertulis sebelumnya untuk memperoleh informasi dan analisis terkait topik penelitian [2]. Dalam penelitian ini, studi literatur difokuskan pada konsep design pattern, khususnya pola Singleton, serta penerapannya dalam pengelolaan koneksi database pada aplikasi berbasis PHP. Referensi yang digunakan meliputi jurnal, artikel ilmiah, dan buku terkait pengembangan aplikasi web, efisiensi koneksi database, dan implementasi design pattern dalam konteks PHP. Studi ini juga mempertimbangkan hasil penelitian terdahulu yang mengkaji performa framework PHP dengan dan tanpa penggunaan Singleton untuk memberikan landasan teoritis dan praktik yang kuat bagi analisis lebih lanjut dalam penelitian ini.

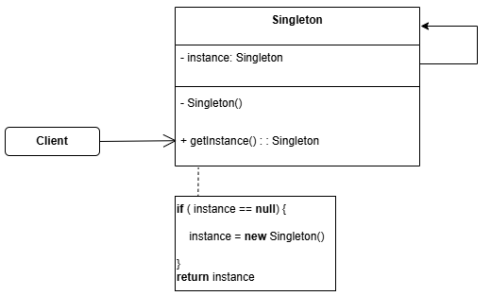
2.2 PHP

PHP adalah bahasa pemrograman yang dirancang khusus untuk pengembangan aplikasi web dinamis [3]. Awalnya, PHP merupakan singkatan dari Personal Home Page dan pertama kali dibuat oleh Rasmus Lerdorf pada

tahun 1995 sebagai sekumpulan skrip untuk mengolah data formulir web dengan nama awal *FI (Form Interpreter)*. Seiring perkembangan teknologi web, PHP berevolusi menjadi bahasa yang lebih kompleks dan kuat, dengan akronim rekursif PHP: Hypertext Preprocessor. PHP memungkinkan pengembang untuk membuat halaman web yang interaktif dan dinamis dengan mengintegrasikan logika server-side secara langsung dalam HTML. Keunggulan PHP terletak pada kemudahan penggunaannya, kompatibilitas luas dengan berbagai sistem operasi dan server web, serta dukungan besar dari komunitas pengembang. Saat ini, PHP menjadi salah satu bahasa pemrograman server-side paling populer dan banyak digunakan dalam pembuatan situs web serta aplikasi web, termasuk sistem manajemen konten, platform e-commerce, dan layanan web lainnya.

2.3 Design Pattern Singleton

Design pattern merupakan solusi desain yang telah teruji untuk masalah yang berulang dalam pengembangan perangkat lunak berorientasi objek. Salah satu pola desain yang paling sering digunakan untuk pengelolaan sumber daya adalah Singleton pattern, yang termasuk dalam kategori creational pattern. Pola ini memastikan bahwa sebuah kelas hanya memiliki satu instance selama siklus hidup aplikasi dan menyediakan akses global ke instance tersebut. Dalam konteks pengelolaan koneksi database, Singleton menghindari pembuatan koneksi berulang yang mahal secara sumber daya dan meningkatkan efisiensi penggunaan memori [4].



Gambar 1. Struktur Design Pattern Singleton

Pada struktur diagram tersebut, pola Singleton diterapkan dengan menyembunyikan konstruktor kelas menggunakan modifier akses private, sehingga instansiasi objek tidak dapat dilakukan secara langsung oleh client. Sebagai gantinya, objek diakses melalui metode statis getInstance(), yang memeriksa apakah instance sudah ada atau belum. Jika belum, maka dibuat instance baru; jika sudah ada, instance yang sama dikembalikan. Pendekatan ini menjamin bahwa hanya satu objek Singleton yang aktif selama runtime aplikasi. Selain itu, pola ini sangat berguna dalam situasi yang memerlukan koordinasi atau kontrol terpusat, seperti dalam pengelolaan log, konfigurasi global, atau koneksi basis data yang ditunjukkan dalam diagram tersebut.

2.4 Koneksi Database

Basis data *(database)* adalah kumpulan informasi yang disimpan secara terstruktur dan sistematis dalam komputer sehingga memudahkan penyimpanan, pengelolaan, dan pengambilan data. Informasi tersebut dapat diakses dan dikelola menggunakan program komputer khusus, seperti sistem manajemen basis data *(DBMS) [5]*. Salah satu *DBMS* yang paling populer dan banyak digunakan adalah *MySQL*, yang memungkinkan penyimpanan data dalam format relasional dengan kemampuan *query* yang efisien menggunakan bahasa *SQL (Structured Query Language)*. Untuk dapat berinteraksi dengan *database MySQL*, aplikasi harus terlebih dahulu melakukan koneksi melalui mekanisme yang telah disediakan, sehingga memungkinkan pengambilan, penyimpanan, dan manipulasi data secara *real-time*. Pengelolaan koneksi database yang efisien menjadi sangat penting, terutama pada aplikasi dengan volume data dan jumlah akses yang besar, agar performa sistem tetap optimal dan sumber daya tidak terbuang sia-sia. Dalam konteks pengembangan perangkat lunak, koneksi database memungkinkan aplikasi untuk mengakses dan memanipulasi data yang tersimpan dalam sistem manajemen basis data *(DBMS)*. Koneksi ini dibangun dengan menyediakan driver atau penyedia yang mendasarinya dengan string koneksi, yang merupakan cara untuk mengalamatkan database atau server tertentu serta kredensial autentikasi pengguna [6].

2.5 Pengukuran Performa Aplikasi

Pengukuran performa aplikasi web sangat penting untuk memahami efisiensi dan

kecepatan sistem, terutama dalam konteks pengelolaan koneksi database yang digunakan berulang kali. Dalam penelitian ini, pengukuran dilakukan untuk menilai dua metrik utama: waktu eksekusi dan penggunaan memori. Untuk itu, digunakan fungsi bawaan PHP seperti microtime(true) untuk mengukur waktu eksekusi dan memory_get_usage() untuk memantau penggunaan memori selama proses pembuatan koneksi database.

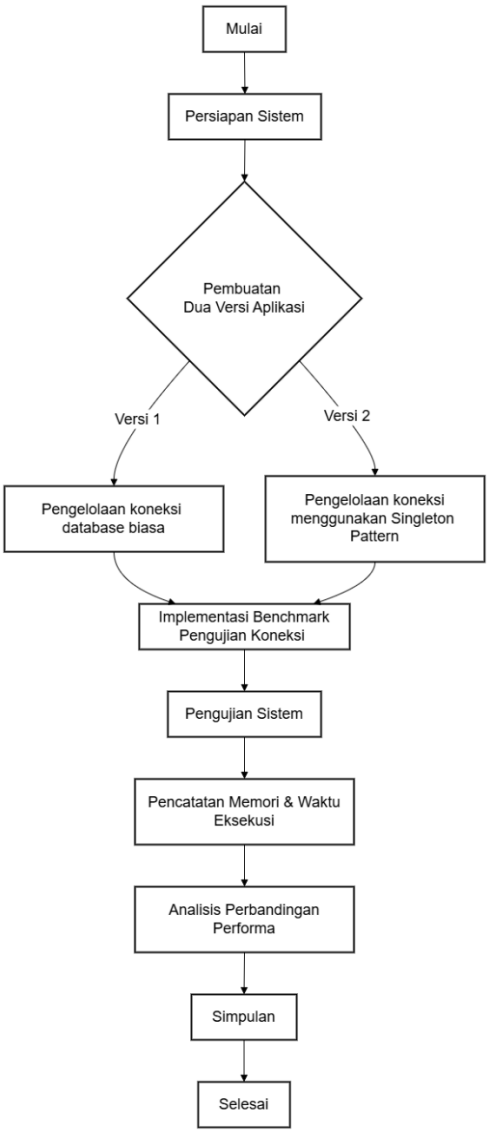
Seperti yang telah diteliti sebelumnya dalam penelitian terkait Perbandingan Performa Framework Laravel, Flask API, dan PHP Native, di mana pengukuran dilakukan untuk menilai berbagai teknologi berdasarkan waktu respon dan kapasitas penanganan request [7]. Penggunaan database pada aplikasi PHP dalam penelitian ini dibandingkan antara dua pendekatan: pengelolaan koneksi menggunakan *Singleton pattern* dan konvensional. Dengan menerapkan teknik yang serupa, pengujian ini bertujuan untuk membandingkan efisiensi kedua metode dalam hal waktu eksekusi dan konsumsi memori.

Pengujian dilakukan dengan menjalankan sejumlah operasi *database* secara berulang, dan hasil pengukuran waktu serta memori digunakan untuk menganalisis perbedaan performa antara kedua metode tersebut. Data yang dikumpulkan selama pengujian ini akan memberikan gambaran mengenai dampak penerapan *Singleton* terhadap penggunaan sumber daya, serta membantu memahami sejauh mana efisiensi sistem dapat ditingkatkan dalam skenario yang melibatkan akses database intensif.

3. METODE PENELITIAN

3.1. Diagram Alir Penelitian

Diagram alir penelitian menggambarkan langkah-langkah sistematis dalam penelitian ini [8]. Dimulai dari penentuan tujuan penelitian untuk membandingkan efisiensi pengelolaan koneksi *database* antara metode konvensional dan Singleton pattern di PHP. Selanjutnya, dilakukan persiapan sistem berupa konfigurasi lingkungan pengembangan, *database* MySQL, dan alat bantu pengukuran kinerja. Kemudian, dilakukan pengembangan dua versi aplikasi yang berbeda dalam pengelolaan koneksi *database*. Setelah itu, sistem diuji dengan skenario pengujian yang terstruktur untuk mencatat waktu eksekusi dan penggunaan memori. Data hasil pengujian dianalisis secara kuantitatif untuk menentukan metode yang paling efisien. Penelitian diakhiri dengan analisis hasil, penarikan simpulan, serta penyusunan laporan dan rekomendasi. Penelitian diakhiri dengan analisis hasil, penarikan simpulan, serta penyusunan laporan dan rekomendasi.



Gambar 2. Diagram Alir Penelitian

Gambar 1 menjelaskan tahapan penelitian secara sistematis, yang terdiri dari langkah-langkah sebagai berikut:

3.1.1. Mulai

Tahapan awal dimulai dengan penetapan tujuan dan ruang lingkup penelitian, yaitu untuk membandingkan efisiensi pengelolaan koneksi *database* antara pendekatan biasa dan penerapan *Singleton* pattern dalam PHP.

3.1.2. Persiapan Sistem

Peneliti mempersiapkan lingkungan pengembangan, termasuk konfigurasi server PHP, basis data MySQL, dan alat bantu untuk profiling seperti Xdebug atau Webgrind. Selain itu, dirancang struktur kode untuk mendukung perbandingan dua pendekatan manajemen koneksi *database*.

3.1.3. Pembuatan Dua Versi Aplikasi

Dua versi sistem dikembangkan dengan arsitektur dan fungsionalitas serupa, namun berbeda dalam cara menangani koneksi *database*:

- Versi 1: Menggunakan koneksi *database* konvensional, yaitu objek koneksi dibuat setiap kali dibutuhkan.
- Versi 2: Menerapkan *Singleton* pattern untuk memastikan hanya satu koneksi dibuat dan digunakan bersama oleh seluruh bagian aplikasi.

3.1.4. Implementasi Benchmark Pengujian Koneksi

Pada kedua versi aplikasi, dilakukan implementasi *benchmark* yang mengukur performa pembuatan koneksi database secara berulang. *Benchmark* ini menggunakan fungsi bawaan PHP untuk mencatat waktu eksekusi dan penggunaan memori selama proses pembuatan koneksi. Data yang dikumpulkan kemudian digunakan untuk membandingkan efisiensi antara metode pengelolaan koneksi konvensional dan menggunakan pola *Singleton*. Berikut adalah penjelasan tentang proses pengujian:

- Pengukuran Waktu Eksekusi:
  Pada setiap percobaan, waktu awal dicatat menggunakan *microtime(true)* sebelum pembuatan koneksi. Setelah koneksi selesai dibuat, waktu akhir dicatat, dan selisihnya memberikan waktu eksekusi untuk percobaan tersebut.
- Pengukuran Penggunaan Memori
  Penggunaan memori diukur menggunakan fungsi *memory_get_usage()*. Fungsi ini digunakan sebelum dan setelah proses pembuatan koneksi untuk mencatat jumlah memori yang digunakan oleh PHP selama percobaan.
- Pengulangan Pengujian:
  Pengujian dilakukan sebanyak 1000 kali untuk setiap metode, dengan data yang dikumpulkan untuk setiap percobaan disimpan dalam *array $results*. Setelah itu, rata-rata waktu eksekusi dan penggunaan memori dihitung untuk memberikan gambaran umum performa kedua metode.

3.1.5. Pengujian Sistem

Setelah sistem siap, dilakukan pengujian dengan menjalankan sejumlah permintaan ke *database* secara berulang dalam kedua versi sistem. Data pengujian dilakukan dengan skenario realistis yang mencerminkan interaksi pengguna terhadap aplikasi.

3.1.6. Pencatatan Memori dan Waktu Eksekusi

Selama proses pengujian, sistem mencatat penggunaan memori *(memory_get_usage)* dan waktu eksekusi *(microtime)* pada setiap versi. Tujuannya adalah untuk mengukur beban sumber daya yang digunakan oleh masing-masing pendekatan.

3.1.7. Analisis Perbandingan Performa

Data hasil pengujian kemudian dianalisis secara kuantitatif. Parameter yang dibandingkan meliputi jumlah memori yang digunakan, kecepatan eksekusi, serta jumlah koneksi yang terbentuk. Analisis ini digunakan untuk mengevaluasi efektivitas Singleton dalam konteks efisiensi sumber daya.

3.1.8. Simpulan

Berdasarkan hasil analisis, ditarik simpulan mengenai efektivitas penerapan *Singleton pattern*. Apabila ditemukan peningkatan

signifikan dalam efisiensi memori dan waktu eksekusi, maka pola ini direkomendasikan untuk digunakan dalam proyek aplikasi berbasis PHP yang berskala menengah hingga besar.

3.1.9. Selesai

Penelitian ditutup dengan dokumentasi hasil, penulisan laporan, dan penyiapan rekomendasi untuk pengembang atau studi lanjutan.

3.2. Analisis Kolaboratif

Analisis kolaboratif dilakukan dengan melibatkan diskusi intensif antara pengembang perangkat lunak dan ahli domain pengelolaan aplikasi web berbasis PHP serta pengelolaan basis data [9]. Tujuan kolaborasi ini adalah untuk mendapatkan pemahaman yang mendalam mengenai kebutuhan teknis dan bisnis dalam pengelolaan koneksi database yang efisien. Melalui proses ini, pengembang dapat merancang solusi yang tepat dengan mempertimbangkan tantangan nyata yang dihadapi dalam lingkungan produksi. Hasil kolaborasi ini menjadi dasar dalam merumuskan skenario pengujian dan pengembangan dua versi aplikasi yang akan dibandingkan dalam penelitian.

3.3. Perancangan Sistem

Langkah awal dalam membuat sebuah sistem adalah melakukan perancangan sistem yang matang dan terstruktur [10]. Perancangan sistem merupakan proses penting yang bertujuan untuk mengembangkan spesifikasi atau rancangan baru yang didasarkan pada hasil rekomendasi dari tahap analisis sistem sebelumnya.

Pada tahap perancangan sistem, dibuat dua versi aplikasi berbasis PHP yang memiliki fungsi utama serupa namun berbeda dalam cara pengelolaan koneksi ke basis data. Hal ini bertujuan untuk melakukan perbandingan secara langsung terhadap efisiensi penggunaan sumber daya antara kedua metode tersebut.

Versi 1: Sistem Koneksi Konvensional
Sistem ini menggunakan pendekatan konvensional di mana setiap kali aplikasi membutuhkan akses ke database, dibuat instance koneksi baru secara terpisah. Metode ini mudah diimplementasikan, namun berpotensi menyebabkan pemborosan memori dan peningkatan waktu eksekusi terutama pada aplikasi dengan jumlah permintaan database yang tinggi.

Versi 2: Sistem Koneksi dengan Singleton Pattern
mengimplementasikan pola Singleton untuk mengontrol pembuatan koneksi database. Dengan pola ini, hanya satu instance koneksi yang dibuat dan digunakan bersama oleh seluruh bagian aplikasi selama siklus hidup proses PHP berjalan. Pendekatan ini dirancang untuk mengurangi beban pembuatan koneksi berulang, menghemat penggunaan memori, dan mempercepat waktu eksekusi.

Perancangan kedua sistem tersebut difokuskan pada mekanisme pembuatan dan pengelolaan koneksi *database* sebagai objek utama yang akan diuji performanya. Pengujian dilakukan dengan mensimulasikan pembuatan koneksi berulang secara sistematis, sehingga hasil pengujian dapat memberikan gambaran akurat tentang dampak metode pengelolaan koneksi terhadap efisiensi penggunaan sumber daya sistem.

3.4. Implementasi

Implementasi adalah tahap krusial yang bermuara pada aktivitas nyata, aksi, atau tindakan yang dilakukan untuk menjalankan sebuah sistem sesuai dengan perancangan yang telah dibuat sebelumnya [11]. Namun, implementasi bukan sekadar melakukan aktivitas secara sembarangan, melainkan merupakan suatu kegiatan yang terencana dengan baik dan sistematis agar tujuan dari pengembangan sistem dapat tercapai secara efektif dan efisien. Pada fase ini, berbagai mekanisme, prosedur, serta sumber daya yang telah dipersiapkan dioptimalkan untuk mewujudkan fungsi-fungsi sistem yang diharapkan. Dengan kata lain, implementasi menjadi jembatan penghubung antara konsep desain dengan penggunaan nyata di lapangan, sehingga memastikan bahwa sistem dapat berjalan sesuai dengan kebutuhan pengguna dan mendukung proses bisnis secara optimal.

Implementasi dilakukan dengan mengembangkan kode program PHP untuk dua versi sistem sesuai rancangan yang telah dibuat. Pada versi konvensional, setiap permintaan menghasilkan pembuatan *instance* koneksi *database* baru secara langsung. Sedangkan pada versi *Singleton*, digunakan metode statis untuk mengakses satu *instance* koneksi yang sama selama siklus hidup aplikasi. Implementasi ini difokuskan pada mekanisme pembuatan dan pengelolaan koneksi database sebagai objek utama yang diukur performanya tanpa melibatkan operasi *CRUD*.

```
<?php
class DatabaseSingleton
{
    private static $instance = null;
    private $connection;

    private function __construct()
    {
        $host = "localhost";
        $dbname = "my_database";
        $username = "root";
        $password = "";

        try {
            $this->connection = new
PDO("mysql:host=$host;dbname=$dbname",
$username, $password);
            $this->connection-
>setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            die("Koneksi gagal: " . $e->getMessage());
        }
    }

    public static function getInstance()
    {
        if (self::$instance === null) {
            self::$instance = new DatabaseSingleton();
        }
        return self::$instance;
    }

    public function getConnection()
    {
        return $this->connection;
    }
}
?>
```

Kode di atas adalah implementasi dari pola Singleton yang saya gunakan dalam pengujian ini. *Database Singleton* memastikan hanya ada satu *instance* koneksi *database* yang digunakan oleh seluruh aplikasi. Koneksi ini akan dibagikan oleh seluruh bagian aplikasi selama siklus hidup proses PHP berjalan, menghindari *overhead* dari pembuatan koneksi berulang pada setiap permintaan.

3.5. Pengujian Sistem

Pengujian merupakan salah satu tahapan penting yang harus dilalui dalam proses pengembangan sistem untuk menghasilkan sebuah produk yang berkualitas dan handal [12]. Dengan demikian, pengujian menjadi langkah krusial yang menentukan keberhasilan akhir dari sebuah sistem sebelum diimplementasikan secara penuh dalam lingkungan nyata.

Pengujian dilakukan dengan menjalankan proses pembuatan koneksi *database* secara berulang pada kedua versi sistem. Pengukuran dilakukan menggunakan fungsi bawaan PHP, yaitu *microtime(true)* untuk menghitung waktu eksekusi dan *memory_get_usage()* untuk mencatat penggunaan memori selama proses pembuatan koneksi. Jumlah iterasi pengujian disesuaikan agar data yang diperoleh cukup representatif dan valid untuk analisis perbandingan performa.

3.6. Analisis Data

Analisis data adalah pendekatan yang menggunakan teknik komputasi dan statistik untuk mengolah serta mengevaluasi kumpulan data secara sistematis [13]. Metode ini berfokus pada penerapan analisis statistik, matematika, atau numerik guna mengekstrak informasi yang bermakna dari data mentah. Dengan menggunakan metode analisis data, berbagai pola, tren, dan hubungan antar variabel dapat diidentifikasi sehingga memudahkan pengambilan keputusan yang lebih tepat dan akurat. Selain itu, metode ini juga membantu dalam memvalidasi hipotesis dan memberikan gambaran yang lebih jelas mengenai fenomena yang sedang diteliti. Oleh karena itu, analisis data menjadi salah satu aspek penting dalam berbagai bidang, mulai dari bisnis, ilmu pengetahuan, hingga teknologi, untuk menghasilkan *insight* yang dapat digunakan sebagai dasar strategi dan kebijakan.

Data hasil pengujian dianalisis menggunakan statistik deskriptif, khususnya rata-rata waktu eksekusi dan penggunaan memori pada kedua

metode pengelolaan koneksi. Analisis bertujuan untuk mengevaluasi dampak penerapan pola *Singleton* terhadap efisiensi sumber daya, dengan fokus pada pengurangan waktu dan memori yang digunakan selama pembuatan koneksi *database*.

3.7. Kesimpulan dan Dokumentasi

Berdasarkan hasil analisis performa, penelitian ini menyimpulkan efektivitas penerapan pola *Singleton* dalam mengelola koneksi *database* pada aplikasi PHP. Seluruh proses implementasi, pengujian, dan analisis didokumentasikan secara lengkap untuk memastikan penelitian ini dapat direplikasi dan menjadi referensi bagi pengembang serta peneliti selanjutnya yang ingin mengembangkan atau menerapkan solusi pengelolaan koneksi yang efisien.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Pengujian Waktu Eksekusi dan Penggunaan Memori

4.1.1. Hasil Benchmark Singleton

Tabel 1. Hasil Benchmark Singleton - Waktu Eksekusi dan Penggunaan Memori

Percobaan	Waktu Eksekusi (detik)	Penggunaan Memori (byte)
1	0.021685	46440
2	0	0
3	0	0
4	0	0
5	0	0
…	…	…
995	0	0
996	0	0
997	0.000001	0
998	0	0
999	0	0
1000	0	0
Rata - rata	**0.000022**	**46.44**

Tabel berikut menunjukkan hasil pengujian waktu eksekusi dan penggunaan memori pada metode *Singleton*. Secara keseluruhan, pengujian menunjukkan bahwa waktu eksekusi rata-rata untuk pembuatan koneksi *database* menggunakan *Singleton* sangat rendah, dengan sebagian besar percobaan mencatatkan waktu mendekati 0 detik. Hal ini mengindikasikan efisiensi metode *Singleton* dalam mengelola koneksi *database* tanpa perlu membuat koneksi baru setiap kali ada permintaan. Penggunaan memori juga terjaga secara konsisten, dengan rata-rata penggunaan memori sebesar 46,44 byte.

4.1.2. Hasil Benchmark Konvensional

Tabel 2. Hasil Benchmark Singleton - Waktu Eksekusi dan Penggunaan Memori

Percobaan	Waktu Eksekusi (detik)	Penggunaan Memori (byte)
1	0.014036	46424
2	0.012448	0
3	0.014413	0
4	0.016602	0
5	0.012209	0
...	...	...
996	0.012091	0
997	0.017288	0
998	0.014396	0
999	0.011366	0
1000	0.014225	0
Rata - rata	**0.012842**	**46.42**

Pada metode Konvensional, setiap percobaan menghasilkan pembuatan koneksi baru setiap kali permintaan ke *database* dilakukan. Hasil pengujian menunjukkan waktu eksekusi yang lebih bervariasi, dengan waktu eksekusi rata-rata tercatat pada 0.012842 detik. Dibandingkan dengan *Singleton*, waktu eksekusi ini lebih tinggi karena proses pembuatan koneksi baru setiap kali permintaan dilakukan.

Penggunaan memori pada Konvensional juga tercatat pada angka yang serupa dengan *Singleton*, yaitu 46.42 byte, meskipun ada fluktuasi kecil pada setiap percobaan. Hal ini menunjukkan bahwa meskipun waktu eksekusi lebih tinggi, penggunaan memori tidak mengalami peningkatan signifikan dibandingkan dengan metode *Singleton*.

4.2. Analisis Perbandingan Performa

4.2.1. Waktu Eksekusi

Berdasarkan hasil pengujian waktu eksekusi, *Singleton* menunjukkan waktu eksekusi yang lebih cepat dibandingkan dengan Konvensional. Hal ini disebabkan oleh prinsip dasar pola *Singleton* yang hanya membuat satu *instance* koneksi yang digunakan bersama, sehingga mengurangi *overhead* yang terjadi akibat pembuatan koneksi berulang. Sebaliknya, Konvensional membuat koneksi baru untuk setiap permintaan, yang menambah waktu eksekusi.

4.2.2. Penggunaan Memori

Penggunaan memori pada *Singleton* lebih efisien, karena hanya ada satu *instance* koneksi yang digunakan bersama oleh aplikasi. Sementara itu, pada Konvensional, meskipun koneksi baru dibuat setiap kali, penggunaan memori tidak jauh berbeda dengan metode *Singleton*, meskipun ada sedikit fluktuasi pada beberapa percobaan. Ini menunjukkan bahwa meskipun waktu eksekusi lebih tinggi pada Konvensional, pengelolaan memori tidak terlalu terpengaruh.

5. KESIMPULAN

Berdasarkan hasil pengujian dan analisis performa, dapat disimpulkan bahwa penerapan pola *Singleton* dalam pengelolaan koneksi *database* pada aplikasi PHP memberikan peningkatan efisiensi yang signifikan terutama pada waktu eksekusi pembuatan koneksi. Penggunaan pola ini mampu mengurangi beban sumber daya yang biasanya terjadi akibat pembuatan koneksi berulang pada metode konvensional. Namun, penerapan Singleton juga memiliki keterbatasan yang perlu diperhatikan agar tidak menimbulkan hambatan pada aplikasi dengan kebutuhan koneksi yang lebih kompleks.

Adapun kesimpulan utama dari penelitian ini adalah sebagai berikut:

1. Pola *Singleton* berhasil menurunkan rata-rata waktu pembuatan koneksi database secara signifikan dibandingkan dengan metode konvensional. Hal ini mendukung peningkatan performa aplikasi secara keseluruhan.
2. Penggunaan memori pada metode *Singleton* relatif lebih stabil dan efisien, meskipun perbedaannya tidak sebesar pada waktu eksekusi.
3. Kelebihan Pola *Singleton*:
   - Mengurangi *overhead* pembuatan koneksi berulang.
   - Mempermudah pengelolaan koneksi *database* dengan menyediakan satu *instance* tunggal yang dapat diakses bersama.
4. Kekurangan Pola Singleton:
   - Kurang fleksibel untuk aplikasi dengan kebutuhan beberapa koneksi *database* berbeda atau skenario koneksi dinamis.
   - Potensi *bottleneck* jika tidak dirancang dengan baik dalam aplikasi berskala besar dan kompleks.
5. Rekomendasi Pengembangan
   - Melakukan pengujian pada skala pengguna simultan yang lebih besar dan skenario beban tinggi.
   - Mengkombinasikan pola Singleton dengan teknik *connection pooling* atau *caching* untuk optimasi lebih lanjut.
   - Pengembangan modul atau framework PHP berbasis *Singleton* yang mudah diintegrasikan dan disesuaikan dengan kebutuhan aplikasi.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada seluruh pihak yang telah berkontribusi dalam penyelesaian penelitian ini. Terima kasih yang sebesar-besarnya kami sampaikan kepada Fakultas Ilmu Komputer Universitas Singaperbangsa Karawang atas dukungan fasilitas dan kesempatan yang diberikan selama pelaksanaan penelitian ini.

Kami juga menghaturkan apresiasi kepada rekan-rekan penulis, yaitu Azhar Adyatma Pratama, Gareth Mu'ammar Dysa, dan Carudin, atas kerja sama, bantuan, dan dedikasi mereka dalam menyusun dan menyelesaikan penelitian ini. Semoga hasil penelitian ini dapat memberikan manfaat bagi pengembangan

aplikasi berbasis PHP dan pengelolaan koneksi database yang lebih efisien.

DAFTAR PUSTAKA

[1] U. Sa'Adah, J. Akhmad, and M. Hisyam, "Implementing Singleton method in design of MVC-based PHP framework," in *Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015*, Institute of Electrical and Electronics Engineers Inc., Jan. 2016, pp. 212–217. doi: 10.1109/ELECSYM.2015.7380843.

[2] R. P. Muhammad and G. El Ibrahim, "RANCANG BANGUN SISTEM PPDB ONLINE STUDI KASUS SMK MUHAMMADIYAH GAMPING MENGGUNAKAN METODE EXTREME PROGRAMMING," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, Apr. 2024, doi: 10.23960/jitet.v12i2.4001.

[3] A. Lutfi, "SISTEM INFORMASI AKADEMIK MADRASAH ALIYAH SALAFIYAH SYAFI'IYAH MENGGUNAKAN PHP DAN MYSQL," 2017.

[4] E. Gamma, H. Richard, J. Ralph, and V. John, "Design Pattern," 2009.

[5] S. ANNISA, "PEMBUATAN DAN KONEKSI DATABASE PADA DELPHI."

[6] A. Pratama *et al.*, "SISTEM APLIKASI PENCATATAN SECARA REAL TIME DAN PENCETAKAN LABEL MATERIAL MASUK BERBASIS VBA EXCEL DENGAN KONEKSI BASIS DATA (STUDI KASUS : PT. HANSUNG FIBER)," 2025.

[7] M. Nur Faizi, I. Yulia, M. Afridon, A. Arizka, J. Nathaniel Sulistyo, and P. Negeri Bengkalis, "Perbandingan Performa Framework Laravel, Flask API Python, dan PHP Native untuk Aplikasi API pada Data AIS Polbeng," 2024.

[8] N. Khesya, "MENGENAL FLOWCHART DAN PSEUDOCODE DALAM ALGORITMA DAN PEMROGRAMAN," 2021.

[9] M. B. Ismiati and L. Hermawan, "Analisis Requirement Menggunakan Teknik CARD (Collaborative Analysis Of Requirement And Design) Dalam Pembuatan Sistem Tutorial Aksara Jawa," 2015.

[10] F. Eko Nugroho, "PERANCANGAN SISTEM INFORMASI PENJUALAN ONLINE STUDI KASUS TOKOKU," *Jurnal SIMETRIS*, vol. 7, no. 2, 2016.

[11] A. M. Rosad, "IMPLEMENTASI PENDIDIKAN KARAKTER MELALUI MANAGEMEN SEKOLAH," *Tarbawi: Jurnal Keilmuan Manajemen Pendidikan*, vol. 5, no. 02, p. 173, Dec. 2019, doi: 10.32678/tarbawi.v5i02.2074.

[12] A. Ijudin and A. Saifudin, "Pengujian Black Box pada Aplikasi Berita Online dengan Menggunakan Metode Boundary Value Analysis," vol. 5, no. 1, 2020, [Online]. Available: http://openjournal.unpam.ac.id/index.php/informatika

[13] Sofwatillah, Risnita, J. Syahran, and A. Deassy, "TEHNIK ANALISIS DATA KUANTITATIF DAN KUALITATIF DALAM PENELITIAN ILMIAH," *Journal Genta Mulia*, 2024.