EXPLORE
Jurnal Sistem Informasi dan Telematika
(Telekomunikasi, Multimedia dan Informatika)

Implementasi Redis Untuk Mempercepat Pengambilan Data (Studi Kasus: Sistem Dasawisma PKK Sumsel)

Dwi Robbi Prasetyo, Fatoni*, Muhammad Nasir, Irman Effendy
[1,2,3]Jurusan Teknik Informatika, Fakultas Sains Teknologi, Universitas Bina Darma Palembang, Indonesia
[1]dwirobbin@gmail.com, [2]fatoni@binadarma.ac.id*, [3]nasir@binadarma.ac.id, [4]irman.effendy@binadarma.ac.id

Abstract-Having quick access to the latest information is essential as the need for information continues to increase. The South Sumatra Province PKK Dasawisma System is used as a place to collect data such as information about Dasawisma programs, residents, and activities. Over time the data will be very large, so a way to handle it (Big data) is needed. In data management, especially when displaying data recaps on the Dasawisma PKK dashboard page, there are often obstacles in terms of the speed of relational operations that affect system responsiveness. To overcome these obstacles, this research proposes the implementation of an in-memory database (Redis) as a solution to speed up relational operations in displaying data recaps. Redis is a memory-based database that enables fast data storage and efficient access. ADDIE, which stands for Analysis, Design, Development, Implementation, and Evaluation, is the development process used in this research. The Analysis stage is carried out by determining the time and place of research, collecting data, and determining needs. At the design stage, activities to design technical designs were carried out. Furthermore, the development stage is carried out implementing redis cache. Then, the implementation stage is tested to measure the speed comparison between before and after redis implementation. The last stage of evaluation, which is analyzing the test result data obtained, by comparing the performance of the system when before and after the implementation of redis. obtained, by comparing system performance when before and after the implementation of redis cache. redis cache implementation. The purpose of this research is to speed up process of retrieving data from the database even though the volume of data continues to growing large. This research resulted in an average response time for the request and so on only takes a short time in loading an increasingly large amount of data as long as there has been no change in the data volume. data as long as there are no data changes.

Keywords: Data, Relational operations, In-memory Database, ADDIE, Redis.

Abstrak-Memiliki akses cepat ke informasi terbaru sangat penting karena kebutuhan akan informasi yang terus meningkat. Sistem Dasawisma PKK Provinsi Sumatera Selatan digunakan sebagai tempat untuk mengumpulkan data seperti informasi tentang program Dasawisma, penduduk, dan kegiatan. Seiring waktu data akan sangat besar, Sehingga diperlukan sebuah cara untuk menanganinya (*Big data*). Dalam pengelolaan data, terutama saat menampilkan rekap data pada halaman *dashboard* Dasawisma PKK, seringkali ditemui kendala dalam hal kecepatan operasi relasional yang mempengaruhi responsifitas sistem. Untuk mengatasi kendala tersebut, penelitian ini mengusulkan implementasi *in-memory database* (Redis) sebagai solusi untuk mempercepat operasi relasional dalam menampilkan rekap data. Redis merupakan basis data berbasis memori yang memungkinkan penyimpanan data secara cepat dan akses yang efisien. ADDIE adalah singkatan dari *Analysis, Design, Development, Implementation, and Evaluation*, merupakan proses pengembangan yang digunakan dalam penelitian ini. Tahap Analisis dilakukan dengan menentukan waktu dan tempat penelitian, melakukan pengumpulan data, dan penentuan kebutuhan. Pada tahap desain, dilakukan kegiatan merancang desain teknis. Selanjutnya tahap pengembangan dilakukan pengimplementasian redis *cache*. Kemudian, tahap implementasi dilakukan pengujian untuk mengukur perbandingan kecepatan antara sebelum dan setelah implementasi redis. Tahap terakhir evaluasi, yaitu menganalisis data hasil pengujian yang diperoleh, dengan membandingkan kinerja sistem ketika sebelum dan setelah implementasi redis *cache*. Tujuan dari penelitian ini adalah untuk mempercepat proses pengambilan data dari *database* meskipun *volume* data terus bertambah besar. Penelitian ini menghasilkan rata-rata waktu respon untuk *request* ke-2 dan seterusnya hanya dibutuhkan waktu yang singkat dalam memuat jumlah data yang semakin besar selama belum ada perubahan data.

Kata Kunci: Data, Operasi relasional, *in-memory Database*, ADDIE, Redis.

1. Pendahuluan

Kemudahan akses internet ke berbagai informasi, menyebabkan sebagian besar aktivitas manusia selalu bersentuhan dengan teknologi saat ini [1]. Sistem Dasawisma PKK Provinsi Sumatera Selatan digunakan sebagai tempat untuk mengumpulkan data seperti informasi tentang program Dasawisma, penduduk, dan kegiatan. Jumlah penduduk Provinsi Sumatera Selatan berdasarkan Sensus Penduduk tahun 2022 sebesar 8,657 juta jiwa. Dasawisma PKK adalah kelompok PKK yang terdiri dari 10 hingga 20 rumah/kepala keluarga di tingkat RT. Tiga tanggung jawab utama kader Dasawisma adalah mendata warga, mengorganisir, dan menyebarluaskan informasi. Pemerintah Provinsi Sumatera Selatan akan menggunakan data yang telah diperolehnya untuk mengoptimalkan layanan masyarakat dan melakukan berbagai intervensi. Memelihara data statistik kependudukan sangat penting untuk mencegah terjadinya kesalahan, salah satunya dalam distribusi bantuan sosial [2].

Permasalahan yang terjadi pada sistem Dasawisma PKK Provinsi Sumatera Selatan adalah terjadinya kelambatan pengambilan data yang akan ditampikan pada sisi *client* dengan jumlah data yang seiring waktu terus bertambah banyak dikarenakan kompleksitas operasi relasional yang tinggi. Operasi relasional yang kompleks berisi, seperti penggabungan data dari beberapa tabel, dan penghitungan data berdasarkan kondisi tertentu menyebabkan terjadinya kendala performa yang signifikan dalam menampilkan data pada sistem *dashboard (web admin)*. *Dashboard (Web Admin)* yang juga dikenal sebagai portal admin *web* adalah sebuah sistem yang digunakan *administrator* untuk memasok dan mengelola data dari informasi berbasis *web* di suatu lembaga organisasi atau perusahaan [3]. Setiap bagian dari data memiliki ruangnya sendiri di situs *web* [4]. Sedangkan sistem, adalah kumpulan bagian yang saling terkait yang bekerja sama untuk mencapai satu tujuan [5]. Sehingga diperlukan sebuah cara baru agar dapat mempercepat proses pengambilan data dari *database*. Salah satunya adalah dengan mengimplementasikan redis.

Menurut Harsono (Dalam [6]), mendefinisikan implementasi sebagai "pertumbuhan kegiatan yang saling menyesuaikan, yang membutuhkan jaringan pelaksana dan birokrasi yang efisien, dan interaksi antara tujuan dan tindakan untuk mencapainya". Sehingga, implementasi dapat didefinisikan sebagai proses menempatkan ide, protokol, atau serangkaian langkah baru ke dalam tindakan dengan harapan bahwa orang lain akan mengadopsinya dan membuat penyesuaian yang diperlukan untuk mengubahnya menjadi tujuan yang dapat dicapai dengan jaringan pelaksana yang dapat dipercaya.

Salvatore Sanfilippo menciptakan Redis (*Remote Dictionary Server*), sebuah basis data berbasis *key-value* yang diliris pada 10 Mei 2009 [7]. Redis adalah *database open-source* yang memiliki keuntungan dapat diakses secara cepat karena penyimpanan *dataset* berbasis memori. Hal ini membuat redis sangat cocok digunakan dalam aplikasi yang membutuhkan *response-time* rendah, seperti sistem *caching* data dalam jumlah yang besar. Alasannya adalah karena kecepatan dan responsifitas yang sangat cepat. Memiliki fleksibilitas dalam menyimpan berbagai jenis data. Dan tentu saja manajemen data *cache* yang sangat fleksibel dan efisien [8].

Tindakan menyimpan sementara teks, gambar, atau materi lain di situs *web* untuk menghemat pemuatan dan lalu lintas *server* dikenal sebagai *cache*. Sederhananya, adalah teknik yang memfasilitasi tampilan halaman *web* yang lebih cepat. Karena data yang ada di dekatnya dapat disalin melalui *cache* [9]. *Caching* terdiri dari dua jenis, yaitu *caching* pada sisi *client* dan sisi *server*. *Caching* sisi *server* terjadi di *server*, sedangkan *caching* sisi *client* terjadi di komputer pengguna dan dapat dikontrol melalui pengaturan *browser*.

Pengembangan projek dilakukan menggunakan bantuan *framework* laravel yang menggunakan arsitektur *Model View Controller* (MVC). Laravel adalah *framework* yang dibuat menggunakan bahasa PHP yang dirancang untuk membuat aplikasi. Definisi *framework* adalah sebuah kerangka kerja yang memfasilitasi pembuatan dan pengembangan perangkat lunak oleh pengembang perangkat lunak [10]. *Framework* Laravel dibuat oleh Taylor Otwell, dirilis di bawah lisensi sumber terbuka pada 9 Juli 2011. Artinya, tidak perlu membayar untuk menggunakannya.

Menurut Teknovasi (Dalam [11]) Kueri gabungan akan memerlukan waktu lebih lama untuk dieksekusi apabila semakin banyak tabel yang digabungkan. Operasi-operasi ini dapat membutuhkan waktu yang lama, terutama ketika data yang harus diolah dalam jumlah besar. Sedangkan menurut Luthfi *Database* relasional menyimpan datanya di dalam *disk* (HDD/SSD). Permasalahan tersebut dapat menjadi penyebab keterbatasan performa. *Disk* memiliki keterbatasan dalam kecepatan akses data, memperlambat waktu eksekusi *query* yang kompleks dan penarikan data dari *database*. Akibatnya, responsifitas sistem menurun dan waktu pemrosesan data menjadi lebih lama.

Untuk mengatasi kendala performa ini, penggunaan *database in-memory* seperti Redis menjadi relevan. Dalam beberapa tahun terakhir, Redis telah menjadi solusi yang populer untuk mengatasi masalah performa ini. Redis adalah *database open-source* yang memanfaatkan memori sebagai media penyimpanan data, yang memungkinkannya untuk memberikan akses langsung ke memori sehingga dapat memberikan kinerja yang jauh lebih cepat dalam pengolahan data dibandingkan dengan *database* relasional berbasis *disk*. Redis juga mendukung banyak struktur data, seperti *strings, hashes, sets, lists, sorted sets* [12].

Terdapat beberapa penelitian terdahulu yang juga menjadi sumber referensi. Seperti penelitian yang dilakukan oleh Mulki Indana Zulfa, Ari Fadli, dan Arief Wisnu Wardhana, misalnya, mempelajari bagaimana menggunakan server Redis untuk solusi *caching* aplikasi dalam memori yang mempercepat akses data relasional. Yang menghasilkan bahwa redis dapat membantu

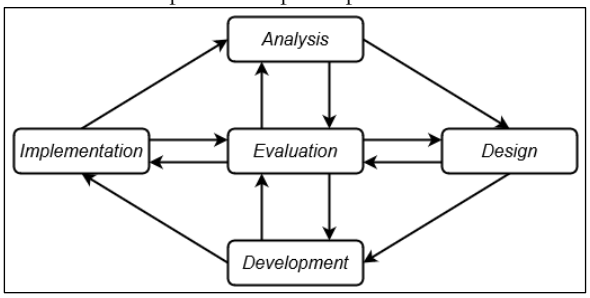
EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

meningkatkan kecepatan sistem MySQL hingga 1,7 kali, mengurangi waktu yang diperlukan untuk operasi *join* guna menampilkan data mahasiswa dan waktu yang diperlukan untuk membaca data IPK dari 61 milidetik menjadi 52 mikrodetik [13]. Penelitian selanjutnya adalah dari Faisal Al Isfahani dan Fuji Nugraha yang berjudul Implementasi *Load Balancing* NGINX dan MongoDB *Cluster* serta Mekanisme Redis *Caching* yang bertujuan untuk membangun arsitektur sistem terdistribusi dengan menggabungkan penyeimbangan beban *server web*, kluster basis data, dan teknologi *caching* yang memanfaatkan redis [14]. Yang ketiga adalah penelitian yang dilakukan oleh Anggi Putri Meriani, Dwi Ramti Asih dan Siti Annisa, berjudul Pengujian *Distributed Cached Database* Dengan Menggunakan Redis Pada Aplikasi MaBaUS. Menghasilkan bahwasanya distributed *cache* dapat mempercepat *request* terhadap data ataupun informasi yang akan di distribusikan. Hal tersebut berdasarkan angka kecepatan yang dihasilkan dari tahap pengujian yaitu terdapat kecepatan sebesar 570ms dengan tambahan 1,18 ms untuk aplikasi MaBaUS yang tidak menggunakan *cache*, dan kecepatan 560 ms untuk Aplikasi MaBaUS yang menggunakan *cache* [12].

Dengan demikian, dalam penelitian ini ingin mengimplementasikan Redis sebagai *database in-memory* dan mengujinya, yang diharapkan dapat mengatasi kendala performa yang ada dan meningkatkan kecepatan dalam mengambil atau menampilkan rekap data, sehingga dapat memberikan manfaat dalam meningkatkan responsifitas sistem, efisiensi pengambilan keputusan, dan memungkinkan pengguna untuk memperoleh informasi secara cepat dan akurat.

2. Metodologi

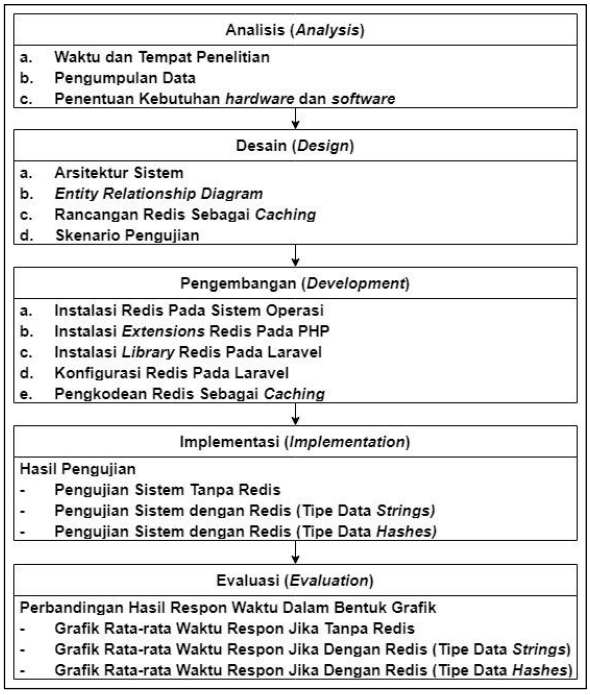
Penelitian ini akan menggunakan metode ADDIE. Berikut ini merupakan tahap-tahap metode ADDIE.



Gambar 1. Metode ADDIE

Metode ADDIE adalah model yang menggabungkan fase analisis, desain, pengembangan, implementasi, dan evaluasi pengembangan model. Dibuat pada tahun 1996 oleh Dick and Carry. Menurut Molenda (Dalam [15]) Pendekatan pembelajaran umum ini sesuai untuk penelitian pengembangan karena mengikuti proses yang berurutan namun interaktif (komunikasi dua arah atau banyak komponen komunikasi). Sedangkan menurut F. G. Constancio (Dalam [16]) Model ADDIE dapat menilai aktivitas pengembangan di setiap level.

Sebelum proses dimulai, ditentukan terlebih dahulu kerangka berfikir agar setiap tahap-tahap mempunyai tugas yang jelas. Menurut Sugiyono (Dalam [17]) kerangka berpikir adalah model konseptual yang menunjukkan bagaimana teori berhubungan dengan beberapa aspek yang telah ditentukan sebagai masalah yang signifikan.. Lalu, menurut Uma Sekaran (Dalam [18]) merupakan, "Model konseptual tentang bagaimana sebuah teori berhubungan dengan berbagai faktor yang telah diidentifikasi sebagai hal yang penting". Sehingga, ide paling mendasar atau sebagai metode untuk melaksanakan proyek penelitian secara lengkap dikenal sebagai kerangka berpikir. berikut adalah kerangka berpikir pada penelitian ini:



Gambar 2. Kerangka Berpikir

A. *Analysis* (Analisis)

Pada tahap awal ini peneliti menentukan waktu dan tempat penelitian, melakukan pengumpulan data, dan penentuan kebutuhan. Dengan menjalani tahapan analisis ini secara cermat dan komprehensif, peneliti akan memiliki landasan yang kuat untuk merencanakan, merancang, dan mengimplementasikan layanan yang efektif, efisien, dan sesuai dengan kebutuhan serta ekspektasi pengguna.

1. Waktu dan Tempat Penelitian

Periode penelitian berlangsung selama empat bulan, dari November 2022 hingga Februari 2023. Kantor Badan Penelitian dan Pengembangan Daerah Provinsi Sumatera Selatan yang berlokasi di Jalan Demang Lebar Daun No. 4864, Kota Palembang menjadi lokasi penelitian ini.

2. Pengumpulan Data

Ada dua pendekatan yang digunakan untuk mengumpulkan data: pendekatan pertama dilakukan

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

untuk memproses data sebelum fase pengembangan sistem:

a. Observasi

Observasi dilakukan langsung terhadap pengguna sistem Dasawisma PKK Provinsi Sumatera Selatan saat mereka menggunakan sistem. Terdapat kurang lebih 3.263 pengguna sistem. Adapun fokus dari data yang dikumpulkan adalah pada bagian rekap data. Misalnya pada bagian rekap data anggota keluarga, ketika jumlah data bertambah dapat mempengaruhi kinerja aplikasi. Adapun data yang ditampilkan yaitu jumlah anggota keluarga, jumlah laki-laki, jumlah perempuan, jumlah yang sudah kawin, jumlah yang belum kawin, jumlah janda, jumlah duda, jumlah yang sudah bekerja, jumlah yang belum atau tidak bekerja, dan jumlah yang pendidikan terakhirnya (SD, SLTP, SLTA, Diploma, S1, S2, S3).

b. Studi Literatur

Selain itu, data juga dikumpulkan melalui pencarian berbagai publikasi, termasuk jurnal, media internet, dan dokumentasi resmi dari para penyedia teknologi yang relevan.

3. Penentuan Kebutuhan

Berikut ini adalah spesifikasi *hardware* yang dibutuhkan:

Tabel 1 Spesifikasi *Hardware* yang dibutuhkan

Item	Deskripsi
Laptop	- *Toshiba Satellite L840* - *Memory: 8192 MB RAM* - *SSD: 256 GB* - *Processor: IntelI CoreI i5-3210M CPU @ 2.50 GHz (4 CPUs), ~ 2.5GHz*

Untuk penelitian ini, beberapa perangkat lunak juga diperlukan selain perangkat keras. Perangkat lunak ini meliputi yang berikut ini:

Tabel 2 Daftar *Software* yang dibutuhkan

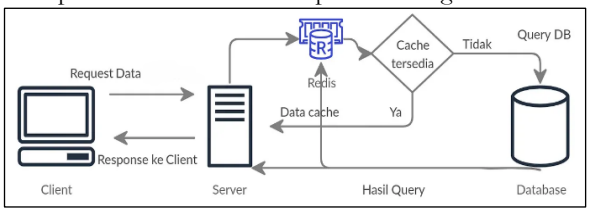
No.	*Software*	Kegunaan
1.	*Windows 10 Pro 64-bit*	Sebagai sistem operasi mode *development*
2.	*Microsoft Office 2021*	Untuk pembuatan, pengolahan, dan dokumentasi data
3.	*Google Chrome*	Untuk melakukan studi literatur dan menguji sistem
4.	*Draw.io*	Untuk membuat berbagai jenis *diagrams*, seperti ERD, *Flowchart* dan kerangka berpikir
5.	*Windows Terminal*	Untuk menjalankan *command* atau perintah
6.	*Visual Studio Code*	Untuk pengkodingan sistem
7.	*Laravel Framework*	Sebagai kerangka kerja dalam mengembangkan sistem
8.	*Laragon* - *Apache Httpd* - *MySQL* - *Redis*	Sebagai *tools* selama proses pengembangan dalam mode *development*
9.	*Laravel Debugbar*	Sebagai *tools* untuk memantau hasil *response-time*
10.	*Biznet Gio*	Sebagai tempat *hosting/vps* dan pembelian *domain*
11.	*PuTTY*	Untuk melakukan koneksi ke *terminal server* via SSH (*Secure Shell*)

A. *Design* (Desain)

Pada tahap *Design* merancang desain arsitektur sistem, merancang *caching* Redis, rancangan beberapa diagram seperti, *flowchart* dan Skenario Pengujian.

1. Arsitektur Sistem

Setelah melakukan analisa, maka dibuatlah arsitektur dari aplikasi dalam melakukan proses *caching* data.



Gambar 3. Arsitektur Sistem Dengan Redis

Gambar arsitektur sistem diatas menjelaskan bahwa *request* pertama akan melakukan perintah dari *server* ke *database* utama, tetapi saat terjadi lagi *request* yang sama, *server* akan melakukan pengecekan ke redis terlebih dahulu, jika data yang diminta tersedia di redis, maka *server* akan mengambilnya dari redis dan diteruskan ke *client*. Jika data yang diminta tidak tersedia di redis, maka *server* melakukan *querying* baru ke *database* utama (MySQL), setelah *database* mengembalikan data yang diminta, data kembalian akan disimpan ke redis agar tersedia pada *request* yang akan datang dan meneruskannya ke *client*. Sehingga *response time* yang didapat lebih cepat dan *client* tidak menunggu terlalu lama.

2. Rancangan *Caching* Redis

Dalam pengelolaan *cache*, terutama ketika menggunakan sistem *caching* seperti Redis, menetapkan konvensi nama *key* dari masing-masing *value* adalah praktik yang dapat membantu dalam pengorganisasian dan pengelolaan data dalam *cache*. Berikut ini adalah penamaan nama *key* agar datanya bisa diakses atau diambil.

Redis		
TTL	Key	Value
1h	data-recap:[area]:{fb/fn/fm/fa}:regencies:page-[number]	data
1h	data-recap:[area]:{fb/fn/fm/fa}:districts:by-regency:[regency]:page-[number]	data
1h	data-recap:[area]:{fb/fn/fm/fa}:villages:by-district:[district]:page-[number]	data
1h	data-recap:[area]:{fb/fn/fm/fa}:dasawismas:by-village:[village]:page-[number]	data
1h	data-recap:[area]:{fb/fn/fm/fa}:family-heads:by-dasawisma:[dasawisma]:page-[number]	data

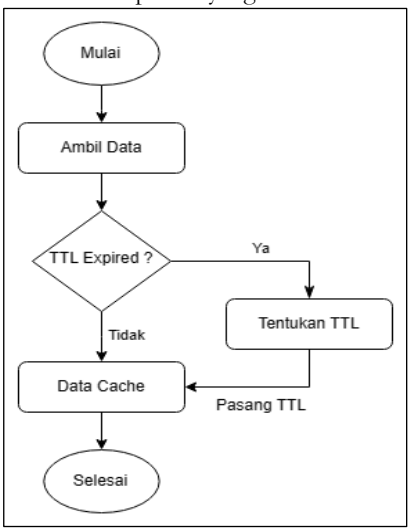
Gambar 4. Rancangan Struktur Data *Cache* di Redis

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

Informasi yang disimpan dalam redis berupa *key* dan *value*. *Key* merupakan sebuah penanda untuk data yang menyimpan hasil dari *query* sql terhadap rekap data yang ada di level kabupaten/kota, kecamatan, kelurahan/desa, dasawisma, keluarga berdasarkan area/dasawisma dan nomor halaman. Sedangkan *value* berisi data yang disimpan dalam bentuk struktur data jenis *strings* dan *hashes*. Lama waktu disimpannya adalah 1 jam.

3. *Flowchart*

FlowChart dibuat dengan tujuan untuk menjelaskan secara sederhana alur proses yang dilakukan.



Gambar 5. *Flowchart* Sistem dengan Redis

Gambar diatas menjelaskan, ketika *client* mengakses sebuah *route* rekap data, sebelum mengambil datanya dari *database* utama (MySQL), dilakukan pengecekan terlebih dahulu menggunakan *key* yang telah ditentukan pada redis apakah datanya ada dan masa TTL (*Time to Live*) masih valid. Jika bernilai benar, maka ambil data dari redis, jika tidak ada maka ambil data dari *database* MySQL dan juga simpan data balikannya pada redis. Ketika *client* selanjutnya ingin mengakses route GET itu lagi selama belum ada perubahan data dan TTL valid, maka akan mengambil datanya di redis, bukan lagi dari *database* MySQL. Lalu bagaimana jika terjadi perubahan data? misal ada penambahan data baru, pengubahan data, atau dihapus? Yang sistem akan lakukan adalah menghapus semua/spesifik data yang berkaitan pada redis dan biarkan nanti melakukan *caching* ulang dari MySQL ke Redis.

4. Skenario Pengujian

Pada tahap ini dijelaskan skenario pengujian sistem dengan jumlah data yang selalu bertambah sebanyak 200 Ribu. Masing-masing pengujian dilakukan sebanyak 3 kali *request* dari sisi *client*. Data yang disimpan pada redis menggunakan 2 jenis tipe data, yaitu *strings* dan *hashes*.

Pengujian pertama dilakukan tanpa menggunakan redis. Selanjutnya, Pengujian kedua dilakukan dengan menggunakan redis (tipe data *strings*. Terakhir, Pengujian ketiga dilakukan dengan menggunakan redis (tipe data *hashes*.

Semua pengujian dilakukan dari mulai *request* pertama sampai ketiga yang kemudian akan diteruskan ke sisi *server* dan dilanjutkan ke *database* MySQL/Redis. Data kembaliannya akan diteruskan kembali atau ditampilkan ke *client*. Tujuan pengujian ini adalah untuk mengetahui berapa lama waktu yang dibutuhkan sistem untuk menangani data yang semakin banyak.

B. *Development* (Pengembangan)

Pada tahap *Development* dilakukan proses pengembangan sistem seperti, penginstalan redis pada sistem operasi, instalasi *extensions* redis pada PHP, Instalasi *library* redis pada laravel, konfigurasi redis pada laravel, dan pengkodean redis sebagai sistem *caching* data.

3. Hasil dan Pembahasan

A. Implementasi (Hasil Pengujian)

Tahap selanjutnya, dilakukan implementasi pengujian sistem untuk memastikan bahwa redis berfungsi dengan baik dan sanggup dalam menangani data dalam jumlah besar. Pengujian dilakukan dengan jumlah data yang dimulai dari 200 Ribu sampai dengan 5 Juta dan dilakukan sebanyak tiga kali *request*. Adapun hasil yang diharapkan adalah bahwa dengan menggunakan redis sebagai sistem *caching* dapat meningkatkan kinerja sistem agar lebih cepat. Berikut ini adalah data yang akan disimpan pada redis

Tabel 3 Data Pengujian Yang Diambil dan Ditampilkan (Jumlah Data 200 Ribu)

Area	Jumlah									Jumlah Berpendidikan							
	Angg. Keluarga	Pria	Wanita	Kawin	Blm Kawin	Janda	Duda	Bekerja	Blm Bekerja	TK	SD	SMP	SMA	D3	S1	S2	S3
Ogan Kom ering Ulu	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480	10.480
Ogan Kom	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902	17.902

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

ering Ilir																		
Muara Enim	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558	13.558
Lahat	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774	21.774
Musi Rawas	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626	14.626
Musi Banyuasin	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980	12.980
Banyuasin	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852	21.852
Ogan Komering Ulu Timur	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542	15.542
Ogan Komering Ulu Selatan	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574	20.574
Ogan Ilir	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458	13.458
Empat Lawang	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338	10.338
Penukal Abab Lematang Ilir	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978	7.978
Musi Rawas Utara	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686	3.686
Palembang	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326	5.326
Prabumulih	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534	3.534
Pagar Alam	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924	1.924
Lubuk Linggau	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418	4.418

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

Data pada **Tabel 3.** Merupakan data yang akan menjadi data uji. Data tersebut diperoleh dari pengambilan beberapa kolom yang ada pada database MySQL lalu melalui proses perhitungan, yang kemudian data balikannya akan ditampilkan pada sisi *client* dan disimpan juga di redis.

1. Pengujian Sistem Tanpa Redis

Berikut ini hasil waktu respon yang dibutuhkan dari *request* pertama, kedua dan ketiga serta dilanjutkan perhitungan rata-rata respon waktunya jika tanpa redis.

Tabel 4 Hasil Pengujian Tanpa Redis

Jumlah Data	Req. ke-1 (s)	Req. ke-2 (s)	Req. ke-3 (s)	Rata-rata (s)
200 Ribu	1,13	0,825	0,735	0,90
400 Ribu	1,67	1,70	1,65	1,67
600 Ribu	7,74	4,70	3,85	5,43
800 Ribu	31,05	21,04	17,90	23,33
1 Juta	39,61	30,4	24,44	31,48
1,2 Juta	57,59	41,50	39,66	46,25
1,4 Juta	74,40	59,21	55,08	62,90
1,6 Juta	85,02	69,70	66,25	73,66
1,8 Juta	94,36	77,84	70,45	80,88
2 Juta	103,88	82,66	80,36	88,97
2,2 Juta	116,91	91,49	91,73	100,04
2,4 Juta	117,47	99,54	97,68	104,90
2,6 Juta	126,25	117,32	108,54	117,37
2,8 Juta	134,38	128,89	117,63	126,97
3 Juta	158,98	153,86	145,72	152,85
3,2 Juta	167,66	150,97	142,93	153,85
3,4 Juta	179,04	161,67	153,12	164,61
3,6 Juta	191,01	173,92	164,89	176,61
3,8 Juta	201,61	185,34	175,94	187,63
4 Juta	211,04	195,76	185,99	197,60
4,2 Juta	221,48	206,73	196,37	208,19
4,4 Juta	231,31	217,47	206,11	218,30
4,6 Juta	241,75	228,15	216,28	228,73
4,8 Juta	252,98	239,76	227,15	239,96
5 Juta	264,43	251,62	238,37	251,47

Dari tabel diatas, terlihat bahwa setiap kali ada *request*, waktu respon yang didapatkan selalu tinggi. Yang artinya *server* akan selalu mengambil datanya dari *database* utama (MySQL).

2. Pengujian Sistem Dengan Redis (Tipe Data *Strings*)

Berikut ini merupakan hasil waktu respon yang dibutuhkan dari *request* kedua, ketiga dan keempat serta dilanjutkan dengan perhitungan rata-rata respon waktunya jika menggunakan tipe data *strings*.

Tabel 5 Hasil Pengujian Redis dengan Jenis Tipe Data Strings

Jumlah Data	Req. ke-2 (s)	Req. ke-3 (s)	Req. ke-4 (s)	Rata-rata (s)
200 Ribu	0,430	0,453	0,464	0,449
400 Ribu	0,439	0,467	0,481	0,462
600 Ribu	0,485	0,519	0,459	0,488
800 Ribu	0,465	0,509	0,477	0,484
1 Juta	0,502	0,477	0,483	0,487
1,2 Juta	0,475	0,409	0,466	0,450
1,4 Juta	0,505	0,452	0,475	0,478
1,6 Juta	0,512	0,429	0,473	0,471
1,8 Juta	0,514	0,439	0,481	0,478
2 Juta	0,526	0,457	0,476	0,486
2,2 Juta	0,530	0,440	0,476	0,482
2,4 Juta	0,545	0,456	0,481	0,494
2,6 Juta	0,548	0,453	0,480	0,494
2,8 Juta	0,557	0,461	0,482	0,500
3 Juta	0,534	0,462	0,481	0,493
3,2 Juta	0,552	0,463	0,484	0,500
3,4 Juta	0,553	0,470	0,485	0,503
3,6 Juta	0,551	0,471	0,485	0,503
3,8 Juta	0,551	0,476	0,486	0,504
4 Juta	0,552	0,478	0,487	0,506
4,2 Juta	0,558	0,482	0,488	0,509
4,4 Juta	0,556	0,485	0,489	0,510
4,6 Juta	0,557	0,488	0,490	0,512
4,8 Juta	0,559	0,492	0,491	0,514
5 Juta	0,560	0,495	0,492	0,516

Pada *request* pertama tidak dimasukkan pada tabel karena data belum ada pada redis, sehingga *server* melakukan perintahnya ke *database* utama (MySQL). Dan pada saat *request* ke-2 sampai ke-4, waktu yang didapatkan sangatlah rendah atau bisa diartikan cepat.

3. Pengujian Sistem Dengan Redis (Tipe Data *Hashes*)

Berikut ini merupakan hasil waktu respon yang dibutuhkan dari *request* kedua, ketiga dan keempat serta nilai rata-rata respon waktunya jika menggunakan tipe data *hashes*.

Tabel 6 Hasil Pengujian Redis dengan Jenis Tipe Data Hashes

Jumlah Data	Req. ke-2 (s)	Req. ke-3 (s)	Req. ke-4 (s)	Rata-rata (s)
200 Ribu	0,450	0,478	0,455	0,461
400 Ribu	0,435	0,437	0,459	0,444
600 Ribu	0,466	0,513	0,468	0,482
800 Ribu	0,502	0,476	0,47	0,483
1 Juta	0,437	0,445	0,473	0,452
1,2 Juta	0,472	0,484	0,471	0,476
1,4 Juta	0,476	0,474	0,478	0,476
1,6 Juta	0,480	0,478	0,481	0,480
1,8 Juta	0,475	0,464	0,482	0,474
2 Juta	0,473	0,473	0,485	0,477
2,2 Juta	0,488	0,478	0,488	0,485
2,4 Juta	0,484	0,471	0,492	0,482
2,6 Juta	0,486	0,472	0,493	0,484
2,8 Juta	0,488	0,473	0,496	0,486
3 Juta	0,493	0,478	0,499	0,490
3,2 Juta	0,495	0,475	0,502	0,491
3,4 Juta	0,495	0,475	0,505	0,492
3,6 Juta	0,499	0,477	0,507	0,495
3,8 Juta	0,502	0,478	0,510	0,497
4 Juta	0,504	0,479	0,513	0,499
4,2 Juta	0,506	0,479	0,516	0,500
4,4 Juta	0,509	0,485	0,518	0,504

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

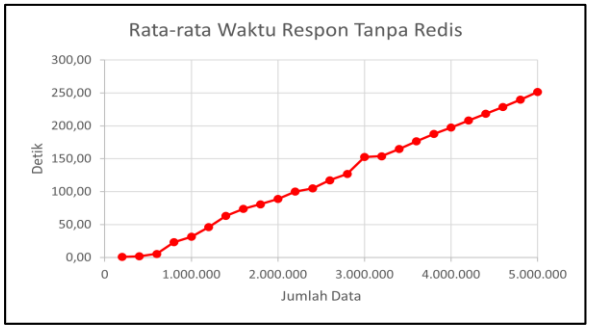
4,6 Juta	0,511	0,484	0,521	0,506
4,8 Juta	0,514	0,486	0,524	0,508
5 Juta	0,516	0,488	0,527	0,510

Sama seperti saat menggunakan tipe data *strings*, Pada *request* pertama tidak dimasukkan pada tabel karena data belum ada pada redis, sehingga *server* melakukan perintahnya ke *database* utama (MySQL). Dan pada saat request ke-2 sampai ke-4, waktu yang didapatkan sangatlah rendah atau bisa diartikan sangat cepat

B. Evaluasi

Tahap ini merupakan informasi hasil rata-rata waktu respon yang disajikan salam bentuk grafik.

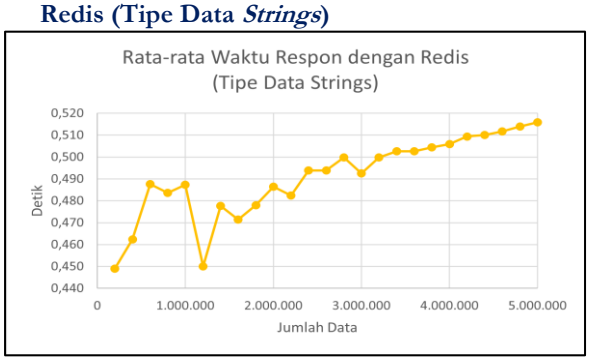
1. Grafik Rata-rata Waktu Respon Jika Tanpa Redis



Gambar 6. Grafik Rata-rata Waktu respon Jika Tanpa Redis

Dari grafik sebelumnya terlihat bahwa waktu respon akan selalu meningkat seiring dengan jumlah data yang selalu bertambah banyak dikarenakan balikan pada MySQL tidak ditangani dengan baik ke redis sehingga menyebabkan proses pada *server* dan *database* utama akan terus besar.

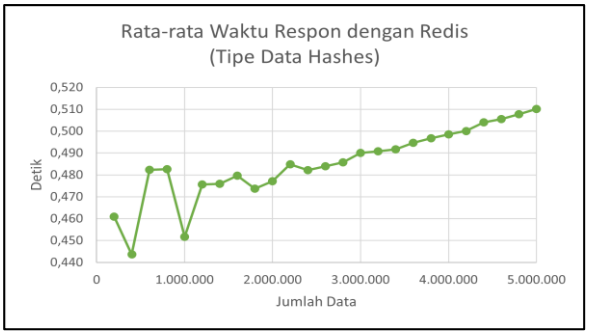
2. Grafik Rata-rata Waktu Respon Jika Dengan Redis (Tipe Data *Strings*)



Gambar 7. Grafik Rata-rata Waktu Respon Jika dengan Redis (Tipe Data *Strings*)

Terlihat bahwa terjadi penurunan waktu respon yang didapat dalam melakukan pengambilan data pada redis dengan menggunakan jenis tipe data *strings*. Ini dikarenakan datanya diambil langsung ke memori, pengambilan data ke memori lebih cepat dari pada ke jenis *storage* lain.

3. Grafik Rata-rata Waktu Respon Jika Dengan Redis (Tipe Data *Hashes*)



Gambar 8. Grafik Rata-rata Waktu Respon jika dengan Redis (Tipe Data *Hashes*)

Terlihat juga bahwa dengan menggunakan redis dengan jenis tipe data *hashes* juga bisa menghasilkan waktu respon yang tidak kalah cepat dengan menggunakan tipe data *strings* dan tentunya lebih cepat dari tanpa menggunakan redis.

Secara keseluruhan dari tiga grafik sebelumnya, menunjukkan bahwa implementasi Redis dengan benar, dapat meningkatkan waktu respon sistem dalam memuat data besar, terutama untuk operasi baca. Redis tidak hanya mempercepat waktu respon secara keseluruhan tetapi juga lebih efektif dalam menangani data berukuran besar, menjaga performa yang baik seiring pertambahan data. Berikut ini merupakan tabel ringkasan rata-rata waktu respon yang dibutuhkan sistem dalam memuat data berdasarkan jumlah data yang tersedia.

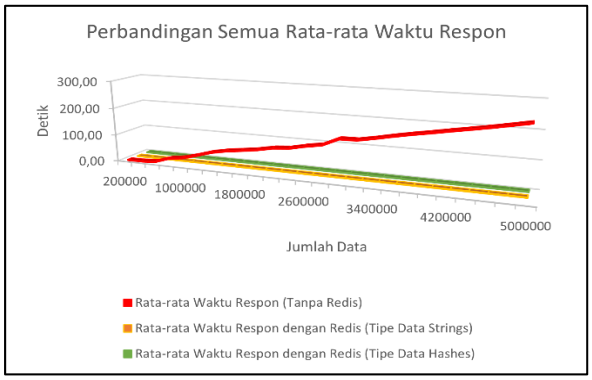
Tabel 7 Hasil Semua Waktu Respon

Jumlah Data	Rata-rata Waktu Respon Tanpa Redis (s)	Rata-rata Waktu Respon Dengan Redis *Strings* (s)	Rata-rata Waktu Respon Dengan Redis *Hashes* (s)
200 Ribu	0,90	0,449	0,461
400 Ribu	1,67	0,462	0,444
600 Ribu	5,43	0,488	0,482
800 Ribu	23,33	0,484	0,483
1 Juta	31,48	0,487	0,452
1,2 Juta	46,25	0,450	0,476
1,4 Juta	62,90	0,478	0,476
1,6 Juta	73,66	0,471	0,480
1,8 Juta	80,88	0,478	0,474
2 Juta	88,97	0,486	0,477
2,2 Juta	100,04	0,482	0,485
2,4 Juta	104,90	0,494	0,482
2,6 Juta	117,37	0,494	0,484
2,8 Juta	126,97	0,500	0,486
3 Juta	152,85	0,493	0,490
3,2 Juta	153,85	0,500	0,491
3,4 Juta	164,61	0,503	0,492
3,6 Juta	176,61	0,503	0,495
3,8 Juta	187,63	0,504	0,497
4 Juta	197,60	0,506	0,499
4,2 Juta	208,19	0,509	0,500
4,4 Juta	218,30	0,510	0,504

EXPLORE : ISSN: 2087-2062, Online ISSN: 2686-181X / DOI: http://dx.doi.org/10.36448/jsit.v15i2.3948

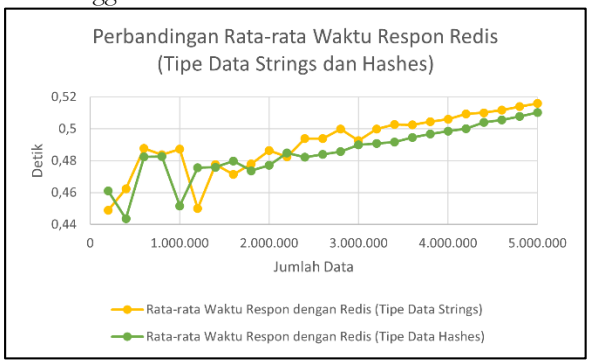
4,6 Juta	228,73	0,512	0,506
4,8 Juta	239,96	0,514	0,508
5 Juta	251,47	0,516	0,510

Dapat dilihat hasil pada kolom ke-1, rata-rata waktu respon yang didapat selalu tinggi atau lambat karena sistem belum mengimplementasikan redis. Sedangkan pada 2 kolom terakhir, dengan menggimplementasikan redis, bisa menghasilkan rata-rata waktu respon yang sangat signifikan menurun atau cepat dan selalu stabil. Jika dibuat dalam 1 grafik maka akan tampil seperti berikut.



Gambar 9. Grafik Perbandingan Semua Waktu Respon

Dari hasil akhir dari perbandingan waktu respon yang ditampilkan pada satu grafik menunjukkan bahwa jika tanpa menggunakan redis ketika data bertambah banyak, maka waktu yang di butuhkan untuk memuat data akan selalu tinggi atau lama.



Gambar 10. Grafik Perbandingan Waktu Respon antara Tipe Data *Strings* dengan *Hashes*

Sedangkan jika sistem telah mengimplementasikan redis, baik itu dengan tipe data *strings* atau *hashes* walaupun data bertambah banyak, hasil rata-rata waktu respon pada *request* ke-2 dan seterusnya selama belum ada perubahan data, hanya dibutuhkan waktu yang singkat dan tentunya stabil dalam memuat jumlah data yang semakin besar. Dan jika dilihat pada gambar sebelumnya antara manakah yang lebih baik antara tipe data *strings* dengan *hashes* untuk kasus masalah ini?. jawabannya adalah pada kasus penelitian ini, tipe data *hashes* mempunyai kinerja yang cepat.

4. Kesimpulan

Dari hasil pengujian dan evaluasi mengenai waktu respon yang didapat dalam memuat data, dapat disimpulkan bahwa dengan menggunakan redis untuk jumlah data sebanyak 5 juta diperoleh rata-rata waktu respon 0,516 detik untuk jenis struktur data *strings*, dan 0,510 detik untuk struktur data *hashes*. Ini lebih cepat dibandingkan tanpa menggunakan redis diperoleh rata-rata waktu respon 251,47 detik. Maka terjadi penurunan sebesar 99,7%. Yang artinya terdapat peningkatan kinerja sebesar 489,05%. Redis cocok untuk digunakan khususnya ketika butuh untuk proses *caching* data. Dan jika dilihat perbandingan dua respon waktu dari dua penggunaan tipe data pada redis untuk sistem *caching*, tipe data *hashes* adalah yang paling efisien, dikarenakan pada saat pengambilan datanya langsung bisa ditampilkan dalam bentuk paginasi karena respon data pada redisnya sudah berupa *array* sehingga waktu yang dibutuhkan untuk memuat datanya walaupun dengan *volume* yang semakin besar, rata-rata waktu yang dihasilkan tetap stabil. Sedangkan untuk tipe data *strings* harus diubah dulu dari *json* ke bentuk *array* agar bisa diproses untuk dipaginasi. Dan rata-rata waktu yang dihasilkan berdasarkan *volume* data yang terus bertambah pun tetap bisa mendapatkan nilai yang stabil. Dengan demikian, jika redis salah satunya digunakan sebagai solusi untuk masalah pada suatu sistem yang sudah mengalami penurunan kinerjanya ini sangat cocok jika di implementasikan dengan benar. Dan tidak lupa untuk selalu memantau kapasitas memori yang masih tersedia guna memastikan redis bisa bekerja secara optimal.

Akan tetapi masih terdapat beberapa kekurangan, sehingga diharapkan dapat dikembangkan lagi. Adapun yang disarankan oleh peneliti yaitu, Untuk sistem sekarang hanya menggunakan 2 GB RAM, Sehingga jika nanti batas maksimal RAM sudah penuh maka perlu adanya penambahan ukuran kapasitas RAM pada sisi *server* agar data yang ditampung dengan *limit* besar bisa tersimpan dengan baik. Dan jika pada sisi *database* atau kode program masih bisa dilakukan normalisasi dan optimasi sangat disarankan untuk dibuat lebih efisien lagi, Sehingga performanya akan lebih meningkat juga.

5. Daftar Pustaka

[1] D. Oktaviani, F. S. Papilaya, dan P. F. Tanaem, "Perancangan Aplikasi E-Menu Restaurant dengan Menggunakan Cloud Computing dan Serverless Architecture Lambda," *Explor. Sist. Inf. Dan Telematika*, vol. 12, no. 1, hlm. 1, Apr 2021, doi: 10.36448/jsit.v12i1.1887.

[2] Aziz dan Yaya Mulyana Abdul, "Model Kebijakan Peningkatan Laporan Kematian dalam Administrasi Kependudukan dan Catatan Sipil di Kabupaten Bandung Barat," *Sosiohumaniora*, vol. 19, no. 2, hlm. 140–148, 2019.

[3] D. Irawan dan A. T. Hidayat, "Rancang Bangun Dashboard Kepegawaian Sekolah Tinggi Ilmu Ekonomi Musi Rawas (Stie Mura) Lubuklinggau," vol. 9, 2019.

[4] W. F. Ramadhan, W. N. Dewi, dan C. Nas, "Aplikasi Web Portal Manajemen Informatika Berbasis Website Dengan Menggunakan Framework Codeigniter Dan Mysql Pada Universitas Catur Insan Cendekia," *J. Digit*, vol. 10, no. 2, hlm. 124, Des 2020, doi: 10.51920/jd.v10i2.164.

[5] A. Frisdayanti, "Peranan Brainware Dalam Sistem Informasi Manajemen," vol. 1, 2019.

[6] A. M. Rosad, "Implementasi Pendidikan Karakter Melalui Managemen Sekolah," *Tarbawi J. Keilmuan Manaj. Pendidik.*, vol. 5, no. 02, hlm. 173, Des 2019, doi: 10.32678/tarbawi.v5i02.2074.

[7] G. H. Irawan dan N. Ramdhani, "Analisa Performa Cache Database Redis dan MySQL dengan PHP".

[8] F. Febriyani, E. S. Pramukantoro, dan F. A. Bakhtiar, "Perbandingan Kinerja Redis, Mosquitto, dan MongoDB sebagai Message Broker pada IoT Middleware".

[9] M. G. Khoshkholgh, K. Navaie, K. G. Shin, V. C. M. Leung, dan H. Yanikomeroglu, "Caching or No Caching in Dense HetNets?," dalam *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakesh, Morocco: IEEE, Apr 2019, hlm. 1–7. doi: 10.1109/WCNC.2019.8885724.

[10] R. Somya dan T. M. E. Nathanael, "Pengembangan Sistem Informasi Pelatihan Berbasis Web Menggunakan Teknologi Web Service Dan Framework Laravel," *J. Techno Nusa Mandiri*, vol. 16, no. 1, hlm. 51–58, Mar 2019, doi: 10.33480/techno.v16i1.164.

[11] M. I. Zulfa, A. Fadli, dan A. W. Wardhana, "Desain Model View Controller Dalam Implementasi Redis Server Untuk Mempercepat Akses Data Relasional," 2019.

[12] A. P. Meriani, D. R. Asih, dan S. Annisa, "Pengujian Distributed Cached Database Dengan Menggunakan Redis Pada Aplikasi MaBaUS".

[13] M. I. Zulfa, A. Fadli, dan A. W. Wardhana, "Application caching strategy based on in-memory using Redis server to accelerate relational data access," *J. Teknol. Dan Sist. Komput.*, vol. 8, no. 2, hlm. 157–163, Apr 2020, doi: 10.14710/jtsiskom.8.2.2020.157-163.

[14] F. A. Isfahani dan F. Nugraha, "Implementasi Load Balancing NGINX dan Mongodb Cluster serta Mekanisme Redis Caching".

[15] S. Rohaeni, "Pengembangan Sistem Pembelajaran Dalam Implementasi Kurikulum 2013 Menggunakan Model Addie Pada Anak Usia Dini," *Instruksional*, vol. 1, no. 2, hlm. 122, Apr 2020, doi: 10.24853/instruksional.1.2.122-130.

[16] H. Yulianti, "Pemanfaatan Sistem Pelatihan E-Learning Pada Pengembangan Kinerja Karyawan di Masa Pandemi Covid-19 Dengan Pengujian ISO 9126," *MULTINETICS*, vol. 7, no. 1, hlm. 65–81, Okt 2021, doi: 10.32722/multinetics.v7i1.3769.

[17] J. Budiraharjo, M. Milisani, dan Y. Marosani, "Pengaruh Proses Rekrutmen Dan Seleksi Terhadap Kualitas Kerja Pegawai Badan Perencanaan Pembangunan Daerah Dki Jakarta," *J. Manaj.*, 2022.

[18] S. M. Hadi dan A. Samad, "Sistem Informasi Pengolahan Data Bantuan Beasiswa Siswa Miskin (BSM) Pada Kantor Wilayah Kementerian Agama Provinsi Maluku Utara," *J. Ilm. Ilk. - Ilmu Komput. Inform.*, vol. 2, no. 1, Jan 2019, doi: 10.47324/ilkominfo.v2i1.15.