Jurnal Ilmu Komputer Agri-Informatika

Tersedia secara daring di:
https://jurnal.ipb.ac.id/index.php/jika

Volume 10 Nomor 1 halaman 57–69
eISSN: 2654-9735, pISSN: 2089-6026

Evaluasi Efisiensi Kinerja *Object Relational Mapping* pada Web API Point of Sale Menggunakan ISO/IEC 25010

Evaluation Of Object Relational Mapping Performance Efficiency On Web Api Point Of Sale Using ISO/IEC 25010

NUR RAHMAT DWI RIYANTO[1*], MUHAMMAD ALFIAN[1], UMI LAILI YUHANA[1]

Abstrak

Aplikasi Point of Sale adalah aplikasi yang menangani suatu kegiatan transaksi penjualan dan pembelian produk. Pada penelitian ini, Aplikasi Point of Sale berbasis Android tersebut dikembangkan dengan menggunakan pendekatan *Object Relational Mapping* untuk mengatasi ketidaksesuaian impedansi (*impedance mismacth*) antara paradigma pemrograman berbasis objek dan basis data relasional dan meningkatkan produktivitas, kinerja, dan kemudahan pemeliharaan sistem, sehingga penerapan pendekatan *Object Relational Mapping* tersebut perlu dilakukan evaluasi untuk mengukur kualitas suatu sistem. ISO/IEC 25010 merupakan salah satu standar pengukuran sistem internasional yang memiliki enam karakteristik pengukuran salah satunya adalah pengukuran *performance efficiency* (efisiensi kinerja) suatu sistem. *Performance efficiency* (efisiensi kinerja) memilik tiga sub karakteristik yaitu *time behavior* (perilaku waktu), *resource utilization* (pemanfaatan sumber daya), dan *capacity* (kapasitas). Web API *Point of Sale* yang menerapkan pendekatan *Object Relational Mapping* (ORM) telah memenuhi standar *performance efficiency* (efisiensi kinerja) yang lebih baik di bandingkan dengan Web API yang tidak menggunakan pendekatan *Object Relational Mapping* dengan rata-rata selisih *response time* 14.8 *milisecond* ketika di uji per satu *user*. Web API yang menerapkan pendekatan *Object Relational Mapping* memiliki *response time* rata-rata yang lebih cepat dan *throughput* yang lebih tinggi ketika diuji dengan banyak *user* secara bersamaan.

Kata Kunci**:** Efisiensi kinerja, JPA, ISO/IEC 25010, *Object Relational Mapping*, *Point of Sale*, WEB API.

Abstract

The Point of Sale application is an application that handles product sales and purchase transactions. This study developed the Android-based Point of Sale application using the Object Relational Mapping approach to overcome (impedance mismatch) and increase productivity, performance, and system maintenance, so the application of the Object Relational Mapping approach needs to be evaluated to measure the quality of a system. ISO/IEC 25010 is one of the international system measurement standards, which has six measurement characteristics, one of which is the measurement of the performance efficiency of a system. Performance efficiency has three sub-characteristics: time behaviour, resource utilization, and capacity. Web API Point of Sale that implements the Object Relational Mapping (ORM) approach has met better performance efficiency standards compared to Web API that does not use the Object Relational Mapping approach with an average response time difference of 14.8 milliseconds when tested per 1 user. Furthermore, Web API applied the Object Relational Mapping approach and has a faster average response time and higher throughput when tested with many users simultaneously.

*Keywords**:** JPA, ISO/IEC 25010, Object Relational Mapping, performance efficiency, Point of Sale, WEB API.*

[1] Jurusan Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember
* Penulis Korespondensi: Tel/Faks: +6285726049824; Surel: nrahmatd@gmail.com

PENDAHULUAN

Aplikasi *point of sale* (PoS) adalah sebuah sistem yang terdiri dari perangkat keras dan perangkat lunak yang dirancang untuk memenuhi kebutuhan bisnis dan dapat diintegrasikan dengan berbagai perangkat pendukung untuk mempercepat proses transaksi. Sebagian besar perusahaan perdagangan menggunakan sistem PoS untuk menunjang kegiatan bisnisnya. Secara umum, PoS memungkinkan terjadinya transaksi, termasuk penggunaan mesin kasir (Hendry 2010).

Dengan meningkatnya penggunaan dan perkembangan teknologi *smartphone*, sistem PoS saat ini dapat dikembangkan dan dijalankan pada platform *smartphone*. Pada penelitian ini, dikembangkan aplikasi PoS yang berjalan pada perangkat *smartphone* dengan sistem operasi populer, salah satunya adalah Android. Sistem Operasi Android merupakan *project open-source* yang dirancang untuk perangkat *mobile*. Berdasarkan data statistik sistem operasi perangkat *mobile* tahun 2023 menjelaskan bahwa sistem operasi perangkat *mobile* yang paling banyak digunakan sampai saat ini dan memiliki pengguna terbanyak adalah sistem operasi Android (Statista 2023).

Aplikasi PoS berbasis Android yang dikembangkan dalam penelitian ini memiliki kemampuan untuk menangani proses transaksi penjualan dan pembelian produk. Aplikasi ini dikembangkan dengan menggunakan pendekatan O*bject Relational Mapping* untuk mengatasi ketidaksesuaian impedansi *(impedance mismatch*) antara paradigma pemrograman berorientasi objek dan paradigma basis data relasional. Beberapa ketidaksesuaian tersebut mencakup aspek *granularity*, *subtypes*, *identity*, *association*, dan *data navigation* (Bauer 2005).

Object Relational Mapping (ORM) adalah suatu teknik yang memungkinkan untuk mentransparansi dan mengotomasi dari *object persistence* ke dalam tabel pada basis data dengan menggunakan metadata yang mendeskripsikan pemetaan antara objek dan basis data (Bauer 2005). ORM melakukan pemetaan tabel pada basis data relasional dengan suatu kelas (*class)* yang ada dalam bahasa pemrograman berorientasi objek. Salah satu framework *Object Relational Mapping* yang digunakan oleh peneliti adalah framework JPA/Hibernate. Framework JPA/Hibernate merupakan framework ORM yang populer dan mendukung pengembangan sistem dengan arsitektur *high cohesion* and *low coupling* (Torres *et al.* 2017). Dengan kemampuan yang dimiliki framework ORM tersebut, dapat meningkatkan produktifitas dalam pengembang sistem PoS, meningkatkan kinerja sistem dan memudahkan dalam memelihara sistem (Bauer 2005).

Untuk mendapatkan hasil yang diharapkan dari aplikasi PoS pada sisi performa yang sesuai dengan harapan para ahli ketika menggunakan aplikasi menerapkan *framework* ORM, maka perlu diadakannya penilaian. Penilaian terhadap sebuah sistem informasi adalah penilaian yang dilakukan dengan mengacu pada standar yang diakui secara internasional untuk aspek kualitas dari sistem informasi. Menurut Miguel *et al*. (2014), terdapat beberapa standar pengukuran yang telah dikembangkan seperti model McCall, Boehm, FURPS, Dromey, ISO/IEC 9126, dan ISO/IEC 25010. Di antara semua model pengukuran tersebut, ISO/IEC 25010 dianggap sebagai model pengukuran yang paling lengkap dalam aspek pengukuran. Model ISO/IEC 25010 mencakup delapan aspek atau karakteristik yaitu: *functional suitability* (kegunaan fungsional), *performance efficiency* (efisiensi kinerja), *compatibility* (kompatibilitas), *usability* (kegunaan), *portability* (portabilitas), *security* (keamanan), *maintainability* (kemudahan perawatan), dan *reliability* (keandalan) (BS ISO/IEC 25010:2011, 2011).

Pada penelitian sebelumnya yang dilakukan oleh Huang *et al.* (2022) mengusulkan HBSniff *tools* untuk manganalisa dan mendeteksi *code smell* pada sebuah arsiteksur aplikasi yang menggunakan pendekatan *Object Relational Mapping* (ORM) Java Hibernate. Kelebihan dari penelitian ini adalah peneliti dapat mengembangkan, menguji,

dan mendokumentasikan *tool* HBSniff yang dapat mendeteksi 14 *code smell* dan mampu meningkatkan domain *low coupling* dan *high cohesion* dalam pengembangan sistem menggunakan *framework* ORM Java Hibernate.

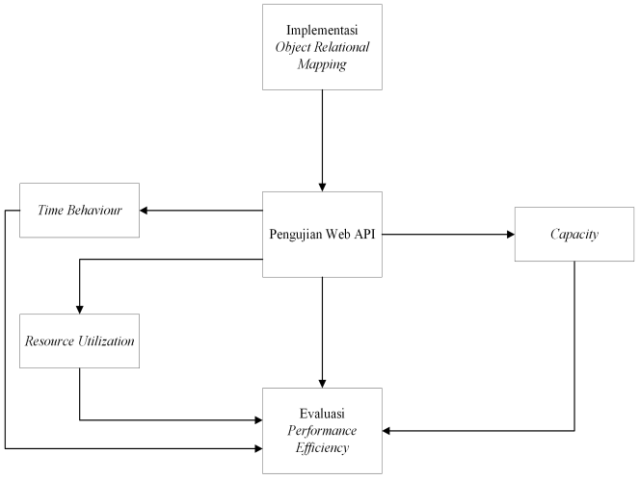
Penelitian yang dilakukan Colley *et al. (2018*) menjelaskan dampak dalam penggunaan *Object Relational Mapping* (ORM) pada kinerja kueri basis data relasional dan menyimpulkan dengan menyarankan arah penelitian di masa depan untuk membantu mengurangi masalah yang ditemukan dengan ORM dan menandai beberapa solusi potensial. Kelebihan dari penelitian ini adalah peneliti dapat menyimpulkan statistik kinerja ORM pada *layer* kueri basis data dan membandingkan performa kinerja ORM dengan kinerja kueri basis data yang tidak menggunakan pendekatan ORM.

Penelitian yang dilakukan Riyanto *et al*. (2018) memberikan gambaran bahwa pendekatan *Object Relational Mapping* (ORM) dapat menyelesaikan masalah ketidaksesuaian impedansi (*impedansi mismatch*) pada aspek asosiasi antara paradigma pemrograman berorientasi objek dengan basis data relasional yang digunakan untuk mengembangkan aplikasi PoS. Kelebihan dari penelitian ini adalah peneliti mampu mengimplementasikan *framework* ORM dengan LinQ dan memberikan keuntungan dalam pengembangan aplikasi dengan menyelesaikan masalah ketidaksesuaian impedansi (*impedance mismatch*).

Penelitian ini akan mengukur karakteristik *performance efficiency* (efisiensi kinerja) Aplikasi PoS setelah mengimplementasikan *framework Object Relational Mapping* pada sistem. Pengukuran *performance efficiency* (efisiensi kinerja) menurut situs ISO 25010 (2022) memiliki tiga sub karakteristik yaitu: *time behavior, resource utilization* dan *capacity*. Penelitian ini dilakukan dengan tujuan untuk mengevaluasi kualitas sistem informasi PoS yang menggunakan *framework* ORM dalam *Software Quality Assurance* (SQA), sehingga dapat memenuhi harapan dan standar kualitas yang diinginkan oleh pengguna. Hasil penelitian ini memberikan saran dan referensi yang bermanfaat bagi para pengembang aplikasi dalam pengembangan aplikasi menggunakan *framework* ORM di masa mendatang. Penelitian ini juga dilakukan untuk mengevaluasi sejauh mana tingkat efisiensi kerangka kerja *Object Relational Mapping* (ORM) dalam melakukan fungsinya di dalam sistem yang belum pernah diuji pada penelitian-penelitian sebelumnya.

METODE

Penelitian ini terdiri dari beberapa tahapan, yaitu implementasi *Object Relational Mapping*, pengujian Web API, dan evaluasi *performance efficiency* (efisiensi kinerja). Tahapan penelitian dapat dilihat pada Gambar 1.



Gambar 1 Tahapan penelitian

Implementasi *ORM*

Penerapan pendekatan *ORM* dilakukan pada Web API Aplikasi PoS dengan menggunakan suatu *framework* ORM yaitu Spring Boot Data JPA pada tahapan ini. Menurut Bauer (2005), penerapan pendekatan *ORM* dilakukan bertujuan untuk mengatasi ketidaksesuaian impedansi *(impedance mismatch*) antara paradigma pemrograman berorientasi objek dan paradigma basis data relasional yang digunakan pada pengembangan Aplikasi PoS. Web API ORM Aplikasi PoS dirancang dengan menyesuaikan proses pada Web API PoS yang telah dikembangkan sebelumnya dengan menggunakan bahasa *native* PHP.

Pengujian Web API

Tahapan ini digunakan untuk menguji dan mengevaluasi Web API yang dikembangkan dengan menggunakan *framework* ORM dimulai dari melakukan *request* Web API sesuai fungsinya, kemudian membandingkan kinerjanya dengan Web API yang telah di kembangkan sebelumnya dengan yang tidak menggunakan *framework* ORM. Perbandingan tersebut didasarkan pada pendekatan ISO/IEC 25010 dengan aspek *performance efficiency* (BS ISO/IEC 25010:2011, 2011).

Evaluasi Efisiensi Kinerja (*Performance Efficiency)*

Pada tahapan ini evaluasi *performance efficiency* dan analisa perbandingan Web API dengan *framework* ORM dengan Web API yang tidak menggunakan *framework* ORM dilakukan dengan memperhatikan 3 sub karakteristik yaitu *time behaviour*, *resource utilization* dan *capacity* (ISO 25010, 2022).

1. *Time Behaviour* (perilaku waktu), yaitu sejauh mana suatu sistem memenuhi persyaratan saat menjalankan fungsinya, termasuk waktu tanggapan (*response time*) dan pemrosesan serta laju *throughput*.
2. *Resource utilization* (pemanfaatan sumber daya), yaitu sejauh mana jumlah dan jenis sumber daya yang digunakan oleh suatu sistem pada saat menjalankan fungsinya, memenuhi persyaratan.
3. *Capacity* (kapasitas), yaitu sejauh mana batas maksimum suatu produk atau parameter sistem memenuhi persyaratan.

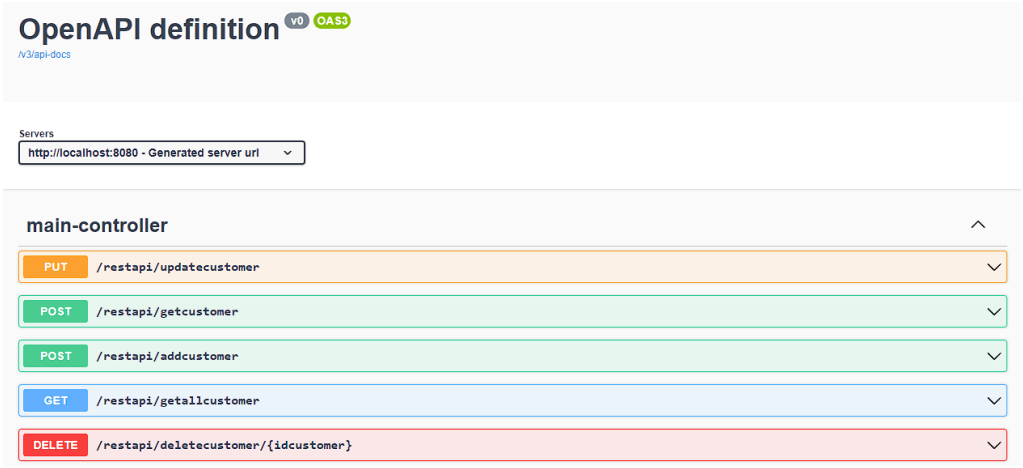
HASIL DAN PEMBAHASAN

Implementasi *Object Relational Mapping* (ORM)

Penerapan pendekatan ORM pada Aplikasi PoS dilakukan dengan menggunakan *framework* Spring Boot. Spring Boot adalah *tool* sumber terbuka (*open-source*) yang mempermudah penggunaan *framework* berbasis Java untuk membuat *microservices* dan aplikasi berbasis *website* (Microsoft 2023). Dengan menggunakan *framework* Spring Boot, pengembangan aplikasi Web API PoS lebih mudah, cepat, dan aman (Spring Boot 2023). Spring Boot mendukung pendekatan ORM yang disebut dengan Spring Data JPA (Java Persistence API) dan Hibernate (Spring Data Access 2023). Pada penelitian ini penulis menggunakan Spring Data JPA untuk mengembangkan Web API ORM.

Arsitektur aplikasi PoS yang terdiri dari 3 tingkat (*tier*) yaitu *presentation tier, logic tier*, dan *data tier.* Implementasi pendekatan ORM berada pada tingkat logika (*logic tier*) berupa aplikasi Web API yang dikembangkan dengan dengan menggunakan *framework* Spring Boot Data JPA versi 2.7.9. Web API ORM yang dikembangkan memiliki layanan (*services*) seperti proses menampilkan data *customer*, menambah data *customer*,

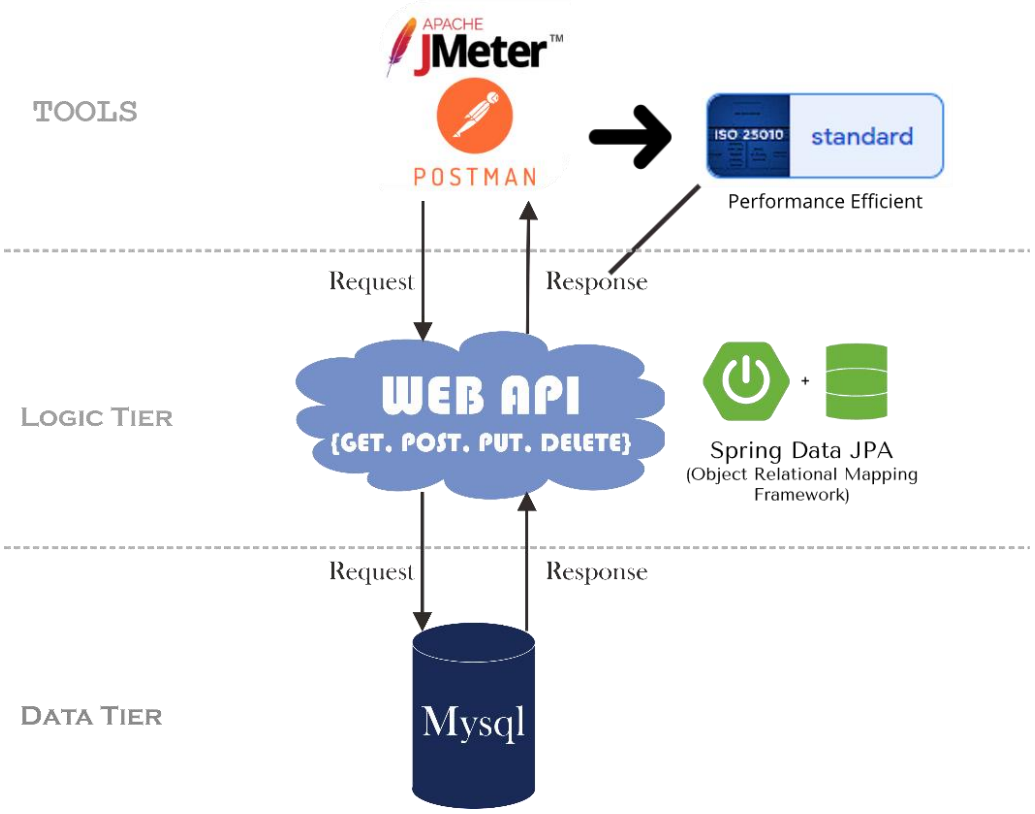
mengubah data customer, dan menghapus data *customer* seperti yang di sajikan pada Gambar 2.



Gambar 2 Aplikasi Web API Point of Sale yang menggunakan ORM

Pengujian Web API

Penelitian ini melakukan pengujian dengan menggunakan dua *tools* untuk menguji *performance efficiency* pada Aplikasi PoS yaitu Postman dan JMeter. Kedua *tools* tersebut digunakan untuk menguji *response time*, *resource utilization*, dan *capacity* dari *framework* ORM yang diimplementasikan pada Web API Aplikasi PoS. Kemudian dilakukan perbandingan hasil *response time* Web API Aplikasi PoS yang mengimplementasikan *framework* ORM Spring Boot yang dikembangkan menggunakan bahasa Java/Kotlin versi 11 dengan *response time* Web API Aplikasi PoS yang tidak mengimplementasikan *framework* ORM yang dikembangkan menggunakan bahasa *Native* PHP versi 5.6.



Gambar 3 Arsitektur pengujian Web API Point of Sale

Tabel 1 *Access* Web API per 1 *user* dengan 12,500 data *customer*

Request	Status	*Response Time ORM* (*ms*)	*Response Time Not ORM* (*ms*)	*Difference Response Time (ms)*	*Response Size ORM*	*Response Size Not ORM*
Get All Customer	200 OK	138	164	26	3.2 MB	4.92 MB
Get Customer By Id	200 OK	5	24	19	433 B	491 B
Add Customer	200 OK	13	24	11	214 B	303 B
Update Customer	200 OK	10	25	15	214 B	303 B
Delete Customer	200 OK	26	29	3	214 B	303 B

Gambar 3 merupakan arsitektur pengujian Web API PoS. Pengujian di mulai dengan melakukan simulasi *request* ke lapisan atau tingkat logika (*logic tier*) Web API dari tingkat *tools* (*tools tier*) menggunakan perangkat lunak Postman dan JMeter. Kemudian dari lapisan atau tingkat logika (*logic tier*), masing-masing Web API meneruskan *request* ke tingkat data (*data tier*) basis data Mysql untuk membaca atau memanipulasi data yang di perlukan. Ketika transaksi berhasil di tingkat data (*logic tier*) maka basis data akan mengembalikan *response* data ke masing-masing Web API yang berada di tingkat logika (*logic tier*). Terakhir, *response* data dari Web API akan diteruskan kembali ke tingkat *tools* (*tools tier*). Pada *response* antara tingkat logika (*logic tier*) dengan tingkat *tools* (*tools tier*), dilakukan analisis dan pemetaan informasi hasil pengujian dari *tools* yang digunakan untuk mengevaluasi *performance efficiency* Web API dengan menggunakan framework *ORM*.

1. Pengujian Menggunakan Postman

Postman merupakan salah satu *tool* yang di peruntukan bagi pengembang sistem untuk merancang, membangun, dan menguji API (Postman 2023). Pengujian dilakukan dengan mensimulasikan 1 *user* yang mengakses 12,500 data *customer* dari *database* dengan simulasi uji *request* Web API melalui Postman. Hasil dari *response time* masing-masing Web API dapat dilihat pada Tabel 1.

Berdasarkan Tabel 1 terdapat perbedaan *response time* dan *response size* antara Web API yang menggunakan *framework* ORM dengan Web API yang tidak menggunakan ORM. "Request Get All Customer" dengan menggunakan *framework* ORM memiliki *response time* 26 *ms* lebih cepat dibandingkan dengan tidak menggunakan *framework* ORM. Kemudian "Request Get Customer By Id" dengan menggunakan *framework* ORM memiliki *response time* 19 *ms* lebih cepat dibandingkan dengan tidak menggunakan *framework* ORM. "Request Add Customer" dengan menggunakan *framework* ORM memiliki *Response Time* 11 *ms* lebih cepat dibandingkan dengan tidak menggunakan *framework* ORM. "Request Update Customer" dengan menggunakan *framework* ORM memiliki *response time* 15 *ms* lebih cepat dibandingkan dengan tidak menggunakan *framework* ORM. "Request Delete Customer" dengan menggunakan *framework* ORM memiliki *response time* 3 *ms* lebih cepat dibandingkan dengan tidak menggunakan *framework* ORM.

Untuk menghitung rata-rata total selisih perbedaan *response time* dapat menggunakan rumus:

$$Average = \frac{a_1, a_2, a_3, a_4, \ldots, a_n}{n} \quad (1)$$

Keterangan:

Average = Rata-rata

$a_1, a_2, a_3, a_4, \ldots, a_n$ = *Response time*

n = Jumlah *response time*

Dengan menggunakan Persamaan 1 diperoleh nilai rata-rata total selisih perbedaan *response time* sebagai berikut:

Average = *(*26 + 19 + 11 + 15 + 3) / 5 = 14.8 *ms*

Berdasarkan hasil perhitungan rata-rata selisih *response time* ketika Web API menggunakan *framework* ORM memiliki kecepatan rata-rata 14.8 *miliseconds* lebih cepat dibandingkan dengan tidak menggunakan *framework* ORM.

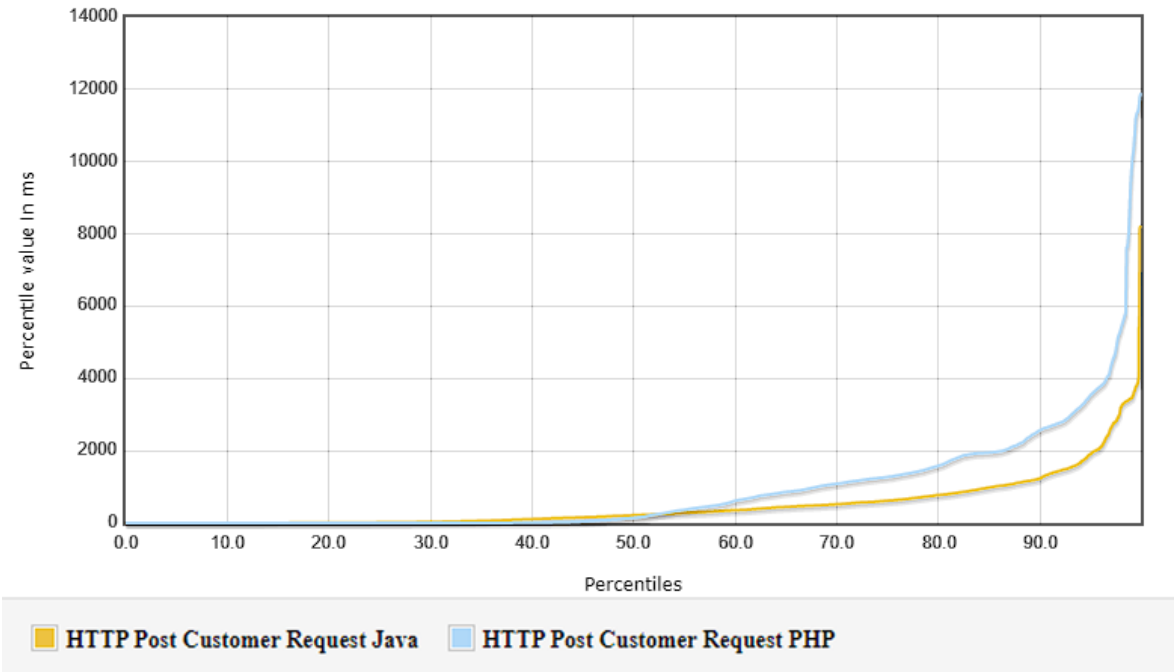
2. Pengujian Menggunakan JMeter

JMeter adalah suatu perangkat lunak *open source* yang dibangun menggunakan bahasa pemrograman Java yang dirancang untuk melakukan pengujian fungsional dan kinerja suatu sistem (JMeter 2023). JMeter awalnya dirancang untuk menguji aplikasi berbasis *website*, dan sekarang telah diperluas ke fungsi pengujian lainnya. Pengujian dilakukan dengan melakukan simulasi ke banyak *user* yang mengakses data *customer* dari *database* melalui *request* masing-masing Web API dengan skenario uji yang telah di buat pada *tool* JMeter. Pengujian Web API PoS menggunakan JMeter dibagi menjadi beberapa pengujian berdasarkan *request method* POST, GET, PUT dan DELETE. Hasil dari masing-masing pengujian berdasarkan *request method* dapat dilihat pada Tabel 2.

Pada Tabel 2 disimulasikan skenario pengujian dengan menambahkan data *customer* dengan *Thread* / *User* 5000 pada masing-masing Web API yang menggunakan *framework* ORM dan Web API yang tidak menggunakan *framework* ORM. Hasilnya adalah Web API yang menggunakan *framework* ORM memiliki rata-rata *response time* lebih cepat yaitu 709 *ms* dibandingkan dengan rata-rata *response time* Web API yang tidak menggunakan framework ORM yaitu 1187 *ms*. Pada aspek *throughput* Web API ORM memiliki transaksi per detik lebih tinggi yaitu 193,46102/s dibandingkan dengan Web API bukan ORM yaitu 193,05392/s. Hasil perbandingan Web API *request* POST data *customer* dapat di lihat pada Gambar 2.

Tabel 2 *Access* Web API *Add Customer* per 5000 *user*

Request	Samples Data	Average (*ms*)	Throughput (s)	Received KB/sec	Sent KB/sec	Avg. Bytes
POST Add Customer Request ORM	5000	709	193,46102	226.95	38.33	1201,2
POST Add Customer Not ORM	5000	1187	193,05392	233.3	41.41	1237,5



Gambar 4 Grafik Perbandingan *response time Post Customer*

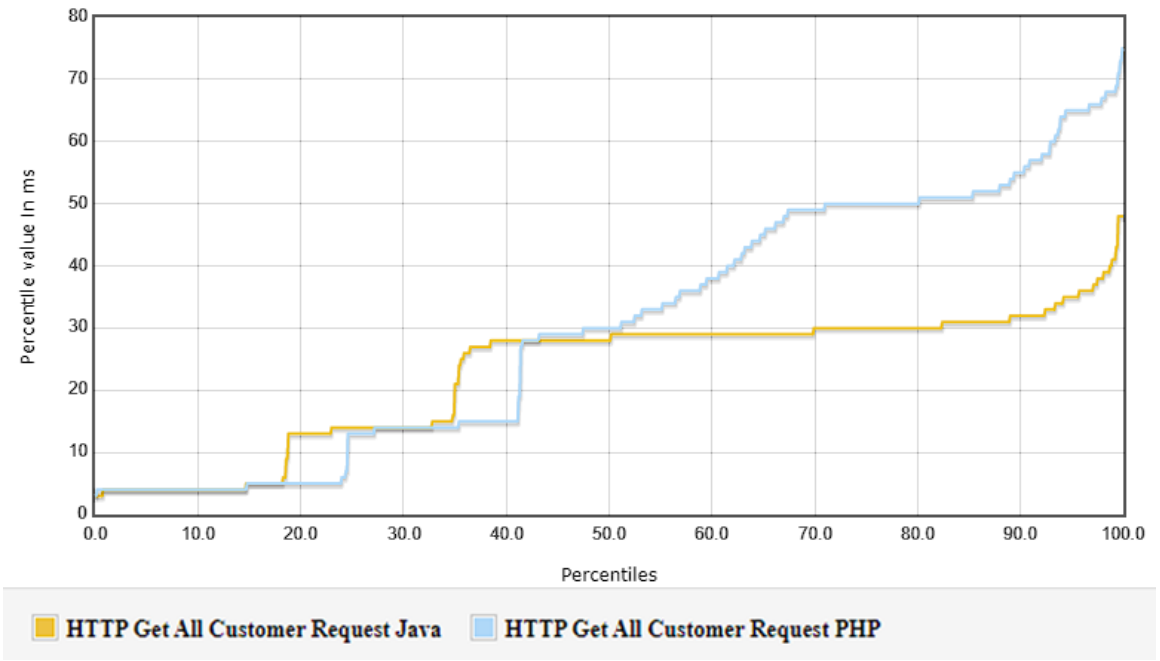
Gambar 4 menjelaskan tentang perbandingan Web API *request* POST data *customer* yang menggunakan *framework* ORM yang ditandai dengan garis berwarna kuning dan Web API yang tidak menggunakan *framework* ORM yang di tandai dengan garis berwarna biru. Web API yang menggunakan ORM cenderung memiliki *response time* terus naik dari waktu ke waktu dengan *response time percentile* adalah 8100 *ms*. Sedangkan Web API yang tidak menggunakan *framework* ORM memiliki *response time percentile* lebih tinggi yaitu 12,000 *ms.*

Tabel 3 menampilkan hasil simulasi skenario pengujian yang mengambil data *customer* sebanyak 10,000 data, dengan *Thread / User* 500. Hasil dari skenario pengujian tersebut adalah Web API yang menggunakan *framework* ORM memiliki rata-rata *response time* lebih cepat yaitu 22 ms dibandingkan dengan rata-rata response time Web API yang tidak menggunakan framework ORM yaitu 30 ms. Pada aspek *throughput* Web API ORM memiliki transaksi per detik lebih tinggi yaitu 8,35143/s dibandingkan dengan Web API bukan ORM yaitu 8,34529/s.

Hasil perbandingan *request* Web API GET data customer dapat dilihat pada Gambar 5. Perbandingan Web API dengan *request* GET data *customer* yang menggunakan *framework* ORM ditandai dengan garis berwarna kuning dan Web API yang tidak menggunakan *framework* ORM yang di tandai dengan garis berwarna biru. Web API yang menggunakan ORM cenderung memiliki *response time* terus naik dari waktu ke waktu dengan *response time percentile* adalah 48 *ms*. Sedangkan Web API yang tidak menggunakan *framework* ORM memiliki *response time percentile* lebih tinggi yaitu 75 *ms.*

Tabel 3 *Access* Web API Get All Customer per 500 *user* dengan data *customer* 10000

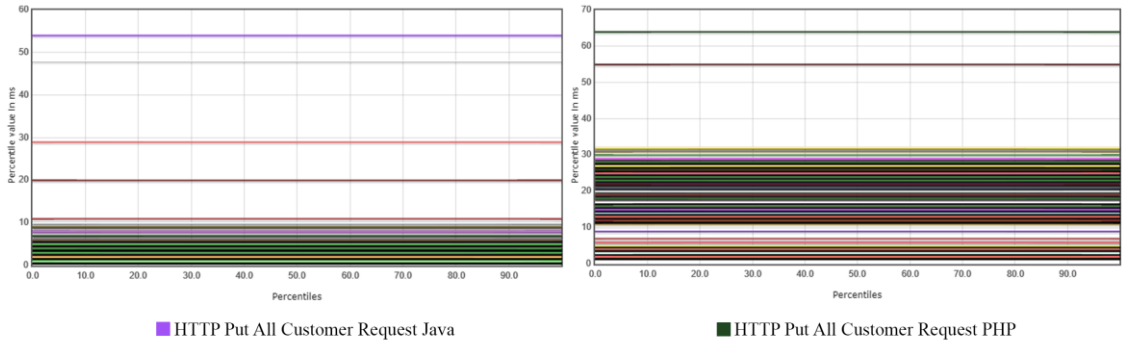
Request	Samples Data	Average (*ms*)	Throughput (s)	Received KB/sec	Sent KB/sec	Avg. Bytes
GET Get All Customer ORM	500	**22**	**8,35143**	3002.38	0.71	368133,6
GET Get All Customer Not ORM	500	30	8,34529	4182.06	0.76	513155,4



Gambar 5 Grafik Perbandingan *response time Get Customer*

Tabel 4 *Access* Web API Edit Customer per 5000 *user* dengan data *customer* 10000

Request	Samples Data	Average (*ms*)	Throughput (s)	Received KB/sec	Sent KB/sec	Avg. Bytes
PUT Edit Customer ORM	5000	**2**	**355,39129**	85.93	115.49	247.6
PUT Edit Customer Not ORM	5000	12	77,98609	22.74	27.04	298.6



Gambar 6 Grafik Perbandingan *response time Per Customer*

Tabel 5. *Access* Web API Delete Customer per 5000 *user* dengan data customer 10000

Request	Samples Data	Average (*ms*)	Throughput (s)	Received KB/sec	Sent KB/sec	Avg. Bytes
DELETE Delete Customer ORM	5000	**1**	**619,19505**	184.57	138.94	305.2
DELETE Delete Customer Not ORM	5000	10	94,65036	27.6	25.88	298.6

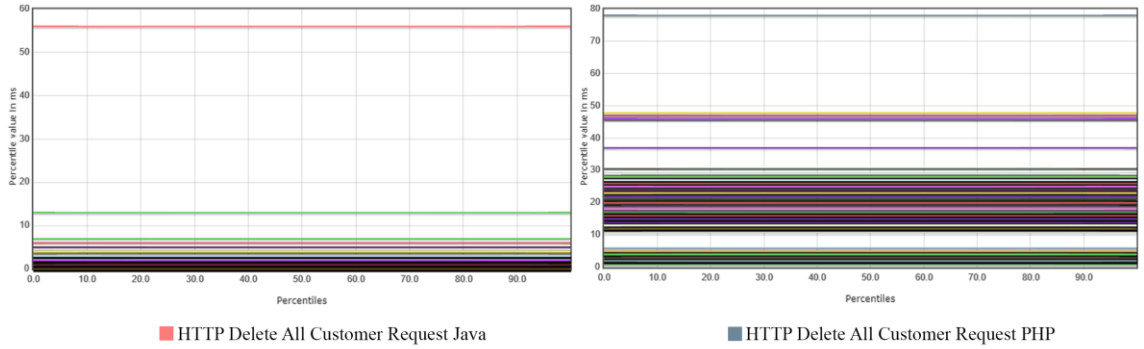
Hasil skenario pengujian untuk meng *update* data customer sebanyak 10,000 data, dengan *Thread / User* 5000 pada setiap Web API disajikan pada Tabel 4. Didapatkan Web API yang menggunakan *framework* ORM memiliki rata-rata *response time* lebih cepat yaitu 2 *ms* dibandingkan dengan rata-rata *response time* Web API yang tidak menggunakan *framework* ORM yaitu 12 *ms*. Pada aspek Throughput Web API ORM memiliki transaksi per detik lebih tinggi yaitu 355,39129/s dibandingkan dengan Web API bukan ORM yaitu 77,98609/s.

Pada Gambar 6 menjelaskan tentang perbandingan Web API dengan *request* PUT data *customer* yang menggunakan *framework* ORM yang ditandai dengan garis berwarna kuning dan Web API yang tidak menggunakan *framework* ORM yang di tandai dengan garis berwarna biru. Web API yang menggunakan ORM cenderung memiliki *response time* terus naik dari waktu ke waktu dengan *response time percentile* adalah 55 *ms*. Sedangkan Web API yang tidak menggunakan *framework* ORM memiliki *response time percentile* lebih tinggi yaitu 65 *ms.*

Tabel 5 menyajikan hasil simulasi skenario pengujian untuk menghapus data *customer* sebanyak 10000 data, dengan *Thread / User* 5000. Didapatkan Web API yang menggunakan *framework* ORM memiliki rata-rata *response time* lebih cepat yaitu 1 *ms* dibandingkan dengan rata-rata *response time* Web API yang tidak menggunakan *framework* ORM yaitu 10 *ms*. Pada aspek Throughput Web API ORM memiliki transaksi per detik lebih tinggi yaitu 619,19505/s dibandingkan dengan Web API bukan ORM yaitu 94,65036/s.

Perbandingan Web API dengan *request* DELETE data *customer* dapat di lihat pada Gambar 7. Web API yang menggunakan *framework* ORM ditandai dengan garis berwarna kuning dan Web API yang tidak menggunakan *framework* ORM yang di tandai dengan garis berwarna biru. Web API yang menggunakan ORM cenderung memiliki *response time* terus naik dari waktu ke waktu dengan *response time percentile* adalah 57 *ms*. Sedangkan

Web API yang tidak menggunakan *framework* ORM memiliki *response time percentile* lebih tinggi yaitu 79 *ms.*



Gambar 7 Grafik Perbandingan *response time Delete Customer*

Evaluasi Efisiensi Kinerja (*Performance Efficiency)*

Penilaian efisiensi kinerja (*performance efficiency*) dalam penelitian ini adalah berdasarkan standar ISO 25010. Ada tiga aspek yang dinilai dari efisiensi kinerja ISO 25010, yaitu perilaku waktu (*time behavior*), pemanfaatan sumber daya (*resource utilization*), dan penilaian kapasitas (*capacity*).

1. Perilaku Waktu (*Time Behaviour*)

Perilaku waktu (*time behaviour*) mengukur berapa waktu yang dibutuhkan suatu sistem untuk menjalankan suatu proses yang diberikan atau biasa disebut dengan *response time*.Dari hasil ekperimen pengujian ini didapatkan bahwa Web API yang menggunakan *framework Object Relational Mapping* (ORM) memiliki rata-rata *response time* 14.8 *miliseconds* lebih cepat di bandingkan dengan Web API yang tidak menggunakan *framework Object Relational Mapping* (ORM). Hal ini telah memenuhi persyaratan pengujian dari pengembangkan perangkat lunak yang lebih memilih menggunakan ORM untuk menyelesaikan permasalahan sebelumnya yaitu ketidaksesuaian impedansi (*impedance mismatch*) antara paradigma permograman berorientasi objek dan basisdata relasional.

2. Pemanfaatan Sumber Daya (*Resource Utilization*)

Pemanfaatan sumber daya digunakan untuk mengukur dan menganalisis total sumber daya digunakana pada saat melakukan pengujian. Salah satu aspek penggunaan sumber daya adalah rata-rata penggunaan memori yang digunakan untuk menerima *response* dari Web API atau disebut dengan *response size*. Pada Tabel 1 dijelaskan pada *request "*Get All Customer"**,** Web API yang menggunakan ORM memiliki *response size* 3.2 MB lebih kecil dibandingkan dengan Web API yang tidak menggunakan ORM dengan *response size* 4.92 MB. Pada *request "*Get Customer By Id", Web API yang menggunakan ORM memiliki *response size* 433 B lebih kecil dibandingkan dengan Web API yang tidak menggunakan ORM dengan *response size* 491 B. Sedangkan pada request "Add Customer", "Update Customer", dan "Delete Customer" Web API yang menggunakan ORM memiliki *response size* rata-rata sama yaitu 214 B lebih kecil dibandingkan dengan Web API yang tidak menggunakan ORM dengan *response size* 303 B.

Dari hasil perbandingan *response size* Web API tersebut, Web API dengan menggunakan *framework* ORM memiliki penggunaan sumber daya memori yang lebih sedikit di banding dengan Web API yang tidak menggunakan ORM. Hal ini telah

memenuhi persyaratan yang telah dibuat pengembang bahwa penggunakan ORM memiliki efek penggunakan memori yang lebih baik dibandingkan dengan tidak menggunakan ORM.

3. Penilaian Kapasitas (*Capacity*)

Penilaian kapasitas bertujuan untuk menganalisis batas maksimal penggunaan suatu sistem. Aspek yang digunakan untuk mengukur penilaian kapasitas di setiap kerangka, yaitu kesesuaian akses pengguna dan peningkatan akses pengguna. Pada pengujian ini dilakukan menggunakan *tools* JMeter dengan skenario uji 500 sampai 10,000 pengguna (*user*) mengakses Web API secara bersamaan (*simultaneously*). Pada penelitian ini, Web API ORM memiliki *response time* yang lebih baik ketika di akses oleh banyak *user* secara bersamaan dibandingkan dengan Web API yang tidak menggunakan ORM seperti pada Gambar 5. Disamping itu, Web API ORM memiliki rata-rata *throughput* lebih tinggi dibandingkan dengan Web API yang tidak menggunakan ORM. Sehingga Web API ORM juga telah memenuhi persyaratan kembali dari sisi kapasitas (*capacity*) yang memiliki *throughput* lebih tinggi dan mampu memberikan *response time* lebih cepat dari Web API sebelumnya ketika di akses banyak *user*.

SIMPULAN

Berdasarkan hasil pengujian *performance efficiency* (efisiensi kinerja) pada Aplikasi PoS, Web API PoS yang menggunakan *framework* ORM mengalami peningkatan efisiensi kinerja yang lebih baik dibandingkan dengan Web API yang tidak menggunakan *framework* ORM. Peningkatan efisiensi kinerja tersebut diukur berdasarkan *time behaviour*, *resource utilization*, dan *capacity*. Web API PoS yang menggunakan framework ORM cenderung memiliki *response time* sebagai parameter pengukuran *time behaviour* yang lebih cepat, *response size* lebih kecil sebagai parameter pengukuran *resource utilization* dan juga memiliki *throughput* yang lebih tinggi sebagai parameter pengukuran *capacity*, sehingga penerapan *framework* ORM pada Web API PoS telah memenuhi standar kualitas ISO/IEC 25010 dalam karakteristik *performance efficiency*.

Untuk mengembangkan penelitian ini selanjutnya, perlu dilakukan pengujian kembali menggunakan ISO/IEC 25010 dengan menambahkan karakteristik lain yaitu *Security* dan *Maintainability*.

DAFTAR PUSTAKA

Anggraini N, Putra MJD and Hakiem N. 2019. *Development of an Islamic Higher Education Institution Tracer Study Information System and It's Performance Analysis using ISO/IEC 25010*. 2019 7th International Conference on Cyber and IT Service Management (CITSM), Jakarta, Indonesia, 2019, pp. 1-6, doi: 10.1109/CITSM47753.2019.8965356.

Barletta V, Caivano D, Colizzi L, Dimauro G, Piattini M. 2023. *Clinical-chatbot AHP evaluation based on "quality in use" of ISO/IEC 25010*. International Journal of Medical Informatics. Vol. 170.

Bauer C, & King G. 2005. *Hibernate in Action*. Retrieved from Manning Publication Co, Unites States of America.

BS ISO/IEC 25010:2011. 2011. *Systems and software Quality Requirements and Evaluation* (SQuaRE) System and software quality models. Japan: Multiple. Distributed through American National Standards Institute (ANSI).

Cahyo PA, Ardiansyah F. 2020. Perancangan Prototipe Mobile User Experience Aplikasi Peningkatan Sumber Daya Desa Menggunakan Metode *Double Diamond.* Jurnal Ilmu Komputer dan Agri-informatika (JIKA). Vol. 7, No. 2, November 2020, hlm. 96-104. DOI: https://doi.org/10.29244/jika.7.2.96-104.

Colley D, Stanier C, and M. Asaduzzaman. 2018. *The Impact of Object-Relational Mapping Frameworks on Relational Query Performance*. 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Southend, UK, pp. 47-52, doi: 10.1109/iCCECOME.2018.8659222.

Firdaus A, Ramadhan A. 2021. Pengembangan Back End Berbasis REST API pada Sistem E-Partisipasi dan E-Inisiatif Patriot Pangan. Jurnal Ilmu Komputer dan Agri-informatika (JIKA). Vol. 8, No. 1, May 2021, hlm. 96-104. DOI: https://doi.org/10.29244/jika.8.1.1-9.

Hakim L, Rochimah S, Fatichah C. 2019. *Evaluation Of Hypernyms And Synonyms Combination For Classification Of Non-functional Requirement Based on ISO/IEC 25010*. Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK). Vol.6, No. 5, Oktober 2019, halaman. 491-500. DOI: 10.25126/jtiik.201961422.

Hendry. 2010. Membangun *Point of Sale* dengan VB 6.0, MySQL, dan PHP. PT Elex Media Komputindo, Jakarta.

Huang Z, Shao Z, Fan G, Yu G, Yang K, Zhou Z. 2022. *HBSniff: A static analysis tool for Java Hibernate object-relational mapping code smell detection*. Journal Science of Computer Programming. Vol.217, 1 May 2022, page. 102778. DOI: 10.1016/J.SCICO.2022.102778.

ISO 25010. 2022. ISO/IEC 25010. Retrieved from https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

JMeter. 2023. *Apache JMeter* [Internet]. [diakses pada 2023 May 19];Tersedia pada: https://jmeter.apache.org

Microsoft. 2023. *What is Java Spring Boot* [Internet]. [diakses pada 2023 May 27];Tersedia pada: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-spring-boot

Miguel JP, Mauricio D, dan Rodríguez G. 2014, *A Review of Software Quality Models for the Evaluation of Software Products*, Int. J. Softw. Eng. Appl., vol. 5, no. 6, pp. 31–53, 2014, doi: 10.5121/ijsea.2014.5603.

Mulyana M, Kumara I, Swamardika I, Saputra K. 2021. Kualitas Sistem Informasi Berdasarkan ISO/IEC 25010: Literature Review. Majalah Ilmiah Teknologi Elektro. vol. 1, *issue*. 1, DOI: https://doi.org/10.24843/MITE.2021.v20i01.P02.

Peak P & Heudecker N. 2006. *Hibernate Quickly*. Manning, United States of America.

Postman. 2023. *What is Postman* [Internet]. [diakses pada 2023 May 19];Tersedia pada: https://www.postman.com

Spring Boot. 2023. *Why Spring Boot* [Internet]. [diakses pada 2023 May 27];Tersedia pada: https://spring.io/why-spring

Spring Data Access. 2023. *Object Relational Mapping (ORM) Data Access* [Internet]. [diakses pada 2023 May 27];Tersedia pada: https://docs.spring.io/spring-framework/reference/data-access/orm.html

Statista. 2023. Mobile operating systems' market share worldwide from 1st quarter 2009 to 4th quarter 2022. Retrieved from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009

Riyanto NRD, Nurfaizah, Bratakusuma T, Utomo FS, Hermanto N. 2018. Implementasi *Object Relational Mapping* (ORM) Pada Aplikasi *Point of Sale* Berbasis Android. *Conference on Information Technology*, Information System and Electrical Engineering (CITISEE) 2018 AMIKOM Purwokerto. ISBN: 978-602-60280-1-3

Torres A, Galante R, Pimenta M, dan Martins A. 2017. *Twenty years of object-relational mapping: A survey on patterns, solutions, and their implications on application design*. Information and Software Technology Journal, (2017), 1-18, 82.

Yuniasri D, Damayanti P, dan Rochimah S. 2020. *Performance Efficiency Evaluation Frameworks Based on ISO 25010*. 10th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), Malang, Indonesia, pp. 254-258, doi: 10.1109/EECCIS49483.2020.9263432.