MIGRATION CODE PADA *BACKEND* CRIMEZONE DARI PHP KE SCALA

Agus Suhendra[1] dan Adam Mukharil Bachtiar[2]

Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia
Jalan Dipatiukur No. 112-116, Coblong, Bandung, Jawa Barat

E-mail: ag.suhendra@gmail.com[1], adammb@outlook.com[2]

Abstract

Crimezone is the application of citizen journalism developed to report crimes in Bandung which is available on windows phone platform. In its use, this app still has some problems in the backend system. One that often occurs is failure in sending image stream data with a large size, the provided web service is slow in delivering the requested data, and the backend system is highly vulnerable of web attacks. The purpose of this research is to improve the performance of Crimezone's backend system. This is achieved by migrating from the current programming language to the new one. The selection of the Scala programming language will improve the performance of the backend system, facilitate use of asynchronous programming, concurrency, and parallelism and improve the security on the system.

Keyword: Crimezone, Code Migration, Scala, *Backend*

Abstrak

Crimezone merupakan aplikasi *citizen journalism* pada *mobile* yang dikembangkan untuk melaporkan bentuk kriminalitas di Kota Bandung yang tersedia dalam platform *windows phone*. Dalam penggunaanya, aplikasi ini masih memiliki beberapa masalah dari segi *backend* yaitu sering terjadinya kegagalan dalam mengirim data *image stream* dengan ukuran yang besar, layanan *web service* yang disediakan lambat dalam memberikan respon data yang diminta, dan sistem *backend*nya sangat rentan terhadap serangan web. Tujuan dari penelitian ini adalah untuk meningkatkan performa dari sistem *backend* Crimezone. Hal ini dicapai dengan memigrasikan bahasa pemrograman yang digunakan ke bahasa pemrograman yang baru. Pemilihan Scala sebagai bahasa pemrograman akan meningkatkan performa sistem *backend*, memudahkan dalam menggunakan pemrograman *asynchronous*, *concurrency*, dan *parallelism* serta meningkatkan keamanan pada sistem.

Kata Kunci: Crimezone, Migrasi Kode, Scala, *Backend*

1. Pendahuluan

Selama beberapa dekade ini teknologi *web* berkembang dengan cepat. Perkembangan ini mengakibatkan banyak bahasa pemrograman baru bermunculan. Bahasa pemrograman baru ini hadir dengan kualitas yang lebih baik sehingga menjadikan dunia *web development* berkembang cepat. Persaingan ini hadir dalam hal performa, keamanan, kesederhanaan sintaks, optimalisasi, dan kemampuan untuk berkembang di masa datang, sehingga banyak *developer* yang memigrasikan kode program atau aplikasinya ke bahasa lain demi menjadikan program yang lebih baik dan cepat. Hal ini juga berlaku untuk aplikasi Crimezone.

Crimezone merupakan aplikasi *citizen journalism* pada *mobile* yang digunakan untuk melaporkan bentuk kriminalitas. Berdasarkan wawancara yang sudah dilakukan dengan *developer* aplikasi Crimezone, diperoleh informasi bahwa aplikasi belum bekerja dengan baik karena masih terdapat beberapa kekurangan dari segi sistem *backend*. Pertama, sistem *backend* saat ini belum bekerja dengan baik dalam melayani akses pengguna. Hal ini dibuktikan de-ngan masih terjadinya kegagalan dalam mengirimkan data *image stream* dengan kapasitas yang besar. Kedua, sistem *backend* belum bekerja dengan cepat dan optimal yang dibuktikan dengan kinerja dari *web service* yang lambat dalam melayani *request* data dari pengguna, sehingga sering terjadinya *request time out*. Ketiga, sistem *backend* belum aman, dikarenakan sintaksis bahasa pemrograman yang digunakan tidak mengikuti standar yang diharuskan.

Secara umum, untuk membangun sistem *backend* yang baik dan bagus ditentukan oleh bahasa pemrograman yang digunakan. Pemilihan bahasa pemrograman yang tepat akan menentukan tingkat kualitas sistem yang dibangun. Ada beberapa kriteria yang menentukan bahasa pemrogra-

man baik dan bagus. Pertama, penggunaan bahasa tersebut dapat menghasilkan kode yang lebih baik; dengan kode yang lebih baik akan menghasilkan performa yang cepat ketika dijalankan. Kedua, penggunaan bahasa tersebut dapat mengatasi masalah *concurrency* dan mendukung *parallelism* sehingga menjadikan program *scalable* [1]. Ketiga, kode yang ditulis secara *default* memiliki keamanan yang tinggi dalam melindung terhadap berbagai jenis serangan *web* [2].

Oleh karena itu, berdasarkan permasalahan yang telah dijelaskan dan berdasarkan studi literatur, didapatkan solusi yaitu dengan memigrasikan kode pada *backend* Crimezone dari PHP ke Scala. Pemilihan Scala sebagai bahasa pemrograman akan meningkatkan performa sistem *backend* [3], memudahkan dalam menggunakan pemrograman *asynchronous, concurrency*, dan *parallelism* serta meningkatkan keamanan pada sistem [4].

2. Metode Penelitian

Untuk memigrasikan sistem *backend* aplikasi Crimezone menggunakan pendekatan *Reference Migration Processes* (ReMiP). ReMiP merupakan model proses umum yang menggambarkan aktivitas yang dibutuhkan sebagai tahapan dalam proses migrasi. Proses-proses atau tahapan yang terjadi pada saat migrasi terdiri dari [3]:

Requirement Analysis

Tahap *requirement analysis* merupakan tahapan pertama di ReMiP. Pada tahap ini hanya dilakukan analisis kebutuhan nonfungsional karena memiliki dampak langsung terhadap pemilihan teknologi. Sementara itu, analisis fungsional tidak perlu dilakukan karena fungsionalitas sistem tidak mengalami perubahan. Selain itu, kebutuhan peralatan migrasi diperiksa dan semua kebutuhan dikelola untuk mengatasi perubahan.

Legacy Analysis

Pada tahap ini dimulai dengan menganalisis *legacy system* kasar. *Legacy system* dinilai kualitas teknis dan nilai bisnisnya.

Target Design

Pada tahap ini, arsitektur, struktur data, dan antarmuka pengguna dari sistem baru didefinisikan. Pada tahap ini juga disertakan *artifact* sementara yang mungkin disertakan selama perubahan. Selain itu, *legacy artifact* dipetakan ke target *artifact* yang sesuai.

Strategy Selection

Pada tahap ini dilakukan pemilihan strategi migrasi yang akan digunakan. Strategi migrasi menjelaskan bagaimana *legacy system* akan dikonversi ke dalam sistem yang dimigrasikan. Strategi yang dipilih adalah strategi *re-implementation* dengan alasan sistem baru di kembangkan ulang berdasarkan ide-ide yang disediakan oleh *legacy software* tanpa mengubah fungsionalitas sistem yang ada. *Reimplementation* dimulai dengan menganalisis *legacy system*, kebutuhan fungsional, dan non-fungsional yang diambil.

Implementation

Pada tahap ini dilakukan implementasi desain sistem yang telah didefinisikan. Selain itu, *legacy system* dimigrasikan sesuai dengan strategi migrasi yang dipilih pada tahapan *strategy selection.*

Test

Pada tahap ini, sistem baru akan diuji, apakah mendukung fungsi yang sama dengan sistem lama.

Deployment

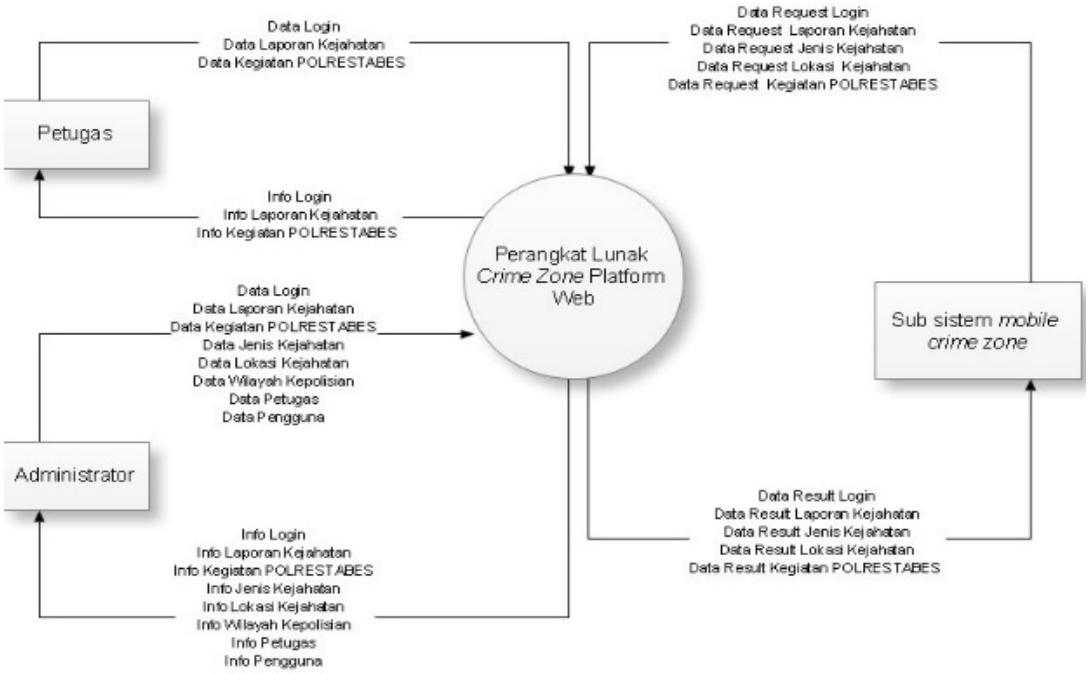
Tahap *deployment* merupakan tahap terakhir dalam tahapan migrasi perangkat lunak. Tahap ini berbicara bagaimana memperkenalkan sistem baru yang sudah dihasilkan dengan menggunakan strategi yang ada.

3. Hasil dan Pembahasan

Analisis Kebutuhan Fungsional Sistem *Legacy*

Analisis kebutuhan fungsional dari sistem *legacy* menggambarkan proses kegiatan yang sedang berjalan pada sistem tersebut. Pada sistem *legacy*, subsistem, yaitu subsistem *web* dan subsistem *mobile*. Pada analisis ini yang akan dibahas hanya subsistem *web*, sedangkan subsistem *mobile* tidak perlu dilakukan analisis. Alasannya, karena migrasi yang dilakukan hanya untuk subsistem *web*. Analisis subsistem *web* ini dilakukan dengan tujuan un-

Gambar 1 Metode Migrasi Perangkat Lunak



Gambar 2 Diagram Konteks Subsistem Web Legacy

tuk mengetahui prosedur dan fungsi apa saja yang dimiliki oleh setiap proses yang ada pada subsistem *Web Legacy*. Analisis kebutuhannya dibagi menjadi dua:

Analisis Fungsional Subsistem *Web Legacy*

Subsistem *Web Legacy* dibangun dengan menggunakan pendekatan terstruktur, sehingga subsistem *legacy* yang akan dianalisis terdiri dari Diagram Konteks, DFD level 1, dan DFD level 2. Pada diagram konteks, kondisi subsistem digambarkan secara keseluruhan, sehingga memudahkan dalam proses analisis. Kemudian, dilanjutkan pada DFD Level 1 yang menampilkan proses-proses yang terlibat secara umum. Proses-proses tersebut tentunya memiliki subproses-subproses. Subproses-subproses ini ditampilkan dalam bentuk yang lebih detail pada DFD Level 2. Berikut adalah penjelasan lebih detail dari analisis subsistem *Web Legacy*.

Diagram Konteks Sub-Sistem *Web Legacy*

Diagram konteks merupakan diagram yang menggambarkan secara umum *input*, *process*, dan *output* yang terjadi dalam sebuah sistem. Berikut ini gambaran setiap proses-proses yang ada di sub-sistem *Web Legacy*.

Diagram konteks mengggambarkan bahwa *legacy* subsistem *web* terdiri dari dua jenis pengguna, yaitu administrator dan petugas, selain itu ia juga menyediakan proses untuk digunakan oleh *legacy* subsistem *mobile*. Proses-proses yang dimiliki oleh *legacy* subsistem ini dapat dilihat lebih detail pada Gambar 3.

DFD Level 1 Subsistem *Web Legacy*

Gambar 3 menunjukan bahwa subsistem *Web Legacy* memiliki sembilan proses. Proses-proses tersebut dibagi untuk dua jenis pengguna, pertama untuk pengguna dengan hak akses administrator (admin) dan kedua untuk pengguna dengan hak akses petugas. Pengguna dengan hak akses admin dapat melakukan 8 proses, yaitu dari proses 1 sampai proses 8, sedangkan pengguna dengan hak akses petugas hanya dapat melakukan tiga proses yaitu login, melakukan pengolahan data laporan kejahatan dan melakukan pengolahan data kegiatan polrestabes.

Dari kesembilan proses tersebut, dilakukan analisis lebih dalam dengan menganalisis DFD level 2 dari setiap proses DFD level 1. Setelah diperoleh proses-proses yang dimiliki dari DFD level 2, maka kode pada setiap proses dianalisis untuk mendapatkan prosedur dan fungsi apa saja yang terlibat. Proses dan prosedur tersebut dapat dilihat pada Tabel 1.

Analisis Kebutuhan Kelas Sistem Baru

Pada tahap sebelumnya, prosedur dan fungsi yang terlibat telah didefinisikan dalam bentuk tabel. Selanjutnya, dari prosedur-prosedur tersebut akan dibentuk kelas yang dibutuhkan oleh sistem baru.

Kelas PetugasControl
Kelas ini terbentuk dari penggabungan *DFD Level 2 Proses Login* dan *Proses Pengolahan Data Petugas*. Kelas ini dibentuk dengan tujuan untuk memberikan autorisasi terhadap pengguna yang tidak berhak atas subsistem *web*, dan juga untuk melakukan pengolahan data petugas. Inilah alasan kenapa kedua proses tersebut digabung.

Kelas PenggunaControl
Kelas ini terbentuk dari penggabungan *DFD Level 2 Proses Pengolahan Data Pengguna* dan *Proses Penyedian Data Login*. Kelas ini dibentuk dengan tujuan untuk mengelola data pengguna subsistem *web*, dan menyediakan data yang diperlukan pengguna subsistem *mobile*.

Kelas LaporanControl
Kelas ini terbentuk dari penggabungan *DFD Level 2 Proses Pengolahan Data Laporan Kejahatan* dan *Proses Penyediaan Data Laporan*. Kelas ini dibentuk dengan tujuan untuk mengelola data laporan kejahatan subsistem *web*, dan menyediakan data laporan kejahatan subsistem *mobile*.

Kelas KegiatanContol
Kelas ini terbentuk dari penggabungan *DFD Level 2 Proses Pengolahan Data Kegiatan Polrestabes* dan *Proses Penyediaan Data Kegiatan Polrestabes*. Kelas ini dibentuk dengan tujuan untuk mengelola data kegiatan polrestabes subsistem *web*, dan menyediakan data kegiatan polrestabes subsistem *mobile*.

Kelas JenisControl
Kelas ini terbentuk dari penggabungan *DFD Level 2 Proses Pengolahan Data Jenis Kejahatan* dan *Proses Penyediaan Data Jenis Kejahatan*. Kelas ini dibentuk dengan tujuan untuk mengelola data jenis kejahatan subsistem *web*, dan menyediakan data jenis kejahatan subsistem *mobile*.

Kelas LokasiControl
Kelas ini terbentuk dari penggabungan *DFD Level 2 Proses Pengolahan Data Lokasi Kejahatan* dan *Proses Penyediaan Data Lokasi Kejahatan*. Kelas ini dibentuk dengan tujuan untuk mengelola data lokasi kejahatan subsistem *web*, dan menyediakan data lokasi kejahatan subsistem *mobile*.

Kelas WilayahControl
Kelas ini terbentuk dari *DFD Level 2 Proses Pengolahan Data Wilayah Kepolisian*. Kelas ini dibentuk dengan tujuan hanya untuk mengelola data wilayah kepolisian subsistem *web*.

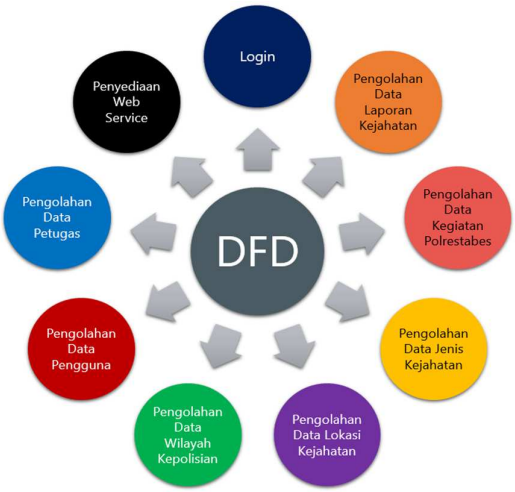
Selanjutnya, dikarenakan sistem baru menggunakan konsep MVC (*Models-Views-Control-*

TABEL 1
PROSEDUR YANG TERLIBAT

Asal Proses	Asal Prosedur
Login	login_admin login_petugas lupa_password
Pengolahan Data Petugas	input_petugas ubah_petugas view_petugas delete_petugas
Pengolahan Data Pengguna	view pengguna delete_pengguna
Penyedian Web Service Proses 9.1	bind_user_data post_user_data
Pengolahan Data Laporan Kejahatan	input_laporan_kejahatan_admin view_laporan_kejahatan_admin update_laporan_kejahatan_admin delete_laporan_kejahatan_admin input_laporan_kejahatan_petugas view_laporan_kejahatan_petugas update_laporan_kejahatan_petugas delete_laporan_kejahatan_petugas dateTime ParserID
Penyediaan Web Service Proses 9.2	bind_laporan_kejahatan post_laporan_kejahatan update_laporan_kejahatan
Pengolahan Data Kegiatan Polrestabes	input_kegiatan_polrestabes_admin view_kegiatan_polrestabes_admin ubah_kegiatan_polrestabes_admin delete_kegiatan_polrestabes_admin input_kegiatan_polrestabes_petugas view_kegiatan_polrestabes_petugas ubah_kegiatan_polrestabes_petugas delete_kegiatan_polrestabes_petugas
Penyediaan Web Service Proses 9.5	bind_kegiatan_polrestabes
Pengolahan Data Jenis Kejahatan	input_jenis_kejahatan view_jenis_kejahatan ubah_jenis_kejahatan delete_jenis_kejahatan
Penyediaan Web Service Proses 9.4	bind_jenis_kejahatan
Pengolahan Data Lokasi Kejahatan	input_lokasi_kejahatan view_lokasi_kejahatan ubah_lokasi_kejahatan delete_lokasi_kejahatan
Penyediaan Web Service Proses 9.3	bind_lokasi_kejahatan
Pengolahan Data Wilayah Kepolisian	input_wilayah_kepolisian view_wilayah_kepolisian ubah_wilayah_kepolisian delete_wilayah_kepolisian

TABEL 2
KELAS MVC

No	models	Nama Kelas services	controllers
1	Petugas	PetugasService	PetugasControl
2	Penggun a	PenggunaService	PenggunaControl
3	Laporan	LaporanService	LaporanControl
4	Kegiatan	KegiatanService	KegiatanControl
5	Jenis	JenisService	JenisControl
6	Lokasi	LokasiService	LokasiControl
7	Wilayah	WilayahService	WilayahControl



Gambar 3 DFD Level 1 Subsistem *Web Legacy*

Gambar 4 Pembentukan Kelas Sistem Baru

lers), maka kelas pada *package controllers* tersebut membutuhkan kelas lainnya, dalam analisis ini adalah kelas *models* yang disimpan di *package models* dan kelas *services* yang disimpan di *pack-age services* yang dapat dilihat pada Tabel 2.

Tabel 2 menunjukan bahwa untuk menerapkan kelas PetugasControl, ia memerlukan kelas Petugas dan kelas PetugasService. Kelas Petugas diperlukan untuk memodelkan *database* petugas dan kelas PetugasService diperlukan sebagai penyedia *metode* CRUD untuk *controllers*. Hal ini juga berlaku untuk kelas-kelas lainnya.

Analisis *Concurrency* dan *Asynchronous*

Pada tahapan ini, akan dijelaskan mengenai kelas-kelas apa saja yang mengalami penerapan konsep *concurrency* dan *asynchronous* pada kelas-kelas yang terbentuk sebelumnya. Analisis penerapan ini ditandai dengan (√) yang dapat dilihat pada Tabel 3 yang menampilkan semua kelas yang ter-bentuk dari tahapan sebelumnya.

TABEL 3
KELAS ASYNC DAN CONCURRENT

Nama Package	Nama Kelas	Async	Concurrent
controllers	PetugasControl	√	√
	PenggunaControl	√	√
	LaporanControl	√	√
	KegiatanControl	√	√
	JenisControl	√	√
	LokasiControl	√	√
	WilayahControl	√	√
services	PetugasService	√	√
	PenggunaService	√	√
	LaporanService	√	√
	KegiatanService	√	√
	JenisService	√	√
	LokasiService	√	√
	WilayahService	√	√
models	Petugas	-	-
	Pengguna	-	-
	Laporan	-	-
	Kegiatan	-	-
	Jenis	-	-
	Lokasi	-	-
	Wilayah	-	-

Analisis Data

Analisis data yang digunakan pada sistem baru menggunakan FRM (*Functional Relational Mapping*). Beberapa konsep FRM menyerupai dengan konsep ORM, tetapi juga memiliki perbedaan yang signifikan. Perbedaannya, daripada menjembatani jarak *object model* dengan *object database*, FRM membawa *database model* ke dalam Scala, sehingga *developer* tidak perlu menulis kode SQL.

Pemetaan yang dilakukan oleh FRM akan membutuhkan suatu jembatan berupa format data JSON yang menghubungkan FRM dengan *data-*

TABEL 4
KELAS UJI PENGUJIAN UNIT

Kelas Uji	Jenis Pengujian
JenisKejahatan	*Unit Testing*
LokasiKejahatan	*Unit Testing*
WilayahKepolisian	*Unit Testing*
LaporanKejahatan	*Unit Testing*
KegiatanPolrestabes	*Unit Testing*
Petugas	*Unit Testing*
Pengguna	*Unit Testing*

TABEL 5
ROUTE UJI PENGUJIAN INTEGRASI

Route Uji	Jenis Pengujian
JenisKejahatan	*Integration Testing*
LokasiKejahatan	*Integration Testing*
WilayahKepolisian	*Integration Testing*
LaporanKejahatan	*Integration Testing*
KegiatanPolrestabes	*Integration Testing*
Petugas	*Integration Testing*
Pengguna	*Integration Testing*

TABEL 6
FORMAT DATA JSON

No	Nama	JSON	Deskripsi	Keterangan
1	Data array	{ result: "OK", response: [{ "key": "value", "key": "value" }] }	Format JSON ini digunakan ketika mengembalikan data yang berupa data array.	1. key: *field database* 2. value: nilai dari *field* 3. result: report proses pengolahan data 4. response: menandakan data yang akan dikembalikan berupa array
2	Data null	{ result: "NOK", resposnse: null, error: { status: 404, message: "message" } }	Format JSON ini digunakan ketika mengembalikan data yang berupa data null.	5. message: pesan error

base yang ada pada *server*. Struktur format data JSON yang digunakan berupa data array dan null. Berikut adalah format data JSON yang dapat lihat pada Tabel 6.

Hasil Perbandingan Kode

Hasil perbandingan kode dapat dilihat pada Gambar 5 yaitu kode pada sistem Legacy dengan Gambar 6 yang menunjukkan kode dalam sistem baru.

Pengujian Unit dan Integrasi

Kelas uji dan *route* uji yang digunakan pada pengujian unit dan pengujian integrasi dapat dilihat pada Tabel 4 dan 5.

Pengujian unit yang akan dilakukan, ditujukan untuk controllers sistem baru yang terdiri dari 7 kelas. Dari setiap kelas tersebut kemudian akan menguji setiap metode yang ada di kelas tersebut. Selanjutnya, pengujin integrasi ditujukan untuk *route* sistem baru yang terdiri dari 8 *route*. Dari setiap *route* tersebut kemudian akan menguji setiap metode yang ada di *route* tersebut. Tujuannya untuk memeriksa integrasi dengan *database*.

Setelah pengujian unit dan pengujian integrasi dilakukan, maka diperoleh hasil bahwa pada pengujian unit, setiap metode yang diuji telah bekerja sesuai dengan tugas yang diharapkan.

```
<?php
ob_start();
include 'koneksi.php';

if(isset($_POST['kirim']))
{
   $petugas=$_SESSION['id_petugas'];
    mysql_query("INSERT INTO tmp_pelapor (id_pelapor)
                                VALUES ('$_SESSION[id_petugas]')");

    mysql_query("INSERT INTO kegiatan_polrestabes (
                                    judul_kegiatan,
                                    deskripsi_kegiatan,
                                    tanggal_kegiatan,
                                    waktu_kegiatan,
                                    alamat_kegiatan
                                    )
                                VALUES (
                                    '$_POST[judul_kegiatan]',
                                    '$_POST[deskripsi_kegiatan]',
                                    '$_POST[tanggal_kegiatan]',
                                    '$_POST[waktu_kegiatan]',
                                    '$_POST[alamat_kegiatan]')");

    header('location:viewkegiatanpolrestabes.php');
}
?>
```

Gambar 5 Kode Pada Sistem *Legacy*

TABEL7
FUNGSIONAL UJI PENGUJIAN PERFORMA

Fungsional Uji	Jenis Pengujian
Input data laporan kejahatan	*Performance Testing*
View data laporan kejahatan	*Performance Testing*
Edit data laporan kejahatan	*Performance Testing*
Delete data laporan kejahatan	*Performance Testing*
Input data kegiatan polrestabes	*Performance Testing*
View data kegiatan polrestabes	*Performance Testing*
Edit data kegiatan polrestabes	*Performance Testing*
Delete data kegiatan polrestabes	*Performance Testing*
Input data jenis kejahatan	*Performance Testing*
View data jenis kejahatan	*Performance Testing*
Edit data jenis kejahatan	*Performance Testing*
Delete data jenis kejahatan	*Performance Testing*
Input data lokasi kejahatan	*Performance Testing*
View data lokasi kejahatan	*Performance Testing*
Edit data lokasi kejahatan	*Performance Testing*
Delete data lokasi kejahatan	*Performance Testing*
Input data wilayah kepolisian	*Performance Testing*
View data wilayah kepolisian	*Performance Testing*
Edit data wilayah kepolisian	*Performance Testing*
Delete data wilayah kepolisian	*Performance Testing*
View data pengguna	*Performance Testing*
Input data petugas	*Performance Testing*
View data petugas	*Performance Testing*
Edit data petugas	*Performance Testing*
Delete data petugas	*Performance Testing*

Sementara itu, pada pengujian pengujian integrasi, setiap *route* yang diuji telah berintegrasi dengan baik dengan *database*.

Pengujian Performa

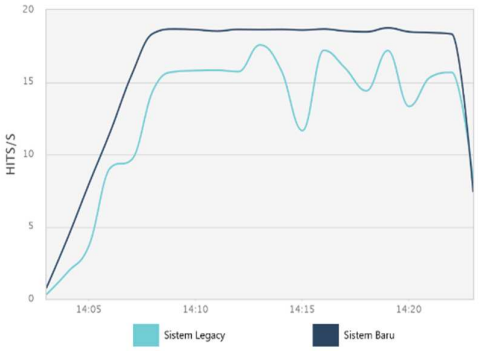
Fungsional uji yang digunakan pada pengujian performa dapat dilihat pada Tabel 7. Setelah fungsional uji didefinisikan, maka langkah selanjutnya yaitu melakukan pengujian performa dengan menggunakan Blazemeter. Blazemeter merupakan salah satu *platform* pengujian performa berbasis *cloud*. Setelah pengujian selesai dilakukan, maka diperolah hasil waktu respon dan nilai *throughput* antara sistem lama dengan sistem baru.

Berdasarkan Gambar 7 dan Gambar 8 dapat disimpulkan bahwa hasil waktu respon dari *backend* sistem baru lebih cepat dibandingkan dengan sistem lama yaitu sebesar 100.38 ms dan unit kerja yang dapat ditangani lebih banyak, yaitu 1.78 Hits/s.

```
    def add = Action.async { implicit request =>
      val body = models.Kegiatan.formKegiatan.bindFromRequest()
      body.fold(e => {
        val response = ErrorResponse(BAD_REQUEST, "invalid json")
        Future(BadRequest(Json.toJson(response)))
      }, { newKegiatan =>
        val future = kegiatanService.addKegiatan(newKegiatan)
        future.flatMap {
          case Some(ok) => Future.successful(
            Created(Json.toJson(SuccessResponse(ok))))
          case None => Future(
            BadRequest(Json.toJson(ErrorResponse(BAD_REQUEST, "add
kegiatan fail"))))
        }
      })
    }
```

Gambar 6 Kode Pada Sistem Baru

Gambar 7 Perbandingan Waktu Respon



Gambar 8 Perbandingan Nilai Throughput

4. Kesimpulan

Penelitian yang bertujuan untuk meningkatkan performa dari sistem backend Crimezone telah dilakukan. Hal ini dicapai dengan memigrasikan bahasa pemrograman yang digunakan ke bahasa pemrograman yang baru. Berikut adalah beberapa kesimpulan yang dapat diambil dari penelitian ini yaitu:

1) Dari setiap fungsionalitas sistem baru yang diuji dengan pengujian unit dan pengujian integrasi dapat disimpulkan bahwa fungsi-fungsinya sudah berjalan sesuai dengan tugasnya masing-masing.
2) Pengujian performa antara sistem lama dengan sistem baru menghasilkan kesimpulan bahwa performa sistem baru berjalan lebih cepat dengan melayani akses data yang dibutuhkan oleh pengguna.
3) Memigrasikan *backend* Crimezone ke Scala dapat memudahkan penerapan konsep pemrograman *asynchronous*, *concurrency*, dan *parallelism*.

Referensi

[1] M. Odersky, L. Spoon and B. Venners, Programming in Scala, California: Artima Press is an imprint of Artima, Inc., 2010.

[2] Dwarampudi, Venkatreddy; Dhillon, Shahbaz Singh; Shah, Jivitesh; Sebastian, Nikhil Joseph; Kanigicharla, Nitin Satyanarayan; "Comparative study of the Pros and Cons of Programming languages Java, Scala, C++, Haskell, VB .NET, AspectJ, Perl, Ruby, PHP & Scheme," 2010.

[3] thomas, "thomasknierim," [Online]. Available: http://www.thomasknierim.com/119/java/performance-java-vs-php-vs-scala/. [Accessed Monday August 2015].

[4] N. Raychaudhuri, Scala in Action, Shelter Island: Manning, 2013.

[5] E. Ackerman, J. Ebert and A. Winter, "Ein Referenz-Prozessmodell zur Software-Migration," 2005.