ABSTRAK

Pengembangan perangkat lunak sekarang ini sudah semakin kompleks. Umumnya masalah yang terjadi pada pengembangan perangkat lunak terdapat pada saat *developer* akan melakukan integrasi kode program perangkat lunak. Penggunaan *Continuous Integration* merupakan salah satu cara yang mampu digunakan untuk mengatasi masalah dari integrasi kode perangkat lunak dan *testing* yang terjadi, sehingga *Continuous Integration* dapat mengurangi waktu tempuh yang digunakan untuk menghasilkan suatu perangkat lunak [1]. *Continuous Integration* adalah proses mengotomatisasi pembuatan dan pengujian kode setiap kali anggota tim melakukan perubahan pada kontrol versi. *Continuous Integration* mendorong pengembang untuk membagikan kode dan unit test dengan menggabungkan perubahan ke dalam repositori kontrol versi bersama setelah melakukan perubahan pada kode program [2]. Banyak *CI Tools* yang dapat membantu pengguna untuk menerapkan *Continuous Integration* dalam proses pengembangan perangkat lunak, salah satu contohnya *TeamCity*. *TeamCity* pun memberikan kemudahan dalam konfigurasi alat *CI Server*, pembuatan *build step*, bahkan memberikan statistik yang dapat digunakan untuk menentukan kualitas dari perangkat lunak [3]. Untuk konfigurasi optimum pada *TeamCity* saat ini dapat dipilih berdasarkan bahasa pemrograman yang digunakan pada *project*, yang dimana pada penelitian ini digunakan bahasa pemrograman *C#* karena pada *TeamCity* sendiri tidak memiliki runner yang dikhususkan untuk platform *desktop* ataupun *web*. Sehingga *build configuration* yang digunakan pada penelitan ini sama untuk dua *project* yang digunakan.

Kata kunci: *Continuous Integration*, *CI Tools*, *CI Server*, Perangkat Lunak, Statistik, *TeamCity*.

Universitas Kristen Maranatha

ABSTRACT

Software development is now increasingly complex. Generally the problems that occur in software development are when developer going to integrate software program code. The use of Continuous Integration is one of the ways that can overcome the problem of integration of software code and testing that occurs, so that it can reduce the time used to produce a software [1]. Continuous Integration is the process of automating the creation and testing of code every time a team member changes the version control. Continuous Integration encourages developers to share code and unit tests by combining changes into the shared version control repository after each changes on code program [2]. Many CI Tools can help users to implement Continuous Integration in the software development process, one example is TeamCity. TeamCity also provides ease of configuration of CI Server, make the build steps, even provide statistical metrics that can be used to determine the quality of the software [3]. The optimum configuration for TeamCity can now be selected based on the programming language used in the project, which in this study used the C # programming language because TeamCity itself does not have runners specifically for desktop or web platforms. So that the build configuration used in this research is the same for the two projects used.

Keywords: Continuous Integration, CI Tools, Software Development, Statistic, TeamCity

DAFTAR ISI

LEMBAR PENGESAHAN i

PERNYATAAN ORISINALITAS LAPORAN PENELITIAN ii

PERNYATAAN PUBLIKASI LAPORAN PENELITIAN iii

PRAKATA iv

ABSTRAK vi

ABSTRACT vii

DAFTAR ISI viii

DAFTAR GAMBAR xiii

DAFTAR TABEL xiv

DAFTAR NOTASI/ LAMBANG xv

BAB 1 PENDAHULUAN 1

1.1 Latar Belakang 1

1.2 Rumusan Masalah 2

1.3 Tujuan Pembahasan 2

1.4 Ruang Lingkup 2

1.5 Sumber Data 2

1.6 Sistematika Penyajian 3

BAB 2 KAJIAN TEORI 4

2.1 DevOps 4

2.2 Continuous Integration 5

2.2.1 Best Practice 5

2.2.1.1 Single Source Repository 6

2.2.1.2 Automate the Build 6

2.2.1.3 Make Your Build Self-Testing 6

2.2.1.4 Everyone Commit to the Mainline Every Day 6

2.2.1.5 Every Commit Should Build the Mainline 6

2.2.1.6 Fix Broken Builds Immediately 7

2.2.1.7 Keep the Build Fast 7

2.2.1.8 Test in a Clone of the Production Environment 7

2.2.1.9 Make it Easy for Anyone to Get the Latest Executable 7

2.2.1.10 Everyone can See what's Happening 7

2.2.2 Continuous Integration Tools 8

2.2.2.1 Hosting Requirement 8

2.2.2.2 Budget 8

2.2.2.3 Features 8

2.2.2.4 Community Support 9

2.2.2.5 Return on Investment 9

2.3 TeamCity 9

2.3.1 Basic Concept 9

2.3.2 Basic Worklfow 10

2.3.3 REST API TeamCity 11

2.3.4 Metrik dalam *TeamCity* 12

BAB 3 ANALISIS DAN RANCANGAN SISTEM 13

3.1 Kekurangan pada *TeamCity* 13

3.2 Proses Desain Konfigurasi *Build TeamCity* 13

3.2.1 Atribut yang Didapatkan 14

3.3 Use Case Diagram 14

3.3.1 Use Case Skenario 15

3.3.1.1 Use Case Skenario Melihat Data *Build* 15

3.3.1.2 Use Case Skenario Melihat Hasil *Build* 15

Universitas Kristen Maranatha

3.3.1.3 Use Case Skenario *Input* Saran 16

3.4 Activity Diagram 16

3.4.1 Activity Diagram Melihat Data *Build* 16

3.4.2 Activity Diagram Melihat Hasil *Build* 17

3.4.3 Activity Diagram Input Saran 18

3.5 Class Diagram 19

3.5.1 BaseBuild 19

3.5.2 TeamCityBuildList 19

3.5.3 BuildStatisticList 20

3.5.4 StatisticValue 20

3.5.5 BuildDataService 20

3.5.6 StatisticDataService 21

BAB 4 IMPLEMENTASI 22

4.1 Konfigurasi *TeamCity* 22

4.1.1 VCS Root 22

4.1.2 Build Configuration 23

4.1.3 Trigger 23

4.2 Implementasi *Class Diagram* 24

4.2.1 Implementasi *class BaseBuild* 24

4.2.2 Implementasi *class TeamCityBuildList* 24

4.2.3 Implementasi *class StatisticValue* 25

4.2.4 Implementasi *class BuildStatisticList* 25

4.2.5 Implementasi *class BuildDataService* 25

4.2.6 Implementasi *class StatisticDataService* 26

4.3 API Source Code 26

4.3.1 API Get Build Data 26

Universitas Kristen Maranatha

4.3.2 API Get Build Statistic Data 27
4.4 Data yang Didapatkan 28
4.4.1 Artifact Size 28
4.4.2 Build Duration 28
4.5 Aplikasi 29
BAB 5 PENGUJIAN 31
5.1 *Project* yang digunakan 31
5.2 *Parallel Build* 31
5.3 Build Step 32
5.3.1 Project WebMVC 32
5.3.2 Project WinFormMVP 32
5.4 Statistik yang didapat setelah *build* 33
5.4.1 Project WebMVC 33
5.4.1.1 Statistik *Artifact Size* 33
5.4.1.2 Statistik *Build Duration* 34
5.4.2 Project WinFormMVP 34
5.4.2.1 Statistik *Artifact Size* 34
5.4.2.2 Statistik *Build Duration* 35
5.4.3 Kesimpulan dari statistik yang didapat 35
5.4.3.1 Artifact Size 35
5.4.3.2 Build Duration 35
BAB 6 SIMPULAN DAN SARAN 36
6.1 Simpulan 36
6.2 Saran 36
DAFTAR PUSTAKA 38
LAMPIRAN A Posisi Aplikasi Pada *Workflow TeamCity* A-1

Universitas Kristen Maranatha

LAMPIRAN B Kode Program *WebMVC* B-1

LAMPIRAN C Kode Program *WinFormMVP* C-1

LAMPIRAN D *Build Step* D-1

LAMPIRAN E Tampilan Aplikasi E-1

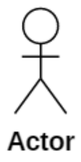
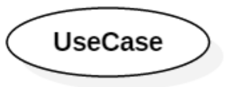
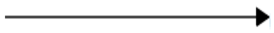
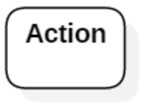
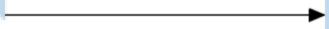
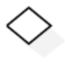
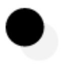
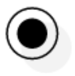
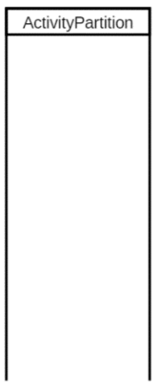
DAFTAR GAMBAR

Gambar 2.1 Siklus *DevOps* 5
Gambar 2.2 *TeamCity Workflow* 11
Gambar 3.1 Proses Konfigurasi *TeamCity* 14
Gambar 3.2 *Use Case Diagram* 15
Gambar 3.3 *Activity Diagram* Melihat data *Build* 17
Gambar 3.4 *Activity Diagram* Melihat Hasil *Build* 18
Gambar 3.5 *Activity Diagram Input* Saran 18
Gambar 3.6 *Class Diagram BaseBuild* 19
Gambar 3.7 *Class Diagram TeamCityBuildList* 19
Gambar 3.8 *Class Diagram BuildStatisticList* 20
Gambar 3.9 *Class Diagram StatisticValue* 20
Gambar 3.10 *Class Diagram BuildDataService* 21
Gambar 3.11 *Class Diagram StatisticDataService* 21
Gambar 4.1 *Create Project* 22
Gambar 4.2 *Create Project Selesai* 22
Gambar 4.3 Tampilan *Build Configuration* 23
Gambar 4.4 *Setting Build Step* 23
Gambar 4.5 *Setting Trigger* 24
Gambar 4.6 *Source Code Get Build Data* 27
Gambar 4.7 *Source Code Get Build Statistic Data* 28
Gambar 4.8 Tampilan Aplikasi 29
Gambar 4.9 Contoh Grafik 30
Gambar 4.10 Posisi Aplikasi Pada *WorkFlow TeamCity* 30
Gambar 5.1 *Build Step WebMVC* 32
Gambar 5.2 *Build Step WinFormMVP* 33
Gambar 5.3 Statistik *Artifact Size* 34
Gambar 5.4 Statistik *Build Duration* 34
Gambar 5.5 Statistik *Artifact Size* 34
Gambar 5.6 Statistik *Build Duration* 35

DAFTAR TABEL

Tabel 3.1 Atribut Hasil *Build* 14
Tabel 3.2 *Use Case* Skenario Melihat Data *Build* 15
Tabel 3.3 *Use Case* Skenario Melihat Hasil *Build* 15
Tabel 3.4 *Use Case* Skenario *Input* Saran 16
Tabel 4.1 Tabel Implementasi *class BaseBuild* 24
Tabel 4.2 Tabel Implementasi *class TeamCityBuildList* 25
Tabel 4.3 Tabel Implementasi *class StatisticValue* 25
Tabel 4.4 Tabel Implementasi *class BuildStatisticList* 25
Tabel 4.5 Tabel Implementasi *class BuildDataService* 25
Tabel 4.6 Tabel Implementasi *class StatisicDataService* 26

DAFTAR NOTASI/ LAMBANG

Jenis	Notasi/ Lambang	Nama	Arti
UML (*Use Case)*		*Actor*	Pengguna sebuah sistem.
		*Use Case*	Entitas yang berhubungan dengan sistem dan pengguna.
		*Directed Association*	Hubungan satu arah antara dua buah objek.
UML (*Activity Diagram)*		*Action*	Aktivitas yang dilakukan oleh pengguna.
		*Control Flow*	Menunjukkan arah suatu proses.
		*Decision*	Sebuah pilihan yang dapat dipilih pengguna.
		*Initial*	Menandakan mulainya sebuah aktivitas.
		*Final*	Menandakan berakhirnya sebuah aktivitas.
		*Swimlane*	Pemisahan antara lingkungan sebuah sistem bekerja.

Referensi:

Notasi/ Lambang UML dari *The Unified Modeling Language Reference Manual* [4] Notasi/ Lambang ERD dari *Design Database Diagrams (Visual Database Tools)* [5]