# Development B

# Continuous Delivery

# Opdracht C: Release-documentation output-subsystem

By Robin Bakker en Robert Kraaijeveld, INF2C

Table of contents

[Stakeholders 3](#_Toc440297966)

[Asset and configuration management 4](#_Toc440297967)

[Deployment technology 5](#_Toc440297968)

# Stakeholders

Within this project, the following key stakeholders can be identified:

- The CityGis company: CityGis has a stake in this project because (within the ‘scenario’) they commissioned and provide the resources for this project. They want their data to be presented to the users and other stakeholders in a way that will reflect the quality of their work.

- The Car maintainer: The vehicles that use CityGis-equipment are maintained by a maintainer. This company/person would obviously like to know which cars have defects or exhibit strange behavior as soon as possible, since vehicles for emergency services are very scarce.

- The data-analyst/Commercial-user: The data-analyst or commercial user, from here on referred to as ‘the user’ wants to have the clearest and most complete (over)view of the CityGis data as possible to get his/hers moneys’ worth.

# Asset and configuration management

Our Project56 output-subsystem has been programmed in PHP5, combined with the usage of the Laravel-framework. Below you will find all the assets, dependencies and libraries being used within our output subsystem together with their version numbers.

**Dependencies necessary for running and building:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Version number** | **Short description** | **Preferred download method** |
| PHP programming language | 5.6.16 | PHP server-side programming language. | Archive download (<http://php.net/downloads.php>) or Aptitude install on Linux. |
| Composer | 1.0-dev | PHP package manager. | Curl-install script (https://getcomposer.org/download) |
| The Laravel Framework | 5.1.26 | MVC-style framework for PHP. | Composer:  ‘composer global require "laravel/installer=~1.1" ‘ |
| PHPunit\* | 5.1.3 | PHP unit-testing framework. | Composer:  ‘composer global require "phpunit/phpunit=5.1.\*" ’ |
| Apache Ant | 1.9.3 | Building engine. | Archive download (<https://ant.apache.org/bindownload.cgi>) or Aptitude install on Linux. |
| PHPDox\* | 0.8.0 | PHP XML documentation generator. | Phar file: ‘https://github.com/theseer/phpdox ‘ |
| PHPloc\* | 2.5.1 | PHP static code analysis interface. | Composer  ‘composer global require 'phploc/phploc=\*' ‘ |

\*= These dependencies are automatically downloaded in the assignment B bash script.

# Deployment technology and implementation

For Project56 we use the Jenkins continuous integration system to build and deliver our Output-subsystem. We use Jenkins version 1.642.

*Note that some of the build steps may not work completely when not using a Linux-style OS.*

Our Jenkins Job has the following steps:

**1. Build: (Triggered every hour, or by a git push to the master branch.)**

**-** Run a shell script:

cd /

cd /home/yourUserName/yourPreferredProject56RepoLocation/

sudo ./yourAssignmentBstyleScript.sh

This shell script clones the Project56 git repository if it is not already present. It then proceeds to download several PHP-phar archives (containing the dependencies marked with an asterisk in the last chapter.) and unpack them in the usr/bin/ directory, so these dependencies can be used directly via their name.

It then proceeds to call ant using the build.xml file located in yourPreferredProject56RepoLocation/http/Laravel. This build file is too long to fully disclose in this document, but it basically does the following things:

- It runs PHPLoc on all the php files within yourPreferredProject56RepoLocation/http/Laravel. PHPLoc produces human readable output that lists a static code analysis of all the PHP source code it finds. PHPLoc lists mostly simple statistics like the amount of methods and classes etc.

- It runs PHPUnit, and PHPUnit runs 2 simple unit-tests. The result of these tests is displayed in the console output.

- PHPDox takes the output of PHPLoc (and creates some other static analysis’ as well) and saves the output in an index.xml file in the Laravel/build/api directory.

**2. Post-build:**

- After the build steps above are complete, we publish the index.xml file that PHPDox generated.