MASARYK UNIVERSITY
FACULTY OF INFORMATICS

Refactoring of Kdyby packages

BACHELOR'S THESIS

Filip Procházka

Brno, Spring 2017

Masaryk University
Faculty of Informatics

Refactoring of Kdyby packages

Bachelor's Thesis

Filip Procházka

Brno, Spring 2017

This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Filip Procházka

Advisor: RNDr. Jaroslav Bayer

Acknowledgement

This is the acknowledgement for my thesis, which can span multiple paragraphs.

Abstract

This is the abstract of my thesis, which can
span multiple paragraphs.

Keywords

Contents

List of Tables

List of Figures

1 Introduction

Kdyby is an Open-source software (OSS) [1] project that I, Filip Procházka, lead and maintain. It is a set of PHP [2] libraries, that aim to ease writing of web applications.

Through my carer, I have used Kdyby in core business applications of companies such as Damejidlo.cz and Rohlik.cz. A lot of people consider my work useful enough to incorporate it to their own applications as well.

As of writing this, the more popular libraries have hundreds of thousands of downloads. Five of Kdyby libraries have over quarter million downloads and one is approaching half a million with staggering amount of 470 thousands of downloads [3]. In conclusion, a sober estimate would be, that Kdyby libraries are used in hundreds of real production applications.

If I account only for the two biggest projects that I can confirm are using Kdyby packages, over a billion Czech crowns[1] has literary flowed through Kdyby. That is a big responsibility.

Over the years, I have had problems keeping up with the demand and the packages began to get obsolete. I wanna use this thesis as way to fix the situation.

I will review the state of each library and decide its future. Which means I will either deprecate it and provide the users a suggestion for a better alternative, or fix the problems and refactor the library.

---

1. Rohlik.cz loni prodal zboží za miliardu, letos chce konečně zisk
    http://tyinternety.cz/e-commerce/rohlik-cz-loni-dosahl-na-
    miliardovy-obrat-letos-chce-konecne-zisk/

2 Background for understanding Kdyby

2.1 A brief history of Kdyby

In 2006 I have started working on my own Content management system (CMS) [4]. A prototype was used in production on few websites I created. The oldest preserved version is archived on my Github[1]. It is a great learning material on how to not write a CMS.

Then the concept of OSS [1] was introduced to me and I have decided to start working on everything openly, under a free license [5]. Sadly, since then, no new working version of Kdyby CMS was ever released, because I have rewritten it from scratch exactly 10 times.

In 2012, I have decomposed the the emerging system into separate libraries that can be used more or less independently and have their own release cycle. This approach was preserved to this day.

2.2 Techniques and design patterns

2.2.1 Dependency Injection

Inversion of control is a design principle in which custom-written portions of a computer program receive the flow of control from a generic framework.

Dependency Injection (DI) is a technique whereby one object supplies the dependencies of another object. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern [6].

2.2.2 Aspect Oriented Programming

In computing, Aspect-oriented Programming (AOP) is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns. It does so by adding additional behavior to existing code (an advice) without modifying the code itself, instead separately specifying which code is modified via a pointcut specification, such as log all function calls when the name of the function

---

1. archived on my Github https://github.com/fprochazka/kdyby-cms-old

begins with 'set'. This allows behaviors that are not central to the business logic (such as logging) to be added to a program without cluttering the code core to the functionality [7].

2.2.3   Event Dispatcher

The Event Dispatcher is a pattern for writing modular code. It allows to create extension points in the library or application that another library or application can hook into and change or extend the behavior.

Typically, the extension points are called hooks or events and the new functionality is provided with objects called listeners.

2.3   Technologies used

2.3.1   Git and Github

Git is a Version Control System, that is decentralized and considered very fast [8]. Github is a collaboration platform for software development using Git.

Each project has a page on Github called a repository, that can be used to inspect the Git history, files and other metadata. On the project repository page, there are issues and pull requests. Pull requests are a way to ask the maintainer of the repository to incorporate provided code patch to the repository. It can be a bugfix or feature.

There are tools around pull requests that allow collaboration, a code review and discussion about the provided code, so that the maintainers can help the contributors to provide the best code possible.

Kdyby is hosted and developed on Github, with the help of several other maintainers and the community, that contributes bugfixes and features.

2.3.2   Continuous Integration

Continuous Integration (CI) is a practice of merging all developer working copies to a shared mainline several times a day, to prevent merging conflicts. [9] But now-days, the term has established to mean CI servers that run prepared task on the provided code.

In practice, it means that as the developer is working on a feature or bugfix, they push the work in progress code into a repository, the code is then picked up by a CI server that executes the tests, checks coding style and runs various other tasks to verify that the code was not broken.

When the work is finished and all the task on CI server completed with success, the code can be probably safely integrated, providing that the tests for new or changed functionality were added.

Some popular CI services are Travis CI[2], CircleCI[3] and GitLab CI[4]. Kdyby is using the Travis CI, that is free for OSS projects.

2.3.3 Nette Framework

Nette Framework is an OSS framework for creating web applications in PHP [10]. Nette is separated into many packages.

The Dependency Injection component nette/di[5] provides a Dependency Injection Container (DIC) that holds the services. The component also allows to prepare a preconfigured DIC, which is then compiled into a PHP class that contains optimized code for the service creation. This compiled DI Container class is cached for reuse. There is a concept (and a class) CompilerExtension that allows the developer to hook into the process of configuring and compiling of the DIC. All of Kdyby packages that are an integration of some other library or tool into Nette provide a CompilerExtension to make the installation easy.

2.3.4 Composer

Composer is a tool for dependency management [11] in PHP. It allows you to declare the libraries your project depends on and it will manage (install or update) them for you [12].

Packages are usually published using Github with metadata in a file named `composer.json`, that is written in JSON [13] format.

---

2. Travis CI https://travis-ci.org/
3. CircleCI https://circleci.com/
4. GitLab CI https://about.gitlab.com/features/gitlab-ci-cd/
5. nette/di https://github.com/nette/di

4

Composer is decentralized, but has a single main metadata repository Packagist[6]. It stores and provides all the package metadata like available versions and where to download them.

All Kdyby libraries are published as Composer packages on Packagist and installing them using the Composer is the only officially supported installation method.

2.3.5 OAuth 2

OAuth is a protocol for authentication and authorization that can be implemented into a web service. It is designed for secure exchange of user information, allowing third party websites to implement a login and registration process that simplifies these tasks for the user essentially allowing them to login or register to services through the OAuth 2 provider with two clicks.

Kdyby provides packages for integrating Nette Framework with OAuth 2 providers, such as Facebook[7], Google[8] and Github[9].

2.3.6 dibi

Dibi is a Database Abstraction Library for PHP. It supports a lot of significant databases: MySQL, PostgreSQL, SQLite, MS SQL, Oracle, Access and generic PDO and ODBC [14].

2.3.7 Doctrine 2 ORM

Doctrine 2 ORM is an Object-Relation Mapper (ORM) [15], which means it allows the programmer to create PHP classes called entities, that represent relational data in a database and are used to actually map the data from the database to the classes and back. In conclusion, it allows the programmer to write fully Object-oriented (OOP) [16] applications.

---

6. Packagist https://packagist.org/
7. Facebook https://developers.facebook.com/docs/facebook-login/manually-build-a-login-flow
8. Google https://developers.google.com/identity/protocols/OAuth2
9. Github https://developer.github.com/v3/oauth/

2.3.8 Symfony Framework

Symfony is a PHP web application framework and a set of reusable PHP components/libraries, similar to Nette [17].

2.3.9 Monolog

Monolog is a logging library that sends your logs to files, sockets, inboxes, databases and various web services. This library implements the PSR-3 [18] interface that you can type-hint against in your own libraries to keep a maximum of interoperability [19].

2.3.10 RabbitMQ

RabbitMQ is OSS message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover [20].

2.3.11 ElasticSearch

Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. It is the most popular enterprise search engine [21].

2.3.12 Redis

Redis is an in-memory database OSS project that is networked, in-memory, and stores keys with optional durability [22].

2.3.13 PhpStan

PHPStan focuses on finding errors in your code without actually running it. It catches whole classes of bugs even before you write tests for the code [23].

2.3.14 Nette\Tester

Nette\Tester is an unit testing [24] framework for the PHP [25].

3 Current state of Kdyby

To be able to lay out the roadmap, first we have to know the current state of each Kdyby package, the original purpose and the current requirements. We will only review those packages that actually made it to production and at least one usable version was released.

I have created few GitHub repositories as a reminder for me to start working on some other web application development problems. I did start to work on some of them, for example on DoctrineForms, but it was never "officially released". We will not discuss these incomplete packages in this thesis.

3.1 State of the project

The most relevant problem is the compatibility with new versions of the libraries, that Kdyby integrates. Current version of Nette Framework is 2.4 and the 3.0 is being developed, but some of Kdyby packages support only Nette Framework 2.2 or older.

The other problems appear only when you interact with the source code which is still really important for me as a maintainer and for the contributors. Also good maintainable code attracts more programmers to use it and contribute. But, there is no coding standard being enforced automatically on any package and no static analysis tool is checking the code. On the other hand almost all of the packages have unit and integration tests and linter checking the code for multiple versions of PHP.

Most of the packages are compatible with PHP 5.4, but PHP 5.4 had end of life at 2015-09-03[1] and is no longer supported by PHP developers.

The PHP 5.5 had end of life at 2016-06-21 and PHP 5.6 is currently in the phase of security fixes only and will have end of life at 2018-12-31. As you can see on the graph 3.1, everyone should be migrating to PHP 7.0 by now, but it is not always as easy as simply upgrading PHP interpreter. The developer has to consider what PHP versions the libraries support, upgrade them first and then migrate the application.

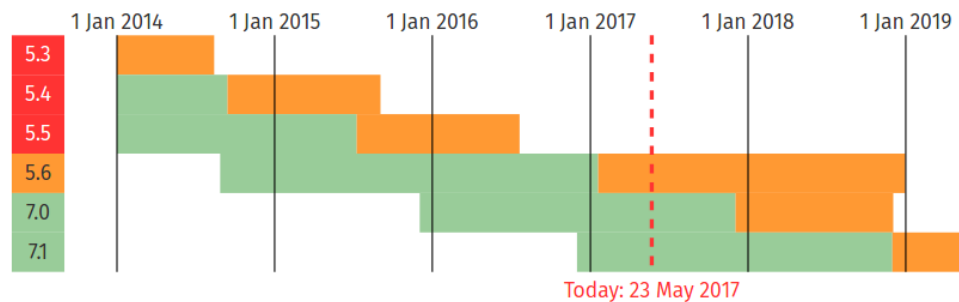---

1. PHP 5.4 had end of life at 2015-09-03 http://php.net/eol.php

Figure 3.1: Supported PHP Versions. Green is active support, orange is security fixes only. Up-to-date version is at http://php.net/supported-versions.php

3.2 State of each package

This section reviews each package separately, considers the original purpose and sums up the current state.

3.2.1 Doctrine

Kdyby\Doctrine is an integration of Doctrine 2 ORM into Nette Framework.

Doctrine 2 ORM itself is separated into several packages, mainly doctrine/orm[2], doctrine/common[3], doctrine/annotations[4], doctrine/cache[5] and doctrine/collections[6]. What started as a monolith integration in Kdyby, got separated into Kdyby\Events 3.2, Kdyby\Console 3.2, Kdyby\Annotations 3.2 and Kdyby\DoctrineCache 3.2 for reusability.

Over the years, it cumulated a lot of responsibilities, that don't belong to it. I have already started extracting few of them in the past, for example an entity prototyping tool 3.2, collection utilities 3.2, 3.2 and helper for loading big SQL scripts to the database 3.2.

---

2. doctrine/orm https://github.com/doctrine/doctrine2
3. doctrine/common https://github.com/doctrine/common
4. doctrine/annotations https://github.com/doctrine/annotations
5. doctrine/cache https://github.com/doctrine/cache
6. doctrine/collections https://github.com/doctrine/collections

There is a big issue Chop up the package[7] that discusses what other parts should be separated and dropped completely.

New versions of Nette and Doctrine 2 ORM were released and completely new versions are being prepared, which the integration cannot be currently used with.

3.2.2 Console

Kdyby\Console is an integration of Symfony Framework `Console` Component, that allows for writing interactive CLI applications. Kdyby\Doctrine 3.2 depends on this package and is the reason this package exists.

There are tasks, that are better suited for console interaction, than a web interface. Among others, Doctrine 2 ORM has tools for generating a database schema from the entities metadata and there is a console command for it, that is written using Symfony `Console`.

3.2.3 Events

Kdyby\Events provides an event dispatcher 2.2 implementation for Nette Framework.

It started as an integration of Doctrine 2 ORM `EventManager`, but then it evolved into a standalone system with support for lazy initialization of listeners and it also contains a naive bridge for Symfony Framework `EventDispatcher` Component.

Creating such interchangeable eventing system turned out to be a mistake, because it is a maintenance hell. The systems should have stayed separate.

3.2.4 Annotations

Kdyby\Annotations is a simple integration of doctrine/annotations into Nette Framework. It exists solely for the purposes of Kdyby\Doctrine.

---

7. Chop up the package https://github.com/Kdyby/Doctrine/issues/238

3.2.5 DoctrineCache

Lorem ipsum.

3.2.6 DoctrineMagicAccessors

Lorem ipsum.

3.2.7 DoctrineCollectionsReadonly

Lorem ipsum.

3.2.8 DoctrineCollectionsLazy

Lorem ipsum.

3.2.9 DoctrineDbalBatchImport

Lorem ipsum.

3.2.10 DoctrineForms

Lorem ipsum.

3.2.11 Curl

Lorem ipsum.

3.2.12 CurlCaBundle

Lorem ipsum.

3.2.13 Autowired

Lorem ipsum.

3.2.14 FormsReplicator

Lorem ipsum.

3.2.15 Translation

Lorem ipsum.

3.2.16 Validator

Lorem ipsum.

3.2.17 RabbitMq

Lorem ipsum.

3.2.18 Money

Lorem ipsum.

3.2.19 DoctrineMoney

Lorem ipsum.

3.2.20 Aop

Lorem ipsum.

3.2.21 Clock

Lorem ipsum.

3.2.22 Redis

Lorem ipsum.

3.2.23 ParseUseStatements

Lorem ipsum.

3.2.24 RedisActiveLock

Lorem ipsum.

3.2.25 TesterParallelStress

Lorem ipsum.

3.2.26 Monolog

Lorem ipsum.

3.2.27 ElasticSearch

Lorem ipsum.

3.2.28 DoctrineSearch

Lorem ipsum.

3.2.29 Geocoder

Lorem ipsum.

3.2.30 CsobPaygateNette

Lorem ipsum.

3.2.31 CsobPaymentGateway

Lorem ipsum.

3.2.32 Wkhtmltopdf

Lorem ipsum.

3.2.33 FakeSession

Lorem ipsum.

3.2.34 RequestStack

Lorem ipsum.

3.2.35 StrictObjects

Lorem ipsum.

3.2.36 Facebook

Lorem ipsum.

3.2.37 Google

Lorem ipsum.

3.2.38 Github

Lorem ipsum.

3.2.39 NettePhpServer

Lorem ipsum.

3.2.40 TesterExtras

Lorem ipsum.

3.2.41 HtmlValidatorPanel

Lorem ipsum.

3.2.42 BootstrapFormRenderer

Lorem ipsum.

3.2.43 PayPalExpress

Lorem ipsum.

3.2.44 PresentersLocator

Lorem ipsum.

3.2.45 SvgRenderer

Lorem ipsum.

3.2.46 QrEncode

Lorem ipsum.

4 Designing roadmap of refactoring

In this chapter, I am going to lay out the plan for the refactoring itself and set some specific goals for each package and for the project itself.

4.1   Common requirements

Lorem ipsum.

4.2   Roadmap for each package

4.2.1   Doctrine

Lorem ipsum.

4.2.2   Console

Lorem ipsum.

4.2.3   Events

Lorem ipsum.

4.2.4   Annotations

Lorem ipsum.

4.2.5   DoctrineCache

Lorem ipsum.

4.2.6   DoctrineMagicAccessors

Lorem ipsum.

4.2.7   DoctrineCollectionsReadonly

Lorem ipsum.

4.2.8 DoctrineCollectionsLazy

Lorem ipsum.

4.2.9 DoctrineDbalBatchImport

Lorem ipsum.

4.2.10 DoctrineForms

Lorem ipsum.

4.2.11 Autowired

Lorem ipsum.

4.2.12 FormsReplicator

Lorem ipsum.

4.2.13 Translation

Lorem ipsum.

4.2.14 Validator

Lorem ipsum.

4.2.15 RabbitMq

Lorem ipsum.

4.2.16 Money

Lorem ipsum.

4.2.17 DoctrineMoney

Lorem ipsum.

4.2.18 Aop

Lorem ipsum.

4.2.19 Clock

Lorem ipsum.

4.2.20 Redis

Lorem ipsum.

4.2.21 ParseUseStatements

Lorem ipsum.

4.2.22 RedisActiveLock

Lorem ipsum.

4.2.23 TesterParallelStress

Lorem ipsum.

4.2.24 Monolog

Lorem ipsum.

4.2.25 ElasticSearch

Lorem ipsum.

4.2.26 DoctrineSearch

Lorem ipsum.

4.2.27 Geocoder

Lorem ipsum.

4.2.28 CsobPaygateNette

Lorem ipsum.

4.2.29 CsobPaymentGateway

Lorem ipsum.

4.2.30 Wkhtmltopdf

Lorem ipsum.

4.2.31 FakeSession

Lorem ipsum.

4.2.32 RequestStack

Lorem ipsum.

4.2.33 StrictObjects

Lorem ipsum.

4.2.34 Facebook

Lorem ipsum.

4.2.35 Google

Lorem ipsum.

4.2.36 Github

Lorem ipsum.

4.2.37 NettePhpServer

Lorem ipsum.

4.2.38 TesterExtras

Lorem ipsum.

4.2.39 HtmlValidatorPanel

Lorem ipsum.

5 The refactoring process of Kdyby

This chapter documents what I have accomplished with each package in detail.

5.1 Doctrine

Lorem ipsum.

5.2 Console

Lorem ipsum.

5.3 Events

Lorem ipsum.

5.4 Annotations

Lorem ipsum.

5.5 DoctrineCache

Lorem ipsum.

5.6 DoctrineMagicAccessors

Lorem ipsum.

5.7 DoctrineCollectionsReadonly

Lorem ipsum.

5.8 DoctrineCollectionsLazy

Lorem ipsum.

5.9 DoctrineDbalBatchImport

Lorem ipsum.

5.10 DoctrineForms

Lorem ipsum.

5.11 Autowired

Lorem ipsum.

5.12 FormsReplicator

Lorem ipsum.

5.13 Translation

Lorem ipsum.

5.14 Validator

Lorem ipsum.

5.15 RabbitMq

Lorem ipsum.

5.16 Money

Lorem ipsum.

5.17 DoctrineMoney

Lorem ipsum.

5.18 Aop

Lorem ipsum.

5.19 Clock

Lorem ipsum.

5.20 Redis

Lorem ipsum.

5.21 ParseUseStatements

Lorem ipsum.

5.22 RedisActiveLock

Lorem ipsum.

5.23 TesterParallelStress

Lorem ipsum.

5.24 Monolog

Lorem ipsum.

5.25 ElasticSearch

Lorem ipsum.

5.26 DoctrineSearch

Lorem ipsum.

5.27 Geocoder

Lorem ipsum.

5.28 CsobPaygateNette

Lorem ipsum.

5.29 CsobPaymentGateway

Lorem ipsum.

5.30 Wkhtmltopdf

Lorem ipsum.

5.31 FakeSession

Lorem ipsum.

5.32 RequestStack

Lorem ipsum.

5.33 StrictObjects

Lorem ipsum.

5.34 Facebook

Lorem ipsum.

5.35 Google

Lorem ipsum.

5.36 Github

Lorem ipsum.

5.37 NettePhpServer

Lorem ipsum.

5.38 TesterExtras

Lorem ipsum.

5.39 HtmlValidatorPanel

Lorem ipsum.

6 Conclusion and future work

We made it!

Bibliography

[1] Wikipedia. *Open-source software — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Open-source_software&oldid=776201239.

[2] Wikipedia. *PHP — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=PHP&oldid=775984544.

[3] *Packages from kdyby - Packagist*. 2017. URL: https://packagist.org/packages/kdyby/.

[4] Wikipedia. *Content management system — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Content_management_system&oldid=775559667.

[5] Wikipedia. *Free software license — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Free_software_license&oldid=776336813.

[6] Martin Fowler. *Inversion of Control Containers and the Dependency Injection pattern*. 2004. URL: https://martinfowler.com/articles/injection.html.

[7] Wikipedia. *Aspect-oriented programming — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2016. URL: https://en.wikipedia.org/w/index.php?title=Aspect-oriented_programming&oldid=751829822.

[8] Scott Chacon and Ben Straub. *Pro Git*. Apress, 2014. ISBN: 1484200772.

[9] Wikipedia. *Continuous integration — Wikipedia, The Free Encyclopedia*. [Online; accessed 28-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Continuous_integration&oldid=777232210.

[10] Wikipedia. *Nette Framework — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Nette_Framework&oldid=766339218.

[11] Wikipedia. *Package manager — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Package_manager&oldid=775128084.

[12] Composer Community. *Introduction - Composer*. 2017. URL: https://getcomposer.org/doc/00-intro.md.

[13] Wikipedia. *JSON — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=JSON&oldid=775860167.

[14] David Grudl. *Dibi is Database Abstraction Library for PHP 5 - Homepage*. 2017. URL: https://dibiphp.com/.

[15] Wikipedia. *Object-relational mapping — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Object-relational_mapping&oldid=769454010.

[16] Wikipedia. *Object-oriented programming — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Object-oriented_programming&oldid=775231020.

[17] Wikipedia. *Symfony — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Symfony&oldid=769598534.

[18] PHP Framework Interop Group. *PHP Standards Recommendations*. 2017. URL: http://www.php-fig.org/psr/.

[19] Jordi Boggiano. *Monolog - Logging for PHP*. 2017. URL: https://seldaek.github.io/monolog/.

[20] Wikipedia. *RabbitMQ — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=RabbitMQ&oldid=774630105.

[21] Wikipedia. *Elasticsearch — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Elasticsearch&oldid=776540688.

[22] Wikipedia. *Redis — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Redis&oldid=774326024.

[23] Ondřej Mirtes. *PHPStan - PHP Static Analysis Tool*. 2017. URL: https://github.com/phpstan/phpstan.

[24] Wikipedia. *Unit testing — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-April-2017]. 2017. URL: https://en.wikipedia.org/w/index.php?title=Unit_testing&oldid=773620225.

[25] David Grudl. *Nette Tester – enjoyable unit testing*. 2017. URL: https://tester.nette.org/en/.