

Code Quality Audit

To Do List Application

December 2020

Caroline Dirat



Formation OpenClassrooms

« Application Developer PHP/Symfony »

Project 8 – **Improve an existing application**

TABLE DES MATIÈRES

Code quality measurement	2
Security	3
Technical debt	3

CODE QUALITY MEASUREMENT


CODECLIMATE

 maintainability **A**

Code Climate is an application which evaluate the code maintainability.

To Do List code obtains a maintainability rate of **A** on Code Climate.

CODACY

 code quality **A**

Codacy is an application which analyzes the code quality, while being configurable. Here, Codacy analyzes the code style of three languages:

- **PHP** (with PHP Code Sniffer and PHP Mess Detector)
- **CSS** (with CSSLint)
- **Javascript** (with JSLint)

Codacy rate also the code **complexity** and the code **duplications**, which have an impact on the code maintainability.

The **To Do List** application code gets an **A** rating, with **0% of code Complexity** and **0% of code Duplication**.

As a result of our work to improve the To Do List application, we get a good code quality rating.

UNIT AND FUNCTIONNAL TESTING

coverage **100.00%**

The **rate of code coverage** by **unit** and **functional tests** via **PHPUnit** is **100%**. This does not mean that the code is 100% tested, while the Code specific to the Symfony framework and the code of dependencies in the vendor/file code is not tested. But the tests are present on **critical business needs** (for example, access to user management only by users with the role `ROLE_ADMIN`).

The application also **presents functional tests** via **Behat**, in order to test user **scenarios** that can be read in the `.feature` files of the `/features` folder.

PHPUnit tests ensure the code quality of the application, and in particular its **maintainability**, as they allow you to evaluate the impact of a code change on existing features.

In addition, let's take a look at security management in the application.

SECURITY

PROTECTION DES FORMULAIRES

- All forms are protected by a token against Cross-Site Request Forgery attacks, including the login form.
- Incoming data are filtered through validation constraints.

PROTECTION CONTRE LA FAILLE XSS - CROSS SITE SCRIPTING

- ▶ Filtering of incoming data (thanks to **Symfony** validation constraints)
- ▶ Escape data in outputs (by **Twig**)

VULNÉRABILITÉ DES DÉPENDANCES

The **Symfony framework** version used to build the app was initially on a version that is no longer maintained (version 3.1). We now use the lts version (4.4.17), which offers 3 years of support for bugs and security patches.

Also, Symfony's binary offers a tool to verify the security vulnerabilities of the application's dependencies:

```
symfony check:security
Symfony Security Check Report
=====
No packages have known vulnerabilities.
```

Therefore, the project **dependencies do not present any security vulnerabilities.**

TECHNICAL DEBT

SYMFONY BEST PRACTICES

Two points need to be improved to be in line with [Symfony's best practices](#), in order to optimize login security for one and code maintainability for the other.

➤ The authentication provider

As explained in the [technical documentation on the authentication's implementation](#), the To Do List application authentication method uses the **form_login** authentication provider.

Symfony recommends creating an authentication by fom login with **Guard**. Indeed, the **Guard authenticator** allows **total control** over the **authentication process** (which is not the case with form_login) and allow to **optimize the security** of the connection to the application.

To implement a Guard authenticator, the process is explained in [Symfony's official documentation](#).

➤ Utiliser **Webpack Encore** pour traiter les ressources Web

The development remains very sketchy on the frontend side. When the application expands on that side, it will be interesting to **optimize** the **management** of **CSS** and **Javascript** files by installing **Webpack Encore**.

Webpack Encore will also integrate **Bootstrap** into the app, instead of loading its files via the CDNs pointed from the basic template (templates/base.html.twig). See [Symfony's official documentation](#).

Making easier to manage CSS and JavaScript files, we improve the **application's maintainability**.

SEO (SEARCH ENGINE OPTIMISATION)

The application will also need improvements from the SEO's point of view, that is to say its ability to be referenced in search engines. The [opquast](#) website lists 80 best practices to follow.

Many of these good practices are already in effect in the To Do List application:

- ✓ ALL TEMPLATES ARE RESPONSIVES
- ✓ Each image has a textual alternative
- ✓ The length of textual alternatives is less than 80 characters.
- ✓ Each page contains one H1 title element
- ✓ There is as many different H1 section titles as pages.
- ✓ The home page shows the nature of the contents and proposed services.
- ✓ Images are consistent with the contents of the page
- ✓ Wording of each hyperlink describes its function or the nature of the content which it points.
- ✓ The URLs do not contain any indication of session settings.
- ✓ The URLs of internal links contain exclusively alphanumeric characters or considered safe.
- ✓ All hyperlinks on the site are valid. (and checked in Behat functional tests)
- ✓ Links use a single URL for each page
- ✓ The title of each page (title element) allows to identify its content
- ✓ Source code of each page contains a metadata that defines the character set
- ✓ The content of the TITLE element on each page does not start with the name of the site
- ✓ The navigation is possible via HTML links
- ✓ The HTML contents are shaped with outsourced CSS style
- ✓ The server send a 404 HTTP code for not found resources.

But we can still improve the HTML **code** (files from the templates/ directory).

Like what:

- ▶ Many **div** tags must be replaced by more specific [HTML5 content tags](#) (**section**, **article**, **main...**)
- ▶ The image textual alternative should be more explicit than the simple « todo list »
- ▶ The terms of images's textual alternatives should be also present in the content of the page
- ▶ Capitalizing for decorative purposes must be made using CSS styles
- ▶ The content targeted for SEO should be highlighted (with **strong** and **em** tags).
- ▶ In the <meta name="description">, content attribute should be provided with the description of the page's own content (a meaningful sentence, not a list of words).
- ▶ In the <meta name="author"> tag, content attribute should be informed
- ▶ The HTML section titles must include key words contained in the **meta keywords** tag