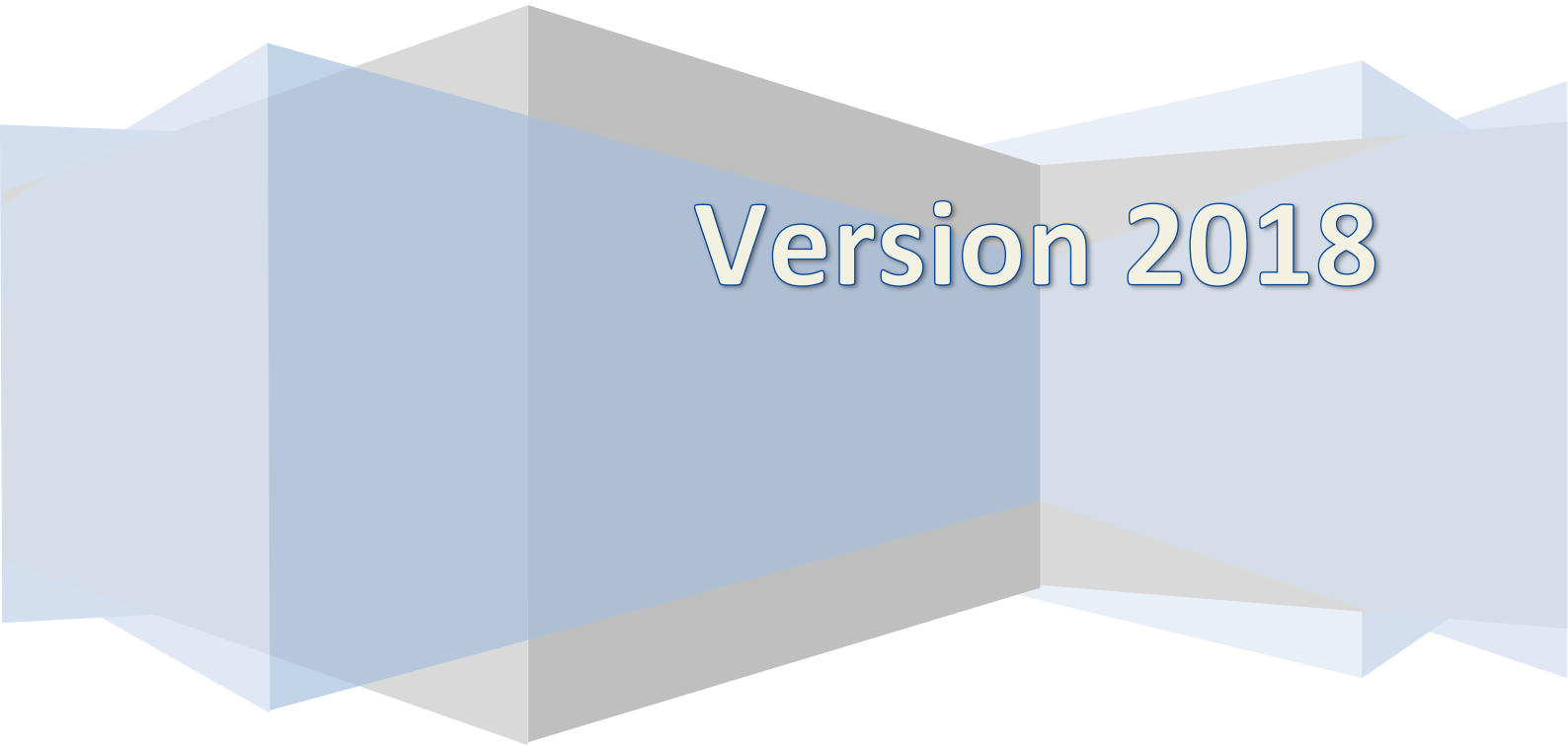
Projet de fin de formation

Formation SF/WebForce 3

Projet – Gestionnaire de stock

Réalisé par Marsiglietti Rémy – admin@frogg.fr – le 23/04/2018

Version 1.00 le 23/04/2018



Objectifs

L'objectif du projet est de pouvoir gérer un stock de produit via une interface web. Le but étant d'appliquer les connaissances acquises pendant la formation.

Liste des technologies utilisées

- Partir du Skeleton fournis par Symfony
  https://symfony.com/doc/current/setup.html
  composer create-project symfony/skeleton Symfony_StockManager
  Afin d'utiliser les derniers outils disponible, j'utilise la version 4.1 (encore en développement), pour ce faire dans la configuration composer j'ai changé
  "symfony/framework-bundle": "dev-master"
- Composant Web server bundle pour lancer un serveur web via la console
  https://symfony.com/doc/current/setup/built_in_web_server.html
  composer require server –dev
- Composant Profiler pour avoir la barre de débogage Symfony
  https://symfony.com/doc/current/profiler.html
  composer require profiler –dev
- Composant Security-checker pour vérifier la viabilité des packages installés
  https://symfony.com/doc/current/security/security_checker.html
  composer require sec-checker --dev
- Composant Twig pour gérer l'affichage des pages web
  https://symfony.com/doc/current/reference/configuration/twig.html
  composer require twig
- Composant Asset pour gérer les éléments des pages web
  https://symfony.com/doc/current/components/asset.html
  composer require asset
- Composant WebPack-encore pour gérer la compilation des éléments des pages web
  https://symfony.com/doc/current/frontend.html
  npm install @symfony/webpack-encore --save-dev
- Composant Doctrine pour gérer la base de données
  https://symfony.com/doc/current/doctrine.html
  composer require doctrine
- Composant Form pour gérer les formulaires
  https://symfony.com/doc/current/forms.html
  composer require form
- Composant Validator pour gérer les validations des données
  https://symfony.com/doc/current/components/validator.html
  composer require validator
- Composant Annotation et pour gérer les Annotation (pour les routes par exemple)
  https://symfony.com/doc/current/routing.html
  composer require annotations

- Composant Param Converter pour faire des manipulations de données dans les annotations
  http://symfony.com/doc/master/bundles/SensioFrameworkExtraBundle/annotations/converters.html
  composer require expression-language
- Composant Maker pour générer des fonctionnalités
  https://symfony.com/blog/introducing-the-symfony-maker-bundle
  composer require maker --dev
- Composant Security pour gérer les utilisateurs
  https://symfony.com/doc/current/security.html
  composer require security
- Composant Mailer pour gérer les envois de mail
  https://symfony.com/doc/current/email.html
  composer require mailer
- Composant Translator pour gérer les traductions
  https://symfony.com/doc/current/translation.html
  composer require translator
- Composant Browser-kit pour gérer le crawl de pages distantes
  https://symfony.com/doc/current/components/browser_kit.html
  composer require browser-kit
- Composant Css-selector pour gérer plus facilement la sélection d'éléments html lors des crawls
  https://symfony.com/doc/current/components/browser_kit.html
  composer require browser-kit
- Composant Thanks pour faire plaisir aux développeurs de Symfony
  https://symfony.com/doc/current/components/browser_kit.html
  composer require thanks --dev

Liste des technologies utilisées pour les tests unitaires

- Composant PHP Unit Bridge pour faire des tests unitaires
  https://symfony.com/doc/current/components/browser_kit.html
  composer require phpunit-bridge –dev
  php bin\phpunit
  Les composants Browser-kit et Css-selector seront aussi utilisés pour les tests unitaires.

Liste des technologies utilisées pour les tests fonctionnels

http://behat.org/en/latest/cookbooks/integrating_symfony2_with_behat.html#installing-behat-in-your-symfony2-project

- Composant behat/mink pour pouvoir utiliser behat
  composer require behat/mink –dev
- Composant behat/mink-browserkit-driver pour pouvoir utiliser lier behat à un navigateur
  composer require behat/mink-browserkit-driver --dev
- Composant behat/mink-extension pour pouvoir ajouter des extensions à behat
  composer require behat/mink-extension –dev
- Composant behat/mink-goutte-driver pour pouvoir utiliser goutte dans behat
  composer require behat/mink-goutte-driver –dev
- Composant behat/mink-selenium2-driver pour pouvoir utiliser selenium2 dans behat
  composer require behat/mink-selenium2-driver –dev
- Composant behat/Symfony2-extension pour pouvoir gérér une application Symfony dans behat

Choix de la version 4.1 de Symfony

Symfony 4.1

Afin de comprendre l'univers Symfony, j'ai décidé d'utiliser la version en cours de développement 4.1 pour découvrir comment une version est mise en place, ainsi que tester la stabilité de la branche « dev ».

De plus Symfony 4.1 apporte de nombreuses nouvelles fonctionnalités, dont certaines qui m'ont particulièrement intéressé, voici la liste des fonctionnalités 4.1 utilisées :

CONSOLE SECTION

https://symfony.com/blog/new-in-symfony-4-1-advanced-console-output

Les sections permettent de créer des zones dynamiques dans la console, ce qui offre de nombreuses possibilité d'affichage en ayant la possibilité de modifié du contenu déjà affiché ce qui n'étais pas possible dans les versions précédente.

Un exemple d'utilisation est disponible dans la commande app:userManager du projet

INLINED ROUTING CONFIGURATION

https://symfony.com/blog/new-in-symfony-4-1-inlined-routing-configuration

Cette fonctionnalité permet d'écrire dans les routes des configurations directement sans avoir à écrire les paramètres requis et les paramètres par défaut.

Par exemple : `{_locale<fr|en>?en}` signifie que les requis sont « fr » ou « en » et que par défaut la valeur est « en »

INTERNATIONALIZED ROUTING

https://symfony.com/blog/new-in-symfony-4-1-internationalized-routing

Une des fonctionnalités les plus intéressantes, depuis Symfony 4.1 il est maintenant possible de traduire les routes en fonction de la « locale ».

Par exemple `@Route( { "fr": "/compte", "en": "/account" }, name="account" )` permet d'afficher /compte en français et /account en anglais.

CUSTOM USER ABSTRACT CLASS

https://symfony.com/blog/new-in-symfony-4-1-deprecated-the-advanceduserinterface

L'advancedUserInterface qui permettait de gérer des options avancée d'un utilisateur va être progressivement retiré de Symfony. Laissant les développeurs libres de mettre en place leur propre système de gestion complexe d'utilisateur.

L'exemple typique d'utilisation est la création d'un compte inactif tant qu'il n'a pas été validé par mail.

La création de ces restrictions soit même permet d'aller plus loin dans la compréhension de la gestion de la sécurité dans symfony.

Planning

Pour réaliser le projet le planning suivant a été mis en place:

SEMAINE 1

- Mise en place du projet sur GitHub
- Réflexion sur les problématiques et la technique a utiliser pour la réalisation du projet
- Maquette du site à l'aide de balsamique
- Réalisation du des pages front au format statique fronts

SEMAINE 2

- Création du projet Symfony
- Mise en place de la compilation des éléments statiques avec Encore
- Mise en place de la sécurité (connexion, création de compte, récupération de mot de passe, et les mails correspondants)
- Mise en place des traductions

SEMAINE 3

- Mise en place de tests unitaires
- Mise en place de tests fonctionnels

SEMAINE 4

- Réalisation de la gestion des produits (scrap des produits, liste des produits, ajout, suppression…)
- Mise en place de commandes (gestion de la base, gestion des utilisateurs, ajout de code bar, envoie de mail automatique lié a l'expiration des produits)

Afin de suivre le planning des milestones ont été créés sur le Github du projet :

https://github.com/FroggDev/Symfony_StockManager/milestones?state=closed

La liste des taches & assignations sont définies dans la partie Project du Github :

https://github.com/FroggDev/Symfony_StockManager/projects

Le suivi des branches est disponible sur les rapports du Github :

https://github.com/FroggDev/Symfony_StockManager/network

Présentation

1. Présentation du Github

2. Présentation du Draft

doc_fonctional.bmpr & Barcode_workflow.png & Database.png

3. Présentation du web statique

4. Présentation du projet

5. Présentation des commandes

```
php bin/console app:userManager
```

```
php bin/console app:database create/update/remove
```

```
php bin/console app:product:add 3250390779100 3256224398264 8002270015786
```

```
php bin/console app:expires:alert
```

6. Présentation des tests unitaires

```
php bin\phpunit --coverage-html public/output/phpunit
```

7. Présentation des tests fonctionnels

Lancer selenium

```
vendor\bin\behat
```

8. Présentation des outils de reporting

http://127.0.0.1:8000/demo/index.html

http://127.0.0.1:8000/demo/phpunit/index.html

http://127.0.0.1:8000/demo/behat/index.html

http://127.0.0.1:8000/demo/phpmetrics.html

http://127.0.0.1:8000/demo/phpdoc/index.html

9. Présentation de la configuration de phpStorm

https://tool.frogg.fr/phpStorm

10. Les bugs rencontrés

Bug phpunit & composer.phar location : https://github.com/symfony/symfony/issues/26637

Bug Command with phpunit (& symfony style) : https://github.com/symfony/symfony/issues/26885

Bug profiler : https://github.com/symfony/symfony/issues/26855

Bug Security AppSecret : https://github.com/symfony/symfony/issues/26860

Liste des commandes principales

Application

```
php bin/console app:userManager
```

```
php bin/console app:database create/update/remove
```

```
php bin/console app:product:add 3250390779100 3256224398264 8002270015786
```

```
php bin/console app:expires:alert
```

Css/js Compilation

```
encore dev
```

```
encore dev --watch
```

```
encore production
```

Routes

```
php bin/console debug:router
```

Database

```
php bin/console doctrine:database:create
```

```
php bin/console doctrine:schema:validate
```

```
php bin/console doctrine:migrations:diff
```

```
php bin/console doctrine:migrations:migrate
```

```
php bin/console doctrine:mapping:convert annotation ./src/ExportedEntity --from-database
```

```
php bin/console doctrine:make:entity Origin
```

```
php bin/console doctrine:database:import sql\country.sql
```

Translation

```
php bin/console debug:translation fr
```

```
php bin/console debug:translation fr --only-missing
```

```
php bin/console translation:update --dump-messages --force fr --output-format xlf
```

Container

```
php bin/console debug:container
```