

# Audit

Améliorez une application existante de ToDo & Co

## Table des matières

<b>Présentation .....</b>	<b>1</b>
<b>Environnement.....</b>	<b>1</b>
<b>Analyse.....</b>	<b>2</b>
<b>Mise à jour .....</b>	<b>4</b>
<b>Symfony .....</b>	<b>4</b>
<b>Dépendances .....</b>	<b>5</b>
<b>Formulaires.....</b>	<b>6</b>
<b>Authentification .....</b>	<b>7</b>
Configuration .....	7
Provider .....	8
Pare-feu .....	8
Contrôle des accès.....	9
Sécurisation et mise à jour .....	9
L'authentification.....	9
<b>Améliorations .....</b>	<b>12</b>
<b>Gestions des rôles .....</b>	<b>12</b>
<b>Limitations des accès.....</b>	<b>13</b>
<b>Token CSRF .....</b>	<b>13</b>
<b>Injections SQL .....</b>	<b>13</b>
<b>Faillles XSS .....</b>	<b>13</b>
<b>Pages d'erreurs .....</b>	<b>14</b>
<b>Fixtures.....</b>	<b>14</b>
<b>Tests.....</b>	<b>14</b>
<b>Autres améliorations .....</b>	<b>15</b>
Tri des tâches .....	15
Liens HyperText .....	15
Règlement général sur la protection des données .....	15
<b>Respect des normes .....</b>	<b>15</b>
<b>Performances .....</b>	<b>15</b>
<b>Analyse initiale.....</b>	<b>17</b>
<b>Analyse après mise à jour.....</b>	<b>17</b>
Piste d'optimisation.....	18

## Présentation

ToDo & Co est une startup récente dont le cœur de métier est une application permettant de gérer ses tâches quotidiennes.

L'entreprise vient tout juste d'être montée et l'application a dû être développée le plus rapidement possible afin de pouvoir montrer, à des investisseurs, que le concept est viable.

**ToDo & Co** a réussi, très récemment, à lever des fonds pour permettre le développement de l'entreprise et surtout de l'application.

## Environnement

L'application a été développée grâce au framework Symfony, sous sa version 3.1.6.

Pour pouvoir utiliser cette version de Symfony, il est nécessaire d'utiliser une version de PHP plus ancienne (PHP 5.5.9 est recommandé)

Afin de pouvoir contribuer au projet, quelques prérequis sont nécessaires :

- S'assurer que **Composer** est installé sur votre machine : avec cela, nous pourrons mettre à jour l'application, ses dépendances et aussi en ajouter de nouvelles.
- S'assurer que **Git** est installé sur votre machine : permet de pouvoir cloner le projet et interagir avec le repository déjà créé par le précédent développeur.
- S'assurer que la version **PHP** correspond à celle qui est déclarée dans le fichier `composer.json` (PHP 5.5.9)

Liste des dépendances de l'application :

- doctrine/orm
- doctrine/doctrine-bundle
- doctrine/doctrine-cache-bundle
- symfony/swiftmailer-bundle
- symfony/monolog-bundle
- symfony/polyfill-openssl
- sensio/distribution-bundle
- sensio/framework-extra-bundle
- innteev/composer-parameter-handler

L'application utilise la dépendance Doctrine afin de pouvoir interagir avec la base de données grâce à l'utilisation de PDO.

## Analyse

Cette application, dans la version actuelle, a été développée dans le but de pouvoir démontrer ses fonctionnalités de base à de futurs investisseurs.

Il n'y a donc très peu voire aucune optimisation de performances.

Lors des tests, l'application a été couplée à Codacy et CodeClimate afin de détecter les éventuelles issues.

Il est possible de les retrouver facilement via les liens suivants :

- <https://codeclimate.com/github/damientabet/test>
- <https://app.codacy.com/manual/damientabet/test/dashboard?bid=19327986>

GRADE ^	FILENAME ^	ISSUES ^	DUPLICATION ^	COMPLEXITY ^
B	var/SymfonyRequirements.php	26	0	25
F	web/css/shop-homepage.css	15	-	-
C	web/config.php	14	0	-
D	web/app_dev.php	9	0	-
C	app/AppKernel.php	3	0	2
D	web/app.php	2	0	-
B	src/AppBundle/Controller/SecurityController.php	1	0	1
B	src/AppBundle/Form/ UserType.php	1	0	1
A	src/AppBundle/Controller/UserController.php	1	0	2
A	src/AppBundle/Entity/Task.php	1	0	1
D	app/autoload.php	1	0	-
A	src/AppBundle/Controller/TaskController.php	1	0	2
F	README.md	1	-	-
C	src/AppBundle/Form/TaskType.php	1	0	1
A	src/AppBundle/Entity/User.php	1	0	1

✓ Auth	>
✓ CSRF	>
! CommandInjection	>
! Cryptography	>
! HTTP	>
✗ InputValidation found in 2 files	>
✗ InsecureModulesLibraries found in 4 files	>
! InsecureStorage	>
! MaliciousCode	>
✓ Regex	>
✓ Routes	>
✓ SQLInjection	>
✓ UnexpectedBehaviour	>
✗ XSS found in 2 files	>
✗ Other found in 5 files	>

Nous pouvons voir des erreurs et points d'attentions qui semblent importantes, seulement il s'agit très souvent d'issues liés au framework Symfony, dont nous n'avons pas la possibilité de modifier sans entrer dans le cœur de l'application.

Il est tout de même à noter que ces logiciels de qualité de code ne prennent pas en compte les spécificités du framework.

Par exemple, les failles XSS sont protégées au niveau du moteur de template Twig, la protection est donc active, mais non reconnue par Codacy.

## Breakdown

56 FILES



TEST COVERAGE

## Codebase summary

MAINTAINABILITY



TEST COVERAGE



Repository stats

CODE SMELLS

26

DUPLICATION

36

OTHER ISSUES

0

Au niveau de CodeClimate, beaucoup de duplication de code sont remontées.

En effet, il s'agit très souvent du fichier **bootstrap.js** qui se trouve dans le dossier

**web/js/bootstrap.js**

Cependant, il n'est pas possible d'interagir avec ce fichier.

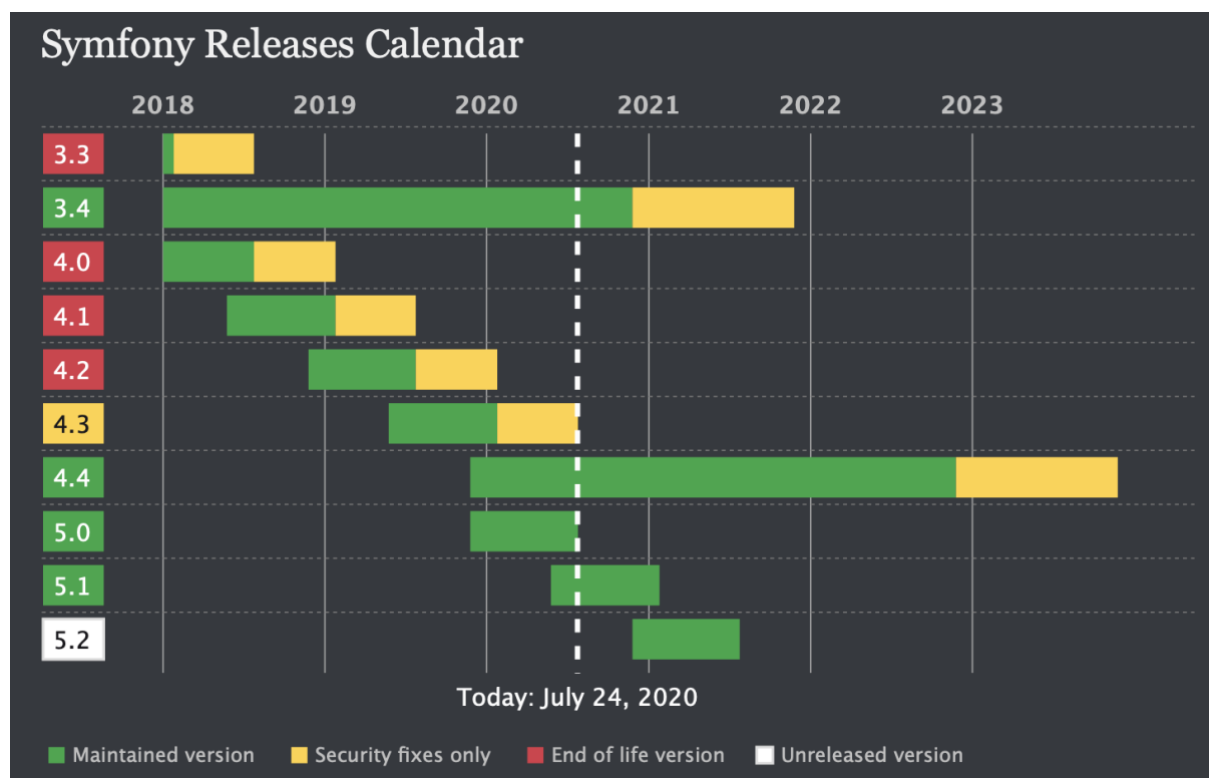
Pour éviter ce type d'issues, il pourrait être possible d'utiliser les CDNs de Bootstrap afin de l'intégrer à l'application, mais cela pourrait créer des ralentissements sur le chargement des pages.

## Mise à jour

### Symfony

Dans un premier temps, nous avons mis à jour l'application vers la dernière version LTS (Long Term Support) de Symfony.

Une version LTS est donc la version de Symfony qui sera le plus longtemps maintenue par la société SensioLabs. Généralement, ce type de version est maintenu pendant environ 3 ans.



Au vu du calendrier fournit par Symfony à l'adresse suivante, nous pouvons voir que la dernière version LTS est la 4.4, et plus précisément la 4.4.10.

SensioLabs continuerons donc de faire des mises à jour de la version 4.4 jusqu'à fin 2022, puis ils appliqueront des correctifs de sécurité jusqu'au courant de l'année 2023.

Ce qui signifie qu'à ce jour, nous sommes assurés que notre application fonctionnera et qu'il sera assez facile de la mettre à jour en fonction de nos besoins ou alerte de sécurité.

<https://symfony.com/releases>

Malheureusement, comme il est spécifié sur cette page, la version LTS ne prend pas en compte plusieurs choses l'ajout des nouvelles fonctionnalités.

Cela est réservée à la version plus récente, mais encore en développement et sans version LTS, Symfony 5.

Actuellement stable en 5.1.3, et la version 5.2 est en cours de développement.

Latest Stable Release	Latest Long-Term Support Release
<b>5.1.3</b>	<b>4.4.11</b>
<ul style="list-style-type: none"><li>• First released in May 2020.</li><li>• <b>Recommended for most users.</b></li><li>• Includes the latest features.</li><li>• It's easier to upgrade to newer versions.</li></ul>	<ul style="list-style-type: none"><li>• First released in November 2019.</li><li>• 3 year support for bugs and security fixes.</li><li>• It doesn't include the latest features.</li><li>• It's harder to upgrade to newer versions.</li></ul>

## Dépendances

Toutes les dépendances doivent également être mise à jour afin de pouvoir faire fonctionner la nouvelle version de Symfony.

Certaines ont été supprimé, ajoutés et d'autres ont été mises à jour.

Pour pouvoir connaître ce type de modifications à apportées, il est nécessaire de se rendre sur la site Packagist (<https://packagist.org/>) et de rechercher **symfony/symfony** (<https://packagist.org/packages/symfony/symfony>).

Liste des dépendances ajoutées :

- blackfire/php-sdk
- doctrine/doctrine-migrations-bundle
- symfony/asset
- symfony/console
- symfony/flex
- symfony/form
- symfony/framework-bundle
- symfony/lts
- symfony/maker-bundle
- symfony/validator
- symfony/security-bundle

- symfony/twig-bundle
- symfony/yaml
- doctrine/doctrine-fixtures-bundle
- liip/test-fixtures-bundle
- symfony/debug-pack
- symfony/dotenv
- symfony/test-pack
- symfony/web-server-bundle

#### Liste des dépendances supprimées :

- doctrine/orm
- doctrine/doctrine-bundle
- doctrine/doctrine-cache-bundle
- symfony/swiftmailer-bundle
- symfony/monolog-bundle
- symfony/polyfill-apcu
- sensio/distribution-bundle
- incenteev/composer-parameter-handler
- sensio/generator-bundle

#### Liste des dépendances mises à jour :

- sensio/framework-extra-bundle
- symfony/phpunit-bridge

Nous pouvons remarquer que la majorité des dépendances ont été supprimé.

Il s'agit principalement du fait que la structure du framework est totalement différente entre la version 3 et la version 4.

## Formulaires

Les formulaires ont été mis à jour grâce à la nouvelle dépendance **symfony/form**.

Dans la globalité, le code est toujours le même, mais avec le passage à la version 4 de Symfony, il était tout de même nécessaire de mettre à jour le paramètre permettant l'utilisation de Bootstrap pour la stylisation des formulaires.



Version 3 de Bootstrap sur Symfony 3.1.6

/app/config/config.yml

```
# Twig Configuration
twig:
    debug:          "%kernel.debug%"
    strict_variables: "%kernel.debug%"
    form_themes: ['bootstrap_3_layout.html.twig']
```

Version 4 de Bootstrap sur Symfony 4.4.10

/config/packages/twig.yaml

```
twig:
    default_path: '%kernel.project_dir%/templates'
    debug: '%kernel.debug%'
    strict_variables: '%kernel.debug%'
    exception_controller: null
    form_themes: ['bootstrap_4_layout.html.twig']
```

## Authentication

Suite au passage à Symfony 4, l'authentification de l'application à dû être également mis à jour.

Ceci a été supprimé, puis elle a totalement été reconstruit grâce au **symfony/maker-bundle** présent dans cette version du framework.

[https://symfony.com/doc/4.4/security/form\\_login\\_setup.html](https://symfony.com/doc/4.4/security/form_login_setup.html)

## Configuration

La configuration du composant de sécurité s'effectue dans le fichier :

*config/packages/security.yaml*

Les mots de passe sont cryptés par le composant de sécurité de Symfony. Pour modifier l'algorithme de cryptage, il faut modifier la section *encoders* du fichier de configuration.

```
1 security:
2     encoders:
3         App\Entity\User:
4             algorithm: auto
```

Les mots de passe sont cryptés grâce à une passphrase définie dans le fichier « *.env* » situé à la racine du projet (APP\_SECRET).

## Provider

Dans le fichier *config/packages/security.yaml*, il est possible de configurer plusieurs fournisseurs d'authentification.

L'entité *User* étant utilisée dans ce projet, l'authentification est configurée dans la section *providers*. Il est également nécessaire de définir l'attribut de la classe *User* qui sera utilisé comme identifiant de connexion.

```
providers:
    # used to reload user from session & other features (e.g. switch_user)
    users_in_memory:
        entity:
            class: App\Entity\User
            property: email
```

Veuillez noter que le nom du provider doit également être renseigné dans la partie *main* du *firewall*

```
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        anonymous: lazy
        provider: users_in_memory
        guard:
            authenticators:
                - App\Security\LoginFormAuthenticator
        logout:
            path: logout
```

---

## Pare-feu

La partie *firewalls* permet de définir les parties de l'application doivent être gérées par le composant de sécurité.

Elle définit également :

- La classe chargée de réaliser l'authentification (Guard : authenticators)
- Le nom du provider utilisé (provider)
- La route permettant la déconnexion (logout)

Le pare-feu indique la partie du site qui doit être gérée par le composant de sécurité, mais ne protège pas l'application.

Les zones à protéger sont configurées dans la partie *access\_control*.

```

firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security: false
  main:
    anonymous: lazy
    provider: users_in_memory
    guard:
      authenticators:
        - App\Security\LoginFormAuthenticator
    logout:
      path: logout

```

## Contrôle des accès

La partie *access\_control* permet de définir les parties de l'application qui doivent être protégées par le processus d'authentification.

Elle permet également de définir les rôles autorisés pour chaque URL.

```

access_control:
  - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
  - { path: ^/users/*, roles: ROLE_ADMIN }
  - { path: ^/, roles: IS_AUTHENTICATED_FULLY }

```

## Sécurisation et mise à jour

Lors du passage à la version 4.4 de Symfony, le formulaire de connexion utilisateur a également été mis à jour.

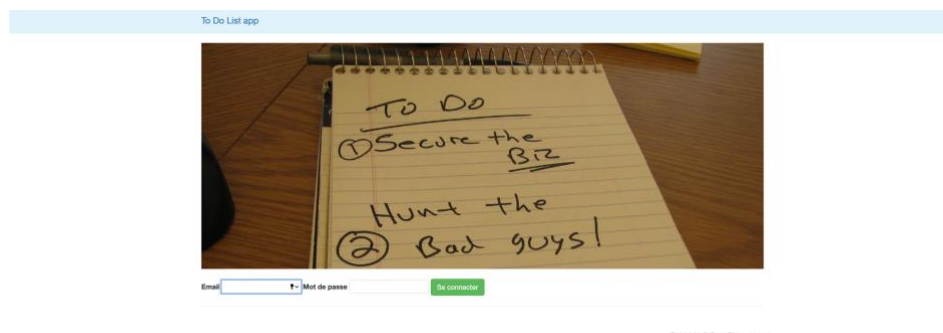
La clé unique est maintenant l'email et non plus le pseudo.

Ce choix a été fait puisque l'email est un identifiant de connexion bien plus actuel et une adresse mail est plus facilement unique sans trop restreindre la création d'un utilisateur.

## L'authentification

### Formulaire

Pour afficher le formulaire de connexion, il est nécessaire d'accéder à l'url **/login**



## Soumission

1. L'utilisateur remplit le formulaire et le soumet grâce au bouton « Se connecter ».
2. Comme nous avons pu voir ci-dessus, dans la partie « [Pare-feu](#) », l'authentification est configurée pour utiliser le composant de sécurité de Symfony, qui est ensuite gérée par la classe *LoginFormAuthenticator* qui se trouve dans *src/Security/LoginFormAuthenticator.php*.

```
/**
 * Class LoginFormAuthenticator
 * @package App\Security
 * @codeCoverageIgnore
 */
class LoginFormAuthenticator extends AbstractFormLoginAuthenticator implements PasswordAuthenticatedInterface
{
    use TargetPathTrait;

    public const LOGIN_ROUTE = 'login';

    private $entityManager;
    private $urlGenerator;
    private $csrfTokenManager;
    private $passwordEncoder;

    public function __construct(EntityManagerInterface $entityManager, UrlGeneratorInterface $urlGenerator, CsrfTokenManagerInterface $csrfTokenManager, PasswordEncoderInterface $passwordEncoder)
    {
        $this->entityManager = $entityManager;
        $this->urlGenerator = $urlGenerator;
        $this->csrfTokenManager = $csrfTokenManager;
        $this->passwordEncoder = $passwordEncoder;
    }

    public function supports(Request $request)
    {
        return self::LOGIN_ROUTE === $request->attributes->get('_route')
            && $request->isMethod('POST');
    }
}
```

3. La soumission est alors interceptée, puis les données envoyées via le formulaire sont stockées dans le tableau **\$credentials** et la valeur email (champ unique) est ajoutée dans une variable de session.

Un 3<sup>ème</sup> paramètre est envoyé, le paramètre « csrf\_token » est une clé de sécurité envoyée par le formulaire de connexion permettant de protéger le formulaire contre le « *Cross site request forgery* » ou contre les éventuelles attaques par des tiers.

```
public function getCredentials(Request $request)
{
    $credentials = [
        'email' => $request->request->get('email'),
        'password' => $request->request->get('password'),
        'csrf_token' => $request->request->get('_csrf_token'),
    ];
    $request->getSession()->set(
        Security::LAST_USERNAME,
        $credentials['email']
    );

    return $credentials;
}
```

4. Après que le token CSRF a été vérifié, La valeur du champ email envoyée est utilisé pour rechercher un utilisateur associé dans la base de données grâce à l'utilisation de l'entité User et Doctrine.

Si une correspondance est trouvée, l'utilisateur est stocké dans un objet `$user`, sinon une exception est levée.

```
public function getUser($credentials, UserProviderInterface $userProvider)
{
    $token = new CsrfToken('authenticate', $credentials['csrf_token']);
    if (!$this->csrfTokenManager->isTokenValid($token)) {
        throw new InvalidCsrfTokenException();
    }

    $user = $this->entityManager->getRepository(User::class)->findOneBy(['email' => $credentials['email']]);

    if (!$user) {
        // fail authentication with a custom error
        throw new CustomUserMessageAuthenticationException('Email could not be found.');
```

5. Le mot de passe crypté envoyé est testé pour vérifier sa correspondance avec celui de l'utilisateur chargé.

```
public function checkCredentials($credentials, UserInterface $user)
{
    return $this->passwordEncoder->isPasswordValid($user, $credentials['password']);
}
```

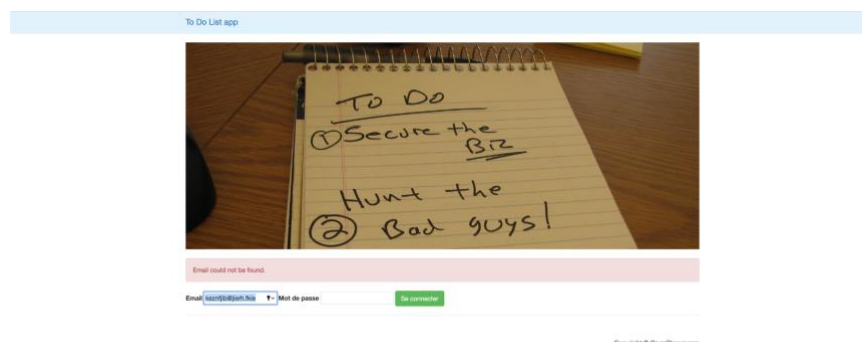
Une fois toutes les informations réunies, vérifiées et qui correspondent parfaitement, alors l'utilisateur est connecté et une redirection est effectuée vers la dernière url visitée, si elle existe ou vers la liste des tâches.

En cas d'échec la fonction `login()` du contrôleur « *SecurityController* » est appelée à nouveau, et le formulaire de connexion est chargé, avec l'erreur affichée.

```
public function onAuthenticationSuccess(Request $request, TokenInterface $token, $providerKey)
{
    if ($targetPath = $this->getTargetPath($request->getSession(), $providerKey)) {
        return new RedirectResponse($targetPath);
    }

    // For example : return new RedirectResponse($this->urlGenerator->generate('some_route'));
    return new RedirectResponse($this->urlGenerator->generate('homepage'));
}

protected function getLoginUrl()
{
    return $this->urlGenerator->generate(self::LOGIN_ROUTE);
}
```



## Améliorations

### Gestions des rôles

Sur la première version de l'application, aucune gestion des rôles n'a été développée.

Pour cela, il a fallu modifier l'entité User afin de faire apparaître une nouvelle propriété « roles ».

Grâce à l'ORM Doctrine et l'utilisation des commandes Symfony, une colonne correspondante à cette propriété a été créée en base de données.

Puis le formulaire de création d'un utilisateur a été modifié afin d'ajouter un champ select « Rôles » permettant de choisir le rôle utilisateur spécifique.

```
/**
 * @ORM\Entity(repositoryClass=UserRepository)
 * @UniqueEntity("email")
 */
class User implements UserInterface
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=180, unique=true)
     * @Assert\NotBlank(message="Votre email est requis")
     * @Assert\Email(message="Le format de l'email n'est pas valide")
     */
    private $email;

    /**
     * @ORM\Column(type="string", length=180, unique=true)
     * @Assert\NotBlank(message="Votre nom d'utilisateur est requis")
     */
    private $username;

    /**
     * @ORM\Column(type="json")
     */
    private $roles = [];
```

	id	email	username	roles	password
<input type="checkbox"/> Éditer <input type="button" value="Copier"/> <input type="button" value="Supprimer"/>	1	administrateur@todolist.com	Administrateur	["ROLE_ADMIN"]	\$argon2id\$v=19\$m=65536,t=4,p=1\$3zamydFrkk+2OaUm/RG...
<input type="checkbox"/> Éditer <input type="button" value="Copier"/> <input type="button" value="Supprimer"/>	2	test2@test.fr	Test2	["ROLE_USER"]	\$argon2id\$v=19\$m=65536,t=4,p=1\$K67AoeqFhNZZ0uMEm6S...

Nom d'utilisateur

Mot de passe

Tapez le mot de passe à nouveau

Email

Rôle

Utilisateur

Register

## Limitations des accès

Grâce à cette gestion des rôles, nous pouvons ainsi configurer les [access\\_controls](#) de Symfony et ajouter des restrictions de pages en fonction des rôles.

Mais également, nous pouvons afficher des informations différentes en fonction du rôle de l'utilisateur avec lequel nous sommes connecté.

```
{% if is_granted('ROLE_ADMIN') %}
    <a href="{{ path('user_create') }}" class="btn btn-primary">Créer un utilisateur</a>
{% endif %}
```

## Token CSRF

Le paramètre « csrf\_token », ajouté au formulaire, mais non visible par les utilisateurs, est une clé de sécurité envoyé par le formulaire de connexion permettant de protéger le formulaire contre le « *Cross site request forgery* » ou contre les éventuelles attaques par des tiers.

## Injections SQL

La protection contre les injections SQL est faite au niveau de Doctrine et PDO, la protection reste donc la même après la mise à jour vers Symfony 4.

## Failles XSS

Les failles XSS sont protégées au niveau du moteur de Template Twig, la protection reste donc la même après la mise à jour du framework.

## Pages d'erreurs

Les pages d'erreur (403, 404, 500 et autres) sont les pages par défaut du Framework Symfony. Ces pages d'erreur peuvent être customisée afin de les faire correspondre avec l'affichage du reste de l'application.

## Fixtures

L'application ne dispose d'aucune fixture pour initialiser le contenu de la base de données avec des données de démonstration.

Des fixtures ont donc être créées, la génération de données « fake » se fera par l'utilisation de la dépendance **doctrine/doctrine-fixtures-bundle**.

Les fixtures sont générées par la librairie Doctrine fixtures, et sont stockées dans le dossier **src/DataFixtures**.

Le lancement de la création des fixtures se fait grâce à la commande Symfony **php bin/console doctrine-fixtures-load**.

## Tests

Aucun test unitaire ou fonctionnel n'était fait sur la première version d'application.

Afin de garantir la fiabilité et le bon fonctionnement de l'application lors de l'ajout de nouvelles fonctionnalités, il est nécessaire de créer tous les tests :

- Tests unitaires pour vérifier les différentes méthodes (Affichage de la page d'accueil, ajout d'un utilisateur)
- Tests fonctionnels pour tester le comportement des différentes fonctionnalités (Clic sur le bouton Créer un utilisateur, remplir un formulaire et le soumettre, puis vérifier que le message de succès s'affiche)

Les tests sont stockés dans le dossier /tests.

Une analyse du taux de couverture des tests doit être réalisée pour garantir que la majeure partie de l'application est testée. s

/Applications/MAMP/htdocs/Cours/Projets/Projet\_B/ToDoList/src / (Dashboard)

	Code Coverage			
	Lines		Functions and Methods	
Total	100.00%	120 / 120	100.00%	37 / 37
■ Controller	100.00%	63 / 63	100.00%	10 / 10
■ DataFixtures	n/a	0 / 0	n/a	0 / 0
■ Entity	100.00%	38 / 38	100.00%	24 / 24
■ Form	100.00%	19 / 19	100.00%	3 / 3
■ Migrations	n/a	0 / 0	n/a	0 / 0
■ Repository	n/a	0 / 0	n/a	0 / 0
■ Security	n/a	0 / 0	n/a	0 / 0
■ Kernel.php	n/a	0 / 0	n/a	0 / 0

### Legend

Low: 0% to 50%   Medium: 50% to 90%   High: 90% to 100%

Generated by php-code-coverage 6.1.4 using PHP 7.3.8 with Xdebug 2.9.6 and PHPUnit 7.5.20 at Fri Jul 31 13:33:32 UTC 2020.



## Autres améliorations

### Tri des tâches

Actuellement, il n'est pas possible de trier les tâches (par auteur, date de création, nom, ...)

Il pourrait donc être intéressant d'ajouter un champ de tri sous forme de liste déroulante.

### Liens HyperText

Dans cette première version de l'application, l'expérience utilisateur est très pauvre.

En effet, nous pouvons remarquer qu'il est assez compliqué d'accéder à l'entièreté des pages.

Pour cela, il serait appréciable de d'ajouter des liens HyperText vers les pages suivantes :

- Page d'accueil
- Liste des utilisateurs

### Règlement général sur la protection des données

Il faudra également ajouter une fonctionnalité très importante.

En effet, depuis le mois de mai 2018, il est nécessaire de rendre notre application conforme au règlement général sur la protection des données (RGPD).

Ceci nécessite donc d'afficher un bandeau ou un pop-up concernant l'acceptation et le paramétrage de gestion des cookies.

## Respect des normes

Le code de l'application semble prendre en compte le respect des normes de codage PSR-1, PSR-2 et PSR-4.

Cependant, nous pouvons remarquer que le code n'est pas suffisamment commenté :

- Commentaires des classes
- Commentaires des méthodes (fonction, paramètres, retour)
- Commentaires dans les méthodes

Il sera donc nécessaire d'ajouter ces commentaires de code afin d'améliorer la maintenabilité, la compréhension du code et la facilité de prise en charge par des futurs développeurs.

A noter que la norme PSR-2 se voit être remplacée par le PSR-12.

## Performances

L'application se verra évoluer rapidement, ce qui impliquera une augmentation de données, ainsi que de trafic.

Il est donc nécessaire, dès que possible, de surveiller les performances de l'application tout au long du développement de celle-ci.

Cela évitera de passer beaucoup de temps en fin de développement à se consacrer uniquement à l'optimisation de l'application.

## Analyse initiale

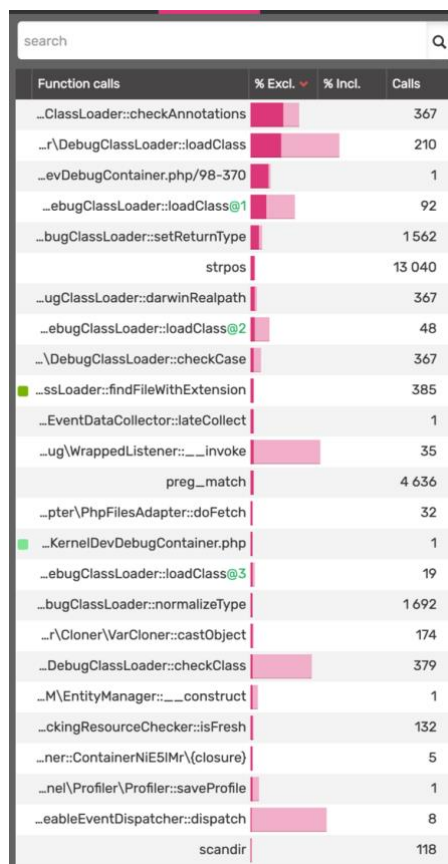
Comme évoqué plus haut dans ce document, la première version de l'application est une version permettant de montrer les principales fonctionnalités basiques, et cela implique donc que l'optimisation n'était pas la priorité.

L'analyse de performances avec l'application Blackfire n'a pas pu être effectué sur la version 3.1.6 du framework et cela pour plusieurs raisons.

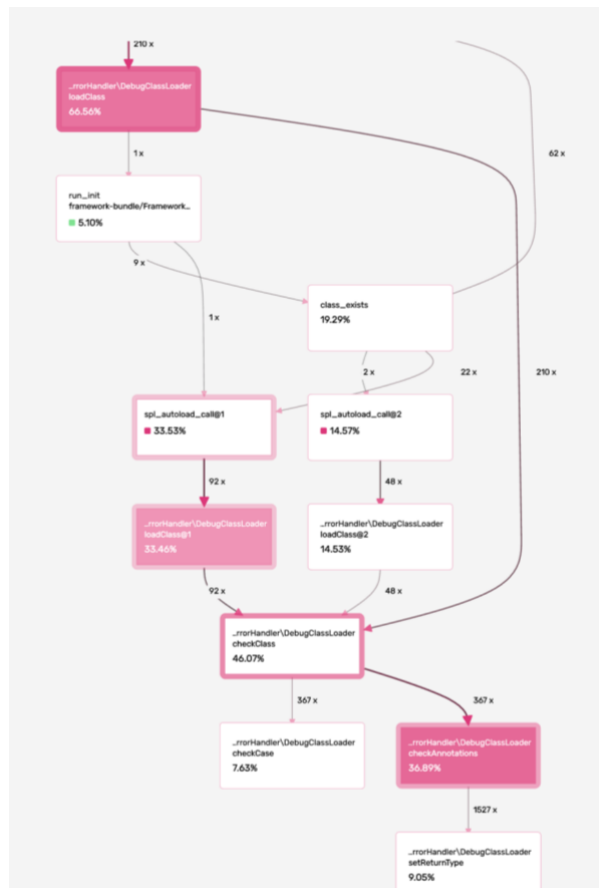
- La version de PHP n'est plus maintenue (5.5.9)
- Souci économique : En souhaitant effectuer l'analyse Blackfire de l'application, il sera nécessaire de souscrire à un abonnement payant de 29€/mois.

En savoir plus : <https://blackfire.io/pricing>

## Analyse après mise à jour



Function calls	% Excl. ▼	% Incl.	Calls
...ClassLoader::checkAnnotations			367
...r\DebugClassLoader::loadClass			210
...evDebugContainer.php/98-370			1
...ebugClassLoader::loadClass@1			92
...bugClassLoader::setReturnType			1 562
strpos			13 040
...ugClassLoader::darwinRealpath			367
...ebugClassLoader::loadClass@2			48
...l\DebugClassLoader::checkCase			367
...ssLoader::findFileWithExtension			385
...EventDataCollector::lateCollect			1
...ug\WrappedListener::__invoke			35
preg_match			4 636
...pter\PhpFilesAdapter::doFetch			32
...KernelDevDebugContainer.php			1
...ebugClassLoader::loadClass@3			19
...bugClassLoader::normalizeType			1 692
...r\Cloner\VarCloner::castObject			174
...DebugClassLoader::checkClass			379
...M\EntityManager::__construct			1
...ckingResourceChecker::isFresh			132
...ner::ContainerNIE5IMr\closure			5
...nel\Profiler\Profiler::saveProfile			1
...eableEventDispatcher::dispatch			8
scandir			118



Sur l'analyse ci-dessus, on peut voir que les performances de l'application sont principalement dues à une forte consommation de l'autoload de classes de Symfony. Nous pouvons également voir que l'autoload de classes est appelé plusieurs fois ; c'est donc ce qui est en cause du ralentissement des pages et cela nous donne un temps global de chargement d'environ 630 ms.

Pour chaque classe instanciée, l'autoloader va analyser les différents dossiers du projet, charger le fichier et inclure la classe correspondante, ce qui va créer un nombre d'appels très important de l'autoloader et donc ralentir le chargement de la page.

Dans la section suivante, nous verrons quelques méthodes permettant de réduire et optimiser les performances de l'application.

### Piste d'optimisation

Il est possible d'optimiser les performances de l'application.

Cela commence bien-sûr par une bonne qualité de code et le respect des normes de codage et les bonnes pratiques de Symfony, mais il est également possible d'améliorer les performances en ajoutant un ou plusieurs systèmes de cache.

- **OPcache :**

Ce système améliore les performances de PHP en stockant le bytecode des scripts pré-compilés en mémoire partagée, faisant ainsi qu'il n'est plus nécessaire à PHP de charger et d'analyser les scripts à chaque demande.

Cette extension est embarquée avec PHP 5.5.0 et suivants, et est [disponible via PECL](#) pour les versions 5.2, 5.3 et 5.4 de PHP.

Pour plus d'informations : <https://www.php.net/manual/fr/book.opcache.php>

- **Varnish :**

Varnish est un accélérateur d'applications Web également connu sous le nom de proxy inverse HTTP de mise en cache. Vous l'installez devant n'importe quel serveur qui parle HTTP et le configurez pour mettre en cache le contenu. Varnish Cache est vraiment très rapide. Il accélère généralement la livraison avec un facteur de 300 à 1000x, en fonction de votre architecture.

- **PHP-FPM :**

FPM (FastCGI Process Manager) est une implémentation alternative à PHP FastCGI avec quelques fonctionnalités additionnelles particulièrement utiles pour les environnements à haute charge.

Ces exemples peuvent être installés et configurés puisqu'ils sont prévus par Symfony et qu'une documentation d'installé est disponible dans la propre documentation de Symfony ou simplement implémenté directement dans PHP depuis la version 5.3.3.

- **OPcache:** <https://symfony.com/doc/4.4/performance.html>
- **Varnish:** [https://symfony.com/doc/4.4/http\\_cache/varnish.html](https://symfony.com/doc/4.4/http_cache/varnish.html)
- **PHP-FPM:** <https://www.php.net/manual/fr/install.fpm.php>

Nous pouvons aussi penser à créer, lors de chaque déploiement en production, un fichier statique listant les classes à charger au démarrage de l'application.

Pour cela, il est nécessaire de lancer la commande suivante :

***composer dump-autoload --no-dev --classmap-authoritative***