Qualité de code et dette technique

- **Problématique :**

Version initiale du projet : Symfony 3.1
Date de sortie : Mai 2016

- **Conséquences :**

Version non maintenue

- **Solution :**

Une migration à la version 3.4 était nécessaire.

- **Mise en place :**

Mettre à jour composer.json

```
"require": {
    "php": ">=5.5.9",
    "symfony/symfony": "3.4.*",
```

Mettre à jour le projet

```
$ composer update
```

Grace à la rétrocompatibilité de symfony notre projet est encore fonctionnel

---

- **Problématique :**

La gestion des autorisations d'accès est complexe.
ROLE_USER : Peut supprimer ses articles seulement.
ROLE_ADMIN : Peut supprimer ses articles ainsi que les articles qui ne sont attaché à un utilisateur.

- **Conséquences :**

Beaucoup de logique de vérification dans les contrôleurs et code répétitifs

```php
/**
 * @Route("/tasks/{id}/delete", name="task_delete")
 */
public function deleteTaskAction(Task $task)
{
    $logged_user = $this->getUser();
    $related_user = $task->getUser();
    if (($logged_user == $related_user) ||
     (is_null($related_user) && in_array("ROLE_ADMIN",$logged_user->getRoles()))){
        $em = $this->getDoctrine()->getManager();
        $em->remove($task);
        $em->flush();

        $this->addFlash('success', 'La tâche a bien été supprimée.');

        return $this->redirectToRoute('task_list');
    }

    $this->addFlash('error', 'Vous n\'êtes pas autorisé à supprimer cette tache.');
    return $this->redirectToRoute('task_list');
}
```

- **Solution :**

Utiliser les Security Voters pour gérer les droits d'accès.

- **Utilité :**

Délégué les contrôles d'autorisation d'accès à un service totalement dédié.

Un voter est un service qui sera appelé par la couche de sécurité de symfony au moment où l'on vérifiera les droits d'acces.
A l'issue de son exécution, le voter peut renvoyer une réponse parmi trois possibilités.

ACCESS_GRANTED: s'il autorise l'access.
ACCESS_DENIED: s'il refuse l'access.
ACCESS_ABSTAIN : s'il reste neutre

- **Mise en place :**

Définir une classe TaskVoter qui étend la classe Voter
Définir la logique qui doit être voter (le droit de supprimer une task)
Définir le voter en tant que service
Dans le contrôleur appeler le voter quand il y a besoin de vérifier les droits.

- **Problématique :**

Le formulaire d'authentification n'est pas protégé contre les attaques CROSS SITE REQUEST FORGERY

- **Conséquence :**

Faire exécuter des actions à l'insu de l'utilisateur.

- **Solution :**

Intégrer la protection CSRF du Security Component dans le formulaire de connexion.

- **Mise en place :**

- Activer la protection

```yaml
# app/config/config.yml
framework:
    # ...
    csrf_protection: ~
```

- Informer le security component du fournisseur des token CSRF

```yaml
# app/config/security.yml
security:
    # ...

    firewalls:
        secured_area:
            # ...
            form_login:
                # ...
                csrf_token_generator: security.csrf.token_manager
```

- Ajouter un champ invisible contenant le token CSRF

```twig
{# src/AppBundle/Resources/views/Security/login.html.twig #}

{# ... #}
<form action="{{ path('login') }}" method="post">
    {# ... the login fields #}

    <input type="hidden" name="_csrf_token"
        value="{{ csrf_token('authenticate') }}"
    >

    <button type="submit">login</button>
</form>
```

- Votre formulaire est désormais protégé.

- **Problématique :**

La liste des taches ou bien la liste des utilisateurs peut avec le temps devenir conséquente.

- **Conséquence :**

La récupération de la totalité des taches ou bien celle des utilisateurs peuvent ralentir notre application.

- **Solution :**

Recourir à des appels Ajax afin de récupérer un nombre limité à chaque fois. Et recourir à un système de pagination pour l'affichage.