**Documentation Technique**

**Authentification TODO & CO**

**1 – Utilisateurs :**

La gestion des utilisateurs se fait en base de données par le biais de l'entité "User.php".   
Les utilisateurs sont des instances de la classe "User" qui implémente l'interface "UserInterface" native de Symfony.  
Actuellement un utilisateur ("User") contient les attributs suivant :   
 - un identifiant unique : "id"  
 - un nom d'utilisateur unique : "username"  
 - un email unique : "email"  
 - un mot de passe : "password"  
 - une liste de rôles : "roles"

**2 – Création d'un utilisateur**

Pour la création d'un utilisateur nous faisons appel à la méthode "createAction" du contrôleur "UserController.php" via la route "user\_create" (url : "/users/create"). Un formulaire ("Form/Type/UserType.php") sera proposé au visiteur reprenant ainsi les attributs de la classe "User", avec une demande de répétition du mot de passe pour éviter l'erreur de frappe. Une fois le formulaire soumis le mot de passe est encodé grâce au service "security.password\_encoder" de Symfony pour l'enregistrement en base de données.

**3 – Edition d'un utilisateur**

Seul un utilisateur avec un rôle "ROLE\_ADMIN" peut avoir accès à cette catégorie.  
Il s'agit du même procéder que la création d'un utilisateur mise à part que le formulaire est pré-rempli des données de l'utilisateur en question, à une exception près les champs "mot de passe" sont vides, cela est évidement dû au cryptage de celui-ci en base de données.

**4 – Connexion :**

La connexion des utilisateurs se fait par la route "login" (url : "/login") appelant ainsi la méthode "loginAction" dans le contrôleur "SecurityController.php". Le formulaire ("views/security/login.html.twig") une fois validé par l'utilisateur, les données seront soumis à la sécurité du Framework Symfony.  
Si l'authentification s'effectue avec succès, le visiteur sera authentifié, dans le cas contraire l'utilisateur se verra renvoyé par Symfony sur le formulaire de connexion avec un message d'erreur d'authentification.

**5 – Sécurité :**

Le principe de base de sécurité Symfony :

**Client/Utilisateur 🡪 Pare-Feu 🡪 Contrôleur d'accès 🡪 Application**

Le **client** demande l'accès à l'**application.**  
Le **Pare-Feu** vérifie l'authentification du client, si OK il passe la main au **Contrôleur d'accès.**Sinon le **Pare-Feu** demandera une authentification.Le **Contrôleur d'accès** vérifie ce que l'**application** requiert comme rôle d'accès, si OK alors le **client** aura accès à l'**application.**  
Sinon le **Contrôleur d'accès** refusera l'accès à cause de droit insuffisant.

La configuration de sécurité se trouve dans le fichier "security.yml"

* **Encoders :**

**encoders:  
 AppBundle\Entity\User:** bcrypt

Nous définissons l'entité utilisé pour le "password" et quelle sera le système de hachage pour celui-ci.  
Le système "bcrypt" a été choisi car c'est l'un des plus sécurisé actuellement.

* **Rôle Hierarchy :**

**role\_hierarchy:  
 ROLE\_ADMIN:** ROLE\_USER

Cela permet de faire de l'héritage de rôle ainsi tous les utilisateurs avec le rôle "ROLE\_ADMIN" hériterons automatiquement du rôle "ROLE\_USER". Il est possible d'ajouter plusieurs roles à hériter.

* **Providers :**

**providers:  
 doctrine:  
 entity:  
 class:** AppBundle:User**property:** username

Nous définissons dans le providers la classe ("AppBundle:User ") et la propriété ("username") qui seront utilisé pour l'identification des utilisateurs.

* **Firewalls :**

**firewalls:** **main:** *# nom de notre pare-feu.*

**pattern:** ^/ *# Les routes commençant par "/" (le site entier) protégé par le pare-feu.* **anonymous:** ~ *# Les visiteurs non connecté peuvent accéder à l'application.* **form\_login:** *# Méthode d'authentification (via un formulaire)* **login\_path:** login *# Route pour le formulaire d'authentification* **check\_path:** login\_check *# Route pour la vérification des données du formulaire* **always\_use\_default\_target\_path:** true *# Utilise toujours la route par défaut* **default\_target\_path:** / *# Route par défaut* **logout:** ~ *# Autorise les déconnexions*

* **Access Control :**

**access\_control:** - { **path:** ^/login, **roles:** IS\_AUTHENTICATED\_ANONYMOUSLY }  
 - { **path:** ^/users/create, **roles:** IS\_AUTHENTICATED\_ANONYMOUSLY }  
 - { **path:** ^/users, **roles:** ROLE\_ADMIN }  
 - { **path:** ^/tasks, **roles:** ROLE\_USER }  
 - { **path:** ^/, **roles:** ROLE\_USER }

Les "access\_control" permettent de définir "qui a accès à quoi".  
 - Le "path" défini le chemin d'url sur laquelle nous voulons ajouter un contrôle.  
 - Les "roles" définissent à qui nous donnons l'autorisation d'accès.  
Grace au "role\_hierachy" défini plus haut, il n'est pas nécessaire d'ajouter le "ROLE\_ADMIN" pour le chemin d'url "^/tasks" ou "^/", le rôle "ROLE\_USER" étant hérité automatiquement.