Documentation Identification

- ## Général

L'architecture du projet est en MVC.
src

|_Controller	**templates**
|_DataFixtures	**|_default**
|_Entity	**|_security**
|_Form	**|_task**
|_Repository	**|_user**

- ## Entitée user

Voir et modifier l'entité : src/Entity/user.php
N'hésitez pas a regarder les différent diagramme dans le docier .doc avant de modifier l'entité.
Le controller lier se trouve src/Controller/UserController.php et SecurityControlleur. php pour la connection et déconnection.
Le formulaire de création src/Form/UserType.php
Il est impératif de laisser le $builder->addModelTransformer
Il est utile pour que les rôles en string soit retranscrit en tableau pour la DB

Commandes utile

```
php bin/console make:entity
php bin/console make:controller
php bin/console make:form
```

- ## Configuration

L'identification des users est géré par security-bundle de synfony.
https://symfony.com/doc/current/security.html
La configuration se fait dans config/package/security.yaml

L'utilisateur

Les autorisations dans Symfony sont toujours liées à un objet utilisateur, nommé User dans notre cas.
On le voit sur ligne 6 a 10

```
providers:
    app_user_provider:
        entity:
            class: App\Entity\User
            property: username
```

Le hachage du mot de passe se fait via PasswordAuthenticatedUserInterface, configurable ligne 40

Pare-feu et login

Le pare-feu définit quelles parties de votre application sont sécurisées et comment les utilisateurs pourront s'authentifier.
On peux le configurer dans la section `firewalls`

Contrôle des accés

Selon le rôle de l'user connecter (ROLE_ADMIN ou ROLE_USER), il peux avoir accés ou non a differente page.

```
access_control:
    - { path: ^/login, roles: PUBLIC_ACCESS }
    - { path: ^/users, roles: ROLE_ADMIN }
    - { path: ^/, roles: ROLE_USER }
```

dans notre cas
/login est visible par tous (même non connecter)
/users est uniquement visible par les administrateur (ROLE_ADMIN)
tout le site est visible par les administrateur et user (ROLE_ADMIN et ROLE_USER)
On peux voir l'héritage des rôles ligne 41

```
security:
    role_hierarchy:
        ROLE_ADMIN: ROLE_USER
```

- # Tests

Vérifier les tests après chaque changement.
Vous avez besoin de PHPUnit 9.5

Créé une base de données test (par défaut : ToDoList_test).
Insérer les fixtures.

Pour les tests, le dossiers /test est à la racine du projet. La couverture est dans public/test-coverage

Commandes utile

```
vendor/bin/phpunit
vendor/bin/phpunit --coverage-html public/test-coverage
```

Bon code !