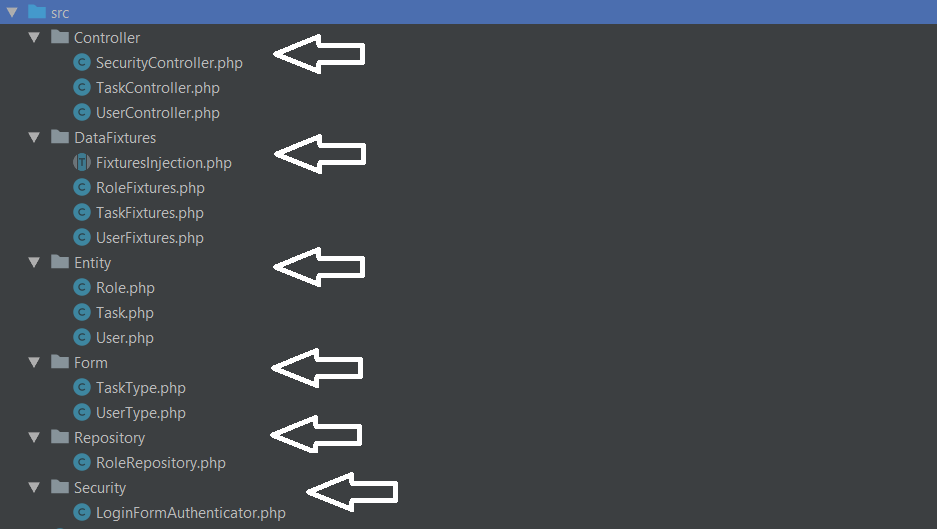
Documentation technique

**Architecture:**

Voici les dossiers/fichiers à connaître :

* **config** : Contiens tous les fichiers de configuration au format YAML.
* **templates** : Contiens les fichiers de Template en langage TWIG.
* **.env.\*** : Ce sont les fichiers de variables d’environnements.
* **tests** : Ce sont les fichiers PHP pour les tests unitaires et fonctionnels.
* **public**: Ce sont tous les assets (CSS/Images)
* **src** : contient tous les fichier PHP de l’application (basée sur le modèle MVC) :



Emplacement des contrôleurs (contiens les routes)

Contiens les classes de génération de données fictives de test

Contiens toutes les entités

Contiens tous les form builder

Contiens les requêtes SQL liées à une entité

Contiens la gestion du formulaire d'authentification

**Les ressources utilisées en front:**

* **Framework CSS bootstrap 4** : [https://getbootstrap.com/docs/4.3](https://getbootstrap.com/docs/4.3/getting-started/introduction/)
* **Fontawesome 5** pour la gestion des icônes : <https://fontawesome.com/icons>
* **Jquery 3** pour la gestion du DOM en Javascript: <https://api.jquery.com/>

**L’authentification :**

L’implémentation de l’authentification a été faite avec le composant Security natif intégré sur Symfony 4. Le **cryptage** des mots de passe est en **mode auto**. C’est-à-dire que Symfony prend le plus sécurisé automatiquement.

Le fichier de configuration « **/config/packages/security.yaml** »

security:  
 encoders: # Permet de configurer l'encodeur des mots de passe  
 App\Entity\User: # Cible l'entité utilisateur  
 algorithm: auto # Choix du cryptage  
  
 providers: # permet d'associer l'authentification une entité et définir un attribut principal  
 app\_user\_provider:  
 entity:  
 class: App\Entity\User # entité ciblée  
 property: username # attributs ciblées attribution utilisé pour l' identifiant de connexion  
  
 firewalls: # gestion globale du pare-feu  
 dev: # permet de charger les assets et la debugbar sans être connecté  
 pattern: ^/(\_(profiler|wdt)|css|images|js)/  
 security: false  
 main:  
 http\_basic: ~  
 anonymous: true # autorise la connexion en anonyme  
 guard:  
 authenticators: # classe de gestion du formulaire d'authentification  
 - App\Security\LoginFormAuthenticator  
 logout: #gestion de la déconnexion  
 path: logout # nom de la route pour se déconnecter  
  
 access\_control: # définit quel rôle peut accéder à quelle route  
 - { path: ^/login, roles: IS\_AUTHENTICATED\_ANONYMOUSLY }  
 - { path: ^/users, roles: ROLE\_ADMIN }  
 - { path: ^/, roles: [ROLE\_ADMIN, ROLE\_USER] }

Le fichier qui gère le formulaire d’authentification « **/src/Security/LoginFormAuthenticator.php »**

//détecte si c'est une requête de soumission formulaire d'authentification

supports(Request *$request*)  
  
//récupère les infos soumises par le formulaire de connexion  
getCredentials(Request *$request*)  
  
//récupération de l'entité utilisateur  
getUser(*$credentials*, UserProviderInterface *$userProvider*)  
  
//vérification du mot de passe  
checkCredentials(*$credentials*, UserInterface *$user*)  
  
//Récupération du mot de passe soumis  
getPassword(*$credentials*): ?string  
  
//redirection après succès de l'authentification  
onAuthenticationSuccess(Request *$request*, TokenInterface *$token*, *$providerKey*)

//récupère l'URL de connexion  
getLoginUrl()

Le contrôleur « **/src/Controller/SecurityController.php**».

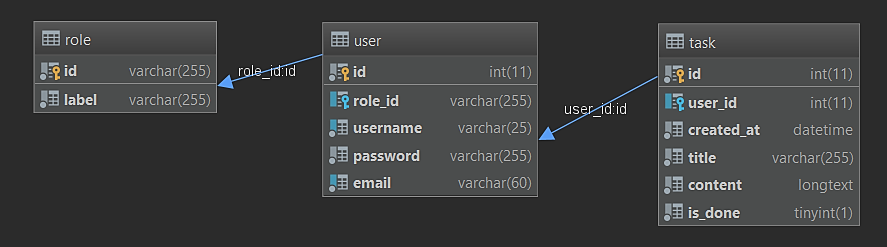
/\*\*  
 \* @Route("/login", name="login") // URL et nom de la route  
 \*/  
public function loginAction(AuthenticationUtils *$authenticationUtils*)  
{  
 $error = *$authenticationUtils*->getLastAuthenticationError() ;  
 $lastUsername = *$authenticationUtils*->getLastUsername();  
  
 return $this->render('security/login.html.twig', array(  
 'last\_username' => $lastUsername,  
 'error' => $error,  
 ));  
}

/\*\*  
 \* @Route("/logout", name="logout")  
 \*/  
public function logout() {}

Voici les **liens utiles** vers la documentation officielle: <https://symfony.com/doc/current/reference/configuration/security.html>

<https://symfony.com/doc/current/security/form_login_setup.html>

**La Structure SQL par rapport aux entités:**



La structure de la base est le reflet des classes entité dans le dossier « /src/Entity », cette conversion et possible grâce à l’ORM doctrine 2. Voici la documentation : <https://symfony.com/doc/current/doctrine.html>

**Les Données de test (fixtures) :**

Les fixtures sont des générateurs de données fictives. Elles permettent de tester l’application. Chaque fichier de fixture est lié à une entité. La documentation : <https://symfony.com/doc/master/bundles/DoctrineFixturesBundle/index.html>

* *Veuillez vous référer aux autres documents pour plus d'information.*