Documentation Technique :

1. *Authentification :*

- *L'authentification pour un système informatique est un processus permettant au système de s'assurer de la légitimité de la demande d'accès faite par un utilisateur et ça afin d'autoriser l'accès à certaines ressources du système. Dans notre application créer avec le framework symfony c'est le firewall qui se charge de l'authentification de notre application, sinon le firewall le redirigera vers la page connexion, la partie de l'application n'est pas gérer par le système de fireWall.*

```yaml
security:
    enable_authenticator_manager: true
    # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
    password_hashers:
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
    # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
    providers:
        # used to reload user from session & other features (e.g. switch_user)
        app_user_provider:
            entity:
                class: App\Entity\User
                property: username

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            lazy: true
            provider: app_user_provider
            pattern: ^/
            form_login:
                login_path: login
                check_path: login_check
                always_use_default_target_path:  true
                default_target_path:  /
            logout:
                path: logout

            # activate different ways to authenticate
            # https://symfony.com/doc/current/security.html#the-firewall
```

2. La class utilisateur(User) :

- *Avant toute chose, il est nécessaire d'avoir défini une entité qui représentera l'utilisateur connectée. Cette class doit implémenter de l'interface UserInterface, donc implémenter les différentes méthodes définis dans celle-ci. Dans ce cas-ci, cette class a déjà été implémentée et se situe dans le fichier « src/Entity/User.php ».*

```
class User implements UserInterface,
{
```

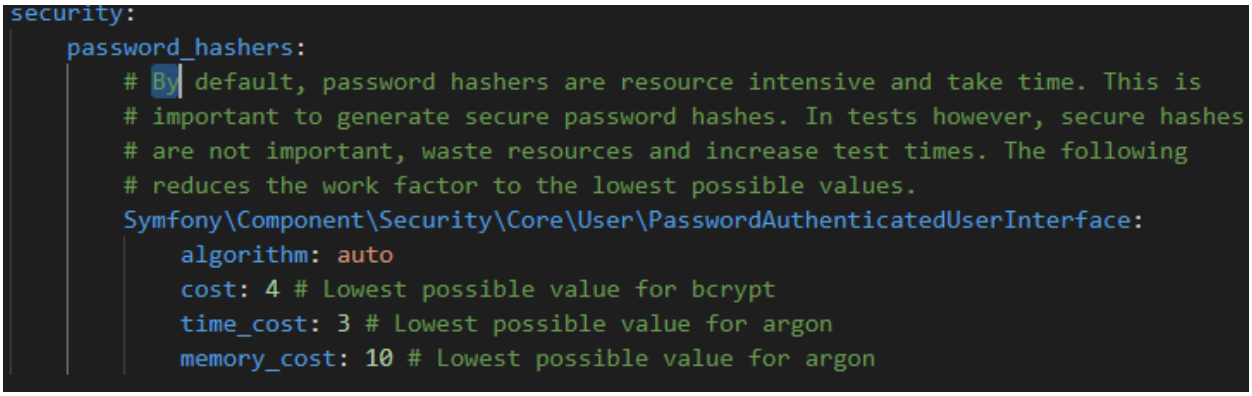
3. Les providers :

- *Un provider va nous permettre d'indiquer où se situe les informations que l'on souhaite utiliser pour authentifie l'utilisateur, dans ce cas-ci, on indique qu'on récupèrera les utilisateurs via Doctrine grâce à l'entité User dont la propriété username sera utiliser pour s'authentifier sur le site. Attention, on peut indiquer ici là class User car celle-ci implémente L'interface UserInterface.*

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

4. *Password hashers (hachage du mot de passe) :*

- *Hacheur du mot de passe va simplement nous permettre de déterminer quel est l'algorithme que l'on souhaite utiliser lors de du hachage du mot de passe. Dans ce cas-ci on utilisera l'algorithme :'auto' qu'est par défaut bcrypt, maintenant que l'on sait comment on veut hacher le mot de passe on peut utiliser le* `UserPasswordHasherInterface,` *service pour le faire avant d'enregistrer à la base de données.*



```yaml
security:
    password_hashers:
        # By default, password hashers are resource intensive and take time. This is
        # important to generate secure password hashes. In tests however, secure hashes
        # are not important, waste resources and increase test times. The following
        # reduces the work factor to the lowest possible values.
        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
            algorithm: auto
            cost: 4 # Lowest possible value for bcrypt
            time_cost: 3 # Lowest possible value for argon
            memory_cost: 10 # Lowest possible value for argon
```

5. *Les firewalls :*

- *Lors du processus d'authentification l'utilisateur utilise la page de login (formulaire de connexion) l'utilisateur peux ensuite renseigner son nom d'utilisateur et son mot de passe il est ensuite transmis vers le check_path pour vérifier l'identification de l'utilisateur en cas d'erreurs d'authentification le système réinvite l'utilisateur à s'authentifier et dans le cas d'un succès un token d'accès unique sera créer qui permet à l'utilisateur d'utiliser l'application.*

```
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    main:
        lazy: true
        provider: app_user_provider
        pattern: ^/
        form_login:
            login_path: login
            check_path: login_check
            always_use_default_target_path:  true
            default_target_path:  /
        logout:
            path: logout
```

6. *Formulaire de connexion:*

- *Le formulaire de connexion est défini dans le Controller SecurityController via la méthodes loginAction () c'est l'arborescence est celle par défaut dur le framework symfony, le système de sécurité est géré automatiquement via gestionnaire d'événements de symfony. En cas de succès, l'utilisateur sera authentifié. Dans le cas contraire le système redirigera l'utilisateur vers le formulaire de connexion pour le réinvité a ressaisie ces bonne infos d'authenfication.*

```php
/**
 * @Route("/login", name="login")
 */
public function loginAction(AuthenticationUtils $authenticationUtils) :Response
{
    $error = $authenticationUtils->getLastAuthenticationError();
    $lastUsername = $authenticationUtils->getLastUsername();

    return $this->render('security/login.html.twig', array(
        'last_username' => $lastUsername,
        'error'         => $error,
    ));
}
```

7. Le rendu du formulaire (vue twig) :

- *La vue twig sera afficher dans le navigateur pour que l'utilisateur puisse procéder à l'authentification via un formulaire de connexion (login.html.twig).*

```twig
{% extends 'base.html.twig' %}

{% block body %}
    {% if error %}
        <div class="alert alert-danger" role="alert">{{ error.messageKey|trans(error.messageData, 'security') }}</
    {% endif %}

    <form action="{{ path('login_check') }}" method="post">
        <label for="username">Nom d'utilisateur :</label>
        <input type="text" id="username" name="_username" value="{{ last_username }}" />

        <label for="password">Mot de passe :</label>
        <input type="password" id="password" name="_password" />

        <button class="btn btn-success" type="submit">Se connecter</button>
    </form>
{% endblock %}
```