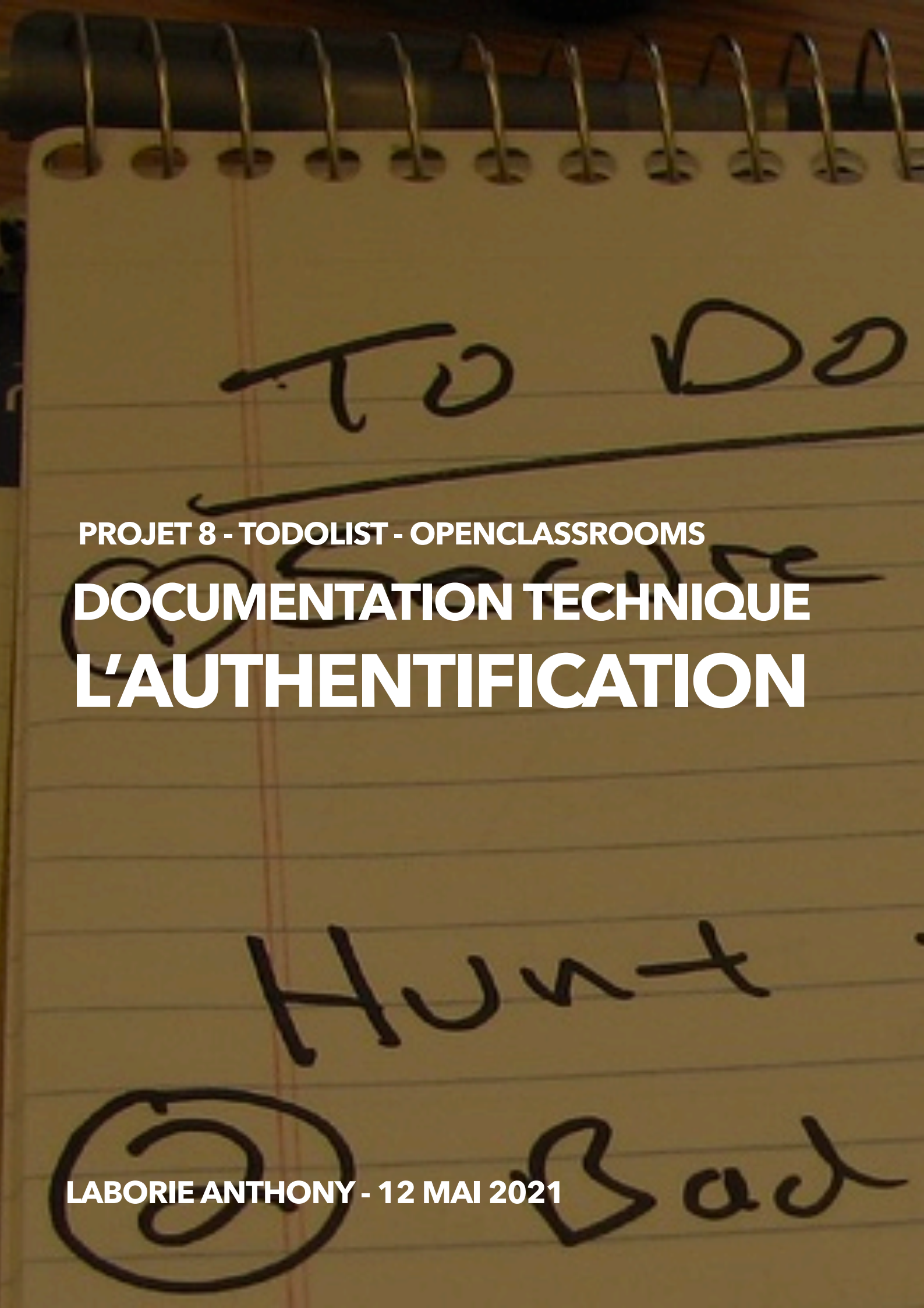

PROJET 8 - TODOLIST - OPENCLASSROOMS

DOCUMENTATION TECHNIQUE L'AUTHENTIFICATION

LABORIE ANTHONY - 12 MAI 2021

SOMMAIRE

INSTALLATION DU BUNDLE

Veuillez installer le bundle Symfony Security-Bundle.

Via Composer :

```
$ composer require symfony/security-bundle
```

CRÉATION DE L'ENTITÉ USER

Votre nouvelle entité doit obligatoirement implémentée **UserInterface** *(Symfony\Component\Security\Core\User\UserInterface)*.

Cette interface vous force à ajouter ces 5 méthodes :

- `getRoles()`
- `getPassword()`
- `getSalt()`
- `getUsername()`
- `eraseCredentials()`

CONSEIL *: Enumérez tous les rôles que vous souhaitez utiliser dans votre application en ajoutant des constantes dans cette classe. Vous pourrez ainsi éviter des fautes de frappes dans votre code.*

```
// ALL ROLES
const USER_ADMIN_ROLE = "ROLE_ADMIN";
const USER_USER_ROLE = "ROLE_USER";
```

Rendez-vous maintenant dans *config/packages/security.yaml* et modifiez les *encoders* et les *providers* :

```
encoders:
    App\Entity\[YOUR-NEW-ENTITY-NAME]: bcrypt

providers:
    doctrine:
        entity:
            class: App\Entity\[YOUR-NEW-ENTITY-NAME]
            property: [YOUR-PROPERTY]
```

La variable *[YOUR-PROPERTY]* correspond au champs unique des utilisateurs (l'username ou l'email par exemple). Pour cela, il faut lui affecter ***unique=true*** dans l'annotation *@ORM\Column*.

Pour faire tout ceci, vous pouvez aussi le faire directement dans votre terminal avec

```
$ php bin/console make:user
```

Ce script vous créera votre entité et son Repository et va mettre à jour le fichier *security.yaml*.

N'oubliez pas de faire une migration et la migrer dans votre base de données :

```
$ php bin/console make:migration
$ php bin/console doctrine:migrations:migrate
```

CRÉATION DES ROUTES POUR L'AUTHENTIFICATION

Créez un Controller que nous nommerons ici *SecurityController*.

Vous aurez trois routes à créer :

1 - Login Action :

```
/**
 * @Route("/login", name="login")
 * @param Request $request
 */
public function loginAction(Request $request)
{
      // Vous pouvez créer votre formulaire, récupérez les erreurs
      // Vous pouvez retourner une vue pour que l'utilisateur puisse s'identifier
      // Le formulaire doit être envoyé à la route login_check
}
```

2 - Login Check :

```
/**
 * @Route("/login_check", name="login_check")
 */
public function loginCheck()
{
      // Cette fonction ne sera jamais exécuté mais il est important de déclarer
      // la route
}
```

3 - Logout Check :

```
/**
 * @Route("/logout", name="logout")
 */
public function logoutCheck()
{
      // Cette fonction ne sera jamais exécuté mais il est important de déclarer
      // la route.
      // Cette route est créé pour permettre la déconnexion des utilisateurs
}
```

FINALISER LA CONFIGURATION

Rendez-vous sur *config/packages/security.yaml*.

Modifiez le *firewalls* comme ceci pour lier les routes créées aux actions d'authentifications :

```
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false

    main:
        anonymous: ~
        pattern: ^/
        form_login:
            login_path: login
            check_path: login_check
            always_use_default_target_path:  true
            default_target_path:  /
        logout: ~
```

Ensuite, nous allons définir les autorisations qu'il faut avoir pour accéder aux différentes routes de l'application. Pour ceci modifier le *access_control*.

```
access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
    - { path: ^/admin, roles: ROLE_ADMIN }
    - { path: ^/, roles: ROLE_USER }
```

Ici, comme nous l'indique la troisième règle, pour accéder à n'importe quelle route de l'application, il faut être un utilisateur avec le rôle ROLE_USER.

Mais n'importe quel utilisateur peut accéder à la page /login (règle 1).

Par contre, toutes les routes avec /admin seront interdites à tous utilisateurs sans le rôle ROLE_ADMIN (règle 3).