Aspects de Sécurité de l'Application "Great Danes"

# Introduction

Cette documentation décrit les mesures de sécurité mises en place dans l'application "Great Danes". Les aspects de sécurité sont divisés en deux sections : back-end et front-end.

# Sécurité du Back-end (Serveur)

## Authentification et Autorisation

- \*\*Formulaire de connexion\*\* : Utilisation de `form\_login` pour gérer l'authentification des utilisateurs.

```yaml  
security:  
 firewalls:  
 main:  
 form\_login:  
 login\_path: app\_login  
 check\_path: app\_login  
 username\_parameter: email  
 password\_parameter: password  
```

- \*\*Gestion des rôles\*\* : Utilisation des rôles (`ROLE\_USER`, `ROLE\_ADMIN`) pour contrôler l'accès aux différentes parties de l'application.

```yaml  
security:  
 access\_control:  
 - { path: ^/admin, roles: ROLE\_ADMIN }  
 - { path: ^/login, roles: IS\_AUTHENTICATED\_ANONYMOUSLY }  
 - { path: ^/register, roles: IS\_AUTHENTICATED\_ANONYMOUSLY }  
```

## Hachage des mots de passe

- \*\*Utilisation de UserPasswordHasherInterface\*\* : Pour hacher les mots de passe avant de les enregistrer dans la base de données.

```php  
private function hashPassword($entity): void  
{  
 if (!$entity instanceof User) {  
 return;  
 }  
  
 if (password\_get\_info($entity->getPassword())['algo'] === 0) {  
 $encodedPassword = $this->passwordHasher->hashPassword(  
 $entity,  
 $entity->getPassword()  
 );  
 $entity->setPassword($encodedPassword);  
 }  
}  
```

## Protection des Routes

- \*\*Accès restreint aux routes\*\* : Configuration des routes protégées pour s'assurer que seuls les utilisateurs autorisés peuvent y accéder.

```yaml  
security:  
 access\_control:  
 - { path: ^/admin, roles: ROLE\_ADMIN }  
```

## Validation des Entrées

- \*\*Contraintes de validation\*\* : Utilisation de contraintes Symfony pour valider les données d'entrée.

```php  
#[ORM\Column(type: 'string', length: 180, unique: true)]  
#[Assert\NotBlank]  
#[Assert\Email]  
private ?string $email = null;  
```

## Protection CSRF

- \*\*Protection CSRF intégrée\*\* : Les formulaires Symfony sont automatiquement protégés contre les attaques CSRF.

```twig  
{{ form\_start(form) }}  
 {{ form\_widget(form) }}  
 <button type="submit">Submit</button>  
{{ form\_end(form) }}  
```

## Vérification par email

- \*\*Confirmation de l'email\*\* : Envoi de liens de confirmation pour vérifier les adresses email des utilisateurs lors de l'inscription.

```php  
public function sendEmailConfirmation(string $verifyEmailRouteName, User $user, TemplatedEmail $email, UrlGeneratorInterface $urlGenerator): void  
{  
 // ...  
}  
  
public function handleEmailConfirmation(Request $request, User $user): void  
{  
 $this->verifyEmailHelper->validateEmailConfirmationFromRequest($request, $user->getId(), $user->getEmail());  
 $user->setIsVerified(true);  
}  
```

# Sécurité du Front-end (Client)

## Protection CSRF

- \*\*Protection CSRF dans les formulaires\*\* : Utilisation de jetons CSRF pour sécuriser les soumissions de formulaires.

```twig  
{{ form\_start(form) }}  
 {{ form\_widget(form) }}  
 <button type="submit">Submit</button>  
{{ form\_end(form) }}  
```

## Validation des Entrées

- \*\*Validation côté client\*\* : Utilisation de HTML5 et JavaScript pour valider les entrées des utilisateurs avant de les envoyer au serveur.

```html  
<input type="email" required>  
<input type="password" required minlength="6">  
```

## Messages d'Erreur

- \*\*Messages d'erreur sécurisés\*\* : Affichage de messages d'erreur génériques pour ne pas révéler d'informations sensibles.

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

## Politique de Sécurité de Contenu (CSP)

- \*\*En-têtes CSP\*\* : Mise en place de politiques de sécurité du contenu pour prévenir les attaques XSS.

```yaml  
framework:  
 session:  
 cookie\_secure: auto  
 cookie\_samesite: strict  
 csrf\_protection: true  
 csp:  
 report\_uri: '/csp-report-endpoint'  
 directives:  
 default\_src: 'self'  
 script\_src: 'self' 'trusted-cdn.com'  
```

## Politique de Référent (Referrer Policy)

- \*\*Referrer Policy\*\* : La politique de référent `no-referrer` et `strict-origin-when-cross-origin` est configurée pour améliorer la confidentialité et la sécurité des requêtes en contrôlant les informations de référent envoyées.

```yaml  
framework:  
 session:  
 cookie\_secure: auto  
 cookie\_samesite: strict  
 referrer\_policy:  
 enabled: true  
 policies:  
 - 'no-referrer'  
 - 'strict-origin-when-cross-origin'  
```