Procédure de cyber sécurité

Il est important d'empêcher n'importe qui d'accéder à toutes les routes s'ils ne sont pas connectés. Pour cela il faut paramétrer les routes et les rôles.

Dans config\packages\security.yaml :

```yaml
        # Easy way to control access for large sections of your site
        # Note: Only the *first* access control that matches will be used
        access_control:
            - { path: ^/login, roles: PUBLIC_ACCESS }
            - { path: ^/register, roles: PUBLIC_ACCESS }
            - { path: ^/, roles: ROLE_USER }
```

De plus dans les .html.twig, toutes les options réserver au Admin doivent être sécuriser pour que seul eux y est accès.

```twig
    {% if is_granted('ROLE_ADMIN') %}
        <h2>Add a God</h2>
        {{ form(form) }}
    {% endif %}
```

```twig
{% if is_granted('ROLE_ADMIN') %}
    <h3>Modifier</h3>
    {{form(edit)}}
{% endif %}
```

De plus, puisque l'option delete du crud ce fait via une route, on la sécurise avec un token crsf qui est vérifier par le backend.

```twig
    {% if is_granted('ROLE_ADMIN')%}
        <form action="{{ path('app_god_delete', {id: god.id}) }}" method="post">
            <input type="hidden" name="csrf" value="{{ csrf_token('delete' ~ god.id) }}">
            <button type="submit">Delete</button>
        </form>
    {% endif %}
```

```php
#[Route('/god/delete/{id}', name: 'app_god_delete')]
public function delete(EntityManagerInterface $em, God $god, Request $r)
{
    if (($this->isCsrfTokenValid('delete' . $god->getId(), $r->request->get('csrf')))) {
        $em->remove($god);
        $em->flush();
    }

    return $this->redirectToRoute('app_god');
}
```

Pour l'affichage individuel de chaque membre du panthéon, ont utilise une url dynamique avec une slug plutôt qu'un id pour plus de sécurité. Pour garantir l'unicité des slug on ajoute un random uniqid a la fin.

```php
#[Route('/god/{slug}', name: 'app_god_show')]
```

```php
$slug = $slugger->slug($god->getTitle() . '-' . uniqid());
$god->setSlug($slug);
```

```twig
<a href="{{ path('app_god_show', {slug: god.slug}) }}">
    see more...
</a>
```

Lors de l'enregistrement de l'utilisateur son mot de passe est haché.

```php
// encode the plain password
$user->setPassword(
    $userPasswordHasher->hashPassword(
        $user,
        $form->get('plainPassword')->getData()
    )
```

Touts les formulaires sont contrôler avant d'être envoyer en bdd.

```php
if ($form->isSubmitted() && $form->isValid()) {
```

Pour éviter les injections de balise html ou script que tous les champs de string ou de texte des Entity ont la méthode htmlspecialchars.

```php
public function setName(string $name): static
{
    $this->name = htmlspecialchars($name);

    return $this;
}
```

```php
public function setTitle(string $title): static
{

    $this->title = htmlspecialchars($title);

    return $this;
}
```

```php
public function setPicture(?string $picture): static
{
    $this->picture = htmlspecialchars($picture);

    return $this;
}
```

```php
public function setUsername(string $username): static
{
    $this->username = htmlspecialchars($username);

    return $this;
}
```

Quand on affiche un élément on n'oublie pas le |raw pour que ce ne soit pas interpréter.

```twig
<h1>{{ god.name|raw }}</h1>
<h2>{{ god.title|raw }}</h2>
```

```twig
<h3>{{ god.name|raw }}</h3>
<p>{{ god.title|raw }}</p>
```