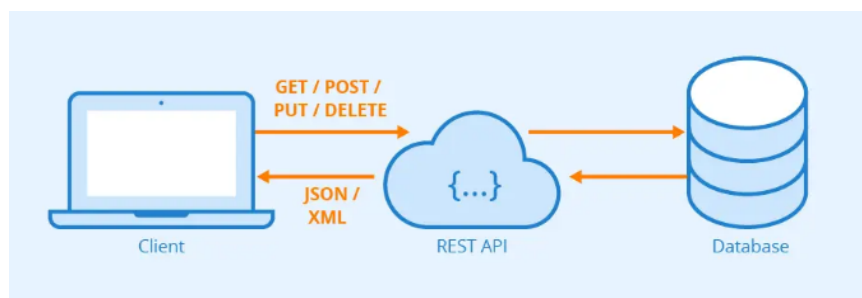


DOCUMENTATION TECHNIQUE

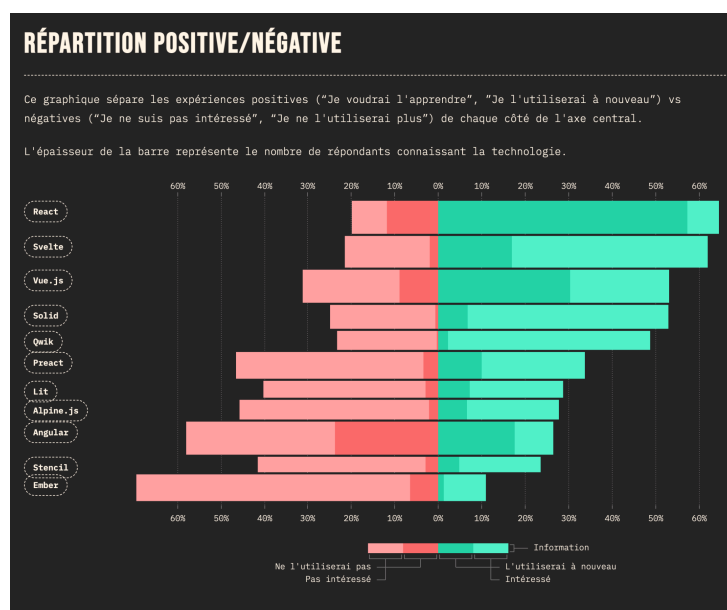
Choix technologiques

Pour réaliser le projet, j'ai choisi de développer une API (Application Programming Interface ou "Interface de programmation d'application") chargée de communiquer avec la base de données et de fournir les données demandées par le Client, le site internet Arcadia Zoo. Cette architecture correspond à une bonne pratique en matière de sécurité car le Client ne communique pas directement avec la base de données. Il consomme une API, qui sert d'intermédiaire pour le transfert des données.



Frontend

Pour le frontend, j'ai choisi le framework **React.js** qui offre une combinaison de performance et de flexibilité qui en fait un outils adapté pour le développement de Single Page Applications, avec une navigation fluide et rapide grâce à la gestion du DOM virtuel et à la mise à jour partielle des pages. Le tableau ci-dessous montre le grand intérêt que suscite React pour le développement d'applications web modernes.



Source : [Le monde de l'informatique](#)

Malgré toutes ses qualités React JS présente un problème pour l'indexation par les moteurs de recherche car la construction du HTML pour le rendu de la page web se fait côté client ce qui crée un temps de latence entre le chargement des premiers éléments et l'affichage du contenu. Le site affiche une page blanche pendant quelques instants ce qui nuit à la performance du SEO.

C'est pourquoi React recommande dans sa documentation l'utilisation d'un framework tel que **Next.js**, Remix ou Gatsby par exemple, qui permettent le rendu côté serveur (SSR). J'ai choisi d'utiliser Next.js qui semble être le plus populaire.

Next.js est un produit de Vercel qui offre une solution d'hébergement très performante et met à disposition des développeurs un espace web gratuit.

Par défaut, Next.js implémente **TypeScript**. J'ai choisi de maintenir l'utilisation de TypeScript qui est un langage très répandu en entreprise.

Pour le développement de styles, j'ai choisi d'utiliser **Tailwind CSS** qui offre une approche moderne et efficace, en mettant l'accent sur la productivité, la performance et la flexibilité.

Back-end

Pour le back-end, j'ai choisi d'utiliser le langage **PHP** qui est l'un des langages les plus populaires pour le développement web. De plus, c'est un langage open-source.

J'ai opté pour l'utilisation de **Symfony** est un ensemble de composants PHP ainsi qu'un framework MVC libre écrit en PHP. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web. Développé par une entreprise française, ce framework est très populaire et de grandes entreprises et organisations l'utilisent pour développer leurs applications web. Parmi elles, on trouve des noms comme BlaBlaCar, Spotify, et la BBC.

De plus, Symfony est l'option que j'ai choisie pour ma formation de développeur web.

Le back-end est chargé de fournir API, j'ai choisi d'utiliser **API Platform** qui est un framework web utilisé pour générer des API REST et qui s'intègre parfaitement à Symfony.

J'ai utilisé le système de gestion de bases de données relationnelles **MySQL** qui est le SGBDR le plus utilisé au monde. MySQL est disponible en version open source et est reconnu pour sa performance et sa capacité à gérer de grandes quantités de données.

Pour stocker les informations de fréquentation du site, j'ai utilisé **MongoDB** qui est un système de gestion de bases de données NoSQL qui a gagné une grande notoriété et popularité depuis sa création en 2009.

Pour le développement de styles, j'ai choisi d'utiliser **Tailwind CSS** et **Flowbite**.

A l'instar de Bootstrap, Flowbite fournit des modules prêt-à-l'emploi tels que le menu responsive, le slideshow...

Installation et configuration de mon environnement de travail

J'utilise le système d'exploitation Microsoft Windows 10 Professionnal et le plus souvent, j'utilise un PC portable ASUS K95VJ.

Pour la réalisation du projet Arcadia, j'ai vérifié que mon poste de travail était correctement configuré pour le développement d'une application en php avec Symfony et une application de type Single Page App.

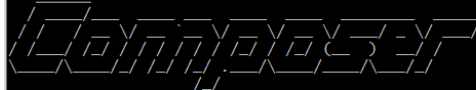
J'ai vérifié que Node.js était correctement installé

```
UTILISATEUR@DESKTOP-R385EEC MINGW64 ~  
$ node -v  
v20.10.0
```

J'ai vérifié que PHP était bien installé avec une version récente

```
UTILISATEUR@DESKTOP-R385EEC MINGW64 ~  
$ php -v  
PHP 8.3.0 (cli) (built: Nov 21 2023 17:48:00) (NTS Visual C++ 2019 x64)  
Copyright (c) The PHP Group  
Zend Engine v4.3.0, Copyright (c) Zend Technologies
```

J'ai vérifié que le gestionnaire de dépendances Composer était bien installé

```
UTILISATEUR@DESKTOP-R385EEC MINGW64 ~  
$ composer -v  
  
Composer version 2.6.6 2023-12-08 18:32:26
```

J'ai vérifié que la stack Apache / MySQL / PHP fournie par Wampserver était fonctionnelle



The image shows the Wampserver configuration page. At the top, there's a Wampserver logo and the text 'Wampserver'. Below it, it says 'Apache 2.4 - MySQL 5 & 8 - MariaDB 10 - PHP 5, 7 & 8'. On the right, there's a dropdown menu for 'Version 3.3.2 - 64bit' and another for 'french' and 'classic'. The main section is titled 'Configuration Serveur' and lists the following details:

- Version Apache : 2.4.58 - [Documentation Apache](#)
- Server Software : Apache/2.4.58 (Win64) PHP/8.2.13 mod_fcgid/2.3.10-dev - Port défini pour Apache : 80
- Version de PHP : [\[Apache module\]](#) 8.2.13 - [Documentation PHP](#) - Extensions PHP chargées - Utilisation versions PHP [FCGI] 7.4.33 - 8.0.30 - 8.1.26 - 8.2.13 - 8.3.0 - Aide mode FCGI
- Version de MySQL : 8.2.0 - Port défini pour MySQL : 3306 - SGBD par défaut - [Documentation MySQL](#)
- Version de MariaDB : 11.2.2 - Port défini pour MariaDB : 3307 - [Documentation MariaDB](#) - MySQL - MariaDB

J'ai vérifié que Git était bien installé

```
UTILISATEUR@DESKTOP-R385EEC MINGW64 ~  
$ git -v  
git version 2.43.0.windows.1
```

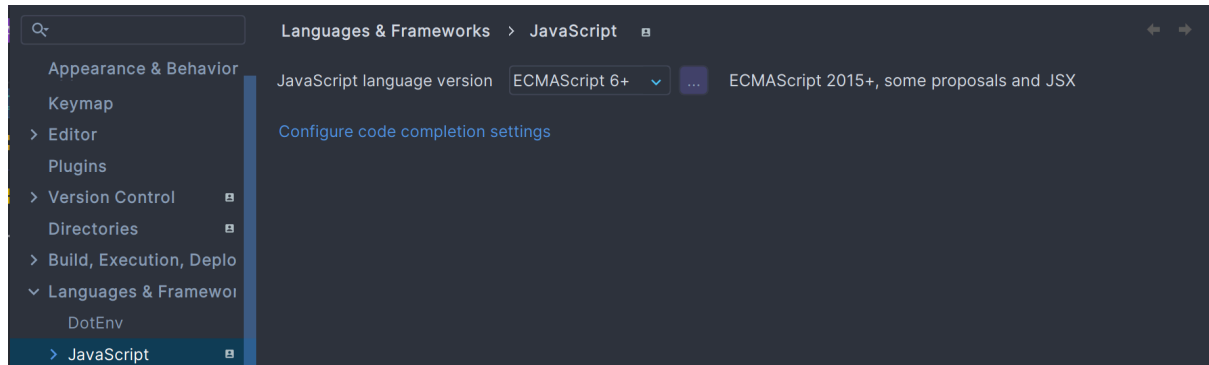
Symfony recommande l'utilisation de Symfony CLI, je l'ai donc installé après avoir installé Scoop qui est un prérequis nécessaire.

```
UTILISATEUR@DESKTOP-R385EEC MINGW64 ~  
$ symfony -v  
Symfony CLI version 5.8.19 (c) 2021-2024 Fabien Potencier (2024-05-10T07:24:31Z - stable)  
Symfony CLI helps developers manage projects, from local code to remote infrastructure
```

Environnement de développement intégré (IDE)

Pour ce projet j'ai utilisé l'éditeur de code **PHPStorm** qui offre un vrai confort d'utilisation et est particulièrement adapté au développement avec Symfony. De plus, PHPStorm intègre Git ainsi que des linters de vérification du code.

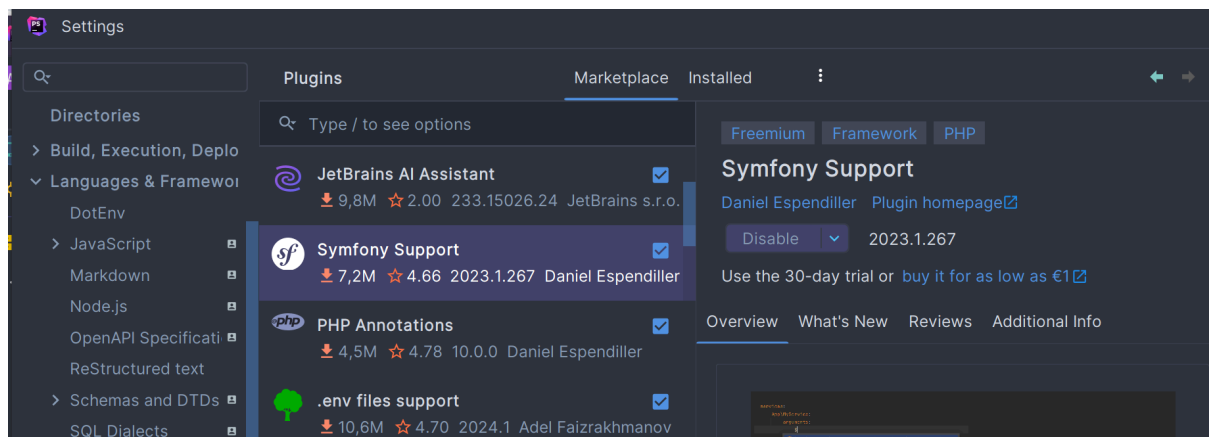
Concernant le javascript et pour obtenir une assistance au codage fiable et efficace, j'ai spécifié la version linguistique qui sera utilisée par défaut dans tous les fichiers JavaScript du projet.



Extensions

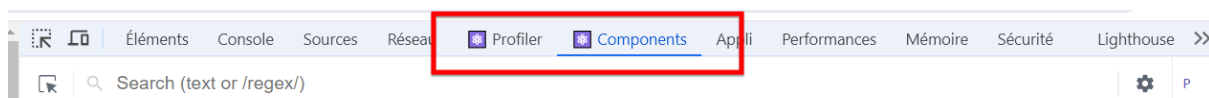
J'ai installé le plugin **Symfony Support** qui fournit une autocomplétion pour le code php mais aussi pour le Twig. De plus, le plugin réalise des imports automatiques de classe (use).

J'ai aussi installé le plugin **PHP Annotations** qui fournit une autocomplétion du code concernant les annotations en PHP.



Navigateur

Pour ce projet, j'ai principalement utilisé le navigateur **Chrome** qui inclut un outil de débogage **DevTools** très efficace. J'y ai ajouté l'extension **React Developer Tools** qui permet de déboguer et d'inspecter facilement les composants React.



Initialisation du projet

Front-end

Pour le front-end, j'ai initialisé le projet avec Next.js, en exécutant la commande :

```
npx create-next-app@latest
```

J'ai accepté par défaut l'ensemble des propositions de configuration

```
C:\Users\UTILISATEUR\Workflow>npx create-next-app@latest
√ What is your project named? ... arcadiazoo
√ Would you like to use TypeScript? ... No / Yes
√ Would you like to use ESLint? ... No / Yes
√ Would you like to use Tailwind CSS? ... No / Yes
√ Would you like to use `src/` directory? ... No / Yes
√ Would you like to use App Router? (recommended) ... No / Yes
√ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in C:\Users\UTILISATEUR\Workflow\arcadiazoo.

Using npm.

Initializing project with template: app-tw
```

j'ai créé un dépôt distant sur Github afin d'y stocker les différentes versions du code

```
echo "# Arcadia Zoo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/brunoturpeau/arcadiazoo.git
git push -u origin main
```

J'ai créé une branche de développement "dev"

```
git branch dev
git checkout dev
```

Back-end

Pour le back-end, j'ai initialisé un projet Symfony, en exécutant la commande :

```
composer create-project symfony/skeleton:"6.4.*"
arcadiazoo_backend
```

J'ai installé une partie des dépendances nécessaires au projet

```
composer require maker --dev
composer require api
composer require form
```

```
composer require debug --dev
```

J'ai installé **Tailwind** avec les commandes suivantes :

```
composer require symfonycasts/tailwind-bundle
symfony console tailwind:init
symfony console tailwind:build --watch
```

Enfin, j'ai installé **Flowbite** en exécutant la commande :

```
npm install flowbite
```

J'ai modifié le fichier de configuration tailwind.config.js

```
/** @type {import('tailwind css').Config} */
module.exports = {
  content: [
    './assets/**/*.js',
    './templates/**/*.html.twig',
    './templates/admin/**/*.html.twig',
    './node_modules/flowbite/**/*.js',
  ],
  thème: {
    extend: {},
  },
  plugins: [require('flowbite/plugin')],
}
```

J'ai ajouté le script js en CDN dans la partie head du fichier base.html.twig

j'y ai ajouté l'attribut *defer* afin de différer l'exécution à la fin du chargement du document

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>{% block title %}Arcadia Dashboard{% endblock %}</title>
    <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3.org/2000/svg%22 viewBox=%220 0 128 128%22>
    {% block stylesheets %}
    {% endblock %}
    <script defer src="https://cdn.jsdelivr.net/npm/flowbite@2.3.0/dist/flowbite.min.js"></script>
    <script defer src="{{ asset('assets/js/lib/ckeditor.js') }}" ></script>
    {% block javascripts %}
    {% block importmap %}{{ importmap('app') }}{% endblock %}
    {% endblock %}
  </head>
```

Modèle Conceptuel de Données

Pour réaliser le Modèle Conceptuel de Données (MCD) du projet, j'ai utilisé l'application en ligne dbdiagram.io qui utilise principalement la méthode Entity-Relationship Diagram (ERD) pour l'affichage des graphiques.

A l'aide du diagramme de cas d'utilisation j'ai créé une représentation abstraite de la structure des données d'un système d'information. Pour cela, j'ai identifié les entités qui auront des informations à gérer, j'ai associé à chaque entité les caractéristiques pertinentes (attributs) qu'elles possèdent et j'ai identifié les relations entre les entités (associations) et spécifié le type de relation (un-à-un, un-à-plusieurs, plusieurs-à-plusieurs).

dbdiagram.io utilise une syntaxe de type Markdown pour permettre aux utilisateurs de définir les entités et les relations.

```
Table report{
  id integer [primary key]
  date datetime [default: `now()`]
  detail text
  suggest text
  user_id int
  animal_id int
}
Ref: report.user_id > user.id
```



Modèle Conceptuel de Données (MCD) - Arcadia zoo

Sécurité

Les injections SQL

Les injections SQL (SQL Injection) sont une technique de piratage qui consiste à insérer du code SQL malveillant dans une requête SQL via une entrée utilisateur non sécurisée. Cette méthode permet à un attaquant de manipuler les requêtes SQL originales pour obtenir un accès non autorisé à la base de données, extraire des données sensibles, modifier des données ou même supprimer des tables entières.

Afin de limiter les risques d'injection, j'ai effectué une validation et un nettoyage des entrées utilisateur. Sur l'application, les points d'entrées sensibles sont le formulaire de contact et le formulaire de commentaire.

Pour valider les entrées, j'ai placé une fonction qui s'exécute lorsque qu'il y a un changement dans un champ

```
<label className={`block mt-10 mb-2`}
htmlFor="pseudo">Pseudo</label>
<input
  name={`pseudo`}
  onChange={handlePseudoChange}
  className={`w-full mb-5 ${statusPseudo === 'valid' ?
'bg-success ' : ''} ${statusPseudo === 'invalid' ? 'bg-danger
' : ''}`}
  type="text"
  required
/>
```

Pour échapper les données entrées par l'utilisateur j'ai utilisé DOMPurify qui nettoie le code HTML et empêche les attaques XSS.

```
function handleMessageChange(e) {
  const value = e.target.value
  if (value !== '') {
    setStatusMessage('valid')
    const cleanMessageHTML = DOMPurify.sanitize(value)
    setContentMessage(cleanMessageHTML)
  } else {
    setStatusMessage('invalid')
  }
}
```


Validation de l'adresse e-mail en utilisant un Regex

```
function handleEmailChange(e) {  
    const value = e.target.value  
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/  
    if (value.match(emailRegex)) {  
        setStatusEmail('valid')  
    } else {  
        setStatusEmail('invalid')  
    }  
}
```

L'utilisation d'une bibliothèque ORM (Object-Relational Mapping) comme Doctrine pour Symfony participe à réduire les risques d'injection car le système ORM va transformer les requêtes SQL en opérations sur des objets.

Dans le but de limiter l'impact d'une éventuelle injection, j'ai mis en place des restrictions pour les utilisateurs de la base de données. Dans le cas du zoo Arcadia, les visiteurs ne disposent pas de compte utilisateur, seuls les employé(e)s et les vétérinaires pourront se connecter et utiliser la base de données.

J'ai défini configuration suivante sur le fichier security.yaml :

Tous les utilisateurs ont accès au back-office.

Les employé(e)s et vétérinaires disposent de droits spécifiques et l'administrateur a tous les droits.

```
35     access_control:  
36         - { path: ^/admin, roles: [ROLE_USER] }  
37     role_hierarchy:  
38         ROLE_VETERINAIRE: ROLE_USER  
39         ROLE_EMPLOYE: ROLE_USER  
40         ROLE_ADMIN: [ROLE_EMPLOYE, ROLE_VETERINAIRE]  
41
```

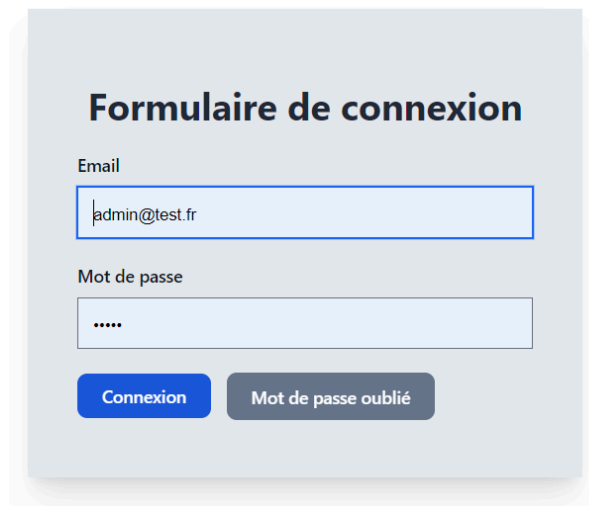
Dans ces conditions, tous les utilisateurs ont accès à toutes les routes de l'admin, il faut donc évaluer la nécessité de mettre une restriction pour chaque route en utilisant la méthode *denyAccessUnlessGranted* du bundle Security de Symfony.

Exemple d'une restriction sur une route permettant une suppression dans la base de données

```
#[Route('/{id}', name: 'app_animal_delete', methods: ['POST'])]
public function delete(Request $request, Animal $animal, EntityManagerInterface $entityManager): Response
{
    $this->denyAccessUnlessGranted('ROLE_ADMIN');
```

Mot de passe oublié

En matière de sécurité sur internet, il est conseillé de changer régulièrement de mot de passe. C'est pourquoi j'ai développé une fonction de réinitialisation du mot de passe.

Le formulaire de connexion est présenté sur un fond gris clair avec un effet d'ombre. Il est intitulé "Formulaire de connexion" en bold. Il contient deux champs de saisie : "Email" avec la valeur "admin@test.fr" et "Mot de passe" avec des pointsillés. En bas, il y a deux boutons : "Connexion" (bleu) et "Mot de passe oublié" (gris).

Le formulaire de connexion contient un lien “Mot de passe oublié” qui mène à un formulaire de réinitialisation.

Lorsque le formulaire est soumis, un token est généré et stocké dans la base de données. Parallèlement, un lien de réinitialisation est envoyé à l'utilisateur contenant le token d'une durée de validité de trois heures.

Le formulaire de réinitialisation du mot de passe est intitulé "Arcadia Zoo" et "Réinitialisation de votre mot de passe". Il se trouve dans une zone grisée et contient un champ "Adresse e-mail" et un bouton "Envoyer".

Lorsque le formulaire est soumis, le token du lien est comparé à celui stocké en base de données. Si il est valide, le mot de passe subit un hachage avant d'être enregistré.

Réinitialisation de mot de passe

Mot de passe

Envoyer

📌 Mot de passe changé avec succès

CSRF

Contrairement aux attaques XSS qui exploitent la confiance qu'un utilisateur a pour un site particulier, le CSRF exploite la confiance qu'un site a dans le navigateur d'un utilisateur. Symfony fournit une méthode `isCsrfTokenValid` qui va permettre de s'assurer que la requête est digne de confiance.

Par exemple, pour la fonctionnalité qui permet à un employé de valider ou d'invalider un commentaire j'ai créé un bouton sous forme de formulaire contenant un token csrf.

Les commentaires

Date	Pseudo	Commentaire	Statut	Publier	
02/07/2024	Famille Champion	Une expérience extraordinaire ! Le zoo offre un...	En attente	<input checked="" type="checkbox"/>	 
02/07/2024	Rocket 79	Quelle façon fantastique de visiter le zoo en f...	Publié	<input type="checkbox"/>	 

```
<form method="post" action="{ path('app_comment_publish', {'id': comment.id}) }"
      onsubmit="return confirm('Êtes-vous sûr de vouloir publier ce commentaire ?');"
>
  <input type="hidden" name="_token" value="{ csrf_token('publish' ~ comment.id) }" >
  <button>
    {{ include('_partials/icons/_check.html.twig') }}
  </button>
</form>
```

Le contrôleur vérifie la validité du token csrf avant la publication du commentaire.

```
#[Route('/{id}/publier', name: 'app_comment_publish', methods: ['POST'])]
public function publish(Request $request, Comment $comment, EntityManagerInterface $entityManager): Response
{
    $this->denyAccessUnlessGranted( attribute: 'ROLE_EMPLOYEE');
    if ($this->isCsrfTokenValid( id: 'publish'.$comment->getId(), $request->getPayload()->get( key: '_token'))){
        $comment->setVisible( is_visible: 1);
        $entityManager->persist($comment);
        $entityManager->flush();
        $this->addFlash( type: 'success', message: 'Commentaire publié avec succès.');
```

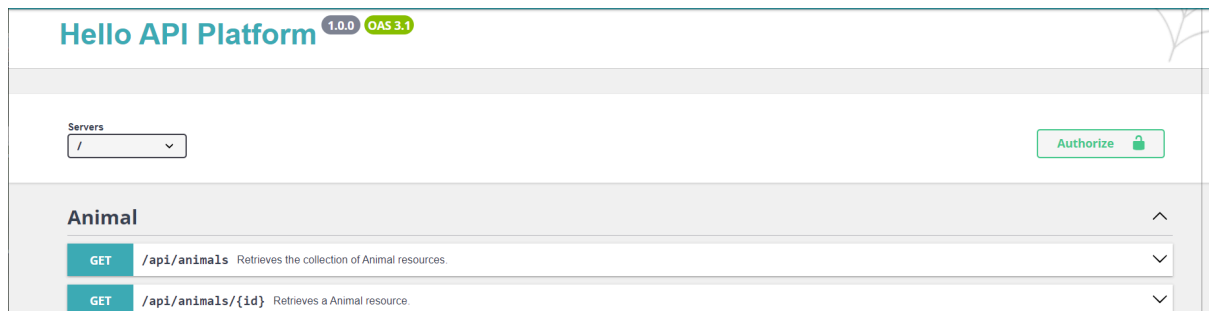
```
    }else{
        $this->addFlash( type: 'danger', message: 'Une erreur est survenue.');
```

```
    }
    return $this->redirectToRoute( route: '/admin/avis/', [], status: Response::HTTP_SEE_OTHER);
}
```

Exposition de l'API

Le Client, le site Arcadia, communique avec la base de données par l'intermédiaire d'une API, ce qui participe à réduire les risques d'une attaque sur la base de données.







En fonction de l'utilisation, seules les routes GET sont disponibles. Les routes DELETE, UPDATE, POST ne sont pas accessibles.



```
#[ORM\Entity(repositoryClass: AnimalRepository::class)]
#[ApiResource(
    operations: [
        new Get(),
        new GetCollection(),
    ]
)]
class Animal
{
    use SlugTrait;
    use CreatedAtTrait;
```

Hachage

Le mot de passe est haché et agit comme une empreinte numérique unique pour les données, ce qui permet de vérifier l'intégrité des données. Le mot de passe n'est pas stocké en clair dans la base de données.

<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	460	admin@test.fr	["ROLE_ADMIN"]	\$2y\$13\$BYU4bw0YjZmZ8zrvH6ykuSRrwmnrMQIHwm3hM4bfMr...	John
<input type="checkbox"/>	 Éditer	 Copier	 Supprimer	461	employee@test.fr	["ROLE_EMPLOYEE"]	\$2y\$13\$eO6lukru1DVl9A1wHmdZOU3LZPgpczQHJOTwnPF7H2...	Jean

Pour hacher un mot de passe avec Symfony il faut utiliser l'interface `UserPasswordHasherInterface`.

```
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\PasswordHasher\Hasher\UserPasswordHasherInterface;
use Symfony\Component\Routing\Attribute\Route;
```

```
$user->setPassword(
    $userPasswordHasher->hashPassword(
        $user,
        $form->get('plainPassword')->getData()
    )
);
```

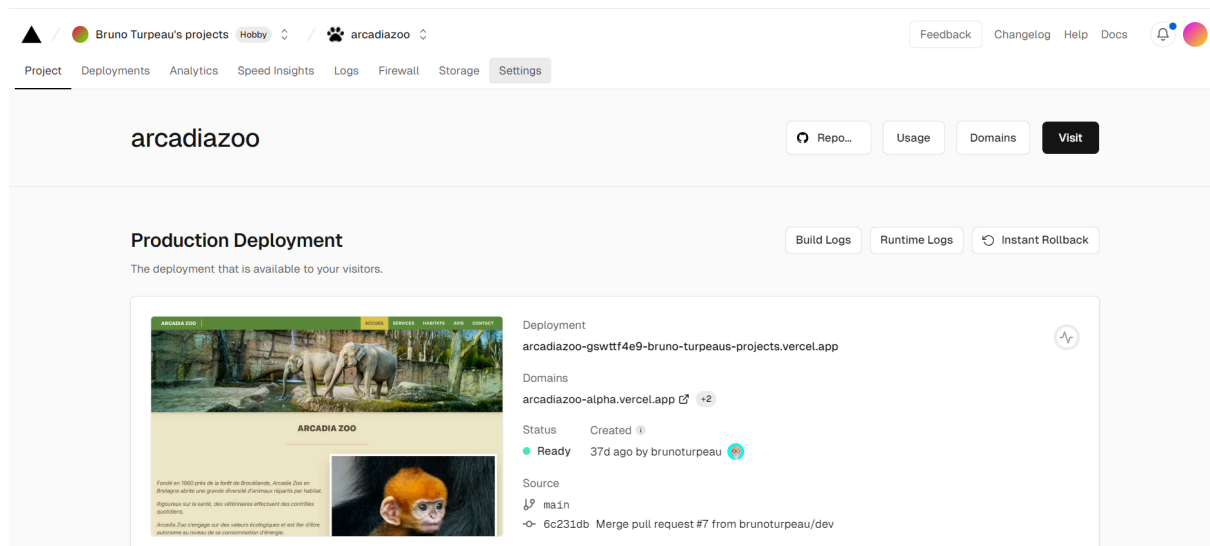
Il est possible de configurer le hasher via le fichier `security.yaml`. Par défaut, c'est l'algorithme SHA256 qui est utilisé.

```
security:
    password_hashers:
        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
```

Déploiement

Pour le déploiement de l'application web, j'ai choisi d'utiliser un hébergement gratuit qui est proposé par Vercel.

Vercel est une solution SaaS (Software as a Service) qui offre une intégration continue (CI) et un déploiement continu (CD) qui va permettre de redéployer à chaque push sur la branche main du dépôt de code. Chaque déploiement est automatiquement sécurisé avec SSL, garantissant que les communications entre les utilisateurs et les applications sont chiffrées.



Lors de la configuration du domaine, Vercel crée un script d'action lié au dépôt de code sur Github.

Veille technologique

Pour assurer une veille technologique sur les vulnérabilités de sécurité, j'ai privilégié le RSS en utilisant l'application Feeder.



J'ai choisi de suivre :

L'ANSSI, Agence nationale de la sécurité des systèmes d'information.

Schneier on security, blog qui vise à fournir des analyses et des commentaires sur divers sujets liés à la sécurité informatique et à la cryptographie.

Info²Sec, site internet dont le but est de fournir des informations, des analyses et des ressources sur la sécurité informatique pour aider les professionnels et les amateurs à se tenir informés des menaces et des meilleures pratiques en matière de cybersécurité.

Microsoft Security Blog qui vise à informer les utilisateurs et les professionnels de la sécurité sur les dernières actualités, tendances, et meilleures pratiques en matière de cybersécurité, ainsi que sur les nouvelles solutions et mises à jour de sécurité proposées par Microsoft.

M.G.M. Solutions, blog en français visant à fournir des services et des conseils en matière de sécurité informatique, en mettant l'accent sur la protection des données et la gestion des risques pour les entreprises et les particuliers.

Seclists.org qui a pour but d'héberger et d'archiver des listes de diffusion et des forums de discussion sur des sujets liés à la sécurité informatique, offrant ainsi une plateforme pour l'échange d'informations et la collaboration entre professionnels et chercheurs en cybersécurité.