Document élève
QuackNet

Le projet “**QuackNet**” a vocation à être utilisé comme base à la réalisation de différents projets en PHP.

Ce document décrit les attentes fonctionnelles autour de cette interface. Il n’a pas vocation à être exhaustif et les fonctionnalités décrites dans ce document peuvent être complétées.

But du projet et sources d’inspiration

Le but de ce projet est de réaliser une interface de réseau social “twitter-like” en ligne. Cette interface doit permettre à un canard de publier des *coincoins* en ligne.

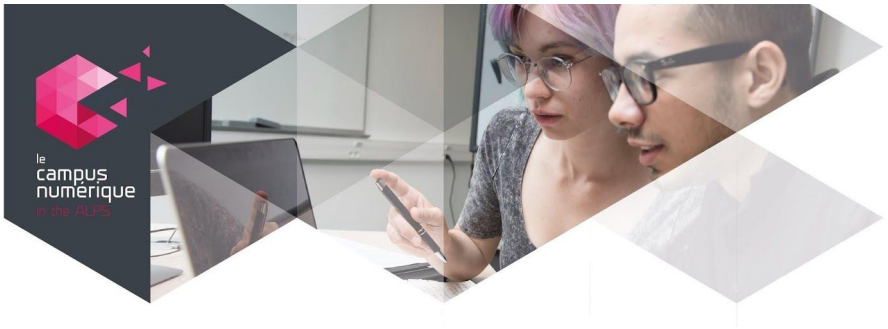
Un “coincoin” est une publication.
Les “canards” sont les utilisateurs.

On peut comparer ce projet à un réseau social “twitter-like”.

Exemples:
https://www.twitter.com
https://www.facebook.com

Contraintes du projet

- Équipe de travail : binôme/seul
- Durée du projet : 9 jours
- Contraintes techniques : utilisation du framework Symfony



Ressources

Symfony :

- Symfony : getting started
- Symfony 4.0 learning guide (un peu outdated mais toujours utilisable)

Doctrine :

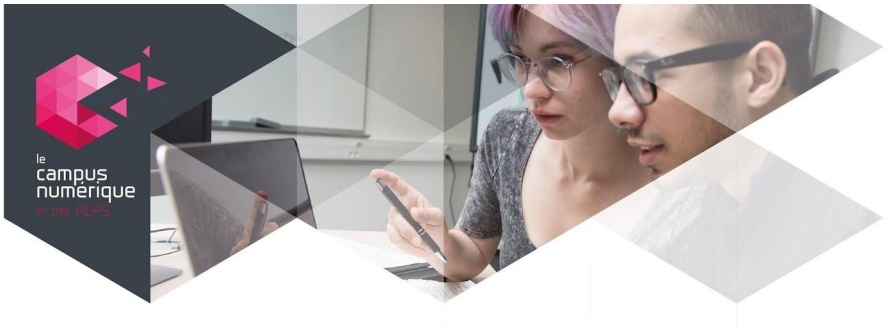
- Doctrine ORM documentation
- Symfony's doctrine documentation

Twig :

- Twig documentation
- Symfony's twig usages

Autre :

- ressources graphiques



Etape 0 - Installation et prise en main du Framework

La première partie du projet consiste à installer et à prendre en main le framework symfony.

Après avoir installé le framework, créer une entité **Quack** définie par:

- un contenu (content : text)
- une date de publication (created_at : datetime)

Créez ensuite un contrôleur qui permet de :

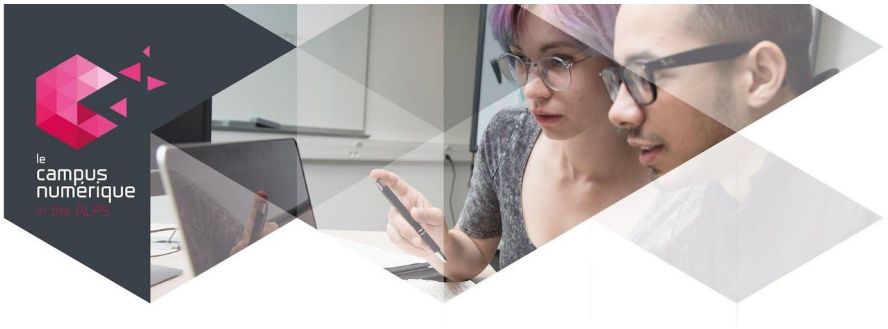
- lister tous les Quacks
- créer un Quack
- modifier un Quack
- supprimer un Quack

Créez les vues associées aux différentes actions.

Ajoutez les outils de développement du projet : le serveur intégré et la barre de debug.

Compétences

- Symfony 4 : Installer le framework
- Symfony 4 : Créer un controller
- Symfony 4 : Créer un formulaire et afficher le contenu du post
- Symfony 4 : connaître l'arborescence
- Symfony 4 : Créer un template twig qui utilise des variables
- Symfony 4 - Doctrine: Générer une entity
- Symfony 4 : Créer une route en mode annotation
- Symfony 4 : Utilisation des outils de développement
- Symfony 4 : Fonctionnement en production



Etape 1 : Inscription et authentification

La lecture des coincoins est ouverte à tous.

Le site doit permettre à un tous les visiteurs de s'enregistrer en ligne via un formulaire, il devient alors un *canard* du site

Un canard (table ducks) dispose :

- D'un Prénom (firstname)
- D'un Nom (lastname)
- D'un nom de canard (unique : duckname)
- D'une adresse email (unique : email).
- D'un mot de passe (password).

Un canard doit pouvoir se connecter au site avec son email ou son duckname et son mot de passe. Une fois connecté, le canard doit pouvoir mettre à jour son Prénom, Nom et mot de passe.

Compétences

- Symfony 4 : Mettre en place l'authentification et autorisation nativement
- Symfony 4 : Utilisation des outils de développement

Etape 2 : Les coincoins

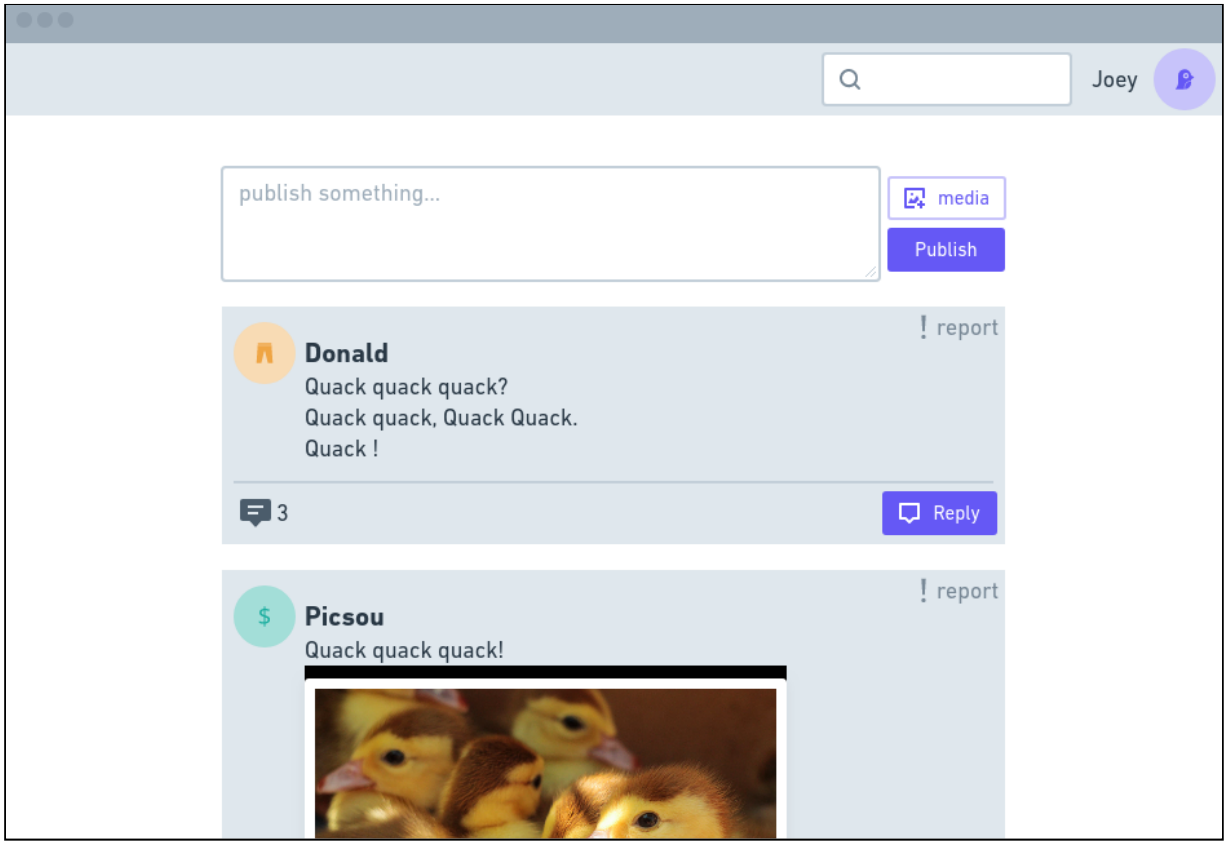
Le canard publie un coincoin. Pour créer un coincoin le canard doit être connecté au site. Un “**quack**” est composée de :

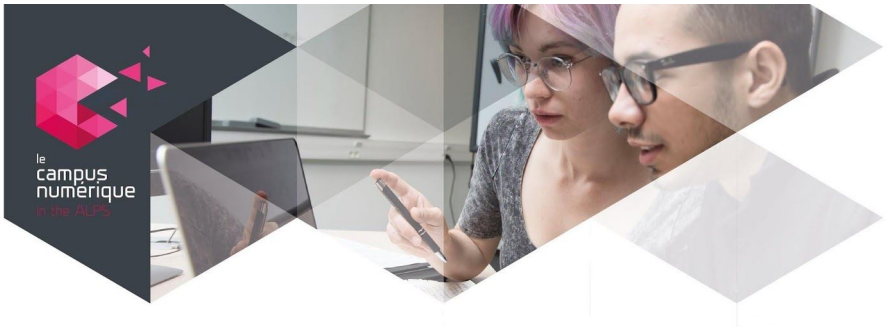
- Un message
- Un auteur
- Une photo
- Une date de publication
- Une série de tags

Les coincoins sont accessibles à tous mais la possibilité de commenter les coincoins est réservée aux canard connectés.

Seul l’auteur de la publication peut supprimer. Il peut consulter son profil et celui des autres.

En reprenant l’étape initiale, modifiez les entités et les vues correspondantes





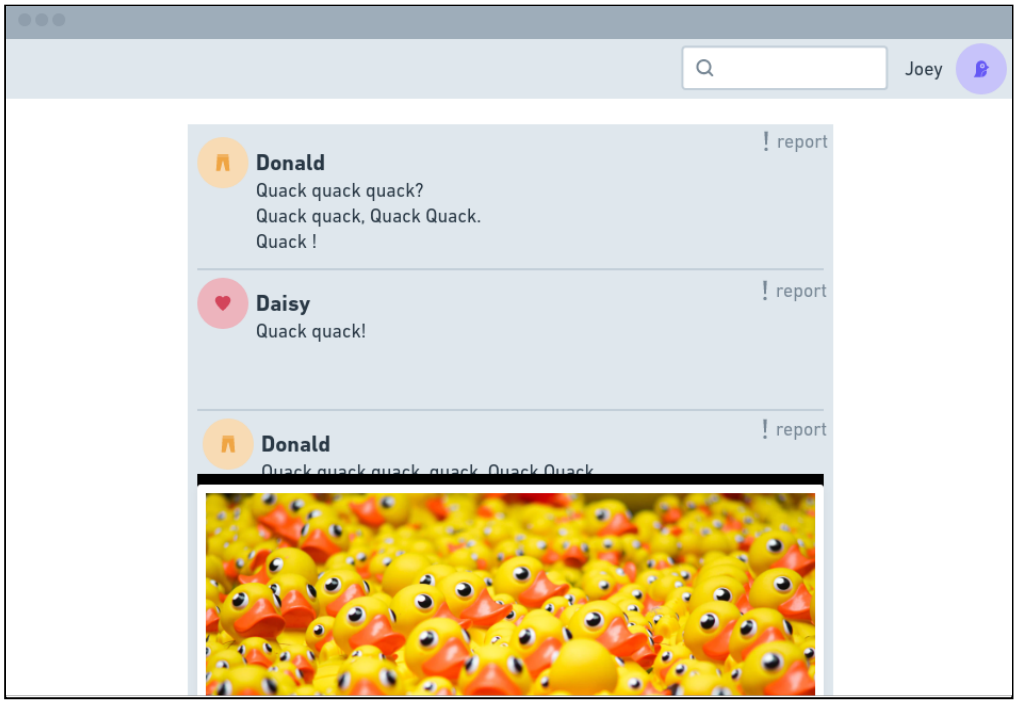
Compétences

- Symfony 4 : Mettre en place l'authentification et l'autorisation nativement
- Symfony 4 : Créer un controller
- Symfony 4 : Créer un formulaire et afficher le contenu du post
- Symfony 4 : connaître l'arborescence
- Symfony 4 : Créer un template twig qui utilise des variables
- Symfony 4 - Doctrine: Générer une entity
- Symfony 4 : Créer une route en mode annotation
- Symfony 4 : Utilisation des fonctions de twig d'extension et d'inclusion
- Symfony 4 : Utilisation des outils de développement

Etape 3 : Commentaires

Un canard peut envoyer un commentaire sur un “coincoin”. lui et l’auteur du coincoin originel sont les seuls à pouvoir supprimer ce commentaire.

Peut-être, peut-on imaginer que le commentaire soit juste un autre “coincoin” lié au premier, par un lien parent enfant.

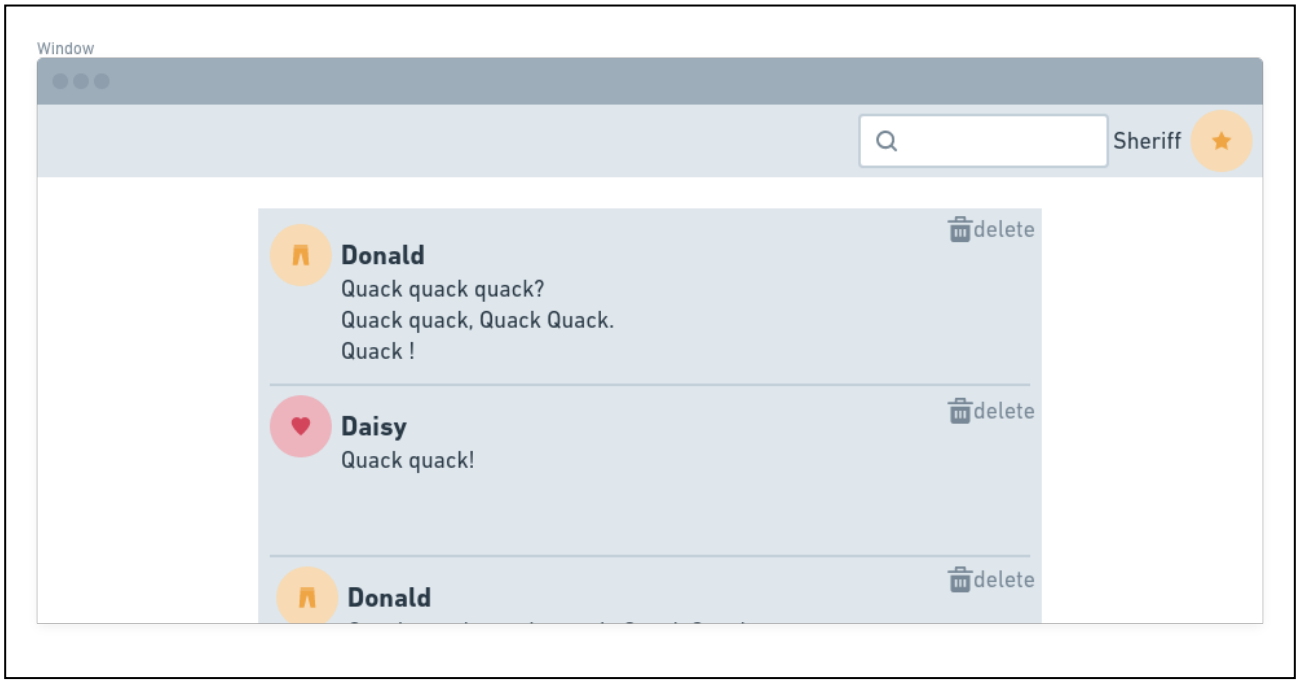


Compétences

- Symfony 4 : Mettre en place l'authentification et l'autorisation nativement
- Symfony 4 - Doctrine: Générer une entity
- Symfony 4 : Créer une relation entre 2 entity et l'afficher dans une page
- Symfony 4 : Utilisation des outils de développement

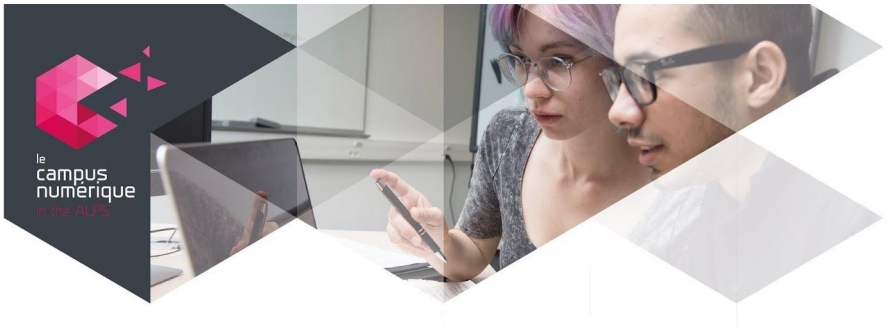
Etape 4 : Modération

Un canard peut être modérateur. Il a le droit de supprimer (masquer) des coincoins qui sont jugées inappropriées. Dès lors, les coincoins ne sont plus visibles sur la plateforme.



Compétences

- Symfony 4 : Mettre en place l'authentification et l'autorisation nativement
- Symfony 4 : Utilisation des outils de développement



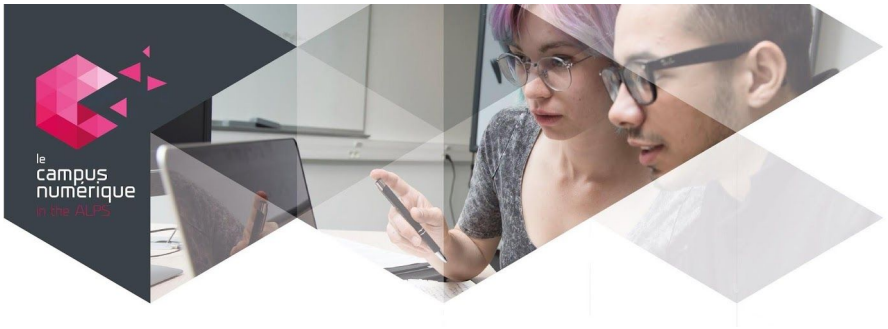
Etape 5 : Recherche

Un canard peut trouver des coincoins par le duckname de l'auteur, ou par tags.
La recherche par mot clé se fera par exemple sous la forme d'une recherche dite "full text":

```
SELECT * FROM quacks WHERE duckname LIKE '%keyword%'
```

Compétences

- Symfony 4 : Utiliser doctrine pour connecter une BDD et afficher le contenu d'une table



Etape 6 : Approfondissement

Le site “QuackNet” devra être accessible via une API.

L’authentification se fait en Basic Auth mais si vous le souhaitez, il est possible de la rendre plus complexe.

Annonces

- Récupérer la liste des derniers coincoins :
  `GET /api/quack`
- Récuperer un coincoin avec son image :
  `GET /api/quack/{id}`
- Rechercher des coincoins par tag :
  `GET /api/quack/search/?q={tag}`
- Supprimer **son** coincoin :
  `DELETE /api/quack/{id}`

Utilisateur

- S’inscrire :
  `POST /api/duck`
- Modifier son compte :
  `PUT /api/duck/{id}`
- S’authentifier/récupérer ses informations :
  `GET /api/whoami`