



Rapport de Projet Web : Application de Gestion et Annotation d'Images



Présentation Générale

Ce projet a été réalisé dans le cadre du cours de Programmation Web. L'objectif était de concevoir une application web complète permettant à un utilisateur de :

- créer un compte personnel,
- se connecter de manière sécurisée,
- envoyer des images sur le serveur,
- et enfin annoter ces images.

L'annotation attendue devait permettre à l'utilisateur de **sélectionner une zone précise de l'image** (comme un visage sur une photo) et d'y associer une description textuelle.

L'image annotée devait ensuite pouvoir être consultée, avec les annotations visibles et modifiables par l'utilisateur.



Structure du Projet

Le projet repose sur une architecture **Dockerisée**, composée de deux conteneurs :

- Un **serveur web**, basé sur PHP avec le serveur intégré (`php -S`), monté dans le conteneur via Docker.
- Une **base de données MySQL 5.7**, initialisée automatiquement via un script SQL.



Fichiers clés

- `docker-compose.yml` : définit les deux services (web et base de données), leurs ports, volumes, et variables d'environnement.
- `Dockerfile` : configure l'environnement PHP.
- `init.sql` : script de création des tables `users`, `images`, `annotations`.
- `public/` : contient les fichiers PHP accessibles par l'utilisateur (authentification, upload, visualisation).

- **includes/** : regroupe les fonctions PHP partagées (connexion à la base, gestion de session, sécurité...).

Parcours utilisateur

L'application permet à un utilisateur de :

- Créer un compte avec une adresse mail, un mot de passe sécurisé et un pseudo.
- Se connecter à son espace personnel.
- Uploader une ou plusieurs images, qui seront stockées sur le serveur.
- Ajouter une **description textuelle** à chaque image.
- Visualiser ses images dans une **galerie personnelle**.



Fonction de description (alternative à l'annotation)

Initialement, on avait prévu de permettre à l'utilisateur de **sélectionner une zone sur l'image avec la souris**, comme un rectangle, et d'y écrire un commentaire (par exemple sur un visage ou un objet). Cette fonctionnalité, bien pensée et commencée, **n'a pas pu être finalisée techniquement**.

À la place, **nous avons mis en place une alternative fonctionnelle** : un système de **description libre pour chaque image**.

- L'utilisateur peut saisir une description globale lors de l'upload.
- Cette description est enregistrée dans la base et affichée lors de la consultation.
- Cela permet de donner un contexte ou une explication à l'image, comme une légende.

! Difficultés rencontrées



1. Mise en place de l'environnement Docker

L'une des premières difficultés a été de faire fonctionner correctement Docker avec deux services qui devaient communiquer entre eux (le web et la base de données).

On a dû notamment :

- configurer correctement le nom d'hôte MySQL (**db**) pour qu'il soit accessible depuis PHP,
- **synchroniser le démarrage** des conteneurs (avec un délai pour attendre MySQL),
- et gérer les **droits d'accès aux volumes** et aux ports.

2. L'annotation graphique (zone + texte)

C'était sans doute la **fonction la plus ambitieuse** du projet.

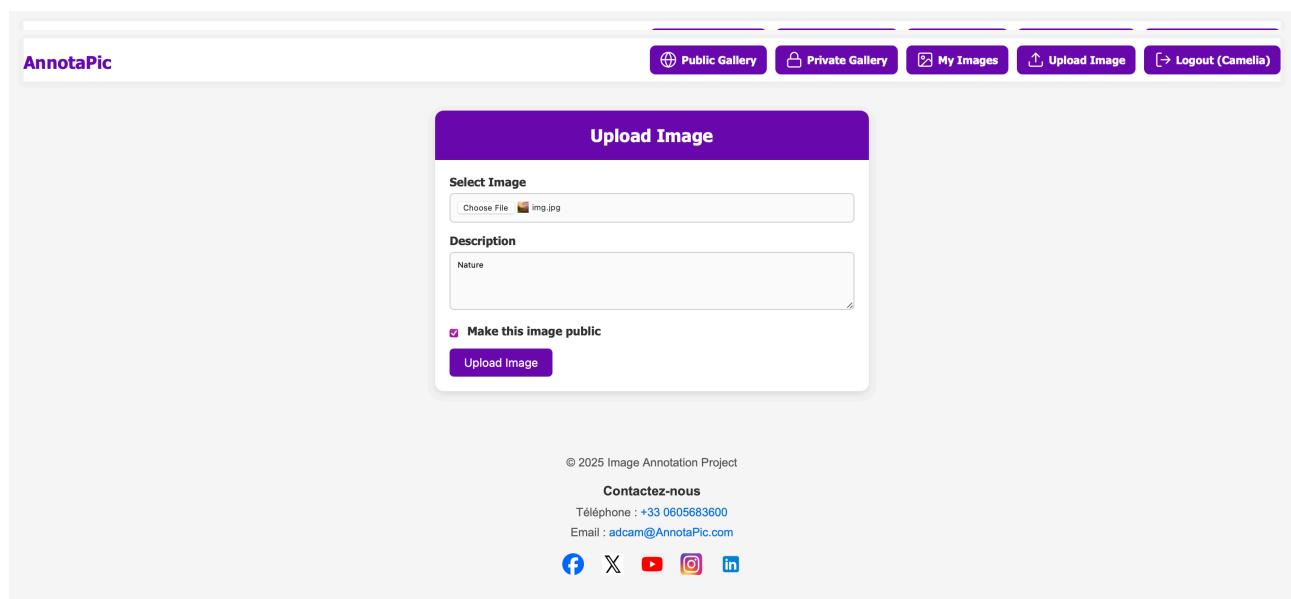
Notre idée était d'utiliser JavaScript pour permettre à l'utilisateur :

- de **cliquer et glisser** sur l'image pour dessiner un rectangle dynamique,
- d'entrer un texte associé à cette zone,
- puis d'envoyer les **coordonnées** (x, y, largeur, hauteur) + la description au serveur en PHP.

Mais plusieurs obstacles se sont présentés :

- La **manipulation du DOM** et des événements souris demandait plus de temps que prévu.
- L'intégration d'un **<canvas>** ou de systèmes HTML/CSS réactifs était techniquement complexe.
- Nous avons rencontré des bugs liés à la **conversion des dimensions réelles de l'image** vers les coordonnées CSS.
- Enfin, pour que l'expérience soit fluide et fiable, une interface bien plus poussée était nécessaire.

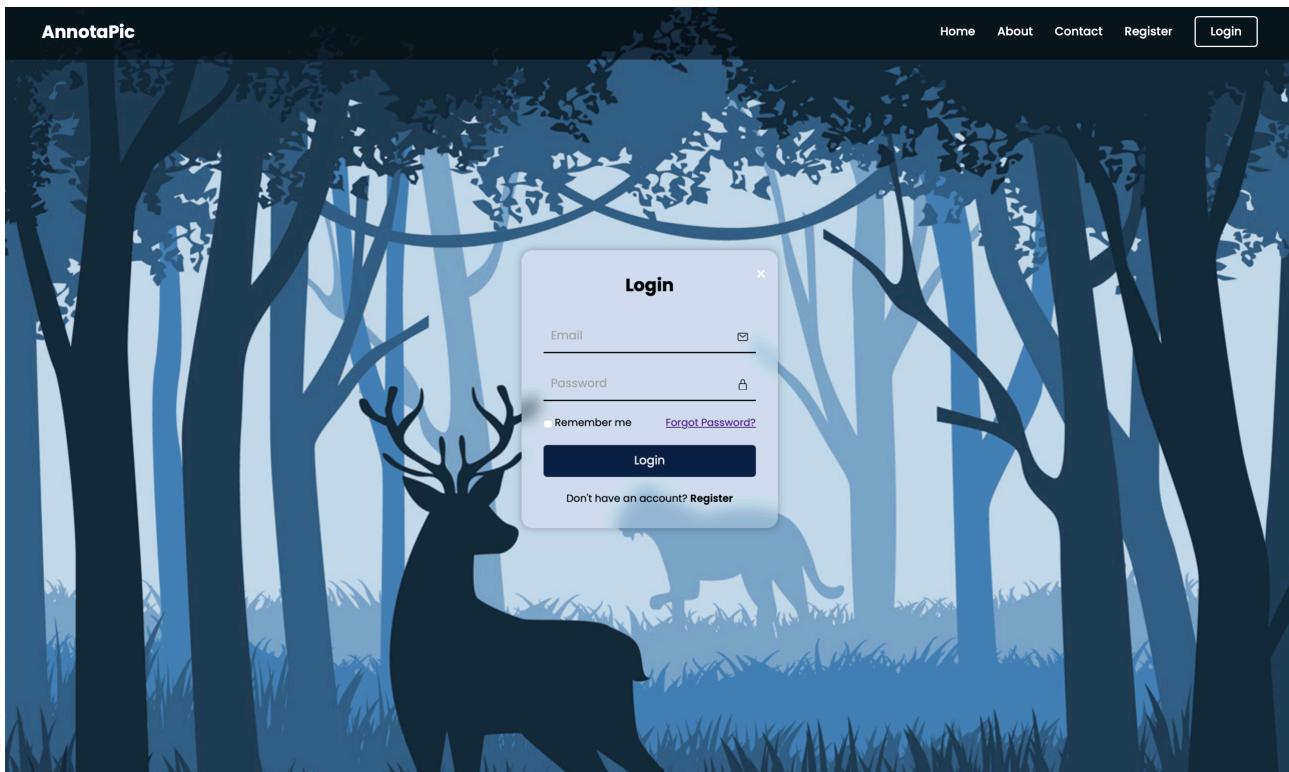
Par souci de **stabilité et de qualité**, nous avons donc pris une décision raisonnable : ne pas forcer l'implémentation de cette fonctionnalité incomplète, et proposer à la place une **description globale bien intégrée et fonctionnelle**.



The screenshot shows the AnnotaPic web application interface. At the top, there's a navigation bar with links for 'Public Gallery', 'Private Gallery', 'My Images', 'Upload Image', and 'Logout (Camelia)'. Below the navigation, a purple header bar contains the text 'Upload Image'. Underneath this, there are two input fields: 'Select Image' (with a 'Choose File' button and a preview area showing 'img.jpg') and 'Description' (with the text 'Nature'). A checkbox labeled 'Make this image public' is checked. At the bottom of the form is a purple 'Upload Image' button. At the very bottom of the page, there's a footer with copyright information ('© 2025 Image Annotation Project'), contact details ('Contactez-nous', 'Téléphone : +33 0605683600', 'Email : adcam@AnnotaPic.com'), and social media links for Facebook, X, YouTube, Instagram, and LinkedIn.

✓ Ce qui fonctionne bien

- Le système d'inscription et de connexion fonctionne parfaitement, avec sessions PHP sécurisées.



- Les utilisateurs peuvent envoyer des images, stockées sur le serveur et liées à leur compte.
- Chaque image peut être accompagnée d'une description personnalisée.
- L'utilisateur retrouve ses images dans une galerie claire, accessible, et bien structurée.
- L'interface est responsive (grâce à du CSS de base) et agréable à utiliser.
- Le tout fonctionne dans un environnement Docker stable, prêt à être déployé sur d'autres machines.
- Lors de l'upload, l'utilisateur peut choisir si l'image est publique ou privée, ce qui permet de gérer la visibilité de ses contenus.

My Private Images



© 2025 Image Annotation Project

Contactez-nous

Téléphone : +33 0605683600

Email : adciam@AnnotaPic.com

Sécurité

- Gestion des **sessions PHP** pour contrôler les connexions.
- Protection des pages sensibles (accès restreint aux utilisateurs connectés).
- Validation et filtrage des entrées utilisateur (**htmlspecialchars**, vérification PHP côté serveur).

```
[mysql]> SELECT *FROM users;
+----+-----+-----+-----+-----+
| id | email          | password          | username | created_at |
+----+-----+-----+-----+-----+
|  1 | ermc1602@gmail.com | $2y$10$W4g5DhuE4MgkKCZYJMYtfu/ehdbuQe2F3yJwzC8.ZW5PCHW.i9Mgu | Camelia | 2025-05-17 18:57:20 |
+----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql> ]
```

- Vérification des fichiers uploadés (type MIME, taille, extension).
- Utilisation de requêtes SQL préparées (**prepare**, **bind_param**) pour éviter les injections.
- **Hachage sécurisé des mots de passe** avec **password_hash()** : les mots de passe ne sont pas stockés en clair dans la base de données, ce qui protège les comptes en cas de piratage.

Sources utilisées

- [PHP.net](https://www.php.net/) – documentation officielle PHP
- [W3Schools](https://www.w3schools.com/) – tutoriels PHP, HTML, SQL
- [Stack Overflow](https://stackoverflow.com/) – pour résoudre des erreurs spécifiques
- Docker Docs – pour la mise en place de l'architecture
- Tutoriels divers (JavaScript, manipulation DOM, gestion d'images)

Installation (manuel technique)

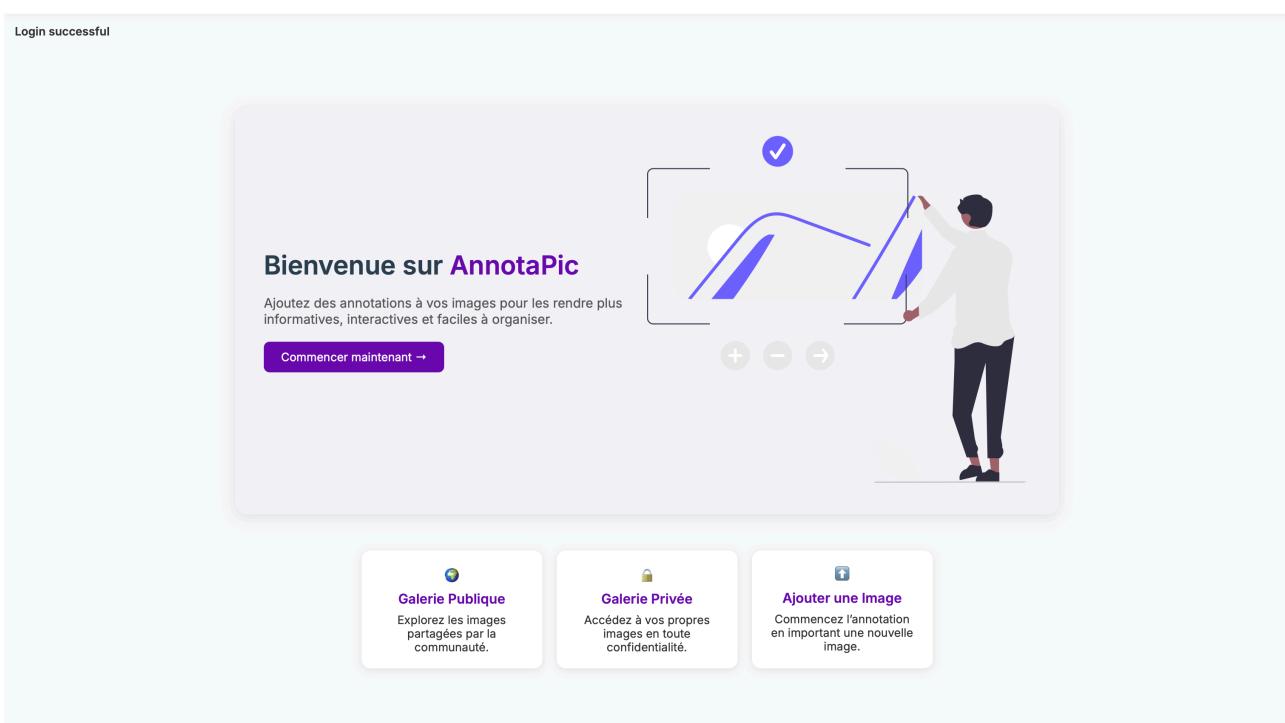
Prérequis :

- Avoir **Docker** et **Docker Compose** installés.

Étapes d'installation :

```
bash  
docker-compose up --build
```

L'application est accessible via : <http://localhost:8081> . Lorsqu'on y accède, on est automatiquement redirigé vers la **page principale du site**, qui permet à l'utilisateur de se connecter ou de s'inscrire.Une fois connecté, il peut accéder à sa galerie personnelle, ajouter des images, et consulter les descriptions associées.



Configuration interne (aucune saisie manuelle)

- Les identifiants MySQL (utilisateur root, mot de passe, nom de base) sont déjà définis dans `docker-compose.yml` et utilisés automatiquement par l'application via le fichier `includes/config.php`.
- L'utilisateur n'a **rien à saisir manuellement** : tout est préconfiguré pour fonctionner immédiatement.

Fichiers exécutés automatiquement

Le script SQL de création de base de données `db/init.sql` est automatiquement exécuté par le conteneur MySQL à son premier lancement.

Il crée les tables suivantes :

- `users` (utilisateurs du site)
- `images` (images uploadées)
- `annotations` (descriptions associées aux images)

Aucune exécution manuelle n'est nécessaire

Idées d'évolutions

- Implémenter la vraie **sélection de zone** avec `<canvas>` ou des `div` interactives.
- Ajouter la possibilité de **taguer des personnes ou des objets** sur une image.
- Permettre à l'utilisateur de **dessiner librement** des formes ou zones personnalisées.
- Générer un **PDF annoté** exportable des images.
- Ajouter un **moteur de recherche** dans les descriptions ou images.

Conclusions

Ce projet nous a permis de travailler sur un **cas concret et complet de développement web**, en mêlant backend, frontend, base de données et infrastructure.

On a appris à :

- gérer toutes les étapes du cycle de développement,
- utiliser **Docker** pour créer un environnement stable et portable,
- faire face à des contraintes techniques (temps, complexité) et **adapter nos choix**,
- livrer un projet **fonctionnel, fiable et utilisable**.

Même si la fonctionnalité d'annotation graphique n'a pas été réalisée, on a su faire preuve de discernement pour proposer une alternative stable et utile. On ressort de ce projet avec une **expérience enrichissante**, une meilleure compréhension des enjeux du développement web, et une motivation renforcée à continuer d'apprendre et à faire évoluer cette application.