

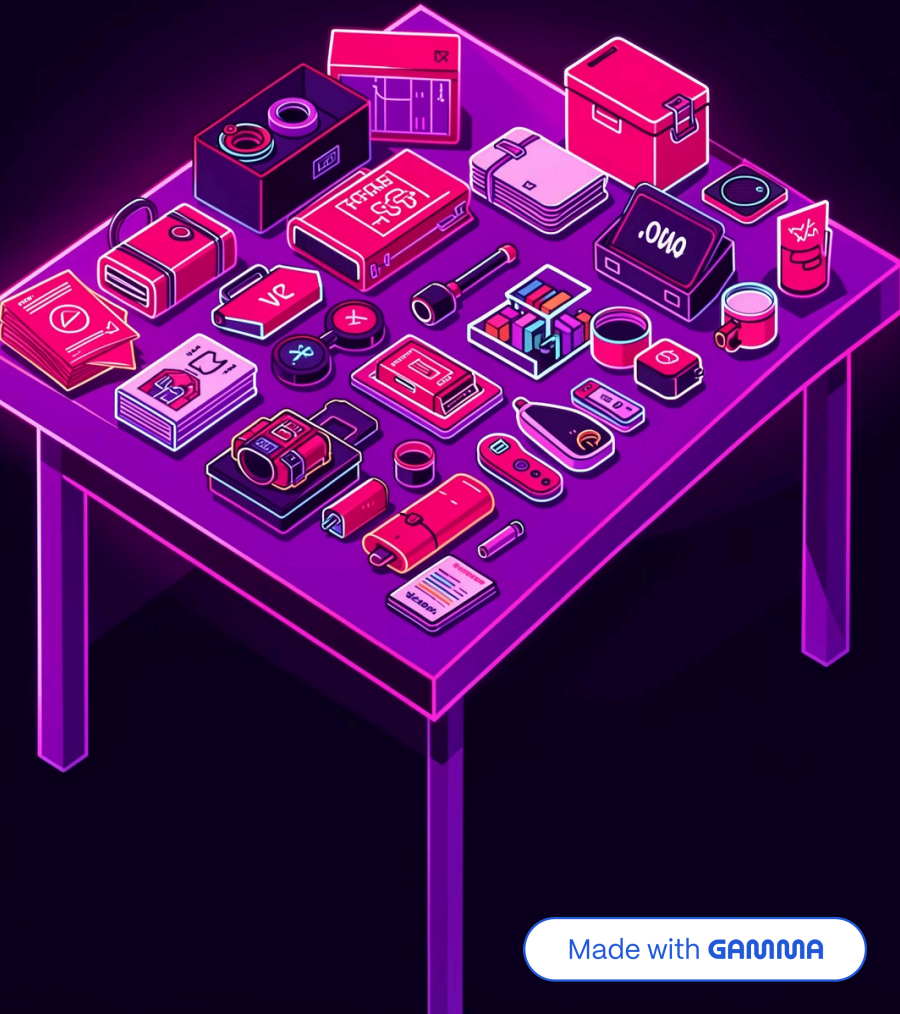
# Plateforme Objets Trouvés & Perdus

## Application Web pour le Signalement d'Objets

**Développeur : nahmad imane**

**Date :** Février 2026

**Superviseur :** Salma Harda



# Implémentation Frontend

La partie frontend de la plateforme est conçue pour être intuitive, réactive et rapide, offrant une expérience utilisateur fluide et moderne.

## Technologies Clés

- **React.js avec TypeScript** : Pour une interface utilisateur dynamique et maintenable.
- **Tailwind CSS** : Pour un stylisme rapide et flexible.
- **React Router** : Pour une navigation SPA (Single Page Application).
- **Axios** : Pour une communication efficace avec l'API backend.

## Composants Principaux

- **Contexte d'Authentification** : Gestion sécurisée des sessions utilisateur.
- **Composant de Navigation** : Barre de navigation intuitive pour une exploration facile.
- **Formulaires/Affichage d'Objets** : Pour le signalement et la consultation des objets perdus/trouvés.
- **Tableau de Bord Administrateur** : Interface pour la gestion des utilisateurs et des contenus.

# Table des Matières

01

Présentation du Projet

02

Énoncé du Problème

03

Approche de la Solution

04

Architecture Technique

05

Fonctionnalités du Système

06

Implémentation

07

Défis & Solutions

08

Améliorations Futures

09

Conclusion

```
225: Slove boend =>((attencndt,>
lesbnonacecne=nnnhlort){
vfroãtts { ltoxiice//y,ic">};
feaplnt iqm =>{(boncteauccehfone)}
Afchen,ero}
Ayenvesal/(T'ponlee lãht(7))
kñon snaryc 't"<;
shoonul)m);
noevo eeat; (ru6iè:f/)}
'tnnuservort")
```



# Présentation du Projet

**Objectif :** Créer une plateforme web pour faciliter le signalement et la récupération d'objets perdus/trouvés.

## Type

Application web full-stack

## Technologies

React.js, Laravel, SQLite/MySQL,  
Cloudinary

## Méthodologie

Développement Agile



# Énoncé du Problème

Il est difficile de connecter les personnes qui perdent des objets avec celles qui les trouvent.

## Manque de Plateforme

Pas de système centralisé pour les objets perdus/trouvés.

## Recherche Manuelle

Processus de recherche manuel chronophage.

## Manque de Vérification

Absence de mécanismes de vérification fiables.

## Portée Limitée

Méthodes traditionnelles avec une portée restreinte.



# Approche de la Solution

Une plateforme web centralisée pour améliorer le taux de récupération des objets perdus.

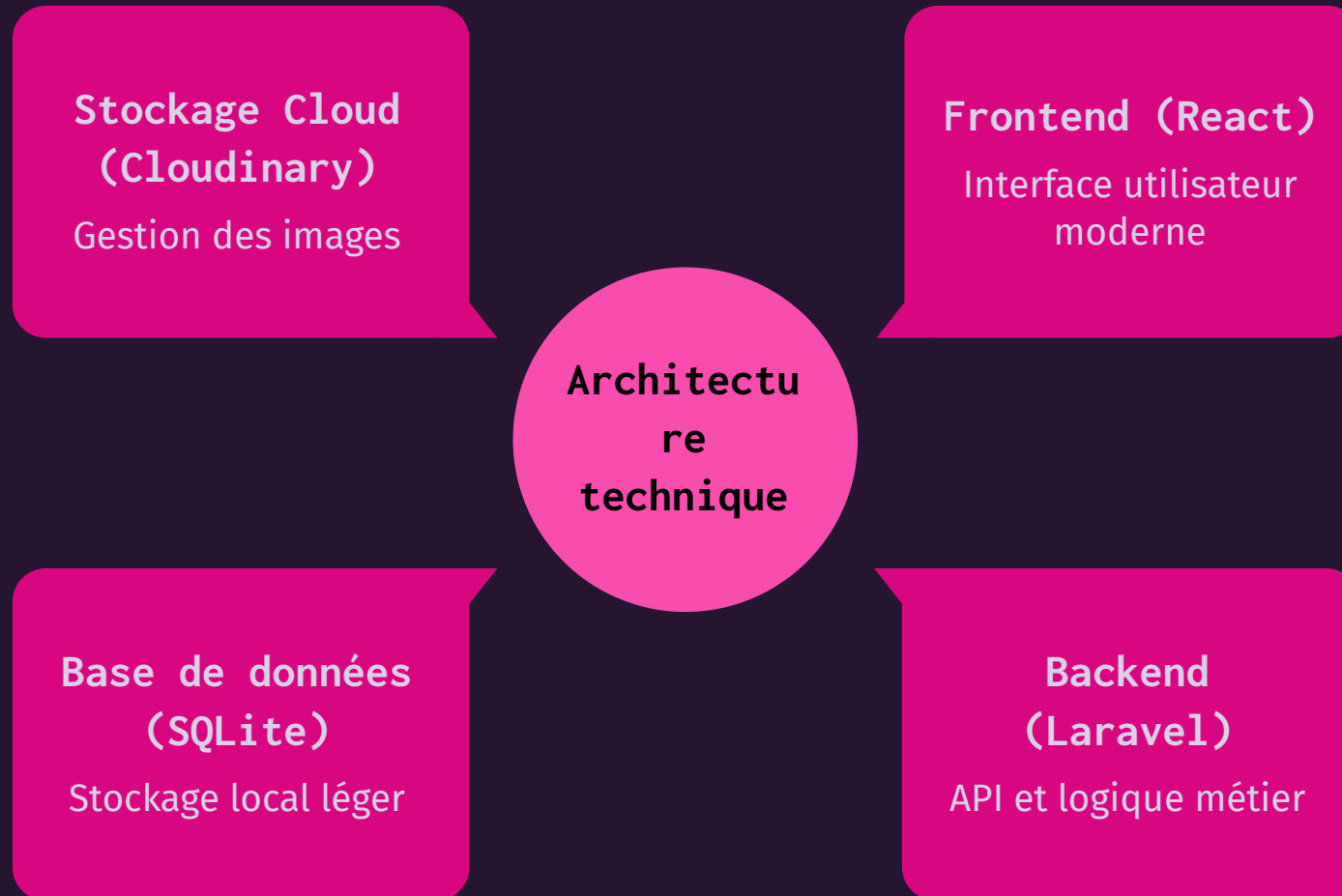
## Rôles Utilisateur

- Utilisateurs réguliers (signalement/réclamation)
- Administrateurs (vérification/modération)

## Fonctions Clés

- Signalement d'objets avec images
- Recherche et filtrage
- Système de vérification
- Gestion des contacts

# Architecture Technique



L'architecture est composée d'un frontend React, d'un backend Laravel, d'une base de données SQLite et d'un stockage cloud Cloudinary pour les images.





# Fonctionnalités du Système : Gestion des Utilisateurs

## Inscription/Connexion

- Authentification sécurisée
- Contrôle d'accès basé sur les rôles
- Gestion des sessions

## Gestion du Profil

- Informations personnelles
- Coordonnées
- Historique d'activité





# Fonctionnalités du Système : Gestion des Objets

## Signalement d'Objets

- Description détaillée
- Informations de localisation
- Téléchargement d'images (Cloudinary)
- Classification (perdu/trouvé)

## Recherche & Filtrage

- Par type, lieu, mots-clés
- Options de recherche avancée
- Affichage des résultats réactif



# Fonctionnalités du Système : Système de Vérification

## Panneau d'Administration

- Examen des objets soumis
- Gestion du statut de vérification
- Modération du contenu

## Processus de Vérification

- En attente → Vérifié → Résolu
- Mesures de contrôle qualité
- Résolution des litiges

# Fonctionnalités Supplémentaires



## Design Réactif

Approche mobile-first et compatibilité multi-navigateurs.



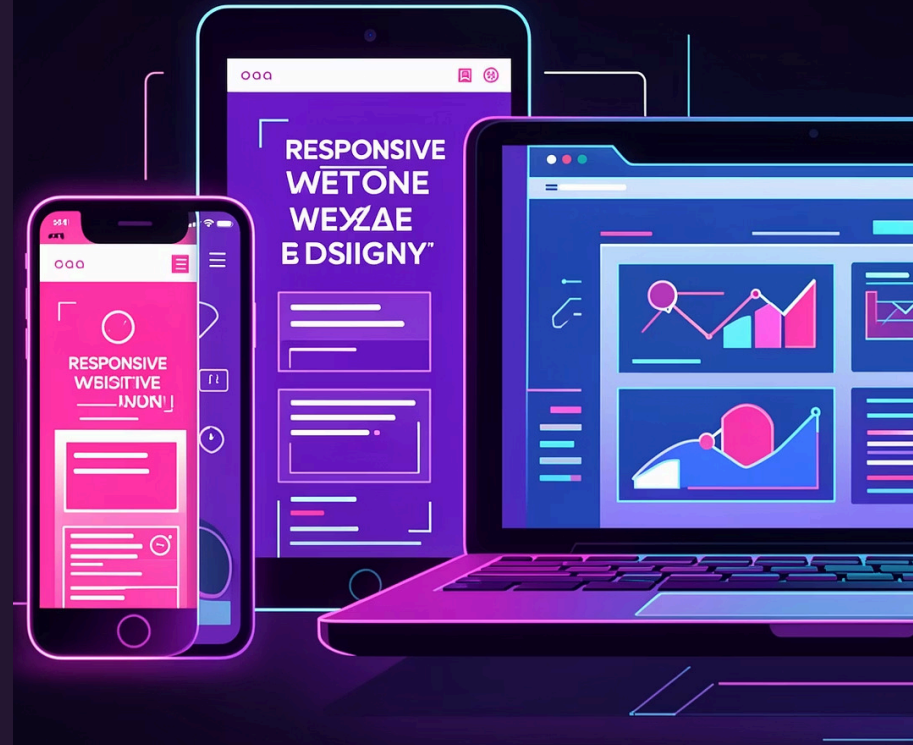
## Gestion d'Images

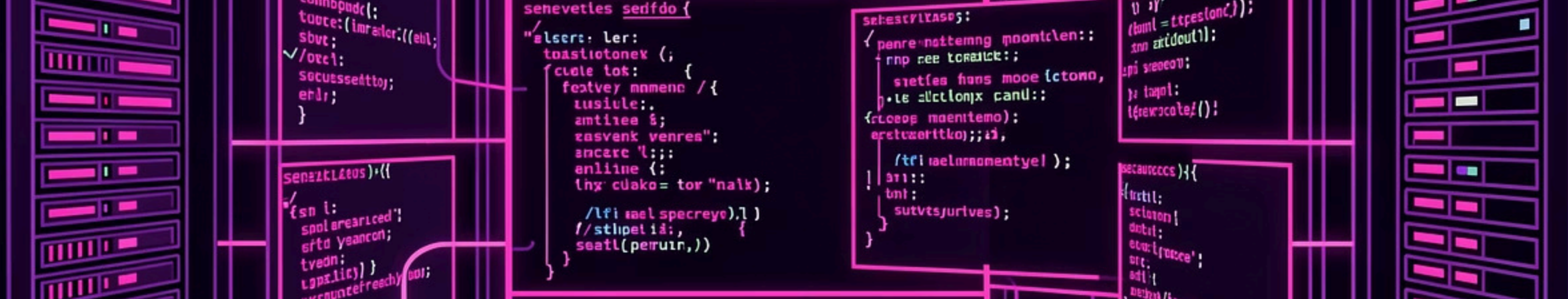
Intégration Cloudinary, optimisation automatique et stockage sécurisé.



## Sécurité

Validation des entrées, gardes d'authentification et protection des données.





# Implémentation Backend

Le backend est le pilier de la plateforme, gérant la logique métier, l'accès aux données et l'authentification de manière sécurisée et performante.

## Technologies Utilisées

- **Laravel Framework** : Pour une architecture robuste et structurée, facilitant le développement rapide et la maintenabilité.
- **PHP 8.2+** : Langage de programmation moderne et performant, offrant une exécution rapide et des fonctionnalités avancées.
- **API RESTful** : Conception d'API pour une communication standardisée et efficace avec le frontend et d'autres services externes.

## Modules Clés

- **Contrôleur d'Authentification** : Gère l'inscription, la connexion, la réinitialisation de mot de passe et la gestion des tokens d'accès.
- **Gestion des Objets** : Module CRUD pour la création, lecture, mise à jour et suppression des objets perdus/trouvés.
- **Service Cloudinary** : Intégration pour le stockage, l'optimisation et la livraison sécurisée des images téléchargées.
- **Modèles de Base de Données** : Définition des schémas de données et des relations pour SQLite ou MySQL, assurant l'intégrité et l'efficacité des requêtes.



# Implémentation : Base de Données

## Schéma de la Base de Données

La structure des données est définie par les tables suivantes :

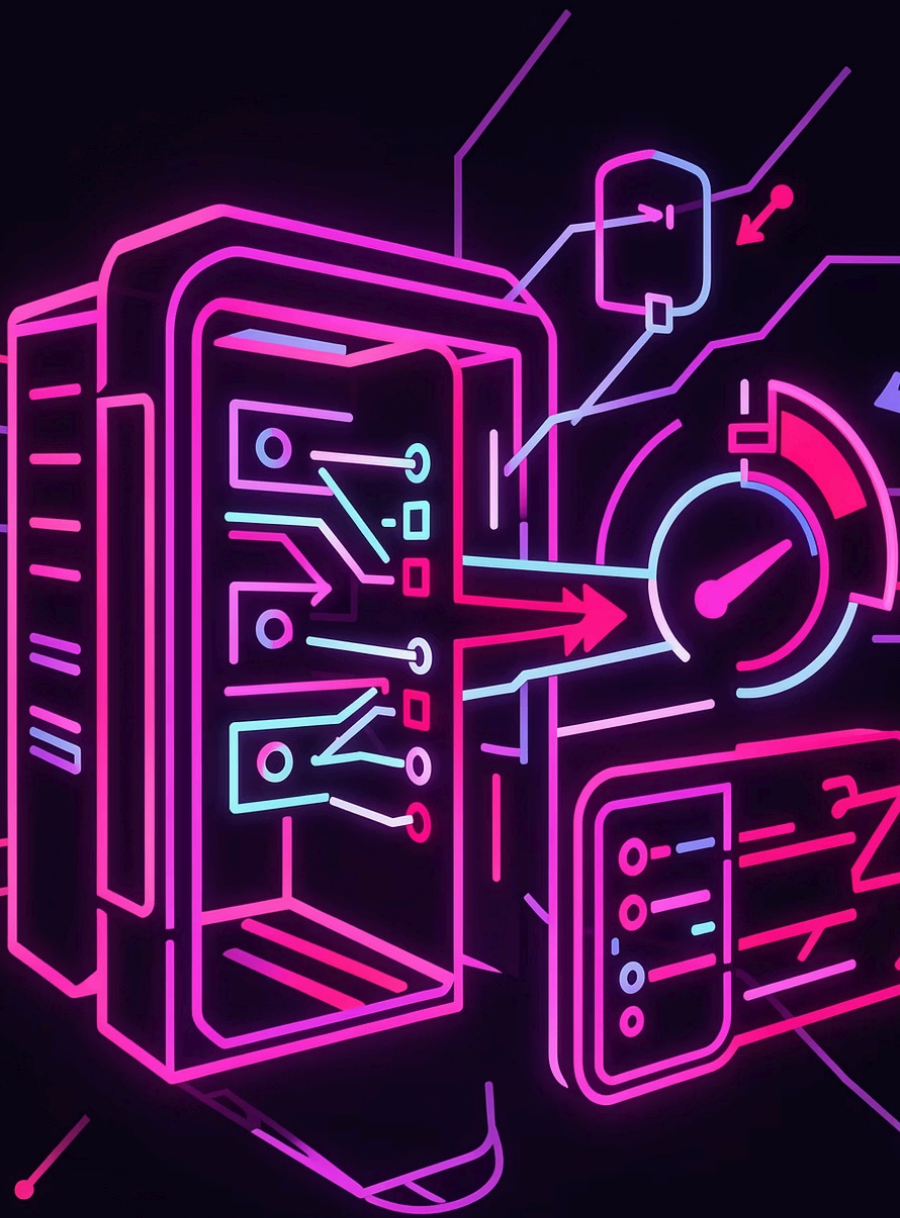
- **Table Users** : id, name, email, password, role
- **Table Items** : id, user\_id, title, description, type, location, status, image\_url
- **Champs de Vérification (dans Items)** : verification\_status, verified\_by, verified\_at

## Relations

Les relations clés entre les entités assurent l'intégrité référentielle :

- **Un Utilisateur** peut signaler **plusieurs Objets**.
- **Un Objet** appartient à **un seul Utilisateur** (celui qui l'a signalé).





# Sécurité et Performance

Le backend est conçu avec une attention particulière à la sécurité des données et à l'optimisation des performances pour garantir une expérience utilisateur fluide et fiable.

## Mesures de Sécurité

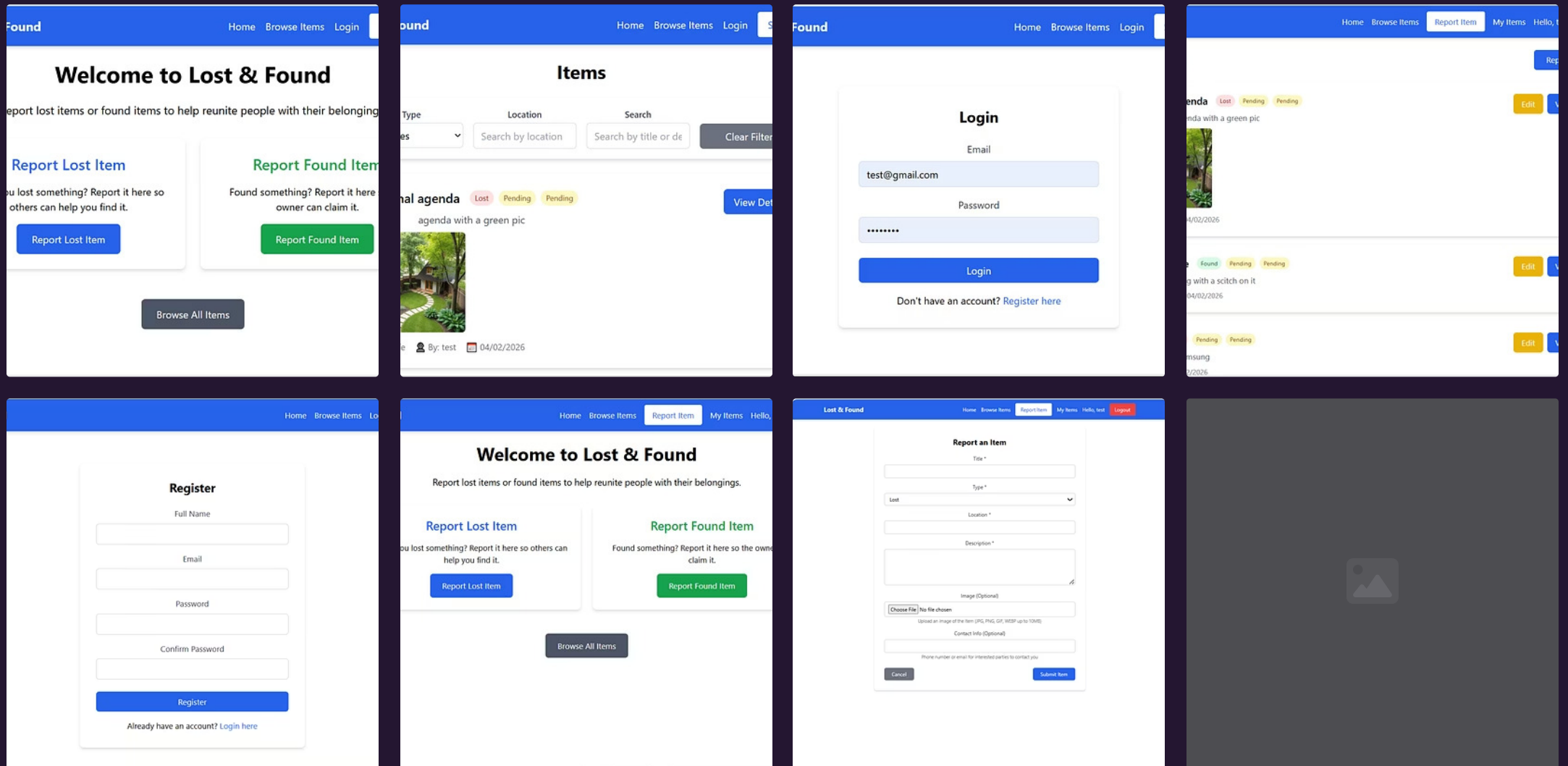
- Validation et assainissement des entrées pour prévenir les vulnérabilités.
- Middleware d'authentification pour un contrôle d'accès robuste.
- Prévention des injections SQL et protection XSS contre les attaques courantes.

## Optimisation des Performances

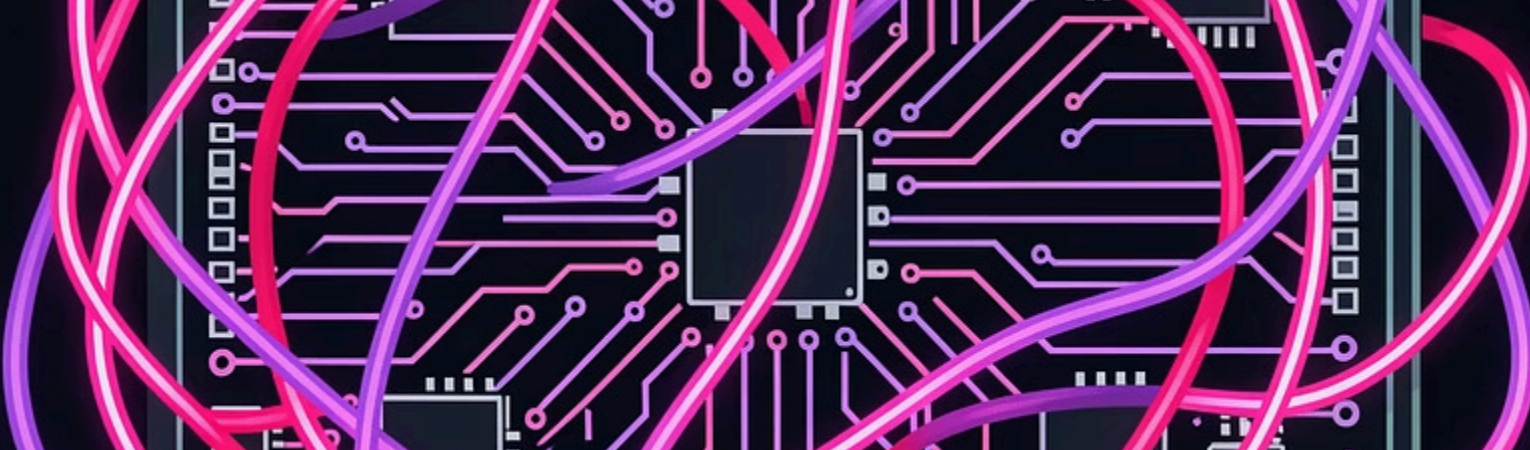
- Diffusion d'images optimisée via Cloudinary pour des chargements rapides.
- Requêtes de base de données efficaces pour une récupération rapide des informations.
- Mise en œuvre de stratégies de mise en cache pour réduire la charge serveur.

# Démo : Captures d'écran

Un aperçu visuel des fonctionnalités clés de la plateforme, illustrant l'interface utilisateur intuitive et le processus de gestion des objets.







# Défis Rencontrés

## Défis Techniques

- Complexité de l'intégration Cloudinary
- Gestion des requêtes inter-origines (CORS)
- Optimisation du téléchargement d'images

## Approches de Solution

- Gestion complète des erreurs
- Mécanismes de secours (fallback)
- Gestion rigoureuse de la configuration

# Résultats & Réalisations

Une synthèse des accomplissements majeurs de la plateforme développée.



## Application web fonctionnelle

Déploiement d'une application web complète et opérationnelle, prête à l'emploi.



## Authentification utilisateur

Système d'inscription et de connexion sécurisé avec gestion des rôles.



## Gestion des images

Intégration réussie de Cloudinary pour le téléchargement et le stockage d'images optimisées.



## Système de vérification

Mise en place d'un processus de modération et de vérification par les administrateurs.



## Interface réactive

Conception d'une interface utilisateur intuitive, conviviale et adaptable à tous les appareils.



## Gestion sécurisée des données

Implémentation de mesures de sécurité robustes pour protéger les informations des utilisateurs.



# Améliorations Futures

## Fonctionnalités Prévues

- Notifications push pour les correspondances d'objets
- Intégration de la localisation GPS pour plus de précision
- Système de chat/messagerie intégré
- Application mobile dédiée
- Tableaux de bord d'analyse avancée

## Optimisations & Innovations

- Apprentissage automatique pour la mise en correspondance d'objets
- Processus de vérification automatisé
- Intégration avec les réseaux sociaux





# Technologies et Outils

Les technologies et outils que nous avons choisis garantissent une plateforme performante, sécurisée et facile à maintenir.

## Technologies de Développement

- **PHP/Laravel** : Pour le backend, offrant un cadre MVC robuste.
- **JavaScript/React** : Pour un frontend dynamique et interactif.
- **TypeScript** : Améliore la qualité du code et la maintenabilité du frontend.
- **MySQL/SQLite** : Solutions de bases de données flexibles pour le stockage des données.

## Outils Clés

- **Docker** : Pour la conteneurisation, assurant un environnement de développement cohérent.
- **Git** : Système de contrôle de version pour une collaboration efficace.
- **Cloudinary** : Pour l'hébergement et l'optimisation des images.
- **Postman** : Pour tester et documenter les API backend.

# Conclusion



## Succès du Projet

- Toutes les exigences fonctionnelles satisfaites.
- Démonstre une compétence technique solide.
- Résout un problème concret du monde réel.



## Acquis d'Apprentissage

- Expérience approfondie en développement full-stack.
- Maîtrise de la conception et de l'intégration d'API.
- Intégration de services cloud (Cloudinary).
- Application des meilleures pratiques de sécurité.



## Impact

La plateforme fournit un service précieux et tangible à la communauté, facilitant la récupération des objets perdus et trouvés.

# Questions & Discussion

Merci de votre attention !

N'hésitez pas à poser vos questions.

---

## Contact

Email : [imanenahmad@example.com](mailto:imanenahmad@example.com)

## Code Source

GitHub : <https://github.com/emmy-na/Lost-Found.git>