Documentation Technique et Utilisateur

Ce projet a été conçu pour proposer un portfolio d'un artiste interactif permettant l'affichage dynamique d'œuvres d'art, avec un système de gestion des utilisateurs et une navigation fluide. Cette documentation vise à expliquer les choix technologiques effectués, la structure du code, ainsi que les étapes d'installation et de prise en main de l'application.

1. Choix du Framework et des Technologies

- **Front-end React** : J'ai choisi React.js pour développer l'interface utilisateur car il offre une approche componentielle permettant une meilleure réutilisabilité du code. Son écosystème est riche et bien documenté, ce qui simplifie la gestion de l'état et le routage avec React Router. De plus, React permet un rendu dynamique et optimisé grâce au Virtual DOM, améliorant ainsi les performances de l'application.

- **Back-end Node.js et Express :** L'utilisation de Node.js avec Express.js pour le backend a été motivée par sa capacité à gérer efficacement des requêtes asynchrones et sa compatibilité naturelle avec MongoDB, la base de données utilisée. Express offre une architecture minimaliste et flexible, idéale pour concevoir une API REST qui communique avec le frontend React.

- **Base de données MongoDB :** J'ai opté pour MongoDB car c'est une base de données NoSQL parfaitement adaptée aux applications web modernes. Elle permet de stocker des documents JSON, ce qui simplifie la gestion des données et la communication avec le frontend. De plus, l'intégration avec Mongoose facilite la validation et la structuration des données côté serveur.

Outils et Bibliothèques Complémentaires

Pour optimiser le développement et garantir une bonne organisation du projet, plusieurs outils et bibliothèques ont été utilisés :

- Webpack et Babel : Pour l'optimisation du code JavaScript.
- Jest : Pour l'écriture et l'exécution de tests unitaires sur les composants React.
- Axios : Pour faciliter les requêtes HTTP et la gestion des réponses API.
- Vite.js : Pour un démarrage rapide du projet et une compilation optimisée en mode développement.

2. Structure du Projet

Le projet est divisé en deux parties principales : le frontend (interface utilisateur) et le backend (API et gestion des données).

Arborescence des fichiers

Le projet est organisé de la manière suivante :

```
├── backend
│   ├── server.js (Configuration du serveur)
│   ├── models (Modèles de données MongoDB : Category, User, Tag, Oeuvre, Content)
│   └── controllers (EmailService.js)
└── frontend
    ├── src
    │   ├── App.js (Structure des blocs)
    │   ├── App.scss
    │   ├── components (Composants réutilisables)
    │   │   ├── Bloc1
    │   │   ├── Bloc2….(autres Blocs…)
    │   │   ├── Page de connexion
    │   │   ├── Page de réinitialisation de mot de passe
    │   │   └── Page d’inscription
    │   ├── styles (Images et fichiers CSS globaux)
    └── public
        └── index.html (Fichiers statiques)"
```

3. Fonctionnalités Principales

L’application propose les fonctionnalités suivantes :

- Affichage dynamique des œuvres : Les œuvres sont récupérées via une API et affichées sans rechargement de la page.

- Gestion de l’état : Utilisation de Context API et hooks React pour maintenir les données utilisateurs.

- Authentification des utilisateurs : Système de connexion/inscription sécurisé avec JWT.

- Interactivité avancée : Chargement dynamique des œuvres avec Ajax et animations CSS.

4.1. Diagrammes UML :

Diagramme de Cas d'Utilisation :

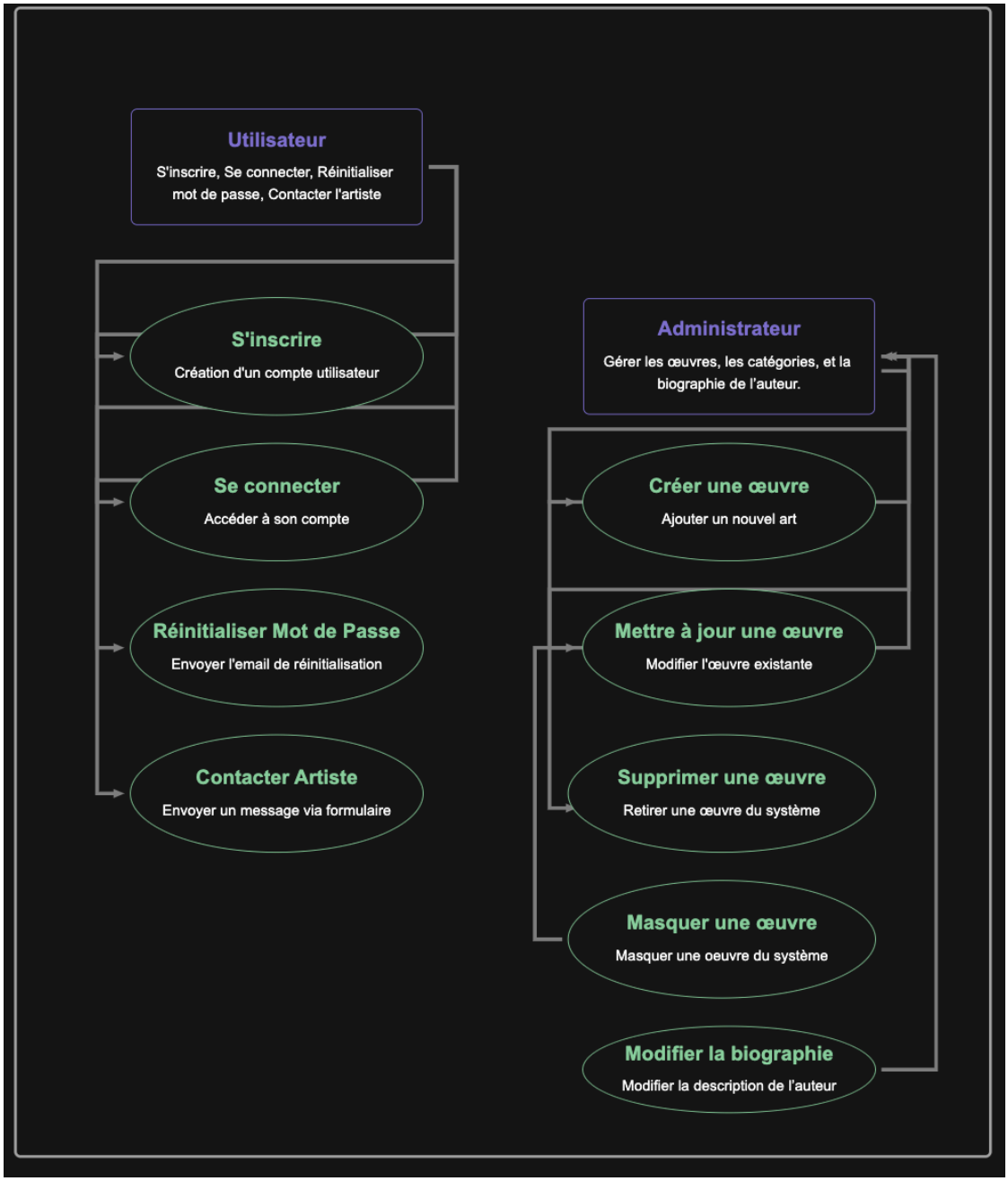


Diagramme Entité-Association :

Diagramme de Séquence - Inscription :

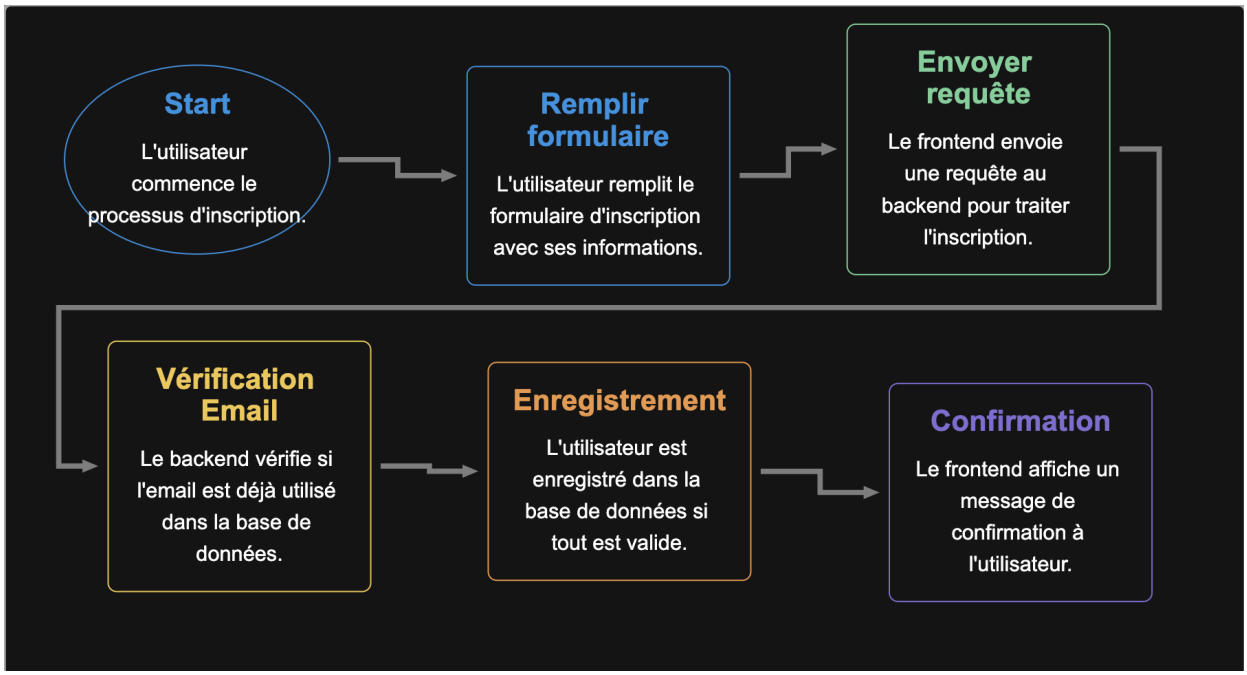
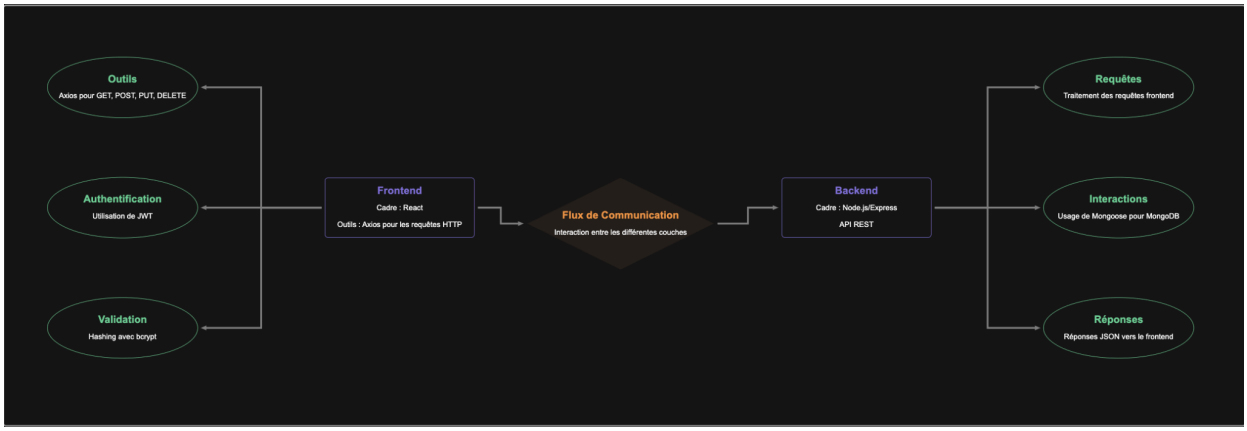


Diagramme d' Architecture :



4. Déploiement et Hébergement

Site déployé : https://portfolio-frontend-hvlx.onrender.com/#home.

L'application a été déployée avec succès sur la plateforme Render, permettant une mise en ligne simplifiée et sécurisée. Pour les besoins de la présentation du projet, une démonstration locale a été réalisée afin de garantir une expérience optimale et de s'assurer du bon fonctionnement des différentes fonctionnalités dans un environnement contrôlé.

5. Tests et Validation

Afin d'assurer la stabilité et la performance de l'application, plusieurs types de tests ont été réalisés :

- Tests unitaires : Les composants React sont testés avec Jest. Exemple de test dans le dossier.
- Tests fonctionnels Media Queries OK avec Responsinator : http://www.responsinator.com/?url=https%3A%2F%2Fportfolio-frontend-hvlx.onrender.com%2F & Browserstack OK - fonctionnel.
- Tests de performance : Analyse avec Lighthouse pour optimiser le temps de chargement. Performance à 91 %.