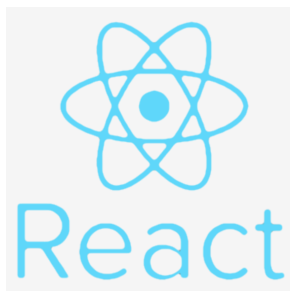




Rapport de Projet FullStack: Portfolio Développeur



Réalisé par :

Stevine AYEMELE

Sous l'encadrement de:

M. Emmanuel BLANCHARD

1) Présentation du projet et de ses objectifs

L'objectif de ce projet est de concevoir un site web fullstack servant de portfolio pour présenter différents projets informatiques personnels. Le site offre également aux visiteurs la possibilité de contacter l'administrateur via un formulaire de contact.

Le projet est structuré en deux grandes parties :

- Backend : Une API REST développée en Node.js et Express.js, connectée à une base de données MongoDB. Cette API permet la gestion complète des projets (création, lecture, mise à jour, suppression) ainsi que l'envoi de messages de contact.
- Frontend : Une application réalisée en React.js. L'interface permet une navigation fluide entre la page d'accueil, la liste des projets, le détail de chaque projet, la mise à jour des projets et le formulaire de contact.

Les objectifs principaux de ce portfolio sont les suivants :

- Maîtriser le développement d'une architecture web complète (API REST pour le backend, application monopage pour le frontend).
- Appliquer les bonnes pratiques de développement web, telles que la structuration du code, la gestion de l'asynchrone, la sécurité des accès, et l'organisation des états côté frontend.
- Renforcer la capacité à documenter et présenter un projet complet, à travers la rédaction de rapports techniques et la création de livrables associés (README, vidéo de démonstration).

2) Stratégie de projet

La conduite du projet s'est appuyée principalement sur deux outils : Git et une To-Do List manuelle.

- Gestion de version :

Le système de gestion de version Git a été utilisé tout au long du développement. Il a permis d'assurer la traçabilité des évolutions du projet, la sauvegarde régulière des avancées et la possibilité de revenir à des versions antérieures si nécessaire. Le projet est hébergé sur GitHub dans un dépôt structuré, séparant clairement le frontend et le backend.

-Organisation des tâches :

Une To-Do List personnelle a été élaborée pour planifier et suivre les différentes étapes du développement.

Chaque fonctionnalité importante (par exemple : "Mettre en place l'authentification admin", "Créer la page de détail d'un projet", "Configurer la base de données MongoDB") a été inscrite et cochée une fois réalisée.

Cette méthode simple m'a permis de garder une vue d'ensemble sur l'avancement du projet et de prioriser les tâches critiques.

Bien que le projet n'ait pas utilisé d'outils de gestion collaborative plus complexes (comme Jira ou Trello), cette approche légère s'est révélée suffisante pour un projet individuel de cette ampleur.

3)Backend

- Présentation de l'objectif

Concernant la partie backend du projet, une API REST a été développée avec Node.js et Express.js ; avec pour objectif de fournir l'ensemble des fonctionnalités nécessaires au portfolio, notamment :

- La gestion des projets (ajout d'un nouveau projet, affichage , mise à jour et suppression des projets : CRUD).
- La réception et l'enregistrement des messages envoyés par les visiteurs via le formulaire de contact.

La base de données utilisée est MongoDB Atlas, accessible à distance. La validation des données est assurée par Mongoose.

- **Architecture et explication de l'implémentation**

Le backend est organisé de manière modulaire pour garantir la lisibilité et la maintenabilité du code.

Fichiers principaux

- **server.mjs** : fichier principal du serveur, configurant Express, la gestion CORS, l'utilisation des requêtes JSON, ainsi que le chargement des routes.
- **model/routes/route.mjs** : fichier définissant les différentes routes liées aux projets et au formulaire de contact.
- **database.mjs** : fichier gérant la connexion sécurisée à MongoDB via Mongoose et la définition des schémas de données.
- **.env** : fichier stockant les variables d'environnement sensibles (URI MongoDB, port du serveur).

- **Principales routes API**

Méthode	URL	Description
GET	<code>/projects</code>	Récupérer tous les projets
GET	<code>/projects/:id</code>	Récupérer un projet spécifique
POST	<code>/projects</code>	Ajouter un nouveau projet
PUT	<code>/projects/:id</code>	Mettre à jour un projet existant
DELETE	<code>/projects/:id</code>	Supprimer un projet
POST	<code>/projects/contact</code>	Envoyer un message de contact

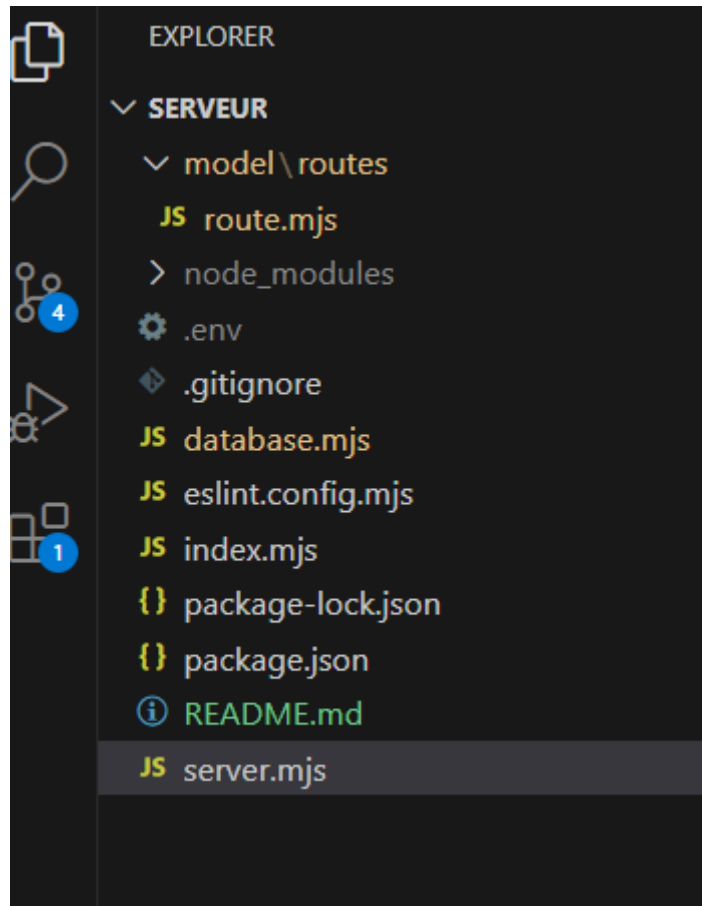


Figure 1 : Arborescence des fichiers du serveur Backend (Node.js + Express.js)

4)Frontend

- **Présentation de l'objectif**

Le frontend de notre application a été développé avec **React.js**. Son objectif est d'afficher de manière fluide les projets, de proposer une navigation simple entre les différentes sections du site et de permettre aux visiteurs de laisser un message via un formulaire de contact.

- **Architecture et explication de l'implémentation**

Le projet React est structuré de manière claire :

Arborescence principale:

- **src/components/** : contient les pages principales il s'agit entre autres de (AdminPage, authContext, authProvider etc).

- **src/components/** : composants réutilisables tels que (Accueil, Projets, Détail du Projet, Contact, Analytics, etc).
- **src/services/** : gestion des appels API avec fetch.
- **src/assets/** : ce dossier contient les différentes images utilisées lors de la mise au point du projet.
- **main.jsx**: configuration des routes et intégration des différentes pages.

URL	Page associée
/	Page d'accueil (présentation et formulaire)
/projets	Liste de tous les projets
/projets/:id	Détail d'un projet
/projets/contact	Formulaire de contact
/admin/login	Espace admin (connexion requise)

/ajouter

Ajouter un nouveau projet

/projets/edit/
:id

Modifier un projet existant

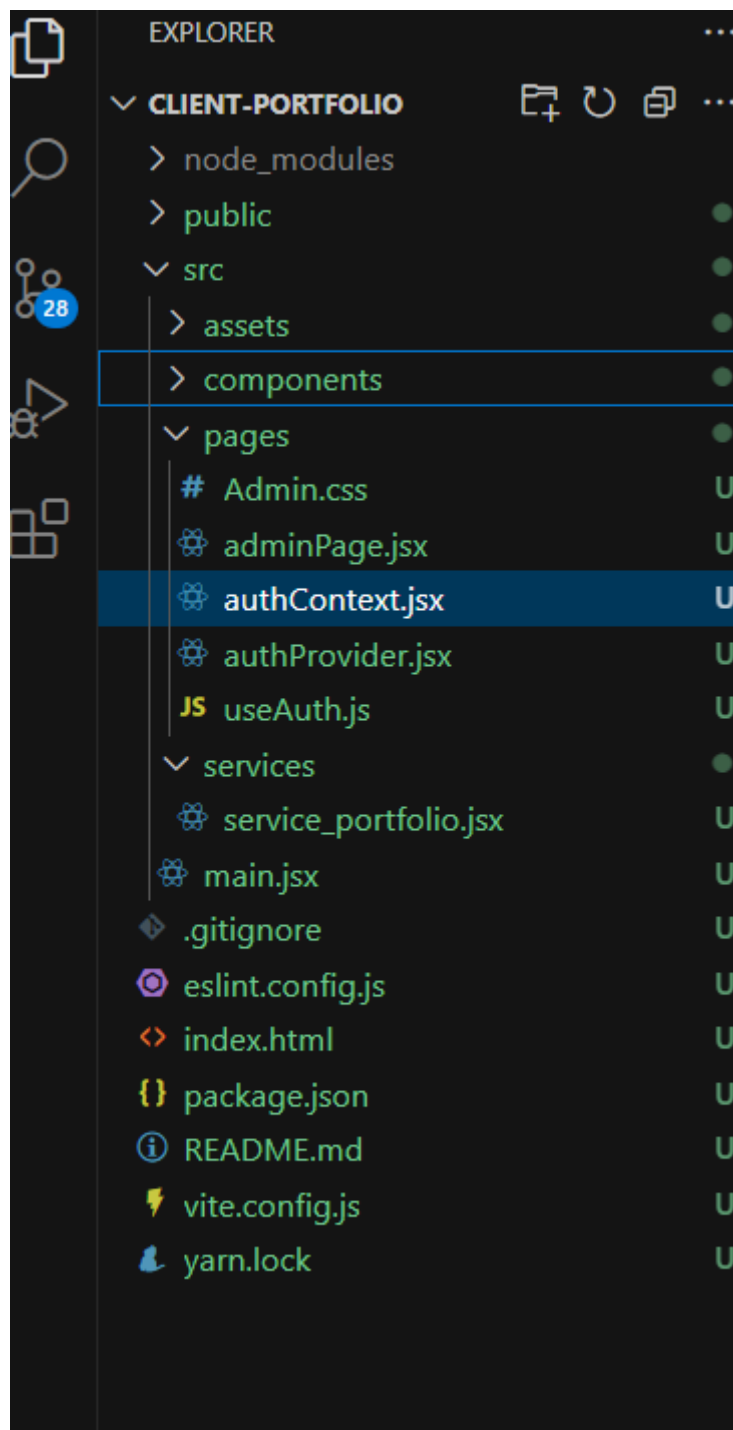


Figure 2 : Arborescence principale de l'application Frontend React.js, avec l'organisation des pages, composants et services.

5)Section Admin

Une section d'administration sécurisée a été développée afin de permettre aux administrateurs de :

- Ajouter de nouveaux projets.
- Modifier ou supprimer des projets existants.
- Consulter des statistiques analytiques sur les projets.

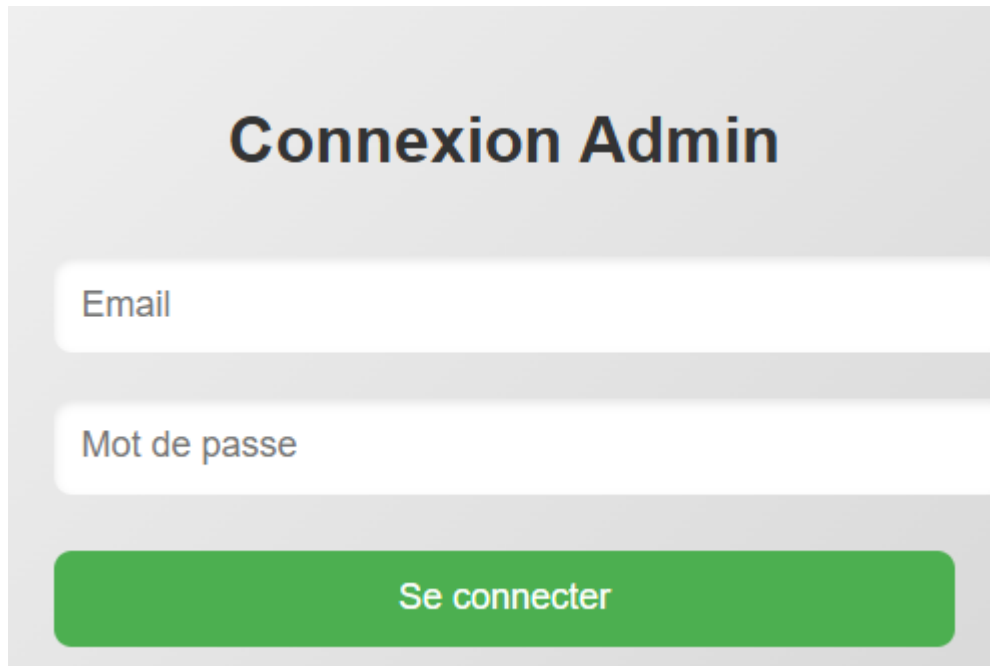
L'accès à la section admin est protégé par une authentification. Seuls les utilisateurs authentifiés peuvent de ce fait effectuer les fonctionnalités spécifiées.

AuthContext : Contexte React permettant de stocker et partager l'état d'authentification de l'utilisateur à travers toute l'application.

AuthProvider : Composant enveloppant l'application pour fournir l'accès au contexte d'authentification.

UseAuth : Hook personnalisé pour accéder facilement aux fonctionnalités d'authentification (login, logout, récupération de l'état connecté).

AdminPage : Page principale de l'espace d'administration, affichant les actions possibles (ajouter un projet, modifier un projet, consulter les statistiques).



The image shows a web form for admin login. It has a light gray background. At the top, the title 'Connexion Admin' is centered in a bold, dark font. Below the title, there are two white input fields with rounded corners. The first field is labeled 'Email' and the second is labeled 'Mot de passe'. Below these fields is a green button with rounded corners and the text 'Se connecter' in white.

Figure 3 : Authentification et à la section d'administration

6) Références

- Documentation Express.js
- Documentation MongoDB Atlas
- Documentation React.js
- Cours et tutoriels