Le Mans Université

ENSIM École d'ingénieurs Le Mans Université

Développement Fullstack

# Compte-rendu de projet

*4A ENSIM FISA*

2025

Professeur:

E. Blanchard

Rédigé par J. Issert

# Sommaire

## Présentation:

*Ce projet de développement Fullstack porte sur la réalisation d'une application complète et fonctionnelle à vocation de portfolio professionnel. Cette webapp propose une section de présentation de contenu publique et de certaines actions (et donc de sections) réservés à des utilisateurs identifiés et ayant des droits d'éditions (les admins). Le site permet donc en résumé de :*

- *Présenter son profil et ses réalisations.*
- *Offrir un système d'authentification et de gestion de rôles.*
- *Proposer des interfaces d'administration sécurisées pour gérer le contenu.*
- *Suivre des statistiques sur le contenu.*

# 1 Stratégie de projet

## Outils utilisés

Par préférence le versionning au cours du développement de ce projet à été réalisé sur Gitlab. Le projet a ensuite été poussé sur le Github partagé pour évaluation. La gestion calendaire a été gérée par des tâches Gitlab.

Le développement a été réalisé sur VSCode avec des packages comme Error Lens et Eslint. Deux linters ont été configurés, un pour le backend et un pour le frontend. Ceux-ci ont permis de traiter des erreurs en plus d'uniformiser la base de code. De même les outils console React ont permis de debugger et de valider la conception du frontend.

Postman Desktop a permis d'effectuer les tests de l'API lors de la conception du backend.

## Organisation du travail

Le temps de réalisation du projet étant assez réduit et travaillant individuellement sur le projet, une méthode itérative à été choisie. Elle s'est composée d'une phase de définition de l'ordre de réalisation en fonction des attendus.

Phase 1: Portée l'initialisation des deux projets, la configuration de la base de données, des environnements et des outils (linters, scripts de lancements) et des repo Gitlab et Github.

Phase 2: Ayant permis la conception et la réalisation de la partie backend (nécessitant l'utilisation de postman afin de vérifier les différentes fonctionnalités sans interface). Une attention particulière fut portée sur la définition claire des routes et des fonctions de l'API, de même qu'à la gestion de l'authentification utilisateur.

Phase 3: Permit le développement de l'interface en React. Après avoir réalisé une rapide maquette de l'organisation des écrans, j'ai développé les premiers visuels et options en commençant par ceux ayant le moins d'imbrications afin d'augmenter graduellement la complexité du projet et de garder une marge de manoeuvre en cas de besoin de changement d'organisation de mon interface.

Phase 4: Une fois une version fonctionnelle obtenue, les différents parcours utilisateurs ont été testés afin de découvrir si des bugs étaient apparus et les corriger le cas échéant.

# 2 Backend

## Objectifs techniques

Pour rappel, dans notre système, le backend doit pouvoir gérer l'authentification d'utilisateurs et gérer leur rôle afin de sécuriser certaines actions. Le serveur doit proposer une API Rest sécurisée permettant l'administration du contenu et publique pour la consultation grand public du site.

## Architecture

Le backend est codé en Javascript et se base sur Node.Js et Express.Js. La base de données est de type NoSql et est hébergée dans le Cloud par le service Mongo Atlas.
Les routes de l'API sont séparés en 5 :

- /api/health : utilisé lors des déploiements pour que les clusters kubernetes et loadbalancer testent le statut du service.
- /api/auth/ : permettant la gestion des utilisateurs (inscription, connexion...)
- /api/admin/ : qui permet à un utilisateur connecté de réaliser des actions protégées sur les statistiques et les projets
- /api/projects/ : api publique pour la visualisation du contenu des projets
- /api/uploads/ : api dédié à la gestion des images.

Le point d'entrée du serveur est le fichier server.js qui fait ensuite appel au fichiers dans /routes/ pour atteindre les contrôleurs /controllers/ nécessaires au traitement de la requête. Les middleware permettant de gérer l'authentification et le traitement des images sont disponibles dans /middleware/ . Enfin les modèles des objets utilisés par Mongoose sont définis dans /models/ .

## Réalisation

Le chargement des images est géré par multer et elles sont stockées dans un répertoire local du projet. Cette solution n'est pas viable à grande échelle mais la mise en place d'un système d'archivage distant n'avait ici pas de réels bénéfices au vu de la faible quantité de documents à stocker.

L'authentification est gérée par des tokens JWT et un middleware permet la vérification de l'authentification et du rôle des utilisateurs lors du traitement des appels API.

Un fichier .gitignore permet de gérer les éléments à transmettre au systèmes de versionning et un fichier .env permet de regrouper les information confidentielles de connexions.

# 3 Frontend

## Objectifs techniques

L'interface de notre projet Réact devait proposer une vue publique permettant la visualisation des informations personnelles et des projets et proposer un moyen de contact. Les projets devaient également pouvoir être visibles plus en détails dans une autre section.
Ensuite, certaines options de visualisation des statistiques du site et de gestion des projets et de leur contenu devaient être utilisables seulement par des administrateurs dans une vue authentifiée.

## Architecture

Par choix de design et d'UX/UI la plupart de la navigation sur le site passe par des popups ou modals qui permettent un parcours utilisateur plus fluide et compréhensible que des redirections vers des pages dédiées. Le site à été divisé en Pages et Composants React :

- page Home : page de départ du site contenant les informations et options de navigation.
- page Login : permettant aux utilisateurs de se connecter.
- page Stats : permettant la visualisation des statistiques du projet

Ensuite les composants (tels que les cards, modals...), sont importés dans les pages. La navigation se fait entre les pages via le routing React Router.
La logique est ensuite répartie entre les services (chargés de gérer les appels aux backend via Axios) et les stores qui permettent de faire la gestion logique intermédiaire.

Un fichier .gitignore permet de gérer les éléments à transmettre au systèmes de versionning et un fichier .env permet de regrouper les information confidentielles de connexions. Néanmoins dans une version de production, il faudrait ajouter une couche de logique pour faire en sorte que les données ici stockées dans le .env soient plutôt requêtées au lancement de l'application cliente ce qui permet une sécurité et une maintenabilité supplémentaire.

# 4 Section Admin

## Objectifs techniques

La section admin regroupe les différentes actions disponibles seulement aux utilisateurs connectés et avec le rôle d'admin. Cela comprend les options d'ajout, de modification et de suppression des projets ainsi que la consultation des statistiques.

Rédigé par J. Issert

### Sécurité

Un utilisateur n'étant pas connecté ou n'ayant pas les bons droits (dans le cas d'une logique avec plusieurs niveaux de droits) ne pourra pas accéder au pages ou composants d'administration grâce à la gestion du React Router. Dans un même temps, les appels API sont également sécurisés. Lorsqu'un utilisateur se connecte dans l'application cliente, son token de connexion JWT est retourné par l'api /api/auth et est stocké dans son "local Storage" et c'est celui-ci qui est ajouté lors de l'envoi de requêtes afin de permettre au backend, à la réception de celles-ci, de vérifier l'authentification et le rôle de l'utilisateur via le middleware d'authentification.

## 5 Éléments additionnels

### Méthode de gestion des images

Le projet utilise le middleware multer pour gérer l'upload des images via multipart/form-data.
Comme évoqué précédemment, les images sont actuellement stockées dans un dossier dans le répertoire du projet. Il faudra dans une version déployée en production, veiller à utiliser un système de stockage distinct pour les images tel qu'un bucket Cloudinary.

### Méthode de gestion des statistiques

Pour cette première version, les statistiques affichées sont requêtées au backend (et calculées en live). Cette méthode n'est pas utilisable en production et il faudrait utiliser un système avec des traitement de batchs qui se lancent à intervalles réguliers dans le serveur afin de les calculer puis les stocker dans une table en base de données afin de pouvoir les récupérer à tout instant en plus de pouvoir implémenter un système d'historique au besoin.

### Possibilités supplémentaires

## 6 Références

1. *Images générées par IA.*
2. *Des requêtes informatives ont été faites à des IA à des fins d'aide théorique.*
3. *Utilisation de la documentation React.*
4. *Utilisation du contenu de cours React de WebDevSimplified.*
5. *Utilisation de la documentation de Node.js.*