Rapport de projet

Développement Web Avancé

Auteurs : Antoine GACHENOT, Thibaut MIDAVAINE, Basile ROSIER

Promotion : FISA A3 2427

Date : 26 juin 2025

CESI - Campus de Nancy

CESI
ÉCOLE D'INGÉNIEURS

Résumé

Dans un contexte où la simplicité, la rapidité et l'accessibilité deviennent des critères essentiels pour les utilisateurs, ce projet vise à concevoir un réseau social léger et réactif, inspiré de Twitter/X, mais pensé pour garantir une neutralité politique et une modération équitable des contenus. L'objectif est de permettre à chacun de publier des messages courts, d'interagir avec la communauté et de profiter d'une expérience fluide, quel que soit le support utilisé.
Pour répondre à ces enjeux, l'équipe a fait le choix de technologies modernes et robustes : un back-end en Node.js avec Express, une architecture microservices, une base de données MongoDB, et un front-end développé avec React. L'ensemble des services est conteneurisé avec Docker, facilitant le déploiement et la gestion des dépendances, tandis que GitLab CI assure l'intégration et le déploiement continus. L'application est hébergée sur OVH, garantissant une disponibilité optimale.
Le projet s'articule autour de fonctionnalités essentielles telles que la création de comptes, l'authentification sécurisée, la publication et la consultation de messages, la gestion des interactions (likes, commentaires, suivis), ainsi que la personnalisation des profils. Des fonctionnalités avancées, comme la gestion des notifications, l'ajout de médias, la recherche par tags ou encore la messagerie privée, viennent enrichir l'expérience utilisateur et renforcer la dimension communautaire.
Au-delà des aspects techniques, ce projet a permis à l'équipe de relever de nombreux défis : gestion des CORS, sécurité des données, automatisation du déploiement, et adaptation à des contraintes d'hébergement. Cette expérience a été l'occasion de mettre en pratique des compétences en développement web avancé, en gestion de projet collaboratif et en résolution de problèmes concrets, tout en posant les bases d'une plateforme évolutive, accessible à tous et soucieuse de la neutralité des échanges.

Abstract

In a context where simplicity, speed, and accessibility are becoming essential criteria for users, this project aims to design a lightweight and responsive social network inspired by Twitter/X, but conceived to ensure political neutrality and fair content moderation. The objective is to allow everyone to post short messages, interact with the community, and enjoy a smooth experience on any device.
To meet these challenges, the team chose modern and robust technologies : a Node.js back-end with Express, a microservices architecture, a MongoDB database, and a front-end developed with React. All services are containerized with Docker, facilitating deployment and dependency management, while GitLab CI ensures continuous integration and deployment. The application is hosted on OVH, guaranteeing optimal availability.
The project is built around essential features such as account creation, secure authentication, posting and viewing messages, managing interactions (likes, comments, follows), and profile customization. Advanced features like notifications, media uploads, tag-based search, and private messaging further enrich the user experience and strengthen the community aspect.
Beyond the technical aspects, this project enabled the team to tackle numerous challenges : CORS management, data security, deployment automation, and adaptation to hosting constraints. This experience was an opportunity to apply advanced web development skills, collaborative project management, and practical problem-solving, while laying the foundations for a scalable platform accessible to all and committed to neutral exchanges.

Table des matières

Chapitre 1

Introduction

1.1 Présentation du sujet

À l'ère du numérique, les réseaux sociaux occupent une place centrale dans la communication et le partage d'informations. Cependant, la majorité des plateformes existantes sont souvent gourmandes en ressources et peu adaptées aux environnements contraints ou aux utilisateurs recherchant simplicité et rapidité. C'est dans ce contexte que notre projet s'inscrit : concevoir et développer un réseau social léger, réactif et accessible, inspiré de Twitter/X, mais optimisé pour fonctionner efficacement sur des dispositifs à faibles ressources.
L'objectif principal est de permettre à chaque utilisateur de publier des messages courts, d'interagir facilement avec la communauté, et de profiter d'une expérience fluide, quel que soit le support utilisé. Pour atteindre ce but, nous avons fait le choix de technologies modernes et robustes : un back-end en Node.js avec Express pour la gestion des données et de la sécurité, un front-end en React pour garantir une interface intuitive et adaptable, et l'utilisation de Docker pour simplifier le déploiement et la maintenance de l'application.
Le projet s'articule autour de fonctionnalités essentielles telles que la création de comptes, l'authentification sécurisée, la publication et la consultation de messages, la gestion des interactions (likes, commentaires, suivis), ainsi que la personnalisation des profils. Des fonctionnalités avancées, comme la gestion des notifications, l'ajout de médias, la recherche par tags ou encore la messagerie privée, viennent enrichir l'expérience utilisateur et renforcer la dimension communautaire de la plateforme.
Ce rapport présente la démarche suivie, les choix techniques réalisés, ainsi que les perspectives d'évolution pour ce réseau social nouvelle génération, pensé pour être à la fois performant, sécurisé et accessible au plus grand nombre.

1.2 Présentation du plan

Ce rapport est structuré de la manière suivante :

- **Étapes du projet** : Présentation des différentes phases de développement, de l'analyse des besoins à la mise en production.
    - Étape 1 : Analyse des besoins du sujet et comment nous avons défini les fonctionnalités principales et secondaires du réseau social.
    - Étape 2 : Conception de l'architecture du projet, incluant le schéma d'architecture et la description des choix techniques.
    - Étape 3 : Prémices du développement et mise en place de l'environnement de travail, ainsi que les premières implémentations des services back-end et front-end.
    - Étape 4 : CI-CD et hébergement de l'application sur OVH, incluant la configuration du pipeline CI/CD avec GitLab et les tests de déploiement.
    - Étape 5 : Développement avancé : Mise en place des fonctionnalités principales du front-end, intégration avec le back-end et tests fonctionnels.
    - Répartition des tâches : Détail des contributions de chaque membre de l'équipe et des outils utilisés pour la gestion de projet.
- **Problèmes rencontrés** : Description des défis techniques et organisationnels rencontrés durant le projet, ainsi que les solutions apportées.
- **Conclusion** : Bilan du projet, perspectives d'évolution et recommandations pour les développements futurs.
- **Bibliographie** : Liste des références et ressources utilisées pour la réalisation du projet.
- **Glossaire** : Définitions des termes techniques et spécifiques au projet.

Chapitre 2

Étapes du projet

2.1 Étape 1 : Analyse des besoins du sujet

2.1.1 fonctionnalités principales

Le projet de réseau social léger s'articule autour de plusieurs fonctionnalités principales, visant à offrir une expérience utilisateur fluide et intuitive. Voici les principales caractéristiques que nous avons identifiées :

- Création de comptes utilisateurs
- Authentification sécurisée (connexion/déconnexion)
- Publication de messages courts : des "Touites"
- Interaction avec la communauté (likes, commentaires, suivis)
- Personnalisation des profils
- Flux de messages : affichage des publications des utilisateurs suivis et non suivis

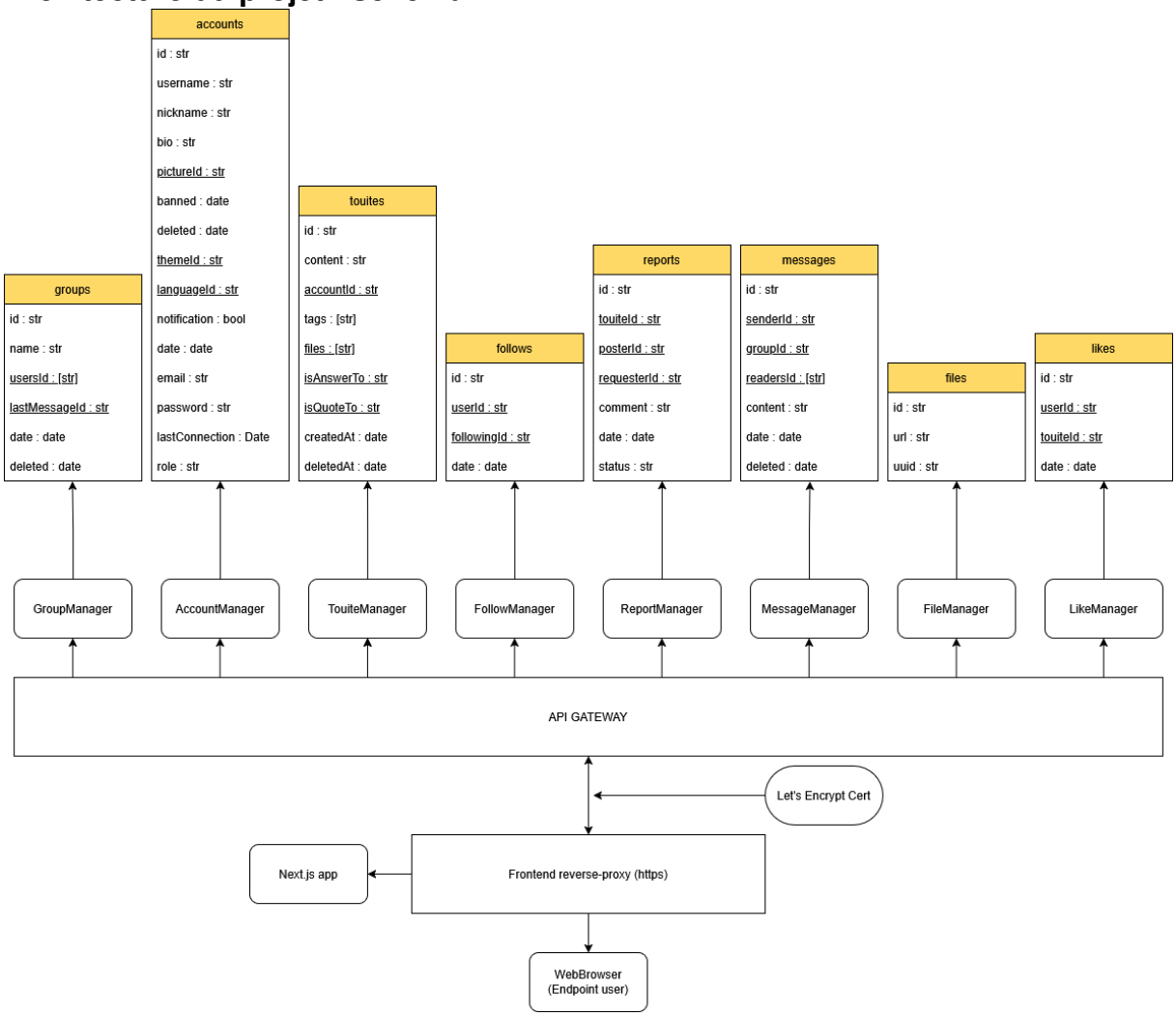
2.1.2 fonctionnalités secondaires

Parmi les fonctionnalités secondaires, nous pouvons envisager :

- Gestion des notifications
- Ajout de médias (images, vidéos)
- Recherche par tags
- Messagerie privée

2.2 Étape 2 : Conception de l'architecture

Architecture du projet : schéma



Architecture du projet : description L'architecture du projet repose sur une séparation claire entre le front-end et le back-end, facilitant ainsi la maintenance et l'évolution de l'application. Le back-end, développé en Node.js avec Express via une architecture microservices, gère les requêtes API, l'authentification des utilisateurs et l'accès à la base de données MongoDB. Le front-end, quant à lui, est construit avec React et communique avec le back-end via des appels API grâce à Axios. Cette approche permet une grande flexibilité et scalabilité, tout en garantissant une expérience utilisateur fluide.

La base de données MongoDB est unique mais chaque service dispose de sa propre collection, permettant une gestion efficace des données. Les services sont conteneurisés avec Docker, ce qui simplifie le déploiement et la gestion des dépendances. Le projet est conçu pour être facilement extensible, permettant l'ajout futur de nouvelles fonctionnalités sans perturber l'architecture existante.

2.3 Étape 3 : Prémices du développement

Les premières étapes du développement ont consisté à mettre en place l'environnement de travail et à configurer les outils nécessaires. Nous avons choisi d'utiliser Docker pour containeriser les différents services, ce qui facilite le déploiement et la gestion des dépendances. Chaque service a été développé de manière indépendante, en suivant les principes de l'architecture microservices.
Nous avons commencé par mettre en place le back-end, en créant les routes API des différents services, les modèles de données pour MongoDB, et en implémentant les fonctionnalités d'authentification. Ensuite, nous avons développé le front-end avec React, en intégrant Axios pour les appels API et en créant les composants nécessaires pour l'authentification.
Ceci nous a permis de poser les bases solides du projet, en nous assurant que chaque service fonctionne correctement avant de passer à l'implémentation des fonctionnalités plus avancées. Nous avons également mis en place un système de gestion des versions avec Git, permettant une collaboration efficace entre les membres de l'équipe.

2.4 Étape 4 : CI-CD et hébergement

Avec l'utilisation de Docker, nous avons pu mettre en place un pipeline CI/CD (Intégration Continue / Déploiement Continu) efficace. De plus, nous avons choisi d'héberger notre application sur OVH, ce qui nous a permis de réaliser les tests de déploiement et de configuration nécessaires pour assurer la disponibilité et la performance de notre réseau social.
Via GitLab CI, nous avons automatisé le processus de build et de déploiement. Chaque fois qu'une modification est poussée dans le dépôt, un pipeline est déclenché pour construire les images Dockers et déployer les services sur notre serveur OVH. Cela garantit que chaque version de l'application est mise dans l'environnement de tests et de production sans intervention manuelle.

2.5 Étape 5 : Développement avancée

Une fois la majorité des services back-end en place et du CI-CD, nous avons commencé à développer les fonctionnalités principales du front-end. En utilisant React, nous avons créé des composants réutilisables pour les différentes parties de l'application, comme la page d'accueil, le profil utilisateur, les messages et les flux de messages.

2.6 Répartition des tâches

La répartition des tâches entre les membres de l'équipe a été réalisé au "compte-goutte", en fonction des compétences et des intérêts de chacun, mais aussi en fonction des besoins lors du développement. Le back-end et le front-end ont été abordés par chacun des membres. Voici un aperçu de la répartition des tâches :

- Antoine GACHENOT :
  - Développement imitial des microservices.
  - Hébergement du projet sur OVH.
  - Mise en place du CI/CD avec GitLab.
  - Vérification et validation des Merges Requests.
  - Développement de la vérification des addresses e-mail.
  - Développement de la gestion des notifications.
- Thibaut MIDAVAINE :
  - Développement initial des microservices.
  - Développement de l'authentification des utilisateurs.
  - Développement de la gestion des "flux infinis" (infinite scroll) pour le fil d'actualités.
  - Développement de la page de profil utilisateur.
- Basile ROSIER :
  - Développement initial du front-end.
  - Développement de la page d'accueil et de la page de connexion.
  - Définition du design et de l'ergonomie de l'application.
  - Développement des composants de touites et de leurs interactions.
  - Développement de la partie administration (tableaux de bord de modération), système de report.

Ce tableau résume les principales contributions de chaque membre de l'équipe. Cependant, il est important de noter que la collaboration a été constante, avec des échanges réguliers pour s'assurer que chaque partie du projet s'intègre bien avec les autres. De plus, il était courant que les membres de l'équipe aident les autres sur leurs tâches respectives, notamment pour le débogage et les tests. Finalement, la répartition des tâches a été flexible, permettant à chacun de contribuer là où il était le plus utile.

Chapitre 3

Problèmes rencontrés

Au cours du projet, nous avons rencontré plusieurs problèmes techniques, notamment :

- Environnement de développement et CI-CD : La configuration de l'environnement de développement et du pipeline CI/CD a été complexe, notamment en raison des différences entre les environnements locaux et le serveur OVH.
- Gestion des CORS : La gestion des CORS (Cross-Origin Resource Sharing) a posé des problèmes lors des appels API entre le front-end et le back-end, nécessitant des ajustements dans la configuration du serveur Nginx et des microservices entre la partie locale et le serveur OVH.
- Hacking de la base de données hébergée : Lors de la mise en place de la base de données MongoDB, nous avons rencontré des problèmes de sécurité liés à l'accès non autorisé aux données. Nous avons dû mettre en place des mesures de sécurité supplémentaires pour protéger les données des utilisateurs.
- Non intuitivité de l'interface d'OVH : L'interface d'administration d'OVH n'était pas très intuitive, ce qui a compliqué la prise en main de la plateforme.
- Enregistrement auprès de la belgique : Le nom de domaine "touiteur.be" a dû être enregistré auprès des autorités belges, ce qui a ajouté une couche de complexité administrative au projet.

Ces problèmes ont été résolus grâce à la collaboration de l'équipe et à la recherche de solutions adaptées. Nous avons appris à mieux gérer les environnements de développement et de production, ainsi qu'à sécuriser les données des utilisateurs.

Chapitre 4

Conclusion

4.1 Bilan

Le projet de développement d'un réseau social léger et réactif, inspiré de Twitter/X, a été une expérience enrichissante et formatrice pour l'équipe. Nous avons réussi à créer une application fonctionnelle, respectant les principes de l'architecture microservices, tout en garantissant une expérience utilisateur fluide grâce à l'utilisation de React pour le front-end et Node.js avec Express pour le back-end. L'utilisation de Docker pour containeriser les services a grandement facilité le déploiement et la gestion des dépendances, permettant une intégration continue efficace via GitLab CI. L'hébergement sur OVH a également été un choix judicieux, offrant une infrastructure fiable pour notre application et nous permettant de réaliser les test des fonctionnalités en conditions réelles.
Nous avons pu implémenter les fonctionnalités principales du réseau social, telles que la création de comptes, l'authentification sécurisée, la publication de messages, et l'interaction avec la communauté. De plus, nous avons mis en place des fonctionnalités avancées comme la gestion des notifications et le système de "flux infinis" pour le fil d'actualités, la messagerie privée, le système de report et la partie administration.
Cependant, il reste encore de nombreuses fonctionnalités à implémenter pour atteindre une version complète du réseau social.

4.2 Perspectives

Le projet étant initialement prévu pour être une maquette, nous avons réussi à développer un réseau social fonctionnel. Cependant, il reste encore de nombreuses fonctionnalités à implémenter pour atteindre une version complète, de plus, nous avons consciemment choisi de ne pas implémenter certaines fonctionnalités clés de la sécurité pour simplifier le projet et se concentrer sur les aspects essentiels du développement web avancé. Voici quelques perspectives d'évolution pour le projet :

- Vérification des tokens JWT, de l'utilisateur et des rôles dans les microservices.
    - Implémenter une vérification des tokens JWT dans chaque microservice pour s'assurer que les requêtes sont authentifiées et autorisées.
    - Vérifier que l'utilisateur a les droits nécessaires pour accéder aux ressources demandées, en fonction de son rôle (utilisateur, modérateur, administrateur).
    - Un utilisateur ne peut pas accéder aux ressources d'un autre utilisateur sans autorisation.

  Nous avons choisi de ne pas implémenter cette fonctionnalité pour simplifier le projet, mais elle est essentielle pour garantir la sécurité et la confidentialité des données des utilisateurs.
- Hébergement de la base de données MongoDB sur un serveur dédié, voire sur un service cloud comme MongoDB Atlas, AWS ou Azure.
- Mise en place du load balancing pour répartir la charge entre plusieurs instances de l'application, améliorant ainsi la scalabilité et la disponibilité.
- Optimisation des performances du front-end, notamment en améliorant le temps de chargement des pages et en réduisant la taille des ressources (images, scripts, etc.).

Bibliographie

[1] X, https://x.com/, consulté régulièrement durant le projet en juin 2025.

[2] Docker Inc. *Docker Documentation*, https://docs.docker.com/, consulté régulièrement durant le projet en juin 2025.

[3] GitLab Inc. *GitLab Documentation*, https://docs.gitlab.com/, consulté régulièrement durant le projet en juin 2025.

[4] Node.js Foundation. *Node.js Documentation*, https://nodejs.org/en/docs/, consulté régulièrement durant le projet en juin 2025.

[5] Meta Platforms, Inc. *React Documentation*, https://react.dev/, consulté régulièrement durant le projet en juin 2025.

[6] OpenAI. *ChatGPT*, modèle de langage basé sur GPT-4, https://chat.openai.com/, consulté régulièrement durant le projet en juin 2025.

Glossaire

Touites Message court publié sur le réseau social, similaire à un tweet.

API Interface de Programmation d'Application, permettant la communication entre le front-end et le back-end.

Docker Outil de conteneurisation permettant de déployer des applications dans des environnements isolés.

Node.js Environnement d'exécution JavaScript côté serveur, utilisé pour développer le back-end de l'application.

React Bibliothèque JavaScript pour construire des interfaces utilisateur, utilisée pour le front-end de l'application.

MongoDB Base de données NoSQL utilisée pour stocker les données de l'application.

Express Framework web pour Node.js, facilitant la création de serveurs et la gestion des routes.

OVH Fournisseur d'hébergement web où l'application est déployée.

GitLab Plateforme de gestion de code source et de collaboration, utilisée pour le versionnage du projet.

CI/CD Intégration Continue / Déploiement Continu, pratiques de développement logiciel visant à automatiser le processus de test et de déploiement des applications.

GitLab CI Outil d'intégration continue permettant d'automatiser le processus de build et de déploiement de l'application.

Ngnix Serveur web utilisé pour gérer les requêtes HTTP et HTTPS, ainsi que pour la configuration des CORS.