Document Technique

ECF 2023

Quai Antique

Rédacteur : BOUCLET Florian

Spécifications techniques

Serveur :

- Docker
- GraphQl / PostgreSQL

Front End :

- Reactjs
- CSS

Back End :

- NestJs
- TypeOrm

Outils :

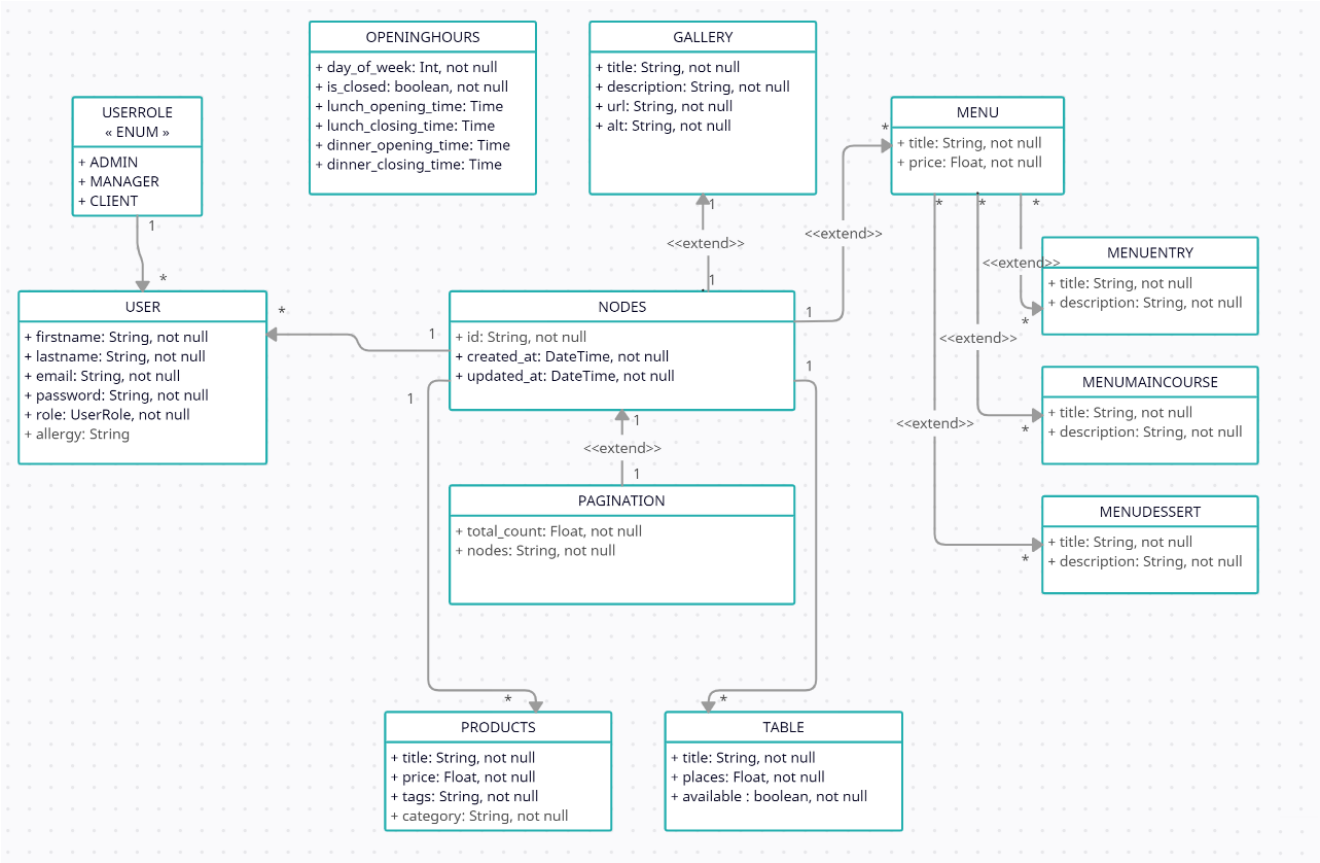
- GitHub
- Figma
- Docker
- VsCode
- GraphQl Playground
- Trello

Déploiement :

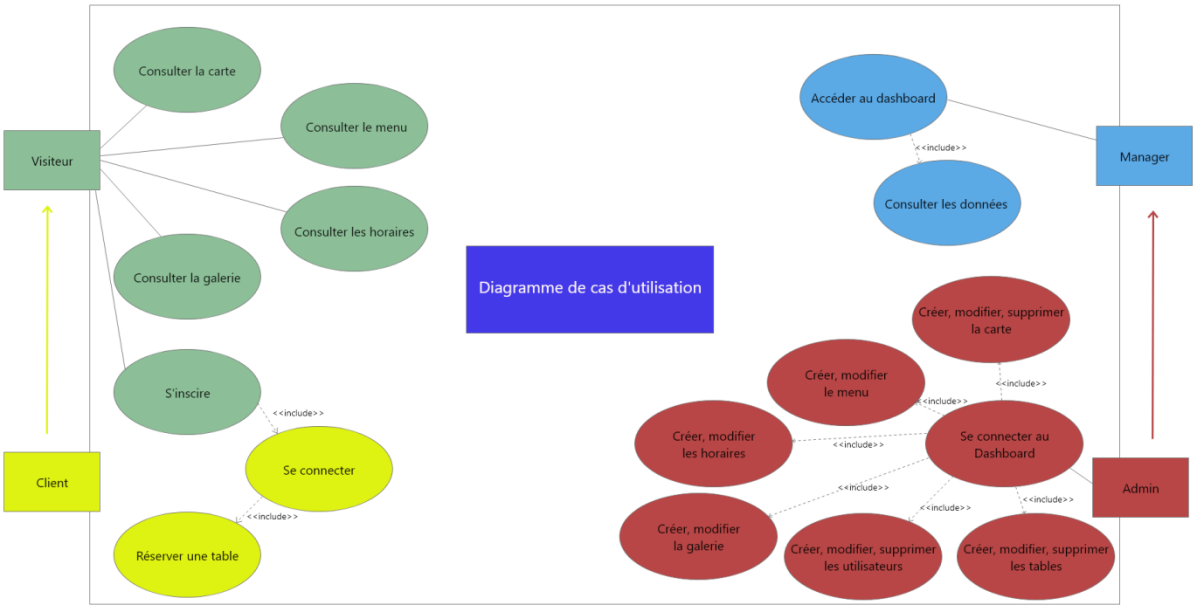
- GitHub
- Netlify
- Clever Cloud

Les Diagrammes

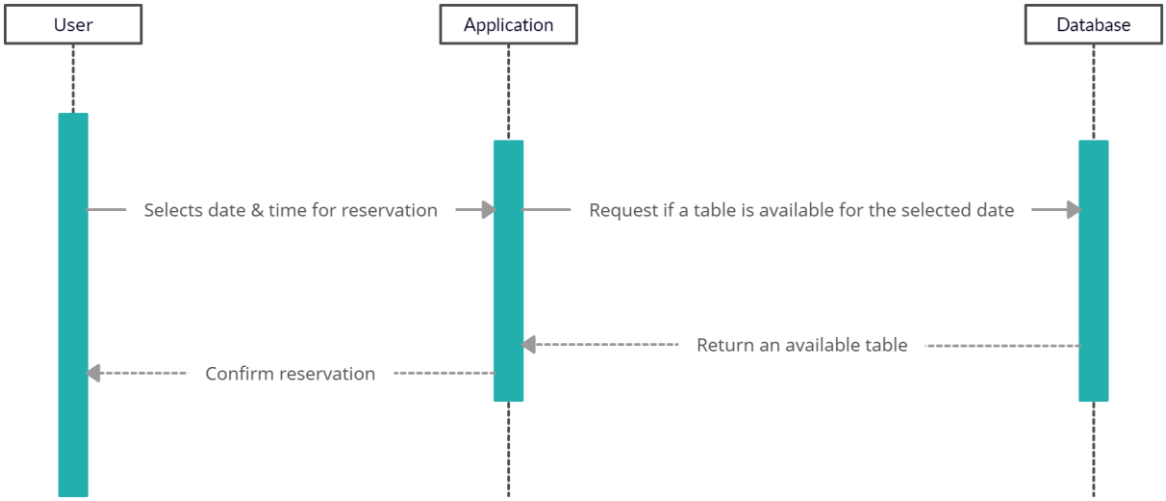
1. Diagramme de classes



2. Diagramme de cas d'utilisation



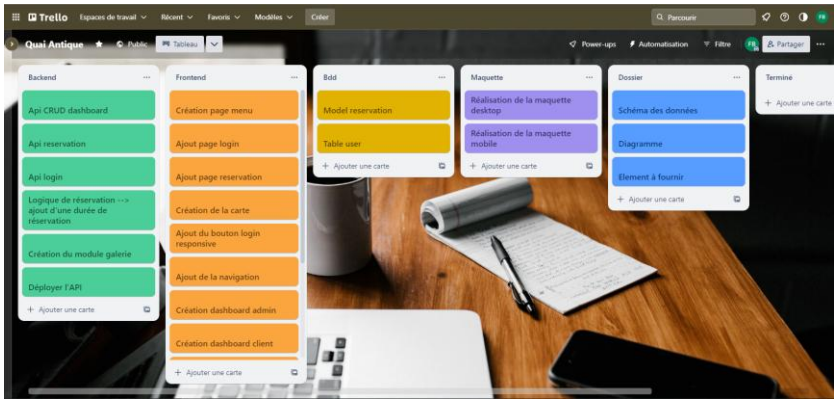
3. Diagramme de séquence

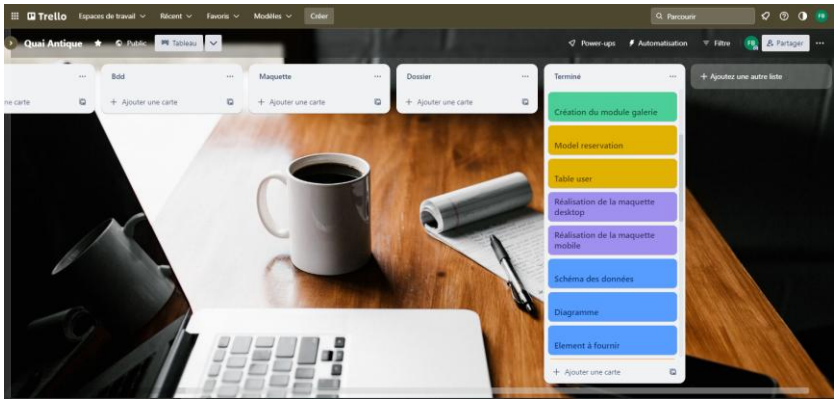


Organisation

Concernant l'organisation du projet j'ai utilisé Trello qui est un gestionnaire de tâches utilisant la méthode KANBAN. Cet outil m'a permis de planifier les grandes lignes de l'application. Ci-dessous vous pourrez retrouver une partie de mon espace de travail à titre démonstratif.

Lien Trello : https://trello.com/b/VGLINIOT/quai-antique





Choix des technologies

Pour la réalisation de ce projet, j'ai choisi de concevoir une application fullstack en utilisant JavaScript. Pour le backend, j'ai opté pour NestJs, un framework basé sur Node.js, avec une architecture GraphQL.

Le choix de GraphQL s'est imposé pour plusieurs raisons. Tout d'abord, GraphQL permet un rendu dynamique des données en limitant les requêtes au serveur. Contrairement à une API REST traditionnelle, où chaque requête peut renvoyer des données supplémentaires non nécessaires, GraphQL permet de spécifier précisément les données requises pour chaque requête. Ainsi, l'application reçoit uniquement les données nécessaires, évitant les surcharges de réseau inutiles et améliorant les performances globales.

De plus, GraphQL facilite la création de relations complexes entre les données. Au lieu de multiplier les requêtes pour récupérer des informations liées, il permet de définir des schémas de données clairs et de récupérer tous les éléments nécessaires en une seule requête. Cela permet d'optimiser les performances et de réduire la charge sur le serveur et la base de données.

Pour le frontend, j'ai choisi ReactJS, une bibliothèque JavaScript très populaire pour la construction d'interfaces utilisateurs interactives. L'un des principaux avantages de ReactJS est sa gestion efficace du « virtual DOM ». Plutôt que de recharger toute la page à chaque modification, ReactJs met à jour uniquement les éléments nécessaires du DOM, offrant ainsi une expérience utilisateur plus fluide et réactive.

En résumé, le choix de ces technologies est une combinaison puissante pour le développement d'une application fullstack JavaScript. Cette approche permet d'optimiser les performances, de simplifier la gestion des données et de créer des interfaces utilisateur interactives, réactives et sécurisées.