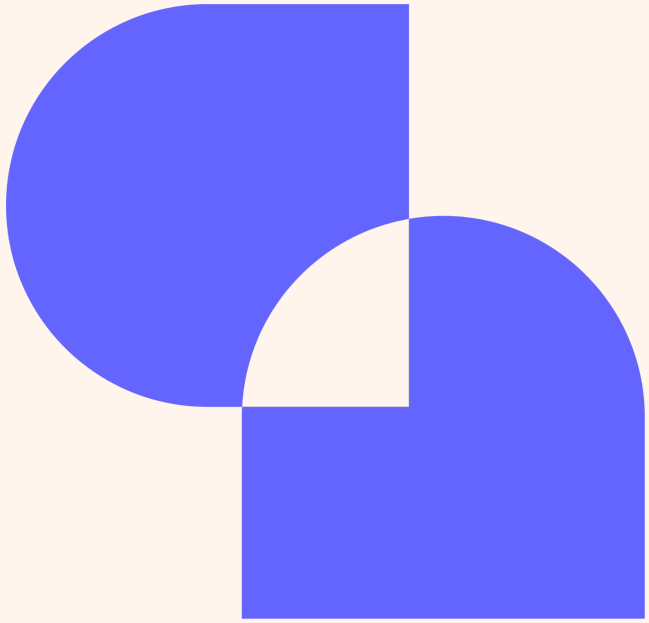
GoMind

“Agir avec l’esprit éveillé”

DOSSIER DE COMPÉTENCES

Développeur fullstack ReactJS/NodeJS

Spécialisé : NextJS



PRINCIPALES EXPÉRIENCES

DASSAULT : Développeur ReactJS/NodeJS (2020-2025)
BORNSAFETY : Développeur ReactJS/NodeJS (2019)
CITADELLE : Développeur ReactJS/NodeJS (2018)

CONNAISSANCES TECHNIQUES

Langages : HTML, CSS, Sass, JavaScript ES5/ES6, TypeScript, PHP

Librairies/framework :
Frontend : React.JS, NextJS, Hooks, Context API, HOC, Custom Hooks, Pure Component/Function, Styled-components, React Form , Redux, Redux Toolkit, Redux Saga, Zustand, Redux Off-line, React Query, Lazy ReactJS, Material-UI, Tailwind CSS, Lodash, Tanstack form, Tanstack virtual, Radix primitive, ShadCN, Tanstack table, Xstate store, Recharts, D3.js

Backend : NodeJS, Express, TypeScript, Socket.IO, Mongoose, Express Validator,.RabbitMQ, Multithreading,

Architecture/Design Pattern : Document Object- Model, SGBD, MicroServices, UML, Template, Singleton, Architecture Hexagonale, Container and Presentational Components, Middleware Pattern, Event-Driven Architecture, API Gateway Pattern, Serverless

Base de données : MySQL, MongoDB, Firebase, SQLite, PostgreSQL

Bonnes pratiques : Jest, Mocha, ChaiJS, TDD, SOLID, Linting, Documentation, Pair Programming, Refactoring,

Outils : Git, GitHub, Jenkins, Bitbucket, Postman, Swagger

Gestion de projet : Confluence, JIRA, Agile

EXPÉRIENCES TECHNIQUES

DASSAULT : Développeur ReactJS/NodeJS (2020-2025)

Refonte from scratch d'un logiciel qui permet d'assister les ingénieurs dan la maintenance des rafales

- Utilisation de maquettes
- Équipe de 30 développeurs :
- Gestion de projet en SCRUM :
    - Cérémonies : daily, démo, rétro
    - Participation à la présentation et au chiffrage des tâches
    - Écriture d'US sur Jira avec l'aide des PPO
    - Participation et animation d'entretien technique pour présenter des librairies, paternes, et pratiques qui pourrait bénéficier le projet
- Mise en place de l'architecture Microservices :
    - Un serveur d'API, un serveur de notifications (Socket.IO), une applications d'exécution de tâches de fond en multithreading, une application ReactJS
    - Mise en place de l'architecture en couches
    - Pratique du clean code (DeadCode, SOLID...)
    - Utilisation des hooks (Redux, Style, React Router)
- Développement d'application en ReactJS 16/Redux :
    - Développement d'application en ReactJS 16
        - Utilisation des Hooks (Redux, Material-UI Style, React Router)
        - Gestion du multilingue du site (les langues : FR, EN, ...)
        - Utilisation de composants Material-UI pour intégrer toutes les applications React
        - Création d'une librairie de composants interne génériques pour les applications, basé sur Material-UI
        - Utilisation de React Context API pour plusieurs composants
        - Gestion de la connexion avec le Microservice de notifications à l'aide de Socket.IO, Redux, et Redux Saga
        - Création de CRUD pour les données de l'application
        - Créations de graphes avec Recharts et D3.js
        - Mise à jour des données en temp réel avec Socket.IO
- Développement NodeJS :
    - Mise en place d'un système de filtrage avec pagination dans plusieurs routes du serveur d'API, facilitant la recherche par plusieurs mots-clés
    - Conception et développement de services RESTful, incluant l'implémentation de middlewares pour assurer la sécurité, la

validation des requêtes et la gestion des erreurs
    - Gestion des middlewares
    - Utilisation du Logger pour la requêtes API
    - Utilisation des Try/catch
    - Utilisation de Promise
- Versionning sur Git
- Mise en place des tests unitaires:
    - Mocha/Chai pour les services NodeJS
    - Jest/Chai pour l'application en ReactJS
- Utilisation de React tools et Redux tools pour le debug et affichage des states dans le store
- Migration de CRA vers Vite, version React (16 vers 18), version Material-UI (4 vers 5)
- Utilisation de Jenkins
- Bonnes pratiques :
    - Écriture de documentation du serveur d'API avec Swagger
    - Pair programming
    - Écriture de documentation et bonnes pratiques techniques divers avec des fichiers Markdown

Environnement technique :
Microservices, Clean Code, Solid, ReactJS, Socket.IO, Redux Toolkit, Material-UI, TypeScript, NodeJS, Express, MongoDB, RabbitMQ, Swagger, Jenkin, Jest/Mocha/Chai, Bitbucket, Jira, Confluence, Scrum

---

BORNSAFETY : Développeur ReactJS/NodeJS (2019)

Création from scratch d'un site pour regarder des vidéos

- Analyse du besoin
- Rédaction et Suivi des spécifications techniques et fonctionnelles
- Équipe de 2 développeurs :
- Gestion de projet en SCRUM :
    - Participation à la présentation et au chiffrage des tâches
    - Écriture d'US sur ClickUp
- Mise en place de l'architecture Monolithique
    - Un serveur d'API, une application ReactJS
    - Pratique du clean code (DeadCode, SOLID...)
    - Utilisation de React Pattern Design Container/Presentational pour la gestion des composants et séparer la couche UI
    - Utilisation des hooks (Redux, React Router)
- Développement d'application en ReactJS 16/Redux :
    - Développement d'application en ReactJS 16
        - Utilisation des Hooks
        - Utilisation de composants Material-UI pour intégrer toutes l'application React
        - Création des composants génériques pour les applications en utilisant Material-UI

        - Création de CRUD pour les données de l'application
        - Créations de graphes avec Recharts
- Développement NodeJS :
    - Mise en place d'un système de filtrage avec pagination dans plusieurs routes du serveur d'API
    - Conception et développement d'un serveur RESTful, incluant l'implémentation de middlewares pour assurer la sécurité, la validation des requêtes et la gestion des erreurs
    - Gestion des middlewares
    - Utilisation des Try/catch
    - Utilisation de Promise
- Versionning sur Git
- Utilisation de React tools et Redux tools pour le debug et affichage des states dans le store

Environnement technique :
Microservices, Clean Code, Solid, ReactJS, Redux, TypeScript, MongoDB, Github, ClickUp, Scrum

---

CITADELLE : Développeur ReactJS/NodeJS (2018)

Création from scratch d'un jeu (ex clash of clan)

- Analyse du besoin
- Rédaction et Suivi des spécifications techniques et fonctionnelles
- Équipe de 4 développeurs :
- Gestion de projet en SCRUM :
    - Cérémonies : daily, démo, rétro
    - Écriture d'US
- Mise en place de l'architecture Monolithique
    - Un serveur d'API, une application ReactJS
    - Pratique du clean code (DeadCode, SOLID...)
    - Utilisation de React Pattern Design Container/Presentational pour la gestion des composants et séparer la couche UI
    - Utilisation des hooks (Redux, React Router)
- Développement d'application en ReactJS 16/Redux :
    - Développement d'application en ReactJS 16
        - Utilisation des Hooks
        - Utilisation de composants Material-UI pour intégrer toutes l'application React
        - Création des composants génériques pour les applications en utilisant Material-UI
        - Création de CRUD pour les données de l'application
        - Gestion de la connexion avec le serveur d'API à l'aide de Socket.IO/Redux
        - Création de CRUD pour les données de l'application
        - Mise à jour des données en temp réel avec Socket.IO
- Développement NodeJS :

    - Mise en place d'un système de filtrage avec pagination dans plusieurs routes du serveur d’API
    - Conception et développement d’un serveur RESTful, incluant l'implémentation de middlewares pour assurer la sécurité, la validation des requêtes et la gestion des erreurs
    - Gestion des middlewares
    - Utilisation des Try/catch
    - Utilisation de Promise
- Versionning sur Git
- Utilisation de React tools et Redux tools pour le debug et affichage des states dans le store
- Bonnes pratiques :
    - Pair programming

Environnement technique :
Microservices, Clean Code, Solid, ReactJS, Socket.IO, Redux, TypeScript, NodeJS, ExpressJS, MongoDB, Github, Scrum

FORMATION

LANGUES

Français

Anglais