PROJET / MISSION ENTREPRISE 2023 - 2034

Efrei – Mastère 1 Développeur Manager Full Stack

Réalisé par :

Jong Hoa CHONG (Alternant)

El Mehdi BOUMHICHA (Lead Développeur & Tuteur)

Céleste BRONZETTI (Product Owner)
jc.rangon.formateur@gmail.com

Dédicace

A ma chère sœur,

Même si nous nous disputions souvent lorsque nous étions enfants, ta patience et ta tolérance envers moi étaient remarquables. Maintenant que j'ai grandi, je prends conscience de la chance que j'ai de t'avoir comme sœur. Je te suis très reconnaissant pour tout ce que tu as fait pour moi. Sache que je serai toujours là pour toi, prêt à t'apporter mon soutien et mon amour, peu importe les circonstances. Merci d'avoir été une sœur aimante, attentionnée et dévouée.

A ma mère,

Cette dédicace est dédiée à une femme extraordinaire, une mère qui a sacrifié sans réserve pour ses enfants. Seule, elle a élevé son fils et sa fille, travaillant ardemment pour subvenir à leurs besoins et leur offrir un foyer. Arrivée en France, dans cette terre étrangère sans amis ni famille, elle ne parlait pas un mot de français. Malgré cela, travaillant jusqu'à soixante heures par semaine, elle s'est efforcée de garantir à ses enfants une vie meilleure.

Grâce à sa détermination, son dévouement et son amour inconditionnel, son fils a réussi. Il a persévéré et a concrétisé ses rêves, sachant toujours que sa mère était là pour le soutenir.

Cette femme mérite l'admiration et la gratitude du monde entier pour tout ce qu'elle a accompli. Elle est une source d'inspiration pour tous ceux qui ont eu la chance de la rencontrer, incarnant la puissance de l'amour maternel.

Chère mère, nous te remercions pour tous les sacrifices consentis pour tes enfants. Nous t'aimons plus que tout au monde et nous te serons éternellement reconnaissants pour tout ce que tu as fait pour nous.

Table des matières

Résumé

Ce document représente l'association dans laquelle j'effectue mon alternance. Je commencerai par présenter l'association, puis aborderai les outils de gestion de projet, et enfin la stratégie de test. Pendant mes deux années au sein de l'association, nous avons connu des changements de délégué général et de conseil d'administration. Actuellement, nous sommes en plein changement de stratégie et de vision, ce qui implique le développement d'un nouveau produit.

En tant qu'assistant développeur web junior sous la supervision du lead développeur, ma mission au sein de l'association est polyvalente. Elle englobe plusieurs aspects, tels que :

- L'évolution du frontend du site mavoie.org
- La conception de nouveaux projets
- L'amélioration des fonctionnalités existantes
- Le référencement naturel (SEO)
- La maintenance du frontend et du backend
- La maîtrise du pipeline et des livraisons dans les différents environnements.

De plus, je contribue à enrichir et à affiner le backlog à travers les rituels Scrum. Je réalise des démonstrations des nouvelles fonctionnalités à l'équipe interne et aux partenaires. J'effectue des études techniques sur les nouvelles demandes de produits et je rédige des documents explicatifs à leur sujet. J'ai également la charge de générer des indicateurs clés à partir de la base de données et d'en effectuer une première analyse. Je mets également en place des tests unitaires sur les fonctionnalités développées et je m'assure de leur exécution correcte.

Technologie et outils - Stack

- Frontend :
    - Interface d'utilisateur : React JS
    - Design : Sass, Material-UI
    - Management du state : Redux
    - Typage statique : Flow JS
    - Gestion des Formulaires : Formik
    - Validations des formulaires : Yup
    - Framework de tests : Jest
    - Build : react-app-rewired
- Github : hosting du code et gestion de versionning et collaboration du code
- Github Actions : gestion pipelines de livraisons
- Firebase Services :
    - Firebase Hosting: Serveur d'hébergement du site
    - Firestore : Base de données nosql
    - Cloud functions : Blocs serverless qui permet d'exécuter un code dans un environnement donné (par exemple nodejs, ...) et déclencher des évènements.
- Godaddy : Fournisseur du nom de domaine et gestion du dns.
- Slack : Plateforme de communication
- Figma : Application de conception graphique pour des maquettes

Mission 1 - Présentation : MaVoie.org c'est quoi ?

Introduction

MaVoie.org, c'est l'association d'insertion professionnelle qui veut casser l'isolement et l'invisibilisation des jeunes générations en recherche d'emploi.

Lancé en 2020 autour d'un consortium de plusieurs acteurs de l'insertion (Chance, Bayes Impact, Article 1), MaVoie.org propose une expérience unifiée et personnalisée d'accompagnement en ligne vers l'emploi. Autour d'une même plateforme, chaque jeune utilisateur bénéficie des meilleures ressources de l'insertion professionnelle. Alliant outils de connaissance et de confiance en soi, propositions de formation et préparation aux entretiens, MaVoie.org accompagne ses jeunes bénéficiaires de A à Z. Les solutions qu'on propose permettent de :

1. **Mieux se connaitre**
2. **Se former à un métier**
3. **Savoir Postuler**
4. **Convaincre en entretien**

Au-delà d'un accompagnement repensé, MaVoie.org change la manière de cibler les bénéficiaires isolés. À travers ses maraudes numériques, notamment via les réseaux sociaux, MaVoie.org capte et retient l'attention des jeunes isolés mais hyper connectés.

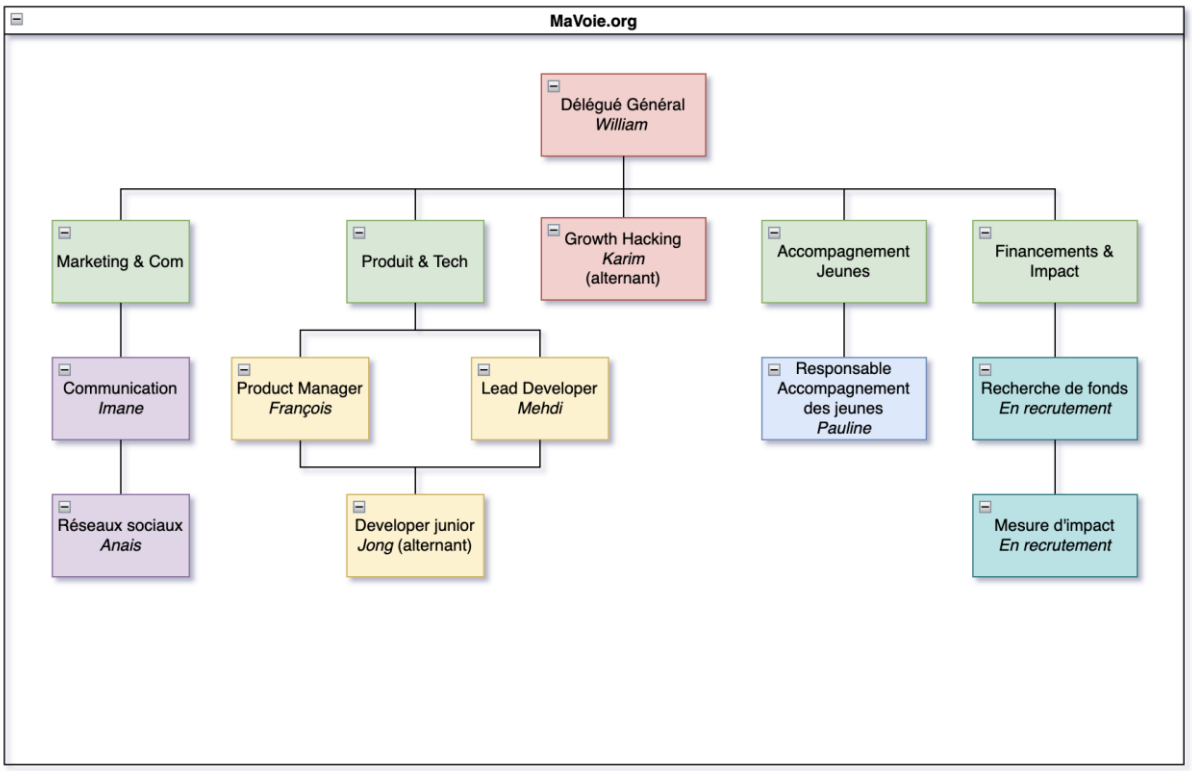
Depuis 4 ans, MaVoie.org a accompagné plus de 50 000 jeunes vers l'emploi, dont près de 60% de niveau baccalauréat ou inférieur et deux tiers de femmes.

MaVoie.org, association de loi 1901, bénéficie du soutien historique de plusieurs grands acteurs de l'insertion professionnelle, du mécénat et de la philanthropie. Parmi ces soutiens, on retrouve notamment Google.org ou la BPI.

1. La fiche signalétique

Siège social	27 bd Saint Martin 75003 Paris
Date de création	03-11-2020
Effectif	10
Directeur général	William ELLAND-GOLDSMITH
Site Web	Mavoie.org
Forme juridique	Association déclarée

2. Organigramme



Architecture du projet MaVoie.org

L'application MaVoie est une application web responsive qui s'adapte aux appareils mobiles à 70% et aux ordinateurs de bureau.

Il s'agit d'une application RestFull Serverless qui n'est pas connectée à un serveur backend, mais qui utilise plutôt des services Firebase (fonctions, Firestore, ...) en utilisant l'architecture Rest, ce qui signifie que la communication entre le front-end et le back-end se fait via des appels à l'aide du protocole.

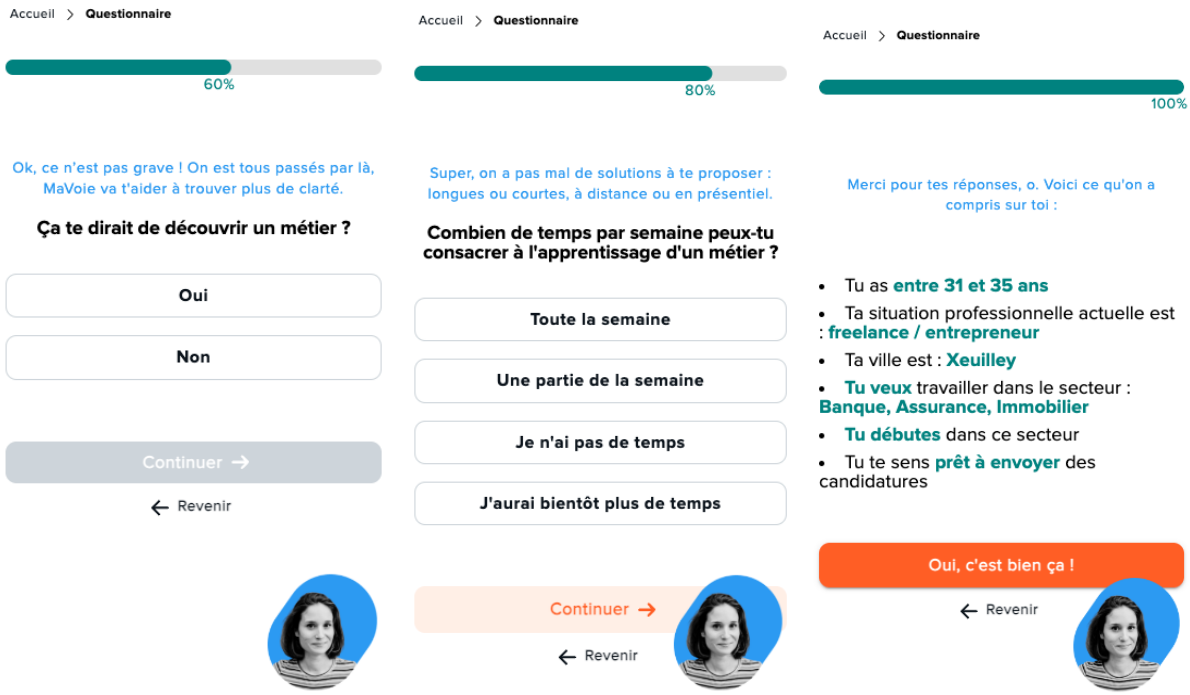
Projets dans MaVoie :

1. La Plateforme MaVoie : Site principal de MaVoie pour diagnostiquer les jeunes et leur proposer des solutions correspondant à leur besoin.
2. Module 1 entretien 1 job : Un outil d'entrainement à 1 entretien d'embauche développé comme sous-module de la plateforme principale.
3. BackOffice : Site d'admin pour gérer les utilisateurs/contenus du site de mavoie.

    - Gestion de Solutions : Module qui permet l'Ajout/Edition/Publication des solutions dans le site principal de MaVoie.

1. Diagnostique MaVoie

MaVoie propose un questionnaire permettant de diagnostiquer l'utilisateur. En se basant sur ce diagnostic, quelques questions lui sont posées afin d'établir un profil personnalisé. En fonction des résultats obtenus, des solutions de formation adaptées à ses besoins sont proposées.

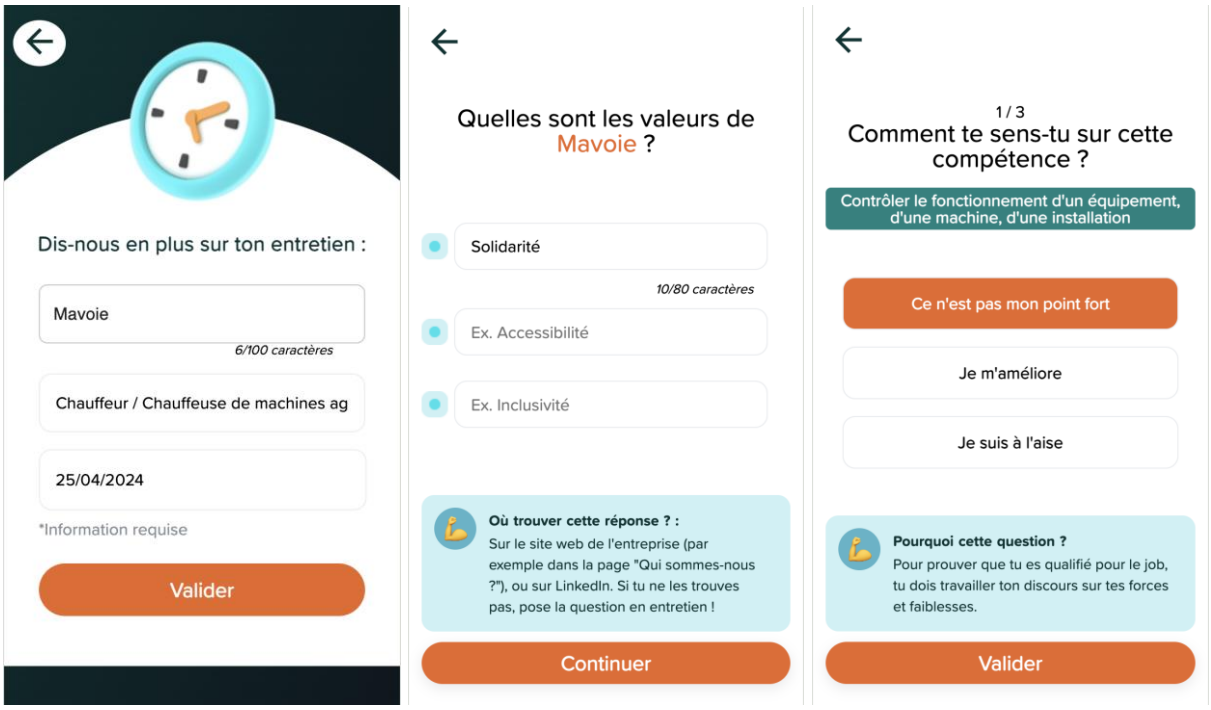


Le questionnaire comporte plusieurs étapes où nous collectons les informations suivantes :

- Son âge.
- Sa situation professionnelle actuelle.
- Sa ville de résidence.
- Son secteur dans lequel la personne souhaiterait travailler.
- S'adresse électronique.
- Son code postal.
- Sa disponibilité pour postuler à des offres d'emploi.
- Son besoin éventuel de formation.
- Son intérêt pour la découverte d'un nouveau métier.
- Sa disponibilité en temps libre pour suivre une formation professionnelle.
- Son besoin d'accompagnement ou d'orientation professionnelle.

2. 1entretien 1job

1entretien 1job (1e1j) est un outil développé par MaVoie qui s'adresse à tous les bénéficiaires de MaVoie. L'objectif du projet est de permettre aux bénéficiaires de s'entrainer à mettre en valeur les bons éléments pendant l'entretien : ne pas improviser les réponses. En s'entrainant, les bénéficiaires gagnent de la confiance et permet aux recruteurs d'avoir des candidatures de qualité. Les informations récupérées sur les bénéficiaires vont orienter les choix et les évolutions fonctionnels futures de MaVoie.



L'entrainement est organisé autour de 3 objectifs principaux :

- Se renseigner sur l'entreprise.
- Comprendre le poste.
- Présenter son profil.

Chaque objectif se compose de quelques tâches à remplir : réponses à des questions ouvertes, activité de recherche d'information sur internet et prise de notes. Il est possible d'interrompre l'entrainement et de le continuer plus tard.

L'idée est de pousser les bénéficiaires à le compléter en leur donnant tout de suite une vision concrète et accessible des objectifs de chaque partie.

3. Outils Logiciels (Front)

I. Firebase Services

Firebase, développé par Google, est une plateforme de développement d'applications mobiles et web offrant une gamme complète de services backend. Parmi ses fonctionnalités, on retrouve le stockage en nuage, l'authentification utilisateur, la base de données en temps réel, la messagerie en temps réel, l'analyse, les notifications push, et bien plus encore. En simplifiant la gestion de l'infrastructure sous-jacente, Firebase permet aux développeurs de créer des applications efficaces, évolutives et sécurisées.

II. Interface d'utilisateur : ReactJS

ReactJS est une bibliothèque JavaScript open-source, développée par Facebook, qui permet de créer des interfaces utilisateur interactives pour les applications web. Utilisée par de nombreuses entreprises, ReactJS se distingue par la création de composants réutilisables qui favorisent la modularité et la scalabilité des interfaces utilisateur. Son approche déclarative simplifie la gestion des états, permettant aux développeurs de se concentrer sur la logique métier.

III. Design System: Material-UI

Material-UI est une bibliothèque de composants React qui permet aux développeurs de créer rapidement et facilement des interfaces utilisateur modernes en suivant les principes du Material Design de Google. Avec une vaste sélection de composants pré-conçus et une personnalisation facile, Material-UI est un choix populaire pour le développement d'applications web réactives.

IV. Design : Sass

Sass (Syntactically Awesome Style Sheets) est un préprocesseur CSS qui enrichit le langage CSS avec des fonctionnalités telles que les variables, les boucles et d'autres outils de programmation. Il permet d'écrire des feuilles de style de manière plus rapide et plus facile, de les diviser en plusieurs fichiers, de générer des styles pour différents environnements, et de réutiliser des styles entre différents projets. Sass est compilé en CSS avant d'être utilisé dans un site web ou une application.

V. Management des States : Redux

Redux est une bibliothèque open-source de gestion d'état pour les applications JavaScript. Il centralise l'état global dans un "store" accessible de manière prévisible, facilitant la gestion des applications complexes. Les modifications de l'état se font par des actions explicites, rendant l'application plus prévisible et testable.

VI. Typage Statique : FlowJS

Flow est un outil statique d'analyse de typage pour JavaScript. Il permet de détecter les erreurs de typage avant l'exécution du code, ce qui peut aider à améliorer la qualité du code et à éviter les erreurs de runtime. Flow est principalement utilisé pour les projets JavaScript de grande envergure, où la complexité du code rend difficile la maintenance et la détection des erreurs de typage. Flow est développé par Facebook et est open source.

VII. Gestion des formulaires : Formik

Formik est une bibliothèque de gestion de formulaire pour React. Elle fournit un moyen simple et efficace de gérer les données de formulaire et les validations, tout en évitant la nécessité de gérer manuellement les événements de formulaire. Formik fournit également des fonctionnalités utiles telles que la gestion de l'état du formulaire, la gestion des erreurs de validation, la gestion de la soumission du formulaire et la gestion de la mise à jour des champs de formulaire en temps réel.

VIII. Validations des formulaires : Yup

Yup est une bibliothèque JavaScript qui permet de créer des schémas de validation de données. Elle est souvent utilisée en conjonction avec Formik pour valider les formulaires dans les applications web. Yup fournit des méthodes pour définir des règles de validation telles que la vérification des types de données, la validation des chaînes, la validation des nombres, etc. Ces règles sont définies en utilisant une syntaxe simple et expressive qui permet de créer des schémas de validation complexes et personnalisés.

4. Outils Logiciels (Back)

I. Plateforme côté serveur : NodeJS

Node.js est une plateforme logicielle libre en JavaScript qui permet d'exécuter du code JavaScript côté serveur. Il utilise le moteur JavaScript V8 de Google pour exécuter le code en dehors du navigateur web. Node.js offre une architecture événementielle et orientée vers les modules, ce qui facilite la création d'applications web rapides et évolutives. Il est également utilisé pour créer des outils en ligne de commande, des applications de streaming, des applications de réseau, des applications de base de données, des robots de chat et bien plus encore. Node.js est open source et est disponible gratuitement pour une utilisation commerciale et non commerciale.

II. Framework Backend : ExpressJS

ExpressJS est un framework minimaliste pour Node.js qui permet de développer des applications web du côté serveur en utilisant le protocole HTTP. Il fournit une multitude de fonctionnalités pour la création d'API RESTful, le routage, le traitement des requêtes et des réponses, la gestion des sessions et des cookies, la gestion des erreurs, etc. ExpressJS est très populaire dans l'écosystème Node.js et est souvent utilisé en combinaison avec d'autres bibliothèques et frameworks tels que MongoDB, Mongoose, Socket.IO, etc.

III. Framework Backend : NestJS

NestJS est un framework backend progressif pour Node.js, construit avec TypeScript. Il facilite la création d'applications serveur robustes et évolutives en utilisant une architecture modulaire et des concepts de programmation orientée objet, fonctionnelle et réactive. Basé sur Express, NestJS intègre facilement d'autres bibliothèques et frameworks, offrant une structure claire et maintenable pour des projets de toutes tailles.

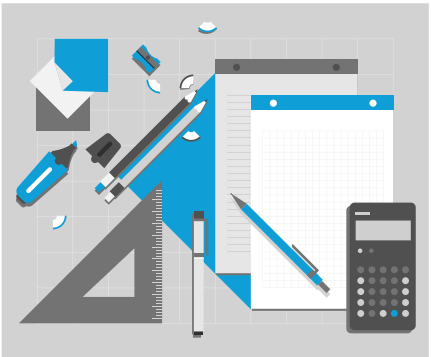
Perspectives Personnelles – Mission 1

En travaillant sur la première mission, j'ai eu l'opportunité d'approfondir ma compréhension de nombreux aspects du développement informatique et de la gestion de projets. Mes perspectives personnelles à la suite de cette expérience :

- **Évolution de Mes Compétences Techniques :** La mission m'a permis de renforcer mes compétences techniques. J'ai eu l'occasion de travailler sur des outils et des technologies variés, ce qui m'a permis de mieux comprendre leur utilité et leur application pratique.
- **Importance de la Collaboration :** J'ai appris que la collaboration est essentielle dans tout projet informatique. Travailler en étroite collaboration avec mon collègue m'a permis de mieux saisir l'importance de la communication et de la coordination. Les échanges d'idées et les résolutions de problèmes ensemble ont non seulement facilité notre travail, mais ont également enrichi mon expérience professionnelle.
- **Résilience Face aux Défis :** Cette mission m'a également enseigné l'importance de la résilience. Les défis et les obstacles sont inévitables, mais il est crucial de ne pas abandonner face aux premières difficultés. Trouver des solutions alternatives et rester persévérant sont des compétences essentielles que j'ai développées au cours de cette mission.
- **Vision et Innovation :** En participant à cette mission, j'ai pu observer l'impact des nouvelles technologies, notamment l'IA, sur notre travail. Cela m'a ouvert les yeux sur l'importance d'avoir une vision claire et de toujours être à l'affût des innovations technologiques. La période de changement que nous traversons est excitante et me motive à continuer à apprendre et à m'adapter aux évolutions rapides de notre domaine.

J'entrevois plusieurs façons de contribuer à l'amélioration de notre association :

1. **Mise en Place de Salesforce :** Pour une meilleure gestion des relations clients et l'automatisation des processus.
2. **Formation Continue :** Organiser des ateliers de formation réguliers pour renforcer les compétences de l'équipe.
3. **Adoption de Nouveaux Outils :** Intégrer des outils de collaboration et de gestion de projet pour améliorer notre organisation.
4. **Veille Technologique :** Encourager la veille technologique et lancer des projets pilotes pour tester de nouvelles idées.



Mission 2 – Outils de gestion de projet

1. Planification du projet

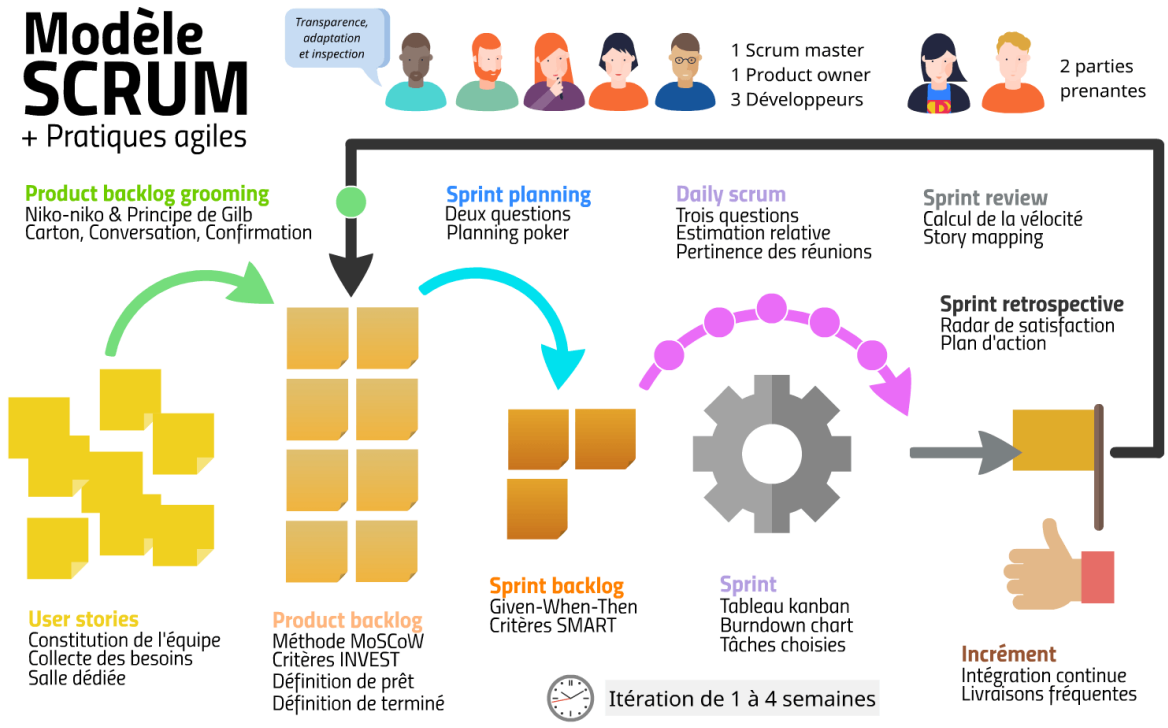
Etapes du projet	Description	Personnes impliquées	Contributeur	Jours
Design	Identification de la cible et du problème à résoudre, définition du périmètre fonctionnel, benchmark graphique, design du workflow, découpage en écrans, brief graphique, maquettage, allers-retours produit-UX	Celeste, Chloé	Philippe	15
Etudes technique	Analyse et documentation des fonctionnalités : récupération d'informations sur les métiers, informations légales et financières des entreprises, recherche des sites et comptes sociaux officiels des entreprises	Mehdi, Jong	Celeste	3
Implémentation BackEnd	Setup du serveur BackEnd (cloud functions), architecture des données (Firestore), création des opérations sur la base de données, liaison Backend-FrontEnd	Mehdi, Jong	Celeste	15
Implémentation FrontEnd	Intégration de la charte graphique (HTML et Sass), création de 15 à 20 écrans, gestion des informations utilisateur (Redux), enregistrement des réponses utilisateurs, gestion des événements des formulaires	Mehdi, Jong	Celeste	15

2. Méthode de Gestion de projet

La méthodologie agile est une approche qui met l'accent sur la flexibilité, la collaboration, la communication continue et l'adaptation aux changements tout au long du développement. Son objectif est de fournir des résultats de haute qualité en favorisant l'interaction entre les membres de l'équipe et en s'ajustant aux besoins changeants du projet. Agile encourage également la transparence, l'apprentissage continu et la livraison régulière de produits fonctionnels.

Scrum, de son côté, est une méthodologie spécifique de gestion de projet axée sur la planification et l'exécution des tâches en courtes itérations appelées "sprints". Elle divise le projet en petites fonctionnalités gérables, appelées "user stories", qui sont développées, testées et livrées à chaque sprint. Cette approche itérative permet une meilleure visibilité du progrès et offre la possibilité d'obtenir rapidement des retours, facilitant ainsi les ajustements nécessaires.

La méthodologie Scrum est largement adoptée dans le développement de logiciels, mais elle est également applicable à d'autres types de projets. Elle présente de nombreux avantages, tels que la flexibilité, l'adaptabilité, la transparence, une communication régulière avec les parties prenantes et une livraison régulière de produits fonctionnels de haute qualité.



Notre équipe est composée d'une Product owner (Celeste), un lead développeur (Mehdi) et moi-même.

3. Le déroulement d'un sprint

Toutes les deux semaines, nous organisons une session de backlog refinement pour définir les user stories à réaliser lors des prochains sprints. Chaque sprint dure deux semaines et commence par un sprint planning où nous assignons les tâches en fonction des compétences et des disponibilités de chaque membre de l'équipe. Chaque matin, nous tenons un daily meeting de 15 minutes pour partager nos progrès et signaler les éventuels blocages.

Tout au long du sprint, nous pratiquons le peer programming, une technique de travail en binôme qui améliore la qualité du code et favorise l'apprentissage. À la fin du sprint, nous procédons à une mise en production et organisons un sprint demo pour présenter le travail accompli à l'équipe. Nous concluons le sprint par une sprint rétrospective, une session qui nous permet de discuter du déroulement du sprint et d'identifier des actions à améliorer pour les futurs sprints.

Grâce à cette méthode agile, nous livrons des produits de qualité en respectant les délais et les attentes de nos clients, tout en renforçant la collaboration et la communication au sein de notre équipe.

4. Outils Logiciels

I. Plateforme de communication : Slack

Slack est une plateforme de communication en ligne qui permet aux équipes de collaborer, de discuter et de partager des fichiers en temps réel. C'est un outil de messagerie instantanée qui peut être utilisé sur ordinateur et sur mobile. Slack propose également des fonctionnalités avancées telles que la possibilité de créer des canaux de discussion, d'intégrer des applications tierces et de rechercher des messages. L'objectif principal de Slack est d'améliorer la communication et la collaboration au sein des équipes, en favorisant l'échange d'idées et la résolution de problèmes de manière plus rapide et efficace.

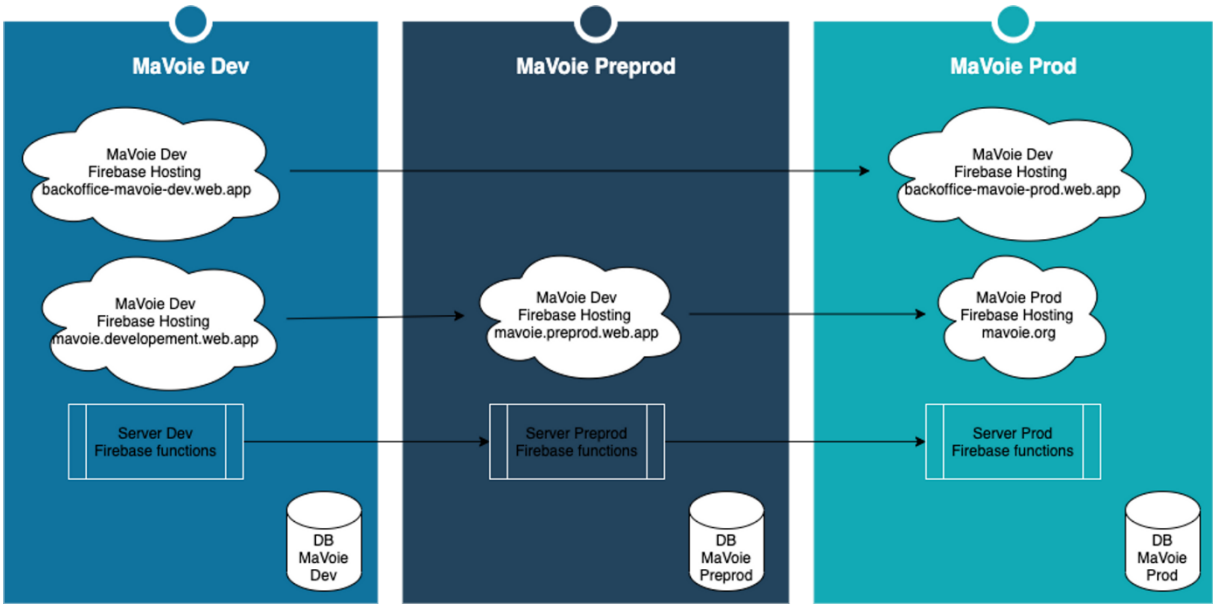
II. Gestion de projet Agile de répartition des tâches : Airtable

Airtable est un outil de gestion de bases de données en ligne qui permet aux utilisateurs de créer des tableaux et des bases de données personnalisées pour organiser et suivre des informations. Il propose des fonctionnalités de collaboration en temps réel, de visualisation de données, de filtrage et de tri, ainsi que des intégrations avec d'autres outils tels que Slack, Trello et Zapier. L'interface utilisateur intuitive et flexible d'Airtable le rend adapté à une variété de cas d'utilisation, des projets de développement aux projets créatifs et de marketing.

Stratégie de Déploiements

Introduction

L'objectif principal est d'assurer des livraisons continues en publiant du code soumis à un contrôle technique rigoureux. Cela inclut des tests unitaires, la vérification de la syntaxe et de la structure du code, ainsi que la gestion des dépendances. Ce processus est automatisé via des pipelines de déploiement. De plus, il est essentiel de respecter les bonnes pratiques de gestion de projet telles que la démonstration de sprint, la revue de code, et la validation des fonctionnalités par le Product Owner (PO) pour garantir un développement efficace et de qualité.



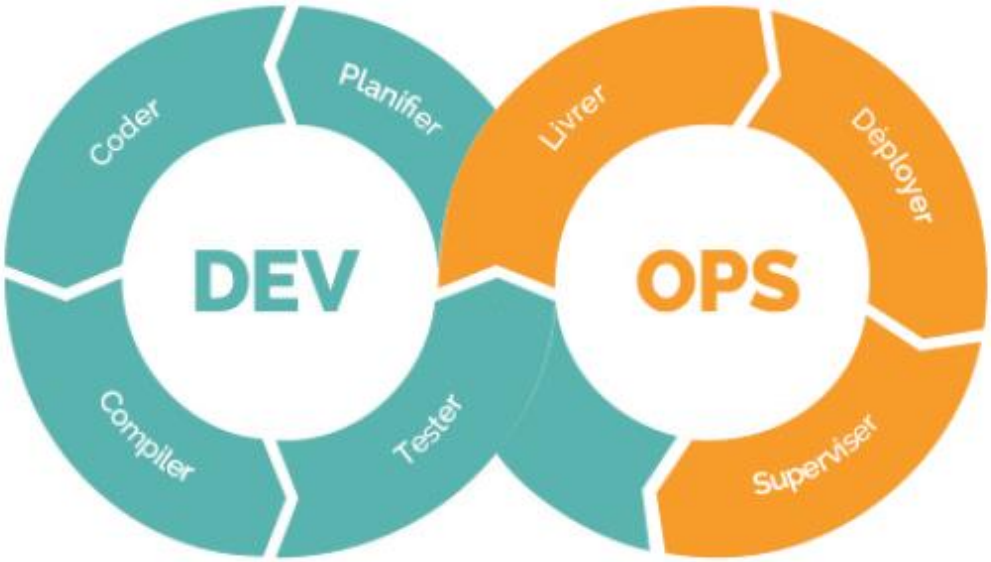
Les composants sous forme de nuage représentent le serveur statique d'hébergement du Front et le rectangle représente le serveur de Firebase Cloud functions où les APIs sont implémentés.

—> Il existe trois environnements associés à trois branches respectives dans github qui vont héberger la web app, le serveur du backend et les bases de données :

- **Develop** :
  - Url : https://mavoie-development.web.app/accueil
  - Objectif : Ce serveur héberge la version de test de l'application. Chaque fonctionnalité développée est d'abord déployée ici avant d'être mise en production. C'est sur ce serveur que le Product Owner (PO) et les autres membres de l'équipe effectuent des tests pour détecter d'éventuelles anomalies.
- **Preprod** :
  - Url : https://mavoie-preprod.web.app/accueil
  - Objectif : Ce serveur contient la version de test la plus stable de l'application, généralement destinée au PO dans les grandes équipes. Chaque fonctionnalité validée dans l'environnement de développement (develop) est déployée ici. Ce serveur imite la configuration de la production et constitue la dernière étape de test avant le déploiement en production. Il permet de valider plusieurs scénarios de test et de détecter d'éventuelles régressions.
- **Prod** :
  - Url : https://mavoie.org/accueil
  - Objectif : Il s'agit de la version live en production, destinée aux utilisateurs finaux du site.

1. Processus de livraison

L'approche DevOps est une méthodologie de développement logiciel visant à accélérer le cycle de mise sur le marché en automatisant les étapes de développement, de test et de déploiement. Fondée sur la collaboration entre les équipes de développement et d'exploitation, elle vise à améliorer la qualité et la stabilité des produits. Les pratiques DevOps englobent l'intégration continue, la livraison continue, le déploiement continu, la surveillance et l'automatisation des processus. L'objectif premier est de raccourcir le délai entre le développement d'une fonctionnalité et sa disponibilité pour les utilisateurs, tout en assurant une qualité optimale et une stabilité du système.



Le processus de livraison suit cette démarche : **Develop** ⇒ **Preprod** ⇒ **Prod**.

On crée un Pull Request de la branche **Develop** vers la branche **Preprod**, on résout les conflits et on attend la validation du pipeline. Ensuite, on valide le PR en fusionnant le code de **Develop** vers **Preprod**, et on répète la même opération entre **Preprod** et **Prod**.

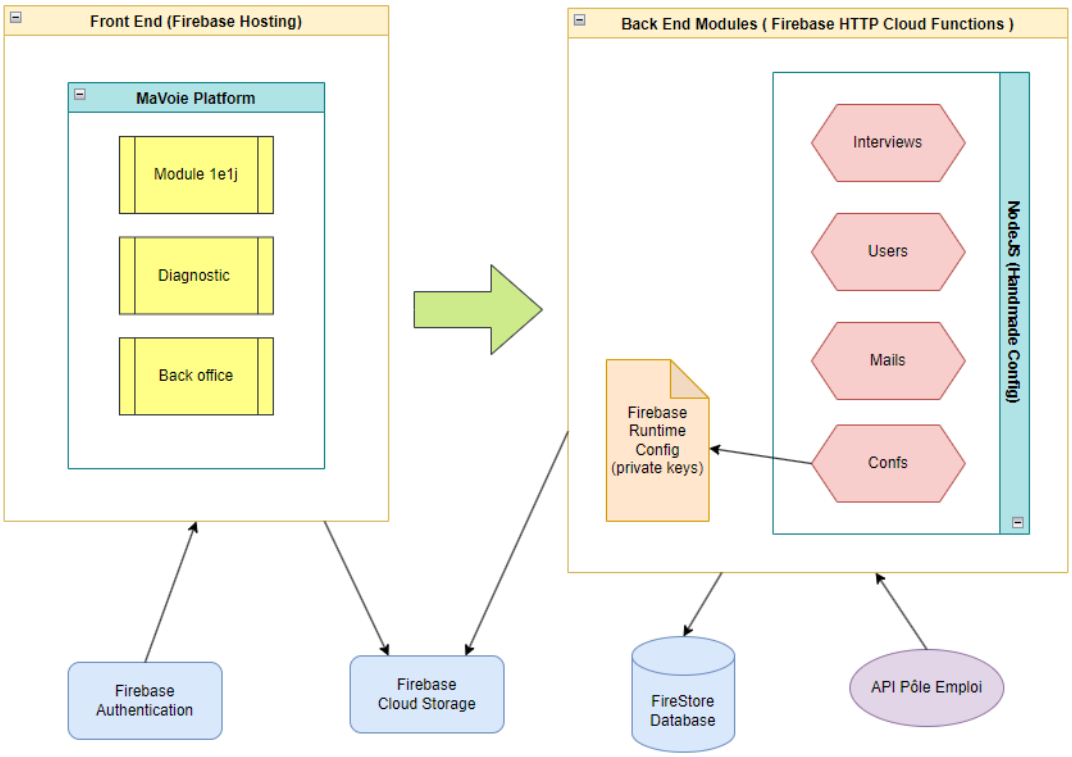
Généralement, lorsque nous voulons livrer une fonctionnalité, nous devons être synchronisés avec la dernière version de **Develop** (en exécutant git pull) et créer localement une branche en respectant la nomenclature suivante :

⇒ s{num-du-sprint}/{feat, fix, config, analytics, refactor}/{sujet}/{description-courte-de-la-tâche}

Avant de publier la branche sur Git, il est nécessaire de procéder à une vérification du code et de s'assurer qu'il compile correctement en local :

- Validation des tests unitaires : yarn test
- Validation de Prettier : yarn prettier:fix (format du code)
- Validation de ESLint : yarn lint:fix (règles de codage)

2. Architecture Logicielle



L'architecture de ce module repose sur un modèle à deux niveaux, comprenant un frontend modulaire pour l'interface utilisateur et un backend géré par Firebase Functions pour la gestion des opérations métier. La communication entre ces deux parties est standardisée, favorisant ainsi une grande modularité et une évolutivité accrue du système. De plus, l'application est facilement connectable à des services tiers tels que les APIs de Sendinblue, Calendly, Pole Emploi, un serveur de stockage de fichiers, un service d'authentification, un serveur SMTP pour l'envoi de courriels et une base de données. Cette architecture offre une meilleure sécurité, évolutivité et interopérabilité avec d'autres services.

3. Exigences techniques

Le module doit répondre aux exigences fonctionnelles ainsi qu'à plusieurs exigences techniques :

- Respect des bonnes pratiques en matière de codage.
- Développement de composants réutilisables pour assurer l'extensibilité de l'application à l'avenir.
- Mise en place d'un design responsive pour chaque page de l'application, afin de garantir une compatibilité avec les smartphones et les tablettes.
- Validation du format de code avec Prettier.
- Respect des règles de codage avec ESLint.
- Stockage de tous les textes de l'application dans un fichier JSON.
- Utilisation de l'outil de versioning Git, permettant la conservation de différentes versions du projet ainsi que le partage du code avec le lead développeur.

4. Outils Logiciels

I. Outil de versionning du code : Git

Git est un système de contrôle de version open source qui permet aux développeurs de suivre les changements de code source lorsqu'ils travaillent en collaboration sur un projet. Il a été créé par Linus Torvalds en 2005 et est largement utilisé pour le développement de logiciels. Git permet aux développeurs de travailler sur des versions différentes du code source en parallèle et de fusionner facilement les modifications apportées par différents membres de l'équipe. Il offre également une trace complète de toutes les modifications apportées au code source, ainsi que la possibilité de revenir en arrière à tout moment en cas de problème.

II. Plateforme de versionning du code : Github

GitHub est une plateforme en ligne qui permet aux développeurs de collaborer sur des projets de logiciels en utilisant Git, un système de contrôle de version. Les développeurs peuvent stocker et partager leur code source, suivre les problèmes, soumettre des demandes de tirage et examiner les modifications apportées au code. GitHub est devenu une plateforme essentielle pour de nombreux développeurs, car elle facilite la collaboration et le partage de code à grande échelle.

III. Outil de gestion de pipeline : Github Actions

GitHub Actions est un outil de CI/CD (intégration continue/déploiement continu) fourni par GitHub qui permet d'automatiser des tâches récurrentes dans le cadre d'un workflow de développement logiciel. Il permet d'écrire des scripts qui sont exécutés automatiquement en réponse à des événements spécifiques sur un référentiel GitHub, comme la création d'une nouvelle demande de fusion ou la modification d'un fichier. Les actions sont exécutées dans des conteneurs Docker, ce qui permet de créer un environnement de développement cohérent et portable. Les avantages de l'utilisation de GitHub Actions incluent une automatisation efficace des workflows de développement, une intégration transparente avec les référentiels GitHub existants et une grande flexibilité dans la configuration des workflows.

Perspective Personnelle – Mission 2

La Mission 2 m'a immergé dans l'univers captivant des outils de gestion de projet, m'offrant une exploration approfondie des plateformes et des méthodologies utilisées dans ce domaine crucial du développement d'application. Voici comment cette expérience a enrichi mes connaissances et contribué à ma croissance professionnelle :

- **Compréhension des Méthodologies Agile :** Travailler avec des méthodologies agiles telles que Scrum m'a permis de comprendre l'importance de la flexibilité, de l'itération et de la collaboration dans la gestion de projets logiciels. J'ai appris à adapter notre approche en fonction des besoins changeants du projet, ce qui a renforcé ma capacité à réagir efficacement aux défis rencontrés.
- **Maîtrise des Outils de Communication :** L'expérience pratique avec des outils tels que Slack et Airtable m'a donné une compréhension approfondie de leur rôle dans la facilitation de la communication et de la collaboration au sein de l'équipe. J'ai appris à utiliser ces plateformes de manière efficace pour suivre les tâches, partager des mises à jour et résoudre les problèmes rapidement.
- **Automatisation et DevOps :** L'intégration de GitHub Actions pour l'intégration continue et le déploiement continu m'a sensibilisé à l'importance de l'automatisation dans le processus de développement logiciel. J'ai appris à configurer et à gérer des workflows automatisés, ce qui a accéléré nos cycles de développement et amélioré la qualité de nos livraisons.
- **Croissance Personnelle et Professionnelle :** J'ai développé ma capacité à travailler efficacement en équipe, à résoudre des problèmes complexes et à gérer efficacement les projets dans un environnement dynamique

Voici quelques améliorations que j'ai pu imaginer :

1. **Personnalisation des Flux de Travail :** Adapter les flux de travail des outils de gestion de projet à nos besoins spécifiques pourrait améliorer notre efficacité. En personnalisant les champs, les étiquettes et les flux de travail, nous pourrions mieux suivre nos processus et optimiser nos pratiques de travail.
2. **Intégration avec d'Autres Outils :** Chercher des moyens d'intégrer nos outils de gestion de projet avec d'autres outils que nous utilisons fréquemment, tels que les outils de développement, de communication et de collaboration. Cette intégration pourrait réduire la duplication des tâches et améliorer la cohérence des données à travers différentes plateformes.
3. **Analyse et Amélioration Continue :** Mettre en place un processus d'analyse régulière de nos pratiques de gestion de projet pourrait nous aider à identifier des domaines à améliorer. En examinant nos performances passées et en recueillant les commentaires de l'équipe, nous pourrions itérer et affiner nos processus pour une efficacité accrue.
4. **Partage des Meilleures Pratiques :** Encourager le partage des meilleures pratiques au sein de l'équipe pourrait favoriser une adoption plus large et plus efficace des outils de gestion de projet. En identifiant et en diffusant les méthodes qui ont bien fonctionné pour certains membres de l'équipe, nous pourrions accélérer l'apprentissage et l'adoption des nouveaux outils.

Mission 3 – Stratégie de tests

Introduction

L'objectif de cette mission est d'explorer la stratégie de test mise en place dans notre association à travers la réalisation de notre projet informatique. Quel que soit le domaine d'intervention, que ce soit le développement ou les systèmes et réseaux, il est impératif d'effectuer un certain nombre de tests avant la mise en production de toute solution. Notre équipe de développement est composée de seulement deux développeurs, moi et mon tuteur. En raison de cette taille réduite, nous ne sommes pas toujours en mesure de mettre en œuvre toutes les pratiques de test souhaitées, principalement en raison du manque de temps et de ressources. Ce rapport détaillera la stratégie de tests en suivant les points clés mentionnés, tout en tenant compte des contraintes spécifiques liées à notre équipe.

1. Définition de la Stratégie de Tests

I. Objectifs des Tests

Les tests ont pour objectif principal d'assurer la qualité et la fiabilité du système avant sa mise en production. Ils permettent de :

- Vérifier que chaque fonctionnalité répond aux spécifications.
- Détecter et corriger les anomalies avant que les utilisateurs finaux ne les rencontrent.
- Garantir la performance, la sécurité et la robustesse du système.
- Assurer que les mises à jour ou les nouvelles fonctionnalités n'introduisent pas de régressions.

II. Critères de Qualité du Système à Tester

1. **Respect des Bonnes Pratiques en Matière de Codage :** Assurer que le code suit les standards de l'industrie pour maintenir la lisibilité, la maintenabilité et la performance.
2. **Développement de Composants Réutilisables :** Créer des composants qui peuvent être réutilisés pour faciliter l'extensibilité future de l'application.
3. **Design Responsive :** Garantir que chaque page de l'application est compatible avec les smartphones et les tablettes.
4. **Validation du Format de Code avec Prettier :** Utiliser Prettier pour s'assurer que le formatage du code est cohérent.
5. **Respect des Règles de Codage avec ESLint :** Utiliser ESLint pour vérifier que le code suit les conventions et éviter les erreurs potentielles.
6. **Stockage des Textes dans un Fichier JSON :** Centraliser tous les textes de l'application dans un fichier JSON pour faciliter la gestion et la localisation.
7. **Utilisation de Git pour le Versioning :** Utiliser Git pour conserver différentes versions du projet et faciliter le partage du code avec le lead développeur.

III. Moyens mis à la disposition des équipes de tests

En tant que petite équipe composée uniquement de deux membres, notre approche pour la mise en place de la stratégie de tests repose sur une collaboration étroite et une utilisation efficace des ressources techniques disponibles. Nous n'avons pas d'équipe de tests dédiée, ce qui signifie que nous sommes responsables de la planification, de l'exécution et de l'analyse des tests pour garantir la qualité de notre solution informatique. Malgré nos ressources limitées, nous avons adopté plusieurs pratiques pour optimiser nos capacités de test.

- **Dispositif humain : Collaboration étroite**

Notre taille réduite favorise une collaboration étroite entre les membres de l'équipe, ce qui nous permet de partager facilement nos connaissances, de discuter des problèmes rencontrés et de prendre des décisions rapidement. Cette approche favorise une meilleure communication et une compréhension plus approfondie des besoins du projet, ce qui améliore la qualité de nos tests.

- **Dispositif technique : Automatisation des tests dans les pipelines**

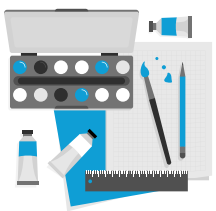
Pour maximiser notre efficacité, nous avons intégré l'automatisation des tests dans nos pipelines de développement. Cette automatisation nous permet d'exécuter rapidement et de manière cohérente un ensemble de tests prédéfinis à chaque modification du code source. En automatisant les tests, nous réduisons les erreurs humaines et accélérons le processus de validation, ce qui nous permet de livrer des fonctionnalités de manière plus rapide et fiable.

- **Propositions pour l'amélioration continue**

Malgré nos efforts actuels, nous reconnaissons qu'il y a toujours place à l'amélioration. Pour renforcer nos capacités de test, nous envisageons d'explorer l'utilisation de modèles de test pour standardiser nos processus et améliorer la reproductibilité des tests. De plus, nous nous engageons à investir dans une formation continue pour rester à jour avec les meilleures pratiques et les nouvelles technologies dans le domaine des tests logiciels.

IV. Modes d'archivage de tests

Dans notre processus de test, nous accordons une grande importance à la documentation et à l'archivage pour garantir la traçabilité et faciliter la référence future. Pour cela, nous utilisons activement **Notions.so** pour centraliser nos informations de test. Chaque test est soigneusement documenté, enregistrant les scénarios, les résultats et les observations. Une fois terminés, les tests sont archivés pour référence future. Cette approche garantit une documentation complète et accessible, favorisant la transparence et l'amélioration continue.



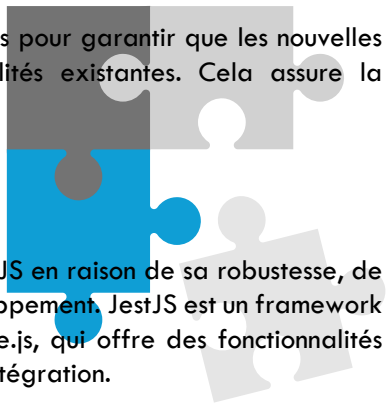
2. Définition d'un plan de tests

I. Planification des tests

Dans notre approche de planification des tests, nous reconnaissons l'importance de maximiser l'efficacité tout en garantissant une couverture adéquate. Compte tenu des ressources limitées, nous adoptons une approche stratégique en priorisant les tests sur les fonctionnalités critiques et les scénarios d'utilisation les plus courants. Cela nous permet de concentrer nos efforts sur les domaines les plus sensibles et les plus impactantes, assurant ainsi la stabilité et la qualité du système dans son ensemble.

Pour affiner notre planification des tests, nous proposons les mesures suivantes :

1. **Identification des fonctionnalités critiques :** Déterminer les fonctionnalités qui ont un impact direct sur la satisfaction de l'utilisateur ou sur la performance globale du système. Ces fonctionnalités seront testées en priorité.
2. **Analyse des risques :** Évaluer les risques associés à chaque fonctionnalité et orienter les tests en fonction de ces évaluations. Les fonctionnalités à haut risque seront testées en priorité pour minimiser les impacts négatifs.
3. **Tests exploratoires :** Utiliser des sessions de tests exploratoires pour découvrir des défauts imprévus et évaluer la qualité globale du système. Cette approche permet une flexibilité dans l'exploration des fonctionnalités et des scénarios d'utilisation.
4. **Tests automatisés :** Automatiser les tests pour les fonctionnalités récurrentes ou critiques, ce qui permet de gagner du temps et d'assurer une couverture régulière des tests même avec des ressources limitées.
5. **Tests de régression :** Planifier des tests de régression réguliers pour garantir que les nouvelles fonctionnalités n'impactent pas négativement les fonctionnalités existantes. Cela assure la stabilité du système à long terme.



II. Mise en place des outils de test

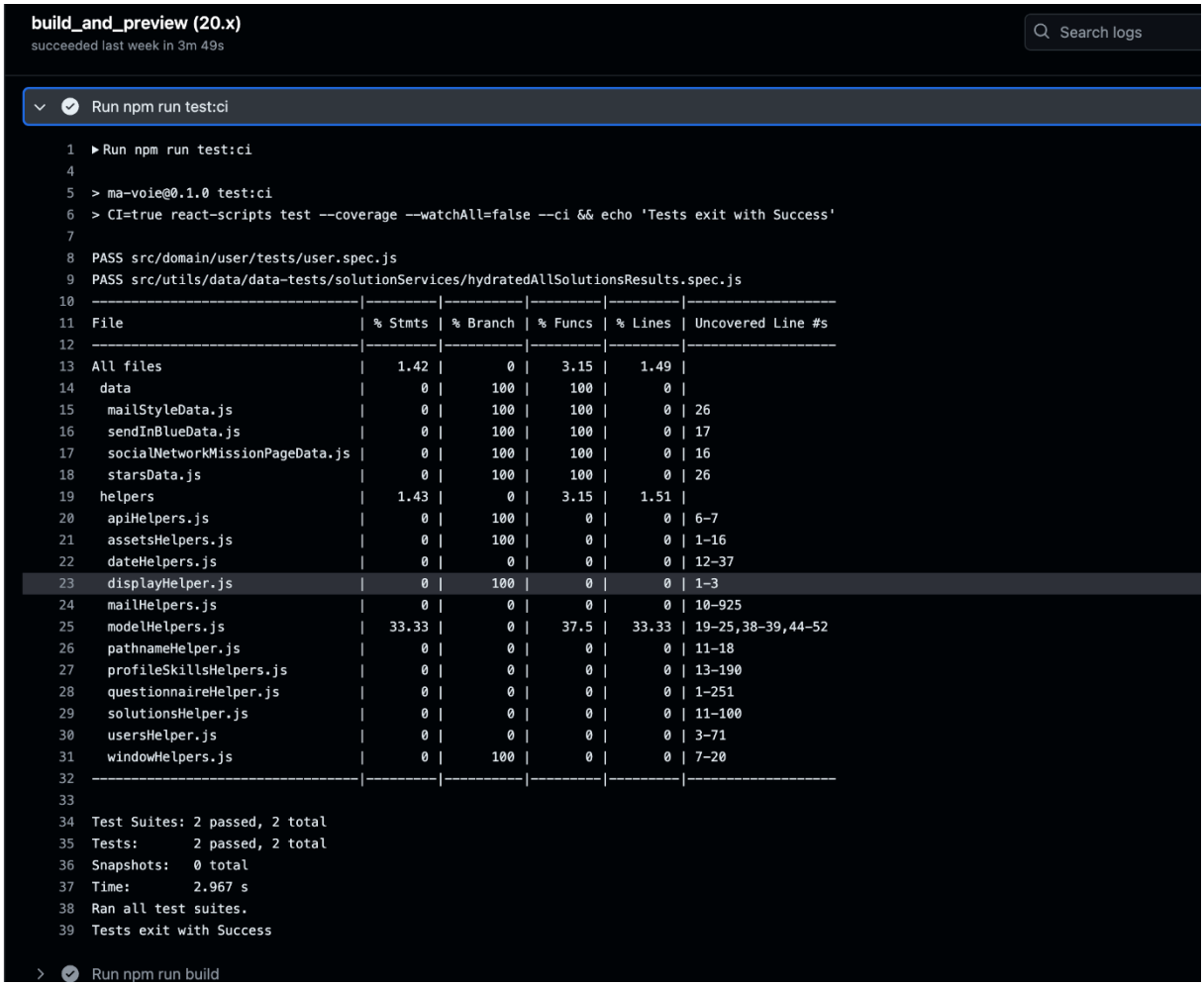
Pour la mise en place des outils de tests, nous avons choisi d'utiliser JestJS en raison de sa robustesse, de sa simplicité et de sa compatibilité avec notre environnement de développement. JestJS est un framework de test JavaScript moderne, largement utilisé dans l'écosystème Node.js, qui offre des fonctionnalités puissantes pour l'écriture et l'exécution de tests unitaires et de tests d'intégration.

En intégrant JestJS dans nos pipelines de développement, nous maximisons l'efficacité de nos processus de test en automatisant la détection et la correction des erreurs dès que du code est poussé dans le référentiel de versionnement. Cela garantit une rétroaction rapide aux développeurs, ce qui favorise une boucle de rétroaction rapide et une amélioration continue de la qualité du code.

Pour améliorer davantage notre processus de test, nous envisageons les propositions suivantes :

1. **Intégration de l'intégration continue (CI) :** Utiliser des outils CI tels que Jenkins, Travis CI ou GitHub Actions pour automatiser l'exécution de nos suites de tests à chaque modification du code source. Cela garantit que toutes les modifications sont testées de manière cohérente et fiable.
2. **Utilisation de Docker :** Docker peut être utilisé pour créer des environnements de test isolés et reproductibles, ce qui facilite le déploiement et l'exécution de nos tests dans différents environnements.
3. **Monitoring des performances :** Intégrer des outils de surveillance des performances tels que New Relic ou Prometheus pour évaluer l'impact des modifications de code sur les performances du système et détecter les goulots d'étranglement potentiels.
4. **Tests de bout en bout (E2E) :** Envisager l'utilisation de frameworks de tests E2E tels que Cypress ou Selenium pour automatiser les tests qui simulent le comportement de l'utilisateur final dans le navigateur.

Aperçu de nos tests déjà mise en place :



build_and_preview (20.x)
succeeded last week in 3m 49s

Search logs

Run npm run test:ci

```
▶ Run npm run test:ci

> ma-voie@0.1.0 test:ci
> CI=true react-scripts test --coverage --watchAll=false --ci && echo 'Tests exit with Success'

PASS src/domain/user/tests/user.spec.js
PASS src/utils/data/data-tests/solutionServices/hydratedAllSolutionsResults.spec.js
-----------------------------------|---------|----------|---------|---------|-------------------
File                               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-----------------------------------|---------|----------|---------|---------|-------------------
All files                          |    1.42 |        0 |    3.15 |    1.49 |
 data                              |       0 |      100 |     100 |       0 |
  mailStyleData.js                 |       0 |      100 |     100 |       0 | 26
  sendInBlueData.js                |       0 |      100 |     100 |       0 | 17
  socialNetworkMissionPageData.js  |       0 |      100 |     100 |       0 | 16
  starsData.js                     |       0 |      100 |     100 |       0 | 26
 helpers                           |    1.43 |        0 |    3.15 |    1.51 |
  apiHelpers.js                    |       0 |      100 |       0 |       0 | 6-7
  assetsHelpers.js                 |       0 |      100 |       0 |       0 | 1-16
  dateHelpers.js                   |       0 |        0 |       0 |       0 | 12-37
  displayHelper.js                 |       0 |      100 |       0 |       0 | 1-3
  mailHelpers.js                   |       0 |        0 |       0 |       0 | 10-925
  modelHelpers.js                  |   33.33 |        0 |    37.5 |   33.33 | 19-25,38-39,44-52
  pathnameHelper.js                |       0 |        0 |       0 |       0 | 11-18
  profileSkillsHelpers.js          |       0 |        0 |       0 |       0 | 13-190
  questionnaireHelper.js           |       0 |        0 |       0 |       0 | 1-251
  solutionsHelper.js               |       0 |        0 |       0 |       0 | 11-100
  usersHelper.js                   |       0 |        0 |       0 |       0 | 3-71
  windowHelpers.js                 |       0 |      100 |       0 |       0 | 7-20
-----------------------------------|---------|----------|---------|---------|-------------------

Test Suites: 2 passed, 2 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        2.967 s
Ran all test suites.
Tests exit with Success
```

Run npm run build

III. Rédaction des procédures de tests

Pour rédiger des procédures de tests détaillées, voici quelques propositions que nous pourrions envisager :

1. **Identification des scénarios de test :** Commencer par identifier les scénarios de test pertinents pour chaque fonctionnalité ou composant à tester. Cela peut inclure des cas de test positifs (pour vérifier que le système fonctionne comme prévu) et des cas de test négatifs (pour vérifier le comportement du système en cas d'erreurs ou de situations inattendues).
2. **Description des étapes de test :** Pour chaque scénario de test identifié, décrire les étapes spécifiques à suivre pour exécuter le test avec succès. Cela peut inclure des actions à effectuer dans l'interface utilisateur, des commandes à exécuter dans le terminal, ou des requêtes à envoyer à l'API, selon le type de test.
3. **Préparation de l'environnement de test :** Définir les prérequis nécessaires pour exécuter chaque test, tels que la configuration de l'environnement, la création de données de test, ou le déploiement de la version de code à tester.
4. **Validation des résultats attendus :** Cela peut inclure la vérification des valeurs retournées, des messages d'erreur affichés, ou des changements dans l'état de l'application.
5. **Documentation des résultats :** Une fois le test exécuté, documenter les résultats obtenus, y compris les éventuelles erreurs rencontrées, les problèmes identifiés, et les actions correctives prises le cas échéant.
6. **Révision et mise à jour régulières :** Cela garantit que les tests restent pertinents et efficaces tout au long du cycle de vie du projet.

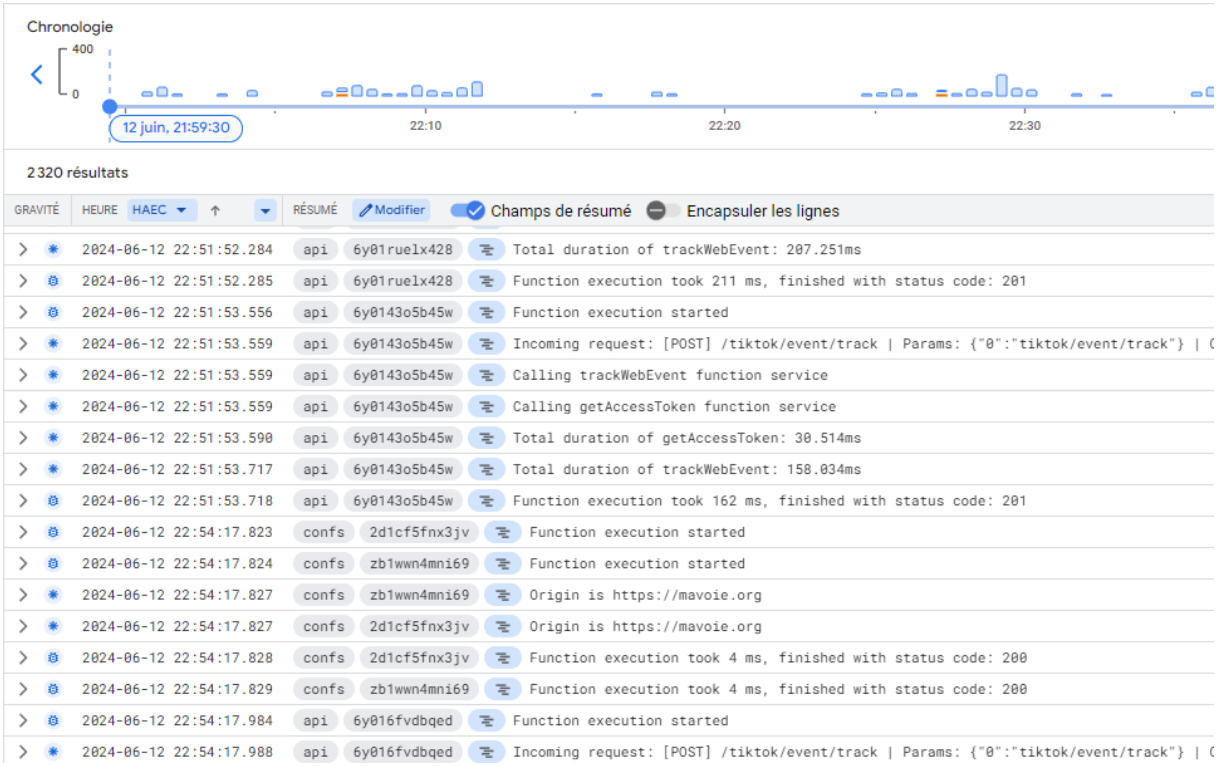
3. Réalisation des tests

I. Rapports de validation

Pour les rapports de validations, on souhaitera améliorer les points suivants :

- **Développement de meilleurs logs :** L'amélioration des logs peut être considérée comme une initiative transversale qui affecte plusieurs parties de la stratégie de tests. Tout d'abord, cela peut être considéré comme une amélioration des critères de qualité du système à tester, en veillant à ce que les logs soient informatifs, précis et complets. Ensuite, cela peut également faire partie des procédures de tests, en ajoutant des tests spécifiques pour vérifier la qualité des logs générés par l'application. Enfin, l'utilisation accrue des logs peut également être intégrée dans les rapports de validation, en mettant en évidence les informations importantes extraites des logs lors de l'évaluation de la qualité du système.
- **Consultation accrue des logs :** Cela peut être considéré comme une pratique recommandée dans la phase de réalisation des tests, notamment lors de l'exécution de tests exploratoires ou de tests de non-régression. Les membres de l'équipe de développement peuvent consulter les logs pour comprendre le comportement de l'application, identifier les problèmes potentiels et vérifier si les fonctionnalités répondent aux spécifications. Cette pratique peut être intégrée dans les procédures de tests et les réunions de revue des tests pour garantir une utilisation efficace des logs tout au long du processus de test.
- **Journalisation via un outil de monitoring :** Cela peut être considéré comme une pratique recommandée pour assurer la détection rapide des anomalies et la collecte d'informations pour l'analyse post-mortem en cas d'incidents. Cette initiative peut être intégrée dans la phase de réalisation des tests en mettant en place des mécanismes pour capturer et analyser les logs générés par l'outil de monitoring, ainsi que dans la phase de gestion des anomalies pour identifier et résoudre rapidement les problèmes détectés.

Aperçu de notre Google Cloud Platform :



II. Gestion des anomalies

Pour assurer une gestion efficace des anomalies détectées lors des tests, nous avons mis en place un processus structuré :

1. **Création de tickets dans le système de suivi des problèmes :** Chaque anomalie détectée est enregistrée sous la forme d'un ticket dans notre système de suivi des problèmes. Ces tickets contiennent des informations détaillées sur l'anomalie, y compris sa description, les étapes de reproduction, et les résultats attendus.
2. **Estimation de la priorité :** Une fois qu'un ticket est créé, il est évalué pour déterminer sa priorité. Nous attribuons une priorité en fonction de l'impact de l'anomalie sur le système, sa criticité et son urgence.
3. **Embarquement immédiat pour les anomalies importantes :** Les anomalies jugées importantes ou critiques sont traitées immédiatement, sans attendre. Elles sont assignées aux membres de l'équipe appropriés pour une résolution rapide.
4. **Suivi régulier des tickets :** Nous effectuons un suivi régulier des tickets pour surveiller leur statut et leur progression. Cela garantit que les problèmes ne sont pas oubliés et que des mesures appropriées sont prises pour leur résolution.
5. **Collaboration étroite pour la résolution des anomalies :** L'équipe collabore étroitement pour résoudre les anomalies, en partageant des informations, des idées et des solutions potentielles. Cette collaboration favorise une résolution rapide et efficace des problèmes.
6. **Validation et clôture des tickets :** Une fois qu'une anomalie est résolue, elle est soumise à une validation pour confirmer que le problème a été correctement résolu. Une fois validée, le ticket est clos et les parties prenantes sont informées de sa résolution.

III. Temps consacré aux tests / durée du projet

Répartition du Temps entre les Phases du Projet

Actuellement, nous ne consacrons pas de temps spécifique aux tests formels dans notre cycle de développement. Cependant, voici une proposition de répartition du temps qui pourrait améliorer notre processus de test et assurer une meilleure qualité du produit final :

Phase du Projet	**Pourcentage du Temps Total**	**Détails**
Planification	10%	Définition des objectifs, spécifications, et planification des ressources.
Développement	50%	Codage, revue de code, et intégration continue.
Tests	20%	Tests unitaires, tests d'intégration, et tests de régression.
Correction des Anomalies	10%	Identification, documentation, et résolution des anomalies.
Documentation et Validation	5%	Documentation des résultats, validation des fonctionnalités, et création des rapports de validation.
Déploiement et Maintenance	5%	Déploiement du produit final et maintenance post-déploiement.

Propositions pour Améliorer la Répartition du Temps

1. **Introduction des Tests Automatisés :** Utiliser des outils comme JestJS et les intégrer davantage dans les pipelines CI/CD pour automatiser les tests unitaires et les tests de régression.
2. **Définir des Phases de Tests Spécifiques :** Allouer des périodes spécifiques pour les tests dans le cycle de développement, par exemple, des sprints dédiés aux tests et à la validation des fonctionnalités.
3. **Renforcer la Collaboration :** Encourager la collaboration entre les développeurs et les parties prenantes pour identifier et résoudre les anomalies plus rapidement.
4. **Formation Continue :** Participer à des formations sur les meilleures pratiques de test et sur l'utilisation des outils de test pour améliorer les compétences de l'équipe.
5. **Utilisation de Modèles de Test :** Implémenter des modèles de test standardisés pour assurer une couverture de test complète et cohérente.

IV. Estimation du coût de la phase de test / durée du projet

Dans notre méthodologie actuelle, nous estimons le coût de la phase de test de la même manière que nous estimons les autres tâches du projet, en utilisant la suite de Fibonacci (1, 3, 5, 7). Cela permet une évaluation rapide et flexible du temps et des ressources nécessaires pour les tests en fonction de la complexité perçue de chaque tâche.

Méthode d'Estimation

- 1 Point : Tests très simples, peu de cas à vérifier, généralement pour des modifications mineures.
- 3 Points : Tests de complexité moyenne, avec plusieurs cas de test et interactions à vérifier.
- 5 Points : Tests complexes, incluant de multiples scénarios et dépendances.
- 7 Points : Tests très complexes, nécessitant une analyse approfondie, plusieurs itérations de test, et une coordination importante.

Exemple d'Estimation du Coût de la Phase de Test

Voici un exemple d'estimation de coût pour un projet fictif :

Tâche	Points Fibonacci	Heures Estimées (1 point = 2 heures)	Coût Estimé (Heures * Taux Horaire)
Tests Unités - Fonction A	1	2	(2 heures * 50 €) = 100 €
Tests d'Intégration - Fonction B	3	6	(6 heures * 50 €) = 300 €
Tests de Régression - Module C	5	10	(10 heures * 50 €) = 500 €
Tests de Performance - Système D	7	14	(14 heures * 50 €) = 700 €
Total Phase de Test	16	32	1600 €

V. Tests de non-régression

Voici un template de test fonctionnel de non-régression que nous utilisons et dupliquons pour différentes fonctionnalités :

👇👇 *Voici un template de test à dupliqué* 👇👇

Checklist de Tests Fonctionnels de Non-Régression

Page d'Accueil (Homepage)

Vérification de la fonctionnalité "Laisser le mail en newsletter"

- ☐ Le champ d'e-mail est visible.
- ☐ Le bouton d'inscription à la newsletter est fonctionnel.
- ☐ La confirmation visuelle de l'inscription est affichée.
- ☐ Les données sont correctement transmises au serveur (Backend).
- ☐ Vérification des logs du serveur pour s'assurer que l'e-mail est enregistré.
- ☐ Les données sont correctement stockées dans la base de données Firestore.

Pages de Questionnaires

Saisie de tous les formulaires dans Redux

- ☐ Les données saisies sont correctement gérées dans l'état Redux.
- ☐ Les actions Redux sont correctement déclenchées lors de la saisie.
- ☐ Les données sont correctement transmises au serveur (Firestore).
- ☐ Vérification des logs du serveur pour s'assurer que les données sont enregistrées.
- ☐ Les données sont correctement stockées dans la base de données Firestore.

Formulaire d'Email

- ☐ Les données de l'e-mail sont correctement transmises au serveur (Firestore).
- ☐ Vérification des logs du serveur pour s'assurer que l'e-mail est enregistré.

4. Tests et qualité

Actuellement, nous n'avons pas de processus formalisé pour aligner nos tests avec un plan qualité ou des normes de qualité spécifiques. Cependant, intégrer les tests à un plan qualité peut grandement améliorer la fiabilité et la robustesse de notre logiciel. Voici quelques propositions pour y parvenir :

Propositions pour l'Intégration des Tests à un Plan Qualité

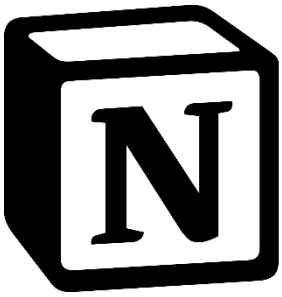
1. Adoption de Normes de Qualité Internationales
   - ISO 9001 : Intégrer les tests dans un cadre de gestion de la qualité basé sur la norme ISO 9001 pour assurer que tous les processus, y compris les tests, respectent des critères de qualité reconnus internationalement.
   - ISO/IEC 25010 : Utiliser cette norme pour définir et évaluer la qualité des systèmes et des logiciels, en se concentrant sur des aspects comme la fiabilité, l'efficacité et la sécurité.
2. Mise en Place d'un Plan Qualité Global
   - Définition d'un Plan Qualité : Créer un document détaillant les objectifs de qualité, les critères de succès et les processus de validation, y compris les tests.
   - Alignement des Tests : Faire en sorte que les tests de validation, de non-régression, et de performance soient alignés sur les objectifs et les exigences définis dans le plan qualité.
3. Documentation et Processus Formels
   - Procédures Documentées : Établir des procédures documentées pour chaque type de test, y compris des étapes claires et des critères d'acceptation.
   - Archivage Structuré : Utiliser des outils de gestion documentaire comme Notion.so pour centraliser et structurer la documentation des tests, facilitant ainsi l'audit et la révision.
4. Revue et Audits de Qualité
   - Revue Périodique des Tests : Organiser des revues régulières des processus de test pour s'assurer qu'ils sont toujours alignés sur le plan qualité.
   - Audits Internes : Effectuer des audits internes pour vérifier la conformité des tests aux normes de qualité et identifier les domaines d'amélioration.
5. Formation et Sensibilisation
   - Formation Continue : Former les membres de l'équipe aux normes de qualité et aux meilleures pratiques de test pour renforcer leur compétence et leur sensibilisation à l'importance de la qualité.
   - Culture de la Qualité : Promouvoir une culture de la qualité au sein de l'équipe en soulignant l'importance des tests et de leur alignement avec les normes de qualité.

5. Outils Utilisés

I. Framework de tests : JestJS

Jest est un framework de test JavaScript, développé par Facebook, et conçu pour tester Babel, TypeScript, Node, React, Angular, Vue et plus encore. Il permet d'écrire des tests unitaires, des tests d'intégration et des tests de bout en bout. Jest fournit des fonctionnalités telles que la gestion des mocks, des spies, des assertions, des snapshots, et des tests en parallèle pour améliorer l'efficacité des tests. Il est également livré avec une interface en ligne de commande conviviale qui facilite l'exécution et la configuration des tests.

II. Plateforme Collaborative : Notion.so



Notion.so est une plateforme collaborative tout-en-un permettant de gérer des projets, de prendre des notes, et de suivre des tâches. Offrant une interface flexible et personnalisable, Notion intègre des fonctionnalités de bases de données, de wikis et de calendriers. Utilisée par les équipes pour organiser leur travail et améliorer la productivité, Notion permet également une intégration fluide avec divers outils externes. Idéale pour le travail en équipe, elle favorise la collaboration en temps réel et la centralisation des informations.

Perspectives Personnelles – Mission 3

Voici de quelle manière cette expérience a élargi mes connaissances et favorisé ma croissance professionnelle :

- **Apprentissage de la Stratégie de Tests :** J'ai appris l'importance de planifier soigneusement les tests, en identifiant les scénarios critiques et en définissant des cas de test pertinents pour garantir la qualité d'application.
- **Maîtrise des Outils de Test :** De la mise en place de tests automatisés à l'exécution de tests manuels, j'ai pu explorer différentes méthodes et comprendre comment choisir les outils les plus adaptés à chaque situation.
- **Collaboration et Communication :** La mission a souligné l'importance de la collaboration étroite avec les développeurs et les parties prenantes pour garantir une couverture de test complète. Communiquer efficacement sur les résultats des tests et les problèmes identifiés est essentiel pour garantir une résolution rapide et efficace des défauts.
- **Gestion des Risques :** J'ai appris à évaluer les risques associés à chaque fonctionnalité et à prioriser les tests en conséquence. Cette approche m'a permis de concentrer nos efforts là où ils sont le plus nécessaires, maximisant ainsi l'impact de nos tests sur la qualité globale du produit.

Voici quelques points d'amélioration :

- **Expansion de la Couverture des Tests :** Nous pourrions envisager d'élargir la couverture de nos tests pour inclure des cas d'utilisation plus diversifiés et des scénarios de test plus complexes. Cela garantirait une validation plus approfondie de notre application et réduirait les risques d'anomalies dans des cas non couverts.
- **Renforcement de l'Automatisation des Tests** Cela permettrait d'accélérer notre processus de test et de libérer du temps pour d'autres activités de développement.
- **Intégration de la Sécurité dans les Tests :** En tant qu'association, la sécurité des données et des systèmes est primordiale. Nous pourrions intégrer des tests de sécurité dans notre stratégie de tests pour détecter et corriger les vulnérabilités potentielles dès les premiers stades du développement.
- **Optimisation des Ressources de Test :** Avec une équipe de taille réduite, il est crucial d'optimiser l'utilisation de nos ressources de test. Nous pourrions examiner nos processus actuels et identifier les inefficacités ou les redondances pour les éliminer et améliorer ainsi notre productivité globale.
- **Analyse des Données de Test :** Nous pourrions mettre en place des mécanismes pour analyser les données générées par nos tests, afin d'identifier les tendances, les patterns d'échec récurrents et les domaines à problèmes. Cette analyse nous permettrait de prendre des décisions éclairées pour améliorer nos tests et notre application.

La nouvelle vision – Le futur de MaVoie.org

Dans le cadre du projet initial, notre objectif était de concevoir des solutions pour aider les jeunes éloignés de l'emploi à trouver des opportunités professionnelles. Cependant, nous avons pris conscience de la nécessité de revoir notre approche pour mieux répondre aux besoins de cette population. Ainsi, nous avons décidé de changer de vision, mettant l'accent sur la création d'outils alimentés par l'IA pour permettre aux jeunes de mieux se connaître, de s'entraîner en entretien, de développer leurs compétences et soft skills, le tout accompagné de la création d'espaces utilisateurs dédiés.

Pour la transition vers une nouvelle vision axée sur l'aide aux jeunes éloignés de l'emploi, nos trois missions nous ont permis de poser des bases solides pour la création d'outils novateurs visant à les accompagner dans leur parcours professionnel.

La Mission 1, consacrée à la découverte des besoins des jeunes en matière d'emploi, a mis en lumière le besoin crucial pour ces individus de mieux se connaître, d'identifier leurs forces et leurs faiblesses, ainsi que leurs intérêts professionnels. Sur cette base, nous envisageons de développer des outils interactifs et personnalisés, exploitant les capacités de l'IA pour fournir des évaluations approfondies et des recommandations pertinentes.

La Mission 2, qui portait sur l'exploration des outils de gestion de projet, nous a permis de comprendre l'importance de la planification, de la collaboration et de la flexibilité dans la réalisation de notre vision. En utilisant ces outils, nous pouvons organiser notre travail de manière plus efficace et garantir une progression cohérente vers nos objectifs.

Enfin, **la Mission 3**, centrée sur la stratégie de tests, a souligné l'importance de la qualité et de la fiabilité de nos futurs outils. En adoptant une approche rigoureuse des tests, nous nous assurons que nos solutions informatiques répondent aux normes les plus élevées et offrent une expérience utilisateur optimale.

Dans le contexte de notre nouvelle vision, cela se traduit par la conception d'une plateforme où les jeunes pourront suivre leur progression, définir des objectifs professionnels et accéder à des ressources pour les aider à développer les compétences nécessaires à leur insertion professionnelle.

Conclusion

Ces trois missions ont été une véritable aventure d'apprentissage et de croissance pour moi. En explorant les outils de gestion de projet, j'ai réalisé l'importance de la flexibilité, de la collaboration et de l'automatisation dans la gestion efficace des tâches. La méthodologie agile, en particulier, m'a permis de comprendre comment les processus itératifs et la communication transparente peuvent conduire à des résultats plus rapides et plus satisfaisants.

En ce qui concerne la stratégie de tests, j'ai approfondi ma compréhension de l'importance cruciale des tests dans le développement logiciel. Les tests permettent non seulement de détecter les erreurs, mais aussi de garantir la qualité, la fiabilité et la sécurité des produits. De plus, j'ai appris que l'automatisation des tests et l'intégration continue sont des pratiques essentielles pour accélérer les cycles de développement et améliorer la robustesse des applications.

Quant au changement de vision vers la création d'outils alimentés par l'IA, cette transition a été une étape importante dans notre parcours. Cela a renforcé ma conviction en la puissance de la technologie pour transformer positivement la vie des individus.

Dans l'ensemble, ces missions m'ont permis de développer non seulement mes compétences techniques, mais aussi ma capacité à m'adapter au changement, à collaborer efficacement en équipe et à voir au-delà des défis pour saisir les opportunités. Je suis reconnaissant pour cette expérience enrichissante et je suis impatient de continuer à contribuer à des projets qui ont un impact positif sur la vie des autres.