Le cahier des charges

Application SaaS (de gestion d’entreprise)

**1. Contexte et Objectifs**

**Contexte :**

L'objectif est d'offrir une solution complète et intuitive professionnels, leur permettant de simplifier la gestion des devis, factures, et la communication client. En centralisant ces tâches, l'application permet de gagner du temps et de mieux suivre la performance de l'entreprise. La solution doit non seulement être fonctionnelle, mais aussi ergonomique et fiable, garantissant une expérience utilisateur fluide.

**Objectifs :**

* **Créer, gérer, et envoyer des devis et des factures** : Intégrer une fonctionnalité de création rapide, avec des modèles personnalisables, facilitant le calcul des prix et des taxes, et l'exportation en formats courants (PDF, Excel).
* **Faciliter la communication** : Intégrer une messagerie centralisée, avec notifications et suivi des interactions.
* **Tableau de bord** : Suivi en temps réel des métriques clés (revenus, interactions clients, etc.), avec des options de filtrage avancées pour affiner les rapports.
* **Accessibilité, intuitivité, et sécurité** : Interface claire et agréable, accessible sur toutes les plateformes avec des normes de sécurité strictes (authentification, chiffrement des données sensibles).

### 2. ****Public Cible****

La solution est ciblée principalement vers des PME et professionnels libéraux, tout secteur confondu. Elle pourrait s'adresser également à des secteurs plus spécifiques comme les startups, les Freelancer en design, marketing, etc., offrant un environnement flexible.

**3. Fonctionnalités Requises**

**a. Gestion des Devis :**

* Ajout de champs personnalisés pour plus de flexibilité (par exemple, mode de paiement, conditions spéciales).
* Option pour sauvegarder les devis en tant que modèles réutilisables.

**b. Gestion des Factures :**

* Intégration d'un suivi des paiements partiels avec génération automatique de rappels pour relancer les clients en cas de retard.

**c. Communication :**

* Intégration d'une fonctionnalité de suivi de statut des messages (lu/non lu).
* Historique des messages de chaque client pour un suivi facilité.

**d. Tableau de Bord :**

* Filtres avancés pour les statistiques : par période, par client, par produit/service.
* Alerte sur les objectifs de performance (par exemple, seuil de revenu mensuel).

**e. Sécurité :**

* Ajouter des tests de pénétration réguliers pour assurer une sécurité maximale.
* Utilisation de protocoles de sécurité avancés, comme le HSTS et la protection contre les attaques de type injection SQL.

**4. Contraintes Techniques**

**a. Frontend :**

* L'implémentation de React.js avec Tailwind CSS permet de garantir une interface responsive et moderne.
* Prévoir un support pour les PWA (Progressive Web App), afin que l'application soit utilisée aussi bien en mode hors ligne qu'en ligne.

**b. Backend :**

* L'utilisation de Spring Boot et de JWT pour l'authentification est un bon choix, surtout pour garantir des performances élevées et une gestion simple des sessions utilisateurs.
* Ajouter des mécanismes de gestion de la charge, comme des files d'attente pour les tâches lourdes (envois massifs de factures, génération de rapports).

**c. Hébergement :**

* Choisir des solutions cloud pour l'évolutivité. Par exemple, AWS Lambda pour certaines fonctionnalités serveurless ou Google Cloud Functions pour les tâches ponctuelles et autonomes.

**5. Design et Expérience Utilisateur (UX)**

**Landing Page**

* **Couleurs et Interface** : Prévoir des couleurs neutres (gris clair, bleu, vert) pour ne pas surcharger visuellement l'utilisateur, tout en mettant en avant des éléments interactifs par des nuances plus vives.
* **Mode sombre** : Une option permettant de basculer entre le mode clair et sombre améliore l'expérience, en particulier pour les utilisateurs longs, réduisant la fatigue oculaire.
* **Navigation intuitive** : La barre de navigation doit être simple et accessible à tout moment, avec des menus bien structurés.
* **Avantage**: page pour lister les avantages d’application et aussi leurs fonctionnalités comme (Contact : Export de PDF et aussi envoie de mail) / (Produit : gestion de produit et votre stock ou ajout le produit) / (Projet : Dashboard personnaliser pour suivre les performances de votre entreprise) / (Vente et achat : création de devis personnaliser et de facture) / (Contapiliter : Gestion simplifier des factures et aussi exportation en Excel)

**6. Livrables**

Les livrables sont répartis en quatre phases, avec des jalons définis :

* **Phase 1** : Développer et tester la page d’accueil et le système d’authentification.
* **Phase 2** : Se concentrer sur la gestion des devis et factures avec une interface utilisateur simple.
* **Phase 3** : Ajouter la messagerie et les graphiques pour le tableau de bord.
* **Phase 4** : Optimisation des performances et renforcement de la sécurité avec des tests approfondis.

### 7. ****Budget Prévisionnel****

Le budget estimé semble raisonnable pour le développement et la mise en œuvre des fonctionnalités proposées. Une suggestion est d'ajouter un budget pour des licences logicielles spécifiques ou des outils tiers (comme des services d'emailing pour l'envoi des factures, ou un service d'authentification 2FA).

### 8. ****Technologie pour le développement****

Développement de l’application full stack

**1. Frontend (Interface utilisateur)**

**a. Framework JavaScript :**

* **React.js/Next.js** : Utilisé pour créer des interfaces dynamiques et interactives. React permet une gestion efficace des états et une meilleure expérience utilisateur avec des rendus rapides.

**b. Styling et Design :**

* **Tailwind CSS** : Framework CSS utilitaire pour un design réactif et rapide, qui permet de créer des interfaces élégantes et modernes avec un minimum d'effort.
* **CSS Grid et Flexbox** : Pour un agencement fluide et flexible des éléments sur la page.

**c. Gestion des états :**

* **Redux** ou **Context API** : Pour gérer l’état global de l'application, comme les utilisateurs connectés, les données dynamiques (devis, factures), et les informations du tableau de bord.

**d. Outils de routage :**

* **React Router** : Pour gérer la navigation entre les différentes pages de l'application.

**e. API et requêtes HTTP :**

* **Axios** : Pour effectuer des requêtes API (GET, POST, PUT, DELETE) de manière simple et flexible.

**f. Sécurité et authentification :**

* **JWT (JSON Web Tokens)** : Pour la gestion de l'authentification et des sessions utilisateur côté frontend, en collaboration avec le backend.

**g. Tests Frontend :**

* **Jest** et **React Testing Library** : Pour les tests unitaires et d’intégration des composants React.

**2. Backend (Serveur et logique métier)**

**a. Framework Backend :**

* **Spring Boot (Java)** : Framework puissant pour créer des applications backend robustes et sécurisées. Spring Boot facilite le développement rapide avec des configurations minimales et une architecture modulaire.

**b. API :**

* **RESTful API** : Utilisation d'API REST pour la communication entre le frontend et le backend, en utilisant des verbes HTTP standards (GET, POST, PUT, DELETE).

**c. Sécurité :**

* **Spring Security** : Pour sécuriser l'application backend, gérer l'authentification et les autorisations (par exemple, 2FA, rôles utilisateur).
* **JWT (JSON Web Token)** : Pour l'authentification et la gestion des sessions utilisateurs.

**d. Gestion des bases de données :**

* **JPA (Java Persistence API)** avec **Spring Data** : Pour l’interaction avec la base de données et la gestion des entités de manière élégante et simple.
* **MySQL** ou **PostgreSQL** : Bases de données relationnelles, robustes et bien supportées. PostgreSQL peut être préféré pour une gestion avancée des transactions.
* **Flyway** ou **Liquibase** : Pour la gestion des migrations de bases de données.

**e. Gestion des tâches en arrière-plan :**

* **Spring Batch** ou **Quartz** : Pour exécuter des tâches planifiées en arrière-plan, comme l'envoi de rappels de paiement ou la génération de rapports périodiques.

**f. Tests Backend :**

* **JUnit** et **Mockito** : Pour les tests unitaires et d’intégration du backend.

**3. Base de Données**

* **MySQL** ou **PostgreSQL** : Pour stocker les données des utilisateurs, des devis, des factures, et autres informations de l'entreprise.
* **Redis** (optionnel) : Utilisé pour le stockage en cache pour améliorer la performance de l'application, notamment pour les sessions et les données fréquemment consultées.

**4. Hébergement et Infrastructure**

**a. Frontend :**

* **Netlify** ou **Vercel** : Pour héberger l'application frontend. Ces plateformes offrent des déploiements faciles, optimisés pour les applications React, et incluent des fonctionnalités comme la gestion des certificats SSL.

**b. Backend :**

* **AWS EC2**, **Google Cloud App Engine** ou **Heroku** : Pour héberger l'application backend. Ces services offrent des solutions évolutives et une gestion simplifiée de l'infrastructure serveur.
* **Docker** (optionnel) : Pour containeriser l'application backend et faciliter le déploiement sur différentes infrastructures.

**c. Base de Données Cloud :**

* **AWS RDS (MySQL/PostgreSQL)** ou **Google Cloud SQL** : Pour héberger la base de données sur le cloud, avec une gestion simplifiée des backups et de la scalabilité.

**d. CDN (Content Delivery Network) :**

* **Cloudflare** ou **AWS CloudFront** : Pour distribuer le contenu statique rapidement et de manière sécurisée à travers le monde.

**5. Outils de Collaboration et de Gestion de Projet**

* **GitHub** ou **GitLab** : Pour la gestion du code source avec des branches et des workflows de développement.
* **Jira** ou **Trello** : Pour la gestion de projet et le suivi des tâches.
* **Slack** : Pour la communication d'équipe.

**6. Outils de Monitoring et Logs**

* **Sentry** : Pour le suivi des erreurs côté frontend et backend.
* **Prometheus** et **Grafana** : Pour la surveillance de la performance et la gestion des alertes.

**7. Tests et Validation**

* **Selenium** ou **Cypress** : Pour les tests de bout en bout (tests d’intégration sur toute l’application).
* **SonarQube** : Pour l’analyse statique du code et la détection des vulnérabilités.

**8. Autres Technologies Complémentaires**

* **Mailgun** ou **SendGrid** : Pour l’envoi des emails (factures, devis, notifications).
* **Twilio** : Pour l’intégration des notifications par SMS si nécessaire (rappels de paiement ou alertes importantes).