Spécifications techniques

[Menu Maker by Qwenta]

Version	Auteur	Date	Approbation
1.0	Natacha - Webgencia	05/11/2024	John - Qwenta

I. Choix technologiques

- État des lieux des besoins fonctionnels et de leurs solutions techniques :

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Maquette du site	- Doit être intuitive et refléter l'identité de la marque. - Facilité de partage et de collaboration.	**- Figma** *(outil de design collaboratif populaire)*	Utiliser **Figma** pour créer une maquette interactive du site. *(Wireframes / Prototypes cliquables / Design visuel complet)*	1. Collaboration en temps réel, permettant à l'équipe de travailler ensemble et d'apporter des modifications instantanément. 2. Prototypage interactif qui aide à visualiser l'expérience utilisateur avant le développement.
Landing	- Doit être intuitive pour les nouveaux utilisateurs. - Chargement rapide pour ne pas perdre d'internautes.	**- React** *(Framework Frontend moderne, permet d'intégrer facilement des composants interactifs et d'assurer une expérience utilisateur fluide)*	Utiliser **React** pour construire une landing page dynamique et réactive. *(Bannière / Section "Personnalisez votre menu" / Explications étape par étape)*	1. Amélioration de l'expérience utilisateur avec une interface engageante et réactive. 2. Composants réutilisables pour simplifier la maintenance et les futures mises à jour.
Connexion	- Sécurisé pour protéger les informations des utilisateurs. - Accessible depuis toutes les pages de l'application.	**- Firebase Authentication** *(gérer l'authentification, y compris la gestion des sessions et des envois d'emails de confirmation)*	Utilisation de **Firebase Authentication** pour gérer la connexion et la déconnexion des utilisateurs, avec une **modale** pour l'entrée de l'email et un envoi automatique d'email de confirmation.	1. Réduction de la complexité de la gestion des utilisateurs grâce à un service d'authentification prêt à l'emploi. 2. Mise en place rapide et sécurisée, assurant une bonne expérience utilisateur.

Dashboard (Accès aux articles de blog)	- Afficher dynamiquement les 3 derniers articles du blog de Qwenta. - Afficher le titre, l'image et le lien de chaque article.	**- React** *(pour le développement du dashboard)* **- Firebase Firestore** *(base de données NoSQL orientée documents, qui se prête bien au stockage de données flexibles et à la mise à jour en temps réel)*	Utiliser un composant **React** qui interroge **Firestore** pour récupérer les trois derniers articles de blog et les afficher sous forme de cartes (cards), avec titre, image de couverture, et lien.	1. **Firestore** permet une mise à jour automatique des articles sans reconfiguration de serveur, restant cohérent avec Firebase Authentication. 2. **React** fournit une structure réactive et facile à maintenir, en plus de s'adapter facilement aux besoins d'affichage du dashboard.
Ajout de catégories / Ajout de plats	- Interface claire et rapide à utiliser pour l'ajout de catégories. - Validation du nom de la catégorie. - Les champs de saisie doivent être bien validés (image, nom, prix, description dans une modale). - Télécharger une image en toute simplicité.	**- React** *(framework JS populaire qui permet de construire des interfaces utilisateur dynamiques et réactives)* **- React Modal** *(bibliothèque React qui permet d'afficher des fenêtres modales sur la page)* **- React Hook Form** *(bibliothèque qui facilite la gestion des formulaires en React – reduit le code nécessaire pour manipuler les données du formulaire)* **- Form Validation** *(processus permettant de vérifier les entrées des utilisateurs dans un formulaire avant de soumettre les données)* **- File Upload** *(fonction qui permet à un utilisateur de télécharger des fichiers depuis son appareil vers une plateforme en ligne)*	**React** est utilisé pour gérer l'état des champs et la dynamique de l'interface. **React Modal** permet d'afficher les fenêtres modales pour l'ajout des catégories et des plats. Les champs sont validés avec des **inputs contrôlés** pour le nom, le prix et la description. Le téléchargement d'images est facilité avec un **file input**. La validation des données est réalisée à l'aide de **React Hook Form** pour une gestion facile des formulaires.	1. **React** assure une gestion fluide et réactive de l'interface utilisateur, notamment avec les composants contrôlés pour valider chaque champ au fur et à mesure. 2. **React Modal** simplifie l'affichage dynamique des formulaires dans une fenêtre modale sans perturber l'interface principale. 3. **React Hook Form** facilite la gestion des formulaires complexes, y compris la validation, le suivi de l'état des champs et l'envoi des données du formulaire à une API.

Personnalisation du menu (style)	- enregistrer leurs préférences de branding (Le restaurateur doit pouvoir ajouter / modifier / supprimer les couleurs de base ou le logo). - choisir de façon dynamique la typographie, la couleur du texte, la mise en page.	**- styled components** *(bibliothèque pour créer des composants avec des styles dynamiques tout en encapsulant ces styles dans des composants React)*	Utiliser Styled Components pour gérer le style dynamique et personnalisable des éléments du menu. Chaque composant peut être stylisé de manière unique en fonction des préférences sauvegardées du restaurateur.	1. **Styles dynamiques** : les styles peuvent être modifiés en fonction des préférences de l'utilisateur, ce qui permet une adaptation facile et personnalisée. 2. **Maintenance simplifiée** : les styles encapsulés dans les composants assurent une structure propre et réutilisable.
Stocker les images des plats	- Permettre l'upload d'images de plats lors de leur création ou modification. - Assurer un accès rapide et sécurisé aux images lors de la consultation des menus.	**- Firebase Storage** *(service de stockage d'images robuste, sécurisé et facilement intégrable avec d'autres outils Firebase)*	Utiliser Firebase Storage pour stocker les images des plats de manière sécurisée et optimiser le chargement des images via un CDN. Les images sont stockées dans le cloud et accessibles via une URL unique.	1. **Stockage sécurisé et évolutif** : Firebase Storage offre une gestion sécurisée et adaptée aux besoins d'évolution, avec un accès rapide et sécurisé. 2. **Optimisation pour le chargement** : le CDN de Firebase permet de minimiser les temps de chargement des images.
accès à une vue regroupant les menus créés précédemment	- Afficher les menus avec la date de création. - Permettre la modification, la suppression de menus existants et la création d'un nouveau menu. - Assurer la synchronisation en temps réel pour voir les changements immédiatement.	**- Firebase Firestore** *(base de données NoSQL orientée documents, qui se prête bien au stockage de données flexibles et à la mise à jour en temps réel)*	Utiliser Firebase Firestore pour stocker les informations des menus (nom, date de création, détails, etc.). La vue est mise à jour en temps réel pour afficher les ajouts, modifications ou suppressions de menus.	1. **Mise à jour en temps réel** : Firestore permet de gérer les données en temps réel, garantissant une interface utilisateur réactive où les modifications apparaissent immédiatement. 2. **Gestion flexible des données** : Firestore facilite l'ajout, la modification et la suppression de documents (menus) sans complexité de gestion.

Diffuser/Exporter un menu	- Possibilité d'exporter le menu en PDF pour le télécharger. - Publier le menu sur Deliveroo. - Partager le menu sur Instagram.	**- React PDF** *(idéal pour générer un PDF directement au sein de l'application React)* **- API d'intégration pour Deliveroo** *(permet une publication du menu de façon automatisée sur Deliveroo)* **- API de partage pour Instagram** *(permet de partager le menu sous forme d'image ou de lien direct)*	- **Export PDF** : Utiliser React PDF pour générer dynamiquement un fichier PDF du menu dans l'application React, permettant aux utilisateurs de télécharger un fichier PDF à partir de l'interface. - **Intégration Deliveroo** : Configurer l'intégration API pour envoyer le menu directement sur Deliveroo. - **Partage Instagram** : Utiliser l'API Instagram pour publier le menu sous forme d'image ou de lien.	1. **Expérience utilisateur intégrée** : React PDF s'intègre directement avec React, permettant de générer et de personnaliser le PDF dans l'application sans nécessiter de solutions externes. 2. **Automatisation** : Les API de Deliveroo et Instagram assurent une diffusion rapide sur les plateformes externes, réduisant les tâches manuelles.
Commander l'impression d'un menu	- Permettre à l'utilisateur de commander facilement l'impression du menu en un clic. - Ouvrir un lien vers le backoffice de Qwenta dans un nouvel onglet pour confirmer et gérer l'impression.	**- React** *(bouton react qui peut être configuré pour ouvrir un nouvel onglet vers le backoffice de Qwenta)* **- Backoffice Qwenta** *(peut afficher les dernières informations du menu et offrir une personnalisation pour l'impression)*	Intégrer un bouton dans l'application utilisant React pour permettre l'impression en un clic. Le bouton redirige vers le backoffice de Qwenta (nouvel onglet) où les utilisateurs peuvent confirmer et personnaliser l'impression du menu.	1. **Expérience utilisateur simplifiée** : l'utilisateur peut commander l'impression en un seul clic et accéder directement au backoffice pour la gestion sans quitter l'application. 2. **Personnalisation flexible** : rediriger vers le backoffice permet d'ajuster les paramètres d'impression et de vérifier les dernières mises à jour du menu avant impression.

Gestion Compte	- Permettre aux utilisateurs de lier plusieurs adresses e-mail à leur compte. - Offrir la possibilité de modifier l'adresse e-mail principale de manière sécurisée.	**- Firebase Authentication** *(service de gestion des utilisateurs sécurisé, offrant des fonctionnalités pour ajouter, vérifier, et modifier des adresses e-mail)* **- Firebase Firestore** *(Chaque compte peut avoir un document unique dans Firestore contenant une liste d'adresses e-mail associées, permettant une gestion centralisée et sécurisée des e-mails)*	Utiliser **Firebase Authentication** pour gérer l'authentification et permettre l'ajout, la vérification et la modification d'adresses e-mail. Enregistrer les adresses e-mail secondaires dans **Firestore**, permettant de lier plusieurs e-mails au même compte.	1. **Sécurité et vérification** : Firebase Authentication fournit des fonctionnalités intégrées pour la vérification des e-mails et la gestion sécurisée de l'authentification et des permissions. 2. **Flexibilité pour les utilisateurs** : Firestore permet de stocker et gérer des e-mails supplémentaires, améliorant la flexibilité pour les utilisateurs souhaitant lier plusieurs adresses à leur compte.
Mentions légales	- Les informations légales doivent être accessibles rapidement et clairement. - Elles doivent être présentées dans une modale pour éviter de rediriger l'utilisateur. - Les éléments texte sont statiques et ne nécessitent pas de mises à jour fréquentes.	**- React Modal** *(composant React Modal est utilisé pour afficher les informations légales dans une fenêtre modale)*	- **Modale d'affichage** : Utiliser un composant React Modal pour afficher les informations légales sous forme de pop-up, permettant aux utilisateurs de les lire sans quitter la page en cours. - **Contenu statique** : Stocker les informations légales directement dans le code de l'application, car elles ne sont pas amenées à changer fréquemment.	1. **Simplicité et performance** : Puisque le contenu est statique, il est plus simple et performant de le stocker directement dans le code sans nécessiter un système de gestion externe comme Firestore. 2. **Accessibilité et expérience utilisateur** : La modale assure que l'utilisateur puisse consulter les informations légales de manière rapide et fluide sans interrompre la navigation dans l'application.

I. Liens avec le back-end

- Quel langage pour le serveur ?

  - **Node.js** : *Plateforme JavaScript qui permet d'exécuter du JavaScript côté serveur. Il est très populaire et performant pour les applications web en temps réel.*

  - **Express :** *Framework minimaliste pour Node.js qui facilite la gestion des routes, des requêtes HTTP et des middlewares. Il est utilisé pour créer des API backend.*

  => Compatibilité avec Firebase / Performance (rapide et adapté pour les applications) / Écosystème riche (grande bibliothèque de packages)

- A-t-on besoin d'une API ? Si oui laquelle ?

  - **API de Firebase** : *gestion identification utilisateur (Authentication) / gestion des données stockées dans une base NoSQL (Firestore) / Stockage de fichiers (storage).*

  - **API Instagram** : *partage de menu.*

  - **API Deliveroo** : *diffusion de menu.*

- Base de données choisie :

  - **Firebase Firestore** (Base de données NoSQL)

  => Intégration fluide / Base de données flexible et évolutive / Gestion en temps réel des données / Simplicité et rapidité de développement.

I. Préconisations concernant le domaine et l'hébergement

- Nom du domaine :

  - *Voir si Qwenta possède déjà un nom de domaine pour envisager un sous-domaines, autrement* ***menu-maker.fr*** *serait idéal pour le site.*

- Nom de l’hébergement :

  - **Firebase Hosting :** *Idéal pour les applications web React qui utilisent Firebase pour la base de données et l’authentification. Firebase Hosting permettra de gérer à la fois le front-end et les intégrations backend en un seul endroit avec peu de configuration supplémentaire, tout en bénéficiant d’un excellent temps de chargement et de sécurité.*

- Adresses e-mail :

  - *Voir en fonction du nom du domaine ou sous domaine choisi, mais voici un exemple* : **contact@menu-maker.fr** ou alors **contact@qwenta.fr**

I. Accessibilité

- Compatibilité navigateur :

  - *Les dernières versions de **Chrome, Safari et Firefox**.*

- Types d’appareils :

  - *Le site devra être en version **desktop**. Pas de version mobile à développer ni à prévoir.*
  (L'application devra être accessible au minimum : navigable depuis le clavier, et lisible par un lecteur d'écran).

I. Recommandations en termes de sécurité

- **Sécurité des comptes utilisateurs** *: activer l’authentification à deux facteurs (2FA) pour les comptes administrateurs / Limiter le nombre de tentatives de connexion pour éviter les attaques par force brute (rate limiting) / Mettre en place des règles de complexité pour les mots de passe / Envoyer des notifications par e-mail en cas de connexion depuis un nouvel appareil ou une nouvelle localisation.*

- Protection des Données dans Firebase *: S’assurer que les règles de sécurité dans Firebase Firestore et Firebase Storage sont correctement configurées / Chiffrement des données (Firebase chiffre automatiquement les données en transit (HTTPS) et les données au repos) / Audit des accès (Surveiller les accès et actions effectués dans Firebase pour repérer des anomalies).*

- Sécuriser les APIs Externes (Deliveroo, Instagram) *: Stocker les clés API dans des variables d'environnement sécurisées (plutôt que dans le code) / Donner uniquement les permissions strictement nécessaires pour chaque API.*

- Sécurité des Plugins et Bibliothèques *: Utiliser des plugins et des bibliothèques provenant de sources fiables (comme npm ou GitHub) et les maintenir à jour / Accorder aux plugins uniquement les permissions dont ils ont besoin pour fonctionner / Effectuez des audits de sécurité réguliers pour identifier des vulnérabilités potentielles dans les plugins ou dépendances.*

- **Sécurité de l'Hébergement et du Déploiement** : *S’assurer que l’application utilise HTTPS pour toutes les communications / Configurer des permissions de déploiement.*

- **Sécurité des Fichiers Téléchargés et des Images** *: Scan de sécurité pour les fichiers téléchargés / Limiter la taille et le type des fichiers.*

- **Surveillance des Activités** *: Enregistrer les activités clés des utilisateurs (connexion, modification de données, etc.) pour pouvoir identifier toute action suspecte / Configurer des alertes pour les connexions ou actions anormales, comme un nombre inhabituel de tentatives de connexion.*

- Sauvegarde et Récupération des Données *: S’assurer d’avoir des sauvegardes régulières de la base de données Firebase Firestore et Firebase Storage, pour pouvoir restaurer les données en cas d’incident / Préparer un plan de reprise après incident pour garantir que l'application peut être remise en ligne rapidement après un problème.*

I. Maintenance du site et futures mises à jour

- <u>Maintenance du site</u> :

- **Veille Technologique avec Feedly :** *suivre les évolutions des outils et bibliothèques utilisés (React, Firebase, etc.) pour assurer la compatibilité continue et à intégrer les mises à jour ou nouvelles fonctionnalités susceptibles d’améliorer la performance et la sécurité.*

- **Mises à jour régulières** *des dépendances et plugins (React, Firebase, etc.) pour prévenir les vulnérabilités et garantir la compatibilité.*

- **Surveillance de sécurité** : *Contrôles périodiques pour identifier les failles et renforcer les règles de sécurité (authentification, protection*

des données).

- **Correction des bugs** *: Résolution des problèmes techniques signalés par les utilisateurs dans un délai rapide, priorisant les dysfonctionnements bloquants.*

- **Adaptations aux nouvelles technologies** *et aux exigences des plateformes externes (API Deliveroo, Instagram) pour assurer la compatibilité continue.*

- Futures Mises à Jour :

- Ajout de nouvelles fonctionnalités sur demande : *par exemple, options de personnalisation pour les menus, ou intégration avec d'autres plateformes de livraison.*

- **Optimisation des performances** : *Améliorations pour réduire les temps de chargement et assurer une expérience utilisateur toujours fluide (ex : lazy loading pour le chargement différé des images, optimisation du code et des images...).*

- **Extension des APIs** : *Intégration de nouvelles APIs selon les besoins de Qwenta, pour faciliter la publication et le partage des menus sur d'autres canaux.*

- **Évolution de la base de données** : *Mise à jour de la structure ou ajout de nouvelles collections dans Firestore pour gérer plus efficacement les données à mesure que l'application évolue.*