Webgencia

Spécifications techniques

Menu Maker by Qwenta

Version	Auteur	Date	Approbation
1.0	Louis, Webgencia	05/09/2024	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)**
Authentification utilisateur et déconnexion	Connexion et gestion sécurisée des sessions	Firebase Authentication	Service sécurisé pour gérer à la fois l'authentification des utilisateurs et leur déconnexion	- Firebase offre une solution d'authentification clé en main, incluant la gestion des sessions et des identités sécurisées. - Intégration native avec d'autres services Firebase.
Personnalisation des menus et style de menu	Modification en temps réel du contenu et du style des menus	React.js + SCSS	Gestion de la personnalisation du contenu et du style des menus via des composants dynamiques, stylisés avec SCSS	- Modularité des composants - Flexibilité pour la maintenance et futures mises à jour grâce à SCSS
Gestion des images et branding restaurateur	Téléchargement sécurisé d'images et de fichiers de branding	Firebase Storage + SCSS	Stockage des images de menus et des fichiers de branding (logos, couleurs) via Firebase, stylisation personnalisée via SCSS	- Sécurité liée à Firebase Authentication - SCSS facilite la gestion du style et des futurs changements

Partage des menus sur réseaux sociaux et Instagram	Publication des menus sur différentes plateformes de réseaux sociaux	API Instagram + Firebase Cloud Functions	Utilisation d'un module commun pour partager les menus sur plusieurs réseaux sociaux	- Automatisation des partages - Réduction des coûts et erreurs humaines
Dashboard utilisateur et gestion des infos	Gestion des informations utilisateurs	React.js + Firebase Realtime Database	Composants pour la gestion des informations utilisateurs à partir du dashboard	- Gestion en temps réel - Centralisation des données
Gestion des réservations	Interface utilisateur fluide et réactive	React.js + SCSS	Bibliothèque JavaScript pour des interfaces utilisateur modulaires et réactives, stylisées avec SCSS pour une gestion simple des modifications de style	- Mises à jour rapides sans rechargement - Grande flexibilité pour des changements futurs grâce à SCSS
Gestion des commandes	Persistance des données en temps réel	Firebase Realtime Database	Base de données NoSQL pour la gestion des données en temps réel.	- Synchronisation des données - Intégration native avec Firebase
Génération de rapports	Export des rapports de commandes en PDF	Firebase Cloud Functions + PDFKit	Bibliothèque PDFKit avec Firebase pour la génération de PDF sans serveur	- Exécution sans serveur - Flexibilité dans la génération de rapports PDF

Exportation PDF des menus	Exportation des menus personnalisés en PDF	PDFKit + Firebase Cloud Functions	Utilisation de PDFKit pour générer des fichiers PDF personnalisés	- Génération rapide - Compatible avec le stockage Firebase
Page de login	Authentification sécurisée	Firebase Authentication	Gestion sécurisée des connexions et des utilisateurs	- Facilité d'intégration - Conformité avec les standards de sécurité
Catégorie de plat	Gestion et création de catégories de plats	React.js	Composants interactifs pour organiser les plats par catégorie	- Flexibilité et réutilisabilité - Modularité des composants
Menus précédents	Historique des menus créés	Firebase Realtime Database	Sauvegarde des menus dans la base Firebase pour consultation ultérieure	- Sauvegarde en temps réel - Accès rapide à l'historique
Landing page non connectée	Page de présentation pour utilisateurs non-inscrits	React.js	Création d'une landing page informative avant inscription	- Vitesse de chargement - Interactivité grâce à React

Informations légales	Section dédiée aux mentions légales	React.js + SCSS	Composant dédié aux informations légales, stylisé avec SCSS	- Réutilisabilité - Facilité de mise à jour du style et de la structure grâce à SCSS
Tarifs	Affichage des tarifs des plats	Firebase Realtime Database	Stockage des informations tarifaires en temps réel	- Mise à jour en temps réel - Accessibilité rapide

II. Liens avec le back-end

- Le projet Menu Maker by Qwenta prévoit d'utiliser **Node.js** et **Express** pour la création du back-end. Cette architecture permettra de gérer les requêtes des utilisateurs tout en assurant une intégration fluide avec Firebase pour la gestion des données et des tâches côté serveur.

APIs :

- **API REST via Express.js** : Express.js sera utilisé pour mettre en place les API REST qui permettront à l'application de communiquer avec Firebase pour la gestion des menus, commandes, et utilisateurs.
    - **API Deliveroo**: Si disponibles, l'APIs de **Deliveroo** sera intégrées pour permettre l'import/export automatique des menus sur ces plateformes de livraison. Cela permettra aux restaurateurs de gérer leurs menus depuis une seule interface.
    - **API Graph de Meta** : Des API telles que l'API Graph de Meta seront prévues pour permettre le partage des menus sur des plateformes comme Instagram.

Base de données :

- **Firebase Realtime Database** sera utilisée pour stocker les données de manière réactive. Cette base de données NoSQL permet une synchronisation en temps réel, idéale pour gérer les commandes, menus, et informations utilisateurs dans un contexte de mise à jour continue.

III. Préconisations concernant le domaine et l'hébergement

Nom du domaine :

- Le nom de domaine sera potentiellement un sous-domaine de Qwenta. Cela devra être confirmé avec l'équipe de Qwenta avant la phase de développement.

Hébergement :

- **Firebase Hosting** sera utilisé pour l'hébergement de l'application. Ce service est intégré avec les autres fonctionnalités Firebase, ce qui facilitera la gestion de l'application, et garantira des performances élevées avec une tarification basée sur l'utilisation réelle des ressources.

Adresse email :

- contact@qwenta.fr
- infos@qwenta.fr
- assistance@qwenta.fr

IV. Accessibilité

Compatibilité navigateur :

- L’application sera compatible avec les dernières versions des navigateurs les plus utilisés (Chrome, Safari, Firefox).

Types d’appareils :

- Une version desktop sera développée en priorité, avec une interface responsive pour s’adapter aux différentes résolutions d’écran courantes.

V. Recommandations en termes de sécurité

Étant donné que l'équipe ne dispose pas d'expertise dédiée à la cybersécurité ou aux réseaux, nous mettons en place des solutions simples mais robustes, en nous appuyant sur des services externalisés qui assurent une sécurité éprouvée. Voici les recommandations de sécurité tenant compte des compétences front-end et back-end disponibles :

1. Utilisation des outils de sécurité de Firebase

Firebase propose des outils de sécurité clés en main qui seront largement utilisés pour réduire les risques de sécurité sans nécessiter une expertise interne :

- **Firebase Authentication** : Firebase gère automatiquement la sécurité des sessions utilisateurs et la gestion des identités. Cette solution offre des fonctionnalités avancées comme la vérification par e-mail et l'authentification multi-facteurs (MFA) que l'équipe pourra activer sans avoir à implémenter des systèmes de sécurité complexes.

- **Firebase Security Rules** : Firebase Security Rules sera utilisé pour contrôler l'accès aux données stockées dans la Firebase Realtime Database et Firebase Storage. Ces règles permettent de définir des permissions basées sur les rôles des utilisateurs (par exemple, seuls les restaurateurs peuvent modifier leurs propres menus et images). Une configuration initiale de base sera mise en place, mais elle pourra être facilement ajustée par les développeurs back-end à mesure que le projet évolue.

2. Validation côté serveur et côté client

- **Sanitisation des entrées utilisateur** : Toutes les entrées utilisateur dans les formulaires (comme les noms de plats, descriptions, etc.) seront vérifiés et validées à la fois côté client (front-end) et côté serveur (back-end). Cela permettra d'éviter les attaques courantes comme les injections de scripts (XSS) ou l'injection SQL, bien que Firebase utilise une base de données NoSQL.
- **Utilisation des middlewares de sécurité** : En utilisant **Express.js** pour le back-end, nous mettrons en place des middlewares de sécurité comme **Helmet.js** pour renforcer les en-têtes HTTP et protéger contre des vulnérabilités courantes (par exemple, clickjacking, XSS, sniffing de contenu).

3. HTTPS par défaut

- **Utilisation du protocole HTTPS** : Firebase Hosting prend en charge HTTPS par défaut, garantissant que toutes les communications entre les utilisateurs et le serveur sont cryptées. Cela réduit considérablement les risques d'attaques par interception (man-in-the-middle) ou d'espionnage des communications.

4. Gestion des tokens et sessions

- **Durée des sessions contrôlée** : Firebase Authentication permet de définir des durées de session pour les utilisateurs. Une session utilisateur expirera automatiquement après une période de 12 heures, et un lien d'authentification envoyé par e-mail sera valide pendant une durée de 1 heure. Cette configuration prévient les risques liés à l’utilisation prolongée de tokens compromis.
- **Revocation des tokens** : En cas de suspicion de compromission (par exemple, une tentative de connexion non autorisée), les tokens d'authentification de Firebase peuvent être révoqués à tout moment, forçant les utilisateurs à se reconnecter.

5. Journalisation et surveillance automatique

- **Firebase Crashlytics et Monitoring** : Firebase Crashlytics sera mis en place pour détecter les erreurs et plantages de l'application en temps réel, tandis que Firebase Monitoring sera utilisé pour surveiller les performances et identifier les problèmes potentiels (temps de réponse anormalement longs, erreurs de serveur). Ces outils permettront aux développeurs back-end de réagir rapidement en cas de problème, sans avoir à mettre en place des systèmes de surveillance complexes.

6. Sauvegarde et reprise après incident

- **Sauvegardes automatiques avec Firebase** : Firebase gère automatiquement la sauvegarde et la résilience des données, ce qui élimine le besoin pour l'équipe d'implémenter des processus de sauvegarde manuels. En cas de problème, Firebase permet une récupération rapide des données.

7. Limitation des accès administratifs

- **Gestion des rôles utilisateurs** : Seuls les utilisateurs disposant de rôles administratifs (comme les propriétaires de restaurants) auront accès à certaines fonctionnalités sensibles (modification des menus, gestion des commandes, etc.). Cette gestion des rôles sera implémentée via Firebase Security Rules et Firebase Authentication, afin d'assurer que seules les personnes autorisées peuvent accéder aux données sensibles.

VI. Maintenance du site et futures mises à jour

Dans le cadre de l’évolution continue de l’application, plusieurs étapes de maintenance et de développement seront planifiées pour améliorer l'expérience utilisateur et élargir les fonctionnalités. Les objectifs à moyen et long terme incluent :

- **Ajout de nouvelles fonctionnalités** : À mesure que l'application évoluera, des fonctionnalités comme la gestion multi-profils pour les restaurateurs ayant plusieurs établissements ou la possibilité d'intégrer des moyens de paiement en ligne pourront être ajoutées.
- **Intégration des retours utilisateurs** : Un suivi des feedbacks sera mis en place pour identifier les besoins et ajuster les fonctionnalités en conséquence.

- **Mises à jour des technologies et des performances** : L'infrastructure Firebase permettra une mise à jour régulière et une optimisation des performances au fil des itérations.
- **Optimisation des coûts** : En utilisant Firebase pour la plupart des services, nous réduirons la complexité de l'infrastructure tout en maintenant une approche flexible et scalable. Les coûts seront régulièrement ajustés en fonction des besoins de l'application et de son succès.

Le but est d'assurer une application qui puisse évoluer facilement tout en restant rentable et fonctionnelle, avec des itérations courtes et des améliorations continues.

VII. Tests et validation

Pour garantir la stabilité du code et s'assurer que chaque fonctionnalité fonctionne correctement, des **tests unitaires** seront mis en place à l'aide de **Jest**.

- **Composants React** : Les composants de l'interface utilisateur seront testés pour s'assurer que les éléments sont correctement rendus, que les états sont bien gérés et que les événements (comme les clics ou les soumissions de formulaires) fonctionnent comme prévu.
- **Appels API** : Les appels aux APIs (Firebase, Deliveroo) seront simulés et testés pour vérifier que les données sont correctement envoyées et reçues, et que les erreurs sont gérées de manière appropriée.
- **Logique métier** : La logique métier, comme les fonctions de calcul de prix ou de gestion des commandes, sera également testée pour garantir l'exactitude des résultats.

Ces tests unitaires seront intégrés dans le processus de développement pour être exécutés automatiquement avant chaque livraison de code. Ils garantiront une meilleure robustesse et faciliteront la maintenance lors des futures mises à jour.