

# Spécifications techniques

[Menu Maker by Qwenta]

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

I. Choix technologiques	2
II. Liens avec le back-end	7
III. Préconisations concernant le domaine et l'hébergement	7
IV. Accessibilité	8
V. Recommandations en termes de sécurité	8
VI. Maintenance du site et futures mises à jour	9

## 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	<ul style="list-style-type: none"> <li>- Doit être intuitive et refléter l'identité de la marque.</li> <li>- Facilité de partage et de collaboration.</li> </ul>	<b>- Figma</b> <i>(outil de design collaboratif populaire)</i>	Utiliser <b>Figma</b> pour créer une maquette interactive du site. <i>(Wireframes / Prototypes cliquables / Design visuel complet)</i>	<ol style="list-style-type: none"> <li>1. Collaboration en temps réel, permettant à l'équipe de travailler ensemble et d'apporter des modifications instantanément.</li> <li>2. Prototypage interactif qui aide à visualiser l'expérience utilisateur avant le développement.</li> </ol>
Landing	<ul style="list-style-type: none"> <li>- Doit être intuitive pour les nouveaux utilisateurs.</li> <li>- Chargement rapide pour ne pas perdre d'internautes.</li> </ul>	<b>- React</b> <i>(Framework Frontend moderne, permet d'intégrer facilement des composants interactifs et d'assurer une expérience utilisateur fluide)</i>	Utiliser <b>React</b> pour construire une landing page dynamique et réactive. <i>( Bannière / Section "Personnalisez votre menu" / Explications étape par étape)</i>	<ol style="list-style-type: none"> <li>1. Amélioration de l'expérience utilisateur avec une interface engageante et réactive.</li> <li>2. Composants réutilisables pour simplifier la maintenance et les futures mises à jour.</li> </ol>
Connexion	<ul style="list-style-type: none"> <li>- Sécurisé pour protéger les informations des utilisateurs.</li> <li>- Accessible depuis toutes les pages de l'application.</li> </ul>	<b>- Firebase Authentication</b> <i>(gérer l'authentification, y compris la gestion des sessions et des envois d'emails de confirmation)</i>	Utilisation de <b>Firebase Authentication</b> pour gérer la connexion et la déconnexion des utilisateurs, avec une <b>modale</b> pour l'entrée de l'email et un envoi automatique d'email de confirmation.	<ol style="list-style-type: none"> <li>1. Réduction de la complexité de la gestion des utilisateurs grâce à un service d'authentification prêt à l'emploi.</li> <li>2. Mise en place rapide et sécurisée, assurant une bonne expérience utilisateur.</li> </ol>

<p>Dashboard</p> <p>(Accès aux articles de blog)</p>	<ul style="list-style-type: none"> <li>- Afficher dynamiquement les 3 derniers articles du blog de Qwenta.</li> <li>- Afficher le titre, l'image et le lien de chaque article.</li> </ul>	<p><b>- React</b> (pour le développement du dashboard)</p> <p><b>- Firebase Firestore</b> (base de données NoSQL orientée documents, qui se prête bien au stockage de données flexibles et à la mise à jour en temps réel)</p>	<p>Utiliser un composant <b>React</b> qui interroge <b>Firestore</b> pour récupérer les trois derniers articles de blog et les afficher sous forme de cartes (cards), avec titre, image de couverture, et lien.</p>	<ol style="list-style-type: none"> <li>1. <b>Firestore</b> permet une mise à jour automatique des articles sans reconfiguration de serveur, restant cohérent avec Firebase Authentication.</li> <li>2. <b>React</b> fournit une structure réactive et facile à maintenir, en plus de s'adapter facilement aux besoins d'affichage du dashboard.</li> </ol>
<p>Ajout de catégories / Ajout de plats</p>	<ul style="list-style-type: none"> <li>- Interface claire et rapide à utiliser pour l'ajout de catégories.</li> <li>- Validation du nom de la catégorie.</li> <li>- Les champs de saisie doivent être bien validés (image, nom, prix, description dans une modale).</li> <li>- Télécharger une image en toute simplicité.</li> </ul>	<p><b>- React</b> (framework JS populaire qui permet de construire des interfaces utilisateur dynamiques et réactives)</p> <p><b>- React Modal</b> (bibliothèque React qui permet d'afficher des fenêtres modales sur la page)</p> <p><b>- React Hook Form</b> (bibliothèque qui facilite la gestion des formulaires en React – réduit le code nécessaire pour manipuler les données du formulaire)</p> <p><b>- Form Validation</b> (processus permettant de vérifier les entrées des utilisateurs dans un formulaire avant de soumettre les données)</p> <p><b>- File Upload</b> (fonction qui permet à un utilisateur de télécharger des fichiers depuis son appareil vers une plateforme en ligne)</p>	<p><b>React</b> est utilisé pour gérer l'état des champs et la dynamique de l'interface.</p> <p><b>React Modal</b> permet d'afficher les fenêtres modales pour l'ajout des catégories et des plats.</p> <p>Les champs sont validés avec des <b>inputs contrôlés</b> pour le nom, le prix et la description.</p> <p>Le téléchargement d'images est facilité avec un <b>file input</b>.</p> <p>La validation des données est réalisée à l'aide de <b>React Hook Form</b> pour une gestion facile des formulaires.</p>	<ol style="list-style-type: none"> <li>1. <b>React</b> assure une gestion fluide et réactive de l'interface utilisateur, notamment avec les composants contrôlés pour valider chaque champ au fur et à mesure.</li> <li>2. <b>React Modal</b> simplifie l'affichage dynamique des formulaires dans une fenêtre modale sans perturber l'interface principale.</li> <li>3. <b>React Hook Form</b> 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.</li> </ol>

Personnalisation du menu (style)	<ul style="list-style-type: none"> <li>- enregistrer leurs préférences de branding (Le restaurateur doit pouvoir ajouter / modifier / supprimer les couleurs de base ou le logo).</li> <li>- choisir de façon dynamique la typographie, la couleur du texte, la mise en page.</li> </ul>	<b>- styled components</b>  <i>(bibliothèque pour créer des composants avec des styles dynamiques tout en encapsulant ces styles dans des composants React)</i>	<p>Utiliser Styled Components pour gérer le style dynamique et personnalisable des éléments du menu.</p> <p>Chaque composant peut être stylisé de manière unique en fonction des préférences sauvegardées du restaurateur.</p>	<p><b>1. Styles dynamiques</b> : les styles peuvent être modifiés en fonction des préférences de l'utilisateur, ce qui permet une adaptation facile et personnalisée.</p> <p><b>2. Maintenance simplifiée</b> : les styles encapsulés dans les composants assurent une structure propre et réutilisable.</p>
Stocker les images des plats	<ul style="list-style-type: none"> <li>- Permettre l'upload d'images de plats lors de leur création ou modification.</li> <li>- Assurer un accès rapide et sécurisé aux images lors de la consultation des menus.</li> </ul>	<b>- Firebase Storage</b>  <i>(service de stockage d'images robuste, sécurisé et facilement intégrable avec d'autres outils Firebase)</i>	<p>Utiliser Firebase Storage pour stocker les images des plats de manière sécurisée et optimiser le chargement des images via un CDN.</p> <p>Les images sont stockées dans le cloud et accessibles via une URL unique.</p>	<p><b>1. Stockage sécurisé et évolutif</b> : Firebase Storage offre une gestion sécurisée et adaptée aux besoins d'évolution, avec un accès rapide et sécurisé.</p> <p><b>2. Optimisation pour le chargement</b> : le CDN de Firebase permet de minimiser les temps de chargement des images.</p>
accès à une vue regroupant les menus créés précédemment	<ul style="list-style-type: none"> <li>- Afficher les menus avec la date de création.</li> <li>- Permettre la modification, la suppression de menus existants et la création d'un nouveau menu.</li> <li>- Assurer la synchronisation en temps réel pour voir les changements immédiatement.</li> </ul>	<b>- Firebase Firestore</b>  <i>(base de données NoSQL orientée documents, qui se prête bien au stockage de données flexibles et à la mise à jour en temps réel)</i>	<p>Utiliser Firebase Firestore pour stocker les informations des menus (nom, date de création, détails, etc.).</p> <p>La vue est mise à jour en temps réel pour afficher les ajouts, modifications ou suppressions de menus.</p>	<p><b>1. Mise à jour en temps réel</b> : Firestore permet de gérer les données en temps réel, garantissant une interface utilisateur réactive où les modifications apparaissent immédiatement.</p> <p><b>2. Gestion flexible des données</b> : Firestore facilite l'ajout, la modification et la suppression de documents (menus) sans complexité de gestion.</p>

Diffuser/Exporter un menu	<ul style="list-style-type: none"> <li>- Possibilité d'exporter le menu en PDF pour le télécharger.</li> <li>- Publier le menu sur Deliveroo.</li> <li>- Partager le menu sur Instagram.</li> </ul>	<p><b>- React PDF</b></p> <p><i>(idéal pour générer un PDF directement au sein de l'application React)</i></p> <p><b>- API d'intégration pour Deliveroo</b></p> <p><i>(permet une publication du menu de façon automatisée sur Deliveroo)</i></p> <p><b>- API de partage pour Instagram</b></p> <p><i>(permet de partager le menu sous forme d'image ou de lien direct)</i></p>	<p><b>- Export PDF</b> : 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.</p> <p><b>- Intégration Deliveroo</b> : Configurer l'intégration API pour envoyer le menu directement sur Deliveroo.</p> <p><b>- Partage Instagram</b> : Utiliser l'API Instagram pour publier le menu sous forme d'image ou de lien.</p>	<p><b>1. Expérience utilisateur intégrée</b> : React PDF s'intègre directement avec React, permettant de générer et de personnaliser le PDF dans l'application sans nécessiter de solutions externes.</p> <p><b>2. Automatisation</b> : Les API de Deliveroo et Instagram assurent une diffusion rapide sur les plateformes externes, réduisant les tâches manuelles.</p>
Commander l'impression d'un menu	<ul style="list-style-type: none"> <li>- Permettre à l'utilisateur de commander facilement l'impression du menu en un clic.</li> <li>- Ouvrir un lien vers le backoffice de Qwenta dans un nouvel onglet pour confirmer et gérer l'impression.</li> </ul>	<p><b>- React</b></p> <p><i>(bouton react qui peut être configuré pour ouvrir un nouvel onglet vers le backoffice de Qwenta)</i></p> <p><b>- Backoffice Qwenta</b></p> <p><i>(peut afficher les dernières informations du menu et offrir une personnalisation pour l'impression)</i></p>	<p>Intégrer un bouton dans l'application utilisant React pour permettre l'impression en un clic.</p> <p>Le bouton redirige vers le backoffice de Qwenta (nouvel onglet) où les utilisateurs peuvent confirmer et personnaliser l'impression du menu.</p>	<p><b>1. Expérience utilisateur simplifiée</b> : l'utilisateur peut commander l'impression en un seul clic et accéder directement au backoffice pour la gestion sans quitter l'application.</p> <p><b>2. Personnalisation flexible</b> : rediriger vers le backoffice permet d'ajuster les paramètres d'impression et de vérifier les dernières mises à jour du menu avant impression.</p>

Gestion Compte	<ul style="list-style-type: none"> <li>- Permettre aux utilisateurs de lier plusieurs adresses e-mail à leur compte.</li> <li>- Offrir la possibilité de modifier l'adresse e-mail principale de manière sécurisée.</li> </ul>	<p><b>- Firebase Authentication</b></p> <p><i>(service de gestion des utilisateurs sécurisé, offrant des fonctionnalités pour ajouter, vérifier, et modifier des adresses e-mail)</i></p> <p><b>- Firebase Firestore</b></p> <p><i>(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)</i></p>	<p>Utiliser <b>Firebase Authentication</b> pour gérer l'authentification et permettre l'ajout, la vérification et la modification d'adresses e-mail.</p> <p>Enregistrer les adresses e-mail secondaires dans <b>Firestore</b>, permettant de lier plusieurs e-mails au même compte.</p>	<p><b>1. Sécurité et vérification :</b> Firebase Authentication fournit des fonctionnalités intégrées pour la vérification des e-mails et la gestion sécurisée de l'authentification et des permissions.</p> <p><b>2. Flexibilité pour les utilisateurs :</b> Firestore permet de stocker et gérer des e-mails supplémentaires, améliorant la flexibilité pour les utilisateurs souhaitant lier plusieurs adresses à leur compte.</p>
Mentions légales	<ul style="list-style-type: none"> <li>- Les informations légales doivent être accessibles rapidement et clairement.</li> <li>- Elles doivent être présentées dans une modale pour éviter de rediriger l'utilisateur.</li> <li>- Les éléments texte sont statiques et ne nécessitent pas de mises à jour fréquentes.</li> </ul>	<p><b>- React Modal</b></p> <p><i>(composant React Modal est utilisé pour afficher les informations légales dans une fenêtre modale)</i></p>	<p><b>- Modale d'affichage :</b> 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.</p> <p><b>- Contenu statique :</b> Stocker les informations légales directement dans le code de l'application, car elles ne sont pas amenées à changer fréquemment.</p>	<p><b>1. Simplicité et performance :</b> 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.</p> <p><b>2. Accessibilité et expérience utilisateur :</b> La modale assure que l'utilisateur puisse consulter les informations légales de manière rapide et fluide sans interrompre la navigation dans l'application.</p>

## 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 :
  - **Firestore Hosting** : *Idéal pour les applications web React qui utilisent Firestore pour la base de données et l'authentification. Firestore 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](mailto:contact@menu-maker.fr) ou alors [contact@qwenta.fr](mailto: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

- Maintenance du site :
  - **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.