Spécifications techniques

Menu Maker by Qwenta

Version	Auteur	Date	Approbation
1.0	Marie, Webgencia	Février 2025	Soufiane, Webgencia John, Quenta

Table des matières

Choix technologiques

État des lieux des besoins fonctionnels et de leurs solutions techniques

Structure du site et navigation

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Landing page	Présenter clairement l'outil	React (Static Page)	Utilisation de React pour afficher une page interactive et fluide	1) Réactivité et modularité de React 2) Facilité de maintenance
Navigation fluide	Interface intuitive	React Router	Gestion des routes et navigation entre pages	1) Permet des transitions fluides 2) Facilite la gestion des URLs dynamiques

Authentification et gestion utilisateur

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Connexion et inscription	Accès sécurisé	Firebase Authentication	Authentification via e-mail ou comptes Google/Facebook	1) Sécurisé et intégré 2) Gestion simplifiée des utilisateurs
Gestion des utilisateurs	Modifier les informations personnels	Firebase Firestore	Stockage des infos utilisateurs dans une base NoSQL	1) Données facilement accessibles 2) Permet des mises à jour en temps réel

Gestion des menus

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Création d'un menu	Interface intuitive	React avec modales	Interface interactive avec ajout dynamique de plats	1) Expérience utilisateur fluide 2) Moins de rechargements de page
Ajout/modification des plats	Temps réel	Firebase Firestore	Stockage des menus sous format JSON	1) Pas besoin de backend lourd 2) Synchronisation en temps réel
Stockage des images des plats	Bonne gestion des fichiers	Firebase Cloud Storage	Sauvegarde des images et récupération via URL	1) Sécurisé et scalable 2) Optimisé pour le web

Personnalisation du style et du design de l'interface

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Personnalisation du style	Personnalisation avancée du design	Material UI + SASS	Utilisation de Material UI pour les composants interactifs et SASS pour personnaliser les couleurs et typographies.	1) **Material UI** offre une base de composants réutilisables et un design moderne qui garantit une cohérence visuelle. 2) **SASS** permet de gérer de manière modulaire et flexible les styles via des variables et mixins, facilitant ainsi les personnalisations spécifiques pour les restaurateurs.
Composants réutilisables	Réutilisation des composants	Material UI	Material UI fournit des composants prêts à l'emploi (boutons, cartes, formulaires) pour accélérer le développement.	1) **Gain de temps :** les composants prêts à l'emploi permettent de réduire le temps de développement. 2) **Cohérence :** un design moderne et cohérent est intégré, ce qui assure une expérience utilisateur uniforme.
Personnalisation avancée	Nécessité d'un style unique	SASS	SASS permet une personnalisation avancée avec des variables, mixins et un contrôle plus fin sur les styles.	1) **Flexibilité :** offre un contrôle détaillé sur la personnalisation des couleurs, typographies, espacements, etc. 2) **Évolutivité :** permet d'adapter facilement les styles au fur et à mesure de l'évolution du projet et des besoins des utilisateurs.

Diffusion et exportation

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Exportation en PDF	Format universel	React PDF	Génération de fichiers PDF depuis le front-end	1) Permet aux restaurateurs d'imprimer facilement 2) Solution légère
Partage sur Instagram	Format adapté	API Instagram	Génération d'images adaptées au format Insta	1) Facilite la promotion des menus 2) Automatisation possible
Exportation vers Deliveroo	Intégration plateforme externe	API Deliveroo	Connexion avec Deliveroo pour afficher le menu	1) Attrait commercial 2) Gain de temps pour le restaurateur

Hébergement et sécurité

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Hébergement du site	Performance et sécurité	Firebase Hosting	Déploiement rapide et sécurisé	1) CDN optimisé par Google 2) Intégré avec Firebase
Protection des données	Sécurité des utilisateurs	Firebase Authentication + Firestore Rules	Règles de sécurité personnalisées pour chaque utilisateur	1) Protection des données sensibles 2) Accès restreint par rôle
Accès HTTPS	Sécurité renforcée	Certificat SSL (fourni par Firebase)	Activation automatique avec Firebase Hosting	1) Améliore la confiance des utilisateurs 2) Sécurise les échanges

Maintenance et évolutivité

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Mises à jour	Fiabilité du service	Firebase Functions	Permet d'exécuter du code backend sans serveur	1) Automatisation des tâches 2) Évolutif
Suivi des performances	Optimisation continue	Google Lighthouse	Audit des performances et accessibilité	1) Améliore l'expérience utilisateur 2) Identifie les points faibles

Liens avec le back-end

Langage pour le serveur

Le back-end sera développé en **Node.js** avec **Express.js** pour gérer les requêtes et les réponses HTTP de manière efficace.

Qu'est-ce que Node.js ?

- **Node.js** est un environnement d'exécution JavaScript côté serveur basé sur le moteur **V8 de Chrome**.
- Il permet d'exécuter du JavaScript en dehors du navigateur, ce qui est idéal pour le développement d'applications web rapides et évolutives.

Pourquoi utiliser Node.js ?

1. **Performances optimisées** : Grâce à son architecture asynchrone et événementielle, Node.js permet un traitement rapide des requêtes, réduisant les temps de réponse.
2. **Grande communauté et nombreux modules** : Un vaste écosystème de packages NPM permet d'ajouter facilement des fonctionnalités.
3. **Unification du langage** : Utiliser JavaScript à la fois sur le front-end et le back-end simplifie le développement.

Qu'est-ce qu'Express.js ?

- **Express.js** est un framework minimaliste pour Node.js qui facilite la création d'API et d'applications web.
- Il permet de structurer les applications en fournissant des outils de gestion des routes et des middlewares.

Pourquoi utiliser Express.js ?

1. **Simplicité et rapidité de développement** : Permet de créer une API avec peu de code tout en restant flexible.
2. **Flexibilité et modularité** : Compatible avec de nombreux modules NPM pour étendre ses fonctionnalités.
3. **Performance et légèreté** : Express est optimisé pour exécuter des applications rapides sans surcharge inutile.
4. **Grande communauté et documentation** : Large écosystème avec une documentation complète et de nombreuses ressources.

API ?

Nous utiliserons une API **RESTful** pour la communication entre le front-end et le back-end. Mais également deux API tierces à savoir l'API Instagram et l'API Deliveroo

Pourquoi une API RESTful ?

1. Standard largement adopté, assurant compatibilité et scalabilité.
2. Simplicité d'intégration avec Firebase et les services tiers.

Gestion des erreurs API

Le back-end devra implémenter une gestion des erreurs API claire et normalisée en utilisant des codes HTTP appropriés (ex. 400 pour une requête invalide, 401 pour une authentification manquante, 403 pour un accès interdit, 500 pour une erreur serveur) et fournir des messages d'erreur détaillés pour faciliter le débogage et améliorer l'expérience utilisateur.

API tierces

- **API Instagram** : Permet le partage de menus sous forme d'image sur Instagram.
- **API Deliveroo** : Permet d'exporter les menus vers Deliveroo pour la mise en ligne.

Pourquoi ces APIs ?

1. Elles offrent une **visibilité accrue** aux restaurateurs.
2. Elles automatisent la publication des menus sur des plateformes populaires.

Base de données

Utilisation de **Firebase Firestore**, une base de données **NoSQL** permettant le stockage et la synchronisation en temps réel des données utilisateurs et des menus.

Pourquoi Firebase Firestore ?

1. **Synchronisation instantanée** : Firestore met automatiquement à jour les données sur tous les clients connectés, idéal pour une gestion en temps réel des menus.
2. **Scalabilité et flexibilité** : Firestore est conçu pour évoluer avec le nombre d'utilisateurs et permet de structurer les données sous forme de documents et collections, facilitant les mises à jour.
3. **Gestion avancée des permissions** : Grâce aux **Firestore Security Rules**, il est possible de définir précisément qui peut lire ou modifier chaque ressource, garantissant un accès sécurisé aux données.
4. **Intégration avec Firebase Authentication** : Permet d'associer facilement les menus aux utilisateurs connectés, simplifiant la gestion des comptes et des permissions.

Préconisations concernant le domaine et l'hébergement

Afin d'assurer une mise en ligne efficace et sécurisée de l'application, voici les choix recommandés concernant le domaine, l'hébergement et les adresses e-mails associés.

Nom du domaine

Actuellement, il est prévu que le site soit **un sous-domaine de Qwenta** mais cela reste **en cours de validation**. Cela permettrait de bénéficier de la notoriété et du référencement de la marque.

Exemple : `menumaker.qwenta.fr`

Si Qwenta souhaite à terme **détacher Menu Maker en un service indépendant**, il peut être judicieux de réserver un **nom de domaine propre** (

Exemples : `menumaker.com` ou `menumaker.app`

Un domaine dédié renforcerait l'identité propre de l'application et faciliterait sa communication auprès des utilisateurs.

Nom de l'hébergement

L'application sera hébergée sur **Firebase Hosting**, qui offre plusieurs avantages :

- **Performance** : Firebase utilise un **CDN (Content Delivery Network) global** pour une distribution rapide du contenu, optimisant les temps de chargement.
- **Sécurité** : Intégration native du **HTTPS** et gestion simplifiée des certificats SSL, garantissant des échanges sécurisés.
- **Déploiement rapide** : Firebase permet un déploiement instantané via une simple ligne de commande, ce qui facilite les mises à jour continues.
- **Intégration avec Firebase** : Puisque l'application repose sur Firebase Firestore, Authentication et Storage, l'utilisation de Firebase Hosting assure une **cohérence technique** et simplifie la gestion globale du projet.

Adresses e-mail

Pour la communication avec les utilisateurs et les partenaires, plusieurs adresses e-mail professionnelles sont envisageables :

- **contact@qwenta.fr** : Si l'application est sous le domaine de Qwenta, cette adresse maintiendrait une cohérence avec la marque.
- **contact@menu-maker.fr** : Si l'application dispose d'un domaine propre, cette adresse renforcerait la crédibilité et la communication avec les clients.

Accessibilité

L'accessibilité ne se limite pas à la compatibilité navigateur et aux types d'appareils. Elle vise à garantir une expérience inclusive pour tous les utilisateurs, y compris ceux ayant des handicaps visuels, moteurs ou cognitifs.

Les spécifications du projet indiquent que l'application doit être **navigable au clavier** et **lisible par un lecteur d'écran**, ce qui nécessite une approche plus large englobant plusieurs bonnes pratiques d'accessibilité.

Conformité aux standards d'accessibilité

L'application devra respecter les **WCAG 2.1 niveau AA** (W3C), qui couvrent notamment :

- Une navigation fluide sans souris (clavier uniquement).
- La compatibilité avec les lecteurs d'écran.
- Un contraste suffisant pour garantir la lisibilité.

Navigation et interaction

Navigation au clavier :

- Tous les éléments interactifs (boutons, liens, formulaires) doivent être accessibles via **Tab** et activables avec **Entrer**.
- L'ordre du focus doit suivre une **logique intuitive**.
- Des styles de focus clairs doivent indiquer visuellement la position actuelle (**outline, box-shadow**).

Compatibilité avec les lecteurs d'écran :

- Ajout d'attributs **ARIA** (W3C ARIA Authoring Practices) pour améliorer la compréhension des éléments interactifs.
- Utilisation de balises HTML sémantiques (<header>, <nav>, <main>, <footer>) pour structurer le contenu.
- Ajout de **textes alternatifs** (alt) sur les images importantes, comme le logo du restaurant.

Lisibilité et confort visuel

Contraste et visibilité :

- Respect des **normes WCAG 2.1** pour les couleurs (WebAIM Contrast Checker) :
- Éviter d'utiliser uniquement la couleur pour transmettre des informations (ex: erreurs de formulaire).

Personnalisation :

- Permettre l'**ajustement de la taille du texte** sans casser la mise en page.
- Offrir un **mode contraste élevé** ou **mode sombre** pour améliorer la lisibilité.

Compatibilité technique

Compatibilité navigateur

Navigateurs pris en charge : Chrome, Firefox, Safari, Edge.

Types d'appareils

Appareils : Version **desktop uniquement** (pas de version mobile prévue).

Tests et validation

Tests de l'accessibilité

Outils recommandés pour tester l'accessibilité :

- **Axe DevTools** (Deque Systems)
- **WAVE** (WebAIM)
- **Google Lighthouse** (Chrome DevTools)

Tests utilisateurs

- Vérifier la navigation avec **NVDA (Windows), VoiceOver (macOS/iOS) et TalkBack (Android)**.
- Impliquer des utilisateurs en situation de handicap pour identifier d'éventuelles barrières.

Tests de gestion des erreurs

Il est recommandé de tester systématiquement la gestion des erreurs en simulant des entrées invalides, des pannes serveur et des erreurs 404, afin de s'assurer que les messages affichés sont clairs, utiles et guidant l'utilisateur vers une résolution du problème.

Recommandations en termes de sécurité

Accès sécurisé aux comptes

Utilisation de **Firebase Authentication** avec des règles Firestore pour limiter l'accès aux données sensibles aux utilisateurs authentifiés.

Connexion sécurisée

Activation du **HTTPS** avec un **certificat SSL/TLS** pour chiffrer les échanges de données.

Protection des données

Mise en place des règles de sécurité **Firebase Cloud Storage** pour restreindre l'accès aux fichiers multimédias aux utilisateurs autorisés.

Validation des entrées

Filtrage strict des entrées utilisateur pour prévenir les **attaques XSS** (injection de script exécutable malveillant) **et injections SQL** (injection d'une requête SQL dans le but d'accéder à la base de données).

Sécurisation du code

Éviter de stocker des informations sensibles dans le code source.

Mises à jour régulières

Vérification et mise à jour des **dépendances, plugins et frameworks** pour corriger les failles de sécurité.

Protection des sessions

Utilisation de mécanismes sécurisés pour éviter le **session hijacking** (détournement de session utilisateur).

Sécurisation des Firebase Cloud Functions

Configuration stricte des règles d'accès pour empêcher tout accès non autorisé.

Surveillance et détection des menaces

Activation de l'**audit Firebase** pour détecter toute activité suspecte.

Gestion des erreurs

La gestion des erreurs doit être sécurisée en évitant toute fuite d'informations sensibles, en filtrant strictement les entrées utilisateur pour prévenir les injections, et en mettant en place des logs de surveillance pour détecter les comportements suspects.

Maintenance du site et futures mises à jour

Mises à jour et corrections

- Effectuer des mises à jour régulières via **Firebase Functions** pour assurer la stabilité et la sécurité de l'application.
- Surveiller et mettre à jour les **API tierces et dépendances** afin d'éviter les vulnérabilités de sécurité et garantir leur bon fonctionnement.
- Appliquer les correctifs de sécurité dès qu'ils sont disponibles pour éviter d'éventuelles failles exploitables.

Suivi des performances et surveillance

- Utiliser **Google Lighthouse** pour évaluer les performances du site, l'accessibilité et les bonnes pratiques SEO.
- Mettre en place une surveillance continue des erreurs et des performances via Firebase Monitoring et les logs d'activité.
- Configurer des **sauvegardes automatiques** pour garantir la récupération rapide des données en cas d'incident.

Contrôles réguliers et prévention

- Auditer régulièrement la sécurité et la conformité du code pour identifier et corriger les éventuelles failles.
- Effectuer des tests de régression après chaque mise à jour pour garantir la stabilité du site.
- Assurer une veille technologique active pour anticiper les évolutions et intégrer les meilleures pratiques du secteur.

Évolutions possibles du site

- **Accessibilité sur d'autres appareils** : Bien que la première version soit prévue uniquement pour desktop, une évolution vers une compatibilité tablette et mobile pourra être étudiée selon les besoins des utilisateurs.
- **Amélioration de l'accessibilité** : Intégration de nouvelles fonctionnalités pour rendre le site encore plus accessible (ex. mode sombre, meilleure prise en charge des lecteurs d'écran, navigation vocale).
- **Personnalisation avancée** : Possibilité d'ajouter plus d'options de personnalisation pour les menus (ex. choix de modèles prédéfinis, ajout d'images, etc.).
- **Intégration avec d'autres plateformes** : Développement d'API pour connecter **Menu Maker** à d'autres services populaires utilisés par les restaurateurs (Google My Business, TripAdvisor, etc.).
- **Optimisation des performances** : Amélioration continue du temps de chargement et optimisation des ressources pour garantir une expérience fluide même avec une forte demande.
- **Anticipation d'une alternative à Firebase** : Une étude sera menée pour évaluer des alternatives à Firebase, afin d'assurer plus de flexibilité, une meilleure maîtrise des coûts et garantir la continuité des services en cas de migration ou changement d'infrastructure.