Webgencia

Spécifications techniques

Menu Maker by Qwenta, Qwenta

Version	Auteur	Date	Approbation
1.0	Saber, Webgencia	10/04/2025	Soufiane, Webgencia

I. Choix technologiques 2

II. Liens avec le back-end 3

III. Préconisations concernant le domaine et l'hébergement 3

IV. Accessibilité 3

V. Recommandations en termes de sécurité 3

VI. Maintenance du site et futures mises à jour 4

I. Choix technologiques

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

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Création d'une catégorie de menu	L'ajout d'une catégorie doit pouvoir se faire directement sur l'écran de création de menu depuis une modale.	React-modal	Cette librairie React permet de créer simplement des modales performantes, accessibles avec un minimum de code.	1) Nous avons choisi de développer en React, la librairie est cohérente avec ce choix. 2) Il s'agit de la librairie la plus utilisée.
Création de menu dynamique	Interface fluide avec mise à jour en temps réel lors de l'ajout de plats, prix, etc…	React	Framework frontend basé sur les composants réutilisable avec une base de données NoSQL adaptée à des structures complexes et flexibles	1) Performance optimale pour les interfaces dynamiques. 2) Large communauté, outils modernes (hooks, context).
Branding du restaurant	Le restaurateur peut enregistrer ses préférences (logo, couleurs).	Stockage en base de données MongoDB + Context API côté Front	Les préférences sont liées à l'utilisateur et chargées via le Context API dans React	1) Centralise les données branding. 2) Le Context API permet un accès rapide à tout moment dans l'interface.
Export PDF	Le restaurateur doit pouvoir générer un PDF.	Librairie html2pdf	Bibliothèques JS pour convertir une page HTML en fichier PDF côté client	1) Pas besoin de serveur pour l'export. 2) Intégration simple dans React

Diffusion sur Deliveroo	Le restaurateur doit pouvoir diffuser le menu sur Deliveroo.	Export CSV avec la librairie csvjson	Génération d'un fichier CSV contenant les données du menu conforme aux attentes du Partner Hub de Deliveroo.	1) Le Partner Hub de Deliveroo accepte les imports de menu en CSV. 2) Le CSV est un format simple, universel et facile à générer depuis Node.js.
Partage sur Instagram	Le restaurateur doit pouvoir partager un visuel du menu via Instagram.	Génération d'image et lien de partage	Génération d'un visuel à partir du menu avec la librairie html-to-image et création d'un lien partageable.	1) Instagram autorise le partage d'images. 2) Permet de conserver l'identité visuelle du menu
Imprimer un menu	Impression directe depuis le navigateur ou en PDF.	Fonction d'impression native (window.print)	Impression classique du navigateur ou via le téléchargement d'un fichier PDF.	1) Simple, sans configuration. 2) Garantit une impression fiable et cohérente, quel que soit le type de contenu du menu.
Connexion front-back	Échanges de données entre client et serveur asynchrones.	Express.js et fetch	Express expose les routes backend, fetch envoie les requêtes depuis React.	1) Express est léger, idéal pour les APIs. 2) Fetch gère bien les erreurs et intercepteurs.
Authentification du restaurateur	Interface privée avec authentification sécurisée.	Librairies JWT et bcrypt	Hachage des mots de passe avec bcrypt + génération de tokens JWT pour les sessions.	1) Pratique pour les API. 2) Sécurité assurée sans gestion de session côté serveur.

Upload du logo	Permettre au restaurateur de téléverser son logo.	Multer et Cloudinary	Middleware Express js pour le traitement du fichier	1) Multer est le standard Node.js pour l'upload. 2) Cloudinary offre des URLs optimisées, faciles à intégrer dans le frontend.
Pas de version mobile	L'interface est exclusivement desktop.	Design responsive fixe	Application conçue uniquement pour une utilisation sur grand écran	1) Limite la complexité. 2) Permet de se concentrer sur l'expérience desktop optimale.
Choix des couleurs	Interface personnalisable avec sélection de police et de couleur.	Librairie styled-components	Liste déroulante personnalisée pour le choix, appliquée dynamiquement aux composants avec du CSS-in-JS.	1) Bonne expérience utilisateur. 2) styled-components facilite l'application de thèmes dynamiques.
Compatibilité navigateur	Doit fonctionner sur Chrome, Safari, Firefox (dernières versions).	Tests manuels et Babel	Utilisation de Babel pour transpiler le code + phase de test sur navigateurs ciblés.	1) Babel garantit une compatibilité JS maximale. 2) Tests ciblés permettent de limiter les bugs et les surprises d'affichage.
Accessibilité	Application navigable au clavier et lisible par lecteur d'écran.	Respect des standards ARIA et React best practices	Utilisation des rôles ARIA, tabIndex, composants accessibles (modale, formulaire, etc...).	1) Conformité aux bonnes pratiques d'accessibilité. 2) Facilite l'utilisation pour tous les profils d'utilisateurs.

II. Liens avec le back-end

- Quel langage pour le serveur ? *Ex. : NodeJS / PHP / Python, etc.*

➤ **Node.js**
Justification : choisi pour sa compatibilité avec MongoDB, son écosystème riche, et sa cohérence avec React côté frontend.

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

➤ **Oui**, une **API** construite avec **Express.js**
Justification : Express est léger, performant, et permet une structuration simple des routes pour gérer les menus, les utilisateurs, les préférences de branding, etc…

- Base de données choisie : *Ex : SQL / NO SQL*.

➤ **NoSQL** (**MongoDB)**
Justification : MongoDB permet une structure souple pour les menus, parfait pour des documents contenant catégories, plats, préférences, etc…

III. Préconisations concernant le domaine et l'hébergement

- Nom du domaine.

➤ Un **sous-domaine de Qwenta**, comme indiqué dans les spécifications fonctionnelles *(*en cours de validation *selon le PDF)*.

- Nom de l'hébergement.

➤ Préconisation : frontend (React) sur **Vercel**, le backend (Node.js + Express) sur **Hostinger VPS** et **MongoDB Atlas** pour la base de données.

- Adresses e-mail.

➤ Exemple : contact@qwenta.fr. À prévoir selon le nom de domaine utilisé.

IV. Accessibilité

- Compatibilité navigateur.

➤ Chrome, Firefox, Safari (dernières versions)
Conformément aux spécifications fonctionnelles. Pas besoin d'optimisation Internet Explorer ou autres navigateurs secondaires.

- Types d'appareils.

➤ **Uniquement desktop**
Pas de version mobile prévue.

V. Recommandations en termes de sécurité

- Accès aux comptes, plugins...

➤ Authentification sécurisée avec **JWT** côté client et **bcrypt** pour le hashage des mots de passe.

➤ Les plugins tiers (ex : Cloudinary pour les images) doivent être configurés avec des **variables d'environnement** pour sécuriser les clés API.

➤ Utiliser **HTTPS** sur l'ensemble du site, et gérer les droits d'accès par rôle côté back (admin, restaurateur, etc…).

VI. Maintenance du site et futures mises à jour

- Grandes lignes du contrat de maintenance.
  - **Corrections de bugs** : résolution rapide des dysfonctionnements signalés.
  - **Mises à jour techniques** : dépendances npm.
  - **Évolutions fonctionnelles** : ajout de nouveaux canaux de diffusion (ex : Uber Eats, WhatsApp), nouveau format d'export, templates de menus.
  - **Support** : via e-mail dédié.