Webgencia

Spécifications techniques

[Menu Make by QWENTA]

Version	Auteur	Date	Approbation
1.0	Shafiq Mouawiyya	16/05/2025	Soufiane

I. Choix technologiques 2
II. Liens avec le back-end 5
III. Préconisations concernant le domaine et l'hébergement 6
IV. Accessibilité 6
V. Recommandations en termes de sécurité 7
VI. Maintenance du site et futures mises à jour 7

I. Choix technologiques

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

Besoin	**Contraintes**	**Solution**	**Description de la solution**	**Justification**
Application dynamique	L'utilisateur doit avoir une expérience rapide et performante	React + Vite	React permet de construire des interfaces dynamiques basées sur les composants, et Vite offre un environnement de développement rapide.	1) React est une bibliothèque adaptée pour les interfaces interactives et réactives. 2) Vite permet un chargement rapide, il est performant et moderne
Création d'un menu	La création du menu doit se faire sur une page dédiée accessible en cliquant sur "Créer un menu" depuis le dashboard	React Router	Permet de gérer la navigation entre les pages dans une application React en définissant des routes dynamiques.	1) Solution standard et bien intégrée à React pour le routage. 2) Facile à maintenir et à étendre pour d'autres pages du site

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.
Ajout d'un plat par catégorie	Lors de l'ajout d'un plat via une modale dédiée, l'utilisateur doit pouvoir ajouter une photo et voir un aperçu	react-dropzone + FileReader	react-dropzone permet d'uploader une image facilement via glisser-déposer ou clic ; FileReader (API JS) génère un aperçu instantané de l'image.	1) Permet une UX fluide et moderne (drag & drop ou clic). 2) Affichage immédiat de l'image sans upload serveur, ce qui améliore la réactivité et réduit les erreurs.
Personnalisation de la police	L'utilisateur doit pouvoir choisir une police parmi une liste	Google Fonts API	Fournit un large catalogue de polices web facilement intégrables dans une app React.	1) Large choix de polices compatibles web. 2) Facile à intégrer via CSS ou React.
Enregistrement du logo	L'utilisateur doit uploader son logo et le voir apparaître immédiatement	react-dropzone	Permet de gérer les uploads de fichiers de façon intuitive (glisser-déposer).	1) UX moderne et fluide pour l'utilisateur. 2) Facile à intégrer dans un formulaire React.

Exportation en PDF	Le rendu PDF doit être fidèle à la version affichée à l'écran	react-pdf	Librairie permettant de générer des documents PDF à partir de composants React.	1) Permet un rendu précis basé sur les composants React. 2) Évite de recréer une version HTML différente pour l'export.
Partage sur Instagram	Le menu doit pouvoir être partagé sous forme d'image	html-to-image +(upload manuel)	Convertit un composant React en image (PNG/JPEG), pouvant être ensuite téléchargée ou partagée manuellement.	1) Permet de capturer visuellement un menu. 2) Alternative simple sans intégration directe complexe avec Instagram.
Publier sur Deliveroo	L'utilisateur doit pouvoir rendre son menu disponible sur Deliveroo directement depuis la plateforme	intégration API Deliveroo	intégration avec l'API partenaire Deliveroo pour automatiser la publication.	1) Permet une synchronisation fluide entre la plateforme et Deliveroo. 2) Réduction des doubles saisies pour les restaurateurs, donc gain de temps.
Génération de menus prêts à l'impression	Le menu imprimé doit conserver la mise en page choisie par le restaurateur	react-pdf	Génère un fichier PDF à partir des composants React, en respectant le style et la structure définis dans l'interface.	1) Garantit un rendu fidèle entre l'écran et le PDF. 2) Évite de dupliquer la logique d'affichage : même code pour affichage et export.

II. Liens avec le back-end

- Node.js avec Express.js est une solution optimale pour le back-end. car il est compatible naturellement avec React, qui permet d'unifier le langage utilisé (JavaScript). Node.js dispose également d'un écosystème mature, offrant une large bibliothèques utiles, et il est particulièrement performant pour la mise en place d'API REST.

- Une API REST sera nécessaire pour assurer la communication entre l'interface front-end développée en React et le serveur Node.js. Cette API permet notamment de gérer les menus, les utilisateurs. Le choix d'une API REST se justifie par sa simplicité d'implémentation, sa large adoption dans l'industrie et sa parfaite adéquation avec les besoins fonctionnels du projet.

- Pour la base de données, il est pertinent d'opter pour une solution NoSQL (MongoDB). Ce type de base orientée documents est particulièrement adapté pour stocker des données complexes et hiérarchiques comme les menus (avec catégories, plats, prix, images), ainsi que les coordonnées de connexion et les préférences utilisateur. MongoDB se distingue par sa flexibilité, sa facilité de mise en œuvre, et son intégration native avec Node.js.

III. Préconisations concernant le domaine et l'hébergement

- Le nom de domaine utilisé pour le projet sera très probablement un sous-domaine du site principal de Qwenta, conformément aux indications des spécifications fonctionnelles. Par exemple : menu.qwenta.com.

- Pour l'hébergement de l'application, il est important d'utiliser une solution qui offre une excellente compatibilité avec les stacks JavaScript (Node.js + React). Il est également essentiel de pouvoir déployer rapidement, avec des options gratuites ou peu coûteuses. On peut donc envisager d'utiliser des plateformes comme Render, Railway ou Vercel.

- Enfin, pour gérer les échanges clients, les notifications et autres communications, il est recommandé de créer une adresse e-mail professionnelle dédiée au projet, comme par exemple menu@qwenta.com.

IV. Accessibilité

- L'application devra être compatible avec les dernières versions stables des navigateurs Chrome, Firefox et Safari. Pour cela, la transpilation du code JavaScript sera assurée via Babel, intégré à l'outil de build Vite, afin de garantir le support des fonctionnalités modernes d'ECMAScript. Les bibliothèques utilisées devront également être compatibles avec les standards ESModules.

- Le développement suivra une approche desktop-first, sans breakpoints CSS, en ciblant une résolution standard pour écrans larges. Il n'est donc pas nécessaire d'implémenter de comportement responsive ou d'adaptation pour tablettes ou smartphones. La navigation clavier et la compatibilité avec les lecteurs d'écran devra être respectée. Tous les composants interactifs devront être accessibles via la touche "Tab" et signaler leur état via des attributs WAI-ARIA si nécessaire.

V. Recommandations en termes de sécurité

L’authentification des utilisateurs devra être sécurisée par l’utilisation de tokens transmis via le protocole HTTPS. Cette méthode garantit la confidentialité des identifiants et protège contre les attaques. Le front-end sera responsable de la gestion de l’interface d’authentification, tandis que le back-end devra vérifier et valider les tokens. Si l’hébergeur propose des services de sécurité intégrés, une partie de cette gestion pourra lui être déléguée

Les accès aux comptes administrateurs et utilisateurs devront être protégés par des mots de passe forts, et. L'accès aux plugins, outils d'administration ou à la base de données devra être restreint par rôles et géré côté back-end avec une journalisation des accès.

VI. Maintenance du site et futures mises à jour

Il y aura une maintenance côté front end pour garantir les fonctionnalités du site et l'expérience utilisateur avec la mise à jour des bibliothèques React et de ses dépendances, ainsi que la correction des éventuels bugs affectant l’expérience utilisateur.

Côté back-end, l’hébergeur sera responsable de la maintenance de la base de données, de la sécurité des flux , et de la disponibilité des services. L’usage d’une solution managée comme MongoDB Atlas permet de déléguer une partie de cette maintenance (sauvegardes, monitoring, mises à jour automatiques).

Le contrat de maintenance devra prévoir :

- Des mises à jour régulières des dépendances front et back.
- Une surveillance des vulnérabilités de sécurité.
- Un support en cas de panne ou d’incident critique.