Webgencia

Spécifications techniques

Menu Maker by Qwenta

Version	***Auteur***	***Date***	***Approbation***
1.0	*William/Webagencia*	*25/10/25*	*Sofiane/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)***
Création et modification d'un menu	*Le restaurateur doit pouvoir ajouter ou modifier un menu complet depuis une interface réactive, sans rechargement de page.*	*React.js*	*Bibliothèque JavaScript permettant de créer des interfaces modulaires et dynamiques via des composants réutilisables.*	*1) React assure un rendu instantané des éléments lors des modifications. 2) Elle est parfaitement adaptée aux applications monopage (SPA).*
Gestion des formulaires de menu	*Les restaurateurs doivent pouvoir remplir, modifier et valider les formulaires de création de menu sans erreurs ni rechargement.*	*Formik (librairie React)*	*Librairie permettant de gérer facilement l'état, la validation et la soumission des formulaires dans React.*	*1) Réduit le code et facilite la validation des champs (ex : nom, prix, catégorie).2) S'intègre parfaitement avec React Hook Form pour une meilleure gestion des erreurs.*
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.*

Upload d'image de plat ou de logo	*L'utilisateur doit pouvoir importer une image locale pour l'associer à un plat.*	*Multer (middleware Express)*	*Middleware Node.js gérant la réception, le nommage et le stockage des fichiers envoyés via des formulaires.*	*1) S'intègre directement dans une API Express. 2) Solution simple et sécurisée pour les fichiers légers (JPEG, PNG).*
Authentification et connexion sécurisée	*Seuls les restaurateurs autorisés doivent accéder à l'espace d'administration.*	*JWT + bcrypt + Express.js*	*Le mot de passe est chiffré avec bcrypt lors de l'inscription, et un token JWT est généré à la connexion pour authentifier l'utilisateur.*	*Respect des standards actuels de sécurité. Compatible avec React pour la gestion des tokens côté client.*
Exportation du menu en PDF	*Le restaurateur doit pouvoir exporter son menu imprimable directement depuis le navigateur.*	*react-to-print*	*Ce composant React permet d'imprimer ou d'enregistrer une partie de l'interface directement en PDF via le moteur d'impression du navigateur.*	*1) Solution légère et simple à intégrer dans une application React. 2) Offre un rendu fidèle, sans dépendre d'une librairie externe complexe ou d'un traitement serveur.*
Sauvegarde des menus créés	*Les menus doivent être stockés et consultables à tout moment.*	*MongoDB (NoSQL)*	*Base de données orientée documents permettant de gérer des structures flexibles.*	*1) Le modèle NoSQL est adapté à des données évolutives. 2) Intégration fluide avec Node.js via Mongoose.*
Personnalisation du style du menu	*Le restaurateur doit pouvoir modifier les couleurs, polices et agencements.*	*React Context + Styled Components*	*Utilisation du contexte React pour stocker les préférences et des composants stylés*	*1) Permet de gérer des thèmes globaux et dynamiques. 2) Sépare la logique de la mise en*

			dynamiques.	*forme pour un code plus propre.*
Interface responsive et ergonomique	*L'application doit être utilisable sur mobile, tablette et ordinateur.*	*TailwindCSS*	*Framework CSS utilitaire permettant de concevoir des interfaces adaptatives rapidement.*	*1) Permet un prototypage rapide et cohérent. 2) Facilite la gestion du responsive design sans surcharge CSS.*
Communication entre front et back	*Le front doit interagir en toute sécurité avec les routes API du back-end.*	*Axios (client HTTP)*	*Librairie facilitant les appels HTTP avec gestion des tokens JWT et des erreurs.*	*1) Syntaxe simple et promesses natives. 2) Meilleure gestion des erreurs que fetch().*
Déploiement et hébergement	*L'application doit être accessible en ligne, avec base de données connectée.*	*Vercel (front) + MongoDB Atlas*	*Déploiement du front via Vercel, hébergement de la base de données avec MongoDB Atlas.*	*1) CI/CD intégrée pour React. 2) Hébergement sécurisé et évolutif pour MongoDB.*

II. Liens avec le back-end

- ***Quel langage pour le serveur ?*** *Le serveur* ***sera développé*** *avec* ***Node.js****, associé au framework* ***Express.js****. Node.js* ***permettra*** *d'exécuter du code JavaScript côté serveur et d'utiliser un seul langage sur l'ensemble du projet. Express.js servira à créer les routes,* ***gérer les middlewares et établir la communication entre le front-end et la base de données.***

- ***A-t-on besoin d'une API ? Si oui laquelle ?*** *Une* ***API REST*** *sera mise en place avec Express.js pour relier le*

front React à la base de données MongoDB. Les échanges se feront au format JSON via les routes principales (GET, POST, PUT, DELETE). Des API externes pourront être ajoutées ultérieurement :

***Deliveroo** → synchronisation automatique des menus et des mises à jour d'articles.*
***Instagram Graph** → publication ou partage visuel des menus sur les comptes professionnels.*

- ***Base de données choisie ?** Le projet utilisera **MongoDB**, une base **NoSQL hébergée sur MongoDB Atlas**. Elle sera adaptée à des données flexibles comme les plats, les images ou les préférences de style. Grâce au module Mongoose, les données seront structurées et validées avant chaque enregistrement.*

III. Préconisations concernant le domaine et l'hébergement

- *Nom de domaine : Le site **pourrait être accessible** via le domaine : **www.menu-maker-qwenta.fr** Ce nom de domaine **serait réservé** par le client Qwenta afin de garantir la cohérence avec son identité de marque. Un nom de domaine court et explicite favoriserait la visibilité et la mémorisation par les utilisateurs.*

- *Nom de l'hébergement : Le projet pourrait être hébergé sur **Hostinger**, un hébergeur professionnel **proposant des offres cloud compatibles avec Node.js**. Cette solution permettrait de centraliser le front-end, le back-end et le nom de domaine au même endroit, tout en conservant la base de données sur **MongoDB Atlas**.*

- *Adresses e-mail : Des adresses e-mail dédiées pourraient être créées afin d'assurer une gestion*

professionnelle du site et des échanges : ***contact@menu-maker-qwenta.fr*** *→ service client et demandes d'informations.* ***admin@menu-maker-qwenta.fr*** *→ administration technique du site.*
no-reply@menu-maker-qwenta.fr *→ envoi automatique des notifications (confirmation adresse mail, etc.).*
Ces adresses contribueraient à renforcer la crédibilité et la fiabilité du service auprès des utilisateurs.

IV. Accessibilité

- *Compatibilité navigateur :* ***Google Chrome****,* ***Mozilla Firefox****,* ***Safari conformément aux spécifications fournies par le client.*** *Les standards du* ***W3C*** *seraient suivis pour garantir le bon comportement du site sur chaque navigateur.*

- *Types d'appareils : Le site* ***serait optimisé principalement pour un usage sur ordinateur****. Une* ***adaptation partielle sur tablette*** *pourrait être envisagée ultérieurement.*

V. Recommandations en termes de sécurité

- *Accès aux comptes, plugins... :*
  - *L'accès aux comptes* ***serait protégé*** *par un système d'authentification* ***JWT (JSON Web Token)****. Chaque utilisateur* ***devrait se connecter*** *avec son e-mail et un mot de passe unique. Les mots de passe* ***seraient chiffrés*** *grâce à la librairie* ***bcrypt*** *avant leur enregistrement dans la base de données. Un* ***système de limitation des tentatives de connexion*** *serait également mis en place afin de* ***prévenir les***

***attaques par force brute**. Les tokens JWT **auraient une durée de validité limitée** pour éviter les connexions prolongées non sécurisées.*

- *Les **plugins et dépendances** (React, Express, Mongoose, Multer, etc.) **seraient régulièrement mis à jour** afin de corriger les éventuelles failles de sécurité. Un contrôle automatique via **npm audit** ou **Snyk permettrait d'identifier** les vulnérabilités connues dans les bibliothèques externes.*

VI. Maintenance du site et futures mises à jour

- *Grandes lignes du contrat de maintenance.*
  Une phase de maintenance régulière serait prévue après la mise en production du site. Elle porterait principalement sur :
  - *la **correction des anomalies mineures** (affichage, compatibilité ou bugs liés aux mises à jour),*
  - *la **mise à jour des dépendances** (React, Express, Mongoose, etc.),*
  - *la **surveillance de la base de données MongoDB Atlas** pour garantir la continuité du service,*
  - *la **vérification périodique des sauvegardes** et de la sécurité des accès utilisateurs.*
  - *Les futures mises à jour **pourraient inclure** :l'intégration d'**API externes** (Deliveroo, Instagram Graph),*
  - *une **optimisation de l'interface tablette ou mobile**,*
  - ***optimisation du référencement naturel ou payant** à termes afin de gagner en visibilité,*

- *ou encore l'ajout de* ***nouvelles options de personnalisation des menus***.