Webgencia

Spécifications techniques

Menu Maker by Qwenta + John-Qwenta

Version	Auteur	Date	Approbation
1.0	Stéphanie	29 janvier 2025	Soufiane/ John de Qwenta

I. Choix technologiques : React + Vite.js

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

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Landing page	Elle doit permettre aux utilisateurs arrivant sur le site de comprendre l'utilité de l'application et inciter ces derniers à s'inscrire.	-React -Framework TailwindCSS	React permet d'afficher dynamiquement des composants interactifs, tandis que TailwindCSS aide à styliser rapidement avec un design moderne et responsive.	1) React est optimisé pour des interfaces dynamiques, réactives avec des composants réutilisables. 2) Tailwind permet une mise en page rapide et efficace sans styles CSS externes lourds.
Authentification et Connexion	L'utilisateur doit pouvoir se connecter via une session sécurisée.	- Express -JWT	L'authentification se fera via **JSON Web Token (JWT)**, qui permet de gérer les sessions sans stocker d'informations côté serveur.	1) JWT est un standard sécurisé, utilisé par de nombreuses applications. 2) Il permet une authentification rapide sans dépendre d'une base de données côté session.
Gestion de l'état global	L'état des menus créés doit être géré efficacement sans trop recharger les composants React.	-Zustand (state management)	Zustand est une alternative plus légère que Redux, idéale pour gérer l'état des menus et la session utilisateur.	1) Plus simple et performant que Redux pour un projet de taille moyenne. 2) Permet d'éviter le prop drilling et facilite le partage de l'état entre composants.
Création d'une catégorie de menu	L'ajout d'une catégorie doit pouvoir se faire directement sur l'écran	-React-Modal	Cette bibliothèque React permet de créer simplement des modales performantes,	1) Compatible avec React, elle s'intègre facilement dans l'application.

	de création du menu depuis une modale.		accessibles avec un minimum de code.	2) Très utilisée dans la communauté, elle garantit un bon support et des mises à jour régulières.
Sauvegarde des menus créés	Les menus doivent être enregistrés et accessibles aux restaurateurs pour modification.	-PostgreSQL	PostgreSQL est une base relationnelle open source idéale pour organiser les menus et catégories.	1) PostgreSQL est robuste pour gérer des relations complexes entre restaurants, menus et catégories. 2) Il offre une meilleure gestion des transactions et de la cohérence des données que MongoDB.
Export des menus en PDF	Les restaurateurs doivent pouvoir générer un fichier PDF de leur menu avec un design personnalisé et une mise en page claire.	-React-pdf	**React-pdf** permet de générer des documents PDF sous forme de composants React avec un contrôle précis de la mise en page.	1) Intégration fluide avec React : permet d'utiliser des composants JSX, rendant la structure plus lisible et modulaire. 2) Personnalisation avancée : gestion des polices, styles CSS, intégration d'images **(ex. : logo du restaurant)** et mise en page flexible.
Publication sur Deliveroo et Instagram	Les menus doivent pouvoir être partagés sur Deliveroo et Instagram via leur API.	-API Graph de Meta pour Instagram -API Deliveroo	L'API Graph permet de publier des images de menus directement sur Instagram. Deliveroo nécessite une intégration plus avancée via leur API.	1) Automatisation du partage de menus, augmentant la visibilité des restaurants. 2) Ces APIs sont largement utilisées et documentées, facilitant l'intégration.
Hébergement éventuel du projet	L'application doit être accessible en ligne sans maintenance	-Vercel (front) + -Railway (back-end et BDD PostgreSQL)	Vercel héberge facilement les applications React, et Railway offre un	1) Vercel est optimisé pour React, avec un déploiement rapide et un CDN intégré.

	complexe.		hébergement simple pour Node.js et PostgreSQL.	2) Railway permet un hébergement PostgreSQL sans configuration complexe.
Gestion des utilisateurs	Les restaurateurs doivent pouvoir créer, modifier et gérer leur compte utilisateur.	-Express -PostgreSQL	Création d'une table **Users** en base de données pour stocker les informations utilisateurs (email, mot de passe **hashé**, préférences). Une API REST pour la gestion du profil.	1) PostgreSQL offre une gestion robuste des relations entre utilisateurs et menus. 2) Sécurisation des comptes avec **bcrypt** pour hasher les mots de passe.
Modification des informations utilisateur	Les restaurateurs doivent pouvoir modifier leurs emails, mots de passe et préférences.	-API REST (Express)	Une route API *PUT /users/:id* permettra aux utilisateurs de modifier leurs informations.	1) Expérience utilisateur améliorée avec des paramètres personnalisables. 2) Gestion sécurisée avec JWT pour éviter les accès non autorisés.
Suppression d'un compte	Un restaurateur doit pouvoir supprimer son compte et ses données associées.	-API REST (Express) + PostgreSQL	Une route API *DELETE /users/:id* supprimera l'utilisateur et ses menus de la base de données.	1) Conformité aux réglementations de protection des données (ex. RGPD). 2) Assure une gestion complète des données utilisateurs.
Gestion des rôles et permissions	Certains utilisateurs peuvent avoir des accès restreints.	-RBAC (Role-Based Access Control)	Définition des rôles (Admin, restaurateur). Chaque rôle aura des droits spécifiques (ex. : un restaurateur ne peut modifier que ses propres menus).	1) Sécurité renforcée avec des accès restreints selon les permissions. 2) Facilite la gestion des fonctionnalités accessibles par utilisateur.

II. Liens avec le back-end :

Langage serveur : **Node.js avec Express.js**

- **API REST** : **avec Express.js et PostgreSQL comprendra les endpoints suivants :**

Authentifications:

- POST /auth/login => connexion utilisateur.
- POST /auth/signup => création d'un compte utilisateur.
- POST /auth/logout => déconnexion utilisateur.

Gestion des menus :

- GET /menus => récupère les menus disponibles.
- GET /menus/ :id => récupère un menu spécifique.
- POST /menus => créer un nouveau menu.
- PUT /menus/:id => mettre à jour entièrement un menu.
- PATCH /menus/:id => mettre à jour partiellement un menu.
- DELETE /menus/:id => supprimer un menu.

Gestion des utilisateurs :

- GET /users/:id => récupérer les informations d'un utilisateur spécifique.
- PUT /users/:id => mettre à jour les informations utilisateur (email, mot de passe, préférences).
- DELETE /users/:id => supprime un utilisateur et ses données associées.
- *GET /users (optionnel, pour un Admin) => récupérer tous les utilisateurs (si gestion admin).*

- **Base de données** : **PostgreSQL**
  - Hébergée sur **Railway**.
  - Structure :
    - **Users** (id, email, password hash).
    - **Restaurants** (id, name, logo).
    - **Menus** (id, restaurant_id, name, description).
    - **Plats** (id, menu_id, name, price, category).

III. Préconisations concernant le domaine et l'hébergement :

Le site sera très probablement hébergé sous un **sous-domaine de Qwenta** (cf doc spécifications fonctionnelles).

- Exemple : menu.qwenta.com
- En cas de nom de domaine dédié, une alternative pourrait être menumaker.qwenta.com.

Adresses e-mail :

- **Email principal** pour le support : contact@qwenta.com
- **Email technique** (gestion des bugs et mises à jour) : support@qwenta.com
- **Email automatisé pour les notifications (noreply)** : noreply@qwenta.com

IV. Accessibilité :

Compatibilité navigateur :

Le site sera compatible avec :

- **Dernières versions de** :
  - **Google Chrome** (navigateur le plus utilisé).
  - **Mozilla Firefox**.
  - **Safari** (important pour les utilisateurs Apple).
- **Testé avec Microsoft Edge**, mais **Internet Explorer ne sera pas supporté**.

Types d'appareils :

- **Version desktop uniquement**.
- **Pas de version mobile prévue** pour le moment.
- **Optimisé pour les écrans entre 1024px et 1920px**.

Accessibilité :

Respect des normes WCAG 2.1 (Web Content Accessibility Guidelines) niveau AA.

- Navigation au clavier **(tab, enter, esc pour fermer les modales)**.
- Compatibilité avec **les lecteurs d'écrans**.
- **Contraste des couleurs** conforme aux normes d'accessibilité.

V. Recommandations en termes de sécurité :

Accès aux comptes utilisateurs :

- **Authentification sécurisée via JWT** (JSON Web Token).
- **Mot de passe hashé avec bcrypt** => protège les mots de passe en les chiffrant avant stockage.
- **Vérification en deux étapes (2FA) facultative** pour les comptes administrateurs.

Gestion des rôles :

- **Rôles utilisateurs** :
  - **Admin** : Gestion complète (utilisateurs, menus, branding).
  - **Restaurateur** : Création et modification des menus.
- Sécurisation avec **RBAC (Role-Based Access Control)**.

Protection des API :

- **Rate Limiting (Express Rate Limit)** => bloque les IP après X tentatives de connexion .
- **CORS activé** => autorise uniquement certaines requêtes du front-end.
- **HTTPS obligatoire** => chiffre les communications entre client et serveur.

Sécurité des plugins et dépendances :

- Vérification des mises à jour régulières avec **Dependabot (GitHub)**.
- Suppression des packages inutilisés pour éviter les vulnérabilités.

VI. Maintenance du site et futures mises à jour :

Grandes lignes du contrat de maintenance :

Le projet nécessitera **un suivi régulier** pour assurer son bon fonctionnement. Voici les principales **missions de maintenance** :

1. **Maintenance corrective**
   - Correction des **bugs critiques** sous 24h.
   - Surveillance des **logs d'erreurs** avec **Sentry**.

2. **Maintenance préventive**
   - Mise à jour des **dépendances NPM** tous les mois.
   - Mise à jour des **packages de sécurité** (ex. : Express, bcrypt).

3. **Maintenance évolutive**
   - Possibilité d'ajouter des **nouvelles fonctionnalités**.
   - Optimisation des performances si le nombre d'utilisateurs augmente.

Fréquence des mises à jour :

Type de mise à jour	Fréquence
Corrections de bugs	dès qu'un problème critique est détecté
Mises à jour de sécurité	1 fois par mois
Optimisation des performances	tous les 6 mois
Nouvelles fonctionnalités	selon les demandes des utilisateurs

Webgencia

Suivi des performances :

- Monitoring avec **Google Lighthouse** (performance + accessibilité).
- Suivi des erreurs en temps réel avec **Sentry**.
- Logs et erreurs serveur stockés avec **LogRocket ou Papertrail**.