

Spécifications techniques

Menu Maker Qwenta

Version	Auteur	Date	Approbation
1.0	Ousmane DIOP	01/09/ 2025	Sofiane

I. Choix technologiques.....	1
II. Liens avec le back-end.....	2
III. Préconisations concernant le domaine et l'hébergement.....	3
IV. Accessibilité.....	4
V. Recommandations en termes de sécurité.....	5
VI. Maintenance du site et futures mises à jour.....	6

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'un compte utilisateur	Connexion sécurisée, création de compte	JWT (JSON Web Token)	Utilisation d'un token généré côté serveur (Express.js) pour sécuriser les échanges et authentifier les utilisateurs.	1. Sécurité sans stocker les identifiants côté client. 2. Fonctionne parfaitement avec une SPA React.
Interface dynamique pour créer un menu	Interface fluide et modifiable sans rechargement	React.js	Framework front-end basé sur des composants réactifs, parfait pour créer des interfaces dynamiques.	1. Composants réutilisables pour les plats, les sections, etc. 2. Très bien documenté et rapide à implémenter.
Organisation et réorganisation des plats (drag & drop)	Glisser-déposer dans l'éditeur	React Beautiful DnD	Librairie React pour gérer simplement le drag & drop avec accessibilité intégrée.	1. Maintenue par Atlassian, utilisée dans Trello. 2. Intégration simple avec React.
Dashboard	Interface de gestion regroupant toutes les fonctionnalités (menus, compte, restaurant).	React.js + React Router	Le Dashboard est une Single Page Application (SPA) organisée en sections accessibles via le routage.	1. Centralise toutes les fonctionnalités pour l'utilisateur. 2. Simplifie la navigation et améliore l'expérience.

Sauvegarde des menus (plats + catégories)	Stockage flexible avec relations simples	PostgreSQL	Base de données relationnelle adaptée aux relations entre menus, plats et catégories.	1. Forte cohérence des données. 2. Large communauté et support cloud.
API pour communication front <-> back	Architecture découplée	Express.js + REST API	Création d'une API RESTful côté serveur avec Express.js pour gérer les routes.	1. Framework Node simple à utiliser. 2. Compatible avec JWT et PostgreSQL.
Partage d'un menu via un lien	Lien public	React Router	Génération d'un lien unique affichable sans authentification.	1. Simple à mettre en œuvre. 2. Permet la consultation sans modifier le menu.
Responsive design	Consultation sur smartphone et tablette	Flexbox + Media Queries	Mise en page adaptable aux écrans.	1. Standards CSS modernes. 2. Compatible avec tous navigateurs.
Accessibilité	Menu consultable par tous, même avec handicap	WCAG + ARIA	Respect des bonnes pratiques (contrastes, navigation clavier, ARIA)	1. Compatible lecteurs d'écran. 2. Meilleure note Lighthouse.
Sécurité	Protection des données utilisateur	Helmet.js + Bcrypt + CORS	Helmet sécurise les headers HTTP, Bcrypt hash les mots de passe, CORS limite les origines.	1. Standards de sécurité Web. 2. Conformes RGPD.

II. Liens avec le back-end

- Quel langage pour le serveur ?

Node.js

Langage JavaScript côté serveur. Il permet d'utiliser la même syntaxe que le front-end, favorisant la cohérence entre les deux couches.

- A-t-on besoin d'une API si oui laquelle

API REST

L'API permettra au front-end React de communiquer avec le back-end de manière structurée via des requêtes HTTP (GET, POST, PUT, DELETE).

- Base de données SQL

PostgreSQL

Base de données relationnelle (SQL), idéale pour gérer les relations entre plats, catégories et menus, avec forte cohérence et intégrité des données (plats, catégories, prix).

III. Préconisations concernant le domaine et l'hébergement

- Nom du domaine
menumaker.qwenta.fr
- Nom de l'hébergement

Front-end : Vercel ou Netlify

Back-end : Render ou Railway

Base de données : Google Cloud SQL (PostgreSQL managé) Adresses e-mail : support@qwenta.fr ou

- Adresse e-mail
Exemple professionnel :
 - support@qwenta.fr
 - contact@menu maker.qwenta.fr
 - Utilisation d'un service comme GandiMail, ZohoMail ou Google Workspace si besoin d'une solution email complète

IV. Accessibilité

- Compatibilité navigateur

Tests réalisés sur les navigateurs principaux :

- Google Chrome
- Mozilla Firefox
Des tests automatiques (axe DevTools, Lighthouse) et manuels seront réalisés pour vérifier la conformité aux standards WCAG 2.1 niveau AA.
- Safari
- Microsoft Edge

Le projet respecte les standards HTML5/CSS3, assurant un haut niveau de compatibilité inter-navigateurs.

- Types d'appareils

Le site est responsive :



- Ordinateurs de bureau
- Tablettes (iOS, Android)

- Smartphones


La conception responsive est assurée par Flexbox, Media Queries, et une structure en composants React réutilisables.

V. Recommandations en termes de sécurité

1. Authentification & gestion des comptes

- Utiliser JWT pour toutes les sessions (signé côté serveur).
- Stocker les tokens dans un cookie HttpOnly + Secure plutôt que dans le localStorage.
-  Mettre en place une rotation et révocation des tokens.
-  Valider l'adresse e-mail lors de l'inscription pour éviter les comptes frauduleux.

2. Protection des points de terminaison de l'API

- Toutes les routes `/api/*` protégées par un middleware :
 1. Vérifie la présence et la validité du JWT.
 2. Contrôle le rôle de l'utilisateur (admin, restaurateur, etc.).
 3. Valide les données entrantes pour se prémunir des injections (XSS, SQL Injection).
-  Mettre en place un rate limiting pour limiter les attaques par force brute.
- Appels vers des services externes
Les intégrations tierces (paiement, réseaux sociaux, etc.) utilisent des jetons API stockés dans des variables d'environnement côté serveur ; jamais en clair dans le code.
Les requêtes sortantes transportent ces jetons dans l'en-tête Authorization, garantissant que seuls les clients autorisés accèdent aux services externes.
- Transport & infrastructure
 - Chiffrement HTTPS forcé (certificat SSL Let's Encrypt fourni par Vercel/Render).
 - Sécurisation des en-têtes HTTP avec Helmet.js.

- Politique CORS restreinte aux domaines de production/staging.

VI. Maintenance du site et futures mises à jour

- Plan de maintenance continue
 - Mises à jour mensuelles des dépendances : npm audit, correctifs de sécurité, montées de version React/Node.
 - Sauvegardes automatiques quotidiennes de la base Google Cloud SQL (PostgreSQL managé) (7 jours de rétention).
 - Monitoring des performances et erreurs via Vercel Analytics + Sentry.
- Processus de déploiement
 - CI/CD GitHub Actions : tests → build → déploiement Vercel/Render à chaque merge sur main.
 - Environnement staging identique à la prod pour valider les nouvelles fonctionnalités avant mise en ligne.
- Évolutivité
 - Architecture modulaire (routes, contrôleurs, services) qui permet d'ajouter facilement de nouveaux endpoints ou micro-services.
 - PostgreSQL évolutif et adapté aux relations complexes.
- Documentation & transfert de compétence
 - Guide de contribution + README technique mis à jour à chaque sprint.
 - Commentaires de code systématiques sur les composants React et les routes Express pour faciliter la prise en main par d'autres développeurs.