Spécifications techniques

Menu-Maker by Qwenta

Version	Auteur	Date	Approbation
1.0	Thomas Sifferlé	08/09/25	Soufiane

I. Choix technologiques

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

Besoin	**Contraintes**	**Solution**	**Description de la solution**	**Justification (2 arguments)**
Ex. : *Création d'une catégorie de menu*	*Ex. :* *L'ajout d'une catégorie doit pouvoir se faire directement sur l'écran de création de menu depuis une modale.*	*Ex. :* *react-modal*	*Ex. :* *Cette librairie React permet de créer simplement des modales performantes, accessibles avec un minimum de code.*	*Ex. :* *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.*
Interface utilisateur dynamique	Interface moderne et réactive Compatible avec les maquettes Figma Gestion d'état complexe	React.js + Tailwind CSS	Framework JavaScript pour créer des interfaces utilisateur composants, couplé à un framework CSS utility-first pour un design cohérent et responsive.	1) React est imposé par les spécifications et offre une excellente réactivité pour les interfaces dynamiques 2) Tailwind CSS assure une cohérence visuelle et accélère le développement avec ses classes utilitaires.
Drag & Drop des éléments de menu	- Réorganisation intuitive des plats	@hello-pangea/dnd	Bibliothèque React spécialisée dans les	1) Bibliothèque mature et performante,

	- Expérience utilisateur fluide - Compatible React		interactions drag-and-drop avec gestion des états et animations.	parfaitement intégrée à l'écosystème React. 2) Gestion native des états de glisser-déposer et excellente accessibilité.
Gestion des formulaires	- Validation en temps réel - Performance optimisée - UX fluide	React Hook Form + Zod	React Hook Form pour la gestion performante des formulaires, Zod pour la validation des schémas de données.	1) React Hook Form minimise les re-rendus et améliore les performances. 2) Zod assure une validation TypeScript-native cohérente frontend/backend.
Gestion d'état global	- État partagé entre composants - Gestion du cache API - Synchronisation données	React Query + Zustand	React Query pour la gestion du cache et synchronisation serveur, Zustand pour l'état local global simple.	1) React Query optimise automatiquement les requêtes et gère le cache intelligemment. 2) Zustand offre une API simple sans boilerplate pour l'état global.
Authentification utilisateur	- Sécurité robuste - Gestion des sessions - Reset mot de passe	Supabase Auth	Service d'authentification complet avec gestion des sessions, OAuth, et sécurité intégrée.	1) Solution clé en main avec toutes les fonctionnalités d'auth (signup, login, reset, etc.).

				2) Sécurité enterprise-grade avec JWT et sessions automatiques.
Base de données	- Relationnel pour menus/plats - Performances optimales - Sécurité avancée	PostgreSQL via Supabase	Base de données relationnelle hébergée avec Row Level Security et API REST auto-générée.	1) PostgreSQL est performant pour les données relationnelles complexes (menus → catégories → plats). 2) Supabase génère automatiquement une API REST sécurisée avec RLS.
Stockage des fichiers	- Upload logos restaurants - Images de plats - Sécurité uploads	Supabase Storage	Service de stockage avec validation MIME, redimensionnement automatique et CDN intégré.	1) Intégration native avec l'authentification Supabase (sécurité par utilisateur). 2) Validation automatique des types de fichiers et optimisation images.
Génération PDF	- Export haute qualité - Personnalisation design - Performance	React-PDF	Bibliothèque React pour générer des PDFs programmatiquement avec contrôle total du layout.	1) Rendu cohérent avec l'interface React, même logique de composants. 2) Contrôle précis de la mise en page pour respecter le design des menus

Intégrations externes	- Paiement sécurisé - Partage Instagram - Export Deliveroo	Stripe + Instagram API + Deliveroo API	- Stripe pour les paiements d'impression - Instagram Basic Display API pour le partage - API Deliveroo Partner	1) Stripe est la référence pour les paiements en ligne avec excellente UX. 2) APIs officielles pour garantir la compatibilité long terme.
Hébergement frontend	- Déploiement automatique - Performance globale - Domaine custom	Vercel	Plateforme d'hébergement optimisée pour React avec CI/CD automatique et CDN mondial.	1) Intégration native avec React et GitHub pour déploiements automatiques. 2) CDN global et optimisations automatiques (lazy loading, compression).

II. Liens avec le back-end

Langage serveur

Supabase (PostgreSQL + Edge Functions)

- Backend-as-a-Service avec PostgreSQL comme base de données
- Edge Functions en TypeScript pour la logique métier complexe

- API REST auto-générée avec authentification intégrée

API choisie

API REST Supabase + APIs externes

- API REST automatique générée par Supabase à partir du schéma PostgreSQL
- APIs externes : Stripe (paiements), Instagram (partage), Deliveroo (export)
- Real-time subscriptions pour les mises à jour en temps réel

Base de données

PostgreSQL (via Supabase)

- Base relationnelle pour gérer les relations complexes (restaurants → menus → catégories → plats)
- Row Level Security (RLS) pour la sécurité au niveau utilisateur
- Performances optimisées avec indexation automatique

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

III. Préconisations concernant le domaine et l'hébergement

- **Nom de domaine**
- **menu.qwenta.fr** (sous-domaine Qwenta)
- Cohérence avec l'écosystème Qwenta
- SSL automatique via Vercel

- ## Hébergement

- **Frontend** : Vercel (CDN mondial, déploiements automatiques)
- **Backend/DB** : Supabase (infrastructure européenne RGPD-compliant)
- **CDN** : Intégré Vercel + Supabase pour optimisations globales

- ## Adresses e-mail

- **Notifications** : Via Supabase Auth (reset password, confirmations)
- **Support** : Intégration possible avec le système e-mail Qwenta

IV. Accessibilité

- **Compatibilité navigateur**
- **Chrome** : Dernières 2 versions
- **Safari** : Dernières 2 versions
- **Firefox** : Dernières 2 versions
- Support JavaScript ES2020+ requis

- ## Types d'appareils

- **Desktop uniquement** (selon spécifications)
- Résolution minimum : 1280×720px
- Navigation clavier complète

- Compatible lecteurs d'écran (ARIA labels)

V. Recommandations en termes de sécurité

- **Sécurité des accès**
- **Authentification** : JWT via Supabase Auth avec refresh tokens
- **Sessions** : Expiration automatique et renouvellement sécurisé
- **HTTPS** : Obligatoire sur toutes les communications

Sécurité des données

- **Validation** : Zod pour validation frontend/backend cohérente
- **RLS** : Row Level Security PostgreSQL pour isolation des données
- **Upload** : Validation MIME types et taille des fichiers

Protection RGPD

- **Consentement** : Gestion des cookies via Vercel Analytics
- **Données** : Hébergement EU (Supabase région européenne)
- **Suppression** : Outils Supabase pour right-to-be-forgotten

VI. Maintenance du site et futures mises à jour

- **Stratégie de maintenance**
- **Monitoring** : Supabase Dashboard pour surveillance temps réel
- **Logs** : Centralisés via Vercel Analytics et Supabase
- **Backups** : Automatiques via Supabase (point-in-time recovery)

- ### Mises à jour et évolutions

- **CI/CD** : GitHub Actions pour tests automatiques + déploiement Vercel
- **Versioning** : Git flow avec branches de feature et review code
- **Rollback** : Rollback instantané via Vercel Dashboard

- ### Montée en charge

- **Auto-scaling** : Automatique via Supabase et Vercel
- **Cache** : React Query côté client + CDN Vercel
- **Optimisations** : Code splitting React + lazy loading automatique