Spécifications techniques

[Nom du projet + nom du client]

Version	Auteur	Date	Approbation
1.0	Karelle Table	28/05/2025	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)

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.*

Landing non connecté	Afficher une page d'accueil publique pour les visiteurs non connectés	React Router	Un composant React affiché sur la route `/`, contenant une page pour comprendre l'utilité de cette application.	1) React Router permet de séparer la landing du reste de l'application 2) Il s'agit de la librairie la plus utilisée.
Page de login	L'utilisateur doit pouvoir se connecter avec son adresse mail et mot de passe via un modal	React -modal + Supabase Auth	Authentification sécurisée via lien mail	1) Librairie React qui permet de créer des modales 2)Permet un login rapide sans serveurs supplémentaires.

Catégorie de plat	Ajouter une catégorie (ex: entrées, plats, desserts) dans un menu, Ajout via modale sur l'écran de création	`react-modal` + Supabase	Modale React pour formulaire d'ajout, enregistrement dans la base Supabase	1) Librairie React qui permet de créer des modales 2Ajout via) supabase
Création de plat	Ajout complet (image, description, prix...) via modale	`React-modal` + Supabase Storage pour l'image	Formulaire contrôlé React pour envoyer les infos, image uploadée sur Supabase	1) Librairie React qui permet de créer des modales 2) Supabase gère le stockage sécurisé des images avec URL publique
Personnalisation du menu	Sélection typographie, couleur	Google Fonts API + CSS variables	Intégration dynamique de polices via Google Fonts, sélectionnées et appliquées avec CSS	1) Large choix de polices gratuites, 2.)Facilité de mise à jour sans surcharge
Exportation PDF	Un clic pour télécharger le fichier	react-pdf	Utiliser React-pdf pour générer le PDF directement à partir des composants React	1) Exportation coter client donc pas besoin d'utiliser un serveur 2) Simple à intégrer dans React

Impression de menu	Lien vers back-office, ouverture dans un nouvel onglet	Lien externe HTML	Simple lien ouvrant une nouvelle fenêtre	1) Ne nécessite pas de développement complexe 2)Permet de centraliser la gestion sur back-office Qwenta
Vue regroupant les menus précédents	Liste, modification, suppression, création depuis la même vue	Requête Supabase + react-router	Lecture des menus liés à l'utilisateur via Supabase, affichés dans une grille	1) Supabase permet une requête sécurisée par ID utilisateur 2) React-router pour créer une vue dédiée aux menu précédent
Informations légales	Accessible depuis toutes les pages, texte statique	React-modal statique	Modale simple avec contenu figé, toujours accessible	1) Pas de traitement dynamique requis 2) Intégration rapide avec React
Tarifs pour internautes	Lien vers une URL Qwenta, ouverture dans un nouvel onglet	Lien externe HTML	Lien ouvrant une page de tarification externe	1) Gestion centralisée par Qwenta 2) Aucune logique métier à intégrer

Exportation vers Deliveroo	Lien vers Deliveroo dans la section "Exportez et diffusez"	API Deliveroo/ React-router/ Supabase	À l'aide des données stockées dans Supabase (menus, catégories, items), un appel `POST` est effectué vers l'API Deliveroo pour exporter le menu du restaurateur. Si le compte Deliveroo n'est pas connecté, une authentification OAuth2 est déclenchée avant l'export.	1) Supabase permet d'accéder facilement aux données structurées du menu, ce qui facilite la génération automatique du payload requis par l'API Deliveroo. 2) L'intégration via OAuth2 garantit une connexion sécurisée au compte Deliveroo du restaurateur, tout en respectant les standards d'authentification actuels.

Partage sur Instagram	Génération image + redirection Instagram	Génération d'image (canvas) + redirection avec L'api Instagram	Création d'images au format Instagram + redirection	1) Utiliser React pour générer dynamiquement les visuels garantit une intégration fluide avec l'interface existante et permet un rendu personnalisable du menu. 2) La génération d'images localement évite les contraintes d'upload côté serveur et respecte les limitations de l'API Instagram en facilitant le partage manuel par l'utilisateur.
Déconnexion	Permettre à l'utilisateur de se déconnecter	Fonction de logout Supabase	Appel à Supabase pour supprimer la session active	1) Gestion centralisée de l'authentification 2) Standard sécurisé fourni par Supabase

Modification des infos du restaurateur	Permettre à l'utilisateur de modifier ses informations personnelles	Formulaire React + Supabase DB	Interface utilisateur + enregistrement en base	1) Simplicité de gestion côté utilisateur 2) Connexion directe aux tables utilisateurs Supabase
Dashboard	Vue regroupant les actions clés et articles du blog	Dashboard React + API Blog Qwenta	Affichage de raccourcis + récupération dynamique des articles	1) Améliore l'UX dès l'arrivée sur la page 2)Connexion possible avec CMS Qwenta via API
Branding restaurateur	Ajouter / modifier / supprimer logo et couleurs de base	Formulaires React + stockage Supabase (bucket + DB)	Interface graphique pour modification, stockage fichiers dans bucket Supabase et infos en base	1) Permet une personnalisation poussée du menu pour chaque restaurateur 2) Utilisation optimale des fonctionnalités de Supabase (stockage, DB)

I. Liens avec le back-end

- **Quel langage pour le serveur ?**

Le langage serveur choisi est **Node.js**, pour sa compatibilité native avec **Supabase** et sa parfaite intégration avec une application front-end développée en **React**. Cela permet :

- Une cohérence technologique (JavaScript/TypeScript sur toute la stack),
- Une prise en main rapide,
- La possibilité d'utiliser des middlewares et librairies modernes pour enrichir les

fonctionnalités.

- **A-t-on besoin d'une API ? Si oui laquelle ?**

Oui, plusieurs API sont nécessaires dans ce projet :

- **L'API Instagram** : pour permettre le partage de visuels du menu sur le compte du restaurateur.
- **L'API Deliveroo** : pour exporter automatiquement le menu vers la plateforme Deliveroo.
- **L'API REST de Supabase** : générée automatiquement à partir de la base PostgreSQL, elle permet de gérer toutes les opérations CRUD sur les données (menus, utilisateurs, etc.) sans développement backend spécifique.
  → Elle peut être enrichie par des fonctions "Edge" personnalisées pour effectuer des traitements avancés ou renforcer la sécurité si nécessaire.

- **Base de données choisie :**

Le choix de **PostgreSQL**, fourni et géré par Supabase, s'explique par :

- Sa robustesse et sa maturité dans les systèmes relationnels,
- Sa capacité à gérer les relations complexes entre utilisateurs, menus, catégories et plats,
- L'intégration directe avec Supabase Auth, Storage, et la gestion des règles de sécurité (Row Level Security).

I. Préconisations concernant le domaine et l'hébergement

- Nom du domaine: Le nom de domaine sera probablement un sous-domaine de Qwenta.
- Nom de l'hébergement:

- **Front-end** : hébergé sur **Vercel**, pour sa compatibilité directe avec React, ses déploiements CI/CD automatiques, et son temps de chargement optimisé.
- **Back-end** : géré par **Supabase**, qui intègre à la fois la base de données, l'authentification, le stockage et les éventuelles fonctions serveur.

- Adresses e-mail: nous pourrions utiliser des emails comme : contact@menu-marker.com, infos@menu-marker.com

I. Accessibilité

- Compatibilité navigateur: Chrome, Safari, Firefox (dernières versions)
- Types d'appareils: Desktop uniquement (mobile non prévu pour l'instant)

I. Recommandations en termes de sécurité

- **Authentification sécurisée** avec **Supabase Auth**, reposant sur des tokens JWT.
- **Accès restreint aux données** grâce aux **Row Level Security (RLS)**, qui permet de définir des permissions personnalisées sur chaque ligne de la base.
- **Plugins et dépendances** tiers strictement validés pour éviter toute vulnérabilité externe.
- **Actions sensibles** (comme l'export PDF ou la publication vers Deliveroo) protégées par des vérifications d'authentification.
- **Prévention XSS** via l'encodage systématique des entrées utilisateur et des contrôles rigoureux des formulaires.
- **CSRF non nécessaire** avec des appels API REST stateless et sécurisés par header (avec tokens).

I. Maintenance du site et futures mises à jour

• Maintenance régulière

Un **contrat de maintenance** est prévu pour assurer la stabilité et l'évolutivité de l'application. Il inclut :

- **Correction des bugs** : tout dysfonctionnement critique sera corrigé **sous 72 heures maximum**, afin de garantir une continuité de service pour les restaurateurs.

- **Mises à jour des dépendances** (React, Supabase, bibliothèques tierces) : celles-ci seront **effectuées tous les trimestres**, pour rester à jour sur les fonctionnalités, la sécurité et les performances.

- **Surveillance de l'infrastructure** (hébergée sur Vercel et Supabase) : des outils d'alerte seront mis en place pour anticiper les anomalies ou interruptions.

• Organisation des évolutions

- Les évolutions de l'application seront gérées **via un backlog produit partagé** avec l'équipe Qwenta. Cela permet :

- Une **priorisation claire des nouvelles fonctionnalités** selon les besoins des utilisateurs,

- Un **suivi structuré sur Notion**, en lien avec les maquettes Figma,

- Un processus de validation avant mise en production (test, QA, déploiement).

• Moyens de communication pour la maintenance

- **Support quotidien via Slack ou e-mail**, pour tout signalement ou besoin d'assistance.

- **Partage d'avancement et coordination via Notion et Figma**, utilisés comme outils collaboratifs entre les équipes technique, design et produit.