Webgencia

Spécifications techniques

[Nom du projet + nom du client]

Version	Auteur	Date	Approbation
1.0	[nom de l'auteur]	[date de réalisation du document]	[nom de la personne qui doit valider le document]

I. Choix technologiques 2

II. Liens avec le back-end 3

III. Préconisations concernant le domaine et l'hébergement 3

IV. Accessibilité 3

V. Recommandations en termes de sécurité 3

VI. Maintenance du site et futures mises à jour 4

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.*
Ajouter des plats à la catégorie sélectionnée	*Un formulaire d'ajout de plat doit s'ouvrir dans une modale, et l'utilisateur doit pouvoir ajouter plusieurs plats.*	*Formulaire avec gestion de l'état React*	*Utilisation d'un formulaire React pour capturer les informations du plat et les stocker dans l'état du composant pour une mise à jour en temps réel.*	*1) Réactif et conforme à la logique des composants React.* *2) Facilite l'ajout dynamique de plusieurs plats avec un rendu immédiat sur la prévisualisation à droite.*
Modifier et valider les éléments ajoutés	*Les modifications doivent être visibles immédiatement sur l'aperçu du menu à droite sans rechargement de page.*	*Zustand*	*Zustand est une bibliothèque de gestion d'état légère pour React, permettant des mises à jour fluides et réactives des éléments de l'interface.*	*1) Zustand est léger, performant, et adapté pour des projets nécessitant une gestion d'état localisé et réactif.* *2) Il permet une implémentation flexible sans introduire de complexité excessive.*
Personnaliser l'apparence du menu	*L'utilisateur doit pouvoir choisir une typographie et une couleur, et visualiser immédiatement les*	*Styled-Components*	*Cette approche permet de générer dynamiquement des styles basés sur les choix de l'utilisateur (couleurs, polices, etc.) et de les*	*1) Flexibilité de personnalisation en fonction des sélections utilisateurs.* *2) Solution propre et modulaire adaptée aux changements de*

	changements.		*appliquer en temps réel sur l'aperçu du menu.*	*style dynamiques.*
Exporter et diffuser le menu	*L'exportation doit être simple, en PDF, et permettre la diffusion sur des plateformes comme Deliveroo ou Instagram.*	*Html2pdf.js, API d'intégration Deliveroo/Instagram*	*Html2pdf.js permet de transformer du HTML en fichier PDF facilement. L'intégration via API pour Deliveroo et Instagram permet de publier directement les menus en ligne.*	*1) Solution éprouvée pour la génération de PDF à partir de HTML. 2) Les API permettent une automatisation du partage sur les plateformes extérieures.*
Imprimer un menu	*Le restaurateur doit pouvoir imprimer une version papier (physique) de son menu.*	*Utilisation de la fonctionnalité window.print en React*	*En ajoutant un bouton d'impression qui utilise la méthode JavaScript window.print, l'utilisateur peut imprimer le menu directement depuis son navigateur, avec un style adapté.*	*1) Méthode simple et native, fonctionnant dans tous les navigateurs. 2) Aucun besoin d'intégration complexe avec des bibliothèques externes, donc facile à maintenir.*

II. Liens avec le back-end

- **Quel langage pour le serveur ?** ***Ex. : NodeJS / PHP / Python, etc.***

Solution : *NodeJs + Express.js*

NodeJs est basé sur JavaScript, ce qui permet d'uniformiser le langage utilisé côté client et côté serveur. Cela simplifie le développement, surtout si l'équipe est déjà familière avec JavaScript pour la partie front-end.

*Le back-end de l'application utilisera le framework **Express.js** pour structurer et gérer les routes, les middlewares, et les requêtes HTTP. **Express.js** est un framework minimaliste, très populaire dans l'écosystème Node.js, qui permet de créer rapidement des API robustes et modulaires.*

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

Solution : Oui, une API REST

Pourquoi une API REST ? L'application nécessite un échange fluide de données entre le front-end (React) et le back-end, notamment pour la gestion des menus, la personnalisation, et la diffusion des contenus (PDF, Instagram, Deliveroo). Une API REST permet une communication claire et structurée entre le front-end et le back-end, avec des endpoints pour manipuler les données. C'est une solution standardisée, largement utilisée et bien adaptée pour ce type d'application.

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

Solution : SQL

Une base de données relationnelle (SQL) est plus adaptée pour ce projet, car nous avons déjà une idée claire du type de données à gérer. Les données sont structurées et définies à l'avance (nom des plats, catégories, style des menus, polices, etc.). SQL offre une organisation en tableaux, avec des relations bien établies entre ces données, ce qui simplifie la gestion des requêtes et des transactions.

III. Préconisations concernant le domaine et l'hébergement

- **Nom du domaine :** https://www.menu-maker.qwenta.com

Inclure le nom de l'outil dans le sous-domaine rend l'URL plus explicite et claire pour l'utilisateur. Cela aide aussi à renforcer l'identité du projet "menu-maker" tout en respectant l'architecture sous-domaine de Qwenta.

- Nom de l'hébergement : **Hostinger**

- Prix très compétitif avec des offres abordables pour les petites entreprises ou startups.
- Interface utilisateur intuitive et facile à prendre en main.
- Très bon temps de chargement des sites (performance).
- Certificat SSL gratuit et intégration Cloudflare.

Adapté pour des projets de taille moyenne, comme ton projet "menu-maker", avec une gestion simple et des besoins en performance sans exiger de configurations trop complexes.

- **Adresses e-mail : contact.menu-maker@qwenta.com**

L'adresse e-mail proposée suit une logique similaire avec le nom de domaine.

Cela permet de centraliser le support technique pour ce projet tout en le distinguant des autres services ou sous-domaines de Qwenta. En plus, cela donne une image plus professionnelle et organisée.

IV. Accessibilité

L’application devra être compatible avec les dernières versions des principaux navigateurs web utilisés par les utilisateurs.

Navigateurs compatibles :

- Chrome (denière version)
- Safari (denière version)
- Firefox (denière version)

L'application devra être accessible au minimum : navigable depuis le clavier, et lisible par un lecteur d'écran.
Pour garantir une bonne accessibilité aux utilisateurs de lecteurs d'écran, voici quelques bonnes pratiques à respecter :

- **Structure sémantique correcte** : Utiliser les balises **HTML** appropriées (titres avec **<h1>, <h2>**, etc.) permet aux lecteurs d'écran de comprendre et d'annoncer correctement la structure de la page.

- **Textes alternatifs pour les images** : Chaque image non décorative doit avoir une balise **alt** descriptive qui permet aux utilisateurs de savoir ce que représente l'image.
- **Navigation au clavier** : S'assurer que tous les éléments interactifs (boutons, liens, formulaires) sont accessibles et fonctionnels avec la navigation au clavier via la touche **Tab**. Cela inclut l'ajout de **aria-labels** et d'indicateurs visuels de focus.

V. Recommandations en termes de sécurité

1. Accès aux comptes :

❿ Authentification sécurisée :

Utiliser JWT : JSON Web Tokens pour gérer l'authentification des utilisateurs. Cela permet de créer des sessions sécurisées, d'éviter le vol de session et de mieux gérer les permissions d'accès.

Authentification à deux facteurs (2FA) : En plus du mot de passe, l'utilisateur doit fournir un second facteur (généralement un code envoyé par SMS ou via une application d'authentification). Cela ajoute une couche de sécurité, car même si un mot de passe est compromis, un attaquant aurait besoin de ce second facteur pour accéder au compte.

Mot de passe robuste : Imposer des règles pour la complexité des mots de passe (ex. : minimum 8 caractères, mélange de lettres, chiffres, symboles) diminue les risques d'attaques par force brute. Couplé à une politique de renouvellement régulier, cela améliore grandement la sécurité.

❿ Gestion des erreurs d'authentification :

En cas d'échec de connexion, ne jamais renvoyer un message indiquant si le compte existe ou non. Par exemple, éviter des messages comme "Compte non trouvé" ou "Mauvais mot de passe". Renvoyer un message générique pour ne pas donner d'indications à un éventuel attaquant.

➓ Verrouillage de compte :

Mettre en place un verrouillage temporaire après plusieurs tentatives de connexion échouées pour prévenir les attaques de force brute.

Par exemple, après 5 tentatives infructueuses, le compte est verrouillé pendant 30 minutes.

2. Plugins :

➓ Vérification des plugins tiers :

Limiter l'utilisation de plugins tiers à des sources fiables et régulièrement mises à jour. Chaque plugin utilisé doit provenir de fournisseurs reconnus et être maintenu activement pour éviter les failles de sécurité non corrigées.

➓ Permissions minimales :

Veiller à ce que chaque plugin utilisé ait des permissions strictement nécessaires. Par exemple, si un plugin n'a pas besoin d'accéder aux données des utilisateurs ou aux fichiers système, restreindre son accès pour limiter les risques d'attaque en cas de compromission.

3. Chiffrement des données :

➓ Chiffrement TLS (Transport Layer Security) :

Toute communication entre le client (navigateur) et le serveur doit être protégée par un chiffrement. Cela permet de garantir la confidentialité des données échangées (comme les mots de passe ou les informations personnelles).

➓ Protocole HTTPS :

S'assurer que le site utilise toujours le protocole **HTTPS** pour toutes les pages, même celles qui ne contiennent pas directement d'informations sensibles.

4. Prévention des attaques courantes :

➓ Protection DDOS :

Si QWENTA prévoit une montée en charge importante ou un public large, mettre en place des outils de prévention contre les attaques par déni de service distribué (DDoS), comme des solutions CDN avec protection DDoS intégrée.

➓ **Protection contre les injections SQL :**

Utiliser des requêtes préparées pour éviter les attaques d'injection SQL dans toutes les requêtes interagissant avec la base de données. Cela empêche un attaquant d'exécuter des commandes non prévues via des entrées utilisateur malveillantes.

VI. Maintenance du site et futures mises à jour

- **Service Level Agreement (SLA)**

1. **Durée de l'engagement :**
   - ➓ Le prestataire s'engage à maintenir le site Menu Maker pour une durée de **12 mois** à compter de la mise en ligne officielle du service.
   - ➓ Après cette période, une prolongation de ce **SLA** pourra être discutée et définie dans un avenant au contrat.

2. **Maintenance technique :**
   - ➓ Le prestataire s'engage à maintenir à jour les librairies et les dépendances techniques du site (frameworks, bibliothèques, API, etc.) afin d'assurer la sécurité et le bon fonctionnement du service.
   - ➓ Les failles de sécurité seront traitées en priorité et corrigées dans un délai de **48 heures** à partir de la notification de la faille.
   - ➓ Les plugins intégrés au site seront mis à jour tous les **3 mois**, ou dès qu'une mise à jour critique sera disponible.
   - ➓ Un suivi régulier des performances et de la disponibilité du site sera assuré. En cas d'incident majeur, le prestataire interviendra sous **24 heures**.

3. **Évolutions et nouvelles fonctionnalités :**
   - ➓ Toute demande de nouvelles fonctionnalités ou d'amélioration sera étudiée, et un devis sera soumis au client dans un délai de **5 jours ouvrés** après la réception de la demande.
   - ➓ Les nouvelles fonctionnalités seront facturées en fonction de leur complexité, avec une tarification estimative :
   - ➓ **Simple** (petites modifications, ajout d'éléments mineurs) : **300€ à 500€**
   - ➓ **Moyenne** (ajout de fonctionnalités telles que l'ajout de catégories personnalisées pour le menu) : **800€ à 1 500€**
   - ➓ **Complexe** (grandes refontes, ajout de modules importants) : **à partir de 2 000€**

Un devis détaillé sera fourni pour chaque nouvelle fonctionnalité avec une estimation du temps de développement et des délais de livraison.

4. **Éxclusions :**
   - Les incidents causés par des facteurs externes au service fourni (problèmes d'hébergement, infrastructure client, attaques DDoS, etc.) ne sont pas couverts par le présent SLA.
   - Les demandes de modifications en dehors du cadre initial du projet nécessitent une négociation séparée.