**Spécifications techniques**

Menu Maker – Qwenta

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Auteur** | **Date** | **Approbation** |
| 1.0 | Amir AFKIR | 07/05/2025 | Soufiane |

[I. Choix technologiques](#_409jm82qlgst) 2

[II. Liens avec le back-end](#_ex0md8td8283) 4

[III. Préconisations concernant le domaine et l’hébergement](#_a61v3uh5vs19) 4

[IV. Accessibilité](#_n56najuvb5ys) 5

[V. Recommandations en termes de sécurité](#_l80x4g9stizp) 6

[VI. Maintenance du site et futures mises à jour](#_vbbknpfyus5o) 7

## Choix technologiques

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

| ****Besoin**** | ****Contraintes**** | ****Solution**** | ****Description de la solution**** | ****Justification**** |
| --- | --- | --- | --- | --- |
| **Responsive Design** | Compatibilité avec tous les appareils | TailwindCSS + Media Queries | Utilisation de TailwindCSS avec media queries pour garantir une expérience optimale sur mobiles, tablettes et ordinateurs. | 1. Flexibilité rapide d'intégration. 2. Fortement utilisé dans l'écosystème front-end moderne. |
| **Créer un menu** | Interface claire, fluide et rapide | React Hook Form + Yup | Création d'un formulaire structuré et validation des champs directement côté client. | 1. Léger, simple à implémenter. 2. Très intégré dans l'écosystème React. |
| **Personnaliser le menu** | Changement de polices, couleurs et styles en temps réel | Styled Components + React-pdf + React-Dom | Application dynamique des styles via props React. Gestion de l'affichage PDF en temps réel avec React-pdf. | 1. Cohérence de style. 2. Dynamisme sans rechargement de la page. |
| **Enregistrer le branding (logo)** | Upload rapide et prévisualisation immédiate | React Dropzone + Cloudinary | Drag & drop pour upload d’images, hébergement optimisé sur Cloudinary. | 1. Solution scalable et fiable. 2. Intégration fluide avec React. |
| **Mise à jour dynamique du menu** | Modification en temps réel sans rechargement | React Context + State | Gestion de l'état global via Context API, permettant une mise à jour instantanée des composants sans rechargement. | 1. Communication rapide entre composants. 2. Approche moderne dans React. |
| **Gestion des utilisateurs** | Authentification sécurisée, différenciation des rôles | JWT + Bcrypt + MongoDB | JWT pour la gestion sécurisée des sessions. Bcrypt pour le hash des mots de passe. MongoDB pour stocker les utilisateurs et leurs rôles. | 1. Haut niveau de sécurité. 2. Architecture simple avec Node.js. |
| **Visualisation des menus précédents** | Historique de menus enregistrés | MongoDB + Mongoose | Stockage des anciens menus en base de données NoSQL, manipulation facilitée avec Mongoose. | 1. Adapté à des structures de données évolutives. 2. Accès rapide et souple. |
| **Affichage des tarifs** | Modification aisée des tarifs | React + API REST | Modification dynamique et récupération des tarifs via appel API. | 1. Facilement extensible. 2. Actualisation rapide sans rechargement complet. |
| **Export PDF** | Génération de PDF côté client, sans serveur | html2pdf.js | Conversion directe d'un composant HTML en fichier PDF et téléchargement automatique par l’utilisateur. | 1. Pas besoin de back-end pour cette fonctionnalité. 2. Solution légère et rapide. |
| **Partager sur Instagram** | Publier automatiquement sur un compte professionnel Instagram | html-to-image + Back-end Node.js + Instagram Graph API | Génération d'une image (html-to-image), envoi via back-end sécurisé à l'API officielle Instagram (Graph API) après authentification OAuth. | 1. Expérience pro, adaptée aux restaurateurs. 2. Automatisation via API officielle et sécurisée. |

## Liens avec le back-end

* **Quel langage pour le serveur ?**

*Le back-end sera développé en* ***Node.js****, un environnement JavaScript côté serveur, performant et bien adapté aux applications web modernes.*

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

*Oui, une* ***API REST*** *sera mise en place pour permettre la communication entre le front-end (interface utilisateur) et le back-end. Cette API permettra la récupération des menus disponibles, l’ajout, la modification et la suppression de menus, et la gestion des utilisateurs (authentification, droits, etc.).*

* **Base de données choisie** :   
  *MongoDB (NoSQL), adaptée aux structures de données flexibles et évolutives.*

## Préconisations concernant le domaine et l’hébergement

* **Nom du domaine** : *Menumaker.qwenta.fr*
* **Nom de l’hébergement** :*Vercel*
* **Adresses e-mail** :

- [*contact@Menumaker.qwenta.fr*](mailto:contact@Menumaker.fr)

- *support@Menumaker.qwenta.fr*

## Accessibilité

**• Compatibilité navigateur :**

Le site sera compatible avec les principaux navigateurs modernes :

Google Chrome, Mozilla Firefox, Microsoft Edge, Safari

Cela permet d'assurer un fonctionnement optimal quelle que soit la plateforme utilisée.

**• Types d’appareils :**

Le site sera optimisé pour une utilisation sur :

📱 Smartphones

💻 Ordinateurs (PC / Mac)

📱 Tablettes

L’approche choisie est desktop-first, avec une adaptation responsive pour tous les formats d’écran.

**• Accessibilité pour les malvoyants et utilisateurs avec besoins spécifiques :**

Intégration de textes alternatifs (attribut alt) sur toutes les images importantes

Respect des contrastes de couleurs suffisants selon WCAG

Taille de police adaptée et lisible (au moins 16px)

Focus visible et navigation clavier facilitée sur tous les éléments interactifs (boutons, liens, champs)

Utilisation des rôles ARIA pour faciliter l'interprétation du contenu par les lecteurs d’écran.

Vérification de l’expérience utilisateur via un lecteur d’écran (comme NVDA ou VoiceOver)

**• Outils de contrôle :**

Utilisation de WAVE Evaluation Tool pour identifier les problèmes d’accessibilité dans l’interface

Test avec l'extension Lighthouse de Chrome pour évaluer les performances et l’accessibilité

Relecture du site avec les critères WCAG 2.1 comme référence

## Recommandations en termes de sécurité

**🔐 Accès aux comptes**

• Mise en place d’un système d’authentification sécurisé via JWT (JSON Web Token), permettant une gestion efficace des sessions utilisateurs sans stocker d’informations sensibles côté client.

• Les mots de passe sont hachés avec bcrypt avant d’être stockés dans la base de données afin d’assurer leur confidentialité en cas de fuite.

• Application de règles de validation strictes lors de la création de compte : longueur minimale, complexité, interdiction des mots de passe trop courants.

• Possibilité d’ajouter une date de péremption pour inviter les utilisateurs à changer leur mot de passe régulièrement.

• Mise en place d’un système de rate limiting pour éviter les attaques par force brute sur les formulaires de connexion.

**🔧 Plugins et dépendances**

• Contrôle des plugins et packages npm utilisés :

◦ Utilisation de versions stables et maintenues

◦ Mise à jour régulière pour éviter les failles connues

◦ Suppression des dépendances non utilisées

• Vérification de sécurité via des outils comme npm audit et Snyk

**🌐 Communication sécurisée**

• Activation du protocole HTTPS sur l’ensemble du site pour sécuriser les échanges entre le client et le serveur.

• Utilisation de certificats SSL valides via Let’s Encrypt ou une autre autorité de certification.

• Intégration du middleware Helmet dans Express pour ajouter des en-têtes HTTP de sécurité (CSP, X-Content-Type-Options, etc.).

**🧼 Sécurité des données**

• Sanitisation des entrées utilisateurs pour éviter les injections (XSS, NoSQL injection, etc.)

• Les données envoyées à l'API sont validées côté serveur via des schémas stricts définis avec Joi, afin d’éviter toute injection ou traitement inattendu.

• Gestion des erreurs sans fuite d’information sensible dans les réponses du serveur

## **Maintenance du site et futures mises à jour**

Grandes lignes du contrat de maintenance :

* **Maintenance corrective**

Correction des anomalies ou dysfonctionnements constatés sur le site, signalés par le Client ou détectés par le Prestataire via des outils de surveillance. Les corrections seront apportées dans les meilleurs délais, en fonction du niveau de gravité de l’anomalie.

* **Maintenance évolutive**

Ajout de nouvelles fonctionnalités, amélioration de l’expérience utilisateur, et mise à jour du contenu (textes, images, modules) selon les besoins exprimés par le Client.

* **Maintenance préventive**

- Mise à jour régulière des dépendances techniques (ex : packages npm)

- Tests de sécurité récurrents

- Optimisation du code, nettoyage et veille technique

**Support technique**

Le Prestataire s’engage à fournir une assistance par e-mail, téléphone ou système de ticket, selon le niveau de contrat souscrit, avec des délais d’intervention adaptés à la criticité de la demande.

**Surveillance et alertes**

À la demande du client, un système de monitoring peut être mis en place pour suivre :

* Les erreurs applicatives
* La performance du site
* Sa disponibilité (uptime)

Outils possibles : Sentry, LogRocket, UptimeRobot

**Exemple de contrat de maintenance** :

- https://www.creerweb.fr/files/CONTRAT-DE-MAINTENANCE-INFORMATIQUE-MISE-A-JOUR-SITE-INTERNET--CREERWEB-2.pdf