Webgencia

Spécifications techniques

[MenuMaker+ Qwenta]

Version	Auteur	Date	Approbation
1.0	Sarciat Pierre-Henri	14/11/2025	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)
Authentification par e-mail (magic link)	Connexion sans mot de passe via validation de l'adresse e-mail	Front React + back Node.js/Express avec base de données, génération de token via magic link, gestion des modales via addEventListener et useEffect	Composant <LoginModal> avec formulaire email et submit, envoi POST pour création du token, validation côté serveur, ouverture <ConfirmModal> et démarrage de session	1) Évite le stockage et la gestion de mots de passe, simplifiant la connexion. 2) Séparation front/back pour maintenabilité, sécurité et fluidité.
Création d'une catégorie de plat	La modale s'ouvre depuis la page "Créer un menu", bouton actif uniquement si le champ est rempli, enregistrement via API POST, mise à jour dynamique	Composant React <CategoryModal> relié à Express/MongoDB	Formulaire contrôlé via useState, POST à l'API pour enregistrer, mise à jour dynamique du menu et liste déroulante	1) Amélioration de l'expérience utilisateur grâce à la mise à jour en temps réel. 2) Architecture claire front/back avec synchronisation fiable via useState.
Création d'un plat	Modale ouverte après sélection d'une catégorie, champs obligatoires, image optionnelle, mise à jour dynamique	Composant React <DishModal> avec useState, URL.createObjectURL pour prévisualisation	Formulaire contrôlé via useState, FormData envoyé en POST, enregistrement back avec Express/MongoDB , front met à jour le state du menu	1) UI fluide avec feedback immédiat. 2) Architecture front/back cohérente, synchronisation dynamique via state React.
Personnalisation du menu (typographie et couleurs)	Création d'un aperçu , mise à jour en temps réel, persistance des choix via base de données	Composant React, useState, API PUT/GET	Composant React <menuPreview>qui reçoit les données en props, useState pour mise à jour, envoi PUT pour sauvegarder, GET pour récupérer les styles enregistrés	1) Rendu interactif et instantané. 2) Persistance des préférences via API pour cohérence inter-sessions.

Exportation du menu en PDF	Bouton de génération PDF , téléchargement automatique	Composant React “Exporter en PDF” avec handleExport(), jsPDF	Récupération des données du menu, génération PDF conforme au rendu, téléchargement automatique	1) Intégration simple et réactive via React. 2) jsPDF assure une génération fiable côté client, rapide et performante.
Impression du menu	Création de l’ encart, lien vers back-office Qwenta,	Ouverture du back-office dans un nouvel onglet, liste dynamique des menus.	Composant React avec lien target="_blank" et vue “Mes menus” récupérant les menus via API GET	1) target="_blank" pour la fluidité de la navigation. 2) Vue React dynamique avec API garantit synchronisation front/back.
Gestion des menus existants	Accès via “Mes menus”, affichage complet avec modification, suppression, création	Liste des menus, actions intégrées pour modifier, supprimer ou créer, mise à jour dynamique après chaque opération	Composant React “MesMenusView” avec API GET/PUT/DELETE/POST	1) Centralisation des actions simplifiant l’expérience. 2) Composants React assurent synchronisation et mise à jour cohérente.
Mentions légales	Accessible sur toutes les pages, modale statique, mention “Tous droits réservés”	Composant React pour modale avec state local	Ouverture/fermeture via useState, texte statique intégré, mention visible sur toutes les pages	Contrôle simple et fiable de l’affichage de la modale.
Tarifs	Ouverture dans un nouvel onglet, style conforme maquette	Composant React dans barre de navigation	Lien HTML target="_blank" pointant vers URL stockée en variable d’environnement	Variable d’environnement facilite la maintenance.
Diffusion sur Deliveroo	Lien de redirection vers Deliveroo dans un nouvel onglet	Composant React “Diffuser sur Deliveroo”	Lien HTML target="_blank", URL stockée en variable d’environnement, aucune logique back nécessaire	1) Redirection simple et fiable. 2) URL configurable facilement via variable d’environnement.
Partage sur Instagram	Création d’ un encart , génération image carrée du menu, redirec-	Composant React “Partager sur Instagram”, html-to-image	Capture du menu en PNG/JPG/WebP, téléchargement local via <a down-	1) Génération locale rapide et fidèle. 2) Redirection simple pour expérience intuitive.

	tion vers Instagram		load>, ouverture Instagram target="_blank"	
Déconnexion	Bouton de déconnexion sur toutes les pages connectées, suppression session front/back, redirection vers page d' accueil	Composant React "Déconnexion" avec suppression token JWT	Suppression token localStorage, mise à jour state global setUser(null), redirection via navigate('/')	1) Empêche accès aux pages protégées. 2) Expérience utilisateur claire avec redirection immédiate.
Dashboard	Centralisation sur une même page des fonctionnalités et des derniers articles du blog	Composant React <Dashboard> avec react-router et fetch	Trois encarts interactifs, section "Pour aller plus loin" récupérant articles via fetch et state local.	1) Navigation fluide centralisant toutes les actions. 2) Rendu réactif et redirections fiables via React.
Branding restaurateur	Ajout/modification/suppression logo et couleurs	Composant React "BrandingModal" avec input file et Sketch-Picker	Aperçu en temps réel, envoi au serveur via API, stockage dans la base, support PNG/JPG/WebP	1) Aperçu dynamique qui améliore l'expérience utilisateur. 2) Input file et color-picker standardisent l'upload et la sélection de couleurs pour une meilleure cohérence graphique.

I. Liens avec le back-end

- Quel langage pour le serveur ?

Le serveur utilise Node.js avec le framework Express pour gérer les routes, la logique métier et la communication avec la base de données.

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

Oui, une API REST est indispensable.
Elle permet au front-end React de communiquer avec le back-end pour :

- gérer l'authentification par magic link (ex : POST /auth/login, GET /auth/validate-token),
- créer / modifier / supprimer catégories**,** plats**,** menus (ex : POST /categories, POST /dishes, GET /menus, PUT /menus/:id, etc.),
- enregistrer les styles du menu (typographie, couleurs) (ex : PUT /style, GET /style),
- gérer l'upload du **logo** (ex : POST /upload/logo),
- gérer les adresses e-mail de l'utilisateur (ex : GET /user/emails, PUT /user/emails),

- Base de données choisie :

La base utilisée est MongoDB (NoSQL)**.**

Elle est adaptée car :

- les menus, plats et catégories ont des structures flexibles,
- elle permet une évolution simple du schéma (ajout de champs, de préférences de style, etc.),
- elle s'intègre très bien avec Node.js via Mongoose**.**

-

II. Préconisations concernant le domaine et l'hébergement

- Nom du domaine.

  Opter pour un nom de domaine clair et professionnel, par exemple : menumanker.fr

- Nom de l'hébergement.

S’appuyer sur une solution d’hébergement moderne capable de gérer la montée en charge et les déploiements continus : OVH Cloud, Scale-way, Vercel.

- Adresses e-mail.

Mettre en place des adresses professionnelles basées sur le domaine : contact@menumaker.fr (support et communication), no-reply@menumaker.fr (envoi des magic links), admin@menumaker.fr (gestion interne)

III. Accessibilité

- Compatibilité navigateur :

Le service doit fonctionner sur tous les navigateurs modernes : **Chrome, Firefox, Safari.**

- Types d’appareils : ordinateurs, tablettes.

IV. Recommandations en termes de sécurité

- Accès aux comptes, plugins… :

  La sécurité du site repose sur une gestion stricte des accès (permissions limitées, tokens sécurisés), la mise à jour régulière des plu-

gins et dépendances, ainsi que la protection des données sensibles via des variables d'environnement.

V. Maintenance du site et futures mises à jour

- Grandes lignes du contrat de maintenance :

  La maintenance inclut les mises à jour techniques et de sécurité, la correction des bugs, la surveillance du serveur, les sauvegardes planifiées, l'ajout d'améliorations évolutives et un support technique continu afin d'assurer la stabilité, la performance et la pérennité du site.