Spécifications techniques

[Menu Maker + Qwenta]

Version	Auteur	Date	Approbation
1.0	Emeline	09/09/2024	Soufiane

I. Choix technologiques

1.1.État des lieux des besoins fonctionnels et de leurs solutions techniques :

Besoin	**Contraintes**	**Solution**	**Description de la solution**	**Justification**
Landing non connectée	*L'internaute doit comprendre l'utilité de Menu Maker*	*React (static page)*	*Utilisation de composants React pour créer une landing page interactive mais légère*	*1) Composants réutilisables avec React 2) Maintient la rapidité et l'optimisation des performances*
Connexion / création de compte	*L'utilisateur doit pouvoir se connecter ou créer un compte via une modale*	*Auth0 (API Auth0 pour la gestion des utilisateurs) +MongoDB*	*Auth0 gère l'authentification via son API pour la création de comptes et la connexion Stockage des informations des utilisateurs via MongoDB et gérées par Auth0*	*1) Auth0 permet une gestion simplifiée des utilisateurs 2) Grande sécurité des accès utilisateur*
Déconnexion	*Le restaurateur doit pouvoir se déconnecter à tout*	*Auth0 (logout API)*	*Utilisation des fonctionnalités d'Auth0 pour gérer la*	*1) Auth0 offre une gestion sécurisée des sessions utilisateur 2)*

	moment		*déconnexion sécurisée*	*Permet de gérer les accès multi-dispositifs*
Dashboard utilisateur	*Le restaurateur doit accéder à une vue d'ensemble de ses activités (menus, diffusions, impressions, articles du blog)*	*React (composants de dashboard)* *API Node.js (GET) + MongoDB*	*Interface sous forme de tableau de bord affichant les différentes options (création, diffusion, impression de menus) ainsi que des informations contextuelles (articles du blog)* *L'API gère les requêtes GET pour récupérer les informations liées au restaurateur*	*1) Dashboard simple et clair pour une vue d'ensemble rapide 2) Utilisation des composants React pour une expérience utilisateur optimale* *3) Utilisation de MongoDB pour centraliser les données* *4) Réactivité assurée par Node.js et les appels API*
Création/modification de menu	*Le restaurateur doit pouvoir créer ou modifier son menu dans une interface intuitive avec des modales qui permettent l'ajout de catégories ou de plats directement sur l'écran de création de menu.*	*React avec gestion des états et modales* *API Node.js + MongoDB*	*Utilisation de composants dynamiques pour gérer les modales et les formulaires d'ajout/édition de menus.* *L'API gère les requêtes pour créer/modifier des menus, avec des opérations CRUD sur*	*1) Flexibilité pour ajouter des éléments via modales 2) Réactivité et performance de React* *3) MongoDB NoSQL est flexible pour stocker des données structurées ou semi-structurées (menus, plats) 4) Node.js permet une gestion efficace des requêtes*

	Il peut également y personnaliser le style de son menu.		*MongoDB*	*asynchrones*
Stockage d'images	*Le restaurateur doit pouvoir associer une image à chaque plat*	*Cloudinary (API de stockage) + MongoDB*	*Utilisation de l'API Cloudinary pour gérer le stockage et la gestion des images , avec stockage de l'URL dans MongoDB*	*1) Optimisation des images pour le web via Cloudinary 2) Capacité de gestion de grande quantité d'images avec efficacité 3) MongoDB permet de lier facilement l'image au plat avec une référence URL*
Exportation PDF du menu	*Le restaurateur doit pouvoir exporter son menu en PDF*	*Librairie HTML2PDF (côté front-end)*	*Génération de fichiers PDF directement à partir du front-end*	*1) Permet l'export immédiat du menu en PDF 2) Librairie largement utilisée et maintenue*
Commander l'impression de menu	*Le restaurateur doit pouvoir commander une impression via Qwenta*	*Redirection via un lien externe*	*Un lien est fourni vers le back-office de Qwenta pour l'impression*	*1) Simplifie la commande d'impression 2) Externalise le processus complexe d'impression à la maison mère spécialisée*
Partage sur Instagram	*Le restaurateur doit pouvoir partager son*	*API Instagram et génération d'images carrées*	*API Instagram permet la publication, et le format carré des*	*1) Simplifie la diffusion du menu sur les réseaux sociaux 2) Répond aux*

	menu sur Instagram	*côté front-end*	*images est généré côté front*	*exigences visuelles d'Instagram*
Export vers Deliveroo	*Le restaurateur doit pouvoir exporter son menu sur Deliveroo*	*API Deliveroo*	*Connexion directe via l'API Deliveroo*	*1) Facilite l'intégration avec une plateforme externe populaire 2) Flux simple pour les restaurateurs*
Page des menus	*Le restaurateur doit pouvoir voir, modifier ou supprimer des menus précédemment créés*	*React avec gestion des états* *API Node.js + MongoDB (CRUD)*	*Utilisation de composants React pour afficher les menus précédemment créés avec des options de modification ou suppression* *Les menus créés sont récupérés via des requêtes GET, PUT et DELETE dans MongoDB*	*1) Gestion simple des menus via React 2) Mise à jour dynamique des menus sans rechargement complet de la page 3) MongoDB offre une flexibilité pour modifier les menus 4) Les API Node.js assurent une interaction fluide avec le front-end*
Infos utilisateurs	*Le restaurateur doit pouvoir consulter et modifier ses informations personnelles*	*React avec gestion des états* *API Node.js + MongoDB pour les données*	*Page dédiée sur laquelle l'API permet la récupération et la modification des informations utilisateurs via*	*1) React permet de gérer dynamiquement les modifications des informations 2) MongoDB stocke efficacement les*

		utilisateur	*MongoDB (e-mails ou autres informations)*	*informations utilisateur 3) L'API permet des modifications simples des données*
Pages statiques (Mentions légales, tarifs)	*L'internaute doit pouvoir accéder aux pages statiques depuis les différentes sections du site*	*React (static pages)*	*Utilisation de composants React pour afficher des pages statiques telles que "Mentions légales" ou "Tarifs"*	*1) Réduction de la complexité côté front-end 2) Pages faciles à maintenir et à mettre à jour*

1.2.Détails supplémentaires pour la Gestion des appels API avec modification de base de données :

Connexion / création de compte (Auth0) :

- Appel API Auth0 : Gestion des requêtes POST pour la création d'un compte (via l'email) et des requêtes POST/GET pour la connexion.
- Base de données MongoDB : Aucune modification directe, les informations d'authentification sont gérées par Auth0.

Création / modification de menu :

- Appel API Node.js :
  - POST : Création d'un nouveau menu.
  - PUT : Modification d'un menu existant.
  - GET : Récupération d'un menu pour le restaurateur.
  - DELETE : Suppression d'un menu existant.
- Base de données MongoDB :
  - Sauvegarde des menus, des plats associés (nom, description, image via URL Cloudinary) sous forme de

documents.
    - Mise à jour ou suppression de ces documents en fonction des requêtes API.

Stockage d'images :

- Appel API Cloudinary :
    - POST : Envoi d'images depuis le front-end pour stockage sur Cloudinary.
    - GET : Récupération des images depuis Cloudinary via une URL.
- Base de données MongoDB :
    - de l'image est stockée dans le document correspondant au plat dans MongoDB.

Vue des menus précédents :

- Appel API Node.js :
    - GET : Récupération des menus précédemment créés.
    - PUT/DELETE : Modification ou suppression des menus existants.
- Base de données MongoDB : Chaque menu est stocké en tant que document dans MongoDB, avec une relation entre l'utilisateur et ses menus.

Infos utilisateurs :

- Appel API Node.js :
    - GET : Récupération des informations utilisateurs (email, nom, etc.).
    - PUT : Modification des informations (ex : changement d'adresse e-mail).
- Base de données MongoDB : Stockage des informations utilisateurs avec la possibilité de les mettre à jour via l'API.

Dashboard utilisateur :

- Appel API Node.js :

    - GET : Récupération des données relatives au restaurateur (ex : liste des menus créés, diffusion, commandes d'impression).
- Base de données MongoDB : Les informations sont stockées sous forme de documents dans MongoDB et récupérées via l'API.

1.3.Architecture :

Front-end : React
Back-end : Node.js
Base de données : MongoDB (NoSQL)
Authentification : Auth0
Stockage des images : Cloudinary
Hébergement : OVH

II. Liens avec le back-end

- Langage pour le serveur : Node.js (JavaScript)
- API : API RESTful avec Express.js pour gérer les requêtes du front-end vers le back-end
- Base de données choisie : MongoDB (NoSQL)
    - Utilisation de MongoDB pour la gestion des données des utilisateurs et des menus.
    - Sauvegarde des plats, catégories, et autres informations liées à chaque utilisateur.

III. Préconisations concernant le domaine et l'hébergement

- Nom du domaine :
    - Possibilité 1 : **créer un sous domaine à partir du nom de domaine de Qwenta** (normalement cette possibilité est celle choisie)
    - Possibilité 2 : Créer un nouveau nom de domaine, possibilité : menu-maker-byqwenta.com
- Hébergement : OVH
- Adresses e-mail :
    - Pour possibilité 1 : contact-menu-maker@qwenta.com
    - Pour possibilité 2 : contact@ menu-maker-byqwenta.com

IV. Accessibilité

- Compatibilité navigateur :
    - Le site doit être compatible avec les navigateurs récents tels que Chrome, Firefox, Safari et Edge.
    - Utilisation de pollyfills pour assurer la compatibilité avec les versions plus anciennes des navigateurs majeurs.
- Types d'appareils :
    - L'application doit être accessible sur ordinateurs, tablettes, et smartphones.
    - Interface adaptative pour garantir une expérience fluide sur tous les types d'appareils (responsive design).

V. Recommandations en termes de sécurité

- Accès aux comptes : Utilisation d'Auth0 pour l'authentification sécurisée.
    - Auth0 assure la gestion des identifiants, la protection contre les attaques par force brute, et les violations de données.

    - Authentification par e-mail pour éviter l’utilisation de mots de passe faibles.
- Sécurité des plugins :
    - Utilisation de librairies et plugins régulièrement maintenus et mis à jour pour éviter les vulnérabilités.
    - Application des correctifs de sécurité immédiatement après leur publication.
- Gestion des données sensibles :
    - Chiffrement des données sensibles telles que les informations personnelles des restaurateurs.
    - Utilisation du protocole HTTPS pour toutes les communications entre le front-end et le back-end.

VI. Maintenance du site et futures mises à jour

- Suivi des incidents techniques et résolutions dans un délai de 24h pour les problèmes critiques.
- Mises à jour régulières des dépendances du front-end et du back-end pour s'assurer de la compatibilité avec les dernières versions des navigateurs et des systèmes.
- Contrôle de la sécurité des données des utilisateurs avec une surveillance proactive des failles de sécurité.
- Ajout de nouvelles fonctionnalités et support pour les évolutions du projet (comme l'intégration avec d'autres plateformes externes, si nécessaire).