Webgencia

Spécifications techniques

[Menu Maker by Qwenta]

Version	Auteur	Date	Approbation
1.0	Manon Ruault, Webgencia	8/10/25	Soufiane, Webgencia

Webgencia

I. Choix technologiques

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

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Cadre technique global				
Front-end	Performance d'affichage en temps réel (Menu Maker). Accessibilité (conformité WCAG) obligatoire. Petite équipe de 2 développeurs (dont un Front-end).	Framework React.js	React est une librairie JavaScript déclarative et basée sur les composants réutilisables. Elle sera utilisée pour : gérer l'état de l'application (données utilisateur et du menu), assurer le rendu conditionnel des vues (pages publiques vs. Dashboards sécurisés), optimiser la mise à jour de l'interface grâce au DOM Virtuel.	1) React est la librairie Front-end la plus utilisée sur le marché, offrant le meilleur écosystème de librairies tierces (routage, modales, etc.) 2) Le DOM Virtuel garantit des mises à jour fluides et rapides de l'interface, ce qui est essentiel pour une expérience utilisateur de "temps réel" lors de la personnalisation du menu (couleurs, logo, etc.).

Back-end	Taille de l'équipe : 1 développeur back-end + 1 développeur front-end	Environnement Node.js	Utiliser pour développer l'API REST qui exposera les endpoints sécurisés et centralise la logique métier.	1) Scalabilité 2) Simplicité de déploiement : permet au développeur Back-end de se concentrer sur le développement et non le déploiement 3) Permet aux deux développeurs d'intervenir en soutien sur l'autre partie (back ou front)
UI				
Création d'une page d'accueil publique	La page doit présenter clairement la valeur ajoutée de l'outil. Elle doit contenir un appel à l'action connexion. Respect de la charte graphique du site.	Utilisation d'un système de routage : React Router	Définir une route spécifique (/accueil) pour cette page qui est publique et accessible sans authentification. Ajouter les composants banner et footer ainsi que les 3 composants distincts de la page (présentation du concept, personnaliser son menu, étape par étape. Banner : doit intégrer le CTA connexion/inscription.	1) Assure un accès rapide à la page pour les nouveaux utilisateurs et les moteurs de recherche 2)Sépare la logique d'interface publique de l'application sécurisée, nécessitant une connexion
Création du dashboard pour	Accès protégé, uniquement après connexion.	Routes Protégées (Front-end) etqu'es Middleware	Le Front-end (React Router) vérifie la présence et la validité du JWT avant d'autoriser l'affichage du composant Dashboard. Le Back-end utilise un	1) Assure la sécurité et l'intégrité des données en empêchant l'accès non autorisé au contenu utilisateur.

l'utilisateur connecté		d'Authentification (Back-end)	middleware pour sécuriser l'API et valider ce token à chaque requête.	2) Garantit que le token est valide et non expiré avant l'accès à l'application
	Composant permettant d'afficher différents liens : créer un menu, diffuser un menu ; importer un menu.	Composants atomiques réutilisables (ex: Card/Tile Component)	Créer un composant générique pour les cartes du Dashboard qui gère le style et la navigation, puis instancier chaque lien comme une variation de cette carte.	1) Améliore la maintenabilité du code en utilisant le principe DRY, facilitant les modifications de style futures. 2) Assure une cohérence visuelle parfaite pour tous les éléments d'action du Dashboard (UX).
	Rubrique pour aller plus loin : le contenu doit être facile à mettre à jour par Qwenta	Utilisation d'une route API Node.js + requête SQL pour l'affichage dynamique dans un composant React	Une route backend /api/articles/latest interroge la base de données SQL pour récupérer les 3 articles les plus récents. Le frontend React appelle cette API via fetch() et met à jour automatiquement l'affichage sans intervention manuelle.	1) Automatisation : la rubrique se met à jour dès qu'un nouvel article est ajouté en base, sans modification du code. 2) Maintenabilité : séparation claire des responsabilités entre backend (données) et frontend (affichage), ce qui facilite les évolutions futures.
Création d'un menu via une fenêtre modale (depuis le tableau de bord)	La navigation à travers les 3 étapes de la création sont séquentielles.	Création d'un composant parent en React gérant l'étape courante. Utilisation d'une libraire de composants React pour la modales (react-modal)	Structure et navigation : Gestion de l'état avec useState pour gérer l'index de l'étape actuelle. Le contenu doit s'ajouter à l'étape courante. Le bouton "Suivant" incrémente l'index de l'étape, et n'est possible qu'avec une validation minimale. Cette librairie permet de créer simplement des modales performantes.	1) L'approche séquentielle (Wizard) rend la création d'un contenu complexe plus gérable et moins intimidante pour l'utilisateur. 2) Isoler la logique de navigation dans un composant Wizard permet de réutiliser ce modèle pour d'autres formulaires multi-étapes.

Webgencia

Création (et réutilisation) d'une catégorie de plat à l'intérieur de l'étape 1 du Wizard	L'ajout d'une catégorie doit pouvoir se faire sur l'écran de création de menu. Une catégorie déjà créée doit pouvoir être sélectionnée.	Création d'un composant "Catégorie" qui gère la collection des objets de catégorie via l'état du Wizard. Le composant doit intégrer un mécanisme de sélection basé sur les données d'API des catégories existantes.	Récupérer la liste des catégories créés ultérieurement. Seul le nom de la catégorie est réutilisé et non les plats associés. Développer le composant contenant le bouton d'ajout d'une nouvelle catégorie. La nouvelle catégorie ou celle sélectionnée est enregistrée comme faisant partie du menu lors de la sauvegarde finale (POST /api/menus).	1) Réduit l'effort de saisie pour l'utilisateur en automatisant la structure de base du menu. 2) Assure une uniformité dans la terminologie des menus de l'utilisateur.
Création d'un plat	Après avoir choisi une catégorie seulement, au clic "+plat" une nouvelle fenêtre modale s'ouvre. Le prix doit être stocké comme un nombre décimal précis. Le plat est lié à la catégorie.	Utilisation de React Hook Form pour la gestion des formulaires et de Yup pour la validation du schéma.	Le formulaire pour ajouter un plat utilise React Hook Form pour gérer les valeurs et les erreurs côté client, et Yup pour la validation du schéma (nom, prix, catégorie). Lorsque le formulaire est soumis avec succès (validation Yup), les données du plat sont insérées dans l'état Front-end temporaire du menu en cours de création. L'état est mis à jour et la prévisualisation du menu est rafraîchi immédiatement.	1) Améliore l'UX en donnant un feedback instantané sur les erreurs de saisie (grâce à Yup) et évite les appels API inutiles avant la sauvegarde complète du menu. 2) Assure que les champs obligatoires et le format du prix sont corrects avant même d'entrer dans l'état de création.

Personnaliser l'apparence du menu	L'utilisateur doit pouvoir visualiser en temps réel les changements, sélectionner une typographie parmis 3 proposées et choisir une couleur. Les polices sélectionnables doivent être prédéfinies et disponibles.	Utilisation de UseSTate + react-color pour le sélecteur de couleur. Les polices sont gérées par des classes CSS dynamiques.	Le formulaire permet la sélection de la typographie (via un <select>) et de la couleur (via le ChromePicker de react-color). Les sélections mettent à jour l'état temporaire du menu. Le composant de révisualisation écoute cet état. Les changements sont appliqués instantanément via des styles ou des classes CSS conditionnelles, sans rechargement de la page.	1) L'utilisateur voit immédiatement les changements de texte, de police et de couleur, ce qui améliore l'expérience et réduit les erreurs. 2) L'utilisation de React et d'une librairie dédiée (react-color) permet de séparer clairement la logique et la présentation, facilitant les évolutions futures.
Exportation du menu en PDF	Le PDF doit conserver la mise en page choisie par l'utilisateur. Le processus d'exportation doit être rapide pour ne pas bloquer l'interface. Le rendu doit être de haute qualité.	Utilisation de html2canvas + jsPDF pour capturer le menu et générer un PDF	Le contenu du menu affiché sur le site est capturé côté client en tant qu'image grâce à html2canvas. Pour assurer un rendu de haute qualité, l'option de mise à l'échelle (scale) de html2canvas sera augmentée pour améliorer la densité de pixels de l'image. Cette image est ensuite utilisée par jsPDF pour générer un fichier PDF téléchargeable par l'utilisateur.	1) Permet la fidélité visuelle du rendu grâce à la capture directe du contenu Front-end. 2) Solution rapide et côté client, sans dépendance serveur
Partager le menu sur Instagram	L'encart "partager sur Instagram" doit s'afficher dans la catégorie "Exportez et diffusez".	Bouton "Partager sur Instagram" : authentification au compte Instagram Business, image générées et envoyées aux	Intégration du Login Facebook/Meta pour obtenir un Token d'Accès (Access Token). Utilisation du endpoint /media de l'API Graph d'Instagram pour téléverser	1) Le restaurateur peut partager rapidement le menu sous un format adapté à Instagram sans manipulations complexes. 2) Les images générées conservent la typographie, couleurs et mise en page du

	Au clic, des images du menu au format carré sont générées. Le restaurateur doit être redirigé vers son compte IG avec les photos carrées.	serveurs d'Instagram via l'API puis publication sur le compte.	l'image carrée (avec html2canvas) et créer un conteneur média. Utilisation du endpoint /media_publish de l'API Graph pour finaliser la publication sur le compte.	menu, assurant un rendu professionnel sur Instagram.
Exporter le menu pour l'application Deliveroo	Au clic, l'utilisateur doit être redirigé vers l'application Deliveroo pour exporter son menu. Nécessite l'utilisation de l'API Menu de Deliveroo. Le menu doit être mappé aux champs obligatoires et aux contraintes de format de Delivroo.	Intégration API Deliveroo Menu (v3) Mappage de données et validation du schéma.	L'application doit s'authentifier (via clé API ou token) auprès de l'API Deliveroo et envoyer le menu complet au format JSON spécifique requis par Deliveroo. Mise en place d'un processus pour transformer le modèle de données du menu interne en un schéma de données conforme à l'API Deliveroo, avec vérification des champs obligatoires	1) Assure la transmission structurée et complète de toutes les données du menu au système Deliveroo. 2) Permet un déploiement direct du menu sur la plateforme sans ressaisie manuelle, garantissant rapidité et précision.
Accéder aux mentions légales	Depuis les pages connectées et déconnectées, ouverture d'une modale. La mention "Tous droits réservés" figure sur toutes les pages.	Composant Modale et composant Footer	Le texte des mentions légales est chargé dans un composant modal de React. Composant Footer inséré dans le Layout principal de l'application.	1) Assure la conformité légale sur l'ensemble du site. 2) Améliorer l'expérience utilisateur en évitant un chargement de page pour un contenu informatif secondaire.

Webgencia

Rendre l'application accessible	Le site devra être navigable depuis le clavier et lisible par un lecteur d'écran (conformité avec les normes WCAG).	Implémentation des rôles ARIA et tests avec des outils dédiés (Lighthouse).	Utiliser les attributs sémantiques corrects (ex: button au lieu de div cliquable), les attributs ARIA appropriés (pour les modales, les formulaires), et assurer le focus trap dans tous les composants modaux.	1) Répond à une exigence contractuelle du client et assure l'inclusivité du produit (critère fonctionnel et éthique). 2) Le respect des normes garantit un code sémantique et de meilleure qualité, réduisant les erreurs de développement.
Modèle de données				
Stockage des menus, plats, catégories, utilisateurs	L'application doit être capable de stocker une diversité de menus, plats, catégories et ingrédients. Le besoin de stockage est défini de manière durable dès la phase de conception du projet.	Base de données MySQL	Utilisation d'une base de données relationnelle pour stocker les utilisateurs, les menus, les catégories, les plats et leurs relations.	1) Permet d'établir des liens forts entre menus-catégories-plats-ingrédients 2) Stabilité de la structure de la base 3) Facilité d'installation et d'utilisation
Créer le branding de son restaurant	Le restaurateur doit pouvoir ajouter/modifier/supprimer le logo et les couleurs de base.	Endpoint API PUT/PATCH dédié Stockage local, couleurs gérées via CSS Variables	Le Logo est géré par un upload vers le système de fichiers local du serveur Node.js. Seul le chemin d'accès au fichier est stocké dans la base SQL. Les couleurs sont stockées en base et mises à jour dans le Front-end via des CSS Variables.	1) Simplicité de déploiement en n'ayant pas à configurer un service tiers pour le stockage des images. 2) Les CSS Variables permettent d'appliquer les changements de couleurs en temps réel à tous les menus, assurant une expérience utilisateur cohérente.

Modifier les informations utilisateurs	L'accès au formulaire doit être protégé par authentification. Le restaurateur doit pouvoir modifier son adresse e-mail de base. Il doit aussi pouvoir lier plusieurs adresses e-mail à son compte.	Endpoint API PUT/PATCH sécurisé et Système de Vérification d'e-mail.	Le Front-end envoie la nouvelle adresse e-mail à l'API. Le Back-end envoie un lien de vérification unique et à durée limitée à la nouvelle adresse. L'adresse n'est mise à jour dans la base de données qu'après confirmation par l'utilisateur via ce lien.	1) Assure la sécurité maximale du compte en vérifiant la propriété de la nouvelle adresse e-mail avant toute modification. 2) Garantit une structure de données évolutive en base (ex: table user_emails) permettant de stocker plusieurs adresses par utilisateur, répondant aux critères de succès
Gestion des données / communication Front-Back				
Création de la page connexion/inscription	Fenêtre modale s'ouvrant au clic "Se connecter". Processus d'authentification sécurisé. Le formulaire doit offrir une validation côté client (email pour s'authentifier ou de confirmation pour l'inscription).	Front-end : Création d'un composant modale en React et appel POST à l'API pour l'envoi du lien. Back-end : Création d'un endpoint REST API (POST /api/auth/magiclink) et d'un endpoint de validation (GET /api/auth/verify?token=...), utilisant un	**Front-end** : Développer un composant affichant un champ pour l'e-mail et un bouton "Envoyer le lien de connexion". Au clic, déclencher un appel POST à l'endpoint de l'API (ex: /api/auth/magiclink) contenant l'adresse e-mail. Afficher un message de succès clair (ex: "Veuillez consulter vos e-mails pour vous connecter/confirmer votre compte.") et rediriger vers une page d'attente/instruction. **Back-end :**	1) Le "Magic Link" améliore la sécurité (pas de mot de passe à stocker) et simplifie l'expérience utilisateur (pas de mot de passe à retenir). 2) Le système utilise des tokens à usage unique, réduisant les risques d'attaques par force brute. 3) Le Back-end gère à la fois l'inscription (si nouvel e-mail) et la connexion (si e-mail existant) via la même logique.

	Redirection de l'utilisateur vers l'outils après le succès de la connexion. Lien "Besoin d'aide" pour envoyer un email à l'équipe	service d'e-mailing et un Token Temporaire.	**POST /api/auth/magiclink** : Recevoir l'e-mail. Générer un token unique et le stocker en base de données avec l'e-mail et l'horodatage d'expiration. Construire le lien magique (/api/auth/verify?token=[TOKEN]). Envoyer l'e-mail contenant ce lien (Message : "Cliquez ici pour vous connecter/confirmer votre compte"). **GET /api/auth/verify** : Récupérer le token de l'URL. Vérifier sa validité (existence, non-expiré). Si valide : Invalider le token temporaire, générer et retourner un JWT de session permanent pour le Front-end. Si invalide : Retourner une erreur (Status 401).	
Fonction de déconnexion	L'action doit invalider la session utilisateur de manière sécurisée, de manière immédiate. Elle doit se faire depuis n'importe	Appel API POST /api/auth/logout (Back-end) et Redirection Client (React Router)	Au clic sur "Se déconnecter", le Front-end effectue un appel API sécurisé au Back-end (POST /api/auth/logout). Le Back-end prend alors la responsabilité d'invalider le JWT et de demander au navigateur de supprimer le Cookie HttpOnly qui contenait le jeton. La librairie de routage est ensuite utilisée	1) L'invalidation se fait **côté serveur** (où est géré le Cookie HttpOnly), assurant la suppression immédiate du jeton et empêchant l'accès continu aux données, conformément aux bonnes pratiques. 2) La déconnexion serveur permet d'invalider immédiatement le JWT**,** ce qui est essentiel pour la sécurité en cas de

	quelle page connectée.		pour rediriger l'utilisateur vers la route /.	vol de session. Le jeton est ainsi rendu inutilisable avant son expiration naturelle.
Accéder aux menus créés précédemment	Les menus doivent pouvoir être supprimé ou édité. La date de création doit apparaître. Un nouveau menu doit pouvoir être créé sur la même vue.	API CRUD (GET, DELETE, PUT/PATCH) et librairie de Formatage de Date (date-fns)	Front-end : Appel initial à GET /api/menus et affichage des résultats. Utilisation de la librairie date-fns pour formater la date de création dans un format lisible par l'utilisateur (ex: "Il y a 2 jours", "15/10/2025"). Implémenter les boutons "Éditer" (redirige vers le Wizard) et "Supprimer" (appel DELETE). Création du Bouton "Ajouter un nouveau menu" (ouvre le Wizard). L'API Back-end gère les requêtes GET pour la liste, DELETE pour la gestion des menus. La date de création est stockée en base, formatée par date, côté Front-end.	1) Assurer la gestion complète des données (Back-end) et offrir une meilleure UX en optimisant le formatage localisé des dates côté client. 2) Offre une expérience utilisateur fluide en centralisant les actions sur une seule vue sans navigation de page.
Commander l'impression d'un menu	Le lien doit s'ouvrir dans un nouvel onglet, vers le back-office de Qwenta. Le lien doit inclure l'identifiant unique du menu créé.	Lien Front-end configuré avec l'attribut target="_blank" et intégrant des paramètres d'URL.	Le Front-end construit l'URL complète du back-office Qwenta en ajoutant un paramètre de requête (?menuId=XYZ) pour identifier le menu à imprimer, puis utilise un élément <a href="..." target="_blank"> standard.	1) Assurer la continuité de l'expérience utilisateur en conservant l'application Menu Market ouverte. 2) Facilite l'intégration entre les systèmes en transmettant l'identifiant unique du menu, pour que le back-office sache quel fichier traiter.
Accéder aux tarifs de Manu Maker	La page des tarifs est externe (gérée par Qwenta).	Lien HTML <a> avec target="_blank" et	Le lien vers la page externe Qwenta est intégré sur le site. Lors du clic, il s'ouvre dans un nouvel onglet grâce à target="_blank". L'attribut	1) Assure la continuité de l'expérience utilisateur en ne fermant pas l'application

	Le lien doit s'ouvrir dans un nouvel onglet. L'URL est basée sur la logique Qwenta (/tarifs/menumaker).	rel="noopener noreferrer"	rel="noopener noreferrer" est ajouté pour la sécurité et la confidentialité.	Menu Maker lors de la consultation des tarifs. 2) Le rel="noopener noreferrer" est une bonne pratique de sécurité (technique) pour les liens externes qui s'ouvrent dans de nouveaux onglets Il empêche la page externe de manipuler la page d'origine et ne transmet pas l'URL de référence.

II. Liens avec le back-end

∉ Quel langage pour le serveur ? JavaScript avec l'environnement d'exécution **Node.js**

- A-t-on besoin d'une API ? Si oui laquelle ? Oui :
    - **API REST** utilisée pour l'authentification, la gestion des menus et l'utilisateur. Elle est construite avec le Framework Express.js qui lui-même s'exécute grâce à Node.js.
- Base de données choisie : **Base de données relationnelle MySQL**.

III. Préconisations concernant le domaine et l'hébergement

- Nom du domaine : En cours de validation, il sera très probablement un sous-domaine de Qwenta.
- Les différentes possibilités d'hébergement : 3 solutions peuvent être envisagées :
    - **L'hébergement mutualisé :** Le site web est hébergé sur un serveur partagé avec plusieurs clients. Les ressources (CPU, RAM, espace disque) sont allouées dynamiquement et partagées entre tous. Les performances du site peuvent être affectées par le pic de trafic d'un autre site. Il est idéal pour les petits sites. Solution la plus économique.

Webgencia

- **Le serveur dédié physique (ou cloud managé) :** Il correspond à la location d'une machine physique entière. L'utilisation de la plateforme d'exécution doit être gérée par le fournisseur. Il offre un contrôle total et des performances maximales mais il représente la solution la plus coûteuse et complexe à gérer. Il est adapté pour des grandes entreprises nécessitant des performances maximales.
- **Le Serveur Privé Virtuel (VPS)** : Il s'agit d'un serveur physique partitionné en plusieurs serveurs virtuels indépendants. Chaque VPS agit comme un serveur dédié avec des ressources garanties. Il assure une performance stable et prévisible. La scalabilité verticale facile permet d'augmenter progressivement la puissance des serveurs tout en maîtrisant les coûts. Il offre une solution intermédiaire et flexible.

- Nom de l'hébergement : OVH VPS (préconisation)
- Adresses e-mail : Les adresses e-mail suivantes doivent être provisionnées sur le domaine principal [nom-du-domaine].com :
  - contact@[nom-du-domaine].com : Point de contact général pour les demandes d'information, partenariats ou retours non techniques
  - support[nom-du-domaine].com : Adresse dédiée à la gestion du support client et au signalement des bugs
  - no-reply@[nom-du-domaine].com : pour toutes les communications automatiques de l'application (confirmation d'inscription, alertes, e-mail de connexion)

IV. Accessibilité

- Compatibilité navigateur : compatibilité avec les dernières versions de Chrome, Safari et Firefox.
- Types d'appareils : uniquement en version desktop pour le moment.
- L'application devra être accessible au minimum être navigable depuis le clavier et lisible par un lecteur d'écran.

Webgencia

V. Recommandations en termes de sécurité

L'objectif est d'assurer la sécurité globale de l'application en prenant en compte, en respectant les bonnes pratiques issues de l'OWASP Top 10 : l'authentification et la sécurisation des échanges de données.

Sécurité de l'authentification et des utilisateurs
- Email de confirmation pour vérification de l'e-mail - Gestions des sessions : utilisation des JWT stockées, en sécurité, dans un cookie HttpOnly sur le Front-end React - Durée de validité limité des tokens - Déconnexion : supprime la session et invalide le token côté serveur - Validation serveur des données soumises par le Front-end via des middlewares (express-validator)
Sécurité des échanges (API et base de données)
- Connexion sécurisée (HTTPS) entre le client et le serveur - Stockage des clés secrètes, tokens JWT et identifiants de base de données dans des variables d'environnement - Les messages d'erreur ne doivent contenir des informations sensibles - Contrôle d'accès : chaque endpoint de l'API doit vérifier si l'utilisateur est le propriétaire du menu concerné (Broken Access Control) - Protection de l'API REST contre les attaques par injection SQL grâce à des requêtes préparées ou d'un ORM (ex: Sequelize) - Accès à la base MySQL restreint avec des droits limités + sauvegardes régulières des données
Protection du Front-end
- Données affichées dans React sont échappées pour empêcher toute exécution de code malveillant (XSS) - Aucunes données sensibles stockées dans le code front-end ou localStorage - Configurer les en-têtes HTTP du serveur Node.js pour définir les sources de contenu fiables (CSP) - Mettre à jour régulièrement les dépendances - Utilisation des packages NPM officiels

VI. Maintenance du site et futures mises à jour

- Maintenance du site - *périodicité* :

Webgencia

Maintenance corrective
- Priorisation des anomalie et traitement des anomalies critiques empêchant l'utilisation du site + Correction rapide des anomalies détectées après la mise en ligne – *sous 48h* - Suivi des correctifs via Jira (gestion de la priorité, description du problème) - *selon la criticité du bug* - Archivage et versionning du code via GitHub - *selon la criticité du bug*
Maintenance préventive
- Mise à jour régulière des dépendances + tests de non-régression - *mensuelle* - Surveillance des performances du serveur et des logs d'erreurs - *hebdomadaire* - Sauvegarde régulière de la base de données - *quotidienne* - Contrôle de sécurité - *mensuelle* - Tests sur l'accessibilité et SEO - *à la demande*

- Futures mises à jour :

- ❑ Page d'accueil : Ajouter des animations sur la photo de la bannière + sur les formes géométriques des sections.
- ❑ Intégrer le tarif directement sur MenuMaker
- ❑ Infos utilisateur : créer la possibilité de changer de moyen de paiement
- ❑ Possibilité de créer un blog interne à MenuMaker
- ❑ Branding : intégrer un éditeur de thème visuel plus avancé (ombre, bordures, espacement, etc)
- ❑ Ajout d'une fonctionnalité de multilingue pour les menus

Lien du Kanban : https://www.notion.so/287d8bfab9d58084ae90f02887146ab4?v=287d8bfab9d5807fbe52000c5eec18dd&source=copy_link