Webgencia

Spécifications techniques

Menu Maker by Qwenta

Version	Auteur	Date	Approbation
1.0	Cécile Pecquerie	Octobre 2024	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)**
Compréhension et organisation du site	*Permettre à l'utilisateur de comprendre l'utilité du site en un coup d'œil, ainsi que ces fonctionnalités*	*Réalisation d'une maquette avec Figma*	*Création d'un modèle visuel du site via Figma pour représenter tous les éléments constitutifs de l'app et de ses fonctionnalités*	• *Outil user friendly, adapté aux débutants avec une prise en main rapide.* • *Outil majoritairement gratuit en plus d'être collaboratif*
Landing page	*Mettre en avant les fonctionnalités principales du site en un clin d'œil, transformer le visiteur en utilisateur*	*Landing page avec CTA (Call to Action) et présentation des fonctionnalités*	*La landing page mettra en avant les fonctionnalités clés avec des visuels attractifs et des boutons d'appel à l'action pour convertir les visiteurs en utilisateurs.*	• *Conçu pour maximiser la conversion des visiteurs* • *Crée une première impression engageante et claire des atouts du service.*
Connexion/ inscription	*Gestion de la connexion et de l'inscription de l'utilisateur de manière sécurisée*	*Django's Built-in Authentication System*	*Solution de gestion de connexion sécurisée intégrée à Django*	• *Django inclut des fonctionnalités de sécurité robuste par défaut comme le hachage des mots de passe et la protection contre les attaques courantes.* • *Le système d'authentification de Django est facile à configurer et à utiliser*
Dashboard	*Accès limité suivant le*	*Composant Vue.js*	*La création du menu*	• *Gestion granulaire des*

	type d'utilisateur (admin, manager, collaborateur…) permettant la création, diffusion et impression des menus. Affiche les 3 derniers articles du blog Qwenta sur Menu Maker	*pour les différents éléments du tableau de bord et Django pour la gestion des rôles*	*permet une prévisualisation en direct grâce à Vue.js pour l'interaction visuelle, tandis que Django assure la gestion des données et la persistance.*	*permissions pour garantir l'accès sécurisé aux fonctionnalités selon le rôle* • *L'affichage dynamique via Django simplifie la gestion des contenus et rôles.*
Création d'un menu	*Voir les grandes lignes de la création du menu avec une prévisualisation en directe du rendu*	*Modale interactive pour ajouter des plats, incluant nom, prix, description et image.*	*La création du menu permet une prévisualisation en direct grâce à Vue.js pour l'interaction visuelle, tandis que Django assure la gestion des données et la persistance*	• *Vue.js rend l'expérience plus fluide avec une prévisualisation instantanée sans rechargement.* • *Django gère la logique métier, garantissant la cohérence et l'intégrité des données.*
Création de catégorie de plat	*Avoir une modale pour ajouter une catégorie à chaque plat qui pourra être réutilisable pour faciliter l'expérience de l'utilisateur et rendre la création plus intuitive*	*Composant modale réutilisable Vue.js et enregistrement des données dans la bdd PostgreSQL via l'API Django*	*Une modale permet d'ajouter/modifier des catégories de plat réutilisables, stockées par Django dans la bdd. Vue.js assure une interaction fluide sans rechargement de page.*	• *Simplifie la gestion des catégories avec des modales intuitives et assure une réutilisabilité des catégories pour faciliter la création de menus.* • *Django gère la persistance des données de manière robuste.*
Création de plat	*Nouvelle fenêtre modale permettant l'ajout des spécificités du plat en reprenant la catégorie réutilisable mais aussi la modifier, pouvoir*	*Modale interactive pour ajouter des plats, incluant nom, prix, description et image.*	*La création de plat est gérée dans une modale où les informations comme le prix, la description et la photo sont saisies. Vue.js assure*	• *Offre une expérience fluide sans interruption du flux utilisateur grâce à Vue.js.* • *Garantit l'intégrité des données avec validation et persistance côté serveur via Django.*

	renseigner le prix, le nom, une description et charger une photo		*l'interaction utilisateur et Django gère les données et la validation.*	
Personnalisation du style du menu	*Possibilité de personnaliser le style via la typo et les couleurs. Utilisation de boutons dédiés à ces éléments. Application du changement en temps réel sans rechargement de la page*	*Gestion des états via Vuex pour les boutons de personnalisation et sauvegarde via Django*	*Création de bouton permettant le changement de style (typo et couleur) avec Vue.js, mise en forme gérée par CSS, utilisation de Vuex pour gérer l'état des boutons et permettre la mise à jour en directe de la prévisualisation. Enregistrement de la nouvelle mise en forme via Django*	• *Vue.js gère parfaitement le changement d'état et la mise à jour en temps réel du style.* • *Django permet de gérer la sauvegarde et l'application de styles personnalisés.*
Exportation du menu en pdf	*Export du menu au format PDF en un clic. Doit être optimisé pour l'impression*	*Utilisation de Python ReportLab*	*Génération du PDF coté serveur avec ReportLab, envoi via l'API Django et téléchargement géré par l'interface Vue.js*	• *ReportLab est une bibliothèque Python puissante pour générer des PDF optimisées et complexes* • *Vue.js facilite la gestion de l'interface utilisateur pour télécharger ou afficher le fichier*
Commande d'impression	*Commande d'impression en un clic*	*Utilisation de la solution intégrée de Qwenta*	*Qwenta propose une offre d'impression accessible via un appel API*	• *Solution existante et fait partir de l'offre principale de Qwenta.* • *Ne nécessite pas la mise en place d'un service supplémentaire couteux*
Display menu	*Affichage de tous les menus déjà créé avec*	*Affichage des menus créés avec options*	*Django gère l'affichage des menus stockés, et*	• *Vue.js offre une recherche rapide et fluide sans rechargement.*

	possibilité de les filtrer	*de filtrage.*	*Vue.js permet de filtrer et trier ces menus de manière dynamique sans recharger la page.*	• *Django assure une gestion sécurisée des menus avec une présentation organisée.*
Mentions légales	*Affichage des mentions légales dans une modale à texte fixe*	*Réutilisation d'une modale Vue.js*	*Modale simple en Vue.js affichant un contenu statique*	• *Vue.js est léger et performant pour gérer les modales sans complexités.* • *Gestion facile du contenu statique en front end sans surcharge du serveur*
Tous droits réservés	*Mention « tous droits réservés » doit apparaitre sur toutes les pages mais sans être intrusives*	*Composant réutilisable Vue.js*	*Création d'un composant réutilisable Vue.js dont le placement est géré avec CSS*	• *Modularité des composants entre eux et entre les pages* • *Réutilisabilités des composants sur différentes pages*
Tarification	*Affichage directe des tarifs suivant les options choisies par l'utilisateur*	*Utilisation de Stripe pour la gestion de la tarification et du paiement des utilisateurs*	*Stripe est une plateforme de paiement en ligne qui offre une suite d'API pour intégrer des fonctionnalités de paiement dans les applications.*	• *Stripe permet l'utilisation de la majeure partie des moyens de paiements : carte bancaire, PayPal, Apple Pay, Google Pay, etc… Et permet aussi la gestion des erreurs de paiement.* • *Stripe propose des API facilement intégrable avec les langages les plus connus.*
Exportation Deliveroo	*Connexion en un clic à Deliveroo pour exporter le menu avec une optimisation des*	*Intégration de l'API Deliveroo avec Django views*	*Utilisation de l'API gratuite de Deliveroo en l'intégrant directement avec Django views*	• *Django views permettent de créer un endpoint pour interagir avec une API tierce* • *L'intégration directe dans*

	données			*Django simplifie la gestion de l'exportation vers Deliveroo*
Partage Instagram	*Possibilité de partager le menu sur Instagram en respectant les formats attendus par Instagram*	*Utiliser Instagram Graph API pour publier directement les menus ou les stories.*	*Permet aux développeurs de récupérer des données et d'exécuter des actions sur les comptes Instagram Business ou créateurs.*	• *L'API Graph utilise l'authentification OAuth 2.0 pour garantir la sécurité des données et des autorisations.* • *L'API Graph permet de récupérer les données en temps réel.*
Déconnexion	*Déconnexion en un clic avec suppression du token d'authentification*	*Utilisation de Django's Built-in Authentication System*	*Utiliser Django's Built-in Authentication System pour gérer la déconnexion de manière sécurisée. Il gère par défaut la connexion et déconnexion*	• *Django inclut des fonctionnalités de sécurité robustes par défaut, comme la suppression du token d'authentification.* • *Le système d'authentification de Django est facile à configurer et à utiliser, ce qui simplifie le développement.*
Infos utilisateur	*Possibilité de modifier les données de l'utilisateur et de lier plusieurs comptes*	*Formulaire Vue.js avec lise à jour Django Models*	*Création d'un formulaire dynamique en Vue.js pour modifier les données utilisateurs et gérer les comptes, avec persistances des modifications via Django models*	• *Vue.js permet une gestion fluide des formulaires avec validation directe* • *Django models facilitent grandement la mise à jour des informations utilisateur.*
Branding restaurateur	*Gestion de l'image de l'entreprise avec ajout/modification du logo, gestion de la*	*Composant Vue.js avec Django image upload*	*Création d'une interface réutilisable en Vue.js et gestion de l'ajout du logo grâce à Django image*	• *Vue.js offre une expérience utilisateur intuitive pour ajouter/modifier l'image de l'entreprise.*

	couleur du couleur du thème de l'utilisateur		*upload*	• *Django image upload intégré facilite grandement la gestion des fichiers images.*

II. Liens avec le back-end

- Quel langage pour le serveur ?

Utilisation de Python pour le langage back end. Python est un langage plus robuste et plus sécurisé que JavaScript. Il offre également l'avantage d'avoir une meilleure lisibilité du code. De plus, c'est un langage très utilisé, il y a donc une communauté très active utilisant ce langage. Pour mieux tirer parti des capacités de Python, le framework Django sera le plus adapté car il comporte une documentation très complète, une gestion par défaut de l'authentification sécurisée et plusieurs extensions qui seront très pratique pour la gestion des appels API. De plus c'est un framework possédant un écosystème très mature avec des outils très bien intégrés.

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

L'utilisation de Python via Django et Vue.js fait qu'il n'est pas nécessaire d'utiliser une API dédiée. En effet, Django permet une organisation, stockage, et un traitement des données, ainsi que du rendering directement coté serveur sans avoir à demander au front de traiter les données reçues (comme avec React et node.js par exemple) Cela permet la séparation des responsabilités entre le front end et le back end, ce qui facilite le développement, le maintien et l'évolution de chaque partie indépendamment. De plus, cette approche permet d'utiliser les ressources du serveur et non de la machine de l'utilisateur en passant par un traitement des données directement par le navigateur client. De cette façon, nous nous assurons d'un parfait fonctionnement du site, peu importe la qualité de la connexion internet de l'utilisateur. Les seuls API utilisées seront des API tierces (Deliveroo, Qwenta, Instagram, Stripe...) qui seront intégrée grâce à Django.

- Base de données choisie

Une base de données relationnelle semble plus appropriée pour ce projet. En effet, les plats pouvant être liés à une catégorie, mais aussi à un ou plusieurs menu, une relation claire se dessine entre les différentes données qui seront stockées. Mon choix se porte sur PostgreSQL pour la gestion de la base de données. Il a l'avantage d'être puissant et fiable et d'offrir plus de fonctionnalités que MySQL. De plus il a un excellent support de la programmation orienté objet via PL/Python. Il permettra également une meilleure adaptabilité suivant l'évolution de l'application dans le temps.

III. Préconisations concernant le domaine et l'hébergement

- Nom du domaine :

En ce qui concerne le nom de domaine, la meilleure action serait de faire au plus simple. Une option serait menumaker.fr, par exemple. On peut également je rattacher au nom de domaine déjà existant de Qwenta avec menumaker.qwenta.fr

- Nom de l'hébergement.

Deux options sont envisageables pour l'hébergement ; un hébergement avec services managés ou un serveur dédié/VPS. Il faut prendre en compte les exigences du RGPD. Il faut donc un hébergeur en France, ou éventuellement en Suisse. Hostinger propose à la fois de l'hébergement à services managés ainsi que du serveur dédié et du VPS. Ils intègrent parfaitement les architectures Vue.js et Django. Ils ont une des offres la plus large du marché, un bon service client et une facturation très concurrentielle.

- Adresses e-mail.

Pour la gestion des e-mails de type notifications, confirmation d'inscription, etc., la meilleure option est d'utiliser un service dédié pour assurer une bonne délivrabilité et simplifier la gestion des e-mails transactionnels. SendGrid est un service facilement intégrable à Django. Pour les e-mails professionnels (au sein des équipes Qwenta), si aucun service n'est déjà en place, Google Workplace offre l'avantage d'avoir un hébergement fiable pour les e-mails de domaine personnalisé. L'interface est également familière et a un bon support de l'envoie via SMTP.

IV. Accessibilité

- Compatibilité navigateur

Il est prévu pour le moment une compatibilité avec les dernières versions de Chrome, Safari et Firefox (ainsi que les navigateurs tournant sur cœur Chromium). La compatibilité Edge est à prendre en compte en dernière version uniquement. Si on ne souhaite pas vérifier la compatibilité manuellement, on pourra utiliser des outils tels que BrowserStack ou LambdaTest.

- Types d'appareils

Le site est uniquement prévu en version desktop.

- Accessibilité et SEO

Le site devra répondre aux critères d'exigences SEO pour un bon référencement (exemple : méta données, schéma.org etc...). Le site devra également répondre aux standards d'accessibilité minimum tels que la navigation depuis un clavier ou être lisible par un lecteur d'écran. Il sera possible de contrôler l'accessibilité du site grâce à l'extension Wave. Il faudra s'assurer que le code couleur du site ci-dessous réponds bien aux critères de contrastes de Wave pour une bonne accessibilité :

- ✓ Beige : #FFF4E8
- ✓ Green : #8BC7B1
- ✓ Black : #000
- ✓ White : #FFF
- ✓ Brown : #C5A073

V. Recommandations en termes de sécurité

- Authentification et autorisation

Afin de gérer l'authentification robuste, les sessions et la protection contre les attaques courantes, on utilisera les fonctionnalités intégrées de Django à cet effet (Django's Built-in Authentication System). On implémentera également un système de permissions et de rôles pour contrôler l'accès aux différentes parties de l'application en fonction du type d'utilisateur (admin, manager, collaborateur, etc.).

- Protection des données

On utilisera un algorithme de chiffrement robuste pour protéger les données sensibles (tels que mots de passe et informations de paiement). Il faudra également utiliser le protocole HTTPS pour la communication entre le client et le serveur, grâce à un certificat SSL.

- Protection contre les attaques courantes

Nous utiliserons des bibliothèques de sécurité pour échapper aux attaques XSS (Cross-Site Scripting), des token CSRF pour protéger les formulaires et les requêtes contre les attaques CSFR (Cross-Site Request Forgery)

- Sécurisation des fichiers uploadés

Il faudra limiter les types de fichiers acceptés et utiliser une bibliothèque tel que ClamAV pour analyser les fichiers uploader et

détecter les contenus malveillants

- Mise à jour et audit régulier des dépendances

Il faudra garder en tout temps les dépendances et plugins à jours pour éviter les vulnérabilités. Il faudra donc s'assurer d'utiliser uniquement des dépendances et plugins qui sont maintenus régulièrement. Un audit régulier des dépendances pourra se faire via 'pip audit' pour s'assurer qu'aucune vulnérabilité n'est détectée.

VI. Maintenance du site et futures mises à jour

- Maintenance du site

- Correction de bugs : Après le déploiement, une équipe dédiée doit être en place pour corriger rapidement les problèmes. Un délai de réponse doit être défini en fonction de la gravité du bug.

- Mises à jour de sécurité : Les mises à jour de sécurité, incluant celles des dépendances, bibliothèques et frameworks, doivent être effectuées régulièrement afin de prévenir les vulnérabilités.
- Surveillance continue : Mettre en œuvre un système de monitoring, pour détecter en temps réel les anomalies de performance, les erreurs ou les problèmes d'infrastructure.

- Support utilisateur : Offrir un service de support aux utilisateurs via différents moyens de contact (email, chat, etc.). Cela inclut également une FAQ, un guide d'utilisation, ou un tutoriel interactif pour accompagner les restaurateurs dans la personnalisation et l'utilisation des menus.

- Mises à jour potentielles

- Ajout de nouvelles fonctionnalités : Intégrer régulièrement de nouvelles fonctionnalités pour améliorer le service.

- Optimisation des performances : Mettre en place des améliorations comme la mise en cache, le lazy loading des composants ou l'optimisation des images pour une meilleure efficacité.

- Évolution de la base de données : Adapter la base de données en fonction de la croissance du nombre d'utilisateurs et de la complexité des menus. Envisager aussi l'ajout de fonctionnalités comme la journalisation et l'historique des modifications des menus.

- Mise à jour des API externes : Les API utilisées (Deliveroo, Instagram) peuvent évoluer, il est donc nécessaire de vérifier la compatibilité de l'application avec leurs mises à jour.

- Cycle de vie des versions

Pour structurer le développement et les mises à jour, il est recommandé d'établir un cycle de vie des versions, tel que :

- Versions majeures : Introduction de nouvelles fonctionnalités importantes, comme une nouvelle interface utilisateur ou une nouvelle intégration.
- Versions mineures : Améliorations et optimisations des fonctionnalités existantes, telles que l'optimisation des performances ou le refactoring du code.
- Versions correctives : Corrections de bugs et déploiement de mises à jour de sécurité.