Webgencia

Spécifications techniques

Projet	Menu Maker By Qwenta

Version	Auteur	Date	Approbation
1.0	Chéveny Sylvain	26/09/23	Soufiane (Product Owner), John (Qwenta)

Le but de ce document est de définir et justifier les spécifications techniques de Menu Maker By Qwenta.

I. Choix technologiques

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

Webgencia

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Ouverture de la modale de connexion sécurisée	Ouverture d'une modale	React-modal React-hook-form	Création de modales performantes avec React et personnalisation avec des propriété CSS Création d'un formulaire avec un champ « mail » et un bouton submit « se connecter »	Facilité d'utilisation et de configuration bibliothèque populaire
Connexion sécurisée **Authentification par mail**	Connexion sécurisée nécessaire Confirmation par mail obligatoire	Passport.js en association avec Cognito	Avec passport.js, créer une stratégie d'authentification personnalisée en interaction avec Cognito en utilisant le mail comme principal identifiant. Envoi d'un mail, avec un jeton d'authentification et connexion de l'utilisateur.	Pertinence pour le besoin Compatibilité avec Node.js
Créer une catégorie de menu	Ajout de catégorie depuis une modale contenant un formulaire Prévisualisation des photos téléchargées et du menu en cours de création	React-modal. React-hook-form File Reader	Idem File Reader est une API JS native qui permet de lire des fichiers depuis le navigateur web.	Cohérence avec le développement Popularité de React Cohérence avec JavaScript Popularité

Créer un plat	Idem Pouvoir renseigner le nom du plat, son prix, sa description et ajouter des photos. Prévisualisation des photos téléchargées et du menu en cours de création	Idem Module Multer de Node.js File Reader	Idem Gestion du téléchargement des fichiers dans les applications express.js File Reader est une API JS native qui permet de lire des fichiers depuis le navigateur web.	Idem Cohérence avec l'utilisation de Node.js et Express.js Module populaire Cohérence avec JavaScript Popularité
Personnaliser l'application et du menu	Personnalisation de l'application en changeant la mise en page, police, couleur etc Personnalisation du menu par le restaurateur (logo, police, couleur)	Bootstrap React-color Google Font	C'est une bibliothèque CSS qui facilite la création de style personnalisé. Sélecteur de couleurs Bibliothèque de polices sur Google	Bibliothèque très populaire et donc bien documentée. Personnalisation facile à l'aide de classes prédéfinies et de variables personnalisables. Cohérence avec l'utilisation de REACT Bibliothèque de police complète et populaire
Diffuser le menu	Exportation du menu au format PDF vers Instagram et Deliveroo	FileSaver.js API Deliveroo API Instagram	FileSaver est une bibliothèque JS qui facilite le téléchargement de fichiers depuis le navigateur. Elle permet de générer un fichier à partir de données et de le proposer en téléchargement à l'utilisateur.	Cohérence avec JavaScript Cela correspond au besoin

Imprimer un menu	Impression du menu.	React-pdf	Génération d'une version PDF imprimable avec React	On utilise déjà React Cela correspond au besoin
Sauvegarde des menus **Menus précédents**	Après création d'un menu, on doit pouvoir le sauvegarder dans la base de données et ainsi le consulter et/ou le modifier ultérieurement Avec le lien « mes menus », créer une modale avec les menus précédemment créés affichées dynamiquement ainsi qu'un lien pour la création de nouveaux menus.	API créée avec Express Serveur Node Base PostgreSQL React-Modal API de l'application créée avec Express.js gérée par Node.js	Solution dans le chapitre II	Expliquées dans le chapitre II

Webgencia

II. Liens avec le back-end

Quel langage pour le serveur ?

➔ **Node.js** est un environnement de développement serveur qui permet aux développeurs de créer des applications web performantes et adaptables en **utilisant JavaScript** (JS utilisé à la fois pour le frontend et le backend ce qui permet une cohérence dans la logique de programmation, réduit le besoin d'apprendre plusieurs langages et simplifie la communication entre les équipes backend et frontend). Node.js est conçu pour gérer efficacement les opérations asynchrones et non bloquantes ce qui le rend particulièrement adapté aux applications en temps réel et aux chargements élevés et donc peut gérer simultanément de nombreuses connexions sans utiliser de grandes ressources système (parfaitement adapté à un projet qui peut croître en terme de trafic).

Webgencia

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

➔ Création de l'API de l'application : **Express,js**, c'est un des frameworks les populaires pour la création d'API en utilisant Node.js. Il simplifie la gestion des routes, des requêtes et des réponses, ce qui permet de se concentrer plus sur la logique métier. Express.js est un **framework non restrictif**, ce qui signifie une plus grande flexibilité de structurer l'application selon les besoins.

De plus Node.js et Express.js a une **communauté aussi très active** et donc a une **bonne documentation** pour faire face à d'éventuels problèmes courants.

De plus, nous pourrons utiliser les API de Deliveroo et Instagram, pour exporter les menus et partager sur Instagram respectivement.
Nous pourrons utiliser aussi Cognito (Amazon Web Services) qui propose des fonctionnalités d'authentification et de gestion de l'identité utilisateurs pour notre application.

➔ Type de base de données choisie : **PostgreSQL** est pertinent pour ce projet car le trafic sur ce projet peut tendre à croître. Cette base de données peut facilement **évoluer avec nos besoins sans nécessiter de changement majeurs dans l'architecture du projet**. En effet, cette base de données **open source** est réputée pour sa robustesse, sa conformité aux normes SQL et sa capacité à gérer des charges de trafic plus importantes. De plus, PostgreSQL a une communauté active qui fournit des mises à jour et une documentation solide.

III. Préconisations concernant le domaine et l'hébergement

➔ Nom du domaine : **qwentamaker.com** Nom unique et distinctif qui reflète le nom de l'application de manière claire et mémorable. En allant sur **Google Domains**, on constate que le nom est disponible. De plus, choisir l'extension de domaine .com est plus pertinent que .fr dans l'optique que le public de l'application devienne international.

En attente de validation.

➔ Nom de l'hébergement : **Amazon web services (Amazon Elastic Compute Cloud ou AWS EC2)**
On a préféré un VPS à un hébergement partagé car en cas d'augmentation de trafic, on aurait eu trop de problèmes de performances sur les temps de chargement.
AWS EC2, bien que populaire, est aussi le choix qui offre le plus de **flexibilité tant qu'à la tarification**, la **variété d'instance** avec différentes combinaison de ressources (CPU, mémoire, stockage) pour s'adapter à diverses charges de travail. Ce service est connu aussi pour sa **performance élevée** grâce à l'utilisation des disques **SSD**.
Disponibilité et fiabilité grâce à sa distribution mondiale de centres de données.

➔ Adresses e-mail : c**ontact@qwenta.com**.

Webgencia

IV. Accessibilité

➔ Compatibilité navigateur : Chrome, Firefox et Safari. Ce sont les 3 principaux navigateurs utilisés par les internautes et ainsi on touche le plus large public possible.

➔ Types d'appareils : Desktop, tablette et smartphone. Les 3 supports de navigations les plus utilisés par les internautes. Donc besoin d'un site web responsive.
NB : pas d'application mobile prévue.

➔ On nous a demandé d'utiliser les couleurs suivantes pour le design de l'application :

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

➔ Il est important de suivre les normes et recommandations du **W3C** concernant l'accessibilité visuelle sur le web.
On recommande d'utiliser **ARIA** (Accessible Rich Internet Applications) : c'est un semble de technologies permettant d'améliorer l'accessibilité des applications web développées en utilisant des technologies comme JavaScript.
Pour le **HTML5**, il recommande d'utiliser les **attributs alt** pour la possibilité d'ajouter des descriptions textuelles.
Le W3C a également travaillé sur des normes pour rendre les graphiques et les images vectorielles accessibles, ce qui est essentiel pour les personnes ayant une basse vision.

V. Services tiers

➔ Aucun outil intégré pour le suivi du comportement utilisateur n'est à prévoir.

➔ **SendGrid** : c'est un pionnier dans le domaine de l'envoi d'emails transactionnels avec un **solide réputation** en matière de délivrabilité.
Il offre une **large gamme de modèles de campagne prédéfinis** pour vous aider à créer des emails attrayants rapidement.
SendGrid propose de nombreuses intégrations avec des plateformes tierces, ce qui facilite l'intégration dans votre flux de travail existant.
Il offre des **outils de suivi** des emails détaillés pour surveiller les performances des emails.
➔ https://docs.sendgrid.com/fr/

➔ **NewRelic** : cette fonctionnalité se concentre sur la surveillance des performances des applications, des serveurs et de l'infrastructure, ainsi que sur l'optimisation des performances
On peut créer des **tableaux de bord** personnalisés pour surveiller les métriques et les KPI spécifiques à une application. On peut configurer des **alertes personnalisés** en fonction de divers critères de performances.
➔ https://newrelic.com/fr

VI. Recommandations en termes de sécurité

➔ Utiliser **HTTPS** : chiffrage des données en transit, protégeant ainsi la confidentialité des utilisateurs.

➔ Gestion des **mots de passe** : stockage des mots de passe en utilisant des méthodes de hachage robustes. Aussi, il faut recommander aux utilisateurs de créer des mots de passe forts et à les changer régulièrement.
(à voir à l'avenir)

➔ **Authentification et autorisation** : mettre en place un système d'authentification sécurisé (à double facteur) pour vérifier l'identité des utilisateurs.
Utiliser des mécanismes d'autorisation pour contrôler l'accès aux ressources de l'application en fonction des rôles et des privilèges.

➔ **Protection contre les injections SQL** : utiliser des requêtes préparées ou des ORM (Object-Relational Mapping) pour éviter les injections SQL.

➔ **Protection contre les attaques XSS et les attaques DDoS** : filtrer et échapper correctement les données utilisateurs pour éviter les attaques de script intersite (XSS).
Mettre en place des mesures de protection DdoS, telles que la limitation des taux et le filtrage du trafic malveillant.

➔ **Sécurité des sessions** : utiliser des jetons de session aléatoires et des mécanismes anti-contrefaçon de demande (CSRF).

➔ **Tests de sécurité** : effectuer des tests de pénétration réguliers pour identifier et corriger les vulnérabilités de sécurité.

➔ **Conformité légale** : s'assurer de respecter toutes les lois de et réglementations en matière de protection des données.

VII. Maintenance du site et futures mises à jour

➔ Sauvegardes régulières des données importantes hors site pour pouvoir le restaurer en cas de problème. Dans ce cas, on peut utiliser **Amazon S3**.
Nous pouvons instaurer une sauvegarde automatique tous les jours à une heure de basse affluence. (à voir avec le client)

➔ Mises à jour régulières du site incluant des corrections de sécurité, des améliorations de performances.