Webgencia

Spécifications techniques

[Menu Maker by Qwenta John, Qwenta]

Version	Auteur	Date	Approbation
1.0	Brichet Emmie	28/11/2024	John, Qwenta

1. Choix technologiques

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

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Landing non connectée	L'internaute doit pouvoir avoir accès aux différentes sections de la landing page non Connectée : Bannière Personnalisez votre menu Explications étape par étape	React.js	**React.js** permet d'avoir accès aux différentes parties du landing page en mode non connecté, telles que : La bannière La personnalisation du menu Les explications étape par étape. Il permet de réaliser le frontend du site et, à l'avenir, facilitera la gestion efficace des composants, en permettant de les réutiliser pour la mise en page de mon site, notamment pour la gestion des images et des boutons. React.js offre également la possibilité d'ajouter facilement des animations, ce qui permettra, par la suite, d'animer les éléments de ma page de manière simple et intuitive.	**React** Modularité et Réutilisabilité : React permet de créer des composants réutilisables pour chaque section de la landing page (bannière, personnalisation, étapes). Ces composants sont faciles à maintenir et à modifier, ce qui rend le développement plus rapide et évolutif. Dynamisme et Performance : Grâce au Virtual DOM, React offre une expérience fluide et rapide, idéale pour intégrer des interactions dynamiques (animations, transitions) et améliorer l'expérience utilisateur sans recharger la page.
Page login Créer un compte	La fenêtre doit s'ouvrir sous forme de modale	React Node.js avec Express	**React** permet de créer le formulaire	**React** : Gère l'interface utilisateur pour la fenêtre

Connecter	L'utilisateur peut Entrer son adresse mail Qu'il se soit déjà connecté ou non, un mail lui est envoyé pour lui permettre de s'authentifier, ou au contraire de confirmer son mail pour accéder pour la première fois à l'application Un lien "Besoin d'aide" permet d'envoyer directement un e-mail à nos équipes	Nodemailer	d'inscription/connexion, en ajoutant des champs pour l'adresse email et des boutons pour s'inscrire ou se connecter. Il gère l'action sur le formulaire, en prenant en charge les événements nécessaires pour envoyer un email d'authentification ou de confirmation. Concernant le lien d'aide, React ajoute un lien qui redirige vers un formulaire ou lance une action pour envoyer un email. Pour la gestion de l'email d'authentification, **Node.js avec Express** permet de gérer l'API pour l'envoi d'e-mails et le traitement des utilisateurs, tandis que **Nodemailer** est utilisé pour l'envoi des emails.	modale de login, y compris les champs de formulaire pour l'adresse email et les boutons d'inscription/connexion. Il gère également les événements du formulaire pour envoyer des emails d'authentification ou de confirmation, ainsi que le lien "Besoin d'aide" qui déclenche l'envoi d'un email d'assistance. **Node.js avec Express et Nodemailer** : Node.js avec Express gère les requêtes API pour envoyer des emails d'authentification ou de confirmation d'adresse email. Nodemailer est utilisé pour envoyer ces emails, ainsi que pour gérer l'envoi d'un message d'assistance lorsqu'un utilisateur clique sur le lien "Besoin d'aide".
Renseigner une catégorie	L'utilisateur doit pouvoir : Créer une nouvelle catégorie La création de catégorie s'ouvre dans une modale spécifique, et doit pouvoir être validée	React Node express JWT (JSON Web Token) PostgreSQL	**React :** Affiche une modale pour saisir le nom d'une nouvelle catégorie. Envoie une requête POST au backend pour ajouter la catégorie. Met à jour l'interface pour inclure les nouvelles catégories dans la liste déroulante.	**Complémentarité pour une solution complète et sécurisée** : React gère une interface utilisateur fluide et interactive, tandis que Node.js avec Express traite les requêtes backend et sécurise les données avec JWT. PostgreSQL assure une gestion fiable et performante

	Sélectionner une catégorie créée précédemment		**Node.js + JWT :** Vérifie que l'utilisateur est authentifié. Reçoit la requête contenant le nom de la nouvelle catégorie. Enregistre cette catégorie dans la base de données. **PostgreSQL :** Ajoute la catégorie dans la table des catégories. Fournit les catégories disponibles lorsqu'une requête GET est envoyée depuis le frontend.	des catégories en base de données. **Évolutivité et modernité** : Ce stack moderne est conçu pour évoluer facilement. React permet d'ajouter des fonctionnalités à l'interface, tandis que Node.js et PostgreSQL offrent une structure robuste pour gérer des données croissantes et des besoins de sécurité renforcés.
Ajouter un plat / des plats dans mon menu	Le restaurateur doit pouvoir : Sélectionner une catégorie Modifier la catégorie sélectionnée Entrer les informations de son plat Une modale s'ouvre pour saisir les différents plats de la catégorie Chaque plat peut avoir : Une photo associée Un nom Un prix Une description	React Node JWT PostgreSQL	**React** : Le restaurateur sélectionne une catégorie ou en crée une nouvelle. Une modale s'ouvre pour saisir les informations des plats (nom, prix, description, photo). Les données sont envoyées à l'API pour être enregistrées. **Node.js** : Reçoit les données du plat et vérifie les permissions via le JWT. Insère les données dans PostgreSQL ou les met à jour. **PostgreSQL** : Les plats et leurs informations sont stockés dans des tables,	**Gestion dynamique et sécurisée des données** : React permet de gérer dynamiquement l'interface, comme l'ajout de plats et la sélection de catégories via des modales, tout en offrant une expérience utilisateur fluide. Node.js, avec JWT, assure que seules les actions des utilisateurs authentifiés sont validées, garantissant la sécurité des données. **Stockage fiable et évolutif** : PostgreSQL offre une gestion robuste des données des plats et des catégories, permettant de maintenir des relations complexes entre elles. Ce système est scalable, idéal

	Il est possible de créer autant de plats que l'on souhaite		avec des relations entre les catégories et les plats. **JWT** : S'assure que chaque requête est authentifiée et que l'utilisateur est autorisé à effectuer l'action.	pour gérer un grand nombre de plats et de catégories à mesure que le menu se développe.
Pouvoir personnaliser le style	Le restaurateur doit pouvoir : Visualiser le menu actuellement créé Sélectionner une typographie Choisir une couleur de texte	React Node.js + Express.js PostgreSQL JWT	**React** gère le rendu des composants de l'interface utilisateur, en mettant à jour automatiquement la vue lorsque l'état du menu ou des préférences de style change. **Express.js** facilite la gestion des routes HTTP (par exemple, obtenir un menu, enregistrer des préférences de style). **Node.js** permet d'exécuter JavaScript sur le serveur pour gérer les requêtes en temps réel. **PostgreSQL** permet de stocker et de récupérer des données, comme le contenu du menu ou les paramètres de style. Il est accessible via des requêtes SQL effectuées depuis l'API Express.js. Lorsqu'un restaurateur se connecte, un token **JWT** est généré et envoyé au client. Ce token est ensuite utilisé pour authentifier les	**React** : Permet d'afficher et de mettre à jour dynamiquement l'interface utilisateur, en offrant au restaurateur la possibilité de visualiser et personnaliser le style du menu en fonction des choix effectués (typographie, couleur de texte). **Node.js, PostgreSQL, JWT** : Node.js gère les requêtes API pour enregistrer et récupérer les préférences de style, PostgreSQL stocke ces préférences et le menu, et JWT assure que seules les actions des utilisateurs authentifiés sont autorisées, garantissant ainsi la sécurité de la personnalisation.

			requêtes suivantes, garantissant que seul un utilisateur authentifié puisse accéder ou modifier certaines ressources (comme la personnalisation du menu).	
Commander en un clic l'impression d'un menu	L'encart "Imprimer un menu" doit être visible depuis la page d'accueil Il s'agit d'un lien qui s'ouvre dans un nouvel onglet Le lien doit être fait vers le back-office de Qwenta	React Node.js + Express.js PostgreSQL JWT	**React** affiche le menu et l'encart "Imprimer un menu". Gère les interactions Utilise des composants réutilisables Gère les erreurs **Node.js** gère les routes API. Fournit l'API de génération du menu à imprimer Assure la connexion au back-office Qwenta **PostgreSQL** stocke les menus et la gestion des utilisateurs. **JWT** est utilisé pour sécuriser les sessions utilisateurs.	**React** : Affiche l'encart "Imprimer un menu" et gère l'interaction avec l'utilisateur en ouvrant un lien vers le back-office Qwenta. **Node.js, PostgreSQL, JWT** : Gère la logique backend pour récupérer les menus à imprimer, sécuriser l'accès via JWT et interagir avec la base de données PostgreSQL.
Exporter mon menu en PDF.	Le restaurateur doit pouvoir en un clic télécharger le fichier PDF correspondant à son menu	React Node.js avec Express PostgreSQL JWT	**React:** Crée un bouton "Télécharger le PDF" qui envoie une requête au backend pour télécharger le PDF. **Express :** Crée une route qui génère un fichier PDF à partir des données du menu et le renvoie au	**React et Node.js avec Express** permettent une architecture claire avec une séparation du frontend et du backend, chacun gérant ses propres responsabilités. **PostgreSQL** permet de gérer les données de manière fiable et structurée, tandis que

			frontend pour être téléchargé. **PostgreSQL :** Assure-toi que les informations des menus sont stockées et accessibles pour générer le contenu du PDF. **JWT** : Vérifie que seul un utilisateur authentifié peut générer et télécharger un PDF.	JWT assure la sécurité de l'application, en validant les utilisateurs avant de leur donner accès à des fonctionnalités sensibles (modification, suppression, téléchargement**).**
Accès à une vue regroupant les menus créés précédemment.	Au clic sur "Mes menus", le restaurateur doit avoir accès aux menus créés précédemment La date de création s'affiche Il est possible de modifier un menu précédent Il est possible de supprimer un menu précédent Sur la même vue, le restaurateur doit pouvoir créer un nouveau menu	React Node.js avec Express PostgreSQL JWT	**React** gère l'interface utilisateur dynamique, en affichant les menus et en envoyant les requêtes API. **Node.js avec Express** traite les requêtes API et gère la logique métier côté serveur. **PostgreSQL** stocke les menus et permet de les récupérer, modifier, ajouter et supprimer de manière fiable. **JWT** sécurise l'application en authentifiant les utilisateurs avant d'effectuer des actions sensibles comme la création, modification ou suppression des menus**.**	**Gestion de l'interface utilisateur avec React** : React est responsable de l'affichage dynamique des menus créés par le restaurateur. L'utilisateur peut visualiser, ajouter, modifier ou supprimer des menus via des interactions sur la page. La date de création est également affichée pour chaque menu. **Sécurisation et gestion des données avec Node.js, PostgreSQL et JWT** : Node.js avec Express gère les requêtes pour effectuer des actions CRUD sur les menus. Express assure également la vérification du token JWT pour garantir que seules les actions autorisées sont effectuées. PostgreSQL est utilisé pour stocker et

				récupérer les menus, tandis que le JWT garantit que seules les personnes authentifiées peuvent interagir avec ces données.
Avoir accès aux tarifs de MenuMaker	La page va être créée côté Qwenta. Je n'ai pas encore l'URL, mais elle suivra la logique. https://URL_DE_QWENTA/tarifs/menumaker Succès L'internaute doit pouvoir cliquer sur l'onglet Tarifs, et ouvrir un nouvel onglet	React Node.js avec Express PostgreSQL JWT	**React :** L'utilisateur clique sur "Tarifs". Si l'utilisateur est authentifié (avec JWT), une requête est envoyée au backend. Sinon, l'utilisateur est redirigé vers la page de **connexion.** **Node.js avec Express :** Vérifie le token JWT de l'utilisateur. Si le token est valide, il renvoie l'URL des tarifs de MenuMaker. **PostgreSQL :** Permet de vérifier les droits de l'utilisateur via les données stockées (rôle, permissions, etc.). **JWT** : Assure une authentification sécurisée de bout en bout entre le frontend et le backend.	**Gestion de l'authentification et de la requête avec React** : React gère l'interaction utilisateur en permettant à l'internaute de cliquer sur l'onglet "Tarifs". Si l'utilisateur est authentifié, une requête est envoyée au backend. Sinon, il est redirigé vers la page de connexion, garantissant que seul un utilisateur authentifié peut accéder aux informations sensibles. **Vérification et gestion de la sécurité avec Node.js, Express, et JWT** : Node.js avec Express vérifie le token JWT pour s'assurer que l'utilisateur est bien authentifié. Si le token est valide, Express renvoie l'URL des tarifs de MenuMaker. PostgreSQL est utilisé pour stocker les données utilisateur, telles que les rôles et permissions, permettant ainsi de vérifier les droits d'accès. Le JWT garantit une communication sécurisée

				entre le frontend et le backend.
Mentions légales" dans une modale, et l'information “Tous droits réservés” doit être affichée.	Les éléments texte contenus dans la modale sont statiques et ne sont pas amenés à bouger pour le moment ; elle peut donc être statique. L'internaute doit pouvoir cliquer sur "Mentions légales” depuis les pages connectées et déconnectées La modale s'ouvre alors La mention "Tous droits réservés” doit figurer sur la page d'accueil et sur toutes les autres pages	Node.js avec Express React	**Node.js avec Express** héberge l'application React et expose des routes pour les API lorsque des données dynamiques sont nécessaires. **React** s'occupe d'afficher l'interface utilisateur (pages, modales, mentions légales).	**Gestion de l'affichage avec React** : React est utilisé pour afficher la modale contenant les mentions légales et la mention "Tous droits réservés" sur toutes les pages. Il s'occupe de la gestion dynamique de l'interface utilisateur, y compris l'ouverture de la modale lors du clic sur "Mentions légales". **Hébergement et gestion des routes avec Node.js et Express** : Node.js avec Express héberge l'application React et peut exposer des routes pour des données dynamiques si nécessaire. Dans ce cas, Express peut être utilisé pour gérer les requêtes liées à la

				récupération ou à l'affichage des mentions légales, bien que l'affichage soit principalement statique.
Exportation Deliveroo	⇒ Quelle connexion avec Deliveroo ? Succès L'encart "Diffuser sur Deliveroo" doit s'afficher dans la catégorie "Exportez et diffusez" Au clic sur l'encart, l'utilisateur doit être redirigé sur l'application Deliveroo	React Node.js + Express PostgreSQL JWT API Deliveroo	**React :** Affiche l'encart "Diffuser sur Deliveroo" dans l'interface utilisateur. Envoie une requête (via fetch ou axios) au serveur Express lors de certaines actions. **Node.js avec Express :** Reçoit la requête, valide l'utilisateur (via JWT). Fournit les données nécessaires ou redirige vers l'application Deliveroo si besoin. **PostgreSQL** : Stocke les données relatives aux utilisateurs, à l'encart et à l'intégration Deliveroo. Permet de gérer l'état ou les configurations nécessaires pour cette fonctionnalité. **JWT :** Valide les droits de l'utilisateur avant de lui permettre d'accéder à certaines fonctionnalités ou données.	**Gestion de l'interface avec React** : React permet d'afficher dynamiquement l'encart "Diffuser sur Deliveroo" dans l'interface, et envoie une requête au backend pour traiter l'action de redirection de l'utilisateur vers l'application Deliveroo. **Sécurisation et gestion des données avec Node.js, PostgreSQL et JWT** : Node.js avec Express valide l'utilisateur via JWT avant de lui permettre d'accéder à la fonctionnalité, PostgreSQL stocke les données relatives à l'utilisateur et l'intégration Deliveroo, et JWT assure une authentification sécurisée des actions de l'utilisateur.
Partage sur Instagram	⇒ Quelle connexion avec Instagram ?	React PostgreSQL l'API Instagram Graph	**React :** L'utilisateur clique sur "Partager sur Instagram".	**Gestion de l'interface utilisateur dynamique avec React** : React est responsable

	L'encart "Partager sur Instagram" doit s'afficher dans la catégorie "Exportez et diffusez" Au clic sur l'encart, des images du menu au format carré (pour les mettre sur Instagram) sont générées Le restaurateur est redirigé vers son compte Instagram avec les photos carrées des menus		Une requête est envoyée au backend pour générer les images. Les images générées sont affichées en aperçu. **Node.js + Express :** Récupère les données des menus depuis PostgreSQL. Génère les images des menus au format carré. Renvoie les images à React. **PostgreSQL :** Fournit les données des menus nécessaires pour personnaliser les visuels. **JWT :** Vérifie que l'utilisateur est authentifié. Assure une interaction sécurisée avec les API, y compris Instagram. **Partage vers Instagram :** Une fois les images validées par l'utilisateur, le système peut rediriger vers Instagram ou fournir une option pour télécharger les images localement.	de l'affichage de l'encart "Partager sur Instagram" et de la génération dynamique des images au format carré pour Instagram. Il permet une interaction fluide en envoyant des requêtes au backend pour récupérer les données nécessaires et afficher les images générées en aperçu. L'utilisateur peut ainsi prévisualiser les images avant de les partager sur Instagram. **Gestion des données et intégration avec l'API Instagram Graph** : **PostgreSQL** stocke les données des menus, permettant de récupérer et personnaliser les visuels des menus. **L'API Instagram Graph** permet d'automatiser le processus de partage sur Instagram, en intégrant la génération et la redirection des images vers le compte Instagram de l'utilisateur, tout en assurant une interaction sécurisée avec les API externes.
Déconnexion	Le restaurateur doit pouvoir se déconnecter depuis n'importe quelle page connectée	React Node.js avec Express.js JWT PostgreSQL	**React** : L'utilisateur clique sur un bouton "Déconnexion".	**Gestion de l'interface et de la redirection avec React** : React supprime le token JWT du stockage local/session et redirige l'utilisateur vers la

			Le frontend supprime le token JWT du stockage local/session. React redirige l'utilisateur vers la page de connexion ou une page publique. **Node.js/Express :** Si nécessaire, Express peut gérer la suppression ou l'invalidation des tokens sur le serveur. La réponse de l'API confirme que l'utilisateur a été déconnecté. **JWT :** Le token JWT est supprimé sur le client, ce qui empêche toute demande future avec ce token d'être authentifiée. **PostgreSQL :** Si une session est stockée en base de données, elle peut être mise à jour ou supprimée.	page de connexion, assurant ainsi une déconnexion fluide. **Sécurisation de la session et gestion des données avec JWT et PostgreSQL** : JWT empêche toute utilisation future du token pour des requêtes authentifiées, et PostgreSQL peut être utilisé pour supprimer ou mettre à jour les sessions si elles sont stockées dans la base de données.
Modifier mes informations utilisateur	Le restaurateur doit pouvoir : Lier plusieurs adresses e-mail à son compte Modifier son adresse e-mail de base	React JWT Node.js avec Express PostgreSQL	**React :** L'utilisateur se connecte et modifie ses informations via un formulaire. **JWT** : Lors de la connexion, un token JWT est généré et envoyé au frontend. **Node.js avec Express :** Le backend vérifie le token JWT, authentifie l'utilisateur, puis effectue les modifications dans la	**Gestion de l'interface et des interactions avec React** : React permet à l'utilisateur de modifier ses informations via un formulaire, tout en mettant à jour l'interface en temps réel. **Sécurisation et gestion des données avec JWT et PostgreSQL** : JWT assure l'authentification sécurisée de l'utilisateur lors des

			base de données PostgreSQL. **PostgreSQL :** Les modifications (comme l'ajout ou la mise à jour d'adresses e-mail) sont stockées dans la base de données.	modifications, tandis que PostgreSQL stocke les informations modifiées (adresses e-mail) dans la base de données.
Accès à un Dashboard	L'internaute doit pouvoir avoir accès : À l'encart "Créer un menu" À l'encart "Diffuser un menu" À l'encart "Imprimer un menu" À la section "Pour aller plus loin" Aux 3 derniers articles de blog qui parlent de MenuMaker Titre Photo de couverture Lien	React Node.js avec Express PostgreSQL JWT	**React :** Crée l'interface utilisateur, affiche les différentes sections et articles de blog, et interagit avec l'API via des requêtes HTTP. **Node.js avec Express :** Gère les requêtes du frontend (par exemple, la création d'un menu ou la récupération des articles de blog) et fournit des réponses appropriées en interagissant avec la base de données. **PostgreSQL :** Stocke toutes les informations sur les menus, les articles de blog, et les utilisateurs dans des tables et permet d'effectuer des recherches et des mises à jour. **JWT :** Assure que seules les personnes authentifiées peuvent accéder à des ressources protégées du backend (comme créer un	**Gestion de l'interface utilisateur avec React** : React crée une interface dynamique qui permet d'afficher les différentes sections et articles du Dashboard, tout en interagissant avec l'API pour récupérer et afficher les données. **Sécurisation et gestion des données avec Node.js, PostgreSQL et JWT** : Node.js avec Express gère les requêtes du frontend et interagit avec PostgreSQL pour récupérer et stocker les informations. JWT assure que seuls les utilisateurs authentifiés ont accès aux fonctionnalités protégées du Dashboard.

			menu, accéder aux articles réservés, etc.).	
Créer le Branting Restaurateur	Le restaurateur doit pouvoir ajouter / modifier / supprimer les éléments suivants : Logo Couleurs de base	React Node.js avec Express PostgreSQL JWT	**React :** Crée l'interface utilisateur, affiche les différentes sections et articles de blog, et interagit avec l'API via des requêtes HTTP. **Node.js avec Express :** Gère les requêtes du frontend (par exemple, la création d'un menu ou la récupération des articles de blog) et fournit des réponses appropriées en interagissant avec la base de données. **PostgreSQL :** Stocke toutes les informations sur les menus, les articles de blog, et les utilisateurs dans des tables et permet d'effectuer des recherches et des mises à jour. **JWT :** Assure que seules les personnes authentifiées peuvent accéder à des ressources protégées du backend (comme créer un menu, accéder aux articles réservés, etc.).	**Gestion dynamique de l'interface avec React** : React permet au restaurateur de télécharger un logo et de sélectionner des couleurs de manière intuitive, puis d'envoyer ces données via un formulaire au backend pour leur traitement et leur mise à jour. **Validation et sécurisation avec Node.js, PostgreSQL et JWT** : Node.js avec Express gère les requêtes de mise à jour du branding, en s'assurant que l'utilisateur est authentifié via JWT et en enregistrant les nouveaux éléments dans PostgreSQL pour garantir la persistance des données.

2. Liens avec le back-end

Avec l’aide de mes recherche de veille je pense qu’afin de créer une expérience fluide et sécurisée pour l'utilisateur, chaque

fonctionnalité du site, comme la gestion du menu, l'authentification, et la personnalisation de l'interface, utilise un stack technologique robuste incluant React pour l'interface utilisateur, Node.js avec Express pour la gestion des requêtes backend, PostgreSQL pour le stockage des données, et JWT pour sécuriser les actions des utilisateurs, assurant ainsi une modularité, une évolutivité et une sécurité optimales.

3. Préconisations concernant le domaine et l'hébergement

Le nom de domaine sera très probablement un sous-domaine de Qwenta en cours de validation.

4. Accessibilité

Le site respecte dans l'ensemble les normes WCAG 2.1, niveau AA, afin de garantir une meilleure accessibilité pour les personnes en situation de handicap. Les consignes spécifiques à respecter sont les suivantes :

- L'application doit être navigable au clavier et compatible avec les lecteurs d'écran.
- Pour le moment, la compatibilité est requise uniquement avec les dernières versions des navigateurs Chrome, Safari et Firefox.

De plus, il est indispensable de :

1. Prévoir des tests utilisateurs avec des personnes en situation de handicap pour identifier et corriger d'éventuels problèmes d'accessibilité.
2. Planifier des tests automatisés afin de simuler différents navigateurs et appareils.
3. Effectuer des vérifications sur des appareils réels pour les scénarios critiques.

5. Recommandations en termes de sécurité

Avec l'aide de mes recherches de veille, je pense qu'afin de créer une expérience fluide et sécurisée pour l'utilisateur, chaque fonctionnalité du site, comme la gestion du menu, l'authentification et la personnalisation de l'interface, utilise une stack technologique robuste. Celle-ci inclut React pour l'interface utilisateur, Node.js avec Express pour la gestion des requêtes backend, PostgreSQL pour le stockage des données, et JWT pour sécuriser les actions des utilisateurs, garantissant ainsi modularité, évolutivité et sécurité optimales.

6. Maintenance du site et futures mises à jour

Pas de version mobile à développer ni à prévoir. La stratégie d'acquisition ne prévoit pas d'exigences particulières en termes de SEO. Ajouter des animations sur la photo de la bannière + sur les formes géométriques des sections.
Il n'y aura pas d'outil à intégrer pour capter le comportement des utilisateurs pour le moment.