Webgencia

Spécifications techniques

Menu Maker by Qwenta

Version	Auteur	Date	Approbation
1.0	Webgencia	07/05/2024	John, Qwenta

I. Choix technologiques

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

Besoin	Contraintes	Solution	Description de la solution	Justification (2 arguments)
Création d'une catégorie de menu	L'ajout d'une catégorie doit pouvoir se faire directement sur l'écran de création de menu depuis une modale.	*react-modal*	Cette librairie React permet de créer simplement des modales performantes, accessibles avec un minimum de code.	1) Nous avons choisi de développer en React, la librairie est cohérente avec ce choix. 2) Il s'agit de la librairie la plus utilisée.
Connection	L'utilisateur doit pouvoir se connecter	*Firebase Authentication*	Firebase Authentication fournit une solution d'authentification robuste et sécurisée pour les applications web et mobiles.	1) Intégration facile avec React grâce à la bibliothèque Firebase. 2) Offre des options d'authentification multiplateforme.
Créer un menu	Pour créer son menu, le restaurateur pourra ajouter : - le nom des plats qui le composent par catégorie (entrée, plat ou dessert) ; - le prix du plat ; - la description du plat.	*react-hook-form*	React Hook Form est une bibliothèque légère et performante pour la gestion des formulaires en React. Elle permet une validation facile et une gestion optimisée des champs de formulaire	1) Facilite la création de formulaires dynamiques et complexes. 2) Améliore les performances en minimisant les rendus inutiles.

Personnaliser un menu	Les restaurateurs pourront : - enregistrer leurs préférences de sa stratégie de marque, notamment pour enregistrer le logo du restaurant ; - choisir de façon dynamique la police et sa couleur.	*styled-components*	Styled Components permet de créer des composants React avec des styles CSS encapsulés, offrant ainsi une modularité et une maintenabilité accrues	1) Encapsulation des styles pour éviter les fuites de style. 2) Supporte la gestion dynamique des styles basée sur les props
Sauvegarder un menu	Une fois le menu prêt, il est possible de le sauvegarder pour une utilisation future	*Firebase Realtime Database*	Firebase Realtime Database offre une solution de stockage de données en temps réel qui permet de sauvegarder les menus de manière efficace et sécurisée. En utilisant Firebase Realtime Database, chaque menu peut être enregistré sous forme de données structurées dans le cloud, offrant une synchronisation instantanée entre les clients et le serveur.	1) Fiabilité : Firebase Realtime Database offre une fiabilité élevée grâce à sa capacité à gérer les sauvegardes et les restaurations de données de manière transparente, garantissant ainsi la disponibilité des menus sauvegardés en tout temps. 2) Scalabilité : Avec Firebase Realtime Database, la capacité de stockage des menus peut facilement s'adapter à la croissance de l'application, offrant une solution scalable et flexible pour répondre aux besoins futurs de sauvegarde des menus.

Diffuser un menu	Une fois le menu prêt, il est possible de : - l'exporter en PDF ; - le diffuser sur Deliveroo ; - le partager sur Instagram.	*React PDF, Deliveroo API, Instagram API*	1) React PDF : permet de générer des fichiers PDF à partir de composants React. 2) Deliveroo API : offre des fonctionnalités pour intégrer directement les menus dans l'application Deliveroo. 3) Instagram API : permet de partager automatiquement les menus sur Instagram.	1) Assure une expérience utilisateur cohérente en proposant plusieurs options de partage. 2) Facilite la diffusion du menu sur différentes plateformes populaires.
Imprimer un menu	Une fois le menu prêt, il est également possible commander son impression.	*Utiliser React avec une intégration email en backend.*	1) Bouton "Imprimer mon menu" : Un bouton sera intégré à l'interface utilisateur, permettant aux utilisateurs de déclencher le processus d'impression de leur menu. 2) Intégration email en backend : Lorsque l'utilisateur appuie sur le bouton "Imprimer mon menu", une requête sera envoyée au backend. En backend, une fonction sera déclenchée pour envoyer un email à l'équipe chargée de l'impression du menu. Cet email pourrait contenir une notification automatique indiquant la demande d'impression du menu et les détails pertinents (tel que le contenu du menu ou les instructions spécifiques).	1) Simplicité d'utilisation : En intégrant un bouton dédié sur l'interface utilisateur, les utilisateurs peuvent facilement exprimer leur souhait d'imprimer leur menu sans avoir à chercher d'autres options complexes. Cela simplifie le processus pour les utilisateurs et encourage leur participation. 2) Automatisation et traçabilité : En utilisant une intégration email en backend, le processus d'envoi de la demande d'impression du menu est automatisé, ce qui réduit la charge de travail manuel pour l'équipe en charge de l'impression. De plus, l'email envoyé fournit une trace écrite de la demande, assurant ainsi une meilleure traçabilité et réduisant les risques d'erreurs ou d'omissions.

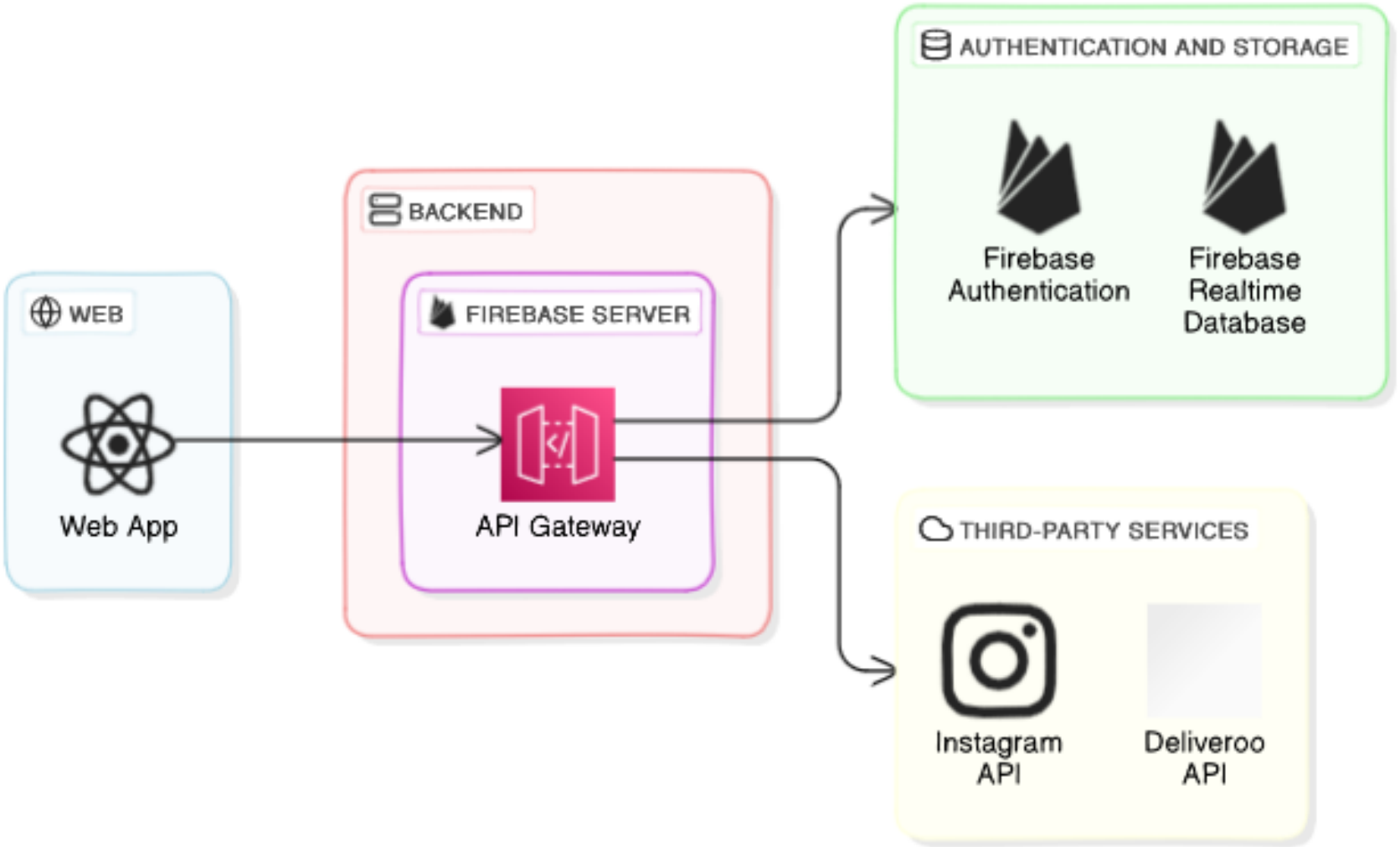
II. Architecture de l'application

L'ensemble de l'architecture de notre application web est construit autour des technologies suivantes :

- **Frontend avec ReactJS :** Pour offrir une expérience utilisateur fluide et réactive, nous utilisons ReactJS pour développer l'interface utilisateur de notre application.
- **Backend avec NodeJS et Express :** NodeJS agit comme le moteur principal de notre backend, orchestrant toutes les requêtes et les opérations de données. Nous utilisons Express pour simplifier la création des routes et la gestion des requêtes HTTP, garantissant ainsi une architecture robuste et évolutive.
- **Services Firebase :** Nous intégrons Firebase Authentication pour gérer l'authentification des utilisateurs de manière sécurisée. Cette solution offre une variété d'options d'authentification flexibles, notamment par e-mail/mot de passe et par l'intermédiaire de fournisseurs tiers tels que Google, Facebook et Twitter. Firebase Realtime Database est utilisé pour sauvegarder les menus des utilisateurs en temps réel, offrant ainsi une synchronisation efficace des données entre les clients et le backend.
- **Hébergement avec Firebase Hosting :** Pour assurer la disponibilité en ligne de notre application, nous utilisons Firebase Hosting. Cette solution offre une plateforme fiable et évolutive pour héberger notre application web.
- **Intégration avec des API tierces via le backend :** Notre backend interagit avec différentes API tierces :
  - Il utilise l'API de Firebase Authentication pour gérer l'identification des utilisateurs.
  - Il vérifie l'authentification à l'aide du jeton JWT avec l'API de Firebase Authentication.
  - Il communique avec les API d'Instagram et de Deliveroo pour la publication des menus créés sur ces plateformes.

(cf. diapo suivante pour voir un diagramme explicite)

Web Application Architecture



III. Préconisations concernant le domaine et l'hébergement

- Le nom de domaine sera très probablement un sous-domaine de Qwenta.
- Nous prévoyons d'utiliser tous les services offerts par Firebase, dont **Firebase Authentication** pour la gestion sécurisée de l'authentification des utilisateurs, **Firebase Realtime Database** pour l'hébergement de la base de données en temps réel, et **Firebase Hosting** pour l'hébergement de l'application elle-même. *Cette approche garantira non seulement la cohérence de notre infrastructure, mais également bénéficiera de toutes les qualités offertes par la plateforme Firebase, telles que la rapidité, la sécurité, la scalabilité automatique, la facilité de déploiement, la surveillance en temps réel et la gestion simplifiée des mises à jour.*

IV. Accessibilité

- Le site devra être en **version desktop**. Pas de version mobile à développer ni à prévoir.
- Pour le moment, on se contente de la **compatibilité avec les dernières versions** de Chrome, Safari et Firefox.
- L'application devra être accessible au minimum : **navigable depuis le clavier**, et **lisible par un lecteur d'écran**.

V. Recommandations en termes de sécurité

- Accès aux comptes et gestion des mots de passe :

- Nous utiliserons Firebase Authentication pour gérer l'authentification des utilisateurs, ce qui inclut la gestion sécurisée des mots de passe, la création de comptes utilisateur et la gestion des sessions d'utilisateur.
- Nous configurerons des politiques de mot de passe robustes via Firebase Authentication, en encourageant l'utilisation de mots de passe forts et la mise en place de l'authentification à deux facteurs lorsque cela est possible.
- Nous mettrons en place des mécanismes de gestion des comptes utilisateur via Firebase Authentication, comme la désactivation des comptes inactifs ou la limitation des privilèges d'accès en fonction des besoins de chaque utilisateur.

- Plugins :

- Nous sélectionnerons avec soin les plugins à utiliser dans l'application, en privilégiant ceux qui sont régulièrement mis à jour et dont la sécurité est vérifiée de manière régulière.
- Nous mettrons en place un processus de surveillance et de mise à jour régulière des plugins afin de nous assurer qu'ils ne présentent pas de vulnérabilités de sécurité connues.

VI. Maintenance du site et futures mises à jour

- Veille technologique :
  - Maintenir une veille technologique active pour suivre les dernières tendances, les nouvelles fonctionnalités et les meilleures pratiques en matière de développement web.
  - Participer à des forums de discussion, lire des blogs spécialisés et suivre les mises à jour des technologies utilisées dans l'application pour rester informé des évolutions du secteur.
- Contrôles et mises à jour régulières :
  - Effectuer des contrôles réguliers du code source et de l'hébergement du serveur pour détecter et corriger les éventuelles vulnérabilités de sécurité.
  - Mettre en place un processus de gestion des mises à jour régulières pour garantir que toutes les composantes de l'application, y compris les dépendances tierces, sont maintenues à jour avec les derniers correctifs de sécurité.
- Surveillance continue :
  - Utiliser des outils de surveillance tels que LightHouse pour évaluer périodiquement les performances et la qualité de l'application, en identifiant les zones à améliorer et en suivant les progrès au fil du temps.
  - Configurer des sauvegardes automatiques régulières (snapshots) du site pour pouvoir restaurer rapidement en cas de problème majeur ou de perte de données.
  - Examiner régulièrement les logs du serveur pour détecter les anomalies, les erreurs ou les tentatives d'exploitation et prendre les mesures nécessaires pour y remédier.