ChatGPT

Cahier des Charges – Tunnel de Paiement pour site de location de bennes

Contexte et objectifs

Le projet consiste à développer un **tunnel de paiement complet** pour un site de location de bennes à destination des professionnels et des particuliers. Ce tunnel doit être intégré dans une application React hébergée sur Vercel, avec une architecture modulaire permettant d'ajouter ou modifier facilement des prestataires de paiement (Stripe, PayPal, etc.). L'objectif est d'offrir aux utilisateurs un parcours de réservation fluide (avec ou sans création de compte), de la sélection du service à la confirmation de paiement, tout en garantissant la conformité aux normes de sécurité et de protection des données personnelles (RGPD). L'interface doit être responsive (adaptée aux mobiles et tablettes) et la solution globalement **scalable** pour supporter la croissance du trafic.

Périmètre fonctionnel – Tunnel de paiement

Le tunnel de paiement prend en charge l'ensemble des étapes de commande sur le site : du choix du service à la réception du paiement et de la confirmation. Les principales fonctionnalités attendues sont :

- **Choix du service :** l'utilisateur sélectionne un type de benne (volume, type de déchet autorisé, durée/location) parmi les prestations disponibles. L'architecture doit être **modulaire** : les prestations (natures de bennes, durées, tarifs) sont gérées dynamiquement via une interface d'administration (ajout/modification/suppression).
- **Calcul du prix dynamique :** en fonction des paramètres choisis (type de benne, durée, distance de livraison éventuelle), le système calcule automatiquement le montant à payer. La grille tarifaire doit pouvoir être mise à jour via l'admin.
- **Saisie de l'adresse :** l'utilisateur renseigne l'adresse de livraison/retrait de la benne. Le formulaire adresse peut être enrichi (auto-complétion Google Places ou équivalent) pour éviter les erreurs de saisie.
- **Choix du créneau :** l'utilisateur sélectionne une date et un créneau horaire disponibles pour la livraison ou la récupération de la benne. Les créneaux doivent être gérés dynamiquement (planning des équipes, disponibilités) via l'administration.
- **Récapitulatif de commande :** avant le paiement, l'utilisateur vérifie le récapitulatif : détails du service choisi, adresse, créneau, et montant total. Il peut revenir en arrière modifier ses choix.
- **Paiement sécurisé :** l'utilisateur choisit son moyen de paiement (Stripe pour carte bancaire, PayPal, etc.) et effectue la transaction. Le système prend en charge le *mode sandbox* pour les tests, puis le *mode production* avec des clés API sécurisées. Par exemple, une fonction serverless Vercel peut créer une session Stripe Checkout puis rediriger l'utilisateur vers la page de paiement Stripe [1]. Pour PayPal, l'utilisateur est redirigé vers l'interface PayPal pour autoriser le paiement, puis revient sur le site [2]. Le tunnel doit gérer le paiement en entier, notamment en recevant et traitant les *webhooks* ou notifications de paiement réussis ou échoués.
- **Confirmation de commande :** une fois le paiement validé, l'utilisateur voit une page de confirmation et reçoit immédiatement un e-mail de confirmation (avec récapitulatif, numéro de commande, etc.). Le statut de la commande passe alors à « Confirmée ».

Notifications par e-mail : À chaque étape clé, un e-mail automatique est envoyé au client (prospect enregistré ou non) et aux équipes internes : - Confirmation de commande (immédiatement après paiement réussi).
- Notification de changement de statut (ex. la benne est en route, la commande est terminée, etc.).
- Rappel (ex. veille du créneau réservé, pour éviter les oublis).
Ces e-mails doivent être configurables (objet, contenu) et gérés par le back-end (SMTP ou service mail tiers).

Gestion des utilisateurs et parcours

- **Réservation avec compte :** si l'utilisateur a un compte (professionnel ou particulier), il peut se connecter et retrouver ses informations (historique des commandes, adresses enregistrées). L'interface client propose un espace personnel pour modifier ses coordonnées, consulter ses commandes passées et lancer une nouvelle réservation.
- **Réservation sans compte :** un client non inscrit peut réserver en tant qu'invité. Il fournit ses coordonnées (nom, email, téléphone, adresse) durant le tunnel de commande. À la fin, il peut recevoir la proposition de création de compte pour garder une trace de sa commande.
- **Responsive UI :** l'application React doit être responsive (tests sur mobiles/tablettes), avec une interface claire et épurée. Les formulaires doivent guider l'utilisateur (validation en temps réel des champs d'adresse, sélection de date/heure intuitive, etc.).

Tableau des rôles utilisateurs

Rôle	Actions clés	Accès
Particulier / Professionnel (client)	Passer commande, payer en ligne, consulter l'historique des commandes, modifier son compte (si inscrit)	Interface publique (tunnel de commande, espace client)
Administrateur (interne)	Gérer commandes (statuts, modifications, export), gérer utilisateurs, suivre paiements (journal, remboursements), configurer prestations et créneaux, consulter le dashboard KPI	Panneau d'administration sécurisé (back-office)

Panneau d'administration

Le back-office permet aux équipes internes de gérer l'activité et les données métier. Les fonctionnalités attendues sont :

- **Gestion des commandes :** liste des commandes avec filtres (date, statut, client, etc.), recherche textuelle. Pour chaque commande : consultation du détail (client, adresse, prestation, paiement), possibilité de modifier le statut (par ex. « Confirmée », « En cours », « Terminée », « Annulée »), de corriger les données (adresse, créneau), d'enregistrer des notes internes. Les opérateurs peuvent également exporter la liste des commandes (CSV/XLSX) et suivre l'historique des changements de statut.
- **Gestion des utilisateurs :** liste des utilisateurs (pros, particuliers), avec filtrage et recherche. Possibilité de consulter/modifier les comptes, réinitialiser les mots de passe, désactiver un compte (par exemple sur demande ou pour gestion de litiges). Le panneau doit afficher le rôle (client, admin) et l'historique des commandes de chaque client.

- **Gestion des paiements :** vue complète du journal des transactions par commande : statuts de paiement, montants, références Stripe/PayPal. Possibilité de lancer des remboursements partiels ou totaux via l'interface (en utilisant l'API Stripe ou PayPal). Détection et signalement des anomalies de paiement (échecs, transactions suspectes).
- **Dashboard KPI :** tableau de bord avec indicateurs clés en temps réel : *Nombre de ventes*, *chiffre d'affaires* (CA) cumulé, *volume de bennes louées*, *panier moyen*, etc. Ces KPI aident à suivre la performance commerciale. Des graphiques simples (courbes de CA par période, répartition clients pro/particuliers, etc.) sont appréciés.
- **Gestion des prestations :** interface pour configurer les offres : création/modification des types de benne (capacité, description), tarifs associés (formule de prix par jour, par km, etc.), types de déchet autorisés, quantités maximales. On peut également gérer les créneaux et périodes de non-disponibilité (ex. jours fériés). Ces données impactent directement les étapes du tunnel de commande.

Architecture technique

Composant	Technologie / Outil	Description
Front-end	React (Next.js sur Vercel)	Application client en React, routée via Next.js (pour SSR/optimisation SEO éventuelle), hébergée sur Vercel. L'interface utilisateur gère le tunnel de commande, l'espace client et le site public.
Back-end / API	Node.js (Vercel Functions)	API RESTful (ou GraphQL au choix) sécurisée, déployée en serverless sur Vercel. Gère la logique métier : créations de commandes, calculs tarifs, gestion utilisateurs, communication avec les prestataires de paiement. Utilise JWT pour authentification.
Base de données	PostgreSQL ou équivalent	Stockage des données métier (utilisateurs, commandes, prestations, créneaux, transactions). Hébergée sur un service cloud (ex. AWS RDS, Supabase, PlanetScale, etc.) pour la persistance. Les accès sont sécurisés via des variables d'environnement sur Vercel.
Passerelles de paiement	Stripe et PayPal (API)	Intégration des API Stripe et PayPal pour traiter les paiements. L'application backend crée des sessions de paiement et reçoit des webhooks. Le front-end redirige vers Stripe Checkout ou PayPal Checkout selon le choix.
Notifications e-mail	Service SMTP / API (SendGrid, Mailgun…)	Envoi automatisé d'e-mails transactionnels à chaque étape clé (confirmation, modification, rappel). Peut utiliser un service cloud d'e-mails pour la fiabilité et la délivrabilité.
Authentification	JWT (JSON Web Tokens)	Gestion sécurisée de la connexion utilisateur. Après authentification (login), le serveur génère un token JWT envoyé au client pour les requêtes suivantes. Les mots de passe sont hachés (bcrypt) dans la base.

Composant	Technologie / Outil	Description
Sécurité / SSL	HTTPS, variables d'env.	Le site est servi en HTTPS (certificat SSL/TLS), les clés API (Stripe, PayPal) sont stockées en variables d'environnement sur Vercel. Les cookies sensibles sont sécurisés et le trafic est chiffré.

Vercel est choisi pour l'hébergement car il offre des **fonctions serverless** adaptées aux applications React, avec mise à l'échelle automatique en fonction de la charge [3]. Les fonctions Vercel facilitent la connexion aux bases de données et aux APIs tierces (Stripe, PayPal) sans gestion de serveur dédié.

Sécurité et conformité RGPD

- **Conformité RGPD :** Le site doit respecter le RGPD (UE) et la CNIL. Un bandeau de consentement aux cookies est obligatoire : les cookies non essentiels (analytics, traceurs tiers) ne sont déposés qu'après accord explicite de l'utilisateur [4]. La politique de confidentialité (accessible depuis le pied de page) explique quelles données sont collectées et dans quel but. L'utilisateur doit pouvoir exercer ses droits (accès, rectification, effacement). En particulier, le site doit permettre la **suppression des données personnelles** sur demande (droit à l'effacement, article 17 RGPD) [5]. Les données sensibles (informations de paiement) ne doivent jamais être stockées en clair par notre système (Stripe ou PayPal gèrent en partie la tokenisation des cartes). Les mots de passe sont hachés (bcrypt) et seuls les jetons d'accès (JWT) circulent côté client après connexion.
- **Sécurité des paiements :** Toutes les pages du tunnel de paiement doivent être servies en HTTPS. Les clés secrètes Stripe/PayPal sont stockées côté serveur uniquement. Les webhooks (Stripe ou PayPal) doivent être validés (signature, secret) pour empêcher les notifications frauduleuses [6]. L'architecture est conçue pour ne pas exposer directement le système aux données bancaires : on privilégie les solutions hébergées (Stripe Checkout, PayPal Checkout) qui gèrent le PCI DSS.
- **Stockage sécurisé :** Les données critiques (numéros de commande, montants, emails) sont en base cloud sécurisée. Les journaux d'accès et de paiement sont conservés avec contrôle d'accès restreint. Les backups réguliers sont effectués pour éviter toute perte de données.
- **Authentification et autorisations :** Les administrateurs accèdent à l'interface back-office via un login sécurisé (login/MDB). Un système d'authentification basé sur JWT assure que seules les personnes autorisées (rôle « admin ») peuvent consulter et modifier les données internes. Des contrôles d'accès au niveau des routes API empêchent toute lecture ou écriture non autorisée des ressources.

Exigences non fonctionnelles et architecture

- **Scalabilité :** Le système doit pouvoir monter en charge (nombre de visites et de commandes élevé) grâce à l'infrastructure serverless. Vercel s'occupera de déployer dynamiquement des instances des fonctions selon la demande [3]. La base de données utilisée doit également être scalable (ex : PostgreSQL managé ou base NoSQL si besoin).
- **Performance :** Réduction du temps de réponse par l'utilisation de mises en cache (HTTP caching, CDN intégrée avec Vercel). Les images et ressources sont optimisées. Le front-end React (Next.js) génère certaines pages en SSR (rendu serveur) pour le SEO et la rapidité.
- **API REST/GraphQL :** L'API de l'application exposera des endpoints clairs (par ex. `/api/commandes`, `/api/utilisateurs`, `/api/prestations`). Les endpoints sensibles nécessitent une authentification JWT. GraphQL est possible pour regrouper des requêtes

complexes (ex : chargement simultané de commandes+utilisateur), mais il faut sécuriser le schéma.

- **Journalisation et monitoring :** Toutes les actions critiques sont journalisées (création/modification de commande, paiements, connexions admin). Un système de logs et d'alertes (via Vercel ou un outil tiers) surveille les erreurs serveurs ou échecs de paiement.
- **Qualité et tests :** Le code doit suivre les bonnes pratiques (linting, formatage). Des tests unitaires et d'intégration sont mis en place pour valider le calcul de prix, le workflow de commande, et l'intégration Stripe/PayPal (en mode sandbox).

Schéma d'architecture (exemple simplifié)

Composant	Technologie	Rôle
Front-End UI	React / Next.js	Interface utilisateur (tunnel de commande, pages publiques, espace client)
API Back-End	Node.js (Vercel)	Logique métier, base de données, intégration Stripe/PayPal, gestion utilisateurs
Base de données	PostgreSQL / Supabase	Stockage utilisateurs, commandes, prestations, journaux
Paiement	Stripe, PayPal	Gestion des transactions (sessions de paiement, webhooks)
Emails	SendGrid / SMTP	Envoi des notifications (confirmation, rappels)
Auth / JWT	JSON Web Token	Gestion sécurisée des sessions utilisateur/admin

Livrables attendus

- **Spécifications détaillées** (ce document cahier des charges).
- **Maquettes UI/UX** (wireframes ou prototypes du tunnel de commande et du back-office).
- **Documentation technique** : description de l'architecture, listes des API, clés de configuration.
- **Code source** structuré : dépôt Git avec branches de développement, tests automatisés, intégration continue.
- **Déploiement** : pipeline CI/CD sur Vercel (pré-production et production), configuration des variables d'environnement (clés Stripe/PayPal en sandbox et prod).

Les éléments ci-dessus forment un cahier des charges complet, prêt à être remis à l'équipe de développement. Il décrit le périmètre fonctionnel et technique du projet et respecte les contraintes (RGPD, sécurité, scalabilité) pour la mise en œuvre d'un tunnel de paiement robuste et évolutif.

Sources : Exemples d'intégration Stripe/React sur Vercel [1], fonctionnement de PayPal Checkout pour SPA [2], recommandations RGPD CNIL (consentement cookies [4], droit à l'effacement [5]), documentation Vercel sur les fonctions serverless [3].

---

[1] Create a React app with Stripe and Vercel
https://divjoy.com/guide/react-stripe-vercel

[2] Acceptez des paiements PayPal | Stripe
https://stripe.com/fr/payment-method/paypal

3 Vercel Functions
https://vercel.com/docs/functions

4 Cookies et traceurs : que dit la loi ? | CNIL
https://www.cnil.fr/fr/cookies-et-autres-traceurs/que-dit-la-loi

5 Fichiers informatiques et données personnelles | Service-Public.fr
https://www.service-public.fr/particuliers/vosdroits/F2024

6 Receive Stripe events in your webhook endpoint | Stripe Documentation
https://docs.stripe.com/webhooks