Guide de Deploiement — Epicerie Africaine

Version : 1.0 **Date :** Fevrier 2026 **Auteur :** Equipe technique **Confidentialite :** Document interne — Ne pas partager avec le client final

---

Table des matieres

1. Prerequis techniques
2. Architecture du projet
3. Installation locale
4. Configuration Sanity (CMS)
5. Configuration Supabase (Auth + BDD)
6. Configuration Stripe (Paiements)
7. Configuration Cal.com (Rendez-vous)
8. Alimenter le catalogue (Sanity Studio)
9. Tests fonctionnels
10. Deploiement en production
11. Personnalisation pour un nouveau client
12. Maintenance et operations courantes
13. Checklist de livraison
14. Depannage (FAQ)

---

1. Prerequis techniques

Logiciels requis

Logiciel	Version minimum	Lien de telechargement
Node.js	18.x ou 20.x	https://nodejs.org
npm	9.x+ (inclus avec Node)	—
Git	2.x+	https://git-scm.com
Stripe CLI	Derniere version	`scoop install stripe` (Windows)
Navigateur moderne	Chrome/Edge/Firefox	—
Editeur de code	VS Code recommande	https://code.visualstudio.com

Comptes a creer (gratuits)

Service	URL	Role
Sanity.io	https://www.sanity.io	CMS (gestion produits, categories)
Supabase	https://supabase.com	Auth + Base de donnees

Stripe	https://dashboard.stripe.com	Paiements en ligne
Cal.com	https://cal.com	Prise de rendez-vous (optionnel)
Vercel	https://vercel.com	Hebergement (recommande)

Verifier l'installation

Ouvrir un terminal et executer :

```
node --version    # Doit afficher v18.x.x ou v20.x.x
npm --version     # Doit afficher 9.x.x+
git --version     # Doit afficher git version 2.x.x
```

---

2. Architecture du projet

```
my-shop/
├── apps/
│   ├── web/                    # Application Next.js (frontend + API)
│   │   ├── app/                # Pages et routes (App Router)
│   │   ├── components/         # Composants React reutilisables
│   │   ├── lib/                # Clients (Sanity, Supabase, Stripe)
│   │   ├── hooks/              # Hooks React personnalises
│   │   ├── types/              # Types TypeScript
│   │   ├── styles/             # CSS global + config Tailwind
│   │   ├── public/             # Fichiers statiques (images, favicon)
│   │   ├── .env.local          # Variables d'environnement (SECRETS)
│   │   └── .env.example        # Template des variables (sans secrets)
│   │
│   └── studio/                 # Sanity Studio (back-office CMS)
│       ├── schemaTypes/        # Schemas : Product, Category, Order
│       ├── .env                # Config Sanity (project ID, dataset)
│       └── sanity.config.ts    # Configuration du studio
│
├── supabase/
│   └── migrations/             # Scripts SQL (tables, RLS, indexes)
│
├── seed/                       # Script de peuplement Sanity
└── docker/                     # Configuration Docker (optionnel)
```

Roles de chaque service

Service	Gere quoi	Interface admin
Sanity Studio	Produits, categories, images, descriptions, tags, cross-sell	`localhost:3333`

Supabase	Comptes clients, paniers persistants, commandes, avis, fidelite	Dashboard web
Stripe Dashboard	Paiements, remboursements, factures, litiges	Dashboard web
Cal.com	Creneaux RDV, confirmations, annulations	Dashboard web

---

3. Installation locale

Etape 3.1 — Cloner le projet

```
git clone <URL_DU_REPO> my-shop
cd my-shop
```

Etape 3.2 — Installer les dependances

```
# Application web
cd apps/web
npm install

# Studio Sanity
cd ../studio
npm install
```

Etape 3.3 — Creer les fichiers d'environnement

```
# Application web
cd apps/web
cp .env.example .env.local

# Studio Sanity
cd ../studio
cp .env.example .env
```

> ***SECURITE** : Les fichiers `.env.local` et `.env` contiennent des secrets. Ils sont deja dans `.gitignore` — ne JAMAIS les commiter dans Git. Ne JAMAIS partager ces fichiers par email, Slack ou tout autre canal non securise.*

Etape 3.4 — Verifier le .gitignore

Ouvrir `.gitignore` a la racine et confirmer la presence de :

```
.env
.env.local
.env.development.local
.env.test.local
.env.production.local
```

---

4. Configuration Sanity (CMS)

Sanity gere le catalogue de produits (noms, prix, images, categories, descriptions). Sans Sanity configure, l'application affiche des produits de demonstration (mock data).

Etape 4.1 — Creer un compte Sanity

1. Aller sur **https://www.sanity.io**
2. Cliquer **"Get started"** puis **"Sign up"**
3. Creer un compte (Google, GitHub, ou email)

Etape 4.2 — Creer un nouveau projet Sanity

1. Aller sur **https://www.sanity.io/manage**
2. Cliquer **"Create new project"**
3. Remplir :
   - **Project name** : Nom du client (ex: `Epicerie Africaine - Client XYZ`)
   - **Use the default dataset configuration** : Oui (cela cree un dataset `production`)
4. **Noter le Project ID** affiche en haut (ex: `abc12xyz`) — on en aura besoin

Etape 4.3 — Configurer les CORS Origins

1. Dans le dashboard Sanity (https://www.sanity.io/manage)
2. Selectionner le projet
3. Aller dans **API > CORS Origins**
4. Ajouter ces origines :
   - `http://localhost:3007` (developpement)
   - `http://localhost:3333` (studio local)
   - `https://votre-domaine.com` (production — a ajouter plus tard)
5. Pour chaque origine, cocher **"Allow credentials"**

Etape 4.4 — Creer un token API

1. Dans le dashboard Sanity > **API > Tokens**
2. Cliquer **"Add API token"**
3. Remplir :
   - **Token name** : `Next.js App`
   - **Permissions** : **Editor** (lecture + ecriture)
4. Cliquer **"Save"**
5. **COPIER LE TOKEN IMMEDIATEMENT** — il ne sera plus visible apres

> ***ATTENTION*** *: Ce token donne acces en ecriture a tout le contenu. Ne JAMAIS l'exposer cote client. Il est utilise uniquement cote serveur.*

Etape 4.5 — Remplir les variables d'environnement

Fichier `apps/web/.env.local` :

```
NEXT_PUBLIC_SANITY_PROJECT_ID=votre_project_id_ici
NEXT_PUBLIC_SANITY_DATASET=production
NEXT_PUBLIC_SANITY_API_VERSION=2024-01-01
SANITY_API_TOKEN=votre_token_api_ici
```

Fichier `apps/studio/.env` :

```
SANITY_STUDIO_PROJECT_ID=votre_project_id_ici
SANITY_STUDIO_DATASET=production
```

Etape 4.6 — Mettre a jour la config du studio

Ouvrir `apps/studio/sanity.config.ts` et verifier que le `projectId` correspond :

```
export default defineConfig({
  name: 'default',
  title: 'Epicerie Africaine',
  projectId: 'votre_project_id_ici',  // <-- Mettre le bon ID
  dataset: 'production',
  // ...
})
```

Etape 4.7 — Lancer le studio et verifier

```
cd apps/studio
npm run dev
```

Le studio s'ouvre sur `http://localhost:3333`. Verifier que vous voyez les sections : **Product**, **Category**, **Order**.

Etape 4.8 — (Optionnel) Peupler avec des donnees de demo

Si un script de seed existe :

```
cd seed
node seed.js
```

Sinon, passer a la section 8 pour alimenter manuellement le catalogue.

---

5. Configuration Supabase (Auth + BDD)

Supabase gere l'authentification (inscription/connexion), les profils utilisateur, les paniers persistants, les commandes et les points de fidelite.

Sans Supabase, l'app fonctionne en **mode invite** (panier localStorage, pas de comptes).

Etape 5.1 — Creer un compte Supabase

1. Aller sur **https://supabase.com**
2. Cliquer **"Start your project"**
3. Se connecter avec GitHub (recommande) ou email

Etape 5.2 — Creer un nouveau projet

1. Cliquer **"New Project"**
2. Remplir :
   - **Organization** : Selectionner ou creer une organisation
   - **Name** : Nom du client (ex: `epicerie-africaine-clientxyz`)
   - **Database Password** : Generer un mot de passe fort et **LE NOTER QUELQUE PART** (coffre-fort de mots de passe recommande)
   - **Region** : Choisir la plus proche du client
     - Canada : `East US (North Virginia)` ou `Central Canada`
     - France : `West EU (Ireland)` ou `Central EU (Frankfurt)`
3. Cliquer **"Create new project"**
4. **Attendre ~2 minutes** que le projet soit provisionne

Etape 5.3 — Recuperer les cles API

1. Dans le dashboard Supabase, aller dans **Settings** (icone engrenage) > **API**
2. Vous verrez :

```
┌─────────────────────────────────────────────────────┐
│ Project URL                                         │
│ https://abcdefghijk.supabase.co             [Copy]  │
│                                                     │
│ Project API keys                                    │
│ anon public    eyJhbGciOiJIUzI1NiIs...      [Copy]  │
│ service_role   eyJhbGciOiJIUzI1NiIs...      [Copy]  │
└─────────────────────────────────────────────────────┘
```

3. Copier chaque valeur

Etape 5.4 — Remplir les variables d'environnement

Fichier `apps/web/.env.local` :

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://abcdefghijk.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJhbGciOiJIUzI1NiIs...
SUPABASE_SERVICE_ROLE_KEY=eyJhbGciOiJIUzI1NiIs...
```

> ***SECURITE CRITIQUE*** *: La cle* `service_role` *bypass TOUTES les regles de securite (RLS). Elle ne doit JAMAIS etre exposee cote client (jamais dans une variable* `NEXT_PUBLIC_` *). Elle est utilisee uniquement dans les routes API serveur (* `/api/webhook/stripe` *).*

Etape 5.5 — Creer les tables (migrations SQL)

1. Dans le dashboard Supabase, aller dans **SQL Editor** (icone terminal dans la sidebar)
2. Cliquer **"New Query"**
3. Ouvrir le fichier `supabase/migrations/` de votre projet
4. **Copier-coller le contenu de CHAQUE fichier de migration** dans l'ordre chronologique :
   - `001_create_tables.sql` (ou similaire) — Tables principales
   - `002_rls_policies.sql` — Regles de securite Row Level Security
   - `003_indexes.sql` — Index de performance
   - `004_functions.sql` — Fonctions (trigger auto-profil, fidelite)
   - `005_reviews.sql` — Table des avis (si existe)
5. Pour chaque fichier, coller le SQL et cliquer **"Run"** (ou `Ctrl+Enter`)
6. Verifier le message : **"Success. No rows returned."**

Etape 5.6 — Verifier les tables creees

Aller dans **Table Editor** (sidebar) et confirmer la presence de :

Table	Description	Colonnes principales
`profiles`	Profils utilisateur	id, full_name, phone, loyalty_points
`carts`	Paniers persistants	id, user_id, status (active/converted)
`cart_items`	Articles dans les paniers	cart_id, product_slug, name, price_cents, quantity
`orders`	Commandes (creees par webhook)	stripe_session_id, status, customer_email, total_cents
`order_items`	Articles dans les commandes	order_id, product_slug, name, price_cents, quantity
`reviews`	Avis clients	user_id, product_slug, rating, comment

Etape 5.7 — Configurer l'authentification

1. Aller dans **Authentication** (icone cadenas) > **URL Configuration**
2. Configurer :
   - **Site URL** : `http://localhost:3007` (dev) ou `https://votre-domaine.com` (prod)

- **Redirect URLs** : Cliquer "Add URL" et ajouter :
  - `http://localhost:3007/auth/callback`
  - `https://votre-domaine.com/auth/callback` (ajouter pour la prod)

Etape 5.8 — (Dev uniquement) Desactiver la confirmation email

Pour faciliter les tests en developpement :

1. **Authentication > Providers > Email**
2. Decocher **"Confirm email"**
3. Sauvegarder

> ***IMPORTANT** : Reactiver cette option avant la mise en production !*

Etape 5.9 — Verifier les policies RLS

1. Aller dans **Authentication > Policies**
2. Confirmer que chaque table a ses policies de securite :
   - `profiles` : lecture/modification de son propre profil uniquement
   - `carts` / `cart_items` : CRUD sur ses propres paniers
   - `orders` / `order_items` : lecture de ses propres commandes
   - `reviews` : lecture publique, CRUD sur ses propres avis

---

6. Configuration Stripe (Paiements)

Stripe gere le paiement en ligne via Checkout Sessions. Sans Stripe, le bouton "Payer" ne fonctionnera pas.

Etape 6.1 — Creer un compte Stripe

1. Aller sur **https://dashboard.stripe.com/register**
2. Creer un compte (email + mot de passe)
3. Pas besoin de verifier son identite pour le **mode test**

Etape 6.2 — Activer le mode test

1. Dans le dashboard Stripe, verifier en haut a droite
2. Le toggle **"Test mode"** doit etre **active** (couleur orange)
3. Si ce n'est pas le cas, cliquer dessus pour l'activer

> *En mode test, aucun vrai paiement n'est effectue. Les cartes de test simulent les transactions.*

Etape 6.3 — Recuperer les cles API

1. Aller dans **Developers > API keys**
2. Ou directement : https://dashboard.stripe.com/test/apikeys
3. Vous verrez :

```
┌──────────────────────────────────────────────────────┐
│ Publishable key    pk_test_51abc...          [Copy]  │
│ Secret key         sk_test_51abc...    [Reveal] [Copy] │
└──────────────────────────────────────────────────────┘
```

4. Copier les deux cles

Etape 6.4 — Remplir les variables d'environnement

Fichier `apps/web/.env.local` :

```
STRIPE_SECRET_KEY=sk_test_51abc...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_test_51abc...
STRIPE_WEBHOOK_SECRET=whsec_...  # Rempli a l'etape suivante
```

> ***SECURITE** : La cle secrete (`sk_test_...` ou `sk_live_...`) ne doit JAMAIS etre dans une variable `NEXT_PUBLIC_` .*

Etape 6.5 — Configurer le webhook (developpement local)

Le webhook est essentiel : il recoit la confirmation de paiement de Stripe et cree la commande dans la base de donnees.

A. Installer Stripe CLI

```
# Windows (avec Scoop)
scoop install stripe

# Windows (avec Chocolatey)
choco install stripe-cli

# macOS
brew install stripe/stripe-cli/stripe

# Linux (Debian/Ubuntu)
curl -s https://packages.stripe.dev/api/security/keypair/stripe-cli-gpg/public | gpg --dearmor | sudo
tee /usr/share/keyrings/stripe.gpg
echo "deb [signed-by=/usr/share/keyrings/stripe.gpg] https://packages.stripe.dev/stripe-cli-debian-
local stable main" | sudo tee -a /etc/apt/sources.list.d/stripe.list
sudo apt update && sudo apt install stripe
```

B. Se connecter a Stripe

```
stripe login
```

Un navigateur s'ouvre. Cliquer **"Allow access"** pour autoriser la CLI.

C. Ecouter les webhooks en local

Dans un **terminal separe** (laisser tourner pendant le developpement) :

```
stripe listen --forward-to localhost:3007/api/webhook/stripe
```

La commande affiche :

```
Ready! Your webhook signing secret is whsec_abc123def456...
```

D. Copier le secret webhook

Copier le `whsec_...` et le coller dans `apps/web/.env.local` :

```
STRIPE_WEBHOOK_SECRET=whsec_abc123def456...
```

> ***NOTE** : Ce secret change a chaque fois que vous relancez `stripe listen` . En production, il est fixe (voir section 10).*

Etape 6.6 — Cartes de test Stripe

Scenario	Numero de carte	Date	CVC
Paiement reussi	`4242 4242 4242 4242`	`12/34`	`123`
Paiement refuse	`4000 0000 0000 0002`	`12/34`	`123`
Authentification 3D Secure	`4000 0025 0000 3155`	`12/34`	`123`
Fonds insuffisants	`4000 0000 0000 9995`	`12/34`	`123`

Etape 6.7 — Configurer le webhook (production)

Quand le site est deploye en production :

1. Aller dans **Developers > Webhooks** dans le dashboard Stripe
2. Cliquer **"Add endpoint"**
3. Remplir :
   - **Endpoint URL** : `https://votre-domaine.com/api/webhook/stripe`
   - **Events to send** : Selectionner `checkout.session.completed`
4. Cliquer **"Add endpoint"**
5. Copier le **"Signing secret"** affiche
6. Le mettre dans la variable `STRIPE_WEBHOOK_SECRET` de l'hebergeur (Vercel)

---

7. Configuration Cal.com (Rendez-vous)

Cal.com permet aux clients de prendre rendez-vous en ligne. **Ce service est optionnel.** Sans configuration, la page `/appointments` affiche un fallback avec un lien de contact.

Etape 7.1 — Creer un compte Cal.com

1. Aller sur **https://cal.com**
2. Cliquer **"Get Started"**
3. Creer un compte (Google, email, etc.)
4. Choisir un username (ex: `epicerie-africaine`)

Etape 7.2 — Creer un Event Type

1. Dans le dashboard Cal.com, aller dans **Event Types**
2. Cliquer **"New Event Type"**
3. Configurer :
   - **Title** : ex. "Consultation en boutique"
   - **Duration** : 30 minutes (ou selon le besoin)
   - **Location** : En personne / Telephone / Video
4. Configurer les **disponibilites** (jours, heures)
5. Sauvegarder

Etape 7.3 — Recuperer l'URL de l'event

1. Dans **Event Types**, cliquer sur l'event cree
2. L'URL est en haut, au format : `votre-username/nom-event`
3. Exemple : `epicerie-africaine/consultation`

Etape 7.4 — Remplir la variable d'environnement

Fichier `apps/web/.env.local` :

```
NEXT_PUBLIC_CALCOM_EMBED_URL=epicerie-africaine/consultation
```

> ***NOTE** : Mettre uniquement `username/event-name`, pas l'URL complete.*

---

8. Alimenter le catalogue (Sanity Studio)

Etape 8.1 — Lancer le studio

```
cd apps/studio
```

```
npm run dev
```

Ouvrir **http://localhost:3333** dans le navigateur.

Etape 8.2 — Creer les categories

Aller dans **Category** > cliquer le bouton **"+"** pour creer.

Categories recommandees pour une epicerie africaine :

#	Titre	Slug (auto)	Ordre	Description
1	Epices	epices	1	Epices et condiments d'Afrique
2	Cereales et Feculents	cereales-et-feculents	2	Riz, semoule, farine, igname...
3	Sauces et Pates	sauces-et-pates	3	Sauces tomate, pate d'arachide...
4	Boissons	boissons	4	Jus, bissap, gingembre...
5	Snacks	snacks	5	Chips de plantain, chin-chin...
6	Huiles	huiles	6	Huile de palme, huile de coco...
7	Produits frais	produits-frais	7	Plantain, manioc, gombo...
8	Soins et Beaute	soins-et-beaute	8	Beurre de karite, savon noir...

Pour chaque categorie :

1. Remplir le **titre**
2. Cliquer **"Generate"** a cote du slug pour le generer automatiquement
3. Mettre le numero d'**ordre** (pour le tri)
4. Ajouter une **description** courte
5. Uploader une **image** representative (format carre, min 600x600px)
6. Cliquer **"Publish"**

Etape 8.3 — Creer les produits

Aller dans **Product** > cliquer **"+"** pour creer un nouveau produit.

Pour chaque produit, remplir **TOUS** les champs suivants :

Champs obligatoires :

Champ	Description	Exemple
Title	Nom du produit	"Piment Camerounais Fume"
Slug	Cliquer "Generate"	`piment-camerounais-fume`
Price	Prix en dollars (nombre)	`8.99`
Currency	Devise	`CAD` (par defaut)
Images	Min 1 image, idealement 3-4	Photos HD du produit

Category	Reference a une categorie	Selectionner "Epices"
Stock	Nombre d'unites disponibles	`50`

> ***IMPORTANT** : Si le stock est a **0**, le bouton "Ajouter au panier" sera remplace par "Rupture de stock" et le client ne pourra pas acheter.*

Champs recommandes :

Champ	Description	Exemple
Description	Texte riche (gras, italique, listes)	Description detaillee du produit
Tags	Etiquettes filtrable	Bio, Pimente, Surgele
Origin Country	Pays d'origine	Cameroun
Spicy Level	0-3	2 (Moyen)
Is Frozen	Produit surgele ?	Non
Is Organic	Produit bio ?	Oui
Is Featured	Afficher en best-seller sur l'accueil	Oui (8 max recommandes)
Related Products	Produits similaires/complementaires	Selectionner 2-4 produits

Conseils pour les images :

- **Format** : JPEG ou WebP (plus leger)
- **Taille** : Minimum 800x800px, idealement 1200x1200px
- **Ratio** : Carre (1:1) pour uniformite dans la grille
- **Fond** : Fond blanc ou neutre pour un rendu professionnel
- **Alt text** : Toujours remplir le texte alternatif (SEO + accessibilite)
  - Exemple : "Piment camerounais fume dans un bol en bois"

Etape 8.4 — Nombre minimum de produits recommande

Type de boutique	Nombre minimum	Ideal
Demo / MVP	10 produits	15
Boutique standard	20 produits	30-50
Grande boutique	50+ produits	100+

Etape 8.5 — Verifier la publication

Apres avoir cree les produits :

1. S'assurer que **CHAQUE produit et categorie est publie** (bouton "Publish")
2. Aller sur `http://localhost:3007/shop` et rafraichir la page
3. Les produits doivent apparaitre dans la grille

4. Tester les filtres (categorie, tags, prix, recherche)

---

9. Tests fonctionnels

Avant de livrer le site, effectuer TOUS les tests suivants.

Test 1 : Navigation generale

- [ ] Page d'accueil (`/`) : hero, categories, best-sellers s'affichent
- [ ] Page boutique (`/shop`) : tous les produits avec filtres
- [ ] Page produit (`/product/[slug]`) : image, prix, description, produits similaires
- [ ] Pages legales (`/legal/terms` , `/legal/privacy` , `/legal/refunds` , `/legal/imprint`)
- [ ] Page rendez-vous (`/appointments`) : embed Cal.com ou fallback
- [ ] Header : logo, navigation, icone panier avec badge
- [ ] Footer : liens fonctionnels vers toutes les sections
- [ ] Responsive : tester sur mobile (375px), tablette (768px), desktop (1440px)

Test 2 : Panier

- [ ] Ajouter un produit depuis la page boutique (bouton "+")
- [ ] Ajouter un produit depuis la page produit detail
- [ ] Le badge du panier se met a jour en temps reel
- [ ] Page panier (`/cart`) : produits affiches avec images, prix, quantites
- [ ] Modifier la quantite (+ / -)
- [ ] Supprimer un article
- [ ] Le total se recalcule correctement
- [ ] Livraison gratuite affichee si total >= 75$
- [ ] Panier persiste apres rafraichissement de la page (localStorage)

Test 3 : Checkout et paiement

Pre-requis : Stripe et Supabase configures, `stripe listen` en cours.

- [ ] Ajouter des produits au panier puis aller sur `/checkout`
- [ ] **Mode livraison** :
  - [ ] Formulaire : nom, email, telephone, adresse complete
  - [ ] Frais de livraison affiches (5.99$ ou gratuit si >= 75$)
  - [ ] Cliquer "Payer" → redirection vers Stripe Checkout
  - [ ] Payer avec `4242 4242 4242 4242` / `12/34` / `123`
  - [ ] Redirection vers `/success` avec details de la commande
  - [ ] Panier vide apres le paiement
- [ ] **Mode retrait (pickup)** :
  - [ ] Selectionner un creneau de retrait
  - [ ] Pas de frais de livraison
  - [ ] Meme flux de paiement

- [ ] Verifier dans le **terminal Stripe CLI** : evenement `checkout.session.completed` recu
- [ ] Verifier dans **Supabase > Table Editor > orders** : commande creee
- [ ] Verifier dans **Stripe Dashboard > Payments** : paiement visible

Test 4 : Authentification

Pre-requis : Supabase configure.

- [ ] **Inscription** (`/auth/register`) : creer un compte avec email/mot de passe
- [ ] Si confirmation email desactivee : connexion immediate
- [ ] Si confirmation email activee : verifier l'email puis se connecter
- [ ] **Connexion** (`/auth/login`) : se connecter avec le compte cree
- [ ] Le header affiche l'icone utilisateur au lieu de "Se connecter"
- [ ] **Page compte** (`/account`) : profil, historique de commandes, points de fidelite
- [ ] **Deconnexion** : fonctionne correctement, retour en mode invite
- [ ] **Fusion panier** : ajouter des produits en invite → se connecter → les produits sont conserves

Test 5 : Cas limites

- [ ] Produit en rupture de stock (stock = 0) : bouton "Rupture de stock" desactive
- [ ] Checkout avec panier vide : redirection vers la boutique
- [ ] Paiement refuse (carte `4000 0000 0000 0002`) : message d'erreur correct
- [ ] Page 404 : tester une URL inexistante (ex: `/page-qui-nexiste-pas`)

---

10. Deploiement en production

Option recommandee : Vercel

Etape 10.1 — Creer un compte Vercel

1. Aller sur **https://vercel.com**
2. Se connecter avec **GitHub** (recommande)
3. Autoriser l'acces au repository du projet

Etape 10.2 — Importer le projet

1. Cliquer **"Add New" > "Project"**
2. Selectionner le repository Git
3. Configurer :
   - **Framework Preset** : Next.js (auto-detecte)
   - **Root Directory** : `apps/web`
   - **Build Command** : `npm run build` (par defaut)
   - **Output Directory** : `.next` (par defaut)

Etape 10.3 — Configurer les variables d'environnement

Dans **Settings > Environment Variables**, ajouter TOUTES les variables :

```
# Sanity
NEXT_PUBLIC_SANITY_PROJECT_ID=votre_project_id
NEXT_PUBLIC_SANITY_DATASET=production
NEXT_PUBLIC_SANITY_API_VERSION=2024-01-01
SANITY_API_TOKEN=votre_token

# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
SUPABASE_SERVICE_ROLE_KEY=eyJ...

# Stripe (MODE LIVE !)
STRIPE_SECRET_KEY=sk_live_...
NEXT_PUBLIC_STRIPE_PUBLISHABLE_KEY=pk_live_...
STRIPE_WEBHOOK_SECRET=whsec_...

# Cal.com
NEXT_PUBLIC_CALCOM_EMBED_URL=username/event

# App
NEXT_PUBLIC_BASE_URL=https://votre-domaine.com
NEXT_PUBLIC_SITE_NAME=Epicerie Africaine

# Shipping
NEXT_PUBLIC_SHIPPING_COST=5.99
NEXT_PUBLIC_FREE_SHIPPING_THRESHOLD=75
```

> ***SECURITE** : En production, utiliser les cles **LIVE** de Stripe (`sk_live_` , `pk_live_`), pas les cles de test. Verifier que l'identite est validee dans le dashboard Stripe avant de passer en live.*

Etape 10.4 — Deployer

1. Cliquer **"Deploy"**
2. Attendre la fin du build (~2-3 minutes)
3. Vercel fournit une URL (ex: `epicerie-africaine.vercel.app`)

Etape 10.5 — Configurer le domaine personnalise

1. Dans Vercel, aller dans **Settings > Domains**
2. Ajouter le domaine du client (ex: `epicerie-africaine.ca`)
3. Suivre les instructions pour configurer les DNS :
   - **Type A** : `76.76.21.21`
   - **Type CNAME** (www) : `cname.vercel-dns.com`
4. Attendre la propagation DNS (~5-30 minutes)
5. Vercel configure automatiquement le certificat SSL (HTTPS)

Etape 10.6 — Mises a jour post-deploiement

Apres le deploiement, mettre a jour :

1. `NEXT_PUBLIC_BASE_URL` dans Vercel : `https://votre-domaine.com`
2. **Supabase** > Authentication > URL Configuration :

- Site URL : `https://votre-domaine.com`
- Redirect URLs : ajouter `https://votre-domaine.com/auth/callback`

3. **Sanity** > API > CORS Origins : ajouter `https://votre-domaine.com`
4. **Stripe** > Webhooks : creer l'endpoint de production (voir section 6.7)
5. **Supabase** > Reactiver la confirmation email si desactivee

---

11. Personnalisation pour un nouveau client

Quand tu configures le site pour un nouveau client, voici les elements a adapter :

11.1 — Branding

Element	Fichier(s) a modifier	Quoi changer
Nom du site	`.env.local` (`NEXT_PUBLIC_SITE_NAME`)	Nom du client
Logo	`apps/web/components/Header.tsx` et `Footer.tsx`	Remplacer le logo/texte
Couleurs	`apps/web/tailwind.config.ts`	Modifier les couleurs accent
Favicon	`apps/web/public/favicon.ico`	Remplacer
Meta SEO	`apps/web/app/layout.tsx`	Title, description, OG images

11.2 — Couleurs (Design Tokens)

Les couleurs sont centralisees dans `tailwind.config.ts` :

```
colors: {
  background: '#F9F9F7',  // Fond principal
  foreground: '#1A1A1A',  // Texte principal
  accent: {
    DEFAULT: '#CCA43B',   // Or / doré (CTAs, liens)
    light: '#E0C36A',     // Version claire
    dark: '#B08A2A',      // Version foncée
  }
}
```

11.3 — Contenu legal

Page	Fichier	A adapter
CGV	`apps/web/app/legal/terms/page.tsx`	Nom societe, domaine, juridiction
Confidentialite	`apps/web/app/legal/privacy/page.tsx`	Responsable, DPO, cookies
Retours	`apps/web/app/legal/refunds/page.tsx`	Delais, conditions
Mentions legales	`apps/web/app/legal/imprint/page.tsx`	Raison sociale, SIRET/NEQ, adresse

11.4 — Livraison

Dans `.env.local` :

```
NEXT_PUBLIC_SHIPPING_COST=5.99          # Frais de livraison
NEXT_PUBLIC_FREE_SHIPPING_THRESHOLD=75  # Seuil livraison gratuite
```

11.5 — Devise

La devise par defaut est **CAD** (Dollar canadien). Pour changer en EUR, modifier dans Sanity les produits (champ `currency`). Le format d'affichage est gere dans le composant `ProductCard`.

---

12. Maintenance et operations courantes

Ajouter/modifier des produits

1. Ouvrir Sanity Studio (`npm run dev` dans `apps/studio/`)
2. Creer ou modifier les produits
3. Cliquer **"Publish"**
4. Les changements apparaissent sur le site en **~60 secondes** (revalidation ISR)

Gerer les commandes

1. Ouvrir le **dashboard Supabase** > Table Editor > `orders`
2. Les commandes sont creees automatiquement par le webhook Stripe
3. Statuts possibles : `pending` → `paid` → `shipped` → `delivered`
4. Pour mettre a jour un statut, modifier directement dans Supabase

Gerer les paiements

1. Ouvrir le **Stripe Dashboard**
2. Voir les paiements dans **Payments**
3. Effectuer des remboursements si necessaire
4. Gerer les litiges dans **Disputes**

Voir les clients

1. **Supabase** > Authentication > Users : liste des comptes
2. **Supabase** > Table Editor > `profiles` : details des profils

---

13. Checklist de livraison

Avant de livrer le site au client, verifier chaque point :

Configuration

- [ ] Sanity : projet cree, project ID configure, token API genere
- [ ] Supabase : projet cree, tables creees, RLS actif, auth configuree
- [ ] Stripe : compte cree, cles API configurees, webhook configure
- [ ] Cal.com : (si applicable) event type cree, URL configuree
- [ ] Variables d'environnement : TOUTES remplies dans `.env.local` ET sur l'hebergeur
- [ ] `.env.local` est dans `.gitignore`

Contenu

- [ ] Categories creees dans Sanity (min 5)
- [ ] Produits crees dans Sanity (min 15-20) avec images HD
- [ ] Best-sellers marques (`isFeatured = true` , 8 max)
- [ ] Stock > 0 pour les produits disponibles
- [ ] Pages legales adaptees au client (nom, adresse, juridiction)

Tests

- [ ] Navigation : toutes les pages accessibles sans erreur
- [ ] Panier : ajout, modification, suppression, persistance
- [ ] Paiement : flux complet teste avec carte test
- [ ] Auth : inscription, connexion, deconnexion, page compte
- [ ] Responsive : teste sur mobile, tablette, desktop
- [ ] SEO : meta tags, robots.txt, sitemap presents

Production

- [ ] Deploye sur Vercel (ou equivalent)
- [ ] Domaine personnalise configure avec HTTPS
- [ ] Stripe en **mode Live** avec cles de production
- [ ] Webhook Stripe de production configure
- [ ] Confirmation email activee dans Supabase
- [ ] CORS Sanity mis a jour avec le domaine de prod
- [ ] URLs Supabase mises a jour avec le domaine de prod
- [ ] `NEXT_PUBLIC_BASE_URL` mis a jour avec le domaine de prod

Documentation client

- [ ] Acces Sanity Studio donne au client (pour gerer ses produits)
- [ ] Formation basique sur l'ajout/modification de produits
- [ ] Acces Stripe Dashboard (pour voir les paiements)
- [ ] Procedure de remboursement expliquee

---

14. Depannage (FAQ)

"Les produits ne s'affichent pas"

1. Verifier que les produits sont **publies** dans Sanity Studio
2. Verifier que `NEXT_PUBLIC_SANITY_PROJECT_ID` est correct dans `.env.local`
3. Verifier les CORS dans Sanity (l'origine du site doit etre autorisee)
4. Attendre 60 secondes (revalidation ISR) et rafraichir

"Le paiement echoue"

1. Verifier que `STRIPE_SECRET_KEY` est correct
2. Verifier que `stripe listen` tourne (dev) ou que le webhook est configure (prod)
3. Verifier la console du navigateur et les logs serveur pour les erreurs
4. Tester avec la carte `4242 4242 4242 4242`

"L'inscription/connexion ne fonctionne pas"

1. Verifier que `NEXT_PUBLIC_SUPABASE_URL` et `NEXT_PUBLIC_SUPABASE_ANON_KEY` sont corrects
2. Verifier les Redirect URLs dans Supabase Authentication
3. Si "confirmation email" est activee : verifier la boite mail (et les spams)

"Le webhook ne cree pas de commande"

1. Verifier que `STRIPE_WEBHOOK_SECRET` correspond au secret affiche par `stripe listen`
2. Verifier que les tables `orders` et `order_items` existent dans Supabase
3. Verifier que `SUPABASE_SERVICE_ROLE_KEY` est correct
4. Regarder les logs du terminal ou le webhook echoue

"Erreur CORS"

1. Aller dans Sanity > API > CORS Origins
2. Ajouter l'URL exacte du site (avec `https://`)
3. Cocher "Allow credentials"
4. Sauvegarder et recharger la page

"Le site est lent"

1. Verifier la taille des images dans Sanity (optimiser si > 2MB)
2. Verifier la region Supabase (choisir la plus proche des utilisateurs)
3. Activer le cache CDN sur Vercel (actif par defaut)
4. Verifier le nombre de produits charges par page

Fin du guide

Document cree en fevrier 2026. Mettre a jour en cas de changement majeur dans l'architecture ou les services utilises.