🌍 J A D A R I S E L A B S

Laboratoire IA Tout-en-1

Conçu pour l'Afrique de l'Ouest et le Grand Public

CAHIER DES CHARGES | **MVP — 2 Full-Stack Devs**

Équipe	2 Développeurs Full-Stack (Front + Back)
Délai	2 à 3 semaines (14-21 jours)
Budget	0 EUR (APIs gratuites + free tiers)
Langues	Français / English (Bilingue)
Modèle éco.	Freemium (Gratuit + Abonnements)

1. Vision & Modules IA du MVP

1.1 Vision JadaRiseLabs

Plateforme web tout-en-un qui démocratise l'accès aux IA génératives pour le grand public africain. Priorité : simplicité, rapidité, accessibilité mobile, paiements Mobile Money.

Nom : Jada (sagesse en Haoussa) + Rise (élévation) + Labs (innovation) — ancrage culturel fort pour l'Afrique de l'Ouest.

1.2 Modules IA — Mix recommandé (ordre de priorité)

Priorité	Module	Cas d'usage	API gratuite
🥇 P1	Génération d'images	Texte → Image (portraits, logos, art)	Hugging Face (FLUX / SDXL)
🥇 P1	Chat IA / Texte	Assistant, copywriting, traduction	Groq API (LLaMA 3.3 70B)
🥈 P2	Génération vidéo	Texte → courte vidéo (5-15s)	Replicate (Wan 2.1 ou Luma Dream)
🥈 P2	Amélioration image	Upscale, background removal	Hugging Face / Remove.bg
🥉 P3	Génération audio	Musique, voix off, jingles	Suno AI / Bark (HF)
🥉 P3	Code assistant	Aide au dev, debug, refacto	Groq (CodeLlama)

✅ **MVP SCOPE : Semaine 1-2 → Images + Chat | Semaine 3 → Vidéo + Audio (si temps)**

⚠️ **Commencer avec P1 uniquement garantit un MVP déployable en 14 jours.**

2. Fonctionnalités Utilisateur Finales

2.1 Authentification & Profil

- Inscription / Connexion via email + mot de passe
- OAuth Google (optionnel, simplifie l'onboarding)
- Profil utilisateur : avatar, pseudo, langue préférée (FR/EN)
- Historique complet des générations (accessible via dashboard)

2.2 Dashboard Utilisateur

- Vue d'ensemble : crédits restants, générations ce mois-ci, dernières créations
- Statistiques personnelles : outils les plus utilisés, évolution mensuelle
- Accès direct aux modules IA (cartes cliquables : Image, Chat, Vidéo...)

2.3 Galerie Personnelle (SANS communauté publique)

- Stockage de toutes les créations (images, vidéos, textes générés)
- Organisation par type (filtres : images / vidéos / texte / audio)
- Téléchargement instantané (HD pour Premium, SD pour Gratuit)
- SUPPRESSION de la galerie publique / communauté → focus sur l'utilisateur

2.4 Partage Social Direct (Remplacement de la communauté)

Au lieu d'une galerie communautaire interne, chaque génération inclut un bouton de partage social optimisé pour l'Afrique de l'Ouest :

- Partage WhatsApp (priorité n°1 en Afrique de l'Ouest)
- Partage Facebook
- Partage Twitter/X
- Téléchargement direct (avec watermark discret pour plan gratuit)
- Copie du lien (si hébergement public activé)

> ⚠️ **Watermark automatique sur images/vidéos gratuites : 'Created with JadaRiseLabs' en bas à droite (transparent, petit).**

2.5 Système de Crédits & Plans Freemium

Plan	Crédits/mois	Image HD	Vidéo	Audio	Watermark	Prix
Gratuit	50	❌	❌	❌	✅ Oui	0 F CFA
Starter	200	✅	✅ (5s)	✅	❌ Non	500 F CFA/m
Pro	Illimité	✅	✅ (15s)	✅	❌ Non	1500 F CFA/

2.6 Paiement Mobile Money (Intégration CinetPay)

- Orange Money (Côte d'Ivoire, Sénégal, Mali, Burkina Faso, Guinée)
- Wave (Sénégal, Côte d'Ivoire, Mali, Burkina)
- MTN Mobile Money (Côte d'Ivoire, Ghana, Bénin)
- Moov Money (Côte d'Ivoire, Togo, Bénin)
- Carte bancaire Visa/Mastercard (diaspora)

```
API recommandée : CinetPay (https://cinetpay.com) — SDK Node.js + webhook
```

3. Architecture Technique Complète

3.1 Stack Technologique (100% gratuit pour MVP)

Couche	Technologie	Pourquoi ?
Frontend	Next.js 14 (App Router)	SSR, SEO, perf 3G, routing moderne
UI Library	Tailwind CSS + shadcn/ui	Composants beaux, customisables, légers
Backend API	Next.js API Routes (serverless)	Pas de serveur séparé, déploiement simple
Base de données	Supabase (PostgreSQL)	Auth intégrée + Storage + Row Level Security
Auth	Supabase Auth	Email/password + Google OAuth gratuit
Stockage fichiers	Supabase Storage	10 GB gratuit, CDN intégré
Hébergement	Vercel	Free tier Next.js, déploiement Git automatique
IA Images	Hugging Face Inference API	SDXL, FLUX gratuits (rate limited)
IA Texte	Groq API	LLaMA 3.3 70B ultra rapide, free tier généreux
IA Vidéo	Replicate API	Wan 2.1, Luma Dream (free credits au départ)
Paiement	CinetPay	Mobile Money panafricain, frais raisonnables

3.2 Schéma d'architecture (simplifié)

```
User (Browser) → Next.js (Frontend + API Routes)
  ↓                          ↓
Supabase Auth           AI APIs (HF, Groq, Replicate)
Supabase DB             CinetPay (paiement)
Supabase Storage (images/vidéos générées)
```

3.3 Structure du projet (Next.js App Router)

```
/app
  /page.tsx                    # Landing page
  /dashboard/page.tsx          # Dashboard utilisateur
  /studio/[module]/page.tsx    # Studio IA (image, chat, video...)
  /gallery/page.tsx            # Galerie personnelle
  /api/                        # API Routes (backend)
    /auth/                     # Endpoints auth (Supabase wrapper)
    /generate/                 # Endpoints génération IA
      /image/route.ts
      /chat/route.ts
      /video/route.ts
    /payment/                  # Webhooks CinetPay
```

```
/components                    # Composants React réutilisables
/lib                           # Utils, configs, types
/public                        # Assets statiques
/supabase                      # Migrations SQL, seed data
```

4. Base de Données Supabase (PostgreSQL)

4.1 Schéma des tables principales

Voici les tables essentielles pour le MVP. Supabase Auth gère déjà la table 'auth.users', donc on crée uniquement nos tables métier.

TABLE : profiles

```
id              uuid PRIMARY KEY (foreign key → auth.users.id)
username        text UNIQUE
avatar_url      text
preferred_lang  text DEFAULT 'fr' (fr | en)
plan            text DEFAULT 'free' (free | starter | pro)
credits         integer DEFAULT 50
created_at      timestamp
updated_at      timestamp
```

TABLE : generations

```
id              uuid PRIMARY KEY
user_id         uuid REFERENCES profiles(id) ON DELETE CASCADE
type            text (image | chat | video | audio | code)
prompt          text (le prompt utilisateur)
result_url      text (URL fichier généré dans Supabase Storage)
metadata        jsonb (params: model, size, duration, etc.)
credits_used    integer
created_at      timestamp
```

TABLE : subscriptions

```
id              uuid PRIMARY KEY
user_id         uuid REFERENCES profiles(id) ON DELETE CASCADE
plan            text (starter | pro)
status          text (active | canceled | expired)
start_date      timestamp
end_date        timestamp
payment_id      text (CinetPay transaction ID)
created_at      timestamp
```

TABLE : payments

```
id              uuid PRIMARY KEY
user_id         uuid REFERENCES profiles(id)
subscription_id uuid REFERENCES subscriptions(id)
amount          integer (en centimes de F CFA)
provider        text (orange_money | wave | mtn | moov | card)
```

```
status        text (pending | success | failed)
transaction_id text (CinetPay transaction ID)
created_at    timestamp
```

4.2 Row Level Security (RLS) — Sécurité critique

Supabase PostgreSQL permet de gérer les permissions au niveau des lignes. Cela empêche les utilisateurs de voir les données des autres.

```
-- Activer RLS sur toutes les tables
ALTER TABLE profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE generations ENABLE ROW LEVEL SECURITY;
ALTER TABLE subscriptions ENABLE ROW LEVEL SECURITY;
ALTER TABLE payments ENABLE ROW LEVEL SECURITY;
```

```
-- Policy : un user ne voit que ses propres données
CREATE POLICY 'Users can view own profile'
  ON profiles FOR SELECT USING (auth.uid() = id);
```

```
CREATE POLICY 'Users can view own generations'
  ON generations FOR SELECT USING (auth.uid() = user_id);
```

⚠️ **RLS est OBLIGATOIRE. Sans lui, tous les users voient toutes les données. À configurer dès le début !**

5. Workflow de Développement — 2 Full-Stack Devs

> ✅ **Principe clé : Vous faites TOUS LES DEUX du front ET du back en parallèle. Éviter les conflits = découpage par MODULES, pas par couches.**

5.1 Stratégie Git — Branching Model

Utiliser un modèle de branches simple mais efficace pour travailler en parallèle sans se bloquer :

- main — branche de production (déployée sur Vercel)
- develop — branche de développement (fusion de toutes les features)
- feature/nom-module — une branche par module IA ou fonctionnalité

Exemple de branches actives simultanément :

```
feature/auth-setup          (Dev 1 : authentification)
feature/image-generation    (Dev 2 : module image)
feature/dashboard-ui        (Dev 1 : dashboard après auth)
feature/chat-generation     (Dev 2 : module chat après image)
```

5.2 Règles Git pour éviter les conflits

- Toujours créer une branche depuis develop, jamais depuis main
- Pull develop avant de créer une nouvelle branche (git pull origin develop)
- Commits fréquents avec messages clairs : 'feat: add image generation API route'
- Pull Request (PR) obligatoire pour merger dans develop → l'autre dev review
- Merge dans main uniquement après tests complets sur develop
- Éviter de toucher les mêmes fichiers en parallèle (découpage par modules)

5.3 Conventions de Code (à respecter strictement)

Pour que le code soit homogène même si vous codez en parallèle :

- Nommage fichiers : kebab-case pour les fichiers (image-generator.tsx)
- Nommage composants React : PascalCase (ImageGenerator)
- Nommage variables : camelCase (userCredits, generationResult)
- Nommage API routes : /api/generate/image (pas /api/generateImage)
- Formatage : Prettier + ESLint (config partagée dans le repo)
- TypeScript strict : pas de 'any', toujours typer les props et retours

```
// .prettierrc (à ajouter dans le repo)
{
  "semi": true,
```

```
  "singleQuote": true,
  "trailingComma": "es5",
  "printWidth": 100
}
```

5.4 Communication & Synchronisation Quotidienne

- Daily standup rapide (5-10 min) : Qu'est-ce que j'ai fait ? Quoi aujourd'hui ? Blocages ?
- Utiliser un tableau Kanban (Trello / Notion / GitHub Projects)
- Colonnes : TODO | IN PROGRESS | REVIEW | DONE
- Chaque tâche assignée à Dev 1 ou Dev 2 (pas de zone grise)
- Slack / WhatsApp pour communication rapide en cas de blocage

6. Découpage des Tâches — Dev 1 vs Dev 2

Principe : découpage par MODULES COMPLETS (front + back + DB), pas par couches. Chaque dev prend un module de A à Z.

Module / Tâche	Dev 1	Dev 2	Dépendances
Setup projet + Git + Supabase	✅ Lead	✅ Participe	—
Auth (login, signup, profil)	✅ Complet	—	Aucune
Dashboard utilisateur (UI)	✅ Complet	—	Auth terminé
Module Image (front + back)	—	✅ Complet	Auth terminé
Module Chat IA (front + back)	✅ Complet	—	Auth terminé
Galerie personnelle (UI + fetch)	—	✅ Complet	Auth + Image
Module Vidéo (front + back)	✅ Complet	—	Image terminé
Système de crédits (DB + logic)	✅ Complet	—	Auth terminé
Paiement CinetPay (webhook + UI)	—	✅ Complet	Crédits terminé
Partage social (boutons + métadonnées)	✅ Complet	—	Galerie terminée
Optimisation mobile (responsive)	✅ 50%	✅ 50%	Toutes features OK
Tests + Debug + Déploiement	✅ 50%	✅ 50%	MVP complet

⚠️ **Si un dev finit en avance, il prend des tâches de polish (animations, loading states, erreurs UX) ou aide l'autre sur sa feature.**

7. Planning Détaillé — 3 Semaines (21 Jours)

SEMAINE 1 — Fondations + Modules Core (Jours 1-7)

Jour	Dev 1	Dev 2
J1	Setup Next.js + Supabase + Git + Prettier	Setup Next.js + Supabase + Git + Prettier
J2	Auth : signup/login UI + Supabase Auth	DB : créer tables (profiles, generations)
J3	Auth : profil utilisateur + session gestion	Module Image : UI studio (prompt input, params)
J4	Dashboard : UI overview + stats vides	Module Image : API route + HuggingFace SDXL
J5	Dashboard : fetch générations + affichage	Module Image : intégration front ↔ back
J6	Module Chat : UI chat interface	Module Image : téléchargement + storage Supabase
J7	Module Chat : API route + Groq LLaMA	Debug + polish module Image

✅ ✅ **Fin Semaine 1 : Auth OK, Dashboard OK, Module Image OK, Module Chat en cours**

SEMAINE 2 — Complétion Modules + Galerie (Jours 8-14)

Jour	Dev 1	Dev 2
J8	Module Chat : intégration front ↔ back	Galerie : UI liste générations (filtres, search)
J9	Module Chat : historique conversations	Galerie : téléchargement HD/SD selon plan
J10	Système crédits : logique déduction par génération	Galerie : watermark automatique (gratuit)
J11	Système crédits : affichage dashboard + warnings	Module Vidéo : UI studio (prompt, duration)
J12	Module Vidéo : API route + Replicate Wan	Module Vidéo : intégration front ↔ back
J13	Partage social : boutons WhatsApp/FB/Twitter	Module Vidéo : téléchargement + storage
J14	Partage social : métadonnées Open Graph	Debug + polish module Vidéo

✅ ✅ **Fin Semaine 2 : Tous modules IA OK, Galerie OK, Crédits OK, Partage social OK**

SEMAINE 3 — Paiement + Polish + Déploiement (Jours 15-21)

Jour	Dev 1	Dev 2
J15	Tests end-to-end : auth + génération + crédits	Paiement : intégration CinetPay SDK
J16	Optimisation mobile : responsive toutes pages	Paiement : webhook handler + DB update
J17	Animations & loading states (skeletons)	Paiement : UI checkout + confirmation
J18	Gestion erreurs : messages clairs (API fails, crédits épuisés)	Paiement : tests sandbox CinetPay
J19	Textes bilingues FR/EN (i18n ou hardcodé)	Polish UI : couleurs JadaRiseLabs, typo
J20	Tests de charge API + DB (rate limiting)	Déploiement Vercel + env vars production
J21	Tests finaux + corrections bugs critiques	Tests finaux + corrections bugs critiques

✅ 🚀 **Fin Semaine 3 (J21) : MVP COMPLET déployé en production sur Vercel**

8. Gestion des Conflits de Code (Stratégies Critiques)

8.1 Principe de base : Découpage modulaire

Le plus gros risque de conflit Git vient de travailler sur les MÊMES fichiers en même temps. Avec le découpage par modules, chaque dev travaille sur des fichiers différents 90% du temps.

- Dev 1 travaille dans /app/auth/, /app/dashboard/, /app/chat/
- Dev 2 travaille dans /app/studio/image/, /app/gallery/, /app/studio/video/
- Les deux ne touchent presque jamais les mêmes fichiers simultanément

> ⚠️ **Cas à risque : fichiers partagés (/lib/types.ts, /lib/supabase.ts, /components/ui/). Solution : coordonner avant modification.**

8.2 Fichiers partagés — Comment les gérer ?

Certains fichiers sont partagés entre tous les modules. Voici comment éviter les conflits :

- /lib/types.ts — types TypeScript partagés
- → Chaque dev ajoute ses types dans une SECTION dédiée (commentaires clairs)
- → Pull avant d'ajouter un nouveau type, push immédiatement après

- /lib/supabase.ts — client Supabase
- → Modifier uniquement si absolument nécessaire
- → Prévenir l'autre dev avant de modifier

- /components/ui/ — composants shadcn/ui
- → Dev 1 installe les composants nécessaires (Button, Input, Card, Dialog)
- → Dev 2 réutilise sans modifier, ou ajoute de nouveaux (Select, Checkbox)

8.3 Résolution d'un conflit Git (procédure)

Si malgré tout un conflit apparaît lors d'un merge, voici comment le résoudre :

```
1. git pull origin develop (déclenche le conflit)
2. Git marque les fichiers en conflit dans VS Code
3. Ouvrir le fichier → sections marquées :
   <<<<<<< HEAD (votre code)
   ======= (code de l'autre dev)
   >>>>>>> develop
4. Choisir la bonne version ou fusionner manuellement
5. Supprimer les marqueurs <<<, ===, >>>
6. git add <fichier-résolu>
```

```
7. git commit -m 'resolve: merge conflict in types.ts'
8. git push origin feature/votre-branche
```

⚠️ **En cas de doute, demander à l'autre dev avant de résoudre un conflit sur son code.**

8.4 Base de données — Éviter les conflits de migrations

Supabase stocke le schéma DB dans des fichiers de migration SQL. Si les deux devs créent des migrations en parallèle, risque de conflit.

✅ **Solution : Un seul dev (Dev 1) gère les migrations SQL pendant la semaine 1. Dev 2 pull régulièrement.**

- Dev 1 crée TOUTES les tables initiales (profiles, generations, subscriptions, payments)
- Dev 2 ne touche pas aux migrations avant la semaine 2
- Semaine 2+ : coordonner avant toute nouvelle migration (Slack/WhatsApp)

```
supabase migration new add_metadata_to_generations
# ⚠️ Prévenir l'autre dev avant de run cette commande
```

9. Optimisations Critiques — Connexions Lentes (3G/4G)

L'Afrique de l'Ouest a des connexions souvent lentes et instables. Ces optimisations sont NON-NÉGOCIABLES pour l'adoption.

9.1 Frontend — Optimisations de performance

- Lazy loading : charger les images uniquement quand elles entrent dans le viewport
- Compression WebP : toutes les images en WebP (30-50% plus léger que PNG/JPG)
- Skeleton loaders : afficher des placeholders pendant le chargement (meilleure UX que spinners)
- Service Worker (PWA) : mise en cache des assets pour navigation partielle hors-ligne
- Code splitting : charger uniquement le JS nécessaire à chaque page (Next.js le fait par défaut)
- Minimize bundle size : analyser avec next build && next analyze

```
// next.config.js - Optimisation images
module.exports = {
  images: {
    formats: ['image/webp'], // Force WebP
    deviceSizes: [640, 750, 828, 1080, 1200], // Breakpoints mobiles
  },
};
```

9.2 Backend — Optimisations API

- Rate limiting : max 10 requêtes/minute par IP pour éviter abus
- Compression Gzip : activer sur Vercel (automatique, mais vérifier)
- Réponses JSON minimales : ne renvoyer que les champs nécessaires
- Timeouts courts : max 30s pour génération IA (sinon erreur claire)
- Mise en cache : stocker résultats identiques (même prompt = même image)

```
// Exemple cache simple (Redis gratuit via Upstash)
const cacheKey = `image:${prompt}:${model}`;
const cached = await redis.get(cacheKey);
if (cached) return cached; // Retour immédiat
```

9.3 Mobile-First Responsive Design

- Breakpoint minimum : 320px (petits smartphones)
- Touch-friendly : boutons min 44x44px (recommandation Apple/Google)
- Navigation bottom bar sur mobile (plus accessible que top nav)

- Formulaires simplifiés : gros inputs, autocomplete, validation inline
- Éviter les hover effects : inutiles sur mobile

10. Design & Identité Visuelle JadaRiseLabs

10.1 Charte Graphique

Élément	Valeur	Usage
Couleur primaire	#7B4F2E (Terre)	Titres, boutons primaires, logo
Couleur secondaire	#C9A84C (Or)	Accents, badges, highlights
Couleur accent	#2D6A4F (Vert savane)	Succès, validation, CTAs
Couleur highlight	#E76F51 (Terre cuite)	Alertes positives, créativité
Fond principal	#FDF6E3 (Crème)	Background général
Fond carte	#FFFFFF (Blanc)	Cards, modales, panels
Texte principal	#1A1A1A (Noir doux)	Corps de texte
Texte secondaire	#555555 (Gris)	Descriptions, légendes
Typographie titre	Plus Jakarta Sans ou Sora	Moderne, lisible
Typographie corps	Inter ou Nunito	Optimisée écrans

10.2 Composants UI — Design System

Utiliser shadcn/ui (composants Tailwind pré-construits) avec customisation couleurs JadaRiseLabs :

- Boutons : rounded-lg, transitions smooth (200ms), hover effect subtil
- Cards : border subtile, shadow légère, padding généreux
- Inputs : border focus avec couleur primaire, labels au-dessus
- Modales : overlay semi-transparent, animation fade-in
- Toasts/notifications : position top-right, auto-dismiss 5s

```
// tailwind.config.js - Couleurs JadaRiseLabs
module.exports = {
  theme: {
    extend: {
      colors: {
        earth: '#7B4F2E',
        gold: '#C9A84C',
        savanna: '#2D6A4F',
        terracotta: '#E76F51',
      },
    },
  },
};
```

10.3 Pages Clés — Structure & Priorités

Landing Page (priorité UX/marketing) :

- Hero section : titre accrocheur, sous-titre explicatif, CTA fort (Commencer gratuitement)
- Démo visuelle : GIF ou vidéo montrant une génération d'image en action
- Section modules : cartes présentant Image, Chat, Vidéo, Audio avec icônes
- Témoignages : 3-4 faux témoignages réalistes (en attente de vrais users)
- Pricing : tableau des 3 plans (Gratuit, Starter, Pro) avec comparaison
- FAQ : 5-7 questions fréquentes
- Footer : liens légaux, réseaux sociaux, contact

Dashboard (priorité productivité) :

- Header : logo, crédits restants, dropdown profil (logout)
- Stats : nombre générations ce mois, crédits utilisés, temps économisé
- Modules IA : cartes cliquables (Image, Chat, Vidéo, Audio) → redirige vers studio
- Générations récentes : 5 dernières créations avec preview + bouton télécharger
- CTA upgrade : si plan gratuit, promouvoir Starter/Pro

11. Sécurité, Conformité & Éthique IA

11.1 Sécurité Backend

- Variables d'environnement : jamais commiter les API keys (utiliser .env.local)
- HTTPS obligatoire : Vercel le fait par défaut (Let's Encrypt)
- Rate limiting : limiter appels API à 10/min par IP (express-rate-limit ou Vercel Edge)
- Validation inputs : vérifier tous les inputs user (longueur prompt, format fichiers)
- Row Level Security (RLS) : activer sur toutes tables Supabase (cf. section 4.2)
- CORS : autoriser uniquement votre domaine (config Next.js API routes)

11.2 Modération Contenu Généré

- Filtre NSFW : utiliser Hugging Face Content Moderation API (gratuit)
- Blocklist keywords : empêcher prompts avec mots-clés interdits (violence, haine, etc.)
- Watermark : ajouter 'JadaRiseLabs' en bas à droite sur créations gratuites
- Signalement : bouton 'Report' sur chaque génération (stocke dans table moderation)
- Review manuelle : si > 10 signalements, review admin manuel

11.3 Conformité Légale

- CGU (Conditions Générales d'Utilisation) : page /legal/terms
- Politique de confidentialité : page /legal/privacy (conforme RGPD même si AOF)
- Mentions légales : identité entreprise, contact, hébergement
- Droit à l'effacement : bouton 'Supprimer mon compte' dans profil
- Exportation données : bouton 'Télécharger mes données' (JSON de toutes générations)

12. Déploiement & Stratégie Post-Lancement

12.1 Checklist Pré-Déploiement

- Tests end-to-end : auth, génération, paiement, téléchargement
- Tests mobiles : iPhone SE (petit), iPhone 14, Android (Chrome)
- Tests connexion lente : throttling Chrome DevTools (Fast 3G)
- Vérifier env vars production : toutes les clés API sont bien dans Vercel
- Activer RLS Supabase : vérifier policies sur toutes tables
- Backup DB initial : export SQL Supabase avant lancement
- Monitoring : configurer Vercel Analytics + Sentry (erreurs JS)
- Domain custom : acheter jadalabs.com ou jadarise.io (Namecheap 10$/an)

12.2 Déploiement Vercel (procédure)

```
1. Connecter repo GitHub à Vercel
2. Framework preset : Next.js
3. Ajouter toutes les env vars (SUPABASE_URL, GROQ_API_KEY, etc.)
4. Deploy → Vercel build automatiquement depuis main
5. Configurer domain custom (DNS A record)
6. Activer Vercel Analytics (gratuit)
```

12.3 Post-Lancement — KPIs à Suivre

KPI	Objectif J+7	Objectif J+30	Outil
Inscriptions	100 users	500 users	Supabase Analytics
Générations totales	500	5000	Table generations
Taux de conversion freemium	2%	5%	Table subscriptions
Temps de génération image	< 15s	< 10s	Vercel logs
Taux d'erreur API	< 5%	< 2%	Sentry
NPS (satisfaction)	> 30	> 50	Formulaire in-app

12.4 Roadmap V2 (après MVP)

- Application mobile native (React Native) — iOS + Android
- API JadaRiseLabs pour devs tiers (B2B)
- Génération de présentations IA (style Gamma.app)
- Support langues africaines : Wolof, Haoussa, Dioula, Mooré
- Programme d'affiliation créateurs de contenu
- Intégration WhatsApp Business API (génération via chatbot)

13. Résumé Exécutif — L'Essentiel en 1 Page

Projet	JadaRiseLabs — Laboratoire IA Tout-en-1
Équipe	2 Devs Full-Stack (Front + Back en parallèle)
Délai	2-3 semaines (14-21 jours)
Modules MVP	P1: Image + Chat | P2: Vidéo + Audio | P3: Code
Stack	Next.js 14 + Supabase + Vercel + HF + Groq + CinetPay
Découpage	Dev 1: Auth, Chat, Vidéo, Crédits | Dev 2: Image, Galerie, Paiement
Git Strategy	Branches par module + PR review + merge dans develop
Objectif	**500 users J+30 | 20 abonnements payants J+30**

"Jada Rise — Construire l'Afrique de demain, aujourd'hui."

Document préparé avec ❤️ pour JadaRiseLabs — Confidentiel