



PRODUCT REQUIREMENTS DOCUMENT (PRD) :

AgentOps - Plateforme Micro-SaaS d'Automatisation IA pour Développeurs

1. Synthèse Exécutive (Executive Summary)

AgentOps est une plateforme micro-SaaS Laravel + React qui révolutionne le développement logiciel en fournissant un assistant IA autonome capable de générer, tester et déployer du code automatiquement. La plateforme s'adresse aux développeurs solo, freelances, CTOs de petites boîtes et équipes techniques de startups.

Proposition de valeur unique : Contrairement aux 1000+ assistants IA existants qui se limitent à l'autocomplétion de code, AgentOps orchestre l'intégralité du workflow de développement - de la compréhension du repo à la génération de tests, en passant par le déploiement automatique via CI/CD.

Modèle économique : Abonnement à 39 \$/mois (solo) ou 99 \$/mois (équipe), avec un objectif de 100 utilisateurs actifs et 20 payants générant 1 000 \$/mois de MRR en 90 jours.

Inspiration : Le projet s'inspire de la philosophie de Pieter Levels (@levelsio) : "Build once, automate forever" - capitaliser sur ses compétences existantes (Laravel, React, CI/CD, TDD), créer un produit unique sans marketing de masse, et itérer rapidement.

2. Objectifs et Métriques de Succès (Goals & Success Metrics)

Objectifs Business (3 mois)

- **Utilisateurs** : 1 000 inscrits bêta
- **Conversion** : 100 payants à 39 \$/mois
- **Revenus** : 3 900 \$/mois
- **Croissance** : 100 utilisateurs payants → 1 000 000 \$/ARR (objectif à 12 mois)

Objectifs Produit (90 jours)

- **J+30** : MVP fonctionnel et montrable avec authentification, gestion de projets, et connexion GitLab/GitHub
- **J+60** : Bêta publique via Product Hunt + Hacker News avec 100 utilisateurs actifs
- **J+90** : Produit rentable et autonome avec facturation Stripe activée et séquence d'emails IA (nurture)

KPIs de Succès

- **Adoption** : 10% de conversion bêta → payant
 - **Rétention** : < 1 000 \$ de dépenses initiales, 95% de chances de réussite
 - **Technique** :
 - 90% de temps de développement (automatisation workflow IA)
 - +40% de vitesse d'onboarding (Code Intelligence Map)
 - 60% de coûts API (LLM Router optimisé)
 - 70% de temps de review (Explain & Review IA)
 - **Traction** : 10 clients closés via campagne LinkedIn de 4 semaines (100 prospects ciblés)
-

3. Problème Résolu et Public Cible (Problem & Target Audience)

Problème Principal

Les développeurs disposent de milliers d'outils IA d'autocomplétion, mais aucun ne comprend véritablement leur codebase, ne génère de tests pertinents, ne lance de pipelines CI/CD ou ne déploie en production. Résultat : **ils travaillent encore à la ligne de code, pas au niveau du projet.**

Problématiques Spécifiques Résolues

1. **"Je ne peux pas commercialiser une application que je développe pour un client"** (contexte upEngage) → AgentOps offre une alternative 100% légale, rapide et alignée avec les compétences existantes.
2. **Manque de régularité de publication** (risque de traction nulle) → Solution : automatisation de la création de contenu Twitter via calendrier.
3. **Over-engineering du MVP** → Focus sur "Done > Perfect" avec release à J+30.
4. **Sous-monétisation du retard Stripe** → Bouton "Upgrade" dès le MVP.

Public Cible

Persona Principal : Pieter Levels-like Developer

- **Qui** : Créateur solo de Nomad List, Remote OK, Rebase, etc.
- **Caractéristiques** :
 - 100% indie hacker, aucun employé, aucun investisseur
 - Génère +3 M\$/an en revenus récurrents
 - Stack : PHP, JavaScript, Docker, Supabase, GPT, cronjobs
 - Philosophie : "Build once, automate forever"
- **Motivations** :
 - Expert Laravel/React, CI/CD, TDD → capitalise sur sa stack forte
 - Veut scaler sans lever de fonds
 - Cherche une niche forte (dev tools + IA automation) où les utilisateurs payent vite
 - Dans une niche forte où les utilisateurs paient vite

Segments Secondaires :

1. Développeurs solo, freelances (tarif : 39 \$/mois)
2. CTOs de petites boîtes (tarif : 99 \$/mois équipe)
3. Tarification : 39 \$/mois (solo) / 99 \$/mois (équipe)

Anti-Persona :

- Influenceurs startup cherchant du buzz
 - Personnes sans compétences techniques
 - Ceux qui attendent qu'on crée une idée à leur place
-

4. Spécifications des Fonctionnalités (Feature Specifications)

4.1. 🤖 Workflow IA Autonome "Code → Test → Deploy"

Description : Premier agent IA capable de livrer un commit complet, testé et déployé sans intervention humaine.

User Story :

En tant que développeur, je veux écrire une tâche naturelle (ex: "Add authentication to the API using Sanctum") et qu'AgentOps clone mon repo, analyse sa structure, génère le code, crée les tests PHPUnit, exécute la pipeline GitLab CI/CD et me notifie du résultat, afin de réduire mon sprint de 3 jours à 30 minutes.

Flux Technique :

1. Clone du repo (GitLab/GitHub)
2. Analyse de la structure et détection des dépendances via AST parser
3. Génération du code (Controller, Tests, Routes) via MCP (FastAPI) + Laravel Actions
4. Exécution des tests unitaires (PHPUnit, Jest, ESLint)
5. Push du commit sur branche `feature/ia-task-123` avec merge request automatique

6. WebSocket broadcasting via Laravel Echo pour suivi en temps réel

Impact Mesurable :

- Réduction de 90% du temps dev sur tâches répétitives
- Copilot ne fait pas ça : il ne comprend pas l'ensemble du projet ni ne déploie

Justification : C'est le différenciateur clé qui fait d'AgentOps un "orchestrateur de workflows IA complet" et non un simple plugin d'éditeur. Copilot, Cody et Codelum sont des "assistants de code" qui travaillent à la ligne, pas au niveau du projet.

4.2. 🧠 "Code Intelligence Map" – Vision Sémantique du Projet

Description : Graphe interactif généré automatiquement par IA pour comprendre instantanément les relations entre classes, services, modèles et migrations d'un projet Laravel.

User Story :

En tant que développeur rejoignant un projet existant, je veux visualiser instantanément les dépendances entre mes services, contrôleurs et modèles sous forme de graphe interactif, afin de réduire mon temps d'onboarding de plusieurs jours à quelques minutes.

Flux Technique :

- Parser AST (nikic/php-parser + Babel parser JS)
- Génération d'un graphe Neo4j-like rendu via React Flow
- Résumé contextuel via LLM ("Explain this service in plain English")

Impact Mesurable :

- +40% de vitesse d'onboarding
- Outil de compréhension instantanée pour équipes techniques (idéal pour onboarding, review, ou debug)

Justification : Visualiser un projet Laravel entier devient aussi simple que lire un schéma de base de données. C'est un outil de compréhension instantanée.

4.3. "TDD Copilot" – Générateur de Tests Intelligents

Description : Agent IA qui surveille les commits Sentry, analyse les endpoints non testés et génère automatiquement les fichiers PHPUnit/Pest/Jest correspondants.

User Story :

En tant que développeur ayant mergé du code sans tests, je veux qu'AgentOps détecte automatiquement les endpoints manquants, génère les tests unitaires via un prompt LLM contextualisé ("Generate a Pest test for this failing endpoint"), et me propose une PR automatique, afin d'atteindre +30% de couverture de test sans effort humain.

Flux Technique :

- Surveillance des commits Sentry + erreurs
- Analyse des endpoints non testés
- Génération automatique des fichiers PHPUnit/Pest/Jest via prompt LLM contextualisé
- Option "auto-commit test" (activable)

Impact Mesurable :

- Couverture de test +30% sans effort humain
- Tests plus pertinents car basés sur de vraies erreurs en production

Justification : Les tests sont souvent négligés par manque de temps. AgentOps les génère proactivement, basés sur des vraies erreurs en production.

4.4. "LLM Router" – Intelligence Multi-Modèles Automatique

Description : Service intelligent qui choisit automatiquement le meilleur LLM (GPT pour génération de code, Mistral pour refactor, Local Ollama pour devs auto-hébergés) selon la tâche, optimisant coûts et cohérence.

User Story :

En tant qu'utilisateur d'AgentOps, je ne veux pas avoir à choisir manuellement entre GPT-4, Mistral ou Ollama pour chaque action IA, car le système doit automatiquement router ma demande vers le modèle le plus performant et

économique selon le contexte (génération, refactor, test, documentation), afin de réduire mes coûts API de 60% tout en conservant une qualité optimale.

Concept :

Chaque action IA est routée dynamiquement :

- GPT pour génération de code
- Mistral pour refactor rapide
- Local (Ollama) pour devs auto-hébergés

Flux Technique :

- Service `LLMRouter` + Provider pattern
- Stockage du token usage (coût réel par job)
- Benchmarks automatiques
- Transparence : tableau des performances par modèle

Impact Mesurable :

- Réduction de 60% des coûts API
- Résultats plus fiables et cohérents
- Transparence : tableau des performances par modèle

Justification : L'utilisateur ne choisit plus, c'est toujours le meilleur modèle au meilleur coût. Argument rationnel : prévisibilité des coûts, ROI mesurable. Les entreprises adorent ça.

4.5. "Explain & Review" – Code Review IA Contextuelle

Description : Agent IA qui commente le code comme un senior dev, explique les intentions, détecte les failles possibles et les incohérences de logique, puis peut réécrire la PR entière si besoin.

User Story :

En tant que développeur solo ou membre d'une petite équipe sans QA ni lead technique, je veux qu'AgentOps analyse automatiquement mes Pull Requests, explique les intentions du code, détecte les failles de logique et propose des

améliorations concrètes (avec option de réécriture complète via "Explain this code to me like I'm new"), afin de gagner des heures de review et éviter les bugs en production.

Flux Technique :

- GitLab MR API
- Prompt LLM avec AST + diff contextuel
- Plugin VSCode (facultatif) relié à `/api/review`
- Option "Explain this code to me like I'm new" intégrée à l'IDE (extension optionnelle)

Impact Mesurable :

- Gain de plusieurs heures de review
- Parfait pour petites équipes sans QA ni lead technique

Justification : Une IA qui commente ton code comme un senior dev. Explique les intentions, les failles possibles, les incohérences de logique. Peut réécrire la PR entière si besoin.

4.6. Fonctionnalités Core Backend (Laravel 12 + PostgreSQL + Redis + Docker)

Services Principaux

- **GitProviderService** (GitLab/GitHub) : Connexion repos Git
- **MCPService** (FastAPI endpoint) : Moteur MCP pour génération IA
- **LLMService** (Mistral, OpenAI, Claude) : Routage intelligent multi-LLMs
- **Actions Laravel :**
 - `AnalyzeRepositoryAction`
 - `GenerateCodeAction`
 - `RunTestsAction`
 - `DeployPipelineAction`

- **Events :** `WorkflowStarted` , `CodeGenerated` , `TestFailed` , `Deployed`
- **TDD complet** avec PHPUnit + Pest
- **Logs dans PostgreSQL + Sentry**

Moteur IA (FastAPI/MCP)

- **Endpoints :**
 - `POST /analyze` : extrait la structure du repo
 - `POST /generate` : génère ou modifie du code
 - `POST /test` : exécute tests + coverage
 - `POST /deploy` : déclenche pipeline GitLab

Auth : Laravel Sanctum + Users + Teams

4.7. Fonctionnalités Core Frontend (React + Vite + Tailwind)

Dashboard Multi-Tenant

- Contexte utilisateur (équipe, rôle, permissions)

Pages Principales

- `/dashboard` → workflows & jobs
- `/repositories` → connexions GitLab/GitHub
- `/workflows/:id` → logs, étapes, résultats
- `/settings` → clés API, tokens LLM

Composants UI

- **Pricing + Billing page** (Stripe intégré)
- **UI clean (Tailwind Pro / shadcn)** : interface brute, markdowns, graphiques ASCII, prix clair
- **Landing page minimaliste** : formulaire + paiement Stripe avant le MVP complet

- **Onboarding "Créer un compte" + "Connecter GitLab"**
-

5. Exigences Non-Fonctionnelles (Non-Functional Requirements)

5.1. Sécurité

- **Auth** : Laravel Sanctum + authentification à deux facteurs (2FA) recommandée
- **Secrets** : Tokens GitLab/API stockés chiffrés en base
- **Permissions** : Multi-tenancy (Team ID) avec isolation stricte
- **Webhooks** : Signature vérifiée pour endpoints publics

5.2. Performance

- **Temps de réponse API** : < 200ms pour endpoints standards
- **Génération IA** : Feedback temps réel via WebSocket (Laravel Echo)
- **Cache** : Redis pour analyse de repos (évite re-parsing inutile)
- **Queue** : Laravel Horizon pour jobs asynchrones (génération code, tests, déploiement)

5.3. Scalabilité

- **Infrastructure** : Docker + PostgreSQL + Redis
- **Déploiement** : CI/CD GitLab (tests + lint + deploy Docker)
- **Hébergement** : DigitalOcean droplet (app_opentaps + nginx + PostgreSQL)
- **Coûts initiaux** : ~150 \$ (hébergement, domaine, OpenAI/Mistral API, outils design, ads/promotion)

5.4. Monitoring & Observabilité

- **Logs** : PostgreSQL + Sentry pour erreurs
- **Métriques** : Stats mensuelles sur X (transparence croissance organique)
- **Facturation** : Stripe fees + email automation + outils visuels/vidéos

5.5. Accessibilité & UX

- **Interface** : Clean, markdown, graphiques ASCII, pas de fioritures
 - **Transparence** : Simplicité extrême (bouton "Upgrade" dès MVP, pas de surprises)
 - **Mobile** : Responsive (Tailwind), pas d'app native requise
-

6. Dépendances et Risques (Dependencies & Risks)

6.1. Dépendances Techniques

- **APIs Externes** :
 - GitLab/GitHub API (connexion repos)
 - OpenAI/Mistral/Claude API (génération IA)
 - Stripe API (facturation)
- **Services Tiers** :
 - DigitalOcean (hébergement)
 - Laravel Sanctum (auth)
 - WebSocket (Laravel Echo)

6.2. Risques Identifiés

Risque	Impact	Probabilité	Mitigation
Manque de régularité de publication	Pas de traction Twitter	Élevée	Planifier 3 posts/semaine (calendrier Twitter automatisé via Typefully ou Hypefury)
Over-engineering du MVP	Produit jamais testé	Moyenne	Release J+30 même incomplet. "Done > Perfect"

Risque	Impact	Probabilité	Mitigation
Sous-monétisation du retard Stripe	Audience non convertie car absence de plan clair de pricing	Moyenne	Mettre le bouton "Upgrade" dès le MVP (même si tout n'est pas prêt)
Conflit de propriété intellectuelle (upEngage)	Impossibilité juridique et éthique de commercialiser	BLOQUANT (résolu)	AgentOps offre une alternative 100% légale, rapide et alignée
Coûts API LLM élevés	Burn rate insoutenable	Moyenne	LLM Router pour optimisation, monitoring strict des coûts
Adoption lente	MRR insuffisant	Moyenne	Campagne LinkedIn ciblée (100 prospects B2B, 4 semaines) + Product Hunt + Hacker News

6.3. Hypothèses Critiques




- Les développeurs Laravel/React recherchent activement des outils d'automatisation IA
- Le marché accepte un pricing 39-99 \$/mois pour un assistant dev IA complet
- Les early adopters tolèrent un MVP "brut mais fonctionnel"
- La stratégie "build in public" sur Twitter génère suffisamment de traction organique



7. Critères d'Acceptation (Acceptance Criteria)

7.1. Sprint 1 – Fondation (Semaine 1, Jours 1-7)

Objectif : Back + Front minimal viable pour login et gestion de projets

Livrables :

-  Init Laravel 12, Docker, PostgreSQL, Redis
-  Auth Sanctum + Users + Teams
-  Routes REST `/api/projects` , `/api/auth`

-  Services : `GitProviderService` (mock)
-  TDD : tests unitaires utilisateurs, login, création projet





Critères de Validation :

- Setup React/Vite/Tailwind
 - Login + Dashboard basique
 - Affichage des projets mockés
 - Tests : auth + création projet
 - Connexion / Création projet / Dashboard vide : fonctionnels
 - Déploiement Docker réussi en local
-




7.2. Sprint 2 – Connexions Git et LLM (Semaine 2, Jours 8-14)

Objectif : Relier le SaaS à GitLab et à un LLM



Livrables Backend :

-  `GitProviderRepository` + MCPService (stub)
-  Routes : `GET /api/repositories` (liste projets Git), `POST /api/repositories/sync`
-  Service `LLMService` : `generateCode(prompt, context)`
-  Tests : mocks Git + tests génération IA

Livrables Frontend :

-  Page "Connect Repository"
-  Formulaire d'intégration GitLab + token
-  UI "Ask Agent" (prompt → génération code mocké)



Critères de Validation :

-  Génération IA simulée à partir d'un repo
 -  Tests : génération + persistance job
 - Connexion GitLab fonctionnelle avec affichage des repos
 - Prompt IA génère du code (mocké ou via API réelle)
-




7.3. Sprint 3 – Workflow Engine (Semaine 3, Jours 15-21)

Objectif : Exécuter un vrai workflow (analyze → generate → test → deploy)



Livrables Backend :

-  Models : `Workflow` , `Step` , `Job` , `Log`
-  Actions :
 - `AnalyzeRepositoryAction`
 - `GenerateCodeAction`
 - `RunTestsAction`
 - `DeployPipelineAction`
-  Events + observers pour logging
-  WebSocket broadcasting via Laravel Echo

Livrables Frontend :

-  Page "Workflow Viewer" : timeline + logs
-  Statut live (WebSocket)
-  Actions : "Run Workflow" / "Cancel"





Critères de Validation :

-  Workflow complet mocké
-  Tests : séquence d'actions + broadcast + fail cases
- Workflow visible en temps réel dans l'interface
- Logs détaillés accessibles
- Possibilité d'annuler un workflow en cours





7.4. Sprint 4 – Déploiement & Monétisation (Semaine 4, Jours 22-30)

Objectif : Rendre l'application monétisable et publiable



Livrables Backend :

-  Multi-tenancy (Team ID)
-  Stripe Billing (via Cashier)
-  Endpoint public `/api/public/demo`
-  Endpoint webhook `stripe/webhook`




Livrables Frontend :

-  Pricing + Billing page
-  UI clean (Tailwind Pro / shadcn)
-  Landing page minimaliste
-  Onboarding "Créer un compte" + "Connecter GitLab"

Livrables Infra :

-  CI/CD GitLab (tests + lint + deploy Docker)
-  Hébergement : DigitalOcean droplet (app_opentaps + nginx + PostgreSQL)

Critères de Validation :

-  MVP complet & monétisable
-  Tests : paiement + multi-tenant
-  Démo prêt pour Product Hunt
- Paiement Stripe fonctionnel (test + prod)
- MVP déployé et accessible publiquement
- Feuille de route TDD (tests clés + coverage ciblé) respectée pour ces 4 sprints

7.5. Objectif Business (3 mois) – Post-MVP

Phase 1 – Construction minimaliste (Jours 1-30)

- But : avoir un MVP fonctionnel et montrable
- Dépenses : env. 500 \$ (hébergement + domaine + OpenAI/Mistral API + outils design + ads/promotion)

Phase 2 – Audience & feedback (Jours 31-60)

- But : attirer les premiers utilisateurs et itérer vite
- Actions :
 1. Construire ton audience "build in public" : 3 threads/semaine sur X (Twitter) → devlogs, mini-demos, transparence
 2. Vidéo Loom < 60s : montre ton produit
 3. Collecter des feedbacks : 20 personnes dans une communauté (Discord ou Notion) → testeurs
 4. Simplifier encore : retirer 50% du code inutile, automatiser déploiement (CI/CD)
 5. Lancer la bêta publique via Product Hunt + Hacker News
 6. Suivre 3 KPIs : visiteurs → inscrits → payants

Phase 3 – Monétisation & automatisation (Jours 61-90)

- But : rendre le produit rentable et autonome
- Actions :
 1. Activer la facturation Stripe : 39 \$/mois (Solo) / 99 \$/mois (Team)
 2. Mail d'onboarding automatique via Laravel Mail + Queue
 3. Automatiser les feedbacks (Google Form ou in-app modal)
 4. Créer une séquence d'e-mails IA (nurture) : 5 mails automatiques (découverte → conversion)
 5. Optimiser UX/UI minimalement : design clean, pas de fioritures
 6. Publier stats mensuelles sur X : transparence → croissance organique

Dépenses (env. 300 \$)

- Stripe fees
- Email automation
- Outils visuels / vidéos

Résultat attendu à J+90 :

Indicateur	Cible
100 utilisateurs actifs	10 % conversion
20 abonnés payants à 39 \$	≥ 780 \$/mois
ROI positif	< 1 000 \$ dépensés

7.6. Canaux Uniques de Distribution

Canal unique : LinkedIn outbound hautement personnalisé

Stratégie :

- Optimiser ton profil** : orienté "SaaS RH white-label pour consultants et startups"
- Lister 100 cibles B2B** (agences RH, SaaS, intégrateurs) avec Hunter.io + Sales Navigator
- Séquence de messages LinkedIn sur 5 jours** :
 - Jour 1 : Message de connexion ("J'ai bossé ↓ un moteur RH/IA prêt à intégrer, j'aimerais ton avis 5 min ?")
 - Jour 3 : Lien de démo + call Calendly
 - Jour 5 : Message vocal LinkedIn (très efficace)
 - Jour 7 : Lien de démo + call Calendly
- Clôture** : 1/10 t'achètera (en moyenne avec bonne démo)

Projection chiffrée :

Étape	Durée	Objectif	Résultat
Construction MVP white-label (basé sur upEngage)	2 semaines	API + UI personnalisable, prêt à démo	2 semaines
Campagne LinkedIn	4 semaines	100 prospects	10 clients closés
Recettes	Mois 2	50 000 \$ + 10 000 \$/mois récurrents	1 M \$ + < 10 mois

7.7. Canal unique : Twitter/X + Product Hunt + newsletter technique

Pourquoi c'est le canal naturel des devs et fondateurs indie :

- C'est le canal naturel des devs et fondateurs indie
- Tu capitalises sur ta stack (Laravel/React, TDD, CI/CD, IA) → "me déjà époustouflé"
- Tu gardes 100% de la propriété
- Tu peux itérer seul et rapidement
- Tu es dans une niche forte (dev tools + IA automation) où les utilisateurs paient vite

Stratégie simple :

1. Publier ton process chaque jour sur X (Twitter) :

- 3 posts/semaine (calendrier Twitter automatisé via Typefully ou Hypefury)
- Threads "build in public" : devlogs, mini-demos, transparence
- Vidéos Loom courtes (mini-features), chiffres, process, itérations
- Partages techniques : TDD, Docker, CI/CD, intégrations Notion/GitLab

2. Documenter la création d'AgentOps (ex : "Day 1 - building an AI ops agent that writes & deploys Laravel code automatically") :

- Exemple : "Day 3 of building AgentOps: I just made the first AI-generated test pass in the pipeline. Here's the Loom (60s)"

3. Publier 3 threads / semaine :

- Chiffres, process, itérations
- Transparence : "build in public" = audience gratuite + feedback instantané

4. Lance sur Product Hunt après 30 jours avec :

- Vidéo Loom < 60s : montre ton produit
- 3 KPIs : visiteurs → inscrits → payants

5. Publier stats mensuelles sur X : transparence → croissance organique

Ressources à étudier :

- 📱 Twitter @levelsio → il documente tout en temps réel
 - 🎤 Talk "Make Something People Want (and keep it simple)" de Pieter Levels
 - 📝 Blog : levels.io – Build Once, Automate Forever
 - 💻 Indie Hackers profile → revenus, stack, outils
-

7.8. Stratégie de Contenu Twitter pour 30 jours (pour te rendre visible et amorcer la traction)

Objectif : Élaborer un plan d'action simple sur 90 jours qui nécessite moins de 1 000 \$ d'investissement initial et qui a à 95% de chances de réussir.

Solution :

- Tu n'es pas besoin d'un "influenceur startup" ou d'un "guru IA"
 - Tu as déjà le niveau technique, la rigueur d'un architecte, et une vision produit claire
 - Ce qu'il te faut, c'est un modèle de réussite dans le "build → automate → scale" individuel, c'est-à-dire quelqu'un qui a transformé son expertise tech en business récurrent sans levée de fonds
 - *La personne numéro 1 dont tu devrais t'inspirer : Pieter Levels (@levelsio)**
-

8. Annexes

8.1. Architecture Technique Détaillée

Stack Technique Complète

Backend (Laravel 12 + PHP 8.4 + PostgreSQL + Redis + Docker)

- Laravel Sanctum (auth)
- FastAPI (pour le moteur MCP)
- React pour le front (manager)
- Services :

- `GitProviderService` (GitLab/GitHub)
- `MCPService` (FastAPI endpoint)
- `LLMService` (Mistral, OpenAI, Claude)
- Intégrations déjà testées : Notion, GitLab, OpenAI, Mistral

Frontend (React + Vite + Tailwind)

- Dashboard multi-tenant (contexte utilisateur)
- Pages principales : `/dashboard` , `/repositories` , `/workflows/:id` , `/settings`
- Infra : CI/CD GitLab (tests + lint + deploy Docker)

Moteur IA (FastAPI + MCP)

- Endpoints :
 - `POST /analyze` : extrait structure du repo
 - `POST /generate` : génère/modifie code
 - `POST /test` : exécute tests + coverage
 - `POST /deploy` : déclenche pipeline GitLab
- Connecté via WebSocket + API REST Laravel

Hébergement & Infrastructure

- **Hébergement** : DigitalOcean droplet
- **Configuration** : app_opentaps + nginx + PostgreSQL
- **CI/CD** : GitLab (tests + lint + deploy Docker)
- **Monitoring** : Logs PostgreSQL + Sentry

8.2. Plan de Production sur 4 Sprints (Feuille de Route TDD)

Le document détaille la feuille de route TDD (tests clés + coverage ciblé) pour ces 4 sprints, avec pour chaque sprint :

- Objectif clair
- Livrables backend et frontend

- Critères de validation testables

Résumé :

- **Sprint 1 (J1-7)** : Fondation - Backend minimal + Auth
 - **Sprint 2 (J8-14)** : Connexions Git et LLM
 - **Sprint 3 (J15-21)** : Workflow Engine complet
 - **Sprint 4 (J22-30)** : Déploiement & Monétisation
-

8.3. Modèle Business Détaillé

Tarification

- **Plan Solo** : 39 \$/mois (solo) ou 99 \$/mois (équipe)
- **Objectif** : 1 000 abonnés → 1 000 000 \$/ARR

Business Model Engage Core (Offre Alternative B2B)

Contexte : API SaaS RH B2B d'intégration RH/LLM (type "upEngage Core")
vendue en white-label

Caractéristiques :

- Une API SaaS RH + IA prête à intégrer
 - Vendue à des startups et agences RH en white-label
 - Plan mensuel white-label à 1 000 \$/mois (minimum 10 clients ciblés)
 - Frais d'intégration initiale : 5 000 \$
 - **Formule de revenus** :
 - 10 clients = 50 000 \$ upfront + 10 000 \$/mois récurrents
 - Projection : 1 M \$ + en < 10 mois
-

8.4. Positionnement et Argumentation Différenciatrice

Réponse Structurée : "Oui, il y en a 1000... mais aucun ne fait ça."

1. Reconnaître sans se justifier

"Oui, il y a des dizaines d'outils IA pour les développeurs. Mais ils ont tous le même défaut : ils travaillent à la ligne de code, pas au niveau du projet."

Sous-texte : Tu ne nies pas la concurrence, tu la recontextualises. Tu montres que tu comprends le marché, et que tu identifies la faille.

2. Distinguer ton positionnement fonctionnel

"AgentOps n'est pas un assistant de code. C'est un orchestrateur de workflows IA complet : il comprend ton repo, génère, teste, déploie et documente."

Positionnement clair :

- Copilot, Cody, Codelum = "assistants de code"
 - AgentOps = assistant de livraison
-

3. Distinguer ton positionnement philosophique

"Les autres outils t'aident à écrire plus vite. AgentOps t'aide à livrer sans douleur. C'est la différence entre un copilote et un opérateur autonome."

Message : C'est un changement de paradigme.

- Les autres = productivité micro (autocomplétion)
 - Toi = productivité macro (pipeline automatisé)
-

4. Distinguer ton approche technique

"Les autres outils sont fermés et centralisés. AgentOps est open, auto-hébergeable, modulaire et relié à ton CI/CD. Tu peux l'intégrer dans ton infra, pas dépendre d'une SaaS opaque."

Argument rationnel : C'est l'argument qui fera vibrer les CTOs, devs, et équipes sensibles à la souveraineté des données. Tu gagnes sur le terrain de la confiance technique.

5. Distinguer ton modèle économique

"Les autres te facturent des tokens à perte de vue. AgentOps optimise les coûts via un LLM Router et te montre chaque centime dépensé. Tu gardes le contrôle,

| pas la surprise."

Argument rationnel : Prévisibilité des coûts, ROI mesurable. Les entreprises adorent ça.

6. Distinguer ton ADN d'utilisateur

| "AgentOps est construit par un développeur qui «n» les mêmes galères. Pas par une startup qui vend du buzz autour de l'IA."

Sous-texte émotionnel : "Je construis l'outil que j'aurais voulu avoir."
C'est le langage Pieter Levels par.

8.5. Citation à Garder en Tête

Citation inspirante (Pieter Levels) :

| "If you're a developer, you already have superpowers. You just need to stop asking for permission and start shipping."
— Pieter Levels

8.6. Réponse Courte Type Pitch

Version ultra-concise (pour pitch ou landing page) :

| "Oui, il existe 1000 assistants IA. Mais aucun ne comprend ton codebase, ne génère des tests, ne lance les pipelines et ne push un commit en production. AgentOps ne code pas pour toi — il livre pour toi."

(pause, sourire)

| "Et c'est là que tout change."

8.7. Réponse Bonus si on te Challenge Fort (Investisseur, Tech Sceptique)

Question : "Tu as raison. Mais les 1000 outils IA existants viennent les développeurs isolés. AgentOps cible les équipes qui livrent. Et c'est un marché 10× plus gros."

Réponse structurée :






1. **Sous-texte** : Tu ne nies pas la concurrence, tu la recontextualises
 2. **Positionnement** : Copilot = assistant, AgentOps = orchestrateur
 3. **Différenciation philosophique** : Productivité micro vs macro
 4. **Approche technique** : Open, modulaire, souverain
 5. **Modèle économique** : Transparent, prévisible
 6. **ADN** : Construit par un dev pour des devs
-

9. Conclusion et Prochaines Étapes

Récapitulatif des Priorités

1. **Semaines 1-4** : Construction du MVP en suivant les 4 sprints définis
2. **Jours 31-60** : Lancement de la stratégie "build in public" sur Twitter/X
3. **Jour 60** : Lancement Product Hunt + Hacker News
4. **Jours 61-90** : Activation Stripe, optimisation conversion, campagne LinkedIn B2B

Métriques de Validation du PRD

-  MVP prêt à J+30
-  100 utilisateurs actifs à J+60
-  20 payants générant 780+ \$/mois à J+90
-  < 1 000 \$ investis au total
-  95% de chances de réussite (hypothèse conservatrice)

Validation Finale

Ce PRD sera considéré comme validé lorsque :

1. L'équipe d'ingénierie confirme la faisabilité technique des 4 sprints
2. Les critères d'acceptation de chaque sprint sont testables et mesurables

3. Le modèle business atteint le seuil de rentabilité à J+90
4. La stratégie de distribution génère au minimum 10 prospects qualifiés par semaine

Document préparé par : Product Manager (AI Assistant)

Date : 22 octobre 2025

Version : 1.0

Statut : En attente de validation équipe technique

Note finale : Ce PRD est basé exclusivement sur les spécifications fournies dans le document PDF. Aucune fonctionnalité n'a été ajoutée ou inventée. Toutes les sections reflètent fidèlement la vision, les objectifs et les contraintes du projet AgentOps tel que décrit dans le document source.