UPRISING NODE

Partner Command Center — Product Requirements Document (PRD)

Version: v1.1 • Date: 2026-01-25 • Propriétaire: Uprising Studio

But: supprimer toute friction entre un partenaire et l'exécution stratégique. Dépôt de leads, visibilité temps réel, génération d'audit IA, commissions, cash-out — et collaboration interne.

Esthétique: Command Center Lindy — noir/blanc, serif premium, espaces aérés, bordures 1px, zéro bruit.

TABLE DES MATIÈRES

- 1. Vision, objectifs et métriques
- 2. Personas, rôles et permissions (RBAC)
- 3. Portée: MVP vs V2
- 4. UX/UI: Design System Lindy
- 5. Architecture technique & environnements
- 6. Modèle de données (PostgreSQL/Prisma)
- 7. Fonctionnalités par page
- 8. Workflows critiques (Lead Drop → War Room)
- 9. API, Webhooks et Temps réel (Socket.IO)
- 10. Sécurité, conformité et anti-abus
- 11. Observabilité, tests et qualité
- 12. Dépendances & services externes
- 13. Guide d'installation (instructions pour un agent IA)
- 14. Annexes: Références
- 15. Collaboration multi-personnes & Chat temps réel

1. Vision, objectifs et métriques

Proposition: un portail partenaire B2B qui industrialise l'apport d'affaires et rend l'exécution (opérateur/IA) observable, mesurable et monétisable.

Objectifs (12 semaines)

- Lead Drop → Dossier War Room **10 minutes** (P95).
- Conversion Lead validé → Call confirmé ≥ **15%** (MVP).
- Aucune divergence Ledger vs Payout (0 incident).
- Dashboard **1.5s** (P75) desktop.

2. Personas, rôles et permissions (RBAC)

Partenaire: dépose des leads, suit, consomme analyses, cash-out.

Opérateur: qualifie, pilote pipeline, valide commissions, modère.

Admin: settings, monitoring, accès global.

3. Portée: MVP vs V2

MVP

- Auth + onboarding.
- Lead Drop + Kanban + War Room IA + PDF.
- Temps réel (updates pipeline + jobs).
- Ledger + cash-out (statuts).
- Assets (lecture seule).

V2

- Mode Expert.
- API Sandbox + quotas.
- Annotations PDF collaboratives versionnées.
- Leaderboard + anti-fraude.

4. UX/UI: Design System Lindy

- Monochrome strict: #FFF / #000 / séparateurs #EAEAEA.
- Titres serif; data/codes en Inter/Mono.
- Bordures 1px; espaces négatifs généreux; aucune déco.

5. Architecture technique & environnements

- Monorepo (pnpm).
- Web: Next.js App Router.
- API: NestJS + Socket.IO.
- DB: Postgres + Prisma.
- Queue: Redis + BullMQ.
- AI: FastAPI (scrape + analyse).

6. Modèle de données (PostgreSQL/Prisma)

Ledger double entrée (append-only).

Table	Champs essentiels	Notes
User	id, email, role	auth + email vérifié
Workspace	id, name	multi-projets / équipes
Membership	workspaceId, userId, role	RBAC workspace
Project	id, workspaceId, name, status	projet interne (delivery)
Lead	id, ownerId, projectId, url, status	lié à un projet
LeadAnalysis	leadId, pains, angles, scripts	JSON structuré
AuditPdf	leadId, storageKey, version	versionné
PipelineEvent	leadId, from, to, actorId	audit trail
Deal	leadId, amount, status	commission source
LedgerEntry	partnerId, dealId, type, amount	append-only
PayoutRequest	partnerId, amount, status	Requested/Processing/Paid
Channel	id, projectId, kind	chat rooms par projet/lead
Message	channelId, senderId, body, attachmentKey	persisté + modération
ReadReceipt	messageId, userId, readAt	statut lecture

7. Fonctionnalités par page

- Welcome / Features / Simulateur de gains.
- Auth (OTP).
- Dashboard: Lead Drop, Pipeline, War Room, Gains, Assets, API, Settings.
- Pipeline Kanban: Prospect → Audit → Pitch → Signé (+ Rejeté).
- War Room: analyse IA + PDF + validation finale.
- Gains: potentiel/earned/withdrawn + ledger + payouts.

8. Workflows critiques (Lead Drop → War Room)

- Jobs BullMQ: scrape → extract → enrich → ai_analysis → generate_pdf → notify.
- State machine: NEW → ENRICHING → ANALYZING → PDF_GENERATING → READY / FAILED.
- Retries expo (max 3), timeouts stricts, circuit breaker domaine.

9. API, Webhooks et Temps réel (Socket.IO)

- REST pour CRUD; Socket.IO pour updates (pipeline, jobs, payouts, chat).
- Scaling multi-instances: adapter Redis.
- Webhooks signés HMAC optionnels pour READY/PAID.

10. Sécurité, conformité et anti-abus

- OTP rate limit + blocage IP suspecte.
- API keys hashées, quotas, rotation.
- Ledger append-only; corrections via entrées compensatoires.
- Modération chat: anti-spam + suppression logique + audit log.

11. Observabilité, tests et qualité

- Logs JSON + traceId/leadId.
- Sentry (front/back).
- Metrics: latence, durée jobs, queues, taux échec, SLA P95.
- E2E: login → lead drop → war room ready → chat.

12. Dépendances & services externes

- Socket.IO + Redis adapter (scaling).
- BullMQ + Redis.
- Prisma + Postgres.
- FastAPI + Playwright/Puppeteer.
- Option collaboration managée: Liveblocks ou Supabase Realtime.

13. Guide d'installation (instructions pour un agent IA)

Toujours installer les dépendances avant de lancer quoi que ce soit.

```
# Infra
docker compose up -d

# Node deps (root)
pnpm install

# DB
pnpm --filter api prisma:generate
pnpm --filter api prisma:migrate

# AI deps
cd apps/ai
python -m venv .venv
source .venv/bin/activate # Windows: .venv\Scripts\activate
pip install -r requirements.txt

# Run
pnpm dev
```

14. Annexes: Références

- Docs Next.js, NestJS, Prisma, BullMQ/Redis, Socket.IO, FastAPI.

15. Collaboration multi-personnes & Chat temps réel

Objectif: permettre à une équipe (opérateurs + admins + partenaires autorisés) de travailler sur le même lead/projet, de se coordonner en direct et de laisser des traces exploitables (audit trail).

15.1 Cas d'usage (MVP)

- **Chat par contexte**: un channel par projet + un channel par lead (War Room).
- **Présence**: voir qui est en ligne et qui regarde la même War Room (liste + statut).
- **Typing**: indicateur de saisie en temps réel (sans persistance).
- **Mentions**: @user déclenche notif (in-app + email optionnel).
- **Pièces jointes**: lien vers PDF audit / images (stockage objet).
- **Read receipts**: 'vu' par utilisateur (optionnel MVP, utile V2).

15.2 Cas d'usage (V2)

- **Commentaires ancrés** sur sections du PDF + résolution (workflow review).
- **Notes collaboratives** (éditeur temps réel) pour synthèse opérateur.
- **Threading** (réponses) + épinglage + recherche.
- **Règles de confidentialité**: certains channels visibles uniquement opérateurs.

15.3 Architecture recommandée (self-host)

- Temps réel: **Socket.IO** (namespace / rooms par workspace/project/lead).
- Scaling horizontal: **Redis adapter** pour diffuser entre plusieurs serveurs.
- Persistance: Messages en Postgres (Message table) + index full-text si besoin.
- Événements éphémères: typing/presence via Socket.IO memory + TTL.
- Upload: stockage objet (S3/GCS) + URL signée pour download.

Rooms & naming

- room:workspace:{workspaceId}
- room:project:{projectId}
- room:lead:{leadId}
- room:channel:{channelId}

15.4 Alternative managée (accélérer)

- **Supabase Realtime**: Broadcast (messages éphémères), Presence, et events sur changements Postgres.
- **Liveblocks**: building blocks prêts (presence, comments, notifications) si tu veux aller vite sur commentaires/annotations.

15.5 Collaboration sur documents/notes

- Si besoin d'édition simultanée (notes, checklist, doc): utiliser un CRDT type **Yjs**.
- Transport: y-websocket ou ton transport Socket.IO; persistance via un store (Redis/Postgres) si nécessaire.

15.6 Spécification temps réel (événements)

- chat.message.new (channelId, messageId, body, senderId, createdAt)
- chat.message.edited (messageId, body, editedAt) — optionnel
- chat.message.deleted (messageId, deletedAt) — suppression logique
- presence.update (scope, users[])
- typing.start / typing.stop (channelId, userId)
- comment.created / comment.resolved (V2, sur PDF ou note)

15.7 Anti-abus & modération

- Rate limit par user/channel (ex: 5 msg/sec burst).
- Filtrage contenu (liens) + pièces jointes whitelist.
- Suppression logique + audit log (qui a supprimé, pourquoi).

15.8 Dépendances (si self-host Socket.IO)

```
pnpm --filter api add socket.io @socket.io/redis-adapter redis
pnpm --filter web add socket.io-client
# Option notes collaboratives
pnpm --filter web add yjs
```

15.9 Dépendances (si Supabase / Liveblocks)

```
# Supabase
pnpm --filter web add @supabase/supabase-js

# Liveblocks (React)
pnpm --filter web add @liveblocks/client @liveblocks/react
```