UPRISING NODE

Assets & Setup Starter — Kit de production + installation des dépendances

Version: v1.0 • Date: 2026-01-25

Objectif: te donner un plan exécutable pour (1) créer les assets nécessaires (UI/brand) et (2) démarrer le projet correctement (dépendances, monorepo, infra locale) — en visant **desktop d'abord** mais compatible mobile dès le départ.

SOMMAIRE

1. Inventaire des assets (obligatoire)

Avant d'écrire 5000 lignes de code, tu dois verrouiller les assets. Sinon chaque écran devient différent.

1.1 Brand kit (Lindy Monochrome)

- Logo: version primaire (noir) + inverse (blanc) + favicon (simple).
- Couleurs: #FFFFFF (fond), #000000 (texte/actions), #EAEAEA (dividers), #666666 (muted).
- Typographies: titres Serif (Playfair/Bodoni) + UI Sans (Inter) + Mono (Roboto Mono).
- Iconographie: set minimal (12–20 icônes) cohérent.

1.2 UI kit (Design System)

- Composants: ButtonBlack, MinimalInput, DataCard, SerifTitle, Divider, Toast, Modal.
- États: default/hover/active/disabled + loading skeletons.
- Patterns: table, kanban column, empty states, error states.

1.3 Assets fonctionnels (produit)

- PDF templates (Audit): header, sections, grids, disclaimers.
- Graphiques: style noir/blanc (courbes simples) + axes lisibles.
- Avatars: fallback (initiales) + présence (dot).
- Marketing assets (Assets Library): decks, scripts, guides (PDF).

2. Structure Figma (UI Kit Lindy)

Un seul fichier 'Uprising Node UI Kit'. Pas de fichiers dispersés.

- **Pages Figma**: 00-Cover • 01-Tokens • 02-Components • 03-Patterns • 04-Screens • 05-PDF Templates.
- **Tokens**: couleurs, typo scale, spacing, border (1px), radius (0–12), shadows (0).
- **Components**: variants (size/state) et constraints (responsive).
- **Screens**: Welcome, Auth, Dashboard, Lead Drop, Pipeline, War Room, Gains, Chat.

```
Naming (exemples)
- c/Button/Primary
- c/Input/Minimal
- p/Kanban/Column
- s/Dashboard/LeadDrop
```

3. Règles d'export et naming (dev-ready)

Règle: tout ce qui est exporté doit être utilisable sans retouche par un dev.

3.1 Icônes

- Format: SVG (stroke 1.5–2).
- Nom: icon-name.svg (kebab-case).
- Couleur: currentColor (pas de noir fixe), pour hériter du texte.

3.2 Images

- Format: PNG (si nécessaire) sinon SVG.
- Résolutions: 1x/2x.
- Nom: feature-xxx@2x.png

3.3 PDF templates

- Concevoir en grille A4: marges constantes + sections répétables.
- Prévoir variables: companyName, url, pains[], angles[], scripts[].
- Exporter une version 'spec' (exemple rempli) et une version 'blank'.

4. Responsive: desktop-first sans casser mobile

Tu construis desktop d'abord. Mais tu n'écris pas du CSS qui casse mobile. Le web doit rester fluide sur téléphone.

- Approche: layout desktop (large) puis dégrader proprement.
- Toujours définir une base simple (mobile) puis ajouter des améliorations en md/lg.
- Navigation: sidebar desktop → drawer mobile (V2 si tu veux aller vite).
- Kanban: desktop = colonnes; mobile = liste par statut (toggle).

```
Tailwind mindset:
- base = s'applique partout
- md:... = amélioration desktop
Ex: className="px-4 md:px-8"
```

5. Setup du repo (monorepo) + dépendances

Stack: Next.js (Web), NestJS (API), BullMQ workers (Worker), FastAPI (AI), Postgres+Redis (Infra).

5.1 Pré-requis machine

- Node.js LTS + pnpm
- Python 3.11+ (venv)
- Docker Desktop (compose)
- Git

5.2 Structure monorepo (recommandée)

```
/uprising-node
/apps
/web (Next.js)
/api (NestJS)
/worker (BullMQ workers)
/ai (FastAPI)
/packages
/ui (Design System)
/shared (types + schemas)
/infra
docker-compose.yml
```

6. Infra locale (Postgres + Redis) + Prisma

Tu dois pouvoir boot l'écosystème en 1 commande (compose).

```
# infra
docker compose up -d

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

- DATABASE_URL pointe vers Postgres (compose).
- REDIS_URL pointe vers Redis (compose).
- Prisma: migrations versionnées, jamais de changements manuels en prod.

7. Services applicatifs (Web / API / Worker / AI)

Démarrage local: 4 services, 1 source de vérité (API + DB).

7.1 Web (Next.js)

```
# création web (si pas encore)
pnpm dlx create-next-app@latest web --typescript --eslint --app

# tailwind (suivre guide officiel)
# puis lancer:
pnpm --filter web dev
```

7.2 API (NestJS)

```
# nest cli
npm i -g @nestjs/cli
nest new api

# lancer:
pnpm --filter api start:dev
```

7.3 Worker (BullMQ)

```
# deps
pnpm --filter worker add bullmq ioredis
pnpm --filter worker start
```

7.4 AI (FastAPI + Playwright)

```
cd apps/ai
python -m venv .venv
source .venv/bin/activate
pip install fastapi uvicorn playwright
playwright install
fastapi dev main.py
```

8. Starter checklists

8.1 Checklist assets (avant dev)

- Logo + favicon validés.
- Tokens figma (couleurs/typo/spacing) figés.
- Composants core (button/input/card) avec états.
- Navigation sidebar (desktop) + empty/error states.
- Icônes exportées en SVG currentColor.
- Template PDF audit (spec rempli + blank).

8.2 Checklist setup (avant de coder les flows)

- docker compose up (postgres/redis OK).
- pnpm install à la racine (monorepo).
- prisma migrate OK (DB).
- web/api/worker/ai démarrent sans erreur.
- health endpoints: API + AI répondent.
- socket.io: connexion client OK.

8.3 Checklist mise en ligne (plus tard)

- Web: déploiement (Vercel conseillé) + variables d'env.
- API/Worker/AI: déploiement sur un runtime (Railway/Fly/Render) + Redis + Postgres managés.
- Storage: S3/GCS pour PDFs et assets, URLs signées.
- Sentry activé + logs centralisés.