## **Stack Technique Complète SolarPerform V1**

### **Backend**

| **Outil / Framework** | **Usage** |
| --- | --- |
| **AdonisJS (v6)** | Framework backend principal (TypeScript, opinionated) |
| **Lucid ORM** | ORM SQL intégré pour PostgreSQL (multi-schema OK) |
| **JWT Auth** | Authentification intégrée + middleware |
| **Zod** ou Validator | Validation des payloads, requêtes |
| **Axios** (inter-service) | Communication entre microservices (si besoin) |
| **Winston / Pino** | Logging structuré |
| **Docker** | Conteneurisation de chaque service |
| **Docker Compose** | Orchestration locale + CI |
| **Prisma (optionnel)** | Pour un service technique non-Adonis (files) |
| **Node-cron** / worker | Pour le files-service, SFTP, traitement en background |

**Communication entre services**

| **Type** | **Stack** |
| --- | --- |
| REST | API Gateway avec Adonis (/api/\*) |
| Auth | JWT (Authorization: Bearer) |
| Données IoT | MQTT (via mqtt.js) |
| Fichiers | SFTP (via ssh2-sftp-client) |

### 

### 

### 

### **Base de données**

| **Base** | **Stack** |
| --- | --- |
| **PostgreSQL** (1 instance) | Base centralisée avec schémas par service |
| ORMs | Lucid pour Adonis, Prisma pour tâches techniques (optionnel) |
| Séparation | auth.users, user.devices, monitoring.measurements, etc. |
| Init / Migrations | Via Adonis CLI ou SQL brut partagé |

### **Frontend**

| **Outil / Lib** | **Usage** |
| --- | --- |
| **React** (Next.js) | Framework frontend (SPA + SSR possible) |
| **TypeScript** | Langage front sécurisé |
| **TailwindCSS** | Styling rapide, scalable |
| **shadcn/ui** | Composants UI pro + accessibilité + thème intégré |
| **Recharts** | Graphiques (courbes, barres, donut) pour dashboard |
| **React Hook Form + Zod** | Formulaires typés & validés |
| **TanStack Query** | Appels API + cache + loading states |
| **MQTT.js** (optionnel) | Client temps réel si dashboard live |
| **Figma** | Conception UI/UX, design system partagé avec l’équipe |

### **DevOps / Qualité**

| **Outil** | **Usage** |
| --- | --- |
| **Docker** | Conteneurisation des services |
| **docker-compose** | Orchestration locale (dev/prod) |
| **GitHub Actions** | CI/CD : tests, lint, déploiement auto |
| **Eslint + Prettier** | Code quality (TS) |
| **Husky + Commitlint** | Git hooks pour les bonnes pratiques |
| **Dotenv** | Environnement multi-env (.env.dev, .env.prod) |
| **Monitoring** | Prometheus, Grafana, Sentry, Logtail |

**Sécurité**

| **Type** | **Implémentation** |
| --- | --- |
| Auth | JWT (dans Adonis) + refresh tokens |
| API | Middleware auth, role, owner-only... |
| RGPD | Routes /me, /me/delete, politique claire |
| Base de données | Droits limités, hash bcrypt, logs d’accès |
| Docker | Isolation réseau entre services |

### **Structure du projet (monorepo)**

solarperform/

├── backend/

│ ├── auth-service/ # Adonis

│ ├── user-service/ # Adonis

│ ├── monitoring-service/ # Adonis

│ ├── files-service/ # Node.js / Worker

│ ├── mqtt-service/ # Node.js / MQTT listener

│ ├── sftp-service/ # Node.js / SSH handler

│ └── gateway/ # Adonis API Gateway

├── frontend/ # Next.js + React + Tailwind

├── shared/ # Types, utils, constantes

├── docker/

│ └── postgres/, mqtt/, sftp/ (configs spécifiques)

├── docker-compose.yml

└── README.md

### 

### 

### **Hébergement & Infrastructure**

| **Élément** | **Description** |
| --- | --- |
| **Serveur principal** | **VPS OVH – Ubuntu 22.04** |
| **Conteneurisation** | Docker + Docker Compose |
| **Nom de domaine** | géré via OVH (ou Cloudflare) |
| **Accès SSH sécurisé** | Utilisateur solaradmin + clés SSH |
| **Dossiers montés** | /home/solaradmin/solarperform/ |
| **Services tournant sur le VPS** |  |
| **PostgreSQL** | base centralisée, stockée en volume Docker |
| **MQTT (Mosquitto)** | broker IoT pour messages temps réel |
| **SFTP (Atmoz)** | pour réception de fichiers CSV |
| **Backend (AdonisJS)** | tous les microservices via Docker |
| **Frontend (Next.js)** | buildé, servi via Nginx |
| **Reverse Proxy** | **Nginx** + Let’s Encrypt certbot (SSL) |

### **Sécurité & bonnes pratiques**

| **Élement** | **Recommandation** |
| --- | --- |
| SSH | Clés SSH uniquement, accès root désactivé |
| SSL | Certificat automatique via **Let’s Encrypt** |
| Docker | Réseaux isolés + volumes persistants |
| Base de données | PostgreSQL exposé uniquement en interne |
| Backup | Cron job ou GitHub Actions pour pg\_dump + volume SFTP |

### 

### 

### **Déploiement**

| **Étape** | **Outils** |
| --- | --- |
| Build des images | GitHub Actions (build & push) |
| Déploiement | SSH, scp, rsync ou runner local |
| Env. prod | .env.production, secrets injectés |
| Redémarrage | docker compose up -d |

### **Exemple d’arborescence serveur**

/home/solaradmin/solarperform/

├── docker-compose.yml

├── .env.production

├── nginx/

│ └── solarperform.conf

├── apps/

│ ├── auth-service/

│ ├── user-service/

│ ├── files-service/

│ └── ...

├── frontend/

├── data/

│ ├── postgres/

│ ├── mqtt/

│ └── sftp/