Une architecture native, claire et maintenable

Routes, contrôleurs, services, modules déclaratifs et stockage JSON transactionnel forment le cœur natif de Portix V3.

Un cœur PHP natif

Portix V3 fonctionne avec un contrôleur frontal, des routes explicites, des contrôleurs spécialisés et des services métier. Cette organisation évite de mélanger l’affichage, les règles fonctionnelles et l’accès aux données.

Des modules isolés

Chaque service dispose d’une déclaration, de routes, de modèles et de ressources propres. Le registre des modules détermine ce qui est visible, actif et accessible depuis l’administration.

Principes techniques

- routage centralisé avec alias de compatibilité ;
- gabarits séparés des contrôleurs ;
- services réutilisables pour les opérations sensibles ;
- stockage JSON avec écritures temporaires et remplacement atomique ;
- répertoires privés protégés contre l’accès Web direct ;
- journalisation des actions techniques et de sécurité.

Compatibilité sans dépendance SQL obligatoire

Le portail peut fonctionner sans serveur de base de données imposé. Les fichiers JSON sont structurés, sauvegardés et migrés par schéma lors du premier démarrage d’une nouvelle version.

Gabarits et sorties

Les pages publiques, l’administration, l’impression et les PDF partagent des composants cohérents. Les articles natifs sont désormais maintenus comme une documentation courante, sans blocs de changelog répétitifs, et restent synchronisés sans écraser les publications personnalisées.

MaintenanceUne nouvelle version met à niveau le code et les données par une migration versionnée, sans réécrire les contenus personnalisés qui ne correspondent pas aux articles natifs.

Préparer un environnement cohérent

L’installation doit disposer d’une version PHP compatible, des extensions requises, d’un accès en écriture limité aux répertoires de stockage et d’un serveur Web correctement configuré. Les fichiers applicatifs et les données ne doivent pas être mélangés. Cette séparation facilite les mises à jour, les sauvegardes et le diagnostic en cas d’erreur.

Comprendre l’organisation de Portix

Le cœur de Portix reçoit la requête, identifie la route, applique les contrôles d’accès puis

appelle le contrôleur correspondant. Les services regroupent les règles métier et les gabarits se chargent de l’affichage. Les modules ajoutent leurs propres routes et fonctions sans modifier directement les données des autres composants.

Mettre à jour sans perdre les données

Les archives d’importation doivent contenir uniquement les fichiers autorisés et un manifeste conforme. Les données locales, comptes, contenus et réglages restent en dehors du paquet automatique. Une mise à jour sérieuse commence par une sauvegarde, vérifie la version de départ, contrôle les empreintes puis déclenche les migrations prévues pour la version cible.

Contrôles après installation

Une fois Portix installé, il convient de tester la première connexion, le changement du mot de passe provisoire, l’accès à l’administration, les courriels, les PDF, les tâches planifiées et les principaux modules. Les permissions du système de fichiers et les journaux PHP doivent également être vérifiés avant l’ouverture du portail aux utilisateurs.

Des manifestes vérifiés plutôt que supposés

Le routage modulaire repose sur des déclarations qui doivent rester synchronisées avec le manifeste runtime réellement chargé. Les contrôles récents comparent désormais ces sources afin d’éviter qu’une route existe dans un module mais soit absente en production. Cette vérification est particulièrement importante pour les routes dynamiques de paiement et d’administration.

Le contenu natif suit aussi les versions

Les articles et éditoriaux officiels sont maintenus dans un snapshot de distribution. À partir de cette révision, le changement de version peut resynchroniser les contenus natifs gérés par Portix tout en laissant en place les publications personnelles créées par l’administrateur.