📈 Plan de Refactorisation Sacré

🎯 Objectif Global

Transformer notre architecture en une base de code sacrée, maintenable et évolutive.

🗺️ Plan d'Action en 7 Phases

Phase 1: Centralisation du Code Partagé

1.1 Types Globaux (`packages/types/`)

- [ ] Identifier tous les types dupliqués
- [ ] Créer une hiérarchie claire des types
- [ ] Migrer vers un système de types unique

1.2 Services (`packages/services/`)

- [ ] Centraliser les appels API
- [ ] Standardiser la gestion des erreurs
- [ ] Implémenter une couche d'abstraction uniforme

1.3 Hooks (`packages/hooks-shared/` & `packages/hooks-auth/`)

- [ ] Auditer l'utilisation des hooks
- [ ] Éliminer les duplications
- [ ] Documenter les cas d'usage

1.4 Contextes (`packages/contexts/`)

- [ ] Unifier les contextes globaux
- [ ] Documenter la hiérarchie des contextes
- [ ] Optimiser les re-renders

PROF

1.5 Composants UI (`packages/ui-core/`)

- [ ] Identifier les composants communs
- [ ] Créer une bibliothèque UI unifiée
- [ ] Mettre en place un système de design tokens

Phase 2: Automatisation du Typage

2.1 Génération des Types TypeScript

- [ ] Installer `datamodel-code-generator`
- [ ] Configurer la génération depuis Pydantic
- [ ] Intégrer dans le pipeline de build

2.2 Schémas Zod

- [ ] Configurer la génération depuis TypeScript
- [ ] Valider la compatibilité bidirectionnelle
- [ ] Mettre en place des tests de type

2.3 Configuration des Formulaires

- [ ] Automatiser la génération des configs
- [ ] Valider les schémas générés
- [ ] Documenter le processus

Phase 3: Nettoyage & Dette Technique

3.1 Analyse du Code Mort

- [ ] Configurer ts-prune
- [ ] Identifier les imports inutilisés
- [ ] Nettoyer les dépendances obsolètes

3.2 Tests de Couverture

- [ ] Analyser la couverture backend
- [ ] Identifier les zones critiques
- [ ] Augmenter la couverture des tests

3.3 Linting & Format

- [ ] Standardiser les règles ESLint
- [ ] Configurer Prettier
- [ ] Mettre en place les hooks pre-commit

PROF

Phase 4: Sécurité & Backend

4.1 Authentification

- [ ] Auditer le middleware JWT
- [ ] Renforcer la validation des tokens
- [ ] Documenter les flux d'auth

4.2 Autorisations

- [ ] Implémenter RBAC complet
- [ ] Tester tous les cas d'usage
- [ ] Documenter les rôles et permissions

4.3 Configuration

- [ ] Sécuriser la gestion des secrets
- [ ] Optimiser la configuration NGINX
- [ ] Documenter les best practices

Phase 5: Infrastructure Docker

5.1 Composition des Services

- [ ] Optimiser docker-compose.yml
- [ ] Implémenter les healthchecks
- [ ] Configurer les réseaux isolés

5.2 Persistance

- [ ] Configurer PostgreSQL
- [ ] Optimiser Redis si nécessaire
- [ ] Documenter la gestion des données

5.3 Configuration

- [ ] Centraliser les variables d'env
- [ ] Versionner les images Docker
- [ ] Documenter le déploiement

Phase 6: Documentation

6.1 Architecture

- [ ] Documenter les décisions techniques
- [ ] Maintenir le changelog
- [ ] Créer des guides de contribution

6.2 API

PROF

- [ ] Générer la doc OpenAPI
- [ ] Documenter les endpoints
- [ ] Créer des exemples d'usage

6.3 Développement

- [ ] Mettre à jour les README
- [ ] Documenter le setup local
- [ ] Créer des guides de débogage

Phase 7: CI/CD

7.1 Tests Automatisés

- [ ] Configurer la CI GitHub Actions
- [ ] Implémenter les tests e2e
- [ ] Mettre en place les rapports

7.2 Qualité de Code

- [ ] Configurer SonarQube
- [ ] Implémenter les gates de qualité
- [ ] Automatiser les revues

7.3 Déploiement

- [ ] Automatiser les releases
- [ ] Configurer les environnements
- [ ] Documenter le processus

📊 Métriques de Succès

- [ ] Couverture de tests > 80%
- [ ] Zéro duplication de code
- [ ] Documentation à jour
- [ ] CI/CD stable
- [ ] Temps de build < 10 minutes

🎓 Formation

- [ ] Sessions de formation équipe
- [ ] Documentation des processus
- [ ] Guides de contribution

📝 Notes

PROF

- Priorité à la stabilité
- Tests obligatoires
- Documentation continue
- Communication transparente