ProEvalis — Document Technique Frontend

Objectif du document

Ce document constitue **la référence unique** pour aligner le frontend ProEvalis avec :

- le cahier des charges (client & gérant du projet),
- l'API backend (multi-tenant, JWT, rôles & permissions),
- l'architecture TanStack/React moderne.

Il regroupe **sans omission** :
1. l'architecture cible frontend,
2. le mapping Cahier des charges → modules frontend,
3. le plan de refactorisation fichier par fichier.

1 Architecture cible frontend (validée)

Principes directeurs

- Séparation stricte **UI / Logique / Données**
- Aucun appel API dans les composants
- Multi-tenant géré **au niveau infrastructure** (interceptors)
- Rendu conditionnel **par rôle & permission**
- Alignement natif avec TanStack Router & Query

Contexte du projet

ProEvalis est une plateforme SaaS de suivi-évaluation multi-tenant. Chaque organisation (tenant) est isolée et dispose de son propre administrateur global, capable de gérer utilisateurs, rôles, permissions et projets.

3. Stack technique imposée

- React 19

- TypeScript strict

- TanStack Router & Query

- Zustand (auth uniquement)

- Tailwind CSS

4. Problèmes identifiés

- Mélange UI / logique métier

- Auth et tenant non centralisés

- Absence de layouts conditionnels par rôle

- Priorité excessive au dashboard management

- Présence de mocks et fichiers inutiles

5. Architecture cible

Une architecture modulaire par domaines (auth, tenant, users, projects, dashboard) avec des layouts conditionnels et des guards centralisés.

6. Règles strictes

- Pas de logique métier dans les composants UI

- Pas de fetch direct dans les pages

- Auth et tenant via providers globaux

- RBAC géré au niveau routing

7. Plan de refactorisation

1. Auth & Tenant

2. Providers globaux

3. Layouts & Header

4. Routing & Guards

5. Dashboard Admin

6. Users / Roles / Permissions

7. Projects & Management

8. Nettoyage des mocks

Arborescence cible

```
src/
├── app/
│   ├── providers/
│   │   ├── auth.provider.tsx
│   │   ├── tenant.provider.tsx
│   │   └── query.provider.tsx
│   ├── layouts/
│   │   ├── AuthLayout.tsx
```

```
│   │   ├── AdminLayout.tsx
│   │   ├── ClientLayout.tsx
│   │   └── ManagementLayout.tsx
│   └── router.tsx
│
├── domains/
│   ├── auth/
│   │   ├── api/
│   │   ├── hooks/
│   │   ├── schemas/
│   │   └── types.ts
│   ├── users/
│   ├── projects/
│   ├── dashboard/
│   │   ├── admin/
│   │   ├── manager/
│   │   └── client/
│   ├── scrum/
│   ├── planning/
│   └── roles-permissions/
│
├── shared/
│   ├── ui/        # composants visuels purs
│   ├── components/ # header, navigation
│   ├── hooks/     # hooks génériques
│   └── utils/
│
├── pages/         # pages fines (assemblage uniquement)
├── services/      # API http + interceptors
├── store/         # états globaux (auth, tenant)
└── types/
```

Compatibilité TanStack

✅ Cette architecture **ne rentre pas en conflit** avec TanStack natif :

- `app/router.tsx` reste la source de vérité
- `layouts` sont branchés sur les routes protégées
- `domains` est orthogonal au routing

2️⃣ Mapping Cahier des charges → Frontend

Authentification & Sécurité

Besoin	Module	État
Login / Logout	domains/auth	✅ présent
Reset password	domains/auth	✅ présent
JWT sécurisé	services/api	⚠ à renforcer
Multi-tenant	tenant.provider	⚠ partiel

Dashboard Admin

Fonction	Fichier	Action
Vue globale	DashboardAdminPage.tsx	refactor
Gestion rôles	rolePermission.tsx	refactor
Stats admin	ui/adminComponents	conserver

Dashboard Management

Fonction	Module	État
KPI projets	domains/dashboard/manager	✅
Gantt	domains/planning	⚠ isoler
Budget	charts	✅

Projets

Fonction	Page	Action
Liste projets	AllProjectPage.tsx	refactor
Détails	ProjectDetailsPage.tsx	refactor

Scrum / Suivi

Fonction	Module	État
Board	domains/scrum	✅ solide
Modals	scrum/BoardModals	⚠ centraliser

3️⃣ Plan de refactorisation fichier par fichier

✅ À CONSERVER (structure saine)

- `hooks/auth/*`

- `schemas/auth/*`
- `store/useAuthStore.ts`
- `components/shared/scrum/*`
- `ui/*` (composants visuels)

À DÉPLACER

Fichier	De	Vers
useDashboard.ts	hooks	domains/dashboard
useGantt.ts	hooks	domains/planning
tasksService.ts	services	domains/projects/api

À REFACTORISER

Fichier	Problème	Action
DashboardAdminPage.tsx	logique mêlée	séparer hooks
rolePermission.tsx	permissions hardcodées	brancher API
Header	statique	rendu par rôle

À SUPPRIMER

Élément	Raison
mocks/*	incompatibles prod
mockProject.ts	doublon API

Règles finales (non négociables)

1. Aucun appel API dans les composants
2. Aucun JSX dans `shared/ui`
3. Permissions pilotent le rendu
4. Token jamais manipulé hors auth provider
5. Architecture = contrat

Conclusion

Ce document fige **la vision technique unique** de ProEvalis Frontend.
Il permet :

- travail fluide avec Codex,
- intégration backend sans friction,
- évolutivité long terme.

☞ Toute évolution future devra respecter ce cadre.