app nextjs version 15 +

better auth

zustand

zod

prisma

…

Liste des fichiers de app-builder

# Les routes

## auth

### app/api/auth/[...all]/route.ts (DONE)

#### Description :

Route catch-all pour l'authentification

Fonctionnalités :

Login, logout, register, reset password, verify email

Utilise : lib/auth/auth-server.ts, lib/prisma.ts

#### Utilisé par :

Toutes les pages d'authentification

#### Export :

{ GET, POST}

## Project

### app/api/projects/route.ts (DONE)

Description :

CRUD des projets utilisateur

Fonctionnalités :

Liste tous les projets de l’utilisateur (GET)

Permet la création de projet (POST).

Utilise :

* lib/prisma.ts,
* 🔐 lib/auth/auth-server.ts,
* validators/projectSchemas.ts

Utilisé par :

app/projects/page.tsx,

components/models/ProjectList.tsx,

Export :

{ GET, POST }

### app/api/projects/[projectId]/route.ts (DONE)

Description :

Gestion d'un projet spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, lib/auth/auth-server.ts, validators/projectSchemas.ts

Utilisé par :

app/projects/[projectId]/page.tsx, components/models/ProjectCard.tsx, stores/useProjectStore.ts

Export :

{ GET, PUT, DELETE }

## roadmap

### app/api/roadmap/route.ts

#### Description :

Gestion des timelines projet avec phases et dépendances

#### Fonctionnalités :

GET (par projet avec phases), POST (création)

Utilise :

* lib/prisma.ts, validators/roadmapSchemas.ts,
* lib/auth/auth-server.ts

#### Utilisé par :

* app/roadmap/page.tsx,
* components/models/RoadmapList.tsx,
* stores/useRoadmapStore.ts

Export : { GET, POST }

### app/api/roadmap/[roadmapId]/route.ts

#### Description :

Gestion d'une roadmap spécifique

#### Fonctionnalités :

* GET (détails), PUT (mise à jour), DELETE (suppression)

#### Utilise :

* lib/prisma.ts, validators/roadmapSchemas.ts, lib/auth/auth-server.ts

#### Utilisé par :

* app/roadmap/[roadmapId]/page.tsx, components/models/RoadmapForm.tsx

#### Export :

{ GET, PUT, DELETE }

## features

### app/api/features/route.ts

#### Description :

Gère la liste hiérarchique des features d’un projet (GET : par projectId), et la création de nouvelles features (POST).

#### Fonctionnalités :

* GET (hiérarchie par projet),
* POST (création)

#### Utilise :

* lib/prisma.ts, validators/featureSchemas.ts,
* lib/auth/auth-server.ts

#### Utilisé par :

* app/features/page.tsx,
* components/models/FeaturesList.tsx,

#### Export :

{ GET, POST }

### app/api/features/ [featuresId]/route.ts

Description :

Gestion d'une feature spécifique

Fonctionnalités :

Gère une feature spécifique (GET : détail, PUT : modification, DELETE : suppression). Gère aussi les relations parent/enfant.

Utilise :

lib/prisma.ts, validators/featureSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/features/[featureId]/page.tsx, components/models/FeaturesForm.tsx

Export :

{ GET, PUT, DELETE }

## tasks

### app/api/tasks/route.ts

Description :

CRUD des tâches

Fonctionnalités :

GET (avec filtres), POST (création avec assignation)

Utilise :

lib/prisma.ts, validators/taskSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/tasks/page.tsx, components/models/TaskList.tsx, stores/useTasksStore.ts

Export :

{ GET, POST }

### app/api/tasks/[taskId]/route.ts

Description :

Gestion d'une tâche spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/taskSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/tasks/[taskId]/page.tsx, components/models/TaskForm.tsx

Export :

{ GET, PUT, DELETE }

## files

### app/api/files/route.ts

#### Description :

Gestion des fichiers

#### Fonctionnalités :

Liste les fichiers d’un projet (GET : par projectId),

gère l’upload (POST). Valide le type de fichier et rattache au projet.

Utilise : lib/prisma.ts, lib/upload.ts, lib/auth/auth-server.ts

#### Utilisé par :

* app/files/page.tsx,
* components/models/FilesList.tsx,
* utils/uploadFile.tsx

#### Export :

{ GET, POST }

### app/api/files/[fileId]/route.ts (DONE)

#### Description :

Gestion d'un fichier spécifique

#### Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

#### Utilise :

* lib/prisma.ts,
* lib/upload.ts,
* lib/auth/auth-server.ts

#### Utilisé par :

* app/files/[fileId]/page.tsx,
* components/models/files/FilesForm.tsx

#### Export :

{ GET, PUT, DELETE }

## User stories

### app/api/user-stories/route.ts

Description :

CRUD des user stories

Fonctionnalités :

GET (par projet avec hiérarchie), POST (création)

Utilise : lib/prisma.ts, validators/userStorySchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/user-stories/page.tsx, components/models/UserStoryList.tsx, stores/useUserStoriesStore.ts

Export :

{ GET, POST }

### app/api/user-stories/[storyId]/route.ts

Description :

Gestion d'une user story spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/userStorySchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/user-stories/[storyId]/page.tsx, components/models/UserStoryForm.tsx

Export :

{ GET, PUT, DELETE }

## sprints

### app/api/sprints/route.ts

Description :

Gestion d'un sprint spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/sprintSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/sprints/[sprintId]/page.tsx, components/models/SprintForm.tsx

Export :

{ GET, PUT, DELETE }

### app/api/sprints/[sprintId]/route.ts

Description :

Gestion d'un sprint spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/sprintSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/sprints/[sprintId]/page.tsx, components/models/SprintForm.tsx

Export :

{ GET, PUT, DELETE }

## shema

### app/api/schema-fields/route.ts

Description :

CRUD des schémas de champs

Fonctionnalités :

GET (par projet), POST (création)

Utilise :

lib/prisma.ts, validators/schemaSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/schema/page.tsx, components/models/SchemaList.tsx, stores/useSchemaStore.ts

Export :

{ GET, POST }

### app/api/schema-fields/[fieldId]/route.ts

Description :

Gestion d'un champ spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/schemaSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/schema/[fieldId]/page.tsx, components/models/SchemaForm.tsx

Export :

{ GET, PUT, DELETE }

## teams

### app/api/teams/route.ts

Description :

CRUD des équipes

Fonctionnalités :

GET (liste avec membres), POST (création)

Utilise :

lib/prisma.ts, validators/teamSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/teams/page.tsx, components/models/TeamList.tsx, stores/useTeamsStore.ts

Export :

{ GET, POST }

### app/api/teams/[teamId]/route.ts

Description :

Gestion d'une équipe spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/teamSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/teams/[teamId]/page.tsx, components/models/TeamForm.tsx

Export :

{ GET, PUT, DELETE }

## Time log

### app/api/time-logs/route.ts

Description :

CRUD des logs de temps

Fonctionnalités :

GET (statistiques), POST (enregistrement)

Utilise :

lib/prisma.ts, validators/timeLogSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/roadmap/timelog/page.tsx, components/timelog/TimeLogList.tsx, utils/getTimeLogs.tsx

Export :

{ GET, POST }

### app/api/time-logs/[logId]/route.ts

Description :

Gestion d'un log spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/timeLogSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

components/timelog/TimeLogForm.tsx

Export :

{ GET, PUT, DELETE }

## comments

### app/api/comments/route.ts

Description :

CRUD des commentaires

Fonctionnalités :

* GET (par entité), POST (création)

Utilise :

* lib/prisma.ts, validators/commentSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

* app/comments/page.tsx, components/models/CommentList.tsx, stores/useCommentsStore.ts

Export :

{ GET, POST }

### app/api/comments/[commentId]/route.ts

Description :

Gestion d'un commentaire spécifique

Fonctionnalités :

GET (détails), PUT (mise à jour), DELETE (suppression)

Utilise :

lib/prisma.ts, validators/commentSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/comments/[commentId]/page.tsx, components/models/CommentForm.tsx

Export :

{ GET, PUT, DELETE }

## user

### app/api/users/route.ts

Description :

CRUD des utilisateurs (admin)

Fonctionnalités :

GET (liste filtrée), POST (création admin)

Utilise :

lib/prisma.ts, validators/userSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/admin/users/page.tsx, components/admin/UserList.tsx

Export :

{ GET, POST }

### app/api/users/[userId]/route.ts

Description :

Gestion d'un utilisateur spécifique

Fonctionnalités :

GET (profil), PUT (mise à jour), DELETE (admin)

Utilise :

* lib/prisma.ts, validators/userSchemas.ts, lib/auth/auth-server.ts

Utilisé par :

app/admin/users/[userId]/page.tsx, app/user/profile/page.tsx

Export : {

GET, PUT, DELETE }

## Activités

### app/api/activity-logs/route.ts

Description :

Logs d'activité système

Fonctionnalités :

* GET (historique filtré), POST (création automatique)

Utilise :

lib/prisma.ts, lib/auth/auth-server.ts

Utilisé par :

* app/activitylogs/page.tsx,
* app/admin/activity-logs/page.tsx,
* utils/getActivityLogs.tsx

Export :

{ GET, POST }

# Les pages

## app

### app/layout.tsx

Description :

Layout racine avec providers

Fonctionnalités :

Metadata, fonts, providers Auth/Zustand/Theme, navigation

Utilise : components/layout/Header.tsx, components/layout/Footer.tsx, components/providers/Providers.tsx, lib/auth/auth-client.ts

Utilisé par : T

outes les pages

Export : default function RootLayout

### App/page.tsx

Description :

Dashboard principal/Landing page

Fonctionnalités :

Vue d'ensemble, statistiques, actions rapides

Utilise :

* components/dashboard/DashboardStats.tsx,
* stores/useProjectsStore.ts,
* utils/getProjects.tsx

Utilisé par :

Navigation principale

Export :

default function HomePage

### App/loading.tsx

Description :

Composant de chargement global

Fonctionnalités :

Skeleton, spinner, état de chargement

Utilise :

components/ui/skeleton.tsx

Utilisé par :

Next.js automatiquement

Export :

default function Loading

### not-found.tsx

Description :

Page 404 personnalisée

Fonctionnalités :

Message d'erreur, navigation retour

Utilise : components/ui/button.tsx

Utilisé par :

Next.js automatiquement

Export :

default function NotFound

## auth

### app/auth/forgot-password/page.tsx

#### Description :

#### Fonctionnalités :

#### Utilise :

*Utilisé par :*

Export :

### app/auth/forgot-password/page.tsx

#### Description :

Demande de reset password

#### Fonctionnalités :

Formulaire email, envoi lien reset

Utilise :

lib/auth/auth-client.ts, validators/authSchemas.ts

#### Utilisé par :

Page sign-in

Export :

default function ForgotPasswordPage

### app/auth/reset-password/page.tsx

Description : Reset password avec token

Fonctionnalités : Validation token, nouveau mot de passe

Utilise : lib/auth/auth-client.ts, validators/authSchemas.ts

Utilisé par : Email de reset

Export : default function ResetPasswordPage

### app/auth/goodbye/page.tsx

#### Description :

#### Fonctionnalités :

Nettoie les stores Zustand et supprime les cookies de session

#### Utilise :

#### Utilisé par :

#### Export :

### app/auth/sign-in/page.tsx

#### Description :

Page de connexion

#### Fonctionnalités :

Formulaire login, validation, redirection vers User/hello/page.tsx

Utilise : lib/auth/auth-client.ts, validators/authSchemas.ts, components/auth/SignInForm.tsx

#### Utilisé par :

Navigation, redirections

#### Export :

default function SignInPage

### app/auth/sign-up/page.tsx

Description : Page d'inscription

#### Fonctionnalités :

* Formulaire register, validation email

#### Utilise :

lib/auth/auth-client.ts, validators/authSchemas.ts, components/auth/SignUpForm.tsx

Utilisé par :

* Page sign-in,

#### Export :

default function SignUpPage

### app/auth/verify-email/page.tsx

Description :

Page de déconnexion/suppression

Fonctionnalités :

Message de confirmation, nettoyage session

Utilise :

lib/auth/auth-client.ts

Utilisé par :

Actions de déconnexion

Export :

default function GoodbyePage

## User

### User/hello/page.tsx

Description : Page d'accueil utilisateur connecté

Fonctionnalités : Informations de base, actions rapides

Utilise : lib/auth/auth-client.ts, stores/useAuthStore.ts

Utilisé par : Navigation utilisateur

Export : default function HelloPage

### User/profile/page.tsx

Description : Profil utilisateur détaillé

Fonctionnalités : Modification infos, paramètres, historique

Utilise : lib/auth/auth-client.ts, components/user/ProfileForm.tsx, app/api/users/[userId]/route.ts

Utilisé par : Menu utilisateur

Export : default function ProfilePageExport :

### User/dashboard

Description : Dashboard personnel utilisateur

Fonctionnalités : Mes tâches, mes projets, statistiques

Utilise : stores/useTasksStore.ts, stores/useProjectsStore.ts, components/dashboard/UserStats.tsx

Utilisé par : Navigation utilisateur

Export : default function UserDashboardPage

### App/user/notifications/page.tsx

## comments

### app/comments/page.tsx

Description : Vue globale des commentaires

Fonctionnalités : Commentaires récents, par entité

Utilise : components/models/CommentList.tsx, stores/useCommentsStore.ts, utils/getComments.tsx

Utilisé par : Navigation, notifications

Export : default function CommentsPage

### app/comments/[commentId]page.tsx

Description :

Détails d'un commentaire

Fonctionnalités :

Thread de discussion, réponses

#### Utilise :

components/models/CommentForm.tsx, stores/useCommentsStore.ts, app/api/comments/[commentId]/route.ts

Utilisé par :

Liste commentaires

Export :

default function CommentDetailPage

## Dev

### app/dev/architecture/page.tsx

Description : exemple

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### app/dev/link/page.tsx

#### Description : exemples

#### Fonctionnalités :

#### Utilise :

Utilisé par :

#### Export :

### app/dev/roadmap/pages.tsx

#### Description : exemple

#### Fonctionnalités :

#### Utilise :

Utilisé par :

#### Export :

### app/dev/schema/page.tsx

Description : exemple

Fonctionnalités :

Utilise :

Utilisé par :

Export :

## Projects

### app/projects/page.tsx (Donne)

Description :

Affiche la liste des projets avec filtres, vue grille/liste, et possibilité d'ajout.

#### Fonctionnalités :

#### 1. Chargement des données avec SWR :

#### Utilise useSWR pour récupérer les projets depuis /api/projects et met a jour le store useProjectListStore

#### Gestion automatique du cache, revalidation et états de chargement

#### Gestion des erreurs intégrée

#### 2. Filtrage en temps réel :

#### Recherche textuelle : Filtre par nom de projet (insensible à la casse)

#### Filtrage par statut : Filtre selon le statut du projet (TODO, IN\_PROGRESS, etc.)

#### Filtrage combiné : Les deux filtres fonctionnent ensemble

#### 3. Gestion des vues :

#### Vue grille : Affichage en cartes (par défaut) Vue liste : Affichage en tableau,

#### Boutons de basculement : Interface pour changer de vue passe en props le type de vue souhaité à // components/models/ProjectList.tsx

#### 4. Interface utilisateur :

#### Composant de filtres : ProjectFilters pour la recherche et filtres passe en props la liste à afficher

#### États de chargement : Affichage "Chargement..." pendant le fetch

#### Gestion d'erreurs : Affichage des erreurs en rouge

#### Utilise

* ProjectList.tsx,
* ProjectFilters.tsx,
* ProjectForm.tsx,
* useProjectsStore Zustand (useProjectsStore),
* use-Modal.ts

Utilisé par :

Navigation principale (HeaderNav), dashboard

Export :

default function ProjectsPage

### app/projects/[projectId]/page.tsx

#### Description :

Vue détaillée d'un projet

#### Fonctionnalités :

Infos projet, équipe, métriques, navigation sous-sections (features/tasks/etc)

Utilise :

* components/models/ProjectCard.tsx,
* utils/getProjects.tsx,
* stores/useProjectStore.ts

#### Utilisé par :

* Liste projets,
* navigation

Export :

default function ProjectDetailPage

## features

### app/ features/page.tsx

#### Description :

Affiche une liste a l’aide du composant feature List. L’affichage est mis a jour via le store.

#### Fonctionnalités :

* Vue arborescente des features
* Création, modification et suppression de features
* Drag & drop pour réorganisation
* Interaction avec Zustand pour mise à jour globale

#### Utilise :

* components/models/FeaturesList.tsx,
* stores/useFeaturesStore.ts,
* utils/getFeatures.tsx

#### Utilisé par :

Navigation projet

#### Export :

default function FeaturesPage

### app/features/[featuresId]/page.tsx

Description :

Détails d'une feature spécifique

Fonctionnalités :

Infos feature, sous-features, user stories liées

Utilise :

components/models/FeaturesForm.tsx, stores/useFeaturesStore.ts, app/api/features/[featureId]/route.ts

Utilisé par :

Liste features

Export : default function FeatureDetailPage

### app/user-stories/page.tsx

Description :

Liste des user stories

Fonctionnalités :

Affichage par epic, priorisation, estimation

Utilise :

components/models/UserStoryList.tsx, stores/useUserStoriesStore.ts, utils/getUserStories.tsx

Utilisé par :

Navigation projet

Export :

default function UserStoriesPage

### app/user-stories/[storyId]/page.tsx

Description :

Détails d'une user story

Fonctionnalités :

Critères d'acceptation, tâches, commentaires

Utilise :

components/models/UserStoryForm.tsx, stores/useUserStoriesStore.ts, app/api/user-stories/[storyId]/route.ts

Utilisé par :

Liste user stories

Export :

default function UserStoryDetailPage

### app/sprints/page.tsx

Description :

Gestion des sprints

Fonctionnalités :

Board Kanban, planification, métriques

Utilise :

components/sprints/SprintBoard.tsx, stores/useSprintStore.ts, utils/getSprints.tsx

Utilisé par :

Navigation projet

Export : default function SprintsPage

### App/sprints/[sprintId]/page.tsx

Description : Détails d'un sprint

Fonctionnalités :

Burndown, rétrospective, clôture sprint

Utilise :

components/models/SprintForm.tsx, stores/useSprintStore.ts, app/api/sprints/[sprintId]/route.ts

Utilisé par : Liste sprints

Export : default function SprintDetailPage

### app /tasks/page.tsx

Description :

Liste des tâches

Fonctionnalités :

Filtres avancés, vue Kanban/liste, assignations

Utilise :

components/models/TaskList.tsx, stores/useTasksStore.ts, utils/getTasks.tsx

Utilisé par :

Navigation projet, sprints

Export :

default function TasksPage

### app /tasks/[taskId]/page.tsx

#### Description :

Détails d'une tâche

#### Fonctionnalités :

Infos complètes, sous-tâches, temps passé

#### Utilise :

* components/models/TaskForm.tsx,
* stores/useTasksStore.ts,
* app/api/tasks/[taskId]/route.ts

#### Utilisé par :

* Liste tâches,
* board sprint

Export :

default function TaskDetailPage

### app/ schemas/page.tsx

#### Description :

Gestion des schémas de données

#### Fonctionnalités :

Création tables, champs, relations

Utilise :

* components/models/SchemaList.tsx,
* stores/useSchemaStore.ts,
* utils/getSchemaFields.tsx

#### Utilisé par :

Navigation projet

#### Export :

default function SchemaPage

### app/schema/[schemaId]/page.tsx

#### Description :

Détails d'un schéma

#### Fonctionnalités :

* Edition champs,

#### Utilise :

* components/models/SchemaForm.tsx,
* stores/useSchemaStore.ts, app/api/schema-fields/[fieldId]/route.ts

#### Utilisé par :

* Liste schémas

#### Export :

default function SchemaDetailPage

### app /files/page.tsx

Description :

Gestionnaire de fichiers

Fonctionnalités :

Upload, organisation, preview

Utilise :

* components/models/FilesList.tsx,
* stores/useFilesStore.ts, utils/getFiles.tsx

#### Utilisé par :

Navigation projet

Export : default function FilesPage

### app/ /files/[fileId]/page.tsx

Description : affiche le détail du fichier présent dans le store

Détails d'un fichier

Fonctionnalités :

Metadata, versions, partage

Utilise :

* components/models/FilesForm.tsx,
* stores/useFilesStore.ts,
* app/api/files/[fileId]/route.ts

Utilisé par : Liste fichiers

Export : default function FileDetailPage

### app/reports/activity/page.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

## teams

### app/teams/page.tsx

Description :

Gestion des équipes

Fonctionnalités :

Liste équipes, membres, permissions

Utilise :

* components/models/TeamList.tsx,
* stores/useTeamsStore.ts,
* utils/getTeams.tsx

#### Utilisé par :

Navigation, admin

Export :

default function TeamsPage

### app/teams/[teamId]/page.tsx

Description :

Détails d'une équipe

#### Fonctionnalités :

Gestion membres, projets assignés

#### Utilise :

* components/models/TeamForm.tsx,
* stores/useTeamsStore.ts,
* app/api/teams/[teamId]/route.ts

#### Utilisé par :

Liste équipes

#### Export :

default function TeamDetailPage

## Suivie d’activité

### app/roadmap/page.tsx

Description :

Roadmap du projet

Fonctionnalités :

Vue timeline, phases, dépendances

Utilise :

* components/roadmap/RoadmapView.tsx, stores/useRoadmapStore.ts, utils/getRoadmaps.tsx

#### Utilisé par :

Navigation projet

#### Export :

default function RoadmapPage

### app/roadmap/timelog/page.tsx

Description :

Suivi du temps par roadmap

Fonctionnalités :

Rapports temps, comparaison estimations

Utilise :

* components/timelog/TimeLogList.tsx,
* utils/getTimeLogs.tsx

Utilisé par :

Navigation roadmap

Export :

default function TimeLogPage

### app/roadmap/activitylogs/page.tsx

Description :

Historique des activités

Fonctionnalités :

Filtres par utilisateur, type, date

Utilise :

components/activity/ActivityList.tsx, utils/getActivityLogs.tsx

Utilisé par :

Navigation, monitoring

Export :

default function ActivityLogsPage

## Admin du projet

### app/admin/page.tsx

#### Description :

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

#### Export :

### app/admin/users/page.tsx

#### Description :

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

#### Export :

### app/admin/users/[userId]/page.tsx

#### Description :

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

#### Export :

### app/admin/teams/page.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### app/admin/projects/page.tsx

#### Description :

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

#### Export :

### app/admin/activity-logs/page.tsx

#### Description :

Page d'administration pour la consultation de tous les logs d'activité du système.

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

#### Export :

### app/admin/settings/page.tsx

#### Description :

Page où les administrateurs peuvent configurer les paramètres globaux de l'application.

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

app/admin/page.tsx via un lien de navigation.

#### Export :

default function AdminSettingsPage

# components

## layout

### components/layout/header/Header.tsx

#### Description :

En-tête principal

#### Fonctionnalités :

* Logo et navigation principale
* Menu utilisateur
* Notifications
* Recherche globale

Utilise :

* components/layout/header/Logo.tsx,
* components/layout/header/HeaderNav.tsx,
* compoents/layout/header/IsConnected

#### Utilisé par :

app/layout.tsx

#### Export :

default function Header

### components/layout/header/Logo.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### components/layout/header/HeaderNav.tsx

#### Description :

Contient les liens de navigation principaux de l'application

#### Fonctionnalités :

Affiche les liens vers les sections clés (ex: Projets, Tâches, Roadmap). Peut mettre en évidence le lien actif en fonction de la route actuelle. S'adapte potentiellement pour les écrans mobiles (menu burger).

#### Utilise :

#### Utilisé par :

components/layout/header/Header.tsx.

#### Export :

default function HeaderNav

### components/layout/header/Isconected.tsx

#### Description :

Gère l'affichage des options utilisateur (connecté) ou des appels à l'action (déconnecté) dans l'en-tête.

#### Fonctionnalités :

Si l'utilisateur est connecté: affiche son nom/avatar et un menu déroulant avec des liens vers Profil, Paramètres, Déconnexion. Si déconnecté: affiche les boutons/liens Connexion et Inscription. Interagit avec useAuthStore pour connaître l'état d'authentification.

#### Utilise :

#### Utilisé par :

components/layout/header/Header.tsx.

#### Export :

default function IsConnected

### components/layout/footer.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### components/layout/Background.tsx

#### Description :

Composant optionnel pour gérer un arrière-plan global ou spécifique à certaines sections de l'application.

#### Fonctionnalités :

#### Utilise :

#### Utilisé par :

app/layout.tsx

#### Export :

default function Background

## models

### components/models/Projects/ProjectList.tsx (DONE)

Description : Liste des projets avec filtres et sélection d’un projet.

#### Fonctionnalités :

* Affichage de la liste des projets :
* Vue grille : Cartes avec image, nom, description, statut et date de création (doit uitiliser projectsCard)
* Vue liste : Tableau avec colonnes (Nom, Statut, Priorité, Date, Actions) (doit utiliser projectTab.tsx a creer)
* Création : Bouton "Ajouter un projet" qui ouvre une modale avec ProjectForm
* Modification : Bouton crayon (icône Pencil) qui ouvre la modale en mode édition et Suppression : Bouton corbeille (icône Trash2) avec confirmation avant suppression (responsabilité des composant enfants)
* Navigation : Boutons précédent/suivant avec affichage "Page X / Y" (doit utiliser un composant)
* Contrôle : Désactivation des boutons aux limites (première/dernière page)
* Modale responsive : Pour création/modification avec fond semi-transparent
* États de chargement : Désactivation des boutons pendant les opérations
* Labels traduits : Statuts en français via statusLabel

#### Utilise :

* ProjectCard.tsx,
* projectTab.tsx
* ProjectFilters.tsx,
* stores/useProjectStore.ts,
* hooks/use-Modal.ts

#### Utilisé par :

app/projects/page.tsx

#### Export :

default function ProjectList

### components/models/projects/ProjectFilters.tsx (DONE)

#### Description :

Filtres dynamiques pour la liste des projets.

#### Fonctionnalités :

* Filtres dynamiques pour la liste des projets (statut, équipe, recherche).

#### Utilise :

useProjectListStore.ts

#### utilisé par :

* ProjectList.tsx,
* app/projects/page.tsx

#### Export :

### components/models/projects/ProjectCard.tsx

#### Description :

Affiche une carte résumant les informations clés d'un projet individuel.

#### Fonctionnalités :

* Affichage titre, description, statut
* Boutons d'action (voir, modifier, archiver)
* Design adaptatif (mobile/désktop)

#### Utilise :

shadcn/ui, tailwind.

#### Utilisé par :

components/models/Projects/ProjectList.tsx

Export :

default function ProjectCard

### components/models/projects/ProjectForm.tsx (DONE)

Description :

Formulaire projet

#### Fonctionnalités :

Création/édition

Utilise :

* useProjectStore
* lib/validators/projectSchemas,
* app/api/Project

#### Utilisé par :

* app/projects/page.tsx,
* ProjectList.tsx

Export :

### components/models/features/FeaturesList.tsx

#### Description :

Liste hiérarchique des features

#### Fonctionnalités :

* Affichage hiérarchique/arborescent des features (indentation, expand/collapse).
* Drag & drop pour réorganisation.
* Actions contextuelles (ajout, édition, suppression).

Utilise :

* components/models/FeaturesForm.tsx,
* stores/useFeatureStore.ts,
* schadcn accordion
* hooks/use-Modal.ts

Utilisé par :

Export : default function FeaturesList

### components/models/features/FeaturesForm.tsx

#### Description :

Formulaire pour la création ou l'édition d'une feature.

#### Fonctionnalités :

* Formulaire de création/édition de feature.
* Gestion parent/enfant et validation Zod.

#### Utilise :

* validators/featureSchemas.ts
* app/features/[featureId]/page.tsx

Utilisé par :

* components/models/features/FeaturesList.tsx
* app/features/[featureId]/page.tsx

Export :

default function FeaturesForm

### components/models/tasks/TaskList.tsx

#### Description :

Affiche une liste de tâches, potentiellement avec des options de vue (liste, Kanban).

#### Fonctionnalités :

* Liste des tâches (vue Kanban/liste).
* Filtres avancés,
* actions rapides, gestion de l’ordre affiche les icones trach edit.

#### Utilise :

* components/models/tasks/TaskCard.tsx
* components/tasks/TaskForm.tsx,
* stores/useTasksStore.ts
* react-beautiful-dnd

#### Utilisé par :

* app/tasks/page.tsx,
* components/sprints/SprintBoard.tsx

#### Export :

default function TaskList

### components/models/task/TaskForm.tsx

#### Description :

Formulaire pour la création ou l'édition d'une tâche.

#### Fonctionnalités :

* Formulaire création/édition de tâche.
* Assignation,
* gestion sous-tâches,
* validation Zod.

#### Utilise :

* stores/useTasksStore.ts,
* validators/taskSchemas.ts

#### Utilisé par :

* components/models/tasks/TaskList.tsx,
* app/tasks/[taskId]/page.tsx

#### Export :

default function TaskForm

### components/models/files/FilesList.tsx

#### Description :

Affiche la liste des fichiers associés à une entité (ex: projet, tâche).

#### Fonctionnalités :

* Liste des fichiers d’un projet,
* filtrage,
* actions (téléchargement, suppression, preview).

#### Utilise :

* FilesForm.tsx,
* stores/useFilesStore.ts

#### Utilisé par :

app/files/page.tsx et potentiellement dans les pages de détail d'autres entités (projet, tâche) pour montrer les fichiers attachés.

#### Export : default function FilesList

### components/models/files/files/FilesForm.tsx

#### Description :

formulaire pour la cretion d’un fichier et l'édition des métadonnées.

#### Fonctionnalités :

* Formulaire édition de fichier,

#### Utilise :

app/api/files/route.ts

#### Utilisé par :

* components/models/files/FilesList.tsx,
* app/files/[fileId]/page.tsx

#### Export :

default function FilesForm

## User Stories

### components/models/UserStoryList.tsx

#### Description :

Liste des user stories

#### Fonctionnalités :

* Affichage par epic/feature
* Prioritization drag & drop
* Estimation en story points
* Statuts visuels

#### Utilise :

* components/models/UserStoryForm.tsx,
* stores/useUserStoryStore.ts

#### Utilisé par :

* app/user-stories/page.tsx

#### Export :

default function UserStoryList

### components/models/UserStoryForm.tsx

#### Description :

Formulaire pour la création ou l'édition d'une user story.

#### Fonctionnalités :

* Formulaire création/édition de user story,
* Gestion hiérarchie,
* Validation Zod.

#### Utilise :

* validators/userStorySchemas.ts
* stores/useUserStoriesStore.ts

#### Utilisé par :

* components/models/UserStoryList.tsx
* app/user-stories/[storyId]/page.tsx

#### Export :

default function UserStoryForm

## Sprint

### components/sprints/SprintBoard.tsx

#### Description :

Board Kanban pour sprints

#### Fonctionnalités :

* Colonnes par statut,
* Drag & drop des tâches,
* Colonnes par statut (TODO, IN\_PROGRESS, REVIEW, DONE)
* Compteurs par colonne

#### Utilise :

* react-beautiful-dnd
* components/sprints/SprintColumn,
* stores/useSprintStore
* stores/useTasksStore.ts

#### Utilisé par :

app/sprints/page.tsx

#### Export :

### components/models/SprintList.tsx

#### Description :

Affiche une liste des sprints d'un projet.

#### Fonctionnalités :

Présente les sprints avec leur nom, dates de début et de fin, statut (planifié, actif, terminé). Permet de naviguer vers le SprintBoard ou la page de détail du sprint. Offre une option pour créer un nouveau sprint (ouvrant SprintForm.tsx)

#### Utilise :

* components/models/SprintForm.tsx
* stores/useSprintStore.ts

#### Utilisé par :

* app/sprints/page.tsx

#### Export :

default function SprintList

### components/sprints/SprintColumn.tsx

#### Description :

Colonne du board

Fonctionnalités :

* Affichage des tâches d’un statut,
* drop zone pour drag & drop.

#### Utilise :

components/tasks/TaskCard,

hooks/useTaskMutations

#### Utilisé par :

* components/sprints/SprintBoard.tsx

#### Export :

### components/models/sprint/SprintForm.tsx

#### Description :

Formulaire pour la création ou l'édition d'un sprint.

#### Fonctionnalités :

* Formulaire création/édition de sprint, gestion dates/statut.

#### Utilise :

* validators/sprintSchemas.ts
* stores/useSprintStore.ts

#### Utilisé par :

* app/sprints/[sprintId]/page.tsx
* components/models/SprintList.tsx

#### Export :

### components/models/RoadmapList.tsx

#### Description :

Affiche une liste des roadmaps ou des éléments majeurs d'une roadmap.

#### Fonctionnalités :

* Liste les roadmaps avec leur titre, une description sommaire, les dates clés, et un indicateur de progression.
* Permet de filtrer ou trier les roadmaps.
* Chaque élément de la liste peut mener à une vue détaillée (RoadmapView.tsx ou app/roadmap/[roadmapId]/page.tsx) ou à un formulaire d'édition (RoadmapForm.tsx).

#### Utilise :

* stores/useSchemaStore.ts
* components/models/RoadmapForm.tsx

#### Utilisé par :

app/roadmap/page.tsx

#### Export :

default function RoadmapList

### components/models/RoadmapForm.tsx

#### Description :

#### Formulaire pour la création ou l'édition d'une roadmap et de ses phases/jalons.

#### Fonctionnalités :

Formulaire création/édition de roadmap, gestion phases/dépendances.

#### Utilise :

* validators/roadmapSchemas.ts
* stores/useRoadmapStore.ts

#### Utilisé par :

* RoadmapList.tsx, app/roadmap/[roadmapId]/page.tsx
* *components/models/RoadmapList.tsx*

#### Export : default function RoadmapForm

### components/roadmap/RoadmapView.tsx

#### Description :

#### Fonctionnalités :

* Vue timeline graphique des roadmaps, gestion du zoom, navigation temporelle.

#### Utilise :

* useRoadmapStore
* react-big-calendar

#### Utilisé par :

app/roadmap/page.tsx

#### Export :

## Commentaires

### components/models/comments/CommentList.tsx

#### Description :

#### Fonctionnalités :

* Liste des commentaires,
* threads, réponses,
* filtres par entité.
* Boutton add

#### Utilise :

* CommentForm.tsx, stores/useCommentsStore.ts

#### Utilisé par :

* app/comments/page.tsx

#### Export :

### components/models/comments/CommentForm.tsx

#### Description :

Formulaire pour la création ou l'édition d'un commentaire.

#### Fonctionnalités :

* dois recuperrer le « theme » tâche, feature, etc.
* Formulaire création/édition de commentaire (textarea) (on utilisera un éditeur de texte enrichi ulterieurement)
* gestion parent/enfant,
* validation zod.

#### Utilise :

* validators/commentSchemas.ts,

#### Utilisé par :

* Components/comments/CommentList.tsx,
* app/comments/[commentId]/page.tsx

#### Export :

default function CommentForm

### components/timelog/TimeLogList.tsx

#### Description :

#### Fonctionnalités :

* Affichage des logs de temps par tâche/utilisateur,
* Statistiques, graphiques.

#### Utilise :

utils/getTimeLogs.tsx

#### Utilisé par :

app/roadmap/timelog/page.tsx

Export :

### components/timelog/TimeLogForm.tsx

#### Description :

Formulaire permettant aux utilisateurs d'enregistrer ou de modifier une entrée de temps.

#### Fonctionnalités :

Saisie/édition d’un log de temps, validation.

#### Utilise :

validators/timeLogSchemas.ts

#### Utilisé par :

* components/timelog/TimeLogList.tsx
* app/roadmap/timelog/page.tsx

#### Export :

default function TimeLogForm

## Actity

### components/activity/ActivityList.tsx

Description :

#### Fonctionnalités :

* Liste des logs d’activité, filtres par utilisateur/type/date.

#### Utilise :

utils/getActivityLogs.tsx

#### Utilisé par :

app/roadmap/activitylogs/page.tsx

#### Export :

## Team

### components/models/TeamList.tsx

#### Description : Affiche la liste des équipes

#### Fonctionnalités :

* Liste des équipes, membres, actions (ajout, suppression, gestion droits).
* Ouvre app/teams/[teamId]/page.tsx en cliquand sur une equipe

#### Utilise :

* TeamForm.tsx,
* stores/useTeamsStore.ts

#### Utilisé par :

app/teams/page.tsx, app/admin/teams/page.tsx

#### Export :

default function TeamList

### components/models/TeamForm.tsx

#### Description : Formulaire pour la création ou l'édition d'une équipe.

#### Fonctionnalités :

* Formulaire création/édition d’équipe, gestion membres,
* validation.

#### Utilise :

* validators/teamSchemas.ts
* stores/useTeamsStore.ts

#### Utilisé par :

* TeamList.tsx,
* app/teams/[teamId]/page.tsx

#### Export :

### components/dashboard/DashboardStats.tsx

#### Description : Affiche des cartes ou des widgets de statistiques clés sur le tableau de bord principal (app/page.tsx).

#### Fonctionnalités :

* Présente des chiffres agrégés : nombre total de projets, tâches en cours, tâches terminées récemment, progression moyenne des projets, nombre d'utilisateurs actifs, etc. Peut inclure des mini-graphiques ou des indicateurs visuels. Les données sont souvent récupérées via des stores Zustand qui eux-mêmes les ont obtenues des API..

#### Utilise :

* stores/useProjectsStore.ts,
* stores/useTasksStore.ts,
* stores/useUsersStore.ts

#### Utilisé par :

app/page.tsx (dashboard principal)

#### Export :

default function DashboardStats

### components/providers/Providers.tsx

#### Description :

Composant de haut niveau qui encapsule l'application avec tous les fournisseurs de contexte nécessaires.

#### Fonctionnalités :

#### Utilise :

* SessionProvider
* ThemeProvider

#### Utilisé par :

app/layout.tsx

#### Export : default function Providers

# lib

## auth

### lib/auth/auth.ts (DONE)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### lib/auth/auth-client.ts (DONE)

Description : Contient la logique d'authentification côté serveur, utilisée par les API Routes. Gère la vérification des sessions/tokens, l'accès à la base de données pour les utilisateurs, la création/validation de tokens.

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### lib/auth/auth-server.ts (DONE)

Description : Contient la logique d'authentification côté serveur, utilisée par les API Routes. Gère la vérification des sessions/tokens, l'accès à la base de données pour les utilisateurs, la création/validation de tokens.

Fonctionnalités :

Utilise :

Utilisé par :

Toutes les API

Export :

## generated

### lib/generated/prisma/client.ts (DONE)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

## autres

### lib/prisma.ts (DONE)

Description : nitialise et exporte une instance unique du client Prisma

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### lib/utils.ts

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### lib/constants.ts

Description :

constantes globales

Fonctionnalités :

Utilise :

Utilisé par :

Export :

# stores

### stores/useAuthStore.ts

#### Description :

#### Store Zustand pour gérer l'état global de l'authentification de l'utilisateur côté client.Fonctionnalités :

* Stocke l'état de l'utilisateur connecté (objet utilisateur, null si non connecté).
* Maintient les informations sur les permissions et rôles de l'utilisateur.
* Fournit des actions pour login (met à jour le store avec les données utilisateur après une connexion réussie via lib/auth/auth-client.ts), logout (nettoie les données utilisateur du store et potentiellement appelle une API de déconnexion), et setUser (pour initialiser ou mettre à jour l'état de l'utilisateur).

Gère la persistance de la session côté client si nécessaire (ex: synchronisation avec localStorage pour certains aspects non sensibles ou pour restaurer l'état au rechargement, bien que "Better Auth" devrait gérer la session principale).

Utilise :

* lib/auth/auth-client.ts
* zustand

Utilisé par :

* components/layout/header/IsConnected.tsx,
* app/layout.tsx

Export :

useAuthStore

### stores/useProjectStore.ts (DONE)

#### Description :

Store de gestion d’un projets et expose les information d’un seul projet. Expose en autre l’id du projet selectionné

#### Fonctionnalités :

* 📌 Stocke le projet actuellement sélectionné (`project`)
* 🧭 Stocke son identifiant (`projectId`) pour navigation ou lookup rapide
* 🔄 Permet de mettre à jour ou réinitialiser le projet actif
* 🧹 Peut être vidé via `clearProject` (ex. : changement d’espace de travail)

#### Utilise :

#### Utilisé par :

#### Export :

useProjectStore

### stores/useProjectListeStore.ts (DONE)

Description :

Store de gestion des projets. Expose la liste des projets

Fonctionnalités :

• 📌 Stocke la liste des projet

• 🧹 Peut être vidé via `clearProject` (ex. : changement d’espace de travail)

Utilise :

Utilisé par :

components/models/ProjectFilters.tsx

Export :

UseProjecListeStore

### stores/useFeaturesStore.ts ()

#### Description :

Store des features

#### Fonctionnalités :

* Features par projet avec hiérarchie
* Gestion des relations parent/enfant
* Calcul de progression
* Cache avec invalidation

#### Utilise :

utils/getFeatures.tsx, API route

#### Utilisé par :

Export :

useFeatureStore

### stores/useSprintStore.ts

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### stores/useTasksStore.ts

Description : Store des tâches

Fonctionnalités :

* Tâches par contexte (projet, sprint, feature)
* Gestion des assignations
* Suivi des statuts
* Métriques de performance

Utilise : utils/getTasks.tsx, API routes

Utilisé par :

Export : useTaskStore

# Hooks

### Hooks/use-toast.ts (DONE)

Description :

Hook pour notifications toast

Fonctionnalités :

* Gestion des messages de succès/erreur
* Auto-dismiss configurable
* Types de toast (info, warning, error)

Utilise :

components/ui/toast.tsx

Utilisé par :

Export : useToast

### Hooks/use-Modal.ts (DONE)

#### Description :

Hook pour gestion des modales

#### Fonctionnalités :

État d'ouverture/fermeture

Gestion du focus et accessibility

Fermeture par ESC/backdrop

Utilise : components/ui/dialog.tsx

#### Utilisé par :

#### Export :

useModal

# Utils

### utils/getRoadmaps.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### utils/getTeams.tsx

### utils/getUsers.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getProjects.tsx (byUser) (donne)

Description : utilise useSession de lib/auth/auth-client.ts

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getFeatures.tsx (byProjectId)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getUserStories.tsx (byProjectId)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getSprints.tsx (byProjectId)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getTasks.tsx (byProjectId)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getSchemafields.tsx (byProjectId)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getFiles.tsx (byProjectId)

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getComments.tsx

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/getTimeLogs.tsx (byTaskId, byUserId)

Description :

Fonction pour récupérer les suivis de temps par tâche ou utilisateur.

### Utils/getActivityLogs.tsx (byUserId)

Description : Fonction pour récupérer les journaux d’activité par utilisateur ou type.

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### Utils/uploadFile.tsx

Description : Fonction pour gérer le téléversement de fichiers

Fonctionnalités :

Utilise :

Utilisé par :

Export :

# Validators

### validators/projectSchemas.ts

Description : Schémas de validation Zod pour projets

Fonctionnalités :

Validation création/édition projet

Règles métier (dates, priorités)

Validation des relations équipes

Utilise : zod

Utilisé par :

Export : createProjectSchema, updateProjectSchema

### validators/authSchemas.ts

### validators/taskSchemas.ts

### validators/featureSchemas.ts

### validators/sprintSchemas.ts

# Autres

### middleware.ts

Description :

Fonctionnalités :

Utilise :

Utilisé par :

Export :

### next.config.js

Configuration Next.js

### tailwind.config.js

- Configuration Tailwind CSS

### tsconfig.json

- Configuration TypeScript

### package.json

- Dépendances et scripts

### .env

- Variables d'environnement

### prisma/schema.prisma

- Votre schéma Prisma

### components.json

- Configuration shadcn/ui

Ecris /api/projects/[projectId]/files

* Données des fichiers via SWR depuis /api/projects/[projectId]/files et pourquoi pas les récupérer ici ? puis que l’id du projet est connue ?