# Routes d'Authentification et Gestion du Profil

## Routes Publiques

* **POST /register** : Inscription d'un nouvel utilisateur avec validation des données
* **POST /login** : Connexion d'un utilisateur avec validation des données
* **POST /forgot-password** : Demande de réinitialisation de mot de passe
* **POST /reset-password** : Réinitialisation du mot de passe
* **GET /verify-email/:token** : Vérification de l'email avec un token

## Routes Protégées (nécessitent une authentification)

* **GET /me** : Récupération des informations de l'utilisateur connecté
* **PUT /profile** : Mise à jour du profil utilisateur
* **POST /change-password** : Modification du mot de passe
* **POST /enable-2fa** : Activation de l'authentification à deux facteurs
* **POST /verify-2fa** : Vérification de l'authentification à deux facteurs
* **POST /disable-2fa** : Désactivation de l'authentification à deux facteurs
* **POST /logout** : Déconnexion de l'utilisateur

# Routes de Gestion des Projets

* **POST /projects** : Création d'un nouveau projet (authentification requise)
* **GET /projects** : Récupération de tous les projets
* **GET /projects/:id** : Récupération d'un projet spécifique
* **PUT /projects/:id** : Mise à jour d'un projet (authentification requise)
* **DELETE /projects/:id** : Suppression d'un projet (authentification requise)

# Routes de Gestion des Contributions

* **POST /contributions** : Création d'une nouvelle contribution (authentification requise)
* **GET /contributions/project/:projectId** : Récupération des contributions pour un projet spécifique

# Routes de Gestion des Commentaires

* **POST /comments** : Création d'un nouveau commentaire (authentification requise)
* **GET /comments/project/:projectId** : Récupération des commentaires pour un projet spécifique

# Routes de Paiement Stripe

* **POST /create-checkout-session** : Création d'une session de paiement (authentification requise)
* **POST /webhook** : Gestion des webhooks Stripe pour le traitement des événements de paiement

## Notes importantes

* Les routes protégées nécessitent l'utilisation du middleware d'authentification (authMiddleware)
* Les routes d'inscription et de connexion utilisent une validation des données via

validateRequest

* L'architecture est modulaire avec une séparation claire des responsabilités par domaine
* Le système inclut une gestion complète de l'authentification avec support de la 2FA
* Integration avec Stripe pour la gestion des paiements

# Middlewares de Sécurité

## Authentification (auth.middleware.ts)

* Vérifie la présence et validité du token JWT dans les en-têtes
* Décode le token et vérifie l'existence de l'utilisateur
* Attache l'utilisateur à la requête pour utilisation ultérieure
* Gestion des erreurs d'authentification (401)

## Sécurité Générale (security.middleware.ts)

* Utilisation de Helmet pour la sécurisation des en-têtes HTTP
* Configuration des sessions Express avec options de sécurité
* Protection contre les injections NoSQL
* En-têtes de sécurité additionnels (X-Content-Type-Options, X-Frame-Options, X-XSS-Protection)

## Protection CSRF (csrf.middleware.ts)

* Protection contre les attaques CSRF
* Bypass pour les méthodes GET, HEAD, OPTIONS
* Exceptions pour les routes d'authentification (login/register)
* Configuration des cookies sécurisés

## Validation Captcha (captcha.middleware.ts)

* Vérification du token reCAPTCHA
* Intégration avec l'API Google reCAPTCHA
* Validation de la réponse du serveur Captcha

# Middlewares de Gestion des Accès

## Accès Premium (premium.middleware.ts)

* Vérifie si l'utilisateur a un abonnement premium actif
* Bloque l'accès aux fonctionnalités premium pour les non-abonnés
* Retourne une erreur 403 si l'accès est refusé

# Middlewares de Validation et Traitement

## Validation des Données (validation.middleware.ts)

* Utilisation de Zod pour la validation des schémas
* Schémas définis pour :
  + Inscription (username, email, password)
  + Connexion (email, password, code TOTP optionnel)
  + Formules (titre, LaTeX)
* Middleware générique de validation des requêtes

## Gestion des Fichiers (upload.middleware.ts)

* Configuration de Multer pour l'upload de fichiers
* Restrictions sur les types de fichiers (images uniquement)
* Limite de taille (5MB)
* Génération de noms de fichiers uniques

## Gestion des Erreurs (error.middleware.ts)

* Gestion centralisée des erreurs
* Traitement spécifique pour :
  + Erreurs CSRF
  + Erreurs de validation Mongoose
  + Erreurs API personnalisées
  + Erreurs serveur génériques
* Format de réponse standardisé pour les erreurs

# Points Clés de Sécurité

* Validation systématique des entrées utilisateur
* Protection contre les attaques courantes (CSRF, XSS, injections)
* Gestion sécurisée des sessions et cookies
* Système d'authentification robuste
* Traitement sécurisé des fichiers uploadés
* Gestion centralisée et standardisée des erreurs

# Schémas de Validation pour l'Authentification

## Validation de l'Inscription (Register)

#### Username :

* + Type : chaîne de caractères
  + Longueur : minimum 3 caractères, maximum 30 caractères

#### Email :

* + Format email valide requis

#### Mot de passe :

* + Minimum 8 caractères
  + Doit contenir au moins :
    - Une lettre minuscule
    - Une lettre majuscule
    - Un chiffre

## Validation de la Connexion (Login)

#### Email :

* + Format email valide requis

#### Mot de passe :

* + Champ obligatoire, sans contrainte de format

#### Code TOTP :

* + Champ optionnel pour l'authentification à deux facteurs

## Validation de la Mise à Jour du Profil

#### Username :

* + Optionnel
  + Si fourni : minimum 3 caractères, maximum 30 caractères

#### Biographie :

* + Optionnel
  + Si fournie : maximum 500 caractères

# Notes Techniques

* Utilisation de la bibliothèque Zod pour la validation des données
* Les schémas sont réutilisables dans différentes parties de l'application
* Structure claire et maintenable des règles de validation
* Validation synchrone et asynchrone possible

# Routes Frontend (Vue Router)

## Pages Publiques

* **Accueil** : /
* **Découverte** : /discover

#### Authentification :

* + Login : /login
  + Inscription : /register

#### Informations :

* + Comment ça marche : /how-it-works
  + À propos : /about
  + Centre d'aide : /help
  + Confiance et sécurité : /trust-safety
  + Contact : /contact

#### Projets :

* + Liste des projets : /projects
  + Détail d'un projet : /projects/:id

## Pages Protégées (authentification requise)

* **Profil** : /profile

#### Projets :

* + Création : /projects/new
  + Édition : /projects/:id/edit
  + Démarrer un projet : /start
* **Tableau de bord** : /dashboard

## Gestion des Erreurs

* Page 404 : Gestion des routes inexistantes

# Configuration de Déploiement

## Docker

### Phase de Build

* Utilisation de Node.js 20 Alpine
* Installation des dépendances
* Build de l'application

### Phase de Production

* Serveur Nginx Alpine
* Copie des fichiers buildés
* Exposition du port 80

## Configuration Nginx

### Règles Générales

* Écoute sur le port 80
* Configuration pour Single Page Application (SPA)
* Gestion du routing côté client

### Optimisations

#### Cache :

* + Assets statiques : 1 an
  + Images, CSS, et autres médias : 1 an
  + Headers de cache appropriés

### Proxy API

* Redirection des requêtes /api/ vers le backend
* Configuration des headers pour le proxy

### Sécurité

* Headers appropriés pour les types de contenu
* Configuration worker optimisée
* Gestion des connexions