Spécifications pour la Base de Données

1. Introduction

1.1 Objectif

Le but de cette base de données est de soutenir la plateforme en ligne visant à organiser et à consulter efficacement les projets de fin d'études des étudiants de l'Université Espoir de Calvary Chapel. Elle se basera sur leur discipline académique et leur thème respectif.

1.2 Portée

La base de données doit permettre aux utilisateurs (étudiants, enseignants, personnel administratif) de rechercher, accéder et se familiariser facilement avec les projets de fin d'études, en offrant une classification par discipline académique.

2. Spécifications Techniques

2.1 Type de Base de Données

La base de données sera implémentée en utilisant MongoDB en raison de sa stabilité et de sa compatibilité avec les besoins du projet.

2.2 Structure de la Base de Données

La base de données sera structurée pour prendre en charge les catégories et sous-catégories principales suivantes :

- Informatique
    - App mobile
    - Web application
    - Desktop application
- Comptabilité
    - Plan d'affaire
    - Système comptable
- Gestion
    - Plan d'affaire
    - Rédaction de projet
    - Mémoire
- Éducation
    - Rédaction projet
    - Mémoire

3. Modèle de Données

3.1 Tables Principales

- Users

  Cette table stocke les informations relatives aux utilisateurs autorisés de la plateforme.
  - user_id (GUID, Clé primaire)
  - username (VARCHAR, Nom d'utilisateur)
  - password (VARCHAR, Mot de passe hashé, à gérer avec des techniques de hachage sécurisé)
  - role (VARCHAR)
  - created_at (Date)
  - updated_at (Date)

- Projects

  Cette table stocke les détails des projets de fin d'études.
  - project_id (GUID, Clé primaire)
  - project_name (VARCHAR, Nom du projet)
  - description (TEXT, Description détaillée du projet)
  - cover (VARCHAR)
  - discipline (VARCHAR, Discipline académique du projet)
  - type (VARCHAR, Type de projet)
  - authors (VARCHAR, Auteurs du projet)
  - project_url (VARCHAR)
  - deleted (BOOLEAN, indiquant si le projet a été supprimé)
  - added_by (GUID, Clé étrangère référençant Users.user_id, indiquant l'utilisateur qui a ajouté le projet)
  - last_modified_by (GUID, Clé étrangère référençant Users.user_id, indiquant l'utilisateur qui a modifié le projet pour la dernière fois)
  - created_at (Date)
  - updated_at (Date)

4. Fonctionnalités

4.1 Recherche et Consultation

La base de données doit permettre la recherche et la consultation facile des projets en fonction de la discipline académique, du type, de l'auteur, etc.

4.2 Ajout et Modification

Les utilisateurs autorisés doivent être en mesure d'ajouter de nouveaux projets et de modifier les détails des projets existants.

4.3 Sécurité

Un système d'authentification et d'autorisation doit être mis en place pour garantir que seuls les utilisateurs autorisés peuvent ajouter, modifier ou supprimer des projets.

5. Contraintes

5.1 Contraintes Techniques

- La base de données doit être conforme aux normes de sécurité pour protéger les informations sensibles.
- La conception doit permettre une évolutivité future.

5.2 Contraintes de Délais

- Le projet doit respecter les contraintes budgétaires et temporelles définies.

Spécifications pour le Backend

2. Spécifications Techniques

2.1 Environnement de Développement

Le backend sera développé en utilisant Node.js avec Express comme framework. Les modules complémentaires incluent Knex pour l'accès à la base de données, JWT pour l'authentification, Bcryptjs pour le hachage des mots de passe, et Joi pour la validation des données.

2.2 API Endpoints

- /api/projects
    - GET: Récupérer la liste des projets basée sur les filtres (discipline, type, auteur, etc.).
    - POST: Ajouter un nouveau projet à la base de données.

- /api/projects/:projectId

    - GET: Récupérer les détails d'un projet spécifique.
    - PUT: Mettre à jour les détails d'un projet existant.
    - DELETE: Marquer un projet comme supprimé.

- /api/users
    - POST: Créer un nouvel utilisateur.
    - PUT: Mettre à jour les informations de l'utilisateur.
    - DELETE: Supprimer un utilisateur (pour les administrateurs uniquement).

- /api/auth
    - POST: Authentifier un utilisateur et générer un token JWT.

2.3 Sécurité

- L'API doit utiliser HTTPS pour toutes les communications.
- Implémentation de JWT pour l'authentification des utilisateurs avec une durée de vie du token limitée.
- Utilisation de Bcryptjs pour le hachage sécurisé des mots de passe stockés en base de données.
- Validation approfondie des données d'entrée avec Joi pour prévenir les attaques par injection.

3. Fonctionnalités

3.1 Authentification

- Seuls les utilisateurs authentifiés auront accès aux fonctionnalités de modification et d'ajout de projets.
- Les administrateurs peuvent créer, modifier, et supprimer des comptes utilisateur.

4. Contraintes

4.1 Contraintes Techniques

- Le backend doit être capable de gérer simultanément un nombre spécifié de requêtes pour garantir la performance.