SUPINFO

Projet SUPFile - Application de stockage cloud

Master Informatique - Data Engineer

Réalisé par:
Fadoua BELMOKHTAR

Année universitaire: 2025/2026

Table des matières

Table des figures & tableaux :

1. Introduction

1.1. Présentation du projet

SUPFile est une application web de stockage cloud permettant aux utilisateurs de stocker, organiser et partager leurs fichiers en ligne. L'application offre une interface moderne et intuitive similaire aux services comme Google Drive ou Dropbox.

1.2. Objectifs

- Fournir un espace de stockage sécurisé de 30 Go par utilisateur
- Permettre la gestion complète des fichiers et dossiers
- Offrir des fonctionnalités de partage par lien public
- Assurer une expérience utilisateur fluide et réactive

SUPINFO

2. Choix technologiques

2.1 Frontend

2.1.1. React

Framework moderne, performant et largement adopté. Permet de créer des interfaces réactives avec un système de composants réutilisables.

Figure 1: Logo React.

2.1.2. Vite

Outil de build rapide avec Hot Module Replacement (HMR) pour un développement efficace.

Figure 2: Logo Vite.

2.1.3. Tailwind CSS

Framework CSS utilitaire permettant un développement rapide et une personnalisation complète du design.

Figure 3: Logo Tailwind CSS.

2.1.4. React Router

Gestion du routage côté client pour une navigation fluide sans rechargement de page.

Figure 4: Logo React Router.

2.1.5. Axios

Client HTTP pour les appels API, avec gestion des intercepteurs pour l'authentification.

Figure 5: Logo Axios.

2.1.6. Lucide

Bibliothèque d'icônes moderne et légère.

Figure 6: Logo Lucide.

2.2. Backend

SUPINFO

2.2.1. Node.js

Figure 7: logo Node.js

Environnement JavaScript côté serveur, performant pour les opérations I/O.

2.2.2. Express.js

Figure 8: Logo Express.js

Framework web minimaliste et flexible pour créer des API REST.

2.2.3. Sequelize

Figure 9: Logo Sequelize.

ORM pour Node.js facilitant les interactions avec la base de données MySQL.

2.2.4. JWT

Figure 10: Logo JWT.

JSON Web Tokens pour l'authentification stateless et sécurisée.

2.2.5. Multer

Middleware pour la gestion des uploads de fichiers multipart/form-data.

2.2.6. Bcrypt.js

Figure 11: Logo Bcrypt.js

Hachage sécurisé des mots de passe.

2.2.7. Passport.js

Middleware d'authentification pour OAuth2 (Google, GitHub).

Figure 12: Logo Passport.js

2.2.8. Archiver

Création d'archives ZIP pour le téléchargement de dossiers.

2.3. Base de données

MySQL Base de données relationnelle robuste et performante. Idéale pour les données structurées avec relations (utilisateurs, fichiers, dossiers, partages).

Figure 13: Logo MySQL.

2.4. Conteneurisation

Docker Conteneurisation des services pour un déploiement uniforme et reproductible

Figure 14: Logo Docker.

Docker Compose Orchestration des conteneurs (frontend, backend, base de données) avec une seule commande.

Figure 15: Logo Docker Compose.

3. Architecture de l'application

3.1. Architecture 3-tiers

L'application respecte une architecture 3-tiers avec séparation claire des responsabilités :

SUPINFO

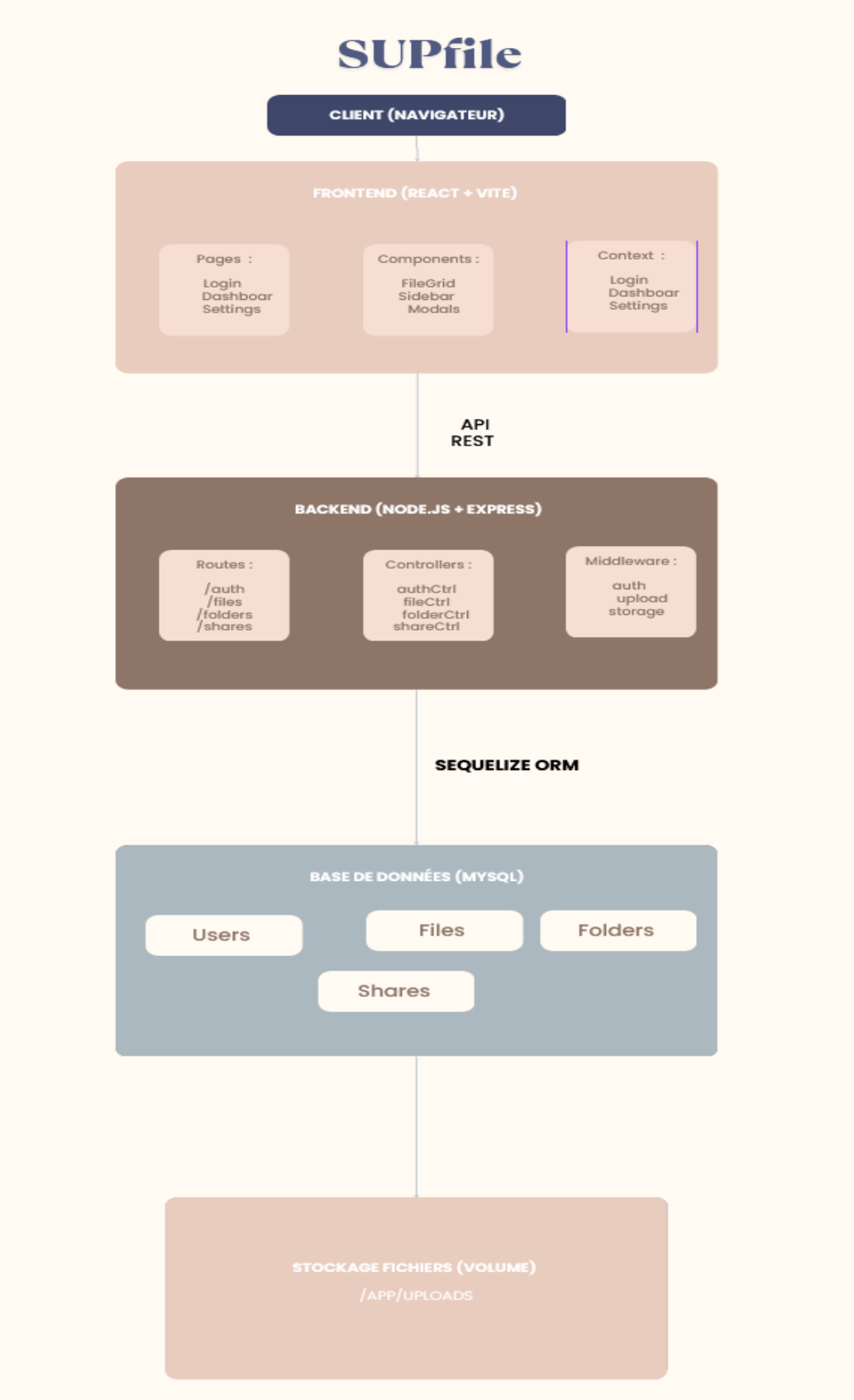

Figure 16: Architecture 3-tiers de l'application SUPFile.

SUPINFO

3.2. Structure des dossiers

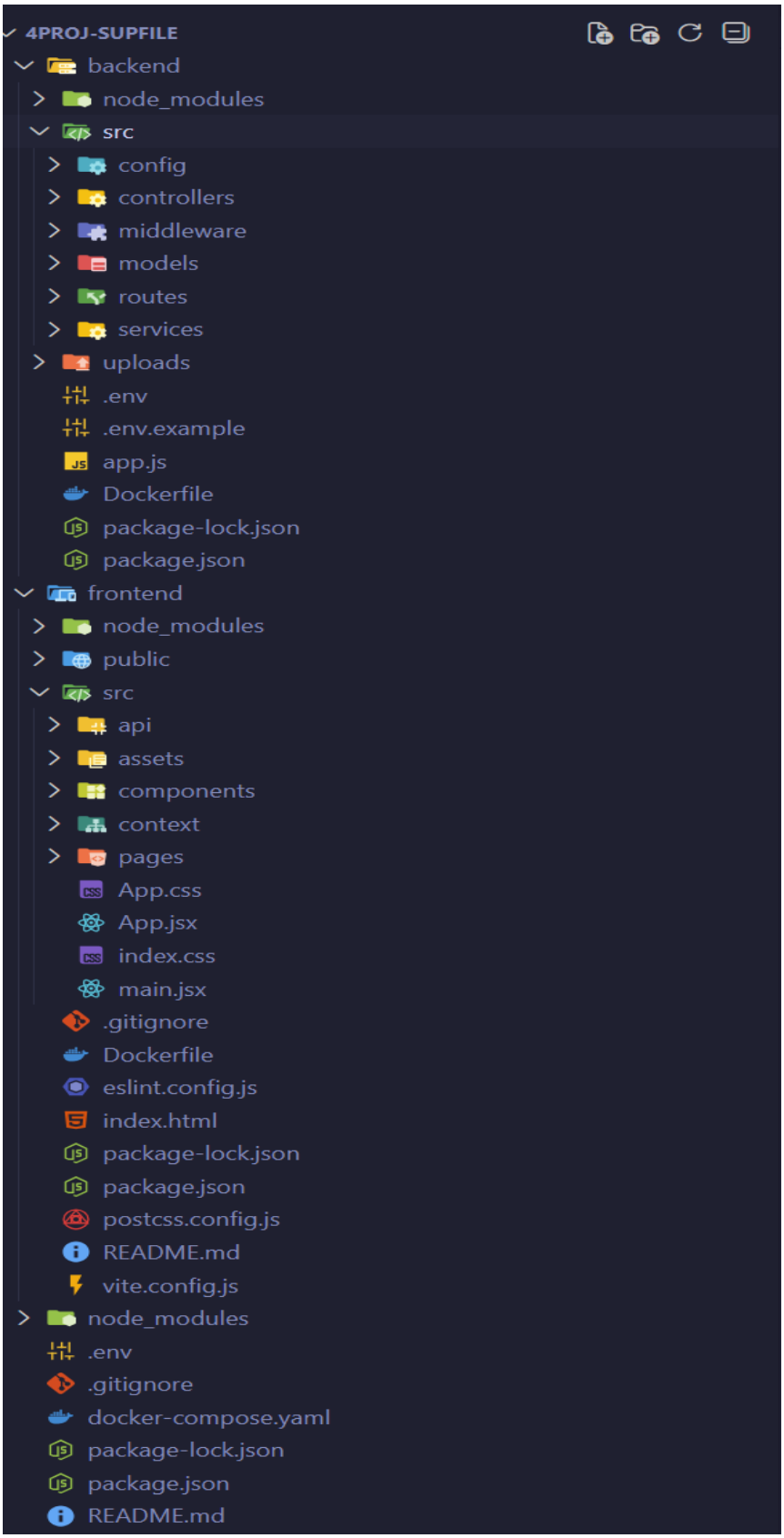

Figure 17: Structure des dossiers du projet SUPFile (frontend et backend).

4. Schéma de la base de données :

4.1. Vue d'ensemble du modèle relationnel

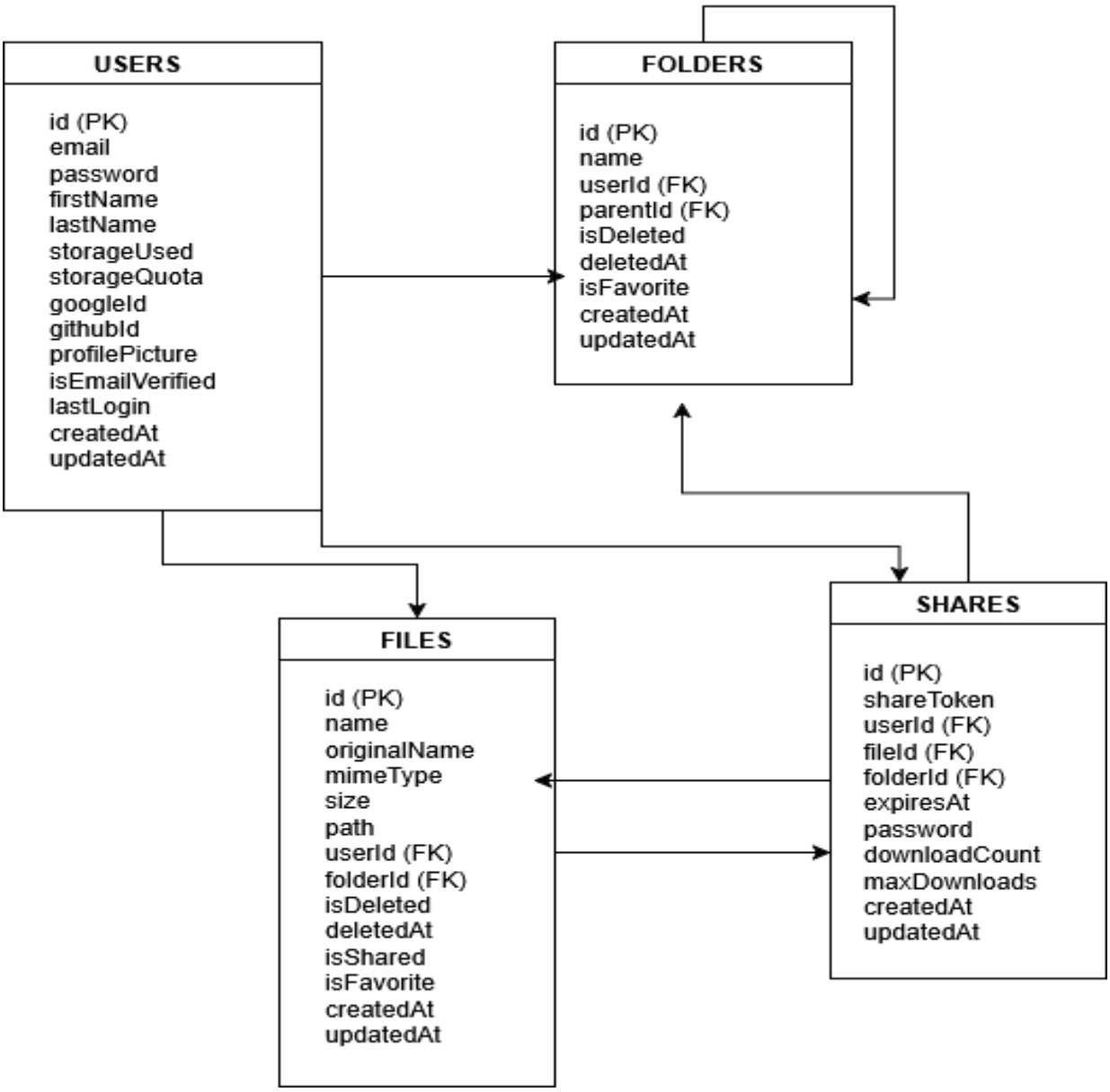

Figure 18: Schéma relationnel de la base de données SUPFile.

4.2. Description des tables :

4.2.1. Table Users

Colonne	Type	Description
id	INT (PK)	Identifiant unique auto-incrémenté

Colonne	Type	Description
email	VARCHAR (255)	Email unique de l'utilisateur
password	VARCHAR (255)	Mot de passe haché (bcrypt)
firstName	VARCHAR (255)	Prénom
lastName	VARCHAR (255)	Nom
storageUsed	BIGINT	Espace utilisé en octets
storageQuota	BIGINT	Quota de stockage (30 Go par défaut)
googleId	VARCHAR (255)	ID Google OAuth (nullable)
githubId	VARCHAR (255)	ID GitHub OAuth (nullable)
profilePicture	VARCHAR (255)	URL de la photo de profil
isEmailVerified	BOOLEAN	Email vérifié
lastLogin	DATETIME	Dernière connexion
createdAt	DATETIME	Date de création
updatedAt	DATETIME	Date de modification

Tableau 1: Description de la structure de la table Users.

4.2.2. Table Files

Colonne	**Type**	**Description**
id	INT (PK)	Identifiant unique auto-incrémenté
name	VARCHAR (255)	Nom du fichier stocké (UUID)
originalName	VARCHAR (255)	Nom original du fichier
mimeType	VARCHAR (255)	Type MIME du fichier
size	BIGINT	Taille en octets
path	VARCHAR (255)	Chemin physique du fichier
userId	INT (FK)	Propriétaire du fichier
folderId	INT (FK)	Dossier parent (nullable = racine)
isDeleted	BOOLEAN	Fichier dans la corbeille
deletedAt	DATETIME	Date de suppression
isFavorite	BOOLEAN	Marqué comme favori

Colonne	Type	Description
createdAt	DATETIME	Date de création
updatedAt	DATETIME	Date de modification

Tableau 2: Description de la structure de la table Files.

4.2.3. Table Folders

Colonne	Type	Description
id	INT (PK)	Identifiant unique auto-incrémenté
name	VARCHAR (255)	Nom du dossier
userId	INT (FK)	Propriétaire du dossier
parentId	INT (FK)	Dossier parent (nullable = racine)
isDeleted	BOOLEAN	Dossier dans la corbeille
deletedAt	DATETIME	Date de suppression
isFavorite	BOOLEAN	Marqué comme favori
createdAt	DATETIME	Date de création
updatedAt	DATETIME	Date de modification

Tableau 3: Description de la structure de la table Folders.

4.2.4. Table Shares

Colonne	Type	Description
id	INT (PK)	Identifiant unique auto-incrémenté
shareToken	VARCHAR (64)	Token unique pour l'URL de partage
userId	INT (FK)	Utilisateur ayant créé le partage
fileId	INT (FK)	Fichier partagé (nullable)
folderId	INT (FK)	Dossier partagé (nullable)
expiresAt	DATETIME	Date d'expiration (nullable)
password	VARCHAR (255)	Mot de passe haché (nullable)
downloadCount	INT	Nombre de téléchargements
maxDownloads	INT	Limite de téléchargements (nullable)
createdAt	DATETIME	Date de création
updatedAt	DATETIME	Date de modification

SUPINFO

Tableau 4: Description de la structure de la table Shares.

5. Description de l'API REST

5.1. Authentification

POST	/api/auth/register	Inscription d'un utilisateur
POST	/api/auth/login	Connexion
GET	/api/auth/me	Récupérer le profil connecté
PATCH	/api/auth/profile	Mettre à jour le profil
PATCH	/api/auth/change-password	Changer le mot de passe
GET	/api/auth/google	Connexion OAuth Google
GET	/api/auth/google/callback	Callback Google
GET	/api/auth/github	Connexion OAuth GitHub
GET	/api/auth/github/callback	Callback GitHub
GET	/api/auth/linked-accounts	Obtenir les comptes liés
DELETE	/api/auth/unlink-google	Délier compte Google
DELETE	/api/auth/unlink-github	Délier compte GitHub

Tableau 5: Endpoints de l'API REST pour l'authentification.

5.2. Fichiers

GET	/api/files	Lister les fichiers
POST	/api/files/upload	Upload de fichiers
GET	/api/files/trash	Lister la corbeille
DELETE	/api/files/trash/empty	Vider la corbeille
GET	/api/files/:id/download	Télécharger un fichier
GET	/api/files/:id/preview	Prévisualiser un fichier
GET	/api/files/:id/preview-info	Info de prévisualisation
PATCH	/api/files/:id/rename	Renommer un fichier
PATCH	/api/files/:id/move	Déplacer un fichier

PATCH	/api/files/:id/favorite	Ajouter/retirer des favoris
DELETE	/api/files/:id	Supprimer (corbeille)
PATCH	/api/files/:id/restore	Restaurer de la corbeille
DELETE	/api/files/:id/permanent	Supprimer définitivement

Tableau 6: Endpoints de l'API REST pour la gestion des fichiers.

5.3. Dossiers

GET	/api/folders	Lister les dossiers
POST	/api/folders	Créer un dossier
GET	/api/folders/:id/download	Télécharger en ZIP
PATCH	/api/folders/:id/rename	Renommer un dossier
PATCH	/api/folders/:id/move	Déplacer un dossier
PATCH	/api/folders/:id/favorite	Ajouter/retirer des favoris
DELETE	/api/folders/:id	Supprimer (corbeille)
PATCH	/api/folders/:id/restore	Restaurer de la corbeille
DELETE	/api/folders/:id/permanent	Supprimer définitivement

Tableau 7: Endpoints de l'API REST pour la gestion des dossiers.

5.4. Partage

POST	/api/shares	Créer un lien de partage
GET	/api/shares	Lister mes partages
DELETE	/api/shares/:id	Supprimer un partage
GET	/api/shares/public/:token	Accéder au partage public
POST	/api/shares/public/:token/verify	Vérifier mot de passe
GET	/api/shares/public/:token/download	Télécharger fichier partagé

Tableau 8: Endpoints de l'API REST pour la gestion des partages.

5.5. Autres

GET	/api/search	Rechercher fichiers/dossiers
GET	/api/recent	Fichiers récents

GET	/api/favorites	Fichiers favoris

Tableau 9: Endpoints de l'API REST pour les fonctionnalités complémentaires (recherche, récents et favoris).

6. Sécurité

6.1. Authentification

L'application utilise plusieurs mécanismes d'authentification pour sécuriser l'accès :

Mécanisme	**Description**
JWT (JSON Web Tokens)	Tokens signés avec une clé secrète, expiration de 7 jours par défaut
Stockage côté client	Token stocké dans localStorage, envoyé dans le header Authorization
OAuth2	Intégration Google et GitHub via Passport.js

Tableau 10: Mécanismes d'authentification utilisés dans l'application SUPFile.

6.2. Mots de passe

La sécurité des mots de passe est assurée par :

Aspect	**Implémentation**
Hachage	Bcrypt avec 10 rounds de salage
Validation	Minimum 8 caractères requis
Stockage	Jamais en clair, uniquement le hash en base de données

Tableau 11: Sécurisation des mots de passe utilisateurs.

6.3. Protection des routes API

Protection	**Description**
Middleware d'authentification	Vérifie le token JWT sur toutes les routes protégées
Vérification de propriété	Un utilisateur ne peut accéder qu'à ses propres fichiers
Validation des entrées	Vérification des paramètres avant traitement

Tableau 12: Protection des routes de l'API REST.

6.4. Partage sécurisé

Sécurité	**Implémentation**
Tokens uniques	Génération de tokens aléatoires de 32 caractères (crypto.randomBytes)
Mot de passe optionnel	Protection supplémentaire avec hachage bcrypt
Expiration	Date limite configurable pour les liens de partage

Tableau 13: Mécanismes de sécurisation des liens de partage.

6.5. Variables d'environnement

Les secrets sont stockés dans des fichiers .env qui ne sont **pas versionnés** dans Git (ajoutés dans .gitignore)

7. Guide de déploiement

7.1. Prérequis

- Docker (version 20.x ou supérieure)
- Docker Compose (version 2.x ou supérieure)
- Git

7.2. Installation

7.2.1. Cloner le projet

Figure 19: Clonage du dépôt Git du projet SUPFile.

7.2.2. Configurer les variables d'environnement

Créer le fichier backend/.env

7.2.3. Lancer l'application

Figure 20: Lancement de l'application avec Docker Compose.

7.3. Accéder à l'application

- Frontend : http://localhost:3000
- Backend API : http://localhost:5000

7.4. Persistance des données

- **Base de données** : Volume Docker db_data pour MySQL
- **Fichiers uploadés** : Volume Docker uploads_data pour les fichiers utilisateurs

Les données persistent après un redémarrage des conteneurs.

Conclusion

Cette documentation technique présente l'architecture et les choix technologiques du projet SUPFile. L'application respecte les bonnes pratiques de développement avec une séparation claire des responsabilités, une API REST bien structurée et une conteneurisation complète pour faciliter le déploiement.