SafeBase

Plateforme de sauvegarde automatisée

pour bases de données MySQL & PostgreSQL

Contexte et Problématique

Enjeux Critiques

- **Perte de données** : Erreur SQL irréversible (DROP DATABASE)
- **Sauvegardes manuelles** : Processus oublié ou non fiable
- **Organisation** : Fichiers dispersés, pas de versioning
- **Risque métier** : Coût financier et perte de confiance

Besoins Entreprise

- **Automatisation** : Sauvegardes régulières sans intervention
- **Centralisation** : Gestion unifiée de toutes les bases
- **Traçabilité** : Historique complet des versions

Vision

Plateforme centralisée pour automatiser la sauvegarde et la restauration de bases de données existantes

Valeurs

- **Simplicité** : Interface intuitive
- **Fiabilité** : Automatisation complète
- **Sécurité** : Chiffrement et isolation
- **Performance** : Architecture optimisée

Résultats

- **13 endpoints REST** opérationnels
- **100% des tests** passent

Composants

- **Frontend** : React 18 + Vite (port 5173)
- **Backend** : Fastify + TypeScript (port 8080)
- **Scheduler** : Conteneur Alpine + Cron
- **Bases** : MySQL 8 + PostgreSQL 16

Communication

- Frontend ↔️ Backend : **HTTP REST API**
- Scheduler → Backend : **POST /backup-all**
- Backend → Bases : **mysqldump / pg_dump**

Gestion des Bases

- `GET /databases` - Liste des bases
- `POST /databases` - Ajouter une base
- `GET /databases/available` - Bases disponibles

Sauvegardes

- `POST /backup/:id` - Backup unitaire
- `POST /backup-all` - Backup global
- `GET /backups/:id` - Liste des versions

Versions

- `POST /restore/:versionId` - Restaurer
- `POST /versions/:versionId/pin` - Épingler
- `GET /versions/:versionId/download` - Télécharger

- **Vite** : Build tool ultra-rapide
- **TypeScript** : Sécurité de types
- **Responsive** : Adaptatif mobile/desktop
- **Thème** : Clair/sombre

Fonctionnalités

- Formulaire d'ajout de base
- Liste interactive des bases
- Boutons Backup / Restore
- Modal de gestion des versions
- Statut API en temps réel
- Messages de succès/erreur

Flux Utilisateur

- **Fréquence** : Toutes les heures (`0 * * * *`)
- **Script** : `backup_all.sh`
- **Action** : Appel `POST /backup-all`
- **Isolation** : Conteneur Docker dédié
- **Fiabilité** : Redémarrage automatique

Monitoring

- **Heartbeat** : Mise à jour toutes les 5 min
- **Endpoint** : `/scheduler/heartbeat`
- **État** : Stocké dans `scheduler.json`
- **Alertes** : Webhook en cas d'échec
- **Logs** : Traçabilité complète

- **Configuration** : Variable d'environnement

Chiffrement

- **Algorithme** : AES-256-GCM
- **Champ** : Mots de passe bases
- **Clé** : Variable `ENCRYPTION_KEY`
- **Stockage** : Jamais en clair
- **Déchiffrement** : À la volée

Headers Sécurisés

- `X-Content-Type-Options: nosniff`
- `X-Frame-Options: DENY`

Entité : RegisteredDatabase

Description : Base de données MySQL ou PostgreSQL enregistrée dans SafeBase

RegisteredDatabase

id : UUID (Clé Primaire)

name : string (Nom de la connexion)

engine : enum (mysql | postgres)

host : string (Adresse serveur)

port : number (Port de connexion)

username : string (Nom d'utilisateur)

password : string (Chiffré AES-256-GCM)

database : string (Nom de la base)

createdAt : ISO 8601 (Date création)

Entité : BackupVersionMeta

Description : Version sauvegardée d'une base de données

Relation entre les Entités

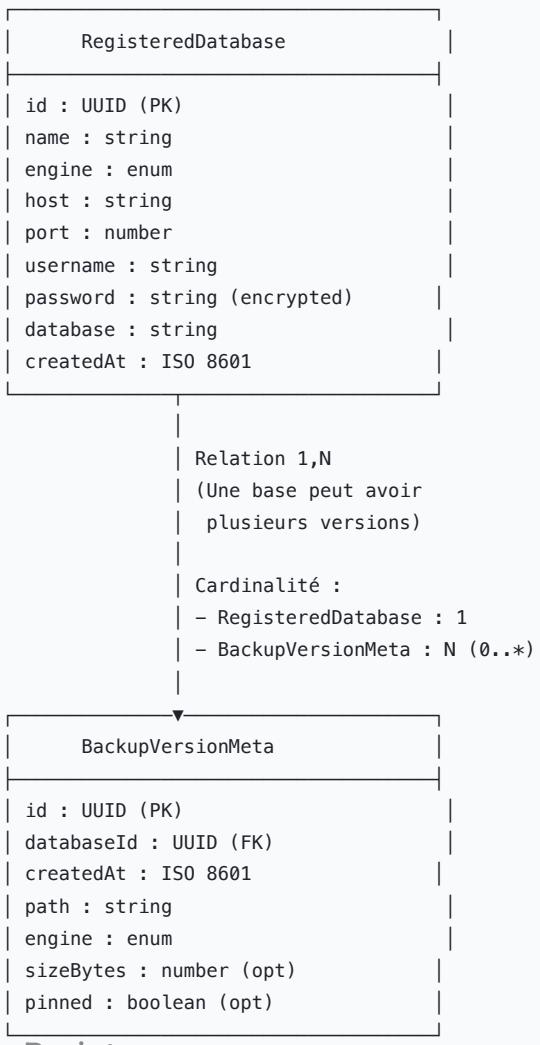

Table : RegisteredDatabase

Type : Table principale

Clé Primaire : `id`

Index Secondaires : `name`

```
interface RegisteredDatabase {
  id: string;                    // UUID, PK
  name: string;                  // Nom connexion
  engine: 'mysql' | 'postgres'; // Moteur
  host: string;                  // Serveur
  port: number;                  // Port
  username: string;              // Utilisateur
  password: string;              // Chiffré AES-256
  database: string;              // Nom base
  createdAt: string;             // ISO 8601
}
```

Contraintes :

- `id` : UNIQUE, NOT NULL

Table : BackupVersionMeta

Type : Table de relation

Clé Primaire : `id`

Clé Étrangère : `databaseId` → `RegisteredDatabase.id`

Index Secondaires : `databaseId` , `createdAt` , `pinned`

```
interface BackupVersionMeta {
  id: string;                    // UUID, PK
  databaseId: string;            // FK → RegisteredDatabase.id
  createdAt: string;             // ISO 8601
  path: string;                   // Chemin fichier SQL
  engine: 'mysql' | 'postgres';  // Moteur
  sizeBytes?: number;             // Taille (optionnel)
  pinned?: boolean;               // Épinglé (optionnel, défaut: false)
}
```

Contraintes :

- `id` : UNIQUE, NOT NULL
- `databaseId` : NOT NULL, FOREIGN KEY

- **Structure** : Tableau de `RegisteredDatabase`
- **Emplacement** : `/app/data/databases.json`
- **Format** : JSON array
- **Sécurité** : Mots de passe chiffrés (AES-256-GCM)

2. versions.json

- **Structure** : Tableau de `BackupVersionMeta`
- **Emplacement** : `/app/data/versions.json`
- **Format** : JSON array
- **Relation** : `databaseId` référence `RegisteredDatabase.id`

3. scheduler.json

- **Structure** : Objet avec `lastHeartbeat`
- **Emplacement** : `/app/data/scheduler.json`
- **Format** : JSON object

Structure des Répertoires

```
  {
    "id": "uuid-1",
    "name": "FitTracker Production",
    "engine": "mysql",
    "host": "127.0.0.1",
    "port": 8889,
    "username": "root",
    "password": "iv:salt:ciphertext:tag",
    "database": "fittracker",
    "createdAt": "2025-01-09T10:00:00.000Z"
  }
]
```

Exemple : versions.json

```
[
  {
    "id": "version-uuid-1",
    "databaseId": "uuid-1",
    "createdAt": "2025-01-09T12:00:00.000Z",
    "path": "/backups/uuid-1/FitTracker_2025-01-09T12-00-00.sql",
    "engine": "mysql",
    "sizeBytes": 1048576,
```

- **Format** : Fichiers `.sql` horodatés
- **Structure** : `backups/{db-id}/{name}_{timestamp}.sql`
- **Métadonnées** : `versions.json`
- **Volumes** : Persistance Docker

Fonctionnalités

- **Pin/Unpin** : Protéger versions importantes
- **Download** : Télécharger un backup
- **Delete** : Supprimer une version
- **Liste** : Historique chronologique
- **Tri** : Épinglées en premier

Politique de Rétention

Fastify 4.28

Zod

Vitest

Technologies Frontend

React 18.3

Vite

TypeScript

Infrastructure

Docker Compose

MySQL 8

Tests Backend

- **Framework** : Vitest
- **Coverage** : Santé API, sécurité, scheduler
- **Intégration** : Endpoints REST complets
- **Mocks** : Commandes système isolées

Tests Frontend

- **Framework** : Vitest + Testing Library
- **Composants** : Rendu et interactions
- **Scénarios** : Flux utilisateur complets

CI/CD

3. **mysql** : MySQL 8 (port 3306)
4. **postgres** : PostgreSQL 16 (port 5432)
5. **scheduler** : Alpine + Cron

Volumes

- `backups` : Stockage fichiers SQL
- `mysql_data` : Données MySQL persistantes
- `postgres_data` : Données PostgreSQL persistantes
- `data` : Métadonnées JSON

Démarrage

```
docker-compose up --build
```

- Vérifier statut API (vert)

2. Ajout d'une Base

- Formulaire : MySQL, host `mysql`, port `3306`
- Test de connexion automatique
- Enregistrement réussi

3. Backup Manuel

- Clic sur "Backup"
- Message de confirmation
- Vérification dans "Versions"

4. Restauration

- Ouvrir modal "Versions"

- **13 endpoints REST** opérationnels
- **Interface moderne** et intuitive
- **Automatisation complète** via cron
- **Sécurité** : API Key + chiffrement AES-256
- **Tests** : 100% de réussite
- **Documentation** : Complète et détaillée
- **Docker** : Déploiement simplifié

Évolutions Possibles

- **Base de données** : Migrer JSON → PostgreSQL
- **Authentification** : Système utilisateurs/roles
- **Compression** : Gzip des backups
- **Cloud** : Stockage S3/Azure Blob
- **Monitoring** : Dashboard avec métriques

Questions ?

Merci pour votre attention

SafeBase - Plateforme de sauvegarde automatisée