

Documentation : Biblio

Bienvenue sur Biblio, une application permettant la gestion de vos livres ! Elle se compose d'une API REST et d'un site web.

Architecture

API

L'API est organisée selon un pattern MC (Modèle-Contrôleur) avec DAO (Data Access Object). Elle utilise l'ORM Prisma(v7) pour interagir avec une base de données PostgreSQL, tout en restant adaptable à d'autres types de bases. Disponible dans /API. Les routes sont accessibles à l'aide de curl par exemple, car elles utilisent le CRUD, et ne sont pas accessible par le protocole http, pour le DELETE, UPDATE. Le GET est nativement disponible sur navigateur si vous le souhaitez. Le backend, fonctionne à l'aide du framework 'express.js' pour le routage et de prisma(v7) pour la bdd (utilisait anciennement http)

Organisation du code source :

1. Route : Contient toutes les routes de l'API.
2. Model : Vérifie l'intégrité des données à envoyer à la base de données.
3. DAO : Effectue les requêtes et vérifications des données.
4. Controller : Intermédiaire entre les routes et la base, gère la logique de vérification et d'envoi.
5. Test : contient des tests unitaires et fonctionnel. Vérifie le fonctionnement de l'API par url, du DAO et du model. Un problème est présent, dû à l'exploitation de la bdd, si vous souhaitez les vérifier, veuillez exécuter chaque fichier / fonction manuellement, sinon une erreur Prisma sera élevée. Les tests utilisent 'jest' et 'supertest' pour pouvoir fonctionner.

```
npm run test
```

Installation

1. Installez les dépendances : `npm install`.

2. Configurez la base de données dans un fichier `.env` avec : `DATABASE_URL`.

```
DATABASE_URL="postgresql://user:password@localhost:5432/mydb?schema=public"
```

En cas d'utilisation d'un autre type de base de données, autre que PostgreSQL, veuillez aller dans `Prisma/schema.prisma` et de modifier en valeur conforme à votre base.

```
provider = "postgresql"
```

3. reset bdd : `npx prisma migrate reset`

3. Appliquez les migrations avec : `npx prisma migrate dev --name init`.

4. Pour régénérer les fichiers nécessaires :

```
npx prisma generate
```

5. Démarrer l'API avec : `npm start`. Elle sera disponible sur <http://localhost:3000/api/v1>.

6. Une base de données de test a été réalisée avec `npm run bdd`

Routes Disponibles

Livres

- GET : /books - Récupère tous les livres
- POST : /books - Crée un livre
- GET : /books/:id - Récupère un livre spécifique
- PUT : /books/:id - Modifie un livre
- DELETE : /books/:id - Supprime un livre
- GET : /books/author/:id - Récupère les livres d'un auteur
- DELETE : /books/author/:id - Supprime les livres d'un auteur
- GET : /books/categories/:id - Récupère les livres d'une catégorie
- DELETE : /books/categories/:id - Supprime les livres d'une catégorie

Catégories

- GET : /categories - Récupère toutes les catégories
- GET : /categories/:id - Récupère une catégorie spécifique
- POST : /categories - Crée une catégorie
- PUT : /categories/:id - Modifie une catégorie
- DELETE : /categories/:id - Supprime une catégorie et ses livres associés
- GET : /categories/books/count - Récupère le nombre de livres par catégorie

Auteurs

- GET : /authors - Récupère tous les auteurs
- GET : /authors/:id - Récupère un auteur spécifique
- POST : /authors - Crée un auteur
- PUT : /authors/:id - Modifie un auteur
- DELETE : /authors/:id - Supprime un auteur et ses livres associés

Serveur Frontend

Le frontend est dissocié du backend sur un second serveur, disponible dans le dossier / Vue | Angular

Il utilise le Framework Vue.js, pourquoi ? Comme demander dans le Cahier des charges, nous avions le choix entre Angular et Vue.js. En raison de la taille du projet, du fait que je ne connaissais aucun des 2 Framework. Vue.js étant plus adapté aux petits projets comparés à Angular qui lui est plus prévue pour des plus importants, ce choix a donc été fait.

Mais, Angular étant plus Michelin friendly (composant michelin déjà accessible) et possédant une plus grande communauté et probabilité de maintenance dans le futur (#maintenue par Google), un second projet Angular a été créé, basé sur des libs réutilisable sur d'autre projet (plus scalable que vue).

[Vue]

Le projet est structuré autour de fichiers principaux comme Assets, Components, Config, Model, Router, et Service. Il utilise axios, pour les interactions avec l'API.

Installation

1. Installez les dépendances avec : `npm install`.
2. Configurez l'accès à l'API dans `src/config/config.js`.
3. Lancez le serveur avec : `npm run dev`.
2. Le serveur tourne sur <http://localhost:5173> par défaut.

Routes Frontend

- `/authors/:id/edit`
- `/authors, /authors/:id`
- `/authors/create`
- `/books, /books/create`
- `/books/:id`
- `/books/:id/edit`
- `/categories, /categories/:id`
- `/categories/create`
- `/categories/:id/edit`
- `/statistiques`

[Angular]

Installation

1. Installez les dépendances avec : `npm install`.
 2. Configurez l'accès à l'API dans `src/config/config.js`.
 3. Lancez le serveur avec : `npm run start`.
2. Le serveur tourne sur <http://localhost:4200> par défaut.

Routes Frontend

- `/authors/:id/edit`
- `/authors, /authors/:id`
- `/authors/create`
- `/books, /books/create`
- `/books/:id`
- `/books/:id/edit`
- `/categories, /categories/:id`
- `/categories/create`
- `/categories/:id/edit`
- `/statistiques`