**📚 Documentation créée**

**1. README.md - Documentation complète**

* Vue d'ensemble du projet
* Architecture technique détaillée
* Schéma GraphQL complet
* Guide d'installation et démarrage
* Fonctionnalités principales
* Sécurité et bonnes pratiques
* Base de données et relations
* Points forts du projet

**2. PRESENTATION.md - Guide de présentation**

* Structure de présentation (15-20 minutes)
* Points clés à souligner
* Ordre de démonstration recommandé
* Questions fréquentes et réponses
* Conseils pour la présentation
* Compétences à mettre en avant

**3. GRAPHQL\_EXAMPLES.md - Exemples de requêtes**

* Requêtes de base (books, authors)
* Authentification (login/register)
* Gestion des emprunts
* CRUD admin complet
* Requêtes complexes
* Guide d'utilisation du GraphQL Playground

**🎯 Points clés pour votre présentation**

**Introduction (2-3 min)**

* "Système complet de gestion de bibliothèque avec GraphQL"
* Architecture full-stack JavaScript moderne
* API GraphQL flexible vs REST traditionnel

**Démonstration (8-10 min)**

1. **Page d'accueil** → Statistiques admin
2. **Catalogue** → Recherche et filtres
3. **Authentification** → Inscription/connexion
4. **Emprunt** → Processus complet avec modal
5. **Gestion admin** → CRUD livres/auteurs
6. **GraphQL Playground** → Requêtes directes

**Questions fréquentes à préparer**

* **"Pourquoi GraphQL ?"** → Flexibilité, évite sur-fetching
* **"Sécurité ?"** → JWT + bcrypt + validation
* **"Architecture ?"** → Modulaire, maintenable
* **"Tests ?"** → Manuels complets + Playground
* **"Améliorations ?"** → Tests automatisés, notifications

**🚀 Conseils pour réussir**

**Avant la présentation**

1. Testez toutes les fonctionnalités
2. Préparez des comptes de test
3. Vérifiez que le serveur démarre
4. Ayez les documents ouverts

**Pendant la présentation**

1. Soyez confiant dans vos choix techniques
2. Expliquez pourquoi GraphQL plutôt que REST
3. Montrez la flexibilité de l'API
4. Démontrez la sécurité implémentée
5. Répondez aux questions avec assurance

**Points forts à souligner**

* ✅ **GraphQL moderne** vs REST traditionnel
* ✅ **Sécurité robuste** (JWT + bcrypt)
* ✅ **Interface intuitive** et responsive
* ✅ **Architecture modulaire** et maintenable
* ✅ **Fonctionnalités complètes** (CRUD, auth, emprunts)

**🚀 Services Web Principaux**

**1. GraphQL API (Service Web Principal)**

* **Technologie** : Apollo Server Express
* **Endpoint** : /graphql
* **Fonctionnalités** :
* Gestion des livres, auteurs, utilisateurs et emprunts
* Authentification et autorisation
* Recherche et filtrage
* Mutations pour CRUD complet

**2. Serveur Web Express.js**

* **Port** : 4000 (par défaut)
* **Fonctionnalités** :
* Servir l'interface web statique
* Middleware de sécurité (Helmet, CORS)
* Authentification JWT
* API REST complémentaire (/api/me)

**🔧 Services Web de Support**

**3. Base de Données SQLite**

* **Service** : Base de données locale
* **Fichier** : database/bibliotheque.db
* **Gestion** : Via sqlite3 Node.js

**4. Authentification JWT**

* **Service** : Gestion des tokens d'authentification
* **Bibliothèque** : jsonwebtoken
* **Fonctionnalités** : Login, register, session management

**5. Sécurité Web**

* **Helmet** : Protection des en-têtes HTTP
* **CORS** : Gestion des requêtes cross-origin
* **bcryptjs** : Hachage sécurisé des mots de passe

**�� Architecture des Services**

text

Apply to server.js

Frontend (HTML/CSS/JS)

    ↓

Express.js Server (Port 4000)

    ↓

├── GraphQL API (/graphql)

├── Static Files (/public)

├── REST API (/api/me)

└── Database (SQLite)

**🎯 Points d'Entrée des Services**

1. **Interface Web** : http://localhost:4000
2. **GraphQL Playground** : http://localhost:4000/graphql
3. **API Documentation** : http://localhost:4000/api
4. **API REST** : http://localhost:4000/api/me

Votre application utilise principalement **GraphQL** comme service web principal, avec Express.js comme serveur web et SQLite comme service de base de données. L'architecture est moderne et suit les bonnes pratiques de développement web.