📚 Projet Final - API Bibliothèque NodeJS/Express

Module : API NodeJS/Express – Master
Date limite de rendu : 29-12-2025

🎯 Objectifs pédagogiques

À la fin de ce projet, vous devez maîtriser :

- ✅ Créer un serveur Express avec middleware (Helmet, CORS, Morgan)
- ✅ Structurer une API en couches (controllers, models, routes, schemas)
- ✅ Implémenter des opérations CRUD complètes avec Firebase Firestore
- ✅ Valider les données avec Joi ou Zod
- ✅ Sécuriser les routes avec JWT (JSON Web Token)
- ✅ Hasher les mots de passe avec bcrypt
- ✅ Créer des vues EJS dynamiques
- ✅ Documenter et tester l'API avec Postman
- ✅ Déployer le code sur GitHub

📋 Spécifications fonctionnelles

Architecture générale

`text`

```
📁 projet-biblio/
├── 📁 controllers/        # Logique métier
├── 📁 models/             # Accès aux données (Firebase)
├── 📁 routes/             # Définition des endpoints
├── 📁 schemas/            # Validation Joi/Zod
├── 📁 views/              # Templates EJS
├── 📁 middlewares/        # Middlewares custom
├── server.js              # Point d'entrée
```

```
├── .env                    # Variables d'environnement (NE PAS
COMMIT)
├── .gitignore
├── package.json
├── README.md
└── firebase-config.json    # Config Firebase (NE PAS COMMIT)
```

Endpoints obligatoires

📚 LIVRES (`/api/books`)

Méthode	Route	Description	Auth
GET	`/api/books`	Lister tous les livres (pagination)	❌ Non
GET	`/api/books/:id`	Détails d'un livre spécifique	❌ Non
POST	`/api/books`	Créer un nouveau livre	✅ JWT
PUT	`/api/books/:id`	Modifier un livre	✅ JWT
DELETE	`/api/books/:id`	Supprimer un livre	✅ JWT

Schéma livre :

```
json

{
```

```
  "id": "UUID auto-généré",
  "title": "string (requis, max 200 caractères)",
  "author_id": "UUID (référence auteur)",
  "isbn": "string unique (13 caractères)",
  "published_year": "number (1900-2100)",
  "pages": "number",
  "description": "string",
  "available": "boolean (défaut: true)",
  "created_at": "timestamp",
  "updated_at": "timestamp"
}
```

👥 AUTEURS (`/api/authors`)

Méthode	Route	Description	Auth
GET	`/api/authors`	Lister tous les auteurs	❌ Non
GET	`/api/authors/:id`	Détails d'un auteur	❌ Non
POST	`/api/authors`	Créer un nouvel auteur	✅ JWT
PUT	`/api/authors/:id`	Modifier un auteur	✅ JWT
DELETE	`/api/authors/:id`	Supprimer un auteur	✅ JWT

Schéma auteur :

```
json
```

```
{
  "id": "UUID auto-généré",
  "name": "string (requis, 2-100 caractères)",
  "birth_year": "number (optionnel)",
  "nationality": "string (optionnel)",
  "biography": "string (optionnel)",
  "created_at": "timestamp",
  "updated_at": "timestamp"
}
```

🔐 AUTHENTIFICATION (`/api/auth`)

Méthode	Route	Description	Auth
POST	`/api/auth/register`	Créer un compte utilisateur	❌ Non
POST	`/api/auth/login`	Connexion (retourne JWT)	❌ Non
POST	`/api/auth/refresh`	Refresh (retourne JWT)	❌ Non

Schéma utilisateur :

```
json

{
  "id": "UUID auto-généré",
  "email": "string (unique, format email)",
  "password": "string hashé avec bcrypt",
```

```
  "created_at": "timestamp"
}
```

🛠️ Stack technique obligatoire

Dépendances principales

json

```
{
  "express":,
  "cors":,
  "helmet":,
  "morgan":,
  "ejs":,
  "firebase-admin":,
  "jsonwebtoken":,
  "bcrypt":,
  "joi":, // ou zod (Vérification de schéma)
  "dotenv":
}
```

Dépendances développement

json

```
{
  "nodemon"
}
```

📊 Grille de notation (100 points)

Critère	Points	Détails
Structure du projet	15	✅ Dossiers controllers/routes/schemas/models/views bien organisés
Serveur & Middleware	15	✅ Express configuré + Helmet/CORS/Morgan fonctionnels
CRUD Livres complet	20	✅ GET/POST/PUT/DELETE fonctionnels avec Firebase
CRUD Auteurs	15	✅ GET/POST/PUT/DELETE implémentés
Authentification JWT	15	✅ Register/login + middleware auth sur routes protégées
Validation & Sécurité	10	✅ Schemas Joi/Zod + bcrypt pour mots de passe
Vues EJS	5	✅ Page accueil + liste livres HTML dynamique
Documentation & Tests	5	✅ README + Collection Postman exportée

Bonus (+5 points max) :

- Pagination avec compteur total (+2)
- Recherche/Filtrage livres (+2)

📝 Livrables obligatoires

À rendre sur GitHub

1. Code source complet (tout le projet)
2. README.md contenant :
    - Description du projet
    - Instructions d'installation
    - Variables d'environnement à configurer
    - Exemples d'appels API (curl ou Postman)
    - Structure des dossiers
3. Collection Postman (fichier JSON)
    - Export de 15+ requêtes testées
    - Environnement avec variables {{base_url}}, {{token}}

À envoyer à votre formateur

- Lien GitHub du projet (public)
- Fichier Collection Postman si vous le souhaitez
- Toute question ou problème rencontré

Ressources utiles

- Express.js Official Docs
- Firebase Firestore Documentation
- JWT Introduction
- Joi Validation
- Bcrypt npm
- EJS Template Engine

⚠️ Points d'attention

Sécurité

- ❌ Ne commitez JAMAIS votre `.env` ou `firebase-config.json`
- ✅ Ajoutez-les à `.gitignore`

- ✅ Utilisez des variables d'environnement pour les secrets
- ✅ Validez TOUJOURS les inputs utilisateurs
- ✅ Hashez les mots de passe avec bcrypt (min 10 rounds)

Firebase

- Créez un projet Firebase gratuit sur
- firebase.google.com
- Générez une clé privée dans Project Settings → Service Accounts
- Activez Firestore Database (mode test ou règles sécurisées)
- Testez votre connexion avant de coder les controllers

Pagination

- Implémentez `?page=1&limit=10` sur GET /api/books
- Exemple : page 1 = doc 0-9, page 2 = doc 10-19

Tests

- Testez tous les endpoints avec Postman avant le rendu
- Incluez des cas d'erreur (400, 401, 404, 500)
- Exportez la collection JSON de Postman

🎬 Checklist finale avant rendu

- Code pushé sur GitHub (public)
- README.md complet avec instructions
- Collection Postman exportée (JSON)
- Tous les endpoints testés et fonctionnels
- `.env` et `firebase-config.json` dans `.gitignore`
- Messages de commit clairs et en français
- Pas d'erreurs console ou warnings
- Code formaté et lisible (indentation cohérente)
- Lien GitHub envoyés au formateur

Bon courage et amusez-vous bien avec ce projet ! 🚀