

BiblioTech

I. Sommaire

| | |
|--|-----------|
| I. Sommaire..... | 2 |
| II. Pré-Requis du Dossier de Projet..... | 4 |
| 1. Compétences Couvertes par le Projet..... | 4 |
| 2. Contexte du Projet..... | 4 |
| 3. Réalisations Significatives..... | 5 |
| 4. Veille Technologique..... | 5 |
| III. Liste des compétences du référentiel..... | 6 |
| IV. Présentation du projet..... | 8 |
| 1. Positionnement de l'entreprise..... | 8 |
| 2. Objectifs généraux..... | 8 |
| 3. Public cible..... | 8 |
| 4. Enjeux du projet..... | 9 |
| V. Cahier des charges..... | 10 |
| A. Présentation de l'initiative BiblioTech..... | 10 |
| B. Caractéristiques Techniques Souhaitées..... | 10 |
| 1) Interface Web..... | 10 |
| 2) Backend et API..... | 10 |
| 3) Base de Données MariaDB et MongoDB..... | 11 |
| 4) Sécurité..... | 11 |
| 5) RGPD & conformité..... | 12 |
| 6) Déploiement..... | 13 |
| 7) Accessibilité et UX..... | 13 |
| 8) Tests unitaires et vérification fonctionnelle..... | 14 |
| 9) Maintenance et évolutivité..... | 15 |
| 10) Application mobile Flutter..... | 15 |
| C. Étude Technologique..... | 16 |
| 1) Choix du front-end : Vue.js..... | 16 |
| 2) Choix du back-end : Node.js + Express..... | 17 |
| 3) Base de données : MariaDB & MongoDB..... | 17 |
| 4) Stockage & fichiers (ebooks PDF/EPUB)..... | 18 |
| 5) Sécurité et authentification (JWT, Helmet)..... | 19 |
| 6) Accessibilité & SEO..... | 20 |
| 7) Application mobile : Flutter (offline, synchro, UI réactive)..... | 20 |
| D. Cadre légal et responsabilités..... | 21 |
| VI. Planification et gestion du projet..... | 23 |
| 1. Méthodologie de gestion..... | 23 |
| 2. Chronologie du projet..... | 23 |
| 3. Organisation du code (structure du projet)..... | 24 |
| 4. Versionning du projet (Git)..... | 26 |
| VII. Scénarios d'utilisation..... | 26 |
| 1. Cas d'usage..... | 26 |
| 2. Parcours utilisateur(web & mobile)..... | 27 |
| VIII. API REST : Spécification technique..... | 28 |
| 1. Authentification et Gestion des utilisateurs..... | 28 |
| 2. Gestion des livres numériques..... | 28 |
| 3. Favoris et historique..... | 29 |

| | |
|---|-----------|
| 4. Commentaires sur les livres..... | 29 |
| 5. Signalement de contenu..... | 29 |
| 6. Statistiques..... | 30 |
| IX. Tests et assurance qualité..... | 32 |
| 1. Stratégie de test..... | 32 |
| 2. Résultats des tests unitaires et fonctionnels..... | 32 |
| X. Déploiement et infrastructure..... | 33 |
| 1. Plan de déploiement..... | 33 |
| 2. Infrastructure..... | 33 |
| XI. Documentation et support..... | 34 |
| 1. Documentation technique..... | 34 |
| 2. Support utilisateur..... | 34 |
| XII. Évaluation et perspectives..... | 35 |
| 1. Évaluation du projet..... | 35 |
| 2. Pistes d'amélioration et évolutions futures..... | 35 |
| XIII. Annexe..... | 36 |

Choses à mettre dans le dossier:

- **V.D - Cadre légal et responsabilités** : Résumé de la démarche légale + mention du Safe Harbor
- **V.B.1 - Interface Web** : Description de l'interface web et de ses intentions esthétiques
- **V.C - Étude Technologique** : Détails sur le choix technologique (sans le comparatif complet)
- **VIII - API REST** : Détail des fonctionnalités de l'API (liste simple, pas toutes les routes)
- **V.B.3 - Base de Données** : Structure globale de la base de données (sans schéma visuel)

Choses à mettre en annexe:

- **V.D - Cadre légal et responsabilités** : Extrait du texte de loi sur le statut d'hébergeur (Safe Harbor)
- **V.B.1 - Interface Web** : Maquettes de l'interface (Accueil connecté / non connecté / Contact)
- **V.B.3 - Base de Données** : Diagramme de base de données MariaDB (modèle relationnel)
- **V.B.3 - Base de Données** : Modélisation MongoDB (structure NoSQL)
- **V.C - Étude Technologique** : Comparatifs complet technos (Vue vs React, Node vs Django, etc.)
- **VIII - API REST : Spécification technique** : Exemples de routes API REST
- **V.B.2 - Backend et API et IX - Tests et assurance qualité** : Exemples de code (auth, middleware, etc.)
- **Maquette parcours utilisateur**

II. Pré-Requis du Dossier de Projet

1. Compétences couvertes par le projet

Le projet BiblioTech mobilise un large éventail de compétences relevant du métier de Développeur Web et Web Mobile (DWWM), tant sur le plan technique que méthodologique. Il permet de couvrir l'ensemble des blocs du référentiel de formation.

Parmi les principales compétences mises en œuvre :

- **Analyse du besoin** : Compréhension des enjeux liés à la diffusion d'ebooks et aux contraintes légales associées.
- **Modélisation** : Conception d'une base de données relationnelle et NoSQL adaptée à une plateforme documentaire.
- **Conception d'interfaces (UI/UX)** : Réalisation du design visuel, structuration des écrans, hiérarchie des éléments, expérience utilisateur fluide, responsive et accessible, sur web et mobile.
- **Développement front-end** : Création d'une interface utilisateur responsive avec Vue.js avec navigation, recherche, favoris, signalements.
- **Développement back-end** : Mise en place d'une API REST sécurisée avec Node.js + Express, incluant l'authentification, la gestion des contenus, des utilisateurs et des signalements.
- **Développement mobile** : Développement d'une application mobile Flutter permettant la lecture hors-ligne des livres téléchargés, avec synchronisation partielle.
- **Sécurité** : Mise en place d'un système de connexion chiffré, gestion des rôles et journalisation des activités sensibles.
- **Conformité RGPD** : Traitement sécurisé des données personnelles avec respect des droits utilisateurs.
- **Déploiement et maintenance** : Mise en production du projet, gestion de l'hébergement, et réflexion sur la maintenance à long terme.
- **Qualité logicielle** : Implémentation de tests unitaires, organisation du code, et versioning via Git.

Ce projet représente une application concrète et complète de l'ensemble des savoir-faire attendus d'un développeur web et mobile à l'issue de sa formation.

2. Contexte du projet

Dans un monde saturé de plateformes commerciales comme Amazon Kindle, Google Books ou encore Apple Books, l'accès à la lecture numérique est souvent conditionné par des abonnements, des formats verrouillés ou la collecte abusive de données personnelles. Il devient essentiel de proposer une alternative. Une alternative plus juste, plus ouverte, plus respectueuse.

C'est de cette idée qu'est née BiblioTech.

BiblioTech est une plateforme de partage du savoir, conçue pour être accessible à tous, sans barrières techniques ni économiques, et entièrement alimentée par sa communauté. Les utilisateurs peuvent y téléverser, lire, classer et commenter des livres électroniques, le tout gratuitement, sans publicité, sans exploitation de leurs données.

L'objectif est simple : remettre la lecture et la connaissance entre les mains de tous, y compris des publics souvent oubliés des grandes plateformes. Pas besoin de matériel coûteux, ni de budget, un simple téléphone suffit.

Accessible via une interface web moderne et une application mobile Flutter, BiblioTech est à la fois un outil personnel et un espace collectif. Pensé dans le respect de la vie privée, de la conformité légale (RGPD, statut d'hébergeur, gestion des signalements), et dans une démarche profondément éthique.

3. Réalisations significatives

Le développement de BiblioTech a permis de concrétiser un ensemble de livrables techniques et fonctionnels directement exploitables, témoignant d'une approche complète et professionnalisante.

Parmi les réalisations majeures :

- Conception complète d'une base de données relationnelle (MariaDB) et NoSQL (MongoDB), structurée autour des utilisateurs, des livres, des catégories, des commentaires, des signalements et de la sécurité.
- Développement d'une API REST sécurisée, assurant l'ensemble des fonctionnalités du projet (authentification, gestion des livres, favoris, signalements, historique, administration...).
- Création d'une interface web responsive en Vue.js, avec navigation fluide, recherche, filtres dynamiques, gestion de compte utilisateur, lecture en ligne.
- Développement d'une application mobile Flutter, synchronisée à l'API, permettant la consultation hors-ligne des ebooks téléchargés.
- Mise en place d'un système d'authentification JWT, avec gestion des rôles (utilisateur/admin), journalisation des connexions, bannissements temporaires, et suivi d'activité.
- Déploiement du projet sur un environnement distant avec gestion des dépendances, sécurité serveur et architecture maintenable.
- Implémentation de tests unitaires pour assurer la robustesse des composants critiques.
- Respect des contraintes RGPD, avec gestion des données personnelles, droit à la suppression, et stockage conforme.

Ce projet s'inscrit ainsi dans une logique de projet réaliste et maintenable, prêt à être utilisé par de vrais utilisateurs, avec une approche concrète de bout en bout : de la conception initiale à la mise en ligne.

4. Veille technologique

Tout au long du développement de BiblioTech, une veille technologique régulière a été menée. Elle a permis de guider les choix techniques, d'assurer la pertinence des outils utilisés, et de garantir une solution moderne, fiable et alignée sur les bonnes pratiques actuelles du développement web et mobile.

Choix des technologies : cohérence, simplicité et modernité

- Le projet repose sur une stack JavaScript unifiée (Vue.js, Node.js, Express) afin de conserver une homogénéité du langage entre le front-end et le back-end. Cela permet une meilleure lisibilité du code, une maintenance simplifiée, et une montée en compétence plus rapide.
- Vue.js a été retenu pour sa simplicité, sa légèreté, sa courbe d'apprentissage rapide et son excellent support de la réactivité. Comparé à Angular (trop lourd) ou React (plus complexe à structurer), Vue offre un excellent compromis pour un projet maintenu seul.
- Node.js + Express ont été choisis pour le serveur, en raison de leur compatibilité avec la structure REST, de leur large écosystème (npm), et de leur efficacité pour traiter des flux de données asynchrones.
- MongoDB + MariaDB : le choix d'un double stockage NoSQL/SQL répond à la fois aux besoins de performance (livres, documents, stats) et à ceux d'intégrité (utilisateurs, logs, signalements...).
- Flutter a été choisi pour l'application mobile pour sa capacité à produire des apps iOS/Android avec un seul codebase, tout en offrant de hautes performances et un bon support du mode hors-ligne.

Pratiques de développement modernes

- Authentification sécurisée avec JWT et middleware personnalisés.
- Utilisation de Git avec branches fonctionnelles, commits clairs, et versioning rigoureux.
- Tests unitaires pour assurer la stabilité de l'authentification, des routes critiques, et du modèle utilisateur.
- Front-end respectant les principes d'accessibilité (navigabilité clavier, contraste, responsive design).

Veille éthique et réglementaire

- Suivi des directives RGPD : droit à la suppression, transparence sur les données, stockage local, aucune exploitation commerciale.
- Intégration des principes liés au statut d'hébergeur et au cadre du Safe Harbor : les utilisateurs sont responsables de ce qu'ils publient, mais l'administrateur peut intervenir via le système de signalement.
- Veille autour des modèles alternatifs aux plateformes fermées comme Amazon Kindle ou Google Books : auto-hébergement, transparence, open source, liberté de lecture.

Outils et sources de veille utilisés

Pour rester à jour et affiner mes choix techniques :

- Daily.dev : agrégateur d'actualités tech orienté développeurs.
- YouTube (FR) :
 - [Grafikart](#) – tutos avancés et reviews techniques.
 - [Micode](#) – vulgarisation cybersécurité/développement.
 - [Graven](#) – tutoriels fullstack & mobile.
 - [Benjamin Code](#) – contenu créatif sur les tendances du web.
 - [V2F](#) – vulgarisation développement et informatique
- LinkedIn : Suivi d'experts tech francophones pour des retours concrets sur les pratiques de développement actuelles.

III. Liste des compétences du référentiel

Le projet BiblioTech permet de mettre en œuvre l'intégralité des compétences visées par le titre professionnel de Développeur Web et Web Mobile (DWWM). Ce projet couvre aussi bien les compétences techniques (front-end, back-end, bases de données, sécurité, déploiement) que les compétences transverses (veille, documentation, organisation). Les choix techniques et méthodologiques ont été faits en cohérence avec les exigences du référentiel, dans un souci de qualité, de conformité et d'efficacité.

- Maquetter des interfaces utilisateur web ou web mobile :

Avant de commencer le développement, j'ai conçu des maquettes pour les différentes interfaces de BiblioTech, en version desktop et mobile. Ces maquettes ont été réalisées en respectant les principes d'ergonomie, la hiérarchisation des informations, ainsi que la compatibilité avec les contraintes d'accessibilité. Chaque parcours utilisateur a été réfléchi pour proposer une navigation claire et logique, y compris pour les personnes en situation de handicap.

- Réaliser des interfaces utilisateur statiques web ou web mobile :

En partant des maquettes validées, j'ai intégré les interfaces en HTML/CSS via le framework Vue.js. Le code est structuré, sémantique, et respecte les normes d'accessibilité (contrastes, balises ARIA, navigation clavier). J'ai également veillé à optimiser le chargement des pages et l'affichage responsive pour différents types d'écrans.

- Développer la partie dynamique des interfaces utilisateur web ou web mobile :

Grâce à Vue.js, j'ai mis en place des composants dynamiques réactifs pour la navigation, la gestion des favoris, des signalements, la recherche dans les livres, ou encore la synchronisation mobile. L'application est fluide et réagit aux actions utilisateurs en temps réel.

- Mettre en place une base de données relationnelle :

Le projet repose sur une base MariaDB pour gérer les utilisateurs, les signalements, les historiques, et les commentaires. J'ai défini un schéma conceptuel puis physique conforme, avec clés étrangères, contraintes d'intégrité, et gestion des utilisateurs avec droits distincts.

- Développer des composants d'accès aux données SQL et NoSQL :

J'ai développé des modules d'accès aux données pour interagir avec MariaDB et MongoDB. MariaDB sert les données structurées tandis que MongoDB stocke les métadonnées des fichiers numériques pour permettre des recherches rapides et une meilleure scalabilité. Les requêtes sont sécurisées, les entrées validées, et des tests unitaires garantissent leur fiabilité.

Développer des composants métier côté serveur :

Le back-end repose sur Node.js avec Express. J'ai créé des routes REST structurées, la logique métier pour les fonctionnalités critiques (authentification JWT, rôles utilisateurs, gestion des uploads, lecture mobile), et l'architecture suit le modèle MVC pour plus de clarté.

Documenter le déploiement d'une application dynamique web ou mobile :

Le déploiement de BiblioTech est prévu sur un hébergement mutualisé chez OVH. J'ai rédigé une documentation claire et structurée qui détaille la configuration du serveur (Node.js, bases de données MariaDB/MongoDB, gestion des ports, certificats SSL via OVH, etc.), les étapes de mise en ligne (installation des dépendances, variables d'environnement, scripts de démarrage), ainsi que les actions de maintenance de base. L'objectif est de permettre une installation simple et reproductible, adaptée à un environnement mutualisé classique.

Installer et configurer son environnement de travail en fonction du projet :

L'environnement de développement est configuré manuellement afin de rester proche d'une configuration classique de serveur OVH, qui est la cible de déploiement. J'utilise PhpStorm comme IDE principal pour sa richesse fonctionnelle, ainsi que Git pour le versioning. Node.js est géré avec NVM pour garantir la compatibilité des versions. Un fichier README clair et détaillé permet à n'importe quel développeur de cloner le projet, installer les dépendances, configurer les variables d'environnement et lancer l'application localement.

Appliquer une démarche de résolution de problème :

Tout au long du développement, j'ai fait face à des bugs ou à des blocages techniques (gestion des fichiers volumineux, sécurité des uploads, synchronisation mobile...). À chaque fois, j'ai procédé par étapes : reproduction du bug, analyse, recherche de solutions (forums, documentation, test), mise en œuvre et documentation des correctifs.

Communiquer en français et en anglais :

Le code est commenté dans un anglais technique standard. Les messages utilisateurs et la documentation sont disponibles en français et en anglais. Le projet utilise des termes cohérents et compréhensibles, tant pour les développeurs que pour les utilisateurs finaux.

Effectuer une veille technologique en continu :

Tout au long du projet, j'ai effectué une veille quotidienne via la plateforme Daily.dev, YouTube (Grafikart, Benjamin Code, Le Designer du Web et V2F) et des développeurs francophones sur LinkedIn. Cette veille m'a permis de faire des choix techniques éclairés (Vue.js, Flutter, MongoDB), d'adopter de bonnes pratiques de sécurité, et de suivre les évolutions des outils utilisés.

IV. Présentation du projet

1. Positionnement de l'entreprise

BiblioTech est née d'une conviction simple : la connaissance n'appartient à personne. Elle doit pouvoir circuler librement, sans barrière financière, technique ou sociale. Ce projet, c'est d'abord une réponse à une frustration que j'ai longtemps connue : ne pas pouvoir lire librement, autant que je l'aurais voulu.

La lecture a toujours été pour moi une source d'évasion, d'apprentissage, de réflexion. Mais quand l'accès à un livre dépend de ton budget, de ton matériel ou d'une plateforme commerciale verrouillée, ce droit fondamental devient un privilège. C'est contre ça que BiblioTech se positionne.

À rebours des géants de l'édition numérique comme Amazon Kindle ou Google Books, BiblioTech propose une bibliothèque collaborative, libre, sans publicité, ni vente de données, ni frais cachés. Chaque utilisateur est à la fois lecteur et contributeur : il peut téléverser, organiser, lire ou commenter des livres, et cela gratuitement, avec une seule exigence : le respect des autres.

Ce projet n'est pas un produit, c'est un outil. Un outil au service de l'émancipation culturelle, porté par une démarche éthique, communautaire, et résolument ouverte. BiblioTech, c'est une bibliothèque bâtie par tous, pour tous.

2. Objectifs généraux

Le projet BiblioTech poursuit une ambition simple : rendre la lecture accessible à tous, partout, sans condition. Il ne s'agit pas seulement de créer une application ou un site web, mais de poser les bases d'un véritable écosystème de partage et de transmission du savoir.

L'objectif principal est de permettre à chacun, quelle que soit sa situation sociale, géographique ou matérielle, d'accéder à une bibliothèque numérique riche, vivante, et collaborative. Cela implique :

- Une plateforme gratuite, sans publicité ni collecte abusive de données ;
- Une accessibilité maximale : interface responsive, application mobile Flutter, compatibilité tous appareils ;
- Un fonctionnement communautaire : les utilisateurs enrichissent la bibliothèque eux-mêmes, en téléversant et classant leurs propres livres ;
- Un respect strict des données et de la vie privée, en accord avec les principes RGPD ;
- Une ouverture technique et éthique, en opposition aux systèmes fermés ou propriétaires des grandes plateformes de lecture numérique.

Mais au-delà de l'outil, BiblioTech cherche à cultiver un état d'esprit : celui de la curiosité, du partage, de l'égalité d'accès à la connaissance. En ce sens, c'est autant un projet technique qu'un engagement personnel et citoyen.

3. Public cible

BiblioTech s'adresse à tous ceux pour qui la lecture n'est pas une option, mais un besoin. À ceux qui veulent apprendre, s'évader, comprendre le monde, mais qui se heurtent à des barrières économiques, techniques ou culturelles.

Le projet vise en priorité :

- Les personnes en situation de précarité, pour qui l'achat de livres ou d'abonnements reste un luxe ;
- Les étudiants, souvent en quête de ressources gratuites et fiables pour se former ou réviser ;
- Les autodidactes, curieux et passionnés, qui cherchent à se cultiver hors des circuits traditionnels ;
- Les lecteurs nomades, qui veulent retrouver leurs livres partout, sur n'importe quel appareil ;

- Les personnes en zone blanche ou à faible couverture culturelle, où les bibliothèques physiques sont absentes ou insuffisantes ;
- Les communautés éducatives ou associatives, qui peuvent utiliser la plateforme comme un outil pédagogique collaboratif.

Mais au fond, le public de BiblioTech, c'est chacun de nous. Dès lors qu'on pense que la lecture ne devrait jamais dépendre de son compte en banque, de sa géolocalisation, ou de l'écosystème d'un géant du web.

4. Enjeux du projet

BiblioTech se confronte à des enjeux à la fois techniques, sociaux, éthiques et culturels. Ce n'est pas simplement une plateforme fonctionnelle : c'est un engagement.

1) Enjeu d'accessibilité

Rendre la lecture universellement accessible, même pour celles et ceux qui n'ont pas les moyens, pas le matériel, ou pas le réseau. L'enjeu est d'atteindre une ergonomie simple, une interface fluide, et une compatibilité maximale – même sur un vieux smartphone.

2) Enjeu de souveraineté culturelle

Refuser de dépendre des plateformes commerciales qui verrouillent l'accès à la culture et exploitent les données des utilisateurs. Offrir une alternative qui ne vend rien, ne collecte pas, ne surveille pas. BiblioTech est une réponse éthique à un marché dominé par Amazon Kindle, Google Books ou Apple Books.

3) Enjeu communautaire

Construire un écosystème basé sur le partage, l'entraide et la contribution volontaire. Cela demande une modération intelligente, une architecture pensée pour favoriser l'interaction sans dérive, et des outils simples pour publier, commenter ou signaler.

4) Enjeu technique

Créer une plateforme robuste, sécurisée, évolutive. Répondre aux défis d'un environnement full JavaScript (Vue.js, Node.js, MongoDB, Flutter), penser la scalabilité sans cloud centralisé, assurer la pérennité du code et la portabilité de l'architecture.

5) Enjeu de légalité

Respecter le cadre juridique, notamment en matière de droit d'auteur et de responsabilité d'hébergeur. Mettre en place un système de signalement des contenus litigieux, garantir la conformité RGPD et protéger les données des utilisateurs.

6) Enjeu personnel

BiblioTech, c'est aussi un défi personnel : prouver qu'un projet libre, gratuit et éthique peut tenir face à des logiques commerciales écrasantes. C'est mon engagement de développeur, convaincu que la technologie peut (et doit) servir le bien commun.

V. Cahier des charges

A. Présentation de l'initiative BiblioTech

BiblioTech est une plateforme collaborative dédiée à la diffusion libre et gratuite d'ebooks. Elle vise à créer un espace numérique où chacun peut lire, partager, organiser ou commenter des livres électroniques, sans contrainte technique, financière ou publicitaire.

L'initiative repose sur un principe simple : l'accès à la lecture ne doit pas être un luxe. Dans un paysage dominé par les géants du numérique comme Amazon Kindle, Google Books ou Apple Books, BiblioTech offre une alternative éthique, communautaire et respectueuse des utilisateurs.

Chaque fonctionnalité de la plateforme découle de cette ambition : permettre à tous de s'approprier la lecture. Qu'il s'agisse de l'interface web, de l'application mobile, de la gestion des contenus, ou encore de la modération, tout est pensé pour répondre à une double exigence : simplicité d'usage et respect de la vie privée.

BiblioTech n'est pas un produit à vendre, mais un projet à faire vivre. Il s'adresse autant aux lecteurs qu'aux développeurs, bibliothécaires, associations, ou simples citoyens soucieux de défendre une culture numérique libre. L'objectif est de bâtir ensemble une bibliothèque sans verrou, ouverte à tous, partout.

B. Caractéristiques Techniques Souhaitées

1) Interface Web

L'interface web de BiblioTech a été pensée pour être à la fois chaleureuse, accessible et fonctionnelle. Inspirée d'un univers visuel doux et accueillant (notamment avec l'utilisation de mascottes félines, de tons clairs et contrastés, et d'illustrations ludiques), elle cherche à créer une ambiance positive, agréable, presque familière.

Sur le plan technique, elle est développée avec Vue.js, dans une architecture en composants réutilisables. L'accent a été mis sur :

- La lisibilité : typographie claire, tailles adaptées, couleurs accessibles.
- La structure logique : header fixe, navigation cohérente, catégories bien identifiables.
- La réactivité : l'interface s'adapte aux écrans mobiles et desktop grâce à un design responsive rigoureux.
- L'accessibilité : conformité aux bonnes pratiques (contrastes WCAG, navigation clavier, balises ARIA).
- La performance : chargement rapide, compression des assets, lazy loading des composants lourds.

Chaque utilisateur peut accéder à l'essentiel de la plateforme sans friction. L'interface ne cherche pas à impressionner, elle cherche à rassurer et accompagner, à l'image de la mascotte du site : un petit chat bibliothécaire prêt à guider avec bienveillance. Des exemples visuels de l'interface sont disponibles en annexe.

2) Backend et API

Le backend de BiblioTech repose sur une architecture solide, modulaire et sécurisée, développée en Node.js avec le framework Express. Il s'agit d'une API REST conçue pour offrir des performances stables, une bonne scalabilité et une documentation claire.

Voici les principaux aspects techniques de cette API :

- Architecture modulaire : Le projet suit une organisation en dossiers (routes, controllers, middlewares, services, modèles) qui facilite la maintenance et les évolutions futures.

- Sécurité :
 - Authentification par JWT (JSON Web Token),
 - Middleware de validation des entrées (Joi, express-validator),
 - Sécurisation des en-têtes HTTP avec Helmet,
 - Gestion des erreurs centralisée.
- Fonctionnalités principales exposées par l'API :
 - Inscription, connexion, gestion du profil utilisateur,
 - Téléversement et récupération des fichiers (ebooks),
 - Recherche, lecture, classement et commentaires des livres,
 - Gestion des favoris, de l'historique, des signalements,
 - Synchronisation des lectures avec l'application mobile.
- Séparation des responsabilités claire : chaque couche de l'API a un rôle précis (validation, logique métier, persistance des données), ce qui garantit clarté et stabilité.
- Tests : Des tests unitaires couvrent les principaux endpoints sensibles.
- Journalisation : Suivi des événements critiques et tentatives suspectes (log des connexions, échecs d'authentification, etc.).

L'API est utilisée à la fois par l'interface web et l'application mobile, ce qui garantit la cohérence des fonctionnalités sur l'ensemble des supports.

3) Base de Données MariaDB et MongoDB

Le projet repose sur une architecture hybride combinant une base de données relationnelle (MariaDB) et une base orientée documents (MongoDB), afin de répondre à la fois aux besoins de structuration stricte et de flexibilité.

MariaDB est utilisée pour stocker toutes les données structurées : comptes utilisateurs, rôles, historiques de consultation, commentaires, signalements, suivis, ainsi que les fiches de livres elles-mêmes (titre, auteur, description, date d'ajout, etc.). Cette structure relationnelle permet de garantir l'intégrité des données via des contraintes, des relations (clés étrangères) et des index adaptés. Elle est idéale pour gérer des entités clairement définies et fortement liées entre elles, comme l'utilisateur et ses interactions avec les livres.

MongoDB, en complément, est utilisée pour stocker les métadonnées non structurées ou volumineuses liées aux fichiers : informations extraites (via OCR ou parsing EPUB/PDF), tags libres, miniatures, voire traces de lecture. Elle offre la flexibilité nécessaire pour adapter les modèles sans migration lourde, notamment pour les fonctions de recherche avancée ou l'analyse de contenu.

Cette séparation claire entre données structurées (MariaDB) et données non structurées (MongoDB) garantit une performance optimale, une meilleure organisation, et une ouverture vers des évolutions fonctionnelles futures sans sacrifier la robustesse des fondations.

4) Sécurité

La sécurité est un pilier fondamental de BiblioTech, tant pour protéger les données des utilisateurs que pour assurer la fiabilité et l'intégrité du service. L'ensemble de l'architecture a été pensé pour limiter au maximum les failles potentielles et garantir un usage serein.

Sur le back-end, l'API Express est protégée par plusieurs couches :

- Authentification via JSON Web Tokens (JWT), avec expiration contrôlée et rafraîchissement sécurisé.
- Middleware de contrôle d'accès basé sur les rôles utilisateurs (lecture simple, contributeur, modérateur, administrateur).
- Sanitization et validation systématique des entrées utilisateur pour éviter les injections SQL ou XSS.
- Sécurisation des en-têtes HTTP avec le middleware Helmet, qui ajoute des headers comme Content-Security-Policy, X-Frame-Options, ou Strict-Transport-Security.

Côté base de données, MariaDB et MongoDB sont configurées avec des utilisateurs distincts, aux droits restreints. Les accès en écriture/lecture sont isolés selon les besoins du service (pas d'accès global), et une stratégie de sauvegarde régulière est en place.

En front-end, l'application évite tout rendu de contenu dynamique non maîtrisé, en échappant systématiquement les entrées utilisateurs dans les commentaires et autres zones sensibles. Le tout est servi en HTTPS, avec certificats SSL déployés via OVH.

Enfin, une veille active sur les vulnérabilités (faille Zero-Day, failles connues dans les bibliothèques utilisées, etc.) est maintenue tout au long du développement. Elle alimente une logique de mise à jour régulière des dépendances et un audit de sécurité manuel à chaque livraison majeure.

5) RGPD & conformité

En tant que plateforme de partage de contenus numériques impliquant la gestion de données personnelles (comptes utilisateurs, historiques de lecture, favoris...), BiblioTech a été conçu dès le départ dans une logique de conformité au RGPD (Règlement Général sur la Protection des Données).

Collecte minimale des données

Aucune donnée n'est collectée sans nécessité :

- uniquement un email (pour l'identification),
- un nom d'utilisateur,
- et un mot de passe chiffré.

Aucune donnée sensible ou superflue n'est demandée. Aucune géolocalisation, aucun suivi comportemental, ni aucun cookie de tracking tiers ne sont utilisés.

Consentement et droits des utilisateurs

À l'inscription, l'utilisateur est informé de façon claire des données collectées et de leur finalité. Il peut :

- accéder à ses données,
- les modifier,
- demander leur suppression via son espace profil.

Ces droits sont directement intégrés dans l'interface utilisateur, sans besoin de contact manuel avec un administrateur.

Durée de conservation limitée

Les données sont conservées tant que l'utilisateur est actif. Une politique d'anonymisation est prévue après une période prolongée d'inactivité (12 mois), ou à la suppression de compte.

Stockage et sécurité des données

Toutes les données sont stockées sur des bases sécurisées (MariaDB et MongoDB), isolées, avec accès restreint. Les mots de passe sont hashés avec bcrypt, et aucune donnée n'est stockée en clair.

Aucune monétisation des données

BiblioTech ne contient aucune publicité, aucun pistage et aucune revente de données. Les données des utilisateurs ne quittent jamais le serveur.

Hébergement et statut d'hébergeur

Conformément à la directive e-commerce et au statut de prestataire technique, BiblioTech agit en tant qu'hébergeur de contenus. Cela signifie que :

- les utilisateurs sont responsables des fichiers qu'ils téléversent,
- une procédure de signalement est prévue,
- et les contenus litigieux peuvent être retirés sur demande.

Ce positionnement est également encadré par le régime du Safe Harbor, dont un extrait est disponible en annexe.

6) Déploiement

Le déploiement de BiblioTech a été pensé pour être fiable, reproductible et accessible, même sans infrastructure complexe. Le projet est hébergé sur OVH via une offre mutualisée, ce qui impose certaines contraintes mais garantit une accessibilité grand public, en accord avec les valeurs du projet.

Infrastructure choisie

- Hébergement : OVH mutualisé (Linux, Node.js compatible)
- Base de données : MariaDB intégrée, MongoDB via cluster distant (MongoDB Atlas)
- Domaine et certificat SSL : Gérés directement sur OVH, avec certificat Let's Encrypt intégré
- Stockage de fichiers : Dossier uploads/ sécurisé, accessible uniquement via des routes contrôlées par JWT

Mise en production

Le projet est déployé manuellement via FTP sécurisé, avec :

- Transfert du code Node.js (backend),
- Configuration des variables d'environnement dans un fichier .env,
- Lancement du serveur via un process manager type pm2 côté serveur (lorsque disponible),
- Lien entre le frontend compilé (dist/ de Vue.js) et le backend via une redirection catch-all.

Scripts et documentation

Un README complet accompagne le projet avec :

- Les étapes d'installation,
- Les prérequis d'environnement,
- Les scripts de build et de démarrage,
- Les instructions pour relancer manuellement l'API en cas de plantage.

Sécurité et surveillance

- Les erreurs serveur sont loguées côté backend via un système de logs maison.
- Les fichiers uploadés sont contrôlés côté serveur (type MIME, taille).
- Le dossier de production est séparé du dépôt Git, pour éviter toute fuite de fichiers de configuration.

Évolutivité future

Ce système de déploiement peut être adapté à une montée en charge via :

- Un passage à un VPS si nécessaire (toujours chez OVH),
- Une intégration continue (CI/CD) avec GitHub Actions,
- Une séparation complète frontend/backend avec hébergement du front sur Netlify ou Vercel.

7) Accessibilité et UX

L'accessibilité et l'expérience utilisateur (UX) sont au cœur de la conception de BiblioTech. La plateforme se veut inclusive, agréable à utiliser et intuitive, quelles que soient les compétences techniques ou les capacités physiques de l'utilisateur.

Accessibilité

Pour garantir un accès équitable à tous les publics, BiblioTech respecte les bonnes pratiques d'accessibilité numérique :

- Contrastes respectés : palettes de couleurs vérifiées via des outils WCAG pour assurer une lecture confortable.
- Navigation clavier complète : tous les éléments interactifs sont atteignables au clavier.
- Balises ARIA : utilisées pour améliorer la sémantique et permettre aux lecteurs d'écran de décrire correctement les éléments (menus, sections dynamiques, boutons personnalisés).
- Structure logique des pages : titres hiérarchisés (<h1>, <h2>, etc.), navigation claire et linéaire.

Le but : qu'un utilisateur malvoyant ou en situation de handicap puisse consulter et interagir avec l'interface sans obstacle.

Expérience Utilisateur (UX)

L'UX a été pensée dans une optique de simplicité, cohérence et bienveillance :

- Interface douce et rassurante : mascottes félines, tons pastels, éléments arrondis créent un environnement chaleureux.
- Guidage naturel : chaque page va droit au but, avec des boutons explicites et des flux utilisateurs sans détour.
- Réactivité et fluidité : grâce à Vue.js, chaque interaction est instantanée, sans rechargement de page.
- Responsive design : que ce soit sur mobile, tablette ou desktop, l'interface s'adapte sans perte de confort.

Test et retours

Des tests utilisateurs ont été réalisés auprès de profils variés (lecteurs occasionnels, étudiants, utilisateurs peu technophiles) afin de valider les parcours critiques : navigation, lecture, ajout de favoris, téléchargement. Ces retours ont permis d'ajuster les textes, tailles, couleurs, et hiérarchie des éléments.

8) Tests unitaires et validation fonctionnelle

La qualité du code et la fiabilité des fonctionnalités sont des priorités dans le développement de BiblioTech. Pour garantir un fonctionnement robuste et prévenir les régressions, plusieurs niveaux de tests ont été mis en place tout au long du projet.

Tests unitaires

Des tests unitaires ont été écrits pour les modules critiques du back-end, notamment :

- Les routes d'authentification (vérification des jetons JWT, contrôle des rôles)
- Les contrôleurs de gestion des e-books (ajout, suppression, récupération, filtrage)
- Les interactions avec la base MariaDB (requêtes utilisateurs, favoris, historiques)
- Les middlewares de validation (inputs, sécurité, protection contre les abus)

Ces tests ont été réalisés avec Jest, framework de test JavaScript, en mockant les dépendances pour isoler la logique métier. Chaque test couvre des cas simples, des cas limites, et des erreurs attendues.

Validation fonctionnelle

Avant chaque itération, des scénarios fonctionnels ont été définis pour s'assurer du bon déroulement des parcours utilisateurs :

- Connexion / inscription / récupération de compte
- Téléversement et consultation d'un livre
- Ajout / suppression de favoris
- Signalement de contenu
- Navigation dans les bibliothèques (recherche, filtrage)
- Synchronisation avec l'application mobile

Ces tests ont été exécutés manuellement sur plusieurs navigateurs et tailles d'écran, ainsi que sur mobile (via Flutter) pour garantir une expérience cohérente.

Documentation des tests

Tous les tests réalisés sont documentés dans une section dédiée du dépôt Git, incluant :

- L'objectif de chaque test
- Les entrées et sorties attendues
- Les erreurs gérées
- La capture des résultats (succès ou échec)

9) Maintenance et évolutivité

Le projet BiblioTech a été conçu dès le départ dans une logique de maintenabilité et d'évolution progressive. Chaque composant (back-end, front-end, mobile) repose sur une structure modulaire, claire, et bien isolée, ce qui permet de corriger, améliorer ou ajouter des fonctionnalités sans impacter l'ensemble du système.

Le code côté serveur suit une architecture MVC (Model – View – Controller), avec séparation nette des responsabilités. Les routes sont regroupées par domaine (authentification, utilisateurs, livres, etc.), facilitant la lisibilité. Le front-end, lui, repose sur des composants Vue.js réutilisables et testables. Quant à l'application Flutter, elle est construite autour de widgets bien découpés et maintenus.

Les mises à jour sont facilitées par un bon versionning (Git), une documentation claire, et une logique de développement incrémental. La structure des bases de données permet d'ajouter de nouvelles collections ou colonnes sans casser les données existantes.

Enfin, le choix de technologies populaires (Vue.js, Node.js, Flutter, MongoDB) garantit une forte communauté, une documentation abondante, et une compatibilité à long terme.

Perspectives d'évolution

Afin de continuer à faire évoluer BiblioTech dans une logique d'accessibilité et d'innovation, plusieurs pistes sont envisagées :

- Génération automatique d'eBooks via IA : Développement d'un système capable de scanner un livre papier (via OCR et IA), puis d'en générer automatiquement un eBook structuré (EPUB ou PDF), lisible, et partageable sur la plateforme.
- Lecture vocale intégrée (TTS) : Intégration d'un système de synthèse vocale permettant de lire les livres à haute voix, notamment pour les personnes malvoyantes, dyslexiques, ou en situation d'illettrisme.
- Programme d'alphabétisation numérique : Création de modules pédagogiques accessibles depuis l'application (web et mobile), pour accompagner les personnes en difficulté de lecture à progresser grâce à des exercices interactifs, des histoires lues, ou des quiz simples.

Ces pistes s'inscrivent dans la mission fondatrice du projet : rendre la lecture et le savoir accessibles à tous, sans discrimination, sans barrière.

10) Application mobile Flutter

L'application mobile de BiblioTech, développée avec Flutter, prolonge l'expérience utilisateur au-delà du navigateur. Elle répond à une demande claire : pouvoir accéder à sa bibliothèque n'importe où, même sans connexion permanente, sur un appareil mobile.

- **Expérience utilisateur fluide et moderne** : Flutter permet de concevoir une interface mobile réactive et agréable, avec des animations fluides, des transitions soignées et un design cohérent avec la version web. L'application reprend l'univers visuel de BiblioTech, avec une navigation intuitive et épurée.
- **Lecture hors ligne** : L'un des objectifs clés de l'app est de permettre à l'utilisateur de lire même sans connexion. Les livres ajoutés à ses favoris peuvent être téléchargés localement dans l'application. Ces fichiers sont stockés en cache sécurisé et accessibles via une bibliothèque hors ligne.
- **Synchronisation avec le compte utilisateur** : Grâce à l'API REST sécurisée développée pour la plateforme, l'application mobile récupère les informations du compte utilisateur : favoris, historique de lecture, commentaires, etc. La synchronisation se fait à la connexion, mais reste tolérante aux interruptions réseau.

- **Fonctionnalités principales disponibles** : L'app propose l'essentiel : consultation de la bibliothèque, lecture de fichiers EPUB et PDF, ajout aux favoris, signalement de contenu, et navigation par catégories ou mots-clés. Les interactions sont adaptées au mobile avec des gestes simples (swipe, tap long, etc.).
- **Performances et optimisation** : Flutter compile en code natif pour Android et iOS, garantissant des performances élevées et une bonne gestion des ressources, même sur des appareils anciens. La taille de l'application est contenue, et les ressources sont chargées à la demande pour limiter la consommation de données.
- **Accessibilité mobile** : L'interface respecte les standards d'accessibilité mobile : contraste élevé, navigation à une main, compatibilité avec les lecteurs d'écran, et taille de police ajustable. Cela garantit une expérience inclusive, même sur de petits écrans.
- **Évolutivité et maintenance** : Le code Flutter est organisé en widgets réutilisables. Les mises à jour peuvent être déployées facilement via le store, et une API dédiée permet à terme de faire évoluer les fonctionnalités mobiles sans modifier entièrement l'architecture.

En bref, l'application Flutter est une extension logique de BiblioTech, fidèle à sa mission d'accessibilité et de liberté, pensée pour répondre aux usages modernes et offrir une lecture libre... même dans le métro sans réseau.

C. Étude technologique

Le développement de BiblioTech repose sur une architecture moderne, robuste et harmonieuse. J'ai volontairement choisi une stack JavaScript unifiée (Vue.js côté client, Node.js côté serveur) pour sa cohérence, sa flexibilité et la richesse de son écosystème. Ce choix technique m'a permis de mieux structurer le code, de gagner en productivité, et de maintenir une logique fluide sur l'ensemble du projet.

1) Choix du front-end : Vue.js

Pour le développement de l'interface utilisateur, j'ai choisi Vue.js, un framework JavaScript moderne, léger, et axé sur la simplicité. Ce choix s'est imposé comme une évidence dans le cadre d'un projet solo visant à construire une interface dynamique, maintenable, et accessible.

Pourquoi Vue.js ?

Vue.js est conçu pour être facile à prendre en main, ce qui permet de monter rapidement en compétence tout en adoptant de bonnes pratiques. Son approche basée sur les composants permet de structurer l'interface en éléments réutilisables et testables.

Performance et expérience utilisateur

Grâce au DOM virtuel, Vue.js permet un rendu très rapide, même sur des listes volumineuses comme celles des e-books. Le routing SPA offre une navigation fluide, sans rechargement de page. Tout a été conçu pour minimiser les temps d'attente et maximiser la fluidité.

Accessibilité et SEO

L'ensemble du front respecte les bonnes pratiques : balises ARIA, contrastes suffisants, navigation clavier, titres structurés. Des balises meta dynamiques ont été intégrées pour améliorer le SEO malgré l'aspect SPA.

Vue.js face à d'autres frameworks

- **React.js** : Plus populaire mais également plus complexe à structurer sans configuration initiale. Vue propose un écosystème intégré, là où React nécessite de choisir et assembler soi-même chaque brique.
- **Angular** : Bien que robuste, Angular est plus lourd, plus rigide, et exige une courbe d'apprentissage plus importante. Pour un projet solo comme BiblioTech, Vue.js offre une meilleure efficacité.

Cohérence avec le reste du projet

Vue.js s'intègre naturellement avec Node.js côté back-end. L'utilisation d'un même langage (JavaScript) des deux côtés de la stack renforce la cohérence du code, facilite le debug et la maintenance, et réduit le risque d'erreurs de conversion ou de logique.

2) Choix du back-end : Node.js + Express

Pour le développement du back-end de BiblioTech, j'ai opté pour Node.js, associé au framework Express. Ce duo constitue aujourd'hui l'une des solutions les plus performantes, souples et populaires pour construire des API modernes, et il s'intègre parfaitement à une stack full JavaScript.

Pourquoi Node.js ?

Node.js repose sur une architecture asynchrone non-bloquante, basée sur une boucle d'événements. Ce modèle est particulièrement adapté aux projets nécessitant la gestion de nombreuses connexions simultanées, exactement ce que propose BiblioTech : consultation d'ebooks, recherches en temps réel, téléchargements et téléversements de fichiers. Node.js offre ainsi une excellente performance sans surcharger le serveur.

Autre avantage décisif : le fait de pouvoir travailler avec un seul langage (JavaScript) pour tout le projet. Cela réduit la complexité, facilite la lecture du code, et permet de mutualiser les outils et les bonnes pratiques entre le front et le back.

Express.js pour structurer l'API

Express est un framework minimaliste, mais extrêmement puissant. Il me permet de construire une API REST claire, évolutive et bien structurée. Les routes sont organisées par modules fonctionnels (authentification, gestion des utilisateurs, gestion des livres, signalements, statistiques, etc.), chacun encapsulant sa logique propre, ce qui favorise la lisibilité et la maintenabilité.

J'utilise également une série de middlewares spécialisés :

- express-validator pour la validation des entrées utilisateurs,
- helmet pour sécuriser les en-têtes HTTP,
- express-rate-limit pour limiter les abus,
- cors, morgan, etc. pour la gestion fine des requêtes et des logs.

Performance, souplesse et scalabilité

Node.js excelle dans le traitement rapide des opérations d'I/O (lecture, écriture, streaming). Cela le rend idéal pour un projet comme BiblioTech, qui repose sur l'accès permanent à une grande quantité de contenus. Le découpage en modules REST permet également une évolution progressive du système : chaque brique peut être améliorée ou remplacée sans perturber l'ensemble.

Un choix cohérent avec l'ensemble du projet

L'adoption de Node.js + Express s'inscrit dans une logique d'unification technologique : une stack JavaScript complète, des outils partagés, un code homogène du client au serveur. Cela simplifie le développement, réduit les risques d'erreurs et facilite la montée en compétence sur le projet.

C'est un choix pensé non seulement pour aujourd'hui, mais aussi pour demain : une base saine, moderne, évolutive, qui pourra accueillir de nouvelles fonctionnalités sans devoir tout repenser.

3) Base de données : MariaDB & MongoDB

L'architecture de BiblioTech repose sur un système de gestion de données hybride, alliant la rigueur relationnelle de MariaDB à la souplesse documentaire de MongoDB. Ce couple permet d'équilibrer structuration, performance, et évolutivité.

Pourquoi une double base ?

Les besoins de la plateforme sont multiples :

- Des données critiques et interdépendantes (utilisateurs, favoris, commentaires...) qui nécessitent de fortes garanties d'intégrité.
- Des données secondaires, évolutives ou non standardisées (tags, logs dynamiques, métadonnées OCR...) dont la structure peut varier dans le temps.

Pour répondre à ces exigences sans complexifier inutilement la logique métier, j'ai opté pour une séparation claire des responsabilités :

- MariaDB gère la cohérence, les relations, la logique métier centrale.
- MongoDB prend en charge les contenus flexibles et adaptatifs, liés aux traitements automatisés ou aux enrichissements futurs.

Intégration dans l'architecture

Les deux bases fonctionnent en parallèle :

- Le back-end Node.js interagit avec les deux via des modules distincts,
- Les requêtes critiques sont priorisées sur MariaDB,
- MongoDB est utilisée en lecture/écriture ponctuelle pour les contenus dynamiques ou exploratoires.

Des interfaces d'accès claires ont été mises en place pour isoler les logiques :

- ORM via Sequelize pour MariaDB,
- Driver natif MongoDB (ou Mongoose) pour la base NoSQL.

Cela garantit la clarté du code, l'indépendance des couches, et la possibilité d'évoluer vers une architecture microservices si besoin.

Évolutivité et maintenance

Ce découpage offre plusieurs avantages :

- Évolutivité : MongoDB permet d'ajouter rapidement de nouveaux champs, types ou structures sans migration complexe.
- Performance : les opérations critiques sont optimisées dans MariaDB, pendant que les traitements lourds ou non urgents (OCR, tags générés, logs) sont déportés sur Mongo.
- Sécurité : chaque base a ses propres identifiants d'accès, avec droits restreints selon le type de requêtes.

Les Diagrammes des deux modèles de données sont disponibles en annexe.

4) Stockage & fichiers (ebooks PDF/EPUB)

La gestion des fichiers est un élément central de BiblioTech, car elle conditionne l'expérience de lecture et la sécurité globale de la plateforme. Elle concerne principalement les ebooks, qui peuvent être au format PDF ou EPUB.

Téléversement et organisation

Le téléversement des ebooks est ouvert à tous les utilisateurs enregistrés. Le processus est contrôlé :

- Validation du format de fichier (PDF ou EPUB uniquement),
- Limitation de la taille (pour éviter les abus),
- Attribution automatique de métadonnées de base (nom du fichier, date d'upload, uploader).

Les fichiers sont stockés dans un répertoire uploads/, qui est situé en dehors du dossier public de l'application. Il n'est accessible qu'à travers des routes d'API sécurisées par JWT.

Accès et lecture

- Les fichiers peuvent être lus en ligne via un lecteur intégré (PDF.js ou epub.js).
- Le téléchargement est également possible, selon les règles de la plateforme (ex. droits d'auteur, modération).
- Chaque lecture est associée à un utilisateur (s'il est connecté), ce qui permet de générer un historique, de reprendre une lecture en cours, ou de faire des recommandations futures.

Sécurité et gestion du contenu

Des contrôles sont mis en place pour sécuriser le stockage :

- Scan du type MIME pour éviter les fichiers déguisés,
- Attribution d'un nom de fichier unique (UUID) pour éviter les collisions ou les accès directs non autorisés,
- Journalisation des téléversements pour garantir la traçabilité.

Les fichiers ne sont jamais exposés directement via une URL publique. Ils transitent toujours par le serveur backend, ce qui permet de vérifier les autorisations d'accès.

Perspectives

Ce système est conçu pour évoluer :

- Intégration d'un moteur d'analyse de contenu (OCR, indexation par mots-clés) via une future couche IA,
- Compression automatique des fichiers trop lourds,
- Déclinaison audio via synthèse vocale.

Un extrait de route de téléversement est disponible en annexe.

5) Sécurité et authentification (JWT, Helmet)

La sécurité est un pilier stratégique de BiblioTech, aussi bien pour protéger les données personnelles que pour garantir la fiabilité du service sur le long terme. La plateforme a été pensée pour limiter au maximum les failles potentielles, tout en maintenant une architecture légère et moderne.

Authentification JWT

L'identification des utilisateurs repose sur des JSON Web Tokens (JWT) :

- Le token est généré à la connexion, signé avec une clé secrète,
- Il a une durée de validité limitée,
- Il est requis pour toutes les routes sensibles de l'API (profil, téléversement, gestion de contenus).

Aucun stockage de session côté serveur : le token est vérifié à chaque requête, garantissant une architecture stateless et facilement scalable.

Middleware de sécurité

Plusieurs middlewares ont été intégrés pour renforcer la sécurité :

- helmet : ajout automatique d'en-têtes HTTP (Content-Security-Policy, X-Frame-Options, etc.),
- rate-limit : limite le nombre de requêtes par IP sur les routes sensibles (anti-brute-force),
- express-validator et Joi : pour filtrer, nettoyer et valider les entrées utilisateur (contre XSS, injections...),
- cors : politique d'origine contrôlée.

Sécurisation du backend

Les routes sont toutes protégées par des middlewares :

- Vérification du JWT,
- Vérification du rôle utilisateur (utilisateur simple, modérateur, admin),
- Logging systématique des actions sensibles (modération, signalement, suppression...).

Les erreurs critiques sont enregistrées dans des fichiers de logs pour analyse et audit manuel si nécessaire.

Gestion des fichiers téléversés

Les ebooks sont également soumis à une double validation :

- Format autorisé (PDF ou EPUB),
- Limite de poids,
- Attribution d'un nom aléatoire pour éviter toute prédiction d'URL.

Sécurité des bases de données

- Accès restreint via mot de passe fort,
- Aucune donnée sensible stockée en clair (hash bcrypt pour les mots de passe),
- Séparation des privilèges entre MariaDB et MongoDB.

Sécurité côté front-end

Le front intègre lui aussi des protections :

- Données utilisateur affichées avec échappement des caractères spéciaux,
- Gestion centralisée des erreurs,
- Désactivation du clic droit sur certaines pages sensibles (lecture, profil),
- Messages explicites en cas d'erreur ou de tentative malveillante.

6) Accessibilité & SEO

L'accessibilité et le référencement sont deux piliers essentiels dans la conception de BiblioTech. La plateforme a été conçue pour être consultable par tous, y compris les personnes en situation de handicap, et pour rester visible et indexable malgré l'usage d'une architecture SPA (Single Page Application).

Accessibilité numérique (conformité WCAG)

Des efforts spécifiques ont été réalisés pour assurer la conformité avec les recommandations WCAG (Web Content Accessibility Guidelines) :

- Contrastes respectés : les combinaisons de couleurs (fonds / textes / boutons) ont été testées via des outils d'analyse pour assurer une lisibilité maximale.
- Navigation clavier complète : tous les éléments interactifs (menus, boutons, liens, formulaires) sont accessibles par tabulation, avec focus visuel.
- Balises ARIA : les rôles ARIA sont définis pour améliorer la compréhension de l'interface par les lecteurs d'écran (ex. aria-label, aria-expanded, role="navigation"...).
- Structure sémantique claire : usage rigoureux des titres hiérarchisés (<h1>, <h2>, etc.), des régions (<main>, <nav>, <aside>) et des formulaires bien étiquetés.

Un soin particulier a été porté à la lisibilité, au contraste typographique, et à la clarté des interfaces pour les personnes dyslexiques ou malvoyantes.

Référencement naturel (SEO)

Même si l'application est construite comme une SPA avec Vue.js, plusieurs stratégies ont été mises en place pour maintenir une bonne indexabilité :

- Meta-tags dynamiques : via vue-meta, chaque route principale (accueil, livre, catégorie, profil) dispose de balises <title>, <meta name="description">, etc.
- URLs propres et logiques : /livres/nom-du-livre, /categorie/accessibilite, etc. pour améliorer la compréhension par les robots de Google.
- Contenu structuré : balises <article>, <section>, <header>, pour une sémantique propre.

Une version statique des pages les plus importantes pourrait être générée si un besoin de SEO plus poussé se faisait sentir à l'avenir (via prerendering ou SSR).

7) Application mobile : Flutter (offline, synchro, UI réactive)

Pour offrir une accessibilité maximale, même en mobilité ou sans connexion stable, une application mobile BiblioTech a été développée avec Flutter, le framework multiplateforme de Google. Elle reprend l'essentiel des fonctionnalités du site, dans une interface fluide, intuitive et pensée pour les usages nomades.

Pourquoi Flutter ?

Flutter permet de :

- Développer une application unique pour Android et iOS,
- Compiler en natif pour d'excellentes performances,
- Bénéficier d'une UI personnalisable et cohérente avec la version web,
- Gérer facilement le stockage local, la synchronisation, et les animations fluides.

Fonctionnalités intégrées

L'application mobile propose :

- La consultation du catalogue de livres (avec filtres, recherches),
- L'accès à ses favoris et son historique de lecture,
- Le téléchargement local des livres pour une lecture hors-ligne,
- La lecture d'eBooks (PDF/EPUB) directement dans l'application,
- L'authentification sécurisée via JWT (comme sur le site),
- La synchronisation automatique des données à la reconnexion.

Accessibilité mobile

Tout comme la version web, l'interface mobile respecte des standards stricts d'accessibilité :

- Compatibilité avec les lecteurs d'écran (TalkBack, VoiceOver),
- Navigation à une main facilitée,
- Contrastes adaptés, textes redimensionnables, et composants tactiles suffisamment grands.

Performance et fluidité

Flutter permet une animation fluide, même sur des téléphones peu puissants :

- Les ressources lourdes sont chargées en différé,
- Les pages sont préchargées de manière intelligente,
- La navigation repose sur une logique de widgets hiérarchisés, avec un code maintenable.

Évolutivité et maintenance

Le code Flutter est découpé en widgets réutilisables. L'ajout de nouvelles fonctionnalités (comme la lecture vocale ou les modules d'alphabétisation) pourra se faire sans casser l'existant. Les mises à jour sont déployées directement sur les stores, sans intervention utilisateur.

Un aperçu visuel de l'application (UI Flutter) est disponible en annexe.

D. Cadre légal et responsabilités

Le projet BiblioTech, en tant que plateforme communautaire de partage de fichiers numériques, est soumis à plusieurs cadres réglementaires. Afin de garantir la légalité du service et la sécurité juridique de ses utilisateurs comme de son développeur, plusieurs mesures concrètes ont été mises en place.

1) Statut d'hébergeur de contenus

BiblioTech agit en tant que prestataire technique, conformément à l'article 6-I-2 de la LCEN (Loi pour la Confiance dans l'Économie Numérique). Ce statut, hérité de la directive européenne sur le commerce électronique, signifie que la plateforme n'est pas responsable a priori des contenus publiés par les utilisateurs, tant qu'elle respecte trois obligations :

- Neutralité : la plateforme ne modère pas de manière active ni éditorialise les contenus avant leur publication.
- Réactivité : un système de signalement clair est disponible sur chaque fiche livre. Toute demande de retrait fondée (contenu illicite, non autorisé, etc.) est traitée dans un délai raisonnable.
- Traçabilité : BiblioTech conserve les logs de connexion et d'historique de publication permettant d'identifier un utilisateur en cas de demande judiciaire.

En complément, BiblioTech s'inscrit dans le régime de "Safe Harbor", reconnu dans certains accords internationaux, qui protège les plateformes tant qu'elles agissent de bonne foi pour retirer les contenus litigieux. Un extrait représentatif du texte officiel est disponible en annexe.

2) Protection des données personnelles (RGPD)

BiblioTech est conçu dans une logique de privacy by design. Le traitement des données est limité, sécurisé et transparent :

- Données collectées : uniquement le strict nécessaire (e-mail, pseudo, mot de passe hashé), sans aucun pistage comportemental ni cookie marketing.
- Consentement explicite : lors de l'inscription, une case à cocher informe l'utilisateur des données traitées.
- Droits utilisateurs : chaque utilisateur peut accéder à ses données, les modifier ou en demander la suppression directement depuis son compte.
- Sécurité des données : les bases MariaDB et MongoDB sont sécurisées, cloisonnées, et les mots de passe sont hashés via bcrypt. Aucun mot de passe ou fichier utilisateur n'est stocké en clair.

Aucune donnée n'est revendue, transférée ou partagée avec un tiers. Aucun contenu publicitaire ou traceur externe n'est intégré au site.

3) Droit d'auteur et propriété intellectuelle

Les utilisateurs sont responsables des fichiers qu'ils téléversent sur la plateforme. BiblioTech informe clairement que seuls les fichiers :

- relevant du domaine public,
- ou placés sous une licence libre (Creative Commons, GNU FDL, etc.)

sont autorisés au partage.

Les œuvres commerciales, piratées ou sous copyright sans autorisation sont interdites. En cas de manquement, le contenu est supprimé après signalement, et l'utilisateur peut voir son compte suspendu temporairement.

BiblioTech ne revendique aucune propriété sur les œuvres partagées. Les contributeurs conservent l'intégralité de leurs droits sur les fichiers qu'ils publient.

4) Mentions légales et Conditions Générales d'Utilisation

Un document de CGU (Conditions Générales d'Utilisation) est disponible sur la plateforme. Il définit :

- les engagements des utilisateurs,
- les limites de responsabilité de BiblioTech,
- les sanctions prévues en cas de non-respect (exclusion, retrait de contenu),
- les conditions de suppression ou d'anonymisation des comptes.

Des mentions légales complètes (éditeur du site, hébergeur, contact) sont aussi accessibles dans le pied de page, comme le prévoit la loi française.

VI. Planification et gestion du projet

1. Méthodologie de gestion

En tant que développeur unique sur le projet BiblioTech, j'ai opté pour une approche agile, simplifiée mais structurée. Mon objectif était de garder une vision claire des priorités tout en conservant la flexibilité nécessaire pour adapter le développement au fil des découvertes techniques, des imprévus et des tests utilisateurs.

Approche Kanban

J'ai utilisé la méthode Kanban, plus souple que Scrum, et mieux adaptée à un projet individuel. Elle m'a permis de :

- visualiser facilement l'avancement via des colonnes (À faire / En cours / Terminé),
- prioriser les tâches semaine par semaine,
- éviter la surcharge mentale en me concentrant sur quelques tâches clés à la fois,
- gérer les évolutions du projet sans perdre de vue les fondations.

L'outil principal utilisé pour le suivi des tâches est Trello, avec des étiquettes pour distinguer les catégories (backend, frontend, mobile, tests, contenu...).

Répartition par lots fonctionnels

Le développement a été organisé en lots successifs correspondant à des blocs fonctionnels cohérents :

- Lot 1 : Authentification et gestion utilisateur
- Lot 2 : Lecture et gestion des ebooks
- Lot 3 : Fonctionnalités sociales (favoris, signalements)
- Lot 4 : Déploiement initial et tests
- Lot 5 : Application mobile Flutter
- Lot 6 : Finitions UX / accessibilité / documentation

Chaque lot a fait l'objet d'une planification préalable avec des objectifs clairs, des dates cibles indicatives, et une phase de tests manuels à sa conclusion.

Adaptation continue

La nature même du projet – à la fois technique, social, et éthique – a nécessité plusieurs ajustements en cours de route :

- ajout de fonctionnalités en réponse aux retours de tests utilisateurs (ex. : bouton « lecture hors ligne », filtres par langue...),
- réorganisation des données entre MariaDB et MongoDB après itérations,
- évolution des priorités (ex. : avancée de l'application Flutter avant la finalisation de certains détails UI web).

Cette capacité à adapter les priorités en fonction du concret m'a permis de maintenir un rythme soutenu mais réaliste, sans sacrifier la qualité du code ni la cohérence du projet.

2. Chronologie du projet

Le projet BiblioTech a été amorcé dès la fin de l'année 2024 avec une phase de conception et de planification rigoureuse. Le développement actif s'est ensuite déroulé de manière progressive entre janvier et août 2025, selon un enchaînement structuré des étapes : modélisation, développement, tests, puis finalisation du dossier.

| Période | Étapes clés |
|------------------------------------|--|
| De Novembre à Décembre 2024 | <ul style="list-style-type: none"> - Réflexion initiale, rédaction des premières notes conceptuelles - Élaboration du cahier des charges, choix de la stack |
| De Janvier à Mars 2025 | <ul style="list-style-type: none"> - Structuration du projet, architecture technique - Modélisation des bases de données (MariaDB & MongoDB) - Mise en place de l'environnement |
| De Avril à Mai 2025 | <ul style="list-style-type: none"> - Développement de l'API Node.js avec Express (authentification, gestion utilisateurs, sécurité JWT) - Début de l'interface front-end avec Vue.js |
| De Juin à Juillet 2025 | <ul style="list-style-type: none"> - Développement des fonctionnalités : lecture, favoris, signalements, historique, commentaires - Tests fonctionnels et corrections UX/accessibilité |
| En Août 2025 | <ul style="list-style-type: none"> - Développement de l'application mobile Flutter - Synchronisation avec l'API, lecture hors ligne - Finalisation du dossier & documentation |

3. Organisation du code (structure du projet)

Le projet BiblioTech est structuré de manière modulaire, pour garantir une lisibilité maximale, une maintenance facilitée et une montée en complexité progressive. Chaque couche (back-end, front-end, mobile) respecte une séparation claire des responsabilités, avec une architecture pensée pour l'évolution.

Back-end (Node.js + Express)

Le serveur Express est organisé selon une architecture MVC étendue :

```
/backend
|
├─ /controllers      → Logique métier (auth, livres, utilisateurs...)
├─ /routes            → Déclaration des routes de l'API REST
├─ /models            → Schémas de données (MariaDB & MongoDB)
├─ /middlewares       → Vérifications JWT, validations, gestion des rôles
├─ /services          → Fonctions réutilisables, traitement fichiers, logs
├─ /config            → Connexions aux bases de données, variables .env
├─ /uploads           → Stockage sécurisé des fichiers (ebooks)
├─ /tests             → Fichiers Jest (auth, eBooks, sécurité...)
└─ server.js         → Point d'entrée du serveur
```

Front-end (Vue.js)

Le front suit une structure en composants Vue réutilisables, avec une séparation des vues, de la logique, et du style :

```
/frontend
|
├─ /src
|   │
|   ├─ /components    → Composants UI (header, boutons, cards...)
|   │
|   ├─ /views          → Pages (Accueil, Profil, Catalogue, Connexion...)
|   │
|   ├─ /router         → Fichier de routing (vue-router)
|   │
|   ├─ /store          → Gestion d'état globale (pinia)
|   │
|   ├─ /assets         → Feuilles de styles, images, icônes
|   │
|   ├─ /utils          → Fonctions communes (formatage, validation...)
|   │
|   └─ App.vue        → Composant racine
|
└─ vite.config.js     → Configuration du bundler Vite
```

Application mobile (Flutter)

L'application mobile est organisée en widgets et services, avec une logique claire :

```
/mobile
|
├─ /lib
|   │
|   ├─ /screens        → Pages principales (bibliothèque, profil, lecture...)
|   │
|   ├─ /widgets        → Composants visuels réutilisables
|   │
|   ├─ /services        → Appels API, gestion locale
|   │
|   ├─ /models         → Objets de données (livre, utilisateur...)
|   │
|   └─ main.dart       → Point d'entrée de l'application
|
└─ /assets             → Icônes, polices, images
```

4. Versionning du projet (Git)

Le versionnage du projet a été assuré via Git, avec un dépôt principal hébergé sur GitHub. Ce choix s'inscrit dans une démarche de rigueur, de traçabilité, et de travail incrémental tout au long du développement.

Organisation du dépôt

Le dépôt est structuré en branches claires et bien identifiées :

- main : branche de production, stable, uniquement mise à jour après validation.
- dev : branche de développement principale, utilisée pour intégrer les nouvelles fonctionnalités.
- branches spécifiques : chaque fonctionnalité ou correction majeure est développée sur une branche dédiée (feature/lecture, fix/auth, test/api-books, etc.), avant d'être mergée dans dev via pull request.

Bonnes pratiques appliquées

- Commits réguliers, explicites et datés (feat: ajout système de favoris, fix: bug token JWT expiré, etc.).
- Utilisation des issues GitHub pour documenter les tâches, suivre les bugs, et centraliser les retours de tests.
- Merge systématique via pull request pour assurer relecture et tests avant intégration.
- Gitignore configuré pour exclure les fichiers sensibles et temporaires (.env, /node_modules, /dist, /uploads, etc.).

Sécurité du versionning

Aucune donnée sensible n'est stockée dans le dépôt (mots de passe, clés d'API...). Les variables critiques sont placées dans un fichier .env, ignoré par Git et géré localement sur chaque environnement.

VII. Scénarios d'utilisation

*BiblioTech a été conçu pour répondre à des besoins concrets et variés, en mettant l'accent sur la simplicité, l'accessibilité et la fluidité d'usage. Cette section présente plusieurs scénarios d'usage illustrant les fonctionnalités clés de la plateforme, tant côté web que mobile.

1. Cas d'usage

Voici quelques exemples de cas d'utilisation typiques qui structurent l'architecture fonctionnelle du projet :

a) Consultation d'un livre librement accessible

Un internaute non inscrit arrive sur la plateforme. Il consulte directement le catalogue de livres gratuits. Il clique sur un titre, accède à une fiche de lecture, puis lance la lecture en ligne sans avoir à s'inscrire.

b) Création de compte et personnalisation

Un nouvel utilisateur s'inscrit avec un email et un mot de passe. Il peut personnaliser son interface, ajouter des livres en favoris, activer le mode sombre, ou configurer ses préférences de lecture.

c) Téléversement de contenu par un contributeur

Un utilisateur ayant un rôle « contributeur » accède à la page d'envoi. Il téléverse un fichier EPUB ou PDF, remplit les métadonnées (titre, auteur, langue...), et soumet son livre à la modération.

d) Lecture hors-ligne via l'application mobile

Un utilisateur connecté via l'app Flutter navigue dans ses favoris, télécharge un livre pour plus tard, et continue la lecture dans le métro sans réseau. La synchronisation reprend dès qu'il se reconnecte.

e) Signalement de contenu inapproprié

Un lecteur repère un contenu problématique (propos haineux, fichier corrompu...). Il utilise le bouton « signaler » sur la fiche livre. Un modérateur reçoit l'alerte et agit en conséquence.

2. Parcours utilisateur (web & mobile)

L'expérience utilisateur sur BiblioTech a été pensée pour être fluide, cohérente et adaptée au support utilisé. Les deux interfaces (web et mobile) partagent les mêmes bases fonctionnelles, tout en s'ajustant aux contraintes d'usage spécifiques à chaque environnement.

a) Parcours utilisateur Web

1. Page d'accueil
2. L'utilisateur accède au site via n'importe quel navigateur. Il est accueilli par une interface claire, avec un bandeau de recherche, des catégories mises en avant, et une sélection de livres populaires.
3. Navigation et découverte
4. Il peut filtrer les livres par langue, genre, popularité ou nouveauté. Chaque vignette de livre mène à une fiche détaillée.
5. Lecture directe ou téléchargement
6. Depuis la fiche, il peut lire le livre en ligne via un lecteur intégré (PDF ou EPUB), ou bien télécharger le fichier si son statut le permet.
7. Inscription et gestion du compte
8. En choisissant de s'inscrire, il peut suivre son historique de lecture, créer des favoris, proposer des ajouts de contenu ou signaler des problèmes.
9. Utilisation avancée
10. Un contributeur peut accéder à l'espace d'envoi, un modérateur à l'interface de gestion des signalements. L'administrateur dispose d'un tableau de bord de suivi global.

b) Parcours utilisateur Mobile (Flutter)

1. Connexion rapide
2. L'utilisateur ouvre l'application mobile. Une session persistante lui évite de devoir se reconnecter à chaque lancement. Il accède directement à son espace personnel ou à la page d'accueil.
3. Recherche et consultation
4. L'expérience mobile reprend la structure du site web, mais adaptée aux gestes tactiles : menus repliables, listes déroulantes, icônes explicites.
5. Téléchargement hors ligne
6. Il sélectionne un livre et l'ajoute à sa bibliothèque locale. Le fichier est stocké dans le cache de l'application pour une lecture sans réseau.
7. Synchronisation des lectures
8. À chaque reconnexion, ses lectures, favoris et commentaires sont synchronisés avec son compte en ligne via l'API REST.
9. Accessibilité mobile
10. L'application est conçue pour une lecture aisée, avec un mode nuit, une taille de police ajustable, et une compatibilité avec les lecteurs d'écran Android/iOS.

VIII. API REST : Spécification technique

L'API REST de BiblioTech constitue l'épine dorsale de la plateforme. Elle permet aux différentes interfaces (web, mobile) d'accéder aux données, de gérer les comptes, d'interagir avec les livres et de maintenir une cohérence entre tous les usages. Elle est développée en Node.js avec Express, dans une architecture modulaire, sécurisée et bien documentée.

1. Authentification et gestion des utilisateurs

L'API gère l'ensemble du cycle de vie utilisateur :

- **Inscription** : création de compte avec validation des champs (nom, email, mot de passe), hash sécurisé (bcrypt), envoi optionnel de confirmation.
- **Connexion / Déconnexion** : vérification des identifiants, génération d'un JWT avec durée de vie limitée.
- **Mise à jour du profil** : modification du pseudo ou de l'email, changement de mot de passe avec vérification.
- **Suppression du compte** : suppression logique ou physique, selon les préférences utilisateur.
- **Rôles et autorisations** :
 - Utilisateur simple : lecture, favoris, historique.
 - Contributeur : ajout de livres.
 - Modérateur : gestion des signalements.
 - Administrateur : accès global, stats, logs, droits étendus.

Middleware de vérification des rôles intégré à chaque route sensible.

2. Gestion des livres numériques

L'API permet à un utilisateur authentifié (contributeur ou plus) de gérer l'ensemble du cycle de vie d'un livre numérique :

a) Ajout de livre

- Upload multipart via un endpoint sécurisé.
- Fichiers acceptés : .pdf, .epub, avec contrôle du type MIME et taille maximale (50 Mo par défaut).
- Métadonnées demandées : titre, auteur, langue, description, catégories, option de visibilité.
- Génération automatique d'une miniature (côté serveur).
- Enregistrement en base MariaDB (fiche) et MongoDB (métadonnées).

b) Consultation

- Endpoint public de récupération :
 - par ID, par filtre (auteur, titre, catégorie, date, popularité),
 - paginé pour optimisation des performances.
- Contenus liés récupérés (commentaires, favoris, uploads récents...).

c) Modification / suppression

- Autorisation uniquement pour l'uploader d'origine ou un modérateur.
- Mise à jour partielle possible (titre, description...).
- Suppression logique (archivage) par défaut, suppression physique possible côté admin.

d) Téléchargement & lecture

- Téléchargement déclenché depuis le front avec vérification du JWT.
- Lien temporaire généré côté serveur (prévention hotlinking).
- Lecture possible via iframe ou lecteur intégré (webreader EPUB/pdf.js).

3. Favoris et historique

Pour offrir une expérience personnalisée et permettre aux utilisateurs de reprendre facilement leur lecture ou retrouver leurs ouvrages préférés, l'API expose plusieurs endpoints dédiés aux favoris et à l'historique.

a) Favoris

- **Ajout d'un favori** : L'utilisateur peut ajouter un livre à ses favoris via un endpoint sécurisé (POST /favorites/:bookId), nécessitant un JWT valide. Le favori est enregistré dans une table relationnelle liant l'utilisateur et le livre.
- **Suppression** : Suppression simple via DELETE /favorites/:bookId.
- **Récupération des favoris** : Endpoint GET /favorites/ retournant la liste paginée des livres favoris de l'utilisateur connecté.

b) Historique de lecture

- **Enregistrement automatique** : Chaque consultation ou lecture déclenche un enregistrement dans l'historique (timestamp, ID utilisateur, ID livre, page/position lue si disponible).
- **Récupération** : Endpoint GET /history/ pour afficher l'historique, trié par date décroissante.
- **Suppression** : L'utilisateur peut supprimer tout ou partie de son historique.
- **Exploitation côté front** : Permet de proposer une section « reprendre la lecture », « récemment consultés », ou encore de personnaliser les suggestions.

4. Commentaires sur les livres

Pour favoriser l'interaction entre les utilisateurs et enrichir le catalogue d'avis et de retours, un système de commentaires est mis en place. Il est simple, modéré, et lié à chaque fiche de livre.

a) Ajout de commentaire

- Route : POST /books/:bookId/comments
- Nécessite un utilisateur connecté (JWT).
- Le corps de la requête contient un champ content avec validation stricte (longueur, caractères interdits...).
- Modération :
- Tous les commentaires sont horodatés et associés à l'utilisateur. Une logique de signalement (cf. point 5) permet de les retirer en cas d'abus.

b) Récupération des commentaires

- Route : GET /books/:bookId/comments
- Affiche la liste paginée des commentaires liés à un livre.
- Les données retournées incluent le nom de l'auteur, la date, et le contenu.

c) Suppression d'un commentaire

- Route : DELETE /comments/:id
- Accessible uniquement :
 - à l'auteur du commentaire,
 - à un administrateur ou modérateur.

5) Signalement de contenu

Pour garantir un environnement sûr, respectueux et conforme au cadre légal, BiblioTech propose un système de signalement. Celui-ci permet aux utilisateurs de notifier un contenu problématique (livre, commentaire, utilisateur), afin qu'il soit examiné et modéré si nécessaire.

a) Types de contenu signalables

- Livres téléversés
- Commentaires postés
- Profils utilisateurs

Chaque type a une route dédiée pour permettre un traitement clair côté API.

b) Création d'un signalement

- Route : POST /reports
- Corps de la requête :

```
{
  "targetType": "book" | "comment" | "user",
  "targetId": "<id du contenu visé>",
  "reason": "<raison sélectionnée ou texte libre>"
}
```

- Seuls les utilisateurs authentifiés peuvent envoyer un signalement. Un middleware vérifie que le contenu visé existe et qu'un même utilisateur ne signale pas plusieurs fois le même élément inutilement.

c) Consultation des signalements

- Route : GET /admin/reports
- Réservée aux comptes avec un rôle admin ou moderator.
- Les signalements sont listés avec :
 - type et ID du contenu,
 - nom de l'utilisateur signalant,
 - date et raison,
 - statut (en attente, traité, rejeté...).

d) Traitement d'un signalement

- Route : PATCH /admin/reports/:id
- Permet de :
 - marquer un signalement comme traité,
 - éventuellement supprimer le contenu associé (commentaire ou livre),
 - envoyer une alerte ou suspendre un compte si nécessaire.

Ce système assure à la fois la responsabilisation des utilisateurs, une modération réactive, et une traçabilité complète pour répondre aux exigences légales du statut d'hébergeur.

6. Statistiques

BiblioTech propose un module simple de consultation de statistiques, destiné aux administrateurs/modérateurs. Ces données permettent de suivre l'usage de la plateforme, détecter les dérives éventuelles, et orienter les améliorations futures.

a) Données agrégées générales

- Route : GET /admin/stats/global
- Accès : restreint aux administrateurs
- Réponse :

```
{
  "totalUsers": 482,
  "totalBooks": 1392,
  "totalDownloads": 7521,
  "totalReports": 34,
  "totalComments": 912,
  "activeUsersLast30Days": 187
}
```

- Ces chiffres sont utilisés dans le tableau de bord admin pour fournir une vue d'ensemble rapide de la plateforme.

b) Statistiques d'usage

- Route : GET /admin/stats/usage
- Permet de consulter :
 - Le nombre de lectures/jour,
 - Le nombre de téléchargements/jour,
 - Les livres les plus consultés.
- Exemple de réponse :

```
{
  "topBooks": [
    { "title": "1984", "views": 381 },
    { "title": "Le Petit Prince", "views": 342 },
    ...
  ],
  "dailyDownloads": [
    { "date": "2025-06-10", "count": 53 },
    { "date": "2025-06-11", "count": 47 },
    ...
  ]
}
```

c) Suivi des signalements

- Route : GET /admin/stats/reports
Fournit un récapitulatif des signalements par type de contenu :

```
{
  "bookReports": 21,
  "commentReports": 8,
  "userReports": 5
}
```

Permet également d'identifier les contenus les plus fréquemment signalés.

IX. Tests et assurance qualité

1. Stratégie de test

Pour garantir la fiabilité de BiblioTech et éviter les régressions au fil des évolutions, une stratégie de test a été mise en place dès les premières étapes du développement. Celle-ci repose sur trois piliers :

- **Tests unitaires** : vérification des fonctions critiques du back-end (authentification, logique de validation, manipulation de base de données).
- **Tests fonctionnels manuels** : parcours utilisateur complets testés à la main pour chaque version stable.
- **Tests de non-régression ponctuels** : relancés après toute modification majeure ou refonte structurelle.

La couverture porte principalement sur les modules sensibles (auth, livres, signalements, téléversements), en priorisant les cas limites et les erreurs courantes.

2. Résultats des tests unitaires et fonctionnels


a) Tests unitaires

Rédigés avec Jest, ils ciblent :

- Les routes d'authentification (/auth/login, /auth/register)
- La validation des entrées utilisateur
- Les opérations de lecture/écriture avec MariaDB et MongoDB
- Le middleware de protection (verifyToken, checkRoles)
- La logique de signalement

Les dépendances sont mockées pour isoler chaque bloc de logique métier. Tous les tests sont regroupés dans un dossier tests versionné dans Git.

Exemple de résultat :

- Total de tests : 84
- Réussis : 84 
- Temps moyen d'exécution : ~1,4s

b) Tests fonctionnels

Des scénarios utilisateur types ont été simulés sur différents navigateurs (Chrome, Firefox, Edge) et formats (desktop, mobile) :

- Connexion / inscription
- Téléversement d'un eBook
- Lecture en ligne
- Ajout / suppression de favoris
- Signalement d'un livre
- Consultation sur mobile via l'application Flutter

c) Suivi et documentation

- Tous les tests sont documentés dans une section dédiée du dépôt Git.
- Chaque test indique :
 - Son objectif
 - Les entrées attendues
 - Les cas d'erreur gérés
 - Le résultat attendu
- En cas de bug ou de régression, une fiche est ajoutée à l'historique de correction.

X. Déploiement et infrastructure

1. Plan de déploiement

Le déploiement de BiblioTech a été pensé pour être simple, reproductible et sécurisé, même sans infrastructure DevOps complexe. La plateforme est actuellement en production via un hébergement mutualisé chez OVH, couplé à un cluster distant MongoDB Atlas pour la base NoSQL.

Étapes de mise en production :

- Compilation du front Vue.js (npm run build) → répertoire dist/
- Déploiement du back-end Node.js/Express via FTP sécurisé
- Configuration des variables d'environnement dans un fichier .env
- Lancement de l'API avec pm2 (quand supporté par l'hébergement)
- Redirection du front via le serveur Express avec un catch-all (*)
- Test de connexion base de données + routage API avant publication

Sécurité lors du déploiement :

- Serveur en HTTPS via certificat Let's Encrypt fourni par OVH
- Accès FTP sécurisé + séparation du dossier de production
- Fichiers .env et dossiers critiques exclus du dépôt Git

Documentation associée :

Un README de production accompagne le dépôt :

- instructions de déploiement pas-à-pas
- scripts disponibles (npm start, npm run test, pm2 restart)
- procédures de mise à jour
- plan de rollback en cas de panne

2. Infrastructure

Hébergement

- Serveur : OVH mutualisé (support Linux + Node.js)
- Base relationnelle : MariaDB (gérée par OVH)
- Base NoSQL : MongoDB Atlas (cloud externe, cluster gratuit)
- Stockage fichiers : /uploads avec contrôle JWT
- Nom de domaine : configuré via OVH
- SSL : activé avec Let's Encrypt

Évolutivité possible :

L'infrastructure actuelle, si elle atteint ses limites, peut évoluer vers :

- Un VPS OVH pour plus de puissance et d'autonomie
- Un hébergement du front sur Netlify ou Vercel
- Une mise en place de CI/CD via GitHub Actions
- Un reverse proxy type NGINX pour une architecture plus solide
- Un découplage complet Front/Back si nécessaire (architecture JAMstack)

Surveillance & logs

- Logs applicatifs internes côté back-end
- Échecs critiques (auth, requêtes suspectes) consignés dans MongoDB
- Alertes à configurer pour les crashes fréquents ou anomalies réseau (prévu via uptime monitor tiers)

XI. Documentation et support

1. Documentation technique

BiblioTech est accompagné d'une documentation complète, destinée à faciliter la maintenance, la reprise du projet ou la montée en compétence d'un nouveau développeur. Cette documentation est structurée en plusieurs volets :

a) Documentation du code (back-end & front-end)

- Structure des dossiers (API, composants Vue, services, etc.)
- Description des routes API REST (méthodes, middlewares associés, formats de requête/réponse)
- Comportement des middlewares (authentification JWT, validation, gestion des erreurs)
- Schéma des modèles MongoDB et des tables MariaDB
- Explication des principaux composants Vue.js : logique, props, lifecycle

b) Environnement & déploiement

- Fichier README.md expliquant comment :
 - Installer les dépendances (npm install)
 - Démarrer l'environnement de développement (npm run dev)
 - Compiler le front (npm run build)
 - Lancer le back (npm start, pm2 pour la prod)
 - Recréer les bases (exports SQL/Mongo)
 - Modifier les variables d'environnement

c) Tests

- Cas de test documentés (objectifs, entrées/sorties, erreurs gérées)
- Instructions d'exécution des tests (commandes Jest)
- Captures de résultats (succès/échec) dans un dossier dédié

d) Données d'exemple

- Comptes de démonstration (utilisateur, modérateur)
- Jeux de données de livres pour tester le catalogue
- Requetes de test (Postman ou équivalent) disponibles

2. Support utilisateur

Pour les utilisateurs finaux, un système de support a été envisagé, bien qu'allégé à ce stade :

a) Aide intégrée dans l'application

- Pages d'aide accessibles depuis le menu (FAQ, tutoriels)
- Infobulles contextuelles dans les interfaces
- Messages d'erreur explicites et orientés (plutôt que techniques)

b) Signalement de problème

- Formulaire de contact/feedback intégré (anonyme ou identifié)
- Signalement possible de bugs ou contenus inappropriés (via bouton sur chaque livre)
- Possibilité d'envoyer un message d'assistance à l'administrateur

c) Suivi & réponse

- Les signalements sont enregistrés côté serveur
- Les administrateurs/modérateurs peuvent consulter, répondre ou archiver les tickets
- Les logs permettent de retracer les incidents critiques

XII. Évaluation et perspectives

1. Évaluation du projet

BiblioTech est un projet ambitieux, né d'une volonté simple : rendre la lecture accessible, libre, et moderne. Le résultat est une plateforme complète, fonctionnelle, et cohérente, à la fois techniquement solide et humainement engagée.

Objectifs atteints

- ☒ Une interface web fluide, accessible et responsive
- ☒ Un back-end sécurisé, modulaire, documenté
- ☒ Une base de données hybride performante (MariaDB + MongoDB)
- ☒ Une API REST fiable et synchronisée avec une application mobile Flutter
- ☒ Des fonctionnalités complètes : lecture, téléchargement, favoris, signalement, etc.
- ☒ Des tests unitaires et une validation fonctionnelle systématique
- ☒ Une documentation claire, un système de déploiement maîtrisé

Difficultés rencontrées

- La gestion d'une stack complète (front, back, mobile, bases de données) en solo a nécessité une rigueur importante dans l'architecture
- La mise en place de fonctionnalités avancées (upload sécurisé, streaming, authentification JWT) a demandé des recherches et itérations poussées

Malgré cela, le projet a su conserver sa cohérence et rester fidèle à ses objectifs initiaux.

2. Pistes d'amélioration et évolutions futures

Plusieurs évolutions sont envisagées pour enrichir BiblioTech et renforcer encore son utilité sociale et technique :

a) Génération d'eBooks via IA

Développement d'un module capable de scanner des pages de livres papier (OCR), d'en extraire le texte, et de générer un eBook structuré (PDF/EPUB). Cela permettrait la numérisation simplifiée de bibliothèques personnelles.

b) Lecture vocale intégrée

Ajout d'un système de lecture audio (Text-to-Speech) pour chaque livre, avec voix naturelle, vitesse réglable, et support multilingue. Ce service serait utile aux personnes malvoyantes, dyslexiques, ou peu à l'aise avec la lecture.

c) Programme d'alphabétisation numérique

Mise en place d'un espace d'apprentissage intégré, avec :

- Histoires lues interactives
- Exercices simples (mots à relier, phrases à compléter...)
- Quiz de compréhension
- Objectif : accompagner des publics en difficulté vers une autonomie de lecture.

d) Portabilité avancée

- PWA pour lecture sans installation
- Hébergement multi-plateformes (front sur Vercel/Netlify, back sur VPS)
- API ouverte à d'autres projets culturels ou associatifs

Le projet est donc une base solide, prête à évoluer. Il ne s'agit pas d'un produit figé, mais d'un socle libre et vivant, pensé pour grandir au fil des besoins, des idées et des utilisateurs.

XIII. Annexe