ECOLE TECHNIQUE
ECOLE DES MÉTIERS LAUSANNE

« P_WEB 295 »

LOVBOOKS – Passion Lecture

Réaliser le backend d'une application permettant de partager sa passion pour la lecture

19 FEVRIER 2024

ADRIAN TOLEDO & MATHIS BOTTEUA
Enseignant : ANTOINE MVENG

Table des matières

1. Introduction

Le projet P_Web_295 est destiné à créer le backend d'un site web pour l'enregistrement et la recherche de livres par les utilisateurs.

Ce site, appelé LovBooks, offre un service de catalogue de livres géré par les utilisateurs. Sur cette page, vous trouverez une liste de livres, avec leurs informations, les évaluations et les commentaires faits par les utilisateurs enregistrés.

Pour la construction du backend du site web, une API REST a été développée afin de gérer les informations de la base de données et les utilisateurs avec leur système d'authentification.

Sur une période de 24 périodes, il est prévu d'obtenir un code API complet qui permette d'introduire des routes et d'accéder à la base de données pour utiliser ou modifier les données.

A travers ce rapport, nous allons introduire les notions appliquées dans ce projet qui ont été apprises dans le module Web_295. Ce module dédié au développement web backend, nous offre les nations de l'utilisation et du codage pour les services d'une API REST.

2. Analyse

2.1. Planification

La planification de ce projet a été réalisée à l'aide de la plateforme Trello. Grâce à cette plateforme, nous avons divisé le développement du projet en suivant le modèle Kanban pour avoir une liste des tâches à faire (TO DO), une liste des tâches en cours (In Progress) et enfin une liste des tâches terminées (DONE).

Suivant ce modèle, nous avons divisé le projet en 5 sections principales :

- Planification et division du travail

- L'implémentation de GitHub pour le travail coopératif.

- Réalisation du rapport de projet

- Base de données : qui comprend le développement des modèles MCD, MLD, MPD et la connexion de l'API à un serveur.

- API REST, cette section est divisée en petites tâches de développement de l'API telles que : installation des extensions, création des routes, validation des données, recherche, système d'authentification, documentation Swagger et Test Insomnia.

2.2. Base de données

La base de données est composée de 7 tables, chacune est spécifique à une autre, cela permet d'assurer une structure cohérente. L'objectif est de simplifier l'utilisation et la compréhension de la base de données. La table principale « t_book » est liée avec les autres et permet d'obtenir n'importe quelle donnée d'une autre table.

Les tables "t_Publisher" et "t_Author" ont été isolées, car un auteur n'est pas toujours associé à un livre, et vice versa un éditeur peut ne pas avoir de livre lié. Cette séparation permet une représentation plus précise des relations entre les champs.

La table "t_Category" permet d'assurer une cohérence entre les livres, cela permet aussi à une catégorie de regrouper plusieurs livres. Cela simplifie la gestion des informations liées aux catégories et à leur association avec les livres qui leurs correspondent.

La séparation des tables "t_Comment" et "t_Assessment" permet de faire un commentaire sans nécessairement effectuer une évaluation, et vice versa. Cela offre de l'agilité dans la collecte des retours des utilisateurs.

La table "t_Customer", permet de garder la confidentialité de ses informations, ce qui évite toute interactions directes dans d'autres tables et renforce la sécurité des données.

Les relations entre les tables "t_Book", "t_Comment" et "t_Assessment" permettent d'obtenir des informations détaillées sur le moment des publications d'évaluations, des commentaires ou des livres. Cette liaison facilite la visualisation des événements des livres et des utilisateurs.

En résumé, la base de données à une structure cohérente et simple, elle prend en compte les champs de chaque table et les relations entre elles. Ce qui permet la compréhension et la visualisons de tous les événement des tables.

Ci-dessous les diagrammes de notre base de données en MLD et MCD

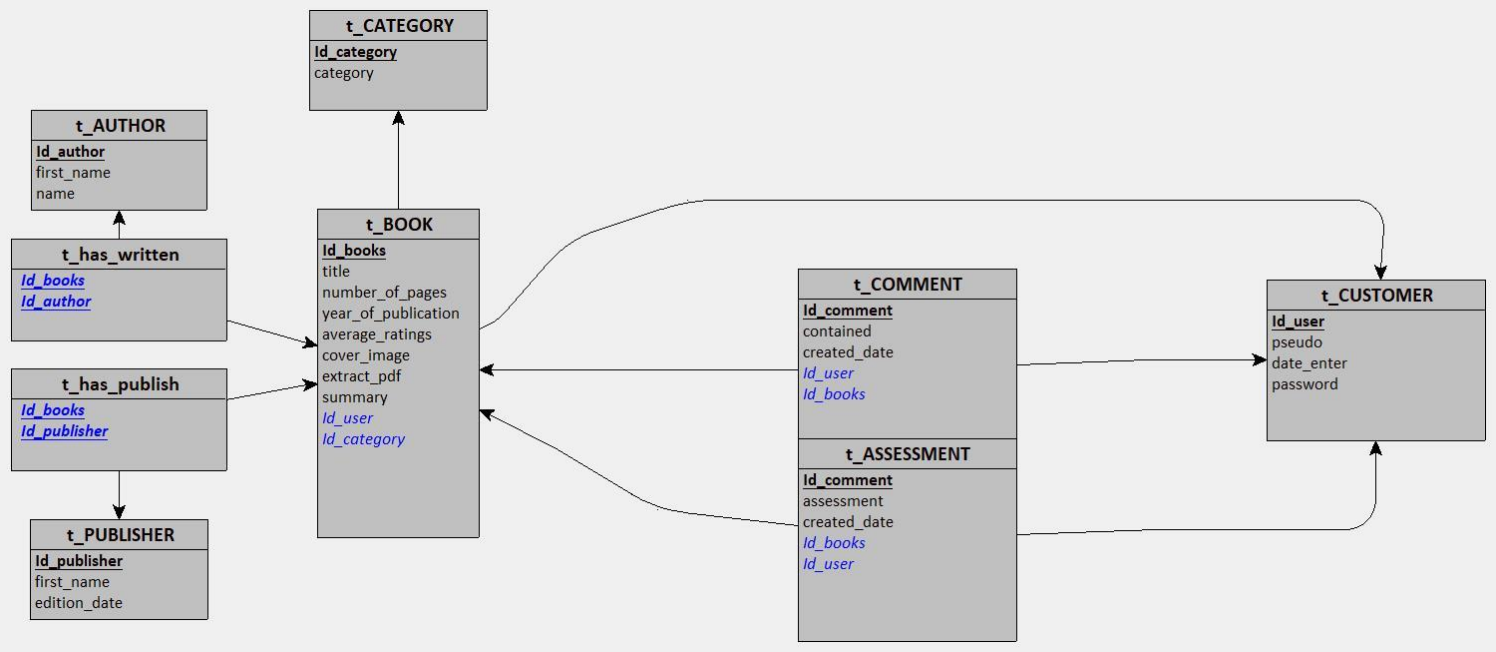

Figure 1 MLD_LovBooks_DB

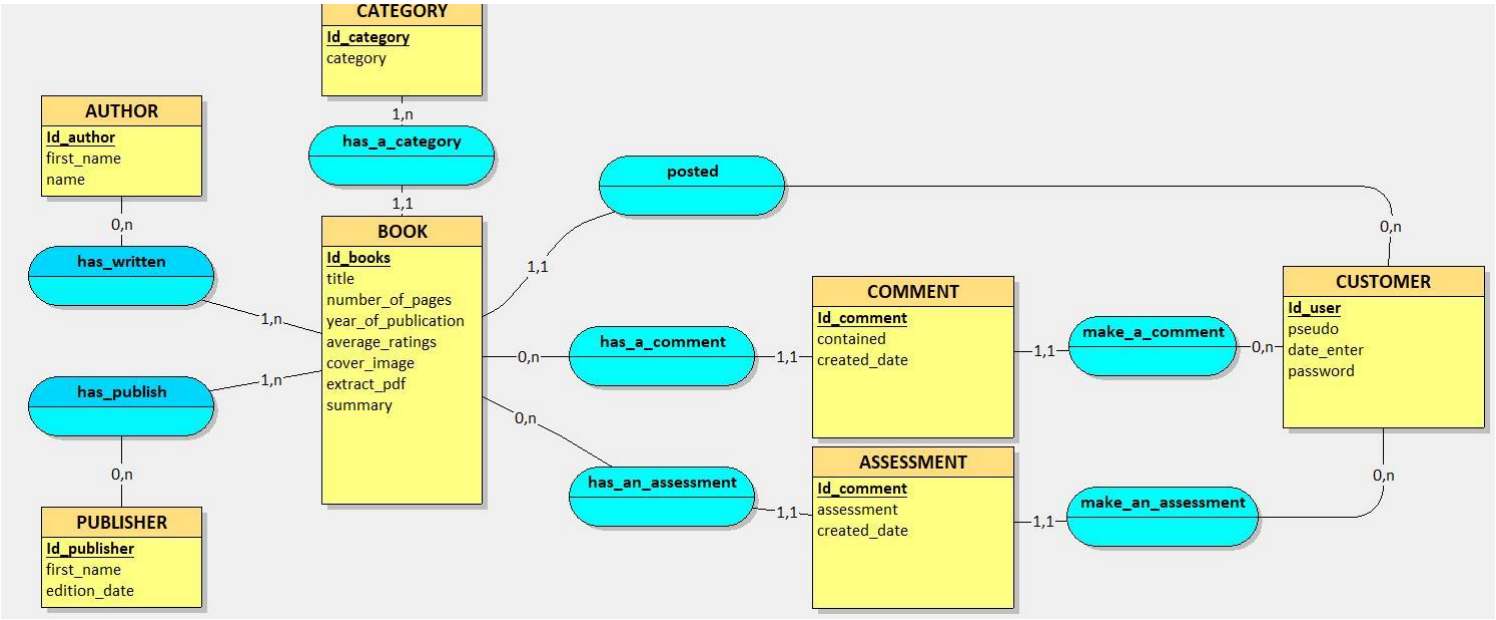

Figure 2 MCD_LovBooks_DB

2.3. API REST

LovBooks gère une grande quantité de données et l'utilisation d'une API nous permet de relier facilement nos données sur les livres et les utilisateurs à nos pages web.

Pour le développement de cette API, nous avons créé un code qui structure les tables de la base de données (Auteur, Livre, Catégorie, Commentaires, Publisher, Utilisateurs et Notes), gère les routes pour l'interaction avec la base de données, un service d'authentification et une documentation du code en Swagger.

2.4. Structure du code

L'api de LovBooks est divisée en 5 branches principales :

- Le projet est réalisé à partir de l'utilisation de node.js et des fichiers package.json. Un fichier pour la documentation en Swagger, et l'ensemble des routes de la page web et du serveur sont gérées par le script principal app.mjs.

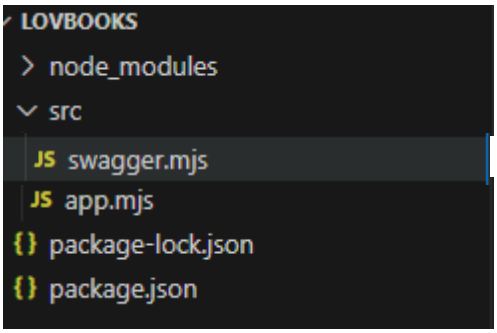

Figure 3 Code - Scripts Base

- Un dossier pour la base de données qui contiendra les données préenregistrées des livres, des utilisateurs et de leurs critiques. En plus d'un sequelize qui se connectera à notre serveur et gérera l'entrée et la sortie des données.

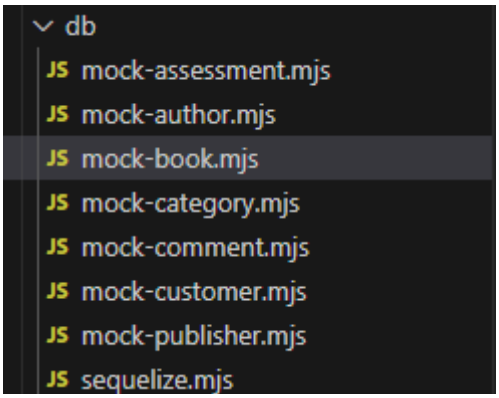

Figure 4 Code - Mock et Sequelize

- Une branche pour la création de modèles par table qui respectent les structures et les données de notre base MySQL db_lovbooks.

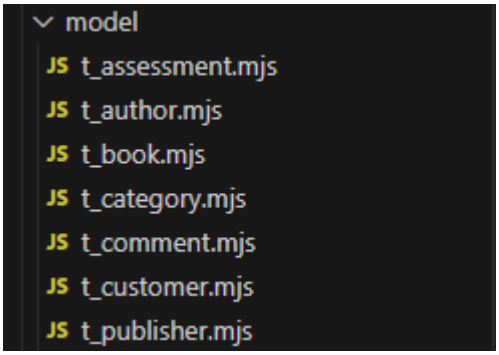

Figure 5 Code - Model table

- Pour le système d'enregistrement et de connexion, ce fichier d'authentification est utilisé.

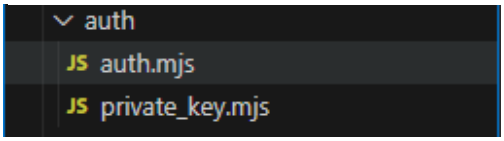

Figure 6 Code - Authentification

- Enfin un dossier destiné à la création des routes selon leurs fonctions pour chaque table et connexion.

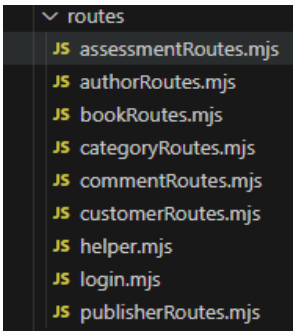

Figure 7 Code - Routes

2.5. Routes

La construction des routes est définie en fonction des fonctionnalités que nous souhaitons utiliser pour gérer les données dans nos tables et notre base de données. De cette manière, une méthode sera créée en fonction de la demande par itinéraire. Dans certaines fonctions comme gérée les utilisateurs, l'authentification de l'utilisateur admin est nécessaire.

En matière de sécurité, si l'utilisateur n'obtient pas l'accès à sa route ou problèmes de connexion, une erreur d'état http ou de validation correspondante s'affichera.

Vous trouverez ci-dessous une liste de routes correspondant à chaque table de notre base de données. Dans chaque case, nous expliquons la fonctionnalité, la route utilisée, l'insertion d'une requête JSON, un exemple de message d'exécution de route réussie et rejetée.

1. **BooksRouter**

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les livres	GET http://localhost:3000 /api/books/		La liste des livres a bien été récupérée	Une erreur est survenue lors de la récupération des livres
Rechercher un livre par son ID	GET http://localhost:3000 /api/books/(id)		Le livre don't l'id vaut 3 a bien été récupéré »	Le livre demandé n'existe pas. Merci de réessayer avec un autre identifiant
Rechercher un livre par son nom	GET http://localhost:3000 /api/books?title=[nom]		Il y a 1 livres qui correspondent au terme de la recherche	Il y a 0 livres qui correspondent au terme de la recherche
Obtenir tous les	GET http://localhost:3000		Il y a 6 comments qui correspondent	Le livre demandé n'existe pas.

commentaires d'un livre	/api/books/ (id)/comments		au terme de la recherche	Merci de réessayer avec un autre identifiant
Obtenir toutes les notes d'un livre	GET http://localhost:3000 /api/books/(id)/notes		Il y a 2 notes qui correspondent au terme de la recherche	Le livre demandé n'existe pas. Merci de réessayer avec un autre identifiant
Créer un livre	POST http://localhost:3000 /api/books/	"title" : "Heidi 2 : The end", "number_of_pages" : 161, "year_of_publication" : 1999, "cover_image":"exe", "extract_pdf":" exe ", "category_id":1 "summary": " exe "	Le livre Heidi 2 : The end a bien été créé	notNull Violation: t_book.title cannot be null
Commenter un livre	POST http://localhost:3000 /api/books/ (id)/comments	"content":"C'est le mieux livre"	Le commentaire 9 a bien été créé !	Validation error: Le contenu du commentaire ne peut pas être vide
Evaluer un livre	POST http://localhost:3000 /api/books/(id)/notes	"assessment": 5	La note id 9 a bien été créé	Validation error: L'évaluation doit être un nombre
Modifier un livre	PUT http://localhost:3000 /api/users/(id)	« title : Fondations»	Le livre Fondation dont l'id vaut 2 a été mis à jour avec success	Le livre demandé n'existe pas. Merci de réessayer avec un autre identifiant.
Supprimer un livre	DELETE http://localhost:3000 /api/books/(id)		Le livre Le Seigneur des Anneaux a bien été supprimé !	Le livre demandé n'existe pas. Merci de réessayer avec un autre identifiant

2. **CustomersRouter**

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les utilisateurs	GET http://localhost:3000 /api/users/		La liste des users a bien été récupérée	Une erreur est survenue lors de la récupération des livres
Rechercher un utilisateur par son ID	GET http://localhost:3000 /api /users/(Id)		Le user don't l'id vaut 1 a bien été récupéré	Le user demandé n'existe pas. Merci de réessayer avec un autre identifiant
Rechercher un utilisateur par son nom	GET http://localhost:3000 /api/ users?pseudo=[nom]		Il y a 1 users qui corresponde nt au terme de la recherche	Une erreur est survenue lors de la récupération des users
Obtenir tous les libres d'un utilisateur	GET http://localhost:3000 /api//users/(id)/books		Il y a 5 livres qui corresponde nt au terme de la recherche	Le user demandé n'existe pas. Merci de réessayer avec un autre identifiant
Créer un utilisateur	POST http://localhost:3000 /api /users/	pseudo" : "etmlsd", date_enter": "2000-01-15T00:00:00.000Z password" : "etml2"	L'utilisateur etmlsd a bien été créé	Validation error: Le pseudo ne peut pas être vide.
Modifier un utilisateur	PUT http://localhost:3000 /api /users/(id)	"pseudo" : "voyageur79 "	Le user voyageur79 dont l'id vaut 1 a été mis à jour avec success	Validation error: Le pseudo ne peut pas être vide.
Supprimer un utilisateur	DELETE http://localhost:3000 /api/users/(id)		Le user sportif_29 a bien été supprimé !	Le user demandé n'existe pas. Merci de réessayer avec un autre identifiant

3. **LoginRouter**

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Vérification de la connexion	POST http://localhost:3000/api/login/	"pseudo" : "etml", "password" : "etml"	L'utilisateur a été connecté avec succès	Le mot de passe est incorrecte.

4. **CategoryRouter**

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les catégories	GET http://localhost:3000/api/categorys/		La liste des category a bien été récupérée	une erreur est survenue lors de la récupération des category
Rechercher une catégorie par son ID	GET http://localhost:3000/api/categorys/(Id)		Le category don't l'id vaut 1 a bien été récupéré	Le category demandé n'existe pas. Merci de réesayer avec un autre identifiant
Rechercher une catégorie par son nom	GET http://localhost:3000/api/categorys?name=[nom]		Il y a 1 categories qui correspondent au terme de la recherche	Il y a 0 categories qui correspondent au terme de la recherche
Obtenir tous les livres de cette catégorie	GET http://localhost:3000/api/categorys/(id)/books		Il y a 9 livres qui correspondent au terme de la recherche	Le category demandé n'existe pas. Merci de réesayer avec un autre identifiant
Créer une catégorie	POST http://localhost:3000/api/categorys /	"name":"Fiction-Romance"	Le category Fiction-Romance a bien été créé !	Validation error: Le nom de la catégorie ne peut pas être vide.

Modifier une catégorie	PUT http://localhost:3000 /api/categorys /(id)	"name":"Adult	Le category Adult dont l'id vaut 1 a été mis à jour avec success	Le nom de la catégorie ne peut pas être vide
Supprimer une catégorie	DELETE http://localhost:3000 /api/categorys /(id)		Le category Adult a bien été supprimé !	Le category demandé n'existe pas. Merci de réessayer avec un autre identifiant

5. AuthorsRouter

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les auteurs	GET http://localhost:3000 /api/authors/		La liste des Author a bien été récupérée	une erreur est survenue lors de la récupération des Author
Rechercher un auteur par son ID	GET http://localhost:3000 /api /authors/(Id)		Le Author don't l'id vaut 1 a bien été récupéré	Le Author demandé n'existe pas. Merci de réesayer avec un autre identifiant
Rechercher un auteur par son nom	GET http://localhost:3000 /api/authors?name=[nom]		Il y a 1 authors qui correspond ent au terme de la recherche	Il y a 0 authors qui correspondent au terme de la recherche
Rechercher un auteur par son prénom	GET http://localhost:3000 /api/authors? first_name=[prénom]		Il y a 1 authors qui correspond ent au terme de la recherche	Il y a 0 authors qui correspondent au terme de la recherche
Obtenir tous les livres de cet auteur	GET http://localhost:3000 /api/authors /(id)/books		Il y a 5 livres qui correspond ent au terme de la recherche	Le Author demandé n'existe pas. Merci de réesayer avec un autre identifiant

Créer un auteur	POST http://localhost:3000 /api/author /	"first_name ":"Toledo", "name": "Adrian"	Le Author Adrian a bien été créé !	Validation error: Le prénom ne peut pas être vide.
Modifier un auteur	PUT http://localhost:3000 /api authors/(id)	"first_name ":"adri"	Le Author adri dont l'id vaut 1 a été mis à jour avec success	Le prénom ne peut pas être vide
Supprimer un auteur	DELETE http://localhost:3000 /api/authors /(id)		Le Author Brown a bien été supprimé !	Le Author demandé n'existe pas. Merci de réessayer avec un autre identifiant

6. PublisherRouter

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les éditeurs	GET http://localhost:3000 /api/publishers/		La liste des Publisher a bien été récupérée	une erreur est survenue lors de la récupération des Publisher
Rechercher un éditeur par son ID	GET http://localhost:3000 /api/ publishers /(Id)		Le Publisher don't l'id vaut 1 a bien été récupéré	Le Publisher demandé n'existe pas. Merci de réessayer avec un autre identifiant
Rechercher un éditeur par son nom	GET http://localhost:3000 /api/publishers? name=[nom]		Il y a 1 publishers qui correspondent au terme de la recherche	Il y a 0 publishers qui correspondent au terme de la recherche
Créer un éditeur	POST http://localhost:3000 /api/publishers /	"name" : "BandaiNamco"	Le Publisher BandaiNamco a bien été créé !	Validation error: Le nom de l'éditeur ne peut pas être vide.
Modifier un éditeur	PUT http://localhost:3000 /api /publishers /(id)	"name":"Bandi"	Le Publisher Bandi dont l'id vaut 1 a été mis à jour avec succès	Le Publisher n'a pas pu être mis à jour. Merci de réessayer

				dans quelques instants
Supprimer un éditeur	DELETE http://localhost:3000 /api/publishers /(id)		Le Publisher Shueisha a bien été supprimé !	Le Publisher demandé n'existe pas. Merci de réessayer avec un autre identifiant

7. AssessmentRouter

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les évaluations	GET http://localhost:3000 /api/assessments/		La liste des notes a bien été récupérée	Une erreur est survenue lors de la récupération des notes
Rechercher une évaluation par son ID	GET http://localhost:3000 /api/assessments/(Id)		La note don't l'id vaut 1 a bien été récupéré	La note demandée n'existe pas. Merci de réessayer avec un autre identifiant
Créer une évaluation	POST http://localhost:3000 /api/assessments/	"assessment": "5"	Le note id 9 a bien été créé !	Validation error: L'évaluation doit être un nombre.
Modifier une évaluation	PUT http://localhost:3000 /api/assessments/(id)	"assessment" : "2"	La note dont l'id vaut 1 a été mis à jour avec succès	La note n'a pas pu être mis à jour. Merci de réessayer dans quelques instants
Supprimer une évaluation	DELETE http://localhost:3000 /api/assessments/(id)		La note 7 a bien été supprimé !	La note demandé n'existe pas. Merci de réessayer avec un autre identifiant

8. CommentRouter

Fonctionnalité	Route	JSON	Message de Réponse	Message d'Erreur
Liste de tous les commentaires	GET http://localhost:3000 /api/comments/		La liste des commentaires a bien été récupérée	Une erreur est survenue lors de la récupération des commentaires
Rechercher un commentaire par son ID	GET http://localhost:3000 /api /comments/(Id)		Le commentaire don't l'id vaut 2 a bien été récupéré	Le commentaire demandé n'existe pas. Merci de réessayer avec un autre identifiant
Créer un commentaire	POST http://localhost:3000 /api /comments/	"content" : "C'est super!", "book_id" : 1	Le commentaire id 9 a bien été créé !	Le commentaire n'a pas pu être ajouté. Merci de réessayer dans quelques instants
Modifier un commentaire	PUT http://localhost:3000 /api /comments/(id)	"book_id":"'	Le commentaire dont l'id vaut 1 a été mis à jour avec succès	Le commentaire n'a pas pu être mis à jour. Merci de réessayer dans quelques instants.
Supprimer un commentaire	DELETE http://localhost:3000 /api/comments/(id)		Le commentaire 6 a bien été supprimé !	Le commentaire demandé n'existe pas. Merci de réessayer avec un autre identifiant

3. Réalisation

3.1. System d'authentification

Les routes des utilisateurs sont gérées à la demande de l'utilisateur admin et de ses droits d'accès. Pour réaliser cette fonction, un système d'authentification « auth » a été créé selon le modèle de la demande d'un nom d'utilisateur et d'un mot de passe confidentiel. En saisissant ces données, les utilisateurs reçoivent un jeton JWT à durée limitée de valeur Bearer qui leur donnera accès à l'itinéraire demandé. L'exécution de ses jetons est réalisée grâce à la dépendance Jswebtoken.

Un point important à suivre est la conservation et confidentialité des données des utilisateurs et des mots de passe. Pour sécuriser les données, nous avons utilisé à la dépendance Bcrypt pour le chiffrement et la vérification des données destinées au mot de passe de l'utilisateur. Dans le code, on utilise le fichier login.mjs qui, en fonction des informations saisies dans la route, va comparer les mots de passe et leur authentification avec la méthode compare().

3.2. Gestion des statuts HTTP

Pour savoir si une requête HTTP a été correctement exécutée, il existe des statuts HTTP qui identifieront si la ligne de routage a été appliquée correctement ou si une erreur s'est produite.

Pour couvrir les éventuelles erreurs qui peuvent être commises dans une API web, nous avons couvert les routes les plus courantes.

L'erreur la plus courante causée par le client est l'erreur 404, qui se produit lorsque le serveur ne trouve pas la ressource demandée. Et aussi l'erreur 401 et 404 qui identifie les autorisations des utilisateurs et si existe.

Du côté des erreurs du serveur, un message d'erreur 500 s'affiche lorsque le serveur a rencontré un problème et n'est pas en mesure de répondre à la demande.

Dans le code, le « then & cath » est utilisé pour identifier l'erreur et faire un retour du «statut» de l'erreur et un message d'explication. Exemples :

```
return res.status(404).json({"Le livre demandé n'existe pas. Merci de
réessayer avec un atre identifiant."})
return res.status(401).json({ `L'utilistaeur n'est pas autorisé à accéder à
cette ressource.`});

return res.status(404).json({L'utilisateur demandé n'existe pas});

res.status(500).json({"Le livre n'a pas pu être mis à jour. Merci de réessayer
dans quelques instants", data:error});
```

3.3. Fonctionnalités techniques

- Documentation Swagger : Swagger est un outil qui vous permet de documenter votre code et de l'afficher sur une page web de manière visuelle sur http://localhost:3000/api-docs/ . Dans ce projet, cet outil a été utilisé pour pouvoir afficher les schémas des tables de notre base de données telles que t_book, t_customer, t_assessment, t_author, t_category, t_comment, et t_publisher.

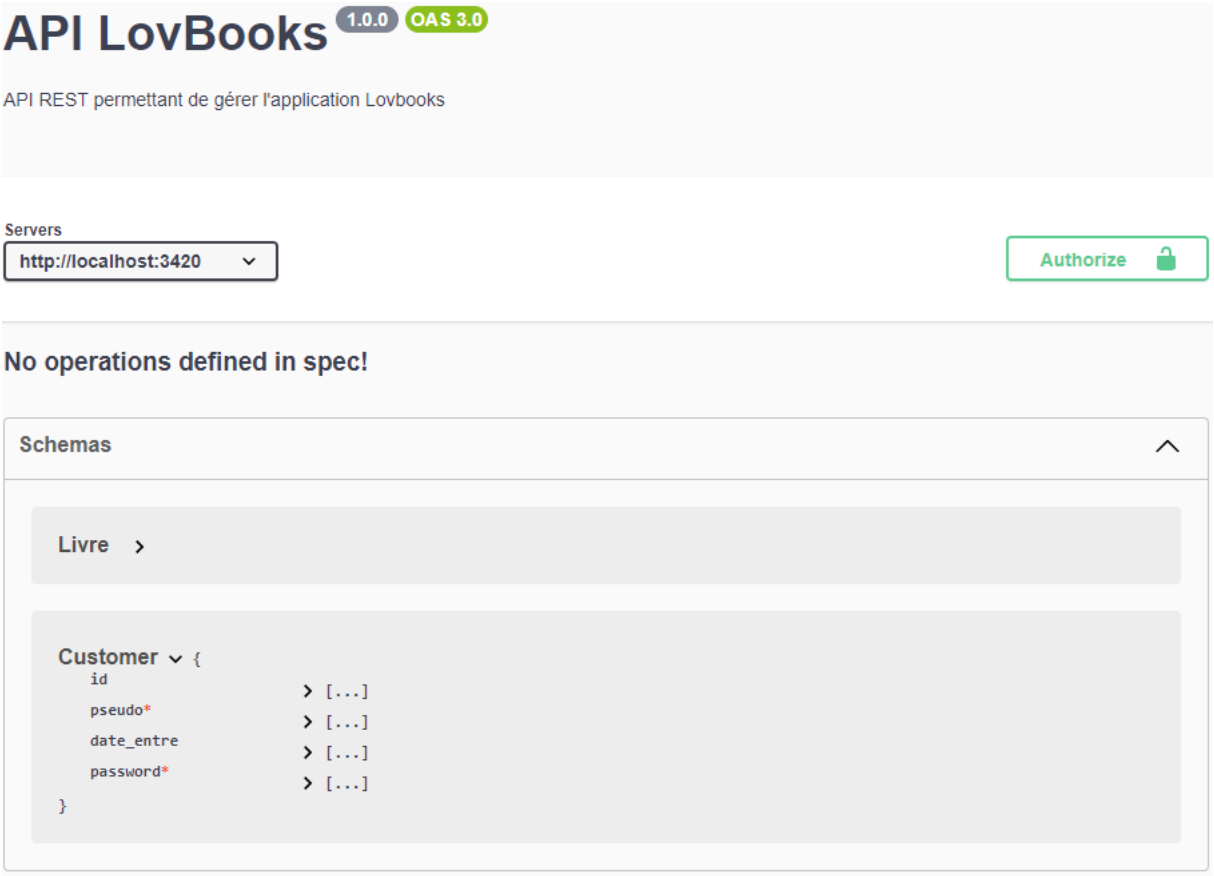

Figure 8 Doc Swagger Exemple

- Dépendances :

La création de ce code API a été réalisée grâce aux dépendances installées et utilisées. Voici une liste des dépendances et ses versions :

```
"dependencies": {
  "bcrypt": "^5.1.1",
  "express": "^4.18.2",
  "jsonwebtoken": "^9.0.2",
  "mysql2": "^3.9.1",
  "sequelize": "^6.37.1",
  "swagger-jsdoc": "^6.2.8",
  "swagger-ui-express": "^5.0.0"
},
"devDependencies": {
  "nodemon": "^3.1.0"
```

Figure 9 Dépendances

4. Test – Insomnia

Pour tester les routes de notre API, nous avons utilisé la plateforme Insomnia. Cette plateforme dédiée au test d'API en local et cloud, offre un environnement dédié à l'envoi et à la réception des réponses de nos routes en fonction de leurs opérations

Liste de routes sur Insomnia REST :

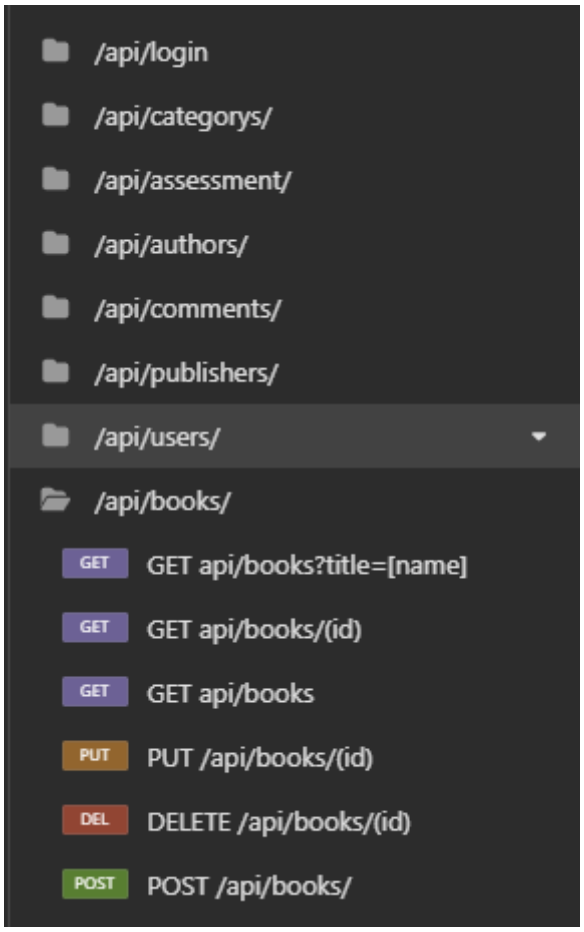

Figure 10 Insomnia Routes

Exemple de route :

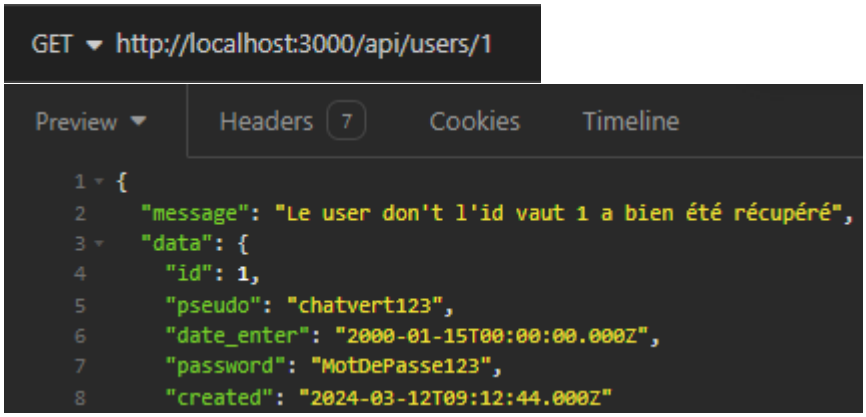

Figure 11 Insomnia Route exemple

5. Conclusions

LovbooksAPI offre plusieurs routes pour la manipulation et l'utilisation des informations de notre base de données. Pour la réalisation de ce projet, il a été très utile de construire une modélisation des données dans un schéma de style MCD et MLD. Ce document a servi de guide pour la construction de l'API et de la base de données. Un autre point important a été la bonne communication au sein de l'équipe, la division du travail et l'utilisation de GitHub pour partager et créer le code ensemble.

Cette API entièrement fonctionnelle attend d'être utilisée pour être reliée à une page web et à son front-end. Ce projet offre une base de données locale déjà créée, de sorte qu'elle peut être testée pour connaître ses fonctionnalités à travers des supports tels qu'Insomnia Rest.

De plus, un modèle de données a été créé pour chaque type de table, ce qui assure une bonne intégrité de l'insertion des données et de la création des tables.

De plus, pour chaque demande de route, un système de validation et d'erreur a été créé. Celui-ci fournit à l'utilisateur un message en fonction de l'erreur rencontrée.

La dernière fonction importante est la création d'un système d'authentification avec un jeton de validation. Ce système nous a permis de créer des profils de connexion, et de tester la sécurité de nos routes destinées exclusivement aux utilisateurs connectés.

- Conclusion d'Adrian :

Mon rôle principal a été la création de la base de code, avec diverses fonctions telles que l'authentification, les routes de départ, la connexion à la base de données, l'intégration dans la plateforme Insomnia, affichage de message de validation d'erreur sur le model, rapport base du cette projet et la création de routes imbriquées. La plus grande difficulté de ce projet a été de pouvoir utiliser et créer des routes imbriquées, mais grâce à notre chef de projet et à la documentation de Sequelize, cela a été réalisé sans le moindre problème.

- Conclusion de Mathis

Mon rôle principal dans ce projet a été la gestion de la base de données, comprenant la gestion des modèles, des mocks, la connexion et l'insertion des données dans la base de données, ainsi que les relations entre les modèles afin que cela soit cohérent avec le MCD. Mon second rôle a été la création des routes avec la gestion des erreurs 404 et 500. Nous n'avons pas rencontré beaucoup de problèmes dans ce projet, car avec Adrian, nous avons communiqué sur toutes les tâches qui étaient faites et sur celles sur lesquelles nous travaillions chacun. Grâce à Trello, la planification et la visualisation de l'avancement du projet ont permis qu'il se déroule sans difficulté. Si je devais noter un problème, cela serait la création et l'utilisation des routes imbriquées ainsi que les erreurs de génération de données dans le mock. Mais cela a pu être résolu par la suite pour donner un produit fonctionnel

6. Autoévaluation

ÉVALUATION DES COMPÉTENCES EN PRATIQUE POUR LA FORMATION

informatique-ETML

INFORMATICIEN-NE

Nom et Prénom : Toledo Adrian & Mathis Botteau
Année de formation - classe : 2éme CID2B
Enseignant : Antoine Mveng

Nom du projet : P_DB - 165
Semaines : 8
Dates : 02.02.2024 - 15.03.2024

COMPÉTENCES		LARGEMENT ACQUIS (LA)	ACQUIS (A)	PARTIELLEMENT ACQUIS (PA)	NON ACQUIS (NA)	Ignoré	Résultat
PROFESSIONNELLES	**Rythme de travail Rapidité, Efficacité**	☐ Rapide et soutenu ☒ Optimale ☐	☐ Productivité normale ☐ Respect des délais fixés ☐	☐ Lent ou irrégulier ☐ Hors délais ☐	☐ Trop lent ☐ Pas concerné par les délais ☐	NO	
	Qualité du travail	☐ Travail utilisable et transmissible ☐ ☐	☐ Travail utilisable et transmissible avec retouches ☒ ☐	☐ Travail nécessitant des améliorations pour être utilisable ☐ ☐	☐ Travail inutilisable ☐ ☐	NO	
	Niveau de maîtrise technique	☐ ☒ Maîtrise ☐	☐ Comprend et applique ☐ ☐	☐ A des lacunes, applique par mimétisme ☐ ☐	☐ Echec dans les notions de base ☐ ☐	NO	
	Autonomie	☐ ☒ Indépendant ☐	☐ Aide justifiée ☐ ☐	☐ Souvent besoin d'aide ☐ ☐	☐ Dépendant ☐ ☐	NO	
METHODOLOGIQUES	**Processus de travail**	☐ Intégration des règles et processus de travail ☐ ☐	☐ Respect des règles de processus de travail ☐ ☐	☐ Peu concerné ☐ ☐	☐ Pas concerné ☐ ☐	YES	Ignoré Ignoré Ignoré
	Expression orale et écrite Technique de présentation	☐ ☒ Maîtrise les différents moyens et outils de communication et de documentation ☐	☐ Utilise les différents moyens et outils de communication et de documentation ☐ ☐	☐ N'utilise pas toujours les différents moyens et outils de communication et de documentation ☐ ☐	☐ Ignore la plupart des moyens et outils de communication et de documentation ☐ ☐	NO	
	Approche écologique et économique	☐ Recours systématique aux technologies et moyens qui ménagent les ressources et les coûts ☐ ☐	☐ Utilisation régulière des technologies et moyens qui ménagent les ressources ☐ ☐	☐ Peu concerné ☐ ☐	☐ Pas concerné ☐ ☐	YES	Ignoré Ignoré Ignoré
SOCIALES	**Aptitude au travail en équipe Gestion des conflits Communication**	☐ Influence positivement le groupe ☐ Réagit de manière réfléchie et cherche des solutions ☐	☐ Maintien les bonnes relations ☐ Ne provoque pas de conflit et participe aux solutions ☐	☐ Ne participe pas à la cohésion du groupe ☐ Réagit de manière irréfléchie et/ou disproportionnée ☐	☐ Influence négative marquée ☐ ☐	YES	Ignoré Ignoré Ignoré

Rythme de travail : nous pensons que notre rythme de travail a été constant et croissant afin d'accomplir toutes les tâches souhaitées et certaines tâches facultatives telles que l'ajout de routes non sollicitées mais mises en œuvre.

Qualité de travail : notre projet suit tous les points souhaités par notre chef d'équipe mais on peut trouver de petites fautes d'orthographe dans le code, la base de données et les commentaires du code.

Niveau de maîtrise : nous pensons que nous avons réussi à réaliser un projet complet et que nous avons réussi à comprendre les tâches de chacun au fur et à mesure que nous travaillions ensemble et que nous avons réutilisé le travail de chacun pour faire avancer le projet.

Presentation : nous considérons que nous avons respecté à tout moment le cahier des charges du projet, que nous avons créé une documentation dès le début et que nous l'avons complétée au fur et à mesure de l'avancement du projet. Les plateformes Trello et Github ont également été respectées et utilisées de manière appropriée.

7. Webgraphie

- Documentation Sequelize : https://sequelize.org/
- Guide de construction de l'API : « Steps[1-14] » fournies par le chef de projet
- Traducteur : https://www.deepl.com/
- ChatGPT : https://chat.openai.com - L'utilisation de chatGPT a été utilisée pour construire des documents Mocks fictifs afin de créer une fausse base de données.