DOCUMENTATION TECHNIQUE

I. L’analyse du besoin et la conception

I. 1. Le contexte

Une bibliothèque universitaire souhaite moderniser son système de gestion des prêts et retours de livres. Actuellement, elle utilise des registres papier et des fichiers Excel, ce qui entraîne des erreurs de suivi et une gestion inefficace des emprunts. Le but du projet est de concevoir et développer une application permettant d’améliorer la gestion des livres, des emprunts et des utilisateurs de la bibliothèque.

I. 2. Le besoin

L’application a besoin d’une connexion. Une connexion valide renvoie des tokens JWT reprenant le principe des access token et des refresh tokens. Les utilisateurs sont séparés en 3 groupes. Les étudiants, les professeurs et enfin les bibliothécaires. Les étudiants peuvent consulter les livres de la bibliothèque. Ils peuvent aussi choisir d'en réserver. Ils ne peuvent réserver qu'un exemplaire d'un livre à la fois. Les professeurs, eux, possèdent les mêmes droits mais ils peuvent réserver plusieurs exemplaires de plusieurs livres à la fois. Les bibliothécaires peuvent ajouter des livres dans la bibliothèque et consulter les statistiques des livres ainsi que la liste des livres empruntés avec la date de l'emprunt.

I. 3 Description des objets

- Les utilisateurs possèdent un id, un nom, un prénom, un email, un mot de passe, un id de département et un id de groupe
- Un groupe possède un id et un nom. Il en existe trois : Etudiant, Professeur et Bibliothécaire
- Les départements possèdent un id et un nom. Exemple de département : Informatique, Génie Civil, Technique de Commercialisation.
- Les livres possèdent un id, un titre, un auteur, un id de catégorie, un résumé, un ISBN, une année de publication et un éditeur.
- Une catégorie possède un id et un nom. Exemple de catégorie : "Fantastique", "science fiction"

- Un exemplaire de livre contient un id, un id de livre, un id d'état, s'il est disponible et la date d'ajout.
- Un état possède un id et un nom. On retrouve ces différentes valeurs : "Neuf", "Très bon", "Bon", "Moyen", "Abimé"
- Un emprunt possède un id, un id d'exemplaire, un id d'utilisateur, une date d'emprunt, une date de retour prévu, une date de retour effectué et un id de statut
- Un statut possède un id et un nom. Exemple de statut : "Emprunté", "En retard".

I. 4 Description des fonctionnalités :

Pour les étudiants :
- Consulter les livres
- Emprunter un livre
- Voir l'historique de ses livres empruntés

Pour les professeurs :
- Consulter les livres
- Emprunter un livre (la durée de l'emprunt est plus longue que celle des étudiants)
- Consulter les livres empruntés
- Voir l'historique de ses livres empruntés
- Faire des demandes d'achats de livres.

Pour les bibliothécaires :
- Consulter les livres
- Consulter les livres empruntés
- Ajouter un livre
- Ajouter un exemplaire
- Voir les statistiques des livres empruntés
- Envoyer des notifications aux personnes qui n'ont pas rendu les livres
- Supprimer les livres
- Modifier les livres

II. Aspect technique

II. 1. Architecture

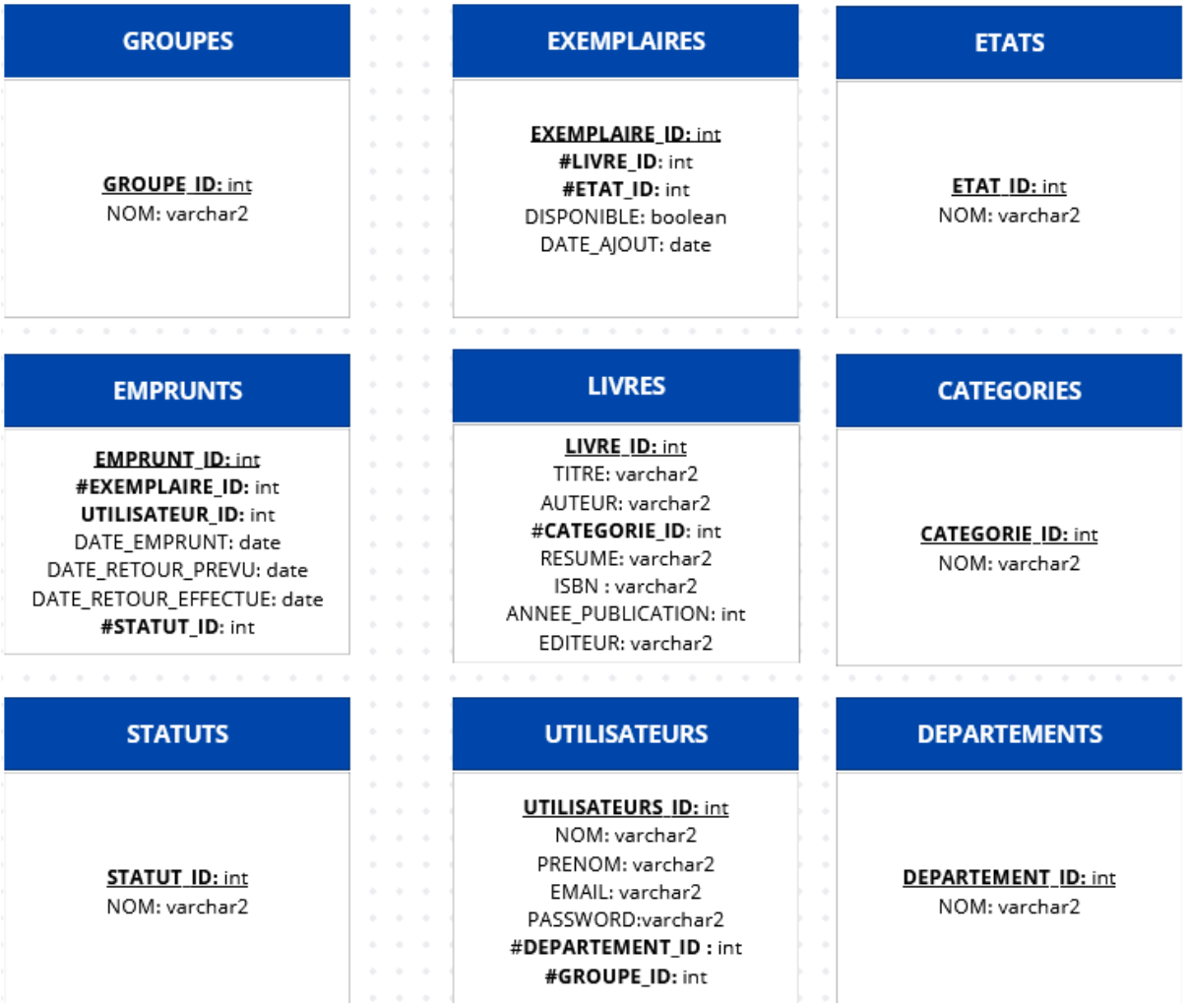
Le projet sera sous docker. L'objectif est de séparer l'ensemble des services pour avoir un résultat propre. Le frontend sera un docker avec nginx et une application react qui est copiée dedans. Le backend est composé d'une api en Python gérée par le framework FastAPI. La base de données est une base MySQL.

Les fichiers sont organisés selon cette architecture :
- backend/

- app/
  - database.py
  - main.py
  - models.py
  - schemas.py
- requirements.txt
- frontend
  - src
    - assets
    - components
    - modules
    - pages

II. 2 Les tables de la base de données

Voici la description des différentes tables de la base de données.



II. 3 La maquette

Une maquette a été faite pour mieux visualiser les composants à réaliser. Voici le lien pour la voir. Elle est aussi dans le repository git sous le nom de maquette.pdf.

II. 4 Les tests

L'API sera testé unitairement. Les tests seront dans le backend et permettront de vérifier que les routes font bien ce qu'elles doivent faire. Exemple : la route pour créer un livre doit bien créer un livre dans la base de données.

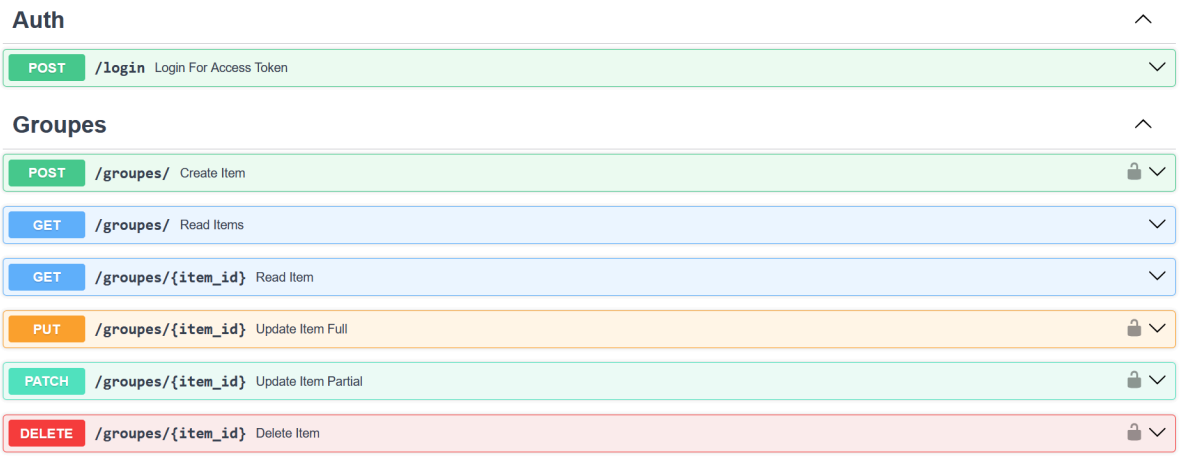
III. Pour la période donnée

III. 1 Restriction du périmètre

En vue du nombre de fonctionnalités et des problèmes que l'on a eu à mettre en place un environnement, pour ce projet nous allons réduire le périmètre du projet. Nous allons juste faire en sorte d'avoir toutes les routes pour créer l'ensemble des objets. La connexion sera implémentée avec la restriction des routes pour les utilisateurs avec un token et le bon groupe.

III. 2 L'API

Voici l'ensemble des routes à avoir :



Etats

- POST /etats/ Create Item
- GET /etats/ Read Items
- GET /etats/{item_id} Read Item
- PUT /etats/{item_id} Update Item Full
- PATCH /etats/{item_id} Update Item Partial
- DELETE /etats/{item_id} Delete Item

Categories

- POST /categories/ Create Item
- GET /categories/ Read Items
- GET /categories/{item_id} Read Item
- PUT /categories/{item_id} Update Item Full
- PATCH /categories/{item_id} Update Item Partial
- DELETE /categories/{item_id} Delete Item

Statuts

- POST /statuts/ Create Item
- GET /statuts/ Read Items
- GET /statuts/{item_id} Read Item
- PUT /statuts/{item_id} Update Item Full
- PATCH /statuts/{item_id} Update Item Partial
- DELETE /statuts/{item_id} Delete Item

Departements

- POST /departements/ Create Item
- GET /departements/ Read Items
- GET /departements/{item_id} Read Item
- PUT /departements/{item_id} Update Item Full
- PATCH /departements/{item_id} Update Item Partial
- DELETE /departements/{item_id} Delete Item

Livres

- POST /livres/ Create Item
- GET /livres/ Read Items
- GET /livres/{item_id} Read Item
- PUT /livres/{item_id} Update Item Full
- PATCH /livres/{item_id} Update Item Partial
- DELETE /livres/{item_id} Delete Item

Exemplaires

POST	/exemplaires/	Create Item
GET	/exemplaires/	Read Items
GET	/exemplaires/{item_id}	Read Item
PUT	/exemplaires/{item_id}	Update Item Full
PATCH	/exemplaires/{item_id}	Update Item Partial
DELETE	/exemplaires/{item_id}	Delete Item

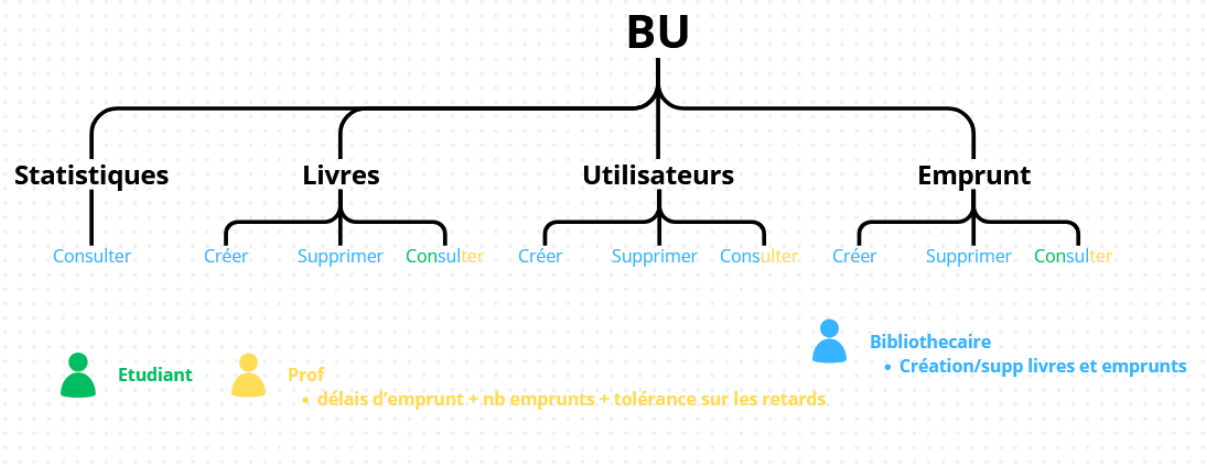
Utilisateurs

POST	/utilisateurs/	Create Item
GET	/utilisateurs/	Read Items
GET	/utilisateurs/{item_id}	Read Item
PUT	/utilisateurs/{item_id}	Update Item Full
PATCH	/utilisateurs/{item_id}	Update Item Partial
DELETE	/utilisateurs/{item_id}	Delete Item

Emprunts

POST	/emprunts/	Create Item
GET	/emprunts/	Read Items
GET	/emprunts/{item_id}	Read Item
PUT	/emprunts/{item_id}	Update Item Full
PATCH	/emprunts/{item_id}	Update Item Partial
DELETE	/emprunts/{item_id}	Delete Item

Réflexion sur les droits de certaines routes :



III. 3 Les changements à faire pour le futur

Au moment de la rédaction de ce document, plusieurs améliorations ont été trouvées mais pas appliquées par manque de temps. Premièrement, nous avons vu en cours comment fusionner les docker-compose et ne plus avoir besoin d'un Dockerfile pour le proxy. Deuxièmement, mieux utiliser le framework FastAPI et créer des constructeurs plus génériques peuvent être envisagés. Enfin l'ajout de différents tests peut être possible. Comme par exemple des tests unitaires en front pour vérifier le visuel et le comportement des composants. Des tests d'intégrations automatisés avec Playwright pour garantir la non régression d'un point de vue client. Et aussi des tests de performances pour s'assurer que l'application fonctionne de manière optimisée.