Micro-services: Rapport

Front:	http://localhost:8080/
API-Catalogue:	http://localhost:5001/
	http://localhost:5001/api/v1/resources/books/all
API-Notes:	http://localhost:5002/
	http://localhost:5002/api/v1/resources/notes/all
API-Recherche:	http://localhost:5003/
	http://localhost:5003/api/v1/resources/books/search?arg=Ann

Présentation orale:

Lors de la présentation orale, nous avions le **front**, le **catalogue**, les **notes** (fonctionnel mais sans vérification si le client a déjà noté) et la **recherche** (fonctionnel uniquement sur des noms d'auteurs complets).

A ce moment nous n'avions rien sur la partie **Docker** et le projet était sous forme d'un **monolithe**.

Les infos sur le projet:

La page d'accueil affiche la liste de tous les livres dans la base avec la moyenne des notes:

- Cette page utilise le catalogue pour afficher tous les livres et notes pour leur associer leur moyenne.

Il est possible sur cette page d'effectuer une recherche et de noter les livres:

- La recherche prend en compte le titre / l'auteur / la date de publication / ou la première phrase. Ces informations n'ont pas besoin d'être renseignées complètement du fait d'une requête LIKE. (ajouter la possibilité de faire une recherche multiple et sur plus de critères. ex: le genre)

  Il y a un problème de recherche lorsque le critère contient un apostrophe.
  Pour la partie Docker, la recherche utilise le même fichier .bd que le catalogue avec la mise en place d'un volume.

- La notation est actualisée en direct, ce qui permet de voir la nouvelle moyenne sans recharger la page. Il est cependant possible de noter plusieurs fois le même livre (à corriger).

Docker:

- Nous avons un fichier docker-compose qui permet de déployer le projet uniquement via la commande up.
  Cette commande permet de créer les services:

1) Catalogue:
   - Dockerfiles/Catalogue/Dockerfile
   - port: 5001
   - Met en place un volume pour partager le fichier de données: /var/lib/sqlite3/data /books.db
2) Notes:
   - Dockerfiles/Notes/Dockerfile
   - port: 5002
3) Recherche:
   - Dockerfiles/Recherche/Dockerfile
   - port: 5003
   - Utilise le fichier de données partagé par le catalogue: /var/lib/sqlite3/data/books.db
   - Dépend du catalogue
4) Netfleeks (Front):
   - Dockerfiles/Netfleeks/Dockerfile
   - port: 8080

Authentification:
Nous avons commencé à mettre en place l'authentification: \Docker\Dockerfiles\Authen...
Ce service est composé d'un fichier .bd pour stocker les informations concernant les identifiants et utilise un système de hash pour crypter les passwords.
A cela s'ajoute un système de token pour vérifier la validité d'une connexion.

Ce service permet de créer un nouvelle utilisateur via: /registration
De se connecter via: /login
D'afficher l'ensemble des utilisateurs: - get:/users
Ou de les supprimer: - delete:/users

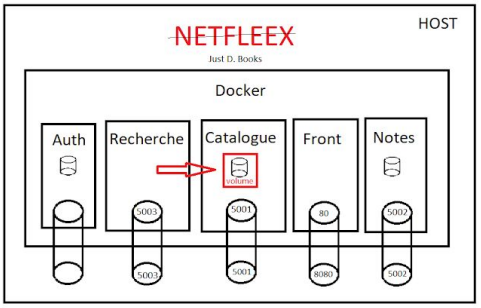
Pour effectuer les tests, nous avons un simple fichier htm: test.html

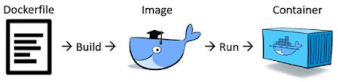
Il est également possible d'accéder à certaines ressources uniquement lorsque qu'un jwt valide est envoyé dans le header d'une requête. Cependant, un fichier html n'est pas suffisant pour tester cela.

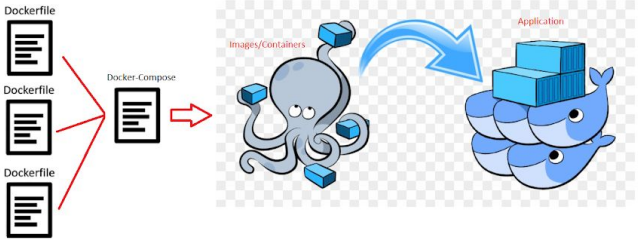
De même pour se déconnecter où il est nécessaire d'avoir un jwt valide pour accéder à la page web. La déconnexion permet de rendre un jwt invalide et empêcher quelqu'un de l'utiliser pour accéder aux différentes ressources avec ce dernier.

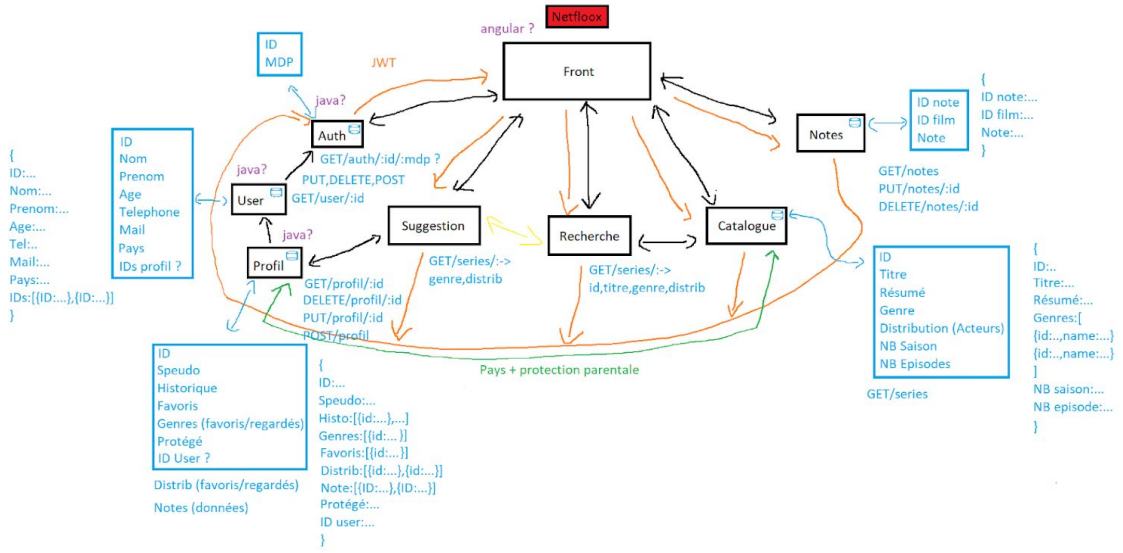
Amélioration envisageables:

1) Rendre fonctionnel l'authentification.
2) Corriger le système de notes et l'imbriquer avec l'authentification.
3) Permettre de mettre des livres favoris grâce à l'authentification.
4) Mettre plusieurs champs de recherche en même temps: (plusieurs genres.)
5) Pouvoir afficher toutes les informations d'un livre sur une page en cliquant dessus.
6) Pouvoir afficher uniquement les favoris.
7) Ajouter des information sur les livres (ex:genres) pour améliorer la recherche
8) Pleins d'autres =p (design, sécurité,)


HOST
NETFLEEX
Just D. Books
Docker
Auth
Recherche
Catalogue
Front
Notes
volume
5003
5001
80
5002
5003
5001
8080
5002


Dockerfile
Build
Image
Run
Container


Dockerfile
Dockerfile
Dockerfile
Docker-Compose
Images/Containers
Application


Netfloox
angular ?
Front
ID
MDP
JWT
java?
Auth
GET/auth/:id/:mdp ?
PUT,DELETE,POST
java?
User
GET/user/:id
java?
Profil
Suggestion
Recherche
Catalogue
Notes
GET/profil/:id
DELETE/profil/:id
PUT/profil/:id
POST/profil
GET/series/:->
genre,distrib
GET/series/:->
id,titre,genre,distrib
Pays + protection parentale
GET/notes
PUT/notes/:id
DELETE/notes/:id
GET/series
{
ID:...
Nom:...
Prenom:...
Age:...
Tel:...
Mail:...
Pays:...
IDs:[{ID:...},{ID:...}]
}
ID
Nom
Prenom
Age
Telephone
Mail
Pays
IDs profil ?
ID
Speudo
Historique
Favoris
Genres (favoris/regardés)
Protégé
ID User ?
Distrib (favoris/regardés)
Notes (données)
{
ID:...
Speudo:...
Histo:[{id:...},...]
Genres:[{id:...}]
Favoris:[{id:...}]
Distrib:[{id:...},{id:...}]
Note:[{ID:...},{ID:...}]
Protégé:...
ID user:...
}
ID note
ID film
Note
{
ID note:...
ID film:...
Note:...
}
ID
Titre
Résumé
Genre
Distribution (Acteurs)
NB Saison
NB Episodes
{
ID:..
Titre:...
Résumé:...
Genres:[
{id:..,name:...}
{id:..,name:...}
]
NB saison:...
NB episode:...
}