WEB SERVICE NETFLIX

FITAMANT Guillaume

EGU Hugo-Jean

OBJECTIF PROJET

- Construire une partie du système d'information de la société NETFLIX

- 4 modules indépendants :
  - Application référentiel Utilisateurs
  - Application Media
  - Application Poster
  - Application Front pour démonstration

- 1 livrable qui sera la composition des 4 applications.

FONCTIONNALITÉ

- Application référentiel clients finaux : Gère les utilisateurs (clients, admin)
  - CRUD
  - Services exposé (utilisateurs actif/suspendu)
- Application Media : Gère les médias (nom, description, url vidéo, date…)
  - CRUD
  - Services exposé (recherche par catégorie)
- Application Poster : Gère les posters
  - CRUD
  - Service exposé (fourni le poster courant en fonction du moment de la journée)
- Application Front :
  - Recherche Media par genre
  - Recherche Media par catégorie
  - Affichage du résultat (nom, poster, description, date, vidéo)
  - Pouvoir jouer le média

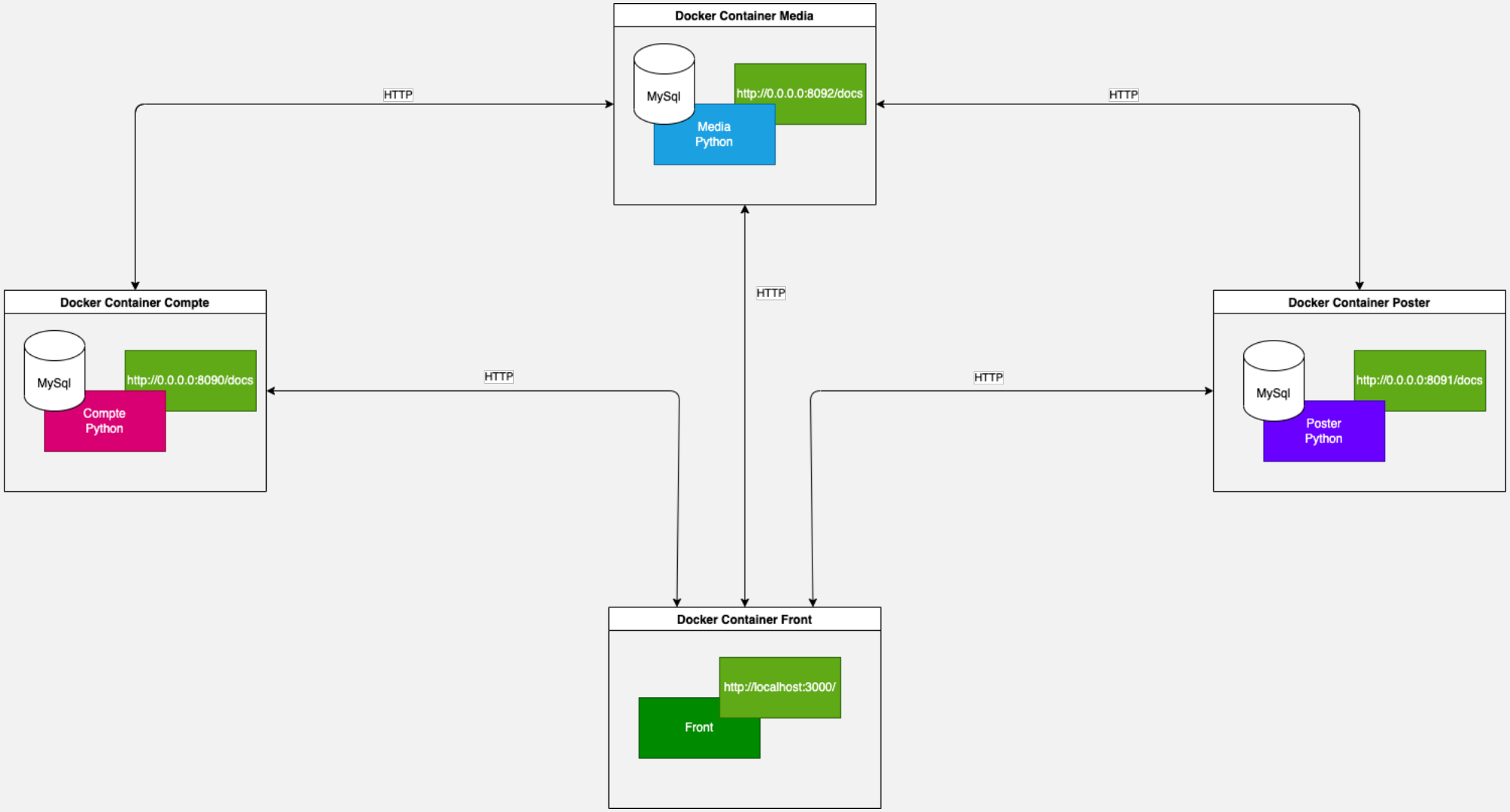
RÈGLES DE GESTION / FONCTION

- RG1 : La liste des médias proposés doit tenir compte de la localisation du client
- RG2 : Les posters des médias dépendent du moment de la journée
- RG3 : Le résultat de la recherche n'est disponible que pour les clients ACTIF
- RG4 : Aucune œuvre cinématographique ne pourra être physiquement supprimé du catalogue

- F1 : Responsable peut modifier uniquement la description d'un média
- F2 : Responsable peut suspendre un compte utilisateur
- F3 : Responsable peut créer/modifier un poster

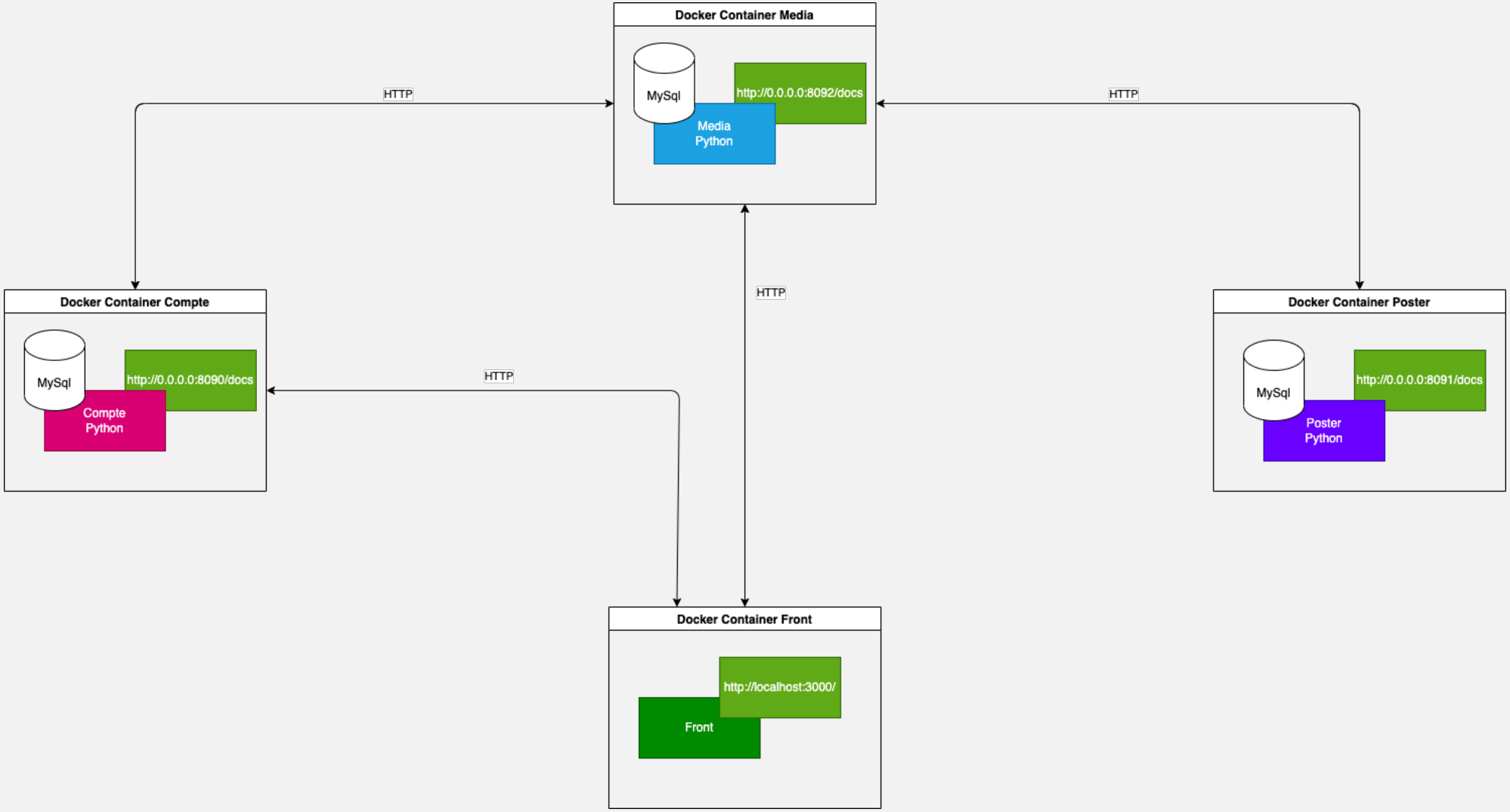
ARCHITECTURE FONCTIONNELLE

Prestataire



ARCHITECTURE FONCTIONNELLE

Utilisateur



ARCHITECTURE LOGICIELLE

Exemple : Recherche Media par catégorie et genre

@EndPoint API : media / gestion / {id_user} / {category} / {country} / {kind}

↳ Vérification statut du User => appel Utilisateur API
Si : Statut User == ACTIF
Alors :
Appel Base de donnée Media (Query SQL avec les paramètres de l'appel)
Appel Base de donnée Poster : retourne le poster en fonction du moment de la journée
Retourner : Model Media et Poster
Statut code HTTP 200 OK

Sinon : On retourne un statut code HTTP : 401 unauthorized

CONTENEURISATION

Hébergeur : Github
Merge branch Release

Build Docker Image avec Github CI
Push Image dans la registry Github

Commande : docker compose up
Appel github registry pour récupérer les images et lancer l'application

STACKS TECHNIQUES

DOCKER

- Docker compose
- Docker build image
- GitHub CI registry

REACT

- Front Application
- TailWind CSS

PYTHON

- Fast API (FLASK)

MYSQL

- Base de données

PROCÉDURE DÉPLOIEMENT

- Installer docker / docker compose
- Git clone https://github.com/hugoj78/servicewebnetflixdocker
- Se mettre dans le dossier « servicewebnetflixdocker »
- Lancer la commande : « docker compose up »
- Ouvrir son navigateur à l'url suivante : « http://localhost:3000/ »

DÉMONSTRATION

LES AXES D'AMÉLIORATIONS

- Déploiement du système complet sur un Cloud (exemple : AWS, Azure, IBM)
- Gérez l'écosystème de conteneurs avec Kubernetes

- Ajouter un système de like/dislike pour les vidéos
  - Introduire un algorithme de proposition de vidéo en fonction des likes et visionnage

- Ajouter une liste de favoris

- Ajouter la notion de saison pour les séries (conception)

CE QUE NOUS AVONS APPRIS

- Web service :
  - Exposition
  - Communication
  - Sécurisation
- Adaptation à un langage que nous avons peu pratiqué
- Conteneurisation

MERCI