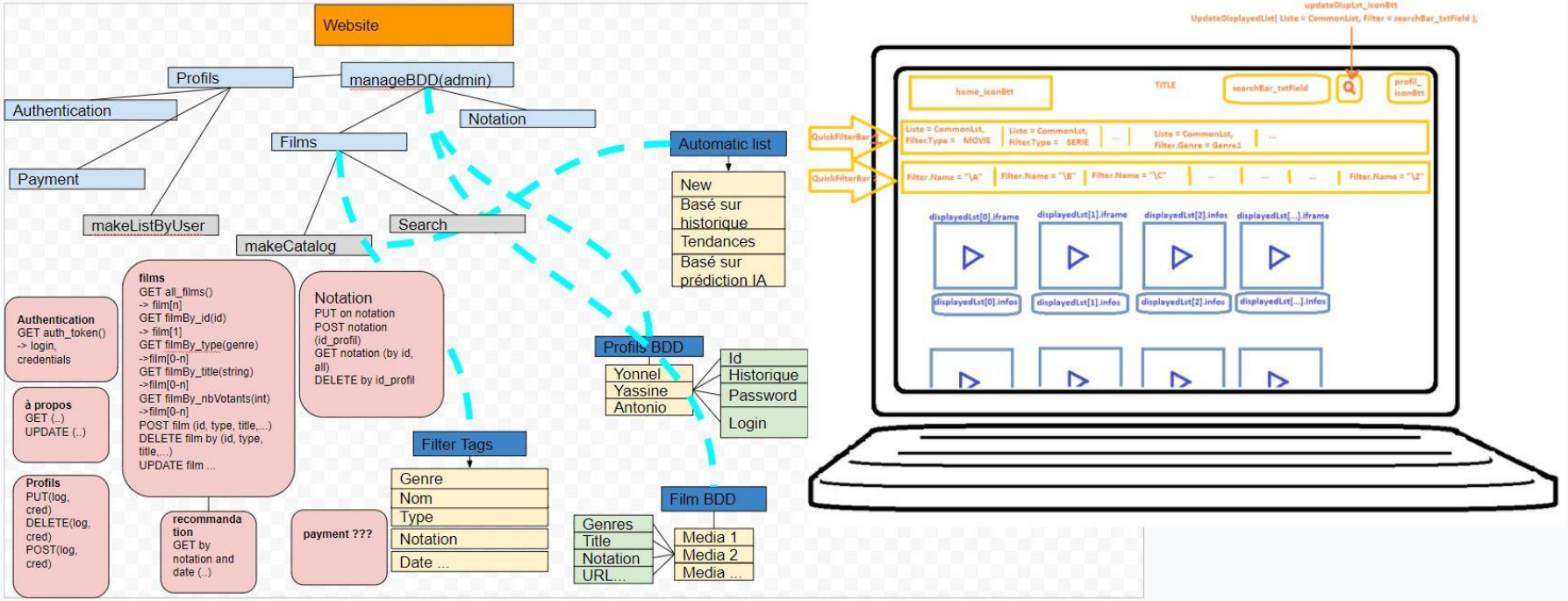
ARCHITECTURE ET MICRO-SERVICES

NETFLOUX

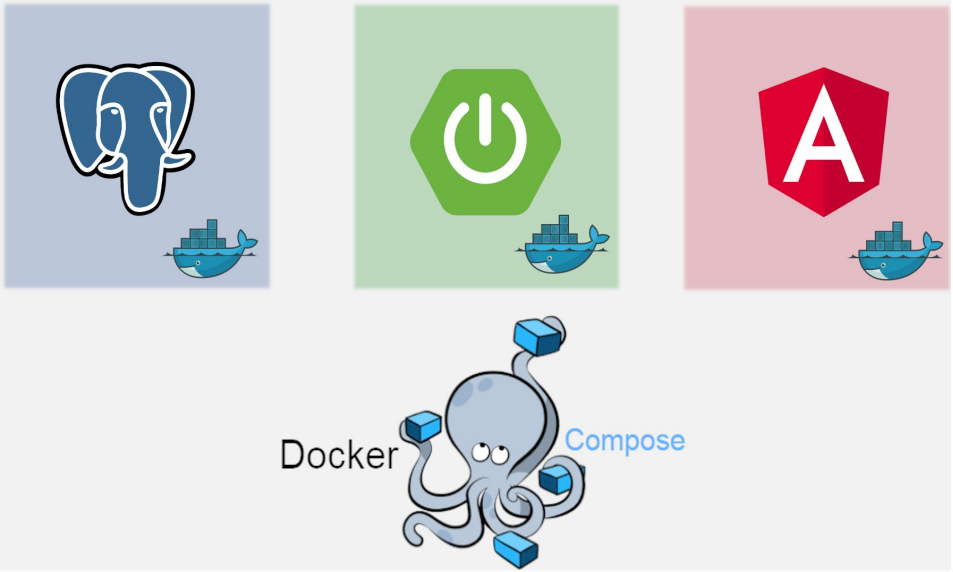
SOMMAIRE

- Schéma de notre architecture
- Technologies utilisées
- Etat d’avancement
- Objectif à atteindre

Schéma de notre architecture



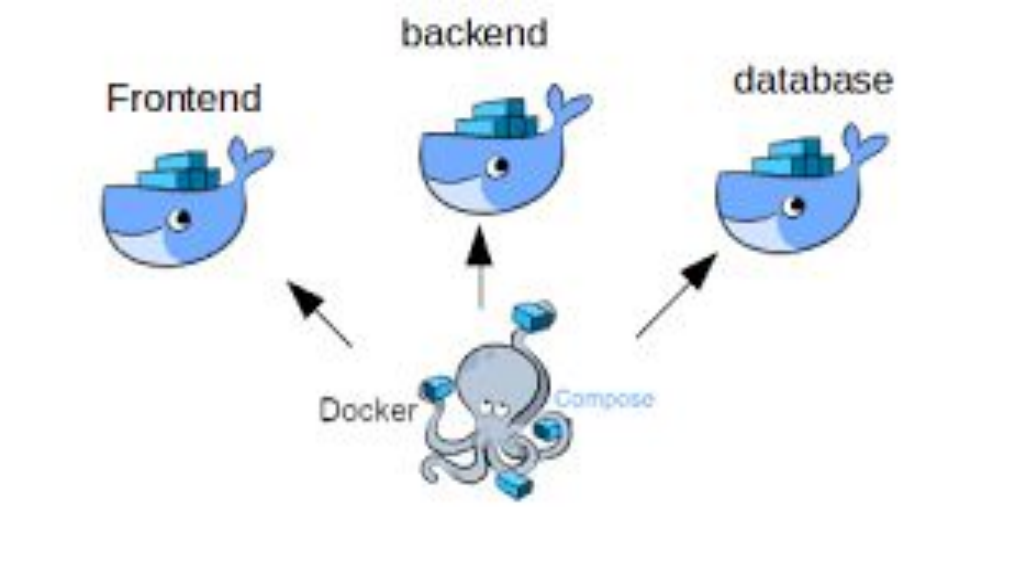
Technologies utilisées



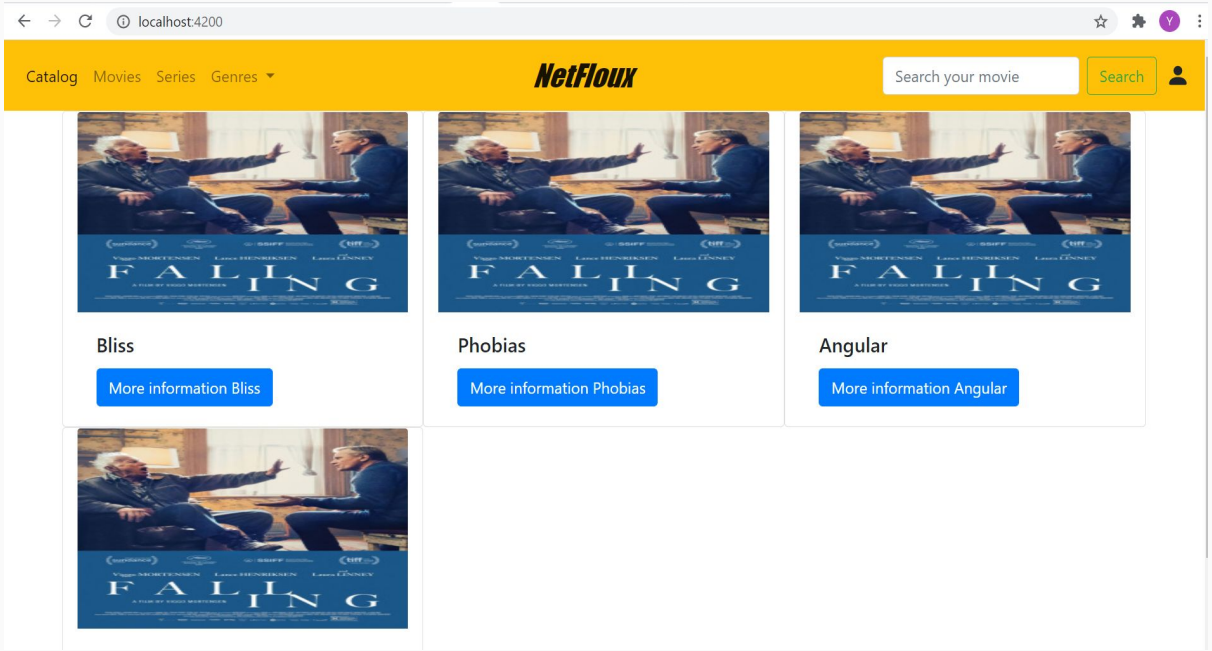
python

Flask
web development,
one drop at a time

Technologies utilisées



Etat d’avancement



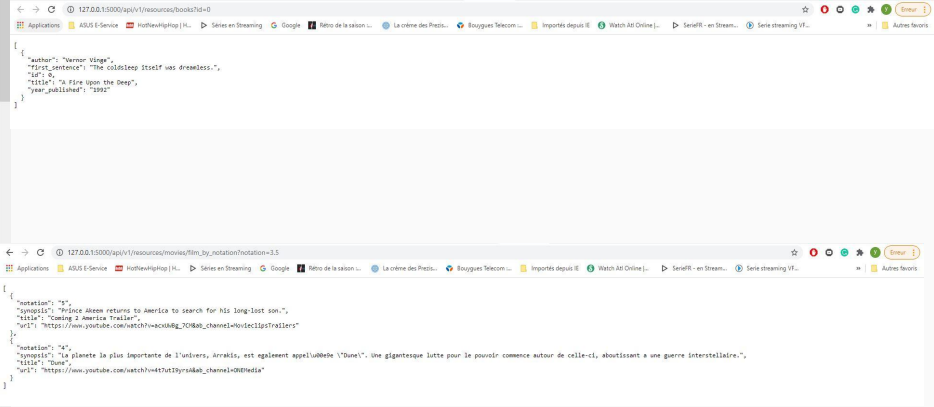
Etat d’avancement

search_films_api.py - C:\Users\ynyokas\Documents\ESIEE cours\E5\Docker & Microservices\projects\api\search_films_api.py (3.7.3)

File Edit Format Run Options Window Help

```
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True

# Create some test data for our catalog in the form of a list of dictionaries.
movies = [
    {'id': 0,
     'title': 'Coming 2 America Trailer',
     'url': 'https://www.youtube.com/watch?v=acxUWBg_7CM&ab_channel=MovieclipsTrailers',
     'synopsis': 'Prince Akeem returns to America to search for his long-lost son.',
     'genre': 'comedy',
     'type': 'movie',
     'date': '2021',
     'notation': '5'},
    {'id': 1,
     'title': 'Dune',
     'url': 'https://www.youtube.com/watch?v=4t7utI9yrsA&ab_channel=ONEMedia',
     'synopsis': 'La planete la plus importante de l\'univers, Arrakis, est egalement appelée "Dune". Une gigantesque
     'genre': 'sci-fi',
     'type': 'movie',
     'date': '2020',
     'notation': '4'},
    {'id': 2,
     'title': 'The White Tiger',
     'url': 'https://www.youtube.com/watch?v=ZMuDvgtuiBI&ab_channel=MovieclipsTrailers',
     'synopsis': 'The epic journey of a poor Indian driver who must use his wit and cunning to break free from servit
     'genre': 'drama',
     'type': 'movie',
     'date': '2021',
     'notation': '3'}
]


#other functions
################
def string_equal(string_1, string_2):
    if (string_1 == string_2) or (string_1+'s' == string_2) or (string_1 == string_2+'s'):
        return True
################


#api functions

@app.route('/', methods=['GET'])
def home():
    return '''<h1>Web API for films search test</h1>
<p>This web api is a test to check if we can access the films by
searching by id, title, genre, type, release_data, notation, ...</p>'''

#get all
@app.route('/api/v1/resources/movies/all', methods=['GET'])
def api_all_movies():
    return jsonify(movies)

#get film by id
@app.route('/api/v1/resources/movies/film_by_id', methods=['GET'])
def api_movie_by_id():
    if 'id' in request.args:
        id = int(request.args['id'])
    else:
        return "Error: No id field provided or exceeding the number of films on the website."

    movies_links = []
```

Ln: 41 Col: 16



Objectifs à atteindre

- Intégrer les web API créés avec Flask/SpringBoot dans des images Docker (Dockerfiles) et les connecter avec la BDD.
- Savoir utiliser les commandes PUT, POST et DELETE pour les web api Flask.
- Faire fonctionner notre architecture avec les images Docker (DockerCompose).
- Compléter le front-end pour gerer les futures améliorations.