

ECOLE SUPÉRIEURE POLYTECHNIQUE



DEPARTEMENT GENIE INFORMATIQUE

Filaire : Informatique

Matière : BDNG

SUJET : Application qui utilise le cluster MongoDB

Membres :

Mohamed MBAYE (mohamedmbaye@esp.sn)

Mouhamadou Amine Diagne
(mouhamadouaminediagne1@esp.sn)

Introduction

Ce projet a pour objectif de développer une application pour visualiser la liste des auteurs . Les données utilisées sont un extrait de la base DBLP qui se trouve dans notre cluster MongoDB.

Les technologies principales utilisées dans ce projet sont :

- **MongoDB** : Base de données NoSQL distribuée, robuste et flexible.
- **Docker** : Plateforme de conteneurisation permettant de déployer et d'exécuter des applications de manière isolée et reproductible.
- **React** : Bibliothèque JavaScript pour construire des interfaces utilisateur réactives.
- **Express** : Framework Node.js pour développer des applications web et des API.

Architecture du Projet

Le projet est divisé en deux principaux composants : le serveur et le frontend.

Serveur

- **Serveur Express** : Gère les requêtes HTTP et interagit avec la base de données MongoDB.
- **MongoDB** : Stocke les données des publications et des auteurs.

Frontend

- **React** : Affiche la liste des auteurs récupérés depuis le serveur backend.

Installation et Configuration

Prérequis

- Docker et Docker Compose installés sur votre machine.
- Node.js et npm installés sur votre machine (pour le développement local).

Configuration de l'API Backend

Créez le fichier `.env` dans le répertoire `serveur` avec le contenu suivant :

```
MONGODB_URI=mongodb://mongo1:27017,mongo2:27017,mongo3:27017/DBLP?replicaSet=myReplicaSet&retryWrites=true&w=majority
PORT=5000
```

Contenu des fichiers :

- `models/index.js`:

```
const { MongoClient } = require('mongodb');
require('dotenv').config();

const db = {};
const client = new MongoClient(process.env.MONGODB_URI);

Pieces: Comment | Pieces: Explain
✓ db.connect = async () => {
✓   try {
      await client.connect();
      console.log("Connected to the database!");
      db.client = client;
      db.publications = client.db("DBLP").collection("publis");
✓   } catch (error) {
      console.error("Cannot connect to the database!", error);
      process.exit();
    }
  };

module.exports = db;
```

Elle permet de nous connecter à la base de données DBLP et d'exécuter des requêtes dans la collecte publis

- `controllers/publication`:

```
const db = require("../models");
const Publication = db.publications;

Pieces: Comment | Pieces: Explain
exports.getAuteur = async (req, res) => {
  try {
    const authors = await Publication.find({}).toArray();
    res.status(200).json(authors);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

Permet de récupérer les documents de la base de donnée et de les afficher dans le corps de la page en json

- `routes/publications.js`:

```

1  module.exports = app => {
2    const publication = require("../controllers/publication.js");
3    let router = require("express").Router();
4
5    // router.get("/", publication.getAuteur);
6    router.get("/", (publication.getAuteur));
7
8    app.use("/authors", router);
9  };
10

```

Permet de définir le chemin pour récupérer le contenu Json de l'API.

- `server.js`:

```

1  const express = require("express");
2  const cors = require('cors');
3  const logger = require("morgan");
4  require('dotenv').config();
5
6  const db = require("../models");
7
8  const app = express();
9
10 const PORT = process.env.PORT || 5000;
11
12 app.use(logger("dev"));
13 app.use(cors());
14 app.use(express.json());
15
16 db.connect().then(() => {
17   require("../routes/publications")(app);
18
19   app.get("/", (req, res) => {
20     res.json({ message: "L'API fonctionne correctement" });
21   });
22
23   app.listen(PORT, () => {
24     console.log(`Backend express server is running on port ${PORT}.`);
25   });
26 }).catch(err => {
27   console.log("La connexion a la BD a echoue", err);
28 });

```

- Dockerfile :

```
1 FROM node:18
2
3 WORKDIR /usr/src/app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 COPY . .
10
11 EXPOSE 5000
12
13 CMD ["node", "server.js"]
14
```

- docker-compose.yml :

```
1 version: '3.8'
2
3 services:
4   serveur:
5     build: ./serveur
6     ports:
7       - "5000:5000"
8     environment:
9       - MONGODB_URI=mongodb://mongo1:27017,mongo2:27017,mongo3:27017/DBLP?replicaSet=myReplicaSet&retryWrites=true&w=majori
10
11     networks:
12       - mongoCluster
13
14   frontend:
15     build: ./frontend
16     ports:
17       - "3000:3000"
18     networks:
19       - mongoCluster
20
21 networks:
22   mongoCluster:
23     external: true
```

Configuration du Frontend React

Contenu des fichiers :

- `src/components/AuteurList.js` :

```
1  import React, { useState, useEffect } from 'react';
2  import axios from 'axios';
3
4  const AuteurListe = () => {
5    const [authors, setAuthors] = useState([]);
6
7    useEffect(() => {
8      axios.get('http://localhost:5000/authors/')
9        .then(response => {
10          const authorsSet = new Set();
11          response.data.forEach(publication => {
12            publication.authors.forEach(author => {
13              authorsSet.add(author);
14            });
15          });
16          setAuthors(Array.from(authorsSet));
17        })
18        .catch(error => {
19          console.error('Affichage impossible', error);
20        });
21    }, []);
22
23    return (
24      <div>
25        <h1>Liste des Auteurs</h1>
26        <ul>
27          {authors.map((author, index) => (
28            <li key={index}>{author}</li>
29          ))}
30        </ul>
31      </div>
32    );
33  };
34  export default AuteurListe;
```

Le composant `AuteurListe` permet de récupérer la liste auteur de notre API en utilisant la fonction `get` du module `axios` puis de les afficher dans une liste `ul` à l'aide de la fonction `map`.

- `src/App.js` :

```

1  import React from 'react';
2  import './App.css';
3  import AuteurListe from './components/AuteurListe';
4
   Pieces: Comment | Pieces: Explain
5  function App() {
6    return (
7      <div className="App">
8        <AuteurListe/>
9      </div>
10     );
11   }
12
13   export default App;
14

```

Utilisation du composant AuteurListe.

Démarrage du Projet

1. Démarrez les conteneurs Docker :

















A l'aide de la commande :

`docker-compose up --build -d`

```

E:\GLSI\SEM2\BD\dblp\projet_bd>docker-compose up --build -d
[+] Building 0.1s (0/0) docker:default
2024/06/14 21:31:57 http2: server: error reading preface from client //.[+] Building 0.0s (0/0) docker:default
2024/06/14 21:31:58 http2: server: error reading preface from client //.[+] Building 1440.5s (17/17) FINISHED
fault
=> [frontend internal] load build definition from Dockerfile      3.1s
=> => transferring dockerfile: 466B                               2.4s
=> [serveur internal] load build definition from Dockerfile      3.2s
=> => transferring dockerfile: 471B                               2.6s
=> [frontend internal] load metadata for docker.io/library/node: 36.2s
=> [serveur internal] load .dockerignore                          1.0s
=> => transferring context: 2B                                     0.7s
=> [frontend internal] load .dockerignore                          1.0s
=> => transferring context: 2B                                     0.7s
=> [frontend 1/5] FROM docker.io/library/node:18@sha256:4149a450f 0.1s
=> [serveur internal] load build context                          27.2s
=> => transferring context: 145.54kB                             21.5s
=> [frontend internal] load build context                          95.0s
=> => transferring context: 3.12MB                                84.1s
=> CACHED [frontend 2/5] WORKDIR /usr/src/app                    0.0s
=> CACHED [serveur 3/5] COPY package*.json ./                    0.0s
=> CACHED [serveur 4/5] RUN npm install                          0.0s
=> CACHED [serveur 5/5] COPY . .                                  0.0s
=> [serveur] exporting to image                                  0.4s
=> => exporting layers                                           0.0s
=> => writing image sha256:e734f9149c9f7cac9ab35387c64ad783590f19 0.1s
=> => naming to docker.io/library/projet_bd-serveur              0.2s
=> CACHED [frontend 3/5] COPY package*.json ./                    0.0s
=> CACHED [frontend 4/5] RUN npm install                          0.0s
=> [frontend 5/5] COPY . .                                       1245.0s
=> [frontend] exporting to image                                  54.6s
=> => exporting layers                                           54.6s
=> => writing image sha256:7db1acb8ef2092a891861902bc85da197f60e 0.0s
=> => naming to docker.io/library/projet_bd-frontend              0.0s
[+] Running 2/2
✔ Container projet_bd-frontend-1 Started 13.5s
✔ Container projet_bd-serveur-1 Started 13.6s

```

<input type="checkbox"/>	▼	 projet_bd		Running (2/2)	0.01%	2 hours ago	  
<input type="checkbox"/>		 frontend-1 e42b6c1916b7  projet_bd-frontend		Running	0.01% 3000:3000 	2 hours ago	  
<input type="checkbox"/>		 serveur-1 7ca368c40129  projet_bd-serveur		Running	0% 5000:5000 	2 hours ago	  

2. Test de l'API et de l'application

- POSTMAN

GET

▼

http://localhost:5000/authors

Send

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinary

Key	Value	Bulk Edit
Key	Value	

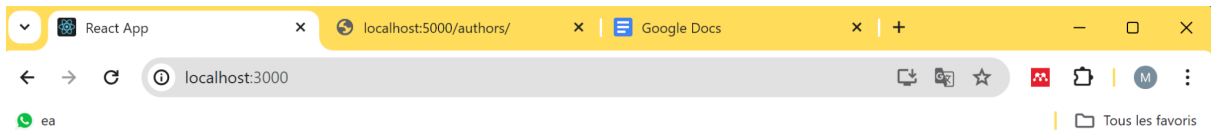
bodyCookiesHeaders (8)Test Results

Status: 200 OKTime: 569 msSize: 5.96 KBSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "_id": "series/cogtech/MayberryC11",
3   "type": "Article",
4   "title": "The Evolution of a Connectionist Model of Situated Human Language Understanding.",
5   "pages": {
6     "start": 143,
7     "end": 167
8   },
9   "year": 2011,
10  "booktitle": "Resource-Adaptive Cognitive Processes",
11  "url": "db/series/cogtech/354089408.html#MayberryC11",
12  "authors": [
13    "Marshall R. Mayberry",
14    "Matthew W. Crocker"
15  ]
16 },
17 }
```

- APPLICATION



Liste des Auteurs

- Marshall R. Mayberry
- Matthew W. Crocker
- Jan Alexandersson
- Norbert Pflieger
- Jürgen Schmidhuber
- Massimo Zancanaro
- Axel Horndasch
- Horst Rapp
- Hans Röttger
- Pádraig Cunningham
- Matthieu Cord
- Sarah Jane Delany
- Frank Wittig
- Jörg Neidig
- Jörg Preißinger
- Martin Charles Golumbic
- Stephan Grashey
- Matthias Schuster
- Patrick Gebhard
- André Berton
- Dirk B.
- Wolfgang Minker
- Boris Brandherm
- Michael Schmitz
- Robert Neelrath
- Frank Lehmann
- Pei Wang
- Wolfgang Wahlster
- Keith Hovel