CouchDB

Dans ce travail, nous nous intéressons à **Apache CouchDB**, un **système de gestion de bases de données orienté documents**.
CouchDB est un SGBD **NoSQL**, open source, soutenu par la **fondation Apache**, qui se distingue par sa **simplicité d'installation**, sa **facilité d'utilisation** et son **fonctionnement basé entièrement sur une API REST**.

Contrairement aux bases de données relationnelles, CouchDB ne repose pas sur des tables mais sur des **documents JSON**, ce qui permet de stocker des données **hétérogènes sans schéma strict**.
Ce choix apporte une grande flexibilité, mais implique que la **logique de structuration** et de cohérence des données est en grande partie déléguée à l'application.

CouchDB expose toutes ses fonctionnalités via le **protocole HTTP**, en utilisant les principaux verbes REST :

- **GET** : récupérer une ressource
- **PUT** : créer ou remplacer une ressource
- **POST** : envoyer des données à une ressource
- **DELETE** : supprimer une ressource

Ainsi, n'importe quel client HTTP (curl, Postman, navigateur, PowerShell, etc.) peut être utilisé pour interagir avec CouchDB.

Dans ce TP, nous allons :

1. Vérifier le fonctionnement de CouchDB
2. Créer une base de données
3. Insérer des documents JSON
4. Insérer un ensemble de documents en masse
5. Consulter et supprimer des documents

MANIPULATIONS RÉALISÉES (COMMANDES EXACTES)

Toutes les commandes ci-dessous sont faites **en PowerShell** avec CouchDB actif sur le port **5984**

Vérification que CouchDB fonctionne

Invoke-WebRequest http://localhost:5984 -UseBasicParsing

⇨ Résultat attendu : informations générales sur CouchDB (version, vendor, etc.)

Accès à l'interface graphique (Fauxton)

Navigateur :

http://localhost:5984/_utils

⇨ Interface graphique pour visualiser les bases et les documents

Création d'un administrateur (obligatoire)

Invoke-WebRequest -Method PUT
http://localhost:5984/_config/admins/youssef `

-Body '"samir"' -UseBasicParsing

Puis test :

Invoke-WebRequest http://youssef:samir@localhost:5984 -UseBasicParsing

Création d'une base de données (PUT)

Invoke-WebRequest -Method PUT
http://youssef:samir@localhost:5984/films -UseBasicParsing

Réponse : {"ok":true}

Récupération des informations sur la base (GET)

Invoke-WebRequest http://youssef:samir@localhost:5984/films -UseBasicParsing

Insertion d'un document JSON avec un identifiant (PUT)

Invoke-WebRequest -Method PUT
http://youssef:samir@localhost:5984/films/doc1 `

-Body '{"titre":"Inception","annee":2010}' -ContentType "application/json" `

-UseBasicParsing

Tentative d'insertion avec un ID existant (conflit)

Invoke-WebRequest -Method PUT
http://youssef:samir@localhost:5984/films/doc1`

-Body '{"titre":"Autre film"}' -ContentType "application/json" `

-UseBasicParsing

-> Résultat : **conflit**, car l'identifiant existe déjà

Insertion d'un document sans identifiant (POST)

Invoke-WebRequest -Method POST
http://youssef:samir@localhost:5984/films `

-Body '{"titre":"Matrix","annee":1999}' -ContentType "application/json"
`

-UseBasicParsing

CouchDB génère automatiquement un _id

Consultation d'un document par son identifiant (GET)

Invoke-WebRequest http://youssef:samir@localhost:5984/films/doc1 -
UseBasicParsing

Insertion d'un document depuis un fichier JSON

Supposons un fichier film.json :

{

"titre": "Avatar",

"annee": 2009,

"genre": ["Science-fiction"]

}

Commande :

Invoke-WebRequest -Method POST http://youssef:samir@localhost:5984/films `

-InFile film.json -ContentType "application/json" -UseBasicParsing

Insertion de plusieurs documents en masse (_bulk_docs)

Fichier films_couchdb.json :

```
{
 "docs": [
   {"_id":"f1","titre":"Film 1","annee":2000},
   {"_id":"f2","titre":"Film 2","annee":2001}
 ]
}
```

Commande :

Invoke-WebRequest -Method POST http://youssef:samir@localhost:5984/films/_bulk_docs `

-InFile films_couchdb.json -ContentType "application/json" -UseBasicParsing

Import massif réussi

Suppression d'un document (DELETE)

Il faut préciser la version (_rev) :

Invoke-WebRequest -Method DELETE `

"http://youssef:samir@localhost:5984/films/doc1?rev=1-xxxxxxxx" `

-UseBasicParsing

POINTS IMPORTANTS

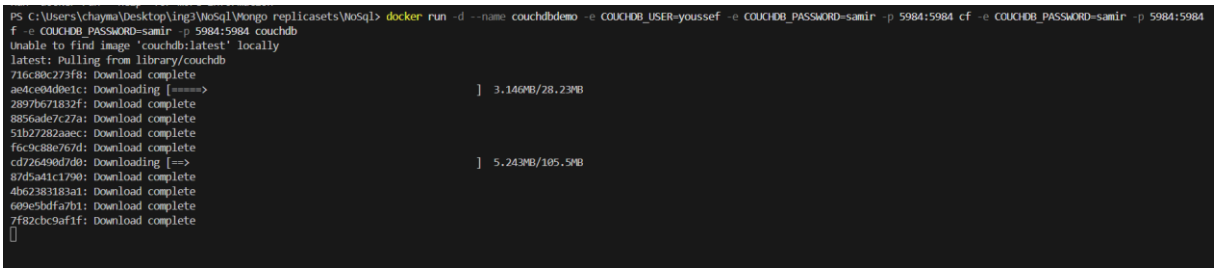
- CouchDB stocke des **documents JSON**
- Il **n'impose pas de schéma**
- Les documents peuvent être **imbriqués**
- L'identifiant automatique est préférable en environnement distribué
- Les bases NoSQL peuvent introduire de la **redondance**
- Les données peuvent devenir incohérentes si mal conçues
- La normalisation des bases relationnelles n'est pas imposée ici

CONCLUSION

Ce TP a permis de manipuler CouchDB via son API REST, en utilisant les méthodes HTTP fondamentales.
La flexibilité offerte par les documents JSON facilite l'insertion de données complexes, mais nécessite une réflexion préalable sur la structuration des données afin d'éviter les incohérences.

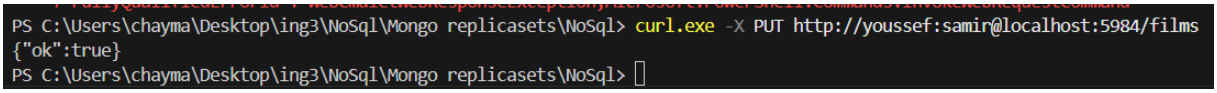
Ci-joint les manipulations de bases effectués

Cette capture illustre le lancement de CouchDB à l'aide de Docker. La commande docker run permet de démarrer un conteneur basé sur l'image officielle CouchDB. Des variables d'environnement sont définies afin de créer un utilisateur administrateur (COUCHDB_USER) et son mot de passe (COUCHDB_PASSWORD). Le port 5984 du conteneur est exposé et mappé sur le port 5984 de la machine locale, ce qui permet d'accéder à CouchDB via un navigateur ou un client HTTP. Cette étape est essentielle pour disposer rapidement d'une instance fonctionnelle de CouchDB sans installation manuelle.



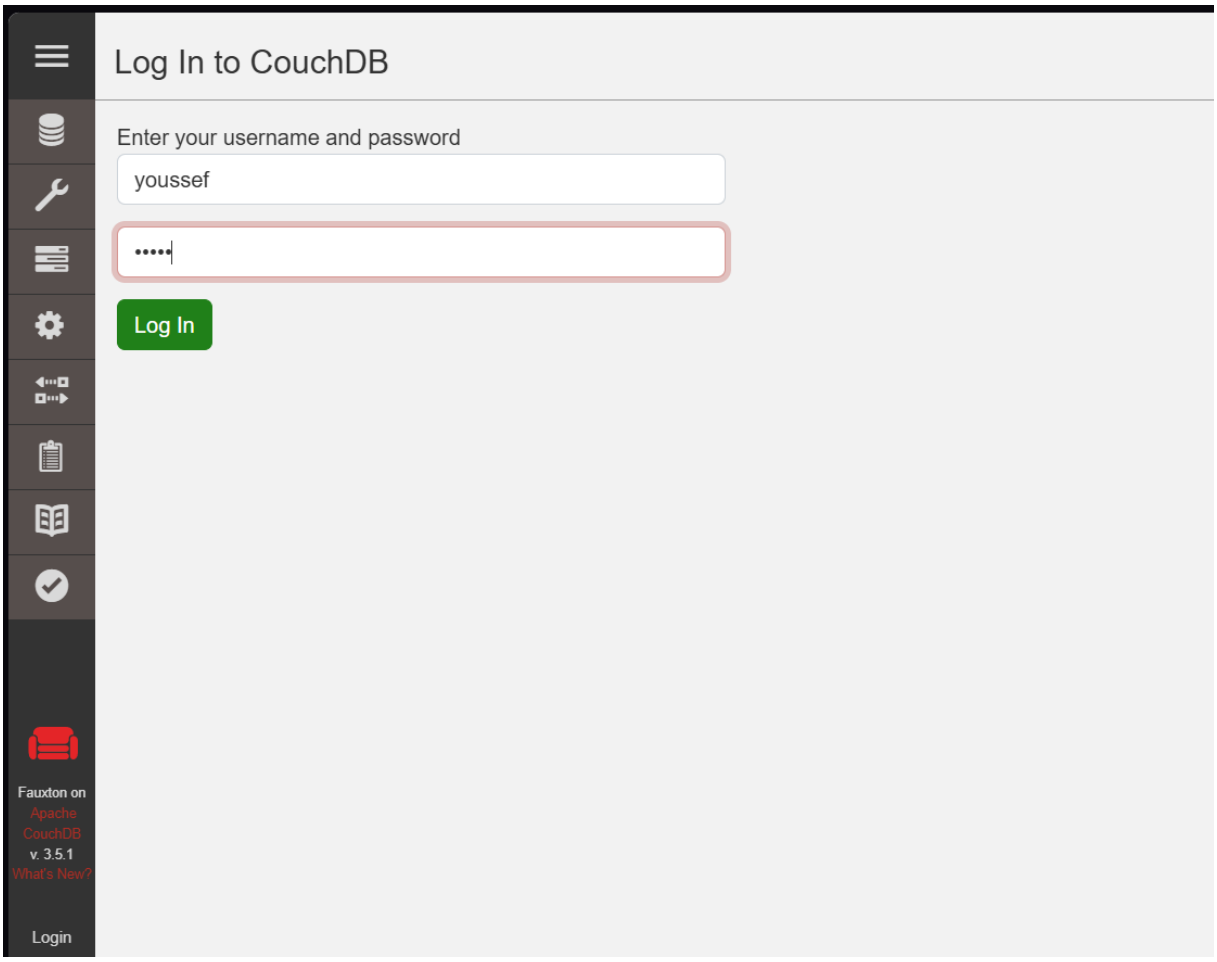
Cette capture montre l'envoi d'une requête HTTP de type GET vers l'URL http://localhost:5984. La réponse retournée par CouchDB confirme que le service est bien actif et accessible. Elle contient des informations générales telles que la version de CouchDB, son identifiant et les fonctionnalités disponibles. Cette requête permet de valider que le serveur CouchDB est correctement lancé avant toute manipulation de bases de données.

Cette capture correspond à l'accès à l'interface graphique de CouchDB, appelée **Fauxton**, disponible à l'adresse http://localhost:5984/_utils. Après authentification avec les identifiants définis lors du lancement du conteneur, l'utilisateur peut visualiser les bases de données, les documents et les paramètres du serveur. Bien que CouchDB soit principalement manipulé via une API REST, cette interface facilite l'exploration et la compréhension de la structure des données.

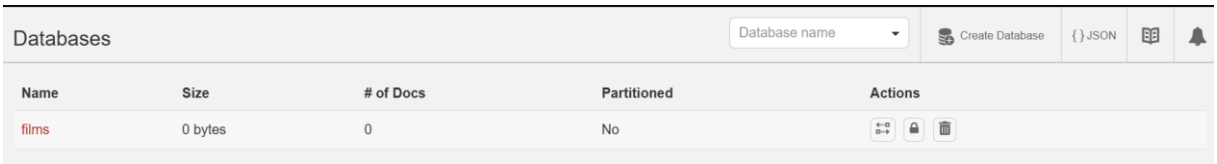


Cette capture illustre la création d'une base de données nommée **films** en utilisant une requête HTTP de type PUT. Dans CouchDB, une base de données est considérée comme une ressource REST, et sa création se fait simplement en envoyant une requête PUT vers une URL portant le nom de la base. La réponse

{"ok":true} indique que la base a été créée avec succès.



Cette capture montre l'insertion d'un document JSON dans la base de données *films*. L'opération est réalisée à l'aide d'une requête PUT ou POST, conformément au modèle REST de CouchDB. Le document est stocké sous forme de paires clé–valeur et ne nécessite aucun schéma prédéfini. Cette flexibilité permet d'insérer des documents hétérogènes, ce qui constitue une caractéristique essentielle des bases de données orientées documents.



Cette capture présente la visualisation des documents stockés dans la base *films* via l'interface graphique Fauxton. Chaque document possède un identifiant unique (_id) et un numéro de révision (_rev), utilisé par CouchDB pour gérer le versionnement. L'affichage confirme que les documents ont été correctement insérés et permet d'examiner leur structure JSON

```
PS C:\Users\chayma\Desktop\ing3\NoSql\Mongo replicasets\NoSql> docker run -d `
>> --name couchdbdemo `
>> -e COUCHDB_USER=youssef `
>> -e COUCHDB_PASSWORD=samir `
>> -p 5984:5984 `
>> couchdb
6d14dfd8de0722dd35bebed4e2a52b708f5164520233b916ce15e14b160a9d09
PS C:\Users\chayma\Desktop\ing3\NoSql\Mongo replicasets\NoSql>
```