TP 03 – ElasticSearch

Mise en place d’un moteur de recherche pour l’application MongoDB

---

1. Objectifs du TP

L’objectif de ce troisième travail pratique est d’intégrer **ElasticSearch** à l’application développée lors des TPs précédents, afin d’améliorer la fonctionnalité de recherche.

Les objectifs principaux étaient :

- Découvrir ElasticSearch et son fonctionnement
- Mettre en place ElasticSearch via Docker
- Indexer les données MongoDB dans ElasticSearch
- Implémenter une recherche textuelle performante
- Connecter ElasticSearch à l’application PHP existante

Ce TP s’inscrit dans la continuité des TPs précédents et vient compléter l’architecture globale de l’application.

---

2. Présentation d’ElasticSearch

ElasticSearch est un moteur de recherche **full-text** distribué, basé sur Apache Lucene.

Ses principales caractéristiques sont :

- recherche textuelle très rapide
- indexation des données
- tolérance aux fautes (fuzzy search)
- scoring et pertinence des résultats

Dans ce projet, ElasticSearch est utilisé uniquement pour la **recherche**, tandis que MongoDB reste la base de données principale.

---

3. Mise en place d’ElasticSearch avec Docker

ElasticSearch est ajouté à l’infrastructure via Docker Compose.

```
tpmongo-elasticsearch:
  image: elasticsearch:8.12.2
  container_name: tpmongo-elasticsearch
  ports:
    - "9200:9200"
  environment: ['CLI_JAVA_OPTS=-Xms512m -
Xmx512m','bootstrap.memory_lock=true','discovery.type=single-
node','xpack.security.enabled=false',
'xpack.security.enrollment.enabled=false', 'http.cors.allow-
```

```
origin=http://localhost:8180', 'http.cors.enabled=true']
  volumes:
    -   ./elasticsearch/synonyms:/usr/share/elasticsearch/config/synonyms
  networks:
    - local
  deploy:
    resources:
      limits:
        cpus: '1.0'
        memory: 2048M
      reservations:
        cpus: '0.50'
        memory: 1024M
```

Une interface graphique (ElasticVue) est également utilisée pour :

- visualiser les index
- tester les requêtes
- vérifier l'état du cluster

4. Connexion à ElasticSearch en PHP

La connexion à ElasticSearch est centralisée dans le fichier init.php :

```php
function getElasticClient(): ? ElasticClient
{
    if (!isset($_ENV['ELASTIC_HOST'])) {
        return null;
    }

    try {
        $client = ClientBuilder::create()
            ->setHosts([$_ENV['ELASTIC_HOST']])
            ->build();

        $response = $client->info();
        if (!empty($response)) {
            return $client;
        }

        throw new Exception("ElasticSearch non joignable.");
    } catch (Exception $e) {
        error_log("Erreur ElasticSearch : " . $e->getMessage());
        return null;
    }
}
```

Cette fonction :

- initialise le client ElasticSearch

- vérifie la disponibilité du service
- retourne null en cas d'erreur

5. Indexation des données

Les documents MongoDB (livres) sont indexés dans ElasticSearch avec les champs principaux :

- title
- author

Chaque document ElasticSearch utilise le même identifiant que MongoDB, ce qui permet de faire le lien entre les deux systèmes.

Cette indexation permet ensuite d'effectuer des recherches textuelles performantes sans impacter MongoDB.

6. Implémentation de la recherche

Lorsque l'utilisateur saisit une recherche, une requête ElasticSearch est exécutée.

```
$response = $es->search([
    'index' => $_ENV['ELASTIC_INDEX'],
    'body'  => [
        'query' => [
            'bool' => [
                'should' => [
                    [
                        'match' => [
                            'title' => [
                                'query'     => $q,
                                'fuzziness' => 'AUTO',
                                'boost'     => 2
                            ]
                        ]
                    ],
                    [
                        'match' => [
                            'author' => [
                                'query'     => $q,
                                'fuzziness' => 'AUTO',
                                'boost'     => 1
                            ]
                        ]
                    ]
                ]
            ]
        ]
    ]
]);
```

Caractéristiques de la recherche :

- tolérance aux fautes (fuzziness)
- pondération des champs (boost)
- recherche sur plusieurs champs

7. Intégration avec MongoDB

ElasticSearch ne stocke pas les documents complets affichés à l'utilisateur. Il retourne uniquement les identifiants des documents correspondants.

```
$ids = array_map(
    fn ($hit) => new MongoDB\BSON\ObjectId($hit['_id']),
    $response['hits']['hits']
);
```

Ces identifiants sont ensuite utilisés dans MongoDB pour récupérer les documents complets via un `$match`.

Cette approche permet :

- une séparation claire des responsabilités
- une cohérence totale des données
- une meilleure maintenabilité

8. Conclusion du TP3

Ce TP a permis de :

- comprendre le fonctionnement d'ElasticSearch
- mettre en place un moteur de recherche performant
- enrichir l'application existante sans la complexifier
- combiner efficacement MongoDB, Redis et ElasticSearch

L'application finale repose désormais sur une architecture complète et moderne :

- MongoDB pour le stockage
- Redis pour le cache
- ElasticSearch pour la recherche