МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

ЗВІТ

з лабораторної роботи №2
«Пошук документів за метаданими за допомогою Elasticsearch»

Виконав:
студент 4-го курсу, групи КП-91, спеціальності 121 – Інженерія програмного забезпечення
Власюк Сергій Петрович

Перевірив:
ас. каф. ПЗКС
Юсин Яків Олексійович

Київ 2023

Постановка задачі за варіантом

Реалізувати інформаційно-пошукову систему, що інтегрована з системою Elasticsearch та надає можливість пошуку збережених документів за метаданими. Робота з розробленим програмним забезпеченням повинне задовільнятися наступними вимогами:

- Для реалізації програмного забезпечення в ході лабораторної роботи може використовуватись будь-який стек (мова програмування, фреймворк і так далі) технологій.
- Для роботи з Elasticsearch допускається використання REST API напряму або будь-яких бібліотек-обгорток.
- Програмне забезпечення може мати будь-який з перелічених інтерфейсів користувача: консольний, веб, мобільний, настільний.
- Програмне забезпечення повинно надавати користувачу стандартний Create- Read-Delete (Update не є обов’язковим, може бути реалізованим за бажанням студента) інтерфейс для роботи з документами певної предметної галузі. Предметна галузь визначається відповідно до варіанту студента (див. далі).
- Документ повинен мати мінімум чотири поля різних типів (не рахуючи id), визначених відповідно до предметної галузі. Приклад: для предметної галузі «кіно», документ може містити наступні поля:
- назва фільму (тип keyword);
- дата виходу (тип date);
- актори (масив keyword);
- касові збори (тип integer).
- Використання типу text для повнотекстового пошуку в рамках цієї лабораторної роботи не допускається.
- Програмна система повинна надавати користувачу можливість фільтрації документів, що відображаються, за кожним із визначених полів.
- Для фільтрації необхідно використовувати запити-фільтри самого Elasticsearch. Зчитування з Elasticsearch всіх документів та їх фільтрація на стороні розробленого ПЗ не допускається.
- Обов’язкові запити-фільтри для реалізації: term (наприклад, для полів keyword), range (наприклад, для полів date, integer) + один складний запит,

визначений відповідно до варіанту студента (див. далі). За бажанням, студент може реалізувати інші типи запитів-фільтрів.

відповідно до варіанту №4

4	ігри	fuzzy

1)Реалізація програми

```
 const elasticsearch = require('elasticsearch');
const readline = require('readline');

const client = new elasticsearch.Client({
        host: 'localhost:9200',
});

const indexName = 'games';

const rl = readline.createInterface({
        input: process.stdin,
        output: process.stdout,
});

const indexMapping = {
        properties: {
        name: { type: 'keyword' },
        main_character: { type: 'keyword' },
        date: { type: 'date' },
        developers: { type: 'keyword' }
        }
};

function createDocument(callback) {
        rl.question('Name: ', (name) => {
        rl.question('Main character: ', (main_character) => {
        rl.question('Publication Date (YYYY-MM-DD): ', (date) => {
        rl.question('Developers: ', (developers) => {
```

```
                client.index({
                index: indexName,
                body: {
                name,
                main_character,
                date,
                developers: developers.split(',').map((keyword) => keyword.trim())
                }
                })
                .then((response) => {
                console.log('Document created:', response);
                callback();
                })
                .catch((error) => {
                console.error('Error creating document:', error);
                callback();
                });
        });
        });
        });
        });
}

function searchDocuments(callback) {
        rl.question('Search by (name, main_character, date, developers): ', (field) => {
        rl.question('Search value: ', (value) => {
        const filter =
        field === 'date' ? 'range' : 'term';
        const query =
        filter === 'range'
                ? { range: { [field]: { gte: value } } }
                : { term: { [field]: value } };
        client
        .search({
                index: indexName,
```

```
                body: {
                query: {
                bool: {
                filter: [query],
                },
                },
                },
        })
        .then((response) => {
                console.log('Search results:', JSON.stringify(response.hits.hits, null, 4));
                callback();
        })
        .catch((error) => {
                console.error('Error searching documents:', error);
                callback();
        });
        });
        });
}

function deleteDocument(callback) {
        rl.question('Document ID to delete: ', (id) => {
        client
        .delete({
        index: indexName,
        id,
        })
        .then((response) => {
        console.log('Document deleted:', response);
        callback();
        })
        .catch((error) => {
        console.error('Error deleting document:', error);
        callback();
        });
        });
```

```
}

function fuzzinessSearch(callback) {
        rl.question('Search by (name, main_character): ', (field) => {
        rl.question('Search value (fuzzy syntax): ', (value) => {
        client
        .search({
                index: indexName,
                body: {
                query: {
                fuzzy: {
                [field]: {
                        value: value,
                        fuzziness: 2
                }
                },
                },
                },
        })
        .then((response) => {
                console.log('Search results:', JSON.stringify(response.hits.hits, null, 4));
                callback();
        })
        .catch((error) => {
                console.error('Error searching documents:', error);
                callback();
        });
        });
        });
}

function getAllDocuments(callback) {
        client.search({
        index: indexName,
        body: {
        query: {
```

```
        match_all: {}
        }
        }
        }).then((response) => {
        console.log('All documents:', JSON.stringify(response.hits.hits, null, 4));
        callback();
        }).catch((error) => {
        console.error('Error getting all documents:', error);
        callback();
        });
}

function mainMenu() {
        rl.question('Choose action (1-create, 2-search, 3-delete, 4-fuzzy, 5-all): ', (action) => {
        if (action === 'create' || action === '1') {
        createDocument(mainMenu);
        } else if (action === 'search' || action === '2') {
        searchDocuments(mainMenu);
        } else if (action === 'delete' || action === '3') {
        deleteDocument(mainMenu);
        } else if (action === 'fuzziness' || action === '4') {
        fuzzinessSearch(mainMenu);
        } else if (action === 'get all' || action === '5') {
        getAllDocuments(mainMenu);
        } else {
        console.error('Invalid action');
        rl.close();
        }
        });
}


client.indices.exists({ index: indexName }).then((exists) => {
        if (!exists) {
        createIndex();
        } else {
```

```
        mainMenu();
        }
}).catch((error) => {
        console.error('Error checking index existence:', error);
});

function createIndex() {
        client.indices.create({
        index: indexName,
        body: {
        mappings: indexMapping
        }
        }).then((response) => {
        console.log('Index created:', response);
        mainMenu();
        }).catch((error) => {
        console.error('Error creating index:', error);
        });
}
```

Результати виконання роботи

```
PS C:\Users\Haylevel_SV\Desktop\search\lab2> node .\1.js
Choose action (1-create, 2-search, 3-delete, 4-fuzzy, 5-all): 5
All documents: [
    {
        "_index": "games",
        "_id": "F63zVYcBirBL2QRUr5RA",
        "_score": 1,
        "_source": {
            "name": "Stalkeer",
            "main_character": "Strelok",
            "date": "2008-03-09",
            "developers": [
                "Anton",
                "Serhii",
                "Masha"
            ]
        }
    }
]
Choose action (1-create, 2-search, 3-delete, 4-fuzzy, 5-all): 1
Name: WOT
Main character: Tanks
Publication Date (YYYY-MM-DD): 2010-02-03
Developers: Victor, Kislyi, Anton, Pankov
Document created: {
  _index: 'games',
  _id: 'GK0vVocBirBL2QRUqpST',
  _version: 1,
  result: 'created',
  _shards: { total: 2, successful: 1, failed: 0 },
  _seq_no: 1,
  _primary_term: 1
}
Choose action (1-create, 2-search, 3-delete, 4-fuzzy, 5-all): 2
Search by (name, main_character, date, developers): name
Search value: WOT
Search results: [
    {
        "_index": "games",
        "_id": "GK0vVocBirBL2QRUqpST",
```

```
        "_source": {
            "name": "WOT",
            "main_character": "Tanks",
            "date": "2010-02-03",
            "developers": [
                "Victor",
                "Kislyi",
                "Anton",
                "Pankov"
            ]
        }
    }
]
Choose action (1-create, 2-search, 3-delete, 4-fuzzy, 5-all): 4
Search by (name, main_character): main_character
Search value (fuzzy syntax): sknat
Search results: []
Choose action (1-create, 2-search, 3-delete, 4-fuzzy, 5-all): 4
Search by (name, main_character): main_character
Search value (fuzzy syntax): Tands
Search results: [
    {
        "_index": "games",
        "_id": "GK0vVocBirBL2QRUqpST",
        "_score": 0.55451775,
        "_source": {
            "name": "WOT",
            "main_character": "Tanks",
            "date": "2010-02-03",
            "developers": [
                "Victor",
                "Kislyi",
                "Anton",
                "Pankov"
            ]
        }
    }
]
```

Висновки

В ході виконання даної лабораторної роботи, я ознайомився з пошуком документів за метаданими за допомогою Elasticsearch. Розробка програмного забезпечення виконувалась з використанням мови програмування JavaScript. Було реалізовано інформаційно-пошукову систему, що інтегрована з системою Elasticsearch та надає можливість пошуку збережених документів за метаданими.