

QUẢN LÝ QUY TRÌNH PHẦN MỀM

PROOF OF CONCEPT

Mạng xã hội chia sẻ công thức nấu ăn

GVHD: TS. Ngô Huy Biên

SVTH: Nhóm 3 (22HCB_LT)



Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa Học Tự Nhiên TP HCM

MỤC LỤC

Nội dung

1	Mục tiêu	3
1.1	Tổng quan.....	3
1.2	Vấn đề	3
1.3	Giải pháp	4
2	Tích hợp Elasticsearch với Strapi	4
2.1	Môi trường và cài đặt cấu hình cần được thiết lập trước.	4
2.2	Tiến hành thực hiện.....	4
	Bước 1: Cấu hình chứng chỉ để kết nối với Elasticsearch	4
	Bước 2: Thiết lập kết nối với Elasticsearch.....	5
	Bước 3: Thêm/xóa dữ liệu đã được đánh index.....	6
	Bước 4: Tạo các request để gửi đến elasticsearch một cách bất đồng bộ	7
	Bước 5: Thiết lập một cron job để thực hiện cập nhật dữ liệu bất đồng bộ	8
	Bước 6: Cung cấp API search bằng elasticsearch.....	10
	Bước 7: Cấp quyền truy cập công khai cho API search	13
3	Kiểm thử và kiểm tra kết quả.....	13

1 Mục tiêu

1.1 Tổng quan

- Thiếu nền tảng chia sẻ công thức nấu ăn chuyên biệt: hiện nay, không có một nền tảng chia sẻ công thức nấu ăn chuyên biệt cho người yêu thích nấu ăn và thực hiện công thức một cách dễ dàng:
- Chưa có ứng dụng nào có thể đáp ứng nhu cầu mua nguyên liệu: hiện nay chưa có ứng dụng nào có thể cung cấp cho người dùng chức năng có thể mua nguyên liệu từ những công thức nấu ăn tìm được.
- Khó khăn trong tìm kiếm công thức nấu ăn: người yêu thích nấu ăn thường phải tốn nhiều thời gian để tìm kiếm các công thức cụ thể theo nhu cầu của họ, bất kể là theo tên món, thành phần, hoặc thời gian chuẩn bị.

1.2 Vấn đề

- Đối với một trang mạng xã hội chia sẻ công thức nấu ăn thì chức năng khó nhất và đòi hỏi phải hoạt động hiệu quả nhất chính là tính năng tìm kiếm. Khi mọi người đến với trang web thì điều đầu tiên một người dùng nghĩ đến chính là tìm kiếm món ăn mình muốn.
- Chính vì thế tính năng tìm kiếm sẽ có nhu cầu sử dụng cao nhất đòi hỏi hiệu năng phải ở mức tốt nhất và được tối ưu một cách hiệu quả.

1.3 Giải pháp

- Cần phải đánh index tất cả dữ liệu có thể được tìm kiếm. Elasticsearch là một giải pháp tốt cho việc này. Nó hỗ trợ việc đánh index dữ liệu và cung cấp các restful API hỗ trợ việc tìm kiếm và CRUD dữ liệu một cách nhanh chóng và tiện lợi.

2 Tích hợp Elasticsearch với Strapi

2.1 Môi trường và cài đặt cấu hình cần được thiết lập trước.

- Tài liệu này sẽ nhằm đến việc trình bày cách tích hợp Elasticsearch vào một dự án back-end được tạo bằng Strapi. Vì thế, tài liệu này sẽ không đi chi tiết vào hướng dẫn cài đặt các môi trường, ứng dụng và các cấu hình cần thiết cho việc tích hợp.
- Các sự chuẩn bị cho việc tích hợp bao gồm:
 1. Tải và cấu hình Elasticsearch.
 2. Hệ quản trị cơ sở dữ liệu MySQL. Tạo 1 database để lưu trữ dữ liệu mà người dùng muốn.
 3. Tạo một dự án back-end bằng Strapi. Tạo một collection blog-post với các trường dữ liệu đơn giản để lưu dữ liệu và một collection Search-indexing-requests lưu các request CRUD dữ liệu gửi đến elasticsearch, sau đó thực hiện việc cập nhật dữ liệu với Elasticsearch một cách bất đồng bộ.

2.2 Tiến hành thực hiện

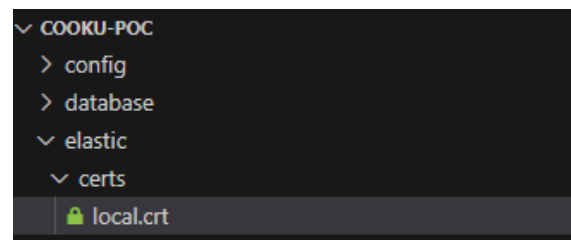
Bước 1: Cấu hình chúng chỉ để kết nối với ElasticSearch

- Tạo một thư mục **elastic** trong source code để chứa mã nguồn cho elasticsearch.

```
PS D:\cooku-poc> mkdir elastic
```

- Tạo một thư mục **certs** để chứa certificate kết nối với elasticsearch. Sau đó, sao chép tệp **http_ca.crt** trong **..\elasticsearch-8.11.0\config\certs** vào thư mục vừa tạo và đổi tên thành **local.crt**.

```
PS D:\cooku-poc> mkdir elastic/certs
```

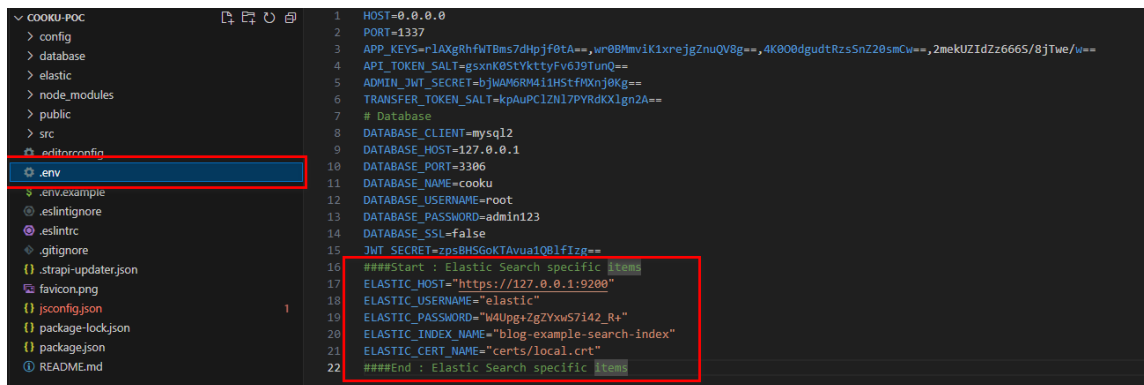


Bước 2: Thiết lập kết nối với ElasticSearch

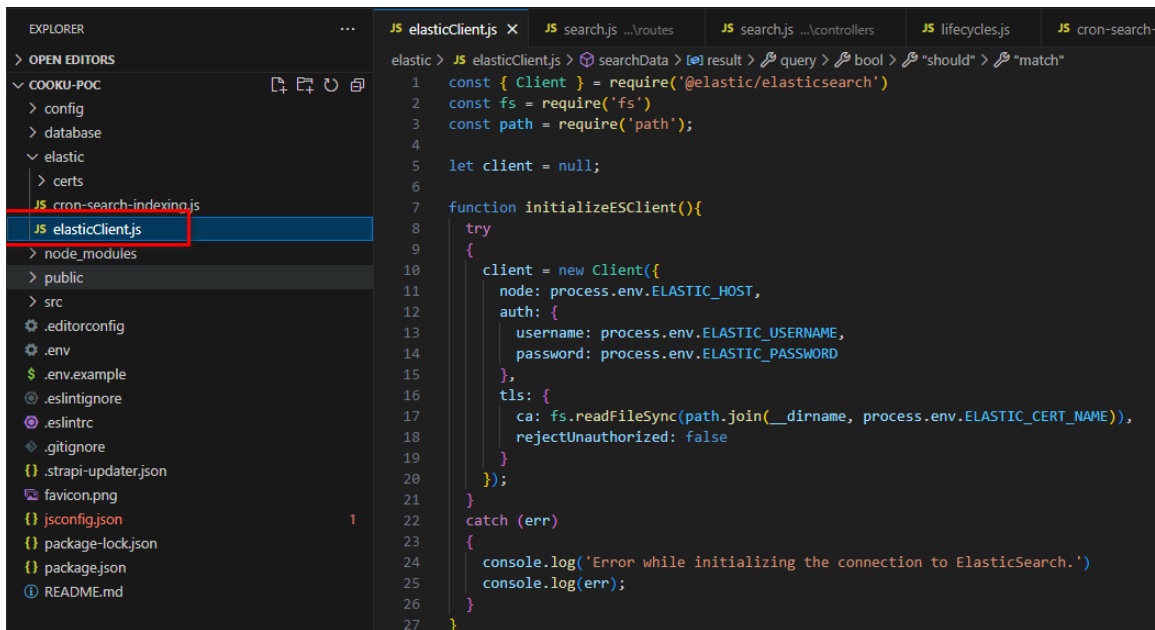
- Cài đặt thư viện **@elastic/elasticsearch** để hỗ trợ việc kết nối và tương tác với elasticsearch.

```
PS D:\cooku-poc> npm install @elastic/elasticsearch
```

- Cấu hình kết nối đến elasticsearch.

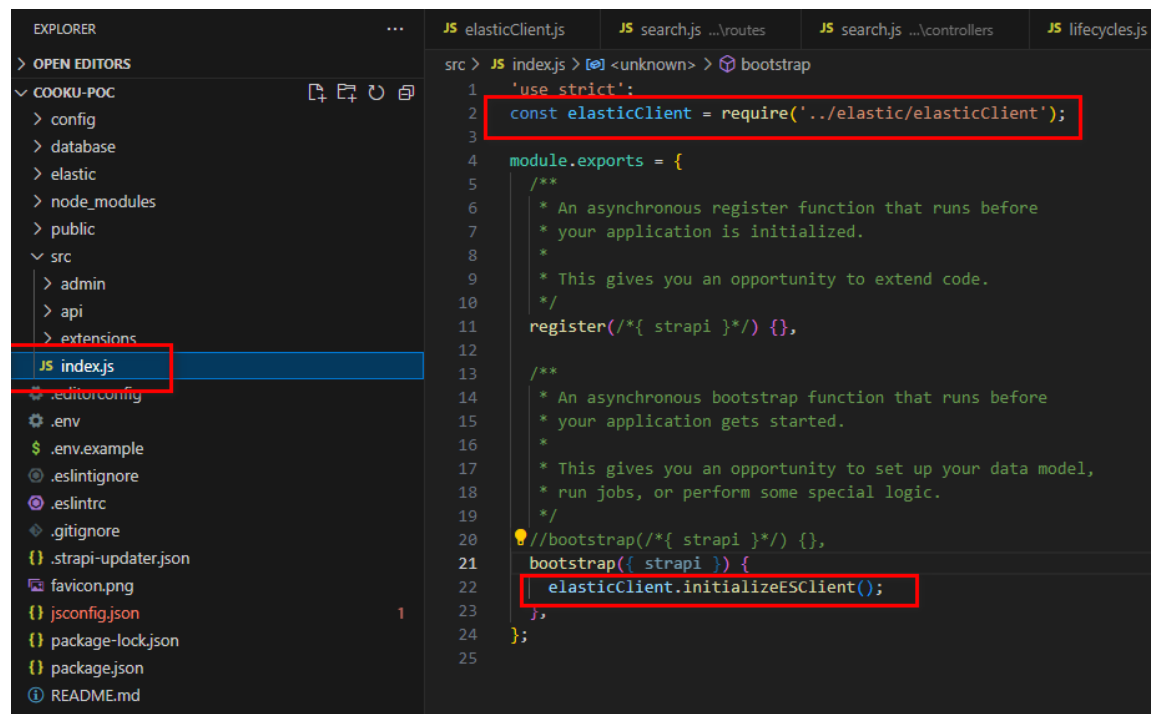


- Đoạn code sau sẽ thiết lập 1 kết nối đến elasticsearch khi start dự án back-end Strapi. Tạo tệp **elasticClient.js** trong thư mục **elastic** để chứa đoạn mã.



```
elastic > JS elasticClient.js > searchData > result > query > bool > should > match
1  const { Client } = require('@elastic/elasticsearch')
2  const fs = require('fs')
3  const path = require('path');
4
5  let client = null;
6
7  function initializeESClient(){
8    try
9    {
10     client = new Client({
11       node: process.env.ELASTIC_HOST,
12       auth: {
13         username: process.env.ELASTIC_USERNAME,
14         password: process.env.ELASTIC_PASSWORD
15       },
16       tls: {
17         ca: fs.readFileSync(path.join(__dirname, process.env.ELASTIC_CERT_NAME)),
18         rejectUnauthorized: false
19       }
20     });
21   }
22   catch (err)
23   {
24     console.log('Error while initializing the connection to Elasticsearch.')
25     console.log(err);
26   }
27 }
```

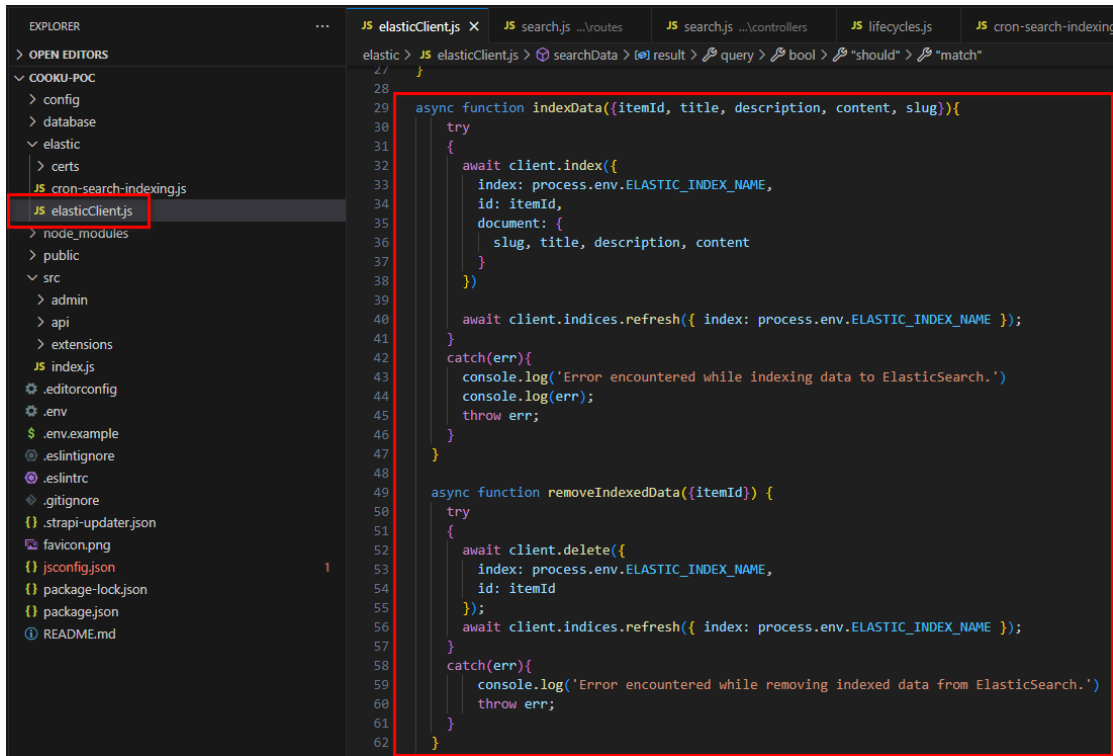
- Gọi hàm **initializeESClient** ở trong tệp **index.js** để thiết lập kết nối.



```
src > JS index.js > <unknown> > bootstrap
1  'use strict';
2  const elasticClient = require('../elastic/elasticClient');
3
4  module.exports = {
5    /**
6     * An asynchronous register function that runs before
7     * your application is initialized.
8     * This gives you an opportunity to extend code.
9     */
10   register(/*{ strapi }*/) {},
11
12   /**
13    * An asynchronous bootstrap function that runs before
14    * your application gets started.
15    * This gives you an opportunity to set up your data model,
16    * run jobs, or perform some special logic.
17    */
18   //bootstrap(/*{ strapi }*/) {},
19   bootstrap({ strapi }) {
20     elasticClient.initializeESClient();
21   },
22 };
23
24
25
```

Bước 3: Thêm/xóa dữ liệu đã được đánh index.

- Thêm đoạn code sau vào tệp **elasticClient.js**, nó sẽ hỗ trợ gọi hàm để thêm dữ liệu và xóa dữ liệu đã đánh index vào elasticsearch bằng đối tượng **client** của thư viện **@elastic/elasticsearch**



```
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

async function indexData({itemId, title, description, content, slug}){
  try
  {
    await client.index({
      index: process.env.ELASTIC_INDEX_NAME,
      id: itemId,
      document: {
        slug, title, description, content
      }
    })

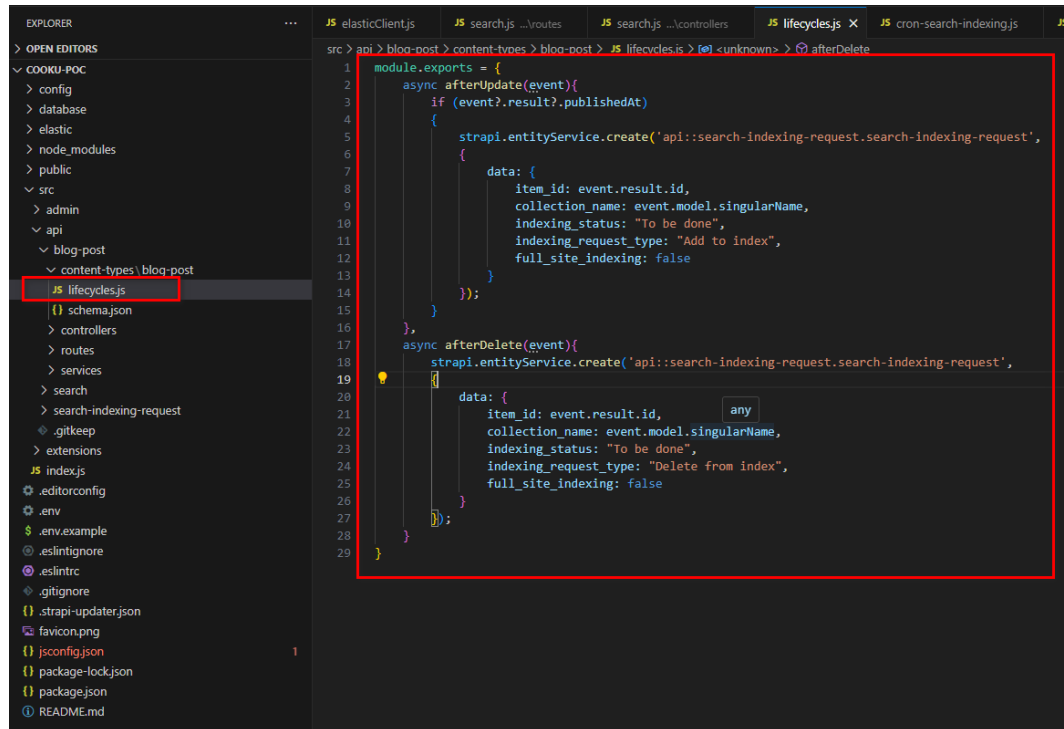
    await client.indices.refresh({ index: process.env.ELASTIC_INDEX_NAME });
  }
  catch(err){
    console.log('Error encountered while indexing data to ElasticSearch.')
    console.log(err);
    throw err;
  }
}

async function removeIndexedData({itemId}) {
  try
  {
    await client.delete({
      index: process.env.ELASTIC_INDEX_NAME,
      id: itemId
    });
    await client.indices.refresh({ index: process.env.ELASTIC_INDEX_NAME });
  }
  catch(err){
    console.log('Error encountered while removing indexed data from ElasticSearch.')
    throw err;
  }
}
```

Bước 4: Tạo các request để gửi đến elasticsearch một cách bất đồng bộ

- Việc gọi trực tiếp hàm thêm và xóa dữ liệu ở bước 3 thường xuyên mỗi khi thêm một entry vào collection sẽ dẫn đến vấn đề một lượng lớn request sẽ được gửi đến elasticsearch và elasticsearch có thể sẽ chậm phản hồi dẫn đến việc thêm hoặc xóa dữ liệu cũng sẽ bị ảnh hưởng làm giảm hiệu năng của ứng dụng. Vì thế, ta cần một collection khác để chứa các request gửi đến elasticsearch và thực hiện việc cập nhật dữ liệu với elasticsearch một cách bất đồng bộ.

- Mỗi khi CRUD một entry trong collection **blog-post** một request sẽ được thêm vào **Search-indexing-requests**. Thêm đoạn code sau vào tệp **lifecycle.js** trong thư mục **blog-post**:



```
1 module.exports = {
2   async afterUpdate(event){
3     if (event?.result?.publishedAt)
4     {
5       strapi.entityService.create('api::search-indexing-request.search-indexing-request',
6       {
7         data: {
8           item_id: event.result.id,
9           collection_name: event.model.singularName,
10          indexing_status: "To be done",
11          indexing_request_type: "Add to index",
12          full_site_indexing: false
13        }
14      });
15    }
16  },
17  async afterDelete(event){
18    strapi.entityService.create('api::search-indexing-request.search-indexing-request',
19    {
20      data: {
21        item_id: event.result.id,
22        collection_name: event.model.singularName,
23        indexing_status: "To be done",
24        indexing_request_type: "Delete from index",
25        full_site_indexing: false
26      }
27    });
28  }
29 }
```

Bước 5: Thiết lập một cron job để thực hiện cập nhật dữ liệu bất đồng bộ

- Tạo tệp **cron-search-indexing.js** trong thư mục **elastic** và thêm đoạn code bên dưới vào. Đoạn code này sẽ lấy tất cả các request với status là To be done trong collection **Search-indexing-requests**. Sau đó, dùng id để tìm kiếm entry trong collection **blog-post** để lấy dữ liệu. Tiếp theo gọi hàm đã được cài đặt ở bước 3 để gửi request đến elasticsearch và cập nhật status của request là Done sau khi đã hoàn thành.


```

1 const { indexData, removeIndexedData } = require('../elasticClient');
2
3 module.exports = {
4   performIndexingForSearch: async ({ strapi }) => {
5
6     const recs = await strapi.entityService.findMany('api::search-indexing-request.search-indexing-request', {
7       filters: { indexing_status: 'To be done' },
8     });
9
10    for (let r=0; r< recs.length; r++)
11    {
12      const col = recs[r].collection_name;
13
14      if (recs[r].item_id)
15      {
16        if (recs[r].indexing_type !== "Delete from index")
17        {
18          const api = 'api::' + col + '::' + col;
19          const item = await strapi.entityService.findOne(api, recs[r].item_id);
20          const indexItemId = col + "::" + item.id;
21          const {title, description, content, slug} = item;
22          await indexData({itemId: indexItemId, title, description, content, slug})
23          await strapi.entityService.update('api::search-indexing-request.search-indexing-request', recs[r].id, {
24            data: {
25              'indexing_status': 'Done'
26            }
27          });
28        }
29        else
30        {
31          const indexItemId = col + '::' + recs[r].item_id;
32          await removeIndexedData(indexItemId: indexItemId);
33          await strapi.entityService.update('api::search-indexing-request.search-indexing-request', recs[r].id, {
34            data: {
35              'indexing_status': 'Done'
36            }
37          });
38        }
39      }
40      else
41      {
42        // TBD : Code to index the entire collection.
43      }
44    }
45  }
46 }

```

- Tạo tệp **cron-task.js** trong thư mục **config** và thêm đoạn code bên dưới vào. Đoạn code này sẽ hẹn giờ thực hiện việc cập nhật bất đồng bộ dữ liệu giữa back-end Strapi và elasticsearch với thời gian được cài đặt là mỗi phút 1 chạy 1 lần.

```

1 const { performIndexingForSearch } = require('../elastic/cron-search-indexing');
2
3 module.exports = {
4   performIndexingForSearch: {
5     task: async({strapi}) => {
6       return await performIndexingForSearch({strapi});
7     },
8     options: {
9       rule: "* * * * *",
10    },
11  }
12 }

```

- Thêm đoạn code bên dưới vào tệp **server.js** để ứng dụng cho phép chạy cron task cập nhật dữ liệu bất đồng bộ với elasticsearch đã cài đặt ở phía trên.

```

1 const cronTasks = require('./cron-tasks');
2
3 module.exports = ({ env }) => ({
4   host: env('HOST', '0.0.0.0'),
5   port: env.int('PORT', 1337),
6   app: {
7     keys: env.array('APP_KEYS'),
8   },
9   webhooks: {
10    populateRelations: env.bool('WEBHOOKS_POPULATE_RELATIONS', false),
11  },
12  cron: {
13    enabled: true,
14    tasks: cronTasks
15  }
16 });

```

Bước 6: Cung cấp API search bằng elasticsearch

- Thêm đoạn code bên dưới vào tệp **elasticClient.js** trong thư mục **elastic**.

Đoạn code này sẽ tạo và gọi search request được cung cấp bởi thư viện **@elastic/elasticsearch** đến elasticsearch để thực hiện việc tìm kiếm từ khóa mà người dùng đưa vào.

```

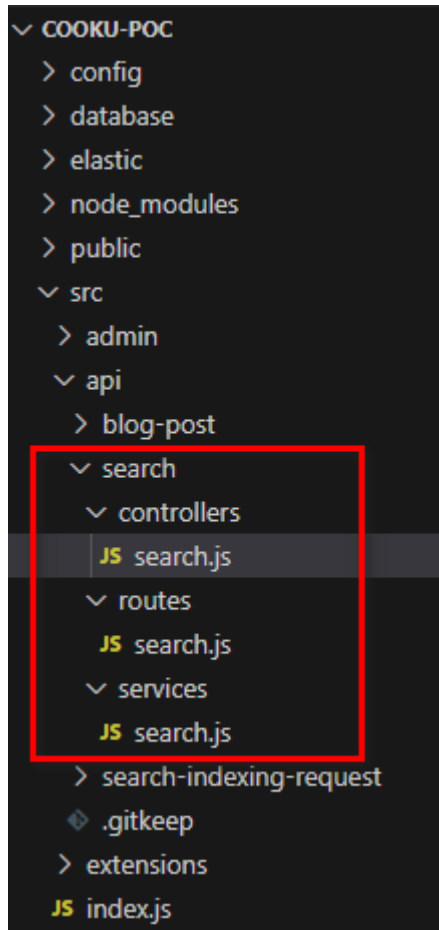
59 console.log('Error encountered while removing indexed data from ElasticSearch.')
60 throw err;
61 }
62 }
63
64 async function searchData(searchTerm){
65   try {
66     const result= await client.search({
67       index: process.env.ELASTIC_INDEX_NAME,
68       size: 100,
69       query: {
70         bool: {
71           "should" : [
72             {
73               "match" : {
74                 "content" : searchTerm
75               }
76             },
77             {
78               "match" : {
79                 "title" : searchTerm
80               }
81             },
82             {
83               "match" : {
84                 "description" : searchTerm
85               }
86             }
87           ]
88         }
89       });
90     return result;
91   } catch(err) {
92     console.log('Search : elasticClient.searchData : Error encountered while making a search request to ElasticSearch.')
93     throw err;
94   }
95 }

```

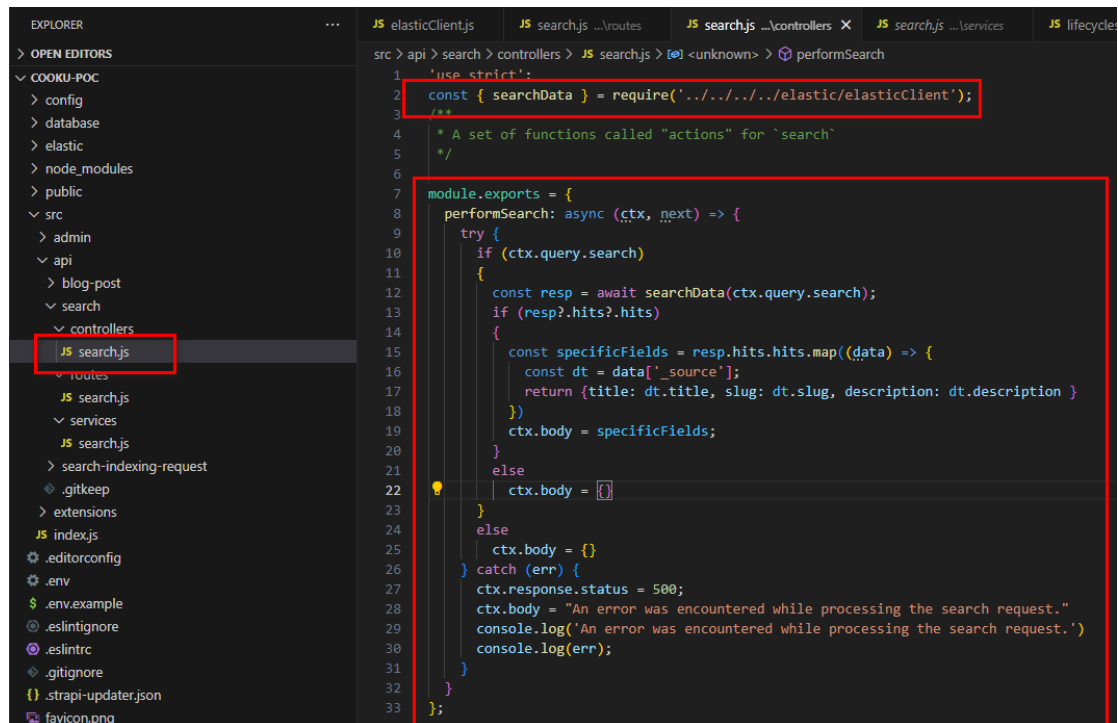
- Tạo một bộ api search trong thư mục src bằng các câu lệnh sau:

```
PS D:\cooku-poc> npx strapi generate api  
? API name search  
? Is this API for a plugin? (Y/n) n
```

- Kết quả 1 bộ tệp api sẽ được tạo như sau:



- Thêm đoạn code bên dưới vào tệp **search.js** trong thư mục **controllers**. Nó sẽ gọi hàm gửi search request đến elasticsearch đã được cài đặt ở trên và trả các kết quả đã tìm được về.

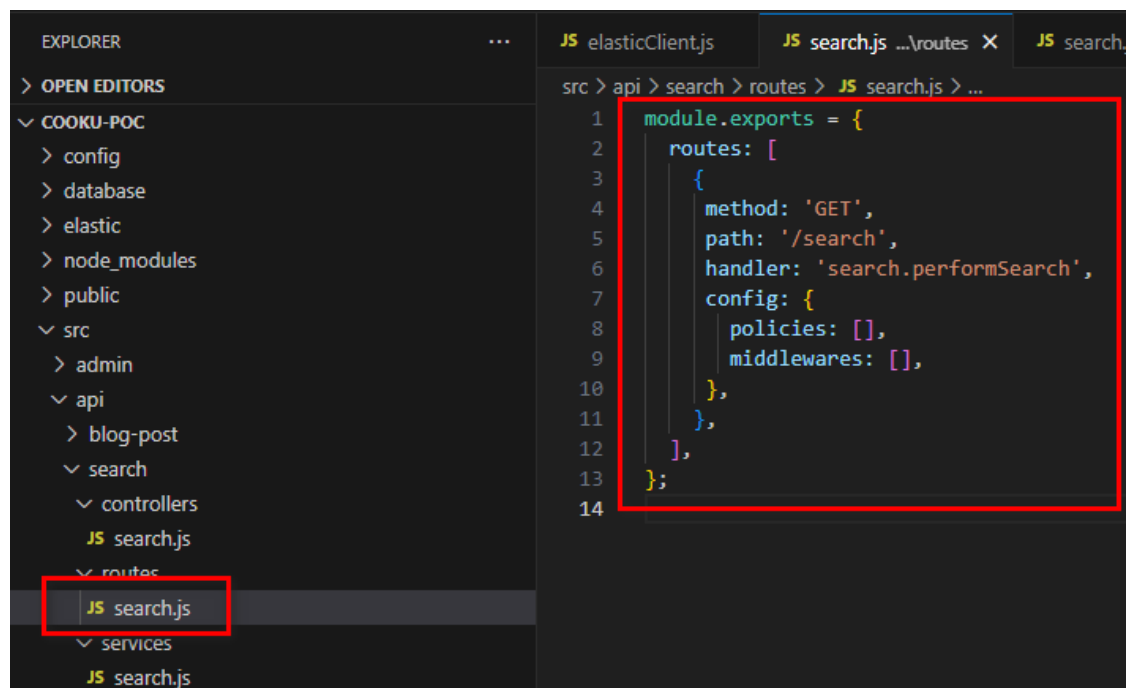


The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. In the Explorer, the file `JS searchjs` under the `controllers` directory is selected. In the Editor, the `performSearch` function is implemented in `search.js`. The code includes a `use strict` directive, a `const searchData` import, and a `performSearch` function that handles the search request by querying the database and returning the results.

```
src > api > search > controllers > JS searchjs > [unknown] > performSearch
1 'use strict';
2 const { searchData } = require('../../../../elastic/elasticClient');
3 /**
4  * A set of functions called "actions" for "search"
5  */
6
7 module.exports = {
8   performSearch: async (ctx, next) => {
9     try {
10       if (ctx.query.search) {
11         const resp = await searchData(ctx.query.search);
12         if (resp?.hits?.hits) {
13           const specificFields = resp.hits.hits.map((data) => {
14             const dt = data['_source'];
15             return {title: dt.title, slug: dt.slug, description: dt.description }
16           });
17           ctx.body = specificFields;
18         }
19       } else {
20         ctx.body = {};
21       }
22     } catch (err) {
23       ctx.response.status = 500;
24       ctx.body = "An error was encountered while processing the search request."
25       console.log('An error was encountered while processing the search request.')
26       console.log(err);
27     }
28   }
29 };
30
31
32
33
```

- Thêm đoạn code bên dưới vào tệp **search.js** trong thư mục **routes**. Đoạn code này tạo 1 api trong backend Strapi như sau:

<http://localhost:1337/api/search?search=looking-for-what?>

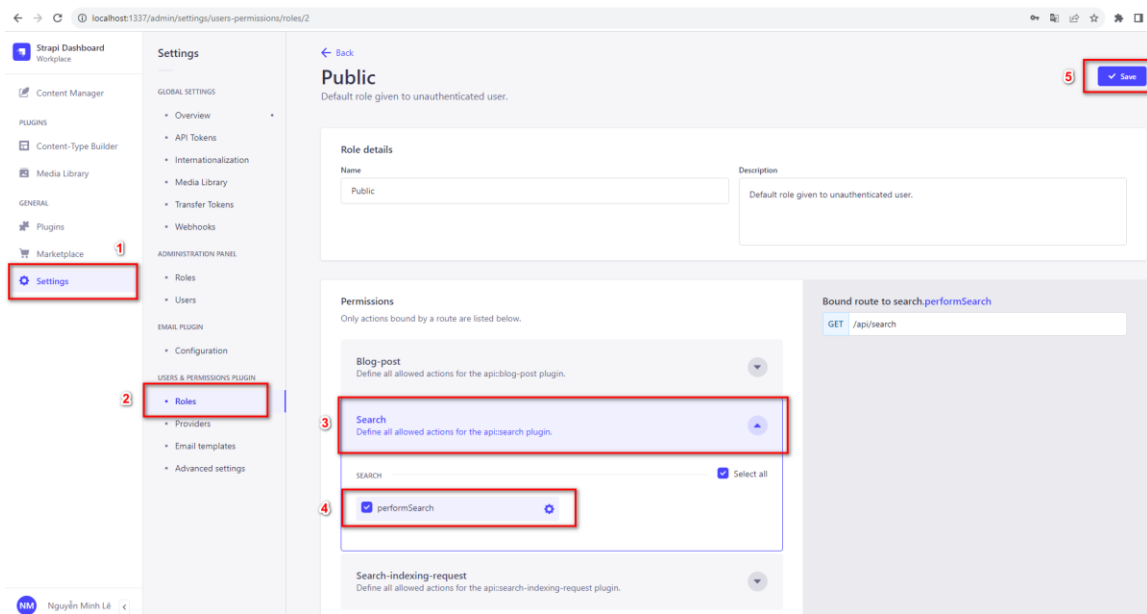


The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. In the Explorer, the file `JS searchjs` under the `routes` directory is selected. In the Editor, the `routes` array is implemented in `search.js`. The code defines a single route with the `GET` method, the `/search` path, and the `search.performSearch` handler.

```
src > api > search > routes > JS searchjs > ...
1 module.exports = {
2   routes: [
3     {
4       method: 'GET',
5       path: '/search',
6       handler: 'search.performSearch',
7       config: {
8         policies: [],
9         middlewares: [],
10       },
11     },
12   ],
13 };
14
```

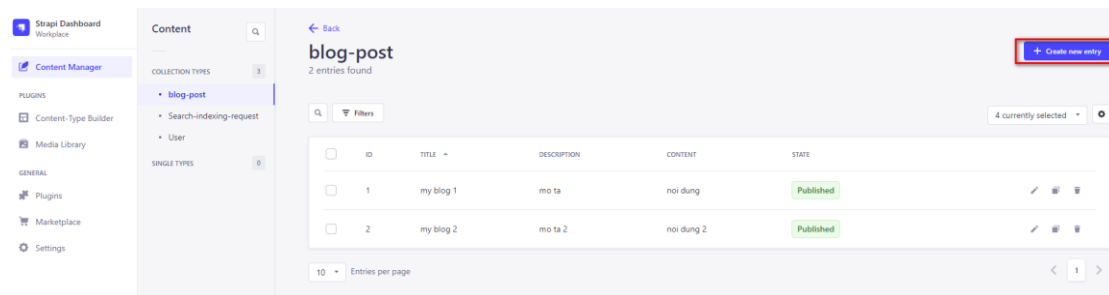
Bước 7: Cấp quyền truy cập công khai cho API search

- Để search api vừa tạo có thể được truy cập công khai bằng các ứng dụng front-end khác. Ta cần cấp quyền truy cập công khai cho API theo các bước như sau:

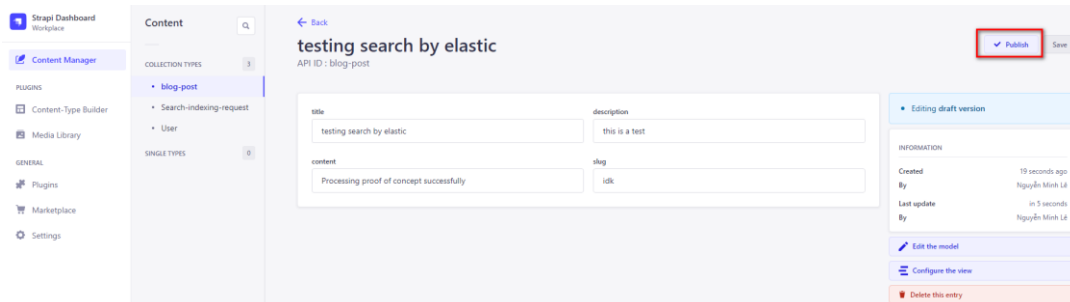
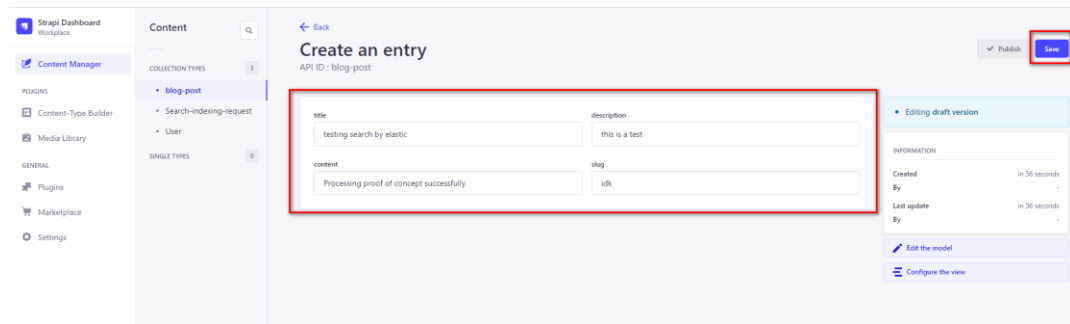


3 Kiểm thử và kiểm tra kết quả

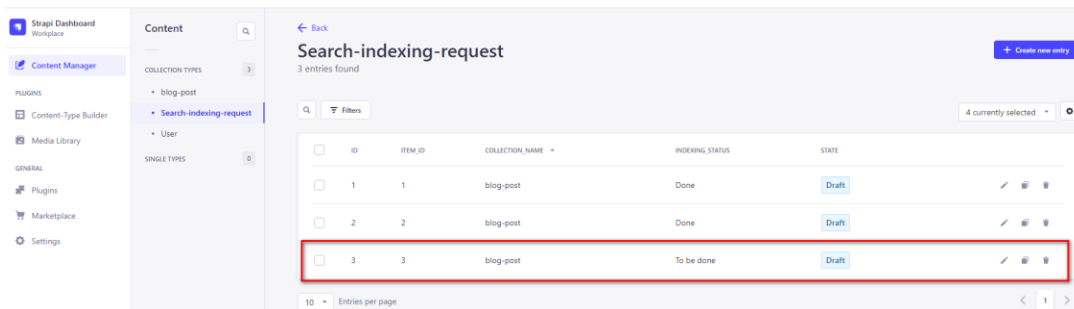
- Thêm một entry vào collection **blog-post**



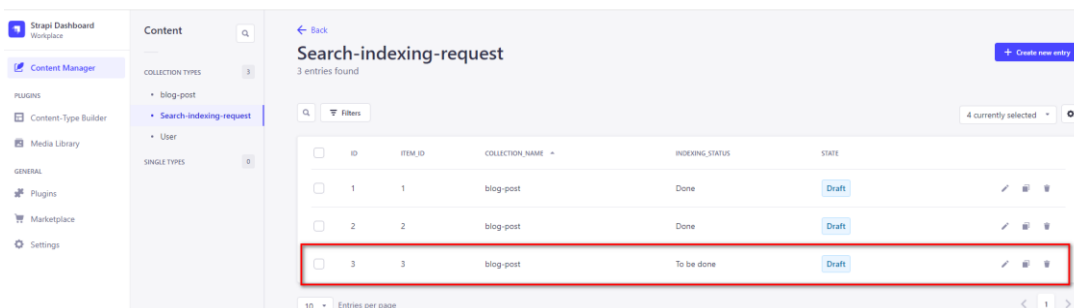
- Nhập thông tin các trường và nhấn nút **Save -> Publish**:



- Một entry được tự động thêm vào collection **Search-indexing-requests**



- Sau 1 phút dữ liệu được cập nhật sang elasticsearch và status được cập nhật thành **Done**.



- Thực hiện gọi search api và kết quả đã trả về entry vừa tạo ở collection **blog-post**

localhost:1337/api/search?search=test

```
[
  - {
    title: "testing search by elastic",
    slug: "idk",
    description: "this is a test"
  }
]
```

- Dữ liệu trong elasticsearch

Multi Elasticsearch Head | chrome-extension://cpmmilfkofoeimbmgiclohpodggeheim/elasticsearch-head/index.html

Elasticsearch New elasticsearch (https://localhost:9200/) (8.11.0) cluster health: yellow (1 of 2)

Overview Indices Browser Structured Query [+] Any Request [+]

Search blog-example-search-index (3 docs) for documents where:

must match_all + -

Search Output Results: JSON Number of Results: 10 Show query source

```
{
  "took": 2,
  "timed_out": false,
  "shards": {
    "total": 1,
    "successful": 1,
    "skipped": 0,
    "failed": 0
  },
  "hits": {
    "total": {
      "value": 3,
      "relation": "eq"
    },
    "max_score": 1,
    "hits": [
      {
        "index": "blog-example-search-index",
        "id": "blog-post:1",
        "score": 1,
        "source": {
          "slug": "no",
          "title": "my blog 1",
          "description": "mo ta",
          "content": "noi dung"
        }
      },
      {
        "index": "blog-example-search-index",
        "id": "blog-post:2",
        "score": 1,
        "source": {
          "slug": "no 2",
          "title": "my blog 2",
          "description": "mo ta 2",
          "content": "noi dung 2"
        }
      },
      {
        "index": "blog-example-search-index",
        "id": "blog-post:3",
        "score": 1,
        "source": {
          "slug": "idk",
          "title": "testing search by elastic",
          "description": "this is a test",
          "content": "Processing proof of concept successfully"
        }
      }
    ]
  }
}
```