

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

XÂY DỰNG WEBSITE QUẢN LÝ TÀI LIỆU SỬ DỤNG CÔNG NGHỆ ELASTICSEARCH

Giảng viên hướng dẫn : TS. Nguyễn Quốc Tuấn

Sinh viên thực hiện : Trương Tiến Đạt

Mã sinh viên : 211204032

Lớp : Công nghệ thông tin 2

Khóa : 62

Hà Nội - 2025

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

XÂY DỰNG WEBSITE QUẢN LÝ TÀI LIỆU SỬ DỤNG CÔNG NGHỆ ELASTICSEARCH

Giảng viên hướng dẫn : TS. Nguyễn Quốc Tuấn
Sinh viên thực hiện : Trương Tiến Đạt
Mã sinh viên : 211204032
Lớp : Công nghệ thông tin 2
Khóa : 62

Hà Nội - 2025

LỜI CẢM ƠN

Trải qua 4 năm rèn luyện và học tập tại trường Đại học Giao thông vận tải, bên cạnh sự nỗ lực rèn luyện, học tập và trau dồi kiến thức, em đã rất may mắn được các thầy, cô trong trường, đặc biệt là các thầy, cô trong khoa Công nghệ Thông tin tận tình chỉ dạy, trang bị cho em các kiến thức cần thiết và vô cùng quý giá để em có một nền tảng vững chắc cùng những tri thức quý báu cho sự nghiệp sau này.

Em xin chân thành cảm ơn Tiến sĩ Nguyễn Quốc Tuấn – giảng viên hướng dẫn của em. Thầy đã luôn tận tâm chỉ dẫn, hỗ trợ, góp ý kịp thời và theo sát quá trình thực hiện đồ án đã giúp đỡ em nghiên cứu và thành công thực hiện đồ án “Xây dựng website quản lý tài liệu sử dụng công nghệ Elasticsearch”.

Đồng thời em cũng xin trân trọng cảm ơn những bài giảng, kinh nghiệm quý báu cùng với sự nhiệt tình trong giảng dạy mà các thầy cô trong trường Đại học Giao thông Vận tải đã truyền đạt để em có thể áp dụng những kinh nghiệm, kỹ thuật và cách thức ấy trong việc xây dựng, hoàn thiện đồ án tốt nghiệp và cả sự nghiệp sau này.

Tuy nhiên, do thời gian có hạn nên em không thể phát huy hết những ý tưởng, khả năng hỗ trợ của ngôn ngữ và kỹ thuật lập trình vào đề tài. Trong quá trình xây dựng website em không thể tránh khỏi những sai sót, mong nhận được sự cảm thông và đóng góp của quý thầy cô và các bạn.

Em xin chân thành cảm ơn.

Hà Nội, ngày 29 tháng 05 năm 2025

Sinh viên thực hiện

Trương Tiến Đạt

MỤC LỤC

| | |
|---------------------------------------------------------------------------------|-----------|
| LỜI CẢM ƠN..... | 1 |
| MỤC LỤC..... | 2 |
| DANH MỤC BẢNG BIỂU..... | 4 |
| DANH MỤC HÌNH ẢNH..... | 5 |
| MỞ ĐẦU..... | 7 |
| CHƯƠNG 1: NGHIÊN CỨU CÔNG NGHỆ ELASTICSEARCH..... | 9 |
| 1.1 Giới thiệu..... | 9 |
| 1.1.1 Elasticsearch là gì..... | 9 |
| 1.1.2 Mô hình dữ liệu..... | 9 |
| 1.1.3 Mục đích..... | 9 |
| 1.1.4 Khả năng sử dụng..... | 10 |
| 1.1.5 Ưu điểm..... | 10 |
| 1.1.6 Nhược điểm..... | 11 |
| 1.1.7 Bảo mật..... | 11 |
| 1.2 Ngôn ngữ truy vấn..... | 11 |
| 1.3 Tổ chức dữ liệu và Index..... | 13 |
| 1.3.1 Tổ chức dữ liệu..... | 13 |
| 1.3.2 Index..... | 17 |
| 1.4 Backup, recovery, partition, sharding, replication, cluster..... | 18 |
| 1.4.1 Backup..... | 18 |
| 1.4.2 Recovery..... | 19 |
| 1.4.3 Partition..... | 20 |
| 1.4.4 Sharding..... | 20 |
| 1.4.5 Replication..... | 21 |
| 1.4.6 Cluster..... | 21 |
| CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG..... | 24 |

| | |
|-------------------------------------------------------------|-----------|
| 2.1 Bài toán..... | 24 |
| 2.2 Hướng giải quyết..... | 24 |
| 2.3 Phân tích và thiết kế yêu cầu..... | 25 |
| 2.3.1 Yêu cầu chức năng | 25 |
| 2.3.2 Biểu đồ phân rã chức năng và phân tích chức năng..... | 25 |
| 2.3.3 Đặc tả yêu cầu chức năng..... | 27 |
| 2.3.4 Biểu đồ tuần tự | 29 |
| 2.4 Thiết kế cơ sở dữ liệu | 35 |
| 2.4.1 Danh sách các bảng..... | 35 |
| 2.4.2 Mô hình cơ sở dữ liệu quan hệ..... | 39 |
| CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH..... | 40 |
| 3.1 Cài đặt chương trình..... | 40 |
| 3.1.1 Fontend..... | 40 |
| 3.1.2 Backend..... | 40 |
| 3.1.3 Cơ sở dữ liệu | 40 |
| 3.1.4 Một số hàm xử lý chính..... | 41 |
| 3.2 Giao diện chương trình..... | 43 |
| 3.2.1 Chức năng tài khoản..... | 43 |
| 3.2.2 Chức năng quản lý file | 45 |
| 3.2.3 Chức năng tìm kiếm toàn văn..... | 53 |
| 3.2.4 Chức năng quản lý tài khoản..... | 54 |
| KẾT LUẬN..... | 56 |
| TÀI LIỆU THAM KHẢO | 57 |

DANH MỤC BẢNG BIỂU

| | |
|----------------------------------------------|----|
| Bảng 2.1. Bảng mô tả chi tiết chức năng..... | 26 |
| Bảng 2.2. Bảng mô tả các tác nhân | 27 |
| Bảng 2.3. Bảng organization_unit..... | 35 |
| Bảng 2.4. Bảng employee..... | 35 |
| Bảng 2.5. Bảng file..... | 36 |
| Bảng 2.6. Bảng role..... | 37 |
| Bảng 2.7. Bảng role_detail..... | 38 |
| Bảng 2.8. Bảng user..... | 38 |

DANH MỤC HÌNH ẢNH

| | |
|--------------------------------------------------------------------------------------|----|
| Hình 1.1. Sơ đồ lấy dữ liệu của elasticsearch..... | 12 |
| Hình 1.2. Cấu trúc node bên trong một cluster | 14 |
| Hình 1.3. Câu lệnh truy vấn tìm kiếm của elasticsearch..... | 16 |
| Hình 1.4. Ảnh minh họa cách elasticsearch thực hiện truy vấn..... | 17 |
| Hình 1.5. Ảnh minh họa một cluster..... | 21 |
| Hình 1.6. Kiến trúc của Elasticsearch..... | 22 |
| Hình 2.1. Biểu đồ phân rã chức năng..... | 25 |
| Hình 2.2. Sơ đồ tổng quát use-case..... | 28 |
| Hình 2.3 Biểu đồ tuần tự đăng nhập..... | 29 |
| Hình 2.4. Biểu đồ tuần tự đăng kí | 29 |
| Hình 2.5. Biểu đồ tuần tự hiển thị danh sách tài khoản | 30 |
| Hình 2.6. Biểu đồ tuần tự tạo tài khoản mới..... | 30 |
| Hình 2.7. Biểu đồ tuần tự cập nhật thông tin tài khoản..... | 31 |
| Hình 2.8. Biểu đồ tuần tự xóa tài khoản..... | 31 |
| Hình 2.9. Biểu đồ tuần tự hiển thị danh sách file | 32 |
| Hình 2.10. Biểu đồ tuần tự tải file lên website..... | 32 |
| Hình 2.11. Biểu đồ tuần tự tải file về máy tính cá nhân..... | 33 |
| Hình 2.12. Biểu đồ tuần tự chia sẻ file | 33 |
| Hình 2.13. Biểu đồ tuần tự xóa file..... | 34 |
| Hình 2.14. Biểu đồ tuần tự tìm kiếm thông minh..... | 34 |
| Hình 2.15. Mô hình cơ sở dữ liệu..... | 39 |
| Hình 3.1. Hàm xác thực Authenticate và hàm tạo token đăng nhập generateJwtToken..... | 41 |
| Hình 3.2. Hàm cấu hình Elasticsearch..... | 42 |
| Hình 3.3. Hàm tìm kiếm dữ liệu trên Elasticsearch..... | 42 |
| Hình 3.4. Hàm UploadFile..... | 43 |
| Hình 3.6. Thư lấy mã xác thực email..... | 44 |

| | |
|------------------------------------------------------------------------------|----|
| Hình 3.7. Form điền thông tin để hoàn tất đăng ký..... | 45 |
| Hình 3.8. Giao diện quản lý file cá nhân..... | 45 |
| Hình 3.9. Giao diện quản lý file đơn vị..... | 46 |
| Hình 3.10. Giao diện chọn file để tải lên website | 46 |
| Hình 3.11. Giao diện sau khi chọn file tải lên..... | 47 |
| Hình 3.12. Giao diện tải file lên thành công..... | 47 |
| Hình 3.13. Giao diện tải file của trang đơn vị | 48 |
| Hình 3.14. Giao diện xem nội dung file..... | 48 |
| Hình 3.15. Thông báo không có quyền xóa file..... | 49 |
| Hình 3.16. Dialog xác nhận xóa file..... | 49 |
| Hình 3.17. Thông báo xóa file thành công..... | 50 |
| Hình 3.18. Thông báo không có quyền tải file..... | 50 |
| Hình 3.19. Kết quả sau khi tải file..... | 51 |
| Hình 3.20. Giao diện trang chia sẻ tài liệu..... | 51 |
| Hình 3.21. Dialog chọn file để chia sẻ cho người dùng khác | 52 |
| Hình 3.22. Thông báo chia sẻ thành công..... | 52 |
| Hình 3.23. File đã được chia sẻ với tài khoản của người dùng mới được thêm.. | 53 |
| Hình 3.24. Nội dung file vừa tải lên..... | 53 |
| Hình 3.25. Danh sách file có chứa từ khóa vừa tìm kiếm..... | 54 |
| Hình 3.26. Danh sách nhân viên..... | 54 |
| Hình 3.27. Form điền thông tin của nhân viên mới..... | 55 |
| Hình 3.28. Form thêm vai trò người dùng mới..... | 55 |

MỞ ĐẦU

Ngày nay, với việc công nghệ ngày càng phát triển nhanh chóng, mạnh mẽ và càng ngày càng trở nên phổ biến hơn với mọi người thì việc sử dụng những tài liệu mềm cùng với các thiết bị điện tử hiện đại phục vụ cho việc lưu trữ và quản lý tài liệu đóng vai trò hết sức quan trọng đối với các tổ chức, công ty và các doanh nghiệp lớn nhỏ.

Đầu tiên, việc sử dụng tài liệu mềm sẽ giảm thiểu đáng kể không gian và diện tích để lưu trữ so với việc sử dụng tài liệu cứng dạng giấy, chưa kể chi phí để đóng các kệ sách, giá sách cho việc lưu trữ một số lượng lớn tài liệu bản cứng là vô cùng lớn. Thứ hai là việc số hóa tài liệu dưới dạng mềm sẽ giúp cho việc bảo quản và lưu giữ cho tuổi thọ của những tài liệu truyền thống được lâu hơn. Thứ ba là dễ dàng mở rộng phạm vi cộng đồng người sử dụng nguồn tài nguyên thông tin của cơ quan thông tin, thư viện vì việc truyền tay nhau những tài liệu mềm chắc chắn luôn dễ dàng và hiệu quả hơn so với tài liệu bản cứng. Thứ tư là tiện ích trong việc truy xuất tìm kiếm thông tin ở bất kỳ đâu vào bất cứ thời điểm nào một cách nhanh chóng, dễ dàng. Thật vậy, nếu so với việc thao tác tìm kiếm tài liệu trên Internet với việc tìm và tra sách bằng cách đọc từng cuốn sách trong thư viện sẽ nhanh chóng hơn, tiết kiệm thời gian hơn và hiệu quả hơn rất nhiều. Cuối cùng là thuận lợi trong việc chia sẻ nguồn tài nguyên thông tin của thư viện với các thư viện khác.

Bên cạnh những ưu điểm trên, việc sử dụng và quản lý tài liệu mềm cũng có một số khuyết điểm, hạn chế gây ra những khó khăn và thử thách không hề nhỏ đối với các công ty, tổ chức và doanh nghiệp trong việc quản lý các dạng tài liệu mềm. Đầu tiên là chi phí cho việc đầu tư máy móc, trang thiết bị hiện đại cho việc quản lý và lưu trữ một lượng lớn tài liệu là khá đắt đỏ. Tiếp theo là nguy cơ về việc tài liệu chính thống có thể bị sửa đổi và sao chép rồi phát tán trái phép nhằm b López méo sự thật và lừa truyền những thông tin giả gây ảnh hưởng tới mọi người. Và cuối cùng là vấn đề đồng bộ cũng như là bảo mật những hồ sơ, tài liệu bí mật khỏi bị rò rỉ và những cuộc tấn công nhằm mục đích chiếm đoạt đang là khó khăn cũng như là thách thức cho hàng ngàn các doanh nghiệp, công ty, tổ chức trên toàn cầu. Do đó các công ty, doanh nghiệp cần phải phát huy tối đa các ưu điểm và hạn chế nhược điểm xuống mức tối thiểu của việc quản lý tài liệu mềm để có thể tối ưu hóa các lợi ích cho doanh nghiệp của mình.

Hiện nay, có rất nhiều trang web và website giúp cho việc quản lý và lưu trữ tài liệu của doanh nghiệp trở nên đơn giản và bảo mật hơn. Tuy nhiên, bên cạnh những trang web uy tín, cũng xuất hiện nhiều website lừa đảo nhằm trực lợi và thu thập trái phép thông tin cá nhân của người dùng.

Để tăng cường độ tin cậy cũng như tính hữu ích của các website quản lý tài liệu, đồ án của em tập trung xây dựng một website uy tín, bảo mật, dễ sử dụng và thân thiện với người dùng trên nền tảng web website quản lý tài liệu - Datalink .

CHƯƠNG 1: NGHIÊN CỨU CÔNG NGHỆ

ELASTICSEARCH

1.1 Giới thiệu

1.1.1 Elasticsearch là gì

Elasticsearch là một công cụ tìm kiếm dựa trên nền tảng Apache Lucene. Nó cung cấp một bộ máy tìm kiếm dạng phân tán, có đầy đủ công cụ với một giao diện web HTTP có hỗ trợ dữ liệu JSON. Elasticsearch được phát triển bằng Java và được phát hành dạng nguồn mở theo giấy phép Apache.

Elasticsearch thực chất hoạt động như 1 web server, có khả năng tìm kiếm nhanh chóng thông qua giao thức RESTful. Elasticsearch là 1 hệ thống phân tán và có khả năng mở rộng tuyệt vời. Bổ sung thêm node cho nó là nó tự động mở rộng cho bạn.

1.1.2 Mô hình dữ liệu

Elasticsearch là một hệ cơ sở dữ liệu NoSQL. Cơ sở dữ liệu NoSQL là cơ sở dữ liệu được xây dựng dành riêng cho mô hình dữ liệu và có sơ đồ linh hoạt để xây dựng các website hiện đại. Cơ sở dữ liệu NoSQL được công nhận rộng rãi vì khả năng dễ phát triển, chức năng cũng như hiệu năng ở quy mô lớn. Các Cơ sở dữ liệu này sử dụng nhiều mô hình dữ liệu đa dạng, trong đó có văn bản, đồ thị, khóa – giá trị, trong bộ nhớ và tìm kiếm.

Elasticsearch là một hệ document store database. Document stores, hay còn gọi là hệ thống database hướng document (document-oriented) là một kiểu database mà ở đó data được tổ chức một cách tự do không theo một lược đồ nào cả.

1.1.3 Mục đích

- Công cụ hỗ trợ cho những app có chức năng search hoặc yêu cầu phức tạp.
- Elasticsearch có khả năng phân tích và thống kê dữ liệu.
- Elasticsearch chạy trên server riêng và đồng thời giao tiếp thông qua RESTful do vậy nên nó không phụ thuộc vào client viết bằng gì hay hệ thống hiện tại của bạn viết

bằng gì. Nên việc tích hợp nó vào hệ thống bạn là dễ dàng, bạn chỉ cần gửi request http lên là nó trả về kết quả.

1.1.4 Khả năng sử dụng

Khi bạn chạy một web bán hàng online, bạn cho phép khách hàng có thể tìm kiếm tìm kiếm sản phẩm của bạn. Trong trường hợp này bạn có thể sử dụng ElasticSearch để lưu trữ toàn bộ danh mục sản phẩm, cung cấp các đề xuất tìm kiếm. Lưu trữ các khoảng không quảng cáo và làm đầy chúng khi cần thiết.

Bạn muốn có một tập log hoặc có tập dữ liệu trao đổi và bạn muốn phân tích chúng thành các data dưới dạng xu hướng, thống kê, tóm tắt, hoặc không phải các loại trên. Trong trường hợp này bạn có thể sử dụng Logstash (một phần của ElasticSearch) để thu thập, tổng hợp và phân tích cú pháp dữ liệu của bạn, sau đó chuyển dữ liệu từ Logstash vào ElasticSearch. Lúc này, bạn có thể search hoặc tổng hợp thông tin theo cách mình muốn.

Bạn có nhu cầu phân tích, hoặc kinh doanh mà muốn điều tra, phân tích, hoặc cái nhìn trực quan hoặc đặt câu hỏi quảng cáo cho một dữ liệu rất lớn. Trường hợp này, bạn cần sử dụng ElasticSearch để lưu trữ data của bạn và sau đó sử dụng Kibana (một phần của Elasticsearch/ Logstash/Kibana stack) để xây dựng bảng điều tra, điều đó giúp bạn có thể trực quan hóa data của mình, đây cũng là điều rất quan trọng.Thêm vào đó, bạn có thể sử dụng chức năng tổng hợp của Elasticsearch để xây dựng những câu truy vấn thông minh phù hợp với yêu cầu bài toán.

1.1.5 Ưu điểm

Tìm kiếm dữ liệu rất nhanh chóng, mạnh mẽ dựa trên Apache Lucene (near - real time searching).

Có khả năng phân tích dữ liệu (Analysis data).

Khả năng mở rộng theo chiều ngang tuyệt vời.

Hỗ trợ tìm kiếm mờ (fuzzy), tức là từ khóa tìm kiếm có thể bị sai lỗi chính tả hay không đúng cú pháp thì vẫn có khả năng Elasticsearch trả về kết quả tốt.

Hỗ trợ Structured Query DSL (Domain-Specific Language), cung cấp việc đặc tả những câu truy vấn phức tạp một cách cụ thể và rõ ràng bằng JSON.

Hỗ trợ nhiều Elasticsearch client như Java, PHP, Javascript, Ruby, .NET, Python.

1.1.6 Nhược điểm

Elasticsearch được thiết kế cho mục đích search, do vậy với những nhiệm vụ khác ngoài search như CRUD thì Elasticsearch kém thê hơn so với những database khác như Mongoddb, Mysql Do vậy người ta ít khi dùng Elasticsearch làm database chính, mà thường kết hợp nó với 1 database khác.

Trong Elasticsearch không có khái niệm database transaction , tức là nó sẽ không đảm bảo được toàn vẹn dữ liệu trong các hoạt động Insert, Update, Delete. Tức khi chúng ta thực hiện thay đổi nhiều bản ghi nếu xảy ra lỗi thì sẽ làm cho logic của mình bị sai hay dẫn tới mất mát dữ liệu. Đây cũng là 1 phần khiến Elasticsearch không nên là database chính.

Không thích hợp với những hệ thống thường xuyên cập nhật dữ liệu. Sẽ rất tốn kém cho việc đánh index dữ liệu.

1.1.7 Bảo mật

Các tính năng bảo mật của Elasticsearch cung cấp cho người dùng quyền truy cập phù hợp. Các nhóm công nghệ thông tin, vận hành và website có thể tận dụng các khả năng phù hợp này để quản lý người dùng có thiện chí và đẩy lùi các tác nhân độc hại.

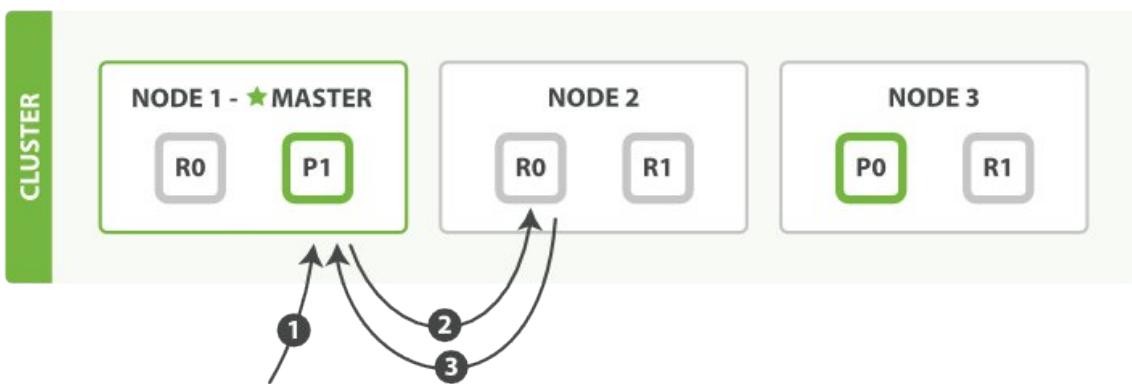
1.2 Ngôn ngữ truy vấn

Elasticsearch cung cấp full Query DSL (Domain Specific Language) dựa trên JSON để định nghĩa truy vấn. Hãy nghĩ về Query DSL như một AST (Abstract Syntax Tree) của truy vấn, bao gồm 2 loại:

Câu truy vấn đơn: Các câu lệnh truy vấn đơn tìm kiếm một giá trị cụ thể trong một field cụ thể, ví dụ như math, term hay range. Những câu lệnh này có thể được sử dụng bởi chính nó.

Câu lệnh truy vấn kép: Các câu lệnh truy vấn kép bao gồm những câu vấn đơn khác hoặc truy vấn kép và được sử dụng

Quá trình lấy dữ liệu:



Hình 1.1. Sơ đồ lấy dữ liệu của elasticsearch

Hình trên mô phỏng quá trình lưu trình truy vấn dữ liệu theo id của document. Có thể chia thành 3 bước:

- Bước 1: Request được gửi đến node master (Node 1). Tại đây xác định primary shard cho document sẽ là 0.
- Bước 2: Do tất cả các node đều lưu dữ liệu, nên master node sẽ chọn ra 1 node và lấy dữ liệu ở shard số 0. Việc chọn này giúp giảm tập trung vào 1 node. Thuật toán Round-robin được sử dụng để các shard được chọn khác nhau ở mỗi request. Trong trường hợp này Node 2 được chọn.
- Bước 3: Replica 0 ở Node 2 trả về kết quả cho master node.

Quá trình truy vấn dữ liệu:

- Node nhận request (node trung gian) sẽ gửi broadcast request đó đến tất cả các node khác. Tại mỗi node này sẽ chỉ định shard thực hiện công việc tìm kiếm theo yêu cầu. Shard có thể là primary hoặc replica shard.
- Mỗi shard sẽ thực hiện công việc tìm kiếm, trả về id và score của document. Trong đó score là giá trị dùng để sắp xếp. Danh sách gồm id, score này là một danh sách đã được sắp xếp, mang tính chất cục bộ.

- Node trung gian sau khi nhận kết quả trả về từ các node khác sẽ thực hiện công việc sắp xếp toàn cục tất cả các document, dựa theo id và score. Cuối cùng trả ra kết quả về phía client.

1.3 Tổ chức dữ liệu và Index

1.3.1 Tổ chức dữ liệu

Tài liệu và index:

Elasticsearch sử dụng "index" để lưu trữ và xử lý dữ liệu, nó tương tự một cơ sở dữ liệu thông thường. Dữ liệu sẽ được lưu trữ thành các "tài liệu" dưới dạng JSON. Mỗi một index có nhiều "type" (tương tự như bảng trong cơ sở dữ liệu), điều này sẽ giúp chúng ta phân tách các loại dữ liệu khác nhau khi lưu vào cùng một index.

Mọi tài liệu được lưu vào trong cùng một type của một index sẽ có cấu trúc các trường và kiểu dữ liệu giống nhau (giống như mỗi bảng có một schema trong các hệ cơ sở dữ liệu quan hệ)

Mỗi một index của Elasticsearch có một hoặc nhiều shard (tùy cấu hình, mặc định là 5). Các shard này lại tồn tại trên các node khác nhau. Nội dung về shard và node này liên quan nhiều đến kiến trúc của Elasticsearch, cũng là một phần rất quan trọng giúp Elasticsearch có thể thực hiện công việc tìm kiếm một cách nhanh chóng.

Shard cũng có hai loại là primary và replica, mỗi một primary shard sẽ có một vài (có thể là không có) shard replica (mặc định là 1). Elasticsearch sẽ đảm bảo là primary shard và replica shard sẽ tồn tại trên các node khác nhau. Một đặc điểm rất quan trọng, đó là số lượng primary shard cho một index không thể thay đổi được sau khi index đã được tạo. Do đó, trong thực tế, chúng ta sẽ phải cân nhắc rất nhiều yếu tố để quyết định xem cần bao nhiêu primary shard cho mỗi index. Trừ khi bạn muốn tạo lại index và thêm dữ liệu vào từ đầu.

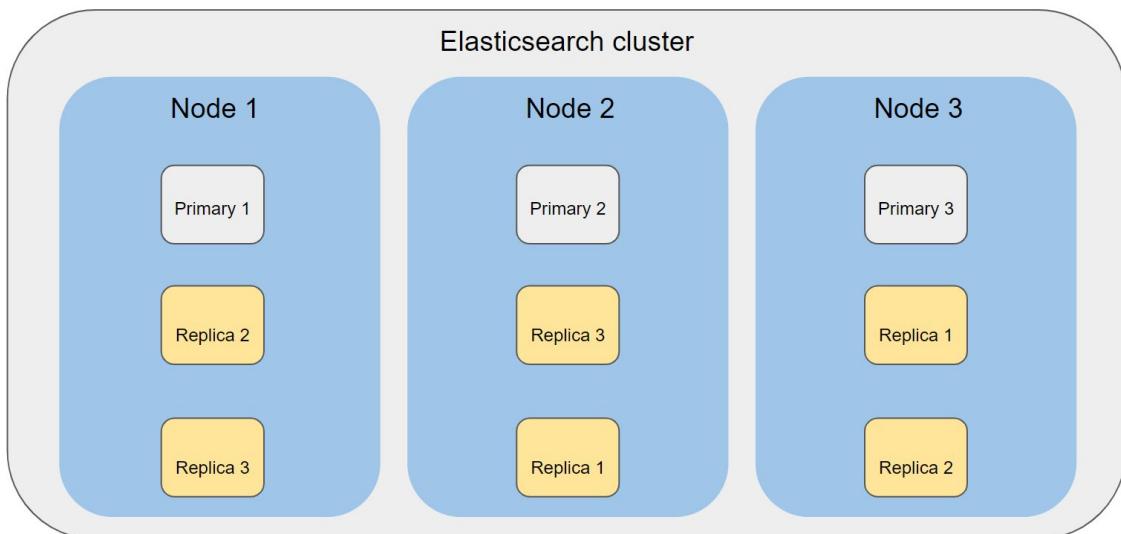
Elasticsearch hoạt động dựa trên Lucene của Apache. Một shard thực chất là một Lucene index, đó mới là nơi thực sự lưu trữ dữ liệu, và bản thân shard cũng là một search engine. Là một Lucene index, nên một shard lại được tạo nên từ nhiều segment và mỗi segment là một inverted index với đầy đủ chức năng.

Segment cho phép Lucene có thể thêm tài liệu vào index một cách dễ dàng mà không cần quá nhiều xử lý với index.

Với mỗi yêu cầu tìm kiếm, mọi segment trong một index sẽ được sử dụng, và mỗi segment như vậy sẽ hoạt động một cách khá độc lập (điều này khiến nó sử dụng CPU và bộ nhớ). Điều đó có nghĩa là, nếu có quá nhiều segments, performance của thao tác tìm kiếm sẽ bị giảm đi, nhưng nếu ít segment quá thì lại không tận dụng hết được khả năng của máy chủ.

Để xử lý vấn đề này, Lucene thực hiện một thao tác khá thông minh là "gộp" các segment nhỏ lại thành một segment lớn hơn. Tuy nhiên, điều này cũng có hai mặt: nếu thao tác gộp này được tiến hành không cẩn thận, Elasticsearch có thể rơi vào sai lầm là gộp tất cả segment vào làm một và bỏ phí rất nhiều tài nguyên (điều này cũng khiến performance giảm đi).

Node và cluster: Node là cơ sở trong kiến trúc server của Elasticsearch, và nhiều node hợp với nhau tạo thành cluster.



Hình 1.2. Cấu trúc node bên trong một cluster

Mô hình lưu trữ:

- Elasticsearch sử dụng Lucene của Apache - một thư viện full text search viết bằng Java. Về bản chất, nó sử dụng một cấu trúc dữ liệu gọi là inverted index (chỉ mục đảo ngược) để có thể thực hiện tìm kiếm với hiệu suất cao.

- Tài liệu chính đơn vị cơ sở để quản lý dữ liệu trong Elasticsearch và inverted index được tạo ra bằng việc tokenize (thuật ngữ trong Elasticsearch) các khái niệm trong tài liệu. Bằng việc sử dụng kỹ thuật inverted index, một bảng chỉ mục các khái niệm và danh sách tài liệu liên quan đến khái niệm đó sẽ được tạo ra.
- Nó khá tương đồng với chỉ mục của một cuốn sách, nơi hiển thị một danh sách các khái niệm được sử dụng cùng với số trang mà nó xuất hiện. Trong Elasticsearch, khi nói rằng một tài liệu được index, thì chúng ta hiểu rằng, inverted index của tài liệu đó đã được tạo ra.
- Ví dụ, khi chúng ta cần tìm từ khoá "Việt Nam", Elasticsearch sẽ thực hiện tìm kiếm trên inverted index (rất nhanh do các từ được sắp xếp), tìm được từ khoá "Việt Nam", sau đó trả về ID của các tài liệu tương ứng. Để có thể tìm kiếm nâng cao hơn (ví dụ, tìm với từ khoá "Viet Nam"), thì quá trình phân tích và index tài liệu là rất quan trọng.
- Ngoài ra, quá trình tìm kiếm thực tế còn phức tạp hơn nữa, khi mà Elasticsearch còn phải đánh giá mức độ tương qua giữa những từ trong index với từ khoá cần tìm.

Tìm kiếm: Quá trình tìm kiếm tài liệu sẽ diễn ra theo 2 giai đoạn:

Giai đoạn query:

- Đầu tiên, truy vấn tìm kiếm sẽ đến node điều phối, ở đây, truy vấn này sẽ được chuyển tiếp đến mọi shard (cả primary và replica) trong index. Mỗi shard sẽ thực hiện thao tác tìm kiếm tương ứng với truy vấn một cách độc lập và trả về ID của các tài liệu có độ tương quan cao nhất (mặc định sẽ là 10 tài liệu có điểm tương quan cao nhất được trả về).
- Các ID này được trả về cho node điều phối, ở đây, nó sẽ được gộp với kết quả của các shard khác và sắp xếp lại để tìm ra những tài liệu có độ tương quan cao nhất.

Giai đoạn fetch:

- Sau khi node điều phối gộp và sắp xếp lại kết quả nhận được từ các shard, nó sẽ thực hiện thao tác lấy thông tin của tài liệu đó. Các shard lại thực hiện công

việc của mình và trả về tài liệu (tất cả tài liệu chứ không chỉ ID như giai đoạn trước) cho node điều phối.

- Sau khi quá trình lấy dữ liệu đã xong, kết quả sẽ được trả về cho client.

Lưu ý rằng, để thực hiện tìm kiếm hiệu quả, mọi node trong cluster cần biết được trạng thái của cluster đó. Trạng thái của cluster sẽ bao gồm các thông tin như mỗi node có chứa index và shard nào. Nhờ đó, mọi node đều có thể trở thành node điều phối (và Elasticsearch sẽ thay đổi node điều phối trong trường hợp node được chọn không hoạt động).

```
# Tìm kiếm trên nhiều trường với các mức độ khác nhau
"query": {
    "function_score": {
        "query": {
            "multi_match": {
                "query": "Lambda Expressions",
                "fields": [
                    "title",
                    "tags^0.8",
                    "speakers.name^0.6",
                    "description^0.3"
                ]
            }
        },
        "functions": [
            {
                # Nội dung kỹ sẽ có độ quan trọng thấp hơn
                # dựa trên khoảng cách Gauss
                "gauss": {
                    "publishedOn": {
                        "scale": "130w",
                        "offset": "26w",
                        "decay": 0.3
                    }
                }
            }
        ]
    }
}
```

Hình 1.3. Câu lệnh truy vấn tìm kiếm của elasticsearch

1.3.2 Index

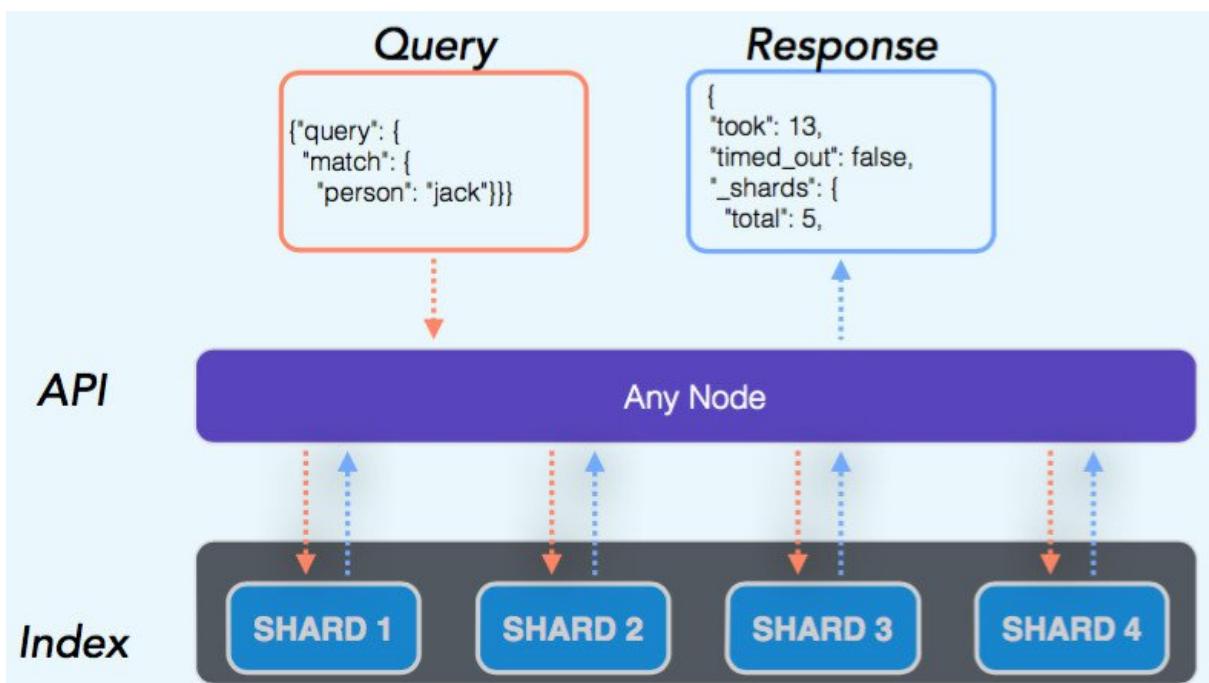
Tổng quan về index

Elasticsearch còn có những điểm khác giúp searching document hiệu quả hơn. Khi lưu trữ document trong Elasticsearch, nó tạo ra một số internal data structures làm cho query performance tốt hơn.

Mỗi document gửi tới ES được lưu trữ qua một thuật toán và sau đó được gửi đến shard. Elasticsearch cố gắng để phân tán document thông qua các shard. Khi lưu trữ document, Elasticsearch tạo ra inverted index, map các thuật ngữ/từ khóa xuất hiện trong document này tới chính document đó.

Khi sử dụng inverted index, nó có thể tìm kiếm thông qua terms như một binary tree (sử dụng thứ tự chữ cái) làm giảm thời gian tìm kiếm.

Một điều quan trọng khi lưu trữ document đó là được quyết định cách tốt nhất để lưu trữ chúng giúp nâng cao tốc độ truy vấn. Khi thiết kế các giải pháp sử dụng Elasticsearch, điều đáng lưu tâm nhất khi lưu trữ document chính là: truy vấn document này như thế nào? “First query” này tiếp cận với việc sử dụng cho tất cả các khả năng ES indexing để thực hiện truy vấn cực kỳ nhanh chóng.



Hình 1.4. Ảnh minh họa cách Elasticsearch thực hiện truy vấn

Index Refresh

- Khi có một truy vấn yêu cầu index một tài liệu, nội dung đó sẽ được thêm vào translog và được lưu tạo vào buffer trên bộ nhớ trong. Khi tiến hành index refresh (mặc định mỗi giây một lần), tiến trình refresh sẽ tạo segment ngay trên bộ nhớ trong từ buffer trước đó. Tới lúc này, tài liệu cần index đã có thể tìm kiếm được.
- Sau đó, buffer trên bộ nhớ trong sẽ bị xoá đi. Dần dần quá trình này sẽ tạo ra một tập các segment, rồi các segment lại được gộp lại với nhau thành segment lớn hơn để đảm bảo việc sử dụng các tài nguyên của máy chủ (CPU, bộ nhớ, v.v...) được tối ưu.
- Quá trình index refresh là một hoạt động rất tốn kém, do đó nó thường được đặt lịch chạy định kỳ chứ không phải được kích hoạt ngay sau khi index.
- Nếu chúng ta cần index một lượng lớn tài liệu và không có nhu cầu gấp trong việc tìm kiếm ngay lập tức các tài liệu đó, chúng ta có thể tìm cách tối ưu hiệu năng của quá trình index bằng cách kéo giãn khoảng thời gian mà index refresh được thực hiện.

1.4 Backup, recovery, partition, sharding, replication, cluster

1.4.1 Backup

Một Elasticsearch snapshot là một back up của cụm (cluster) Elasticsearch. Snapshot module cho phép bạn tạo nhiều snapshot của nhiều hơn một hoặc các chỉ số riêng lẻ hoặc một snapshot của toàn cụm. Snapshot của các chỉ số cụ thể hoặc toàn bộ cụm và lưu trữ trên hệ thống tệp được chia sẻ.

Có nhiều loại kho khác nhau được hỗ trợ. Nếu bạn có một hệ thống tập tin chia sẻ. Một hệ thống tệp NFS có thể truy cập được bởi tất cả các nút Elasticsearch tại cùng một điểm gắn kết, sau đó bạn có thể sử dụng hệ thống đó để lưu trữ các chỉ số riêng lẻ cụ thể của mình hoặc toàn bộ cluster snapshot.

Elasticsearch được thiết kế để chạy trong các môi trường khác nhau, cũng hỗ trợ các snapshot và restore module khu lưu trữ đám mây khác nhau như:

- AWS

- Google Cloud Storage
- Azure Cloud etc

1.4.2 Recovery

Trả về thông tin về các phục hồi shard đang diễn ra và đã hoàn thành, tương tự như API index recovery. Đây là một cái nhìn nhỏ gọn hơn về API index recovery JSON.

Recovery phân đoạn là quá trình đồng bộ phân đoạn bản sao từ phân đoạn chính. Sau khi hoàn thành, bản sao có sẵn để tìm kiếm.

Recovery tự động xảy ra trong các quá trình sau:

- Node khởi động hoặc thất bại. Đây được gọi là local store recovery (khôi phục địa phương).
- Primary shard replication.
- Di dời một mảnh vỡ đến một node khác trong cùng một cluster.
- Snapshot restoration.

Tham số đường dẫn: (Tùy chọn, chuỗi) Danh sách được phân tách bằng dấu phẩy hoặc biểu thức ký tự đại diện của các tên chỉ mục được sử dụng để giới hạn yêu cầu.

Tham số truy vấn:

- Active_only: (Tùy chọn, boolean) Nếu “true”, phản hồi chỉ bao gồm các phục hồi phân đoạn đang diễn ra. Mặc định là “false”.
- Bytes: (Tùy chọn, đơn vị đo kích thước byte) Đơn vị được sử dụng để hiển thị giá trị byte.
- Detailed: (Tùy chọn, boolean) Nếu “true”, phản hồi bao gồm thông tin chi tiết về phục hồi phân đoạn. Mặc định là “false”.
- Format: (Tùy chọn, chuỗi) Phiên bản ngắn của tiêu đề chấp nhận HTTP. Các giá trị hợp lệ bao gồm JSON, YAML, ...
- Help
- Index

1.4.3 Partition

Trong Elasticsearch, type được định nghĩa cho các tài liệu có một tập các trường chung. Nó là một logical category/ partition của một index có ngữ nghĩa học hoàn toàn tùy thuộc vào người dùng. Bạn cũng có thể xác định nhiều type trong một index.

Type: Có thể có nhiều Type trong cùng một Index. Ví dụ, một website thương mại điện tử có thể sử dụng các sản phẩm cũ trong một Type và những sản phẩm mới trong một Type khác của cùng Index. Một Index có thể có nhiều Type, giống như có nhiều Table trong một Database.

1.4.4 Sharding

Mỗi một index của Elasticsearch có một hay nhiều shard (tùy cấu hình, mặc định là 5). Elasticsearch sử dụng index để lưu trữ và xử lý dữ liệu, nó tương tự một cơ sở dữ liệu thông thường. Các shard này tồn tại trên các node khác nhau. Nội dung về shard này liên quan nhiều đến kiến trúc của Elasticsearch, cũng là một phần quan trọng giúp cho Elasticsearch có thể thực hiện công việc tìm kiếm một cách nhanh chóng.

Shard cũng có hai loại: “Primary” và “Replica”, mỗi một primary shard sẽ có một vài shard replica. Elasticsearch sẽ đảm bảo là primary shard và replica shard sẽ tồn tại trên các node khác nhau.

Một đặc điểm rất quan trọng, đó là số lượng primary shard cho một index không thể thay đổi được sau khi index đã được tạo. Do đó, trong thực tế, chúng ta sẽ phải cân nhắc rất nhiều yếu tố để quyết định xem cần bao nhiêu primary shard cho mỗi index. Trừ khi bạn muốn tạo lại index và thêm dữ liệu vào từ đầu.

Một shard thực chất là một Lucene index(Elasticsearch hoạt động dựa trên Lucene của Apache), đó mới thực sự là nơi lưu dữ liệu, và bản thân shard cũng là một search engine. Là một Lucene index, nên một shard lại được tạo nên từ nhiều segment và mỗi segment là một inverted index với đầy đủ chức năng.

Segment cho phép Lucene có thể thêm tài liệu vào index. Với mỗi yêu cầu tìm kiếm, mọi segment có trong một index sẽ được sử dụng và mỗi segment sẽ hoạt động độc lập. Nếu có nhiều segment thì performance của thao tác bị giảm đi còn nếu ít segment thì không tận dụng được khả năng của máy chủ. Vì thế Lucene gộp các segment

nhỏ thành segment lớn. Tuy nhiên nếu không cẩn thận Elasticsearch có thể rơi vào sai lầm và bỏ phí nhiều tài nguyên dẫn đến performance giảm.

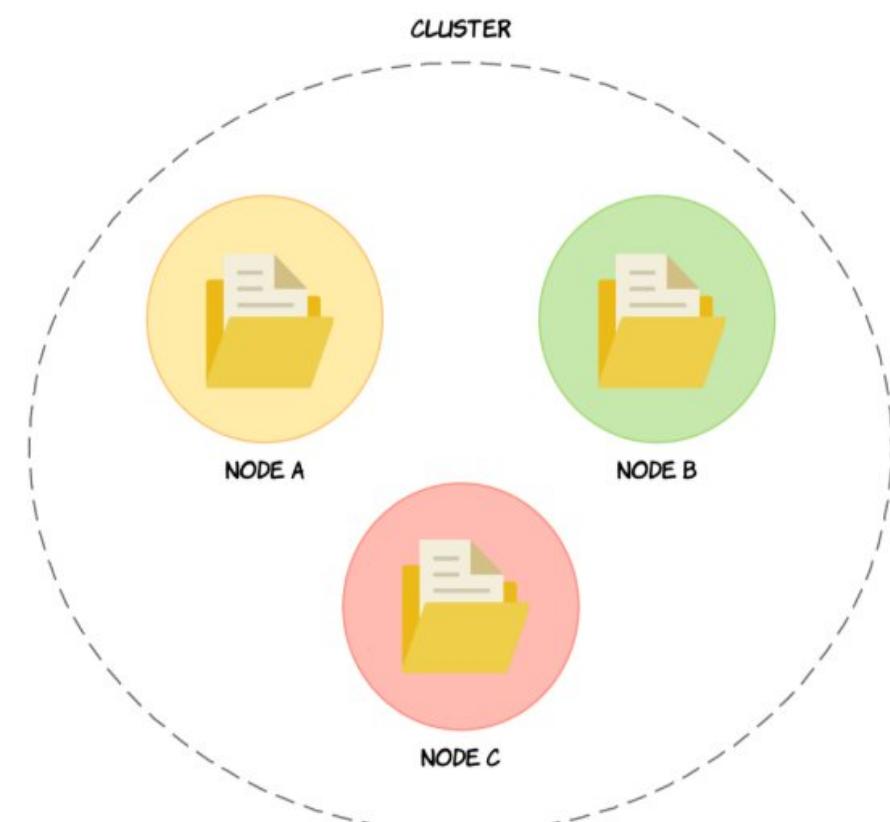
1.4.5 Replication

Replica Shard là nơi lưu trữ dữ liệu nhân bản của Primary Shard.

Replica Shard có vai trò đảm bảo tính toàn vẹn của dữ liệu khi primary shard xảy ra vấn đề. Ngoài ra replica shard có thể giúp tăng cường tốc độ tìm kiếm vì chúng ta có thể setup lượng replica shard nhiều hơn mặc định của Elasticsearch.

1.4.6 Cluster

Node là cơ sở trong kiến trúc server của Elasticsearch, và nhiều node hợp với nhau tạo thành cluster.



Hình 1.5. Ảnh minh họa một cluster

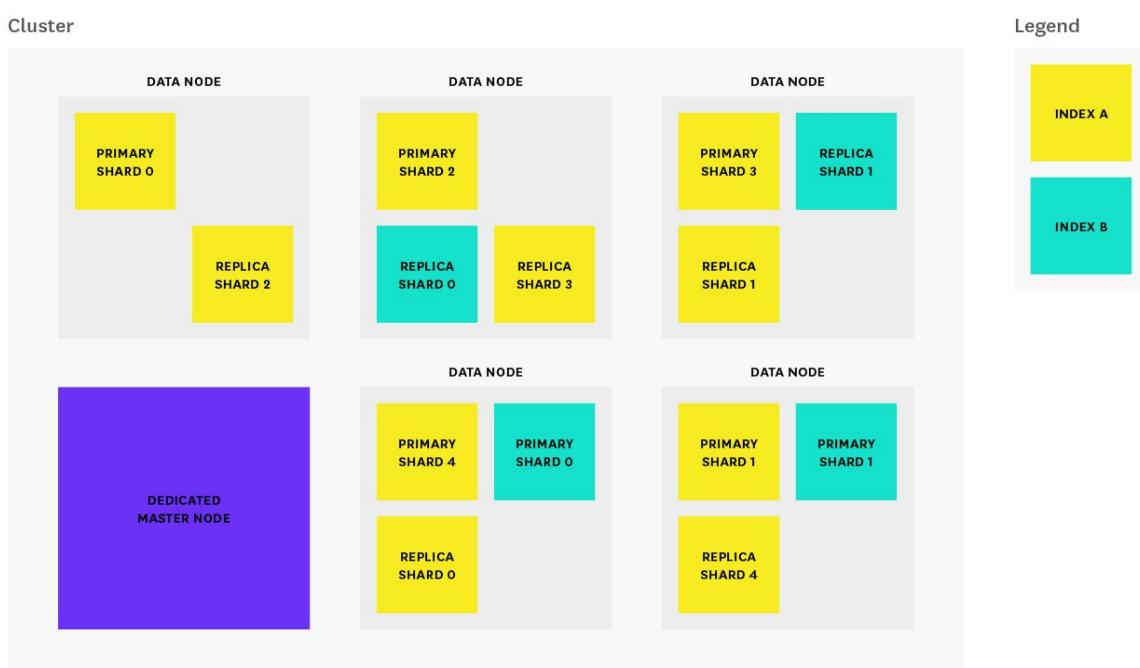
Tập hợp các node hoạt động cùng với nhau tạo thành một cluster, mỗi node trong cluster chứa một phần dữ liệu của cluster đó (và toàn bộ dữ liệu của một cluster sẽ được chia ra cho các node).

Mỗi node sẽ tham gia vào quá trình index và tìm kiếm dữ liệu trong cluster. Các truy vấn của người dùng đề thông qua HTTP request, và cluster sẽ chuyển những truy vấn đó cho các node của nó.

Các node trong cùng một cluster hiểu được các node khác nên có thể chuyển tiếp truy vấn của người dùng cho một node khác. Việc này được thực hiện nội bộ bên trong cluster.

Node có 3 loại khác nhau: master, data và client. Một cluster sẽ tự động chọn ra một node làm master từ các node của nó. Node master sẽ có nhiệm vụ điều phối công việc của cluster, ví dụ: Phân bổ các shard, tạo/ xóa index,... Chỉ có node master mới có khả năng cập nhật trạng thái của cluster.

Node data là node mà các shard được phân bổ, có nhiệm vụ thực hiện các công việc như index tài liệu, tìm kiếm và thống kê. Node client hoạt động như một load balancer có nhiệm vụ định tuyến các truy vấn của người dùng như index hay tìm kiếm. Một cluster có thể không có node client.



Hình 1.6. Kiến trúc của Elasticsearch.

Cluster và node được định danh bởi tên (duy nhất). Mặc định, tên của cluster sẽ là “elasticsearch” còn node được gán tên là một UUID (Universally Unique Identifier - Mã định danh toàn cầu). Một cluster về mặt kỹ thuật là có bao nhiêu node cũng được, thậm chí 1 node cũng không sao. Kiến trúc Elasticsearch giúp nó có khả năng mở rộng theo chiều ngang một cách nhanh chóng mà không hề cần một sự thay đổi nào trong code.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Bài toán

Bài toán chính mà hệ thống cần giải quyết bao gồm:

- Tìm kiếm tài liệu: Làm thế nào để người dùng có thể tìm kiếm tài liệu một cách nhanh chóng và hiệu quả dựa trên từ khóa, nội dung, hoặc các thuộc tính khác?
- Quản lý tài liệu: Làm thế nào để tổ chức và lưu trữ tài liệu một cách có hệ thống, dễ dàng cập nhật và quản lý phiên bản?
- Bảo mật thông tin: Làm thế nào để đảm bảo rằng tài liệu chỉ được truy cập bởi những người dùng có quyền hợp lệ?
- Khả năng mở rộng: Hệ thống cần có khả năng mở rộng khi khối lượng tài liệu tăng lên.

2.2 Hướng giải quyết

Để giải quyết các bài toán trên, hệ thống sẽ được thiết kế với các thành phần chính:

- Sử dụng Elasticsearch: Công nghệ này cho phép thực hiện tìm kiếm mạnh mẽ, hỗ trợ tìm kiếm toàn văn, phân tích dữ liệu và xử lý truy vấn phức tạp. Việc tích hợp Elasticsearch sẽ giúp tối ưu hóa hiệu suất tìm kiếm.
- Giao diện người dùng thân thiện: Xây dựng một giao diện dễ sử dụng, cho phép người dùng dễ dàng tìm kiếm, xem và quản lý tài liệu.
- Quản lý phân quyền: Thiết lập hệ thống phân quyền người dùng để đảm bảo an toàn thông tin và quyền truy cập.
- Tính năng lưu trữ và phiên bản: Phát triển chức năng lưu trữ có tổ chức và quản lý phiên bản tài liệu, giúp người dùng theo dõi thay đổi và phục hồi tài liệu khi cần thiết.

Với những giải pháp này, hệ thống quản lý tài liệu sẽ đáp ứng được các yêu cầu về hiệu suất, bảo mật và khả năng mở rộng trong tương lai.

2.3 Phân tích và thiết kế yêu cầu

2.3.1 Yêu cầu chức năng

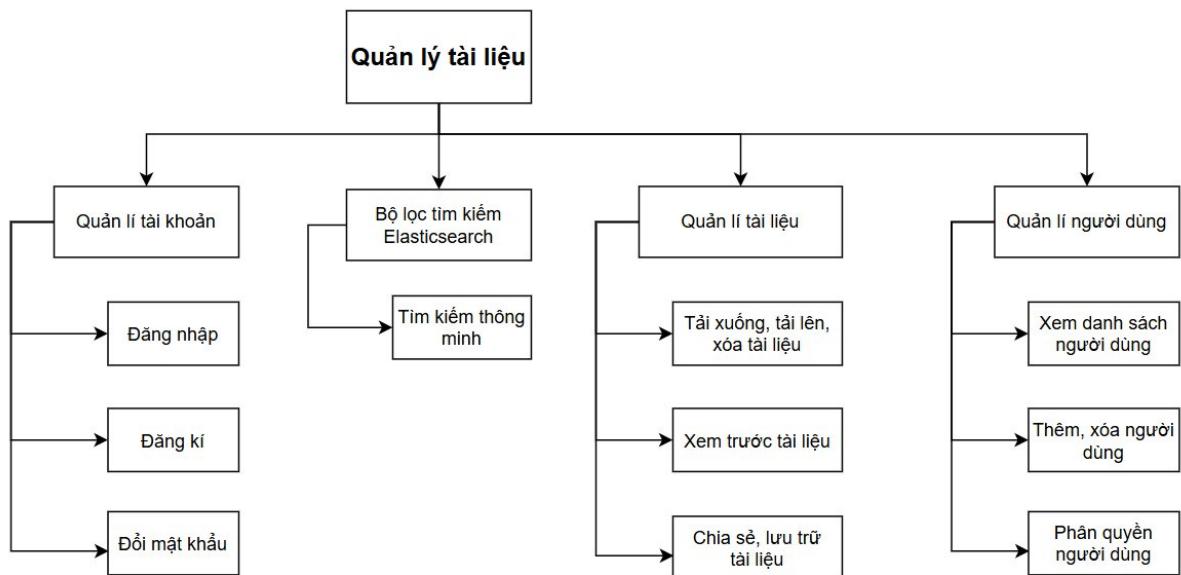
Quản trị viên:

- Đăng ký tài khoản với đầy đủ quyền hạn và chức năng của website.
- Đăng nhập vào website.
- Thêm, sửa, xóa quyền người dùng.
- Thêm, sửa, xóa và phân quyền người dùng .
- Thêm, sửa, xóa nhân viên và các đơn vị.
- Tải lên, tải xuống, óa, chia sẻ và lưu trữ tài liệu.
- Tìm kiếm nhanh tài liệu dựa vào công nghệ Elasticsearch.

Người dùng:

- Đăng nhập với tài khoản mà quản trị viên đã cung cấp
- Tải lên, tải xuống, óa, chia sẻ và lưu trữ tài liệu.
- Tìm kiếm nhanh tài liệu dựa vào công nghệ Elasticsearch

2.3.2 Biểu đồ phân rã chức năng và phân tích chức năng



Hình 2.1. Biểu đồ phân rã chức năng

Bảng 2.1. Bảng mô tả chi tiết chức năng

| MỤC | TÊN CHỨC NĂNG | MÔ TẢ |
|----------------------------------|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Quản lý tài khoản | ĐĂNG KÍ | Quản trị viên sẽ đăng ký một tài khoản. Đây là tài khoản có đầy đủ các quyền hạn và có đầy đủ các chức năng của web. |
| | ĐĂNG NHẬP | Quản trị viên đăng nhập tài khoản được đăng ký. Người dùng đăng nhập bằng tài khoản được quản trị viên cấp. |
| | ĐỔI MẬT KHẨU | Người dùng vào phần đổi mật khẩu trong hồ sơ cá nhân, sau khi nhập mật khẩu mới thỏa mãn các tiêu chí đề ra sẽ đổi mật khẩu thành công. Quên mật khẩu: Có thể đặt lại mật khẩu qua email đã đăng ký. |
| Bộ lọc tìm kiếm Elasticsearch | TÌM KIẾM THÔNG MINH | Người dùng nhập từ khóa vào bộ lọc tìm kiếm và kết quả trả về là tất cả các tài liệu có chứa từ khóa. |
| Quản lý người dùng | XEM DANH SÁCH NGƯỜI DÙNG | Quản trị viên xem danh sách người dùng trong phần thiết lập. |
| | THÊM SỬA XÓA NGƯỜI DÙNG | Quản trị viên thêm sửa và xóa người dùng trong phần thiết lập. |
| | PHÂN QUYỀN NGƯỜI DÙNG | Quản trị viên chỉnh sửa quyền hạn của người dùng. |

| | | |
|------------------|-----------------------------------------|----------------------------------------------------------------------------|
| Quản lý tài liệu | TẢI XUỐNG, TẢI LÊN, XÓA TÀI LIỆU | Người dùng tải lên, tải xuống và xóa tài liệu trong phần cá nhân của mình. |
| | XEM TRƯỚC TÀI LIỆU | Người dùng xem tài liệu đã tải lên hoặc tài liệu được chia sẻ. |
| | CHIA SẺ TÀI LIỆU | Người dùng chia sẻ và nhận chia sẻ tài liệu từ người dùng khác. |

2.3.3 Đặc tả yêu cầu chức năng

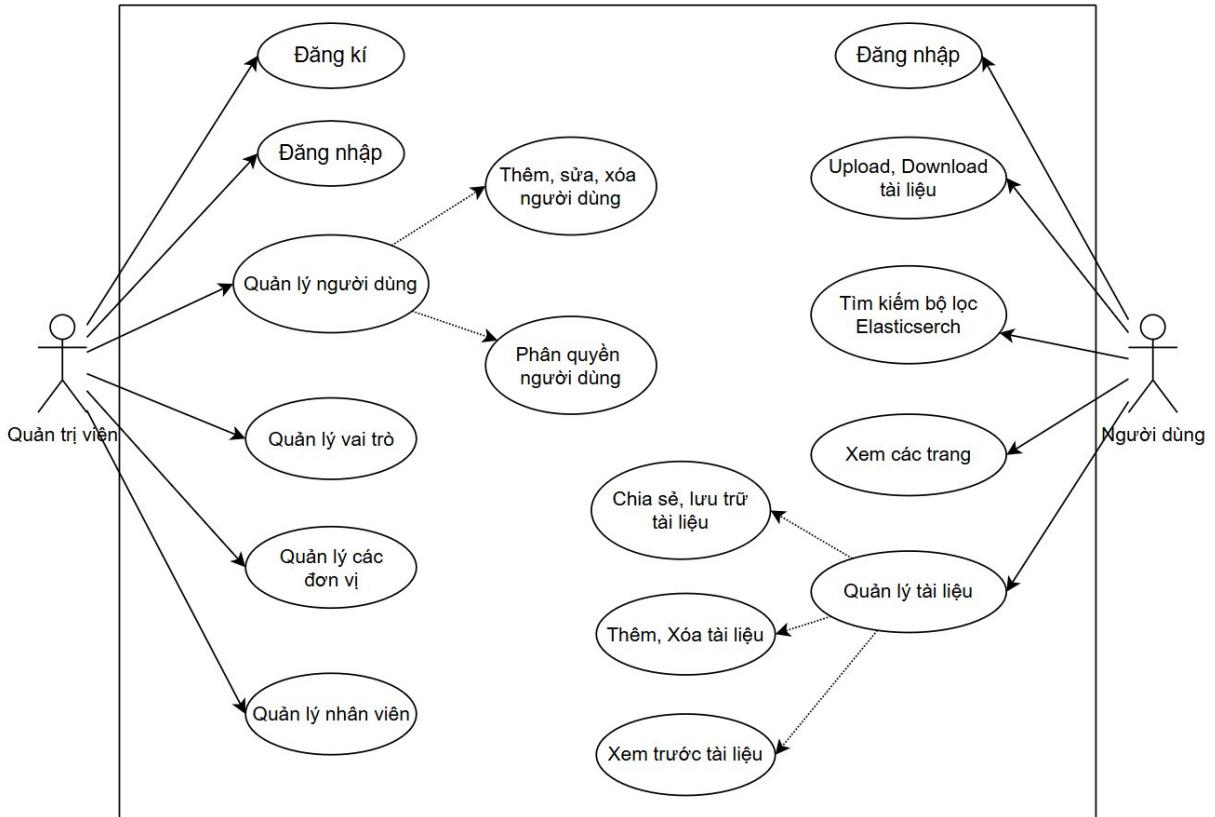
2.3.3.1 Các tác nhân

Website quản lý tài liệu gồm 2 tác nhân chính là quản trị viên và người dùng

Bảng 2.2. Bảng mô tả các tác nhân

| STT | Tác nhân | MÔ TẢ VAI TRÒ |
|-----|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1 | Người dùng | <ul style="list-style-type: none"> • Đăng nhập, khôi phục mật khẩu. • Tải tài liệu lên website, tải tài liệu về thiết bị. • Sử dụng bộ lọc tìm kiếm nhanh tài liệu. • Quản lý tài liệu: xem trước, chia sẻ, xóa tài liệu |
| 2 | Quản trị viên | <ul style="list-style-type: none"> • Đăng ký, đăng nhập, khôi phục mật khẩu • Quản lý tài khoản người dùng • Quản lý vai trò. • Quản lý nhân viên và các đơn vị. |

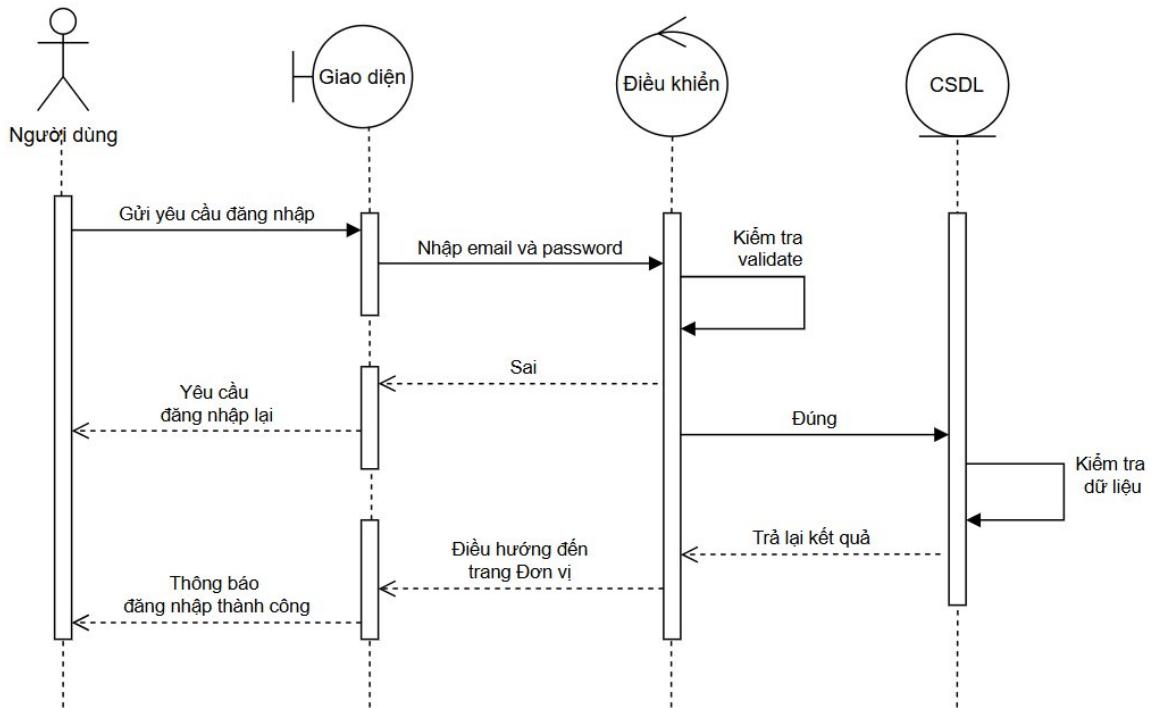
2.3.3.2 Sơ đồ use-case tổng quát



Hình 2.2. Sơ đồ tổng quát use-case

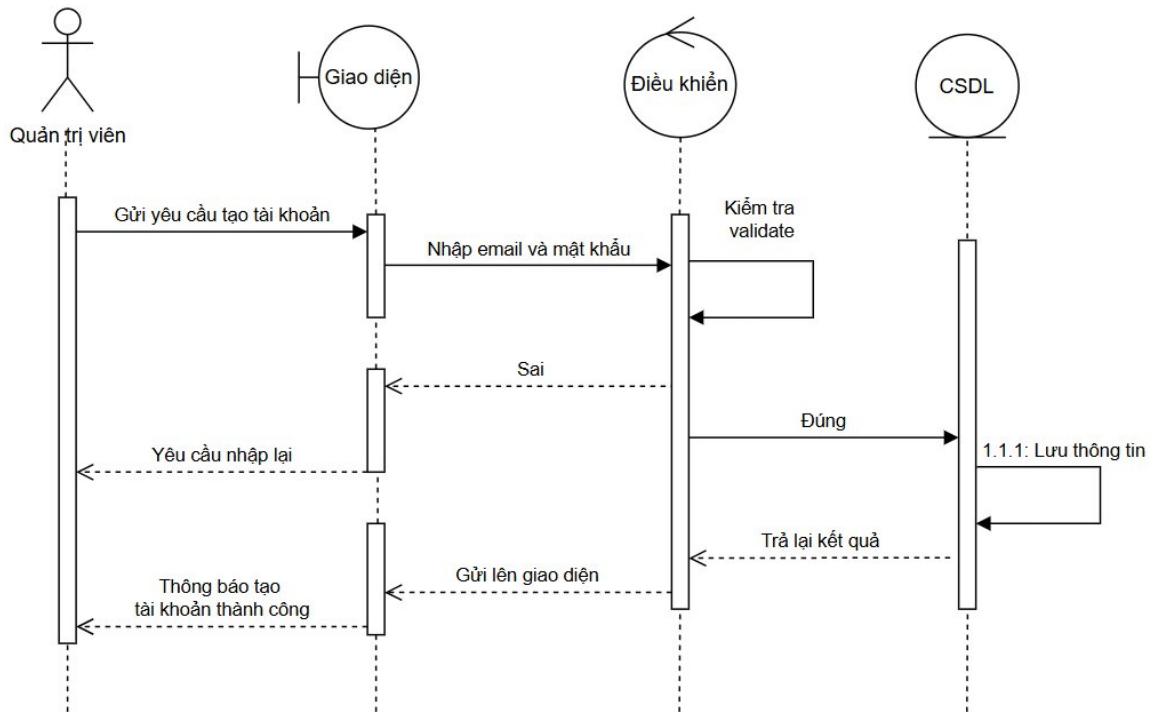
2.3.4 Biểu đồ tuần tự

2.3.4.1 Biểu đồ tuần tự chức năng đăng nhập



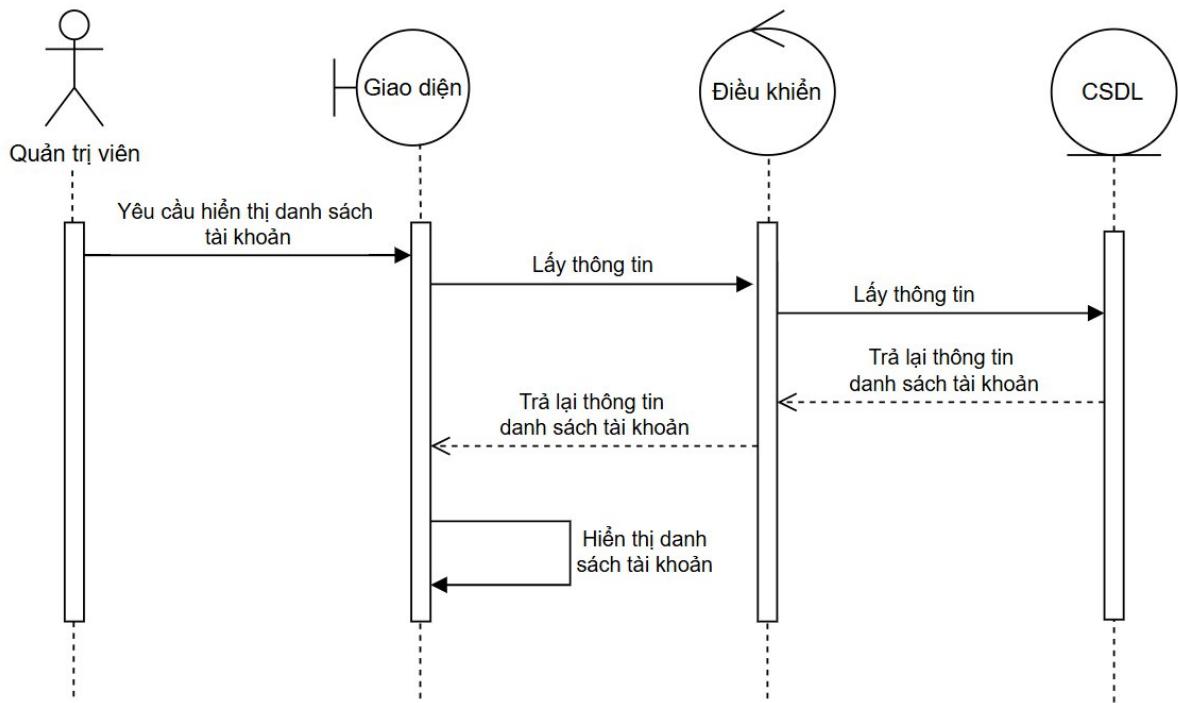
Hình 2.3 Biểu đồ tuần tự đăng nhập

2.3.4.2 Biểu đồ tuần tự chức năng đăng ký



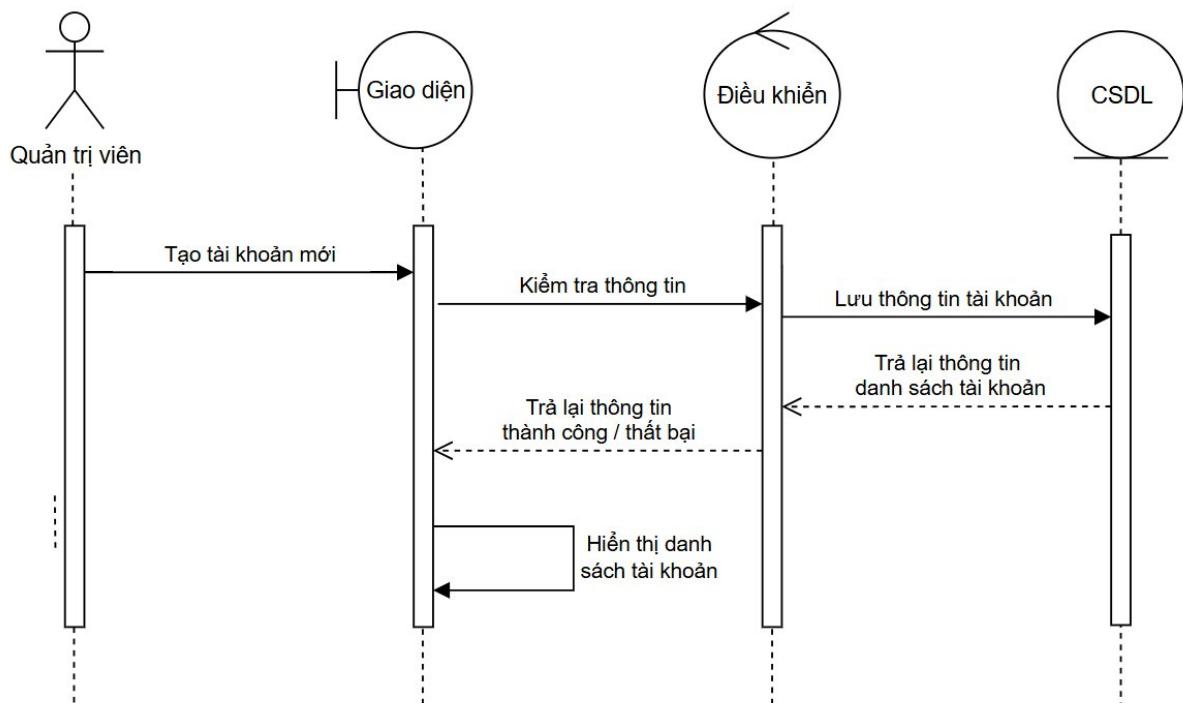
Hình 2.4. Biểu đồ tuần tự đăng ký

2.3.4.3 Biểu đồ tuần tự chức năng hiển thị danh sách tài khoản



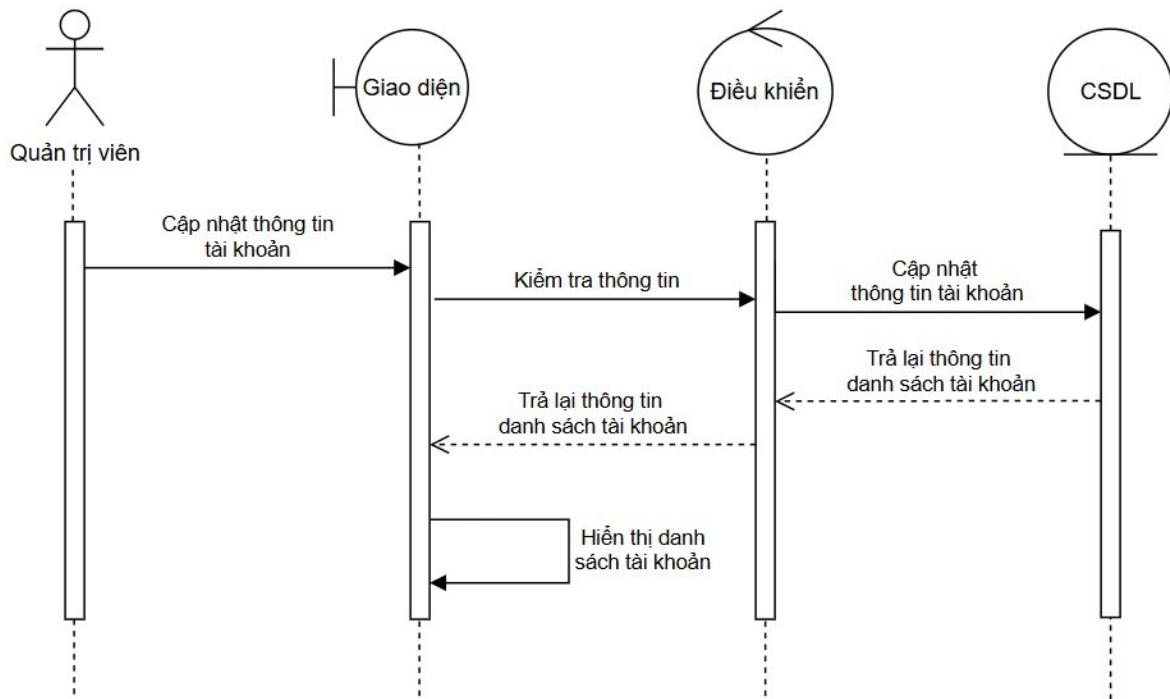
Hình 2.5. Biểu đồ tuần tự hiển thị danh sách tài khoản

2.3.4.4 Biểu đồ tuần tự chức năng tạo tài khoản mới



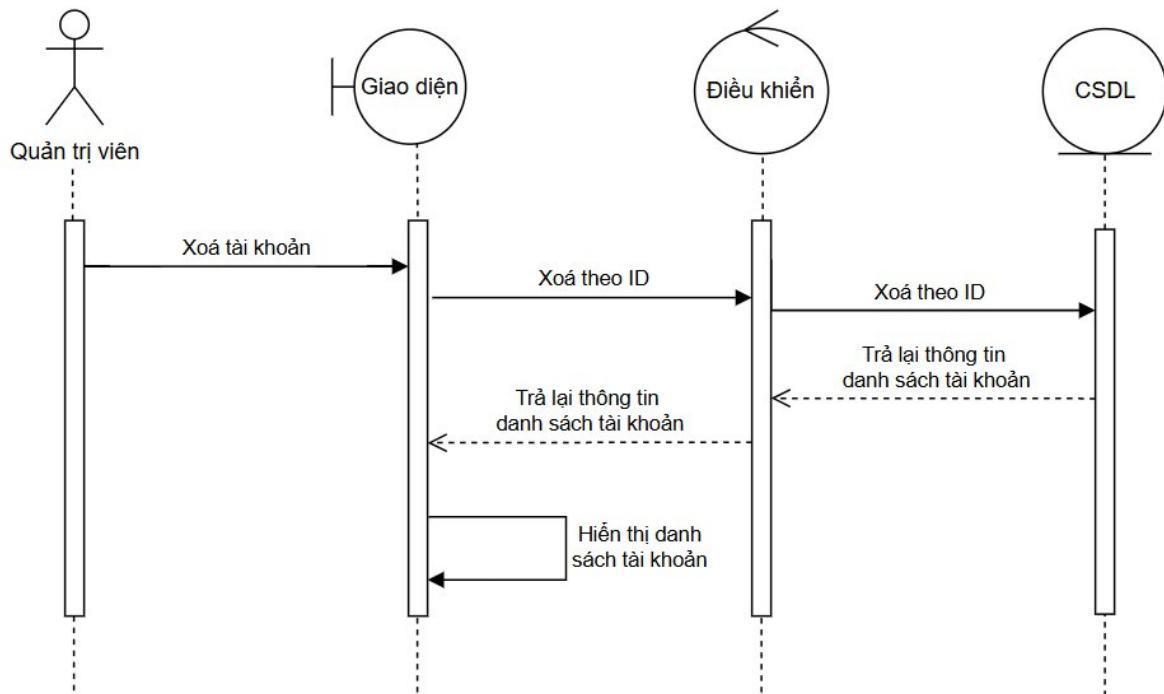
Hình 2.6. Biểu đồ tuần tự tạo tài khoản mới

2.3.4.5. Biểu đồ tuần tự chức năng cập nhật thông tin tài khoản



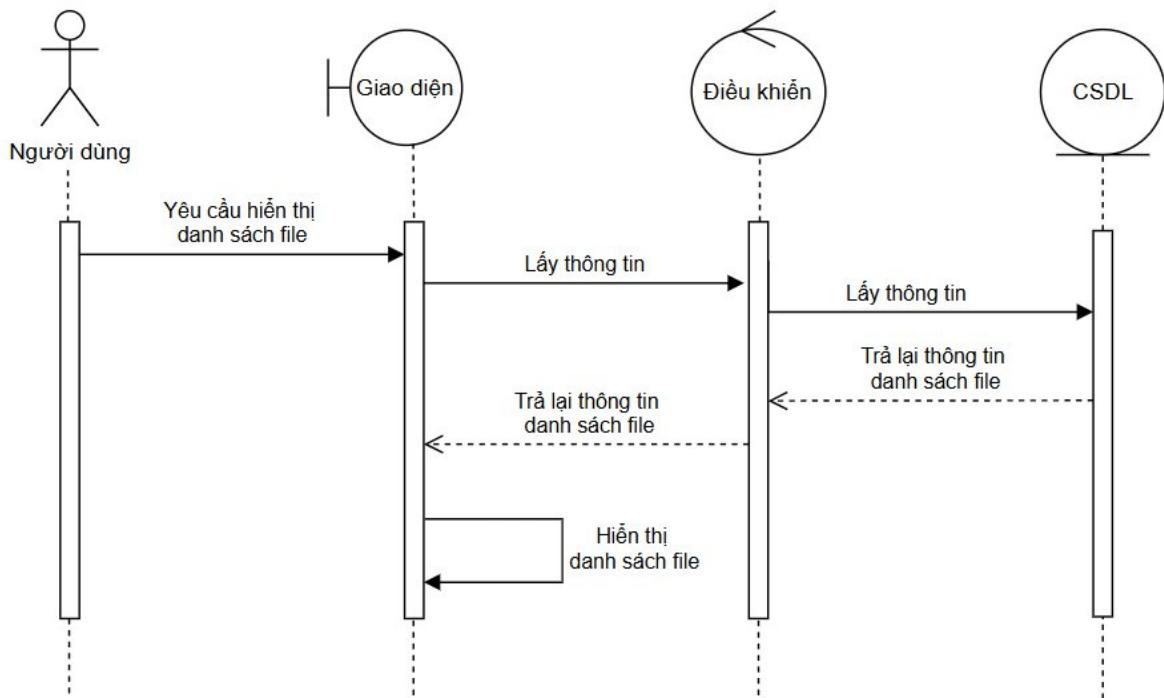
Hình 2.7. Biểu đồ tuần tự cập nhật thông tin tài khoản

2.3.4.6 Biểu đồ tuần tự chức năng xóa tài khoản



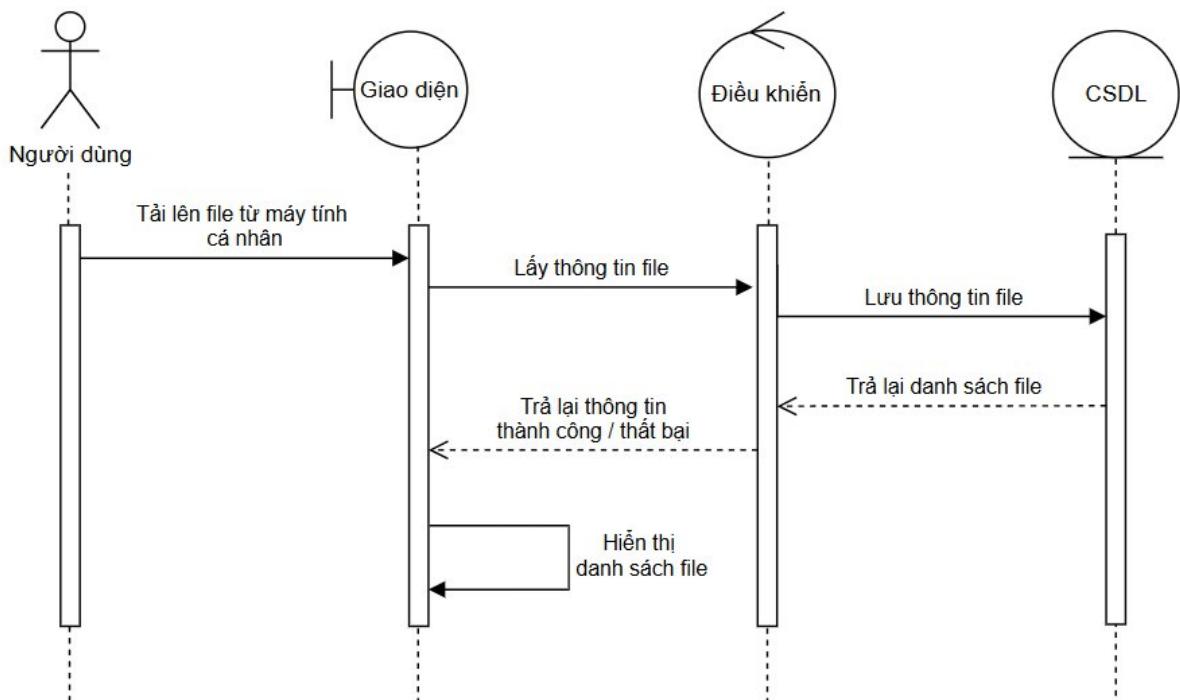
Hình 2.8. Biểu đồ tuần tự xóa tài khoản

2.3.4.7 Biểu đồ tuần tự chức năng hiển thị danh sách file



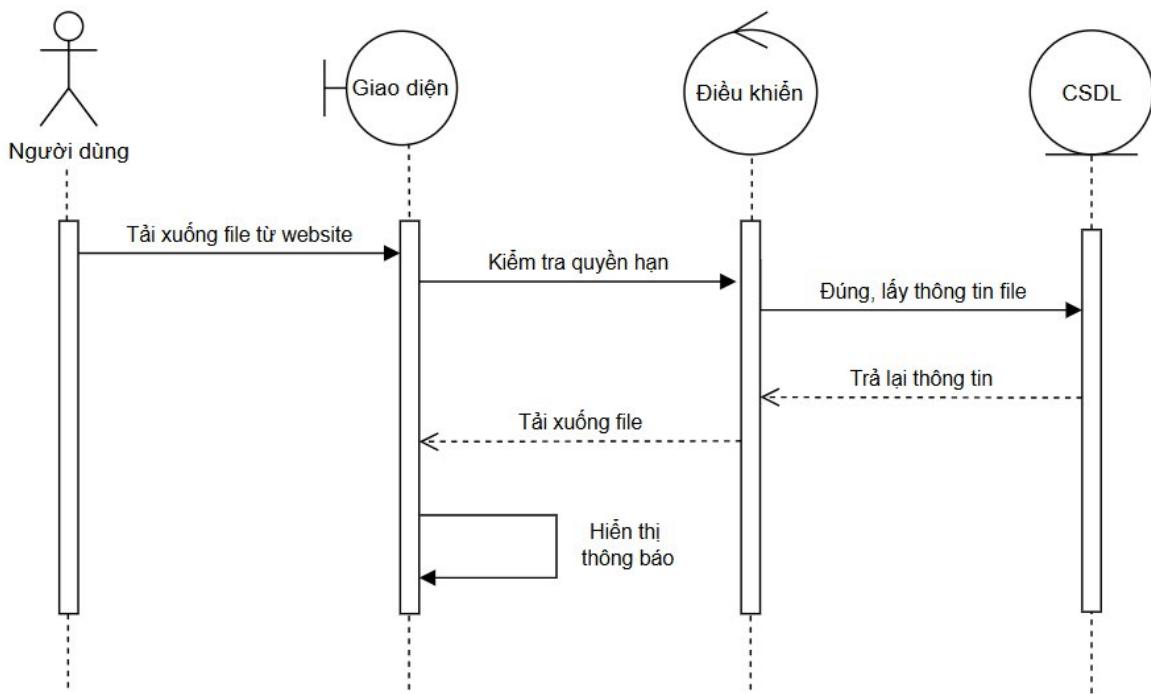
Hình 2.9. Biểu đồ tuần tự hiển thị danh sách file

2.3.4.8 Biểu đồ tuần tự chức năng tải file từ máy tính cá nhân



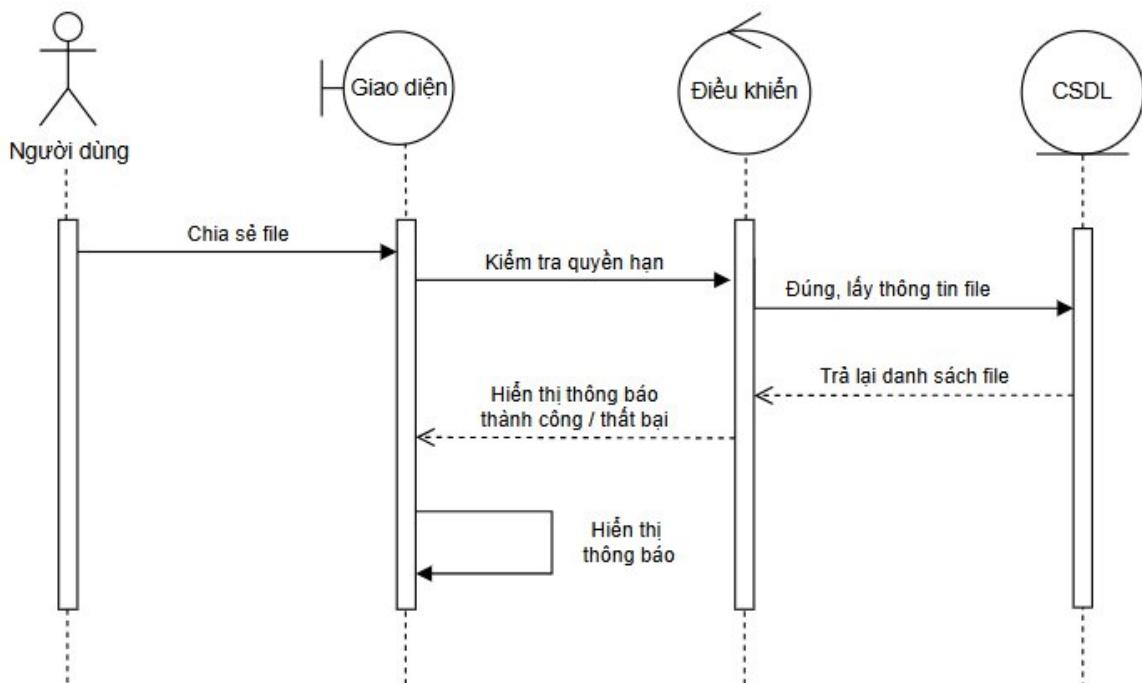
Hình 2.10. Biểu đồ tuần tự tải file lên website

2.3.4.9 Biểu đồ tuần tự chức năng tải file về máy tính cá nhân



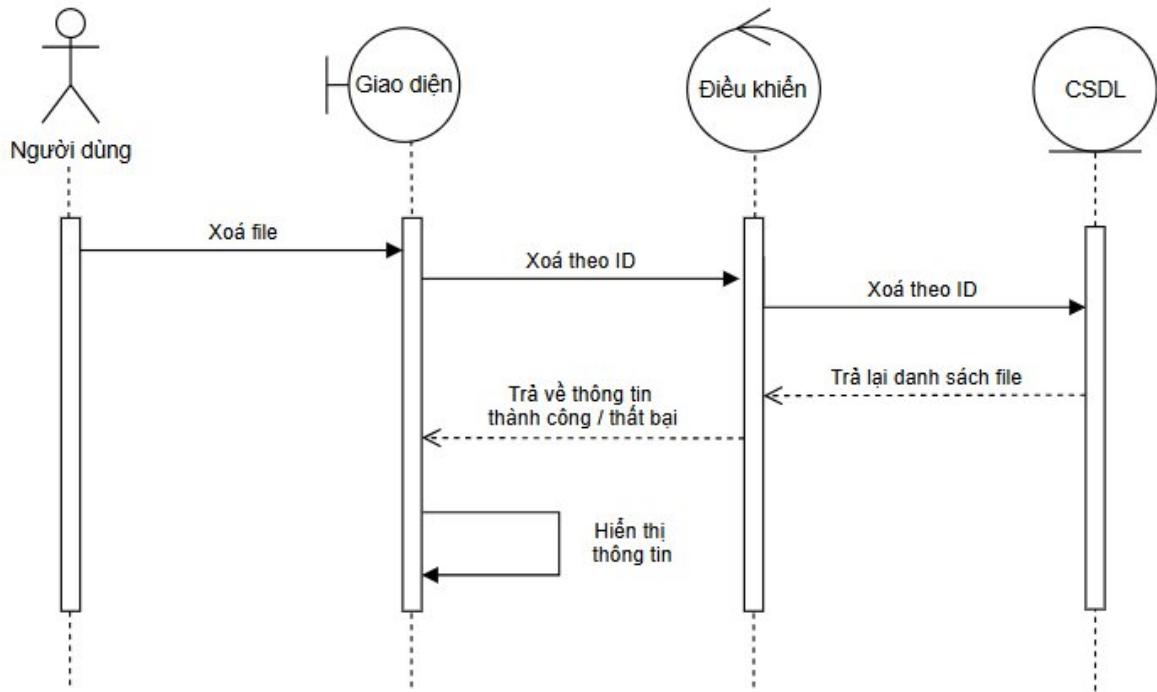
Hình 2.11. Biểu đồ tuần tự tải file về máy tính cá nhân

2.3.4.10 Biểu đồ tuần tự chức năng chia sẻ file cho người dùng khác



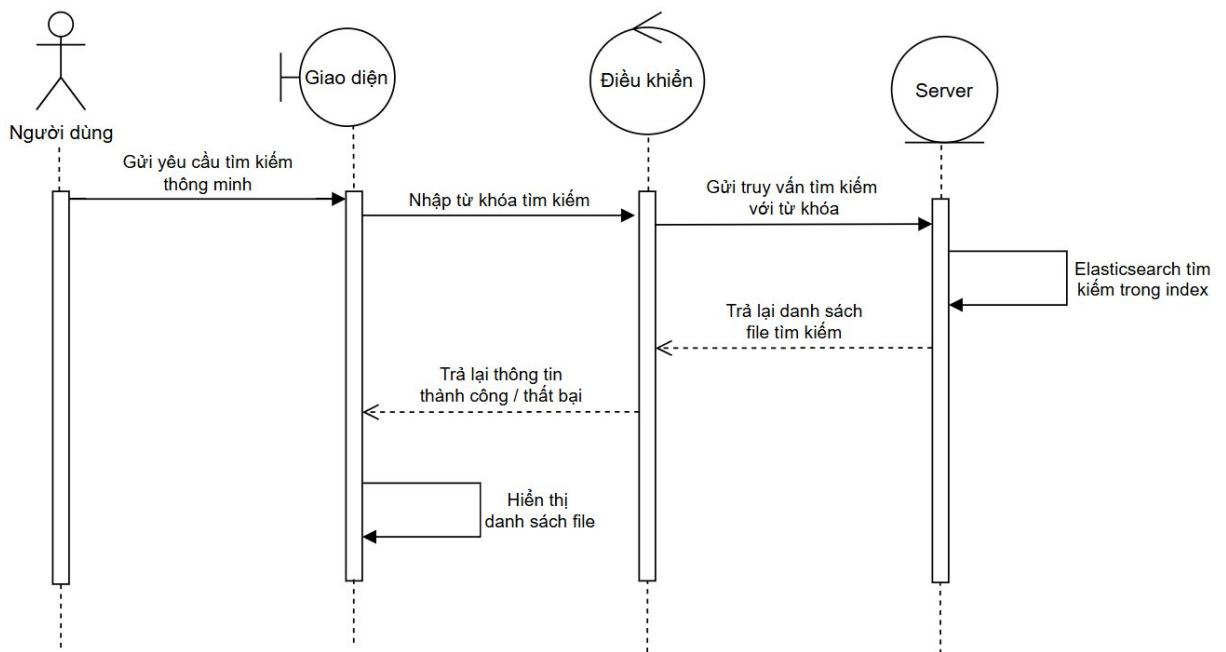
Hình 2.12. Biểu đồ tuần tự sự chia sẻ file

2.3.4.11 Biểu đồ tuần tự chức năng xóa file



Hình 2.13. Biểu đồ tuần tự xóa file

2.3.4.12 Biểu đồ tuần tự chức năng tìm kiếm thông minh



Hình 2.14. Biểu đồ tuần tự tìm kiếm thông minh

2.4 Thiết kế cơ sở dữ liệu

2.4.1 Danh sách các bảng

Bảng dữ liệu organization_unit dùng để quản lý thông tin về các đơn vị tổ chức, như phòng ban, chi nhánh, bộ phận và tổ chức dữ liệu nhân sự và tài liệu.

Bảng 2.3. Bảng organization_unit

| Thuộc tính | Kiểu dữ liệu | Mô tả |
|----------------------|--------------|----------------------|
| OrganizationUnitID | Int | ID của đơn vị |
| OrganizationUnitName | Char | Tên của đơn vị |
| ParentID | Int | ID của đơn vị cha |
| Address | Char | Địa chỉ đơn vị |
| Code | Char | Mã đơn vị |
| Status | Boolean | Trạng thái hoạt động |

Bảng dữ liệu employee dùng để lưu trữ dữ liệu về nhân viên như thông tin cá nhân, tổ chức công tác, vị trí, mã nhân viên.

Bảng 2.4. Bảng employee

| Thuộc tính | Kiểu dữ liệu | Mô tả |
|--------------|--------------|-------------------------|
| EmployeeID | Int | ID nhân viên |
| EmployeeCode | Char | Mã code nhân viên |
| EmployeeName | Char | Họ và tên nhân viên |
| Email | Char | Email nhân viên |
| Birthday | Datetime | Ngày tháng năm sinh của |

| | | |
|----------------------|--------|-------------------------------------------|
| | | nhân viên |
| Gender | Char | Giới tính nhân viên |
| Avatar | Char | Ảnh đại diện |
| Phone | String | Số điện thoại nhân viên |
| Address | String | Địa chỉ nhân viên |
| JobPositionName | String | Tên chức vụ |
| OrganizationUnitID | Int | ID của đơn vị nhân viên đang làm việc |
| OrganizationUnitName | Char | Tên của đơn vị nhân viên đang làm việc |

Bảng dữ liệu file dùng để lưu trữ thông tin về tài liệu/tập tin được tạo, quản lý và chia sẻ trong tổ chức.

Bảng 2.5. Bảng file

| Thuộc tính | Kiểu dữ liệu | Mô tả |
|------------|--------------|------------------------|
| FileID | Int | ID của file |
| FileName | Char | Tên file |
| ParentID | Int | ID của thư mục cha |
| TypeFile | Char | Loại file |
| Path | Char | Đường dẫn lưu trữ file |
| Content | Char | Nội dung của file |

| | | |
|----------------------|----------|------------------------------------------|
| CreatedBy | Int | ID của nhân viên sở hữu file |
| CreatedDate | Datetime | Ngày tạo file |
| SharedIDs | Int | ID của tài khoản được chia sẻ file |
| SharedDate | Datetime | Ngày chia sẻ file |
| Note | Char | Ghi chú |
| NoteShared | Char | Ghi chú chia sẻ |
| Size | Bigint | Kích thước file |
| OrganizationUnitID | Int | ID của đơn vị (nếu là file của đơn vị) |
| OrganizationUnitName | Char | Tên của đơn vị (nếu là file của đơn vị) |

Bảng dữ liệu role dùng để quản lý các vai trò người dùng trong website.

Bảng 2.6. Bảng role

| Thuộc tính | Kiểu dữ liệu | Mô tả |
|------------|--------------|-----------------|
| RoleID | Int | ID vai trò |
| RoleCode | Char | Mã vai trò |
| RoleName | Char | Tên vai trò |
| Note | Char | Ghi chú vai trò |

Bảng dữ liệu role_detail quản lý các chi tiết hành động (permission) cho từng vai trò (role), như “Xem”, “Thêm”, “Xóa”, “Tải xuống”, “Chia sẻ” cho từng trang website.

Bảng 2.7. Bảng role_detail

| Thuộc tính | Kiểu dữ liệu | Mô tả |
|--------------|--------------|----------------------------------------------|
| RoleDetailID | Int | ID của chi tiết vai trò |
| RoleID | Int | ID vai trò sở hữu quyền |
| SubCode | Char | Mã cho từng loại quyền |
| SubName | Char | Tên mô tả cho quyền |
| Action | Char | Hành động cụ thể mà quyền cho phép thực hiện |
| ActionName | Char | Mô tả hành động tương ứng |
| Value | boolean | Kích hoạt quyền |

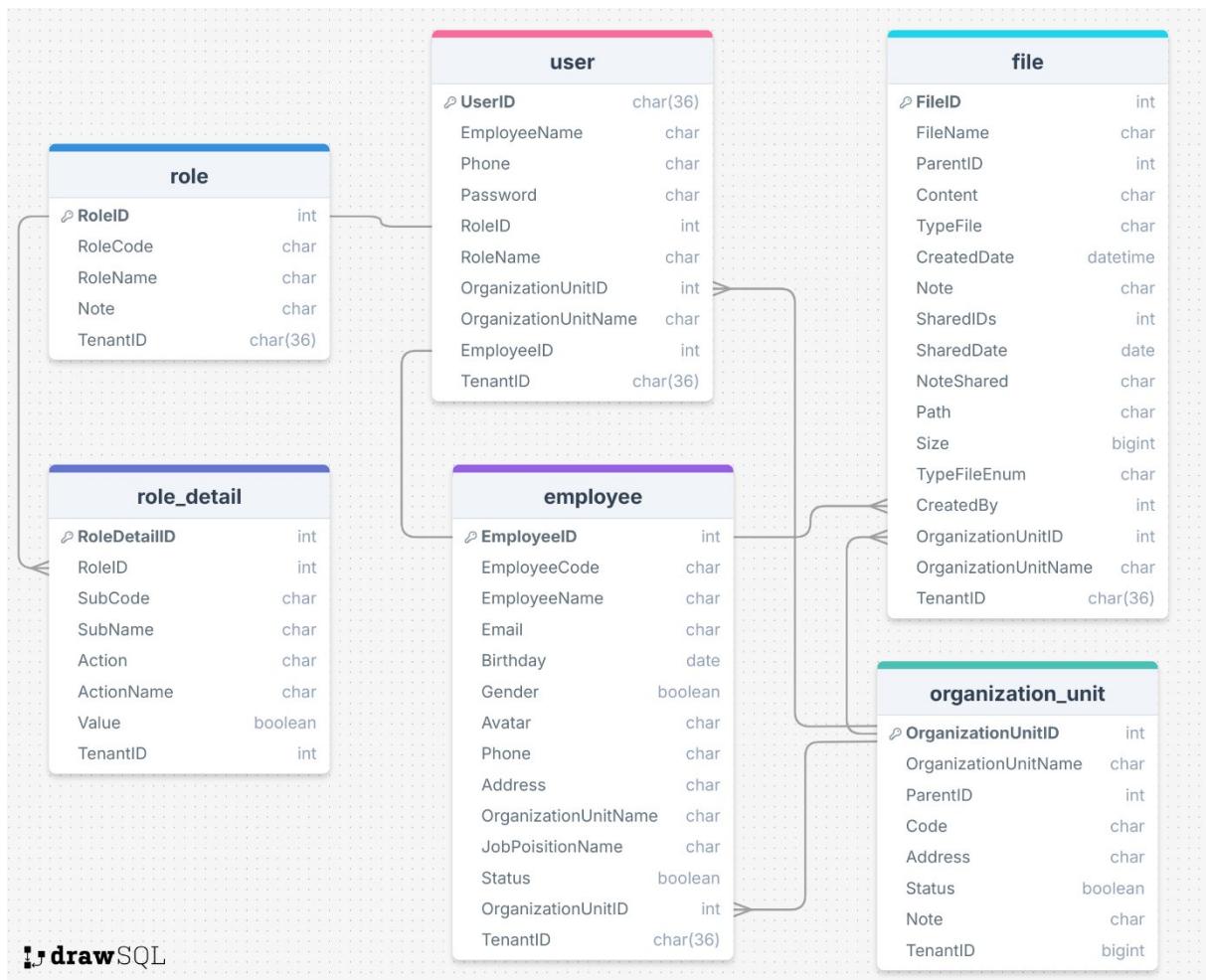
Bảng dữ liệu user dùng để quản lý tài khoản người dùng hệ thống. Thường liên kết với nhân viên để phân quyền đăng nhập và thao tác.

Bảng 2.8. Bảng user

| Thuộc tính | Kiểu dữ liệu | Mô tả |
|------------|--------------|-------------------------|
| UserID | Char | ID của người dùng |
| RoleID | Int | ID vai trò sở hữu quyền |
| SubCode | Char | Mã cho từng loại quyền |
| SubName | Char | Tên mô tả cho quyền |

| | | |
|------------|---------|----------------------------------------------|
| Action | Char | Hành động cụ thể mà quyền cho phép thực hiện |
| ActionName | Char | Mô tả hành động tương ứng |
| Value | Boolean | Kích hoạt quyền |

2.4.2 Mô hình cơ sở dữ liệu quan hệ



Hình 2.15. Mô hình cơ sở dữ liệu quan hệ

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

3.1 Cài đặt chương trình

3.1.1 Fontend

Vue.js Core: Framework chính để xây dựng giao diện. Cho phép tạo component và quản lý dữ liệu reactive.

Vue Router: Quản lý điều hướng giữa các trang, tạo các đường dẫn URL cho từng trang trong website

Vuex: Lưu trữ và cập nhật state tập trung.

Element UI: Cung cấp buttons, tables, forms, modals.

Axios: Xử lý các request GET, POST, PUT, DELETE.

Vue Doc Preview: Hiển thị tài liệu trực tiếp trong website.

3.1.2 Backend

Sử dụng framework ASP.NET Core API với phiên bản .NET 8.0 cùng các thư viện:

- Microsoft.AspNetCore: Để xây dựng các controller cho website web/API và làm việc với yêu cầu HTTP và truy cập thông tin về ngữ cảnh HTTP.
- DocumentFormat.OpenXml: Để làm việc và xử lý các tệp Office
- Nest: Thư viện cho Elasticsearch, sử dụng IElasticClient để thực hiện các thao tác với Elasticsearch.
- Dapper: Dùng để tương tác với cơ sở dữ liệu một cách hiệu quả hơn thông qua các truy vấn SQL.
- Microsoft.IdentityModel.Tokens: Dùng để tạo và xác thực JWT .
- System.IdentityModel.Tokens.Jwt: Dùng quản lý JWT.
- System.Net.Mail: Để gửi email thông qua SMTP.

3.1.3 Cơ sở dữ liệu

MySQL: hệ quản trị cơ sở dữ liệu quan hệ phổ biến, lưu trữ hầu hết các thông tin của hệ thống.

Elasticsearch: hệ quản trị cơ sở dữ liệu NoSQL hướng tài liệu, dùng để lưu trữ, quản lý tài liệu và đặc biệt mạnh về tìm kiếm toàn văn siêu nhanh, theo thời gian thực.

3.1.4 Một số hàm xử lý chính

3.1.4.1 Hàm xác thực tài khoản

Hàm Authenticate có nhiệm vụ xác thực người dùng khi đăng nhập bằng cách kiểm tra thông tin email hoặc số điện thoại và mật khẩu trong cơ sở dữ liệu. Nếu thông tin không hợp lệ, hàm trả về null. Nếu hợp lệ, hệ thống sẽ tạo một mã xác thực (JWT token) thông qua hàm generateJwtToken. Token này chứa thông tin người dùng, được ký bằng khóa bảo mật và có hiệu lực trong 7 ngày. Kết quả trả về là thông tin người dùng kèm theo token, giúp người dùng truy cập các chức năng của hệ thống mà không cần đăng nhập lại trong thời gian hiệu lực.

```
2 references
public async Task<AuthenticateResponse> Authenticate(AuthenticateRequest model)
{
    //Kiểm tra dưới database có tài khoản này không
    string sql = $"SELECT * FROM user WHERE (Email = '{model.Username}' OR Phone = '{model.Username}') AND Password = '{model.Password}'";
    var res = await QueryCommandTextAsync<User>(sql);
    // return null if user not found
    if (res == null) return null;
    // authentication successful so generate jwt token
    var token = generateJwtToken((User)res);
    return new AuthenticateResponse((User)res, token);
}

1 reference
private string generateJwtToken(User user)
{
    // generate token that is valid for 7 days
    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(_appSettings.Secret);
    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(new[] { new Claim("UserID", user.UserID.ToString()) }),
        Expires = DateTime.UtcNow.AddDays(7),
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };
    var token = tokenHandler.CreateToken(tokenDescriptor);
    return tokenHandler.WriteToken(token);
}
```

Hình 3.1. Hàm xác thực Authenticate và hàm tạo token đăng nhập generateJwtToken

3.1.4.2 Hàm sử dụng công nghệ Elasticsearch

Website sử dụng Elasticsearch để lưu trữ và tìm kiếm dữ liệu, trong đó hàm AddElasticsearch thực hiện cấu hình và khởi tạo Elasticsearch client bằng cách đọc các thông tin như cloudId, tài khoản đăng nhập và chỉ mục mặc định từ file cấu hình. Nếu thiếu các thông tin này, hệ thống sẽ báo lỗi. Sau khi cấu hình thành công, client được đăng ký dưới dạng singleton để có thể sử dụng trong toàn bộ ứng dụng.

```

1 reference
public static void AddElasticsearch(this IServiceCollection services, IConfiguration configuration)
{
    services.AddSingleton<IElasticClient>(sp =>
    {
        var config = sp.GetRequiredService< IConfiguration>();

        // Lấy thông tin từ cấu hình
        var cloudId = config["elasticsearch:cloudId"]; // Cloud ID từ Elasticsearch Cloud
        var username = config["elasticsearch:user"]; // Tên người dùng
        var password = config["elasticsearch:password"]; // Mật khẩu
        var defaultIndex = config["elasticsearch:index"]; // Tên chỉ mục mặc định

        // Kiểm tra nếu thiếu thông tin
        if (string.IsNullOrEmpty(cloudId) || string.IsNullOrEmpty(username) || string.IsNullOrEmpty(password))
        {
            throw new ArgumentException("Elasticsearch configuration is missing required fields (cloudId, user, password).");
        }

        // Cấu hình client
        var settings = new ConnectionSettings(cloudId, new BasicAuthenticationCredentials(username, password))
            .DefaultIndex(defaultIndex) // Chỉ định chỉ mục mặc định
            .DefaultMappingFor<File>(m => m.IndexName(defaultIndex)); // Mapping mặc định cho File

        return new ElasticClient(settings);
    });
}

```

Hình 3.2. Hàm cấu hình Elasticsearch

Sau đó dựa trên client đã cấu hình, hàm Find cho phép thực hiện tìm kiếm tài liệu theo từ khóa. Cụ thể, nó gửi truy vấn tìm kiếm đến Elasticsearch, kiểm tra kết quả trả về và ghi log nếu có lỗi. Nếu truy vấn thành công, hàm sẽ trích xuất và trả về danh sách các tệp tin (File) phù hợp với từ khóa được cung cấp.

```

1 reference
public async Task<List<Models.File>> Find(string query)
{
    var response = await _elasticClient.SearchAsync<Models.File>(
        s => s.Query(q => q.Match(d => d.Query(query))));
    if (!response.IsValid)
    {
        _logger.LogError("Failed to search documents");
        return new List<Models.File>();
    }
    List<Models.File> files = new List<Models.File>();
    foreach(var hit in response.Hits)
    {
        files.Add(hit.Source);
    }
    return files;
}

```

Hình 3.3. Hàm tìm kiếm dữ liệu trên Elasticsearch

3.1.4.3 Hàm UploadFile

Hàm UploadFile có nhiệm vụ tiếp nhận và xử lý một tệp tin do người dùng tải lên. Đầu tiên, tệp được lưu vào thư mục Resources/Document trong máy chủ. Sau đó, dựa vào định dạng của tệp (như .docx, .xlsx, .pdf, .txt), hệ thống đọc nội dung bên trong bằng các hàm chuyên biệt. Đồng thời, xác định loại tệp (Word, Excel, v.v.) lưu về một giá trị mã hóa (typeTypeEnum). Sau khi xử lý xong, tệp được tải lên index của Elasticsearch. Kết quả trả về là một đối tượng File chứa thông tin mô tả, nội dung, định dạng và dung lượng tệp đã tải lên.

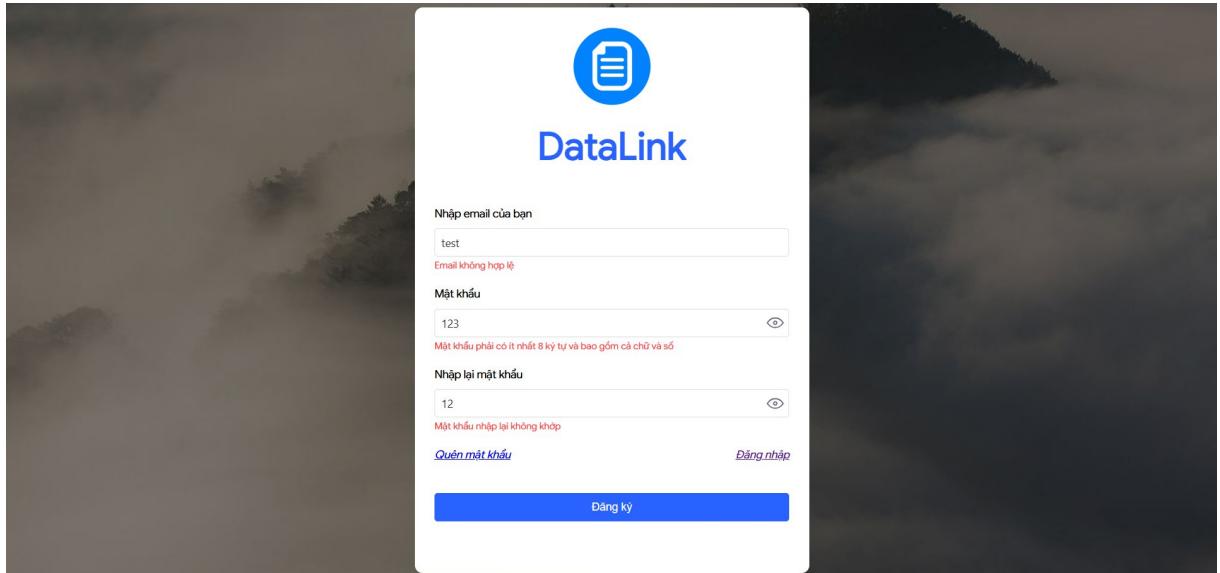
```
public object UploadFile(IFormFile file)
{
    var folderName = Path.Combine("Resources", "Document");
    var pathToSave = Path.Combine(Directory.GetCurrentDirectory(), folderName);
    if (file.Length > 0)
    {
        var fileName = ContentDispositionHeaderValue.Parse(file.ContentDisposition).FileName.Trim('"');
        var listName = fileName.Split('.').ToList();
        string typeFile = listName[listName.Count - 1].ToString().ToUpper();
        var fullPath = Path.Combine(pathToSave, fileName);
        var dbPath = Path.Combine(folderName, fileName);
        using (var stream = new FileStream(fullPath, FileMode.Create)) { file.CopyTo(stream); }
        string content = string.Empty;
        string type = string.Empty;
        int typeTypeEnum = 0;
        switch (typeFile)...
        string _path = string.Empty;
        var drivePath = _googleBL.UploadFile(fullPath);
        Models.File fileResponse = new Models.File()
        {
            FileName = fileName,
            TypeFile = type,
            Path = drivePath,
            Size = (int)file.Length,
            TypeTypeEnum = typeTypeEnum,
            Content = content
        };
        return fileResponse;
    }
}
```

Hình 3.4. Hàm UploadFile

3.2 Giao diện chương trình

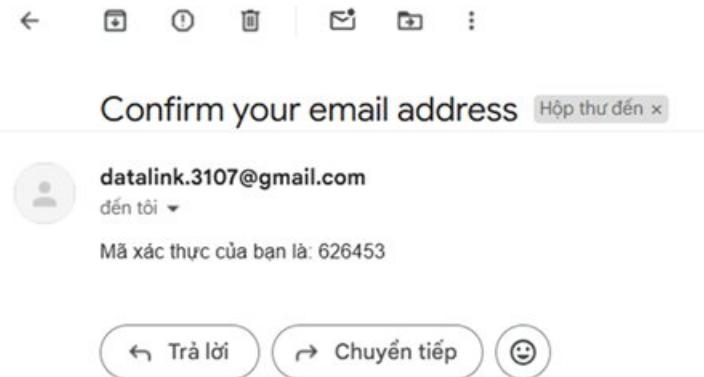
3.2.1 Chức năng tài khoản

Để truy cập được vào website, người dùng cần thực hiện đăng ký tài khoản và đây là tài khoản với quyền hạn cao nhất trong website.



Hình 3.5. Form đăng kí tài khoản

Để đăng kí được thì người dùng phải nhập chính xác thông tin và sau đó sẽ nhận được email có xác thực.



Hình 3.6. Thư lấy mã xác thực email

Sau khi nhập mã xác thực, bước cuối cùng là nhập đầy đủ thông tin và bắt đầu sử dụng website.

The screenshot shows a registration form titled "DataLink". The fields include:

- Họ và tên (Name): Nhập họ và tên, required.
- Số điện thoại (Phone Number): test, required. Error message: Số điện thoại không hợp lệ. Phải bắt đầu bằng 0 (10 số) hoặc 84 (11 số).
- Ngày sinh (Date of Birth): 12 / mm / yyyy, required. Error message: Vui lòng chọn ngày sinh.
- Giới tính (Gender): Chọn giới tính, required. Error message: Vui lòng chọn giới tính.
- Tên công ty (Company Name): Nhập tên công ty, required. Error message: Vui lòng nhập tên công ty.
- Chức vụ (Position): Nhập chức vụ, required. Error message: Vui lòng nhập chức vụ.
- [Quay lại đăng nhập](#) (Forgot password link).

Hình 3.7. Form điền thông tin để hoàn tất đăng kí

3.2.2 Chức năng quản lý file

3.2.2.1 File cá nhân

Tại đây người có thể quản lý file mà chính mình đã tải lên.

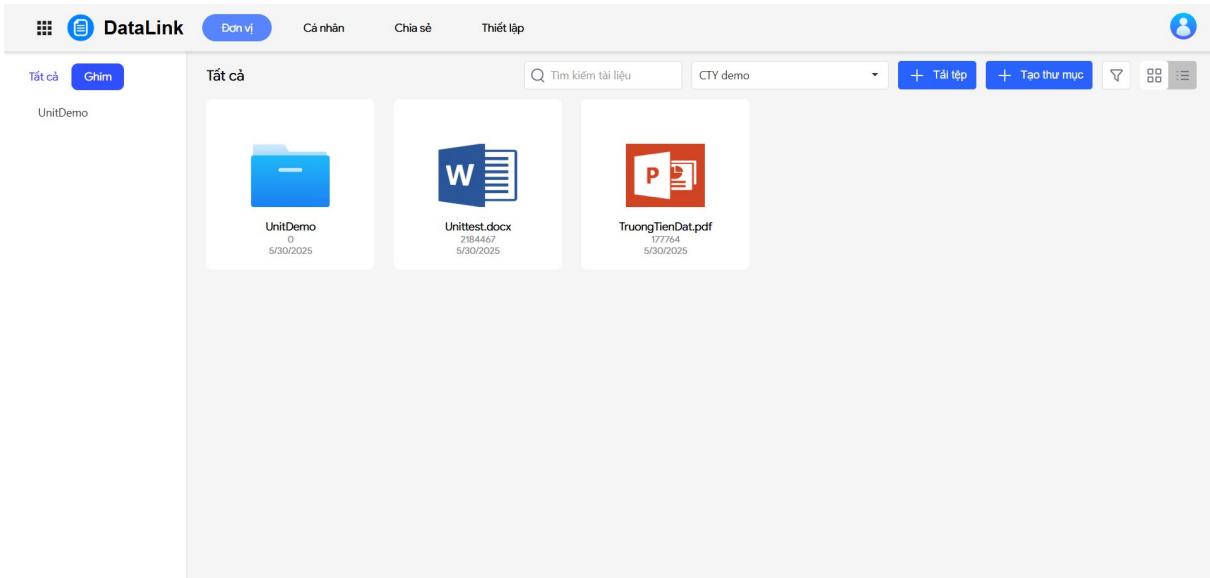
The screenshot shows the DataLink file management interface. The files listed are:

- Thư mục demo (0kb) 5/30/2025
- drawSQL-image-export-2... (506455kb) 5/30/2025
- CNTT1_K62_211204093_Lê... (201690kb) 5/29/2025
- TruongTienDat.pdf (177764kb) 5/29/2025
- btvn.txt (8kb) 5/30/2025

Hình 3.8. Giao diện quản lý file cá nhân

3.2.2.2 File đơn vị

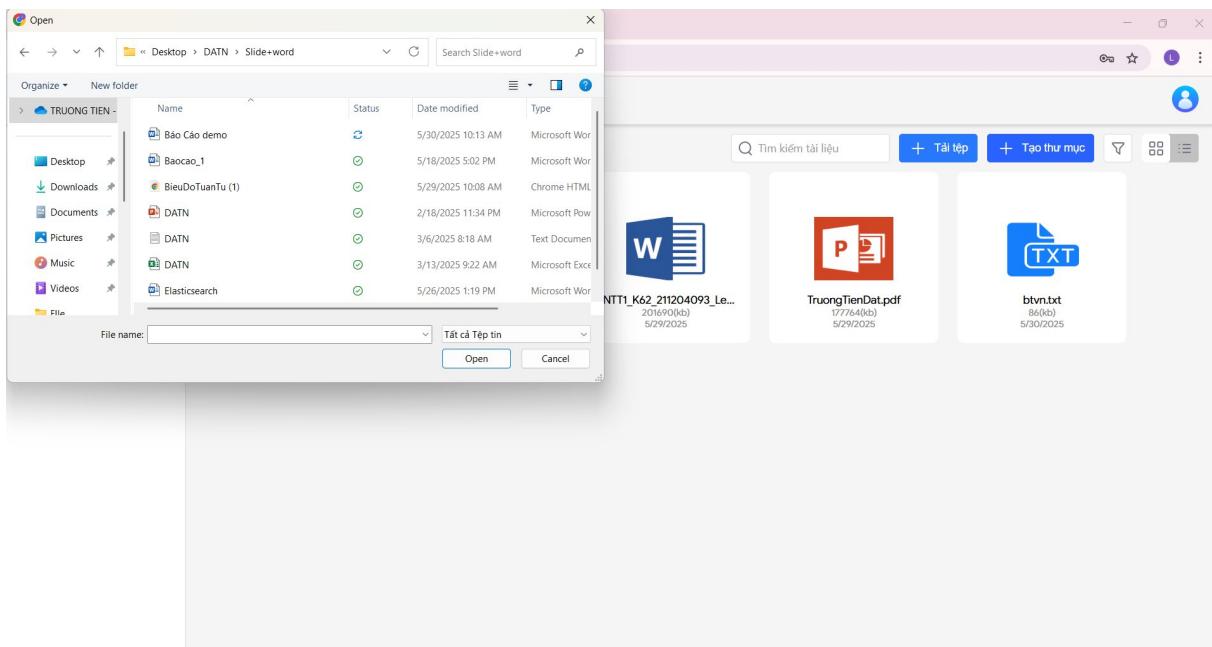
Đây là trang mà các người dùng xem và quản lý file có cùng đơn vị.



Hình 3.9. Giao diện quản lý file đơn vị

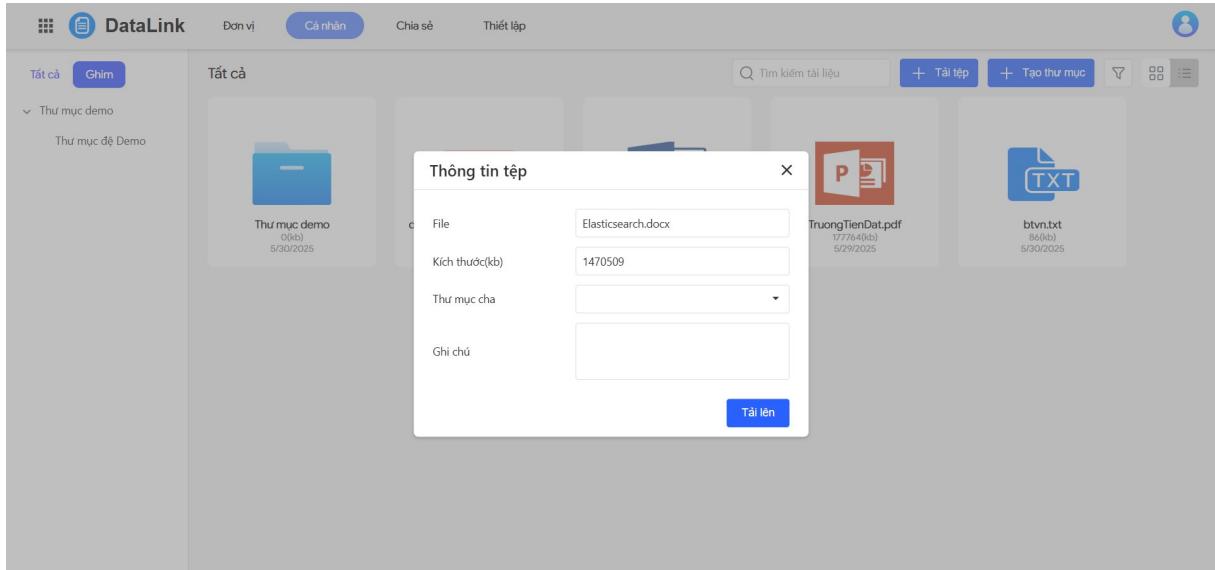
3.2.2.3 Tải lên tài liệu

Người dùng có thể chọn file từ máy tính cá nhân để tải lên bằng cách click vào **tải tệp**.

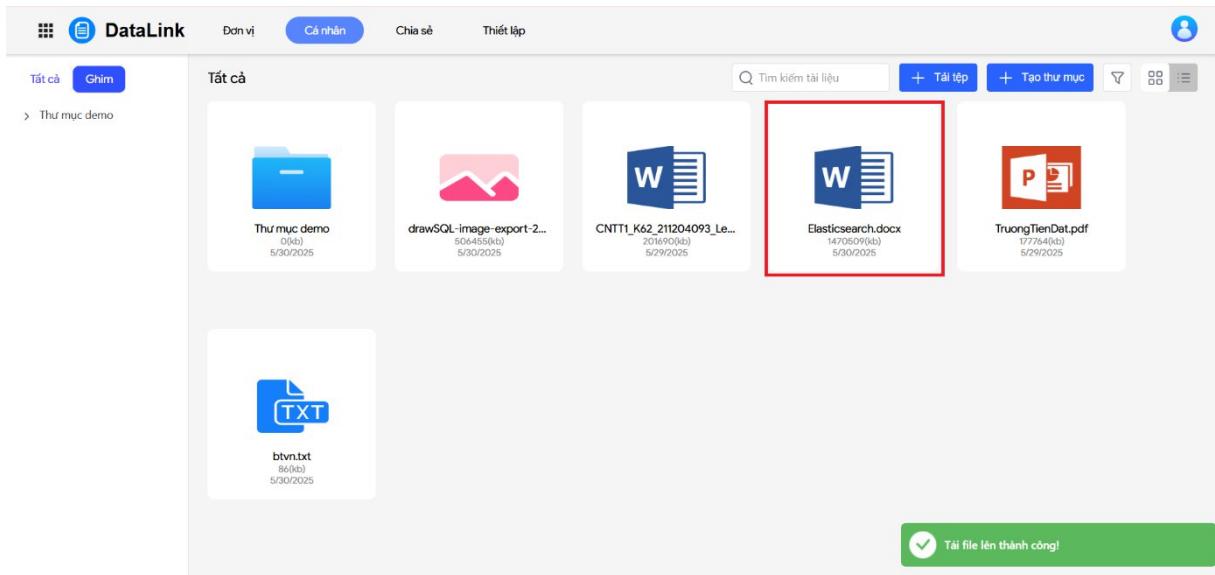


Hình 3.10. Giao diện chọn file để tải lên website

Sau khi chọn thành công, người dùng click vào Open và xác nhận tải file lên website.

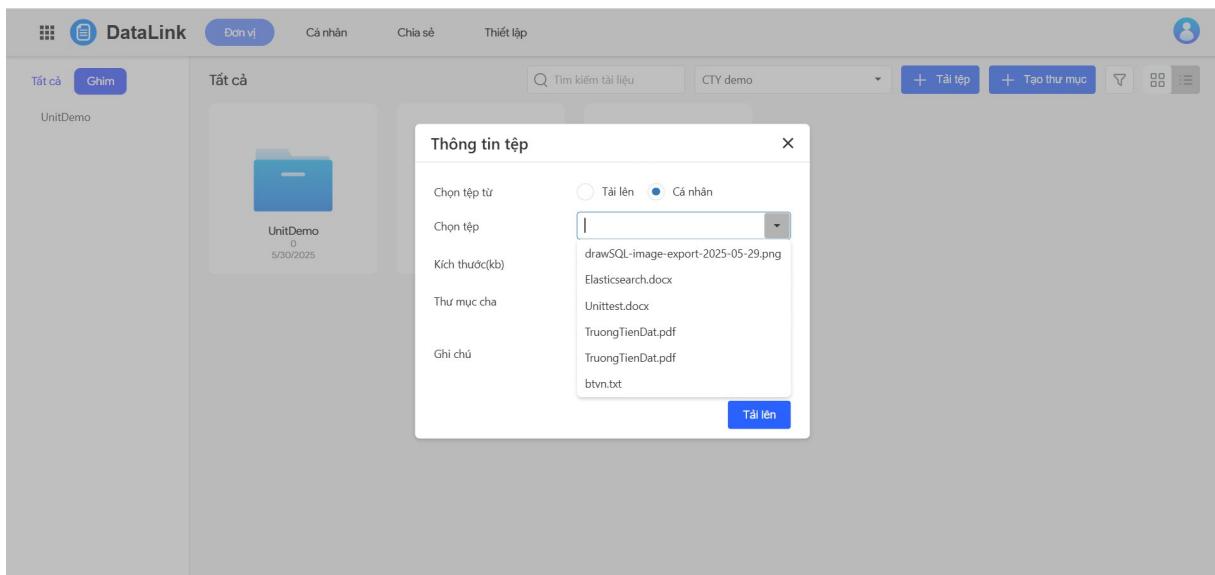


Hình 3.11. Giao diện sau khi chọn file tải lên



Hình 3.12. Giao diện tải file lên thành công

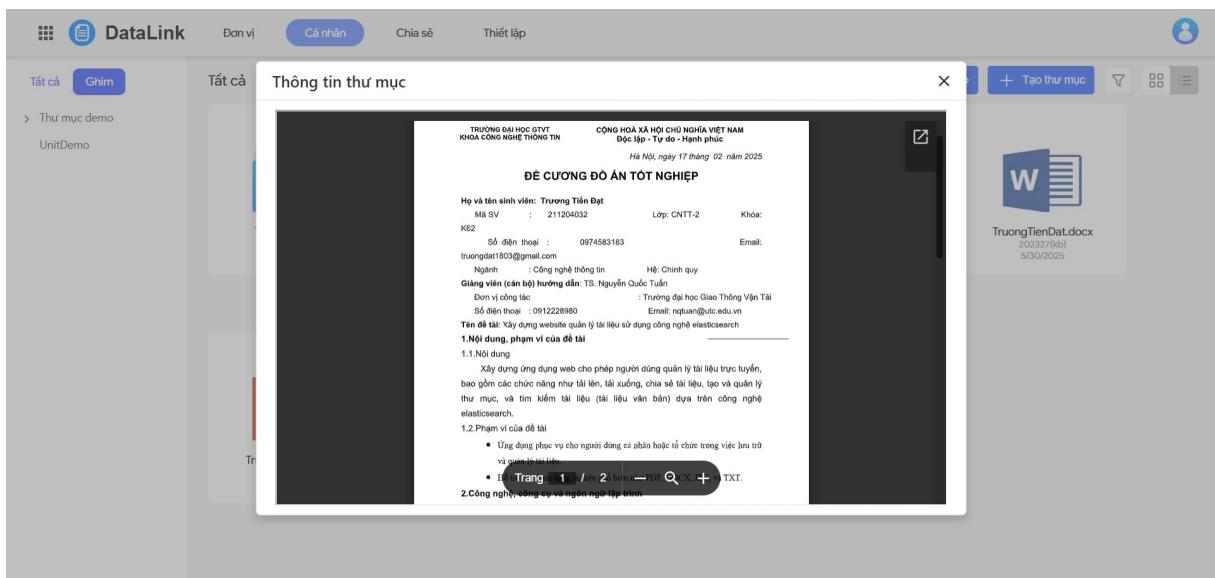
Người dùng tải lên file ở trang đơn vị thì có hai cách là tải lên từ máy tính cá nhân hoặc tải lên từ trang cá nhân của mình. Nếu là tải lên từ máy tính cá nhân thì giống như các bước ở trên, còn tải lên từ trang cá nhân thì người dùng chỉ cần chọn file và xác nhận tải lên.



Hình 3.13. Giao diện tải file của trang đơn vị

3.2.2.4 Xem trước file

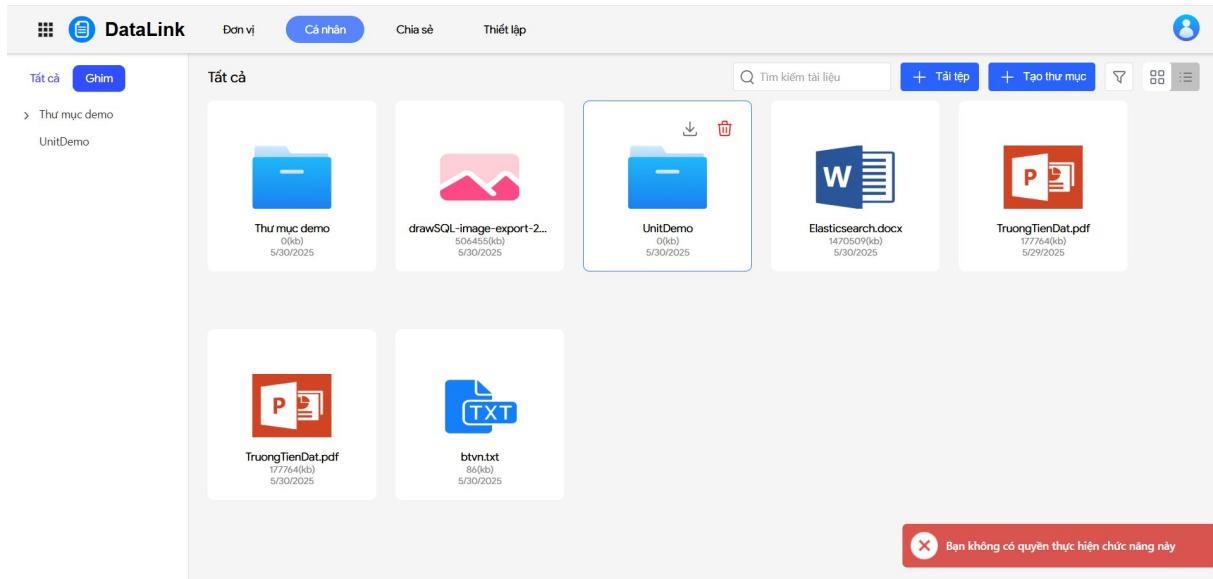
Người dùng có thể xem trước file bằng cách nhấp vào file, website sẽ tự động mở file để hiển thị nội dung.



Hình 3.14. Giao diện xem nội dung file

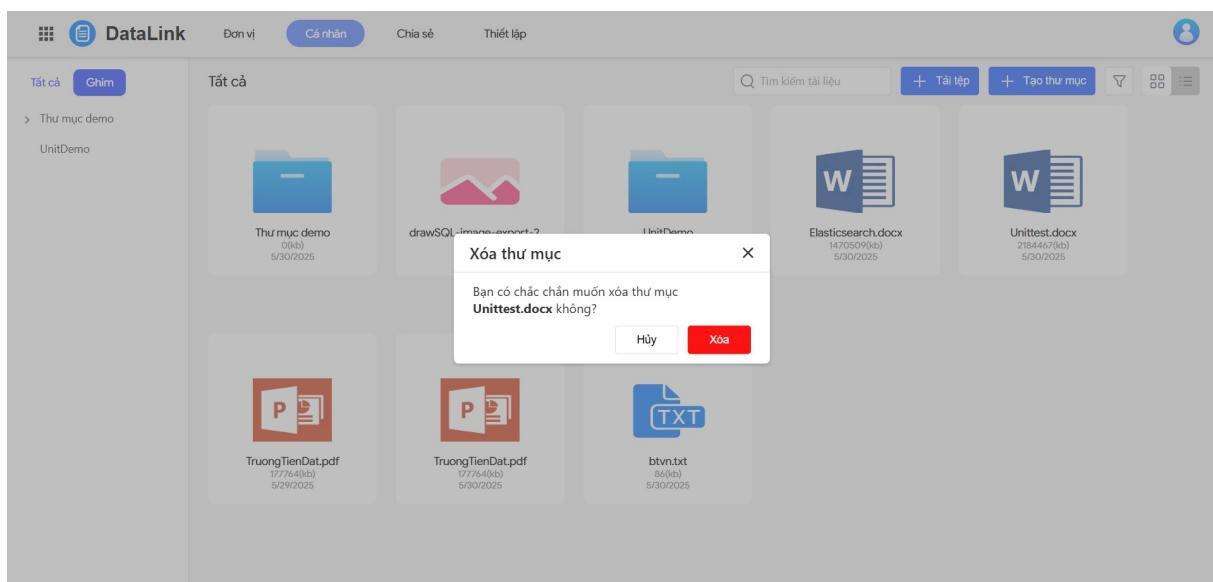
3.2.2.5 Xóa tài file

Người dùng chọn file cần xóa và click vào icon delete để mở thông báo xác nhận. Nếu người dùng không có quyền xóa thì website sẽ thông báo như sau:



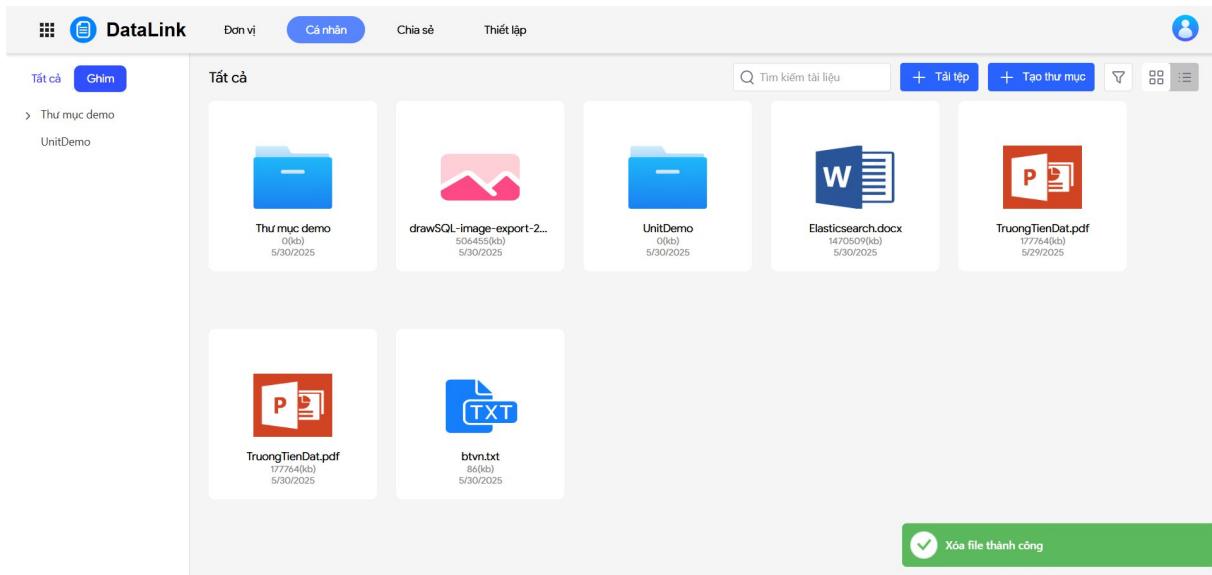
Hình 3.15. Thông báo không có quyền xóa file

Ngược lại nếu người dùng có đủ quyền thì sẽ hiển thị thông báo xác nhận xóa.



Hình 3.16. Dialog xác nhận xóa file.

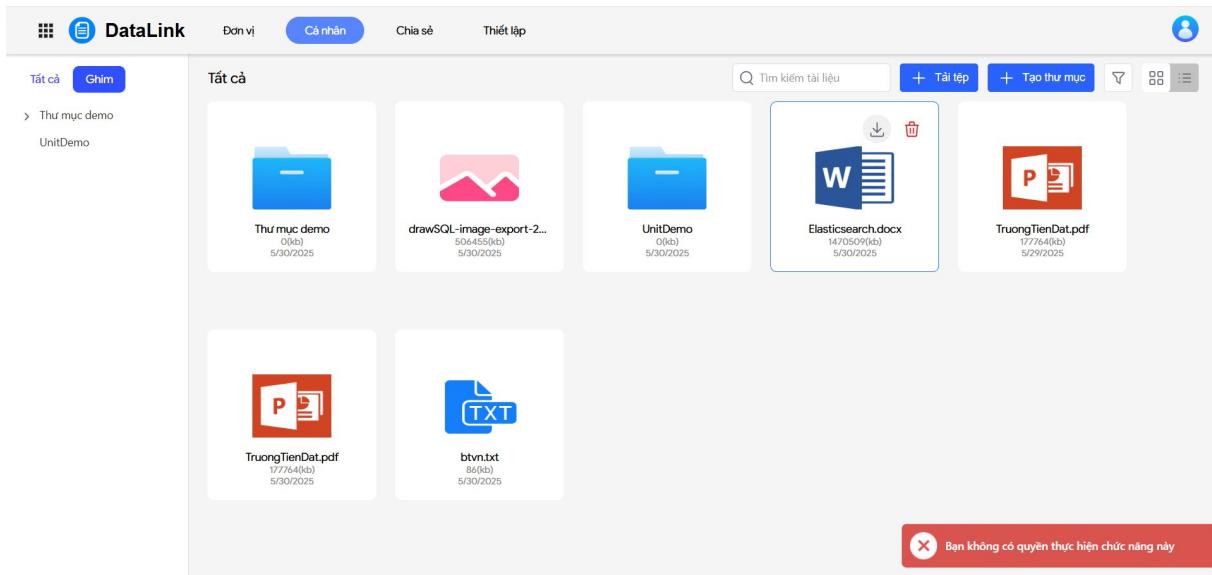
Sau khi xác nhận xóa thì website sẽ thông báo cho người dùng kết quả xóa file



Hình 3.17. Thông báo xóa file thành công

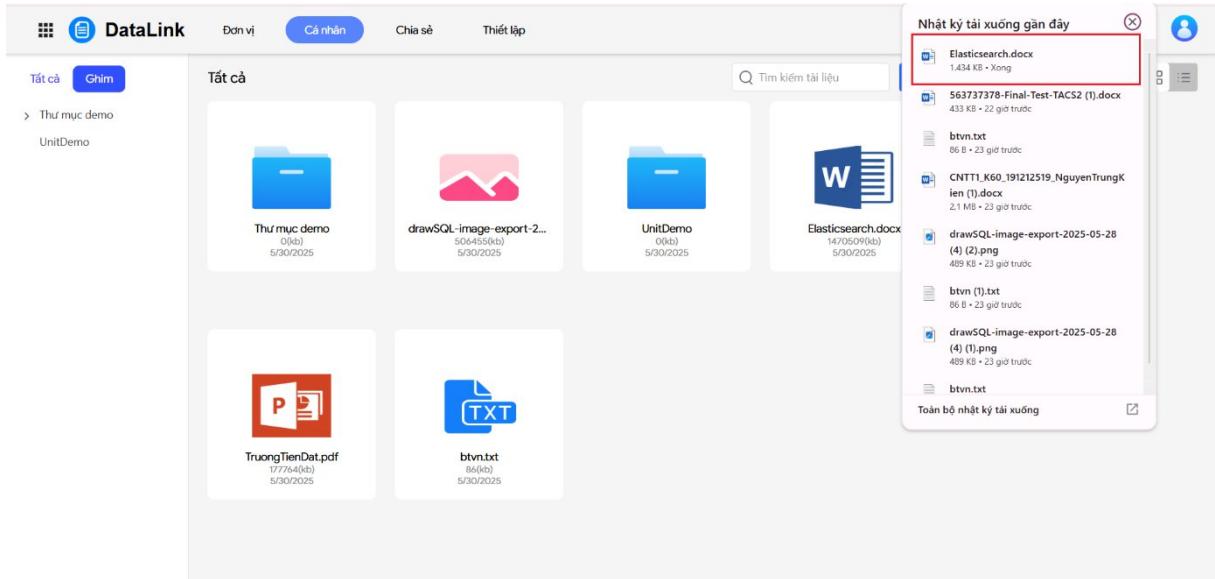
3.2.2.6 Tải xuống file

Người dùng chọn file cần tải xuống và click vào icon download. Nếu người dùng không có quyền tải xuống thì website sẽ thông báo.



Hình 3.18. Thông báo không có quyền tải file

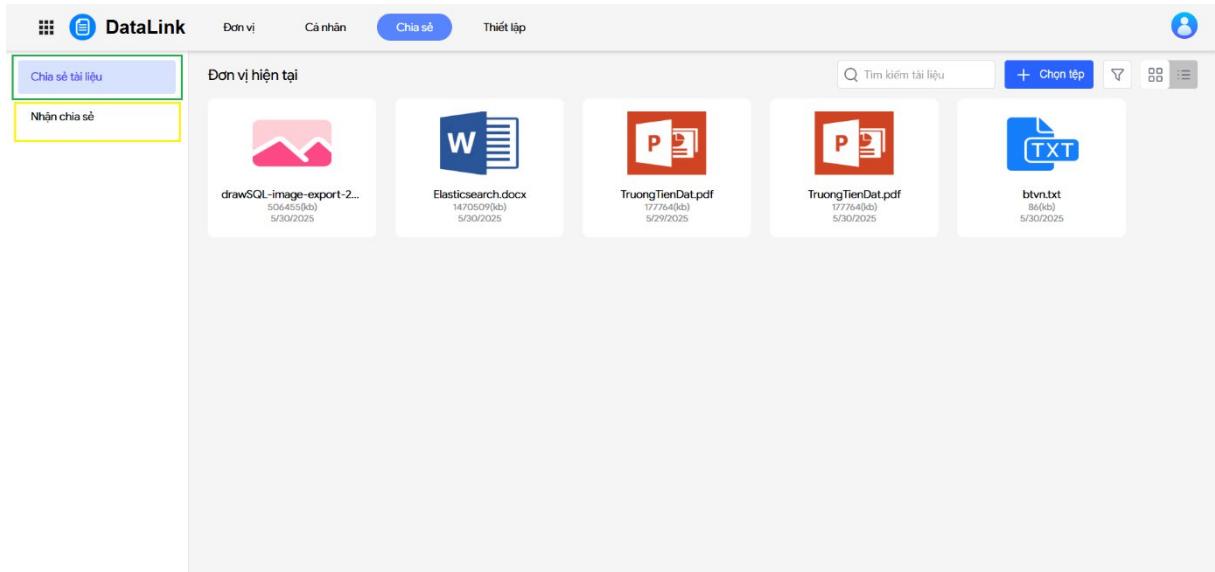
Nếu người dùng có quyền thì file sẽ được tải xuống ở thư mục download của máy tính cá nhân.



Hình 3.19. Kết quả sau khi tải file

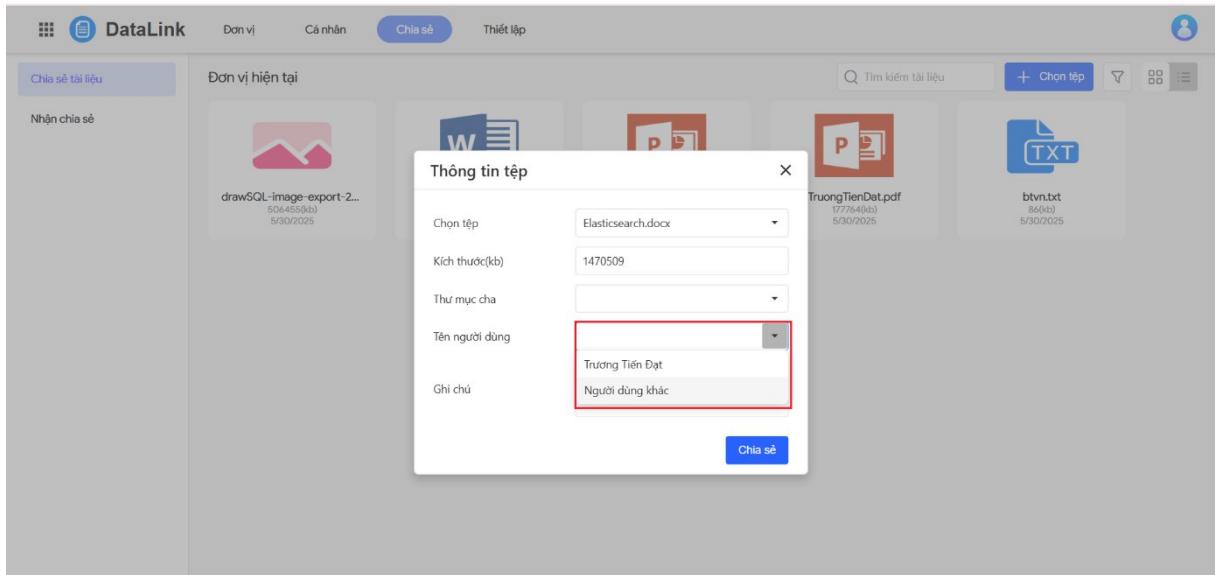
3.2.2.7 Chia sẻ tài liệu

Ở trang này sẽ có 2 trang nhỏ là: Chia sẻ file và nhận file được chia sẻ.



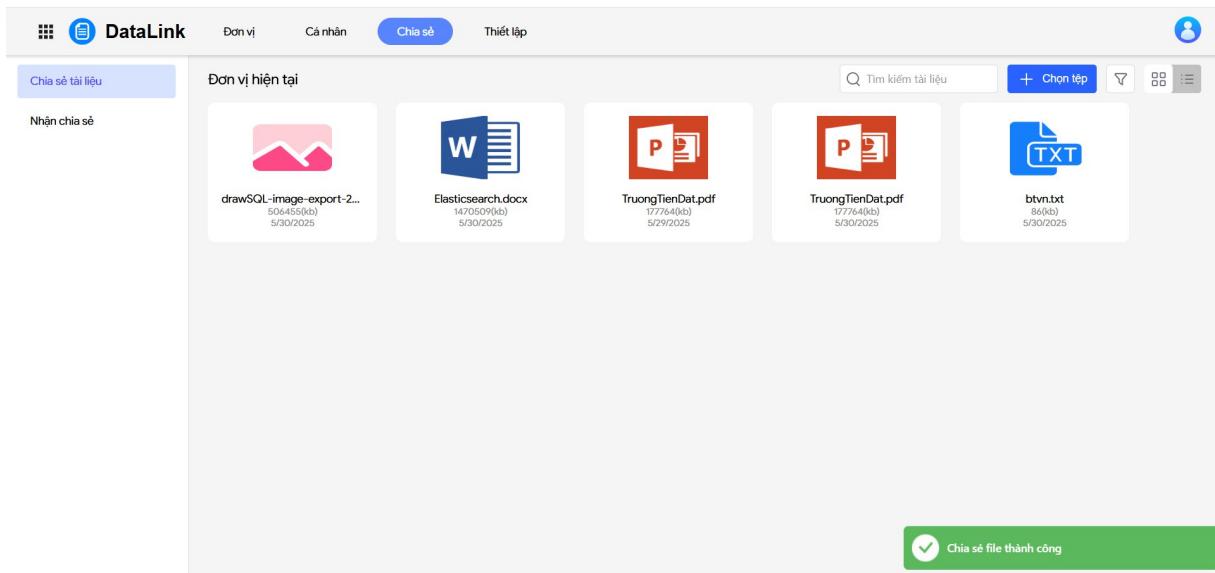
Hình 3.20. Giao diện trang chia sẻ tài liệu

Nếu người dùng muốn chia sẻ tài liệu, hãy nhấn “Chọn tệp” và chọn file cần chia sẻ. Tiếp theo, chọn người dùng mà bạn muốn chia sẻ file.



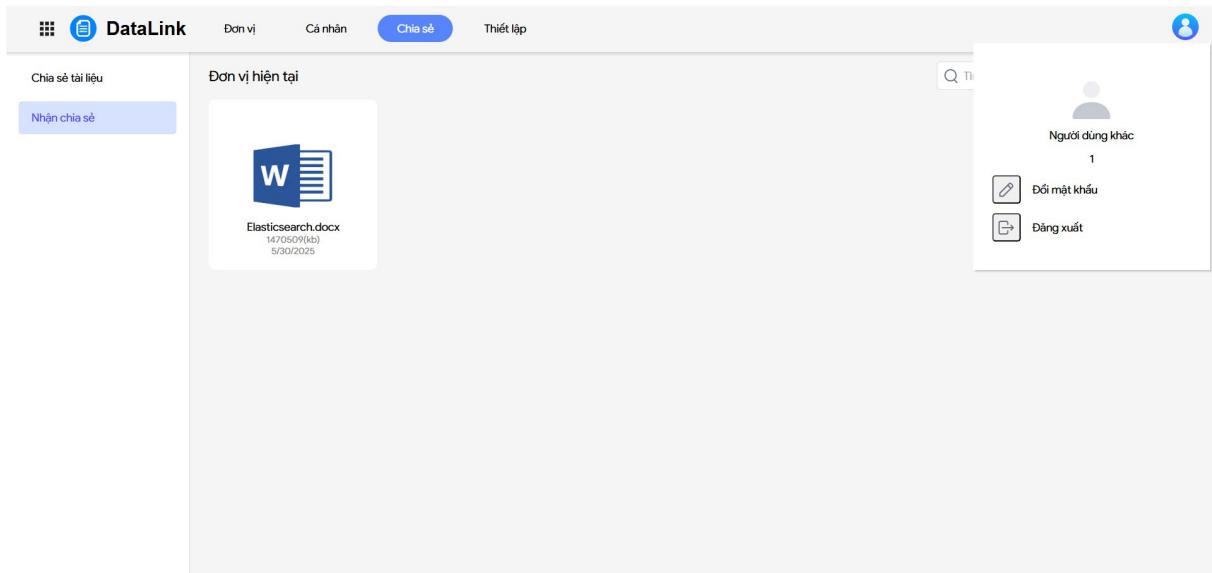
Hình 3.21. Dialog chọn file để chia sẻ cho người dùng khác

Sau đó, nhấn “Chia sẻ”, website sẽ cấp quyền xem file cho người dùng được chọn.



Hình 3.22. Thông báo chia sẻ thành công.

Đây là kết quả hiển thị của tài khoản “Người dùng khác” sau khi file được chia sẻ thành công.



Hình 3.23. File đã được chia sẻ với tài khoản của người dùng mới được thêm.

3.2.3 Chức năng tìm kiếm toàn văn

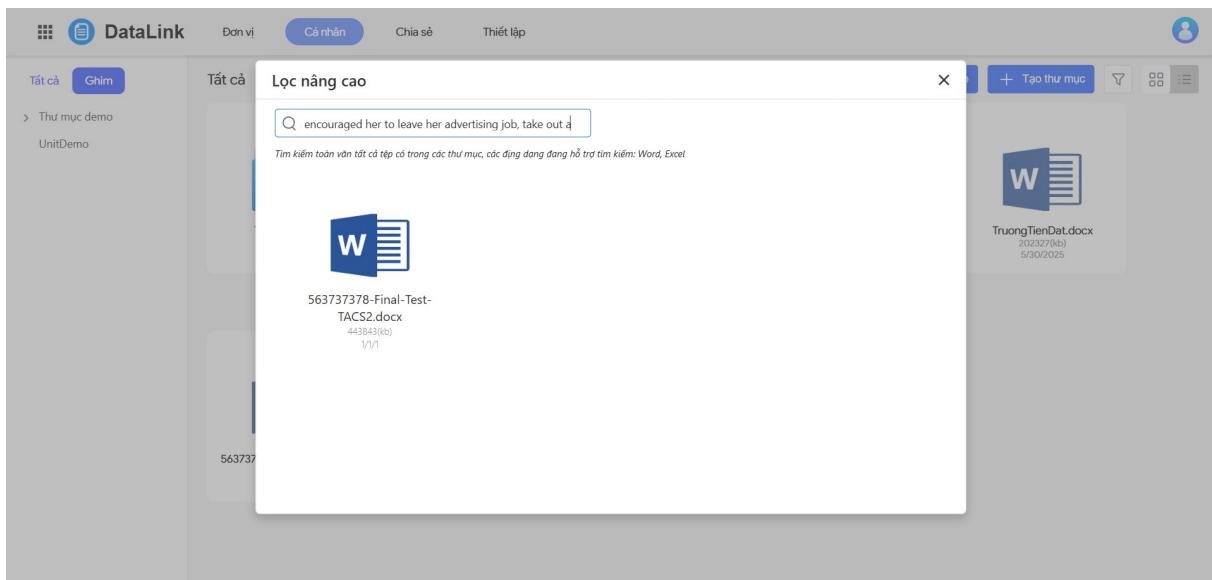
Trước khi thực hiện chức năng này, em có tải lên một file có nội dung như sau:

The Young Achiever of the Year
Kal Kaur Rai has always been interested in fashion and has just won the title of Young Achiever of the Year at the Asian Business Awards. Ever since she was a child, she has drawn clothes and designed patterns. She never told her hard-working parents, who own a supermarket, that she wanted to turn her hobby into a career. She thought they expected her to go into a more established business, so she went to university to do a management degree. After university, she moved to London and worked in an advertising agency. She had to attend industry events but couldn't afford the designer clothes she liked. She started making skirts and tops for herself. When her friends saw her clothes, they asked her to make things for them. She then found a small shop in London willing to take her designs on a sale-or-return basis. They were very popular and nothing came back. This encouraged her to leave her advertising job, take out a \$20,000 loan and begin her own womenswear label. Kal's parents were not angry about her career change and said they would support her, which really pleased her. Her clothes are now on sale in over 70 stores and her business has an income of over €500,000. Her clothes appear in fashion magazines, she designs for pop stars and she has just gained public recognition by winning this award. Her business has come a long way and she knows she is extremely lucky. 'What I do is my hobby — and I get paid for it! But remember, I've worked hard for this.'

16. What is the writer trying to do in the text?
A encourage fashion designers to make better business plans.
B compare a job in fashion with other choices of career.

Hình 3.24. Nội dung file vừa tải lên.

Sau đó, ta copy một đoạn nội dung văn bản được bôi đen như trên làm từ khóa tìm kiếm. Tiếp theo click vào biểu tượng góc phải màn hình để mở bộ lọc tìm kiếm thông minh. Nhập nội dung cần tìm và nhấn Enter, kết quả tìm kiếm sẽ được hiển thị bên dưới.



Hình 3.25. Danh sách file có chứa từ khóa vừa tìm kiếm

3.2.4 Chức năng quản lý tài khoản

Ở trang này, người dùng có thể quản lý “Thông tin nhân viên”, “Đơn vị công tác”, “Tài khoản người dùng” và “Vai trò”. Người dùng được phép xem danh sách, thêm, chỉnh sửa hoặc xóa nhân viên, đơn vị, tài khoản và vai trò.

The screenshot shows the "Danh sách nhân viên" (Employee List) screen. On the left sidebar, there are three categories: "Thông tin nhân viên" (selected), "Đơn vị công tác", and "Tài khoản người dùng". The main area displays a table with columns: Mã nhân viên, Tên nhân viên, Đơn vị, Ngày sinh, Giới tính, SĐT, and Email. Two employees are listed: NV202 (Trương Tiến Đạt) and NVtest (Người dùng khác). At the top right, there is a success message "Thành công" with a checkmark icon. A search bar "Tim kiem nhân viên" and a "Thêm" button are also visible.

| Mã nhân viên | Tên nhân viên | Đơn vị | Ngày sinh | Giới tính | SĐT | Email |
|--------------|-----------------|----------|-----------|-----------|------------|-------------------------|
| NV202 | Trương Tiến Đạt | CTY demo | 3/18/2003 | Nam | 0974583183 | datphuong0304@gmail.com |
| NVtest | Người dùng khác | CTY demo | 5/6/2025 | Nữ | 0123456789 | Test@gmail.com |

Hình 3.26. Danh sách nhân viên

The screenshot shows the DataLink application interface. On the left sidebar, under the 'Thống tin nhân viên' section, the 'Vai trò' button is highlighted. The main content area displays the 'Thêm nhân viên' (Add Employee) form. This form contains fields for basic information: Name (Tên nhân viên), Gender (Giới tính), Birthdate (Ngày sinh), Phone Number (Số điện thoại), Department (Đơn vị), Address (Địa chỉ), Employee ID (Mã nhân viên), and Notes (Ghi chú). There is also a placeholder for a profile picture. At the top right of the form are 'Hủy' (Cancel) and 'Lưu' (Save) buttons.

Hình 3.27. Form điền thông tin của nhân viên mới.

The screenshot shows the DataLink application interface. On the left sidebar, under the 'Thống tin nhân viên' section, the 'Vai trò' button is highlighted. The main content area displays the 'Thêm vai trò' (Add Role) form. It includes a field for the role name ('Tên vai trò') and a notes field ('Ghi chú'). Below these, there is a table titled 'Danh sách quyền chức năng' (List of function permissions) with two columns of checkboxes. The first column lists entities: 'Đơn vị', 'Cá nhân', 'Chia sẻ', 'Thiết lập/Nhân viên', 'Thiết lập/Đơn vị', 'Thiết lập/Người dùng', 'Thiết lập/Vai trò', and 'Thiết lập/Email hệ thống'. The second column contains four checkboxes each, labeled 'Xem', 'Thêm', 'Xóa', and 'Tải xuống'. To the right of each row of checkboxes are additional checkboxes for 'Thêm', 'Xóa', 'Sửa', and 'Xóa'.

Hình 3.28. Form thêm vai trò người dùng mới

KẾT LUẬN

Sau quá trình thực hiện đề tài, em đã có cơ hội củng cố và mở rộng thêm nhiều kiến thức về công nghệ, đặc biệt là cách triển khai và khai thác sức mạnh của Elasticsearch trong việc xây dựng hệ thống tìm kiếm toàn văn. Em cũng hiểu rõ hơn về quy trình phát triển một ứng dụng web hoàn chỉnh, từ thiết kế giao diện, xây dựng API, cho đến việc tích hợp công nghệ tìm kiếm vào hệ thống.

Trong quá trình xây dựng ứng dụng, em đã gặp phải một số khó khăn nhất định, như việc cấu hình Elasticsearch để tối ưu tốc độ tìm kiếm và độ chính xác khi xử lý dữ liệu văn bản tiếng Việt, cũng như việc đồng bộ dữ liệu giữa hệ quản trị cơ sở dữ liệu chính và Elasticsearch index.

Tuy nhiên, nhờ sự góp ý tận tình của thầy và sự hỗ trợ của bạn bè, em đã hoàn thành đề tài với các chức năng trọng tâm. Cụ thể, chức năng tìm kiếm tài liệu được cải tiến mạnh mẽ nhờ Elasticsearch, cho phép người dùng tìm nhanh các tài liệu liên quan dù chỉ nhớ một phần nội dung hoặc tiêu đề.

Dù vậy, phần mềm vẫn còn một số hạn chế như việc chưa có giao diện quản lý nâng cao dành cho admin, khả năng mở rộng quy mô dữ liệu chưa tối ưu khi triển khai trên môi trường thực tế. Tuy nhiên, đề tài vẫn còn tiềm năng phát triển lớn, như tích hợp thêm hệ thống phân quyền chi tiết, mở rộng tìm kiếm trên nhiều định dạng tài liệu hơn, hoặc phát triển phiên bản dành cho thiết bị di động.

TÀI LIỆU THAM KHẢO

1. <https://viblo.asia/p/elasticsearch-zero-to-hero-2-co-che-hoat-dong-cua-elasticsearch-38X4ENMXJN2>
2. <https://minhphong306.wordpress.com/2021/03/24/elasticsearch-co-gi-o-ben-trong-mot-cluster-life-inside-a-cluster/>
3. <https://vuejs.org/>