**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙜🙢🙠🙞

**TIỂU LUẬN MÔN HỌC**

**CƠ SỞ DỮ LIỆU NÂNG CAO**

**ĐỀ TÀI: QUẢN LÝ SINH VIÊN**

Giảng viên hướng dẫn: TS. Văn Thế Thành

Thầy Phan Minh Tiến

Sinh viên thực hiện:

1. 48.01.104.041 – Trịnh Trung Hiển

2. 48.01.104.104 – Nguyễn Khánh Hoài

TP. HỒ CHÍ MINH – 2025

# LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn sự quan tâm và hỗ trợ của giảng viên bộ môn - thầy Văn Thế Thành cùng thầy Phan Minh Tiến. Với sự chỉ dạy tận tình và luôn tạo điều kiện thoải mái, các thầy đã giúp chúng em học hỏi và nghiên cứu hiệu quả trong quá trình học tập bộ môn “Cơ sở dữ liệu nâng cao”. Những kiến thức mà các thầy đã truyền đạt là nền tảng quan trọng, giúp chúng em từng bước nghiên cứu và phát triển dự án.

Dù dự án đã được nghiên cứu và hoàn thiện đáng kể, nhưng do khối lượng kiến thức của các thành viên trong nhóm còn hạn chế, không tránh khỏi việc tồn tại những thiếu sót. Tuy nhiên, chúng em tin rằng với sự đánh giá, nhận xét từ thầy và các bạn, cũng như từ mọi người xung quanh, chúng em sẽ có cơ hội học hỏi thêm và vận dụng để đề tài ngày càng hoàn thiện và hiệu quả hơn.

Chúng em xin cảm ơn các thầy vì sự nhiệt huyết và tâm huyết trong quá trình giảng dạy, đồng thời mong rằng sẽ nhận được sự góp ý và chỉ dẫn quý báu từ thầy và mọi người để chúng em có thể tiếp tục phát triển dự án một cách tốt nhất. Hy vọng rằng những kiến thức và kinh nghiệm thu được sẽ là hành trang quý giá trên con đường học tập và sự nghiệp sau này của chúng em.

# PHÂN CÔNG CÔNG VIỆC

1. **Phân công nhiệm vụ từng thành viên**

|  |  |
| --- | --- |
| Sinh viên | Nhiệm vụ |
| Trịnh Trung Hiển | Xây dựng cơ sở dữ liệu, Xây dựng các metadata, Thiết kế cơ sở dữ liệu hướng đối tượng, Thiết kế dữ liệu multimedia, Phát triển ứng dụng |
| Nguyễn Khánh Hoài | Xây dựng cơ sở dữ liệu, Cơ sở dữ liệu phân tán, Thiết kế dữ liệu multimedia, Phát triển ứng dụng |

1. **Đánh giá quá trình làm việc và kết quả**

Tất cả thành viên tham gia họp nhóm đầy đủ, đúng giờ, tích cực phát biểu ý kiến.

Các thành viên đều hoàn thành phần nhiệm vụ được phân công đúng hạn.

Các nội dung tìm kiếm được có chất lượng tốt.

Các nội dung được chuẩn bị kỹ lưỡng, có ví dụ, hình ảnh minh họa và phần giải thích rõ ràng.

1. **Đánh giá mức độ đóng góp của từng thành viên trong nhóm (theo thang điểm A, B, C, D)**

|  |  |
| --- | --- |
| Sinh viên | Đánh giá mức độ đóng góp |
| Trịnh Trung Hiển | A |
| Nguyễn Khánh Hoài | A |

**MỤC LỤC**

[LỜI CẢM ƠN 2](#_Toc186834741)

[PHÂN CÔNG CÔNG VIỆC 3](#_Toc186834742)

[MỞ ĐẦU 6](#_Toc186834743)

[CHƯƠNG 1. XÂY DỰNG CƠ SỞ DỮ LIỆU 7](#_Toc186834744)

[1.1. Giới thiệu 7](#_Toc186834745)

[1.2. Sơ đồ quan hệ thực thể (ERD) 7](#_Toc186834746)

[1.3. Sơ đồ mức logic 8](#_Toc186834747)

[1.4. Sơ đồ mức vật lý 9](#_Toc186834748)

[CHƯƠNG 2. CƠ SỞ DỮ LIỆU PHÂN TÁN 11](#_Toc186834749)

[2.1. Giới thiệu 11](#_Toc186834750)

[2.2. Phân tán cơ sở dữ liệu 12](#_Toc186834751)

[2.3. Kiến trúc phân tán 13](#_Toc186834752)

[CHƯƠNG 3. XÂY DỰNG CÁC METADATA 14](#_Toc186834753)

[3.1. Giới thiệu 14](#_Toc186834754)

[3.2. Cấu trúc metadata 14](#_Toc186834755)

[CHƯƠNG 4. THIẾT KẾ CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG 16](#_Toc186834756)

[4.1. Giới thiệu 16](#_Toc186834757)

[4.2. Mô hình hóa dữ liệu 17](#_Toc186834758)

[4.2. Triển khai cơ sở dữ liệu hướng đối tượng 19](#_Toc186834759)

[CHƯƠNG 5. THIẾT KẾ DỮ LIỆU MULTIMEDIA 27](#_Toc186834760)

[5.1. Giới thiệu 27](#_Toc186834761)

[5.2. Thiết kế mô hình multimedia trên Cloundinary 27](#_Toc186834762)

[5.2.1. Lưu trữ dữ liệu multimedia 27](#_Toc186834763)

[5.2.2. Tích hợp multimedia với hệ thống 28](#_Toc186834764)

[5.2.3. Mô hình dữ liệu 30](#_Toc186834765)

[CHƯƠNG 6. PHÁT TRIỂN ỨNG DỤNG 32](#_Toc186834766)

[6.1. Thiết kế giao diện 32](#_Toc186834767)

[6.2. Triển khai ứng dụng 34](#_Toc186834768)

[TÀI LIỆU THAM KHẢO 41](#_Toc186834769)

# MỞ ĐẦU

Trong thế giới hiện đại, nơi thông tin và dữ liệu trở thành nguồn tài nguyên quý giá, cơ sở dữ liệu nổi lên như một trong những trụ cột quan trọng nhất của hệ thống công nghệ. Không chỉ đơn thuần là nơi lưu trữ thông tin, cơ sở dữ liệu còn là nhịp tim của mọi hệ thống, giữ cho toàn bộ cấu trúc vận hành một cách nhịp nhàng và bền vững.

Đặc biệt, trong lĩnh vực giáo dục, nơi quản lý thông tin sinh viên ngày càng trở nên phức tạp và đa chiều, nhu cầu về một hệ thống cơ sở dữ liệu mạnh mẽ và hiệu quả càng trở nên cấp thiết. Việc quản lý thông tin không chỉ dừng lại ở những danh sách sinh viên hay bảng điểm, mà còn mở rộng đến việc tổ chức lớp học, lịch thi, hay thậm chí là theo dõi quá trình học tập và phát triển cá nhân của từng sinh viên.

Xuất phát từ nhu cầu đó, chúng em tiến hành xây dựng cơ sở dữ liệu cho một hệ thống quản lý sinh viên – không chỉ như một bài tập học thuật, mà còn như một bước khởi đầu cho sự hình thành của một giải pháp thực tiễn. Chúng em không chỉ hướng tới việc thiết kế một hệ thống đúng về mặt kỹ thuật, mà còn đặt mục tiêu xây dựng một nền tảng linh hoạt, dễ mở rộng, và tối ưu hóa hiệu suất.

Thông qua các bước từ phân tích yêu cầu, thiết kế dữ liệu, đến triển khai ứng dụng, bài tiểu luận này không chỉ mang tính mô tả mà còn phản ánh những tư duy, những quyết định có cơ sở lý luận vững chắc và tầm nhìn dài hạn. Đây không chỉ là một công trình nghiên cứu, mà còn là niềm tin vào giá trị của sự cẩn trọng và sáng tạo trong mỗi bước đi của hành trình xây dựng hệ thống.

# CHƯƠNG 1. XÂY DỰNG CƠ SỞ DỮ LIỆU

## 1.1. Giới thiệu

Trong bối cảnh phát triển nhanh chóng của công nghệ thông tin, việc quản lý dữ liệu trở nên vô cùng quan trọng. Đối với hệ thống quản lý sinh viên, cơ sở dữ liệu không chỉ là nền tảng lưu trữ thông tin mà còn là chìa khóa để đảm bảo sự chính xác, hiệu quả và khả năng mở rộng.

Chương này trình bày quy trình xây dựng cơ sở dữ liệu của hệ thống quản lý sinh viên, bao gồm các bước từ phân tích yêu cầu, thiết kế sơ đồ quan hệ thực thể (ERD), đến chuyển đổi sang sơ đồ mức logic và triển khai sơ đồ mức vật lý. Qua đó, chúng em mong muốn mang lại một giải pháp quản lý toàn diện và bền vững.

## 1.2. Sơ đồ quan hệ thực thể (ERD)

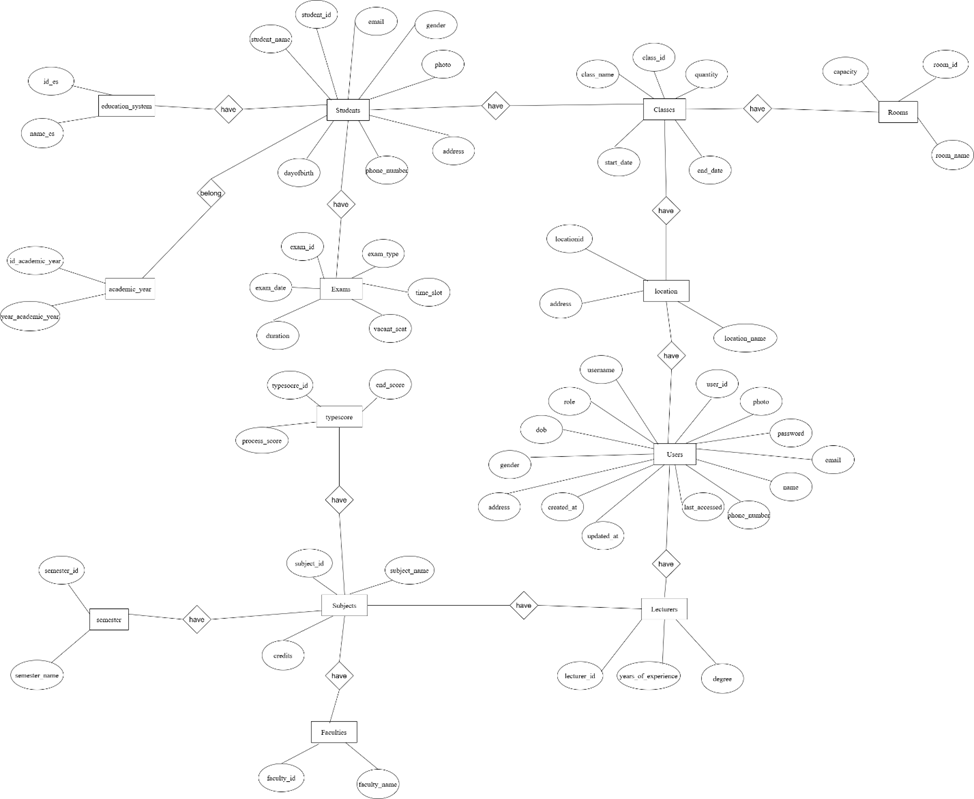
Sơ đồ quan hệ thực thể (ERD) là bước đầu tiên trong thiết kế cơ sở dữ liệu, đóng vai trò như một bản thiết kế chi tiết cho hệ thống. Thông qua việc biểu diễn trực quan các thực thể, thuộc tính và mối quan hệ, ERD giúp chúng ta hiểu rõ cấu trúc và tổ chức của dữ liệu trong hệ thống.

Hệ thống quản lý sinh viên được thiết kế với các thực thể chính, bao gồm:

1. Students: Quản lý thông tin sinh viên với các thuộc tính như mã sinh viên, họ tên, ngày sinh, giới tính, email, địa chỉ, số điện thoại.
2. Education\_System: Lưu trữ thông tin về các hệ thống giáo dục như mã hệ thống và tên hệ thống.
3. Academic\_Year: Cung cấp dữ liệu về các năm học, với thông tin chi tiết về mã năm học và tên năm học.
4. Classes: Đại diện cho các lớp học, với các thông tin như mã lớp, tên lớp, số lượng sinh viên, ngày bắt đầu và ngày kết thúc.
5. Rooms: Lưu trữ thông tin về phòng học, bao gồm mã phòng, tên phòng và sức chứa.
6. Subjects: Quản lý dữ liệu về môn học với mã môn, tên môn, tín chỉ, học kỳ, và khoa.
7. Exams: Tổ chức thông tin các kỳ thi, bao gồm mã kỳ thi, loại kỳ thi, ngày thi, thời gian, và số ghế trống.
8. Lecturers: Lưu trữ thông tin giảng viên với mã giảng viên, bằng cấp, và số năm kinh nghiệm.

Các mối quan hệ chính được xác định như sau:

1. Mỗi sinh viên thuộc về một lớp nhất định.
2. Các lớp học được tổ chức tại các phòng học cụ thể.
3. Giảng viên phụ trách giảng dạy các môn học.
4. Sinh viên tham gia các kỳ thi theo lịch thi được lên kế hoạch trước.



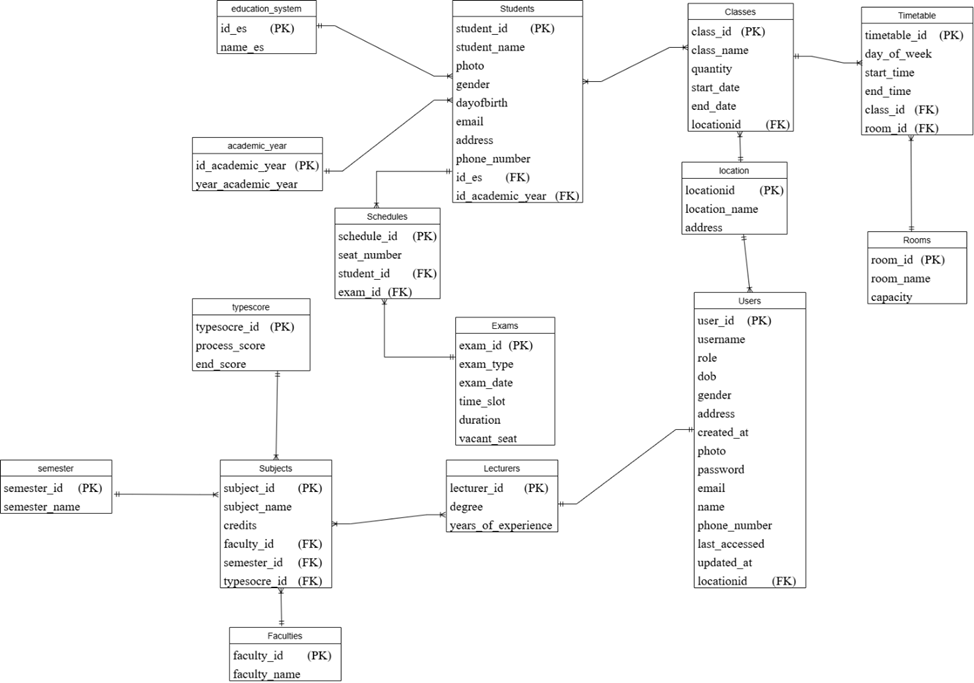
*Hình 1.1. Sơ đồ ERD*

## 1.3. Sơ đồ mức logic

Sau khi thiết kế ERD, chúng em tiến hành chuyển đổi sang sơ đồ mức logic. Ở giai đoạn này, các thực thể được biểu diễn dưới dạng bảng, các thuộc tính trở thành cột trong bảng, và các mối quan hệ được thể hiện thông qua khoá chính (Primary Key) và khoá ngoại (Foreign Key).

Các bảng chính trong hệ thống bao gồm:

1. Students: Chứa các trường student\_id (PK), name, gender, dob, email, address, phone\_number, id\_es (FK), id\_academic\_year (FK).
2. Education\_System: Gồm các trường id\_es (PK), name\_es.
3. Academic\_Year: Chứa các trường id\_academic\_year (PK), year\_academic\_year.
4. Classes: Gồm các trường class\_id (PK), class\_name, quantity, start\_date, end\_date, locationid (FK).
5. Subjects: Bao gồm subject\_id (PK), subject\_name, credits, semester\_id (FK), faculty\_id (FK), typescore\_id (FK).
6. Lecturers: Chứa các trường lecturer\_id (PK), degree, years\_of\_experience.
7. Exams: Gồm các trường exam\_id (PK), exam\_type, exam\_date, time\_slot, vacant\_seat.

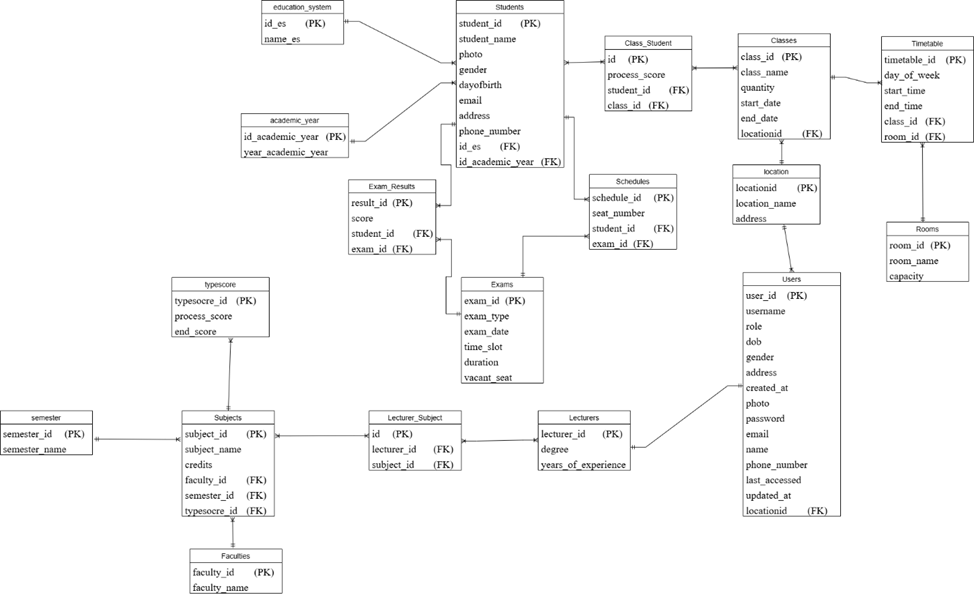


*Hình 1.2. Sơ đồ mức logic*

## 1.4. Sơ đồ mức vật lý

Sơ đồ mức vật lý là bước cuối cùng trong việc triển khai cơ sở dữ liệu. Đây là quá trình định nghĩa chi tiết cấu trúc bảng, kiểu dữ liệu, ràng buộc và quan hệ trong hệ quản trị cơ sở dữ liệu. Dựa trên sơ đồ logic ta có các bảng chính sau:

* Students:
* student\_id INT PRIMARY KEY
* name VARCHAR(100)
* gender CHAR(1)
* dob DATE
* email VARCHAR(100)
* address TEXT
* phone\_number VARCHAR(15)
* id\_es INT FOREIGN KEY REFERENCES Education\_System(id\_es)
* id\_academic\_year INT FOREIGN KEY REFERENCES Academic\_Year(id\_academic\_year)
* Classes:
* class\_id INT PRIMARY KEY
* class\_name VARCHAR(100)
* quantity INT
* start\_date DATE
* end\_date DATE
* locationid INT FOREIGN KEY REFERENCES Location(locationid)
* Subjects:
* subject\_id INT PRIMARY KEY
* subject\_name VARCHAR(100)
* credits INT
* semester\_id INT FOREIGN KEY REFERENCES Semester(semester\_id)
* faculty\_id INT FOREIGN KEY REFERENCES Faculties(faculty\_id)
* typescore\_id INT FOREIGN KEY REFERENCES Typescore(typescore\_id).



*Hình 1.3. Sơ đồ mức vật lý*

# CHƯƠNG 2. CƠ SỞ DỮ LIỆU PHÂN TÁN

## 2.1. Giới thiệu

Cơ sở dữ liệu phân tán là một hệ thống quản lý dữ liệu được thiết kế để lưu trữ trên nhiều máy chủ hoặc địa điểm khác nhau, nhưng hoạt động như một hệ thống thống nhất. Trong mô hình này, cơ sở dữ liệu toàn cục (**Global Database**) được quản lý trên **SQL Server** nhằm đảm bảo tính nhất quán, quản lý tập trung, và cung cấp dữ liệu cốt lõi cho toàn hệ thống. Dữ liệu từ cơ sở toàn cục này sẽ được phân tán ra các node MongoDB để tối ưu hóa việc lưu trữ và truy cập trong môi trường phân tán.

**Đặc điểm chính của hệ thống:**

**Cơ sở dữ liệu toàn cục (Global Database):**

SQL Server đóng vai trò trung tâm trong việc lưu trữ các thông tin quan trọng như danh sách sinh viên, thông tin lớp học, và các dữ liệu quản lý quan trọng khác. Hệ thống đảm bảo:

* Tính đồng bộ cao, dữ liệu luôn chính xác trên mọi node.
* Xử lý hiệu quả các truy vấn phức tạp từ các thành phần khác nhau trong hệ thống.
* Tích hợp dễ dàng với các thành phần như API hoặc các ứng dụng phụ trợ.

**Phân tán dữ liệu (MongoDB):**Dữ liệu từ cơ sở dữ liệu toàn cục được phân phối đến các **node MongoDB** dựa trên khu vực địa lý. Trong trường hợp này:

* **Node MongoDB Hà Nội:** Lưu trữ dữ liệu liên quan đến sinh viên và lớp học ở khu vực miền Bắc.
* **Node MongoDB TP.HCM:** Lưu trữ dữ liệu của sinh viên và lớp học ở khu vực miền Nam.

**Tính đồng bộ:**Hệ thống đảm bảo rằng dữ liệu giữa SQL Server và các node MongoDB luôn được đồng bộ hóa thông qua các giao thức tích hợp hoặc API trung gian. Điều này đảm bảo:

* Khi có cập nhật từ SQL Server, các node MongoDB sẽ nhận được thông tin kịp thời.
* Dữ liệu trên các node luôn đồng nhất với cơ sở dữ liệu trung tâm.

**Khả năng mở rộng:**

Thêm các node MongoDB cho các khu vực khác nếu cần, ví dụ như Đà Nẵng hoặc các tỉnh thành khác.

Mở rộng SQL Server để lưu trữ thêm dữ liệu mà không làm gián đoạn hệ thống hiện tại.

## 2.2. Phân tán cơ sở dữ liệu

**Cơ sở dữ liệu gốc (Global Database):**  
Toàn bộ dữ liệu quan trọng ban đầu được lưu trữ tại **SQL Server** với vai trò trung tâm. Hệ thống này đảm bảo tính nhất quán, xử lý các truy vấn phức tạp và cung cấp một môi trường an toàn cho dữ liệu cốt lõi.

*Hình 2.1. Lưu trữ cơ sở dữ liệu toàn cục (SQL Server)*

*Hình 2.2. Dữ liệu sinh viên trong bảng Students - SQL Server*

**Phân tán qua MongoDB:**  
Để đáp ứng nhu cầu mở rộng và truy xuất nhanh chóng, dữ liệu được phân phối từ SQL Server đến các node MongoDB. MongoDB lưu trữ metadata hoặc các phần dữ liệu cần thiết tại các địa điểm khác nhau, giúp tăng cường hiệu suất truy cập cục bộ và giảm tải cho máy chủ trung tâm.

*Hình 2.3.* *Lưu trữ cơ sở dữ liệu phân tán (MongoDB)*

A screenshot of a computer program

Description automatically generated

*Hình 2.4.* *Phân phối dữ liệu trên MongoDB*

## 2.3. Kiến trúc phân tán

A diagram of a data processing process

Description automatically generated

*Hình 2.5. Sơ đồ kiến trúc phân tán*

# CHƯƠNG 3. XÂY DỰNG CÁC METADATA

## 3.1. Giới thiệu

Metadata đóng vai trò quan trọng trong hệ thống quản lý dữ liệu, giúp cung cấp thông tin bổ sung để hỗ trợ việc tìm kiếm, theo dõi và quản lý hiệu quả. Trong hệ thống này, metadata chủ yếu tập trung vào các **auditable metadata**, được thiết kế để theo dõi lịch sử hoạt động và thay đổi dữ liệu.

Các auditable metadata này bao gồm thông tin như thời điểm tạo, người thực hiện, thời điểm cập nhật, hoặc trạng thái xóa của dữ liệu. Những metadata này không chỉ giúp đảm bảo tính toàn vẹn và minh bạch của dữ liệu mà còn hỗ trợ trong việc kiểm tra và khôi phục thông tin khi cần thiết.

Hệ thống sử dụng **MongoDB** để lưu trữ metadata, tận dụng khả năng lưu trữ dữ liệu phi cấu trúc và tốc độ truy vấn cao. Việc lưu metadata trong MongoDB đảm bảo tính linh hoạt và hỗ trợ tốt cho các chức năng tìm kiếm và quản lý dữ liệu phân tán.

A close up of a word

Description automatically generated

*Hình 3.1. Metadata lưu trên MongoDB*

## 3.2. Cấu trúc metadata



*Hình 3.2. Ví dụ metadata*

# CHƯƠNG 4. THIẾT KẾ CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

## 4.1. Giới thiệu

Cơ sở dữ liệu hướng đối tượng (Object-Oriented Database - OODB) là một hệ quản trị cơ sở dữ liệu cho phép lưu trữ dữ liệu dưới dạng các đối tượng, phù hợp với cách tiếp cận của lập trình hướng đối tượng (Object-Oriented Programming - OOP). Trong mô hình này, dữ liệu không chỉ bao gồm các thuộc tính (attributes) mà còn đi kèm với các phương thức (methods) giúp xử lý dữ liệu trực tiếp. Điểm khác biệt quan trọng của OODB so với cơ sở dữ liệu quan hệ (RDB) chính là việc nó lưu trữ và quản lý dữ liệu dựa trên các đối tượng thay vì bảng.

Với khả năng hỗ trợ các loại dữ liệu phức tạp như đa phương tiện (hình ảnh, video) và các cấu trúc dữ liệu lồng nhau, cơ sở dữ liệu hướng đối tượng cho phép xây dựng và tổ chức dữ liệu theo cách gần gũi hơn với thế giới thực. Điều này đặc biệt hữu ích trong các ứng dụng hiện đại như hệ thống quản lý học sinh, quản lý tài liệu, hay các ứng dụng liên quan đến trí tuệ nhân tạo và xử lý dữ liệu lớn.

**Mục tiêu và lợi ích của việc sử dụng thiết kế hướng đối tượng:**

**Mục tiêu:**

* Mô hình hóa dữ liệu trực quan: Thiết kế hướng đối tượng giúp mô hình hóa dữ liệu dưới dạng các lớp (class), gần gũi với cách con người hiểu và xử lý thông tin.
* Tích hợp với lập trình hướng đối tượng: Dữ liệu được lưu trữ và xử lý trực tiếp trong các ứng dụng lập trình, giảm sự phức tạp khi phải chuyển đổi giữa các môi trường dữ liệu khác nhau.
* Hỗ trợ dữ liệu phức tạp: Thiết kế hướng đối tượng phù hợp với các ứng dụng yêu cầu quản lý dữ liệu đa phương tiện hoặc các cấu trúc dữ liệu có tính phân cấp.

**Lợi ích:**

* Quản lý dễ dàng hơn: Các đối tượng được lưu trữ trong cơ sở dữ liệu hướng đối tượng tương ứng trực tiếp với các đối tượng trong lập trình, giúp giảm thời gian phát triển và quản lý.
* Tự động ánh xạ (Mapping): Khi sử dụng các công cụ như Entity Framework (EF), việc ánh xạ giữa các lớp (class) trong ứng dụng và các đối tượng trong cơ sở dữ liệu được thực hiện tự động, hỗ trợ hiệu quả các thao tác CRUD (Create, Read, Update, Delete).
* Hiệu suất cao: Nhờ lưu trữ trực tiếp các đối tượng, OODB loại bỏ nhu cầu ánh xạ phức tạp giữa bảng và đối tượng, cải thiện hiệu suất trong việc truy xuất và xử lý dữ liệu.

## 4.2. Mô hình hóa dữ liệu

Class Student:

|  |
| --- |
| public partial class Student  {  public string StudentId { get; set; } = null!;  public string? StudentName { get; set; }  public string? Photo { get; set; }  public string? Gender { get; set; }  public DateOnly? Dayofbirth { get; set; }  public string? Email { get; set; }  public string? Address { get; set; }  public string? PhoneNumber { get; set; }  public string? IdEs { get; set; }  public string? IdAcademicYear { get; set; }  public string? FacultyId { get; set; }  public int? LocationId { get; set; }  public string? AuditableId { get; set; }  public virtual ICollection<ClassStudent> ClassStudents { get; set; } = new List<ClassStudent>();  public virtual ICollection<ExamResult> ExamResults { get; set; } = new List<ExamResult>();  public virtual Faculty? Faculty { get; set; }  public virtual AcademicYear? IdAcademicYearNavigation { get; set; }  public virtual EducationSystem? IdEsNavigation { get; set; }  public virtual Location? Location { get; set; }  public virtual ICollection<Schedule> Schedules { get; set; } = new List<Schedule>();  } |

Class Exam:

|  |
| --- |
| public partial class Exam  {  public string ExamId { get; set; } = null!;  public string? SubjectId { get; set; }  public string? ExamType { get; set; }  public DateOnly? ExamDate { get; set; }  public TimeOnly? TimeSlot { get; set; }  public int? Duration { get; set; }  public string? InvigilatorId { get; set; }  public int? VacantSeat { get; set; }  public string? RoomId { get; set; }  public string? AuditableId { get; set; }  public int? LocationId { get; set; }  public virtual ICollection<ExamResult> ExamResults { get; set; } = new List<ExamResult>();  public virtual Lecturer? Invigilator { get; set; }  public virtual Location? Location { get; set; }  public virtual Room? Room { get; set; }  public virtual ICollection<Schedule> Schedules { get; set; } = new List<Schedule>();  public virtual Subject? Subject { get; set; }  } |

## 4.2. Triển khai cơ sở dữ liệu hướng đối tượng

Thêm sinh viên vào cơ sở dữ liệu:

|  |
| --- |
| public async Task<ServiceResponse<bool>> CreateStudentAsync(StudentCU\_DTO studentDto, string made\_by, int location\_id)  {  var response = new ServiceResponse<bool>();  try  {  if (string.IsNullOrEmpty(studentDto.StudentName) || string.IsNullOrEmpty(studentDto.Email) || string.IsNullOrEmpty(studentDto.PhoneNumber))  {  response.Success = false;  response.Message = "Tên sinh viên, email và số điện thoại là các trường bắt buộc.";  return response;  }  if (await \_studentRepository.IsEmailOrPhoneExistsAsync(studentDto.Email, studentDto.PhoneNumber))  {  response.Success = false;  response.Message = "Email hoặc số điện thoại đã tồn tại.";  return response;  }  string imageUrl = null;  var student = \_mapper.Map<Student>(studentDto);  student.LocationId = location\_id;  student.StudentId = await \_studentRepository  .GenerateStudentIdAsync(student.IdAcademicYear, student.IdEs, student.FacultyId, student.LocationId.Value);  var stopwatch = new Stopwatch();  if (studentDto.ImageFile != null && studentDto.ImageFile.Length > 0)  {  stopwatch.Start();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Bắt đầu đẩy ảnh lên Cloudinary.");  imageUrl = await \_cloudinaryHelper.UploadImageAsync(studentDto.ImageFile, student.StudentId);  student.Photo = imageUrl;  stopwatch.Stop();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Ảnh đã được đẩy lên Cloudinary thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  }  student.AuditableId = Guid.NewGuid().ToString();  var mgstudent = \_mapper.Map<MgStudentDTO>(studentDto);  mgstudent.LocationId = location\_id;  mgstudent.Photo = imageUrl;  mgstudent.StudentId = student.StudentId;  var auditable = new Auditable  {  auditable\_id = student.AuditableId,  created\_at = DateTime.UtcNow,  created\_by = made\_by,  status = 1  };  stopwatch.Restart();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Bắt đầu thêm sinh viên vào SQL.");  await \_studentRepository.CreateStudentAsync(student);  stopwatch.Stop();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Sinh viên đã được thêm vào SQL thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  stopwatch.Restart();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Bắt đầu thêm sinh viên vào MongoDB.");  await \_mongoStudentRepository.AddStudentAsync(mgstudent);  stopwatch.Stop();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Sinh viên đã được thêm vào MongoDB thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  stopwatch.Restart();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Bắt đầu thêm metadata vào Meta.");  await \_metadata.AddMetadaAsync(auditable);  stopwatch.Stop();  LogWithColor(ConsoleColor.Cyan, $"[{GetVietnamTime()}] Metadata đã được thêm vào Meta thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  response.Data = true;  response.Message = "Sinh viên đã được tạo thành công.";  }  catch (DbUpdateException ex)  {  var innerException = ex.InnerException?.Message ?? ex.Message;  response.Success = false;  response.Message = $"Đã xảy ra lỗi: {innerException}";  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Lỗi DbUpdateException: {innerException}");  }  catch (Exception ex)  {  response.Success = false;  response.Message = $"Đã xảy ra lỗi: {ex.Message}";  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Lỗi tổng quát: {ex.Message}");  }  return response;  } |

Cập nhật sinh viên trong cơ sở dữ liệu:

|  |
| --- |
| public async Task<ServiceResponse<bool>> UpdateStudentAsync(string made\_by, string id, StudentUpdateDTO studentDto)  {  var response = new ServiceResponse<bool>();  try  {  var existingStudent = await \_studentRepository.GetStudentByIdAsync(id);  if (existingStudent == null)  {  response.Success = false;  response.Message = "Sinh viên không tồn tại.";  return response;  }  if (await \_studentRepository.IsEmailOrPhoneExistsAsync(studentDto.Email, studentDto.PhoneNumber, id))  {  response.Success = false;  response.Message = "Email hoặc số điện thoại đã tồn tại.";  return response;  }  string imageUrl = null;  var stopwatch = new Stopwatch();  if (studentDto.ImageFile != null && studentDto.ImageFile.Length > 0)  {  if (!string.IsNullOrEmpty(existingStudent.Photo))  {  stopwatch.Start();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Bắt đầu xóa ảnh cũ trên Cloudinary.");  bool isDeleted = await \_cloudinaryHelper.DeleteImageAsync(existingStudent.Photo);  stopwatch.Stop();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Xóa ảnh cũ trên Cloudinary thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  if (!isDeleted)  {  response.Success = false;  response.Message = "Không thể xóa ảnh cũ trên Cloudinary.";  return response;  }  }  if (studentDto.ImageFile != null)  {  stopwatch.Restart();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Bắt đầu đẩy ảnh mới lên Cloudinary.");  imageUrl = await \_cloudinaryHelper.UploadImageAsync(studentDto.ImageFile, existingStudent.StudentId);  stopwatch.Stop();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Ảnh mới đã được đẩy lên Cloudinary thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  existingStudent.Photo = imageUrl;  }  else  {  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Không có ảnh mới để upload lên Cloudinary.");  }  }  stopwatch.Restart();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Bắt đầu cập nhật thông tin sinh viên trong SQL.");  \_mapper.Map(studentDto, existingStudent);  await \_studentRepository.UpdateStudentAsync(existingStudent);  stopwatch.Stop();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Cập nhật thông tin sinh viên trong SQL thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  stopwatch.Restart();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Bắt đầu cập nhật thông tin sinh viên trong MongoDB.");  var mgstudent = \_mapper.Map<MgStudentDTO>(existingStudent);  await \_mongoStudentRepository.UpdateStudentAsync(mgstudent);  stopwatch.Stop();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Cập nhật thông tin sinh viên trong MongoDB thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  stopwatch.Restart();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Bắt đầu cập nhật metadata trong hệ thống.");  await \_metadata.UpdateMetadaAsync(made\_by, existingStudent.AuditableId, existingStudent.LocationId.Value, 1);  stopwatch.Stop();  LogWithColor(ConsoleColor.DarkYellow, $"[{GetVietnamTime()}] Metadata đã được cập nhật thành công. Thời gian thực hiện: {stopwatch.ElapsedMilliseconds} ms.");  response.Data = true;  response.Message = "Cập nhật sinh viên thành công.";  }  catch (Exception ex)  {  response.Success = false;  response.Message = $"Đã xảy ra lỗi: {ex.Message}";  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Lỗi tổng quát: {ex.Message}");  }  return response;  } |

Xóa sinh viên ra khỏi SQL:

|  |
| --- |
| public async Task<ServiceResponse<bool>> DeleteStudentAsync(string made\_by, string id)  {  var response = new ServiceResponse<bool>();  try  {  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Bắt đầu lấy thông tin sinh viên với ID: {id}.");  var student = await \_studentRepository.GetStudentByIdAsync(id);  if (student == null)  {  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Sinh viên với ID: {id} không tồn tại.");  response.Success = false;  response.Message = "Sinh viên không tồn tại.";  return response;  }  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Bắt đầu xóa sinh viên với ID: {id}.");  await \_studentRepository.DeleteStudentAsync(id);  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Bắt đầu cập nhật metadata cho sinh viên với ID: {id}.");  await \_metadata.UpdateMetadaAsync(made\_by, student.AuditableId, student.LocationId.Value, 2);  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Xóa sinh viên với ID: {id} thành công.");  response.Data = true;  response.Message = "Xóa sinh viên thành công.";  }  catch (Exception ex)  {  LogWithColor(ConsoleColor.Red, $"[{GetVietnamTime()}] Đã xảy ra lỗi khi xóa sinh viên với ID: {id}. Lỗi: {ex.Message}");  response.Success = false;  response.Message = $"Đã xảy ra lỗi: {ex.Message}";  }  return response;  } |

Lọc sinh viên theo Khoa (Faculty) và Khóa (Academic Year)

|  |
| --- |
| public async Task<ServiceResponse<List<StudentDTO>>> GetStudentByFacultyAndAcademicYear(string facultyId, string academicyear\_id, int location\_id)  {  var response = new ServiceResponse<List<StudentDTO>>();  try  {  var students = await \_studentRepository.GetStudentByFacultyAndAcademicYear(facultyId, academicyear\_id, location\_id);  response.Data = \_mapper.Map<List<StudentDTO>>(students);  response.Message = "Lấy danh sách thành công";  }  catch (Exception ex)  {  response.Success = false;  response.Message = $"Đã xảy ra lỗi: {ex.Message}";  }  return response;  } |

# CHƯƠNG 5. THIẾT KẾ DỮ LIỆU MULTIMEDIA

## 5.1. Giới thiệu

Trong hệ thống quản lý sinh viên được nhóm phát triển, vai trò của admin là trung tâm trong việc quản lý dữ liệu sinh viên, bao gồm việc xử lý ảnh đại diện và các tài nguyên multimedia khác. Admin có quyền thực hiện các thao tác như thêm mới, cập nhật, hoặc xóa ảnh đại diện của sinh viên.

Để tối ưu hóa việc lưu trữ và truy xuất ảnh, nhóm đã sử dụng **Cloudinary**, một nền tảng quản lý multimedia dựa trên đám mây. Cloudinary cho phép lưu trữ, nén và tối ưu hóa ảnh, giúp giảm tải cho hệ thống nội bộ và hỗ trợ tích hợp dễ dàng thông qua API.

## 5.2. Thiết kế mô hình multimedia trên Cloundinary

### 5.2.1. Lưu trữ dữ liệu multimedia

**Ứng dụng hệ thống:**

Admin có thể tải lên hoặc thay đổi ảnh đại diện cho sinh viên thông qua giao diện quản trị. Những ảnh này được lưu trữ trên Cloudinary, nơi cung cấp URL để quản lý và hiển thị.

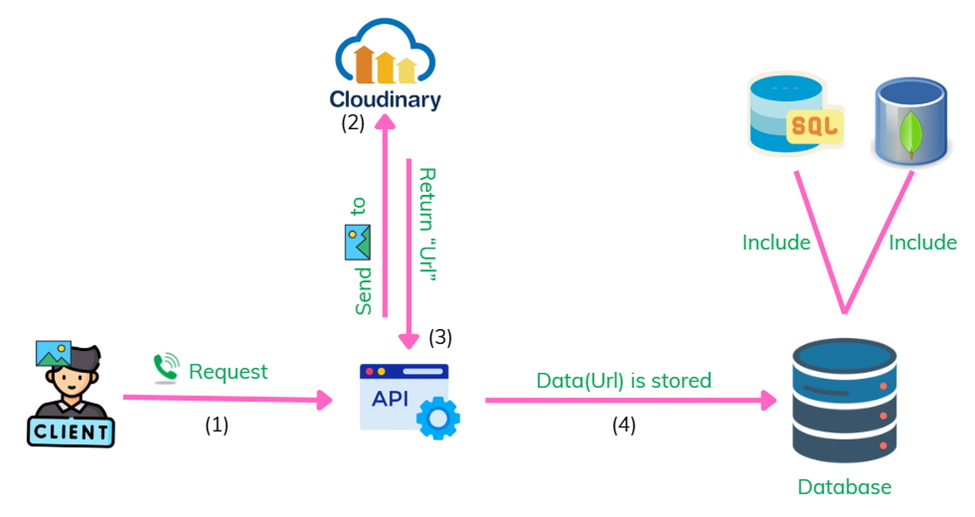
**Quy trình lưu trữ:**

Admin tải ảnh lên từ giao diện quản trị.

Ảnh được gửi đến Cloudinary API để lưu trữ.

Cloudinary trả về một URL công khai, URL này được lưu trong cơ sở dữ liệu (SQL Server và MongoDB).

Hệ thống sử dụng URL này để hiển thị ảnh đại diện trong các giao diện quản trị hoặc báo cáo.



*Hình 5.1. Sơ đồ quy trình lưu trữ dữ liệu multimedia*

### 5.2.2. Tích hợp multimedia với hệ thống

**Quy trình tích hợp:**

Hệ thống sử dụng các tham số được cung cấp bởi Cloudinary để xác thực và cấu hình lưu trữ:

* **CloudName:** Tên tài khoản Cloudinary.
* **API Key và API Secret:** Thông tin bảo mật dùng để truy cập dịch vụ.

Các tham số này được cấu hình thông qua tệp cấu hình secret.json và đọc bằng Iconfiguration.

*Hình 5.2.* *Cấu hình Cloudinary qua IConfiguration*

**Các thao tác chính:**

Tải ảnh lên (Upload):

* Ảnh được gửi từ phía admin qua giao diện quản trị.
* Hệ thống chuyển đổi tệp ảnh thành luồng byte (byte[]) và gửi đến Cloudinary để lưu trữ.
* URL ảnh được Cloudinary trả về và lưu trong cơ sở dữ liệu.

|  |
| --- |
| public async Task<string> UploadImageAsync(byte[] imageBytes, string studentId)  {  try  {  string publicId = NormalizePublicId(studentId);  using (var imageStream = new MemoryStream(imageBytes))  {  var uploadParams = new ImageUploadParams()  {  File = new FileDescription($"student\_photos/{studentId}.jpg", imageStream),  AssetFolder = "student\_photos",  PublicId = publicId  };  var uploadResult = await \_cloudinary.UploadAsync(uploadParams);  if (uploadResult?.StatusCode != System.Net.HttpStatusCode.OK)  {  return null;  }  return uploadResult?.SecureUrl?.ToString();  }  }  catch (Exception ex)  {  // Log lỗi hoặc báo cho người dùng về sự cố  Console.WriteLine($"Error uploading image: {ex.Message}");  return null;  }  } |

Xóa ảnh (Delete):

Khi admin muốn cập nhật hoặc xóa ảnh đại diện, hệ thống sẽ xóa ảnh cũ trên Cloudinary dựa trên PublicId.

|  |
| --- |
| public async Task<bool> DeleteImageAsync(string publicId)  {  try  {  var deleteParams = new DeletionParams(publicId);  var deletionResult = await \_cloudinary.DestroyAsync(deleteParams);  return deletionResult.Result == "ok" || deletionResult.Result == "not found";  }  catch  {  return false;  }  } |

### 5.2.3. Mô hình dữ liệu

Ảnh sinh viên được lưu trữ trên Cloundinary trong folder “student\_photos”:

A screenshot of a computer

Description automatically generated

*Hình 5.3.* *Folder student\_photos*

Url nhận từ Cloundinary sau khi upload hình ảnh lên multimedia này. Sau đó sẽ được lưu vào trong cơ sở dữ liệu:

A screenshot of a computer code

Description automatically generated

Url

*Hình 5.4.* *Url nhận từ Cloundinary*

# CHƯƠNG 6. PHÁT TRIỂN ỨNG DỤNG

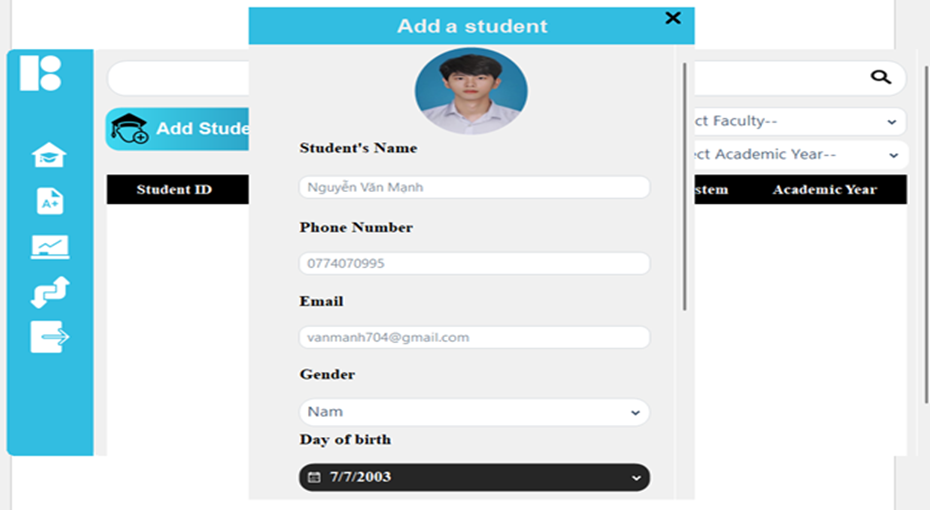
## 6.1. Thiết kế giao diện

Giao diện chính của hệ thống quản lý sinh viên cung cấp các chức năng chính như thêm mới, chỉnh sửa thông tin, và xóa sinh viên. Người dùng có thể thao tác trực tiếp thông qua các nút chức năng và bộ lọc dữ liệu để dễ dàng quản lý danh sách sinh viên.

*Hình 6.1. Giao diện chính*

Giao diện nhập liệu sinh viên:

* Giao diện thêm sinh viên cho phép admin nhập các thông tin cơ bản như mã sinh viên, tên, khoa, và năm học. Sau khi hoàn tất, thông tin này được gửi tới hệ thống để lưu trữ trong cơ sở dữ liệu.



*Hình 6.2. Giao diện thêm sinh viên*

* Giao diện chỉnh sửa sinh viên: Chức năng chỉnh sửa cho phép admin cập nhật thông tin cá nhân của sinh viên, bao gồm ảnh đại diện và các thông tin liên quan. Giao diện được thiết kế đơn giản, thân thiện với người dùng.

Ảnh được lưu trữ trên Cloudinary

*Hình 6.3. Giao diện sửa thông tin sinh viên*

* Giao diện xác nhận khi admin chọn xóa sinh viên: Giao diện xóa sinh viên yêu cầu admin xác nhận trước khi thực hiện để tránh thao tác nhầm. Sau khi xác nhận, dữ liệu của sinh viên sẽ được xóa khỏi hệ thống.

A screenshot of a computer screen

Description automatically generated

*Hình 6.4. Giao diện xóa sinh viên*

Giao diện chờ thông tin phản hồi từ phía Server: Trong quá trình xử lý các thao tác như lưu trữ metadata, upload dữ liệu multimedia lên Cloudinary, hoặc đồng bộ dữ liệu giữa MongoDB và SQL Server, việc thiết kế một giao diện đợi phản hồi là rất cần thiết. Giao diện này giúp người dùng hiểu rằng hệ thống đang hoạt động, giảm cảm giác chờ đợi và nâng cao trải nghiệm người dùng.

A screen shot of a computer

Description automatically generated

*Hình 6.5. Giao diện chờ phản hồi*

Giao diện kết quả phản hồi:

A blue and white rectangle with white text

Description automatically generated

*Hình 6.6. Giao diện kết quả phản hồi*

## 6.2. Triển khai ứng dụng

Bước 1: Thu thập dữ liệu.

Thu thập dữ liệu là bước đầu tiên và quan trọng trong việc xây dựng hệ thống quản lý sinh viên. Quá trình này bao gồm việc nhập thông tin chi tiết của sinh viên và tải lên các tệp đa phương tiện (như ảnh đại diện). Nhờ vào sự tích hợp giữa các thành phần hệ thống, dữ liệu được xử lý và lưu trữ một cách chính xác, đồng thời đảm bảo trải nghiệm mượt mà cho người sử dụng.

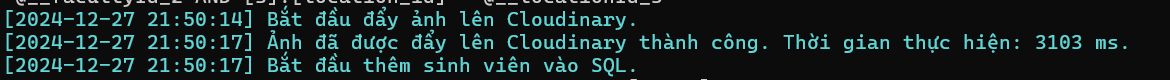
**Nhập thông tin sinh viên từ form WinForm:**  
Người dùng, thường là admin hoặc nhân viên quản lý, sẽ nhập các thông tin chi tiết của sinh viên như mã số sinh viên, họ tên, email, số điện thoại, ngày sinh, địa chỉ, và các thông tin học tập liên quan vào một giao diện trực quan được thiết kế trên nền tảng WinForm. Giao diện thân thiện và dễ sử dụng giúp việc nhập liệu trở nên nhanh chóng và chính xác, đồng thời giảm thiểu sai sót.

**Upload hình ảnh sinh viên:**  
Ứng dụng hỗ trợ chức năng tải lên ảnh đại diện của sinh viên. Hình ảnh được người dùng chọn từ máy tính cá nhân và tự động tải lên nền tảng Cloudinary thông qua API tích hợp.

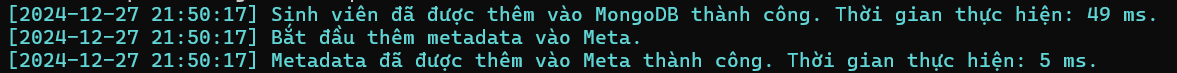
Bước 2: Xử lý dữ liệu.

Backend ASP.NET nhận dữ liệu, xác minh, và lưu trữ vào SQL Server, MongoDB.

Metadata được tự động lưu vào MongoDB để hỗ trợ tìm kiếm.

****

****



*Hình 6.7. Khâu xử lý dữ liệu*

Bước 3: Truy vấn.

Tìm kiếm thông tin sinh viên từ cơ sở dữ liệu (truy vấn kết hợp SQL Server và MongoDB).

**A screenshot of a computer

Description automatically generated**

*Hình 6.8. Giao diện truy vấn thông tin*

Bước 4: Hiển thị.

Hiển thị thông tin của sinh viên và ảnh đại diện.



*Hình 6.9. Giao diện hiển thị thông tin sinh viên*KẾT LUẬN

1. **Đánh giá hiệu suất**

Nhìn chung, hệ thống được thiết kế để tích hợp đa nền tảng, bao gồm cơ sở dữ liệu SQL Server, MongoDB, và dịch vụ lưu trữ đám mây Cloudinary. Điều này mang lại khả năng quản lý dữ liệu toàn diện và mở rộng linh hoạt. Tuy nhiên, việc sử dụng dịch vụ bên thứ ba cũng mang lại một số thách thức về hiệu suất và tính ổn định.

**Ưu điểm:**

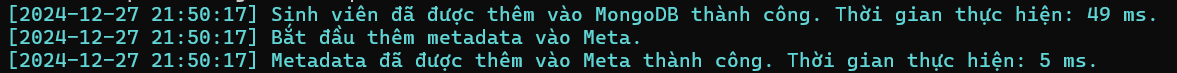
1. **Tích hợp nhiều nền tảng hiệu quả:**

Hệ thống kết hợp giữa lưu trữ cục bộ (SQL Server), lưu trữ phi cấu trúc (MongoDB), và dịch vụ đám mây (Cloudinary), tạo ra một quy trình quản lý dữ liệu mạnh mẽ và linh hoạt.

1. Tối ưu hóa các thao tác xử lý dữ liệu:

Các thao tác lưu trữ dữ liệu vào SQL Server và MongoDB diễn ra nhanh chóng, cho thấy hệ thống đã được tối ưu hóa về mặt hiệu suất.

****



*Hình 7.1.Tối ưu hóa hiệu suất*

1. Khả năng mở rộng:

Với MongoDB lưu trữ metadata và khả năng hỗ trợ xử lý tải lớn, hệ thống dễ dàng mở rộng quy mô khi số lượng dữ liệu và người dùng tăng.

1. Quản lý dữ liệu tốt:

Metadata cho phép theo dõi lịch sử thay đổi và trạng thái dữ liệu, hỗ trợ quá trình kiểm soát và vận hành hiệu quả hơn.

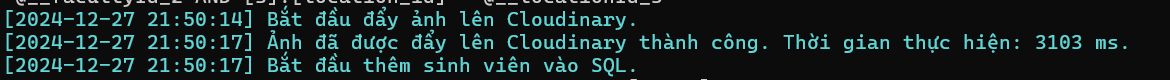
**Nhược điểm:**

1. Phụ thuộc vào bên thứ ba (Cloudinary):

Do sử dụng dịch vụ Cloudinary cho lưu trữ hình ảnh, hệ thống phụ thuộc vào sự ổn định của dịch vụ này. Nếu Cloudinary gặp sự cố (như sập hệ thống), toàn bộ quy trình xử lý ảnh sẽ bị gián đoạn, gây ảnh hưởng nghiêm trọng đến hoạt động.

1. Thời gian xử lý phụ thuộc vào Cloudinary:

Việc tải ảnh lên Cloudinary thường tốn nhiều thời gian hơn so với các thao tác khác, ảnh hưởng đến tốc độ tổng thể của hệ thống. Điều này có thể dẫn đến trải nghiệm không tốt cho người dùng nếu thời gian tải chậm.

****

*Hình 7.2. Thời gian tải ảnh lên Cloudinary*

1. Rủi ro về tính bảo mật và quyền riêng tư:

Lưu trữ dữ liệu quan trọng trên nền tảng bên thứ ba có thể mang lại rủi ro bảo mật nếu hệ thống của bên thứ ba bị xâm phạm hoặc có vấn đề về bảo mật.

1. Không có giải pháp dự phòng:

Nếu Cloudinary ngừng hoạt động hoặc mất kết nối, hệ thống không có phương án dự phòng để tiếp tục xử lý ảnh, dẫn đến tình trạng “ngừng hoạt động toàn bộ”.

1. Phức tạp trong bảo trì và vận hành:

Hệ thống sử dụng nhiều nền tảng khác nhau (SQL Server, MongoDB, Cloudinary, API). Điều này có thể gây khó khăn trong việc bảo trì, khắc phục lỗi, và đồng bộ giữa các phần nếu không có sự quản lý chặt chẽ.

1. **Hướng phát triển**

Để cải thiện hiệu suất, tối ưu hóa trải nghiệm người dùng và tăng cường tính ổn định của hệ thống trong tương lai, chúng tôi đề xuất một số hướng phát triển sau:

1. Xây dựng giải pháp lưu trữ ảnh dự phòng:

Nhằm khắc phục rủi ro khi Cloudinary gặp sự cố, hệ thống cần triển khai thêm một giải pháp lưu trữ ảnh dự phòng. Các lựa chọn bao gồm sử dụng dịch vụ lưu trữ đám mây khác (như AWS S3, Google Cloud Storage) hoặc thiết lập một hệ thống lưu trữ ảnh nội bộ để giảm phụ thuộc hoàn toàn vào bên thứ ba.

1. Cải thiện hiệu suất xử lý ảnh:

Để giảm thời gian tải ảnh, cần tối ưu hóa quy trình tải lên Cloudinary bằng cách nén ảnh trước khi tải hoặc áp dụng công nghệ tải ảnh không đồng bộ (asynchronous upload). Điều này sẽ đảm bảo người dùng không phải chờ đợi lâu khi thao tác với hệ thống.

1. Tăng cường tính bảo mật dữ liệu:

Để hạn chế rủi ro bảo mật từ bên thứ ba, cần mã hóa dữ liệu trước khi lưu trữ trên Cloudinary hoặc MongoDB. Đồng thời, triển khai các cơ chế xác thực bảo mật cao như OAuth2 và tăng cường giám sát các hoạt động truy cập hệ thống.

1. Tích hợp khả năng xử lý ngoại tuyến:

Xây dựng khả năng xử lý ngoại tuyến cho các tác vụ quan trọng, đặc biệt là các thao tác liên quan đến ảnh và dữ liệu. Ví dụ, khi Cloudinary ngừng hoạt động, hệ thống có thể tạm thời lưu trữ ảnh trên máy chủ nội bộ và đồng bộ lại khi dịch vụ hoạt động trở lại.

1. Đơn giản hóa quy trình bảo trì:

Phát triển công cụ giám sát và tự động hóa quy trình bảo trì nhằm giảm sự phức tạp khi hệ thống sử dụng nhiều nền tảng. Các công cụ này có thể bao gồm giám sát thời gian thực, tự động sao lưu dữ liệu, và cảnh báo lỗi khi có sự cố xảy ra.

1. Nghiên cứu áp dụng cơ sở dữ liệu đa mô hình (Multi-model Database):

Để giảm sự phức tạp khi sử dụng đồng thời SQL Server và MongoDB, hệ thống có thể chuyển sang các cơ sở dữ liệu đa mô hình (như ArangoDB hoặc Azure Cosmos DB) để quản lý cả dữ liệu cấu trúc và phi cấu trúc trên một nền tảng duy nhất.

1. Phát triển API hiệu năng cao:

Để cải thiện trải nghiệm người dùng, các API cần được tối ưu hóa hơn nữa nhằm tăng tốc độ xử lý và giảm độ trễ trong giao tiếp giữa hệ thống với các thành phần như cơ sở dữ liệu hoặc dịch vụ Cloudinary.

1. Tăng cường khả năng mở rộng:

Dựa trên nhu cầu sử dụng ngày càng tăng, cần nghiên cứu áp dụng các kiến trúc phân tán như microservices hoặc cloud-native để tăng khả năng mở rộng của hệ thống theo cả chiều ngang và chiều dọc.

1. Tích hợp công nghệ AI để tối ưu hóa dữ liệu:

Sử dụng các thuật toán học máy để phân tích và tối ưu hóa dữ liệu trong hệ thống, chẳng hạn như gợi ý cách tổ chức lớp học, dự đoán nhu cầu tài nguyên, hoặc phát hiện các bất thường trong dữ liệu.

1. Đánh giá và cải tiến liên tục:

Thiết lập quy trình đánh giá định kỳ về hiệu suất, bảo mật, và tính ổn định của hệ thống để nhanh chóng phát hiện và giải quyết các vấn đề mới nảy sinh. Các báo cáo này sẽ là cơ sở để định hình các kế hoạch phát triển tiếp theo.

Với các định hướng trên, chúng tôi kỳ vọng rằng hệ thống không chỉ khắc phục được những hạn chế hiện tại mà còn sẵn sàng đáp ứng những thách thức và yêu cầu mới trong tương lai, mang lại giá trị lâu dài cho người dùng và tổ chức.

# TÀI LIỆU THAM KHẢO

**Tiếng Việt**

[1] Nguyễn Gia Tuấn Anh, Cơ Sở Dữ Liệu Nâng Cao, NXB Đại Học Quốc Gia Tp. Hồ Chí Minh, 2018.

**Internet**

[2] <https://www.youtube.com/watch?v=QHtY3hwy9m8>

[3] <https://www.youtube.com/watch?v=MSbZCXHgkOM>