**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**TRUNG TÂM ĐÀO TẠO SAU ĐẠI HỌC**



**TIỂU LUẬN CƠ SỞ DỮ LIỆU NÂNG CAO**

***KẾT THÚC HỌC PHẦN***

**Đề tài: Ứng dụng cơ sở dữ liệu MongoDB vào bài**

**toán đặt vé xem phim**

**HỌC VIÊN : TRƯƠNG THỊ THU PHƯƠNG**

**NGUYỄN TIẾN DUY**

**LỚP : CAO HỌC KHÓA 14**

**HỌC PHẦN : CƠ SỞ DỮ LIỆU NÂNG CAO**

**GIẢNG VIÊN: TS. PHẠM VĂN HÀ**

**Hà Nội, tháng 8 năm 2024**

# LỜI CẢM ƠN

Ngày nay, hệ thống thông tin đã được tích hợp vào mọi lĩnh vực của cuộc sống xã hội. Tùy thuộc vào quan điểm, có thể phân loại các hệ thống thông tin theo nhiều tiêu chí khác nhau. Xã hội đang trải qua sự phát triển mạnh mẽ, một phần lớn nhờ vào đổi mới công nghệ trong thời đại Công nghệ 4.0.

Trong giai đoạn hiện tại, vai trò của các ứng dụng hệ thống thông tin trong cuộc sống đang trải qua sự phát triển như một cơn bão trên mọi lĩnh vực. Hệ thống thông tin không chỉ giúp chúng ta thực hiện công việc một cách thuận lợi hơn mà còn nâng cao hiệu suất làm việc, tiết kiệm thời gian và nỗ lực. Nó còn hỗ trợ xây dựng nền tảng kiến thức vững chắc, định hình tương lai và đóng góp vào quá trình đổi mới của đất nước trong thời kỳ công nghệ ngày càng phát triển và chiếm ưu thế.

Nhóm chúng em đã áp dụng những kiến thức được học để thực hiện đề tài: ***“Ứng dụng CSDL MongoDB vào bài toán đặt vé xem phim”.*** Trong thời gian làm đề tài đôi lúc không tránh khỏi những thiếu sót, vì vậy chúng em rất mong được nhận những ý kiến đóng góp đến từ Thầy và các bạn để có thể ngày càng hoàn thiện hơn.

Chúng em xin trân trọng cảm hơn sâu sắc đến Thầy TS. Phạm Văn Hà đã tận tình hướng dẫn chúng em hoàn thành được bài tiểu luận này. Chúng em cũng gửi lời cảm ơn chân thành tới các bạn cùng lớp đã có những gợi ý quý báu giúp chúng em hoàn thiện hơn trong bài tiểu luận của mình.

**MỤC LỤC**

[LỜI CẢM ƠN 2](#_Toc175281684)

[DANH MỤC HÌNH ẢNH 3](#_Toc175281685)

[DANH MỤC BẢNG BIỂU 4](#_Toc175281686)

[CHƯƠNG 1: TỔNG QUAN VỀ NOSQL VÀ MONGODB 5](#_Toc175281687)

[1.1. Giới thiệu về NoSQL 5](#_Toc175281688)

[1.1.1. Khái niệm NoSQL 5](#_Toc175281689)

[1.1.2. Các loại cơ sở dữ liệu NoSQL 5](#_Toc175281690)

[1.1.3. Ưu điểm và nhược điểm của NoSQL 7](#_Toc175281691)

[1.1.4. Sự khác biệt giữa cơ sở dữ liệu NoSQL và SQL 8](#_Toc175281692)

[1.2. Giới thiệu về MongoDB 11](#_Toc175281693)

[1.2.1. Lịch sử phát triển của MongoDB 11](#_Toc175281694)

[1.2.2. Những tính năng chính của MongoDB 12](#_Toc175281695)

[1.2.3. Những lợi ích khi MongoDB 15](#_Toc175281696)

[1.3. Cấu trúc dữ liệu trong MongoDB 15](#_Toc175281697)

[1.3.1. Document 15](#_Toc175281698)

[1.3.2. Collection 16](#_Toc175281699)

[1.3.3. Database 16](#_Toc175281700)

[CHƯƠNG 2: TỔNG QUAN VỀ FLUTTER 17](#_Toc175281701)

[2.1. Giới thiệu về Dart và Flutter 17](#_Toc175281702)

[2.2. Một số framework trong lập trình Flutter 18](#_Toc175281703)

[2.2.1. Flutter SDK 18](#_Toc175281704)

[2.2.2. AngularDart 19](#_Toc175281705)

[2.2.3. Tổng quan về mô hình MVC 19](#_Toc175281706)

[CHƯƠNG 3: THIẾT KẾ CƠ SỞ DỮ LIỆU CHO HỆ THỐNG 23](#_Toc175281707)

[3.1. Mô tả bài toán 23](#_Toc175281708)

[3.2. Các quy tắc nghiệp vụ 23](#_Toc175281709)

[3.3. Các Collection 23](#_Toc175281710)

[CHƯƠNG 4: PHÁT TRIỂN PHẦN MỀM QUẢN LÝ VÀ ĐẶT VÉ XEM PHIM 32](#_Toc175281711)

[4.1. Màn hình khởi động phần mềm 32](#_Toc175281712)

[4.2. Màn hình trang chủ 33](#_Toc175281713)

[4.3. Quản lý phim 34](#_Toc175281714)

[4.4. Chức năng quản lý phòng chiếu 35](#_Toc175281715)

[4.5. Chức năng đặt vé 37](#_Toc175281716)

[KẾT LUẬN 38](#_Toc175281717)

[TÀI LIỆU THAM KHẢO 39](#_Toc175281718)

# DANH MỤC HÌNH ẢNH

[Hình 1.1: Kiến trúc của MongoDB 13](#_Toc175281793)

[Hình 1.2: Nền tảng dữ liệu vận hành thông minh 14](#_Toc175281794)

[Hình 2.1: Mô hình MVC 19](#_Toc175281795)

[Hình 3.1: Dữ liệu mẫu của Collection User 24](#_Toc175281796)

[Hình 3.2: Dữ liệu mẫu của Collection Rooms 25](#_Toc175281797)

[Hình 3.3: Dữ liệu mẫu của Collection Showtimes 26](#_Toc175281798)

[Hình 3.4: Dữ liệu mẫu của Collection Showtimes 27](#_Toc175281799)

[Hình 3.5: Dữ liệu mẫu của Collection Showtimes 28](#_Toc175281800)

[Hình 3.6: Biểu đồ usecase tổng quát 29](#_Toc175281801)

[Hình 3.7: Lớp users 30](#_Toc175281802)

[Hình 3.8: Lớp rooms 30](#_Toc175281803)

[Hình 3.9: Lớp Showtimes 31](#_Toc175281804)

[Hình 4.1: Màn hình quản lý đăng nhập 32](#_Toc175281805)

[Hình 4.2: Màn hình quản lý phim 34](#_Toc175281806)

[Hình 4.3: Màn hình thêm phim thành công 35](#_Toc175281807)

[Hình 4.4: Màn hình quản lý phòng 35](#_Toc175281808)

[Hình 4.5: Màn hình thêm mới phòng 36](#_Toc175281809)

[Hình 4.6: Màn hình chức năng đặt vé 37](#_Toc175281810)

# DANH MỤC BẢNG BIỂU

[Bảng 3.1: Collection Users 23](#_Toc175281811)

[Bảng 3.2:Collection Rooms 24](#_Toc175281812)

[Bảng 3.3: Collection Films 25](#_Toc175281813)

[Bảng 3.3: Collection Showtimes 26](#_Toc175281814)

[Bảng 3.3: Collection Tickets 27](#_Toc175281815)

# TỔNG QUAN VỀ NOSQL VÀ MONGODB

## Giới thiệu về NoSQL

### Khái niệm NoSQL

NoSQL còn có nghĩa là Non-Relational (NoRel) – không ràng buộc. Tuy nhiên, thuật ngữ đó ít phổ dụng hơn và ngày nay người ta thường dịch NoSQL thành Not Only SQL – Không chỉ SQL. NoSQL ám chỉ đến những cơ sở dữ liệu không dùng mô hình dữ liệu quan hệ để quản lý dữ liệu trong lĩnh vực phần mềm.

Thuật ngữ NoSQL được giới thiệu lần đầu vào năm 1998 sử dụng làm tên gọi chung cho các lightweight open source relational database (cơ sở dữ liệu quan hệ nguồn mở nhỏ) nhưng không sử dụng SQL cho truy vấn.

Vào năm 2009, Eric Evans, nhân viên của Rackspace giới thiệu lại thuật ngữ NoSQL trong một hội thảo về cơ sở dữ liệu nguồn mở phân tán. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ database mới: distributed (phân tán) + non-relational (không ràng buộc).

### Các loại cơ sở dữ liệu NoSQL

Có một số hệ thống cơ sở dữ liệu NoSQL khác nhau do nhiều điểm khác biệt trong cách quản lý và lưu trữ dữ liệu không lược đồ. Chúng tôi giải thích một số loại phổ biến dưới đây.

* *Cơ sở dữ liệu khóa-giá trị*

Cơ sở dữ liệu khóa-giá trị rất dễ phân chia và cho phép thay đổi quy mô theo chiều ngang tại các cấp độ mà các loại cơ sở dữ liệu NoSQL không thể làm được. Cơ sở dữ liệu khóa-giá trị lưu trữ dữ liệu dưới dạng một tập hợp các cặp khóa-giá trị mà tại đó, khóa đóng vai trò là mã định danh duy nhất. Khóa và giá trị đều có thể là bất cứ thứ gì, từ đối tượng đơn giản đến đối tượng phức hợp phức tạp. Thiết kế dữ liệu lưu trữ khóa–giá trị được dùng cho trò chơi, công nghệ quảng cáo và đặc biệt thích hợp cho IoT.

* *Cơ sở dữ liệu tài liệu*

Cơ sở dữ liệu tài liệu có cùng một định dạng mô hình tài liệu mà các nhà phát triển sử dụng trong mã ứng dụng của họ. Chúng lưu trữ dữ liệu dưới dạng các đối tượng JSON với bản chất linh hoạt, bán cấu trúc và phân cấp. Tính chất linh hoạt, bán cấu trúc và phân cấp của các tài liệu và cơ sở dữ liệu tài liệu cho phép chúng phát triển phù hợp với yêu cầu của ứng dụng. Mô hình cơ sở dữ liệu tài liệu phát huy hiệu quả với danh mục, hồ sơ người dùng và hệ thống quản lý nội dung, nơi từng văn bản là duy nhất và phát triển theo thời gian.

* *Cơ sở dữ liệu đồ thị*

Mục đích của cơ sở dữ liệu đồ thị là giúp mọi người dễ dàng xây dựng và chạy ứng dụng hoạt động với tập dữ liệu kết nối cao. Chúng sử dụng nút để lưu trữ các thực thể dữ liệu và biên để lưu trữ mối quan hệ giữa các thực thể. Biên luôn có một nút bắt đầu, nút kết thúc, kiểu và hướng. Biên có thể mô tả các mối quan hệ cha-con, hành động và quyền sở hữu và các mối quan hệ tương tự. Không giới hạn về số lượng cũng như loại mối quan hệ mà một nút có thể sở hữu. Bạn có thể sử dụng cơ sở dữ liệu đồ thị để xây dựng và chạy ứng dụng hoạt động với bộ dữ liệu kết nối cao. Các trường hợp sử dụng điển hình của cơ sở dữ liệu đồ thị bao gồm mạng xã hội, công cụ đề xuất, phát hiện gian lận và đồ thị kiến thức.

* *Cơ sở dữ liệu nằm trong bộ nhớ*

Trong khi các cơ sở dữ liệu phi quan hệ khác lưu trữ dữ liệu trên đĩa hoặc SSD, các kho dữ liệu trong bộ nhớ được thiết kế để loại bỏ nhu cầu truy cập đĩa. Chúng lý tưởng cho các ứng dụng yêu cầu thời gian phản hồi micro giây hoặc có lưu lượng truy cập tăng đột biến lớn. Bạn có thể sử dụng chúng trong các ứng dụng trò chơi và công nghệ quảng cáo cho các tính năng như bảng xếp hạng, cửa hàng phiên và phân tích thời gian thực.

* *Tìm kiếm cơ sở dữ liệu*

Cơ sở dữ liệu công cụ tìm kiếm là một loại cơ sở dữ liệu phi quan hệ được dành riêng cho việc tìm kiếm nội dung dữ liệu, chẳng hạn như bản ghi đầu ra ứng dụng được sử dụng bởi các nhà phát triển để khắc phục sự cố. Chúng sử dụng các chỉ mục để phân loại các đặc điểm tương tự giữa dữ liệu và tạo điều kiện cho khả năng tìm kiếm. Cơ sở dữ liệu công cụ tìm kiếm được tối ưu hóa để sắp xếp dữ liệu phi cấu trúc như hình ảnh và video.

### Ưu điểm và nhược điểm của NoSQL

Các ứng dụng hiện đại phải đối mặt với một số thách thức có thể được giải quyết bằng cơ sở dữ liệu NoSQL. Ví dụ, các ứng dụng xử lý một khối lượng dữ liệu lớn từ các nguồn khác nhau như phương tiện truyền thông xã hội, cảm biến thông minh và cơ sở dữ liệu của bên thứ ba. Tất cả dữ liệu khác nhau này không phù hợp với mô hình quan hệ. Việc thực thi các cấu trúc dạng bảng có thể dẫn đến dư thừa, sao chép dữ liệu và các vấn đề về hiệu suất trên quy mô lớn.

Cơ sở dữ liệu NoSQL chuyên dành cho các mô hình dữ liệu phi quan hệ và có sơ đồ linh hoạt để xây dựng các ứng dụng hiện đại. Chúng được công nhận rộng rãi vì khả năng dễ phát triển, chức năng cũng như hiệu năng ở quy mô lớn. Lợi ích của cơ sở dữ liệu NoSQL được liệt kê dưới đây.

**Tính linh hoạt:** Cơ sở dữ liệu NoSQL thường cung cấp các sơ đồ linh hoạt giúp công đoạn phát triển nhanh hơn và có khả năng lặp lại cao hơn. Mô hình dữ liệu linh hoạt biến cơ sở dữ liệu NoSQL thành lựa chọn lý tưởng cho dữ liệu không được tổ chức thành cấu trúc hoặc có cấu trúc chưa hoàn chỉnh.

**Khả năng mở rộng:** Cơ sở dữ liệu NoSQL thường được thiết kế để tăng quy mô bằng cách sử dụng các cụm phần cứng được phân phối thay vì tăng quy mô bằng cách bổ sung máy chủ mạnh và tốn kém. Một số nhà cung cấp dịch vụ đám mây xử lý các hoạt động này một cách không công khai dưới dạng dịch vụ được quản lý đầy đủ.

**Hiệu năng cao:** Cơ sở dữ liệu NoSQL được tối ưu hóa cho các mô hình dữ liệu cụ thể và các mẫu truy cập. Chúng cho phép hiệu suất cao hơn so với khi bạn đang cố gắng thực hiện chức năng tương tự với cơ sở dữ liệu quan hệ.

**Chức năng cao:** Cơ sở dữ liệu NoSQL cung cấp các API và kiểu dữ liệu cực kỳ thiết thực được xây dựng riêng cho từng mô hình dữ liệu tương ứng.

### Sự khác biệt giữa cơ sở dữ liệu NoSQL và SQL

|  |  |  |
| --- | --- | --- |
|  | **Cơ sở dữ liệu quan hệ** | **Cơ sở dữ liệu NoSQL** |
| **Khối lượng công việc tối ưu** | Cơ sở dữ liệu quan hệ được thiết kế dành cho các ứng dụng xử lý giao dịch trực tuyến (OLTP) trong giao dịch có độ ổn định cao. Chúng cũng hữu ích để xử lý phân tích trực tuyến (OLAP). | Các cơ sở dữ liệu NoSQL được thiết kế cho các mẫu truy cập dữ liệu, bao gồm các ứng dụng có độ trễ thấp. Cơ sở dữ liệu tìm kiếm NoSQL được thiết kế để phục vụ phân tích dữ liệu có cấu trúc chưa hoàn chỉnh. |
| **Mô hình dữ liệu** | Mô hình quan hệ chuẩn hóa dữ liệu vào bảng được hình thành từ hàng và cột. Sơ đồ quy định rõ ràng bảng, hàng, cột, chỉ mục, mối quan hệ giữa các bảng và các thành tố cơ sở dữ liệu khác. Cơ sở dữ liệu sẽ thực thi tính toàn vẹn tham chiếu trong mối quan hệ giữa các bảng. | Các cơ sở dữ liệu NoSQL cung cấp nhiều mô hình dữ liệu khác nhau như khóa-giá trị, tài liệu, biểu đồ và cột, được tối ưu hóa để đạt hiệu năng và quy mô tối ưu. |
| **Thuộc tính ACID** | Cơ sở dữ liệu quan hệ có các thuộc tính mang tính nguyên tố, nhất quán, tách biệt và bền vững (ACID):   * Tính nguyên tố đòi hỏi giao dịch phải được thực thi đầy đủ hoặc hoàn toàn không thực hiện. * Tính nhất quán đòi hỏi rằng khi dữ liệu phải nhất quán với sơ đồ cơ sở dữ liệu giao dịch được thực hiện. * Tính tách biệt đòi hỏi rằng các giao dịch diễn ra đồng thời phải được thực thi tách biệt với nhau. * Tính bền vững đòi hỏi phải có khả năng phục hồi từ tình trạng hư hỏng hệ thống hoặc mất điện đột ngột về trạng thái đã biết cuối cùng. | Hầu hết cơ sở dữ liệu NoSQL thường phải đánh đổi bằng cách nới lỏng một số thuộc tính ACID này của cơ sở dữ liệu quan hệ để có mô hình dữ liệu linh hoạt hơn có khả năng thay đổi quy mô theo chiều ngang. Việc này biến các cơ sở dữ liệu NoSQL thành lựa chọn tuyệt vời cho các trường hợp sử dụng cần thông lượng cao, độ trễ thấp cần thay đổi quy mô theo chiều ngang vượt qua giới hạn của một phiên bản duy nhất. |
| **Hiệu năng** | Hiệu năng thường phụ thuộc vào hệ thống con của ổ đĩa. Thông thường, việc tối ưu hóa các truy vấn, chỉ mục và cấu trúc bảng bắt buộc phải được thực hiện để đạt mức hiệu năng tối đa. | Hiệu năng thường được xem là chức năng của kích cỡ cụm phần cứng ngầm, độ trễ mạng và ứng dụng đưa ra lệnh gọi. |
| **Điều chỉnh quy mô** | Cơ sở dữ liệu quan hệ thường tăng quy mô bằng cách tăng năng lực điện toán của phần cứng hoặc tăng quy mô bằng cách thêm bản sao của khối lượng công việc chỉ đọc. | Cơ sở dữ liệu NoSQL thường có thể phân mảnh. Điều này là do các mẫu truy cập khóa-giá trị có khả năng tăng quy mô bằng cách sử dụng kiến trúc được phân phối để tăng thông lượng, đem đến hiệu năng ổn định với quy mô gần như không giới hạn. |
| **API** | Yêu cầu lưu trữ và truy xuất dữ liệu được truyền đạt bằng cách sử dụng các truy vấn nhất quán với ngôn ngữ truy vấn có cấu trúc (SQL). Các truy vấn này được phân tích và thực thi bởi cơ sở dữ liệu quan hệ. | API trên cơ sở đối tượng cho phép các nhà phát triển ứng dụng dễ dàng lưu trữ và truy xuất cấu trúc dữ liệu trong bộ nhớ. Khóa phân mảnh tìm kiếm các cặp khóa-giá trị, tập hợp cột hoặc văn bản có cấu trúc chưa hoàn chỉnh có chứa đối tượng và thuộc tính của ứng dụng được xếp theo chuỗi. |

## Giới thiệu về MongoDB

MongoDB là một cơ sở dữ liệu tài liệu nguồn mở và cơ sở dữ liệu hàng đầu của NoSQL. MongoDB được viết bằng C++.

MongoDB là một cơ sở dữ liệu đa nền tảng, định hướng tài liệu cung cấp, hiệu suất cao, tính sẵn sàng cao và khả năng mở rộng dễ dàng. MongoDB hoạt động trên khái niệm về collection và document. MongoDB lưu trữ dữ liệu dưới dạng JSON (trong MongoDB được gọi là dạng BSON vì nó lưu trữ dưới dạng binary từ 1 JSON document).

### Lịch sử phát triển của MongoDB

MongoDB được bắt đầu phát triển vào đầu năm 2007 khi công ty 10gen đang phát triển một nền tảng tương tự dịch vụ Azure của Microsoft. Công ty 10gen là một công ty phần mềm có trụ sở tại New York, nay được đổi tên thành MongoDB Inc. Việc phát triển ban đầu tập trung vào xây dựng PaaS (một nền tảng dịch vụ) nhưng sau đó vào năm 2009, MongoDB đã xuất hiện trên thị trường như một dự án mã nguồn mở máy chủ cơ sở dữ liệu và được duy trì bởi chính tổ chức này.[6]

Tháng 3 năm 2010, MongoDB Inc. đã tung ra sản phẩm sẵn sàng đầu tiên của mình là phiên bản 1.4. Phiên bản ổn định tiếp theo của MongoDB là phiên bản 2.4.9 được phát hành vào ngày 10 tháng 1 năm 2014.

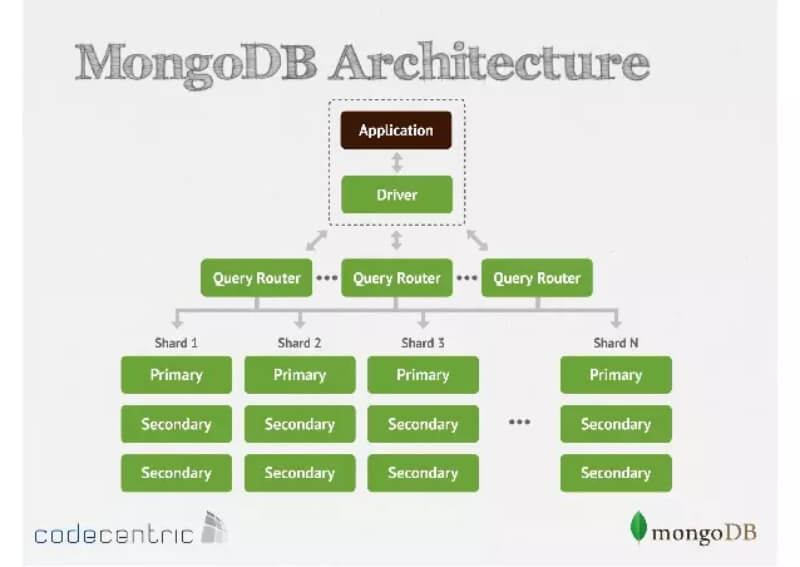
Đầu năm 2015, phiên bản 3.0 được phát hành, cuối năm 2015 phiên 3.2 ra đời đi kèm với công cụ quản trị trên giao diện đồ họa MongoDB Compass.

### Những tính năng chính của MongoDB

MongoDB có những tính năng cơ bản được nhấn mạnh như sau:

* *Truy vấn ad hoc*

Truy vấn ad hoc được xem là tính năng tốt nhất, có khả năng hỗ trợ các trường, truy vấn các phạm vi và tìm kiếm các biểu thức. Từ đó trả về các trường tài liệu cụ thể bao gồm các hàm JavaScript do người dùng xác định hoặc truy vấn được cấu hình trả về mẫu có kích thước nhất định ngẫu nhiên. Ngoài ra, các trường trong MongoDB có thể được dùng để lập các mục chính, chỉ mục phụ.



Hình 1.1: Kiến trúc của MongoDB

* *Nhân rộng*

Đây là tính năng Replica set bao gồm hai hoặc nhiều bản sao dữ liệu, mỗi bản sao có thể giữ vai trò chính hoặc phụ. Trong quá trình nhân rộng, những dữ liệu ghi và đọc thực hiện trên bản sao chính, bản sao thứ cấp sẽ dùng để tích hợp để duy trì dữ liệu từ bản sao chính. Với trường hợp bản sao chính thất bại, Replica set chọn bản sao thứ cấp để làm bản sao chính tiếp theo.

* *Cân bằng tải*

Với cách sử dụng các Sharding, MongoDB sẽ chia theo tỷ lệ chiều ngang để người dùng có thể chọn một Shard key. Nhìn chung, dữ liệu được chia thành các phạm vi sau đó sẽ phân phối đồng đều qua các Shard key.

* *Lưu trữ tệp*

Với tính năng này, MongoDB sử dụng như một hệ thống tệp GridFS để cân bằng tải, sao chép dữ liệu đến nhiều máy tính. GridFS sẽ chia các tệp thành nhiều phần, nhiều đoạn để lưu trữ thành các tài liệu riêng biệt.

* *Tập hợp*

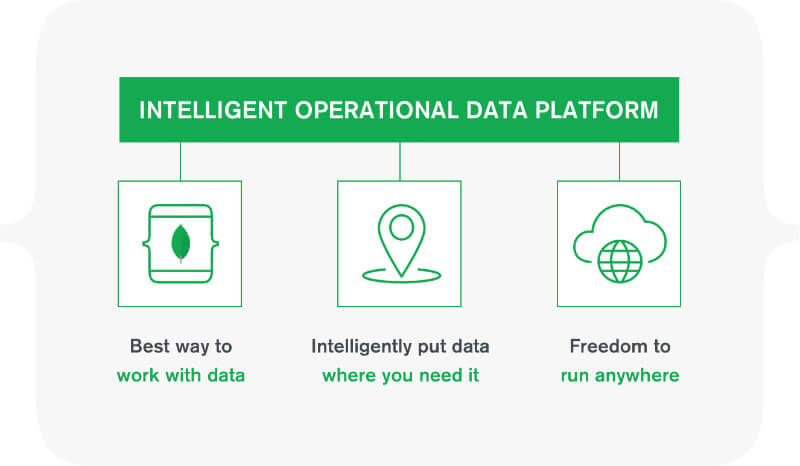
Tính năng này, chương trình cung cấp 3 cách chính để tập hợp là Aggregation Pipeline, Single-purpose Aggregation và Mapreduce. Trong đó, Aggregation Pipeline được ghi nhận có hiệu suất tốt hơn hầu hết những hoạt động tổng hợp.

* *Thực thi Javascript*

JavaScript thường thực thi trong truy vấn, các hàm tổng hợp được và được gửi thủ tục đến cơ sở dữ liệu.

* *Giới hạn kích thước*

MongoDB hỗ trợ collection có kích thước cố định, từ đó làm tăng các hiệu suất các hoạt động khác có liên quan đến dữ liệu. Nếu như dữ liệu vượt qua mức giới hạn, các tài liệu cũ sẽ bị xóa mà không cần thiết nhập bất cứ dòng lệnh nào.



Hình 1.2: Nền tảng dữ liệu vận hành thông minh

### Những lợi ích khi MongoDB

MongoDB mang đến cho người dùng rất nhiều lợi ích khác nhau, cụ thể như sau:

* Tính linh hoạt trong lưu trữ dữ liệu theo nhiều kích thước khác nhau dưới dạng tài liệu JSON nên bạn có thể chèn vào thoải mái bất cứ thông tin nào bạn muốn. Bởi những dữ liệu JSON không có sự ràng buộc hay tuân theo khuôn khổ nhất định, nên bạn sẽ tiết kiệm được thời gian cho sự thỏa mãn về cấu trúc.
* Dễ dàng mở rộng hệ thống với phương pháp thêm node vào cluster.
* Tốc độ truy vấn nhanh hơn so với nhiều hệ quản trị CSDL quan hệ RDBMS bởi dữ liệu được truy vấn lên RAM để lượt truy vấn sau diễn ra nhanh hơn, không cần thiết đọc từ ổ cứng.

Tuy nhiên, MongoDB cũng thể hiện những ưu điểm mà người thao tác cần lưu ý như:

* Dữ liệu trong MongoDB không bị ràng buộc nên cần thao tác cẩn thận để tránh ảnh hưởng đến dữ liệu.
* Chương trình này thường khá tốn bộ nhớ của dữ liệu do được lưu ở dạng key-value, dễ lặp key dẫn đến thừa dữ liệu.

## Cấu trúc dữ liệu trong MongoDB

### Document

Một Document trong MongoDB, có cấu trúc tương tự như kiểu dữ liệu JSON, là một tập hợp các cặp key-value. Các Document có schema động, nghĩa là Document trong cùng một Collection không cần thiết phải có cùng một tập hợp các trường hoặc cấu trúc giống nhau, và các trường chung trong Document của một Collection có thể giữ các kiểu dữ liệu khác nhau.

### Collection

Collection là một nhóm các Document trong MongoDB. Nó tương đương như một bảng trong RDBMS. Do đó, một Collection tồn tại bên trong một cơ sở dữ liệu duy nhất. Các Collection không có ràng buộc Relationship như các hệ quản trị cơ sở dữ liệu khác nên việc truy xuất rất nhanh, chính vì thế mỗi collection có thể chứa nhiều thể loại khác nhau không giống như table trong hệ quản trị mysql là các field cố định. Các Document bên trong một Collection có thể có nhiều trường khác nhau. Đặc biệt, tất cả các Document trong một Collection là tương tự nhau hoặc với cùng mục đích liên quan.

### Database

Database là một nơi chứa vật lý cho các Collection. Mỗi Database lấy tập hợp các file riêng của nó trên hệ thống file. Mỗi MongoDB Server có thể có nhiều cơ sở dữ liệu.

# TỔNG QUAN VỀ FLUTTER

## Giới thiệu về Dart và Flutter

* Dart là một ngôn ngữ lập trình được phát triển bởi Google vào năm 2011, đặc biệt được sử dụng để phát triển ứng dụng di động và web. Dart được thiết kế để cải thiện các vấn đề của JavaScript và cung cấp nhiều tính năng hiện đại hơn. Dart hỗ trợ kiểu tĩnh và kiểu động, cú pháp đơn giản và dễ đọc, hỗ trợ lập trình hướng đối tượng, hàm bậc cao, và bộ thu gom rác tự động.
* Dart cũng hỗ trợ viết code đa nền tảng, cho phép phát triển ứng dụng di động cho cả Android và iOS, cũng như ứng dụng web. Flutter, một framework phát triển ứng dụng di động đa nền tảng, cũng được phát triển bởi Google và sử dụng Dart làm ngôn ngữ lập trình chính. Flutter hỗ trợ phát triển ứng dụng di động với tốc độ và hiệu suất cao, và được đánh giá là một trong những framework phát triển ứng dụng di động tốt nhất hiện nay.
* Ưu điểm:

- Tính năng hiện đại: Dart được thiết kế để cải thiện các vấn đề của JavaScript và cung cấp nhiều tính năng hiện đại hơn, bao gồm hỗ trợ kiểu tĩnh và kiểu động, hàm bậc cao, và bộ thu gom rác tự động.

- Đơn giản và dễ đọc: Cú pháp của Dart đơn giản và dễ đọc, giúp cho các nhà phát triển dễ dàng hiểu và sử dụng ngôn ngữ này.

- Lập trình hướng đối tượng: Dart hỗ trợ lập trình hướng đối tượng, cho phép phát triển ứng dụng với cấu trúc logic rõ ràng và dễ bảo trì.

- Viết code đa nền tảng: Dart hỗ trợ viết code đa nền tảng, cho phép phát triển ứng dụng di động cho cả Android và iOS, cũng như ứng dụng web.

- Flutter: Flutter, một framework phát triển ứng dụng di động đa nền tảng của Google, được sử dụng Dart làm ngôn ngữ lập trình chính. Flutter hỗ trợ phát triển ứng dụng di động với tốc độ và hiệu suất cao, và được đánh giá là một trong những framework phát triển ứng dụng di động tốt nhất hiện nay.

Với những ưu điểm trên, Dart đang trở thành một ngôn ngữ lập trình ngày càng phổ biến và được nhiều nhà phát triển tin tưởng sử dụng cho các dự án phát triển ứng dụng di động và web.

## Một số framework trong lập trình Flutter

### Flutter SDK

Flutter SDK là một SDK hỗ trợ phát triển ứng dụng di động đa nền tảng sử dụng Flutter, cung cấp các tính năng như widget, animation, routing, và hỗ trợ viết ứng dụng đa nền tảng cho cả Android và iOS.

### AngularDart

AngularDart là một framework phát triển ứng dụng web, sử dụng Dart làm ngôn ngữ lập trình chính. AngularDart cung cấp các tính năng như mô hình MVC, dependency injection, routing, và có thể tích hợp với các thư viện khác để phát triển ứng dụng web phong phú.

### Tổng quan về mô hình MVC

Mô hình MVC (Model-View-Controller) là một mô hình thiết kế phần mềm được sử dụng để phát triển các ứng dụng có giao diện người dùng. Mô hình này bao gồm ba thành phần chính: Model, View và Controller.



Hình 2.1: Mô hình MVC

* *Model*

Model là thành phần chịu trách nhiệm xử lý dữ liệu. Nó đại diện cho các đối tượng dữ liệu và quy trình xử lý dữ liệu. Model không biết gì về giao diện người dùng, nó chỉ tập trung vào việc xử lý dữ liệu.

* *View*

View là thành phần chịu trách nhiệm hiển thị dữ liệu và tương tác với người dùng. View đại diện cho giao diện người dùng, bao gồm các thành phần như các nút bấm, trường nhập liệu và các thành phần khác. View không biết gì về dữ liệu và quy trình xử lý, nó chỉ tập trung vào việc hiển thị và tương tác với người dùng.

* *Controller*

Controller là thành phần chịu trách nhiệm điều khiển quá trình tương tác giữa Model và View. Controller đóng vai trò là trung gian giữa Model và View, nó nhận lệnh từ View, xử lý dữ liệu từ Model và cập nhật lại View để hiển thị kết quả cho người dùng. Controller là thành phần duy nhất có thể tương tác với cả Model và View.

Mô hình MVC giúp tách biệt logic xử lý dữ liệu và giao diện người dùng, giúp cho việc phát triển và bảo trì ứng dụng trở nên dễ dàng hơn. Nó cũng giúp cho việc thay đổi giao diện người dùng hoặc cơ sở dữ liệu không ảnh hưởng đến logic xử lý dữ liệu và ngược lại.

* *Một ứng dụng thực tế có thể được thiết kế theo mô hình MVC*

- Model: Chứa các đối tượng dữ liệu như thông tin người dùng, thông tin sản phẩm, danh sách đơn hàng, v.v. Nó cũng chứa các phương thức để truy xuất và xử lý dữ liệu.

- View: Chứa các màn hình giao diện người dùng để hiển thị thông tin và tương tác với người dùng, ví dụ như màn hình đăng nhập, màn hình danh sách sản phẩm, màn hình đặt hàng, v.v.

- Controller: Chứa các phương thức để điều khiển quá trình tương tác giữa Model và View, ví dụ như phương thức để xử lý đăng nhập, phương thức để lấy danh sách sản phẩm, phương thức để đặt hàng, v.v.

Với việc sử dụng mô hình MVC, việc phát triển và bảo trì ứng dụng sẽ trở nên dễ dàng hơn và có thể được thực hiện bởi các nhóm phát triển khác nhau độc lập với nhau.

***\* Ưu điểm***

+ Thực hiện Unit testing bây giờ sẽ rất dễ dàng, vì bạn thực sự không phụ thuộc vào view

+ MVC sẽ tạo sự tương tác hiệu quả giữa designer và developer

+ Tăng khả năng sử dụng lại các thành phần hay việc thay đổi giao diện chương trình mà không cần phải viết lại code quá nhiều

+ Phát triển ứng dụng nhanh, đơn giản, dễ nâng cấp, bảo trì…

***\* Nhược điểm***

+ Khả năng duy trì khi view có thể gán cả biến và biểu thức, các logic không liên quan sẽ tăng dần theo thời gian, ảnh hưởng đến việc thêm code vào XML

+ Đối với dự án nhỏ việc áp dụng mô hình MVVM gây cồng kềnh, tốn thời gian trong quá trình phát triển. Tốn thời gian trung chuyển dữ liệu của các thành phần

+ Đối với dự án lớn hơn, nó gây khó khăn và mất thời gian để thiết kế các ViewModel

+ Việc liên kết dữ liệu cho tất cả các thành phần gây khó khăn trong việc gỡ lỗi khi cơ sở dữ liệu phức tạp

# THIẾT KẾ CƠ SỞ DỮ LIỆU CHO HỆ THỐNG

## Mô tả bài toán

Bài toán đặt vé xem phim là một bài toán phổ biến trong lĩnh vực phát triển phần mềm, đặc biệt là trong các ứng dụng quản lý rạp chiếu phim hoặc các hệ thống dịch vụ giải trí.

Mục đích của bài toán đặt vé xem phim là tạo ra một hệ thống giúp người dùng dễ dàng tìm kiếm, đặt vé, và chọn chỗ ngồi cho các suất chiếu một cách tiện lợi và nhanh chóng. Đồng thời, hệ thống này cũng giúp rạp chiếu phim quản lý hiệu quả lịch chiếu, tình trạng chỗ ngồi, và doanh thu, đồng thời cung cấp các công cụ phân tích dữ liệu để cải thiện dịch vụ và tăng cường trải nghiệm khách hàng. Việc cung cấp một nền tảng trực tuyến không chỉ mở rộng khả năng tiếp cận khách hàng mà còn nâng cao hiệu quả hoạt động và tối ưu hóa quy trình kinh doanh của rạp chiếu phim.

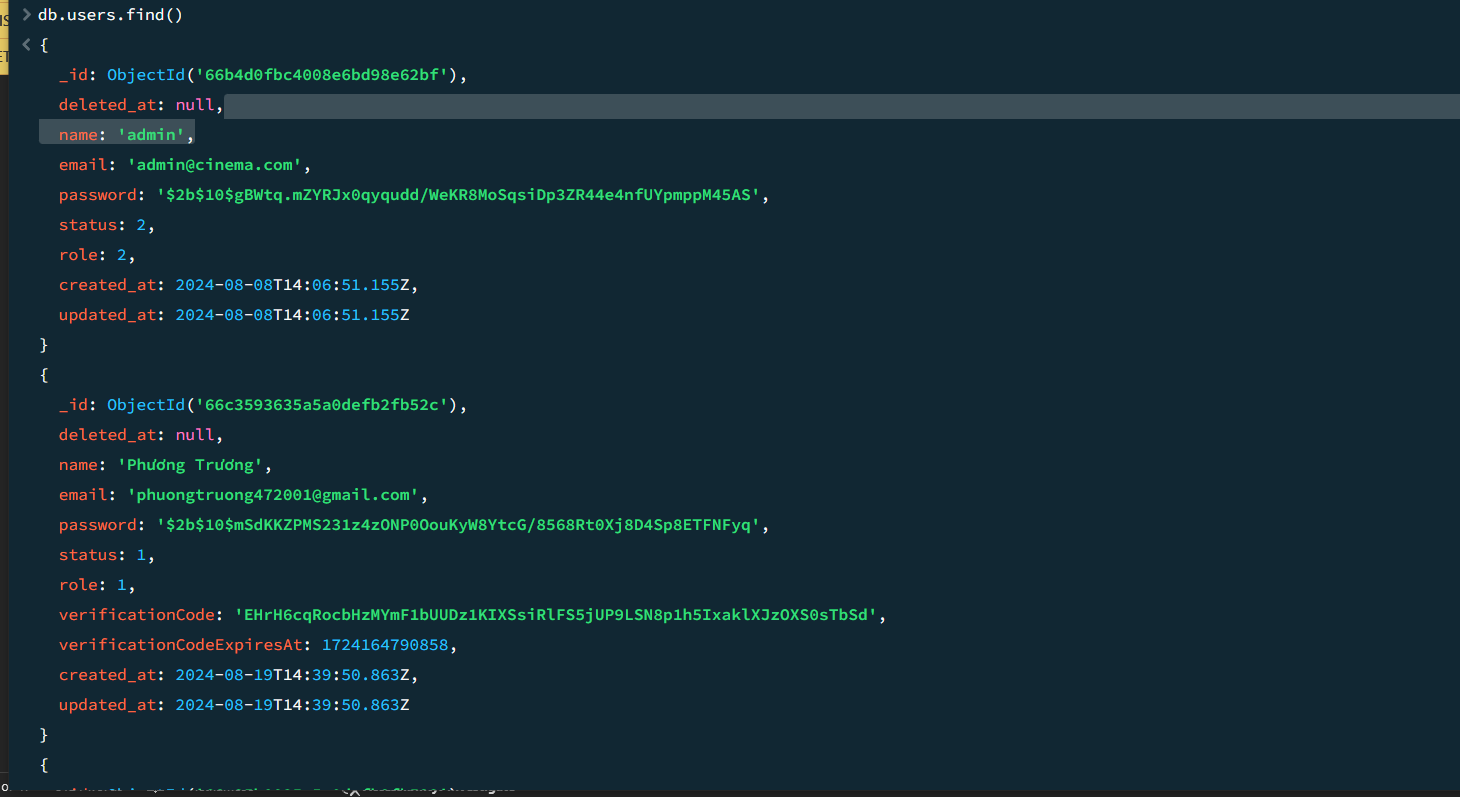
## Các quy tắc nghiệp vụ

* Một rạp chiếu phim bao gồm nhiều phòng chiếu
* Mỗi người có thể đặt nhiều vé xem phim
* Có nhiều khung giờ chiếu

## Các Collection

Bảng 3.1: Collection Users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Collection Users | | | | |
| STT | Tên | Kiểu dữ liệu | Mô tả | Tùy chọn |
| 1 | \_id | string | Mã người dùng | Bắt buộc |
| 2 | name | String | Tên người dùng | Bắt buộc |
| 3 | email | String | Email | Bắt buộc |
| 4 | password | String | Mật khẩu | Bắt buộc |
| 5 | Status | Int | Kích hoạt |  |
| 6 | Role | Int | Quyền admin/ user |  |

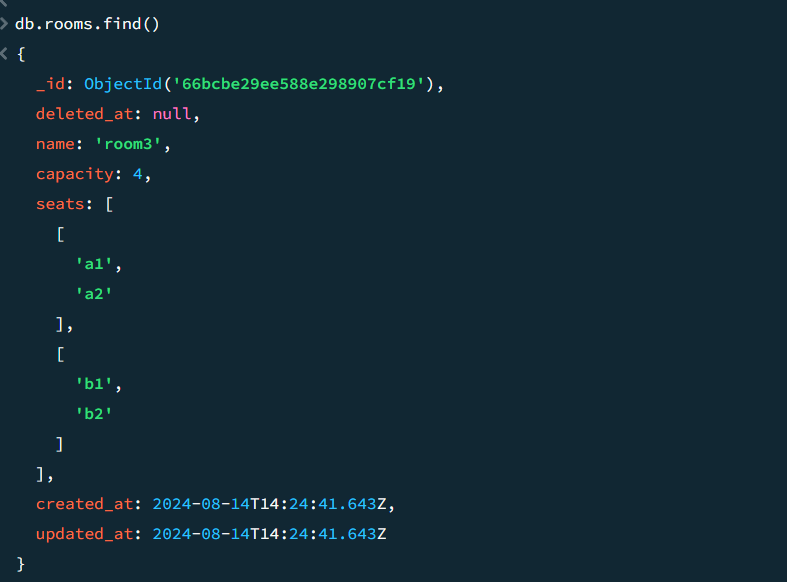


Hình 3.1: Dữ liệu mẫu của Collection User

Bảng 3.2:Collection Rooms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Collection PhongHoc | | | | |
| STT | Tên | Kiểu dữ liệu | Mô tả | Tùy chọn |
| 1 | \_id | string | Mã room | Bắt buộc |
| 2 | name | String | Tên phòng | Bắt buộc |
| 3 | capacity | int | Số lượng ghế | Bắt buộc |
| 4 | seats | List<String,String> | Vị trí ngồi | Bắt buộc |

Hình 3.2: Dữ liệu mẫu của Collection Rooms



Bảng 3.3: Collection Films

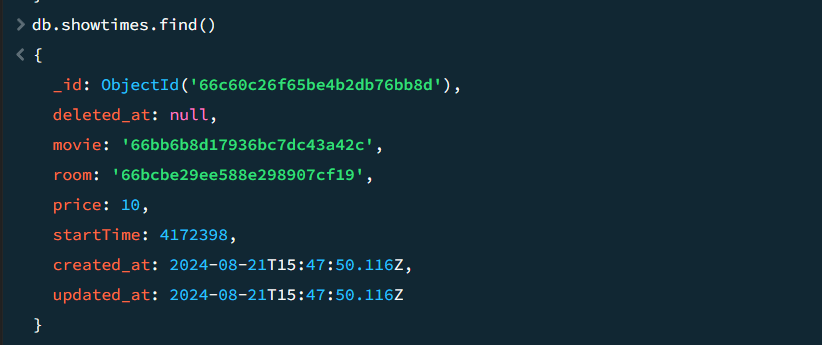
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Collection LichSu | | | | |
| STT | Tên | Kiểu dữ liệu | Mô tả | Tùy chọn |
| 1 | \_id | Int | Mã Films | Bắt buộc |
| 2 | name | String | Tên phim | Bắt buộc |
| 3 | duration | int | Thời lượng | Bắt buộc |
| 4 | description | string | Mô tả | Bắt buộc |
| 5 | thumbnail | string | Link ảnh |  |



Hình 3.3: Dữ liệu mẫu của Collection Showtimes

Bảng 3.3: Collection Showtimes

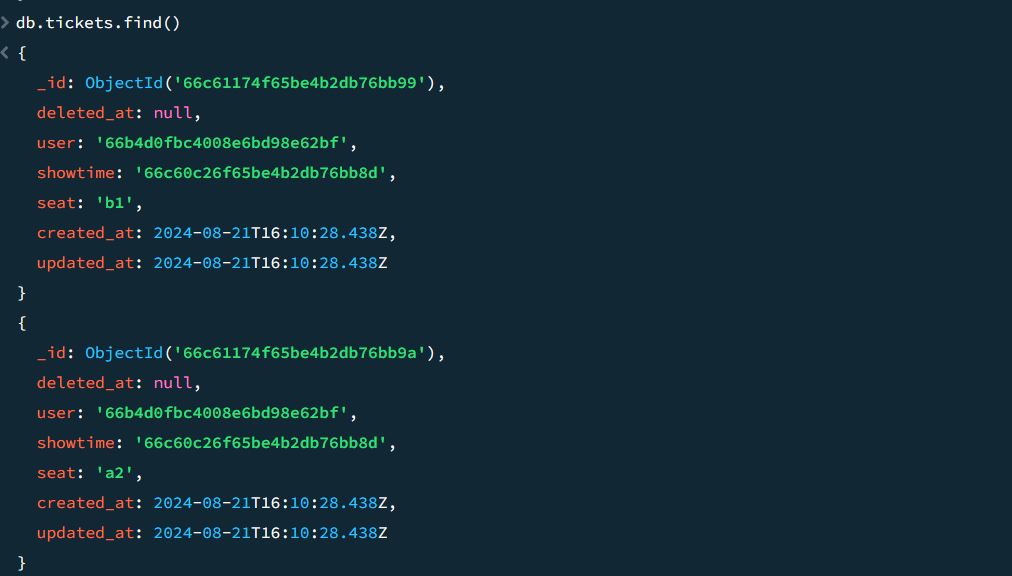
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Collection LichSu | | | | |
| STT | Tên | Kiểu dữ liệu | Mô tả | Tùy chọn |
| 1 | \_id | Int | Mã Showtimes | Bắt buộc |
| 2 | movie | String | ID phim | Bắt buộc |
| 3 | room | Datetime | ID phòng | Bắt buộc |
| 4 | price | double | Giá | Bắt buộc |



Hình 3.4: Dữ liệu mẫu của Collection Showtimes

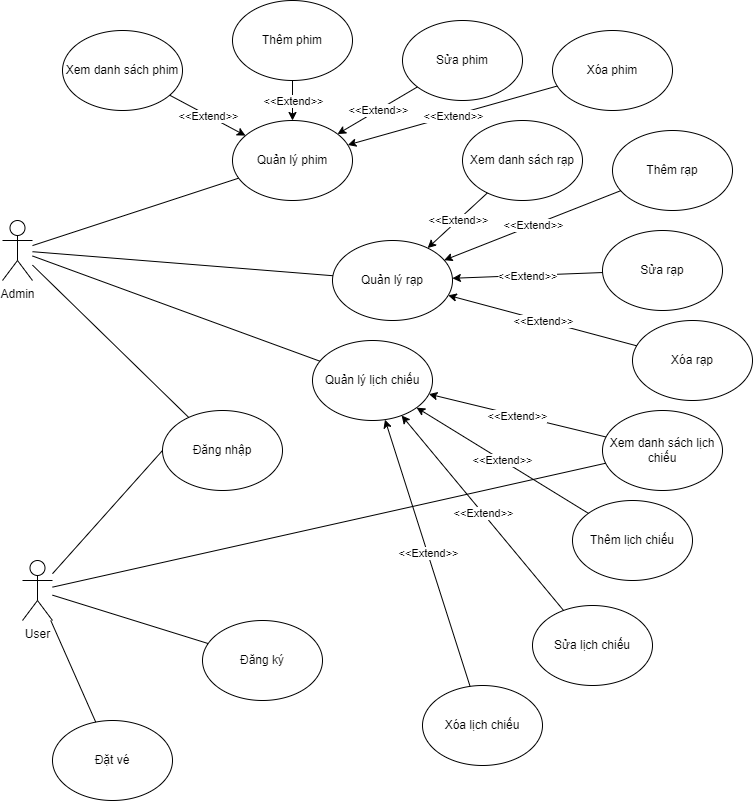
Bảng 3.3: Collection Tickets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Collection LichSu | | | | |
| STT | Tên | Kiểu dữ liệu | Mô tả | Tùy chọn |
| 1 | \_id | Int | Mã Showtimes | Bắt buộc |
| 2 | user | String | ID người đặt | Bắt buộc |
| 3 | showtime | String | ID showtime | Bắt buộc |
| 4 | seat | String | Chỗ ngồi | Bắt buộc |



Hình 3.5: Dữ liệu mẫu của Collection Showtimes

* Biểu đồ usecase tổng quát

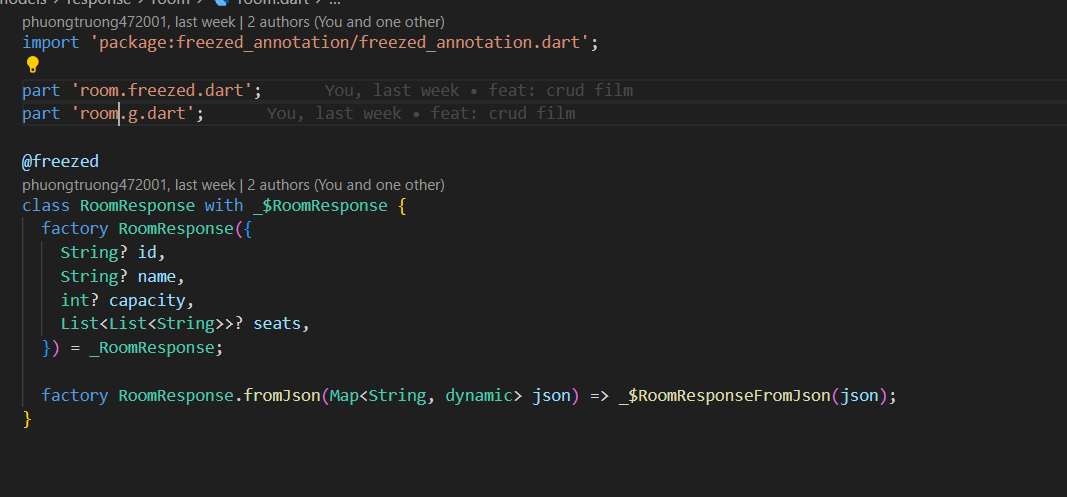


Hình 3.6: Biểu đồ usecase tổng quát

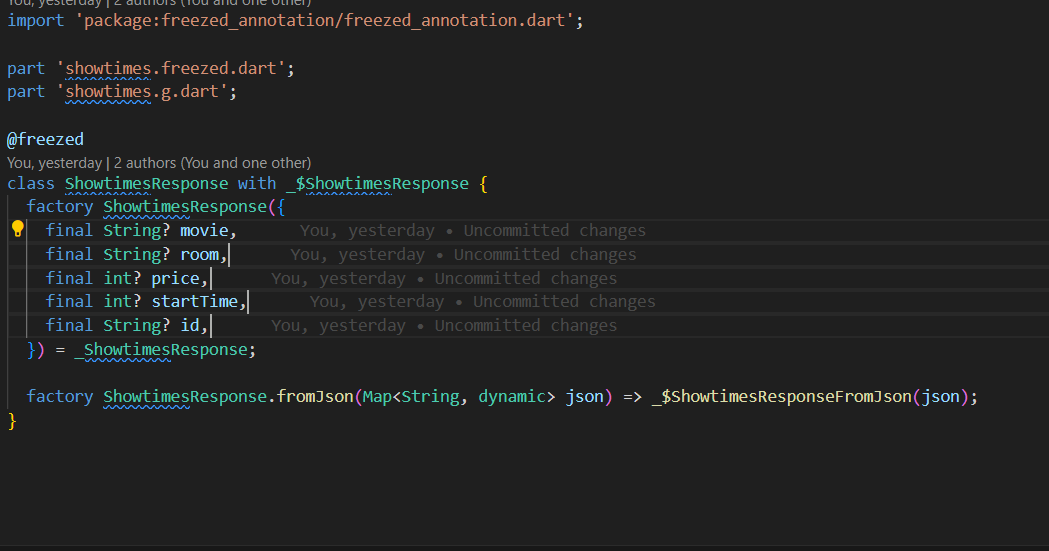
* Xây dựng chương trình với Flutter



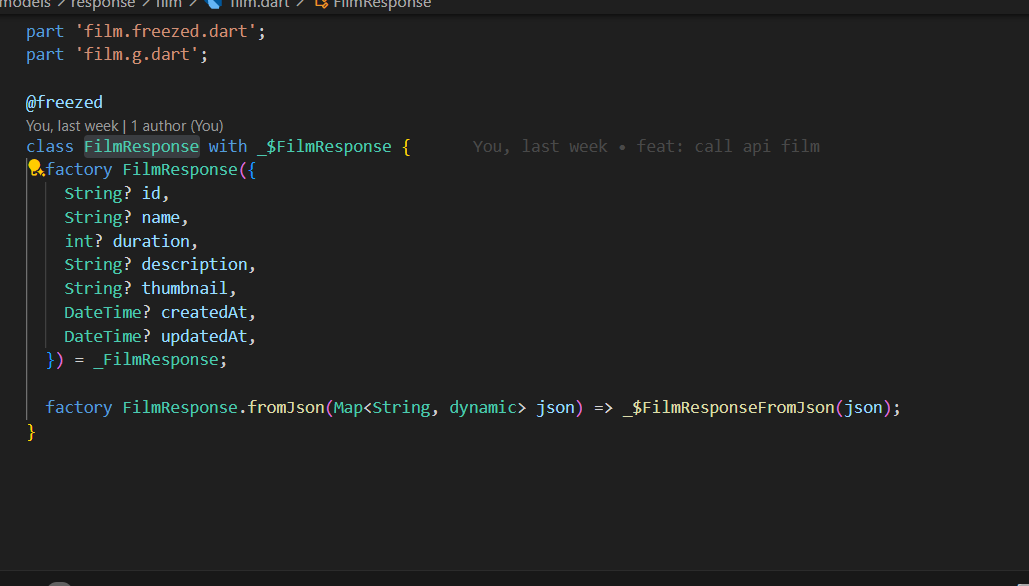
Hình 3.7: Lớp users



Hình 3.8: Lớp rooms



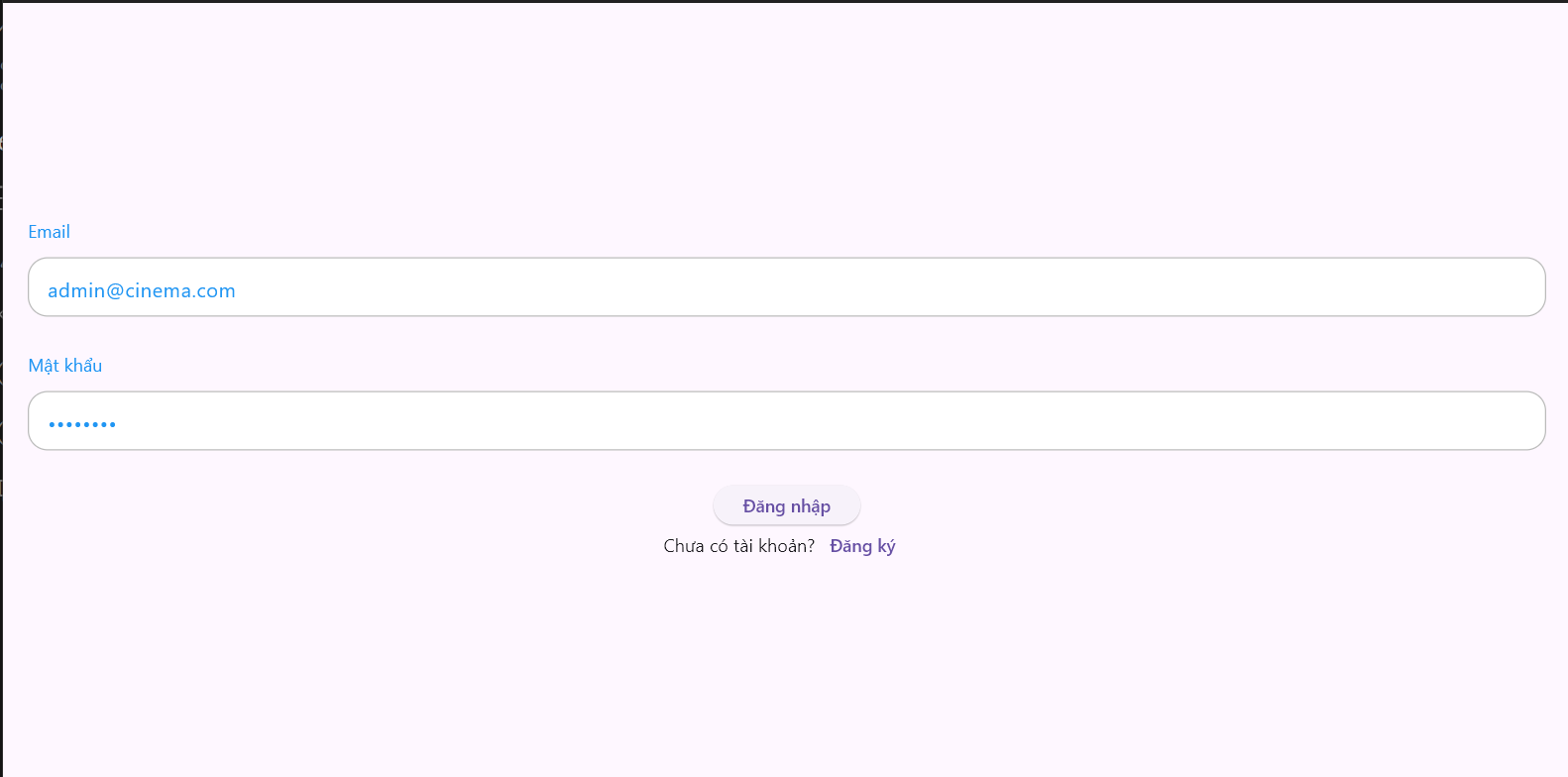
Hình 3.9: Lớp Showtimes



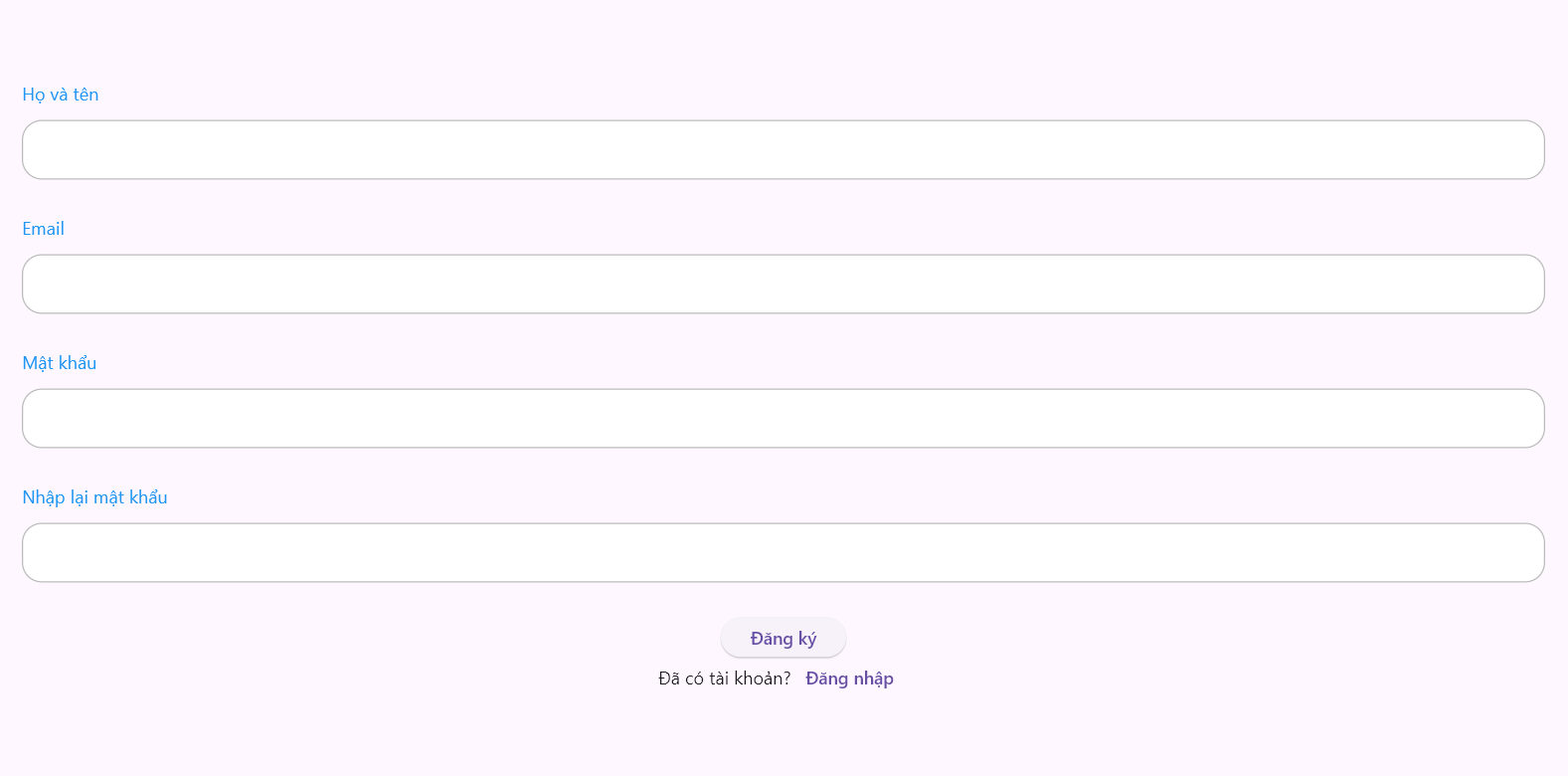
Hình 3.11: Lớp Films

# PHÁT TRIỂN PHẦN MỀM QUẢN LÝ VÀ ĐẶT VÉ XEM PHIM

## Màn hình khởi động phần mềm



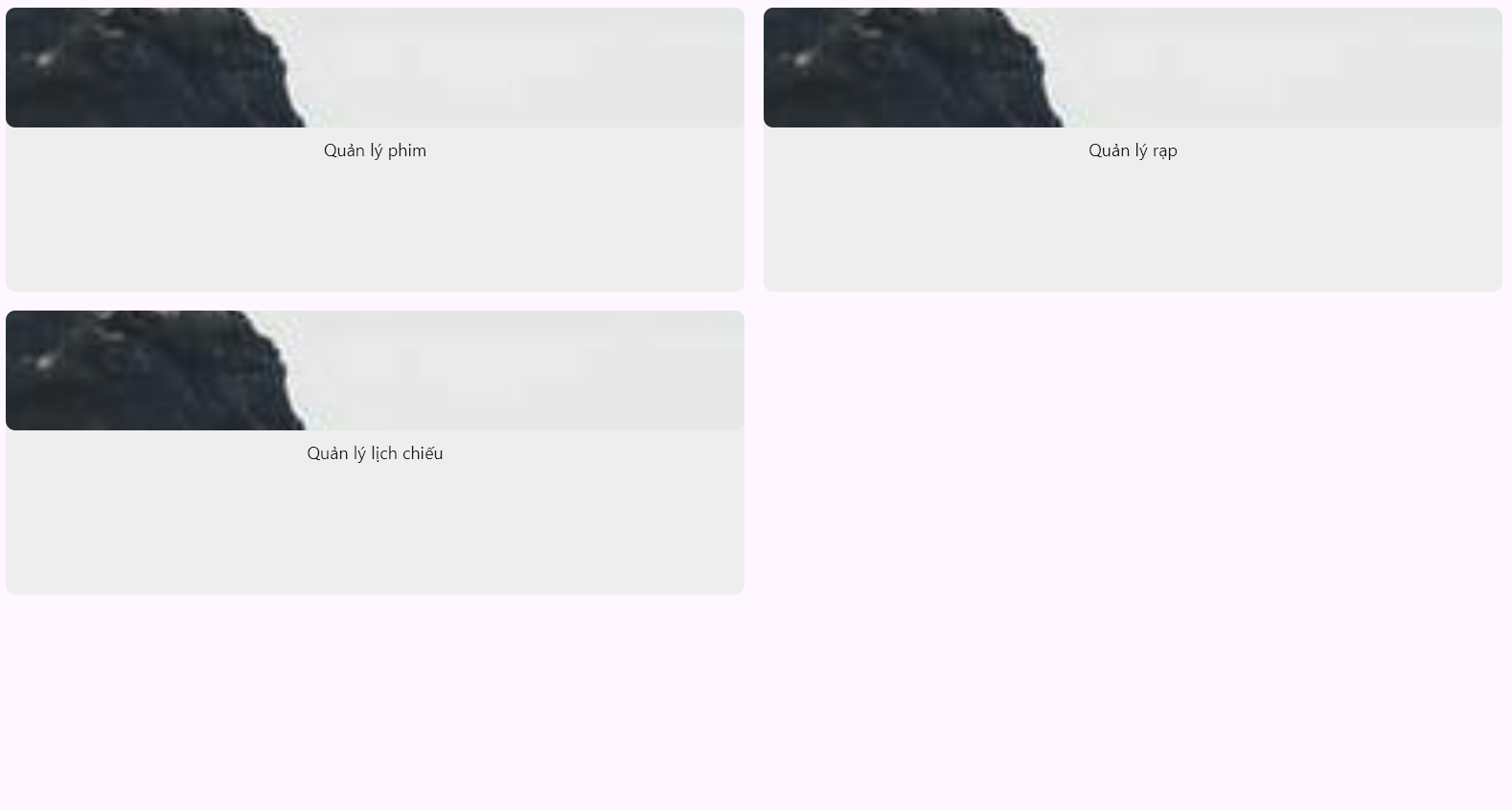
Hình 4.1: Màn hình quản lý đăng nhập



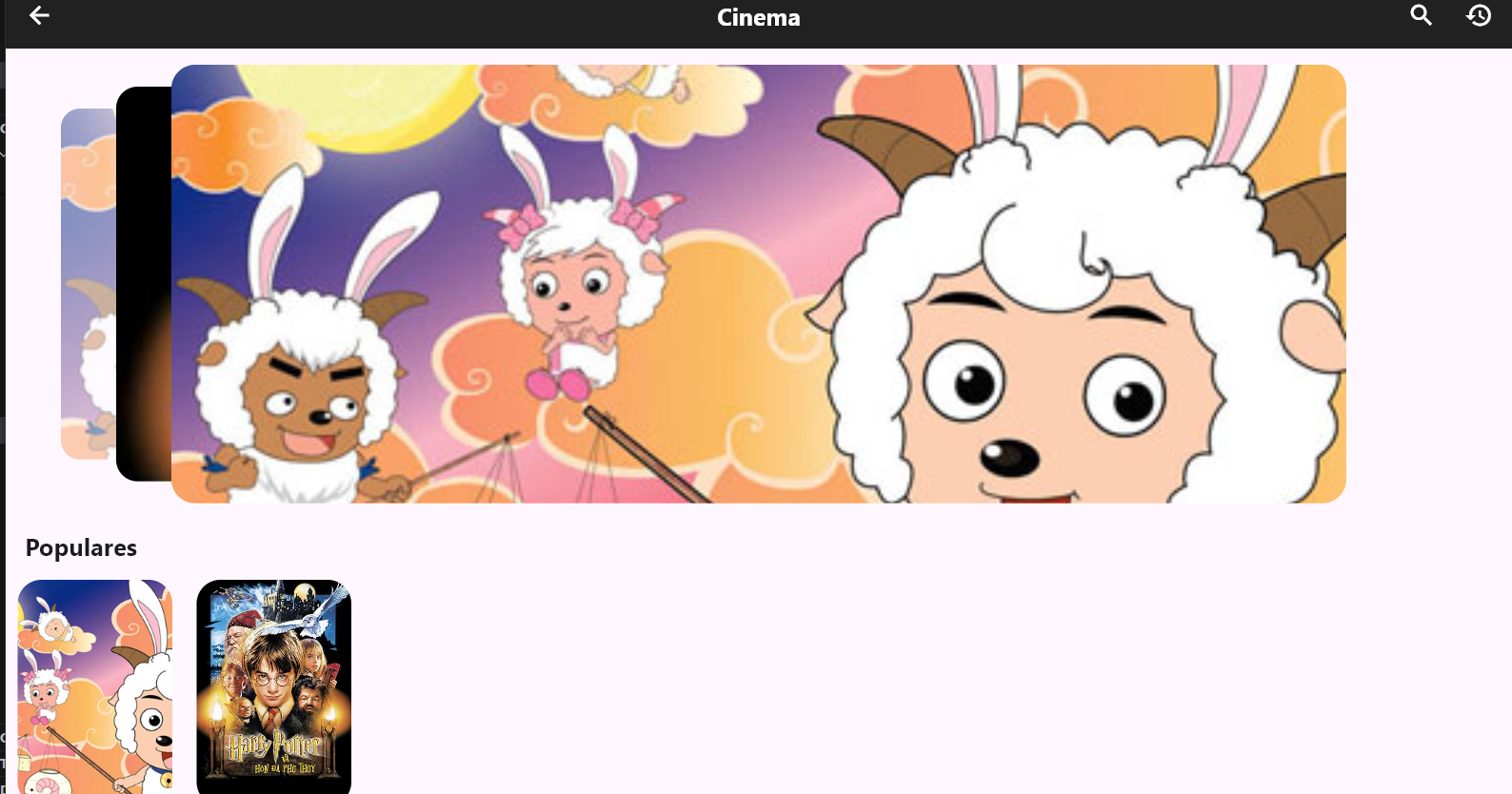
Hình 4.2: Màn hình quản lý đăng ký

## Màn hình trang chủ

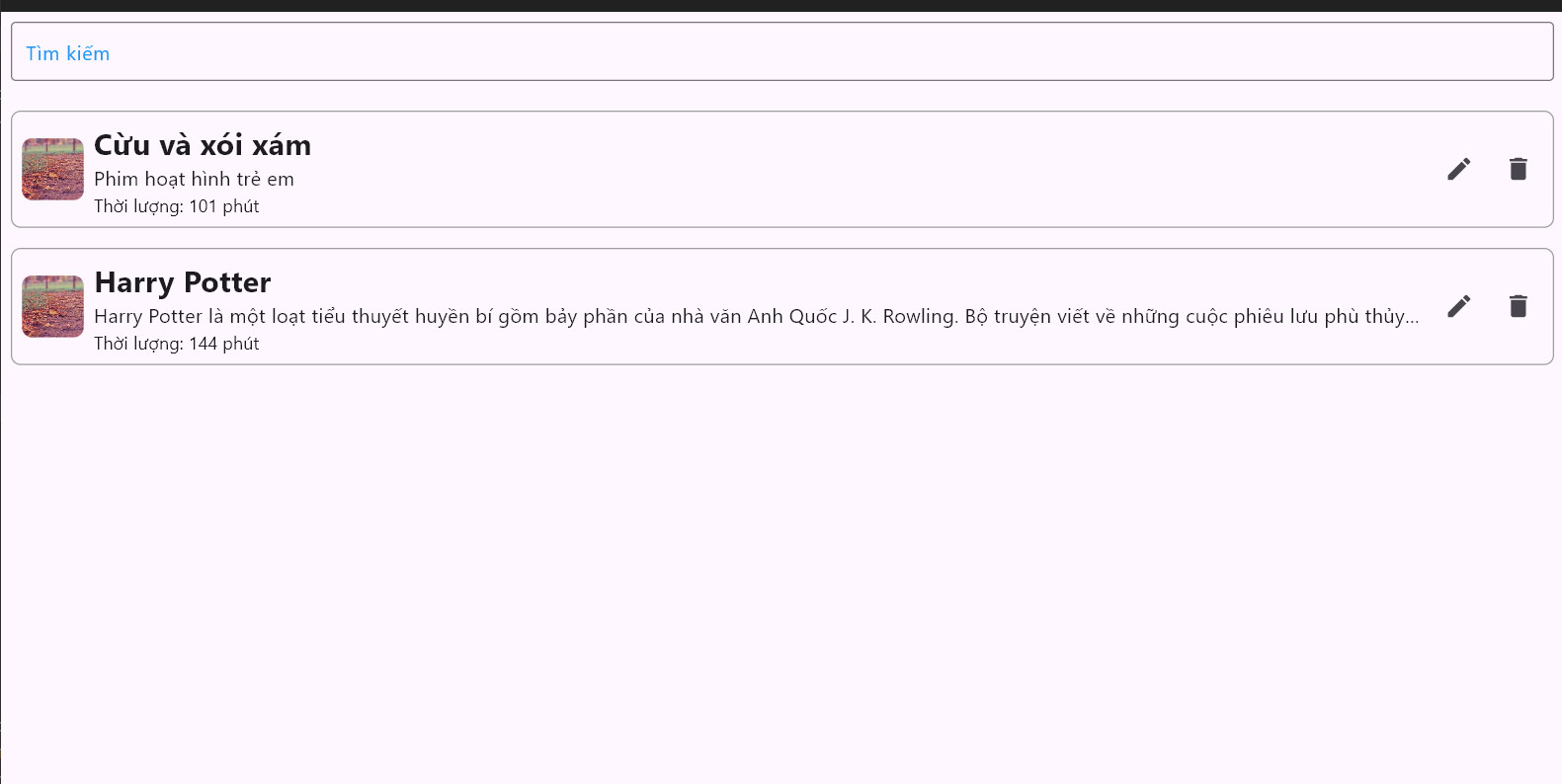
* ADMIN



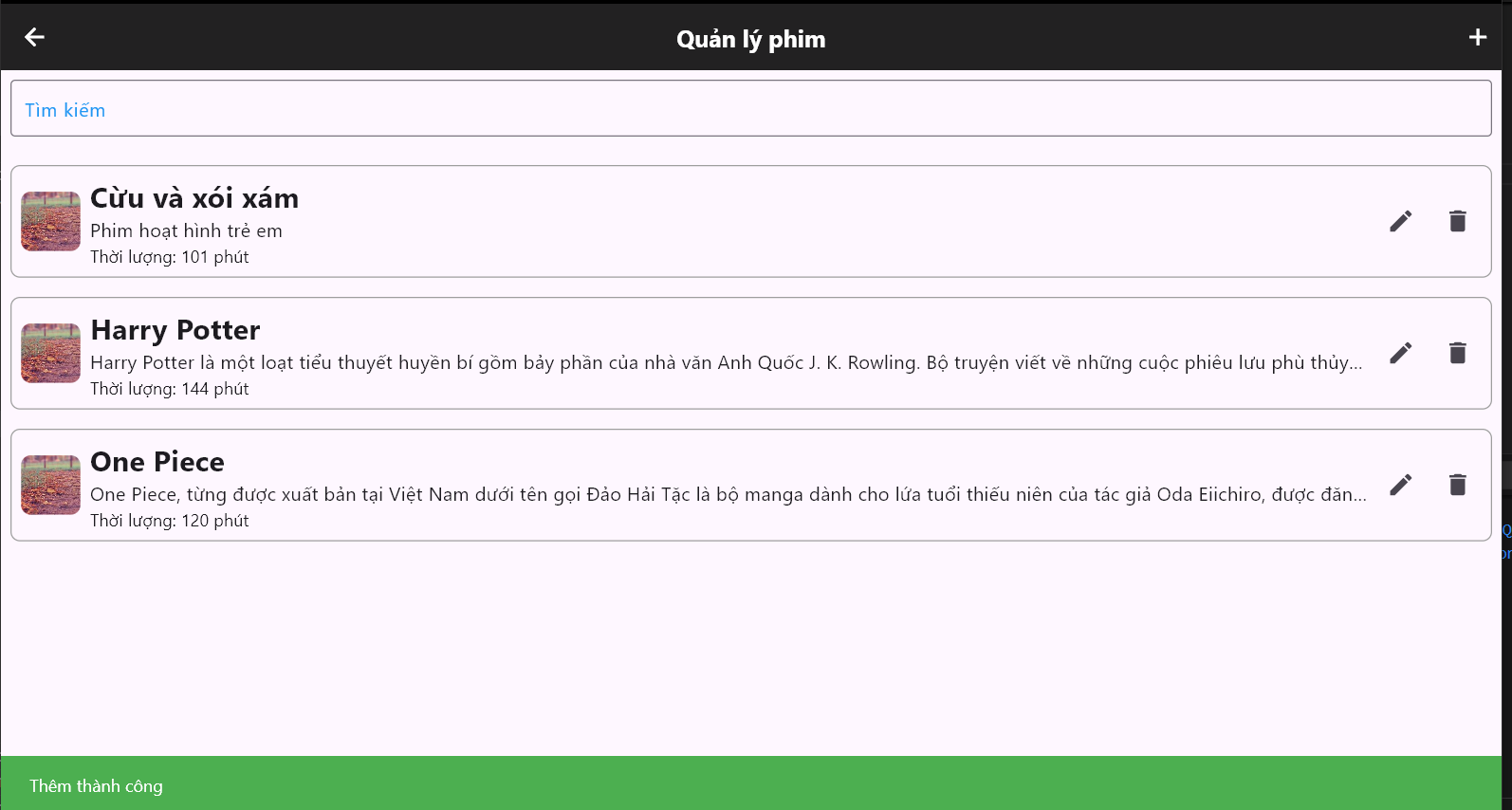
* USER



## Quản lý phim

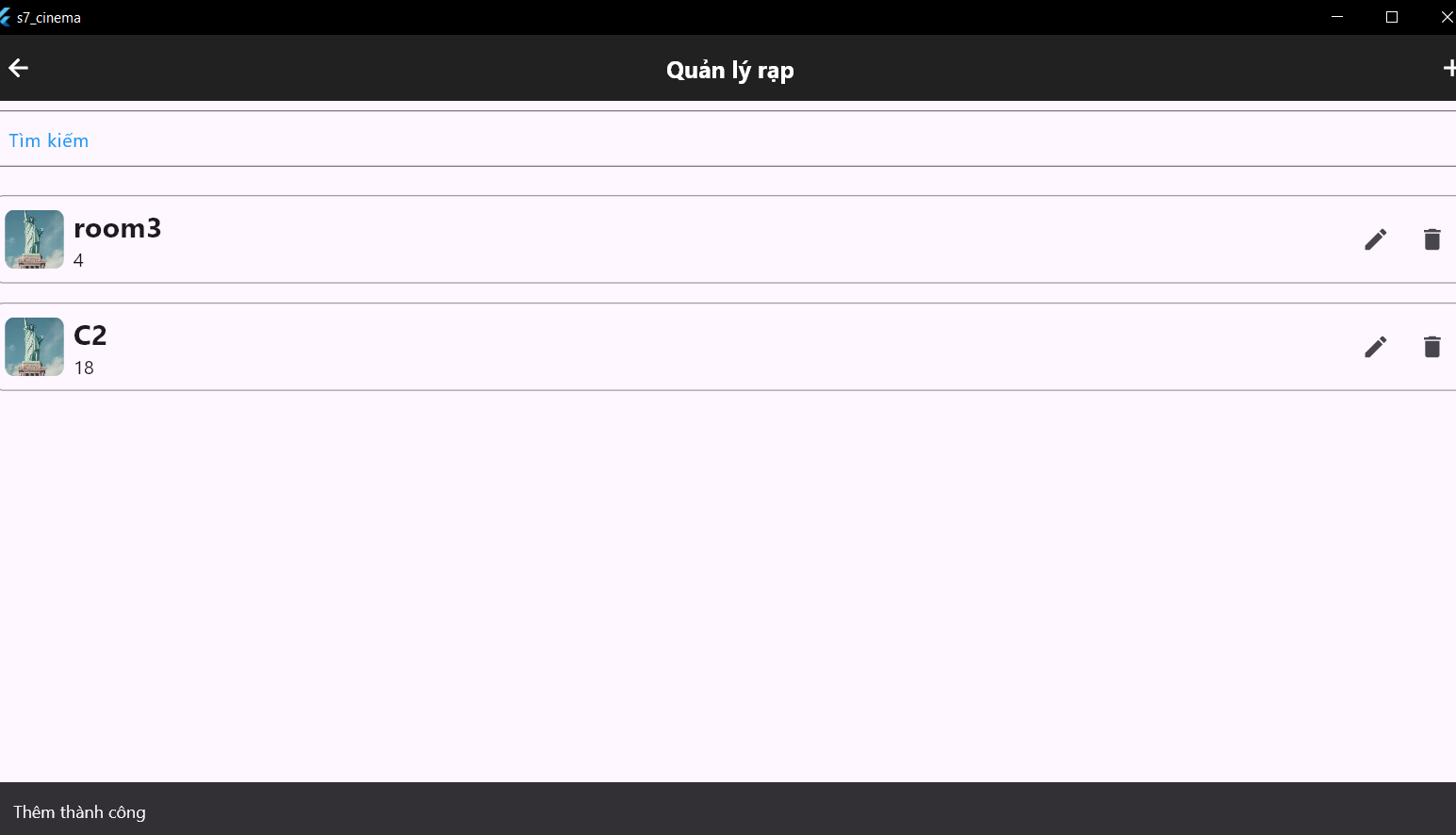


Hình 4.2: Màn hình quản lý phim

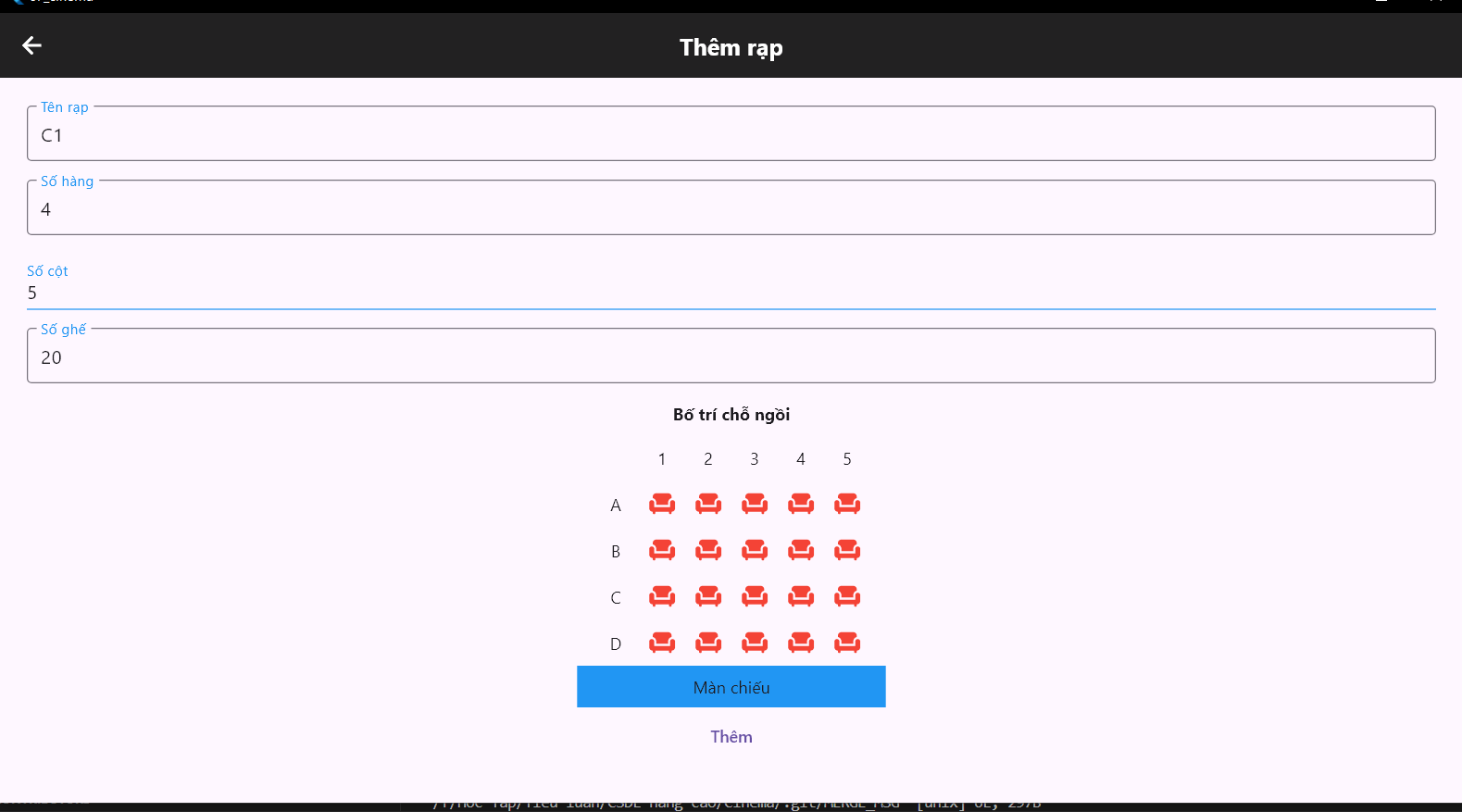


Hình 4.3: Màn hình thêm phim thành công

## Chức năng quản lý phòng chiếu



Hình 4.4: Màn hình quản lý phòng



Hình 4.5: Màn hình thêm mới phòng

## Chức năng đặt vé

A screenshot of a movie book

Description automatically generated

Hình 4.6: Màn hình chức năng đặt vé

# KẾT LUẬN

Kết luận về việc ứng dụng cơ sở dữ liệu MongoDB vào bài toán đặt vé xem phim nêu bật rõ ràng các lợi thế của MongoDB trong việc xây dựng một hệ thống hiện đại và hiệu quả. Với khả năng xử lý dữ liệu phi cấu trúc, MongoDB cho phép lưu trữ và quản lý một lượng lớn thông tin đa dạng liên quan đến người dùng, lịch chiếu phim, ghế ngồi, và các giao dịch thanh toán. Điều này đặc biệt quan trọng trong bối cảnh một hệ thống đặt vé cần phải linh hoạt trong việc xử lý các loại dữ liệu khác nhau và nhanh chóng phản hồi các yêu cầu của người dùng.

Ngoài ra, khả năng mở rộng quy mô của MongoDB là một điểm cộng lớn, cho phép hệ thống dễ dàng đáp ứng nhu cầu tăng trưởng về số lượng người dùng và lượng giao dịch mà không gặp phải các vấn đề về hiệu suất. Trong bối cảnh ngành công nghiệp giải trí, việc có thể mở rộng hệ thống một cách linh hoạt và hiệu quả là yếu tố sống còn để đảm bảo trải nghiệm người dùng luôn mượt mà và không bị gián đoạn.

Khả năng truy vấn mạnh mẽ của MongoDB, kết hợp với khả năng lưu trữ các tài liệu có cấu trúc phức tạp, cho phép việc truy xuất thông tin diễn ra nhanh chóng và chính xác. Điều này rất quan trọng trong một hệ thống đặt vé, nơi mà việc xác nhận và xử lý giao dịch phải được thực hiện trong thời gian thực, đảm bảo người dùng có thể mua vé một cách dễ dàng và tiện lợi.

Cuối cùng, việc sử dụng MongoDB không chỉ giúp tối ưu hóa hiệu suất của hệ thống hiện tại mà còn mở ra nhiều cơ hội cho sự phát triển và mở rộng trong tương lai. Với những tính năng ưu việt như vậy, MongoDB xứng đáng là một lựa chọn hàng đầu cho các hệ thống đặt vé xem phim, góp phần nâng cao trải nghiệm người dùng và đảm bảo sự phát triển bền vững của doanh nghiệp.

# TÀI LIỆU THAM KHẢO

[1] Tài liệu về MongoDB - <https://www.mongodb.com/docs/manual/> . Thời gian truy cập gần nhất 08:32 22/08/2024

[2] Tài liệu về Flutter - <https://docs.flutter.dev/> . Thời gian truy cập gần nhất 20:25 22/08/2024

[3] “Mongodb To Study” by Melanie mitchell - NXB PelicanBook