## 0.1 Module View

Phần này tập trung vào các Màn hình, và phân tích chúng theo hướng MVVM.

Trong phần này có dùng nhiều biểu đồ tuần tự (sequence diagram) để minh họa tương tác của ba thành phần MVVM (cùng với một số thành phần liên quan) trong các ca sử dụng. Có một số điểm chung về các biểu đồ này:

- Trừ khi cần thiết, thành phần View sẽ được lược bỏ cho ngắn gọn.
- Đường thẳng nét đứt thể hiện tính năng data binding (tự động cập nhật View), và thường trỏ về ViewModel. Đáng ra, mũi tên này phải trỏ về View, nhưng do View bị ẩn đi, nên nó trỏ về ViewModel. Mặc dù không được đề cập đến trong phần giới thiệu về MVVM, đây thực ra là một chi tiết đúng về mặt kĩ thuật: ViewModel hoàn toàn đọc được luồng dữ liệu gửi đến View (hoặc ít nhất là đúng trong cách viết ứng dụng Android thông thường).

Các biểu đồ trạng thái cũng có một số chi tiết chung:

- Trừ khi nêu rõ, mọi trạng thái đều có thể là trạng thái bắt đầu (trạng thái khi mở ứng dụng) hoặc kết thúc (khi đóng ứng dụng).
- Mũi tên chuyển trạng thái tương ứng với *tương tác của người dùng*, do vậy thường được ánh xạ đến một phương thức trong View.
- Kí hiệu hình tròn đen chỉ dùng để tả trạng thái đầu *khi lần đầu dùng* yacv.

Biểu đồ lớp thường có một phương thức chung là "Get InputStream from ID". Cách truy cập để lấy `InputStream` của ảnh từ `ComicID` có thể tham khảo từ Màn hình Đọc truyện.

### 0.1.1 Nguồn dữ liệu - Repository - DAO - `ComicParser`

Như đã đề cập ở **??**, yacv sử dụng Kiến trúc Google khuyên dùng, vốn dựa trên MVVM. Phần này nêu rõ hơn cách triển khai MVVM của yacv trong phần nguồn dữ liệu (Model/Repository).

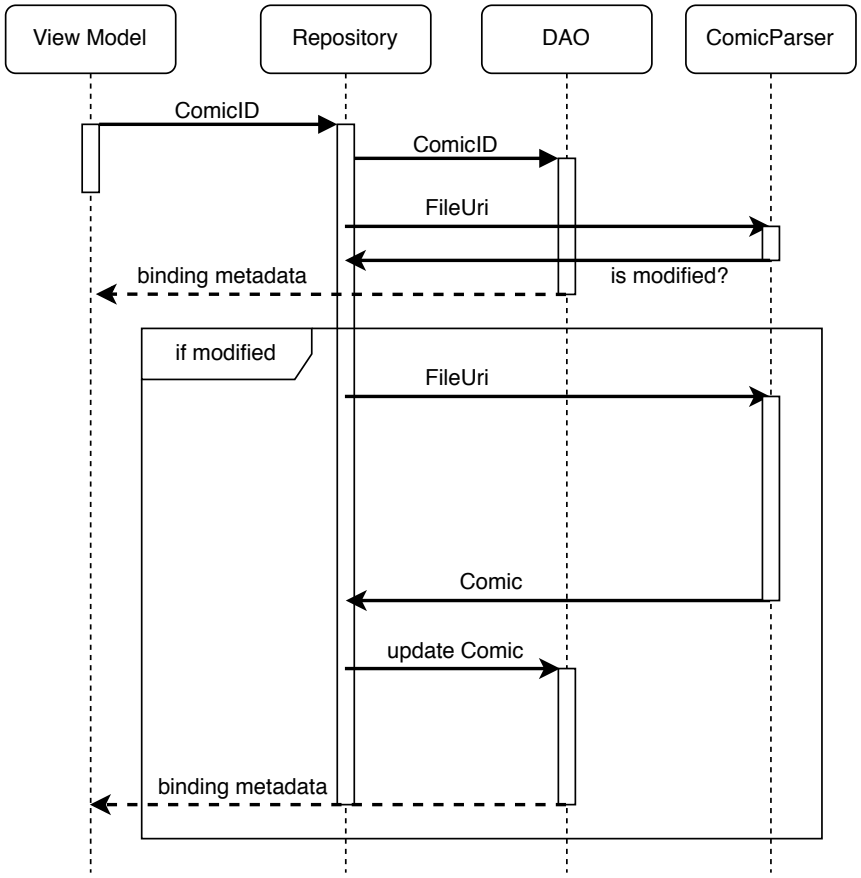
Dựa vào **??**, ta thiết kế được ba nguồn dữ liệu (model) sau:

Bảng 1: Ba nguồn dữ liệu tương đương với **??**

| Tên | Tương đương với | Mục đích |
|---|---|---|
| `ComicParser` | Remote Data Source | Quét tệp để lấy metadata cập nhật nhất |
| DAO | Model | Lấy metadata từ cơ sở dữ liệu, tránh quét đi quét lại |
| Repository | Repository | Tổng hợp hai nguồn trên |

Cụ thể hơn, `ComicParser` là bộ quét metadata tệp truyện (thuộc module Parser & Scanner, sẽ được mô tả sau). Lớp này nhận vào URI rồi trả về metadata của tệp truyện tương ứng dưới dạng đối tượng `Comic`.

Khi cần đọc dữ liệu metadata từ tệp truyện, ba thành phần này tương tác như sau:




Hình 1: Tương tác của ba nguồn dữ liệu, mũi tên gạch đứt thể hiện tính năng data binding

Mấu chốt ở đây là `ComicParser` dù có dữ liệu chính xác (trong trường hợp một ứng dụng khác sửa metadata tệp truyện) nhưng tốc độ rất chậm, còn cơ sở dữ liệu không chính xác nhưng rất nhanh, do đó *cơ sở dữ liệu làm bộ đệm cho parser*.

Repository làm nhiệm vụ gọi cả hai nguồn dữ liệu trên và cập nhật cơ sở dữ liệu (nếu cần) thay cho View/ViewModel. Hiện tại, Repository có thể không làm được nhiều, tuy

nhiên nó giúp ích cho *khả năng mở rộng* của ứng dụng. Ví dụ, trong tương lai yacv có thể liên kết với một bên thứ ba cung cấp metadata cho truyện, khi đó để tích hợp API thì chỉ cần sửa phần Repository.

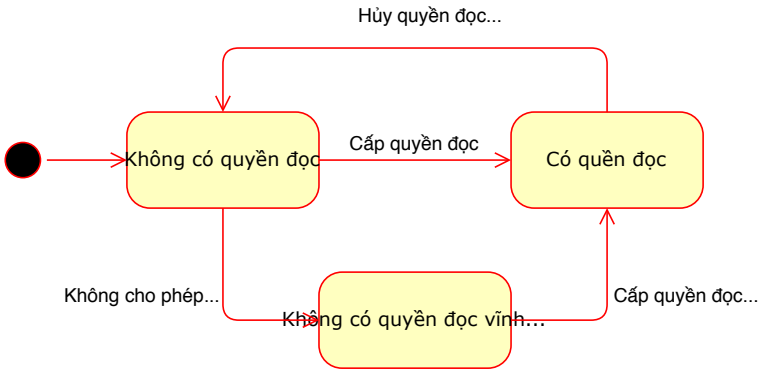
Cũng cần chú ý rằng việc đọc metadata từ tệp tin không phải là yêu cầu của mọi màn hình (cụ thể chỉ Màn hình Metadata cần), do đó trong đa số các ca sử dụng, *DAO đóng vai trò Model*, thay cho Repository. Tương tác trong Hình 1 vẫn được duy trì, tuy không có cả Repository lẫn `ComicParser`.

### 0.1.2 Màn hình Quyền đọc

Do sự phức tạp trong việc xin quyền của Android, một màn hình riêng để xin quyền đọc dữ liệu được tách ra khỏi Màn hình Thư viện, gọi là *Màn hình Quyền đọc*. Màn hình này sẽ là *màn hình đầu tiên* hiển thị khi dùng ứng dụng.

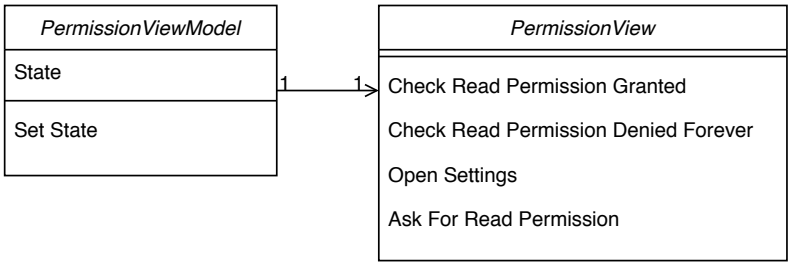
- Nếu có quyền đọc: chuyển ngay sang Màn hình Thư viện
- Nếu không: nêu lí do cần quyền, gợi ý người dùng cấp quyền

Hình sau mô tả trạng thái cấp quyền đọc của yacv (cũng như mọi quyền của một ứng dụng Android cơ bản nói chung).




Hình 2: Trạng thái cấp quyền của một ứng dụng Android

Dựa theo Hình 2, ta có biểu đồ lớp của ViewModel và View như sau:




Hình 3: Biểu đồ lớp của Màn hình Quyền đọc

### 0.1.3 Màn hình Thư viện

Màn hình Thư viện là một trong hai màn hình của ca sử dụng Duyệt truyện, bên cạnh Màn hình Thư mục.

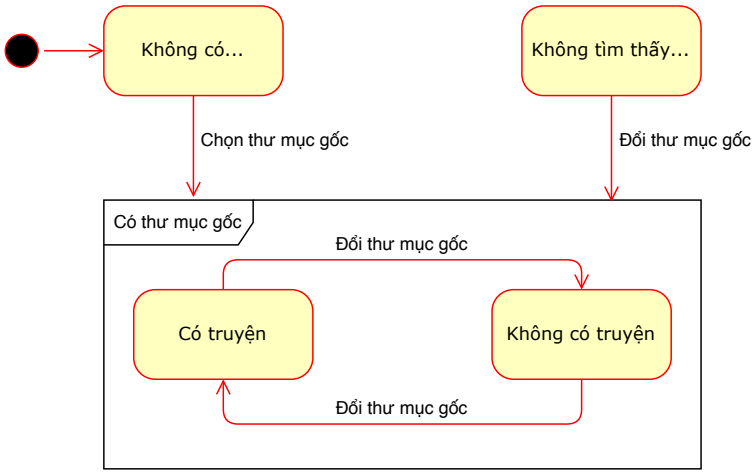
Như đã phân tích ở **??**, Màn hình Thư viện cần hiển thị cả lỗi và gợi ý, bên cạnh việc hiển thị danh sách thư mục và chọn thư mục gốc. Do đó, phần này chia ra làm hai phần con tương ứng.

#### Chọn thư mục gốc và hiển thị danh sách thư mục

Để chọn thư mục gốc, người dùng ấn nút Đổi thư mục gốc để kích hoạt hộp thoại Chọn thư mục (picker), rồi chọn một thư mục trong đó. Luồng chạy của yacv sẽ được nêu sau, ở mục riêng về Scanner.

#### Ngoại lệ trong Màn hình Thư viện

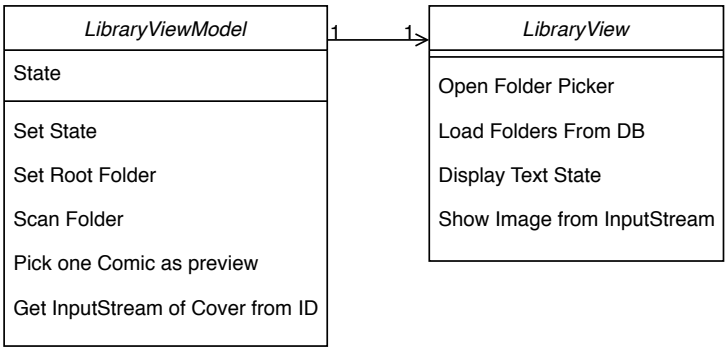
Ngoại lệ ở đây chỉ cả trường hợp không tìm thấy thư mục, lẫn trường hợp không quét được thư mục vì các lí do đã nêu trong **??**. Khi này, ứng dụng hiển thị một hàng chữ để gợi ý về việc nên làm.




Hình 4: Trạng thái của Màn hình Thư viện

Hình sau là biểu đồ trạng thái, cũng là mô tả về nội dung gợi ý. Riêng trạng thái "Có truyện" là trạng thái hiển thị danh sách thư mục trong luồng cơ bản đã nêu trên, tức thư mục được hiển thị đầy đủ thay vì chỉ hiện thông báo lỗi.

Tổng hợp lại, ta có biểu đồ lớp của Màn hình Thư viện như sau:




Hình 5: Biểu đồ lớp của Màn hình Thư viện

Nhắc lại, cách truy cập để lấy `InputStream` của ảnh từ `ComicID` có thể tham khảo từ Màn hình Đọc truyện.

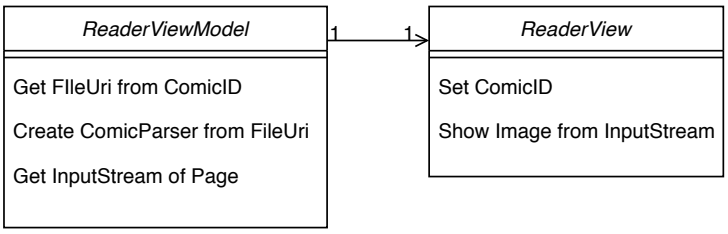
### 0.1.4 Màn hình Thư mục

Màn hình Thư mục là một trong hai màn hình của ca sử dụng Duyệt truyện, bên cạnh Màn hình Thư viện.

Trong khi phân tích yêu cầu, ta đã phân tích được rằng màn hình hiển thị danh sách truyện - một phần trong ca sử dụng tìm kiếm - phải có giao diện giống Màn hình Thư mục, vì đều hiển thị danh sách truyện. Do đó, hai màn hình này được gộp lại, gọi chung là *Màn hình Danh sách truyện*, và sẽ được mô tả sau.

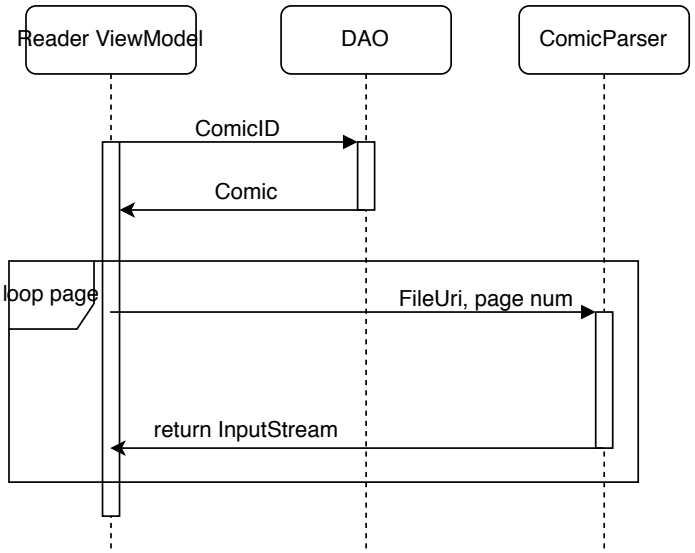
### 0.1.5 Màn hình Đọc truyện

Để hiển thị các trang truyện, Màn hình Đọc truyện cần nhận `ComicID` (hoặc một đối tượng `Comic` hoàn chỉnh, tuy nhiên cốt yếu vẫn là thông tin `ComicID`) của một tệp truyện, sau đó đưa cho `ComicParser` để lấy luồng đọc cho từng trang truyện. Hình 7 là biểu đồ tuần tự của màn hình này.




Hình 6: Biểu đồ lớp của Màn hình Đọc truyện

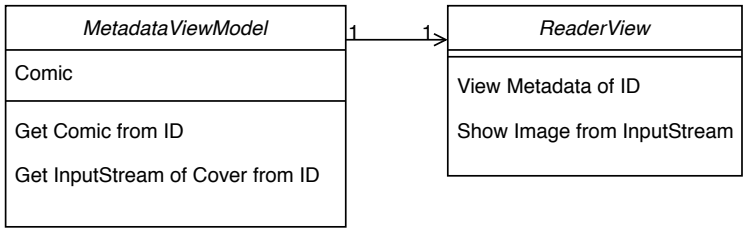
Ta cũng có biểu đồ lớp tương ứng trong Hình 6.




Hình 7: Biểu đồ tuần tự của Màn hình Đọc truyện

### 0.1.6 Màn hình Metadata

Màn hình Metadata tuân theo biểu đồ tuần tự đã nêu ở Hình 1. Biểu đồ lớp tương ứng của màn hình này như sau:




Hình 8: Biểu đồ lớp của Màn hình Metadata

### 0.1.7 Màn hình Tìm kiếm

Màn hình Tìm kiếm là hai trong ba màn hình của ca sử dụng tìm kiếm truyện, bên cạnh Màn hình Danh sách truyện (là tổng quát hóa của Màn hình Thư mục, đã nhắc ở trên). Màn hình Danh sách truyện sẽ được thiết kế ở ngay mục sau, còn mục này tập trung vào Màn hình Tìm kiếm.

Không khó để thấy thực ra Màn hình Tìm kiếm Tổng quan và Màn hình Tìm kiếm Chi tiết thực ra là một màn hình, về mặt thị giác:

- Điểm giống:
  - Cả hai cùng hiển thị danh sách.

  - Các phần tử cùng loại trong hai màn hình có cách hiển thị giống nhau, chuyển đến các màn hình giống nhau.

- Điểm khác: Danh sách trong Màn hình Tìm kiếm Tổng quan có *thêm*:
  - Hiển thị bìa với một số kết quả
  - Có thanh ngăn cách
  - Có nút "Xem thêm"

Do vậy, nếu thiết kế phù hợp, hoàn toàn có thể gộp hai màn hình này. Thiết kế sau giúp thỏa mãn việc này:

- Màn hình nhận vào một tham số chứa *câu truy vấn*. Tham số này thuộc một trong hai kiểu:
  - `QuerySingleType`: chứa câu truy vấn và *một* bảng để tìm kiếm
  - `QueryMultipleTypes`: chứa câu truy vấn và một *danh sách* bảng để tìm kiếm

  "Bảng để tìm kiếm" thực ra là một số quy định trước, ví dụ nếu là số `0` thì bảng được tìm là `Comic`,…
- ViewModel tìm kiếm dựa vào tham số truy vấn
  - Nếu tham số là `QuerySingleType`: truy vấn và hiển thị kết quả như thông thường
  - Nếu tham số là `QueryMultipleTypes`: truy vấn các bảng, gộp kết quả lại và thêm kiểu kết quả đặc biệt là *Placeholder* và *SeeMore* vào vị trí phù hợp để hiển thị lần lượt nhóm kết quả và nút "Xem thêm"
- Các kết quả, bao gồm hai dạng kết quả đặc biệt ở trên, cài đặt chung giao diện `Metadata`, để có thể được gộp thành một danh sách

Nói ngắn gọn, hai màn hình cùng hiển thị một danh sách, danh sách này có một số phần tử đánh dấu đặc biệt. Ta dùng lại ví dụ về truy vấn `Watchmen` ở **??** để minh họa:

- Truy vấn `QueryMultipleTypes` được gửi đến Màn hình Tìm kiếm. Câu truy vấn là `Watchmen`, các bảng cần tìm là mọi bảng.
- ViewModel tìm `Watchmen` trong mọi bảng, tìm được:
  - 3 tệp truyện trong bảng `Comic`

  - 2 bộ truyện trong bảng `Series`

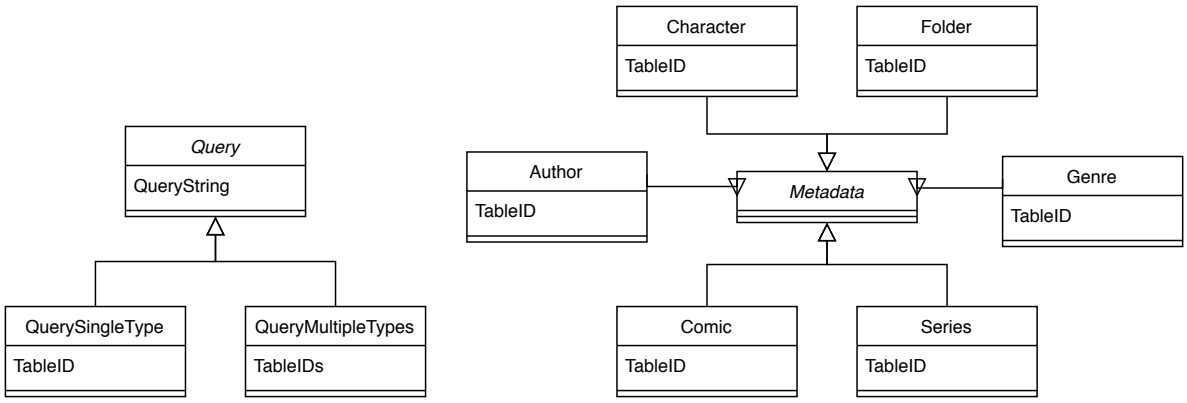
- Màn hình hiển thị:

  1. Dòng `Truyện`, rồi 3 tệp truyện (cùng với ảnh bìa)
  2. Dòng `Bộ truyện`, 1 bộ truyện, rồi dòng `Xem thêm`

- Khi ấn vào:

  - Một trong ba tệp truyện: Đưa đến Màn hình Đọc truyện tương ứng.
  - Một bộ truyện: Chuyển đến Màn hình Danh sách truyện, chứa các truyện trong bộ đó.
  - Nút `Xem thêm`: Chuyển đến Màn hình Tìm kiếm, lần này tham số là một `QuerySingleType`, với câu truy vấn là `Watchmen`, còn bảng để tìm là `Series`. Hai bộ truyện kết quả được hiển thị đầy đủ. Chọn một bộ truyện lúc này giống với chọn bộ truyện ở trên (sang Màn hình Danh sách truyện).

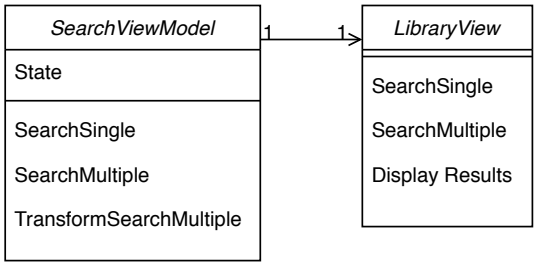
Biểu đồ lớp của các đối tượng liên quan như sau:




(a) Biểu đồ lớp `Query` (b) Biểu đồ lớp `Metadata`

Hình 9: Biểu đồ các lớp liên quan đến Màn hình Tìm kiếm

Biểu đồ lớp của bản thân Màn hình Tìm kiếm như sau:




Hình 10: Biểu đồ lớp của Màn hình Tìm kiếm

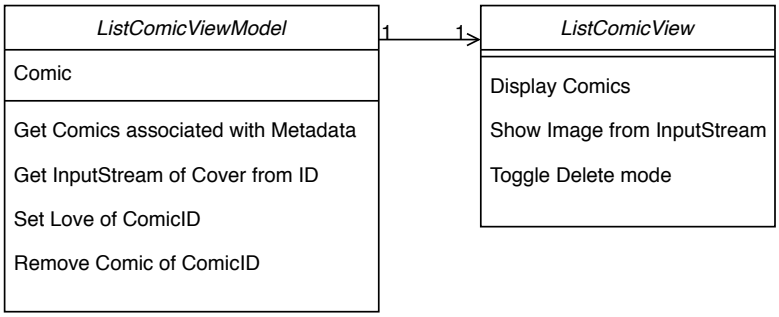
Do mỗi DAO trả kết quả của bảng tương ứng về ở dạng danh sách, nên khi dùng `QueryMultipleTypes`, các danh sách kết quả lẻ này tổng hợp, và thêm hai kiểu kết quả đặc biệt. Hàm `TransformSearchMultiple()` là để "làm phẳng" mảng kết quả hai chiều như trên.

### 0.1.8 Màn hình Danh sách truyện

Màn hình này là màn hình thứ ba trong chuỗi các màn hình liên quan đến ca sử dụng tìm kiếm, đồng thời đóng vai trò của Màn hình Thư mục (do là phiên bản tổng quát hơn của nó).

Màn hình này nhận vào một tham số kiểu `Metadata` thay vì một `Query`, và trả về danh sách các *tệp truyện* - `Comic` - có liên kết với tham số đầu vào.

Biểu đồ lớp của Màn hình Danh sách truyện như sau:




Hình 11: Biểu đồ lớp của Màn hình Danh sách truyện