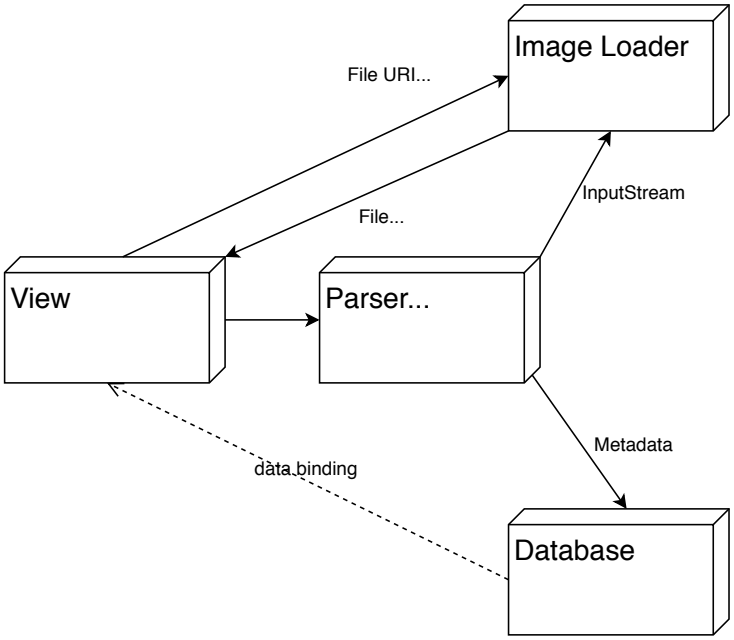
Chương 1

Thiết kế

Chương này tập trung vào thiết kế của ứng dụng, là triển khai cụ thể của **??**.



Hình 1.1: Kiến trúc tổng quan của ứng dụng

Kiến trúc tổng quan của yacv rất đơn giản, gồm 4 module như hình:

1. yacv *quét* metadata tệp truyện bằng module **Parser & Scanner** và lưu kết quả quét vào *cơ sở dữ liệu*, tức module **Database**
2. Các *Màn hình* trong module **View** hiển thị dữ liệu cho người dùng
3. Khi người dùng đọc truyện, yacv trích xuất và lưu đệm tệp ảnh bằng module **Image Loader**

4. Khi người dùng xem metadata, yacv trích xuất và hiển thị thông tin tệp truyện lên Màn hình bằng data binding với module **Database**

Ở ?? - ??, trong phần về MVVM, yacv được giới thiệu là có sử dụng kiến trúc này. Tuy nhiên, MVVM chỉ là một phần nhỏ của ứng dụng, chủ yếu liên quan đến việc hiển thị dữ liệu nên chỉ có ý nghĩa khi xét đến các thành phần trong module **View**.

yacv chỉ thiết kế cho *một người dùng*, do đó có rất ít tương tác, dẫn đến kiến trúc tối giản và rời rạc như trên. Các tiểu mục sau sẽ đi sâu vào các module này.

1.1 Module Database

Thông thường mục này được tách riêng ra, xếp vào mục *Thiết kế cơ sở dữ liệu*, ngang hàng với mục Thiết kế hướng đối tượng. Tuy nhiên, yacv còn cần xử lí dữ liệu khác quan trọng không kém là dữ liệu ảnh. Do không còn có vai trò trung tâm, duy nhất, phần cơ sở dữ liệu chỉ được coi là một module trong thiết kế hướng đối tượng của ứng dụng.

yacv chọn SQLite vì đây là một cơ sở dữ liệu gọn nhẹ nhúng sẵn trong Android. SQLite sử dụng mô hình quan hệ, do đó thiết kế bảng cần đảm bảo được chuẩn hóa (normalization).

Do không cần quản lí người dùng, cơ sở dữ liệu của yacv chỉ dùng để *lưu thông tin metadata*, cho phép ứng dụng quét dữ liệu ít lần hơn và tìm kiếm truyện. Theo như yêu cầu về metadata ở hai Phụ lục, và sau khi chuẩn hóa, ta có lược đồ cơ sở dữ liệu như sau:

Các bảng thực thể gồm:

- `Comic`: lưu thông tin *tập truyện lẻ*, là bảng trung tâm
- `Series`: lưu thông tin *bộ truyện*
- `Author`: lưu tên tác giả
- `Role`: lưu vai trò của tác giả trong một tập truyện
- `Character`: lưu tên nhân vật
- `Genre`: lưu tên thể loại truyện

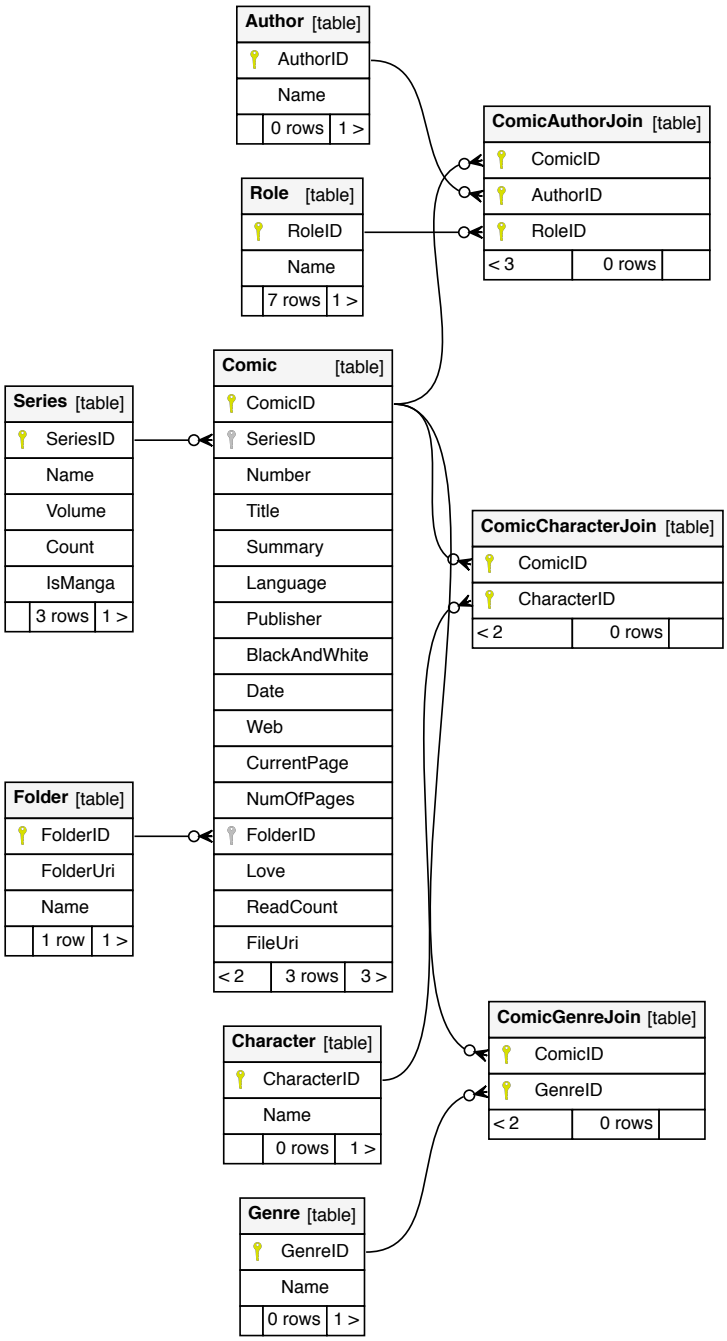
Một hạn chế quan trọng của các bảng `Character` và `Author` là chúng chỉ lưu thông tin tên, và chỉ phân biệt với nhau bằng tên. Nếu có hai tác giả/nhân vật trùng tên, yacv không thể phát hiện và hiển thị riêng.

Xét bảng trung tâm `Comic`. Bảng này có một số trường không phải metadata mà dùng để lưu thông tin của riêng ứng dụng, gồm:

- `CurrentPage`: lưu trang đang đọc
- `Love`: lưu trạng thái Yêu thích
- `ReadCount`: lưu số lần đọc

Trong lược đồ, có nhiều trường nhìn qua không cần thiết nhưng thực tế có ích, do thư viện SAF đã mô tả ở **??**:

- Trường `FileUri` trong `Comic`: Lưu đường dẫn của tệp truyện ở dạng URI.
- Trường `FolderUri` trong `Folder`: Lưu đường dẫn của thư mục ở dạng URI.
- Trường `Name` trong `Folder`: Tên thư mục. Thông thường nếu có đường dẫn, có thể tìm ra tên thư mục rất nhanh, tuy nhiên cũng do SAF mà việc này trở nên khó khăn, nên cần lưu riêng trường này.



Hình 1.2: Lược đồ cơ sở dữ liệu của yacv

Các trường URI đều cần có ràng buộc `UNIQUE`, do mỗi URI trỏ đích danh đến một đối tượng.

Ta xem xét đến các bảng nối:

- `ComicCharacterJoin`:
  - Mỗi tập truyện có thể có nhiều nhân vật và ngược lại, do đó `Comic` và `Character` có quan hệ Nhiều - Nhiều.

  - Chú ý rằng các nhân vật có quan hệ với tập truyện chứ không phải bộ truyện, vì có nhân vật phụ (không xuất hiện trong mọi tập truyện).

- `ComicAuthorJoin`:

  - Mỗi tập truyện có thể có nhiều tác giả và ngược lại, do đó `Comic` và `Author` có quan hệ Nhiều - Nhiều.
  - Chú ý rằng các tác giả có quan hệ với tập truyện chứ không phải bộ truyện, vì mô hình xuất bản nhiều truyện tranh là nhà xuất bản sở hữu nhân vật và thuê người viết.
  - Đồng thời, một tác giả có thể giữ vai trò khác nhau trong các bộ truyện khác nhau, do đó bảng này còn nối với bảng `Role`.

- `ComicGenreJoin`: Mỗi tập truyện có thể có nhiều thể loại khác nhau và ngược lại, do đó `Comic` và `Genre` có quan hệ Nhiều - Nhiều.

Do dùng Room, mỗi bảng ứng với một lớp. Các truy vấn với bảng cần đóng gói dữ liệu vào các lớp này, trước khi gửi đến hoặc nhận về từ *DAO* (Data Access Object). Mỗi câu lệnh lại được chuyển thành một hàm trong DAO.