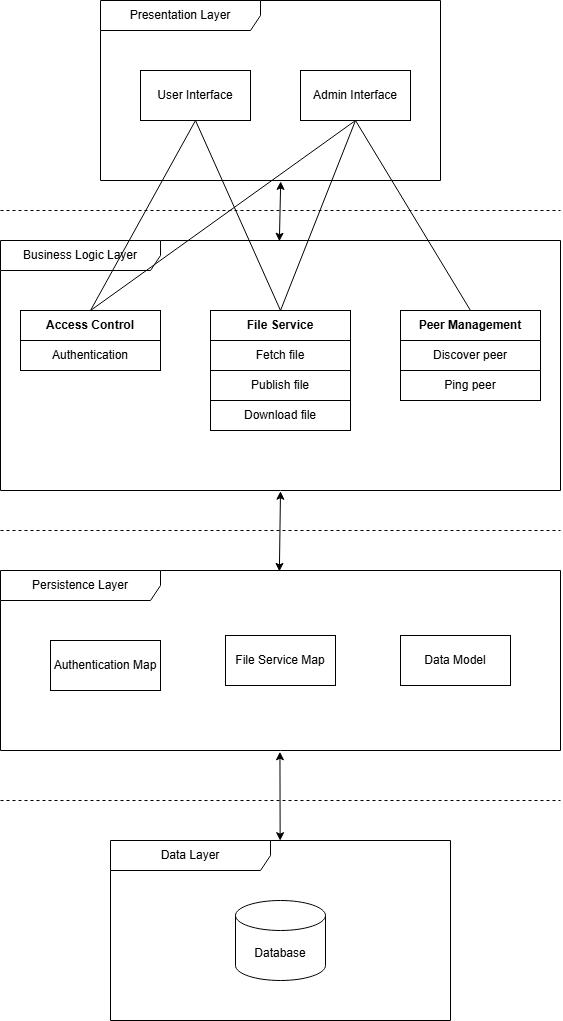
**ARCHITECTURE DESIGN**

1. **Layer Architecture**

Kiến trúc phân tầng:



Hệ thống gồm 2 giao diện chính là dành cho user và cho admin quản lý. Lớp Presentation biểu diễn giao diện dành cho người dùng, đứng đầu tiên trong sơ đồ.

Trong hệ thống, các chức năng chính được biểu diễn ở tầng Business Logic Layer, user có thể đăng nhập, sử dụng dịch vụ fetch file, publish file và download file, admin có thể quản lí các file và peer.

Lớp Persistence là cầu nối đến database, có các mapping đến các loại hình tương ứng.

Lớp database là nơi lưu trữ dữ liệu của hệ thống, cần có mapping để có thể truy suất nhanh chóng cho các dịch vụ phía trên.

1. **Data storage**

Đối với bài toán lưu trữ các thông tin cần thiết cho ứng dụng chia sẻ file P2P, nhóm xác định có 2 thực thể cần thiết lưu trữ là File và Peer. Nhóm sử dụng hệ quản trị cơ sở dữ liệu MongoDBCompass để lưu trữ. Document lưu trữ như sau:

* File:

{

\_id: ObjectID,

name: string,

size: number,

hash\_info: string,

peer: {

address: string,

port: string

}

}

* Peer:

{

\_id: ObjectID,

address: string,

port: string,

username: string,

password: string

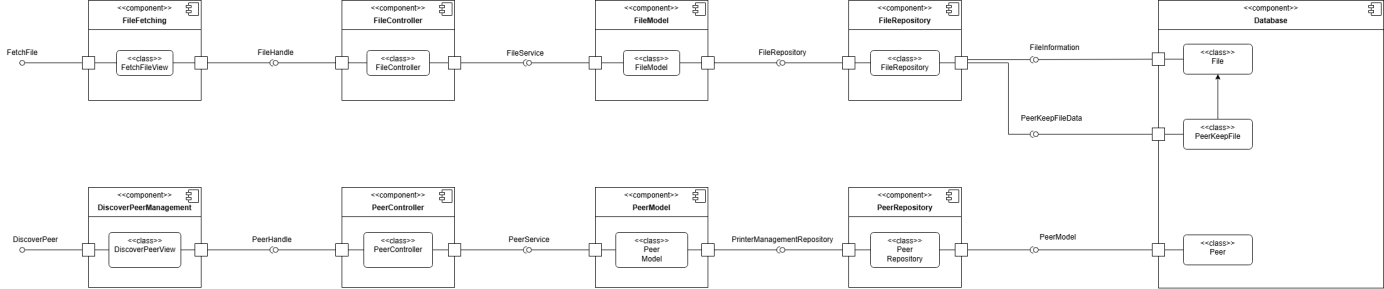
}

1. **API management**

API cung cấp khả năng gửi yêu cầu từ peer đến tracker và peer nhận thông tin từ tracker. Các API chính của ứng dụng chia sẻ file:

* API xác thực người dùng: Đảm bảo an toàn khi sử dụng ứng dụng chia sẻ file và cung cấp khả năng sử dụng ứng dụng thông qua đăng nhập và xác thực người dùng.
* API thông báo và truy suất file: Cho phép người dùng thông báo cho tracker file peer muốn chia sẻ và peer có thể tìm kiếm các file đang được chia sẻ.
* API kiểm tra các peer: Cho phép người dùng với vai trò admin kiểm tra tình trạng hoạt động và thông tin của các peer đang tham gia chia sẻ.

1. **Component diagram**

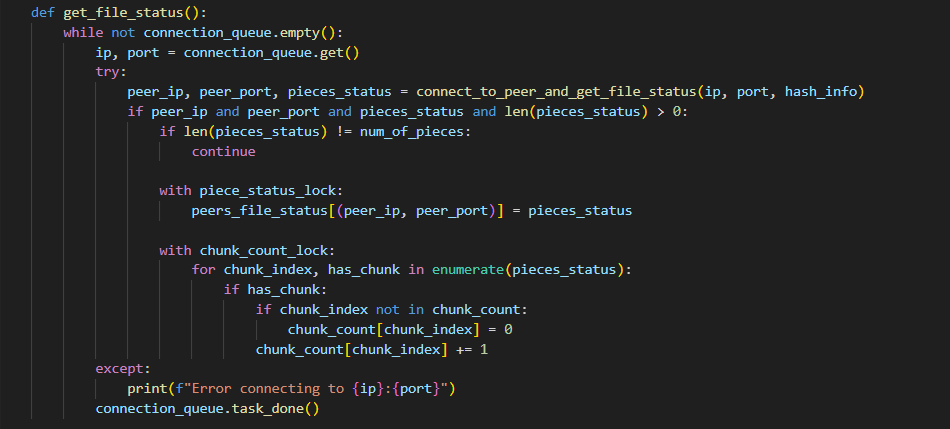


Các component diagram cho fetch file module và discover peer module:

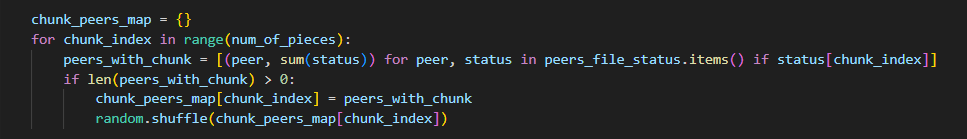
* Fetch view và Discover peer view: chứa các giao diện người dùng và dịch vụ fetch các file đang được chia sẻ và các peer đang tham gia chia sẻ.
* File Controller và Peer Controller: xử lí yêu cầu từ View, sau đó chuyển yêu cầu đến các dịch vụ ở component tiếp theo.
* File Model và Peer Model: thực hiện các yêu cầu và tương tác với kho dữ liệu để đáp ứng dịch vụ.
* File Repository và Peer Repository: cung cấp dữ liệu cho các dịch vụ và thực hiện thao tác trên cơ sở dữ liệu.
* Database: Chứa các dữ liệu về File - peer chứa file và Peer.

**EXTRA - Download strategies: Rarest-first**

* Sử dụng chunk\_count để theo dõi số lượng peer sở hữu mỗi mảnh file. chunk\_count được cập nhật cho mỗi mảnh mà một peer sở hữu.



* Tạo chunk\_peers\_map, ánh xạ mỗi mảnh với các peer có chứa mảnh đó. Bằng cách này, hệ thống biết peer nào có mảnh hiếm nhất và có thể ưu tiên tải xuống từ các peer đó trước.



* chunk\_queue chứa các mảnh cần tải xuống. Hệ thống sẽ sử dụng chunk\_queue để duyệt qua các mảnh theo thứ tự ưu tiên, trong đó mảnh hiếm nhất sẽ được tải xuống trước tiên.

