Chương 5 tổng hợp các giải pháp và đóng góp mà em cho là quan trọng và nổi bật nhất trong quá trình xây dựng hệ thống NFT marketplace. Khác với các chương trước chủ yếu mô tả yêu cầu, thiết kế, chương này tập trung vào cách em phân tích để xác định các vấn đề gặp phải, cách lựa chọn giải pháp phù hợp và kết quả đạt được.

0.1 Đồng bộ dữ liệu từ blockchain về cơ sở dữ liệu

0.1.1 Giới thiệu

Trong NFT marketplace, các thao tác cốt lõi như tạo NFT, niêm yết, mua/bán đều phát sinh giao dịch trên blockchain. Trong khi đó, hệ thống off-chain (backend và cơ sở dữ liệu) lại cần lưu trữ và tổ chức dữ liệu để phục vụ tìm kiếm, thống kê, hiển thị lịch sử và tối ưu trải nghiệm người dùng. Vì vậy, một bài toán trọng tâm là làm sao để trạng thái hiển thị off-chain bám sát trạng thái thực tế on-chain, đồng thời vẫn giữ được trải nghiệm sử dụng mượt mà và dễ kiểm soát.

0.1.2 Giải pháp

Hệ thống chỉ thực hiện cập nhật dữ liệu phía backend sau khi giao dịch trên blockchain đã có kết quả xác định. Cụ thể, phía frontend thực hiện giao dịch thông qua ví, chờ giao dịch được ghi nhận thành công hoặc tối thiểu nhận được transaction hash hợp lệ. Sau đó, frontend gọi các API để yêu cầu backend xử lý và cập nhật dữ liệu off-chain.

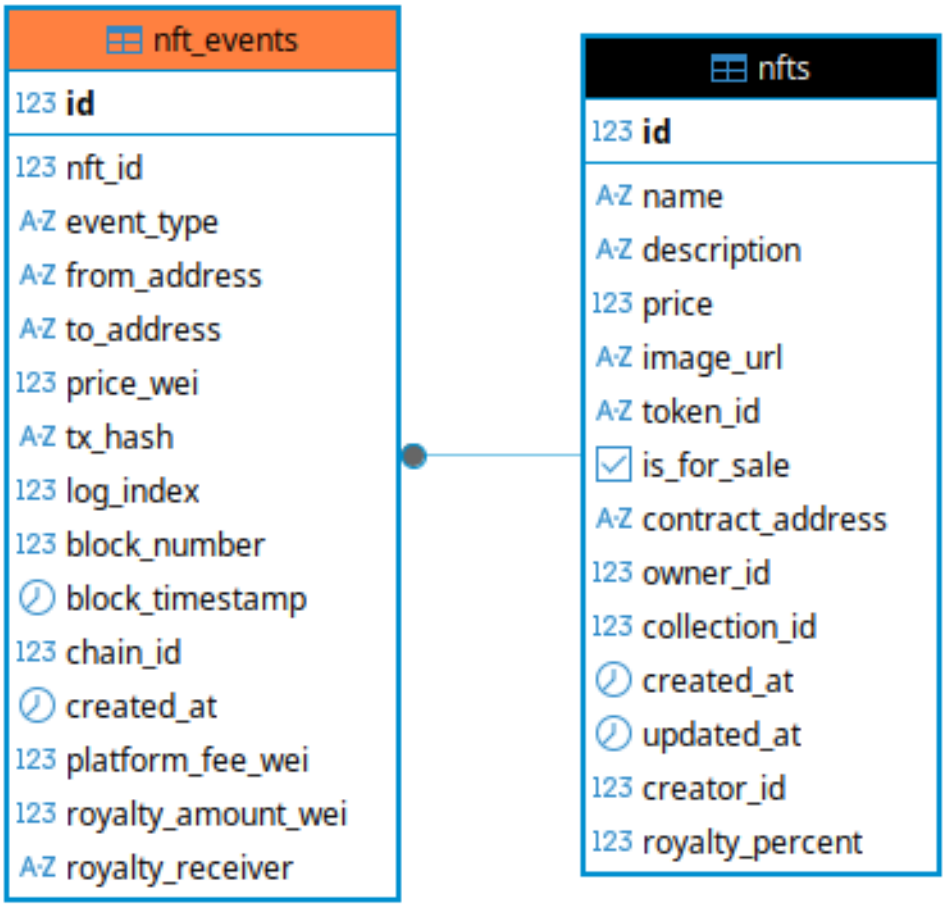
Backend không sử dụng trực tiếp toàn bộ dữ liệu do client gửi lên để cập nhật trạng thái, mà lấy transaction hash làm căn cứ kiểm tra lại giao dịch trên blockchain. Backend truy vấn transaction receipt để xác định trạng thái thực thi của giao dịch (thành công hoặc thất bại). Trên cơ sở đó, backend trích xuất các thông tin cần thiết phục vụ cập nhật hệ thống như tokenId, địa chỉ người mua hoặc người bán, giá trị giao dịch và địa chỉ hợp đồng từ dữ liệu receipt hoặc từ các log cơ bản của giao dịch.

Quá trình cập nhật dữ liệu được tổ chức theo nguyên tắc on-chain đóng vai trò nguồn dữ liệu chuẩn cho các thông tin liên quan đến quyền sở hữu và giao dịch. Dữ liệu off-chain được cập nhật nhằm mục đích tối ưu truy vấn và hiển thị cho người dùng. Trong trường hợp phát sinh sai lệch, hệ thống ưu tiên đối chiếu và hiệu chỉnh theo trạng thái on-chain để đảm bảo tính nhất quán của thông tin quan trọng.

0.1.3 Kết quả đạt được

Với cách làm này, backend sẽ luôn được đồng bộ dữ liệu từ blockchain, giúp hiệu năng và trải nghiệm người dùng tốt hơn thay vì phải truy vấn trực tiếp trên blockchain(tốc độ rất chậm). Đồng thời khi làm chủ dữ liệu, hệ thống có thể dễ

dàng phát triển và mở rộng tính năng mới dựa vào lượng data sẵn có.



Hình 0.1: Table lưu trữ thông tin đồng bộ từ blockchain

0.2 Cơ chế xác minh giao dịch để tránh cập nhật sai hoặc bị giả mạo

0.2.1 Giới thiệu

Trong hệ thống NFT marketplace, dữ liệu off-chain trên backend được sử dụng để phục vụ truy vấn, hiển thị và quản lý trạng thái nghiệp vụ. Nếu backend chấp nhận yêu cầu cập nhật trạng thái chỉ dựa trên dữ liệu do client gửi lên như tokenId, giá hoặc địa chỉ người mua, hệ thống có thể phát sinh cập nhật sai lệch. Nguyên nhân có thể đến từ lỗi phía frontend, thao tác không mong muốn của người dùng, hoặc các yêu cầu giả mạo được gửi trực tiếp đến API.

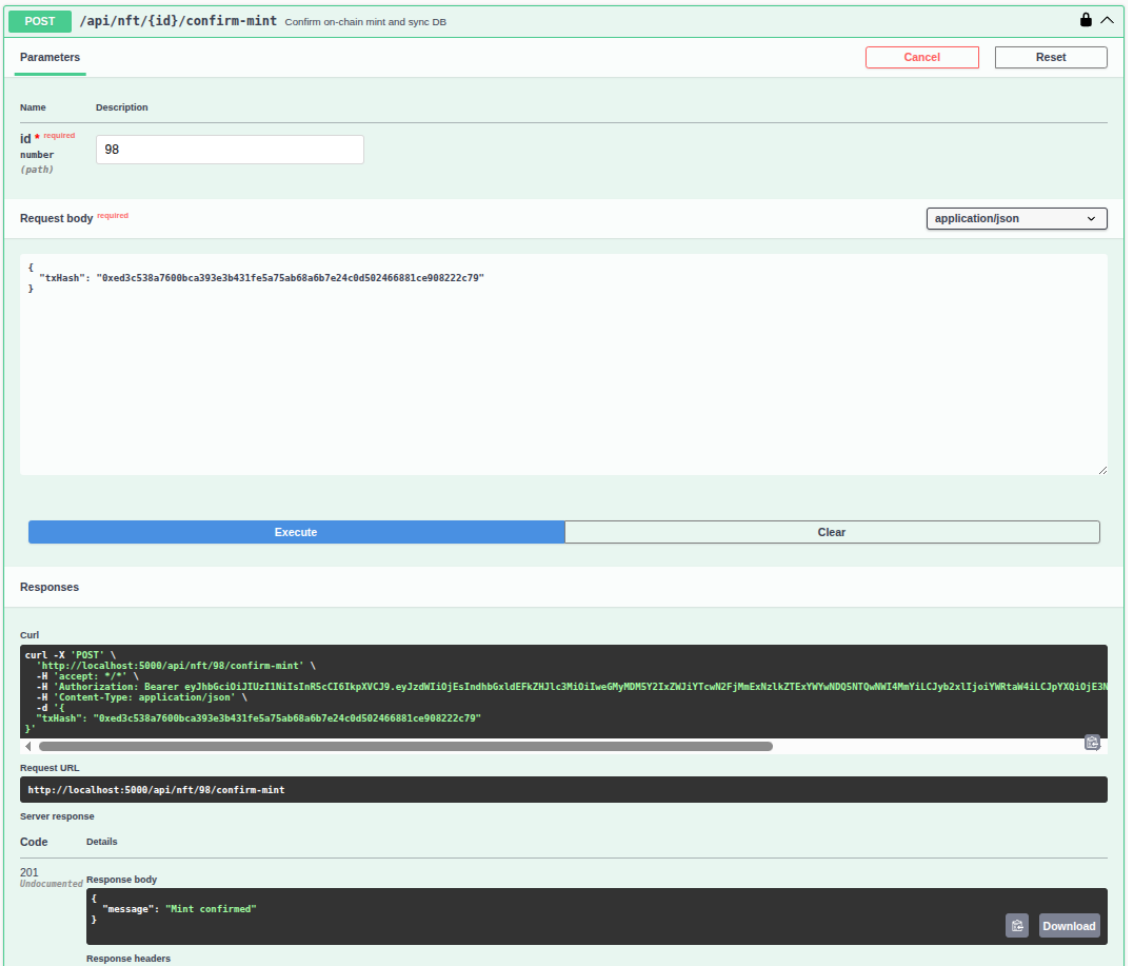
Khi trạng thái quan trọng bị cập nhật sai như trạng thái đã mua, đã niêm yết hoặc đã mint, dữ liệu hiển thị sẽ không còn phản ánh đúng trạng thái thực tế trên blockchain. Điều này ảnh hưởng trực tiếp đến tính tin cậy của hệ thống và trải nghiệm người dùng. Vì vậy, yêu cầu đặt ra là mỗi lần cập nhật off-chain cần có căn cứ từ giao dịch on-chain và hạn chế tối đa việc client tự khai báo trạng thái.

0.2.2 Giải pháp

Giải pháp của hệ thống được xây dựng dựa trên nguyên tắc sử dụng transaction hash như bằng chứng giao dịch và bắt buộc xác minh trước khi cập nhật dữ liệu off-chain. Cụ thể, sau khi người dùng thực hiện giao dịch thông qua ví trên frontend và gửi yêu cầu xác nhận lên backend, hệ thống sẽ truy vấn transaction receipt để kiểm tra giao dịch có tồn tại trên blockchain và đã thực thi thành công hay chưa. Đồng thời, backend đối chiếu địa chỉ ví tham gia giao dịch với danh tính người dùng đang đăng nhập nhằm đảm bảo giao dịch được tạo bởi đúng ví và đúng ngữ cảnh nghiệp vụ. Tiếp theo, backend kiểm tra giao dịch có tương tác đúng với hợp đồng thông minh và phương thức tương ứng của chức năng (ví dụ: mint/list/buy/cancel), tránh trường hợp client cung cấp nhầm một transaction hash không liên quan. Bên cạnh đó, các API thay đổi trạng thái được thiết kế theo hướng hạn chế quyền cập nhật trực tiếp: thay vì cho phép client gửi trạng thái tuỳ ý để backend ghi nhận ngay, các endpoint quan trọng chỉ chấp nhận transaction hash và một số thông tin tối thiểu cần thiết cho việc đối chiếu, qua đó giảm rủi ro cập nhật sai hoặc bị thao túng dữ liệu off-chain.

0.2.3 Kết quả đạt được

Cơ chế xác minh theo transaction hash giúp dữ liệu off-chain không phụ thuộc hoàn toàn vào dữ liệu do client cung cấp, đồng thời tăng độ tin cậy cho các luồng nghiệp vụ chính như mint, niêm yết và mua NFT. Trong quá trình kiểm thử, hệ thống có thể phát hiện và từ chối cập nhật đối với các trường hợp transaction hash không hợp lệ, giao dịch không liên quan đến chức năng hoặc giao dịch thực thi thất bại. Nhờ đó, hệ thống hạn chế được sai lệch giữa dữ liệu hiển thị và trạng thái thực tế trên blockchain.



Hình 0.2: API xác nhận giao dịch thông qua blockchain

0.3 Thiết kế cập nhật dữ liệu sau giao dịch theo hướng idempotent và có trạng thái trung gian

0.3.1 Giới thiệu

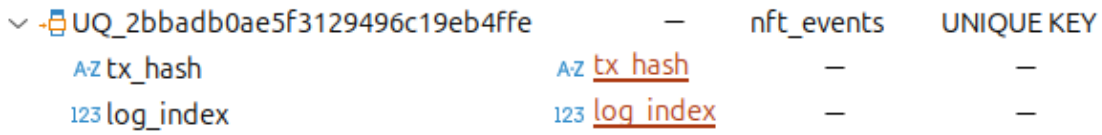
Trong môi trường thực tế, người dùng có thể gặp tình trạng mạng chập chờn hoặc thao tác lặp lại (refresh trang, bấm lại nút xác nhận), dẫn tới việc FE gọi API xác nhận nhiều lần cho cùng một giao dịch. Nếu backend mỗi lần nhận request đều cập nhật thêm một lần nữa, hệ thống có thể phát sinh lỗi như tạo bản ghi trùng, cộng thống kê sai, hoặc đẩy trạng thái NFT sang sai trạng thái.

0.3.2 Giải pháp

Dùng idempotency theo txHash. Mỗi giao dịch sau khi được xác nhận sẽ tương ứng với một bản ghi xử lý ở database. Khi nhận request xác nhận, backend kiểm tra xem txHash đã được xử lý thành công trước đó hay chưa; nếu rồi thì trả về kết quả hiện tại thay vì cập nhật lại. Nhờ vậy, việc FE gọi lại API không làm thay đổi kết quả cuối.

0.3.3 Kết quả đạt được

Thiết kế này giúp hệ thống ổn định hơn khi người dùng thao tác lặp hoặc khi mạng không ổn định. Đồng thời, việc dùng txHash làm khoá kiểm soát giúp tránh trùng dữ liệu và tránh cập nhật sai thống kê trong các luồng mua/bán.



Hình 0.3: Ràng buộc unique theo txHash và log index trong bảng nft events

0.4 Tối ưu các API đọc nhiều bằng cache và tổ chức truy vấn phục vụ UI

0.4.1 Giới thiệu

NFT marketplace có nhiều màn hình đọc nhiều: danh sách NFT, chi tiết NFT, trang collection, dashboard thống kê cơ bản, lịch sử giao dịch/giá. Nếu mỗi lần tải trang đều thực hiện truy vấn nặng hoặc tính toán lặp lại, thời gian phản hồi sẽ giảm khi dữ liệu tăng.

0.4.2 Giải pháp

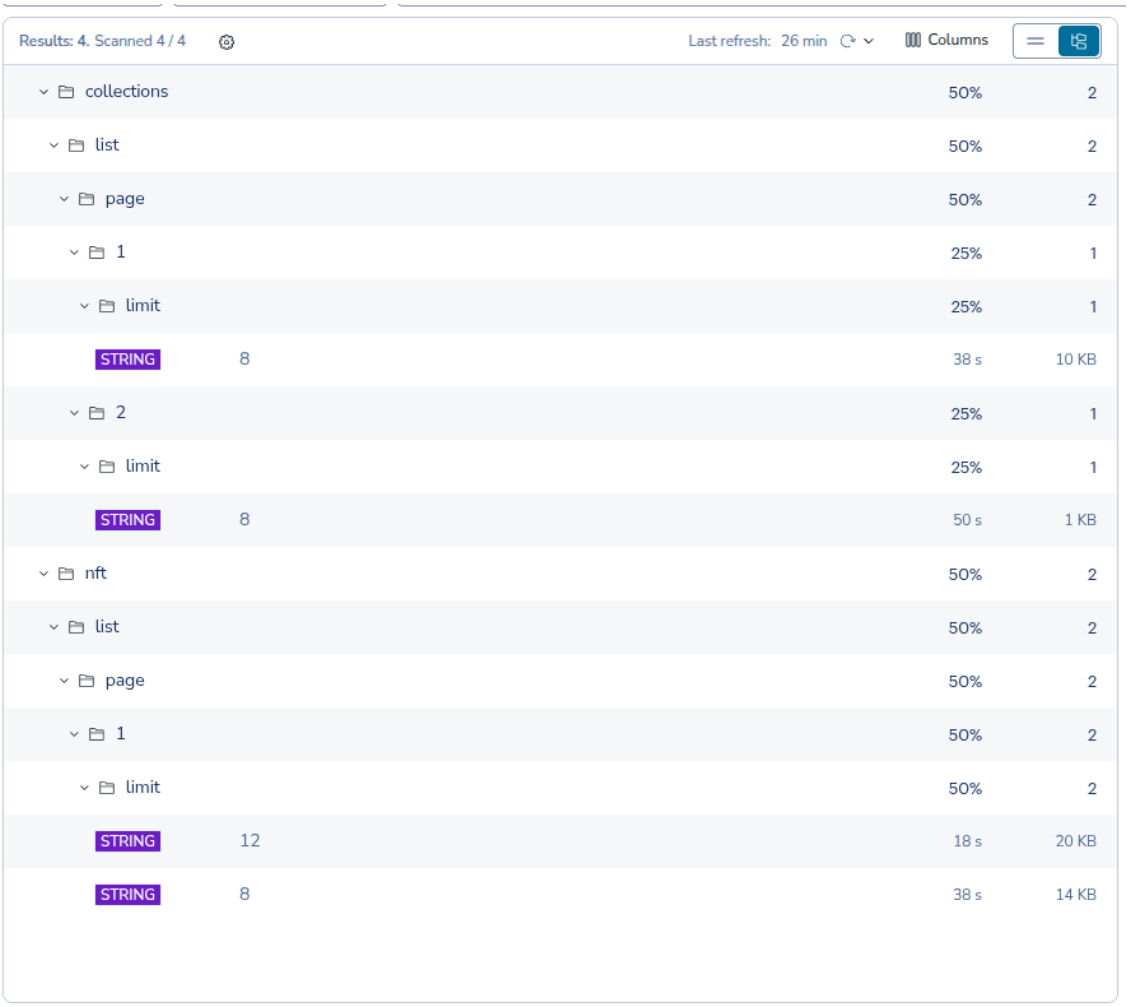
Em tập trung vào hai hướng.

Thứ nhất là cache các dữ liệu đọc nhiều và ít thay đổi trong một khoảng thời gian (thống kê tổng quan, thông tin người dùng, NFT,...). Cache được đặt TTL hợp lý để cân bằng giữa độ "mới" của dữ liệu và hiệu năng.

Thứ hai là tổ chức truy vấn theo đúng nhu cầu hiển thị của UI. Các endpoint danh sách đều có phân trang và chỉ trả về trường dữ liệu cần thiết cho danh sách, tránh trả về toàn bộ dữ liệu như trang chi tiết. Với những dữ liệu có thể phát sinh nhiều bản ghi (lịch sử), em ưu tiên truy vấn theo trang và theo thứ tự thời gian, giúp tải dần và giảm tải cho hệ thống.

0.4.3 Kết quả đạt được

Giải pháp giúp các màn hình đọc chính đạt thời gian phản hồi ổn định hơn trong quá trình demo và kiểm thử. Đồng thời, cache giúp giảm số truy vấn lặp đối với các dữ liệu phổ biến.



Hình 0.4: Cache key cho NFT và Collection trong redis

```
[CACHE] Collection List - MISS → SET - Key: collections:list:page:1:limit:8 - DB: 28ms
 - Total: 30ms - Items: 8
[CACHE] NFT List - MISS → SET - Key: nft:list:page:1:limit:8 - DB: 24ms - Total: 29ms
- Items: 8
[CACHE] NFT List - MISS → SET - Key: nft:list:page:1:limit:8 - DB: 21ms - Total: 25ms
- Items: 8
[CACHE] Collection List - MISS → SET - Key: collections:list:page:1:limit:8 - DB: 25ms
 - Total: 28ms - Items: 8
[CACHE] Collection List - HIT - Key: collections:list:page:1:limit:8 - Time: 2ms
[CACHE] NFT List - HIT - Key: nft:list:page:1:limit:8 - Time: 2ms
[CACHE] Collection List - HIT - Key: collections:list:page:1:limit:8 - Time: 4ms
[CACHE] NFT List - HIT - Key: nft:list:page:1:limit:8 - Time: 3ms
[CACHE] Collection List - HIT - Key: collections:list:page:1:limit:8 - Time: 3ms
[CACHE] NFT List - HIT - Key: nft:list:page:1:limit:8 - Time: 2ms
```

Hình 0.5: Hiệu năng khi cache hit trong redis chỉ mất vài ms

0.5 Chiến lược migration an toàn để giảm rủi ro gián đoạn

0.5.1 Giới thiệu

Trong quá trình phát triển, việc thay đổi yêu cầu kéo theo thay đổi cấu trúc dữ liệu. Nếu thay đổi trực tiếp trên cột đang được sử dụng (ví dụ đổi kiểu dữ liệu), rủi ro phát sinh lỗi hoặc khoá bảng là điều cần cân nhắc, đặc biệt khi hệ thống phát

triển và dữ liệu tăng.

0.5.2 Giải pháp

Em áp dụng chiến lược migration theo hướng tạo cột mới (shadow column): thay vì sửa trực tiếp cột cũ, em tạo cột mới với kiểu dữ liệu/ý nghĩa đúng, thực hiện backfill dữ liệu theo lô, cập nhật ứng dụng để chuyển dần sang đọc/ghi theo cột mới, và chỉ loại bỏ cột cũ khi đã ổn định. Cách tiếp cận này giúp giảm rủi ro và cho phép quay lui dễ hơn nếu phát sinh sai sót.

0.5.3 Kết quả đạt được

Chiến lược này giúp quá trình thay đổi schema an toàn hơn, hạn chế tác động tới các chức năng đang hoạt động, đồng thời làm rõ cách em tiếp cận bài toán theo hướng có kiểm soát rủi ro.

0.6 Kết luận chương

Các đóng góp trong chương tập trung vào những vấn đề em trực tiếp giải quyết trong phạm vi triển khai hiện tại: đồng bộ dữ liệu on-chain/off-chain theo mô hình FE giao dịch và backend xác minh bằng *txHash*; tăng độ tin cậy của cập nhật dữ liệu bằng cơ chế đối chiếu giao dịch; đảm bảo ổn định thông qua idempotency và trạng thái trung gian; tối ưu trải nghiệm bằng chiến lược tối ưu API đọc qua cache; và cuối cùng là cách tiếp cận migration an toàn để hệ thống dễ thay đổi trong giai đoạn phát triển.