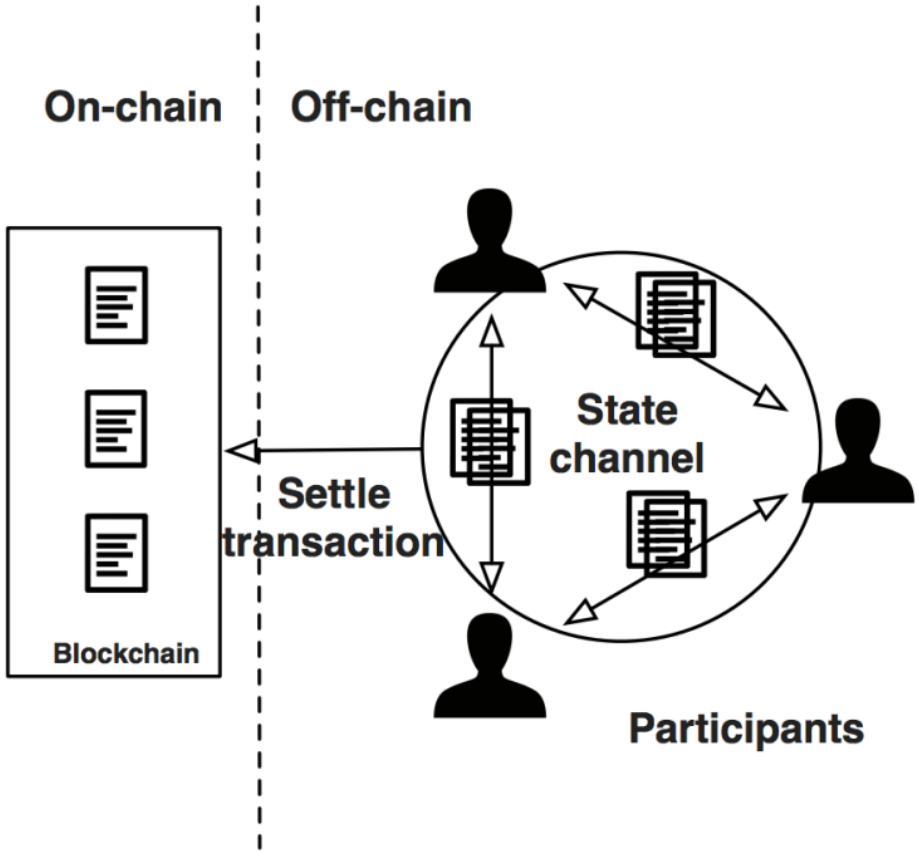
Chương này trình bày nền tảng lý thuyết và các công nghệ được lựa chọn trong đồ án. Khác với ứng dụng web truyền thống, NFT marketplace cần làm việc với một nguồn dữ liệu có tính công khai và có thể kiểm chứng là blockchain. Quyền sở hữu NFT và các giao dịch mua bán không chỉ được ghi nhận trong cơ sở dữ liệu nội bộ, mà phải phản ánh đúng trạng thái trên chuỗi. Vì vậy, việc nắm được các khái niệm như giao dịch, chữ ký số, nonce, phí gas, tiêu chuẩn NFT và cơ chế event/log là cơ sở để giải thích rõ thiết kế hệ thống và lựa chọn công nghệ của đồ án.

0.1 Cơ sở lý thuyết blockchain

0.1.1 Khái niệm giao dịch và trạng thái trên blockchain

Blockchain có thể xem như một sổ cái phân tán, trong đó mọi thay đổi trạng thái được thực hiện thông qua giao dịch và được mạng lưới xác nhận. Đối với các nền tảng tương thích EVM, trạng thái của hệ thống bao gồm trạng thái tài khoản và trạng thái của smart contract (dữ liệu lưu trong vùng storage). Khi người dùng gửi giao dịch tương tác với smart contract, mạng lưới sẽ thực thi logic của hợp đồng để quyết định giao dịch thành công hay thất bại; nếu thành công thì trạng thái được cập nhật, nếu thất bại thì toàn bộ thay đổi bị hoàn tác.



Hình 0.1: Đồng bộ hóa trạng thái trên blockchain

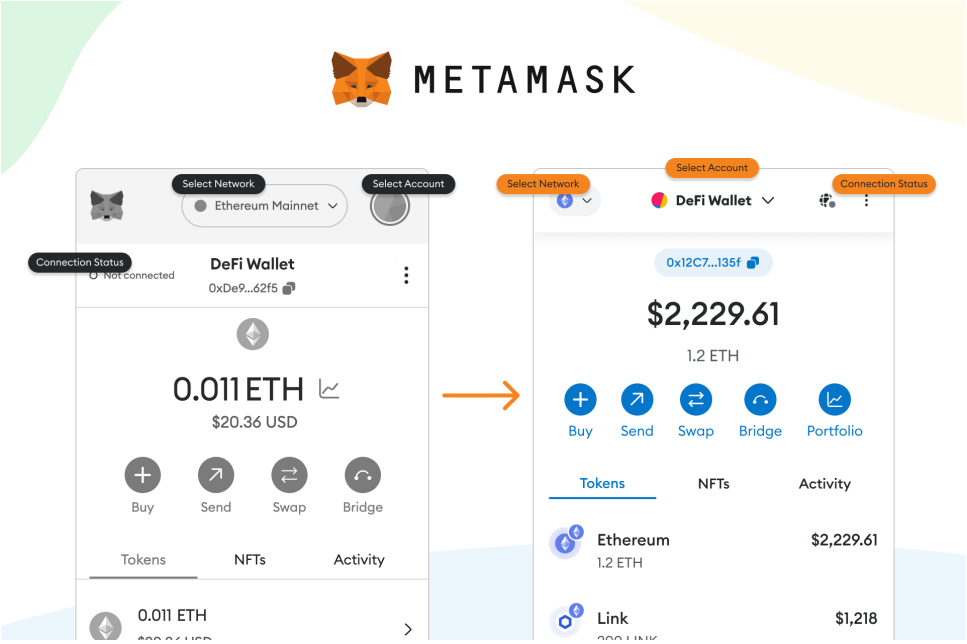
Trong bối cảnh NFT marketplace, các thao tác như mint NFT, niêm yết hoặc

mua bán đều làm thay đổi trạng thái on-chain. Điểm khác biệt quan trọng so với hệ thống web thông thường là quá trình cập nhật không diễn ra tức thời. Giao dịch cần được ký trong ví, phát tán lên mạng, chờ đưa vào block và chờ xác nhận. Do đó, hệ thống cần xem các trạng thái trung gian như đang chờ xác nhận, thành công hoặc thất bại là một phần của luồng nghiệp vụ và phản ánh rõ trên giao diện để người dùng theo dõi.

Bên cạnh độ trễ tự nhiên, blockchain còn mang đặc trưng về khả năng kiểm chứng công khai. Quyền sở hữu và lịch sử chuyển nhượng của NFT có thể đối chiếu qua dữ liệu on-chain, do đó khi hệ thống hiển thị trạng thái sở hữu hay trạng thái giao dịch, dữ liệu cần bám sát thông tin trên chuỗi, hạn chế sai lệch giữa dữ liệu hiển thị off-chain và thực tế on-chain.

0.1.2 Ví điện tử, chữ ký số và nonce

Ví điện tử là công cụ quản lý khóa và thực hiện ký số. Địa chỉ ví đại diện cho danh tính người dùng trên blockchain, còn khóa riêng là yếu tố cốt lõi để chứng minh quyền sở hữu địa chỉ đó. Trong mô hình Web3, ứng dụng không nắm giữ khóa riêng của người dùng; mọi thao tác quan trọng đều cần người dùng xác nhận thông qua ví bằng cách ký thông điệp hoặc ký giao dịch.



Hình 0.2: Ví điện tử MetaMask

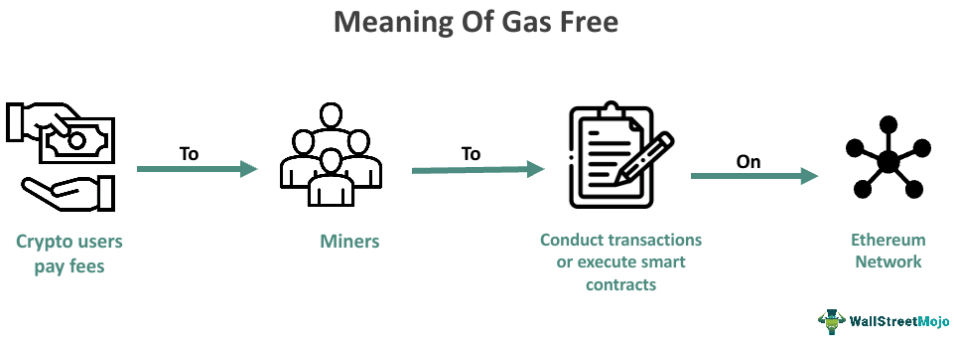
Ký thông điệp thường dùng để xác thực danh tính, ví dụ đăng nhập bằng ví, và không làm thay đổi trạng thái on-chain. Ký giao dịch dùng khi cần thay đổi trạng thái on-chain như mint, list hoặc buy. Tuy nhiên, nếu hệ thống chỉ dựa vào chữ ký thông điệp mà không có cơ chế ràng buộc theo từng phiên, có thể xảy ra rủi ro phát

lại chữ ký, tức là sử dụng lại chữ ký cũ để giả mạo đăng nhập. Để hạn chế rủi ro này, nonce được đưa vào thông điệp ký như một giá trị dùng một lần. Hệ thống kiểm tra chữ ký hợp lệ và đồng thời đảm bảo nonce chưa từng được sử dụng; sau khi xác thực thành công, nonce sẽ bị vô hiệu hóa.

Trong phạm vi đồ án, cơ chế đăng nhập bằng ví dựa trên chữ ký và nonce phù hợp với đặc trưng định danh theo địa chỉ ví, đồng thời tăng mức độ an toàn khi xác thực người dùng.

0.1.3 Gas và phí giao dịch

Trên blockchain, việc thực thi giao dịch tiêu tốn tài nguyên tính toán và lưu trữ, do đó cần có phí giao dịch để ngăn hành vi spam và phân bổ tài nguyên hợp lý. Với các nền tảng tương thích EVM, phí thường gắn với khái niệm gas, được tính dựa trên lượng gas tiêu thụ và mức giá gas tại thời điểm gửi giao dịch. Người dùng phải thanh toán phí cho các thao tác on-chain, và trong nhiều trường hợp vẫn có thể mất phí ngay cả khi giao dịch thất bại do hợp đồng từ chối thực thi.



Hình 0.3: Phí gas trên blockchain

Đối với một marketplace, phí gas ảnh hưởng trực tiếp đến trải nghiệm vì mỗi thao tác on-chain có thể yêu cầu người dùng ký và chờ xác nhận. Nếu một luồng nghiệp vụ đòi hỏi quá nhiều giao dịch, người dùng phải ký nhiều lần và chịu chi phí nhiều lần. Vì vậy, khi thiết kế hệ thống cần cân nhắc rõ thao tác nào bắt buộc phải thực hiện on-chain để đảm bảo ràng buộc tài sản và tính kiểm chứng, và thao tác nào nên đưa về off-chain để giảm chi phí và tăng tốc độ phản hồi.

0.1.4 EVM, smart contract và event/log

EVM là môi trường thực thi tiêu chuẩn cho các blockchain tương thích Ethereum. Smart contract là chương trình chạy trên EVM, có khả năng lưu trữ trạng thái và cung cấp các hàm để người dùng tương tác. Khi một giao dịch gọi vào hợp đồng, EVM thực thi theo logic đã định nghĩa; nếu điều kiện không thỏa thì giao dịch có thể thất bại và toàn bộ thay đổi bị hoàn tác.

Bên cạnh việc cập nhật trạng thái, smart contract có thể phát sinh event/log để ghi nhận các sự kiện quan trọng trong quá trình thực thi. Event không thay thế cho trạng thái, nhưng là nguồn dữ liệu hữu ích để theo dõi các hành động đã xảy ra, chẳng hạn mint, chuyển nhượng hoặc mua bán. Trong các hệ thống Web3, event/log thường được sử dụng để hỗ trợ truy vết giao dịch và làm đầu vào cho quá trình đồng bộ dữ liệu về hệ thống off-chain nhằm phục vụ hiển thị danh sách, lịch sử giao dịch và thống kê.

0.2 Cơ sở lý thuyết NFT và mô hình marketplace

0.2.1 Chuẩn ERC-721 và các thao tác cốt lõi

NFT đại diện cho tài sản số không thể thay thế, trong đó mỗi token có danh tính riêng và có một chủ sở hữu tại một thời điểm. ERC-721 là tiêu chuẩn phổ biến trên các blockchain tương thích EVM, quy định các thao tác cơ bản như truy vấn chủ sở hữu, chuyển nhượng token và cơ chế ủy quyền. Việc tuân theo ERC-721 giúp NFT tương thích với hệ sinh thái sẵn có như ví, thư viện tương tác blockchain và các công cụ theo dõi giao dịch.

Trong phạm vi đồ án, ERC-721 phù hợp với mục tiêu xây dựng marketplace tập trung vào các luồng giao dịch rõ ràng như tạo NFT, niêm yết, mua bán và theo dõi lịch sử sở hữu.

0.2.2 Metadata NFT và tokenURI

Bên cạnh quyền sở hữu, NFT cần metadata để mô tả nội dung hiển thị như tên, mô tả, hình ảnh và thuộc tính. Trong ERC-721, tokenURI thường được dùng để trỏ đến một tài nguyên chứa metadata, phổ biến là một tệp JSON. Metadata thường được lưu ngoài chuỗi vì lưu trực tiếp on-chain có chi phí cao và khó thay đổi.

Từ góc nhìn hệ thống, điều này dẫn đến yêu cầu cần có cơ chế lưu trữ nội dung số ổn định và thuận tiện cho việc tải lên, đồng thời cần quản lý dữ liệu nghiệp vụ phục vụ hiển thị và tra cứu. Trong đồ án, ảnh và tệp nội dung được lưu trên hệ thống lưu trữ đối tượng, còn các thông tin nghiệp vụ và dữ liệu phục vụ thống kê được lưu trong cơ sở dữ liệu off-chain để tối ưu truy vấn và trải nghiệm người dùng.

0.2.3 Cơ chế approve/transfer và ý nghĩa trong giao dịch

Để hoàn tất giao dịch mua bán, hợp đồng marketplace cần chuyển NFT từ người bán sang người mua theo điều kiện hợp lệ. Tuy nhiên, smart contract không thể tự ý chuyển tài sản của người dùng nếu không được ủy quyền. Vì vậy, cơ chế approve được sử dụng để chủ sở hữu cấp quyền cho một địa chỉ hoặc một hợp đồng được thay mặt mình thực hiện chuyển nhượng token khi cần.

Trong thực tế, một số luồng nghiệp vụ sẽ bao gồm bước ủy quyền trước khi niêm

yết hoặc trước khi hợp đồng thực hiện chuyển nhượng. Do đó, hệ thống cần trình bày rõ ràng ý nghĩa của bước ủy quyền và trạng thái giao dịch, giúp người dùng, đặc biệt là người mới, hiểu được tiến trình thao tác và hạn chế nhầm lẫn giữa việc cấp quyền và việc chuyển tài sản.

0.2.4 Vai trò event trong đồng bộ on-chain/off-chain

Một NFT marketplace thường có nhiều chức năng đọc dữ liệu như danh sách NFT, danh sách collection, tìm kiếm, lọc, lịch sử giá và thống kê. Nếu hệ thống truy vấn trực tiếp blockchain cho mọi thao tác đọc, tốc độ phản hồi sẽ chậm và khó mở rộng. Vì vậy, cách tiếp cận phổ biến là kết hợp dữ liệu on-chain và off-chain: blockchain đảm bảo quyền sở hữu và giao dịch, còn cơ sở dữ liệu off-chain phục vụ các truy vấn hiển thị và thống kê.

Event/log là cơ chế quan trọng để đồng bộ. Khi smart contract phát sinh các sự kiện như mint, niêm yết hoặc mua bán, backend có thể ghi nhận các sự kiện đó và cập nhật vào cơ sở dữ liệu. Nhờ vậy, hệ thống vừa đảm bảo dữ liệu hiển thị bám sát hoạt động on-chain, vừa đạt được hiệu năng phù hợp cho trải nghiệm người dùng.

0.3 Công nghệ sử dụng trong đồ án

Phần này trình bày các công nghệ được lựa chọn để hiện thực hệ thống NFT marketplace theo từng lớp. Định hướng lựa chọn công nghệ của đồ án là thống nhất stack, giảm rủi ro tích hợp, triển khai được đầy đủ các luồng nghiệp vụ cốt lõi và đảm bảo hệ thống vận hành ổn định ở mức demo.

0.3.1 TypeScript trong phát triển ứng dụng web

TypeScript là ngôn ngữ mở rộng từ JavaScript, bổ sung hệ kiểu tĩnh và các cơ chế hỗ trợ xây dựng phần mềm quy mô vừa và lớn như interface, generics, decorator và kiểm tra kiểu ở thời điểm biên dịch. Trên thực tế, nhiều hệ thống web chọn TypeScript không phải vì hiệu năng cú pháp, mà vì cần kiểm soát rủi ro: dữ liệu qua API, dữ liệu UI state, và dữ liệu nghiệp vụ thường biến đổi liên tục trong quá trình phát triển; nếu chỉ dựa vào JavaScript thuần, lỗi kiểu dữ liệu thường xuất hiện và khó truy vết.

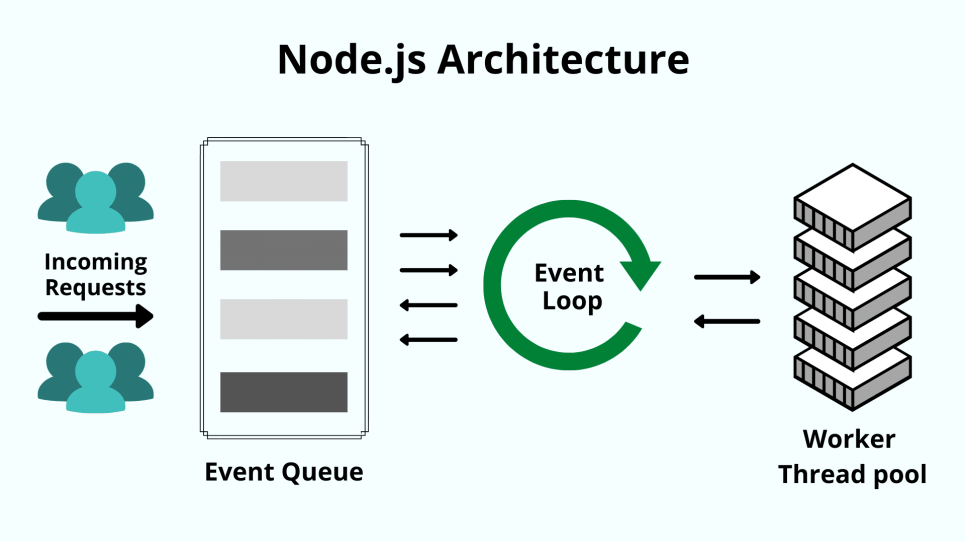
Hình 0.4: Công nghệ TypeScript

Trong đồ án, lượng mô hình dữ liệu không hề nhỏ. TypeScript giúp mô tả rõ cấu trúc request/response của API, giảm sai lệch giữa frontend và backend, đồng thời tăng độ an toàn khi xử lý các trường dễ nhầm như địa chỉ ví, chain id, giá, phí và trạng thái giao dịch. Đặc biệt với UI hiển thị lịch sử giá và thống kê, dữ liệu thường là mảng thời gian và phép tổng hợp; việc có kiểu dữ liệu rõ ràng giúp tránh lỗi không mong muốn và giúp quá trình phát triển thống nhất, đồng bộ hơn.

JavaScript thuần là một lựa chọn thay thế hợp lệ, nhất là khi muốn viết nhanh giai đoạn đầu. Tuy nhiên, trong bối cảnh dự án cần sự ổn định, TypeScript làm cho code dễ đọc hơn khi nhiều file cùng tham gia một luồng nghiệp vụ. Vì vậy, TypeScript được chọn làm ngôn ngữ thống nhất cho cả frontend và backend.

0.3.2 Node.js và mô hình I/O bất đồng bộ

Node.js là môi trường chạy JavaScript phía server dựa trên event loop, tối ưu cho các ứng dụng có nhiều tác vụ I/O như gọi cơ sở dữ liệu, gọi API, đọc/ghi file, hoặc giao tiếp mạng. Thế mạnh của Node.js không nằm ở xử lý tính toán nặng, mà ở khả năng xử lý đồng thời nhiều kết nối và nhiều request trong khi phần lớn thời gian là chờ I/O.



Hình 0.5: Công nghệ Node.js

Đồ án có đặc điểm tương đối phù hợp với Node.js: số lượng API đọc nhiều (danh sách, chi tiết, search, lịch sử giao dịch, dashboard), còn các tác vụ ghi chủ yếu là cập nhật dữ liệu off-chain sau các hành động của người dùng. Ngoài ra, hệ sinh thái Web3 hỗ trợ Node.js rất tốt; việc tích hợp RPC, ký message, đọc event, và tương tác EVM trở nên thuận lợi nhờ các thư viện phổ biến (chẳng hạn ethers.js). Đây là lý do thực tiễn khiến Node.js phù hợp hơn trong đồ án so với việc dùng công nghệ khác.

Tất nhiên, backend có thể được viết bằng Java (Spring Boot) hoặc .NET; những nền tảng này rất mạnh trong doanh nghiệp và có hệ sinh thái lớn. Tuy nhiên, chọn chúng trong đồ án sẽ làm công nghệ sử dụng trở nên không đồng nhất, tăng thời gian tích hợp, trong khi mục tiêu là hoàn thiện một marketplace có luồng nghiệp vụ rõ ràng. Vì vậy, TypeScript/Node.js là lựa chọn hợp lý về cả kỹ thuật lẫn tiến độ.

0.3.3 React và Next.js cho giao diện người dùng

React là thư viện xây dựng giao diện theo hướng thành phần (component), cho phép chia UI thành các khối độc lập, tái sử dụng và quản lý trạng thái theo cách có cấu trúc. Next.js là framework dựa trên React, cung cấp cơ chế routing theo file, tối ưu build, và hỗ trợ nhiều chế độ render (SSR/SSG/CSR) để cân bằng giữa tốc độ tải trang và tính tương tác. Trong nhiều dự án web hiện đại, Next.js được chọn vì nó cung cấp sẵn các tính năng sẵn có: cách tổ chức trang, cách build, cách tối ưu tài nguyên, và cách mở rộng.

Định hướng UI/UX ở Chương 2 yêu cầu giao diện đơn giản hơn các marketplace lớn. Với Next.js/React, giao diện dễ được tổ chức theo component. Việc tách UI theo component giúp duy trì sự nhất quán giữa các trang, ví dụ danh sách và chi tiết có cùng cách trình bày trạng thái của NFT, cùng kiểu hiển thị giá và phí, và cùng chuẩn thông báo lỗi.

Nếu dùng React SPA thuần (chẳng hạn Vite), dự án vẫn có thể hoàn thành, nhưng phải tự quyết nhiều thứ hơn về routing, tổ chức trang và tối ưu build. Angular là một lựa chọn khác, cung cấp framework đầy đủ với DI, routing, form; tuy nhiên Angular thường nặng hơn và không phổ biến bằng React trong hệ sinh thái Web3. Vue/Nuxt cũng là phương án đáng cân nhắc, nhưng trong phạm vi đồ án, Next.js có lợi thế về mức độ phổ biến và tài liệu, giúp giảm rủi ro khi triển khai và khi viết báo cáo tham khảo.

Hình 0.6: Công nghệ Next.js

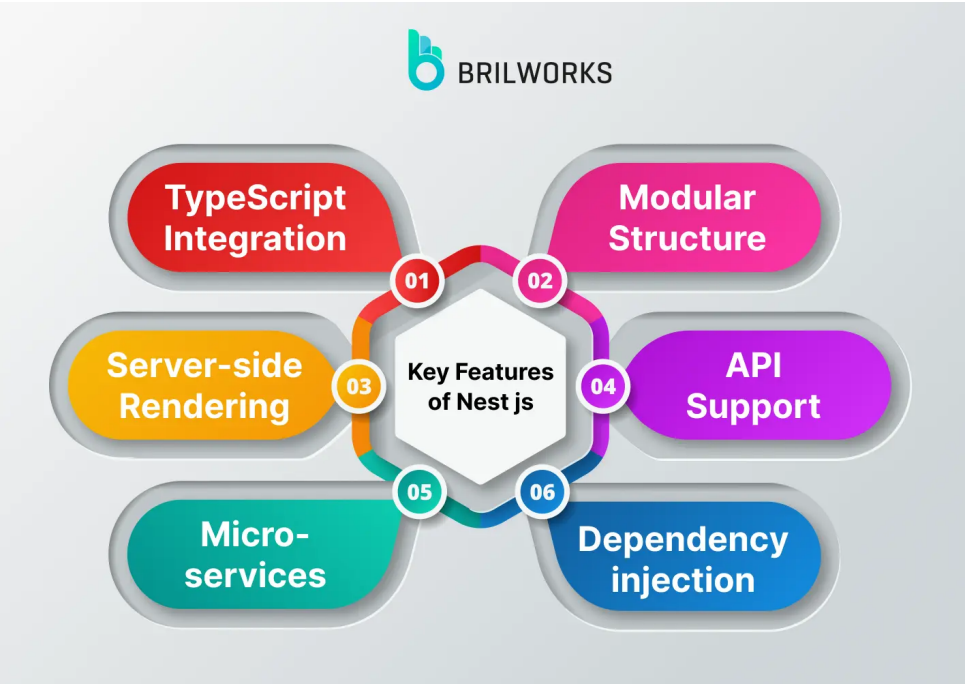
0.3.4 Tương tác ví và quản lý trạng thái giao dịch trên giao diện

Một trang web thông thường chỉ cần tương tác với trang web chính, còn Web3 thì khác: kết nối ví, ký message để đăng nhập, ký giao dịch để mint, ký giao dịch để list/unlist, rồi chờ mạng xác nhận. Nếu UI/UX không đươc thiết kế tốt, người dùng mới sẽ cảm thấy khó sử dụng và không hiểu luồng nó hoạt động như thế nào. Vì vậy, ngoài việc gọi RPC, giao diện phải thể hiện rõ trạng thái: đang yêu cầu ký, đang chờ xác nhận, đã thành công, hay thất bại và cần thử lại. Đây là phần trực tiếp đáp ứng yêu cầu trải nghiệm ở Chương 2: thao tác quan trọng phải có xác nhận rõ ràng và thông báo minh bạch.

Về mặt kỹ thuật, các thao tác này thường được thực hiện thông qua provider của ví và thư viện tương tác EVM (ví dụ ethers.js). Những thư viện này không tự làm UI; chúng chỉ giúp gọi giao dịch đúng cách. Vì vậy, lựa chọn Next.js/React không chỉ để phát triển giao diện, mà còn để tổ chức các trạng thái UI quanh những điểm cốt lõi để người dùng hiểu rõ hơn: phí gas, thời gian chờ, và các trường hợp người dùng từ chối ký.

0.3.5 NestJS và tổ chức backend theo module

NestJS là framework Node.js hướng kiến trúc, khuyến khích tổ chức dự án theo module và tách lớp rõ ràng giữa controller, service và data access, đồng thời cung cấp dependency injection để quản lý phụ thuộc. Trong thực tế, NestJS thường được chọn khi đội phát triển muốn tổ chức mã nguồn theo hướng nghiệp vụ dễ mở rộng: thay vì một tập hợp route handler rời rạc, hệ thống được tổ chức theo miền nghiệp vụ và có chuẩn chung cho validation, error handling, auth guard và logging.



Hình 0.7: Công nghệ NestJS

Trong đồ án, backend đóng vai trò trung tâm để xử lý nghiệp vụ và điều phối dữ liệu giữa các thành phần của hệ thống. Cụ thể, backend quản lý dữ liệu off-chain trong PostgreSQL, sử dụng Redis để giảm tải cho các truy vấn đọc nhiều, làm việc với S3 để lưu trữ hình ảnh, đồng thời theo dõi và đồng bộ các thay đổi quan trọng từ smart contract lên cơ sở dữ liệu nhằm phục vụ hiển thị và thống kê.

Các luồng nghiệp vụ chính như tạo NFT, niêm yết hay mua bán đều cần backend phối hợp nhiều bước liên tiếp: vừa đảm bảo dữ liệu hiển thị đầy đủ, vừa đảm bảo trạng thái phản ánh đúng kết quả giao dịch on-chain. Vì vậy, nếu mã nguồn không được tổ chức tốt, các phần xử lý dễ bị làm theo cách phức tạp và khó bảo trì khi mở rộng chức năng hoặc xử lý lỗi. NestJS được lựa chọn nhằm hỗ trợ cấu trúc dự án theo hướng module hoá, giúp tách bạch rõ các phần xử lý và phù hợp với yêu cầu bảo trì, mở rộng.

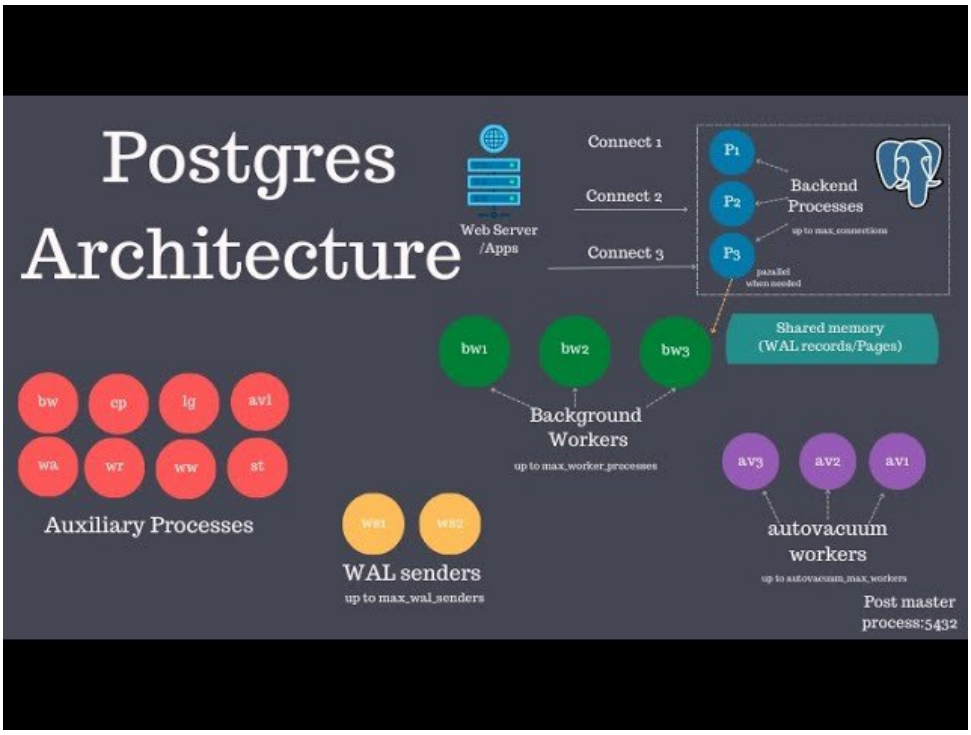
Express hoặc Fastify là lựa chọn thay thế phổ biến, nhẹ và linh hoạt. Tuy nhiên, dùng nền tảng thuần đồng nghĩa nhóm phải tự đặt chuẩn kiến trúc, tự tổ chức module, tự thống nhất cách trả lỗi và validation; gây ra tốn thời gian và không nhất quán khi dự án lớn dần. Spring Boot cũng là một đối thủ mạnh; nhưng với phạm vi của đồ án và nhu cầu tích hợp Web3 nhanh, NestJS giúp giảm thời gian phát triển và giữ công nghệ sử dụng được thống nhất.

0.3.6 PostgreSQL cho dữ liệu nghiệp vụ off-chain

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được phát triển lâu đời và được sử dụng rộng rãi trong các hệ thống web hiện đại. Điểm mạnh

của PostgreSQL nằm ở khả năng đảm bảo tính đúng đắn của dữ liệu thông qua giao dịch và các ràng buộc toàn vẹn, đồng thời hỗ trợ SQL tương đối đầy đủ và linh hoạt. Hệ thống cho phép mô hình hoá dữ liệu theo dạng bảng với khoá chính, khoá ngoại, ràng buộc duy nhất và các quy tắc kiểm tra, nhờ đó giảm đáng kể rủi ro sinh dữ liệu sai lệch khi nhiều chức năng cùng đọc/ghi. PostgreSQL cũng nổi bật ở khả năng xử lý đồng thời tốt nhờ cơ chế MVCC, nghĩa là việc đọc và ghi có thể diễn ra song song mà hạn chế tình trạng "chặn" nhau như các cơ chế khoá truyền thống, phù hợp với các ứng dụng có nhiều lượt truy cập.

Về hiệu năng, PostgreSQL cung cấp hệ thống chỉ mục phong phú và nhiều kỹ thuật tối ưu truy vấn. Ngoài chỉ mục B-tree phổ biến cho các truy vấn so sánh và sắp xếp, PostgreSQL còn hỗ trợ các loại chỉ mục khác như GIN/GiST, hữu ích trong những trường hợp tìm kiếm theo tập hợp, dữ liệu bán cấu trúc hoặc truy vấn theo điều kiện phức tạp. Bên cạnh mô hình quan hệ, PostgreSQL cũng hỗ trợ kiểu dữ liệu JSON/JSONB, giúp lưu trữ một số trường dữ liệu linh hoạt khi cần mà vẫn có thể truy vấn và lập chỉ mục ở mức nhất định. Đây là lý do PostgreSQL thường được lựa chọn trong các dự án cần vừa chặt chẽ về quan hệ dữ liệu, vừa có một phần dữ liệu có thể thay đổi theo thời gian.



Hình 0.8: Công nghệ PostgreSQL

Trong đồ án, PostgreSQL được sử dụng làm kho dữ liệu chính cho phần off-chain, phục vụ lưu trữ thông tin nghiệp vụ và cung cấp dữ liệu cho các màn hình hiển thị. Nhờ cơ chế truy vấn và tổng hợp tốt, hệ thống có thể trích xuất lịch sử giao

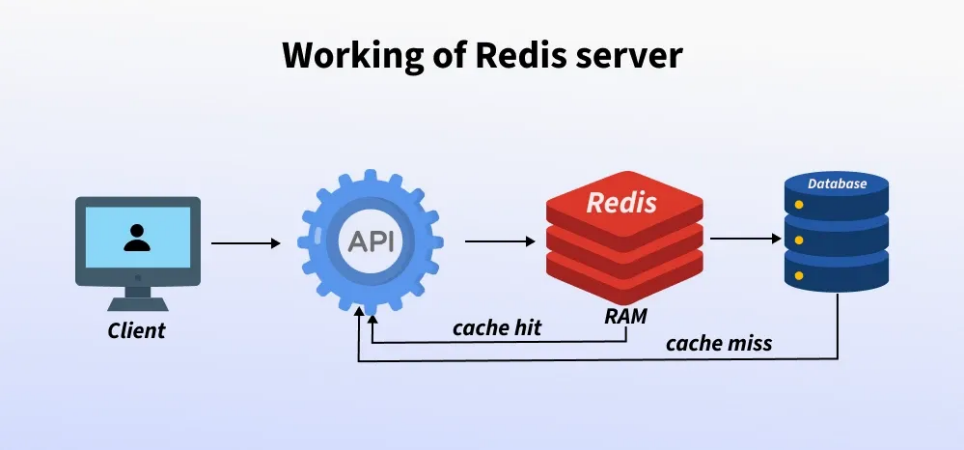
dịch theo thời gian để xây dựng lịch sử giá và các thống kê dashboard. Ngoài ra, việc lưu metadata như tên và mô tả ở cơ sở dữ liệu giúp thao tác cập nhật nội dung diễn ra thuận tiện hơn, trong khi các phần ràng buộc sở hữu và trạng thái giao dịch vẫn được đảm bảo bởi smart contract.

MySQL là phương án thay thế trực tiếp và cũng rất phổ biến trong các ứng dụng web. Với các thao tác CRUD cơ bản, cả MySQL và PostgreSQL đều đáp ứng tốt. Tuy nhiên, trong phạm vi đồ án, PostgreSQL được ưu tiên vì thuận lợi hơn khi xây dựng các truy vấn tổng hợp và thống kê, đặc biệt ở các chức năng như dashboard và lịch sử giá theo thời gian.

Một lựa chọn khác là MongoDB, phù hợp khi dữ liệu cần lưu dưới dạng document và cấu trúc thay đổi linh hoạt. Dù vậy, bài toán marketplace thường có nhiều quan hệ rõ ràng giữa các thực thể và nhu cầu tổng hợp dữ liệu theo nhiều chiều xuất hiện thường xuyên, nên mô hình quan hệ giúp quản lý ràng buộc chặt chẽ hơn và việc truy vấn theo nghiệp vụ cũng dễ kiểm soát hơn.

0.3.7 Redis cho caching và dữ liệu ngắn hạn

Redis là một hệ thống lưu trữ dữ liệu dạng key–value hoạt động chủ yếu trên bộ nhớ, vì vậy tốc độ đọc/ghi thường rất nhanh. Redis được dùng phổ biến trong vai trò cache cho ứng dụng web nhờ cơ chế lưu trữ đơn giản, hỗ trợ đặt thời gian hết hạn cho dữ liệu và có thể giảm đáng kể số lần truy vấn về cơ sở dữ liệu chính. Ngoài cache, Redis cũng hay được tận dụng cho các dữ liệu "ngắn hạn" như nonce phục vụ xác thực, giới hạn tần suất truy cập hoặc các trạng thái tạm trong một khoảng thời gian nhất định, tuỳ theo cách thiết kế hệ thống.



Hình 0.9: Redis cho caching

Trong đồ án, Redis được sử dụng nhằm hỗ trợ yêu cầu hiệu năng đã nêu ở Chương 2, đặc biệt với các luồng đọc như danh sách NFT/collection, trang chi tiết và các số liệu thống kê. Đặc trưng của marketplace là số lượt đọc thường lớn hơn

nhiều so với số lượt ghi; nếu mỗi lần tải trang đều phải truy vấn và tổng hợp lại toàn bộ từ PostgreSQL thì độ trễ sẽ tăng và hệ thống dễ bị quá tải khi lượng truy cập lớn. Khi có Redis, một số dữ liệu được truy cập lặp lại có thể được lưu tạm trong cache và phục vụ nhanh hơn; dữ liệu sẽ được làm mới theo thời gian hết hạn hoặc theo cơ chế xoá cache khi có cập nhật. Nhờ đó, thời gian phản hồi ổn định hơn và trải nghiệm người dùng cũng mượt hơn.

Về hướng tiếp cận thay thế, hệ thống có thể cache trực tiếp trong bộ nhớ của ứng dụng. Cách này đơn giản nếu chỉ chạy một instance, nhưng khi cần mở rộng nhiều instance thì cache bị phân tán và dễ không nhất quán. Memcached cũng là một lựa chọn phổ biến cho cache thuần, tuy nhiên Redis thường linh hoạt hơn và được hỗ trợ tốt trong hệ sinh thái Node.js, đồng thời tài liệu và cộng đồng cũng đông đảo giúp cho phần sửa lỗi trong quá trình phát triển dễ dàng hơn.

0.3.8 Amazon S3 và cơ chế presigned URL cho lưu trữ nội dung số

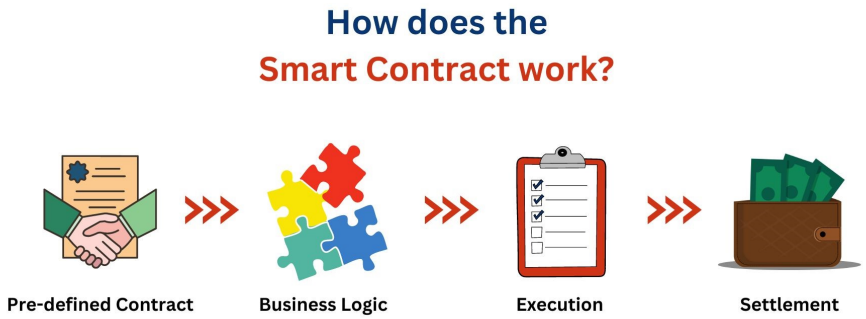
Amazon S3 là dịch vụ lưu trữ đối tượng của AWS, phù hợp để lưu trữ các nội dung dạng file như hình ảnh. Trong đồ án, ảnh của NFT và collection được lưu trên S3 nhằm tách phần lưu trữ nội dung số khỏi cơ sở dữ liệu nghiệp vụ, giảm dung lượng và chi phí vận hành ở tầng dữ liệu quan hệ.

Hình 0.10: Công nghệ Amazon S3

Quy trình upload được triển khai bằng presigned URL: backend cấp một đường dẫn tải lên có thời hạn, frontend sử dụng đường dẫn này để upload trực tiếp lên S3, sau đó hệ thống lưu lại thông tin file trong cơ sở dữ liệu. Cách làm này giúp giảm tải cho backend và hạn chế việc client phải nắm giữ thông tin truy cập dịch vụ lưu trữ.

0.3.9 Solidity, smart contract và môi trường triển khai

Smart contract được viết bằng Solidity và triển khai trên nền tảng tương thích EVM. Đồ án sử dụng chuẩn ERC-721 để đảm bảo tính tương thích với hệ sinh thái và thuận tiện cho quá trình tích hợp. Các thao tác ràng buộc tài sản và giao dịch được thực hiện trên smart contract để đảm bảo tính kiểm chứng, trong khi dữ liệu phục vụ hiển thị và thống kê được xử lý off-chain để tối ưu hiệu năng.

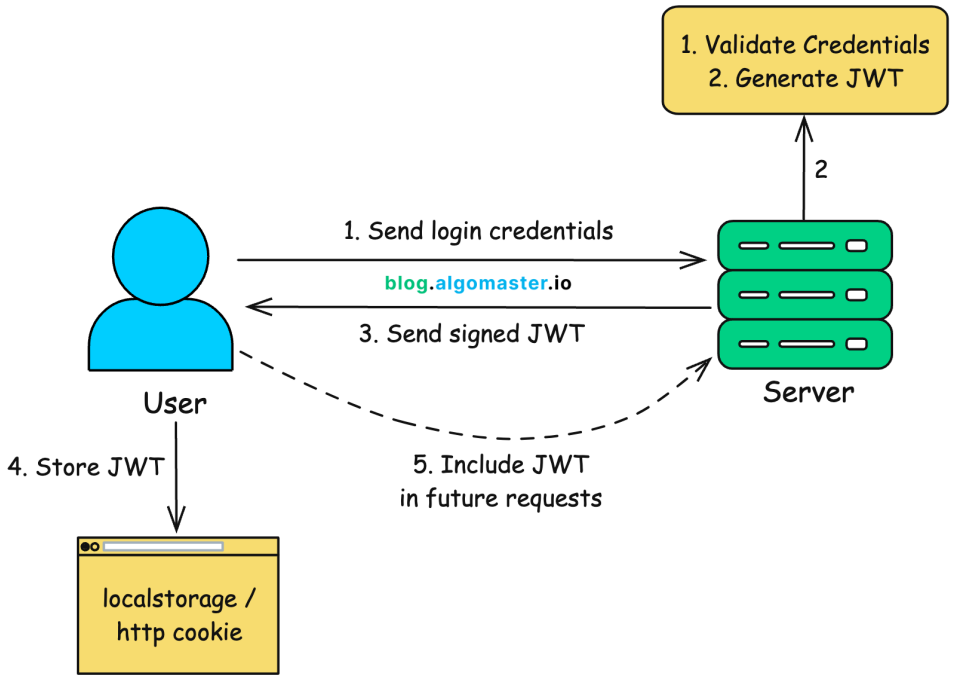


Hình 0.11: Công nghệ smart contract

Trong phạm vi đồ án, môi trường testnet được sử dụng cho mục tiêu phát triển, kiểm thử và trình diễn, đồng thời giúp giảm chi phí trong quá trình triển khai.

0.3.10 SIWE và JWT cho xác thực và bảo vệ API

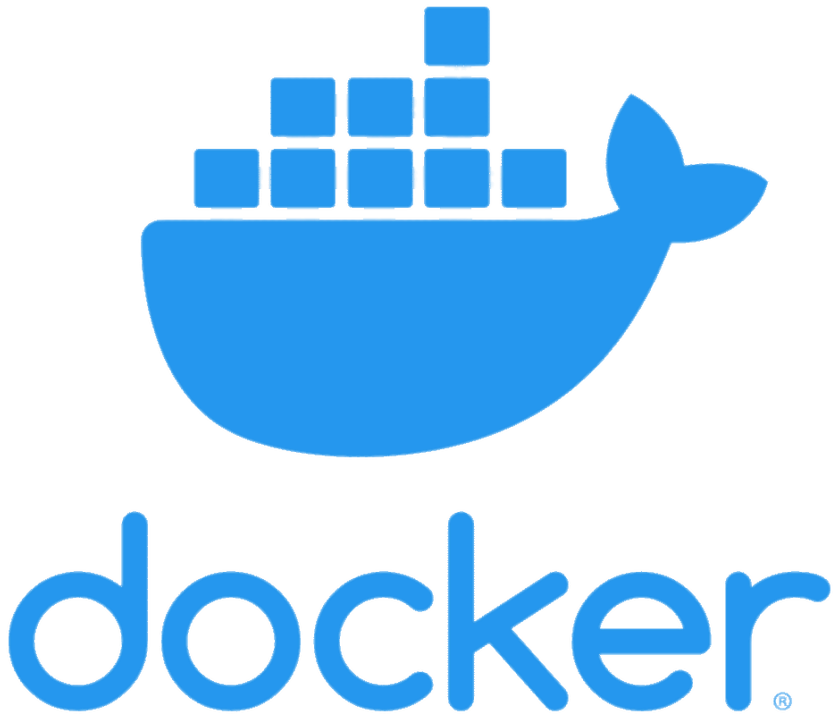
Hệ thống áp dụng đăng nhập bằng ví dựa trên chữ ký số và nonce, phù hợp với mô hình định danh theo địa chỉ ví trong Web3. Sau khi xác thực thành công, backend phát hành JWT để frontend sử dụng khi gọi các API cần quyền truy cập. Cách kết hợp này giúp giảm số lần người dùng phải ký trong các thao tác off-chain như cập nhật hồ sơ hoặc xem dashboard, đồng thời vẫn giữ nguyên yêu cầu ký giao dịch trong ví đối với các thao tác on-chain như mint, niêm yết và mua bán.



Hình 0.12: Công nghệ JWT

0.3.11 Docker và định hướng triển khai

Docker được sử dụng để đóng gói ứng dụng và tái lập môi trường chạy nhất quán giữa các máy phát triển và môi trường demo. Trong phạm vi đồ án, nội dung triển khai được trình bày theo định hướng tách các thành phần chính của hệ thống và cấu hình thông qua biến môi trường, nhằm thuận tiện vận hành và mở rộng về sau.



Hình 0.13: Công nghệ Docker

0.4 Kết luận chương

Chương này đã trình bày nền tảng lý thuyết cần thiết để hiểu mô hình hoạt động của một NFT marketplace, bao gồm giao dịch và trạng thái trên blockchain, cơ chế ký số qua ví, phí gas, EVM/smart contract và vai trò event/log. Trên cơ sở đó, chương mô tả các công nghệ được lựa chọn để hiện thực hệ thống theo hướng thống nhất, dễ triển khai và bám sát yêu cầu ở Chương 2. Blockchain đảm bảo tính kiểm chứng cho quyền sở hữu và giao dịch, trong khi dữ liệu off-chain và các công nghệ web giúp hệ thống đạt được hiệu năng và trải nghiệm phù hợp cho ứng dụng marketplace.