

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO THỰC NGHIỆM
HỌC PHẦN MỘT SỐ CÔNG NGHỆ PHÁT TRIỂN PHẦN MỀM
Đề tài
NGHIÊN CỨU PHÁT TRIỂN ỨNG DỤNG NFT VỚI
BLOCKCHAIN

GVHD: TS. Hà Mạnh Đào

Nhóm - Lớp Nhóm 09 – 20241IT6024001

Thành viên: Trần Văn Đức - 2021601381

Phan Thế Hoàng - 2021600173

Nguyễn Văn Mạnh - 2021600169

Hà Nội, 2024

LỜI CẢM ƠN

Chúng em là nhóm sinh viên thực hiện bài tập lớn của học phần Một số Công nghệ phát triển phần mềm, xin gửi lời cảm ơn chân thành đến Trường Đại học Công nghiệp Hà Nội và Thầy Hà Mạnh Đào.

Trước hết, chúng em xin gửi lời cảm ơn đến Trường Đại học Công nghiệp Hà Nội đã cung cấp cho chúng em môi trường học tập và nghiên cứu tốt nhất. Những cơ sở vật chất và dịch vụ hỗ trợ tại trường đã giúp chúng em tiếp cận được những tài nguyên quan trọng và hoàn thành bài tập lớn một cách hiệu quả.

Chúng em cũng muốn gửi lời cảm ơn đến Thầy Hà Mạnh Đào, giảng viên của chúng em, đã cung cấp cho chúng em sự hỗ trợ và giúp đỡ trong quá trình thực hiện bài tập lớn. Những lời khuyên, định hướng và sự hỗ trợ của thầy đã giúp chúng em hoàn thành nhiệm vụ của mình một cách tốt nhất.

Học phần Một số Công nghệ phát triển phần mềm là một học phần thú vị, vô cùng bổ ích, cung cấp đầy đủ kiến thức thực tế, giúp chúng em nâng cao được kỹ năng làm việc nhóm và là nền tảng vững chắc cho kỳ thực tập sắp tới của chúng em. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài báo cáo khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong thầy xem xét và góp ý để bài báo cáo của chúng em được hoàn thiện hơn.

Nhóm 9 xin chân thành cảm ơn.

MỤC LỤC

LỜI CẢM ƠN.....	1
MỤC LỤC.....	2
PHỤ LỤC HÌNH ẢNH.....	5
LỜI MỞ ĐẦU.....	9
1. Lý do chọn đề tài.....	9
2. Mục tiêu.....	9
3. Bố cục chính.....	9
CHƯƠNG 1 TÌM HIỂU, XÂY DỰNG ỦNG DỤNG MICROSERVICES QUẢN LÝ NGƯỜI DÙNG.....	10
1.1. Phương pháp thiết kế DDD.....	10
1.1.1 Tổng quan DDD.....	10
1.1.2 Các khái niệm liên quan.....	11
1.2. Triển khai dự án theo link được giao.....	12
1.2.1. Phân tích cấu trúc chương trình.....	12
1.2.2. Cài Đặt chương trình.....	16
1.2.2.1. Hướng dẫn cài đặt.....	16
1.2.2.2. Chi tiết cụ thể từng service.....	23
1.2.2.3. Kết quả chạy chương trình.....	31
1.2.2.4. Trình bày chi ra thiết kế DDD chương trình.....	40
1.3. Triển khai với Docker Swarm hoặc Kubernetes hoặc Docker với Kubernetes.....	43
1.3.1. Lý thuyết, kiến thức về Docker.....	43
1.3.1.1 Khái niệm.....	43
1.3.1.2 Các thành phần cơ bản của Docker.....	43
1.3.1.3 Cơ chế hoạt động.....	44
1.3.2. Hướng dẫn đóng gói dự án bằng docker.....	44
1.3.2.1 Đóng gói PostgreSQL.....	44
1.3.2.2 Đóng gói ZooKeeper.....	46
1.3.2.3 Đóng gói Kafka.....	46
1.3.2.4 Đóng gói Kafka-ui.....	48
CHƯƠNG 2 PHÁT HIỆN ẢNH ĐÃ QUA CHỈNH SỬA ỦNG DỤNG CNN, ELA VÀ KERAS.....	50
2.1. Tổng quan bài toán.....	50
2.1.1 Giới thiệu bài toán.....	50
2.1.2. Cơ sở lý thuyết.....	51
2.1.2.1 Trí tuệ nhân tạo và học máy.....	51

2.1.2.2. Mạng nơ-ron tích chập CNN.....	53
2.1.2.3 Kỹ thuật phân tích mức độ lỗi (ELA).....	57
2.2. Phân tích cấu trúc bài toán.....	60
2.3. Cài đặt và chạy chương trình.....	64
2.3.1. Các công cụ và thư viện cần thiết.....	64
2.3.2. Thu thập dữ liệu.....	66
2.3.2. Xây dựng và huấn luyện mô hình.....	71
2.3.2.1. Tiết xử lý dữ liệu.....	71
2.3.2.2. Huấn luyện mô hình CNN.....	73
2.3.3. Chạy chương trình dự đoán.....	76
2.4. Phân tích kết quả chạy chương trình.....	80
2.4.1. Độ chính xác của mô hình.....	80
2.4.2. Hiệu quả của phương pháp ELA.....	83
2.4.3. Ứng dụng thực tế.....	84
2.4.3.1. Độ chính xác trên tập dữ liệu phức tạp.....	84
2.4.3.2. Khả năng mở rộng.....	85
2.4.3.3. Tương thích với thực tế.....	85
2.4.3.4. Ứng dụng trong các trường hợp chỉnh sửa phức tạp.....	86
2.4.3.5. Đề xuất cải thiện để áp dụng thực tế.....	86
CHƯƠNG 3 NGHIÊN CỨU PHÁT TRIỂN ỨNG DỤNG NFT VỚI BLOCKCHAIN.....	87
3.1. Tổng quan Blockchain.....	87
3.1.1. Công nghệ Blockchain là gì?.....	87
3.1.1.1. Khái niệm công nghệ Blockchain?.....	87
3.1.1.2. Ứng dụng công nghệ dựa trên Blockchain.....	88
3.1.2. Tại sao blockchain lại quan trọng?.....	89
3.1.3. Công nghệ Blockchain có những đặc điểm gì?.....	90
3.1.4. Chuỗi khối hoạt động như thế nào?.....	91
3.1.5. Có những loại mạng lưới blockchain nào?.....	92
3.1.6. Các giao thức Blockchain là gì?.....	93
3.1.6.1. Khái niệm về các giao thức Blockchain.....	93
3.1.6.2. Các loại giao thức Blockchain chính.....	94
3.1.7. Công nghệ blockchain đã phát triển như thế nào?.....	94
3.1.8. Lợi ích.....	95
3.1.9. Sự khác biệt giữa Bitcoin và chuỗi khối là gì?.....	96
3.2. Nghiên cứu NFT.....	98
3.2.1. NFT là gì?.....	98

3.2.2. Ứng dụng phổ biến của NFT là gì?.....	99
3.2.3. Lợi ích ẩn tượng của NFT là gì?.....	100
3.2.4. Điểm danh các NFT token tiềm năng hiện nay.....	102
3.2.5. Lạm dụng NFT quá mức sẽ gây ra nguy cơ tiềm ẩn nào?.....	103
3.2.6. Các thuật ngữ liên quan đến NFT.....	104
3.2.7. Tiêu chuẩn NFT Token ERC-721 và ERC-1155.....	106
3.2.7.1. ERC-721.....	106
3.2.7.2. ERC-1155.....	107
3.3. Quy trình phát triển NFT.....	108
3.3.1. Hợp đồng Thông minh NFT (Smart Contracts).....	108
3.3.2. Tạo smart contract cho NFT.....	109
3.3.3. Triển khai smart contract cho NFT.....	109
3.3.4. Minting, Quản lý và Giao dịch NFT.....	110
3.4. Phát triển ứng dụng thực tế.....	111
3.4.1. Giới thiệu.....	111
3.4.2. Cấu trúc dự án.....	112
3.4.3. Triển khai dự án.....	117
3.4.3.1. Chạy dự án trên máy tính bằng mạng cục bộ.....	117
3.4.3.2. Triển khai dự án trên vercel.....	120
TỔNG KẾT.....	124
TÀI LIỆU THAM KHẢO.....	125

PHỤ LỤC HÌNH ẢNH

Hình 1.1. Sơ đồ toàn cảnh DDD.

Hình 1.2. Sơ đồ dự án.

Hình 1.3. Sơ đồ Microservice ví dụ.

Hình 1.4. Câu lệnh để clone dự án về.

Hình 1.5. Dự án sau khi kéo về.

Hình 1.6. Cấu trúc mã nguồn dự án sau khi kéo về trong IDE IntelliJ.

Hình 1.7. Chạy câu lệnh trên terminal của IDE IntelliJ.

Hình 1.8. Các container được khởi động sau khi chạy câu lệnh.

Hình 1.9. Tüm services đang chạy đồng thời trên IDE IntelliJ.

Hình 1.10. Cấu trúc thư mục auth-service.

Hình 1.11 Cấu trúc thư mục file-storage.

Hình 1.12. Cấu trúc thư mục job-service.

Hình 1.13. Cấu trúc thư mục notification-service.

Hình 1.14. Cấu trúc thư mục user-service.

Hình 1.15. Cấu trúc thư mục config-server.

Hình 1.16. Cấu trúc thư mục Eureka Server.

Hình 1.17. Cấu trúc thư mục API Gateway.

Hình 1.18. Test API Đăng ký tài khoản.

Hình 1.19. Test API Đăng nhập tài khoản.

Hình 1.20. Test API Lấy thông tin tài khoản.

Hình 1.21. Test API Tạo Category.

Hình 1.22. Test API Cập nhật Category.

Hình 1.23. Test API Tạo job.

Hình 1.24. Test API Cập nhật job.

Hình 1.25. Test API Lấy toàn bộ thông tin người dùng.

Hình 1.26. Test API Cập nhật thông tin người dùng.

Hình 1.27. Test API Xóa tài khoản người dùng.

Hình 1.28. Mô hình kiến trúc Domain-Driven Design (DDD).

Hình 1.29. Biểu đồ Domain-Driven Design (DDD).

Hình 1.30. Cấu hình Postgres.

Hình 1.31. Cấu hình Zookeeper.

Hình 1.32. Cấu hình Kafka.

Hình 1.33. Cấu hình Kafka-ui.

Hình 2.1. Vòng tròn mối liên kết giữa 3 khái niệm.

Hình 2.2. Ví dụ Convolution.

Hình 2.3. Ví dụ model CNN.

Hình 2.4. Ví dụ các bộ lọc.

Hình 2.5. Ví dụ bước nhảy là 2.

Hình 2.6. Ví dụ hàm ReLU.

Hình 2.7. Ví dụ Max pooling 2x2.

Hình 2.8. Ảnh gốc.

Hình 2.9. Hình ảnh ELA.

Hình 2.10. Sơ đồ quá trình xử lý ảnh và huấn luyện model.

Hình 2.11. Tập dữ liệu đầu vào.

Hình 2.12. Quá trình chuẩn bị dữ liệu.

Hình 2.13. Hình ảnh gốc và hình ảnh ELA tương ứng.

Hình 2.14. Quá trình huấn luyện mô hình.

Hình 2.15. Hình một số thư viện cân thiết cho đề tài.

Hình 2.16. Cấu trúc thư mục lưu trữ hình ảnh cho tập dữ liệu.

Hình 2.17. Một số hình ảnh về động vật trong tập dữ liệu.

Hình 2.18. Một số hình ảnh về kiến trúc trong tập dữ liệu.

Hình 2.19. Một số hình ảnh về nghệ thuật trong tập dữ liệu.

Hình 2.20. Một số hình ảnh về con người trong tập dữ liệu.

Hình 2.21. Một số hình ảnh về thiên nhiên và cây cối trong tập dữ liệu.

Hình 2.22. Hình ảnh giả mạo 1.

Hình 2.23. Hình ảnh giả mạo 2.

Hình 2.24. Hình ảnh giả mạo 3

Hình 2.25. Ảnh ELA.

Hình 2.26. Định nghĩa hàm chuyển đổi ảnh gốc thành ảnh ELA.

Hình 2.27. Định nghĩa hàm chuẩn bị hình ảnh.

Hình 2.28. Định nghĩa hàm để build model CNN.

Hình 2.29. Tạo model CNN và xem bản tóm tắt về model.

Hình 2.30. Thực hiện huấn luyện model.

Hình 2.31. Tập các hình ảnh chưa qua chỉnh sửa.

Hình 2.32. Tập các hình ảnh đã qua chỉnh sửa.

Hình 2.33. Giao diện phần mềm chạy mô hình CNN.

Hình 2.34. Cửa sổ “Select Image File”.

Hình 2.35. Ảnh sau khi được chọn.

Hình 2.36. Kết quả dự đoán của mô hình “model_casia_run1”.

Hình 2.37. Kết quả dự đoán của mô hình “model_casia_run1”(2).

Hình 2.38. Kết quả dự đoán của mô hình “model_casia_run1”(3).

Hình 2.39. Kết quả dự đoán của mô hình “model_casia_run1”(4).

Hình 2.40. Biểu đồ thể hiện mức độ lỗi và độ chính xác của mô hình.

Hình 2.41. Biểu đồ ma trận nhầm lẫn cho mô hình “model_casia_run1”.

Hình 3.1. Những quan điểm phổ biến về khái niệm “Công nghệ Blockchain”.

Hình 3.2. Minh họa hoạt động chia sẻ sở cói trong chuỗi khối.

Hình 3.3. Minh họa giao thức trong Blockchain.

Hình 3.4. Dự án challenge-0-simple-nft.

Hình 3.5. Cấu trúc dự án tổng thể.

Hình 3.6. Cấu trúc thư mục và tệp trong thư mục hardhat.

Hình 3.7. Cấu trúc thư mục và tệp trong thư mục nextjs.

Hình 3.8. Logo Hardhat.

Hình 3.9. Chạy lệnh khởi động mạng blockchain Ethereum cục bộ.

Hình 3.10. Chạy lệnh triển khai hợp đồng thông minh cục bộ.

Hình 3.11. Chạy lệnh triển khai giao diện frontend.

Hình 3.12. Giao diện web của ứng dụng.

Hình 3.13. Các giao dịch gửi tiền qua lại bằng địa chỉ ví.

Hình 3.14. Các NFT được sinh ra sau khi được đúc.

Hình 3.15. Các giao dịch NFT qua lại bằng địa chỉ ví.

Hình 3.16. Logo Vercel.

Hình 3.17. Chính sửa thuộc tính defaultNetwork trong hardhat.config.ts.

Hình 3.18. Chính sửa thuộc tính defaultNetwork trong scaffold.config.ts.

Hình 3.19. Chạy lệnh triển khai trên Vercel.

Hình 3.20. Dự án đã được triển khai trên Vercel.

Hình 3.21. Giao diện ứng dụng web của dự án trên máy tính.

Hình 3.22. Giao diện ứng dụng web của dự án trên điện thoại.

LỜI MỞ ĐẦU

1. Lý do chọn đề tài.

Trong bối cảnh cuộc cách mạng công nghiệp 4.0 đang diễn ra với tốc độ nhanh chóng, việc áp dụng công nghệ blockchain vào hầu hết các ngành đã trở thành xu hướng, đặc biệt là NFT - một dạng các tác phẩm số đang phổ biến toàn cầu. Để đáp ứng nhu cầu của thị trường sở hữu, giao dịch tác phẩm số, nhóm chúng em đã chọn đề tài "Nghiên cứu phát triển ứng dụng NFT với Blockchain". Đây là một đề tài có tính ứng dụng cao, tạo ra một không gian sống động, kết nối cộng đồng các nghệ sĩ, nhà phê bình và người hâm mộ nghệ thuật với công nghệ, thúc đẩy sự sáng tạo và giúp tạo ra cầu nối văn hóa, nghệ thuật giữa con người và công nghệ trong thời đại số hóa ngày nay. Bên cạnh đó, việc nghiên cứu và phát triển ứng dụng cũng giúp nâng cao kỹ năng lập trình và hiểu sâu hơn về quy trình phát triển phần mềm.

2. Mục tiêu.

Mục tiêu của đề tài là nghiên cứu phát triển ứng dụng NFT với Blockchain . Trong bối cảnh cuộc cách mạng công nghiệp 4.0 đang diễn ra với tốc độ nhanh chóng, sự phát triển của công nghệ thông tin đã đem lại nhiều tiện ích và cơ hội cho việc phát triển, chia sẻ văn hóa, nghệ thuật. Tuy nhiên, để tận dụng hết tiềm năng của công nghệ thông tin trong nghệ thuật số, các hệ thống cần phải có những cải tiến và nâng cấp hiện có để đáp ứng được nhu cầu sử dụng của tác giả và người dùng.

3. Bố cục chính.

Chương 1: Tìm hiểu, xây dựng ứng dụng microservices quản lý người dùng.

Chương 2: Phát hiện ảnh đã qua chỉnh sửa ứng dụng CNN, ELA và Keras

Chương 3: Nghiên cứu phát triển ứng dụng NFT với blockchain.

CHƯƠNG 1

TÌM HIỂU, XÂY DỰNG ỨNG DỤNG MICROSERVICES QUẢN LÝ NGƯỜI DÙNG

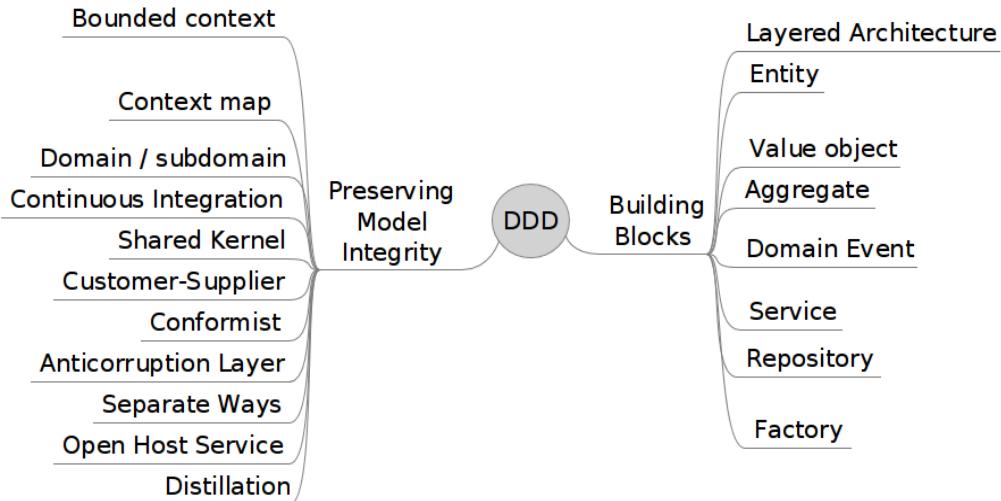
1.1. Phương pháp thiết kế DDD.

1.1.1 Tổng quan DDD.

Domain Driven Design (DDD) là một cách tiếp cận để phát triển những phần mềm phức tạp thông qua sự kết nối chặt chẽ giữa việc triển khai ứng dụng với sự phát triển của mô hình kinh doanh. Tiền đề tạo nên DDD là :

- Đặt trọng tâm dự án vào nghiệp vụ chính (core domain) và các logic của nghiệp vụ (domain logic).
- Mô hình hóa là trọng tâm, là nền tảng cho các thiết kế phức tạp.
- Sự cộng tác đầy sáng tạo giữa nhóm developer và các domain expert (chuyên gia nghiệp vụ) tạo nên tiếng nói chung để xác định và giải quyết hiệu quả các vấn đề.

Domain Driven Design (DDD) là một tư tưởng, một hướng tiếp cận trong việc giải quyết các bài toán phức tạp. DDD tập trung vào khái niệm domain (nghiệp vụ) và bóc tách bài toán dựa trên các domain đó. Đây là cái khách hàng (domain expert) nắm rõ nhất. Chúng ta phát triển ứng dụng theo yêu cầu của khách hàng nên hiển nhiên không ai hiểu các yêu cầu của hệ thống bằng khách hàng. Và khi khách hàng giải thích hệ thống cho chúng ta hiểu, họ sẽ giải thích về các domain của nó. Chính vì thế các domain sẽ làm trọng tâm và công việc của chúng ta là xây dựng nó thành các mô hình để cho tất cả mọi người cùng nắm vững đề. DDD là thiết kế sao cho không chỉ lập trình viên hiểu mà ngay cả khách hàng, những người không biết gì về mặt kỹ thuật cũng có thể nhìn vào nắm được trọng tâm của vấn đề. Trong thực tế, đây là một quá trình đòi hỏi rất nhiều kỹ năng và quá trình tiếp cận xây dựng có hệ thống.



Hình 1.1. Sơ đồ toàn cảnh DDD.

1.1.2 Các khái niệm liên quan.

Ubiquitous Language: Đây là một ngôn ngữ chung mà tất cả các thành viên trong team (dev, BA, PO, v.v.) sử dụng để giao tiếp. Nó giúp đảm bảo rằng mọi người hiểu đúng ý nghĩa của các thuật ngữ trong Domain.

Domain: Là domain (nghiệp vụ) chính của hệ thống. Nó bao gồm các quy tắc, chức năng và đối tượng liên quan đến domain đó, trong domain chính có thể bao gồm các subdomain. Sub Domain: Là 1 domain con hoạt động bên trong một domain lớn hơn. Chúng giúp chia nhỏ Domain thành các phần nhỏ hơn, dễ quản lý hơn.

Bounded Context: Là một ranh giới được định nghĩa để phân tách các Subdomain khác nhau, giúp giảm thiểu sự phức tạp và tăng khả năng mở rộng của hệ thống. Mỗi Bounded Context đại diện cho một Subdomain độc lập với những domain model của riêng subdomain đó. Chúng ta sẽ gặp khái niệm này khi nói đến kiến trúc Microservices.

Entity: Là các thực thể trong hệ thống, có thể như "Product", "Customer", "Order".

Value Object: Là các thực thể không có định danh (ID), để bổ sung thông tin cho các Entity, ví dụ như Address, tuy nhiên đối với từng bài toán, ta sẽ có cách định nghĩa các Value object riêng, không phải bài toán nào cũng

có Address là 1 Value Object, có thể trong bài toán quản lý địa chỉ, Address lại đóng vai trò là 1 Entity.

Aggregate Root: Là một đối tượng trong Aggregate chịu trách nhiệm duy trì tính nhất quán và đúng đắn của dữ liệu. Nó là điểm duy nhất để truy cập và thao tác các thành viên bên trong Aggregate.

Aggregate: Trong DDD, Aggregate là một nhóm các đối tượng (Entities và value objects) được tổ chức lại theo một quy tắc cụ thể. Mỗi Aggregate đảm bảo tính nhất quán và đúng đắn của dữ liệu bên trong nó thông qua các ràng buộc. Aggregate Root là một entity chịu trách nhiệm cho việc duy trì tính consistency của các đối tượng trong Aggregate. Khi giao tiếp giữa các Aggregate, chỉ cho phép giao tiếp thông qua Aggregate root.. Ví dụ Order Aggregate có nhiệm vụ thêm Order, xóa Order, cập nhật Order thông qua Aggregate Root là Order và entity như OrderItem. Aggregate đảm bảo tính nhất quán và đúng đắn của dữ liệu bên trong nó thông qua các ràng buộc. 1 Aggregate chứa 1 Aggregate Root để giao tiếp với các Aggregate Root ở những Aggregate Root khác.

1.2. Triển khai dự án theo link được giao.

1.2.1. Phân tích cấu trúc chương trình.

Dự án xây dựng hệ thống quản lý người dùng dựa trên kiến trúc **microservices** được phát triển bằng **Spring Boot**, nhằm giới thiệu một hệ thống có cấu trúc mạnh mẽ với các công nghệ thiết yếu giúp xây dựng dịch vụ mở rộng, an toàn và hiệu quả đối với người sử dụng hệ thống . Dự án sử dụng **Eureka Server** để quản lý dịch vụ, **Config Server** để quản lý cấu hình tập trung và **API Gateway** để điều hướng và bảo mật. Các thành phần tiên tiến như **Kafka** được sử dụng để xử lý thông điệp, trong khi **Redis** cung cấp bộ nhớ đệm nhằm cải thiện hiệu suất. **Docker** cũng được tích hợp để triển khai và mở rộng dễ dàng.

Mục tiêu dự án là xây dựng một hệ thống microservices phân tán với khả năng mở rộng cao, được tổ chức thành nhiều dịch vụ nhỏ dễ quản lý.

Đảm bảo rằng chỉ người dùng có quyền mới có thể truy cập vào các dịch vụ thông qua cơ chế xác thực và ủy quyền, quản lý cấu hình tập trung, cải thiện hiệu suất và tích hợp các dịch vụ nhằm tối ưu dự án, giúp dự án hoạt động một cách mượt mà và logic.

Cấu trúc dự án Microservice : Hệ thống bao gồm 8 service chính, mỗi service đảm nhận một chức năng riêng biệt và giao tiếp với các dịch vụ khác thông qua API Gateway:

- **Config Server:** Cung cấp khả năng quản lý cấu hình tập trung cho tất cả các microservice, đảm bảo rằng mọi thay đổi cấu hình có thể được áp dụng đồng bộ.

- **Eureka Server:** Là thành phần quan trọng để quản lý và duy trì thông tin về các dịch vụ hoạt động, giúp các microservice tự động tìm kiếm và kết nối với nhau một cách linh hoạt.

- **API Gateway:** Đóng vai trò như một cổng vào duy nhất, chịu trách nhiệm định tuyến các yêu cầu đến đúng dịch vụ và thực hiện các nhiệm vụ bảo mật như xác thực JWT.

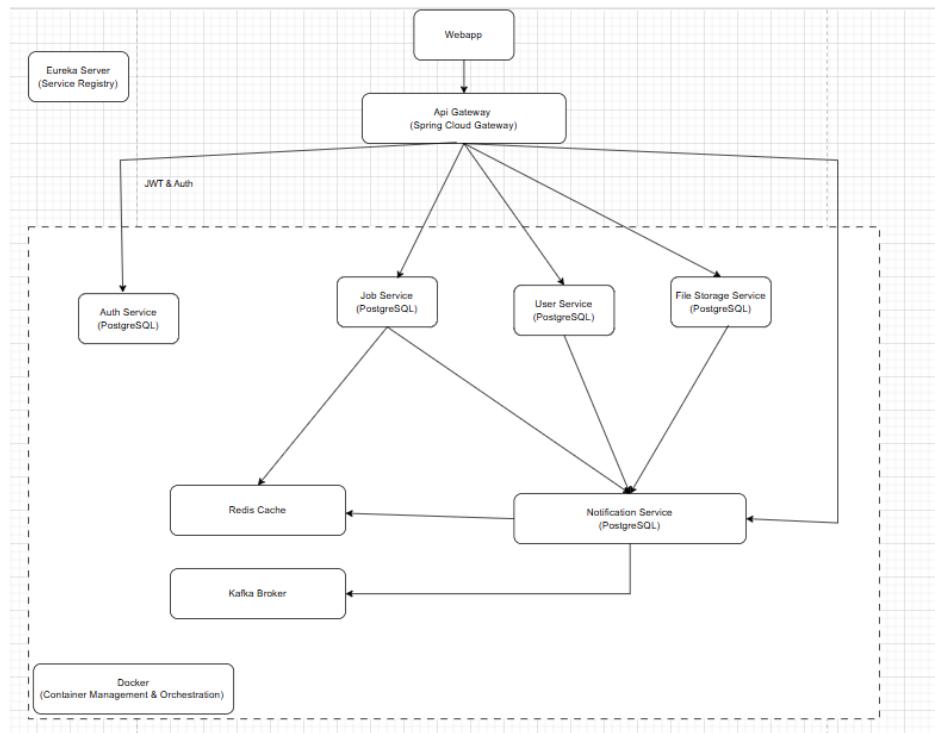
- **Auth Service:** Quản lý xác thực và phân quyền người dùng. Người dùng có thể đăng ký và đăng nhập, hệ thống hỗ trợ các vai trò khác nhau (ADMIN và USER) để đảm bảo tính bảo mật và quyền hạn khác nhau.

- **Job Service:** Xử lý các yêu cầu liên quan đến công việc, bao gồm quản lý dữ liệu công việc, truy vấn và các nghiệp vụ phức tạp.

- **User Service:** Cung cấp chức năng quản lý thông tin người dùng, bao gồm cập nhật hồ sơ, lưu trữ và truy xuất dữ liệu cá nhân.

- **Notification Service:** Đảm bảo việc gửi thông báo và cảnh báo đến người dùng một cách hiệu quả, sử dụng **Kafka** để truyền tải thông điệp và đảm bảo tính ổn định.

- **File Storage Service:** Dịch vụ chuyên biệt để quản lý các tệp tin, hỗ trợ người dùng tải lên và tải về tệp một cách an toàn và nhanh chóng.



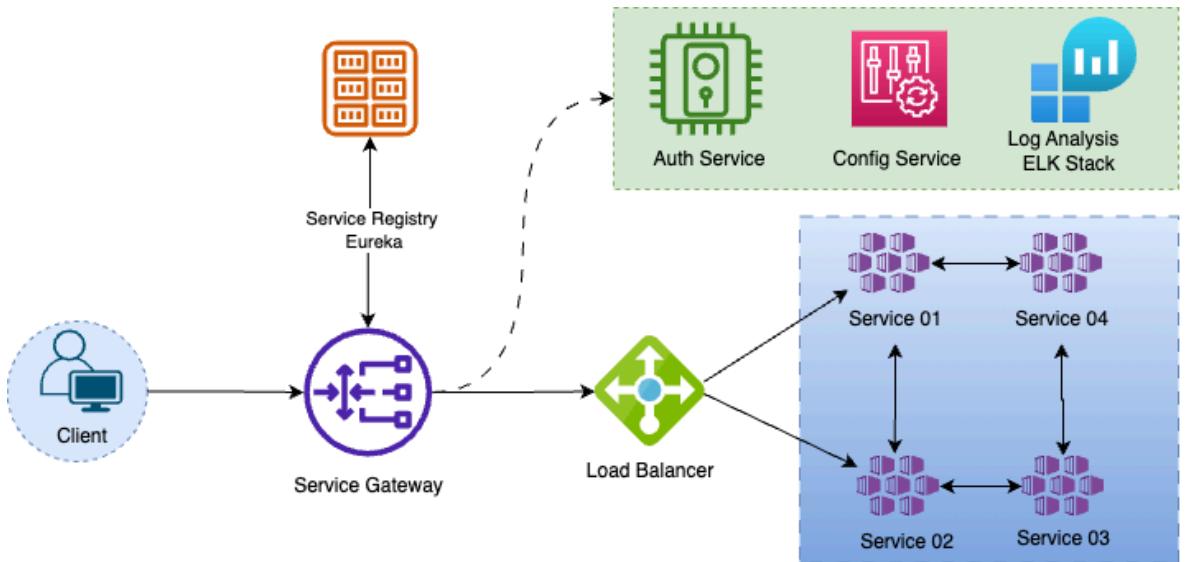
Hình 1.2. Sơ đồ dự án.

Các công nghệ sử dụng:

- **Eureka Server** là một máy chủ đăng ký dịch vụ trong hệ thống Microservices. Nó đảm nhiệm việc đặt tên cho mỗi microservice. Khi có nhiều microservices được triển khai và hoạt động trên nhiều instance khác nhau, không cần phải mã hóa địa chỉ IP cứng của mỗi service, thay vào đó, chúng ta có thể sử dụng tên service đã đăng ký trên Eureka Server để tìm kiếm và truy cập các dịch vụ này. Điều này giúp cho việc quản lý và mở rộng các dịch vụ một cách dễ dàng và hiệu quả hơn.

Vì vậy, mỗi dịch vụ đăng ký với Eureka và gửi yêu cầu ping tới Eureka server để thông báo rằng nó đang hoạt động.

Nếu Eureka server không nhận được bất kỳ thông báo nào từ một dịch vụ, dịch vụ đó sẽ bị hủy đăng ký tự động từ Eureka server.



Hình 1.3. Sơ đồ Microservice ví dụ.

Spring Cloud Config là một mô-đun của Spring Cloud cung cấp việc lưu trữ và phục vụ các cấu hình phân tán trên nhiều ứng dụng và môi trường. Trong các ứng dụng thực tế, các ứng dụng có các thông tin cấu hình chung và riêng. Chẳng hạn như các ứng dụng cần dùng chung thông tin về cơ sở dữ liệu để truy cập, ngoài ra mỗi ứng dụng cần chạy trên một cổng port riêng. Trong bài viết này, chúng tôi sẽ thiết lập một ứng dụng ConfigServer quản lý tất cả các tệp cấu hình được lưu trên folder và một ứng dụng Client Service tiêu thụ nhiều tệp cấu hình (multiple config file) này khi khởi động.

- **API Gateway** là một cổng trung gian, nó là cổng vào duy nhất tới hệ thống microservices của chúng ta, api gateway sẽ nhận các requests từ phía client, chỉnh sửa, xác thực và điều hướng chúng đến các API cụ thể trên các services phía sau. Ngoài nhiệm vụ chính là proxy request thì một hệ thống API Gateway thường sẽ đảm nhận luôn vài vai trò khác như bảo mật API, monitoring, analytics số lượng requests cũng như tình trạng hệ thống phía sau.

- **Kafka** là hệ thống truyền thông điệp phân tán, độ tin cậy cao, dễ dàng mở rộng và có thông lượng cao. Kafka cung cấp cơ chế offset (có thể hiểu như tương tự như chỉ số của một mảng) để lấy thông điệp một cách linh hoạt, cho phép các ứng dụng xử lý có thể xử lý lại dữ liệu nếu việc xử lý trước đó bị lỗi. Ngoài ra, cơ chế “đăng ký” theo dõi cho phép việc lấy thông điệp ra gần

như tức thời ngay khi dữ liệu đi vào hàng đợi. Kafka được thiết kế hỗ trợ tốt cho việc thu thập dữ liệu thời gian thực. Kafka sử dụng mô hình truyền thông public-subscribe, bên public dữ liệu được gọi là producer bên subscribe nhận dữ liệu theo topic được gọi là consumer. Kafka có khả năng truyền một lượng lớn dữ liệu trong thời gian thực, trong trường hợp bên nhận chưa nhận dữ liệu vẫn được lưu trữ sao lưu trên một hàng đợi và cả trên ổ đĩa bảo đảm an toàn.

- **JWT (JSON Web Token)** là một tiêu chuẩn mã nguồn mở (RFC 7519) dùng để truyền tải thông tin an toàn, gọn nhẹ và khép kín giữa các bên tham gia dưới format JSON. Thông tin được chia sẻ trong JWT được xác thực và tin cậy thông qua chữ ký số (Digital signature). Các bên sẽ sử dụng mật mã khoá đối xứng (cùng với HMAC) hoặc dùng mật mã khoá công khai (cùng public và private key) để thực hiện ký số (sign). JWT gồm 3 phần chính, và phần tách nhau bằng một dấu chấm (.): Header, Payload, Signature. Được sử dụng trong xác thực và phân quyền.

- **Redis** là một hệ quản trị cơ sở dữ liệu NoSQL, lưu trữ dữ liệu dưới dạng cặp khóa-giá trị (key-value) và hỗ trợ nhiều cấu trúc dữ liệu như chuỗi (strings), danh sách (lists), tập hợp (sets), bản đồ (hashes), và nhiều loại dữ liệu phức tạp khác. Redis thường được sử dụng như một **cache** (bộ nhớ đệm) hoặc một **message broker** (dịch vụ truyền tin)

- **Docker** là một nền tảng mã nguồn mở dùng để phát triển, vận hành và chạy các ứng dụng trong môi trường **container**. Container là những môi trường ảo hóa nhẹ cho phép chạy ứng dụng một cách độc lập và có thể chạy trên bất kỳ máy chủ nào mà không lo về sự khác biệt giữa các môi trường phát triển, thử nghiệm và sản xuất.

1.2.2. Cài Đặt chương trình.

1.2.2.1. Hướng dẫn cài đặt.

Phần này sẽ cung cấp chi tiết các bước cần thiết để cài đặt phần mềm quản lý người dùng ứng dụng mô hình microservices một cách nhanh chóng và hiệu quả. Các yêu cầu hệ thống, công cụ hỗ trợ và lưu ý quan trọng cũng sẽ

được đê cập nh m giảm thiểu các l i phát sinh trong quá trình triển khai. Cách tiếp cận từng bước s  gi p người dùng d ng d ng theo d i và c i d t ph n m m th nh công ngay từ lần đầu ti n.

B rc 1: C i d t các công cụ Docker và IDE IntelliJ:

- Đ c c i d t công cụ Docker trên máy tính truy cập trang web Docker theo link sau: <https://www.docker.com/products/docker-desktop/>

- Đ c c i d t công cụ IDE IntelliJ trên máy tính truy cập trang web sau theo link: <https://www.jetbrains.com/idea/download/?section=windows>

B rc 2: Truy cập vào link sau để thực hiện việc tải xuống dự án: <https://github.com/devxsy/spring-boot-microservices>.

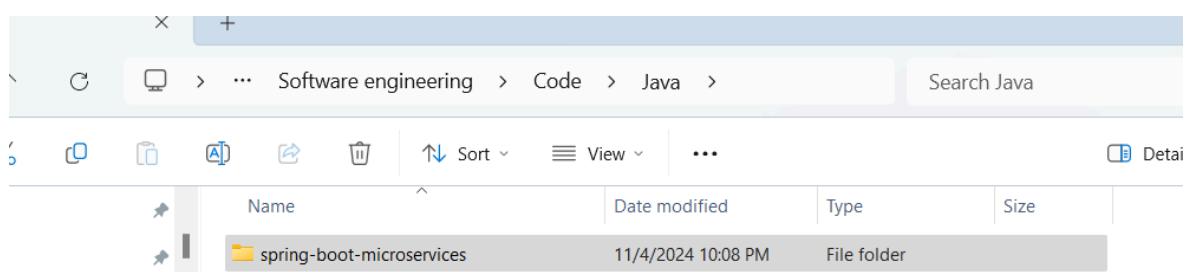
Chọn thư mục để lưu dự án và thực hiện clone dự án về bằng câu lệnh:



```
MINGW64:/d/Software engineering/Code/Java
ADMIN@HOANGPHAN MINGW64 /d/Software engineering/Code/Java
$ git clone https://github.com/devxsy/spring-boot-microservices.git
```

Hình 1.4. Câu lệnh để clone dự án về.

Dự án sau khi được kéo về ta thực hiện mở nó bằng IDE IntelliJ:

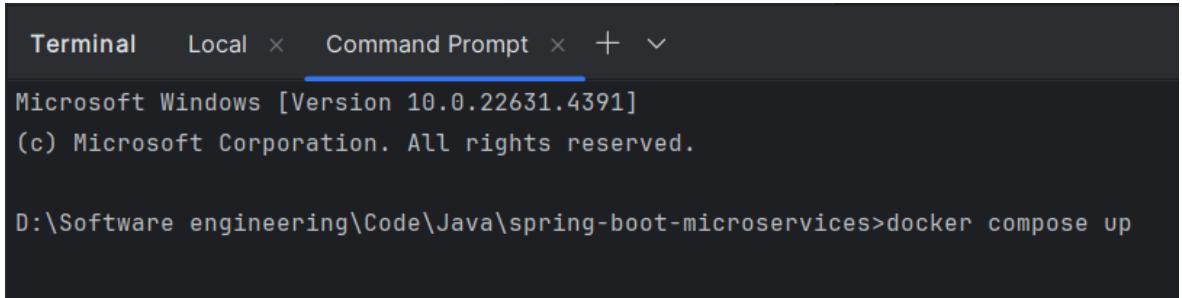


Hình 1.5. Dự án sau khi kéo về.



Hình 1.6. Cấu trúc mã nguồn dự án sau khi kéo về trong IDE IntelliJ.

Bước 3: Mở terminal, điều hướng đến thư mục dự án và chạy câu lệnh:



Hình 1.7.Chạy câu lệnh trên terminal của IDE IntelliJ.

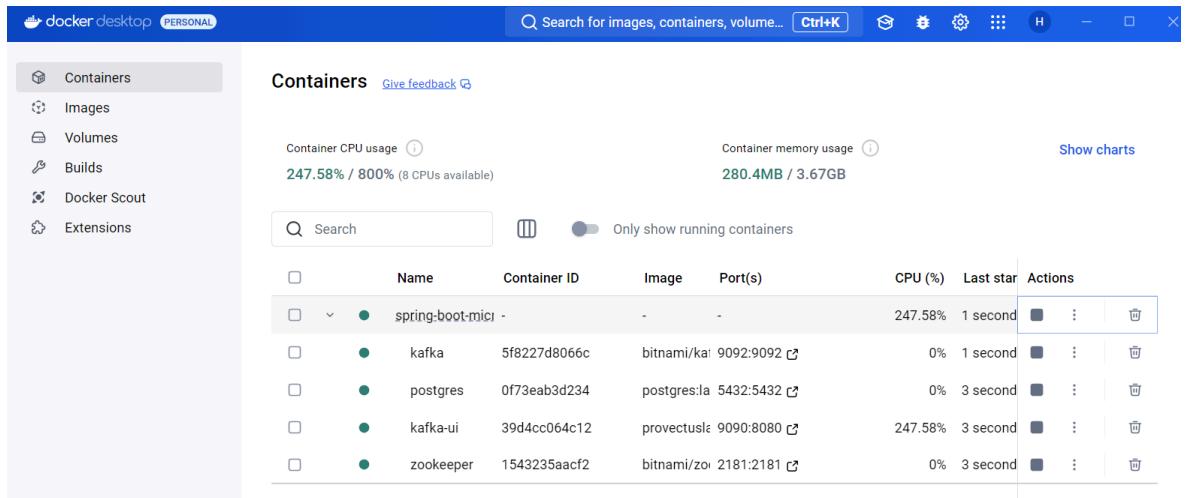
Câu lệnh “docker compose up” sẽ khởi động các container được định nghĩa trong tệp cấu hình docker-compose.yml hiện tại. Cụ thể:

- Tạo và chạy container: Nếu các container chưa được tạo, “docker compose up” sẽ tạo chúng theo định nghĩa trong docker-compose.yml.
- Liên kết các dịch vụ: Docker Compose sẽ khởi động tất cả các dịch vụ (containers) đã được khai báo và thiết lập các kết nối mạng cần thiết giữa chúng.

- Theo dõi logs: “docker compose up” theo dõi và hiển thị logs của tất cả các dịch vụ, giúp bạn dễ dàng quan sát các thông báo từ từng container trong thời gian thực.

Lệnh này cũng có tùy chọn “-d” (“docker compose up -d”) để chạy các container ở chế độ nền, không hiển thị logs trên terminal.

Kết quả sau khi chạy câu lệnh “docker compose up”, mở docker lên và ta sẽ thấy các container đã được khởi động:



Hình 1.8. Các container được khởi động sau khi chạy câu lệnh.

Hình 1.8 cho biết các container đang hoạt động và thông số của nó. Cụ thể:

kafka:

- Image: bitnami/kafka.
- Port(s): 9092.
- CPU usage: 0%.

postgres:

- Image: postgres.
- Port(s): 5432 (port mặc định cho PostgreSQL).
- CPU usage: 0%.

kafka-ui:

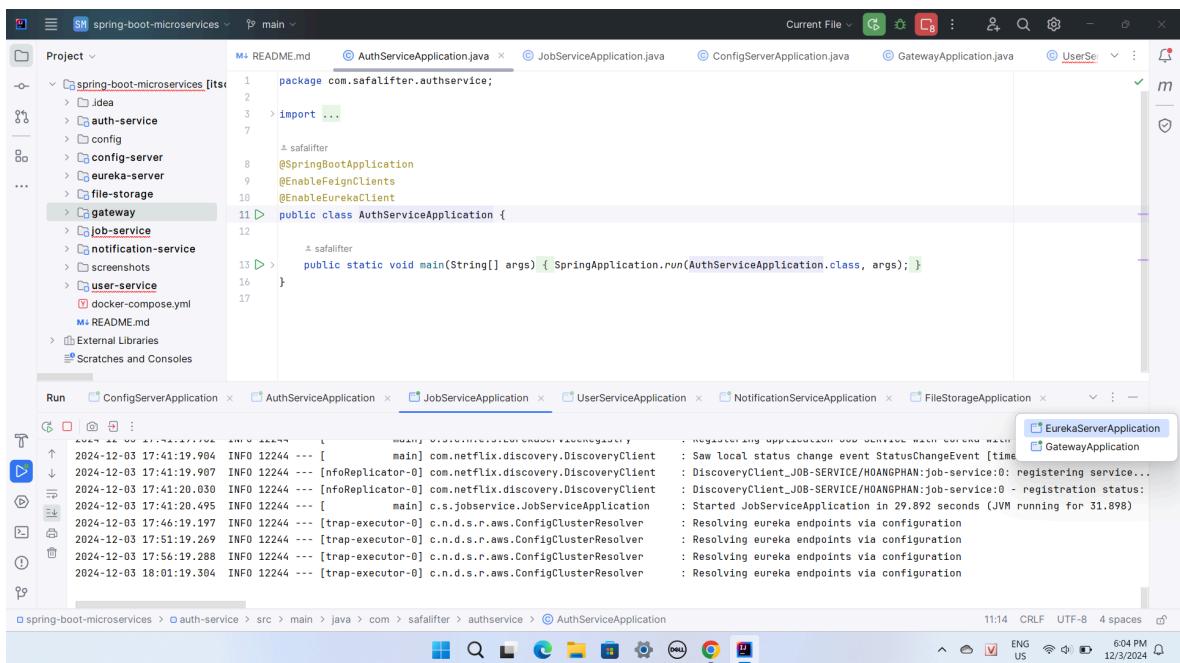
- Image: provectuslabs/kafka-ui.
- Port(s): 9090-8080.

- CPU usage: 247.58% (bằng với container spring-boot-mci).

zookeeper:

- Image: bitnami/zookeeper.
- Port(s): 2181.
- CPU usage: 0%.

Bước 4: Trên IDE IntelliJ, thực hiện chạy đồng thời tất cả tám services đó là: Eureka Server, Gateway, Config Server, auth-service, user-service, job-service, notification-service và file-storage) bằng cách truy cập file có đuôi cuối trong tên là “...Application” ở mỗi chương trình và nhấn nút mũi tên xanh lá góc phía bên phải hoặc tổ hợp phím “Shift + F10”.



Hình 1.9. Tám services đang chạy đồng thời trên IDE IntelliJ.

Trên đây là các bước cần thiết để cài đặt phần mềm quản lý người dùng ứng dụng mô hình microservices. Các dependencies được dùng trong dự án microservices là:

- Core và Spring Framework:
 - + Spring: Một framework mạnh mẽ cho phát triển ứng dụng Java, cung cấp các module và công cụ giúp quản lý các thành phần ứng dụng và các dependency một cách dễ dàng.

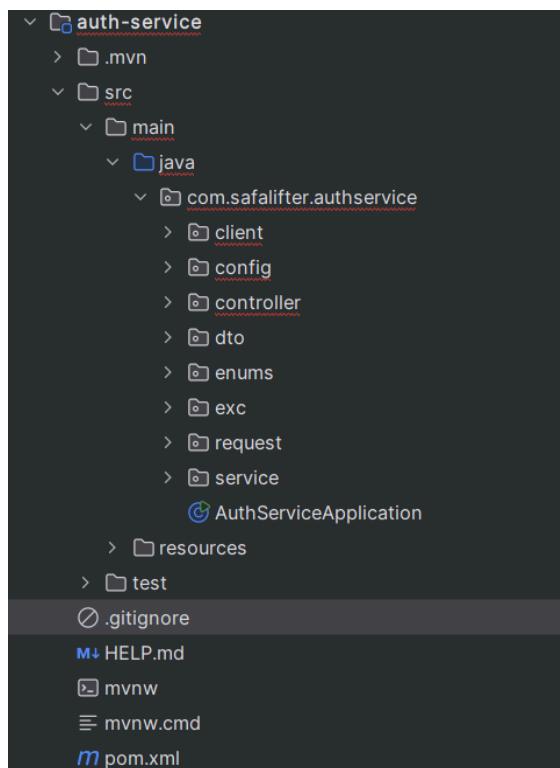
- + Spring Boot: Tạo các ứng dụng Spring một cách đơn giản với các cấu hình tự động và tích hợp tốt, giảm thiểu code cấu hình, giúp khởi chạy ứng dụng nhanh chóng.
- + Spring Security: Một module bảo mật của Spring, hỗ trợ xác thực (authentication) và phân quyền (authorization), bảo vệ ứng dụng khỏi các mối đe dọa bảo mật phổ biến.
- + Spring Security JWT: Hỗ trợ tạo và xác thực các JSON Web Tokens (JWT) trong Spring Security, thường dùng để quản lý các phiên đăng nhập không trạng thái trong ứng dụng RESTful.
- + Authentication: Cung cấp cơ chế xác thực người dùng, đảm bảo chỉ người dùng hợp lệ có thể truy cập vào ứng dụng.
- + Authorization: Xác định quyền truy cập cho từng người dùng hoặc nhóm người dùng, bảo đảm rằng người dùng chỉ có thể truy cập vào những chức năng được phép.
- + Spring Web: Cho phép xây dựng các ứng dụng web dựa trên RESTful hoặc MVC, xử lý các request và response HTTP.
- Communication và Data Handling:
 - + FeignClient: Một HTTP client giúp gọi đến các dịch vụ khác trong hệ thống microservices theo cách dễ dàng, tạo các request HTTP một cách tự động và quản lý kết nối.
 - + Spring Data: Một bộ công cụ giúp thao tác và xử lý dữ liệu dễ dàng, cung cấp các repository mặc định để thực hiện các thao tác CRUD.
 - + Spring Data JPA: Tích hợp với JPA (Java Persistence API), hỗ trợ truy vấn và xử lý dữ liệu từ cơ sở dữ liệu quan hệ, như PostgreSQL.
- 11. PostgreSQL: Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, lưu trữ dữ liệu của ứng dụng.
- Cloud và Microservices:

- + Spring Cloud: Một bộ công cụ của Spring dành cho các ứng dụng phân tán, cung cấp các chức năng như phát hiện dịch vụ, điều phối, cấu hình và cân bằng tải.
 - + Spring Cloud Gateway Server: Làm nhiệm vụ gateway, xử lý và định tuyến các request đến các dịch vụ khác, đồng thời thực hiện bảo mật và xử lý request.
 - + Spring Cloud Config Server: Một máy chủ cấu hình trung tâm, quản lý cấu hình của các microservices từ một nguồn duy nhất.
 - + Spring Cloud Config Client: Cho phép các ứng dụng lấy cấu hình từ Spring Cloud Config Server.
 - + Netflix Eureka Server: Cung cấp dịch vụ quản lý và phát hiện các dịch vụ (service discovery), lưu trữ danh sách các microservices và các endpoint tương ứng.
 - + Netflix Eureka Client: Cho phép các microservices tự động đăng ký vào Eureka Server để dễ dàng phát hiện bởi các dịch vụ khác.
 - Database và Message Broker:
 - + PostgreSQL: Như đã giải thích, đây là hệ quản trị cơ sở dữ liệu lưu trữ dữ liệu chính cho ứng dụng.
 - + Kafka: Một message broker phân tán, hỗ trợ giao tiếp và xử lý các message giữa các hệ thống với khả năng lưu trữ, xử lý dòng dữ liệu mạnh mẽ.
 - + Redis: Hệ thống lưu trữ dữ liệu trên bộ nhớ đệm (cache) tốc độ cao, giúp tối ưu hiệu năng ứng dụng, đặc biệt là trong các yêu cầu đọc dữ liệu.
 - Docker và DevOps:
 - + Docker: Một nền tảng ảo hóa cấp container giúp đóng gói ứng dụng và các dependency thành một package có thể chạy ở bất cứ đâu, dễ dàng quản lý và triển khai ứng dụng.
 - Tiện ích:
 - + Validation: Thực hiện các kiểm tra và xác thực dữ liệu đầu vào để đảm bảo tính chính xác và bảo mật.

- + File Storage: Quản lý việc lưu trữ các file như hình ảnh, tài liệu và video cho ứng dụng.
- + ModelMapper: Một thư viện dùng để ánh xạ dữ liệu giữa các đối tượng khác nhau (DTO và Entity), hỗ trợ việc chuyển đổi dữ liệu dễ dàng.
- + OpenAPI UI: Cung cấp giao diện để tương tác và thử nghiệm các API của ứng dụng, giúp kiểm thử và tài liệu hóa API.
- + Lombok: Một thư viện Java giúp giảm thiểu lượng code lặp lại trong các class Java như getter, setter, và constructor.
- + Log4j2: Một thư viện logging giúp ghi nhận các thông tin, lỗi trong ứng dụng, hỗ trợ ghi nhận log với hiệu suất cao và nhiều tùy chọn cấu hình.

1.2.2.2. Chi tiết cụ thể từng service.

a. auth-service.



Hình 1.10. Cấu trúc thư mục auth-service.

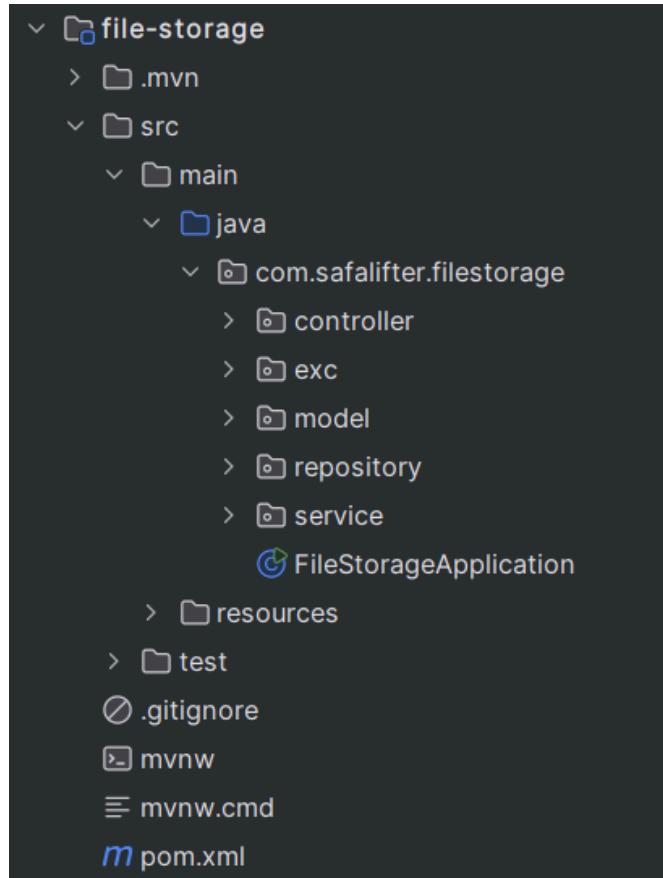
Chức năng: Dịch vụ xác thực và ủy quyền người dùng.

Vai trò: Xử lý các yêu cầu đăng nhập, đăng ký, xác minh token, và kiểm tra quyền truy cập của người dùng đến các tài nguyên.

Cấu trúc:

- Model: UserDTO , TokenDTO , RegisterDTO.
- Controller: AuthController.
- Service: AuthServiceApplication.
- Request :LoginRequest , RegisterRequest.

b. file-storage.



Hình 1.11 Cấu trúc thư mục file-storage.

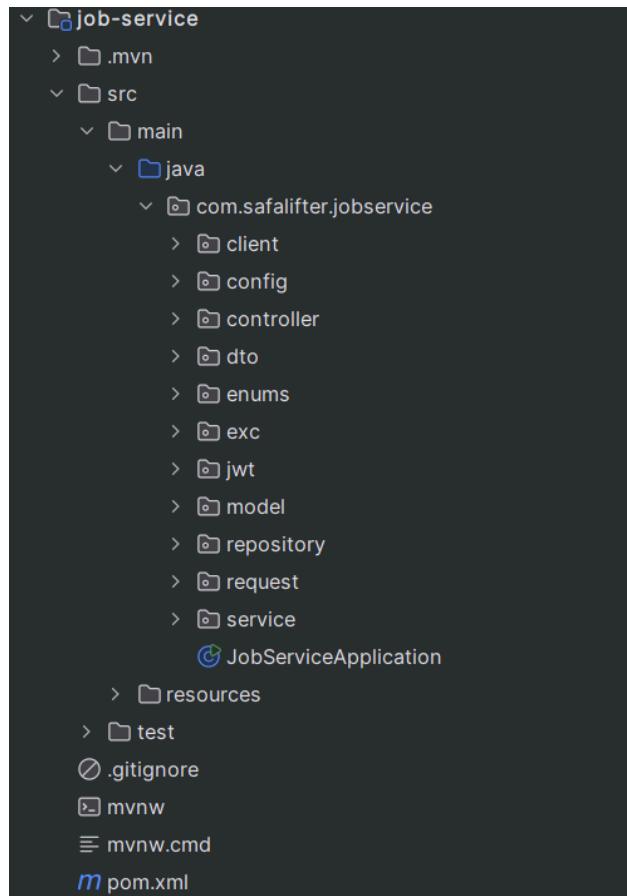
Chức năng: Biểu diễn file được lưu trữ.

Vai trò: Lưu file và đường dẫn file.

Cấu trúc:

- Controller: StorageController.
- Model: File.
- Repository : FileRepository.
- Service : FileStorageApplication.

c. job-service.



Hình 1.12. Cấu trúc thư mục job-service.

Chức năng:

- Thực thi các tác vụ định kỳ: Chạy các công việc theo lịch trình định sẵn (cron expression) hoặc theo chu kỳ nhất định (ví dụ: hàng ngày, hàng tuần, hàng tháng).
 - Thực thi các tác vụ theo yêu cầu: Chạy các công việc khi nhận được yêu cầu từ các dịch vụ khác.
 - Quản lý trạng thái của các tác vụ: Theo dõi trạng thái thực thi (thành công, thất bại, đang thực thi), ghi log lỗi, v.v.

Vai trò:

- Tăng hiệu suất cho hệ thống: Chuyển các tác vụ nặng về xử lý dữ liệu, gửi email, tạo báo cáo, v.v. ra khỏi luồng xử lý chính của ứng dụng, giúp cải thiện tốc độ phản hồi cho người dùng.

- Tăng độ tin cậy cho hệ thống: Cho phép thực thi các tác vụ quan trọng một cách tự động, đảm bảo các công việc được thực hiện đúng lịch trình, dù có sự cố xảy ra.

- Tăng tính linh hoạt cho hệ thống: Cho phép điều chỉnh và quản lý các tác vụ một cách dễ dàng.

Cấu trúc:

- Client: FileStorageClient, UserServiceClient.

- Controller: AdvertController, CategoryController, JobController, OfferController.

- DTO: AdvertDto, CategoryDTO, JobDTO, OfferDTO, UserDTO.

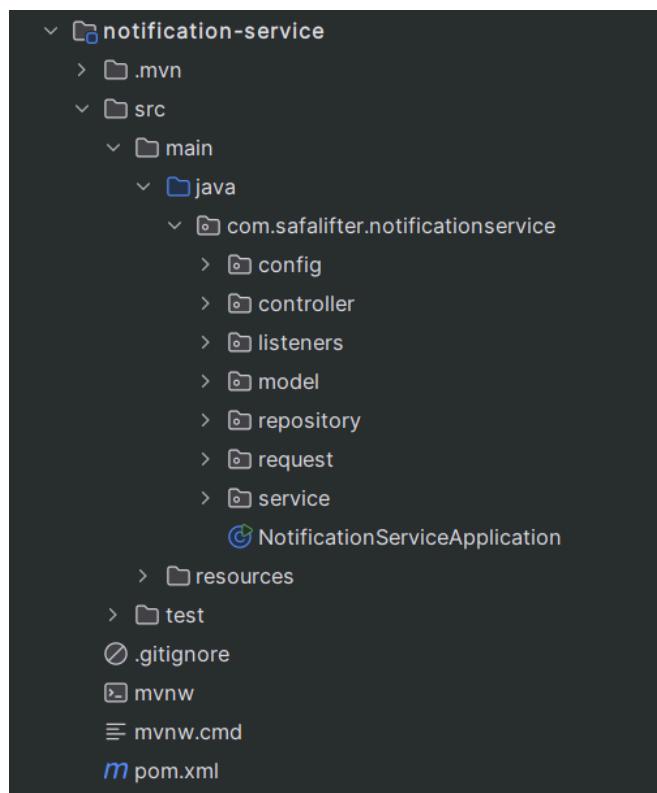
- Model: Advert, BaseEntity, Category, Job, Offer.

- Repository: AdvertRepository, CategoryRepository, JobRepository, OfferRepository.

- Request: advert, category, job, offer, notification.

- Service: JobServiceApplication.

d. notification-service.



Hình 1.13. Cấu trúc thư mục notification-service.

Chức năng:

- Nhận thông báo từ các dịch vụ khác: Dịch vụ này sẽ nhận được các thông báo từ các dịch vụ khác, chứa nội dung thông báo, người nhận, loại thông báo, v.v.
- Xử lý thông báo: Phân tích nội dung, định dạng thông báo, lựa chọn kênh truyền thông phù hợp (ví dụ: email, SMS, push notification).
- Gửi thông báo: Sử dụng các dịch vụ bên ngoài (ví dụ: dịch vụ email, SMS, push notification) để gửi thông báo cho người dùng.
- Quản lý trạng thái thông báo: Theo dõi trạng thái của thông báo (gửi thành công, thất bại), lưu trữ thông tin, v.v.

Vai trò:

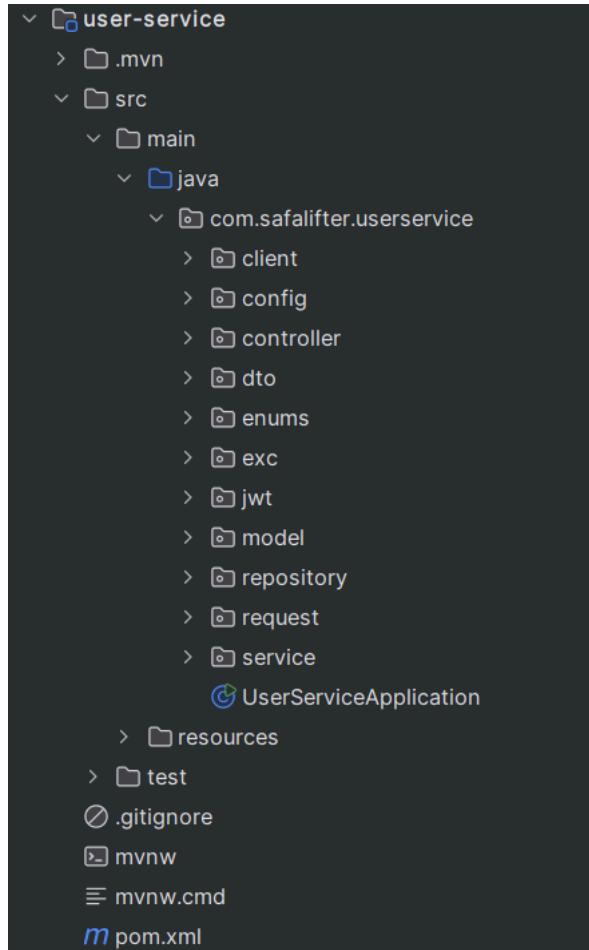
- Cải thiện trải nghiệm người dùng: Thông báo kịp thời giúp người dùng cập nhật thông tin quan trọng, theo dõi các sự kiện, và tương tác với ứng dụng một cách hiệu quả.
- Thúc đẩy tương tác: Thông báo có thể được sử dụng để thu hút người dùng, nhắc nhở họ về các hoạt động trong ứng dụng, tăng tỷ lệ sử dụng ứng dụng.
- Tăng tính cá nhân hóa: Thông báo có thể được cá nhân hóa dựa trên hành vi, sở thích của từng người dùng, giúp tăng sự hài lòng và gắn kết với ứng dụng.
- Nâng cao độ tin cậy: Cung cấp cơ chế thông báo cho người dùng về các sự cố, lỗi, hoặc cập nhật quan trọng, giúp tăng độ tin cậy của ứng dụng.
- Tăng hiệu quả vận hành: Giúp các dịch vụ khác tập trung vào chức năng cốt lõi, thay vì phải tự xử lý việc gửi thông báo.

Cấu trúc:

- Controller: NotificationController.
- Listeners: NotificationListeners.
- Model: Notification.
- Repository: NotificationRepository.

- Request: SendnotificationRequest.
- Service: NotificationServiceApplication.

e. user-service.



Hình 1.14. Cấu trúc thư mục user-service.

Chức năng:

- Quản lý thông tin người dùng: Lưu trữ, cập nhật, xóa, tìm kiếm thông tin người dùng như: Tên người dùng, Mật khẩu, Email, Số điện thoại, Địa chỉ, Vai trò (role), Quyền hạn (permission) ...
- Xác thực (Authentication): Kiểm tra tính hợp lệ của thông tin đăng nhập (tên người dùng, mật khẩu) để xác định người dùng đang truy cập là ai.
- Ủy quyền (Authorization): Kiểm tra xem người dùng có quyền truy cập vào tài nguyên hay chức năng cụ thể hay không.
- Cung cấp API cho các dịch vụ khác: Cho phép các dịch vụ khác truy xuất, cập nhật, xóa, tìm kiếm thông tin người dùng.

- Quản lý vai trò và quyền hạn: Cho phép quản lý vai trò và quyền hạn của người dùng, cấp quyền truy cập vào các chức năng cụ thể.

- Quản lý tài khoản người dùng.

Vai trò:

- Bảo mật thông tin người dùng: Giúp bảo vệ thông tin người dùng khỏi truy cập trái phép, đảm bảo tính riêng tư và an toàn cho dữ liệu.

- Quản lý quyền truy cập: Kiểm soát quyền truy cập của người dùng vào các tài nguyên và chức năng của hệ thống, đảm bảo tính bảo mật và quyền riêng tư.

- Cung cấp dữ liệu người dùng cho các dịch vụ khác: Cho phép các dịch vụ khác sử dụng thông tin người dùng một cách an toàn và hiệu quả.

- Giúp đơn giản hóa việc phát triển ứng dụng: Các dịch vụ khác không cần phải tự quản lý thông tin người dùng, giúp giảm thiểu lỗi và tăng hiệu quả phát triển.

- Nâng cao tính linh hoạt: Cho phép dễ dàng cập nhật và thay đổi các chức năng liên quan đến quản lý người dùng mà không ảnh hưởng đến các dịch vụ khác.

Cấu trúc:

- Client: CustomErrorDecoder , FileStorageClient.

- Config: BeanConfig, FeignConfig, OpenApiConfig, SecurityConfig.

- Controller: UserController.

- DTO: AuthUserDTO , UserDTO.

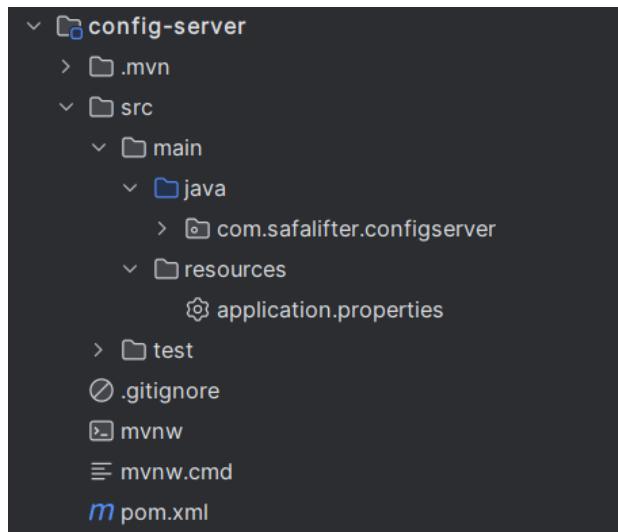
- Model: BaseEntity ,User, UserDetails.

- Repository: UserRepository.

- Request: RegisterRequest, UserUpdateRequest.

- Service: UserServiceApplication.

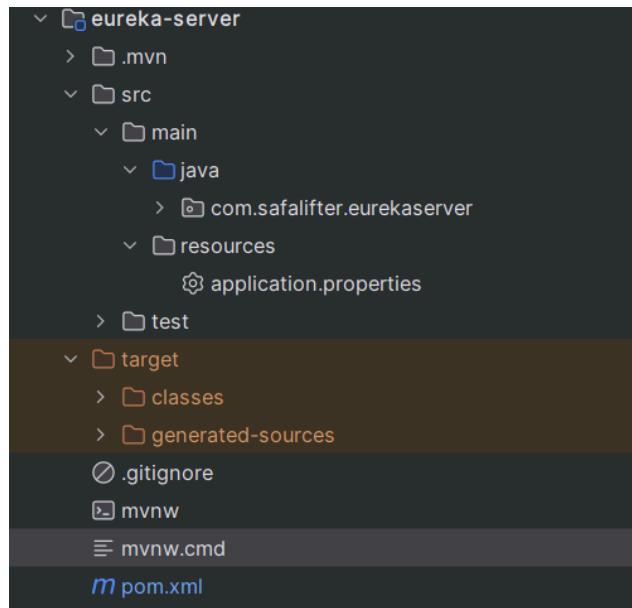
f. Config Server.



Hình 1.15. Cấu trúc thư mục config-server.

Chức năng: Cung cấp khả năng quản lý cấu hình tập trung cho tất cả các microservice, đảm bảo rằng mọi thay đổi cấu hình có thể được áp dụng đồng bộ.

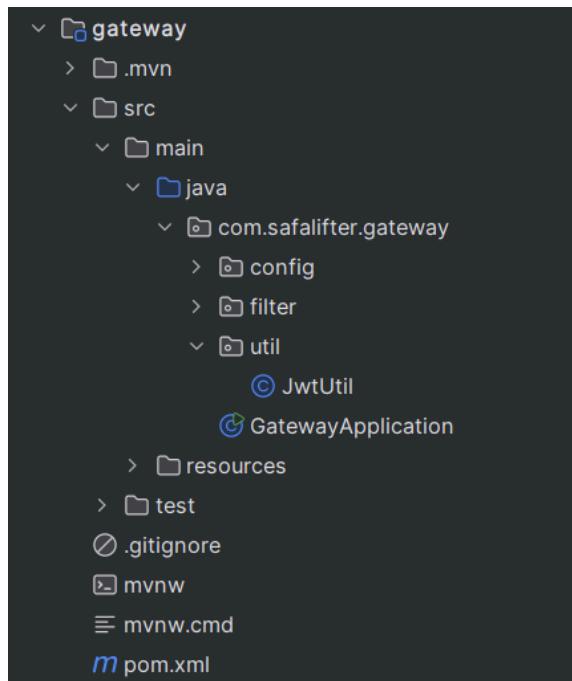
g. Eureka Server.



Hình 1.16. Cấu trúc thư mục Eureka Server.

Chức năng : Service cung cấp chức năng quản lý và duy trì thông tin về các dịch vụ hoạt động, giúp các microservice tự động tìm kiếm và kết nối với nhau một cách linh hoạt.

h. API Gateway.



Hình 1.17. Cấu trúc thư mục API Gateway.

Chức năng: service này đóng vai trò là cổng vào duy nhất, chịu trách nhiệm định tuyến các yêu cầu đến đúng dịch vụ và thực hiện các nhiệm vụ bảo mật như xác thực JWT.

1.2.2.3. Kết quả chạy chương trình.

Trong phần này, nhóm sẽ sử dụng công cụ Postman để thực hiện test api mà các services cung cấp trong dự án.

a, Đăng ký tài khoản người dùng.

Request:

- URL: `http://localhost:8080/v1/auth/register`
- Method: POST.
- Body(raw): `{ "username": <Tên username>, "password": <Mật khẩu>, "email": <Email> }`

Response: Ta có cấu trúc response như sau:

```
{ "id": <Id user vừa đăng ký>, "username": <Tên username>, "email": <Email> }
```

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/v1/auth/register
- Body (raw JSON):**

```
{
  "username": "hoangpt21",
  "password": "12345678Pth",
  "email": "hoangpt21@gmail.com"
}
```
- Response:** 200 OK


```
{
  "id": "a55fb05-94cc-4233-8c32-837ae3565a7a",
  "username": "hoangpt21",
  "email": "hoangpt21@gmail.com"
}
```

Hình 1.18. Test API Đăng ký tài khoản.

b, Đăng nhập tài khoản người dùng.

Request:

- URL: http://localhost:8080/v1/auth/login
- Method: POST.
- Body(raw): { “username”: <Tên username>, “password”: <Mật khẩu> }

Response: Ta có cấu trúc response như sau:

{ “token”: <Access token của user> }

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/v1/auth/login
- Body (raw JSON):**

```
{
  "username": "hoangpt",
  "password": "12345678Pth"
}
```
- Response:** 200 OK


```
{
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJob2FuZ3B0IiwiaXNzIjoiUk9MRV9VU0VSIiwiaWF0IjoxNzMxMTQyMzkwLCJleHAiOjE3MzExNDU5OTB9.eyJvBR1ir3gKCNYvGWh5qILGY9xFouM4nnwxTXSqTbes"
}
```

Hình 1.19. Test API Đăng nhập tài khoản.

c, Lấy thông tin tài khoản người dùng.

Request:

- URL: http://localhost:8080/v1/user/getUserById/{id}
- Method: GET.
- Authorization: Bearer Token.

Response: Ta có cấu trúc response như sau:

{ “id”: <Id user vừa đăng ký>, “username”: <Tên username>, “email”: <Email> }

The screenshot shows a Postman test environment. The URL is `http://localhost:8080/v1/user/getUserById/a55fdbd05-94cc-4233-8c32-837ae3565a7a`. The method is set to `GET`. In the Authorization section, the `Auth Type` is set to `Bearer Token`. The token value is a long string of characters: `eyJhbGciOiJIUzI1NiJ9.eyJzdWliOiJob2FuZ3B0MjEiLCJpc3MiOiJST0xFX1VTRVliLCJpYXQiOjE3MzExNDYyMjgslmV4cCl6MTczMTE0OTgyOH0.IIfelmXlFYmk0HuUSnhT0Xp7QoIHSMYGyDEM63VZITE`. The Body tab shows the JSON response in Pretty format:

```

1   {
2     "id": "a55fdbd05-94cc-4233-8c32-837ae3565a7a",
3     "username": "hoangpt21",
4     "email": "hoangpt21@gmail.com"
5   }

```

Hình 1.20. Test API Lấy thông tin tài khoản.

d, Tạo Category.

Request:

- URL: `http://localhost:8080/v1/job-service/category/create`
- Method: POST.
- Authorization: Bearer Token Admin.
- Body: form data: { "request": { "name": "string", "description": "string" }, "file": "string" }

Response: Ta có cấu trúc response như sau:

{ “id”: <Id category vừa tạo>, “name”: <Tên category>, “description”: <Description> }.

HTTP Test microservice / Create category

POST <http://localhost:8080/v1/job-service/category/create>

Params Authorization • Headers (11) Body • Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL

	Key	Value
<input checked="" type="checkbox"/>	request	Text <input type="button" value="▼"/> {"name": "category 1", "description": "very good"}
<input checked="" type="checkbox"/>	file	File <input type="button" value="▼"/> Topic-20-10-loài-hoa-đẹp-nhất-thế-giới-và-ý-nghĩa-...

Body Cookies (1) Headers (13) Test Results

Pretty Raw Preview Visualize **JSON**

```

1  {
2    "id": "73784dfe-b88a-4f0c-9a0e-7521da506f87",
3    "name": "category 1",
4    "description": "very good"
5  }

```

Hình 1.21. Test API Tao Category.

e, Cập nhật Category.

Request:

- URL: <http://localhost:8080/v1/job-service/category/update>
- Method: PUT.
- Authorization: Bearer Token Admin.
- Body: form data: { "request": { "id": "string", "name": "string", "description": "string" }, "file": "string" }

Response: Ta có cấu trúc response như sau:

```
{ "id": <Id category>, "name": <Tên category>, "description": <Description>, "categoryId": <Id category> }
```

The screenshot shows the Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:8080/v1/job-service/category/update
- Body (JSON):**

Key	Value
request	{"id": "858a41c3-fcb0-4437-897f-507751b06bf2", "name": "category thứ 1", "description": "nice"}
- Response Body (Pretty JSON):**

```

1  {
2    "id": "858a41c3-fcb0-4437-897f-507751b06bf2",
3    "name": "category thứ 1",
4    "description": "nice",
5    "categoryId": "858a41c3-fcb0-4437-897f-507751b06bf2"
6  }

```

Hình 1.22. Test API Cập nhật Category.

f, Tạo Job.

Request:

- URL: http://localhost:8080/v1/job-service/job/create
- Method: POST.
- Authorization: Bearer Token Admin.
- Body: form data: { "request": { "name": "string", "description": "string", "categoryId": "string", "keys": ["string"] }, "file": "string" }

Response: Ta có cấu trúc response như sau:

```
{ "id": <id job>, "name": <tên job>, "description": <description>, "categoryId": <id category>, "keys": [<mảng các keys>] }.
```

The screenshot shows the Postman interface for testing a microservice API. The URL is `http://localhost:8080/v1/job-service/job/create`. The request method is `POST`. The `Body` tab is selected, showing a `form-data` structure with two fields:

- request**: Value is a JSON object: `{"name": "job 1", "description": "very goo...", "categoryId": "858a41c3-fcb0-4437-897f-507751b06bf2", "keys": ["key1", "key2", "key3"]}`
- file**: Value is a file named `c0cea8b60af9e3a7bae8.jpg`.

The `Content-Type` is set to `application/json`. Below the body, the response is shown in `Pretty` format:

```

1  {
2    "id": "ed4845d5-80a4-41e0-a2d2-5dc9d827f87d",
3    "name": "job 1",
4    "description": "very good job",
5    "categoryId": "858a41c3-fcb0-4437-897f-507751b06bf2",
6    "keys": [
7      "key1",
8      "key2",
9      "key3"
10   ]
11 }

```

Hình 1.23. Test API Tạo job.

g, Cập nhật job.

Request:

- URL: `http://localhost:8080/v1/job-service/job/update`
- Method: PUT.
- Authorization: Bearer Token Admin.
- Body: form data: `{ "request": { "name": "string", "description": "string", "categoryId": "string", "keys": ["string"] }, "file": "string" }`

Response: Ta có cấu trúc response như sau:

```
{
  "id": <id job>, "name": <tên job>, "description": <description>,
  "adverts": [<các adverts>], "categoryId": <id category>, "keys": [<mảng các
  keys>]
}
```

The screenshot shows the Postman interface for testing a microservice. The URL is `http://localhost:8080/v1/job-service/job/update`. The method is set to `PUT`. The `Body` tab is selected, showing a `form-data` structure with one field named `request`. The value is a JSON object with fields: `"id": "07c6c933-2bdc-4cbb-8b9c-b207f5d7ce48"`, `"name": "job thứ 1"`, `"description": "good job"`, `"categoryId": "858a41c3-fcb0-4437-897f-507751b06bf2"`, `"adverts": []`, and `"keys": ["key1", "key2", "key3"]`. The `Content-Type` is set to `application/json`. Below the body, the response is shown in pretty JSON format:

```

1  {
2      "id": "07c6c933-2bdc-4cbb-8b9c-b207f5d7ce48",
3      "name": "job thứ 1",
4      "description": "good job",
5      "categoryId": "858a41c3-fcb0-4437-897f-507751b06bf2",
6      "adverts": [],
7      "keys": [
8          "key1",
9          "key2",
10         "key3"
11     ]
12 }

```

Hình 1.24. Test API Cập nhật job.

h, Lấy toàn bộ thông tin người dùng.

Request:

- URL: `http://localhost:8080/v1/user/getAll`
- Method: GET.
- Authorization: Bearer Token Admin.

Response: Ta có cấu trúc response như sau:

[<Danh sách tất cả các đối tượng người dùng>].

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/v1/user/getAll`. The response body is displayed in a JSONpretty format:

```

1  [
2   {
3     "id": "81b463f8-a647-45ff-a7d9-4c2c979d2300",
4     "username": "Hoangpt21",
5     "email": "hoangpt21@outlook.com"
6   },
7   {
8     "id": "ae574220-4640-4b7d-a87c-c12ed7716704",
9     "username": "HoangPT18",
10    "email": "HoangPT18@outlook.com"
11 }
12 ]

```

Hình 1.25. Test API Lấy toàn bộ thông tin người dùng.

i, Cập nhật thông tin người dùng.

Request:

- URL: `http://localhost:8080/v1/user/update`
- Method: PUT.
- Authorization: Bearer Token Admin or User.
- Body: form-data: { "request": { "id": "string", "username": "string", "password": "string", "userDetails": { "firstName": "string", "lastName": "string", "phoneNumber": "string", "country": "string", "city": "string", "address": "string", "postalCode": "string", "aboutMe": "string", "profilePicture": "string" } }, "file": "string" }

Response: Ta có cấu trúc response như sau:

```
{
  "id": <Id user>, "username": <tên user>, "email": <email>,
  "userDetails": { "firstName": <Tên user>, "lastName": <Họ user>,
  "phoneNumber": <Số điện thoại>, "country": <Tên quốc gia>, "city": <Tên
  thành phố>, "address": <Tên địa chỉ>, "postalCode": <mã bưu chính>,
  "aboutMe": <Mô tả bản thân>, "profilePicture": <Tài liệu ảnh> } }.
```

The screenshot shows a POST request to `http://localhost:8080/v1/user/update`. The request body is a JSON object with the following structure:

```

1 {
2   "id": "ae574220-4640-4b7d-a87c-c12ed7716704",
3   "username": "HoangPT18",
4   "email": "HoangPT18@outlook.com",
5   "userDetails": {
6     "firstName": "string",
7     "lastName": "string",
8     "phoneNumber": "string",
9     "country": "string",
10    "city": "string",
11    "address": "string",
12    "postalCode": "string",
13    "aboutMe": "string",
14    "profilePicture": "string"
15  }
16 }

```

The response status is 200 OK.

Hình 1.26. Test API Cập nhật thông tin người dùng.

j, Xóa tài khoản người dùng.

Request:

- URL: `http://localhost:8080/v1/job-service/job/deleteJobById/{id}`
- Method: DELETE.
- Authorization: Bearer Token Admin hoặc User.

Response: Ta có cấu trúc response như sau: Không có.

The screenshot shows a DELETE request to `http://localhost:8080/v1/user/deleteUserById/ae574220-4640-4b7d-a87c-c12ed7716704`. The request includes an Authorization header set to Bearer Token.

The response status is 200 OK, and the response body is empty.

Hình 1.27. Test API Xóa tài khoản người dùng.

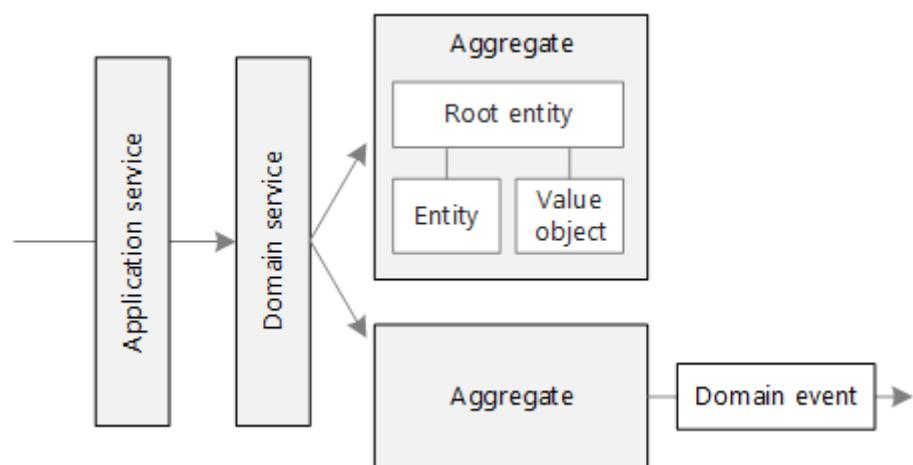
k, Kết luận về kết quả chạy chương trình.

Dưới đây là một số đánh giá kết quả chạy chương trình:

- Các containers chạy trên Docker hoạt động ổn định, không bị dừng đột ngột trong quá trình chạy chương trình.
- Các services chạy trong IDE IntelliJ hoạt động ổn định, không gây ra lỗi trong quá trình chạy chương trình.
- Liên kết giữa các services, services và containers trên Docker không bị ảnh hưởng và được giữ ổn định tạo luồng chạy mượt mà .
- Kết quả từ các bài test cho thấy chương trình đã trả về đúng với form kết quả đặt ra (cấu trúc response).

1.2.2.4. Trình bày chi ra thiết kế DDD chương trình.

Ubiquitous Language: DDD nhấn mạnh việc sử dụng một ngôn ngữ chung để tất cả các thành viên trong dự án đều hiểu rõ các khái niệm của domain. Trong dự án Spring Boot Microservices này, các thuật ngữ như "User", "Auth", "Notification", "Job" được định nghĩa và thống nhất trong toàn bộ quá trình phát triển. Điều này giúp các kỹ sư phần mềm, nhà phân tích, và các bên liên quan giao tiếp với nhau dễ dàng và chính xác, đảm bảo hiểu đúng các quy tắc và logic nghiệp vụ.



Hình 1.28. Mô hình kiến trúc Domain-Driven Design (DDD).

Bounded Contexts: Là một phạm vi rõ ràng nơi một mô hình domain cụ thể được áp dụng. Trong kiến trúc microservices, mỗi dịch vụ độc lập được coi là một Bounded Context riêng biệt. Điều này giúp chia nhỏ hệ thống thành

các phần dễ quản lý, với các quy tắc riêng biệt không bị ảnh hưởng bởi các dịch vụ khác.

Trong dự án này:

- Auth Service là một bounded context riêng, chỉ xử lý các vấn đề liên quan đến xác thực và phân quyền.
- User Service quản lý thông tin người dùng và các nghiệp vụ liên quan đến người dùng.
- Job Service chịu trách nhiệm quản lý các công việc, bao gồm logic nghiệp vụ và quy trình.
- Notification Service tập trung vào việc gửi thông báo và quản lý các phương thức liên lạc.
- Việc phân chia thành các bounded context giúp đảm bảo mỗi microservice có trách nhiệm rõ ràng và không chồng chéo chức năng với các dịch vụ khác.

Entities và Value Objects: Các Entities và Value Objects là yếu tố cơ bản trong DDD để mô hình hóa các đối tượng trong domain. Entities là các đối tượng có định danh duy nhất, tồn tại độc lập. Ví dụ, trong User Service, User là một entity với các thuộc tính như id, username, email. Value Objects không có định danh riêng và thể hiện các giá trị bất biến. Ví dụ, trong AuthService, một đối tượng JWT Token có thể được coi là một value object, chứa thông tin xác thực nhưng không có bản sắc riêng.

Domain Services và Application Services: Domain Services và Application Services giúp phân tách rõ ràng giữa logic nghiệp vụ và các chức năng ứng dụng:

- Domain Services: Chứa logic nghiệp vụ phức tạp không thuộc về một entity cụ thể. Ví dụ, trong Job Service, một dịch vụ domain có thể chịu trách nhiệm phân công công việc và thực hiện các bước xử lý nghiệp vụ cần thiết.
- Application Services: Làm nhiệm vụ điều phối giữa các domain services và các hành động từ bên ngoài. Chúng chịu trách nhiệm xử lý các

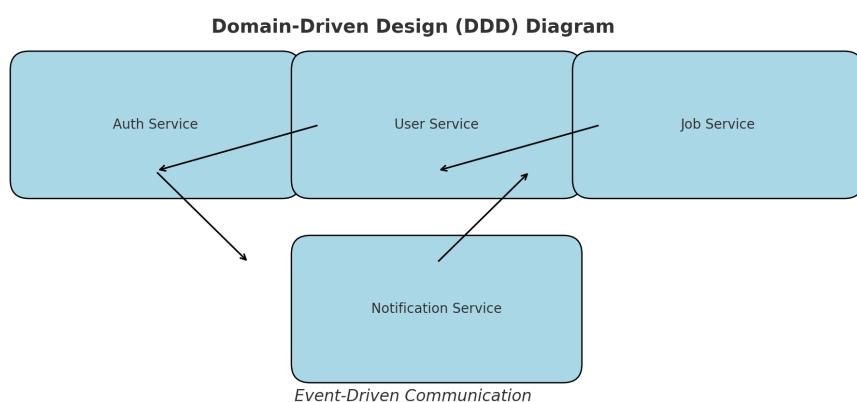
yêu cầu từ API Gateway, xác thực và chuyển tiếp chúng tới các dịch vụ domain thích hợp.

Aggregate: Aggregate là tập hợp các entity và value object được nhóm lại, tạo thành một khối logic chặt chẽ. Một aggregate có một Aggregate Root – entity chính chịu trách nhiệm điều phối và duy trì tính nhất quán trong phạm vi aggregate.

- Trong User Service, User có thể là root của một aggregate bao gồm các entity phụ như Profile, Preferences.
- Trong Auth Service, User là aggregate root khi xử lý các quyền và xác thực.

Repositories: Là lớp trung gian giữa domain và tầng cơ sở dữ liệu, được sử dụng để truy xuất và lưu trữ các entity và aggregate. Trong dự án này, mỗi microservice có một repository riêng để quản lý dữ liệu của domain tương ứng, chẳng hạn như UserRepository trong User Service hoặc JobRepository trong Job Service.

Event-Driven Communication: Dự án sử dụng Kafka để triển khai mô hình truyền thông dựa trên sự kiện, giúp các dịch vụ giao tiếp với nhau mà không bị ràng buộc chặt chẽ (loose coupling). Ví dụ, khi một người dùng đăng ký mới, Auth Service có thể phát một sự kiện qua Kafka, và Notification Service lắng nghe sự kiện đó để gửi thông báo chào mừng.



Hình 1.29. Biểu đồ Domain-Driven Design (DDD).

1.3. Triển khai với Docker Swarm hoặc Kubernetes hoặc Docker với Kubernetes.

1.3.1. Lý thuyết, kiến thức về Docker.

1.3.1.1 Khái niệm.

Docker là một công cụ đóng gói phần mềm và triển khai ứng dụng trong các container. Các ứng dụng chạy trong các container được gọi là các container Docker. Docker giúp các lập trình viên chỉ cần chạy các container lên mà không cần phải lo về việc cài các thư viện hoặc môi trường, điều này giúp các lập trình viên phát triển ứng dụng 1 cách nhanh chóng, đảm bảo tính nhất quán giữa môi trường phát triển và môi trường triển khai.

VD. Có 2 lập trình viên đang làm việc trên một dự án sử dụng framework NestJS và yêu cầu sử dụng NodeJS. Bình thường thì cả 2 lập trình viên cần phải cài cắm các thư viện cần thiết như Nodejs, npm... . Nhưng nếu bạn sử dụng MacOS và cài phiên bản Node 14.x, còn người kia dùng Windows và cài phiên bản Node 12.x, thì trường hợp này sẽ khá dễ phát sinh xung đột vì môi trường phát triển không đồng nhất. Trong trường hợp này, để tránh xung đột giữa các lập trình viên thì Docker cung cấp 1 môi trường đồng nhất giữa cả 2 bằng cách sử dụng image node:14.x, image này đã cung cấp đủ các công cụ và thư viện cần thiết để phát triển xây dựng và triển khai ứng dụng Node.js (nó giống như bạn cài Node trên máy tính cá nhân của bạn). Chính vì thế mà khi 1 người khác muốn vào phát triển ứng dụng cùng, thì họ chỉ cần chạy container lên mà không cần phải nghĩ về việc cài môi trường và các thư viện đi kèm.

1.3.1.2 Các thành phần cơ bản của Docker.

Docker daemon: Là một tiến trình nền chạy trên một máy tính Docker host (máy cài Docker), và quản lý các hoạt động Docker như tạo và quản lý các container, images, networks và volumes.

Docker client: Là một ứng dụng dòng lệnh hoặc giao diện người dùng đồ họa (GUI) để tương tác với Docker daemon và thực hiện các hoạt động Docker. Docker client sử dụng Docker API để giao tiếp với Docker daemon.

Docker images: Là một gói đóng gói của một ứng dụng và các tài nguyên cần thiết để chạy ứng dụng đó trong một container. Một image có thể được tạo từ một Dockerfile hoặc tải từ một kho chứa image trên internet như Docker Hub.

Docker container: Là một môi trường đóng gói độc lập, chứa tất cả các thành phần cần thiết để chạy một ứng dụng trong một môi trường cô lập, container được tạo ra từ image sau khi đã đóng gói.

Docker network: cho phép các container tương tác với nhau và với các dịch vụ khác. Một Docker network được tạo ra để tạo một mạng ảo cho các container chạy trên cùng một máy Docker host.

Docker volume: Cho phép các container lưu trữ và truy cập dữ liệu được sử dụng bởi các ứng dụng. Volume giúp dữ liệu được bảo vệ và giữ cho đồng bộ giữa các container.

1.3.1.3 Cơ chế hoạt động.

Docker hoạt động thông qua một Docker engine, kết hợp 2 yếu tố:

- 1 server và 1 client.
- Giao tiếp giữa server và client thông qua REST API.

Trên hệ thống Window/Mac cũ, bạn có thể tận dụng Docker toolbox, nó cho phép điều khiển Docker engine bằng cách sử dụng Compose và Kitematic.

1.3.2. Hướng dẫn đóng gói dự án bằng docker.

Đối với dự án code bằng Java Spring boot sẽ có 2 cách đóng gói dự án bằng docker

1.3.2.1 Đóng gói PostgreSQL.

Một cơ sở dữ liệu PostgreSQL được chạy trong một container Docker. Nó được cấu hình với các biến môi trường để xác định người dùng, mật khẩu

và cơ sở dữ liệu. Nó cũng sử dụng một volume để lưu trữ dữ liệu của cơ sở dữ liệu, đảm bảo dữ liệu không bị mất khi container được khởi động lại.

```
postgres:
  container_name: postgres
  image: postgres:latest
  environment:
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: 55
    POSTGRES_DB: microservice
  volumes:
    - postgres_data:/var/lib/postgresql/data
  ports:
    - "5432:5432"
```

Hình 1.30. Cấu hình Postgres.

Trong đó:

- container_name: postgres: Tên của container PostgreSQL.
- image: postgres:latest: Sử dụng hình ảnh PostgreSQL chính thức từ Docker Hub, phiên bản mới nhất.
- environment: Thiết lập các biến môi trường cho container:
 - + PostgreSQL: POSTGRES_USER: postgres: Tên người dùng của cơ sở dữ liệu.
 - + POSTGRES_PASSWORD: 55: Mật khẩu của cơ sở dữ liệu.
 - + POSTGRES_DB: microservice: Tên cơ sở dữ liệu mặc định.
- volumes: postgres_data:/var/lib/postgresql/data: Liên kết một volume với thư mục /var/lib/postgresql/data trong container. Điều này sẽ lưu trữ dữ liệu của PostgreSQL vào một thư mục trên máy chủ, đảm bảo dữ liệu không bị mất khi container được khởi động lại.
- ports: 5432:5432: Ánh xạ port 5432 của container PostgreSQL đến port 5432 trên máy chủ. Điều này cho phép bạn kết nối đến PostgreSQL từ máy chủ.

1.3.2.2 Đóng gói ZooKeeper.

Một máy chủ ZooKeeper được chạy trong một container Docker. Nó được sử dụng để quản lý các metadata của Kafka và cung cấp khả năng đồng bộ hóa. Nó cũng sử dụng một volume để lưu trữ dữ liệu.

```
zookeeper:
  container_name: zookeeper
  image: "docker.io/bitnami/zookeeper:3"
  ports:
    - "2181:2181"
  volumes:
    - "zookeeper_data:/bitnami"
  environment:
    - ALLOW_ANONYMOUS_LOGIN=yes
```

Hình 1.31. Cấu hình Zookeeper.

Trong đó:

- container_name: zookeeper: Tên của container ZooKeeper.
- image: docker.io/bitnami/zookeeper:3: Sử dụng hình ảnh ZooKeeper chính thức từ Docker Hub, phiên bản 3.
 - ports: 2181:2181: Ánh xạ port 2181 của container ZooKeeper đến port 2181 trên máy chủ. Điều này cho phép bạn kết nối đến ZooKeeper từ máy chủ.
 - volumes: zookeeper_data:/bitnami: Liên kết một volume với thư mục /bitnami trong container. Điều này sẽ lưu trữ dữ liệu của ZooKeeper vào một thư mục trên máy chủ, đảm bảo dữ liệu không bị mất khi container được khởi động lại.
 - environment: ALLOW_ANONYMOUS_LOGIN=yes: Cho phép kết nối ẩn danh đến ZooKeeper.

1.3.2.3 Đóng gói Kafka.

Một máy chủ Kafka được chạy trong một container Docker. Nó được cấu hình để kết nối với máy chủ ZooKeeper và sử dụng một volume để lưu

trữ dữ liệu. Nó cũng được cấu hình để cho phép kết nối plain text và sử dụng hai listeners (INSIDE và OUTSIDE) để kết nối đến các service khác trong cùng một mạng với docker-compose và từ bên ngoài.

```

kafka:
  container_name: kafka
  image: "docker.io/bitnami/kafka:2-debian-10"
  ports:
    - "9092:9092"
  expose:
    - "9093"
  volumes:
    - "kafka_data:/bitnami"
  environment:
    - KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181
    - ALLOW_PLAINTEXT_LISTENER=yes
    - KAFKA_ADVERTISED_LISTENERS=INSIDE://kafka:9093,OUTSIDE://localhost:9092
    - KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT
    - KAFKA_LISTENERS=INSIDE://0.0.0.0:9093,OUTSIDE://0.0.0.0:9092
    - KAFKA_INTER_BROKER_LISTENER_NAME=INSIDE
    - KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181
  depends_on:
    - zookeeper

```

Hình 1.32. Cấu hình Kafka.

Trong đó:

- container_name: kafka: Tên của container Kafka.
- image: docker.io/bitnami/kafka:2-debian-10: Sử dụng hình ảnh Kafka chính thức từ Docker Hub, phiên bản 2-debian-10.
- ports: 9092:9092: Ánh xạ port 9092 của container Kafka đến port 9092 trên máy chủ.
- expose: 9093: Kafka cũng expose port 9093 bên trong mạng docker-compose cho các service khác.
- volumes: kafka_data:/bitnami: Liên kết một volume với thư mục /bitnami trong container. Điều này sẽ lưu trữ dữ liệu của Kafka vào một thư mục trên máy chủ, đảm bảo dữ liệu không bị mất khi container được khởi động lại.
- environment:

- + KAFKA_CFG_ZOOKEEPER_CONNECT=zookeeper:2181: Thiết lập địa chỉ của máy chủ ZooKeeper để Kafka kết nối.
- + ALLOW_PLAINTEXT_LISTENER=yes: Cho phép kết nối plain text đến Kafka.
- +KAFKA_ADVERTISED_LISTENERS=INSIDE://kafka:9093,OUTSIDE://localhost:9092: Thiết lập địa chỉ cho các service khác kết nối đến Kafka.
- +KAFKA_LISTENER_SECURITY_PROTOCOL_MAP=INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT: Thiết lập giao thức bảo mật cho các listener.
- +KAFKA_LISTENERS=INSIDE://0.0.0.0:9093,OUTSIDE://0.0.0.0:9092: Thiết lập các listener của Kafka.
- + KAFKA_INTER_BROKER_LISTENER_NAME=INSIDE: Thiết lập tên listener được sử dụng bởi các broker Kafka.
- + KAFKA_ZOOKEEPER_CONNECT=zookeeper:2181: Thiết lập địa chỉ của máy chủ ZooKeeper để Kafka kết nối.
 - depends_on: - zookeeper: Kafka phụ thuộc vào ZooKeeper, do đó nó sẽ được khởi động sau khi ZooKeeper được khởi động.

1.3.2.4 Đóng gói Kafka-ui.

Một ứng dụng web được chạy trong một container Docker. Nó cung cấp một giao diện người dùng để quản lý và theo dõi Kafka. Nó được cấu hình để kết nối với Kafka và ZooKeeper.

```
kafka-ui:
  container_name: kafka-ui
  image: provectuslabs/kafka-ui
  ports:
    - "9090:8080"
  restart: always
  environment:
    - KAFKA_CLUSTERS_0_NAME=local
    - KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9093
    - KAFKA_CLUSTERS_0_ZOOKEEPER=localhost:2181
```

Hình 1.33. Cấu hình Kafka-ui.

Trong đó:

- container_name: kafka-ui: Tên của container Kafka UI.

- image: provectuslabs/kafka-ui: Sử dụng hình ảnh Kafka UI từ Docker Hub.

- ports: 9090:8080: Ánh xạ port 8080 của container Kafka UI đến port 9090 trên máy chủ. restart:

- always: Kafka UI sẽ tự động khởi động lại nếu nó bị dừng hoặc lỗi.

- environment:

+ KAFKA_CLUSTERS_0_NAME=local: Thiết lập tên của cluster Kafka.

+KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=kafka:9093: Thiết lập địa chỉ của máy chủ Kafka để Kafka UI kết nối.

+ KAFKA_CLUSTERS_0_ZOOKEEPER=localhost:2181: Thiết lập địa chỉ của máy chủ ZooKeeper để Kafka UI kết nối.

CHƯƠNG 2

PHÁT HIỆN ẢNH ĐÃ QUA CHỈNH SỬA ÚNG DỤNG CNN, ELA VÀ KERAS

2.1. Tổng quan bài toán.

2.1.1 Giới thiệu bài toán.

Trong thời đại kỹ nguyên số hiện nay, việc chỉnh sửa ảnh trở nên phổ biến và dễ dàng với các công cụ như Adobe Photoshop, GIMP, và các phần mềm di động khác. Các ảnh đã qua chỉnh sửa có thể làm thay đổi độ chính xác của thông tin được truyền tải và gây ra nhiều vấn đề nghiêm trọng, như:

- Gây hiệu ứng nhạy cảm sai lệch trong các tài liệu truyền thông.
- Gian lận trong các cuộc thi nhằm mục đích thu lợi bất chính.
- Tạo bạo chứng giả trong các vụ án tố tụng.

Việc phát hiện xem một bức ảnh đã bị chỉnh sửa hay chưa trở thành một yêu cầu bắt buộc trong các ứng dụng thực tiễn như truyền thông, công lý, an ninh mạng và kiểm duyệt nội dung. Thông qua đó là lý do để tài ra đời nhằm ứng dụng thực tiễn giúp bảo mật, an toàn trong thông tin, tránh giả mạo và gian lận trong thời đại AI, công nghệ số phát triển ngày nay.

Đề tài là sự kết hợp của các công nghệ sau :

- **Mạng nơ-ron tích chập (CNN):** CNN được sử dụng để phát hiện các đặc điểm phức tạp trong hình ảnh và học các mẫu phân biệt giữa ảnh thật và ảnh đã qua chỉnh sửa.

- **Phân tích mức năng lượng lõi (ELA):** Kỹ thuật này giúp tìm kiếm sự khác biệt nhỏ trong các khu vực ảnh dễ bị bóc lõi trong quá trình chỉnh sửa.

- **Keras:** Framework Python này giúp xây dựng và huấn luyện mô hình CNN một cách nhanh chóng và hiệu quả.

Việc phát triển các hệ thống nhận diện ảnh đã qua chỉnh sửa có thể được ứng dụng trong nhiều lĩnh vực thực tế như:

- **Báo chí và truyền thông:** Giúp phát hiện và kiểm duyệt các nội dung hình ảnh tránh việc đánh lừa người dùng.

- **An ninh mạng:** Giúp phân tích các ảnh được tải lên hệ thống, ngăn ngừa các hành vi gian lận.

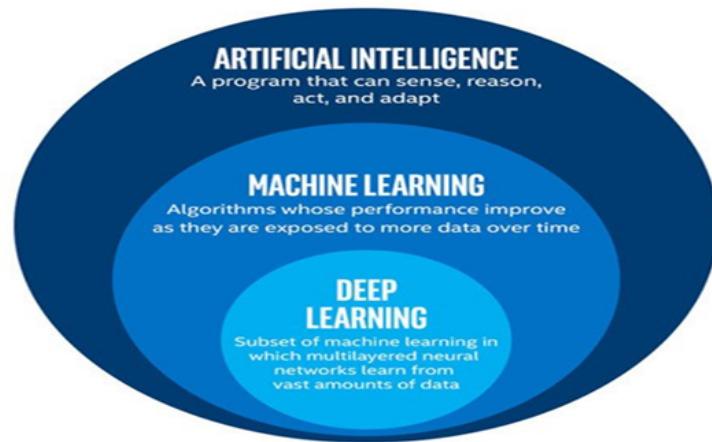
- **Pháp luật:** Tăng cường bằng chứng đối với các vụ án có yếu tố hình ảnh.

2.1.2. Cơ sở lý thuyết.

2.1.2.1 Trí tuệ nhân tạo và học máy.

Trong những năm vừa qua, cùng với sự bùng nổ của cuộc cách mạng công nghiệp 4.0, các thuật ngữ như trí tuệ nhân tạo (AI), học máy (machine learning) và học sâu (deep learning) đang dần trở nên phổ biến và trở thành những khái niệm mà các công dân của kỷ nguyên 4.0 buộc phải nắm được.

Có thể giải thích mối liên hệ giữa 3 khái niệm này bằng cách tưởng tượng chúng như những vòng tròn, trong đó AI - ý tưởng xuất hiện sớm nhất - là vòng tròn lớn nhất, tiếp đến là machine learning - khái niệm xuất hiện sau, và cuối cùng là deep learning - thứ đang thúc đẩy sự bùng phát của AI hiện nay - là vòng tròn nhỏ nhất.



Hình 2.1. Vòng tròn mối liên kết giữa 3 khái niệm.

Machine Learning(Học Máy) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể.

Học máy dựa trên ý tưởng rằng các hệ thống có thể học hỏi từ dữ liệu và cải thiện hiệu suất của chúng theo thời gian. Điều này trái ngược với lập

trình truyền thông, trong đó các nhà phát triển cần viết mã để chỉ định cho hệ thống cách thực hiện một nhiệm vụ cụ thể.

Quy trình hoạt động của Machine Learning:

- Thu thập dữ liệu (Data Collection): Đây là công việc quan trọng và tiêu tốn nhiều thời gian nhất trong cả quá trình. Để máy móc hiểu và giải quyết vấn đề, chúng ta cần cung cấp cho chúng các tệp dữ liệu. Chất lượng dữ liệu thu được sẽ ảnh hưởng trực tiếp đến kết quả của Machine Learning. Vì thế dữ liệu thu được phải đảm bảo độ chính xác, mức độ đáng tin cậy cao để tránh làm kết quả dự đoán sai sót.

- Tiên xử lý dữ liệu (Data Processing): Lượng lớn dữ liệu được thu về chắc chắn sẽ có những thuộc tính không cần dùng đến, thông tin bị thiếu/thừa,... và công việc của chúng ta là loại bỏ những thành phần này. Việc chuẩn hóa dữ liệu sẽ giúp máy tính học nhanh hơn và hiệu quả học tập mang lại cao hơn, ít sai sót hơn.

- Huấn luyện cho mô hình (Training Model): Sau khi có được tập dữ liệu “sạch”, chúng ta sẽ bắt đầu truyền dữ liệu cho mô hình học máy. Sau quá trình học tập, kết hợp các dữ liệu, máy tính sẽ có khả năng đưa ra các dự đoán hoặc hoàn thành nhiệm vụ đã đề ra.

- Đánh giá mô hình (Evaluate Model): Sau quá trình đào tạo chúng ta cần kiểm tra kết quả hiệu quả học tập của máy tính như thế nào. Chúng ta có thể kiểm tra thông qua các bài toán với dữ liệu chưa từng cung cấp cho máy tính trước đó. Nếu thử nghiệm được thực hiện trên cùng một dữ liệu đã được sử dụng để đào tạo, chúng ta sẽ không có được thước đo chính xác, vì mô hình có thể tìm thấy các mẫu giống nhau trong tập dữ liệu trước đây. Điều này sẽ cung cấp cho chúng ta độ chính xác cao không tương xứng.

- Cải thiện hiệu quả (Improve): Quá trình đánh giá kết thúc chúng ta sẽ nhận thấy những lỗi của mô hình. Ta cần tìm cách khắc phục lỗi đó bằng cách huấn luyện lại. Như vậy công việc huấn luyện và đánh giá sẽ được thực hiện xoay vòng liên tục cho đến khi mô hình cho ra kết quả có độ chính xác cao.

2.1.2.2. Mạng nơ-ron tích chập CNN.

Thuật toán Convolutional neural network còn gọi là Mạng nơ-ron tích chập, thường được viết tắt là CNN. Đây là một trong những mô hình của Deep Learning (Deep learning là tập hợp các thuật toán để cố gắng mô hình dữ liệu trừu tượng hóa ở mức cao bằng cách sử dụng nhiều lớp xử lý với cấu trúc phức tạp hoặc bằng cách khác).

Mạng nơ-ron tích chập được ứng dụng phổ biến trong nhiều nghiên cứu, đặc biệt liên quan đến xử lý hình ảnh như nhận diện đối tượng, nhận diện khuôn mặt, Phân loại ảnh, dự đoán chuỗi thời gian, xử lý ngôn ngữ tự nhiên, xử lý video và chuỗi hình ảnh và còn nhiều ứng dụng khác nữa.

Hiểu đơn giản, CNN cũng chính là một dạng Artificial Neural Network nhưng mang thêm 1 vài cải tiến. Mạng nơ-ron tích chập gồm 4 lớp chính là:

- Lớp tích chập (Convolution) sử dụng bộ lọc (kernel) trượt trên một ma trận để trích xuất các đặc điểm của ảnh, cụ thể ở đây là ma trận pixel của ảnh như mô tả hình dưới.

7	2	3	3	8
4	5	3	8	4
3	3	2	8	4
2	8	7	2	7
5	4	4	5	4

*

1	0	-1
1	0	-1
1	0	-1

=

6		

$$\begin{aligned}
 & 7 \times 1 + 4 \times 1 + 3 \times 1 + \\
 & 2 \times 0 + 5 \times 0 + 3 \times 0 + \\
 & 3 \times -1 + 3 \times -1 + 2 \times -1 \\
 = & 6
 \end{aligned}$$

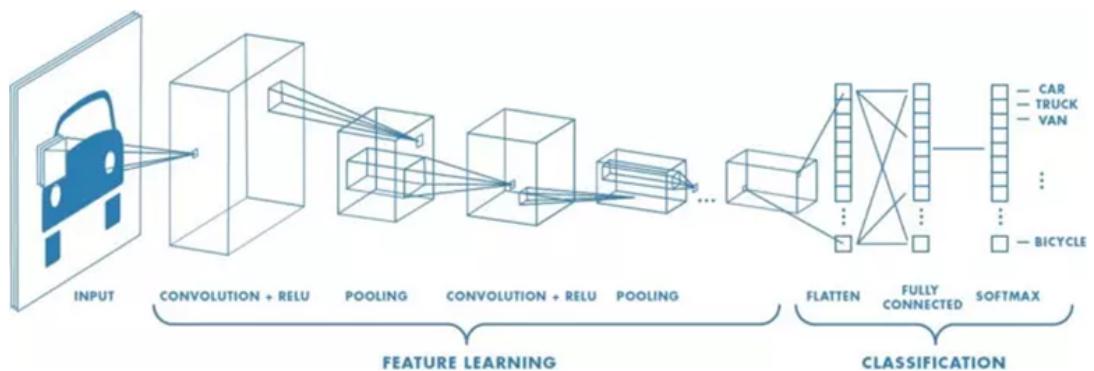
Hình 2.2. Ví dụ Convolution.

- Lớp hội tụ (Pooling) là lớp có mục đích giảm kích thước của dữ liệu bằng cách tính giá trị trung bình hoặc tối đa của mỗi vùng dữ liệu, mà vẫn giữ được các thông tin quan trọng của dữ liệu gốc.

- Lớp kích hoạt (Activation) sử dụng hàm hàm phi tuyến để thêm tính phi tuyến vào mô hình giúp mô hình học được các quan hệ phức tạp giữa các đặc điểm, phổ biến là ReLU, sigmoid và tanh.

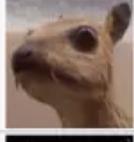
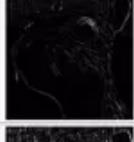
- Lớp kết nối đầy đủ (Fully connected) lớp kết nối tất cả các nơ-ron sau cùng thành một vector xác suất.

Nguyên tắc hoạt động của mô hình CNN: Về kỹ thuật, mô hình CNN để huấn luyện và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc, tổng hợp lại các lớp được kết nối đầy đủ và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.



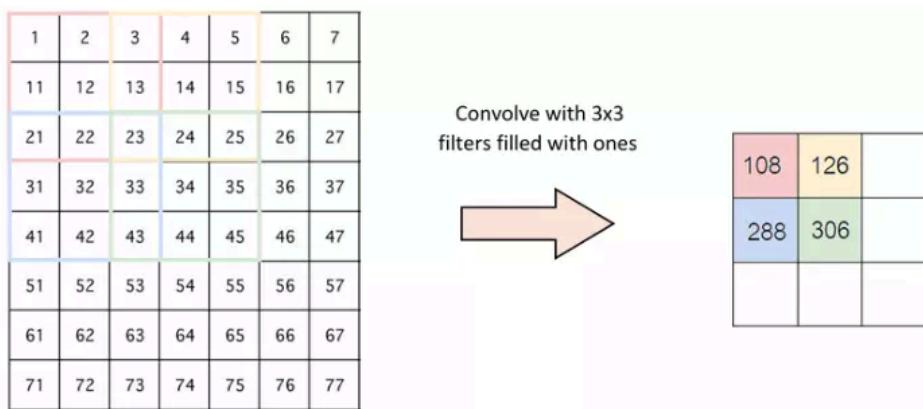
Hình 2.3. Ví dụ model CNN.

Lớp tích chập (Convolution) là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân. Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các bộ lọc khác nhau.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

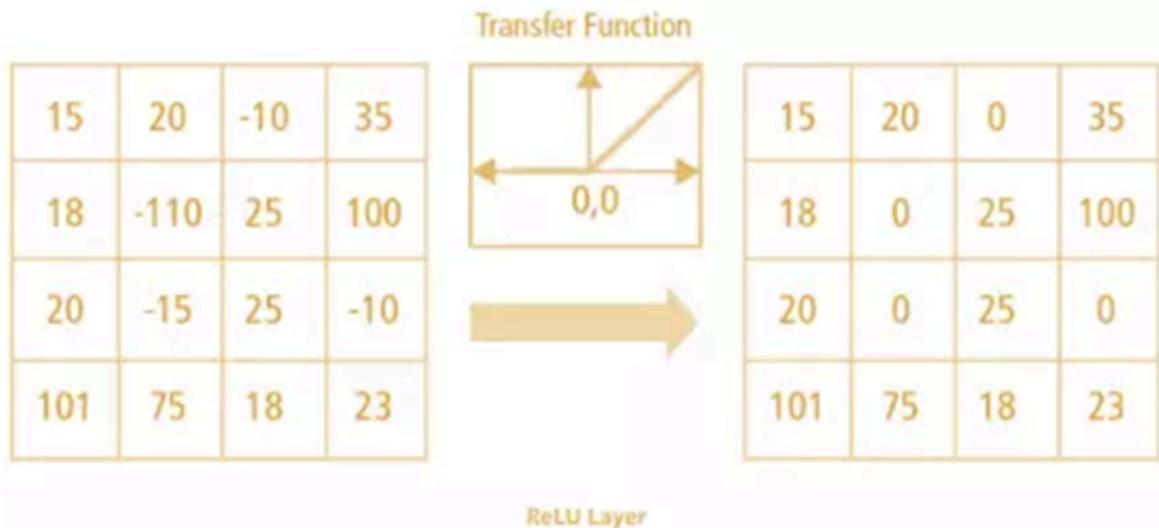
Hình 2.4. Ví dụ các bộ lọc.

Bước nhảy (Stride) là số pixel thay đổi trên ma trận đầu vào. Khi bước nhảy là 1 thì ta di chuyển lớp tích chập 1 pixel. Khi stride là 2 thì ta di chuyển các lớp tích chập đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride là 2.



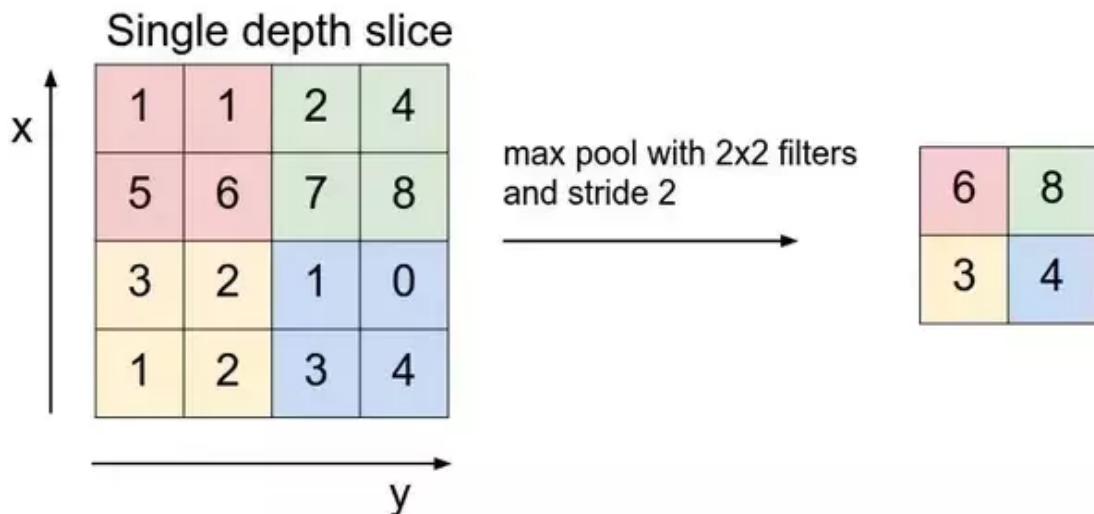
Hình 2.5. Ví dụ bước nhảy là 2.

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: $f(x) = \max(0, x)$. Hiểu đơn giản mục đích của hàm ReLU là để giới hạn miền giá trị của ma trận pixel là các số không âm. Nó quan trọng là vì tập dữ liệu số là các giá trị dương.



Hình 2.6. Ví dụ hàm ReLU.

Lớp pooling thường được sử dụng ngay sau lớp tích chập để đơn giản hóa thông tin đầu ra và giảm bớt kích thước ma trận mà vẫn giữ lại các thông tin quan trọng. Dưới đây là ví dụ về Max pooling 2x2.



Hình 2.7. Ví dụ Max pooling 2x2.

2.1.2.3 Kỹ thuật phân tích mức độ lỗi (ELA).

Trong thế giới kỹ thuật số ngày càng phát triển, phân tích mức độ lỗi (ELA) đã nổi lên như một công cụ mạnh mẽ để phát hiện sự không nhất quán và khả năng thao túng trong hình ảnh kỹ thuật số.

ELA là viết tắt của Error Level Analysis, là phân tích các thành phần nén trong dữ liệu số có tính năng nén bị mất dữ liệu như JPEG hay còn hiểu là một phương pháp phân tích ảnh được sử dụng để phát hiện sự thay đổi không mong muốn trong chất lượng hình ảnh, thường là dấu hiệu của sự chỉnh sửa hoặc làm giả.

ELA hoạt động bằng cách phân tích mức độ nén để phát hiện những điểm không nhất quán có thể cho thấy sự thao túng hoặc chỉnh sửa. ELA so sánh các mức độ lỗi (error level) trong một ảnh. Mỗi lần lưu ảnh lại với chất lượng JPEG, mức độ lỗi sẽ tăng lên. Khi một hình ảnh được chỉnh sửa bằng phần mềm (như Photoshop) hoặc tạo ra những thay đổi không mong muốn, vùng chỉnh sửa thường sẽ có mức độ lỗi khác biệt so với phần còn lại của ảnh.

Error Level Analysis (ELA) có một số ứng dụng quan trọng trong lĩnh vực phân tích ảnh và pháp lý. Dưới đây là một số ứng dụng chính của ELA:

- Trong phát hiện sự chỉnh sửa hình ảnh: ELA được sử dụng để phát hiện các vùng của hình ảnh đã được chỉnh sửa hoặc làm giả. Các vùng có mức độ lỗi cao hơn so với các vùng khác có thể chỉ ra sự thay đổi không mong muốn hoặc các kỹ thuật làm giả trong hình ảnh.

- Trong pháp luật:

- + Phân tích ảnh kỹ thuật số trong pháp luật: ELA hữu ích trong việc phân tích ảnh kỹ thuật số trong các cuộc điều tra pháp lý, giúp phát hiện sự sửa đổi không mong muốn trong bằng chứng số.

- + Phân tích tội phạm trên mạng: Trong lĩnh vực an ninh mạng, ELA được sử dụng để phát hiện sự sửa đổi không mong muốn trong hình ảnh, như trong các vụ việc liên quan đến lừa đảo, tấn công mạng hoặc tội phạm trực tuyến.

+ Xác minh hình ảnh trên mạng: Trong việc xác minh tính xác thực của hình ảnh trên mạng, đặc biệt là trong các trường hợp mà sự chân thực của hình ảnh đó là quan trọng (ví dụ: tin tức, chứng minh sự kiện), ELA có thể giúp phát hiện ra các hình ảnh đã được chỉnh sửa hoặc làm giả.

Nguyên tắc hoạt động của ELA: Về cốt lõi, ELA hoạt động bằng cách phân tích mức độ nén của hình ảnh kỹ thuật số.

Nguyên tắc hoạt động ELA dựa trên cách mà chất lượng hình ảnh JPEG thay đổi mỗi khi hình ảnh được lưu lại với mức độ nén khác nhau. ELA so sánh các vùng của hình ảnh với nhau để phát hiện sự thay đổi trong chất lượng và cấu trúc của ảnh.

Các bước của nguyên tắc hoạt động ELA:

- Bước 1: Lưu ảnh với chất lượng JPEG khác nhau: Ảnh gốc được lưu lại với chất lượng JPEG ban đầu. Sau đó, ảnh này được lưu lại nhiều lần với các mức độ nén khác nhau, tạo ra các bản sao của ảnh với chất lượng khác nhau.

- Bước 2: So sánh các phiên bản ảnh với nhau: Các phiên bản ảnh được so sánh với nhau để xác định sự khác biệt giữa chúng.

- Bước 3: Phân tích mức độ lỗi: ELA tìm kiếm các vùng trong ảnh có mức độ lỗi khác biệt so với các vùng khác. Mức độ lỗi thường tăng lên ở các vùng đã được chỉnh sửa hoặc làm giả, trong khi các vùng không bị ảnh hưởng nhiều thì có mức độ lỗi thấp hơn.

- Bước 4: Xác định các vùng có thay đổi đáng kể: Dựa trên sự khác biệt trong mức độ lỗi, ELA có thể giúp xác định các vùng của hình ảnh đã được chỉnh sửa hoặc làm giả, thông qua việc xác định các vùng có mức độ lỗi cao hơn so với các vùng khác.

- Bước 5: Đánh giá kết quả: Kết quả của ELA cung cấp thông tin hữu ích để đánh giá tính chân thực của hình ảnh và phát hiện sự thay đổi không mong muốn.

Ví dụ: Phân tích mức độ lỗi cho thấy các mức độ lỗi khác nhau trong toàn bộ hình ảnh này, gợi ý rõ ràng về một số hình thức thao túng kỹ thuật số.



Hình 2.8. Ảnh gốc.

Các khu vực cần lưu ý là môi và áo cũng như mắt. Tất cả đều có mức độ lỗi khác biệt đáng kể so với môi trường xung quanh. Có lẽ, màu sắc đã được thay đổi và các vùng sáng lên.



Hình 2.9. Hình ảnh ELA.

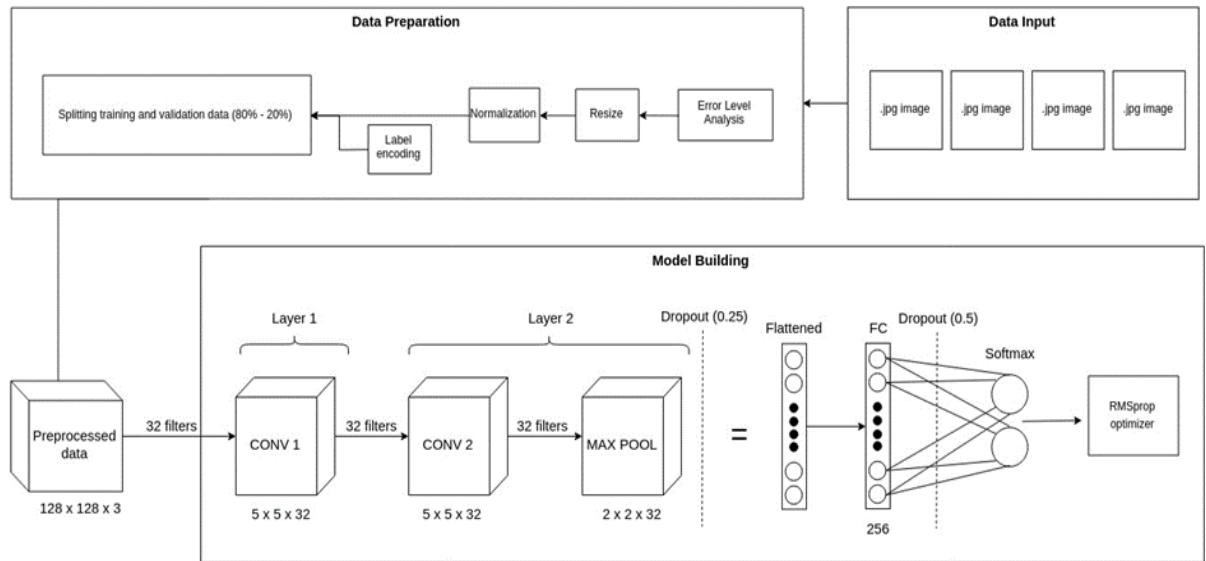
Giải thích: Phân tích mức lỗi (ELA) hoạt động bằng cách cố tình lưu lại hình ảnh ở tỷ lệ lỗi đã biết, chẳng hạn như 95%, sau đó tính toán sự khác biệt giữa các hình ảnh. Nếu hầu như không có thay đổi nào thì ô đã đạt đến mức tối thiểu cục bộ về lỗi ở mức chất lượng đó. Tuy nhiên, nếu có nhiều thay đổi thì các pixel không ở mức tối thiểu cục bộ và thực sự là nguyên bản. Kết quả là hình ảnh ELA đã được tăng cường độ sáng để phân biệt rõ hơn những khác biệt.

2.2. Phân tích cấu trúc bài toán.

Để tổng hợp thông tin thu được từ các ước lượng chứng cứ, các thuật

toán phát hiện giả mạo hiện tại sử dụng nhiều phương pháp mô tả. Tuy nhiên, mỗi kỹ thuật mô tả đều có những hạn chế và khuyết điểm riêng. Tầm quan trọng của mạng nơ-ron tích chập (CNN) trong phân loại hình ảnh và các ứng dụng thị giác máy tính đã bùng nổ trong những năm gần đây. Vì chưa thông tin về hình ảnh như màu sắc và các yếu tố cấu trúc, mạng nơ-ron truyền thống sử dụng hình ảnh gốc trong các kênh RGB làm đầu vào. Những ứng dụng mới và có giá trị đã được thực hiện nhờ những đột phá gần đây trong kỹ thuật xử lý và cải thiện hình ảnh số. Do đó, nhóm chúng em đã nghiên cứu các hình thức tạo ra và phát hiện hình ảnh khác nhau, với trọng tâm đặc biệt vào các hệ thống phát hiện biến dạng hình ảnh dựa trên pixel. Một lượng lớn dữ liệu hình ảnh phải được xử lý để xác định hình ảnh giả mạo, và một mô hình kiến trúc có thể xử lý nhiều pixel trong hình ảnh. Ngoài dự án, dữ liệu huấn luyện phải hiệu quả và linh hoạt để hỗ trợ việc sử dụng trong cuộc sống hàng ngày. Do đó, kiến trúc Mạng Nơ-ron Tích chập (CNN), sử dụng Phân tích Mức lỗi (ELA), được sử dụng trong báo cáo của nhóm chúng em. Hình ảnh được phân tích ở hai cấp độ trong nghiên cứu này. Nó kiểm tra metadata của hình ảnh ở cấp độ đầu tiên. Vì metadata hình ảnh có thể được thay đổi bằng các công cụ đơn giản, nó không rất đáng tin cậy. Tuy nhiên, hầu hết các hình ảnh sẽ có metadata không bị thay đổi, điều này sẽ giúp xác định các thay đổi. Nếu một hình ảnh được chỉnh sửa bằng Adobe Photoshop, ví dụ, metadata sẽ bao gồm phiên bản Adobe Photoshop được sử dụng. Hình ảnh được chuyển đổi thành định dạng phân tích mức lỗi và thu nhỏ thành hình ảnh 100px x 100px ở cấp độ thứ hai. Sau đó, 10.000 pixel có giá trị RGB này (30.000 đầu vào) được đưa vào lớp đầu vào của mạng perceptron đa lớp. Nhóm em quyết định xem hình ảnh có bị biến dạng hay không dựa trên giá trị của các đầu ra nơ-ron này, cũng như đầu ra của trình phân tích metadata, và xác suất hình ảnh bị biến đổi là bao nhiêu.

Cụ thể về sự kết hợp giữa CNN và ELA trong đề tài này, ta có thể xem hình dưới đây:



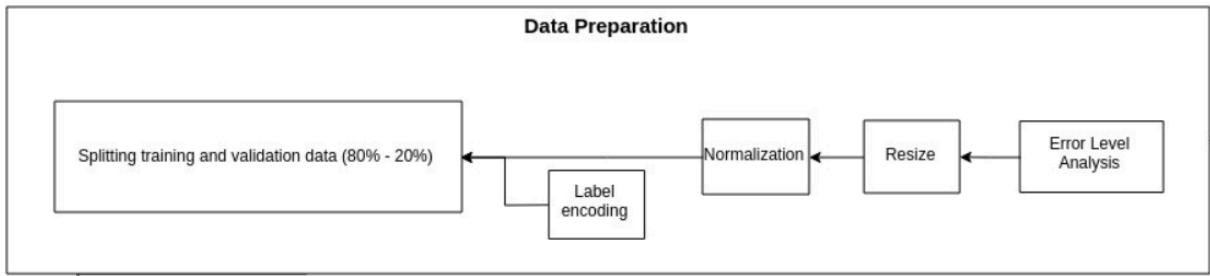
Hình 2.10. Sơ đồ quá trình xử lý ảnh và huấn luyện model.

Hình trên thể hiện kiến trúc của chương trình thực hiện xây dựng mô hình mạng nơron tích chập với dữ liệu ảnh đầu vào đã qua xử lý.



Hình 2.11. Tập dữ liệu đầu vào.

Với khối hình đầu tiên là tập dữ liệu đầu vào, hệ thống sẽ nhận đầu vào là các hình ảnh “.jpg”. Tập dữ liệu này thường có số lượng lớn hoặc rất lớn các hình ảnh với đủ thể loại đa dạng để đảm bảo mô hình huấn luyện có tính khách quan để ngày càng thông minh hơn. Tuy nhiên, đây là các dữ liệu cần được tiền xử lý trước khi đưa vào mô hình học máy CNN.



Hình 2.12. Quá trình chuẩn bị dữ liệu.

Quá trình chuẩn bị dữ liệu hay quá trình tiền xử lý dữ liệu trong học máy là một bước quan trọng và cần thiết để đảm bảo chất lượng và hiệu suất của mô hình. Trong phần này ta sẽ đi thực hiện tiền xử lý các hình ảnh trong tập dữ liệu ban đầu. Dữ liệu ảnh thô được tiền xử lý thông qua các bước như phân tích mức độ lỗi, thay đổi kích thước, chuẩn hóa, mã hóa nhãn và chia dữ liệu huấn luyện và kiểm định (80% - 20%). Cụ thể các bước đó như sau:

- Phân tích mức độ lỗi (Error Level Analysis): Đây là bước đầu tiên, nơi mà dữ liệu hình ảnh được kiểm tra để xác định và phân loại các lỗi có thể xuất hiện. Quá trình này giúp đánh giá chất lượng của dữ liệu hình ảnh và xác định các vấn đề có thể ảnh hưởng đến hiệu suất của mô hình. Kết quả là ta thu được các hình ảnh Error Level Analysis từ một hình ảnh gốc.



Hình 2.13. Hình ảnh gốc và hình ảnh ELA tương ứng.

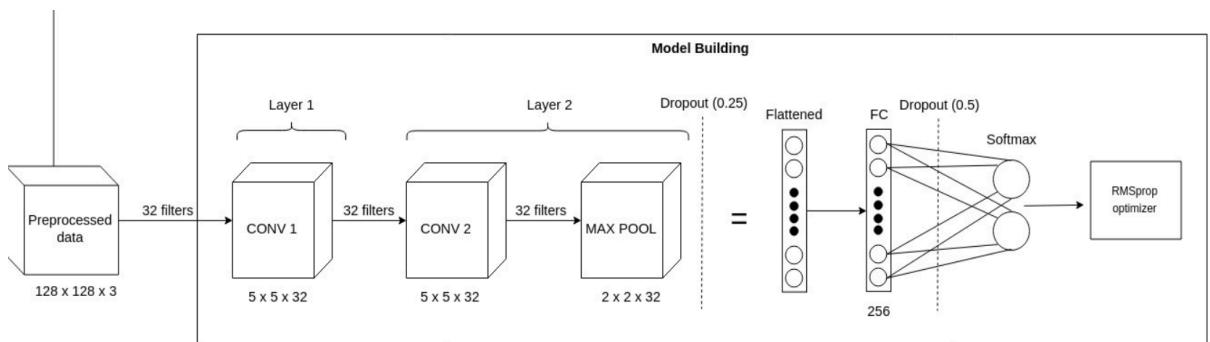
- Thay đổi kích thước (Resize): Dữ liệu hình ảnh sau đó được điều chỉnh kích thước để phù hợp với yêu cầu của mô hình hoặc để tối ưu hóa việc xử lý. Đối với dữ liệu hình ảnh, việc thay đổi kích thước hình ảnh giúp đảm bảo rằng tất cả hình ảnh đều có cùng kích thước khi đưa vào mô hình.

- Chuẩn hóa (Normalization): Dữ liệu hình ảnh thường có sự biến đổi lớn về đơn vị và phạm vi giá trị. Việc chuẩn hóa giúp đưa dữ liệu về cùng một thang đo, thường là từ 0 đến 1, giúp quá trình học của mô hình diễn ra nhanh và ổn định hơn.

Mã hóa nhãn (Label Encoding): Nhãn của dữ liệu hình ảnh thường là dạng văn bản hoặc danh mục, do đó cần được chuyển đổi thành dạng số để máy tính có thể hiểu và xử lý. Quá trình này được gọi là mã hóa nhãn.

Chia dữ liệu huấn luyện và kiểm định (Splitting Training and Validation Data): Dữ liệu cuối cùng sẽ được chia thành hai tập, 80% cho huấn luyện và 20% cho kiểm định. Việc chia dữ liệu như vậy giúp đánh giá mô hình một cách công bằng và tránh hiện tượng quá khớp (overfitting).

Dữ liệu đầu vào đã qua tiền xử lý có kích thước “128 x 128 x 3”, nghĩa là hình ảnh có độ phân giải 128x128 pixel và có ba kênh màu là đỏ, xanh lá cây và xanh dương. Tập dữ liệu được chia 80% tiếp tục được đưa vào mô hình phục vụ cho việc huấn luyện.



Hình 2.14. Quá trình huấn luyện mô hình.

Ở hình 2.14 ta thực hiện huấn luyện một mô hình CNN. Dữ liệu huấn luyện sẽ đi qua tuần tự các lớp trong mô hình, cụ thể:

- Conv 1 và Conv 2: Đây là hai lớp tích chập đầu tiên trong mô hình, mỗi lớp sử dụng 32 bộ lọc để trích xuất đặc trưng từ dữ liệu hình ảnh đầu vào. Các bộ lọc này di chuyển qua hình ảnh đầu vào để học các đặc trưng cục bộ.

- Max Pool: Sau hai lớp tích chập, dữ liệu được truyền qua một lớp gộp tối đa. Lớp này giảm kích thước không gian của dữ liệu đầu vào (chiều rộng và chiều cao) nhưng giữ lại thông tin quan trọng nhất.

- Dropout: Đây là kỹ thuật ngăn chặn sự quá khớp của mô hình bằng cách ngẫu nhiên đặt một tỷ lệ phần tử đầu vào thành 0 trong mỗi cập nhật trong thời gian huấn luyện. Trong hình, có hai lớp dropout với tỷ lệ lần lượt là 0.25 và 0.5 tức là 25% và 50% số nơron sẽ bị tắt ngẫu nhiên trong mỗi lần cập nhật.

- Flattened: Sau khi đi qua lớp dropout đầu tiên, dữ liệu được làm phẳng thành một mảng một chiều để có thể đưa vào lớp kết nối đầy đủ.

- FC: Đây là lớp kết nối đầy đủ, nơi mà mỗi nơron trong lớp này được kết nối với mọi nơron trong lớp tiếp theo. Lớp này có thể học các mô hình phức tạp dựa trên các đặc trưng đã được trích xuất bởi các lớp trước đó.

- Softmax: Đây là hàm kích hoạt cuối cùng, được sử dụng để chuyển đổi điểm số của mô hình thành xác suất, giúp phân loại hình ảnh đầu vào thành các lớp khác nhau.

- RMSprop optimizer: Đây là trình tối ưu hóa được sử dụng trong quá trình huấn luyện mô hình. RMSprop giúp điều chỉnh tốc độ học trong quá trình huấn luyện để hội tụ nhanh hơn.

Như vậy ta đã nói về lý do chọn CNN cùng với sự kết hợp của ELA và thấy được luồng thực thi của chương trình xây dựng mô hình CNN. CNN thực sự là một lựa chọn tuyệt vời trong các nghiên cứu liên quan đến xử lý hình ảnh.

2.3. Cài đặt và chạy chương trình.

2.3.1. Các công cụ và thư viện cần thiết.

Để thực hiện đề tài nhóm chúng em sẽ sử dụng các công cụ sau:

- Python: là một ngôn ngữ lập trình bậc cao, dễ học, và phổ biến với cú pháp rõ ràng, hỗ trợ nhiều lĩnh vực như phát triển web, phân tích dữ liệu, AI, và nhiều ứng dụng khác. Với thư viện phong phú như NumPy, Pandas, TensorFlow, Python trở thành lựa chọn hàng đầu cho khoa học dữ liệu và học máy. Python cũng có cộng đồng lớn mạnh và tài liệu phong phú, giúp người dùng dễ dàng tiếp cận và học hỏi.

- Jupyter Notebook: là một công cụ mã nguồn mở cho phép tạo và chia sẻ tài liệu chứa mã nguồn, văn bản, hình ảnh, và biểu đồ. Đặc biệt phù hợp cho khoa học dữ liệu, học máy, và phân tích dữ liệu với hỗ trợ trực quan hóa kết quả ngay trong giao diện. Jupyter hỗ trợ nhiều ngôn ngữ lập trình, trong đó Python là phổ biến nhất.

- Kaggle: là một nền tảng trực tuyến chuyên dành cho khoa học dữ liệu và học máy, cung cấp các dữ liệu mẫu, và môi trường mã hóa miễn phí. Người dùng có thể truy cập tài nguyên phong phú, học hỏi từ cộng đồng, và thực hành trực tiếp trên nền tảng. Kaggle cũng tích hợp sẵn Jupyter Notebook và thư viện Python, giúp thực hiện các dự án dễ dàng.

Tiếp theo ta cùng đi tìm hiểu về các thư viện cần thiết cho đề tài. Dưới đây là hình ảnh về một số thư viện cần sử dụng:

```
#import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
np.random.seed(2)
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import EarlyStopping
from PIL import Image, ImageChops, ImageEnhance
import os
import itertools
```

Hình 2.15. Hình một số thư viện cần thiết cho đề tài.

- **numpy**: Thư viện Numpy (Numerical Python) hỗ trợ các phép toán đại số tuyến tính, biến đổi Fourier và các khả năng xử lý mảng nhiều chiều khác.

- **matplotlib.pyplot**: Matplotlib là một thư viện vẽ đồ thị 2D trong Python. Pyplot là một module trong Matplotlib cung cấp một giao diện giống

MATLAB.

- **sklearn**: Thư viện Scikit-learn (hay sklearn) là một thư viện trong Python hỗ trợ học máy và phân tích dữ liệu. Nó cung cấp nhiều công cụ cho các tác vụ như phân loại, hồi quy, gom nhóm, tiền xử lý dữ liệu, chọn mô hình và đánh giá mô hình.

- **keras**: Keras là một thư viện học sâu trong Python được xây dựng trên TensorFlow, Theano hoặc CNTK. Nó cung cấp một giao diện trực quan và dễ sử dụng để xây dựng và huấn luyện các mô hình học sâu. Keras hỗ trợ nhiều loại mô hình, bao gồm mô hình tuần tự (Sequential), mô hình chức năng (Functional) và mô hình mô phỏng (Model subclassing).

- **PIL**: Thư viện Python Imaging Library (PIL) hỗ trợ mở, thao tác và lưu nhiều định dạng hình ảnh khác nhau. Nó cung cấp các chức năng để làm việc với hình ảnh như xoay, cắt, thay đổi kích thước, v.v.

- **os**: Thư viện này cung cấp các hàm tương tác với hệ điều hành, bao gồm đọc, ghi file và quản lý đường dẫn.

- **itertools**: Thư viện này tạo ra các công cụ hiệu quả để xử lý vòng lặp.

2.3.2. Thu thập dữ liệu.

Việc huấn luyện một mô hình học máy bất kì đều không thể thiếu quá trình thu thập các dữ liệu cần thiết và phù hợp. Trong đề tài này, nhóm chúng em phải thực hiện việc thu thập tập dữ liệu về các hình ảnh không bị chỉnh sửa và các hình ảnh đã qua chỉnh sửa bằng một số phần mềm. Trang web Kaggle cung cấp một số lượng lớn các bộ dữ liệu huấn luyện để lựa chọn. Nhóm em đã sử dụng bộ dữ liệu CASIA, có thể tìm thấy trên TensorFlow. Chúng em có hai bộ dữ liệu, một bao gồm các bức ảnh thật được lưu trữ trong thư mục Au và bộ khác chứa các hình ảnh đã được chỉnh sửa được lưu trữ trong thư mục Tp như hình 2.16. Đây là một bộ dữ liệu huấn luyện tuyệt vời vì nó chứa đa dạng các hình ảnh ở mỗi lớp. Trong các mô hình phân loại cũng như mô hình CNN này, nhóm em luôn đảm bảo rằng mỗi lớp được đại diện 13 lần trong bộ dữ liệu, nếu có thể. Nhóm em sử dụng tổng cộng 20% số

ảnh trong bộ dữ liệu huấn luyện cho bộ dữ liệu kiểm tra. Mỗi bức ảnh này có độ phân giải là 3232 pixel. Mỗi số nguyên cho biết một mã màu, và mỗi pixel có giá trị từ 0 đến 255. Do đó, chúng em chia mỗi giá trị pixel cho 255 để đưa các giá trị pixel vào phạm vi từ 0 đến 1.

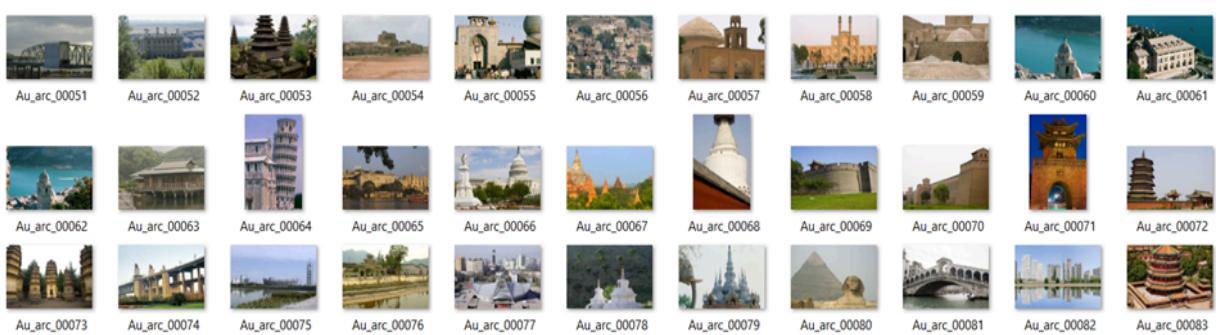
Name	Date modified	Type	Size
Au	2/19/2024 12:36 AM	File folder	
Tp	2/19/2024 1:22 AM	File folder	

Hình 2.16. Cấu trúc thư mục lưu trữ hình ảnh cho tập dữ liệu.

Với 7491 hình ảnh chưa qua chỉnh sửa hay ảnh thật sẽ bao gồm đa dạng các thể loại hình ảnh như động vật, con người, cây cối, kiến trúc,... giúp cho mô hình CNN có thể nhận biết được nhiều hơn.



Hình 2.17. Một số hình ảnh về động vật trong tập dữ liệu.



Hình 2.18. Một số hình ảnh về kiến trúc trong tập dữ liệu.



Hình 2.19. Một số hình ảnh về nghệ thuật trong tập dữ liệu.



Hình 2.20. Một số hình ảnh về con người trong tập dữ liệu.



Hình 2.21. Một số hình ảnh về thiên nhiên và cây cối trong tập dữ liệu.

Với 5123 hình ảnh đã qua chỉnh sửa hay ảnh giả mạo cũng bao gồm đa dạng các thể loại hình ảnh như bị cắt ghép, thay đổi màu ảnh,... ví dụ như 3 hình dưới đây



Hình 2.22. Hình ảnh giả mạo 1.



Hình 2.23. Hình ảnh giả mạo 2.



Hình 2.24. Hình ảnh giả mạo 3.

Các hình ảnh trong tập dữ liệu sau đó sẽ được xử lý bằng kỹ thuật ELA để cho ra ảnh ELA với cá điểm sáng tối:



Hình 2.25. Ảnh ELA.

Từ những hình ảnh ELA này, mô hình CNN dựa vào các đặc điểm sau để học hỏi và phân loại ảnh đã qua chỉnh sửa:

- Sự khác biệt về mức độ lỗi giữa các vùng ảnh:

- + ELA làm nổi bật các khu vực có độ nén khác nhau trong ảnh. Các vùng này thường xuất hiện khi ảnh bị chỉnh sửa (ví dụ: cắt ghép, làm mờ, chỉnh sáng).

- + CNN sẽ học cách phát hiện các mẫu lỗi từ những khác biệt này, chẳng hạn như các vùng có cường độ sáng hoặc màu sắc không đồng nhất.

- Cấu trúc không tự nhiên trong ảnh:

- + Những vùng chỉnh sửa thường có cấu trúc khác biệt so với các vùng tự nhiên trong ảnh, ví dụ: các cạnh bị mờ, màu sắc không liên tục, hoặc độ phân giải khác biệt.

- + CNN học các mẫu này bằng cách sử dụng các tầng tích chập (convolution layers) để nhận diện các đặc trưng hình học bất thường.

- Sự tương phản trong phân bố ánh sáng và màu sắc:

- + Ảnh ELA thường hiển thị các vùng có độ tương phản cao (do nén lại sau chỉnh sửa). Những vùng này trở thành tín hiệu mạnh mẽ để CNN nhận diện.

- + Các tầng pooling trong CNN giúp giảm nhiễu và tập trung vào những đặc trưng quan trọng, chẳng hạn như các điểm sáng và tối không đồng đều.

- Chi tiết bị mất trong vùng chỉnh sửa:

- + Chỉnh sửa ảnh thường làm mất các chi tiết nhỏ ở vùng chỉnh sửa, điều này sẽ thể hiện rõ trên ảnh ELA.

- + Mô hình CNN tận dụng các thông tin này để phát hiện các đặc trưng ảnh mà mắt người khó nhận biết.

Trên đây ta đã cùng tìm hiểu về bộ dữ liệu huấn luyện cho đề tài. Phần tiếp theo sẽ là triển khai cài đặt chương trình để huấn luyện mô hình CNN.

2.3.2. Xây dựng và huấn luyện mô hình.

2.3.2.1. Tiền xử lý dữ liệu.

Ta sẽ có một số định hàm sau:

```
def convert_to_ela_image(path, quality):
    temp_filename = 'temp_file_name.jpg'
    ela_filename = 'temp_ela.png'

    image = Image.open(path).convert('RGB')
    image.save(temp_filename, 'JPEG', quality = quality)
    temp_image = Image.open(temp_filename)

    ela_image = ImageChops.difference(image, temp_image)

    extrema = ela_image.getextrema()
    max_diff = max([ex[1] for ex in extrema])
    if max_diff == 0:
        max_diff = 1
    scale = 255.0 / max_diff

    ela_image = ImageEnhance.Brightness(ela_image).enhance(scale)

    return ela_image
```

Hình 2.26. Định nghĩa hàm chuyển đổi ảnh gốc thành ảnh ELA.

Giải thích: Hàm convert_to_ela_image nhận vào hai tham số là path (đường dẫn đến hình ảnh cần phân tích) và quality (chất lượng khi lưu hình ảnh JPEG). Trong hàm, trước hết ta sẽ đặt tên cho các tệp tạm thời sẽ được sử dụng trong quá trình phân tích vào các biến temp_filename và ela_filename. Tiếp theo, mở hình ảnh từ đường dẫn path, chuyển đổi nó thành định dạng RGB, sau đó lưu hình ảnh này dưới dạng JPEG với chất lượng quality đã chỉ định. Mở hình ảnh JPEG đã lưu này, ta tiến hành thực hiện các tính toán sự khác biệt giữa hình ảnh gốc và hình ảnh JPEG đã lưu bằng phương thức ImageChops.difference(). Kết quả là một hình ảnh mới cho thấy mức độ khác biệt giữa hai hình ảnh. Ta tiếp tục tính toán giá trị tối đa của sự khác biệt và sử dụng nó để tính toán tỷ lệ mà hình ảnh ELA sẽ được nhân lên, sao cho giá trị tối đa là 255. Cuối cùng, tăng độ sáng của hình ảnh ELA bằng cách nhân lên với tỷ lệ đã tính toán và sau đó trả về hình ảnh đó.

```
image_size = (128, 128)
def prepare_image(image_path):
    return np.array(convert_to_ela_image(image_path, 90)).resize(image_size).flatten() / 255.0
```

Hình 2.27. Định nghĩa hàm chuẩn bị hình ảnh.

Giải thích: Đầu tiên ta sẽ đặt kích thước mong muốn cho việc thay đổi kích thước hình ảnh thành một tuple 128x128 pixel. Sau đó, định nghĩa hàm `prepare_image` để xử lý hình ảnh tại `image_path`. Trong hàm này, có một lệnh `return` thực hiện một chuỗi các thao tác:

- Gọi hàm khác `convert_to_ela_image(image_path, 90)`, giả sử chuyển đổi hình ảnh gốc thành hình ảnh ELA (Error Level Analysis) với tham số chất lượng là 90.
- Thay đổi kích thước hình ảnh ELA này thành 128x128 pixel bằng cách sử dụng `.resize(image_size)`.
- Làm phẳng hình ảnh đã thay đổi kích thước thành một mảng một chiều bằng cách sử dụng `.flatten()`.
- Chia mỗi phần tử của mảng đã làm phẳng này cho 255.0 để chuẩn hóa giá trị pixel trong phạm vi từ 0 đến 1.

Bây giờ, ta sẽ định nghĩa hai mảng `X=[]` chứa các ảnh đã được chuyển đổi thành ảnh ELA và `Y=[]` dùng để lưu các nhãn có giá trị 0 ứng với ảnh giả mạo và 1 ứng với ảnh thật.

```
import random
path = '/kaggle/input/real-and-fake-images/Au/'
for dirname, _, filenames in os.walk(path):
    for filename in filenames:
        if filename.endswith('jpg') or filename.endswith('png'):
            full_path = os.path.join(dirname, filename)
            X.append(prepare_image(full_path))
            Y.append(1)
            if len(Y) % 500 == 0:
                print(f'Processing {len(Y)} images')

random.shuffle(X)
X = X[:2100]
Y = Y[:2100]
print(len(X), len(Y))
```

Đoạn mã trên thực hiện lấy tất cả dữ liệu ảnh thật từ thư mục Au và xử lý từng ảnh bằng hàm `prepare_image()` rồi được đẩy vào X, đồng thời đẩy 1 vào Y. Khi thực hiện xong vòng lặp X sẽ được trộn bằng phương thức `random.shuffle()`. Sau đó ta sẽ chỉ lấy ngẫu nhiên khoảng 2100 phần tử trong X và 2100 tương ứng trong Y và gán lại cho X, Y. Việc lấy số lượng ngẫu

nhiên các phần tử trong X và tương ứng trong Y có thể được tùy chỉnh.

```
path = '/kaggle/input/real-and-fake-images/Tp/'
for dirname, _, filenames in os.walk(path):
    for filename in filenames:
        if filename.endswith('jpg') or filename.endswith('png'):
            full_path = os.path.join(dirname, filename)
            X.append(prepare_image(full_path))
            Y.append(0)
        if len(Y) % 500 == 0:
            print(f'Processing {len(Y)} images')

print(len(X), len(Y))
```

Tương tự đoạn mã trên thực hiện lấy tất cả dữ liệu ảnh giả mạo từ thư mục Tp và xử lý từng ảnh bằng hàm prepare_image() rồi được đẩy thêm vào X, đồng thời đẩy thêm 0 vào Y.

Tiếp tục, chuyển đổi danh sách X chứa các hình ảnh đã được chuẩn bị thành một mảng numpy X=np.array(X). Chuyển đổi danh sách Y chứa nhãn của các hình ảnh thành một ma trận nhị phân Y=to_categorical(Y,2). Hàm to_categorical() của Keras được sử dụng để chuyển đổi một mảng lớp (integers) thành một ma trận lớp nhị phân. Tham số thứ hai, ở đây là 2, chỉ ra số lượng các lớp khác nhau.

```
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2, random_state=5)
X = X.reshape(-1,1,1,1)
print(len(X_train), len(Y_train))
print(len(X_val), len(Y_val))|
```

Đoạn mã trên thực hiện việc chia tập dữ liệu thành tập huấn luyện và tập kiểm định, thay đổi hình dạng của tập đặc trưng. Cụ thể, Sử dụng hàm train_test_split từ scikit-learn để chia tập dữ liệu X và Y thành tập huấn luyện và tập kiểm định. 20% dữ liệu được phân bổ cho tập kiểm định (test_size=0.2) và quá trình này có thể tái tạo được do đặt random_state. Tiếp theo, thay đổi hình dạng của tập đặc trưng X sử dụng phương thức reshape của numpy. Cuối cùng, in ra độ dài của cả tập huấn luyện và tập kiểm định cho đặc trưng (X) và nhãn (Y) để xác nhận kích thước của chúng.

2.3.2.2. Huấn luyện mô hình CNN.

Như vậy ta đã thực hiện xong các bước cần thiết để tiền xử lý dữ liệu. Phần này sẽ đi vào cài đặt CNN và train model.

```
def build_model():
    model = Sequential()
    model.add(Conv2D(filters = 32, kernel_size = (5, 5),
                    padding = 'valid', activation = 'relu', input_shape = (128, 128, 3)))
    model.add(Conv2D(filters = 32, kernel_size = (5, 5),
                    padding = 'valid', activation = 'relu', input_shape = (128, 128, 3)))
    model.add(MaxPool2D(pool_size = (2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(256, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation = 'softmax'))
    return model
```

Hình 2.28. Định nghĩa hàm để build model CNN.

Đoạn mã trên định nghĩa hàm build_model để tạo mô hình CNN. Đầu tiên, khởi tạo một mô hình tuần tự (Sequential) mới. Mô hình tuần tự là một loại mô hình trong Keras cho phép chúng ta thêm các lớp một cách tuần tự. Tiếp theo, thêm hai lớp tích chập 2D vào mô hình. Mỗi lớp sử dụng 32 bộ lọc kích thước 5x5, hàm kích hoạt ReLU, và không sử dụng padding (padding='valid'). Đầu vào của mỗi lớp là hình ảnh có kích thước 128x128 với 3 kênh màu. Thêm một lớp gộp tối đa 2D vào mô hình với kích thước pool là 2x2. Lớp này giảm kích thước không gian của đầu vào bằng cách lấy giá trị lớn nhất trong mỗi cửa sổ kích thước 2x2. Thêm một lớp Dropout vào mô hình với tỷ lệ dropout là 0.25. Lớp này ngẫu nhiên đặt một phần của đầu vào thành 0 trong quá trình huấn luyện, giúp ngăn chặn hiện tượng quá khớp. Thêm một lớp làm phẳng vào mô hình. Lớp này chuyển đổi đầu vào thành một mảng một chiều, chuẩn bị cho việc kết nối đầy đủ. Ta tiếp tục thêm một lớp kết nối đầy đủ vào mô hình với 256 nơron và hàm kích hoạt ReLU. Thêm một lớp Dropout khác vào mô hình với tỷ lệ dropout là 0.5. Thêm một lớp kết nối đầy đủ khác vào mô hình với 2 nơron và hàm kích hoạt softmax. Hàm softmax được sử dụng trong lớp đầu ra của mô hình phân loại để chuyển đổi điểm số của mỗi lớp thành xác suất.

Sau đó ta gọi build_model() để tạo mô hình CNN và lưu vào biến model. Gọi phương thức model.summary() để xem tóm tắt của mô hình.

```

model = build_model()
model.summary()

Model: "sequential"
=====
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 124, 124, 32)	2432
conv2d_1 (Conv2D)	(None, 120, 120, 32)	25632
max_pooling2d (MaxPooling2D)	(None, 60, 60, 32)	0
dropout (Dropout)	(None, 60, 60, 32)	0
flatten (Flatten)	(None, 115200)	0
dense (Dense)	(None, 256)	29491456
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 2)	514

```

=====
```

Total params: 29,520,034
Trainable params: 29,520,034
Non-trainable params: 0

Hình 2.29. Tạo model CNN và xem bản tóm tắt về model.

Ta tiếp tục đặt biến epochs = 30 và batch_size = 32.

```

init_lr = 1e-4
optimizer = Adam(lr = init_lr, decay = init_lr/epochs)
model.compile(optimizer = optimizer, loss = 'binary_crossentropy', metrics = [ 'accuracy' ])
early_stopping = EarlyStopping(monitor = 'val_acc',
                               min_delta = 0,
                               patience = 2,
                               verbose = 0,
                               mode = 'auto')

```

Đoạn mã trên thực hiện Khởi tạo tốc độ học là 0.0001. Sau đó, thiết lập trình tối ưu hóa Adam với tốc độ học và suy giảm đã chỉ định. Tiếp theo, biến dịch mô hình với trình tối ưu hóa Adam, hàm mất mát là binary_crossentropy và chỉ số đánh giá là độ chính xác. Thực hiện dừng sớm để dừng việc huấn luyện khi độ chính xác của tập kiểm định ('val_acc') không cải thiện ít nhất 'min_delta' trong 'patience' số lượng epochs.

Cuối cùng ta thực hiện huấn luyện mô hình model với phương thức fit() với tập dữ liệu huấn luyện X_train và Y_train, kích thước batch_size, số lần lặp epochs được chỉ định, có tập dữ liệu xác thực validation_data và có hàm early_stopping() để thực hiện dừng mô hình sớm. Ta có thể thu được các kết quả trong quá trình huấn luyện model ở mỗi epochs như hình sau.

```

hist = model.fit(X_train,
                  Y_train,
                  batch_size = batch_size,
                  epochs = epochs,
                  validation_data = (X_val, Y_val),
                  callbacks = [early_stopping])

Epoch 1/30
105/105 [=====] - 185s 2s/step - loss: 0.4658 - accuracy: 0.7838 - val_loss: 0.3392 - val_accuracy: 0.8
715
Epoch 2/30
105/105 [=====] - 187s 2s/step - loss: 0.2984 - accuracy: 0.8976 - val_loss: 0.2790 - val_accuracy: 0.9
052
Epoch 3/30
105/105 [=====] - 185s 2s/step - loss: 0.2709 - accuracy: 0.9003 - val_loss: 0.2447 - val_accuracy: 0.9
100
Epoch 4/30
105/105 [=====] - 188s 2s/step - loss: 0.2041 - accuracy: 0.9279 - val_loss: 0.2093 - val_accuracy: 0.9
172
Epoch 5/30
105/105 [=====] - 186s 2s/step - loss: 0.1789 - accuracy: 0.9328 - val_loss: 0.1867 - val_accuracy: 0.9
268
Epoch 6/30
105/105 [=====] - 186s 2s/step - loss: 0.1539 - accuracy: 0.9367 - val_loss: 0.1852 - val_accuracy: 0.9
328
Epoch 7/30
105/105 [=====] - 191s 2s/step - loss: 0.1356 - accuracy: 0.9517 - val_loss: 0.1786 - val_accuracy: 0.9
256

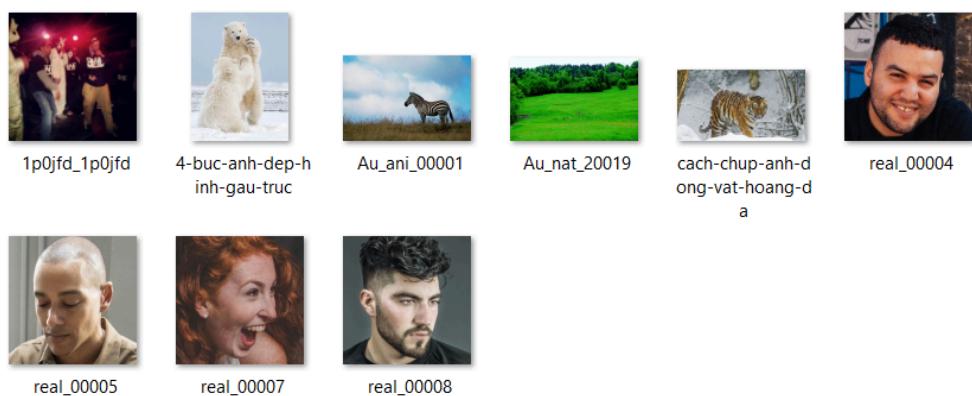
```

Hình 2.30. Thực hiện huấn luyện model.

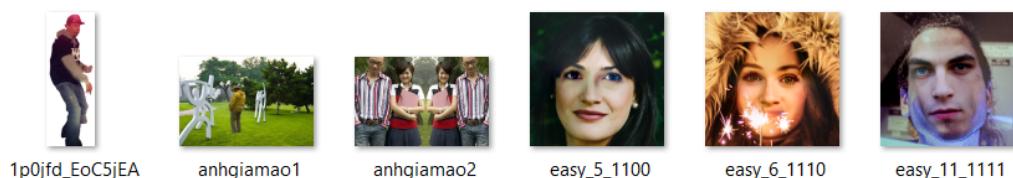
Mô hình sau đó sẽ được lưu lại dưới dạng file .h5 bằng câu lệnh sau với tên gọi là “model_casia_run1”: model.save('model_casia_run1.h5')

2.3.3. Chạy chương trình dự đoán.

Phần này sẽ thực hiện dùng mô hình CNN “model_casia_run1” đã huấn luyện để dự đoán trên hình ảnh mới. Dưới đây là một số hình ảnh sẽ được mô hình tiến hành dự đoán:

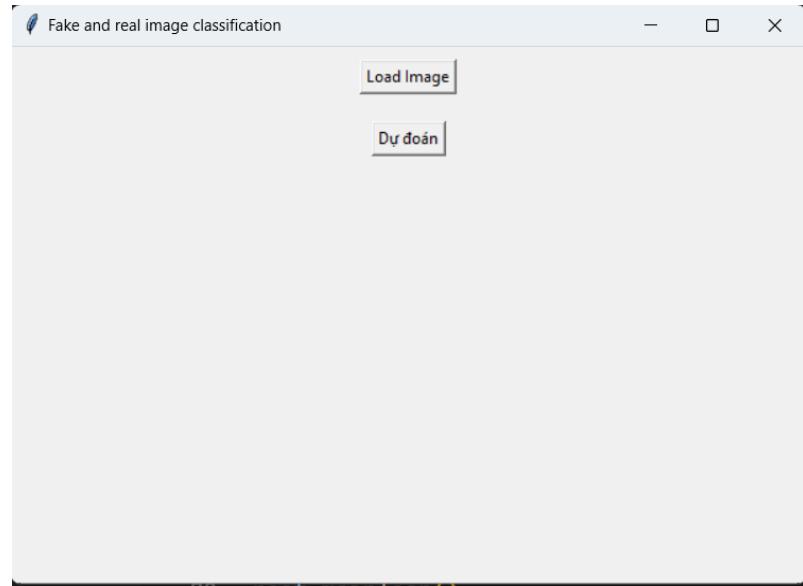


Hình 2.31. Tập các hình ảnh chưa qua chỉnh sửa.



Hình 2.32. Tập các hình ảnh đã qua chỉnh sửa.

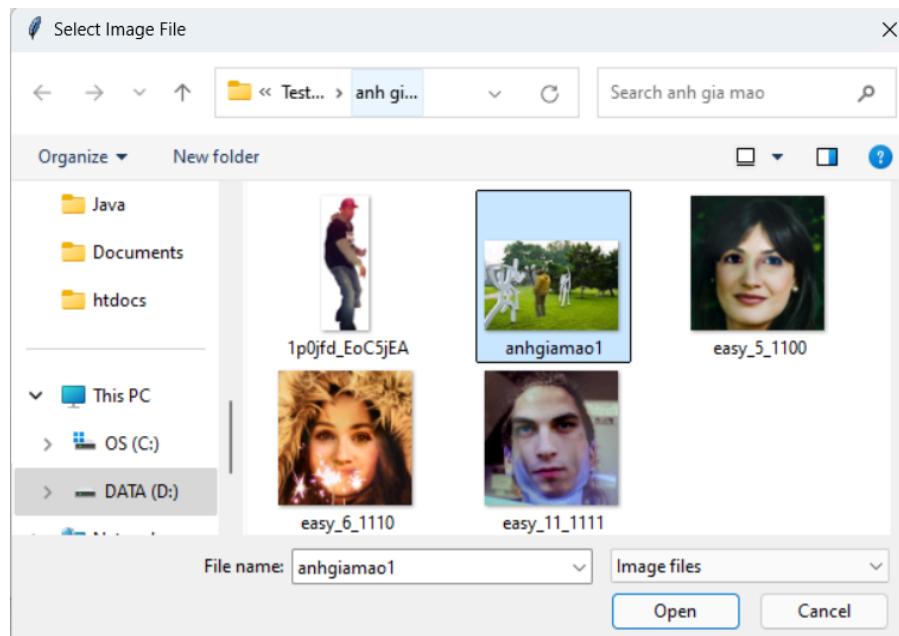
Nhóm sẽ dùng thư viện tkinter để tạo ra một phần mềm desktop đơn giản có tích hợp mô hình CNN “model_casia_run1” với chức năng là phát hiện ảnh đã qua chỉnh sửa.



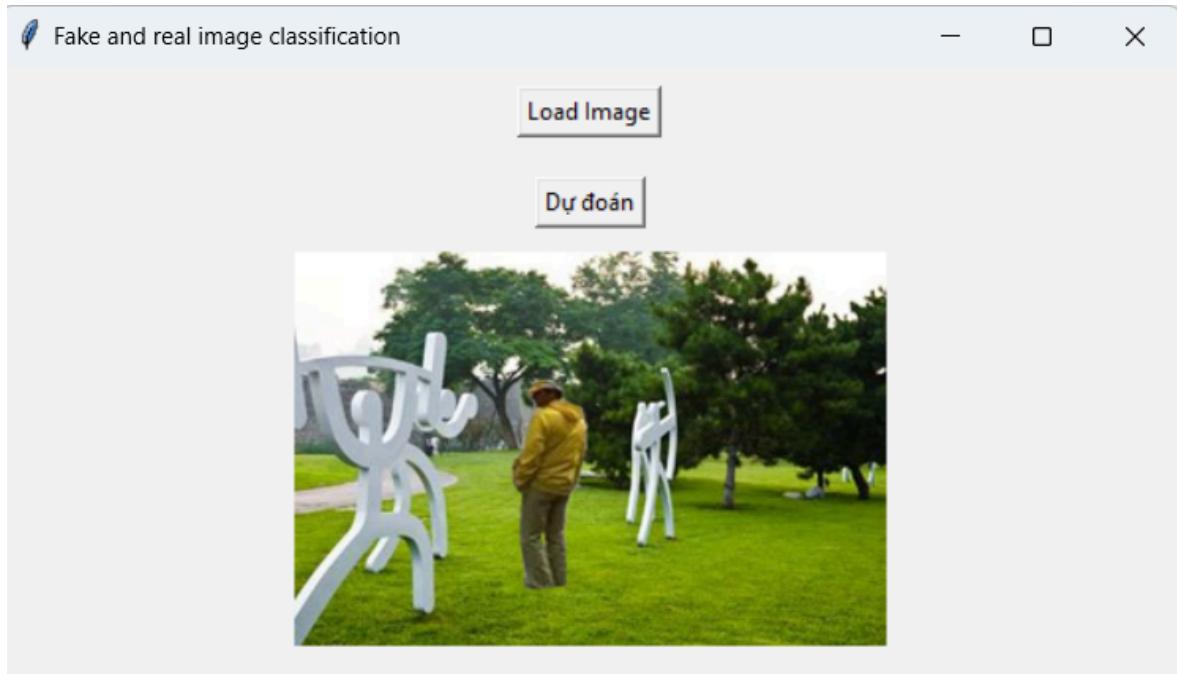
Hình 2.33. Giao diện phần mềm chạy mô hình CNN.

Các bước làm việc với phần mềm:

- Bước 1: Kích nút “Load Image” => Cửa sổ “Select Image File” hiện lên => Chọn ảnh cần mô hình dự đoán => Kích nút “Open” để ảnh được chọn hiển thị lên phần mềm.



Hình 2.34. Cửa sổ “Select Image File”.



Hình 2.35. Ảnh sau khi được chọn.

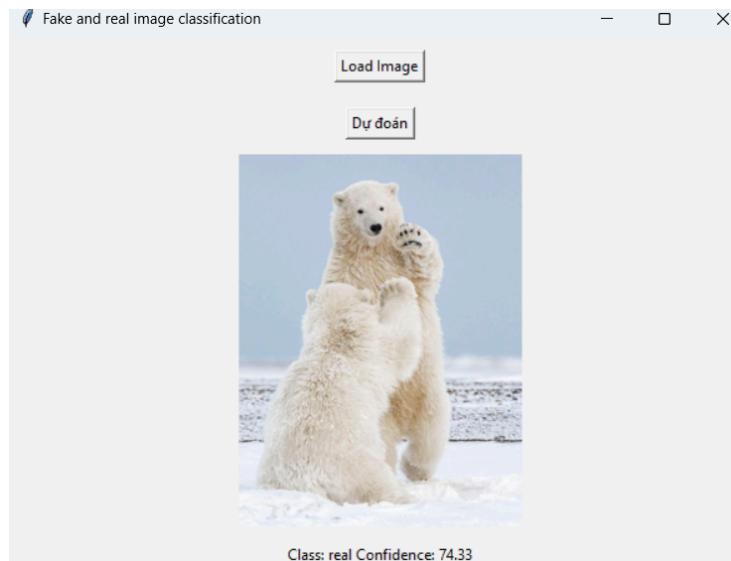
- Bước 2: Kích nút “Dự đoán”, lúc này mô hình “model_casia_run1” sẽ được kích hoạt để tiến hành dự đoán.



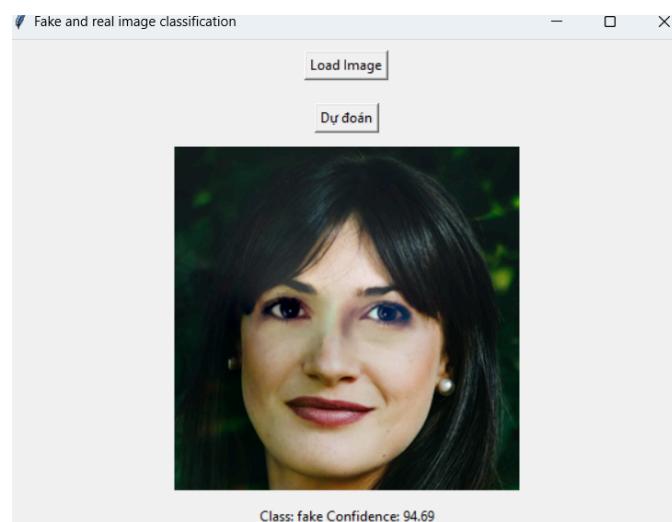
Hình 2.36. Kết quả dự đoán của mô hình “model_casia_run1”.

Hình 2.36 cho ta thấy ảnh được chọn là ảnh đã qua chỉnh sửa(Class: fake) và mức độ tự tin của mô hình(Confidence) là 94,41%.

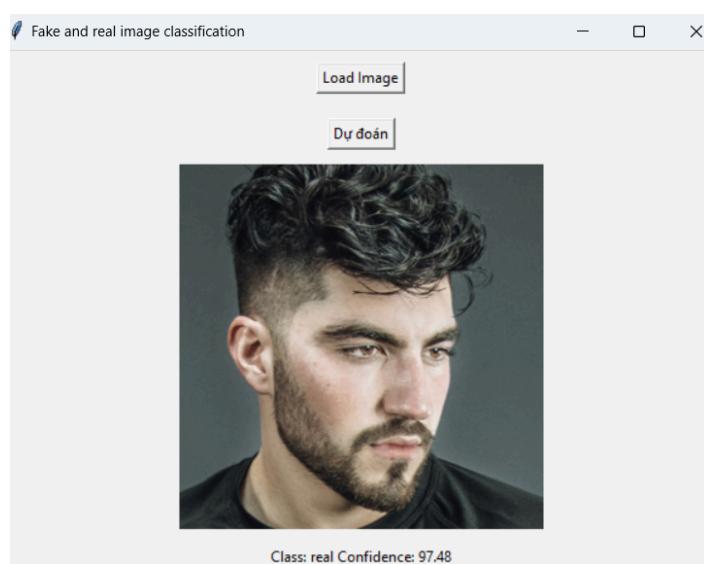
Một số kết quả dự đoán khác khi chạy mô hình “model_casia_run1”:



Hình 2.37. Kết quả dự đoán của mô hình “model_casia_run1”(2).



Hình 2.38. Kết quả dự đoán của mô hình “model_casia_run1”(3).

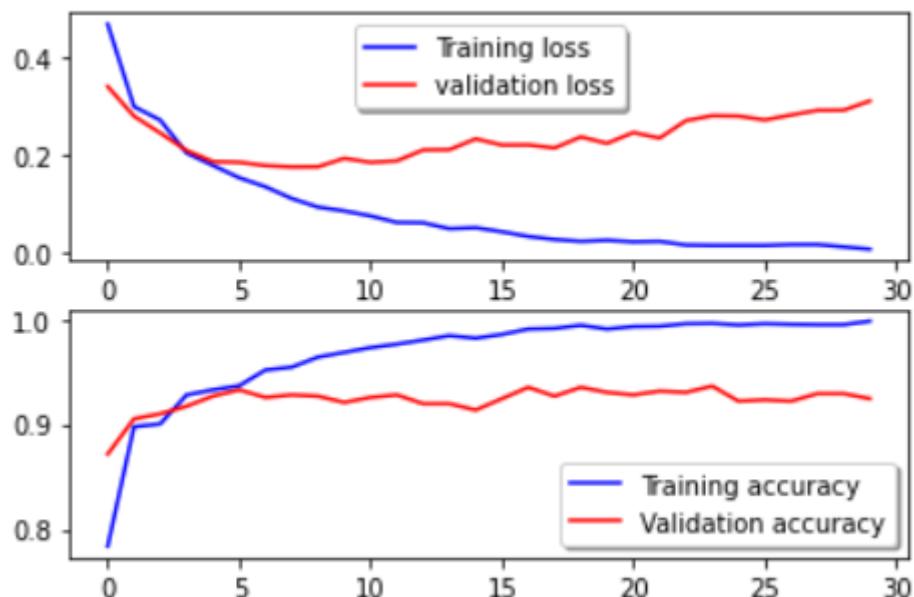


Hình 2.39. Kết quả dự đoán của mô hình “model_casia_run1”(4).

2.4. Phân tích kết quả chạy chương trình.

2.4.1. Độ chính xác của mô hình.

Để đánh giá độ chính xác của mô hình CNN “model_casia_run1”, nhóm sẽ sử dụng độ mất mát(loss) và độ chính xác(accuracy) để thực hiện đo. Dưới đây là kết quả sau khi tính toán độ mất mát(loss) và độ chính xác(accuracy) của mô hình:



Hình 2.40. Biểu đồ thể hiện mức độ lỗi và độ chính xác của mô hình.

Đối với biểu đồ "Training loss" và "Validation loss" (Trục trên):

- Đường màu xanh (Training loss): Độ mất mát trên tập huấn luyện giảm dần và hội tụ ổn định về gần 0. Đây là dấu hiệu tốt, cho thấy mô hình học dần các đặc trưng từ dữ liệu huấn luyện.
- Đường màu đỏ (Validation loss): Độ mất mát trên tập xác thực giảm trong giai đoạn đầu, nhưng sau đó dao động nhẹ và không giảm nhiều như tập huấn luyện. Đường này ổn định nhưng có vẻ không hội tụ về 0.

Đánh giá chi tiết:

- Hiện tượng overfitting: Ta thấy khoảng cách giữa "Training loss" và "Validation loss" ngày càng lớn sau một số epoch (validation loss cao hơn training loss). Điều này cho thấy mô hình có thể đã bị quá khớp (overfitting),

tức là mô hình học quá mức các chi tiết trên tập huấn luyện và không tổng quát hóa tốt trên tập xác thực.

- Tối ưu hóa: Hiệu năng trên tập xác thực không cải thiện đáng kể trong các epoch cuối cùng. Điều này gợi ý rằng mô hình có thể cần:

+ Thêm regularization: Áp dụng dropout, L1/L2 regularization để giảm overfitting.

+ Tăng dữ liệu huấn luyện: Sử dụng thêm dữ liệu hoặc các kỹ thuật augmentation để làm giàu dữ liệu.

+ Early stopping: Dừng huấn luyện sớm khi validation loss không giảm thêm.

Đối với biểu đồ "Training accuracy" và "Validation accuracy" (Trục dưới):

- Đường màu xanh (Training accuracy): Độ chính xác trên tập huấn luyện tăng nhanh và đạt giá trị cao (gần như tiệm cận 1.0). Điều này chứng minh mô hình đã học tốt trên tập huấn luyện.

- Đường màu đỏ (Validation accuracy): Độ chính xác trên tập xác thực tăng ban đầu nhưng dần ổn định và dao động quanh một giá trị cố định. Giá trị này thấp hơn so với độ chính xác trên tập huấn luyện, cho thấy khả năng tổng quát hóa của mô hình không hoàn hảo.

Đánh giá chi tiết:

- Hiện tượng overfitting: Độ chính xác trên tập huấn luyện và tập xác thực có sự chênh lệch rõ ràng ở các epoch cuối (training accuracy cao hơn validation accuracy). Đây là dấu hiệu của overfitting.

- Tiềm năng cải thiện: Nếu độ chính xác trên tập xác thực không cải thiện thêm sau một số epoch, có thể xem xét các điều chỉnh như:

+ Sử dụng cross-validation để kiểm tra khả năng tổng quát hóa.

+ Kiểm tra và cân nhắc cân bằng dữ liệu nếu có sự mất cân bằng giữa các lớp.

- + Thực hiện thử nghiệm các kiến trúc mô hình khác hoặc điều chỉnh hyperparameters như learning rate, batch size.

Tóm tắt tổng quát

- Tích cực:

- + Mô hình học tốt trên tập huấn luyện với độ măt măt giảm và độ chính xác tăng rõ rệt.

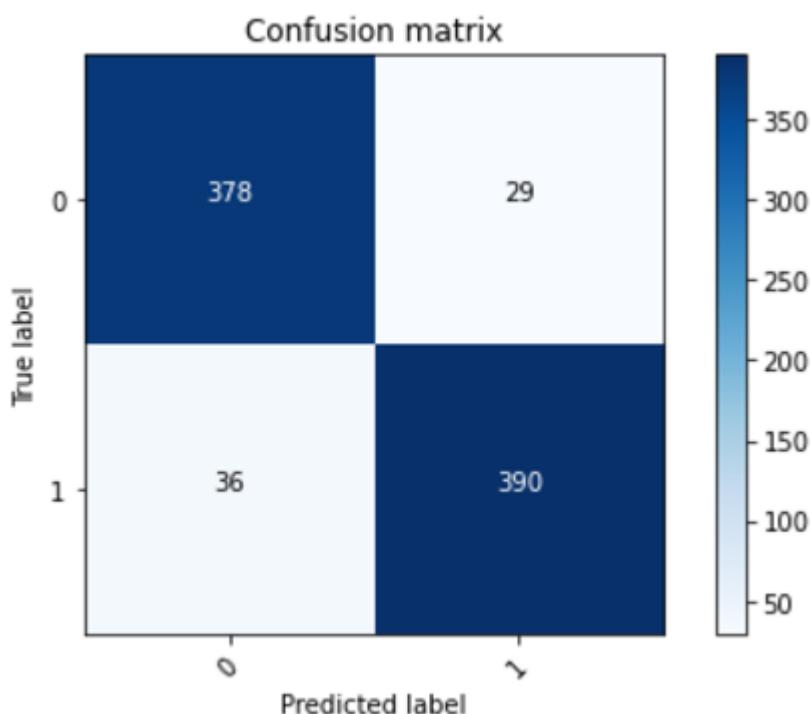
+ Đường "Validation loss" và "Validation accuracy" cho thấy mô hình hoạt động ổn định trên tập xác thực, không có dấu hiệu học quá ít (underfitting).

- Hạn chế:

+ Khoảng cách giữa kết quả trên tập huấn luyện và tập xác thực chỉ ra hiện tượng overfitting.

+ Mặc dù validation accuracy ổn định, giá trị này có thể chưa đạt yêu cầu nếu yêu cầu độ chính xác cao hơn.

Ngoài ra, nhóm thực hiện vẽ **ma trận nhầm lẫn (Confusion Matrix)**, một công cụ quan trọng để đánh giá hiệu suất của các mô hình học máy như CNN. Dưới đây là biểu đồ biểu diễn cho mô hình:



Hình 2.41. Biểu đồ ma trận nhầm lẫn cho mô hình “model_casia_run1”.

Dựa trên ma trận nhầm lẫn của mô hình CNN "model_casia_run1", chúng ta có thể đánh giá hiệu suất của mô hình.

Thông số chính:

- Số lượng Dương tính Thực (TP): 390.
- Số lượng Âm tính Thực (TN): 378.
- Số lượng Dương tính Giả (FP): 29.
- Số lượng Âm tính Giả (FN): 36.

Các chỉ số hiệu suất:

- Độ chính xác (Accuracy) = $\frac{TP+TN}{TP+TN+FP+FN} = \frac{390+378}{390+378+29+36} \approx 0.93$
- Độ chính xác Dương tính (Precision) = $\frac{TP}{TP+FN} = \frac{390}{390+36} \approx 0.93$
- Tỷ lệ Nhận diện (Recall) = $\frac{TP}{TP+FN} = \frac{390}{390+36} \approx 0.92$
- Điểm F1 (F1 Score) = $2 \times \frac{Precision \times Recall}{Precision + Recall} \approx 2 \times \frac{0.93 \times 0.92}{0.93 + 0.92} \approx 0.93$

Nhận xét:

- Mô hình có độ chính xác cao khoảng 93%, cho thấy nó hoạt động tốt tổng thể.
 - Độ chính xác Dương tính và Tỷ lệ Nhận diện đều cao, cho thấy mô hình hiệu quả trong việc xác định các trường hợp Dương tính thực trong khi giảm thiểu các trường hợp Dương tính giả và Âm tính giả.
 - Điểm F1 cũng cao, xác nhận tính ổn định của mô hình trong việc phát hiện hình ảnh giả mạo.

2.4.2. Hiệu quả của phương pháp ELA.

Phương pháp ELA (Error Level Analysis) là một kỹ thuật hữu ích trong việc phát hiện các hình ảnh giả mạo bằng cách phân tích các mức độ lỗi trong hình ảnh. Việc sử dụng phương pháp ELA để huấn luyện mô hình CNN "model_casia_run1" đã cho thấy hiệu quả rõ rệt qua các chỉ số hiệu suất cao như độ chính xác, độ chính xác Dương tính, tỷ lệ Nhận diện và điểm F1. Dưới đây là một số nhận xét về hiệu quả của phương pháp ELA:

- Phát hiện chính xác: Mô hình đạt được độ chính xác khoảng 93%, cho thấy khả năng phát hiện hình ảnh giả mạo rất tốt. Điều này cho thấy rằng các đặc trưng mà ELA cung cấp đã giúp mô hình học được các mẫu và đặc điểm quan trọng trong hình ảnh giả mạo.

- Giảm thiểu sai sót: Với số lượng Dương tính Giả (FP) chỉ là 29 và Âm tính Giả (FN) là 36, mô hình đã chứng minh khả năng phân loại chính xác giữa hình ảnh thật và giả. Điều này cho thấy rằng ELA đã giúp mô hình giảm thiểu các sai sót trong việc phân loại.

- Tính ổn định: Điểm F1 cao (khoảng 0.93) cho thấy sự cân bằng giữa độ chính xác và tỷ lệ nhận diện, điều này rất quan trọng trong các ứng dụng thực tế, nơi mà cả hai chỉ số này đều cần thiết để đảm bảo hiệu quả.

Tóm lại, phương pháp ELA đã chứng minh là một công cụ hiệu quả trong việc huấn luyện mô hình CNN "model_casia_run1". Việc sử dụng ELA không chỉ giúp cải thiện độ chính xác mà còn tăng cường khả năng phát hiện hình ảnh giả mạo, từ đó nâng cao hiệu suất tổng thể của mô hình.

2.4.3. Ứng dụng thực tế.

Để đánh giá khả năng áp dụng thực tế của mô hình CNN “model_casia_run1” trong việc phát hiện hình ảnh đã qua chỉnh sửa, bao gồm các loại chỉnh sửa phức tạp hơn, chúng ta cần xem xét các khía cạnh sau:

2.4.3.1. Độ chính xác trên tập dữ liệu phức tạp.

Mô hình hiện tại:

- Nếu mô hình được huấn luyện trên bộ dữ liệu CASIA, nó đã được tối ưu để phát hiện các dạng chỉnh sửa phổ biến như cắt ghép, làm mờ, thay đổi màu sắc.

- Tuy nhiên, các tình huống thực tế thường phức tạp hơn, bao gồm:
 - + **Chỉnh sửa sâu:** Sử dụng AI để tái tạo hoặc sửa đổi hình ảnh.
 - + **Xóa đối tượng và điền lại nền (Content-Aware Fill).**
 - + **Chỉnh sửa nhỏ nhưng tinh vi:** Thay đổi ánh sáng, thêm bóng, hoặc chỉnh màu tự nhiên hơn.

Đánh giá:

- Mô hình có thể gặp khó khăn với các chỉnh sửa phức tạp nếu tập dữ liệu huấn luyện không chứa các ví dụ tương tự.
- Để cải thiện, cần huấn luyện thêm trên các bộ dữ liệu có các dạng chỉnh sửa hiện đại hơn, ví dụ: dữ liệu chỉnh sửa bằng AI hoặc Deepfake.

2.4.3.2. Khả năng mở rộng.

Ưu điểm:

- CNN có khả năng học tốt các đặc trưng từ dữ liệu hình ảnh, do đó, nếu được huấn luyện thêm, mô hình có thể mở rộng khả năng phát hiện các dạng chỉnh sửa khác.

- Có thể tích hợp với các công cụ bổ sung như:
 - + **ELA (Error Level Analysis):** Để phát hiện các lỗi nén do chỉnh sửa.
 - + **Tích hợp kỹ thuật khác:** Như SIFT, SURF để tìm các bất thường về kết cấu.

Thách thức:

- Khả năng xử lý hình ảnh độ phân giải cao trong thời gian thực là một vấn đề.
- Nếu áp dụng cho môi trường yêu cầu tốc độ cao (như kiểm duyệt hình ảnh trên mạng xã hội), cần tối ưu hóa hiệu năng.

2.4.3.3. Tương thích với thực tế.

Các loại dữ liệu thực tế:

- **Hình ảnh từ các nguồn khác nhau:** Ảnh từ máy ảnh chuyên nghiệp có ít lỗi nén hơn so với ảnh tải lên từ mạng xã hội.
- **Định dạng và chất lượng ảnh:** Ảnh bị nén hoặc giảm chất lượng (JPEG artifacts) có thể làm giảm hiệu suất của mô hình.

Yêu cầu nâng cấp mô hình:

- Sử dụng **Transfer Learning** trên các tập dữ liệu chỉnh sửa mới, hiện đại hơn.
- Bổ sung dữ liệu với đa dạng định dạng và chất lượng ảnh.

2.4.3.4. Ứng dụng trong các trường hợp chỉnh sửa phức tạp.

Khả năng hiện tại: Với các chỉnh sửa phổ thông (cắt ghép, làm mờ, điều chỉnh độ sáng), mô hình có khả năng phát hiện tốt nếu các đặc trưng chỉnh sửa đã xuất hiện trong tập huấn luyện.

Giới hạn:

- Chỉnh sửa bằng AI:

- + Deepfake và GAN có thể tạo ra hình ảnh với độ chính xác cao, giảm thiểu lỗi chỉnh sửa rõ ràng.

- + Với loại hình này, mô hình CNN thông thường cần kết hợp thêm kỹ thuật phân tích dựa trên video hoặc động lực học khuôn mặt.

- **Kết hợp nhiều kỹ thuật chỉnh sửa:** Các hình ảnh bị chỉnh sửa nhiều bước thường phức tạp hơn, và mô hình có thể bỏ sót.

2.4.3.5. Đề xuất cải thiện để áp dụng thực tế.

Tăng cường dữ liệu:

- Thu thập và bổ sung dữ liệu hình ảnh đã qua chỉnh sửa từ các công cụ hiện đại (Photoshop, AI-based editing, Deepfake).

- Bao gồm các chỉnh sửa tinh vi như thay đổi ánh sáng, loại bỏ đối tượng, hoặc chỉnh sửa mượt mà hơn.

Sử dụng các kỹ thuật hỗ trợ:

- **ELA (Error Level Analysis):** Phát hiện các thay đổi nén.

- **Đặc trưng nâng cao:** Kết hợp đặc trưng kết cấu (texture) và phân tích histogram.

Huấn luyện trên các bộ dữ liệu hiện đại: Các bộ dữ liệu như FaceForensics++, DeepFake Detection Challenge dataset giúp mô hình học thêm về các dạng chỉnh sửa bằng AI.

Đánh giá định kỳ trong môi trường thực tế:

- Chạy thử nghiệm mô hình trên các tập dữ liệu thực tế chưa từng xuất hiện trong huấn luyện.

- Phân tích các trường hợp sai để tiếp tục tối ưu.

CHƯƠNG 3

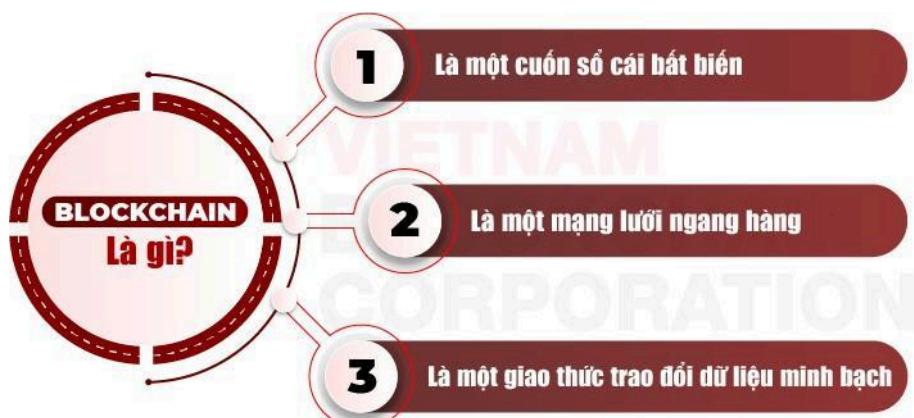
NGHIÊN CỨU PHÁT TRIỂN ÚNG DỤNG NFT VỚI BLOCKCHAIN

3.1. Tổng quan Blockchain.

3.1.1. Công nghệ Blockchain là gì?

3.1.1.1. Khái niệm công nghệ Blockchain?

Công nghệ Blockchain là một hệ thống cơ sở dữ liệu hiện đại, cho phép chia sẻ thông tin một cách minh bạch trong các mạng lưới kinh doanh. Dữ liệu trong chuỗi khối được lưu trữ dưới dạng các khối, liên kết chặt chẽ với nhau thành một chuỗi. Vì chuỗi dữ liệu được duy trì theo thứ tự thời gian và không thể xóa hay chỉnh sửa nếu không có sự đồng thuận từ toàn bộ mạng lưới, Blockchain tạo ra một sổ cái không thể thay đổi để quản lý các đơn đặt hàng, thanh toán, tài khoản, cùng các giao dịch khác. Công nghệ này tích hợp các cơ chế bảo mật để ngăn chặn truy cập trái phép và đảm bảo tính nhất quán trong cách các giao dịch được hiển thị chung. Tuy nhiên, tùy thuộc vào từng lĩnh vực cụ thể, Blockchain được đánh giá và áp dụng dưới nhiều góc nhìn khác nhau. Theo nhận định của nhiều chuyên gia, Blockchain có thể được hiểu theo ba khía cạnh chính:



Hình 3.1. Những quan điểm phổ biến về khái niệm “Công nghệ Blockchain”.

Tuy nhiên, dù định nghĩa theo cách nào thì Blockchain luôn có năm thành phần cơ bản:

- Sổ cái điện tử (Distributed Ledger): Sổ cái phân tán là cơ sở dữ liệu dùng chung trong mạng lưới chuỗi khối lưu trữ các giao dịch, chẳng hạn như

một tệp dùng chung mà mọi người trong nhóm có thể chỉnh sửa. Trong hầu hết các trình chỉnh sửa văn bản dùng chung, bất kỳ ai có quyền chỉnh sửa đều có thể xóa toàn bộ tệp.

- Mạng ngang hàng (Peer-to-peer network–P2P): Mạng P2P cho phép mỗi bên hoạt động như một máy khách và máy chủ. Điều này có nghĩa là sau khi mạng được hình thành, người tham gia đều sở hữu một bản sao của sổ cái. Từ đó có thể sử dụng để chia sẻ và lưu trữ tệp mà không cần sự trợ giúp của người trung gian.

- Mật mã học (Cryptography): Thành phần này nhằm để đảm bảo tính bí mật, toàn vẹn và xác thực của thông tin trong sổ cái điện tử hay các thông tin truyền đi giữa các nút. Nhờ xây dựng dựa trên nền tảng toán học (đặc biệt là lý thuyết xác suất) cùng với những kiến thức về lý thuyết trò chơi, mật mã học đã đưa ra được những phương thức mã hóa mà để phá vỡ nó là bất khả thi.

- Cơ chế đồng thuận (Consensus Mechanism): Cơ chế đồng thuận quy định các tập luật để các nút tham gia vào mạng ngang hàng có thể hoạt động một cách đồng bộ với nhau và thống nhất về các giao dịch nào là hợp pháp và nên được thêm vào Blockchain thông qua tương tác với hợp đồng thông minh – smart contract.

- Máy ảo (Virtual Machine): Máy ảo là một chương trình mô phỏng một hệ thống máy tính. Nó có một CPU, bộ nhớ và bộ lưu trữ ảo. Về cơ bản, máy ảo hoạt động giống như một máy tính vật lý, nó có thể dùng để lưu trữ dữ liệu, chạy các chương trình ứng dụng và tồn tại để cùng vận hành một mạng Blockchain với các máy ảo khác.

3.1.1.2. Ứng dụng công nghệ dựa trên Blockchain.

Internet of Things (IoT): Các thiết bị thông minh tạo nên Internet of Things (IoT). Bằng cách lưu trữ dữ liệu được thu thập từ các thiết bị này vào blockchain, người dùng có thể yên tâm về tính bảo mật của các thông tin, đặc biệt ngăn chặn xâm nhập của các dịch vụ giả mạo.

Smart Contracts: Smart Contracts (hợp đồng thông minh) được tạo ra như một thỏa thuận giữa hai hoặc nhiều bên mà không có sự tham gia của bất kỳ bên trung gian nào. Hợp đồng tồn tại trên một mạng lưới Blockchain phân tán và phi tập trung. Hợp đồng thông minh hiện là một phần ứng dụng blockchain quan trọng trong lĩnh vực chăm sóc sức khỏe, bất động sản, và thậm chí đối với các cơ quan chính phủ.

Bảo mật danh tính cá nhân: Web 2.0 tập trung hiện tại vẫn cho phép các bên thứ ba can thiệp vào hành vi sử dụng mạng của người dùng mạng, vì vậy, vẫn xảy ra những tình trạng bán thông tin cá nhân bị lộ. Cụ thể hơn, nó không thể triển khai danh tính tự chủ (Self-sovereign identity – SSI) cho người dùng. Tuy nhiên, blockchain có thể cung cấp cho người dùng quyền kiểm soát toàn diện thông tin về danh tính kỹ thuật số của họ. Vì vậy, bảo mật danh tính cá nhân là một trong những trường hợp sử dụng blockchain để mở đường cho SSI.

NFT (Non-fungible token): Các token không thể thay thế, hay NFT, đã trở nên phổ biến đáng kể trong thế giới tài chính. NFT có thể được sử dụng để tạo tài liệu làm bằng chứng xác thực cho các tài sản vật chất như tranh vẽ và đồ trang sức, nhạc kỹ thuật số và tác phẩm nghệ thuật, v.v. Hiện tại, cơn sốt NFT đã và đang bùng nổ trong giới nghệ sĩ, nhạc sĩ và những người có ảnh hưởng trên toàn cầu. Họ tận dụng công nghệ để nâng cao giá trị cho những tác phẩm nghệ thuật độc đáo của mình.

3.1.2. Tại sao blockchain lại quan trọng?

Các công nghệ cơ sở dữ liệu truyền thống đặt ra nhiều thách thức trong việc ghi lại các giao dịch tài chính. Chẳng hạn như hãy xét trường hợp bán một tài sản. Sau khi đã giao tiền, quyền sở hữu tài sản được chuyển cho người mua. Cả người mua và người bán đều có thể từng người ghi lại các giao dịch tiền tệ, nhưng không nguồn nào là đáng tin cậy. Người bán có thể dễ dàng khẳng định rằng họ chưa nhận được tiền ngay cả khi họ đã nhận được và người mua cũng có thể phản bác rằng họ đã chuyển tiền ngay cả khi họ chưa

thanh toán.

Để tránh các vấn đề pháp lý có thể xảy ra, cần phải có một bên thứ ba đáng tin cậy để giám sát và xác thực các giao dịch. Sự hiện diện của cơ quan trung tâm này không chỉ làm giao dịch phức tạp thêm mà còn tạo ra một lỗ hổng. Nếu cơ sở dữ liệu trung tâm bị xâm phạm, cả hai bên đều có thể chịu thiệt hại.

Chuỗi khối giảm thiểu những vấn đề như vậy bằng cách tạo ra một hệ thống chống làm giả, phi tập trung để ghi lại các giao dịch. Trong trường hợp giao dịch tài sản, người mua và người bán đều được chuỗi khối tạo cho một sổ cái riêng. Tất cả các giao dịch phải được cả hai bên chấp thuận và được cập nhật tự động vào sổ cái của cả hai trong thời gian thực. Các giao dịch trước đây có bất cứ sai sót nào cũng sẽ làm toàn bộ sổ cái sai lệch theo. Những đặc tính đó của công nghệ chuỗi khối đã dẫn đến việc công nghệ này được sử dụng trong nhiều lĩnh vực khác nhau, bao gồm cả việc tạo ra tiền kỹ thuật số như Bitcoin.

3.1.3. Công nghệ Blockchain có những đặc điểm gì?

Công nghệ chuỗi khối có các đặc điểm chính sau:

- Phi tập trung: Phi tập trung trong chuỗi khối là chỉ việc chuyển quyền kiểm soát và ra quyết định từ một thực thể tập trung (cá nhân, tổ chức hoặc nhóm) sang một mạng lưới phân tán. Các mạng lưới chuỗi khối phi tập trung sử dụng tính minh bạch để giảm nhu cầu phải có sự tin tưởng giữa những người tham gia. Các mạng lưới này cũng ngăn cản những người tham gia sử dụng quyền hạn hoặc quyền kiểm soát lẫn nhau theo những cách làm suy yếu chức năng của mạng lưới.

- Bất biến: Bất biến có nghĩa là một cái gì đó không thể thay đổi hay biến đổi được. Không người tham gia nào có thể làm giả giao dịch sau khi ai đó đã ghi lại giao dịch này vào sổ cái được chia sẻ. Nếu bản ghi giao dịch có lỗi, bạn phải thêm giao dịch mới để bù trừ cho lỗi và cả hai giao dịch đều được hiển thị trong mạng lưới.

- Đồng thuận: Một hệ thống chuỗi khối thiết lập các quy tắc về sự đồng thuận của người tham gia cho phép ghi lại các giao dịch. Bạn chỉ có thể ghi lại các giao dịch mới khi đa số người tham gia mạng lưới đồng thuận.

3.1.4. Chuỗi khối hoạt động như thế nào?

Trong khi các cơ chế chuỗi khối cơ bản rất phức tạp, chúng tôi sẽ trình bày tổng quan ngắn gọn trong các bước sau. Phần mềm chuỗi khối có thể tự động hóa hầu hết các bước sau:

Bước 1 - Ghi lại giao dịch: Một giao dịch chuỗi khối cho thấy sự lưu động của các tài sản vật lý hoặc kỹ thuật số từ bên này đến bên khác trong mạng lưới chuỗi khối. Giao dịch được ghi lại dưới dạng một khối dữ liệu và có thể bao gồm các thông tin chi tiết như sau:

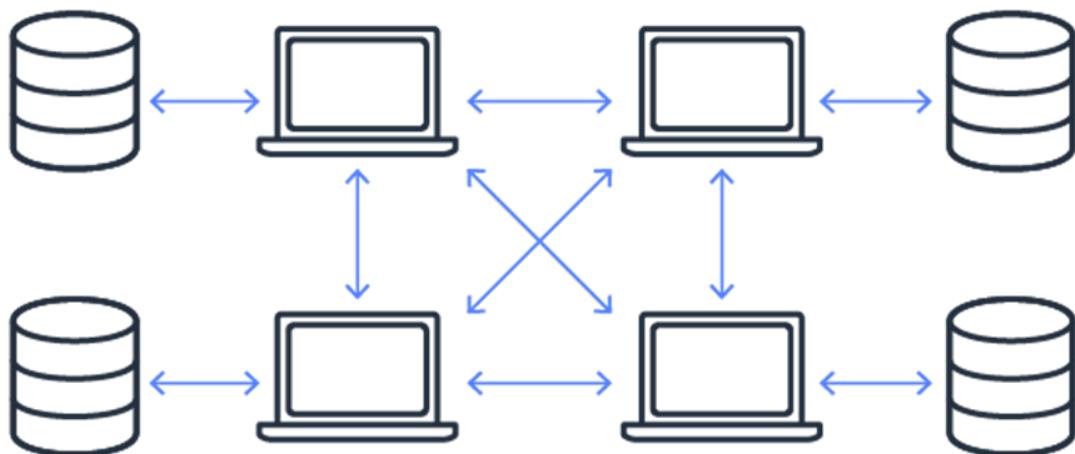
- Giao dịch gồm những ai tham gia?
- Điều gì đã xảy ra trong quá trình giao dịch?
- Giao dịch xảy ra khi nào?
- Giao dịch xảy ra ở đâu?
- Giao dịch xảy ra vì lý do gì?
- Phần tài sản được trao đổi là bao nhiêu?
- Có bao nhiêu điều kiện tiên quyết đã được đáp ứng trong quá trình giao dịch?

Bước 2 - Đạt được sự đồng thuận: Hầu hết những người tham gia trên mạng lưới chuỗi khối phân tán phải đồng ý rằng giao dịch được ghi lại là hợp lệ. Tùy thuộc vào loại mạng lưới, các quy tắc thỏa thuận có thể khác nhau nhưng thường được thiết lập khi bắt đầu mạng lưới.

Bước 3 - Liên kết các khối: Khi những người tham gia đã đạt được sự đồng thuận, các giao dịch trên chuỗi khôi sẽ được viết vào khối, tương đương với trang giấy trong một cuốn sổ cái. Cùng với các giao dịch, một hàm băm mật mã cũng được thêm vào khối mới. Hàm băm đóng vai trò như một chuỗi liên kết các khối với nhau. Nếu nội dung của khối bị cố ý hoặc vô ý sửa đổi, giá trị băm sẽ thay đổi, mang đến một cách thức để phát hiện dữ liệu bị làm

giả. Do đó, các khối và chuỗi được liên kết an toàn và bạn không thể chỉnh sửa chúng. Mỗi khối được thêm lại tăng cường cho quá trình xác minh khối trước đó và do đó tăng cường cho toàn bộ chuỗi khối. Điều này giống như xếp chồng các khối gỗ để tạo thành một tòa tháp. Bạn chỉ có thể xếp khối lên trên, và nếu bạn rút một khối ở giữa tháp thì cả tháp sẽ đổ sụp.

Bước 4 – Chia sẻ sổ cái: Hệ thống phân phối bản sao mới nhất của sổ cái trung tâm cho toàn bộ người tham gia.



Hình 3.2. Minh họa hoạt động chia sẻ sổ cái trong chuỗi khối.

3.1.5. Có những loại mạng lưới blockchain nào?

Có 4 loại mạng lưới phi tập trung hoặc phân tán chính trong chuỗi khối:

- Mạng lưới chuỗi khối công khai: Các chuỗi khối công khai không yêu cầu quyền và mọi người đều được phép tham gia. Tất cả các thành viên của chuỗi khối này đều có quyền đọc, chỉnh sửa và xác thực chuỗi khối như nhau. Mọi người chủ yếu sử dụng các chuỗi khối công khai để trao đổi và đào các loại tiền điện tử như Bitcoin, Ethereum và Litecoin.

- Mạng lưới chuỗi khối riêng tư: Một tổ chức duy nhất sẽ kiểm soát các chuỗi khối riêng tư, còn gọi là các chuỗi khối được quản lý. Cơ quan này xác định ai có thể là thành viên và họ có những quyền gì trong mạng lưới. Các chuỗi khối riêng tư chỉ phi tập trung một phần vì những chuỗi khối này có các hạn chế về quyền truy cập. Ripple, một mạng lưới trao đổi tiền kỹ thuật số dành cho các doanh nghiệp, là một ví dụ về chuỗi khối riêng tư.

- Mạng lưới chuỗi khối hỗn hợp: Các chuỗi khối hỗn hợp kết hợp các yếu tố từ cả mạng lưới riêng tư và mạng lưới công khai. Các công ty có thể thiết lập những hệ thống riêng tư, dựa trên quyền hạn bên cạnh một hệ thống công khai. Bằng cách này, họ kiểm soát quyền truy cập vào dữ liệu cụ thể được lưu trữ trong chuỗi khối trong khi vẫn công khai những dữ liệu còn lại. Họ sử dụng các hợp đồng thông minh để các thành viên công cộng có thể kiểm tra xem những giao dịch riêng tư đã được hoàn thành hay chưa. Ví dụ: các chuỗi khối hỗn hợp có thể cấp quyền truy cập công khai vào tiền kỹ thuật số trong khi giữ đồng tiền thuộc sở hữu của ngân hàng ở chế độ riêng tư.

- Các mạng lưới chuỗi khối liên hợp: Một nhóm các tổ chức quản lý các mạng lưới chuỗi khối liên hợp. Các tổ chức được chọn từ trước chia sẻ trách nhiệm duy trì chuỗi khối và quyết định về quyền truy cập dữ liệu. Các ngành trong đó nhiều tổ chức có cùng mục tiêu và hưởng lợi từ trách nhiệm chung thường thích dùng mạng lưới chuỗi khối liên hợp. Ví dụ: Global Shipping Business Network Consortium là một liên hợp chuỗi khối phi lợi nhuận nhằm mục đích số hóa ngành vận tải biển và tăng cường sự hợp tác giữa các đơn vị khai thác ngành hàng hải.

3.1.6. Các giao thức Blockchain là gì?

3.1.6.1. Khái niệm về các giao thức Blockchain.



Hình 3.3. Minh họa giao thức trong Blockchain.

Trong thế giới lập trình máy tính, các giao thức (protocol) là các quy tắc tiêu chuẩn hóa quy định hệ thống nên làm gì hoặc không nên làm gì. Do

đó, trong thế giới của blockchain, các giao thức blockchain là một tập hợp các mã hoặc yêu cầu chi phối cách thức hoạt động của một blockchain.

Một giao thức blockchain có thể đặt ra một loạt các quy tắc, chẳng hạn như giao diện của blockchain, sự tương tác của các máy tính tham gia, loại dữ liệu nên được chia sẻ, các ưu đãi cho các nhà phát triển tham gia vào mạng, v.v.

3.1.6.2. Các loại giao thức Blockchain chính.

Có hàng trăm giao thức blockchain đang tồn tại và mỗi giao thức cung cấp một chức năng khác nhau tùy thuộc vào công ty điều hành chúng.

Dưới đây là các loại giao thức blockchain chính trong thế giới tiền điện tử:

- Bitcoin: Giao thức giao dịch của Bitcoin là giao thức blockchain mang tính bước ngoặt đã cách mạng hóa thanh toán kỹ thuật số. Một số giao thức của nó đã đặt nền tảng cho các giao dịch tiền điện tử: giao dịch ngang hàng (peer-to-peer transactions), băm (hashes), chữ ký kỹ thuật số, v.v.

- Ethereum: Giao thức của Ethereum dựa trên các hợp đồng thông minh, nơi các giao dịch được tự động thực hiện khi đáp ứng các tiêu chí đã thiết lập trên mạng.

- Cardano: Tiền điện tử mới nổi Cardano chạy trên một giao thức được gọi là Ouroboros, được thiết kế để giảm tiêu thụ năng lượng trong các giao dịch tiền điện tử.

- Hyperledger: Hyperledger được nhắm mục tiêu vào các doanh nghiệp từ các lĩnh vực khác nhau và nhằm mục đích trao quyền cho các giao dịch kinh doanh và các dịch vụ tài chính khác.

Trong thị trường tiền điện tử, nhiều altcoin là nhánh của Bitcoin và Ethereum. Một fork tiền điện tử về cơ bản là một “bản sao” của giao thức của một loại tiền điện tử khác.

3.1.7. Công nghệ blockchain đã phát triển như thế nào?

Công nghệ chuỗi khối bắt nguồn từ cuối những năm 1970, khi một nhà

khoa học máy tính tên là Ralph Merkle được cấp bằng sáng chế cho cây Băm hay cây Merkle. Những cây này là một cấu trúc khoa học máy tính để lưu trữ dữ liệu bằng cách liên kết các khối có sử dụng mật mã. Vào cuối những năm 1990, Stuart Haber và W. Scott Stornetta đã sử dụng cây Merkle để triển khai một hệ thống trong đó dấu thời gian của tài liệu không thể bị làm giả. Đây là trường hợp chuỗi khối xuất hiện đầu tiên trong lịch sử.

Công nghệ này đã tiếp tục phát triển qua 3 thế hệ sau:

- Thế hệ đầu tiên – Bitcoin và các loại tiền ảo khác: Vào năm 2008, một cá nhân hoặc một nhóm cá nhân ẩn danh chỉ được biết đến với cái tên Satoshi Nakamoto đã dựng lên “bộ khung” cho công nghệ chuỗi khối ở hình thái hiện đại. Ý tưởng của Satoshi về chuỗi khối Bitcoin đã sử dụng khối thông tin 1 MB cho các giao dịch Bitcoin. Nhiều tính năng của hệ thống chuỗi khối Bitcoin vẫn đóng vai trò then chốt trong công nghệ chuỗi khối cho đến ngày nay.

- Thế hệ thứ hai – hợp đồng thông minh: Một vài năm sau khi những đồng tiền thế hệ đầu tiên xuất hiện, các nhà phát triển bắt đầu xem xét về các ứng dụng chuỗi khối ngoài tiền điện tử. Ví dụ: những người phát minh ra Ethereum đã quyết định sử dụng công nghệ chuỗi khối trong các giao dịch chuyển nhượng tài sản. Đóng góp đáng kể của họ là tính năng hợp đồng thông minh.

Thế hệ thứ ba – tương lai: Khi các công ty khám phá và triển khai các ứng dụng mới, công nghệ chuỗi khối vẫn tiếp tục cải tiến và phát triển. Các công ty đang giải quyết những hạn chế về quy mô cũng như điện toán và trong cuộc cách mạng chuỗi khối đang diễn ra này, tồn tại vô vàn cơ hội.

3.1.8. Lợi ích.

Công nghệ chuỗi khối mang lại nhiều lợi ích cho việc quản lý giao dịch tài sản. Dưới đây, chúng tôi liệt kê một vài lợi ích trong số đó:

- Bảo mật nâng cao: Hệ thống chuỗi khối cung cấp mức độ bảo mật và sự tin cậy cao mà các giao dịch kỹ thuật số hiện đại yêu cầu. Luôn tồn tại nỗi

sợ rằng ai đó sẽ thao túng phần mềm cơ sở để tạo ra tiền giả cho bản thân họ. Nhưng chuỗi khối sử dụng 3 nguyên tắc mật mã, phi tập trung và đồng thuận để tạo ra một hệ thống phần mềm cơ sở có độ bảo mật cao, gần như không thể bị làm giả. Không có một điểm lỗi làm chết cả hệ thống và một người dùng sẽ không thể thay đổi các bản ghi giao dịch.

- Cải thiện hiệu quả: Các giao dịch giữa doanh nghiệp với nhau có thể tồn tại rất nhiều thời gian và tạo ra tắc nghẽn trong hoạt động, đặc biệt là khi có sự tham gia của các cơ quan quản lý và quản lý bên thứ ba. Tính minh bạch và các hợp đồng thông minh trong chuỗi khối làm cho các giao dịch kinh doanh như vậy nhanh hơn và hiệu quả hơn.

- Kiểm tra nhanh hơn: Doanh nghiệp phải có khả năng tạo, trao đổi, lưu trữ và xây dựng lại các giao dịch điện tử một cách an toàn theo cách thức có thể kiểm tra được. Các bản ghi trong chuỗi khối là bất biến theo thứ tự thời gian, có nghĩa là tất cả các bản ghi luôn được sắp xếp theo thời gian. Tính minh bạch của dữ liệu này giúp cho việc xử lý kiểm tra nhanh hơn hẳn.

3.1.9. Sự khác biệt giữa Bitcoin và chuỗi khối là gì?

Blockchain:

- Định nghĩa: Blockchain là một cơ sở dữ liệu phân tán, không thể thay đổi, được bảo mật bằng mã hóa và được sử dụng để ghi lại các giao dịch. Mỗi khối trong blockchain chứa một tập hợp các giao dịch và được liên kết với khối trước đó bằng một hàm băm.

- Các thành phần chính:

- + Khối (block): Đơn vị cơ bản của blockchain, chứa thông tin về các giao dịch.

- + Chuỗi (chain): Liên kết các khối lại với nhau theo thứ tự thời gian.

- + Hàm băm: Một hàm toán học tạo ra một chuỗi ký tự duy nhất từ một khối dữ liệu đầu vào.

- + Mạng lưới ngang hàng (peer-to-peer): Các máy tính tham gia vào mạng lưới blockchain, xác thực và lưu trữ dữ liệu.

- Các ứng dụng tiềm năng:

+ Tiền điện tử: Bitcoin là ví dụ điển hình, nhưng còn nhiều loại tiền điện tử khác.

+ Hợp đồng thông minh: Các chương trình tự thực thi được viết trên blockchain.

+ Quản lý chuỗi cung ứng: Theo dõi hàng hóa từ lúc sản xuất đến khi đến tay người tiêu dùng.

+ Y tế: Lưu trữ hồ sơ bệnh án điện tử an toàn và bảo mật.

+ Bất động sản: Ghi nhận quyền sở hữu tài sản.

Bitcoin:

- Định nghĩa chi tiết: Bitcoin là một loại tiền điện tử phi tập trung, được tạo ra và quản lý bởi một mạng lưới các máy tính trên toàn thế giới.

- Cách thức hoạt động:

+ Khai thác (mining): Quá trình sử dụng máy tính để giải các bài toán phức tạp và xác nhận các giao dịch, nhận được bitcoin làm phần thưởng.

+ Ví điện tử: Một chương trình phần mềm hoặc thiết bị vật lý để lưu trữ và quản lý bitcoin.

+ Giao dịch: Việc chuyển bitcoin từ một ví điện tử sang ví điện tử khác.

- Ưu điểm

+ Phi tập trung: Không có ngân hàng trung ương kiểm soát.

+ Bảo mật cao: Dựa trên công nghệ mã hóa mạnh mẽ.

+ Minh bạch: Mọi giao dịch đều được ghi lại công khai trên blockchain.

- Nhược điểm:

+ Biến động giá: Giá trị của bitcoin có thể thay đổi mạnh trong thời gian ngắn.

+ Tốc độ giao dịch: Có thể chậm hơn so với các phương thức thanh toán truyền thống.

+ Tính phức tạp: Công nghệ blockchain và bitcoin khá phức tạp để hiểu và sử dụng.

Các yếu tố khác có thể bổ sung:

- An toàn và bảo mật: Thảo luận về các cuộc tấn công vào blockchain và các biện pháp bảo vệ.
- Vấn đề pháp lý: Quy định của các quốc gia về tiền điện tử và blockchain.
- Tác động môi trường: Tiêu thụ năng lượng của quá trình khai thác bitcoin.
- Các loại tiền điện tử khác: Ethereum, Ripple, v.v.
- Tương lai của blockchain: Các xu hướng phát triển và tiềm năng ứng dụng trong tương lai.

3.2. Nghiên cứu NFT.

3.2.1. NFT là gì?

NFT, viết tắt của Non-Fungible Token (mã thông báo không thể thay thế), là một dạng tài sản kỹ thuật số độc đáo được xác thực trên nền tảng blockchain. Khác với tiền điện tử như Bitcoin có thể thay thế lẫn nhau, mỗi NFT là một tác phẩm nghệ thuật kỹ thuật số duy nhất. Giá trị của NFT đến từ tính độc đáo, khan hiếm và quyền sở hữu độc quyền mà nó mang lại. Các NFT có thể đại diện cho nhiều loại tài sản khác nhau, từ tác phẩm nghệ thuật kỹ thuật số, âm nhạc, video, đến các vật phẩm trong game, bộ sưu tập, thậm chí cả bất động sản ảo. Ưu điểm nổi bật của NFT là tính minh bạch, bảo mật và tiềm năng đầu tư. Tuy nhiên, bên cạnh những lợi ích, NFT cũng đi kèm với một số thách thức như biến động giá mạnh, công nghệ còn mới và rủi ro lừa đảo. Với sự phát triển của công nghệ blockchain và sự quan tâm ngày càng tăng của cộng đồng, NFT đang trở thành một xu hướng đầu tư mới nổi và hứa hẹn sẽ cách mạng hóa nhiều ngành công nghiệp trong tương lai.

Để hiểu rõ hơn về NFT, chúng ta có thể lấy ví dụ về các tác phẩm nghệ thuật kỹ thuật số được mã hóa thành NFT. Khi bạn mua một NFT đại diện cho một bức tranh kỹ thuật số, bạn không chỉ sở hữu một bản sao của bức tranh đó mà còn sở hữu một chứng nhận số về quyền sở hữu độc quyền đối với tác

phẩm gốc. Điều này giống như việc sở hữu một bức tranh truyền thống, nhưng với những lợi ích bổ sung như tính minh bạch và khả năng xác minh nguồn gốc.

Tuy nhiên, cũng cần lưu ý rằng thị trường NFT còn khá mới và biến động. Giá trị của NFT có thể tăng hoặc giảm mạnh trong một thời gian ngắn. Ngoài ra, việc đánh giá giá trị của một NFT cũng khó khăn hơn so với các loại tài sản truyền thống. Do đó, trước khi quyết định đầu tư vào NFT, bạn nên nghiên cứu kỹ và cân nhắc các rủi ro tiềm ẩn.

3.2.2. Ứng dụng phổ biến của NFT là gì?

Nghệ thuật kỹ thuật số:

- Sở hữu tác phẩm độc đáo: NFT cho phép nghệ sĩ tạo ra các tác phẩm nghệ thuật kỹ thuật số độc nhất và bán chúng trực tiếp cho người mua mà không cần qua trung gian.

- Xác thực tác quyền: Blockchain giúp xác minh quyền sở hữu và nguồn gốc của tác phẩm, ngăn chặn hành vi sao chép trái phép.

Tạo ra cộng đồng: NFT tạo ra các cộng đồng người hâm mộ xung quanh các nghệ sĩ và tác phẩm của họ.

Game:

- Vật phẩm trong game: NFT được sử dụng để tạo ra các vật phẩm trong game độc đáo và có thể giao dịch.

- Thế giới ảo: NFT tạo ra các thế giới ảo nơi người chơi có thể sở hữu đất đai, nhà cửa và các vật phẩm khác.

- Kiếm tiền từ game: Người chơi có thể kiếm tiền bằng cách mua bán các NFT trong game.

Âm nhạc:

- Album và bài hát: Nghệ sĩ có thể phát hành album và bài hát dưới dạng NFT, cho phép người hâm mộ sở hữu các bản thu âm độc đáo.

- Vé concert: Vé concert dưới dạng NFT có thể được giao dịch và chuyển nhượng.

Royalty: Nghệ sĩ có thể nhận được phần trăm lợi nhuận từ mỗi lần giao dịch NFT của tác phẩm của họ.

Thời trang:

- Bộ sưu tập thời trang kỹ thuật số: Các thương hiệu thời trang có thể tạo ra các bộ sưu tập thời trang kỹ thuật số và bán chúng dưới dạng NFT.
- Đồ vật độc quyền: NFT có thể được sử dụng để bán các sản phẩm thời trang độc quyền và có giới hạn.

Bất động sản ảo:

- Đất đai trong metaverse: NFT đại diện cho quyền sở hữu đất đai trong các thế giới ảo.
- Nhà cửa và công trình: Người dùng có thể xây dựng và sở hữu các tòa nhà, công trình trong metaverse.

Các lĩnh vực khác:

- Tài liệu: Chứng chỉ, bằng cấp, giấy tờ quan trọng có thể được token hóa thành NFT để đảm bảo tính xác thực.
- Thương hiệu: Các thương hiệu có thể sử dụng NFT để tạo ra các sản phẩm độc quyền và tương tác với khách hàng.
- Vé sự kiện: Vé tham dự các sự kiện thể thao, âm nhạc, hội nghị có thể được token hóa thành NFT.

3.2.3. Lợi ích ẩn tượng của NFT là gì?

Xác thực quyền sở hữu rõ ràng: NFT hoạt động như một "chứng chỉ số" độc nhất vô nhị, chứng minh rằng bạn là chủ sở hữu hợp pháp của một tài sản kỹ thuật số cụ thể.

- Mỗi giao dịch NFT đều được ghi lại trên blockchain, một sổ cái công khai và không thể thay đổi. Điều này tạo ra một hồ sơ giao dịch rõ ràng và minh bạch, giúp bạn dễ dàng chứng minh quyền sở hữu của mình.

- Vì mỗi NFT đều có một mã duy nhất, nên việc làm giả hoặc sao chép một NFT là gần như không thể. Điều này bảo vệ quyền sở hữu trí tuệ của người sáng tạo và đảm bảo giá trị của NFT.

Tạo ra thị trường mới: NFT đã mở ra những cánh cửa mới cho các thị trường trước đây chưa được khai thác đầy đủ.

- Nghệ sĩ giờ đây có thể bán tác phẩm nghệ thuật kỹ thuật số của mình trực tiếp cho người mua, cắt giảm chi phí trung gian và nhận được phần lớn lợi nhuận từ việc bán hàng.

- Người chơi game có thể sở hữu và giao dịch các vật phẩm trong game như một tài sản có giá trị thực, tạo ra một nền kinh tế ảo sôi động.

- NFT giúp tạo ra các bộ sưu tập độc đáo và có thể giao dịch được, từ thẻ bài đến các vật phẩm sưu tầm khác, mở ra một thị trường mới cho các nhà sưu tập.

Tăng tính minh bạch và tin cậy:

- Công nghệ blockchain là nền tảng của NFT, đảm bảo tính minh bạch và tin cậy cho mọi giao dịch. Mọi thông tin về NFT, từ việc tạo ra đến các lần giao dịch, đều được ghi lại một cách công khai và không thể xóa bỏ.

- NFT giúp loại bỏ các trung gian trong nhiều giao dịch, từ mua bán nghệ thuật đến giao dịch bất động sản ảo. Điều này giúp giảm chi phí và tăng tốc độ giao dịch.

Tạo ra cộng đồng: NFT không chỉ là về sở hữu tài sản, mà còn về việc xây dựng cộng đồng.

- NFT tạo ra các cộng đồng người hâm mộ xung quanh các nghệ sĩ, thương hiệu và dự án. Người sở hữu NFT có thể tham gia vào các sự kiện đặc biệt, nhận được các ưu đãi độc quyền và tương tác với nhau.

- Việc sở hữu một NFT mang lại cảm giác thuộc về một cộng đồng đặc biệt, nơi họ chia sẻ cùng một niềm đam mê.

Tiềm năng đầu tư:

- NFT được coi là một loại tài sản mới, mang đến cơ hội đầu tư hấp dẫn cho những người muốn đa dạng hóa danh mục đầu tư của mình.

- Giống như các tác phẩm nghệ thuật truyền thống, giá trị của NFT có thể tăng lên theo thời gian, đặc biệt là đối với các tác phẩm độc đáo và có giới

hạn.

- Một số NFT có thể mang lại thu nhập thụ động cho người sở hữu thông qua các cơ chế chia sẻ lợi nhuận hoặc staking.

Ứng dụng đa dạng: NFT có tiềm năng ứng dụng trong rất nhiều lĩnh vực khác nhau, ngoài những lĩnh vực đã đề cập ở trên.

- Vé concert, sự kiện thể thao có thể được token hóa thành NFT, tạo ra trải nghiệm độc đáo cho người hâm mộ.

- NFT có thể đại diện cho quyền sở hữu đất đai trong các metaverse, mở ra một thị trường bất động sản ảo hoàn toàn mới.

- Các thương hiệu có thể sử dụng NFT để tạo ra các sản phẩm độc quyền và tương tác với khách hàng một cách sáng tạo.

3.2.4. Điểm danh các NFT token tiềm năng hiện nay.

Hiện nay, có một số NFT token nổi bật và tiềm năng mà bạn có thể quan tâm. Dưới đây là một vài cái tên nổi bật:

- Bored Ape Yacht Club (BAYC): Một trong những bộ sưu tập NFT nổi tiếng nhất, với hình ảnh những con vượn có phong cách khác nhau. BAYC không chỉ là một bộ sưu tập NFT mà còn mang lại quyền lợi cho người sở hữu như tham gia vào các sự kiện đặc biệt.

- CryptoPunks: Đây là một trong những dự án NFT đầu tiên và vẫn giữ được giá trị rất cao. Các CryptoPunks là những hình ảnh pixel của những nhân vật khác nhau và đã trở thành biểu tượng trong thế giới NFT.

- Art Blocks: Dự án này tập trung vào việc tạo ra các tác phẩm nghệ thuật thông qua lập trình, với nhiều bộ sưu tập khác nhau từ các nghệ sĩ kỹ thuật số. Art Blocks ngày càng thu hút được sự chú ý nhờ tính độc đáo và sự sáng tạo trong từng tác phẩm.

- Axie Infinity: Một trò chơi NFT nổi tiếng cho phép người chơi mua, bán và nuôi dưỡng các sinh vật gọi là Axies. Axie Infinity đã tạo ra một cộng đồng lớn và tạo ra thu nhập thực tế cho người chơi.

- World of Women (WoW): Dự án NFT này tập trung vào việc tôn vinh

phụ nữ trong nghệ thuật kỹ thuật số, với các hình ảnh độc đáo và đầy màu sắc của phụ nữ từ nhiều nền văn hóa khác nhau. WoW đã nhận được sự quan tâm đáng kể và được xem là một trong những bộ sưu tập NFT phát triển nhanh nhất.

- Loot (for Adventurers): Đây là một dự án NFT độc đáo khi chỉ cung cấp các danh sách văn bản về trang bị trong game, mà không có hình ảnh. Tuy nhiên, Loot đã thu hút được một lượng lớn người tham gia phát triển nội dung dựa trên các danh sách này.

- Mutant Ape Yacht Club (MAYC): Là phiên bản biến thể của BAYC, với hình ảnh các con vượn bị biến đổi thành các dạng khác nhau. MAYC cũng mang lại nhiều quyền lợi cho người sở hữu, giống như BAYC.

- Pudgy Penguins: Một dự án NFT với hình ảnh những chú chim cánh cụt dễ thương, đã thu hút được một cộng đồng lớn và trở thành một trong những bộ sưu tập NFT được săn đón.

- Những NFT token này không chỉ có giá trị về mặt sưu tầm mà còn có tiềm năng về giá trị kinh tế trong tương lai. Tuy nhiên, đầu tư vào NFT cũng tiềm ẩn nhiều rủi ro, vì vậy bạn nên nghiên cứu kỹ lưỡng trước khi tham gia.

3.2.5. Lạm dụng NFT quá mức sẽ gây ra nguy cơ tiềm ẩn nào?

Lạm dụng NFT quá mức có thể gây ra một số nguy cơ tiềm ẩn như sau:

- Bong bóng thị trường: Giống như những tài sản khác, NFT có thể tạo ra một bong bóng đầu cơ. Khi giá trị của các NFT tăng lên nhanh chóng và không bền vững, thị trường có thể trải qua một đợt sụp đổ đột ngột khi bong bóng vỡ, gây thiệt hại lớn cho các nhà đầu tư.

- Thiếu thanh khoản: Mặc dù một số NFT được bán với giá trị cao, nhưng không phải tất cả NFT đều có thanh khoản tốt. Nhiều NFT có thể khó bán lại, dẫn đến việc người mua không thể thu hồi vốn đầu tư hoặc chỉ bán được với giá thấp hơn nhiều so với giá mua ban đầu.

- Vấn đề bản quyền và quyền sở hữu: NFT không phải lúc nào cũng đảm bảo quyền sở hữu hoặc bản quyền thực sự của tác phẩm gốc. Việc này có

thể dẫn đến các tranh chấp pháp lý và gây ra rủi ro cho người mua nếu họ không nắm rõ các quyền đi kèm với NFT.

- Tác động môi trường: Nhiều blockchain mà NFT được xây dựng trên đó, chẳng hạn như Ethereum, tiêu tốn năng lượng rất lớn cho quá trình xác thực giao dịch. Việc phát hành và giao dịch NFT có thể góp phần vào tác động tiêu cực đối với môi trường, gây tranh cãi và phản đối từ cộng đồng.

- Rủi ro pháp lý: Khi thị trường NFT phát triển, các cơ quan quản lý có thể tăng cường kiểm soát, đặc biệt liên quan đến vấn đề rửa tiền, gian lận và tuân thủ quy định về chứng khoán. Những người tham gia vào thị trường NFT có thể phải đối mặt với các rủi ro pháp lý nếu không tuân thủ các quy định này.

- Vấn đề đạo đức và lừa đảo: Có nhiều trường hợp các tác phẩm nghệ thuật được biến thành NFT mà không có sự cho phép của tác giả gốc. Ngoài ra, việc phát hành các dự án NFT lừa đảo hoặc không có giá trị thực sự cũng gia tăng, khiến người dùng có thể bị mất tiền.

- Phân hóa giàu nghèo: NFT đang tạo ra một tầng lớp nhà đầu tư mới giàu có nhờ vào sự tăng giá đột ngột của các tài sản số. Điều này có thể dẫn đến sự bất bình đẳng lớn hơn trong xã hội, khi mà chỉ một số ít người có khả năng tiếp cận và hưởng lợi từ thị trường này.

- Mất mát tài sản số: Vì NFT được lưu trữ trên các ví điện tử, mất quyền truy cập vào ví (do quên mật khẩu, mất khóa riêng tư, hoặc sự cố kỹ thuật) có thể dẫn đến mất mát tài sản không thể phục hồi.

Việc lạm dụng hoặc đầu tư vào NFT mà không có sự hiểu biết đầy đủ có thể mang lại rủi ro lớn. Người tham gia thị trường này cần cẩn thận và có chiến lược đầu tư hợp lý để tránh những hậu quả tiêu cực.

3.2.6. Các thuật ngữ liên quan đến NFT.

Một số thuật ngữ quan trọng liên quan đến NFT (Non-Fungible Token):

- NFT (Non-Fungible Token): Một loại token số đại diện cho quyền sở hữu duy nhất của một tài sản kỹ thuật số hoặc vật lý. Không thể thay thế hoặc

hoán đổi lẫn nhau như các loại tiền điện tử khác.

- Blockchain: Công nghệ số cái phân tán mà NFT được xây dựng trên đó. Blockchain đảm bảo tính bảo mật, minh bạch và không thể thay đổi của các giao dịch NFT.

- Minting: Quá trình tạo ra một NFT mới và ghi nó vào blockchain. Khi một tài sản kỹ thuật số được "minted", nó sẽ trở thành một phần của blockchain với một mã định danh duy nhất.

- Smart Contract: Các hợp đồng thông minh là các chương trình tự động thực hiện các điều khoản và điều kiện đã được định trước khi các điều kiện đó được đáp ứng. NFT thường được quản lý thông qua các hợp đồng thông minh trên blockchain.

- ERC-721: Một tiêu chuẩn trên blockchain Ethereum dành cho việc tạo ra các NFT. Mỗi token theo tiêu chuẩn này là duy nhất và không thể thay thế lẫn nhau.

- ERC-1155: Một tiêu chuẩn trên Ethereum cho phép tạo ra cả NFT và token có thể thay thế được trong cùng một hợp đồng. Nó tiết kiệm chi phí giao dịch và tạo điều kiện cho việc quản lý nhiều loại tài sản số.

- Gas Fee: Phí giao dịch mà người dùng phải trả khi thực hiện các giao dịch trên blockchain, chẳng hạn như khi minting, mua hoặc bán NFT. Phí này thường được tính bằng đơn vị tiền mã hóa của blockchain đó (ví dụ: Ether trên Ethereum).

- Marketplace: Các sàn giao dịch trực tuyến nơi người dùng có thể mua, bán và giao dịch NFT. Ví dụ nổi tiếng bao gồm OpenSea, Rarible, và SuperRare.

- Metaverse: Một thế giới ảo nơi các NFT có thể được sử dụng, chẳng hạn như để sở hữu đất đai kỹ thuật số, tài sản trong game, hoặc các bộ sưu tập kỹ thuật số.

- Rarity: Mức độ hiếm có của một NFT, thường quyết định giá trị của nó. Các NFT càng hiếm thì càng có giá trị cao.

- Wallet: Ví kỹ thuật số dùng để lưu trữ NFT và các loại tiền mã hóa. Ví có thể là phần mềm, phần cứng hoặc dựa trên đám mây.

- Floor Price: Giá thấp nhất mà một NFT có thể được mua trên thị trường. Đây là chỉ số quan trọng để đánh giá giá trị của một bộ sưu tập NFT.

- Gas War: Xảy ra khi nhiều người dùng cố gắng thực hiện giao dịch trên blockchain cùng lúc, dẫn đến việc phí gas tăng cao do cạnh tranh.

- Royalties: Phần trăm doanh thu mà người sáng tạo nhận được từ mỗi lần bán lại NFT trên thị trường thứ cấp. Điều này được quản lý tự động bởi hợp đồng thông minh.

- Airdrop: Quá trình phát tặng NFT miễn phí cho các ví của người dùng, thường được thực hiện bởi các dự án mới để quảng bá sản phẩm hoặc tạo dựng cộng đồng.

- Fractional Ownership: Khái niệm chia nhỏ quyền sở hữu của một NFT thành các phần nhỏ hơn, cho phép nhiều người cùng sở hữu một phần của NFT đó.

3.2.7. Tiêu chuẩn NFT Token ERC-721 và ERC-1155.

ERC-721 và ERC-1155 là hai tiêu chuẩn chính dành cho NFT (Non-Fungible Token) trên blockchain Ethereum. Mỗi tiêu chuẩn có những đặc điểm riêng biệt, phù hợp với các nhu cầu và ứng dụng khác nhau trong việc tạo và quản lý NFT.

3.2.7.1. ERC-721.

Tổng quan:

- ERC-721 là tiêu chuẩn đầu tiên được tạo ra cho NFT trên blockchain Ethereum, được giới thiệu bởi William Entriken và một nhóm các nhà phát triển vào năm 2018.

- ERC-721 cho phép tạo ra các token duy nhất, không thể thay thế lẫn nhau. Điều này có nghĩa là mỗi token có một giá trị riêng biệt và không thể hoán đổi một cách trực tiếp với bất kỳ token nào khác.

Đặc điểm chính:

- Tính duy nhất: Mỗi ERC-721 token có một mã định danh duy nhất (tokenID), giúp phân biệt nó với các token khác.

- Tính không thể thay thế: ERC-721 token không thể thay thế bằng một token khác với giá trị tương đương, vì mỗi token có các thuộc tính và giá trị riêng biệt.

- Chuyển nhượng: Các token ERC-721 có thể được chuyển nhượng từ người này sang người khác trên blockchain, với các điều kiện được quy định bởi hợp đồng thông minh.

- Ứng dụng: ERC-721 thường được sử dụng để tạo ra các tài sản số duy nhất như tác phẩm nghệ thuật, vật phẩm trong game, đồ sưu tầm, và bất động sản kỹ thuật số.

3.2.7.2. ERC-1155.

Tổng quan:

- ERC-1155 là một tiêu chuẩn mới hơn, được giới thiệu bởi Enjin vào năm 2019. Nó được thiết kế để cải thiện một số hạn chế của ERC-721 và cung cấp tính linh hoạt cao hơn.

- ERC-1155 cho phép tạo ra cả NFT (không thể thay thế) và FT (có thể thay thế) trong cùng một hợp đồng thông minh.

Đặc điểm chính:

- Hỗ trợ đa dạng tài sản: Với ERC-1155, cùng một hợp đồng thông minh có thể quản lý nhiều loại token khác nhau, bao gồm cả NFT và token có thể thay thế được.

- Tính tiết kiệm gas: ERC-1155 cho phép thực hiện nhiều giao dịch trong một lệnh gọi duy nhất, giúp giảm chi phí gas so với ERC-721, nơi mỗi token yêu cầu một giao dịch riêng biệt.

- Batch Transfer: ERC-1155 hỗ trợ việc chuyển nhiều token khác nhau trong một lần giao dịch, làm tăng hiệu quả và giảm phí giao dịch.

Ứng dụng: ERC-1155 thường được sử dụng trong các trò chơi và các ứng dụng cần quản lý nhiều loại tài sản kỹ thuật số khác nhau, chẳng hạn như

vũ khí, tiền tệ trong game, vật phẩm sưu tập, và các tài sản khác.

Bảng 3.1. Bảng so sánh tiêu chuẩn ERC-721 và ERC-1155.

Đặc điểm	ERC-721	ERC-1155
Loại tài sản	Chỉ hỗ trợ NFT	Hỗ trợ cả NFT và FT
Độ độc nhất	Mỗi token là duy nhất	Có thể tạo nhiều bản sao hoặc các token khác nhau trong cùng một hợp đồng
Hiệu quả gas	Mỗi token yêu cầu một giao dịch riêng	Hỗ trợ chuyển tiền nhiều token trong một giao dịch
Ứng dụng	Nghệ thuật số, đồ sưu tầm, bất động sản	Game, tài sản hỗn hợp(NFT và FT), tài sản trong game

3.3. Quy trình phát triển NFT.

3.3.1. Hợp đồng Thông minh NFT (Smart Contracts).

Do tính chất không thể thay thế, mỗi NFT là duy nhất và chỉ một người có thể sở hữu nó tại một thời điểm, khác với các token tiền điện tử có sẵn hàng tỷ. Do đó, NFT thường được gắn liền với một mã định danh token (token ID) hoặc metadata, lưu trữ bằng chứng về quyền sở hữu. Đây là lúc hợp đồng thông minh xuất hiện. Tài liệu kỹ thuật số tự thực thi này kiểm soát và phân công quyền sở hữu cho NFT, cho phép bạn tạo, giao dịch hoặc phá hủy nó (chúng ta sẽ đề cập sau). Nói cách khác, NFT không thể tồn tại mà không có hợp đồng thông minh. NFT và hợp đồng thông minh gắn kết với nhau. Sự ra đời của hợp đồng thông minh đã tạo ra một thị trường mạnh mẽ và ngày càng phát triển cho NFT và các sàn giao dịch NFT ngày nay.

3.3.2. Tạo smart contract cho NFT.

Hầu hết các hợp đồng thông minh NFT hiện tại đều dựa trên hai tiêu chuẩn phổ biến của blockchain Ethereum, ERC 721 và ERC 1155.

ERC721 (Ethereum Request for Comment 721) là giao thức đầu tiên cho NFT trên Ethereum. Tiêu chuẩn này yêu cầu các token phải là duy nhất và không thể thay thế, trong khi các tiêu chuẩn khác không yêu cầu.

ERC-1155 (Ethereum Request for Comments 1155) là sự kết hợp của ERC-20 và ERC-721. Nó tạo ra cả token có thể thay thế và không thể thay thế trong một hợp đồng thông minh duy nhất. Tiêu chuẩn này mang lại nhiều lợi thế nổi bật so với các tiêu chuẩn của nó, đặc biệt là tính linh hoạt và tiết kiệm gas (nó tuyên bố cắt giảm phí gas lên đến 90% khi mint token mới). Nó được sử dụng chủ yếu trong các trò chơi NFT.

Bây giờ, đây là các bước để viết một hợp đồng thông minh NFT tốt:

- B1: Chọn loại NFT của bạn: ERC 721 hoặc ERC 1155.
- B2: Chọn chuỗi (chain) để triển khai hợp đồng thông minh của bạn.
- B3: Chọn nơi lưu trữ metadata của NFT: on-chain hoặc off-chain.
- B4: Chọn bộ công nghệ của bạn và mã hóa hợp đồng thông minh.
- B5: Kiểm tra các yêu cầu bảo mật của bạn bao gồm các công cụ và phương pháp kiểm tra.
- B6: Kiểm tra và tối ưu hóa việc sử dụng gas nếu cần.
- B7: Triển khai nó, đảm bảo chức năng minting và admin hoạt động đúng.

3.3.3. Triển khai smart contract cho NFT.

Bước 1: Kết nối với Blockchain:

- Thiết lập môi trường: Cài đặt các công cụ cần thiết như Node.js, Truffle hoặc Hardhat.
- Tạo tài khoản trên dịch vụ kết nối blockchain: Đăng ký tài khoản trên Infura hoặc Alchemy và tạo một dự án mới để nhận URL endpoint.
- Cấu hình dự án: Cấu hình Truffle hoặc Hardhat để sử dụng endpoint

của Infura hoặc Alchemy.

Bước 2: Triển khai hợp đồng thông minh:

- Viết script triển khai: Tạo một file migration mới trong thư mục migrations của Truffle.

- Chạy lệnh triển khai: Sử dụng Truffle CLI để triển khai hợp đồng.

Bước 3: Xác minh triển khai:

- Kiểm tra trên trình duyệt blockchain: Sử dụng Etherscan hoặc trình duyệt blockchain tương ứng để xác minh rằng hợp đồng đã được triển khai đúng cách.

- Chạy các bài kiểm tra: Sử dụng các công cụ như Truffle hoặc Hardhat để chạy các bài kiểm tra tự động.

Bước 4: Cấu hình các chức năng quản trị:

- Tạo các chức năng quản trị: Viết các hàm trong smart contract để quản lý quyền sở hữu, minting, và các chức năng đặc biệt khác.

3.3.4. Minting, Quản lý và Giao dịch NFT.

Bước 1: Minting NFT:

- Viết script minting: Tạo một script để mint các NFT mới.

- Chạy lệnh minting: Sử dụng Truffle CLI để chạy script minting.

Bước 2: Quản lý NFT:

- Metadata và lưu trữ: Quyết định lưu trữ metadata on-chain hay off-chain. Off-chain thường dùng các dịch vụ như IPFS hoặc Arweave.

Bước 3: Giao dịch NFT:

- Phát triển giao diện người dùng (UI): Sử dụng React và web3.js để tạo UI cho việc hiển thị, mua bán, và chuyển nhượng NFT.

Bước 4: Tích hợp ví điện tử:

- Tích hợp MetaMask: Thêm hỗ trợ MetaMask để người dùng có thể đăng nhập và tương tác với hợp đồng thông minh.

Bước 5: Kiểm tra và tối ưu hóa:

- Kiểm tra kỹ lưỡng: Viết và chạy các bài kiểm tra tự động để đảm bảo

tất cả các chức năng hoạt động đúng.

- Tối ưu hóa gas: Kiểm tra và tối ưu hóa việc sử dụng gas để giảm chi phí giao dịch.

Bước 6: Hỗ trợ và bảo trì:

- Cung cấp hỗ trợ người dùng: Đảm bảo rằng người dùng có thể nhận hỗ trợ khi gặp vấn đề về mua, bán, hoặc chuyển nhượng NFT.

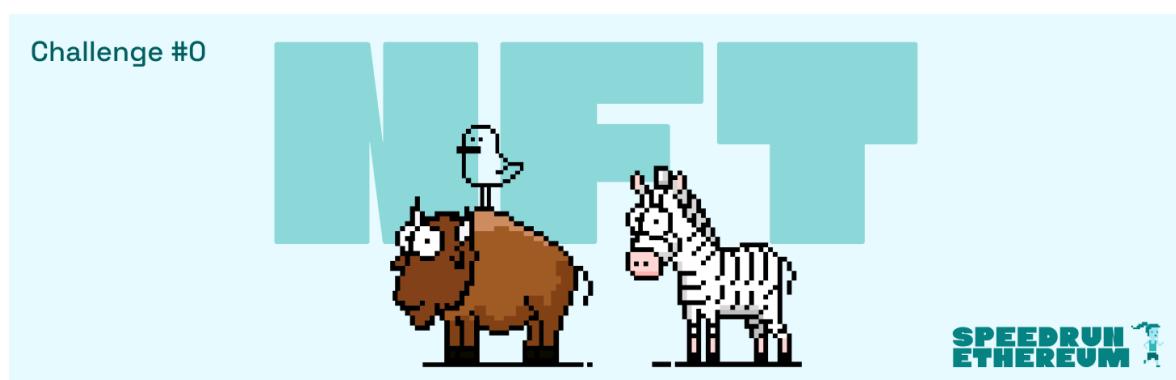
- Bảo trì hợp đồng và ứng dụng: Thường xuyên cập nhật và bảo trì hợp đồng thông minh và giao diện người dùng để đảm bảo hoạt động mượt mà và bảo mật.

3.4. Phát triển ứng dụng thực tế.

3.4.1. Giới thiệu.

“challenge-0-simple-nft” là dự án ứng dụng web dùng để giao dịch NFT đơn giản. Dự án này giúp người dùng học cơ bản về scaffold-eth và biến đổi hợp đồng thông minh.

Github: https://github.com/hoangpt21/BTL_NHOM09_LOP20241IT6024001.git



Hình 3.4. Dự án challenge-0-simple-nft.

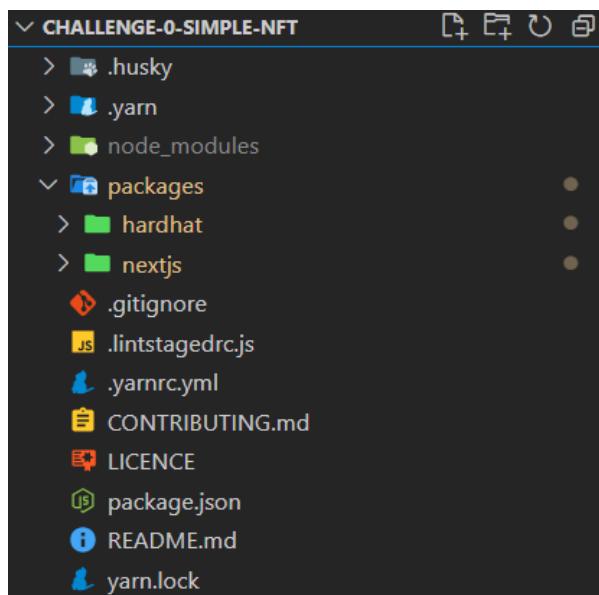
Ta sẽ tiến hành biên dịch và triển khai hợp đồng thông minh đầu tiên. Sau đó, sử dụng một ứng dụng React mẫu được tích hợp đầy đủ các thành phần và hook quan trọng của Ethereum. Cuối cùng, ta sẽ triển khai một NFT lên mạng công khai để chia sẻ với bạn bè!

Kết quả cuối cùng sẽ là một ứng dụng cho phép người dùng mua và chuyển NFT. Ta có thể triển khai hợp đồng lên testnet, sau đó xây dựng và tải

ứng dụng lên một máy chủ web công khai. Thực hiện gửi đường dẫn URL lên [SpeedRunEthereum.com!](https://SpeedRunEthereum.com)

3.4.2. Cấu trúc dự án.

Dưới đây là phân tích về cấu trúc của dự án:

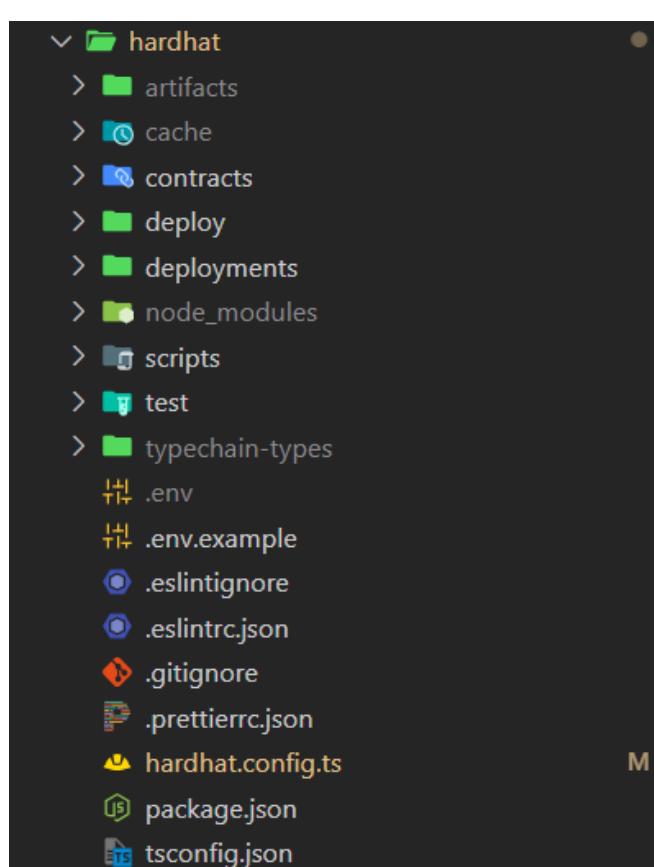


Hình 3.5. Cấu trúc dự án tổng thể.

Trong đó:

- **.husky**: Chứa các hook của Git, dùng để tự động hóa các tác vụ trước hoặc sau khi thực hiện các lệnh Git (như pre-commit, pre-push). Ví dụ, kiểm tra mã nguồn trước khi commit.
- **.yarn**: Chứa các tệp liên quan đến Yarn (một công cụ quản lý package). Có thể bao gồm cấu hình cho Yarn Plug'n'Play (PnP).
- **node_modules**: Chứa các thư viện và phụ thuộc của dự án được cài đặt thông qua Yarn hoặc npm.
- **packages**: Đây là nơi chứa các gói (packages) con của dự án:
 - + **hardhat**: Một framework để phát triển, triển khai và kiểm tra hợp đồng thông minh trên Ethereum.
 - + **nextjs**: Một framework React dùng để xây dựng giao diện phía client, hỗ trợ server-side rendering và static site generation.
- **.gitignore**: Tệp cấu hình để chỉ định các tệp/thư mục không được theo dõi bởi Git.

- **.lintstagedrc.js**: Cấu hình cho lint-staged, công cụ giúp lint (kiểm tra code) chỉ trên các tệp được staged để commit.
- **.yarnrc.yml**: Cấu hình của Yarn, ví dụ: thiết lập registry, cache, hoặc các plugin.
- **CONTRIBUTING.md**: Hướng dẫn đóng góp vào dự án, dành cho các nhà phát triển khác.
- **LICENSE**: Tệp chứa thông tin về bản quyền sử dụng của dự án.
- **package.json**: Tệp quản lý thông tin của dự án, bao gồm:
 - + Tên, phiên bản, mô tả.
 - + Các phụ thuộc (dependencies, devDependencies).
 - + Các script để chạy các lệnh.
- **README.md**: Tệp tài liệu chính giới thiệu về dự án, cách cài đặt, sử dụng.
- **yarn.lock**: Tệp lockfile của Yarn, dùng để đảm bảo các phụ thuộc được cài đặt đồng nhất trên mọi môi trường.



Hình 3.6. Cấu trúc thư mục và tệp trong thư mục hardhat.

Trong đó:

- **artifacts/**: Chứa các tệp biên dịch (compiled contracts) của Hardhat, bao gồm ABI (Application Binary Interface) và mã bytecode. Đây là kết quả khi biên dịch các hợp đồng thông minh.
- **cache/**: Lưu trữ các tệp tạm thời để tăng tốc quá trình biên dịch và triển khai hợp đồng. Có thể xóa an toàn nếu cần làm mới.
- **contracts/**: Nơi chứa mã nguồn các hợp đồng thông minh viết bằng Solidity (thường có phần mở rộng .sol). Đây là nơi bạn viết các hợp đồng thông minh chính của mình.
- **deploy/**: Chứa các script triển khai hợp đồng thông minh. Thường được viết bằng JavaScript hoặc TypeScript để tự động hóa việc triển khai lên các mạng blockchain.
- **deployments/**: Lưu trữ các thông tin triển khai, chẳng hạn như địa chỉ hợp đồng đã triển khai trên các mạng blockchain (ví dụ: testnet hoặc mainnet).
- **node_modules/**: Chứa các thư viện và phụ thuộc được cài đặt qua npm hoặc yarn.
- **scripts/**: Chứa các script bổ sung, chẳng hạn như script để kiểm tra, tương tác với hợp đồng, hoặc thực hiện các tác vụ cụ thể.
- **test/**: Nơi lưu trữ các tệp kiểm tra (test files) cho hợp đồng thông minh. Thường sử dụng các thư viện như Mocha hoặc Chai để viết và chạy kiểm tra.
- **typechain-types/**: Chứa các tệp TypeScript được tự động tạo từ các hợp đồng thông minh thông qua **TypeChain**. Giúp hỗ trợ lập trình kiểu tĩnh (type-safe) khi làm việc với hợp đồng thông minh trong TypeScript.
- **.env / .env.example:**
 - + **.env**: Chứa các biến môi trường như khóa API, URL mạng blockchain.
 - + **.env.example**: Mẫu file .env để hướng dẫn người dùng.

- **.eslintignore / .eslintrc.json:** Cấu hình cho ESLint, dùng để kiểm tra chất lượng và style code.

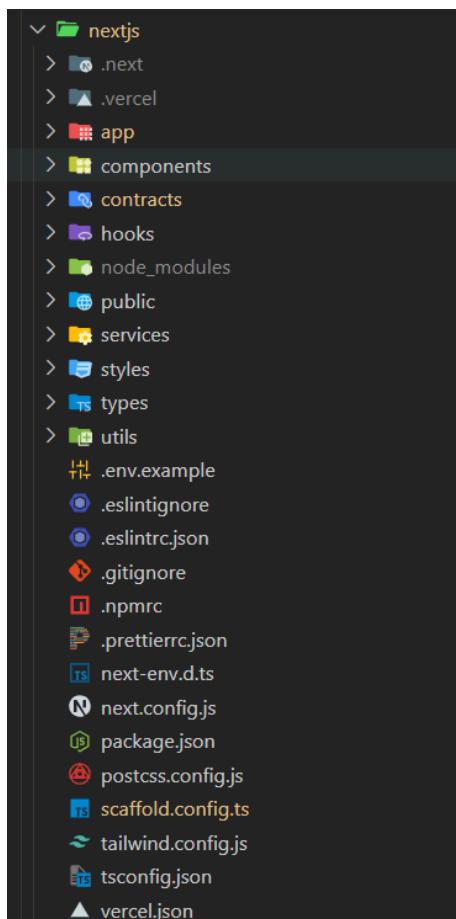
- **.gitignore:** Liệt kê các tệp hoặc thư mục không được theo dõi bởi Git (ví dụ: node_modules, artifacts).

- **.prettierrc.json:** Cấu hình cho Prettier, công cụ định dạng code tự động.

- **hardhat.config.ts:** Tệp cấu hình chính của Hardhat, nơi thiết lập mạng blockchain, plugin, và thông số liên quan đến dự án.

- **package.json:** Tệp quản lý thông tin dự án Node.js, liệt kê các phụ thuộc và script.

- **tsconfig.json:** Cấu hình cho TypeScript, bao gồm các tùy chọn biên dịch (compiler options).



Hình 3.7. Cấu trúc thư mục và tệp trong thư mục nextjs.

Trong đó:

- **.next**: Chứa các tệp build output và cache được tạo ra bởi Next.js.
- **.vercel**: Chứa các tệp cấu hình để triển khai dự án trên Vercel.
- **app**: Có thể chứa mã nguồn chính của ứng dụng.
- **components**: Chứa các component React có thể tái sử dụng.
- **contracts**: Chứa các tệp liên quan đến hợp đồng thông minh, nếu dự án là về blockchain.
- **hooks**: Chứa các custom hooks của React.
- **node_modules**: Chứa tất cả các package npm được cài đặt cho dự án.
- **public**: Chứa các tệp tĩnh như hình ảnh, font chữ, v.v.
- **services**: Chứa các tệp liên quan đến dịch vụ, chẳng hạn như các lời gọi API.
- **styles**: Chứa các tệp CSS hoặc các tệp định dạng khác.
- **types**: Chứa các định nghĩa type của TypeScript.
- **utils**: Chứa các hàm tiện ích.
- **.env.example**: Tệp ví dụ cho các biến môi trường.
- **.eslintignore**: Định rõ các tệp và thư mục mà ESLint nên bỏ qua.
- **.eslintrc.json**: Tệp cấu hình cho ESLint.
- **.gitignore**: Định rõ các tệp và thư mục mà Git nên bỏ qua.
- **.npmrc**: Tệp cấu hình cho npm.
- **.prettierrc.json**: Tệp cấu hình cho Prettier.
- **next-env.d.ts**: Tệp khai báo TypeScript cho Next.js.
- **next.config.js**: Tệp cấu hình cho Next.js.
- **package.json**: Chứa metadata về dự án và các phụ thuộc của nó.
- **postcss.config.js**: Tệp cấu hình cho PostCSS.
- **scaffold.config.ts**: Có thể là một tệp cấu hình tùy chỉnh, có thể cho việc tạo scaffold.
 - **tailwind.config.js**: Tệp cấu hình cho Tailwind CSS.
 - **tsconfig.json**: Tệp cấu hình cho TypeScript.
 - **vercel.json**: Tệp cấu hình để triển khai dự án trên Vercel.

3.4.3. Triển khai dự án.

3.4.3.1. Chạy dự án trên máy tính bằng mạng cục bộ.

a. Giới thiệu về hardhat.

Hardhat là một trong những công cụ phổ biến nhất trong việc phát triển các ứng dụng phi tập trung (dApps) trên nền tảng Ethereum. Hardhat cung cấp một môi trường phát triển toàn diện, cho phép các nhà phát triển tạo, thử nghiệm, triển khai và quản lý các hợp đồng thông minh một cách hiệu quả.

Một số tính năng chính của Hardhat bao gồm:

- Môi trường phát triển cục bộ: Hardhat cho phép bạn chạy một mạng Ethereum cục bộ trên máy tính của mình, giúp bạn có thể thử nghiệm và gỡ lỗi các hợp đồng thông minh mà không cần phải kết nối với mạng chính thức của Ethereum.

- Tích hợp với các công cụ khác: Hardhat có thể tích hợp với các công cụ khác như Truffle, Web3.js, và Ethers.js, giúp bạn có thể tận dụng các tính năng của các công cụ này trong quá trình phát triển.

- Hỗ trợ nhiều loại mạng: Hardhat hỗ trợ nhiều loại mạng khác nhau, bao gồm cả mạng chính thức của Ethereum, mạng thử nghiệm (testnet), và mạng cục bộ (localhost).

- Tích hợp với các trình soạn thảo mã: Hardhat có thể tích hợp với các trình soạn thảo mã như Visual Studio Code, giúp bạn có thể tận dụng các tính năng của trình soạn thảo mã trong quá trình phát triển.



Hình 3.8. Logo Hardhat.

b. Chạy dự án.

Trước tiên, cần đảm bảo cài đặt các công cụ cần thiết sau:

- Node.js (phiên bản $\geq v18.17$)

- Yarn (phiên bản 1 hoặc 2+)

- Git

Các bước để chạy thực thi dự án “challenge-0-simple-nft” bằng mạng cục bộ (localhost):

- Bước 1: Mở công cụ cmd và điều hướng đường dẫn đến thư mục dự án “challenge-0-simple-nft”, thực hiện chạy lệnh “yarn chain” để khởi động mạng blockchain Ethereum cục bộ.

```
D:\ADMIN\Downloads\challenge-0-simple-nft>yarn chain
WARNING: You are currently using Node.js v21.7.2, which is not supported by Hardhat. This can lead to unexpected behavior. See https://hardhat.org/nodejs-versions

Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/
Accounts
=====
```

Hình 3.9. Chạy lệnh khởi động mạng blockchain Ethereum cục bộ.

- Bước 2: Tiếp tục mở công cụ cmd thứ 2 và điều hướng đường dẫn đến thư mục dự án “challenge-0-simple-nft”, thực hiện chạy lệnh “yarn deploy” để triển khai hợp đồng thông minh (Smart Contract) cục bộ.

```
D:\ADMIN\Downloads\challenge-0-simple-nft>yarn deploy
WARNING: You are currently using Node.js v21.7.2, which is not supported by Hardhat. This can lead to unexpected behavior. See https://hardhat.org/nodejs-versions

Nothing to compile
No need to generate any newer typings.
deploying "YourCollectible" (tx: 0xf4e4591a6d25190e61e2c8bc5ab981a8103dac27382590807eed0de04c5580d7)...: deployed at 0x5FbDB2315678afech367f032d93F642f64180aa3 with 1758831 gas
📝 Updated TypeScript contract definition file on ../nextjs/contracts/deployedContracts.ts
```

Hình 3.10. Chạy lệnh triển khai hợp đồng thông minh cục bộ.

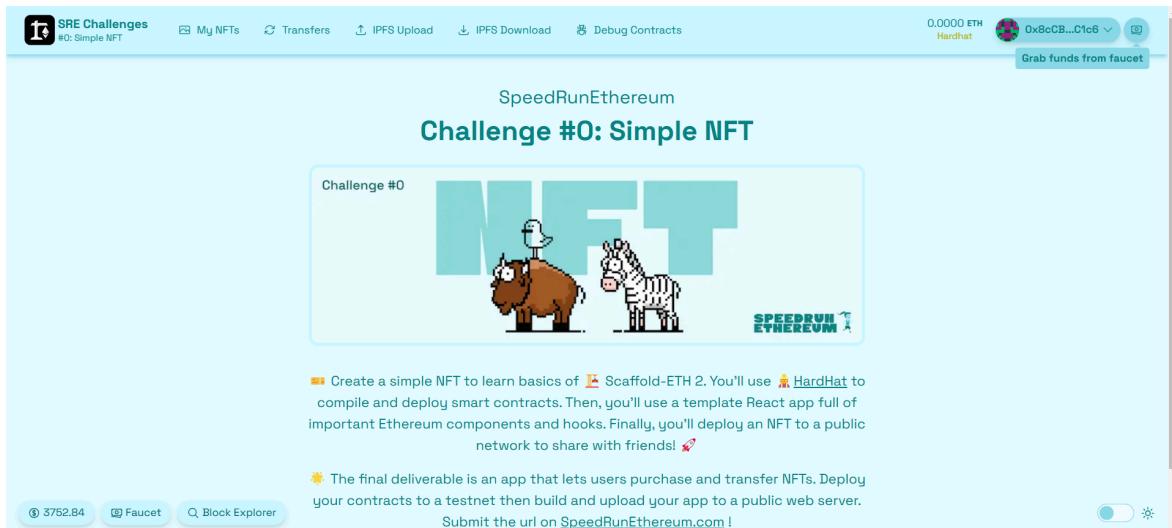
- Bước 3: Tiếp tục mở công cụ cmd thứ 3 và điều hướng đường dẫn đến thư mục dự án “challenge-0-simple-nft”, thực hiện chạy lệnh “yarn start” để khởi chạy giao diện frontend.

```
D:\ADMIN\Downloads\challenge-0-simple-nft>yarn start
  ▲ Next.js 14.1.0
    - Local:      http://localhost:3000

  ✓ Ready in 6.5s
```

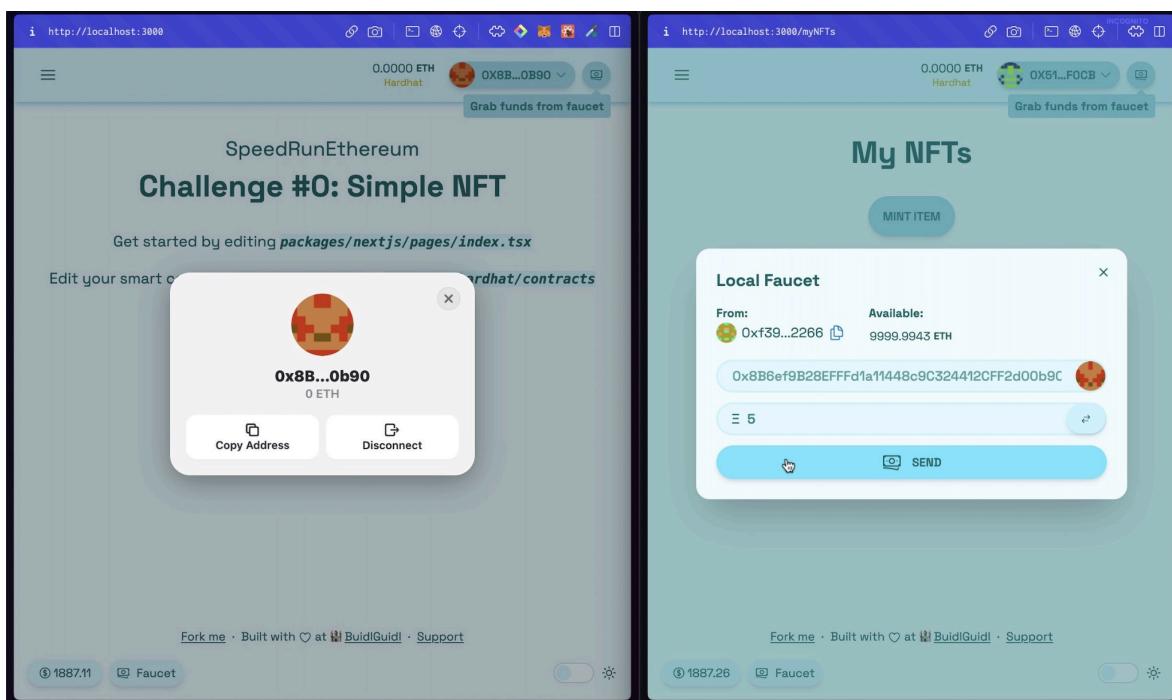
Hình 3.11. Chạy lệnh triển khai giao diện frontend.

Mở `http://localhost:3000` trên trình duyệt để tương tác với ứng dụng.



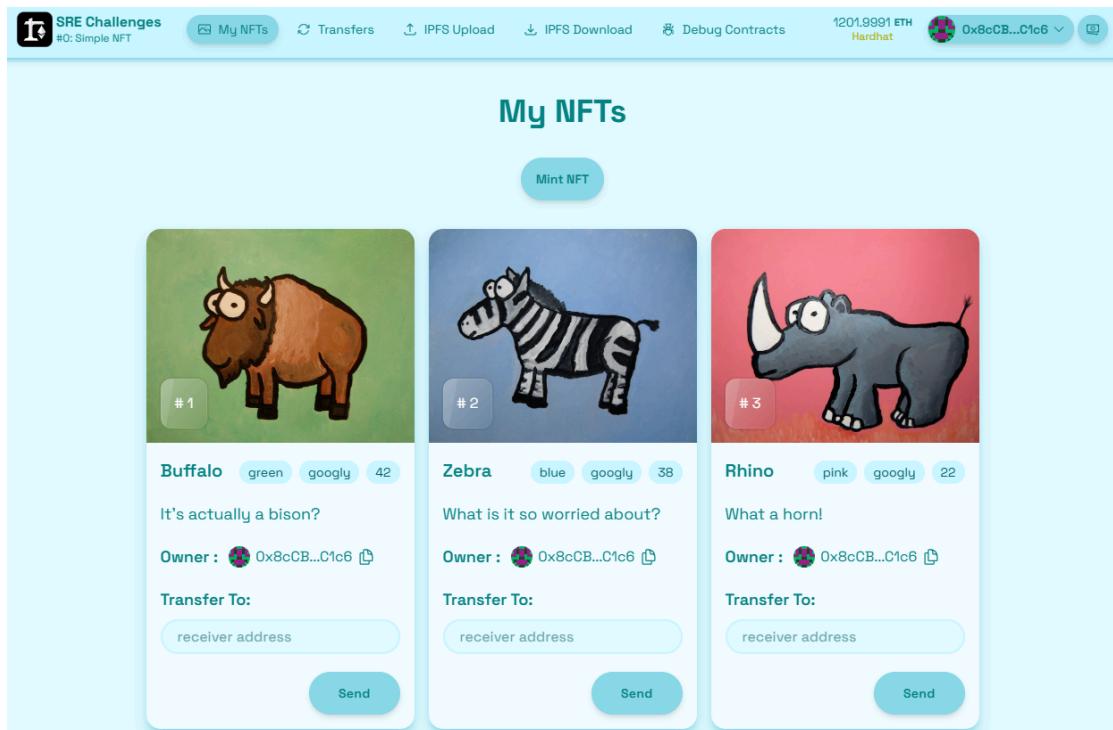
Hình 3.12. Giao diện web của ứng dụng.

Tiếp theo ta có thể thực hiện một số giao dịch trên mạng cục bộ này.

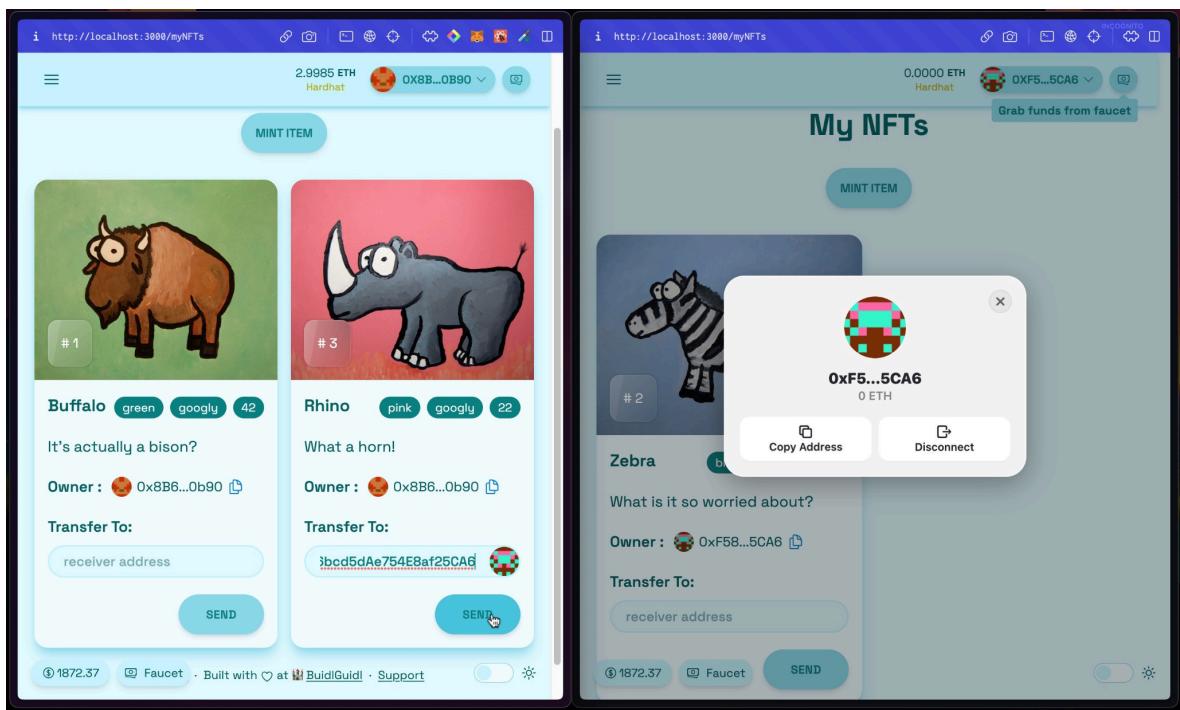


Hình 3.13. Các giao dịch gửi tiền qua lại bằng địa chỉ ví.

Hình 3.13 cho ta thấy việc chuyển tiền qua lại bằng địa chỉ ví. Ví Burner sẽ tự động tạo địa chỉ mới. Sao chép địa chỉ từ cửa sổ ẩn danh và sử dụng Faucet để gửi tiền từ trình duyệt chính. Tuy nhiên, tài khoản Burner sẽ mất khi đóng cửa sổ ẩn danh. Ví Burner rất hữu ích khi phát triển cục bộ, nhưng khi làm việc với mạng công khai, ta thường sử dụng các ví vĩnh viễn.



Hình 3.14. Các NFT được sinh ra sau khi được đúc.



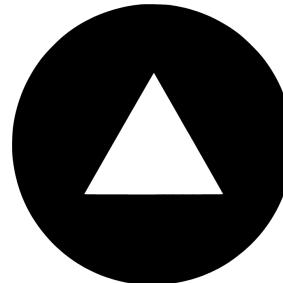
Hình 3.15. Các giao dịch NFT qua lại bằng địa chỉ ví.

3.4.3.2. Triển khai dự án trên vercel.

a. Giới thiệu về vercel.

Trong phần này ta sẽ cùng triển khai dự án lên Vercel. Vercel là nền tảng dịch vụ triển khai và lưu trữ ứng dụng web (Deployment & Hosting)

được xây dựng dành riêng cho các lập trình viên và nhóm phát triển phần mềm. Vercel tập trung vào việc hỗ trợ các ứng dụng front-end, đặc biệt là những ứng dụng sử dụng các framework hiện đại như Next.js, React, Angular, Vue, và Svelte.



Hình 3.16. Logo Vercel.

b, Triển khai dự án.

Các bước triển khai dự án “challenge-0-simple-nft” lên Vercel:

1) Triển khai hợp đồng:

- Chuẩn bị triển khai dự án lên mạng công khai: Chính sửa thuộc tính defaultNetwork trong tệp “packages/hardhat/hardhat.config.ts” thành sepolia (hoặc optimismSepolia để giảm phí gas).

```
const config: HardhatUserConfig = {
  solidity: "0.8.2",
  defaultNetwork: "sepolia",
```

Hình 3.17. Chính sửa thuộc tính defaultNetwork trong hardhat.config.ts.

- Tạo địa chỉ deployer: Sử dụng lệnh “yarn generate” để tạo một địa chỉ deployer độc nhất và lưu mnemonic cục bộ. Sau đó, kiểm tra số dư tài khoản deployer bằng lệnh “yarn account”

- Cấp ETH cho địa chỉ deployer: Sử dụng faucet trên mạng Sepolia hoặc Optimism Sepolia để nạp ETH thử nghiệm.

- Triển khai hợp đồng: Chạy lệnh “yarn deploy”. Có thể chỉ định mạng bằng cách thêm “--network sepolia” hoặc “--network optimismSepolia”.

2) Phát hành giao diện frontend:

- Chính sửa cấu hình frontend: Thay đổi thuộc tính targetNetwork trong

tệp có đường dẫn “packages/nextjs/scaffold.config.ts” thành chains.sepolia (hoặc chains.optimismSepolia nếu bạn dùng Optimism Sepolia).

```
const scaffoldConfig = {
  // The networks on which your DApp is live
  targetNetworks: [chains.sepolia],
```

Hình 3.18. Chính sửa thuộc tính defaultNetwork trong scaffold.config.ts.

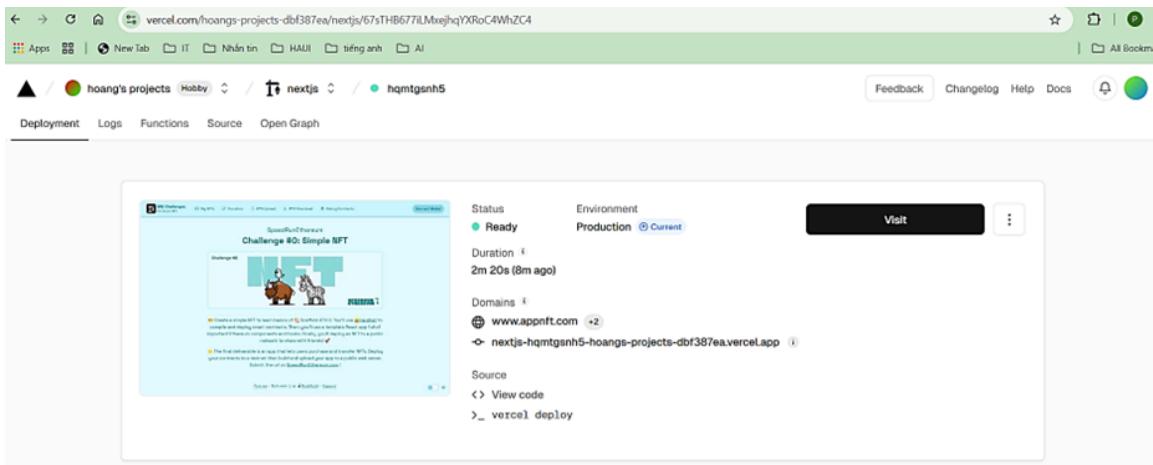
- Kết nối ví: Nếu triển khai lên mạng công khai, ta cần sử dụng ví của mình hoặc bật ví Burner cho tất cả mạng bằng cách chỉnh sửa thuộc tính “onlyLocalBurnerWallet: false” trong tệp “scaffold.config.ts”.

- Tối ưu hóa hiệu suất ứng dụng: Trong tệp page.tsx có đường dẫn là “packages/nextjs/app/transfers/page.tsx”, chỉnh sửa tham số fromBlock trong useScaffoldEventHistory để chỉ tải dữ liệu từ block khi hợp đồng của bạn được triển khai (VD: fromBlock: 3750241n).

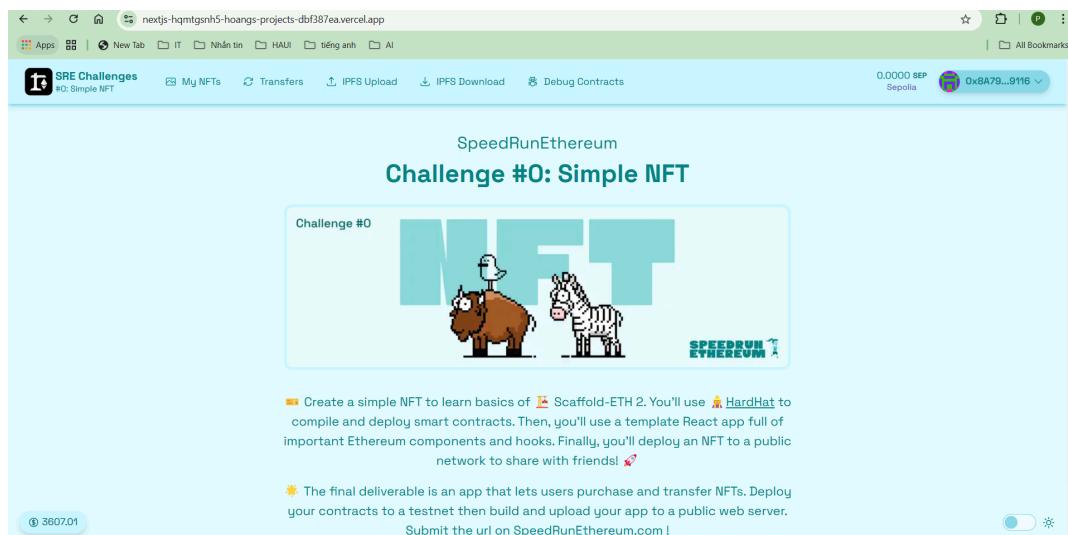
- Triển khai ứng dụng Next.js: Sử dụng lệnh “yarn vercel” để triển khai ứng dụng lên Vercel. Làm theo các bước được gợi ý sau khi gõ lệnh “yarn vercel” để triển khai lên Vercel. Ta có thể cần đăng nhập (bằng email, GitHub, v.v.), các tùy chọn mặc định sẽ hoạt động. Hệ thống sẽ cung cấp cho bạn một URL công khai. Để triển khai lại URL production sử dụng “yarn vercel --prod”.

```
D:\ADMIN\Downloads\challenge-0-simple-nft>yarn vercel --prod
(node:4712) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Vercel CLI 32.7.2
! Your Project was either deleted, transferred to a new Team, or you don't have access to it anymore.
? Set up and deploy "D:\ADMIN\Downloads\challenge-0-simple-nft\packages\next
js"? [Y/n] y
? Which scope do you want to deploy to? hoang's projects
? Link to existing project? [y/N] n
? What's your project's name? nextjs
? In which directory is your code located? ./
Local settings detected in vercel.json:
- Install Command: yarn install
Auto-detected Project Settings (Next.js):
- Build Command: next build
- Development Command: next dev --port $PORT
- Output Directory: Next.js default
? Want to modify these settings? [y/N] n
🔗 Linked to hoangs-projects-dbf387ea/nextjs (created .vercel)
🌐 Inspect: https://vercel.com/hoangs-projects-dbf387ea/nextjs/67sTHB677iLMxejhqYXRoC4khzc4 [2s]
✅ Production: https://nextjs-hqmtgsnh5-hoangs-projects-dbf387ea.vercel.app [2s]
```

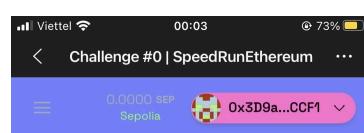
Hình 3.19. Chạy lệnh triển khai trên Vercel.



Hình 3.20. Dự án đã được triển khai trên Vercel.



Hình 3.21. Giao diện ứng dụng web của dự án trên máy tính.



SpeedRunEthereum Challenge #0: Simple NFT



Create a simple NFT to learn basics of Scaffold-ETH 2. You'll use HardHat to compile and deploy smart contracts. Then, you'll use a template React app full of important Ethereum components and hooks. Finally, you'll deploy an NFT to a public network to share with friends!

④ 3590.98 units and hooks. Finish! ↗

Hình 3.22. Giao diện ứng dụng web của dự án trên điện thoại.

TỔNG KẾT

Cả ba đề tài Nghiên cứu phát triển ứng dụng NFT với Blockchain, Phát hiện ảnh đã qua chỉnh sửa ứng dụng CNN, ELA và Keras, và Tìm hiểu, xây dựng ứng dụng microservices quản lý người dùng đều mang đến những đóng góp quan trọng trong việc áp dụng công nghệ hiện đại vào các lĩnh vực thực tiễn.

Nghiên cứu phát triển ứng dụng NFT với Blockchain đã chứng minh tiềm năng to lớn của Blockchain trong việc tạo ra những hệ thống minh bạch, bảo mật, và phi tập trung. Ứng dụng NFT không chỉ mở ra cơ hội số hóa tài sản mà còn thúc đẩy những mô hình kinh doanh sáng tạo, phù hợp với xu hướng kinh tế số toàn cầu.

Phát hiện ảnh đã qua chỉnh sửa ứng dụng CNN, ELA và Keras đã khai thác sức mạnh của trí tuệ nhân tạo để giải quyết các thách thức trong lĩnh vực xử lý hình ảnh. Đề tài không chỉ góp phần vào việc phát hiện gian lận kỹ thuật số mà còn đặt nền móng cho các ứng dụng liên quan đến bảo mật, pháp lý và truyền thông số.

Tìm hiểu, xây dựng ứng dụng microservices quản lý người dùng nhấn mạnh vào tầm quan trọng của kiến trúc phần mềm hiện đại trong việc phát triển hệ thống lớn. Với microservices, các ứng dụng quản lý người dùng có thể tối ưu hóa khả năng mở rộng, bảo trì, và đáp ứng nhu cầu cao trong môi trường công nghệ không ngừng thay đổi.

Ba đề tài tuy thuộc các lĩnh vực khác nhau, nhưng đều phản ánh xu hướng công nghệ tiên tiến và đặt nền tảng cho các nghiên cứu, ứng dụng trong tương lai. Qua các dự án này, chúng ta không chỉ tiếp thu được kiến thức kỹ thuật mà còn hình thành tư duy đổi mới sáng tạo, chuẩn bị cho những bước tiến mạnh mẽ trong lĩnh vực công nghệ thông tin.

TÀI LIỆU THAM KHẢO

- [1]. K. Nakamoto (2008), Bitcoin: A Peer-to-Peer Electronic Cash System, <https://bitcoin.org/bitcoin.pdf>.
- [2]. A. Antonopoulos (2017), Mastering Bitcoin: Unlocking Digital Cryptocurrencies, O'Reilly Media.
- [3]. L. J. Conlan (2021), NFTs & Cryptoart: The Beginner's Guide to Non-Fungible Tokens & the Future of Art, Blockchain, and Digital Assets, Independently Published.
- [4]. Nguyễn Thanh Sơn (2020), Công nghệ Blockchain và ứng dụng trong phát triển phần mềm, Nhà xuất bản Khoa học và Kỹ thuật.
- [5]. Phạm Ngọc Hùng (2022), Phát triển ứng dụng phi tập trung với Ethereum và Solidity, Nhà xuất bản Đại học Quốc gia Hà Nội.
- [6]. A. M. Antonopoulos, G. Wood (2018), Mastering Ethereum: Building Smart Contracts and DApps, O'Reilly Media.
- [7]. Eric Evans (2003), Domain-Driven Design: Tackling Complexity in the Heart of Software, Addison-Wesley Professional.
- [8]. Phạm Huy Hoàng (2020), "Domain-Driven Design: Hiểu và ứng dụng cơ bản", Viblo.
- [9]. Hội đồng CNTT Việt Nam (2023), "Áp dụng Domain-Driven Design vào kiến trúc Microservices", ICTNews.
- [10]. Vaughn Vernon (2013), Implementing Domain-Driven Design, Addison-Wesley Professional.
- [11]. Nguyễn Phương Nga(Chủ biên), Trần Hùng Cường, giáo trình trí tuệ nhân tạo, Trường Đại Học Công nghiệp Hà Nội: Nhà xuất bản thống kê.
- [12]. CASIA 2.0 Image Tampering Detection Dataset. Truy cập ngày 9/9/2024, link: <https://www.kaggle.com/datasets/divg07/casia-20-image-tampering-detection-dataset?resource=download-directory>.
- [13]. Morphed image detection using ela and cnn techniques. Truy cập ngày 9/9/2024, link:https://www.researchgate.net/publication/369235655_MO

RPHED_IMAGE_DETECTION_USING_ELA_AND_CNN_TECHNIQUE

[14]. Digital-Fake-Image-Detection-Based-on-ANN-and-ML. Truy cập ngày 4/10/2024, link: <https://github.com/Shivangraj/Digital-Fake-Image-Detection-Based-on-ANN-and-ML>

[15]. Error level analysis. Truy cập ngày 4/10/2024, link: https://en.wikipedia.org/wiki/Error_level_analysis

[16]. Shedding Light on ELA: A Comprehensive Guide to Error Level Analysis. Truy cập ngày 3/10/2024, link: https://www.fakeimagedetector.com/blog/shedding-light-ela-comprehensive-guide-error-level-analysis/#google_vignette

[17]. Fake images: The effects of source, intermediary, and digital media literacy on contextual assessment of image credibility online. Truy cập vào ngày 10/10/2024, link: <https://journals.sagepub.com/doi/10.1177/1461444818799526>