ĐẠI HỌC HUẾ

TRƯỜNG ĐẠI HỌC KHOA HỌC

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----🙞 🕮 🙜-----**

KHOÁ LUẬN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

XÂY DỰNG WEBSITE CUNG CẤP KHÓA HỌC TRỰC TUYẾN VÀ ỨNG DỤNG BLOCKCHAIN ĐỂ LƯU TRỮ CHỨNG CHỈ TRÊN MẠNG LƯỚI SEPOLIA

**Sinh viên thực hiện: MAI TIẾN DŨNG**

**Khóa: K45 - CÔNG NGHỆ THÔNG TIN**

Huế, tháng 06 – năm 2025

ĐẠI HỌC HUẾ

TRƯỜNG ĐẠI HỌC KHOA HỌC

**KHOA CÔNG NGHỆ THÔNG TIN**

**-----🙞 🕮 🙜-----**

KHOÁ LUẬN

TỐT NGHIỆP ĐẠI HỌC

Đề tài:

XÂY DỰNG HỆ THỐNG TUYỂN DỤNG VÀ TÌM KIẾM VIỆC LÀM SỬ DỤNG ASP.NET VÀ ENTITYFRAMEWORK

**Sinh viên thực hiện: PHAN ĐÌNH MINH ĐỨC**

**Khóa: K45 - CÔNG NGHỆ THÔNG TIN**

**Giáo viên hướng dẫn: HUỲNH BẢO QUỐC DŨNG**

Huế, tháng 06 – năm 2025

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn quý Thầy Cô giáo trường Đại học Khoa học, đại học Huế nói chung, và đặc biệt quý Thầy Cô Khoa Công nghệ thông tin nói riêng đã dạy dỗ cho em kiến thức về các môn đại cương, cũng như các môn chuyên ngành, đồng thời tạo điều kiện cho em trong suốt quá trình học tập và hoàn thành bản khóa luận tốt nghiệp này.

Đặc biệt, em xin tỏ lòng biết ơn sâu sắc đến thầy Nguyễn Văn Trung đã giành nhiều thời gian tận tình hướng dẫn, định hướng, chỉ bảo em trong suốt quá trình thực hiện khóa luận.

Em cũng xin gửi lời cảm ơn chân thành đến bạn bè vì đã luôn động viên, quan tâm giúp đỡ em trong quá trình học tập và hoàn thành khóa luận tốt nghiệp.

Huế, tháng 06 năm 2024

Sinh viên thực hiện

Mai Tiến Dũng

**MỤC LỤC**

CHƯƠNG 1: TỔNG QUAN VỀ SPRING

1.1. Giới thiệu chung về Java và lập trình hướng đối tượng

1.2. Tổng quan về nền tảng Spring

    1.2.1. Lịch sử hình thành và phát triển

    1.2.2. Mục tiêu và triết lý thiết kế của Spring

    1.2.3. Kiến trúc tổng quan của Spring Framework

    1.2.4. Các module chính của Spring (Core, AOP, MVC, JDBC, v.v.)

1.3. Spring Boot là gì?

    1.3.1. Lý do ra đời Spring Boot

    1.3.2. Các đặc điểm nổi bật của Spring Boot

    1.3.3. Kiến trúc và cách Spring Boot hoạt động

1.4. So sánh giữa Spring Framework và Spring Boot

1.5. Lợi ích khi sử dụng Spring Boot trong phát triển ứng dụng web

1.6. Ứng dụng thực tiễn của Spring Boot trong các dự án phần mềm

1.7. Kết luận chương

CHƯƠNG 2: TỔNG QUAN VỀ NEXTJS

2.1 Tổng quan về Nextjs

2.2 Tính năng nổi bật của Nextjs

2.3 Kiến trúc hoạt động của Nextjs

2.4 Ưu điểm và hạn chế của Nextjs

CHƯƠNG 3: TỔNG QUAN VỀ BLOCKCHAIN VÀ ERC721

CHƯƠNG 4: XÂY DỰNG WEBSITE CUNG CẤP KHÓA HỌC TRỰC TUYẾN

CHƯƠNG 5: ỨNG DỤNG CÔNG NGHỆ BLOCKCHAIN ĐỂ LƯU TRỮ CHỨNG CHỈ

**DANH MỤC CÁC TỪ VIẾT TẮT**

**DANH MỤC CÁC BẢNG BIỂU - HÌNH VẼ**

**LỜI MỞ ĐẦU**

**CHƯƠNG 1: TỔNG QUAN VỀ SPRING VÀ SPRING BOOT**

* 1. **Tổng quan về Spring**
     1. **Giới thiệu**

Spring là một công cụ phát triển đa nền tảng, cung cấp các mô hình lập trình và cấu hình cho các ứng dụng doanh nghiệp hiện đại dựa trên ngôn ngữ lập trình Java.

**1.1.2 Lịch sử của Spring**

Vào năm 2000, Rod Johnson một chuyên viên cố vấn trong lĩnh vực tài chính tại London, đã phát triển thêm một phần mã nguồn minh họa nhằm thể hiện quan điểm của mình về cách mà các thành phần trong Java EE nên được tổ chức và liên kết một cách đơn giản hơn trong cuốn sách *Expert One-on-one J2EE Design and Development*. Spring được tạo ra như một giải pháp thay thế nhẹ nhàng và linh hoạt hơn, tập trung vào việc sử dụng các đối tượng Java đơn giản. Rod Johnson được công nhận là người xây dựng framework dựa trên những yêu cầu mà lập trình viên cho là những phương pháp tốt nhất, tạo cơ hội để nó có thể được sử dụng trong việc phát triển các ứng dụng ở các nền tảng khác nhau.

Vào tháng 2 năm 2003, một nhóm nhỏ các nhà phát triển đã tạo một dự án mã nguồn mở tại Sourceforge và tiếp tục mở rộng framework và phát hành phiên bản đầu tiên (1.0) vào tháng 03 năm 2004.

Phiên bản Spring 2.0 được phát hành vào tháng 10 năm 2006, cho phép cấu hình dễ dàng hơn với XML DSL, hỗ trợ tốt hơn cho annotations và giao diện người dùng tốt hơn với Spring MVC cải tiến. Tiếp theo phiên bản Spring 2.5 được phát hành vào tháng 11 năm 2007.

Từ năm 2009, Spring tung ra ba phiên bản Spring 3.0 vào tháng 12 năm 2009, Spring 3.1 vào tháng 12 năm 2011 và Spring 3.2.5 vào tháng 11 năm 2013. Ở các phiên bản này, Spring hỗ trợ Java 5 và bắt đầu chuyển sang sử dụng annotations thay vì XML đồng thời cũng hỗ trợ RESTful API với Spring MVC.

Spring 4.0 được phát hành vào tháng tháng 12 năm 2013 và những cải tiến đáng chú ý trong Spring 4.0 bao gồm hỗ trợ cho Java SE 8, Groovy 2, Java EE7 và WebSocket.

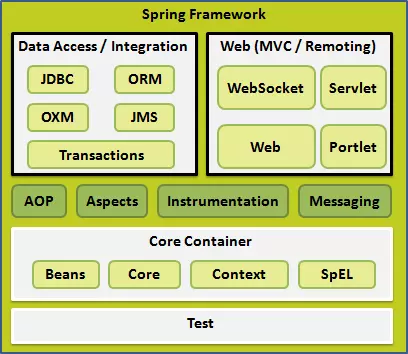
Spring 5 được công bố là được xây dựng dựa trên Reactor Core tương thích vơi Reactive Streams.

Spring 6.0 được phát hành vào ngày 16 tháng 11 năm 2022 và đi kèm với Java 17+ và chuyển sang Jakarta EE9+, tập trung vào các API Jakarta EE10 mới phát hành gần đây như Servlet 6.0 và JPA 3.1.

**1.1.3 Kiến trúc tổng quan của Spring Framework**

Kiến trúc của Spring Framework được xây dựng theo mô hình phân tầng. Mỗi tầng trong Spring đảm nhận một vai trò nhất định và có thể hoạt động độc lập hoặc kết hợp với các tầng khác tùy theo nhu cầu của ứng dụng.

Cấu trúc tổng thể của Spring có thể chia thành 7 nhóm module chính:



*Hình 1 Kiến trúc của Spring*

* **Testing Module:** Spring cung cấp các tiện ích để hỗ trợ kiểm thử, bao gồm:

- Khả năng khởi tạo context kiểm thử với @SpringBootTest, @ContextConfiguration.

- Tích hợp tốt với JUnit, TestNG.

- Cho phép mock các bean hoặc phụ thuộc dễ dàng thông qua @MockBean, @WebMvcTest, ...

* **Core Container:** Core Container là nền tảng cốt lõi cho toàn bộ hệ thống Spring, nó chứa các module quan trọng như:

- Beans (cung cấp các tính năng tạo và quản lý các Beans);

- Core (cung cấp các lớp tiện ích, cơ chế Inversion of Control và chức năng Dependency Injection);

- Context(cung cấp ApplicationContext, hỗ trợ i18n và xử lý sự kiện, …);

- SpEL ( ngôn ngữ biểu thức hỗ trợ truy vấn và thao tác với các đối tượng tại thời điểm runtime).

- Transactions module: module này hỗ trợ quản lí transaction.

* **AOP Module (Aspect-Oriented Programming):**

**-** Spring hỗ trợ lập trình hướng khía cạnh thông qua module AOP, giúp lập trình viên dễ dàng tách biệt các cross-cutting concerns như logging, transaction management, bảo mật khỏi logic nghiệp vụ chính.

- Spring sử dụng các proxy để can thiệp vào hành vi của các đối tượng mà không làm thay đổi mã nguồn ban đầu. Việc này không những tăng tính module hóa mà còn cải thiện khả năng bảo trì mã nguồn.

* **Data Access / Integration:** hỗ trợ làm việc với cơ sở dữ liệu và tích hợp vào hệ thống nhờ các module sau:

- JDBC module: module này cung cấp một lớp JDBC-abstraction và dễ quản lý khi làm việc với database.

- ORM module: module này cung cấp một lớp abstraction giúp dễ dàng thực hiện việc mapping với Hibernate, JPA, JDO.

- JMS (Java message service): module này cung cấp các chức năng cho việc gửi và nhận message.

- OXM: module này hỗ trợ XML binding.

* **Messaging Module:** Module này giúp các thành phần trong hệ thống giao tiếp qua các kênh message. Nó cũng tích hợp với các hệ thống nhắn tin như RabbitMQ, Apache Kafka.
* **Instrumentation Module:** Dùng để tích hợp với các công cụ giám sát, profiling hoặc các công nghệ thao tác bytecode như ASM, Javassist. Module này hỗ trợ các nhu cầu cao cấp về hiệu suất hoặc giám sát hoạt động của ứng dụng khi runtime.
* **Web Layer:** chứa đựng các modules: Web, Web-MVC, Web-Socket và Web-Portlet.

- Web module cung cấp các chức năng tích hợp Web-oriented như là chức năng file-upload và khởi tạo IoC container thứ mà sử dụng servlet listener và context ứng dụng web-oriented.

- Web MVC module chứa sự các thành phần Model-View-Controller cho 1 web application.

- Web Portlet module cung cấp sự thực hiện MVC để sử dụng trong môi trường portlet và phản ảnh chức năng của module Web-Servlet.

- Websocket: Cung cấp giải pháp giao tiếp real-time dựa trên giao thức WebSocket.

* 1. **Tổng quan về Spring Boot**
     1. **Lý do ra đời của Spring Boot**

Spring Framework là một nền tảng phát triển mạnh mẽ và linh hoạt, nhưng nó đòi hỏi lập trình viên phải thực hiện rất nhiều công việc cấu hình để xây dựng một ứng dụng hoàn chỉnh. Các công việc này bao gồm việc khai báo bean, thiết lập cấu hình XML, quản lý các dependency, cấu hình web server, xử lý transaction, và kết nối database. Điều này làm cho quá trình phát triển ứng dụng bằng Spring trở nên phức tạp, đặc biệt đối với những lập trình viên mới bắt đầu hoặc các dự án nhỏ cần triển khai nhanh chóng.

Khi Spring ngày càng phát triển và được sử dụng rộng rãi, nhằm giúp lập trình viên giảm bớt gánh nặng về cấu hình và tập trung vào việc phát triển logic nghiệp vụ. Chính vì vậy, Spring Boot ra đời vào năm 2014, như một phần mở rộng của Spring Framework, nhằm đơn giản hóa quá trình khởi tạo, cấu hình và triển khai các ứng dụng Spring một cách nhanh chóng, đơn giản, hiệu quả cho các lập trình viên.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Năm | Spring Framework | Java yêu cầu | Spring Boot | Java yêu cầu |
| 2003 | 1.0 | Java 1.3+ | - | - |
| 2006 | 2.0 | Java 1.4+ | - | - |
| 2009 | 3.0 | Java 5+ | - | - |
| 2013 | 4.0 | Java 6+ | - | - |
| 2014 | - | - | 1.0.0 | Java 6+ |
| 2017 | 5.0 | Java 8+ | 2.0.0 | Java 8+ |
| 2018 | 5.1 | Java 8+ | 2.1.x | Java 8+ |
| 2019 | 5.2 | Java 8+ | 2.2.x | Java 8+ |
| 2020 | - | - | 2.3.x | Java 8+ |
| 2021 | - | - | 2.5.x | Java 8+ |
| 2022 | 5.3 | Java 8+ | 2.7.x | Java 8+ |
| 2022-11 | 6.0 | Java 17+ | 3.0.0 | Java 17+ |
| 2023 | 6.1 | Java 17+ | 3.1.x | Java 17+ |
| 2024 | - | - | 3.2.x | Java 17+ |

*Bảng 1 Các phiên bản của Spring và Spring Boot*

* + 1. **Các đặc điểm nổi bật của Spring Boot**
* **Tự động cấu hình**

Spring Boot tự động cấu hình ứng dụng dựa trên các thư viện có trong classpath, giảm thiểu công sức thiết lập thủ công. Tính năng này cung cấp các thiết lập mặc định hợp lý, giúp lập trình viên nhanh chóng khởi tạo dự án mà không cần quản lý phức tạp các phụ thuộc. Quá trình tự động cấu hình hoạt động dựa trên việc phân tích các thư viện và cấu hình hiện có, đảm bảo tính linh hoạt và hiệu quả.

* **Máy chủ nhúng**

Spring Boot tích hợp sẵn các máy chủ như Tomcat, Jetty hoặc Undertow, cho phép ứng dụng chạy độc lập mà không cần triển khai trên máy chủ bên ngoài. Điều này giúp quá trình phát triển và kiểm thử trở nên thuận tiện.

* **Spring Boot Actuator**

Spring Boot Actuator cung cấp các tính năng giám sát và quản lý ứng dụng trong môi trường sản xuất. Các endpoint của Actuator cho phép theo dõi trạng thái, hiệu suất và thông tin môi trường của ứng dụng, giúp lập trình viên dễ dàng đánh giá và duy trì chất lượng hệ thống.

* **Gói khởi đầu Spring Boot**

Các gói khởi đầu là tập hợp các dependency được cấu hình sẵn, hỗ trợ tích hợp các chức năng như phát triển web, kết nối cơ sở dữ liệu và các thư viện. Việc sử dụng các gói này giúp giảm thiểu công việc cấu hình thủ công, tăng tốc quá trình phát triển.

* **Spring Boot DevTools**

Spring Boot DevTools cung cấp các công cụ hỗ trợ như tự động khởi động lại ứng dụng khi có thay đổi mã nguồn, tải lại tài nguyên tĩnh và hỗ trợ gỡ lỗi từ xa. Các tính năng này cải thiện hiệu suất phát triển, đặc biệt trong giai đoạn phát triển phần mềm.

* **Tích hợp với Spring Data JPA**

Spring Boot tích hợp chặt chẽ với Spring Data JPA, giúp đơn giản hóa việc truy vấn đến cơ sở dữ liệu và giảm đáng kể mã nguồn lặp lại, nâng cao hiệu quả phát triển.

* **Spring Security**

Spring Security hỗ trợ triển khai các cơ chế xác thực và phân quyền một cách dễ dàng, đồng thời đơn giản hóa cấu hình bảo mật, quản lý người dùng và kiểm soát quyền truy cập.

* **Cấu hình bên ngoài**

Spring Boot cho phép cấu hình ứng dụng thông qua tệp properties, yaml hoặc biến môi trường. Phương pháp này giúp tùy chỉnh ứng dụng linh hoạt và phù hợp với các môi trường phát triển khác nhau.

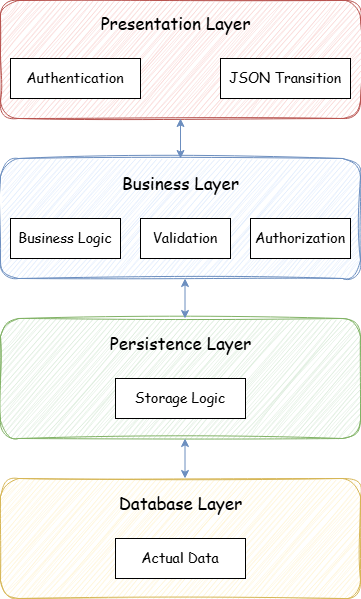
* **Spring Testing Framework**

Spring Boot cung cấp một công cụ hỗ trợ kiểm thử giúp làm các bài kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử hệ thống nhanh chóng.

* **Thống kê hiệu suất**

Tính năng cung cấp số liệu về hiệu suất ứng dụng như độ trễ, tỷ lệ lỗi và mức sử dụng tài nguyên (CPU, RAM, …). Các số liệu này có thể được tích hợp với các công cụ giám sát như Prometheus và Grafana, hỗ trợ quản lý và tối ưu hóa hệ thống.

* + 1. **Kiến trúc Spring Boot**

****

*Hình 2 Kiến trúc của Spring Boot*

* **Presentation Layer:**

Đây là tầng trên cùng của Spring Boot bao gồm có View và một phần front-end của ứng dụng. Ở tầng này Spring Boot sẽ nhận các request từ người dùng rồi tiến hành xử lý xác thực.

Nó sẽ xử lý chuyển đổi từ đối tượng JSON sang đối tượng Java và ngược lại. Khi xác thực được thực hiện thành công nó sẽ được chuyển sang tầng tiếp theo là Business Layer.

* **Business Layer:**

Tầng này sẽ xử lý tất cả các yêu cầu nghiệp vụ (Business Logic) bao gồm các lớp Service và sử dụng các service được cung cấp bởi tầng DAO. Nó chịu trách nhiệm xác nhận/kiểm duyệt dữ liệu và ủy quyền cho user.

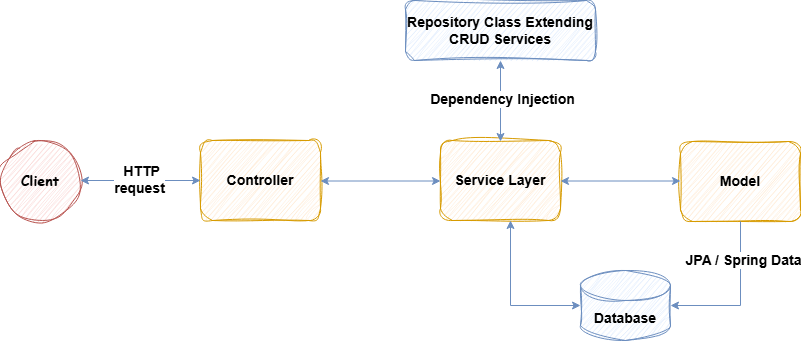
* **Persistence Layer:**

Bao gồm tất cả các logic liên quan đến database. Tầng này chịu trách nhiệm chuyển đổi đối tượng Java sang đối tượng database và ngược lại.

* **Database Layer:**

Bao gồm tất cả các database RDBMS, NoSQL. Nó chịu trách nhiệm thực hiện CRUD dữ liệu vào database.

* + 1. **Sơ đồ luồng của Spring Boot**

****

*Hình 3 Sơ đồ luồng của Spring Boot*

**Bước 1: Client gửi HTTP Request**

Client (trình duyệt, ứng dụng, v.v.) gửi các yêu cầu HTTP (GET, POST, PUT, DELETE,...) đến ứng dụng Spring Boot.

**Bước 2: Controller xử lý Request**

Các request được ánh xạ tới các phương thức trong Controller (được đánh dấu bằng *@RestController* hoặc *@Controller*).

Controller chịu trách nhiệm nhận request, xác thực đầu vào (nếu cần), và điều hướng request đến Service Layer phù hợp.

**Bước 3: Service Layer xử lý Business Logic**

Service (thường được đánh dấu bằng *@Service*) chứa toàn bộ business logic của ứng dụng.

Controller gọi các phương thức trong Service để thực hiện các tác vụ như tính toán, xử lý dữ liệu, hoặc tương tác với cơ sở dữ liệu được ánh xạ thông qua các Model.

Service Layer giúp tách biệt logic nghiệp vụ khỏi tầng Controller, tăng tính tái sử dụng và dễ bảo trì.

**Bước 4: Tương tác với Database qua Model**

Spring Boot sử dụng ORM (Object-Relational Mapping), thường là Spring Data JPA, để ánh xạ dữ liệu từ cơ sở dữ liệu vào các Model (được đánh dấu bằng *@Entity*).

Các Repository (thường là interface kế thừa từ JpaRepository) được sử dụng để thực hiện các thao tác CRUD (Create, Read, Update, Delete) với cơ sở dữ liệu.

Service Layer gọi Repository để truy vấn hoặc cập nhật dữ liệu.

**Bước 5: Controller trả về Response**

Sau khi Service Layer xử lý xong, kết quả được trả về cho Controller.

Controller định dạng kết quả (thường là JSON) và gửi HTTP Response về cho Client, kèm theo mã trạng thái HTTP (200 OK, 404 Not Found, ...).

* 1. **So sánh giữa Spring và Spring Boot**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Spring Framework** | **Spring Boot** |
| **Cấu hình** | Yêu cầu cấu hình chi tiết (XML, Java Config) | Hạn chế cấu hình thủ công, sử dụng mặc định thông minh |
| **Khởi tạo dự án** | Tự thiết lập cấu trúc và thêm thư viện | Hỗ trợ công cụ Spring Initializr, sinh project tự động |
| **Quản lý thư viện** | Thêm và quản lý từng dependency riêng lẻ | Dùng starter (spring-boot-starter-\*) giúp quản lý phụ thuộc nhanh chóng |
| **Máy chủ ứng dụng** | Cần tích hợp và cấu hình máy chủ ngoài (Tomcat, Jetty, v.v.) | Tích hợp sẵn máy chủ nhúng (embedded Tomcat, Jetty, Undertow) |
| **Thời gian phát triển** | Có thể kéo dài do khối lượng cấu hình lớn | Rút ngắn đáng kể nhờ cấu trúc và công cụ đơn giản hóa |
| **Tích hợp Actuator, Monitoring** | Không có sẵn, cần cài đặt và cần cấu hình riêng | Tích hợp sẵn Spring Boot Actuator phục vụ giám sát và theo dõi hệ thống |
| **Tính học tập và tiếp cận** | Khá phức tạp đối với người mới | Dễ tiếp cận hơn, phù hợp cả với người mới |
| **Mục tiêu sử dụng** | Phù hợp với các ứng dụng yêu cầu cấu hình tinh chỉnh, kiểm soát chi tiết | Phù hợp với những dự án cần phát triển nhanh, ứng dụng microservice, REST API, …) |

*Bảng 2 So sánh Spring và Spring boot*

* 1. **Lợi ích khi sử dụng Spring Boot trong phát triển ứng dụng web**

Spring Boot là một phần mở rộng của Spring Framework, được thiết kế để đơn giản hóa quá trình phát triển các ứng dụng Java, đặc biệt là các ứng dụng web. Trong bối cảnh phát triển phần mềm hiện đại, nơi tốc độ phát triển, khả năng mở rộng và dễ bảo trì là những yếu tố then chốt, Spring Boot mang lại nhiều lợi ích nổi bật, cụ thể như sau:

**1.4.1. Cấu hình tối thiểu**

Spring Boot giảm đáng kể lượng cấu hình cần thiết cho một dự án phần mềm. Điều này giúp các lập trình viên tập trung hơn vào logic nghiệp vụ thay vì dành nhiều thời gian cho các thiết lập hệ thống phức tạp.

**1.4.2. Máy chủ nhúng tích hợp**

Spring Boot sử dụng các máy chủ ứng dụng nhúng tích hợp như Tomcat, Jetty hoặc Undertow. Do đó, việc chạy một ứng dụng web chỉ đơn giản là thực thi một tệp .jar, mà không cần triển khai lên máy chủ bên ngoài. Điều này rất thuận tiện cho việc phát triển, thử nghiệm và triển khai.

**1.4.3. Khởi tạo dự án nhanh chóng**

Công cụ Spring Initializr cho phép khởi tạo một dự án Spring Boot chỉ trong vài phút với cấu trúc thư mục chuẩn, các thư viện cần thiết và cấu hình ban đầu được thiết lập sẵn theo nhu cầu của lập trình viên. Điều này làm giảm đáng kể thời gian bắt đầu dự án.

**1.4.4. Hệ sinh thái phong phú**

Spring Boot cung cấp các dependency như spring-boot-starter-web, spring-boot-starter-data-jpa, spring-boot-starter-security, v.v. Các trình khởi động này đơn giản hóa việc thêm các thư viện và cấu hình tương ứng, đồng thời đảm bảo khả năng tương thích giữa các thành phần trong dự án.

**1.4.5. Dễ dàng mở rộng và tích hợp**

Spring Boot hỗ trợ tích hợp mạnh mẽ với các công nghệ phổ biến như cơ sở dữ liệu quan hệ (PostgreSQL, MySQL), NoSQL (MongoDB), dịch vụ RESTful, công cụ giám sát (Actuator), ghi nhật ký (Logback, SLF4J) và nền tảng đám mây (AWS, GCP, Docker, ...). Điều này giúp các ứng dụng web được xây dựng bằng Spring Boot có khả năng mở rộng cao và dễ dàng thích ứng với môi trường phát triển, kiểm thử và triển khai.

**1.4.6. Phù hợp với kiến trúc Microservices**

Spring Boot được thiết kế để phù hợp với kiến trúc Microservices nhờ khả năng triển khai độc lập, khởi chạy nhanh và hỗ trợ tốt cho giao tiếp giữa các dịch vụ (REST API, RabbitMQ, Kafka, ...) thông qua giao tiếp đồng bộ và bất đồng bộ. Đây là một lợi thế lớn trong việc phát triển các hệ thống phân tán hoặc các ứng dụng web quy mô lớn.

**1.4.7. Hỗ trợ mạnh mẽ cho môi trường kiểm thử**

Spring Boot tích hợp các công cụ kiểm thử như JUnit, Mockito và Spring Test, hỗ trợ kiểm thử đơn vị và kiểm thử tích hợp dễ dàng. Điều này giúp cải thiện đáng kể chất lượng mã nguồn và giảm thiểu lỗi phát sinh trong quá trình phát triển.

**CHƯƠNG 2: TỔNG QUAN VỀ NEXT.JS**

**2.1 Giới thiệu chung**

Next.js là một framework mã nguồn mở được xây dựng trên nền tảng React.js, do công ty Vercel phát triển và duy trì. Kể từ khi ra mắt lần đầu vào năm 2016, Next.js đã nhanh chóng trở thành một công cụ phổ biến cho việc phát triển các trang web tĩnh có tốc độ siêu nhanh và thân thiện với người dùng, cũng như xây dựng các ứng dụng web React, nhờ khả năng kết hợp giữa phát triển phía client và server, cùng với sự hỗ trợ mạnh mẽ cho các tính năng tối ưu hóa hiệu suất, SEO và trải nghiệm người dùng.

Với triết lý “Zero Config” và khả năng mở rộng linh hoạt, Next.js đã trở thành lựa chọn hàng đầu cho nhiều công ty và tổ chức trong việc xây dựng các website quy mô lớn và ứng dụng web phức tạp.

Sự kết hợp của các tính năng như Server-side Rendering (SSR) với Static Site Generation (SSG) đã giúp NextJS trở thành sự lựa chọn hấp dẫn cho nhiều dự án website hiện đại.

**2.2 Các tính năng nổi bật của Next.js**

**2.2.1 Hệ thống định tuyến trong Next.js**

Next.js cung cấp hệ thống định tuyến (routing) mạnh mẽ và linh hoạt, được xây dựng trên nguyên tắc “file-based routing”, nghĩa là các tuyến đường (route) được tự động ánh xạ từ cấu trúc thư mục của dự án. Cơ chế định tuyến này giúp đơn giản hóa quá trình phát triển ứng dụng web bằng cách giảm thiểu cấu hình thủ công.

**2.2.1.1 Tự động định tuyến**

Trong Next.js, bất kỳ tệp JavaScript hoặc TypeScript nào được tạo trong thư mục pages/ sẽ tự động trở thành một route tương ứng trong ứng dụng. Ví dụ, việc tạo tệp about.js trong thư mục pages/ sẽ dẫn đến việc hình thành route /about mà không cần cấu hình thêm. Điều này góp phần rút ngắn thời gian phát triển và giảm thiểu lỗi cấu hình thủ công.

**2.1.1.2 Định tuyến lồng nhau**

Next.js cho phép tổ chức định tuyến theo dạng phân cấp thông qua việc tạo các thư mục con bên trong thư mục pages/. Ví dụ, nếu có cấu trúc thư mục pages/blog/post.js, thì route tương ứng sẽ là /blog/post.

**2.1.1.3 Định tuyến động**

Để xử lý các đường dẫn không cố định, Next.js hỗ trợ định tuyến động thông qua cú pháp dấu ngoặc vuông []. Ví dụ, tệp có tên pages/blog/[slug].js sẽ ánh xạ với các route như /blog/bai-viet-dau-tien hoặc /blog/huong-dan-nextjs, trong đó slug là một tham số động đại diện cho nội dung bài viết. Điều này đặc biệt hữu ích trong các ứng dụng có nội dung động như blog, sản phẩm hoặc người dùng.

**2.1.1.4 Thành phần liên kết**

Next.js cung cấp thành phần Link từ thư viện next/link để hỗ trợ việc điều hướng giữa các trang trong ứng dụng. Việc sử dụng Link thay vì thẻ <a> truyền thống giúp tối ưu hóa hiệu suất bằng cách tránh việc tải lại toàn bộ trang, đồng thời hỗ trợ chuyển trang nhanh.

**2.1.1.5 Tham số truy vấn**

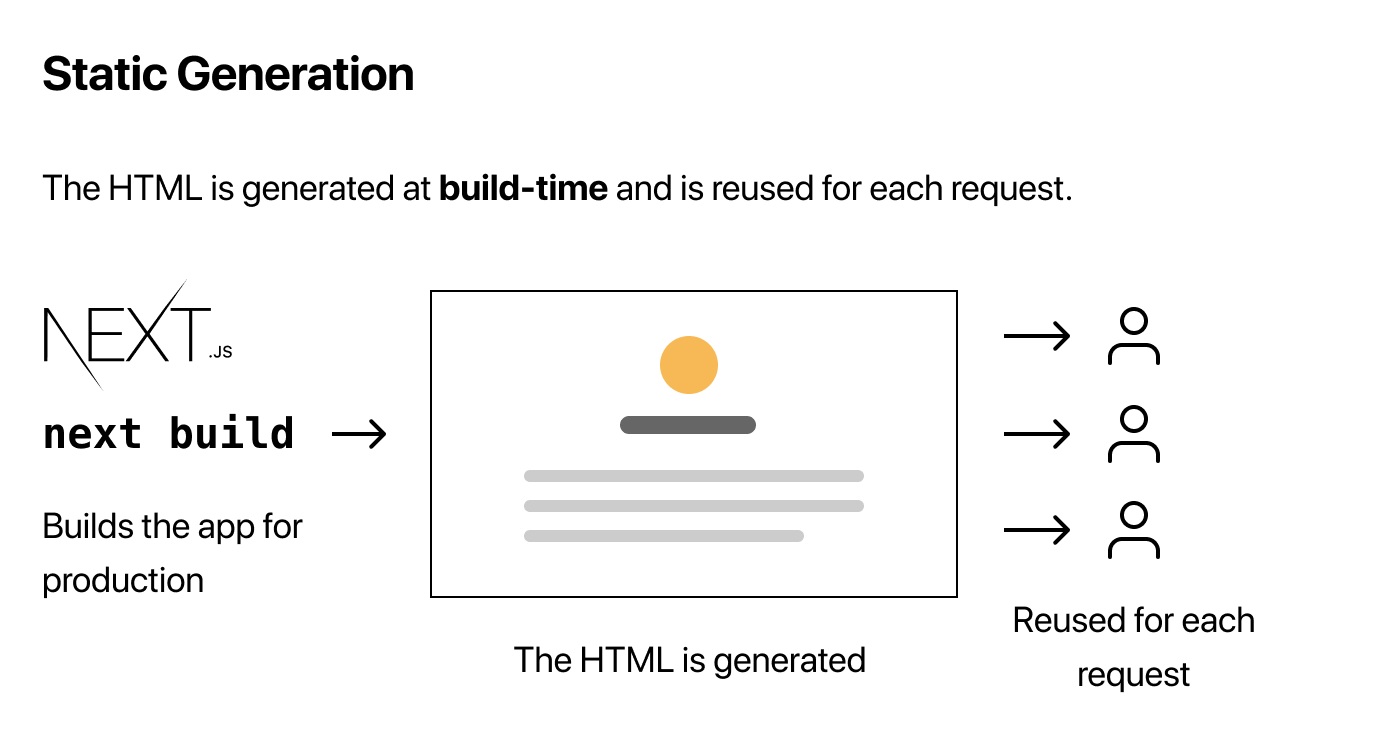
Next.js cũng hỗ trợ việc truyền dữ liệu giữa các trang thông qua tham số truy vấn (query parameters). Các tham số này có thể được đính kèm vào URL bằng dấu chấm hỏi ?. Ví dụ, một route như /product?id=123 có thể được ánh xạ tới pages/product.js, và tham số id có thể được truy xuất trong trang thông qua useRouter từ next/router. Cơ chế này giúp truyền dữ liệu linh hoạt mà không cần tạo thêm các route động phức tạp.

**2.2.2 Hệ thống kết xuất dữ liệu trong Next.js**

Trong quá trình xây dựng các ứng dụng web hiện đại, cơ chế rendering (kết xuất dữ liệu lên giao diện) đóng vai trò quan trọng trong việc tối ưu hóa hiệu suất, cải thiện trải nghiệm người dùng và hỗ trợ các công cụ tìm kiếm (SEO). Next.js – một framework phát triển trên nền tảng React – cung cấp nhiều phương pháp rendering linh hoạt, giúp nhà phát triển lựa chọn chiến lược phù hợp tùy theo từng loại dữ liệu và mục đích sử dụng. Các phương pháp rendering trong Next.js bao gồm:

**2.2.2.1 Static Site Generation**

Static Site Generation là một phương pháp mà NextJS cung cấp sẵn, cho phép lập trình viên tạo các trang tĩnh và lưu chúng xuống dưới dạng file html tĩnh. Điều này giúp cải thiện hiệu suất tải trang và cung cấp trải nghiệm người dùng tốt hơn vì nội dung được lấy từ file html và hiển thị ngay lập tức mà không cần đợi việc tải về từ phía server.



*Hình 4 Cơ chế hoạt động của Static Site Generation*

**2.2.2.2 Server-Side Rendering**

Server-Side Rendering cho phép Next.js tạo HTML tại thời điểm người dùng gửi request. Khi một trang sử dụng SSR, Next.js sẽ thực thi hàm getServerSideProps() trên server, lấy dữ liệu mới nhất và kết hợp với React để kết xuất ra HTML.

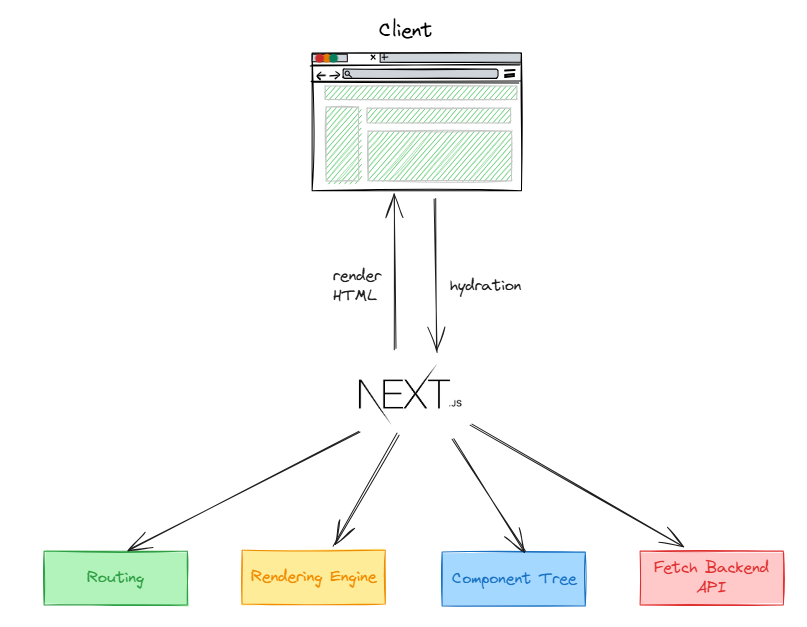
**2.2.2.3 Client-Side Rendering**

Client-Side Rendering là phương pháp rendering truyền thống của React: ban đầu, một file HTML rỗng được gửi tới client, sau đó trình duyệt sẽ tải JavaScript, khởi tạo React và fetch dữ liệu từ API để hiển thị nội dung.

**2.2.2.4 Incremental Static Regeneration**

Incremental Static Regeneration là tính năng kết hợp giữa Static Site Generation và Server-Side Rendering. Các trang vẫn được sinh ra tại thời điểm build như Static Site Generation, nhưng có khả năng được tái tạo lại định kỳ phía server trong nền, mà không cần build toàn bộ website.

**2.3 Kiến trúc hoạt động của Next.js**



Hình 5 Cơ chế hoạt động của Next.js

**2.3.1 Hệ thống định tuyến (Routing)**

Next.js sử dụng cơ chế file-based routing, tức là cấu trúc định tuyến của ứng dụng được xác định trực tiếp từ hệ thống tệp tin trong thư mục /pages/. Mỗi tệp TypeScript (hoặc JavaScript) trong thư mục này sẽ tương ứng với một đường dẫn (route) cụ thể trên trình duyệt. Điều này giúp loại bỏ nhu cầu cấu hình router thủ công như trong React truyền thống.

**2.3.2 Cơ chế kết xuất dữ liệu (Rendering Engine)**

Next.js hỗ trợ nhiều cơ chế render khác nhau, cho phép lập trình viên lựa chọn chiến lược phù hợp tùy theo từng loại trang và yêu cầu hiệu suất:

|  |  |  |  |
| --- | --- | --- | --- |
| **Cơ chế** | **Thời điểm render** | **Ưu điểm** | **Nhược điểm** |
| SSG (Static Site Generation) | Render sẵn khi build | Tốc độ cao, thân thiện SEO | Không phù hợp với dữ liệu thay đổi thường xuyên |
| SSR (Server-Side Rendering) | Render mỗi khi có request | Luôn cập nhật dữ liệu mới, tốt cho SEO | Phụ thuộc máy chủ, thời gian phản hồi chậm hơn |
| CSR (Client-Side Rendering) | Render trên trình duyệt | Tương tác linh hoạt, tải nhanh sau lần đầu | Không thân thiện SEO, tải lần đầu chậm |
| ISR (Incremental Static Regeneration) | Render khi build và mỗi khi có request | Cân bằng giữa SSG và SSR, hiệu suất cao | Cần thiết lập revalidate, không phản hồi tức thì dữ liệu mới |

Bảng 3 So sánh các cơ chế kết xuất dữ liệu của Next.js

**2.3.3 Cây component (Component Tree)**

Kiến trúc giao diện trong Next.js được xây dựng dựa trên mô hình React component-based, trong đó mỗi trang là một component cấp cao (Page Component), và được chia nhỏ thành nhiều component con để tái sử dụng, dễ kiểm thử và bảo trì.

**2.3.4 Hydration và tương tác client**

Sau khi server gửi về HTML đã được render (thông qua SSR hoặc SSG), trình duyệt sẽ thực hiện một bước quan trọng gọi là hydrate. Quá trình này kết nối HTML tĩnh với mã React động, giúp giao diện có thể phản hồi các tương tác của người dùng như click, nhập liệu, hoặc gửi form.

**2.3.5 Tích hợp API từ backend**

Việc tách frontend và backend theo kiến trúc micro-frontend hoặc (Backend for Frontend) mang lại sự linh hoạt cao: frontend có thể được triển khai độc lập, mở rộng dễ dàng, và tích hợp với nhiều loại backend khác nhau thông qua chuẩn REST hoặc GraphQL.

**2.4 Ưu điểm của Next.js trong phát triển website**

Next.js là một framework React mã nguồn mở do Vercel phát triển, được thiết kế nhằm đơn giản hóa việc xây dựng các ứng dụng web hiện đại có khả năng mở rộng cao. Trong bối cảnh nhu cầu phát triển các ứng dụng có hiệu suất tốt, tối ưu SEO và thân thiện với trải nghiệm người dùng ngày càng tăng, Next.js nổi lên như một giải pháp toàn diện cho frontend. Dưới đây là các ưu điểm nổi bật của Next.js trong việc phát triển website hiện đại:

**2.4.1 Hỗ trợ đa dạng cơ chế render: SSG, SSR, ISR và CSR**

* SSG (Static Site Generation): tạo các trang HTML sẵn tại thời điểm build, phù hợp với nội dung tĩnh, tải cực nhanh và thân thiện SEO.
* SSR (Server Side Rendering): tạo HTML động trên server mỗi khi người dùng truy cập, đảm bảo luôn hiển thị dữ liệu mới nhất.
* ISR (Incremental Static Regeneration): kết hợp ưu điểm của SSG và SSR, cho phép tái tạo trang tĩnh theo thời gian hoặc theo lượt truy cập.
* CSR (Client Side Rendering): linh hoạt trong tương tác người dùng, phù hợp với các trang cá nhân hóa như dashboard hoặc trang quản trị.

Việc tùy chọn chiến lược phù hợp theo từng trang giúp tối ưu cả hiệu năng và trải nghiệm người dùng.

**2.4.2 Tối ưu SEO vượt trội**

Không giống như các Single Page Application thông thường chỉ render phía client, Next.js có thể render HTML ngay trên server thông qua SSG hoặc SSR. Điều này cho phép các công cụ tìm kiếm như Google dễ dàng lập chỉ mục (index) nội dung website, cải thiện xếp hạng SEO.

**2.4.3 Cấu trúc rõ ràng và dễ bảo trì**

Next.js sử dụng file-based routing, giúp đơn giản hóa cấu hình route so với React truyền thống. Tệp trong thư mục /pages tự động trở thành một route tương ứng, đồng thời hỗ trợ route động và nested routes.

**2.4.4 Hiệu năng cao và tối ưu tự động**

* Automatic code splitting: chia nhỏ mã nguồn, chỉ tải những gì cần thiết cho từng trang.
* Lazy loading: tải thành phần khi cần thiết.
* Image Optimization (next/image): tự động resize, nén ảnh, hỗ trợ CDN và lazy-load ảnh.
* Prefetching route: tự động tải trước các route khi người dùng di chuột vào link.

**2.4.5 Tích hợp dễ dàng với backend**

Next.js có thể dễ dàng tương tác với các hệ thống backend độc lập (Spring Boot, Node.js, Laravel, v.v) thông qua RESTful API hoặc GraphQL. Người phát triển có thể sử dụng fetch, axios, hoặc React Query để lấy dữ liệu phía client hoặc server.

Đặc biệt, Next.js còn hỗ trợ các hàm như getServerSideProps, getStaticProps, getInitialProps để truy vấn dữ liệu trong quá trình render server hoặc build.

**2.4.6 Hệ sinh thái mạnh mẽ và cộng đồng lớn**

Là một framework chính thức do Vercel phát triển, Next.js có tài liệu đầy đủ, lộ trình phát triển rõ ràng, và được sử dụng bởi nhiều công ty lớn (TikTok, Hulu, Netflix…). Hệ sinh thái phong phú cho phép tích hợp dễ dàng với TailwindCSS, TypeScript, CMS (Headless), hệ thống xác thực (Auth.js) và các dịch vụ đám mây.

**2.4.7 Hỗ trợ TypeScript**

Next.js hỗ trợ TypeScript, giúp kiểm tra kiểu dữ liệu tại thời điểm biên dịch, giảm thiểu lỗi runtime. Ngoài ra, framework còn tích hợp hot reload, fast refresh và hỗ trợ viết các ứng dụng Progressive Web App, giúp đẩy nhanh quá trình phát triển.

**CHƯƠNG 3: TỔNG QUAN VỀ BLOCKCHAIN, ERC-721 VÀ MẠNG LƯỚI SEPOLIA**

**3.1 Tổng quan về Blockchain**

**3.1.1 Giới thiệu về Blockchain**

Blockchain là một mạng lưới phi tập trung được duy trì bởi các máy tính trên toàn thế giới đóng vai trò là một nút trong hệ thống. Do đó, công nghệ blockchain được đảm bảo là có tính an toàn và đáng tin cậy, cho phép ghi lại những hoạt động giao dịch, lưu trữ và trao đổi dữ liệu dựa trên công nghệ sổ cái phân tán.

Về bản chất, Blockchain là một cơ sở dữ liệu phân cấp, nơi thông tin được lưu trữ trong các khối và các khối này được liên kết với nhau bằng mã băm và theo thời gian sẽ tạo thành một chuỗi khối. Trong đó, mỗi khối sẽ chứa thông tin về thời gian khởi tạo, thông tin về dữ liệu giao dịch và mã băm của chính nó và của khối trước đó.

**3.2 Tổng qua về mạng lưới Sepolia**

**3.3 Tổng quan về hợp đồng thông minh và ERC-721**

**3.3.1 Giới thiệu về hợp đồng thông minh**

Hợp đồng thông minh là những hợp đồng điện tử hay những chương trình máy tính được lưu trữ trên các khối trong mạng lưới blockchain. Chúng sẽ tự động hoạt động khi một thỏa thuận đáp ứng đủ các điều kiện đã được xác định trước đó mà không cần bất kỳ tổ chức nào làm vị trí trung gian để quản lý và can thiệp.

Hợp đồng thông minh có thể được xem là một cơ sở hạ tầng mạnh mẽ phục vụ cho tự động hóa bởi vì chúng hoạt động dựa trên cơ chế đồng thuận và ngang hàng. Vì vậy, chúng không cần được kiểm soát bởi một quản trị viên trong hệ thống do đó sẽ tránh được các cuộc tấn công vào các nút đơn lẻ trong hệ thống. Từ đó có thể giảm rủi ro, tăng hiệu quả, giảm chi phí và cung cấp mức độ tin tưởng cao hơn trong các quy trình tự động.

Các mạng lưới blockchain cung cấp một số tính năng cơ bản cho phép các lập trình viên có thể viết các điều kiện thực thi bằng mã theo nhu cầu dựa trên sự đồng thuận phi tập trung mà blockchain cung cấp. Qua đó, mở ra một mối quan hệ phân cấp giữa blockchain hợp đồng thông minh và các ứng dụng phi tập trung. Mối quan hệ phân cấp được biểu diễn như sau: Các ứng dụng phi tập trung được tạo bởi một hay nhiều mã nguồn của các hợp đồng thông minh được kết nối với nhau. Các hành động được thực thi tự động trong hợp đồng thông minh sẽ được thực hiện và lưu trữ bởi các mạng lưới blockchain.

**3.3.2 Giới thiệu về ERC-721**

**3.3.2.1 Giới thiệu**

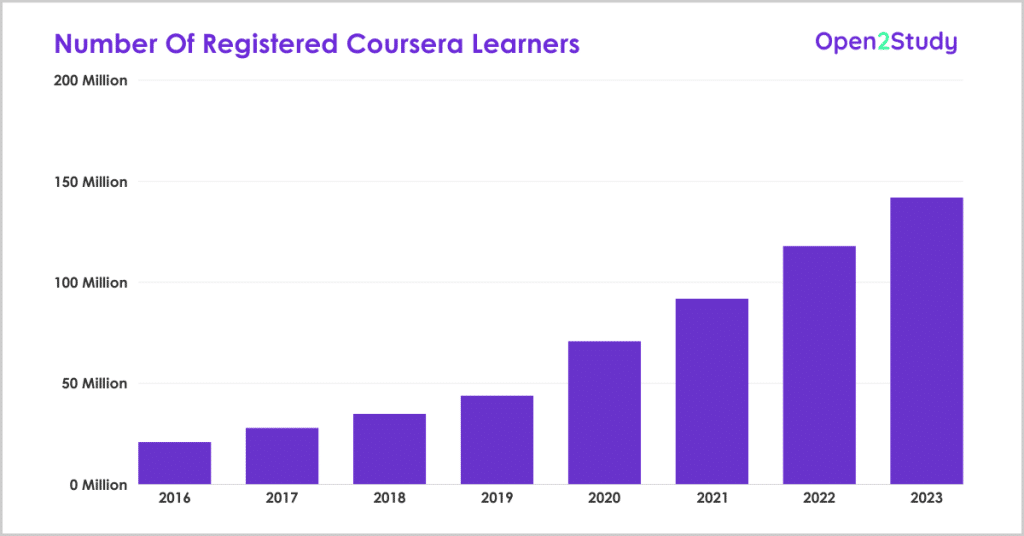
ERC là viết tắt của Ethereum Request for Comment hay có thể hiểu là một tiêu chuẩn kỹ thuật được các lập trình viên đề xuất, thảo luận và hoàn thiện để trở thành một tiêu chuẩn chính thức trong hệ sinh thái Ethereum.

ERC-721 là một tiêu chuẩn hợp đồng với nhiều phần mở rộng được tùy chỉnh, được đề xuất trong hệ sinh thái Ethereum bởi bốn nhà phát triển là William Entriken, Dieter Shirley, Jacob Evans và Nastassia Sachs vào năm 2018. Nó được tạo ra nhằm định nghĩa cách thức tạo và quản lý các tài sản không thể thay thế trên nền tảng blockchain. Đồng thời tiêu chuẩn ERC-721 sử dụng một ID định danh riêng biệt để cho phép các tài sản là duy nhất, từ đó đảm bảo tính độc nhất và không thể hoán đổi lẫn nhau.

**CHƯƠNG 4: WEBSITE CUNG CẤP KHÓA HỌC TRỰC TUYẾN**

**4.1 Phân tích thực trạng**

Hiện nay, trong bối cảnh phát triển nhanh chóng về kiến thức và công nghệ, nhu cầu tự học để phát triển các kỹ năng và kiến thức của bản thân để đáp ứng thị trường việc làm cũng như học tập và giảng dạy ngày càng tăng cao. Trên thị trường hiện nay có nhiều website cung cấp các khóa học trực tuyến như Udemy, Coursera, … cung cấp hàng nghìn khóa học với nhiều chủ đề đa dạng.



Hình : Số lượng học viên của Coursera ( nguồn: Open2Study)

Tuy nhiên các website này vẫn còn tồn tại những hạn chế trong việc xác thực tính tin cậy và minh bạch của chứng chỉ mà người học nhận được sau khi hoàn thành khóa học. Lí do là đa số các chứng chỉ hiện nay được cung cấp dưới dạng ảnh hoặc tệp PDF, rất dễ bị giả mạo, chỉnh sửa, hoặc không thể xác minh một cách trực tiếp và minh bạch. Điều này khiến nhà tuyển dụng gặp khó khăn trong việc đánh giá năng lực thực sự của ứng viên, đồng thời những chứng chỉ này cũng không dễ dàng để tích hợp vào hệ thống tuyển dụng hay quản lý nhân sự của họ.

Vì vậy, việc xây dựng một website cung cấp khóa học mới và cung cấp những chứng chỉ dưới dạng Non-Fungible Token là một giải pháp tiềm năng để khắc phục những hạn chế trên. Với công nghệ blockchain, mỗi chứng chỉ sẽ được mã hóa dưới dạng một token độc nhất, không thể bị sao chép hoặc chỉnh sửa và có thể được xác thực công khai, minh bạch trên hệ thống blockchain. Qua đó:

* Các nhà tuyển dụng có thể dễ dàng xác minh nguồn gốc của chứng chỉ nhờ vào địa chỉ khối do website cung cấp khóa học này cung cấp thông qua các trình khám phá blockchain như Etherscan, Blockscout, OKLink, ….
* Các chứng chỉ NFT có thể dễ dàng tích hợp vào hệ thống quản lý nhân sự, hồ sơ số hoặc CV điện tử nhờ vào việc sử dụng URI để dẫn đến dữ liệu metadata của NFT trên một hệ thống tập tin phân tán ngang hàng chứa thông tin chi tiết của chứng chỉ như tên người học, tên khóa học, ngày hoàn thành, địa chỉ của tổ chức cấp phát và liên kết tới bản chứng chỉ dạng ảnh. Nhờ đó các hệ thống chỉ cần truy xuất URI là có thể lấy toàn bộ thông tin cần thiết mà không cần phụ thuộc vào cơ sở dữ liệu tập trung. Điều này giúp cho quá trình tích hợp vào các nền tảng như hệ thống quản lý nhân sự trở nên nhanh chóng, thống nhất và có khả năng mở rộng cao.

**4.2 Yêu cầu bài toán**

Đây là website có thể giúp người dùng tham gia học tập và cung cấp những khóa học để tham gia giảng dạy. Đồng thời khi hoàn thành khóa học, học viên có thể nhận những chứng chỉ NFT được xem như là một chứng nhận thành tích rằng học viên đó đã hoàn thành khóa học.

**4.3 Đặc tả yêu cầu phần mềm**

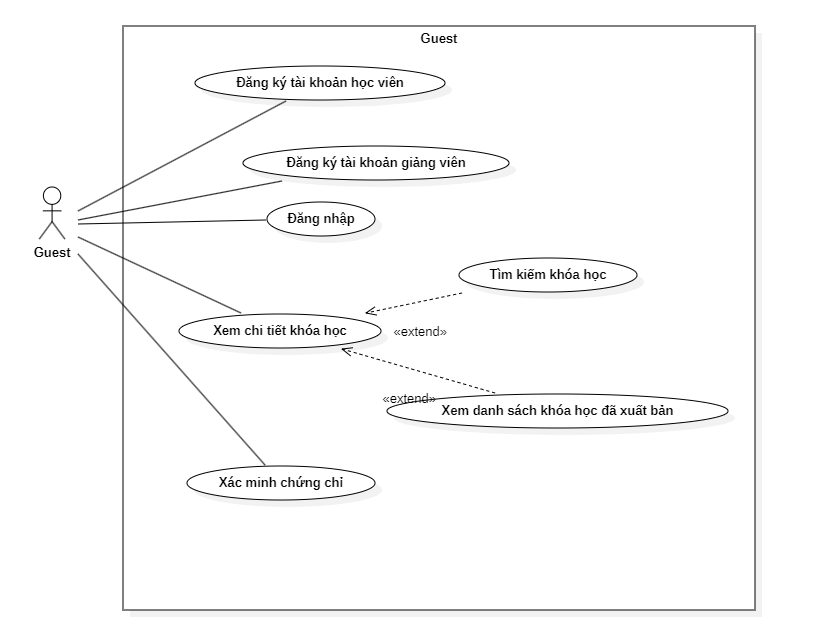
**4.3.1 Yêu cầu người dùng**

Đối tượng sử dụng hệ thống bao gồm: người dùng ẩn danh, học viên, giảng viên và Admin.

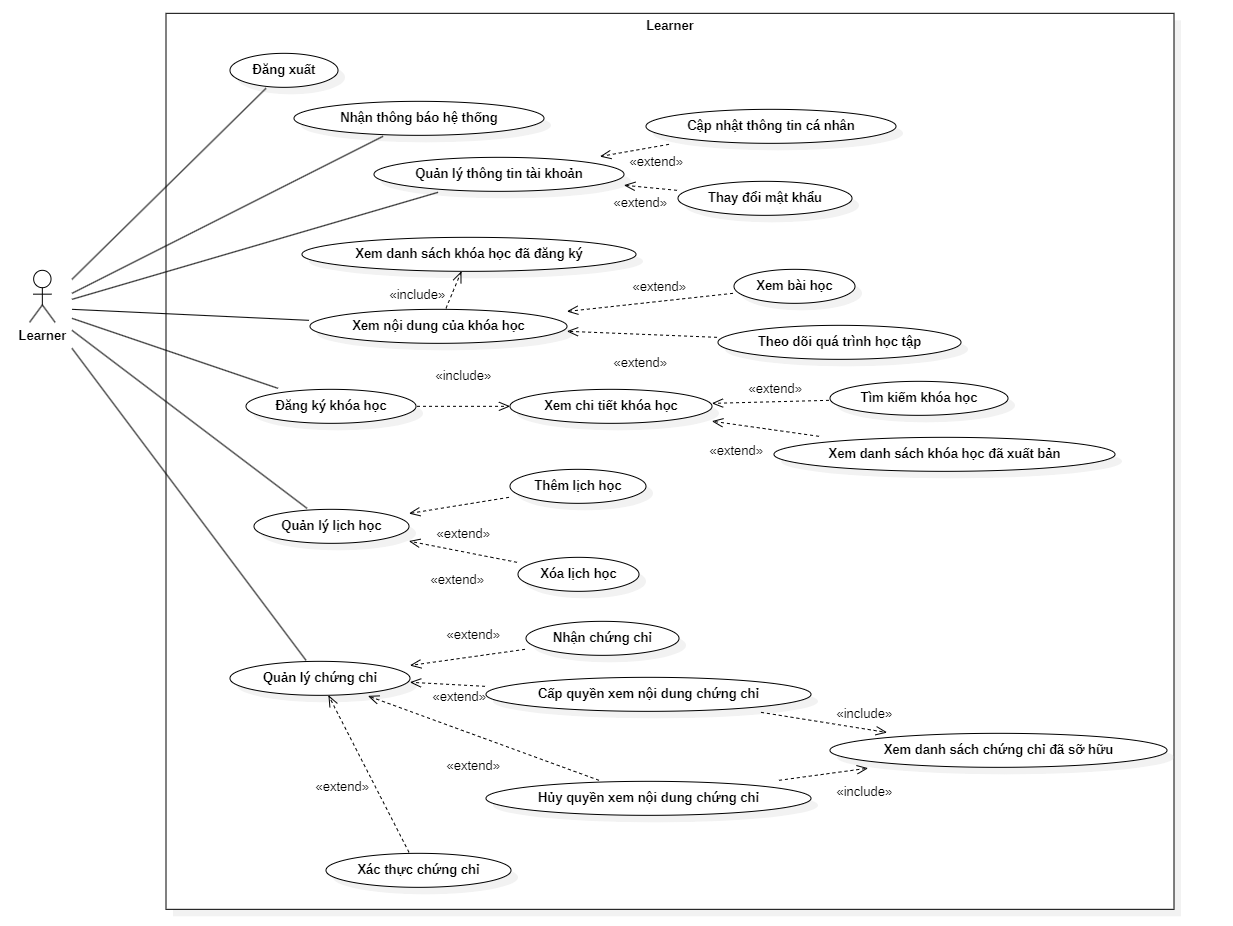
|  |  |  |
| --- | --- | --- |
| Mã đối tượng | Tên đối tượng | Mô tả |
| US-01 | Guest | - Là đối tượng khi truy cập vào hệ thống nhưng chưa đăng nhập.  - Có thể sử dụng các chức năng cơ bản như xem thông tin chi tiết khóa học, tìm kiếm khóa học, tra cứu chứng chỉ, đăng ký, đăng nhập. |
| US-02 | Learner | - Là đối tượng được nâng cấp từ US-01 sau khi đăng nhập với vai trò là học viên  - Có thể sử dụng các chức năng của học viên  + Xem chi tiết, cập nhật thông tin cá nhân  + Đổi mật khẩu hoặc lấy lại mật khẩu khi quên mật khẩu  + Xem chi tiết khóa học, tìm kiếm khóa học  + Thời khóa biểu + Xem bài học  + Nhận chứng chỉ NFT + Nhận thông báo realtime |
| US-03 | Instructor | - Là đối tượng được nâng cấp từ US-01 sau khi đăng nhập với vai trò là giảng viên - Có thể sử dụng các chức năng của giảng viên và học viên + Thêm, sửa, xóa khóa học + Thêm, sửa, xóa chương học + Thêm, sửa, xóa bài học |
| US-04 | Admin | - Là đối tượng có toàn quyền sử dụng toàn bộ chức năng của hệ thống  - Có thể sử dụng các chức năng của Admin  + Quản lý user  + Quản lý danh mục  + Quản lý khóa học |

**4.3.2 Mô hình Use Case**

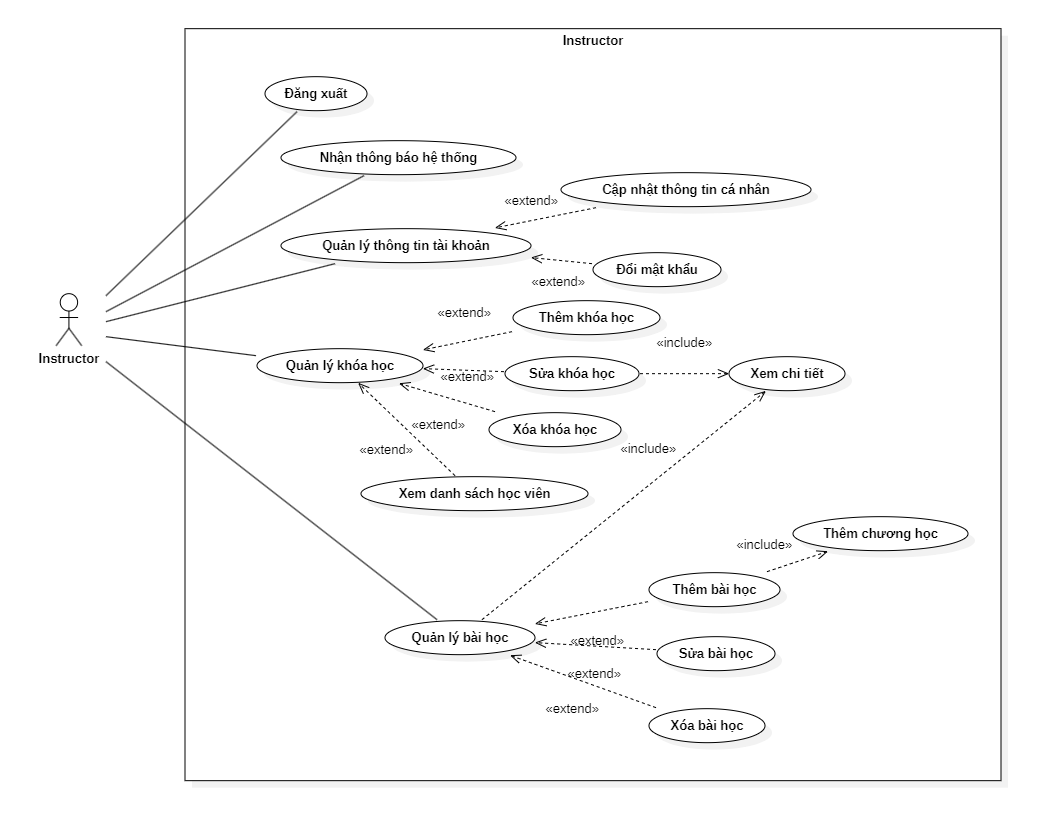
4.3.2.1 Use case của Guest



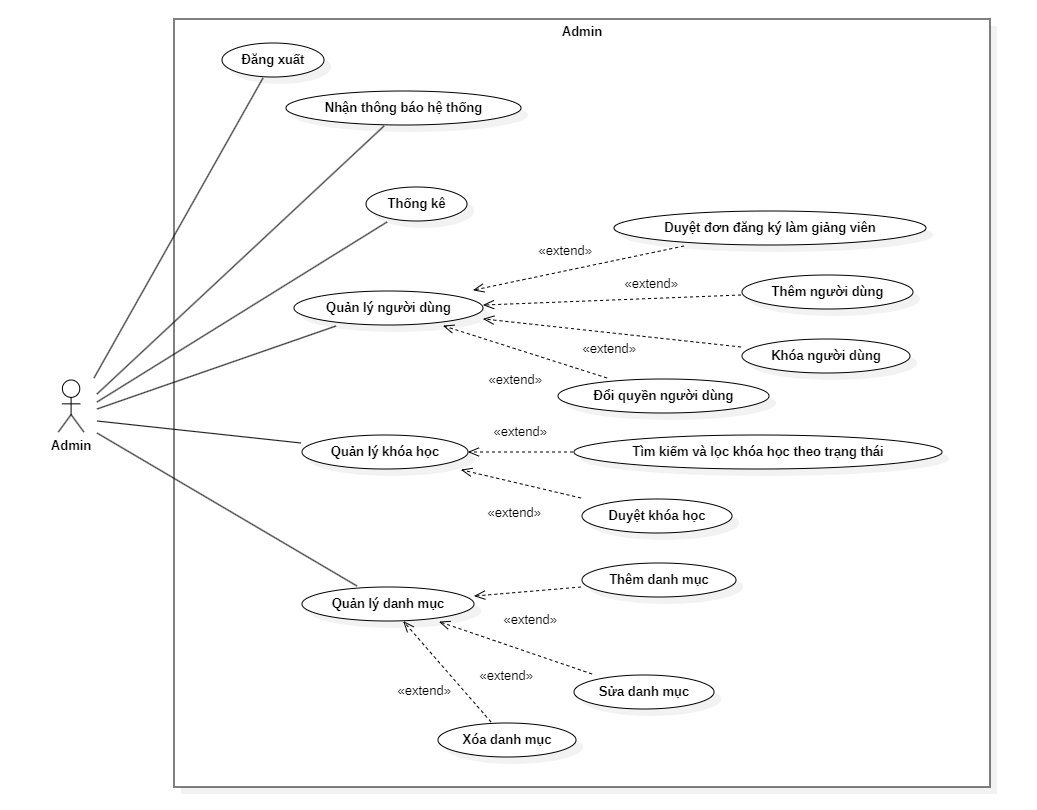
4.3.2.2 Use case của Learner



4.3.2.3 Use case của Instructor



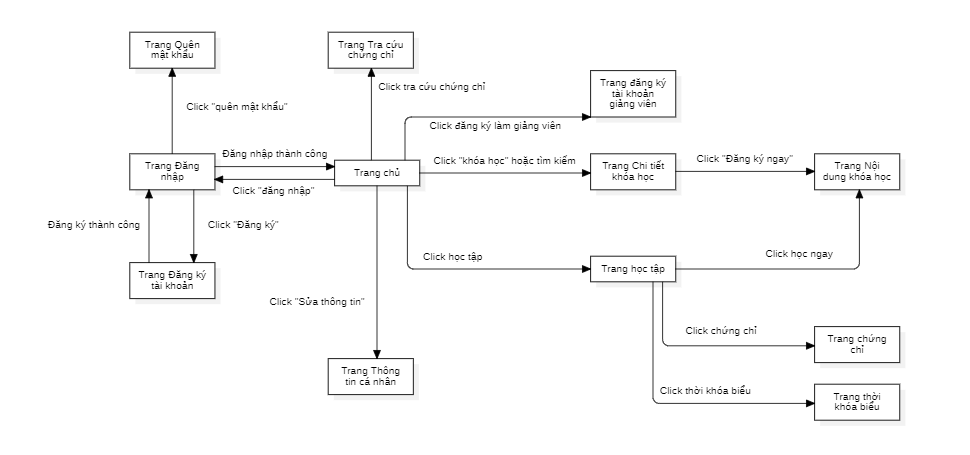
4.3.2.4 Use case của Admin



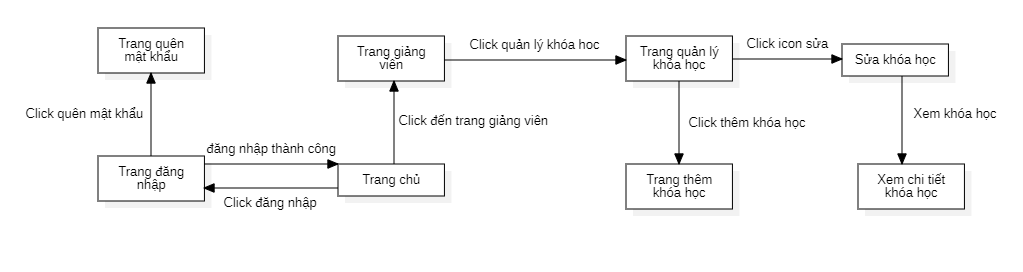
4.3.3 Yêu cầu chức năng

4.3.3.1 Luồng chạy màn hình

4.3.3.1.1 Màn hình của Learner



4.3.3.1.2 Màn hình của Instructor



4.3.3.1.3 Màn hình của Admin

**TÀI LIỆU THAM KHẢO**

1. <https://en.wikipedia.org/wiki/Spring_Framework>
2. <https://viblo.asia/p/spring-architecture-DzVkpBKgGnW>
3. <https://medium.com/javarevisited/10-spring-boot-features-that-make-java-development-easier-6bbee204c867>
4. <https://200lab.io/blog/nextjs-la-gi>
5. <https://chain.link/education-hub/blockchain#who-invented-blockchain-technology>
6. <https://chain.link/education/smart-contracts>