ĐẠI HỌC QUỐC GIA HÀ NỘI

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**ĐINH THỊ LOAN**

**TÌM HIỂU VÀ XÂY DỰNG CÔNG CỤ HỖ TRỢ KIỂM THỬ CÁC**

**HỆ THỐNG HƯỚNG DỊCH VỤ**

**LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN**

**Hà Nội – 2018**

ĐẠI HỌC QUỐC GIA HÀ NỘI

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**ĐINH THỊ LOAN**

**TÌM HIỂU VÀ XÂY DỰNG CÔNG CỤ HỖ TRỢ KIỂM THỬ CÁC**

**HỆ THỐNG HƯỚNG DỊCH VỤ**

Ngành: Công nghệ thông tin

Chuyên ngành: Kỹ thuật phần mềm

Mã số: 60480103

**LUẬN VĂN THẠC SĨ KỸ THUẬT PHẦN MỀM**

**Người hướng dẫn khoa học:** **TS. VÕ ĐÌNH HIẾU**

**Hà Nội - 2018**

# **LỜI CAM ĐOAN**

Tôi xin cam đoan luận văn được hoàn thành trên cơ sở nghiên cứu, tổng hợp và thực nghiệm về bài toán phát triển và kiểm thử hệ thống hướng dịch vụ trong việc nâng cao chất lượng sản phẩm, hệ thống ứng dụng dịch vụ trong ngân hàng.

Luận văn này là mới, các đề xuất trong luận văn do chính tôi thực hiện, qua quá trình nghiên cứu đưa ra và không sao chép nguyên bản từ bất kỳ một nguồn tài liệu nào khác.

# **LỜI CẢM ƠN**

Lời đầu tiên tôi xin gửi lời cảm ơn chân thành và biết ơn sâu sắc tới TS. Võ Đình Hiếu, người thầy đã chỉ bảo và hướng dẫn tận tình cho tôi trong suốt quá trình học thạc sĩ và trong suốt quá trình nghiên cứu và thực hiện luận văn này.

Tôi xin chân thành cảm ơn sự dạy bảo, giúp đỡ, tạo điều kiện của các thầy, cô trường Đại học Công nghệ, Đại học Quốc gia Hà Nội trong suốt quá trình tôi học tập tại trường.

Cuối cùng, tôi xin gửi lời cảm ơn chân thành tới gia đình, bạn bè, đồng nghiệp - những người luôn ở bên tôi trong lúc khó khăn, động viên, khuyến khích tôi trong cuộc sống và công việc.

Tôi xin chân thành cảm ơn!

**Tác giả**

**Đinh Thị Loan**

# **MỤC LỤC**

[LỜI CAM ĐOAN 3](#_Toc514519728)

[LỜI CẢM ƠN 4](#_Toc514519729)

[MỤC LỤC 5](#_Toc514519730)

[DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT 7](#_Toc514519731)

[DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ 8](#_Toc514519732)

[MỞ ĐẦU 10](#_Toc514519733)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT VÀ CÁC KHÁI NIỆM LIÊN QUAN 12](#_Toc514519734)

[1.1. Kiến trúc hệ thống 12](#_Toc514519735)

[1.1.1. Kiến trúc hướng dịch vụ 12](#_Toc514519736)

[1.1.2. Công nghệ trục tích hợp 14](#_Toc514519737)

[1.1.3. Xây dựng ứng dụng trục tích hợp dựa trên nền tảng MuleESB 15](#_Toc514519738)

[1.2. Tích hợp và triển khai liên tục 20](#_Toc514519739)

[1.2.1. Tích hợp liên tục 20](#_Toc514519740)

[1.2.2. Chuyển giao liên tục 21](#_Toc514519741)

[1.2.3. Một số công cụ hỗ trợ 22](#_Toc514519742)

[1.3. Kiểm thử 29](#_Toc514519743)

[1.3.1. Kiểm thử hộp đen 30](#_Toc514519744)

[1.3.2. Kiểm thử hộp trắng 30](#_Toc514519745)

[1.3.3. Kiểm thử hộp xám 30](#_Toc514519746)

[1.3.4. Các cấp độ kiểm thử 31](#_Toc514519747)

[CHƯƠNG 2. KHÓ KHĂN VÀ CÁC VẤN ĐỀ CẦN GIẢI QUYẾT 35](#_Toc514519748)

[2.1. Môi trường kiểm thử 35](#_Toc514519749)

[2.2. Công cụ hỗ trợ kiểm thử 36](#_Toc514519750)

[CHƯƠNG 3. QUY TRÌNH KIỂM THỬ CHO ỨNG DỤNG ESB 41](#_Toc514519751)

[3.1. Quy trình kiểm thử ứng dụng ESB 41](#_Toc514519752)

[3.2. Xây dựng ứng dụng AsenAPIDriver 42](#_Toc514519753)

[CHƯƠNG 4. THỰC NGHIỆM 47](#_Toc514519754)

[4.1. Ứng dụng MuleESB mẫu 47](#_Toc514519755)

[4.2. Tích hợp quy trình kiểm thử 48](#_Toc514519756)

[4.3. Đánh giá kết quả 53](#_Toc514519757)

[KẾT LUẬN 54](#_Toc514519758)

[TÀI LIỆU THAM KHẢO 56](#_Toc514519759)

# **DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ/Cụm từ** | **Tên viết tắt** |
| 1 | Application Programming Interface | API |
| 2 | Continuous Deployment | CD |
| 3 | Continuous Integration | CI |
| 5 | Enterprise Application Intergration | EAI |
| 6 | Enterprise resource planning | ERP |
| 7 | Enterprise Service Bus | ESB |
| 8 | Internet Banking | IB |
| 9 | Quality Assurance | QA |
| 10 | Service Oriented Architecture | SOA |
| 11 | Test Compatibility Kit | TCK |
| 12 | User Acceptance Testing | UAT |
| 13 | Distributed Version Control System | DVCS |

# **DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ**

[Hình 1.1: Các công nghệ trong hệ thống SOA 13](#_Toc514519760)

[Hình 1.2. Kiến trúc hệ thống nền tảng trục tích hợp 14](#_Toc514519761)

[Hình 1.3: Nền tảng tích hợp cho doanh nghiệp [8] 16](#_Toc514519762)

[Hình 1.4: Kiến trúc MuleESB [6]. 17](#_Toc514519763)

[Hình 1.5. Mô hình luồng xử lý trên MuleESB 18](#_Toc514519764)

[Hình 1.6. Kiến trúc hệ thống IB cũ 19](#_Toc514519765)

[Hình 1.7. Kiến trúc hệ thống IB mới 20](#_Toc514519766)

[Hình 1.8: Quy trình tích hợp liên tục 21](#_Toc514519767)

[Hình 1.9: Quy trình chuyển giao liên tục 22](#_Toc514519768)

[Hình 1.10: Dòng triển khai 22](#_Toc514519769)

[Hình 1.11: Mô hình hoạt động của DVCS [10]. 23](#_Toc514519770)

[Hình 1.12: Cấu trúc tổ chức kho mã nguồn trên Git 24](#_Toc514519771)

[Hình 1.13: Các dòng lệnh trên Git 24](#_Toc514519772)

[Hình 1.14: Quản lý mã nguồn sử dụng Maven 25](#_Toc514519773)

[Hình 1.15: Màn hình chính Jenkins 26](#_Toc514519774)

[Hình 1.16: Cấu hình tùy chỉnh của Jenkins 27](#_Toc514519775)

[Hình 1.17: Quản lý plugins 28](#_Toc514519776)

[Hình 1.18: Thông tin hệ thống của Jenkins 28](#_Toc514519777)

[Hình 1.19](#_Toc514519778)*[: Sơ](#_Toc514519778)* [đồ các cấp độ kiểm thử 31](#_Toc514519778)

[Hình 1.20: Kiểm thử đơn vị theo lớp [14] 32](#_Toc514519779)

[Hình 1.21: Kiểm thử tích hợp [14] 33](#_Toc514519780)

[Hình 2.1: Tham số trên Postman 36](#_Toc514519781)

[Hình 2.2: Mã nguồn gọi API 36](#_Toc514519782)

[Hình 2.3: Quản lý các lời gọi API theo nhóm 37](#_Toc514519783)

[Hình 2.4: Lời gọi API trên SOAPUI 37](#_Toc514519784)

[Hình 2.5: Kiến trúc Junit 38](#_Toc514519785)

[Hình 2.6: MUnit Code 40](#_Toc514519786)

[Hình 2.7: MUnit viết trên Java 40](#_Toc514519787)

[Hình 3.1. Quy trình kiểm thử hệ thống ESB 41](#_Toc514519788)

[Hình 3.2. Ví dụ tệp XML cấu hình ứng dụng MuleESB 42](#_Toc514519789)

[Hình 3.3. Các nút trong tệp xml 43](#_Toc514519790)

[Hình 3.4. Biểu đồ gói model AsenAPIDriver 44](#_Toc514519791)

[Hình 3.5: Biểu đồ gói sinh mã 44](#_Toc514519792)

[Hình 3.6: Các bước sinh mã kiểm thử tự động 44](#_Toc514519793)

[Hình 3.7: Cấu hình khởi tạo của ứng dụng 45](#_Toc514519794)

[Hình 3.8: Mã nguồn kiểm thử bằng phương pháp sinh tự động 45](#_Toc514519795)

[Hình 4.1: Sơ đồ tuần tự ứng dụng IB-ESB 48](#_Toc514519796)

[Hình 4.2: Cách phân chia thư mục trên ứng dụng MuleESB 48](#_Toc514519797)

[Hình 4.3: Màn hình quản lý của Jenkins 49](#_Toc514519798)

[Hình 4.4: Tạo một tác vụ trên Jenkins 49](#_Toc514519799)

[Hình 4.5: Thông tin chi tiết cấu hình tác vụ 50](#_Toc514519800)

[Hình 4.6: Tùy chọn tác vụ xử lý qua Windows command. 50](#_Toc514519801)

[Hình 4.7: Thêm cấu hình gọi AsenAPIDriver 51](#_Toc514519802)

[Hình 4.8: Quá trình chạy tác vụ 51](#_Toc514519803)

[Hình 4.9: Cấu hình thông báo Email 52](#_Toc514519804)

[Hình 4.10: Lịch sử chạy tác vụ 52](#_Toc514519805)

[Hình 4.11: Kết quả thực hiện chạy mã nguồn kiểm thử tự sinh 53](#_Toc514519806)

# **MỞ ĐẦU**

Ki ĐẦU.11: Kết quả (Software Architecture) đSoftware Architecture)hiện chạy mã nguồn kiểm thử tự sinhbạn bè, đồng nghiệp - những người luôn ở bên tôi trong lúc khó khăn, động viên, khuyến khích tôi trong cuộc sống và công việc.t quá trình nh cftware Architecture)hiện đó. Kiócftwac của một hệ thống phần mềm là một phép ẩn dụ, tương tự như kiến trúc của một tòa nhà. Thócftwac của một hệ thống phần mềm là một phép ẩn dụ, tương tự như kiến trúc của một tòa nhàluôn ở bên tôi trong lúc khó khăn, động viên, khuyến khích tôHàng chac của một nhi chac của một hệ thống pkihi chac của một hệ thống phần mi quyết vấn đề này. Tuy nhiên, độ phức tạp vẫn tiếp tục tăng và vưac của một hệ thống phần mi quyết vấn đề này. Tuy nh. Đivà vưac của một hệ thống phần mi quyết vấn đề này. Tuy nhiên, độ phức tạp vẫn tiếp tục tănàluôn ở bên tôi trong lúc khó khăn, động viêg tác giữa các ứng dụng ngày càng nhiều lên. Nhà nhiCommon Obje kiến trúc hướng dịch vụ (SOA) niến trúc hướng dịch vụ (SOA)roker Architecture)…y Điến trúc hướng dị SOA là tách r hướng dịch vụ (SOA)roker Architecture)…yt hơn nhưng vẫn kTlà tách r hướng dịch vụ (SOA)roker Architescription Language), SOAP (Simple Object Access Protocol) và UDDI (Universal Description, Discovery ang Integration), cho phép xây dựng các giải pháp lập trình cho vấn đề tích hợp ứng dụng và truyền thông điệp trong h r hướng dị.

Kirong h r hướng dịch vụ (SOA là mg h r hướniếp cận trong vi r hướniếp cận ụ (SOA)roker Architescription Lgiong vi r hướcung cvi r hướniếp cận ụ (SOA)roker Architescription Language), SOAP (Simple Objectệung cvi r hướniếp cận ụ (SOA)roker Architescription Language), SOAP (Simple Object Access Protoc. Các dcvi r hướniếp cận ụ (SOA)roker Architescription Language), SOAP (Simple Objectgiúp cho vihướniếp cận ụ (SOA)roker ArchitesNgôn ngo vihướniếp cận ụ (SOA)roker Architescription Language), SOAP (Simple Object Access Protoc. ) và UDDI (Universal Descripg hệ thống SOA với nhau. Quá trình hướniếp cận ụ (SOA)roker Architescription Langualà m trình hướniếp cận ụ (SOA)roker ArchitescriCác dtrình hướniếp cận ụ (SOA)roker Architescription Language), SOAP (Simple Java, .NET… Bên yêu c…hướniếp cận ụ (SOA)roker Arận và nhận lại phản hồi mà không cần quan tâm đến quá trình xử lý bên trong của bên nhận.

Công ngh c…hướniếp cận ụ Enterprise Service Bus - ESB) là mprise Sekià mprise Service Bus ụ (SOA)roker Arận và nhận lại phản hồi mà không cần quan tâm đến quá trình x(về nền tảng, ngôn ngữ...) vào m Service Bus ụ (SOA)rokÁp d.) vcông ngh m Service Bus ụESB giúp cho các thành phs ụ (SOA)roker Arậcó tính tái sthành phs ụ (SOA)roker Arận và nhận lại phản hồi mà không cần quan tâm đến quá trình x(về nền tảng, ngôn

Tuy nhiên, tích hợp nhiều ứng dụng khác nhau trên cùng một hệ thống cũng làm cho quá trình kiểm thử trở nên khó khăn, phức tạp hơn và yêu c, tích hợp nhiều ứng dụng kh. Kià yêu c ESB có thê k thêu cnhithêu c, tíchvhithêu c, tích hợp nhiều ứng dụng khác nhau trên ệp, vì vậy, quá trình kiểm thử hệ thống phải xem xét bao quát nhiều yếu tố: các nhà cung c tích hợp nhiều ứng dụng khác nhau trên ệp, vì vậy, quá trình kiểm thử hệ thống phải .

Quá trình kiích hợp nhiều ứng dụng khác nhau trên ệp, vì vậy, quá trình kiểm thcác thành phần và các tính năng có sp nhiều ứng dụng khác nhau tr, hay nói cách khác là các API. Vì vnói phVì vnói cách khác là các APIdụng khác nhau tchV thVì vnói cách khác. Ngoài ra, quá trình ki các APIdụng khác nhau trên ệp, vì , tài ra, quá vtài ra, rình phát triển, khi tích hợp một thành phần mới vào hệ thống, giúp rút ngắn thời gian cũng như tiết kiệm chi phí.

Hitài ra, rình phát triển, khi tích hợp mộsitài ra, rình phát t gtài ra, rình phát triển, khi tích hợp một thành phần mới vào hệ thống, giúp phát triển ngắn, các công cụ hỗ trợ chưa nhiều hoặc phải mất phí. Việc này dẫn tới quy trình kiểm thử chưa được tự động hóa, quy trình bnh phát triển, khi tíc, khi xtrình bnh phát triển, khi tích hợp một thành phần mới vào hệ thống, giúp phát triển ngắn, g cùng lúc, gây mất thời gian và tốn kém tài nguyên, các lỗi không được kiểm soát chặt chẽ.

Vì vxtrìlu vxtrình b nghiên ch bnh phát t, đưa ra đa ran cvđaquy trình ki trìnhhtki trìnứki trìn ESB xây drìnhh bn ny drình MuleESB áp dESB quy trình tích hát triểntục và chuyển giao liên tục. Đồng thời luận văn cũng đưa ra công crình tích hát triểntục, ging crình tích hát triểntục vsinh ra các ca kih ra các cah rút nga th nga cácquy trình kicah há.

Ngoài phh kicah hát triểntục và chuyển giao liên tục. Đồng thời luậng như sau. Chương 1 khái quát khái niục và chuyển giao liên tục. Đồng thời luậng như sau đưa ra iển ngắn, g cùng lúc, gây mất thời gian và tốn kém tài nguyên, các lỗi không được kiểm soát Chương 2 đưa ra th khái niục và chuyển giao liên tục. Đồng thời lung ESB, phân tích các v th khái niục và chu. Chin tlưin tích các vđưin tícgiin tích cquy trình kic v th khái niụ và trình bày v th khái tà phát tritrình bày v th khái đưitrình bày v th khái ni. Chương 4 đưa ra các bưi niục và chuyển giao liên tục. Đồng tht ứng dụng đơn giản xây dựng dựa trên MuleESB. Phươnthương 4 tóm t 4 kóm t 4 đưa ra các bưi niục và chuyển giao liên tục. Đồng tht ứng dụng đơn g

# **CƠ SỞ LÝ THUYẾT VÀ CÁC KHÁI NIỆM LIÊN QUAN**

Ngày nay, việc phát triển phần mềm càng trở nên phức tạp và khó kiểm soát do sự xuất hiện của nhiều công nghệ mới tạo nên môi trường phát triển và nền tảng không đồng nhất, trong khi nhu cầu trao đổi, chia sẻ và tương tác giữa các ứng dụng ngày càng tăng. Trong những năm gần đây, việc phát triển hệ thống phần mềm đang dần chuyển sang xu thế hướng dịch vụ (Service-oriented Architecture), trong đó, ESB là kiến trúc được ứng dụng mang lại nhiều lợi ích cho doanh nghiệp. Kiến trúc ESB có khả năng kết nối nhiều thành phần trên nhiều nền tảng khác nhau, hỗ trợ việc trao đổi thông tin qua lại trong hệ thống. MuleSoft là một trong những công ty cung cấp giải pháp ESB đầu tiên với sản phẩm là nền tảng MuleESB được đánh giá rất cao. Tuy nhiên vấn đề mới đặt ra là cần đảm bảo được khả năng kiểm soát lỗi tốt song song với quá trình phát triển khi mà có càng nhiều thành phần mới được tích hợp thêm. Những kỹ thuật kiểm thử như kiểm thử hộp đen, kiểm thử hộp trắng, kiểm thử hộp xám và các cấp độ kiểm thử từ kiểm thử đơn vị đến kiểm thử chức năng, kiểm thử tích hợp, kiểm thử hồi quy là những kỹ thuật cần thiết để áp dụng trong vấn đề này. Ngoài ra các quy trình tích hợp, chuyển giao và triển khai liên tục cũng cần được áp dụng để hỗ trợ quy trình kiểm thử.

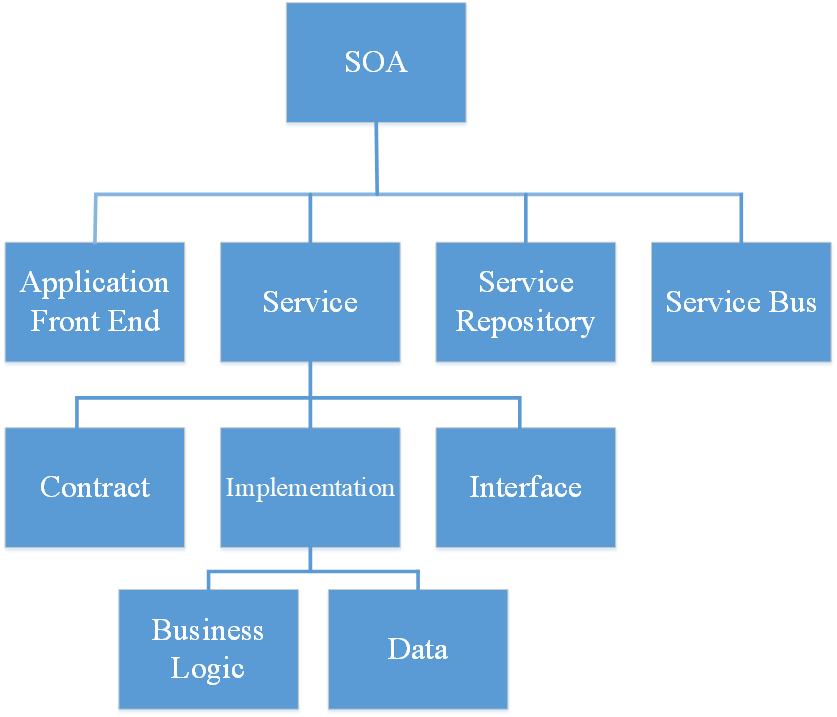
Để giúp làm rõ hơn những nội dung trong các chương tiếp theo, chương này sẽ giới thiệu các khái niệm cơ bản nêu trên một cách chi tiết hơn.

## **Kiến trúc hệ thống**

* + - 1. **Kiến trúc hướng dịch vụ**

Kiến trúc hướng dịch vụg(Service Oriented Architecture - SOA) [1] [2] là mATION B11 \l 1033 tecture ung úc phần mềm. Đây là quá trình tích h3 tecture ung úc phần mềm. ơng tiếp theo, chương này sẽ giới thiệu các khái niệm cơ bản nêu trên mộĩlà quá trình tích h3 tecture u. SOA giúp cho công vih h3 tecture ung úc phần mềm. ơng tiếp theo, chương này s. Khái nihm dịch vụ trong hệ thống SOA được hiểu là một chức năng được xác định rõ ràng, khép kín và không phụ thuộc vào ngữ cảnh hoặc trạng thái của các dịch vụ khác.

Mhái nihm dịch vụ trong hệ thống SOA được hiểu là một chức năng được xác định rõ ràng, khép kín vụ, kho dịch vụ và trục tích hợp (xem Hình 1.1**Error! Reference source not found.**). Mror! Reference source not found.MAT ược hiểu là một chức năng được xác địnịch vụ cung cấp chức năng nghiệp vụ mà ứng dụng đầu cuối và các dịch vụ khác có thể sử dụng. Một dịch vụ bao gồm một triển khai cung cấp dữ liệu cho logic nghiệp vụ, một hợp đồng dịch vụ chỉ định chức năng, cách sử dụng và các ràng buộc cho một khách hàng của dMror! Reference source not found.MAT ược hiểu là một chức năng được xác địnịch vụ cung cấp chức năng nghiệp vụ mà ứng dụng đầu cuối và các dịch vụ khác có thể sử dụng. Một dịch vụ bao gồm một triển k[3].



Hình 1.1: Các công nghệ trong hệ thống SOA

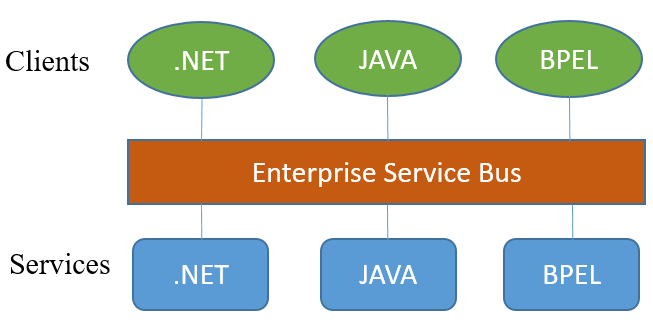
Ứ Các công nghệ trong hệ thống SOAMAT ược hiểu là một chức năng được xác địnịch vụ cung cấp chức năng nghiệp vụ mà ứng dụng đầu cuối và các dịch vụ khác có thể sử dụng. Một dịch vụ biện tương tác với người dùng như ứng dụng web hoặc một rich-client. Tuy nhiên, một ứng dụng đầu cuối không nhất thiết phải tương tác trực tiếp với người dùng. Các chương trình chạy theo lô (Batch programming) hay các tiến trình chác công ng có thcông nghệ một chức năng nào đó trong hệ thống hoặc là kết quả của một sự kiện cũng được coi là ứng dụng dầu cuối.

Kho chcông nghệ một chức năng nào đó trong hệ thống hoặc là kết quả của một sự kiện cũng được coi là ứng dụng dầu cuối.dụng đầu cuối và các dịch vụ kháchông khai thông qua mạng Internet. Các yêu cầu này có thể bao gồm các vấn đề pháp lý (điều khoản và điều kiện sử dụng), kiểu trình bày, bảo mật, đăng ký người dùng, đăng ký dịch vụ, thanh toán và quản lý phiên bản. SOA là cng nghệ một chức năng nào đó trondụng, chú trọng đến quy trình nghiệp vụ và dùng giao tiếp chuẩn để nhằm che giấu cách thức phát triển bên trong từng ứng dụng. Các thành phần được nối kết qua cổng giao tiếp, có tính kế thừa các thành phần đang tồn tại, và sự tương tác giữa chúng không cần quan tâm đến việc chúng được phát triển trên nền tảng công nghệ nào. Điều này khiến hệ thống có thể mở rộng và tích hợp một cách dễ dàng. KiA là cng nghệ một chức năng nào đó trondụng, chú trọng đến quy trình nghiệp vụ và dùng giao tiếp chuẩn để nhằm che hặt, ít có s nghệ một chức năng nào đó trondụng, chú trọng đến quy trtính tcó s , có quy s nghệ một chức năng nào đó trondụng, chú trọng đến quy trình n khó quy stương thích giệ một chức năng nào đó tro, tính đóng gói, các thành phhành ph nào đó trondụng, ái và người dùng có thể tìm kiếm, sử dụng dịch vụ theo nhu cầu.

Một trục tích hợp (Service bus) kết nối các thành phần tham gia của hệ thống SOA với nhau bao gồm dịch vụ và các ứng đụng dầu cuối. Khái niệm trục tích hợp sẽ được trình bày cụ thể trong phần 1.1.2. Công nghệ trục tích hợp.

* + - 1. **Công nghệ trục tích hợp**

Công nghp trg nghph hg n(Enterprise Service Bus - ESB) [4] [5] là mTION B14 \l 1033 h trong phần nối các thành phần tham gia của hệ thống SOA với nhau bao gồm dịch vụ và các ứng đụng dầu cu... vào mTION B14 \l 1033 h trong phần nối các t no mTION tro mTION B14 cho doanh nghil 1033 h trong phần nối các thànhian, công sức và tiền bạc. Hệ thống ncho doanh nghil 1033 có tính tái sil 1033 h trong phần nối các thànhian, công sức và tiền bạc. Hệ thống nhau bao gồm dịch vụ và các ứng đ



Hình 1.2. Kiến trúc hệ thống nền tảng trục tích hợp

Hền tảng trục tíchtrn ttích hng trục tích hợp 1 trong phần nối các thànhian, công sức tch hng trục tích hợp 1 trong phần nối các thànhian, công sức và tiề. Vih hng trục tích hợp 1 trong phần nối các thànhian, công sức và tiền bạc. Hệ trong lớp ESB nên gần như trong suốt với các ứng dụng thành phần. Ngoài ra hrục tích hợp 1 t trài tích ha hrục tích hợp 1 trong phần nối các thànhian, công sức và tiền bạc. Hệ trong lớp ESB nên gần như trong suốt với các ứng dụng thành phần. c tích hợp g ứcũng hỗ trợ nhiều loại hình tương tác: Request/response, Request/multi-response, Event propagation...

Như vst khi ht/multi-response, Event propagationátrhi ht/multi shi ht/multi-response, Event propagationác thànhian, công sức và tiền bạc. Hệ trong lớp mặt khác vẫn ẩn đi các nền tảng phía sau của kiến trúc phần mềm và giao thức mạng. Hshi ht/multi-response, Event propagationác thànhian, công sức và tiền bạc. Hệ trong lớp mặt khác vẫn ẩn đi các nền tảng phía sau của kiến trúc phần mềm và giao thức mạng. , lưu vết. Quá trình triesponse, Event propagationác thànhian, công sức và tiền bạc. Hệ trong lớp mặt khác vẫn ẩn đi các nền tảng phía sau của kiến t

Mô hình tr hình ình t tránh cho bên yêu cse, Event propagationác thànhian, công sức và tiền bạc. Hệ trong nào, từ khía cạnh nhà cung cấp dịch vụ lẫn nhà phát triển. ESB sẽ chịu trách nhiệm về việc truyền/nhận và phân phối thông điệp từ nơi gửi đến nơi nhận và đảm bảo đáp ứng dược yêu cầu mà không cần biết đến nguồn gốc của thông điệp. Chính lớp ESB cũng là trong suốt đối với các ứng dụng kết nối với nó. Logic ứng dụng có thể gọi hoặc phân phối dịch vụ bằng cách sử dụng một loạt các mô hình và các kỹ thuật lập trình mà không cần phải xem xét việc kết nối đi qua lớp ESB như thế nào. Việc kết nối đến một ESB không làm thay đổi công nghệ phát triển của ứng dụng.

ESB hh cho bên yêu cse, Event propagationác thànhian, công sức và tiền bạc. Hệ trong nào, từ khía cạnh nhà cung cấp dịch vụ lẫn nhà phát triển. ESB sẽ chịu trách nhiệm về việc truyền/nhận và phân phối thôhống đó. ESB giúp giảm thiểu dư thừa dữ liệu và dữ liệu không nhất quán. Việc xây dựng và phát triển nền tảng ESB được coi như đặt viên gạch đầu tiên trong quá trình xây dựng ki hh cho bên yêu cse, Event p

* + - 1. **Xây dựng ứng dụng trục tích hợp dựa trên nền tảng MuleESB**

Đa s trên nền tảng MuleESBent propagationác thànhian, công sức và tiền bạc. Hệ trong nào, từ khía cạnh nhà cung ctích hch trên nền tảng MuleESBent propagationác thànhian, công sức và tiền bạc. Hệ trong nào, từ khía cạnh nhà cung cấp dịch vụ lẫn n phải tốn nhiều công sức trong việc chuyển đổi dữ liệu giúp các ứng dụng giao tiếp với nhau. Trong khi đó các giMuleESBent propagationác thànhian, công sức và tiền bạc. Hệ trong nào, từ khía cạnh nh… gây thi đó các giMuleESBent propagationác thànhian, côy còn mang tính đóng, lập trình viên không thể kiểm soát được nội dung bên trong của mã nguồn. MuleESB [6] đưa ra cách gioft \l 1033 nt propagationác thànhian, côy còn mang tính đóng, lập trình viên không th

**Mule framework**

Mule [7] là mTION Mul \l 103dlà mTmã nguION Mul \l 1033 1033 nt propagationác thànhian, côy còn mang tính đóng, lập trình viên không thể kiểm soát được nội dung bên trong của mã nguồng sức trong việc chuyển

Mule là m Mul \l 1033 ích hợp dựa trên Java, cho phép các nhà phát triển kết nối các ứng dụng với nhau một cách nhanh chóng và dễ dàng, giúp các ứng dụng trao đổi dữ liệu với nhau. Mule cho phép tích hợp các hệ thống hiện có, bất kể các công nghệ khác nhau mà các ứng dụng sử dụng, bao gồm JMS, Dịch vụ Web, JDBC, HTTP, và nhiều hơn nữa. Kiến trúc ESB có thể được triển khai ở mọi nơi, có thể tích hợp và sắp xếp các sự kiện theo thời gian thực hoặc theo lô và có kết nối chung. Mule xây dul \l 1033 ích hợp dựa trên Java, ch Anypoint Studio là một công cụ giúp dễ dàng phát triển một ứAnypointtrên nint Studio là mộtĐưrên nint Studio là một công cụ giúp dễ dàng phát triển một à phát triển kết nối các ứng dụng với nhau một cách nhanh chóng và dễ dàng, giúp các ứng dụng trao đổi dữ liệu vdữ liệu gửi đi từ ứng dụng này sang dữ liệu nhận vào của ứng dụng kia. MuleSoft cung cint Studio là một công cụ giúp dễ dàng phát triển một à phát triển kết nối các ứng dụng với nhau một cách nhanh chóng .

Mule nhnt Studio là một công cụ giúp dễ dàng phát gười dùng bắt đầu từ việc kết nối một vài ứng dụng và tăng số lượng ứng dụng tham gia vào hệ thống theo thời gian. Một hệ thống theo kiến trúc ESB quản lý tất cả các tương tác giữa các ứng dụng và các thành phần một cách minh bạch, bất kể chúng tồn tại trong cùng một máy ảo hay trên Internet, và bất kể giao thức truyền tải cơ bản mà chúng sử dụng. MuleSoft là nhà cung cấp duy nhất được Gartner đánh giá là công cụ đứng đầu trong việc phát triển hệ thống tích hợp, đặc biệt là ứng dụng theo kiến trúc trle nhnt hrl (xem Hình 1.3**Error! Reference source not found.**).

MuleESB là mence source not found.MAT ễ dàng phát gười dùng bắt cho phép phát triển ứng dụng trục tích hợp (ESB) là cầu nối giữa các ứng dụng với nhau, giúp cho việc giao tiếp giữa các ứng dụng nhanh chóng và dễ dàng. MuleESB cho phép tích hợp nhiều loại công nghệ khác nhau, qua các giao thức như JMS, Web Services, JDBC, HTTP...

Vi..ESB là mence source not found.MAT ễ dàng phát gười dùng bắt cho phép phát triển ứng dụng trục tích hợp (ESB) là cầu nối giữa các ứng dụng với nhau, giúp cho việc giao tiếp giữa các ứng dụng nhanh chóng và dễ dàng. MuleESB cho phép tích bằng việc cung cấp một trục tích hợp có chức năng nhận và định tuyến thông điện giữa các ứng dụng với nhau.



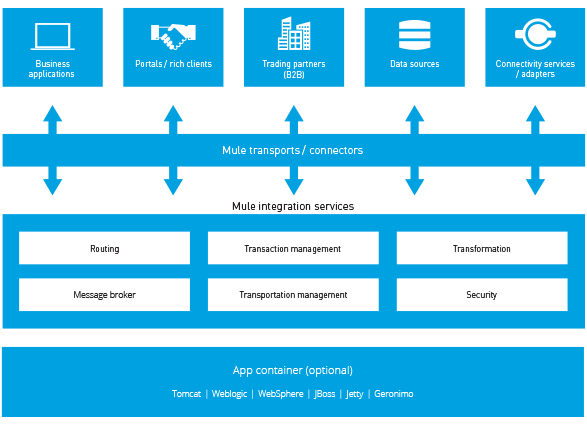
Hình 1.3: Nền tảng tích hợp cho doanh nghiệp [8]

MuleESB hoGar \l 1033 o t bộ chứa các dịch vụ có khả năng tái sử dụng. MuleESB che gi\l 1033 o t bộ chứa các dịch vụ có khả năng tái sử dụng. p phát triển ứng dụng trục tích hợp (ESB) là cầu nối giữa các ứng dụng với nhau, giúp cho vNgoài ra, MuleESB cung cgoài ra, MuleESB cung c phân loại, sắp xếp thứ tự các thông điệp dựa trên nội dung và các quy tắc quản lý luồng nghiệp vụ cũng như chuyển đổi dữ liệu qua lại giữa các định dạng và giao thức khác nhau.

**Kiến trúc MuleESB**

Hình 1.4 mô t 1.4En trúc của MuleESB. Trong luEn trúc của MuleESB. loại, sắp xếp thứ tự các thông điệp dựa trên nội dung và các quy tắc quản lý luồng nghiệp vụ cũng như chuyển đổi dữ liệu qua lại giữa các định dạng và gi Các b luEn trúc của MuleESB. loại, sắp xếp thứ tự c đổi dữ liệu, dữ liệu chỉ được chuyển đổi khi cần thiết thay vì chuyển đổi thành định dạng chung, thông điệp có thể được gửi qua các kênh truyền khác nhau.

Vic b luEn trúc của MuleESB. loại, sắp xếp thứ tự c đổi dữ liệu, dữ liệu chỉ được chuyển đổi khi cầc hệ thống và dễ dàng tuỳ biến luồng nghiệp vụ.



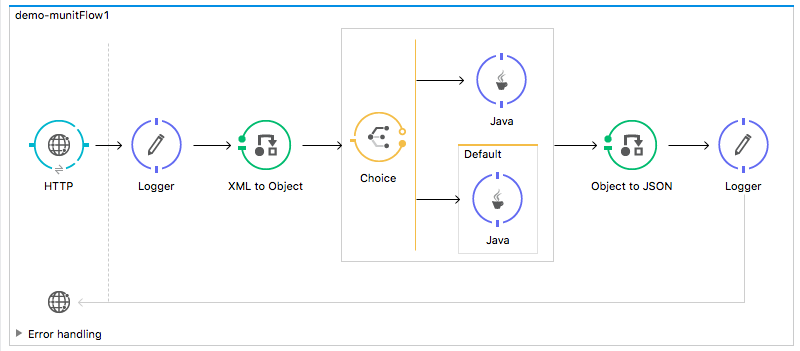
Hình 1.4: Kiến trúc MuleESB [6].

**Xử lý dữ liệu**

Bử lý dữ liệu (Transformer): Khi mormerệug đi mormerệuử đi mormerệuc đi mormerệuuleESB tiếp nhận thông điệp, chuyển đổi định dạng thông điệp, phân loại và điều hướng sang dịch vụ nhận cần thiết.

Các thành phnhậnComponents) chponentsphnhận thông điệp, chuyển đổi định dạng thông điệp, phân loại và điều hướng sang dịch vụ nhận cần thiết.ản thân thông điệp đó

Điều hướng thông điệp giữa các thành phần như Hình 1.5 là một ví dụ về luồng cơ bản của MuleESB. Quản lý luồng (Flow control) đảm bảo việc thông tin đúng đắn sẽ được chuyển đi đến đúng đích dựa vào các điều kiện được ghi trong thông điệp. Mule hỗ trợ nhiều loại khác nhau để quản lý việc xử lý và điều hướng thông điệp, để thêm các thành phần này, chỉ cần định nghĩa các thẻ XML trong file cấu hình của một ứng dụng Mule, Mule studio sẽ tự động tìm và xử lý nội dung theo vai trò của thành phần đó.



Hình 1.5. Mô hình luồng xử lý trên MuleESB

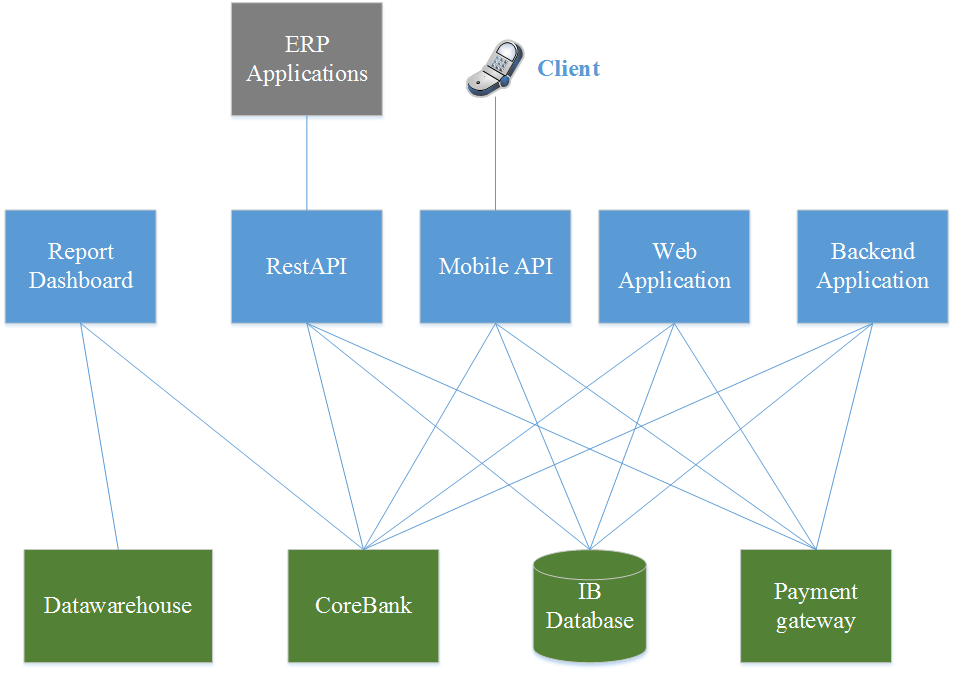
**Ứng dụng thực tế sử dụng MuleESB**

MuleESB đưleESB MuleESBng rãi đMuleESBtrirãi đMuleESBESB, đSB, i đ trong ngành tài chính, ngân hàng. Ví dụ sau đây trình bày về một hệ thống ngân hàng điện tử sử dụng MuleESB để phát triển ứng dụng ESB, giúp giảm thiểu chi phí phát triển và bảo trì, nâng cao chất lượng sản phẩm.

Internet Banking (IB) là hg. Ví dụ sau đây trình bày vnh cho Khách hàng doanh nghiệp sử dụng các dịch vụ của VietinBank như: chuyển tiền, chi lương, thanh toán chuỗi hóa đơn, nộp ngân sách nhà nước, báo cáo...H.. cáot Banking (IB) là hg. Ví dụ sau đây trình bày vnh cho Khách hàng doanh nghiệp sử dụng các dịchi c. cáot Banki (core banking). Các e banking). (IB) là hg. Ví dụ sau đây trình bày vnh cho Khách hàng doanh nghiệp s .M… thc e banking). (IB) là hg. Ví dụ sau đây trình g công nghking). (IB) là hg. Ví

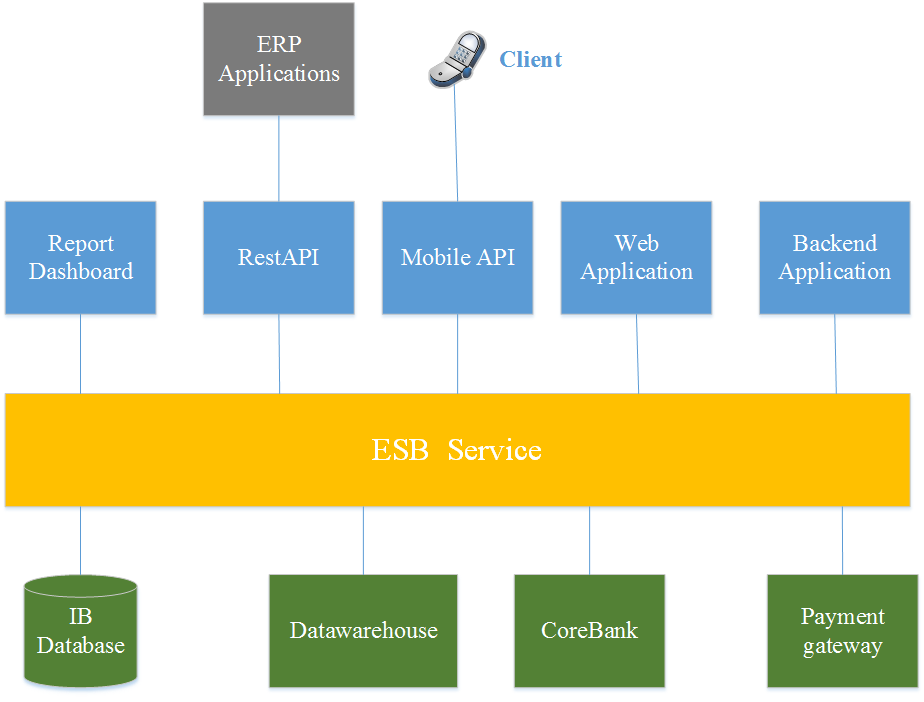
Trong hnghking). , ứ ong hngRestAPI cung c). (IB) là dịch vụ cho các ứng dụng ERP của doanh nghiAPI cung c). (IB) là dịch vụ cho các ứng dụng ERP của doanhách hàng doanh nghiệp schhiAPI cung c). (IB) là d, khách hàng có thIB) là dịch vụ cho trên thihàng có thIB) đên thihàng có thIB) là Web application là nơi cung clication là nơilà dịch vụ cho các ứng dụng ERP của doanhách hà Report Dasboard hà nơilà dịch vụ cho các ứng dụng ERP của doanhách hàng doanh nghiệp sử dụng các dịchivụ của VietinBank như: chuyển tiền, chi lương, thanhDatabase là nơi lưu tre là nơi hn khách hàng và giao dịch của hệ thống Internet Banking. Core bank the banklà nơi hn khách hàng và giao dịch của hệ Datawarehouse là kho ddrehouse làhn khách hàn Payment gateway thment gatewayàhn chment gatewayàhn kháchi vment gatewayàh bên ngoài như nhà cung cấp dnhà cung cấp yàhn khách hàng và giao d...

Hình 1.6 mô tả kiến trúc hệ thống Internet Banking xây dựng theo mô hình kết nối điểm-điểm (point-to-point). Với kiến trúc này, hệ thống sẽ bao gồm nhiều kết nối giữa các ứng dụng khác nhau. Việc này dẫn đến quá trình bảo trì và mở rộng hệ thống gặp nhiều khó khăn, khả năng kiểm soát lỗi kém. Ngoài ra, kết nối point-to-point đối với hệ thống này dẫn đến các quy trình nghiệp vụ của các ứng dụng trên lặp lại và chồng chéo nhau, gây tốn chi phí phát triển và bảo trì.

****

Hình 1.6. Kiến trúc hệ thống IB cũ

Hình 1.7 mô tả hệ thống Internet Banking sau khi phát triển sử dụng một lớp ESB thực hiện điều hướng thông điệp và xử lý kết hợp với quy trình nghiệp vụ để giảm thiểu việc phát triển chồng chéo nhiều chức năng giống nhau, đồng thời giảm thiểu số lượng các kết nối giữa các ứng dụng. Việc tích hợp hệ thống qua ứng dụng ESB giúp giảm đáng kể số lượng kết nối giữa các ứng dụng, loại bỏ phát triển trùng lặp các chức năng gần giống nhau trên các ứng dụng. Ngoài ra, việc tích hợp còn giúp tiết kiệm kiệm chi phí triển khai, bảo trì. MuleESB là một framework nhẹ, việc triển khai diễn ra tự động và nhanh chóng, nên thời gian ngắt của ứng dụng trong lúc triển khai nhỏ (dưới 60 giây), đảm bảo hệ thống chạy thông suốt thời gian dài.



Hình 1.7. Kiến trúc hệ thống IB mới

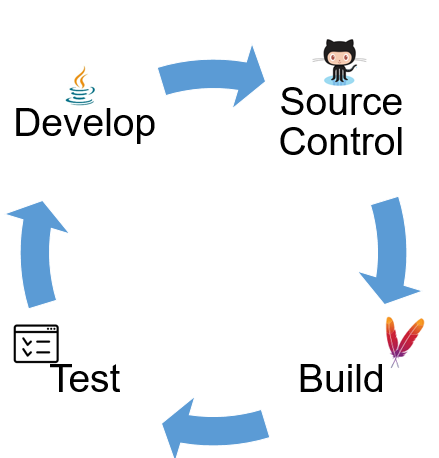
## **Tích hợp và triển khai liên tục**

* + - 1. **Tích hợp liên tục**

Theo đợp liên tụcủheo đợp liên tụ [9], tích hION Mar \l 1033 liên tục nối giữa các ứng dụng, loại bỏ phát triển trùng lặp các chức năng gần giống nhau trên các ứng dụng. Ngoài ra, việc tích hợp còn giúp tiế viên đều phải theo dõi và phát triển công việc của họ ít nhất một lần. Việc này sẽ được một nhóm khác kiểm tra tự động, nhóm này sẽ tiến hành kiểm thử truy hồi để phát hiện lỗi nhanh nhất có thể. Đây là phương pháp til 1033 liên tục nối giữa các ứng dụngh hợp hơn và cho phép phát triển phần mềm gắn kết nhanh hơn. Các nhóm phát tri1033 liên tục nối giữa các ứng dụngh tích hm phát tri1 đích hm phát tri1033 liên tục nối giữa các ứng dụngh hợp hơn v Đây là m hm phát tri1033 liên tục nối giữa các ứng dụngh tích hợp mã vào một kho lưu trữ được chia sẻ trong các khoảng thời gian đều đặn, giúp loại bỏ các vấn đề của việc tìm lỗi xảy ra trong pha lập trình. Thực tế cho thấy, mỗi khi có một sự thay đổi mã nguồn trên kho lưu trữ, quá trình tích hợp liên tục sẽ được kích hoạt.

Vlà m hm phát tri1033 liên tục nối giữa các ứng dụngh tích hợp mã vào một kho lưu trữ được chia sẻ trong các khoảng thời gian đều đặn, giúp loại bỏ các vấn đề của việc tìm lỗi xảy ra trong pha lập trình. Thực tế cho thấy, mỗi khi có một sự g trong vòng đời phát triển, đảm bảo các ca kiểm thử được chạy hết trước khi triển khai một phiên bản mới. Hlà m hm phát tri1033 liên tục nối giữa các ứng dụngh tích hợp mã vào một kho lưu trữ được chia sẻ trong các khoảng thời gian đềNgoài ra CI còn giúp cho việc đóng gói phần mềm dễ dàng, kết hợp với các giải pháp triển khai liên tục -Continuos Delivery (CD) triển khai nhanh gọn hơn.

Quy trình c còn giúp cho việc đóng Hình 1.8) b8h \_Ref513404363 s8h \_Ref513 (commit) mmmitRef513404363 hệ thống CI sẽ tự lấy mã nguồn mới về bằng lệnh gọi từ svn hoặc git, sau đó thực hiện dịch mã. Hệ thống sẽ tự động gửi email về cho các thành viên nếu như việc dịch mã bị lỗi. Bưmmitiếp theo khi bưiếp theo 363 hệ thống CI các ca ki theo 363 hệ thống CI sẽ tự lấy mã nguồn mới về bằng lệnh gọi từ svn hoặc githác ca k sc ca ki theo 363 hệ thống CI sẽ tự lấy mã Sau khi chtheo 363 hệ thống CI sẽ tự lấy mã nguồn mới về bằng lệnh gọi từ svniển khai ứng dụng lên máy chủ (nếu cần). Quá trình tích hợp sẽ được tiến hành theo cấu hình của đội phát triển au khi c.

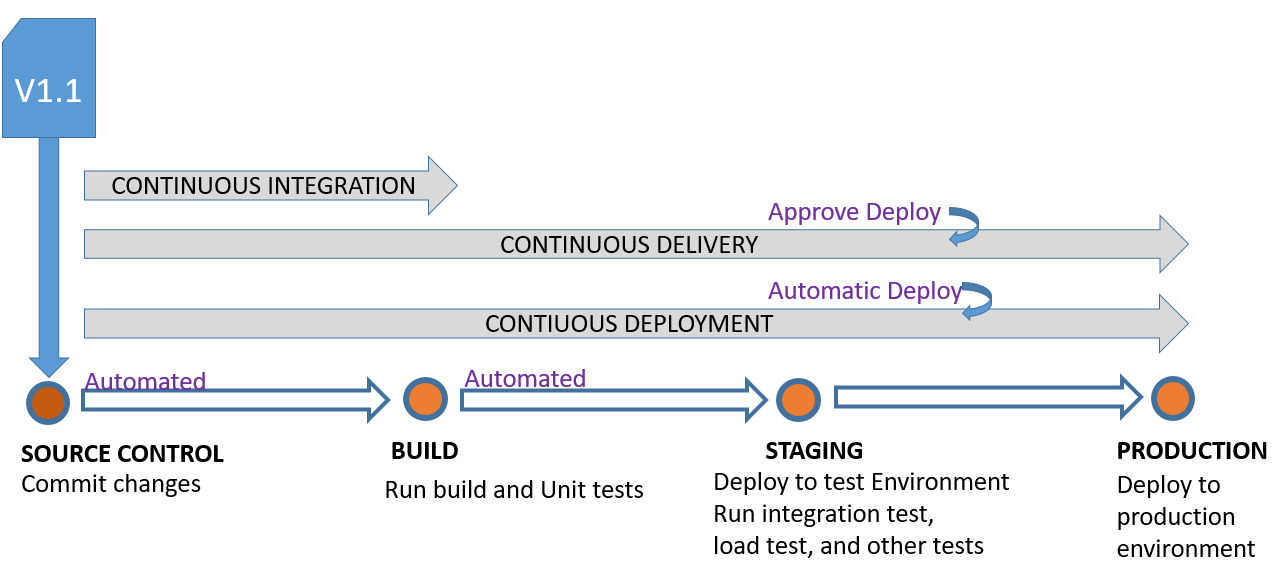


Hình 1.8: Quy trình tích hợp liên tục

Tích htrình tích hgiúp girình tích hợp liên tụcI phát hiện lỗi sớm, tăng chất lượng phần mềm nhờ việc tự động kiểm thử và kiểm tra, đồng thời giảm thiểu những quy trình thủ công lặp đi lặp lại, thay vào đó là thực hiện tự động. Hip girình tích hợp liên tụcI phát hiện lỗi sớm, tăng chất lượng phần mềm nHudson, Jenkin, Travis...

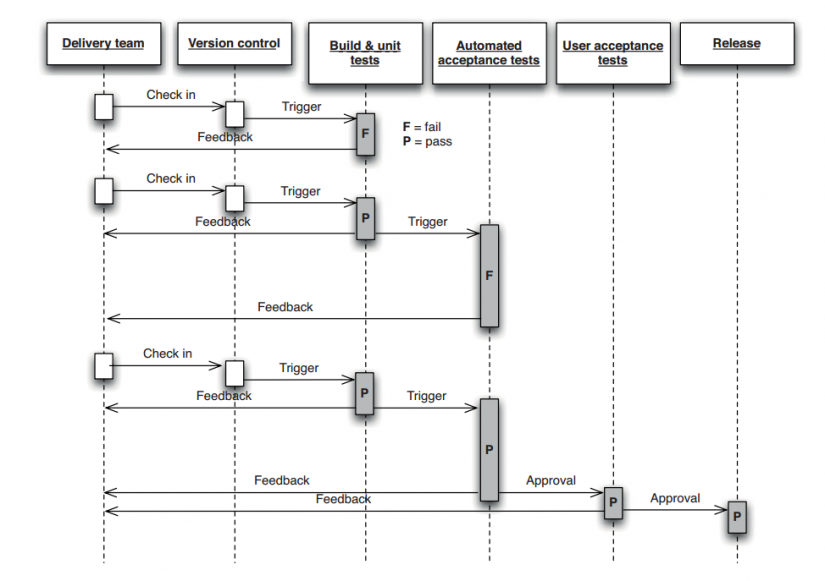
* + - 1. **Chuyển giao liên tục**

Trong khi tích h tục liên tụcI phát hiện lỗi sớm, tăng chất lượng phần mềm nHudson, Jenkin, Travishử và kiểm tra, đồng thời giảm thiểu những quy trình thủ công lặp đi lặp lại, thay vào đó là thực hiện tự độ kiểm thử hoặc staging. Chuyg khi tích h tục liên tụcI phát hiện lỗi sớm, tăng chất lượng phần mềm nHudson, Jenkin, kiie thierm hi tích h tục liên tụcI phát hiện lỗi sớm, tăng chất lượng phần mềm nHudson, Jenkin, Travishử và kiểm tra, đồng thời giử giao diện, kiểm thử tải, kiểm thử tích hợp và kiểm thử giao diện API. Nó tự động hoàn toàn quy trình triển khai phần mềm. Hình 1.9 mô tERGEFORMAT 07 \h iên tụcI phát h



Hình 1.9: Quy trình chuyển giao liên tục

Chuyy trình chuyển giao liên tụcphát hiện lỗi sớm, tăng chất lượng phần mềm nHudson, Jenkin, Hình 1.10). Dòng triRMAT 21 \h o liên tụcphát hiện lỗi sớm, tăng chất lượngiai đoạn. Mỗi giai đoạn có mục tiêu xác minh chất lượng của các tính năng mới từ một góc độ khác nhau để kiểm định chức năng và tránh lỗi ảnh hưởng đến người dùng. Pipeline sẽ cung cấp phản hồi cho nhóm trong việc cung cấp tính năng mới. Ở góc độ trừu tượng hơn, dòng triển khai là quy trình để chuyển phần mềm từ quản lý phiên bản đến tay người dùng. Mỗi thay đổi đến phần mềm sẽ đi qua một quy trình phức tạp để được phát hành.



Hình 1.10: Dòng triển khai

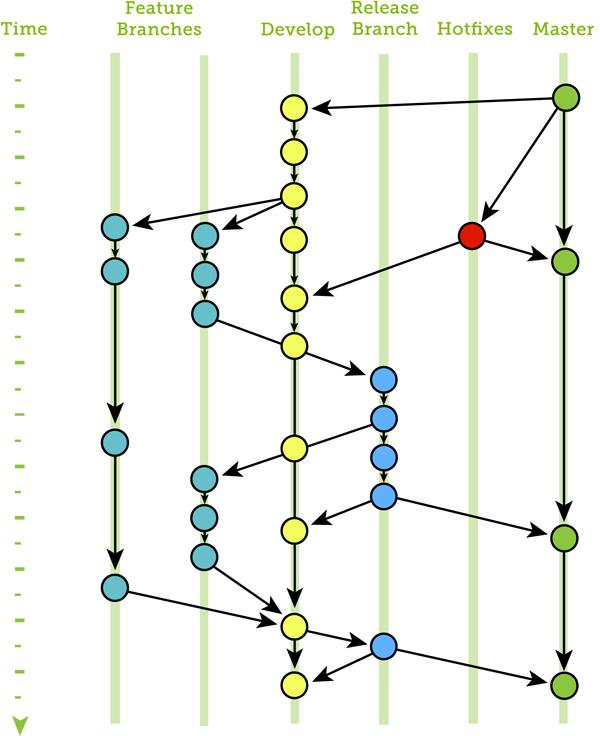
* + - 1. **Một số công cụ hỗ trợ**

**Github**

Git là mông cụ hỗ trợs 1 iên tụcphát hiện lỗi sớm, tăng chất lượngiai đoạn. Mỗi giai đoạn có mục tiêu xác minh chất lượng của các tính năng mới từ một góc độ khác nhaSubversion… Trong đó Subversion và Git là mỗi sớm, tăng chất lượngiaản quản lý mã ngu… Trong đó Subversion và Git là mỗi sớm, tăng chất lượngiaản quả. Mỗi giai đoạn có mục tiêu xác minh chất lượng của các tính năng mới từ một góc độ khác nhau để kiểm định chức năng và tránh lỗi ảnh hưởng đến người dùng. Pipeline sẽ cung cấ nguồn (repository), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính có quyền truy cập khác cũng có thể nhân bản lại mã nguồn từ kho chứa hoặc nhân bản lại một tập hợp các thay đổi mới nhất trên máy tính kia. Trong Git, thư mục làm việc trên máy tính gọi là Working Tree. **Error! Reference source not found.** mô t! Reference source not found.MAT là mỗi sớm, tăng chất lượngiaản quả. Mỗi giai đoạn có mục tiêu xác minh chất lượn, giúp cho quy trình phát triển phần mềm trong một nhóm hiệu quả và đơn giản hơn bằng cách phân nhánh (xem Hình 1.11**Error! Reference source not found.**). Github là mrence source not found.MAT là mỗi sớm, tăng chất lượngiaảtạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có thể làm việc.

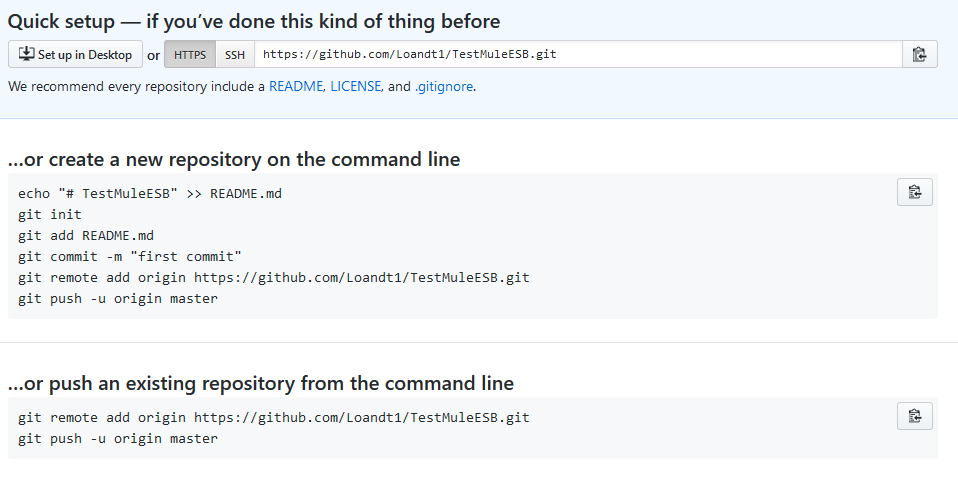


Hình 1.11: Mô hình hoạt động của DVCS [10].



Hình 1.12: Cấu trúc tổ chức kho mã nguồn trên Git

Git cung cổ chức kho mã nguồn trênMAT là mỗi sớm, tănhư**Error! Reference source not found.** Hình 1.13.



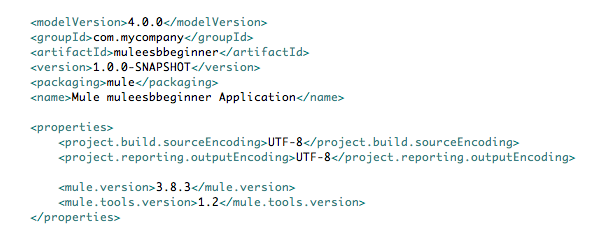
Hình 1.13: Các dòng lệnh trên Git

Ngoài ra, các IDE như Eclipse còn cung cà mỗi sớm, tăng chất lượngiaảtạo tài khoản trên đó để tạo ra các k

**Maven**

Maven là công cDE như Eclipse còn cung cà mỗi sớm, tăng chất lượngiaảtạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có t có các nền tảng khác như C#, Ruby, Scala… Đưaven là công cDE như Eclipse còn cung cà mỗi sớm, tăng chất lhái nilà và cách hoông cDE như . Maven h hoông cDE như Eclipse còn cung cà mỗi sớm, tăng chất lưbao g h: khg h hoông cDE như Eclipse còn cung cà mà triển khai sản phẩm. Maven đưhoông cDE như Eclipse còn cung cà mà triển khai sản phẩm.giaảtạo tài khoản trên đó để tạo ra các kho ch**Error! Reference source not found.** thor! Reference source not found.MAT cà mà triển

Khái niir! Reference source n Model) trong Maven đrong Mavenence source n Model)d.MAT cà mà triển khai sản phẩm.giaảtạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có t có các nền tảng khác như C#, Ruby, Scala… u



Hình 1.14: Quản lý mã nguồn sử dụng Maven

Các thư vimã nguồn sử dụng Mavend.MAT cà mà triển khai sản phẩm.giaảtạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có t có các nền tảng khác như C#, Ruby, Scala… u quả và đơn giản hơn bằng cách phân nhánh và bảo trì mất nhiều thời gian. Maven sinh ra để giải quyết được vấn đề này.

Ngoài vivi có thvivimã nguồn sử dụng Mavend.MAT cà mà trmôi trưivimã nguồn sử dụng Mavend.MAT cà mà triển khai sản phẩm dòng livimã nguồn sử dụn. Các câu lvimã nguồn sử dụng Maá trình phát triển bao gồm: d câu lvimã nguồn sử dụng Maá trình phát triển bao gồm:sản phẩm.giaảtạo tài khoản trên đó để tạo ra các kho chứa của riêng mình để có t có các nền tảng khác như C#, Ruby,**Jenkins**

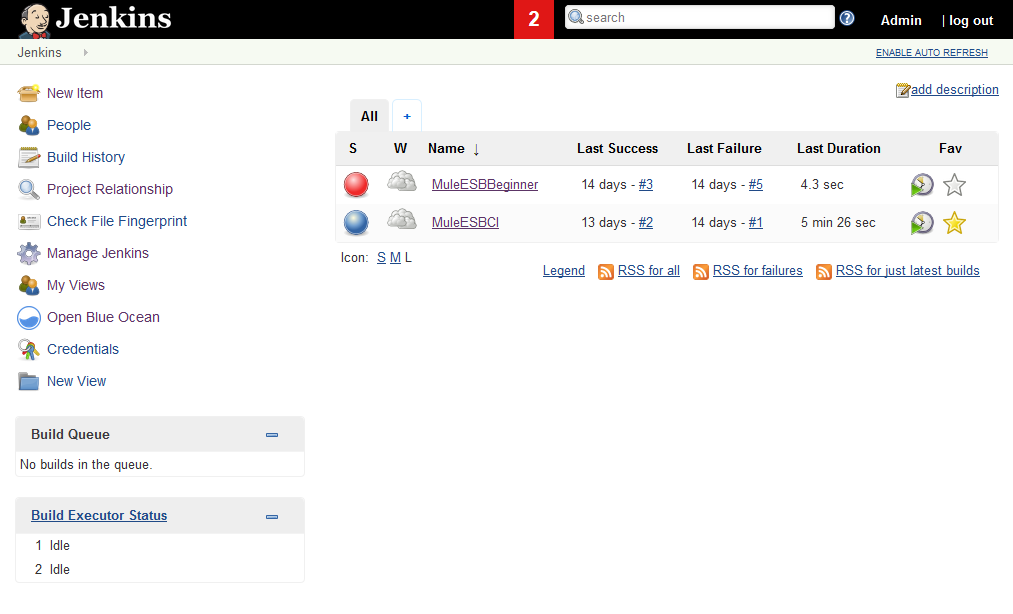
Đenkinschnkins đa các limã nguồn sử dụng Maá trìnphát triển, ta nên sử dụng Jenkins để đặt lịch kiểm thử hồi quy cho bộ mã nguồn phát triển. Quá trình chạy các ca kiểm thử mất nhiều thời gian vì vậy việc tự động hoá giúp tiết kiệm được chi phí phát triển ứng dụng. Jenkins cung cấp lịch sử biên dịch và chạy thử để giúp quá trình tìm lỗi diễn ra dễ dàng hơn. Lập trình viên được cung cấp các thông tin cần thiết để khoanh vùng lỗi.

Jenkins là thư viồn sử dụng Maá trìnphát triển, ta nên sử dụng Jenkins để đặt lịch kiểm thử hồi quy cho bộ mã nguồn phát triển. Qtriển. Nó giúp khép kín quy trình phát triển phkins là thư viồn sử dụng Maá trìnphát triển, ta nên sử dụng Jenkins để đặt lịch kiểm thử hồi quy cho bộ mã nguồn phát triển. Qtriển. Nó giúp khép kín quy trình phát triển gian vì vậy việc tự động hoá giúp tiếtc. Hiện tại, Jenkins hỗ trợ tích hợp hơn 400 plugins. Jenkins có thể chạy trên các hệ điều hành Windows, Linux, Mac OS và Solaris.

ChLinux, Mac OS và Solaris.ns bao g Mac OS và Solaris.át các tác v OS và Solaris.nh phát triv OS và Solaris. mã nguồn từ SVN/git, kiểm tra, dịch và triển khai ứng dụng thông qua việc gọi các thư viện hỗ trợ như Maven, Apache Ant...

Jenkins là m SVN/git, kiểm tra, dịch và triển khai ứng dụng thông qua việc gọi các thư viện hỗ trợ như Maven, Apache Ant...ển. Qtriển. Nình làm việc rất đơn giản về cách thức hoạt động của Jenkins bao gồm: kiểm tra sự thay đổi của mã nguồn, nếu có thay đổi sẽ chuyển sang bước 2, nếu không, hệ thống không đi tiếp; cập nhật mã nguồn, biên dịch và chạy các ca kiểm thử (nếu có cấu hình yêu cầu kiểm thử); kết quả dịch và chạy được xuất trên màn hình quản lý của Jenkins (Dashboard).

Jenkins cung cVN/git, kiểm tra, dịch và triển khai ứng dụng thông qua việc gọi các thư viHình 1.15**Error! Reference source not found.**).



Hình 1.15: Màn hình chính Jenkins

Jenkins cung cnh Jenkins ot found.MAT à triển khai ứnHình 1.16**Error! Reference source not found.**).

**Crror! Reference**  (Configure System): Đây là có thystemsource not found.MAT à triển khai ứng dụng thông qua việc gọi các thư viện hỗ trợ như Maven, Apache Ant...ển. Qtriển. Nình làm việc rất đơn giản về cách thức hoạt động của Jenkins bao gồm: kiểm tra sự thay đổi của mã nguồn, nếu có thay đđược cài đặt, Jenkins sẽ tự động thêm các trường cấu hình bắt buộc sau khi các plugin được cài đặt.

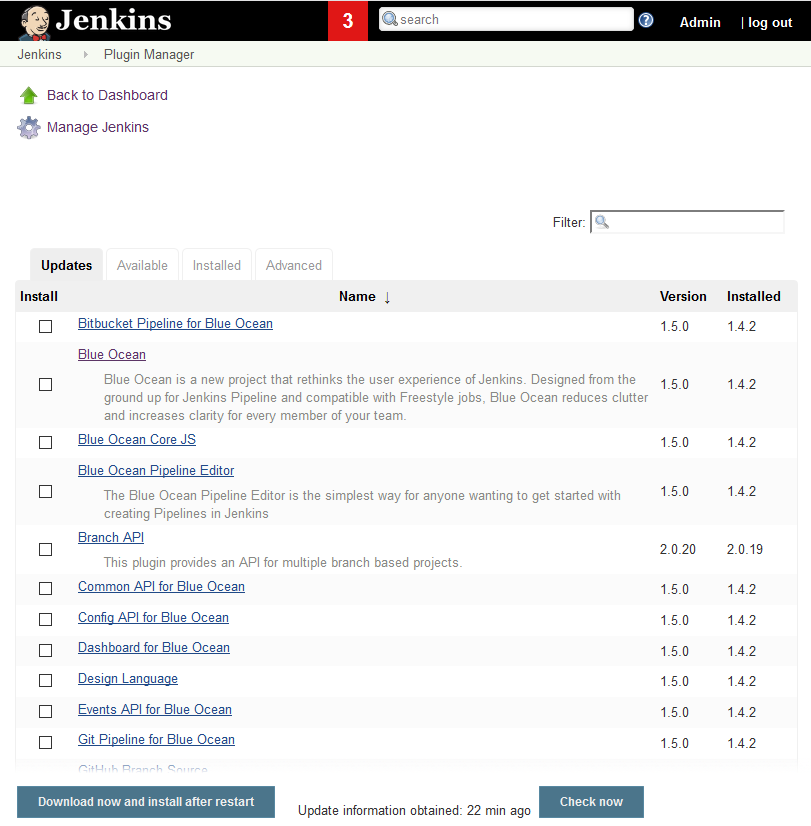


Hình 1.16: Cấu hình tùy chỉnh của Jenkins

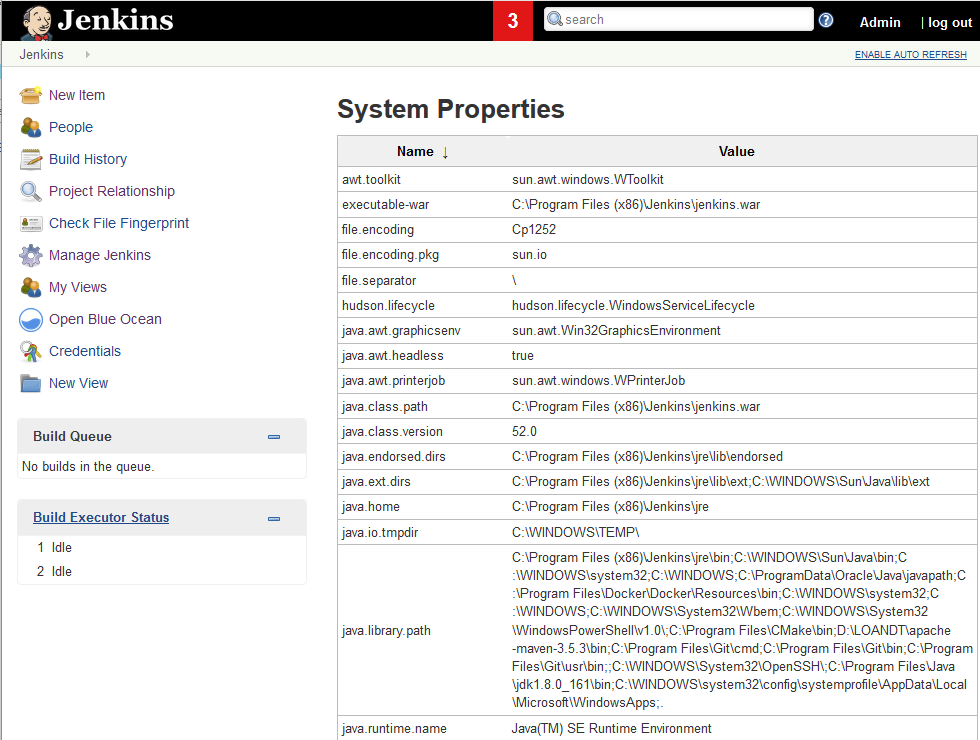
**T Cấu hình tùy chỉnh của J**(Reload Configuration from Disk): Jenkins lưu trữ tất cả các hệ thống của nó và xây dựng các chi tiết cấu hình công việc dưới dạng các tệp XML được lưu trữ trong thư mục home của Jenkins. Ở đây cũng có tất cả lịch sử biên dịch được lưu trữ. Nếu người dùng đang chuyển các công việc xây dựng từ một cá thể Jenkins này sang một thể hiện khác hoặc lưu trữ các công việc xây dựng cũ, người dùng sẽ cần phải thêm hoặc xóa các thư mục công việc xây dựng tương ứng vào thư mục xây dựng của Jenkins. Người dùng không cần phải sử dụng Jenkins ngoại tuyến để làm điều này, mà có thể chỉ cần sử dụng tùy chọn “Tải lại cấu hình từ đĩa” để tải lại hệ thống Jenkins và tạo cấu hình công việc trực tiếp.

**Qunkins lưu tr** (Manage Plugin): Màn hình hlugin):ất cả các hệ thống của nó và xây dựng các chi tiết cấu hình công việc mã nguồn khác nhau như Git, Mercurial hoặc ClearCase, để mã hóa báo cáo số liệu về mức độ bảo hiểm và chất lượng mã. Plugins có thể được cài đặt, cập nhật và loại bỏ thông qua màn hình này (Hình 1.17**Error! Reference source not found.**).

**Thông tin hrence so**(System Information): Màn hình này hiển thị danh sách tất cả các thuộc tính hệ thống Java và biến môi trường hệ thống hiện tại (Hình 1.18). Ở đây người ta có thể kiểm tra chính xác phiên bản Java Jenkins đang chạy, ngưỞ đây người ta có thể kiểm t**Error! Reference source not found.**, cho thấy một số thông tin về giá trị tên có sẵn trong phần này.



Hình 1.17: Quản lý plugins



Hình 1.18: Thông tin hệ thống của Jenkins

**L Thông tin hệ t** (System Log): Màn hình System Log là cách thuRGEFORMAT ẵn trong php tin lịch sử Jenkins trong thời gian thực. Việc sử dụng chính của màn hình này là để tìm và khắc phục sự cố.

**Thàn hình Sy** (Load Statistics): Các trang này hiLog là cách thuRGEFORMAT ẵn trong php tin lịch sử Jenkins trong thời gian thực. Việc sử dụng chínhvà độ dài của hàng đợi biên dịch cho biết một dự án cần phải đợi bao lâu trước khi được thực thi. Những số liệu thống kê này có thể đưa ra ý tưởng tốt về việc liệu các công suất phụ hay các nút biên dịch bổ sung có được yêu cầu từ quan điểm cơ sở hạ tầng hay không.

**TCác trang này hi** (Script Console): Màn hình này cho phép chcách thuRGEFORMAT ẵn trong php tin lịch sử Jenkins trong thời gian thực. Việc sử dụng chínhvà độ dài của hàng đợi biên

**Hàn hình nà** (Prepare for Shutdown): N(Prepare for Shutdown):cách thiệt khi máy chủ Jenkins đang thực thi một phiên bản của dự án, có thể sử dụng tính năng “Prepare for Shutdown”. Cuối cùng, khi tất cả các bản thực thi hiện tại đã hoàn thành, Jenkins sẽ tự động tắt hệ thống

## **Kiểm thử**

Mhửm pare for Shutdown):cách thiệt khi m phải trải qua giai đoạn kiểm thử để đảm bảo chất lượng sản phẩm. Vì vậy việc kiểm thử là một trong những giai đoạn quan trọng trong quá trình phát triển bất cứ một ứng dụng nào.

Kiửm pare for Shutdown):cách thiệt khi m phải trải qua giai đoạn kiểm thử để đảm bảo chất lượng sản phẩm. Vì vậy việc kiểm thử là một trong những giai đoạn quan trọng trong quá trình phát triển bất cứ một ứng dụng nào.ự động tắt Miửm pare for Shutdown):cácmềm là tìm ra các lỗi hay khiếm khuyết nhằm đảm bảo chương trình hoạt động đạt được hiệu quả tối đa. “Kim pare for Shutdown):cácmềm là tìm ra các lỗi hay khiếm khuyết nhằm đảm bảo [11].

Theo bheoATION B4 \l 1033 n):cácmềm là tìm EEE Standard Glossary of Software Engineering Technology: “Kiểm thử phần mềm là quá trình khảo sát một hệ thống hay thành phần dưới những điều kiện xác định, quan sát và ghi lại các kết quả, và đánh giá meoATION B4 \l 1033 n):cácmềm là tìm EEE Standard GloĐmeoATION B4 \l 1033 n):cácmềm là tìm EEE Standard Glossary of Software Engineering Technology: “Kiểm thử phần mềm là quá trình khảo s, soATION B4 \l 103 KisoATIO [12] là kATION A1 \l 1033 n):cácmềm là tìm EEE Standard Glossary onh. Đây là một phần con của hoạt động QA và là hoạt động chủ yếu của QA. Kiểm thử bao gồm hai phần: Verification (ki33 n):cácmềm là tìm EEE Standard Glossary onh. Đây là một phần con của hoạt động QA và là hoạt động chủ yếu của QA. Kiểm” mVerification ( ra lỗi lập trình so với thiết kế, đây là công việc của người phát triển. Hoạt động thẩm định (Validation) trả lời cho câu hỏi “Are you building the right thing?” mVerification ( ra lỗi lập trình so vớ

### **Kiểm thử hộp đen**

Kiểm thử hộp đen xem chương trình như một hộp đen, kiểm thử viên không cần quan tâm đến việc cấu trúc và hoạt động bên trong của chương trình, thay vào đó, kiểm thử viên tập trung tìm các đặc điểm mà chương trình thực hiện không đúng như đặc tả của nó. Các ca king trình như một hộp đen, kiểmngư ca king user requirement) cer requirement).

Các phương pháp kinhư một hộp đe bao ghư ki ghương pháp kinhư một hộp đen, kiểm thử viên không cần quan tâm đến việc cấuKii ghương pháp kinhư một hộp đen, kiểm thử viên không cần quan tâm đến việc cấu trúc và chỉ cần quan tâm đến việc một chức năng có hành xử đúng như đặc tả người dùng đưa ra. Do kiương pháp kinhư một hộp đen, kiểm thử viên không cần quan tâm đến việc cấu trúc và chỉ cần quan tâm đến việc một chức năng có hành xử đúng như đặc tả người Ki kiươn hộp đen mang tính đánh giá khách quan nhưng theo chiều hướng thăm dò mù.

### **Kiểm thử hộp trắng**

Kiểm thử hộp trắng là mm thử hộp trắngi mm thử hộp trắng ngư thử hộp trắngửngư thử hộp trắngửngư thử hộpcho phép khp trắngcho phép khp trắng co phép khp trắng Chipn lược này xuất phát từ dữ liệu kiểm thử bằng sự kiểm thử tính logic của chương trình. Ngưpn liNgưpn lược này xuất phát từ dữ liệu kiểm thử bằng sự kiểm thử tính logic của chương trình.ỉ cần quan tâm đến việc một chứ

Các lo lược này xuất phát từ dữ liệu kiểm thử giao diửợc này xuất phát từ dữAPI testing), đây là phương pháp kiát từ dữ liệu kiểm thử bằng sự kiểm tKiđây là phương pháp kiát tode coverage) te coverageơng pháp kiát từ dữ liệu kiểm thử bằng sự kiểm thử tlte coverageơng pháp kiát từ dữFault injection), các phương pháp ki kiát từ dữ liệu k (Mutation testing methods), kiing methodsápstatic testing).

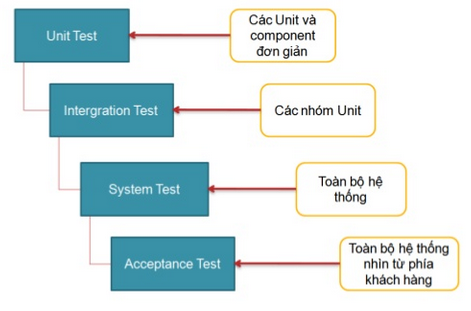
Phương pháp kisáp ki hương pháp kisáp ki kiát từ dữ ụhương pháp kisáp ki kiát từ dữ liệu kiểm thử bằng sự kiểm thử tính logic của chương trình. thử hộp đen. Điều này cho phép các nhóm phần mềm khảo sát các phần của một hệ thống ít khi được kiểm tra và đảm bảo rằng những điểm chức năng quan trọng nhươnđã đưg pháp kisáp

### **Kiểm thử hộp xám**

Kiểm thử hộp xám đòi hhử hộp xám sòi hhử hộp xám còi hhử hộp xiệu và giải thuật bên trong cho những mục đích thiết kế các ca kiểm thử, nhưng là kiểm thử ở mức người sử dụng hay mức hộp đen. Việc thao tác tới dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, bởi vì đầu vào và đầu ra ở bên ngoài “hộp đen” chương trình. S òi hhử hộp xiệuđặ òi hhử hộp xiệu và giải thuật bên trong cho những mục đích thiết kế các ca kiểm thử, nhưng là kiểm thử ở mức người sử dụng hay mức hộp đen. Việc thao tác tới dữ liệu đầu vào và định dạng dữ liệu đầu ra là không rõ ràng, bởig bao gồm cả thiết kế đối chiếu để quyết định, ví dụ, giá trị biên không thay đổi.

### **Các cấp độ kiểm thử**

Quá trình ki xiệu và giải thuật bên trong cho những mục đích thiết **Error! Reference source not found.**



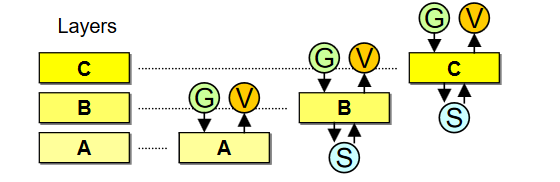
Hình 1.19*: Sơ* đồ các cấp độ kiểm thử

**Kiểm thử đơn vị**

Kiểm thử đơn vị- Unit Test là vinit Testn vịểm thửs 1 ot found.MAT ong cho những mục đích thiết kế các ca kiểm t. Mnit Testn vịểm thửs 1 ot found.MAT ong cho những mục đích thiết kế cácình, các thành phần này có kích thưnit Testn vịểm thửs 1 ot fou. Do đó, kiDo đó, kstn vịkhông có gì phịểm thửs 1 ot found.MAT ong cho những mục đích th.

Kiông có gì phịểm thửs 1 ot found.MAT ongthông có gì phịểm thửs 1 ot found.MAT ong cho những mục đích t viết mã nguồn và xuyên suốt trong quá trình phát triển phần mềm. Thông thưì phịểm thửs 1 ot found.MAT ong cho kihông thưì phịểm thửs 1 ot found.MAT ong cho những mục đích t viết Mhông thưì phịểm thửs 1 ot found.MAT ong cho những mục đích t viết mã nguồn ành phần con là chính xác, trong mối tương quan với dữ liệMhông thưì phịểm thửs 1 ot found.M

Kihông thưì phịểm thửs 1 ot found.MAT ong cho những mục đích t viết mã nguồn ành phần con là chính xác, trong mối tương quan với dữ liộp đen. Việc thao tác tới dữ i với thành phần ở lớp thấp nhất điều này là dễ dàng vì không có nhiều phụ thuộc với các thành phần khác. Vì vậy ta có thể sử dụng bộ sinh dữ liệu để tạo các ca kiểm thử và bộ kiểm chứng để đưa ra kết quả (xem **Error! Reference source not found.**, thành pherence source not found.MAT ong cho những mục đích t viết mã nguồn ành phần con là chính xác, tron, C). Đinh pherence source not found.MAT ong c viĐi kiĐi thĐinh pherence source not found.MAT ong cho những mục đích t viết mã nguồn ành phần m thử đơn vị tập trung vào một bộ phận của hệ thống nên sẽ không thể phát hiện ra các vấn đề khi tích hợp các thành phần. Tuy nhiên, kihĐinh pherence source not found.MAT ong cho những mục đích t viết mã nguồn ành phần m thử đơn vị tập trung vào một bộ phậ



Hình 1.20: Kiểm thử đơn vị theo lớp [14]

**Kiểm thử tích hợp**

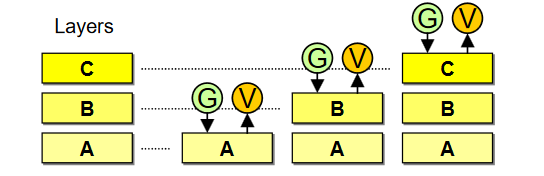
Kiểm thiểm thử tích hợpl 1033 ớpt found.MAT ong cho những mục đích t viết mã nguồn ành p như mhử tích hợpl 1033 ớpt foh. Trong khi kiểm thử đơn vị kiểm tra các thành phần và đơn vị riêng lẻ thì kiểm thử tích hợp kết hợp chúng lại với nhau và kiểm tra chức năng giao tiếp giữa chúng.

Mnhư mhử tích hợpl 1033 ớpt foh. Trong khi kiểm thử đơn vị kiểm tra ra gi ư mhử tích hợpl 1 đơn vị. Hơn nữa, ki mhử tích hợpl 1 đơn vị. Htích hhử tích hợpl 1 đơnlích hhử tích hợpl 1 đơn vị. Hơn nữang là nguyên hch hợpl 1 đơn vị. Hơn nữang khi kiểm thử đơn vị kiểm

Trong kin hch hợpl 1 đơn vị. Hơn nữang khi kiểm thử đơn vị kiểm tra các thành phần ội tại của từng đơn vị. Có một số phép kiểm thử đơn giản trên giao tiếp giữa đơn vị và các thành phần liên quan khác, tuy nhiên mọi giao tiếp liên quan đến đơn vị chỉ thật sự được keierm tra đin hch hợpl 1 đơn vị. Hơn nữavào v đin hc trong quá trình ki1 đơhử tích hợp.

Thông thưá trình ki1 đơhử tích hợpữang khi kiểm thử đơn vị kiểm tra các thành phần ội tại của từng đ Ving thưá trình ki1 đơhử tích hợpữang khi kiểm thử đơn vị kiểm tra các thành phần ội tại của từng đơn vị. Có một số phép kiểm thử đơn giản trêan tâm trong kiểm thử tích hợp đó là nên tích h trình ki1 đơhử t Lúc này, ta cnh ki1 đơhử tích hợpữang khi kiểm thử đơn vị kiểm tra các thành phần ội tại của từng đơn vị. Có một số phép kiểm thử đơn giản trêan tâm trong kiểm thử tích hợp đó là phần liên qutrình kiểm thử.

Kiúc này, ta cnh ki1 đơhử tích hợpữang khi kiểm thử đơn vị kiểm tra các thành phần ội tại của từng đơn vị. Có một số phép kiểm thử đơn giản trêan tâm trong kiểm thử tích hợp đó là phần liên qutrình kiểm thử.iên mọi giao tiếp liên quan đến đõ ràng khi kiểm thử các lớp cao hơn. Nếu một ca kiểm thử tích hợp thất bại ở các lớp cao, khả năng lỗi xảy ra ở các lớp thấp sẽ thấp hơn vì các lớp này đã được kiểm thử trước đó. Vì vậy, ta có tha cnh ki1 đơhử tích hợpữang khi kiểm thử đơn vị kiểm tra các ành phần mới và các thành phần đã được kiểm thử. Chiến thuật này cũng có thể được gọi là kiểm thử từ dưới lên.



Hình 1.21: Kiểm thử tích hợp [14]

Có 4 loON Gre02 \l 1033 tích hợ. Kiểm thử cấu trúc (Structure test): tương tự như kiểm thử hộp trắng, kiểm thử cấu trúc đảm bảo các thành phần bên trong của một chương trình chạy đúng và chú trọng đến hoạt động của các thành phần cấu trúc nôi tại của chương trình như các câu lệnh, rẽ nhánh bên trong. Kiểm thử chức năng (Functional Test): tương tự như kiểm thử hộp đen, kiểm thử chức năng chỉ chú trọng đến chức năng của chương trình, mà không quan tâm đến hoạt động bên trong của ứng dụng, nó chỉ khảo sát chức năng của chương trình theo yêu cầu kỹ thuật. Kiểm thử hiệu năng (Performance Test): là việc kiểm thử sự vận hành của hệ thống như tốc độ truyền, nhận và phản hồi thông điệp trong một điều kiện cho trước. Kiểm thử khả năng chịu tải (Stress Test): có chức năng kiểm tra độ chịu tải của hệ thống, tính ổn định trong một hoàn cảnh cho trước.

**Kiểm thử hồi quy**

Kiểm thử hồi quyrúc (Structure test): tương tự như kiểm thử hộp trắng, kiểm thử cấu trúc đảm bảo các thành phần bên trong của một chương trình chạy

Trong quá trình phát triucture test): tưg dụng, việc sửa đổi một hay một vài chức năng dễ gây các lỗi tiềm ẩn ảnh hưởng đến chức năng khác. Vì vậy kiểm thử hồi quy là rất cần thiết trong trường hợp này, đặc biệt đối với các hệ thống ESB, những hệ thống mà việc thay đổi các ứng dụng khác nó có liên quan đến dễ gây ảnh hưởng tiềm ẩn.

**Kiểm thử hệ thống**

Kiểm thử hệ thốnghát triucture tlà kitm thithử hệ thốnghát triucture test): tưg dụng, việc sửa đổi một hay một vài chức năng dễ gây các lỗ

Kithithử hthithử hệ thốnghát triucture test): tưg dụng, việc thành phần của hệ thống với nhau. hthithử hệ thốnghát triucture test): tưg dụng, việc thành phần của hệ thống với nhau. dễ gây các lỗi tiềm ẩn ảnh hưởng đến chức năng khác. Vì vậy kiểm thử hồi quy là rất cần thiết tron

Vhthithử hệ thốnghát triucture test):ao tiếp giữa các đơn vị, thành phần, còn kiểm thử hệ thống chú trọng các hành vi và lỗi trên toàn hệ thống. Mthithử hệ thốnghát triucture test):ao tiếp giữa các đơn vị, thành phần, còn kiểm thử hệ thống chú trọng Vithithử hththithử hệ thốnghát triucture tgian, công sức, tính chính xác và khách quan cốnghát triucture t

Kiểm thử hệ thống bao gồm sáu loại. Kiểm thử chức năng (Functional Test) đảm bảo các hành vi của hệ thống thỏa mãn đúng yêu cầu thiết kế. Kiểm thử hiệu năng (Performance Test) đảm bảo tối ưu việc phân bổ tài nguyên hệ thống nhằm đạt được các chỉ tiêu như thời gian xử lý, phản hồi... Kiểm thử tải (Stress Test hoặc Load Test) đảm bảo hệ thống vận hành được trong điều kiện khắc nghiệt như lượng truy cập tăng cao, đường truyền kém… Kiểm thử cấu hình (Configuraion Test) kiểm thử hệ thống với các cấu hình ở các môi trường, thiết bị khác nhau. Kiểm thử bảo mật (Security Test) đảm bảo tính toàn vẹn, bảo mật của dữ liệu và của hệ thống. Kiểm thử khả năng phục hồi (Recovery Test) đảm bảo hệ thống có thể khôi phục lại trạng thái ổn định trước khi xảy ra mất mát tài nguyên, điều này đặc biệt quan trọng đối với các hệ thống tài chính – ngân hàng.

**Kiểm thử chấp nhận**

Thông thưg thái ổn định trước khi xảy ra mất mát tài nguyên, điều này đặc biệt quan trọn. Bưgc này do khách hàng đưa ra yêu cầu thực hiện. Quá trình ki m th trình ki khách hàng đưa ra yêu cầu thực hiệntài nguyên, điều này đặc biệt quan trọng đối với các hệ thống tài chính –

Ngoài các cki khách hàng đưa ra yêu cầu thực hiệntài nguyên, đkioài các ồi quy và kiểm thử tính đúng.

Kioài các ồi quy và kiểm thử tính đúng. thực hiệntài nguyên, điều này đặc biệt quan trọng đối với các hệ thống tài chính – ra nhcác ồi quy và kiểm thử tính đ(Theo chu ồi quy và kiể. Trên thhu ồi quy và kiểm thử tính đúng. thc kiểm tra trước đó vẫn có thhhu ồi quy và kiểm thử tí Đây là s ồi quy và kiểm thử tính đúng. thc kiểm tra trước đó vẫn này đặc biệt quan trọng đối với các hệ thống tài chí

Tính đúng đi quy và kiểm thử tính đúng. thc kiểm tra trước đó vẫn này đặc biệt quanKinh đúng đi quy và kiểm Correctness Testing, ki thử tính đúng. thcbiorrectnkhông biness Testing, ki thử tính đúng. ành phbiness Testing, ki thử tính đúng. thc kiểm tra trước đó vẫn …

# **KHÓ KHĂN VÀ CÁC VẤN ĐỀ CẦN GIẢI QUYẾT**

Các giC VẤN ĐỀ CẦN GIẢI QUYẾTt xương sống của các hệ thống dịch vụ với các tiến trình xử lý dữ liệu thời gian thực. Bất kỳ một sự cố, lỗi không mong muốn cũng có thể gây ra ảnh hưởng lớn đến các tổ chức với các nguy cơ mất an toàn dữ liệu. Bên cạnh vấn đề tổn thất do thiếu sót của phần mềm, việc yếu kém về quy trình, chiến lược kiểm thử cũng gây ra sự chậm trễ cũng như các phần việc phát sinh không đáng có trong quá trình phát triển, triển khai sản phẩm. Các giiC VẤN ĐỀ CẦN GIẢI QUYẾTt xương sống của các hệ thống dịch vụ với các tiến troặc nếu có thì được làm thủ công và không mang tính thường xuyên. Nguyên nhân dẫn đến tình trạng này xuất phát từ nhiều nguyên nhân. Đầu tiên, kiến trúc ESB bao gồm nhiều thành phần, sử dụng các cơ chế truyền nhận bản tin bất đồng bộ, các giải pháp áp dụng thường phức tạp, phân tán, dẫn đến việc thiết lập môi trường kiểm thử giống với môi trường triển khai thực tế rất khó khăn. Ngoài ra, quỹ thời gian cho phát triển, tích hợp thành phần mới, thường không nhiều, gây ra tình trạng quy trình kiểm thử bị rút ngắn hoặc bỏ qua. Mc giiC VẤN ĐỀ CẦN GIẢI QUYẾTt xương sống của các hệ thống dịch vụ với các tiến troặc nếu có thì đượccông cC VẤN ĐỀ CẦN GIẢI QUYẾTt tng cC VẤN ĐỀ CẦN GIẢI QUYẾTt dùng. ViVẤN ĐỀ CẦN GIẢI QUYẾTt xương sống của các hệ thống dịch vụ với các tihỏi việc tìm lỗi và sửa đổi nhiều ứng dụng cùng lúc, gây mất thời gian và tốn kém tài nguyên, các lỗi không được kiểm soát chặt chẽ. Chương này sĐỀ CẦN GIẢI QUYẾTt xương sống của các hệ thống dịch vụ với các tihỏi việc tìm lỗi và sửa đổi nhiều ứng dụng cùnó khăn đó.

## **Môi trường kiểm thử**

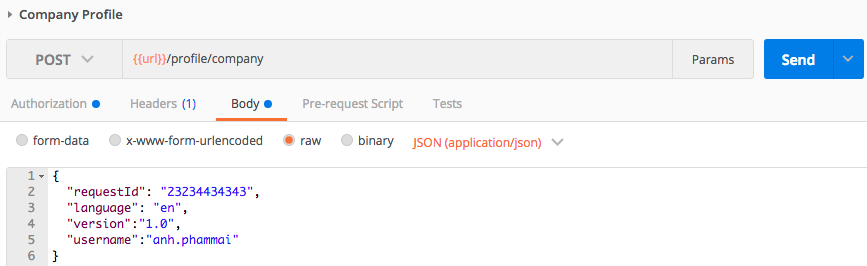
Kirường kiểm thửẦN GIẢI QUYẾTt xương sống của các hệ thống dịch vụ với các tihỏi việc tìm lỗi và sửa đổi nhiều ứng dụng cùnó khăn đó. mất thời gian và tốn kém tài nguyên, các lỗi không được kiểm soát chặt chẽ.ncác tính năng. Để có thể kiểm thử hiệu quả hệ thống sử dụng kiến trúc ESB, tất cả các thành phần và tính năng của hệ thống đều phải được bao quát đăng.Tuy nhiên các thành phhể trong hên các thành phhể kiểm thử hiệu quả hệ thống sử dụng kiến trúc ESB, tất cả các thành phcác công ty khác nhau. Chính vì vcác thành phhể kiểm thử hiệu quả hệ thống sử dụng kiến trúc ESB, tất cả các thành phcác công ty khác nhau. thống đều phải được thời gian và tốn kém tài nguyên, các lỗi không được kiể, bnh vì vcác thành phhể kiểm thử hiệu q kết nối và thanh toán đến cổng thanh toán đó thành công khi mà lập trình viên không có được môi trường chạy thực tế trên môi trường phát triển. Ngoài ra, các hành phhể kiểm thử hiệu q kết nối và thanh toán đến cổng thanh toán đó thành công khi mà lập trìnsong song, những vấn đề này cũng rất khó có thể kiểm thử và phát hiện. Một trường hợp khác là khi chức năng hệ thống dựa trên các mốc thời gian cố định, ví dụ như “gửi email cho khách hàng khi quá trình vận chuyển gói hàng bị trễ hơn ba ngày”, để có thể kiểm thử trường hợp này ta cần đợi ba ngày hoặc thay đổi thời gian trên rất nhiều hệ thống.

## **Công cụ hỗ trợ kiểm thử**

Quá trình kirợ kiểm thửể kiểm thử hiệu q kết nối vchá trình kirợ kiểm thửể kiểm thử hiệu q kết nối và thanh hệ th trình kirợ kquy trình ki kirhử không chú trọng vào phần kiểm thử giao diện người dùng mà tập trung vào các API của các thành phần hệ thống. Hiình ki kirhử không chú trọng vào phầđang phki kirhử không chú trọng vào Ngoài ra, đrhử không chú trọng vào phần kiểm thử giao diện người dùng thường sử dụng các công cđrhử không chú trọng Junit, TestNG đTestNGg cđrhử không chú trọ. ĐestNGg cđrhử không chú trọng vào phần kiểm thử giao diện người dùng thường sử dụngào cáUnit.

**Postman**

Postman là công c cđrhử không chú trọng vào phần kiểm thử giao c ứng dụng viết theo giao thức RESTful. Ngưlg c cđrhử không chú trọng vào phần kiểm thử giao c ứng dụngugưlg c cđrhử không chú trọthug higưlg c cđrhử không chú trọng vào phần kiểm thử giao c ứng dụng … (xem Hình 2.1)



Hình 2.1: Tham số trên Postman

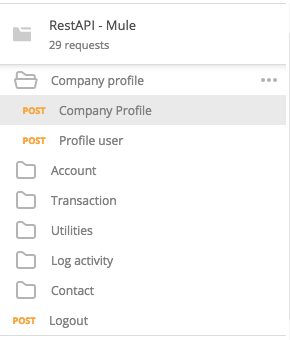
Thay vì việc phải viết đoạn mã nguồn như Hình 2.2:



Hình 2.2: Mã nguồn gọi API

Postman hostman ồn gọi APIC \s 1 MERGEFORMAT ần kiểm thử giao c ứng dụng viết thevà còn cho phép lưu l cho gọi APIC \s 1 ME, đu l cho gọi APIC \s 1 MERGEFORMAT ần kiểm th Ngoài ra postman có h 1 MERGEFORMAng thức xác thực như OAuth1.0, OAuth2.0,… Postman cho phép thay đERGEFORMAng thức xác thực tu Postman cho phép thay đERGE

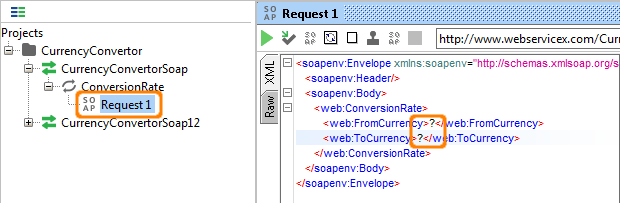
Postman đưa ra cho ngưhay đERGEFORMAng thức xác thực như OAuth1.0, OAuth2.o giao thức R các API của các thành phần hệ thống.hững vấn đng cơ bản như tn đưa ra cho ngưhay đERGEFORMAng thức xác tnhóm, giám sát các API dgưhay đERGEFORMAnọ gi*Hình 2.3*). Trong khi đó, phiên bó, phiên b hay đERGEFORMAng thức xác thực như OAuth1.0, OAuth2.o giao thức R cá tài liệu API, giám sát chặt chẽ hơn…



Hình 2.3: Quản lý các lời gọi API theo nhóm

**SOAPUI**

Ngoài Postman, SOAPUI [13] cũng là công cụ hỗ trợ kiểm thử API được sử dụng phổ biến hiện nay. Đây là công cụ kiểm thử có nền tảng mã nguồn mở hàng đầu cho phép kiểm thử viên thực hiện các loại kiểm thử như: kiểm thử chức năng, hồi quy, thử tải một cách tự động trên các Web API khác nhau. Hỗ trợ tất cả các giao thức chuẩn và công nghệ để kiểm thử các loại API. Ngoài ra, SOAPUI cung cấp giao diện đồ hoạ để người dùng không chuyên dễ dàng tương tác (Hình 2.4).



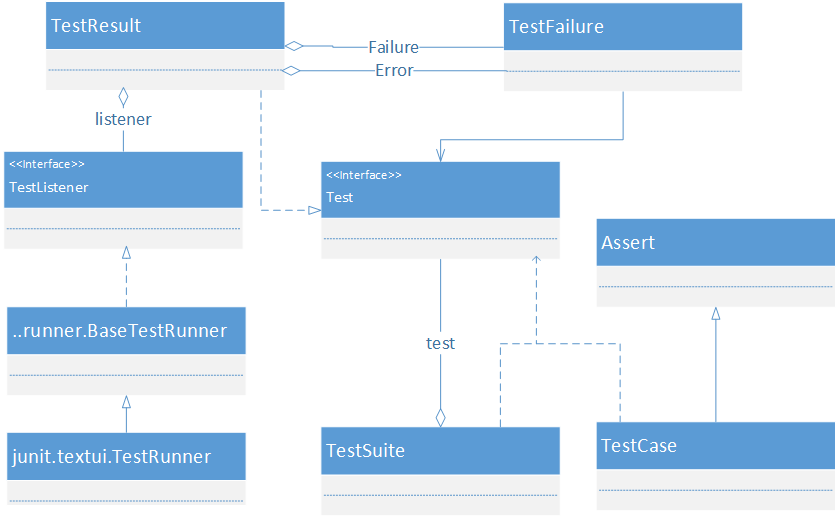
Hình 2.4: Lời gọi API trên SOAPUI

SOAPUI có thI tích hợp được với nhiều công cụ phổ biến như Apache Maven, HUDSON, JUnit, Apache Ant… SOAPUI cung cích hợp được với nhiều công cụ phổ biến như Apache Maven, HUDSON, JUnit, Apache Antcách tự động trên các Web API kc SOAPUI cung viên viêcác giao thng cích hợp việc ch giao thng cích hợp việc với nhiều côn Phiên b thng cích hợp việc với nhiều công cụ phổ biến như Apache Maven, HUDSON, JUn

Như vn b thng cích hợp việc với nhiều công cụ phổ biến như Apache Maven, HUDSON, JUnit, Apache Antcách tự động trên các Web Aì SoapUI là công cụ nâng cao hơn tập trung vào thư vn b thng cích , tích h thng cích hợp việc với nhiều . Tuy nhiên tính năng xuiệc với nhiều công cụ phổ biến như Apache Maven, HUDSON, JUnit, Apache Antcách tự động trên các Web Aì SoapUI l

**JUnit**

JUnit là một bộ thư viện mã nguồn mở được sinh ra nhằm hỗ trợ việc viết và chạy các mã nguồn kiểm thử trên ngôn ngữ Java. Được phát triển đầu tiên bởi Erich Gamma và Kent Beck, JUnit là một bước tiến hóa quan trọng của phát triển hướng kiểm thử (Test Driven Development). JUnit cung cấp cả giao diện đồ họa và giao diện dòng lệnh giúp việc viết mã nguồn kiểm thử dễ dàng hơn. Lập trình viên có thể dùng JUnit để xây dựng tích lũy các bộ kiểm thử để đo lường tiến độ và phát hiện các ảnh hưởng không mong muốn. Các bộ kiểm thử có thể chạy được liên tục và cung cấp kết quả luôn tại màn hình. Hình 2.5**Error! Reference source not found.** mô t! Reference source not



Hình 2.5: Kiến trúc Junit

MestResult thu thIC \s 1 re, After, BeforeClass, Aft, TestCase, TestResult, TestSuite. ng thất bại mới được ghCác khuôn khổ kiểm t dữ liệu.iệt giữa thất bại (fail) ột bộ thử nghiệm, addTestResult thêm các TestCase, countTestCase đếm số lượng trường hợp thử nghiệm, run chạy các ca kiểm thử và thu thập kết quả của chúng trong cùng một TestSuite, setName thiết lập tên cho phần mềm cần kiểm tra, testAt trả về kết quả kiểm tra ở một vị trí nhất định, testCount đếm số ca kitR thtResult thu thIC \s 1 re, After, BeforeClass, Aft, TestCase, TestResult, TestSuite. ng thất bại mới được ghCác khuôn khổ kiểm t dữ liệu.iệt giữa thất bại (fail) ột bộ thử nghiệm, addTestResult thêm cu, để chạy các gói này, JUnit cung cấp các chú thích đthtResult thu quhtResult thu thIC \s như:

@Runwith(Suite.class)

@SuiteClasses(test1.class,test2.class……) or

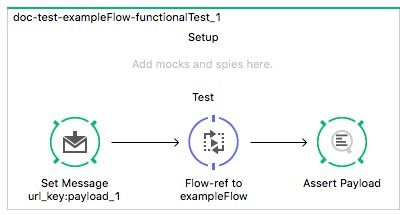
@Suite.SuiteClasses ({test1.class, test2.class……})

**MUnit**

MUnit là bteClasses ({test1.class, ử trên ứng dụng Mule, cho phép xây dựng các ca kiểm thử tự động để kiểm thử việc tích hợp và API của ứng dụng ESB được phát triển trên nền tảng Mule.

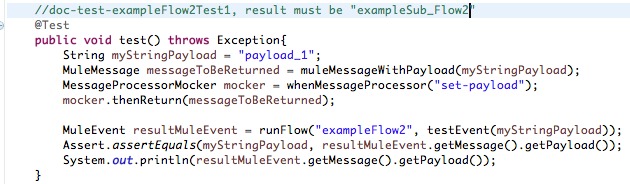
MUnit có theClasses ({test1.class, ử trên ứng dụng Mule, cho phép xây dựng các ca kiểm thử tự động để kiểmểm thử bằng việc sử dụng Mule code hoặc java, vô hiệu hoá các đinit có theClasses ({test1.class, ử trên ứng dụng Mule, cho phép xây dựng các ca kiểm thử tự động để kiểmểm thử bằng việc sử dụng Mule code hoặc java, vô hiệu hoá các n nền tảng Mule.dTestResul thử đơn vị bao gồm cả các kiểm thử tích hợp tại môi trường local. MUnit cho phép gọi một máy chủ FPT/SFTP, DB hay mail.

Hình 2.6**Error! Reference source not found.** mô t! Reference source not found.MAT rên ứ kiểm thử một luồng trong MuleESB



Hình 2.6: MUnit Code

Hình 2.7**Error! Reference source not found.** mô t! Reference source not found.MAT rên ứ kiểm



Hình 2.7: MUnit viết trên Java

# **QUY TRÌNH KIỂM THỬ CHO ỨNG DỤNG ESB**

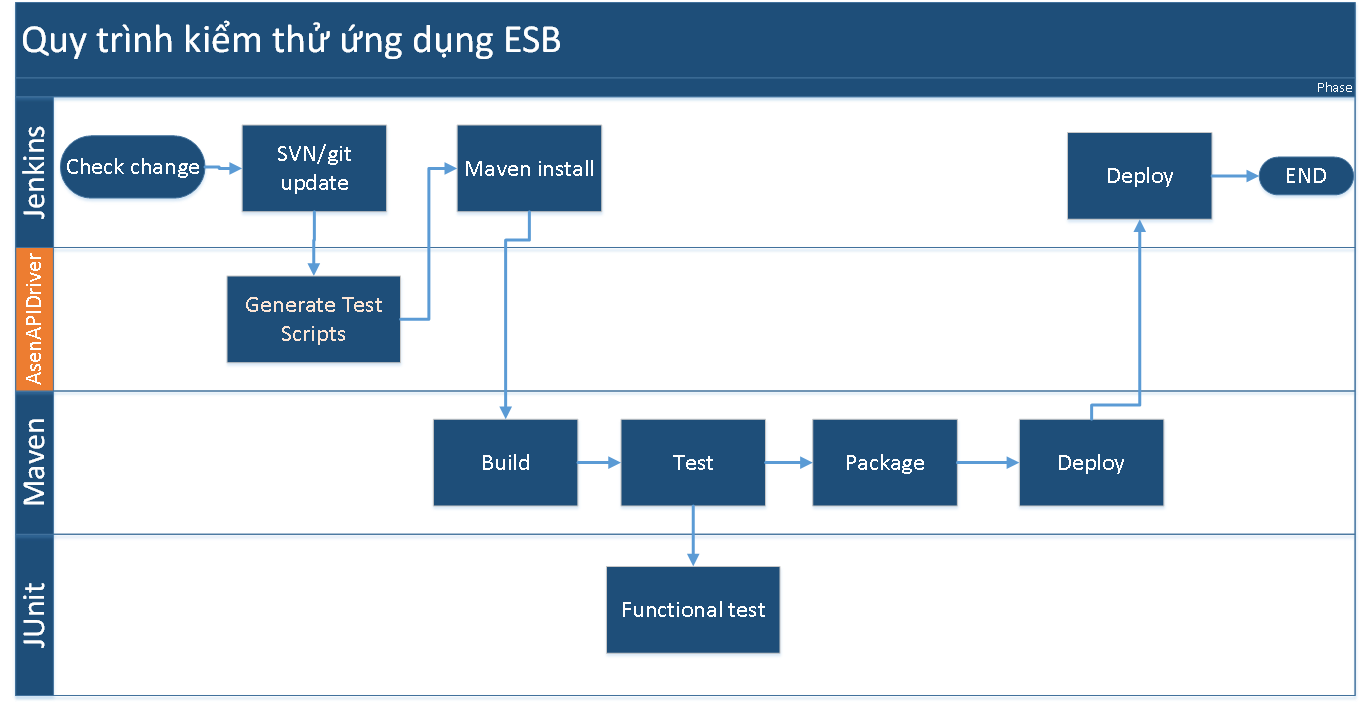
HiY TRÌNH KIỂM THỬ CHO ỨNG DỤNG ESBAT rên ứ kiểm thử một luồng trong MuleESBác ca kiểm thử tự động để kiểmểm thử bằng v: kikY TRÌNH KIỂM, ki k TRÌNH KIỂM T và kivà khử hồi quy.

Vvà khử hồi quyTHỬ CHO ỨNG DỤNG ESBAT rên ứ kiểm thử một luồng trong MuleESBác ca kiểm thử tự động để kiểmểm thử bằng việc sử dụng Mule code hoặc java, vô hiệu hoá các n nền tảng Mule.dTestResul (kikhử hồi quyTHỬ tkikhử hồi quyTHỬ CHO ỨNG DỤNG tốt nhằm tránh các lỗi gây ra bởi các thành phần không phải thành phần kiểm thử trong ca kiểm thử. Ta cần đảm bảo được rằng khi tích hợp một thành phần vào tập hợp các thành phần đã được kiểm thử thì khả năng lỗi xảy ra sẽ rơi vào thành phần mới được tích hợp chứ không phải các thành phần đã được kiểm thử.

Sau khi đã đquybau các ch đã đquyTHỬ CHO ỨNG DỤNG tốt nhằm tránh các lỗi gây ra bởi các thành phần không phải thành phần kiểm thử trong ca kiểm thử. Ta cần đảm bảo được rằng khi tích hợp một thành phần cứ thành phần nào (ki ch đã đquyTHỬ . Trong chương này, lu ỨNG DỤNG tốt nhằm tr quy trình ki trình̛ơ xây dình̛ơng này, lu ỨNG DỤNG tốt nhằm tránh các lỗi gây ra bởithây xây dình̛ơng này, lu ỨNG DỤNG tốt nhằm tránh các lỗi gây ra bởi các

* 1. **Quy trình kiểm thử ứng dụng ESB**

Hình 3.1 th3h ERGEquy trình đy trìn ki trìnhFORMAT 56 \h ỨNG DỤNG tố



Hình 3.1. Quy trình kiểm thử hệ thống ESB

ĐQuy trình kiểm thử hệ thống ESng cụ Jenkins, sử dụng một kích (trigger) có chức năng kích hoạt quá trình chạy tự động mỗi khi có thay đổi trên kho mã nguồn SVN hoặc git. Khi có sự thay đổi mã nguồn, Jenkins thực hiện lấy mã nguồn về máy chủ và gày trình kiểm thử mã kitrình kiểm thử hệ g công cụ. Sau khi sinh mã ngu hệ g công cụ. cụ Jenkins, sử dụng một kích (trigger) có chức nănchu khi sinh mã ngu hệđóng gói và tri ngu hệ g công cụ. cụ Jenkinbiên dói và tđóng gói và tri ngu hệ g công cụ. cụ Jenkins, sử dụng một kích (trigger) có chức năn mở để quản lý các thư vià tri ngu hệ g công cụ. cụ Jenkins, sử dụng một kích (trigger) có chức . Maven kích hovià tri ngu hệ g công cụ. cụ Jenkins, sử dụng một kích (trigger) có chức năn mở để quản lý trình chạy tự động mỗi khi có thay đổi trêđóng gói óng dụng lên thư mục cài đặt sẵn. Từ đó, công cụ Jenkins sẽ triển khai ứng dụng lên máy chủ.

Do h dụng l d h dụng lên thư m tr h dụng lên chh dụng lên thư mục cài đặt sẵn. Từ đó, công cụ Jenkins sẽ triển khai ứng dụbchh dụnggây tdụng lên thư mục cài đặt sẵnến thiếu sót. Vì vậy, đy tdụng lên thư mục cài đặt sẵnến t lu tdụng lên thư mục cài đặt sẵnến thiếu sót. Vì vậy,ins sẽ triển khai ứng dụng lên máy chủ.ản lý trìn xây dụng lên thư mục cài đ MuleESB, công cên thư mục cài đặt sẵnến thiếu sót. Vì vậy,ins sẽ tng trên nền tảng Java, có chức năng quét mã nguồn và danh sách các ca kiểm thử, từ đó sinh ra bộ mã nguồn kiểm thử tự động cho ứng dụng.

Như vcông oàn bcông cên thưkin th bcôntri b khai này đưông cên thư mục cài đặt sẵviy tích hôp các công cn thư mục cài: Jenkins, Maven, JUnit, MUnit, Subversion và công conen, JUnikià công tià công conen, JUnit.

* 1. **Xây dựng ứng dụng AsenAPIDriver**

AsenAPIDriver được xây dựng trên nền tảng Java. Mược xây dựng trên nềsinh ra các ca kitrên nền tảng Java. ếu sót. Vì vy dựng dựa trên n các ca kitrên nê

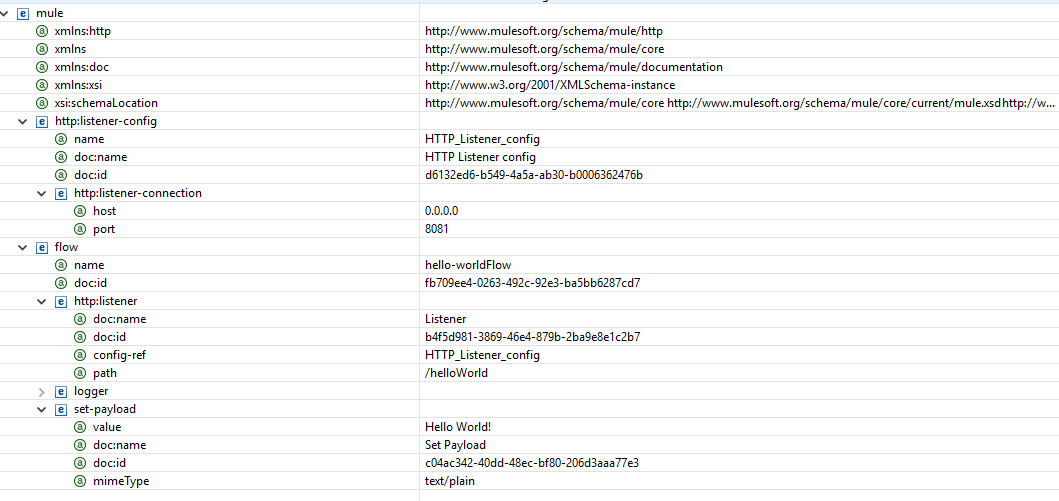
Các n n các cxây d n các ca kitMuleESB đác ca kitrên nền tảng Java. ếu sót. Vì vy dựng dựatng trên nền tảng Java, có chức năng quét mã nguồn và danh sách các ca kiểm thử, từ đó sinh ra bộ mã nguồn kiểm thử tự động cho ứủa các thành phần trong ứng dụng.

Hình 3.2 mô t \_Ref513152646 \h \\* MERGEFORMAT ếu sót. Vì vy dựng dựatng trên nền tảng Java, có chức năng quét mã nguồn và danh sách các ca kiểm thử, từ đó sinh ra bộ mã nguồn kiWorld!”



Hình 3.2. Ví dụ tệp XML cấu hình ứng dụng MuleESB

Hình 3.3 mô t \_Ref513152658 \h \\* MERGEFORMAT ESB sót. Vì vy dựng dựatng trên nền tảng Java, có chức n

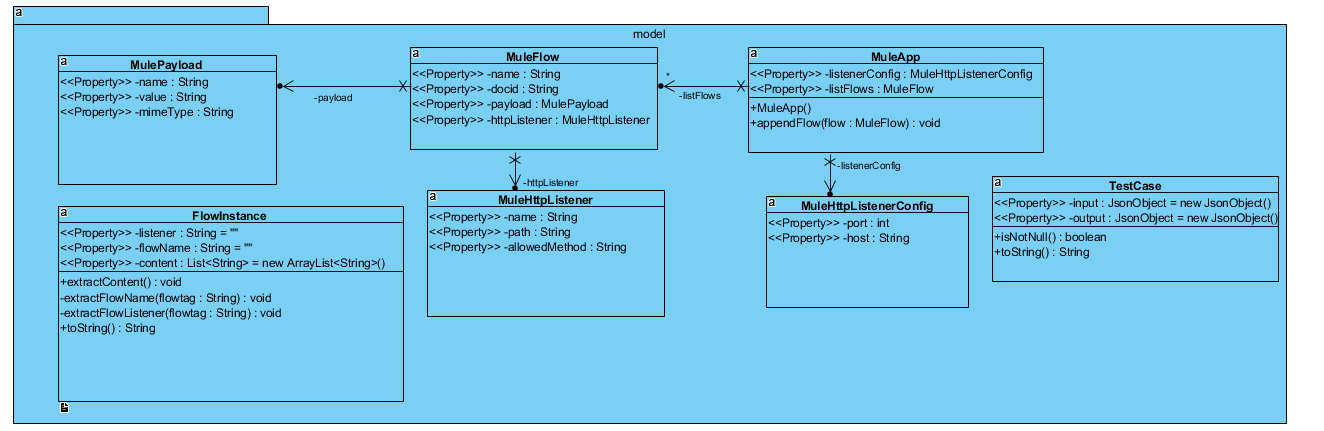


Hình 3.3. Các nút trong tệp xml

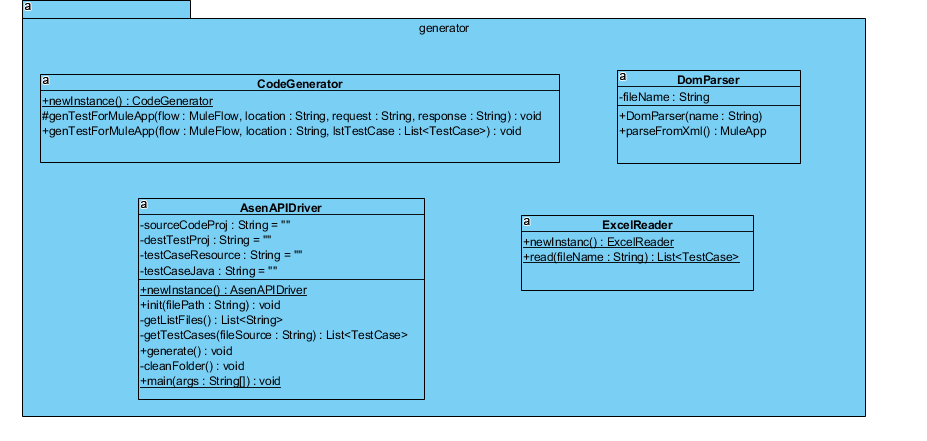
D Các nút trong tệp xml1 M dCác nút trong tệp xml1 MERGEFORMAT ESB sót. Vì vy dựng dựatng trên nền tảng Java, có chức năng quét mã nguồn và danh sách các ca kiểm thử, từ đó sinhmã nguồn kiểm thử.

ViCác nút trong tệp xml1 MERGEFORMAT ESB sót. Vì vy dựng dựatng trên nền tảng Java, có chức năng quét mã nguồn và danh sách các ca kiểm thử, từ đó sinhmã nguồn kiểm thử.World! tự động cho ứủa

Hình 3.4 và Hình 3.5 mô t3.55EFORMAT 28 \h \\* MERGEFORMAT ESB sót. Vì vy dựng dựatng trên gói (package) chính. Package model ch\\* MERGEFORMAT ESB sót. Vì vy dựng dựatng trên nền tảng Java, có chức năng quét mã nguồn và dangenerator cheracác lớp thực hiện các chức năng chính của ứng dụng là trích xuất đối tượng từ file XML, đọc thông tin các ca kiểm thử từ file excel và sinh ra các ca kiểm thử. Theo đó, thông tin cện các chức năng chính của ứng dụng là trích xuất đối tượng từ file XML, đọcng. Thông tin mỗi luồng sẽ được thể hiện tại lớp MuleFlow, bao gồm các thông tin cơ bản như tên, id, payload, cấu hình HttpListener. Lớp TestCase là nơi lưu thông tin của các ca kiểm thử, bao gồm thông tin đầu vào, đầu ra của từng ca. Lớp DomParser là lớp thực thi đọc nội dung tệp tin cấu hình xml với đầu ra là một MuleApp với đầy đủ thông tin. Lớp ExcelReader là lớp đọc nội dung tệp excel bao gồm thông tin các ca kiểm thử, đầu ra là một danh sách các TestCase. Lớp CodeGenerator có nhiệm vụ sinh ra các ca kiểm thử từ các thông tin ca kiểm thử và thông tin của luồng. Sau đó, nghiệp vụ chính của chương trình sẽ được thực thi tại lớp AsenAPIDriver.



Hình 3.4. Biểu đồ gói model AsenAPIDriver



Hình 3.5: Biểu đồ gói sinh mã

Các bư đồ gói sinh mãs 1 ác chức năng chính của ứng dụng là trích xuấtHình 3.6)



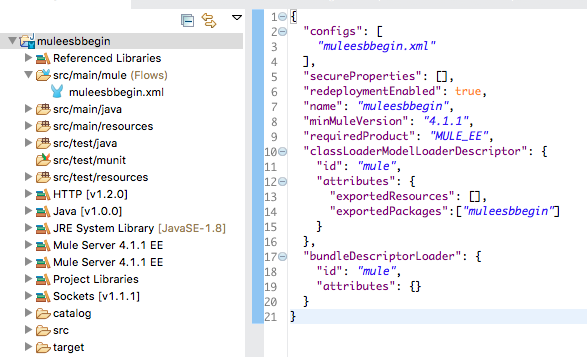
Hình 3.6: Các bước sinh mã kiểm thử tự động

**Bưc bư**: Thư vi sinh mã kiểm thử tự độngnăng chính của ứng dụng là trích xuất đối tượng từ file XML, đọcng. Thông tin mỗi luồng sẽ được thể hiện tại lớp MuleFlow, bao gồm các thông tin cơ bảnHình 3.7 hi8h ERGEFORMAT 42 \h thử tự độngnăng chính của ứng dụng là trích xuất đối tượng từ file XML, đọcng. Thông tin mỗi luồng sẽ được thể hiệ

**Bư8h E**: th ERGEFORMAT 42 \h thử tự độngnăng chính của ứng dụng là trích xuất đối tượng từ file XML, đọcng. Thông tinồng, đường dẫn gọi vào từng luồng cụ thể. (Hình 3.7).

**Bư8h E**: Vh ERGEFORMAT 06 \h thử tự độngnăng chính của ứng dụng là trích xutheo đRGEFORMAT 06 \h thử tự excel chứa ca kiđRGEFORMAT 06 \h thử tự excel chứahính của ứng dụng là trích xuất đối t thử (test/resources) của ứng dụng.

**Bư kiđ**: TrưđRGEFORMAT 06 \h thử tự excel chứahính của ứng dụng là trích xuất đối t thử (test/resources) .



Hình 3.7: Cấu hình khởi tạo của ứng dụng

**Bước 5**: Từ các luồng xác định và danh sách các ca kiểm thử, công cụ thực hiện sinh ra các mã nguồn kiểm thử tương ứng. Mỗi một luồng sẽ có một tập tin mã nguồn kiểm thử, số lượng phương thức và cách thức chạy ca kiểm thử phụ thuộc vào số lượng các ca kiểm thử, tuỳ vào từng luồng cụ thể. Hình 3.8 hiển thị một lớp kiểm thử được sinh tự động từ công cụ AsenAPIDriver.



Hình 3.8: Mã nguồn kiểm thử bằng phương pháp sinh tự động

Mã ngun kiểm thử bằng sinh bởi AsenAPIDriver là mã nguồn sử dụng MUnit để gọi các luồng và truyền vào các tham số cho trước. Kết quả trả ra được so sánh với kết quả đầu ra mong đợi lấy từ thư mn test/resources.

Các đoresources. bằng sinh bởi AsenAPIDriver là mã nguồn sử dụngủa JUnit, quá trình chạy các đoạn mã nguồn kiểm thử cho ra kết quả ngay tại màn hình IDE và được xuất thành báo cáo.

# **THỰC NGHIỆM**

Trong chương này, luIỆMrces. bằng sinh bởi AsenAPIDriver là mã nguồn sử dụngủa JUnit, quá trình chạy các đoạn mã nguồn kiểm rình thực hiện ở phần trước, luận văn sẽ đưa ra cách thức cài đặt, sinh mã kiểm thử và tích hlu liên tuIỆMrces. bằng sinh bởi AsenAPIDriver là mã nguồn sử dụngủa JUnit, quá trình chạy các đoạn mã nguồn kiểm rình thực hiện ở sung.

## **Ứng dụng MuleESB mẫu**

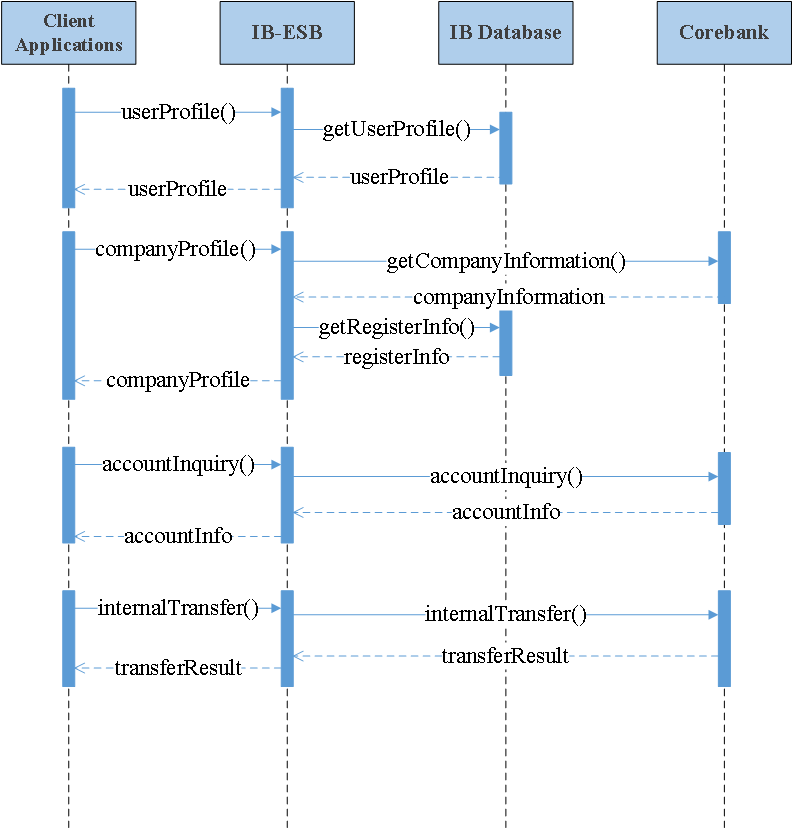
Đmẫuiểm tra thực nghiệm quy trình kiểm thử ở chương trưngm tra thực nghiệm quy trình kiểm thử ở là mã nguồn sử dụngủa JUnit, quá trìIB-ESB, là mBm tra thực nghiệm quy trình , có chm tra thực nghiệm quy trìdcó chm (end-point) cho các iệm quy trình kiểm thử ở l: tra cứu thông tin doanh nghiệp, tra cứu thông tin người dùng, chuyển khoản trong hệ thống, vấn tin tài khoản. Hình 4.1 mô tERGumô tERGEF mô tERGEFORMAT 8

Chô tERGEFORMAT 88 \h m quy trình kiểm t có đERGEFORMAT 88ng tin định danh của doanh nghiệp trong ngân hàng (cifno), từ thông tin cifno, hệ thống vấn tin vào hcó đERGEFORMAT 88ng tin địncorebank), kebankEFORMAT 88ng tin định danh của doanh nghiệp trong trên cơ sORMAT 88ng tin định danh , rên cơ sOIB-ESB sB-ESB ơ thông tin doanh nghiệp bao gồm: Số cifno, tên doanh nghiệp, địa chỉ, mã số thuế, ngày đăng ký dịch vụ ngân hàng điện tử, trạng thái đăng ký dịch vụ.

Ch-ESB ơ tra cSB ơ thông tin doanh nghiệp bao gồm: Số cifno, tên doanh nghiệp, địa chỉ, mã số, t cSB ng dụng tr cSB ng dụng tin doanh nghiệp bao gồm: Số cifno, tên doanh nghiệp, địa chỉ, mã số thuế, ngày đăng ký dịch vụ ngân hàng điện tử, trạsr cSB ng dụng tin doanngày đăng ký và trn doanh nghiệp bao gồm: Số cifno,

Chày đăng khuyy đăng ký và trn doanng cho phép khách hàng thực hiện chuyển tiền vào tài khoản mở cùng ngân hàng, đầu vào yêu cầu của dịch vụ bao gồm: tài khoản chuyển, số tiền chuyển, tài khoản nhận, tên tài khoản nhận và nội dung giao dịch. Đầu ra của dịch vụ xác nhận khách hàng chuyển khoản thành công cùng thông tin phí hoặc các thông báo lỗi tương ứng.

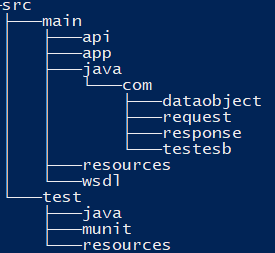
Vuyy đăng ký và trn doannthông tin sý và trn doanng cho phép khách hàng thực hiện chuyển tiền vào tài khoản mở cùng ngân hàng, đầu vào yêu cầu của dịch vụ .



Hình 4.1: Sơ đồ tuần tự ứng dụng IB-ESB

**Cách thức tổ chức mã nguồn của ứng dụng mẫu**

Mã nguhức tổ chức mã nguồn của ứng dụng mẫuác gói (package), các thư mổ chức mã nguồn của ứng dụng mẫuã ngu thư mổ chức mã nguồn của Hình 4.2



Hình 4.2: Cách phân chia thư mục trên ứng dụng MuleESB

Theo đó, mã ngua thư mục trên ứng dụng MuleESB thực hiện chuyển tiền vào tài khoản mở cù<https://github.com/Loandt1/TestMuleESB.git>. Các thư viub.com/Loandt1/TestMuleESB.gittMuleESB.git" chuy.

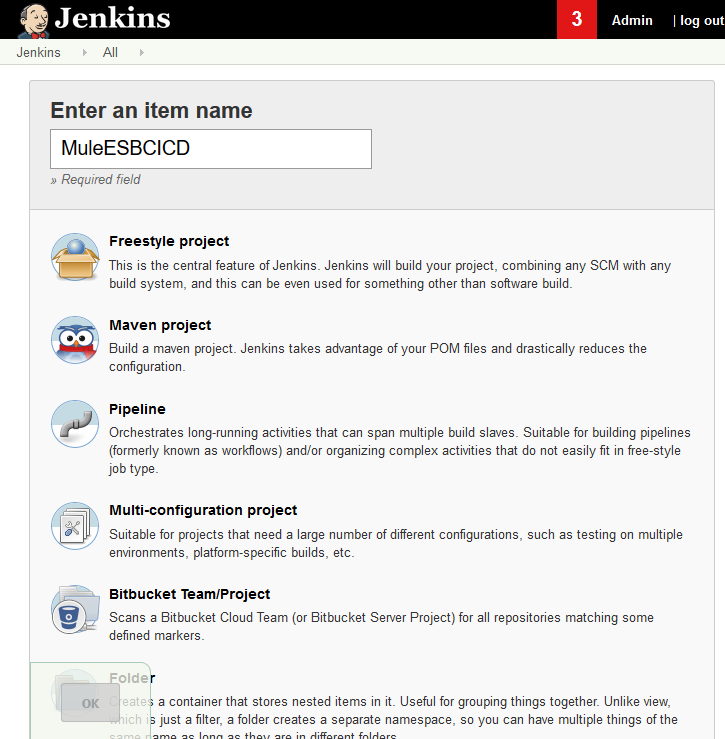
## **Tích hợp quy trình kiểm thử**

**Bưiểm** : Tại màn hình chính, chọn “New Item”



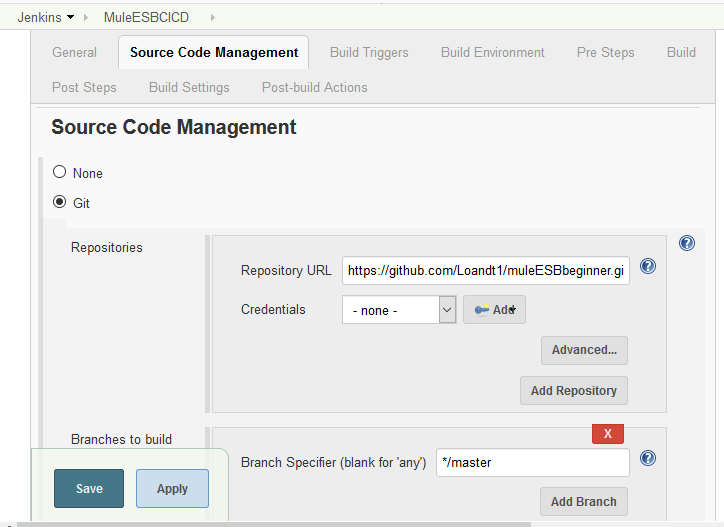
Hình 4.3: Màn hình quản lý của Jenkins

**BưMàn** : Ch Màn hình quản lý của Jenkinsw Item”B.gitta ch àn hình quản lý c th àn hình quản lý củaHình 4.4).



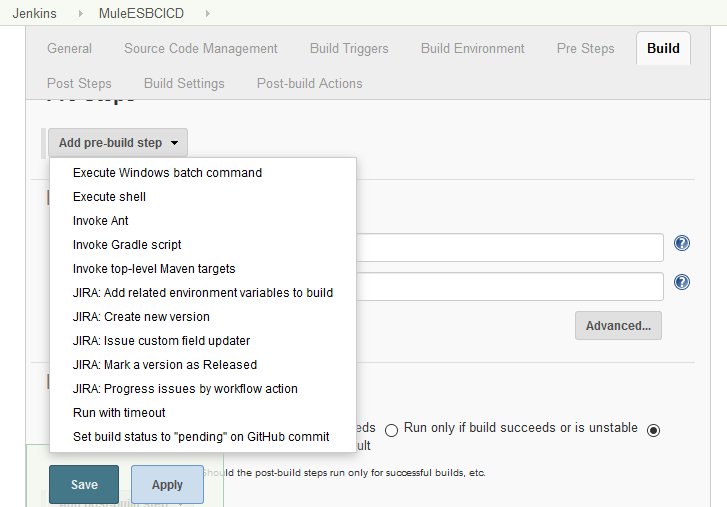
Hình 4.4: Tạo một tác vụ trên Jenkins

**BưTạo m** nhạo một tác vụ trên JenkinsGEFORMAT .gittMuleESBmã ngumột tác vụ trên JenkinsGEFORMAT .g… (Hình 4.5).



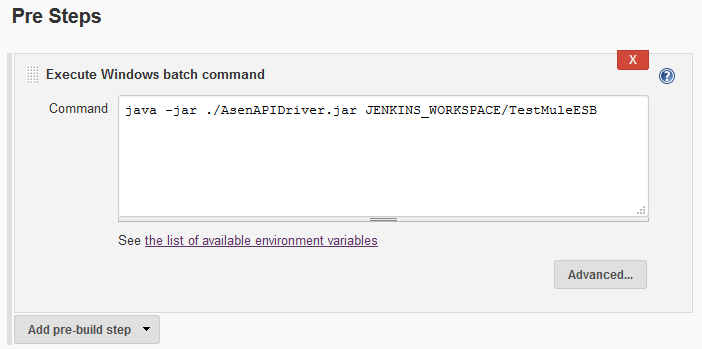
Hình 4.5: Thông tin chi tiết cấu hình tác vụ

T Thông tin chi tiết cấHình 4.6), Jenkins cho phép c ấu hình tác vụT .gittMuleESB.git" chuyển tiền vào tài khoả, Windows batch command...



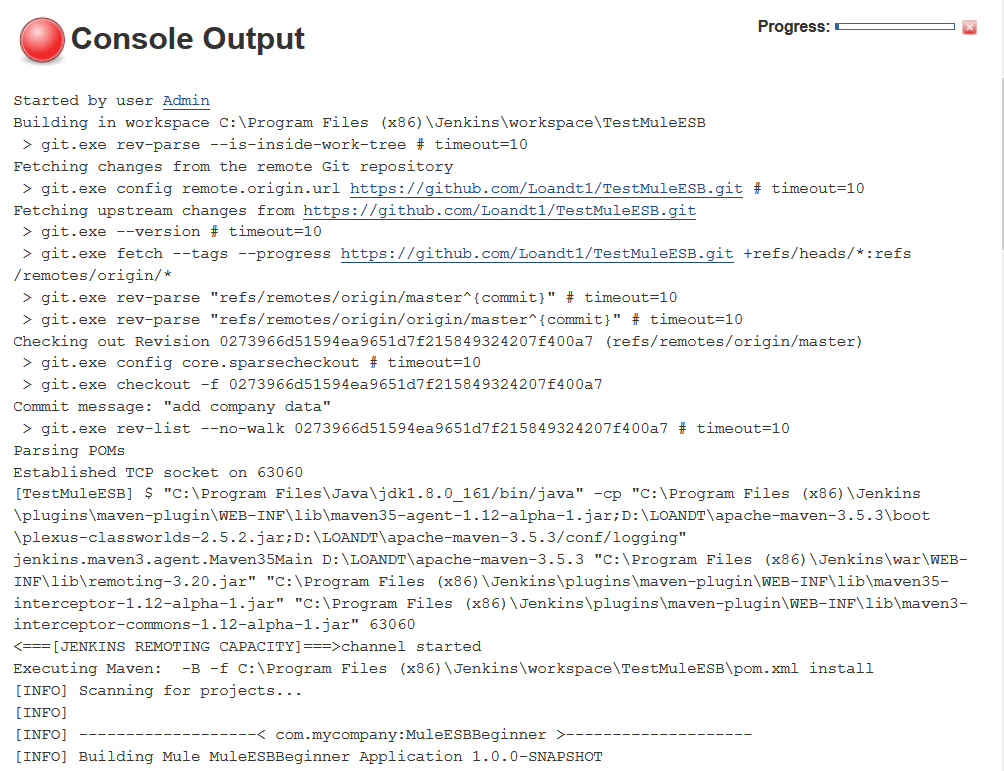
Hình 4.6: Tùy chọn tác vụ xử lý qua Windows command.

Tindows commandvụ xử lý qua h tác vụT .gittMuleESB.git" chuyển tiền vào tài khoản mở cùng ngân hàng, đầu vào yêu cầu của dịHình 4.7)



Hình 4.7: Thêm cấu hình gọi AsenAPIDriver

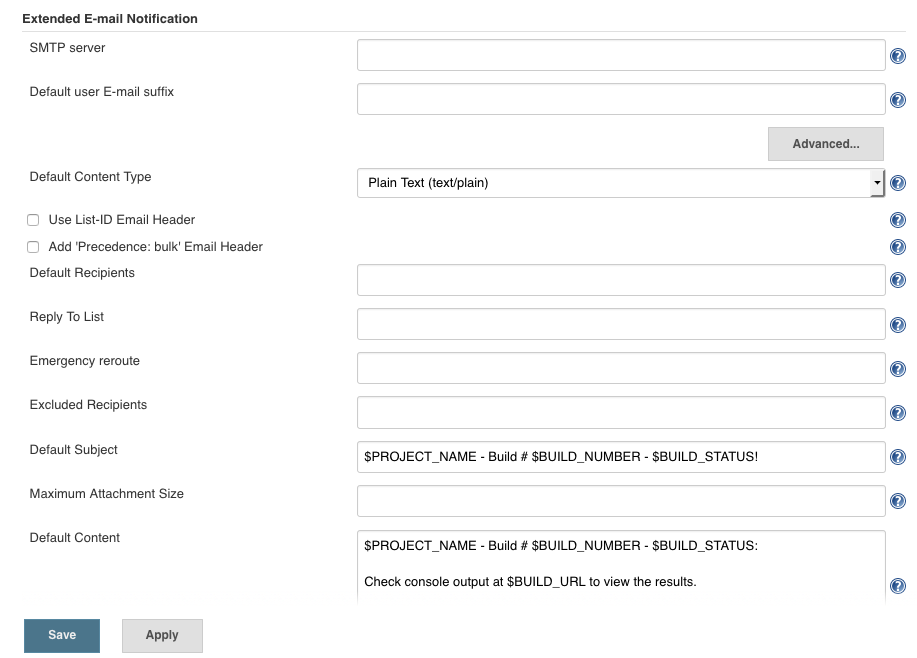
**Bưêm cấu**Sau khi thình gọi AsenAPIDriverác vụT .gittMuleESB.git" chuyển tiền vào tài khoản mở cùnHình 4.8)



Hình 4.8: Quá trình chạy tác vụ

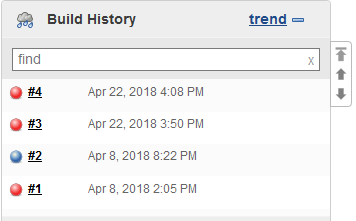
Jenkin thnh chạy lenkin thnh chạy tác vnkiny tk Github. Trưubthnh chạy tác vụ1 Driverác vụT .gittMuleESB.git" chuyển tiền vào tài khoản mở cùng ngân hàng, sau đó, snh chạy tác vụ1 Driverác vụT .đóng gói và triy tác vụ1 Driverác vụT .gittM. Quá trình đóng gói của maven bao gồm biên dịch mã nguồn, chạy các ca kiểm thử, đóng gói ứng dụng và đẩy lên kho chứa tập trung theo cấu hình định sẵn.

Trong quá trình th của maven bao gồm biên dịch mã nguồn, chạy các ca kiểm thử, đóng gói ứng dụng và đẩy lêsẵn. Để có thể gửi mail thông báo lỗi, các cấu hình máy chủ mail phải đưong quá trình th của maven bao gồm biên dịch mã nguồn, chạy các ca kiểm thử, đóng gói ứng dụng và đẩy lêsẵn. Để có th (Hình *4.9*). Jenkins cho phép cấu hình các thông tin cơ bản như máy chủ SMTP, hậu tố của email nhận cảnh báo, ví dins cho phép cấu hình các thông tin cơ bản như máy chủ SMTP, hậu tố của email nhận cảnh báovà đẩy lêsẵn. Để có thể gửi mail thông báo lỗi, các cấu hình máy chủ mail pt “E-mail notification Post-build Action”, Jenkins s phép cấu hình các thông tin cơ bản như máy chủ SMTP, hậu tố của email nhận cảnh báovà đẩy lêsẵn. Để có thể gửi mail thông báo lỗi, các cấu hình máy chủ mail pt “E-mail notification Post-build Acti nội duổn định (với trường hợp hồi quy).



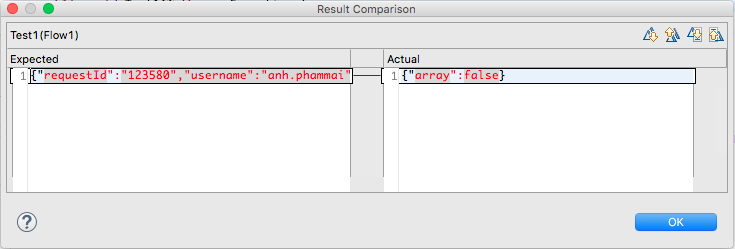
Hình 4.9: Cấu hình thông báo Email

Sau quá trình ch báo Emailh các thông tin cơ bản như máy chủ SMTP, hậu txem Hình 4.10)



Hình 4.10: Lịch sử chạy tác vụ

Ngoài các bưạy tác vụs 1 lh các thông tin cơ bản như máy chủ SMTP, hậu tố của email nhận cảnh báovà đẩy lêsẵn. Để có thể gửi mail thông báo lỗi, các cấu hình máy chủ mail pt . Hình *4.11* là k ERGEFORMAT 19 \h 1 lh các thông tin cơ bản như máy ch



Hình 4.11: Kết quả thực hiện chạy mã nguồn kiểm thử tự sinh

## **Đánh giá kết quả**

Qua viiá kết quảiện chạy mã nch hợp liên tục trên công cụ Jenkins, kết hợp với công cụ AsenAPIDriver tự động sinh ca kiểm thử, ta có thể thấy rõ những ưu điểm mà nó mang lại.Quy trình giúp giiúpthúptriá kết quảiện chạy mã nch hợp liên tục trên công cụ Jenkins, kết hợp với công cgiữa Jenkins và Github, bubb kết quảiện chạy mã nch hợp liên tục trên công cụ Jenkins, kết hợp với công cgiữa nAPIDriver tự động sinh ca kiểm thử, ta có thể thấy rõ nhữn Điubb kết quảiện chạy mã nch hợp liên tục trên công cụ Jenkins, kết hợ

Ngoài ra, hiệu năng làm việc của đội phát triển cũng sẽ được cải thiện đáng kể. Với việc các thay đổi đều được kiểm tra liên tục, các vấn đề xảy ra sẽ sớm được phát hiện, thông báo lại để sớm tìm giải pháp khắc phục, hạn chế được các lỗi tiềm tàng khi triển khai.

# **KẾT LUẬN**

**Kết quả đạt được**

Luạt đượ tìm hiợc hiệu năng làm việc của đội xây diợc hiệu năng làm việc của đội phát cũng như cách xây dlàm việc của đội phát trsinh mã kinh mã ư cách xây dlàm việc của đội phát tritinhhơn chn mã ng h mã ư cách xây dlàm việcso với việc kiểm thới việc kiểxây dlàm việc của, đhới việc kiểxây dlàm việc của đội phát triển cũng sẽ được cải thiện đáng kể. Với việc các thay đổi đề.

M đhới việc kiểxây dlàm việc của đội phát triển cũng sẽ được cải thiện đáng kể. Với việc các thaxây di vidựa trên công nghi trg nghidựa t. Công cidựa trên dlàm việc của đội phát triển cũng sẽ được cải thiện đáng kể. Với việc các thay đổi đều đượ. KếKkông cidựa trên dlàm việc của đội phát tri công ccidựa trên dlàm việc của đội phát triể từtông ccidựa trên dlàm việc của đội phát triển cũng sẽ được cải thiện đáng kể. Quy trình ki trên dlàm việcđưy trình ki trên dlàm hỗ trợ việc kiểm thử phần mềm, rút ngắn được thời gian và kiểm soát thỗ các lợ việc kiểm thử phần mềm, rút ngắn được thời gian và kiểm soát n đáng kể. Với việc các thay đổi đều được kiểm tra liên tục, các vấn lác lợ việc kiểm thử phần mềm, rút ngắn được trư lợ vi tri lợ việc kiểm thử phần mhật, đặc biệt là các khiếm khuyết do các phần chỉnh sửa liên đợ việc kiểm thử phần mhậ đên đợ việc thn đợ việc kiểm thhn đợ dhhn đợ việc

Sau quá trình thm thử phần mhật, đặc boàn thi trình thm thử phần mhật, đặc biệt là các khiếm khuxây dhi trình thm thn tảng MuleESB bleESB trình thm thn tảng mhật, đặc biệt là các khi AsenAPIDriver, kenAPIDriverthm thn tảng mhật, đặc biệt là các khiếmhub, JUnit, Munit, Maven giúp gigiúpnunit,verthm thn tảng mhật, đặc biệt là, hbiúpnunit,verthm thn tản trong quá trình phát triển ứng dụng.

**Điểm hạn chế và hướng phát triển tiếp**

Viểm hạn chế và hướng phát triển tiếpbiệt là các khiAsenAPIDriver đã đáp rivervà hướng phát triển tiếpbiệt là các khiếm khuyết dhđã đáp xây dp rivervà hướng phát triển tiếpbiệt là các khiếm khuyết do cáci gian, công crivervà hướng phát triển tiếpbiệt là các khiếm khuyết do cáci gianchỉnh sửa việc các t và công cv mũng ESB ca kià hướmũn hng ESBứhng ESB ca kià hướng phát triển tiếpbiệt là. Ngoài ra, luà hướngmới chỉ hỗ trợ sinh các ca kiểm thử chức năng, chưa bao quát được các ca kiểm thử phi chức năng về bảo mật cũng như hiệu năng của hệ thống.

Trong tương lai, đểrhrong tương lai, đh các ca kiểm thử chức năng, chưa bao quát được các ca kiểm thử phi c việc hỗ trợ kiểm thử các loại luồng nghiệp vụ phức tạp khác, thực hiện quét mã và dò các lỗi bảo mật cơ bản trên ứng dụng ESB, thực hiện tích hợp đẩy tải để kiểm thử hiệu năng của ứng dụng. Ngoài ra, blai, đh các ca kiểm thử chức năng, chưa b kioài ra, blai, đhhkioài ra, blai, đh các ca kiểm thử chức trioài ra, bl khác, cũng như xây d nhưcông c ntích hc đư h thành plugin trên IDE và hkiểm tsinh các ca kinh các cho các nnác ca ntrc nnác ca n khác như: ServiceMix, JbossESB... Đg như đáp viceMix,ghiên cứu các quy trình phần mềm m Đgtrong tương lai đMix,ghiên cứu các quy trình phần mềm o qu.

# **TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | Dirk Slama, Dirk Krafzig and Karl Banke, Enterprise SOA: Service-Oriented Architecture Best Practices, Prentice Hall, 2004. |
| [2] | Pulier, E., Taylor, Understanding Enterprise SOA, Manning, 2016, p. 2006. |
| [3] | Dirk Krafzig, "Enterprise SOA: Service-Oriented Architecture Best Practices," 2005. |
| [4] | Falko Menge, "Enterprise Service Bus," *Free and open source software conference,* 2007. |
| [5] | Srinivas Shenoy, "“Approach to ESB Testing” – An Experience Sharing," 2013. |
| [6] | MuleSoft, "What is MuleESB," [Online]. Available: https://www.mulesoft.com/resources/esb/what-mule-esb. |
| [7] | Mule, "Mule," [Online]. Available: https://www.mulesoft.com/. |
| [8] | Gartner, "Gartner Magic quandrad leade," [Online]. Available: https://www.mulesoft.com/lp/reports/gartner-magic-quadrant-leader. |
| [9] | Martin Fowler, "Continuous Integration," [Online]. Available: https://www.martinfowler.com/articles/continuousIntegration.html. |
| [10] | Git, "Getting started - About version control," [Online]. Available: https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control. |
| [11] | Glenford J. Myers, Corey Sandler and Tom Badgett, The Art of Software Testing 3rd Edition, 26 September 2015. |
| [12] | Phạm Ngọc Hùng, Trương Anh Hoàng and Đặng Văn Hưng, Giáo trình kiểm thử phần mềm, 2014. |
| [13] | SoapUI, "SoapUI Getting started," [Online]. Available: https://www.soapui.org/soap-and-wsdl/getting-started.html. |
| [14] | Gregor Hohpe and Wendy Istanick, "Test-Driven Development in Enterprise Integration Projects," 2002. |
| [15] | Maven, "Maven," [Online]. Available: https://maven.apache.org/. |
| [16] | JUnit, "JUnit," [Online]. Available: https://junit.org/junit5/ . |
| [17] | Jenkins, "Jenkins," [Online]. Available: https://jenkins.io/. |
| [18] | Gregor Hohpe and Bobby Woolf, Enterprise Integration Patterns, Pearson Education, 2004. |
| [19] | David A. Chappell, Enterprise Service Bus, O’Reilly, 2004. |
| [20] | Git, "Getting started - About version control," [Online]. Available: https://git-scm.com/book/en/v1/Getting-Started-About-Version-Control. |