

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM**

HUỲNH TẤN PHÁT – NGUYỄN CHÍ THÀNH

**NGHIÊN CỨU VÀ CẢI TIẾN HỆ THỐNG
CAMUDA ĐỂ PHÁT TRIỂN NHANH CÁC HỆ
THỐNG QUẢN LÝ VỚI BPMN**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP. HCM, 2018

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

HUỖNH TẤN PHÁT – 1412383

NGUYỄN CHÍ THÀNH – 1412495

**NGHIÊN CỨU VÀ CẢI TIẾN HỆ THỐNG CAMUDA ĐỂ PHÁT TRIỂN
NHANH CÁC HỆ THỐNG QUẢN LÝ VỚI BPMN**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

GIÁO VIÊN HƯỚNG DẪN

ThS. Ngô Chánh Đức

KHÓA 2014 - 2018

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TP.HCM, ngày tháng năm

Giáo viên hướng dẫn

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Khóa luận đáp ứng yêu cầu của Khóa luận cử nhân CNTT.

TP.HCM, ngày tháng năm

Giáo viên phản biện

LỜI CẢM ƠN

Chúng em chân thành cảm ơn Khoa Công Nghệ Thông Tin, trường Đại Học Khoa Học Tự Nhiên, Đại học Quốc gia Tp. Hồ Chí Minh đã tạo điều kiện thuận lợi cho chúng em trong quá trình học tập và thực hiện đề tài tốt nghiệp.

Chúng em xin nói lên lòng biết ơn sâu sắc đối với thầy Ngô Chánh Đức, thầy đã luôn quan tâm, tận tình hướng dẫn chúng em trong quá trình học tập, nghiên cứu và thực hiện đề tài.

Chúng em xin chân thành cảm ơn quý Thầy Cô trong Khoa Công Nghệ Thông Tin đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quý báu trong suốt quá trình học tập và thực hiện đề tài. Chúng em cũng xin gửi lòng biết ơn đến thầy cô và bạn bè trong lớp đã giúp đỡ, động viên tinh thần chúng em rất nhiều trong suốt quá trình thực hiện luận văn này.

Chúng em nhớ mãi công ơn gia đình đã chăm sóc, động viên và tạo mọi điều kiện thuận lợi cho chúng em hoàn thành tốt khóa luận này.

Mặc dù đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót, kính mong nhận được sự góp ý và tận tình chỉ bảo của quý Thầy Cô và các bạn.

Một lần nữa, xin chân thành cảm ơn và mong luôn nhận được những tình cảm chân thành của tất cả mọi người

TP. Hồ Chí Minh, tháng 6 năm 2018

Huỳnh Tấn Phát – Nguyễn Chí Thành

ĐỀ CƯƠNG CHI TIẾT

Tên Đề Tài: NGHIÊN CỨU VÀ CẢI TIẾN HỆ THỐNG CAMUDA ĐỂ PHÁT TRIỂN NHANH CÁC HỆ THỐNG QUẢN LÝ VỚI BPMN
Giáo viên hướng dẫn: ThS. Ngô Chánh Đức
Thời gian thực hiện: Từ ngày 15/1/2018 đến 18/06/2018
Sinh viên thực hiện: 1412383 – Huỳnh Tấn Phát 1412495 – Nguyễn Chí Thành
Loại đề tài: Nghiên cứu lý thuyết, tìm hiểu và cải tiến công cụ

Nội Dung Đề Tài: Đây là đề tài theo hướng nghiên cứu lý thuyết, tìm hiểu công nghệ và xây dựng ứng dụng. Nội dung đề tài bao gồm:

- Tìm hiểu ngôn ngữ mô hình hóa BPMN
- Tìm hiểu kiến trúc hướng dịch vụ Microservice
- Tìm hiểu các công cụ hỗ trợ tự động hóa các quy trình quản lý
- Tìm hiểu cách cài đặt công cụ Camunda Modeler để tiến hành cải tiến
- Tìm hiểu cách cài đặt một Plugin trên Camunda Platform để Data Extension Plugin
- Tìm hiểu công nghệ Java Web Application để xây dựng Visual Executor
- Cải tiến Camunda Modeler cho phép người dùng có thể thao tác với cơ sở dữ liệu và dịch vụ Spreadsheet của Google
- Xây dựng Data Extension Plugin theo dạng thư viện Java (.jar) và chạy trên môi trường Apache Tomcat
- Xây dựng Visual Executor trên môi trường web

Kế Hoạch Thực Hiện:

- Tháng 1/2018: Nhận khóa luận
- 24/02/2018 – 28/02/2018: Tìm hiểu về BPMN và một số công cụ hỗ trợ xây dựng quy trình.
- 01/03/2018 – 15/03/2018: Tìm hiểu về Camunda (cách thức hoạt động và cách thêm plugin, mở rộng modeler)
- 16/03/2018 – 20/03/2018: Phân tích yêu cầu hệ thống và thiết kế hệ thống
- 21/03/2018 – 28/04/2018: Thử nghiệm kết hợp giữa Camunda Modeler và Plugin với MySQL, bao gồm việc tổ chức thông tin trên tập tin với Modeler, lấy thông tin, tạo kết nối, xây dựng câu SQL và thực thi từ tập tin (.bpmn) với Plugin
- 29/03/2018 – 02/05/2018: Hoàn thiện Plugin và Modeler với MySQL
- 03/05/2018 – 06/05/2018: Hoàn thiện Plugin và Modeler với SQL Server
- 07/03/2018 – 10/05/2018: Hoàn thiện Plugin và Modeler với Oracle SQL
- 11/05/2018 – 18/05/2018: Cài đặt Modeler và Plugin để thực hiện thao tác với Google Spreadsheet.
- 19/05/2018 – 31/05/2018: Tiến hành cài đặt Visual Executor theo những gì đã phân tích.
- 01/06/2018 – 18/06/2018: Viết khóa luận tốt nghiệp
- 19/06/2018: Kết thúc khóa luận tốt nghiệp

Xác nhận của GVHD**Ngày.....tháng.....năm.....****SV Thực hiện
Huỳnh Tấn Phát – Nguyễn Chí Thành**

MỤC LỤC

LỜI CẢM ƠN	i
ĐỀ CƯƠNG CHI TIẾT.....	ii
MỤC LỤC.....	iv
DANH MỤC HÌNH	viii
DANH MỤC BẢNG.....	x
DANH MỤC MÃ NGUỒN	xi
BẢNG CHỮ CÁI VIẾT TẮT	xii
TÓM TẮT KHÓA LUẬN.....	xiii
CHƯƠNG 1: TỔNG QUAN.....	1
1.1. Giới thiệu đề tài.....	1
1.1.1. Lí do thực hiện đề tài.....	1
1.1.2. Mục tiêu đề tài.....	1
CHƯƠNG 2: BPMN - NGÔN NGỮ MÔ HÌNH HÓA NGHIỆP VỤ.....	2
2.1. Thực trạng phát triển các phần mềm quản lí.....	2
2.2. Khái niệm về mô hình hóa và mô hình hóa nghiệp vụ[17].....	2
2.3. Các giải pháp mô hình hóa nghiệp vụ hiện tại	2
2.4. Bối cảnh ra đời của BPMN	4
2.5. Cấu trúc của mô hình BPMN và các vấn đề có liên quan.....	4
2.5.1. Các thành phần trong BPMN [17] [15] [6] [16]	4
2.5.1.1. Participant.....	5
2.5.1.2. Activity.....	5

2.5.1.2.1. Task[4]	6
2.5.1.2.2. Subprocess.....	10
2.5.1.3. Gateway.....	10
2.5.1.4. Events	13
2.5.1.5. Data Object.....	18
2.5.1.6. Flow.....	18
2.5.2. Cấu trúc của mô hình BPMN [9]	19
2.5.3. Ưu khuyết điểm khi sử dụng mô hình BPMN[17] [14]	21
2.5.4. Các vấn đề liên quan	21
CHƯƠNG 3: GIỚI THIỆU VỀ CAMUNDA	23
3.1. Khái niệm về Microservice	23
3.2. Lợi ích và hạn chế khi sử dụng Microservice	24
3.3. Các Engine hỗ trợ thiết kế và triển khai mô hình BPMN với kiến trúc Microservice.....	25
3.4. Giới thiệu về Camunda.....	28
3.4.1. Process engine và Cơ sở hạ tầng	28
3.4.2. Các ứng dụng web mà Camunda cung cấp	28
3.4.3. Các công cụ hỗ trợ:	29
3.5. Các thành phần trong BPMN 2.0 mà Camunda có thể hỗ trợ.....	30
3.6. Các môi trường Camunda hỗ trợ.....	32
3.6.1. Container	32
3.6.2. Cơ sở dữ liệu	33

3.6.3. Web browser:	33
3.6.4. Java.....	33
3.6.5. Java Runtime	33
3.6.6. Camunda Modeler	34
3.7. Tổng quan kiến trúc logic của Process Engine trong Camunda	34
3.8. Một số mô hình triển khai của Camunda	36
3.8.1. Embedded Process Engine	36
3.8.2. Shared, Container-Managed Process Engine	36
3.8.3. Standalone Process Engine	37
3.8.4. Clustering Model.....	38
3.9. Một số hình ảnh giao diện của Camunda:	39
CHƯƠNG 4: GIỚI THIỆU VỀ HỆ THỐNG SAU KHI CẢI TIẾN	41
4.1. Những mong muốn khi cải tiến hệ thống.....	41
4.1.1. Những điểm cần cải tiến trong hệ thống Camunda.....	41
4.1.2. Những tính năng cải tiến	42
4.1.3. Cách tiếp cận trong việc cải tiến	43
CHƯƠNG 5: QUÁ TRÌNH CÀI ĐẶT	44
5.1. Sơ đồ thực hiện tổng quát của hệ thống Camunda sau khi cải tiến	44
5.2. Mở rộng hệ thống mô hình hóa BPMN.....	45
5.2.1. Phương pháp tiếp cận	45
5.2.2. Giới thiệu về Camunda Modeler.....	46
5.2.3. Cấu trúc các tập tin mã nguồn của công cụ Modeler	47

5.2.4. Cách thức tổ chức của công cụ để có thể vẽ mô hình BPMN.....	48
5.2.4.1. Cách tổ chức các thành phần trên client.....	48
5.2.4.2. Cách tổ chức trên server.....	51
5.2.5. Mở rộng mô hình hóa các thao tác trên cơ sở dữ liệu.....	51
5.2.5.1. Mở rộng trên client.....	51
5.2.5.2. Mở rộng phần Server.....	57
5.2.6. Mở rộng các chức năng liên quan đến Google Spreadsheet	58
5.2.6.1. Mở rộng client	58
5.2.6.2. Mở rộng server	59
5.3. Mở rộng hệ thống vận hành quy trình bằng DEP	60
5.3.1. Giới thiệu về Plugin trong Camunda:	60
5.3.2. Phương pháp tiếp cận.....	61
5.3.3. Mô hình kiến trúc của DEP.....	61
5.3.4. Cấu trúc thư mục	63
5.3.5. Cách hoạt động của DEP.....	65
5.4. Thiết kế giao diện cho hệ thống thực thi BPMN	66
5.4.1. Phương pháp tiếp cận.....	66
5.4.2. Các thư viện hỗ trợ.....	67
5.4.3. Mô tả giao diện mới và cách thức thực hiện	68
CHƯƠNG 6: KẾT LUẬN.....	71
DANH MỤC THAM KHẢO	74

DANH MỤC HÌNH

Hình 2.1- Pool và Lane	5
Hình 2.2 – Task	5
Hình 2.3 – Sub-Process	6
Hình 2.4 - Các loại Task trong BPMN.....	6
Hình 2.5 – Sub-Process dạng thu gọn(bên trái) và dạng mở rộng(bên phải).....	10
Hình 2.6 - Các loại Gateway	11
Hình 2.7 - Exclusive Gateway	11
Hình 2.8 - Event-based Gateways	12
Hình 2.9 - Parallel Gateway	12
Hình 2.10 - Inclusive Gateway.....	13
Hình 2.11 – Cách sử dụng Event giữa 2 Activity	17
Hình 2.12 – Cách sử dụng Event trên Activity	17
Hình 3.1 - Mô hình Microservice cho hệ thống đặt xe [1].....	23
Hình 3.1 – Các loại Activity Camunda hỗ trợ [8].....	30
Hình 3.2 – Gateway và các loại khác [8]	32
Hình 3.3 – Kiến trúc của hệ thống Camunda [7]	34
Hình 3.4 – Mô hình Embedded Process Engine [7].....	36
Hình 3.5 – Mô hình Shared, Container-Managed Process Engine [7].....	36
Hình 3.6 – Mô hình Standalone Process Engine [7]	37
Hình 3.7 – Mô hình Standalone Process Engine [7]	38
Hình 3.8 – Giao diện của Camunda Admin	39
Hình 3.9 – Giao diện của Camunda Cockpit	39
Hình 3.10 – Giao diện của Camunda Tasklist	40
Hình 3.11 – Giao diện của Camunda Modeller	40
Hình 5.1 - Sơ đồ thực hiện của hệ thống.....	44

Hình 5.2 - Cấu trúc các thư mục trong Modeler	47
Hình 5.3 - Cách thức tổ chức các module trên client trong modeler	49
Hình 5.4 - Cách tổ chức trên Server.....	51
Bảng 5.1 – Chi tiết các thẻ được thêm vào cấu trúc lưu trữ.....	52
Hình 5.3 – Mô hình Data Extension Plugin (DEP).....	62
Hình 5.4 – Cấu trúc thư mục của DEP.....	63
Hình 5.5 – Luồng xử lí của việc ghi, cập nhật và xóa bỏ dữ liệu	65
Hình 5.6 – Luồng xử lí của việc đọc dữ liệu.....	66
Hình 5.5 – Giao diện đăng nhập.....	68
Hình 5.6 – Giao diện chính	68
Hình 6.1 – Sơ đồ kiến trúc của DEP trong tương lai	73

DANH MỤC BẢNG

Bảng 2.1 – Bảng so sánh giữa BPMN, Flow Chart và Activity Diagram	3
Bảng 2.2 – Các Marker được sử dụng trong Task	9
Bảng 2.3 – Các Event cơ bản trong BPMN	14
Bảng 2.4 – Các Event chi tiết trong BPMN	16
Bảng 2.5 – DataObject trong BPMN	18
Bảng 2.6 – Flow Object trong BPMN.....	19
Bảng 3.1 – Bảng so sánh giữa Camunda và Flowable.....	27
Bảng 3.2 – Cách thành phần sự kiện được cài đặt trong Camunda [8].....	32
Bảng 5.2 – Các thẻ mô tả thông tin google spreadsheet	58
Bảng 5.3 – Các phụ thuộc cơ bản của DEP	64
Bảng 5.4 – Các tài nguyên truy cập	67
Bảng 5.5 – Thao tác trên các tài nguyên	67

DANH MỤC MÃ NGUỒN

Mã nguồn 5.1 – Thêm một Tab mới	53
Mã nguồn 5.2 – Thao tác trên DataStoreReference	53
Mã nguồn 5.3 – Thêm loại cơ sở dữ liệu	54
Mã nguồn 5.4 – Thêm tên truy cập và mật khẩu truy cập.....	55
Mã nguồn 5.5 – Phương thức Get và Set	56
Mã nguồn 5.6 – Gửi thông điệp từ Client đến Server.....	57
Mã nguồn 5.7 - Xử lý thông tin trên Server.....	57
Mã nguồn 5.8 - Tập tin bpm-platform.xml	60

BẢNG CHỮ CÁI VIẾT TẮT



Chữ viết tắt	Chữ viết đầy đủ
BPMN	Bussiness Process Modeling Notation
BPEL	Business Process Execution Language
OMG	Object Management Group
BPMI	Business Process Management Initiative
UML	Unified Modeling Language
SQL	Structured Query Language
DEP	Data Extension Plugin
JDBC	Java DataBase Connectivity
API	Application Programming Interface

TÓM TẮT KHÓA LUẬN

Các vấn đề nghiên cứu trong khóa luận bao gồm:

- Tìm hiểu về ngôn ngữ mô hình hóa BPMN
- Tìm hiểu các công cụ có sẵn cho phép thực thi mô hình BPMN trên kiến trúc Microservice
- Nghiên cứu các ưu khuyết điểm trên các công cụ hiện đang phát triển từ đó đưa ra giải pháp cải tiến cho một trong số các công cụ đó
- Cải tiến và bổ sung việc mô hình hóa và việc thực thi các quy trình dựa trên công cụ có sẵn
- Xây dựng một ứng dụng đơn giản bằng việc dùng các công cụ sau khi đã cải tiến
- Đánh giá khả năng phát triển của các công cụ sau khi cải tiến trong tương lai

Kết quả đạt được

Nội dung khóa luận gồm có 6 chương

Chương 1: Trình bày những khó khăn trong việc phát triển một hệ thống phần mềm quản lý hiện nay. Từ những thực trạng đó đưa ra lí do và mục tiêu mong muốn khi thực hiện đề tài.

Chương 2: Giới thiệu mô hình hóa nghiệp vụ BPMN cũng như cấu trúc của một mô hình BPMN, các thành phần chi tiết trong BPMN. Đưa ra những dẫn chứng về lí do chọn BPMN trong việc phát triển nhanh các hệ thống quản lí.

Chương 3: Tóm tắt sơ lược về Microservice, khảo sát các cộng cụ hỗ trợ BPMN hiện có, phân tích ưu khuyết điểm, từ đó lựa chọn một công cụ hiện có để cải tiến, giới

thiệu về Camunda, các thành phần được cài đặt sẵn để tự động hóa quy trình, môi trường cài đặt, ngôn ngữ và các kiểu kiến trúc của Camunda

Chương 4: Giới thiệu về hệ thống sau khi cải tiến, những điểm mong muốn cải tiến, các tính năng đã cải tiến và cách tiếp cận trong việc cải tiến

Chương 5: Mô tả quy trình, cách thiết kế và cài đặt cho hệ thống

Chương 6: Đánh giá kết quả thu được sau khi cài đặt hoàn thiện hệ thống Camunda sau khi cải tiến và định hướng phát triển cho hệ thống trong tương lai

CHƯƠNG 1: TỔNG QUAN

Trong chương này, chúng em xin trình bày về những lí do thực hiện đề tài và mục tiêu đề tài hướng đến.

1.1. Giới thiệu đề tài

1.1.1. Lí do thực hiện đề tài.

Trong thời kì công nghệ phát triển nhanh và tự động hiện nay, việc các doanh nghiệp có nhu cầu giảm thiểu thời gian, chi phí và nhân lực để phát triển các hệ thống quản lí cho riêng mình là điều không thể tránh khỏi. Đồng thời phát sinh thêm nhu cầu về việc tự phát triển và tự triển khai các quy trình, hệ thống nội bộ, kết quả là nhu cầu cần có một công cụ có thể hỗ trợ việc thiết kế, tự động hóa một phần việc thực thi quy trình trở nên ngày càng thiết yếu.

Nhưng điều khó khăn ở đây vẫn là khả năng tiếp cận việc cài đặt đối với các doanh nghiệp còn rất hạn chế nên yêu cầu của hệ thống hỗ trợ phải trực quan, tiện dụng, dễ sử dụng và tối ưu hóa khả năng tự động.

1.1.2. Mục tiêu đề tài

Mục tiêu đề tài lần này chúng em hướng đến việc nâng cấp một trong các công cụ hỗ trợ thiết kế, cài đặt và triển khai BPMN phổ biến hiện nay, đó chính là Camunda. Chúng em đề xuất cài đặt thêm thành phần **Data Store Reference Element** của BPMN dựa trên **Form Data** của Camunda, đồng thời đề xuất cài đặt thêm **Data Object Reference Element** của BPMN cũng dựa trên **Form Data** của Camunda.

Lý do chúng em chọn đề xuất cài đặt cho các thành phần này là do sau khi tìm hiểu Camunda cũng như các hệ thống tương tự, chúng em nhận thấy các hệ thống này đều chưa có hỗ trợ sử dụng cơ sở dữ liệu cũng như là các hình thức lưu trữ thông tin cho các nghiệp vụ. Điều đó làm cho việc truy vấn và quản lý thông tin của quy trình trở nên khó khăn, gần như là không thể, nên chúng em quyết định bổ sung thêm các thành phần trên để tăng khả năng lưu trữ thông tin.

CHƯƠNG 2: BPMN - NGÔN NGỮ MÔ HÌNH HÓA NGHIỆP VỤ

Trong chương này, chúng em xin trình bày một số khái niệm về ngôn ngữ mô hình hóa nghiệp vụ, lý do sử dụng BPMN cùng với một số thành phần chính của BPMN, những lợi ích mà BPMN mang lại

2.1. Thực trạng phát triển các phần mềm quản lý.

Ngày nay, trong thời đại công nghệ, nhiều tổ chức ra đời cùng với sự phát triển của khoa học và công nghệ hiện đại đòi hỏi của các hệ thống quản lý ngày càng tăng cao để có thể đáp ứng nhu cầu phát triển của doanh nghiệp. Bên cạnh đó, với phương pháp phát triển phần mềm hiện tại, cùng với sự mở rộng về quy mô, tính chất của các tổ chức doanh nghiệp cũng đòi hỏi việc quản lý trở nên phức tạp hơn dẫn đến chi phí dùng cho bảo trì và nâng cấp hệ thống cũng trở nên cao hơn.

Để đáp ứng được nhu cầu đó, đòi hỏi cần phải có một hệ thống hỗ trợ thiết kế được quy trình nghiệp vụ của tổ chức, đồng thời có thể chuyển được mô hình đó thành một ngôn ngữ có thể thực thi được để tạo ra được một hệ thống quản lý.

2.2. Khái niệm về mô hình hóa và mô hình hóa nghiệp vụ[17]

Mô hình hóa là việc mô tả lại một sự vật hay hiện tượng một cách trừu tượng. Trong đó, mô hình hóa nghiệp vụ là việc mô tả lại chuỗi các hoạt động trong một quy trình nghiệp vụ bằng các kí hiệu

Tại sao cần thiết phải mô hình hóa nghiệp vụ? Trong quá trình phát triển hệ thống, mô hình hóa nghiệp vụ giúp cho các bên liên quan hiểu rõ về một nghiệp vụ do việc mô hình hóa được mô tả một cách trực quan, rõ ràng. Đồng thời, việc mô hình hóa còn là cơ sở dùng để nâng cấp hệ thống trong tương lai.

2.3. Các giải pháp mô hình hóa nghiệp vụ hiện tại

Hiện nay, có rất nhiều mô hình có thể hỗ trợ cho việc phát triển và triển khai nhanh một hệ thống quản lý, trong đó các mô hình thuộc kỹ thuật Flowcharting là các mô hình có tính phổ biến cao tại thời điểm hiện tại. Các mô hình có thông dụng có 3 mô hình tiêu

biểu có thể kể ra bao gồm: BPMN, Flow Chart, Activity Diagram. Sau khi tìm hiểu và phân tích các mô hình với nhau chúng em đã đưa ra bảng so sánh như sau:

	BPMN	Flow Chart	Activity Diagram
Giống nhau	Đều là mô hình đồ họa biểu diễn một quy trình nghiệp vụ trong một mô hình nghiệp vụ cụ thể.		
Khác nhau	<p>Các kí hiệu có tiêu chuẩn rõ ràng và đa dạng, phù hợp để mô tả cụ thể các quy trình phức tạp, cần nhiều xử lí. [3]</p> <p>Hỗ trợ chuyển đổi được sang ngôn ngữ thực thi BPEL [2]</p>	<p>Các kí hiệu tương đối đơn giản, dễ hình dung nhưng không quá đa dạng về thành phần gây khó khăn cho việc đặc tả các quy trình có các xử lí đặc biệt (có triggers, events, ...) [13] [11]</p>	<p>Các kí hiệu cũng tương đối đầy đủ như BPMN. [3]</p> <p>Không hỗ trợ chuyển đổi được sang ngôn ngữ thực thi BPEL [2]</p>

Bảng 2.1 – Bảng so sánh giữa BPMN, Flow Chart và Activity Diagram

❖ Tại sao chọn BPMN

Như phân tích bên trên, **BPMN** và **Activity Diagram** có thể dễ hiểu hơn so với các đối tượng không chuyên về công nghệ thông tin nên việc chọn **BPMN** sẽ giúp cho quá trình trao đổi giữa bộ phận phát triển và doanh nghiệp trở nên dễ dàng hơn, từ đó rút ngắn thời gian phân tích, thu thập yêu cầu, tăng thời lượng cài đặt và triển khai, hơn nữa còn giảm thiểu sự sai sót do hiểu nhầm ý nghĩa biểu tượng hoặc không hiểu tường tận quy trình. Bên cạnh đó, **BPMN** có hỗ trợ chuyển đổi sang ngôn ngữ thực thi, mấu chốt quan trọng trong việc phát triển nhanh hệ thống quản lý. Đồng thời các công cụ hỗ trợ việc xây dựng, cài đặt và triển khai các quy trình dựa trên **BPMN** phổ biến hơn so với các mô hình còn lại.

2.4. Bối cảnh ra đời của BPMN

Khi mô hình hóa một nghiệp vụ nào đó sẽ gặp nhiều khó khăn, trong đó khó khăn lớn nhất là không đồng nhất giữa 2 bên – người làm nghiệp vụ và người làm kỹ thuật: mô hình hóa theo cách của một trong 2 bên thì bên còn lại sẽ gặp nhiều khó khăn dẫn đến sự thống nhất về suy nghĩ của 2 bên khi phát triển một hệ thống sẽ gặp không ít khó khăn. Ngoài ra, việc mô hình hóa một nghiệp vụ trong thực tế cũng gặp không ít khó khăn khi một nghiệp vụ rất phức tạp và không đủ những ký hiệu để có thể đáp ứng được việc mô tả nghiệp vụ đó dưới dạng một mô hình.[17]

Cách sử dụng phổ biến hiện nay là có một người đóng vai trò cầu nối giữa người làm nghiệp vụ và kỹ thuật, người này có trách nhiệm chuyển đổi thông tin giao tiếp giữa 2 bên. Tuy nhiên, cần có giải pháp để giúp nhân viên có thể mô hình hóa lại một nghiệp vụ mà cả 2 bên có thể hiểu được: phía khách hàng có thể hiểu được nghiệp vụ của mình trên mô hình có đúng như thực tại hay không, còn phía lập trình viên vẫn có thể dựa vào mô hình để phát triển hệ thống. Từ nhu cầu này dẫn đến một giải pháp đó là BPMN. Vậy BPMN là gì?[17]

BPMN(Business Process Model and Notation) là một tiêu chuẩn của mô hình hóa nghiệp vụ, cung cấp một bộ các ký hiệu đồ họa dùng để đặc tả một quy trình nghiệp vụ dựa vào UML Flowchart. Ngoài ra, BPMN còn cung cấp một cơ chế thực thi quy trình nghiệp vụ dựa vào BPEL từ một mô hình nghiệp vụ ở mức ký hiệu. [15]

2.5. Cấu trúc của mô hình BPMN và các vấn đề có liên quan

2.5.1. Các thành phần trong BPMN [17] [15] [6] [16]

Trong phần này, chúng em xin trình bày một số khái niệm về các thành phần chính trong một mô hình BPMN và ý nghĩa của các thành phần đó.

Có 6 loại thành phần cơ bản trong một mô hình BPMN đó là Participant, Activity, Event và Gateway, Data Object và Flow.

2.5.1.1. Participant

Participant là thành phần được sử dụng để đại diện cho những bên liên quan đến một quy trình nghiệp vụ đang được mô tả. Trong BPMN, Participant gồm có 2 phần, đó là **Pool** và **Lane**



Hình 2.1- Pool và Lane

Hình 2.1 mô tả cách thức tổ chức của một Participant trong BPMN, trong đó **Pool** được sử dụng để đại diện cho một quy trình của một tổ chức, mặt khác, **Lane** được sử dụng để mô tả hoạt động của một bộ phận trong tổ chức đó, do đó **Lane** thường được đặt trong **Pool** và có nhiều **Lane** khác nhau trong một **Pool**

2.5.1.2. Activity

Activity là thành phần dùng để mô tả các hoạt động có thể xảy ra trong một quy trình, các hoạt động này bao gồm 2 loại:

Task: là hoạt động không thể chia nhỏ được nữa



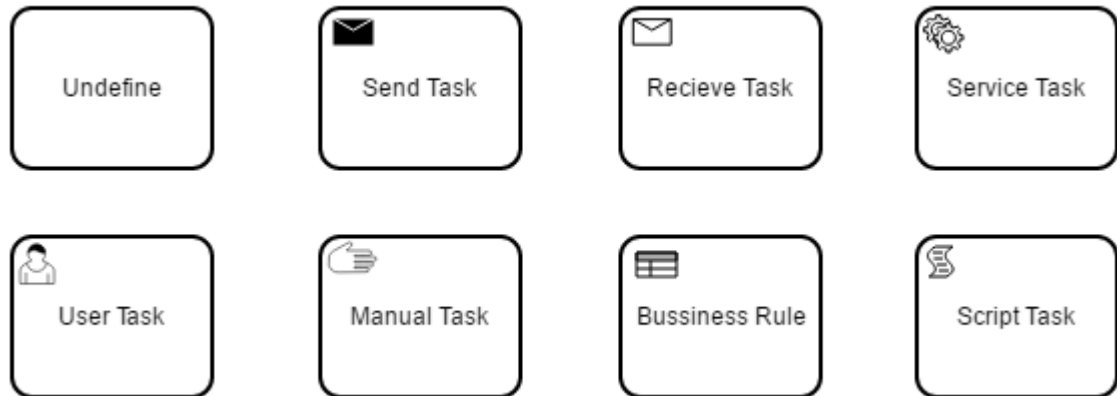
Hình 2.2 – Task

Sub-Process: Là hoạt động có thể chia nhỏ thành các hoạt động nhỏ hơn



Hình 2.3 – Sub-Process

2.5.1.2.1. Task[4]



Hình 2.4 - Các loại Task trong BPMN

Thành phần này dùng để mô tả các tác vụ có trong hệ thống ở mức cơ bản nhất . Thông thường, trong BPMN chúng ta chỉ sử dụng loại Task không xác định (Undefine Task) để biểu diễn các tác vụ. Tuy nhiên ở mức độ chi tiết, BPMN cung cấp nhiều loại Task khác nhau nhằm hỗ trợ cho việc biểu diễn các công việc được thực thi một cách chuyên biệt. Các loại task được cung cấp có tác dụng khác nhau tùy vào mục đích sử dụng và ý nghĩa của nghiệp vụ đó trong thực tế. **Hình 2.4** mô tả các loại Task có trong BPMN, các loại task này bao gồm

- **User Task**

User Task được sử dụng phổ biến trong một quy trình BPMN vì nó đại diện cho một tác vụ điển hình trong một quy trình làm việc. User Task là một tác vụ được thực thi bởi một người và có sự hỗ trợ thông qua giao diện của ứng dụng mà mô hình BPMN đang mô tả nó (ví dụ: một tác vụ được gọi là User Task trong một *Hệ thống quản lý thư*

viện chỉ khi tác vụ này được thực hiện dưới sự hỗ trợ của chính *Hệ thống quản lý thư viện* đó) trong trường hợp các tác vụ này được hoàn thành dưới sự hỗ trợ của một hệ thống độc lập khác (không phải *Hệ thống quản lý thư viện*) thì các tác vụ này được xem là **Manual Task**.

- **Manual Task**

Là loại tác vụ yêu cầu được hoàn thành bởi một người. Ngược lại với User Task, Manual Task là một tác vụ được dự kiến sẽ hoàn thành mà không cần có sự hỗ trợ của hệ thống (giả sử là *Hệ thống quản lý thư viện*). Manual Task không cung cấp giao diện thực hiện cho người mô tả quy trình bởi nó vốn không được thực thi trong hệ thống đang được mô tả.

- **Service Task**

Ngược lại với User Task và Manual Task, Service Task không yêu cầu sự tương tác của người dùng và tác vụ này được gọi là **Automated Task**. Service Task là một tác vụ được thực hiện hoàn thành tự động thông qua các hàm được thêm vào trong quá trình thực thi. Những hàm này thường được BPMN giả định là một dịch vụ web, tuy nhiên những hàm này có nhiều cách triển khai khác nhau (ví dụ: có thể triển khai bằng một Java Class Delegate hoặc bằng một đoạn JavaScript)

- **Script Task**

Script Task là một loại **Automated Task**. Trái ngược với Service Task, Script Task là tác vụ được thực hiện bởi chính công cụ thực thi mô hình nghiệp vụ. Về kỹ thuật, Script Task là một đoạn mã được thực thi bởi công cụ thực thi mô hình nghiệp vụ, người lập mô hình định nghĩa một tập các mã lệnh thành một kịch bản trên các ngôn ngữ mà những công cụ này hỗ trợ. Khi các tác vụ này thực thi, các kịch bản được định nghĩa sẽ được thực thi, khi kịch bản này kết thúc thì tác vụ này cũng được hoàn thành.

- **Send Task**

Là các tác vụ BPMN đơn giản được thiết kế để gửi thông điệp đến những **Participant** khác (thường là khác **Pool**), khi các thông điệp được gửi đi thì các tác vụ này cũng được hoàn thành



- **Recieve Task**




Trái ngược với Send Task, Receive Task là một tác vụ được thiết kế để chờ một thông điệp nào đó từ **Participant** bên ngoài (thường là một **Pool** khác). Khi thông điệp được nhận, tác vụ này cũng được hoàn thành

- **Bussiness Rule**

Mô tả những quy tắc nghiệp vụ dùng để tính toán kết quả theo một nghiệp vụ nào đó

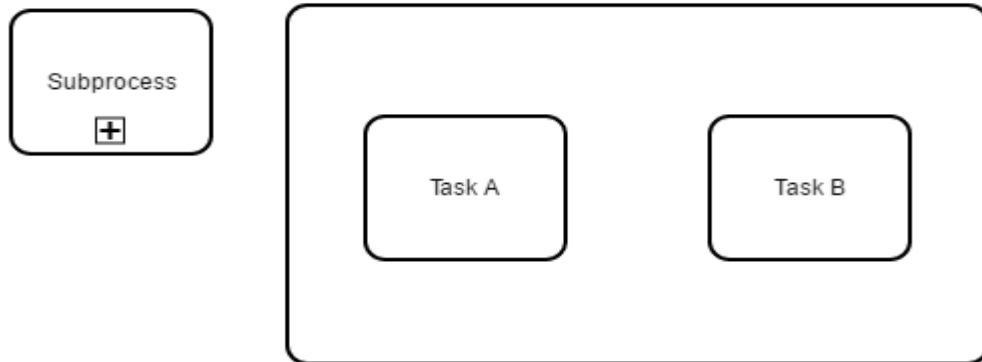
Ngoài một số loại Task trên, BPMN còn cho phép đặt một số Marker giúp mô tả rõ hơn ngữ nghĩa khi mô hình hóa một quy trình nghiệp vụ, các Marker này được mô tả trong Bảng 2.2 bao gồm Loop, Multiple Instance, Compensation, Adhoc, Annotation.

Kí hiệu	Ý nghĩa
 Loop Marker	Được sử dụng đại diện cho một hoạt động được chạy nhiều lần cho đến khi điều kiện thỏa mãn, các điều kiện có thể được kiểm tra ở khi bắt đầu hoặc khi kết thúc hoạt động
 Parallel Multiple Instance Marker	Đại diện cho một hoạt động được thực thi với nhiều Instance song song. Số lượng Instance được xác định thông qua điều kiện ở đầu mỗi hoạt động. Tất cả các Instance được chạy song song và có thể có tham số đầu vào khác nhau. Hoạt động được xem như hoàn thành

	<p>khi tất cả các Instance được hoàn thành. Tuy nhiên, trong một số trường hợp, hoạt động này có thể dừng khi thỏa mãn một điều kiện xác định</p>
 Sequential Multiple Instance Marker	<p>Tương tự như Parallel Multiple Instance nhưng các Instance này được thực hiện tuần tự.</p>
 Adhoc Marker	<p>Marker này có tác dụng đánh dấu cách thức thực thi của các hoạt động trong một Sub-Process có thể là:</p> <ul style="list-style-type: none"> • Thực thi theo bất kì thứ tự nào. • Thực thi nhiều lần. • Bỏ qua. <p>Ngoài ra, trong một mô hình BPMN còn có những quy định về mức độ cần thiết của các thành phần có trong Adhoc Sub-Process, trong đó:</p> <ul style="list-style-type: none"> • Bắt buộc phải có: Activity • Không bắt buộc: Data Objects, Sequence Flows, Associations, Groups, Message Flows, Gateways và Intermediate Events. • Không được phép có: Start và End Event
 Annotation Marker	<p>Là một cơ chế cho các công cụ mô hình hóa cung cấp thêm những thông tin của những thành phần được mô tả nhằm giúp cho người đọc dễ dàng đọc được mô hình BPMN</p>

Bảng 2.2 – Các Marker được sử dụng trong Task

2.5.1.2.2. Subprocess



Hình 2.5 – Sub-Process dạng thu gọn(bên trái) và dạng mở rộng(bên phải)

Trong một môi trường làm việc, mô hình BPMN được sử dụng để truyền đạt mô tả của một quy trình cho các bên liên quan và nhà phát triển trong việc phát triển hệ thống. Tuy nhiên, các bên liên quan thường không muốn sự phức tạp của một mô hình, việc mô tả phức tạp và chi tiết của quy trình chỉ được quan tâm bởi các nhà phát triển. Vì vậy, BPMN cung cấp Sub-Process (**Hình 2.5**) dùng để mô tả một chuỗi các hoạt động chi tiết nhưng không mất nhiều không gian trên lược đồ, cung cấp một cái nhìn tổng quan về hoạt động chính của một quy trình. Giúp người sử dụng có thể mở rộng hoặc thu gọn quy trình để có thể truyền đạt cho cả người phát triển cũng như các bên liên quan đến hệ thống

2.5.1.3. Gateway

Gateway là thành phần điều khiển, có tác dụng để trộn hoặc phân chia các luồng thực thi. Vì vậy nó sẽ quyết định việc rẽ nhánh, trộn các luồng thực thi tùy vào loại được chỉ định.

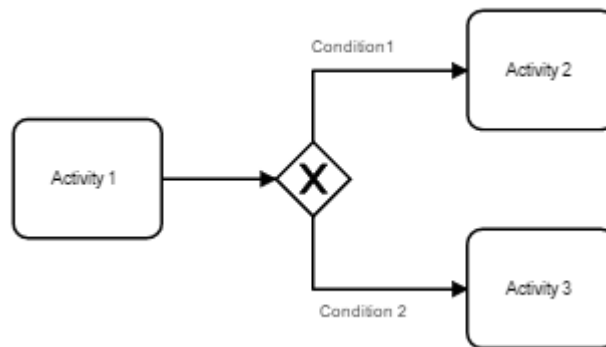


Hình 2.6 - Các loại Gateway

Hình 2.6 mô tả các loại Gateway được sử dụng trong BPMN, đó là:

❖ Exclusive Gateway

Một hoạt động được thực thi tại một thời điểm dựa vào điều kiện logic tại Gateway. Điều kiện tại Exclusive Gateway là **Data-based** tức là nó sử dụng dữ liệu làm điều kiện phân luồng tại Gateway.

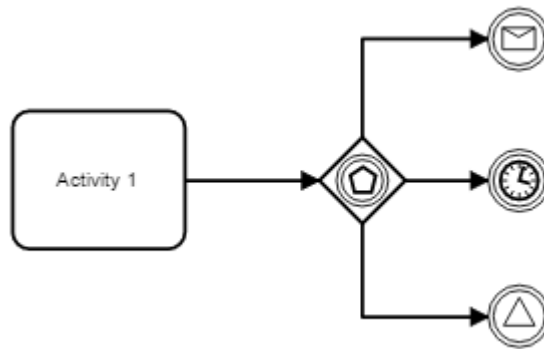


Hình 2.7 - Exclusive Gateway

Hình 2.7 mô tả cách hoạt động của Exclusive Gateway, tại một thời điểm, với Condition1 đúng thì sẽ thực hiện Activity2 và Condition2 đúng thì sẽ thực hiện Activity3

❖ Event-based Gateways

Tương tự như Exclusive Gateway nhưng điều kiện được xác định tại đây là **Event-based** tức là dựa vào sự kiện để phân luồng tại Gateway.

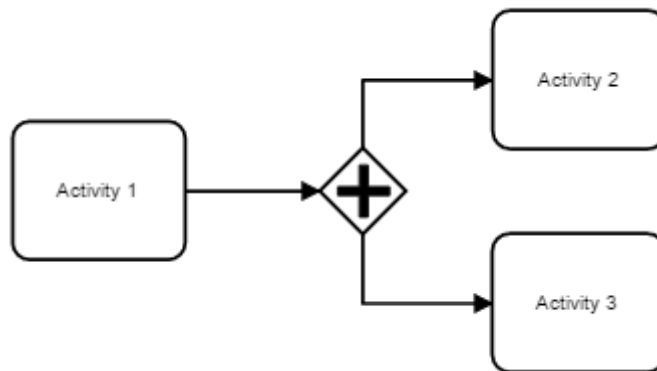


Hình 2.8 - Event-based Gateways

Trong **Hình 2.8**, nếu Timer được kích hoạt thì quy trình sẽ thực hiện các hoạt động trên nhánh có chứa Timer, tương tự cho các sự kiện khác, nếu sự kiện nào xảy ra thì quy trình sẽ thực hiện luồng hoạt động trên nhánh đó.

❖ Parallel Gateway

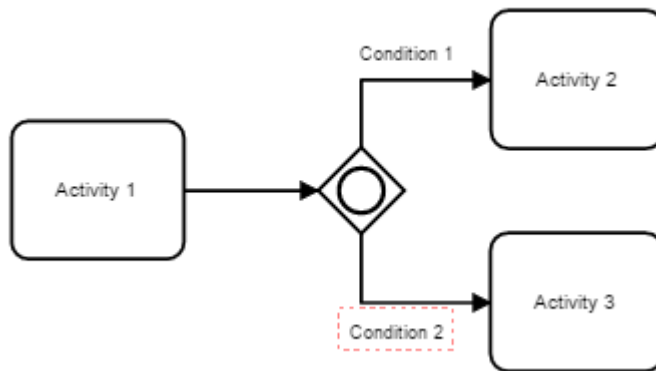
Các luồng được thực thi song song, quy trình được tiếp tục khi tất các hoạt động trong các luồng hoàn tất. Trong ví dụ **Hình 2.9**, Activity 2 và Activity 3 được thực hiện song song.



Hình 2.9 - Parallel Gateway

❖ Inclusive Gateways

Một vài luồng sẽ được thực thi, Gateway này bao hàm chức năng của Exclusive và Parallel Gateway

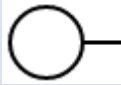




Hình 2.10 - Inclusive Gateway

Trong **Hình 2.10**, tại một thời điểm Activity 2 và Activity 3 có thể chỉ có một hoạt động được thực hiện hoặc cả 2 hoạt động cùng được thực hiện dựa vào điều kiện

2.5.1.4. Events

Task và Gateway là 2 thành phần quan trọng trong một quy trình nghiệp vụ, một task phải được thực hiện trong một số trường hợp nào đó (cần thiết phải dùng Gateway). Ngoài ra, trong một quy trình cần phải có các sự kiện, các sự kiện này cũng có phần quan trọng không kém vì trong một quy trình luôn luôn có những sự kiện xảy ra và chúng ta cần có nhu cầu bắt những sự kiện này. Trong BPMN có 3 loại Event cơ bản được mô tả trong **Bảng 2.3** sau


Start Event		Intermediate Event		End Event	
	Là những sự kiện khi bắt đầu một quy trình, được mô tả bằng một vòng tròn đơn		Là những sự kiện xảy ra trong khi thực thi một quy trình, được mô tả bằng vòng tròn kép		Là những sự kiện xảy ra khi kết thúc quá trình thực thi một quy trình và được mô tả bằng vòng tròn in đậm

Bảng 2.3 – Các Event cơ bản trong BPMN

Khi chúng ta có nhu cầu mô tả một quy trình phức tạp hơn đòi hỏi cần có nhiều sự kiện khác nhau để mô tả các hành động phức tạp trong một quy trình, BPMN cũng cung cấp một bộ các sự kiện phức tạp dùng để đặc tả những Trigger dùng trong hệ thống. Các Event này sẽ đặt một số ràng buộc nhất định vào quy trình mà chúng ta mô tả và đều thuộc 2 dạng sau:


❖ Throwing Events

Sử dụng để gửi những sự kiện trong quá trình thực thi một quy trình.





Throwing Event được mô tả bằng hình vẽ có tô đen (ví dụ )











❖ Catching Events





















Sử dụng để bắt những sự kiện xảy ra trong quá trình thực thi một quy trình

Catch Event được mô tả bằng một hình vẽ chỉ có viền (ví dụ )

Các loại Event này được mô tả trong **Bảng 2.4**

Loại	Start Events	Intermediate Events		End Events	Mô tả
	Catch	Catch	Throw	Throw	
Message					Start Message là loại sự kiện nhận thông điệp từ một Participant khác và kích hoạt một quy trình thực thi, hoặc tiếp tục thực hiện quy trình

					với sự kiện Intermediate Event. End Message biểu thị một thông điệp được tạo ra khi kết thúc một quy trình
Timer					Đặc tả một khoảng thời gian hoặc một chu kỳ cụ thể dùng để bắt đầu một quy trình (Timer Start) hoặc tiếp tục thực thi quy trình (Intermediate Timer). Điều này lý giải tại sao không có End Timer vì khi một quy trình kết thúc thì Timer không thể nào chạy để kích hoạt những luồng hoạt động tiếp theo
Conditional					Sự kiện này sẽ được kích hoạt để bắt đầu một quy trình (Start Event) hoặc tiếp tục thực thi quy trình (Intermediate Event) khi một điều kiện nào đó đúng.
Escalation					Dùng để giao tiếp giữa quy trình chính (Main Process) và quy trình phụ (Sub Process)
Link					Là một trường hợp đặc biệt, nó không có ý nghĩa liên quan đến nội dung. Đây là sự kiện được dùng trong trường hợp chúng ta mô tả một mô hình trong nhiều trang và không có không gian mô tả thì chúng ta sử dụng một Link đánh dấu điểm kết thúc và một Link đánh dấu điểm bắt đầu có tác dụng nối 2 điểm này với nhau thành một quy trình khép kín. Thành phần này có tác dụng hướng người đọc từ trang này sang trang khác.

Error					Được sử dụng để ghi lại lỗi và xử lý chúng.
Cancel					Sự kiện này nghĩa là người dùng đã lựa chọn hủy thực thi một quy trình
Compensation					Được sử dụng để mô tả việc một quy trình được quay lại trạng thái trước đó trong lúc thực
Signal					Được sử dụng để gửi nhận tín hiệu giữa các Pool (cùng hoặc khác Participant) hoặc giữa các quy trình với nhau
Multiple					Dùng để tóm tắt các loại sự kiện có trong một mô hình và được kích hoạt khi một trong số các loại này thỏa mãn điều kiện
Parallel Multiple					Dùng để tóm tắt các loại sự kiện có trong một mô hình nhưng chỉ được kích hoạt khi tất cả các loại này thỏa mãn điều kiện
Terminate					Là sự kiện dừng tất cả. Khi nhận được Event này, toàn bộ quy trình sẽ được dừng lại

Bảng 2.4 – Các Event chi tiết trong BPMN

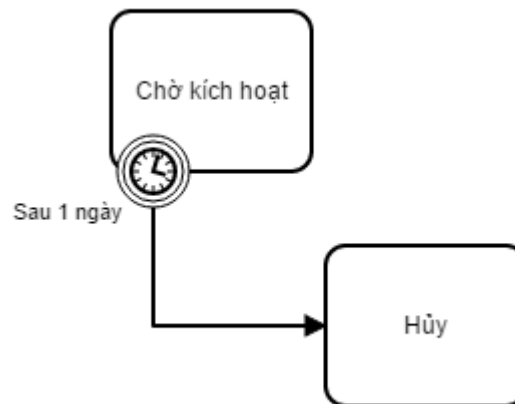
Các Event mô tả trong **Bảng 2.4** có 2 cách sử dụng:

- Đặt giữa 2 Activity có tác dụng như một điều kiện để thực hiện hoạt động tiếp theo



Hình 2.11 – Cách sử dụng Event giữa 2 Activity






- Gắn vào một Activity được sử dụng trong việc quản lý lỗi



Hình 2.12 – Cách sử dụng Event trên Activity

2.5.1.5. Data Object





Trong BPMN, Data Object dùng để mô tả những thành phần trong một mô hình nghiệp vụ có liên quan đến dữ liệu, Data Object bao gồm các loại trong **Bảng 2.5** như sau:

Kí hiệu	Ý nghĩa
 Data Object	Được sử dụng đại diện cho dữ liệu đầu vào hoặc đầu ra của một quy trình
 Data Input	Đại diện cho dữ liệu đầu vào từ các nguồn bên ngoài cho toàn bộ quy trình. Thành phần này được xem như một tham số đầu vào của quy trình
 Data Output	Là dữ liệu kết quả của toàn bộ quy trình, là tham số đầu ra của một quy trình
 Data Store	Là nơi lưu trữ dữ liệu của một quy trình.
 Collection of Data Objects	Chứa tập hợp một bộ các dữ liệu của toàn bộ quy trình

Bảng 2.5 – DataObject trong BPMN

2.5.1.6. Flow

Trong BPMN, Flow dùng để thể hiện luồng thực thi của một quy trình nghiệp vụ, một tập hợp các Flow trong mô hình mô tả cho thứ tự thực hiện các công việc trong một mô hình nghiệp vụ từ lúc bắt đầu cho đến khi quy trình đó kết thúc. **Bảng 2.6** mô tả các loại Flow được BPMN cung cấp

Kí hiệu	Ý nghĩa
 Sequence Flow	Được sử dụng để mô tả thứ tự thực hiện của các hoạt động trong một quy trình nghiệp vụ
 Message Flow	Được sử dụng để mô tả luồng thông điệp giữa các Participant riêng biệt, gửi và nhận thông điệp
 Association	Được sử dụng để liên kết giữa các Marker Anotation với các thành phần khác trong mô hình
 Data Association	Được sử dụng để mô tả đầu vào và đầu ra của một Activity (thường được kết hợp với Data Object)

Bảng 2.6 – Flow Object trong BPMN

2.5.2. Cấu trúc của mô hình BPMN [9]

Ở phần trước, chúng em đã trình bày về một số các thành phần chính trong một mô hình BPMN. Ở phần này chúng em sẽ mô tả về cấu trúc của một mô hình BPMN, cách tổ chức các thành phần trong BPMN.

Một quy trình được mô hình hóa bằng BPMN đó là một tài liệu mô tả về quy trình đó dưới dạng tập tin xml, trong một quy trình BPMN gồm có:

Definition: mọi quy trình được mô hình hóa dưới dạng BPMN đều được bắt đầu bằng thẻ *definitions*, thẻ này chứa các mô tả về thông tin của tài liệu mô hình cùng với những namespace liên quan

Process: là thẻ dùng để chứa các thông tin một cách tổng quan nhất về quy trình nghiệp vụ được mô tả.

Start và End event: là vị trí bắt đầu và kết thúc của một quy trình nghiệp vụ

Components: là các thành phần trong mô hình BPMN dùng để mô tả lại nghiệp vụ của một tổ chức nào đó, các thành phần này chúng em đã trình bày ở phần trước.

Sequence Flow: thẻ Sequence Flow chỉ sự liên kết giữa các thành phần trong một mô hình BPMN, trong thẻ này chỉ rõ nguồn và đích tức là thẻ này dùng để chỉ thứ tự thực hiện của các thành phần với nhau

Incoming và Outcoming: chỉ ra thông tin liên kết giữa thành phần BPMN với sequence flow. Vậy câu hỏi đặt ra là trong sequence flow có chứa liên kết giữa các thành phần trong BPMN, có cần thiết phải có thông tin mô tả liên kết giữa thành phần đó với sequence flow hay không? Câu trả lời là có. Trong một số trường hợp thực thi một quy trình, tại một thời điểm chúng ta cần có nhu cầu trở về trạng thái của bước trước đó, vậy nếu như ở bước hiện tại chúng ta không có một thông tin liên kết gì thì việc truy xuất thông tin liên kết của các bước vô cùng khó khăn. Incoming, outcoming kết hợp với sequence flow tạo nên sự đơn giản trong việc xác định thứ tự chạy của các thành phần trong BPMN.

ExtensionElement: Dùng cho việc mở rộng BPMN mà không làm thay đổi cấu trúc BPMN cũ.

BPMNDiagram: thẻ này dùng để lưu trữ những thông tin liên quan về đồ họa, cho phép người dùng có cái nhìn trực quan về một quy trình dựa trên các công cụ vẽ và hiển thị mô hình BPMN

2.5.3. Ưu khuyết điểm khi sử dụng mô hình BPMN[17] [14]

❖ Ưu điểm

- Hỗ trợ việc quản lí các quy trình nghiệp vụ.
- Cung cấp một cách trực quan và dễ dàng cho người không có chuyên môn về BPMN vẫn có thể hiểu được kí hiệu.
- Giúp người cài đặt hệ thống có được cái nhìn về hệ thống mà họ sẽ triển khai trong tương lai.
- Mô tả một cái nhìn tổng quan về một quy trình phức tạp bằng một hình thức rõ ràng và dễ hiểu.
- Tạo điều kiện trong việc phân tích của người quản lí, phân tích kinh doanh,...

❖ Khuyết điểm

- Có thể có sự mơ hồ trong việc chia sẻ các mô hình BPMN, ví dụ: một quy trình có thể được mô tả với nhiều biến thể kí hiệu khác nhau
- Không có một tiêu chuẩn chung dùng để trao đổi giữa các công cụ BPMN (có chuẩn chung cho BPMN nguyên thủy tuy nhiên các công cụ hiện nay tích hợp và cải tiến nhưng lại không có một chuẩn chung nào quy định khi cải tiến hoặc thêm mới những thành phần nào đó)

2.5.4. Các vấn đề liên quan

Trong quá trình phát triển phần mềm, đặc biệt là các hệ thống quản lí, một số giải pháp thường dùng BPMN để mô hình hóa lại nghiệp vụ của một tổ chức nào đó với mục đích chính là giúp cho người dùng có cái nhìn tổng quan về nghiệp vụ của họ cũng như hệ thống mà những người cung cấp dịch vụ sắp phát triển, đồng thời, mô hình này cũng xem như một tài liệu giúp người phát triển hệ thống có thể xây dựng được một hệ thống

hoàn chỉnh, như vậy vấn đề đặt ra ở đây là nếu như chúng ta mô hình hóa lại một nghiệp vụ bằng BPMN thì liệu rằng có giải pháp nào đưa mô hình này vào hệ thống và có khả năng chạy được không?

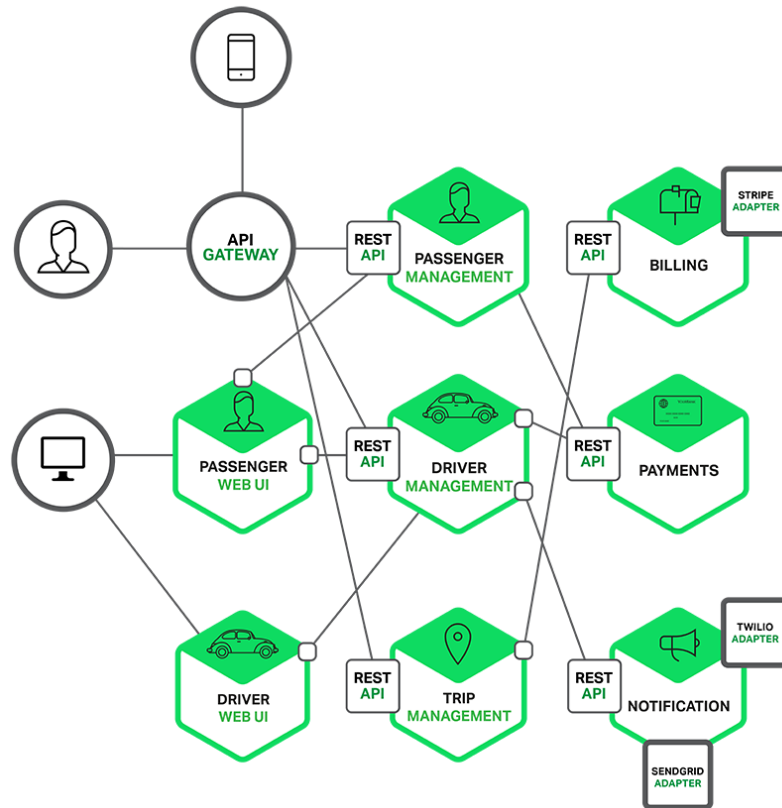
Nhiều năm trở lại đây, nhiều tổ chức đã phát triển ra các hệ thống có thể được thực thi được và đã đạt được những thành công đáng kể tuy nhiên những hệ thống này vẫn còn những thiếu sót và cần phải được cải tiến liên tục. Vậy những hệ thống này ra sao, còn những thiếu sót như thế nào? Ở phần tiếp theo, chúng em sẽ trình bày hệ thống có thể thực thi các quy trình BPMN, từ hệ thống này chúng em sẽ đánh giá những ưu khuyết điểm và đưa ra những giải pháp cải tiến để có thể đáp ứng được nhu cầu sử dụng của người dùng.

CHƯƠNG 3: GIỚI THIỆU VỀ CAMUNDA

Sau khi tham khảo [1] chúng em xin được trình bày sơ lược về *Microservice*, *Camunda*, kiến trúc của *Camunda*, mô hình logic[7] và một số mô hình triển khai của *Camunda* cũng như các thành phần *BPMN* mà *Camunda* có thể hỗ trợ [8]

3.1. Khái niệm về *Microservice*

Là một kiểu xây dựng kiến trúc mà ứng dụng đóng vai trò như một tập hợp được kết nối tương đối với nhau để cài đặt thực hiện cho một nghiệp vụ nào đó. Đây là kiểu kiến trúc hỗ trợ phát triển ứng dụng liên tục, giảm thiểu thời gian và chi phí khi cần phải nâng cấp hoặc thêm mới tính năng khi các ứng dụng trở nên quá lớn, đồ sộ [1] .



Hình 3.1 - Mô hình Microservice cho hệ thống đặt xe [1]

Các thành phần trong **Hình 3.1** được thiết kế thành các service riêng rẽ và chỉ mở ra các cổng *API* riêng để gọi tới từ web và có một *API Gateway* dành cho người dùng di

động. Nếu theo cách cài đặt này, giả sử cần thêm một thành phần mới như **Quảng cáo (Advertisement)** thì chúng ta chỉ cần thêm vào một service riêng lẻ và mở cổng *API* để kết nối tới các service khác cũng như tới người dùng nếu cần mà không ảnh hưởng tới các dịch vụ khác.

3.2. Lợi ích và hạn chế khi sử dụng **Microservice**

Theo [1] kiểu kiến trúc này đã được áp dụng rất nhiều trong nhiều ứng dụng lớn hiện nay như Amazon, Azure, ... Được cho là sẽ trở thành xu thế cho việc thiết kế trong thời gian không xa nhờ vào các điểm mạnh sau:

- Việc đầu tiên có thể dễ dàng thấy được trong kiến trúc **Microservice** là khả năng nâng cấp và mở rộng của kiến trúc rất tuyệt vời bởi mọi việc cần phải làm là thêm một dịch mới vào hệ thống và mở thêm các cổng **API**(Application programming interface) cho các dịch vụ mới.
- Do được cài đặt tách biệt nên các dịch vụ sẽ hoạt động độc lập, có nghĩa là nếu một dịch vụ bị hỏng, trục trặc cũng không ảnh hưởng tới các dịch vụ khác.
- Ngoài ra kiểu kiến trúc này còn có thể hỗ trợ đa ngôn ngữ trong cùng một hệ thống.

Tuy rất được mong đợi sẽ là trở thành xu thế cho việc thiết kế các hệ thống, **Microservice** cũng có những điểm yếu của riêng nó:

- Do phải chia nhỏ thành nhiều dịch vụ khác nhau và có thể đặt ở nhiều nơi khác nhau, từ đó dẫn đến việc độ trễ của đường mạng cũng có thể ảnh hưởng không nhỏ đến các dịch vụ nếu nơi đặt cách quá xa hoặc có tắc nghẽn.
- Mỗi dịch vụ trong **Microservice** lại có cơ sở dữ liệu riêng, việc này yêu cầu cấu trúc dữ liệu phải được thiết kế đồng nhất cho tất cả các dịch vụ.
- Việc thiết kế chia nhỏ thành nhiều dịch vụ còn dẫn đến việc theo dõi, quản lý các dịch vụ này cũng trở nên khó khăn do số lượng của các dịch vụ.

3.3. Các Engine hỗ trợ thiết kế và triển khai mô hình BPMN với kiến trúc Microservice

Nắm bắt được xu thế hiện nay, rất nhiều công ty đã cho ra đời các Engine nhằm phục vụ và hỗ trợ người dùng là doanh nghiệp cũng như các nhà phát triển có thể thiết kế và triển khai các mô hình BPMN cũng như áp dụng cả kiến trúc **Microservice** vào hệ thống công cụ. Một trong số các Engine phổ biến nhất đó tại thời điểm hiện tại có thể kể đến như:

- **Camunda BPM, Camunda Company**
- **Flowable, Flowable Company**

Flowable lẫn Camunda đều có nhân bắt đầu từ Engine Activiti, Alfresco Company, nhưng kể từ năm 2013, Camunda chính thức tách ra khỏi Activiti để trở thành một project mới. Flowable cũng đã có một số tính năng khác biệt so với Activiti nhưng cơ bản vẫn là trên nền tảng của Activiti.

Mỗi Engine sẽ có điểm mạnh và điểm yếu riêng, tùy thuộc vào mục đích và nhu cầu của người sử dụng. Vậy nếu so sánh Camunda BPM và Flowable có nghĩa là cũng một phần so sánh. Từ [12] [5] [14], chúng em đã lập ra một bảng so sánh các Engine trên một số phương diện như sau:

	Camunda	Flowable
Nhân	BPMN 2.0	BPMN 2.0
Database	MS SQL, PostgreSQL, Oracle, MySQL, H2, DB2, MariaDB Có thể cấu hình thay đổi trong quá trình sử dụng Số lượng hỗ trợ nhiều hơn	MS SQL, PostgreSQL, Oracle, MySQL, H2, DB2 Có thể cấu hình thay đổi trong quá trình sử dụng. Hỗ trợ ít hơn
Nền tảng	Java	Java
Thư viện	Hỗ trợ thư viện có thể sử dụng trong các dự án đang thực hiện	Có hỗ trợ thư viện để có thể sử dụng trong các dự án khác
Môi trường	Có 2 công cụ riêng biệt: một công cụ dùng để thiết kế quy trình nghiệp vụ chạy trên desktop, một dùng để thực thi các quy trình nghiệp vụ đó chạy trên môi trường web	Dùng 4 công cụ, mỗi công cụ có một chức năng khác nhau: thiết kế, thực thi, quản lí. Tất cả đều chạy trên môi trường web
Platform	Hỗ trợ Platform thực thi các quy trình nghiệp vụ	Hỗ trợ Platform thực thi các quy trình nghiệp vụ
Tài liệu	Có tài liệu mô tả rõ ràng kiến trúc của hệ thống	Không có tài liệu mô tả kiến trúc hệ thống
Khả năng mở rộng	Hỗ trợ viết plugin thêm vào hệ thống	Không hỗ trợ viết plugin
Khả năng hỗ trợ BPMN 2.0	Hỗ trợ gần như đầy đủ các thành phần của BPMN 2.0	Số thành phần hỗ trợ ít hơn
Hỗ trợ Form cho task	Hỗ trợ thiết kế form theo kiểu liệt kê các thành phần của Form	Hỗ trợ thiết kế kéo thả

Khả năng truy vết dữ liệu	Không hỗ trợ	Không hỗ trợ truy vết Dữ liệu trên form sẽ mất sau khi quy trình hoàn tất
Hỗ trợ lưu dữ liệu	Không	Không
Nghiệp vụ tùy chỉnh	Chỉ hỗ trợ trên project có sử dụng thư viện Camunda, không hỗ trợ trên Platform	Chỉ hỗ trợ trên project sử dụng thư viện Flowable, không hỗ trợ trên Platform
Các thống kê trên OpenHub	Số lượng contributor là 97 người Tổng số Commit trong 1 năm đạt 2,998 Commit	Số lượng contributor là 68 người Tổng số Commit trong 1 năm đạt 2,086 Commit

Bảng 3.1 – Bảng so sánh giữa Camunda và Flowable

Về mặt chức năng, Flowable và Camunda đều được dùng để tạo lập, triển khai và quản lý các quy trình nghiệp vụ. Cho phép người sử dụng thực thi các quy trình đã được định nghĩa. Hỗ trợ hầu như đầy đủ các thành phần của BPMN 2.0 giúp cho việc thực hiện các thao tác trên quy trình nghiệp vụ một cách dễ dàng.

Nhưng bên cạnh đó, cả 2 đều vẫn còn có khuyết điểm như: không hỗ trợ người dùng lưu dữ liệu vào cơ sở dữ liệu của họ, thao tác để thêm một chức năng tùy chỉnh từ ngoài vào gặp nhiều khó khăn.

Về mặt chi tiết hóa, Camunda có tài liệu mô tả kiến trúc một cách rõ ràng giúp cho việc mở rộng hệ thống một cách dễ dàng, có khả năng thêm plugin vào hệ thống trong khi Flowable không làm được, đây là một ưu điểm nổi bật nhất của Camunda so với Flowable

⇒ **Camunda có phần ưu thế hơn nên nhóm sẽ chọn Camunda làm nền tảng để phát triển thêm các tính năng mới mà các công cụ về BPMN chưa đáp ứng được**

3.4. Giới thiệu về Camunda

Camunda là một nền tảng mở có kích thước nhỏ nhẹ dùng cho mục đích phát triển các mô hình **Business Process Management** (BPM). Camunda được phát triển bằng ngôn ngữ **Java** và dựa trên các kiến trúc điển hình của các lập trình viên (*Microservice pattern*), đồng thời cung cấp các giải pháp công nghệ thông tin trong quá trình thiết kế và vận hành các quy trình dựa trên các tiêu chuẩn BPMN 2.0.

Nhân chính của Camunda là được phát triển từ BPMN 2.0 engine chạy trên máy ảo Java (JVM). Được tích hợp với Java Enterprise Edition 6 (JEE 6) và kết hợp rất tốt với Spring Framework. Tính tới thời điểm này, Camunda đã phát triển được một số components quan trọng, thiết yếu dựa theo chuẩn BPMN 2.0.

3.4.1. Process engine và Cơ sở hạ tầng

Process Engine: Là một thư viện Java có trách nhiệm thực thi các mô hình BPMN và các workflow. Sử dụng một nhân POJO (Plain Old Java Object), một cơ sở dữ liệu quan hệ để lưu trữ và MyBatis cho việc mapping ORM (Object Relational mapping).

Tích hợp với Spring Framework.

Có sử dụng Context and Dependency Injection(CDI) và Java EE Integration.

Tích hợp được với các Runtime Container (Tomcat, WildFly, Jboss,...)

3.4.2. Các ứng dụng web mà Camunda cung cấp

Đi kèm theo **Process Engine**, Camunda còn cung cấp thêm các ứng dụng trên nền tảng web để người dùng cũng như các lập trình viên có thể sử dụng **Process Engine** một cách dễ dàng nhất. Các ứng dụng đó bao gồm:

REST API: Ứng dụng được xây dựng với mục đích hỗ trợ việc sử dụng process engine cho các ứng dụng từ xa hoặc các ứng dụng JS.

Camunda Tasklist: Là ứng dụng web cho phép thực thi các quy trình nghiệp vụ và các tác vụ của người dùng (human task). Đồng thời cho phép những người tham gia quy trình có thể theo dõi, giám sát và điều hướng form nghiệp vụ (task forms) để cung cấp dữ liệu đầu vào.

Camunda Cockpit: Ứng dụng web dùng cho việc triển khai các quy trình nghiệp vụ, cho phép tìm kiếm các instance của một process model, theo dõi và sửa chữa những instance có lỗi.

Camunda Admin: Ứng dụng dùng để quản lý các thành phần có trong Camunda như tài khoản, nhóm, quyền, v.v...

Camunda Cycle: Ứng dụng dùng để đồng bộ hóa các quy trình BPMN 2.0 giữa các công cụ modelling và modeler. Từ phiên bản 7.2.0 trở đi

3.4.3. Các công cụ hỗ trợ:

Camunda Modeler: Từ các phiên bản gần đây nhất thì Camunda đã đưa công cụ này thành một ứng dụng riêng viết trên nền tảng NodeJs thay vì là một plug-in của Eclipse Kepler.

Camund-bpmn.js: Một JavaScript Framework dùng để parse, render và thực thi các mô hình BPMN 2.0 từ nguồn XML.

Ngoài ra, Camunda phiên bản gần đây còn cung cấp các công cụ khác như CMMN (Case Management Model and Notation) và DMN (Decision Model and Notation).

3.5. Các thành phần trong BPMN 2.0 mà Camunda có thể hỗ trợ

Màu cam **orange** là những thành phần được Camunda hỗ trợ trong modeler.

Symbols

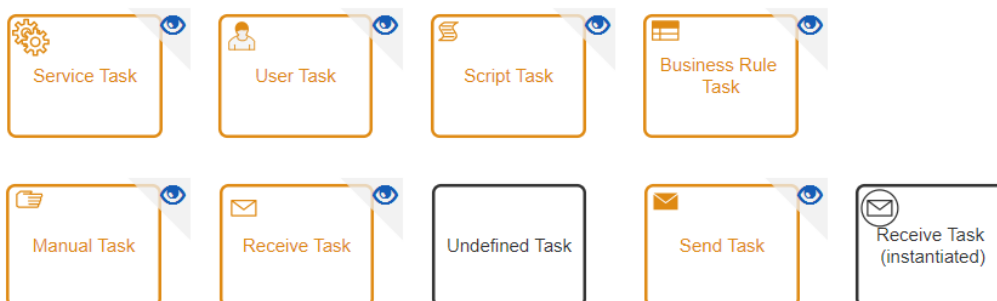
Participants













































Subprocesses



Tasks



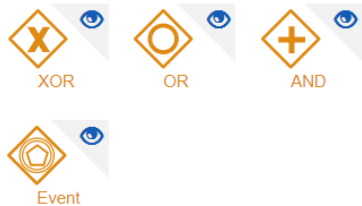
Hình 3.2 – Các loại Activity Camunda hỗ trợ [8]

Loại	Start			Intermediate				End
	Normal	Event Subprocess	Event Subprocess non-interrupt	Catch	Boundary	Boundary non-interrupt	Throw	
None								
Message								
Timer								
Conditional								
Link								
Signal								
Error								
Escalation								

Termination								
Compensation								
Cancel								
Multiple								
Multiple Parallel								

Bảng 3.2 – Cách thành phần sự kiện được cài đặt trong Camunda [8]

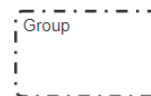
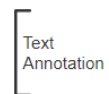
Gateways



Data



Artifacts



Hình 3.3 – Gateway và các loại khác [8]

3.6. Các môi trường Camunda hỗ trợ

3.6.1. Container

Camunda hỗ trợ một số Runtime Container như (không bao gồm Camunda Cycle):

- Apache Tomcat 6.0 / 7.0 / 8.0
- JBoss Application Server 7.2 and JBoss EAP 6.1 / 6.2 / 6.3 / 6.4
- Wildfly Application Server 8.2 / 10.0
- IBM WebSphere Application Server 8.0 / 8.5 (Enterprise Edition only)
- Oracle WebLogic Server 12c (12R1,12R2) (Enterprise Edition only)

3.6.2. Cơ sở dữ liệu

Ngoài việc sử dụng In-memory cơ sở dữ liệu như H2 Engine thì Camunda còn cho phép người dùng có thể customize lại cơ sở dữ liệu để tiện cho việc lưu trữ. Một số cơ sở dữ liệu mà Camunda có hỗ trợ bao gồm:

- MySQL 5.6 / 5.7
- MariaDB 10.0
- Oracle 10g / 11g / 12c
- IBM DB2 9.7 / 10.1 / 10.5 / 11.1 (excluding IBM z/OS for all versions)
- PostgreSQL 9.1 / 9.3 / 9.4 / 9.6
- Microsoft SQL Server 2008 R2/2012/2014/2016 (see Configuration Note)

3.6.3. Web browser:

- Google Chrome latest
- Mozilla Firefox latest
- Internet Explorer 11
- Microsoft Edge

3.6.4. Java

- Java 6 / 7
- Java 8 (if supported by your application server/container)

3.6.5. Java Runtime

- Sun/Oracle Hot Spot 6 / 7 / 8

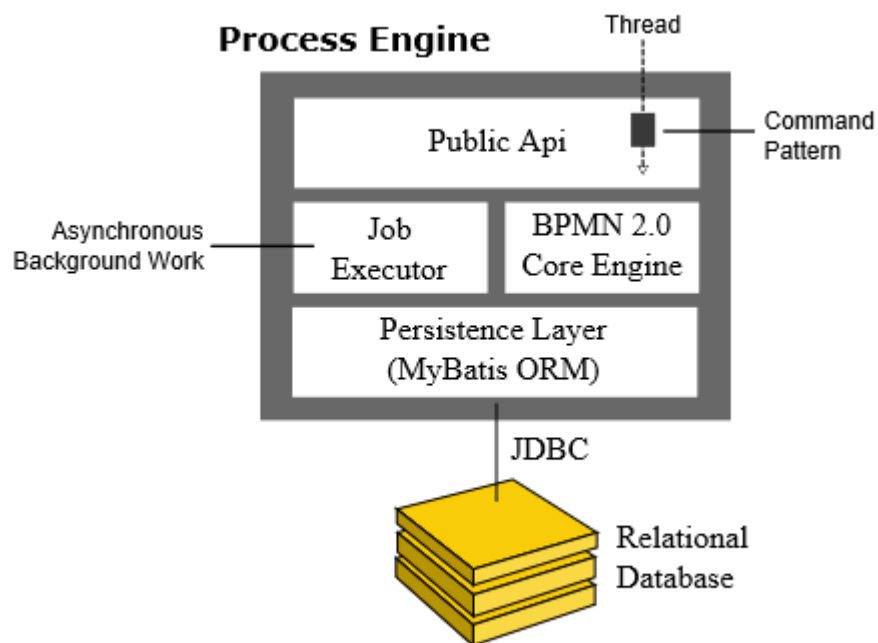
- IBM® J9 virtual machine (JVM) 6 / 7 / 8
- OpenJDK 7 / 8
- Oracle JRockit 6 - R28.2.7

3.6.6. Camunda Modeler

Camunda Modeler được thiết kế phù hợp với các hệ điều hành sau:

- Windows 7 / 10
- Mac OS X 10.11
- Linux

3.7. Tổng quan kiến trúc logic của Process Engine trong Camunda



Hình 3.4 – Kiến trúc của hệ thống Camunda [7]

Camunda Process Engine có 4 thành phần chính:

Process Engine Public API: Các API được cung cấp sẵn trong Process Engine nhằm hỗ trợ các ứng dụng khác có thể tương tác với Engine một cách dễ dàng. Không những hỗ trợ các ứng dụng Java, các API này còn có thể hỗ trợ cho các ứng dụng không sử dụng ngôn ngữ Java. Các chức năng của Camunda đều được chia nhỏ thành các Service riêng biệt. Sử dụng mẫu Command pattern.

BPMN 2.0 Core Engine: Là hạt nhân chính trong Process Engine, có chức năng chuyển đổi BPMN 2.0 XML sang Java Object và BPMN Behavior Implementation.

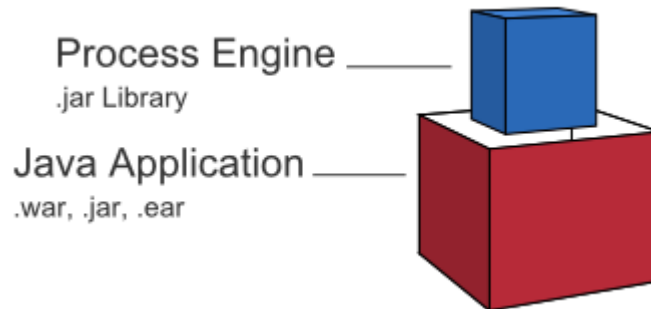
Job executor: Chịu trách nhiệm trong việc thực thi các công việc nền bất đồng bộ (asynchronous background work) như Timer hoặc là các công việc tiếp diễn bất đồng bộ (asynchronous continuous) trong quy trình.

The Persistence Layer: Là tầng dùng để kết nối Process Engine với Cơ sở dữ liệu tương ứng để đồng nhất các thành phần trong quy trình. Sử dụng engine MyBatis mapping để thực hiện việc mapping từ các thành phần trong quy trình thành các dòng lưu trữ xuống cơ sở dữ liệu.

Ngoài ra, Camunda còn được thiết kế như một framework linh hoạt có thể ứng dụng cho nhiều kịch bản khác nhau.

3.8. Một số mô hình triển khai của Camunda

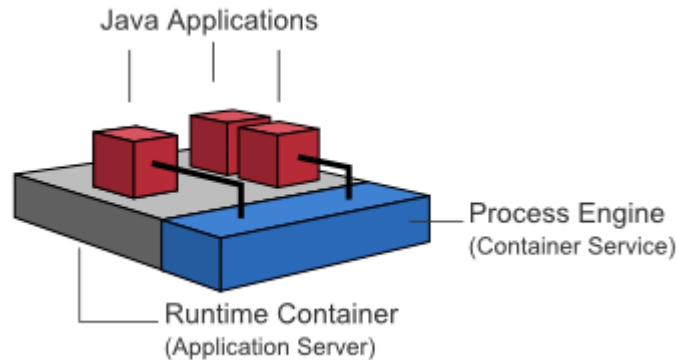
3.8.1. Embedded Process Engine



Hình 3.5 – Mô hình Embedded Process Engine [7]

Theo kịch bản này thì Camunda sẽ là một thư viện đính kèm theo ứng dụng, giúp cho việc khởi động và dừng Camunda trở nên dễ dàng hơn với các ứng dụng Java.

3.8.2. Shared, Container-Managed Process Engine



Hình 3.6 – Mô hình Shared, Container-Managed Process Engine [7]

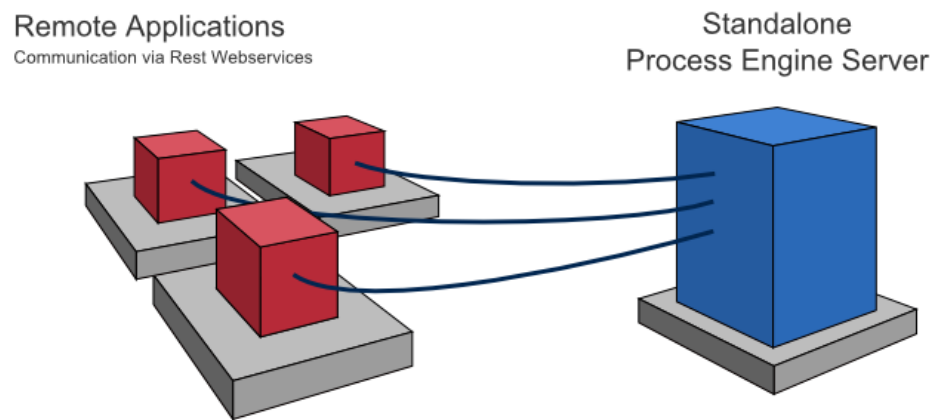
Trong kịch bản này, Camunda được khởi động trong một container (gọi là *Container Service*¹) ngoài so với các ứng dụng khác (được triển khai trong *Runtime Container*²), cung cấp các dịch vụ chung cho tất cả các ứng dụng. Trong trường hợp này

¹ *Container Service*: Là một server dùng để triển khai Camunda

² *Runtime Container*: Là một server được dùng để triển khai riêng cho các ứng dụng khác

một ứng dụng có thể triển khai một quy trình lên Process Engine và ủy nhiệm thực thi cho một ứng dụng khác (delegating).

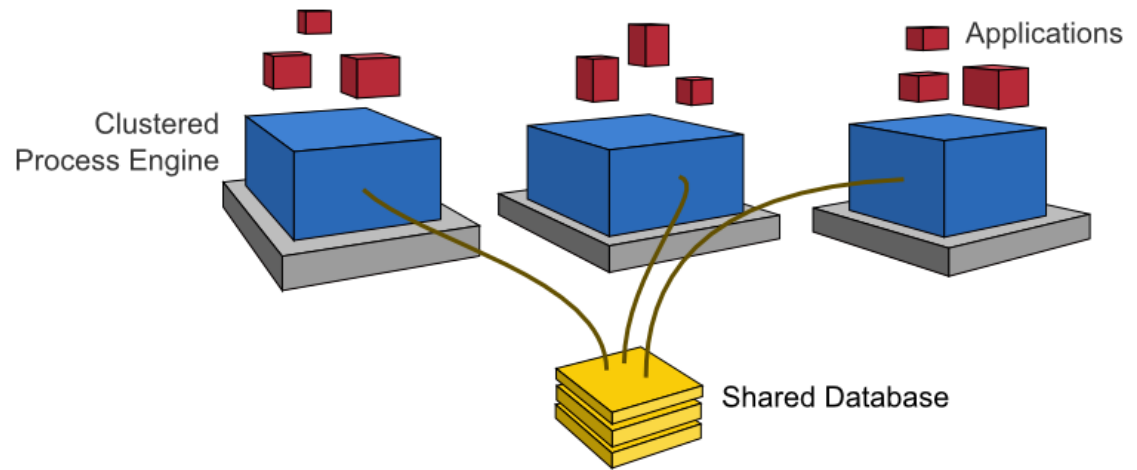
3.8.3. Standalone Process Engine



Hình 3.7 – Mô hình Standalone Process Engine [7]

Trong trường hợp này, Camunda được ứng dụng trong một mạng lưới nội bộ. Thông qua các Rest API, các ứng dụng từ xa có thể giao tiếp, triển khai và thực thi quy trình trên Camunda từ khoảng cách xa hơn so với hai cách trên.

3.8.4. Clustering Model

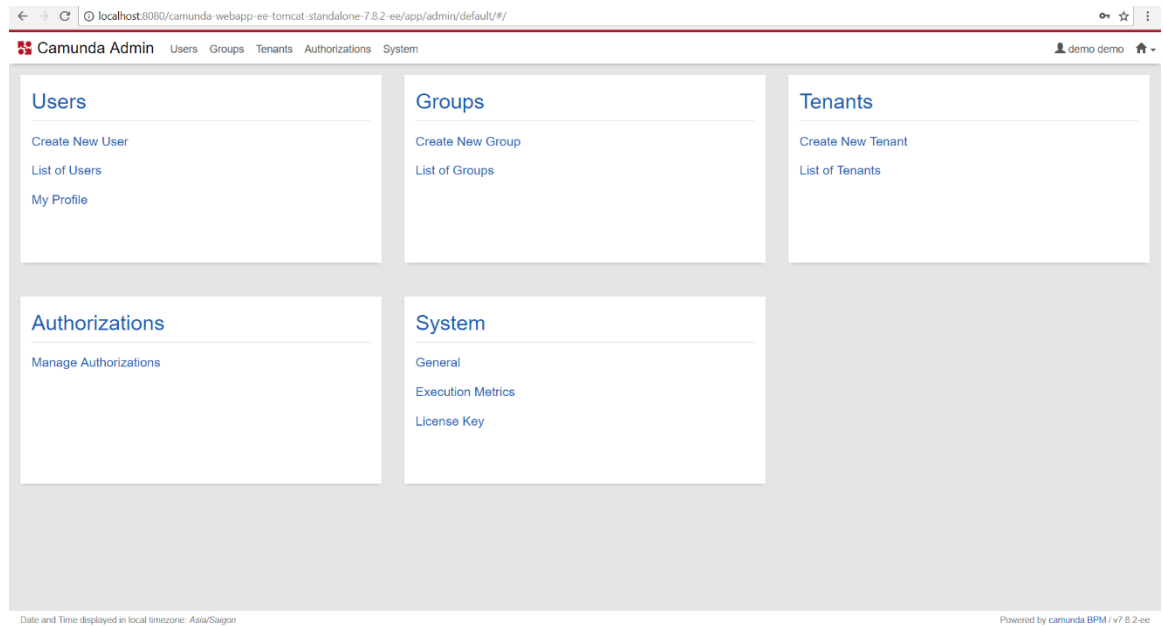


Hình 3.8 – Mô hình Standalone Process Engine [7]

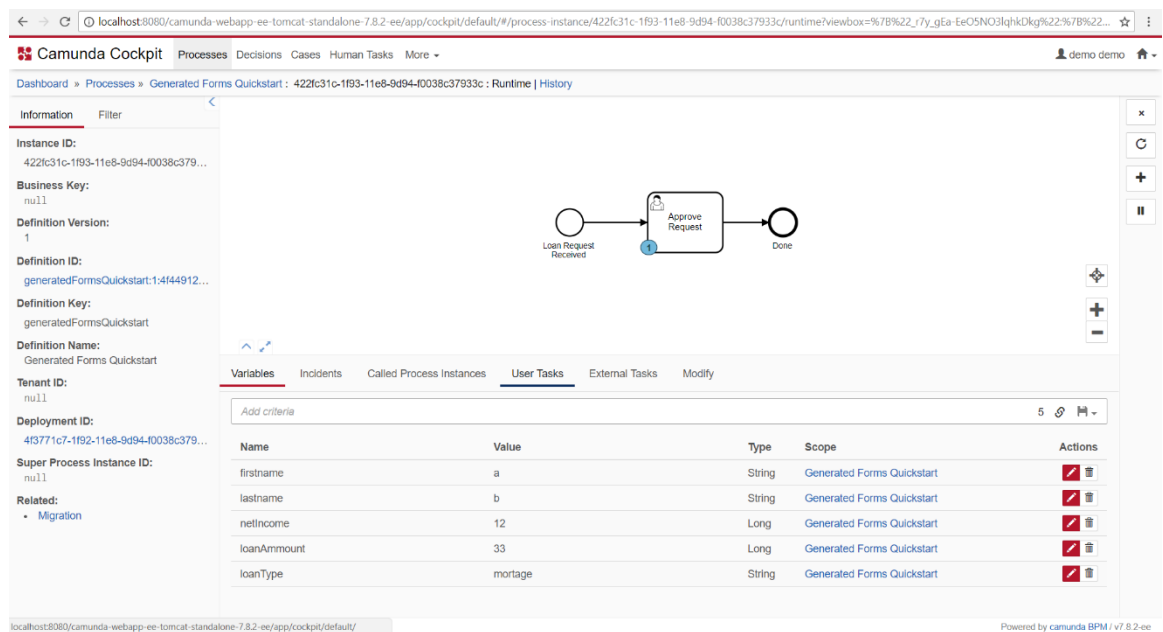
Để hỗ trợ cho việc mở rộng và giảm thiểu tỉ lệ *fail-over*³, các Process Engine được tách ra thành các Cluster node riêng biệt và chia sẻ với nhau cùng một cơ sở dữ liệu.

³ *Fail-over*: Khả năng xảy ra lỗi trên toàn bộ các thành phần của một server

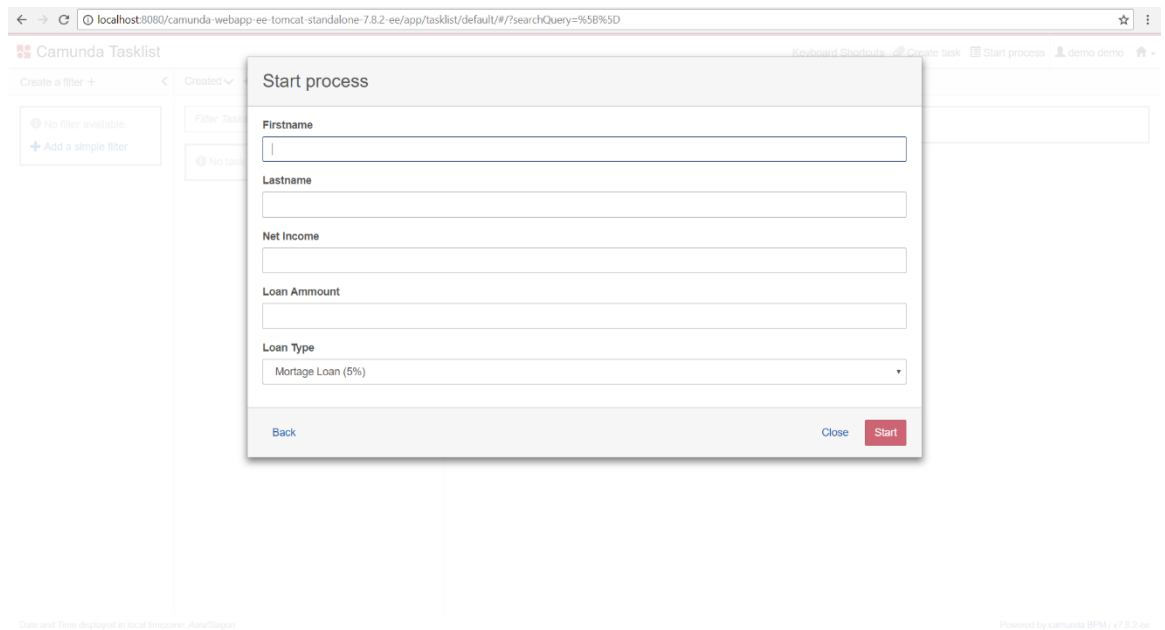
3.9. Một số hình ảnh giao diện của Camunda:



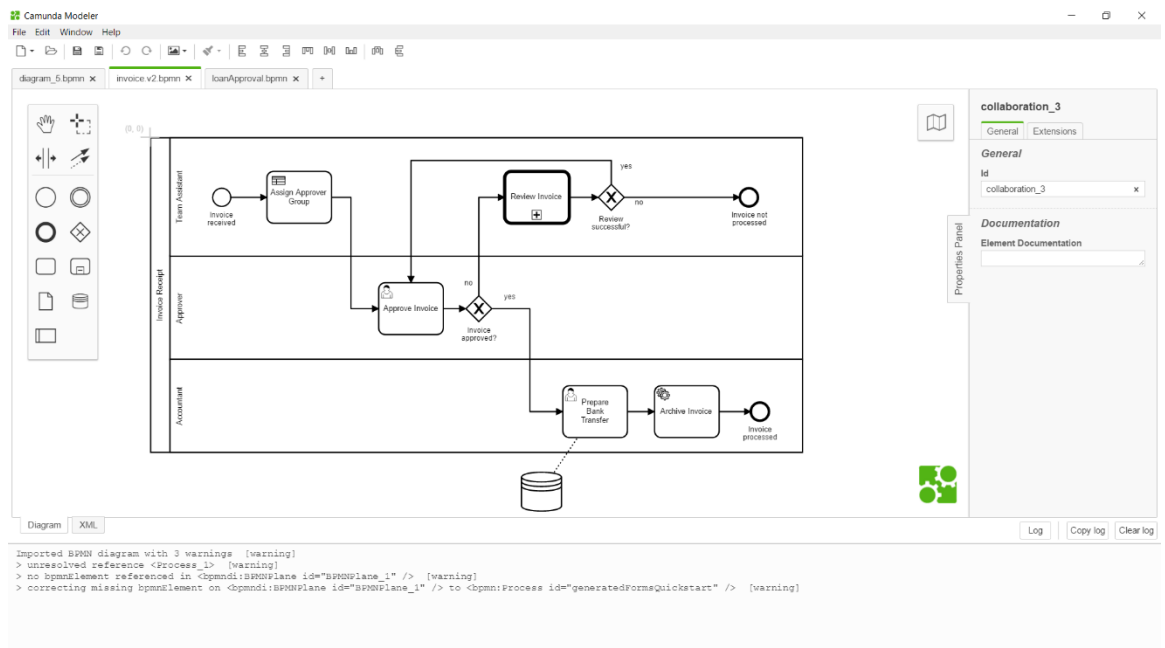
Hình 3.9 – Giao diện của Camunda Admin



Hình 3.10 – Giao diện của Camunda Cockpit



Hình 3.11 – Giao diện của Camunda Tasklist



Hình 3.12 – Giao diện của Camunda Modeller

CHƯƠNG 4: GIỚI THIỆU VỀ HỆ THỐNG SAU KHI CẢI TIẾN

Trong chương này, nhóm em sẽ phân tích những mong muốn về hệ thống sau khi phân tích tích về công cụ có sẵn hiện nay, từ đó đưa ra những giải pháp cùng với những thiết kế mới đáp ứng được những nhu cầu về tính năng cũng như tính khả dụng của hệ thống mới trong việc phát triển các hệ thống quản lý

4.1. Những mong muốn khi cải tiến hệ thống

4.1.1. Những điểm cần cải tiến trong hệ thống Camunda

Camunda tích hợp gần như đầy đủ các tính năng của một hệ thống quản lý. Tuy nhiên bên cạnh đó vẫn còn tồn tại những thiếu sót, đó là trong quá trình thực thi một quy trình nghiệp vụ, đòi hỏi cần thiết phải có nhu cầu lưu trữ thông tin của nghiệp vụ vào cơ sở dữ liệu hoặc một dịch vụ lưu trữ nào đó. Trong BPMN định nghĩa một thành phần đó là Data Store được dùng để thể hiện nơi lưu trữ dữ liệu. Tuy nhiên trong quá trình hiện thực hóa việc thực thi một quy trình nghiệp vụ bằng cách định nghĩa ra quy trình đó và đưa vào một hệ thống để thực thi, hầu như không có một hệ thống nào hỗ trợ việc lưu một thông tin của của quy trình vào cơ sở dữ liệu theo hình thức “vẽ và chạy”. Cách duy nhất để có thể thao tác với cơ sở dữ liệu hiện tại mà Camunda có thể cung cấp là sử dụng Service Task.

Chính vì lý do trên, khi các nhà phát triển muốn thực hiện một nghiệp vụ cần có các thao tác trên cơ sở dữ liệu thì việc mô hình hóa công việc này trở thành một Service Task và thao tác lưu trữ dữ liệu sẽ trở thành một dịch vụ trong khi chúng ta hoàn toàn có thể mô tả việc lưu thông tin bằng mô hình BPMN đó đó việc hỗ trợ thêm thành phần lưu dữ liệu thể hiện trong quá trình mô hình hóa là một việc cần thiết

Camunda là một hệ thống vẽ và thực thi các quy trình nghiệp vụ. Tuy nhiên, việc thực thi quy trình nghiệp vụ này còn thiên khá nhiều về kỹ thuật do đó đối tượng sử dụng của hệ thống này là một người hiểu về kỹ thuật. Trên thực tế, khi triển khai một hệ thống quản lý cho một tổ chức nào đó, chúng ta không nên chỉ quan tâm đến vấn đề kỹ thuật mà

còn hướng đến người sử dụng, một hệ thống với các thao tác phức tạp (cài đặt một Service Task) sẽ tăng độ khó cho người sử dụng đặc biệt là những người không am tường về kỹ thuật và có thể làm chậm đi hiệu suất làm việc của họ khi sử dụng hệ thống trong việc quản lý. Đây là một bất lợi lớn vì hầu hết các phần mềm nói chung và phần mềm quản lý nói riêng được tạo ra nhằm giảm sức lao động thủ công và tăng cao hiệu suất làm việc nên đòi hỏi cần phải cải tiến hệ thống về giao diện giúp cho việc thực hiện các thao tác trong quá trình quản lý được thuận tiện và dễ dàng hơn

4.1.2. Những tính năng cải tiến

Như đã trình bày ở phần trên, nhận thấy sự cần thiết phải thay đổi nên chúng em quyết định thêm những tính năng mới dựa vào hệ thống Camunda nhằm giúp hỗ trợ nhiều hơn trong việc thực thi các quy trình nghiệp vụ dựa trên BPMN, các tính năng đó bao gồm:

- Thay đổi công cụ mô hình hóa bằng BPMN của hệ thống Camunda giúp hỗ trợ người dùng có được những mô tả chi tiết về thông tin của cơ sở dữ liệu, thông tin của các dịch vụ lưu trữ dữ liệu trực tuyến, các ràng buộc về dữ liệu trong quá trình thao tác với cơ sở dữ liệu nhằm hỗ trợ thêm về phần cơ sở dữ liệu cho người sử dụng trong quá trình mô hình hóa nghiệp vụ của một tổ chức nào đó
- Khi thay đổi công cụ mô hình hóa đồng nghĩa với việc sẽ phải thay đổi cách thức thực thi một quy trình để có thể thực thi được những tính năng mới khi thêm vào, nhóm em sẽ thay đổi các thức xử lý, thêm những tính năng xử lý sau khi cải tiến vào hệ thống thực thi quy trình nghiệp vụ của Camunda
- Thay đổi giao diện người dùng khi thực thi một quy trình nào đó. Nhằm giúp người dùng có các thao tác sử dụng đơn giản, dễ hiểu, chúng em quyết định sử dụng lại các thành phần chính của Camunda để phát triển một bộ

giao diện giúp thực thi các quy trình nghiệp vụ được định nghĩa sẵn thông qua việc gọi các dịch vụ mà Camunda cung cấp

4.1.3. Cách tiếp cận trong việc cải tiến

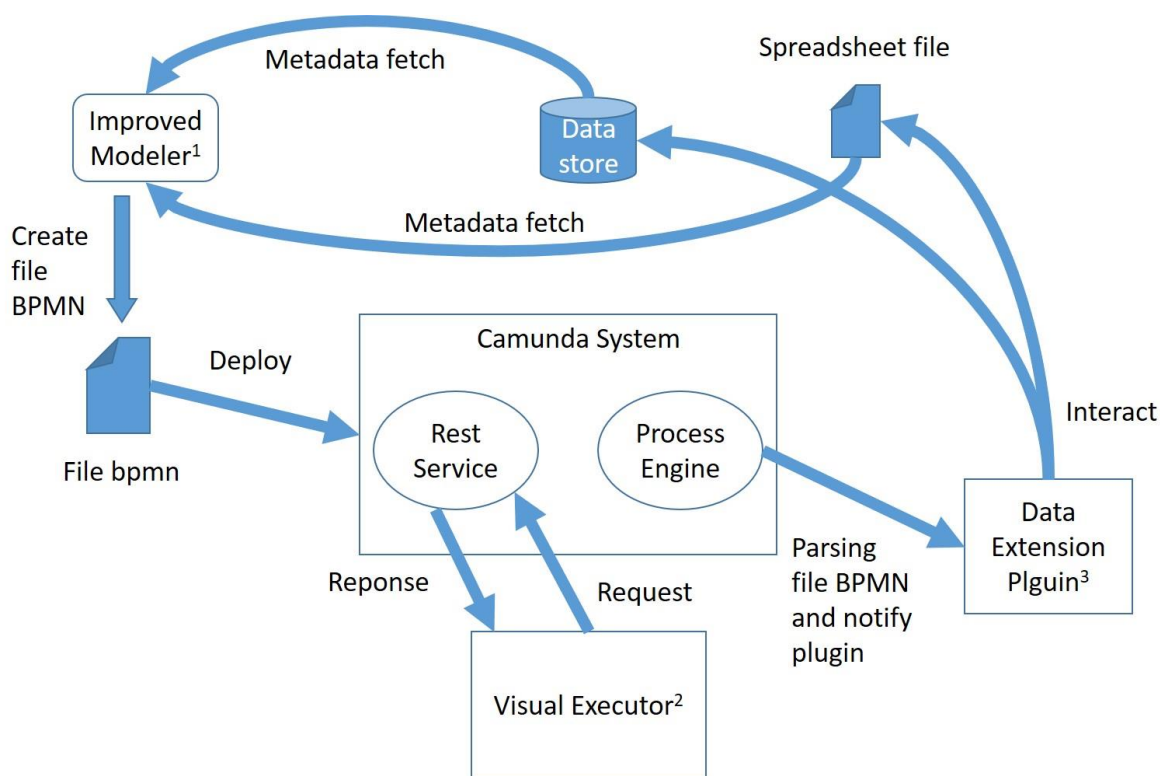
- Khi thay đổi công cụ mô hình hóa của Camunda đòi hỏi việc thêm vào những cấu trúc mới mở rộng từ BPMN nên nhóm em quyết định sử dụng mã nguồn mở của Camunda Modeller để thêm vào những thành phần mới này, việc thêm vào những thành phần này sẽ không làm ảnh hưởng đến quá trình thực thi một quy trình của hệ thống cũ.
- Về phần thực thi một quy trình, Camunda có hỗ trợ thêm vào hệ thống của họ những plugin giúp bắt được những sự kiện trong quá trình thực thi một quy trình, nhóm em quyết định cải tiến việc người dùng sử dụng các thao tác trên cơ sở dữ liệu dựa vào việc thực thi các task, sau đó sẽ bắt được những sự kiện trên các task để xử lý được những thao tác mà nhóm đã cải tiến.
- Về phần cải tiến giao diện, nhóm em sử dụng java spring để xây dựng một trang web giúp thực thi các quy trình nghiệp vụ đã được định nghĩa trước đó. Trang web này dựa vào dịch vụ mà Camunda cung cấp sẵn để gọi và thực thi các quy trình dưới dạng dịch vụ. Phần cải tiến này chỉ có tác dụng thêm mới một giao diện và không làm ảnh hưởng đến hệ thống cũ

CHƯƠNG 5: QUÁ TRÌNH CÀI ĐẶT

Trong chương này chúng em sẽ trình bày về những kỹ thuật trong quá trình cài đặt những tính năng cải tiến dựa trên hệ thống cũ đó là Camunda, những công nghệ, phương pháp tiếp cận đồng thời nêu ra những khó khăn và những hạn chế trong quá trình cài đặt

5.1. Sơ đồ thực hiện tổng quát của hệ thống Camunda sau khi cải tiến

Sau đây chúng em xin được trình bày sơ đồ thực hiện của hệ thống chúng em đề xuất khi thực hiện cài đặt Plugin



Hình 5.1 - Sơ đồ thực hiện của hệ thống

Trong đó (1), (2), (3) là những phần chúng em đã thực hiện cải tiến hoặc cài đặt.

Các thông tin từ các nguồn dữ liệu (Data Source) sẽ được đẩy về cho công cụ mô hình hóa, Camunda Modeler để người dùng có thể mô tả các ràng buộc đối với các quy

trình và xuất ra dạng tập tin .bpmn. Các quy trình sau khi được triển khai và tạo các quy trình trên Camunda System, Process Engine sẽ thông báo sự kiện đến DEP để tiến hành chạy. Hiện tại sự kiện có thể kích hoạt cho DEP chạy là sự kiện hoàn thành biểu mẫu của 1 User Task hoặc sự kiện vừa khởi tạo biểu mẫu của 1 User Task. Theo yêu cầu đặt ra của luận văn, nên DEP chỉ giới hạn hoạt động trong khuôn khổ tương tác với các User Task.

Sau khi được kích hoạt, DEP sẽ lấy thông tin từ tập tin BPMN do Process Engine của Camunda cung cấp và chuyển thành các lớp tương ứng với các thẻ sẽ được mô tả ở phần Camunda Modeler. Các Connector được thiết kế bên trong sẽ nhận các đối tượng được sinh ra từ các thẻ đó và thiết lập các câu truy vấn tương ứng hoặc request (đối với Spreadsheet) tương ứng để tiến hành thao tác với dữ liệu.

Về Visual Executor, có vai trò hiển thị lại các thông tin đã được hiển thị trên Camunda Tasklist. Nói cách khác, nó đóng vai trò là một Tasklist được thiết kế lại sao cho có thể giúp người dùng dễ tương tác, bỏ qua một vài bước để loại bớt sự rườm rà khi cần tạo mới một quy trình, cũng như cải thiện giao diện. Hoạt động tương tự như Camunda Tasklist, công việc của Visual Executor là request tới REST Engine của Camunda và hiển thị màn hình kết quả trả về từ REST Engine.

Chi tiết cài đặt cụ thể của từng thành phần sẽ được mô tả chi tiết ở các phần dưới đây.

5.2. Mở rộng hệ thống mô hình hóa BPMN

5.2.1. Phương pháp tiếp cận

Camunda Modeler là một công cụ mã nguồn mở viết bằng JavaScript, có thể cho phép người dùng có thể viết Plugin mở rộng cho công cụ này. Tuy nhiên, Plugin này chỉ mở rộng cho phần **Client** và không hỗ trợ các Plugin xử lý trên **Server** và với ngữ cảnh hiện tại, nhóm em muốn mở rộng tính năng mô hình hóa lại các thao tác trên **Server** và

thao tác trên Google Spreadsheet đòi hỏi cần phải phân tách cách thức xử lý tại **Server** và **Client** nên chúng em quyết định không sử dụng plugin mà dùng phương pháp thêm một số Module vào mã nguồn của công cụ.

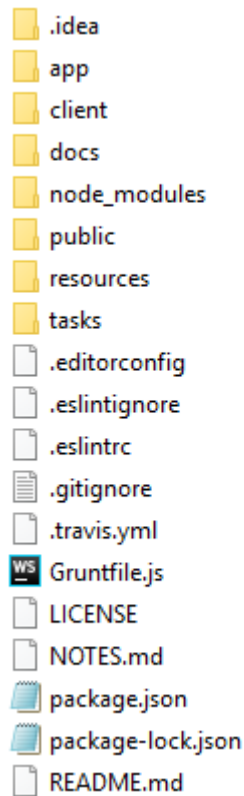
5.2.2. Giới thiệu về Camunda Modeler

Camunda Modeler là một công cụ cho phép mô hình hóa nghiệp vụ bằng BPMN, cho phép người dùng có thể vẽ các sơ đồ BPMN, thể hiện dưới dạng các kí hiệu đồ họa và được lưu trữ dưới dạng các tập tin XML.

Camunda Modeler được viết trên nền tảng Web, sử dụng Nodejs kết hợp với Electron để tạo thành một ứng dụng chạy độc lập như ứng dụng Desktop mà không cần đến Web Browser

Để có thể mở rộng được công cụ này, trong phần tiếp theo chúng em xin trình bày cấu trúc các tập tin mã nguồn chính của công cụ mô hình hóa. Từ đó, chúng em sẽ chỉ rõ chức năng của từng thư mục, sau đó chỉ ra phương pháp dùng để mở rộng công cụ này.

5.2.3. Cấu trúc các tập tin mã nguồn của công cụ Modeler



Hình 5.2 - Cấu trúc các thư mục trong Modeler

Như chúng em đã trình bày, công cụ mô hình hóa BPMN này là một ứng dụng dưới dạng Desktop nhưng bản chất của nó hoạt động dựa trên nền tảng web, sử dụng Electron và được chia tách thành 2 phần, đó là:

- ❖ **Client:** được tổ chức trong thư mục *client*, có chức năng hỗ trợ người dùng trong việc thao tác với các thành phần trong BPMN, client sử dụng Grunt để build các file JavaScript thành một tập tin được đặt trong thư mục *public*.
- ❖ **Server:** được tổ chức trong thư mục *app*, có nhiệm vụ xử lý các sự kiện liên quan đến hoạt động của ứng dụng như: các thao tác trên cửa sổ, các thao tác trên menu chức năng,...

Khi Electron chạy, nó sẽ tìm file JavaScript chính được chỉ định từ từ *package.json* để chạy. Mã nguồn trong thư mục *client* được build vào thư mục *public*, file javascript này load các tập tin trên **Client** tại thư mục *public* và tiến hành khởi chạy ứng dụng.

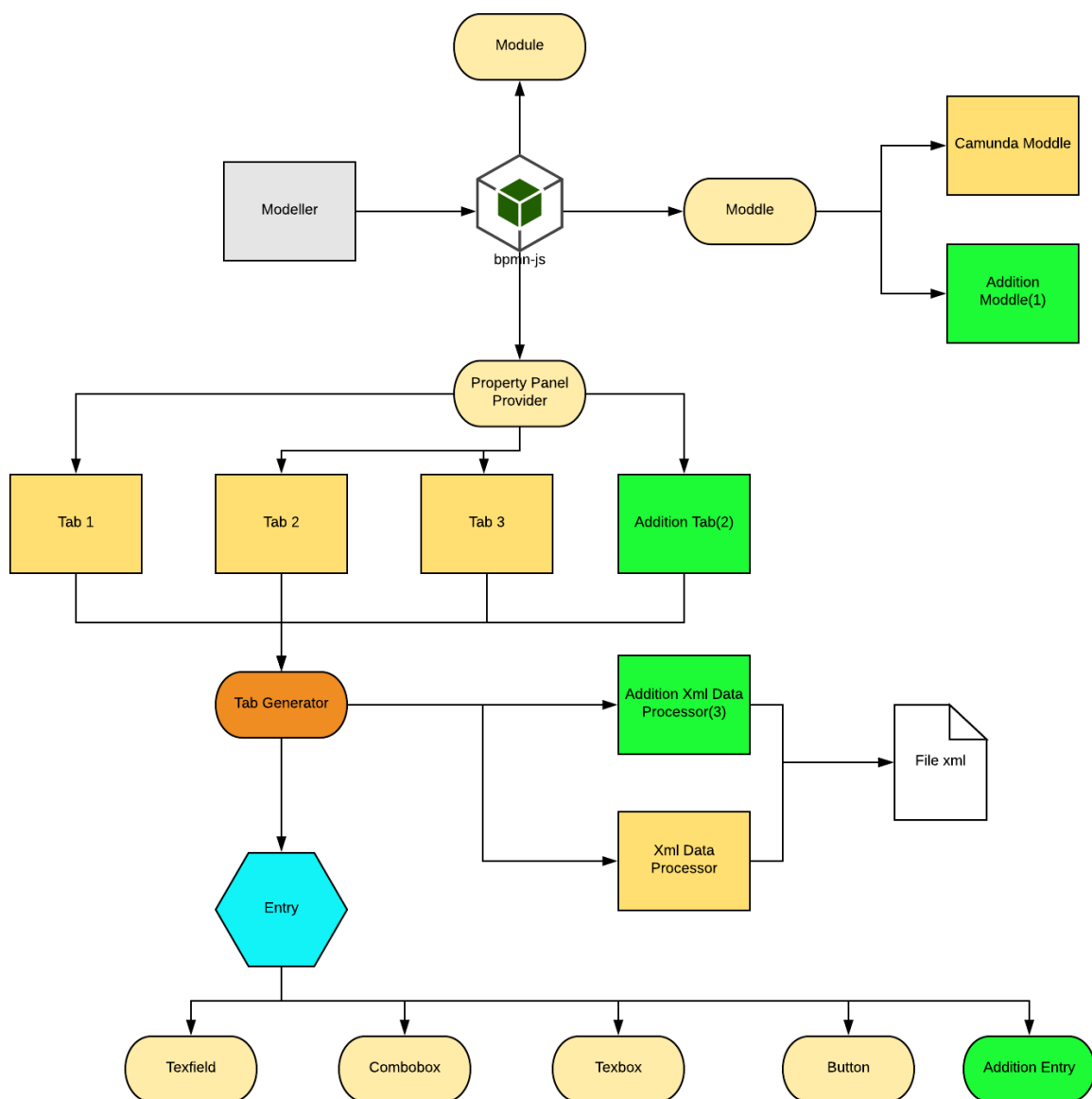
File *Gruntfile.js* mô tả cấu hình giúp build toàn bộ mã nguồn thành một ứng dụng trên desktop

5.2.4. Cách thức tổ chức của công cụ để có thể vẽ mô hình BPMN

5.2.4.1. Cách tổ chức các thành phần trên client

Camunda Modeler sử dụng thư viện **bpmn-js** của Nodejs để vẽ các thành phần của một mô hình BPMN. Thư viện **bpmn-js** cho phép chúng ta thêm những **Module** liên quan đến việc vẽ mô hình BPMN, trong đó có Module **bpmn-js-properties-panel** cho phép chúng ta thêm những thuộc tính dành cho những thành phần của BPMN. Camunda cũng sử dụng module này cho việc mở rộng thêm những thành phần của BPMN nên chúng em lựa chọn mở rộng thêm những thành phần cũng dựa vào cách này.

Sau đây chúng em xin trình bày cách tổ chức cũng như cách thức hoạt động của bpmn-js khi thêm vào những module như **Hình 5.3** sau.



Hình 5.3 - Cách thức tổ chức các module trên client trong modeler

Module chính dùng để thêm các thuộc tính vào BPMN đó là **Provider**, tại đây **Provider** sẽ định nghĩa các Tab dùng trong BPMN, với từng Tab, **Provider** định nghĩa các **Property** cho từng tab, các **Property** này có thể là một *textbox*, *combobox*, *table*, ... được gọi là một **Entry**, mỗi **Entry** có chứa dữ liệu HTML phục vụ cho việc hiển thị giao diện các thuộc tính của một thành phần BPMN. Mỗi **Entry** trong Camunda Modeler có

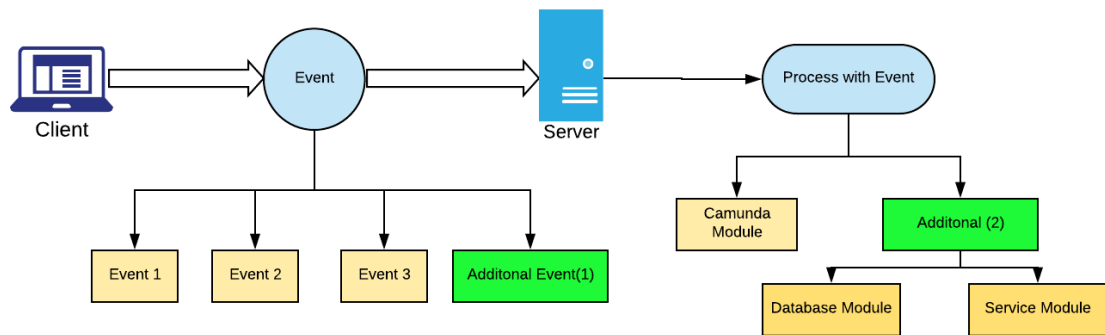
thể cài đặt các phương thức **get** và **set**, trong đó, phương thức **get** dùng để đọc các thuộc tính từ tập tin lưu trữ XML để hiển thị vào các thuộc tính, phương thức **set** dùng để xử lý các thao tác dùng để thêm một thẻ vào tập tin XML lưu trữ dữ liệu của mô hình BPMN.

Trong quá trình xử lý các Entry trong Camunda Modeler, chúng ta có thể thêm một thẻ vào tập tin lưu trữ xml của mô hình, các thẻ này được quy định trong một Module được gọi là **moddle**. Tại đây, các thẻ và mô tả của các thẻ được quy định trong tập tin Json. Khi có yêu cầu thêm một thẻ vào, **bpmn-js** sẽ kiểm tra những **moddle** này nhằm đảm bảo tính hợp lệ của các thẻ.

Trong **Hình 5.3**, chúng em đã đề cập đến cách tổ chức của các module chính dùng cho việc thêm các thuộc tính vào công cụ modeler. Để thêm được các thuộc tính này, chúng em sẽ phải thêm vào những Module mới dùng cho việc lưu trữ và hiển thị dữ liệu các thuộc tính cho các thành phần của BPMN, các thành phần này được tô màu xanh(1)(2)(3)(4) dùng để phân biệt với các thành phần sẵn có hiện tại của Camunda. Các phần này có ý nghĩa như sau:

- ❖ **Additional Tab:** là các tab được thêm vào nhằm mục đích hỗ trợ hiển thị thông tin của thuộc tính theo các thành phần của BPMN
- ❖ **Additional Xml Data Processor** mô tả các ràng buộc về quá trình xử lý dữ liệu được lưu vào tập tin XML, trong đó có một số thẻ và thuộc tính mới của thẻ được thêm vào nhằm hỗ trợ cho việc mô tả thông tin của các chức năng mới – Các thao tác trên dữ liệu. Mô tả và cách thức đặt thẻ này ra sao chúng em sẽ trình bày ở phần sau.
- ❖ **Additional Moddle** mô tả thông tin của các thẻ được thêm vào nhằm hỗ trợ việc kiểm tra tính hợp lệ khi đưa các thẻ mới vào công cụ

5.2.4.2. Cách tổ chức trên server



Hình 5.4 - Cách tổ chức trên Server

Trên **Server**, Electron tiếp nhận các yêu cầu từ **Client** thông qua các sự kiện, dựa vào từng sự kiện sẽ có những cách xử lý và trả về kết quả cho **Client**. Hình 6.3 mô tả hoạt động của công cụ Camunda Modeler trong quá trình gửi dữ liệu yêu cầu từ phía **Client**, phần màu xanh là những sự kiện và những xử lý chúng em đã thêm vào. Để xử lý được các yêu cầu liên quan đến các thao tác trên dữ liệu đòi hỏi việc thao tác trên **Server**, để làm được điều này, trong **Hình 5.4** chúng em sẽ thêm vào những sự kiện mới(1), các sự kiện đó bao gồm những sự kiện liên quan đến kết nối, đọc thông tin của cơ sở dữ liệu, kết nối và xử lý trên các tập tin liên quan đến các dịch vụ lưu trữ, cùng với các sự kiện này, các Module mới được thêm vào(2) để xử lý các sự kiện liên quan và trả về kết quả là những thông tin chứa trạng thái, các gói dữ liệu theo yêu cầu về phía Client để tiếp tục xử lý.

5.2.5. Mở rộng mô hình hóa các thao tác trên cơ sở dữ liệu

5.2.5.1. Mở rộng trên client

Để phục vụ cho việc mở rộng những tính năng liên quan đến việc thao tác trên dữ liệu, chúng em đã thêm một số thẻ mới vào nhằm mục đích mô tả thông tin và các ràng buộc về thao tác trên dữ liệu, các thẻ này được mô tả trong **Bảng 5.1** như sau:

Tên thẻ	Ý nghĩa thẻ	Thuộc tính	Ý nghĩa
databaseInformation	Mô tả thông tin của cơ sở dữ liệu	databaseType	Loại cơ sở dữ liệu
		server	Địa chỉ của cơ sở dữ liệu
		username	Tên truy cập
		password	Mật khẩu truy cập
		database	Tên của cơ sở dữ liệu
dataCondition	Chứa thông tin về thao tác trên cơ sở dữ liệu	databaseTable	Bảng được chọn để thao tác
		action	Thao tác trên cơ sở dữ liệu (insert, update, delete)
dataStoreFields	Chứa các ràng buộc về dữ liệu thao tác trên cơ sở dữ liệu		
field	Thẻ con của dataStoreFields Chứa thông tin về từng ràng buộc.	column	Cột trên cơ sở dữ liệu được thao tác
		variable	Biến trong một process tương ứng với cột được chọn, có thể lấy giá trị biến để thao tác vào cột hoặc lấy giá trị từ một dòng của cột lưu vào giá trị biến

Bảng 5.1 – Chi tiết các thẻ được thêm vào cấu trúc lưu trữ

BPMN cho phép chúng ta mở rộng các thẻ mới bằng cách đặt các thẻ này vào thẻ **Extension Element** nên về mặt cấu trúc, các thẻ mới chúng em thêm vào sẽ được đặt

vào thẻ này để đảm bảo được cấu trúc của BPMN, các thẻ mới được thêm vào được cài đặt như sau

Để thao tác với các thuộc tính liên quan đến giao diện, đầu tiên cần phải tạo các Tab mới để thao tác, để thêm các Tab mới, chúng em đã thêm vào Module **Provider** như **Mã nguồn 5.1** sau

```
function CamundaPropertiesProvider(eventBus, bpmnFactory, elementRegistry, elementTemplates, translate) {

    PropertiesActivator.call(this, eventBus);

    this.getTabs = function(element) {

        .....

        var connectionTab = {
            id: 'connection',
            label: 'Connection',
            groups: createConnectionTabGroups(element, bpmnFactory, elementRegistry, translate)
        };

        .....

        return [
            .....
            connectionTab,
            .....
        ];
    };
}
```

Mã nguồn 5.1 – Thêm một Tab mới

Trong đó hàm **createConnectionTabGroups** sẽ gọi đến Module **XML DataProcessor**, cung cấp giao diện và các thao tác xử lý trên XML.

Về xử lý giao diện Các thao tác với thẻ **databaseInformation** được xử lý trên đối tượng **Data Store Reference**, đầu tiên cần kiểm tra đối tượng đang thao tác là **Data Store Reference**

```
if (is(element, 'bpmn:DataStoreReference')) {
    .....
}
```

Mã nguồn 5.2 – Thao tác trên DataStoreReference

Mã nguồn 5.2 có ý nghĩa là nếu đối tượng được kéo vào màn hình là Data Store Reference thì Tab này sẽ được thêm vào Property Panel của **bpmn-js**.

Tiếp theo chúng em sẽ ví dụ thêm một số thuộc tính vào Tab như sau

- Thêm loại cơ sở dữ liệu

```
var databaseType=entryFactory.comboBoxField({
  id : 'databaseType',
  description : 'Choose Database Type',
  label : 'Database Type',
  modelProperty : 'databaseType',
  selectOptions: [
    { name: 'mssql', value: 'SQL Server' },
    { name: 'mysql', value: 'MySQL' },
    { name: 'oracle', value: 'Oracle' }
  ],
  attributes:attr.databaseType
});
databaseType.get=getValue(getBusinessObject(element));
databaseType.set=setValue(getBusinessObject(element));
group.entries.push(databaseType);
```

Mã nguồn 5.3 – Thêm loại cơ sở dữ liệu

- Thêm tên truy cập và mật khẩu truy cập

```
var username=entryFactory.textField({
    id : 'usn',
    description : 'Username to connect to server',
    label : 'User Name',
    modelProperty : 'username'
});

username.get=getValue(getBusinessObject(element));
username.set=setValue(getBusinessObject(element));
group.entries.push(username);

var password= entryFactory.textField({
    id : 'pw',
    description : 'Password to connect to server',
    label : 'Password',
    modelProperty : 'password'
});

password.get=getValue(getBusinessObject(element));
password.set=setValue(getBusinessObject(element));
group.entries.push(password);
```

Mã nguồn 5.4 – Thêm tên truy cập và mật khẩu truy cập

Trong mã nguồn 5.3 và 5.4, có **entryFactory**, đây là một bộ các HTML giúp cho việc hiển thị các thuộc tính trên giao diện của công cụ, trong phần này chúng em đã thêm một **Entry** mới đó là “Button” nhằm thực hiện một số thao tác kiểm tra

Ngoài ra trong các Entry có các phương thức get và set để xử lý các thao tác liên quan đến đọc và ghi XML, các phương thức đó như sau

```

var getValue = function(businessObject) {
    return function(element) {
        var parent=businessObject.extensionElements;
        if(!parent) {
            return {};
        }
        var properties=getPropertiesElement(parent);
        if(!properties) {
            return {};
        }
        return properties.$attrs;
    };
};

var setValue = function(businessObject) {
    return function(element, values) {
        var cmd=[];
        var parent=businessObject.extensionElements;
        if(!parent) {
            parent = elementHelper.createElement('bpmn:ExtensionElements', {values:
            []}, businessObject, bpmnFactory);
            cmd.push(cmdHelper.updateBusinessObject(element, businessObject, {extensionElements: parent}));
        }
        var properties=getPropertiesElement(parent);
        if(!properties) {
            properties = elementHelper.createElement('kltm:DatabaseInformation', {},
            parent, bpmnFactory);
            cmd.push(cmdHelper.addAndRemoveElementsFromList(
                element,
                parent,
                'values',
                'extensionElements',
                [properties],
                []
            ));
        }
        cmd.push(cmdHelper.updateBusinessObject(element,properties,values));
        if(!properties.$attrs.databaseType){
            properties.$attrs.databaseType='mssql';
        }
        return cmd;
    };
};

```

Mã nguồn 5.5 – Phương thức Get và Set

Mã nguồn 5.5, hàm **getValue** được sử dụng để lấy giá trị từ tập tin XML và hiển thị lên giao diện, hàm **setValue** để lấy giá trị nhập vào từ giao diện lưu vào tập tin XML. Đối với hàm **setValue** có phương thức **createElement** của thư viện **elementHelper** với mục đích tạo ra thẻ mới để lưu vào tập tin XML, các thẻ này đã được mô tả ở **Bảng 5.1**

5.2.5.2. Mở rộng phần Server

Khi các thông tin mô tả cơ sở dữ liệu được thêm vào, việc kiểm tra tín đảm bảo khi kết nối, thêm các ràng buộc về thao tác trên cơ sở dữ liệu đòi hỏi việc đọc thông tin trên cơ sở dữ liệu, công việc này được thực hiện trên **Server**, **Client** yêu cầu server kiểm tra và đọc thông tin của cơ sở dữ liệu dưới dạng các thông điệp, tại phía **Server** sẽ có một thành phần lắng nghe thông điệp và đọc được dữ liệu gửi đến **Server**, sau đó dùng module nodejs cho các loại cơ sở dữ liệu để kiểm tra và đọc thông tin của cơ sở dữ liệu, sau đó sẽ trả về cho client một gói dữ liệu mô tả trạng thái kết nối hoặc thông tin của cơ sở dữ liệu để client hiển thị kết quả cho người dùng và tiếp tục thực hiện các công việc xử lý trên client.

Giả sử như sự kiện người dùng nhấn nút “Test Connection” được bắt, **Client** sẽ gửi một thông điệp như **Mã nguồn 5.6** sau

```
var result=ipcRenderer.sendSync('client:ConnectDatabase',prop);
```

Mã nguồn 5.6 – Gửi thông điệp từ Client đến Server

IpcRenderer là một thành phần của **Electron** cho phép gửi những thông điệp về **Server** trong trường hợp này hàm **sendSync** dùng để gửi thông điệp và sẽ chờ **Server** phản hồi.

Tại phía **Server** luôn có một **Listener** lắng nghe và bắt các sự kiện từ **Client**, thao tác này được mô tả như **Mã nguồn 5.7** sau.

```
ipcMain.on('client:ConnectDatabase', function (event, props) {  
    database.Connect(props.databaseType,props.server,props.username,props.password,props.database,true,function (err) {  
        event.returnValue=err;  
    });  
});
```

Mã nguồn 5.7 - Xử lý thông tin trên Server

IpcMain là một thành phần của **Electron** cho phép Server lắng nghe những sự kiện đến từ **Client**, xử lý và trả về kết quả tương ứng. Trong trường hợp này event.returnValue được sử dụng để trả về kết quả cho **Client** đang chờ tương ứng với hàm **sendSync** trên **Server**.

5.2.6. Mở rộng các chức năng liên quan đến Google Spreadsheet

5.2.6.1. Mở rộng client

Cũng giống như cơ sở dữ liệu, việc thêm các thao tác trên Google Spreadsheet cũng cần thêm các thẻ mới vào cấu trúc tập tin xml các thẻ này được mô tả trong **Bảng 5.2** như sau

Tên thẻ	Ý nghĩa thẻ	Thuộc tính	Ý nghĩa
service	Mô tả các thông tin liên quan đến xác thực và thông tin của tập tin trên các dịch vụ lưu trữ	servicetype	Loại dịch vụ lưu trữ có thể là google drive, onedrive,... Ở đây nhóm em cài đặt trên google drive nhưng vẫn đặt loại dịch vụ để tiện cho việc mở rộng sau này
		token	Chứa dữ liệu xác thực của người dùng, sử dụng cho việc truy cập vào dịch vụ
		fileid	Chứa định danh của tập tin được sử dụng để lưu trữ trong dịch vụ
		sheetId	Chứa định danh của sheet sẽ thao tác trong tập tin

Bảng 5.2 – Các thẻ mô tả thông tin google spreadsheet

Các thao tác thêm giao diện cho thuộc tính vào công cụ giống như các thao tác trên cơ sở dữ liệu và được mô tả như sau:

❖ **Thẻ Service**

Hiện thị các thông tin liên quan đến các dịch vụ lưu trữ trực tuyến. Tại đây, giao diện cho phép người dùng chọn được các service (trong phần này chúng em chỉ cài đặt cho dịch vụ của google drive nhưng vẫn đặt *Combobox* để thuận tiện cho việc mở rộng sau này), cho phép người dùng chọn tập tin dùng để lưu trữ, thao tác này **Client** gửi đến server để thực hiện, trong trường hợp chưa có *token* xác thực thì sẽ có cửa sổ đăng nhập cho người dùng.

Sau khi chọn tập tin, người dùng có thể tạo ra được một tập tin mẫu dùng cho việc lưu trữ dữ liệu trên tập tin bằng cách nhấn “Apply”, **Client** sẽ gửi yêu cầu về **Server** thực hiện

5.2.6.2. Mở rộng server

Sau khi thêm các thẻ vào cấu trúc xml, chúng em cũng thêm vào những xử lý trên **Server** dùng để lấy được *token* truy cập vào tập tin **Spreadsheet**. Khi có yêu cầu đến **Spreadsheet**, **Server** sẽ kiểm tra xem người dùng đã có *token* trên tập tin XML hay chưa, nếu chưa có thì **Server** sẽ mở một cửa sổ mới nhằm giúp người dùng có thể đăng nhập. Sau khi người dùng đăng nhập, **Server** sẽ trả về cho **Client** *token* và danh sách các tập tin.

Khi người dùng xác nhận lưu trữ trên tập tin Spreadsheet bằng cách nhấn “Apply”, Server sẽ thêm một *sheet* mới vào tập tin và dựa vào thông tin của các Form Field trên Usertask để tạo ra các cột mẫu trên Spreadsheet nhằm thuận tiện cho việc thao tác trên dữ liệu khi thực thi một quy trình.

5.3. Mở rộng hệ thống vận hành quy trình bằng DEP

5.3.1. Giới thiệu về Plugin trong Camunda:

Camunda có cung cấp cơ chế cho phép người dùng có thể tự tạo ra Plugin để có thể mở rộng và tùy chỉnh cách mà Camunda hoạt động, chỉ cần khai báo tên lớp của Plugin trong tập tin **bpm-platform.xml** và tập tin nhị phân của lớp đó phải có trong thư mục thư viện của máy chủ chạy Camunda là có thể vận hành được.

```
<?xml version="1.0" encoding="UTF-8"?>
<bpm-platform xmlns="http://www.camunda.org/schema/1.0/BpmPlatform"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.camunda.org/schema/1.0/BpmPlatform
http://www.camunda.org/schema/1.0/BpmPlatform ">
  ...

  <plugins>
    <plugin>
      <class>org.ext.DEP.DataExtensionPlugin</class>

    </plugin>
  </plugins>
  ...

</bpm-platform>
```

Mã nguồn 5.8 - Tập tin bpm-platform.xml

Mã nguồn 5.8 mô tả cách thêm một Plugin vào trong tập tin **bpm-platform.xml** của Camunda, cụ thể là thêm vào DEP. Các thành phần khác của tập tin được giữ nguyên so với tập tin mặc định có sẵn khi tải Camunda về từ trang chủ.

5.3.2. Phương pháp tiếp cận

Từ kiểu kiến trúc Microservice và khả năng mở rộng của Camunda, chúng em quyết định mở rộng cài đặt thêm cho phần cơ sở dữ liệu và thêm dịch vụ dữ liệu cho Google Spreadsheet thông qua cơ chế này.

Trong đó, chúng em sử dụng dịch vụ được Process Engine mặc định của Camunda cung cấp.

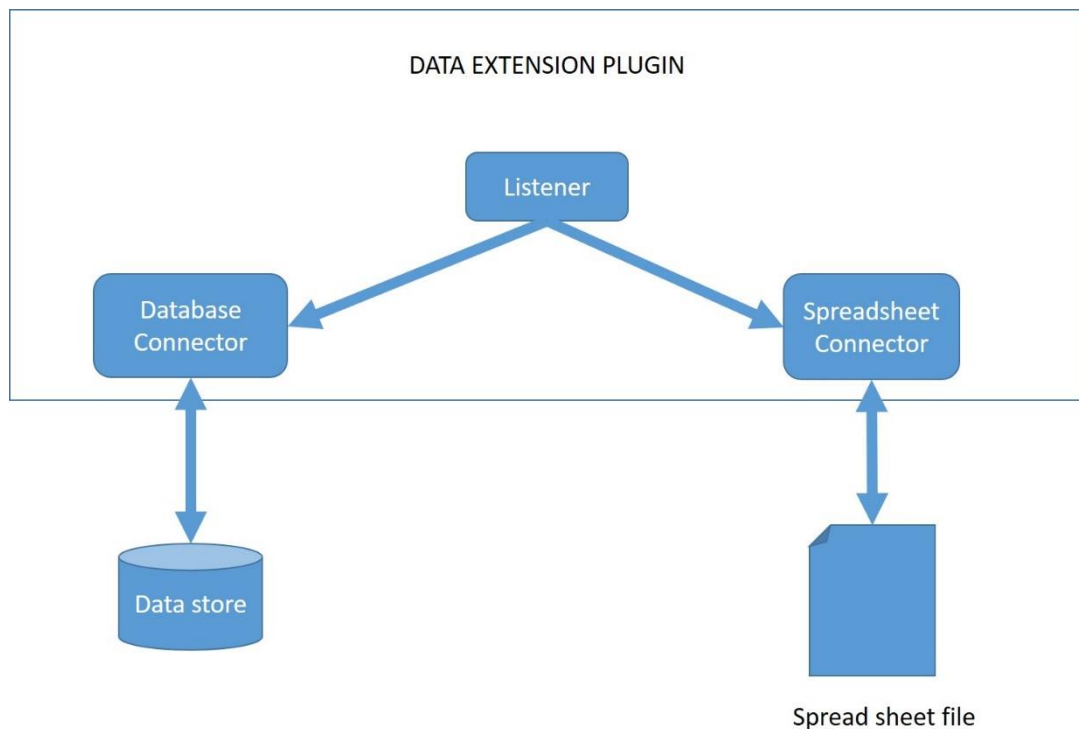
Phương thức thực hiện sẽ chia ra làm 2 luồng chính:

- Phục vụ cho các tác vụ ghi, cập nhật, xóa bỏ các dòng dữ liệu trong cơ sở dữ liệu.
- Phục vụ cho các tác vụ đọc thông tin các dòng dữ liệu trong cơ sở dữ liệu.

Sở dĩ phải phân ra làm 2 hướng do các câu truy vấn SQL của từng hướng được cài đặt khác nhau hoàn toàn về tính chất nên buộc DEP phải có cách xử lí riêng cũng như cách tổ chức dữ liệu riêng cho từng luồng.

5.3.3. Mô hình kiến trúc của DEP

Dưới đây là kiến trúc của mô hình Plugin:

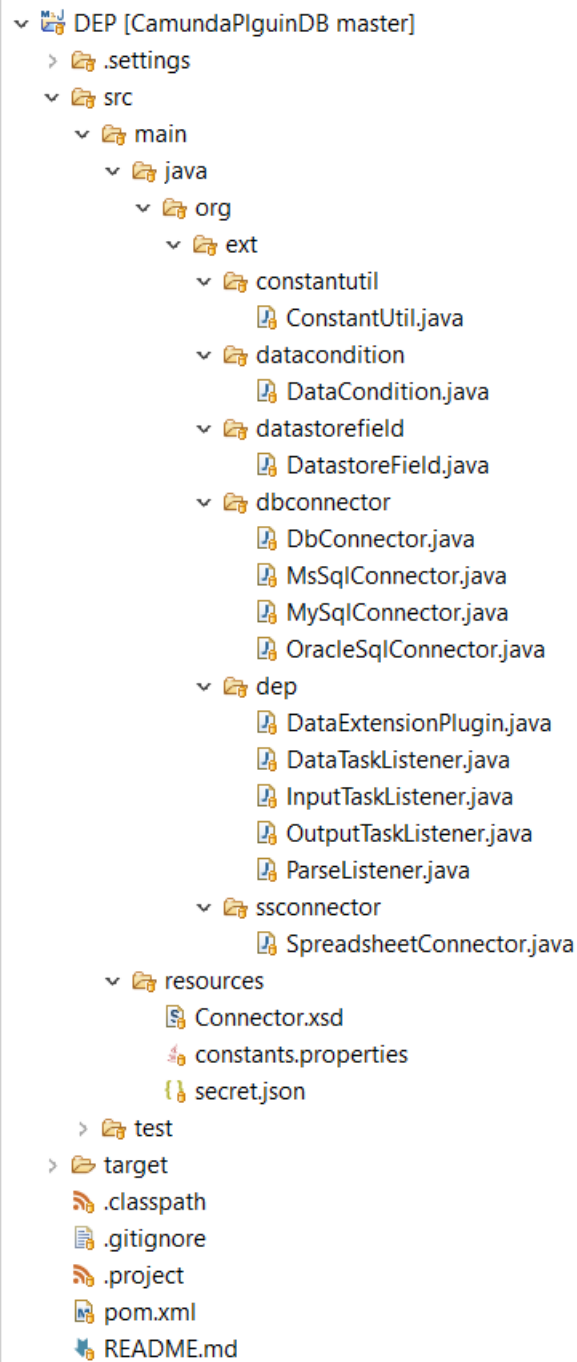


Hình 5.5 – Mô hình Data Extension Plugin (DEP)

Đây là kiến trúc logic tổng quan được thiết kế cho DEP. Trong đó:

- **Listener:** Dùng để lắng nghe các sự kiện sẽ xảy ra với các tác vụ mà người dùng thực hiện để có thể bắt các sự kiện cần thiết. Cũng như chuyển đổi các thông tin từ tập tin **.bpmn** mà người dùng triển khai lên Camunda thành các thành phần mà các Connector có thể đọc được.
- **Database Connector:** Dùng để thao tác dữ liệu với các cơ sở dữ liệu mà chúng em định nghĩa sẵn.
- **Spreadsheet Connector:** Dùng để thao tác với Google Spreadsheet thông qua các API của Google.

5.3.4. Cấu trúc thư mục



Hình 5.6 – Cấu trúc thư mục của DEP

Lý do chọn cách tổ chức thư mục là do kế thừa lại archetype cho Process Plugin của Camunda và đính kèm theo đó là tập tin POM theo chuẩn của Camunda Process Plugin Archetype (tham khảo mã nguồn). Các gói chính cần có trong tập tin POM được liệt kê ở bảng sau:

STT	Tên	Công dụng
1	Camunda Engine	Dùng cho việc giả lập nhân Camunda Engine trong lúc cài đặt
2	Assert J	Dùng cho việc kiểm thử
3	Junit	Dùng cho việc kiểm thử
4	H2 Cơ sở dữ liệu	Dùng cho việc kiểm thử
5	Camunda BPM Process Test Coverage	Dùng cho việc kiểm thử
6	Logback	Dùng cho việc kiểm thử
7	SLF4J	Dùng cho logging
8	MySQL Driver	Dùng cho việc thao tác dữ liệu trên MySQL
9	SQL Server Driver	Dùng cho việc thao tác dữ liệu trên SQL Server
10	Oracle SQL Driver	Dùng cho việc thao tác dữ liệu trên Oracle SQL
11	MySQL Driver	Dùng cho việc thao tác dữ liệu trên MySQL
12	Google API OAuth Client	Dùng cho việc xác nhận thông tin của token (OAuth2)
13	Google API Services Sheets	Dùng cho việc thao tác dữ liệu trên Spreadsheet
14	JSON Simple	Dùng để chuyển đổi dữ liệu dạng JSON

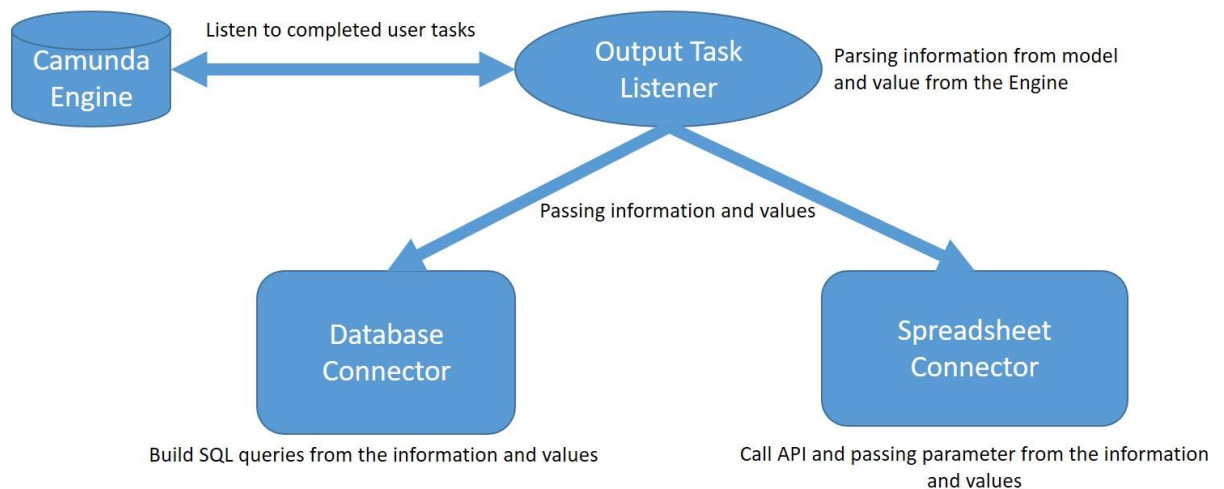
Bảng 5.3 – Các phụ thuộc cơ bản của DEP

Các driver được sử dụng trong **Bảng 5.3** sẽ được sử dụng trong việc kết nối và thao tác dữ liệu với JDBC và các thư viện của Google sẽ hỗ trợ việc cài đặt dịch vụ thao tác với Spreadsheet trên Camunda.

5.3.5. Cách hoạt động của DEP

Như đã đề cập trong **5.3.2** DEP sẽ có 2 luồng xử lý chính cách thức xử lý của từng luồng được mô tả như sau

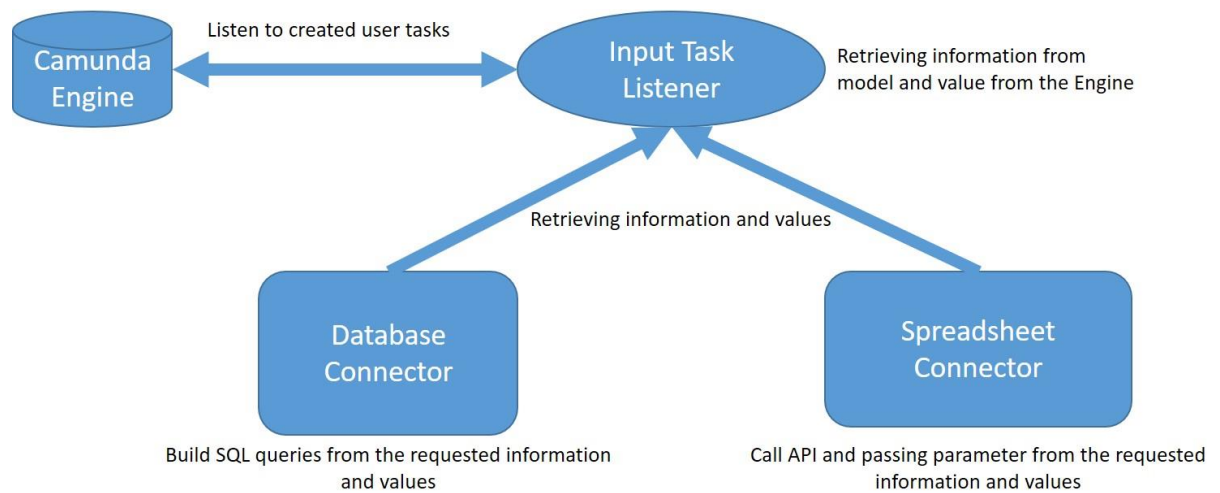
- **Xử lý ghi, cập nhật, xóa bỏ dữ liệu:**



Hình 5.7 – Luồng xử lý của việc ghi, cập nhật và xóa bỏ dữ liệu

Theo **Hình 5.7**, khi người dùng hoàn tất một tác vụ **Camunda Engine** sẽ thông báo sự kiện cho **Output Task Listener** biết để tiến hành các thao tác đọc ghi. **Output Task Listener** sau đó sẽ chuyển đổi các thông tin và giá trị từ các dịch vụ có sẵn của Camunda Engine như **Form Service** rồi sau đó chuyển thành các đối tượng của lớp **Data Condition** và **Data Store Field**. Các đối tượng này sau đó được truyền xuống cho **Database Connector** để xây dựng các câu truy vấn hoặc gọi các API tùy theo yêu cầu của mô hình người dùng triển khai trên Camunda Engine.

- **Xử lý đọc dữ liệu:**



Hình 5.8 – Luồng xử lý của việc đọc dữ liệu

Tuy nhìn có vẻ tương tự như **Hình 5.7**, **Hình 5.8** lại lắng nghe sự kiện các tác vụ **mới khởi tạo** thay vì **mới hoàn thành**. Các thông tin và giá trị khởi tạo trước của tác vụ sẽ được truy vấn từ **Database Connector** và **Spreadsheet Connector** tương tự như **Hình 5.7**, đồng thời cũng sử dụng **Form Service** để tiến hành gán các giá trị vào các biến trong **Camunda Engine**.

5.4. Thiết kế giao diện cho hệ thống thực thi BPMN

5.4.1. Phương pháp tiếp cận

Việc triển khai, thực thi và quản lý các quy trình nghiệp vụ trên Camunda được thực hiện dựa vào REST API. Đồng thời, Camunda cũng cung cấp một thư viện javascript dùng để gọi đến API này. Trong phần cải tiến này, chúng em lựa chọn sử dụng thư viện javascript của Camunda để cải tiến giao diện thực thi của các quy trình nghiệp vụ sử dụng BPMN nhằm hướng đến người sử dụng thông thường, và giao diện người dùng này tương đối giống với một ứng dụng web và có thể giúp phần nào dễ dàng hơn trong quá trình sử dụng

5.4.2. Các thư viện hỗ trợ

Camunda cung cấp một thư viện JavaScript dùng để gọi đến API thực thi các quy trình nghiệp vụ. Trong đó, thư viện này cung cấp một cơ chế truy cập vào Resource của API, cụ thể các Resource đó là:

Tên	Ý nghĩa
process-definition	Truy cập vào các quy trình được định nghĩa và được triển khai trên Camunda
process-instance	Truy cập vào những Instance của những quy trình đang chạy
task	Truy cập vào các Task hiện tại trên một quy trình nào đó

Bảng 5.4 – Các tài nguyên truy cập

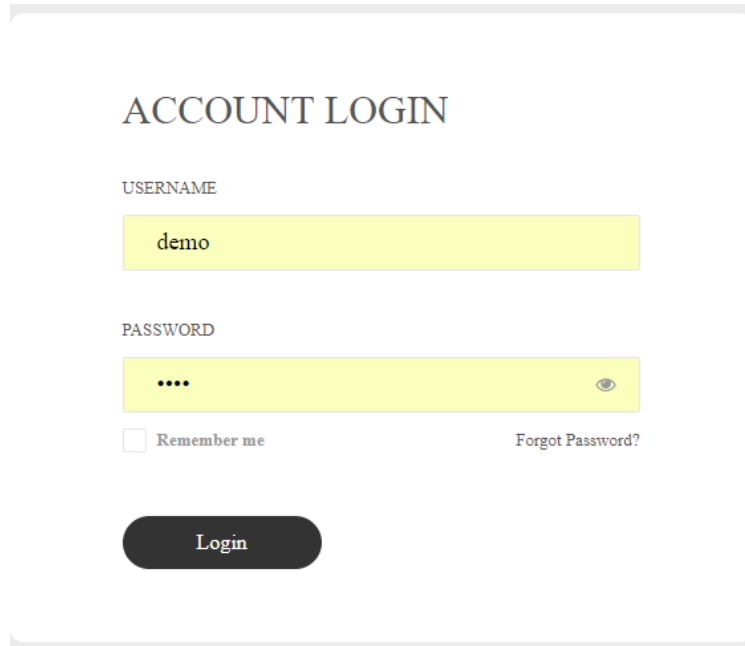
Những Resource này cung cấp những phương thức cho phép chúng ta truy vấn được dữ liệu của các nguồn tài nguyên này, những phương thức đó là:

Tên	Ý nghĩa
list	Lấy danh sách các tài nguyên theo một điều kiện cho trước
delete	Xóa các nguồn tài nguyên theo một điều kiện cho trước

Bảng 5.5 – Thao tác trên các tài nguyên

Ngoài ra, Thư viện này cung cấp một phương thức truy cập vào Form của một Task, phương thức này cho phép lấy được nội dung HTML của các Form từ API và sử dụng để Render lên giao diện thực thi quy trình

5.4.3. Mô tả giao diện mới và cách thức thực hiện



ACCOUNT LOGIN

USERNAME

demo

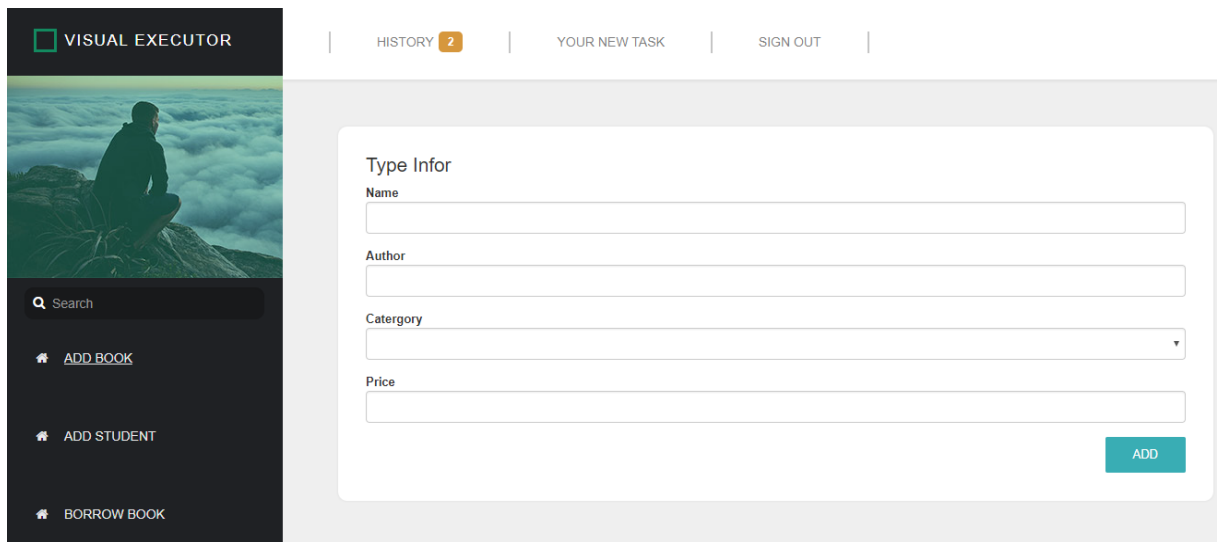
PASSWORD

....

☐ Remember me [Forgot Password?](#)

Login

Hình 5.9 – Giao diện đăng nhập



VISUAL EXECUTOR

HISTORY **2** | YOUR NEW TASK | SIGN OUT

Type Infor

Name

Author

Catergory

Price

ADD

Search

ADD BOOK

ADD STUDENT

BORROW BOOK

Hình 5.10 – Giao diện chính

Camunda có một hệ thống thực thi các quy trình nghiệp vụ BPMN mạnh nhưng lại tồn tại những khó khăn trong việc sử dụng vì hệ thống này còn thiên về xử lý nghiệp vụ nên chúng em lựa chọn thay đổi các phần như sau:

❖ **Cách thức trình bày các quy trình**

Do Camunda thực thi các quy trình thiên về nghiệp vụ nên sử dụng các thuật ngữ dành riêng cho nghiệp vụ, trong phần này, để thực thi được một quy trình, người dùng phải trải qua các bước sau: chọn start process, sau đó chọn quy trình và xác nhận chạy quy trình. Để đơn giản hơn cho thao tác này, chúng em đã thiết kế một giao diện mới, tại đây client sẽ gọi đến API của Camunda yêu cầu lấy các quy trình đã được định nghĩa trong hệ thống sau đó hiển thị thành một danh sách cho người dùng chọn, khi người dùng chọn một quy trình, quy trình đó sẽ được chạy.

❖ **Cách thức thực hiện các công việc trên quy trình**

Khi Camunda thực hiện một quy trình, từng task sẽ được liệt kê vào danh sách, người dùng khi thực hiện xong một task cần phải chọn task tiếp theo để thực hiện. Để đơn giản hóa cho việc này, chúng em sẽ cho người dùng thực hiện các task trong một quy trình một cách liên tục, nghĩa là khi người dùng hoàn thành một task thì client sẽ gọi API yêu cầu thông tin của các task tiếp theo để hiển thị cho người dùng, nhờ đó, thao tác của người dùng trở nên đơn giản và ít thao tác hơn.

❖ **History và New Task:** Khi người dùng thực thi một quy trình, các thông tin của người dùng trên quy trình sẽ được lưu lại. Trong trường hợp người dùng thực thi nhưng chưa kết thúc một quy trình, thông tin của các task sẽ được lưu lại dưới dạng lịch sử người dùng, giúp cho người dùng dễ dàng hơn trong việc truy vết lại các thao tác đã được thực thi. Trong trường hợp các task này do một người dùng khác giao cho thì nó được liệt kê vào danh

sách newtask, cho phép người dùng thực hiện các task do người khác giao cho.

- ❖ **Đăng nhập:** Camunda sử dụng Basic Authentication để xác thực cũng như phân quyền cho các người dùng mà không cung cấp một cơ chế xác thực thông qua API do đó khi chúng em thiết kế giao diện cho phần đăng nhập, khi người dùng nhập thông tin đăng nhập, client sẽ dùng thông tin này để gọi đến API, nếu API trả về mã lỗi 401 thì ứng dụng sẽ nhận biết là xác thực người dùng không thành công và yêu cầu đăng nhập lại

CHƯƠNG 6: KẾT LUẬN

Trong chương này chúng em sẽ trình bày về những kết quả mà nhóm chúng em đạt được trong quá trình cài đặt cũng như những phần chưa đáp ứng một cách mạnh mẽ khi thực hiện việc cải tiến này. Đồng thời đưa ra hướng phát triển cho đề tài trong tương lai.

6.1. Kết quả đạt được

Sau khi kết thúc luận văn, chúng em đã thu được một số kết quả nhất định sau:

- Hiểu biết thêm về BPMN, Camunda và các công cụ hỗ trợ thực thi BPMN
- Hiểu biết thêm về Microservice, cách thiết kế, cài đặt và triển khai
- Cài đặt thành công DEP để có thể triển khai trên hệ thống Camunda
- Nâng cấp công cụ Camunda Modeler để có thể hỗ trợ thiết kế cho DEP
- DEP có thể chạy được trên 3 cơ sở dữ liệu theo yêu cầu
- DEP có thể thực hiện các thao tác đơn giản trên các cơ sở dữ liệu được yêu cầu
- Tạo được một hệ thống giao diện mới sử dụng các REST API của Camunda để một phần cải thiện trải nghiệm người dùng thay cho Camunda Tasklist

6.2. Những khó khăn trong quá trình thực hiện

Tuy những kết quả thu được có phần khả quan, việc gặp những khó khăn trong quá trình thực hiện luận văn là không thể tránh khỏi. Sau đây là một số khó khăn chủ yếu chúng em đã gặp phải trong quá trình thực hiện luận văn:

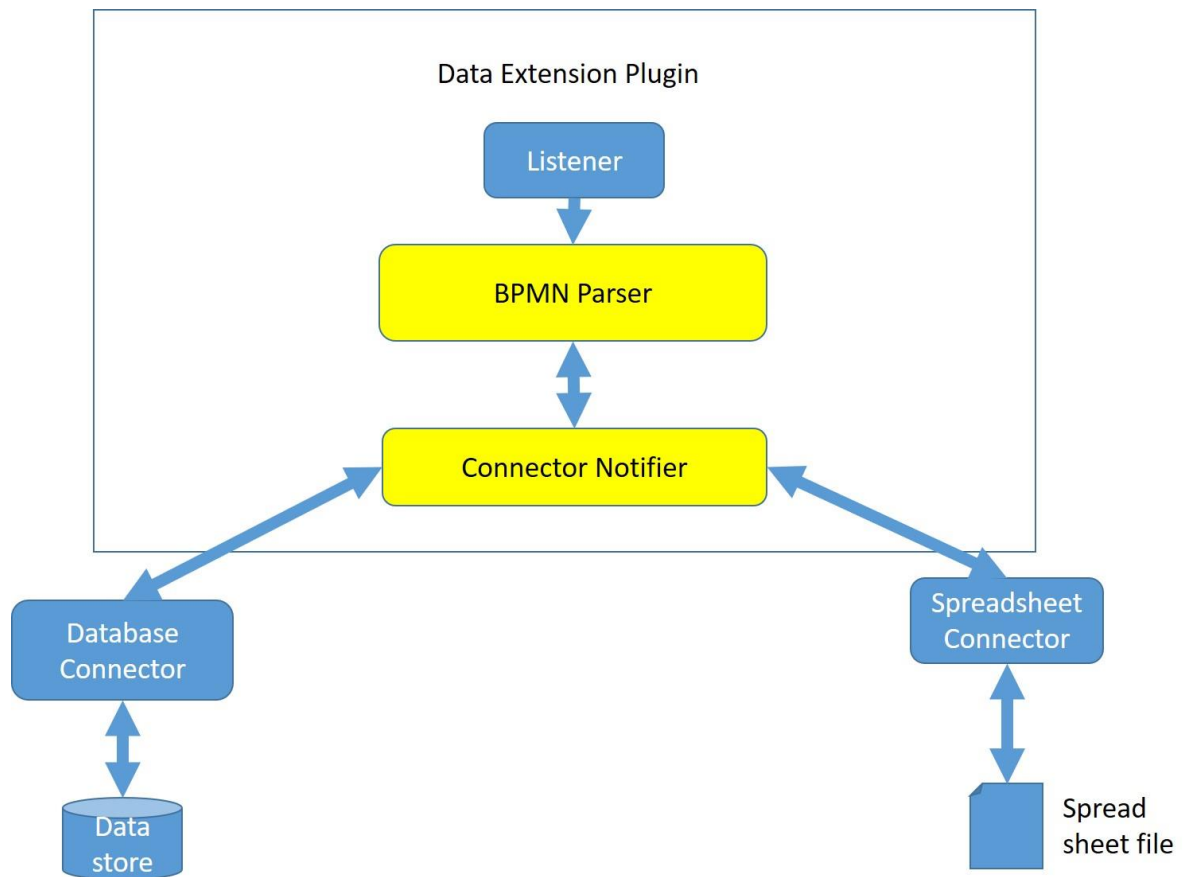
- Thiếu kinh nghiệm trong việc thiết kế kiến trúc, dẫn đến việc kiến trúc của DEP trở thành kiến trúc một khối như hiện tại
- Modeler hiện tại được nâng cấp bằng cách thay đổi mã nguồn của Modeler cũ do chưa tìm được cách để triển khai các thay đổi trên theo kiểu Plugin

- Việc chuyển đổi thông tin từ tập tin .bpmn sang các đối tượng trong DEP được thực hiện thủ công và không có file XSD riêng
- Các tài liệu hướng dẫn có liên quan đến các thành phần dịch vụ của Camunda Engine còn hạn chế dẫn đến việc khó khăn trong cài đặt

6.3. Hướng phát triển đề tài

Việc cải tiến một cách hoàn toàn đáp ứng tối đa nhu cầu của người dùng là một công việc mất nhiều thời gian và chi phí. Trong giới hạn về thời gian và nhân lực, chúng em đã cải tiến một số chức năng của Camunda để đáp ứng được nhu cầu của người dùng, tuy nhiên, bên cạnh đó vẫn còn một số hạn chế. Sau đây là một số định hướng về mặt lý thuyết cũng như ứng dụng trong tương lai:

- Tiếp tục nghiên cứu sâu hơn về các vấn đề của BPMN cùng với những công cụ thực thi BPMN hiện tại
- Nghiên cứu về mặt công nghệ giúp cho kiến trúc của những tính năng đã cải tiến trở nên linh hoạt hơn, dễ dàng mở rộng những tính năng mới trong thời gian sắp tới
- Về mặt ứng dụng, tiếp tục hoàn thiện và mở rộng các tính năng đã cải tiến như:
 - **Modeler**: hiện tại chỉ hỗ trợ các câu truy vấn đơn giản do tập trung vào tính trực quan của công cụ, có thể thêm tính năng nhập câu SQL nâng cao dành riêng cho người dùng có chuyên môn về mặt kỹ thuật
 - **DEP**: Cũng như **Modeler**, DEP cũng chỉ hỗ trợ những câu lệnh SQL đơn giản (chưa có JOIN, INNER JOIN, ...). Đồng thời kiến trúc của DEP vẫn có thể nâng cấp lên để tách riêng việc chuyển đổi thông tin (*Parsing*) và các **Connector** ra khỏi DEP để cài đặt thành các dịch vụ chạy riêng. Mô hình kiến trúc có thể hướng tới như sau:



Hình 6.1 – Sơ đồ kiến trúc của DEP trong tương lai

Với mô hình như vậy, các Connector được tổ chức và chạy độc lập với nhau, giúp cho việc thiết kế và cài đặt thêm Connector mới trở nên dễ dàng hơn và giảm thiểu tỉ lệ fail-over (nếu có).

- **Giao diện:** trong quá trình thực thi, người dùng có thể gán được công việc cho những người có liên quan
- **Deploy:** có thể thay đổi được cách thức mà người dùng có thể deploy một quy trình BPMN, thay vì phải gói tất cả các mô hình BPMN và các nguồn tài nguyên liên quan, người dùng có thể chọn từng file BPMN và các tài nguyên để deploy mà không cần phải đóng gói

DANH MỤC THAM KHẢO

- [1] Chris Richardson, Floyd Smith, “Microservices From Design To Deployment”, NGINX, 2016
- [2] Cristina Venera, “BPMN VS. UML ACTIVITY DIAGRAM FOR BUSINESS PROCESS MODELING”, 2012
- [3] Daniela C. C. Peixoto, Vitor A. Batista, Ana P. Atayde, Eduardo P. Borges, Rodolfo F. Resende, Clarindo Isaías P. S. Pádua, “A Comparison of BPMN and UML 2.0 Activity Diagrams”, 2008
- [4] Gregor Polančič, BPMN 2.0 Task Types Explained, 2013
- [5] <http://www.bpm-guide.de/2016/10/19/5-reasons-to-switch-from-activiti-to-camunda/>
- [6] <https://camunda.com/bpmn/reference/>
- [7] <https://docs.camunda.org/manual/7.8/introduction/architecture/>
- [8] <https://docs.camunda.org/manual/7.8/reference/bpmn20/>
- [9] <https://docs.jboss.org/jbpm/v5.0/userguide/ch04.html>
- [10] https://en.wikipedia.org/wiki/Business_Process_Model_and_Notation
- [11] <https://keniasousa.github.io/2014/10/flowchart-bpmn.html>
- [12] <https://medium.com/capital-one-developers/comparing-and-contrasting-open-source-bpm-projects-196833f23391>
- [13] <https://www.mcftech.com/use-bpmn-flowcharts/>
- [14] https://www.openhub.net/p/_compare?project_0=camunda+BPM+platform&project_1=Activiti&project_2=Flowable
- [15] Mark von Rosing, Stephen White, Fred Cummins, Henk de Man, The complete business process handbook
- [16] Martin Owen and Jog Raj, Popkin Software, BPMN and Business Process Management

[17] Nguyễn Thị Ngọc Hoa, Nguyễn Thúy Ngọc, Phạm Thị Thanh Phương, Phạm Thủy Tú, “TÌM HIỂU BPMN - (BUSINESS PROCESS MODEL NOTATION)”, 2012