HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Phan Hồng Dương

ỨNG DỤNG CÔNG NGHỆ WEB NGỮ NGHĨA VÀ OPENSTREETMAP PHÁT TRIỂN ỨNG DỤNG HỖ TRỢ TÌM KIẾM THÔNG TIN NGƯỜI BỆNH COVID-19

LUẬN VĂN THẠC SĨ KỸ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI - 2021

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Phan Hồng Dương

ỨNG DỤNG CÔNG NGHỆ WEB NGỮ NGHĨA VÀ OPENSTREETMAP PHÁT TRIỂN ỨNG DỤNG HỖ TRỢ TÌM KIẾM THÔNG TIN NGƯỜI BỆNH COVID-19

CHUYÊN NGÀNH : HỆ THỐNG THÔNG TIN

MÃ SỐ : 8.48.01.04

LUẬN VĂN THẠC SĨ KỸ THUẬT

(***Theo định hướng ứng dụng***)

NGƯỜI HƯỚNG DẪN KHOA HỌC: PGS.TS. HOÀNG HỮU HẠNH

HÀ NỘI - 2021

LỜI CAM ĐOAN

Tôi là Phan Hồng Dương, học viên lớp M19CQIS02-B xin cam đoan báo cáo luận văn này được viết bởi tôi dưới sự hướng dẫn của thầy PGS. TS Hoàng Hữu Hạnh. Trong toàn bộ nội dung của luận văn, những điều được trình bày là kết quả của cá nhân tôi hoặc là được kế thừa, tổng hợp từ nhiều nguồn tài liệu khác được liệt kê trong danh mục tài liệu tham khảo rõ ràng.

Hà Nội, ngày 03 tháng 03 năm 2021

Học viên thực hiện

Phan Hồng Dương

LỜI CẢM ƠN

Lời đầu tiên, tôi xin bày tỏ sự cảm ơn chân thành đối với thầy giáo PGS.TS Hoàng Hữu Hạnh - Giáo viên hướng dẫn trực tiếp của tôi. Thầy đã giúp tôi tiếp cận những kiến thức về công nghệ Web ngữ nghĩa cũng như bản đồ thế giới mở OpenStreetMap trong suốt quá trình nghiên cứu và hoàn thiện luận văn thạc sĩ.

Tôi xin gửi lời cảm ơn tới các các thầy, cô khoa Công nghệ thông tin trường Học viện Công nghệ Bưu chính Viễn thông với sự quan tâm, dạy dỗ, chỉ bảo tận tình chu đáo của thầy, cô, đến nay tôi đã có thể hoàn thành luận văn.

Đặc biệt, để hoàn thành luận văn này, tôi đã nhận được sự giúp đỡ từ cơ quan, gia đình, bạn bè. Tôi xin chân thành cảm ơn tất cả đã hỗ trợ động viên và tạo điều kiện hết sức thuận lợi cho tôi.

Tôi xin chân thành cảm ơn!.

Hà Nội, ngày 06 tháng 03 năm 2021

Học viên thực hiện luận văn kỹ thuật

Phan Hồng Dương

MỤC LỤC

LỜI CAM ĐOAN ..... **i**
LỜI CẢM ƠN ii
DANH MỤC CÁC THUẬT NGỮ VÀ CHỮ VIẾT TẮT ..... **vi**
DANH MỤC CÁC HÌNH VẼ VÀ ĐỒ THỊ ..... **vii**
MỞ ĐẦU 1
CHƯƠNG 1: KIẾN THỨC CƠ SỞ **3**
1.1 Web ngữ nghĩa 3
1.1.1 Khái niệm về Web ngữ nghĩa 3
1.1.2 Kiến trúc Web ngữ nghĩa 4
1.2 RDF, RDFS 5
1.2.1 Giới thiệu RDF 5
1.2.2 Giới thiệu về RDFS 9
1.3 Các ngôn ngữ truy vấn của Web ngữ nghĩa 12
1.3.1 Simple Protocol And RDF Query Language (SPARQL) 12
1.4 LINKED DATA 14
1.4.1 Khái niệm 14
1.4.2 Các quy tắc triển khai Linked Data 15
1.4.3 Truy vấn Web dữ liệu 15
1.5 Tiểu kết chương 1 16
CHƯƠNG 2: LẬP TRÌNH VỚI WEB NGỮ NGHĨA **17**
2.1 Môi trường lập trình Web ngữ nghĩa 17
2.1.1 Môi trường lập trình Intellij 17
2.1.2 Jena 19

2.2 Mô hình hóa thông tin với Jena 21
2.2.1 Chương trình Hello Word! trong Jena 21
2.2.2 Tạo mô hình RDF 22
2.2.3 Đọc mô hình RDF 26
2.3 Chuyển đổi dữ liệu Web ngữ nghĩa 28
2.4 Framework Spring MVC 28
2.4.1 Giới thiệu về Framework Spring MVC 28
2.4.2 Ví dụ trang web Spring MVC cơ bản 29
2.5 Tiểu kết chương 2 30
CHƯƠNG 3: ỨNG DỤNG TÌM KIẾM THÔNG TIN BỆNH NHÂN COVID-19 TẠI VIỆT NAM 31
3.1 Giới thiệu bài toán tìm kiếm thông tin bệnh nhân COVID-19 31
3.1.1 Kiến trúc hệ thống 31
3.1.2 Ý nghĩa thực tiễn của hệ thống 32
3.1.3 Chuẩn bị dữ liệu cho bài toán 35
3.2 Giải pháp hệ thống 45
3.2.1 Mô hình hệ thống 45
3.2.2 Tạo dữ liệu RDF 46
3.2.3 Truy vấn dữ liệu 47
3.3 Xây dựng hệ thống 47
3.3.1 Mô tả hệ thống 47
3.3.2 Đặc tả chức năng 49
3.3.3 Phân tích hệ thống 50
3.3.4 Thiết kế giao diện 54

3.4 Kết luận chương.....56
KẾT LUẬN 57
DANH MỤC CÁC TÀI LIỆU THAM KHẢO..... 58
BẢN CAM ĐOAN 67

DANH MỤC CÁC THUẬT NGỮ VÀ CHỮ VIẾT TẮT

Viết tắt	Tiếng Anh	Tiếng Việt
RDF	Resource Description Framework	Framework mô tả tài nguyên
URL	Uniform Resource Locator	Định vị tài nguyên thống nhất
URI	Uniform Resource Identifier	Định dạng tài nguyên thống nhất
XML	eXtensible Markup Language	Ngôn ngữ đánh dấu mở rộng
OWL	Web Ontology Language	Ngôn ngữ bản thể học
HTTP	HyperText Transfer Protocol	Giao thức truyền tải siêu văn bản
HTML	Hypertext Markup Language	Ngôn ngữ đánh dấu siêu văn bản
OSM	OpenStreetMap	Bản đồ đường sá mở
SPARQL	Simple Protocol And RDF Query Language	Giao thức và ngôn ngữ truy vấn RDF
JDT	Java Development Toolkit	Bộ công cụ phát triển Java

DANH MỤC CÁC HÌNH VẼ VÀ ĐỒ THỊ

Hình 1.1. Kiến trúc Web ngữ nghĩa 4

Hình 1.2. Mô hình một bộ ba RDF (RDF triple) 6

Hình 1.3. Mối quan hệ giữa các thành phần trong bộ ba RDF 7

Hình 1.4. Đồ thị RDF 7

Hình 1.5. Cấu trúc phân cấp của các lớp.....10

Hình 1.6. Cấu trúc phân cấp các thuộc tính 10

Hình 1.7. Các tầng RDF và RDFS 11

Hình 1.8. Các lớp và các thuộc tính của RDF.....11

Hình 2.1. Giao diện Intellij 17

Hình 2.2. Các thành phần của Jena API.....20

Hình 2.3. Kiến trúc Jena 3.....21

Hình 2.4. Thêm thư viện Jena 21

Hình 2.5. Biểu diễn phát biểu dưới dạng đồ thị 23

Hình 2.6. Kết quả trên giao diện console. 23

Hình 2.7. Dữ liệu xuất dưới dạng Turtle.....27

Hình 2.8. Các thành phần trong Spring.....29

Hình 3.1. Kiến trúc chương trình 32

Hình 3.2. Thống kê dịch COVID-19 tại Việt Nam tính đến ngày 01/09/2021.....33

Hình 3.3. Mô tả mối liên hệ giữa các bệnh nhân COVID-19 với F0.....34

Hình 3.4. Mô tả mối liên hệ giữa các F0 với địa điểm 35

Hình 3.5. Dữ liệu về thông tin người bệnh 36

Hình 3.6. Dữ liệu về thông tin địa điểm.....36

Hình 3.7. Dữ liệu về thông tin sự kiện 37

Hình 3.8. Kiến trúc OpenStreetMap 39

Hình 3.9. Trích xuất dữ liệu OpenStreetMap.....40

Hình 3.10. Đọc dữ liệu từ OpenStreetMap với JOSM.....41

Hình 3.11. Dữ liệu Học viện Công nghệ Bưu chính Viễn thông trên OpenStreetMap 42

Hình 3.12. Hiển thị thông tin một node dựa vào OSM_ID.....43

Hình 3.13. LinkedGeoData.org.....44

Hình 3.14. Kết quả hiển thị Linkedgeodata.org trên HTML 45

Hình 3.15. Mô hình hệ thống tra cứu thông tin bệnh nhân.....45

Hình 3.16. Mô hình phát triển hệ thống.....47

Hình 3.17. Use case hệ thống tìm kiếm thông tin bệnh nhân COVID-19 48

Hình 3.18. Sơ đồ kiến trúc thông tin.....48

Hình 3.19. Biểu đồ tuần tự chức năng thống kê.....50

Hình 3.20. Biểu đồ tuần tự chức năng xem lịch trình di chuyển của bệnh nhân.....51

Hình 3.21. Biểu đồ tuần tự chức năng xem danh sách bệnh nhân.....53

Hình 3.22. Giao diện trang thống kê.....54

Hình 3.23. Giao diện tra cứu thông tin lịch trình di chuyển của người bệnh 55

Hình 3.24. Giao diện chi tiết thông tin người bệnh COVID-19 56

MỞ ĐẦU

Trong các ghi chép của người đề xuất ra ý tưởng - Tim Berners-Lee, Web ngữ nghĩa được mô tả là xu thế và là thế hệ Web tiếp theo, khắc phục được những hạn chế của web hiện tại, đáp ứng tốt hơn nhu cầu của người sử dụng, cung cấp lượng thng tin đầy đủ, toàn diện, đáng tin cậy và có ý nghĩa hơn với người dùng. Sự ra đời của thế hệ web thứ ba – Web ngữ nghĩa (hay Web ngữ nghĩa) do W3C khởi xướng đã mở ra một bước tiến của công nghệ web giúp giải quyết được thách thức trên.

Một trang web thông thường được biểu diễn bằng HTML, các chương trình tìm kiếm hiện tại sẽ không hiểu được thông tin trên file HTML đó. Ví dụ ta có một thẻ trong trang html là HE IS JOHN. Một chương trình tìm kiếm thông thường đọc được đoạn đó, nó sẽ tách thành các từ: HE, IS, JOHN, coi chúng là các từ khóa tiến hành so sánh, mà không hiểu được nghĩa của mỗi từ và các thông tin liên quan. Web ngữ nghĩa thì khác, đoạn văn bản trên sẽ được trình bày trong những mối liên hệ, như "JOHN" là thể hiện của một thuộc tính có tên là "TÊN", thuộc tính này nằm trong một lớp có tên là "DIỄN VIÊN". Bằng những định nghĩa và thuộc tính này, thay vì phải tìm kiếm theo các từ khóa, các hệ thống tìm kiếm sẽ tìm trong không gian các mối liên hệ phụ thuộc, và sẽ cho ra những kết quả liên quan đến các định nghĩa, liên hệ đó. Như vậy, các hệ thống tìm kiếm sẽ cho ra kết quả chính xác hơn so với tìm kiếm dựa theo từ khóa.

Dựa trên nền tảng XML, các thông tin trong Web ngữ nghĩa sẽ dễ dàng được trao đổi với nhau, bởi các hệ thống đều biết cách "đọc" các cấu trúc dữ liệu dựa trên XML. Các tầng phía trên của Web ngữ nghĩa cho phép định nghĩa, chú giải các thẻ XML. Như vậy, ở mức cao hơn XML, Web ngữ nghĩa không những cho phép trao đổi thông tin mà nó còn cho phép máy tính có thể hiểu được một phần thông tin trong đó và tự động thực hiện một số công đoạn thay con người. Dựa trên nền tảng lô-gic, các đối tượng, thuộc tính sẽ được trình bày bằng các luật lô-gic, điều này sẽ cho phép các chương trình suy luận, lập luận, và chứng minh dựa trên các biểu diễn bằng lô-gic trong Web ngữ nghĩa. Đây chính là điều mà web hiện tại không thể làm được.

OpenStreetMap, hoặc OSM, giống như Wikipedia cho bản đồ. Đó là nguồn mở, người dùng đã đóng góp, và miễn phí để sử dụng, và, giống như Wikipedia, nó được chứng minh là đáng tin cậy đáng kinh ngạc. Ngay cả khi bạn chưa từng nghe về nó trước đây, rất có thể là bạn đã sử dụng dữ liệu của nó. Nó được sử dụng bởi FourSquare, Pokemon Go, Craigslist, Tesla, và một danh sách dài các ứng dụng và dịch vụ khác. Trong khi nó không phải là hữu ích cho chuyển hướng hàng ngày, bộ dữ liệu không gian địa lý rộng lớn của nó đã nhiều lần chứng minh hữu ích không chỉ cho các ứng dụng, nhưng cho công việc nhân đạo và cứu trợ khẩn cấp.

Sự bùng phát dịch COVID-19 đã mang lại những thách thức chưa từng có, được dự báo sẽ có những tác động đáng kể đến sự phát triển nền kinh tế Việt Nam trong năm nay. Dựa trên tình hình hiện tại, chúng tôi tìm hiểu các tác động tiềm ẩn của đợt bùng phát đại dịch COVID-19 đối với kinh tế Việt Nam. Nghiên cứu này đi đôi với một mức độ đáng kể các yếu tố không chắc chắn. Cụ thể là, khi xảy ra đại dịch COVID-19, các dự đoán được kiểm tra lại và điều chỉnh lại mỗi tuần kể từ khi bắt đầu xảy ra đợt bùng phát đại dịch.

Nền kinh tế Việt Nam thường phụ thuộc nhiều vào các nền kinh tế khác. Do vậy, các kịch bản và dự đoán liên quan đến các tác động đối với kinh tế Việt Nam cũng tương quan với các tác động đối với kinh tế của các nước khác sau đợt bùng phát đại dịch COVID-19. Đại dịch COVID-19 hiện nay vẫn đang diễn biến nhanh chóng với số ca nhiễm mới không ngừng gia tăng hàng ngày. Một câu hỏi đặt ra là việc truy xuất các thông tin về các ca nhiễm, nghi nhiễm và tìm kiếm các thông tin liên quan trên nền Web dữ liệu. Đề tài này hướng được việc ứng dụng công nghệ Web ngữ nghĩa trong việc phát triển các ứng dụng Web dữ liệu và hệ thống dữ liệu bản đồ mở - OpenStreetMap nhằm kết nối các datasets để hỗ trợ hiệu quả việc tìm kiếm thông tin dịch COVID-19 và các ca nhiễm, nghi nhiễm tại Việt Nam và đặc biệt tại Hà Nội và vùng phụ cận, cung cấp them các thông tin liên quan đến đến COVID-19 để cho người sử dụng hệ thống. Từ những vấn đề nêu trên, tôi chọn đề tại luận văn là: *"Ứng dụng công nghệ Web ngữ nghĩa và OpenStreetMap phát triển ứng dụng hỗ trợ tìm kiếm thông tin bệnh nhân COVID-19"*.

CHƯƠNG 1: KIẾN THỨC CƠ SỞ

Nội dung chương này sẽ bao gồm giới thiệu chung về công nghệ Web ngữ nghĩa, những khó khăn và ý nghĩa của công nghệ này. Chương này cũng trình bày về các nghiên cứu liên quan như khái niệm, ý nghĩa của Ontology và ngôn ngữ truy vấn. Từ những cơ sở nghiên cứu này ta sẽ có thể xác định rõ hướng nghiên cứu của luận văn.

1.1 Web ngữ nghĩa

1.1.1 Khái niệm về Web ngữ nghĩa

Web ngữ nghĩa không là một phiên bản Web riêng biệt mà là một sự mở rộng của Web hiện tại, theo cách mà các thông tin được xác định ý nghĩa tốt hơn, cho phép máy tính và người cộng tác với nhau một cách tốt hơn. Web ngữ nghĩa được hình thành từ ý tưởng của Tim Berners-Lee, người phát minh ra WWW (World Wide Web), URI (Uniform Resource Identification), HTTP, và HTML. Web ngữ nghĩa là mạng lưới các nguồn thông tin được liên kết sao cho chúng có thể được xử lý và tím kiếm dễ dàng bởi các máy tính ở phạm vi toàn cầu. Web ngữ nghĩa được xem như là một cách mô tả thông tin rất hiệu quả trên World Wide Web, và cũng được xem là một cơ sở dữ liệu có khả năng liên kết với mạng lưới toàn cầu. Web ngữ nghĩa là phương pháp cho phép người dùng định nghĩa và liên kết dữ liệu một cách có ngữ nghĩa hơn nhằm phục vụ cho máy tính có thể "hiểu" và tìm kiếm được.Ngoài ra Web ngữ nghĩa còn cung cấp môi trường được chia sẻ và xử lý dữ liệu tự động.

Ví dụ: Ta cần phải so sánh giá cả để mua một bộ máy vi tính hoặc ta cần tra cứu mẫu của các hãng chế tạo xe máy khác nhau để tìm ra thiết bị có thể thay thế cho các bộ phận bị hư hỏng. Các thông tin mà ta thu thập được trực tiếp trên các Web có thể trả lời các câu hỏi này nhưng cần con người có sự phân tích ý nghĩa của dữ liệu từ chúng và sự liên quan với yêu cầu đề ra và không thể xử lý tự động được bằng máy tính.

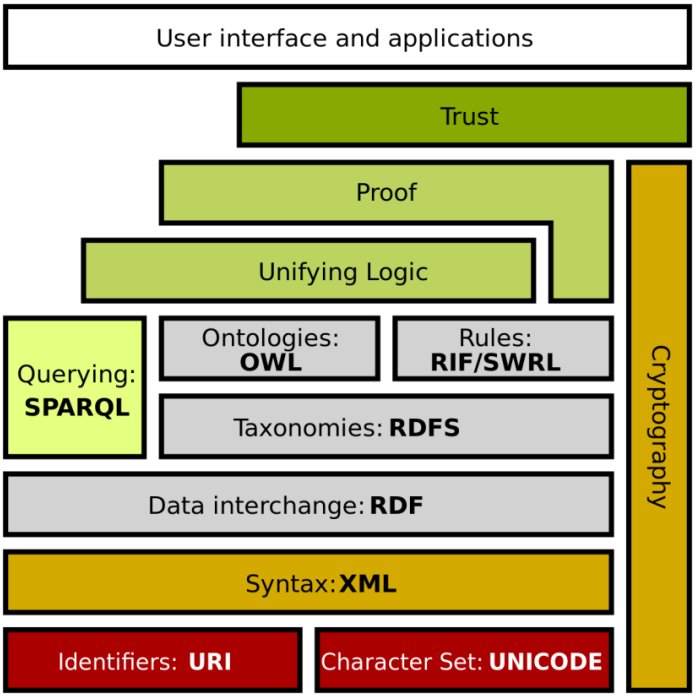
Với Web ngữ nghĩa ta có thể giải quyết vấn đề này bằng 2 cách:

Thứ nhất: Web ngữ nghĩa sẽ mô tả chi tiết những dữ liệu theo cách riêng. Vì vậy ta không cần quan tâm đến các định dạng (format) hay hình ảnh, quảng cáo trên trang Web để tìm ra sự liên quan của các thông tin.

Thứ hai: Web ngữ nghĩa cho phép chúng ta tạo ra một tệp mô tả các mối liên hệ giữa những tập dữ liệu khác nhau. Ví dụ: chúng ta có thể tạo ra một liên kết có ngữ nghĩa giữa cột mã số quốc gia (zip code) trong cơ sở dữ liệu chung với trường "zip" ở trên giao diện nhập liệu nếu chúng có chung ý nghĩa. Điều này cho phép máy tính có thể theo các đường liên kết và có thể tích hợp dữ liệu từ nhiều nơi khác nhau. Sự liên kết các nguồn khác nhau (khái niệm, hình ảnh, video, tài liệu,…) với tập các mối quan hệ mới giữa các nguồn dữ liệu, cho phép có thể mở rộng Web thành một môi trường mới, tạo ra các mối liên hệ ngữ cảnh, điều mà các Web hiện tại không có.

1.1.2 Kiến trúc Web ngữ nghĩa

Web ngữ nghĩa là tập hợp của các ngôn ngữ. Các lớp Web ngữ nghĩa sử dụng để có thể đảm bảo độ an toàn và các giá trị của thông được trở nên tốt nhất.



Hình 1.1. Kiến trúc Web ngữ nghĩa

- *Lớp Unicode & URI*: Bảo đảm việc sử dụng các tập kí tự quốc tế và cung cấp các phương tiện nhằm định danh đối tượng trong Web ngữ nghĩa. URI đơn giản chỉ là một định danh Web giống như các chuỗi bắt đầu bằng "http" hay "ftp" mà bạn thường xuyên thấy trên mạng (ví dụ: http://www.ptit.edu.vn). Bất kỳ ai cũng có thể tạo một URI, và có quyền sở hữu chúng. Vì vậy đã hình thành nên nền tảng để xây dựng một hệ thống mạng toàn cầu thông qua chúng.

- *Lớp XML* cùng với những định nghĩa về không gian tên (namespace) và lược đồ (schema) để bảo đảm rằng chúng ta có thể tích hợp các định nghĩa về Web ngữ nghĩa với các chuẩn XML khác.

- *Lớp RDF và RDFS:* có thể tạo các phát biểu (statement) với những bộ từ vựng và định nghĩa của URI để có thể mô tả các đối tượng. Đây là lớp lớp quan trọng nhất trong kiến trúc của hệ thống Web ngữ nghĩa, từ đó chúng ta có thể gán các kiểu cho các tài nguyên và liên kết.

- *Lớp Ontology:* thực hiện định nghĩa mối liên hệ giữa các khái niệm với nhau, hỗ trợ các bộ từ vựng. Bản thể luận trong logic (Ontology) định nghĩa bộ từ vựng mang tính thông thường và phổ biến, cho phép các nhà nghiên cứu chia sẻ thông tin của một hay nhiều lĩnh vực.

- *Lớp Digital Signature:* dùng để xác định một chủ thể nhất định của tài liệu (ví dụ như tác giả hay nhan đề của một cuốn sách).

- *Các lớp Logic, Proof, Trust:* lớp logic cho phép quy định các luật (rule) thi hành các luật và cùng với lớp thử nghiệm (proof) và lớp chứng nhận (trust) đánh giá nhằm quyết định nên hay không chấp nhận những vấn đề đã được thử nghiệm.

1.2 RDF, RDFS

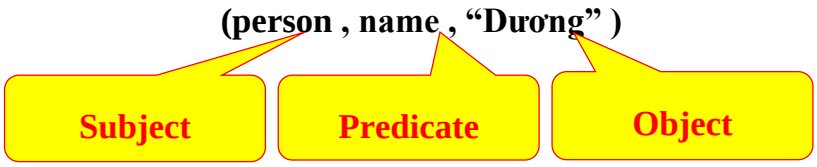
1.2.1 Giới thiệu RDF

RDF (Resource Description Framework) hay khung mô tả tài nguyên, là nền tảng cho việc biểu diễn các dữ liệu trong lĩnh vực Web có ngữ nghĩa. RDF là một tập hợp nhiều nguyên tắc dành cho ngôn ngữ đánh dấu, cung cấp mô hình dữ liệu và các

cú pháp đơn giản với các hệ thống độc lập. Đồng thời, nó được thiết kế để có thể đọc được thông tin và hệ thống máy tính có thể hiểu được, chứ không đơn giản là chỉ để trình bày dữ liệu cho người dùng cuối. Biểu diễn thông tin theo mô hình RDF gồm một phát biểu (statement) ở dạng cấu trúc bộ ba (triple) với thành phần chính là: (Subject, Predicate, Object). Trong đó:

- **Subject** miêu tả đối tượng đóng vai trò là chủ thể.
- **Predicate** (property) là thuộc tính hoặc quan hệ của chủ thể.
- **Object** là giá trị đối tượng hay thuộc tính của chủ thể đã nêu. Object có thể là một giá trị nguyên thủy như số chuỗi kí tự, số nguyên,.. hoặc cũng có thể là một tài nguyên nhất định.

Ví dụ sau minh họa cho một triple:



Hình 1.2. Mô hình bộ ba RDF (RDF triple)

Phát biểu trên mô tả một chủ thể (subject) **person** có thuộc tính (property) **name** với giá trị là Dương (object) . Phát biểu có thể được tạm hiểu là: **person** có tên là **Dương**.

1.2.1.1 Đồ thị RDF

Xét phát biểu:

http://www.ptit.edu.vn/index.html *has a* ***student*** *whose value is* ***Duong.***

Thành phần của phát biểu trên gồm:

- Chủ ngữ là: URL *http://www.ptit.edu.vn/index.html.*
- Vị từ là: *student.*

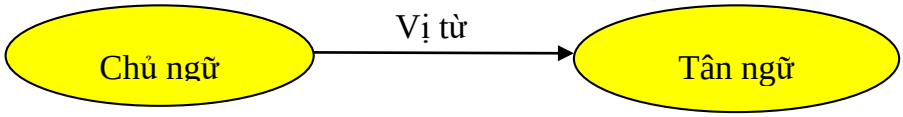
- Tân ngữ là: *Duong*.

Bởi tính tổng quát, RDF sử dụng những URI khác nhau để xác định những chủ ngữ, vị từ và tân ngữ trong một phát biểu. RDF mô tả những phát biểu gồm có một chủ ngữ (subject), vị từ (predicate) và tân ngữ (object):

- Chủ ngữ: *http://www.example.com/index.html.*
- Vị từ: *http://purl.org/dc/elements/1.9/student.*
- Tân ngữ: *http://www.example.com/staffid/9999.*

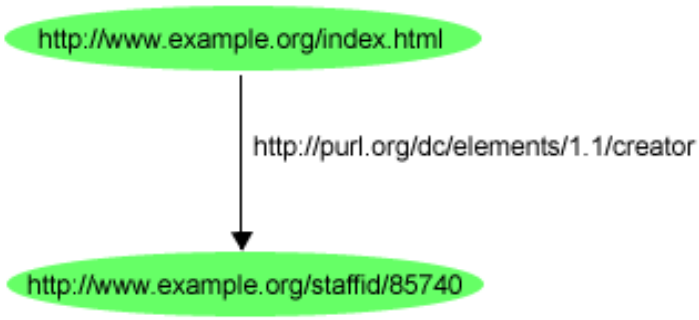
URI thường được sử dụng để xác định không chỉ chủ ngữ của phát biểu trên, mà còn xác định vị từ và tân ngữ, thay vì sử dụng từ "*student*" và "*Duong*", theo một cách riêng biệt.

Đồ thị RDF là tập hợp của các bộ ba hợp lại tạo thành. Các chủ ngữ các là các nút trong đồ thị, trong cung đồ thị là vị từ và tân ngữ trong bộ ba. Bộ ba được mô tả dạng nút - cung - nút. Hướng của các đồ thị rất quan trọng và luôn bắt đầu từ chủ ngữ qua vị từ đến tân ngữ.



Hình 1.3. Mối quan hệ giữa các thành phần trong bộ ba RDF

Vì thế phát biểu RDF ở trên được biểu diễn dưới dạng đồ thị như hình sau:



Hình 1.4. Đồ thị RDF

1.2.1.2 Cú pháp RDF/XML

Để biểu diễn những phát biểu RDF trong một cách thức để máy có khả năng xử lý được, RDF sử dụng rất nhiều ngôn ngữ trong đó có ngôn ngữ XML. XML (Extensible Markup Language) là ngôn ngữ đánh dấu mở rộng. RDF định nghĩa ra một loại ngôn ngữ đánh dấu XML đặc biệt, được biết giống như RDF/XML, để sử dụng trong việc biểu diễn các thông tin của bộ ba trong RDF. Như phần trước đã giới thiệu một đồ thị được xem là một tập những đường đi của các nút, cung vị từ, nút, cung vị từ, nút, cung vị từ,… chúng bao bọc toàn đồ thị. Trong RDF/XML điều này trở thành sự nối tiếp liên tục của các phần tử bên trong các phần tử mà xen kẽ giữa các phần tử đối với những nút và cung vị từ. Điều này được gọi là một loạt những nút/cung. Nút ở vị trí bắt đầu của chuỗi nối tiếp trở thành phần tử ở ngoài cùng, cung vị từ tiếp theo trở thành một phần tử con.

Cú pháp cơ bản RDF:

```
    [1] RDF         ::=['<rdf:RDF>'] description* ['</rdf:RDF>']
    [2] description ::='<rdf:Description' idAboutAttr?'>' propertyEIl*
'</rdf:Description>'
    [3] idAboutAttr ::= idAttr | aboutAttr
    [4] aboutAttr   ::= 'about'="tham chiếu URI"
    [5] idAttr                ::= 'ID' = "IDsymbol"
    [6] propertyEIT ::= '<'propName'>' value '</'propName'>'|'<' propName
resourceAttr '/>'
    [7] propName    ::= QName
    [8] value       ::= description | string
    [9] resource    ::= resource = "tham chiếu URI"
    [10] QName      ::= [ Nsprefix ':'] name
    [11] URI        ::= string, interpreted per
    [12] IDdsymbol  ::= (bất kỳ ID nào hợp lệ của XML)
    [13] name       ::= (bất kỳ tên hợp lệ nào của XML)
    [14] Nsprefix   ::= (bất kỳ tiếp đầu ngữ namspace hợp lệ nào)
    [15] string     ::= (bất kỳ chuỗi nào)
```

Ví dụ: Xét phát biểu sau:

```
    ex:index.html           exterms:DOB "March 22, 1996".
```

Cú pháp RDF/XML:

```
    1. <?xml version="1.0">
    2. <rdf:RDF    xmlns:rdf=http://www.w3c.org/1999/02/22-rdf-syntax-ns
    3.             xmlns:exterms="http://www.example.com/terms/">
    4. <rdf:Description rdf:about=" http://www.example.com/index.html">
    5.      <exterms:DOB>March 22, 1996</exterm:DOB>
```

```
6. </rdf:Description>
7. </rdf:RDF>
```

Giải thích cụ thể của phát biểu trên như sau:

- Dòng 1: khai báo XML, cho ta biết phiên bản XML được sử dụng và nội dung theo sau là dựa trên cú pháp XML.
- Dòng 2 và 3: với thẻ *rdf:RDF*, cho ta biết rằng nội dung XML tiếp theo (bắt đầu từ đây cho đến *</rdf:RDF>* trong dòng 7) mô tả của RDF. Từ khóa trên xác định tài liệu được biểu diễn dưới dạng RDF. Tiếp theo là phần khai báo namespace của XML được sử dụng trong các tài liệu, tùy vào nhu cầu cũng như mục đích sử dụng mà ta có thể định nghĩa các namespace khác nhau cho từng tài liệu.
- Dòng 4, 5, 6: là các phát biểu RDF. Mô tả phát biểu RDF/XML với *rdf:Description*, và *rdf:about,* đây chính là chủ ngữ của phát biểu (*http://www.example.com/index.html*). Thẻ bắt đầu *rdf:Description* trong dòng 4 cho biết bắt đầu mô tả về một tài nguyên, và tiếp tục định danh các tài nguyên dùng thuộc tính *rdf:about* để chỉ ra URI của chủ ngữ tài nguyên. Dòng 5 với QName *exterms:DOB* là thẻ của nó. Nội dung của thuộc tính này là tân ngữ của phát biểu, có giá trị là "March 22, 1996"
- Dòng 7: cho biết đánh dấu kết thúc của thẻ *rdf:RDF* và là thẻ để kết thúc tài liệu RDF.

1.2.2 Giới thiệu về RDFS

RDF sử dụng XML để có thể trao đổi những mô tả tài nguyên web nhưng những tài nguyên mô tả có thể thuộc nhiều loại khác nhau, bao gồm những tài nguyên XML và không XML. RDF dựa vào những tiện ích để cho phép các tài nguyên web được xử lý tự động. RDF được dùng trong nhiều ứng dụng khác nhau, ví dụ: trong khám phá tài nguyên để giúp cho việc tìm kiếm cho kết quả tốt hơn, hay trong việc Các lớp và các thuộc tính RDF chia thành loại để mô tả nội dung và những mối quan hệ sẵn có tại một trang web đặc biệt bằng các phần mềm thông minh để làm cho việc chia sẻ và trao đổi tri thức được dễ dàng hơn. Trong khi RDF là ngôn ngữ không thiết lập các giả định về bất kỳ lĩnh vực ứng dụng cụ thể nào, cũng không định nghĩa ngữ

nghĩa của lĩnh vực chuyên biệt nào thì RDFS cung cấp phương pháp xây dựng một mô hình đối tượng từ dữ liệu thực tế và cho chúng ta ngữ nghĩa của dữ liệu.

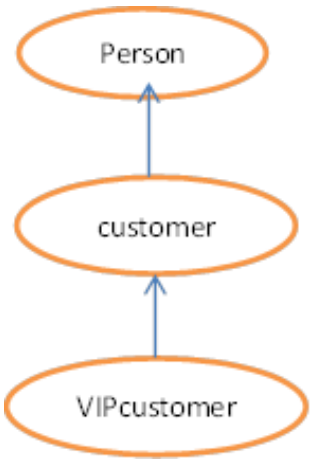
RDFS gồm các thành phần cơ bản sau:

Lớp và thuộc tính:

Lớp (class) là một tập các phần tử cùng chung một số tính chất. Các đối tượng đơn lẻ thuộc về một lớp thường được gọi là các thể hiện (instance) của lớp đó.

Cây phân cấp lớp và sự kế thừa:

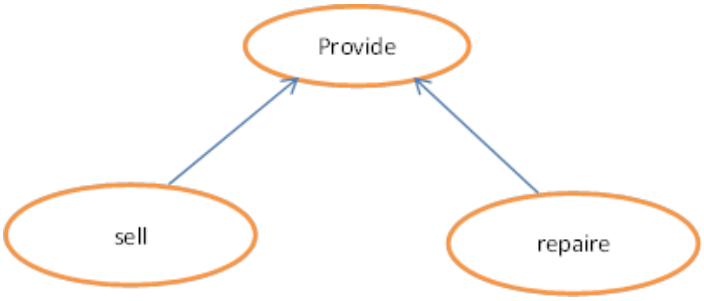
Sau khi đã có các lớp, dựa vào đó chúng ta có thể thiết lập các quan hệ giữa chúng. Ví dụ: Person, customer, VIP-customer.



Hình 1.5. Cấu trúc phân cấp của các lớp

Cây phân cấp thuộc tính:

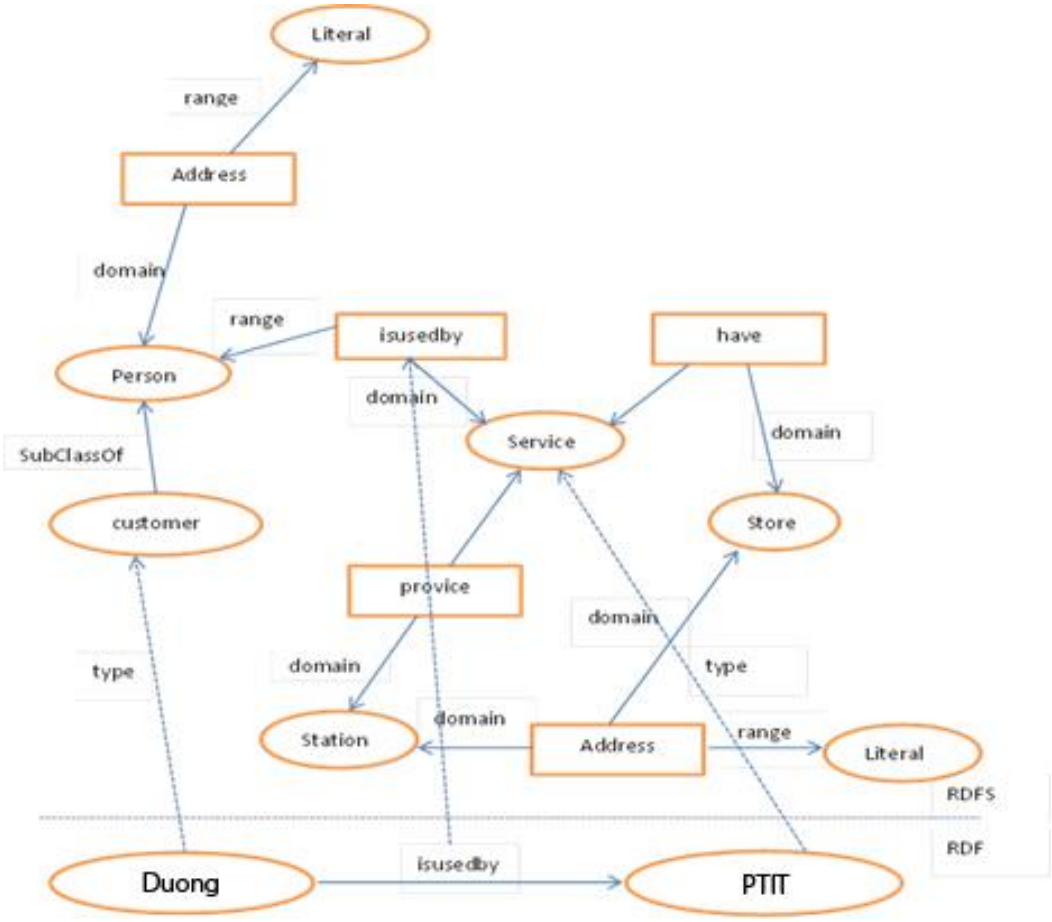
Chúng ta đã có cây phân cấp lớp và tương tự như vậy chúng ta cũng có cây phân cấp thuộc tính. Ví dụ:



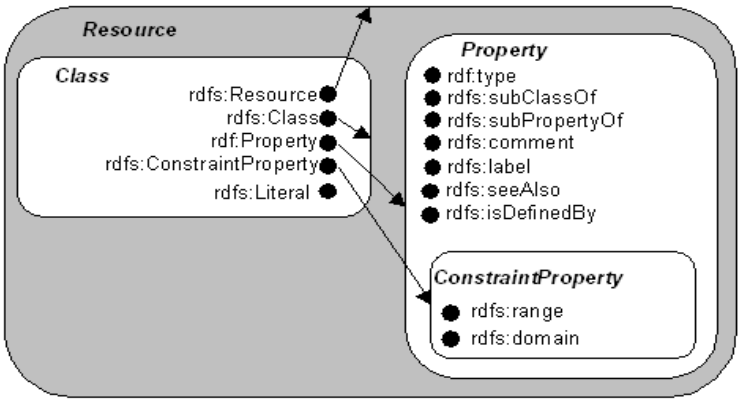
Hình 1.6. Cấu trúc phân cấp các thuộc tính

Các tầng của RDF và RDFS:

Chúng ta đã có cây phân cấp lớp và tương tự như vậy chúng ta cũng có cây phân cấp thuộc tính. Ví dụ:



Hình 1.7. Các tầng RDF và RDFS



Hình 1.8. Các lớp và các thuộc tính của RDF

Hình 1.8 minh họa khái niệm của lớp, lớp con và tài nguyên. Một lớp được mô tả bởi một hình chữ nhật tròn; một tài nguyên là một dấu chấm tròn lớn. Những mũi tên được vẽ từ một tài nguyên tới lớp nó định nghĩa. Một lớp con là một hình chữ nhật tròn nằm trong lớp khác.

1.3 Các ngôn ngữ truy vấn của Web ngữ nghĩa

1.3.1 Simple Protocol And RDF Query Language (SPARQL)

Như chúng ta đã biết RDF là một cách để mô tả thông tin về các tài nguyên Web một cách linh động. Nó được sử dụng để mô tả nhiều loại thông tin ví dụ như thông tin về cá nhân, về hệ thống mạng toàn cầu,các siêu dữ liệu (meta-data) về đa phương tiện như âm thanh và hình ảnh cũng như là cung cấp một phương tiện để tích hợp các nguồn tài nguyên khác. Với một lượng thông tin quá lớn như vậy, để dữ liệu có thể được truy vấn nhanh chóng và chính xác, tổ chức W3C đã phát triển một ngôn ngữ chuẩn để truy vấn dữ liệu RDF với việc cung cấp nhiều chức năng cho các nhà phát triển web và người dùng cuối cách để viết và thực thi một câu truy vấn. Phần này mô tả ngôn ngữ truy vấn Protocol And RDF Query Language (SPARQL) để truy cập dữ liệu RDF được phát triển bởi nhóm RDF Data Access Working Group.

Một đồ thị RDF (RDF Graph) là một tập các bộ ba (triple), mỗi bộ ba bao gồm bộ ba chủ ngữ (subject), vị từ (predicate) và tân ngữ (object). Những bộ ba này có thể đến từ nhiều tài nguyên khác nhau. Ví dụ như: chúng có thể đến từ một tài liệu RDF,

chúng có thể được suy ra từ các RDF bộ ba (RDF triple) khác hoặc là chúng có thể là sự mô tả RDF của các dữ liệu được lưu trữ từ các định dạng khác như XML hay là từ một cơ sở dữ liệu quan hệ.

SPARQL là một ngôn ngữ để truy vấn các thông tin từ các đồ thị RDF. SPARQL có thể cung cấp các tính năng sau:

- Trích xuất các thông tin trong các dạng của URI
- Trích thông tin từ phía các đồ thị con.
- Đồ thị RDF mới được xây dựng trên thông tin trong đồ thị có sẵn.

Ngôn ngữ truy vấn SPARQL dựa trên việc kết hợp những mẫu đồ thị. Mẫu đồ thị đơn giản nhất là mẫu bộ ba, nó tương tự như bộ ba RDF nhưng với khả năng của một biến trong bất kỳ vị trí nào chủ ngữ, vị từ hay tân ngữ. Việc kết hợp này dẫn tới một mẫu đồ thị cơ bản, nơi mà một sự kết hợp chính xác tới một đồ thị thật sự cần thiết để hoàn thành một mẫu. Những phần sau của chương này mô tả những mẫu đồ thị khác có thể được xây dựng sử dụng những toán tử đồ thị như phép chọn (OPTIONAL) và phép hợp (UNION); những mẫu đồ thị có thể được nhóm lại với nhau; những truy vấn có thể trích thông tin từ nhiều một đồ thị. Trong phần này, tôi hướng về những mẫu bộ ba đơn giản, những mẫu đồ thị cơ bản cũng như cú pháp SPARQL cho những truy vấn mẫu cơ bản.

1.3.1.1 Viết một truy vấn đơn giản

Một lệnh truy vấn bao gồm hai phần, mệnh đề SELECT cùng mệnh đề WHERE. Mệnh đề SELECT xác định biến nào xuất hiện trong những kết quả truy vấn, còn mệnh đề WHERE bao gồm các một mẫu bộ ba (tripple pattern). Một mẫu bộ ba là một bộ ba RDF nhưng mỗi thành phần (chủ ngữ, vị từ, tân ngữ) đều có thể là một biến truy vấn.

Mẫu đồ thị cơ bản là một tập các mẫu bộ ba. Ngôn ngữ SPARQL dựa trên nền tảng so sánh các mẫu đồ thị. Một lệnh truy vấn có thể có hoặc không, một hoặc nhiều

kết quả. Ví dụ bên dưới trình bày một truy vấn SPARQL tìm kiếm tựa đề của một cuốn sách từ thông tin trong đồ thị đã cho.

- Dữ liệu:

```
<http://example.org/book/book>
<http://purl.org/dc/elements/1.2/title> "SPARQL"
```

- Truy vấn**:**

```
SELECT ?title
WHERE {<http://example.org/book/book>
<http://purl.org/dc/elements/1.2/title> ?title}
```

- Kết quả truy vấn:

Title
"SPARQL"

1.3.1.2 Cú pháp các biến.

Những biến trong truy vấn SPARQL có phạm vi toàn cục; các biến giống nhau ở khắp mọi nơi trong truy vấn nếu nó có tên giống nhau. Những biến được diễn tả bằng "?" hay "$".

Ví dụ: những biểu thức bên dưới mô tả cùng một truy vấn:

```
SELECT ?x ?y
FROM <http://example.com/sample.rdf>
WHERE { ?x, <dc:name>,  ?y}
USING dc for <http://www.dc.com#>
```

Kết quả trả về các bộ ?x ?y cho biết tên tài nguyên và các giá trị của thuộc tính dc:name cho mỗi tài nguyên.

1.4 LINKED DATA

1.4.1 Khái niệm

Trong các hoạt động tính toán của máy tính, Linked data được mô tả như phương thức tạo ra dữ liệu có cấu trúc để có thể liên kết được với nhau và trở nên có

ích. Dữ liệu Liên kết (Linked Data) là một trong những khái niệm và trụ cột chính của Web ngữ nghĩa (Semantic Web), còn được biết tới như là Web của Dữ liệu. Web ngữ nghĩa tất cả là về các liên kết giữa các tập hợp dữ liệu có thể hiểu được không chỉ cho con người mà còn cho máy tính, và Dữ liệu Liên kết đưa ra các thực hành tốt nhất để tạo ra các liên kết đó. Dữ liệu Liên kết là tập hợp các nguyên tắc thiết kế cho việc chia sẻ dữ liệu được liên kết với nhau trên Web mà máy tính đọc được.

1.4.2 Các quy tắc triển khai Linked Data

Tim Berners-Lee liệt kê bốn quy tắc triển khai Linked data trong các thảo luận của mình được tóm gọn trong các dòng sau:

1. Dùng URI có thể định danh các tài nguyên.

2. Sử dụng HTTP URI để các tài nguyên được tham chiếu và tìm kiếm chúng ("tham chiếu lại") bởi mọi người hay các ứng dụng.

3. Cung cấp các thông tin hữu ích về tài nguyên khi các URI của nó được tham chiếu lại, sử dụng định dạng tiêu chuẩn như RDF/XML.

4. Gồm các liên kết đến liên kết khác, các URL có thể có quan hệ bên trong dữ liệu tìm thấy để đẩy mạnh khả năng tìm kiếm các thông tin liên quan khác trên Web.

1.4.3 Truy vấn Web dữ liệu

Những đồ thị RDF được truy xuất thông qua URI trên Web dữ liệu là một phần của một tập dữ liệu liên kết. Tập dữ liệu liên kết này là một đồ thị RDF chứa dữ liệu về nhiều thực thể hay đối tượng. Để tạo ra tập dữ liệu liên kết này, chúng ta có thể dùng một số cách phổ biến như: truy vấn dữ liệu dựa vào giao diện Linked Data, trên các kho lưu trữ trực tuyến hay trên các giao diện Web. Để xuất bản tập dữ liệu liên kết theo công thức Linked Data, người ta đưa ra khái niệm kết xuất RDF hay một điểm đầu cuối SPARQL để truy vấn vào tập dữ liệu liên kết này. Một kết xuất RDF là một tài liệu RDF lớn chứa đồ thị RDF cấu thành tập dữ liệu liên kết. Một điểm đầu cuối SPARQL là một dịch vụ truy vấn dựa trên giao thức HTTP, xử lý các truy vấn

SPARQL trên tập dữ liệu liên kết. SPARQL – ngôn ngữ truy vấn cho đồ thị RDF, dựa trên các mẫu đồ thị RDF và đối sánh đồ thị con.

Các tiếp cận về xử lý truy vấn có thể kể đến như xếp kho dữ liệu (Data Warehousing), các máy tìm kiếm (Search Engines), truy vấn liên hiệp (Query Federation), truy vấn liên hiệp khám phá (Discovery Query Federation) hay quét liên kết (Linh traversal), dựa trên một số thuộc tính như không gian dữ liệu vào, giao diện yêu cầu, truy xuất đến dữ liệu nguồn, hỗ trợ cấu trúc dữ liệu, tính đáp ứng, mức độ truy vấn dữ liệu, độ chính xác, tính cập nhật.

1.5 Tiểu kết chương 1

Chương này đã trình bày những khái niệm căn bản và quan trọng, là nền tảng lý thuyết và cốt lõi cho các hướng phát triển cao hơn sau này. Phần đầu chương đã giới thiệu Web ngữ nghĩa là gì, tại sao cần phải nghiên cứu Web ngữ nghĩa. Phần thứ hai là tìm hiểu về RDF, RDFS. Tiếp theo là về ngôn ngữ truy vấn RDF SPARQL và một số ngôn ngữ truy vấn Web ngữ nghĩa khác. Phần cuối nói về Linked Data – ý tưởng đang dần được hoàn thiện trong thực tế: một đám mây dữ liệu toàn cầu, duy nhất, đầy đủ, được công bố rộng rãi và rất quan trọng, mọi thứ được liên kết với nhau.

CHƯƠNG 2: LẬP TRÌNH VỚI WEB NGỮ NGHĨA

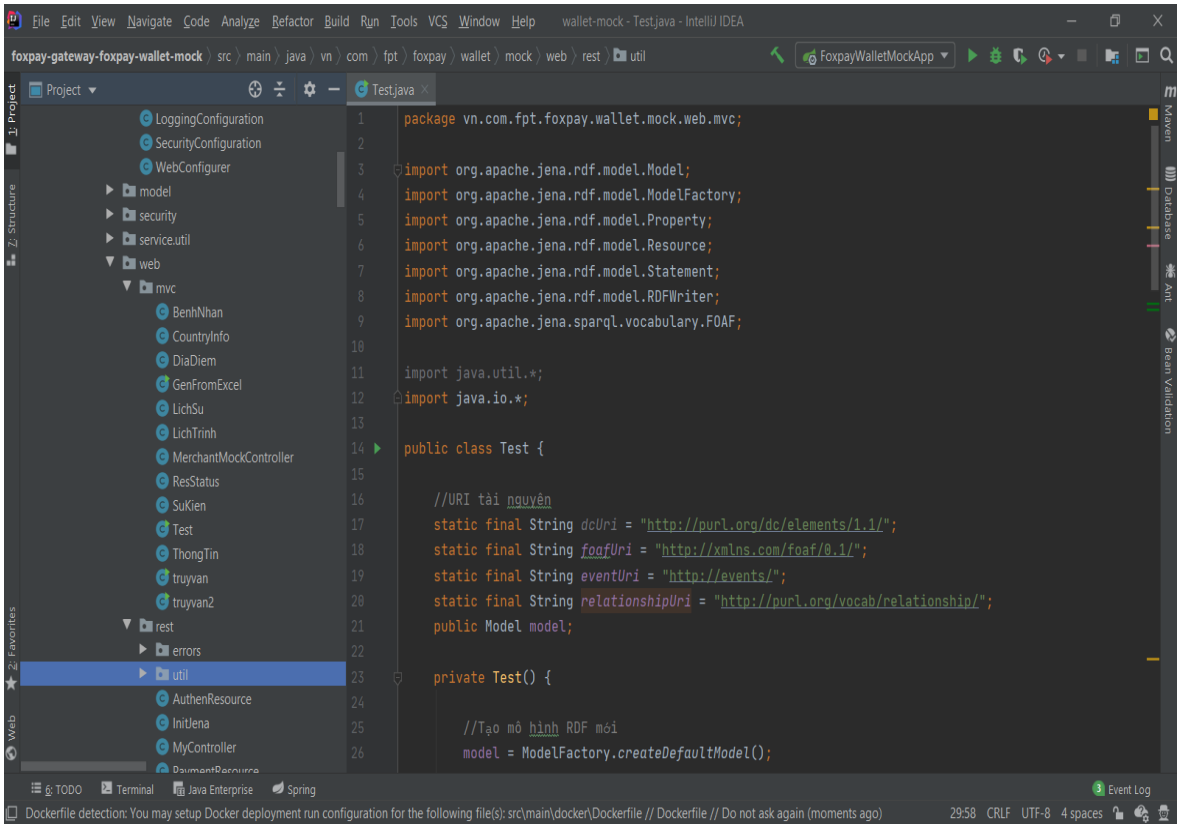
Trình bày một số phương pháp lập trình với Web ngữ nghĩa, các môi trường lập trình cũng như kiến trúc của chúng. Ngoài ra chương này còn giới thiệu Jena Framework dùng cho việc xây dựng ứng dụng Web ngữ nghĩa.

2.1 Môi trường lập trình Web ngữ nghĩa

2.1.1 Môi trường lập trình Intellij

2.1.1.1 *Giới thiệu.*

IntelliJ IDEA là một trình soạn thảo (IDE) để phát triển các phần mềm máy tính. Nó được phát triển bởi nhà sản xuất JetBrains (trước đây gọi là IntelliJ), và được cấp phép Apache. Intellij có 2 phiên bản: Intellij Ultimate (thương mại) và Intellij Community (cộng đồng). Cả hai có thể được sử dụng cho phát triển phần mềm.



Hình 2.1. Giao diện của Intellij

2.1.1.2 *Các thành phần cơ bản*

- *Kiểm soát phiên bản*

IntelliJ cung cấp một giao diện thống nhất cho các hệ thống kiểm soát phiên bản chính bao gồm Git, SVN, Mercurial, CVS, Perforce và TFS. IDE cho phép duyệt qua lịch sử thay đổi, quản lý chi nhánh, hợp nhất xung đột và hơn thế nữa.

- *Xây dựng công cụ*

IntelliJ IDEA hỗ trợ Maven, Gradle, Ant, Gant, SBT, NPM, Webpack, Grunt, Gulp và các công cụ xây dựng khác. Được tích hợp liền mạch, các công cụ này giúp tự động hóa quá trình biên dịch, đóng gói, chạy thử nghiệm, triển khai và các hoạt động khác.

- *Kiểm thử*

IntelliJ IDEA cho phép bạn thực hiện kiểm tra đơn vị một cách dễ dàng. IDE bao gồm các trình chạy thử nghiệm và các công cụ bảo hiểm cho các khuôn khổ thử nghiệm chính, bao gồm JUnit, TestNG, Spock; Cucumber, ScalaTest, spec2 và Karma.

- *Trình biên dịch*

IntelliJ IDEA đi kèm với một trình dịch ngược tích hợp cho các lớp Java. Khi bạn muốn xem bên trong một thư viện mà bạn không có mã nguồn, bây giờ bạn có thể – mà không cần bất kỳ plugin nào của bên thứ ba.

- *Thiết bị đầu cuối*

IDE đi kèm với một thiết bị đầu cuối tích hợp. Tùy thuộc vào nền tảng của bạn, bạn có thể làm việc với dấu nhắc dòng lệnh, Far, powershell hoặc bash. Gọi thiết bị đầu cuối bằng Alt + F12 và thực hiện bất kỳ lệnh nào – mà không cần rời khỏi IDE.

- *Công cụ cơ sở dữ liệu*

Tận dụng sự hỗ trợ mã hóa thông minh khi chỉnh sửa SQL; kết nối với cơ sở dữ liệu trực tiếp; chạy truy vấn; duyệt và xuất dữ liệu; và thậm chí quản lý các lược đồ của bạn trong một giao diện trực quan – ngay từ IDE.

- *Máy chủ ứng dụng*

IntelliJ IDEA hỗ trợ các máy chủ ứng dụng chính: Tomcat, JBoss, WebSphere, WebLogic, Glassfish và nhiều máy chủ khác.Bạn có thể triển khai các tạo tác của mình trên các máy chủ ứng dụng và gỡ lỗi các ứng dụng đã triển khai ngay từ bên trong IDE.

2.1.2 Jena

2.1.2.1 Giới thiệu

Jena là một thư viện của Java framework dùng cho việc xây dựng các ứng dụng Web ngữ nghĩa. Cung cấp một môi trường lập trình cho RDF, RDFS and OWL, SPARQL. Bao gồm các công cụ suy diễn từ các luật cơ sở (rule-based inference engine). Phiên bản đầu tiên ra đời do HP Lab xây dựng Jena 2 ra đời vào 2003 – phiên bản hiện tại là Jena 3.0.

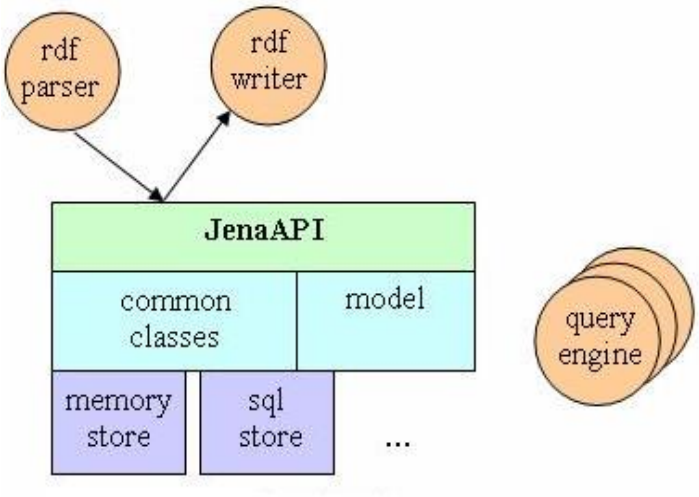
Jena Framework bao gồm:

- RDF API: Giao diện lập trình cho RDF
- OWL API: Giao diện lập trình cho OWL
- Xuất và đọc các file RDF theo dạng RDF/XML, N3 và N-Triples
- Cho phép lưu trữ trong bộ nhớ, lưu trữ cố định trong các tập tin đơn hay trong các hệ quản trị cơ sở dữ liệu (MySQL, HSQLDB, PostgreSQL, Oracle, Microsoft SQL Server)
- Hệ truy vấn SPARQL

2.1.2.2 Jena API

Kiến trúc của Jena được thiết kế để cho phép tích hợp một cách dễ dàng các thành phần xử lý có thể thay thế như bộ phân tích dữ liệu (parser), xuất bản (writer), lưu trữ và truy vấn.

Jena API bao gồm một tập các giao diện Java mô tả các nguồn tài nguyên (resources), các thuộc tính (properties), các phát biểu (statements) và các mô hình (model) dưới dạng khung mô tả tài nguyên RDF (Resource Description Framework).



Hình 2.2. Các thành phần của Jena API

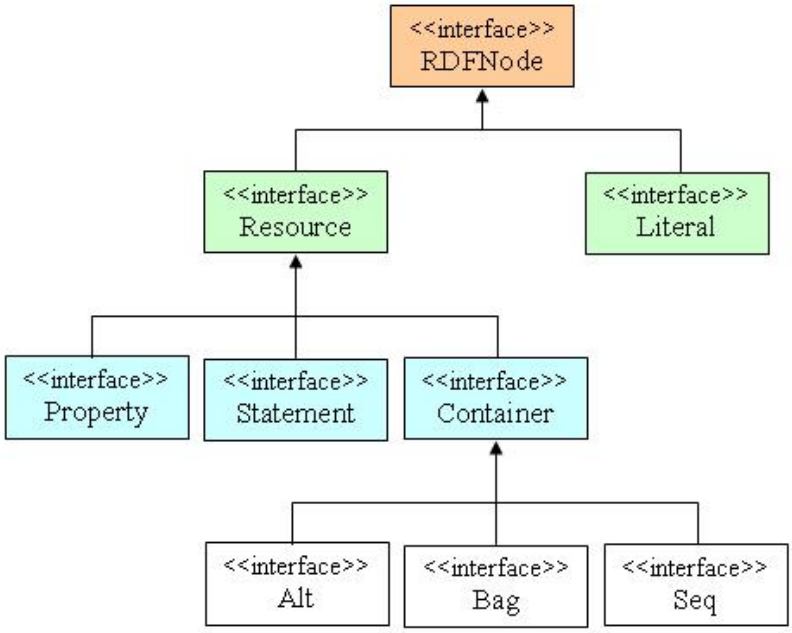
2.1.2.3 Kiến trúc Jena 3

RDFNode interface: Cung cấp các thành phần của các bộ ba RDF {predicate, subject, object}.

Literal interface: Dùng để mô tả các chuỗi và chuyển đổi một số kiểu của Java như String, Int, và Double. Các đối tượng hiện thực giao diện Property có thể là một predicate trong bộ ba {predicate, subject, object}.

Statement interface: mô tả bộ ba {predicate, subject, object}. Đồng thời nó cũng có thể được dùng như một đối tượng .

Các đối tượng hiện thực các giao diện Container, Alt, Bag, hoặc Seq có thể là đối tượng.



Hình 2.3. Kiến trúc Jena 3

2.2 Mô hình hóa thông tin với Jena

2.2.1 Chương trình Hello Word! trong Jena

Chương trình sau mô tả cách tạo một mô hình RDF trong Jena, thêm tài nguyên, thuộc tính và xuất ra dưới định dạng Turtle.

```
 public class Test {
  public static void main(String[] args) {

//Khoi tao mo hinh RDF
        Model m=ModelFactory.createDefaultModel();
        String ns="http://vidu.com/duong/";

//Them cac tai nguyen Resource
        Resource r= m.createResource(ns+"r");

//Them cac thuoc tinh cho tai nguyen
        Property p=m.createProperty(ns+"p");
        r.addProperty(p, "Hello Word!",XSDDatatype.XSDstring);
        m.write(System.out,"Turtle");
   }
}
```

Kết quả xuất ra dưới định dạng Turtle:

```
<http://vidu.com/duong/r>
        <http://vidu.com/duong/p>  "Hello Word!" .
```

Để các hàm và thư viện trong chương trình trên sử dụng được chúng ta phải download thư viện Jena . Sau đó để tạo thêm thư viện vào ta vào file cấu hình và thêm dependencies của Jena.

```
<dependency>
    <groupId>org.apache.jena</groupId>
    <artifactId>jena-core</artifactId>
    <version>3.12.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.jena/apache-jena-libs -->
<dependency>
    <groupId>org.apache.jena</groupId>
    <artifactId>apache-jena-libs</artifactId>
    <version>3.12.0</version>
    <type>pom</type>
</dependency>
```

Hình 2.4. Thêm thư viện Jena

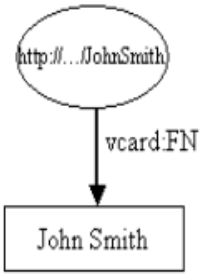
2.2.2 Tạo mô hình RDF

2.2.2.1 Tạo tài nguyên và thêm thuộc tính

Trong phần này, tôi tạo ra một mô hình RDF trống từ đầu và sau đó thêm thuộc tính cho nó.

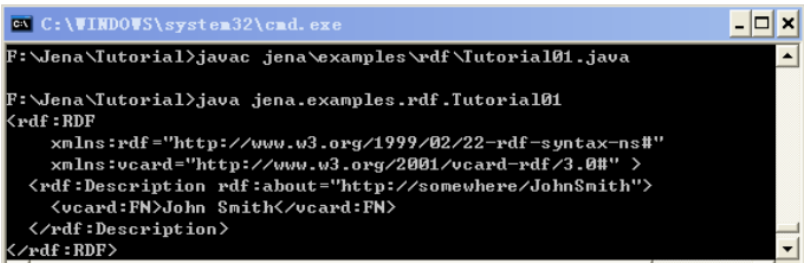
```
public class Test01 extends Object {

      //URI tai nguyen
            static String personURI     = "http://somewhere/JohnSmith";
            static String fullName      = "John Smith";
            public static void main (String args[]) {

      //Tao mo hinh RDF
            Model model = ModelFactory.createDefaultModel();

      //Tao tai nguyen
      Resource johnSmith = (Resource) model.createResource(personURI);

      //Them thuoc tinh
      ((org.apache.jena.rdf.model.Resource)
johnSmith).addProperty(VCARD.FN, fullName);
      //Xuat dang XML
      model.write(System.out);
      }}
```

Biểu diễn dưới dạng đồ thị:



Hình 2.5. Biểu diễn phát biểu dưới dạng đồ thị

Kết quả của chương trình:



```
C:\WINDOWS\system32\cmd.exe
F:\Jena\Tutorial>javac jena\examples\rdf\Tutorial01.java

F:\Jena\Tutorial>java jena.examples.rdf.Tutorial01
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#" >
  <rdf:Description rdf:about="http://somewhere/JohnSmith">
    <vcard:FN>John Smith</vcard:FN>
  </rdf:Description>
</rdf:RDF>
```

Hình 2.6. Kết quả trên giao diện console

2.2.2.2 Thêm mối quan hệ, xuất ra tập tin RDF

Trong ví dụ dưới đây tôi xây dựng một mô hình RDF về gia 1 đình và mối quan hệ giữa các thành viên trong gia đình. Sau đó, xuất thông tin lưu trữ file định dạng RDF.

```
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.rdf.model.Property;
import org.apache.jena.rdf.model.Resource;
import org.apache.jena.rdf.model.Statement;
import org.apache.jena.rdf.model.RDFWriter;

import java.util.*;
import java.io.*;
public class FamilyModel {

//URL tai nguyen
static final String familyUri = "http://family/";
static final String relationshipUri =
"http://purl.org/vocab/relationship/";
public Model model;
private FamilyModel() {

//Tao mo hinh RDF
```

```
model = ModelFactory.createDefaultModel();

//Them cac thuoc tinh xac dinh moi quan he giua cac thanh vien
Property concua = model.createProperty(relationshipUri,"concua");
Property chamecua = model.createProperty(relationshipUri,"chamecua");
Property anhemcua = model.createProperty(relationshipUri,"anhemcua");
Property vochongcua = model.createProperty(relationshipUri,"vochongcua");

//Them cac tai nguyen thanh vien trong gia dinh
Resource hai = model.createResource(familyUri+"hai");
Resource mai = model.createResource(familyUri+"mai");
Resource trung = model.createResource(familyUri+"trung");
Resource hoa = model.createResource(familyUri+"hoa");
Resource Nam = model.createResource(familyUri+"Nam");
Resource Tung = model.createResource(familyUri+"Tung");
Resource Son = model.createResource(familyUri+"Son");
Resource Tuyet = model.createResource(familyUri+"Tuyet");

//Them cac moi quan he
  hai.addProperty(anhemcua,mai);
  hai.addProperty(vochongcua,hoa);
  hai.addProperty(chamecua,Nam);
  hai.addProperty(chamecua,Tung);
  mai.addProperty(anhemcua,hai);
  mai.addProperty(vochongcua,trung);
  trung.addProperty(vochongcua,mai);
  hoa.addProperty(vochongcua,hai);
  hoa.addProperty(chamecua,Nam);
  hoa.addProperty(chamecua,Tung);

//Them cac phat bieu
Statement statement1 = model.createStatement(Nam,concua,hai);
Statement statement2 = model.createStatement(Nam,concua,hoa);
Statement statement3 = model.createStatement(Nam,anhemcua,Tung);

  model.add(statement1);
  model.add(statement2);
  model.add(statement3);

//Them cac phat bieu dang Array
Statement statements[] = new Statement[5];
    statements[0] = model.createStatement(Tung,concua,hai);
    statements[1] = model.createStatement(Tung,concua,hoa);
    statements[2] = model.createStatement(Tung,anhemcua,Nam);
    statements[3] = model.createStatement(Tung,vochongcua,Son);
    statements[4] = model.createStatement(Tung,chamecua,Tuyet);
    model.add(statements);

//Them cac phat bieu dang List
List<Statement> list = new ArrayList<Statement>();
    list.add(model.createStatement(Son,vochongcua,Tung));
    list.add(model.createStatement(Son,chamecua,Tuyet));
    list.add(model.createStatement(Tuyet,concua,Tung));
    list.add(model.createStatement(Tuyet,concua,Son));
model.add(list);
}

      //Main
```

```
public static void main(String args[]) throws FileNotFoundException {

      FamilyModel theFamily = new FamilyModel();

      File file=new File("C:\\family.rdf");

      FileOutputStream f1=new FileOutputStream(file);

      RDFWriter d = theFamily.model.getWriter("RDF/XML-ABBREV");

      d.write(theFamily.model,f1,null);
}
}
```

Kết quả xuất ra tập tin *family.rdf* như sau:

```
      <rdf:RDF
          xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
          xmlns:j.0="http://purl.org/vocab/relationship/">
        <rdf:Description rdf:about="http://family/mai">
          <j.0:vochongcua>
            <rdf:Description rdf:about="http://family/trung">
              <j.0:vochongcua rdf:resource="http://family/mai"/>
            </rdf:Description>
          </j.0:vochongcua>
          <j.0:anhemcua>
            <rdf:Description rdf:about="http://family/hai">
              <j.0:chamecua>
                <rdf:Description rdf:about="http://family/Tung">
                  <j.0:chamecua>
                    <rdf:Description rdf:about="http://family/Tuyet">
                      <j.0:concua>
                        <rdf:Description rdf:about="http://family/Son">
                          <j.0:chamecua
rdf:resource="http://family/Tuyet"/>
                          <j.0:vochongcua
rdf:resource="http://family/Tung"/>
                        </rdf:Description>
                      </j.0:concua>
                      <j.0:concua rdf:resource="http://family/Tung"/>
                    </rdf:Description>
                  </j.0:chamecua>
                  <j.0:vochongcua rdf:resource="http://family/Son"/>
                  <j.0:anhemcua>
```

```
                    <rdf:Description rdf:about="http://family/Nam">
                      <j.0:anhemcua rdf:resource="http://family/Tung"/>
                      <j.0:concua>
                        <rdf:Description rdf:about="http://family/hoa">
                          <j.0:chamecua
rdf:resource="http://family/Tung"/>
                          <j.0:chamecua
rdf:resource="http://family/Nam"/>
                          <j.0:vochongcua
rdf:resource="http://family/hai"/>
                        </rdf:Description>
                      </j.0:concua>
                      <j.0:concua rdf:resource="http://family/hai"/>
                    </rdf:Description>
                  </j.0:anhemcua>
                  <j.0:concua rdf:resource="http://family/hoa"/>
                  <j.0:concua rdf:resource="http://family/hai"/>
                </rdf:Description>
              </j.0:chamecua>
              <j.0:chamecua rdf:resource="http://family/Nam"/>
              <j.0:vochongcua rdf:resource="http://family/hoa"/>
              <j.0:anhemcua rdf:resource="http://family/mai"/>
            </rdf:Description>
          </j.0:anhemcua>
  </rdf:Description>
  </rdf:RDF>
```

2.2.3 Đọc mô hình RDF

Phần này tôi sử dụng truy vấn SPARQL để xuất ra *?s ?p ?o* của tất cả các tài nguyên trong tập tin *family.rdf*.

```
      public class truyvan{
            public truyvan (){
            }
            public static void main(String[] args) {
      //duong dan den tap tin RDF
      final String inputFileName = "C:\\family.rdf";
                  Model model = ModelFactory.createDefaultModel();
```

```
                    InputStream in = FileManager.get().open(inputFileName);
                    model.read( in, "");
                  //chuoi truy van
      String queryString =
                                "PREFIX
quanhe:<http://purl.org/vocab/quanhe/>"+                    "SELECT * " +
                                " {" +
                                "  ?s ?p ?o. }";
            Query query = QueryFactory.create(queryString);
            QueryExecution   qe   =   QueryExecutionFactory.create(query,
model);
                    ResultSet results = qe.execSelect();
            // ResultSetFormatter
                    ResultSetFormatter.out(System.out, results, query);
                    qe.close();
            }
      }
```

Kết quả xuất ra dưới định dạng Turtle:

s	p	o
<http://family/Son>	<http://purl.org/vocab/relationship/vochongcua>	<http://family/Tung>
<http://family/Son>	<http://purl.org/vocab/relationship/chamecua>	<http://family/Tuyet>
<http://family/Tung>	<http://purl.org/vocab/relationship/concua>	<http://family/hai>
<http://family/Tung>	<http://purl.org/vocab/relationship/concua>	<http://family/hoa>
<http://family/Tung>	<http://purl.org/vocab/relationship/anhemcua>	<http://family/Nam>
<http://family/Tung>	<http://purl.org/vocab/relationship/vochongcua>	<http://family/Son>
<http://family/Tung>	<http://purl.org/vocab/relationship/chamecua>	<http://family/Tuyet>
<http://family/hoa>	<http://purl.org/vocab/relationship/vochongcua>	<http://family/hai>
<http://family/hoa>	<http://purl.org/vocab/relationship/chamecua>	<http://family/Nam>
<http://family/hoa>	<http://purl.org/vocab/relationship/chamecua>	<http://family/Tung>
<http://family/Tuyet>	<http://purl.org/vocab/relationship/concua>	<http://family/Tung>
<http://family/Tuyet>	<http://purl.org/vocab/relationship/concua>	<http://family/Son>
<http://family/trung>	<http://purl.org/vocab/relationship/vochongcua>	<http://family/mai>
<http://family/mai>	<http://purl.org/vocab/relationship/anhemcua>	<http://family/hai>
<http://family/mai>	<http://purl.org/vocab/relationship/vochongcua>	<http://family/trung>
<http://family/hai>	<http://purl.org/vocab/relationship/anhemcua>	<http://family/mai>
<http://family/hai>	<http://purl.org/vocab/relationship/vochongcua>	<http://family/hoa>
<http://family/hai>	<http://purl.org/vocab/relationship/chamecua>	<http://family/Nam>
<http://family/hai>	<http://purl.org/vocab/relationship/chamecua>	<http://family/Tung>
<http://family/Nam>	<http://purl.org/vocab/relationship/concua>	<http://family/hai>
<http://family/Nam>	<http://purl.org/vocab/relationship/concua>	<http://family/hoa>
<http://family/Nam>	<http://purl.org/vocab/relationship/anhemcua>	<http://family/Tung>

Hình 2.7. Dữ liệu xuất dưới dạng Turtle

2.3 Chuyển đổi dữ liệu Web ngữ nghĩa

Trong phần này tôi sử dụng thư viện *jxl.jar* để thao tác với tập tin Excel. Sau đó sử dụng Jena để tạo mô hình dữ liệu RDF. Thêm các tài nguyên và thuộc tính với dữ liệu đọc từ file Excel. Cuối cùng xuất dữ liệu ra tập tin RDF với định dạng XML.

Đoạn code sau mô tả quá trình trên:

```
import jxl.Sheet;
import jxl.Workbook;
……
      //Tạo mô hình RDF mới
      model = ModelFactory.createDefaultModel();
      //File Excel chứa dữ liệu bệnh nhân
      File file = new File("C:\\Data_BenhNhan.xls");
            Workbook wb = Workbook.getWorkbook(file2);
            Sheet sheet = wb.getSheet(0);
            int rows = sheet.getRows();
            int cols = sheet.getColumns();

      //Duyệt từng hàng để thêm các tài nguyên
      for(int row1 = 0; row1 < rows; row1++)
            {
                  Resource benhNhan[] = new Resource[100];
                  benhNhan[row1]=model.createResource(benhNhanUri
                  +sheet.getCell(0, row1).getContents());
            }

      //Khởi tạo một đối tượng Bệnh nhân mới
    BENHNHAN_RDF benhnhan = new BENHNHAN_RDF();
    File file2=new File("C:\\BENHNHAN.rdf");
    FileOutputStream f2=new FileOutputStream(file2);
      //Dùng đối tượng RDFWiter của Jena để ghi dữ liệu dưới dạng XML
    RDFWriter d2 = benhnhan.model.getWriter("RDF/XML-ABBREV");
   d2.write(benhnhan.model2,f2,null);
```

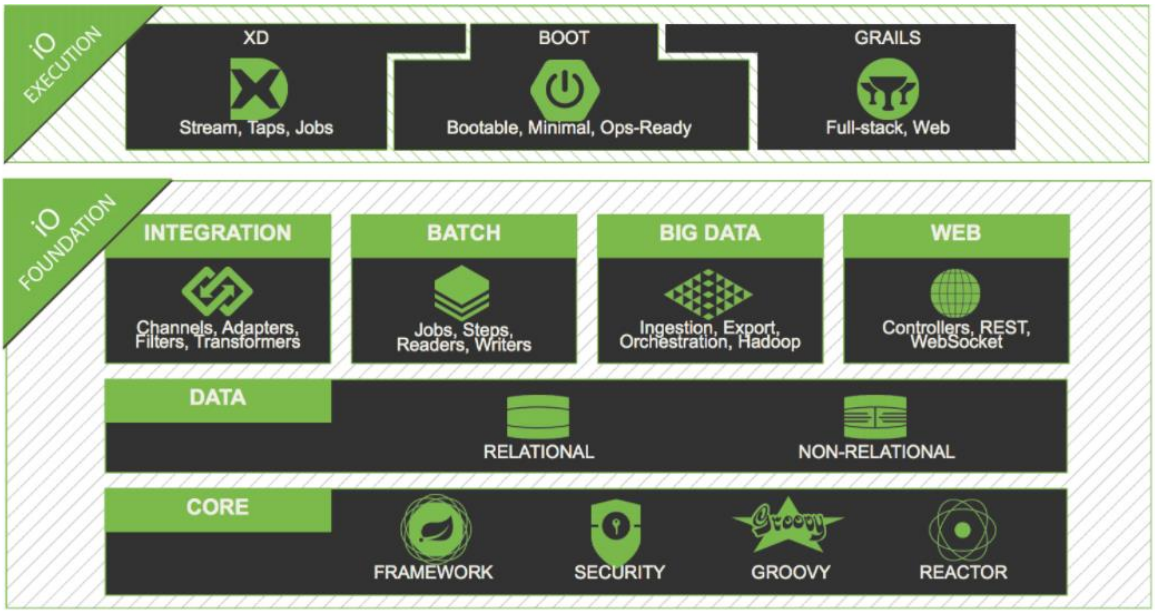
2.4 Framework Spring MVC

2.4.1 Giới thiệu về Framework Spring MVC

Spring là một trong những framework để phát triển các ứng dụng Java được sử dụng bởi hàng triệu lập trình viên trên thế giới. Spring giúp tạo ra các ứng dụng có hiệu năng cao, dễ dàng kiểm thử, sử dụng lại code một cách hợp lý. Spring gọn nhẹ và trong suốt (nhẹ: với kích thước nhỏ, version cơ bản chỉ khoảng 2 megabytes; trong suốt: hoạt động một cách trong suốt với đối lập trình viên), Spring là một mã

nguồn mở, được phát triển, chia sẻ và có cộng đồng người dùng rất lớn trên tehes giới. Spring Framework xây dựng dựa trên 2 nguyên tắc thiết kế chính là: Dependencies Injection và Aspect Orient Programming.

Những tính năng cốt lõi của Spring có thể được sử dụng để phát triển phần mềm máy tính ,các ứng dụng di động hay các Website bằng Java. Mục tiêu chính của Spring là giúp phát triển các ứng dụng Java dễ dàng hơn.



Hình 2.8. Các thành phần chính trong Spring

2.4.2 Ví dụ trang web Spring MVC cơ bản

```
package com.example.servingwebcontent;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class HelloController {

        @GetMapping("/hello")
        public String greeting(@RequestParam(name="name", required=false,
defaultValue="World") String name, Model model) {
                model.addAttribute("name", “World”);
                return "greeting";
        }
```

```
}

<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <p th:text="'Hello, ' + ${name} + '!'" />
</body>
</html>
```

Kết quả sẽ hiển thị dòng chữ *"Hello World!"* trên trình duyệt.

2.5 Tiểu kết chương 2

Chương 2 mô tả các các thành phần, công cụ để tạo nên ứng dụng Web ngữ nghĩa. Đầu tiên kiến trúc và cách cài đặt môi trường lập trình Java Intellij. Tiếp theo là Jena API – một thư viện Java dùng để chuyển đổi, lưu trữ dữ liệu Web ngữ nghĩa. Cách chuyển đổi dữ liệu từ tập tin Excel sang dữ liệu RDF. Chương này cũng trình bày về Spring MVC để lập trình Web ngữ nghĩa, tạo các giao diện hiển thị kết quả các truy vấn SPARQL trên tập tin RDF.

CHƯƠNG 3: ỨNG DỤNG TÌM KIẾM THÔNG TIN BỆNH NHÂN COVID-19 TẠI VIỆT NAM

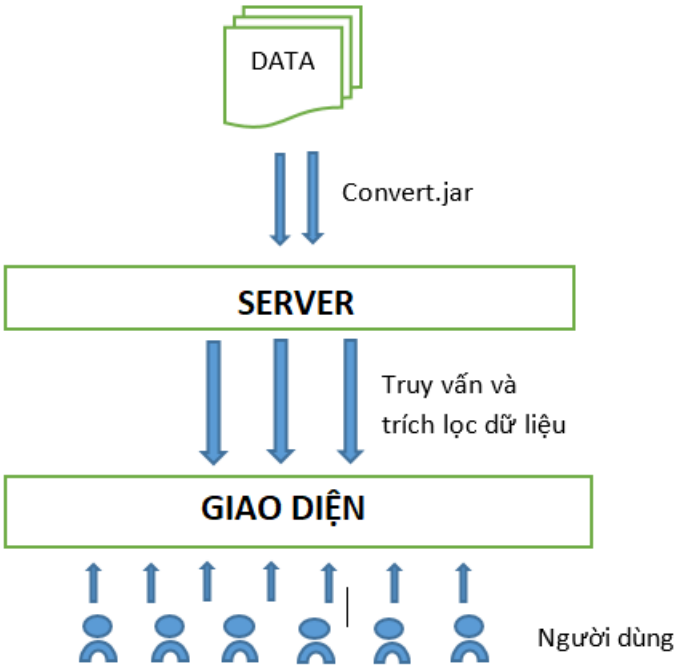
Trong chương này sẽ trình bày các vấn đề: thu thập dữ liệu bệnh nhân COVID-19 tại Việt Nam. Phân tích vấn đề và lập trình dựa trên dữ liệu từ Internet, thực hiện truy vấn để tìm kiếm các thông tin cần thiết.

3.1 Giới thiệu bài toán tìm kiếm thông tin bệnh nhân COVID-19

Từ dữ liệu hồ sơ bệnh án (trích xuất biểu đồ) có thể cho chúng ta có thể biết thêm về bệnh nhân COVID-19 và quá trình bệnh của họ. Những dữ liệu này có thể bao gồm thông tin về nhân khẩu học (tuổi, chủng tộc/dân tộc, giới tính), cũng như các triệu chứng, phương pháp điều trị và kết quả. Các nhà khoa học có thể sử dụng trích xuất biểu đồ để tìm hiểu những người có khả năng mắc bệnh nặng hơn, những dịch vụ chăm sóc y tế bệnh nhân đã nhận được và bệnh nhân đã hồi phục chưa. Với những nội dung đã tìm hiểu ở các phần trên trên, luận văn sẽ sử dụng công nghệ Web ngữ nghĩa để xây dựng một ứng dụng minh họa nhằm phục vụ cho việc lưu trữ, truy vấn, tìm kiếm các thông tin bệnh nhân COVID-19 tại Việt Nam.

3.1.1 Kiến trúc hệ thống

Ứng dụng hỗ trợ thông tin khách hàng gồm ba phần: phần thứ nhất là module chuyển đổi dữ liệu người bệnh hiện có sang định dạng bộ ba RDF sử dụng khung ứng dụng Jena. Phần thứ hai là lưu trữ dữ liệu và truy vấn dữ liệu theo định dạng bộ ba RDF. Phần thứ ba là chương trình hiển thị cho người sử dụng thấy các thông tin đã liên kết được:

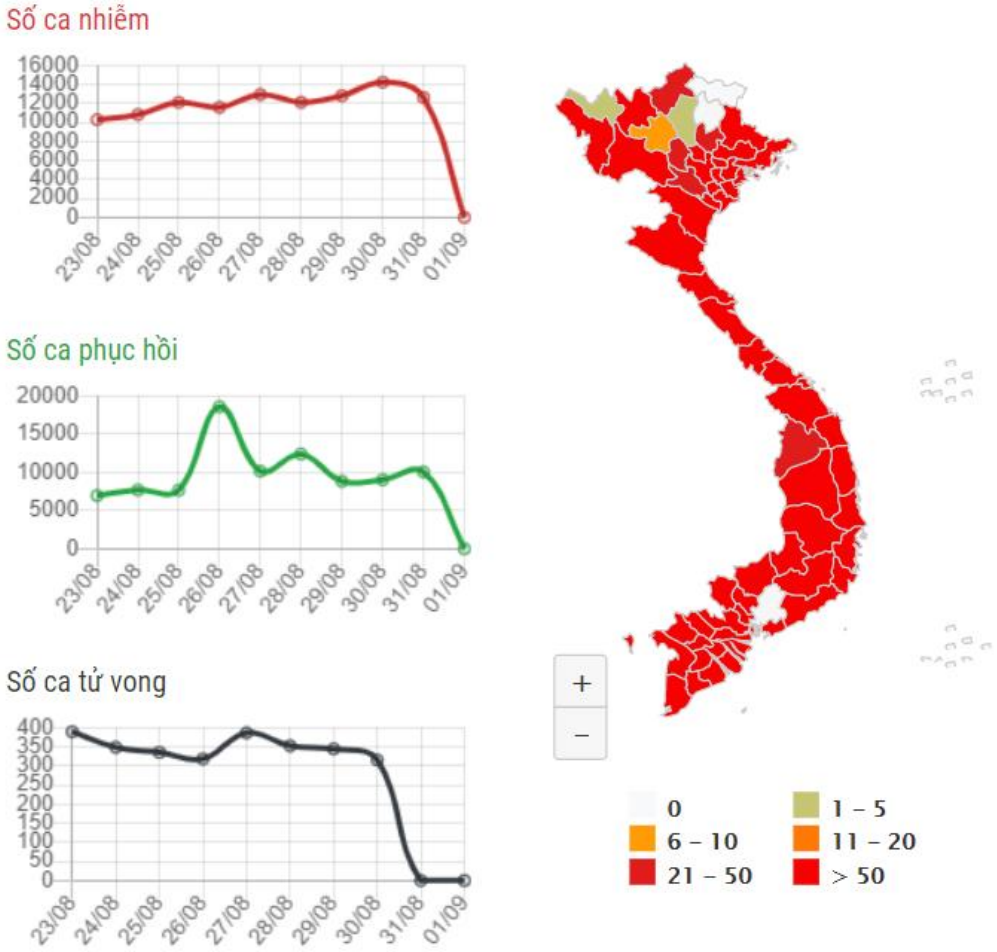


Hình 3.1. Kiến trúc chương trình

Trong Hình 3.1 thì ngôn ngữ sử dụng để cài đặt các thực nghiệm là ngôn ngữ lập trình Java có thể được phân cấp ra để chạy từng phân vùng, kèm theo các chú thích của các vùng trong đó, có thể kết nối tới tài nguyên của máy tính.

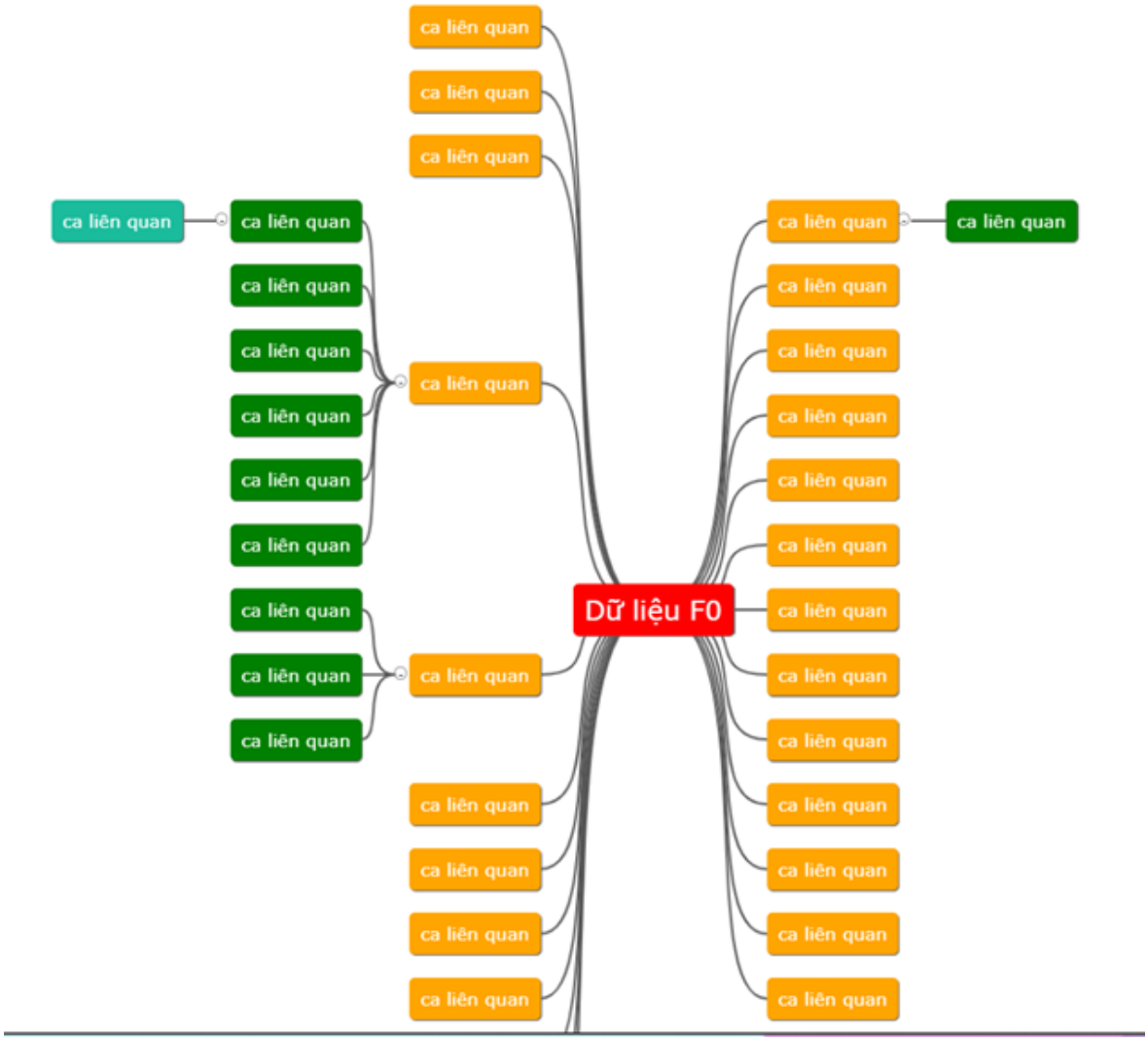
3.1.2 Ý nghĩa thực tiễn của hệ thống

Việc phòng, chống dịch Covid-19 là chưa có tiền lệ với nhân loại nên thực tế hơn một năm qua, các nước vừa làm, vừa học, rút kinh nghiệm. Tuy nhiên, việc quy mô những người nhiễm phải được điều trị trong các bệnh viện - họ là nguồn lây nhiễm chủ yếu tạo ra lây nhiễm cộng đồng và dịch ở các nước - không ngừng tăng lên, chưa có dấu hiệu trở về mức khi công bố có dịch, chứng tỏ việc phòng, chống Covid-19 của nhân loại về tổng thể là chưa đem lại kết quả toàn cầu mong muốn.

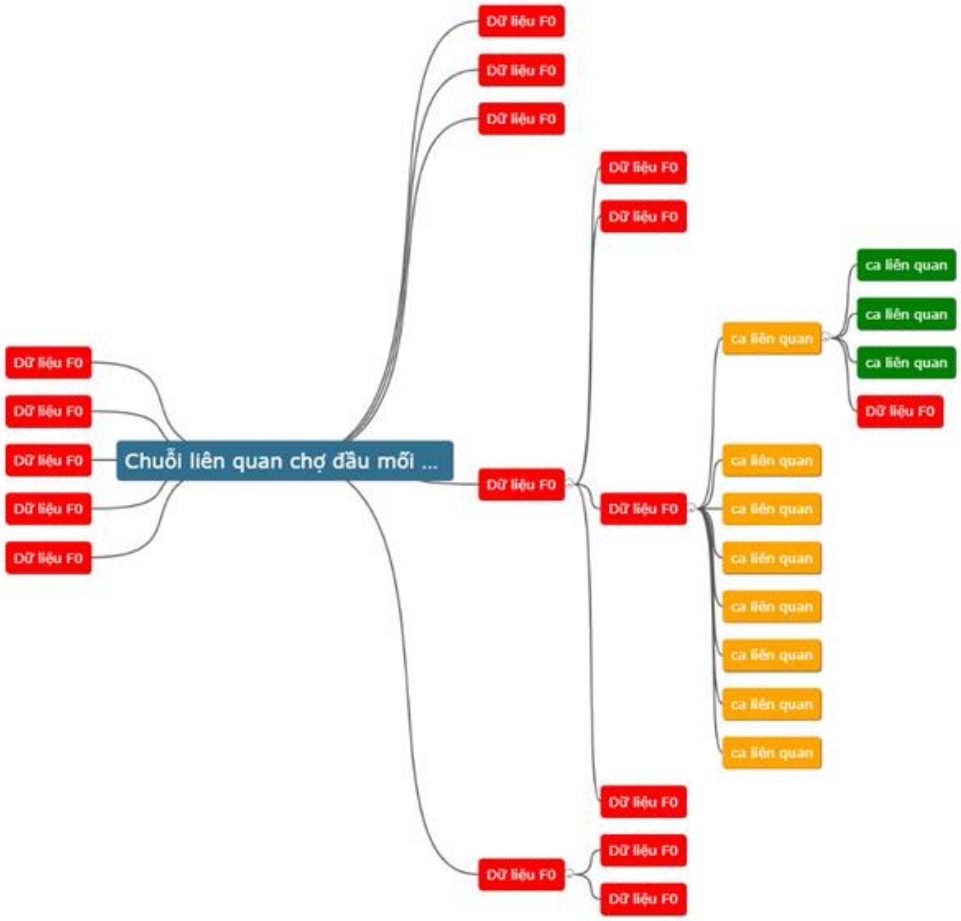


Hình 3.2. Thống kê dịch COVID-19 tại Việt Nam tính đến ngày 01/09/2021

Do đó hệ thống xây dựng để bám sát những thông báo khẩn liên tục được phát đi trên trang thông tin điện tử của Sở Y tế, Trung tâm Kiểm soát bệnh tật và các phương tiện thông tin đại chúng để lưu trữ thông tin di chuyển và tìm kiếm nhanh nhất những người đã từng tiếp xúc với F0. Việc điều tra truy vết, tìm kiếm các đối tượng trên thực tế cũng gặp rất nhiều khó khăn như rà soát tại địa bàn, điều tra dịch tễ để xác định có đúng F1 hay không, chưa kể các F1 có thể nằm rải rác ở các xã, phường khác nhau, thậm chí ở các tỉnh khác. Ngay sau đó, các địa phương lập danh sách F1 tiến hành lấy mẫu xét nghiệm và khẩn trương đưa đến các khu cách ly tập trung. Ngoài ra còn thống kê lại tổng số ca nhiễm, số lượng tử vong, số ca được chữa khỏi, cũng như các thông tin liên quan khác về dịch COVID-19 cho người dùng dễ dàng truy cập.



Hình 3.3. Mô tả mối liên hệ giữa các bệnh nhân COVID-19 với F0



Hình 3.4. Mô tả mối liên hệ giữa các F0 với địa điểm

3.1.3 Chuẩn bị dữ liệu cho bài toán

3.1.3.1 *Dữ liệu về bệnh nhân COVID-19*

Dữ liệu về thông tin người bệnh được lấy từ trang tin về dịch bệnh viêm đường hô hấp cấp COVID-19 của Bộ Y tế (Website: https://ncov.moh.gov.vn/). Dữ liệu sẽ được sao chép vào file Excel gồm các thông tin về bệnh nhân, địa điểm di chuyển của người bệnh.

a) Dữ liệu thử nghiệm

Dữ liệu được trích xuất từ các nguồn thông tin từ bản khai báo y tế và sổ sức khỏe điện tử của bệnh nhân F0 và cùng với đó là dịch trình di chuyển được ghi lại khi bệnh nhân đến các địa điểm khác nhau nếu được cấp phép. Các dữ liệu sẽ được tập trung thành thành tệp Excel để có thể dễ dàng lấy thông tin cho ứng dụng.

Bệnh nhân	Tuổi	Địa chỉ	Tình trạng	Quốc tịch
BN9	30	Vĩnh Phúc	Khỏi	Việt Nam
BN8	29	Vĩnh Phúc	Đang điều trị	Việt Nam
BN7	73	Hồ Chí Minh	Đang điều trị	Hoa Kỳ
BN6	25	Khánh Hòa	Khỏi	Việt Nam
BN5	23	Vĩnh Phúc	Đang điều trị	Việt Nam
BN4	29	Vĩnh Phúc	Khỏi	Việt Nam
BN3	25	Thanh Hóa	Khỏi	Việt Nam
BN2	28	Hồ Chí Minh	Đang điều trị	Trung Quốc
BN1	66	Hồ Chí Minh	Khỏi	Trung Quốc
..	…	…	…	…

Hình 3.5. Dữ liệu về thông tin người bệnh

Tên	Tên địa điểm	ID	Kinh độ	Vĩ độ
DD1	Gò Đống Đa	1111	21.01355	105.82253
DD2	Học Viện Công nghệ Bưu Chính Viễn Thông	1112	20.9809453	105.787094
DD3	Đại học Kiến trúc Hà Nội	1113	20.98039885	105.7896742
DD4	Học viện An Ninh	1114	20.98048795	105.7921328
DD5	Highland Coffee - Hà Đông	1115	20.966005	105.772019
DD6	Trường THPT Nguyễn Huệ	1116	20.9725	105.77552
DD7	Ga Hà Nội	1117	21.03482	105.84704
DD8	Sân bay Nội Bài	1118	10.22511	103.97227
DD9	VP Công chứng Cầu Giấy	1119	21.04135	105.78074
…	…	…	…	…

Hình 3.6. Dữ liệu về thông tin địa điểm

Thời gian	Bệnh nhân	Địa điểm
20/01/2021	BN1	DD1
20/01/2021	BN1	DD2
20/01/2021	BN1	DD3
20/01/2021	BN2	DD1
20/01/2021	BN2	DD4
20/01/2021	BN2	DD5
20/01/2021	BN3	DD4
20/01/2021	BN3	DD6
21/01/2021	BN1	DD2
21/01/2021	BN1	DD7
21/01/2021	BN1	DD8
21/01/2021	BN2	DD9
21/01/2021	BN2	DD3
21/01/2021	BN2	DD2
21/01/2021	BN3	DD8

21/01/2021	BN3	DD1
22/01/2021	BN1	DD1
22/01/2021	BN1	DD2
22/01/2021	BN1	DD3
22/01/2021	BN2	DD2
22/01/2021	BN2	DD9
22/01/2021	BN2	DD1
22/01/2021	BN3	DD5
22/01/2021	BN3	DD1
…	…	…

Hình 3.7. Dữ liệu về thông tin sự kiện

b) Kịch bản thử nghiệm

- Hàng ngày phường/xã/thị trấn sẽ cập nhật thông tin phát sinh trên địa bàn vào phần mềm, kết xuất báo cáo chuyển báo cáo cho cấp quản lý theo qui định.
- Các cơ sở y tế sẽ cập nhật thông tin phát sinh do đơn vị quản lý vào phần mềm, kết xuất báo cáo chuyển báo cáo cho cấp quản lý theo qui định.
- Các Ban, phòng ban thực hiện báo cáo cho thường trực ủy ban cấp quận/huyện/thành phố và báo cáo cho Tỉnh/Thành phố theo qui định: thông tin tập hợp từ các phường/xã/thị trấn, các cơ sở y tế.
- Kiểm tra và tạo dữ liệu cho hệ thống.
- Tạo mô hình RDF từ dữ liệu thử nghiệm.
- Đọc và truy xuất dữ liệu từ mô hình RDF.
- Hiển thị thông tin lên trang web xuất báo cáo.

3.1.3.2 Dữ liệu từ OpenStreetMap

a) Giới thiệu Bản đồ thế giới mở

OpenStreetMap, hoặc OSM, giống như Wikipedia cho bản đồ. Đó là nguồn mở, người dùng đã đóng góp, và miễn phí để sử dụng, và, giống như Wikipedia, nó được chứng minh là đáng tin cậy đáng kinh ngạc. Ngay cả khi bạn chưa từng nghe về

nó trước đây, rất có thể là bạn đã sử dụng dữ liệu của nó. Nó được sử dụng bởi FourSquare, Pokemon Go, Craigslist, Tesla, và một danh sách dài các ứng dụng và dịch vụ khác. Trong khi nó không phải là hữu ích cho chuyển hướng hàng ngày, bộ dữ liệu không gian địa lý rộng lớn của nó đã nhiều lần chứng minh hữu ích không chỉ cho các ứng dụng, nhưng cho công việc nhân đạo và cứu trợ khẩn cấp .

OpenStreetMap ban đầu được tạo ra ở Anh bởi Steve Coast, và kể từ khi thành lập năm 2004, cơ sở dữ liệu thông tin bản đồ của nó đã phát triển đáng kể. Dữ liệu của họ đến từ việc đóng góp hàng loạt từ các thực thể chứa rất nhiều dữ liệu bản đồ và từ những người dùng tặng thời gian và năng lượng của họ để giúp xây dựng và duy trì bản đồ.

Cũng giống như Wikipedia, nếu bạn làm một công việc tốt, các chỉnh sửa của bạn sẽ ở lại. Mặt khác, nếu bạn phá hoại bản đồ hoặc mắc lỗi, một người nào đó sẽ bắt và cuộn nó trở lại. Hệ thống này đã hoạt động khá tốt. Vào năm 2018, hầu hết dữ liệu không gian địa lý của nó tốt hoặc tốt hơn Google Maps.

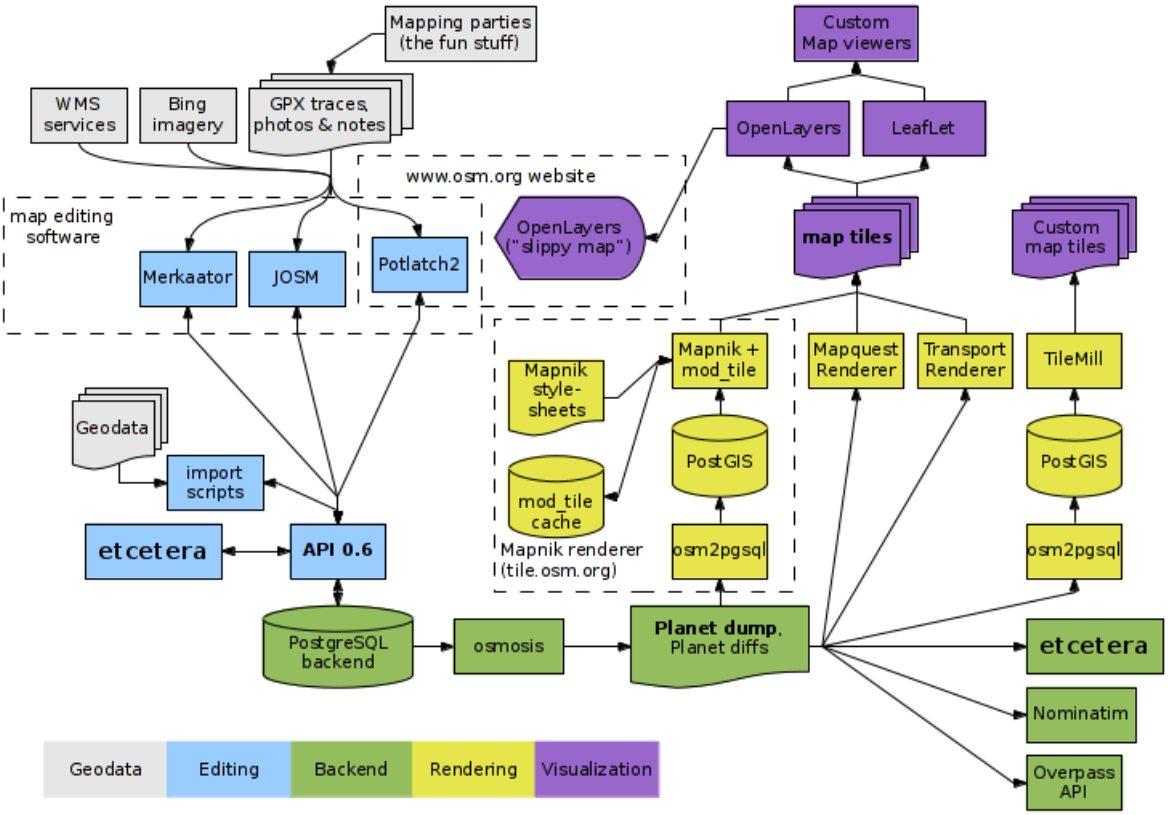
b) Kiến trúc OpenStreetMap

Geodata: là kho chứa dữ liệu thuộc tính và không gian trong hệ quản trị cơ sở dữ liệu.

Editing: là các thành phần chỉnh sửa dữ liệu.

Backend: là các thành phần trích xuất và truy vấn dữ liệu.

Rendering: là các thành phần kiến tạo mô hình dữ liệu.



Hình 3.8. Kiến trúc OpenStreetMap

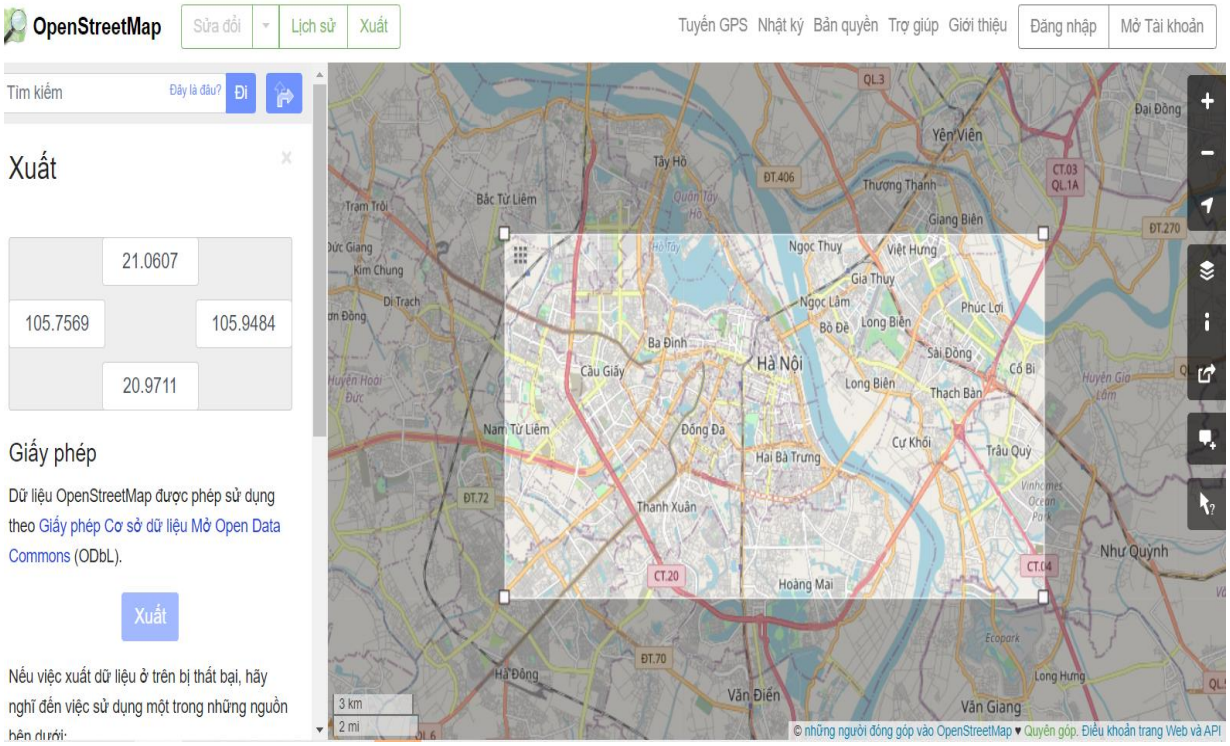
c) Các thao tác trên OpenStreetMap

➢ Trích xuất dữ liệu OpenStreetMap

Bước 1: truy cập https://www.OpenStreetMap.org

Bước 2: chọn Xuất -> di chuyển, zoom vùng bản đồ để chọn vùng bản đồ sẽ xuất dữ liệu. Hoặc chọn vùng thủ công khác để chọn vùng bản đồ.

Bước 3: chọn Xuất để tải bản đồ về dưới dạng *.osm

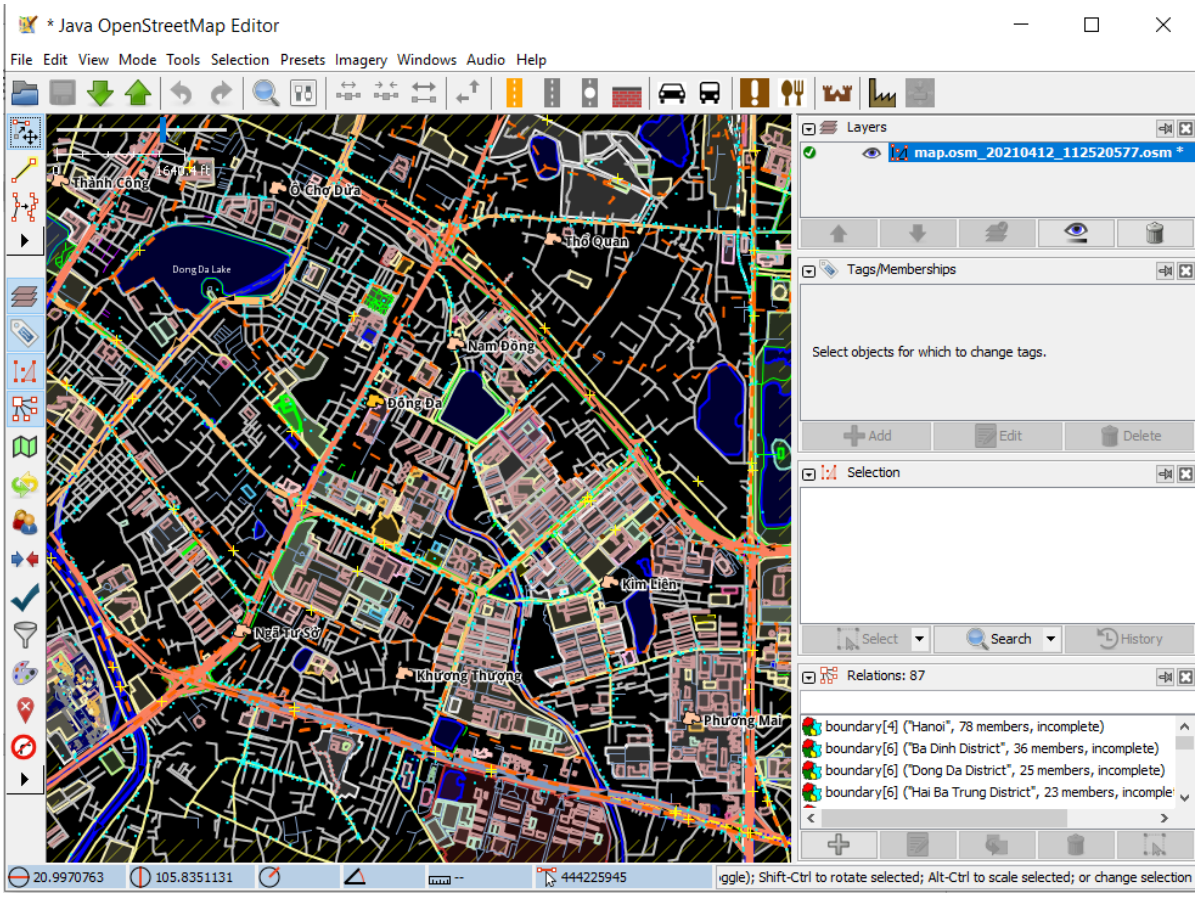


Hình 3.9. Trích xuất dữ liệu OpenStreetMap

➢ Đọc dữ liệu trên OpenStreetMap

Bước 1: sử dụng JOSM download từ https://josm.OpenStreetMap.de

Bước 2: chọn open -> chọn tới file *.osm đã tải về



Hình 3.10. Đọc dữ liệu từ OpenStreetMap với JOSM

c) Các thao tác trên OpenStreetMap

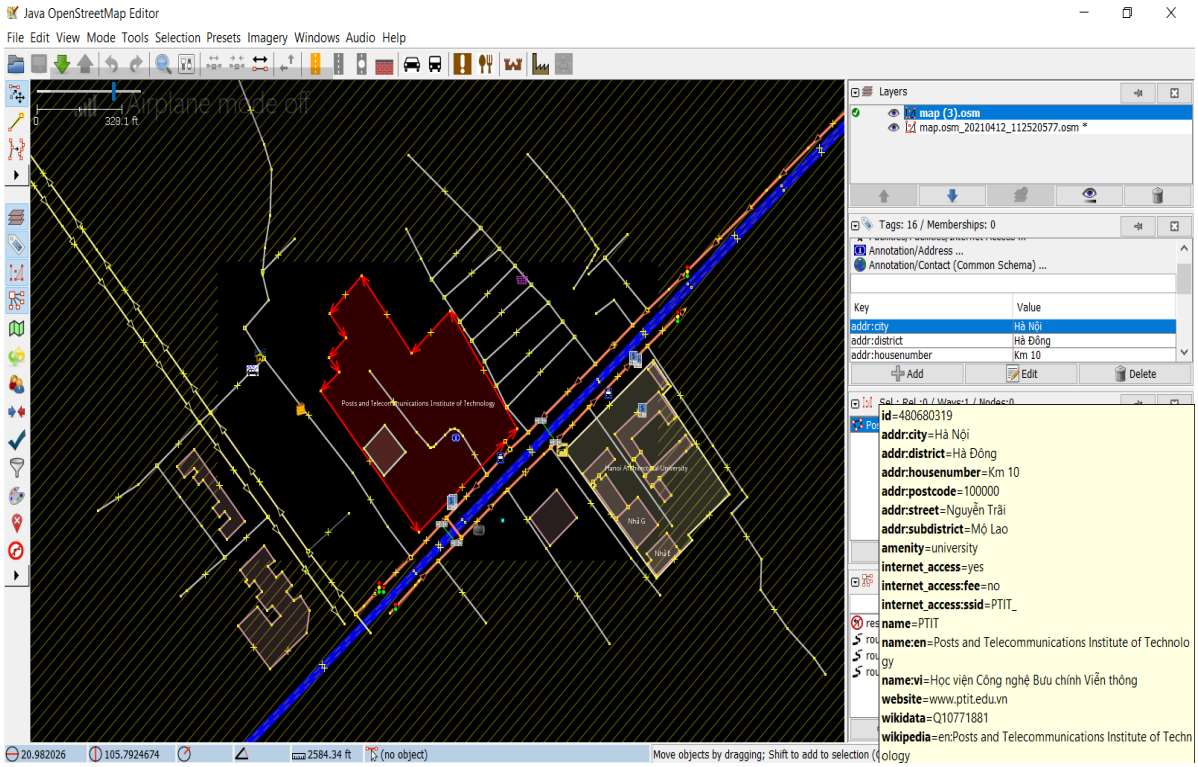
OpenStreetMap Semantic Network là một nguồn tài nguyên Web Semantic trích từ trang web của OSM Wiki, mã hóa như là một từ vựng SKOS. Nó chứa một máy đại diện có thể đọc được các thẻ OSM, và một số quan hệ ngữ nghĩa giữa chúng. Mạng lưới này được liên kết mở dữ liệu. Mạng ngữ nghĩa này được chiết xuất bởi một trình thu thập nguồn mở được phát triển trong Java / Groovy, các OSM Wiki Crawler. Các thẻ OSM được ánh xạ vào các khái niệm tương ứng trong WordNet và LinkedGeoData.

Trong OpenStreetMap, tính năng bản đồ được mô tả bằng các thẻ, chẳng hạn như tiện nghi là đại học, và đường cao tốc là chính. Những thẻ này tương ứng với khái niệm địa lý, đại diện trong mạng ngữ nghĩa OSM của chúng tôi. Sử dụng mạng ngữ nghĩa OSM có thể tính toán sự tương đồng ngữ nghĩa của các thẻ. Ví dụ, một

trường đại học là khái niệm tương tự như một trường học hơn với một dòng sông. Một biện pháp tương tự ngữ nghĩa nhằm định lượng thực tế này với một số thực, thường là giữa 0 và 1

d) Ứng dụng trong hệ thống

Hệ thống tìm kiếm khi nhập vào một địa điểm sẽ được tìm kiếm bởi các tập tin RDF trên hệ thống. Sau khi trích được thông tin dựa trên OSM_ID sẽ được nhúng vào OpenStreetMap để hiển thị lên bản đồ.

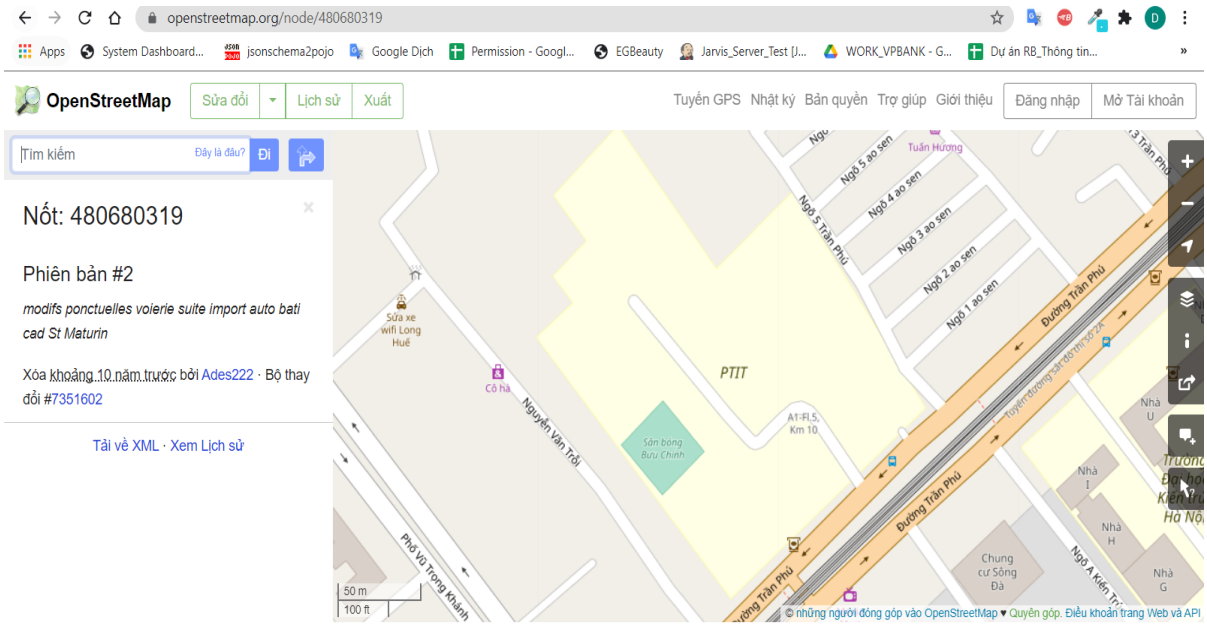


Hình 3.11. Dữ liệu Học viện Công nghệ Bưu chính Viễn thông trên OpenStreetMap

Ví dụ: với từ khóa: “Học viện Công nghệ Bưu chính Viễn thông” khi tìm kiếm sẽ hiển thị kết quả chi tiết. Click “Thêm kết quả” sẽ lấy được node = 480680319, với vĩ độ = 20.9809453 và kinh độ = 105.787094 . Khi muốn hiển thị địa điểm này trên trình duyệt dựa trên địa chỉ sau:

https://www.OpenStreetMap.org/node/480680319

Lúc này ta có chèn vào HTML bằng thẻ iframe khi bấm vào nút chia sẻ. Chi tiết như hình dưới đây:



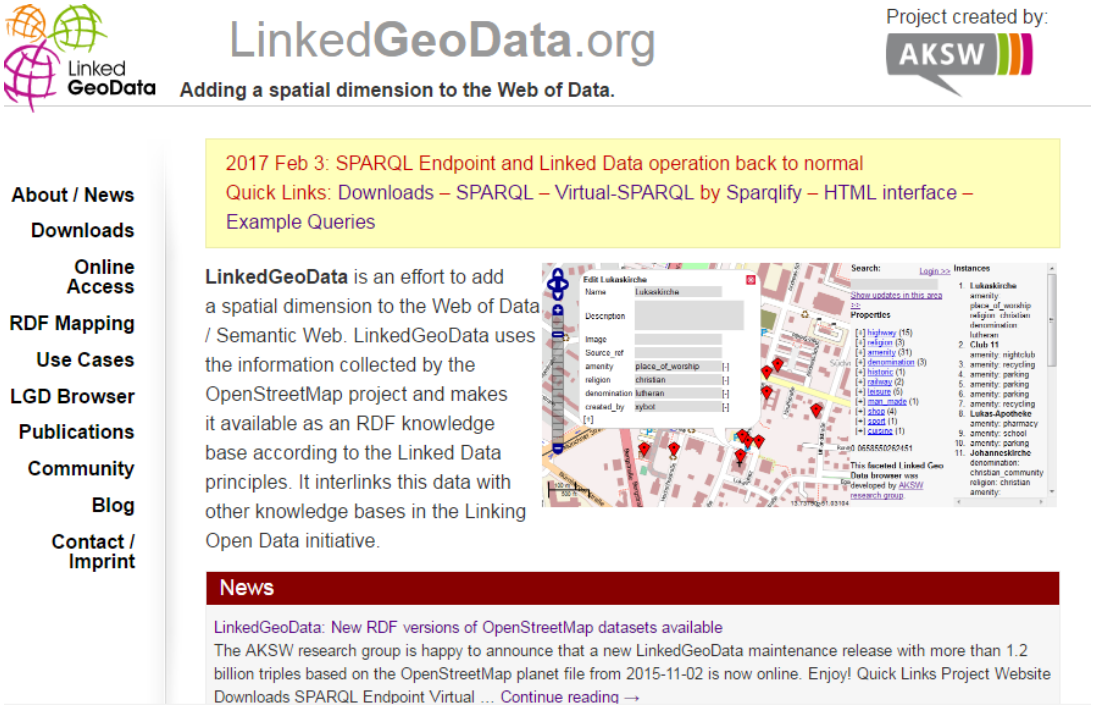
Hình 3.12. Hiển thị thông tin một node dựa vào OSM_ID

3.1.3.3 *Dữ liệu từ LinkedGeoData*

a) LinkedGeoData

LinkedGeoData là dự án của nhóm nghiên cứu AKSW với hơn 1,2 tỉ lượt truy cập dựa thông tin trên tập tin hành tinh OpenStreetMap từ năm 2015.

LinkedGeoData là một dự án để thêm một không gian vào dữ liệu của Web ngữ nghĩa. LinkedGeoData sử dụng thông tin thu thập được bởi dự án OpenStreetMap và làm cho nó có sẵn dưới dạng cơ sở tri thức RDF theo nguyên tắc dữ liệu có liên kết. Nó liên kết dữ liệu này với các cơ sở tri thức khác trong sáng kiến liên kết dữ liệu mở.



Hình 3.13. LinkedGeoData.org

b) OpenStreetMap Dataset

LinkedGeoData chứa các tập tin RDF phiên bản mới cho OpenStreetMap được lưu dưới định dạng *.nt.

Cách tiếp cận của dữ liệu Địa lý liên kết là lưu trữ thông tin lập bản đồ trong cơ sở dữ liệu quan hệ (Post GIS) cùng với dữ liệu OSM và sử dụng trình ghi đè SPARQL-to-SQL Sparqlify để tạo ra RDF.

Các tập lệnh SQL LinkedGeoData mở rộng một giản đồ đơn giản có chứa OSM với các chỉ mục, bảng biểu và khung nhìn bổ sung.

c) Một số truy vấn cơ bản

Liệt kê tất cả các tài nguyên từ Leipzig Central Station.

```
Prefix lgdo: <http://linkedgeodata.org/ontology/>
Prefix geom: <http://geovocab.org/geometry#>
Prefix ogc: <http://www.opengis.net/ont/geosparql#>
Prefix owl: <http://www.w3.org/2020/07/owl#>
Select * {
  ?s
    owl:sameAs <http://dbpedia.org/resource/Leipzig_Hauptbahnhof> ;
    geom:geometry [  ogc:asWKT ?sg  ] .
```

```
  ?x
    a lgdo:Amenity ;    rdfs:label ?l ;
    geom:geometry [   ogc:asWKT ?xg    ] .
    Filter(bif:st_intersects (?sg, ?xg, 0.1)) .
} Limit 10
```

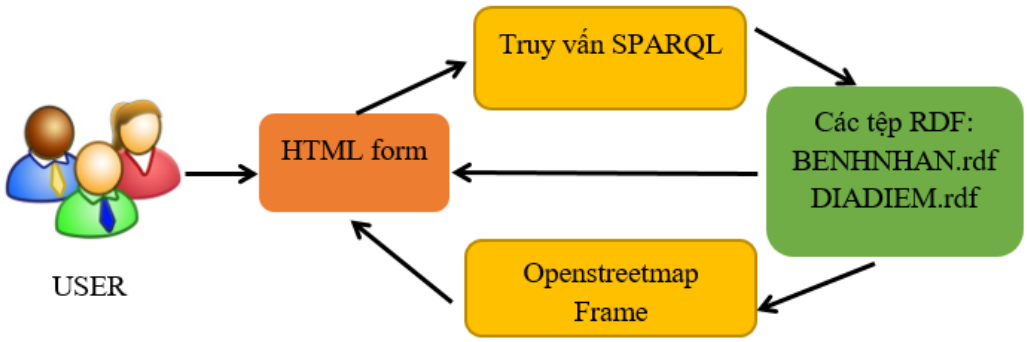
Kết quả hiển thị dưới định dạng HTML:

s	sg	x
http://linkedgeodata.org/triplify/node376142577	"POINT(12.3830858 51.3465518)"^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>	http://linkedgeodata.org/triplify/way108056031
http://linkedgeodata.org/triplify/node376142577	"POINT(12.3830858 51.3465518)"^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>	http://linkedgeodata.org/triplify/way108056036
http://linkedgeodata.org/triplify/node376142577	"POINT(12.3830858 51.3465518)"^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>	http://linkedgeodata.org/triplify/way115108685
http://linkedgeodata.org/triplify/node376142577	"POINT(12.3830858 51.3465518)"^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>	http://linkedgeodata.org/triplify/way115108944
http://linkedgeodata.org/triplify/node376142577	"POINT(12.3830858 51.3465518)"^^<http://www.openlinksw.com/schemas/virtrdf#Geometry>	http://linkedgeodata.org/triplify/way129765234

Hình 3.14. Kết quả hiển thị Linkedgeodata.org trên HTML

3.2 Giải pháp hệ thống

3.2.1 Mô hình hệ thống



Hình 3.15. Mô hình hệ thống tra cứu thông tin bệnh nhân

Khi người dùng chọn người bệnh, quá trình tìm kiếm từ khóa sẽ được thực hiện tại truy vấn SPARQL trên dữ liệu RDF từ các tệp BENHNHAN.rdf ,DIADIEM.rdf và LICHSU.rdf. Các thông tin từ file RDF sẽ được hiển thị trực tiếp lên HTML form, đồng thời dựa vào thông tin từ OpenStreetMap để hiển thị lên trình duyệt thông tin chi tiết địa điểm cần tìm kiếm.

3.2.2 Tạo dữ liệu RDF

Với các dữ liệu tải về từ OpenStreetMap, tôi sẽ dùng Jena API để chuyển đổi sang dữ liệu RDF và được lưu với tên DIADIEM.rdf. Đồng thời để tạo dữ liệu ngữ nghĩa minh họa tôi tạo thêm tập tin BENHNHAN.rdf, LICHSU.rdf từ dữ liệu file Excel để biết thông tin bệnh nhân và lịch sử di chuyển của họ. Tiếp đó, tôi dùng bộ từ vựng FOAF(http://xmlns.com/foaf/0.1/) để mô tả các thông tin trên lần lượt tương ứng với các thuộc tính sau: FOAF.name, FOAF.age, FOAF.openid, FOAF.status, cùng các bộ từ vựng của Dbpedia(https://dbpedia.org/ontology/) và DC (http://purl.org/dc/elements/1.1/)

Về giải pháp thì có sử dụng 3 bộ dữ liệu:

- Bộ Dbpedia: miêu tả 1 địa điểm trên bản đồ
- Bộ DC để miêu tả 1 sự kiện
- Bộ FOAF để miêu tả các thuộc tính (Properties) của bệnh nhân

  Ví dụ:

- ✓ FOAF.name là miêu tả tên bệnh nhân.
- ✓ FOAF.age tuổi bệnh nhân.
- ✓ FOAF.based_near miêu tả địa chỉ bệnh nhân.
- ✓ FOAF.status miêu tả tình trạng của bệnh nhân.

Ngoài ra thì các mối quan hệ (Relationship) còn được miêu tả như sau:

Bệnh nhân 1 sẽ có quan hệ “seeAlso” – tức là đã tham gia sự kiện A. Sự kiện A gồm thuộc tính thời gian sẽ có quan hệ “seeAlso” 1 địa điểm xác định nào đó . Từ đó ta có thể dễ dàng tìm kiếm được bệnh nhân đã đi đến đâu, vào giờ nào. Và từ mối quan hệ giữa bệnh nhân – sự kiện –địa điểm, ta có thể nhanh chóng truy vết được các

địa điểm và thời gian liên quan mà bệnh nhân đến. Tất cả các ca F1 sẽ được truy vết theo thời gian và địa điểm trong sự kiện. Chi tiết quá trình trên được thể hiện bằng đoạn code được thể hiện ở phần Phụ lục.

3.2.3 Truy vấn dữ liệu

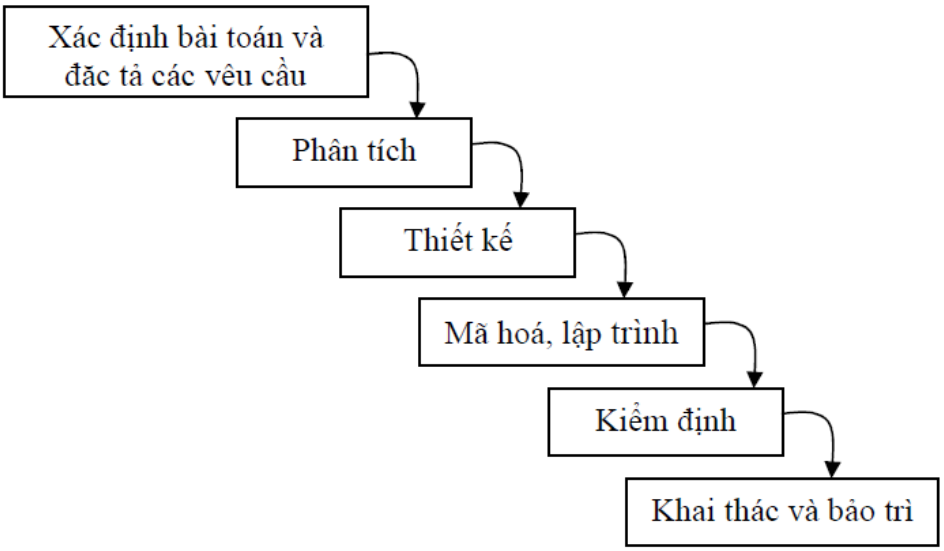
Để thực hiện truy vấn trên tập tin RDF, tôi sử dụng ngôn ngữ truy vấn Web ngữ nghĩa SPARQL với nội dung nhập từ HTML form. Sau đó xuất kết quả truy vấn dưới định dạng HTML. Sử dụng ngôn ngữ lập trình Java.

Để kết quả tìm kiếm một từ khóa bất kì tôi sử dụng truy vấn SPARQL có sử dụng các mẫu (Regex pattern) để trả về các tài nguyên có chứa từ khóa đó (phụ lục).

3.3 Xây dựng hệ thống

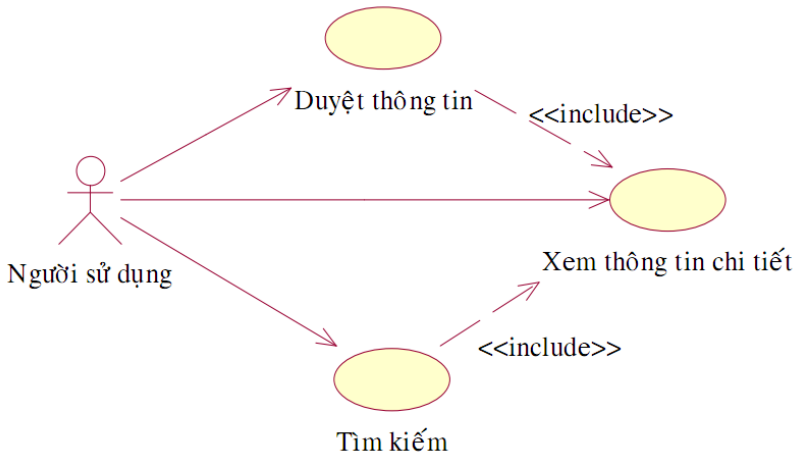
3.3.1 Mô tả hệ thống

Hệ thống được xây dựng trên mô hình thác nước do ứng dụng không quá lớn và các chức năng đã được liệt kê một cách rõ ràng.

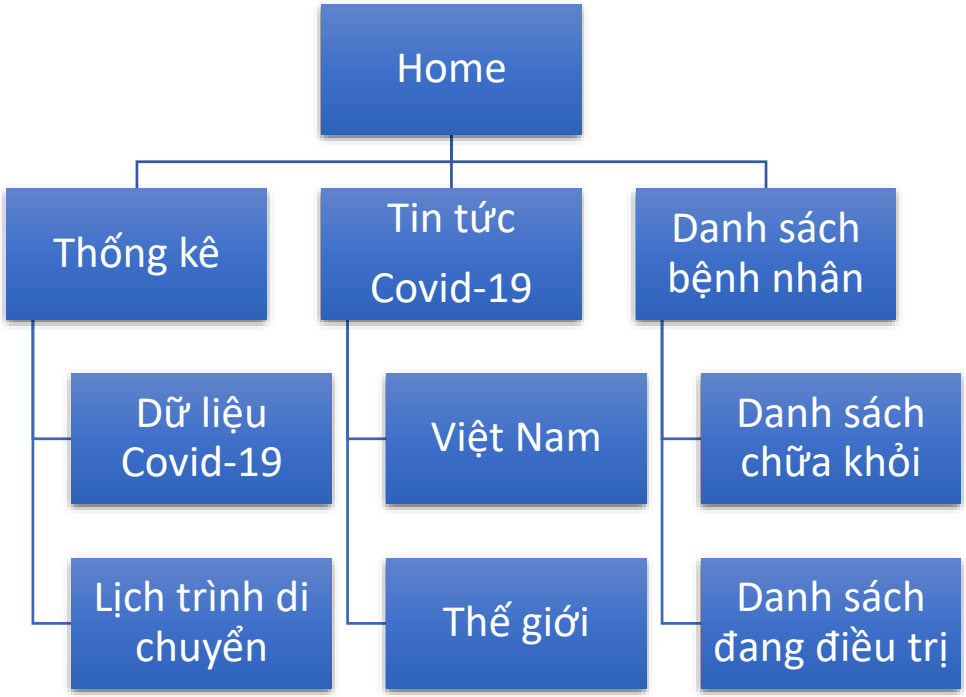


Hình 3.16. Mô hình phát triển hệ thống

Phần này sẽ xây dựng mô hình hệ thống với mục đích cung cấp một cách chi tiết về các chức năng cơ bản như tìm kiếm thông tin, xem thông tin bệnh nhân (duyệt thông tin) và xem các thông tin chi tiết của địa điểm. Mô hình được thể hiện như hình sau:



Hình 3.17. Use case hệ thống tìm kiếm thông tin bệnh nhân COVID-19



Hình 3.18. Sơ đồ kiến trúc thông tin

3.3.2 Đặc tả chức năng

"**Ứng dụng công nghệ Web ngữ nghĩa và OpenStreetMap phát triển ứng dụng hỗ trợ tìm kiếm thông tin người bệnh COVID-19**" nhằm giải quyết các yêu cầu:

✓ Hỗ trợ các đơn vị trong việc quản lý, giám sát tức thời tình hình dịch bệnh trên địa bàn. Tự động truy vết lây nhiễm, tiếp xúc khi có ca bị nhiễm. Giảm tải và nhanh chóng truy vết, truy vết 1 hoặc nhiều chuỗi/ổ dịch, đặc biệt trong các đợt bùng phát dịch quy mô lớn có F0, F1, F2, F3 lớn, không thể thực hiện thủ công được.

✓ Hỗ trợ quản trị cập nhật, bảo trì hệ thống dễ dàng.

Các chức năng chính của người dùng hệ thống bao gồm:

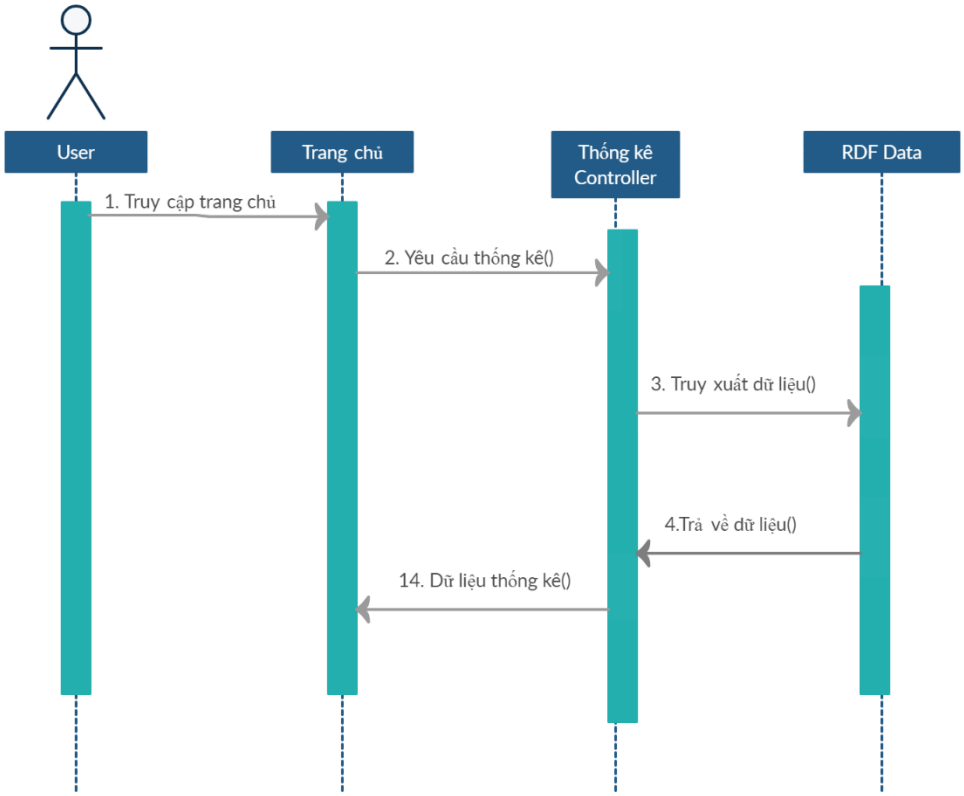
- Chức năng tìm kiếm: cho phép người sử dụng truy xuất thông tin trong hệ thống bằng cách chọn người bệnh trên form. Hệ thống duyệt file DIADIEM.rdf tìm tất cả các tài nguyên liên quan đến từ khóa muốn tìm và trả về kết quả. Khi lấy được thông tin phù hợp sẽ hiển thị tại khung OpenStreetMap.

- Chức năng duyệt thông tin: Chức năng này cho phép người sử dụng xem thông tin trong danh sách các tài nguyên liên quan đến từ khóa bằng cách chọn mục thông tin cần xem. Hệ thống tìm các đối tượng liên quan đến thông tin được chọn và trả về các kết quả cho người sử dụng. Kết quả được hiển thị tại khung OpenStreetMap.

- Chức năng xem thông tin chi tiết: Chức năng này cho phép người sử dụng xem thông tin chi tiết của một tài nguyên trong hệ thống dựa trên khung OpenStreetMap.

3.3.3 Phân tích hệ thống

- **Thống kê**



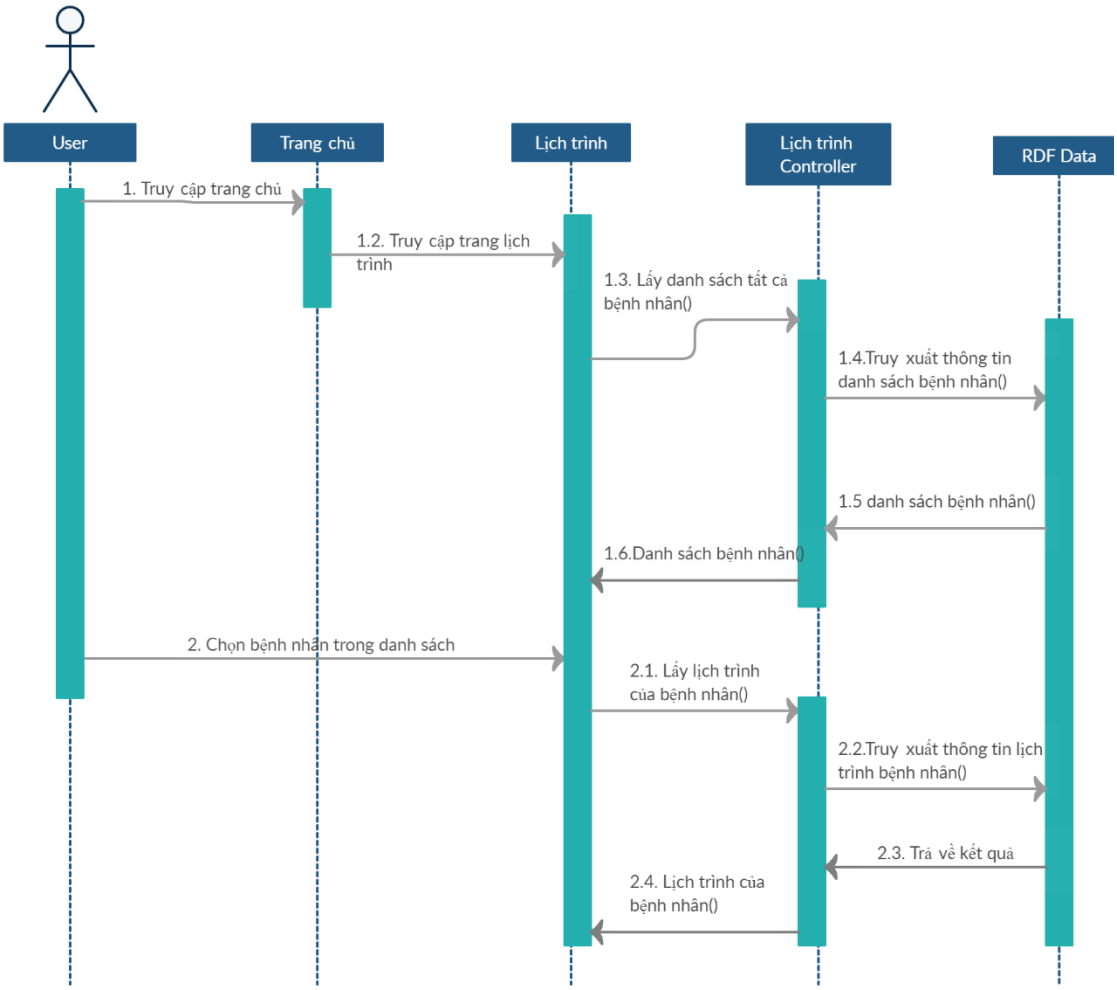
Hình 3.19. Biểu đồ tuần tự chức năng thống kê

Tên Use Case	**Xem danh sách học viên**
Tác nhân chính	Người dùng
Người chịu trách nhiệm	Người dùng
Đảm bảo tối thiểu	Người dùng truy cập vào hệ thống
Đảm bảo thành công	Người dùng xem được thông tin bao gồm: + Tổng số bệnh nhân. + Tổng số bệnh nhân mắc Covid-19 hôm nay. + Số người chết. + Số người được chữa khỏi. + Biểu đồ tổng số ca theo ngày. + Biểu đồ tổng số ca.
Kích hoạt	Click vào button “Thống kê”

Chuỗi sự kiện chính:

1. Người dùng truy cập vào hệ thống.
2. Hệ thống hiển thị danh sách menu gồm Thống kê, Tin tức, Danh sách bệnh nhân..
3. Người dùng chọn "Thống kê".
4. Hệ thống hiển thị toàn bộ thông tin và biểu đồ.

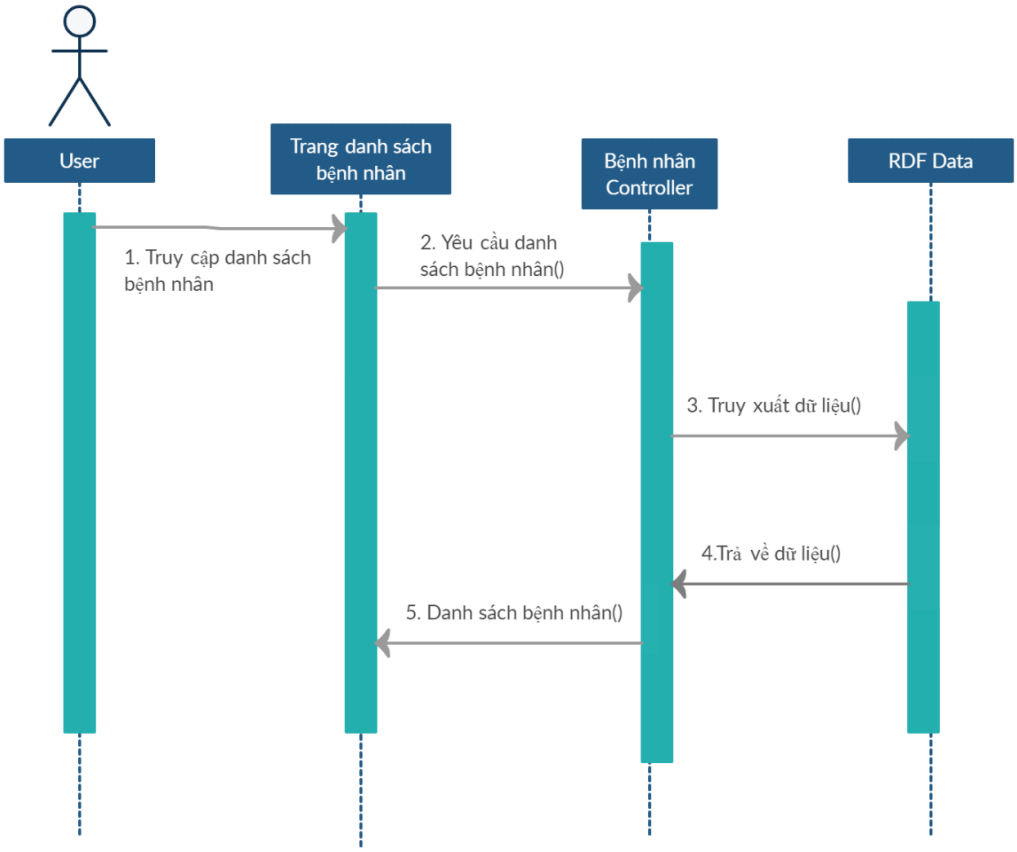
Xem lịch trình di chuyển của bệnh nhân



Hình 3.20. Biểu đồ tuần tự chức năng xem lịch trình di chuyển của bệnh nhân

Tên Use Case	**Xem danh sách học viên**
Tác nhân chính	Người dùng
Người chịu trách nhiệm	Người dùng
Đảm bảo tối thiểu	Người dùng truy cập vào hệ thống
Đảm bảo thành công	Người dùng xem được thông tin bao gồm: + Danh sách các bệnh nhân. + Lịch trình di chuyển của bệnh nhân.
Kích hoạt	Click vào button “Lịch trình”
Chuỗi sự kiện chính: 1. Người dùng truy cập vào hệ thống. 2. Hệ thống hiển thị danh sách menu gồm Thống kê, Tin tức, Danh sách bệnh nhân.. 3. Người dùng chọn “Lịch trình”. 4. Hệ thống hiển thị toàn bộ danh sách bệnh nhân đã được lưu trữ. 5. Người dùng chọn bệnh nhân muốn xem lịch trình di chuyển. 6. Hệ thống hiển thị lịch trình di chuyển của bệnh nhân	

Xem danh sách bệnh nhân



Hình 3.21. Biểu đồ tuần tự chức năng xem danh sách bệnh nhân

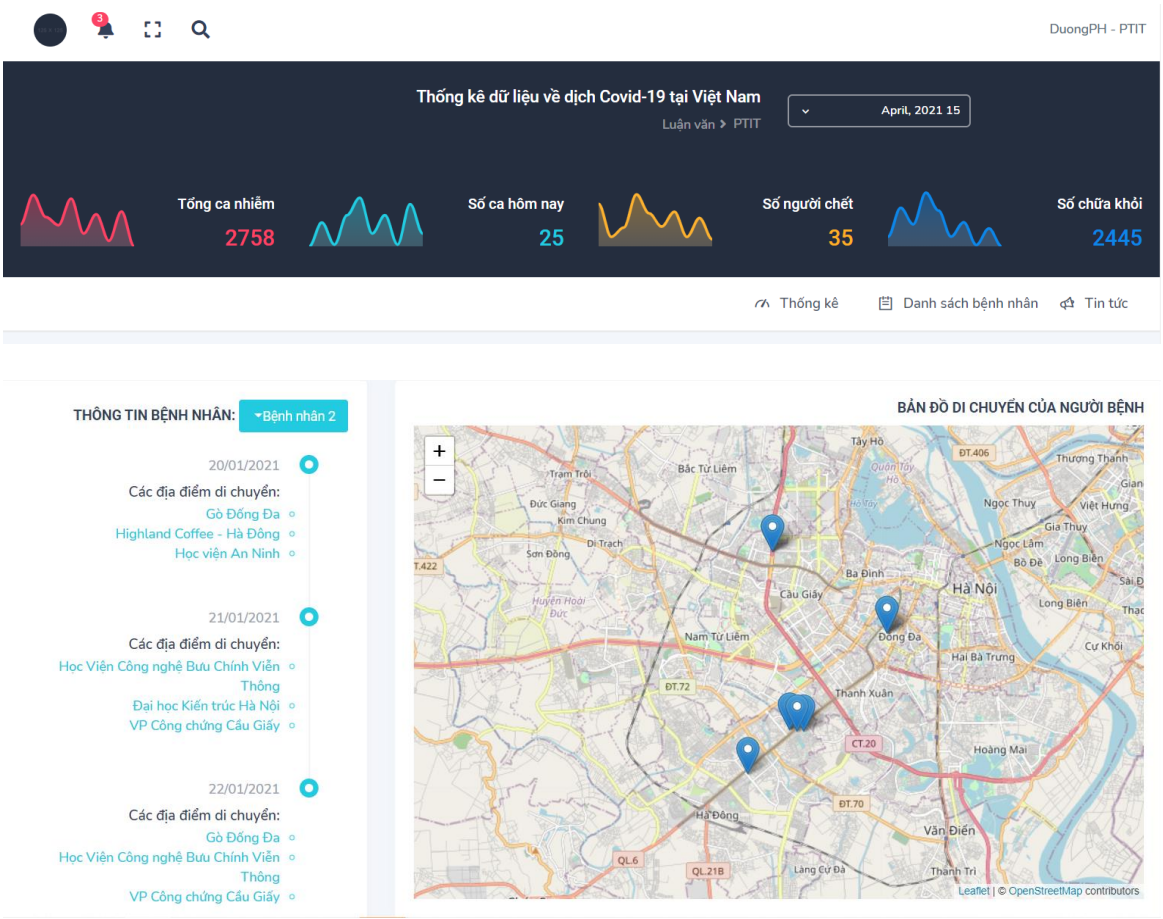
Tên Use Case	**Xem danh sách học viên**
Tác nhân chính	Người dùng
Người chịu trách nhiệm	Người dùng
Đảm bảo tối thiểu	Người dùng truy cập vào hệ thống
Đảm bảo thành công	Người dùng xem được thông tin danh sách tất cả các bệnh nhân và thông tin của họ.
Kích hoạt	Click vào button “Danh sách bệnh nhân”
Chuỗi sự kiện chính: 1. Người dùng truy cập vào hệ thống. 2. Hệ thống hiển thị danh sách menu gồm Thống kê, Tin tức, Danh sách bệnh nhân.. 3. Người dùng chọn “Danh sách bệnh nhân”. 4. Hệ thống hiển thị toàn bộ thông tin và biểu đồ.	

3.3.4 Thiết kế giao diện

- ***Giao diện trang thống kê***

Hình 3.22. Giao diện trang thống kê

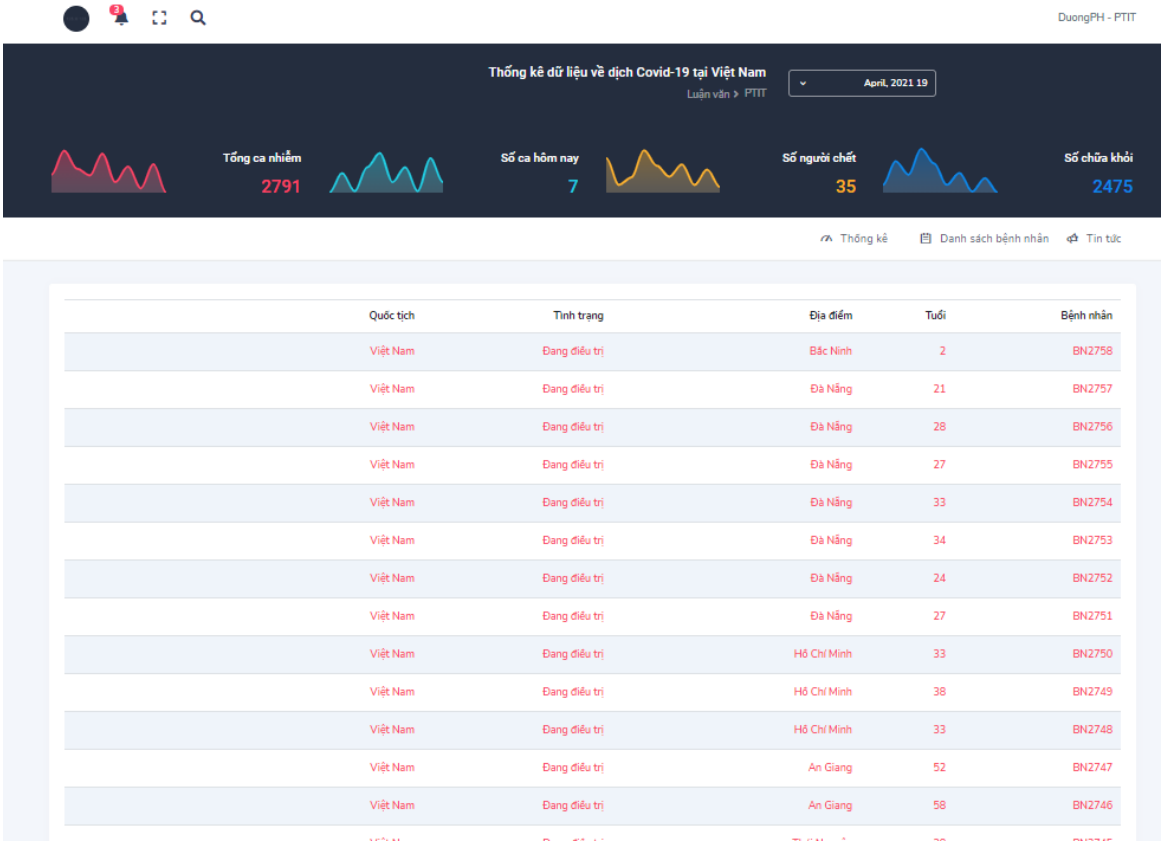
- ***Giao diện tra cứu thông tin lịch trình di chuyển của người bệnh***



Hình 3.23. Giao diện tra cứu thông tin lịch trình di chuyển của người bệnh.

Khi người dùng chọn bệnh nhân hệ thống sẽ hiển thị tất cả các sự kiện, địa điểm chính xác của người bệnh tương ứng di chuyển trong thời gian trước theo ngày cụ thể. Ngoài ra các thông số như Tổng ca nhiễm, Số ca hôm nay, Số người chết, Số chữa khỏi được cập nhật mới nhất từ Website của bộ y tế https://ncov.moh.gov.vn/

- ***Giao diện trang chi tiết thông tin người bệnh***



Hình 3.24. Giao diện chi tiết thông tin người bệnh COVID-19

3.4 Kết luận chương

Trong Chương 3 đã tổng hợp lại các kiến thức về Web ngữ nghĩa để xây dựng ứng dụng minh họa. Trước hết là phần mô tả về bài toán tìm kiếm thông tin bệnh nhân COVID-19 tại Việt Nam. Tiếp theo là giải pháp cho hệ thống tìm kiếm thông tin: trình bày mô hình của hệ thống, chuyển đổi (RDF) và lưu trữ dữ liệu (Jena). Chương này cũng trình bày cách thức tải dữ liệu, chỉnh sửa dữ liệu và cách trực quan hóa dữ liệu Web ngữ nghĩa trên OpenStreetMap. Cuối cùng là ứng dụng minh họa cụ thể về hệ thống tìm kiếm thông tin người bệnh.

KẾT LUẬN

Luận văn đã nghiên cứu và trình bày những kiến thức căn bản về Web ngữ nghĩa như kiến thức về RDF, RDF Schema, ngôn ngữ truy vấn dữ liệu Web ngữ nghĩa SPARQL, môi trường lập trình Web ngữ nghĩa và trực quan hóa dữ liệu OpenStreetMap.

Qua đó luận văn đạt được một số kết quả như sau:

Về lý thuyết, luận văn đã đi sâu nghiên cứu được nhiều kiến thức về RDF và RDFS, từ đó hiểu được công nghệ Web ngữ nghĩa để có thể dựa vào đó triển khai các ứng dụng khác. Trình bày cụ thể phương pháp lập trình với Web ngữ nghĩa, cách chuyển đổi dữ liệu cho Web ngữ nghĩa. Luận văn cũng đã trình bày cách trích xuất và xử lý dữ liệu từ OpenStreetMap.

Về ứng dụng minh hoạ, với mục tiêu làm rõ thêm lý thuyết, luận văn ứng dụng xây dựng Web ngữ nghĩa với các công cụ hỗ trợ như Intellij và Spring MVC. Cụ thể là xây dựng được dữ liệu RDF cơ bản về thông tin bệnh nhân mắc COVID-19 tại Việt Nam, đồng thời trực quan hóa với OpenStreetMap dựa trên OSM_ID trích xuất từ dữ liệu, khai thác các tính năng truy xuất trên một tài liệu có mô tả ngữ nghĩa nhằm chia sẻ tài nguyên thông tin bệnh nhân và thực hiện tìm kiếm với những kết quả chính xác.

DANH MỤC CÁC TÀI LIỆU THAM KHẢO

1. Tiếng Việt

[1]. Hoàng Hữu Hạnh - Lê Mạnh Thạnh, *Giáo trình Web ngữ nghĩa*, NXB giáo dục.

2. Tiếng Anh

[2]. David Wood - Marsha Zaidman and Luke Ruth with Michael Hausenblas, *Linked data Structured Data on the Web.*

[3]. Glen Hart – Catherine Dolbear, *Linked Data A Geographic Perspective.*

[4]. Liyang Yu, *A Developer's Guide to the Web ngữ nghĩa*.

[5]. Leslie F.Sikos, PhD., *Mastering Structured Data On The Semantec Web.*

Frederik Ramm, Jochen Topf, Steve Chilton - UIT Cambridge, 1st edition, 2010 *OpenStreetMap Using and Enhancing the Free Map of the World.*

3. Website

[6]. http://www.OpenStreetMap.info/

[7]. http://wiki.OpenStreetMap.org/wiki/OSM_Semantic_Network

[8]. https://www.w3.org/2004/02/skos/

[9]. https://www.ncov.moh.gov.vn/

PHỤ LỤC

Mã nguồn lấy dữ liệu từ file Excel:

```
public class GenFromExcel {

    //URI cac tai nguyen
    static final String dbUri = "https://dbpedia.org/ontology/";
    static final String dcUri = "http://purl.org/dc/elements/1.1/";
    static final String foafUri = "http://xmlns.com/foaf/0.1/";
    public Model model;

    private GenFromExcel() {

        //Tao mo hinh RDF moi
        model = ModelFactory.createDefaultModel();

        //Them cac thuoc tinh va moi quan he
        Property dbPlaceId = model.createProperty(dbUri, "id");
        Property dbPlaceName = model.createProperty(dbUri, "name");
        Property dbPlaceLatitude = model.createProperty(dbUri,
"latitude");
        Property dbPlaceLongitude = model.createProperty(dbUri,
"longitude");

        Property dbSeeAlso = model.createProperty(dbUri, "seeAlso");
        Property dcDate = model.createProperty(dcUri, "date");
        Property dcPlace = model.createProperty(dcUri, "place");

        ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
        InputStream inputStream =
classLoader.getResourceAsStream("data/Data_BenhNhan.xlsx");
        InputStream inputStream2 =
classLoader.getResourceAsStream("data/Data_DiaDiem.xlsx");
        InputStream inputStream3 =
classLoader.getResourceAsStream("data/Data_LichSu.xlsx");
        List<BenhNhan> benhNhanList = new ArrayList<>();
        List<DiaDiem> diaDiemList = new ArrayList<>();
        List<LichSu> lichSuList = new ArrayList<>();
        Set<SuKien> suKienSet = new HashSet<>();
        try {
            OPCPackage pkg = OPCPackage.open(inputStream);
            XSSFWorkbook wb = new XSSFWorkbook(pkg);
            XSSFSheet sheet = wb.getSheetAt(0);
            for (Row row : sheet
            ) {
                BenhNhan benhNhan = new BenhNhan();
                benhNhan.setId(row.getCell(0).getStringCellValue());

benhNhan.setTuoi(String.valueOf(row.getCell(1).getNumericCellValue()));
                benhNhan.setDiaChi(row.getCell(2).getStringCellValue());

benhNhan.setTinhTrang(row.getCell(3).getStringCellValue());

benhNhan.setQuocTich(row.getCell(4).getStringCellValue());
```

```
                benhNhanList.add(benhNhan);
            }

            OPCPackage pkg2 = OPCPackage.open(inputStream2);
            XSSFWorkbook wb2 = new XSSFWorkbook(pkg2);
            XSSFSheet sheet2 = wb2.getSheetAt(0);
            for (Row row : sheet2
            ) {
                DiaDiem diaDiem = new DiaDiem();
                diaDiem.setId(row.getCell(0).getStringCellValue());
                diaDiem.setTen(row.getCell(1).getStringCellValue());

diaDiem.setOsmId(String.valueOf(row.getCell(2).getNumericCellValue()));

diaDiem.setLatitude(String.valueOf(row.getCell(3).getNumericCellValue()))
;

diaDiem.setLongitude(String.valueOf(row.getCell(4).getNumericCellValue())
);
                diaDiemList.add(diaDiem);
            }

            OPCPackage pkg3 = OPCPackage.open(inputStream3);
            XSSFWorkbook wb3 = new XSSFWorkbook(pkg3);
            XSSFSheet sheet3 = wb3.getSheetAt(0);
            for (Row row : sheet3
            ) {
                LichSu lichSu = new LichSu();
                lichSu.setThoiGian(row.getCell(0).getStringCellValue());

lichSu.setIdBenhNhan(row.getCell(1).getStringCellValue());
                lichSu.setIdDiaDiem(row.getCell(2).getStringCellValue());
                lichSuList.add(lichSu);

                SuKien suKien = new
SuKien(lichSu.getThoiGian(),lichSu.getIdDiaDiem());
                suKienSet.add(suKien);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }

        //Them cac tai nguyen
        List<Resource> benhNhanRsList = new ArrayList<>();
        for (BenhNhan benhNhan : benhNhanList
        ) {
            String name = benhNhan.getId();
            Resource benhNhanRs = model.createResource(foafUri + name);
            benhNhanRs.addProperty(FOAF.name, "Bệnh nhân " +
benhNhan.getId().replaceAll("BN", ""));
            benhNhanRs.addProperty(FOAF.openid, name);
            benhNhanRs.addProperty(FOAF.age, benhNhan.getTuoi());
            benhNhanRs.addProperty(FOAF.based_near,
benhNhan.getDiaChi());
            benhNhanRs.addProperty(FOAF.status, benhNhan.getTinhTrang());
            benhNhanRsList.add(benhNhanRs);
        }
```

```
        List<Resource> diaDiemRsList = new ArrayList<>();
        for (DiaDiem diaDiem : diaDiemList
        ) {
            String name = diaDiem.getId();
            Resource diaDiemRs = model.createResource(dbUri + name);
            diaDiemRs.addProperty(dbPlaceId, name);
            diaDiemRs.addProperty(dbPlaceName, diaDiem.getTen());
            diaDiemRs.addProperty(dbPlaceLatitude,
diaDiem.getLatitude());
            diaDiemRs.addProperty(dbPlaceLongitude,
diaDiem.getLongitude());
            diaDiemRsList.add(diaDiemRs);
        }

        List<SuKien> suKienLst = new ArrayList<>();
        suKienLst.addAll(suKienSet);
        Collections.reverse(suKienLst);
        List<Resource> suKienRsList = new ArrayList<>();
        for (int i = 0; i < suKienLst.size(); i++) {
            Resource resource = model.createResource(dcUri + i);
            resource.addProperty(dcDate,suKienLst.get(i).getNgay());
            resource.addProperty(dcPlace,suKienLst.get(i).getDiaDiem());
            suKienRsList.add(resource);
        }

        Set<LichSu> lichSuSet = new HashSet<>();
        for (LichSu lichSu : lichSuList
        ) {
            SuKien suKien = new
SuKien(lichSu.getThoiGian(),lichSu.getIdDiaDiem());
            for (SuKien suKien1: suKienLst
            ) {

if(suKien.getNgay().equalsIgnoreCase(suKien1.getNgay())&&suKien.getDiaDie
m().equalsIgnoreCase(suKien1.getDiaDiem())){
                    suKien= suKien1;
                }
            }

            Resource resource1 = null;
            for (Resource rs : benhNhanRsList
            ) {
                if
(rs.getProperty(FOAF.openid).getObject().toString().equalsIgnoreCase(lich
Su.getIdBenhNhan())) {
                    resource1 = rs;
                }
            }

            Resource resource2 = null;
            for (Resource rs : diaDiemRsList
            ) {
                if
(rs.getProperty(dbPlaceId).getObject().toString().equalsIgnoreCase(lichSu
.getIdDiaDiem())) {
                    resource2 = rs;
                }
            }
```

```
            lichSu.setDiaDiem(resource2);

            Resource resource3 = null;
            for (Resource rs : suKienRsList
            ) {
                if
(rs.getProperty(dcDate).getObject().toString().equalsIgnoreCase(lichSu.ge
tThoiGian()) &&
rs.getProperty(dcPlace).getObject().toString().equalsIgnoreCase(lichSu.ge
tIdDiaDiem())) {
                    resource3 = rs;
                }
            }

            lichSuSet.add(lichSu);

            lichSu.setSuKien(resource3);

            Statement statement = model.createStatement(resource1,
dbSeeAlso, resource3);
            model.add(statement);
        }

        List<LichSu> lichSuList1 = new ArrayList<>();
        lichSuList1.addAll(lichSuSet);

        for (LichSu lichSu: lichSuList1
             ) {
            Statement statement =
model.createStatement(lichSu.getSuKien(), dbSeeAlso,
lichSu.getDiaDiem());
            model.add(statement);
        }
        System.out.println("DONE");
    }
```

Mã nguồn lấy dữ liệu bằng SPARQL

```
Model model = ModelFactory.createDefaultModel();
InputStream in = FileManager.get().open(inputFileName);
model.read(in, "");
String queryString = "PREFIX foaf: <http://xmlns.com/foaf/0.1/> " +
        "PREFIX db: <https://dbpedia.org/ontology/> " +
        "PREFIX dc: <http://purl.org/dc/elements/1.1/> " +
        "SELECT ?ten ?ngay ?tenDiaDiem ?lat ?long" +
        "      WHERE {  " +
        "        ?x  foaf:name ?ten ; " +
        "            db:seeAlso ?sukien . " +
        "        OPTIONAL{?sukien dc:date ?ngay;" +
        "                  db:seeAlso ?diaDiem ." +
        "                 ?diaDiem db:name ?tenDiaDiem ." +
        "                 ?diaDiem db:latitude ?lat ." +
        "                 ?diaDiem db:longitude ?long ." +
        "                }." +
        "                FILTER regex(?ten, '"+ten+"') " +
        "     }";

Query query = QueryFactory.create(queryString);
```

```
QueryExecution qe = QueryExecutionFactory.create(query, model);
ResultSet results = qe.execSelect();
```

Mã nguồn chương trình tìm kiếm thông tin bệnh nhân COVID-19 tại Việt Nam:

```
package vn.com.fpt.foxpay.wallet.mock.web.mvc;

import com.fasterxml.jackson.databind.ObjectMapper;
import io.micrometer.core.annotation.Timed;
import org.apache.jena.query.*;
import org.apache.jena.rdf.model.Model;
import org.apache.jena.rdf.model.ModelFactory;
import org.apache.jena.util.FileManager;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.*;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;
import org.springframework.web.servlet.ModelAndView;

import java.io.IOException;
import java.io.InputStream;
import java.util.*;

@Controller
public class MerchantMockController {

    private final Logger log =
LoggerFactory.getLogger(MerchantMockController.class);

    @RequestMapping(value = { "", "" }, method = RequestMethod.GET)
    @ResponseBody
    @Timed
    public ModelAndView checkout(org.springframework.ui.Model model2,
@RequestParam(required = false) String ten) {

        if(ten == null){
            ten = "Bệnh nhân 1";
        }

        String resourceUrl
                = "https://corona.lmao.ninja/v2/countries/vn";
        RestTemplate restTemplate = new RestTemplate();
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
        headers.add("user-agent", "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99
Safari/537.36");
        HttpEntity<String> entity = new HttpEntity<String>("parameters",
headers);

        ResponseEntity<String> response =
```

```
restTemplate.exchange(resourceUrl, HttpMethod.GET,entity,String.class);
        ObjectMapper mapper = new ObjectMapper();
        try {
            ResStatus resStatus =
mapper.readValue(response.getBody(),ResStatus.class);
            model2.addAttribute("resStatus", resStatus);
        } catch (IOException e) {
            e.printStackTrace();
        }

        final String inputFileName = "D:\\test3.rdf";
        Model model = ModelFactory.createDefaultModel();
        InputStream in = FileManager.get().open(inputFileName);
        model.read(in, "");
        String queryString = "PREFIX foaf: <http://xmlns.com/foaf/0.1/> "
+
                "PREFIX db: <https://dbpedia.org/ontology/> " +
                "PREFIX dc: <http://purl.org/dc/elements/1.1/> " +
                "SELECT ?ten ?ngay ?tenDiaDiem ?lat ?long" +
                "      WHERE {  " +
                "        ?x  foaf:name ?ten ; " +
                "            db:seeAlso ?sukien . " +
                "        OPTIONAL{?sukien dc:date ?ngay;" +
                "                  db:seeAlso ?diaDiem ." +
                "                 ?diaDiem db:name ?tenDiaDiem ." +
                "                 ?diaDiem db:latitude ?lat ." +
                "                 ?diaDiem db:longitude ?long ." +
                "                }." +
                "                FILTER regex(?ten, '"+ten+"') " +
                "     }";

        Query query = QueryFactory.create(queryString);
        QueryExecution qe = QueryExecutionFactory.create(query, model);
        ResultSet results = qe.execSelect();

        List<String> benhNhatList = getlistBenhNhan();
        List<ThongTin> thongtinList = new ArrayList<>();
        List<LichTrinh> lichTrinhList = new ArrayList<>();
        model2.addAttribute("benhNhatList", benhNhatList);
        model2.addAttribute("benhNhan", ten);

        List kq=results.getResultVars();
        while(results.hasNext())
        {
            QuerySolution qs=results.nextSolution();
            String val0=qs.get(kq.get(0).toString()).toString();
            String val1=qs.get(kq.get(1).toString()).toString();
            String val2=qs.get(kq.get(2).toString()).toString();
            String val3=qs.get(kq.get(3).toString()).toString();
            String val4=qs.get(kq.get(4).toString()).toString();
            ThongTin thongTin = new ThongTin();
            thongTin.setBenhNhan(val0);
            thongTin.setNgay(val1);
            thongTin.setDiaDiem(val2);
            thongTin.setLat(val3);
            thongTin.setLon(val4);
            thongtinList.add(thongTin);
        }
```

```
        qe.close();
        Set<String> ngaySet = new HashSet<>();
        for (ThongTin thongtin: thongtinList
             ) {
            ngaySet.add(thongtin.getNgay());
        }

        List<String> ngayLst = new ArrayList<>();
        List<DiaDiem> diaDiemList2 = new ArrayList<>();
        ngayLst.addAll(ngaySet);

        for (String s: ngayLst
             ) {
            LichTrinh lichTrinh = new LichTrinh();
            lichTrinh.setNgay(s);
            List<DiaDiem> diaDiemList = new ArrayList<>();
            for (ThongTin thongtin: thongtinList
            ) {
                if(thongtin.getNgay().equals(s)){
                    DiaDiem diaDiem = new DiaDiem();
                    diaDiem.setTen(thongtin.getDiaDiem());
                    diaDiem.setLatitude(thongtin.getLat());
                    diaDiem.setLongitude(thongtin.getLon());
                    diaDiemList.add(diaDiem);
                    diaDiemList2.add(diaDiem);

                }
            }
            lichTrinh.setDiaDiem(diaDiemList);
            lichTrinhList.add(lichTrinh);
        }


        model2.addAttribute("lichTrinh", lichTrinhList);
        model2.addAttribute("diaDiem", diaDiemList2);

        return new ModelAndView("merchant-mock-checkout");
    }

    @RequestMapping(value = "/danh-sach", method = RequestMethod.GET)
    @ResponseBody
    @Timed
    public ModelAndView danhSach(org.springframework.ui.Model model2,
@RequestParam(required = false) String ten) {

        String resourceUrl
                = "https://corona.lmao.ninja/v2/countries/vn";
        RestTemplate restTemplate = new RestTemplate();
        HttpHeaders headers = new HttpHeaders();
        headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
        headers.add("user-agent", "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.99
Safari/537.36");
        HttpEntity<String> entity = new HttpEntity<String>("parameters",
headers);

        ResponseEntity<String> response =
restTemplate.exchange(resourceUrl, HttpMethod.GET,entity,String.class);
```

```
        ObjectMapper mapper = new ObjectMapper();
        try {
            ResStatus resStatus =
mapper.readValue(response.getBody(),ResStatus.class);
            model2.addAttribute("resStatus", resStatus);
        } catch (IOException e) {
            e.printStackTrace();
        }

        return new ModelAndView("danh-sach");
    }

    public List<String> getlistBenhNhan(){
        final String inputFileName = "D:\\test3.rdf";
        Model model = ModelFactory.createDefaultModel();
        InputStream in = FileManager.get().open(inputFileName);
        model.read(in, "");
        String queryString = "PREFIX foaf: <http://xmlns.com/foaf/0.1/> "
+
                "PREFIX db: <https://dbpedia.org/ontology/> " +
                "PREFIX dc: <http://purl.org/dc/elements/1.1/> " +
                "SELECT ?ten " +
                "      WHERE {   " +
                "        ?x  foaf:name ?ten ; " +
//                "            db:seeAlso ?sukien . " +
//                "        OPTIONAL{?sukien dc:date ?ngay;" +
//                                 db:seeAlso ?diaDiem ." +
//                                ?diaDiem db:name ?tenDiaDiem ." +
//                                ?diaDiem db:latitude ?lat ." +
//                                ?diaDiem db:longitude ?long ." +
//                               }." +
//                               FILTER regex(?ten, 'Bệnh nhân 1') " +
                "     }";

        Query query = QueryFactory.create(queryString);
        QueryExecution qe = QueryExecutionFactory.create(query, model);
        ResultSet results = qe.execSelect();
        List kq=results.getResultVars();
        List<String> lst = new ArrayList<>();
        while(results.hasNext())
        {
            QuerySolution qs=results.nextSolution();
            String val0=qs.get(kq.get(0).toString()).toString();
            lst.add(val0);
//            String val5=qs.get(kq.get(5).toString()).toString();
//            out.println("<br><a
href=?txtDiaDiem="+val0+">"+val1+"</a>");
        }
        Collections.sort(lst);
        qe.close();
        return lst;
    }

}
```

BẢN CAM ĐOAN

Tôi cam đoan đã thực hiện việc kiểm tra mức độ tương đồng nội dung luận văn qua phần mềm DoIT một cách trung thực và đạt kết quả mức tương đồng 9% toàn bộ nội dung luận văn. Bản luận văn kiểm tra qua phần mềm là bản cứng luận văn đã nộp để bảo vệ trước hội đồng. Nếu sai tôi xin chịu các hình thức kỷ luật theo quy định hiện hành của Học viện Công nghệ Bưu chính Viễn thông.

Hà Nội, ngày 06 tháng 03 năm 2021

Học viên thực hiện luận văn kỹ thuật

Phan Hồng Dương

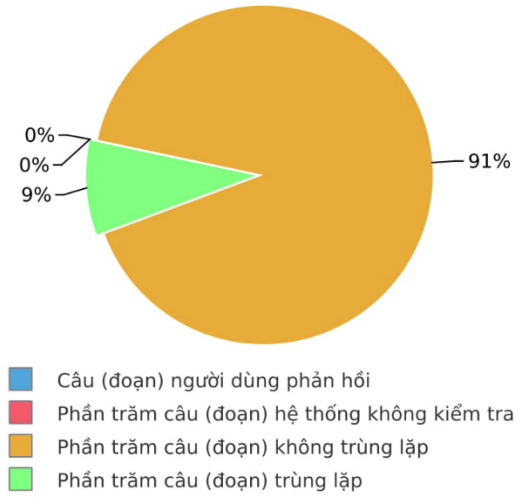
Hệ thống hỗ trợ nâng cao chất lượng tài liệu

KẾT QUẢ KIỂM TRA TRÙNG LẶP TÀI LIỆU

THÔNG TIN TÀI LIỆU

Tác giả	duong
Tên tài liệu	RDF-PhanHongDuong_v1
Thời gian kiểm tra	21-04-2021, 10:09:26
Thời gian tạo báo cáo	21-04-2021, 10:11:23

KẾT QUẢ KIỂM TRA TRÙNG LẶP



Điểm	9
Nguồn trùng lặp tiêu biểu	[text.123doc.org, vi.wikipedia.org, 123doc.org]

(*) Kết quả trùng lặp phụ thuộc vào dữ liệu hệ thống tại thời điểm kiểm tra

Học viên thực hiện luận văn | Người hướng dẫn luận văn

Phan Hồng Dương | PGS. TS. Hoàng Hữu Hạnh