

TRƯỜNG ĐẠI HỌC TIỀN GIANG

**KHOA CÔNG NGHỆ THÔNG TIN**

-----🙞🙜🕮🙞🙜-----

****

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI**

**XÂY DỰNG PHÂN HỆ KHÁM CHỮA BỆNH TRONG ỨNG DỤNG QUẢN LÝ HỆ THỐNG   
BÁC SĨ GIA ĐÌNH**

Người hướng dẫn: **Nguyễn Thị Phương Linh**

**Nguyễn Thái Duy**

Sinh viên thực hiện:

Đặng Duy An 016101002

Võ Thị Tuyết Hương 016101006

**Khóa 2016 - 2020**

**LỜI NÓI ĐẦU**

Bác sĩ gia đình là bác sĩ đa khoa thực hành, chăm sóc toàn diện và liên tục cho người bệnh, có mối quan hệ lâu dài và bền vững với người bệnh , là những thầy thuốc gắn với dân và gần dân nhất. Bác sĩ gia đình là bác sĩ hướng về gia đình, biết rõ từng người bệnh trong hoàn cảnh và gia đình của họ, xem xét vấn đề sức khỏe của người bệnh trong hoàn cảnh của cộng đồng và lối sống của người đó trong cộng đồng.

Trong đó khâu Khám chữa bệnh và các phân hệ liên quan như Cây phả hệ, Hồ sơ bệnh án bệnh nhân trong Bác sĩ gia đình là cực kì quan trọng trong bước đầu quản lý, triển khai quá trình khám chữa bệnh.

Hoạt động bác sĩ gia đình ở nước ta hiện nay là mô hình mới, chưa được quan tâm đầu tư tương xứng, chưa có chức danh bác sĩ gia đình ở các cơ sở y tế, hoạt động còn tản mạn, nhiều hạn chế, bất cập, chưa có đầy đủ cơ sở pháp lý và hiệu quả chưa cao. Hiện nay, mô hình bệnh tật ở nước ta là mô hình bệnh tật kép, các bệnh lây nhiễm, suy dinh dưỡng vẫn ở mức khá cao, trong khi nhóm các bệnh không lây nhiễm và tai nạn thương tích tăng nhanh dẫn đến nhu cầu khám, chữa bệnh của người dân ngày càng tăng; việc sàng lọc, theo dõi, quản lý, điều trị bệnh mạn tính tại cộng đồng là hết sức cần thiết, đòi hỏi việc nâng cao năng lực y tế tuyến cơ sở ngày càng trở nên cấp bách.

Việc xây dựng một hệ thống công nghệ thông tin dành cho bác sĩ gia đình ở Việt Nam càng trở nên cần thiết hơn.

Để hỗ trợ cho quy trình quản lý các thông tin về sức khỏe, tiền sử bệnh, hồ sơ bệnh án, … của các bệnh nhân trong cùng một gia đình có cùng huyết thống theo tiêu chuẩn của bác sĩ gia đình và để phối hợp với các bác sĩ chuyên khoa ở tuyến trên tốt hơn, chúng tôi đã chọn đề tài “XÂY DỰNG PHÂN HỆ KHÁM CHỮA BỆNH TRONG ỨNG DỤNG QUẢN LÝ HỆ THỐNG BÁC SĨ GIA ĐÌNH”.

**LỜI CẢM ƠN**

🙞🙞🕮🙜🙜

Đầu tiên chúng tôi xin chân thành cảm ơn Ban Giám Hiệu cùng quý Thầy Cô Trường Đại học Tiền Giang đặc biệt là toàn thể quý Thầy Cô trong khoa Công nghệ thông tin đã tận tình giảng dạy, trang bị cho chúng tôi những kiến thức quý báu trong quá trình học tập tại Trường để chúng tôi có thể hoàn thành tốt khóa luận tốt nghiệp này.

Tiếp theo, chúng tôi xin chân thành cảm ơn sự giúp đỡ tận tình của các Anh/Chị trong công ty VNPT đã hỗ trợ, giúp cho chúng tôi tiếp cận với công nghệ mới, cách tiếp cận và nắm bắt các công việc cần thiết cho khóa luận.

Để hoàn thành tốt khóa luận tốt nghiệp này, chúng tôi xin tỏ lòng biết ơn sâu sắc, chân thành đến thầy Dương Văn Hiếu – người đã tạo điều kiện để chúng tôi có thể làm khóa luận tốt nghiệp theo đề tài của công ty VNPT và cô   
Nguyễn Thị Phương Linh – người đã luôn theo sát và tận tình hướng dẫn chúng tôi làm khóa luận. Bên cạnh đó là sự giúp đỡ tận tình của bạn bè cùng lớp và các anh chị khóa trên đã góp phần tạo thêm động lực cho chúng tôi vượt qua những khó khăn trong qua trình thực hiện đề tài khóa luận tốt nghiệp này. Qua đó, chúng tôi đã đạt được nhiều tiến bộ về kiến thức cũng như những kĩ năng làm việc bổ ích.

Mặc dù đã rất cố gắng trong quá trình làm khóa luận tốt nghiệp, nhưng chắc chắn không thể tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được sự thông cảm và góp ý tận tình của quý Thầy Cô và các bạn.

Cuối cùng chúng tôi kính chúc quý Thầy/Cô và quý Anh/Chị trong công ty VNPT dồi dào sức khỏe và thành công trong công việc.

Chúng tôi xin chân thành cảm ơn!

Tiền Giang, ngày 31 tháng 7 năm 2020

Sinh viên thực hiện

Đặng Duy An

Võ Thị Tuyết Hương

**ĐỀ CƯƠNG CHI TIẾT KHÓA LUẬN TỐT NGHIỆP**

**Tên đề tài**: Xây dựng phân hệ khám chữa bệnh trong ứng dụng quản lý hệ thống bác sĩ gia đình

**Người hướng dẫn:** Nguyễn Thị Phương Linh

Nguyễn Thái Duy

**Thời gian thực hiện:** 04/05/2020 – 15/07/2020.

**Sinh viên thực hiện:** Đặng Duy An MSSV: 016101002

Võ Thị Tuyết Hương MSSV: 016101006

**Loại đề tài:** Xây dựng ứng dụng

**Nội dung đề tài:**

* **Mục tiêu đề tài:**
  + Tìm hiểu về Microservices.
  + Tìm hiểu về ReactJS, Typescript, Redux, JasperReports, Spring Framework,…và các ngôn ngữ liên quan.
  + Tìm hiểu quy trình quản lý bệnh nhân theo cây phả hệ và các tài liệu thông tin bác sĩ gia đình.
  + Rèn luyện kỹ năng phân chia thời gian, phân tích, giải quyết vấn đề, làm việc nhóm và làm việc thực tế một cách hiệu quả.
  + Ứng dụng những kiến thức đã học và kiến thức đã tìm hiểu xây dựng thành công hai phân hệ Cây phả hệ và Hồ sơ bệnh án Bác sĩ gia đình cho hệ thống Bác sĩ gia đình
* **Kết quả cần đạt được:**
  + Công dụng: Hệ thống quản lý các thông tin cần lưu trữ của bệnh nhân cho hệ thống bác sĩ gia đình.
  + Chức năng:
    - Đăng nhập dựa trên mô hình Microservices.
    - Quản lý bệnh nhân theo hệ thống cây phả hệ bao gồm hiển thị thế hệ ba đời khi tìm kiếm một bệnh nhân, thêm mới, cập nhật thông tin bệnh nhân, thống kê theo giới tính, tình trạng hôn nhân trong một khu vực chuyển giao diện sang hồ sơ bệnh án bác sĩ gia đình.
    - Quản lý các thông tin của bệnh nhân như tài chính, bệnh sử, tiêm ngừa, tai nạn thương tích, kế hoạch hóa gia đình, kinh tế, thói quen, nghề nghiệp,…
  + Đặc điểm chính:
    - Hệ thống tích hợp hai phân hệ là hệ thống cây phả hệ và hệ thống hồ sơ bệnh án bác sĩ gia đình.
    - Hệ thống hoạt động quản lý thông tin bệnh nhân, quản lý thông tin gia đình của bệnh nhân thông qua cây phả hệ và quản lý, lưu trữ thông tin khám chữa bệnh của bệnh nhân trong hồ sơ bệnh án bác sĩ gia đình một cách hiệu quả, nhanh chóng và chính xác.
* **Nhiệm vụ đề tài:**
  + Nghiên cứu thực tế: Ứng dụng công nghệ Microservices để thiết kế hệ thống, kết hợp sử dụng ReactJS, Redux, Typescript, JasperReports, Neo4J và MongoDB Datatabase để giúp người lập trình có thêm công cụ để lập trình hệ thống một cách hiệu quả hơn.
  + Xác định phạm vi đề tài: Đề tài để phục vụ quản lý bệnh nhân trong một gia đình ở phạm vi phường xã.
  + Nghiên cứu tài liệu chuyên môn: Trong quá trình thực hiện đề tài tham khảo các tài liệu chuyên môn về Bác sĩ gia đình do các Thạc sĩ, Bác sĩ biên soạn và đề án Bác sĩ gia đình của Bộ Y Tế.
* **Phân tích, thiết kế, cài đặt (hiện thực), thử nghiệm:**
  + Tham khảo cơ sở dữ liệu của phân hệ Khám chữa bệnh.
  + Hiểu về Graph Database(Neo4J) sau đó áp dụng để thiết kế kiến trúc cơ bản cho cây phả hệ (mối quan hệ Relationship giữa cặp vợ-chồng, anh-em qua ba thế hệ và sự thay đổi quan hệ của họ khi gặp sự cố trong gia đình như ly dị, mất do bệnh, do tuổi già,…) và xác định các thông tin cần lưu trên Neo4j.
  + Thiết lập được một hệ thống Microservices bằng Jhipster, cài đặt Neo4J, MongoDB và JasperReport.
  + Phân tích các trường hợp cần xây dựng câu truy vấn truy xuất chính xác các thế hệ của gia đình bệnh nhân.
  + Thiết kế giao diện cho các chức năng cần thiết.
  + Thiết kế cơ sở dữ liệu trên MongoDB và tạo các báo cáo trên JasperReport.
  + Kết nối giao diện với hệ thống cây phả hệ Neo4J
  + Chạy thử và sửa lỗi.
  + Báo cáo và rút kinh nghiệm.
* **Yêu cầu khoa học công nghệ:**
  + Hệ điều hành: Windows
  + Trình soạn thảo mã nguồn: Visual Studio Code, IntelliJ IDEA 2019
  + Trình quản lý CSDL: Neo4J, MongoDB
  + Trình duyệt web: Google Chrome, IE,....

**Kế hoạch thực hiện**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Các nội dung, công việc chủ yếu cần thực hiện** | **Từ ngày** | **Đến ngày** |
| 1 | Phân tích đề tài | 20/04/2020 | 1/05/2020 |
| 2 | Thiết kế CSDL | 25/04/2020 | 25/05/2020 |
| 3 | Tìm hiểu về ReactJS, Redux, Typescript, JasperReports.... | 20/04/2020 | 01/06/2020 |
| 4 | Lập trình BackEnd | 31/05/2020 | 10/07/2020 |
| 5 | Lập trình FrontEnd | 05/06/2020 | 20/07/2020 |
| 6 | Thiết kế các báo cáo | 10/05/2020 | 15/07/2020 |
| 7 | Chạy thử và sửa lỗi | 15/06/2020 | 21/07/2020 |
| 8 | Viết báo cáo và sửa lỗi đề tài | 05/05/2020 | 15/07/2020 |

**Xác nhận của người hướng dẫn**  *Ngày……tháng……năm 2020*

Sinh viên thực hiện

Đặng Duy An

Võ Thị Tuyết Hương

**MỤC LỤC**

[Chương 1. TỔNG QUAN 1](#_Toc46867318)

[1.1. Giới thiệu đề tài 1](#_Toc46867319)

[1.1.2. Lý do chọn đề tài 3](#_Toc46867320)

[1.1.3. Mục tiêu đề tài 3](#_Toc46867321)

[1.1.4. Nhiệm vụ 4](#_Toc46867322)

[1.1.5. Phạm vi nghiên cứu của đề tài 6](#_Toc46867323)

[1.2. Các vấn đề tập trung giải quyết 6](#_Toc46867324)

[Chương 2: CƠ SỞ LÝ THUYẾT VÀ THỰC TIỄN 7](#_Toc46867325)

[2.1. Cơ sở lý thuyết 7](#_Toc46867326)

[2.1.1. Giả thuyết nghiên cứu 7](#_Toc46867327)

[2.1.2. Phương pháp nghiên cứu 7](#_Toc46867328)

[2.1.3. Công nghệ sử dụng 8](#_Toc46867329)

[2.1.4. Cơ sở lý thuyết 8](#_Toc46867330)

[2.2. Cơ sở thực tiễn và ứng dụng của khóa luận 18](#_Toc46867331)

[2.2.1. Bối cảnh thực tiễn 18](#_Toc46867332)

[2.2.2. Ứng dụng 19](#_Toc46867333)

[Chương 3: PHÂN TÍCH, THIẾT KẾ, CÀI ĐẶT GIẢI PHÁP 20](#_Toc46867334)

[3.1. Phân tích yêu cầu 20](#_Toc46867335)

[3.1.1. Phân tích yêu cầu chức năng 20](#_Toc46867336)

[3.1.2. Phân tích yêu cầu phi chức năng 21](#_Toc46867337)

[3.1.3. Phân tích quy trình 21](#_Toc46867338)

[3.2. Phân tích thiết kế hệ thống 24](#_Toc46867339)

[3.2.1. Hệ thống Cây phả hệ 24](#_Toc46867340)

[3.2.2. Hệ thống Hồ sơ bệnh án bác sĩ gia đình 33](#_Toc46867341)

[3.3. Cài đặt giải pháp 47](#_Toc46867342)

[3.3.1. Xây dựng cấu trúc Microservice cho hệ thống Cây phả hệ và hệ thống Hồ sơ bác sĩ gia đình 48](#_Toc46867343)

[3.3.2. Cài đặt môi trường cho Graph Database (Neo4J) và NoSQL Database (MongoDB) 56](#_Toc46867344)

[3.3.3. Vấn đề và giải pháp 62](#_Toc46867345)

[3.3.4. Cài đặt hệ thống 66](#_Toc46867346)

[3.3.5 Một số hình ảnh giao diện hệ thống 74](#_Toc46867347)

[Chương 4: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN 83](#_Toc46867348)

[4.1. Kết quả đạt được 83](#_Toc46867349)

[4.2. Hạn chế 83](#_Toc46867350)

[4.3. Hướng phát triển 83](#_Toc46867351)

[TÀI LIỆU THAM KHẢO 84](#_Toc46867352)

**DANH MỤC HÌNH**

[Hình 1. Kiến thúc hệ thống bác sĩ gia đình 1](#_Toc46951609)

[Hình 2. Mô hình hoạt động của Redux [3] 13](#_Toc46951610)

[Hình 3. Minh họa tạo dự án Jhipster 18](#_Toc46951611)

[Hình 4. Sơ đồ Use case tổng quát (cán bộ phụ trách cây phả hệ) 24](#_Toc46951612)

[Hình 5. Sơ đồ Use case tổng quát (cán bộ tiếp nhận và bác sĩ) 24](#_Toc46951613)

[Hình 6. Sơ đồ lớp (Graph Database) 30](#_Toc46951614)

[Hình 7. Sơ đồ use case tổng quát 33](#_Toc46951615)

[Hình 8. Sơ đồ cơ sở dữ liệu hồ sơ bệnh án bác sĩ gia đình 47](#_Toc46951616)

[Hình 9. Xây dựng cấu trúc cho hệ thống 49](#_Toc46951617)

[Hình 10. Xây dựng cấu trúc cho hệ thống 49](#_Toc46951618)

[Hình 11. Xây dựng cấu trúc cho hệ thống 50](#_Toc46951619)

[Hình 12. Xây dựng cấu trúc cho hệ thống 50](#_Toc46951620)

[Hình 13. Xây dựng dịch vụ gateway 51](#_Toc46951621)

[Hình 14. Cấu trúc thư mục dịch vụ Gateway 52](#_Toc46951622)

[Hình 15. Kết quả sau khi khởi chạy Service 52](#_Toc46951623)

[Hình 16. Trang chủ của Service Gateway 53](#_Toc46951624)

[Hình 17. Xây dựng dịch vụ phả hệ và hồ sơ bệnh án bác sĩ gia đình 53](#_Toc46951625)

[Hình 17. Cài đặt Service Phả hệ 54](#_Toc46951626)

[Hình 18. Cấu trúc thư mục Service Phả hệ 55](#_Toc46951627)

[Hình 19. Cấu trúc thư mục Service Hồ sơ bệnh án 55](#_Toc46951628)

[Hình 20. Service phả hệ sau khi khởi chạy 56](#_Toc46951629)

[Hình 21. Service hồ sơ bệnh án sau khi khởi chạy 56](#_Toc46951630)

[Hình 22. Dowload Neo4J 56](#_Toc46951631)

[Hình 23. Cài đặt nơi lưu trữ dữ liệu 57](#_Toc46951632)

[Hình 24. Giao diện của Neo4J 57](#_Toc46951633)

[Hình 25. Giao diện tạo Graph và Plugin 58](#_Toc46951634)

[Hình 26. Giao diện quản lý cho một Graph 58](#_Toc46951635)

[Hình 27. Giao diện viết câu truy vấn cho Graph Database (Neo4J) 59](#_Toc46951636)

[Hình 28. Dowload Robo 3T 59](#_Toc46951637)

[Hình 29. Giao diện Robo 3T 60](#_Toc46951638)

[Hình 30. Tạo kết nối với MongoDB 60](#_Toc46951639)

[Hình 31. Kiểm tra kết nối với MongoDB 61](#_Toc46951640)

[Hình 32. Minh họa kết nối với MongoDB 61](#_Toc46951641)

[Hình 33. Giao diện Robo 3T sau khi đã kết nối thành công 62](#_Toc46951642)

[Hình 34. Ví dụ cho cách sử dụng hàm Collect 64](#_Toc46951643)

[Hình 35. Ví dụ cho câu truy vấn một vế trong Neo4J 64](#_Toc46951644)

[Hình 36. Ví dụ khi tách câu truy vấn trong Neo4J 64](#_Toc46951645)

[Hình 37. Câu truy vấn minh họa nếu bệnh nhân là người lớn tuổi nhất 65](#_Toc46951646)

[Hình 38. Minh họa cho câu truy vấn nếu bệnh nhân là người lớn tuổi nhất 65](#_Toc46951648)

[Hình 39. Câu truy vấn minh họa nếu bệnh nhân là người nhỏ tuổi nhất 66](#_Toc46951649)

[Hình 40. Ảnh minh họa cho câu truy vấn nếu bệnh nhân là người nhỏ tuổi nhất 66](#_Toc46951650)

[Hình 41. Câu truy vấn minh họa nếu bệnh nhân thế hệ trung niên trong gia đình 66](#_Toc46951651)

[Hình 42. Ảnh minh họa cho câu truy vấn nếu bệnh nhân là thế hệ trung niên trong gia đình 67](#_Toc46951652)

[Hình 43. Câu truy vấn minh họa lấy hai thế hệ (cha) 67](#_Toc46951653)

[Hình 44. Ảnh minh họa cho câu truy vấn lấy hai thế hệ (cha) 68](#_Toc46951654)

[Hình 45. Câu truy vấn minh họa lấy hai thế hệ (con) 68](#_Toc46951655)

[Hình 46. Ảnh minh họa câu truy vấn lấy hai thế hệ (con) 68](#_Toc46951656)

[Hình 47. Minh họa các trường hợp sử dụng câu truy vấn 69](#_Toc46951657)

[Hình 48. Thêm service vào IntteliJ IDEA 2018 71](#_Toc46951658)

[Hình 49. Thêm Service vào IntelliJ 71](#_Toc46951659)

[Hình 50. Thêm Service vào IntelliJ 72](#_Toc46951660)

[Hình 51. Cấu trúc thư mục của Project sau khi Add 72](#_Toc46951661)

[Hình 52. Khởi chạy các Service 73](#_Toc46951662)

[Hình 53. Cây thư mục Phả hệ sau khi cài đặt 73](#_Toc46951663)

[Hình 54. .Minh họa lấy danh sách cây phả hệ 74](#_Toc46951664)

[Hình 55. Service (dto) 74](#_Toc46951665)

[Hình 56. Reponsitory 75](#_Toc46951666)

[Hình 57. Service (iml) 75](#_Toc46951667)

[Hình 58. Service 76](#_Toc46951668)

[Hình 59. Resource 76](#_Toc46951669)

[Hình 60. Minh họa in thông tin khám bệnh của bệnh nhân theo mã bệnh nhân 77](#_Toc46951670)

[Hình 61. Giao diện của cây phả hệ 1 77](#_Toc46951671)

[Hình 62. Giao diện của Cây phả hệ 2 78](#_Toc46951672)

[Hình 63. Giao diện Cây phả hệ (Khi lấy nhánh phả hệ 3 đời của 1 bệnh nhân) 78](#_Toc46951673)

[Hình 64. Giao diện Cây phả hệ (Khi nhấn vào một node bệnh nhân trong đồ thị) 79](#_Toc46951674)

[Hình 65. Giao diện service hồ sơ bệnh án bác sĩ gia đình 79](#_Toc46951675)

[Hình 66. Giao diện Quản lý thông tin bệnh nhân 80](#_Toc46951676)

[Hình 67. Giao diện thêm thông tin bệnh nhân 80](#_Toc46951677)

[Hình 68. Giao diện cập nhật thông tin bệnh nhân 81](#_Toc46951678)

[Hình 69. Giao diện xóa thông tin bệnh nhân 81](#_Toc46951679)

[Hình 70. Giao diện Quản lý vấn đề hiện tại 82](#_Toc46951680)

[Hình 71. Giao diện thêm vấn đề hiện tại 82](#_Toc46951681)

[Hình 72 .Giao diện cập nhật vấn đề hiện tại 82](#_Toc46951682)

[Hình 73. Giao diện xóa vấn đề hiện tại 83](#_Toc46951683)

[Hình 74. Giao diện vấn đề hiện tại của bệnh nhân 83](#_Toc46951684)

[Hình 75. Giao diện thêm vấn đề hiện tại của bệnh nhân 84](#_Toc46951685)

[Hình 76. Giao diện cập nhật vấn đề hiện tại của bệnh nhân 84](#_Toc46951686)

[Hình 77. Giao diện xóa vấn đề hiện tại của bệnh nhân 85](#_Toc46951687)

[Hình 78. Giao diện in hồ sơ bệnh án bác sĩ gia đình 85](#_Toc46951688)

[Hình 79. Hồ sơ bệnh án sau khi được in từ giao diện hồ sơ bệnh án bác sĩ gia đình 85](#_Toc46951689)

**DANH MỤC BẢNG**

[Bảng 1. Hệ thống cây phả hệ 5](#_Toc46951355)

[Bảng 2. Hồ sơ bệnh án bác sĩ gia đình: 5](#_Toc46951356)

[Bảng 3. Quy trình thêm bệnh nhân vào service Phả hệ 22](#_Toc46951357)

[Bảng 4. Quy trình tạo quan hệ, xóa và sửa quan hệ của bệnh nhân trong cây phả hệ. 23](#_Toc46951358)

[Bảng 5. Bệnh nhân 36](#_Toc46951359)

[Bảng 6. Bác sĩ 37](#_Toc46951360)

[Bảng 7. Thông tin khám bệnh 37](#_Toc46951361)

[Bảng 8. Lịch tiêm ngừa 38](#_Toc46951362)

[Bảng 9. Toa thuốc 39](#_Toc46951363)

[Bảng 10. Thông tin 39](#_Toc46951364)

[Bảng 11. Kế hoạch 40](#_Toc46951365)

[Bảng 12. Vấn đề 40](#_Toc46951366)

[Bảng 13. Khám hệ cơ quan 40](#_Toc46951367)

[Bảng 14. Vấn đề bệnh nhân 40](#_Toc46951368)

[Bảng 15. Người nhà bệnh nhân 41](#_Toc46951369)

[Bảng 16. Vấn đề tầm soát 41](#_Toc46951370)

[Bảng 17. Tiền sử gia đình 41](#_Toc46951371)

[Bảng 18. Vấn đề tầm soát bệnh nhân 42](#_Toc46951372)

[Bảng 19. Khám lâm sàng 42](#_Toc46951373)

[Bảng 20. Tâm lý bệnh nhân 43](#_Toc46951374)

[Bảng 21. Nghề nghiệp 43](#_Toc46951375)

[Bảng 22. Thói quen 44](#_Toc46951376)

[Bảng 23. Tài liệu 44](#_Toc46951377)

[Bảng 24. Tai nạn 44](#_Toc46951378)

[Bảng 25. Danh mục vấn đề khác 45](#_Toc46951379)

[Bảng 26. Tiền sử bệnh nhân 45](#_Toc46951380)

[Bảng 27. Danh mục vấn đề khác của bệnh nhân 46](#_Toc46951381)

[Bảng 28.Kiểu dữ liệu 46](#_Toc46951382)

**TÓM TẮT KHÓA LUẬN**

**1. Mục tiêu**

Tìm hiểu lý thuyết: Nắm được cơ chế hoạt động của Microservices, hiểu được kiến thức nền tảng của lập trình sử dụng ReactJS, Spring Framework, TypeScript, Graph Database (Neo4J), NoSQL Database (MongoDB), JasperReport…Hiểu được cấu trúc của một cây phả hệ theo tiêu chuẩn Graph Database, hiểu được khái quát được tài liệu thông tin bác sĩ gia đình (mẫu in hồ sơ bác sĩ gia đình).

Ứng dụng thực nghiệm: Áp dụng tất cả các kiến thức lý thuyết đã tìm hiểu và đã được học để xây dựng thành công ứng dụng cho hệ thống bác sĩ gia đình.

**2. Nhiệm vụ**

Tìm hiểu lý thuyết: Tìm hiểu nguyên lý hoạt động của Microservices (Jhipster), nắm được các kiến thức, định nghĩa cốt lõi của ReactJS, TypeScript, SpringFramework, Jasper Report, Graph Database (Neo4J), NoSQL Database (MongoDB)…

Ứng dụng thực nghiệm: Đặt ra và phân tích yêu cầu, liệt kê các chức năng cần thiết, thiết kế cơ sở dữ liệu, xây dựng cấu trúc ứng dụng, cấu hình ứng dụng, thiết kế giao diện, thiết kế các báo cáo, cài đặt giải thuật.

**3. Phương pháp giải quyết**

Tìm hiểu lý thuyết: Phương pháp nghiên cứu lý thuyết, phát hiện và giải quyết vấn đề.

Ứng dụng thực nghiệm: Phương pháp thực nghiệm khoa học, phân tích và tổng kết tiếp thu kinh nghiệm.

**4. Kết quả đạt được**

Về lý thuyết: Hiểu được cách hoạt động của Microservices, hiểu được các kiến thức nền tảng để sử dụng ReactJS, TypeScript, SpringFramework, Jasper Report, …

Nắm được quy trình quản lý bệnh nhân theo cây phả hệ, tài liệu thông tin bác sĩ gia đình.

Ứng dụng thực nghiệm: Hoàn thiện phân hệ Cây Phả hệ và Hồ sơ bệnh án của hệ thống Quản Lý Bác Sĩ Gia Đình.

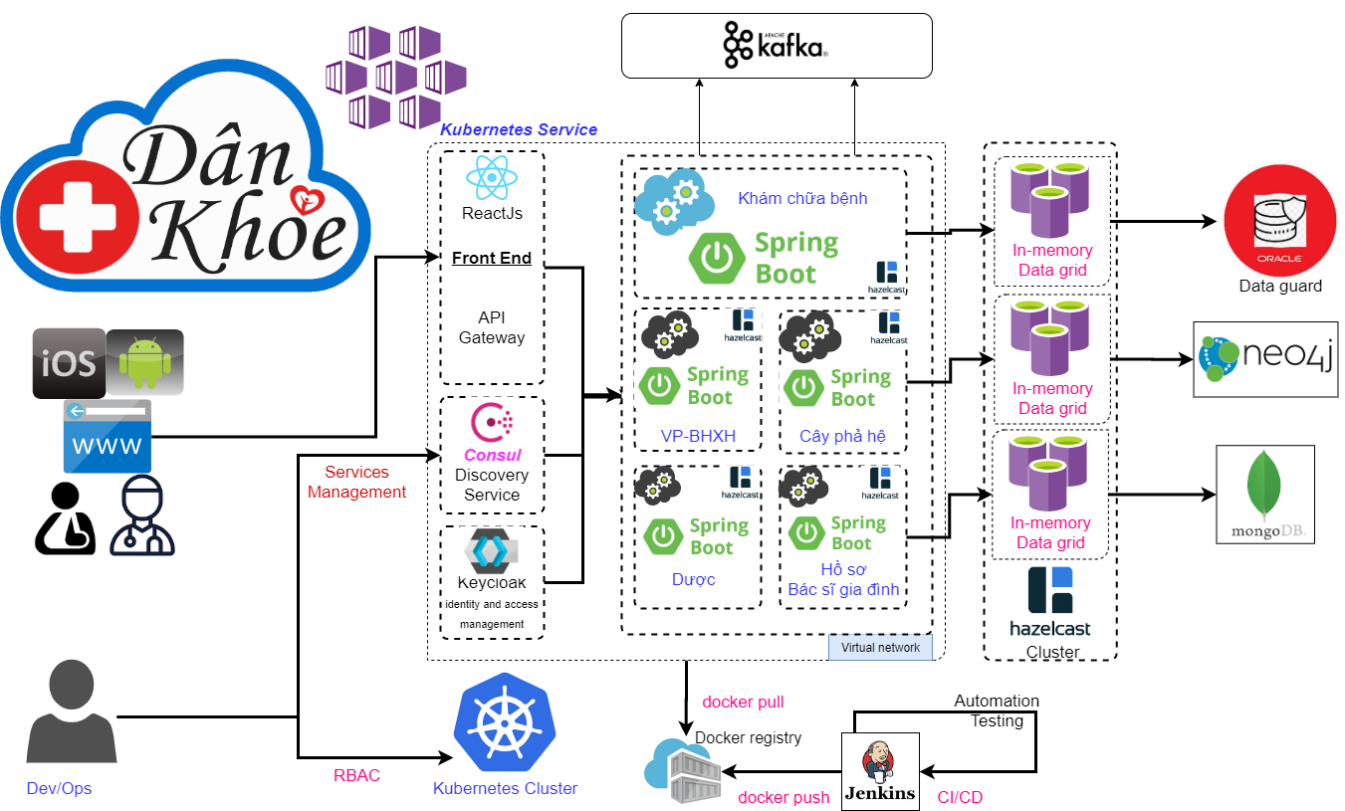
# 

# Chương 1. TỔNG QUAN

## **1.1. Giới thiệu đề tài**

**1.1.1 Đặc tả đề tài**

Để đáp bắt kịp với sự tiến bộ của các công nghệ mới và đáp ứng nhu cầu sử dụng của người dùng. Hệ thống bác sĩ gia đình sẽ kết hợp những công nghệ mới theo kiến trúc như sau:



Hình 1. Kiến thúc hệ thống bác sĩ gia đình

Như hình trên, hệ thống sẽ bao gồm 5 phân hệ: Khám chữa bệnh, Viện phí – Bảo hiểm xã hội, Dược, Cây Phả hệ và Hồ sơ bác sĩ Gia đình, 5 phân hệ trên sẽ được giao cho 4 nhóm thực hiện:

+ Nhóm 1: Thực hiện phân hệ Khám chữa bệnh do các anh ở VNPT đảm nhiệm.

+ Nhóm 2: Thực hiện phân hệ Viện phí – Bảo hiểm xã hội do các anh ở VNPT đảm nhiệm.

+ Nhóm 3: Thực hiện phân hệ Dược do các anh ở VNPT đảm nhiệm.

+ Nhóm 4: Thực hiện 2 phân hệ còn lại đó là Cây phả hệ và Hồ sơ bác sĩ Gia đình sẽ do Đặng Duy An và Võ Thị Tuyết Hương thực hiện (Duy An thực hiện phân hệ Cây phả hệ, Tuyết Hương thực hiện phân hệ Hồ sơ bác sĩ Gia đình).

Về mặt kiến trúc các phân hệ sẽ là các Service riêng biệt với giao diện sử dụng ngôn ngữ React, quản lý các Service sẽ sử dụng Keycloak và Consul, giao diện sẽ giao tiếp với service thông qua Gateway API, mỗi Service sẽ sử dụng Spring Boot để lập trình và giao tiếp trao đổi dữ liệu với nhau thông qua Kafka. Các phân hệ Khám chữa bệnh, Dược, Viện Phí – Bảo hiểm xã hội sẽ dùng cơ sở dữ liệu Oracle, phân hệ Cây phả hệ sẽ sử dụng Neo4J (Graph Database), phân hệ Hồ sơ Bác sĩ gia đình sẽ sử dụng MongoDB (NoSQL database) để quản lý các bệnh án.

Các phân hệ như Dược, Viện phí- BHXH và Khám chữa bệnh là những phân hệ độc lập không phụ thuộc vào phân hệ Khám chữa bệnh, còn hai phân hệ chúng tôi thực hiện sẽ phụ thuộc vào dữ liệu chuyển qua của phân hệ Khám chữa bệnh. Thứ tự ưu tiên như sau:

* Khám chữa bệnh 🡪Viện phí- BHXH 🡪Dược 🡪 Cây phả hệ 🡪 Hồ sơ bệnh án.
* Giao diện của các phân hệ độc lập đang được code trên một source Gateway Frontend chung và các source Backend độc lập. Giao diện của Cây phả hệ và Hồ sơ bệnh án sẽ được tích hợp vào giao diện chung sau khi đã hoàn thiện, các source Backend vẫn giữ độc lập và giao tiếp thông qua Kafka (nếu có nhu cầu trao đổi dữ liệu).

Sau đây là các chức năng và đặc điểm của hai phân hệ Cây phả hệ và Hồ sơ Bệnh Án mà chúng tôi thực hiện:

* Hồ sơ bệnh án bao gồm quản lý các thông tin của bệnh nhân như:
* Quản lý thông tin hành chính của bệnh nhân: quản lý các thông tin cơ bản của bệnh nhân khi được tiếp nhận như: họ tên, địa chỉ, ngày tháng năm sinh, giới tính…
* Quản lý bệnh sử (tiền căn): quản lý các bệnh bệnh nhân đã từng mắc trong quá khứ (lịch sử mắc bệnh).
* Quản lý tài chính (kinh tế): quản lý thu nhập, tình trạng kinh tế của bệnh nhân.
* Quản lý nghề nghiệp theo giai đoạn: quản lý các nghề nghiệp bệnh nhân đã làm tới giai đoạn hiện tại.
* Quản lý tiêm ngừa: quả lý lịch sử tiêm ngừa của bệnh nhân.
* Quản lý thông tin thói quen: quản lý các thói quen của bệnh nhân, như thường làm gì vào lúc nào.
* Quản lý thông tin khám bệnh…

🡪 Để quản lý các thông tin trên trong thực tế sẽ tốn rất nhiều thời gian, chi phí và công sức của các bác sĩ, y tá, chuyên viên y tế và bệnh viện nhưng chưa chắc sẽ có hiệu quả cao, việc thay thế các phương pháp lưu trữ truyền thống bằng hồ sơ bác sĩ gia đình sẽ làm cho việc xây dựng hệ thống Bác sĩ Gia Đình trở nên khả thi hơn (không cồng kềnh, dễ lưu trữ, dễ truy cập tìm kiếm, dễ phục vụ điều trị, tốn ít chi phí).

Cây phả hệ: Quản lý được các thành viên trong một gia đình ở cả gia đình cha và gia đình mẹ bao gồm các thông tin cơ bản của từng thành viên, để tiện cho việc xem các bệnh lây nhiễm qua di truyền, lây nhiễm do tiếp xúc gần với người thân trong gia đình, hay trạng thái tinh thần thể chất của mỗi thành viên. Các chức năng bao gồm:

* Truy xuất cây phả hệ ba đời, thêm, sửa, xóa, cập nhật các bệnh nhân theo dữ liệu đã được lưu từ phân hệ Khám chữa bệnh.
* Thống kê theo độ tuổi của các thành viên trong khu vực.
* Truy xuất thông tin chỉ tiết của mỗi bệnh nhân khi nhấn vào một bệnh nhân trong giao diện cây phả hệ.
* Hiển thị mối quan hệ cơ bản và loại quan hệ hiện tại của các hai bệnh nhân nếu họ có mối quan hệ.
* Hiển thị vai trò chức vụ của các thành viên khi truy xuất cây phả hệ.

### 1.1.2. Lý do chọn đề tài

Y tế tế điện tử (ehealth) là lĩnh vực áp dụng các công nghệ của ngành tin học, máy tính vào ngành y tế, nó được biết đến phổ biến từ năm 1999 và đến nay đang được nhiều quốc gia trên thế giới áp dụng vào lĩnh vực y học. Nó được tạo ra theo cảm hứng của nhiều từ có cấu trúc tương tự (e-word): e-commerce, e-business, e-solution… với mong muốn truyền tải một nội dung tương tự về nguyên tắc, về lợi ích, về tiềm năng… của lĩnh vực thương mại điện tử vào trong ngành y tế. Việt Nam cũng đang tích cực đẩy mạnh việc áp dụng các thành tựu công nghệ của ngành công nghệ thông tin vào quản lý ngành y tế để nâng cao hiệu năng và độ chính xác khi làm việc.

Ngoài ra các loại Cơ sở dữ liệu, công nghệ, thư viện lập trình mới đem lại hiệu năng và tính đa dụng cao xuất hiện ngày càng nhiều mà tiêu biểu là Microservices (Jhipster), Graph Database (Neo4J), NoSQL Database (MongoDB), Spring Framework, ReactJS + Redux, TypeScript,.. giúp tăng cường tính bảo mật, hiệu năng, khả năng tương thích mang lại nhiều lợi ích so với cách lập trình thông thường.

Bên cạnh đó thì tính cấp thiết về hiện thực hóa, tối ưu hóa và áp dụng công nghệ thông tin cho mô hình Bác sĩ Gia đình ở Việt Nam đang ngày càng quan trọng.

Chính vì những lý do trên nên chúng tôi đã chọn đề tài “Xây dựng phân hệ khám chữa bệnh trong ứng dụng quản lý hệ thống bác sĩ gia đình”.

### 1.1.3. Mục tiêu đề tài

Tìm hiểu lý thuyết:

* Tìm hiểu về Microservice (Jhipster Framework).
* Tìm hiểu về các công nghệ Frontend (ReactJS, TypeScript).
* Tìm hiểu về các công nghệ Backend (Java, Spring Framework, Jasper Report).
* Graph Database (Neo4J Database).
* NoSQL Database (MongoDB).
* Nắm được quy trình quản lý bệnh nhân theo cây phả hệ (dùng Graph Database).

Ứng dụng thực nghiệm:

* Xây dựng được hệ thống quản lý bệnh nhân theo cây phả hệ: Quản lý các thông tin của bệnh nhân, quản lý các thông tin liên quan đến gia đình của bệnh nhân và quan hệ giữa các thành viên trong gia đình.
* Quản lý, lưu trữ được các thông tin khám chữa bệnh của bệnh nhân trong hồ sơ bệnh án bác sĩ gia đình kể từ khi bệnh nhân đó sinh ra đến khi bệnh nhân đó mất đi.
* Liên kết giao diện, thông tin với service Khám Chữa Bệnh và các service liên quan được chính xác, nhanh chóng và hiệu quả.
* Ứng dụng có đáp ứng đủ cho Hồ sơ bác sĩ gia đình và có khả năng áp dụng vào thực tế.
* Ứng dụng hoạt động hiệu quả, nhanh chóng.
* Giao diện trực quan và dễ sử dụng.

### 1.1.4. Nhiệm vụ

Tìm hiểu lý thuyết:

* Tìm hiểu về microservice (Jhipster Framework).
* Tìm hiểu về các công nghệ FrontEnd (ReactJs, TypeScript).
* Tìm hiểu về các công nghệ BackEnd (Java, Spring Framework, Jasper Report).
* Tìm hiểu Graph Database (Neo4j Database), NoSQL Database (MongoDB), nắm được ưu khuyết điểm của chúng.
* Tìm hiểu quy trình quản lý bệnh nhân theo cây phả hệ.
* Tìm hiểu tài liệu Thông tin bác sĩ gia đình.

Ứng dụng thực nghiệm:

#### **Bảng 1. Hệ thống cây phả hệ**

|  |  |
| --- | --- |
| ***Yêu cầu thông tin đầu vào*** | ***Yêu cầu thông tin đầu ra*** |
| Thông tin IdBenhNhan, lấy từ service khám chữa bệnh | Chức năng thêm, xóa, sửa quan hệ bệnh nhân theo tiêu chuẩn của Graph Database |
| Thông tin IDBenhNhan | Thông tin IdBenhNhan của tất cả bệnh nhân có quan hệ với bệnh nhân đó |
| Thông tin IDBenhNhan | Thông tin theo dạng đồ thị của tất cả bệnh nhân có quan hệ với bệnh nhân đó |
| Thông tin cơ bản của bệnh nhân khi cần hiển thị ở đồ thị | Hiển thị thông tin cơ bản của bệnh nhân Khi nhấp vào node của đồ thị |
| Khi nhấn vào một nút của bệnh nhân trong giao diện và chọn mở hồ sơ bệnh án | Chuyển giao diện sang giao diện hồ sơ bệnh án bác sĩ gia đình |
| Sự thay đổi các mối quan hệ của bệnh nhân | Quản lý các mối quan hệ của bệnh nhân như lập gia đình, ly dị, sinh con v.v... |

#### **Bảng 2. Hồ sơ bệnh án bác sĩ gia đình:**

|  |  |
| --- | --- |
| **Yêu cầu thông tin đầu vào** | **Yêu cầu thông tin đầu ra** |
| Thông tin chính bệnh nhân | Chức năng thêm sửa xóa Thông tin hành chính của bệnh nhân vào HSBA bác sĩ gia đình. |
| Các danh mục (danh mục địa chỉ, nghề nghiệp, tôn giáo) | Chức năng quản lý nghề nghiệp theo giai đoạn. |
| Kinh tế | Quản lý tài chính theo giai đoạn. |
| Thông tin liên hệ, thông tin người thân | Chức năng thêm sửa xóa thông tin liên hệ, thông tin người thân bệnh nhân vào HSBA bác sĩ gia đình |
| Thông tin bệnh tật bệnh nhân | Quản lý bệnh sử (Tiền căn), quản lý theo biểu đồ giai đoạn, từ bào thai, sơ sinh 🡪 nhủ nhi 🡪 Nhà trẻ mẫu giáo 🡪 v/v |
| Thông tin tiêm ngừa | Quản lý tiêm ngừa |
| Tai nạn | Quản lý tai nạn thương tích |
| Thông tin thói quen | Quản lý thông tin thói quen theo thời gian |
| Quản lý kế hoạch hóa gia đình | Quản lý kế hoạch hóa gia đình |
| An toàn tình duc- phụ khoa | Quản lý thông tin an toàn tình duc- phụ khoa |
| Tâm lý bệnh nhân | Quản lý tâm lý bệnh nhân theo giai đoạn thời gian biến cố gia đình |
| Thông tin môi trường | Quản lý thông tin môi trường địa phương hiện tại |
| Thông tin khám, chuẩn đoán bệnh | Chức năng thêm, xóa, sửa thông tin khám, chẩn đoán vào HSBA bác sĩ gia đình |
| Tài liệu chăm sóc sức khỏe cơ bản | Quản lý tài liệu, để suất thông tin giáo dục sức khỏe phù hợp |
| Kế hoạch điều trị, chăm sóc, tầm soát, dự phòng | Quản lý kế hoạch điều trị, chăm sóc dự phòng bệnh |

### 1.1.5. Phạm vi nghiên cứu của đề tài

Tìm hiểu lý thuyết: Do toàn bộ hệ thống áp dụng rất nhiều công nghệ mới nên chúng tôi chỉ tập trung tìm hiểu các công nghệ sử dụng cho phân hệ Cây Phả hệ và Hồ sơ bệnh án Bác sĩ gia đình mà nhóm sẽ thức hiện.

Ứng dụng thực nghiệm: Hoàn thiện các chức năng đã liệt kê để có thể đưa vào hệ thống tổng thể cùng các phân hệ khác.

## **1.2. Các vấn đề tập trung giải quyết**

Giải quyết các vấn đề sau:

* Thiết kế theo kiến trúc Microservice, đảm bảo tính bảo mật, hoạt động ổn định, kết nối truyền dữ liệu một cách chính xác tới các service liên quan.
* Giao diện đơn giản, dễ sử dụng : có bố cục hợp lí, các menu chức năng rõ ràng.
* Ứng dụng có thể chạy được trên các trình duyệt web trên máy tính như (Google Chrome, IE,…)

# Chương 2: CƠ SỞ LÝ THUYẾT VÀ THỰC TIỄN

## **2.1. Cơ sở lý thuyết**

### 2.1.1. Giả thuyết nghiên cứu

Nếu xây dựng thành công hệ thống sẽ giúp cho quá trình quản lý thông tin các bệnh nhân trong cùng một gia đình, quản lý, lưu trữ thông tin khám chữa bệnh của bệnh nhân được chính xác, dễ dàng và hiệu quả hơn.

### 2.1.2. Phương pháp nghiên cứu

Phương pháp nghiên cứu lý thuyết: Nghiên cứu lý thuyết các vấn đề liên quan như cơ sở dữ liệu, ngôn ngữ lập trình, tài liệu. Nghiên cứu tài liệu:

* Những vấn đề cấp thiết về thực tế lẫn nhu cầu áp dụng Công nghệ thông thông tin vào quản lý mô hình phòng khám Bác sĩ Gia đình trong Tài liệu “ĐỀ ÁN XÂY DỰNG VÀ PHÁT TRIỂN MÔ HÌNH PHÒNG KHÁM BÁC SĨ GIA ĐÌNH GIAI ĐOẠN 2013-2020”
* Những ưu điểm của bệnh án điện tử trong quản lý Hồ sơ bệnh án so với thực tế trong Tài liệu “BỆNH ÁN ĐIỆN TỬ NÂNG CAO CHUYÊN MÔN TRONG THỰC HÀNH Y HỌC GIA ĐÌNH”.
* Những đặc điểm về Hồ sơ bệnh án gia đình và quản lý gia đình bằng cây phả hệ trong Tài liệu “BỆNH ÁN Y HỌC GIA ĐÌNH”.
* Mối liên hệ của Bác sĩ gia đình trong thực tế với các mô hình khác tương tự đặc biệt là Trạm Y tế phường trong Tài liệu “VỊ TRÍ TRẠM Y TẾ PHƯỜNG VÀ MỐI LIÊN HỆ VỚI BỆNH VIỆN QUẬN TRONG MẠNG LƯỚI BÁC SĨ GIA ĐÌNH”.

Phương pháp quan sát khoa học: Quan sát khách quan các vấn đề liên quan đến quy trình quản lý bệnh nhân theo cây phả hệ, tài liệu thông tin bác sĩ gia đình.

Phương pháp thực nghiệm: Áp dụng lấy dữ liệu từ hệ thống thực tế.

Phương pháp nghiên cứu dựa theo quy trình: Quy trình thiết kế gồm các bước sau đây:

* Xác định yêu cầu: Thu thập thông tin từ các văn bản pháp luật và các văn bản liên quan đến bác sĩ gia đình.
* Phân tích: Phân tích, nghiên cứu để giải quyết vấn đề từ các yêu cầu thu thập được, từ bước xác định yêu cầu, vẽ các mô hình, lớp xử lý dữ liệu của bài toán.
* Thiết kế: Thiết kế giao diện và các giải thuật, các lớp xử lí từ việc phân tích yêu cầu để có thể hoàn thành hệ thống.
* Cài đặt: Sau khi hoàn thành cơ bản chức năng của hệ thống, cần cho chạy thử nghiệm để kiểm tra, tìm lỗi và sửa lỗi...
* Kiểm chứng: thu thập, lấy ý kiến đánh giá của người sử dụng. Từ đó, tổng kết để thấy được những cái hoàn thiện, thiếu sót của hệ thống để có thể chỉnh sửa kịp thời.
* Phát triển thực tế: Nếu đề tài được ứng dụng sẽ triển khai và cài đặt trên host để có thể chính thức hoạt động sẽ hỗ trợ cho quá trình quản lý thông tin các bệnh nhân trong cùng một gia đình có cùng huyết thống theo tiêu chuẩn của bác sĩ gia đình bao gồm cây phả hệ và hồ sơ bệnh án của mỗi bệnh nhân.

### 2.1.3. Công nghệ sử dụng

* Hệ điều hành: Windows 10
* Ngôn ngữ lập trình chính: ReactJS
* Hệ quản trị cở sở dữ liệu: Neo4J, MongoDB
* Môi trường lập trình: Trình soạn thảo mã nguồn: Visual Studio Code, IntelliJ IDEA 2019
* Trình duyệt web: Google Chrome, Microsoft Edge

### 2.1.4. Cơ sở lý thuyết

Theo như yêu cầu của hệ thống, do hai phân hệ Cây Phả hệ và Hồ sơ bệnh án mục đích sử dụng và các yêu cầu phần mềm đặc ra là khác nhau. Cây phả hệ cần quản lý bệnh nhân và mối quan hệ của họ một cách chặt chẽ, dễ dàng truy xuất. Trong khi mục đích của Hồ sơ bệnh Án là quản lý thông tin, lưu trữ truy xuất nhanh. Ngoài ra giao diện cần phải hiện đại, hiệu năng tinh chỉnh tốt, chính vì thế Microservices, MongoDb và Neo4J và ReactJs là những công cụ phù hợp

a. Microservices

Kiến trúc nguyên khối:

* Lợi ích của Monolithic: Dễ dàng phát triển, triển khai, mở rộng.
* Hạn chế của Monolithic:
* Khi source trở nên càng ngày càng lớn sẽ gây khó khăn cho lập trình viên, code càng trở nên nhiều và khó hiểu hơn.
* Các công cụ phát triển bị quá tải.
* Giảm hiệu năng khi triển khai và hoạt động do Source code quá lớn.
* Chỉ cần cập nhật một module nhỏ cũng có thể gây lỗi cả hệ thống, mỗi lần cập nhật sẽ rất khó khăn gây ảnh hưởng đến khách hàng.
* Việc mở rộng ứng dụng trở nên khó khăn do chỉ mở rộng theo một chiều (tạo ra nhiều bản sao):
* Tăng mức tiêu thu bộ nhớ và lưu lượng truy cập I/O.
* Không thể phân tỷ lệ tài nguyên cho hợp lý (RAM và CPU).
* Giảm khả năng hoạt động độc lập, việc cập nhật hay sửa đổi của một nhóm có thể ảnh hưởng đến cả những nhóm khác.
* Bị bó buột bởi một số công nghệ nhất định.

Kiến trúc Microservices:

* Để giải quyết những vấn đề của kiến trúc nguyên khối, ý tưởng được đề xuất là chia nhỏ hệ thống lớn ra thành những dịch vụ nhỏ kết nối với nhau. Mỗi dịch vụ nhỏ thực hiện một tập các chức năng chuyên biệt như quản lý đơn hàng, quản lý khách hàng,…Mỗi dịch vụ là một ứng dụng nhỏ kết nối thông qua bộ chuyển đổi (Adapter) khác nhau. Một số dịch vụ có nhiệm vụ làm API cho phép giao tiếp với các dịch vụ nhỏ khác hay các ứng dụng khác gọi tới. Khi vận hành, mỗi dịch vụ nhỏ được chạy trong một máy ảo (Virtual Machine) hoặc Docker Container (ảo hóa tầng ứng dụng).
* Như vậy, ứng dụng web đã có thể chia nhỏ hơn cho từng nhóm đối tượng người dùng. Thiết kế giao diện cho từng nhóm đối tượng người dùng giúp tối ưu hóa trải nghiệm, tốc độ nhanh hơn, dễ tương thích hơn trong khi cấu trúc chức năng đơn giản hơn.
* Kiến trúc Microservices chú trọng nhiều đến quan hệ giữa ứng dụng và CSDL. Thay vì dùng chung một cơ sở dữ liệu giữa các dịch vụ, mỗi dịch vụ sẽ có CSDL riêng. Điều này giúp hạn chế sự dư thừa dữ liệu và phát huy được vai trò riêng biệt của từng dịch vụ.

Ưu điểm của kiến trúc Microservices:

* Cho phép phân phối và triển khai một các liên tục các ứng dụng lớn và phức tạp.
* Cải thiện khả năng bảo trì
* Cải thiện khả năng kiểm thử (các service nhỏ nên việc kiểm thử dễ dàng và nhanh chóng hơn)
* Khả năng triển khai tốt hơn (các service có khả năng triển khai độc lập).
* Kiến trúc Microservices cho phép tổ chức các nhóm tự chủ cho một hoặc một nhóm service. Các nhóm tự chủ này có thể phát triển, thử nghiệm, triển khai, mở rộng quy mô một các độc lập với các nhóm khác.
* Mỗi service tương đối nhỏ.
* Lập trình viên dễ hiểu hơn.
* Các công cụ mà lập trình viên sử dụng cũng hoạt động hiệu quả hơn.
* Các ứng dụng khởi động nhanh hơn và giúp lập trình viên hoạt động hiệu quả hơn.
* Có thể cách ly được lỗi.
* Loại bỏ khả năng phụ thuộc công nghệ nào đó một cách lâu dài. Khi phát triển một service mới chúng ta có thể chọn công nghệ mới cho service mới hoặc có thể lại service cũ bằng một công nghệ khác mà ít ảnh hưởng tới các service khác.

Nhược điểm của Microservices:

* Việc xử lý các request trên nhiều service trở nên khó khăn hơn, phải phối hợp cẩn thận với nhiều nhóm nếu mỗi nhóm đảm nhiệm một Service khác nhau.
* Khó khăn trong việc kiểm thử vì phải tương tác với nhiều service.
* Tốn nhiều bộ nhớ hơn (do chia ra nhiều Service con).
* Việc triển khai cũng trở nên phức tạp khi phải giám sát một số lượng lớn các service.
* Tuy có những khuyết điểm nhưng đó là những hạn chế có thể tránh khỏi nếu lập trình viên làm việc có kỉ luật, đã quen với hệ thống và hoàn toàn không liên quan đến những hạn chế về yếu tố kĩ thuật (không thể thay đổi trong thời gian ngắn được).

b. ReactJS, Redux, Typescript, JasperReports

Giới thiệu về ReactJS:

* ReactJS là một thư viện giao diện người dùng, được Facebook phát triển để hỗ trợ việc xây dựng những thành phần giao diện người dùng có tính tương tác cao, có trạng thái và có thể sử dụng lại được.
* Một trong những điểm hấp dẫn của ReactJS là thư viện này không chỉ hoạt động trên phía người dùng, mà còn được kết xuất (render) trên máy chủ và có thể kết nối với nhau.
* Mô hình đối tượng tài liệu ảo (Virtual DOM – DOM ảo): Công nghệ DOM ảo giúp tăng hiệu năng cho ứng dụng. Việc chỉ nút gốc mới có trạng thái và khi nó thay đổi sẽ tái cấu trúc lại toàn bộ, đồng nghĩa với việc cây DOM cũng sẽ phải thay đổi một phần, điều này sẽ ảnh hưởng đến tốc độ xử lý. ReactJS sử dụng Virtual DOM để cải thiện vấn đề này.
* ReactJS sử dụng cơ chế luồng dữ liệu một chiều. Dữ liệu được truyền từ thành phần cha đến thành phần con thông qua ‘props’. Luồng dữ liệu đơn giản giúp chúng ta dễ dàng kiểm soát cũng như sửa lỗi. Với các đặc điểm ở trên, ReactJS dùng để xây dựng các ứng dụng lớn mà dữ liệu của chúng thay đổi liên tục theo thời gian.

Giới thiệu về JSX:

* JSX là một dạng ngôn ngữ cho phép viết các mã HTML trong Javascript. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javacsript.
* Ngược với Javascript, JSX là kiểu được biên dịch trước khi chạy, giống như Java, C++. Vì thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch. Ngoài ra, nó cũng cung cấp tính năng gỡ lỗi khi biên dịch rất tốt.
* JSX kế thừa dựa trên Javascript, vì vậy rất dễ dàng để cho các lập trình viên Javascripts có thể sử dụng.

Giới thiệu về Components (thành phần):

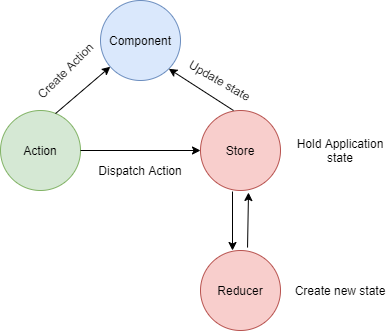
ReactJS được xây dựng xung quanh các thành phần, gọi là Component, chứ không dùng khung mẫu như các framework khác. Trong ReactJS, chúng ta xây dựng trang web sử dụng những Component nhỏ. Chúng ta có thể tái sử dụng một Component ở nhiều nơi, với các trạng thái hoặc các thuộc tính khác nhau, trong một Component lại có thể chứa Component khác. Mỗi Component trong React có một trạng thái riêng, có thể thay đổi, và ReactJS sẽ thực hiện cập nhật Component dựa trên những thay đổi của trạng thái. Component giúp bảo trì mã lệnh khi làm việc với các dự án lớn. Một React Component đơn giản chỉ cần một phương thức kết xuất, gọi là Render Method. Có rất nhiều phương thức khả dụng khác, nhưng Render Method là phương thức chủ yếu.

Giới thiệu về Props và States:

* Props: Giúp các Component tương tác với nhau, Component nhận giá trị trả về gọi là props, và trả thuộc tính mô tả những gì Component con sẽ kết xuất. Props là phần tử không thay đổi.
* States: Thể hiện trạng thái của ứng dụng, khi state thay đồi thì Component đồng thời kết xuất lại để cập nhật giao diện người dùng. States có thể thay đổi bằng cách sử dụng this.setState();

Giới thiệu về Redux:

* Redux là một công cụ quản lý cho các ứng dụng Javascript, có thể gọi Redux là một kho chứa.
* Mặc dù chủ yếu được sử dụng với React, nhưng nó có thể được sử dụng với bất kỳ thư viện hoặc Framework JavaScript nào khác.
* Với Redux, States của ứng dụng được giữ trong một nơi gọi là Store và mỗi Component đều có thể truy cập bất kỳ States nào mà chúng muốn từ Store này.
* Cách hoạt động của Redux:
* Redux có một Store lưu trữ toàn bộ State của các Component. Mỗi Component có thể truy cập trực tiếp đến State được lưu trữ thay vì phải gửi Props từ Component này đến Component khác. Có 3 thành phần của Redux: Actions, Store, Reducers.
* Action: Actions đơn giản là các sự kiện. Chúng là cách mà chúng ta gửi dữ liệu từ ứng dụng đến Store. Những dữ liệu này có thể là từ sự tương tác của người dùng với ứng dụng, gọi từ API hoặc cũng có thể là từ gửi dữ liệu của bảng.
* Reducers: Reducers là các phương thức nguyên thủy (Pure Funtioncs) chúng lấy State hiện tại của ứng dụng, thực hiện một Action và trả về một state mới. Những State này được lưu như những đối tượng và chúng định rõ cách State của một ứng dụng thay đổi trong việc phản hồi một Action được gửi đến Store.
* Store: Store lưu trạng thái ứng dụng và nó là duy nhất trong bất kỳ một ứng dụng Redux nào. Người lập trình có thể truy các State được lưu, cập nhật State.



Hình 2. Mô hình hoạt động của Redux [3]

Giới thiệu về TypeScript: TypeScript là một dự án mã nguồn mở được phát triển bởi Microsoft, nó có thể được coi là một phiên bản nâng cao của Javascript bởi việc bổ sung tùy chọn kiểu tĩnh và lớp hướng đối tượng mà điều này không có ở Javascript.

Giới thiệu JasperReports:

* JasperReports là một công cụ tạo báo cáo mã nguồn mở phổ biến nhất hiện tại. Nó được viết hoàn toàn bằng Java và được cho phép sử dụng với nhiều loại nguồn dữ liệu và tạo ra những báo cáo cho phép hiển thị, in ấn hoặc xuất ra những định dạng như HTML, PDF, Excel,…, hỗ trợ các nhà phát triển với nhiệm vụ bổ sung khả năng báo cáo cho các ứng dụng Java Application.
* Hầu hết các ứng dụng trong thực tế cần phải xuất các báo cáo theo nghiệp vụ cũng như theo yêu cầu của khách hàng. Vì vậy, JasperReports sẽ là một công cụ tạo báo cáo đơn giản, mạnh mẽ sẽ giúp tiết kiệm được nhiều công sức, thời gian. Các báo cáo được thiết kế bằng một phần mềm cung cấp riêng để tạo báo cáo có tên JasperSoft Studio.

c. Giới thiệu về Graph Database (Neo4J)

Graph Database:

* Graph database là database sử dụng cấu trúc graph để lưu trữ dữ liệu, nó khác so với cơ sỡ dữ liệu quan hệ lưu data dưới dạng bảng, hay NoSQL lưu data dưới dạng document key value.
* Các record dữ liệu trong graph database được gọi là Nodes, các node được kết nối với nhau bằng các quan hệ Relationships. Nodes và Relationships đều có thể có các thuộc tính Properties. Các Nodes có thể phân thành các group để dễ quản lý nhờ vào Label.
* Graph database không có mục đích là thay thế relational database hay NoSQL database, nó được dùng cho các ứng dụng focus vào các quan hệ (relationship) giữa các thực thể dữ liệu.
* Với những kiểu ứng dụng như vậy thì cách lưu trữ dữ liệu của relational database hoặc NoSQL database đều không thích hợp khi mà chúng chỉ normalize dữ liệu để lưu vào các bảng hay các document và làm giảm đi tính kết nối. Cách duy nhất để tạo kết nối ở 2 loại database trên là tạo các foreign key/link đến các record/document khác, và việc join các bảng hay lần ngược lại các link là việc phức tạp, kém hiệu quả và dễ gây lỗi, đặc biệt nếu các quan hệ là phức tạp
* Ưu điểm:
* Performance là ưu điểm rất lớn của graph databse so với relational database và Nosql database. Khi dữ liệu trở nên lớn cùng với việc sử dụng nhiều câu lệnh join sẽ khiến performance trở nên rất tệ, trong khi thì graph database với cách query dữ liệu chỉ focus vào 1 phân khu của graph có chức các dữ liệu liên quan nên performance không thay đổi nhiều.
* Có thể thêm vào các nodes, các relationships, các subgraph mà không làm ảnh hưởng đến dữ liệu cũ, các query cũ.

Neo4J:

* Neo4j là một trong những GD phổ biển nhất hiện nay. Neo4j hỗ trợ ngôn ngữ truy vấn Cypher, một ngôn ngữ rất trực quan trong việc truy vấn dữ liệu dưới dạng Graph.
* **Nodes** thường được sử dụng để biểu diễn các thực thể (entities). Đồ thị đơn giản nhất là đồ thì mà trong đó chỉ có duy nhất một node.
* ***Labels*** **có** thể được sử dụng để mô hình hóa miền giá trị của các node lại với nhau, thông thường sẽ gộp nhóm các node có cùng kiểu dữ liệu hoặc là thuộc tính thành một tập hợp rồi sau đó gắn label vào cho chúng.
* Mỗi một node thì có thể có một hoặc nhiều lables. Mỗi một nhãn lúc này sẽ biểu diễn một lớp đối tượng khác nhau. Nhưng trong những bài toán khác khi mà chúng ta muốn biểu diễn thêm những chiều khác nhau của dữ liệu(different dimensions of the data) có thể thêm labels vào cho các nodes.
* ***Relationships*** đúng như cái tên của nó đó là sẽ biểu mối quan hệ, hay liên kết giữa các node với nhau.
* Các ***relationships*** sẽ giúp cho đồ thị sẽ có ý nghĩa hơn, gẫn gũi với bài toán thực tế hơn.
* Mỗi một ***relationship*** chỉ được phép có đúng một ***Relationship type***.
* **Properties** là một cặp name-value, được dùng để biểu diễn cho các thuộc tính của các ***nodes*** cũng như là các ***relationships***.
* Các property có thể lưu trữ các kiểu dữ liệu đa dạng khác nhau như là number, string và boolean với các miền giá trị tương ứng. Để hiểu rõ hơn về vấn đề này chúng ta có thể tìm hiểu thêm phần **Cypher manual.**

d. Giới thiệu về NoSQL Database (MongoDB)

Giới thiệu về NoSQL Database:

* NoSQL là 1 dạng CSDL mã nguồn mở không sử dụng các câu lệnh T-SQL để truy vấn thông tin. NoSQL viết tắt bởi: None-Relational SQL, hay có nơi thường gọi là Not-Only SQL. NoSQL có thể mở rộng dữ liệu một cách thoải mái mà không cần phải lo lắng về khóa ngoại hay là khóa chính, hoặc các công việc như là kiểm tra ràng buộc như hệ quản trị cơ sở dữ liệu quan hệ (RDBMS).
* Ưu điểm của NoSQL: NoSQL có thể giải quyết được các vấn đề dữ liệu lớn (big data) về các hệ thống thông tin hoặc là phân tán dữ liệu

Giới thiệu về MongoDB:

* MongoDB là một mã nguồn mở và là một tập tài liệu dùng cơ chế NoSQL để truy vấn. MongoDB được viết bằng C++.
* Ngoài ra, MongoDB là một cơ sở dữ liệu đa nền tảng, định hướng tài liệu cung cấp, hiệu suất cao, tính sẵn sàng cao và khả năng mở rộng dễ dàng. MongoDB hoạt động trên khái niệm về collection và document. MongoDB lưu trữ dữ liệu dưới dạng JSON (gồm các cặp key – value).
* Ưu điểm của MongoDB: Linh hoạt, cấu trúc đối tượng rõ ràng, dễ mở rộng
* Các thuật ngữ hay sử dụng trong MongoDB:

**+ \_id**: Là trường bắt buộc có trong mỗi document. Trường \_id đại diện cho một giá trị duy nhất trong document MongoDB. Trường \_id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một \_id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.

**+ Collection**: Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.

**+ Document**: Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các tên trường và giá trị (key -value).

**+Field**: là một cặp key-value. Một Document có thể không có trường nào hoặc có thể có nhiều trường. Các trường này tương tự như các cột trong hệ quản trị cơ sở dữ liệu quan hệ (RDBMS).

**+ JSON**: được biết đến như là một ký hiệu đối tượng JavaScript. Đây là một dạng văn bản đơn giản, dễ đọc để thể hiện dữ liệu có cấu trúc. Hiện tại JSON được hỗ trợ trong nhiều ngôn ngữ lập trình.

e. Giới thiệu về Spring

Giới thiệu về Spring trong Java:

* Spring là một Framework phát triển các ứng dụng. Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code…
* Spring là một mã nguồn mở, được phát triển, chia sẻ và có cộng đồng người dùng rất lớn.
* Spring được chia làm nhiều module khác nhau, tùy theo mục đích phát triển ứng dụng mà ta dùng 1 trong các module đó.
* Trong số các module của Spring, chúng tôi sẽ sử dụng module Spring Boot để xây dựng kiến trúc ứng dụng

Giới thiệu về Spring Boot:

* Spring Boot là một module của Spring Framework, cung cấp tính năng RAD (Rapid Application Development) – Phát triển ứng dụng nhanh.
* Spring Boot được dùng để tạo các ứng dụng độc lập dựa trên Spring.
* Spring Boot không yêu cầu cấu hình XML
* Nó là một chuẩn cho cấu hình thiết kế phần mềm, tăng cao năng suất cho developer.
* Ưu điểm của SpringBoot:

+ Có các tính năng của Spring Framework.

+ Tạo ứng dụng độc lập, có thể chạy bằng java -jar (cho cả java web)

+ Nhúng trực tiếp các ứng dụng server (Tomcat, Jetty…) do đó không cần phải triển khai file WAR

+ Cấu hình ít, tự động cậu hình bất kì khi nào có thể (Giảm thời gian viết code, tăng năng suất)

+ Không yêu cầu XML config…

+ Cung cấp nhiều plugin

+ Chuẩn cho Microservices (Cloud support; giảm việc setup, config; các thư viện hỗ trợ…)

f. Sử dụng JHipster để tạo nhanh mô hình Microservice

Giới thiệu về JHipster:

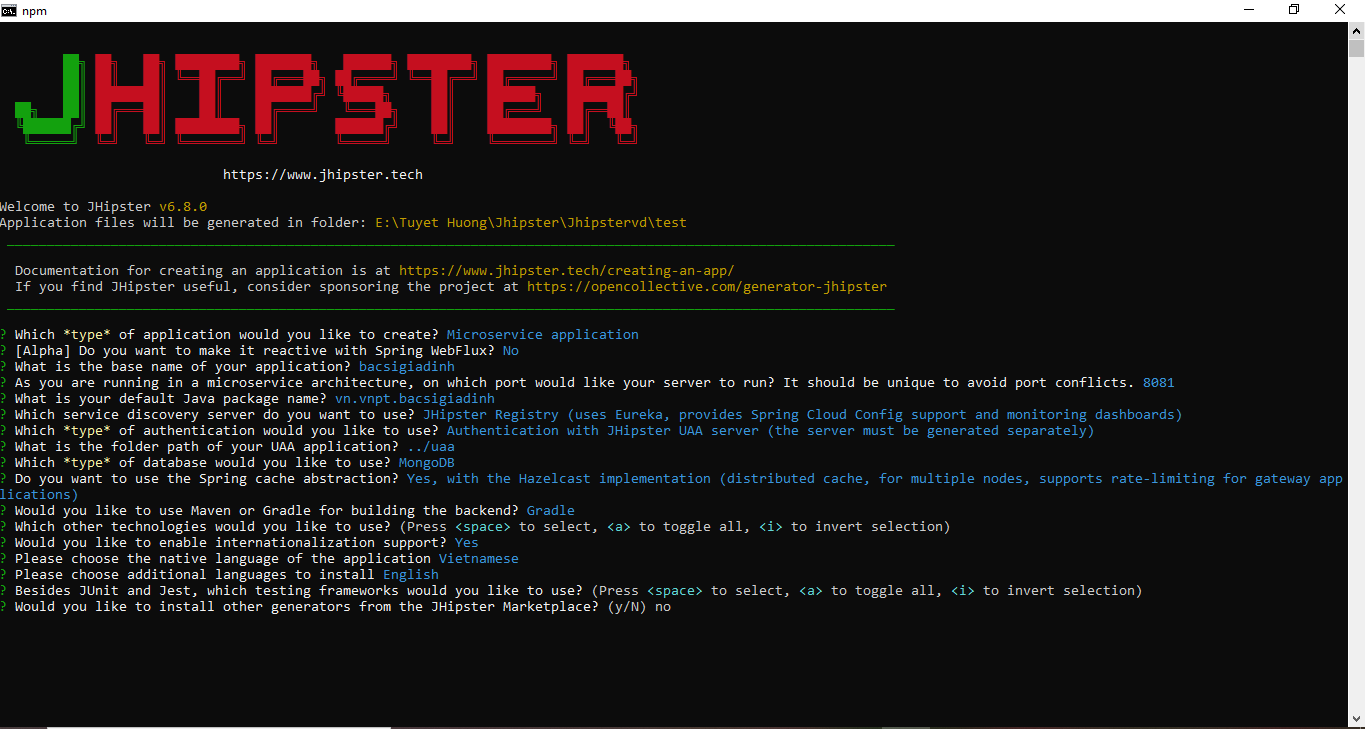
* JHipster là một framework được phát triển bởi Yeoman và được sử dụng để phát triển các dự án Spring Boot + AngularJS/ReactJS/Vue.
* Jhipster sẽ giúp chúng ta dễ dàng có được một project đủ mạnh mẽ đầy đủ những thứ cơ bản để bắt đầu với thời gian nhanh nhất.

Cài đặt thư viện cho Jhipster:

* Cài đặt và cấu hình Java 8 trên máy tính
* Cài đặt Node-js từ trang chủ
* Cài đặt Yeoman sử dụng lệnh: npm install -g yo
* Cài Jhipster sử dụng lệnh: npm install -g generator -jhipster

Khởi tạo dự án: Tất cả câu lệnh đều nhập thủ công trên command prompt (cmd) của máy tính. Để tạo dự án JHipster, thực hiện:

* Tạo một thư mục trống là nơi sẽ chứa dự án của bạn bằng lệnh: mkdir myapplication
* Chuyển đến thư mục đó: cd myapplication
* Để tạo dự án của bạn, gõ: jhipster
* Sau đó lựa chọn những cái phù hợp với dự án của bạn
* Bây giờ thì bạn đã có một dự án với:
* Backend: Spring Boot + Spring Security
* Database: MongoDB hoặc Neo4J hoặc MySQL
* Frontend: Bootstrap + SASS + React (hoặc thấp hơn tùy vào phiên bản JHipster người dùng cài đặt ở trên).
* Sau khi chạy xong bạn sẽ nhận được kết quả như bên dưới. Đó cũng là hướng dẫn để bạn xây dựng dự án của mình trên local.
* Chạy ứng dụng Spring Boot bằng cách nhập: ./mvnw
* Sau đó dùng lệnh để chạy: npm start



Hình 3. Minh họa tạo dự án Jhipster

## **2.2. Cơ sở thực tiễn và ứng dụng của khóa luận**

### 2.2.1. Bối cảnh thực tiễn

Hoạt động bác sĩ gia đình ở nước ta hiện nay là mô hình mới, chưa được quan tâm đầu tư tương xứng, chưa có chức danh bác sĩ gia đình ở các cơ sở y tế, hoạt động còn tản mạn, nhiều hạn chế, bất cập, chưa có đầy đủ cơ sở pháp lý và hiệu quả chưa cao.

Nhu cầu thực tế:

* Xây dựng phần mềm tin học quản lý các thông tin về sức khỏe của cá nhân, gia đình và cộng đồng; kết nối mạng giữa phòng khám bác sĩ gia đình với người bệnh, với các cơ sở khám bệnh, chữa bệnh và các cơ sở y tế khác.
* Xây dựng bệnh án điện tử y học gia đình.
* Bảo đảm bảo mật các thông tin về sức khỏe của người bệnh

### 2.2.2. Ứng dụng

* Khóa luận này là một tài liệu tham khảo có ích cho các sinh viên khi tìm hiểu về mô hình Microservices.
* Sản phẩm phần mềm của khóa luận này có thể ứng dụng vào sử dụng trong hệ thống Bác sĩ gia đình.

# Chương 3: PHÂN TÍCH, THIẾT KẾ, CÀI ĐẶT GIẢI PHÁP

## **3.1. Phân tích yêu cầu**

### 3.1.1. Phân tích yêu cầu chức năng

Phân tích tương đối đầy đủ các chức năng cho hệ thống cây phả hệ và hồ sơ bệnh án bác sĩ gia đình có phân quyền cho nhiều nhóm người sử dụng.

Phân hệ Cây phả hệ:

* Y tá điều dưỡng, bác sĩ:
* Quản lý các thông tin cơ bản của một bệnh nhân khi được lưu vào một node của cây phả hệ, khi nào lưu trữ bệnh nhân vào cây phả hệ, tạo và thay đổi mối quan hệ cho các bệnh nhân, sửa đổi thông tin bệnh nhân trong cây, khi nào được quyền xóa một bệnh nhân trong cây.
* Thống kê số lượng nam nữ, số thành viên đã có gia đình.
* Truy xuất ba đời thế hệ của một bệnh nhân bao gồm: ông bà, cha mẹ, anh chị em, dì chú cậu và con của họ.
* Lấy dữ liệu của bệnh nhân từ service Khám chữa bệnh.
* Bệnh nhân:
  + Xem thông tin cây phả hệ đã được truy xuất.

Phân hệ Hồ sơ bệnh án bác sĩ gia đình:

* Bác sĩ, y tá:
* Quản lý thông tin bệnh nhân: thêm, sửa, xóa thông tin hành chính bệnh nhân.
* Quản lý thông tin bác sĩ: thêm, xóa, sửa thông tin của các bác sĩ khám chữa bệnh.
* Quản lý toa thuốc: thêm, xóa, sửa các toa thuốc được kê cho bệnh nhân.
* Quản lý thông tin nghề nghiệp của bệnh nhân: thêm, xóa, sửa thông tin nghề nghiệp của bệnh nhân.
* Quản lý tài chính (kinh tế) của bệnh nhân: thêm, xóa, thay đổi thu nhập, tình hình kinh tế của bệnh nhân.
* Quản lý thông tin người thân của bệnh nhân: thêm, sửa, xóa thông tin người thân của bệnh nhân.
* Quản lý tiền sử (bệnh sử) của bệnh nhân theo giai đoạn (từ bào thai, sơ sinh 🡪 nhủ nhi (2 đến 12 tháng) 🡪 nhà trẻ mẫu giáo (1 đến 5 tuổi) 🡪 tiểu học (6 đến 11 tuổi) 🡪 tuổi vị thành niên (12 đến 19 tuổi) 🡪 trưởng thành (từ 20 tuổi))
* Quản lý tiền sử gia đình của bệnh nhân: thêm, xóa, sửa các bệnh (tiền sử) mà gia đình bệnh nhân từng mắc.
* Quản lý lịch tiêm ngừa của bệnh nhân: thêm, xóa, sửa lịch tiêm ngừa của bệnh nhân.
* Quản lý các tai nạn thương tích của bệnh nhân: tai nạn trong sinh hoạt, lao động, giao thông,...
* Quản lý thói quen của bệnh nhân.
* Quản lý những vấn đề của bệnh nhân: vấn đề hiện có, vấn đề tầm soát, các vấn đề khác,….
* Quản lý thông tin khám chữa bệnh của bệnh nhân: thêm, sửa, xóa thông tin khám chữa bệnh của bệnh nhân.
* Quản lý các kế hoạch: kế hoạch hóa gia đình, kế hoạch điều trị, chăm sóa dự phòng bệnh,...
* Quản lý các tài liệu: tài liệu chăm sóc sức khỏe, tài liệu đề xuất thông tin giáo dục sức khỏe phù hợp,...
* Quản lý các thông tin: thông tin môi trường địa phương hiện tại, thông tin an toàn tình dục – phụ khoa,….
* Quản lý tâm lý bệnh nhân theo giai đoạn thời gian, biến cố gia đình.
* Quản lý hồ sơ bệnh án bác sĩ gia đình.

### 3.1.2. Phân tích yêu cầu phi chức năng

* Giao diện: đơn giản, dễ sử dụng, phù hợp với các nhân viên, bác sĩ của bệnh viện.
* Bảo mật: Không bị khai thác lỗi.
* Có thể hoạt động trên các trình duyệt cơ bản trên máy tính.
* Chạy nhanh, ổn định.

### 3.1.3. Phân tích quy trình

Phân tích quy trình lấy dữ liệu và thêm bệnh nhân vào service Phả hệ:

Quy trình lấy dữ liệu và thêm bệnh nhân vào cây phả hệ.

#### **Bảng 3. Quy trình thêm bệnh nhân vào service Phả hệ**

| **Trách nhiệm** | **Nội dung quy trình** | **Mô tả, biểu mẫu** |
| --- | --- | --- |
| Cán bộ phụ trách quản lý cây phả hệ | Nhập Id của bệnh nhân vào giao diện phả hệ hoặc tìm dựa theo tên, id, địa chỉ,..  Lấy thông tin của bệnh nhân từ service Khám chữa bệnh | Y tá, điều dưỡng chọn bệnh nhân cần thêm vào cây phả hệ bằng Id của bệnh nhân |
| Cán bộ phụ trách quản lý cây phả hệ |  | Y tá điều dưỡng tìm kiếm chính xác thông tin của bệnh nhân trên phần mềm. |
| Hệ thống | Nếu không tìm thấy hoặc sai | Tìm kiếm thông tin của bệnh nhân đó từ service khám chữa bệnh. |
| Hệ thống |  | Sau khi tìm đúng thông tin của bệnh nhân ở service khám chữa bệnh tiến hành lưu thông tin vào graph database Neo4j. |

Phân tích quy trình tạo quan hệ, xóa và sửa quan hệ của bệnh nhân trong cây phả hệ:

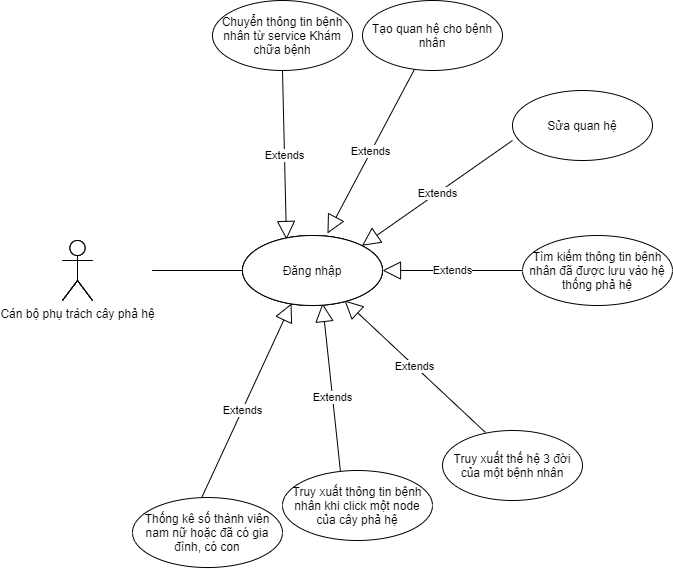
#### **Bảng 4. Quy trình tạo quan hệ, xóa và sửa quan hệ của bệnh nhân trong cây phả hệ.**

| **Trách nhiệm** | **Nội dung quy trình** | **Mô tả, biểu mẫu** |
| --- | --- | --- |
| Y tá, điều dưỡng | Tìm kiếm bệnh nhân trên hệ thống  Lấy thông tin của bệnh nhân từ service Phả hệ | Y tá, điều dưỡng tìm hiểu và xác nhận mối quan hệ của bệnh nhân với các thành viên khác trong gia đình trước khi thêm, xóa hoặc sửa quan hệ. |
| Y tá, điều dưỡng |  | Y tá điều dưỡng tìm kiếm chính xác thông tin của bệnh nhân trên phần mềm. |
| Hệ thống | Nếu không tìm thấy hoặc sai | - Tìm kiếm thông tin của bệnh nhân đó từ service Phả hệ.  -Thông tin chỉ tồn tại khi dữ liệu của bệnh nhân đã được chuyển qua Database Phả hệ. |
| Cán bộ phụ trách quản lý cây phả hệ | Chọn mối quan hệ tương ứng trên giao diện | - Chọn các mối quan hệ tương ứng và người thân muốn liên kết với bệnh nhân đó. |
| Cán bộ phụ trách quản lý cây phả hệ |  | -Thông tin mối quan hệ vào hệ thống cây phả hệ |

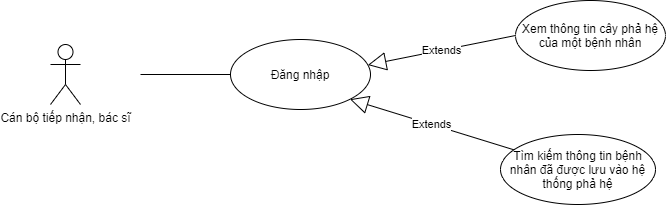
## **3.2. Phân tích thiết kế hệ thống**

### 3.2.1. Hệ thống Cây phả hệ

a) Sơ đồ Use case tổng quát

****

Hình 4. Sơ đồ Use case tổng quát (cán bộ phụ trách cây phả hệ)

****

Hình 5. Sơ đồ Use case tổng quát (cán bộ tiếp nhận và bác sĩ)

b) Đặc tả Use case

Use case Đăng nhập:

| Mã use case | UC01 |
| --- | --- |
| Tên use case | Đăng nhập |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ, cán bộ tiếp nhận, bác sĩ |
| Tiền điều kiện | Không có |
| Hậu điều kiện | Đăng nhập thành công vào hệ thống |
| Điều kiện tối thiểu | Yêu cầu đăng nhập lại |
| Sự kiện chính | 1. Hệ thống hiển thị màn hình đăng nhập  2. Cán bộ nhập thông tin tài khoản và mật khẩu  3. Nhấn “Đăng nhập”  4. Hệ thống chứng thực tài khoản  5. Hệ thống thông báo đăng nhập thành công |
| Ngoại lệ | 4.1. Hệ thống chứng thực thất bại, yêu cầu nhập lại thông tin tài khoản, mật khẩu. |

Use case Chuyển thông tin bệnh nhân từ service Khám chữa bệnh:

| Mã use case | UC02 |
| --- | --- |
| Tên use case | Chuyển thông tin bệnh nhân từ service Khám chữa bệnh |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ |
| Tiền điều kiện | Thông tin bệnh nhân muốn chuyển đã tồn tại trong service Khám chữa bệnh |
| Hậu điều kiện | Thông tin của bệnh nhân được chuyển thành công qua service Phả hệ |
| Điều kiện tối thiểu | Yêu cầu chọn lại đúng bệnh nhân |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện chuyển dữ liệu  2. Cán bộ nhập Id bệnh nhân hoặc tìm bệnh nhân theo các thông tin khác  3. Chọn bệnh nhân cần chuyển.  4. Nhấn “Chuyển dữ liệu”.  5. Hệ thống thông báo dữ liệu được chuyển thành công. |
| Ngoại lệ | 5.1. Hệ thống chuyển dữ liệu thất bại, do thông tin bệnh nhân đã tồn tại trong service Phả hệ.  2.1. Không tìm thấy thông tin bệnh nhân tương ứng, cán bộ xác nhận và tìm lại. |

Use case Tạo quan hệ cho bệnh nhân:

| Mã use case | UC03 |
| --- | --- |
| Tên use case | Tạo quan hệ cho bệnh nhân |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ |
| Tiền điều kiện | Thông tin bệnh nhân muốn chuyển đã tồn tại trong service Phả hệ |
| Hậu điều kiện | Tạo quan hệ thành công |
| Điều kiện tối thiểu | Yêu cầu chọn lại bệnh nhân vào mối quan hệ cần thiết |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện tạo quan hệ  2. Cán bộ chọn 2 bệnh nhân cần tạo quan hệ  3. Chọn mối quan hệ và loại quan hệ muốn tạo  4. Nhấn “Tạo quan hệ”.  5. Hệ thống thông báo quan hệ được tạo thành công. |
| Ngoại lệ | 5.1. Hệ thống tạo quan hệ thất bại, do giữa 2 bệnh nhân đã tồn tại sẵn một mối quan hệ.  2.1. Không tìm thấy thông tin bệnh nhân tương ứng, cán bộ xác nhận và tìm lại. |

Use case Sửa quan hệ cho bệnh nhân:

| Mã use case | UC04 |
| --- | --- |
| Tên use case | Sửa quan hệ cho bệnh nhân |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ |
| Tiền điều kiện | Thông tin bệnh nhân muốn chuyển đã tồn tại trong service Phả hệ |
| Hậu điều kiện | Sửa quan hệ thành công |
| Điều kiện tối thiểu | Yêu cầu chọn lại bệnh nhân và mối quan hệ cần thiết |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện tạo quan hệ  2. Cán bộ chọn 2 bệnh nhân cần sửa quan hệ  3. Chọn mối quan hệ và loại quan hệ muốn sửa  4. Nhấn “Sửa quan hệ”.  5. Hệ thống thông báo quan hệ được thay đổi thành công. |
| Ngoại lệ | 5.1. Hệ thống sửa quan hệ thất bại, do giữa 2 bệnh nhân chưa tồn tại sẵn một mối quan hệ.  2.1. Không tìm thấy thông tin bệnh nhân tương ứng, cán bộ xác nhận và tìm lại. |

Use case Xóa quan hệ cho bệnh nhân:

| Mã use case | UC05 |
| --- | --- |
| Tên use case | Xóa quan hệ cho bệnh nhân |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ |
| Tiền điều kiện | Thông tin và mối quan hệ 2 bệnh nhân muốn xóa đã tồn tại trong database Neo4j |
| Hậu điều kiện | Xóa quan hệ thành công |
| Điều kiện tối thiểu | Yêu cầu chọn lại bệnh nhân cần thiết |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện tạo quan hệ  2. Cán bộ chọn 2 bệnh nhân cần xóa quan hệ  3. Nhấn “Xoá quan hệ”.  4. Hệ thống thông báo quan hệ được xóa thành công. |
| Ngoại lệ | 4.1. Hệ thống tạo quan hệ thất bại, do giữa 2 bệnh nhân chưa tồn tại sẵn một mối quan hệ.  2.1. Không tìm thấy thông tin bệnh nhân tương ứng, cán bộ xác nhận và tìm lại. |

Use case Tìm kiếm thông tin bệnh nhân đã được lưu vào hệ thống Phả hệ:

| Mã use case | UC06 |
| --- | --- |
| Tên use case | Tìm kiếm thông tin bệnh nhân đã được lưu vào hệ thống Phả hệ |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ, cán bộ tiếp nhận, bác sĩ |
| Tiền điều kiện | Thông tin bệnh nhân muốn tìm kiếm đã tồn tại trong service Phả hệ |
| Hậu điều kiện | Tìm kiếm bệnh nhân thành công |
| Điều kiện tối thiểu | Yêu cầu nhập lại các tiêu chí tìm kiếm |
| Sự kiện chính | 1. Hệ thống hiển thị danh sách bệnh nhân  2. Cán bộ nhập vào tiêu chí tìm kiếm  3. Nhấn “Tìm kiếm”.  4. Hệ thống trả về thông tin bệnh nhân tương ứng. |
| Ngoại lệ | 4.1. Hệ thống tìm kiếm không có kết quả, cán bộ xác nhận và tìm kiếm lại. |

Use case Truy xuất thế hệ ba đời của một bệnh nhân:

| Mã use case | UC07 |
| --- | --- |
| Tên use case | Truy xuất thế hệ ba đời của một bệnh nhân |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ, cán bộ tiếp nhận, bác sĩ. |
| Tiền điều kiện | Thông tin bệnh nhân muốn tìm kiếm đã tồn tại trong service Phả hệ |
| Hậu điều kiện | Tìm kiếm bệnh nhân thành công |
| Điều kiện tối thiểu | Yêu cầu nhập lại các tiêu chí tìm kiếm |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện phả hệ  2. Cán bộ nhập tìm kiếm bệnh nhân cần thiết  3. Cán bộ nháy đúp vào thông tin bệnh nhân sau khi tìm thấy để lấy Id và chọn nhánh thế hệ muốn truy xuất “Theo Cha” hoặc “Theo Mẹ”  4. Nhấn vào nút “Lấy phả hệ”.  5. Hệ thống trả về cây phả hệ ba đời của bệnh nhân đó. |
| Ngoại lệ | 3.1. Cán bộ chưa chọn nhánh thế hệ muốn truy xuất hoặc chưa chọn Nhánh Phả hệ, hệ thống báo lỗi.  5.1. Hệ thống trả về rỗng, do thông tin người thân của bệnh nhân đó chưa được cập nhật qua service Phả hệ. |

Use case Truy xuất thông tin bệnh nhân khi click vào một Node của cây Phả hệ:

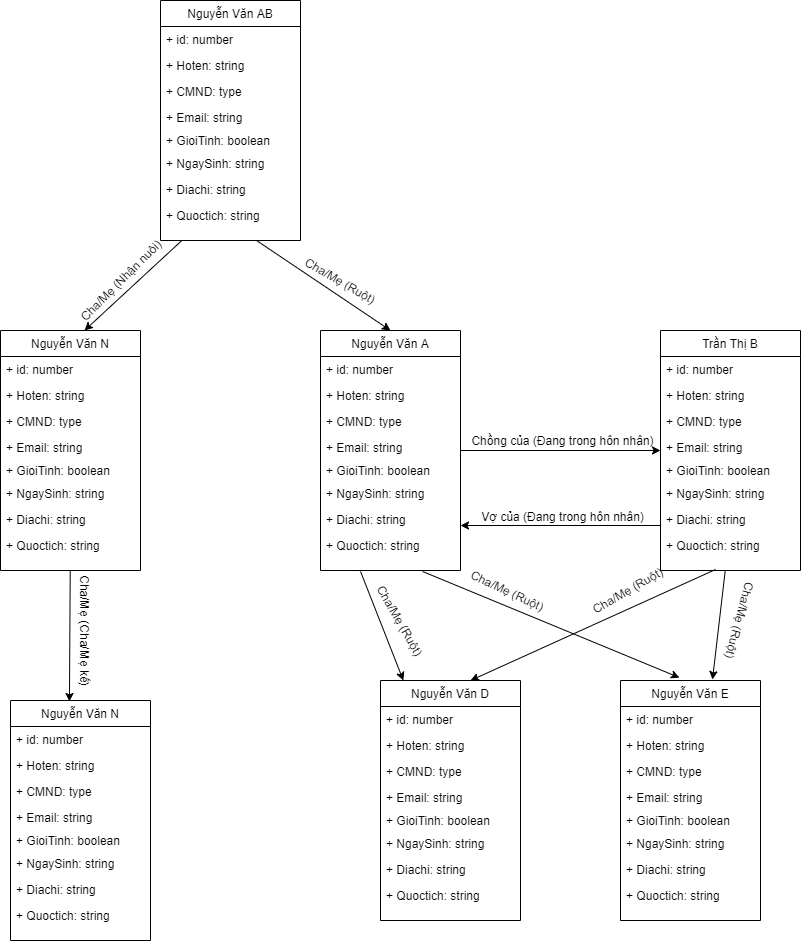
| Mã use case | UC08 |
| --- | --- |
| Tên use case | Truy xuất thế hệ ba đời của một bệnh nhân |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ, cán bộ tiếp nhận, bác sĩ. |
| Tiền điều kiện | Cây phả hệ đã được hiển thị thành công |
| Hậu điều kiện | Truy xuất thông tin thành công |
| Điều kiện tối thiểu | Không có |
| Sự kiện chính | 1. Hệ thống hiển thị cây phả hệ ba đời  2. Cán bộ chọn bệnh nhân cần xem chi tiết thông tin và nháy chuột.  3. Hệ thống hiển thị thông tin của bệnh nhân đó. |
| Ngoại lệ | 1.1. Cây phả hệ chưa được hiển thị hoặc hiển thị rỗng. |

Use case Thống kê số thành viên nam nữ hoặc đã có gia đình có con:

| Mã use case | UC09 |
| --- | --- |
| Tên use case | Thống kê số thành viên nam nữ hoặc đã có gia đình có con … |
| Tác nhân chính | Cán bộ phụ trách cây phả hệ |
| Tiền điều kiện | Thông tin của gia đình bệnh nhân đã được lưu đầy đủ. |
| Hậu điều kiện | Thống kê thành công |
| Điều kiện tối thiểu | Không có |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện phả hệ  2. Cán bộ nhập tìm kiếm bệnh nhân cần thiết  3. Nhấn vào nút “Thống kê”.  4. Hệ thống hiển thị giao diện thống kê  5. Cán bộ chọn tiêu chí muốn thống kê và nhấn “Submit”  6. Hệ thống trả về các biểu đồ thống kê tiêu chí tương ứng. |
| Ngoại lệ | 6.1. Biểu đồ trả về rỗng do thiếu thông tin. |

Mô hình Cơ sở dữ liệu Phả hệ (Neo4j):

Do cơ sở dữ liệu là Graph Database nên đặc trưng các Node (Lớp) sẽ có cùng một bộ thuộc tính với dữ liệu khác nhau, tên Node (className) sẽ là tên của bệnh nhân và các mối quan hệ một chiều tự định nghĩa, kiểu dữ liệu của từng thuộc tính sẽ thoải mái hơn Oracle, MySql do không bị ràng buộc về giới hạn số kí tự, độ lớn của một số…. Ví dụ như hình sẽ là sơ đồ khái quát trong Neo4J:



Hình 6. Sơ đồ lớp (Graph Database)

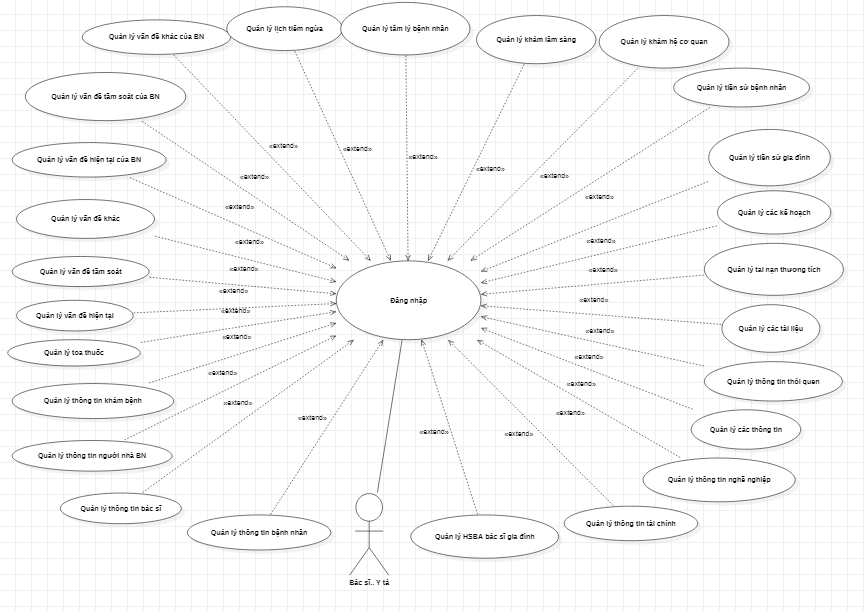
Sơ đồ trên chỉ thể hiện cho một gia đình ba thế hệ, database sẽ lưu trữ nhiều gia đình với cấu trúc tương tự, có thể nhiều hơn ba thế hệ.

Kiểu dữ liệu cho mỗi Node bệnh nhân (Neo4j):

| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| --- | --- | --- | --- |
| 1 | ID | Kiểu số nguyên dương | Id của bệnh nhân |
| 2 | CHI\_CO\_NAM\_SINH | Kiểu số nguyên dương | Trạng thái thông tin bệnh nhân chỉ có năm sinh:\n1: Chỉ có năm sinh.\n0: Đầy đủ ngày tháng năm sinh. |
| 3 | CMND | Kiểu chuỗi ký tự | Thông tin chứng minh nhân dân của bệnh nhân |
| 4 | EMAIL | Kiểu chuỗi ký tự | Thông tin địa chỉ email của bệnh nhân |
| 5 | ENABLED | Kiểu Boolean | Trạng thái của 1 dòng thông tin bệnh nhân\n\n1: Có hiệu lực.\n\n0: Không có hiệu lực. |
| 6 | GIOI\_TINH | Kiểu Boolean | Trạng thái giới tính của bệnh nhân:\n\n1: Nam\n.\n0: Nữ. |
| 7 | KHANG\_THE | Kiểu chuỗi ký tự | Thông tin kháng thể của bệnh nhân |
| 8 | MA\_SO\_THUE | Kiểu chuỗi ký tự | Thông tin mã số thuế của bệnh nhân |
| 9 | NGAY\_CAP\_CMND | Kiểu date | Ngày cấp chứng minh nhân dân của bệnh nhân |
| 10 | NGAY\_SINH | Kiểu date | Thông tin ngày sinh của bệnh nhân |
| 11 | NHOM\_MAU | Kiểu chuỗi ký tự | Thông tin nhóm máu của bệnh nhân |
| 12 | NOI\_CAP\_CMND | Kiểu chuỗi ký tự | Thông tin nơi cấp chứng minh nhân dân |
| 13 | NOI\_LAM\_VIEC | Kiểu chuỗi ký tự | Thông tin nơi làm việc của bệnh nhân |
| 14 | PHONE | Kiểu chuỗi ký tự | Thông tin số điện thoại của bệnh nhân |
| 15 | TEN | Kiểu chuỗi ký tự | Tên bệnh nhân |
| 16 | SO\_NHA\_XOM | Kiểu chuỗi ký tự | Số nhà của bệnh nhân |
| 17 | AP\_THON | Kiểu chuỗi ký tự | Thông tin ấp, thôn, xóm của bệnh nhân |
| 18 | DIA\_PHUONG\_ID | Kiểu số nguyên dương | Mã danh mục địa phương |
| 19 | DIA\_CHI\_THUONG\_TRU | VARCHAR2(500 BYTE) | Thông tin địa chỉ thường trú của bệnh nhân |
| 20 | DAN\_TOC\_ID | Kiểu số nguyên dương | Mã dân tộc của bệnh nhân |
| 21 | NGHE\_NGHIEP\_ID | Kiểu số nguyên dương | Mã nghề nghiệp của bệnh nhân |
| 22 | QUOC\_TICH\_ID | Kiểu số nguyên dương | Mã quốc tịch của bệnh nhân |
| 23 | PHUONG\_XA\_ID | Kiểu số nguyên dương | Mã phường xã của bệnh nhân |
| 24 | QUAN\_HUYEN\_ID | Kiểu số nguyên dương | Mã quận huyện của bệnh nhân |
| 25 | TINH\_THANH\_PHO\_ID | Kiểu số nguyên dương | Mã tỉnh thành phố của bệnh nhân |

### 3.2.2. Hệ thống Hồ sơ bệnh án bác sĩ gia đình

1. Sơ đồ use case tổng quát

****

Hình 7. Sơ đồ use case tổng quát

1. **Đặt tả use case**

* **Use case Quản lý thông tin bệnh nhân**

| Mã use case | UC10 |
| --- | --- |
| Tên use case | Quản lý thông tin bệnh nhân |
| Tác nhân chính | Bác sĩ, y tá |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Hậu điều kiện | Thông tin của bệnh nhân đã được lưu vào hệ thống |
| Điều kiện tối thiểu | Loại bỏ các thông tin bệnh nhân đã nhập và quay về các bước trước |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện hồ sơ bệnh án bác sĩ gia đình  2. Bác sĩ/Y tá chọn chức năng “Thông tin bệnh nhân” trên giao diện  3. Hệ thống hiển thị giao diện thông tin bệnh nhân  4. Bác sĩ/Y tá nhấn vào nút chức năng muốn tực hiện (thêm, sửa, xóa)  5. Bác sĩ/Y tá xác nhận lại thông tin bệnh nhân và cập nhật dữ liệu  6. Hệ thống kiểm tra, xử lý và thông tin của bệnh nhân đã được lưu trữ vào hệ thống. |
| Ngoại lệ | 6.1.Hệ thống kiểm tra thông tin thất bại, quay lại giao diện thông tin bệnh nhân. |

* **Use case Quản lý danh mục vấn đề hiện tại**

| Mã use case | UC11 |
| --- | --- |
| Tên use case | Quản lý danh mục vấn đề hiện tại |
| Tác nhân chính | Bác sĩ, y tá |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Hậu điều kiện | Vấn đề hiện tại đã được lưu vào hệ thống |
| Điều kiện tối thiểu | Loại bỏ các vấn đề hiện tại đã nhập và quay về các bước trước |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện hồ sơ bệnh án bác sĩ gia đình  2. Bác sĩ/Y tá chọn chức năng “Danh mục vấn đề hiện tại” trên giao diện  3. Hệ thống hiển thị giao diện danh mục vấn đề hiện tại  4. Bác sĩ/Y tá nhấn vào nút chức năng muốn thực hiện (thêm, sửa, xóa)  5. Bác sĩ/Y tá xác nhận lại tên vấn đề hiện tại và cập nhật dữ liệu  6. Hệ thống kiểm tra, xử lý và tên vấn đề hiện tại đã được lưu trữ vào hệ thống. |
| Ngoại lệ | 6.1.Hệ thống kiểm tra thông tin thất bại, quay lại giao diện danh mục vấn đề hiện tại. |

* **Use case Quản lý danh mục vấn đề tầm soát**

| Mã use case | UC12 |
| --- | --- |
| Tên use case | Quản lý danh mục vấn đề tầm soát |
| Tác nhân chính | Bác sĩ, y tá |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Hậu điều kiện | Vấn đề tầm soát đã được lưu trữ vào hệ thống |
| Điều kiện tối thiểu | Loại bỏ các vấn đề tầm soát đã nhập và quay về các bước trước |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện hồ sơ bệnh án bác sĩ gia đình  2. Bác sĩ/Y tá chọn chức năng “Danh mục vấn đề tầm soát” trên giao diện  3. Hệ thống hiển thị giao diện danh mục vấn đề tầm soát  4. Bác sĩ/Y tá nhấn vào nút chức năng muốn tực hiện (thêm, sửa, xóa)  5. Bác sĩ/Y tá xác nhận lại tên vấn đề tầm soát và cập nhật dữ liệu  6. Hệ thống kiểm tra, xử lý và tên vấn đề tầm soát đã được lưu trữ vào hệ thống. |
| Ngoại lệ | 6.1.Hệ thống kiểm tra thông tin thất bại, quay lại giao diện danh mục vấn đề tầm soát. |

* **Use case Quản lý thông tin khám bệnh**

| Mã use case | UC13 |
| --- | --- |
| Tên use case | Quản lý thông tin khám bệnh |
| Tác nhân chính | Bác sĩ, y tá |
| Tiền điều kiện | Đăng nhập thành công vào hệ thống |
| Hậu điều kiện | Thông tin khám bệnh của bệnh nhân đã được lưu vào hệ thống |
| Điều kiện tối thiểu | Loại bỏ các thông tin khám bệnh của bệnh nhân đã nhập và quay về các bước trước |
| Sự kiện chính | 1. Hệ thống hiển thị giao diện hồ sơ bệnh án bác sĩ gia đình  2. Bác sĩ/Y tá chọn chức năng “Thông tin khám bệnh” trên giao diện  3. Hệ thống hiển thị giao diện thông tin khám bệnh  4. Bác sĩ/Y tá nhấn vào nút chức năng muốn tực hiện (thêm, sửa, xóa)  5. Bác sĩ/Y tá xác nhận lại thông tin khám bệnh của bệnh nhân và cập nhật dữ liệu  6. Hệ thống kiểm tra, xử lý và thông tin khám bệnh của bệnh nhân đã được lưu trữ vào hệ thống. |
| Ngoại lệ | 6.1.Hệ thống kiểm tra thông tin thất bại, quay lại giao diện thông tin khám bệnh. |

1. **Thiết kế cơ sở dữ liệu**

* **Thiết kế bảng**
* **BENHNHAN**

#### **Bảng 5. Bệnh nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id của bệnh nhân |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | hoTenBN | Kiểu chuỗi ký tự | Họ tên của bệnh nhân |
| 4 | Tuoi | Kiểu số nguyên | Tuổi của bệnh nhân |
| 5 | diaChi | Kiểu chuỗi ký tự | Địa chỉ của bệnh nhân |
| 6 | Email | Kiểu chuỗi ký tự | Email của bệnh nhân |
| 7 | soDienThoai | Kiểu chuỗi ký tự | Số điện thoại của bệnh nhân |
| 8 | maHoSo | Kiểu số nguyên | Mã hồ sơ của bệnh nhân |

* **BACSI**

#### **Bảng 6. Bác sĩ**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý ngĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id của bác sĩ |
| 2 | maBacSi | Kiểu số nguyên | Mã bác sĩ |
| 3 | hoTenBS | Kiểu chuỗi ký tự | Họ tên bác sĩ |
| 4 | gioiTinh | Kiểu chuỗi ký tự | Giới tính |
| 5 | namSinh | Kiểu số nguyên | Năm sinh |
| 6 | hocViHocham | Kiểu chuỗi ký tự | Học vị học hàm |
| 7 | chuyenKhoa | Kiểu chuỗi ký tự | Chuyên khoa của bác sĩ |
| 8 | noiLamViec | Kiểu chuỗi ký tự | Nơi làm việc |
| 9 | Email | Kiểu chuỗi ký tự | Email |
| 10 | khamChuyenKhoa | Kiểu chuỗi ký tự | Khám chuyên khoa |

* **THONGTINKHAMBENH**

#### **Bảng 7. Thông tin khám bệnh**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id thông tin khám bệnh |
| 2 | soTT | Kiểu số nguyên | Số thứ tự |
| 3 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân khám bệnh |
| 4 | maBacSi | Kiểu chuỗi ký tự | Mã bác sĩ khám bệnh |
| 5 | deNghiKham | Kiểu chuỗi ký tự | Đề nghị khám |
| 6 | ngayKhamBenh | Kiểu date | Ngày khám bệnh |
| 7 | noiKhamBenh | Kiểu chuỗi ký tự | Nơi khám bệnh |
| 8 | chanDoanBenh | Kiểu chuỗi ký tự | Chẩn đoán bệnh |
| 9 | huongDieuTri | Kiểu chuỗi ký tự | Hướng điều trị |
| 10 | ngayTaiKham | Kiểu date | Ngày tái khám |
| 11 | lanKham | Kiểu số nguyên | Lần khám |
| 12 | baoHiemYTe | Kiểu chuỗi ký tự | Bảo hiểm y tế |
| 13 | loiKhuyenCuaBS | Kiểu chuỗi ky tự | Lời khuyên của bác sĩ |

* **LICHTIEMNGUA**

#### **Bảng 8. Lịch tiêm ngừa**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id lịch tiêm ngừa |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenVacXin | Kiểu chuỗi ký tự | Tên vacxin |
| 4 | soSinh | Kiểu chuỗi ký tự | Giai đoạn sơ sinh |
| 5 | motThang | Kiểu chuỗi ký tự | Một tháng |
| 6 | haiThang | Kiểu chuỗi ký tự | Hai tháng |
| 7 | baThang | Kiểu chuỗi ký tự | Ba tháng |
| 8 | bonThang | Kiểu chuỗi ký tự | Bốn tháng |
| 9 | chinThang | Kiểu chuỗi ký tự | Chín tháng |
| 10 | muoiHaiThang | Kiểu chuỗi ký tự | Mười hai tháng |
| 11 | muoiLamThang | Kiểu chuỗi ký tự | Mười lăm tháng |
| 12 | muoiTamThang | Kiểu chuỗi ký tự | Mười lăm tháng |
| 13 | tu4den6tuoi | Kiểu chuỗi ký tự | Từ 4 đến 6 tuổi |
| 14 | tu11den12tuoi | Kiểu chuỗi ký tự | Từ 11 đến 12 tuổi |

* **TOATHUOC**

#### **Bảng 9. Toa thuốc**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | Id | Kiểu chuỗi ký tự | Id toa thuốc |
| 2 | soToaThuoc | Kiểu chuỗi ký tự | Số toa thuốc |
| 3 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 4 | maBacSi | Kiểu số nguyên | Mã bác sĩ |
| 5 | tenThuoc | Kiểu chuỗi ký tự | Tên thuốc |
| 6 | ngayDung | Kiểu chuỗi ký tự | Ngày dùng |
| 7 | ngayUong | Kiểu chuỗi ký tự | Ngày uống |

* **THONGTIN**

#### **Bảng 10. Thông tin**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id thông tin |
| 2 | maThongTin | Kiểu chuỗi ký tự | Mã thông tin |
| 3 | tenThongTin | Kiểu chuỗi ký tự | Tên thông tin |
| 4 | loaiThongTin | Kiểu chuỗi ký tự | Loại thông tin |
| 5 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |

* **KEHOACH**

#### **Bảng 11. Kế hoạch**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | Id | Kiểu chuỗi ký tự | Id kế hoạch |
| 2 | maKH | Kiểu chuỗi ký tự | Mã kế hoạch |
| 3 | tenKH | Kiểu chuỗi ký tự | Tên kế hoạch |
| 4 | loaiKH | Kiểu chuỗi ký tự | Loại kế hoạch |
| 5 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |

* **VANDE**

#### **Bảng 12. Vấn đề**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id vấn đề hiện tại |
| 2 | tenVanDe | Kiểu chuỗi ký tự | Tên vấn đề hiện tại |

* **KHAMHECOQUAN**

#### **Bảng 13. Khám hệ cơ quan**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id khám hệ cơ quan |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenVanDe | Kiểu chuỗi ký tự | Tên vấn đề |
| 4 | moTa | Kiểu chuỗi ký tự | Mô tả |

* **VANDEBENHNHAN**

#### **Bảng 14. Vấn đề bệnh nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id vấn đề hiện tại của bệnh nhân |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenVanDe | Kiểu chuỗi ký tự | Tên vấn đề hiện tại |
| 4 | tinhTrang | Kiểu chuỗi ký tự | Tình trạng |
| 5 | phanLoai | Kiểu chuỗi ký tự | Phân loại |
| 6 | ghiChu | Kiểu chuỗi ký tự | Ghi chú |

* **NGUOINHABENHNHAN**

#### **Bảng 15. Người nhà bệnh nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id người thân của bệnh nhân |
| 2 | hoTenNN | Kiểu chuỗi ký tự | Họ tên người thân bệnh nhân |
| 3 | gioiTinh | Kiểu chuỗi ký tự | Giới tính |
| 4 | diaChi | Kiểu chuỗi ký tự | Địa chỉ |
| 5 | soDienThoai | Kiểu chuỗi ký tự | Số điện thoại |
| 6 | quanHe | Kiểu chuỗi ký tự | Quan hệ với bệnh nhân |
| 7 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |

* **VANDETAMSOAT**

#### **Bảng 16. Vấn đề tầm soát**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id vấn đề tầm soát |
| 2 | tenVanDeTamSoat | Kiểu chuỗi ký tự | Tên vấn đề tầm soát |

* **TIENSUGIADINH**

#### **Bảng 17. Tiền sử gia đình**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id tiền sử gia đình bệnh nhân |
| 2 | tenVanDeTamSoat | Kiểu chuỗi ký tự | Tên vấn dề tầm soát |
| 3 | hoTenNN | Kiểu chuỗi ký tự | Họ tên người nhà bệnh nhân |
| 4 | tinhTrang | Kiểu chuỗi ký tự | Tình trạng |
| 5 | ghiChu | Kiểu chuỗi ký tự | Ghi chú |
| 6 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |

* **VANDETAMSOATBENHNHAN**

#### **Bảng 18. Vấn đề tầm soát bệnh nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id vấn đề tầm soát của bệnh nhân |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenVanDeTamSoat | Kiểu chuỗi ký tự | Tên vấn đề tầm soát |
| 4 | ketQua | Kiểu chuỗi ký tự | Kết quả |

* **KHAMLAMSANG**

#### **Bảng 19. Khám lâm sàng**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id khám lâm sàng |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | Mach | Kiểu số nguyên | Mạch |
| 4 | huyetAp | Kiểu chuỗi ký tự | Huyết áp |
| 5 | canNang | Kiểu double | Cân nặng |
| 6 | chieuCao | Kiểu chuỗi ký tự | Chiều cao |
| 7 | bMI | Kiểu double | BMI |

* **TAMLYBENHNHAN**

#### **Bảng 20. Tâm lý bệnh nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id tâm lý bệnh nhân |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tinhTrang | Kiểu chuỗi ký tự | Tình trạng |
| 4 | nguyenNhan | Kiểu chuỗi ký tự | Nguyên nhân |
| 5 | thoiGian | Kiểu chuỗi ký tự | Thời gian |
| 6 | cachDieuTri | Kiểu chuỗi ký tự | Cách điều trị |

* **NGHENGHIEP**

#### **Bảng 21. Nghề nghiệp**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id nghề nghiệp |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | maNgheNghiep | Kiểu chuỗi ký tự | Mã nghề nghiệp |
| 4 | tenNgheNghiep | Kiểu chuỗi ký tự | Tên nghề nghiệp |
| 5 | noiLamViec | Kiểu chuỗi ký tự | Nơi làm việc |
| 6 | tinhChatCongViec | Kiểu chuỗi ký tự | Tính chất công việc |
| 7 | thoiGian | Kiểu chuỗi ký tự | Thời gian |

* **KINHTE**

**Bảng 18. Kinh tế**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id kinh tế |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | mucThuNhap | Kiểu chuỗi ký tự | Mức thu nhập |
| 4 | mucSong | Kiểu chuỗi ký tự | Mức sống |
| 5 | giaiDoan | Kiểu chuỗi ký tự | Giai đoạn |

* **THOIQUEN**

#### **Bảng 22. Thói quen**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi dữ liệu | Id thói quen |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenThoiQuen | Kiểu chuỗi dữ liệu | Tên thói quen |
| 4 | thoiGian | Kiểu chuỗi dữ liệu | Thời gian |

* **TAILIEU**

#### **Bảng 23. Tài liệu**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi dữ liệu | Id tài liệu |
| 2 | maTaiLieu | Kiểu chuỗi dữ liệu | Mã tài liệu |
| 3 | tenTaiLieu | Kiểu chuỗi dữ liệu | Tên tài liệu |
| 4 | loaiTaiLieu | Kiểu chuỗi dữ liệu | Loại tài liệu |
| 5 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |

* **TAINAN**

#### **Bảng 24. Tai nạn**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi dữ liệu | Id tai nạn |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenTaiNan | Kiểu chuỗi dữ liệu | Tên tai nạn |
| 4 | nguyeNhan | Kiểu chuỗi dữ liệu | Nguyên nhân |
| 5 | thoiGianXayRa | Kiểu chuỗi dữ liệu | Thời gian xảy ra |
| 6 | tinhTrang | Kiểu chuỗi dữ liệu | Tình trạng |

* **DANHMUCVANDEKHAC**

#### **Bảng 25. Danh mục vấn đề khác**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi dữ liệu | Id vấn đề khác |
| 2 | TenVanDe | Kiểu chuỗi dữ liệu | Tên vấn đề khác |
| 3 | kieuDuLieu | Kiểu số nguyên | Kiểu dữ liệu (1: number, 2: text, 3: cả 2) |
| 4 | ChuKy | Kiểu chuỗi ký tự | Chu kỳ |
| 5 | donViTinh | Kiểu chuỗi ký tự | Đơn vị tính |

* **TIENSUBENHNHAN**

#### **Bảng 26. Tiền sử bệnh nhân**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | Id | Kiểu chuỗi ký tự | Id tiền sử bệnh nhân |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenVanDe | Kiểu chuỗi ký tự | Tên vấn đề khác |
| 4 | giaiDoanBenh | Kiểu chuỗi ký tự | Giai đoạn bệnh |
| 5 | thoiGianBenh | Kiểu chuỗi ký tự | Thời gian bệnh |

* **DMVANDEKHACCUABENHNHAN**

#### **Bảng 27. Danh mục vấn đề khác của bệnh nhân**

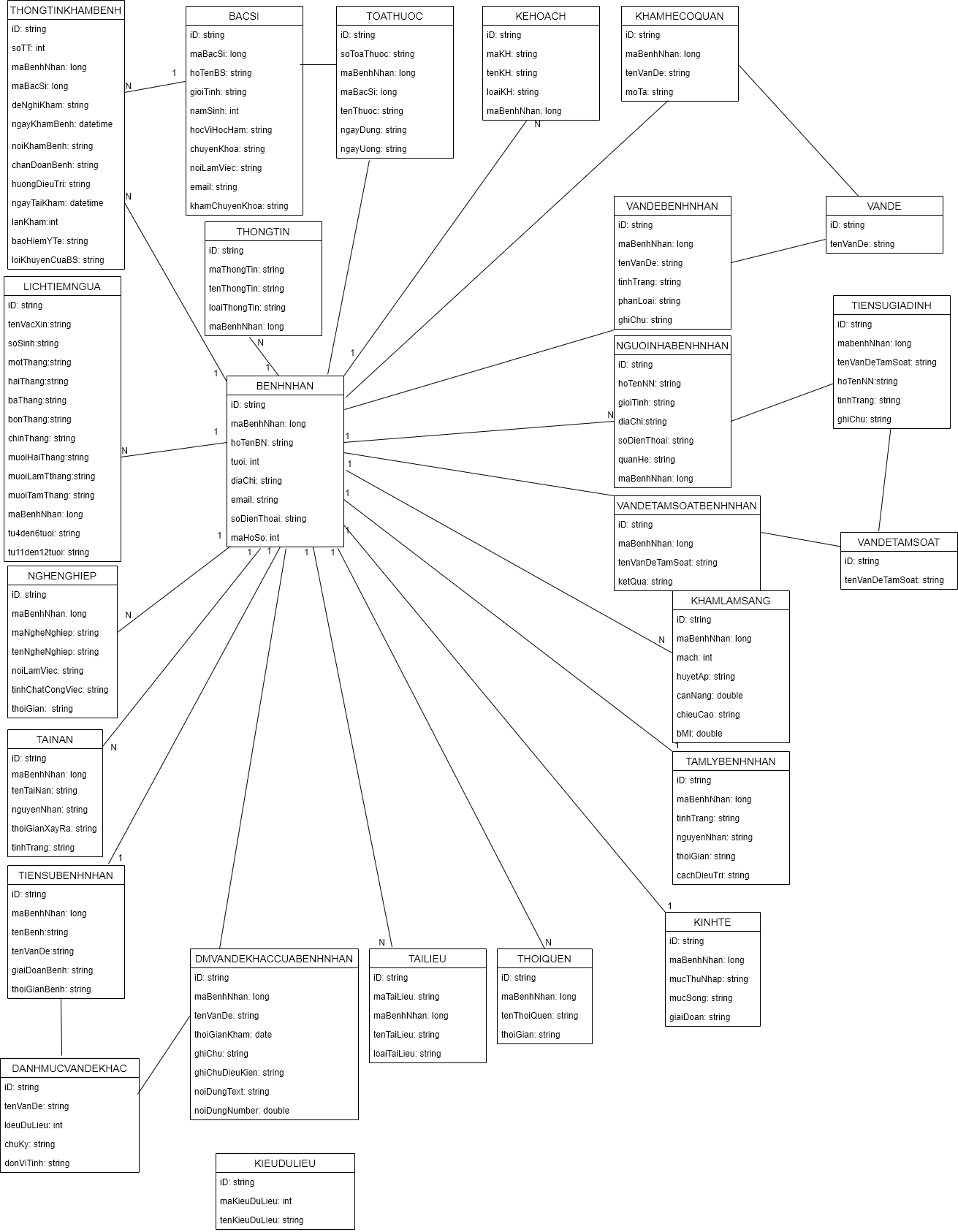
|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id vấn đề khác của bệnh nhân |
| 2 | maBenhNhan | Kiểu số nguyên | Mã bệnh nhân |
| 3 | tenVanDe | Kiểu chuỗi ký tự | Tên vấn đề khác |
| 4 | thoiGianKham | Kiểu date | Ngày khám bệnh |
| 5 | ghiChu | Kiểu chuỗi ký tự | Ghi chú |
| 6 | ghiChuDieuKien | Kiểu chuỗi ký tự | Ghi chú điều kiện |
| 7 | noiDungText | Kiểu chuỗi ký tự | Nội dung kiểu text |
| 8 | noiDungNumber | Kiểu double | Nội dung kiểu number |

* **KIEUDULIEU**

#### **Bảng 28.Kiểu dữ liệu**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | iD | Kiểu chuỗi ký tự | Id kiểu dữ liệu |
| 2 | maKieuDuLieu | Kiểu số nguyên | Mã kiểu dữ liệu |
| 3 | tenKieuDuLieu | Kiểu chuỗi ký tự | Tên kiểu đữ liệu |

* **Sơ đồ cơ sở dữ liệu**

****

Hình 8. Sơ đồ cơ sở dữ liệu hồ sơ bệnh án bác sĩ gia đình

## **3.3. Cài đặt giải pháp**

Hiện trạng hệ thống: Hệ thống phải được tích hợp vào hệ thống có các dịch vụ khác, sử dụng một số thông tin của các dịch vụ đã có. Hệ thống sẽ gồm một số dịch vụ như sau:

Dịch vụ 1: Dịch vụ quản lý về nhân viên, phân quyền người dùng (KeyCloak)

Dịch vụ 2: Dịch vụ về chứng thực đăng nhập (Consul)

Dịch vụ 3: Dịch vụ Cây phả hệ

Dịch vụ 4: Dịch vụ Hồ sợ bác sĩ gia đình

Dịch vụ 5: Dịch vụ Gateway

### 3.3.1. Xây dựng cấu trúc Microservice cho hệ thống Cây phả hệ và hệ thống Hồ sơ bác sĩ gia đình

Dựa vào kiến trúc chung của hệ thống Bác sĩ gia đình, kiến thức đã tiếp thu được và các yêu cầu để xây dựng hệ thống tôi xin đề xuất kiến trúc Microservice của hệ thống Cây phả hệ và Hồ sơ bác sĩ gia đình như sau:

Dịch vụ 1: Dịch vụ chứng thực đăng nhập (Consul).

Dịch vụ 2: Dịch vụ quản lý về nhân viên, phân quyền người dùng (KeyCloak)

Dịch vụ 2: Dịch vụ để hiện thị giao diện cho hệ thống (Microservice gateway).

Dịch vụ 3: Dịch vụ của hệ thống Cây phả hệ (Microservice application).

Dịch vụ 4: Dịch vụ hệ thống Hồ sơ bác sĩ gia đình (Microservice application).

Do 2 hệ thống Cây phả hệ và hồ sơ bác sĩ gia đình sử dụng 2 Cơ sở dữ liệu khác nhau và cho những mục đích khác nhau nên chúng tôi sẽ tách ra thành hai Service riêng biệt với công dụng như sau:

* Service Cây phả hệ: liên kết với cơ sở dữ liệu Neo4J bằng chuỗi kết nối đặc trưng, ánh xạ các dữ liệu các Node bệnh nhân ở Neo4J, quản lý xây dựng các câu truy vấn, các hàm xử lý yêu cầu từ Gateway thông qua Repository lưu về Cơ sở dữ liệu.
* Service Hồ sơ Bác sĩ gia đình: liên kết với cơ sở dữ liệu MongoDB để lưu trữ thông tin hành chính của bệnh nhân, thông tin người thân của bệnh nhân, thông tin khám bệnh, tiền sử bệnh nhân, tiền sử gia đình bệnh nhân, thông tin tài chính, thói quen, nghề nghiêp,…liên kết với thư viện JasperReport để tạo báo cáo (report) và quản lý hồ sơ bệnh án bác sĩ gia đình.

**Xây dựng kiến trúc Microservice cho ứng dụng:**

Từ cấu trúc bên trên, chúng tôi có thể xây dựng được hệ thống Cây phả hệ và Hồ sơ bác sĩ gia đình theo theo một phần kiến trúc của Microservice để đảm bảo hệ thống hoạt động hiệu quả.

Bước 1: Mở Command Prompt trên mát tính bằng phím tắt Windows + R, nhập cmd và nhấn Enter.

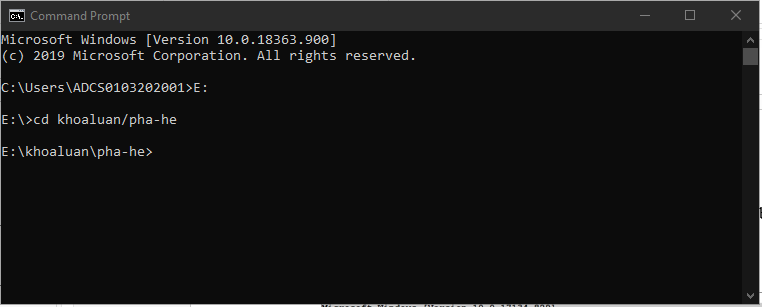


Hình 9. Xây dựng cấu trúc cho hệ thống

Bước 2: Tạo thư mục mới và di chuyển vào thư mục đó bằng cách gõ:

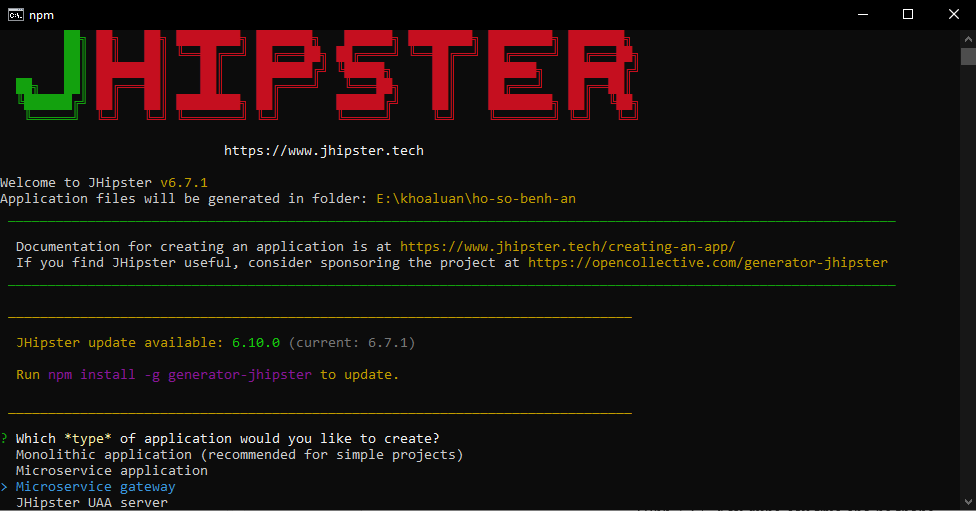
mkdir ten\_thu\_muc

cd ten\_thu\_muc



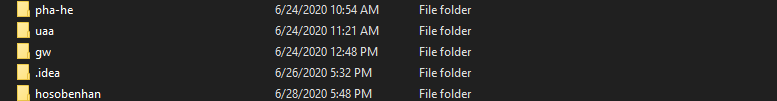
Hình 10. Xây dựng cấu trúc cho hệ thống

Bước 3: Để tạo một ứng dụng: Gõ jhipster và nhấn Enter.



Hình 11. Xây dựng cấu trúc cho hệ thống

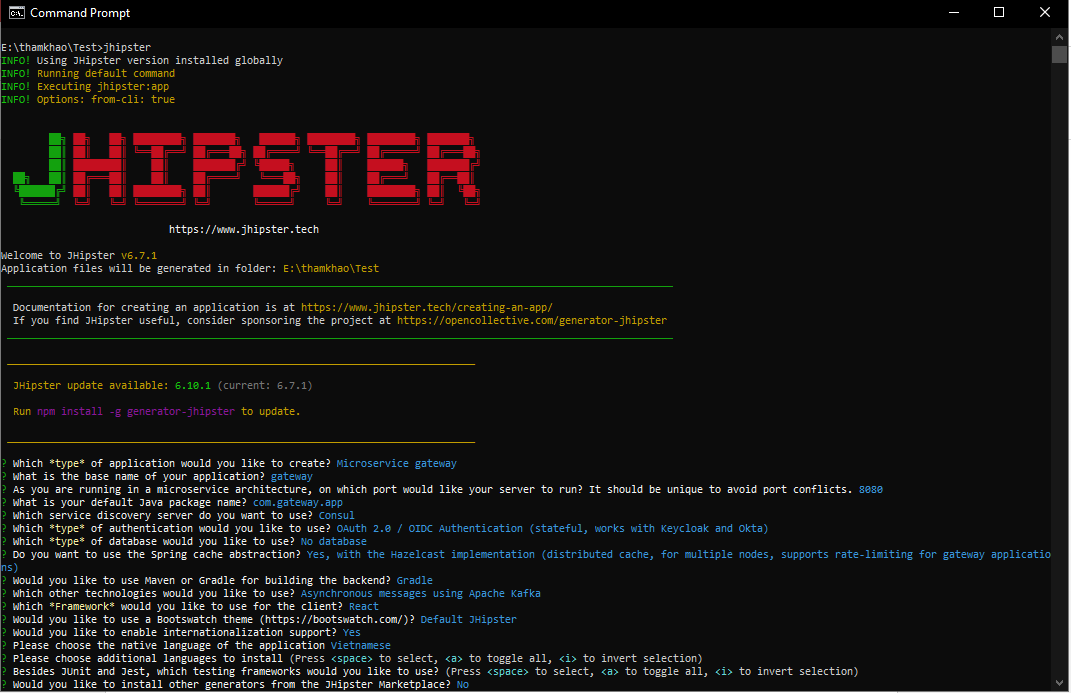
Mỗi một Service sẽ tạo một thư mục riêng và lưu trữ vào. Sau khi xây dựng các Service cần thiết cấu trúc thư mục sẽ như hình sau:



Hình 12. Xây dựng cấu trúc cho hệ thống

**❖ Tạo dịch vụ Gateway:**

Bước 1: Chọn Microservice gateway và nhấn Enter



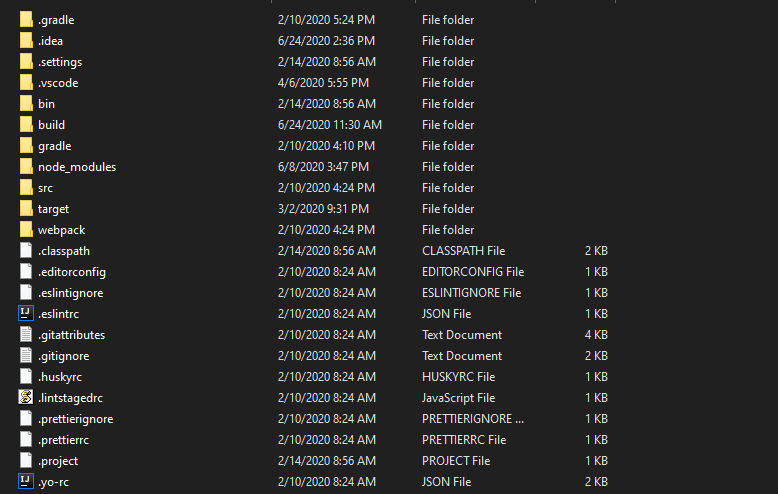
Hình 13. Xây dựng dịch vụ gateway

Bước 2: Các lựa chọn sẽ như hình bên trên, Jhipster sẽ cài đặt các lựa chọn cần thiết cho Service Gateway, theo như đồ án này thì sử dụng ngôn React nên ở lựa chọn *Which Framework would you like to use for the client?* Hãy chọn React.

Sau khi cài đặt xong:

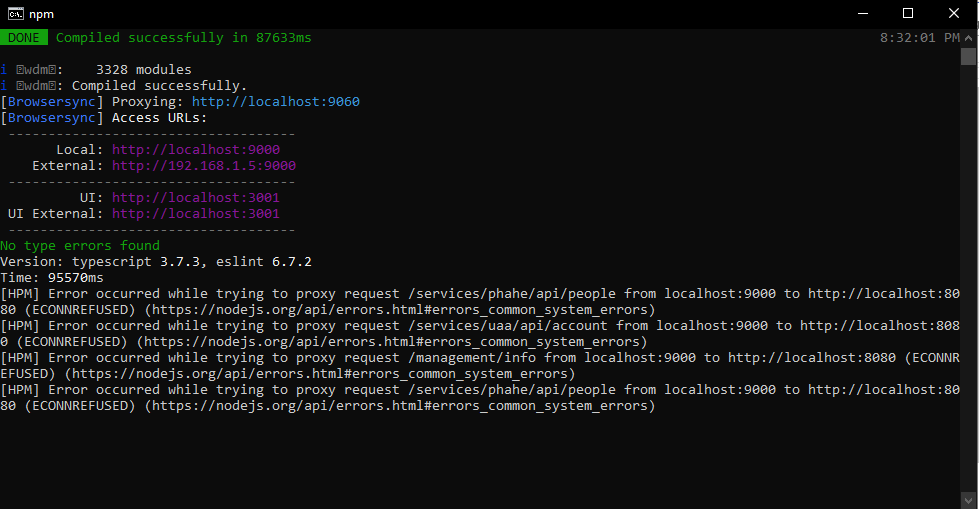
Cấu trúc thư mục sẽ có dạng như sau:

Trong File Explorer:



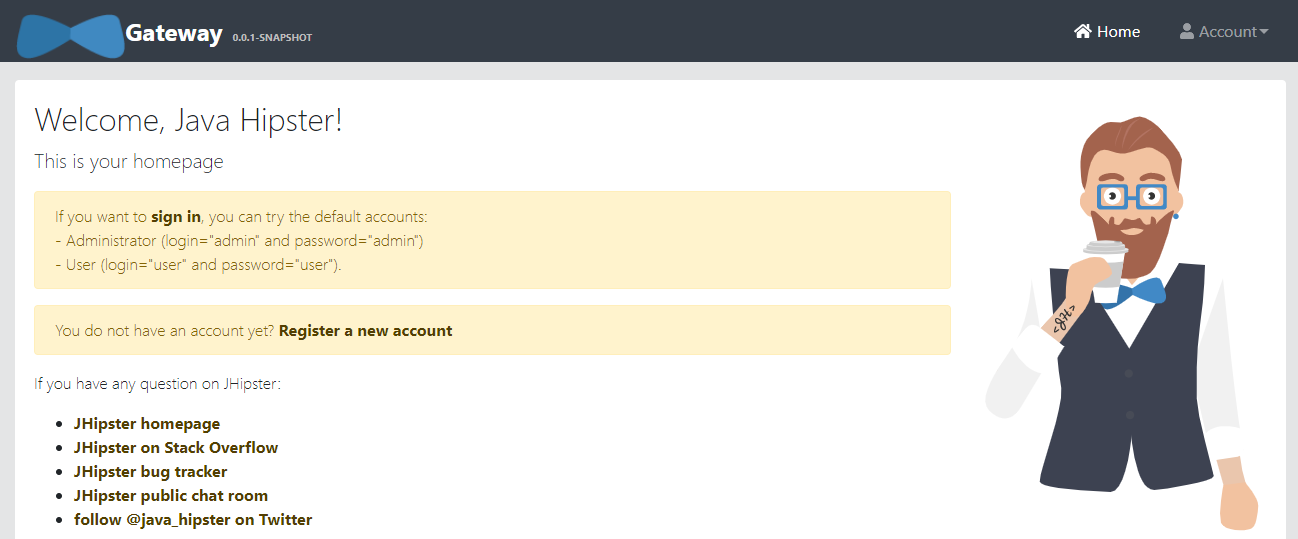
Hình 14. Cấu trúc thư mục dịch vụ Gateway

Bước 3: Nhập npm install để cài đặt các gói cần thiết. Sau đó nhập npm start để chạy dịch vụ



Hình 15. Kết quả sau khi khởi chạy Service

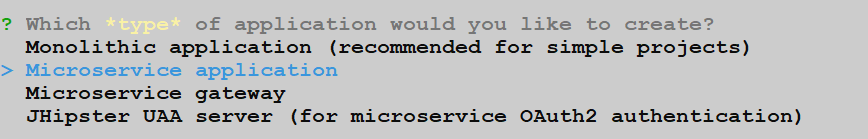
Lúc này, trên trình duyệt mặc định của hệ điều hành Windows sẽ mở ra một trang web có đường dẫn là: <http://localhost:9000/#/> . Trang web như sau là đã thành công:



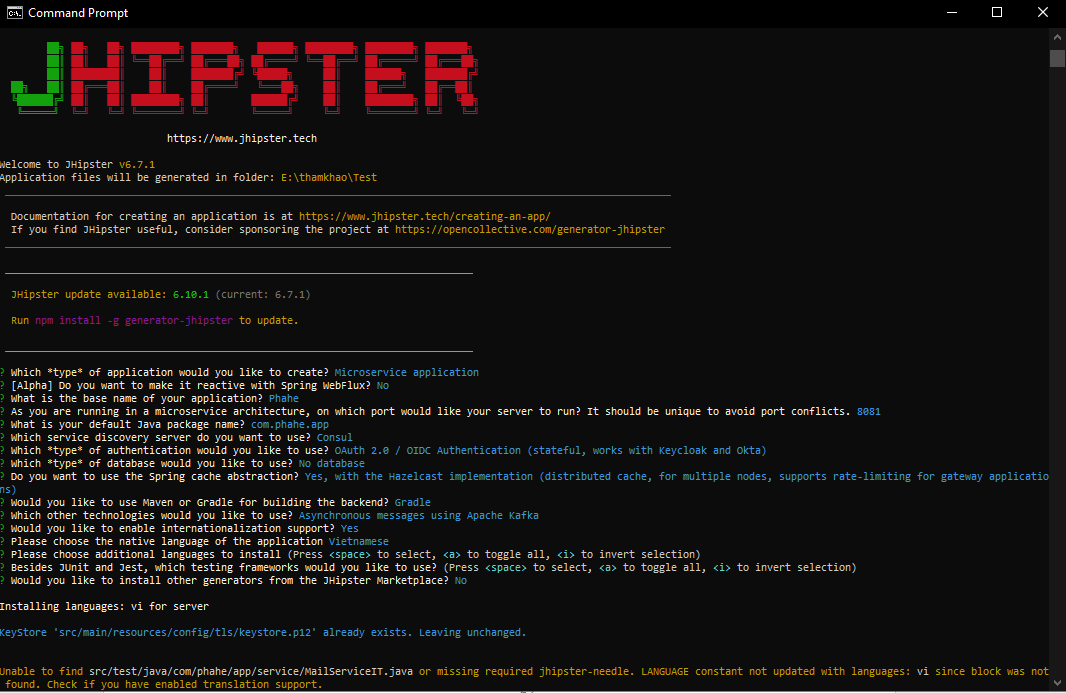
Hình 16. Trang chủ của Service Gateway

**❖ Tạo dịch vụ cho phân hệ Cây phả hệ và Hồ sơ bệnh án:**

Bước 1: Chọn Microservice application và nhấn Enter

  
Hình 17. Xây dựng dịch vụ phả hệ và hồ sơ bệnh án bác sĩ gia đình

Bước 2: Nhập như hình bên dưới, mỗi lần nhập nhấn Enter một lần. Sau khi nhập xong nhấn Enter và JHipster sẽ tiến hành cài đặt cấu trúc cần thiết cho Service Phả hệ và Hồ sơ bệnh án.



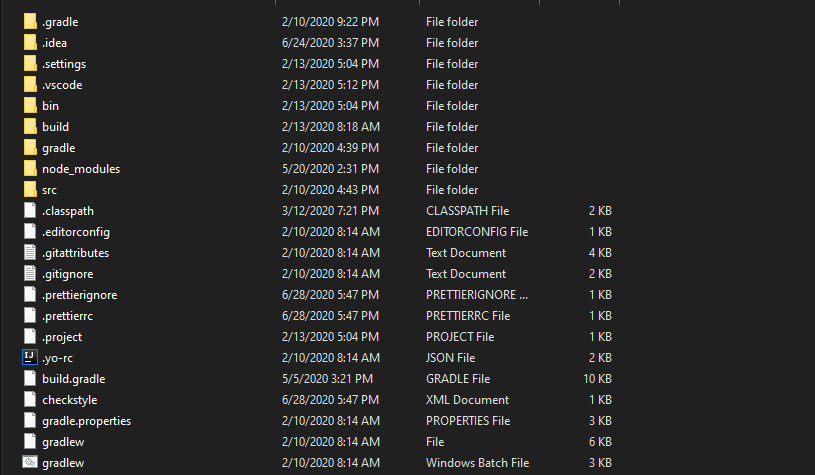
Hình 17. Cài đặt Service Phả hệ

Đối với Service Hồ sơ bệnh án thì ở dòng *Which type of database would you like to use?* Sẽ chọn MongoDB (cho MongoDB) hoặc No DataBase (cho Neo4J, do kiến trúc của Neo4J vẫn chưa được hỗ trợ cho jhipster, chỉ mới hỗ trợ thư viện truy cập)

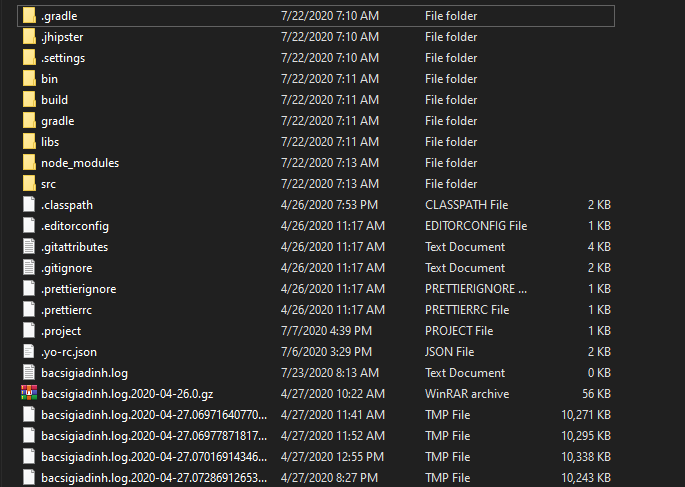
Sau khi cài đặt xong:

Cấu trúc thư mục sẽ có dạng như sau:

Trong File Explorer:

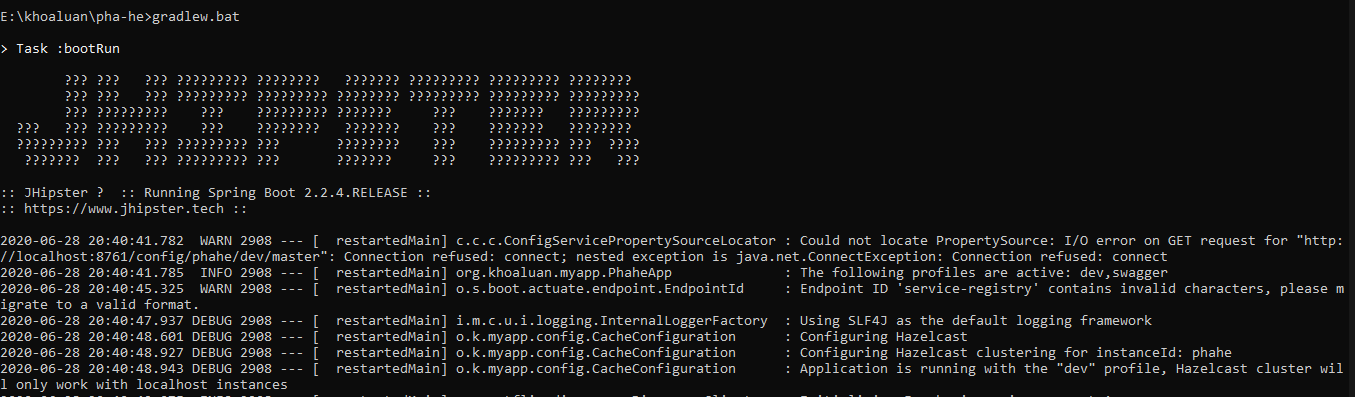


Hình 18. Cấu trúc thư mục Service Phả hệ

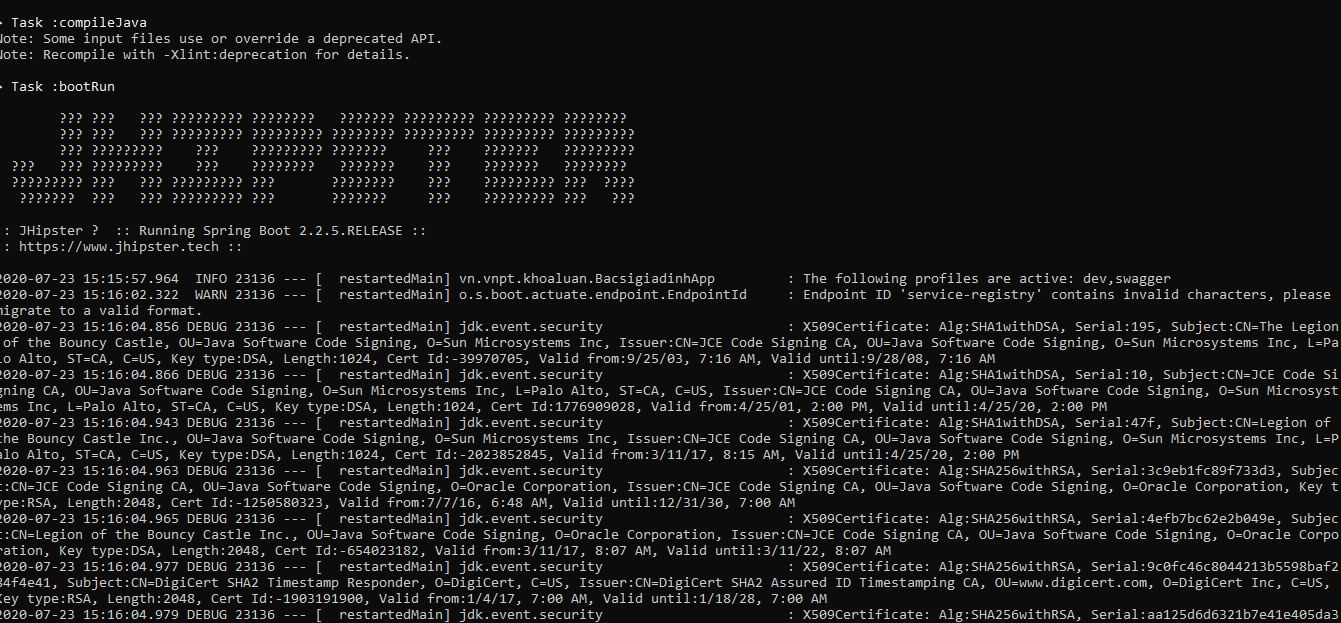


Hình 19. Cấu trúc thư mục Service Hồ sơ bệnh án

Bước 3: Nhập npm install để cài đặt các gói cần thiết. Sau đó nhập gradlew hoặc gradlew.bat để chạy dịch vụ



Hình 20. Service phả hệ sau khi khởi chạy

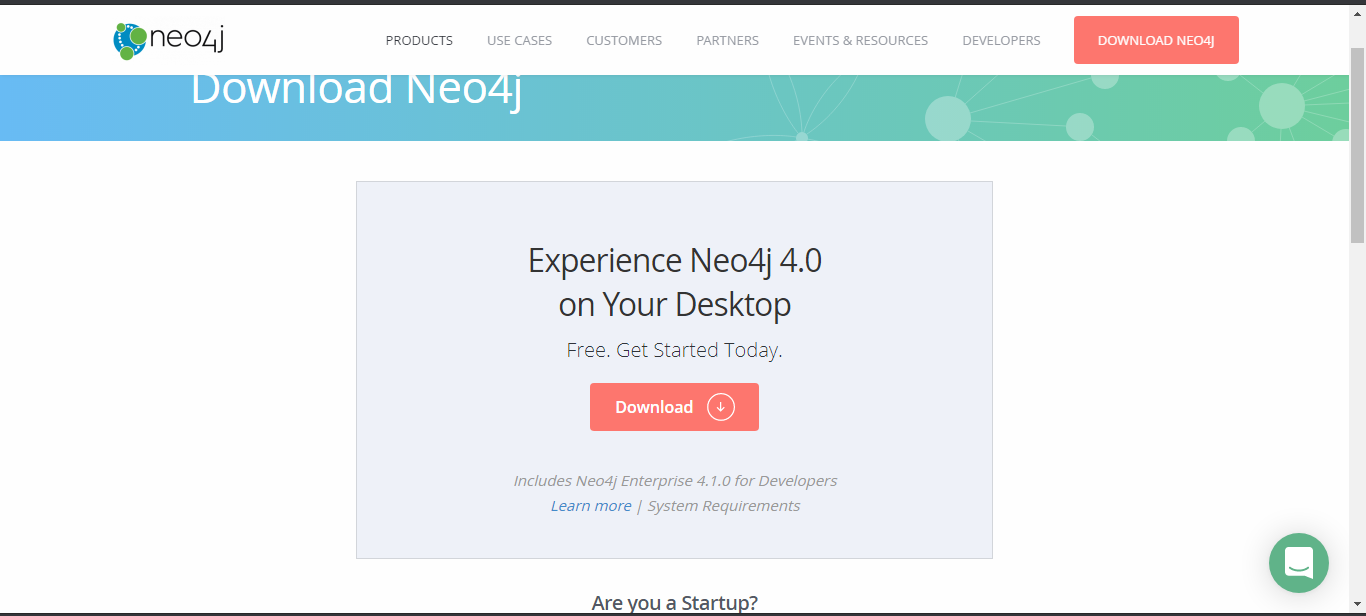


Hình 21. Service hồ sơ bệnh án sau khi khởi chạy

### 3.3.2. Cài đặt môi trường cho Graph Database (Neo4J) và NoSQL Database (MongoDB)

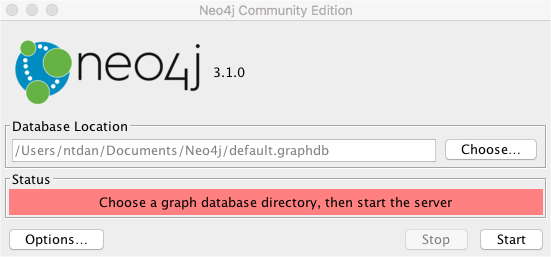
* **Cài đặt Neo4J**

Bước 1: Vào trang chủ của Neo4J: <https://neo4j.com> và tải Neo4J về máy.



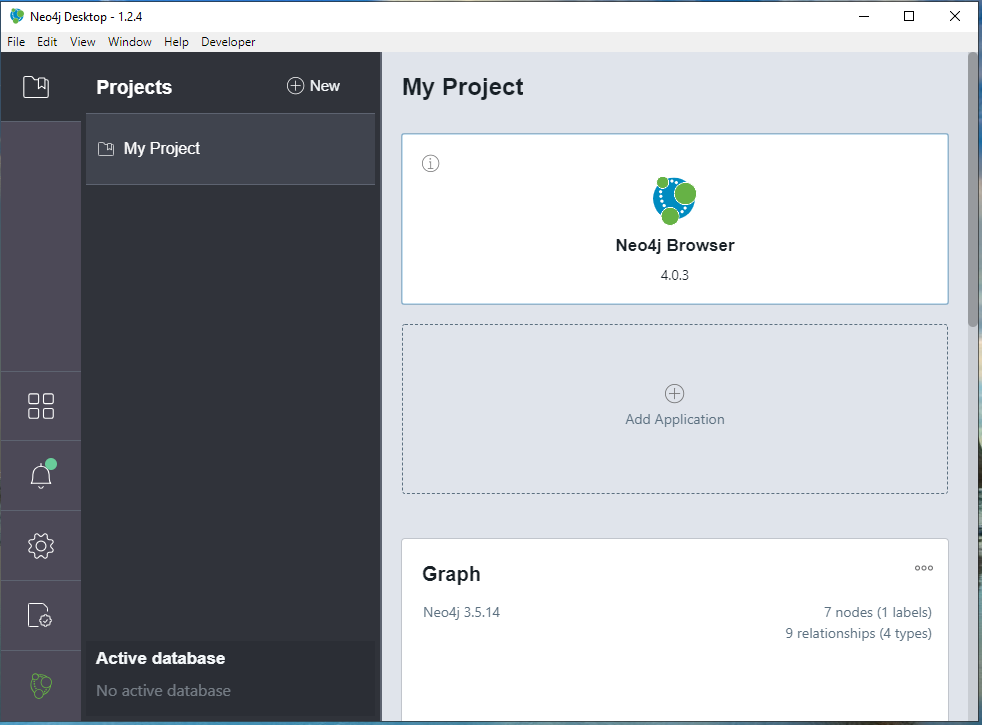
Hình 22. Dowload Neo4J

Bước 2: Sau khi download và cài đặt tiến hành cấu hình nơi lưu trữ dữ liệu:



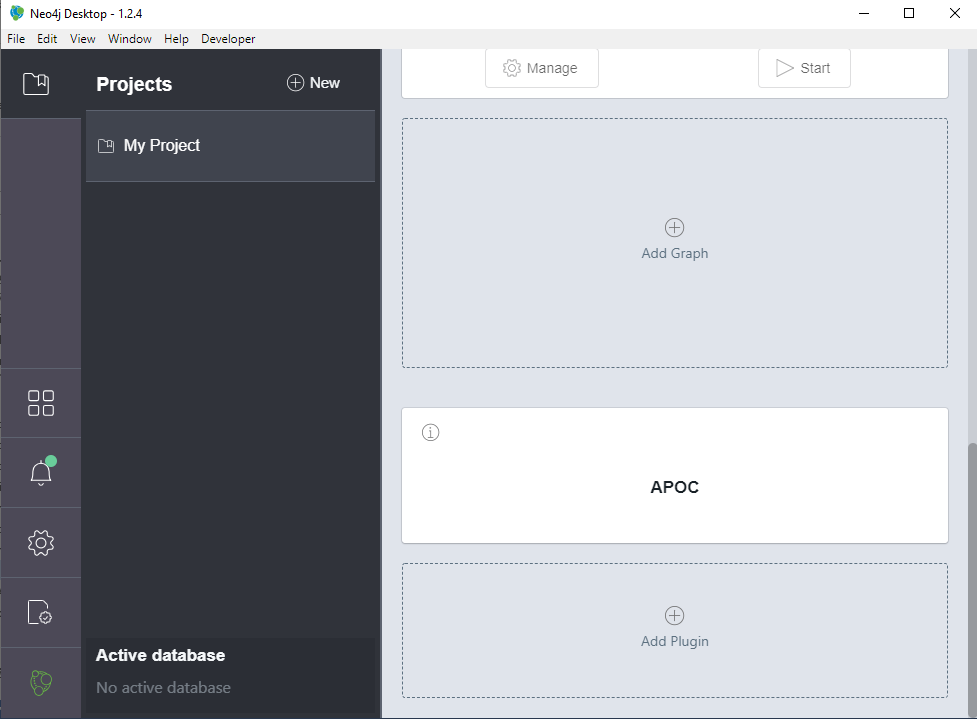
Hình 23. Cài đặt nơi lưu trữ dữ liệu

Bước 3: Nháy đúp vào biểu tưởng Neo4J để mở.



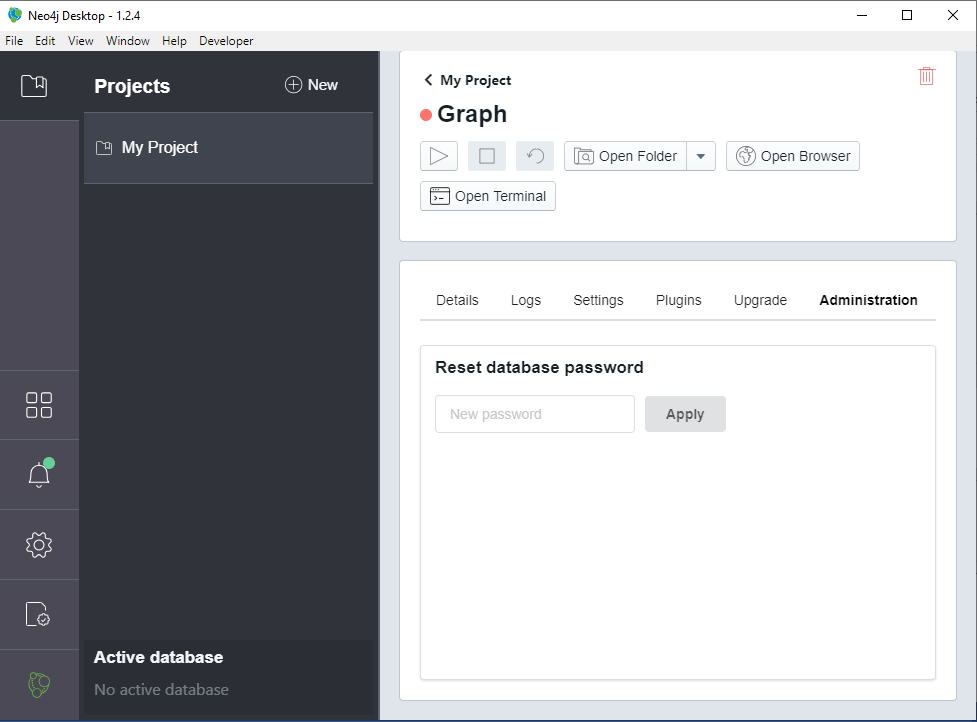
Hình 24. Giao diện của Neo4J

Bước 4: Tiến hành tạo một Graph Database, thiết lập username, password và các gói plugin cần thiết.



Hình 25. Giao diện tạo Graph và Plugin

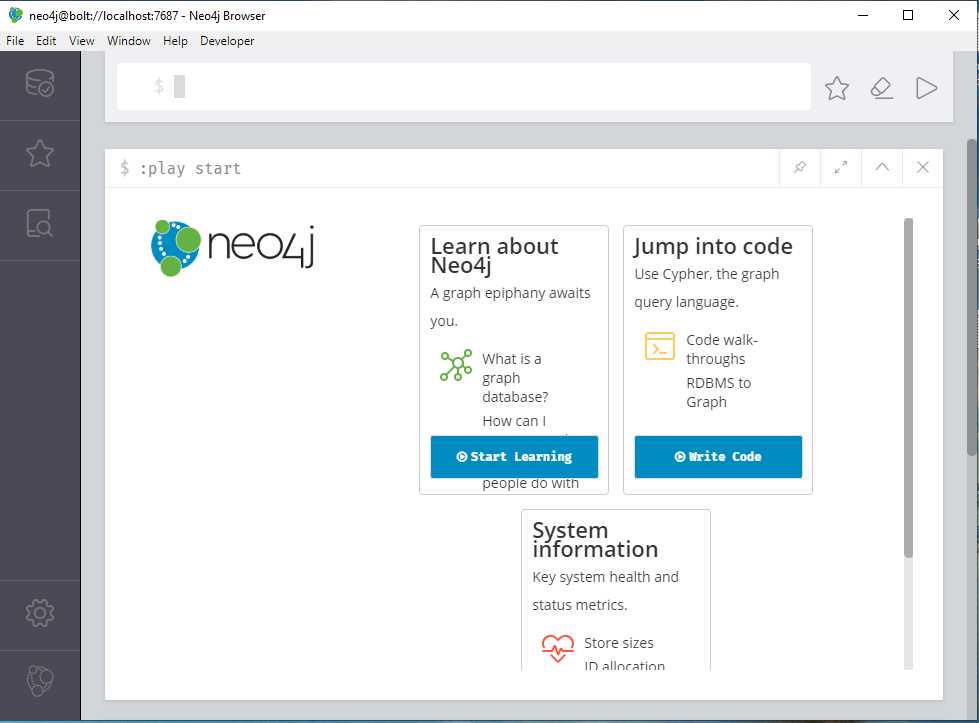
Nhấn vào Add Graph để thêm một Graph Database cho Neo4J hoặc nhấn vào Add Plugin để thêm Plugin cho toàn bộ Graph.



Hình 26. Giao diện quản lý cho một Graph

Tab Plugins để chọn các Plugins cần thiết sau đó nhấn Install để cài đặt Plugin, Tab Adminnistration để thiết lại đặt mật khẩu cho Graph (Username sẽ là neo4j).

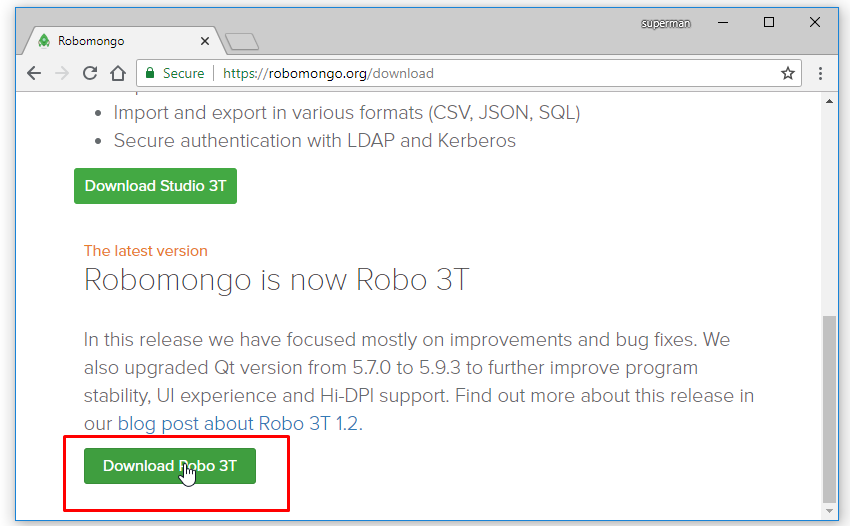
Bước 5: Nhấn  tại hình 3.31 để khởi chạy Graph Database sau đó nhấn Open Browser để tiến hành viết câu truy vấn.



Hình 27. Giao diện viết câu truy vấn cho Graph Database (Neo4J)

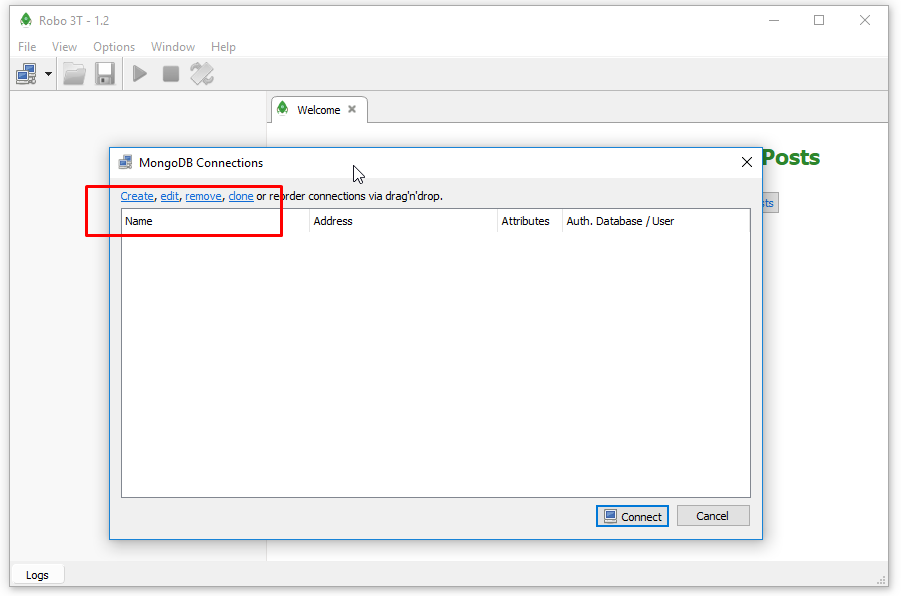
* **Cài đặt Robo 3T (RoboMongoDB)**

Bước 1: Vào trang chủ: <https://robomongo.org/download> để tải Robo 3T về máy



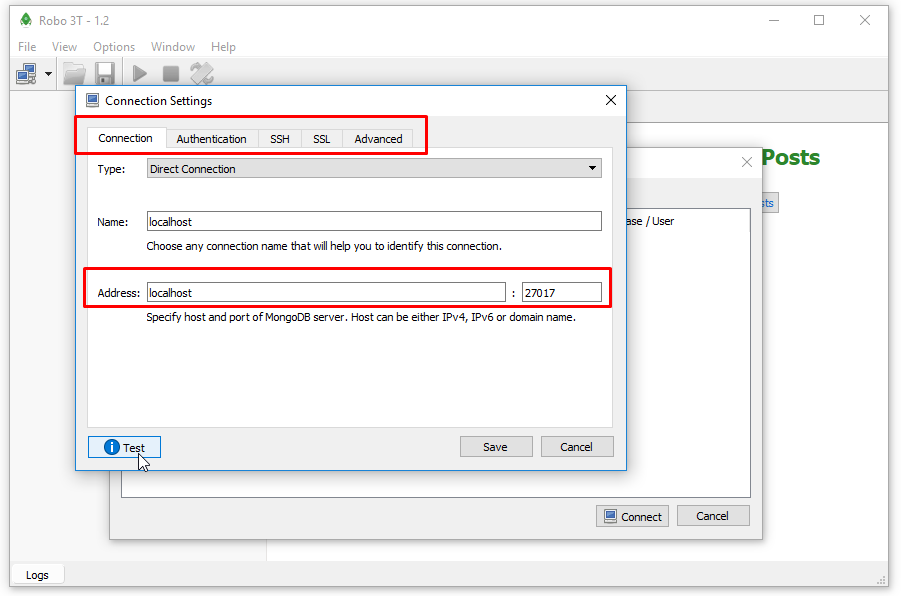
Hình 28. Dowload Robo 3T

Bước 2: Sau khi dowload và cài đặt thì đây là giao diện Robo 3T khi lần đầu mở



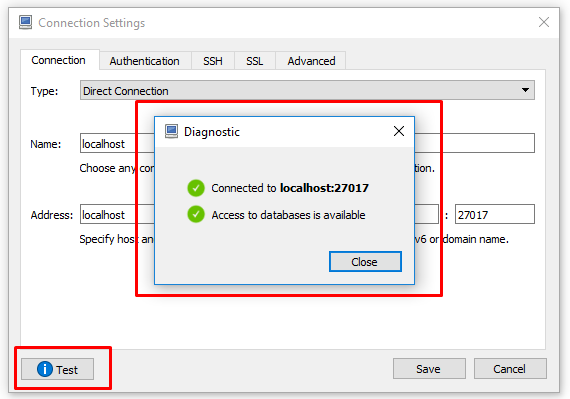
Hình 29. Giao diện Robo 3T

Bước 3: Click vào Create để tạo kết nối tới MongoDB



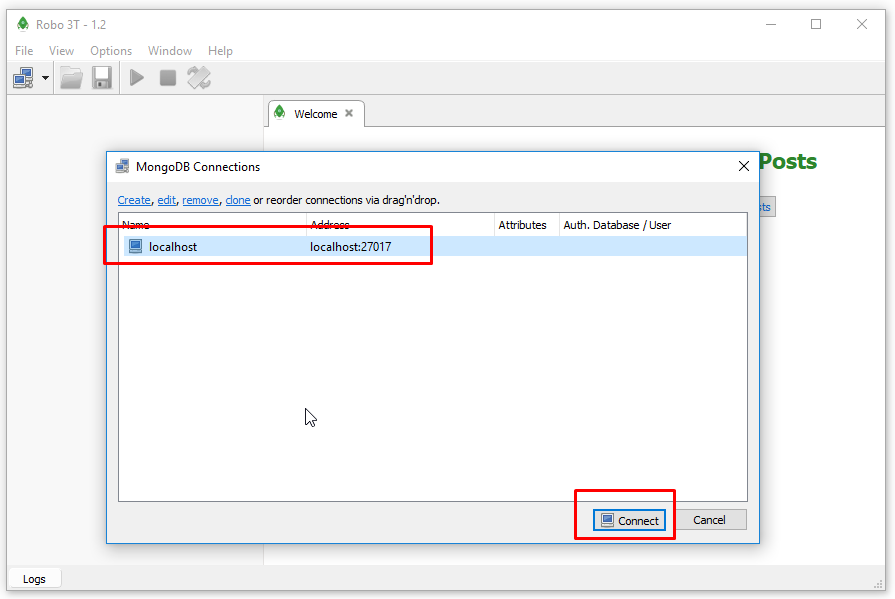
Hình 30. Tạo kết nối với MongoDB

Bước 4: Click vào Test để kiểm tra kết nối



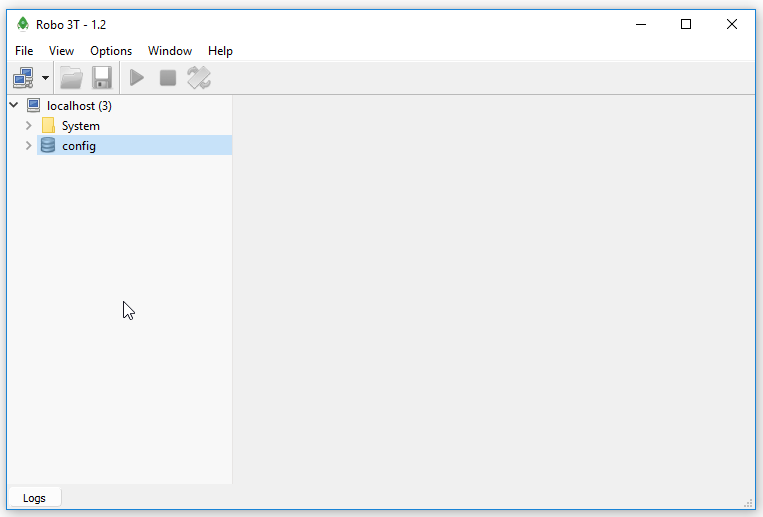
Hình 31. Kiểm tra kết nối với MongoDB

Bước 5: Click vào Save để lưu lại kết nối (connection), sau khi lưu lại kết nối nó sẽ xuất hiện trên bảng connection để chọn kết nối. Sau đó click vào connect để kết nối tới MongoDB



Hình 32. Minh họa kết nối với MongoDB

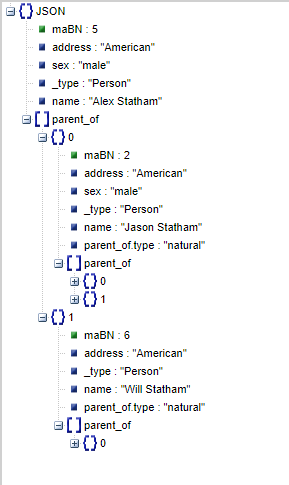
Đây là giao diện của Robo 3T sau khi kết nối thành công



Hình 33. Giao diện Robo 3T sau khi đã kết nối thành công

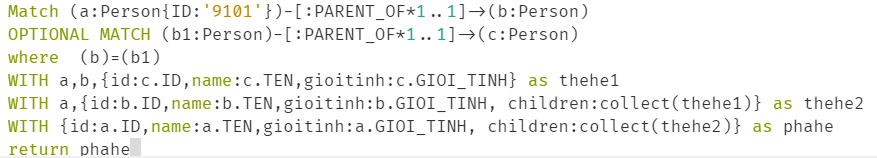
### 3.3.3. Vấn đề và giải pháp

* **Service phả hệ:**
* Vấn đề: Do yêu cầu của chức năng Truy xuất thế hệ 3 đời của một bệnh nhân nên sẽ nãy sinh ra các vấn đề sau:
* Thứ 1: Dữ liệu được chuyển từ Service Phả hệ qua thư viện dùng để hiển thị ở Gateway sẽ dưới dạng JSON có cấu trúc phân lớp như sau:



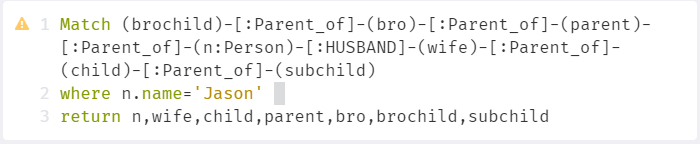
Dựa theo cấu trúc trên, lớp ngoài cùng của JSON thể hiện cho người lớn nhất trong 3 thế hiện được truy xuất, mỗi lớp con sẽ thể hiện qua cụm *“parent\_of “* tức quan hệ người đó có con cái là ai.

* Thứ 2: Do vấn đề thứ nhất phát sinh nên khi viết câu truy vấn từ Service Phả hệ để lấy dữ liệu từ Neo4J thì Neo4J cũng phải trả về dữ liệu JSON có cấu trúc như trên mà theo như truy vấn thông thường dữ liệu trả về theo Entity sẽ không đúng cấu trúc và bị lặp các mối quan hệ con (do quan hệ một chiều giữa mỗi cặp Node).
* Thứ 3: khi vấn đề 1-2 được giải quyết thì vấn đề thứ 3 phát sinh đó là không thấy được các anh chị em cùng nhánh của bệnh nhân nhỏ tuổi nhất được truy xuất (nút lá) và nếu trong cùng một gia đình có một nhánh không đủ ba thế hệ (chưa có con) nhưng các nhánh khác có thì cũng không thể trả về đủ.
* Thứ 4: sau khi giải quyết được các vấn đề trên thì do trả về cấu trúc JSON với lớp ngoài cùng luôn luôn phải là người già nhất trong 3 thế hệ được truy xuất của bệnh nhân, nên chỉ với một câu truy vấn sẽ không thể nào đúng được cho mọi trường hợp. Ngoài ra khi gia đình bệnh nhân đó chỉ mới được lưu trữ 2 thế hệ thì cũng sẽ dẫn đến trường hợp sai.
* Giải pháp cho từng vấn đề trên:
* Giải pháp cho vấn đề 1, 2: sử dụng hàm Collect để gộp các bệnh nhân có cùng thế hệ với nhau. Xây dựng cấu trúc 3 lớp tương ứng với ba thế hệ.



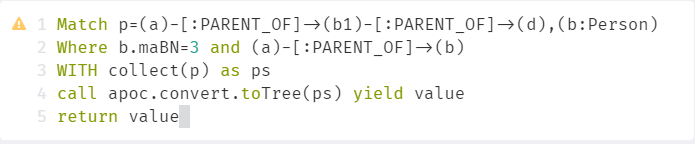
Hình 34. Ví dụ cho cách sử dụng hàm Collect

* Giải pháp cho vấn đề 3: Trường hợp câu truy vấn trả về không đúng giá trị yêu cầu theo như câu truy vấn 1 vế thông thường.



Hình 35. Ví dụ cho câu truy vấn một vế trong Neo4J

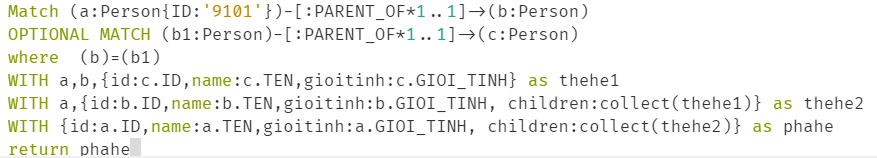
Với câu truy vấn như trên thì không thể nào lấy ra được cả anh chị em đầy đủ được do đặc trưng của cấu trúc rẻ nhánh trong Graph Database. Chính vì thế tôi tách câu truy vấn ra thành các vế tương ứng và ràng buộc các vế đó lại để có thể truy xuất được đầy đủ, ví dụ như hình:



Hình 36. Ví dụ khi tách câu truy vấn trong Neo4J

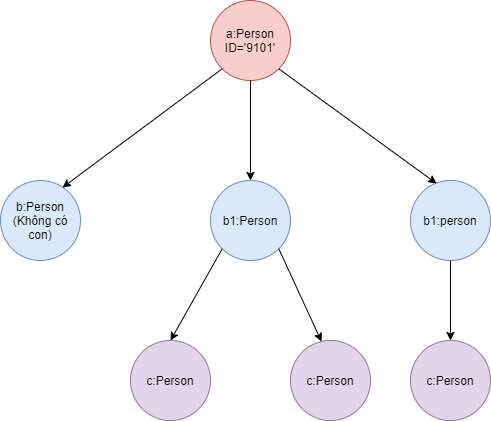
Tách bệnh nhân cần tìm ra một nhánh riêng sau đó ràng buộc lại với cha mẹ chung (a) để có thể lấy đầy đủ các anh chị em.

* Giải pháp cho vấn đề 4: xây dựng các câu truy vấn tương ứng cho từng trường hợp lần lượt như sau:
* Câu truy vấn cho trường hợp bệnh nhân là người lớn tuổi nhất trong gia đình (nút gốc):



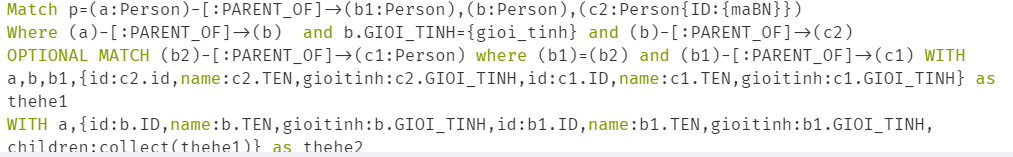
Hình 37. Câu truy vấn minh họa nếu bệnh nhân là người lớn tuổi nhất

Khi đó (a:Person) sẽ là người lớn tuổi nhất trong gia đình, (b:Person) sẽ chỉ chung các người con của a bao gồm đã có con và chưa có con, (b1:Person) thực ra là (b:Person) nhưng được tách ra để khi (b:Person) không có con câu truy vấn sẽ không bị sai.



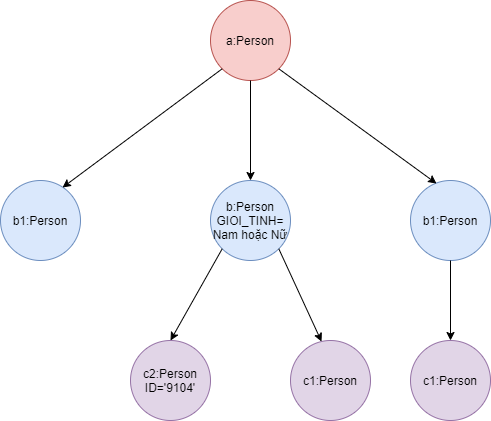
Hình 38. Minh họa cho câu truy vấn nếu bệnh nhân là người lớn tuổi nhất

* Câu truy vấn cho trường hợp bệnh nhân là người nhỏ tuổi nhất trong gia đình (nút lá):



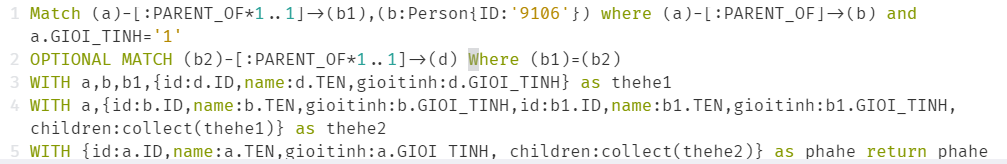
Hình 39. Câu truy vấn minh họa nếu bệnh nhân là người nhỏ tuổi nhất

(a:Person) người lớn tuổi nhất trong ba thế hệ được truy xuất (có thể không phải nút gốc), (b1:Person) sẽ là tất cả con của (a:Person), (b:Person) sẽ là cha/mẹ (tùy giới tính được chỉ định) của (c2:Person) với c2 là bệnh nhân được chỉ định với ID. (b2:Person) chính là (b1:Person) nhưng được tách ra để bao hàm trường hợp (b1:Person) chưa có con và để lấy được anh/chị em của (c2:Person) nếu có.



Hình 40. Ảnh minh họa cho câu truy vấn nếu bệnh nhân là người nhỏ tuổi nhất

* Câu truy vấn cho trường hợp bệnh nhân nằm trong khoảng ở giữa có cả mối quan hệ trước lẫn sau:



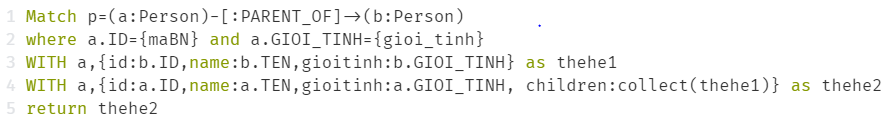
Hình 41. Câu truy vấn minh họa nếu bệnh nhân thế hệ trung niên trong gia đình

(a:Person) người lớn tuổi nhất trong ba thế hệ (có thể không phải nút gốc) được truy xuất đồng thời là cha/mẹ của (b1) tùy giới tính được chỉ định, (b1:Person) sẽ là tất cả con của (a:Person), (b:Person) sẽ là bệnh nhân được chỉ định theo ID đồng thời là cha/mẹ của (d:Person) với (b) sẽ vẫn là con của (a) và là 1 thành viên trong (b1). (b2) thực chất là (b1) nhưng được tách ra để bao hàm trường hợp 1 bệnh nhân trong (b2) chưa có con.



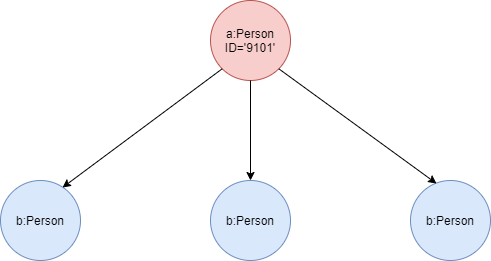
Hình 42. Ảnh minh họa cho câu truy vấn nếu bệnh nhân là thế hệ trung niên trong gia đình

* Câu truy vấn lấy hai thế hệ cho thế hệ cha:



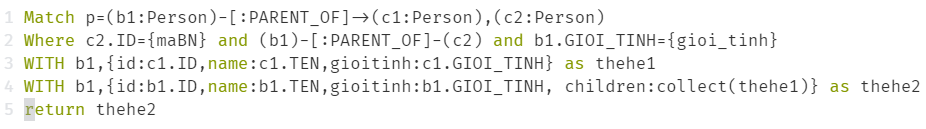
Hình 43. Câu truy vấn minh họa lấy hai thế hệ (cha)

Trường hợp này tương đối ít xảy ra, khi toàn bộ con của (a:Person) không có con, và (a:Person) cũng là người già nhất trong gia đình.



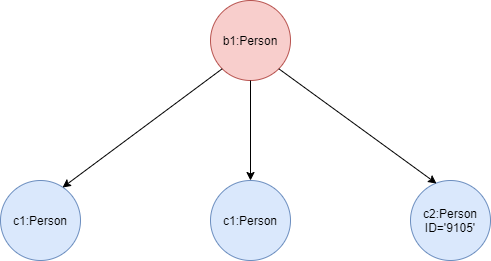
Hình 44. Ảnh minh họa cho câu truy vấn lấy hai thế hệ (cha)

* Câu truy vấn lấy hai thế hệ cho thế hệ con:

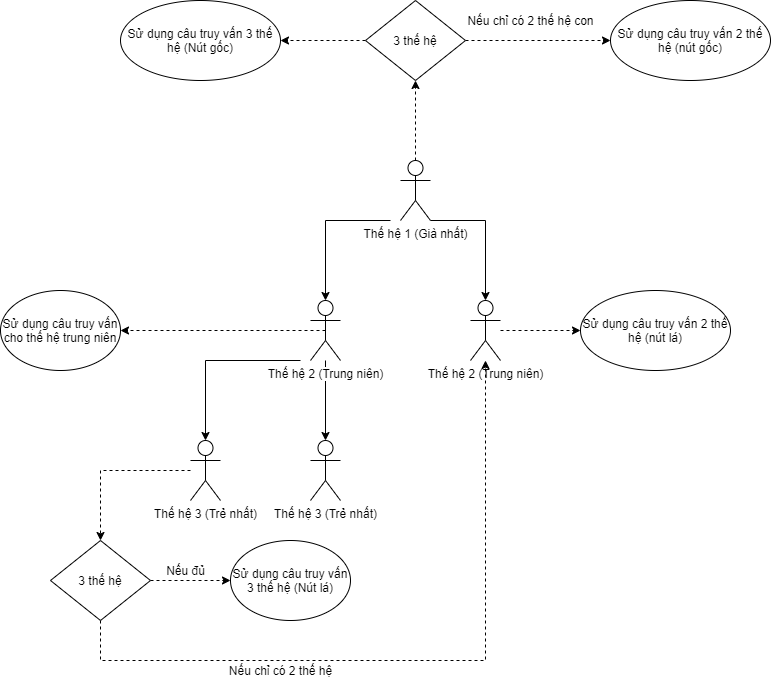


Hình 45. Câu truy vấn minh họa lấy hai thế hệ (con)

Trường hợp khi (c1:Person) là con của (b1:Person), (c1) không có con và (b1) không có cha/mẹ. (c2:Person) được tách ra để chỉ định thông qua ID.



Hình 46. Ảnh minh họa câu truy vấn lấy hai thế hệ (con)



Hình 47. Minh họa các trường hợp sử dụng câu truy vấn

🡪 Sau khi giải quyết được các vấn đề trên dữ liệu trả về sẽ khớp với Thư viện sử dụng.

### 3.3.4. Cài đặt hệ thống

* **Service phả hệ:**

Cài đặt tất cả các chức năng cần thiết của Cây phả hệ trong Microservice application (Dùng Spring Data Neo4J để kết nối tới Cơ sở dữ liệu và Neo4J Migration để quản lý các thay đổi đó). Trong Service Phả hệ sẽ có các thành phần cần thiết như sau:

* Obj: Dùng để chứa các đối tượng cần thiết để ánh xạ dữ liệu từ CSDL.
* Repository: Dùng để chứa các câu truy vấn lấy dữ liệu lên từ Cơ sở dữ liệu đồng thời trả về Obj.
* Service: Tạo các hàm để xử lý các câu truy vấn trong Repository.
* Resource: chứa các Mapping để Get, Post, Put, Delete dữ liệu từ Service, đồng thời trả về những Exception tương ứng cho trình duyệt.
* **Service Hồ sơ bệnh án**

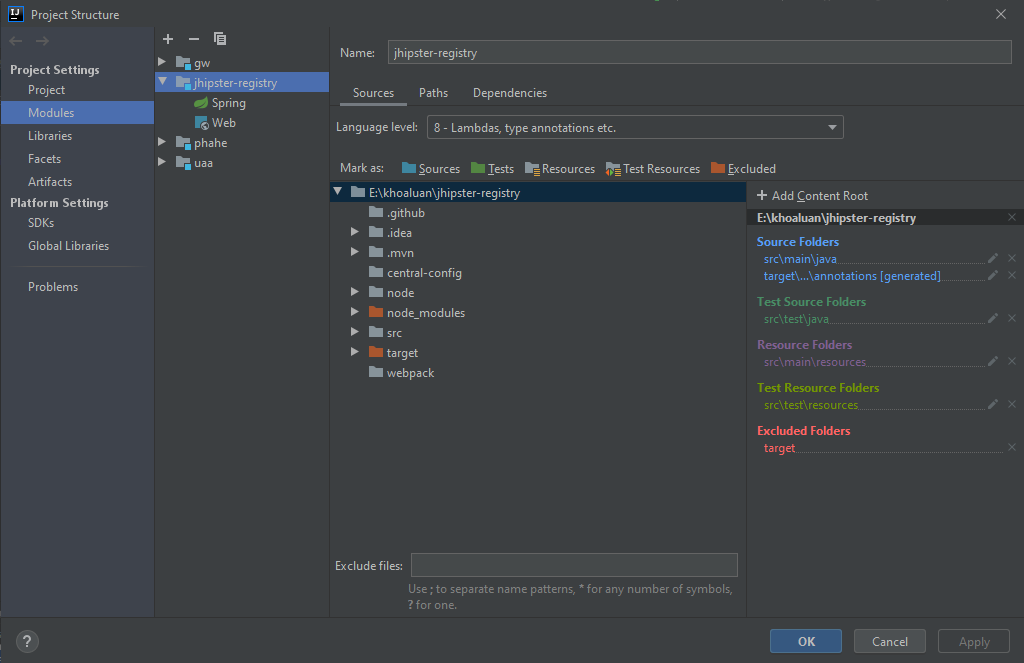
Cài đặt tất cả các chức năng cần có của hồ sơ bệnh án bác sĩ gia đình trong Microservice Application (kết nối Spring Data MongoDB để kết nối cơ sở dữ liệu với MongoDB, kết nối với thư viện JasperReport để thiết kế và quản lý các hồ sơ bệnh án bác sĩ gia đình). Trong service Hồ sơ bệnh án sẽ có các thành phần cần thiết như sau:

* Service (dto): Dùng để chứa các đối tượng cần thiết trong một Collection (bảng) trong cơ sở dữ liệu MongoDB để ánh xạ dữ liệu từ cơ sở dữ liệu và các phương thức get(), set() dữ liệu.
* Reponsitory: Dùng để chứa các câu truy vấn lấy dữ liệu từ cơ sở dữ liệu
* Service (iml): Dùng để tạo các hàm xử lý các câu truy vấn trong Reponsitory.
* Service: Dùng để chứa các hàm xử lý các câu truy vấn trong Service (iml)
* Resource: Dùng để chứa các phương thức Get, Post, Put, Delete.

Do yêu cầu của chức năng và kiến trúc của Microservice nên để hệ thống có thể hoạt động được ta phải lần lượt khởi chạy các Service theo thứ tự:

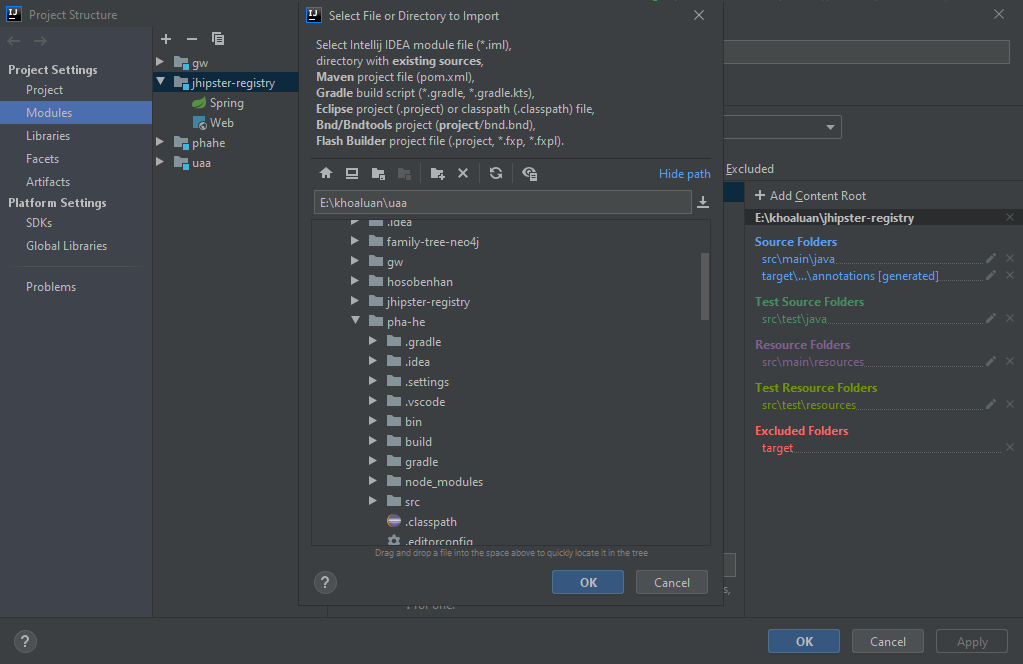
Consul🡪 KeyCloak 🡪 Gateway 🡪 Phả hệ 🡪 Hồ sơ bệnh án

Để chạy lần lượt các Service một cách tường minh và liên tục, chúng tôi sử dụng IntelliJ IDEA để quản lý, các bước như sau:

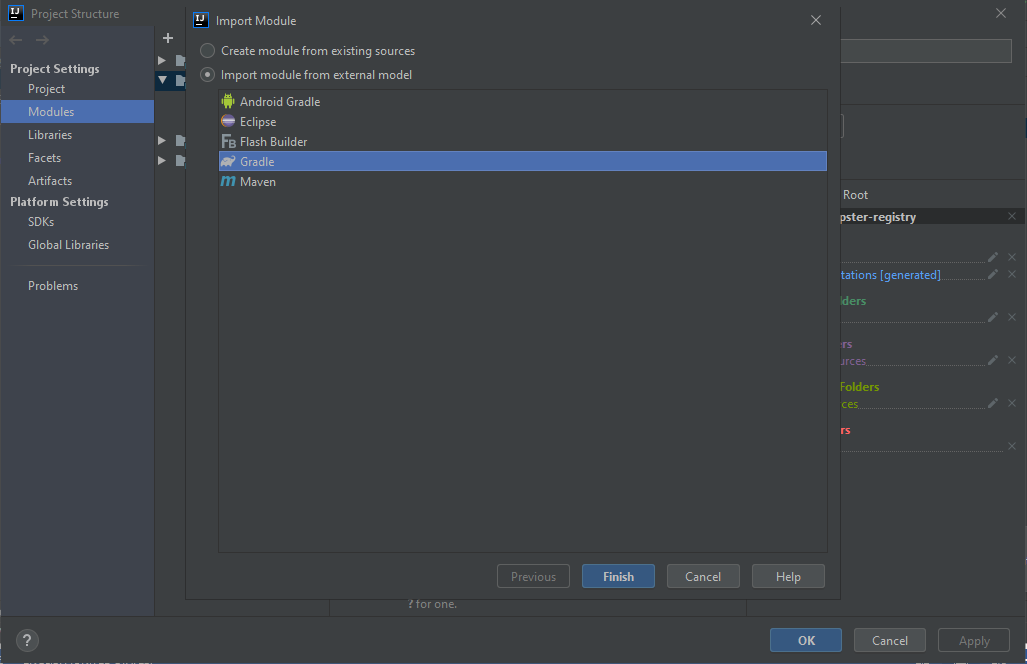


Hình 48. Thêm service vào IntteliJ IDEA 2018

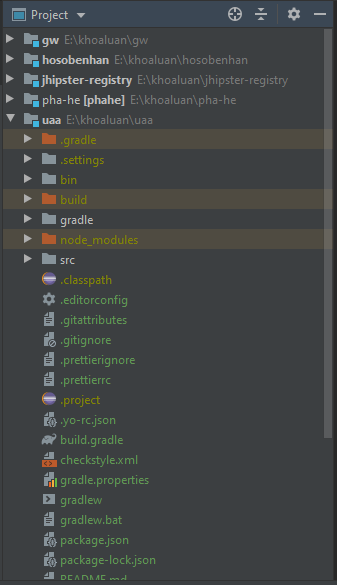
Bước 2: Nhấn vào dấu + 🡪 Chọn Import Modules 🡪 Chọn đường dẫn đến các dịch vụ đã tạo 🡪 Chọn module tương ứng với Service (Gradle, Maven) 🡪 Nhấn OK.



Hình 49. Thêm Service vào IntelliJ



Hình 50. Thêm Service vào IntelliJ



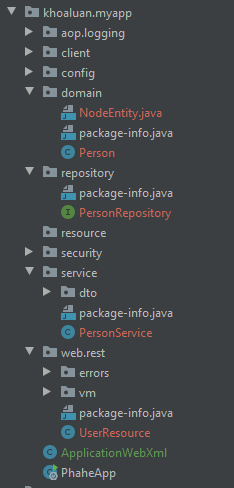
Hình 51. Cấu trúc thư mục của Project sau khi Add

Bước 3: Chọn Service cần khởi chạy và nhấn Debug .



Hình 52. Khởi chạy các Service

**❖ Kết quả cài đặt Service Cây Phả hệ**



Hình 53. Cây thư mục Phả hệ sau khi cài đặt

Do bản chất của Graph Database nên ở Domain sẽ chỉ có một đối tượng Obj ánh xạ dữ liệu đó là Person.

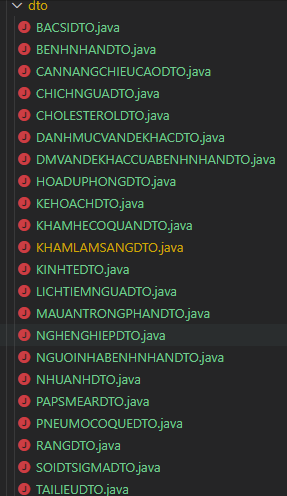
Repository sẽ chứa các câu truy vấn để lấy dữ liệu từ Neo4J, Service sẽ thực hiện xử lý logic với các câu truy vấn đó và Resource sẽ tạo các đường dẫn xử lý các yêu cầu từ Gateway.



Hình 54. .Minh họa lấy danh sách cây phả hệ

* **Kết quả cài đặt Service Hồ sơ bệnh án**

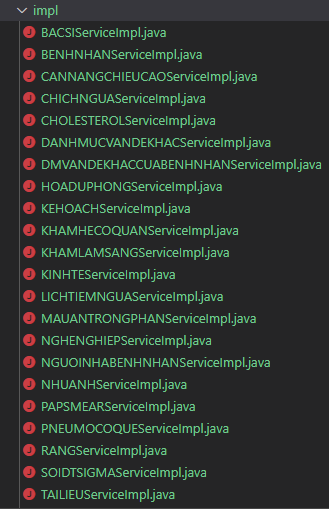
Sau khi cài đặt service Hồ sơ bệnh án thì cấu trúc của các thành phần Service (dto), Reponsitory, Service (iml), Service, Resource sẽ được hiển thị giống như các **Hình 51, 52, 53, 54, 55**



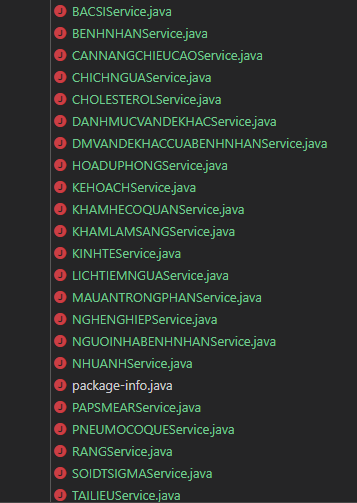
Hình 55. Service (dto)



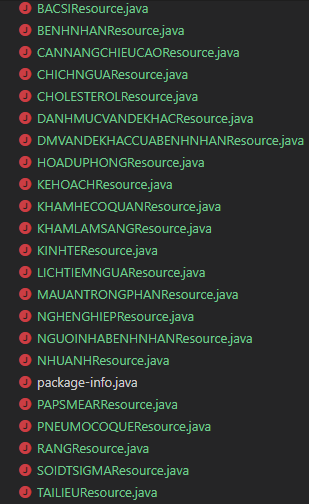
Hình 56. Reponsitory



Hình 57. Service (iml)



Hình 58. Service



Hình 59. Resource

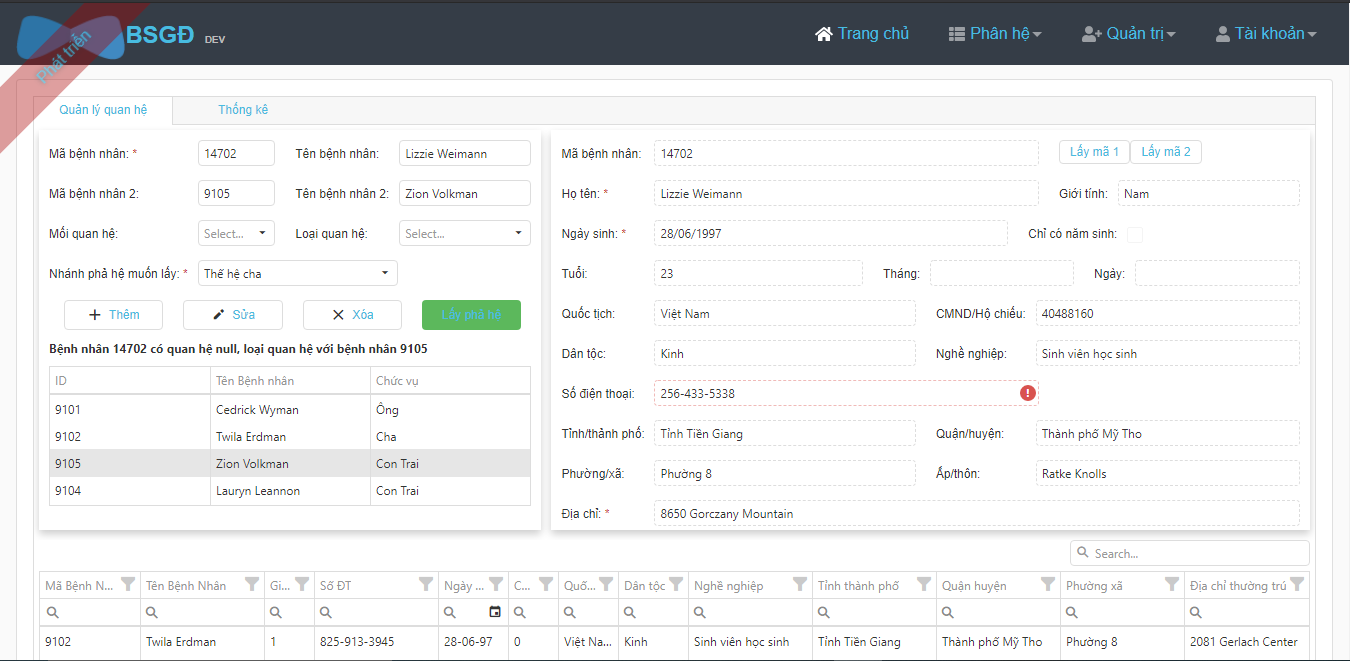
Trong Resource tạo các đường dẫn và tiếp nhận các yêu cầu từ gateway và tạo báo cáo



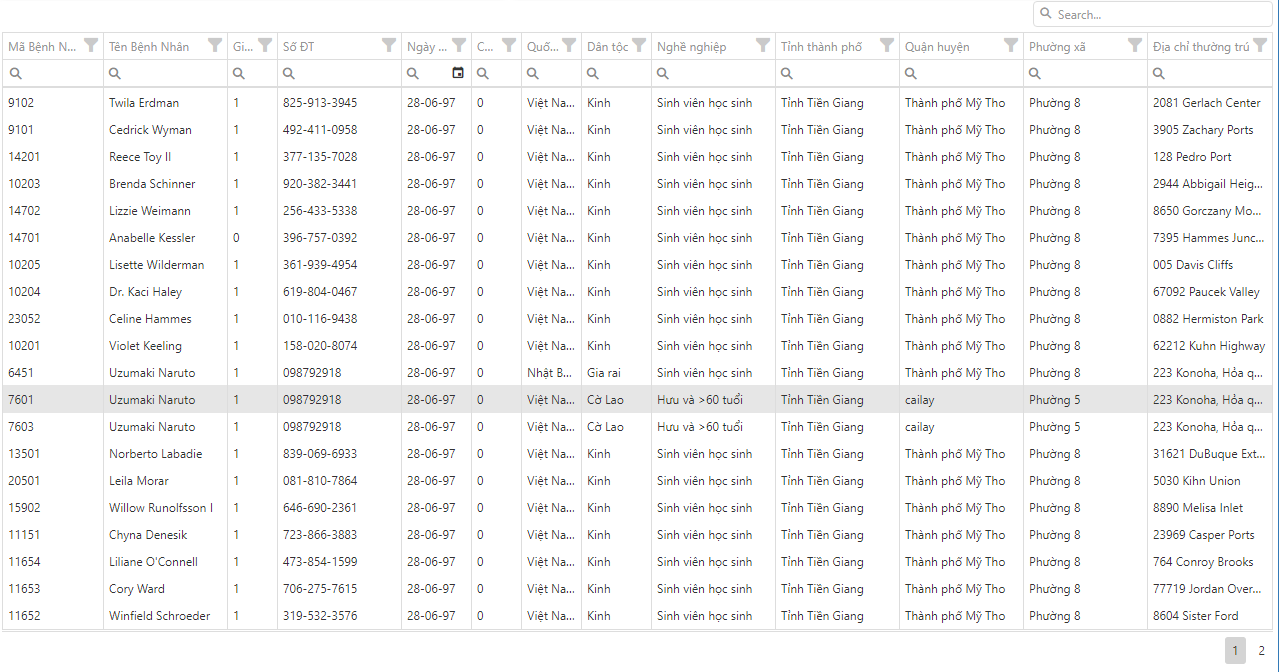
Hình 60. Minh họa in thông tin khám bệnh của bệnh nhân theo mã bệnh nhân

### 3.3.5 Một số hình ảnh giao diện hệ thống

* **Giao diện Cây Phả hệ**



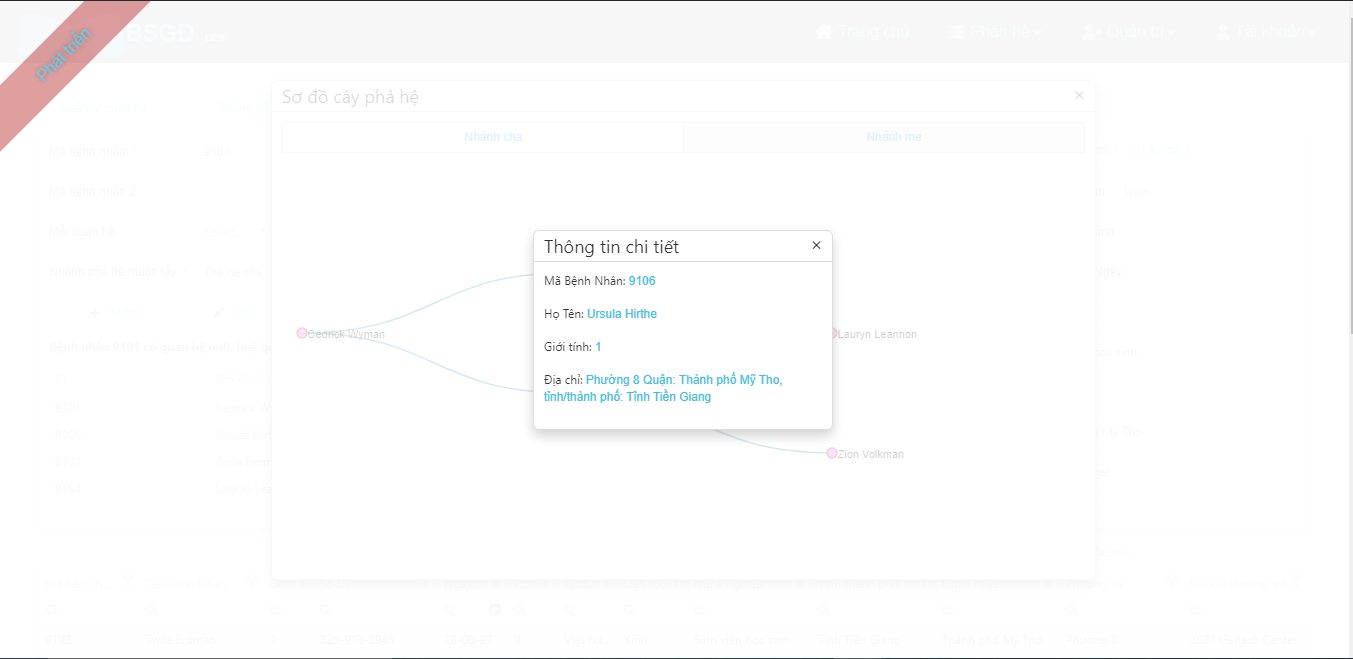
Hình 61. Giao diện của cây phả hệ 1



Hình 62. Giao diện của Cây phả hệ 2

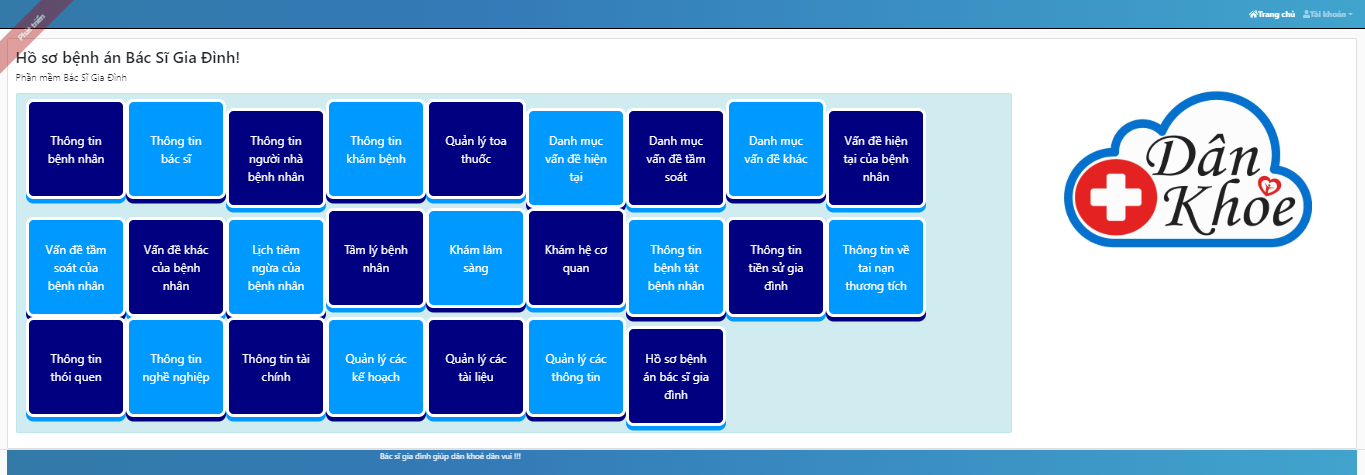


Hình 63. Giao diện Cây phả hệ (Khi lấy nhánh phả hệ 3 đời của 1 bệnh nhân)



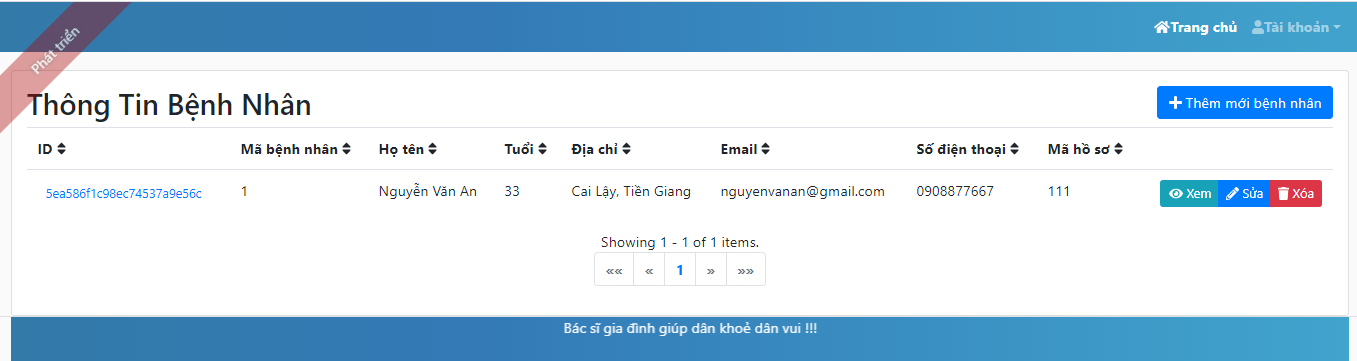
Hình 64. Giao diện Cây phả hệ (Khi nhấn vào một node bệnh nhân trong đồ thị)

* **Giao diện hồ sơ bệnh án bác sĩ gia đình**

****

Hình 65. Giao diện service hồ sơ bệnh án bác sĩ gia đình

* **Giao diện Quản lý thông tin bệnh nhân**

****

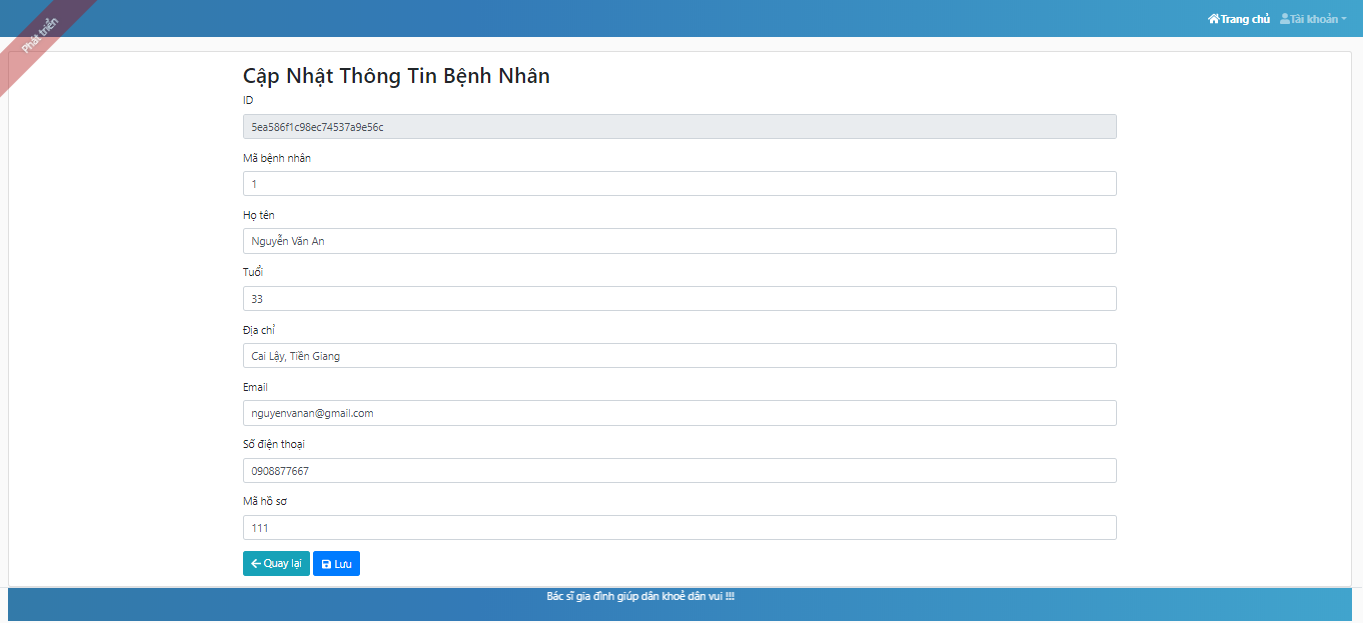
Hình 66. Giao diện Quản lý thông tin bệnh nhân

* **Giao diện Thêm mới thông tin bệnh nhân**

****

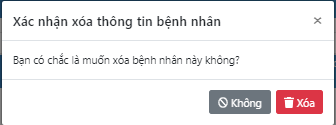
Hình 67. Giao diện thêm thông tin bệnh nhân

* **Giao diện Cập nhật thông tin bệnh nhân**

****

Hình 68. Giao diện cập nhật thông tin bệnh nhân

* **Giao diện Xóa thông tin bệnh nhân**

****

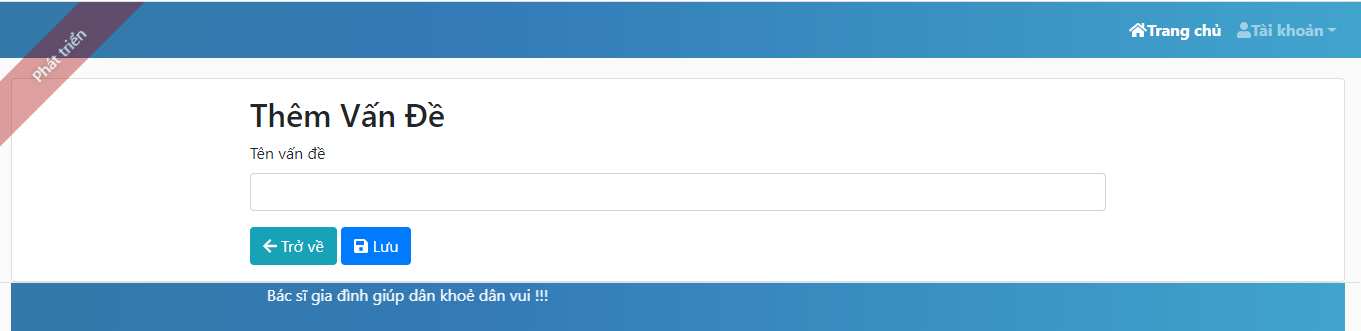
Hình 69. Giao diện xóa thông tin bệnh nhân

* **Giao diện Quản lý vấn đề hiện tại**

****

Hình 70. Giao diện Quản lý vấn đề hiện tại

* **Giao diện Thêm vấn đề hiện tại**

****

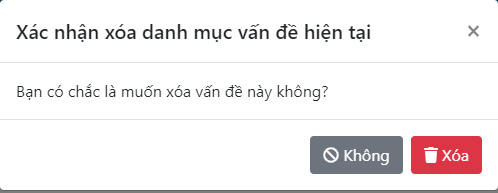
Hình 71. Giao diện thêm vấn đề hiện tại

* **Giao diện Cập nhật vấn đề hiện tại**

****

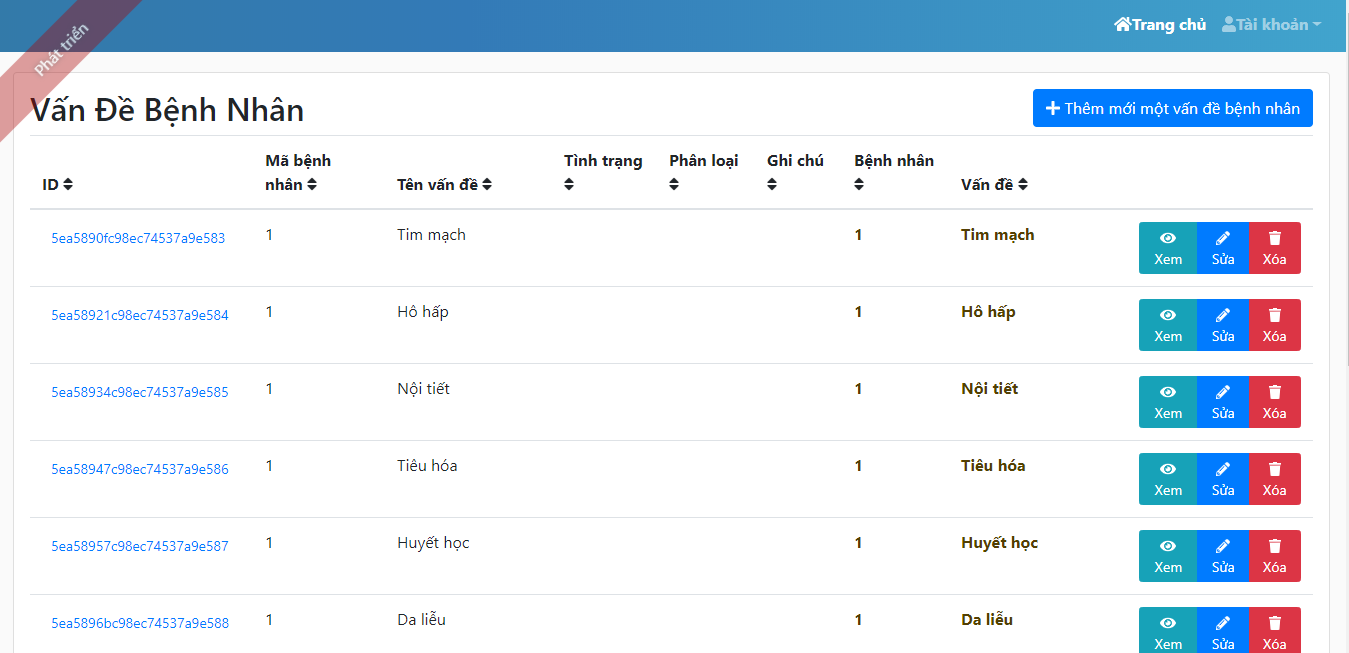
Hình 72 .Giao diện cập nhật vấn đề hiện tại

* **Giao diện Xóa vấn đề hiện tại**

****

Hình 73. Giao diện xóa vấn đề hiện tại

* **Giao diện vấn đề hiện tại của bệnh nhân**

****

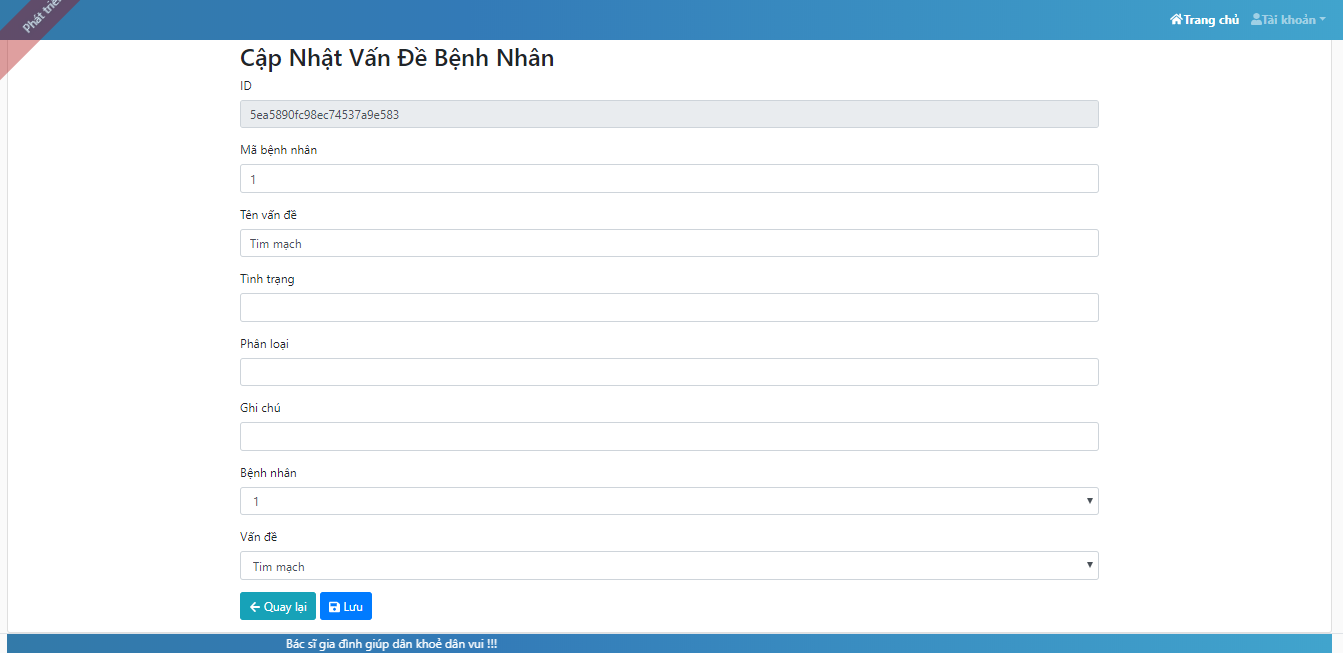
Hình 74. Giao diện vấn đề hiện tại của bệnh nhân

* **Giao diện Thêm vấn đề hiện tại của bệnh nhân**

****

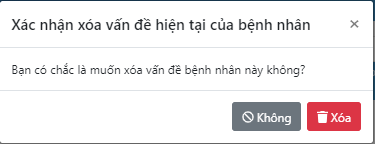
Hình 75. Giao diện thêm vấn đề hiện tại của bệnh nhân

* **Giao diện Cập nhật vấn đề hiện tại của bệnh nhân**

****

Hình 76. Giao diện cập nhật vấn đề hiện tại của bệnh nhân

* **Giao diện Xóa vấn đề hiện tại của bệnh nhân**

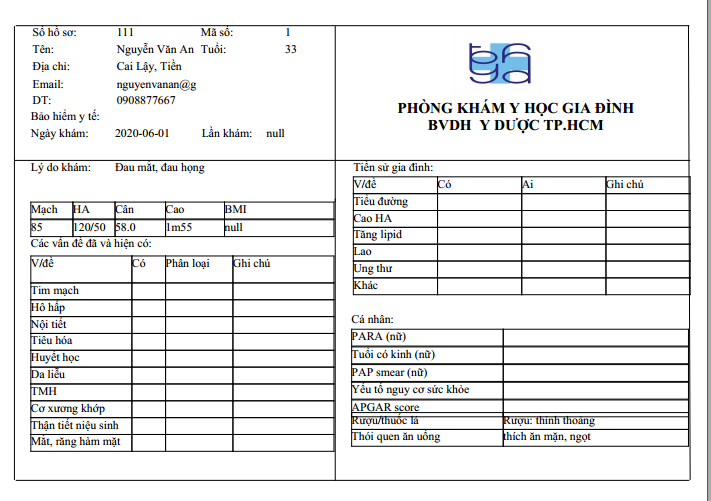
****

Hình 77. Giao diện xóa vấn đề hiện tại của bệnh nhân

* **Giao diện in hồ sơ bệnh án bác sĩ gia đình**

****

Hình 78. Giao diện in hồ sơ bệnh án bác sĩ gia đình

****

Hình 79. Hồ sơ bệnh án sau khi được in từ giao diện hồ sơ bệnh án bác sĩ gia đình

# Chương 4: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

## **4.1. Kết quả đạt được**

Kết quả Tìm hiểu lý thuyết:

* Hiểu được mô hình Microservice (Jhipster Framework), củng cố lại các kiến thức đã được học và nâng cao khả năng lập trình.
* Biết thêm nhiều kiến thức về ReactJS, Redux, TypeScript, JasperReports, Spring Framework, Graph Database (Neo4J), NoSQL Database (MongoDB),…
* Tích lũy thêm kinh nghiệm về phân tích, thiết kế hệ thống theo yêu cầu.
* Hiểu và nắm được quy trình quản lý bệnh nhân theo cây phả hệ và tài liệu thông tin bác sĩ gia đình .

Kết quả Ứng dụng thực nghiệm:

* Xây dựng được ứng dụng cho hệ thống bác sĩ gia đình bao gồm: quản lý mối quan hệ các thành viên trong gia đình, truy xuất cây phả hệ ba đời, tra cứu hiển thị mối quan hệ bệnh nhân trong gia đình, quản lý lưu trữ thông tin khám chữa bệnh của bệnh nhân trong hồ sơ bệnh án bác sĩ gia đình, in các báo cáo.
* Ứng dụng liên kết được giao diện, thông tin với service Khám chữa bệnh và các service liên quan .
* Thực hiện thành công chức năng cơ bản đã đặt ra.
* Ứng dụng được chạy thử nghiệm và kiểm tra.

## **4.2. Hạn chế**

* Cấu hình hệ thống khá phức tạp (bao gồm nhiều cấu hình để kết nối đến Consul, Keycloak, Kafka, đến database, các service khác và cấu hình các thư viện hỗ trợ) và liên kết các service từ nhiều Máy chủ trung gian nên có thể bị ảnh hưởng bởi đường truyền Internet.
* Tiếp cận nhiều công nghệ mới cùng lúc nên có thể còn những hạn chế, chưa thể phát huy tối đa hiệu quả mà các công nghệ đó mang lại.

## **4.3. Hướng phát triển**

Xây dựng chức năng truy xuất lý lịch bệnh án, tiền sử các vấn đề theo mốc thời gian từ service Hồ sơ bệnh án và các service khác khi chọn một bệnh nhân trong đồ thị, tính toán tỉ lệ các bệnh di truyền.

# TÀI LIỆU THAM KHẢO

[1]. Trịnh Minh Cường(11/10/2015), Giới thiệu về Microservices (kiến trúc nhiều dịch vụ nhỏ), [https://techmaster.vn/posts/33594/gioi-thieu-ve-Microservices](https://techmaster.vn/posts/33594/gioi-thieu-ve-microservices), <https://Microservices.io/>, <https://medium.com/coding-for-fun/gi%E1%BB%9Bi-thi%E1%BB%87u-v%E1%BB%81-m%C3%B4-h%C3%ACnh-ki%E1%BA%BFn-tr%C3%BAc-microservice-59c637f956c7>

[2]. Do KhanhToan(30/05/2017), Giới thiệu về ReactJS - Phần I (Các khái niệm cơ bản), <https://viblo.asia/p/gioi-thieu-ve-reactjs-phan-i-cac-khai-niem-co-ban-V3m5WzjblO7>, <https://viblo.asia/p/reactjs-cho-nguoi-moi-bat-dau-LzD5dP9e5jY>

[3]. Chris Ho (09/09/2017), Redux thật là đơn giản! (Phần 1), [https://insights.innovatube.com/redux-redux-thật-là-đơn-giản-phần-1-76a3fa2c31ab](https://insights.innovatube.com/redux-th%E1%BA%ADt-l%C3%A0-%C4%91%C6%A1n-gi%E1%BA%A3n-ph%E1%BA%A7n-1-76a3fa2c31ab),

[4].TypeScript – Overview, <https://www.tutorialspoint.com/typescript/typescript_overview.htm>,

[5]. Stackjava(19/12/2017), Tổng quan về Spring Framework, <https://viblo.asia/p/tong-quan-ve-spring-framework-YWOZryEyKQ0>

[6], HoangManh(20/03/2019), Giới thiệu về JHipster Stack, <https://viblo.asia/p/gioi-thieu-ve-jhipster-stack-6J3ZgmdL5mB>, https://www.jhipster.tech/

[7]. Các tài liệu của JasperReports, <http://chingovan.blogspot.com/2017/06/jasperreport-gioi-thieu-ve-jasperreport.html>, <https://www.baeldung.com/spring-jasper>, <https://techblog.vn/jasperreport-la-gi>

[8]. Tài liệu của SpringBoot, <https://docs.spring.io/spring-data/rest/docs/current/reference/html/#repository-resources.collection-resource>

[9]. Tài liệu MongoDB, <https://docs.mongodb.com/>, <https://viblo.asia/p/tim-hieu-ve-mongodb-4P856ajGlY3>, <https://viblo.asia/p/mongodb-va-nhung-dieu-co-ban-nhat-mongodb-phan-1-naQZRX4m5vx>

[10].Tài liệu về Graph database <http://blog.co-mit.com/post/40/Graph+Database>

[11]. Tài liệu Neo4J các chương <https://neo4j.com/docs/>

[12]. Diễn đàn trao đổi code Stackoverflow <https://stackoverflow.com/>

[13]. Hoàng Nguyễn(27/8/2018), Lifecycle Component trong ReactJS, <https://viblo.asia/p/lifecycle-component-trong-reactjs-gGJ59jzxKX2>