|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI



Nguyễn Văn Quyết

Xây dựng chương trình hỗ trợ quản lý đề xuất/phê duyệt tại một doanh nghiệp

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2020

|  |  |
| --- | --- |
| **bỘ GIÁO DỤC VÀ ĐÀO TẠO** | **BỘ NÔNG NGHIỆP VÀ PTNT** |

TRƯỜNG ĐẠI HỌC THỦY LỢI

Nguyễn Văn Quyết

Xây dựng chương trình hỗ trợ quản lý đề xuất/Phê DUYỆT tại một doanh nghiệp

|  |  |
| --- | --- |
| Ngành : | CNTT |
| Mã số: | 7480201 |

|  |  |
| --- | --- |
| NGƯỜI HƯỚNG DẪN | 1. Th.S Nguyễn Văn Thẩm |
|  |  |

HÀ NỘI, NĂM 2020

|  |  |
| --- | --- |
| Description: Logo-WRU | CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM  **Độc lập - Tự do - Hạnh phúc**  ----------★----------  **NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP** |

**Họ tên sinh viên:** NGUYỄN VĂN QUYẾT **Hệ đào tạo**: Đại học chính quy

**Lớp:**58TH2 **Ngành:** Công nghệ thông tin

**Khoa:** Công nghệ thông tin

**1- TÊN ĐỀ TÀI:**

Xây dựng chương trình hỗ trợ quản lý đề xuất/PHÊ DUYỆT tại một doanh nghiệp

**2- CÁC TÀI LIỆU CƠ BẢN:**

|  |  |  |  |
| --- | --- | --- | --- |
| [1] |  | TS. Đặng Thị Thu Hiền, *Cơ sơ dữ liệu*.: NXB Giao Thông Vận Tải, 2013. | TS. Đặng Thị Thu Hiền, *Cơ sơ dữ liệu*.: NXB Giao Thông Vận Tải, 2013. |
| [2] |  | VNU-UET, *Giáo trình lập trình hướng đối tượng với java, 2014*. | VNU-UET, Giáo trình lập trình hướng đối tượng với java, 2014. |
| [3] |  | O Reilly, Head First Java 2nd Edition, 2010. | O Reilly, Head First Java 2nd Edition, 2010. |
| [4] |  | [Elisabeth Freeman](https://www.google.com/search?sa=X&biw=1366&bih=635&sxsrf=ALeKk00FYxmyDoJrQg7hRqerBgTOc_5AEw:1609278505463&q=Elisabeth+Robson&stick=H4sIAAAAAAAAAOPgE-LSz9U3MKksNLKIV0KwS7RkspOt9JPy87P1y4syS0pS8-LL84uyrRJLSzLyixaxCrjmZBYnJqWWZCgE5ScV5-ftYGUEANbP7lBOAAAA&ved=2ahUKEwjemqbNlfTtAhWVyYsBHe3VCdUQmxMoATAXegQIEhAD), Head First Design Patterns, 2004 | [Online]. <https://viblo.asia/p/doi-dieu-ve-mo-hinh-mvc-E375z0vJZGW> |
| [5] |  | [Craig Walls](https://www.google.com/search?sa=X&biw=1366&bih=635&sxsrf=ALeKk03PGa7o9GylTEOxHsTEf5CQNJQeRg:1609278581737&q=craig+walls&stick=H4sIAAAAAAAAAOPgE-LRT9c3NErKNTE0KkpS4tLP1TcwqcrOTivWkslOttJPys_P1i8vyiwpSc2LL88vyrZKLC3JyC9axMqdXJSYma5QnpiTU7yDlREAOTdvqUsAAAA&ved=2ahUKEwinzdXxlfTtAhWRF6YKHVUzB2cQmxMoATAPegQIDxAD), Spring in Action (In Action series), 2005 | [Online]. <https://loda.me/spring-boot/> |
| [6] |  | [Gavin King](https://www.google.com/search?sa=X&biw=1366&bih=635&sxsrf=ALeKk00dy2YVSYqlnqffP696vhJ9ppO_tw:1609278682711&q=java+persistence+with+hibernate+gavin+king&stick=H4sIAAAAAAAAAOPgE-LRT9c3NDSuKEuyMM9RgvLKK4oLLVO0ZLKTrfST8vOz9cuLMktKUvPiy_OLsq0SS0sy8osWsWplJZYlKhSkFhVnFgMlk1MVyjNLMhQyMpNSi_ISS1IV0hPLMvMUsjPz0newMgIAlmeO62wAAAA&ved=2ahUKEwiJ0-ihlvTtAhVm72EKHddzA0sQmxMoATAPegQIEBAD), Java Persistence with Hibernate, 2006 | [Online]. <https://viettuts.vn/hibernate> |

x

x

**3 - NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:**

|  |  |
| --- | --- |
| **Nội dung cần thuyết minh** | **Tỷ lệ %** |
| Chương 1: Cơ sở lý thuyết   * Tổng quan bài toán * Giải pháp thực hiện | 25% |
| Chương 2: Phân tích và thiết kế chương trình   * Phân tích hệ thống * Thiết kế hệ thống | 60% |
| Chương 3: Thử nghiệm chương trình | 15% |

**4 - GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN**

|  |  |
| --- | --- |
| **Phần** | **Họ tên giáo viên hướng dẫn** |
| Chương 1: Cơ sở lý thuyết   * Tổng quan bài toán * Giải pháp thực hiện | Th.S Nguyễn Văn Thẩm |
| Chương 2: Phân tích và thiết kế chương trình   * Phân tích hệ thống * Thiết kế hệ thống | Th.S Nguyễn Văn Thẩm |
| Chương 3: Thử nghiệm chương trình | Th.S Nguyễn Văn Thẩm |

**5 - NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP**

Ngày ....... tháng ....... năm 2020

|  |  |
| --- | --- |
| **Trưởng Bộ môn**  *(Ký và ghi rõ Họ tên)* | **Giáo viên hướng dẫn chính**  *(Ký và ghi rõ Họ tên)* |

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày…. tháng….năm 2020 **Chủ tịch Hội đồng**

*(Ký và ghi rõ Họ tên)*

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày.... tháng .... năm 2020.

**Sinh viên làm Đồ án tốt nghiệp**

*(Ký và ghi rõ Họ tên)*

|  |  |
| --- | --- |
| logo | TRƯỜNG ĐẠI HỌC THUỶ LỢI  **KHOA CÔNG NGHỆ THÔNG TIN**  BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP |

**TÊN ĐỀ TÀI**: Xây dựng chương trình hỗ trợ quản lý đề xuất/phê duyệt tại một doanh nghiệp.

*Sinh viên thực hiện*: Nguyễn Văn Quyết

*Lớp*: 58TH2  
*Giáo viên hướng dẫn*: Nguyễn Văn Thẩm

**TÓM TẮT ĐỀ TÀI**

**1. Phát biểu bài toán:**

“Nhân lực” là tài sản quan trọng nhất mà một doanh nghiệp có. Sự thành công của doanh nghiệp phụ thuộc vào tính hiệu quả của cách “quản lý nhân sự” của doanh nghiệp - bao gồm cả cách quản lý nhân viên, chấm công, quản lý tiền lương và đặc biệt là quản lý các đề xuất/phê duyệt cho nhân viên. Điều này đem lại những giá trị góp phần vào thành công của doanh nghiệp. Tuy nhiên trong thực tế hiện nay, ở một số các công ty vừa và lớn số lượng nhân viên rất nhiều và các quy trình, thủ tục làm đề nghị/đề xuất của nhân viên vẫn dựa theo tính chất thủ tục, giấy tờ rườm rà, gây ra tình trạng mất thời gian trong việc tạo cũng như phê duyệt đề nghị/đề xuất. Vì vậy, sử dụng công nghệ thông tin để số hóa các quy trình thủ tục đề xuất/phê duyệt là một nhu cầu rất lớn của bất kì doanh nghiệp nào.

Nắm được nhu cầu cấp hiện nay của các doanh nghiệp, cộng với sự phát triển của công nghệ thời đại 4.0. Có rất nhiều sản phẩm phần mềm quản lý đề xuất/phê duyệt đã được ra đời có thể kể đến như Base Request, FastWork Request,… Tuy vậy nhưng các phần mềm kể trên vẫn chưa hỗ trợ số hóa một số thủ tục liên quan đến tiền mặt cần đến văn bản giấy tờ như đề nghị mua sắm vật tư, tạm ứng, thanh toán,… Vậy để kế thừa, hoàn thiện các sản phẩm phần mềm trên thị trường cũng như giải quyết triệt để vấn đề số hóa đề xuất/phê duyệt trong doanh nghiệp, em quyết định chọn đề tài với tên “**Xây dựng chương trình hỗ trợ quản lý đề xuất/phê duyệt tại một doanh nghiệp”** với tính năng mới làtự động tạo văn bản đề nghị thanh toán kèm theo chữ kí.

**2. Giải pháp công nghệ**

Sự ra đời của các công nghệ web như: ASP dotNET, Python, PHP … và đặc biệt SPRING MVC cũng là một trong những công cụ không thể thiếu để đáp ứng yêu cầu của việc xây dựng các ứng dụng web. SPRING MVC cũng như SPRING framwork là một nền tảng ứng dụng web (web application framework) được phát triển và cung cấp bởi Pivotal Software, cho phép những người lập trình tạo ra những trang web động, những [ứng dụng web](https://vi.wikipedia.org/wiki/%E1%BB%A8ng_d%E1%BB%A5ng_web) và những [dịch vụ web](https://vi.wikipedia.org/wiki/D%E1%BB%8Bch_v%E1%BB%A5_web). Cùng với các công nghệ phát triển web, MySQL (Structure Query Language Server) một công cụ quản lý dữ liệu được sử dụng phổ biến ở nhiều lĩnh vực. Hầu hết các ngôn ngữ bậc cao đều có trình hỗ trợ SQL như Java, C#,…

**MỤC TIÊU CỦA ĐỒ ÁN.**

- Vận dụng kiến thức đã học phân tích thiết kế một bài toán trong thực tế.

- Vận dụng kiến thức đã về Java phát triển ứng dụng.

- Tìm hiểu, nghiên cứu và vận dụng các công nghệ web phổ biến hiện nay như Spring MVC, Spring Security, Spring Data JPA, Hibernate, HTML/CSS, JS, JQuery vào trong quán trình sản xuất các phần mềm.

- Phát triển kỹ năng xây dựng và triển khai ứng dụng Web trong thực tế.

- Rèn luyện kỹ năng khảo sát, phân tích thiết kế hệ thống thông tin quản lý, tạo và quản lý cơ sở dữ liệu, lập trình ứng dụng trên nền Web.

**KẾT QUẢ CẦN ĐẠT ĐƯỢC:**

- Các báo cáo theo yêu cầu

- Thiết kế, xây dựng và cài đặt được website hỗ trợ quản lý đề xuất/phê duyệt.

- Website sẽ đảm bảo các chức năng sau cho từng đối tượng người dùng :

**+** Nhân viên (Employee) có các chức năng chính như:

\* Danh sách đề nghị/đề xuất.

\* Lịch sử phê duyệt.

\* Chi tiết các các đề nghị/đề xuất: các thông tin như mục đích, lý do, ngày xin nghỉ, về sớm bao nhiêu phút,…

+ Quản lý cấp phòng (Manager) sẽ có các chức năng chính như:

\* Thêm sửa xoá nhân viên trong doanh nghiệp.

\* Phê duyệt yêu cầu của nhân viên.

\* Danh sách và chi tiết các đề nghị/đề xuất.

+ Người điều hành (Operator) sẽ có tất cả các chức năng của manager, ngoài ra sẽ có thêm các chức năng :

\* Thêm sửa xoá thông tin các doanh nghiệp.

\* Cấu hình người manager cho từng phòng ban

\* Thêm sửa xoá thông tin các doanh nghiệp.

\* Cấu hình người manager cho từng phòng ban.

LỜI CAM ĐOAN

Tác giả xin cam đoan đây là Đồ án tốt nghiệp của bản thân tác giả. Các kết quả trong Đồ án tốt nghiệp/Khóa luận tốt nghiệp này là trung thực, và không sao chép từ bất kỳ một nguồn nào và dưới bất kỳ hình thức nào.Việc tham khảo các nguồn tài liệu (nếu có) đã được thực hiện trích dẫn và ghi nguồn tài liệu tham khảo đúng quy định.

|  |  |
| --- | --- |
|  | **Tác giả ĐATN/KLTN**  *Chữ ký*  **Nguyễn Văn Quyết** |

LỜI CÁM ƠN

Lời đầu tiên em xin chân thành cảm ơn thầy giáo, Ths. **Nguyễn Văn Thẩm**, người đã tận tình chỉ bảo, hướng dẫn em trong suốt quá trình em làm đồ án tốt nghiệp này.

Em xin cảm ơn các thầy giáo, cô giáo trường Đại học Thủy Lợi nói chung và các thầy giáo, cô giáo trong Khoa Công nghệ thông tin nói riêng đã truyền đạt cho em những kiến thức, kỹ năng vô cùng quan trọng trong thời gian em học tập, nghiên cứu tại trường. Đây thật sự là những hành trang quý báu để em bước tiếp trong con đường tương lai phía trước.

Em vô cùng biết ơn gia đình, bạn bè luôn ở bên động viên, giúp đỡ em trong học tập và đặc biệt là khi em hoàn thành đồ án tốt nghiệp này.

Do kiến thức còn hạn chế và thời gian hoàn thành không nhiều nên có thể còn thiếu sót, em rất mong nhận được sự góp ý của thầy cô và bạn bè để đồ án này được hoàn thiện hơn.

Em xin chân thành cảm ơn!

MỤC LỤC

[DANH MỤC HÌNH ẢNH iv](#_Toc59394891)

[DANH MỤC BẢNG BIỂU vi](#_Toc59394892)

[DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ vi](#_Toc59394892)

[CHƯƠNG 1 CƠ SỞ LÝ THUYẾT 1](#_Toc59394893)

[1.1 Phát biểu bài toán 1](#_Toc59394894)

[1.1.1 Đặt vấn đề 1](#_Toc59394895)

[1.1.2 Những khó khăn hạn chế của hệ thống/cách làm cũ 3](#_Toc59394896)

[1.1.3 Nhu cầu đặt ra với hệ thống mới 3](#_Toc59394897)

[1.2 Giải pháp thực hiện 4](#_Toc59394898)

[1.2.1 Mô hình lý thuyết 4](#_Toc59394899)

[1.2.1.1 Mô hình MVC 4](#_Toc59394899)

[1.2.1.2 Mô hình hướng đối tượng 8](#_Toc59394899)

[1.2.1.3 Các nguyên lý lập trình hướng đối tượng SOLID 12](#_Toc59394899)

[1.2.2 Giải pháp công nghệ 16](#_Toc59394900)

[1.2.2.1 Ngôn ngữ lập trình và Framework hỗ trợ 16](#_Toc59394899)

[1.2.2.2 Hệ quản trị cơ sở dữ liệu 20](#_Toc59394899)

[CHƯƠNG 2 PHÂN TÍCH & THIẾT KẾ CHƯƠNG TRÌNH 22](#_Toc59394901)

[2.1 Phân tích hệ thống 22](#_Toc59394902)

[2.1.1 Phân tích hệ thống về chức năng 22](#_Toc59394903)

[2.1.1.1 Xác định và mô tả các tác nhân 22](#_Toc59394903)

[2.1.1.2 Xác định và mô tả các use case 22](#_Toc59394903)

[2.1.1.3 Xây dựng biểu đồ use case 22](#_Toc59394903)

[2.1.1.4 Phân tích luồng nghiệp vụ 28](#_Toc59394903)

[2.2 Thiết kế hệ thống 32](#_Toc59394905)

[2.2.1. Thiết kế hệ thống về dữ liệu 32](#_Toc59394906)

[2.2.1.1 Mô hình thực thể liên hệ 31](#_Toc59394903)

[2.2.1.2 Mô hình quan hệ 32](#_Toc59394903)

[2.2.2 Thiết kế hệ thống về chức năng 36](#_Toc59394907)

[2.2.2.1 Giao diện chính của hệ thống 35](#_Toc59394903)

[2.2.2.2 Giao diện chức năng dành cho nhân viên 37](#_Toc59394903)

[2.2.2.3 Giao diện chức năng dành cho quản lý 42](#_Toc59394903)

[2.2.2.4 Giao diện chức năng dành cho nhà điều hành 43](#_Toc59394903)

[CHƯƠNG 3 THỬ NGHIỆM CHƯƠNG TRÌNH 48](#_Toc59394908)

[3.1 Cấu hình hệ thống 48](#_Toc59394909)

[3.1.1 Yêu cầu phần cứng 48](#_Toc59394910)

[3.1.2 Môi trường chạy chương trình 48](#_Toc59394911)

[3.2 Các kết quả 48](#_Toc59394912)

[3.2.1 Các kết quả về dữ liệu 48](#_Toc59394913)

[3.2.2 Các kết quả về chức năng 49](#_Toc59394914)

[3.3 Phân tích kết quả đạt được 62](#_Toc59394915)

[TÀI LIỆU THAM KHẢO 63](#_Toc59394916)

DANH MỤC HÌNH ẢNH

[Hình 1.1 Kiến trúc MVC 5](#_Toc58051305)

[Hình 1.2 Luồng hoạt động mô hình MVC 6](#_Toc58051305)

[Hình 2.1 Bảng đề nghị 35](#_Toc58051308)

[Hình 2.2 Bảng người dùng 35](#_Toc58051309)

[Hình 2.3 Bảng phòng ban 35](#_Toc58051309)

[Hình 2.4 Bảng vật tư 35](#_Toc58051309)

[Hình 2.5 Bảng đề nghị vật tư 36](#_Toc58051309)

[Hình 2.6 Bảng quyền 36](#_Toc58051309)

[Hình 2.7 Bảng quyền người dùng 36](#_Toc58051309)

[Hình 2.8 Thiết kế giao diện màn hình đăng nhập 37](#_Toc58051309)

[Hình 2.9 Thiết kế giao diện màn hình chính 38](#_Toc58051309)

[Hình 2.10 Thiết kế giao diện màn hình danh sách đề nghị cá nhân 39](#_Toc58051309)

[Hình 2.11 Thiết kế giao diện màn hình tạo đề nghị xin nghỉ phép 40](#_Toc58051309)

[Hình 2.12 Thiết kế giao diện màn hình tạo đề nghị vật tư 41](#_Toc58051309)

[Hình 2.13 Thiết kế file pdf của đề nghị xin nghỉ phép 41](#_Toc58051309)

[Hình 2.14 Thiết kế giao diện màn hình danh sách đề nghị phê duyệt 43](#_Toc58051309)

[Hình 2.15 Thiết kế giao diện màn hình duyệt đề nghị xin nghỉ phép 43](#_Toc58051309)

[Hình 2.16 Thiết kế giao diện màn hình duyệt đề nghị vật tư 44](#_Toc58051309)

[Hình 2.17 Thiết kế giao diện màn hình danh sách đề nghị xử lý 44](#_Toc58051309)

[Hình 2.18 Thiết kế giao diện màn hình xử lý đề nghị xin nghỉ phép 45](#_Toc58051309)

[Hình 2.19 Thiết kế giao diện màn hình xử lý đề nghị vật tư 45](#_Toc58051309)

[Hình 2.20 Thiết kế giao diện màn hình danh sách người dùng 46](#_Toc58051309)

[Hình 2.21 Thiết kế giao diện màn hình thêm mới người dùng 46](#_Toc58051309)

[Hình 2.22 Thiết kế giao diện màn hình danh sách phòng ban 47](#_Toc58051309)

[Hình 2.23 Thiết kế giao diện màn hình thêm mới phòng ban 47](#_Toc58051309)

[Hình 2.23 Thiết kế giao diện màn hình thêm mới phòng ban 48](#_Toc58051309)

[Hình 3.2 Dữ liệu thử nghiệm bảng người dùng 49](#_Toc58051309)

[Hình 3.3 Dữ liệu thử nghiệm bảng vật tư 49](#_Toc58051309)

[Hình 3.4 Dữ liệu thử nghiệm bảng phòng ban 49](#_Toc58051309)

[Hình 3.5 Cấu trúc của mã nguồn code 50](#_Toc58051309)

[Hình 3.6 Màn hình danh sách đề nghị cá nhân 51](#_Toc58051309)

[Hình 3.7 Màn hình tạo đề nghị xin nghỉ phép 52](#_Toc58051309)

[Hình 3.8 Màn hình tạo đề nghị vật tư 53](#_Toc58051309)

[Hình 3.9 File pdf nhân viên nhận được sau khi cấp trên đã duyệt đề nghị vật tư 54](#_Toc58051309)

[Hình 3.10 Màn hình phê duyệt đề nghị xin nghỉ phép 54](#_Toc58051309)

[Hình 3.11 Màn hình phê duyệt đề nghị vật tư 55](#_Toc58051309)

[Hình 3.12 Màn hình danh sách DNDX chờ duyệt 56](#_Toc58051309)

[Hình 3.13 Màn hình danh sách DNDX chờ xử lý 57](#_Toc58051309)

[Hình 3.14 Màn hình xử lý đề nghị vật tư 58](#_Toc58051309)

[Hình 3.15 Màn hình xử lý đề nghị xin nghỉ phép 59](#_Toc58051309)

[Hình 3.16 Màn hình danh sách người dùng 60](#_Toc58051309)

[Hình 3.17 Màn hình thêm mới người dùng 60](#_Toc58051309)

[Hình 3.18 Màn hình danh sách phòng ban 61](#_Toc58051309)

[Hình 3.19 Màn hình thêm mới phòng ban 61](#_Toc58051309)

DANH MỤC BẢNG BIỂU

[Bảng 2.1 Các tác nhân và mô tả 23](#_Toc58051440)

[Biểu đồ 2.1 Phân rã chức năng của hệ thống 23](#_Toc58051441)

[Biểu đồ 2.2 Use case toàn bộ hệ thống 24](#_Toc58051441)

[Biểu đồ 2.3 Phân rã chức năng của nhân viên 25](#_Toc58051442)

[Biểu đồ 2.4 Phân rã chức năng của quản lý 26](#_Toc58051443)

[Biểu đồ 2.5 Phân rã chức năng của nhà điều hành 27](#_Toc58051444)

[Biểu đồ 2.6 Biểu đồ hoạt động của hệ thống 28](#_Toc58051444)

[Biểu đồ 2.7 Biểu đồ trạng thái đề nghị 29](#_Toc58051444)

[Biểu đồ 2.6 Mô hình thực thể liên hệ 31](#_Toc58051445)

DANH MỤC CÁC TỪ VIẾT TẮT VÀ GIẢI THÍCH CÁC THUẬT NGỮ

**CSDL** Cơ sở dữ liệu

**DATN** Đồ án tốt nhiệp

**DNDX** Đề nghị đề xuất

**MVC** Kiến trúc MVC(Model-View-Controller)

# CHƯƠNG 1 CƠ SỞ LÝ THUYẾT

## Phát biểu bài toán

### Đặt vấn đề

Trong thực tế hiện nay, “Nhân sự” là tài sản quan trọng nhất mà một doanh nghiệp có. Nhân sự bao gồm tất cả các tiềm năng của [con người](https://vi.wikipedia.org/wiki/Lo%C3%A0i_ng%C6%B0%E1%BB%9Di) trong một tổ chức hay xã hội (kể cả những thành viên trong ban lãnh đạo doanh nghiệp) tức là tất cả các thành viên trong doanh nghiệp sử dụng kiến thức, khả năng, hành vi ứng xử và giá trị đạo đức để thành lập, duy trì và phát triển doanh nghiệp.

Mỗi hình thái [kinh tế xã hội](https://vi.wikipedia.org/wiki/Kinh_t%E1%BA%BF_x%C3%A3_h%E1%BB%99i) đều gắn liền với một [phương thức sản xuất](https://vi.wikipedia.org/wiki/Ph%C6%B0%C6%A1ng_th%E1%BB%A9c_s%E1%BA%A3n_xu%E1%BA%A5t) nhất định, xu hướng của [quản trị](https://vi.wikipedia.org/wiki/Qu%E1%BA%A3n_tr%E1%BB%8B) ngày càng phức tạp cùng với sự phát triển ngày càng cao của nền kinh tế xã hội. Trên thị trường ngày nay, các doanh nghiệp đang đứng trước thách thức phải tăng cường tối đa hiệu quả cung cấp các sản phẩm dịch vụ của mình. Điều này đòi hỏi phải có sự quan tâm tới chất lượng sản phẩm và [dịch vụ](https://vi.wikipedia.org/wiki/D%E1%BB%8Bch_v%E1%BB%A5), tới các phương thức [Marketing](https://vi.wikipedia.org/wiki/Marketing) và bán hàng tốt cũng như các quy trình nội bộ hiệu quả. Các doanh nghiệp hàng đầu thường cố gắng để tạo sự cân bằng giữa tính chất nhất quán và sự sáng tạo. Để đạt được mục tiêu này, họ dựa vào một số tài sản lớn nhất của mình đó chính là “nguồn nhân lực”.

Công tác quản trị nhân sự giúp tìm kiếm, phát triển và duy trì đội ngũ [nhân viên](https://vi.wikipedia.org/w/index.php?title=Nh%C3%A2n_vi%C3%AAn&action=edit&redlink=1) và quản lý chất lượng - những người tham gia tích cực vào sự thành công của công ty. Các tổ chức trông mong vào các nhà chuyên môn về quản trị nhân sự giúp họ đạt được hiệu quả và năng suất cao hơn với một hạn chế về lực lượng lao động. Một trong những yêu cầu chính của quản trị nhân sự là tìm ra đúng người, đúng số lượng và đúng thời điểm trên các điều kiện thỏa mãn cả doanh nghiệp và nhân viên mới. Khi lựa chọn được những người có kỹ năng thích hợp làm việc ở đúng vị trí thì cả nhân viên lẫn công ty đều có lợi.

Quản lý nhân sự được hiểu là những tác động hợp quy luật của chủ thể quản lý nhân sự đến các khách thể quản lý nhân sự nhằm đạt được các mục tiêu quản lý nhân sự đã đề ra.

Quản lý nhân sự là lĩnh vực theo dõi, hướng dẫn điều chỉnh, kiểm tra sự trao đổi chất (năng lượng, thần kinh, bắp thịt,...) giữa con người và các yếu tố vật chất của tự nhiên (công cụ, đối tượng lao động, năng lượng,...) trong quá trình tạo của cải vật chất, tinh thần để thỏa mãn nhu cầu của con người nhằm duy trì, bảo vệ, sử dụng và phát triển tiềm năng vô hạn của con người. Không một hoạt động nào của tổ chức mang lại hiệu quả nếu thiếu quản lý nhân sự. Hay nói cách khác, mục tiêu của bất kỳ tổ chức nào cũng là nhằm sử dụng một cách có hiệu quả nhân sự của tổ chức để đạt mục tiêu.

Sự tồn tại và phát triển của một doanh nghiệp phụ thuộc rất lớn vào việc khai thác và sử dụng có hiệu quả các nguồn lực: vốn, cơ sở vật chất, tiến bộ [khoa học kỹ thuật](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_k%E1%BB%B9_thu%E1%BA%ADt), người lao động, các yếu tố này có mối quan hệ mật thiết với nhau và tác động lại với nhau. Những yếu tố như: máy móc thiết bị, của cải vật chất, công nghệ kỹ thuật đều có thể mua được, học hỏi được, sao chép được, nhưng con người thì không thể. Vì vậy có thể khẳng định rằng quản trị nhân sự có vai trò thiết yếu đối với sự tồn tại và phát triển của doanh nghiệp .

Con người, với khả năng sáng tạo, lao động cần cù siêng năng của mình, đã phát minh ra những dụng cụ từ ban đầu là thô sơ cho đến phát triển công nghệ cao, khoa học kỹ thuật cao như ngày nay đã phục vụ được nhu cầu bản thân và phát triển xã hội.

Quản trị nhân sự góp phần vào việc giải quyết các mặt kinh tế xã hội của vấn đề lao động. Đó là một vấn đề chung của xã hội, mọi hoạt động kinh tế nói chung đều đi đến một mục đích sớm hay muộn là làm sao cho người lao động hưởng thành quả do họ làm ra.

Quản trị nhân sự gắn liền với mọi tổ chức, bất kỳ một cơ quan tổ chức nào cũng cần phải có bộ phận nhân sự. Quản trị nhân sự là một thành tố quan trọng của chức năng quản trị, nó có gốc rễ và các nhánh trải rộng khắp nơi trong mọi tổ chức. Quản trị nhân sự hiện diện ở khắp các phòng ban, bất cứ cấp quản trị nào cũng có nhân viên dưới quyền vì thế đều phải có quản trị nhân sự. Cung cách quản trị nhân sự tạo ra bầu không khí văn hoá cho một doanh nghiệp . Đây cũng là một trong những yếu tố quyết định đến sự thành bại của một doanh nghiệp .

Với sự phát triển bùng nổ của cuộc cách mạng 4.0 hiện nay, bài toán quản lý nhân sự, hay cụ thể là quản lý, phê duyệt đề nghị/đề xuất nhân sự đang trở thành một cơ hội cũng như thách thức đối với những công ty công nghệ, phần mềm.

### Những khó khăn hạn chế của hệ thống/cách làm cũ

Trong thực tế hiện nay, ở một số các công ty vừa và lớn số lượng nhân viên rất nhiều và các quy trình, thủ tục làm đề nghị/đề xuất của nhân viên vẫn dựa theo tính chất thủ tục, giấy tờ rườm rà, gây ra tình trạng mất thời gian trong việc tạo cũng như phê duyệt đề nghị/đề xuất. Về phía người quản lý cũng khó khan trong việc cập nhật theo dõi các đề nghị/đề xuất của nhân viên.

### Nhu cầu đặt ra với hệ thống mới

Với nhu cầu này, các công ty và tập đoàn rất cần một phần mềm quản lý có thể số hóa quy trình, thủ tục đề nghị/phê duyệt, cụ thể là:

- Nhân viên:

+ Tạo đề nghị/ đề xuất cá nhân.

+ Theo dõi quá trình quản lý phê duyệt.

- Người quản lý:

+ Phê duyệt DNDX của nhân viên chịu sự quản lý của mình.

+ Xem chi tiết tất cả các DNDX trong phòng mình.

- Người quản lý:

+ Cập nhật trạng thái thực hiện của DNDX.

+ Quản lý nhân viên, phòng ban.

Từ đó giúp tăng hiệu quả cho công việc quản lý, kinh doanh và sản xuất cho các công ty/ tập đoàn.

## Giải pháp thực hiện

### Mô hình lý thuyết

#### Mô hình MVC

**Định nghĩa**: MVC là từ viết tắt của 'Model View Controller'. Nó đại diện cho các nhà phát triển kiến trúc áp dụng khi xây dựng các ứng dụng. Với kiến trúc MVC, chúng ta xem xét cấu trúc ứng dụng liên quan đến cách luồng dữ liệu của ứng dụng của chúng ta hoạt động như thế nào.

Dễ hiểu hơn, nó là mô hình phân bố source code thành 3 phần, mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.

**Các thành phần:**

Khối mô hình(Model): là nơi chứa những nghiệp vụ tương tác với dữ liệu hoặc hệ quản trị cơ sở dữ liệu (mysql, oracle… ); nó sẽ bao gồm các class/function xử lý nhiều nghiệp vụ như kết nối database, truy vấn dữ liệu, thêm – xóa – sửa dữ liệu…Bộ phận này là một cầu nối giữa 2 thành phần bên dưới là View và Controller. Model thể hiện dưới hình thức là một cơ sở dữ liệu hoặc có khi chỉ đơn giản là một [file XML](https://monamedia.co/file-xml-la-gi/) bình thường.

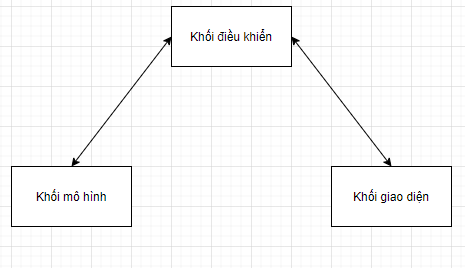
Khối giao diện(View): là nới chứa những giao diện như một nút bấm, khung nhập, menu, hình ảnh… nó đảm nhiệm nhiệm vụ hiển thị dữ liệu và giúp người dùng tương tác với hệ thống. Đây là phần giao diện (theme) dành cho người sử dụng. Nơi mà người dùng có thể lấy được thông tin dữ liệu của MVC thông qua các thao tác truy vấn như tìm kiếm hoặc sử dụng thông qua các website. Thông thường, các ứng dụng web sử dụng MVC View như một phần của  hệ thống, nơi các thành phần [HTML](https://monamedia.co/html-va-css-la-gi/) được tạo ra. Bên cạnh đó, View cũng có chức năng ghi nhận hoạt động của  người dùng để tương tác với Controller. Tuy nhiên, View không có mối quan hệ trực tiếp với Controller, cũng không được lấy dữ liệu từ Controller mà chỉ hiển thị yêu cầu chuyển cho Controller mà thôi.

Ví dụ: Nút “delete” được tạo bởi View khi người dùng nhấn vào nút đó sẽ có một hành động trong Controller.

Khối điều khiển(Controller): là nơi tiếp nhận những yêu cầu xử lý được gửi từ người dùng, nó sẽ gồm những class/ function xử lý nhiều nghiệp vụ logic giúp lấy đúng dữ liệu thông tin cần thiết nhờ các nghiệp vụ lớp Model cung cấp và hiển thị dữ liệu đó ra cho người dùng nhờ lớp View.

Ví dụ: Chức năng thêm bài viết mới trong trang quản trị website. Nơi đây có 2 trường nhập về tiêu đề và nội dung bài viết, thì trong đó:

* View sẽ hiển thị ra phần nhập form tiêu đề và nội dung.
* Controller lấy dữ liệu từ 2 trường và gửi tới Model.
* Model lúc này sẽ nhận dữ liệu từ Controller để lưu trữ trong cơ sở dữ liệu.



Hình 1.1 Kiến trúc MVC

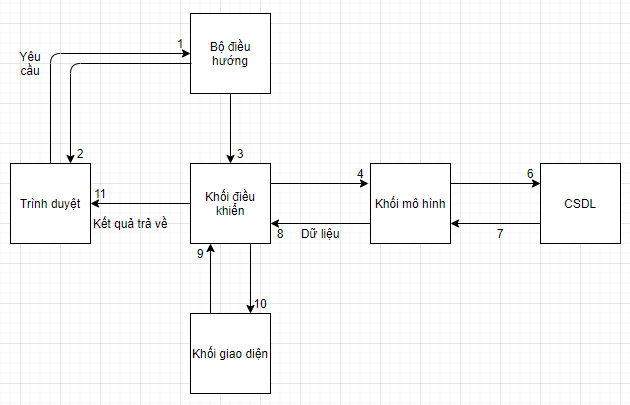
**Sự tương tác giữa các thành phần:**

Khối điều khiển(Controller) tương tác với qua lại với khối giao diện(View).

Khối điều khiển(Controller) tương tác qua lại với khối mô hình(Model).

Khối mô hình(Model) và khối giao diện(View) không có sự tương tác với nhau mà nó tương tác với nhau thông qua khối điều khiển(Controller).

Mỗi bộ phận thực hiện chức năng nhất định, nhưng chúng có sự thống nhất, liên kết với nhau tạo nên **mô hình MVC**. Mô hình này tương đối nhẹ. Nó có thể tích hợp được nhiều tính năng có trong ASP.NET hiện giờ. Ví dụ như authentication (quá trình xác thực).



Hình 1.2 Luồng hoạt động mô hình MVC

**Kịch bản 1**: Người dùng chỉ gửi yêu cầu chuyển từ trang hiện tại sang một trang khác của web không có yêu cầu về dữ liệu.

Nếu như người lập trình thực hiện việc chuyển hướng ở khối điều khiển thì lúc này luồng hoạt động là: 1 -> 3 -> 11. Yêu cầu được gửi từ trình duyệt (Browser) đến bộ điều hướng (nơi định tuyến hay phân định các yêu cầu sẽ được xử lý ở đâu là trực tiếp tại bộ định tuyến hay là tại khối điều khiển nào đó).

Nếu người lập trình để việc chuyển hướng trang ở trực tiếp tại bộ định tuyến thì luồng hoạt động của nó là: 1 -> 2 yêu cầu chuyển trang sẽ được xử lý ngay tại bộ định tuyến mà không cần gọi đến khối điều khiển.

**Kịch bản 2**: Người dùng gửi 1 yêu cầu chuyển hướng về một trang khác của trang web có trả về dữ liệu (ví dụ như xem danh sách các Tour đã đặt).

Luồng hoạt động của nó là: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 10 -> 9 > 11. Sau khi yêu cầu gửi về bộ định tuyến được bộ định tuyến chuyển về xử lý tại khối điều khiển, lúc này các yêu cầu liên quan đến dữ liệu sẽ được xử lý và khối điều khiển sẽ có sự tương tác với khối mô hình để lấy dữ liệu, Khối điều khiển sẽ sử dụng các lớp/hàm trong khối mô hình cần thiết để lấy ra những dữ liệu chính xác. Khối mô hình tương tác với cơ sở dữ liệu để lấy dữ liệu, dữ liệu trả về được gửi về Khối mô hình từ khối mô hình gửi lại về khối điều khiển, khối điều khiển gọi đến Khối giao diện phù hợp với yêu cầu kèm theo dữ liệu cho Khối giao diện, Khối giao diện sẽ lắp dữ liệu tương ứng vào HTML và gửi lại một HTML cho Khối điều khiển sau khi thực hiện xong nhiệm vụ của mình. Hoàn tất các công đoạn trên Khối điều khiển sẽ trả kết quả về trình duyệt.

**Kịch bản 3**: Người dùng chỉ yêu cầu dữ liệu nhưng không chuyển trang

Luồng hoạt động của nó là: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 11. Thứ tự xử lý tương tự ở kịch bản 2 nhưng đến khi khối điều khiển nhận được dữ liệu trả về thì không gọi đến Khối giao diện mà trả kết quả trả lại cho trình duyệt thông qua API, dữ liệu trả về thường sẽ là dạng JSON.

=> Trên đây là 1 số kịch bản thường xảy ra với yêu cầu ở client gửi lên Server và được xử lý theo mô hình MVC.

**Ưu điểm:**

Kiểm tra dễ dàng: Với **MVC**, bạn có thể dễ dàng kiểm tra, rà soát lỗi phần mềm trước khi tới tay người tiêu dùng, đảm bảo chất lượng và độ uy tín cao hơn.

Chức năng control: Trên các nền website thì ngôn ngữ lập trình như CSS, HTML, [Javascript](https://monamedia.co/javascript-la-gi-gioi-thieu-ve-ngon-ngu-js-cho-nguoi-moi-hoc/) có một vai trò vô cùng quan trọng. Việc sử dụng mô hình MVC sẽ giúp bạn có một bộ control ưu việt trên nền tảng các ngôn ngữ hiện đại với nhiều hình thức khác nhau.

View và size: View sẽ là nơi lưu trữ các dữ liệu. Càng nhiều yêu cầu được thực hiện thì kích thước càng tệp càng lớn. Khi  đó, đường truyền mạng cũng giảm tốc độ load. Việc sử dụng mô hình MVC sẽ giúp bạn tiết kiệm được diện tích băng thông một cách tối ưu.

Chức năng Soc (Separation of Concern): Chức năng này cho phép bạn phân tách rõ ràng các phần như Model, giao diện, data, nghiệp vụ.

Tính kết hợp: Việc tích hợp ở **mô hình MVC** cho phép bạn thoải mái viết code trên nền tảng website. Khi đó, server của bạn sẽ được giảm tải khá nhiều.

Đơn giản: Đây là một mô hình với kết cấu tương đối đơn giản. Dù bạn không có quá nhiều chuyên môn cũng có thể sử dụng được.

**Nhược điểm:**

MVC thường được sử dụng vào những  dự án lớn. Do đó, với các dự án nhỏ, mô hình MVC có thể gây cồng kềnh, tốn thời gian trong quá trình phát triển cũng như thời gian trung chuyển dữ liệu.

#### Mô hình hướng đối tượng

Hướng đối tượng là kĩ thuật mô hình hóa một hệ thống thế giới thực trong phần mềm dựa trên các đối tượng. Đối tượng (object) là khái niệm trung tâm của lập trình hướng đối tượng (OOP), nó là một mô hình của một thực thể hay khái niệm trong thế giới thực. Việc mô hình hóa này bao gồm xác định các đối tượng tham gia bài toán – những cái làm nhiệm vụ gì đó hoặc bị làm gì đó. Lập trình theo kiểu hướng đối tượng là hoạt động định nghĩa các thể loại của các đối tượng đó ở hình thức các khuôn mẫu để tạo ra chúng. Trong thời gian chạy, một chương trình OOP chính là một tập các đối tượng gửi thông điệp cho nhau để yêu cầu dịch vụ và thực hiện dịch vụ khi được yêu cầu. Việc một đối tượng thực hiện một dịch vụ có thể dẫn đến việc nó thay đổi trạng thái của bản thân. Một ví dụ có tính chất gần với thế giới thực: ông A đến rút tiền tại máy ATM. Ta có các đối tượng: ông A, máy ATM, cơ sở dữ liệu ngân hàng, và tài khoản của ông A. Trình tự diễn ra như sau: Ông A cho thẻ ngân hàng vào khe máy ATM; 13 đối tượng ATM yêu cầu cơ sở dữ liệu ngân hàng cung cấp đối tượng tài khoản của ông A; ông A yêu cầu rút 100.000 đồng; đối tượng ATM yêu cầu đối tượng tài khoản trừ đi 100.000 đồng. Như vậy giao dịch này bao gồm chuỗi các yêu cầu dịch vụ và việc các đối tượng thực hiện các yêu cầu đó, đồng thời thay đổi trạng thái của mình (tài khoản ông A bị bớt tiền, ông A có thêm tiền, dữ liệu nhật trình ATM có thêm thông tin về một giao dịch).

Mô hình hướng đối tượng có ba nguyên tắc trụ cột: đóng gói, thừa kế và đa hình, còn trừu tượng hóa là khái niệm nền tảng.

**Trừu tượng hóa** (abstraction) là một cơ chế cho phép biểu diễn một tình huống phức tạp trong thế giới thực bằng một mô hình được đơn giản hóa. Nó bao gồm việc tập trung vào các tính chất quan trọng của một đối tượng khi phải làm việc với lượng lớn thông tin. Ví dụ, đối với một con mèo trong ngữ cảnh một cửa hàng bán thú cảnh, ta có thể tập trung vào giống mèo, màu lông, cân nặng, tuổi, đã tiêm phòng dại hay chưa, và bỏ qua các thông tin khác như dung tích phổi, nồng độ đường trong máu, huyết áp, còn đối với một con mèo trong ngữ cảnh bệnh viện thú y thì lại là một chuyện khác. Các đối tượng ta thiết kế trong chương trình OOP sẽ là các trừu tượng hóa theo nghĩa đó, ta bỏ qua nhiều đặc điểm của đối tượng thực và chỉ tập trung vào các thuộc tính quan trọng cho việc giải một bài toán cụ thể. Người ta gọi một trừu tượng hóa là một mô hình của một đối tượng hoặc khái niệm trong thế giới thực. Trừu tượng hóa là một trong những công cụ cơ bản của tất cả các phương pháp lập trình, không chỉ lập trình hướng đối tượng. Khi viết một chương trình giải một bài toán của thế giới thực, trừu tượng hóa là một cách để mô hình hóa bài toán đó. Ví dụ, khi ta viết một chương trình quản lý sổ địa chỉ, ta sẽ dùng các trừu tượng hóa như tên, địa chỉ, số điện thoại, thứ tự bảng chữ cái, và các khái niệm liên quan tới một sổ địa chỉ. Ta sẽ định nghĩa các thao tác để xử lý dữ liệu chẳng hạn như thêm một mục tên mới hoặc sửa một địa chỉ. Trong ngữ cảnh lập trình, trừu tượng hóa là mô hình hóa thế giới thực theo cách mà nó có thể được cài đặt dưới dạng một chương trình máy tính. Phương pháp hướng đối tượng trừu tượng hóa thế giới thực thành các đối tượng và tương tác giữa chúng với các đối tượng khác. Việc mô hình hóa trở thành mô hình hóa các đối tượng tham gia bài toán – một cái nhiệt kế, một người chủ tài khoản ngân hàng, một sổ địa chỉ… mỗi đối tượng cần có đủ các thuộc tính và phương thức để thực hiện được tất cả các dịch vụ mà nó được yêu cầu.

**Đóng gói** (encapsulation): Các trừu tượng hóa của những gì có liên quan đến nhau được đóng gói vào trong một đơn vị duy nhất. Các trạng thái và hành vi của các trừu tượng hóa được bọc lại trong một khối gọi là lớp. Cụ thể, sau khi đã xác định được các đối tượng, rồi đến các thuộc tính và hành động của mỗi đối tượng, mục tiêu là đóng gói trong mỗi đối tượng các tính năng cần thiết để nó có thể thực hiện được vai trò của mình trong chương trình. Thí dụ, một đối tượng nhiệt kế cần có những gì cần thiết để có thể đo nhiệt độ, lưu trữ số liệu của các lần đo nhiệt độ trước và cho phép truy vấn các số liệu này. Định nghĩa lớp là công cụ lập trình chính yếu cho việc thực hiện nguyên tắc đóng gói. Một lớp là mô tả về một tập hợp các đối tượng có cùng các thuộc tính, hành vi.

**Thuộc tính** (attribute) dùng để lưu trữ thông tin trạng thái của một đối tượng. Một thuộc tính có thể chỉ đơn giản là một biến Boolean lưu trữ trạng thái tắt hoặc bật, hay phức tạp hơn khi chính nó lại là một đối tượng khác. Các thuộc tính được khai báo trong định nghĩa lớp và được gọi là các biến của thực thể (instance variable), gọi tắt là biến thực thể. Chúng còn được gọi là các thành viên dữ liệu (data member), hay trường (field).

**Trạng thái** (state) phản ánh các giá trị hiện tại của các thuộc tính của một đối tượng và là kết quả của hành vi của đối tượng đó theo thời gian.

**Hành vi** (behavior) là hoạt động của một đối tượng mà có thể nhìn thấy được từ bên ngoài. Trong đó có việc đối tượng thay đổi trạng thái ra sao hoặc việc nó trả về thông tin trạng thái khi nó được thông điệp yêu cầu.

**Phương thức** (method) là một thao tác hay dịch vụ được thực hiện đối với đối tượng khi nó nhận thông điệp tương ứng. Các phương thức cài đặt hành vi của đối tượng và được định nghĩa trong định nghĩa lớp. Phương thức còn được gọi bằng các cái tên khác như: hàm thành viên (member function) – gọi tắt là 'hàm', thao tác (operation), dịch vụ (service). Khái niệm đóng gói còn đi kèm với khái niệm che giấu thông tin (information hiding) nghĩa là che giấu các chi tiết bên trong của một đối tượng khỏi thế giới bên ngoài. Chẳng hạn khi dùng một cái cầu dao điện, đối với người sử dụng, nó chỉ là một cái hộp mà khi gạt cần sẽ có tác dụng ngắt và nối điện và cái hộp có khả năng tự ngắt điện khi quá tải. Người dùng không biết và không cần biết các mạch điện bên trong được thiết kế ra sao, cơ chế phát hiện quá tải như thế nào. Những chi tiết đó được giấu bên trong, còn từ bên ngoài ta chỉ nhìn thấy cầu dao là một cái hộp có cần gạt. Nói theo phương diện lập trình, nhìn từ bên ngoài một mô-đun chỉ thấy được các giao diện. Các lập trình viên tự do cài đặt chi tiết bên trong, với ràng buộc duy nhất là tuân theo giao diện đã được quy ước từ trước. Ta có thể thực hiện nguyên tắc đóng gói với tất cả các ngôn ngữ lập trình hướng đối tượng cũng như các ngôn ngữ thủ tục. Tuy nhiên, chỉ các ngôn ngữ hướng đối tượng mới cung cấp cơ chế cho phép che giấu thông tin, ngăn không cho bên ngoài truy nhập vào chi tiết bên trong của mô-đun.

**Thừa kế** (inheritance) là quan hệ mang tính phân cấp mà trong đó các thành viên của một lớp được kế thừa bởi các lớp được dẫn xuất trực tiếp hoặc gián tiếp từ lớp đó. Đây là cơ chế cho phép định nghĩa một lớp mới dựa trên định nghĩa của một lớp có sẵn, sao cho tất cả các thành viên của lớp "cũ" (lớp cơ sở hay lớp cha) cũng có mặt trong lớp mới (lớp dẫn xuất hay lớp con) và các đối tượng thuộc lớp mới có thể được sử dụng thay cho đối tượng của lớp cũ ở bất cứ đâu. Thừa kế là một hình thức tái sử dụng phần mềm, trong đó một lớp mới được xây dựng bằng cách hấp thụ các thành viên của một lớp có sẵn và bổ sung những tính năng mới hoặc sửa tính năng có sẵn. Nói cách khác, xuất phát từ một lớp mô hình hóa một khái niệm tổng quát hơn, chẳng hạn Shape, ta có thể dùng quan hệ thừa kế để xây dựng các lớp mô hình hóa các khái niệm cụ thể hơn, chẳng hạn Circle, Triangle. Bằng cách này, ta có thể sử dụng giao diện cũng như cài đặt của lớp cũ cho lớp mới.

**Đa hình** (polymorphism), theo nghĩa tổng quát, là khả năng tồn tại ở nhiều hình thức. Trong hướng đối tượng, đa hình đi kèm với quan hệ thừa kế và nó có nghĩa rằng cùng một cái tên có thể được hiểu theo các cách khác nhau tùy từng tình huống. Các đối tượng thuộc các lớp dẫn xuất khác nhau có thể được đối xử như nhau, như thể chúng là các đối tượng thuộc lớp cơ sở, chẳng hạn có thể đặt các đối tượng Triangle và Circle trong cùng một cấu trúc dữ liệu dành cho Shape, hoặc dùng cùng một lời gọi hàm rotate cho các đối tượng Triangle hay Circle. Và khi nhận được cùng một thông điệp đó, các đối tượng thuộc các lớp khác nhau hiểu nó theo những cách khác nhau. Ví dụ, khi nhận được thông điệp "rotate", các đối tượng Triangle và Amoeba thực hiện các phương thức rotate() khác nhau.

#### Các nguyên lý lập trình hướng đối tượng SOLID

SOLID là viết tắt của 5 chữ cái đầu trong 5 nguyên tắc thiết kế hướng đối tượng, giúp cho người phát triển viết ra những đoạn code dễ đọc, dễ hiểu, dễ bảo trì, được đưa ra bởi Bob Martin và Michael Feathers. Việc theo sát 5 nguyên tắc này nói thì để đáp ứng cả 5 nguyên tắc e là điều không đơn giản. 5 nguyên tắc đó bao gồm:

**Single responsibility priciple (SRP)** : Nguyên lý đầu tiên ứng với chữ S trong SOLID, có ý nghĩa là một class chỉ nên giữ một trách nhiệm duy nhất. Một class có quá nhiều chức năng sẽ trở nên cồng kềnh và trở nên khó đọc, khó bảo trì. Mà đối với ngành IT việc yêu cầu thay đổi, cần thêm sửa chức năng là rất bình thường, nên việc code trong sáng, dễ đọc dễ hiểu là rất cần thiết.

*Ví dụ:*

Thay vì viết một class ôm đồm tất cả các xử lý liên quan đến CSDL như này:

public class DBHelper {

public Connection openConnection() {};

public void saveUser(User user) {};

public List<Product> getProducts() {};

public void closeConnection() {};

}

Chúng ta nên tách thành các class xử lý công việc riêng kiểu như này:

public class DBConnection {

public Connection openConnection() {};

public void closeConnection() {};

}

public class UserRepository {

public void saveUser(User user) {};

}

public class ProductRepository {

public List<Product> getProducts() {};

}

**Open/Closed principle (OCP)** : Nguyên lý thứ 2 ứng với chữ O trong SOLID. Nội dung Có thể thoải mái mở rộng một class nhưng không được sửa đổi bên trong class đó.

*Ví dụ:* Trước đây, logic xử lý tính phí vận chuyển của một đơn hàng được đặt luôn bên trong class Order.

public class Order {

public long calculateShipping(ShippingMethod shippingMethod) {

if (shippingMethod == GROUND) {

/ / Calculate for ground shipping }

else if (shippingMethod == AIR) {

// Calculate for air shipping }

else {

// Default

}

}

}

Giả sử hệ thống cần bổ sung thêm một phương thức vận chuyển mới, chúng ta lại phải bổ sung một case nữa trong method calculateShipping. Điều này sẽ làm code trở nên rất khó quản lý.

Thay vào đó, chúng ta nên tách rời logic xử lý tính phí vận chuyển vào một interface Shipping chẳng hạn. Interface Shipping sẽ có nhiều implementation ứng với từng hình thức vận chuyển: GroundShipping, AirShipping, …

public interface Shipping {

long calculate();

}

public class GroundShipping implements Shipping {

@Override public long calculate() {

// Calculate for ground shipping

}

}

public class AirShipping implements Shipping {

@Override public long calculate() {

// Calculate for air shipping

}

}

public class Order {

private Shipping shipping;

public long calculateShipping(ShippingMethod shippingMethod) {

// Find relevant Shipping implementation then call calculate() method

}

}

**Liskov substitution principe (LSP)** : nguyên tắc thứ 3, ứng với chữ L trong SOLID. Nội dung nguyên tắc này được phát biểu như sau: Bất cứ instance nào của class cha cũng có thể được thay thế bởi instance của class con của nó mà không làm thay đổi tính đúng đắn của chương trình.

*Ví dụ:*

Chúng ta có interface Animal và 2 implementation Bird và Dog như sau:

public interface Animal {

void fly();

}

public class Bird implements Animal {

@Override public void fly() {

// Flying...

}

}

public class Dog implements Animal {

@Override public void fly() {

// Dog can't fly

throw new UnsupportedOperationException();

}

}

**Interface segregation principle (ISP)** : Nguyên tắc thứ 4 ứng với chữ I trong SOLID, nội dung nguyên tắc này như sau: Thay vì dùng 1 interface lớn, ta nên tách thành nhiều interface nhỏ, với nhiều mục đích cụ thể Client không nên phụ thuộc vào interface mà nó không sử dụng. Nguyên tắc này tương đối dể hiểu, thay vì gộp hết lại trong 1 interface lớn, ta có thể chia nhỏ thành nhiều interface nhỏ hơn gồm các method liên quan tới nhau, như vậy sẽ dễ quản lý hơn.

*Ví dụ :* Chúng ta có interface Animal như sau:

public interface Animal {

void eat();

void run();

void fly();

}

Chúng ta có 2 class Dog và Snake implement interface Animal. Nhưng thật vô lý, Dog thì làm sao có thể fly(), cũng như Snake không thể nào run() được? Thay vào đó, chúng ta nên tách thành 3 interface như thế này:

public interface Animal {

void eat();

}

public interface RunnableAnimal extends Animal {

void run();

}

public interface FlyableAnimal extends Animal {

void fly();

}

**Dependency inversion principle (DIP)** : Nguyên tắc thứ 5 ứng với chữ D trong SOLID, nội dung nguyên tắc này như sau: 1. Các module cấp cao không nên phụ thuộc vào các modules cấp thấp. Cả 2 nên phụ thuộc vào abstraction. 2. Abstraction không nên phụ thuộc vào chi tiết, mà ngược lại. Chúng ta hãy cùng quay trở lại với ví dụ đưa ra ở mục 2, nguyên tắc đóng/mở, tuy nhiên sẽ có một chút thay đổi:

Ví dụ, chúng ta có 2 module cấp thấp BackendDeveloper và FrontendDeveloper và 1 module cấp cao Project sử dụng 2 module trên:

public class BackendDeveloper {

private void codeJava() {};

}

public class FrontendDeveloper {

private void codeJS() {};

}

public class Project {

private BackendDeveloper backendDeveloper = new BackendDeveloper(); private FrontendDeveloper frontendDeveloper = new FrontendDeveloper();

public void build() {

backendDeveloper.codeJava();

frontendDeveloper.codeJS();

}

}

Giả sử nếu sau này, dự án thay đổi công nghệ. Các backend developer không code Java nữa mà chuyển sang code C#. Các frontend developer không code JS thuần nữa mà nâng lên các JS framework. Rõ ràng chúng ta không những phải sửa code ở các module cấp thấp (BackendDeveloper và FrontendDeveloper) mà còn phải sửa code ở cả module cấp cao (Project) đang sử dụng các module cấp thấp đó. Điều này cho thấy module cấp cao đang phải phụ thuộc vào các module cấp thấp.

### Giải pháp công nghệ

#### Ngôn ngữ lập trình và Framework hỗ trợ

**a) Ngôn ngữ lập trình** **Java:**

Java là một một ngôn ngữ lập trình hiện đại, bậc cao, hướng đối tượng, bảo mật và mạnh mẽ**.** và là một Platform.

**Platform:** Bất cứ môi trường phần cứng hoặc phần mềm nào mà trong đó có một chương trình chạy, thì được hiểu như là một Platform. Với môi trường runtime riêng cho mình (JRE) và API, Java được gọi là Platform.

Ngôn ngữ lập trình Java ban đầu được phát triển bởi Sun Microsystems do James Gosling khởi xướng và phát hành vào năm 1995. Phiên bản mới nhất của Java Standard Edition là Java SE 8. Với sự tiến bộ của Java và sự phổ biến rộng rãi của nó, nhiều cấu hình đã được xây dựng để phù hợp với nhiều loại nền tảng khác nhau. Ví dụ: J2EE cho các ứng dụng doanh nghiệp, J2ME cho các ứng dụng di động.

Các phiên bản J2 mới đã được đổi tên thành Java SE, Java EE và Java ME. Phương châm của java là **"Write Once, Run Anywhere"** - viết một lần chạy nhiều nơi, nghĩa là bạn chỉ cần viết một lần trên window chẳng hạn, sau đó vẫn chương trình đó bạn có thể chạy trên Linux, Android, các thiết bị J2ME...

Ngôn ngữ lập trình java có các tính năng sau:

Hướng đối tượng - Trong Java, mọi thứ đều là một Object. Java có thể dễ dàng mở rộng và bảo trì vì nó được xây dựng dựa trên mô hình Object.

Nền tảng độc lập - Không giống nhiều ngôn ngữ lập trình khác bao gồm cả C và C ++, khi Java được biên dịch, nó không được biên dịch thành ngôn ngữ máy nền tảng cụ thể, thay vào mã byte - nền tảng độc lập. Mã byte này được thông dịch bởi máy ảo (JVM) trên nền tảng nào đó mà nó đang chạy.

Đơn giản - Java được thiết kế để dễ học. Nếu bạn hiểu khái niệm cơ bản về OOP Java, sẽ rất dễ để trở thành master về java.

Bảo mật - Với tính năng an toàn của Java, nó cho phép phát triển các hệ thống không có virut, giả mạo. Các kỹ thuật xác thực dựa trên mã hoá khóa công khai.

Kiến trúc - trung lập - Trình biên dịch Java tạo ra định dạng tệp đối tượng kiến trúc trung lập, làm cho mã biên dịch được thực thi trên nhiều bộ vi xử lý, với sự hiện diện của hệ điều hành Java.

Portable - Là kiến trúc tập trung và không có khía cạnh thực hiện phụ thuộc của đặc tả này làm cho Java khả chuyển. Trình biên dịch trong Java được viết bằng ANSI C, đó là một tập con POSIX.

Mạnh mẽ - Java làm nỗ lực để loại trừ các tình huống dễ bị lỗi bằng cách kiểm tra lỗi tại thời gian biên dịch và kiểm tra lỗi tại runtime.

Đa luồng - Với tính năng đa luồng của Java có thể viết các chương trình có thể thực hiện nhiều tác vụ đồng thời. Tính năng thiết kế này cho phép các nhà phát triển xây dựng các ứng dụng tương tác có thể chạy trơn tru hơn.

Thông dịch - Mã byte Java được dịch trực tiếp tới các máy tính gốc và không được lưu trữ ở bất cứ đâu.

Hiệu năng cao - Với việc sử dụng trình biên dịch Just-In-Time, Java cho phép thực hiện hiệu năng cao.

Phân tán - Java được thiết kế cho môi trường phân tán của Internet.

Năng động - Java là năng động hơn C hoặc C++ vì nó được thiết kế để thích nghi với môi trường đang phát triển. Các chương trình Java có thể mang một lượng lớn thông tin tại runtime mà có thể được sử dụng để xác minh và giải quyết các truy cập vào các đối tượng tại runtime.

Java đã được sử dụng trong các lĩnh vực khác nhau. Ví dụ:

Desktop App như acrobat reader, media player, antivirus, ...

Web App như irctc.co.in, javatpoint.com, ...

Enterprise App như các ứng dụng về xử lý nghiệp vụ ngân hàng, ...

Thiết bị Mobile như các ứng dụng Android.

Hệ thống nhúng

Smart Card

Robot

Game App

Với tính chất đơn giản( cú pháp dựa trên C++, có bộ dọn rác tự động xóa các đối tượng không được tham chiếu,…), hỗ trợ lập trình hướng đối tượng, em đã quyết định chon Java là ngôn ngữ lập trình chính để phát triển phần mềm hỗ trợ quản lý DXDN cho doanh nghiệp.

**b) Framework hỗ trợ**:

**Spring Framework :** Spring là một Framework phát triển các ứng dụng Java được sử dụng bởi hàng triệu lập trình viên. Nó giúp tạo các ứng dụng có hiệu năng cao, dễ kiểm thử, sử dụng lại code…

Spring nhẹ và trong suốt (nhẹ: kích thước nhỏ, version cơ bản chỉ khoảng 2MB; trong suốt: hoạt động một cách trong suốt với lập trình viên)

Spring là một mã nguồn mở, được phát triển, chia sẻ và có cộng đồng người dùng rất lơn.

Spring Framework được xây dựng dựa trên 2 nguyên tắc design chính là: Dependency Injection và Aspect Oriented Programming.

Những tính năng core (cốt lõi) của Spring có thể được sử dụng để phát triển Java Desktop, ứng dụng mobile, Java Web. Mục tiêu chính của Spring là giúp phát triển các ứng dụng J2EE một cách dễ dàng hơn dựa trên mô hình sử dụng POJO (Plain Old Java Object). Với những ưu điểm kể trên, em Spring Framework sẽ là sự lựa chon của em cho việc phát triển đề tài xây dựng chương trình hỗ trợ đề xuất/phê duyệt tại một doanh nghiệp.

**Spring Security**: Spring Security là một dự án nổi bật trong hệ sinh thái của Spring Framework. Spring Security cung cấp các dịch vụ bảo mật toàn diện cho các ứng dụng doanh nghiệp có nền tảng Java EE.

Spring Security cung cấp 2 cơ chế cơ bản:

Authentication (xác thực): là tiến trình thiết lập một principal. Principal có thể hiểu là một người, hoặc một thiết bị, hoặc một hệ thống nào đó *có thể* thực hiện một hành động trong ứng dụng của bạn.

Authorization (phân quyền): là tiến trình quyết định xem một principal *có được phép* thực hiện một hành động trong ứng dụng của bạn hay không. Trước khi diễn tiến tới Authorization, principal cần phải được thiết lập bởi Authentication.

Nhằm mục đích khiến cho việc xử lý việc đăng nhập, đăng xuất và phân quyền cho các nhóm người dùng của chương trình trở nên dễ dàng hơn, em đã chọn áp dụng Spring Security vào đề tài xây dựng chương trình hỗ trợ quản lý đề xuất/phê duyệt tại một doanh nghiệp.

**Spring MVC**: là một Framewrok mã nguồn mở của Spring Framework cung cấp kiến trúc MVC (Model-View-Controller) và các component được sử dụng để phát triển các ứng dụng web một cách linh hoạt. Với sự hỗ trợ của Spring MVC, em có thể áp dụng mô hình MVC vào việc phát triển phần mềm hỗ trợ quản lý DXDN cho doanh nghiệp một cách dễ dàng hơn.

**Hibernate framework** : là một giải pháp ORM (Object Relational Mapping - giúp ánh xạ giữa các class java và các bảng dữ liệu vật lý) mã nguồn mở, gọn nhẹ. Mỗi table trong database là một object trong Hibernate. Do đó, bạn cần có một java bean cho mỗi table trong database. Các java bean này sẽ có các getters/setters và một số ngoại lệ theo quy ước của Hibernate. Mỗi java annotation ánh xạ (ví dụ @Table giúp ánh xạ từ một đối tượng object trong java tới một bảng dữ liệu dưới database, @Column giúp ánh xạ một thuộc tính của đối tượng object trong java tới một cột dữ liệu của bảng dưới database) có nhiệm vụ đặc tả mỗi quan hệ giữa các thuộc tính của đối tượng và các trường trong bảng CSDL (cột nào ứng với thuộc tính nào của class, kiểu giá trị là gì, quan hệ giữa các bảng chuyển thành quan hệ giữa các lớp ra sao…). Tất cả thông tin về kết nối (database kết nối là gì (MySQL, Oracle, SQL server…), username và password kết nối với database là gì, tên database được kết nối), driver kết nối… được load đầu tin khi chương trình khởi tạo từ file application.properties. Hibernate giúp đơn giản hoá sự phát triển của ứng dụng java để tương tác với cơ sở dữ liệu. Từ đó giúp em có thể dễ dàng tạo các bảng dữ liệu, các quan hệ và thao tác lên dữ liệu một cách dễ dàng hơn.

**Công nghệ thiết kế website**: HTML5, Bootstrap 3, JQuery, Thymeleaf. Đây là một số các công nghệ quen thuộc giúp em xây dựng nên giao diện của phần mềm hỗ trợ quản lý DXDN cho doanh nghiệp.

#### Hệ quản trị cơ sở dữ liệu

MySQL được phát triển bởi công ty công nghệ của Thụy Điển MySQL AB. Năm 2008 MySQL thuộc sở hữu của Sun Microsystems và năm 2010 nó chính thức thuộc quyền sở hữu của công ty phần mềm khổng lồ Oracle. MySQL tương thích với phần lớn các hệ điều hành phổ biến hiện nay như Microsoft Windows, macOS, Linux và Ubuntu.

SQL là ngôn ngữ phổ biến nhất để thêm, truy cập và quản lý nội dung trong cơ sở dữ liệu. Nó được chú ý nhất vì khả năng xử lý nhanh, độ tin cậy đã được chứng minh, dễ sử dụng và linh hoạt. MySQL là một phần thiết yếu của hầu hết mọi ứng dụng [PHP](https://viettuts.vn/php) mã nguồn mở. Các ví dụ điển hình cho các tập lệnh dựa trên PHP và MySQL là WordPress, Joomla, Magento và Drupal. MySQL đang trở nên phổ biến vì nhiều lý do tốt:

MySQL được phát hành theo giấy phép nguồn mở. Vì vậy, bạn không phải trả tiền để sử dụng nó.

MySQL là một chương trình rất mạnh theo đúng nghĩa của nó. Nó xử lý một tập hợp lớn các chức năng của các gói cơ sở dữ liệu mạnh mẽ và đắt tiền nhất.

MySQL sử dụng một dạng chuẩn của ngôn ngữ dữ liệu SQL nổi tiếng.

MySQL hoạt động trên nhiều hệ điều hành và với nhiều ngôn ngữ bao gồm PHP, PERL, C, C ++, JAVA, v.v.

MySQL hoạt động rất nhanh và hoạt động tốt ngay cả với các tập dữ liệu lớn.

MySQL rất thân thiện với PHP, ngôn ngữ được đánh giá cao nhất để phát triển web.

MySQL hỗ trợ cơ sở dữ liệu lớn, lên tới 50 triệu hàng hoặc nhiều hơn trong một bảng. Giới hạn kích thước tệp mặc định cho một bảng là 4GB, nhưng bạn có thể tăng mức này (nếu hệ điều hành của bạn có thể xử lý nó) đến giới hạn lý thuyết là 8 triệu terabyte (TB).

MySQL là tùy biến. Giấy phép GPL mã nguồn mở cho phép các lập trình viên sửa đổi phần mềm MySQL để phù hợp với môi trường cụ thể của riêng họ.

MySQL với ưu điểm miễn phí, khả năng chạy được trên cả window lẫn linux sẽ là sự lựa chọn về hệ quản trị cơ sở dữ liệu quan hệ cho phần mềm hỗ trợ quản lý DXDN cho doanh nghiệp.

# CHƯƠNG 2 PHÂN TÍCH & THIẾT KẾ CHƯƠNG TRÌNH

## Phân tích hệ thống

### 2.1.1 Phân tích hệ thống về chức năng

#### Xác định và mô tả các tác nhân

|  |  |  |
| --- | --- | --- |
| TÁC NHÂN | TIẾNG ANH | MÔ TẢ |
| Nhân viên | Employee | Nhân viên |
| Quản lý | Manager | Quản lý trực tiếp của nhân viên trong một phòng ban |
| Nhà điều hành | Operator | Người điều hành tất cả các hoạt động nhân sự |

Bảng 2. Các tác nhân và mô tả

#### Xác định và mô tả các use case

Đăng nhập: Nhân viên, người quản lý, nhà điều hành đăng nhập vào hệ thống quản lý DNDX cho doanh nghiệp.

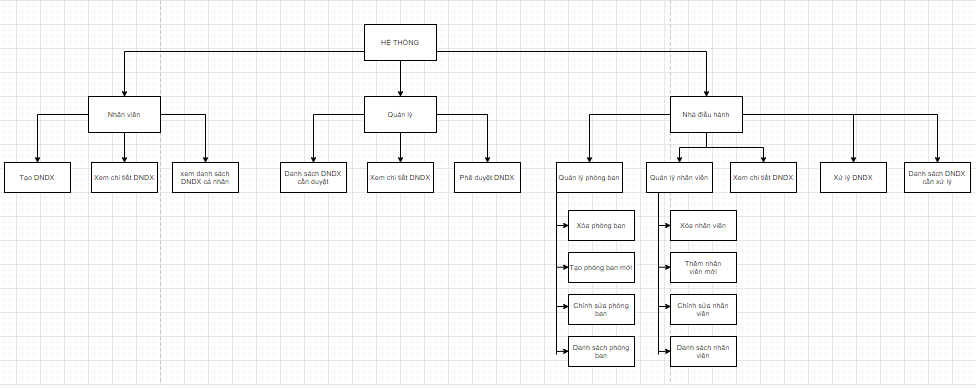
Quản lý DNDX cá nhân: Tạo đề nghị vật tư, tạo đề nghị xin nghỉ phép, xem chi tiết DNDX, hủy DNDX, xem danh sách DNDX cá nhân.

Quản lý phòng ban: Danh sách phòng ban, tạo phòng ban mới, chỉnh sửa phòng ban, xóa phòng ban.

Quản lý người dùng: Danh sách người dùng, thêm người dùng mới, chỉnh sửa thông tin người dùng, xóa người dùng.

Phê duyệt DNDX dành cho người quản lý: Danh sách DNDX cần duyệt, đã duyệt, đã từ chối và đã hoàn thành, chi tiết DNDX, phê duyệt DNDX.

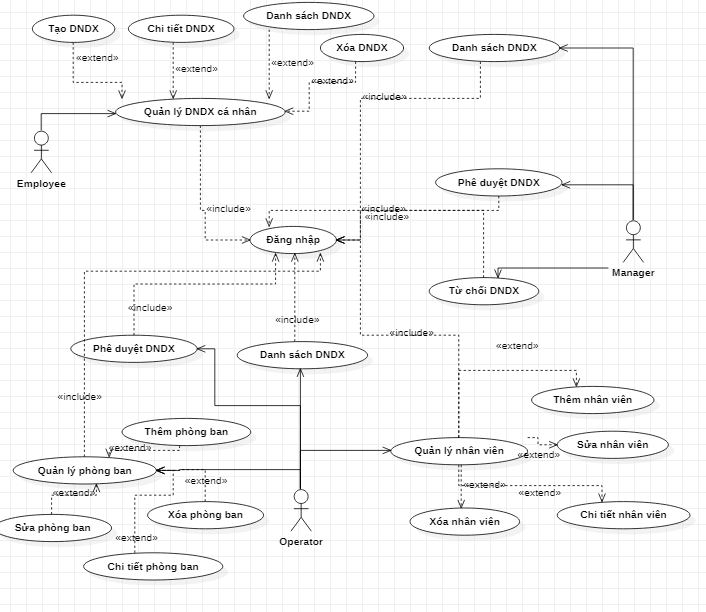
Xử lý DNDX dành cho nhà điều hành: Danh sách DNDX cần xử lý, đã từ chối và đã hoàn thành, chi tiết DNDX, xử lý DNDX.



Biểu đồ 2.1 Phân rã chức năng của hệ thống

#### Xây dựng biểu đồ use case

Chức năng hệ thống chia cho ba nhóm tác nhân : nhân viên, quản lý và nhà điều hành:



Biểu đồ 2.2 Use case toàn bộ hệ thống

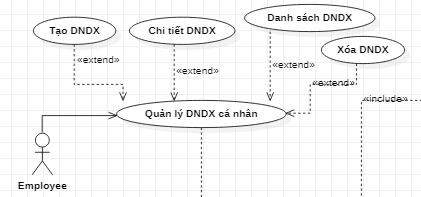
Use case của nhân viên: quản lý DNDX cá nhân

Tạo DNDX: nhân viên có thể tạo DNDX cá nhân, hiện tại hệ thống có hai loại DNDX cá nhân là đề nghị vật tư và đề nghị xin nghỉ phép

Chi tiết DNDX: nhân viên có thể xem chi tiết các thông tin của DNDX mà mình đã tạo bao gồm trạng thái ký duyệt của DNDX đó.

Danh sách DNDX: nhân viên có thể xem danh sách tất cả các DNDX của mình và có thể lọc theo các tiêu chí khác nhau.

Xóa DNDX: nhân viên có thể xóa(hủy) DNDX của mình nếu có nhầm lẫn gì đó và vẫn còn ở trạng thái dự thảo(quản lý chưa duyệt).



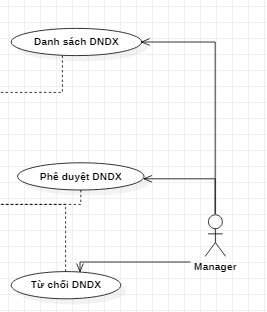
Biểu đồ .3 Phân rã chức năng của nhân viên

Use case của người quản lý: phê duyệt/từ chối DNDX

Danh sách DNDX: người quản lý có thể xem danh sách tất cả các DNDX của nhân viên trong phòng và có thể lọc theo các tiêu chí khác nhau.

Phê duyệt DNDX: người quản lý có thể phê duyệt DNDX của nhân viên.

Từ chối DNDX: người quản lý có thể từ chối DNDX của nhân viên.



Biểu đồ 2.4 Phân rã chức năng của quản lý

Use case của nhà điều hành: quản lý phòng ban, quản lý người dùng, xử lý/từ chối DNDX

Danh sách DNDX: nhà điều hành có thể xem danh sách tất cả các DNDX của nhân viên đã được quản lý phê duyệt và có thể lọc theo các tiêu chí khác nhau.

Xử lý DNDX: nhà điều hành có thể xử lý DNDX của nhân viên.

Từ chối DNDX: nhà điều hành có thể từ chối DNDX của nhân viên.

Danh sách phòng ban: nhà điều hành có thể xem danh sách tất cả các phòng ban và có thể lọc theo các tiêu chí khác nhau.

Thêm mới phòng ban: nhà điều hành có thể thêm mới phòng ban.

Chỉnh sửa phòng ban: nhà điều hành có thể chỉnh sửa phòng ban.

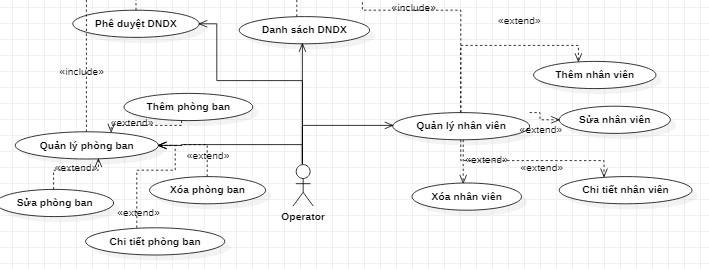
Xóa phòng ban: nhà điều hành có thể xóa phòng ban.

Danh sách nhân viên(người dùng): nhà điều hành có thể xem danh sách tất cả các nhân viên(người dùng) và có thể lọc theo các tiêu chí khác nhau.

Thêm mới nhân viên(người dùng): nhà điều hành có thể thêm mới nhân viên(người dùng).

Chỉnh sửa nhân viên(người dùng): nhà điều hành có thể chỉnh sửa nhân viên(người dùng).

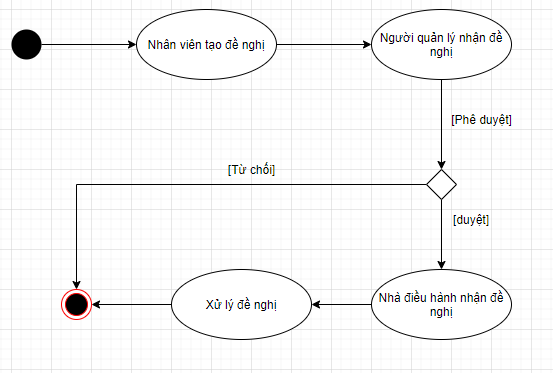
Xóa nhân viên(người dùng): nhà điều hành có thể xóa nhân viên(người dùng).



Biểu đồ 2.5 Phân rã chức năng của nhà điều hành

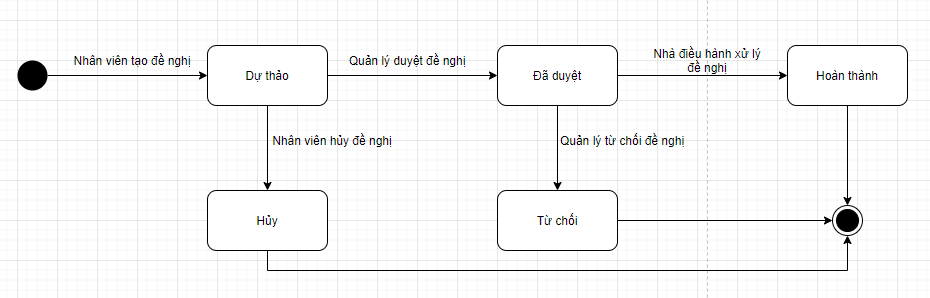
#### Phân tích luồng nghiệp vụ

Nhân viên đăng nhập vào hệ thống, tạo đề nghị cá nhân(có thể là đề nghỉ xin nghỉ phép hoặc đề nghị vật tư). Khi quản lý đăng nhập vào hệ thống thì có thể xem đề nghị cá nhân của nhân viên và chọn phê duyệt hoặc từ chối đề nghị đó. Nếu đề nghị được phê duyệt thì sẽ được tự động chuyển tới nhà điều hành và được xử lý.



Biểu đồ 2.6 Biểu đồ hoạt động của hệ thống

Khi nhân viên một đề nghị, đề nghị đó sẽ ở trạng thái dự thảo. Ở trạng thái này, nhân viên có thể hủy đề nghị. Đề nghị dự thảo sẽ được chuyển đến quản lý. Tại đây, nếu quản lý từ chối đề nghị, đề nghị đó sẽ chuyển sang trạng thái từ chối. Còn nếu quản lý chon duyệt đề nghị, đề nghị đó sẽ chuyển sang trạng thái đã duyệt. Khi đó, nhà điều hành sẽ nhận được thông tin đề nghị, chon xử lý và đề nghị sẽ được chuyển sang trạng thái hoàn thành.



Biểu đồ 2.7 Biểu đồ trạng thái đề nghị

### 2.1.2 Phân tích hệ thống về dữ liệu

#### 2.1.2.1 Mô hình thực thể liên hệ

**Bước 1**: Xác định các thực thể

Các thực thể và thuộc tính được xác định bao gồm :

Người dùng(ID người dùng, tên đăng nhập, họ và tên, địa chỉ, email, số điện thoại, mật khẩu, chữ kí)

Quyền(ID quyền, mã quyền, tên quyền)

Phòng ban(ID phòng ban, mã phòng ban, tên phòng ban)

Vật tư(ID vật tư, mã vật tư, tên vật tư, đơn vị tính)

Đề nghị(ID đề nghị, trạng thái, lý do, từ ngày, đến ngày, ngày tạo)

Đề nghị vật tư(Số lượng)

**Bước 2**: Xác định mối quan hệ giữa các thực thể

Quan hệ giữa **người dùng** và **quyền** là quan hệ n : n. Một người dùng có thể có nhiều quyền và một quyền có thể dành cho nhiều người dùng.

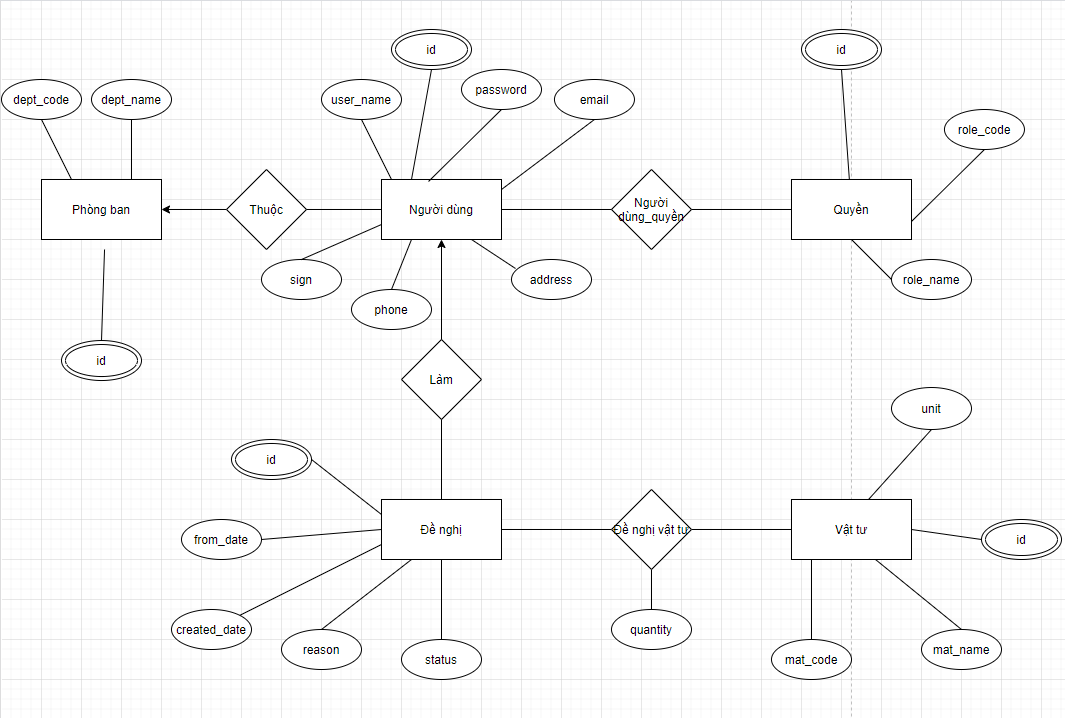
Quan hệ giữa **người dùng** và **đề nghị** là quan hệ 1 : n Một người dùng có nhiều DNDX và một DNDX của một người dùng.

Quan hệ giữa **phòng ban** và **người dùng** là quan hệ 1 : n. Một phòng ban có nhiều người và một người chỉ thuộc một phòng ban. Tuy nhiên với Operator sẽ không thuộc phòng ban nào.

Quan hệ giữa **đề nghị** và **đề nghị vật tư** là quan hệ 1 : n. Một đề nghị có thể có nhiều đề nghị vật tư và một đề nghị vật tư chỉ thuộc một đề nghị.

Quan hệ giữa **vật tư** và **đề nghị** **vật tư** là quan hệ 1 : n. Một vật tư có nhiều đề nghị vật tư và một đề nghị vật tư chỉ có một vật tư.

**Bước 3**: Mô hình thực thể liên hệ



Biểu đồ 2.8 Mô hình thực thể liên hệ

#### 2.1.2.2 Mô hình quan hệ

**Bước 1**: Chuyển các thực thể thành các lược đồ quan hệ

Các lược đồ quan hệ được xác định bao gồm :

1. Người dùng(id, user\_name, password, address, email, phone, sign)
2. Quyền(id, role\_code, role\_name)
3. Phòng ban(id, dept\_code, dep\_name)
4. Vật tư(id, mat\_code, mat\_name, unit)
5. Đề nghị(id, type, status, from\_date, to\_date, created\_date, reason)
6. Số lượng(quantity)

**Bước 2** : Chuyển mối liên hệ thành các lược đồ quan hệ

1. Thuộc(id, dept\_id)
2. Người dùng\_Quyền(user\_id, role\_id)
3. Đề nghị vật tư(mat\_id, req\_id, quantity)
4. Làm(id, user\_id)

**Bước 3:** Gộp các lược đồ quan hệ chung khóa

Gộp (1) và (7) ta được:

1. Người dùng(id, user\_name, password, address, email, phone, sign, dept\_id)

Gộp (5) và (10) ta được:

1. Đề nghị(id, type, status, from\_date, to\_date, created\_date, reason, user\_id)

**Kết quả:** sau khi gộp các lược đồ quan hệ chung khóa ta được các lược đồ quan hệ (11), (2), (3), (4), (12), (6), (8), (9)

**Bước 4:** Chuẩn hóa các lược đồ quan hệ

Tất cả các lược đồ trên đã ở dạng 3NF

## 2.2. Thiết kế hệ thống

### 2.2.1. Thiết kế hệ thống về dữ liệu

Từ mô hình thực thể hiện hệ sau khi thực hiện quá trình chuyển các mô hình thực thể liên hệ thành mô hình quan hệ, ta có được các lược đồ quan hệ để ở dạng chuẩn 3NF. Do đó, đảm bảo việc bảo toàn dữ liệu, không mất mát thông tin. Từ đó, ta có các bảng đây:

Role (Quyền hạn)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | id | bigint | 19 | pk |  |
| 2 | role\_name | varchar | 255 |  | Tên quyền |
| 3 | Role\_code | varchar | 255 |  | Mã quyền |

User (Người dùng/nhân viên)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | id | bigint | 19 | pk |  |
| 2 | user\_name | varchar | 255 |  | Tên tài khoản |
| 3 | password | varchar | 255 |  | Mật khẩu |
| 4 | email | varchar | 255 |  | Email |
| 5 | dept\_id | bigint | 19 | fk | Id phòng ban |
| 6 | address | varchar | 255 |  | Địa chỉ |
| 7 | sign | varchar | 255 |  | Đường dẫn tới file chữ kí |
| 8 | phone | varchar | 255 |  | Số điện thoại |

User\_Role (Gán quyền cho user)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | user\_id | bigint | 19 | fk | User |
| 2 | role\_id | bigint | 19 | fk | Quyền |

Request (DNDX)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | id | bigint | 19 | pk |  |
| 2 | status | int | 10 |  | Trạng thái đề nghị (1: dự thảo/2: đã duyệt/3: đã xử lý/4: từ chối/5: đã xử lý/6: đã hủy) |
| 3 | type | int | 10 |  | Loại đề nghị(1: đề nghị xin nghỉ phép/ 2: đề nghị vật tư) |
| 4 | create\_date | date | Datetime |  | Ngày tạo trên hệ thống |
| 5 | from\_date | date | Datetime |  | Ngày bắt đầu DNDX |
| 6 | to\_date | date | Datetime |  | Ngày kết thúc DNDX |
| 7 | reason | varchar | 255 |  | Lý do DNDX |
| 8 | user\_id | bigint | 19 | pk | Id người tạo đề nghị |

Department (Phòng ban)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | id | bigint | 19 | pk |  |
| 2 | dept\_name | varchar | 255 |  | Tên phòng ban |
| 3 | dept\_code | varchar | 255 |  | Mã phòng ban |

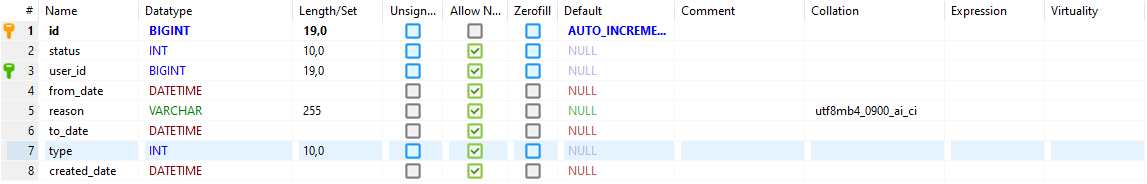
Material (Vật tư)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | id | bigint | 19 | pk | Mã vật tư |
| 2 | mat\_code | varchar | 255 |  | Mã code |
| 3 | mat\_name | varchar | 255 |  | Tên vật tư |
| 4 | unit | varchar | 255 |  | Đơn vị tính |

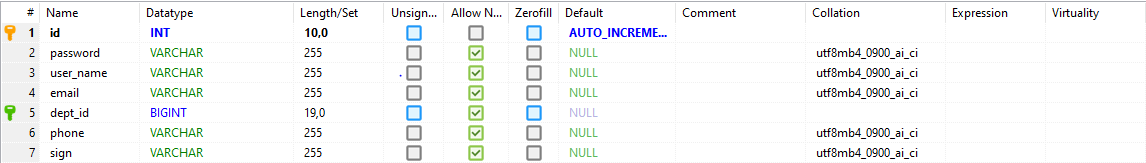
Request\_Material

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | TÊN TRƯỜNG | KIỂU DL | ĐỘ DÀI | KHÓA | MÔ TẢ |
| 1 | mat\_id | bigint | 19 | fk | Id vật tư |
| 2 | req\_id | bigint | 19 | fk | Id đề nghị |
| 3 | quantity | int | 10 |  | Số lượng |

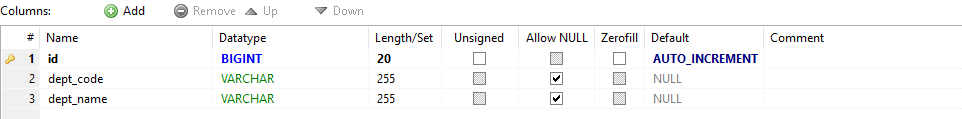
Từ đó, ta sẽ có thiết kế của các bảng dữ liệu sau :



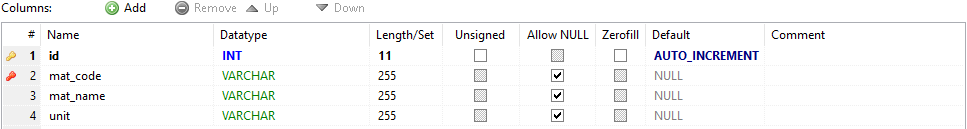
Hình 2.1 Bảng đề nghị



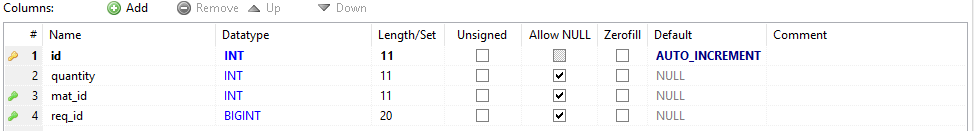
Hình 2.2 Bảng người dùng



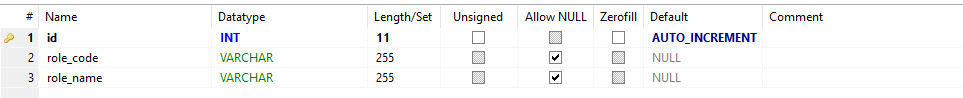
Hình 2.3 Bảng phòng ban



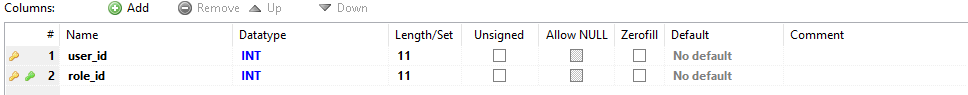
Hình 2.4 Bảng vật tư



Hình 2.5 Bảng đề nghị vật tư



Hình 2.6 Bảng quyền

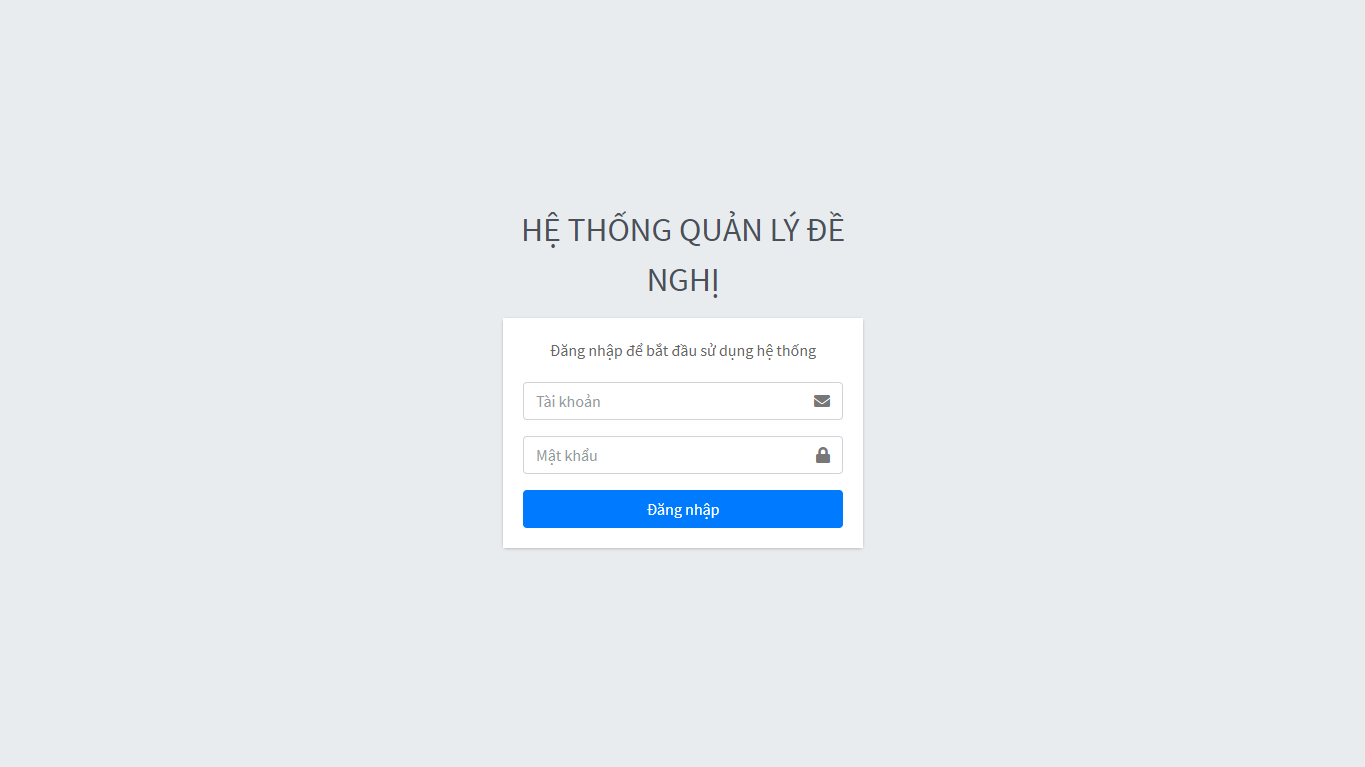


Hình 2.7 Bảng quyền người dùng

### 2.2.2 Thiết kế hệ thống về chức năng

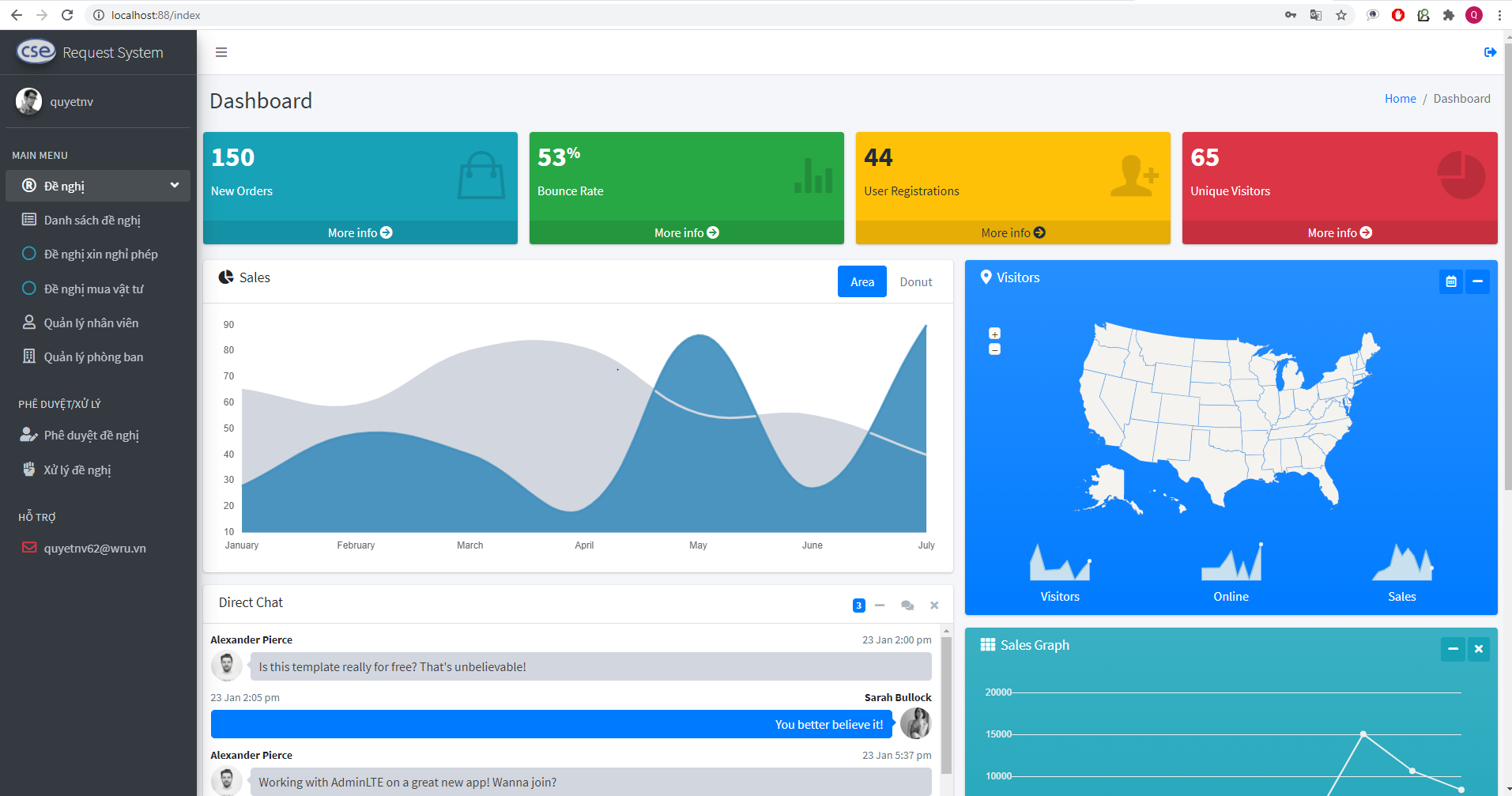
*2.2.2.1 Giao diện chính của hệ thống*

*Màn hình đăng nhập của người dùng*



Hình 2.8 Thiết kế giao diện màn hình đăng nhập

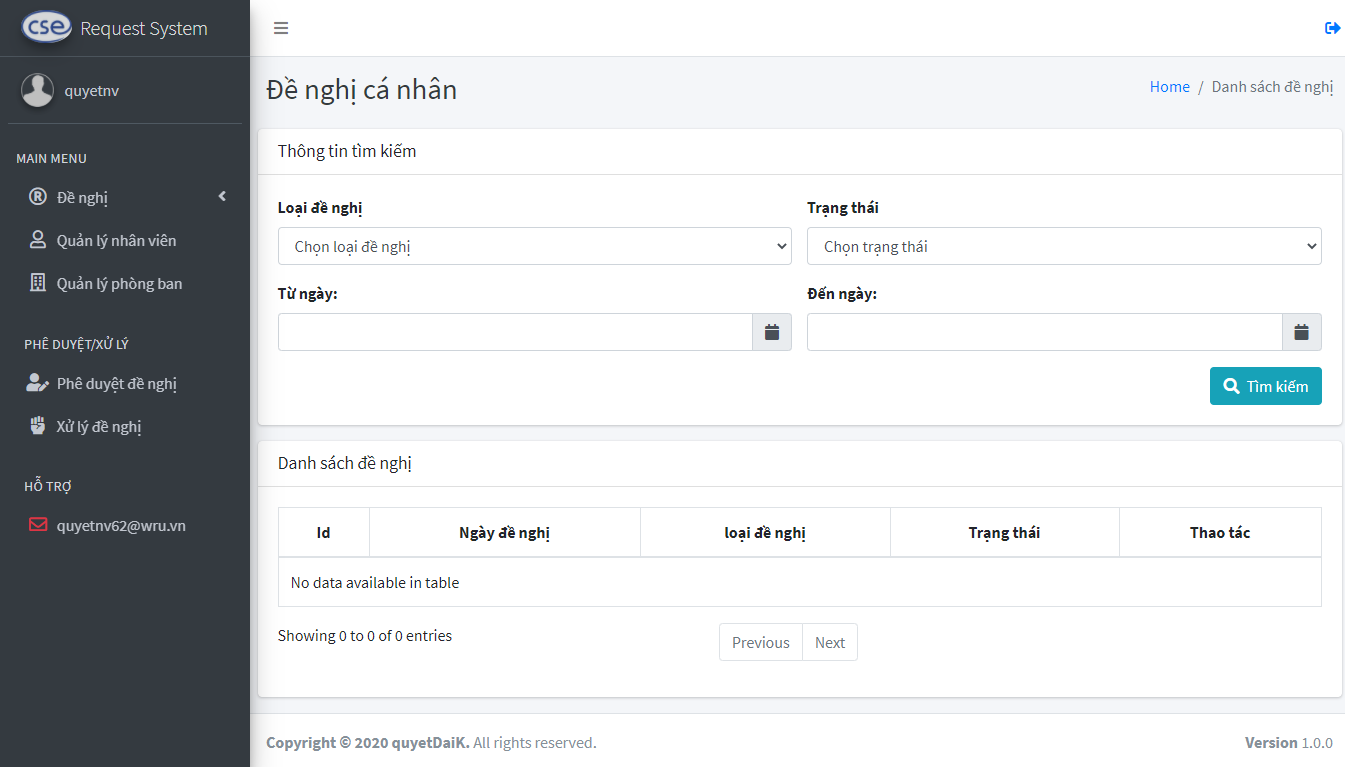
Màn hình khi người dùng đăng nhập thành công vào hệ thống



Hình 2.9 Thiết kế giao diện màn hình chính

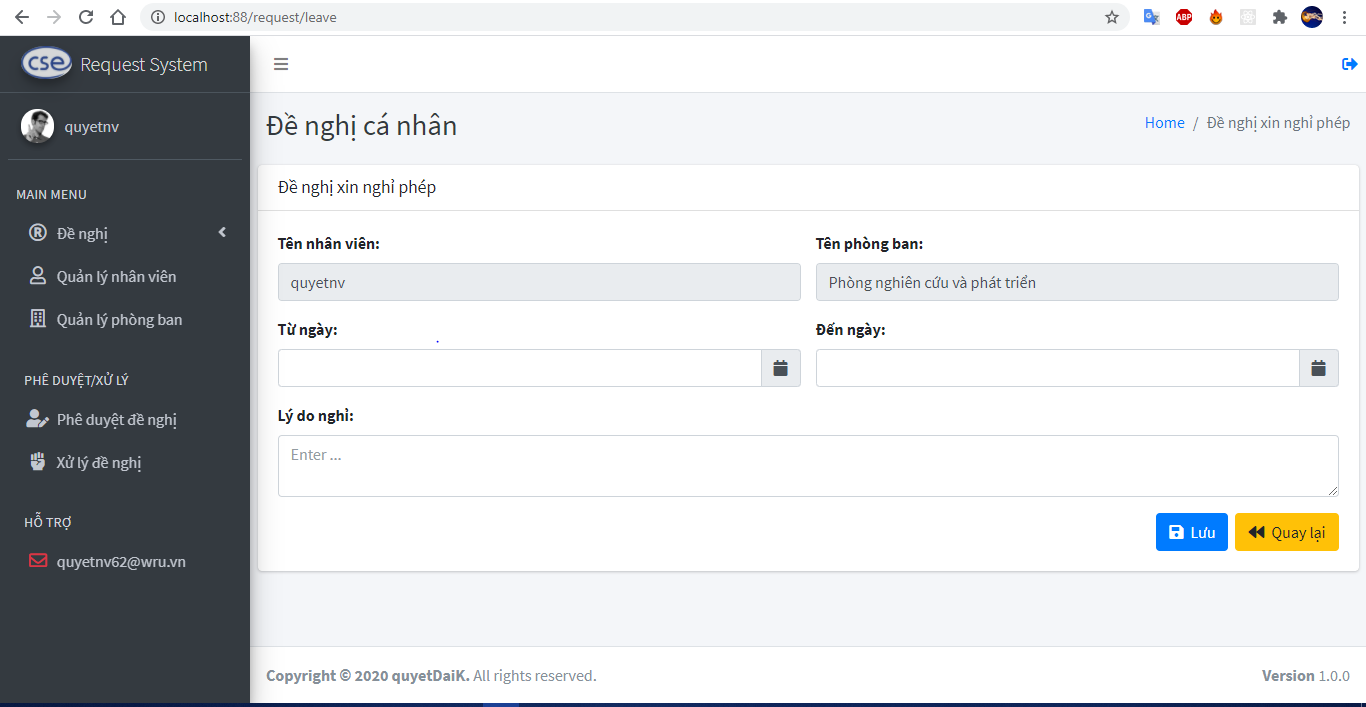
*2.2.2.2 Giao diện chức năng dành cho nhân viên*

*Thiết kế màn hình danh sách đề nghị cá nhân của từng nhân viên*



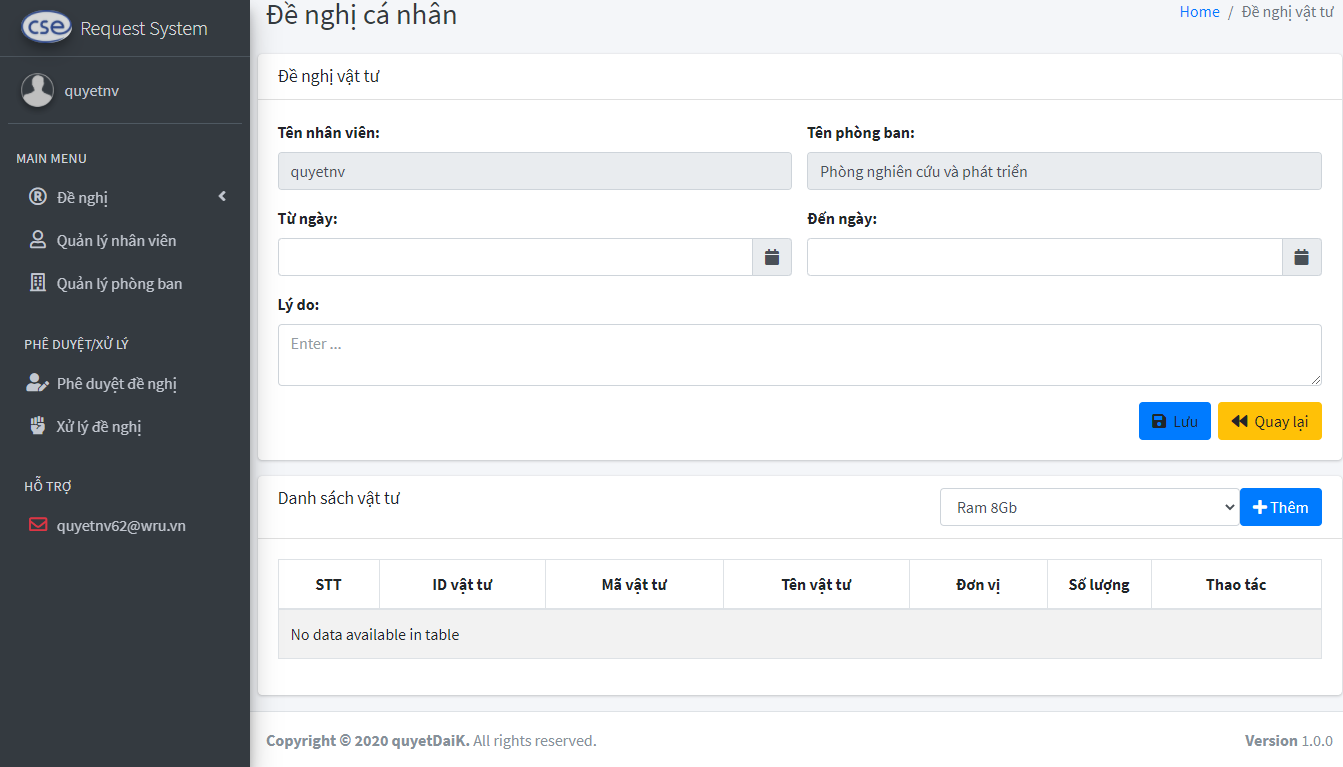
Hình 2.10 Thiết kế giao diện màn hình danh sách đề nghị cá nhân

Thiết kế màn hình khi nhân viên tạo một đề nghị xin nghỉ phép



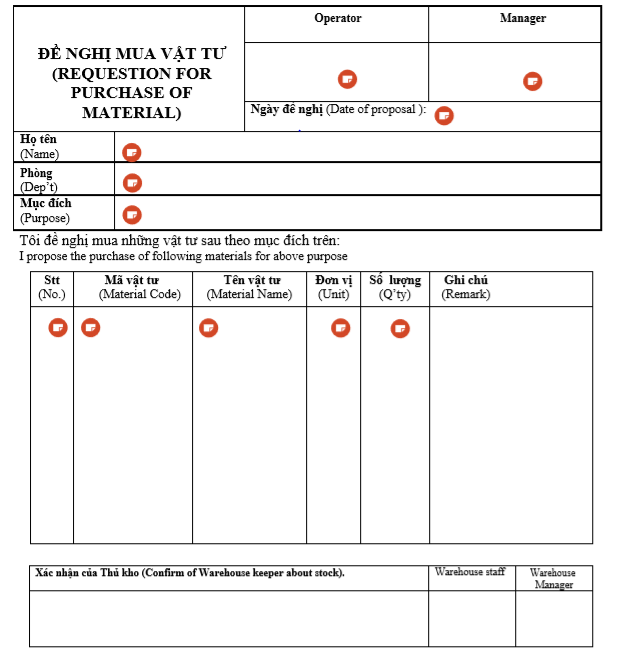
Hình 2.11 Thiết kế giao diện màn hình tạo đề nghị xin nghỉ phép

Thiết kế màn hình khi nhân viên tạo một đề nghị vật tư



Hình 2.12 Thiết kế giao diện màn hình tạo đề nghị vật tư

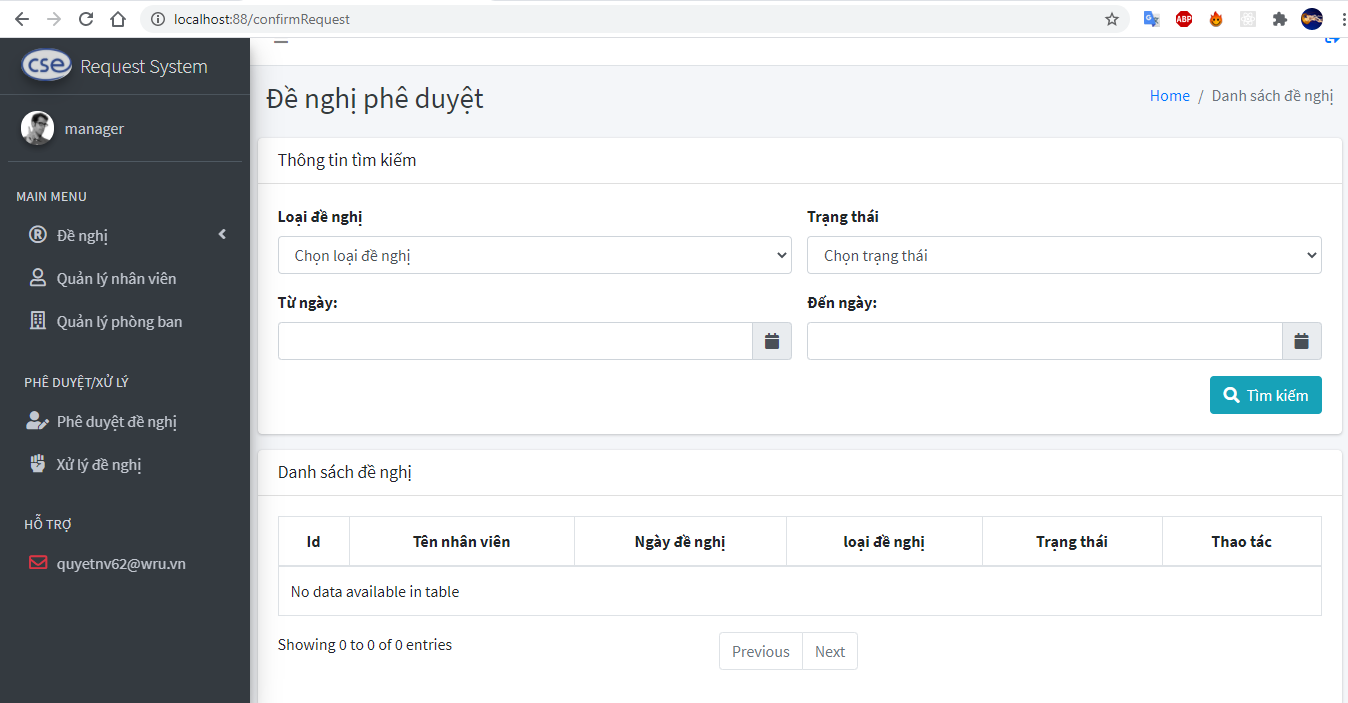
Mẫu template file pdf của đề nghị vật tư, các thông tin sẽ được fill vào đây trong suốt vòng đời của một đề nghị.



Hình 2.13 Thiết kế file pdf của đề nghị xin nghỉ phép

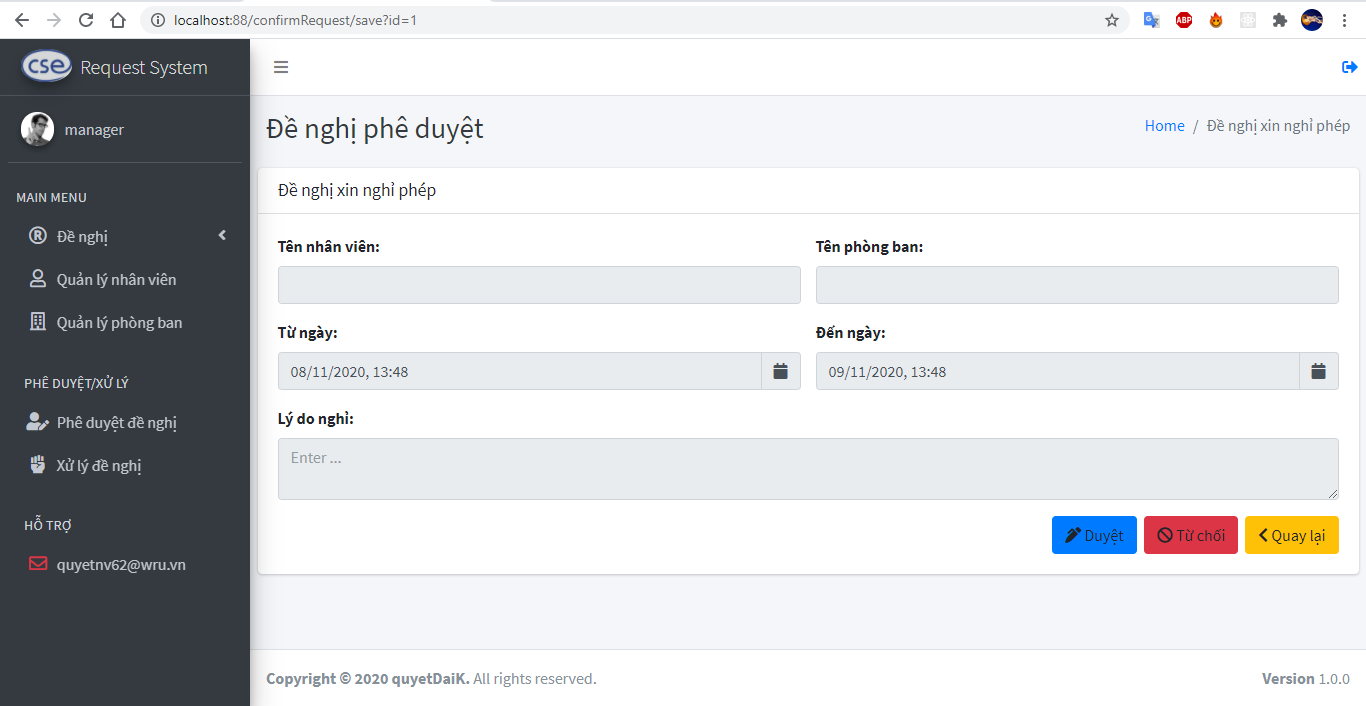
*2.2.2.3 Giao diện chức năng dành cho quản lý*

*Thiết kế giao diện danh sách đề nghị cần phê duyệt của người quản lý*

**

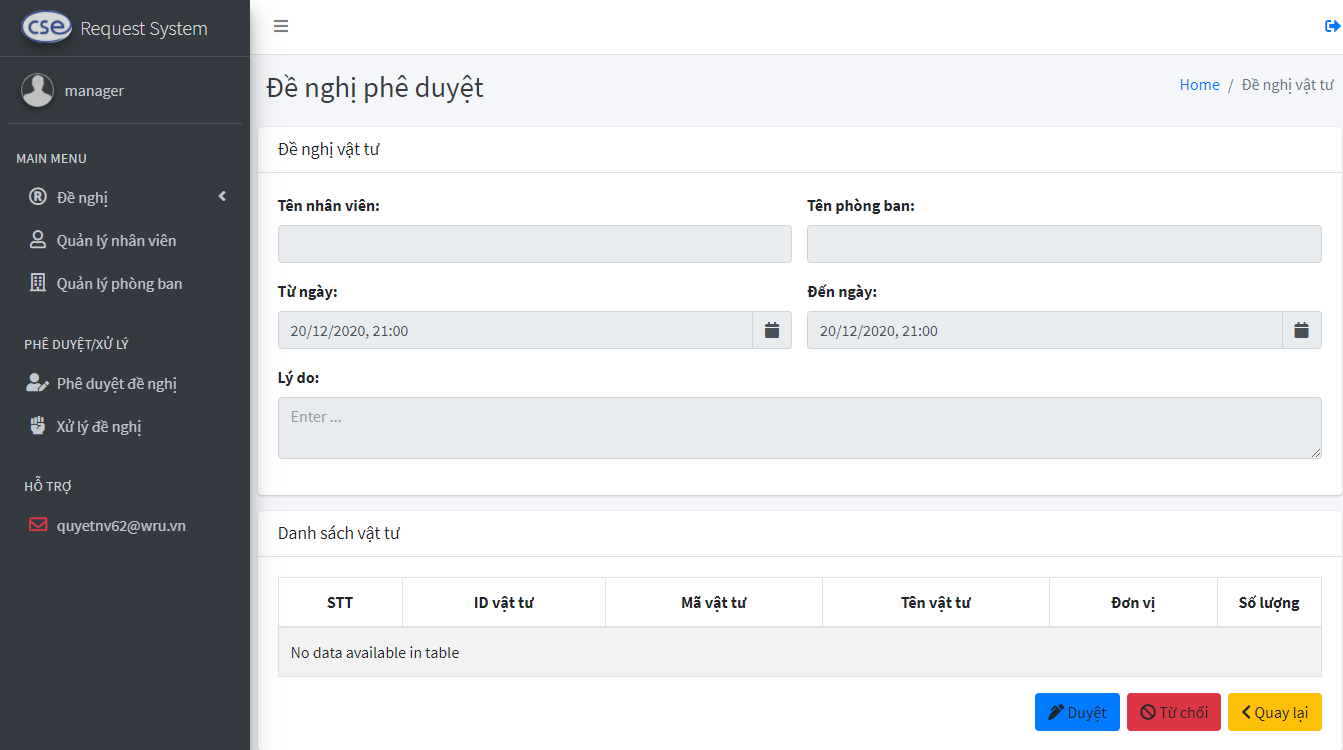
Hình 2.14 Thiết kế giao diện màn hình danh sách đề nghị phê duyệt

Thiết kế giao diện form phê duyệt đề nghị xin nghỉ phép của quản lý



Hình 2.15 Thiết kế giao diện màn hình duyệt đề nghị xin nghỉ phép

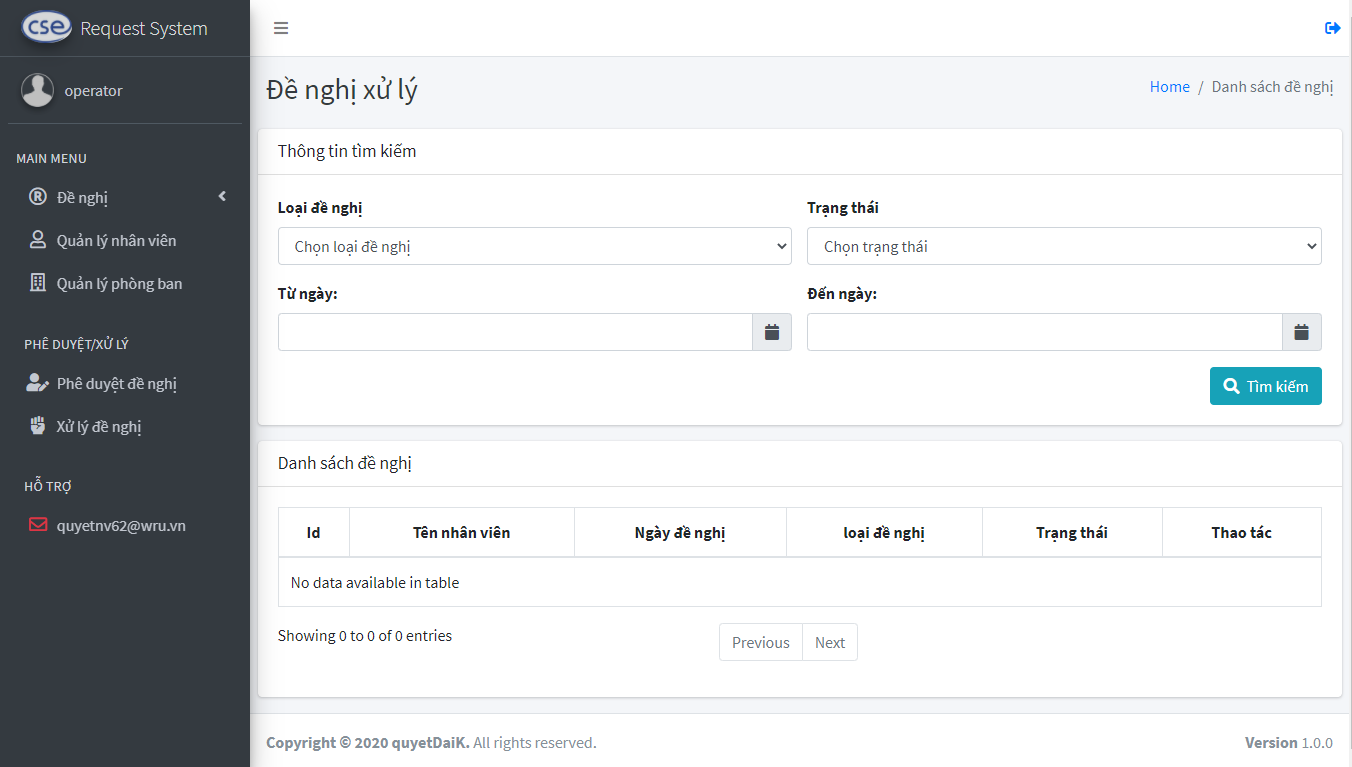
Thiết kế giao diện form phê duyệt đề nghị vật tư của quản lý



Hình 2.16 Thiết kế giao diện màn hình duyệt đề nghị vật tư

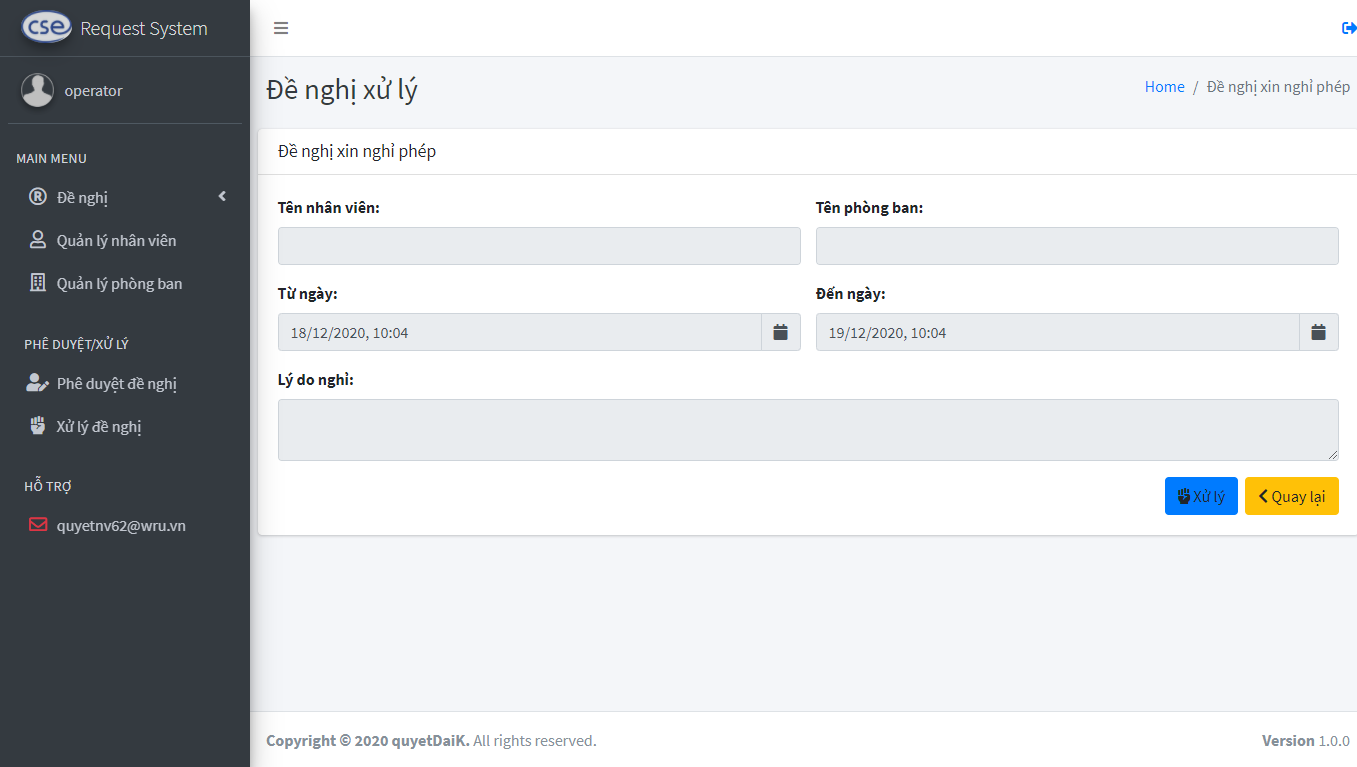
*2.2.2.3 Giao diện chức năng dành cho nhà điều hành*

Thiết kế giao diện danh sách đề nghị cần xử lý dành cho nhà điều hành



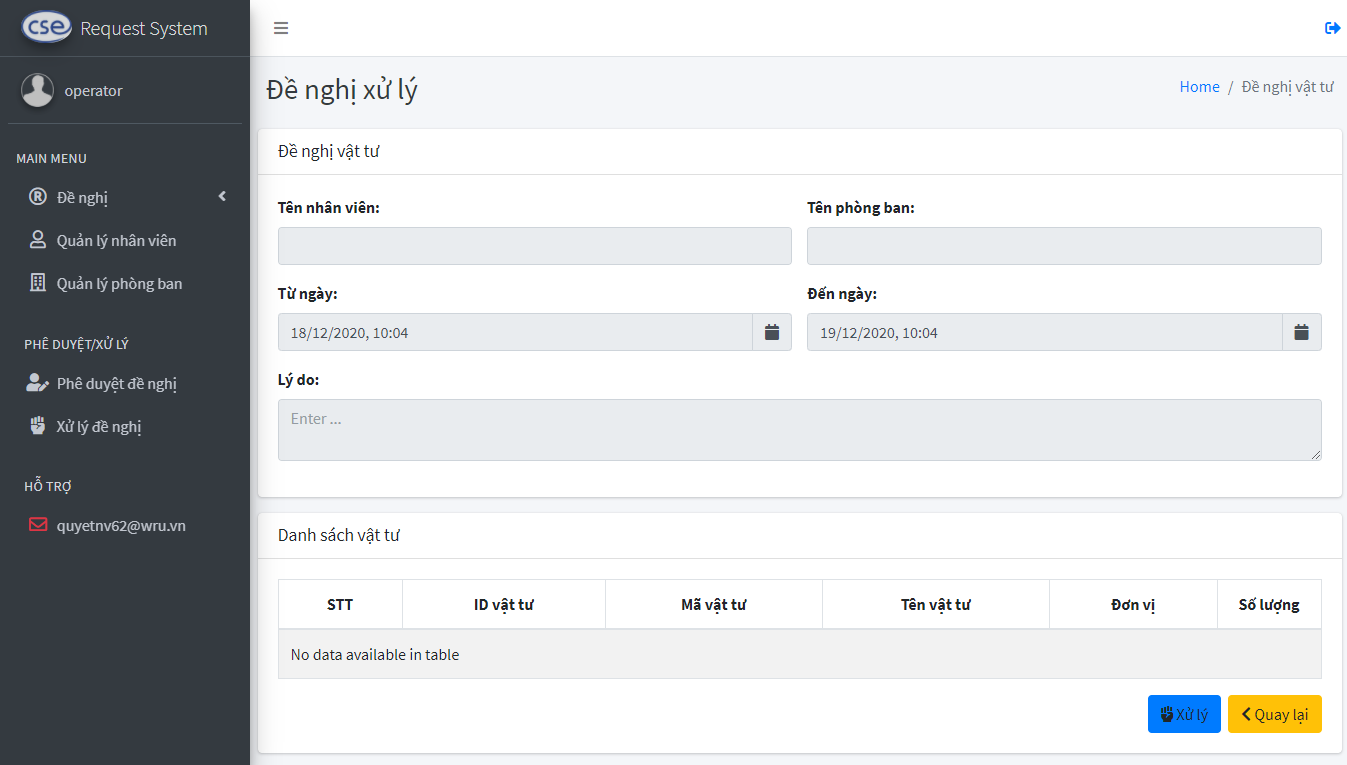
Hình 2.17 Thiết kế giao diện màn hình danh sách đề nghị xử lý

Thiết kế giao diện màn hình xử lý đề nghị xin nghỉ phép dành cho nhà điều hành



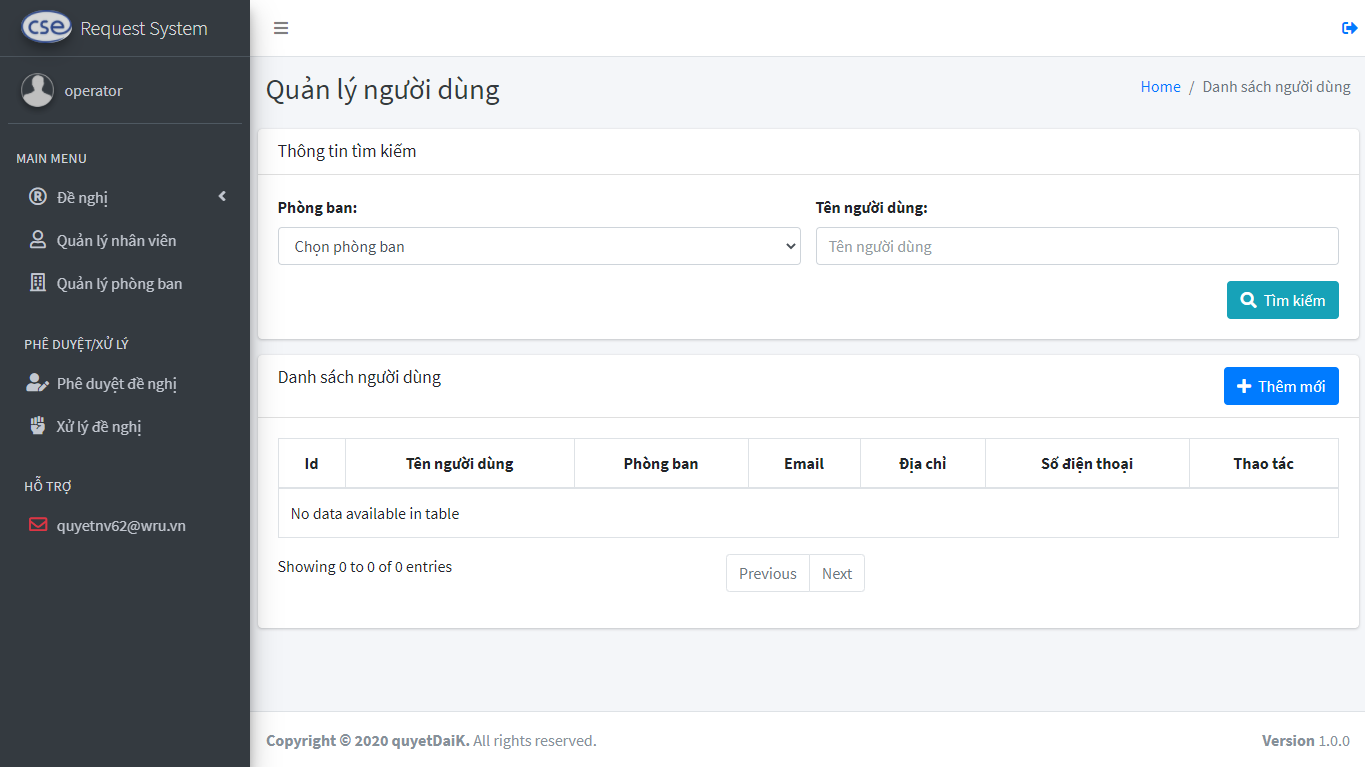
Hình 2.18 Thiết kế giao diện màn hình xử lý đề nghị xin nghỉ phép

Thiết kế giao diện màn hình xử lý đề nghị vật tư dành cho nhà điều hành



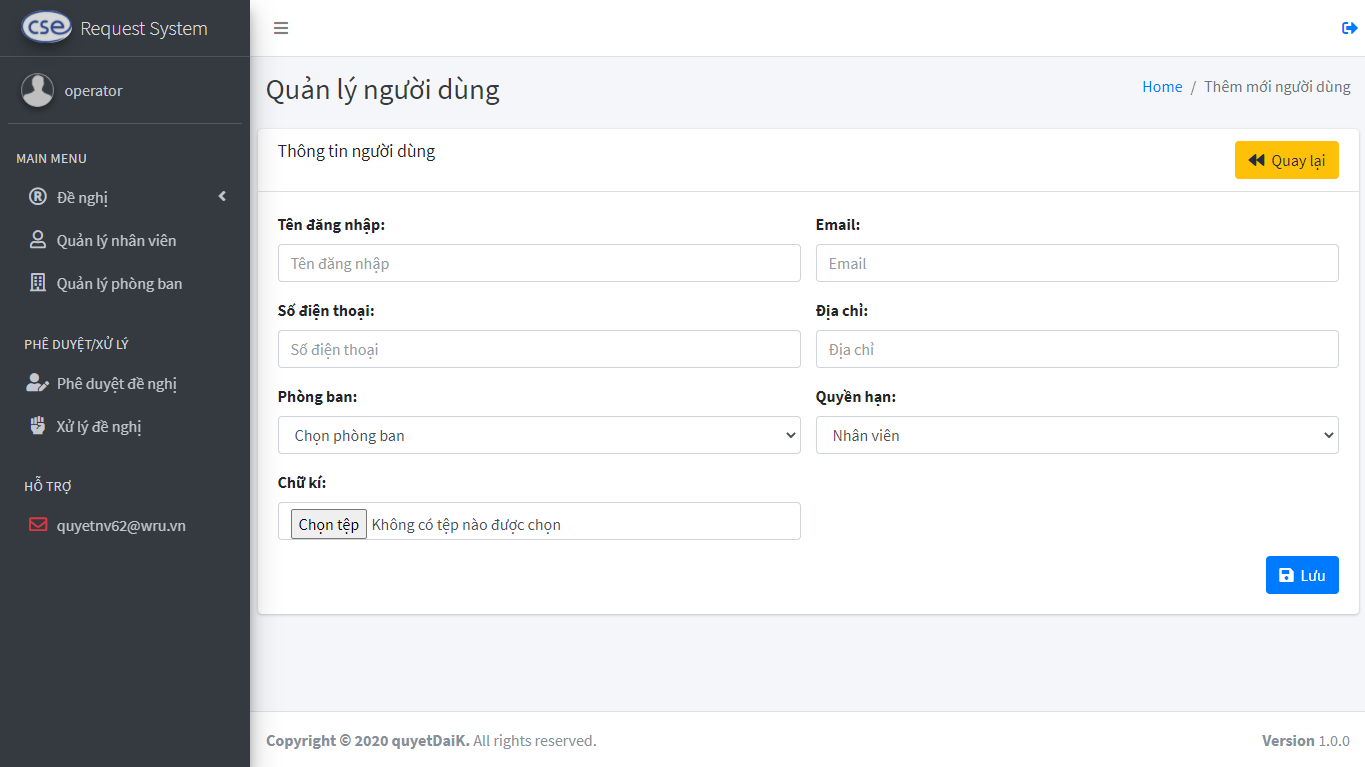
Hình 2.19 Thiết kế giao diện màn hình xử lý đề nghị vật tư

Thiết kế giao diện danh sách người dùng dành cho nhà điều hành



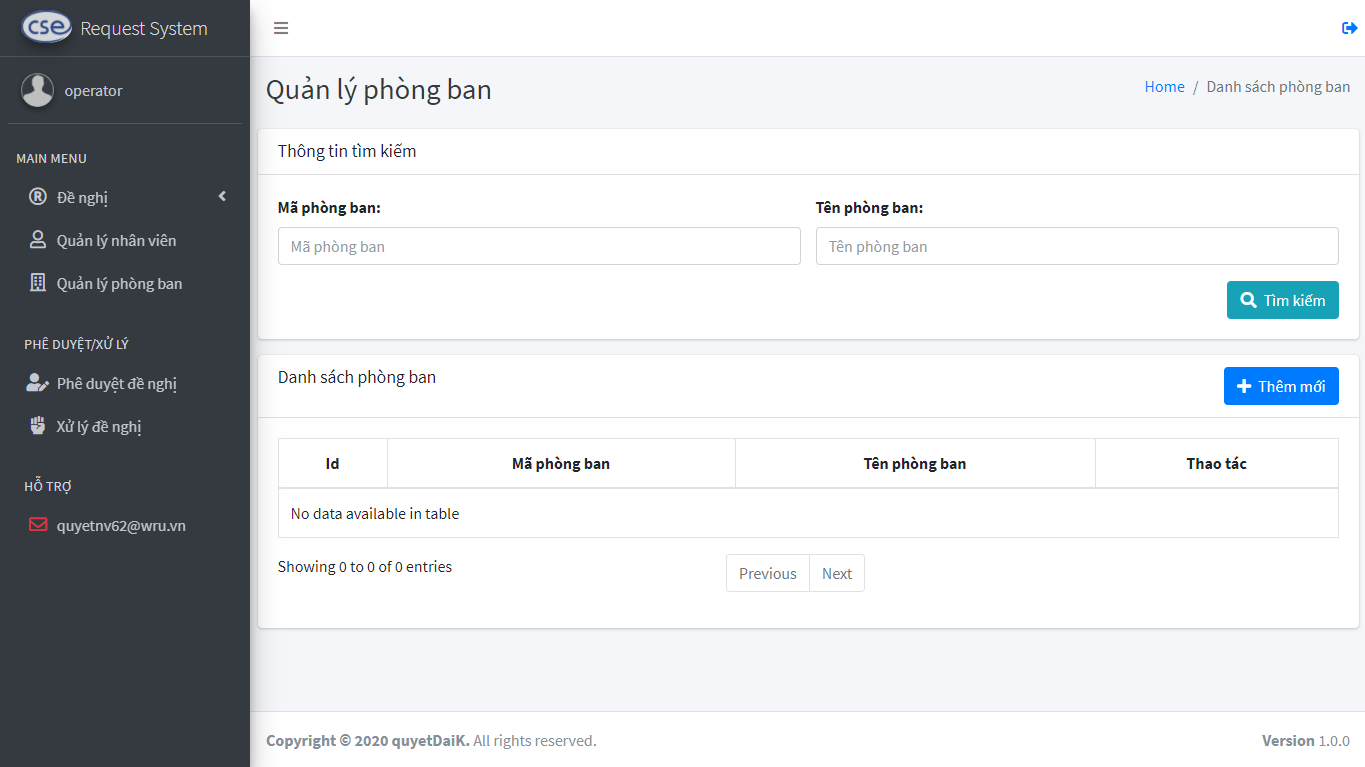
Hình 2.20 Thiết kế giao diện màn hình danh sách người dùng

Thiết kế giao diện thêm mới người dùng dành cho nhà điều hành



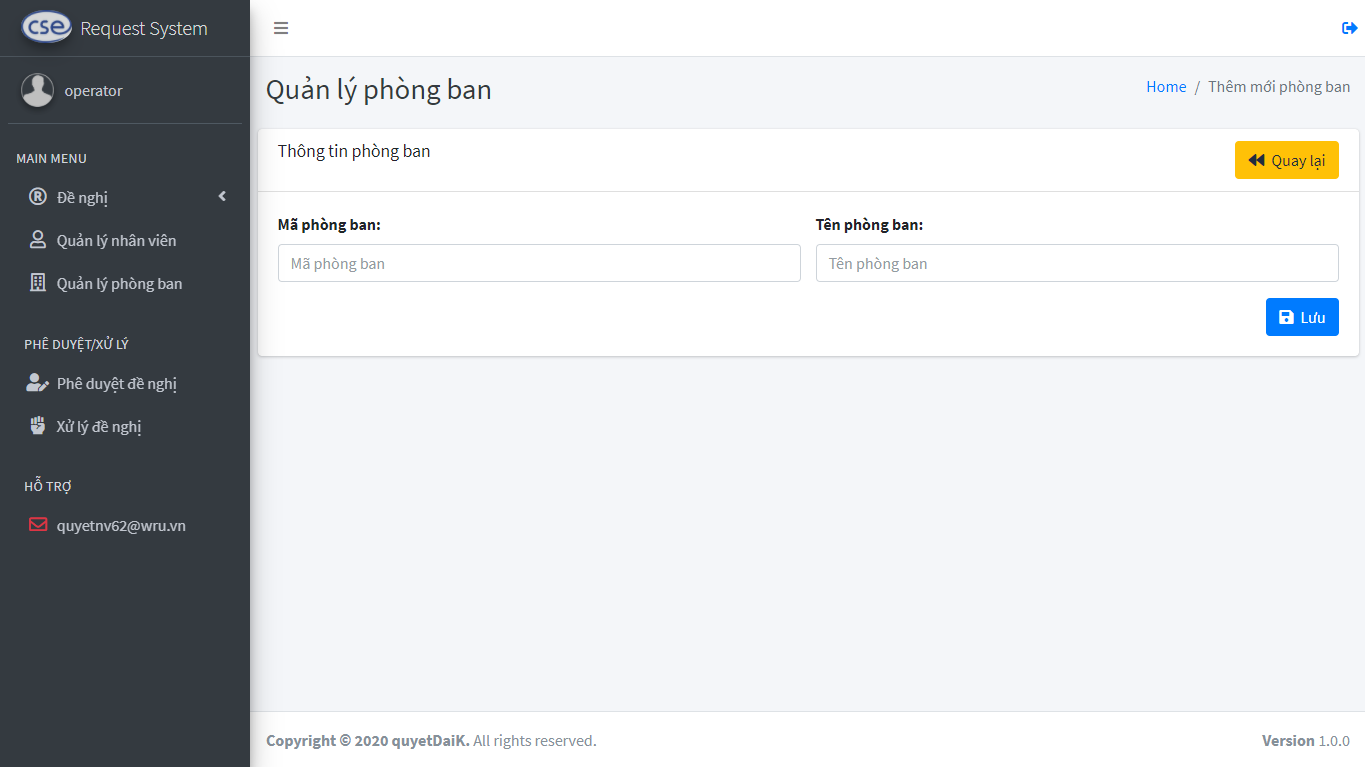
Hình 2.21 Thiết kế giao diện màn hình thêm mới người dùng

Thiết kế giao diện danh sách phòng ban dành cho nhà điều hành



Hình 2.22 Thiết kế giao diện màn hình danh sách phòng ban

Thiết kế giao diện danh sách phòng ban dành cho nhà điều hành



Hình 2.23 Thiết kế giao diện màn hình thêm mới phòng ban

# CHƯƠNG 3 THỬ NGHIỆM CHƯƠNG TRÌNH

## Cấu hình hệ thống

### Yêu cầu phần cứng

Hiện tại phần mềm đang được chạy bình thường và mượt mà trên máy tính cá nhân với cấu hình :

* Chip i5 2.6 Ghz (2 cores, 4 threads)
* SSD 128 Gb

### Môi trường chạy chương trình

Môi trường chạy chương trình bao gồm:

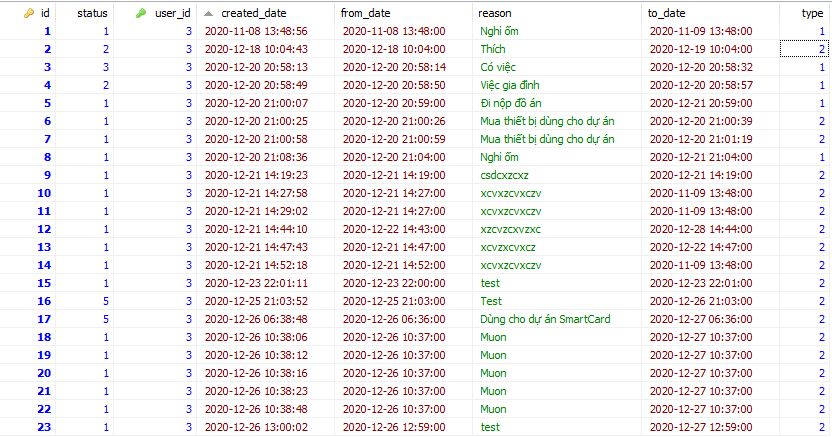
Java 8

Mysql 5.7

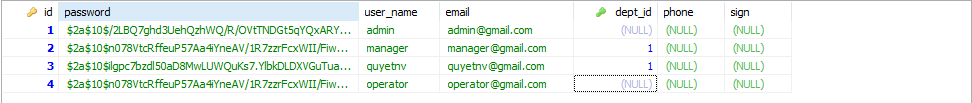
Window 10

## Các kết quả

### Các kết quả về dữ liệu



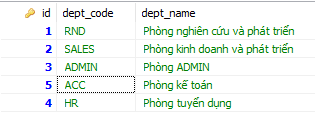
Hình 3.1 Dữ liệu thử nghiệm bảng đề nghị



Hình 3.2 Dữ liệu thử nghiệm bảng người dùng



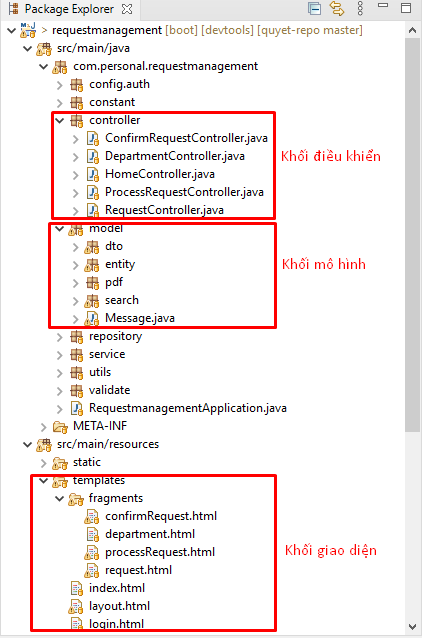
Hình 3.3 Dữ liệu thử nghiệm bảng vật tư



Hình 3.4 Dữ liệu thử nghiệm bảng phòng ban

### Các kết quả về chức năng

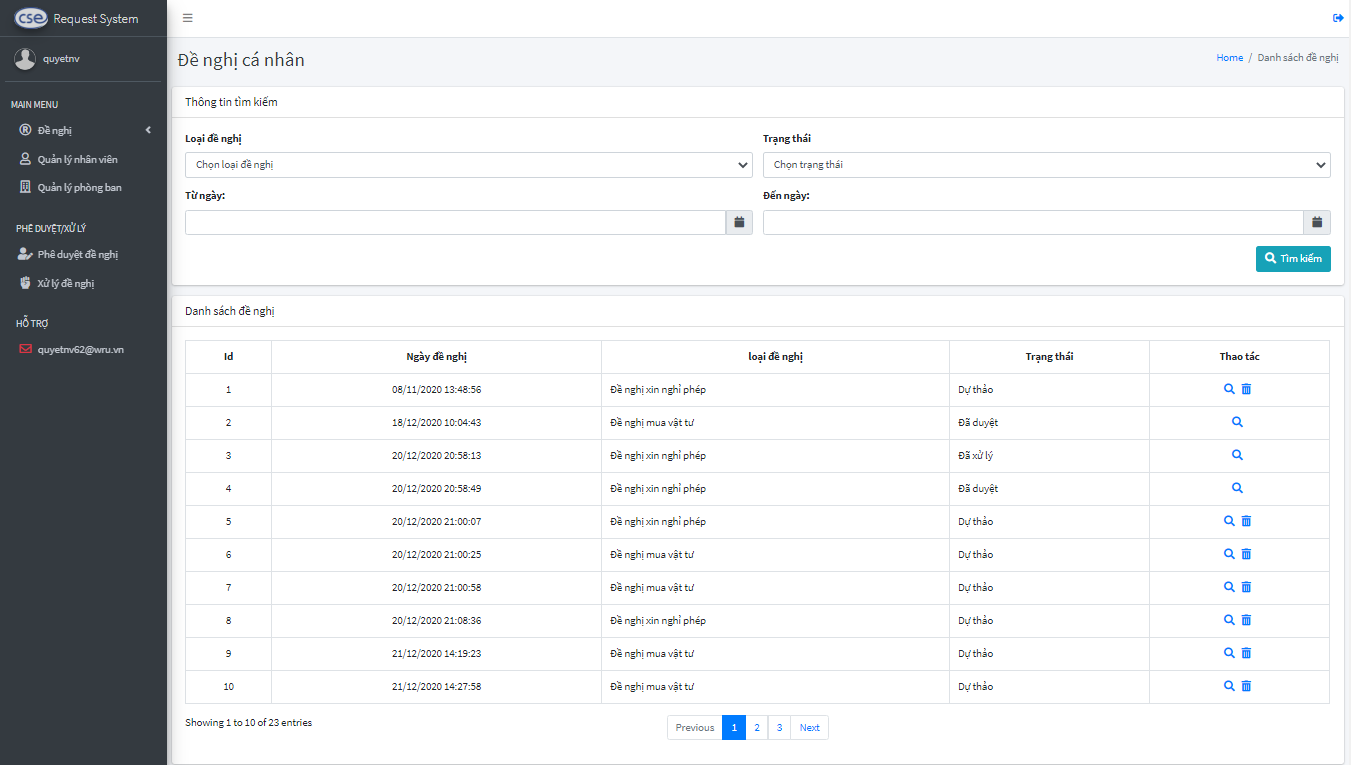
Phần mềm đã được tổ chức và phát triển dựa trên mô hình MVC: các file thuộc khối giao diệncó các định dạng là là \*.html, \*.css, \*.js được đặt trong thư mục src/main/resources. Các file java được đặt trong package **controller**, **service** sẽ đóng vai trò thực hiện nhiệm vụ của lớp khối điều khiển trong mô hình MVC. Các file java được đặt trong package model, repository thực hiện nhiệm vụ của khối mô hình trong mô hình MVC. Các package còn lại trong dự án chứa những file liên quan đến cấu hình, hoặc hỗ trợ nhiệm vụ thêm cho việc thực hiện mô hình MVC.



Hình 3.5 Cấu trúc của mã nguồn code

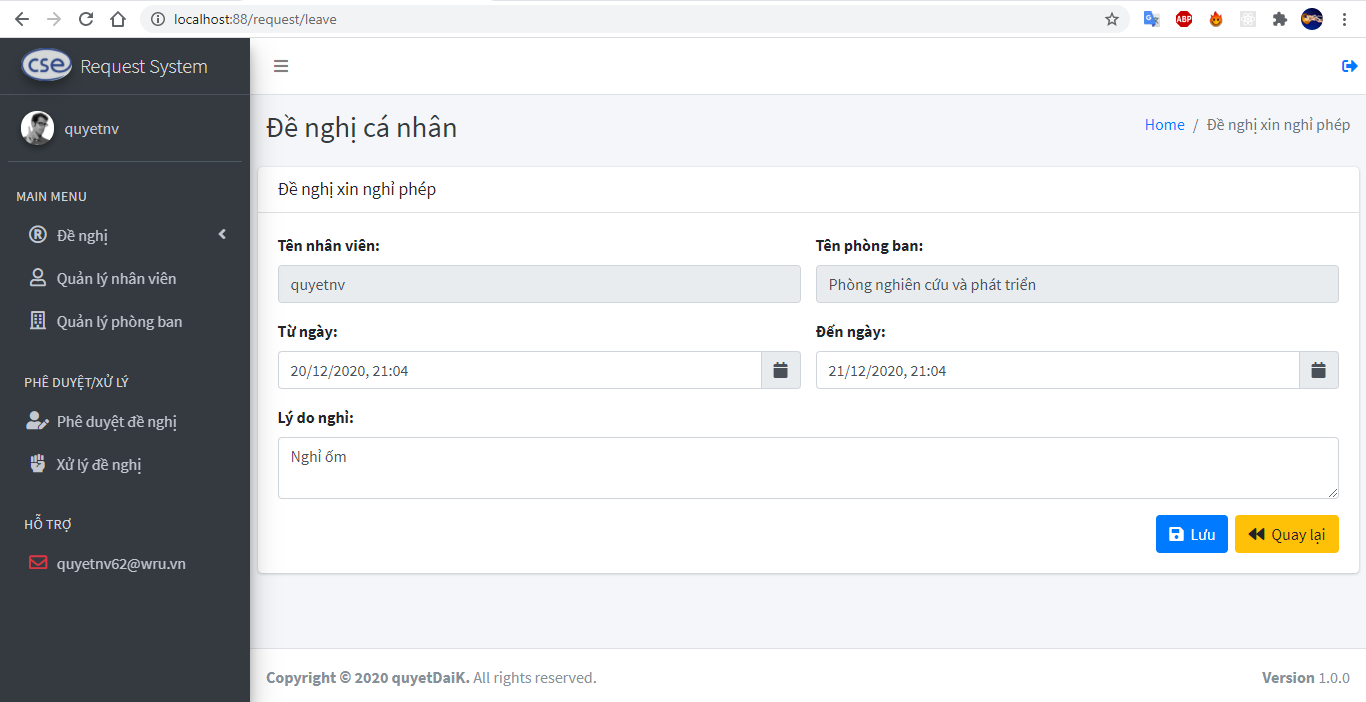
**Chức năng dành cho nhân viên:**

Màn hình danh sách đề nghị cá nhân của nhân viên: Nhân viên có thể theo dõi danh sách tất cả các đề nghị, thông tin và trạng thái của chúng ở màn hình này.



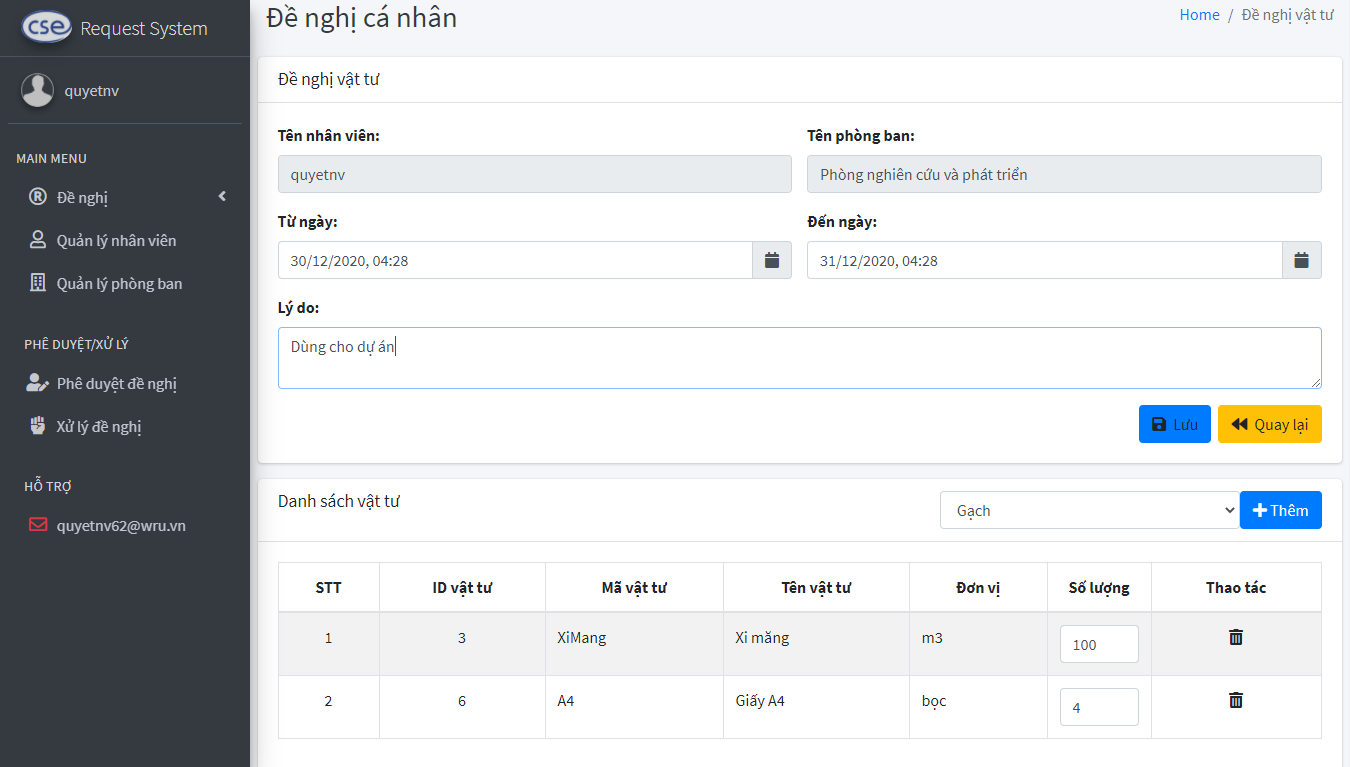
Hình 3.6 Màn hình danh sách đề nghị cá nhân

Màn hình tạo đề nghị xin nghỉ phép dành cho nhân viên, nhân viên có thể gửi yêu cầu đề nghị cho quản lý của phòng mình qua màn hình này.



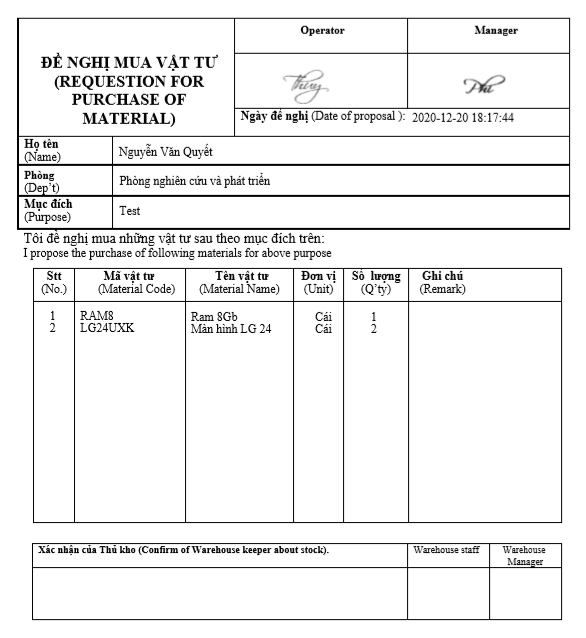
Hình 3.7 Màn hình tạo đề nghị xin nghỉ phép

Màn hình tạo đề nghị vật tư dành cho nhân viên, nhân viên có thể gửi yêu cầu đề nghị cho quản lý của phòng mình qua màn hình này. Khi hoàn thành nhập thì thông tin về đề nghị sẽ tự động fill vào template đề nghị vật tư.



Hình 3.8 Màn hình tạo đề nghị vật tư

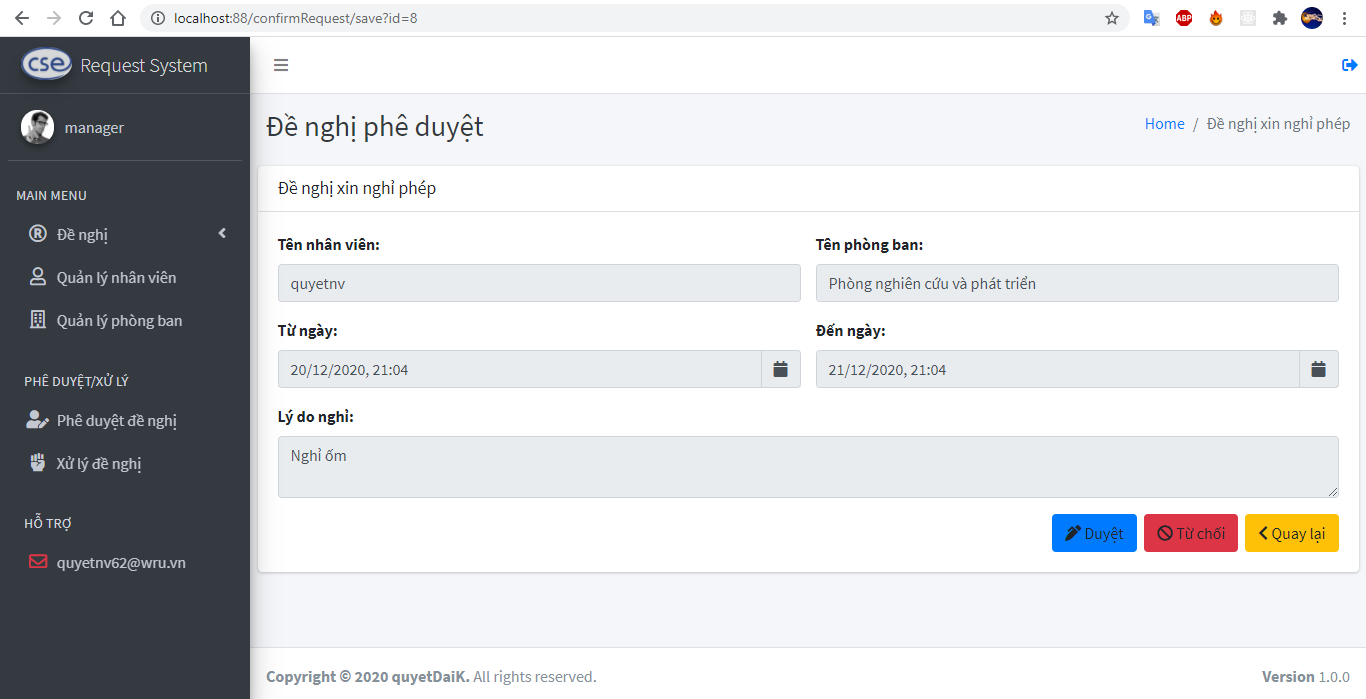
File pdf nhân viên có thể download về từ hệ thống sau khi đề nghị vật tư đã được phê duyệt



Hình 3.9 File pdf nhân viên nhận được sau khi cấp trên đã duyệt đề nghị vật tư

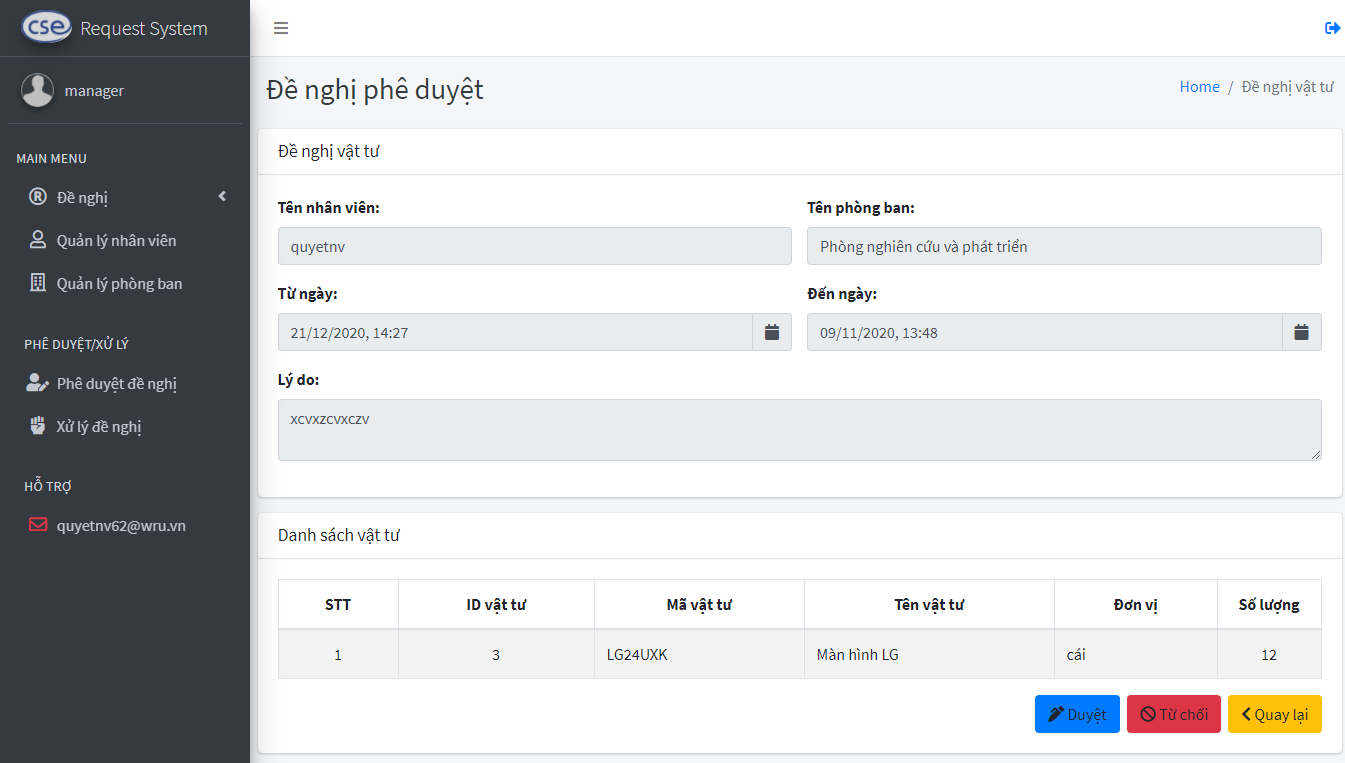
**Chức năng dành cho quản lý:**

Màn hình phê duyệt đề nghị vật tư dành cho nhà quản lý, nhà quản lý có thể xem thông tin đề nghị, phê duyệt hoặc từ chối tại màn hình này.



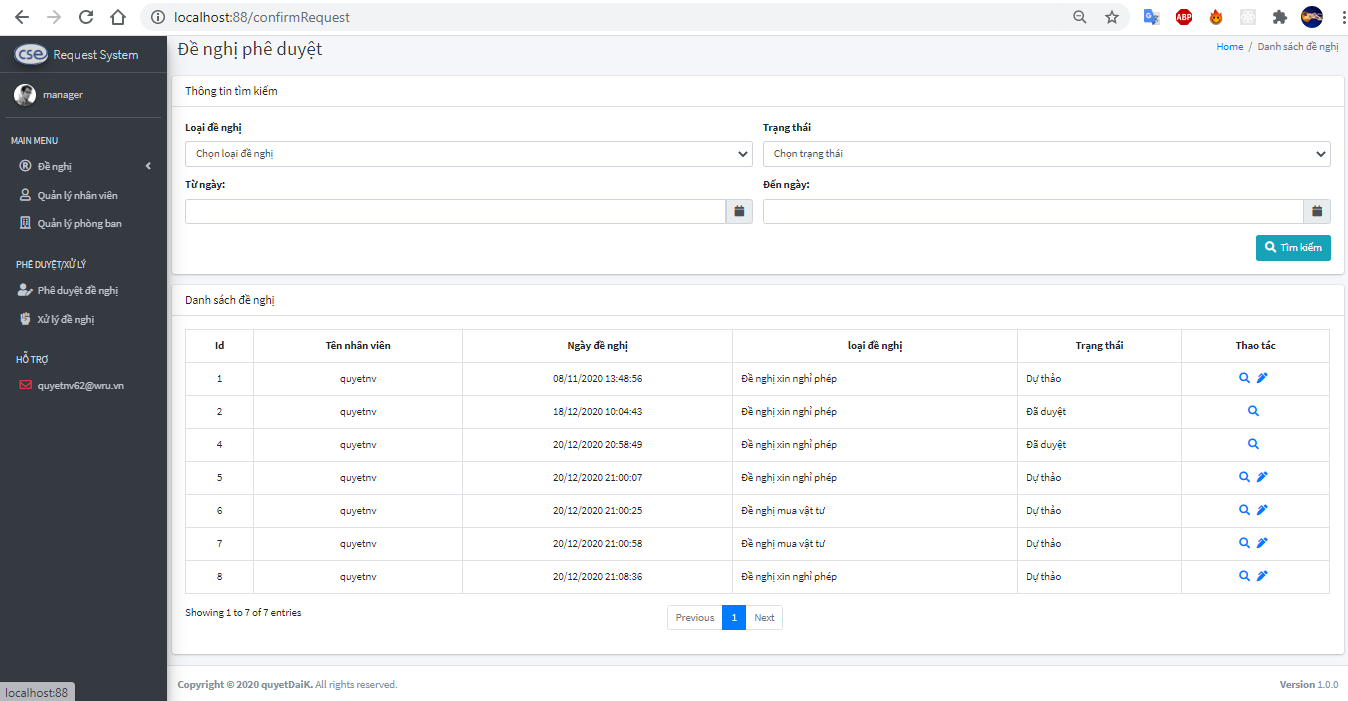
Hình 3.10 Màn hình phê duyệt đề nghị xin nghỉ phép

Màn hình phê duyệt đề nghị vật tư dành cho nhà quản lý, nhà quản lý có thể xem thông tin đề nghị, phê duyệt hoặc từ chối tại màn hình này.



Hình 3.11 Màn hình phê duyệt đề nghị vật tư

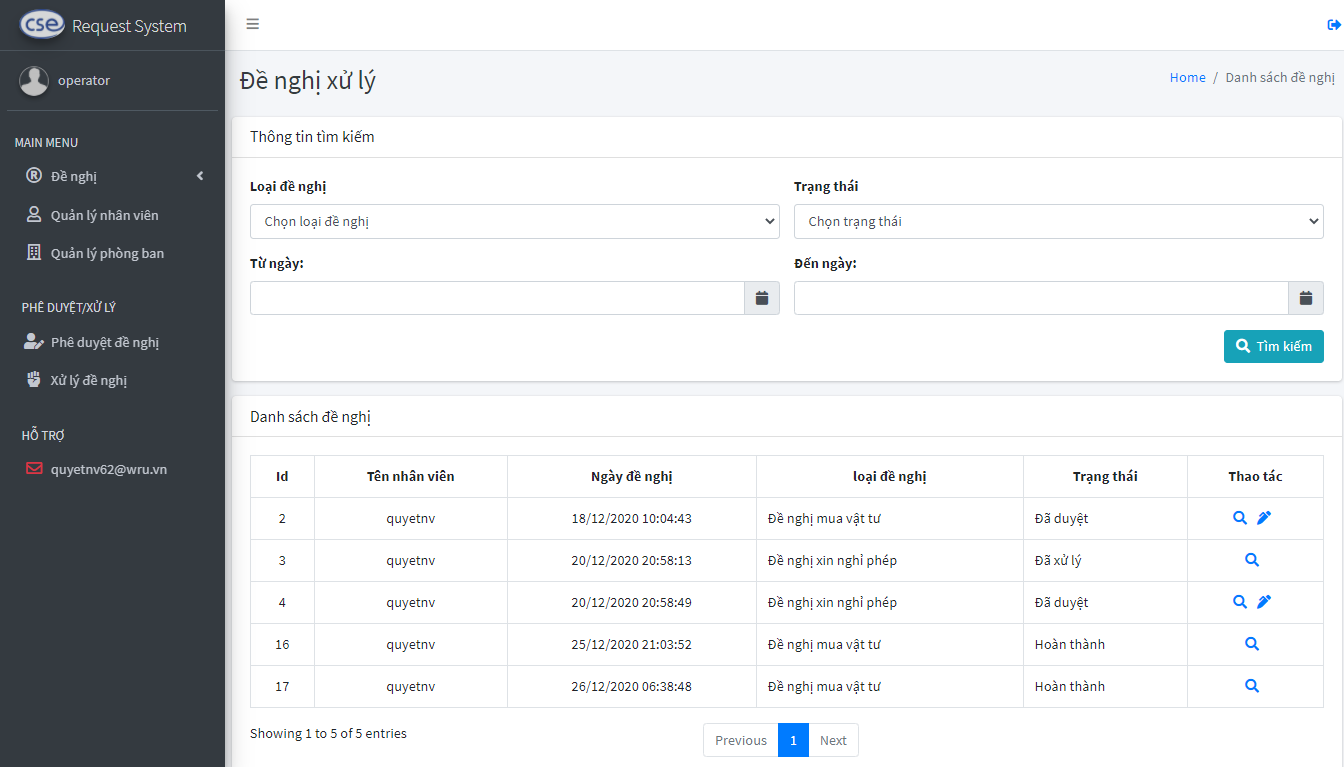
Màn hình danh sách đề nghị chờ phê duyệt của quản lý: Quản lý có thể theo dõi danh sách tất cả các đề nghị, thông tin và trạng thái của chúng ở màn hình này.



Hình 3.12 Màn hình danh sách DNDX chờ duyệt

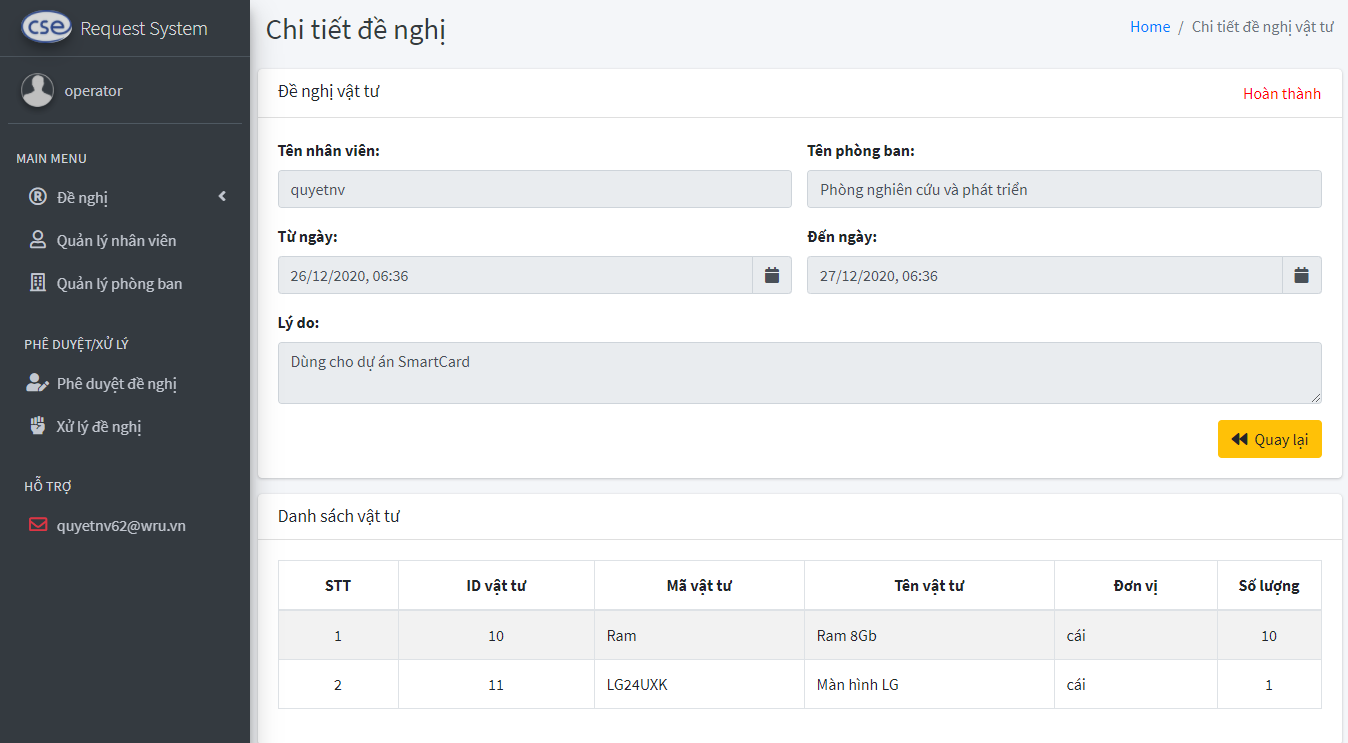
**Chức năng dành cho nhà điều hành:**

Màn hình danh sách đề nghị chờ phê duyệt của nhà điều hành: Nhà điều hành có thể theo dõi danh sách tất cả các đề nghị, thông tin và trạng thái của chúng ở màn hình này.



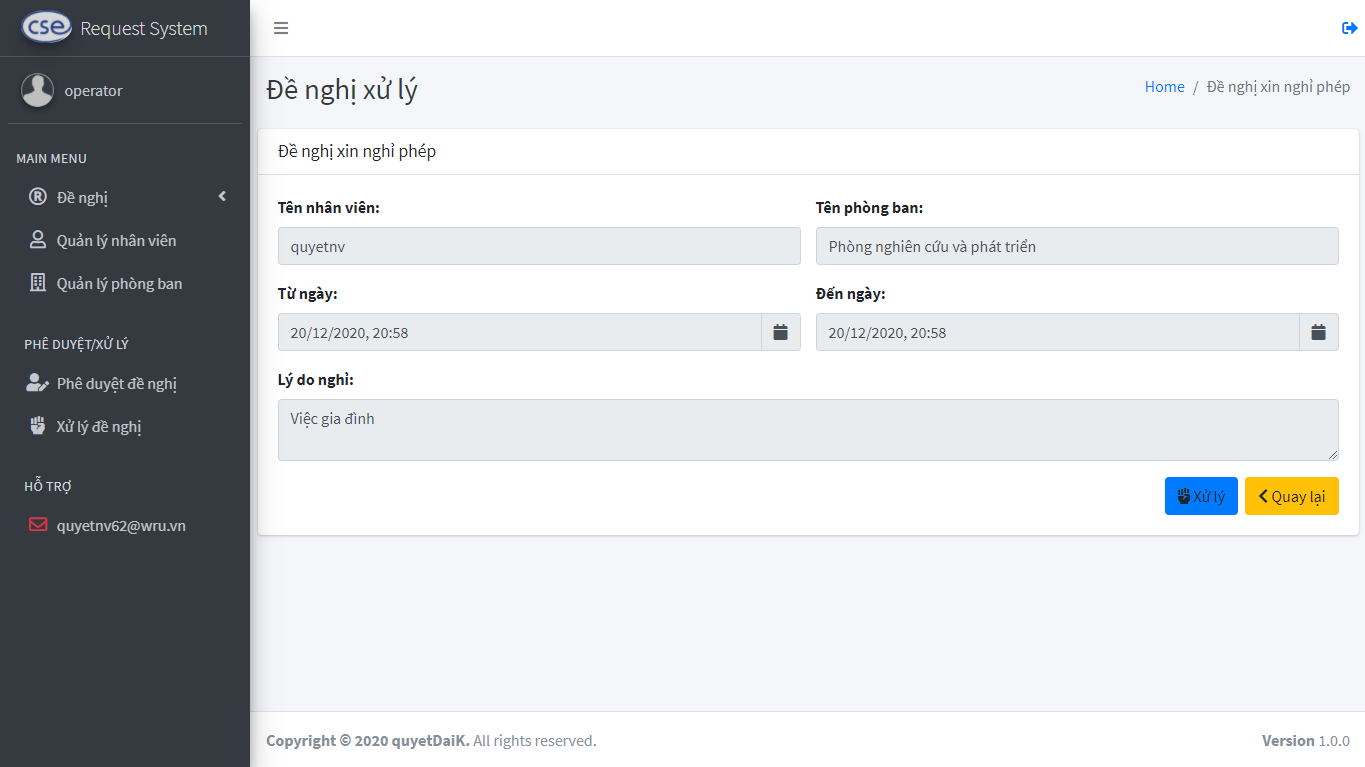
Hình 3.13 Màn hình danh sách DNDX chờ xử lý

Màn hình xử lý đề nghị vật tư dành cho nhà điều hành, nhà điều hành có thể xem thông tin đề nghị, phê duyệt hoặc từ chối tại màn hình này.



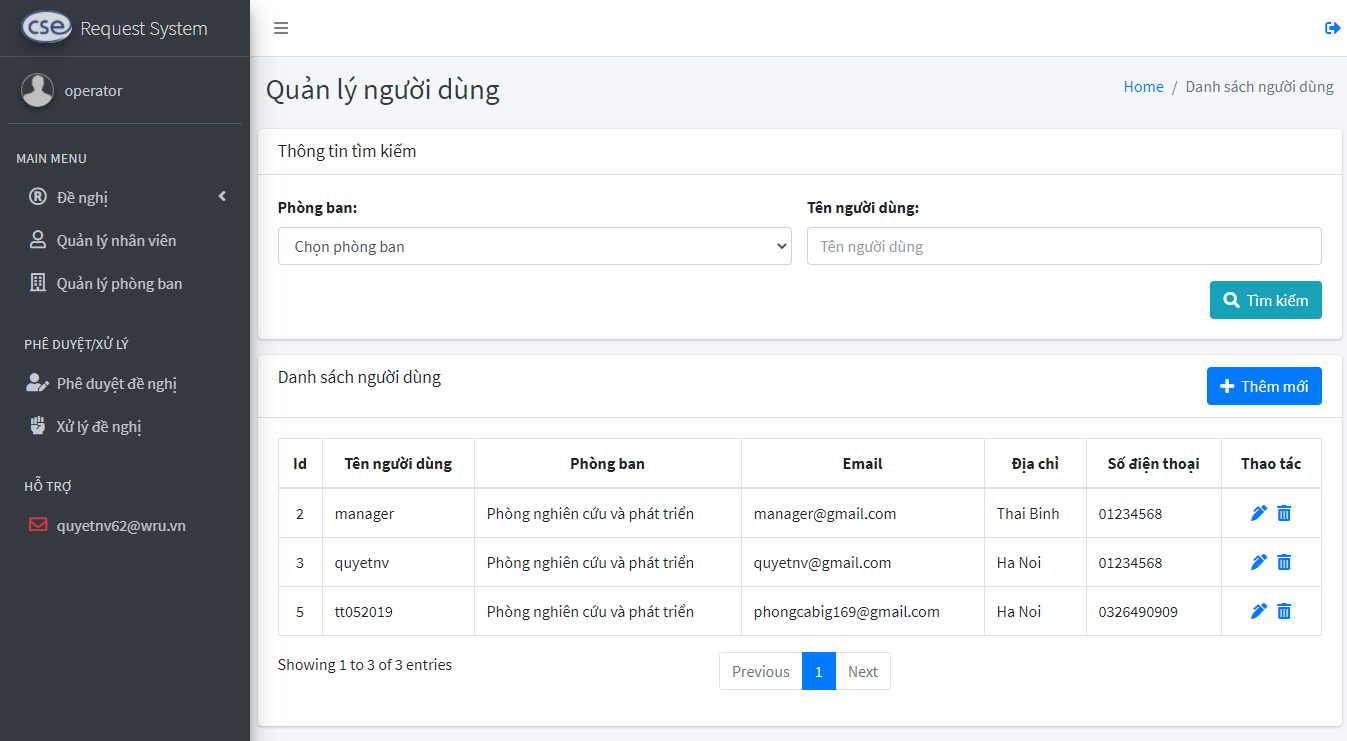
Hình 3.14 Màn hình xử lý đề nghị vật tư

Màn hình xử lý đề nghị xin nghỉ phép dành cho nhà điều hành, nhà điều hành có thể xem thông tin đề nghị, phê duyệt hoặc từ chối tại màn hình này.



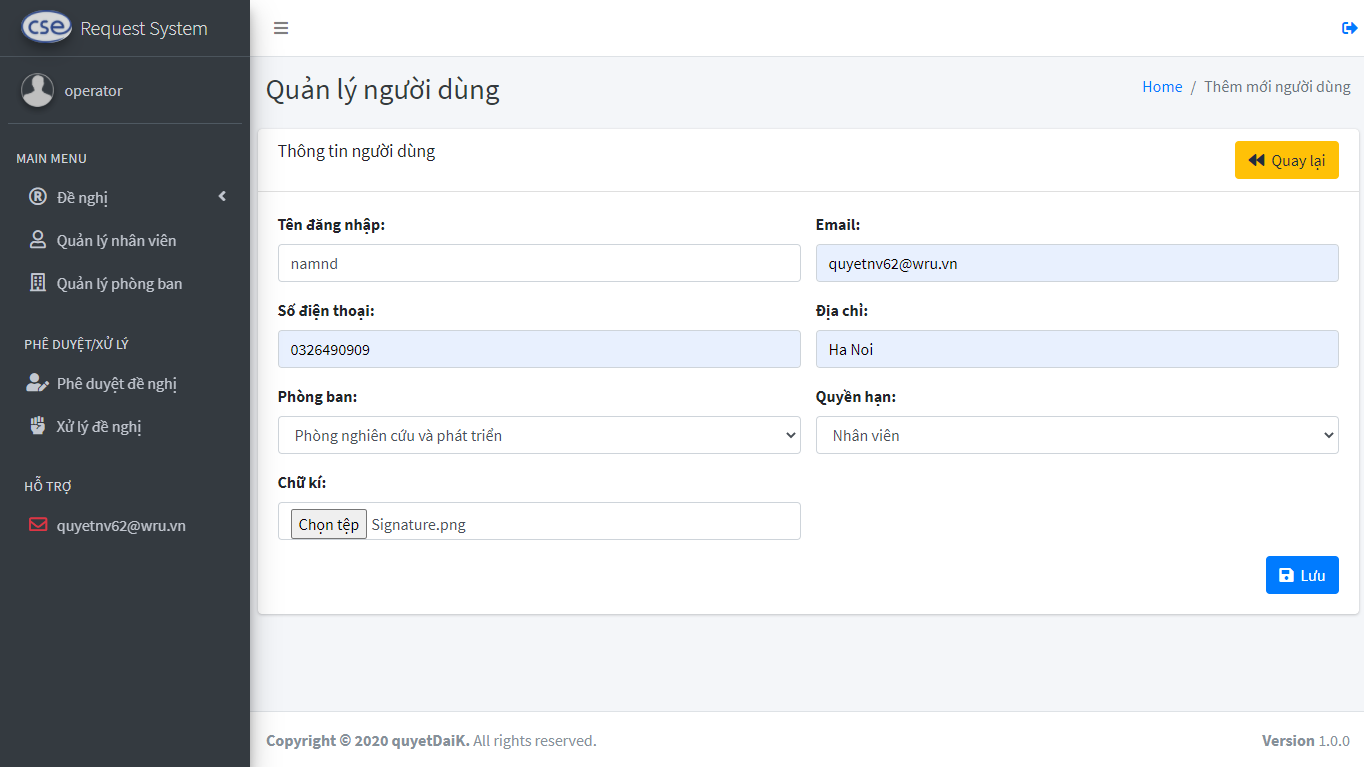
Hình 3.15 Màn hình xử lý đề nghị xin nghỉ phép

Màn hình danh sách người dùng dành cho nhà điều hành: Nhà điều hành có thể theo dõi danh sách tất cả các thông tin người dùng ở màn hình này.



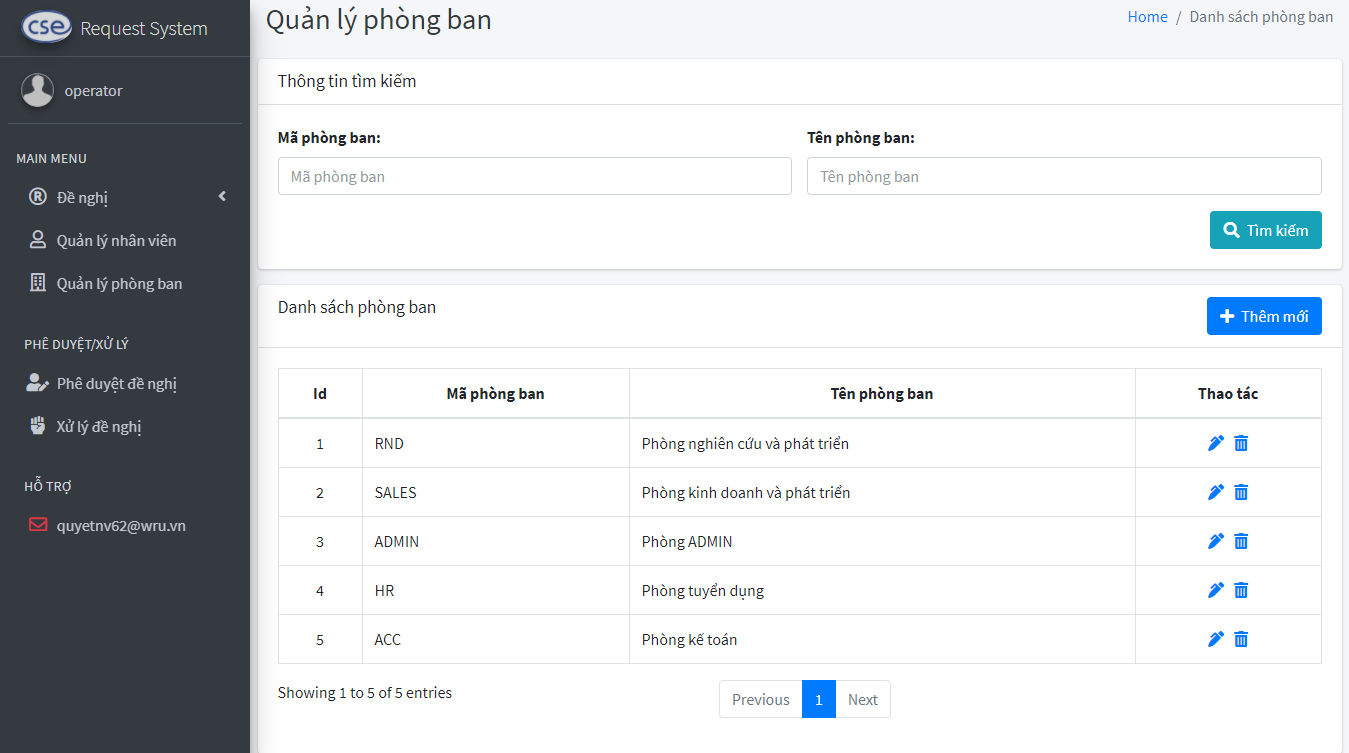
Hình 3.16 Màn hình danh sách người dùng

Màn hình nhà điều hành thêm mới người dùng. Nhà điều hành nhập thông tin người dùng mới vào đây.



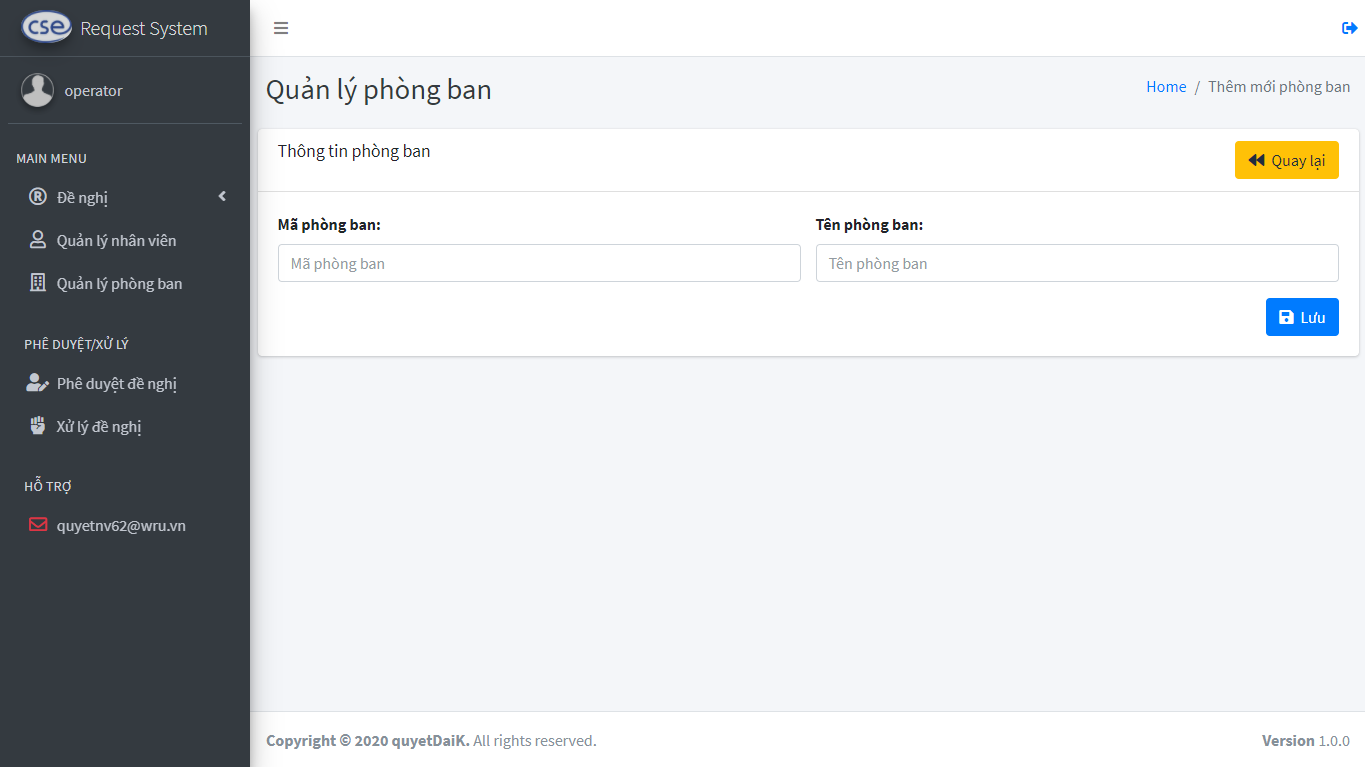
Hình 3.17 Màn hình thêm mới người dùng

Màn hình danh sách phòng bAN dành cho nhà điều hành: Nhà điều hành có thể theo dõi danh sách tất cả các thông tin phòng ban ở màn hình này.



Hình 3.18 Màn hình danh sách phòng ban

Màn hình thêm mới phòng ban dành cho nhà điều hành. Nhà điều hành nhập thông tin phòng ban mới vào đây



Hình 3.19 Màn hình thêm mới phòng ban

## Phân tích kết quả đạt được

**Kết quả đạt được:**

Tổng kết lại thì phần mềm hỗ trợ quản lý DXDN cho doanh nghiệp của em đã đáp ứng được nhu cầu quản lý DNDX cơ bản cho một công ty với hai loại đề nghị là đề nghị xin nghỉ phép và đề nghị vật tư. Điểm mạnh của phần mềm này đó chính là khả năng chuyển đổi linh hoạt giữa việc số hóa hoàn toàn quy trình đơn giản ví dụ như đề nghị xin nghỉ phép, đề nghị xin OT, đề nghị xin đến muộn,… và số hóa một phần cho các loại đề nghị mang tính chất quy trình thủ tục phức tạp cao ví dụ như đề nghị xuất kho, đề nghị mua vật tư, đề nghị tạm ứng,.. Qua đó giúp cho quy trình tạo đề nghị, phê duyệt đề nghị trở nên nhanh chóng, dễ dàng và minh bạch hơn bao giờ hết.

**Hạn chế:**

Mặc dù đã có nhiều cố gắng trong suốt quá trình làm đồ án, nhưng do thời gian có hạn và thiếu kinh nghiệm trong thực tế nên cũng không thể tránh khỏi những nhược điểm và thiếu sót như: Lượng dữ liệu chưa phong phú, thiếu sự gắn kết giữa các chức năng, các tính năng chưa thực sự hoàn thiện, thiết kế còn sơ sài chưa đẹp mắt và đặc biệt còn thiếu xót rất nhiều các loại đề nghị khác nữa.

**Hướng phát triển:**

Để hệ thống hoàn thiện hơn thì hệ thống cần phải được bổ sung thêm các tính năng như chấm công, thống kê dữ liệu chấm công, tính lương, bổ sung them nhiều loại đề nghị hơn.

# TÀI LIỆU THAM KHẢO

x

|  |  |
| --- | --- |
| [1] | TS. Đặng Thị Thu Hiền, *Cơ sơ dữ liệu*.: NXB Giao Thông Vận Tải, 2013. |
| [2] | VNU-UET, *Giáo trình lập trình hướng đối tượng với java, 2014*. |
| [3] | O Reilly, Head First Java 2nd Edition, 2010. |
| [4] | [Elisabeth Freeman](https://www.google.com/search?sa=X&biw=1366&bih=635&sxsrf=ALeKk00FYxmyDoJrQg7hRqerBgTOc_5AEw:1609278505463&q=Elisabeth+Robson&stick=H4sIAAAAAAAAAOPgE-LSz9U3MKksNLKIV0KwS7RkspOt9JPy87P1y4syS0pS8-LL84uyrRJLSzLyixaxCrjmZBYnJqWWZCgE5ScV5-ftYGUEANbP7lBOAAAA&ved=2ahUKEwjemqbNlfTtAhWVyYsBHe3VCdUQmxMoATAXegQIEhAD), Head First Design Patterns, 2004 |
| [5] | [Craig Walls](https://www.google.com/search?sa=X&biw=1366&bih=635&sxsrf=ALeKk03PGa7o9GylTEOxHsTEf5CQNJQeRg:1609278581737&q=craig+walls&stick=H4sIAAAAAAAAAOPgE-LRT9c3NErKNTE0KkpS4tLP1TcwqcrOTivWkslOttJPys_P1i8vyiwpSc2LL88vyrZKLC3JyC9axMqdXJSYma5QnpiTU7yDlREAOTdvqUsAAAA&ved=2ahUKEwinzdXxlfTtAhWRF6YKHVUzB2cQmxMoATAPegQIDxAD), Spring in Action (In Action series), 2005 |
| [6] | [Gavin King](https://www.google.com/search?sa=X&biw=1366&bih=635&sxsrf=ALeKk00dy2YVSYqlnqffP696vhJ9ppO_tw:1609278682711&q=java+persistence+with+hibernate+gavin+king&stick=H4sIAAAAAAAAAOPgE-LRT9c3NDSuKEuyMM9RgvLKK4oLLVO0ZLKTrfST8vOz9cuLMktKUvPiy_OLsq0SS0sy8osWsWplJZYlKhSkFhVnFgMlk1MVyjNLMhQyMpNSi_ISS1IV0hPLMvMUsjPz0newMgIAlmeO62wAAAA&ved=2ahUKEwiJ0-ihlvTtAhVm72EKHddzA0sQmxMoATAPegQIEBAD), Java Persistence with Hibernate, 2006 |

x