HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VIỆN KHOA HỌC KỸ THUẬT BƯU ĐIỆN

BÀI TẬP CUỐI KỲ BỘ MÔN
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG OOP

WING AIRPORT
- WEBSITE QUẢN LÝ SÂN BAY –

THÀNH VIÊN:

1. Nguyễn Hữu Nam – B23DCCC121 (NHÓM TRƯỞNG)
2. Trần Thị Bình – B23DCCC018
3. Trần Thành Duy – B23DCCC053
4. Vũ Long Nhật – B23DCCC125
5. Lê Anh Tú – B23DCCC169

Hà Nội, 11/2024

LỜI NÓI ĐẦU

Trong bối cảnh ngành hàng không phát triển không ngừng như hiện nay, việc nâng cao chất lượng dịch vụ và tối ưu hóa hoạt động của các sân bay là vô cùng cần thiết. Nắm bắt được xu thế đó, đồ án môn học "Lập trình hướng đối tượng" với đề tài "Xây dựng website quản lý sân bay Wings Airport" được thực hiện nhằm ứng dụng công nghệ thông tin vào việc nâng cao hiệu quả quản lý và vận hành của sân bay.

Đồ án tập trung vào việc phân tích, thiết kế và xây dựng một hệ thống website với đầy đủ các chức năng hỗ trợ hoạt động của Wings Airport, từ việc cung cấp thông tin cho hành khách, quản lý chuyến bay, quản lý nhân viên, đến việc thống kê báo cáo và đánh giá hiệu quả hoạt động.

Trong quá trình thực hiện đồ án, chúng em đã nhận được sự hướng dẫn tận tình của thầy Nguyễn Việt Dũng dạy môn Lập trình hướng đối tượng OOP. Mặc dù đã rất cố gắng, nhưng do hạn chế về kinh nghiệm và thời gian, chắc chắn đồ án không tránh khỏi những thiếu sót.

Chúng em rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô và các bạn để đồ án được hoàn thiện hơn.

Xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1. GIỚI THIỆU

1. Tổng quan về sân bay

Sân bay quốc tế Wings Airport hướng đến trở thành một sân bay hiện đại và quan trọng trong hệ thống giao thông hàng không của Việt Nam. Để đáp ứng nhu cầu vận chuyển ngày càng tăng của hành khách và sự phát triển nhanh chóng của ngành hàng không, Wings Airport được áp dụng và triển khai các giải pháp công nghệ tiên tiến.

Những thách thức chính mà Wings Airport phải đối mặt bao gồm:

- **Đảm bảo an ninh, an toàn**: Với số lượng hành khách và chuyến bay lớn, an ninh, an toàn luôn là ưu tiên hàng đầu. Hệ thống an ninh, giám sát phải được triển khai đồng bộ và hiện đại, đảm bảo khả năng kiểm soát nhanh chóng và chính xác, phòng ngừa các tình huống khẩn cấp một cách hiệu quả.
- **Nâng cao chất lượng dịch vụ**: Wings Airport cam kết mang đến trải nghiệm dịch vụ hoàn hảo cho hành khách. Đem đến những trải nghiệm mượt mà và thuận tiện nhất. Hệ thống và thiết kế đơn giản nhưng đẹp mắt mang lại sự hài lòng cho khách hàng dù có là lần đầu sử dụng.
- **Tối ưu hóa hoạt động**: Để duy trì tính cạnh tranh và hiệu quả, Wings Airport tập trung vào quản lý thông minh và tối ưu hóa nguồn lực. Từ việc phân bổ nhân sự, quản lý lịch trình bay đến việc sử dụng các công cụ phân tích và quản lý thông tin, tất cả đều nhằm mục tiêu tối đa hóa hiệu suất vận hành và giảm thiểu chi phí.

2. Lý do chọn đề tài

Việc lựa chọn đề tài "Xây dựng website quản lý sân bay Wings Airport" của chúng em được xem xét dựa trên những lý do sau:

- **Thứ nhất, nhu cầu thực tế**: Với sự phát triển của ngành hàng không, nhu cầu quản lý, tổ chức và cung cấp thông tin tại sân bay ngày càng tăng cao. Việc xây dựng một website quản lý sân bay không chỉ đáp ứng nhu cầu thực tế trong việc cải thiện dịch vụ khách hàng mà còn hỗ trợ quản lý hiệu quả các hoạt động của sân bay, từ lịch trình chuyến bay, đặt vé, cho đến các thông tin về hành lý và an ninh.
- **Thứ hai, tính ứng dụng cao**: Hệ thống quản lý sân bay qua website không chỉ hữu ích trong việc cung cấp thông tin chi tiết cho hành khách, mà còn tạo điều kiện để các bộ phận chức năng của sân bay làm việc một cách nhịp nhàng, đồng bộ. Qua đó, website có thể dễ dàng ứng dụng trong thực tế, hỗ

trợ các hoạt động vận hành, tăng tính tiện lợi cho người dùng và nâng cao chất lượng dịch vụ.

- **Thứ ba, phù hợp với năng lực**: Dự án này phù hợp với khả năng của nhóm trong việc thiết kế và phát triển web, cũng như áp dụng các kiến thức đã học về Lập trình hướng đối tượng cũng như cơ sở dữ . Thông qua đề tài này, chúng em có thể phát huy các kỹ năng lập trình, phân tích và thiết kế hệ thống, đồng thời tích lũy thêm kinh nghiệm thực tiễn trong lĩnh vực quản lý sân bay.
- **Thứ tư, nhu cầu hiện đại hóa hệ thống quản lý sân bay**: Với sự phát triển nhanh chóng của công nghệ và yêu cầu ngày càng cao trong việc tối ưu hóa hoạt động, các sân bay cần các hệ thống quản lý kỹ thuật số hiện đại để cải thiện hiệu suất vận hành. Việc xây dựng website quản lý không chỉ giúp Wings Airport đáp ứng nhu cầu hiện tại mà còn đặt nền tảng cho chuyển đổi số các dịch vụ trong hiện tại và tương lai.
- **Thứ năm, hỗ trợ trong việc ra quyết định**: Website quản lý sân bay có thể tích hợp các công cụ phân tích dữ liệu, giúp các nhà quản lý theo dõi và đánh giá các số liệu quan trọng như lưu lượng hành khách, thời gian khởi hành và hạ cánh, hay tình trạng vận hành của các chuyến bay. Điều này giúp đưa ra các quyết định nhanh chóng, chính xác và kịp thời, nâng cao hiệu quả hoạt động của sân bay.
- **Thứ sáu, tăng tính minh bạch và cải thiện giao tiếp**: Việc cung cấp thông tin chính xác, minh bạch về lịch trình chuyến bay, thời gian làm thủ tục và các dịch vụ hỗ trợ trên một nền tảng duy nhất sẽ giúp hành khách dễ dàng theo dõi và chuẩn bị.

Những lý do trên đã khẳng định sự cần thiết và tính khả thi của đề tài **"Xây dựng website quản lý sân bay Wings Airport"**, đồng thời tạo cơ hội để nhóm vận dụng kiến thức và kỹ năng trong lĩnh vực phát triển web và quản lý hệ thống sân bay. Đề tài này không chỉ mang tính thiết thực mà còn giúp chúng em rèn luyện và phát triển kỹ năng chuyên môn, đồng thời góp phần vào việc học và áp dụng các kiến thức vào thực tiễn xã hội.

3. Mục tiêu dự án

Dự án "Xây dựng website quản lý sân bay Wings Airport" được thực hiện với mục tiêu**:** Xây dựng một hệ thống website cơ bản, hiệu quả, đáp ứng nhu cầu quản lý và vận hành cơ bản của sân bay mô phỏng Wings Airport

3.1. Đối với nhóm thực hiện dự án:

- **Cung cấp hệ thống xử lý thông tin**:
  - **Độ chính xác và độ tin cậy:** Đảm bảo hệ thống có độ chính xác cao trong việc xử lý và quản lý thông tin, giảm thiểu lỗi và nâng cao độ tin cậy của dữ liệu.
  - **Hiệu suất và tốc độ:** Tối ưu hóa hiệu suất và tốc độ của hệ thống để đảm bảo quá trình vận hành diễn ra mượt mà và hiệu quả.
- **Áp dụng kiến thức và kỹ năng**:
  - **Thực hành thực tế:** Tận dụng cơ hội này để áp dụng lý thuyết vào thực tế, từ đó nâng cao kỹ năng lập trình, phát triển web và quản lý dự án.
  - **Học hỏi và phát triển:** Nhóm sẽ học hỏi thêm từ những thách thức gặp phải, từ đó cải thiện và phát triển kỹ năng nghề nghiệp.
- **Khơi nguồn cho các nghiên cứu và phát triển**:
  - **Đặt nền móng cho tương lai:** Dự án là tiền đề cho các nghiên cứu sâu hơn, các dự án tiếp theo về công nghệ thông tin trong lĩnh vực hàng không.
  - **Cập nhật công nghệ mới:** Khuyến khích nhóm cập nhật các xu hướng công nghệ mới và áp dụng chúng vào dự án.

3.2. Đối với người sử dụng Wings Airport:

- **Cung cấp thông tin đầy đủ và tiện lợi**:
  - **Đa dạng và chi tiết:** Cung cấp thông tin đa dạng và chi tiết về các chuyến bay, lịch trình, dịch vụ tại sân bay, các chính sách và quy định để hành khách có thể nắm bắt một cách toàn diện.
  - **Tính tương tác cao:** Website cần có tính tương tác cao, cho phép người dùng dễ dàng tìm kiếm và tra cứu thông tin cần thiết.
- **Nâng cao trải nghiệm người dùng**:
  - **Dịch vụ trực tuyến tiện lợi:** Cung cấp các dịch vụ trực tuyến như đặt vé, check-in trực tuyến, giúp hành khách tiết kiệm thời gian và nâng cao sự tiện lợi.
  - **Hỗ trợ đa ngôn ngữ:** Cung cấp giao diện và hỗ trợ hai ngôn ngữ khác nhau để phục vụ khách hàng trong nước và quốc tế.

4. Phạm vi và giới hạn dự án

4.1. Phạm vi

4.1.1. Đối tượng phục vụ:

- **Quản trị viên:**
  - Quản lý toàn bộ hệ thống và dữ liệu của website.

- Điều chỉnh và bảo trì hệ thống, đảm bảo hoạt động suôn sẻ.
- Phân quyền và quản lý tài khoản người dùng.
- Thống kê, báo cáo hoạt động của sân bay và hệ thống.

- **Hành khách:**
  - Tra cứu thông tin chuyến bay như lịch trình, thời gian khởi hành, và điểm đến.
  - Đặt vé máy bay trực tuyến một cách dễ dàng và nhanh chóng.
  - Thực hiện check-in online để tiết kiệm thời gian khi đến sân bay.
  - Sử dụng các dịch vụ của sân bay như thuê xe, đặt phòng khách sạn, và các dịch vụ tiện ích khác.
  - Gửi phản hồi và đánh giá về chất lượng dịch vụ, giúp nâng cao trải nghiệm người dùng.

4.1.2. Chức năng chính:

- **Đối với quản trị viên:**
  - **Quản lý thông tin chuyến bay:** Cập nhật và theo dõi thông tin các chuyến bay, bao gồm lịch trình, thời gian, và tình trạng chuyến bay.
  - **Quản lý nhân viên:** Điều chỉnh, phân công công việc và quản lý thông tin nhân sự.
  - **Quản lý tài khoản:** Tạo và quản lý tài khoản người dùng, thiết lập quyền truy cập.
  - **Thống kê báo cáo:** Tạo các báo cáo thống kê về hoạt động của sân bay, giúp quản trị viên nắm bắt được tình hình và ra quyết định kịp thời.
- **Đối với hành khách:**
  - **Tra cứu thông tin chuyến bay:** Cung cấp thông tin chi tiết về các chuyến bay, bao gồm thời gian khởi hành, điểm đến, và cổng lên máy bay.
  - **Đặt vé máy bay:** Cho phép hành khách đặt vé trực tuyến một cách dễ dàng và nhanh chóng.
  - **Check-in online:** Hỗ trợ hành khách làm thủ tục check-in trực tuyến, tiết kiệm thời gian và tránh xếp hàng.

4.2. Giới hạn

Do hạn chế về mặt thời gian, nguồn lực và kiến thức, dự án "Xây dựng website quản lý sân bay Wings Airport" có những giới hạn nhất định:

- **Hạn chế về dữ liệu**:
  - **Nhiều ràng buộc và trường hợp chưa xử lý:** Do thời gian có hạn, một số trường hợp và ràng buộc sử dụng dữ liệu chưa được xử lý hoặc lường

trước. Điều này có thể ảnh hưởng đến tính toàn diện và khả năng xử lý của hệ thống trong các tình huống thực tế đa dạng.

- **Dữ liệu mô phỏng:** Dữ liệu được sử dụng trong quá trình xây dựng và demo hệ thống là dữ liệu mô phỏng, không phản ánh hoàn toàn chính xác và đầy đủ các tình huống thực tế có thể xảy ra trong quá trình vận hành sân bay.

- **Hạn chế về thời gian và nguồn lực**:
  - **Thời gian hạn chế:** Với thời gian hạn chế để thực hiện dự án, nhóm không thể giải quyết hết tất cả các tình huống phức tạp và yêu cầu phát sinh.
  - **Nguồn lực hạn chế:** Do hạn chế về nguồn lực, nhóm không thể triển khai đầy đủ các tính năng nâng cao và tối ưu hóa hệ thống như mong muốn.

Mặc dù còn nhiều hạn chế, nhóm đã cố gắng hết sức để hoàn thành tốt nhất các mục tiêu đề ra, đồng thời đảm bảo tính thực tiễn và khả năng ứng dụng của đề tài. Qua đó, dự án giúp nhóm em học hỏi thêm nhiều kiến thức và kinh nghiệm làm dự án thực tiễn, đồng thời đặt nền móng cho các nghiên cứu và phát triển tiếp theo trong lĩnh vực này.

5. Phương pháp thực hiện

Để hoàn thành các mục tiêu đề ra, quá trình thực hiện dự án "Xây dựng website quản lý sân bay Wings Airport" đã nghiên cứu và sẽ áp dụng kết hợp các phương pháp nghiên cứu sau vào dự án:

5.1. Nghiên cứu lý thuyết

5.1.1. Nghiên cứu tài liệu

- **Thu thập thông tin:** Thu thập thông tin từ nhiều nguồn khác nhau như sách, báo, luận văn, và các website chuyên về lĩnh vực quản lý sân bay, công nghệ thông tin, thiết kế website.
- **Phân tích nội dung:** Phân tích các tài liệu thu thập được để hiểu rõ hơn về các khía cạnh quan trọng của quản lý sân bay và công nghệ liên quan.

5.1.2. Khảo sát các hệ thống tương tự:

- **Tìm hiểu:** Khảo sát và tìm hiểu các hệ thống quản lý sân bay hiện có trên thị trường.
- **Phân tích ưu nhược điểm:** Phân tích ưu nhược điểm của các website quản lý sân bay hiện có để rút ra những bài học và kinh nghiệm quý báu. Việc này giúp nhóm đưa ra những giải pháp cải tiến và áp dụng vào dự án của mình.

5.2. Nghiên cứu thực tiễn

5.2.1. Khảo sát thực tế

- **Tiếp cận yêu cầu thực tế:** Tiếp cận và nắm bắt các yêu cầu thực tế cơ bản từ các bên liên quan như hành khách, quản trị viên và nhân viên sân bay.
- **Thu thập thông tin:** Ghi nhận và phân tích các yêu cầu, mong muốn và khó khăn mà người dùng gặp phải trong quá trình sử dụng dịch vụ sân bay.

5.2.2. Trao đổi, thu thập ý kiến

- **Thảo luận nhóm:** Trao đổi với các thành viên trong nhóm để cùng xây dựng và triển khai hệ thống một cách hợp lý và hiệu quả.
- **Thu thập ý kiến:** Lắng nghe và tiếp thu ý kiến phản hồi từ các thành viên trong nhóm và các bên liên quan để hoàn thiện hệ thống.

5.3. Phương pháp phát triển mềm

5.3.1. Mô hình thác nước (Waterfall)

- **Khảo sát:** Thu thập và phân tích yêu cầu của dự án.
- **Phân tích:** Phân tích các yêu cầu và đề xuất giải pháp kỹ thuật.
- **Thiết kế:** Thiết kế hệ thống, bao gồm thiết kế giao diện và thiết kế chức năng.
- **Cài đặt:** Lập trình và phát triển các chức năng của hệ thống.
- **Kiểm thử:** Kiểm thử hệ thống để phát hiện và sửa lỗi.
- **Triển khai:** Triển khai hệ thống và đưa vào sử dụng.

5.3.2. Phương pháp lập trình hướng đối tượng (OOP):

- **Tái sử dụng mã:** Xây dựng hệ thống website dựa trên các đối tượng và lớp đối tượng, giúp tăng tính tái sử dụng, dễ dàng bảo trì và phát triển.
- **Thiết kế modul:** Thiết kế hệ thống theo các module riêng biệt, mỗi module chịu trách nhiệm cho một chức năng cụ thể, từ đó dễ dàng quản lý và nâng cấp hệ thống.

5.4. Phương pháp đánh giá

- **Đánh giá dựa trên tiêu chí**: Kiểm tra sản phẩm và đánh giá dựa trên các mục tiêu của dự án, kiểm tra từng chức năng và nghiệm thu sản phẩm để đảm bảo hệ thống hoạt động đúng như mong đợi.

- **Khảo sát người dùng:** Thực hiện khảo sát và phỏng vấn người dùng sau khi trải nghiệm hệ thống để thu thập phản hồi về mức độ hiệu quả và tiện lợi của hệ thống.
- **Đề xuất cải tiến:** Dựa trên phản hồi của người dùng, đề xuất các cải tiến để nâng cao chất lượng và trải nghiệm sử dụng hệ thống.

6. Công nghệ áp dụng

6.1. Java(1)

- **Lịch sử hình thành:**

  - **1991**: Java bắt đầu dưới cái tên "Oak" với mục tiêu phát triển phần mềm cho các thiết bị điện tử.
  - **1995**: Java chính thức được ra mắt và nhanh chóng trở thành ngôn ngữ lập trình phổ biến trong phát triển web, nhờ tính linh hoạt và bảo mật cao.
  - **2004**: Phiên bản Java 5.0 ra mắt với nhiều cải tiến, bao gồm các tính năng mạnh mẽ như Generics và annotations, giúp Java trở nên phù hợp hơn cho các ứng dụng lớn.
  - **2014**: Java SE 8 ra mắt với các tính năng như Lambda Expressions và API Streams, giúp tối ưu hóa khả năng xử lý đồng thời và nâng cao hiệu suất.
  - **2017 - Nay**: Oracle quyết định chuyển Java sang chu kỳ phát hành 6 tháng, cho phép bổ sung tính năng và cải tiến một cách nhanh chóng, phù hợp với xu hướng phát triển phần mềm hiện đại.

- **Giới thiệu về Java:**

  - Java là ngôn ngữ lập trình đa nền tảng (cross-platform), được phát triển bởi James Gosling tại Sun Microsystems (nay là Oracle Corporation). Ngôn ngữ lập trình này ra đời vào năm 1995 và được thiết kế để có thể chạy trên các nền tảng khác nhau, từ máy tính cá nhân đến thiết bị di động, các máy chủ và thiết bị nhúng.
  - Java sử dụng cấu trúc lập trình hướng đối tượng (object-oriented programming - OOP) và được xây dựng trên cơ sở của ngôn ngữ lập trình C++. Nó cung cấp một môi trường chạy ảo (virtual machine) gọi là Java Virtual Machine (JVM), giúp các chương trình Java có thể chạy trên nhiều nền tảng khác nhau mà không cần phải biên dịch lại.

- **Tính năng của Java**:

- **Đa nền tảng**: Java được thiết kế để có thể chạy trên nhiều nền tảng khác nhau, vì vậy nó rất phù hợp cho việc phát triển các ứng dụng đa nền tảng. Java sử dụng một máy ảo (JVM - Java Virtual Machine) để chạy mã nguồn, vì vậy mã nguồn được viết một lần và có thể chạy trên nhiều hệ điều hành khác nhau mà không cần thay đổi.
- **Quản lý bộ nhớ tự động**: Java có tính năng tự động quản lý bộ nhớ, tức là nó tự động thu dọn các vùng nhớ không sử dụng nữa để giảm thiểu các lỗi bộ nhớ. Điều này giúp cho các ứng dụng được viết bằng Java có thể chạy ổn định và tránh các lỗi liên quan đến bộ nhớ.
- **Hỗ trợ đa luồng**: Java có thể xử lý đa luồng, cho phép chương trình thực hiện nhiều tác vụ cùng một lúc. Điều này giúp cho các ứng dụng có thể chạy nhanh và hiệu quả hơn, đặc biệt là khi phải xử lý nhiều tác vụ cùng một lúc.
- **Tính bảo mật cao**: Java có các tính năng bảo mật như kiểm tra kiểu tĩnh và kiểm tra lỗi trên đường dẫn. Java được thiết kế để giảm thiểu các lỗ hổng bảo mật và các vấn đề liên quan đến an ninh.
- **Tính di động**: Java được sử dụng rộng rãi trong lĩnh vực di động bao gồm: các thư viện hỗ trợ việc phát triển ứng dụng di động, đóng gói ứng dụng thành các file .jar hoặc .apk, cung cấp các tính năng như xử lý đa nhiệm, kết nối mạng và tích hợp với các thiết bị phần cứng như máy ảnh.
- **Tính độc lập với nền tảng**: Java có thể chạy trên nhiều nền tảng khác nhau và không phụ thuộc vào bất kỳ nền tảng cụ thể nào. Điều này giúp cho các ứng dụng Java có thể được triển khai trên nhiều hệ thống khác nhau mà không cần sửa đổi mã nguồn.
- **Tính kế thừa và đa hình**: Java là ngôn ngữ lập trình hướng đối tượng (OOP), vì vậy nó có các tính năng như kế thừa, đa hình và đóng gói. Các tính năng này giúp cho mã nguồn được tái sử dụng và giảm thiểu sự trùng lặp trong mã nguồn.
- **Tính mở rộng**: Java có tính năng mở rộng, cho phép các nhà phát triển thêm các tính năng mới vào ngôn ngữ bằng cách tạo các thư viện và API riêng. Điều này giúp cho Java có thể được sử dụng trong nhiều lĩnh vực khác nhau và được phát triển theo các hướng khác nhau.

6.2. Maven(2)

- **Giới thiệu:**

- Maven là công cụ quản lý và thiết lập tự động 1 dự án phần mềm. Chủ yếu dùng cho các lập trình viên java, nhưng nó cũng có thể được dùng để xây dựng và quản lý các dự án dùng C#, Ruby, Scala hay ngôn ngữ khác.
- Maven phục vụ mục đích tương tự như Apache Ant, nhưng nó dựa trên khái niệm khác và cách hoạt động khác.
- Maven hỗ trợ việc tự động hóa các quá trình tạo dự án ban đầu, thực hiện biên dịch, kiểm thử, đóng gói và triển khai sản phẩm.
- Được phát triển bằng ngôn ngữ Java cho phép Maven chạy trên nhiều nền tảng khác nhau: Windows, Linux và Mac OS…

- **Cách hoạt động:**

- Maven dùng khái niệm Project Object Model (POM) để mô tả việc build project, các thành phần phụ thuộc và các module. Nó định nghĩa trước các target cho việc khai báo task, trình biên dịch, đóng gói và thứ tự hoạt động để mọi việc diễn ra tốt nhất.
- Trong mỗi project Maven tạo ra một file .pom, trong file này định nghĩa ra những task như task khi chạy test, task khi build và khi chạy Maven sẽ dựa vào những định nghĩa này để thao tác với project.

- **Tại sao cần Maven:**

- Khi một project do nhiều nhóm phát triển ví dụ có 2 team cùng tham gia dự án, 2 team đó ở 2 quốc gia khác nhau vì thế chúng ta luôn cần có một sự liên lạc để thông nhất trong việc lập trình vì thế phải có một cái chuẩn nào đó để tất cả mọi người cùng tuân theo, như trong việc sử dụng những thư viện nào, version của thư viện tất cả những thứ như vậy đều được Maven quản lý.
- Đối với những hệ thống lớn, phức tạp sử dụng nhiều thư viện lại đòi hỏi phải release liên tục cho nên công việc đóng gói (build & deploy), quản lý, nâng cấp và bảo trì chúng rất mất thời gian, và lúc đó ta có Maven.

6.3. Spring Boot(3)

- **Giới thiệu**: **Spring Boot** là một framework Java được sử dụng để xây dựng các ứng dụng và dịch vụ web dễ dàng và nhanh chóng. Nền tảng cung cấp các cấu hình mặc định cho một số thư viện và bộ công cụ hỗ trợ xây dựng, triển khai, quản lý ứng dụng Spring-based.

- **Lợi ích của Spring Boot**:

  - **Tối ưu hóa quá trình phát triển**: Spring Boot cung cấp cấu hình mặc định thông minh và tự động, giúp giảm thiểu việc cấu hình thủ công và tối ưu quá trình phát triển ứng dụng Java.
  - **Tích hợp tốt**: Spring Boot tích hợp tốt với nhiều công nghệ và thư viện khác trong hệ sinh thái Spring Framework. Nền tảng cho phép hệ thống dễ dàng tích hợp các module và dịch vụ khác nhau mà không cần phải lo lắng về cấu hình phức tạp.
  - **Embedded server**: Spring Boot đi kèm với các máy chủ nhúng như Tomcat, Jetty, hoặc Undertow. Đây là công cụ không thể thiếu trong việc triển khai ứng dụng một cách đơn giản mà không cần cấu hình thêm bất kỳ máy chủ nào khác.
  - **Tự động cấu hình**: Spring Boot sử dụng cơ chế cấu hình tự động thông minh, cho phép ứng dụng tự cấu hình dựa trên các thư viện và module được sử dụng.
  - **Quản lý phụ thuộc**: Spring Boot cung cấp các công cụ quản lý phụ thuộc mạnh mẽ như Maven hoặc Gradle, giúp quản lý các phụ thuộc của ứng dụng một cách hiệu quả.
  - **Monitoring và quản lý**: Spring Boot cung cấp các công cụ hỗ trợ giám sát và quản lý ứng dụng dễ dàng, bao gồm Spring Boot Actuator cho việc giám sát và quản lý ứng dụng.

6.4. SQL(4)

- **Giới thiệu**:

  - MySQL là 1 hệ thống quản trị về cơ sở dữ liệu với mã nguồn mở (được gọi tắt là RDBMS) và đang hoạt động theo mô hình dạng client-server. Đối với RDBMS - Relational Database Management System thì MySQL đã được tích hợp apache và PHP.
  - Được phát hành chính thức từ thập niên 90s, MySQL hiện đang quản lý dữ liệu qua những cơ sở dữ liệu, với mỗi một cơ sở dữ liệu hoàn toàn có thể có rất nhiều những bản quan hệ có chứa dữ liệu. Ngoài ra, MySQL cũng có cùng 1 cách thức truy xuất cũng như mã lệnh tương tự cùng với ngôn ngữ SQL.

- **Cơ chế hoạt động**:

- MySQL đang tạo ra bảng để có thể lưu trữ dữ liệu và định nghĩa về sự liên quan giữa những bảng đó
- Client sẽ trực tiếp gửi yêu cầu SQL bằng 1 lệnh đặc biệt có trên MySQL.
- Ứng dụng tại server sẽ tiến hành phản hồi thông tin cũng như trả về những kết quả trên máy client.

- **Ưu điểm**:
  - **Nhanh chóng**: Nhờ vào việc đưa ra một số những tiêu chuẩn và cho phép MySQL làm việc hiệu quả cũng như tiết kiệm chi phí, giúp gia tăng tốc độ thực thi.
  - Mạnh mẽ và khả năng mở rộng: MySQL hoàn toàn có thể xử lý số lượng lớn dữ liệu và đặc biệt hơn thế nữa thì nó còn có thể mở rộng nếu như cần thiết.
  - **Đa tính năng**: Ưu điểm MySQL là gì? MySQL hiện đang hỗ trợ nhiều những chức năng SQL rất được mong chờ từ 1 hệ quản trị CSDL quan hệ cả gián tiếp cũng như trực tiếp.
  - **Độ bảo mật cao**: MySQL là gì? Hiện tại nó đang rất thích hợp cho những ứng dụng truy cập CSDL thông qua internet khi sở hữu rất nhiều những tính năng về bảo mật và thậm chí là đang ở cấp cao.
  - **Dễ dàng sử dụng**: MySQL đang là cơ sở dữ liệu dễ sử dụng, ổn định, tốc độ cao và hoạt động trên rất nhiều những hệ điều hành đang cung cấp 1 hệ thống lớn những hàm tiện ích rất mạnh.

6.5. HTML(5)

- **Giới thiệu:**

  - **HTML** là viết tắt của cụm từ **Hypertext Markup Language** (tạm dịch là Ngôn ngữ đánh dấu siêu văn bản). **HTML** được sử dụng để tạo và cấu trúc các thành phần trong trang web hoặc ứng dụng, phân chia các đoạn văn, heading, titles, blockquotes… và **HTML** không phải là ngôn ngữ lập trình.
  - Một tài liệu HTML được hình thành bởi các phần tử HTML (HTML Elements) được quy định bằng các cặp thẻ (tag và attributes). Các cặp thẻ này được bao bọc bởi một dấu ngoặc ngọn (ví dụ `<html>`) và thường là sẽ được khai báo thành một cặp, bao gồm thẻ mở và thẻ đóng. Ví dụ, chúng ta có thể tạo một đoạn văn bằng cách đặt văn bản vào trong cặp tag mở và đóng văn bản

- **Ưu điểm**: HTML là một ngôn ngữ đánh dấu siêu văn bản nên nó sẽ có vai trò xây dựng cấu trúc siêu văn bản trên một website, hoặc khai báo các tập tin kỹ thuật số (media) như hình ảnh, video, nhạc. Cụ thể:

  - Được sử dụng rộng rãi, có rất nhiều nguồn tài nguyên hỗ trợ và cộng đồng sử dụng lớn.
  - Học đơn giản và dễ hiểu.
  - Mã nguồn mở và hoàn toàn miễn phí.
  - Markup gọn gàng và đồng nhất.
  - Tiêu chuẩn thế giới được vận hành bởi World Wide Web Consortium (W3C).
  - Dễ dàng tích hợp với các ngôn ngữ backend như PHP, Python…

- **Cách hoạt động**:

  - Khi bạn gõ ra 1 tên miền, trình duyệt mà bạn đang sử dụng sẽ kết nối tới 1 máy chủ web, bằng cách dùng 1 địa chỉ IP, vốn được thấy bằng cách phân giải tên miền đó (DNS). Máy chủ web chính là 1 máy tính được kết nối tới internet và nhận các yêu cầu tới trang web từ trình duyệt của bạn. Máy chủ sau đó sẽ gửi trả thông tin về trình duyệt của bạn, là 1 tài liệu HTML, để hiển thị trang web
  - Một tập tin HTML sẽ bao gồm các phần tử HTML và được lưu lại dưới đuôi mở rộng là **.html** hoặc **.htm**. Khi một tập tin HTML được hình thành, việc xử lý nó sẽ do trình duyệt web đảm nhận. Trình duyệt sẽ đóng vai trò đọc hiểu nội dung HTML từ các thẻ bên trong và sẽ chuyển sang dạng văn bản đã được đánh dấu để đọc, nghe hoặc hiểu (do các bot máy tính hiểu).

6.6. Docker(6)

Giới thiệu: Nền tảng phần mềm cho phép xây dựng, kiểm thử triển khai ứng dụng trong package virtual containerized environment. Docker cho phép apps hoạt động bình thường ở mọi nơi, dù trên bất cứ OS nào.

- **Docker File**: Design cho Docker Image. File không có phần mở rộng (đuôi file).
- **Docker Image**: chỉ tồn tại ở dạng read-only chứa cách để tạo ra các container. Là snapshot hoặc blueprint của libraries và dependances cần có trong container nhằm phục vụ cho application.
- **Docker container**:

- Môi trường cách ly để chạy ứng dụng. Deploy trên bất cứ machine nào mà không gặp compatibility issue. Giúp software bảo toàn tính nguyên vẹn hệ thống, dễ dàng sử dụng, đơn giản hơn khi develop, dễ dàng bảo trì và deploy.
- Mỗi Container hoạt động như một micro PC, với OS và CPU độc lập, memory, Network resources => add/remove/stop/restart mà không ảnh hưởng các container khác hay host machine
- Mỗi Container thường chạy một công việc nhất định sau đó được nối vào network Khác với Virtual machines, resource được chia sẻ trực tiếp với host => cho phép chạy nhiều Docker Container. Docker ít tốn dung lượng ổ cứng do tái sử dụng file bằng layered file system. Nếu có nhiều Docker image cùng sử dụng một base image một lúc, Docker sẽ chỉ giữ lại một bản copy của file cần thiết và chia sẻ chúng với các container.

- **Docker compose**: Docker Compose là công cụ để định nghĩa và chạy multi-container cho Docker applications. Compose sử dụng file YAML để config các service cho applications. Sau đó dùng command để khởi tạo và chạy từ các config đó.

  - **Simplified control**: Docker compose cho phép định nghĩa và quản lí multi-container trong một file YAML. Giúp dễ dàng replicate application environment.
  - **Efficient collaboration**: Docker compose configuration files dễ chia sẽ giữa developers, ops team, và stakeholders.
  - **Rapid application development**: Compose catches configurations được dùng để tạo ra container. Nếu restart một service cũ, Compose sẽ tái sử dụng lại các containers đã tồn tại.
  - **Portability across environment**: Compose supports variable trong Compose files. Ta có thể dùng variables này để customize composition for các môi trường/user khác nhau.

CHƯƠNG 2. PHÂN TÍCH YÊU CẦU

Để đáp ứng nhu cầu quản lý và vận hành sân bay Wings Airport, cũng như mang đến trải nghiệm tốt nhất cho khách hàng và nhân viên, website cần đáp ứng các yêu cầu chức năng sau:

1. Yêu cầu chức năng

1.1. Yêu cầu chức năng khách hàng

- Giao diện trực quan sinh động dễ nhìn
- Đăng ký thông tin khách hàng
- Đăng ký thông tin đặt chỗ
- Tìm kiếm chuyến bay phù hợp

1.2. Yêu cầu chức năng cho quản trị viên

- Quản lý thông tin khách hàng
- Quản lý thông tin nhân viên
- Quản lý thông tin máy bay
- Quản lý thông tin loại máy bay đang có ở sân bay
- Quản lý thông tin chuyến bay
- Quản lý thông tin lịch bay
- Quản lý thông tin đặt chỗ
- Quản lý thông tin khả năng
- Quản lý thông tin phân công

2. Yêu cầu phi chức năng

Ngoài các yêu cầu chức năng đã đề cập, hệ thống website quản lý sân bay Wings Airport cần đáp ứng các yêu cầu phi chức năng sau để đảm bảo tính khả dụng, hiệu quả và bảo mật:

2.1. Yêu cầu hiệu năng

- **Thời gian đáp ứng:**

  - **Nhanh chóng:** Hệ thống cần đảm bảo thời gian đáp ứng nhanh chóng cho các thao tác của người dùng, đặc biệt là trong các hoạt động tra cứu thông tin và thực hiện các thao tác trực tuyến.
  - **Tối ưu hóa:** Các yếu tố kỹ thuật cần được tối ưu hóa để giảm thiểu thời gian tải trang và xử lý dữ liệu.

- **Khả năng xử lý đồng thời**:

  - **Quản lý tải:** Hệ thống cần có khả năng xử lý được lượng truy cập đồng thời lớn mà không làm giảm hiệu suất, đảm bảo dịch vụ ổn định cho tất cả người dùng.
  - **Cân bằng tải:** Áp dụng các kỹ thuật cân bằng tải để phân phối lượng truy cập đều và tránh tình trạng quá tải.

- **Khả năng mở rộng:**

  - **Mở rộng linh hoạt:** Hệ thống cần có khả năng mở rộng linh hoạt để đáp ứng nhu cầu sử dụng ngày càng tăng của người dùng, đặc biệt là trong các giai đoạn cao điểm.
  - **Thiết kế mô đun:** Thiết kế hệ thống theo mô đun để dễ dàng nâng cấp và mở rộng khi cần thiết.

2.2. Yêu cầu về bảo mật

- **Bảo mật API:**

  - **Ngăn chặn truy cập trái phép:** Hệ thống cần ngăn chặn các truy cập trái phép vào API, đảm bảo chỉ những yêu cầu hợp lệ và được ủy quyền mới được phép truy cập.
  - **Xác thực và ủy quyền:** Áp dụng các phương pháp xác thực và ủy quyền để bảo vệ API, như sử dụng token hoặc OAuth.

- **Phân quyền truy cập API:**

  - **Phân quyền rõ ràng:** Áp dụng cơ chế phân quyền truy cập cho các nhóm người dùng khác nhau, đảm bảo mỗi người dùng chỉ có thể truy cập và thao tác trên những chức năng được phép.
  - **Kiểm soát truy cập:** Đảm bảo mọi truy cập và thao tác trên API đều được ghi nhận và kiểm soát để ngăn ngừa và phát hiện các hoạt động bất thường.

- **Bảo mật tài khoản:**

  - **Mã hóa thông tin:** Thông tin về tài khoản người dùng cần được lưu trữ an toàn trong cơ sở dữ liệu và mật khẩu được mã hóa để bảo vệ trước các tấn công xâm nhập.

- **Bảo vệ dữ liệu người dùng:** Áp dụng các biện pháp bảo vệ dữ liệu người dùng, bao gồm mã hóa dữ liệu và sử dụng các phương pháp bảo mật tiên tiến.

2.3. Yêu cầu về giao diện người dùng

- **Giao diện thân thiện, dễ sử dụng:**
  - **Thiết kế trực quan:** Giao diện website cần được thiết kế trực quan, dễ hiểu và dễ sử dụng, phù hợp với nhiều đối tượng người dùng khác nhau.
  - **Tương tác người dùng:** Tối ưu hóa trải nghiệm tương tác của người dùng, đảm bảo mọi thao tác trên website đều dễ dàng và thuận tiện.
- Tương thích với nhiều thiết bị:
  - **Đa nền tảng:** Website cần hiển thị chính xác và hoạt động ổn định trên nhiều thiết bị khác nhau như máy tính, laptop, điện thoại thông minh, máy tính bảng.
  - **Responsive design:** Thiết kế responsive để đảm bảo website hiển thị tốt trên mọi kích thước màn hình và độ phân giải khác nhau.
- **Ngôn ngữ:** Hỗ trợ đa ngôn ngữ, ít nhất là tiếng Việt và tiếng Anh, giúp tiếp cận được nhiều đối tượng người dùng hơn (hiện đang trong giai đoạn phát triển).

2.4. Yêu cầu về kỹ thuật

- Công nghệ:
  - **Sử dụng công nghệ phổ biến:** Hệ thống được xây dựng bằng các công nghệ phổ biến và đáng tin cậy như HTML, JavaScript, CSS, Java, giúp dễ dàng bảo trì và nâng cấp.
  - **Tính linh hoạt:** Lựa chọn công nghệ linh hoạt để dễ dàng tích hợp với các công nghệ mới và cải tiến trong tương lai.
- **Khả năng tích hợp:** Hệ thống cần có khả năng tích hợp với các hệ thống khác để mở rộng chức năng và dịch vụ (sẽ được phát triển trong tương lai xa).

CHƯƠNG 3. PHÂN TÍCH HỆ THỐNG

Hệ thống website quản lý sân bay Wings Airport được xây dựng nhằm mục tiêu tự động hóa các quy trình, nâng cao hiệu quả quản lý cung cấp dịch vụ tốt hơn cho hành khách. Hệ thống bao gồm các thành phần chính sau:

1. Phân tích hệ thống(7)

1.1. Tổng quan hệ thống

Hệ thống website quản lý sân bay Wings Airport được xây dựng theo mô hình client - server phổ biến hiện nay với các thành phần chính:

1.1.1. Front-end (Client Side)

- Các ngôn ngữ cơ bản trong lập trình web như **HTML, CSS, JavaScript** được sử dụng để xây dựng giao diện người dùng, bao gồm các trang web và các thành phần động có thể tương tác.
- **Spring Boot Thymeleaf**: Sử dụng Thymeleaf như một template engine tích hợp với Spring Boot để tạo ra các trang HTML động, giúp hiển thị dữ liệu từ server một cách dễ dàng và hiệu quả.

1.1.2. Back-end (Server Side)

- **Spring Boot**: Framework Java được sử dụng để phát phần backend của hệ thống, quản lý logic các nghiệp vụ và kết nối với cơ sở dữ liệu
- **RESTful API**: Cung cấp các dịch vụ web thông qua các API RESTful, cho phép client truy cập và khai thác dữ liệu
- **Maven**: Công cụ quản lý dự án và các dependence, giúp dễ dàng xây dựng và triển khai ứng dụng

- Mô hình MVC:

  - **Model**: Chứa các lớp dữ liệu và logic nghiệp vụ cho admin quản lý sân bay
  - **View**: Chứa các trang HTML được tạo bởi Thymeleaf, hiển thị dữ liệu từ Model
  - **Controller**: Xử lý các yêu cầu dữ liệu từ người dùng, tương tác với Model và trả về View phù hợp

1.1.3. Database

- **Cơ sở dữ liệu quan hệ (RDBMS):** Sử dụng hệ quản trị cơ sở dữ liệu MySQL để lưu trữ và quản lý dữ liệu

- **ORM (Object-Relational Mapping):** Sử dụng Hibernate hoặc JPA để ánh xạ các đối tượng Java với các bảng trong cơ sở dữ liệu.

1.2. Cấu trúc hệ thống chi tiết(8)

1.2.1. Front-end

- **Giao diện người dùng UI:**

  - **Trang chủ**: Hiển thị các thông tin cơ bản và cho phép người dùng đăng ký/đăng nhập, đặt vé cũng như tra cứu thông tin về các chuyến bay
  - **Trang Admin**: Hiển thị Dashboard trực quan về thông tin lưu trữ của sân bay. Đồng thời là công cụ cho phép admin tra cứu và thay đổi dữ liệu trong database thông qua UI dễ hiểu

- **Thymeleaf:**

  - **Template engine:** Sử dụng Thymeleaf để tạo các trang HTML động, hiển thị dữ liệu từ server một cách hiệu quả, linh hoạt và dễ hiểu nhất cho người sử dụng.
  - **Tích hợp với Spring Boot**: Thymeleaf được tích hợp chặt chẽ với Spring Boot, giúp dễ dàng phát triển và duy trì website.

1.2.2. Back-end

- **Mô hình MVC:**

  - Model: các class dữ liệu chứa các lớp dữ liệu biểu diễn dữ liệu như thông tin chuyến bay, thông tin của người dùng, đặt vé, mail,...

  - **View:** Là các trang HTML động được tạo ra bởi Thymeleaf, hiển thị dữ liệu từ model cho phép người dùng thao tác với website qua hệ thống giao diện của trang

  - **Controller:**

    - **Xử lý yêu cầu:** Nhận và xử lý các yêu cầu về đặt chỗ, thay đổi chuyến bay, lịch bay,... và các thay đổi từ phía người dùng sau đó tương tác với Model và trả về View phù hợp
    - **Quản lý luồng dữ liệu:** Điều phối luồng dữ liệu giữa người dùng và hệ thống, đảm bảo các thao tác diễn ra mượt mà và đúng logic

2. Mô hình dữ liệu (ERD)

2.1. Mô tả các bảng dữ liệu

Các bảng của cơ sở dữ liệu được xây dựng và chia thành các nhóm đối tượng với đặc trưng và chức năng riêng biệt phục vụ cho các mục đích khác nhau nhưng vẫn đồng bộ và nhất quán:

2.1.1. Nhóm quản trị Admin

- Chứa tên đăng nhập và mật khẩu của các quản lý hệ thống của sân bay. Mật khẩu được mã hóa về dạng SHA không trùng nhau và không thể revert nhằm đảm bảo bảo mật tuyệt đối.
- Đồng thời là công cụ để quản lý và kiểm quyền truy cập vào hệ thống của các quản trị viên

2.1.2. Nhóm Nhân viên

a. NhanVien

- Không một doanh nghiệp nào có thể hoạt động mà thiếu nơi lưu trữ thông tin của toàn bộ nhân viên đang làm việc. Ngoài một số thông tin về danh tính thì bảng nhân viên còn chứa các chức vụ đồng thời là cả lương của các cấp bậc phục vụ cho các công tác nghiệp vụ của kế toán.
- **Quản lý nhân sự:** Giúp các thành viên bên HR theo dõi và quản lý thông tin nhân sự hiện có

b. PhanCong

- **Lịch làm việc**: Được liên kết với bảng Nhân viên và mã chuyến bay quản lý việc phân công và lên lịch trình làm việc cho từng nhân viên theo từng chuyến bay cụ thể, theo dõi lịch trình và trách nhiệm của từng nhân viên
- **Tăng hiệu quả:** Giúp tối ưu hóa việc sử dụng nguồn lực nhân sự

2.1.3. Nhóm Khách Hàng

a. KhachHang

- **Thông tin khách hàng**: Lưu trữ thông tin chi tiết về các khách hàng sử dụng dịch vụ của sân bay
- Làm số liệu cho các điều tra, thống kế cho báo cáo về tình trạng hoạt động của sân bay phục vụ cho nghiệp vụ chuyên ngành khác

b. DatCho

- **Thông tin đặt chỗ**: Được liên kết với bảng ChuyenBay qua mã chuyến bay để quản lý thông tin đặt chỗ của khách hàng và phục vụ cho checkin trước chuyến bay
- **Hỗ trợ dịch vụ:** Giúp cải thiện dịch vụ chăm sóc khách hàng và đảm bảo thông báo cho khách hàng về các thông tin thay đổi và sự hỗ trợ kịp thời

2.1.4. Nhóm Máy bay

a. ChuyenBay

- **Thông tin chuyến bay**: Lưu trữ thông tin chi tiết của các chuyến bay đến và rời sân bay, gồm mã chuyến bay, lịch trình và thông tin khởi hành, đích đến
- **Quản lý chuyến bay:** Giúp quản lý lịch trình và thông tin các chuyến bay không chỉ của Wing Airport mà còn là của các sân bay khác hiệu quả, tạo thành công cụ cho phép trao đổi thông giữa các sân bay với nhau

b. LichBay

- **Lịch trình bay**: Quản lý lịch bay của các chuyến bay, bao gồm ngày đi/đến, mã chuyến và loại máy bay được sử dụng
- **Hỗ trợ điều phối chuyến bay**: Hỗ trợ điều phối và quản lý lịch trình bay cũng như phi hành đoàn và lịch trình của nhân viên trong đội bay

c. MayBay

- **Thông tin máy bay:** Lưu trữ thông tin về máy bay gồm số hiệu, loại máy bay cũng như số ghế
- **Quản lý phương tiện**: Hỗ trợ công tác theo dõi và quản lý của các đài không lưu, bên cạnh đó là cơ sở để xây dựng lịch trình bay

d. LoaiMayBay

- **Lưu trữ thông tin các loại máy bay**: như mã loại và hãng sản xuất
- **Phân loại phương tiện**: Hỗ trợ việc phân loại và quản lý các loại máy bay có mặt ở sân bay

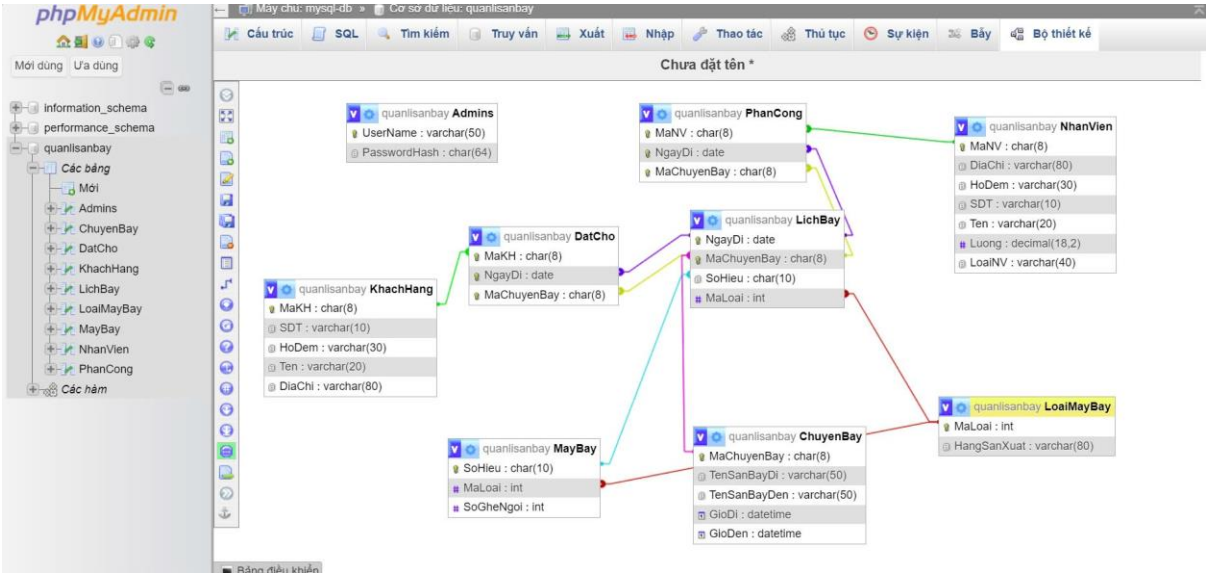
Kết nối giữa các bảng

Các bảng dữ liệu trong hệ thống được liên kết với nhau thông qua các khóa chính (Primary Key) và khóa ngoại (Foreign Key) như sau:

- **Admins:** Không kết nối trực tiếp với các bảng khác nhưng có quyền quản lý và truy cập hệ thống.

- **NhanVien và PhanCong:** Bảng PhanCong sử dụng **MaNV** từ bảng NhanVien làm khóa ngoại để xác định nhân viên được phân công cho các chuyến bay cụ thể.
- **KhachHang và DatCho:** Bảng DatCho sử dụng **MaKH** từ bảng KhachHang làm khóa ngoại để xác định khách hàng đã đặt vé cho các chuyến bay cụ thể.
- **ChuyenBay và LichBay:** Bảng LichBay sử dụng **MaChuyenBay** từ bảng ChuyenBay làm khóa ngoại để lập lịch cho các chuyến bay cụ thể.
- **MayBay và LoaiMayBay:** Bảng MayBay sử dụng **MaLoai** từ bảng LoaiMayBay làm khóa ngoại để xác định loại máy bay.

2.2. Sơ đồ hệ thống Database



Chương3_hình2.2.1. Database Diagram

CHƯƠNG 4. FRONT-END

1. Mô hình Client - Server và Web-based

1.1. Mô hình Client - Server

- Hệ thống bao gồm 2 thành phần chính:

  - Client: Là các thiết bị truy cập của người dùng như máy tính, laptop, điện thoại thông minh, máy tính bảng,... Người dùng tương tác với hệ thống thông qua giao diện website trên trình duyệt web của thiết bị.
  - Server: Là máy chủ lưu trữ website, cơ sở dữ liệu và xử lý các yêu cầu từ phía Client, là máy chủ vật lý của chúng em.

1.2. Mô hình Web-based

- Toàn bộ hệ thống được triển khai trên nền tảng web, người dùng có thể truy cập và sử dụng hệ thống từ bất kỳ đâu có kết nối Internet.
- Việc sử dụng mô hình Web-based giúp đơn giản hóa việc cài đặt, bảo trì và nâng cấp hệ thống.

Giải thích:

- **Client:** Gửi yêu cầu truy cập website đến Web server.
- **Web Server:** Nhận yêu cầu từ Client, xử lý yêu cầu (nếu cần thiết) và truy vấn dữ liệu từ Database server. Sau đó, Web server trả kết quả về cho Client.
- Database Server: Lưu trữ và quản lý dữ liệu của hệ thống.
- Ưu điểm của kiến trúc:

  - Khả năng mở rộng: Dễ dàng mở rộng hệ thống bằng cách nâng cấp phần cứng của Server hoặc bổ sung thêm Server.
  - Khả năng truy cập: Người dùng có thể truy cập hệ thống từ bất kỳ đâu có kết nối Internet.
  - Dễ dàng bảo trì: Việc bảo trì, cập nhật hệ thống được thực hiện tập trung tại Server.
  - Chi phí thấp: Giảm thiểu chi phí đầu tư phần cứng, phần mềm cho Client.
- Nhược điểm:
  - Phụ thuộc vào kết nối Internet: Hệ thống sẽ không hoạt động nếu không có kết nối Internet.

- Vấn đề bảo mật: Cần có biện pháp bảo mật chặt chẽ để ngăn chặn các cuộc tấn công từ chối dịch vụ (DoS/DDoS), tấn công xâm nhập dữ liệu,...

2. Thiết kế giao diện người dùng

2.1. Mục tiêu hệ thống

- **Tầm quan trọng của giao diện người dùng trong hệ thống sân bay:** Giao diện người dùng (UI) là phần trực tiếp tiếp xúc với người dùng và là yếu tố quyết định đến trải nghiệm tổng thể khi họ sử dụng hệ thống. Trong một hệ thống sân bay, UI không chỉ là bề ngoài của website mà còn là cầu nối giữa người dùng và các chức năng của hệ thống. Khi giao diện được thiết kế đẹp mắt, dễ hiểu, người dùng sẽ cảm thấy thoải mái và tự tin khi thao tác, tạo ra ấn tượng tích cực về hệ thống cũng như dịch vụ sân bay.
- **Yêu cầu thẩm mỹ và thân thiện với người dùng:** Giao diện cần đảm bảo yếu tố thẩm mỹ, mang lại cảm giác hiện đại, chuyên nghiệp, và phù hợp với chủ đề của sân bay. Hình ảnh, màu sắc, biểu tượng và cách bố trí phải được lựa chọn kỹ lưỡng để tạo ra một tổng thể hài hòa, giúp người dùng cảm thấy dễ chịu khi sử dụng hệ thống. Bên cạnh đó, giao diện cần thân thiện, có cấu trúc đơn giản và dễ hiểu để mọi đối tượng, kể cả những người không quen thuộc với công nghệ, đều có thể dễ dàng thao tác và tìm kiếm thông tin.
- **Tính dễ sử dụng và phù hợp với từng nhóm đối tượng người dùng:** Việc thiết kế UI cũng cần chú ý đến tính dễ sử dụng, đảm bảo các thao tác được thực hiện nhanh chóng và mượt mà. Các chức năng quan trọng phải được hiển thị rõ ràng, dễ tìm kiếm để người dùng không gặp khó khăn khi muốn thực hiện các thao tác như tra cứu thông tin chuyến bay, đặt vé, hoặc tìm kiếm dịch vụ hỗ trợ. Ngoài ra, cần tối ưu hóa giao diện để phù hợp với từng nhóm đối tượng người dùng khác nhau: từ khách hàng đến quản trị viên và nhân viên sân bay, mỗi nhóm đều có nhu cầu và cách sử dụng khác nhau, đòi hỏi sự điều chỉnh và linh hoạt trong thiết kế để tối ưu hóa trải nghiệm của từng đối tượng.

2.2. Giao diện cho khách hàng

Hệ thống đặt vé và tra cứu thông tin của sân bay được xây dựng nhằm cung cấp cho hành khách một giao diện thân thiện và đơn giản, giúp họ có thể dễ dàng truy cập vào các thông tin cần thiết, nhanh chóng tiến hành đặt vé, và trải nghiệm các dịch vụ của sân bay một cách thuận tiện. Việc tạo ra một hệ thống hiệu quả, thân thiện với người dùng là một trong những ưu tiên hàng đầu nhằm tối ưu hóa sự hài lòng của khách hàng. Thông qua giao diện này, hành khách sẽ có thể thực hiện mọi thao tác cơ bản như tìm kiếm, kiểm tra trạng thái chuyến bay, đăng ký

thông tin cá nhân, theo dõi các thông báo mới nhất từ sân bay, và thậm chí là đăng ký nhận tin tức một cách nhanh chóng. Mục tiêu cuối cùng của hệ thống là giúp hành khách tiết kiệm thời gian và gia tăng tính tiện lợi khi sử dụng dịch vụ, từ đó nâng cao sự tin tưởng và hài lòng của họ với dịch vụ tại sân bay.

2.2.1. Các chức năng chính của hệ thống

Trang web quản lý sân bay được thiết kế với các chức năng cốt lõi đáp ứng nhu cầu của khách hàng, nhằm tạo ra một trải nghiệm người dùng dễ dàng và liền mạch. Một trong những chức năng chính của hệ thống là cung cấp thông tin cập nhật về sân bay, cho phép hành khách dễ dàng theo dõi các tin tức mới nhất, bao gồm thông tin về lịch trình chuyến bay, những thay đổi quan trọng, các sự kiện đặc biệt, và các thông báo từ sân bay. Ngoài ra, trang web còn cung cấp thông tin giới thiệu về sân bay, giúp khách hàng có cái nhìn rõ ràng hơn về các dịch vụ và tiện ích hiện có. Hệ thống cũng bao gồm chức năng đăng ký nhận tin báo, nơi hành khách có thể đăng ký để nhận thông báo qua email hoặc tin nhắn về các thay đổi, cập nhật liên quan đến chuyến bay và dịch vụ của sân bay, giúp họ luôn nhận được những thông tin quan trọng một cách kịp thời.

Ngoài việc cung cấp thông tin, hệ thống còn cho phép khách hàng đăng ký thông tin cá nhân một cách nhanh chóng và bảo mật. Hành khách có thể dễ dàng cung cấp các thông tin cần thiết để lưu vào hồ sơ khách hàng và sử dụng cho các giao dịch khác trong tương lai. Đồng thời, chức năng đăng ký đặt chỗ sẽ hỗ trợ khách hàng trong việc đặt vé nhanh chóng, từ đó rút ngắn quy trình đặt vé truyền thống và giúp tiết kiệm thời gian. Một điểm nổi bật khác của hệ thống là chức năng tìm kiếm chuyến bay phù hợp, với các tùy chọn chi tiết về điểm đến, ngày khởi hành, hạng ghế và loại vé, giúp khách hàng dễ dàng chọn được chuyến bay đáp ứng tốt nhất nhu cầu di chuyển của mình.

2.2.2. Yêu cầu về thiết kế giao diện

Giao diện của hệ thống được thiết kế với tính trực quan và dễ sử dụng làm trọng tâm, đảm bảo rằng khách hàng có thể dễ dàng tương tác với trang web mà không gặp bất kỳ khó khăn nào. Để đạt được điều này, hệ thống sẽ sử dụng hình ảnh, biểu tượng (icon) và bố cục hợp lý để người dùng có thể nhanh chóng tìm kiếm thông tin, truy cập vào các chức năng mà không cần mất nhiều thời gian tìm hiểu. Phần thiết kế cần đặc biệt chú trọng vào việc hiển thị thông tin một cách rõ ràng, dễ đọc và dễ hiểu để tạo cảm giác thân thiện và thoải mái cho khách hàng khi truy cập trang web. Đặc biệt, trang chủ và các trang quan trọng khác nên được

bố trí với một cấu trúc rõ ràng, nhất quán, giúp người dùng dễ dàng điều hướng qua các tính năng.

2.2.3. Tối ưu trải nghiệm người dùng

Hệ thống sẽ được tối ưu hóa để mang lại trải nghiệm người dùng mượt mà nhất, với tốc độ tải trang nhanh, đảm bảo người dùng không phải chờ đợi lâu khi truy cập các thông tin hoặc thực hiện các thao tác. Điều này không chỉ tạo ấn tượng tích cực cho khách hàng mà còn là yếu tố quyết định giúp họ quay lại sử dụng dịch vụ trong tương lai. Hệ thống cần được thiết kế sao cho tương thích với nhiều thiết bị khác nhau, từ máy tính để bàn, máy tính bảng đến điện thoại di động, nhằm phục vụ tối đa nhu cầu của người dùng, cho phép họ truy cập và sử dụng dịch vụ ở bất cứ đâu, bất cứ lúc nào. Tính năng phản hồi linh hoạt (responsive) là điều quan trọng giúp giao diện trang web hiển thị một cách tối ưu trên các kích thước màn hình khác nhau, từ đó mang lại trải nghiệm nhất quán và thuận tiện cho mọi người dùng.

Bằng cách tập trung vào các yếu tố thiết kế thân thiện, dễ sử dụng và tối ưu trải nghiệm người dùng, hệ thống đặt vé và tra cứu thông tin sân bay sẽ trở thành một công cụ hữu ích, đem lại sự thuận tiện và nhanh chóng cho hành khách trong quá trình di chuyển và sử dụng các dịch vụ sân bay.

2.3. Giao diện cho quản trị viên

Hệ thống quản trị sân bay được thiết kế với mục tiêu cung cấp cho quản trị viên một công cụ quản lý toàn diện, giúp họ có thể dễ dàng kiểm soát và điều hành mọi hoạt động diễn ra tại sân bay. Đây sẽ là một nền tảng tập trung giúp quản trị viên giám sát các thông số quan trọng, đưa ra các quyết định kịp thời và tối ưu hóa quy trình quản lý nhằm nâng cao hiệu quả hoạt động của sân bay. Từ việc quản lý nhân sự, theo dõi các chuyến bay, đến điều phối các nguồn lực và thiết bị sân bay, hệ thống này đóng vai trò như một công cụ chiến lược giúp đảm bảo mọi hoạt động diễn ra đồng bộ, hiệu quả và an toàn.

2.3.1. Các chức năng chính của hệ thống quản trị

- **Dashboard:** Đây là trung tâm thông tin chính của hệ thống, cung cấp cho quản trị viên một cái nhìn tổng quan về tình hình hoạt động của sân bay. Dashboard hiển thị các thông số quan trọng dưới dạng biểu đồ trực quan, giúp quản trị viên dễ dàng theo dõi các số liệu như số lượng khách hàng, số lượng chuyến bay, tình trạng đặt chỗ và các chỉ số hiệu suất khác. Những biểu đồ, bảng thống

kê này không chỉ cung cấp thông tin chi tiết mà còn giúp quản trị viên phát hiện kịp thời các xu hướng và biến động trong hoạt động của sân bay.

- **Quản lý thông tin khách hàng:** Chức năng này cho phép quản trị viên lưu trữ và cập nhật thông tin chi tiết của từng khách hàng sử dụng dịch vụ sân bay. Ngoài ra, quản trị viên có thể truy xuất các dữ liệu liên quan đến khách hàng để phục vụ công tác chăm sóc và hỗ trợ khi cần thiết, cũng như phân tích thông tin để xây dựng các chiến lược phục vụ tốt hơn trong tương lai.
- **Quản lý thông tin nhân viên:** Đây là chức năng giúp quản trị viên quản lý thông tin của tất cả nhân viên làm việc tại sân bay, bao gồm thông tin cá nhân, chức vụ, lịch làm việc và phân công công tác. Hệ thống sẽ hỗ trợ quản trị viên theo dõi việc phân bổ nhân sự cho các chuyến bay và nhiệm vụ cụ thể, từ đó đảm bảo mọi nhân viên đều thực hiện đúng công việc và trách nhiệm được giao.
- **Quản lý thông tin máy bay:** Chức năng này lưu trữ các thông tin chi tiết về từng máy bay như số hiệu, loại máy bay, số lượng ghế. Việc quản lý dữ liệu máy bay giúp đảm bảo tất cả phương tiện đều trong tình trạng sẵn sàng phục vụ và đáp ứng tốt nhu cầu khai thác của sân bay. Quản trị viên cũng có thể kiểm tra tình trạng bảo trì và sắp xếp lịch trình hoạt động của các máy bay một cách chính xác.
- **Quản lý thông tin loại máy bay:** Phân loại và lưu trữ thông tin các loại máy bay là một chức năng quan trọng giúp hệ thống có thể phân biệt các loại máy bay, nhà sản xuất, và các đặc điểm kỹ thuật khác nhau. Điều này giúp quản trị viên kiểm soát được các phương tiện có mặt tại sân bay, dễ dàng phân bổ máy bay phù hợp với từng chuyến bay và nhu cầu cụ thể.
- **Quản lý thông tin chuyến bay:** Chức năng này cho phép quản trị viên theo dõi và quản lý chi tiết từng chuyến bay, bao gồm mã chuyến bay, lịch trình, thông tin khởi hành và điểm đến. Quản trị viên có thể kiểm tra tình trạng của các chuyến bay, từ đó đưa ra các điều chỉnh kịp thời để đảm bảo sự chính xác và an toàn trong điều phối.
- **Quản lý thông tin lịch bay:** Lịch bay là một phần quan trọng trong hệ thống, bao gồm ngày đi/đến, mã chuyến và loại máy bay sử dụng. Quản trị viên có thể thiết lập và quản lý các lịch trình bay cụ thể, phối hợp với nhân viên và phi hành đoàn để đảm bảo các chuyến bay diễn ra đúng kế hoạch và tránh xung đột lịch trình.
- **Quản lý thông tin đặt chỗ:** Chức năng này giúp quản trị viên quản lý các thông tin đặt chỗ của khách hàng, liên kết với mã chuyến bay và các yêu cầu đặc biệt. Điều này không chỉ hỗ trợ quá trình check-in trước chuyến bay mà

còn cung cấp dữ liệu về nhu cầu của khách hàng, giúp cải thiện chất lượng dịch vụ khách hàng và tăng tính cá nhân hóa trong chăm sóc khách hàng.

- **Quản lý thông tin phân công:** Đây là công cụ giúp quản trị viên dễ dàng phân công nhiệm vụ cho nhân viên theo lịch trình các chuyến bay và yêu cầu công việc cụ thể. Chức năng này liên kết với dữ liệu nhân viên và chuyến bay để quản lý sự phân bổ nhân sự, đảm bảo mọi nhiệm vụ đều được thực hiện đúng người, đúng việc, và đúng thời gian.

2.3.2. Yêu cầu về thiết kế giao diện quản trị

Giao diện của hệ thống quản trị cần phải khoa học và dễ dàng quản lý. Các khu vực chức năng phải được phân chia rõ ràng, giúp quản trị viên dễ dàng truy cập các thông tin và thực hiện các thao tác mà không gặp khó khăn. Bảng biểu và biểu đồ trực quan sẽ là các công cụ chính trong việc hiển thị và báo cáo dữ liệu, giúp quản trị viên có thể nhanh chóng nắm bắt tình hình và đưa ra các quyết định chính xác. Việc sắp xếp hợp lý và có cấu trúc sẽ giúp giao diện trở nên rõ ràng và dễ điều hướng, từ đó tăng hiệu quả công việc.

2.3.3. Yêu cầu về bảo mật thông tin

Hệ thống quản trị cần đảm bảo tính bảo mật cao để ngăn chặn mọi hành vi truy cập trái phép. Các biện pháp bảo mật như xác thực nhiều yếu tố (multi-factor authentication), mã hóa dữ liệu nhạy cảm, và kiểm soát truy cập theo vai trò sẽ được áp dụng nhằm bảo vệ thông tin quản trị viên, dữ liệu khách hàng, và các dữ liệu quan trọng khác. Điều này không chỉ bảo vệ quyền riêng tư của hành khách và nhân viên mà còn duy trì tính toàn vẹn và an toàn cho toàn bộ hệ thống.

Với những tính năng và yêu cầu thiết kế nêu trên, hệ thống quản trị sân bay sẽ trở thành công cụ đắc lực cho quản trị viên trong việc giám sát, điều phối, và tối ưu hóa các hoạt động của sân bay, góp phần nâng cao chất lượng và hiệu quả vận hành của toàn hệ thống.

CHƯƠNG 5. BACK-END

1. Chức năng hệ thống

1.1. Admin Controller

1.1.1. Thuộc tính và cấu hình cơ bản với JdbcTemplate

JdbcTemplate là một thành phần quan trọng trong Spring Boot để giúp quản lý kết nối và tương tác với cơ sở dữ liệu một cách dễ dàng. Trong lớp **AdminController**, JdbcTemplate được khai báo với @Autowired để Spring có thể tự động khởi tạo và cung cấp đối tượng này. Khi có đối tượng jdbcTemplate, chúng ta có thể dễ dàng thực hiện các truy vấn SQL, chẳng hạn như lấy danh sách chuyến bay, máy bay, hoặc các thống kê. JdbcTemplate giúp giảm thiểu các thao tác xử lý kết nối thủ công, giúp mã dễ đọc và bảo trì hơn.

1.1.2. Trang admin (*/admin*)

Phương thức **admin(Model model, HttpSession session)** xử lý yêu cầu đến URL */admin*. Khi một người dùng truy cập vào trang quản trị, phương thức này sẽ kiểm tra xem người dùng có đang đăng nhập hay không thông qua session. Nếu session không chứa username, tức là người dùng chưa đăng nhập, thì phương thức sẽ chuyển hướng người dùng đến trang */login*. Ngược lại, nếu người dùng đã đăng nhập, phương thức sẽ gọi **getStats()** để lấy dữ liệu thống kê từ cơ sở dữ liệu và thêm dữ liệu này vào model. Dữ liệu thống kê này sẽ được hiển thị trên giao diện của trang quản trị, cung cấp cho người quản trị cái nhìn tổng quan về hệ thống.

1.1.3. API lấy thông tin chuyến bay (*/api/admin/flights*)

Phương thức **getFlights()** là một API giúp trả về danh sách chuyến bay hiện có trong hệ thống. Khi được gọi, phương thức này sẽ thực hiện truy vấn để lấy thông tin bao gồm mã chuyến bay, sân bay đi, sân bay đến, giờ đi, và giờ đến của từng chuyến. Một điểm đặc biệt trong phương thức này là khả năng tạo mã chuyến bay tiếp theo. Cụ thể, **getFlights()** sẽ tìm mã chuyến bay lớn nhất hiện tại, tăng thêm một đơn vị, rồi định dạng theo mẫu "CB000001". Tính năng này giúp đảm bảo rằng các mã chuyến bay luôn là duy nhất và tuân theo một định dạng thống nhất, hỗ trợ dễ dàng trong việc quản lý.

1.1.4. API lấy thông tin máy bay (*/api/admin/aircraft*)

Phương thức **getAircraft()** phục vụ cho việc lấy thông tin về các loại máy bay và máy bay hiện có. Phương thức này trả về danh sách bao gồm mã loại máy bay, hãng sản xuất, số hiệu máy bay, và số ghế ngồi của từng máy bay. Ngoài ra,

phương thức cũng hỗ trợ tạo mã loại máy bay mới bằng cách tìm mã lớn nhất hiện tại, sau đó tăng thêm một đơn vị và định dạng theo mẫu "01". Điều này giúp duy trì sự nhất quán trong mã loại máy bay, giúp hệ thống dễ dàng theo dõi và quản lý các loại máy bay khác nhau theo chuẩn định dạng.

1.1.5. API lấy thông tin đặt chỗ và khách hàng (*/api/admin/bookings*)

Phương thức **getBookings()** lấy danh sách đặt chỗ và thông tin khách hàng, bao gồm các chi tiết về chuyến bay mà khách hàng đã đặt và thông tin liên hệ của khách hàng. Phương thức này thực hiện truy vấn để lấy các thông tin cần thiết, sau đó trả về một danh sách dữ liệu chi tiết về các đặt chỗ đã có. Đặc biệt, **getBookings()** còn có tính năng tạo mã khách hàng mới theo mẫu "KH000001" bằng cách lấy mã khách hàng lớn nhất hiện tại và tăng thêm một đơn vị. Việc này giúp đảm bảo rằng mỗi khách hàng có một mã định danh duy nhất và dễ dàng truy vết.

1.1.6. API chi tiết chuyến bay (*/api/admin/flight-details*)

Phương thức **getFlightDetails(@RequestParam String flight_id)** cung cấp chi tiết giờ đi và giờ đến của một chuyến bay cụ thể. Người dùng sẽ truyền flight_id qua tham số yêu cầu, và phương thức sẽ truy vấn để trả về giờ đi và giờ đến của chuyến bay tương ứng. API này hữu ích trong việc cung cấp thông tin chi tiết cho người quản lý khi cần kiểm tra các thời gian của chuyến bay mà không cần xem toàn bộ danh sách chuyến bay.

1.1.7. API lấy thông tin nhân viên và phân công (*/api/admin/employees*)

Phương thức **getEmployees()** giúp lấy danh sách nhân viên và thông tin về các chuyến bay mà họ được phân công. Phương thức này trả về các chi tiết như mã nhân viên, tên, thông tin liên hệ, loại nhân viên, và mã chuyến bay phân công. Ngoài ra, phương thức còn hỗ trợ tự động tạo mã nhân viên mới, bằng cách tìm mã nhân viên lớn nhất hiện tại và định dạng theo mẫu "NV000001". Tính năng này giúp quản lý nhân viên hiệu quả hơn, đảm bảo mỗi nhân viên có mã định danh duy nhất và phù hợp với quy tắc đặt tên trong hệ thống.

1.1.8. API lấy thống kê (*/api/admin/stats*)

Phương thức **getStats()** cung cấp số liệu thống kê tổng quan về hệ thống, bao gồm số lượng khách hàng, nhân viên, loại máy bay, máy bay, chuyến bay, lịch bay, đặt chỗ, và phân công. Ngoài ra, phương thức này còn bao gồm các thống kê chi tiết như:

- **Thống kê loại máy bay**: Đếm số lượng từng hãng sản xuất máy bay, giúp người quản lý nắm rõ hãng sản xuất nào được sử dụng nhiều nhất.
- **Top chuyến bay**: Xác định top 5 chuyến bay có nhiều lượt đặt nhất, cho thấy chuyến bay nào phổ biến và cần được ưu tiên.
- **Nhân viên theo loại**: Đếm số lượng nhân viên theo từng loại (ví dụ: phi công, tiếp viên), giúp quản lý nhân sự dễ dàng hơn.
- **Chuyến bay theo tháng**: Cung cấp số lượng chuyến bay trong từng tháng của năm hiện tại, giúp phân tích xu hướng và điều chỉnh lịch bay nếu cần thiết.

1.1.9. API lấy thông tin lịch bay (*/api/admin/schedules*)

Phương thức **getSchedules()** trả về chi tiết về lịch bay, bao gồm các chuyến bay đã lên lịch và thông tin về máy bay sẽ phục vụ các chuyến bay đó. Thông tin bao gồm mã chuyến bay, sân bay đi, sân bay đến, giờ đi, giờ đến, loại máy bay và số ghế ngồi. Phương thức này giúp người quản lý dễ dàng kiểm tra lịch trình cụ thể của từng chuyến bay, cũng như thông tin về máy bay sẽ được sử dụng, đảm bảo sự sắp xếp và điều phối hợp lý.

1.2. Sân bay Controller

1.2.1. Thuộc tính JdbcTemplate

Lớp **SanBayController** sử dụng JdbcTemplate, một công cụ của Spring giúp tương tác với cơ sở dữ liệu dễ dàng và hiệu quả. Bằng cách đánh dấu thuộc tính jdbcTemplate với **@Autowired**, Spring sẽ tự động khởi tạo đối tượng JdbcTemplate, nhờ đó, chúng ta không cần phải thiết lập kết nối cơ sở dữ liệu thủ công mỗi khi truy vấn. JdbcTemplate là thành phần trung tâm của các phương thức trong lớp này, giúp đơn giản hóa các thao tác truy vấn SQL như lấy danh sách chuyến bay và tìm kiếm chuyến bay theo tiêu chí người dùng nhập vào.

1.2.2. Trang hiển thị thông tin sân bay (*/san_bay*)

Phương thức **sanBay()** được ánh xạ với đường dẫn */san_bay* và trả về tên của trang hiển thị "san_bay". Khi người dùng truy cập vào đường dẫn */san_bay*, Spring sẽ tìm kiếm và hiển thị trang với tên "san_bay" (có thể là một file HTML hoặc template khác). Phương thức này đơn giản chỉ phục vụ cho mục đích điều hướng, giúp người dùng mở trang hiển thị thông tin liên quan đến sân bay mà không cần tải thêm dữ liệu từ cơ sở dữ liệu.

1.2.3. API lấy danh sách chuyến bay (*/api/flights*)

Phương thức **getFlights()** là một API dùng để lấy danh sách các chuyến bay có trong hệ thống. Trong phương thức này, truy vấn SQL được thực thi để

lấy các thông tin cần thiết như mã chuyến bay (MaChuyenBay), số hiệu máy bay (SoHieuMayBay), loại máy bay (MaLoaiMayBay), sân bay đi (TenSanBayDi), sân bay đến (TenSanBayDen), giờ đi (GioDi), và giờ đến (GioDen). Truy vấn sử dụng LEFT JOIN để đảm bảo rằng tất cả các chuyến bay trong bảng ChuyenBay đều được lấy ra, dù có hoặc không có thông tin về máy bay và lịch bay liên kết. Kết quả sau đó được đưa vào một danh sách và trả về dưới dạng JSON để hiển thị cho người dùng hoặc sử dụng cho các ứng dụng khác.
Nếu có bất kỳ lỗi nào xảy ra trong quá trình thực thi truy vấn, phương thức sẽ bắt ngoại lệ, ghi lại lỗi và trả về một phản hồi với thông tin lỗi. Điều này giúp người dùng biết rằng có vấn đề xảy ra và có thể liên hệ hỗ trợ.

1.2.4. API tìm kiếm chuyến bay (*/api/search*)

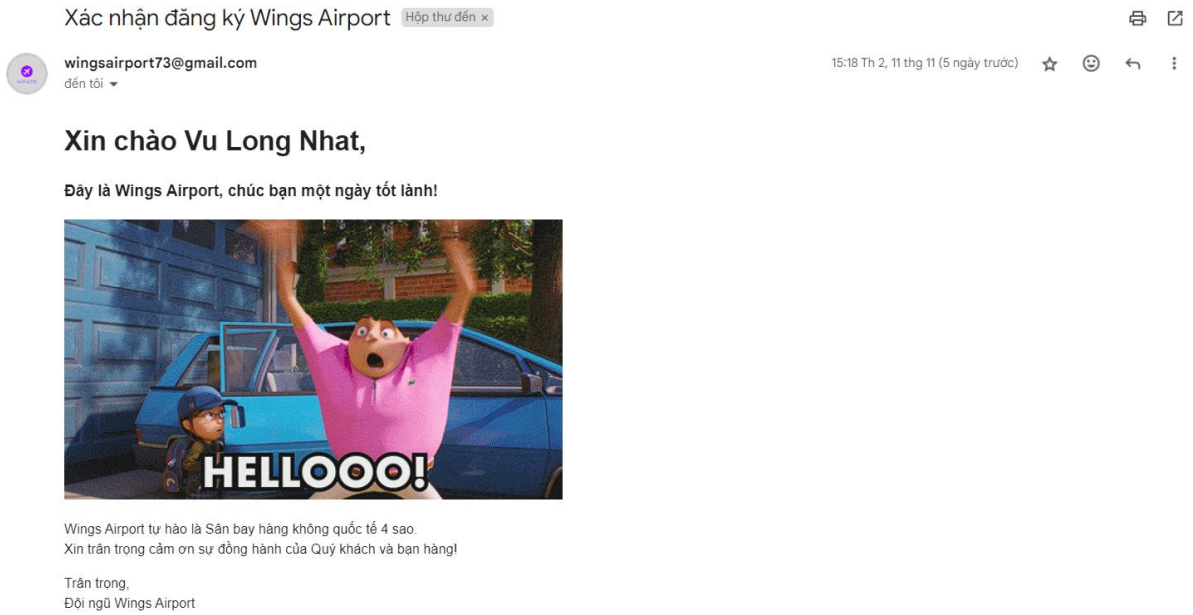
Phương thức **searchFlights()** là một API cho phép người dùng tìm kiếm chuyến bay dựa trên từ khóa nhập vào. Người dùng có thể tìm kiếm theo mã chuyến bay, số hiệu máy bay, loại máy bay, sân bay đi, sân bay đến, giờ đi, và giờ đến. Phương thức nhận một tham số query, đại diện cho từ khóa tìm kiếm. Từ khóa này sẽ được sử dụng để xây dựng mẫu tìm kiếm (searchPattern) với ký tự %, cho phép tìm kiếm không chính xác (fuzzy search) theo nhiều trường.
Trong câu truy vấn SQL, LIKE ? được sử dụng với các trường khác nhau để đối chiếu các giá trị chứa mẫu tìm kiếm. Sau khi lấy kết quả từ cơ sở dữ liệu, phương thức sẽ đóng gói dữ liệu trong một đối tượng Map và trả về dưới dạng JSON. Điều này cho phép các ứng dụng khách, chẳng hạn như frontend hoặc ứng dụng di động, có thể nhận và hiển thị kết quả tìm kiếm một cách dễ dàng. Nếu có lỗi trong quá trình truy vấn, phương thức sẽ trả về phản hồi lỗi với mã trạng thái nội bộ (500), kèm theo thông báo lỗi để người dùng hoặc nhà phát triển biết vấn đề cụ thể.

1.3. Xử lý đăng nhập

Trong ứng dụng quản lý sân bay, hai lớp **MainController** và **AuthController** phối hợp với nhau để xử lý các chức năng xác thực người dùng và điều hướng giao diện dựa trên trạng thái đăng nhập. Khi người dùng truy cập trang chính (/), **MainController** sẽ kiểm tra phiên đăng nhập hiện tại bằng cách kiểm tra sự tồn tại của thuộc tính username trong HttpSession. Nếu người dùng đã đăng nhập, họ sẽ được chuyển hướng tới trang quản trị (/*admin*); nếu không, họ sẽ vào trang san_bay, nơi hiển thị nội dung cho người dùng chưa đăng nhập. Người dùng muốn đăng nhập có thể truy cập đường dẫn */login* để xem trang đăng nhập.

Quá trình xác thực diễn ra trong **AuthController**, nơi nhận thông tin username và password từ người dùng, sau đó băm mật khẩu với thuật toán SHA-256 trước khi so sánh với mật khẩu trong cơ sở dữ liệu. Nếu thông tin đăng nhập hợp lệ, username sẽ được lưu trong session để xác nhận trạng thái đăng nhập thành công; nếu không, người dùng sẽ nhận được thông báo lỗi. Ngoài ra, **AuthController** cung cấp phương thức **logout()** để xóa phiên đăng nhập khi người dùng muốn đăng xuất, sau đó chuyển hướng về trang san_bay. Bằng cách này, hai lớp **MainController** và **AuthController** hỗ trợ xác thực, bảo mật và điều hướng một cách hiệu quả, tạo ra trải nghiệm đăng nhập liền mạch cho người dùng.

1.4. Chức năng gửi email tự động



Chương5_hình 1.4-1. Email tự động

Trong hệ thống quản lý sân bay, lớp **EmailController** cung cấp một API để xử lý việc gửi email thông qua một yêu cầu POST tới endpoint */send-email*. Khi người dùng gửi một yêu cầu, controller này sẽ tiếp nhận dữ liệu dưới dạng một đối tượng JSON, được ánh xạ thành đối tượng **EmailRequest**. Đối tượng này chứa ba trường quan trọng: email (địa chỉ email người nhận), language (ngôn ngữ mà người nhận muốn sử dụng cho thông báo email), và fullName (tên đầy đủ của người nhận).

Sau khi nhận yêu cầu từ client, **EmailController** sẽ gọi đến lớp **EmailService** để thực hiện việc gửi email. **EmailService** sử dụng thông tin từ **EmailRequest** để xây dựng và gửi email tới địa chỉ đã chỉ định. Việc gửi email có thể bao gồm các bước như lựa chọn template email phù hợp với ngôn ngữ của

người nhận (dựa vào trường language) và cá nhân hóa thông điệp bằng tên người nhận (fullName).

Nếu quá trình gửi email thành công, lớp **EmailController** sẽ phản hồi lại client một đối tượng **EmailResponse** với thuộc tính success được đặt là true, cho biết yêu cầu đã được thực hiện thành công. Trong trường hợp có lỗi trong quá trình gửi email, chẳng hạn như lỗi kết nối mạng hoặc lỗi server, **EmailController** sẽ bắt và xử lý ngoại lệ, trả về một đối tượng **EmailResponse** với thuộc tính success là false, đồng thời cung cấp thông tin chi tiết về lỗi qua thuộc tính error để người dùng có thể dễ dàng nắm bắt nguyên nhân vấn đề.

Đối tượng **EmailResponse** là một phần quan trọng trong việc chuẩn hóa phản hồi từ API. Nó giúp giao tiếp giữa hệ thống và client trở nên dễ dàng và rõ ràng hơn, đặc biệt khi client cần phải xử lý phản hồi từ server, xác nhận trạng thái yêu cầu và hiển thị thông báo thích hợp cho người dùng.

Bên cạnh đó, toàn bộ API được cấu hình với chú thích **@CrossOrigin(origins = "*"),** cho phép các yêu cầu từ các nguồn khác nhau (cross-origin requests) được phép truy cập endpoint này, nhằm đảm bảo tính linh hoạt và khả năng tương tác của hệ thống với các ứng dụng hoặc dịch vụ bên ngoài.

2. Chức năng cho khách hàng(9)

2.1. Đăng ký khách hàng

CustomerController: là controller chuyên quản lý các thao tác liên quan đến khách hàng trong hệ thống. Các chức năng của controller này bao gồm:

Thêm khách hàng (*/them_kh*):

- Chức năng này cho phép người dùng thêm một khách hàng mới vào hệ thống. Các thông tin cần thiết như mã khách hàng, số điện thoại, họ tên, và địa chỉ sẽ được truyền qua các tham số HTTP POST. Sau khi thông tin được nhận, hệ thống sẽ gọi đến service **KhachHangService** để thực hiện việc lưu trữ vào cơ sở dữ liệu.
- Phương thức này yêu cầu người dùng phải đăng nhập trước, vì vậy có thể bảo vệ các thao tác quan trọng này thông qua annotation **@LoginRequired**.

Lấy ID khách hàng tiếp theo *(/next_customer_id*):

- Chức năng này phục vụ việc sinh tự động ID cho khách hàng mới. Khi người dùng muốn thêm khách hàng mới mà không cần phải tự xác định ID, hệ thống sẽ trả về ID tiếp theo có thể sử dụng.
- Phương thức này không yêu cầu quyền truy cập đặc biệt, và trả về một đối tượng JSON chứa ID mới qua **ResponseEntity**.

Thêm khách hàng cho frontend (*/them_kh_fe*):

- Đây là một phương thức bổ sung dành riêng cho các ứng dụng frontend. Nó nhận dữ liệu khách hàng từ frontend và gọi service **KhachHangService** để lưu trữ thông tin. Phương thức này trả về kết quả trực tiếp từ service, giúp frontend có thể xử lý phản hồi một cách dễ dàng.
- Phương thức này sử dụng **@ResponseBody** để trả về kết quả dưới dạng JSON.

Cập nhật thông tin khách hàng (*/sua_kh*):

- Chức năng này cho phép người dùng sửa đổi thông tin của khách hàng đã có. Các tham số đầu vào (như mã khách hàng và các trường thông tin khách hàng cần thay đổi) được truyền qua HTTP POST. Sau đó, thông tin này sẽ được truyền đến **KhachHangService** để thực hiện việc cập nhật trong cơ sở dữ liệu.

Xóa khách hàng (*/xoa_kh/{customerId}*):

- Đây là chức năng cho phép xóa thông tin của một khách hàng từ hệ thống thông qua mã khách hàng (customerId). Phương thức này được bảo vệ bằng annotation **@LoginRequired**, chỉ người dùng đã đăng nhập mới có thể thực hiện thao tác này.
- Sau khi nhận được yêu cầu xóa, controller sẽ gọi **KhachHangService** để xóa khách hàng từ cơ sở dữ liệu.

Model **KhachHang** đại diện cho một khách hàng trong hệ thống, bao gồm các thông tin cần thiết để nhận diện và liên lạc với từng khách hàng. Các thuộc tính của model này bao gồm maKH (Mã Khách Hàng) – định danh duy nhất cho mỗi khách hàng, sdt (Số Điện Thoại) – để khách hàng có thể liên lạc dễ dàng, hoDem (Họ Đệm) và ten (Tên) của khách hàng để xác định danh tính, và diaChi (Địa Chỉ) để lưu trữ nơi cư trú của khách hàng. Những thông tin này không chỉ cho phép hệ thống quản lý thông tin cá nhân mà còn giúp xác định khách hàng một cách chính xác khi cần thiết.

Đối với khách hàng, model **KhachHang** cho phép người dùng xem và cập nhật thông tin cá nhân của họ. Người dùng có thể chỉnh sửa các thông tin như số điện thoại, địa chỉ để đảm bảo dữ liệu luôn chính xác và mới nhất. Điều này giúp khách hàng duy trì thông tin cá nhân của họ trong hệ thống, tạo điều kiện thuận lợi cho việc đặt chỗ và nhận các thông báo liên quan đến chuyến bay hoặc các dịch vụ khác.

KhachHangRepository chịu trách nhiệm quản lý và truy xuất dữ liệu liên quan đến khách hàng trong hệ thống. Repository này cung cấp các phương thức giúp quản trị viên quản lý hiệu quả danh sách khách hàng, bao gồm khả năng truy xuất, cập nhật, và xóa thông tin khách hàng khi cần thiết.

Phương thức **findMaxMaKH**() có nhiệm vụ tìm mã khách hàng lớn nhất hiện có trong hệ thống. Phương thức này đặc biệt hữu ích khi cần tạo mã khách hàng mới một cách tự động và theo thứ tự tăng dần. Việc sử dụng phương thức này giúp đảm bảo tính duy nhất cho mã khách hàng mới, tránh trùng lặp mã và đảm bảo tính nhất quán trong hệ thống mã hóa.

Ứng dụng: Khi tạo một khách hàng mới, hệ thống có thể sử dụng mã khách hàng lớn nhất hiện có và tăng lên một giá trị để tạo mã mới. Điều này giúp quản trị viên dễ dàng thêm khách hàng mới vào hệ thống mà không phải lo ngại về việc trùng lặp mã khách hàng.

KhachHangRepository chịu trách nhiệm quản lý và truy xuất dữ liệu liên quan đến khách hàng trong hệ thống. Repository này cung cấp các phương thức giúp quản trị viên quản lý hiệu quả danh sách khách hàng, bao gồm khả năng truy xuất, cập nhật, và xóa thông tin khách hàng khi cần thiết.

Ứng dụng: Khi tạo một khách hàng mới, hệ thống có thể sử dụng mã khách hàng lớn nhất hiện có và tăng lên một giá trị để tạo mã mới. Điều này giúp quản trị viên dễ dàng thêm khách hàng mới vào hệ thống mà không phải lo ngại về việc trùng lặp mã khách hàng.

2.2. Đặt chỗ

BookingController là controller quản lý các thao tác liên quan đến đặt chỗ chuyến bay của khách hàng. Các chức năng của controller này bao gồm:

Thêm đặt chỗ *(/them_dat_cho)*:

- Chức năng này cho phép khách hàng thực hiện việc đặt chỗ cho chuyến bay. Các thông tin cần thiết bao gồm mã khách hàng (customer-id), mã chuyến bay (flight-id), và thời gian khởi hành (departure-datetime) sẽ được truyền vào dưới dạng tham số HTTP POST. Thông tin này sau đó được chuyển đến **DatChoService**, nơi thực hiện việc xử lý và lưu trữ vào cơ sở dữ liệu.
- Phương thức này yêu cầu người dùng phải đăng nhập để bảo vệ các thao tác đặt chỗ quan trọng.

Thêm đặt chỗ cho frontend *(/them_dat_cho_fe)*:

- Phương thức này tương tự như phương thức thêm đặt chỗ, nhưng thêm một tham số nữa là email khách hàng (customer-email). Chức năng này có thể dùng cho frontend để dễ dàng tương tác với API khi cần xử lý thông tin đặt chỗ từ giao diện người dùng.

- Phản hồi sẽ được trả về dưới dạng JSON để frontend có thể dễ dàng xử lý kết quả.

Cập nhật đặt chỗ (*/sua_dat_cho*):

- Chức năng này cho phép người dùng thay đổi thông tin đặt chỗ đã thực hiện trước đó. Các tham số cần thiết như mã khách hàng, mã chuyến bay mới, và thời gian khởi hành mới sẽ được truyền vào và xử lý thông qua **DatChoService**. Sau khi cập nhật, thông tin mới sẽ được lưu trữ lại vào cơ sở dữ liệu.

Xóa đặt chỗ (*/xoa_dat_cho*):

- Chức năng này cho phép xóa một đặt chỗ đã được tạo trước đó. Người dùng sẽ cung cấp mã khách hàng, mã chuyến bay, và ngày khởi hành để hệ thống thực hiện thao tác xóa.
- Các lỗi xảy ra trong quá trình xử lý sẽ được bắt và trả về dưới dạng thông báo lỗi chi tiết, giúp người dùng nhận diện vấn đề và có thể xử lý.

Tóm lại, **BookingController** xử lý các thao tác liên quan đến việc quản lý đặt chỗ chuyến bay, bao gồm việc tạo mới, sửa đổi, và hủy bỏ đặt chỗ. Các phương thức này đều có cơ chế bảo mật và xử lý lỗi hợp lý để đảm bảo các thao tác này diễn ra một cách suôn sẻ và an toàn.

Model **DatCho** đại diện cho một đặt chỗ hoặc một vé mà khách hàng đặt cho các chuyến bay cụ thể. Các thuộc tính chính của model này gồm maKH (Mã Khách Hàng) – liên kết một đặt chỗ với khách hàng cụ thể, maChuyenBay (Mã Chuyến Bay) – định danh chuyến bay mà khách hàng đã đặt, và ngayDi (Ngày Đi) – chỉ ra ngày dự kiến khởi hành của chuyến bay. Các thông tin này giúp hệ thống tổ chức và quản lý các lịch trình đặt chỗ của khách hàng, đảm bảo rằng mỗi đặt chỗ là duy nhất và được quản lý đúng cách.

Đối với khách hàng, model **DatCho** cung cấp chức năng tạo mới một đặt chỗ bằng cách chọn chuyến bay và ngày khởi hành mong muốn. Khách hàng có thể xem và quản lý các đặt chỗ của mình, bao gồm xem chi tiết các chuyến bay sắp tới, hủy đặt chỗ khi cần thiết hoặc khi hệ thống cho phép. Những thao tác này giúp khách hàng kiểm soát lịch trình di chuyển của mình, mang lại sự linh hoạt và thuận tiện khi có nhu cầu thay đổi hoặc hủy bỏ chuyến đi.

DatChoRepository chịu trách nhiệm quản lý và truy xuất dữ liệu liên quan đến đặt chỗ của khách hàng. Repository này hỗ trợ việc kiểm tra, thêm, sửa, xóa các thông tin đặt chỗ, giúp quản lý lịch trình của khách hàng và đảm bảo tính chính xác của hệ thống đặt chỗ.

Phương thức **existsByMaKHAndMaChuyenBayAndNgayDi**(...): Phương thức này kiểm tra xem một khách hàng đã có đặt chỗ cho chuyến bay cụ thể vào ngày nhất định chưa. Phương thức này rất hữu ích để tránh tình trạng khách hàng đặt trùng một chuyến bay vào cùng ngày, đảm bảo mỗi chuyến bay chỉ có một đặt chỗ duy nhất cho mỗi khách hàng vào ngày đó.

Ứng dụng: Phương thức này giúp hệ thống kiểm soát và ngăn ngừa việc tạo các đặt chỗ trùng lặp cho cùng một chuyến bay và ngày khởi hành. Điều này giúp giảm thiểu các lỗi đặt chỗ không cần thiết và giúp quản trị viên dễ dàng quản lý các lịch trình đặt chỗ.

Phương thức **findBookingDetails**(...) truy vấn chi tiết đặt chỗ của một khách hàng dựa trên mã khách hàng, mã chuyến bay, và ngày khởi hành. Phương thức này trả về một bản ghi chi tiết với các trường thông tin quan trọng như mã khách hàng, họ tên, số điện thoại, địa chỉ, mã chuyến bay, tên sân bay đi và đến, giờ đi và đến, cũng như ngày khởi hành.

Kết quả: Trả về một Map<String, Object> chứa các thông tin:

- MaKH: Mã khách hàng
- HoTen: Họ và tên khách hàng
- SDT: Số điện thoại khách hàng
- DiaChi: Địa chỉ khách hàng
- MaChuyenBay: Mã chuyến bay
- TenSanBayDi và TenSanBayDen: Tên sân bay đi và đến
- GioDi và GioDen: Giờ đi và giờ đến của chuyến bay
- NgayDi: Ngày khởi hành

Ứng dụng: Phương thức này cho phép hệ thống lấy thông tin chi tiết về các đặt chỗ của khách hàng, từ đó hỗ trợ việc quản lý và kiểm tra các thông tin đặt chỗ, hoặc cung cấp thông tin cho khách hàng khi họ cần.

DatChoService là lớp dịch vụ quản lý các thao tác liên quan đến đặt chỗ, bao gồm thêm, cập nhật, xóa đặt chỗ, và gửi email xác nhận. Các chức năng chính bao gồm:

Thêm đặt chỗ (themDatCho):

- Chức năng này thêm một đặt chỗ mới cho khách hàng. Nếu đặt chỗ đã tồn tại (ví dụ: khách hàng đã đặt chỗ cho chuyến bay và ngày cụ thể đó), dịch vụ sẽ trả về thông báo lỗi để tránh trùng lặp đặt chỗ.

Thêm đặt chỗ và gửi email xác nhận (themDatChoFe):

- Thực hiện thêm một đặt chỗ mới cho khách hàng và gửi email xác nhận. Nếu việc gửi email thất bại, hệ thống sẽ trả về thông báo rằng đặt chỗ đã thành công nhưng không thể gửi email xác nhận. Điều này giúp khách hàng biết rằng đặt chỗ đã được xác nhận trong hệ thống, mặc dù email có thể chưa được gửi.

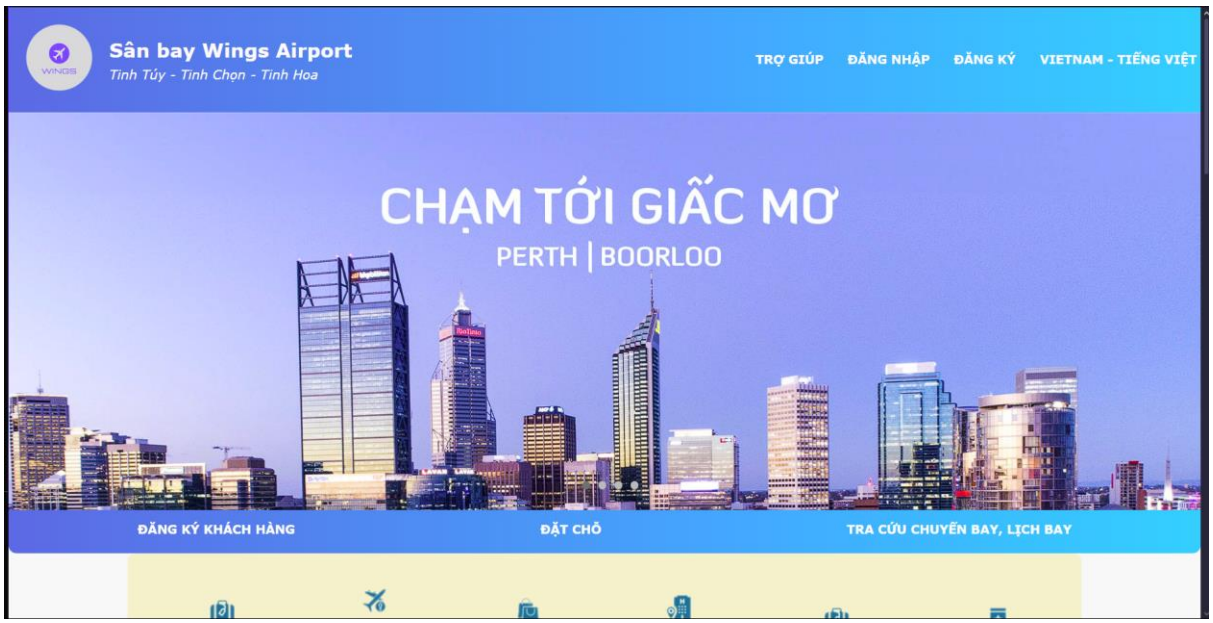
Cập nhật đặt chỗ (suaDatCho):

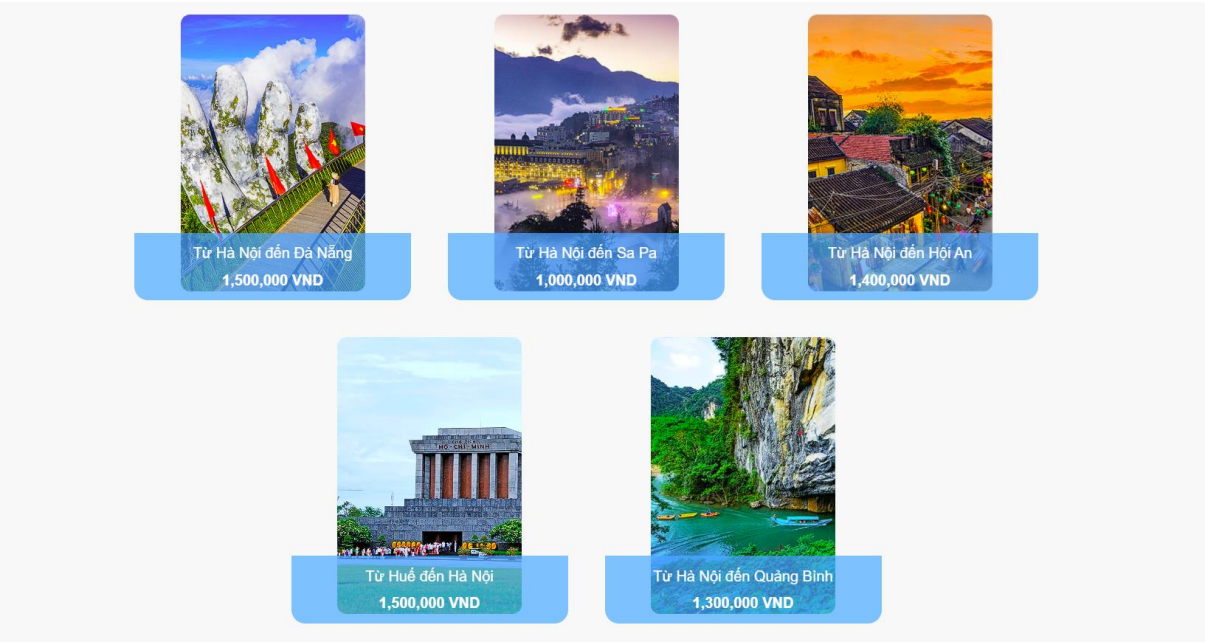
- Cập nhật thông tin của một đặt chỗ hiện có cho khách hàng. Chức năng này giới hạn việc cập nhật khi khách hàng chỉ có một đặt chỗ, giúp tránh việc cập nhật sai thông tin khi khách hàng có nhiều đặt chỗ. Nếu khách hàng có nhiều hơn một đặt chỗ, dịch vụ sẽ trả về thông báo lỗi.

Xóa đặt chỗ (xoaDatCho):

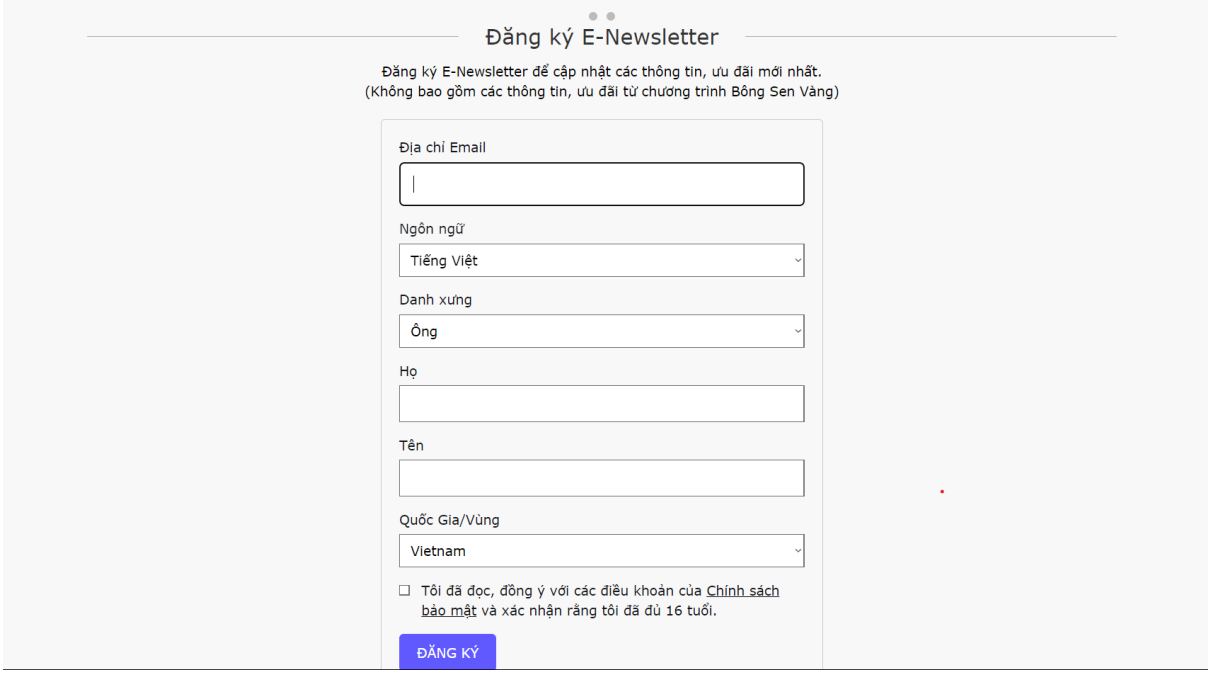
- Xóa đặt chỗ khỏi hệ thống bằng cách sử dụng mã khách hàng, mã chuyến bay và ngày đi. Nếu không tìm thấy đặt chỗ, hệ thống sẽ trả về thông báo lỗi cho người dùng.

2.3. Giao diện cơ bản

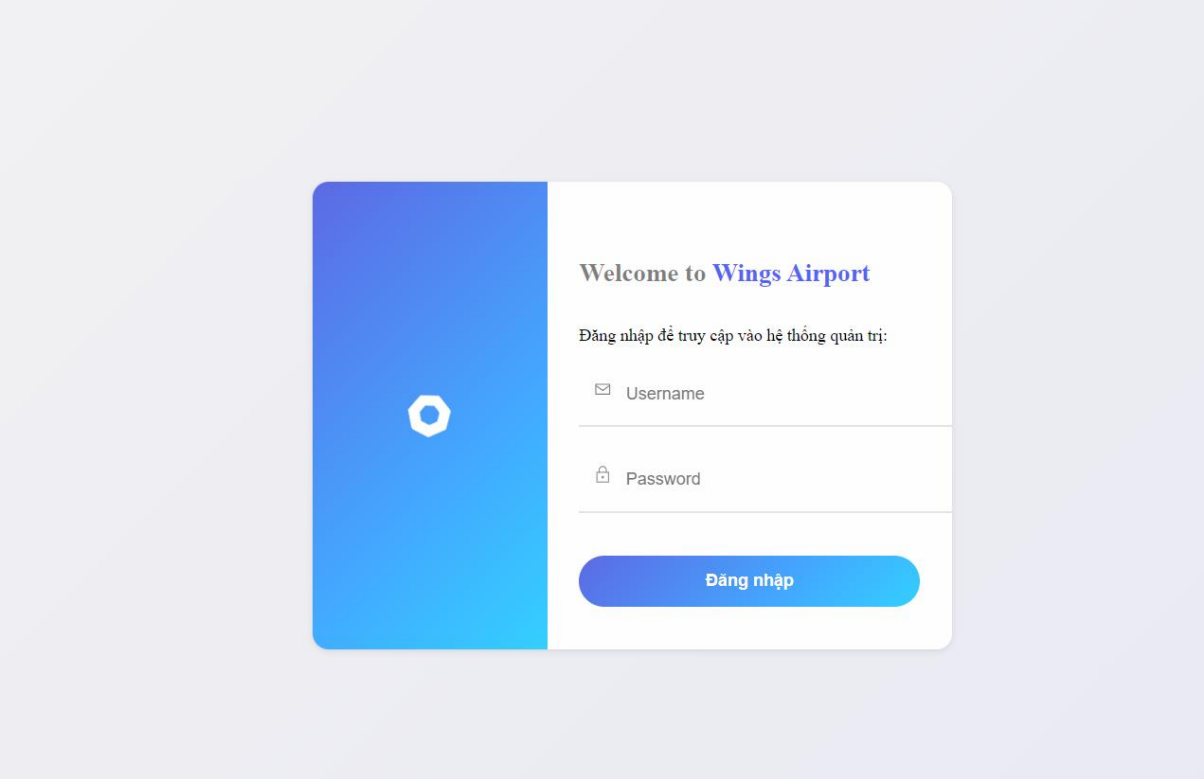




Chương 5_hình 2.3-1,2. Trang chủ



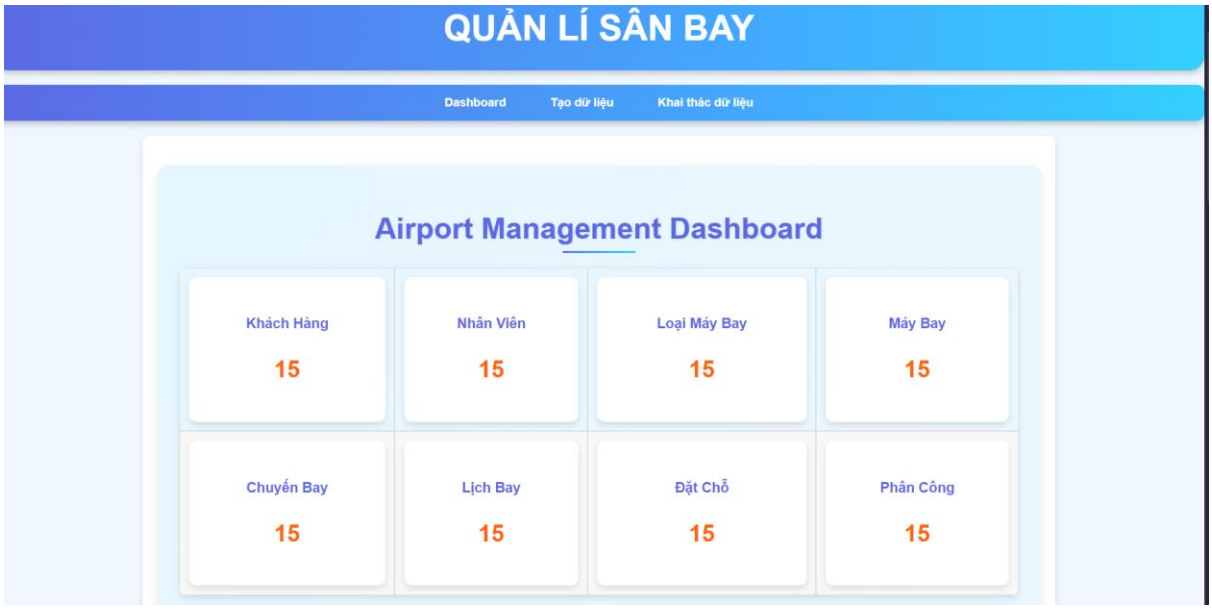
Chương 5_hình 2.3-2. Form đăng ký khách hàng



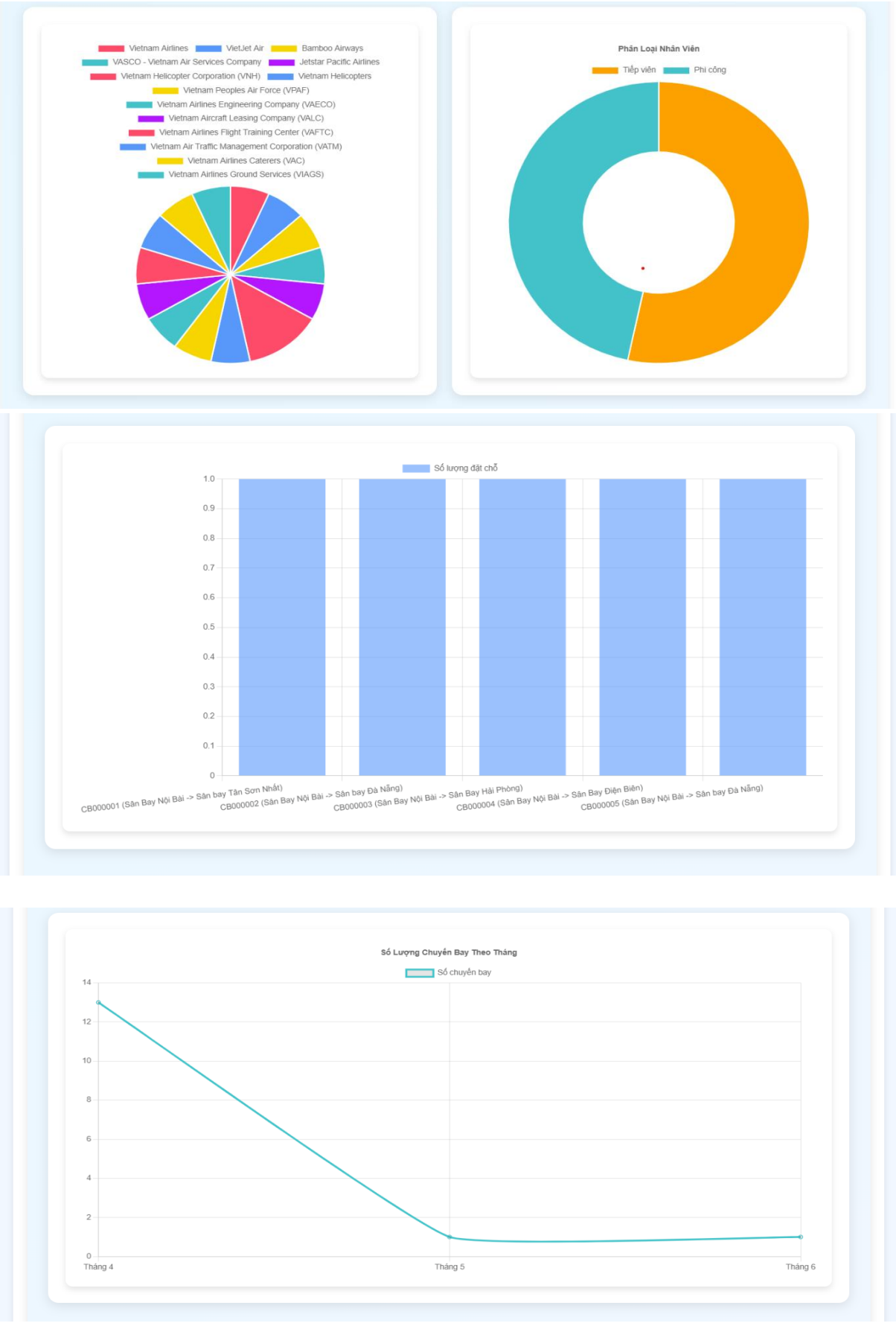
Chương 5_hình 2.3-3. Đăng nhập

3. Chức năng cho quản trị viên

3.1. Giao diện Dashboard



Chương 5_hình 3.1-1. Admin page



Chương 5_hình 3.1-2,3,4. Admin Dashboard

3.2. Giao diện quản lý khách hàng

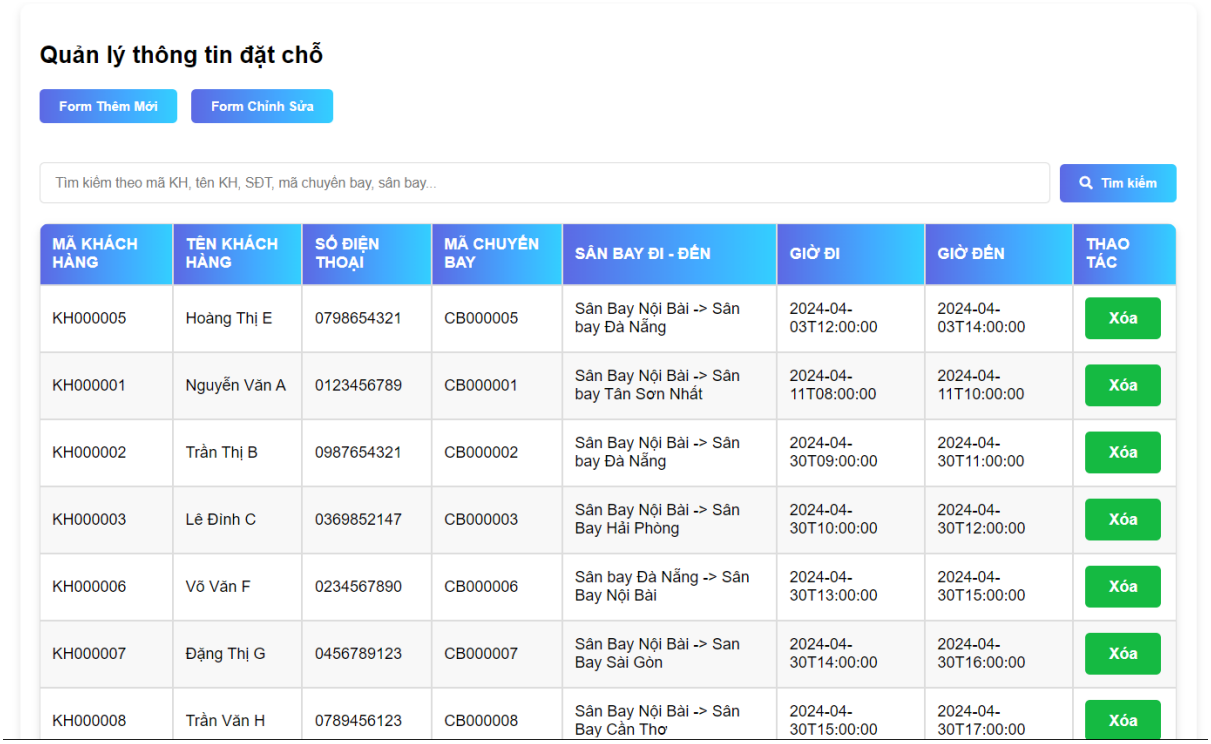
Như đã được phân tích ở trên đối với **quản trị viên**, model quản lý khách cung cấp các chức năng quản lý như xem danh sách các khách hàng đã đăng ký và xem thông tin chi tiết của từng khách hàng. Quản trị viên có thể thực hiện các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) đối với hồ sơ khách hàng, điều này rất quan trọng để duy trì tính chính xác và bảo mật của hệ thống. Ngoài ra, quản trị viên còn có khả năng tìm kiếm khách hàng dựa trên các tiêu chí cụ thể như tên hoặc số điện thoại, giúp họ dễ dàng hỗ trợ khách hàng trong các tình huống như xác minh thông tin hoặc xử lý yêu cầu hỗ trợ.

Về phía Repository được bổ sung thêm phương thức **findMaxMaKH**()có nhiệm vụ tìm mã khách hàng lớn nhất hiện có trong hệ thống. Phương thức này đặc biệt hữu ích khi cần tạo mã khách hàng mới một cách tự động và theo thứ tự tăng dần. Việc sử dụng phương thức này giúp đảm bảo tính duy nhất cho mã khách hàng mới, tránh trùng lặp mã và đảm bảo tính nhất quán trong hệ thống mã hóa. Phương thức này cho phép quản trị viên tạo mã khách hàng tự động khi thêm khách hàng mới vào hệ thống. Bằng cách tạo mã khách hàng theo thứ tự tăng dần, hệ thống giúp duy trì tính nhất quán và đảm bảo mã không trùng lặp, hỗ trợ việc quản lý dễ dàng và chính xác.

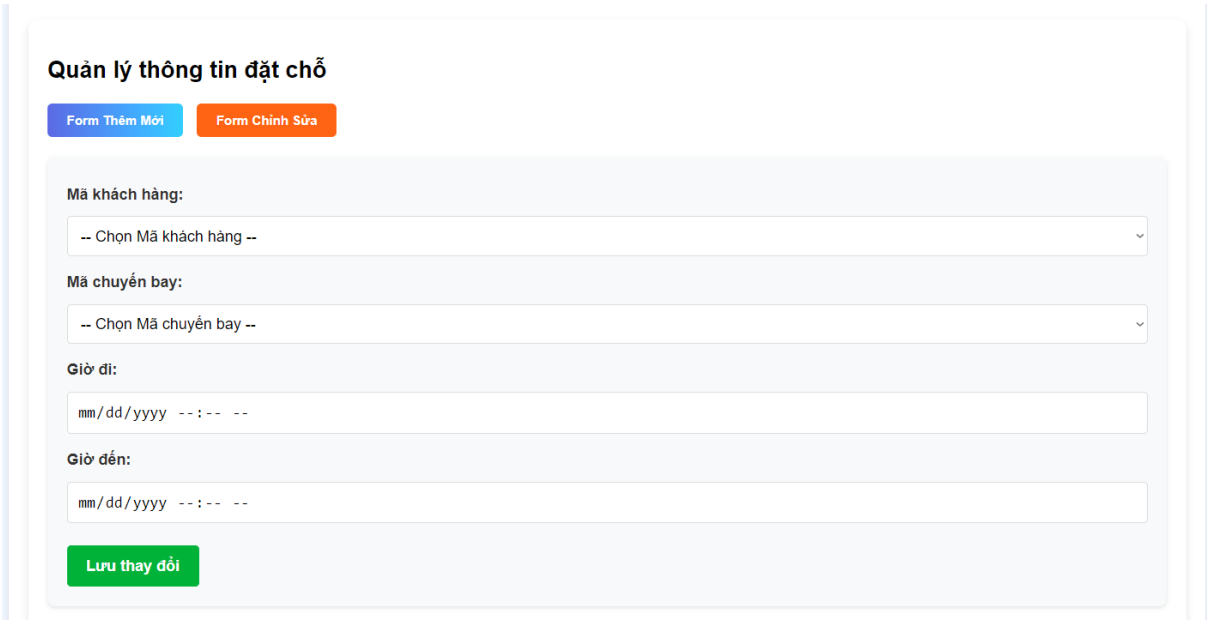
3.3. Giao diện quản lý đặt chỗ

Đối với **quản trị viên**, model **DatCho** cho phép họ xem tất cả các đặt chỗ đã được khách hàng thực hiện. Điều này giúp quản trị viên quản lý lịch trình và các tài nguyên của hệ thống hiệu quả hơn. Quản trị viên có quyền cập nhật hoặc xóa các đặt chỗ khi cần thiết, đặc biệt trong các tình huống như khách hàng yêu cầu hỗ trợ hoặc khi có thay đổi về lịch trình chuyến bay. Ngoài ra, quản trị viên cũng có thể tạo các báo cáo hoặc thực hiện phân tích trên các đặt chỗ dựa trên nhiều yếu tố như chuyến bay, ngày tháng hoặc phân khúc khách hàng cụ thể, giúp tối ưu hóa hoạt động và nâng cao trải nghiệm của khách hàng.

- **Quản lý hồ sơ khách hàng**: Quản trị viên có quyền truy cập đầy đủ vào thông tin chi tiết của các khách hàng, bao gồm khả năng thêm, sửa, xóa và tìm kiếm khách hàng, phục vụ cho nhu cầu quản trị.
- **Quản lý đặt chỗ**: Quản trị viên kiểm soát chi tiết các đặt chỗ, bao gồm các chức năng cập nhật, xóa và phân tích đặt chỗ, giúp hệ thống vận hành trơn tru và đáp ứng tốt yêu cầu của khách hàng.



Chương 5_hình 3.3-1. Thông tin đặt chỗ



Chương 5_hình 3.3-2. Form sửa thông tin đặt chỗ

Repository ở phần đặt chỗ có thêm hương thức **countByMaKH**(...)
Phương thức countByMaKH() đếm tổng số lần đặt chỗ của một khách hàng dựa trên mã khách hàng (MaKH). Phương thức này thường được sử dụng để thống kê số lượng đặt chỗ của một khách hàng, hỗ trợ cho các báo cáo hoặc phân tích hành vi của khách hàng, chẳng hạn như xác định tần suất sử dụng dịch vụ của khách hàng.

Ứng dụng: Phương thức này giúp quản trị viên có cái nhìn tổng quan về tần suất đặt chỗ của từng khách hàng, hỗ trợ cho việc ra quyết định kinh doanh hoặc phân tích hành vi khách hàng.

Chi tiết triển khai đáng chú ý

Hỗ trợ giao dịch (@Transactional):

Cả hai lớp dịch vụ đều sử dụng @Transactional để đảm bảo rằng tất cả các thao tác cơ sở dữ liệu được thực hiện như một giao dịch. Điều này có nghĩa là nếu bất kỳ thao tác nào trong quá trình thêm, cập nhật hoặc xóa bị lỗi, tất cả các thao tác khác trong cùng giao dịch sẽ bị hủy bỏ, giúp bảo toàn tính nhất quán của dữ liệu.

Xử lý lỗi với ResponseEntity:

Các phương thức của dịch vụ trả về đối tượng ResponseEntity để cung cấp các phản hồi chi tiết cho người dùng. ResponseEntity cho phép trả về các mã trạng thái HTTP (như 200 OK, 400 Bad Request, 500 Internal Server Error) cùng với thông báo, giúp người dùng dễ dàng hiểu được kết quả của thao tác.

Ghi log:

Trong DatChoService, các lỗi được ghi lại khi có sự cố xảy ra, chẳng hạn như lỗi khi gửi email hoặc khi xảy ra lỗi trong quá trình đặt chỗ. Điều này rất hữu ích cho việc kiểm tra và gỡ lỗi, giúp quản trị viên hoặc đội ngũ kỹ thuật theo dõi và xử lý các vấn đề kịp thời.

Câu lệnh truy vấn SQL:

DatChoService sử dụng các truy vấn SQL tùy chỉnh để lấy thông tin chi tiết về đặt chỗ. Điều này giúp tăng cường tính linh hoạt và hỗ trợ cho các truy vấn phức tạp mà có thể không được cung cấp bởi các phương thức tiêu chuẩn của JPA. Nhờ đó, hệ thống có thể dễ dàng truy xuất các thông tin cụ thể của đặt chỗ theo yêu cầu của khách hàng hoặc quản trị viên.

3.4. Giao diện quản lý chức năng nhân viên

EmployeeController là controller quản lý các thao tác liên quan đến thông tin nhân viên trong hệ thống. Các chức năng của controller này bao gồm:

Thêm nhân viên (*/them_nv*):

- Chức năng này cho phép người dùng thêm một nhân viên mới vào hệ thống. Các thông tin như mã nhân viên, họ tên, số điện thoại, địa chỉ, lương và loại nhân viên sẽ được truyền qua các tham số HTTP POST. Sau khi nhận được thông tin từ client, controller sẽ gọi đến service NhanVienService để xử lý việc thêm nhân viên vào cơ sở dữ liệu.
- Phương thức này được bảo vệ bằng annotation **@LoginRequired**, yêu cầu người dùng phải đăng nhập trước khi thực hiện thao tác thêm nhân viên.

Cập nhật thông tin nhân viên *(/sua_nv/{employeeId}):*
- Chức năng này cho phép người dùng cập nhật thông tin của một nhân viên đã có trong hệ thống. Thông qua mã nhân viên (employeeId), người dùng có thể thay đổi các thông tin như họ tên, số điện thoại, địa chỉ, lương, và loại nhân viên. Sau khi nhận thông tin từ người dùng, controller sẽ chuyển các dữ liệu này đến NhanVienService để thực hiện cập nhật trong cơ sở dữ liệu.
- Phương thức này cũng yêu cầu người dùng phải đăng nhập thông qua annotation **@LoginRequired**.

Xóa nhân viên *(/xoa_nv/{employeeId}):*
- Chức năng này cho phép người dùng xóa thông tin của một nhân viên khỏi hệ thống. Mã nhân viên sẽ được truyền dưới dạng tham số trong URL. Sau khi nhận mã nhân viên, controller sẽ gọi service **NhanVienService** để thực hiện thao tác xóa dữ liệu liên quan đến nhân viên đó từ cơ sở dữ liệu.
- Cũng giống như các phương thức trước, phương thức này yêu cầu người dùng phải đăng nhập.

EmployeeController cung cấp các chức năng cơ bản cho việc quản lý nhân viên, bao gồm việc thêm mới, sửa đổi và xóa nhân viên. Các phương thức trong controller này đều được bảo vệ bằng annotation **@LoginRequired**, đảm bảo rằng chỉ những người dùng đã đăng nhập mới có quyền thực hiện các thao tác quan trọng này. Mỗi phương thức đều tương tác với service **NhanVienServbice** để xử lý dữ liệu nhân viên trong cơ sở dữ liệu.

Mô hình **NhanVien** đại diện cho một nhân viên trong hệ thống quản lý sân bay. Đây là một entity trong cơ sở dữ liệu, được ánh xạ với bảng NhanVien trong cơ sở dữ liệu thông qua các annotation của JPA (Jakarta Persistence API). Mô hình này có các trường dữ liệu liên quan đến thông tin nhân viên như mã nhân viên, họ tên, số điện thoại, địa chỉ, lương, và loại nhân viên. Mô hình NhanVien được sử dụng để quản lý thông tin nhân viên trong hệ thống. Các thuộc tính của

lớp giúp lưu trữ dữ liệu cần thiết cho mỗi nhân viên, và các phương thức getter/setter đảm bảo tính linh hoạt trong việc truy xuất và thay đổi thông tin nhân viên. Việc sử dụng kiểu dữ liệu BigDecimal cho lương giúp đảm bảo tính chính xác cao trong các phép toán tài chính. Lớp này đóng vai trò quan trọng trong việc kết nối giữa ứng dụng và cơ sở dữ liệu khi quản lý thông tin nhân viên.

NhanVienRepository là một interface dùng để truy cập và thao tác dữ liệu liên quan đến đối tượng NhanVien trong cơ sở dữ liệu. Nó kế thừa từ JpaRepository của Spring Data JPA, giúp dễ dàng thực hiện các thao tác cơ bản với cơ sở dữ liệu như lưu trữ, cập nhật, xóa và tìm kiếm đối tượng.

- **NhanVienRepository** giúp thao tác với bảng NhanVien trong cơ sở dữ liệu và sử dụng JpaRepository để thực hiện các thao tác CRUD một cách đơn giản và hiệu quả.
- Phương thức countPhanCongByMaNV cung cấp tính năng tùy chỉnh để đếm số lượng phân công mà nhân viên với mã maNV đã nhận, giúp hỗ trợ các tính toán hoặc kiểm tra về số lượng công việc đã được giao cho nhân viên đó.

Với cách triển khai này, **NhanVienRepository** có thể dễ dàng được sử dụng trong các service để thực hiện các thao tác cần thiết, bao gồm các thao tác mặc định hoặc những truy vấn tùy chỉnh như countPhanCongByMaNV.

NhanVienService là lớp dịch vụ trong ứng dụng Spring, chịu trách nhiệm xử lý các nghiệp vụ liên quan đến nhân viên (NhanVien). Lớp này sử dụng NhanVienRepository để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên cơ sở dữ liệu và chứa các phương thức logic xử lý yêu cầu từ Controller.

Các thành phần trong NhanVienService:

Tự động inject NhanVienRepository:

- @Autowired cho phép Spring tự động tiêm đối tượng NhanVienRepository vào trong lớp dịch vụ này để có thể truy cập các phương thức thao tác với cơ sở dữ liệu.

Danh sách các loại nhân viên hợp lệ (VALID_EMPLOYEE_TYPES):

- VALID_EMPLOYEE_TYPES là danh sách các loại nhân viên hợp lệ, bao gồm "Tiếp viên" và "Phi công". Phương thức sẽ kiểm tra dữ liệu loại nhân viên (loaiNV) trước khi thực hiện thêm hoặc sửa nhân viên.

Phương thức themNhanVien:

- Thêm mới một nhân viên vào hệ thống.
- Kiểm tra loại nhân viên có hợp lệ không (VALID_EMPLOYEE_TYPES).
- Chuyển đổi lương từ chuỗi (String) thành kiểu BigDecimal.
- Lưu nhân viên vào cơ sở dữ liệu thông qua nhanVienRepository.save(nhanVien).
- Phản hồi trả về thông báo kết quả: thành công hoặc lỗi.

Phương thức suaNhanVien:
- Cập nhật thông tin của nhân viên đã tồn tại.
- Kiểm tra loại nhân viên có hợp lệ không (VALID_EMPLOYEE_TYPES).
- Kiểm tra nhân viên có tồn tại trong cơ sở dữ liệu không bằng cách tìm theo mã nhân viên (maNV).
- Nếu nhân viên tồn tại, cập nhật các thông tin của nhân viên và lưu lại vào cơ sở dữ liệu.
- Phản hồi trả về thông báo kết quả: thành công hoặc lỗi.

Phương thức xoaNhanVien:
- Xóa một nhân viên khỏi hệ thống.
- Kiểm tra xem nhân viên có bị phân công công việc nào không bằng phương thức countPhanCongByMaNV(maNV). Nếu có công việc đã phân công, không thể xóa nhân viên này.
- Nếu không có phân công, xóa nhân viên bằng nhanVienRepository .deleteById(maNV).
- Phản hồi trả về thông báo kết quả: thành công hoặc lỗi.

Các phương thức trả về ResponseEntity<?>:
- ResponseEntity được sử dụng để gửi phản hồi HTTP với mã trạng thái và dữ liệu.
- hản hồi có thể bao gồm thông tin về trạng thái thành công (ok) hoặc lỗi (badRequest, internalServerError), cùng với một thông điệp mô tả.

3.5. Giao diện quản lý loại máy bay

Class **PlaneTypeController** cung cấp ba endpoint chính liên quan đến việc quản lý loại máy bay (plane types) trong hệ thống sân bay. Mỗi endpoint đều yêu cầu người dùng phải đăng nhập để thực hiện hành động, thông qua lớp LoginRequired được sử dụng như một annotation tùy chỉnh, đảm bảo tính bảo mật. Các phương thức trong controller đều gọi đến các service layer tương ứng để xử lý logic nghiệp vụ như thêm, sửa và xóa loại máy bay, giúp cấu trúc mã nguồn rõ ràng và dễ bảo trì.

Thêm loại máy bay *(/them_loai_mb*):

- Endpoint này sử dụng phương thức POST để thêm một loại máy bay mới vào hệ thống. Khi người dùng gửi yêu cầu đến endpoint này, các thông tin về loại máy bay mới, bao gồm plane-type-id (mã loại máy bay) và manufacturer (nhà sản xuất máy bay), sẽ được gửi dưới dạng tham số yêu cầu (**@RequestParam**).
- Sau khi nhận yêu cầu, controller sẽ gọi phương thức **themLoaiMayBay** trong LoaiMayBayService để xử lý việc thêm loại máy bay vào cơ sở dữ liệu. Nếu việc thêm loại máy bay thành công, hệ thống sẽ trả về một phản hồi thành công, ngược lại nếu có lỗi, một thông báo lỗi sẽ được trả về.

Sửa thông tin loại máy bay *(/sua_loai_mb/{planeTypeId}):*

- Endpoint này cho phép người dùng sửa thông tin của một loại máy bay đã có trong hệ thống. Để thực hiện yêu cầu này, người dùng cần cung cấp planeTypeId (ID của loại máy bay cần sửa) thông qua URL (được ánh xạ vào @PathVariable), và manufacturer (nhà sản xuất) qua tham số yêu cầu (@RequestParam).
- Sau khi nhận yêu cầu, controller sẽ gọi phương thức suaLoaiMayBay trong **LoaiMayBayService** để xử lý việc cập nhật thông tin. Cũng giống như các endpoint khác, nếu sửa thành công, hệ thống sẽ trả về một phản hồi thành công, và nếu có lỗi, một thông báo lỗi sẽ được trả về.

Xóa loại máy bay *(/xoa_loai_mb/{planeTypeId})*: Endpoint này sử dụng phương thức POST để xóa một loại máy bay khỏi hệ thống, dựa trên planeTypeId được truyền qua URL. Sau khi nhận yêu cầu, controller sẽ gọi phương thức xoaLoaiMayBay trong LoaiMayBayService để xử lý việc xóa loại máy bay khỏi cơ sở dữ liệu. Cũng như các hành động khác, hệ thống sẽ trả về phản hồi thành công hoặc lỗi tùy theo kết quả thực hiện.

Bảo mật với LoginRequired:

- Một điểm quan trọng trong tất cả các endpoint này là annotation @LoginRequired. Đây là một annotation tùy chỉnh có thể được sử dụng để đảm bảo chỉ những người dùng đã đăng nhập mới có quyền thực hiện các hành động thêm, sửa hoặc xóa loại máy bay. Nếu người dùng chưa đăng nhập, họ sẽ không thể thực hiện các thao tác này, giúp bảo vệ các dữ liệu nhạy cảm của hệ thống.

- Nhờ có LoginRequired, mỗi yêu cầu từ người dùng sẽ được kiểm tra xem có thuộc quyền truy cập của người dùng đã đăng nhập hay không. Nếu không, hệ thống sẽ từ chối yêu cầu và yêu cầu người dùng đăng nhập trước khi tiếp tục

Model **LoaiMayBay** trong đoạn mã trên là một lớp Entity trong ứng dụng sử dụng JPA (Java Persistence API) để ánh xạ với bảng dữ liệu trong cơ sở dữ liệu, cụ thể là bảng LoaiMayBay. Lớp này giúp lưu trữ và quản lý thông tin về các loại máy bay trong hệ thống. Cấu trúc và hoạt động của lớp này có thể được phân tích chi tiết như sau:

- **Annotation và ánh xạ bảng cơ sở dữ liệu**: Class **LoaiMayBay** được chú thích với annotation @Entity, điều này cho biết lớp này là một entity trong JPA và sẽ được ánh xạ vào một bảng trong cơ sở dữ liệu. Cụ thể, bảng mà lớp này ánh xạ đến có tên là LoaiMayBay, được xác định thông qua annotation @Table(name = "LoaiMayBay"). Điều này có nghĩa là tất cả các đối tượng của lớp LoaiMayBay sẽ tương ứng với các bản ghi trong bảng LoaiMayBay trong cơ sở dữ liệu.

- **Khóa chính và các cột trong bảng:** Trong lớp này, thuộc tính maLoai được đánh dấu là khóa chính của bảng thông qua annotation @Id. Điều này có nghĩa là mỗi đối tượng LoaiMayBay phải có một giá trị duy nhất cho maLoai để phân biệt giữa các bản ghi khác nhau trong cơ sở dữ liệu. Annotation @Column(name = "MaLoai") được sử dụng để ánh xạ thuộc tính này tới cột MaLoai trong bảng LoaiMayBay. Tương tự, thuộc tính hangSanXuat cũng được ánh xạ tới cột HangSanXuat trong bảng, thông qua annotation @Column(name = "HangSanXuat").

- **Constructor:** Lớp này bao gồm hai constructor: một constructor mặc định không tham số (public LoaiMayBay() {}) và một constructor có tham số (public LoaiMayBay(String maLoai, String hangSanXuat)). Constructor mặc định được yêu cầu bởi JPA để có thể tạo đối tượng của lớp mà không cần phải cung cấp thông tin ngay lập tức. Trong khi đó, constructor có tham số giúp khởi tạo đối tượng LoaiMayBay với các giá trị cụ thể cho maLoai và hangSanXuat. Việc này giúp thuận tiện hơn khi tạo đối tượng trong các trường hợp cần thiết.

- **Getter và Setter:** Các phương thức getter và setter được cung cấp để truy cập và thay đổi giá trị của các thuộc tính maLoai và hangSanXuat. Phương thức getMaLoai() và setMaLoai(String maLoai) cho phép lấy và cập nhật giá trị

của thuộc tính maLoai, trong khi getHangSanXuat() và setHangSanXuat(String hangSanXuat) cho phép lấy và cập nhật giá trị của thuộc tính hangSanXuat. Các phương thức này là tiêu chuẩn trong Java để đảm bảo tính đóng gói (encapsulation), cho phép kiểm soát việc truy cập và thay đổi dữ liệu của lớp.

Model **LoaiMayBay** là một lớp entity được sử dụng để ánh xạ tới bảng LoaiMayBay trong cơ sở dữ liệu. Lớp này lưu trữ thông tin về mã loại máy bay (maLoai) và hãng sản xuất máy bay (hangSanXuat). Các thuộc tính này được ánh xạ tới các cột tương ứng trong cơ sở dữ liệu, đảm bảo tính nhất quán giữa đối tượng và dữ liệu trong bảng. Lớp còn có các phương thức getter và setter để thao tác với các thuộc tính, cùng với các constructor giúp khởi tạo đối tượng một cách linh hoạt. Nhờ vào JPA, lớp này có thể dễ dàng lưu trữ và truy vấn dữ liệu từ cơ sở dữ liệu mà không cần phải viết SQL thủ công.

LoaiMayBayRepository là một giao diện Spring Data JPA được sử dụng để tương tác với bảng LoaiMayBay trong cơ sở dữ liệu. Giao diện này kế thừa từ JpaRepository<LoaiMayBay, String>, giúp cung cấp các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) cho thực thể LoaiMayBay (được ánh xạ với bảng LoaiMayBay trong cơ sở dữ liệu).
Bên cạnh các phương thức kế thừa, giao diện này còn định nghĩa một phương thức truy vấn tùy chỉnh, countUsedTypes, để kiểm tra số lần mà một loại máy bay (maLoai) được sử dụng trong các bảng khác như **MayBay** và **LichBay**. Phương thức này sử dụng chú thích @Query với một câu truy vấn SQL gốc để đếm số lượng bản ghi trong các bảng này có maLoai trùng với giá trị đầu vào. Câu truy vấn kết hợp kết quả từ hai bảng MayBay và LichBay bằng UNION ALL. Nếu số lượng bản ghi lớn hơn 0, có nghĩa là loại máy bay đang được sử dụng, điều này rất hữu ích khi cần ngăn chặn việc xóa hoặc thay đổi loại máy bay đang được sử dụng. ung cấp một phương thức tùy chỉnh để đếm số lần mà một loại máy bay (maLoai) được sử dụng trong các bảng khác như MayBay và LichBay, giúp ngăn ngừa việc xóa hoặc thay đổi loại máy bay khi nó đang được sử dụng.

LichBayService là lớp dịch vụ xử lý các logic nghiệp vụ liên quan đến lịch bay (LichBay). Lớp này quản lý việc thêm, cập nhật và xóa các lịch bay bằng cách sử dụng LichBayRepository.

Phương thức: themLich:

- Phương thức này dùng để thêm mới một lịch bay. Đầu tiên, nó kiểm tra xem chuyến bay (maChuyenBay) đã tồn tại trong lịch bay chưa thông qua lichBayRepository.existsByMaChuyenBay(maChuyenBay).
- Nếu đã tồn tại, phương thức sẽ trả về phản hồi 400 Bad Request với thông báo lỗi.
- Nếu chuyến bay chưa có lịch bay, nó sẽ tìm kiếm maLoai (loại máy bay) từ số hiệu máy bay (soHieu) bằng cách sử dụng lichBayRepository.findMaLoaiByMayBay(soHieu).
- Nếu không tìm thấy loại máy bay, trả về thông báo lỗi.
- Sau đó, phương thức tạo một đối tượng LichBay mới với thông tin chuyến bay và lưu vào cơ sở dữ liệu.
- Nếu thành công, phương thức sẽ trả về thông báo thành công.

Phương thức: suaLich:
- Phương thức này cập nhật một lịch bay đã tồn tại. Đầu tiên, nó kiểm tra xem lịch bay có tồn tại không bằng lichBayRepository.existsByMaChuyenBay(maChuyenBay).
- Nếu lịch bay không tồn tại, trả về phản hồi 400 Bad Request.
- Nếu tồn tại, phương thức lấy đối tượng LichBay, cập nhật lại số hiệu máy bay (soHieu), và lưu lại đối tượng đã cập nhật vào cơ sở dữ liệu.
- Phương thức trả về thông báo thành công nếu việc cập nhật thành công.

Phương thức: xoaLich:
- Phương thức này dùng để xóa một lịch bay. Trước khi xóa, phương thức kiểm tra xem có bất kỳ đơn đặt chỗ (DatCho) hoặc phân công (PhanCong) nào sử dụng mã chuyến bay (maChuyenBay) không.
- Nếu có các bản ghi liên quan trong bảng DatCho hoặc PhanCong, phương thức sẽ ngăn chặn việc xóa và trả về thông báo lỗi 400 Bad Request.
- Nếu không có các phụ thuộc, lịch bay sẽ bị xóa khỏi cơ sở dữ liệu và phương thức trả về thông báo thành công.

Mỗi phương thức đều được bao bọc trong một khối try-catch, giúp xử lý các lỗi xảy ra trong quá trình thực hiện và trả về phản hồi lỗi server nếu có ngoại lệ. LoaiMayBayRepository cung cấp một phương thức tùy chỉnh để đếm số lần mà một loại máy bay (maLoai) được sử dụng trong các bảng khác như MayBay và LichBay, giúp ngăn ngừa việc xóa hoặc thay đổi loại máy bay khi nó đang được sử dụng.

3.6. Giao diện quản lý máy bay

PlaneController là một controller trong ứng dụng Spring Boot, chịu trách nhiệm xử lý các yêu cầu HTTP liên quan đến việc quản lý máy bay (MayBay). Controller này bao gồm các phương thức để thêm, sửa, và xóa thông tin máy bay. Tất cả các phương thức này đều yêu cầu người dùng phải đăng nhập và có quyền truy cập (được xác nhận qua chú thích **@LoginRequired**).

- **themMayBay**: Phương thức này xử lý yêu cầu thêm máy bay mới vào hệ thống. Người dùng phải cung cấp thông tin bao gồm mã máy bay (plane-id), mã loại máy bay (plane-type-id) và số lượng ghế (seat-quantity). Sau khi nhận được yêu cầu, hệ thống sẽ gọi dịch vụ **MayBayService.themMayBay**() để thực hiện việc thêm máy bay. Nếu thêm thành công, phương thức trả về một phản hồi xác nhận.

- **suaMayBay**: Đây là phương thức xử lý yêu cầu sửa đổi thông tin của một máy bay đã có trong hệ thống. Người dùng cần cung cấp mã máy bay cùng với các thông tin cập nhật như mã loại máy bay và số lượng ghế. Phương thức sẽ gọi dịch vụ **MayBayService.suaMayBay()** để cập nhật thông tin của máy bay trong cơ sở dữ liệu. Khi việc sửa đổi hoàn tất, phản hồi xác nhận sẽ được trả về.

- **xoaMayBay**: Phương thức này cho phép xóa một máy bay khỏi hệ thống. Mã máy bay (planeId) được cung cấp thông qua tham số đường dẫn (path variable). Trước khi xóa máy bay, hệ thống sẽ kiểm tra xem máy bay có đang được sử dụng trong các chuyến bay hoặc lịch bay nào không. Nếu không có vấn đề gì, máy bay sẽ bị xóa và hệ thống trả về phản hồi xác nhận thành công. Nếu có lỗi hoặc máy bay không thể xóa, hệ thống sẽ thông báo lỗi.

Model **MayBay** đại diện cho đối tượng máy bay trong hệ thống. Lớp này sử dụng các annotation của Jakarta Persistence để ánh xạ với bảng MayBay trong cơ sở dữ liệu. Các thành phần của lớp bao gồm:

- **Thuộc tính** soHieu: Đây là mã số hiệu của máy bay, được ánh xạ với cột SoHieu trong bảng MayBay. Nó có độ dài tối đa là 10 ký tự và được đánh dấu là khóa chính với annotation @Id.
- **Thuộc tính** maLoai: Đây là mã loại máy bay, tương ứng với cột MaLoai trong bảng. Nó chỉ định loại máy bay mà máy bay này thuộc về, ví dụ như máy bay thương mại, máy bay chở hàng, v.v.

- **Thuộc tính** soGheNgoi: Đây là số lượng ghế ngồi của máy bay, tương ứng với cột SoGheNgoi trong bảng. Nó cho biết khả năng chở hành khách của máy bay.

Constructors:
- **Constructor mặc định**: Cho phép tạo đối tượng MayBay mà không cần truyền tham số.
- **Constructor có tham số**: Cho phép khởi tạo đối tượng MayBay với các giá trị cụ thể cho soHieu, maLoai, và soGheNgoi.

Getters và Setters: Lớp cung cấp các phương thức getter và setter cho từng thuộc tính, cho phép truy xuất và thay đổi giá trị của các thuộc tính soHieu, maLoai, và soGheNgoi.

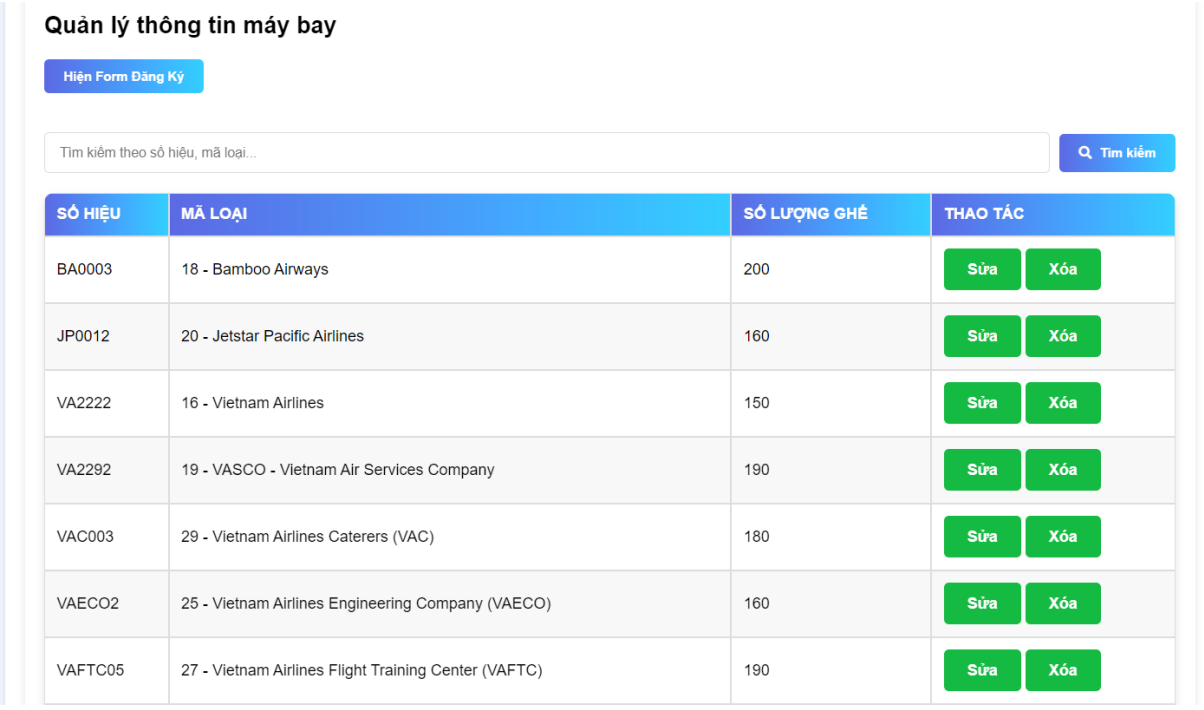
MayBayService là một lớp dịch vụ trong hệ thống quản lý sân bay, có nhiệm vụ xử lý các yêu cầu liên quan đến máy bay. Lớp này tương tác với cơ sở dữ liệu thông qua **MayBayRepository** để thực hiện các thao tác thêm, sửa, và xóa máy bay. Các phương thức trong lớp **MayBayService** đảm bảo tính toàn vẹn dữ liệu và trả về thông báo chi tiết, giúp người dùng dễ dàng theo dõi trạng thái của các thao tác.

Phương thức themMayBay: Phương thức này thực hiện việc thêm một máy bay mới vào hệ thống. Trước khi tiến hành thêm, nó thực hiện các kiểm tra:
- **Độ dài số hiệu máy bay**: Kiểm tra xem số hiệu máy bay có vượt quá 10 ký tự hay không. Nếu có, trả về lỗi.
- **Kiểm tra sự tồn tại của máy bay**: Sử dụng existsBySoHieu để kiểm tra xem số hiệu máy bay đã tồn tại trong hệ thống chưa. Nếu có, trả về thông báo lỗi.
- **Loại máy bay**: Kiểm tra xem loại máy bay (maLoai) có tồn tại trong cơ sở dữ liệu không bằng phương thức countLoaiMayBayByMaLoai. Nếu không tồn tại, trả về lỗi.
- **Số ghế ngồi**: Kiểm tra tính hợp lệ của số ghế ngồi, phải là một số nguyên dương và lớn hơn 0. Nếu không hợp lệ, trả về lỗi. Nếu tất cả các điều kiện trên đều hợp lệ, hệ thống sẽ tạo một đối tượng MayBay mới và lưu vào cơ sở dữ liệu, sau đó trả về thông báo thành công.

Phương thức suaMayBay: Phương thức này được sử dụng để sửa thông tin của một máy bay đã tồn tại trong hệ thống. Các bước kiểm tra trước khi thực hiện sửa thông tin bao gồm:

- **Kiểm tra sự tồn tại của máy bay**: Xác nhận máy bay có tồn tại trong hệ thống không qua existsById. Nếu không, trả về lỗi.
- **Kiểm tra việc sử dụng máy bay trong lịch bay**: Nếu máy bay đã được sử dụng trong lịch bay (kiểm tra qua countLichBayBySoHieu), hệ thống không cho phép sửa đổi và trả về thông báo lỗi.
- Sau khi vượt qua các kiểm tra trên, máy bay sẽ được cập nhật với thông tin mới và lưu vào cơ sở dữ liệu. Thông báo thành công được trả về khi thao tác hoàn tất.

Phương thức xoaMayBay: Phương thức này cho phép xóa một máy bay khỏi hệ thống. Trước khi xóa, hệ thống thực hiện các kiểm tra:
- **Máy bay có đang được sử dụng không**: Sử dụng countLichBayBySoHieu để kiểm tra xem máy bay có đang được sử dụng trong lịch bay không. Nếu có, không thể xóa máy bay.
- **Loại máy bay có đang sử dụng không**: Kiểm tra xem loại máy bay có đang được sử dụng trong các bảng khác không. Nếu có, trả về lỗi không thể xóa. Nếu máy bay không bị ràng buộc bởi bất kỳ bảng nào, máy bay sẽ được xóa và hệ thống trả về thông báo thành công.



Chương 5_hình 3.6-1. Thông tin máy bay

3.7. Giao diện quản lý chuyến bay

FlightController cung cấp ba endpoint chính cho việc quản lý chuyến bay trong hệ thống sân bay. Các endpoint này yêu cầu người dùng phải đăng nhập để thực hiện các hành động, thông qua lớp LoginRequired, giúp đảm bảo tính bảo mật. Các phương thức trong controller này gọi đến ChuyenBayService để xử lý logic nghiệp vụ cho các thao tác thêm, sửa và xóa chuyến bay, giúp mã nguồn được tổ chức tốt và dễ bảo trì.

Thêm chuyến bay *(/them_cb*):

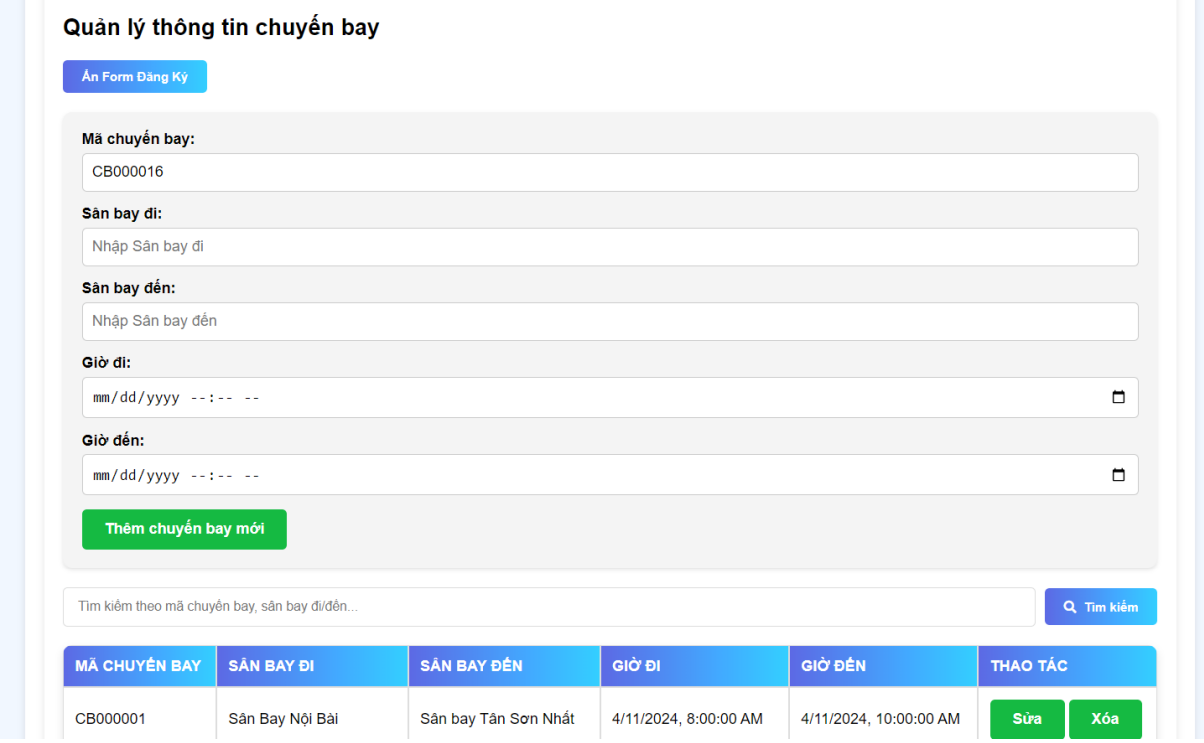
- Endpoint này sử dụng phương thức POST để thêm một chuyến bay mới. Khi người dùng gửi yêu cầu đến endpoint, các thông tin về chuyến bay mới, bao gồm flight-id (mã chuyến bay), departure-airport (sân bay khởi hành), arrival-airport (sân bay đến), departure-time (thời gian khởi hành) và arrival-time (thời gian đến) sẽ được gửi qua @RequestParam.
- Sau khi nhận yêu cầu, **FlightController** sẽ gọi themChuyenBay trong bđể xử lý thêm chuyến bay vào cơ sở dữ liệu. Nếu thêm thành công, hệ thống sẽ trả về phản hồi xác nhận; ngược lại, nếu có lỗi thì thông báo lỗi sẽ được trả về.

Sửa thông tin chuyến bay *(/sua_cb*):

- Endpoint này cho phép người dùng sửa thông tin của một chuyến bay đã có. Để thực hiện, người dùng cần cung cấp flight-id qua tham số yêu cầu **@RequestParam**, cùng với các thông tin khác như departure-airport, arrival-airport, departure-time và arrival-time.
- Sau khi nhận yêu cầu, **FlightController** gọi phương thức **suaChuyenBay** trong **ChuyenBayService** để xử lý việc cập nhật. Nếu thành công, hệ thống sẽ trả về phản hồi xác nhận; nếu có lỗi, một thông báo lỗi sẽ được trả về.

Xóa chuyến bay *(/xoa_cb/{flightId}):*

- Endpoint này sử dụng phương thức POST để xóa một chuyến bay, với flightId được truyền qua URL **@PathVariable**. Sau khi nhận yêu cầu, controller sẽ gọi phương thức xoaChuyenBay trong ChuyenBayService để xử lý việc xóa chuyến bay khỏi cơ sở dữ liệu. Hệ thống sẽ trả về phản hồi xác nhận hoặc thông báo lỗi tùy theo kết quả thực hiện.



Chương 5_hình 3.7-1. Thông tin chuyến bay

Lớp **ChuyenBay** là một entity được sử dụng để ánh xạ tới bảng ChuyenBay trong cơ sở dữ liệu. Lớp này lưu trữ thông tin về mã chuyến bay (maChuyenBay), tên sân bay đi (tenSanBayDi), tên sân bay đến (tenSanBayDen), giờ đi (gioDi), và giờ đến (gioDen). Các thuộc tính này được ánh xạ tới các cột tương ứng trong bảng, đảm bảo tính nhất quán giữa đối tượng và dữ liệu trong cơ sở dữ liệu.

Các thuộc tính của lớp ChuyenBay:

- maChuyenBay: Đây là khóa chính của bảng ChuyenBay, đại diện cho mã chuyến bay, được khai báo với annotation @Id để xác định đây là khóa chính, cùng với @Column(name = "MaChuyenBay") để ánh xạ thuộc tính này đến cột MaChuyenBay.
- tenSanBayDi: Đại diện cho tên sân bay nơi chuyến bay khởi hành, được ánh xạ đến cột TenSanBayDi trong bảng cơ sở dữ liệu.
- tenSanBayDen: Đại diện cho tên sân bay nơi chuyến bay sẽ hạ cánh, được ánh xạ đến cột TenSanBayDen.
- gioDi: Lưu trữ thông tin về thời gian khởi hành của chuyến bay và được ánh xạ đến cột GioDi.
- gioDen: Lưu trữ thông tin về thời gian đến của chuyến bay và được ánh xạ đến cột GioDen.

Constructor của lớp ChuyenBay: Lớp có một constructor mặc định để hỗ trợ JPA trong việc khởi tạo đối tượng, cùng với một constructor đầy đủ tham số giúp khởi tạo một đối tượng ChuyenBay với các giá trị cụ thể cho từng thuộc tính.

Các phương thức getter và setter: Lớp này cung cấp các phương thức getter và setter cho mỗi thuộc tính, giúp truy cập và thay đổi giá trị của từng thuộc tính theo cách thức an toàn.

Hỗ trợ từ JPA: Nhờ vào JPA, lớp ChuyenBay có thể dễ dàng lưu trữ và truy vấn dữ liệu từ cơ sở dữ liệu mà không cần viết SQL thủ công. Điều này giúp tăng tính linh hoạt và giảm thiểu mã nguồn cần thiết trong các thao tác cơ bản với cơ sở dữ liệu.

ChuyenBayRepository là một giao diện Spring Data JPA được sử dụng để tương tác với bảng ChuyenBay trong cơ sở dữ liệu. Giao diện này kế thừa từ JpaRepository<ChuyenBay, String>, giúp cung cấp các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) cho thực thể ChuyenBay (được ánh xạ với bảng ChuyenBay trong cơ sở dữ liệu).

Ngoài các phương thức CRUD kế thừa, **ChuyenBayRepository** còn định nghĩa một phương thức truy vấn tùy chỉnh, countUsedFlights, để kiểm tra số lần một chuyến bay cụ thể (dựa trên mã chuyến bay maChuyenBay) được sử dụng trong các bảng khác như LichBay, DatCho, và PhanCong. Phương thức này sử dụng chú thích @Query cùng với một câu truy vấn SQL gốc để đếm số lượng bản ghi có chứa mã chuyến bay tương ứng trong các bảng này.

Câu truy vấn sử dụng UNION ALL để kết hợp kết quả từ ba bảng LichBay, DatCho, và PhanCong. Mỗi bảng sẽ trả về các bản ghi với mã chuyến bay khớp với giá trị maChuyenBay đầu vào của phương thức. Cuối cùng, COUNT(*) sẽ đếm tổng số bản ghi từ cả ba bảng.

Phương thức countUsedFlights trả về một số nguyên (int), cho biết tổng số lần chuyến bay được sử dụng trong các bảng trên. Nếu số lượng bản ghi lớn hơn 0, điều này nghĩa là chuyến bay đang được sử dụng, thông tin này rất hữu ích để ngăn chặn việc xóa hoặc thay đổi chuyến bay nếu nó đang có liên kết với lịch trình, đặt chỗ, hoặc phân công nhân viên.

ChuyenBayService là một lớp dịch vụ xử lý các logic nghiệp vụ liên quan đến chuyến bay (ChuyenBay). Lớp này quản lý việc thêm, cập nhật, và xóa các chuyến bay thông qua **ChuyenBayRepository**.

Phương thức: **themChuyenBay**

- **Chức năng:** Thêm mới một chuyến bay vào cơ sở dữ liệu.
- **Quy trình:**
  - Kiểm tra điều kiện rằng sân bay đi (**tenSanBayDi**) và sân bay đến (**tenSanBayDen**) phải khác nhau.
    - Nếu giống nhau, phương thức sẽ trả về trang **/admin** kèm thông báo lỗi qua **RedirectAttributes**.
  - Chuyển đổi thời gian khởi hành (**gioDi**) và thời gian hạ cánh (**gioDen**) từ chuỗi (String) sang kiểu **LocalDateTime**.
  - Tạo một đối tượng **ChuyenBay** với các thông tin đầu vào.
  - Gọi phương thức **save** từ **ChuyenBayRepository** để lưu đối tượng vào cơ sở dữ liệu.
  - Nếu thành công, phương thức trả về trang **/admin** kèm thông báo thành công. Nếu có lỗi xảy ra, trả về trang **/admin** với thông báo lỗi.

Phương thức: **suaChuyenBay**

- **Chức năng:** Cập nhật thông tin của một chuyến bay đã tồn tại.
- **Quy trình:**
  - Kiểm tra điều kiện rằng sân bay đi (**tenSanBayDi**) và sân bay đến (**tenSanBayDen**) phải khác nhau.
  - Nếu giống nhau, phương thức trả về phản hồi HTTP 400 với thông báo lỗi.
  - Sử dụng **findById** từ **ChuyenBayRepository** để kiểm tra chuyến bay có tồn tại không.
  - Nếu không tìm thấy, phương thức trả về phản hồi HTTP 400 với thông báo lỗi.
  - Chuyển đổi thời gian khởi hành (**gioDi**) và thời gian hạ cánh (**gioDen**) từ chuỗi (String) sang kiểu **LocalDateTime**.
  - Cập nhật các thông tin của đối tượng **ChuyenBay**, bao gồm sân bay đi, sân bay đến, giờ đi, và giờ đến.
  - Gọi phương thức **save** từ **ChuyenBayRepository** để lưu lại đối tượng đã cập nhật.

- Nếu thành công, trả về phản hồi HTTP 200 kèm thông báo thành công. Nếu có lỗi xảy ra, trả về phản hồi HTTP 500 với thông báo lỗi.

Phương thức: **xoaChuyenBay**

- **Chức năng:** Xóa một chuyến bay.
- **Quy trình:**
  - Sử dụng phương thức **countUsedFlights** từ **ChuyenBayRepository** để kiểm tra xem chuyến bay có đang được sử dụng trong các bảng khác như **LichBay**, **DatCho**, hoặc **PhanCong** không.
  - Nếu chuyến bay đang được sử dụng, trả về phản hồi HTTP 400 với thông báo lỗi, ngăn chặn việc xóa.
  - Nếu không có ràng buộc, gọi phương thức **deleteById** từ **ChuyenBayRepository** để xóa chuyến bay khỏi cơ sở dữ liệu.
  - Nếu thành công, trả về phản hồi HTTP 200 kèm thông báo thành công. Nếu có lỗi xảy ra, trả về phản hồi HTTP 500 với thông báo lỗi.

Xử lý lỗi:

Tất cả các phương thức trong **ChuyenBayService** đều được bao bọc bởi khối **try-catch**, đảm bảo rằng mọi ngoại lệ xảy ra trong quá trình thực hiện sẽ được xử lý và trả về phản hồi lỗi tương ứng. Điều này giúp tăng tính ổn định và thân thiện của hệ thống.

Lớp dịch vụ này cung cấp các logic nghiệp vụ quan trọng để quản lý chuyến bay, đảm bảo tính toàn vẹn dữ liệu và giảm thiểu lỗi người dùng.

3.8. Giao diện quản lý lịch bay

Trong lớp **FlightScheduleController**, ba phương thức được định nghĩa để xử lý các yêu cầu liên quan đến lịch bay. Lớp này sử dụng dịch vụ **LichBayService** để thực hiện các thao tác thêm, sửa và xóa lịch bay. Mỗi phương thức đều được chú thích là yêu cầu phải đăng nhập qua annotation @LoginRequired và trả về kết quả dưới dạng ResponseEntity với thông báo chi tiết về kết quả thao tác.

- **Phương thức** themLich: Phương thức này nhận vào hai tham số qua @RequestParam: flight-id (mã chuyến bay) và aircraft-id (mã máy bay). Nó gọi phương thức themLich trong LichBayService để thêm một lịch bay mới

vào hệ thống. Phương thức này trả về ResponseEntity với thông báo kết quả thao tác. Nếu việc thêm lịch bay thành công, thông báo sẽ cho biết lịch bay đã được thêm, ngược lại sẽ trả về thông báo lỗi nếu gặp vấn đề.

- **Phương thức** suaLich: Phương thức này cũng nhận hai tham số qua @RequestParam: flight-id và aircraft-id, dùng để cập nhật thông tin của một lịch bay đã có. Nó gọi phương thức suaLich trong LichBayService để thực hiện việc sửa thông tin lịch bay. Tương tự như phương thức thêm, kết quả thao tác sẽ được trả về dưới dạng ResponseEntity để thông báo thành công hoặc thất bại của việc sửa.

- **Phương thức** xoaLich: Phương thức này cho phép xóa một lịch bay khỏi hệ thống, nhận vào hai tham số flight-id và aircraft-id. Nó gọi phương thức xoaLich trong LichBayService để thực hiện thao tác xóa lịch bay. Kết quả cũng sẽ được trả về qua ResponseEntity, thông báo thành công nếu lịch bay được xóa thành công hoặc thông báo lỗi nếu có vấn đề trong quá trình xóa.

Mỗi phương thức trong FlightScheduleController đều sử dụng dịch vụ LichBayService để xử lý logic nghiệp vụ và trả về phản hồi chi tiết về kết quả thao tác. Lớp controller này đảm bảo rằng các thao tác quản lý lịch bay được thực hiện một cách hợp lý và người dùng nhận được thông tin phản hồi rõ ràng.

Trong lớp LichBay, một thực thể của bảng "LịchBay" được mô tả với các thuộc tính và phương thức liên quan. Đây là một lớp model trong ứng dụng sử dụng JPA (Java Persistence API) để ánh xạ với cơ sở dữ liệu, lưu trữ thông tin lịch bay của các chuyến bay.

Các thuộc tính của lớp LichBay:

- **maChuyenBay (Mã chuyến bay)**: Đây là thuộc tính dùng để lưu mã của chuyến bay, là khóa chính trong bảng LichBay. Thuộc tính này được ánh xạ với cột MaChuyenBay trong cơ sở dữ liệu.
- **ngayDi (Ngày đi)**: Thuộc tính này lưu trữ ngày khởi hành của chuyến bay, được ánh xạ với cột NgayDi trong cơ sở dữ liệu. Kiểu dữ liệu là LocalDate, đại diện cho một ngày mà không có thời gian.
- **soHieu (Số hiệu máy bay):** Thuộc tính này lưu số hiệu của máy bay, giúp xác định chiếc máy bay thực hiện chuyến bay này. Nó được ánh xạ với cột SoHieu trong cơ sở dữ liệu.

- **maLoai (Mã loại máy bay):** Thuộc tính này lưu mã loại máy bay (ví dụ: A320, B777), ánh xạ với cột MaLoai trong cơ sở dữ liệu.

Các phương thức:

- **Constructor:**
  - LichBay(): Constructor không tham số để khởi tạo đối tượng mà không có dữ liệu.
  - LichBay(String maChuyenBay, LocalDate ngayDi, String soHieu, String maLoai): Constructor với các tham số, giúp khởi tạo đối tượng LichBay với dữ liệu ngay từ đầu.

Getters và Setters: Các phương thức get và set cho phép truy xuất và cập nhật giá trị của các thuộc tính. Ví dụ, getMaChuyenBay() trả về mã chuyến bay, trong **khi setMaChuyenBay(String maChuyenBay)** giúp cập nhật giá trị của mã chuyến bay.

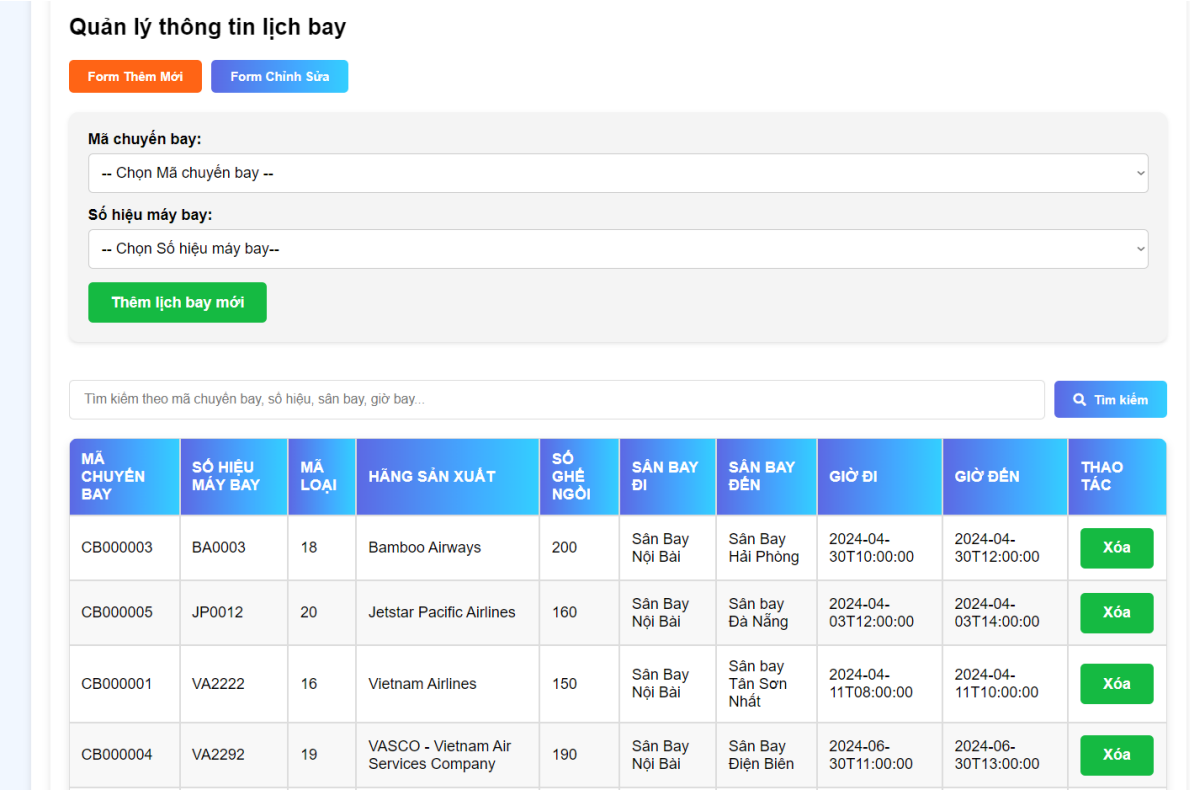
Lớp **LichBay** đóng vai trò quan trọng trong hệ thống quản lý chuyến bay, lưu trữ thông tin chuyến bay bao gồm mã chuyến bay, ngày khởi hành, số hiệu máy bay và loại máy bay. Các phương thức getter và setter giúp làm việc với các thuộc tính này, và constructor cho phép khởi tạo đối tượng dễ dàng khi cần thiết.

Trong lớp **LichBayRepository**, chúng ta định nghĩa các phương thức truy vấn cơ sở dữ liệu liên quan đến thông tin lịch bay. Đây là một lớp repository sử dụng Spring Data JPA để tương tác với cơ sở dữ liệu.
Các phương thức của **LichBayRepository**:

- **existsByMaChuyenBay(String maChuyenBay)**: Phương thức này kiểm tra xem có tồn tại một chuyến bay nào trong cơ sở dữ liệu với mã chuyến bay (maChuyenBay) được cung cấp hay không. Nó trả về true nếu chuyến bay tồn tại và false nếu không.
- **countDatChoByMaChuyenBay(String maChuyenBay)**: Phương thức này sử dụng câu truy vấn SQL thuần (nativeQuery) để đếm số lượng đặt chỗ (DatCho) liên quan đến chuyến bay có mã chuyến bay (maChuyenBay). Câu truy vấn trả về số lượng các bản ghi trong bảng DatCho mà có trường MaChuyenBay tương ứng với mã chuyến bay cung cấp.
- **countPhanCongByMaChuyenBay(String maChuyenBay)**: Phương thức này đếm số lượng phân công (PhanCong) liên quan đến chuyến bay có mã chuyến bay (maChuyenBay). Câu truy vấn tương tự như phương thức trên,

nhưng áp dụng trên bảng PhanCong, trả về số lượng bản ghi có MaChuyenBay tương ứng.

- **findMaLoaiByMayBay(String soHieu)**: Phương thức này tìm mã loại máy bay (MaLoai) dựa trên số hiệu máy bay (soHieu). Câu truy vấn này tra cứu bảng MayBay và trả về MaLoai của máy bay với số hiệu tương ứng.



Chương 5_hình 3.8-1. Lịch bay

Class **LichBayRepository** cung cấp các phương thức quan trọng để làm việc với dữ liệu lịch bay trong cơ sở dữ liệu. Nó không chỉ kiểm tra sự tồn tại của chuyến bay mà còn đếm các bản ghi liên quan như đặt chỗ và phân công, giúp kiểm soát việc sử dụng dữ liệu trong các bảng khác nhau liên quan đến lịch bay. Phương thức **findMaLoaiByMayBay** còn giúp tra cứu thông tin máy bay dựa trên số hiệu, hỗ trợ việc xác thực và xử lý trong các dịch vụ khác của hệ thống.

Trong lớp **LichBayService**, các phương thức xử lý các hoạt động liên quan đến lịch bay, bao gồm thêm mới, cập nhật và xóa lịch bay. Mỗi phương thức đều sử dụng Spring ResponseEntity để trả về kết quả với mã trạng thái HTTP tương ứng và thông điệp chi tiết.

Giải thích chi tiết về các phương thức:

- themLich(String maChuyenBay, String soHieu):

- **Mục đích**: Thêm mới một lịch bay cho chuyến bay (maChuyenBay) và máy bay (soHieu).
- **Quy trình**:
    - Kiểm tra xem mã chuyến bay đã tồn tại trong hệ thống chưa. Nếu có, trả về lỗi với thông báo "Mã chuyến bay này đã có lịch bay!".
    - Lấy mã loại máy bay từ soHieu bằng cách gọi phương thức findMaLoaiByMayBay. Nếu không tìm thấy mã loại máy bay, trả về lỗi "Không tìm thấy MaLoai cho máy bay này".
    - Tạo một đối tượng LichBay mới với thông tin chuyến bay, ngày đi (được lấy từ hệ thống hiện tại hoặc có thể thay thế bằng thông tin từ một đối tượng ChuyenBay nếu có) và lưu vào cơ sở dữ liệu.
- **Kết quả**: Nếu tất cả kiểm tra đều hợp lệ, lịch bay mới sẽ được thêm vào và trả về thông báo thành công. Nếu có lỗi, trả về thông báo lỗi chi tiết.

- suaLich(String maChuyenBay, String soHieu):
    - **Mục đích**: Cập nhật lịch bay của một chuyến bay đã tồn tại.
    - **Quy trình**:
        - Kiểm tra xem lịch bay có tồn tại trong hệ thống không. Nếu không, trả về lỗi "Lịch bay không tồn tại".
        - Nếu tồn tại, lấy đối tượng LichBay từ cơ sở dữ liệu và cập nhật thông tin máy bay (soHieu) cho lịch bay đó.
    - **Kết quả**: Nếu cập nhật thành công, trả về thông báo "Cập nhật lịch bay thành công". Nếu có lỗi trong quá trình thực thi, trả về lỗi chi tiết.
- xoaLich(String maChuyenBay, String soHieu):
    - **Mục đích**: Xóa lịch bay của một chuyến bay nếu không có dữ liệu liên quan trong các bảng khác như DatCho và PhanCong.
    - **Quy trình**:
        - Kiểm tra số lượng đặt chỗ (DatCho) và phân công (PhanCong) liên quan đến chuyến bay (maChuyenBay). Nếu có bất kỳ đặt chỗ hay phân công nào, trả về lỗi "Không thể xóa. Mã chuyến bay đã được sử dụng trong bảng DatCho hoặc PhanCong".
        - Nếu không có liên quan, tiến hành xóa lịch bay nếu số hiệu máy bay (soHieu) khớp với lịch bay đã tồn tại.
    - **Kết quả**: Nếu xóa thành công, trả về thông báo "Xóa lịch bay thành công". Nếu không tìm thấy lịch bay phù hợp hoặc có lỗi trong quá trình xóa, trả về thông báo lỗi chi tiết.

Những điểm nổi bật:

- **Kiểm tra điều kiện trước khi thực hiện hành động**: Mỗi phương thức đều có các kiểm tra để đảm bảo tính toàn vẹn dữ liệu, chẳng hạn như kiểm tra xem mã chuyến bay có tồn tại không, số hiệu máy bay có hợp lệ không, hay liệu chuyến bay đã có đặt chỗ hay phân công chưa.
- **Quản lý lỗi**: Các lỗi trong quá trình xử lý đều được bắt và trả về thông báo chi tiết để giúp người dùng hiểu nguyên nhân gây ra lỗi.
- **Sử dụng** ResponseEntity: ResponseEntity giúp đóng gói kết quả trả về bao gồm mã trạng thái HTTP và thông tin lỗi hoặc thành công, giúp hệ thống frontend có thể xử lý dễ dàng.

3.9. Giao diện quản lý phân công

AssignmentController là lớp điều khiển (controller) xử lý các yêu cầu HTTP liên quan đến việc tạo, cập nhật, xóa và truy xuất thông tin phân công công việc cho nhân viên đối với các chuyến bay. Các yêu cầu được bảo vệ thông qua annotation @LoginRequired, đảm bảo rằng người dùng phải đăng nhập trước khi thực hiện các hành động này.

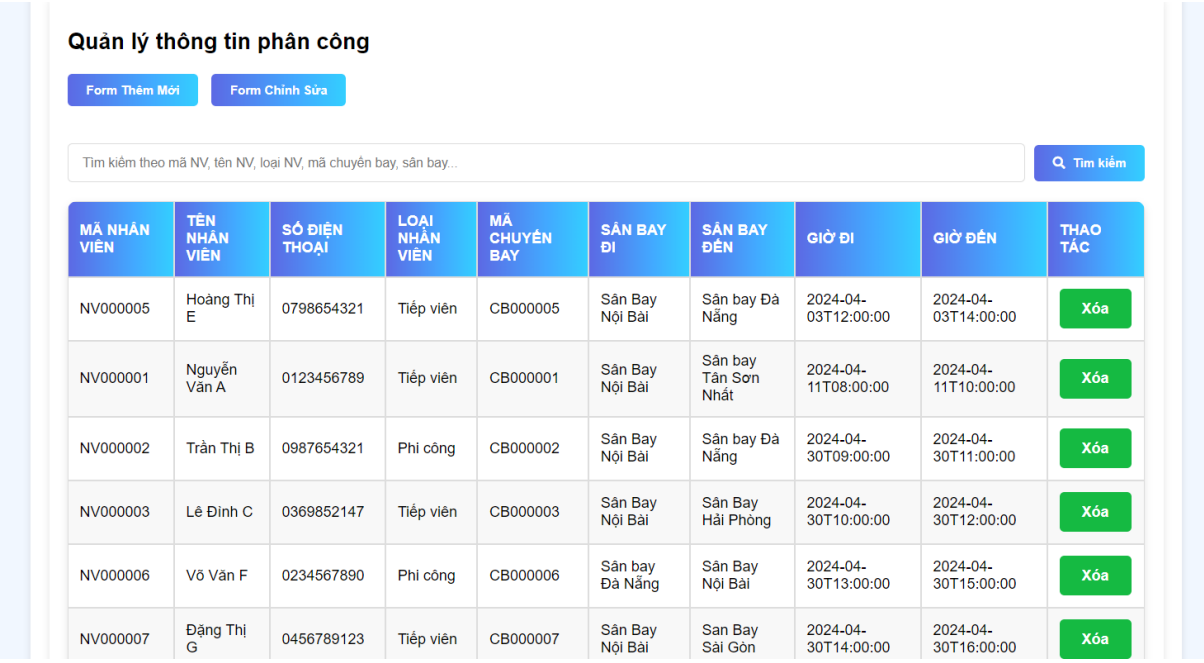
Phân tích chi tiết từng chức năng của các phương thức trong controller:

- Phương thức **themPhanCong** (Thêm phân công): Phương thức này nhận các tham số bao gồm mã nhân viên, mã chuyến bay và thời gian khởi hành của chuyến bay. Đầu tiên, thời gian khởi hành được chuyển thành đối tượng LocalDateTime để trích xuất ngày khởi hành dưới dạng LocalDate. Sau đó, phương thức gọi service phanCongService.themPhanCong để xử lý logic thêm phân công công việc cho nhân viên vào chuyến bay. Nếu có lỗi xảy ra trong quá trình thêm, hệ thống sẽ trả về thông báo lỗi cho người dùng.
- Phương thức **suaPhanCong** (Cập nhật phân công): Phương thức này thực hiện việc cập nhật phân công công việc của nhân viên. Tham số đầu vào gồm mã nhân viên, mã chuyến bay mới và thời gian khởi hành mới. Thời gian khởi hành mới cũng được chuyển thành LocalDateTime để lấy ngày chính xác. Sau khi kiểm tra và xử lý, phương thức sẽ gọi phanCongService.suaPhanCong để cập nhật phân công. Nếu gặp lỗi trong quá trình cập nhật, thông báo lỗi sẽ được trả về.
- Phương thức **xoaPhanCong** (Xóa phân công): Phương thức này giúp xóa phân công công việc của nhân viên đối với chuyến bay. Các tham số bao gồm mã nhân viên, mã chuyến bay và ngày khởi hành. Ngày khởi hành được chuyển thành LocalDate và sau đó gọi **phanCongService.xoaPhanCong** để thực hiện việc xóa phân công. Nếu có lỗi xảy ra, thông báo lỗi sẽ được trả về cho người dùng.

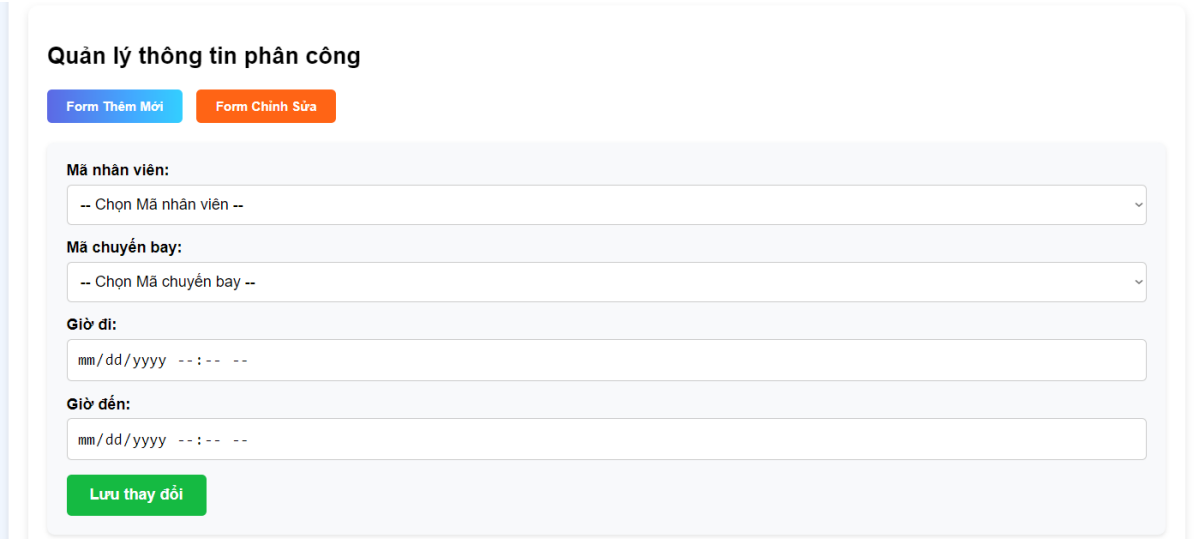
- Phương thức **getFlightDetailsForAssignment** (Lấy chi tiết chuyến bay cho phân công): Phương thức này lấy thông tin chi tiết về chuyến bay dựa trên mã chuyến bay được cung cấp. Phương thức sẽ gọi service **phanCongService.getFlightDetailsForAssignment** để truy xuất thông tin về chuyến bay. Nếu thành công, thông tin chuyến bay sẽ được trả về cho người dùng, nếu gặp lỗi, hệ thống sẽ trả về thông báo lỗi.

Cấu trúc và Quản lý Lỗi: Mỗi phương thức đều có cơ chế xử lý lỗi riêng biệt. Khi có lỗi xảy ra, một đối tượng Map sẽ được tạo ra với thông báo lỗi, sau đó trả về một phản hồi HTTP với mã lỗi tương ứng (thường là 500 Internal Server Error). Điều này giúp ứng dụng dễ dàng theo dõi và thông báo lại cho người dùng nếu có sự cố xảy ra trong quá trình xử lý yêu cầu.

Controller này đóng vai trò quan trọng trong việc quản lý và điều phối công việc phân công cho nhân viên, liên kết giữa các yêu cầu từ phía người dùng và các logic xử lý nghiệp vụ trong service layer. Bằng cách sử dụng Spring Framework và các annotation như @PostMapping và @LoginRequired, mã nguồn này dễ dàng duy trì và mở rộng, giúp cải thiện hiệu quả quản lý công việc tại sân bay.



Chương 5_hình 3.9-1. Phân công



Chương 5_hình 3.6-2. Form chỉnh sửa phân công

Lớp **PhanCong** trong ứng dụng quản lý sân bay đại diện cho một phân công công việc cho nhân viên trong hệ thống. Lớp này có ba thuộc tính chính: maNV (mã nhân viên), maChuyenBay (mã chuyến bay) và ngayDi (ngày khởi hành chuyến bay). Với @Entity, lớp PhanCong được định nghĩa là một thực thể JPA và ánh xạ trực tiếp với bảng PhanCong trong cơ sở dữ liệu. Các trường này được ánh xạ đến các cột tương ứng trong bảng, với maNV là khóa chính của bảng. Để khởi tạo đối tượng PhanCong, lớp này có một constructor mặc định và một constructor có tham số, cho phép người dùng truyền vào các giá trị cụ thể khi tạo đối tượng. Các phương thức getter và setter cung cấp khả năng truy cập và thay đổi các giá trị của các thuộc tính này. Phương thức getter trả về giá trị của các trường, trong khi phương thức setter cho phép thay đổi các giá trị sau khi đối tượng đã được khởi tạo.

Lớp PhanCong không chỉ là một thực thể dữ liệu đơn thuần mà còn giúp quản lý phân công công việc của nhân viên trong hệ thống. Khi một nhân viên được phân công làm việc trên một chuyến bay, hệ thống sẽ tạo ra một bản ghi trong bảng PhanCong với thông tin về nhân viên, chuyến bay và ngày khởi hành. Các bản ghi này sẽ giúp hệ thống theo dõi các phân công công việc, hỗ trợ việc lên lịch, theo dõi và quản lý nguồn lực (nhân viên và chuyến bay) trong quá trình hoạt động của sân bay.

Class **PhanCongRepository** sử dụng Spring Data JPA để tương tác với cơ sở dữ liệu. Các phương thức trong lớp này chủ yếu phục vụ việc tìm kiếm, cập nhật và xóa phân công công việc của nhân viên trên chuyến bay. Cụ thể, phương thức **existsByMaNVAndMaChuyenBayAndNgayDi** kiểm tra xem đã có phân công công việc cho nhân viên (maNV) trên chuyến bay (maChuyenBay) vào ngày

(ngayDi) cụ thể chưa. Nếu có, phương thức trả về true, ngược lại là false. Phương thức findByMaNV tìm kiếm phân công công việc của nhân viên dựa trên mã nhân viên (maNV), trả về một đối tượng Optional<PhanCong>, giúp xử lý trường hợp không tìm thấy bản ghi. Trong khi đó, phương thức updatePhanCong cập nhật thông tin phân công công việc của nhân viên, bao gồm chuyến bay và ngày đi, thông qua một câu truy vấn HQL (Hibernate Query Language). Cuối cùng, phương thức **deleteByMaNVAndMaChuyenBayAndNgayDi** xóa phân công của nhân viên theo mã nhân viên, mã chuyến bay và ngày đi.

Các phương thức trong lớp **PhanCongService** đều có cơ chế xử lý lỗi rõ ràng, trả về các phản hồi chi tiết giúp người dùng dễ dàng nhận biết tình trạng của các thao tác thực hiện. Ví dụ, khi thêm phân công mới, nếu phân công đã tồn tại, phương thức sẽ trả về một lỗi với thông điệp cụ thể để thông báo cho người dùng. Tương tự, khi sửa hoặc xóa phân công, nếu có lỗi xảy ra, hệ thống sẽ trả về thông báo lỗi với thông tin chi tiết. Phản hồi của các phương thức này được chuẩn hóa dưới dạng ResponseEntity, giúp dễ dàng kiểm tra tình trạng thành công hay thất bại của các thao tác từ phía người dùng.

Bảo mật và giao tiếp: Các phương thức trong **PhanCongService** đảm bảo bảo mật và tính toàn vẹn của hệ thống bằng cách sử dụng các cơ chế kiểm tra quyền truy cập. Ví dụ, chỉ những người dùng đã đăng nhập mới có thể thực hiện các thao tác thêm, sửa, xóa phân công. Các lớp giao tiếp với nhau một cách rõ ràng và hiệu quả: lớp **PhanCongRepository** xử lý các thao tác trực tiếp với cơ sở dữ liệu, lớp **PhanCongService** xử lý logic nghiệp vụ và bảo mật, còn lớp Controller (không có trong mã nguồn đã đưa) sẽ nhận các yêu cầu từ phía người dùng và trả về kết quả thông qua các phương thức trong **PhanCongService**.

CHƯƠNG 6. TRIỂN KHAI VÀ KIỂM THỬ

1. Phương pháp kiểm thử

Để đảm bảo hệ thống hoạt động chính xác và ổn định, dự án sẽ thực hiện các phương pháp kiểm thử sau:

- **Kiểm thử đơn vị (Unit Testing)**: Phương pháp này tập trung vào việc kiểm tra các module hoặc function riêng lẻ trong hệ thống. Mục tiêu của kiểm thử đơn vị là đảm bảo mỗi phần nhỏ của hệ thống hoạt động đúng như mong đợi trước khi tích hợp với các phần khác. Các test case sẽ được viết cho từng chức năng đơn lẻ, như việc xác minh các phép toán, chức năng xử lý dữ liệu, hoặc kiểm tra các hàm xử lý logic trong mỗi module.
- **Kiểm thử tích hợp (Integration Testing)**: Sau khi kiểm tra đơn vị, bước tiếp theo là kiểm tra sự tương tác giữa các module trong hệ thống. Kiểm thử tích hợp sẽ đảm bảo rằng các phần khác nhau của hệ thống có thể làm việc cùng nhau một cách hiệu quả. Các lỗi có thể xuất hiện khi một module này kết nối hoặc trao đổi dữ liệu với module khác sẽ được phát hiện và xử lý trong giai đoạn này.
- **Kiểm thử hệ thống (System Testing)**: Đây là giai đoạn kiểm tra toàn bộ hệ thống để xác minh rằng tất cả các thành phần của ứng dụng hoạt động đúng như dự định khi kết hợp lại với nhau. Kiểm thử hệ thống sẽ kiểm tra hệ thống trong môi trường tương tự như môi trường thực tế để đảm bảo tính đầy đủ và chính xác của các tính năng.
- **Kiểm thử chấp nhận (Acceptance Testing)**: Cuối cùng, kiểm thử chấp nhận là bước kiểm tra xem hệ thống có đáp ứng đầy đủ yêu cầu của người dùng hay không. Đây là giai đoạn quan trọng để xác nhận rằng sản phẩm cuối cùng phù hợp với kỳ vọng và yêu cầu mà khách hàng đã đề ra từ đầu.

2. Kết quả kiểm thử

Kết quả kiểm thử sẽ được ghi lại chi tiết để cung cấp một cái nhìn tổng quan về những gì đã đạt được và những vấn đề cần khắc phục. Các kết quả này sẽ được phân loại thành ưu điểm và nhược điểm của hệ thống.

Ưu điểm:

- **Trang Web cơ bản đáp ứng nhu cầu tối thiểu để quản lý sân bay**: Hệ thống đã thực hiện được các chức năng cơ bản như quản lý nhân viên, chuyến bay, và phân công công việc.
- **Giao diện dễ nhìn, đơn giản, dễ làm quen**: Giao diện người dùng đã được thiết kế dễ hiểu, dễ thao tác, giúp người dùng mới có thể sử dụng hệ thống mà không gặp nhiều khó khăn.
- **Các tính năng cơ bản đã hoạt động**: Các chức năng quan trọng như thêm, sửa, xóa nhân viên, chuyến bay, và phân công công việc đều đã hoạt động tốt.
- **Bước đầu có các biện pháp bảo mật**: Hệ thống đã áp dụng các biện pháp bảo mật cơ bản như xác thực người dùng, phân quyền truy cập, giúp bảo vệ thông tin nhạy cảm.

Nhược điểm:

- **Các API hoạt động chưa ổn định**: Một số API chưa đạt hiệu quả cao, có thể gặp lỗi hoặc hoạt động không ổn định khi xử lý yêu cầu đồng thời hoặc dữ liệu lớn.
- **Còn nhiều ràng buộc, quan hệ giữa các bảng chưa được xử lý**: Các mối quan hệ giữa các bảng trong cơ sở dữ liệu chưa được tối ưu hoặc xử lý đầy đủ, dẫn đến một số vấn đề khi thực hiện truy vấn phức tạp hoặc thay đổi dữ liệu.
- **Thiếu các công cụ tìm kiếm hữu ích cho người sử dụng, khả năng import/download dữ liệu**: Hệ thống chưa hỗ trợ tìm kiếm nâng cao hoặc khả năng nhập/xuất dữ liệu (import/export) một cách thuận tiện cho người sử dụng.
- **Chưa có nhiều tính năng: Thanh toán, định vị, ...**: Các tính năng phụ trợ như thanh toán online hoặc định vị chuyến bay chưa được triển khai, điều này có thể hạn chế khả năng sử dụng của người dùng.
- **Chưa áp dụng nhiều công nghệ**: Hệ thống chưa sử dụng một số công nghệ hiện đại hoặc tiên tiến, có thể giúp cải thiện hiệu suất và tính năng của hệ thống trong tương lai.

Những ưu điểm và nhược điểm này sẽ được xem xét và cải thiện trong các giai đoạn tiếp theo của dự án để nâng cao chất lượng và khả năng đáp ứng nhu cầu của người dùng.

CHƯƠNG 7. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Tổng kết

- Qua quá trình xây dựng và phát triển, chức năng quản lý chuyến bay trong dự án "Xây dựng website quản lý sân bay Wings Airport" đã hoàn thiện và đáp ứng được các yêu cầu đặt ra. Hệ thống được triển khai với nền tảng **Java Spring Boot**, tích hợp hệ cơ sở dữ liệu PHPmyadmin (MySQL), cùng các công nghệ hiện đại đảm bảo hiệu năng cao và bảo mật thông tin. Dự án đã chú trọng đến trải nghiệm người dùng, tối ưu hóa quy trình nghiệp vụ, và xây dựng nền tảng quản lý chuyến bay ổn định, có khả năng mở rộng trong tương lai.

2. Kết quả đạt được

- **Hệ thống hoàn chỉnh:** Chức năng quản lý chuyến bay được phát triển đầy đủ, bao gồm thêm, sửa, xóa và kiểm tra các mối liên kết với các chức năng khác như lịch bay, phân công, đặt chỗ,... Điều này đảm bảo tính nhất quán dữ liệu và giúp quản trị viên dễ dàng thao tác.
- **Kiểm tra và xử lý lỗi nghiệp vụ:** Hệ thống xử lý các trường hợp đặc biệt, như ngăn chặn việc thêm chuyến bay có sân bay đi và đến giống nhau hoặc xóa chuyến bay đang được liên kết trong các bảng khác. Điều này giúp bảo vệ tính toàn vẹn dữ liệu.
- **Công nghệ hiện đại:** Sử dụng **Spring Data JPA** để thao tác với cơ sở dữ liệu, tích hợp API RESTful cho phép giao tiếp hiệu quả giữa các thành phần.
- **Hiệu năng và bảo mật:** Hệ thống được thiết kế đảm bảo hiệu năng ổn định, có khả năng xử lý đồng thời nhiều yêu cầu và bảo vệ dữ liệu người dùng với các phương pháp kiểm tra logic nghiệp vụ chặt chẽ.
- **Khả năng triển khai thực tế:** Chức năng quản lý chuyến bay hoạt động trơn tru, phù hợp để áp dụng vào thực tế, giúp tối ưu hóa quy trình vận hành và cải thiện chất lượng dịch vụ trong lĩnh vực hàng không.

3. Khó khăn và thách thức

Bên cạnh những thành công đã đạt được, hệ thống quản lý sân bay "Wings Airport" vẫn tồn tại một số hạn chế cần khắc phục:

- **Chức năng chưa đầy đủ:** Một số tính năng nâng cao chưa được phát triển do giới hạn về thời gian và nguồn lực, ví dụ như tích hợp thanh toán trực tuyến hoặc tính năng quản lý chương trình khách hàng thân thiết.
- **Xử lý dữ liệu mô phỏng:** Hiện tại, hệ thống mới được thử nghiệm dựa trên dữ liệu mô phỏng, chưa có tính thực tế cao, dẫn đến chưa thể kiểm tra hết các trường hợp sử dụng trong môi trường thực tế.

- **Tối ưu hóa hiệu suất:** Một số luồng xử lý nghiệp vụ trong hệ thống chưa đạt mức tối ưu hoàn toàn, đặc biệt khi số lượng yêu cầu tăng cao hoặc hệ thống phải xử lý đồng thời nhiều dữ liệu phức tạp.

4. Hướng phát triển

4.1. Mở rộng tính năng

- Phát triển thêm các chức năng nâng cao như tích hợp thanh toán trực tuyến, quản lý chương trình khách hàng thân thiết, hoặc cung cấp dịch vụ đặt vé cho khách hàng theo gói.
- Ứng dụng trí tuệ nhân tạo (AI) vào việc dự đoán nhu cầu chuyến bay, tối ưu hóa lịch trình bay và phân bổ nguồn lực hiệu quả.
- Đồng bộ hóa dữ liệu với các hệ thống khác trong sân bay, ví dụ như hệ thống check-in, hệ thống giám sát hành lý, và an ninh để xây dựng hệ sinh thái thống nhất.

4.2. Tối ưu hiệu suất

- Tái cấu trúc các luồng xử lý chính trong hệ thống nhằm cải thiện tốc độ phản hồi và khả năng xử lý đồng thời.
- Sử dụng các công nghệ cache như **Redis** hoặc **Ehcache** để giảm tải cho cơ sở dữ liệu trong các tác vụ đọc dữ liệu thường xuyên.
- Triển khai kiến trúc microservices với Java Spring Boot để tách biệt các chức năng chính, từ đó dễ dàng mở rộng và bảo trì từng phần.

4.3. Nâng cao trải nghiệm người dùng

- Tiếp tục cải tiến giao diện với thiết kế thân thiện, dễ sử dụng trên các nền tảng khác nhau, từ máy tính để bàn đến thiết bị di động.
- Tích hợp các API để cung cấp thông báo thời gian thực về chuyến bay, tình trạng đặt vé, và thông tin check-in cho hành khách.
- Sử dụng **Thymeleaf** kết hợp với **AJAX** để làm mới nội dung trang web mà không cần tải lại toàn bộ trang, mang lại trải nghiệm mượt mà hơn.

TÀI LIỆU THAM KHẢO

1. TopDev. Java là gì? Giới thiệu và hướng dẫn cài đặt ngôn ngữ Java chi tiết [Internet]. TopDev. 2020 [cited 2024 Nov 16]. Available from: https://topdev.vn/blog/tong-quan-ve-ngon-ngu-lap-trinh-java/

2. Blog T. Maven Apache [Internet]. TopDev. 2019 [cited 2024 Nov 15]. Available from: https://topdev.vn/blog/maven-apache/

3. FPT C ty C phần B lẻ K. Spring boot là gì? Cách sử dụng Spring boot làm dự án như thế nào nhanh nhất? [Internet]. [cited 2024 Nov 16]. Available from: https://fptshop.com.vn/tin-tuc/danh-gia/spring-boot-la-gi-172291

4. MySQL là gì? Cơ chế hoạt động và cách thức cài đặt MySQL [Internet]. 2024 [cited 2024 Nov 15]. Available from: https://fptcloud.com/mysql-la-gi/

5. Blog T. HTML là gì? Các tag thông dụng của HTML dành cho lập trình viên [Internet]. TopDev. 2021 [cited 2024 Nov 16]. Available from: https://topdev.vn/blog/html-la-gi/

6. Docker Compose will BLOW your MIND!! (a tutorial) - YouTube [Internet]. [cited 2024 Nov 16]. Available from: https://www.youtube.com/watch?v=DM65_JyGxCo&t=71s

7. Airport NBI. viags.vn. [cited 2024 Nov 16]. Noi Bai International Airport. Available from: https://noibaiairport.vn/vi/arrivals

8. Airline Management System using Java Object Oriented Programming (OOP) (Part 1) [Internet]. [cited 2024 Nov 16]. Available from: https://www.youtube.com/watch?v=ymVSs0RCY1w&list=PL-cxzMmn1xXGU8memZZmt7-m0L5raoDRq

9. Vietnam Airlines [Internet]. [cited 2024 Nov 16]. Làm Thủ Tục Trực Tuyến. Available from: http://www.vietnamairlines.com/vn/vi/travel-information/check-in/online-check-in

PHÂN CÔNG NHIỆM VỤ

Vị trí	Thành viên
Project manager	Nguyễn Hữu Nam (chính), Vũ Long Nhật
Front-end	Trần Thị Bình (chính), Nguyễn Hữu Nam, Lê Anh Tú
Back-end	Nguyễn Hữu Nam (chính), Trần Thị Bình, Trần Thành Duy, Lê Anh Tú
Database	Trần Thị Bình (chính), Nguyễn Hữu Nam, Trần ThànhDuy
Report	Vũ Long Nhật (chính), Trần Thành Duy
Slide	Trần Thành Duy (chính), Vũ Long Nhật
Thuyết trình	Trần Thị Bình, Nguyễn Hữu Nam, Vũ Long Nhật