TRƯỜNG ĐẠI HỌC KINH TẾ

**KHOA THỐNG KÊ – TIN HỌC**



**BÁO CÁO THỰC TẬP NGHỀ NGHIỆP**

**NGÀNH HỆ THỐNG THÔNG TIN QUẢN LÝ**

**CHUYÊN NGÀNH QUẢN TRỊ HỆ THỐNG THÔNG TIN**

**ĐỀ TÀI: THIẾT KẾ CRUD CHO**

**HỆ THỐNG QUẢN LÝ CHI TIÊU**

Sinh viên thực hiện : **Trần Thị Diệu Hiền**

Lớp : **46K21.2**

Đơn vị thực tập : **TMA Solutions Bình Định**

Cán bộ hướng dẫn : **Bùi Đức Lực**

Giảng viên hướng dẫn : **Thạc sĩ Nguyễn Thành Thủy**

***Đà Nẵng, 8/202******3***

**NHẬN XÉT CỦA ĐƠN VỊ THỰC TẬP**

Họ và tên sinh viên: Lớp:

Khoa Thống kê – Tin học, Trường Đại học Kinh tế, Đại học Đà Nẵng

Thực tập từ ngày:.…./.…./2023 đến ngày: .…./.…./2023

Tên đơn vị thực tập:

Địa chỉ:

Số điện thoại liên hệ:

Họ tên cán bộ hướng dẫn:

**Sau quá trình thực tập của sinh viên tại đơn vị, chúng tôi có một số đánh giá như sau:**

| **STT** | **Nội dung đánh giá** | **Rất không tốt** | **Không tốt** | **Bình thường** | **Tốt** | **Rất tốt** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Về thái độ, ý thức, đạo đức và việc tuân thủ các quy định và văn hóa đơn vị thực tập | ☐ | ☐ | ☐ | ☐ | ☐ |
| 2 | Kiến thức chuyên môn | ☐ | ☐ | ☐ | ☐ | ☐ |
| 3 | Khả năng hòa nhập, thích nghi và tác phong nghề nghiệp | ☐ | ☐ | ☐ | ☐ | ☐ |
| 4 | Trách nhiệm, sáng tạo trong công việc | ☐ | ☐ | ☐ | ☐ | ☐ |

(*Anh/chị vui lòng đánh dấu X vào ô tương ứng với năng lực của sinh viên*)

**Ý kiến nhận xét và đề xuất** (*Nhằm nâng cao chất lượng đào tạo, Nhà trường rất mong muốn nhận thêm những ý kiến khác từ quý doanh nghiệp*):

……….., ngày .......tháng ......năm 2022

**Xác nhận của đơn vị thực tập**

# LỜI CẢM ƠN

Báo cáo thực tập nghề nghiệp chuyên ngành Quản trị hệ thống thông tin với đề tài “Thiết kế CRUD cho Hệ thống Quản lý chi tiêu” là thành quả của sự cố gắng học hỏi và sự giúp đỡ của thầy cô, Mentor.

Đầu tiên, em xin chân thành tri ân đến Công ty TNHH Giải pháp Phần mềm Tường Minh Bình Định, đặc biệt là anh Bùi Đức Lực – mentor hướng dẫn trực tiếp em trong kỳ thực tập vừa qua, cũng như các anh chị đã tận tình giúp đỡ và chia sẻ các kiến thức giúp em phát triển hơn. Nhờ có các anh chị, em đã nhận thấy mình đã tiến bộ không chỉ về kiến thức chuyên môn mà còn về kỹ năng làm việc nhóm, thái độ chuyên nghiệp và khả năng giải quyết vấn đề. Mọi người đã giúp em nhận ra tiềm năng bản thân và truyền đạt cho em những kinh nghiệm quý báu từ thực tế ngành nghề.

Xin cảm ơn quý lãnh đạo, ban giám hiệu cùng toàn thể các thầy cô giảng viên trường Đại học Kinh Tế - Đại học Đà Nẵng, khoa Thống kê Tin học. Đặc biệt là thầy Nguyễn Thành Thủy đã tạo điều kiện cho chúng em hoàn thành tốt kì thực tập nghề nghiệp của mình.

Một lần nữa, em xin chân thành cảm ơn tất cả quý vị đã đồng hành cùng em trong hành trình thực tập nghề nghiệp này. Hy vọng rằng những mối quan hệ quý báu và những kinh nghiệm thú vị này sẽ giúp em tiếp tục phát triển và duy trì trong tương lai.

**Họ và tên sinh viên**

**Trần Thị Diệu Hiền**

# LỜI CAM ĐOAN

Em xin cam đoan báo cáo đề tài “Thiết kế CRUD cho Hệ thống Quản lý chi tiêu”là thành quả nghiên cứu độc lập trong thời gian em thực tập tại Công ty TMA Solutions Bình Định dưới sự hướng dẫn tận tình của người hướng dẫn và sự hỗ trợ từ các anh chị trong team DC20 và giảng viên hướng dẫn thầy Nguyễn Thành Thủy.

Nội dung và báo cáo thực tập nghề nghiệp là kết quả mà em đã nỗ lực nghiên cứu, học tập và tìm hiểu trong quá trình thực tập tại công ty và đồng thời học hỏi từ các tài liệu tham khảo mà đơn vị thực tập cung cấp. Em xin chịu hoàn toàn trách nhiệm, kỷ luật của bộ môn và nhà trường đề ra nếu như có vấn đề xảy ra.

**Họ và tên sinh viên**

**Trần Thị Diệu Hiền**

# MỤC LỤC

[LỜI CẢM ƠN i](#_Toc142837428)

[LỜI CAM ĐOAN ii](#_Toc142837429)

[MỤC LỤC iii](#_Toc142837430)

[DANH MỤC HÌNH ẢNH iv](#_Toc142837431)

[DANH MỤC BẢNG BIỂU v](#_Toc142837432)

[DANH MỤC CÁC TỪ VIẾT TẮT vi](#_Toc142837433)

[LỜI MỞ ĐẦU 1](#_Toc142837434)

[1.1. Giới thiệu tổng quát về doanh nghiệp thực tập 3](#_Toc142837435)

[1.1.1. Tổng quan về TMA Solutions 3](#_Toc142837436)

[1.1.2. Tầm nhìn, sứ mệnh 4](#_Toc142837437)

[1.2. Tổng quan về Back-end Developer 5](#_Toc142837438)

[2.1. Ngôn ngữ lập trình Java 7](#_Toc142837439)

[2.1.1. Tổng quan về Java 7](#_Toc142837440)

[2.1.2. Đặc điểm của ngôn ngữ lập trình Java 7](#_Toc142837441)

[2.2. Mục 2.2 11](#_Toc142837442)

[3.1. Mục 3.1… 12](#_Toc142837443)

[3.2. Mục 3.2… 12](#_Toc142837444)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 13](#_Toc142837445)

[TÀI LIỆU THAM KHẢO 14](#_Toc142837446)

[CHECK LIST CỦA BÁO CÁO 15](#_Toc142837447)

[PHỤ LỤC 16](#_Toc142837448)

# DANH MỤC HÌNH ẢNH

[Hình 1.1 Logo công ty TMA Solutions 2](#_heading=h.2xcytpi)

[Hình 4.1 Kiến trúc của mô hình 5](#_heading=h.vx1227)

# DANH MỤC BẢNG BIỂU

Bảng 2.1 Bảng các kiểu dữ liệu nguyên thủy

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| API | Application Programming Interface |
| URL | Uniform Resource Locator |
| CRUD | Create, Read, Update, Delete |
| HTTP | Hyper Text Transfer Protocol |
| REST | REpresentational State Transfer |
| OOP | Object Oriented Programming |
| JWT | JSON Web Tokens |

# LỜI MỞ ĐẦU

1. **Mục tiêu của đề tài**

* Nắm được kiến thức về Java core, lập trình OOP
* Nắm được kiến thức về Database, Spring framework
* Tạo ra một hệ thống có các chức năng CRUD với ý tưởng: Trong thời đại hiện nay, việc quản lý chi tiêu đang trở thành thách thức với nhiều người kiếm tiền nhưng không biết cách sử dụng một cách hiệu quả. Vì vậy, hệ thống quản lý chi tiêu được tạo ra nhằm giúp người dùng kiểm soát nguồn thu và chi, đồng thời thiết lập mục tiêu tiết kiệm. Việc này không chỉ giúp họ quản lý tài chính hiệu quả hơn mà còn xây dựng thói quen tài chính tích cực.

1. **Đối tượng và phạm vi nghiên cứu**

* Nghiên cứu thực hiện đề tài trên ngôn ngữ lập trình Java và các nền tảng nguồn mở là Spring Boot Framework, kiến trúc JPA, KeyCloak,... từ đó xây dựng lên các API cơ bản của một trang web.

1. **Phương pháp nghiên cứu**

* Nghiên cứu và tìm hiểu về ngôn ngữ lập trình Java, các Framework như Spring Boot, Hibernate qua các kênh kiến thức.
* Nghiên cứu mô hình kiến trúc xây dựng web: Spring boot, JPA
* Sử dụng Docker và Keycloak cung cấp giải pháp quản lý xác thực và quyền truy cập cho ứng dụng
* Thực hiện trên nền tảng Restful API để tạo ra chức năng CRUD để trang web

hoạt động.

* Việc test các chức năng API dựa trên nền tảng ứng dụng Postman.

1. **Kết cấu của đề tài**

Đề tài được tổ chức gồm phần mở đầu, 4 chương nội dung và phần kết luận. Mở đầu

**Chương 1:** Giới thiệu về Công ty TMA Solutions

**Chương 2:** Tổng quan về công nghệ và công cụ sử dụng

**Chương 3:** Phân tích và thiết kế hệ thống

**Chương 4:** Triển khai và kết quả

Kết luận và hướng phát triển

1. **GIỚI THIỆU CÔNG TY**

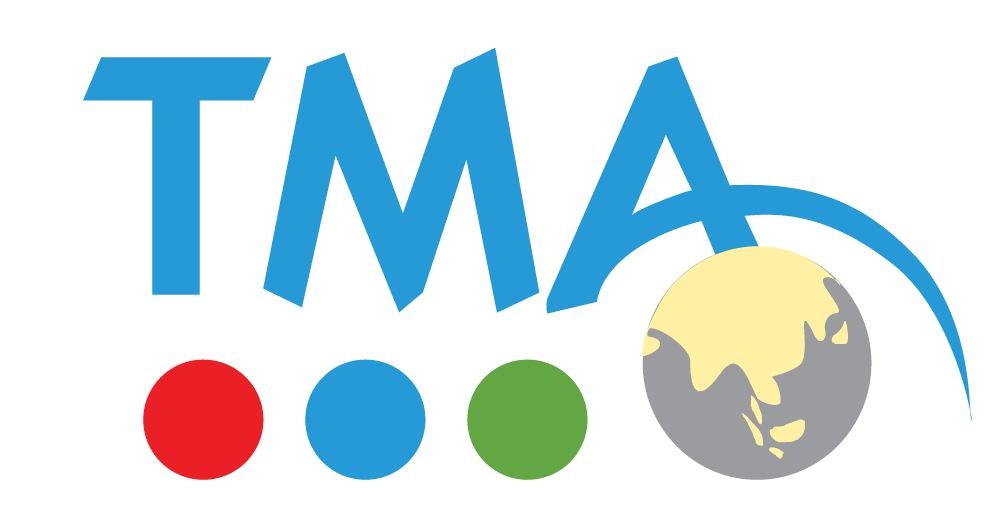
**TỔNG QUAN VỀ CƠ SỞ LÝ THUYẾT**

## Giới thiệu tổng quát về doanh nghiệp thực tập

### Tổng quan về TMA Solutions

TMA là tập đoàn công nghệ hàng đầu Việt Nam, 25 năm phát triển và triển khai các giải pháp phần mềm cho 100 khách hàng từ 30 nước trên Thế giới.

* Thành lập 10/1997, là công ty tư nhân, 100% vốn Việt Nam.
* TMA hiện có 7 chi nhánh ở Việt Nam (6 ở Tp Hồ Chí Minh và 1 ở Tp Quy Nhơn) cùng 6 chi nhánh ở nước ngoài ( Mỹ, Úc, Canada, Nhật, Singapope).
* Đội ngũ nhân viên tại TMA Bình Định được đánh giá là có trình độ chuyên môn cao, năng động và sáng tạo với 4000 kỹ sư đang công tác và làm việc tại TMA.
* Thị trường:
* Bắc Mỹ
* Châu Âu
* Nhật Bản & Châu Á
* Úc
* Các trung tâm:
* Trung tâm Phát triển Phần mềm (Software Development Center)
* Xưởng Phần mềm (Software Factory)
* Trung tâm Khoa học Dữ liệu (Data Science Lab)
* Học viện Công nghệ (TMA Academy)
* Trung tâm Nghiên cứu và Chuyển giao Công nghệ (R&D Center)
* TMA chuyên cung cấp các dịch vụ và giải pháp công nghệ thông tin đa dạng như phát triển phần mềm, thiết kế ứng dụng di động, dịch vụ đám mây, trí tuệ nhân tạo, Internet of Things (IoT), giải pháp Blockchain và nhiều lĩnh vực công nghệ khác. Khách hàng là những công ty hàng đầu thế giới (Nortel, IBM, Alcatel-Lucent, Juniper Networks, Flextronics, Genband, NTT, Toshiba, Samsung, NEC).



Hình 1.1 Logo công ty TMA Solutions

### Tầm nhìn, sứ mệnh

Tầm nhìn: Trở thành 1 trong những công ty hàng đầu về cung cấp giải pháp phần mềm tại Việt Nam và các nước trong khu vực.

Sứ mệnh: Tôn trọng và mang đến cho khách hàng những sản phẩm giải pháp phần mềm tốt nhất với chi phí hợp lý nhất. Đồng thời xây dựng mối quan hệ tin cậy, uy tín, hợp tác cùng phát triển với các đối tác trong lĩnh vực công nghệ thông tin.

* + 1. **Giá trị cốt lõi**

Trong nhiều năm qua mặc dù phải đối mặt với nhiều thách thức, nhưng TMA Bình Định đã luôn chứng minh được khả năng làm việc, giữ vững cam kết chất lượng đối với khách hàng. Để có được những thành công đó, TMA đã và đang duy trì phát triển những giá trị cốt lõi:

* Sự cam kết (Commitment)
* Sự trung thực (Honesty)
* Sự tôn trọng (Respect)
  + 1. **Cơ cấu tổ chức**

TMA Bình Định đang được phát triển mạnh mẽ dưới sự dẫn dắt và lãnh đạo của đội ngũ giàu kinh nghiệm:

* Chủ tịch: TS. Nguyễn Hữu Lệ
* Giám đốc điều hành: Bà Ánh Bùi
* Phó tổng giám đốc, giám đốc tài chính: Bà Dương Phạm
* Phó giám đốc nhân sự và hành chính: Bà Uyên Phạm

## Tổng quan về Back-end Developer

* + 1. Mô tả công việc

Back-end Developer Java là một lập trình viên thiết kế, phát triển và quản lý các ứng dụng và phần mềm dựa trên Java. Với hầu hết các tổ chức lớn sử dụng Java để triển khai các hệ thống phần mềm và dịch vụ phụ trợ, Back-end Developer Java là một trong những công việc được tìm kiếm nhiều nhất hiện nay.

Nhà phát triển chịu trách nhiệm về một số nhiệm vụ liên quan đến Java trong suốt vòng đời phát triển phần mềm, từ ý tưởng và thiết kế đến thử nghiệm. Nhà phát triển được yêu cầu tạo các giải pháp thông tin người dùng thông qua việc phát triển, triển khai và bảo trì các thành phần và giao diện dựa trên Java.

* + 1. Mức lương

Mức lương của Back-end Developer Java tại Việt Nam cũng có sự biến đổi dựa trên nhiều yếu tố như kinh nghiệm, vị trí địa lý, kích thước công ty, và cấp bậc trong công việc. Dưới đây là một phạm vi tham khảo về mức lương của Back-end Developer Java tại Việt Nam hiện nay:

* Nhập môn / Junior: Mức lương thường nằm trong khoảng 10 triệu - 20 triệu VND một tháng.
* Trung bình / Middle: Mức lương trung bình cho Back-end Developer Java ở cấp trung bình thường nằm trong khoảng 20 triệu - 40 triệu VND một tháng.
* Cao cấp / Senior: Các nhà phát triển Java cấp cao cường độ kinh nghiệm và kỹ năng cao thường có mức lương từ 40 triệu trở lên.
  + 1. Lộ trình thăng tiến

Lộ trình thăng tiến cho một Back-end Developer Java tại Việt Nam có thể thay đổi dựa trên công ty, ngành công nghệ thông tin và sự phát triển cá nhân của mỗi người. Dưới đây là một lộ trình tham khảo mà một Back-end Developer Java có thể theo đuổi:

1. Junior Developer :

Phát triển kiến thức cơ bản về lập trình Java và các khái niệm quan trọng. Làm quen với các khung làm việc (framework) phổ biến như Spring hoặc Hibernate. Tham gia vào dự án nhỏ và học hỏi từ những người đồng nghiệp kinh nghiệm.

2. Mid-level Developer :

Xây dựng kỹ năng và kinh nghiệm làm việc với các hệ thống phức tạp hơn. Tham gia vào việc thiết kế kiến trúc hệ thống và đảm bảo tích hợp hiệu quả. Đảm bảo mã nguồn ổn định, tối ưu và tuân thủ các quy tắc lập trình.

3. Senior Developer :

Trở thành chuyên gia trong việc xây dựng và tối ưu hóa ứng dụng sử dụng Java. Dẫn dắt dự án, hướng dẫn các nhà phát triển junior và mid-level, chia sẻ kinh nghiệm. Tham gia vào quá trình đưa ra quyết định kiến trúc và lựa chọn công nghệ.

4. Lead Developer hoặc Tech Lead :

Đứng đầu dự án, đảm bảo việc phát triển theo hướng đúng và đạt được các mục tiêu. Tham gia vào việc định hình chiến lược phát triển và sự phát triển của đội. Đóng góp vào việc xây dựng các chuẩn quy trình phát triển và kiểm tra chất lượng.

5. Architect hoặc Principal Developer :

Trở thành chuyên gia kiến thức về kiến trúc hệ thống, đảm bảo tích hợp và mở rộng các hệ thống phức tạp. Tham gia vào việc thiết kế và xây dựng các hệ thống lớn và có độ phức tạp cao. Dẫn dắt đội công nghệ, đóng góp vào việc lập kế hoạch và phát triển dự án.

1. **CƠ SỞ LÝ THUYẾT**

## Ngôn ngữ lập trình Java

### Tổng quan về Java



Hình 2.1 Logo ngôn ngữ lập trình Java

Java là một ngôn ngữ lập trình hướng đối tượng có ứng dụng đa dạng trong việc phát triển phần mềm, trang web, game và ứng dụng trên các thiết bị di động. Ban đầu được phát triển vào năm 1994, và sau đó năm 2010, Oracle mua lại nền tảng này từ Sun Microsystems.

Tiêu chí quan trọng trong việc tạo ra Java là "Viết (code) một lần, thực thi khắp nơi" (Write Once, Run Anywhere - WORA). Điều này có nghĩa là chương trình phần mềm được viết bằng Java có khả năng hoạt động trên mọi nền tảng khác nhau, bất kể là máy tính cá nhân, máy chủ, thiết bị di động hoặc các thiết bị khác. Điều này thực hiện thông qua việc sử dụng môi trường thực thi Java (Java Virtual Machine - JVM), giúp đảm bảo khả năng tương thích trên các nền tảng khác nhau, miễn là môi trường thực thi thích hợp được cung cấp

### Đặc điểm của ngôn ngữ lập trình Java

Java là ngôn ngữ lập trình được phát triển từ ngôn ngữ lập trình C/C++. Nó kế thừa, phát huy các thế mạnh của ngôn ngữ C/C++ và lược bỏ đi các cú pháp phức tạp của C/C++. Ngôn ngữ lập trình Java có một số đặc trưng tiêu biểu: hướng đối tượng, độc lập phần cứng và hệ điều hành, ngôn ngữ thông dịch, đa luồng, an toàn, bảo mật,…

**a) Ngôn ngữ hướng đối tượng**

Java là ngôn ngữ lập trình thuần hướng đối tượng, mọi chương trình viết  trên Java đều phải được xây dựng dựa trên các đối tượng. Nếu trong C/C++ ta có  thể tạo ra hàm (chương trình con không gắn với đối tượng nào) thì trong Java ta  chỉ có thể tạo ra các phương thức (chương trình con gắn liền với một lớp cụ thể).  Trong Java không cho phép các đối tượng có tính năng đa kế thừa mà được thay  thế bằng các giao diện (interface).

**b) Độc lập phần cứng và hệ điều hành**

Một chương trình viết bằng ngôn ngữ Java có thể chạy tốt ở nhiều môi trường khác nhau. Gọi là khả năng “cross-platform”. Khả năng độc lập phần cứng và hệ điều hành được thể hiện ở 2 cấp độ là cấp độ mã nguồn và cấp độ nhị phân.

**c) Ngôn ngữ thông dịch**

Java là một ngôn ngữ lập trình vừa biên dịch vừa thông dịch. Chương trình  nguồn viết bằng ngôn ngữ lập trình Java có đuôi \*.java và được biên dịch thành  tập tin có đuôi \*.class sau đó được trình thông dịch thông dịch thành mã máy.

**d) Đa luồng**

Java hỗ trợ lập trình đa tiến trình (multi-thread) để thực thi các công việc  đồng thời. Đồng thời cũng cung cấp giải pháp đồng bộ giữa các tiến trình (giải  pháp sử dụng priority…).

* + 1. ***Java Core***

Java Core là phần cơ bản và nền tảng của ngôn ngữ lập trình Java. Nó bao gồm những khái niệm, cú pháp và thành phần cơ bản mà mọi lập trình viên Java cần phải hiểu để xây dựng và phát triển các ứng dụng Java. Dưới đây là một số khía cạnh quan trọng của Java Core:

* Cú pháp và Cấu Trúc: Java Core định nghĩa cú pháp của ngôn ngữ, bao gồm cách viết khai báo biến, định nghĩa lớp, phương thức và cấu trúc điều khiển như if-else, vòng lặp, và switch-case.
* Kiểu Dữ Liệu: Java cung cấp các kiểu dữ liệu như int, double, char, boolean và cả kiểu dữ liệu đối tượng như String và Arrays.
* Biến và Hằng Số: Biến là vị trí lưu trữ dữ liệu, có thể thay đổi giá trị. Hằng số là giá trị không thay đổi.
* Phương Thức và Lớp: Java hỗ trợ lập trình hướng đối tượng. Bạn có thể định nghĩa phương thức (method) để thực hiện các tác vụ cụ thể, và lớp (class) để đóng gói dữ liệu và logic liên quan.
* Xử Lý Chuỗi: Java cung cấp các phương thức cho việc làm việc với chuỗi (String), như nối chuỗi, cắt chuỗi, và so sánh chuỗi.
* Xử Lý Ngoại Lệ: Java hỗ trợ xử lý ngoại lệ (exception handling), cho phép bạn kiểm soát và xử lý các tình huống ngoại lệ một cách kiểm soát.
* Kế Thừa và Ghi Đè: Kế thừa cho phép lớp con thừa hưởng thuộc tính và phương thức của lớp cha. Ghi đè cho phép bạn định nghĩa lại phương thức trong lớp con.
* Xử Lý File: Java cung cấp các lớp và phương thức để thực hiện xử lý file và thư mục.
* Lập Trình Đồng Thời (Multithreading): Java hỗ trợ lập trình đồng thời, cho phép bạn thực hiện nhiều tác vụ cùng lúc thông qua việc tạo và quản lý các luồng (thread).
* Cơ Chế Quản Lý Bộ Nhớ: Java sử dụng bộ quản lý bộ nhớ tự động (Garbage Collector) để giúp bạn quản lý bộ nhớ một cách hiệu quả.

Java Core là nền tảng cho việc xây dựng ứng dụng Java, bất kể là ứng dụng máy tính cá nhân, ứng dụng di động, ứng dụng web hoặc các hệ thống lớn hơn. Hiểu về Java Core giúp bạn có nền tảng mạnh mẽ để tiến xa hơn trong việc học và phát triển ứng dụng Java.

* + 1. **Java OOP**

Java OOP (Object-Oriented Programming) là một phương pháp lập trình mà trong đó mọi thứ được coi là các đối tượng, và các đối tượng này tương tác với nhau để thực hiện các chức năng và tạo ra các ứng dụng. Java là một ngôn ngữ lập trình hướng đối tượng mạnh mẽ và phổ biến, nên việc hiểu về Java OOP là rất quan trọng. Dưới đây là một tóm tắt về Java OOP:

* Lập Trình Hướng Đối Tượng (OOP): OOP là một phương pháp lập trình dựa trên khái niệm về đối tượng. Đối tượng là một thực thể có thuộc tính (properties) và hành vi (behaviors).
* Lớp và Đối Tượng: Trong Java OOP, bạn định nghĩa các lớp để tạo các đối tượng. Lớp là một bản thiết kế cho đối tượng, định nghĩa các thuộc tính và phương thức mà đối tượng cụ thể sẽ có.
* Encapsulation (Đóng Gói): Encapsulation là việc ẩn thông tin riêng tư của đối tượng và chỉ cho phép truy cập thông qua các phương thức công khai. Điều này giúp bảo vệ dữ liệu và logic của đối tượng.
* Inheritance (Kế Thừa): Kế thừa cho phép một lớp con (subclass) kế thừa thuộc tính và phương thức từ một lớp cha (superclass). Điều này giúp tạo ra các lớp con chia sẻ các đặc điểm của lớp cha và mở rộng chức năng.
* Polymorphism (Đa Hình): Polymorphism cho phép các đối tượng cùng kiểu có thể thực hiện các hành động khác nhau dựa trên cách mà chúng được định nghĩa.
* Abstraction (Trừu Tượng): Trừu tượng là việc tạo ra các lớp trừu tượng (abstract class) hoặc các giao diện (interface) để định nghĩa các thuộc tính và phương thức mà các lớp con cần phải triển khai.
* Đối Tượng Trong Java: Trong Java, mọi thứ đều là đối tượng, bao gồm cả kiểu dữ liệu cơ bản như int, double. Mỗi đối tượng có trạng thái(state) và hành vi(behavior).

Java OOP giúp tạo ra mã nguồn có tổ chức, dễ đọc, dễ bảo trì và dễ mở rộng. Nó cũng thúc đẩy tái sử dụng mã, giúp giảm độ phức tạp của chương trình và tạo ra các ứng dụng mạnh mẽ và linh hoạt.

## Spring Boot Framework


Hình 2.3.1 Hệ quản trị cơ sở dữ liệu PostgreSQL



Hình 2.3.1 Hệ quản trị cơ sở dữ liệu PostgreSQL

Spring Boot là một dự án phát triển  bởi JAV (ngôn ngữ Java) trong hệ sinh thái Spring Framework. Nó giúp đơn giản  hóa quá trình lập trình một ứng dụng với Spring, chỉ tập trung vào việc phát triển  business cho ứng dụng.

**2.3.2. Tính năng nổi bật của Spring Boot**

* Giúp xây dựng ứng dụng Spring độc lập (stand-alone) bao gồm REST API,  WebSocket, Web, Streaming …
* Giúp giảm tối ưu công đoạn cấu hình và không yêu cầu cấu hình thông qua  XML.
* Có thể nhúng trực tiếp Tomcat, Jetty, Undertow… do đó không cần phải  triển khai tập tin WAR.
* Tích hợp các loại module khác nhau trong các loại Spring như Spring Data,  Spring MVC, Spring Security, Spring JDBC, Spring ORM…
* Các starter dependency làm cho việc thêm các hỗ trợ phù hợp cho project  giúp cấu hình ban đầu nhanh hơn.
* Có thể dễ dàng phát triển các ứng dụng Spring dựa trên các loại ngôn ngữ khác như Java hoặc Groovy.
* Spring Boot có thể tự động cấu hình Spring mỗi khi cần thiết.
* Hỗ trợ nhiều IDE như Spring Tool Suite, IntelliJ IDEA, NetBeans.
* Không sinh code cấu hình và không yêu cầu phải cấu hình bằng XML.

**2.4. Spring MVC**

**2.4.1. Spring MVC là gì?**

Spring MVC là một module nằm trong Spring Framework, mục đích của  nó là xây dựng các ứng dụng web dựa trên mô hình MVC (Model - View - Controller) đồng thời cũng cấp loose coupling giữa các thành phần.

**2.4.2. Tính năng nổi bật của Spring MVC**

* Spring MVC cho phép bạn cấu hình các thành phần bằng cách sử dụng annotations hoặc XML configuration, tạo sự linh hoạt trong việc quản lý ứng dụng.
* Spring MVC sử dụng DispatcherServlet để điều phối các yêu cầu từ người dùng tới các controller tương ứng, giúp quản lý và phân phối luồng điều hướng.
* Spring MVC cho phép bạn định nghĩa các controller bằng cách sử dụng annotations hoặc implement interface, xử lý các yêu cầu từ người dùng và trả về dữ liệu hoặc view tương ứng.
* Spring MVC hỗ trợ View Resolver để giúp tìm kiếm và render các view dựa trên tên và đường dẫn, giúp tách biệt logic xử lý và hiển thị.
* Spring MVC có thể sử dụng để xây dựng các RESTful web services thông qua sử dụng annotations và các cơ chế hỗ trợ.

**2.5. JPA(Java Persistent API)**

**2.5.1. Giới thiệu về JPA**

Java Persistence API (JPA) là một giao diện lập trình ứng dụng (API) trong Java cho việc quản lý dữ liệu trong các ứng dụng sử dụng ngôn ngữ Java. JPA được định nghĩa như một phần của chuẩn Java EE (Java Platform, Enterprise Edition) và là một phần quan trọng của quá trình lập trình hướng đối tượng và quản lý dữ liệu trong các ứng dụng.

Dưới đây là một số điểm nổi bật về JPA:

**1. Quản lý Đối tượng-Quan hệ (ORM):**

* JPA cho phép ánh xạ các đối tượng Java vào cơ sở dữ liệu một cách tự động, giúp đơn giản hóa việc lưu trữ và truy xuất dữ liệu.
* Thay vì viết các truy vấn SQL, bạn tạo các đối tượng Java và JPA sẽ thực hiện tất cả các thao tác truy vấn dữ liệu phía sau.

**2. Khả năng Đa dạng về Cơ sở dữ liệu:**

* JPA làm cho ứng dụng có khả năng tương thích với nhiều hệ quản trị cơ sở dữ liệu khác nhau như MySQL, PostgreSQL, Oracle, Microsoft SQL Server và nhiều hơn nữa.
* Bạn có thể viết mã một lần và chạy trên nhiều loại cơ sở dữ liệu khác nhau mà không cần thay đổi mã nguồn chính.

**3. Quản lý Chu kỳ Đời (Entity Lifecycle Management):**

* JPA giúp quản lý chu kỳ đời của các đối tượng, từ khi chúng được tạo ra, sửa đổi, đến khi chúng bị xóa.
* Bạn có thể xác định các hành động cần thực hiện trong mỗi giai đoạn của chu kỳ đời của đối tượng.

**4. Sử dụng Annotations hoặc XML:**

* JPA cho phép bạn sử dụng các Annotations để xác định ánh xạ giữa đối tượng Java và cơ sở dữ liệu.
* Bạn cũng có thể sử dụng cấu hình XML để thực hiện ánh xạ.

JPA giúp tạo ra một lớp trừu tượng giữa ứng dụng và cơ sở dữ liệu, giúp đơn giản hóa việc quản lý dữ liệu và tối ưu hóa quá trình phát triển ứng dụng sử dụng ngôn ngữ Java.

**2.6. RESTful web service**

**2.6.1. Web service là gì? Top of Form**

RESTful web service là một kiểu thiết kế và triển khai các dịch vụ web (web service) dựa trên kiến trúc Representational State Transfer (REST). RESTful web service sử dụng các nguyên tắc và quy tắc chuẩn để tạo ra các API (Application Programming Interface) có khả năng tương tác và trao đổi dữ liệu với các ứng dụng khác qua internet.

**2.6.2. Nguyên tắc REST:**

* RESTful web service tuân thủ các nguyên tắc của REST, bao gồm việc sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thao tác với tài nguyên.
* Mỗi tài nguyên được xác định bằng một định danh duy nhất, thường là URL (Uniform Resource Locator).

**2.6.3. Cơ chế Trạng thái và Dữ liệu Đại diện (Representation):**

* RESTful web service sử dụng các trạng thái của tài nguyên (Resource State) để thực hiện các hoạt động khác nhau.
* Dữ liệu trao đổi qua RESTful API thường được đại diện bằng các định dạng như JSON hoặc XML.

**2.6.4. Tính Dễ dàng Sử dụng:**

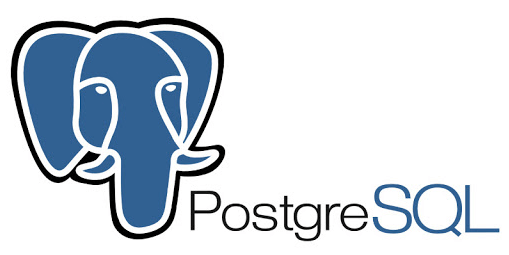
* RESTful web service được coi là dễ dàng sử dụng và hiểu, vì nó sử dụng các phương thức và định dạng dữ liệu phổ biến trong việc giao tiếp qua HTTP.

**2.6.5. Độc lập nền tảng:**

- RESTful web service không yêu cầu sử dụng ngôn ngữ hay công nghệ cụ thể. Điều này đảm bảo tính độc lập nền tảng và khả năng tương thích với nhiều ngôn ngữ lập trình khác nhau.

RESTful web service thường được sử dụng để xây dựng các dịch vụ web đa dạng như ứng dụng di động, ứng dụng web, tích hợp hệ thống, và nhiều ứng dụng khác, giúp tạo ra các giao tiếp linh hoạt và tiêu chuẩn giữa các ứng dụng khác nhau.**Top of Form**

## PostgreSQL



Hình 2.2 Hệ quản trị cơ sở dữ liệu PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở mạnh mẽ và phổ biến. Được viết bằng ngôn ngữ C và có sự hỗ trợ từ cộng đồng mạnh mẽ, PostgreSQL cung cấp các tính năng và khả năng cho việc lưu trữ, truy vấn và quản lý dữ liệu.

Dưới đây là một số điểm nổi bật về PostgreSQL:

**2.7.1. Đặc điểm quan trọng:**

* PostgreSQL hỗ trợ kiểu dữ liệu đa dạng bao gồm số nguyên, số thực, chuỗi ký tự, JSON, mảng và nhiều kiểu dữ liệu tùy chỉnh.
* Cung cấp khả năng thực hiện các thao tác phức tạp như JOIN, SUBQUERY, WINDOW FUNCTION, và CTE (Common Table Expressions).
* Hỗ trợ chuẩn SQL rất tốt, đảm bảo tính tương thích và dễ dàng di chuyển từ hệ quản trị cơ sở dữ liệu khác.

**2.7.2. An toàn và Bảo mật:**

* Hệ thống quản lý quyền truy cập mạnh mẽ cho phép quản lý chính xác quyền truy cập của người dùng và nhóm người dùng.
* Hỗ trợ mã hóa dữ liệu thông qua SSL và GSSAPI để đảm bảo an toàn trên kênh truyền thông.

**2.7.3. Khả năng Mở Rộng:**

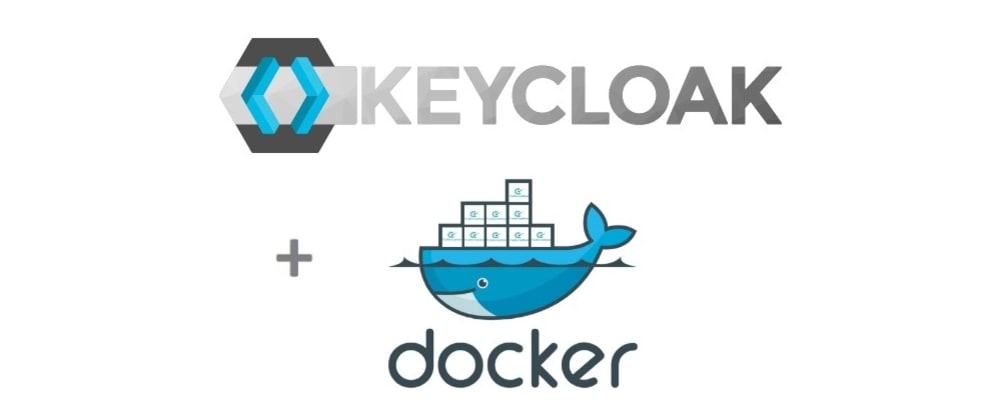
* Hỗ trợ cơ chế REPLICA cho việc sao chép và đồng bộ dữ liệu giữa các máy chủ.
* Cung cấp các phương pháp tối ưu hóa hiệu suất như Indexing, Partitioning và Giảm thiểu Deadlocks.

**2.7.4. Cộng đồng và Mở Rộng:**

* PostgreSQL có một cộng đồng lớn và nhiệt tình, cung cấp hỗ trợ, cập nhật và giải quyết vấn đề liên quan.
* Một loạt các phần mở rộng (extensions) có sẵn để mở rộng khả năng của PostgreSQL như hỗ trợ JSONB, hình ảnh, âm thanh, và nhiều loại dữ liệu khác.

PostgreSQL rất phù hợp cho các dự án yêu cầu cơ sở dữ liệu mạnh mẽ, an toàn và tùy chỉnh. Nó được sử dụng rộng rãi trong các ứng dụng kinh doanh, dự án phần mềm, trang web lớn và hệ thống phức tạp.

## Keycloak và Docker



Hình 2.2 Keycloak và Docker

* + 1. **Keycloak:**
* Quản lý Xác Thực và Quyền Truy Cập: Keycloak cung cấp một hệ thống quản lý xác thực mạnh mẽ, giúp bạn xác minh danh tính người dùng và quản lý quyền truy cập của họ vào các phần khác nhau của ứng dụng
* Single Sign-On (SSO): Keycloak hỗ trợ tính năng Single Sign-On, cho phép người dùng đăng nhập một lần và truy cập vào nhiều ứng dụng khác nhau mà không cần phải đăng nhập lại.
* Bảo Mật Cho Ứng Dụng: Keycloak cung cấp các tiêu chuẩn bảo mật như OpenID Connect và OAuth 2.0, giúp tạo ra các ứng dụng an toàn và bảo mật.
* Tích Hợp Dễ Dàng: Keycloak có khả năng tích hợp với các ứng dụng web, di động và API thông qua các giao thức và tiêu chuẩn quen thuộc.
  + 1. **Đóng gói ứng dụng Keycloak trong Docker**
* Docker cho phép bạn đóng gói ứng dụng cùng với tất cả các phụ thuộc và cấu hình cần thiết vào một container duy nhất. Điều này giúp đảm bảo rằng ứng dụng sẽ chạy đúng cách trên môi trường phát triển và sản phẩm.
* Docker cho phép tạo ra các container nhẹ và cô lập, giúp tăng hiệu suất của ứng dụng và tối ưu hóa sử dụng tài nguyên.
* Khi kết hợp Keycloak và Docker, bạn có thể xây dựng và triển khai các ứng dụng an toàn, bảo mật và dễ dàng quản lý trên các môi trường khác nhau một cách hiệu quả.

1. **PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

## Phân tích yêu cầu đề tài

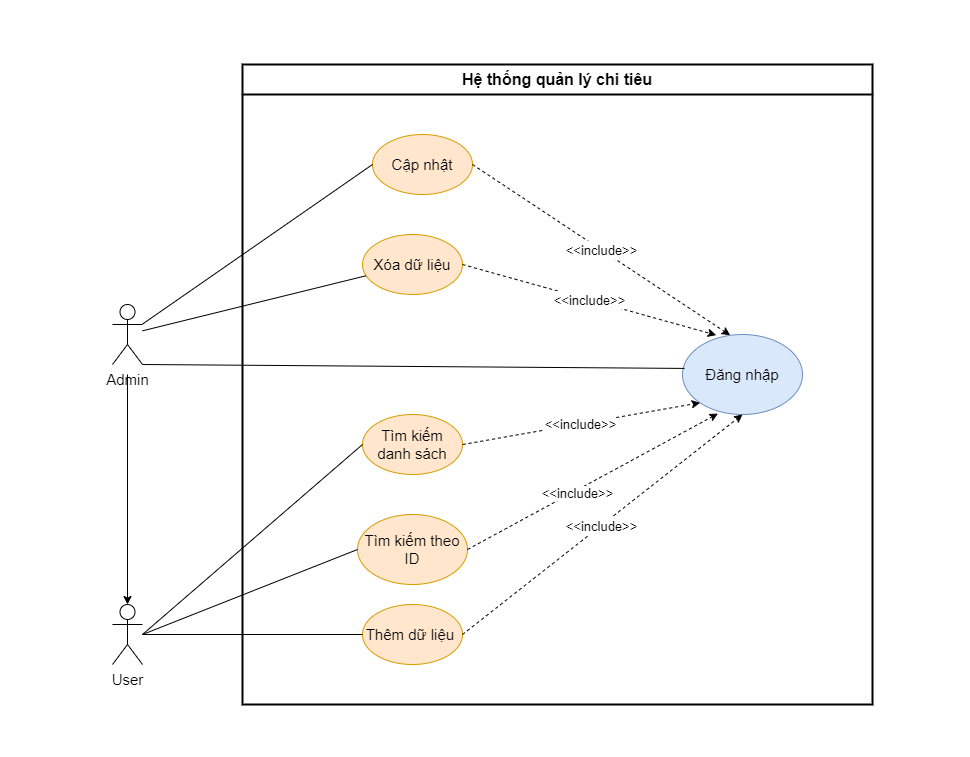
Yêu cầu về thiết kế chức năng tối ưu trải nghiệm của người dùng

* Chức năng đơn giản, thân thiện , dễ dàng sử dụng
* Đảm bảo các tính năng cơ bản của một hệ thống quản lý chi tiêu như xác thực người dùng, phân quyền người dùng, thêm, cập nhật, xoá, tìm kiếm.
  + 1. **Phân tích các chức năng có trong hệ thống**

|  |  |  |
| --- | --- | --- |
| ID | Feature Name | Decription |
| 1 | Đăng nhập | Xác thực tên đăng nhập, mật khẩu |
| 2 | Phân quyền người dùng | Phân quyền người dùng có thể sử dụng chức năng nào của hệ thống |
| 3 | Tìm kiếm danh sách | Người dùng được phân quyền có thể xem danh sách của một Lớp |
| 4 | Tìm kiếm theo Id | Người dùng được phân quyền có thể xem thông tin đầy đủ theo từng đối tượng |
| 5 | Thêm dữ liệu | Người dùng được phân quyền có thể thêm một đối tượng |
| 6 | Cập nhật | Người dùng được phân quyền có thể cập nhật từng đối tượng |
| 7 | Xóa một dữ liệu | Người dùng có thể xóa đối tượng |

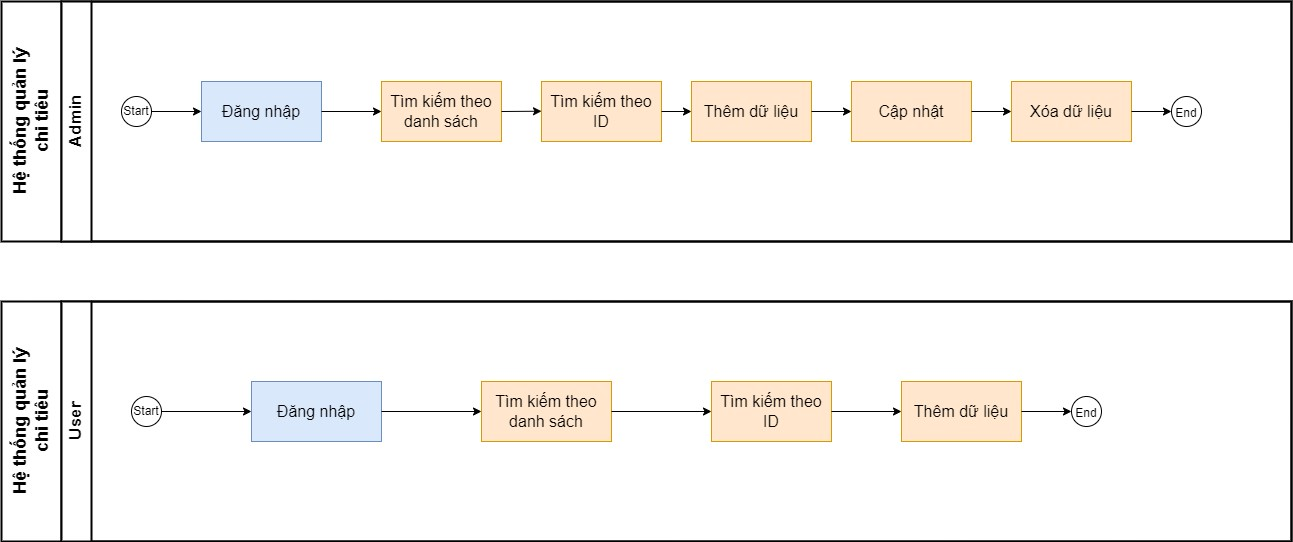
*Bảng 3.1 Bảng các chức năng có trong hệ thống*

* + 1. UseCase hệ thống



*Hình 3.1. UseCase hệ thống*

* + 1. Workflow hệ thống



*Hình 3.2. Workflow hệ thống*

* 1. **Thiết kế hệ thống**
     1. **Tác nhân (Actor) và thực thể có trong hệ thống**

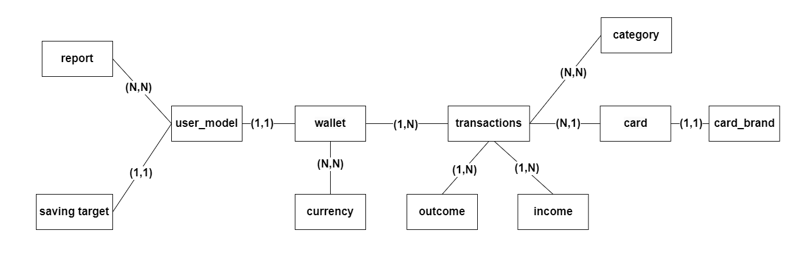
**3.2.1.1. Tác nhân**

* + - Admin
    - User

**3.2.1.2. Thực thể**

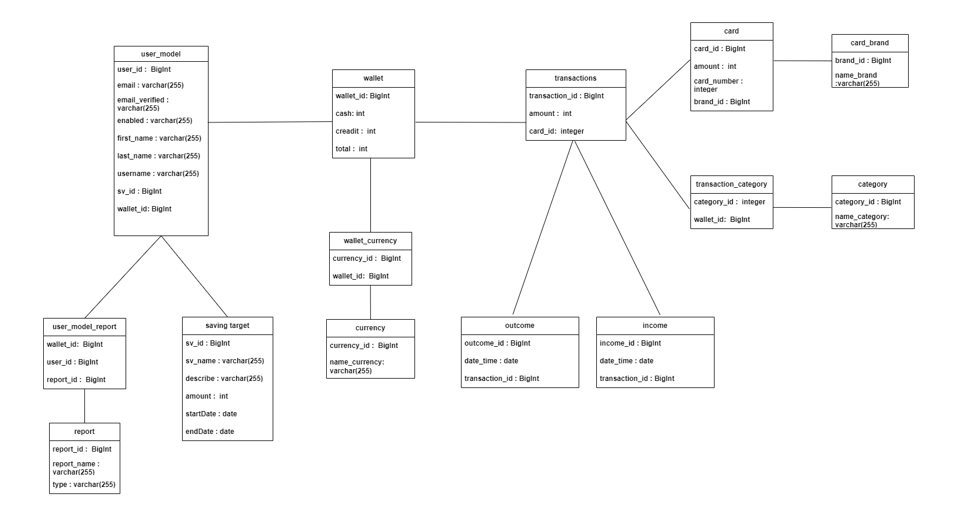
* Giao dịch (Transactions)
* Thu nhập (Income)
* Chi tiêu (Outcome)
* Loại thu chi (Category)
* Thẻ thanh toán (Card)
* Ngân hàng (Card Band)
* Ví (Wallet)
* Loại tiền tệ (Currency)
* Báo cáo (Report)
* Mục tiêu tiết kiệm ( Saving Tagert)
* Người dùng (User Model)

**3.2.1.3. Mối quan hệ giữa các thực thể**

****

*Hình 3.3. Bảng mối quan hệ giữa các thực thể*

**3.2.1.4. Các Class của hệ thống**

****

*Hình 3.4. Các Class của hệ thống*

**CHƯƠNG 4. TRIỂN KHAI VÀ KẾT QUẢ**

**4.1. Triển khai**

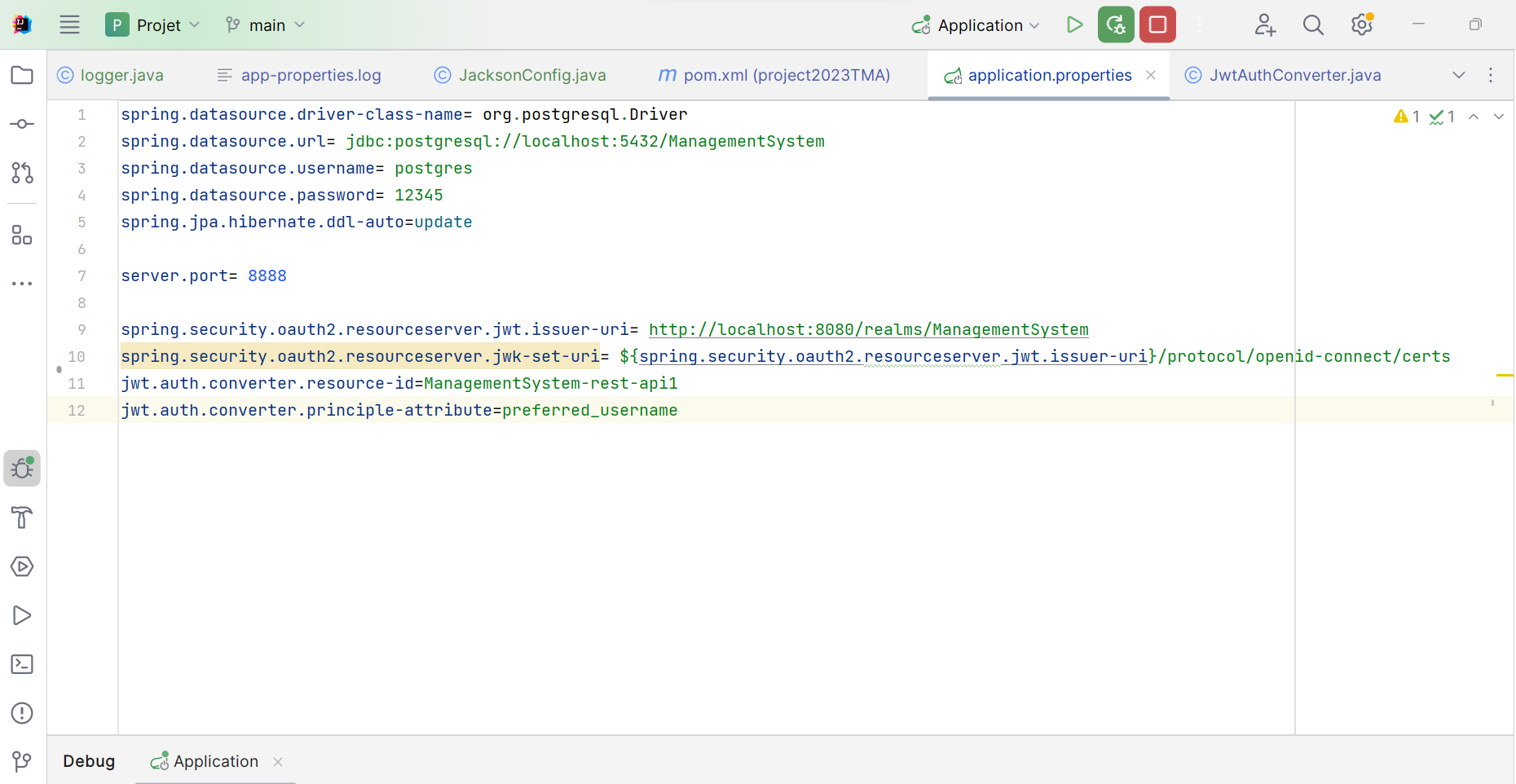
**4.1.1. Cài đặt các công cụ**

* Ngôn ngữ lập trình: Java.
* Thư viện hỗ trợ: Spring Boot.
* Cơ sở dữ liệu: PostgreSQL
* Công cụ lập trình: IntelliJ IDEA Ultimate 2023
* Hệ thống quản lý danh tính và quyền truy cập: Keycloak
* Môi trường đóng gói ứng dụng: Docker
* Lưu code: Github
* Test API: Postman

**4.1.2. Cài đặt các thư viện Spring boot Framework**

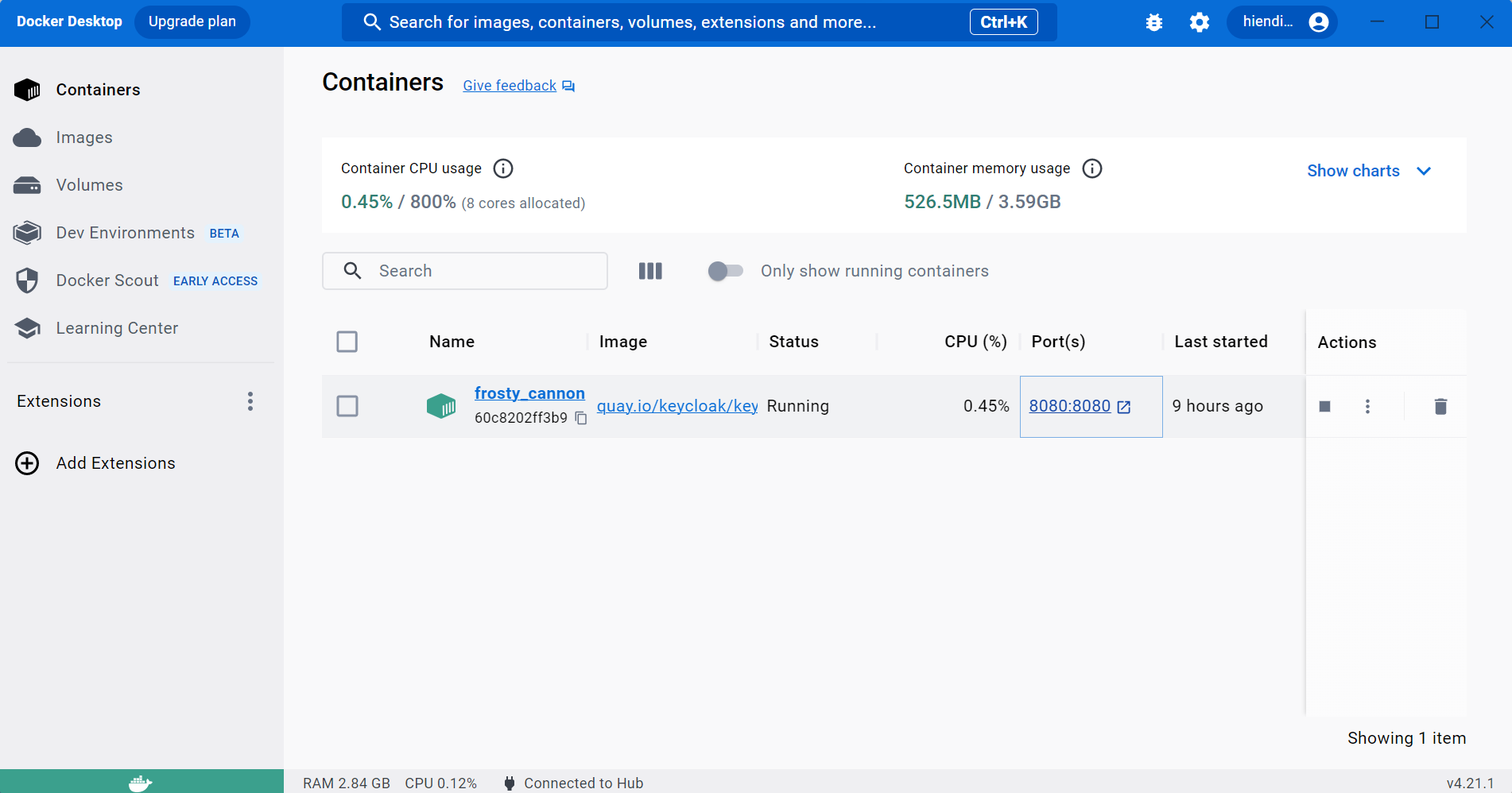
* spring-boot-starter-oauth2-resource-server: Thư viện hỗ trợ tạo một máy chủ tài nguyên OAuth2, cho phép ứng dụng được bảo mật bằng cách sử dụng OAuth2.
* spring-boot-starter-security: Thư viện cung cấp tích hợp Spring Security, giúp quản lý bảo mật và xác thực.
* spring-boot-starter-web: Thư viện cung cấp tích hợp Spring Web, giúp xây dựng các RESTful API.
* spring-security-test: Thư viện dành cho việc kiểm thử các tính năng liên quan đến Spring Security trong các test case.
* lombok: Thư viện Lombok giúp giảm bớt mã boilerplate (các đoạn mã lặp đi lặp lại) bằng cách tự động sinh các getter, setter, constructor và các phương thức khác cho các lớp.
* spring-boot-starter-data-jpa: Thư viện hỗ trợ tích hợp Spring Data JPA cho quản lý cơ sở dữ liệu.
* postgresql: Thư viện JDBC để kết nối với cơ sở dữ liệu PostgreSQL.
* spring-boot-starter-log4j2: Thư viện cung cấp tích hợp Log4j 2 cho việc ghi log.
* jackson-datatype-hibernate,javax.persistence-api,jackson-datatype-hibernate5: Thư viện này giúp xử lý dữ liệu JSON và tương tác với cơ sở dữ liệu thông qua Java trong các dự án phát triển ứng dụng.

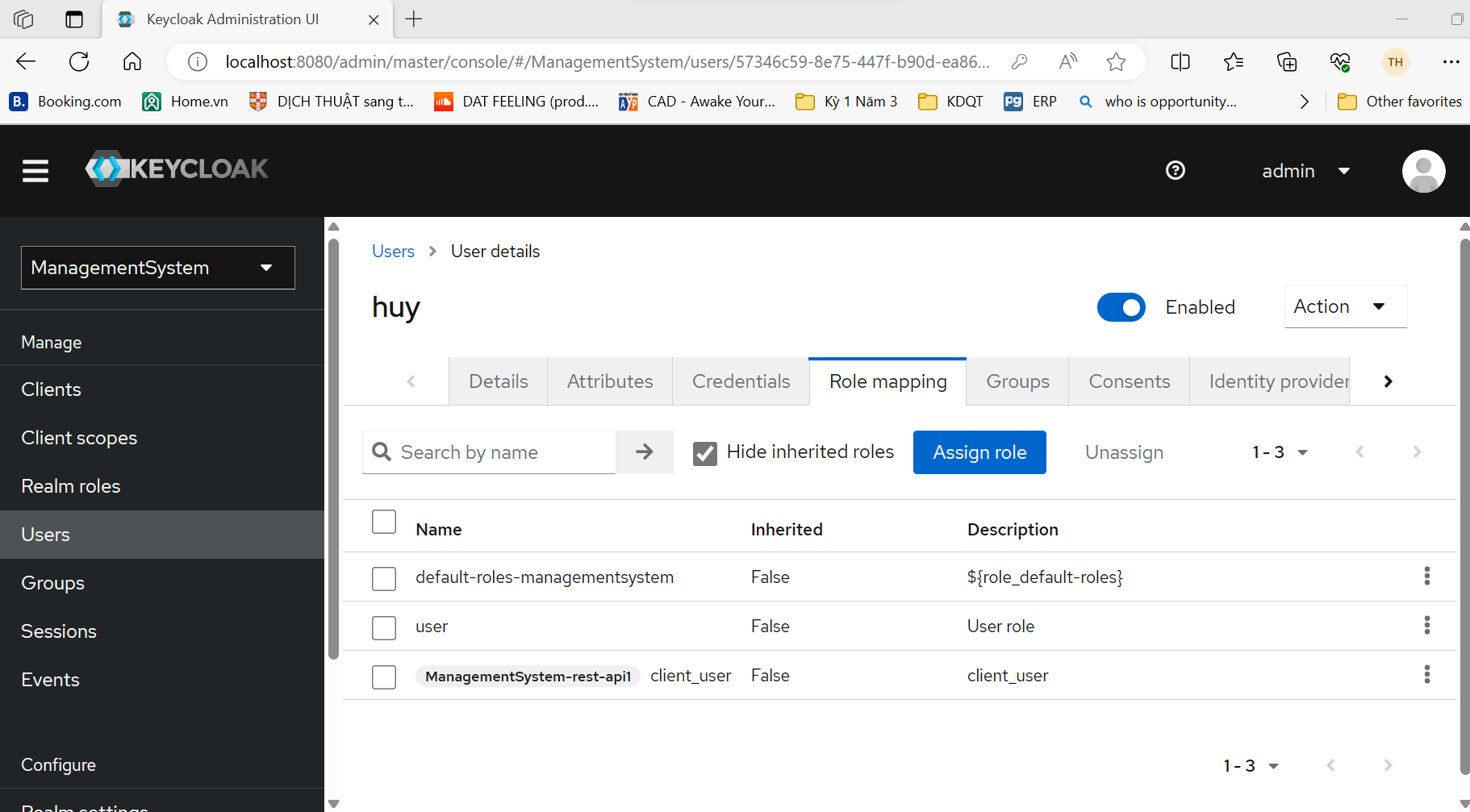
**4.1.3. Kết nối Database từ** **PostgreSQL**

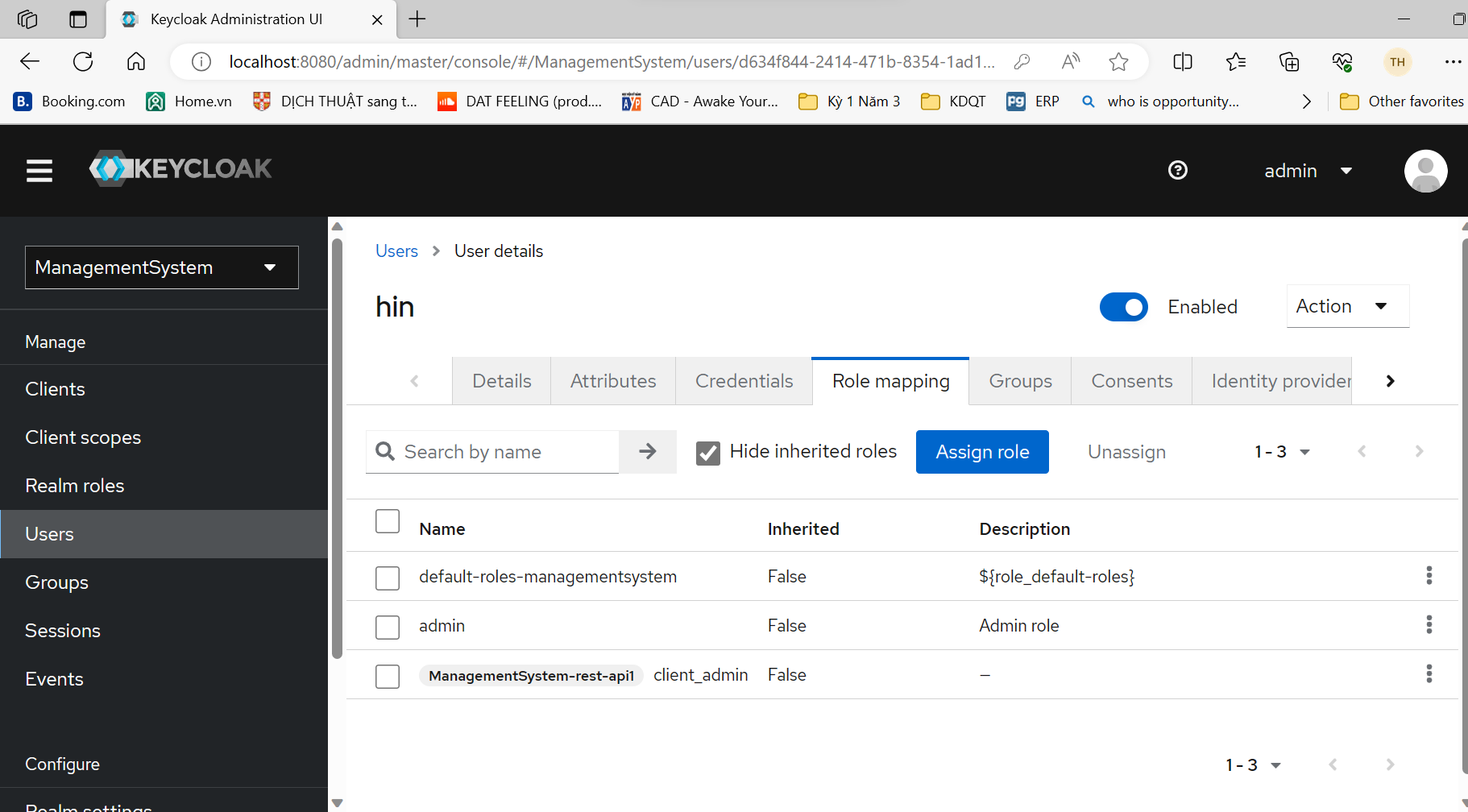


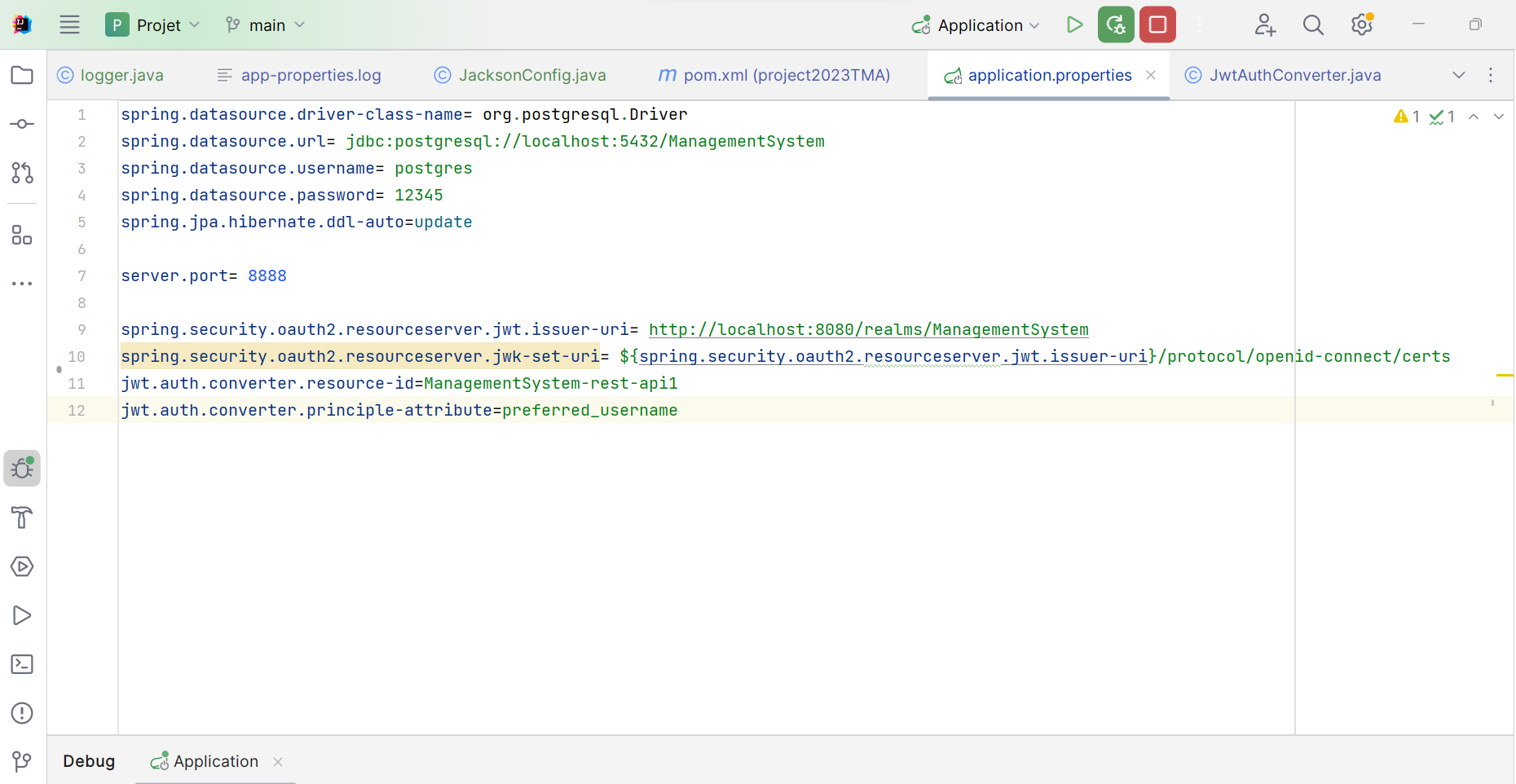
*Hình 4.1. Kết nối với PostgreSQL*

**4.1.4. Kết nối với Keycloak**

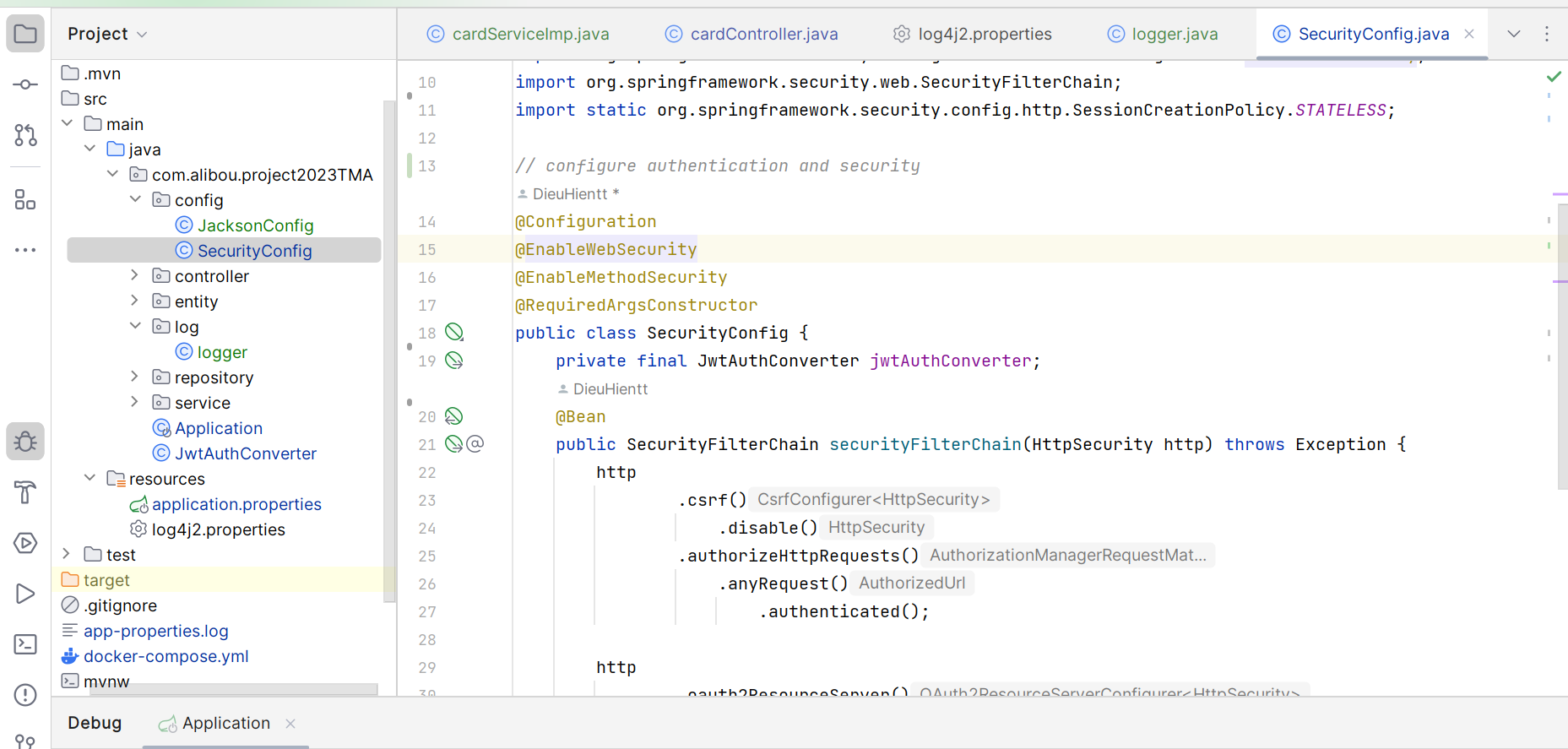
****

****

****

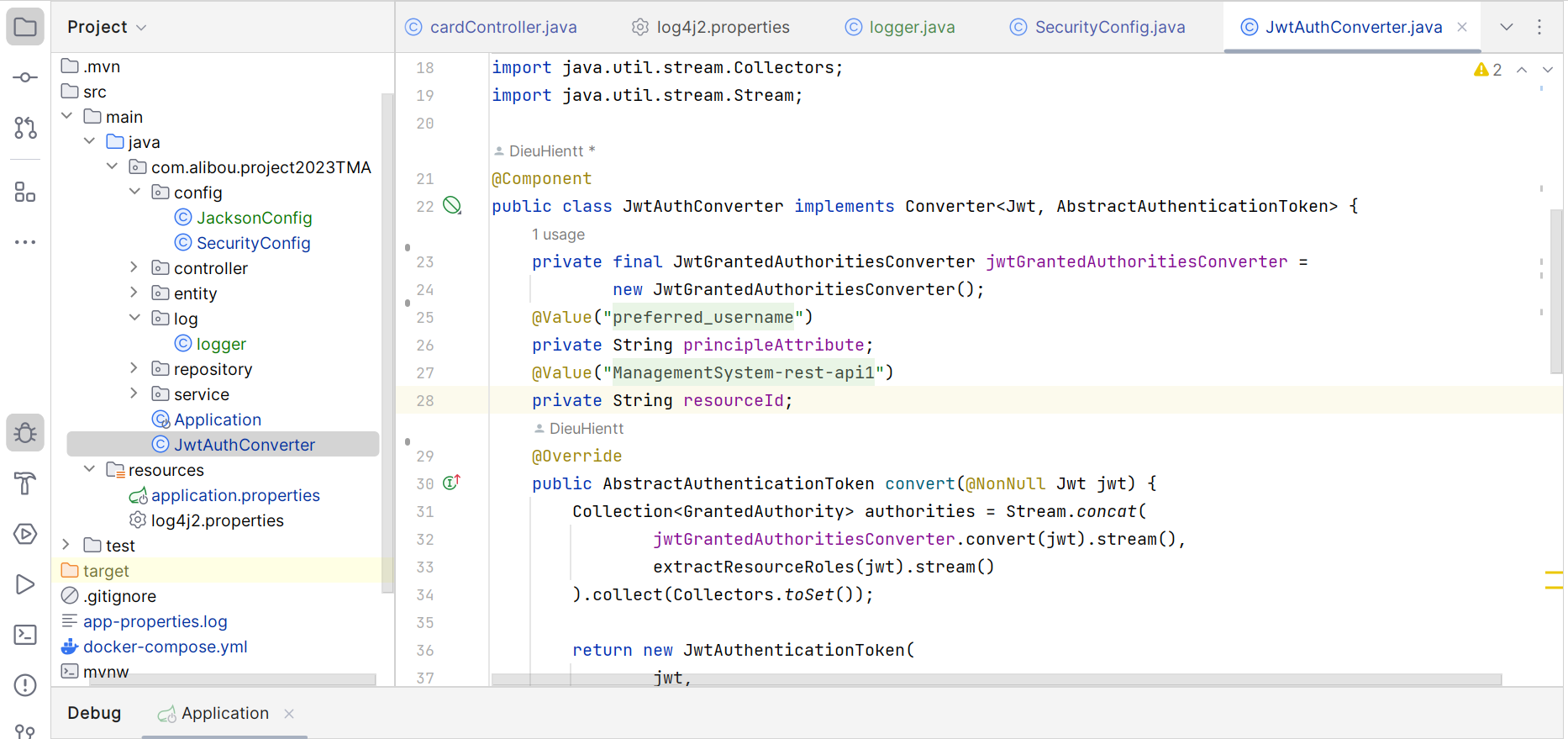


*Hình 4.2. Kết nối với Keycloak*



*Hình 4.3. Class SecurityConfig*

SecurityConfig để bảo mật các yêu cầu HTTP, sử dụng xác thực JWT và chế độ không sử dụng phiên làm việc. Mã code này giúp xác định quyền truy cập và cách xác thực trong hệ thống.



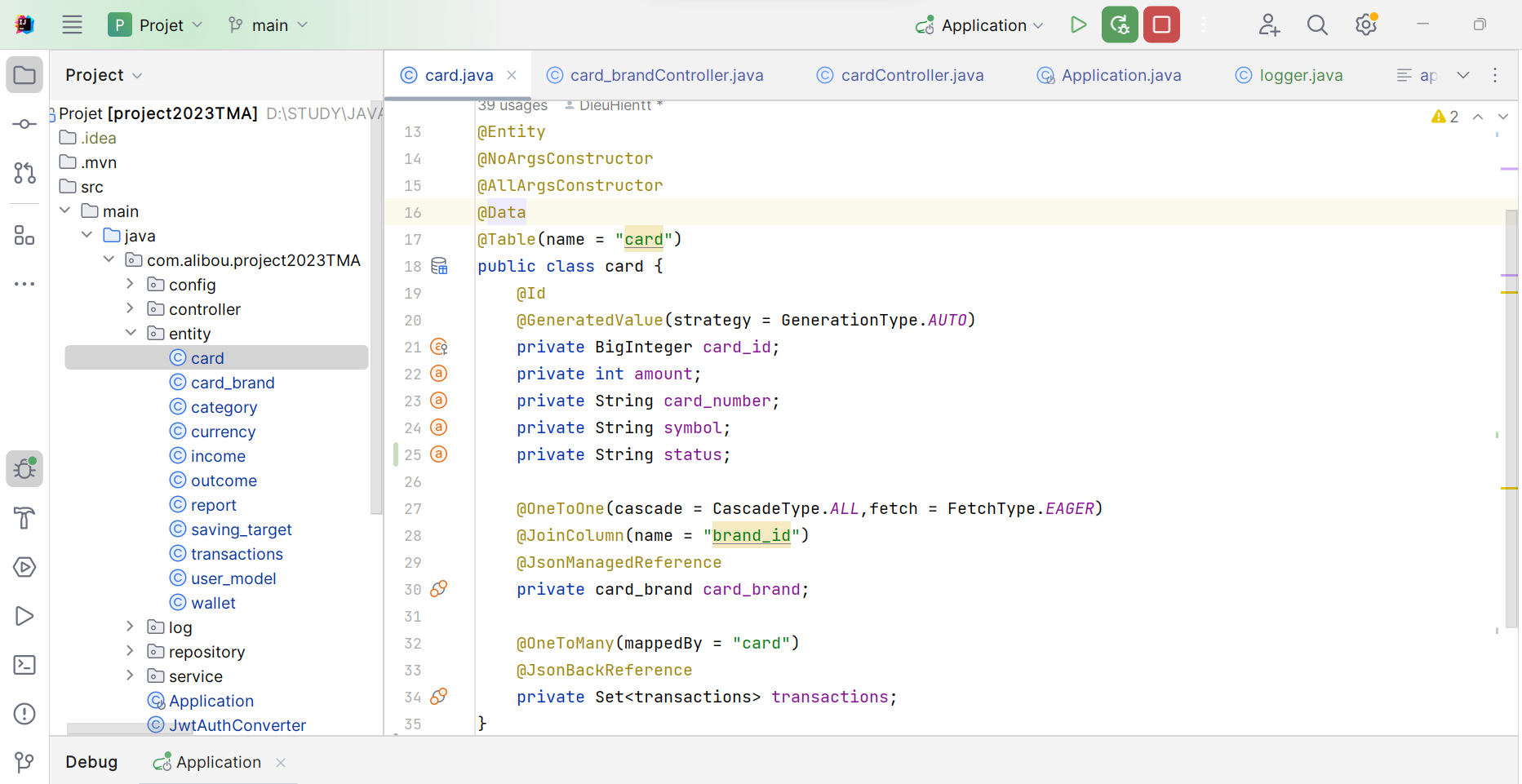
*Hình 4.4. Class JwtAuthConverter*

JwtAuthConverter thực hiện chức năng chuyển đổi và trích xuất thông tin từ JWT để xác thực và bảo mật người dùng trong ứng dụng. Các thông tin quyền và thông tin người dùng chính được chuyển đổi và sử dụng để xác định quyền truy cập vào các tài nguyên của hệ thống.

**4.2. Triển khai mô hình MVC**

**4.2.1.** **Entity**

Package có tên Entity để lưu các Class. Đây là ví dụ về Class card:

****

*Hình 4.2. Xây dựng Class card*

Tạo Class card theo mô hình OOP với các trường như: card\_id, amount, card\_number, symbol, status.

Các animations trong Spring Boot:

@Entity: tạo thực thể

@Table, @Id, @Column: tạo liên kết với PostgreSQL

@GeneratedValue: tạo Id tự tăng

@NoArgsConstructor: tạo constructor tự động không có tham số cho lớp @AllArgsConstructor: tạo constructor tự động có tham số cho lớp

@Data: giảm thiểu việc viết các mã boilerplate: getter, setter, to string,..

@OneToOne: để thiết lập một mối quan hệ một một (One-to-One) giữa hai thực thể.

cascade = CascadeType.ALL: định nghĩa các hoạt động CRUD tự động lan truyền từ thực thể gốc tới thực thể liên quan.

fetch = FetchType.EAGER: chỉ định rằng dữ liệu liên quan sẽ được truy xuất ngay lập tức khi thực thể gốc được truy xuất.

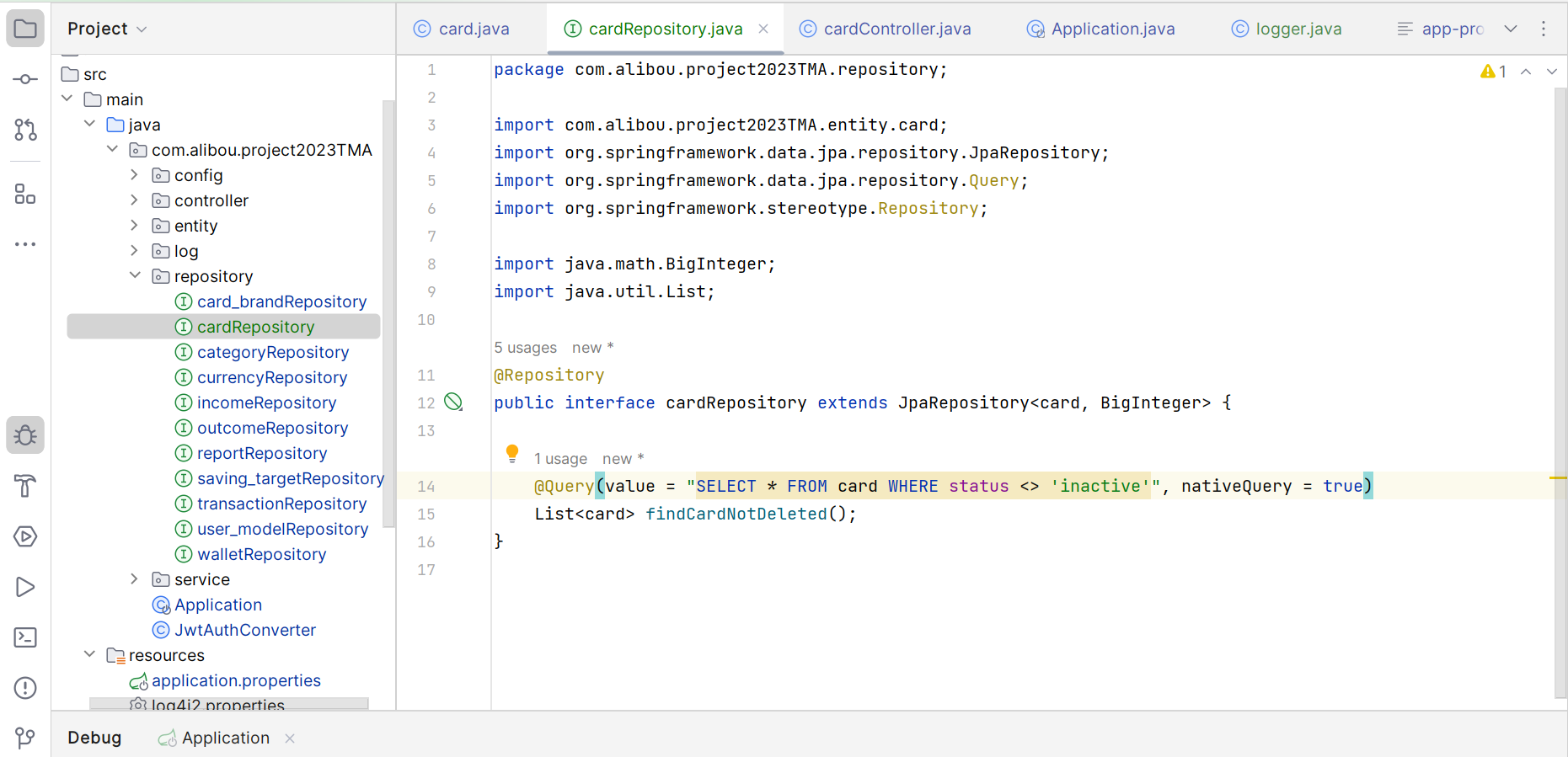
@JoinColumn: để xác định cột trong cơ sở dữ liệu mà sẽ được sử dụng để thực hiện liên kết giữa các thực thể.

@OneToMany: xác định mối quan hệ một nhiều (One-to-Many) giữa hai thực thể.

@JsonManagedReference, @JsonBackReference: hai annotations được sử dụng trong thư viện Jackson của Java để giải quyết vấn đề vòng lặp khi serialize (chuyển từ đối tượng Java sang định dạng JSON) các mối quan hệ đối tượng.

**4.2.2. Repository**

Package Repository là tầng được sử dụng để chứa các lớp (classes) thực hiện truy vấn và giao tiếp với cơ sở dữ liệu. Đây là ví dụ về Class cardRepository:

****

*Hình 4.3. Xây dựng Class cardRepository*

@Repository là một annotation thông báo cho Spring rằng Lớp được đánh dấu là một bean có khả năng xử lý truy cập cơ sở dữ liệu.

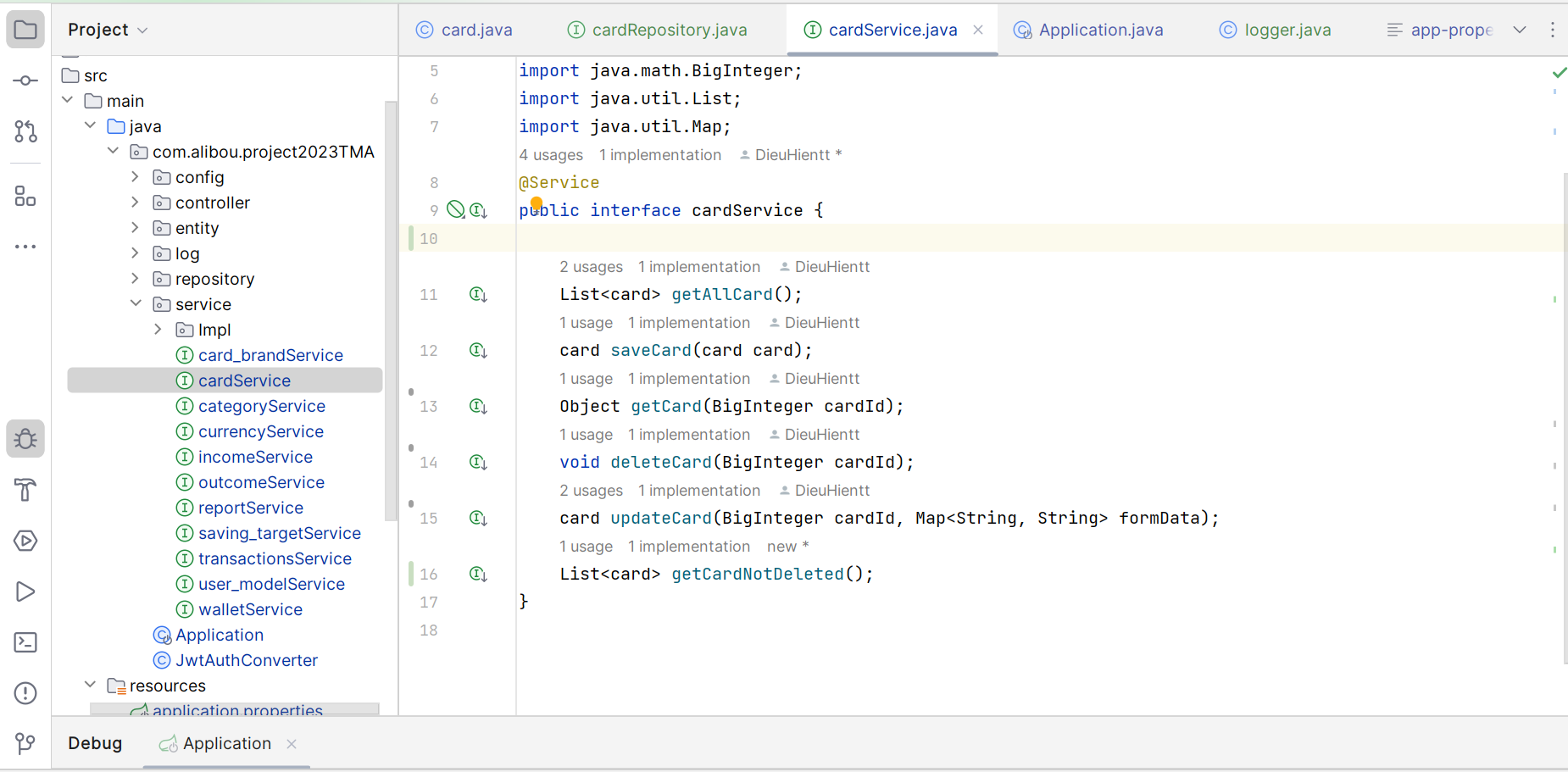
JpaRepository là một interface , mở rộng nó cho mỗi thực thể để thao tác viết mã nguồn thủ công để thực hiện các phương thức cơ bản như thêm, sửa, xóa, tìm kiếm.

@Query: viết câu truy vấn. Trong Class cardRepository là để truy vấn ra danh sách đối tượng card có status <>’inactive’.

**4.2.3. Service**

Package Service là tầng phục vụ các logic CRUD.

**4.2.3.1. Các Class Service**

****

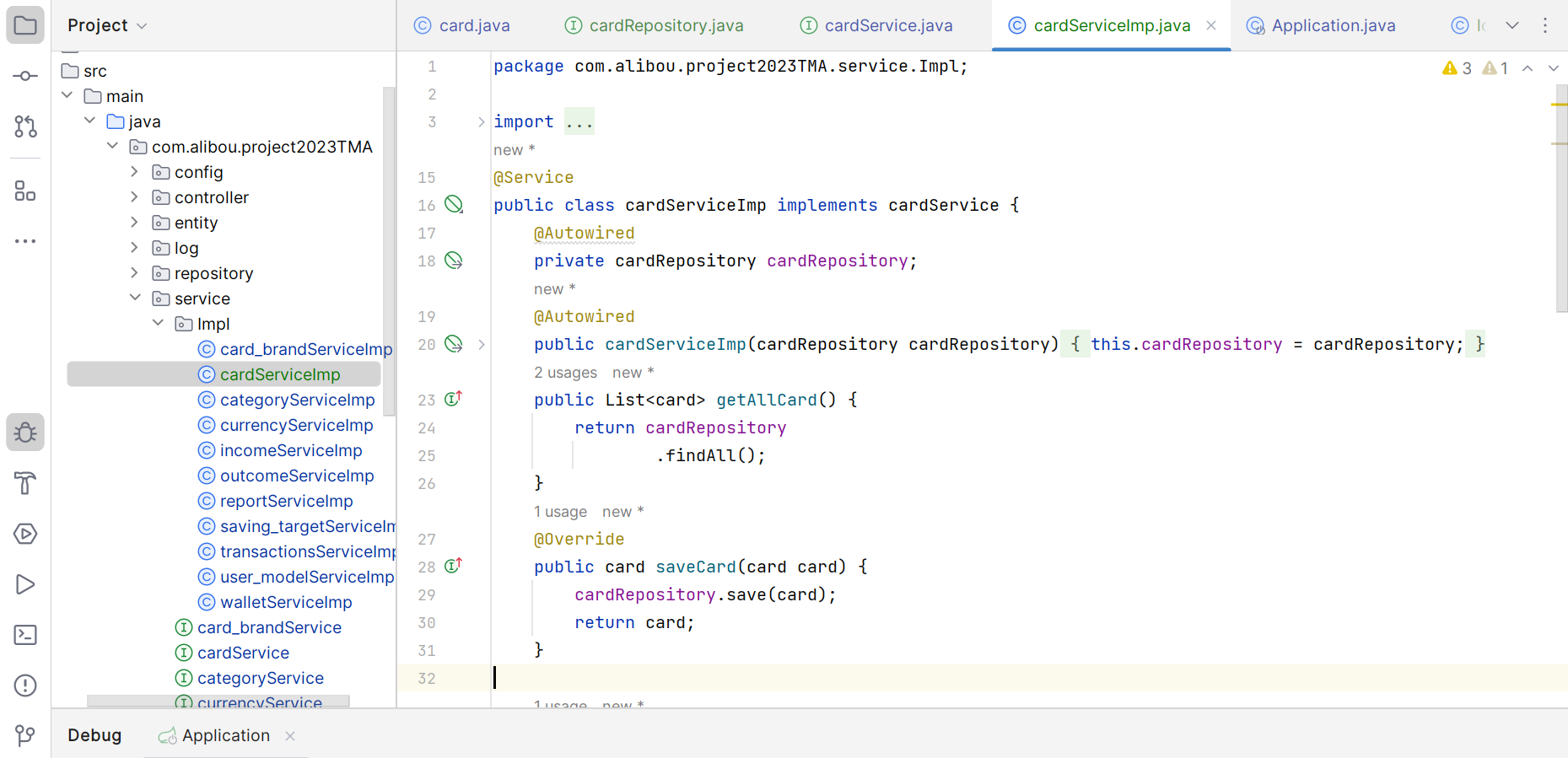
*Hình 4.4. Xây dựng Class cardService*

Interface cardService chứa các chức năng của quản lý sản phẩm gồm:

* getAllCard(): hàm hiển thị danh sách card. Đây được coi là READ trong CRUD.
* saveCard(): hàm lưu một đối tượng card mới khi ta thêm 1 card. Đây được coi là Create trong CRUD.
* getCardById(): hàm hiển thị thông tin card theo card\_Id.
* updateCard(): hàm hiển thị một đối tượng card theo id khi ta chỉnh sửa 1 card. Đây được coi là Update trong CRUD.
* deleteCardtById(): hàm xóa một đối tượng sản phẩm theo id khi ta xóa 1 sản phẩm. Đây được coi là Delete trong CRUD.
* getCardNotDeleted(): hàm hiển thị danh sách card có ‘status’ <>’inactive’ sau khi chỉnh sửa. Do không nên xóa dữ liệu trong một hệ thống mà nên đổi trạng thái hoạt động.

**4.2.3.2. Service Implement**

Package Impl là tầng để triển khai logic cụ thể của Service. Trong Class cardServiceImp:



*Hình 4.5. Xây dựng Class cardServiceImp*

@Autowired sẽ đánh giấu khởi tạo đối tượng cardRepository thay vì phải tạo Constructor trong đối tượng.

@Override lại phương thức getAllCard() từ lớp cha cardService, tiến hành trả về toàn bộ data thông qua hàm findAll().

@Override lại phương thức saveCard() từ cardService, tiến hành lưu đối tượng được khởi tạo thông qua hàm save(card).

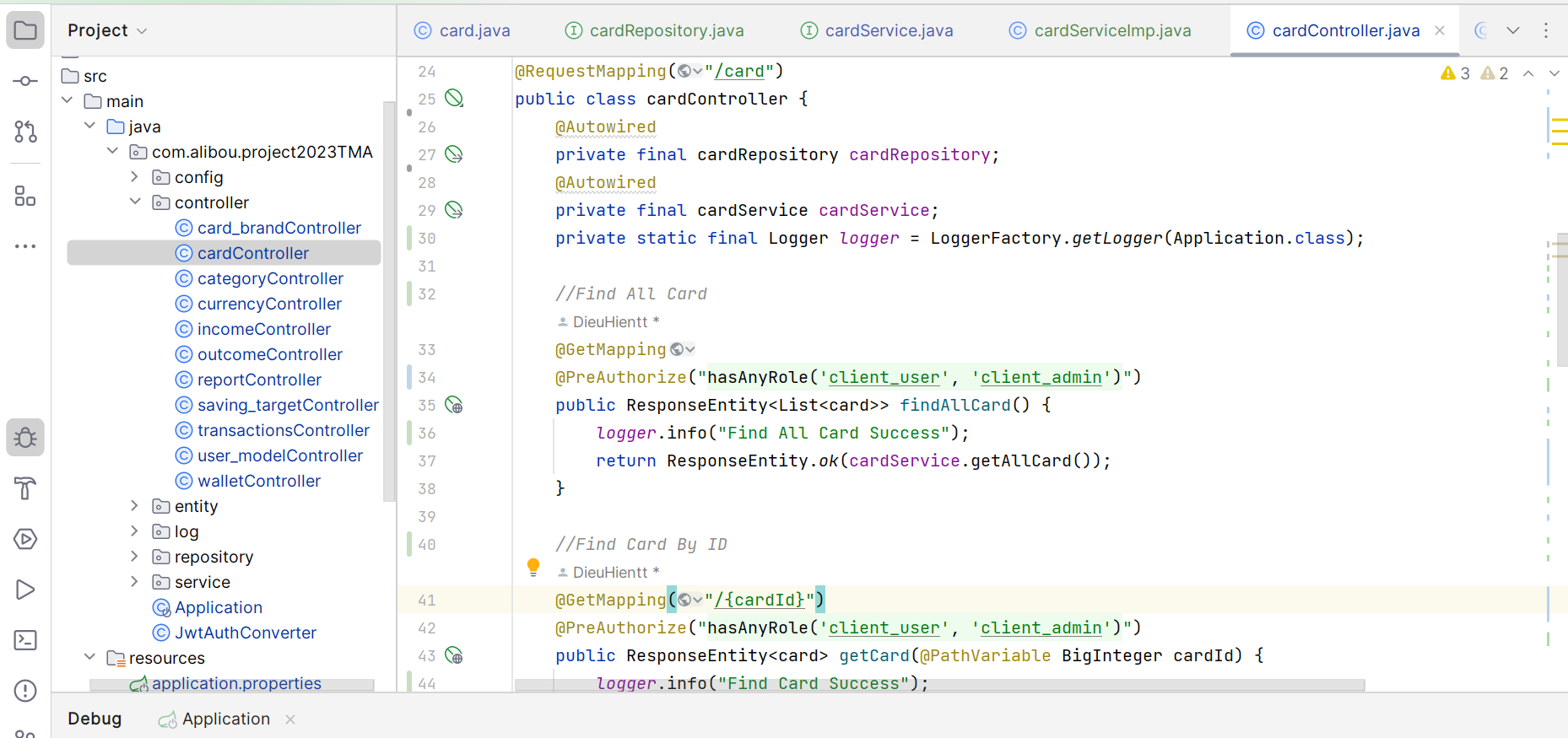
@Override lại phương thức getCardById() từ cardService, tạo đối tượng optional với kiểu dữ liệu card nhằm lưu các đối tượng card trong data thông qua phương thức findById(card\_id). Nếu optional không rỗng thì gọi card được lưu trong card ra theo card\_id. Còn rỗng thì ném ra ngoại lệ.

@Override lại phương thức deleteCardById() từ cardService, tiến hành xóa sản phẩm trong data theo id qua hàm deleteById(card\_id).

@Override lại phương thức getCardNotDeleted() từ cardService, tiến hành trả về toàn bộ data hợp lệ thông qua hàm findCardNotDeleted().

**4.2.4. Tạo Controller**

Package Controller là tầng lấy và sửa đổi dữ liệu cung cấp cho người dùng. Là nơi kết nối giữa View và Model.



Hình 4.6. Xây dựng Class cardController

Import những thư viện, annotations cần thiết trong Class cardController. @PreAuthorize("hasAnyRole('client\_user', 'client\_admin')") dựa trên Keycloak để phân quyền sử dụng API cho ‘user’ và ‘admin’.

Logger để dễ dàng debug lỗi trong quá trình chạy code.

* Hàm findAllCard() để tiến hành tạo API cho client có thể tìm kiếm danh sách card.
* Hàm getCard) để tiến hành tạo API cho client có thể tìm kiếm card theo card\_Id. saveCard(card card).
* Hàm saveCard() để tiến hành tạo API cho client có thể nhập dữ liệu vào, trả về thông tin card đã nhập vào.
* Hàm updateCard() để tiến hành tạo API cho client có thể nhập dữ liệu vào theo card\_id, trả về thông tin card đã nhập vào.
* Hàm getCardNotDeleted() để tiến hành tạo API trả về thông tin card có ‘status’ <>’inactive’.
* Hàm deleteCard() để tiến hành tạo API xóa một đối tượng do client chỉ định, sau đó xóa dữ liệu ở trong database tương ứng theo id.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

* **Đạt được:**

Trong thời gian thực tập dưới sự dẫn dắt của mentor em đã tìm hiểu và nắm được những kiến thức cơ bản của Java, cách sử dụng Github, Postman, PostgreSQL, Keycloak, Docker, IntelliJ. Đồng thời biết cách áp dụng những công nghệ vào trong đề tài đang làm từ đó xây dựng được trang web và có thể thực hành kiến thức để nắm chắc hơn.

Qua đó, em đã nâng cao được khả năng tự học, tự tìm hiểu kiến thức mới thông qua các từ khóa được mentor giao, tiếp thu kiến thức một cách chủ động; nâng cao hiểu biết của bản thân. Được tiếp thu kiến thức và kinh nghiệm của mentor và đồng nghiệp xung quanh.

Cải thiện kỹ năng của cá nhân: giao tiếp, thuyết trình, làm việc nhóm, lên kế hoạch, sắp xếp thời gian.

Được nâng cao kiến thức nhờ các buổi training những kỹ năng: bảo mật trong doanh nghiệp, cách viết Email, phương pháp Agile, mô hình Scrum, nghi thức làm việc, trao đổi về cách hành xử trong doanh nghiệp.

Đặc biệt là khi được thực tập tại Công ty TMA em đã được làm quen với môi trường làm việc chuyên nghiệp, được thử nghiệm thực tế công việc của mình, giúp em hiểu rõ hơn về các nhiệm vụ, thách thức và cơ hội trong lĩnh vực của mình. Thực tập tại đây, giúp em có cơ hội tiếp xúc và làm việc với các công nghệ, framework và công cụ được sử dụng rộng rãi trong ngành; đã giúp em kết nối với những người bạn, anh chị, mentor cùng ngành.

* **Hạn chế:**
* Sinh viên cần chỉ ra các mặt hạn chế của bản thân, những mục tiêu mà mình chưa đạt được so với yêu cầu của doanh nghiệp/GVHD,…;
* …
* **Hướng phát triển:**
* Sinh viên đưa ra các kế hoạch dự kiến để hoàn thiện năng lực của bản thân,…;

# TÀI LIỆU THAM KHẢO

* + - 1. Tên tác giả, Tên tác giả… *Tên công trình*. Nơi xuất bản, năm, trang
      2. Tên tác giả, Tên tác giả… *Tên công trình*. Nơi xuất bản, năm, trang

# CHECK LIST CỦA BÁO CÁO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Nội dung công việc** | **Có** | **Không** | **Ghi chú** |
| 1 | Báo cáo được trình bày (định dạng) đúng với yêu cầu. |  |  |  |
| 2 | Báo cáo có số lượng trang đáp ứng đúng yêu cầu (30-50 trang) |  |  |  |
| 3 | Báo cáo trình bày được phần mở đầu bao gồm: Mục tiêu, Phạm vi và đối tượng, kết cấu … |  |  |  |
| 4 | Báo cáo trình bày về công ty, vị trí việc làm (công việc đó làm gì, kiến thức và kỹ năng cần thiết là gì, con đường phát triển sự nghiệp (career path)), cơ sở lý thuyết phù hợp với nội dung của đề tài (Tối đa 10-12 trang) |  |  |  |
| 5 | Báo cáo có sản phẩm cụ thể phù hợp với mục tiêu đặt ra của đề tài |  |  |  |
| 6 | Báo cáo có phần kết luận và hướng phát triển của đề tài |  |  |  |

**Quy định (SINH VIÊN XÓA PHẦN NÀY KHI TRÌNH BÀY BÁO CÁO):**

* Sinh viên check **X** vào ô Có hoặc Không, ghi chú thêm nếu cần;
* Soạn thảo trên trang A4 (trang dọc, lề trái: 3.0cm; trên, phải, dưới: 1.5 cm), Font Times New Roman, canh đều 2 bên, size 13, cách dòng 1.5, cách đoạn trên 6pt, cách đoạn dưới 3pt, hàng đầu tiên lùi vào 1.27 cm; hình và bảng soạn theo caption, chèn trích dẫn chéo (Cross-reference) cho bảng và hình; các danh mục hình, bảng, mục lục làm tự động; đánh số trang như file mẫu (bìa không có số trang, danh mục + mục lục số trang theo số la mã i, ii, iii…, nội dung chính theo số thứ tự 1, 2, 3…);
* Báo cáo tối đa 30 trang.

# PHỤ LỤC