LỜI CẢM ƠN

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

TÓM TẮT

ABSTRACT

CAM KẾT KẾT QUẢ

Tôi cam đoan luận văn này được hoàn thành dựa trên kết quả nghiên cứu của tôi, các kết quả này chưa được cung cấp cho bất cứ luận văn nào.

Tôi cam đoan mọi tham khảo và sự giúp đỡ để thực hiện luận văn này đã được ghi rõ nguồn gốc.

Cần Thơ, ngày tháng 11 năm 2019

Người thực hiện

Phạm Hữu Thiện

GIỚI THIỆU (1)

ĐẶT VẤN ĐỀ

Hiện nay, việc các cơ quan, tổ chức hay doanh nghiệp trong nước đang dần loại bỏ các quy trình thủ công bằng các hệ thống quản lí điện tử đã không còn xa lạ với chúng ta. Đây là sự phát triển tất yếu tuy còn khá chậm nếu so với các nước phát triển trong khu vực cũng như trên thế giới. Những giải pháp công nghệ thông tin khi đưa vào sử dụng sẽ góp phần lớn làm tăng hiệu suất công việc, giảm thiếu các quy trình lằng nhằng, giảm thời gian xử lý và kinh phí hoạt động cho cơ quan, doanh nghiệp.

Tương tự, việc áp dụng Hệ thống quản lí hiến máu trực tuyến sẽ giúp đơn giản hóa và đồng bộ quá trình tổ chức các đợt hiến máu, tối ưu thông tin đến sinh viên, dễ dàng cập nhật, đăng ký tham gia khi có các đợt hiến máu được tổ chức.

PHẠM VI ĐỀ TÀI

Những chức năng chính

Đăng nhập hệ thống

Phân quyền người dùng

Tạo yêu cầu tổ chức đợt hiến máu mới

Xác nhận yêu cầu

Hủy bỏ yêu cầu

Gửi mail thông báo yêu cầu mới

Đăng ký tham gia đợt hiến máu

Xác nhận phiếu bằng tay

Xác nhận phiếu bằng barcode

Hủy bỏ phiếu

Gửi mail thông báo trạng thái phiếu

Xuất phiếu xác nhận đã tham gia

Thống kê

Công nghệ sử dụng

Back-end framework: Spring Boot, Spring Security

Front-end library: ReactJS

Ngôn ngữ lập trình: Java, Javascript

Hệ quản trị cơ sở dữ liệu: MySQL

Các công cụ hỗ trợ khác: Postman, Git

Trình duyệt web: Chrome, Firefox…

PHƯƠNG PHÁP NGHIÊN CỨU

Lý thuyết:

Phân tích, thiết kế hệ thống thông tin.

Thiết kế, chuẩn hóa cơ sở dữ liệu.

Các kĩ thuật lập trình.

Xây dựng, thiết kế, sử dụng RESTful API.

Thực tiễn:

Sử dụng dự án Spring boot trong hệ sinh thái Spring framework để xây dựng các RESTful API bằng ngôn ngữ lập trình Java với Spring Tool Suite 4 IDE.

Sử dụng thư viện Javascript ReactJS xây dựng giao diện, gọi và xử lí API với Visual Studio Code.

Kiểm thử API với Postman.

Sử dụng hệ quản trị cơ sở dữ liệu MySQL với MySQL Workbench.

CƠ SỞ LÝ THUYẾT (2)

SPRING FRAMEWORK (1)

Đôi nét về lịch sử hình thành (1.1)

Spring Framework ra đời phần lớn do sự thành công của EJB (Enterprise JavaBeans) – một phần quan trọng trong nền tảng Java EE. Nhưng với độ phức tạp cao về mặt kĩ thuật, cộng đồng phát triển EJB đòi hỏi một giải pháp thay thế, vì vậy Spring Framework đã ra đời.



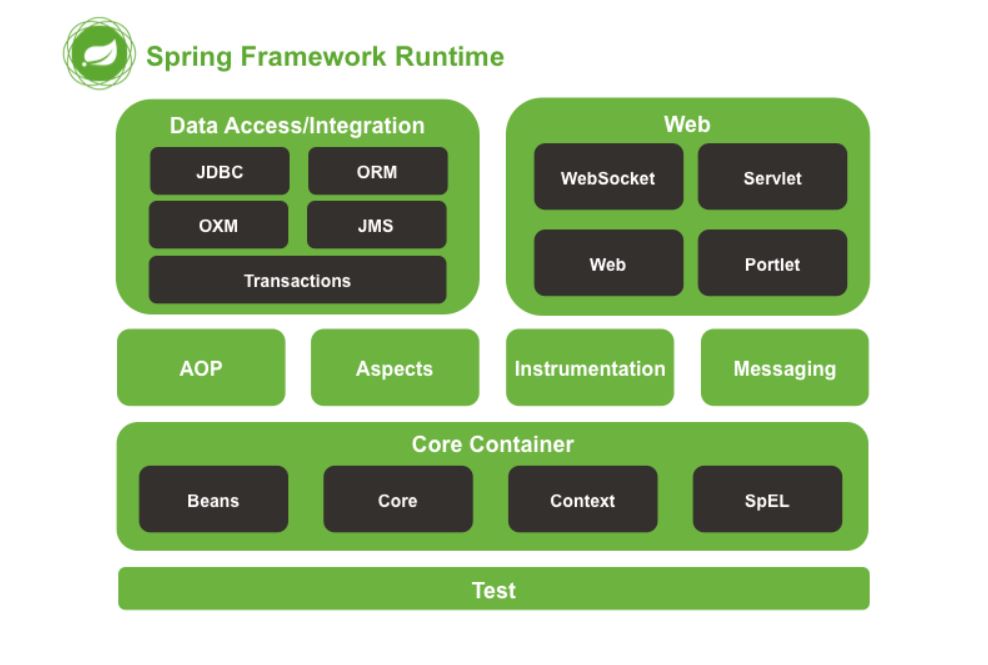
Spring Framework

Năm 2002, Spring Framework phát hành phiên bản đầu tiên bởi Rod Johnson. Việc xây dựng các ứng dụng doanh nghiệp trở nên đơn giản và dễ dàng hơn. Dần dần, nó trở thành framework mã nguồn mở phổ biến nhất để xây dựng các ứng dụng web bằng ngôn ngữ Java. Theo thống kê, hơn 50% ứng dụng Java web hiện nay sử dụng Spring.

Spring Core (1.2)

Spring Core là thành phần cốt lõi của Spring Framework. Đây là thành phần chính để xây dựng các thành phần khác trong hệ sinh thái Spring Framework.

Lược đồ dưới đây mô tả sơ lược mối quan hệ giữa Spring Core với các thành phần khác trong Spring Framework.



Mối quan hệ giữa Spring Core và các thành phần trong Spring Framework

Spring Bean (1.2.1)

Spring Bean là trung tâm của Spring Core và là trái tim của một ứng dụng Spring. Spring Framework được thiết kế từ lõi bằng cách sử dụng các POJO (Plain Old Java Object) hay Spring Bean - các đối tượng Java đơn giản.

Dependency Injection (1.2.2)

Dependency Injection là một sức mạnh nổi bật của Spring Framework. Đây là một mẫu thiết kế được sử dụng để loại bỏ sự phụ thuộc của các mã chương trình, giúp việc quản lí và kiểm thử ứng dụng được dễ dàng hơn.

Dependency Injection là một triển khai cụ thể của khái niệm Inversion of Control (đảo ngược điều khiển).

Spring Context (1.2.3)

Spring Context mang mọi thứ lại với nhau trong Spring Framework. Spring Context kế thừa các tính năng của Spring Bean và bổ sung các hỗ trợ cho internationalization, event propagation, resources loading… Ngoài ra, Spring Context cũng hỗ trợ các tính năng của Java EE như EJB, JMX (Java Management Extensions) và RMI (Remote Method Invocation). Trong đó, interface “ApplicationContext” chính là trọng tâm của Spring Context.

Spring Extension Language (SqEL) (1.2.4)

Spring Extension Language là một ngôn ngữ ngắn gọn giúp việc cấu hình Spring Framework trở nên linh hoạt hơn.

Các dự án trong Spring Framework (1.2.5)

Trên thực tế, Spring Framework là tập hợp của nhiều dự án con. Mỗi dự án con sẽ đảm nhận một chức năng riêng trong việc xây dựng các ứng dụng. Dưới đây là một vài dự án lớn và nổi bật trong hệ sinh thái Spring:

Spring MVC: là một dự án không thể bỏ qua, được thiết kế dành cho việc xây dựng các ứng dụng nền tảng web.

Spring Data: cung cấp một cách tiếp cận đúng đắn để truy cập dữ liệu từ cơ sở dữ liệu quan hệ, phi quan hệ, map-reduce…

Spring Boot: giúp chúng ta phát triển và deploy dự án một cách nhanh chóng.

Spring Security: cung cấp các cơ chế xác thực (authentication) và phân quyền (authorization) cho ứng dụng.

Spring Batch: giúp chúng ta tạo các lịch trình (scheduling) và tiến trình (processing) cho việc xử lí batch job.

Giới thiệu Spring Boot (1.3)

Spring Boot là một dự án khá nổi bật trong hệ sinh thái Spring Framework. Nếu như trước đây, công đoạn khởi tạo một dự án Spring khá vất vả, từ việc khai báo các dependency trong file pom.xml cho đến việc cấu hình bằng XML hoặc annotation phức tạp thì giờ đây với Spring Boot, chúng ta có thể tạo một dự án Spring nhanh chóng và cấu hình đơn giản hơn.

Một số tính năng nổi bật của Spring Boot:

Tạo các ứng dụng Spring độc lập

Nhúng trực tiếp server như Tomcat, Jetty hoặc Undertow nên không cần triển khai thành file WAR.

Các starter dependency có sẵn giúp việc cấu hình Maven trở nên đơn giản hơn.

Không phát sinh code cấu hình, không yêu cầu phải cấu hình bằng XML

Giới thiệu RESTful API (1.4)

Có thể nói, nguyên lí REST và cấu trúc dữ liệu RESTful đã được biết đến rộng rãi trong trong cộng đồng lập trình web nói chung và lập trình ứng dụng nói riêng.

Bản thân REST không phải là một loại công nghệ, nó là phương thức tạo API (Application Programming Interface) với nguyên lý tổ chức nhất định. Những nguyên lý này nhằm hướng dẫn lập trình viên có thể tạo môi trường xử lí API một cách toàn diện.

Để hiểu rõ hơn về RESTful API, chúng ta sẽ từng bước tìm hiểu các khái niệm API, REST hay RESTful.

API (1.4.1)

API hay Application Programming Interface, là một tập hợp các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác.

API có thể trả về kiểu dữ liệu mà bạn cần cho ứng dụng của mình ở nhiều dạng như JSON hay XML…

REST (1.4.2)

REST (Representation State Transfer) là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API.

REST sử dụng phương thức HTTP đơn giản để tạo giao tiếp. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, PUT, DELETE… đến một URL để xử lý dữ liệu.

RESTful API (1.4.3)

RESTful API là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để các ứng dụng (web, di dộng…) khác nhau giao tiếp với nhau.

RESTful không quy định logic của mã chương trình của ứng dụng và không bị giới hạn bởi ngôn ngữ lập trình hoặc bất kì framework nào.

Cách hoạt động của RESTful (1.4.4)

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hành động cơ bản kể trên sẽ sử dụng những phương thức HTTP riêng.

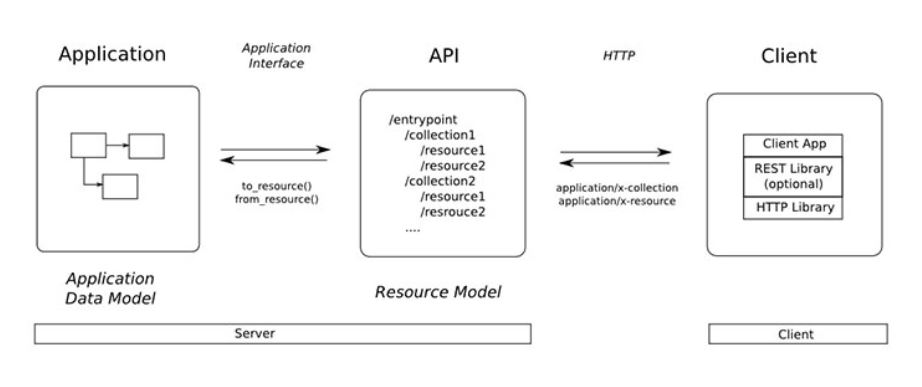
GET (SELECT):

POST (CREATE):

PUT (UPDATE):

DELETE (DELETE):

…



Phương thức hoạt động của RESTful

Status code (1.4.5)

Khi chúng ta gửi yêu cầu một API, sẽ có những status code trả về để nhận biết kết quả của yêu cầu đã được gửi, dưới đây là một số status code phổ biến:

200 OK: yêu cầu được gửi thành công.

201 Created: trả về một resource vừa được tạo thành công.

204 No Content: trả về khi xóa resource thành công.

400 Bad Request: yêu cầu không hợp lệ.

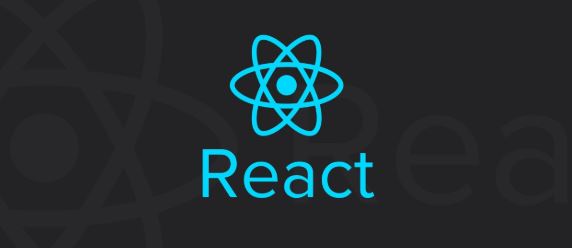
401 Unauthorized: không có quyền thực thi.

…

REACT (2)

Khái niệm (2.1)

React (hay ReactJS, React.js) là một thư viện Javascript mã nguồn mở được phát triển bởi Jordan Walke, một kĩ sư phần mềm tại Facebook. React hỗ trợ xây dựng các thành phần (component) UI có tính tương tác cao, có trạng thái và có thể tái sử dụng.



Thư viện ReactJS

Hiện nay, React đang nhận được rất nhiều sự quan tâm đến từ cộng đồng front-end. Facebook và Instagram đang là hai ông lớn đang đứng sau bảo trì và phát triển React với sự đóng góp của cộng đồng lập trình viên giỏi trên toàn thế giới.

Tại sao lại chọn React? (2.2)

Về cơ bản, việc xây dựng một ứng dụng MVC phía client với ràng buộc dữ liệu hai chiều (two ways data-binding) khá đơn giản. Tuy nhiên, nếu dự án ngày càng mở rộng, ngày càng nhiều tính năng hơn thì việc bảo trì sẽ trở nên rất khó khăn, cũng như hiệu năng của dự án sẽ giảm xuống. So với Backbone.js hay Angular.js, React nhanh hơn, nhẹ hơn, đơn giản hơn và dễ dàng mở rộng hơn.

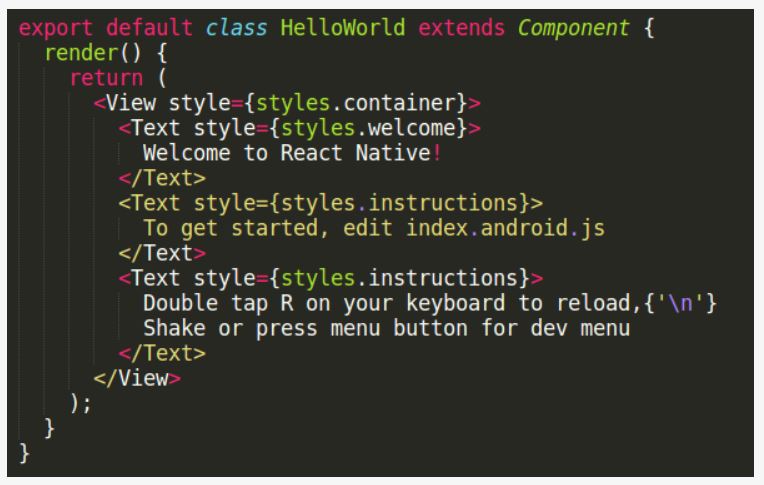
Ưu điểm (2.3)

Viết mã Javascript dễ dàng hơn với JSX (2.3.1)

React sử dụng JSX (Javascript Syntax eXtension) – sự pha trộn giữa Javascript và XML, cho phép viết mã HTML lồng trong mã Javascript.

JSX nhanh hơn: nó thực hiện tối ưu hóa trong khi biên dịch sang mã Javascipt.

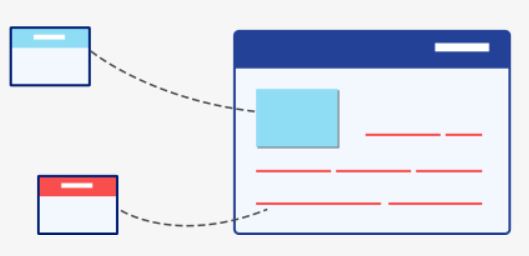
JSX an toàn hơn: JSX biên dịch trước khi chạy (statically-typed), vì vậy các lỗi cú pháp sẽ được phát hiện ngay trong quá trình biên dịch.



Ví dụ JSX

React thành phần hóa giao diện (2.3.2)

React cho phép lập trình viên tạo ra các Component (Thành phần) tương ứng với các phần của giao diện. Các Component này có thể tái sử dụng hoặc kết hợp với các Component khác để tạo ra giao diện hoàn chỉnh.



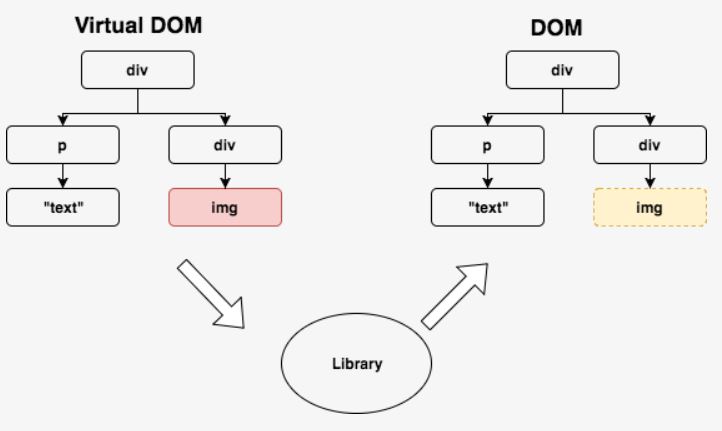
Ví dụ về xây dựng giao diện bằng các Component

Component chính là chìa khóa giải quyết khó khăn khi dự án ngày càng phình to. Giao diện được tạo từ các Component ghép lại với nhau, một Component có thể được tái sử dụng nhiều lần. Do đó, rất dễ dàng để quản lí và bảo trì.

Component thực sự là tương lai của lập trình web, nó cũng là tính năng quan trọng nhất mà React mang lại.

Tăng hiệu năng với Vitural-DOM (2.3.3)

Khi dữ liệu của Component thay đổi, React cần phải vẽ (render) lại giao diện. Thay vì tạo ra thay đổi trực tiếp vào mô hình DOM (Document Object Model), React tạo ra thay đổi trên một DOM ảo (Virtual-DOM). Sau đó nó tính toán sự khác biệt giữa hai DOM này và chỉ cập nhật các khác biệt cho DOM của trình duyệt. Cách tiếp cận này mang lại hiệu năng cho ứng dụng.



Ví dụ minh họa DOM và Virtual-DOM

React thân thiện với SEO (Search Engine Optimization) (2.3.4)

Một trong những vấn đề lớn của các Javascript Framework là nó không thân thiện với các cỗ máy tìm kiếm (Search Engine). Mặc dù đã có nhiều cải thiện nhưng thực sự Search Engine vẫn gặp nhiều khó khăn khi “đọc” các ứng dụng chứa nhiều Javascript.

Do React có thể chạy phía Client, nhưng cũng có thể chạy phía Server, vì vậy dữ liệu trả về là HTML nên không gây khó khăn gì cho Search Engine.

Một số khái niệm React cơ bản (2.4)

Props và State 2.4.1

Props 2.4.1.1

Props là thông tin được truyền từ component này sang component khác.

Props của component là một đối tượng, nó chứa thông tin về thành phần đó. Để xem props của component chứa gì, ta dùng đoạn lệnh “this.props”.

Trong props của mỗi component có một thuộc tính là children, this.props.children sẽ trả về mọi thứ nằm giữa các thẻ JSX mở và đóng của component.

State 2.4.1.2

Một component có thể truy cập thông tin theo hai cách: props hoặc state.

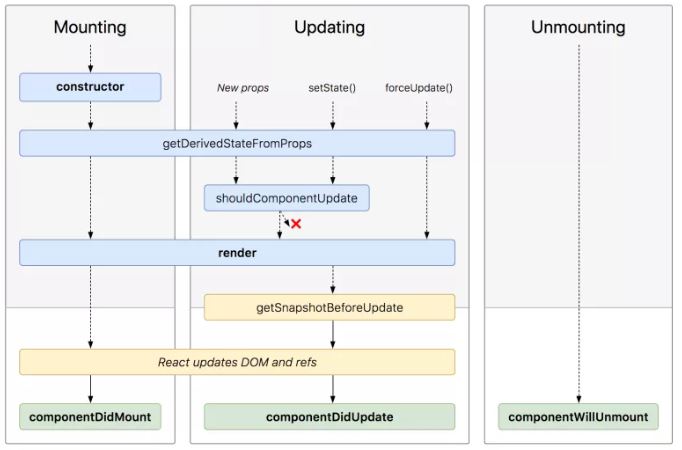
Không giống như props, state của component không thể truyền được ra ngoài. Component sẽ quản lí state của chính nó.

State sẽ được khai báo bên trong phương thức dựng (constructor) của lớp Component.

Chúng ta có thể thay đổi một state bằng cách sử dụng hàm this.setState().

Lifecycle methods 2.4.2

Lifecycle methods là những phương thức được gọi tại một thời điểm nào đó trong vòng đời một Component. Có ba loại Lifecycle methods là Mounting, Updating và Unmounting.



Mounting 2.4.2.1

Một component thực hiện “mount” chỉ khi nó thực hiện render trong lần đầu tiên. Có ba mounting lifecycle methods:

componentWillMount

render

componentDidMount

Updating 2.4.2.2

Sau khi component render lần đầu tiên, các Lifecycle methods của Updating sẽ được gọi bắt đầu với lần render thứ hai. Với cơ chế automatic binding của mình thì chắc chắn các Component sẽ được render nhiều lần trong ứng dụng của bạn. Có năm phương thức trong vòng đời Updating:

componentWillReceiveProps

shouldComponentUpdate

componentWillUpdate

render

componentDidUpdate

Unmounting 2.4.2.3

Quá trình unmounting của component xảy ra khi component bị chuyển từ một DOM. Việc này có thể xảy ra khi một DOM được render mà không có component hoặc nếu người dùng chuyển hướng đến một trang web khác hoặc khi trình duyệt được đóng.

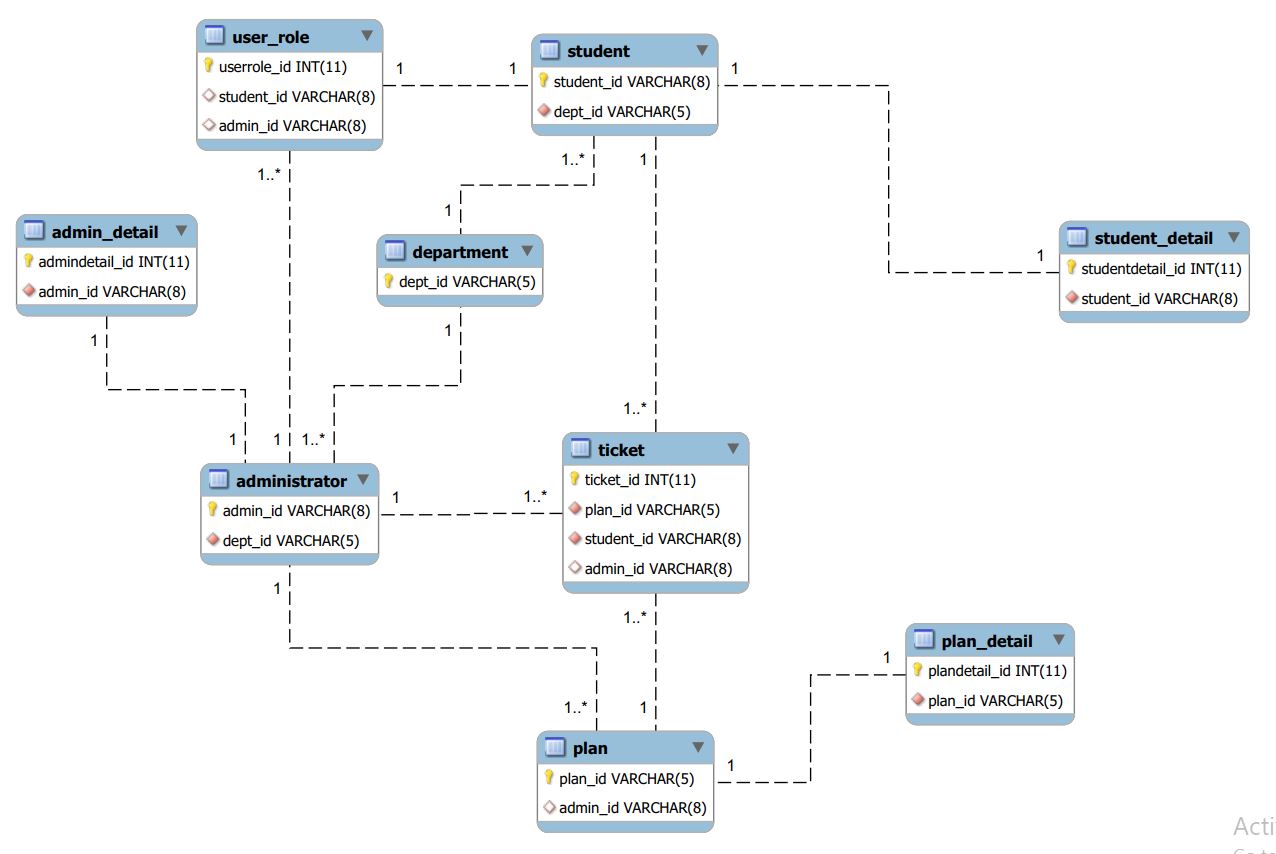
Chỉ có duy nhất một phương thức trong quá trình này: componentWillUnmounting.

NỘI DUNG VÀ KẾT QUẢ NGHIÊN CỨU (3)

THIẾT KẾ VÀ CÀI ĐẶT GIẢI PHÁP 3.1

Mô tả giải pháp

Thiết kế cơ sở dữ liệu



Mô hình Entity Relationship (ER)

Mô tả các chức năng