

**ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC SÀI GÒN**  
**KHOA CÔNG NGHỆ THÔNG TIN**

---



**Họ và tên sinh viên : Trần Anh Tuấn**

**BÁO CÁO**  
**THỰC TẬP TỐT NGHIỆP**

**Công ty thực tập : Công Ty Tư Vấn Khoa Học Công Nghệ An Phát**  
**Chuyên gia hướng dẫn : Chu Bá Long**  
**Giảng viên hướng dẫn : TS.Vũ Ngọc Thanh Sang**

*TP. Hồ Chí Minh, tháng 8 năm 2024*

# MỤC LỤC

<b>NHẬN XÉT CỦA CHUYÊN GIA DOANH NGHIỆP .....</b>	<b>iii</b>
<b>NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN.....</b>	<b>iv</b>
<b>LỜI MỞ ĐẦU .....</b>	<b>1</b>
<b>CHƯƠNG 1. GIỚI THIỆU.....</b>	<b>2</b>
<b>1.1.Giới thiệu công ty thực tập.....</b>	<b>2</b>
<b>1.2. Nhiệm vụ thực tập.....</b>	<b>2</b>
<b>CHƯƠNG 2. NỘI DUNG THỰC TẬP.....</b>	<b>3</b>
<b>2.1 Tìm hiểu, nghiên cứu dự án .....</b>	<b>3</b>
<b>2.1.1 Tìm hiểu nghiệp vụ của dự án EZSalon.....</b>	<b>3</b>
<b>2.1.2 Nghiên cứu về các công nghệ sử dụng trong dự án EZSalon.....</b>	<b>5</b>
<b>2.1.2.1 Framework Springboot và cách ứng dụng vào dự án.....</b>	<b>5</b>
<b>2.1.2.2 Framework Angular và cách ứng dụng vào dự án.....</b>	<b>8</b>
<b>2.2 Xây dựng chức năng mới .....</b>	<b>11</b>
<b>2.2.1 Tạo chức năng xem doanh thu trung bình của chi nhánh .....</b>	<b>11</b>
<b>2.2.2 Tạo chức năng xem số lượng khách hàng mới của chi nhánh .....</b>	<b>16</b>
<b>2.2.3 Tạo chức năng xem số lượng khách hàng không quay lại của chi nhánh .....</b>	<b>22</b>
<b>2.2.4 Tạo chức năng xem số lượng khách hàng sử dụng dịch vụ của chi nhánh .....</b>	<b>29</b>
<b>2.2.5 Cập nhật lại chức năng Dashboard.....</b>	<b>35</b>
<b>2.2.6 Cập nhật lại chức năng xem lại các tin nhắn SMS đã gửi đến khách hàng .....</b>	<b>38</b>
<b>CHƯƠNG 3: KẾT QUẢ THỰC TẬP .....</b>	<b>43</b>
<b>CHƯƠNG 4. KẾT LUẬN VÀ KIẾN NGHỊ .....</b>	<b>48</b>

## This image shows a full page of a handwriting practice worksheet. It consists of numerous horizontal dotted lines spaced evenly across the page, providing a guide for letter height and placement. There are no margins, text, or other markings on the page.

[illegible]

## LỜI MỞ ĐẦU

Lời đầu tiên, em xin gửi lời cảm ơn chân thành tới thầy Vũ Ngọc Thanh Sang đã tận tình giảng dạy và hướng dẫn em trong suốt quá trình học tập tại trường. Em cũng xin gửi lời cảm ơn tới Ban giám đốc và các anh chị nhân viên Công Ty Tư Vấn Khoa Học Công Nghệ An Phát đã tạo điều kiện cho em được tham gia thực tập tại công ty.

Trong thời đại công nghệ phát triển mạnh mẽ như hiện nay, ngành công nghệ thông tin (IT) đang ngày càng trở nên quan trọng và có vai trò to lớn trong mọi lĩnh vực của đời sống xã hội. Với niềm đam mê và mong muốn được học hỏi, trải nghiệm thực tế trong môi trường làm việc chuyên nghiệp, em đã có cơ hội thực tập tại Công Ty Tư Vấn Khoa Học Công Nghệ An Phát, một trong những công ty hàng đầu trong lĩnh vực phát triển phần mềm tại Việt Nam.

Trong thời gian thực tập, em đã được tham gia vào dự án EZSalon, một dự án phát triển phần mềm quản lý sản phẩm theo hình thức thương mại điện tử. Đây là một dự án có quy mô lớn và phức tạp. Với vai trò là một thực tập sinh, em được giao nhiệm vụ như là một lập trình viên chính thức của công ty, được xây dựng các chức năng mới bên cạnh đó là sửa lỗi, điều chỉnh cho các chức năng sẵn có.

Được thực tập tại Công Ty Tư Vấn Khoa Học Công Nghệ An Phát là một trải nghiệm vô cùng quý giá đối với em, đặc biệt là khi được tham gia dự án thực tế EZSalon. Trong quá trình thực tập, em đã được học hỏi rất nhiều kiến thức và kỹ năng mới, đồng thời được tiếp xúc với môi trường làm việc chuyên nghiệp, năng động. Thông qua dự án này, em đã có cơ hội phát triển bản thân, nâng cao kỹ năng làm việc nhóm, kỹ năng giao tiếp và thuyết trình, cũng như kỹ năng giải quyết vấn đề.

# CHƯƠNG 1. GIỚI THIỆU

## 1.1. Giới thiệu công ty thực tập

- Tên công ty: Công Ty Tư Vấn Khoa Học Công Nghệ An Phát
- Địa chỉ: 38/2C Nguyễn Văn Linh, KP 4, P.Bình Thuận, Quận 7, TP.HCM
- Số điện thoại: 02837700142
- Lĩnh vực hoạt động: Viết các chương trình phần mềm, sản xuất phần mềm
- Sản phẩm quan trọng của doanh nghiệp: Dự án EZSalon
- Phòng ban thực tập: Phòng công nghệ
- Vị trí thực tập: Intern Fullstack Developer
- Thông tin chuyên gia doanh nghiệp:
  - + Họ tên: Chu Bá Long
  - + Vị trí: Giám đốc, Tech Lead
  - + Email: chubalong@gmail.com
  - + Số điện thoại: 0919131343
- Nội dung sinh viên học hỏi khi thực tập tại doanh nghiệp:
  - + Thái độ và quy trình làm việc
  - + Java Spring Boot
  - + Ngôn ngữ lập trình TypeScript
  - + Framework Angular

## 1.2. Nhiệm vụ thực tập

- Tìm hiểu về những công nghệ được sử dụng trong dự án EZSalon
- Học tập văn hoá doanh nghiệp, rèn luyện thái độ làm việc chuyên nghiệp và nắm vững quy trình làm việc hiệu quả
- Thực hiện tìm hiểu nghiệp vụ, nghiên cứu các công nghệ sử dụng trong dự án
- Xây dựng chức năng mới bên cạnh sửa lỗi, điều chỉnh các chức năng cho dự án thực tế EZSalon

## CHƯƠNG 2. NỘI DUNG THỰC TẬP

### 2.1 Tìm hiểu, nghiên cứu dự án

#### 2.1.1 Tìm hiểu nghiệp vụ của dự án EZSalon

**Thời gian thực hiện:** Tuần 1 (Từ ngày 24/06/2024 đến ngày 28/06/2024)

**Vấn đề gặp phải:**

- Do đây là dự án mới, nên thiếu hụt về tài liệu dẫn đến mất nhiều thời gian để hiểu được các nghiệp vụ trong dự án EZSalon.
- Chưa hiểu được logic của chức năng

**Giải quyết vấn đề:**

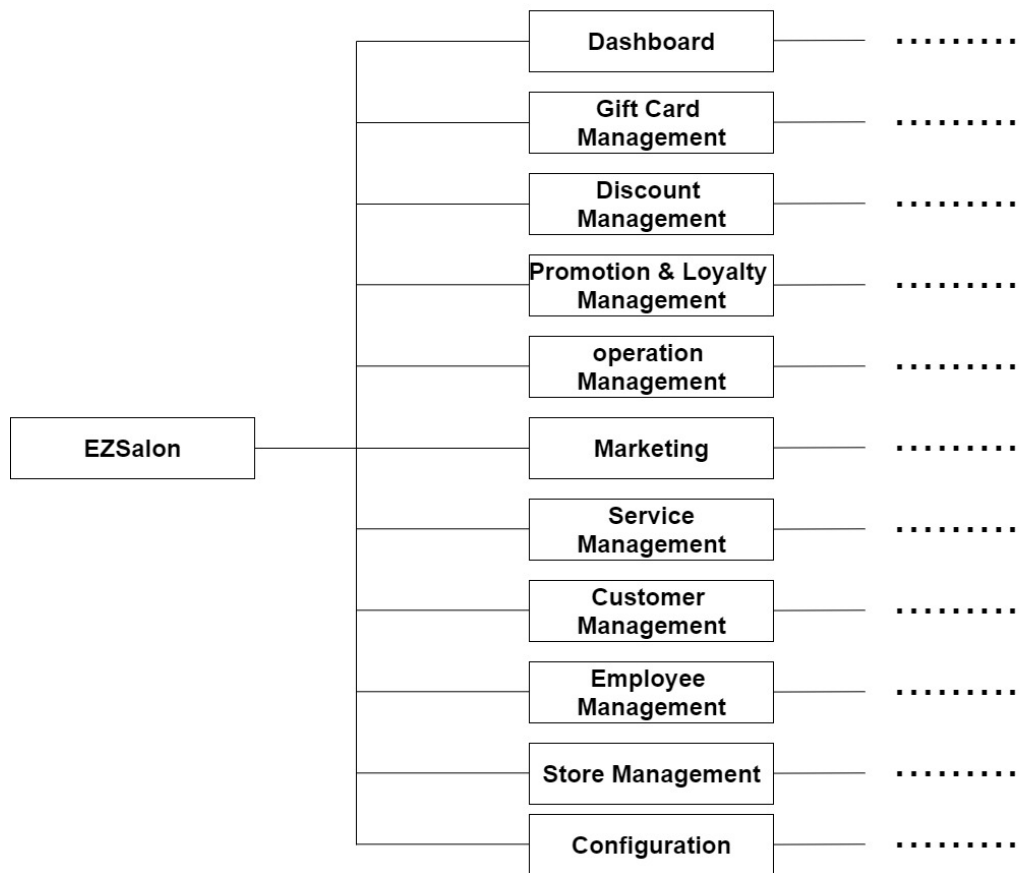
- Vẽ sơ đồ tổng quan các chức năng trong hệ thống
- Đọc kĩ logic code của từng chức năng, nghiệp vụ trong dự án

**Lý Do ra đời của dự án EZSalon:** Dự án EZSalon là dự án quản lý chuỗi các tiệm làm dịch vụ nails tại Mỹ. Ban đầu công ty này, sử dụng phần mềm của một bên cung cấp khác nhưng vì lý do không thể mở rộng các chức năng nên công Ty này đã yêu cầu xây dựng phần mềm quản lý ở Công Ty Tư Vấn Khoa Học Công Nghệ An Phát.



*Hình 2.1. Logo của phần mềm*

**Cơ cấu, cấu trúc của các chức năng, nghiệp vụ trong dự án EZSalon:** Như đã nói, dự án EZSalon là phần mềm quản lý hoạt động giao dịch của chuỗi các tiệm nails ở Mỹ. Cho nên các chức năng của dự án vẫn xoay quanh các nghiệp vụ cơ bản như quản lý (thêm, sửa, xóa) khách hàng, gói dịch vụ nail, nhân viên,.....Tuy nhiên, điểm nổi bật của dự án EZSalon là các công nghệ, kỹ thuật được sử dụng. Một số công nghệ, kỹ thuật có thể kể đến là thanh toán online, thời gian thực,...



*Hình 2.2. Sơ đồ phân rã chức năng(BFD) của phần mềm*

Sơ đồ phân rã chức năng ở Hình 2.2 mô tả đầy đủ và chi tiết các chức năng có trong hệ thống theo đúng cấp bậc của các chức năng đó. Ở mức ngữ cảnh, chỉ mô tả tổng quát về phần mềm. Mức 1 sơ đồ phân ra là sự mô tả các module chứa các chức năng thể hiện các nghiệp vụ của phần mềm. Ở mức này, Phần mềm sẽ có các module như sau:

- **Operation Management:** chứa các chức năng liên quan đến thời gian thực như: đặt lịch, kiểm tra các dịch vụ đang được dùng ở chi nhánh, các nhân viên nào đang trông việc ở tiệm.
- **Gift Card Management:** chứa các chức năng về các thẻ tín dụng sử dụng trong quá trình giao dịch, cho phép sử dụng các gói dịch vụ của chi nhánh.
- **Management:** chứa các thống kê, báo cáo, lịch sử các hóa đơn trong những lần giao dịch giữa phía khách hàng và phía cửa tiệm
- **Discount Management:** chứa các chức năng liên quan đến việc quản lý các đợt giảm giá của các gói dịch vụ của hệ thống của tiệm.



- **Promotion & Loyalty Management:** chứa các chức năng liên quan đến ưu đãi cho khách hàng quen thuộc của cửa tiệm. chức năng nào có thể tăng cấp bậc của khách hàng; thêm, sửa, xóa các ưu đãi cho khách hàng theo cấp bậc.
- **Marketing:** chứa các chức năng liên quan đến tin nhắn gửi cho khách hàng bên cạnh đó là quản lý các chiến dịch mà những tin nhắn liên quan đến.
- **Service Management:** chứa các chức năng về quản lý các gói dịch vụ và chi tiết các dịch vụ trong từng gói theo nghiệp vụ nails của hệ thống mà bên phía người thuê phần mềm đã yêu cầu trước đó.
- **Customer Management:** chứa các dịch vụ về quản lý nhóm khách hàng, khách hàng đã đến và sử dụng dịch vụ ở tiệm. Chức năng nào cho biết thông tin khách hàng và ưu đãi của từng nhóm khách hàng có trong hệ thống.
- **Employee Management:** chứa các chức năng liên quan đến quản lý nhân viên làm việc tại cửa tiệm. Chức năng này tập trung vào việc điều chỉnh thông tin của nhân viên như: thêm, xóa, sửa, tạo lịch làm việc, xem lịch sử làm việc ở tiệm.
- **Store Management:** chứa các chức năng quản lý các chi nhánh trong hệ thống như thêm, sửa, xóa cửa tiệm. Ngoài ra còn cho biết lịch làm việc cũng như lịch đóng cửa của các chi nhánh đang có của cửa tiệm.
- **Configuration:** đây là module cuối cùng có trong hệ thống, đây là module chứa các chức năng liên quan đến điều chỉnh các cấu hình như cấu hình tin nhắn sms, mail; cấu hình chung cho hệ thống...

## 2.1.2 Nghiên cứu về các công nghệ sử dụng trong dự án EZSalon

### 2.1.2.1 Framework Springboot và cách ứng dụng vào dự án

**Thời gian thực hiện:** Tuần 2(Từ ngày 01/07/2024 đến ngày 05/07/2024)

**Vấn đề gặp phải:**

- Do chỉ mới là thực tập sinh, nên thiếu hụt về sự hiểu biết cũng như kinh nghiệm sử dụng các framework sẵn có cho nên sẽ cần thời gian để nghiên cứu.

**Giải quyết vấn đề:**

- Tìm hiểu cấu trúc framework Springboot
- Đọc kĩ logic code có sẵn trong dự án để biết cách sử dụng Springboot

**Khái niệm Spring boot:** 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

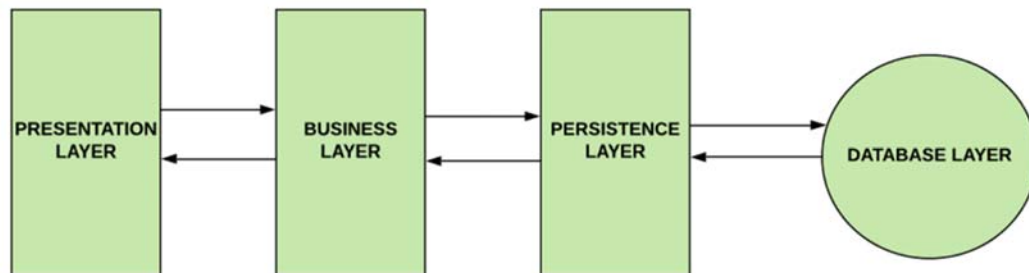
cấu hình mặc định cho một số thư viện và bộ công cụ hỗ trợ xây dựng, triển khai, quản lý ứng dụng Spring-based.

Cách Spring Boot hoạt động nhằm tối ưu hóa quy trình phát triển ứng dụng Java. Điều này sẽ giúp nhà phát triển tập trung vào việc xây dựng tính năng chính của ứng dụng mà không cần phải lo lắng về cấu hình phức tạp.

### Kiến Trúc Spring boot:

Kiến trúc Spring Boot có 4 lớp:

- Lớp Presentation
- Lớp Business
- Lớp Persistence
- Lớp Database



*Hình 2.3: Các lớp đặc trưng của springboot*

#### Lớp Presentation:

- + Lớp này nằm trên cùng của kiến trúc Spring Boot. Nó chịu trách nhiệm:
- + Thực hiện authentication (xác thực).
- + Chuyển đổi dữ liệu JSON thành đối tượng (và ngược lại).
- + Xử lý các HTTP request
- + Truyền authentication tới lớp bussiness
- + Lớp presentation tương ứng với class Controller. Class Controller xử lý tất cả + Các REST API request (GET, POST, PUT, DELETE, PATCH) đến từ client.

#### Lớp Business:

- + Lớp này chịu trách nhiệm:
- + Thực hiện validation.
- + Thực hiện authorization (ủy quyền).
- + Xử lý các logic và quy tắc nghiệp vụ.
- + Lớp này tương ứng với class Service, là nơi xử lý logic nghiệp vụ. Nếu bạn băn khoăn không biết “logic nghiệp vụ” là gì theo ý của chúng tôi, bạn có thể xem một bài thảo

luận thú vị trên StackExchange. Ngăn gọn thì logic nghiệp vụ trong kỹ nghệ phần mềm là những gì mà chúng ta xác định phần mềm cần phải làm. Một ví dụ là validation. Nếu bạn muốn validate gì đó, nó phải được thực hiện trong class Service.

- + Lớp Business giao tiếp với lớp Presentation và lớp Persistence.

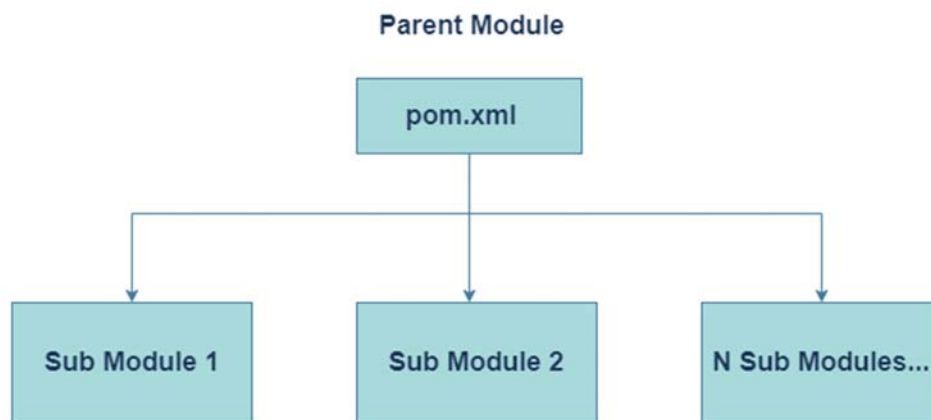
#### **Lớp Persistence:**

- + Lớp này chịu trách nhiệm:
- + Chứa các logic lưu trữ
- + Lấy các đối tượng và chuyển đổi thành các hàng trong database (và ngược lại).
- + Lớp này tương ứng với interface Repository. Chúng ta viết các truy vấn tới database trong interface này.
- + Lớp Persistence là lớp duy nhất giao tiếp với lớp Business và lớp Database.

#### **Lớp Database**

- + Lớp này chịu trách nhiệm:
- + Thực hiện các tác vụ với database (chủ yếu là CRUD).
- + Lớp này đơn giản là database trong thực tế bạn dùng trong ứng dụng của mình.

#### **Kiến Trúc Spring boot:**



*Hình 2.4: Sơ đồ kiến trúc cho ứng dụng multi module*

Thay vì trước đây source viết trong 1 module thì giờ đây spring boot cho phép các nhà phát triển ứng dụng chia nhỏ module ra thành nhiều module con tùy thuộc vào mong muốn người dùng. Và khái niệm multi module trong spring boot cũng được sử dụng trong dự án EZSalon Hình 2.5 bên dưới.



Hình 2.5: Cách dự án EZSalon ứng dụng multi module trong spring boot.

### 2.1.2.2 Framework Angular và cách ứng dụng vào dự án

**Thời gian thực hiện:** Tuần 3(Từ ngày 08/07/2024 đến ngày 12/07/2024)

#### Vấn đề gặp phải:

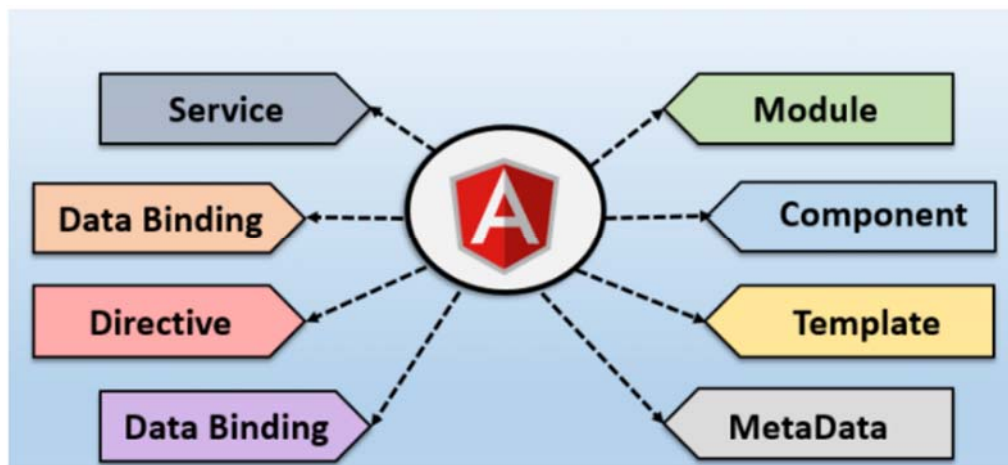
- Do chỉ mới là thực tập sinh, nên thiếu hụt về sự hiểu biết cũng như kinh nghiệm sử dụng các framework sẵn có cho nên sẽ cần thời gian để nghiên cứu

#### Giải quyết vấn đề:

- Tìm hiểu cấu trúc framework Angular
- Đọc kĩ logic code có sẵn trong dự án để biết cách sử dụng Angular

**Khái niệm Angular:** Angular là một JavaScript framework sử dụng để viết giao diện web. Chúng được phát triển bởi Google. Nhờ Angular, hiệu suất xây dựng ứng dụng sẽ tăng gấp nhiều lần. Nếu developer không sử dụng Angular, họ có thể mất rất nhiều thời gian để có thể hoàn thiện giao diện web. Đặc điểm của Angular là Single-page được hiểu là một trang đơn chính có thể gọi đến ở mọi nơi.

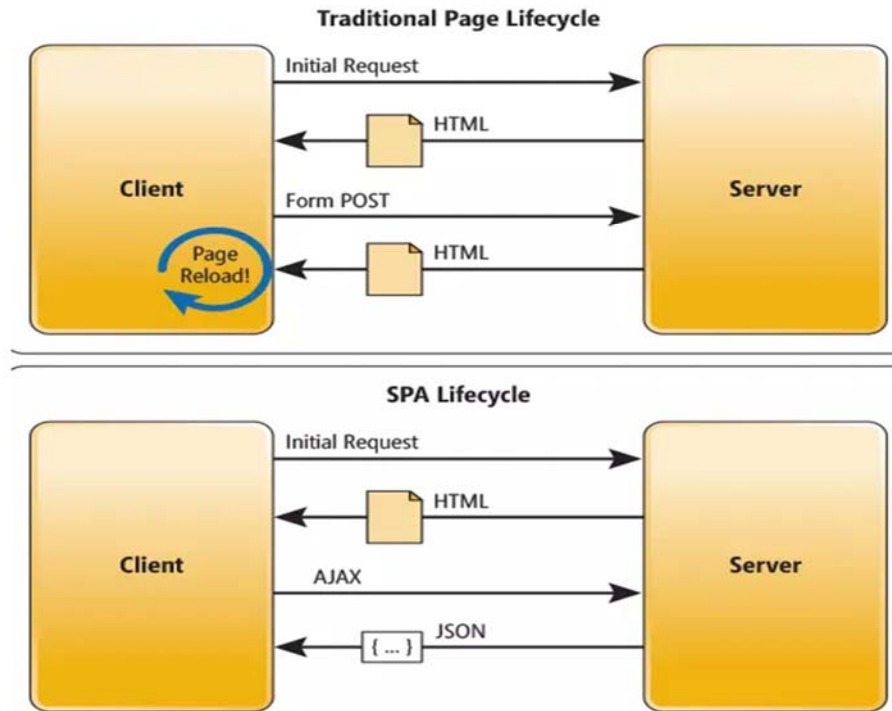
#### Kiến trúc Angular:



Hình 2.6: Những thành phần có trong kiến trúc Angular

- **Module:** Mỗi ứng dụng Angular được gọi là một module và bản thân Angular có riêng một module dùng để quản lý các module khác có tên là Root Module hay NgModule. Root Module thường được đặt tên là AppModule, ngoài root ra thì tùy ứng dụng mà sẽ có thêm các module khác, chúng ta sẽ tìm hiểu về root module trong bài sau.
- **Component:** Component là một khối xây dựng cơ bản trong Angular. Nó đại diện cho một phần giao diện người dùng (UI) cụ thể trong ứng dụng. Mỗi component có thể có logic riêng, dữ liệu đầu vào và đầu ra, và có thể được sử dụng lại trong suốt ứng dụng.
- **Template:** Directive là một đối tượng giúp chúng ta thay đổi hành vi, giao diện hoặc thuộc tính của một phần tử HTML hoặc component khác. Directives có thể hiểu như là các đoạn mã typescript (hoặc javascript) kèm theo cả HTML và khi gọi thì gọi như là HTML luôn.
- **MetaData:** Metadata (siêu dữ liệu) là những thông tin giúp Angular xử lý các lớp. Trong đoạn code ví dụ về Component ở trên, đó chỉ là một lớp bình thường viết bằng TypeScript, không có sự xuất hiện của Angular trong này. Muốn Angular hiểu được đó là một lớp dành cho Angular thì chúng ta phải khai báo metadata
- **Service:** Service là các lớp có khả năng thực hiện một số chức năng thường dùng, nói đơn giản thì chúng giống như thư viện vậy. Một số service phổ biến là: logging service, data service, message bus, tax calculator, application configuration. Ví dụ lớp Logger cho phép chúng ta in các đoạn code báo lỗi, cảnh báo
- **Data Binding:** Data binding trong angular như: Interpolation, Attribute binding, Class and style binding, Event binding, Property binding, Two-way binding. Đây được xem là liên kết dữ liệu hai chiều khi các biến giá trị ở DOM HTML được cập nhật hai chiều khi giá trị của biến thay đổi.
- **Directive:** Directives là một đối tượng giúp chúng ta dễ dàng thay đổi một đối tượng khác và cách áp dụng rất đơn giản và linh hoạt. Directives có thể hiểu như là các đoạn mã typescript (hoặc javascript) kèm theo cả HTML và khi gọi thì gọi như là HTML luôn.

**Cơ chế sử dụng Single Page Application mạnh mẽ trong Angular:** Single Page Application là 1 trang web hay 1 ứng dụng web, khi tất cả những thao tác xử lý đều được diễn ra trên 1 trang duy nhất.

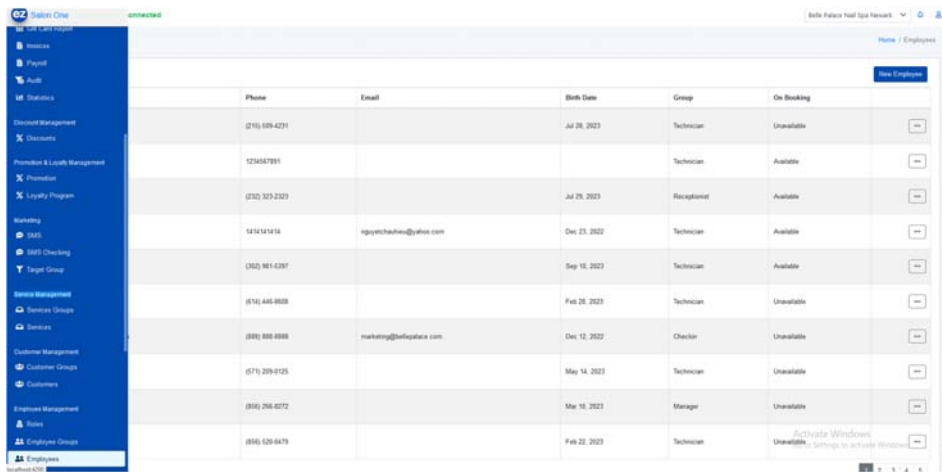


Hình 2.7: Hình ảnh minh họa giữa sự khác biệt khi render data truyền thống và sử dụng Single page

Với các trang web truyền thống, khi user request 1 trang web thì server sẽ tính toán và trả về trang web đó dưới dạng mã HTML và hầu như không có bất kỳ sự liên kết nào giữa 2 yêu cầu gần nhau. Do đó khi có nhiều yêu cầu được đưa ra thì quá trình tính toán sẽ diễn ra lâu hơn. Và trả về 1 trang web hoàn chỉnh.

Còn với SPA thì khi user request lần đầu thì server sẽ tính toán và trả về trang web dưới dạng HTML. tuy nhiên ở những lần yêu cầu tiếp theo thì client chỉ phải request nhưng phần nào 1 cần (AJAX) và server sẽ trả dữ liệu với dạng JSON sẽ giúp ngắn thời gian truyền tải. Hay nói cách khác SPA chỉ load phần trang cần thiết.

Trong dự án EZSalon, Single page được sử dụng để làm trang side bar chính cho trang web, thanh side bar này sẽ luôn đi theo trang web dù ở bất kỳ trang nào. Ngoài ra single page còn được ứng dụng vào các bảng(table), mẫu báo cáo, mẫu tìm kiếm,... trong ứng dụng. Điều đó được minh họa ở hình 2.7.



The screenshot shows the EZSalon application interface. On the left is a blue sidebar menu with various management options. The main area displays a table of employees.

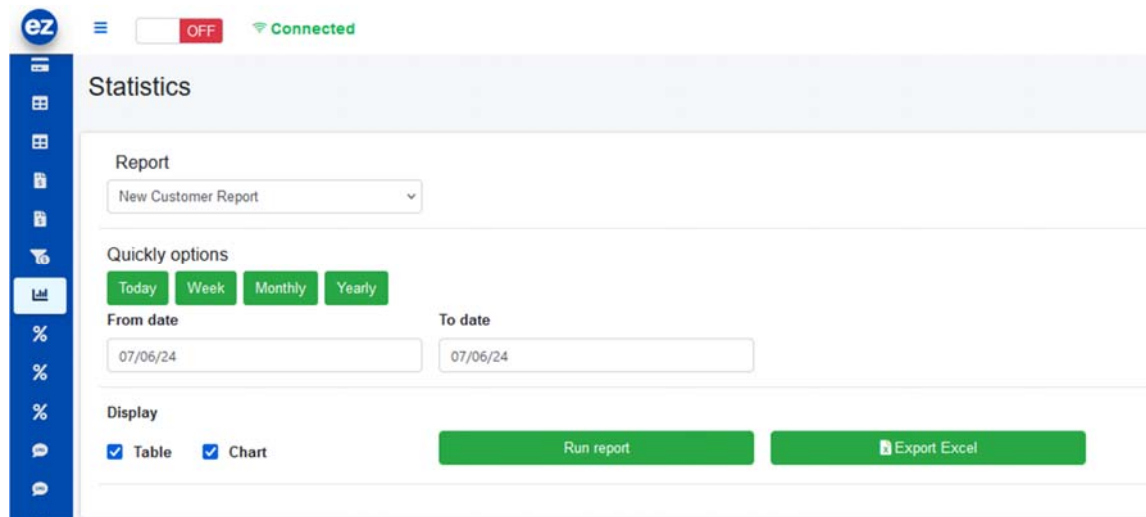
	Phone	Email	Birth Date	Group	On Booking	
	(715) 559-4271		Jul 28, 2023	Technician	Unavailable	[-]
	1234567891			Technician	Available	[-]
	(720) 323-2323		Jul 28, 2023	Receptionist	Available	[-]
	1434141414	nguyenchauha@ezsalon.com	Dec 23, 2022	Technician	Available	[-]
	(762) 981-6387		Sep 16, 2023	Technician	Available	[-]
	(514) 448-8888		Feb 28, 2021	Technician	Unavailable	[-]
	(888) 888-8888	marketing@ezsalon.com	Dec 12, 2022	Checker	Unavailable	[-]
	(571) 209-8125		May 14, 2023	Technician	Unavailable	[-]
	(800) 268-0772		Mar 10, 2023	Manager	Unavailable	[-]
	(888) 529-0479		Feb 25, 2023	Technician	Unavailable	[-]

Hình 2.8: Cách dự án EZSalon sử dụng tính chất single page trong angular

## 2.2 Xây dựng chức năng mới

### 2.2.1 Tạo chức năng xem doanh thu trung bình của chi nhánh

Theo mô tả từ phản hồi của khách hàng, họ cần có 1 chức năng để cung cấp thêm thông tin về doanh thu cho các chi nhánh có trong hệ thống EZSalon. Chức năng nằm trong module Management sẽ cho biết đầy đủ thông tin về doanh thu theo ngày, tháng, năm dưới dạng bảng và biểu đồ. Ngoài ra còn có thể xuất dạng bảng thành file excel để phục vụ cho người dùng sử dụng.



The screenshot shows the EZSalon application interface with the Statistics section. It includes a sidebar menu, a header with the EZ logo and status indicators, and a main content area for generating reports.

**Statistics**

Report:

Quickly options:

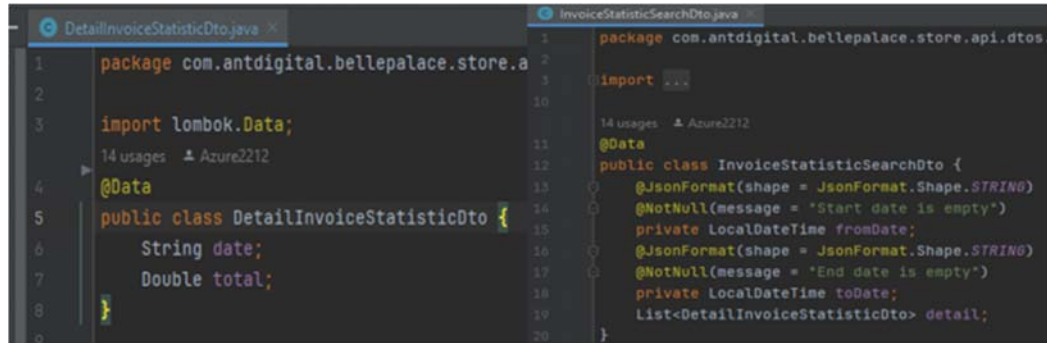
From date:  To date:

Display: ☒ Table ☒ Chart

Hình 2.9. Mẫu thống kê cho chức năng xem doanh thu trung bình của chi nhánh

**Thời gian thực hiện:** Tuần 4(Từ ngày 15/07/2024 đến ngày 19/07/2024)

**Lập Trình Backend bằng Springboot:** Đầu tiên để xây dựng luồng xử lý hoạt động cho chức năng xem doanh thu trung bình của chi nhánh phía backend của phần mềm. Ta cần xây dựng theo kiến trúc mà công ty đã xây dựng sẵn theo thứ tự xây dựng: Dto, Controller(API), Service, Repository hoặc DAL để truy xuất vào cơ sở dữ liệu. Vì phần thao tác đến cơ sở dữ liệu là phần tài sản của công ty cho nên phần đó sẽ không được mô tả trong báo cáo.



```
DetailInvoiceStatisticDto.java
1 package com.antdigital.bellepalace.store.a
2
3 import lombok.Data;
4
5 @Data
6 public class DetailInvoiceStatisticDto {
7     String date;
8     Double total;
9 }

InvoiceStatisticSearchDto.java
1 package com.antdigital.bellepalace.store.api.dtos.
2
3 import
4
5
6
7
8
9
10
11 @Data
12 public class InvoiceStatisticSearchDto {
13     @JsonFormat(shape = JsonFormat.Shape.STRING)
14     @NotNull(message = "Start date is empty")
15     private LocalDateTime fromDate;
16     @JsonFormat(shape = JsonFormat.Shape.STRING)
17     @NotNull(message = "End date is empty")
18     private LocalDateTime toDate;
19     List<DetailInvoiceStatisticDto> detail;
20 }
```

Hình 2.10. Các lớp Dto sử dụng trong chức năng xem doanh thu trung bình của chi nhánh

Dựa trên hình 2.10, sẽ có 2 Dto được xây dựng cho chức năng xem doanh thu trung bình là DetailInvoiceStatisticDto lớp Dto cho biết ngày và số lượng doanh thu trung bình của ngày đó. Và lớp InvoiceStatisticSearchDto cho biết danh sách các ngày và doanh thu trung bình của từng ngày tương ứng dựa trên ngày bắt đầu và ngày kết thúc mà người dùng chọn bên FE gửi qua.



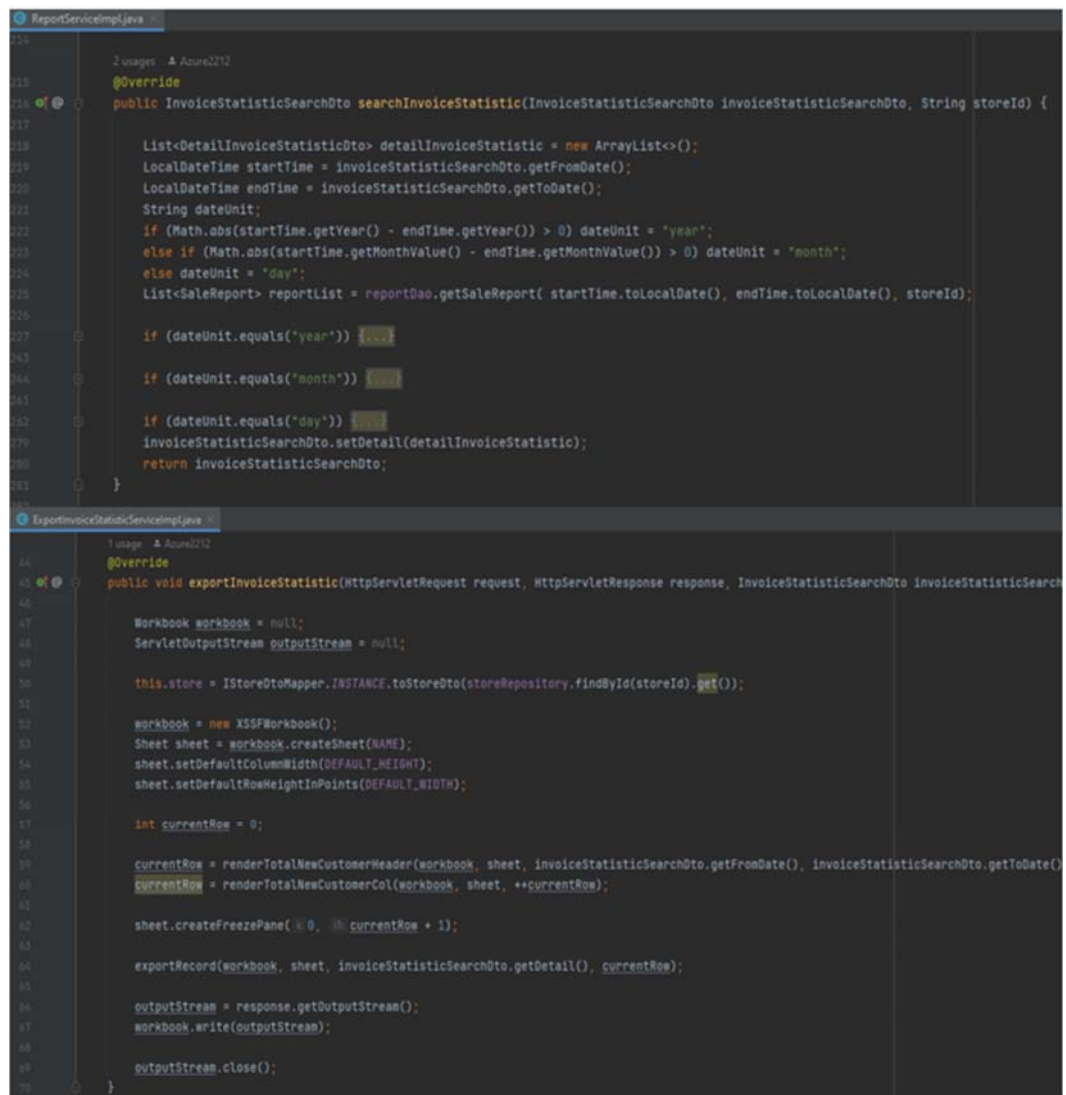
```
ReportController.java
229
230 @PostMapping("/searchInvoicesStatistic")
231 public ResponseEntity<> searchInvoiceStatistic(@RequestBody InvoiceStatisticSearchDto invoiceStatisticSearchDto,
232                                               @RequestParam(required = false) String storeId) {...}
233
234
235 @PostMapping("/exportInvoiceStatistic")
236 public void exportInvoiceStatistic(HttpServletRequest request, HttpServletResponse response,
237                                   @RequestBody InvoiceStatisticSearchDto invoiceStatisticSearchDto,
238                                   @RequestParam(required = false) String storeId) throws IOException {...}
239
240
241
242
243
244
245
```

Hình 2.11. Các Controller được xây dựng chức năng xem doanh thu trung bình của chi nhánh

Dựa trên hình 2.11, ta thấy sẽ có 2 controller được xây dựng. Hàm dùng để lấy báo cáo doanh thu trung bình là searchInvoiceStatistic bên phía BE sẽ thu thập dữ liệu và gửi sang phía FE. Trong khi đó hàm exportInvoiceStatistic sẽ là API thực hiện chức năng xuất ra excel từ báo cáo đó.



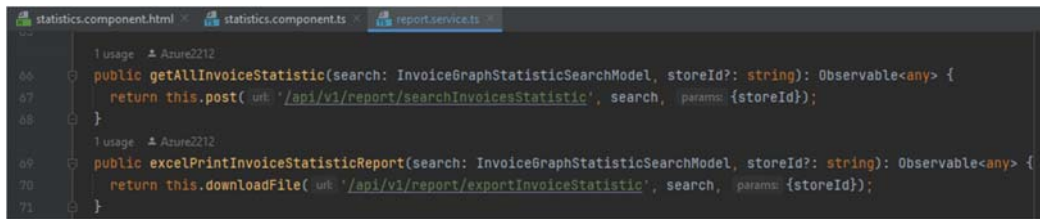
Sau khi xây dựng hoàn tất các lớp Dto và xác định được các controller(API) sẽ có trong chức năng xem doanh thu trung bình. Phần tiếp sau đó chính là xây dựng các luồng xử lý cho chức năng tương ứng. Hàm để lấy cáo thông tin về doanh thu trung bình của chi nhánh theo từng ngày, thời gian người dùng chọn bên FE gửi qua là hàm `searchInvoiceStatistic` trong lớp service `ReportServiceImpl`. Hàm này chỉ đơn giản là lấy hết tất cả hóa đơn của chi nhánh trong thời gian FE gửi qua sau đó, xác định hiện báo cáo theo đơn vị này(ngày, giờ hay năm) sau đó gom các hóa đơn theo từng cụm đơn vị thời gian rồi tính tổng. cuối cùng là trả danh sách các ngày chứa tổng doanh thu trung bình của ngày tương ứng về `SearchInvoiceStatistic` bên Controller và controller trả về FE. Hàm xuất báo cáo ra file excel là `exportInvoiceStatistic` trong lớp service `ExportInvoiceStatisticServiceImpl`. Tương tự như hàm `searchInvoiceStatistic` đã trình bày trước đó, `exportInvoiceStatistic` sẽ lấy các thông tin về ngày và doanh thu trung bình theo từng ngày tương ứng nhưng khác ở chỗ `exportInvoiceStatistic` sẽ không trả về Controller mà trực tiếp xuất file cho người dùng.



Hình 2.12. Các Hàm thực hiện nghiệp vụ được xây dựng cho chức năng xem doanh thu trung bình của chi nhánh

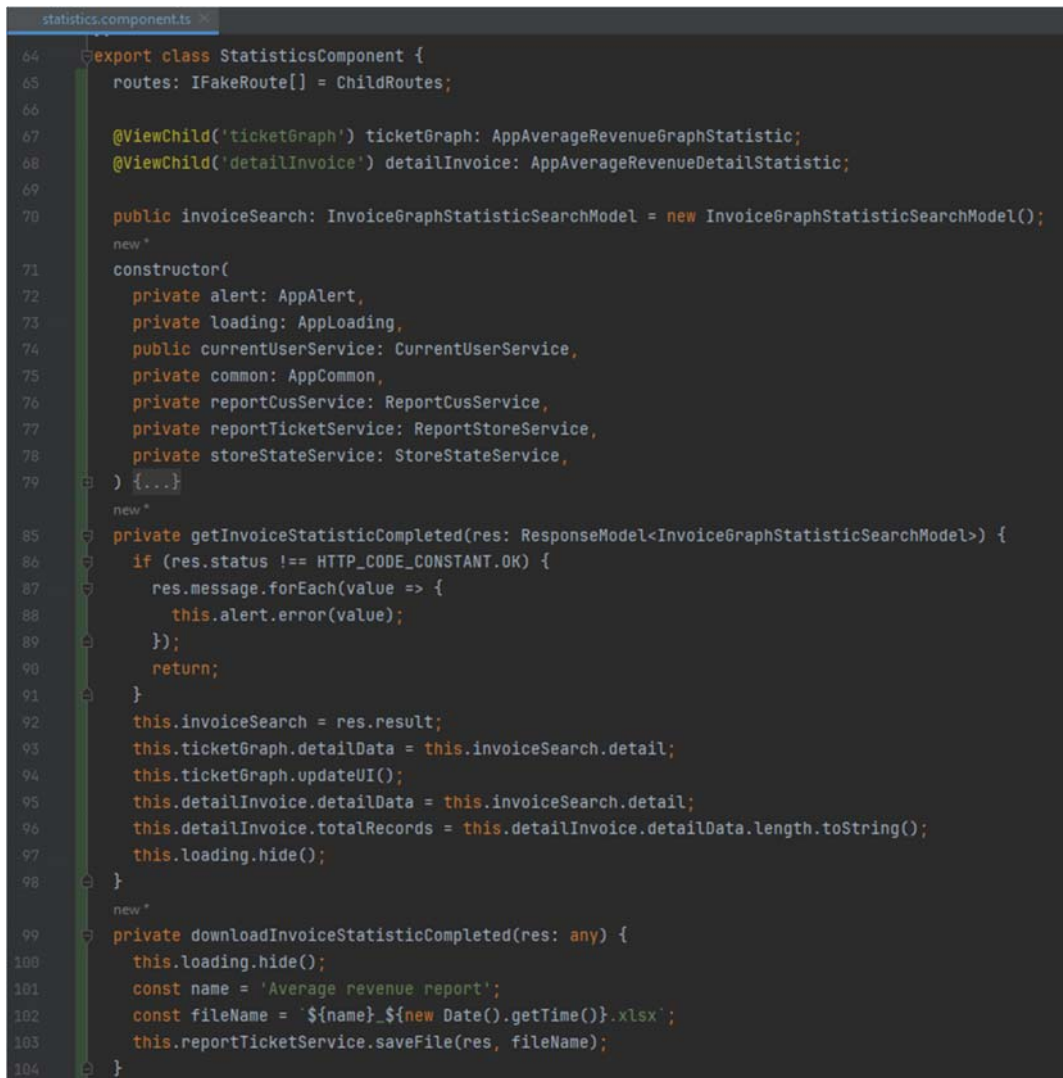
### Lập Trình Fontend bằng Angular:

Ở phần lập trình Backend bằng Spring boot không lâu trước đó đã hoàn thành việc xây dựng các API cho chức năng, để lấy được thông tin cần cho chức năng xem doanh thu trung bình thông qua API và lớp Controller đã thực hiện, thì phía Fontend bằng Angular sẽ phải xây dựng các hàm để truy xuất đến các API đó được thể hiện hình 2.13.



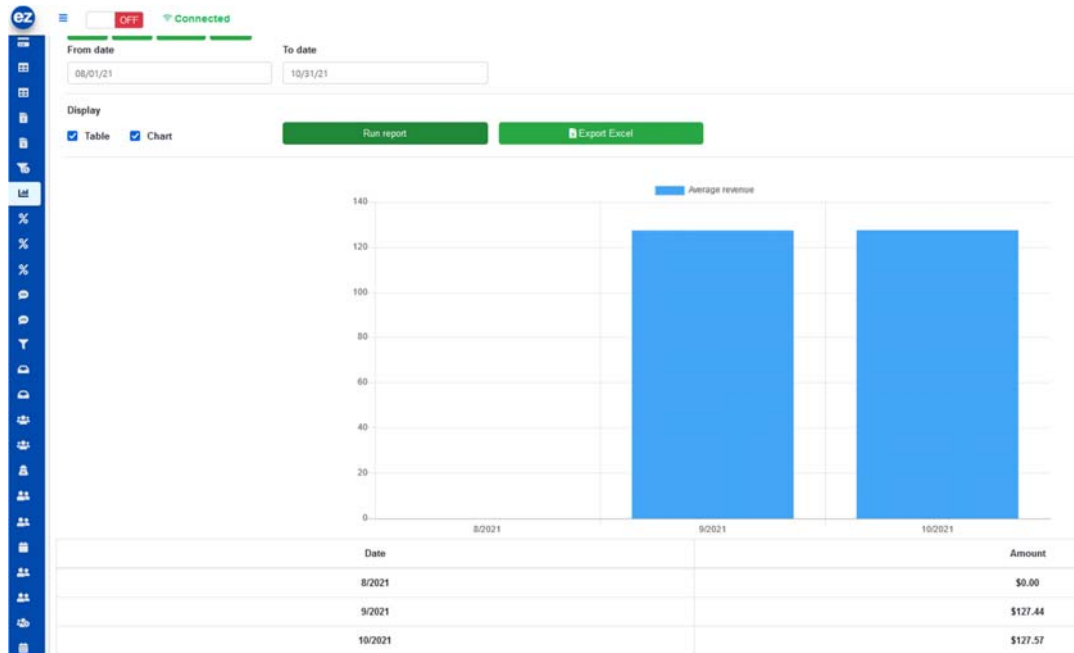
Hình 2.13. Các Hàm gọi đến API để lấy dữ liệu cho chức năng xem doanh thu trung bình của chi nhánh

Trong hình 2.13 cho thấy, hàm getAllInvoiceStatistic sẽ thực hiện lấy thông tin cho báo cáo doanh thu trung bình dựa trên mã chi nhánh(storeId) đã được xây dựng bên BackEnd. Trong khi đó, hàm excelPrintInvoiceStatisticReport sẽ in cáo thông tin cho báo cáo dựa trên mã chi nhánh thành dạng file Excel thông qua API đã được xây dựng trước đó.



Hình 2.14. Lớp StatisticsComponent xây dựng FontEnd cho chức năng xem doanh thu trung bình của chi nhánh

Lớp StatisticsComponent trong hình 2.14 chứa các biến ticketGraph và detailInvoice với tác dụng lần lượt là hiển thị báo cáo dưới dạng đồ thị cột và dạng bảng chi tiết. Hàm getInvoiceStatisticCompleted để lấy dữ liệu thông qua hàm đã xây dựng trong hình 7 là getAllInvoiceStatistic. Trong khi đó hàm xuất báo cáo ra dạng excel là downloadInvoiceStatisticCompleted.

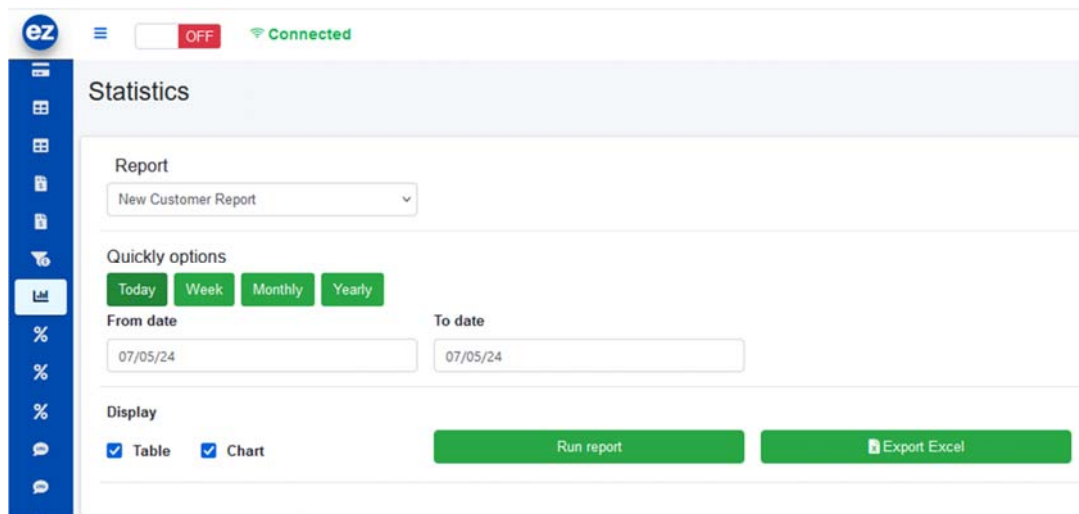


Hình 2.15. chức năng xem doanh thu trung bình của chi nhánh sau khi hoàn thiện

Hình 2.15, cho thấy chức năng xem doanh thu trung bình của chi nhánh khi bấm vào “run report” sau khi hoàn thiện. Chức năng này sẽ cho người dùng xem báo cáo dưới dạng bảng hoặc biểu đồ hoặc cả 2 tùy thuộc vào người dùng chọn trong “Display”. Và xuất Excel tùy ý khi bấm “Export Excel”.

## 2.2.2 Tạo chức năng xem số lượng khách hàng mới của chi nhánh

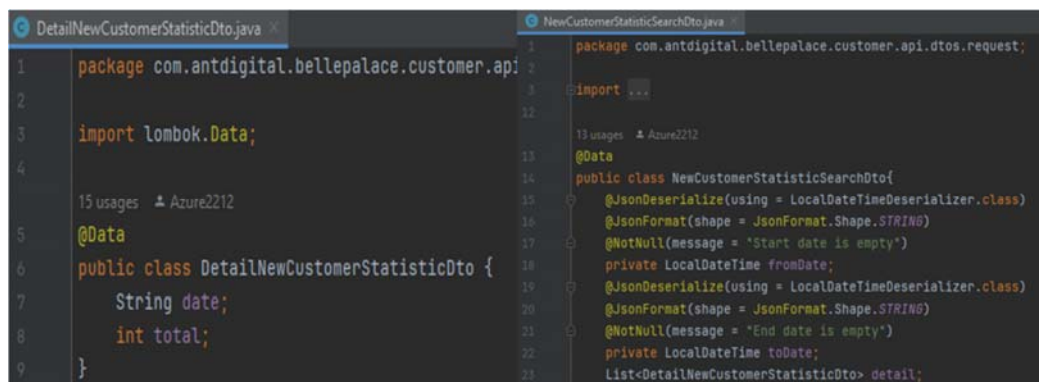
Theo mô tả từ phản hồi của khách hàng, họ cần có 1 chức năng để cung cấp thêm thông tin về số lượng khách hàng mới đến sử dụng dịch vụ lần đầu cho các chi nhánh có trong hệ thống EZSalon. Chức năng nằm trong module Management sẽ cho biết đầy đủ thông tin về số lượng khách hàng mới theo ngày, tháng, năm dưới dạng bảng và biểu đồ. Ngoài ra còn có thể xuất dạng bảng thành file excel để phục vụ cho người dùng sử dụng.



Hình 2.16. Mẫu thống kê số lượng khách hàng mới tại chi nhánh

**Thời gian thực hiện:** Tuần 5(Từ ngày 22/07/2024 đến ngày 26/07/2024)

**Lập Trình Backend bằng Springboot:** Đầu tiên để xây dựng luồng xử lý hoạt động cho chức năng xem thống kê số lượng khách hàng mới tại chi nhánh phía backend của phần mềm. Ta cần xây dựng theo kiến trúc mà công ty đã xây dựng sẵn theo thứ tự xây dựng: Dto, Controller(API), Service, Repository hoặc DAL để truy xuất vào cơ sở dữ liệu như trong chức năng xem doanh thu trung bình của chi nhánh đã được xây dựng trước đó. Vì phần thao tác đến cơ sở dữ liệu là phần tài sản của công ty cho nên phần đó sẽ không được mô tả trong báo cáo.



Hình 2.17. Các lớp Dto sử dụng trong chức năng xem số lượng khách hàng mới của chi nhánh

Dựa trên hình 2.17, sẽ có 2 Dto được xây dựng cho chức năng xem doanh thu trung bình là DetailNewCustomerStatisticDto lớp Dto cho biết ngày và số lượng khách hàng mới của ngày đó. Và lớp NewCustomerSearchDto cho biết danh sách các ngày và

số lượng khách hàng mới của từng ngày tương ứng dựa trên ngày bắt đầu và ngày kết thúc mà người dùng chọn bên FE gửi qua.



```
ReportController.java
144 @PostMapping("/{searchNewCustomerStatistic}")
145 public ResponseEntity<?> searchNewCustomerByDate(@RequestBody NewCustomerStatisticSearchDto newCustomerStatisticSearchDto,
146                                                  @RequestParam(required = false) String storeId) ...
154
155 @PostMapping("/{exportNewCustomerStatistic}")
156 public void exportNewCustomerStatistic(HttpServletRequest request, HttpServletResponse response,
157                                       @RequestBody NewCustomerStatisticSearchDto newCustomerStatisticSearchDto,
158                                       @RequestParam(required = false) String storeId) throws IOException {...}
```

Hình 2.18. Các Controller được xây dựng chức năng xem thống kê số lượng khách hàng mới

Dựa trên hình 2.18, ta thấy sẽ có 2 controller được xây dựng. Hàm dùng để lấy xem thống kê số lượng khách hàng mới là searchNewCustomerByDate bên phía BE sẽ thu thập dữ liệu và gửi sang phía FE. Trong khi đó hàm exportNewCustomerStatistic sẽ là API thực hiện chức năng xuất ra excel từ thống kê đó.

Sau khi xây dựng hoàn tất các lớp Dto và xác định được các controller (API) sẽ có trong chức năng xem thống kê số lượng khách hàng mới. Phần tiếp sau đó chính là xây dựng các luồng xử lý cho chức năng tương ứng. Hàm để lấy thống kê số lượng khách hàng mới của chi nhánh theo từng ngày, thời gian người dùng chọn bên FE gửi qua là hàm searchNewCustomer trong lớp service ReportServiceImpl. Hàm này chỉ đơn giản là lấy hết tất cả khách hàng được tạo từ chi nhánh trong thời gian FE gửi qua sau đó, xác định hiện báo cáo theo đơn vị này(ngày, giờ hay năm) sau đó gom các khách hàng theo từng cụm đơn vị thời gian rồi tính tổng. Cuối cùng là trả về danh sách các ngày chứa tổng khách hàng mới của ngày tương ứng về searchNewCustomerByDate bên controller và controller trả về FE. Hàm xuất báo cáo ra file excel là exportNewCustomerStatistic trong lớp ExportCustomerStatisticServiceImpl. Tương tự như hàm searchNewCustomerByDate đã trình bày trước đó, exportNewCustomerStatistic sẽ lấy các thông tin về số lượng khách hàng mới theo từng ngày tương ứng nhưng khác ở chỗ exportNewCustomerStatistic sẽ không trả về Controller mà trực tiếp xuất file cho người dùng.

```

ReportStatisticImpl.java
A Azumi2712
@Override
public NewCustomerStatisticSearchDto searchNewCustomer(NewCustomerStatisticSearchDto newCustomerStatisticSearchDto
    , String storeId) {
    List<DetailNewCustomerStatisticDto> detailCustomerLastTime = new ArrayList<>();
    LocalDateTime startTime = newCustomerStatisticSearchDto.getFromDate();
    LocalDateTime endTime = newCustomerStatisticSearchDto.getToDate().with(LocalTime.MAX);
    String dateUnit;
    if (Math.abs(startTime.getYear() - endTime.getYear()) > 0) dateUnit = "year";
    else if (Math.abs(startTime.getMonthValue() - endTime.getMonthValue()) > 0) dateUnit = "month";
    else dateUnit = "day";
    List<Customer> listCustomerEntity = this.customerRepository.searchAllNewCustomerByDate(startTime, endTime, storeId);
    List<CustomerDto> customerDtos = ICustomerDtoMapper.INSTANCE.toCustomerDtoList(listCustomerEntity);
    if (dateUnit.equals("year")) {
        int yearStart = startTime.getYear();
        int yearEnd = endTime.getYear();
        for (int i = yearStart; i <= yearEnd; i++) {
            final int year = i;
            List<CustomerDto> customerDtoList = customerDtos.stream()
                .filter(j -> j.getCreateDate().getYear() == year)
                .collect(Collectors.toList());
            DetailNewCustomerStatisticDto detailNewCustomerStatisticDto = new DetailNewCustomerStatisticDto();
            detailNewCustomerStatisticDto.setDate(String.valueOf(i));
            detailNewCustomerStatisticDto.setTotal(customerDtoList.size());
            detailCustomerLastTime.add(detailNewCustomerStatisticDto);
        }
    }
    if (dateUnit.equals("month")) {
    }
    if (dateUnit.equals("day")) {
    }
    newCustomerStatisticSearchDto.setDetail(detailCustomerLastTime);
    return newCustomerStatisticSearchDto;
}

ExportCustomerStatisticImpl.java
A Azumi2712
@Override
public void exportNewCustomerStatistic(HttpServletRequest request, HttpServletResponse response, NewCustomerStatisticSearchDto newCustomerStatisticSearchDto) {
    Workbook workbook = null;
    ServletOutputStream outputStream = null;
    try {
        this.store = storeRequestService.getStore(request, storeId).getResult();
        workbook = new XSSFWorkbook();
        Sheet sheet = workbook.createSheet(NAME);
        sheet.setDefaultColumnWidth(DEFAULT_COLUMN_WIDTH);
        sheet.setDefaultRowHeightInPoints(DEFAULT_ROW_HEIGHT);
        int currentRow = 0;
        currentRow = renderTotalNewCustomerHeader(workbook, sheet, newCustomerStatisticSearchDto.getFromDate(), newCustomerStatisticSearchDto.getToDate(), currentRow);
        sheet.createFreezePane(0, currentRow + 1);
        exportRecord(workbook, sheet, newCustomerStatisticSearchDto.getDetail(), currentRow);
        outputStream = response.getOutputStream();
        workbook.write(outputStream);
        outputStream.close();
    } catch (JAXBException e) {
        throw new RuntimeException(e);
    }
}

```

Hình 2.19. Các Hàm thực hiện nghiệp vụ được xây dựng cho chức năng xem số lượng khách hàng mới của chi nhánh



## Lập Trình Fontend bằng Angular:

Ở phần lập trình Backend bằng Spring boot cho chức năng xem số lượng khách hàng mới của chi nhánh đã hoàn thành việc xây dựng các API cho chức năng. Để lấy được thông tin cần cho chức năng ấy thông qua các API và lớp Controller đã thực hiện, thì phía Fontend bằng Angular sẽ phải xây dựng các hàm để truy xuất đến các API đó được thể hiện hình 2.20.



```
53 public getNewCustomerStatistic(search: NewCustomerGraphStatisticSearchModel, storeId?: string): Observable<any> {  
54     return this.post('/api/v1/report/searchNewCustomerStatistic', search, {storeId});  
55 }  
56  
57 public excelPrintNewCustomerStatistic(search: NewCustomerGraphStatisticSearchModel, storeId?: string): Observable<any> {  
58     return this.downloadFile('/api/v1/report/exportNewCustomerStatistic', search, {storeId});  
59 }
```

*Hình 2.20. Các Hàm gọi đến API để lấy dữ liệu cho chức năng xem số lượng khách hàng mới của chi nhánh*

Trong hình 2.20 cho thấy, hàm getNewCustomersStatistic sẽ thực hiện lấy thông tin cho số lượng khách hàng mới của chi nhánh dựa trên mã chi nhánh(storeId) đã được xây dựng bên BackEnd. Trong khi đó, hàm excelPrintNewCustomerStatistic sẽ in cáo thông tin cho báo cáo dựa trên mã chi nhánh thành dạng file Excel thông qua API đã được xây dựng trước đó.



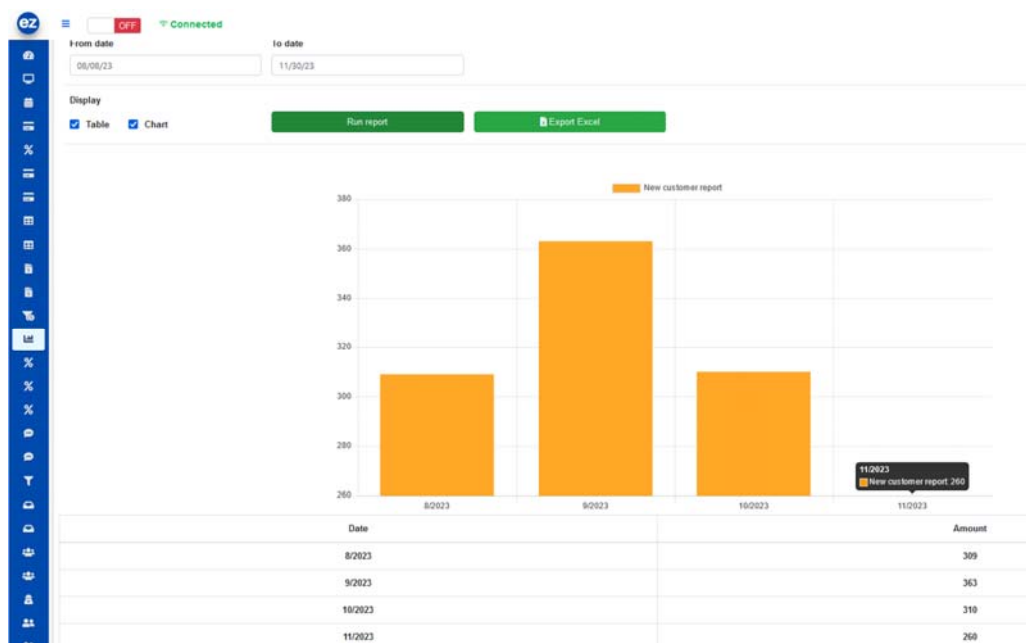
```

64 export class StatisticsComponent {
65   routes: IFakeRoute[] = ChildRoutes;
66   @ViewChild('graph') appGraph: AppNewCustomerGraphStatistic;
67   @ViewChild('detailNewCustomer') detailNewCustomer: AppNewCustomerDetailStatistic;
68   public newCustomerSearch: NewCustomerGraphStatisticSearchModel = new NewCustomerGraphStatisticSearchModel();
69   constructor(
70     private alert: AppAlert,
71     private loading: AppLoading,
72     public currentUserService: CurrentUserService,
73     private common: AppCommon,
74     private reportCusService: ReportCusService,
75     private reportTicketService: ReportStoreService,
76     private storeStateService: StoreStateService,
77   ) {}
78   private getNewCustomerStatisticCompleted(res: ResponseModel<NewCustomerGraphStatisticSearchModel>) {
79     if (res.status !== HTTP_CODE_CONSTANT.OK) {
80       res.message.forEach(value => {
81         this.alert.error(value);
82       });
83       return;
84     }
85     this.newCustomerSearch = res.result;
86     this.appGraph.detailData = this.newCustomerSearch.detail;
87     this.appGraph.updateUI();
88     this.detailNewCustomer.detailData = this.newCustomerSearch.detail;
89     this.detailNewCustomer.totalRecords = this.newCustomerSearch.detail.length.toString();
90     this.loading.hide();
91   }
92   private downloadNewCustomerStatisticCompleted(res: any) {
93     this.loading.hide();
94     const name = 'New customer report';
95     const fileName = `${name}_${new Date().getTime()}.xlsx`;
96     this.reportCusService.saveFile(res, fileName);
97   }
98 }

```

Hình 2.21. Lớp StatisticsComponent xây dựng FontEnd cho chức năng xem Số lượng khách hàng mới của chi nhánh

Lớp StatisticsComponent trong hình 2.21 cũng là lớp trong hình 2.6. Nhưng lúc này StatisticsComponent có thêm các biến Graph và detailNewcustomer với tác dụng lần lượt là hiển thị báo cáo dưới dạng độ thị cột và dạng bảng chi tiết. Hàm getNewCustomerStatisticCompleted để lấy dữ liệu thông qua hàm đã xây dựng trong hình 14 là getNewCustomerStatistic. Trong khi đó hàm xuất thống kê ra dạng excel là downloadNewCustomerStatisticCompleted.



Hình 2.22. chức năng xem số lượng khách hàng mới của chi nhánh sau khi hoàn thiện

Hình 2.22, cho thấy chức năng xem số lượng khách hàng mới của chi nhánh khi bấm vào “run report” sau khi hoàn thiện. Chức năng này sẽ cho người dùng xem thống kê dưới dạng bảng hoặc biểu đồ hoặc cả 2 tùy thuộc vào người dùng chọn trong “Display”. Và xuất Excel tùy ý khi bấm “Export Excel”.

### 2.2.3 Tạo chức năng xem số lượng khách hàng không quay lại của chi nhánh

Theo mô tả từ phản hồi của khách hàng, họ cần có 1 chức năng để cung cấp thêm thông tin về số lượng khách hàng không quay lại cho các chi nhánh có trong hệ thống EZSalon. Chức năng nằm trong module Management sẽ cho biết đầy đủ thông tin về số lượng khách hàng không quay lại theo ngày, tháng, năm dưới dạng bảng. Ngoài ra còn có thể xuất dạng bảng thành file excel để phục vụ cho người dùng sử dụng.

**Statistics**

Report  
No Return Customer Report

Quickly options  
Today Week Monthly Yearly

From date 07/08/24 To date 07/08/24

Run report Export Excel

Hình 2.23. Mẫu thống kê số lượng khách hàng không quay lại chi nhánh

**Thời gian thực hiện:** Tuần 5(Từ ngày 29/07/2024 đến ngày 02/08/2024)

**Lập Trình Backend bằng Springboot:** Đầu tiên để xây dựng luồng xử lý hoạt động cho chức năng xem thống kê số lượng khách hàng không quay lại chi nhánh phía backend của phần mềm. Ta cần xây dựng theo kiến trúc mà công ty đã xây dựng sẵn theo thứ tự xây dựng: Dto, Controller(API), Service, Repository hoặc DAL để truy xuất vào cơ sở dữ liệu như trong chức năng xem doanh thu trung bình của chi nhánh đã được xây dựng trước đó. Vì phần thao tác đến cơ sở dữ liệu là phần tài sản của công ty cho nên phần đó sẽ không được mô tả trong báo cáo.

```

1 package com.antdigital.bellepalace.customer.api.dtos.request;
2
3 import ...
4
12
13 @Data
14 public class NoReturnCustomerSearchDto extends BaseSearchDto<List<CustomerDto>> {
15     @JsonDeserialize(using = LocalDateTimeDeserializer.class)
16     @JsonFormat(shape = JsonFormat.Shape.STRING)
17     @NotNull(message = "Start date is empty")
18     private LocalDateTime fromDate;
19     @JsonDeserialize(using = LocalDateTimeDeserializer.class)
20     @JsonFormat(shape = JsonFormat.Shape.STRING)
21     @NotNull(message = "End date is empty")
22     private LocalDateTime toDate;
23 }

```

Hình 2.24. Các lớp Dto sử dụng trong chức năng xem số lượng khách hàng không quay lại chi nhánh

Đối với chức năng hiển thị thống kê số lượng khách hàng không quay lại chi nhánh. Do không yêu cầu về hiển thị dạng biểu đồ nên có thể tái sử dụng lớp Dto CustomerDto đã được xây dựng sẵn để lấy chi tiết các khách hàng không quay lại chi nhánh. Để lọc ra được những khách hàng không quay lại trong khoảng thời gian được chọn thì việc xây dựng lớp, đối tượng mới để lấy dữ liệu NoReturnCustomerSearchDto là điều bắt buộc phải thực hiện.

```

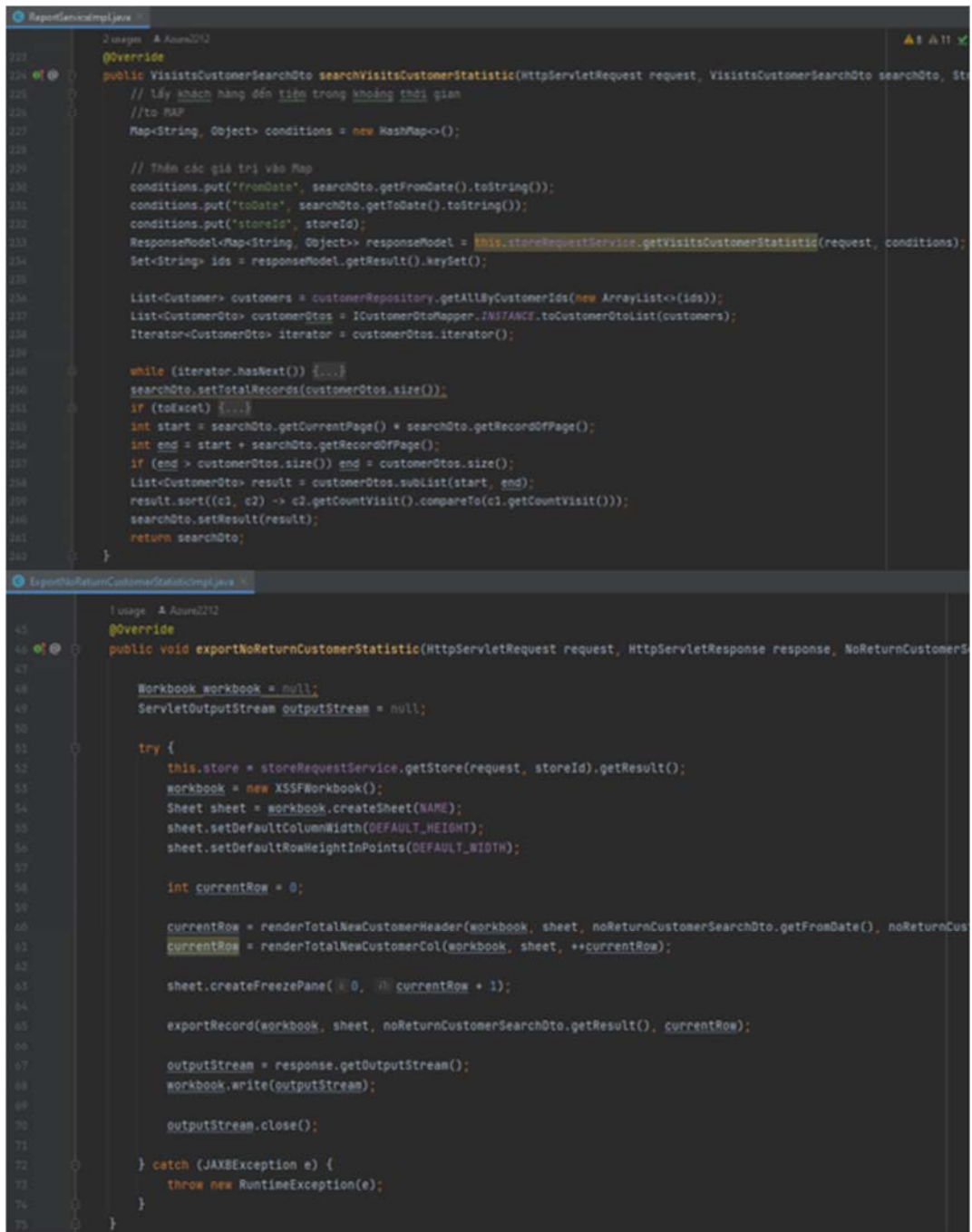
135 @PostMapping("/{searchNoReturnCustomerStatistic}")
136 public ResponseEntity<?> searchNoReturnCustomerStatistic(HttpServletRequest request, @RequestBody NoReturnCustomerSearchDto
137     @RequestParam(required = false) String storeId) throws JAXBException, IOException {
138
139
140
141
142
143
144
145
146 @PostMapping("/{exportNoReturnCustomerStatistic}")
147 public void exportNoReturnCustomerStatistic(HttpServletRequest request, HttpServletResponse response,
148     @RequestBody NoReturnCustomerSearchDto noReturnCustomerSearchDto,
149     @RequestParam(required = false) String storeId) throws IOException {
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165

```

Hình 2.25. Các Hàm API để lấy dữ liệu cho chức năng xem số lượng khách hàng không quay lại chi nhánh

Dựa trên hình 2.25, ta thấy sẽ có 2 controller được xây dựng. Hàm dùng để lấy xem thống kê số lượng khách không quay lại chi nhánh là searchNoReturnCustomerStatistic bên phía BE sẽ thu thập dữ liệu và gửi sang phía FE. Trong khi đó hàm exportNoReturnCustomerStatistic sẽ là API thực hiện chức năng xuất ra excel từ thống kê đó.

Sau khi xây dựng hoàn tất các lớp Dto và xác định được các controller (API) sẽ có trong chức năng xem thống kê số lượng khách không quay lại chi nhánh. Phần tiếp sau đó chính là xây dựng các luồng xử lý cho chức năng tương ứng. Hàm để lấy thống kê số lượng khách hàng không quay lại của chi nhánh theo từng ngày, thời gian người dùng chọn bên FE gửi qua là hàm `searchNoReturnCustomerStatistic` trong lớp `ReportServiceImpl`. Hàm này chỉ đơn giản là lấy hết tất cả khách hàng không có hóa đơn tại chi nhánh trong thời gian FE gửi qua sau đó, xác định hiện báo cáo theo đơn vị này (ngày, giờ hay năm) sau đó gom các khách hàng theo từng cụm đơn vị thời gian rồi tính tổng. Cuối cùng là trả về danh sách các ngày chứa tổng khách hàng mới của ngày tương ứng về `searchNoReturnCustomerStatistic` bên controller và controller trả về FE. Hàm xuất báo cáo ra file excel là `exportNoReturnCustomerStatistic` trong lớp `ExportNoReturnCustomerStatisticServiceImpl`. Tương tự như hàm `exportNoReturnCustomerStatistic` đã trình bày trước đó, `exportNewCustomerStatistic` sẽ lấy các thông tin về số lượng khách hàng không quay lại chi nhánh theo từng ngày tương ứng nhưng khác ở chỗ `exportNoReturnCustomerStatistic` sẽ không trả về Controller mà trực tiếp xuất file cho người dùng.



Hình 2.26. Các Hàm thực hiện nghiệp vụ được xây dựng cho chức năng xem số lượng khách hàng không quay lại của chi nhánh

**Lập Trình Fontend bằng Angular:** Ở phần lập trình Backend bằng Spring boot cho chức năng xem số lượng khách hàng không quay lại của chi nhánh đã hoàn thành việc xây dựng các API cho chức năng. Để lấy được thông tin cần cho chức năng ấy thông qua các API và lớp Controller đã thực hiện, thì phía Fontend bằng Angular sẽ phải xây dựng các hàm để truy xuất đến các API đó được thể hiện hình 2.27.

```

report.service.ts
2 usages  Azure2212
61 public searchNoReturnCustomerStatistic(search: BaseSearchModel<CustomerModel[]>, storeId?: string): Observable<any> {
62     return this.post<url: '/api/v1/report/searchNoReturnCustomerStatistic', search, params: {storeId}>;
63 }
64
1 usage  Azure2212
65 public excelPrintNoReturnCustomerStatistic(search: BaseSearchModel<CustomerModel[]>, storeId?: string): Observable<any> {
66     return this.downloadFile<url: '/api/v1/report/exportNoReturnCustomerStatistic', search, params: {storeId}>;
67 }

```

*Hình 2.27. Các Hàm gọi đến API để lấy dữ liệu cho chức năng xem số lượng khác hàng không quay lại của chi nhánh*

Trong hình 2.27 cho thấy, hàm searchNoReturnCustomerStatistic sẽ thực hiện lấy thông tin cho số lượng khác hàng không quay lại của chi nhánh dựa trên mã chi nhánh(storeId) đã được xây dựng bên Backend. Trong khi đó, hàm excelPrintNoReturnCustomerStatistic sẽ in cáo thông tin cho báo cáo dựa trên mã chi nhánh thành dạng file Excel thông qua API đã được xây dựng trước đó.

```

63  })
64  export class StatisticsComponent {
65      routes: IFakeRoute[] = ChildRoutes;
66      @ViewChild('detailNoReturnCustomer') detailNoReturnCustomer: AppNoReturnCustomerDetailStatistic;
67      public noReturnCustomerSearch: SearchNoReturnCustomerModel = new SearchNoReturnCustomerModel();
68      constructor(
69          private alert: AppAlert,
70          private loading: AppLoading,
71          public currentUserService: CurrentUserService,
72          private common: AppCommon,
73          private reportCusService: ReportCusService,
74          private reportTicketService: ReportStoreService,
75          private storeStateService: StoreStateService,
76      ) {
77          this.reportType = this.reportTypesNames[0].id;
78          const nowDate : string = this.common.nowTzDate();
79          this.fromDate = nowDate;
80          this.toDate = nowDate;
81      }
82      private getNoReturnCustomerCompleted(res: ResponseModel<SearchNoReturnCustomerModel>) :void {
83          if (res.status !== HTTP_CODE_CONSTANT.OK) {
84              res.message.forEach(value : string => {
85                  this.alert.error(value);
86              });
87              return;
88          }
89          this.noReturnCustomerSearch = res.result;
90          this.detailNoReturnCustomer.search = this.noReturnCustomerSearch;
91          this.detailNoReturnCustomer.search.currentPage = 0
92          this.loading.hide();
93      }
94      private downloadNoReturnCustomerStatisticCompleted(res: any) :void {
95          this.loading.hide();
96          const name : "No return customer report" = 'No return customer report';
97          const fileName : string = `${name}_${new Date().getTime()}.xlsx`;
98          this.reportTicketService.saveFile(res, fileName);
99      }

```

Hình 2.28. Lớp StatisticsComponent xây dựng FontEnd cho chức năng xem Số lượng không quay lại của chi nhánh

Lớp StatisticsComponent trong hình 2.28 cũng là lớp trong hình 2.14. Nhưng lúc này StatisticsComponent có thêm biến detailNoReturncustomer với tác dụng là hiển thị báo cáo dưới dạng bảng một cách thật chi tiết và đầy đủ. Hàm getNoReturnCustomerCompleted để lấy dữ liệu thông qua hàm đã xây dựng trong hình 2.27 là getNoReturnCustomerStatistic. Trong khi đó hàm xuất thống kê ra dạng excel là downloadNoReturnCustomerStatisticCompleted.



The screenshot shows the EZSalon Statistics interface. At the top, there's a status bar with 'EZ' logo, 'Connected' status, and 'Belle Palace Hair Spa Network'. The main heading is 'Statistics'. Below it, a 'Report' dropdown is set to 'No Return Customer Report'. There are 'Quickly options' buttons for 'Today', 'Week', 'Month', and 'Yearly'. Date pickers for 'From date' (05/05/23) and 'To date' (12/31/23) are present. Two green buttons, 'Run report' and 'Export Excel', are at the bottom of the filters. Below the buttons, a table displays customer data with columns: First name, Last name, Phone, Visits, and Last visit. The table shows 10 entries. At the bottom right, there's a pagination bar with 'First', 'Previous', '1', '2', '3', '4', '5', 'Next', and 'Last' buttons. A watermark 'Activate Windows' is visible in the bottom right corner.

First name	Last name	Phone	Visits	Last visit
Vickie	Adis	(302) 983-2919	6	2023-11-16T07:57:31
Sydney	Daniela	(951) 234-2548	2	2023-11-17T17:53:44
Beyonce	Harg	(757) 325-0904	2	2023-09-29T17:53:32
Ashley		(919) 358-9595	2	2023-11-14T14:19:46
Daria	Fields	(302) 299-9871	2	2023-10-25T11:10:24
Chelvia	Baldwin	(302) 985-2233	2	2023-11-19T12:18:54
Autumn	Harmon	(302) 241-1911	1	2023-09-26T12:52:25
Valerie	Jones	(404) 805-4141	1	2023-10-11T13:32:20
Anika	Walker	(302) 826-2650	1	2023-09-03T12:54:11
Jordyn	Venick	(443) 801-6566	1	2023-10-08T17:00:32

Hình 2.29. chức năng xem số lượng khách hàng không quay lại chi nhánh sau khi hoàn thiện

Hình 2.29, cho thấy chức năng xem số lượng khách hàng không quay lại chi nhánh khi bấm vào “run report” sau khi hoàn thiện. Chức năng này sẽ cho người dùng xem thống kê dưới dạng bảng do số lượng khách hàng có thể rất nhiều nên sẽ có thêm các nút phân trang bên dưới. Và xuất Excel tùy ý khi bấm “Export Excel”.

## 2.2.4 Tạo chức năng xem số lượng khách hàng sử dụng dịch vụ của chi nhánh

Theo mô tả từ phản hồi của khách hàng, họ cần có 1 chức năng để cung cấp thêm thông tin về số lượng khách hàng sử dụng dịch vụ cho các chi nhánh có trong hệ thống EZSalon. Chức năng nằm trong module Management sẽ cho biết đầy đủ thông tin về số lượng khách hàng sử dụng dịch vụ tại chi nhánh theo ngày, tháng, năm dưới dạng bảng. Ngoài ra còn có thể xuất dạng bảng thành file excel để phục vụ cho người dùng sử dụng.

The screenshot shows a web application interface for generating statistics. On the left is a blue sidebar with icons for different functions. The top of the page has a status bar with an 'ez' logo, a menu icon, a red 'OFF' button, and a green 'Connected' status with a Wi-Fi icon. The main section is titled 'Statistics'. Below the title, there's a 'Report' dropdown menu currently showing 'Visits Customer Report'. Underneath, there's a 'Quickly options' section with four green buttons: 'Today', 'Week', 'Monthly', and 'Yearly'. Below these are two date input fields labeled 'From date' and 'To date', both containing the date '07/08/24'. At the bottom of this section are two large green buttons: 'Run report' and 'Export Excel'.

Hình 2.30. Mẫu thống kê số lượng khách hàng sử dụng dịch vụ tại chi nhánh

**Thời gian thực hiện:** Tuần 6(Từ ngày 29/07/2024 đến ngày 02/08/2024)

**Lập Trình Backend bằng Springboot:** Cũng giống như các chức năng trước để xây dựng luồng xử lý hoạt động cho chức năng xem thống kê số lượng khách hàng sử dụng dịch vụ tại chi nhánh phía backend của phần mềm. Ta cần xây dựng theo kiến trúc mà công ty đã xây dựng sẵn theo thứ tự xây dựng: Dto, Controller(API), Service, Repository hoặc DAL để truy xuất vào cơ sở dữ liệu như trong chức năng xem doanh thu trung bình của chi nhánh đã được xây dựng trước đó. Vì phần thao tác đến cơ sở dữ liệu là phần tài sản của công ty cho nên phần đó sẽ không được mô tả trong báo cáo.

```

1 package com.antdigital.bellepalace.customer.api.dtos.request;
2
3 import ...
4
5 12 usages  ▲ Azure2212
6
7 @Data
8 public class VisistsCustomerSearchDto extends BaseSearchDto<List<CustomerDto>> {
9     @JsonDeserialize(using = LocalDateTimeDeserializer.class)
10    @JsonFormat(shape = JsonFormat.Shape.STRING)
11    @NotNull(message = "Start date is empty")
12    private LocalDateTime fromDate;
13
14    @JsonDeserialize(using = LocalDateTimeDeserializer.class)
15    @JsonFormat(shape = JsonFormat.Shape.STRING)
16    @NotNull(message = "End date is empty")
17    private LocalDateTime toDate;
18 }

```

Hình 2.31. Các lớp Dto sử dụng trong chức năng xem số lượng khách hàng sử dụng dịch vụ tại chi nhánh

Đối với chức năng hiển thị thống kê số lượng khách hàng sử dụng dịch vụ tại chi nhánh. Do không yêu cầu về hiển thị dạng biểu đồ nên có thể tái sử dụng lớp Dto CustomerDto đã được xây dựng sẵn để lấy chi tiết các khách hàng sử dụng dịch vụ tại chi nhánh. Để lọc ra được những khách hàng sử dụng dịch vụ tại chi nhánh trong khoảng thời gian được chọn thì việc xây dựng lớp, đối tượng mới để lấy dữ liệu VisistsCustomerSearchDto là điều bắt buộc phải thực hiện.

```

186
187 ▲ Azure2212
188 @PostMapping("/searchVisitsCustomerStatistic")
189 public ResponseEntity<> searchVisitsCustomerStatistic(HttpServletRequest request,
190     @RequestBody VisistsCustomerSearchDto visistsCustomerSearchDto,
191     @RequestParam(required = false) String storeId) throws Exception {
192 }
193
194 ▲ Azure2212
195 @PostMapping("/exportVisitsCustomerStatistic")
196 public void exportVisitsCustomerStatistic(HttpServletRequest request, HttpServletResponse response,
197     @RequestBody VisistsCustomerSearchDto visistsCustomerSearchDto,
198     @RequestParam(required = false) String storeId) throws Exception {
199 }
200
201
202
203
204
205

```

Hình 2.32. Các Hàm API để lấy dữ liệu cho chức năng xem số lượng khách hàng sử dụng dịch vụ tại chi nhánh

Dựa trên hình 2.32, ta thấy sẽ có 2 controller được xây dựng. Hàm dùng để lấy xem thống kê số lượng khách hàng sử dụng dịch vụ tại chi nhánh là searchVisistsCustomerStatistic bên phía BE sẽ thu thập dữ liệu và gửi sang phía FE.

Trong khi đó hàm `exportVisitsCustomerStatistic` sẽ là API thực hiện chức năng xuất ra excel từ thống kê đó.

Sau khi xây dựng hoàn tất các lớp Dto và xác định được các controller (API) sẽ có trong chức năng xem thống kê số lượng khách hàng sử dụng dịch vụ tại chi nhánh. Phần tiếp sau đó chính là xây dựng các luồng xử lý cho chức năng tương ứng. Hàm để lấy thống kê số lượng khách hàng sử dụng dịch vụ tại chi nhánh theo từng ngày, thời gian người dùng chọn bên FE gửi qua là hàm `searchVisitsCustomerStatistic` trong lớp service `ReportServiceImpl`. Hàm này chỉ đơn giản là lấy hết tất cả khách hàng có hóa đơn tại chi nhánh trong thời gian FE gửi qua sau đó, xác định hiện báo cáo theo đơn vị này(ngày, giờ hay năm) sau đó gom các khách hàng theo từng cụm đơn vị thời gian rồi tính tổng. Cuối cùng là trả về danh sách các ngày chứa tổng khách hàng sử dụng dịch vụ của ngày tương ứng về `searchVisitsCustomerStatistic` bên controller và controller trả về FE. Hàm xuất báo cáo ra file excel là `exportVisitsCustomerStatistic` trong lớp `ExportVisistCustomerStatisticServiceImpl`. Tương tự như hàm `exportNoReturnCustomerStatistic` đã trình bày trước đó, `exportVisistCustomerStatistic` sẽ lấy các thông tin về số lượng khách hàng mới theo từng ngày tương ứng nhưng khác ở chỗ `exportVisistCustomerStatistic` sẽ không trả về Controller mà trực tiếp xuất file cho người dùng.

```

ReportServiceImpl.java
2 usages  ▲ Azure2212
223 @Override
224 public VisistsCustomerSearchDto searchVisitsCustomerStatistic(HttpServletRequest request, VisistsCustomerSearchDto se
225 // lấy khách hàng đến tiệm trong khoảng thời gian
226 //to MAP
227 Map<String, Object> conditions = new HashMap<>();
228
229 // Thêm các giá trị vào Map
230 conditions.put("fromDate", searchDto.getFromDate().toString());
231 conditions.put("toDate", searchDto.getToDate().toString());
232 conditions.put("storeId", storeId);
233 ResponseModel<Map<String, Object>> responseModel = this.storeRequestService.getVisitsCustomerStatistic(request, c
234 Set<String> ids = responseModel.getResult().keySet();
235
236 List<Customer> customers = customerRepository.getAllByCustomerIds(new ArrayList<>(ids));
237 List<CustomerDto> customerDtos = ICustomerDtoMapper.INSTANCE.toCustomerDtoList(customers);
238 Iterator<CustomerDto> iterator = customerDtos.iterator();
239
240 while (iterator.hasNext()) {...}
241 searchDto.setTotalRecords(customerDtos.size());
242 if (toExcel) {...}
243 int start = searchDto.getCurrentPage() * searchDto.getRecordOfPage();
244 int end = start + searchDto.getRecordOfPage();
245 if (end > customerDtos.size()) end = customerDtos.size();
246 List<CustomerDto> result = customerDtos.subList(start, end);
247 result.sort((c1, c2) -> c2.getCountVisit().compareTo(c1.getCountVisit()));
248 searchDto.setResult(result);
249 return searchDto;
250 }

ExportVisitsCustomerStatisticImpl.java
1 usage  ▲ Azure2212
42 @Override
43 public void exportVisitsCustomerStatistic(HttpServletRequest request, HttpServletResponse response, VisistsCustome
44
45 Workbook workbook = null;
46 ServletOutputStream outputStream = null;
47
48 try {...} catch (JAXBException e) {
49     throw new RuntimeException(e);
50 }
51
52 }

```

Hình 2.33. Các Hàm thực hiện nghiệp vụ được xây dựng cho chức năng xem số lượng khách hàng không quay lại của chi nhánh

**Lập Trình Fontend bằng Angular:** Ở phần lập trình Backend bằng Spring boot cho chức năng xem số lượng khách hàng sử dụng dịch vụ của chi nhánh đã hoàn thành việc xây dựng các API cho chức năng. Để lấy được thông tin cần cho chức năng ấy thông qua các API và lớp Controller đã thực hiện, thì phía Fontend bằng Angular sẽ phải xây dựng các hàm để truy xuất đến các API đó được thể hiện hình 2.34.

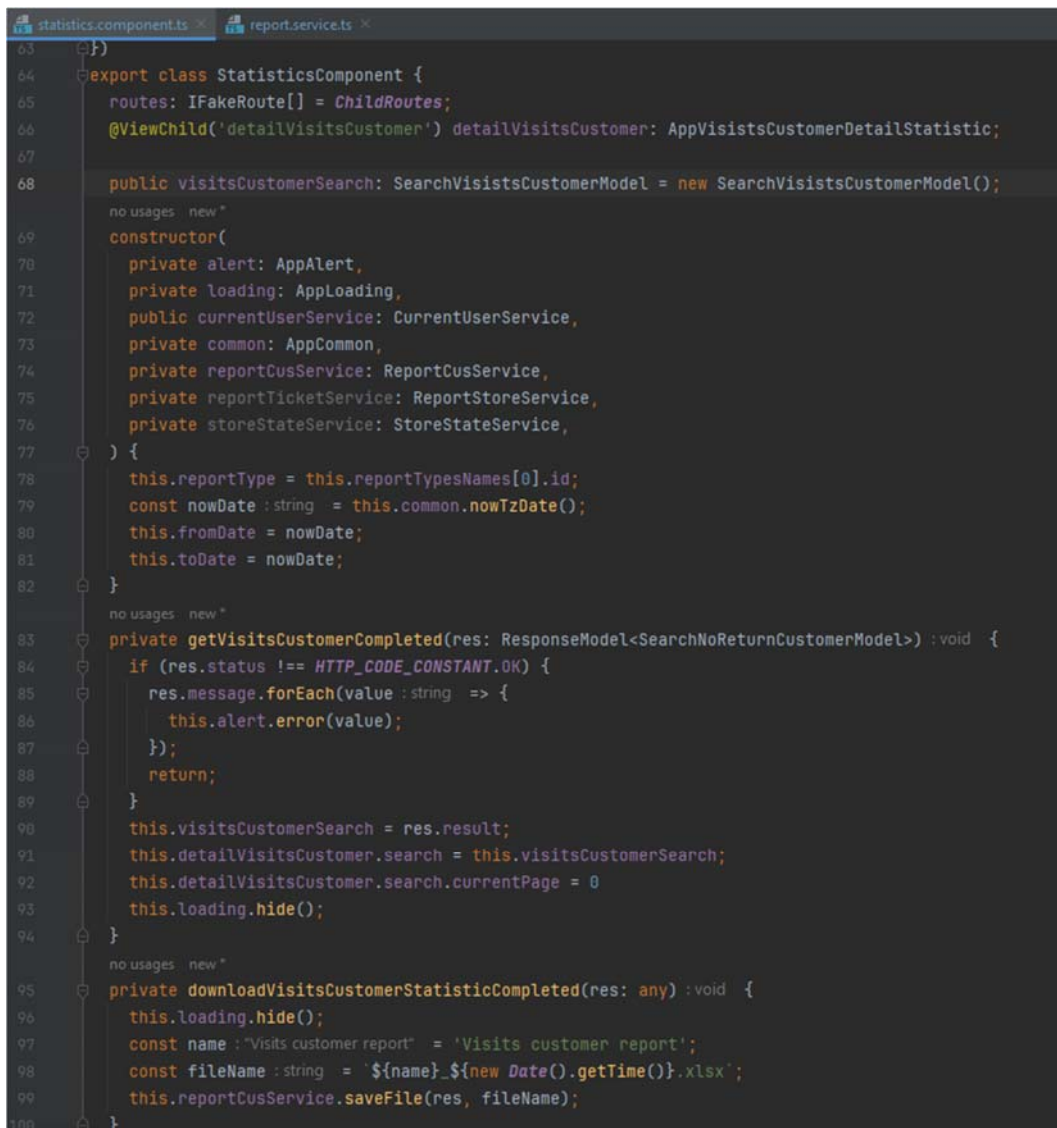
```

report.service.ts
1 usage  ▲ Azure2212
68 public searchVisitsCustomerStatistic(search: BaseSearchModel<CustomerModel[]>, storeId?: string): Observable<any> {
69     return this.post(Url: '/api/v1/report/searchVisitsCustomerStatistic', search, params {storeId});
70 }
71
72 public excelPrintVisitsCustomerStatistic(search: BaseSearchModel<CustomerModel[]>, storeId?: string): Observable<any> {
73     return this.downloadFile(Url: '/api/v1/report/exportVisitsCustomerStatistic', search, params {storeId});
74 }
75 }

```

Hình 2.34. Các Hàm gọi đến API để lấy dữ liệu cho chức năng xem số lượng khách hàng sử dụng dịch vụ tại chi nhánh

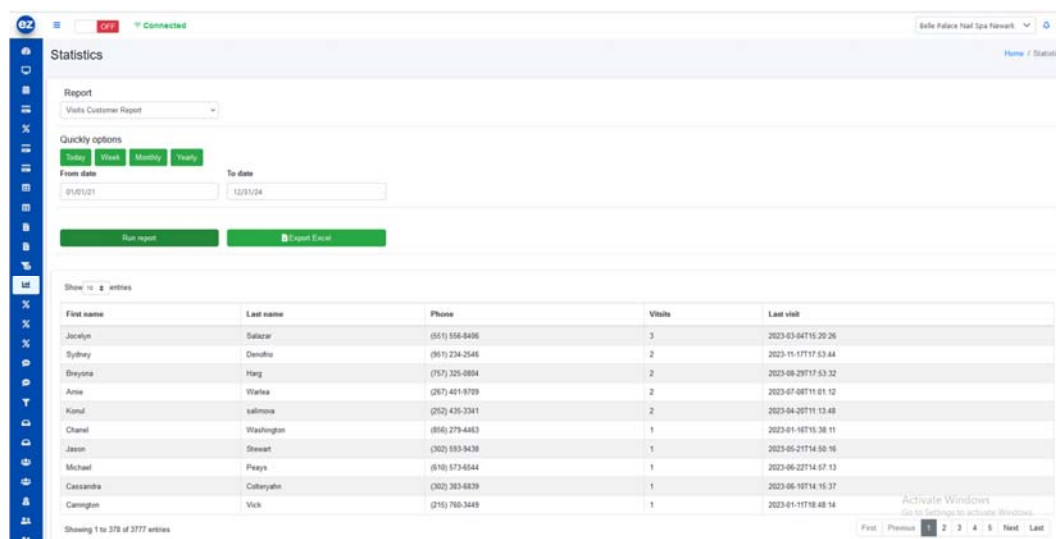
Trong hình 2.34 cho thấy, hàm searchVisitsCustomerStatistic sẽ thực hiện lấy thông tin cho số lượng khách hàng sử dụng dịch vụ tại chi nhánh dựa trên mã chi nhánh(storeId) đã được xây dựng bên Backend. Trong khi đó, hàm excelPrintVisitsCustomerStatistic sẽ in cáo thông tin cho báo cáo dựa trên mã chi nhánh thành dạng file Excel thông qua API đã được xây dựng trước đó.



```
63 }
64 export class StatisticsComponent {
65   routes: IFakeRoute[] = ChildRoutes;
66   @ViewChild('detailVisitsCustomer') detailVisitsCustomer: AppVisitsCustomerDetailStatistic;
67
68   public visitsCustomerSearch: SearchVisitsCustomerModel = new SearchVisitsCustomerModel();
69   constructor(
70     private alert: AppAlert,
71     private loading: AppLoading,
72     public currentUserService: CurrentUserService,
73     private common: AppCommon,
74     private reportCusService: ReportCusService,
75     private reportTicketService: ReportStoreService,
76     private storeStateService: StoreStateService,
77   ) {
78     this.reportType = this.reportTypesNames[0].id;
79     const nowDate : string = this.common.nowTzDate();
80     this.fromDate = nowDate;
81     this.toDate = nowDate;
82   }
83   private getVisitsCustomerCompleted(res: ResponseModel<SearchNoReturnCustomerModel>) :void {
84     if (res.status !== HTTP_CODE_CONSTANT.OK) {
85       res.message.forEach(value : string => {
86         this.alert.error(value);
87       });
88       return;
89     }
90     this.visitsCustomerSearch = res.result;
91     this.detailVisitsCustomer.search = this.visitsCustomerSearch;
92     this.detailVisitsCustomer.search.currentPage = 0
93     this.loading.hide();
94   }
95   private downloadVisitsCustomerStatisticCompleted(res: any) :void {
96     this.loading.hide();
97     const name : "Visits customer report" = 'Visits customer report';
98     const fileName : string = `${name}_${new Date().getTime()}.xlsx`;
99     this.reportCusService.saveFile(res, fileName);
100   }
```

Hình 2.35. Lớp StatisticsComponent xây dựng FontEnd cho chức năng xem Số lượng khách sử dụng dịch vụ tại chi nhánh

Lớp StatisticsComponent trong hình 2.35 cũng là lớp trong hình 2.14. Nhưng lúc này StatisticsComponent có thêm biến detailVisitscustomer với tác dụng là hiển thị báo cáo dưới dạng bảng một cách thật chi tiết và đầy đủ. Hàm getVisitsCustomerCompleted để lấy dữ liệu thông qua hàm đã xây dựng trong hình 2.27 là searchVisitsCustomerStatistic. Trong khi đó hàm xuất thống kê ra dạng excel là downloadVisitsCustomerStatisticCompleted.



The screenshot shows a web application interface for 'EZSalon'. The 'Statistics' section is active, displaying a 'Visits Customer Report'. Below the report title, there are 'Quickly options' for 'Today', '7 Days', 'Monthly', and 'Yearly'. The 'From date' is set to '01/01/21' and the 'To date' is '12/31/24'. There are buttons for 'Run report' and 'Export Excel'. Below these, a table shows customer visit data. The table has columns for 'First name', 'Last name', 'Phone', 'Visits', and 'Last visit'. The data is as follows:

First name	Last name	Phone	Visits	Last visit
Jacelyn	Salazar	(861) 556-8496	3	2023-03-04T16:39:26
Sydney	Denzho	(961) 234-2546	2	2023-11-17T17:53:44
Bryana	Harg	(767) 325-0804	2	2023-08-29T17:53:32
Anne	Worke	(267) 491-9789	2	2023-07-08T11:01:12
Karel	Salmona	(262) 435-2341	2	2023-04-20T11:13:48
Chanel	Washington	(866) 279-4483	1	2023-01-18T16:38:11
Jason	Stewart	(302) 193-3438	1	2023-05-21T14:58:16
Michael	Peays	(610) 573-6544	1	2023-06-22T14:57:13
Cassandra	Colonyah	(302) 383-4839	1	2023-06-18T14:16:37
Carrington	Vick	(216) 780-3448	1	2023-01-11T18:48:14

At the bottom of the table, it says 'Showing 1 to 378 of 3777 entries'. There are also pagination controls: 'First', 'Previous', '2', '3', '4', '5', 'Next', 'Last'.

Hình 2.36. chức năng xem số lượng khách hàng sử dụng dịch vụ tại chi nhánh sau khi hoàn thiện

Hình 2.36, cho thấy chức năng xem số lượng khách hàng sử dụng dịch vụ tại chi nhánh khi bấm vào “run report” sau khi hoàn thiện. Chức năng này sẽ cho người dùng xem thống kê dưới dạng bảng do số lượng khách hàng có thể rất nhiều nên sẽ có thêm các nút phân trang bên dưới. Và xuất Excel tùy ý khi bấm “Export Excel”.

## 2.2.5 Cập nhật lại chức năng Dashboard

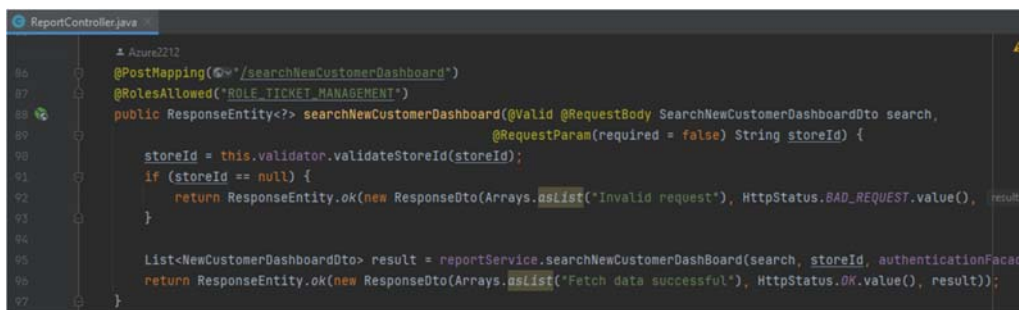
Theo mô tả từ phản hồi của khách hàng, họ cần chức năng Dashboard cung cấp thêm thông tin về số lượng khách hàng mới cho các chi nhánh có trong hệ thống EZSalon. Chức năng sẽ cho biết đầy đủ số lượng khách hàng mới sử dụng dịch vụ tại chi nhánh theo ngày, tháng, năm và tất cả năng dưới dạng số liệu và biểu đồ.

**Thời gian thực hiện:** Tuần 7(Từ ngày 05/08/2024 đến ngày 09/08/2024)

**Lập Trình Backend bằng Springboot:** Theo quy trình lẽ ra ta cần xây dựng theo kiến trúc mà công ty đã xây dựng sẵn theo thứ tự xây dựng: Dto, Controller(API), Service,



Repository hoặc DAL để truy xuất vào cơ sở dữ liệu nhưng vì chức năng này đã có sẵn chỉ cần chỉnh sửa thêm thông tin số lượng khách hàng không quay lại chi nhánh là đủ.

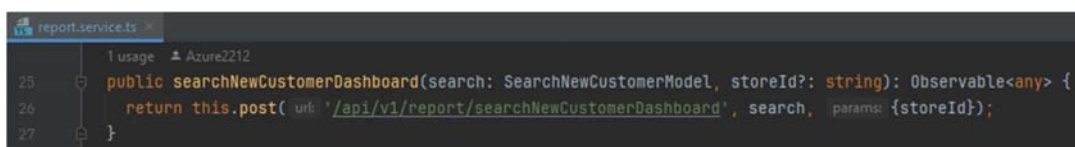


```
ReportController.java
1 Azure2212
24 @PostMapping("/searchNewCustomerDashboard")
25 @RolesAllowed("ROLE_TICKET_MANAGEMENT")
26 public ResponseEntity<?> searchNewCustomerDashboard(@Valid @RequestBody SearchNewCustomerDashboardDto search,
27 @RequestParam(required = false) String storeId) {
28     storeId = this.validator.validateStoreId(storeId);
29     if (storeId == null) {
30         return ResponseEntity.ok(new ResponseDto(Arrays.asList("Invalid request"), HttpStatus.BAD_REQUEST.value(), result));
31     }
32     List<NewCustomerDashboardDto> result = reportService.searchNewCustomerDashBoard(search, storeId, authenticationFacade);
33     return ResponseEntity.ok(new ResponseDto(Arrays.asList("Fetch data successful"), HttpStatus.OK.value(), result));
34 }
```

Hình 2.37. cập nhật thêm API cho biết số lượng khách hàng mới của chi nhánh

Dựa trên hình 2.37, ta thấy sẽ có 1 hàm được xây dựng. Hàm dùng để lấy xem thống kê số lượng khách hàng mới lần đầu sử dụng dịch vụ tại chi nhánh là searchNewCustomerDashboard bên phía BE sẽ thu thập dữ liệu và gửi sang phía FE.

**Lập Trình Fontend bằng Angular:** Ở phần lập trình Backend bằng Spring boot cho chức năng xem số lượng khách hàng mới của chi nhánh đã hoàn thành việc xây dựng các API cho chức năng. Để lấy được thông tin cần cho chức năng ấy thông qua các API và lớp Controller đã thực hiện, thì phía Fontend bằng Angular sẽ phải xây dựng các hàm để truy xuất đến các API đó được thể hiện hình 2.38.



```
report.service.ts
1 usage 1 Azure2212
25 public searchNewCustomerDashboard(search: SearchNewCustomerModel, storeId?: string): Observable<any> {
26     return this.post( url: '/api/v1/report/searchNewCustomerDashboard', search, params: {storeId});
27 }
```

Hình 2.38. Hàm gọi đến API để lấy dữ liệu cho khách hàng mới cho Dashboard

Trong hình 2.38 cho thấy, hàm searchNewCustomerDashboard sẽ thực hiện lấy thông tin cho số lượng khách hàng mới lần đầu sử dụng dịch vụ tại chi nhánh dựa trên mã chi nhánh(storeId) đã được xây dựng bên BackEnd.



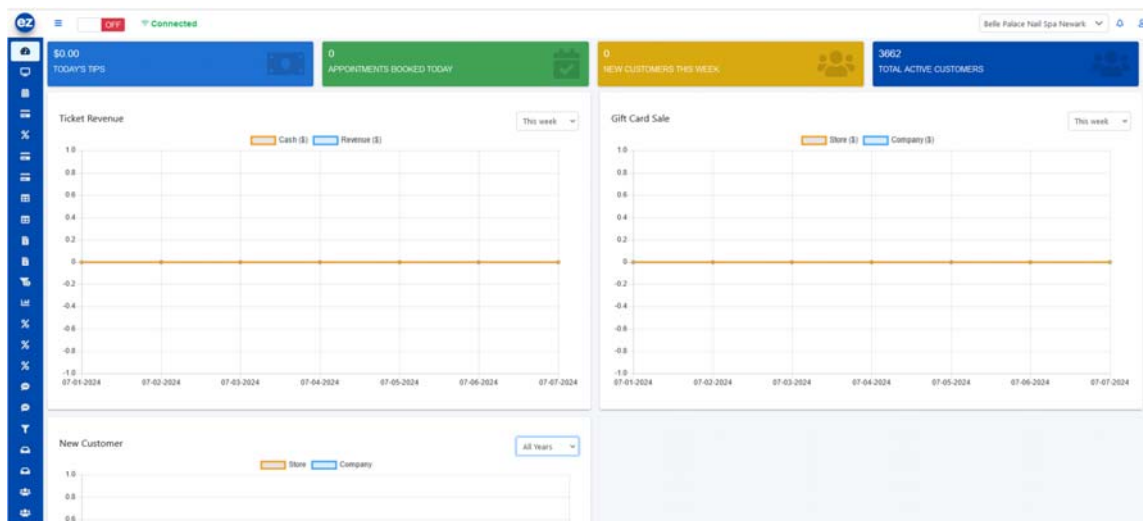
```

176 private searchDataNewCustomer(): void {
177     this.reportCustomerService.searchNewCustomerDashboard(this.searchNewCustomer, this.storeStateService.currentStore?.id);
178     next res => this.searchDataNewCustomerCompleted(res));
179 }
180
181 private searchDataNewCustomerCompleted(res: ResponseModel<NewCustomerDashboardModel[]>): void {
182     this.loading.hide();
183     if (!this.catchErrorMessage(res)) {
184         return;
185     }
186     this.handleNewCustomerData(res.result, this.searchNewCustomer.distinctionType);
187 }
188
189 2 usages
190 private handleNewCustomerData(newCustomers: NewCustomerDashboardModel[], formatType: string): void {
191     const formatTime :string = this.getFormatTimeDistinction(formatType);
192
193     const [labels, storeLists, companyList] = newCustomers
194     .reduce<any>((acc, revenue : NewCustomerDashboardModel ) => {
195         acc[0].push(moment(revenue.date).format(formatTime).toString());
196         acc[1].push(revenue.newCustomerStore);
197         acc[2].push(revenue.newCustomerCompany);
198         return acc;
199     }, [[], [], []]);
200
201     this.newCustomerChartData.labels = labels;
202     this.newCustomerChartData.datasets[0].data = storeLists;
203     this.newCustomerChartData.datasets[1].data = companyList;
204     this.newCustomerChart.refresh();
205 }
206
207 1 usage
208 private getDashboardFromCustomerCompleted(res: ResponseModel<DashboardCustomerModel>): void {
209     this.loading.hide(this.root.nativeElement.querySelector('.new-customer'));
210     this.loading.hide(this.root.nativeElement.querySelector('.active-customer'));
211     this.loading.hide(this.root.nativeElement.querySelector('.gift-card-chart'));
212     this.loading.hide(this.root.nativeElement.querySelector('.new-customer-chart'));
213     if (!this.catchErrorMessage(res)) {
214         return;
215     }
216     this.dashboardFromCustomer = res.result;
217     this.amountNewCustomer = this.dashboardFromCustomer.newCustomerInWeek.reduce((acc : number , current : NewCustomerDashboardModel) => {
218         return acc + current.newCustomerInWeek;
219     }, 0);
220     this.handleGCRevenueData(this.dashboardFromCustomer.revenues, TIME_DISTINCTION.DATE.toString());
221     this.handleNewCustomerData(this.dashboardFromCustomer.newCustomerInWeek, TIME_DISTINCTION.DATE.toString());
222 }

```

Hình 2.39. Bổ sung hàm vào lớp DashboardComponent

Lớp DashboardComponent trong hình 2.39 đã được điều chỉnh thêm thông tin khách hàng mới trong chi nhánh. Hàm searchDataNewCustomerCompleted để lấy dữ liệu thông qua hàm đã xây dựng trong hình 2.38 là searchNewCustomerDashboard.



Hình 2.30. Chức năng xem Dashboard sau khi hoàn thiện

## 2.2.6 Cập nhật lại chức năng xem lại các tin nhắn SMS đã gửi đến khách hàng

Theo mô tả từ phản hồi của khách hàng, họ cần đổi mới chức năng xem lại các sms đã gửi. Chức năng nằm trong module Marketing sẽ cung cấp thêm thông tin chi tiết về các sms đã gửi đến khách hàng bao gồm tất cả tin nhắn đã gửi thành công lẫn thất bại.

**Thời gian thực hiện:** Tuần 8(Từ ngày 12/08/2024 đến ngày 16/08/2024)

**Lập Trình Backend bằng Springboot:** Cũng giống như các chức năng trước để xây dựng luồng xử lý hoạt động cho chức năng xem lại tất cả các sms đã gửi đến cho người dùng phía backend của phần mềm. Ta cần xây dựng theo kiến trúc mà công ty đã xây dựng sẵn theo thứ tự xây dựng: Dto, Controller(API), Service, Repository hoặc DAL để truy xuất vào cơ sở dữ liệu như trong chức năng xem doanh thu trung bình của chi nhánh đã được xây dựng trước đó. Vì phần thao tác đến cơ sở dữ liệu là phần tài sản của công ty cho nên phần đó sẽ không được mô tả trong báo cáo.

```

1 package com.ez.marketing.bpmarketing.dtos.smsSending;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18 @Data
19 @Builder
20 @NoArgsConstructor
21 @AllArgsConstructor
22 public class SmsReportSearchDto extends BaseSearchModel<List<SmsSendingDto>> {
23     private SmsSendingType type;
24     private SmsSendingStatus status;
25     private String phone;
26     @JsonDeserialize(using = LocalDateTimeDeserializer.class)
27     @JsonFormat(shape = JsonFormat.Shape.STRING)
28     @NotNull(message = "Start date is empty")
29     private LocalDateTime fromDate;
30     @JsonDeserialize(using = LocalDateTimeDeserializer.class)
31     @JsonFormat(shape = JsonFormat.Shape.STRING)
32     @NotNull(message = "End date is empty")
33     private LocalDateTime toDate;
34 }

```

Hình 2.31. lớp Dto sử dụng trong chức năng xem lại các tin nhắn đã gửi đến khách hàng

Đối với chức năng xem lại các tin nhắn đã gửi đến khách hàng. Do không yêu cầu về hiển thị dạng biểu đồ nên có thể tái sử dụng lớp Dto CustomerDto đã được xây dựng sẵn để lấy chi tiết các tin nhắn đã gửi đến khách hàng. Điều đó được thể hiện qua lớp SmsReportingSearchDto trong hình 2.31.

```

91
92 @PostMapping("/getAllSMSSending")
93 public Mono<ResponseDto> getAllSMSSending(@RequestBody SmsReportSearchDto searchDto){
94     Mono<SmsReportSearchDto> rs = getSMSSendingBiz.getAllSMSSending(searchDto);
95     return rs.map(data -> ResponseUtils.getSuccessResp(data, "apis.get.success"));
96 }

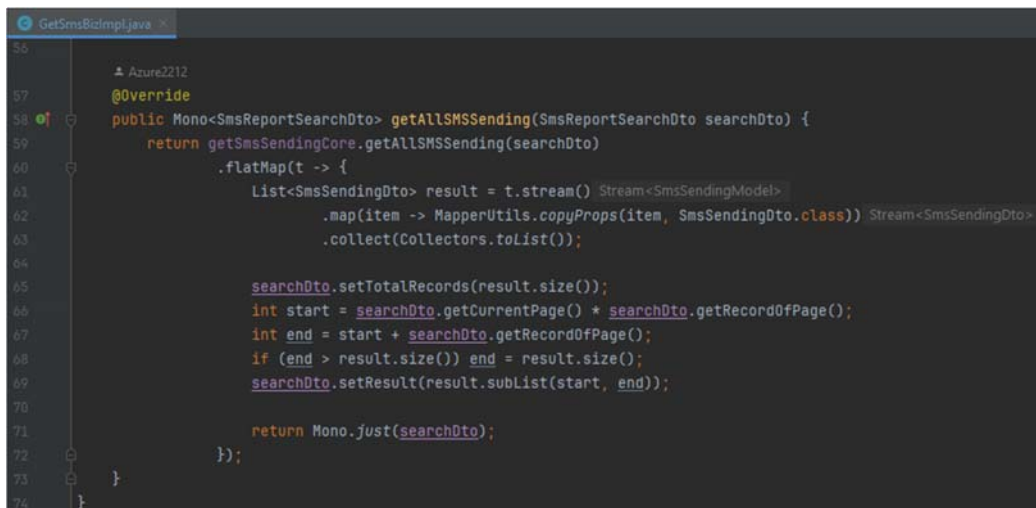
```

Hình 2.32. API để lấy dữ liệu cho chức năng xem lại các tin nhắn đã gửi đến khách hàng

Dựa trên hình 2.32, ta thấy sẽ có 1 controller được xây dựng. Hàm dùng để lấy xem lại các tin nhắn đã gửi đến khách hàng là getAllSMSSending bên phía BE sẽ thu thập dữ liệu và gửi sang phía FE.

Sau khi xây dựng hoàn tất các lớp Dto và xác định được các controller(API) sẽ có trong chức năng xem lại các tin nhắn đã gửi đến khách hàng. Phần tiếp sau đó chính

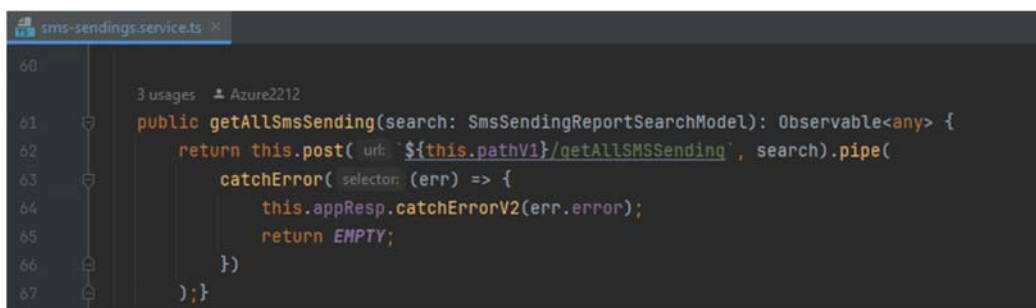
là xây dựng các luồng xử lý cho chức năng tương ứng. Hàm để lấy tất cả các tin nhắn đã gửi đến khách hàng là hàm getAllSMSSending trong lớp service GetSmsBizImpl. Hàm này chỉ đơn giản là lấy hết tất cả các tin nhắn đã được gửi cho khách hàng từ khi có phần mềm đến hiện tại. Cuối cùng là trả về danh sách các ngày chứa tổng khách hàng sử dụng dịch vụ của ngày tương ứng về getAllSMSSending bên controller và controller trả về FE.



```
56
57
58  @Override
59  public Mono<SmsReportSearchDto> getAllSMSSending(SmsReportSearchDto searchDto) {
60      return getSmsSendingCore.getAllSMSSending(searchDto)
61          .flatMap(t -> {
62              List<SmsSendingDto> result = t.stream() Stream<SmsSendingModel>
63                  .map(item -> MapperUtils.copyProps(item, SmsSendingDto.class)) Stream<SmsSendingDto>
64                  .collect(Collectors.toList());
65
66              searchDto.setTotalRecords(result.size());
67              int start = searchDto.getCurrentPage() * searchDto.getRecordOfPage();
68              int end = start + searchDto.getRecordOfPage();
69              if (end > result.size()) end = result.size();
70              searchDto.setResult(result.subList(start, end));
71
72              return Mono.just(searchDto);
73          });
74  }
```

Hình 2.33. Hàm thực hiện nghiệp vụ được xây dựng cho chức năng xem lại các tin nhắn đã gửi đến khách hàng

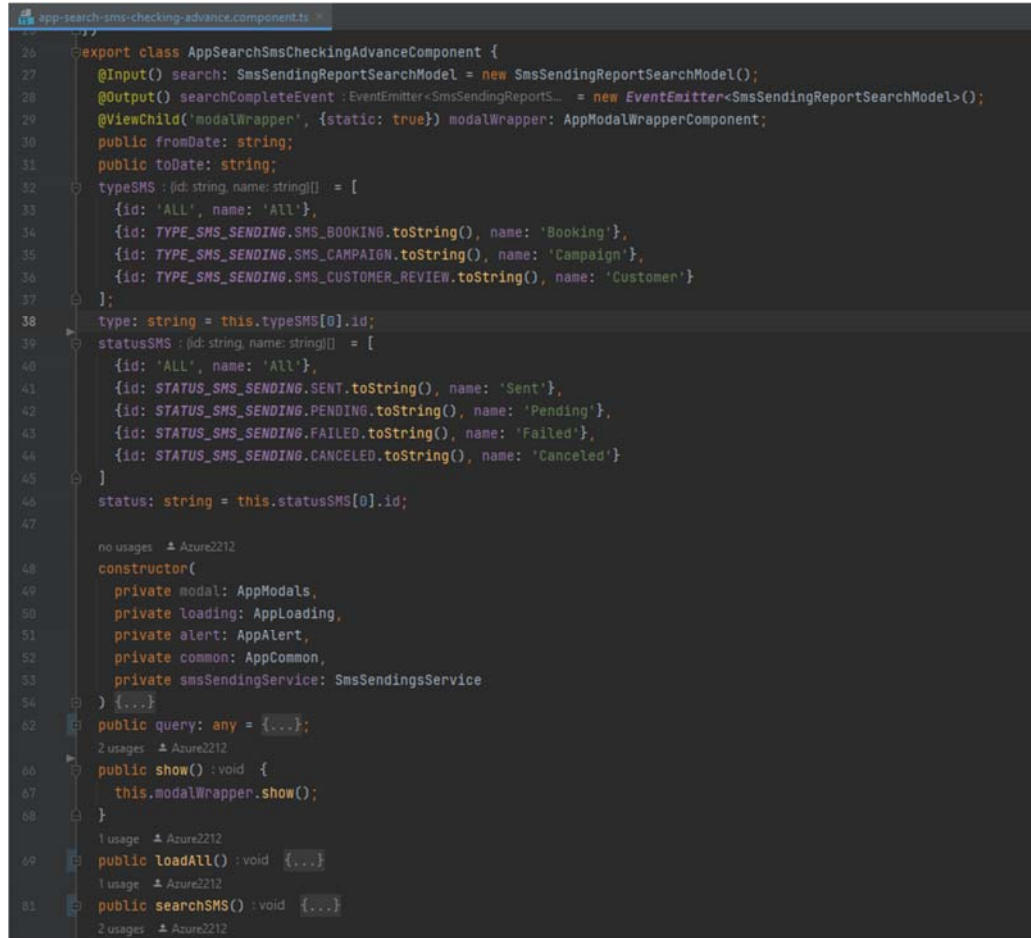
**Lập Trình Fontend bằng Angular:** Ở phần lập trình Backend bằng Spring boot cho chức năng xem lại các tin nhắn đã gửi đến khách hàng đã hoàn thành việc xây dựng các API cho chức năng. Để lấy được thông tin cần cho chức năng ấy thông qua các API và lớp Controller đã thực hiện, thì phía Fontend bằng Angular sẽ phải xây dựng các hàm để truy xuất đến các API đó được thể hiện hình 37.



```
60
61  3 usages  Azure2212
62  public getAllSmsSending(search: SmsSendingReportSearchModel): Observable<any> {
63      return this.post( url: `${this.pathV1}/getAllSMSSending`, search).pipe(
64          catchError( selector: (err) => {
65              this.appResp.catchErrorV2(err.error);
66              return EMPTY;
67          })
68      );
69  }
```

Hình 2.34. Hàm gọi đến API để lấy dữ liệu cho chức năng xem lại các tin nhắn đã gửi đến khách hàng

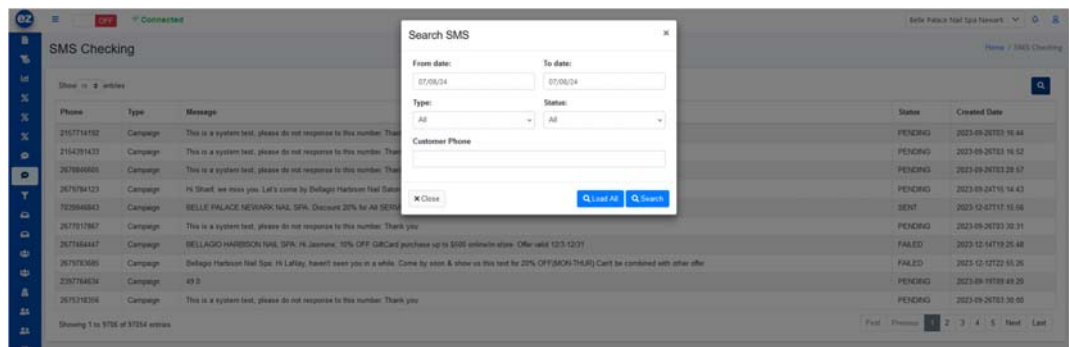
Trong hình 2.35 cho thấy, hàm getAllSmsSending sẽ thực hiện lấy các tin nhắn đã gửi đến khách hàng.



```
26 export class AppSearchSmsCheckingAdvanceComponent {
27   @Input() search: SmsSendingReportSearchModel = new SmsSendingReportSearchModel();
28   @Output() searchCompleteEvent : EventEmitter<SmsSendingReportS... = new EventEmitter<SmsSendingReportSearchModel>();
29   @ViewChild('modalWrapper', {static: true}) modalWrapper: AppModalWrapperComponent;
30   public fromDate: string;
31   public toDate: string;
32   typeSMS : {id: string, name: string[]} = [
33     {id: 'ALL', name: 'All'},
34     {id: TYPE_SMS_SENDING.SMS_BOOKING.toString(), name: 'Booking'},
35     {id: TYPE_SMS_SENDING.SMS_CAMPAIGN.toString(), name: 'Campaign'},
36     {id: TYPE_SMS_SENDING.SMS_CUSTOMER_REVIEW.toString(), name: 'Customer'}
37   ];
38   type: string = this.typeSMS[0].id;
39   statusSMS : {id: string, name: string[]} = [
40     {id: 'ALL', name: 'All'},
41     {id: STATUS_SMS_SENDING.SENT.toString(), name: 'Sent'},
42     {id: STATUS_SMS_SENDING.PENDING.toString(), name: 'Pending'},
43     {id: STATUS_SMS_SENDING.FAILED.toString(), name: 'Failed'},
44     {id: STATUS_SMS_SENDING.CANCELED.toString(), name: 'Canceled'}
45   ]
46   status: string = this.statusSMS[0].id;
47
48   no usages ▲ Azure2212
49   constructor(
50     private modal: AppModals,
51     private loading: AppLoading,
52     private alert: AppAlert,
53     private common: AppCommon,
54     private smsSendingService: SmsSendingsService
55   ) {...}
62   public query: any = {...};
63   2 usages ▲ Azure2212
64   public show(): void {
65     this.modalWrapper.show();
66   }
67   1 usage ▲ Azure2212
68   public loadAll(): void {...}
69   1 usage ▲ Azure2212
70   public searchSMS(): void {...}
71   2 usages ▲ Azure2212
```

Hình 2.36. Lớp AppSearchSmsCheckingAdvanceComponent xây dựng FontEnd cho chức năng xem lại các tin nhắn đã gửi đến khách hàng

Lớp AppSearchSmsCheckingAdvanceComponent trong hình 2.36 chứa các hàm cơ bản để lấy dữ liệu từ bên Backend. Sẽ có nhiều loại tin nhắn đã được gửi, người dùng có thể lọc ra loại tin nhắn, thời gian gửi bằng cách chọn lựa chọn tìm kiếm trên dao diện, lựa chọn đó sẽ kích hoạt hàm SearchSms để tìm theo ý người dùng muốn.



Hình 2.37. chức năng xem lại các tin nhắn đã gửi đến khách hàng sau khi hoàn thiện

### **CHƯƠNG 3: KẾT QUẢ THỰC TẬP**

Kết thúc thời gian thực tập 8 tuần tại công ty Công Ty Tư Vấn Khoa Học Công Nghệ An Phát, em đã đạt được những kết quả sau:

- Về kiến thức:

- + Em đã được cập nhật những kiến thức ngành công nghệ thông tin, đặc biệt là về các công nghệ lập trình web.
- + Em đã có cơ hội hiểu rõ hơn về quy trình phát triển phần mềm, từ các bước phân tích, thực hiện và kiểm thử phần mềm.

- Về kỹ năng:

- + Em đã nâng cao được kỹ năng lập trình, đặc biệt là kỹ năng lập trình hướng đối tượng, kỹ năng xử lý nghiệp vụ.
- + Em đã được bổ xung thêm hiểu biết về các framework thông dụng trong phát triển ứng dụng nói chung, web nói riêng
- + Em đã rèn luyện được kỹ năng làm việc nhóm, kỹ năng giao tiếp, kỹ năng giải quyết vấn đề và kỹ năng quản lý thời gian.

- Về các công việc em đã thực hiện:

- + Số lượng công việc được giao: 22
- + Số lượng công việc hoàn thành: 22
- + Số lượng công việc chưa hoàn thành: 0

### **BẢNG GHI NHẬN KẾT QUẢ THỰC TẬP HÀNG TUẦN**

Sinh viên có thể trình bày chi tiết hơn các công việc đã làm trong mỗi tuần (mẫu 6)

Cuối phần C này đính kèm là bảng GHI NHẬN KẾT QUẢ THỰC TẬP HÀNG TUẦN có chữ ký của chuyên gia hướng dẫn.



## **BẢNG GHI NHẬN KẾT QUẢ THỰC TẬP HÀNG TUẦN**

Sinh viên có thể trình bày chi tiết hơn các công việc đã làm trong mỗi tuần (mẫu 6)

Cuối phần C này đính kèm là bảng GHI NHẬN KẾT QUẢ THỰC TẬP HÀNG TUẦN có chữ ký của chuyên gia hướng dẫn.

## **BẢNG ĐÁNH GIÁ QUÁ TRÌNH THỰC TẬP TỐT NGHIỆP**

(Do chuyên gia doanh nghiệp đánh giá). mẫu 7

PHIẾU ĐÁNH GIÁ KẾT QUẢ THỰC TẬP TỐT NGHIỆP  
(Do giảng viên hướng dẫn đánh giá) mẫu 8

**Kết luận chương 3**

....

## CHƯƠNG 4. KẾT LUẬN VÀ KIẾN NGHỊ

Qua quá trình thực tập tại Công Ty Tư Vấn Khoa Học Công Nghệ An Phát trong vòng 8 tuần, em đã rút ra được những bài học quý giá sau:

- **Kiến thức:** Em đã được học hỏi rất nhiều kiến thức và kỹ năng mới, bao gồm kiến thức về phát triển phần mềm, kiến thức về quản lý dự án, và kiến thức về làm việc nhóm.
- **Kinh nghiệm:** Em đã có cơ hội được tiếp xúc với môi trường làm việc chuyên nghiệp, năng động, và được học hỏi từ các anh chị có kinh nghiệm và cả các thực tập sinh mới vô khác.
- **Phát triển bản thân:** Em đã có cơ hội phát triển bản thân, nâng cao kỹ năng làm việc nhóm, kỹ năng giao tiếp và thuyết trình, cũng như kỹ năng giải quyết vấn đề trong thời gian ngắn.

Em xin chân thành cảm ơn thầy Vũ Ngọc Thanh Sang, Ban giám đốc và các anh chị nhân viên Công Ty Tư Vấn Khoa Học Công Nghệ An Phát đã tạo điều kiện cho em có cơ hội thực tập quý giá này. Em hy vọng sẽ có cơ hội được làm việc tại công ty trong tương lai.