

**ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA CÔNG NGHỆ PHẦN MỀM**



**BÁO CÁO ĐỒ ÁN**  
**SEMINAR VỀ CÁC VẤN ĐỀ HIỆN ĐẠI CỦA**  
**CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI**

**ÁP DỤNG KIẾN TRÚC MICROFRONTEND VÀO  
ỨNG DỤNG ĐẶT PHÒNG KHÁCH SẠN BẰNG  
WEBPACK MODULE FEDERATION**

**Giảng viên phụ trách:** TS. Đinh Nguyễn Anh Dũng

**Sinh viên thực hiện:**

- Trần Tuấn Kiệt – 21522266
- Trần Thị Tuyết Mai – 21520340

# Mục lục

<b>1. Giới thiệu.....</b>	<b>5</b>
1.1. Đặt vấn đề .....	5
1.2. Mục tiêu của báo cáo .....	5
1.3. Phạm vi nghiên cứu .....	5
<b>2. Tổng quan về kiến trúc Microfrontend.....</b>	<b>5</b>
2.1. Khái niệm.....	5
2.2. Lịch sử phát triển.....	6
2.3. So sánh Microfrontend với kiến trúc truyền thống.....	7
2.4. Các thành phần chính.....	7
2.5. Các phương pháp tích hợp.....	8
2.5.1. Định tuyến: tải các ứng dụng riêng biệt .....	8
2.5.2. Phương pháp tiếp cận gói NPM .....	9
2.5.3. Microfrontend dựa trên iframe .....	9
2.5.4. WebComponents .....	10
2.5.5. Module Federation .....	10
2.6. Ví dụ .....	10
<b>2.7. Các loại tích hợp chính.....</b>	<b>12</b>
2.7.1. Build-time integration .....	12
2.7.2. Run-time integration .....	13
2.7.3. Server integration .....	13
2.7.4. Lưu ý .....	13
<b>3. Áp dụng Webpack Module Federation xây dựng Microfrontend cho ứng dụng đặt phòng khách sạn.....</b>	<b>13</b>
3.1. Sơ lược về Webpack .....	13
3.2. Webpack development server.....	14
3.3. Cài đặt Module Federation .....	16
3.3.1. Các bước để cài đặt: .....	16
3.3.2. Cách sử dụng Module Federation trong Webpack với hai ứng dụng: "Products" (Remote) và "Container" (Host): .....	17
3.4. Hiểu hơn về Module Federation .....	18
3.4.1. Quy trình đóng gói (bundling) của Webpack khi sử dụng Module Federation cho ứng dụng "Products" (Remote): .....	18
3.4.2. Quá trình sử dụng Webpack để gộp và đóng gói các tệp JavaScript từ các thành phần khác nhau	
19	
3.4.3. Quá trình tải và thực thi các tệp JavaScript giữa hai Webpack Dev Server khác nhau .....	20
3.5. Hiểu hơn về các tùy chọn cấu hình.....	21
3.5.1. Đối với Host: .....	21
3.5.2. Đối với Remote: .....	21
<b>4. Quy trình phát triển .....</b>	<b>22</b>

<b>4.1.</b>	<b>Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project) .....</b>	<b>23</b>
<b>4.2.</b>	<b>Yêu cầu dẫn đến lựa chọn kiến trúc.....</b>	<b>24</b>
4.2.1.	Không có sự liên kết giữa các dự án con.....	27
4.2.2.	Gần như không có sự kết hợp giữa container và các ứng dụng con.....	28
4.2.3.	CSS từ một dự án không nên ảnh hưởng đến dự án khác .....	28
4.2.4.	Kiểm soát phiên bản (monorepo so với separate) không nên có bất kỳ tác động nào đến toàn bộ dự án	28
4.2.5.	Container phải có khả năng quyết định luôn sử dụng phiên bản mới nhất của microfrontend hoặc chỉ định một phiên bản cụ thể.....	29
<b>4.3.</b>	<b>Tại sao phải sử dụng mount function để render UI?.....</b>	<b>29</b>
<b>5.</b>	<b>Lợi ích của Microfrontend trong phát triển ứng dụng .....</b>	<b>30</b>
5.1.	Triển khai nhanh hơn và quản lý bản phát hành tốt hơn.....	31
5.2.	Nhiều nhóm có trách nhiệm khác nhau .....	31
5.3.	Tự do công nghệ.....	31
5.4.	Dễ dàng mở rộng quy mô .....	31
5.5.	Triển khai liên tục.....	31
<b>6.</b>	<b>Ứng dụng Microfrontend trong lĩnh vực đặt phòng khách sạn .....</b>	<b>31</b>
6.1.	<b>Các module microfrontend chính: .....</b>	<b>31</b>
6.1.1.	Team Khám phá .....	31
6.1.2.	Team Quản lý đặt phòng .....	32
6.1.3.	Team Thanh toán .....	32
6.1.4.	Team Quản lý người dùng .....	32
6.1.5.	Team Đánh giá và Xếp hạng .....	32
6.1.6.	Team Marketing .....	33
6.2.	<b>Một số ví dụ trùu tượng hoá các microfrontend trên trang .....</b>	<b>34</b>
6.2.1.	Trang chủ:.....	34
6.2.2.	Trang chi tiết.....	35
<b>7.</b>	<b>Thách thức khi áp dụng Microfrontend .....</b>	<b>35</b>
7.1.	<b>Kích thước tải trọng .....</b>	<b>36</b>
7.2.	<b>Giao tiếp giữa các trang .....</b>	<b>36</b>
7.3.	<b>Sự khác biệt về thiết kế.....</b>	<b>36</b>
7.4.	<b>Độ phức tạp về mặt vận hành .....</b>	<b>37</b>
<b>8.</b>	<b>Thực tiễn và ví dụ thành công.....</b>	<b>37</b>
8.1.	<b>Các công ty đã áp dụng Microfrontend .....</b>	<b>37</b>
8.2.	<b>Nghiên cứu điển hình về ứng dụng đặt phòng khách sạn (Case Study) .....</b>	<b>37</b>
<b>9.</b>	<b>Kết luận .....</b>	<b>38</b>
9.1.	<b>Tóm tắt các điểm chính .....</b>	<b>38</b>
9.2.	<b>Đề xuất và hướng phát triển tương lai.....</b>	<b>38</b>
<b>10.</b>	<b>Kế hoạch thực hiện.....</b>	<b>39</b>

11. *Tài liệu tham khảo*.....40

## 1. Giới thiệu

### 1.1. Đặt vấn đề

Trong thời đại công nghệ số hiện nay, nhu cầu sử dụng các ứng dụng đặt phòng khách sạn ngày càng gia tăng, kéo theo sự cạnh tranh khốc liệt trong ngành du lịch và khách sạn. Việc phát triển các ứng dụng này không chỉ đơn thuần là cung cấp dịch vụ đặt phòng, mà còn phải đảm bảo trải nghiệm người dùng tốt nhất, tốc độ tải trang nhanh và khả năng mở rộng linh hoạt. Trong bối cảnh đó, kiến trúc Microfrontend đã nổi lên như một giải pháp tối ưu giúp các nhà phát triển ứng dụng giải quyết những thách thức này.

### 1.2. Mục tiêu của báo cáo

Mục tiêu của báo cáo này là trình bày rõ ràng về kiến trúc Microfrontend, các lợi ích mà nó mang lại trong phát triển ứng dụng đặt phòng khách sạn, cũng như các thách thức và giải pháp khi áp dụng kiến trúc này. Báo cáo cũng sẽ xem xét một số ví dụ thực tiễn từ các công ty đã thành công trong việc triển khai Microfrontend trong ứng dụng của họ.

### 1.3. Phạm vi nghiên cứu

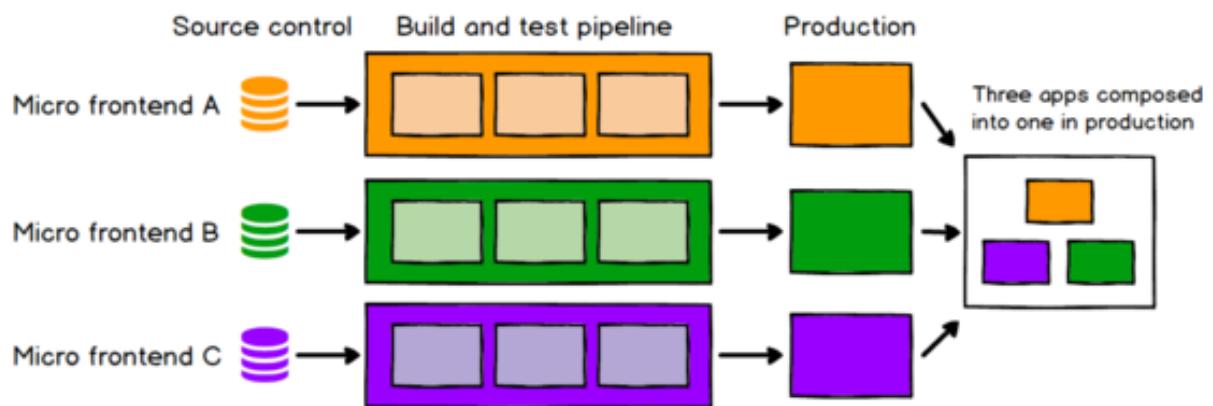
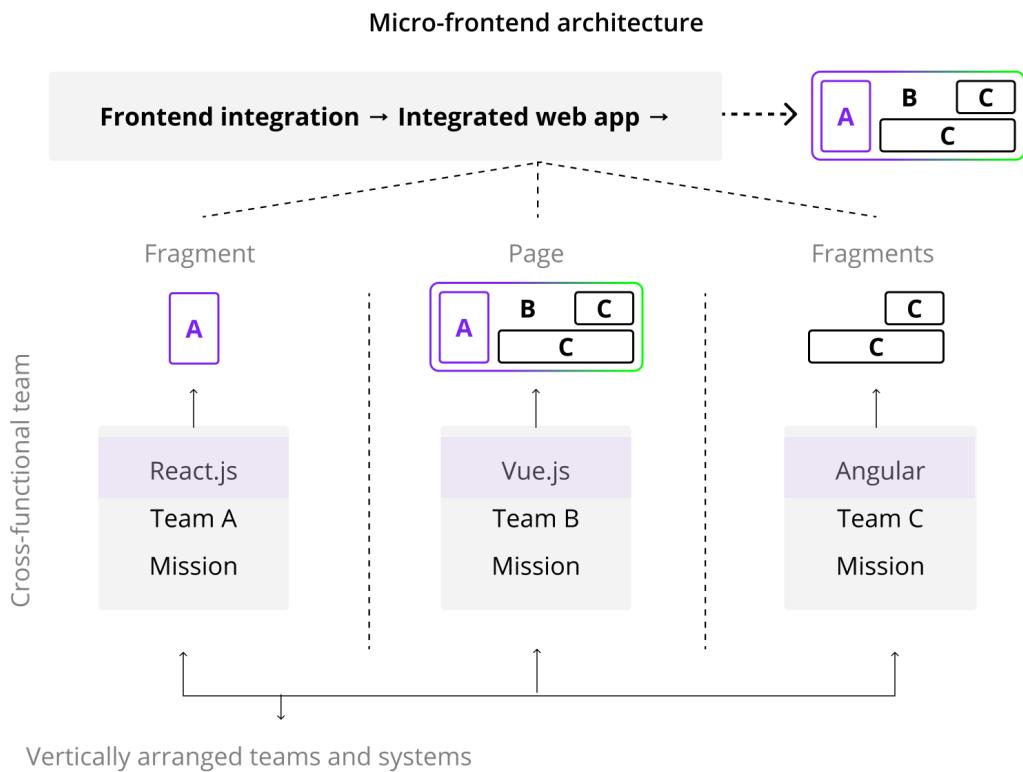
Báo cáo sẽ tập trung vào việc phân tích và đánh giá kiến trúc Microfrontend trong bối cảnh phát triển ứng dụng đặt phòng khách sạn. Các vấn đề liên quan đến quá trình phát triển, triển khai, bảo trì và nâng cấp ứng dụng cũng sẽ được đề cập. Đồng thời, báo cáo cũng sẽ trình bày những xu hướng mới trong lĩnh vực công nghệ phần mềm có liên quan đến Microfrontend và ứng dụng đặt phòng khách sạn.

## 2. Tổng quan về kiến trúc Microfrontend

### 2.1. Khái niệm

Microfrontends là một phong cách kiến trúc của phát triển web frontend, trong đó **một ứng dụng được chia thành các tính năng – microfrontends – và được phân phối độc lập**. Mỗi microfrontend thường đại diện cho một tính năng hoặc một phần của giao diện người dùng, cho phép các nhóm phát triển làm việc song song mà không bị ảnh hưởng lẫn nhau. Điều này được thực hiện để cải thiện chất lượng phân phối và hiệu quả của các nhóm chịu trách nhiệm về mã frontend.

Mỗi microfrontend có thể có kho lưu trữ mã nguồn, bộ phụ thuộc, bộ kiểm thử tự động và đường ống phân phối riêng. Mỗi microfrontend rất có thể thuộc sở hữu của một nhóm frontend duy nhất và được phát triển, thử nghiệm và triển khai độc lập với các tính năng khác, giúp tăng hiệu quả của quy trình phát triển.



<https://martinfowler.com/articles/micro-frontends.html>

## 2.2. Lịch sử phát triển

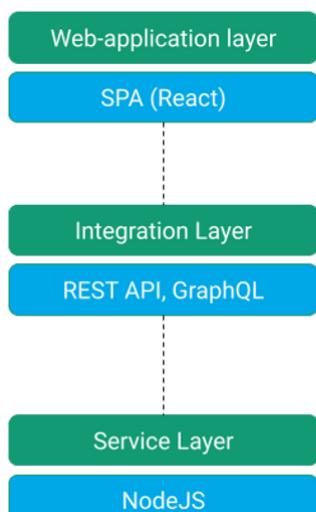
Thuật ngữ “microfrontend” mới được giới thiệu khá gần đây – vào năm 2016, nó được Michael Geers sử dụng và vào cuối năm đó, nó xuất hiện trong ThoughtWorks Technology Radar. Tuy nhiên, các nhà phát triển đã sử

dụng phương pháp tiếp cận kiến trúc này trong một thời gian khá dài để hưởng lợi từ tính mô-đun, triển khai độc lập và khả năng mở rộng trong phát triển frontend.

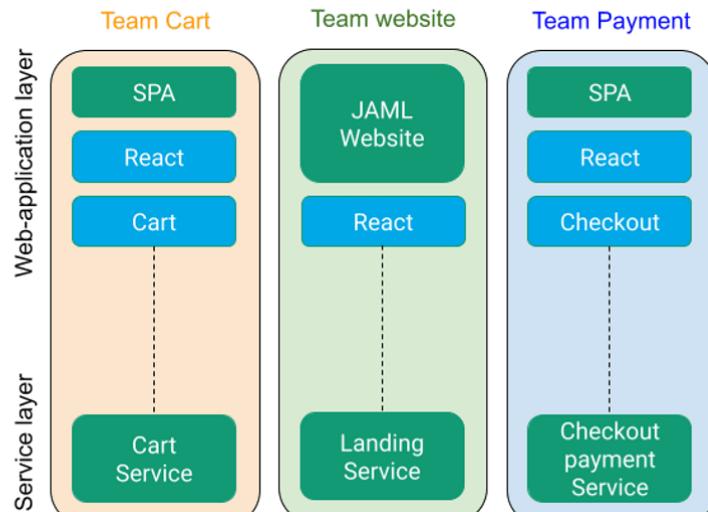
### 2.3. So sánh Microfrontend với kiến trúc truyền thống

- Kiến trúc truyền thống: Thường sử dụng một ứng dụng đơn nhất, trong đó tất cả các phần của ứng dụng được phát triển và triển khai cùng nhau. Điều này có thể dẫn đến việc khó khăn trong việc bảo trì, cập nhật và mở rộng ứng dụng.
- Microfrontend: Mỗi phần của ứng dụng được phát triển độc lập và có thể sử dụng công nghệ khác nhau. Điều này cho phép các nhóm phát triển tự do chọn công nghệ phù hợp nhất cho từng tính năng mà không làm ảnh hưởng đến các phần khác của ứng dụng.

## Monolithic



## Micro Frontends



<https://leanylabs.com/blog/micro-frontends-overview/>

### 2.4. Các thành phần chính

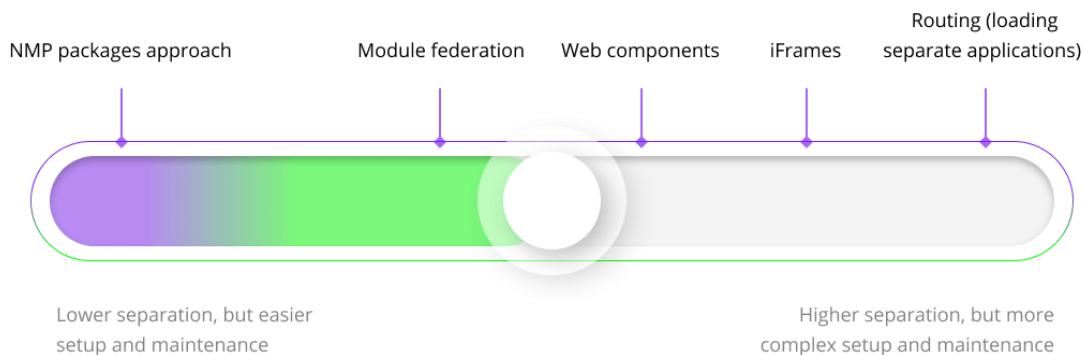
Microfrontend có thể được phân chia thành các thành phần sau:

- Tính năng (Feature): Mỗi microfrontend có thể đại diện cho một tính năng hoặc một phần của giao diện người dùng, như thanh tìm kiếm, giỏ hàng, hoặc trang chi tiết khách sạn.
- Giao tiếp: Các microfrontend cần có cách để giao tiếp với nhau, thường thông qua các API hoặc sự kiện.

- Quản lý trạng thái: Do mỗi microfrontend có thể quản lý trạng thái của riêng nó, việc đồng bộ hóa trạng thái giữa các phần là rất quan trọng để đảm bảo trải nghiệm người dùng liên tục.
- Triển khai: Mỗi microfrontend có thể được triển khai độc lập, giúp cho việc cập nhật hoặc sửa lỗi trở nên dễ dàng và nhanh chóng hơn.

## 2.5. Các phương pháp tích hợp

Các cách tiếp cận tích hợp micro frontend khác nhau được mô tả tốt nhất là một thanh trượt. Ở một bên của thanh trượt, ứng dụng ít tách biệt hơn và do đó có ít lợi ích micro frontend hơn. Để bảo vệ cho nó, các cách tiếp cận này có thiết lập dễ dàng hơn nhiều – chúng yêu cầu một phiên bản khung micro frontend và một phiên bản thư viện. Giả sử bạn di chuyển thanh trượt sang một bên khác có nhiều sự tách biệt hơn. Trong trường hợp đó, ứng dụng sẽ có độ phức tạp thiết lập và chi phí bảo trì lớn hơn, nhưng tất cả các lợi ích micro frontend đều có thể tỏa sáng hoàn toàn.



Hãy cùng xem xét một số phương pháp tiếp cận phổ biến nhất.

### 2.5.1. Định tuyến: tải các ứng dụng riêng biệt

Ý tưởng đằng sau cách tiếp cận này là ứng dụng được tách biệt dựa trên định tuyến và mỗi tuyến tải một ứng dụng hoàn toàn độc lập, bao gồm các gói khung bên trong. Trong trường hợp này, cơ sở hạ tầng hoạt động như một shell và chuyển hướng giữa các ứng dụng xảy ra thông qua ví dụ như API gateway hoặc CDN.

Định tuyến hoạt động cho tích hợp cấp trang khi các nhóm khác nhau sở hữu các trang của ứng dụng và mỗi micro frontend được coi là một ứng dụng một trang.

Định tuyến có thể đạt được bằng các liên kết HTML. Trong trường hợp như vậy, trang sẽ tải lại trên mỗi trang. Không có khả năng chia sẻ dữ liệu với cách tiếp cận này, chỉ có các tham số truy vấn, lưu trữ phiên hoặc lưu trữ dữ liệu trong cơ sở dữ liệu.

Hãy cùng xem cách thức hoạt động theo góc nhìn của người dùng. Khi người dùng điều hướng đến một URL cụ thể trong ứng dụng, ứng dụng định tuyến trung tâm sẽ nhận được yêu cầu. Nó xác định micro frontend nào sẽ xử lý yêu cầu dựa trên URL. Micro frontend đã chọn sẽ được tải và hiển thị cho người dùng, trong khi các micro frontend khác vẫn ẩn.

Nếu cần phải hiển thị một trang mà không cần tải lại, có thể áp dụng một shell ứng dụng chia sẻ hoặc một meta-framework như single-spa. Shell ứng dụng bao gồm HTML, CSS và JavaScript tối thiểu. Người dùng sẽ thấy một trang được hiển thị tĩnh ngay lập tức ngay cả khi dữ liệu được yêu cầu vẫn đang chờ xử lý từ máy chủ.

Tất cả các tương tác và bản cập nhật đều được tải mà không cần tải chúng từ máy chủ, do đó trang không được làm mới theo yêu cầu của mỗi người dùng.

### 2.5.2. Phương pháp tiếp cận gói NPM

Mỗi micro frontend được phát hành dưới dạng một gói riêng lẻ với phương pháp quản lý gói nút (NPM). Sau đó, ứng dụng shell sẽ liệt kê các gói này và kết hợp chúng vào ứng dụng tổng thể. Phương pháp này có lợi thế là thiết lập khá đơn giản. Tuy nhiên, nó đi kèm với một số nhược điểm nhất định, chẳng hạn như yêu cầu đối với các phiên bản phụ thuộc đơn lẻ và không thể phát hành micro frontend mà không có bản phát hành shell, điều này có thể tạo ra sự phụ thuộc giữa các nhóm và cản trở các bản cập nhật.

### 2.5.3. Microfrontend dựa trên iframe

Phương pháp iframe nhúng từng micro frontend vào iframe của riêng nó và một ứng dụng trang đơn (SPA) hoạt động như một container. Về cơ bản, đây là một tài liệu HTML được đặt bên trong một tài liệu HTML khác. SPA đang điều phối giao tiếp giữa các iframe.

Iframe là một phương pháp tương đối cũ để xây dựng kiến trúc micro frontend. Mặc dù nó cung cấp mức độ tách biệt tốt, nhưng cần phải giải quyết nhiều vấn đề để nó hoạt động (ví dụ: bảo mật, chia sẻ dữ liệu).

#### **2.5.4. WebComponents**

Các thành phần web ngũ ý xây dựng từng micro frontend như một thành phần riêng biệt có thể được triển khai độc lập dưới dạng tệp .js. Ứng dụng tải và hiển thị chúng trong các trình giữ chỗ được tạo riêng trong bố cục. Các thành phần web cho trình duyệt biết khi nào và ở đâu để tạo thành phần.

Nhược điểm của cách tiếp cận này là mỗi thành phần web đều lớn và cần phải đóng gói phiên bản khung của nó. Vì vậy, mặc dù bạn có thể tách nhiều thành phần tùy ý, hiệu suất phải được theo dõi cẩn thận.

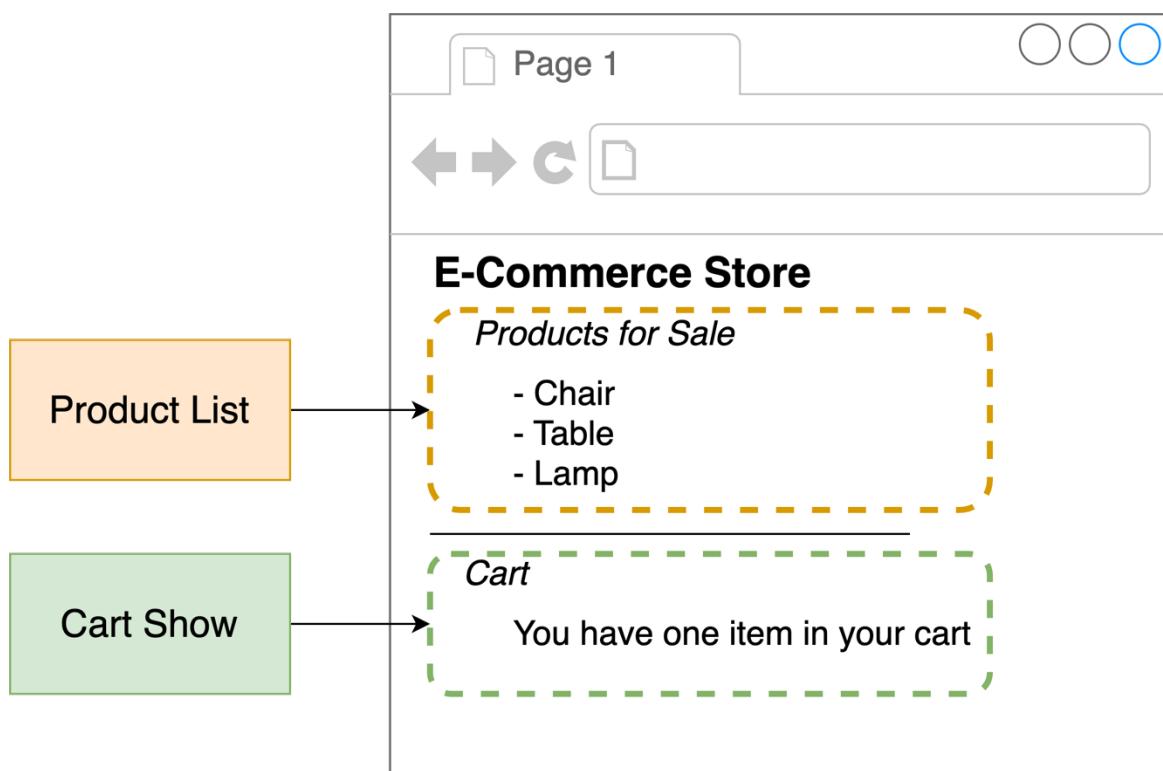
#### **2.5.5. Module Federation**

Module Federation là một cách tiếp cận khác giúp làm cho tất cả các thành phần micro frontend trông giống như một ứng dụng. Trên thanh trượt của các mức độ phân tách khác nhau được sử dụng trong micro frontend, module federation có thể được đặt ở đâu đó ở giữa. Module federation được phát hành cùng với Webpack 5 (2020), vì vậy nó hiện đại và có thể được các khuôn khổ khác nhau áp dụng.

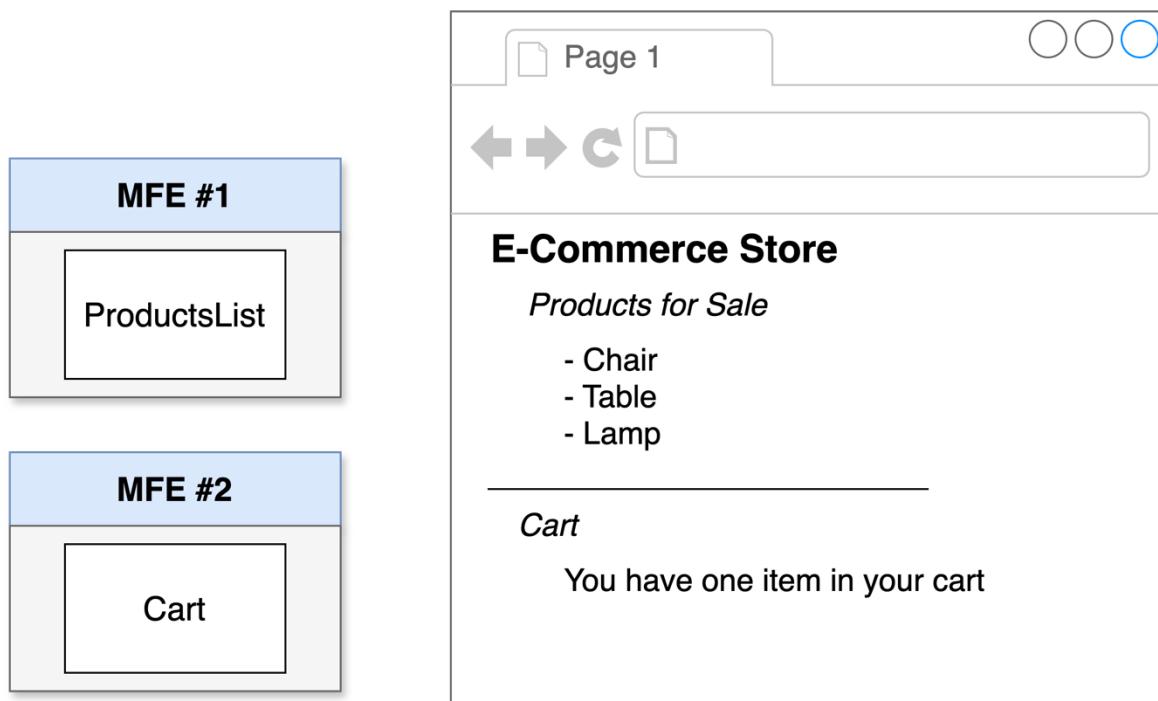
Module Federation cho phép tải micro frontend khi chạy vào ứng dụng shell mà không phụ thuộc vào thời gian xây dựng (chỉ cần xác định 'hợp đồng' tích hợp). Điều này đặc biệt hữu ích khi tùy chỉnh các trang của bạn cho các khách hàng cụ thể. Trang web của bạn có thể có một mô-đun 'thanh toán' được tải từ một gói 'chung' cho hầu hết các khách hàng và một mô-đun 'tùy chỉnh' (thậm chí có thể do nhóm phát triển của khách hàng xây dựng) cho những khách hàng khác. Module federation cho phép bạn thay đổi cấu hình micro frontend của mình khi chạy.

### **2.6. Ví dụ**

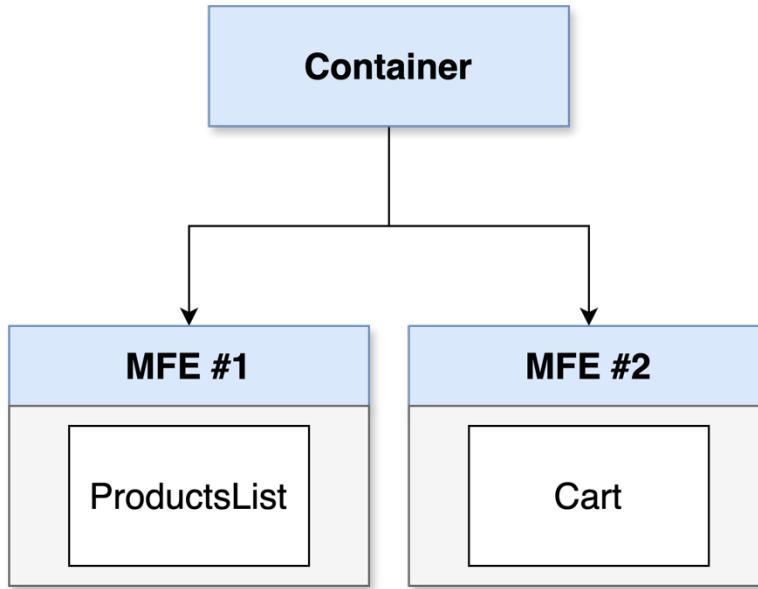
Giả sử chúng ta có một trang thương mại điện tử với 2 thành phần là danh sách sản phẩm và giỏ hàng:



Ta sẽ chia ra được 2 microfrontend cho 2 thành phần này:



Nhìn một cách tổng quan, nó sẽ được biểu diễn như sau:



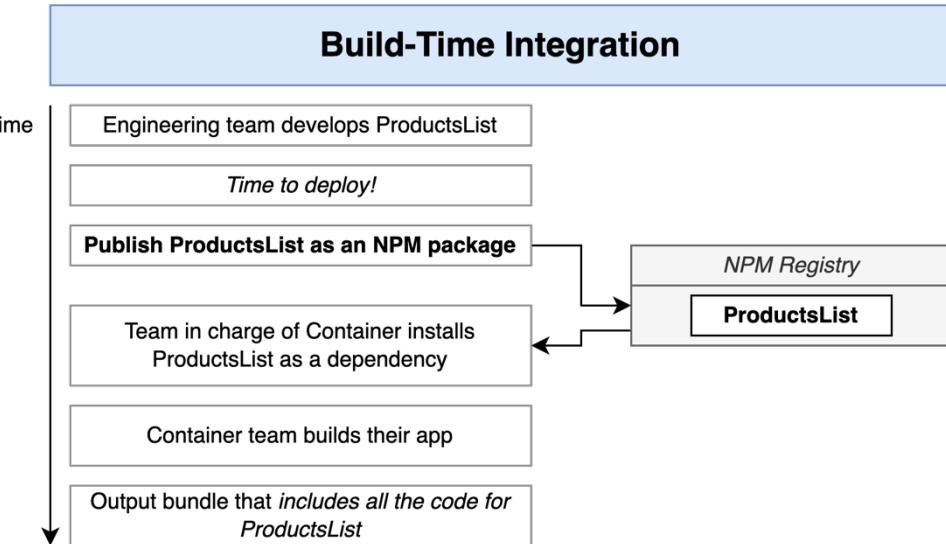
Nhưng vấn đề đặt ra là: **Container có thể truy cập vào mã nguồn trong MFE #1 và #2 khi nào và bằng cách nào?**

Sẽ có 3 phương pháp chính để Container có thể truy cập vào các MFE này, và nó được gọi là các loại tích hợp. Sẽ được làm rõ ở phần tiếp theo.

## 2.7. Các loại tích hợp chính

### 2.7.1. Build-time integration

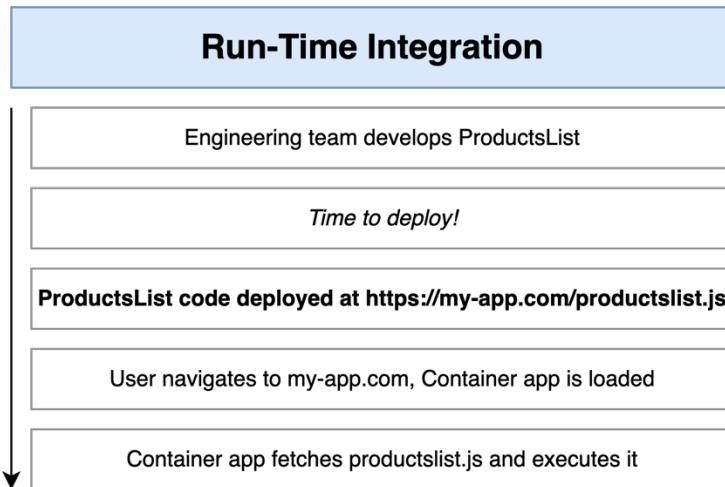
Trước khi Container được tải trong trình duyệt, nó sẽ được truy cập vào mã nguồn ProductsList.



Với cách tiếp cận này, sẽ có lợi ích là dễ cài đặt và dễ hiểu, nhưng Container phải được triển khai lại mỗi khi ProductsList được cập nhật và có ý định (cám dỗ) kết hợp chặt chẽ Container + ProductsList với nhau.

### 2.7.2. Run-time integration

Sau khi Container được tải trong trình duyệt, nó sẽ được truy cập vào mã nguồn ProductsList.



Với cách tiếp cận này, ProductList có thể được độc lập bất cứ lúc nào và có thể triển khai nhiều phiên bản khác nhau của ProductsList và Container có thể quyết định sử dụng phiên bản nào, nhưng công cụ và thiết lập phức tạp hơn nhiều.

### 2.7.3. Server integration

Trong khi gửi JS xuống để tải Container, máy chủ quyết định có bao gồm nguồn ProductsList hay không.

### 2.7.4. Lưu ý

Trong phạm vi đề tài này, chúng ta sẽ chỉ tập trung vào Run-time Integration sử dụng Webpack Module Federation.

## 3. Áp dụng Webpack Module Federation xây dựng Microfrontend cho ứng dụng đặt phòng khách sạn

### 3.1. Sơ lược về Webpack

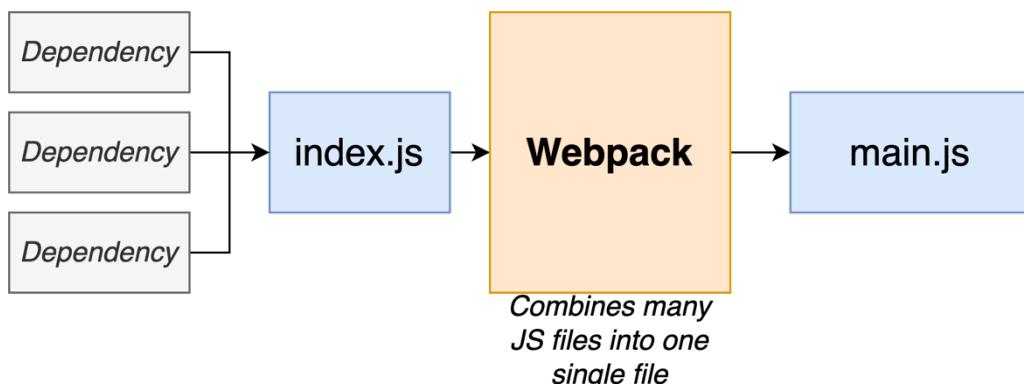
Webpack là một trình đóng gói mô-đun mã nguồn mở và miễn phí cho JavaScript. Nó được tạo chủ yếu cho JavaScript, nhưng nó có thể chuyển đổi các tài sản giao diện người dùng như HTML, CSS và hình ảnh nếu các trình tải

tương ứng được bao gồm. Webpack lấy các mô-đun có phụ thuộc và tạo các tài sản tĩnh biểu diễn các mô-đun đó.

Webpack lấy các phụ thuộc và tạo biểu đồ phụ thuộc cho phép các nhà phát triển web sử dụng phương pháp tiếp cận mô-đun cho mục đích phát triển ứng dụng web của họ. Nó có thể được sử dụng từ dòng lệnh hoặc có thể được định cấu hình bằng tệp cấu hình có tên là webpack.config.js. Tệp này định nghĩa các quy tắc, plugin, v.v. cho một dự án. (Webpack có khả năng mở rộng cao thông qua các quy tắc cho phép các nhà phát triển viết các tác vụ tùy chỉnh mà họ muốn thực hiện khi đóng gói các tệp lại với nhau.)

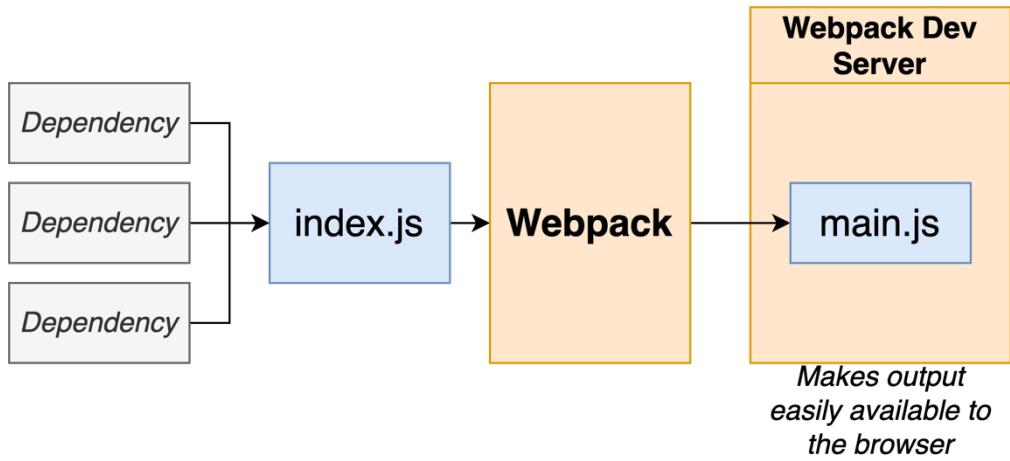
Node.js là bắt buộc để sử dụng Webpack.

Webpack cung cấp mã theo yêu cầu bằng cách sử dụng biệt danh là **code-splitting**. Hai kỹ thuật tương tự được Webpack hỗ trợ khi nói đến phân tách mã động. Phương pháp đầu tiên và được khuyến nghị là sử dụng cú pháp import() tuân thủ đề xuất ECMAScript cho các lần nhập động. Phương pháp cũ dành riêng cho Webpack là sử dụng require.ensure.

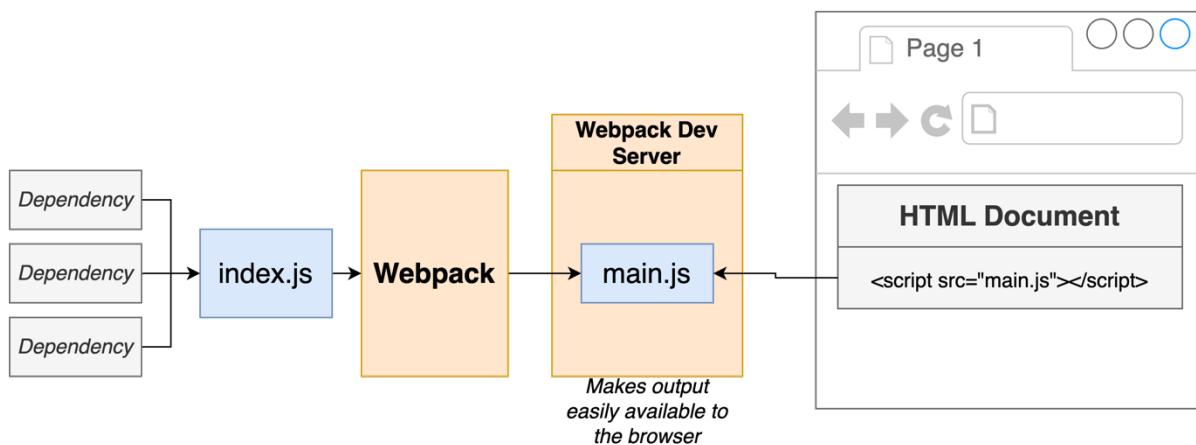


### 3.2. Webpack development server

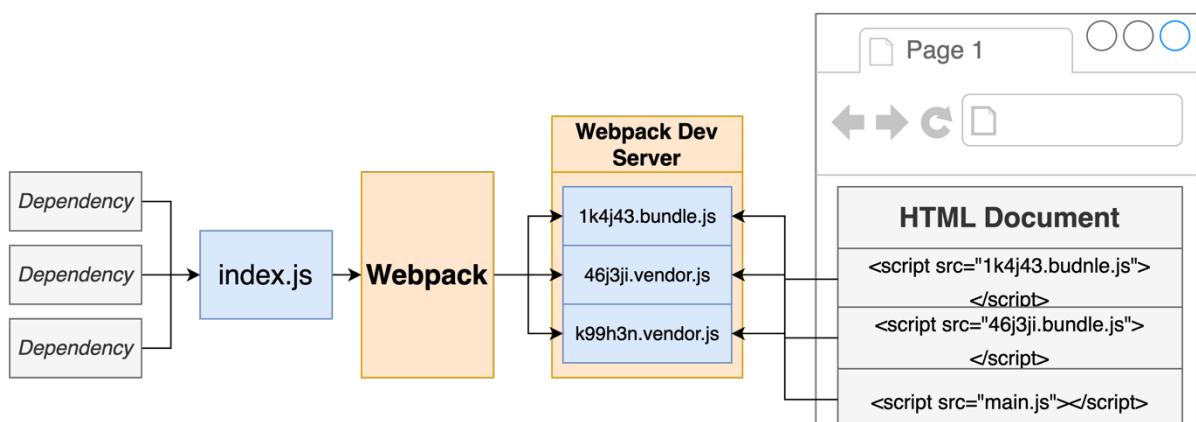
Webpack cũng cung cấp một máy chủ phát triển tích hợp, webpack-dev-server, có thể được sử dụng như một máy chủ HTTP để phục vụ các tệp trong khi phát triển. Nó cũng cung cấp khả năng sử dụng hot module replacement (HMR), cập nhật mã trên trang web mà không yêu cầu nhà phát triển tải lại trang.



Sau khi build webpack dev server sẽ tạo ra một file main.js (main bundle).

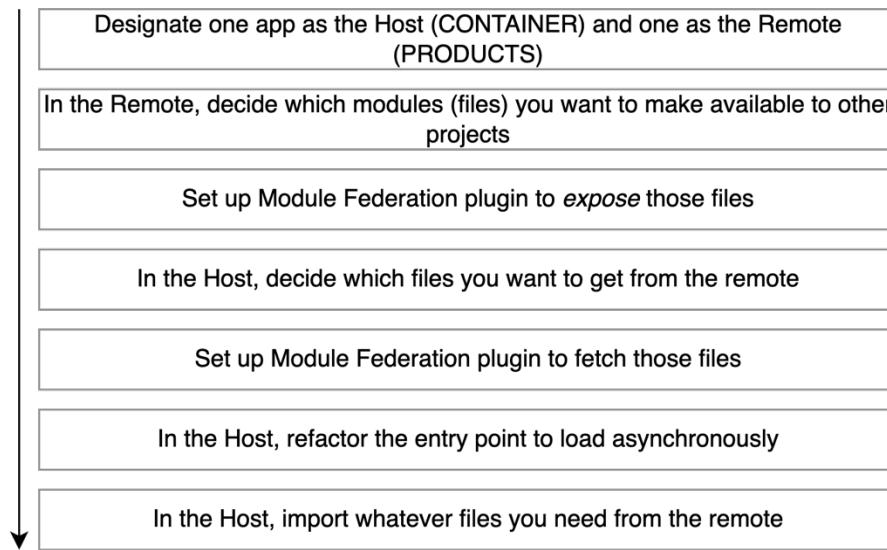


Các phần tử HTML có thể được xử lý logic thông qua các file js đã được build (bundle) này.



### 3.3.Cài đặt Module Federation

#### 3.3.1. Các bước để cài đặt:

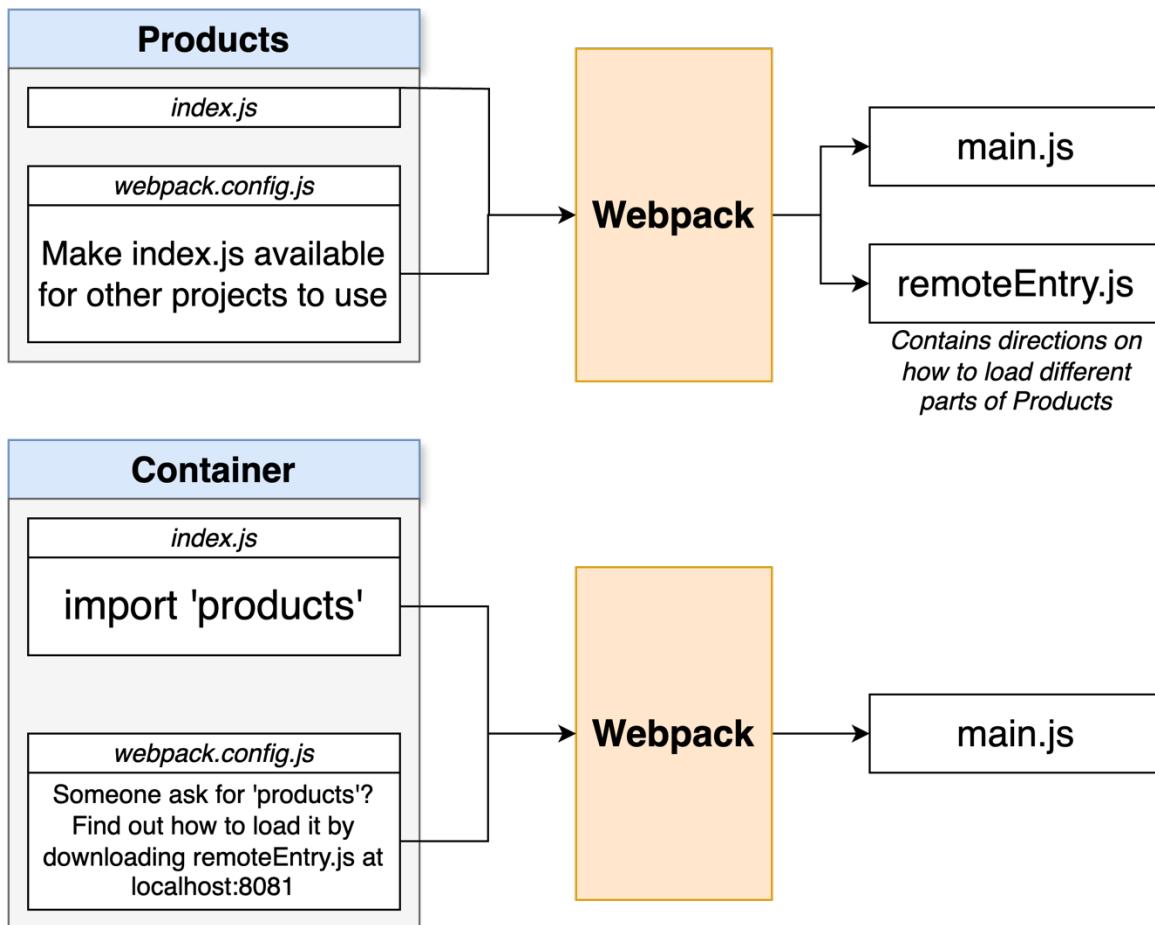


- Chỉ định Host (CONTAINER) và Remote (PRODUCTS): Chọn một ứng dụng sẽ đóng vai trò là "Host" - ứng dụng chính chịu trách nhiệm tải các module từ ứng dụng khác, và một ứng dụng khác sẽ là "Remote" - ứng dụng cung cấp các module mà Host có thể sử dụng.
- Xác định các module trong Remote: Trong ứng dụng Remote, bạn cần xác định rõ các module (tệp JavaScript hoặc các thành phần) mà bạn muốn chia sẻ, cho phép các ứng dụng khác truy cập và sử dụng.
- Thiết lập Module Federation trong Remote: Cấu hình plugin Module Federation trong ứng dụng Remote để "xuất" các module mà bạn đã xác định. Điều này sẽ làm cho các module có sẵn để các ứng dụng khác có thể truy cập vào.
- Xác định các module cần dùng trong Host: Trong ứng dụng Host, quyết định rõ ràng các module mà bạn muốn lấy từ ứng dụng Remote. Điều này có thể bao gồm các thành phần, thư viện, hoặc bất kỳ phần mã nào bạn cần sử dụng từ Remote.
- Cấu hình Module Federation trong Host: Thiết lập plugin Module Federation trong ứng dụng Host để "lấy" các module từ ứng dụng Remote. Plugin này sẽ quản lý việc liên kết và tải các module từ Remote vào Host.
- Điều chỉnh cách tải trong Host để bắt đồng bộ: Trong ứng dụng Host, cần cấu hình lại điểm nhập (entry point) để tải các module từ Remote theo

cách bất đồng bộ (asynchronously). Điều này giúp tối ưu hiệu suất và đảm bảo các module được tải khi cần thiết mà không làm chậm ứng dụng.

- Nhập các module từ Remote vào Host: Khi đã cấu hình xong, bạn có thể nhập các module từ ứng dụng Remote vào ứng dụng Host như thể chúng là các phần của ứng dụng chính. Điều này giúp Host sử dụng trực tiếp các thành phần hoặc chức năng từ Remote mà không cần phải sao chép mã.

### 3.3.2. Cách sử dụng Module Federation trong Webpack với hai ứng dụng: "Products" (Remote) và "Container" (Host):



#### Products (Remote):

- `index.js`: Đây là file chính mà bạn muốn cung cấp cho các dự án khác để sử dụng.
- `webpack.config.js`: Cấu hình này sẽ thiết lập Module Federation để làm cho `index.js` khả dụng cho các ứng dụng khác. Cấu hình này sẽ xuất một file `remoteEntry.js`, chứa thông tin cần thiết để ứng dụng khác (Container) có thể tải các module từ đây.

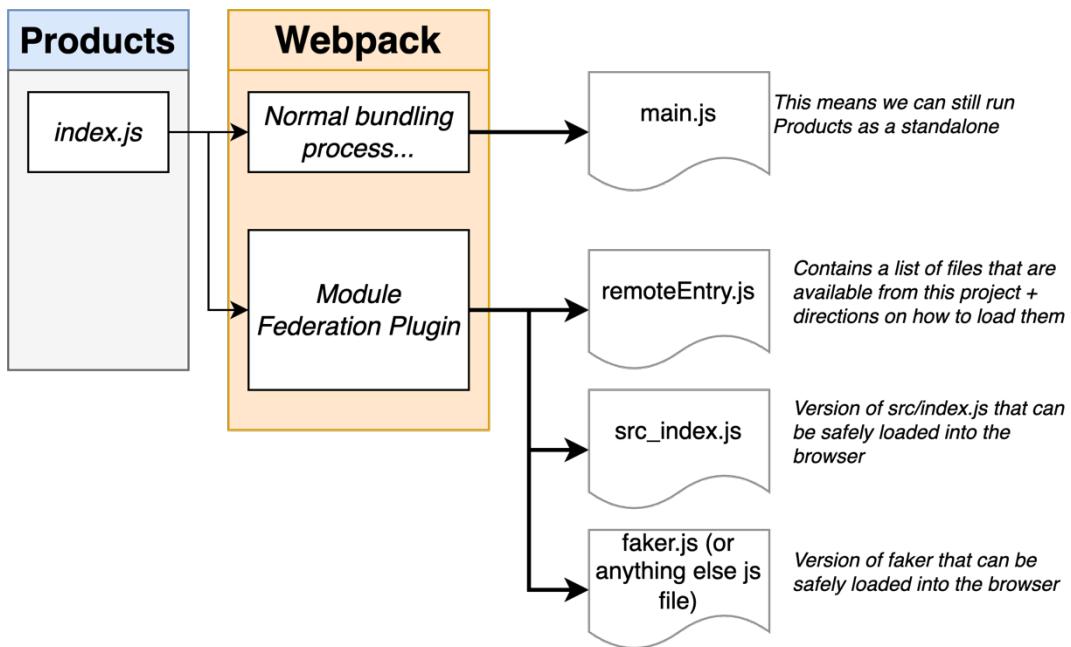
- Kết quả: Sau khi chạy Webpack, bạn sẽ có hai file: main.js (mã chính của ứng dụng) và remoteEntry.js (chứa các thông tin cần thiết cho việc chia sẻ module).

Container (Host):

- index.js: Đây là nơi mà bạn muốn nhập và sử dụng các module từ ứng dụng "Products".
- webpack.config.js: Cấu hình này sẽ thiết lập Module Federation để nhập các module từ Products. Nó chứa các thông tin để tìm và tải remoteEntry.js từ "Products" thông qua một URL (ví dụ: localhost:8081). Khi cần sử dụng products, ứng dụng sẽ tự động tải module từ remoteEntry.js.
- Kết quả: Sau khi chạy Webpack, bạn sẽ có file main.js chứa mã của ứng dụng Container, cùng với cơ chế tải các module từ Remote.

### 3.4. Hiểu hơn về Module Federation

#### 3.4.1. Quy trình đóng gói (bundling) của Webpack khi sử dụng Module Federation cho ứng dụng "Products" (Remote):



Products (Remote):

- index.js: Đây là file nguồn chính mà bạn muốn chia sẻ cho các ứng dụng khác thông qua Module Federation.

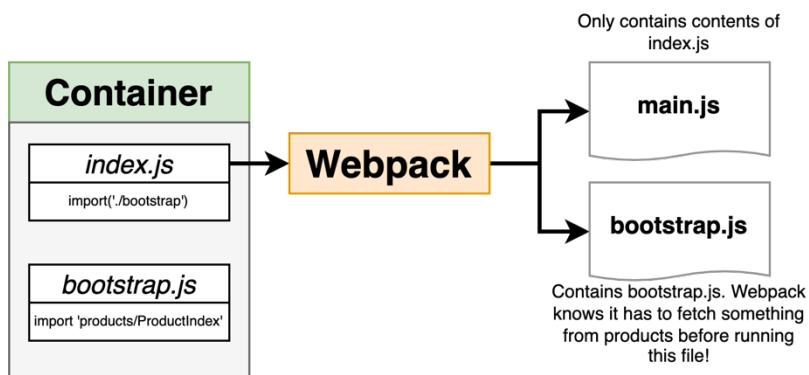
Webpack:

- Normal bundling process: Webpack thực hiện quá trình đóng gói bình thường cho toàn bộ mã nguồn của ứng dụng. Điều này sẽ tạo ra một file bundle chính, ví dụ như main.js.
- Module Federation Plugin: Plugin này được thêm vào cấu hình Webpack để làm cho một số module có thể được chia sẻ và truy cập từ các ứng dụng khác. Nó tạo ra một file đặc biệt, ví dụ như remoteEntry.js, chứa thông tin về các module mà ứng dụng "Products" cung cấp.

Kết quả sau khi chạy Webpack:

- main.js: File chính chứa mã đã được đóng gói của ứng dụng.
- remoteEntry.js: File chứa metadata và các thông tin cần thiết để các ứng dụng khác có thể tải các module từ "Products". Đây là file quan trọng nhất trong cơ chế chia sẻ module.
- src\_index.js, faker.js (hoặc bất kỳ file JavaScript nào khác): Các file này đại diện cho các phần mã khác mà "Products" có thể sử dụng hoặc chia sẻ. Tất cả sẽ được quản lý bởi Webpack.

### 3.4.2. Quá trình sử dụng Webpack để gộp và đóng gói các tệp JavaScript từ các thành phần khác nhau



Container: Đại diện cho thư mục chứa các tệp mã nguồn chính, gồm hai tệp:

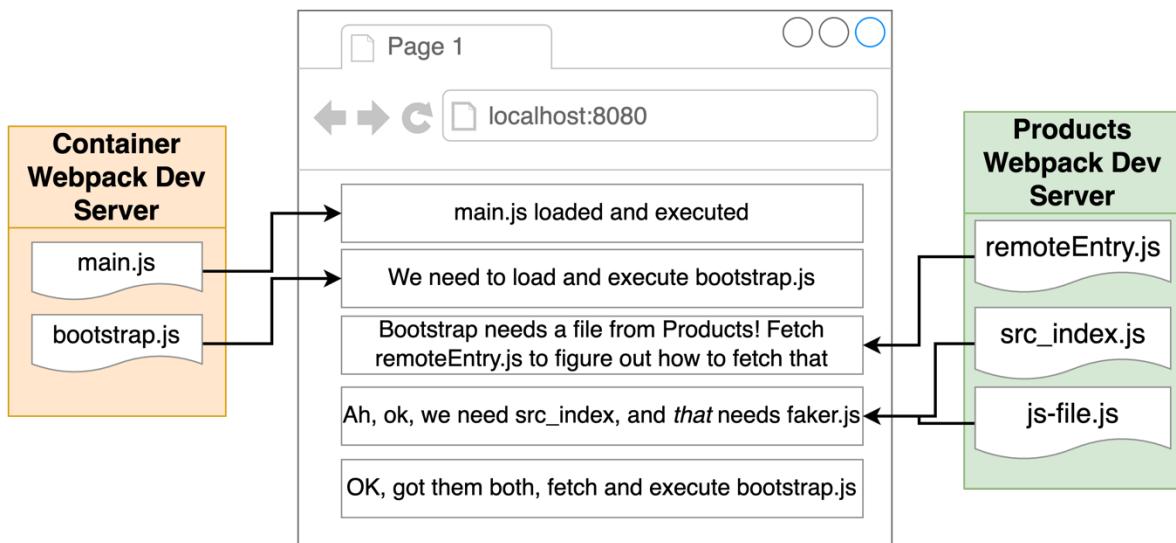
- index.js: Tệp này nhập (import) các module khác, chẳng hạn như ./bootstrap.
- bootstrap.js: Tệp này nhập thêm một module khác, ví dụ như products/ProductIndex.

Webpack: Công cụ này được sử dụng để đóng gói (bundle) các tệp JavaScript từ Container thành các tệp độc lập:

- main.js
- bootstrap.js

Cả hai tệp này sau đó có thể được sử dụng bởi ứng dụng web, giúp tối ưu hóa quá trình quản lý mã nguồn và tăng hiệu suất khi triển khai.

### 3.4.3. Quá trình tải và thực thi các tệp JavaScript giữa hai Webpack Dev Server khác nhau



Container Webpack Dev Server (bên trái) chứa hai tệp:

- main.js
- bootstrap.js

Products Webpack Dev Server (bên phải) chứa ba tệp:

- `remoteEntry.js`
- `src_index.js`
- `js-file.js`

Quá trình này diễn ra như sau:

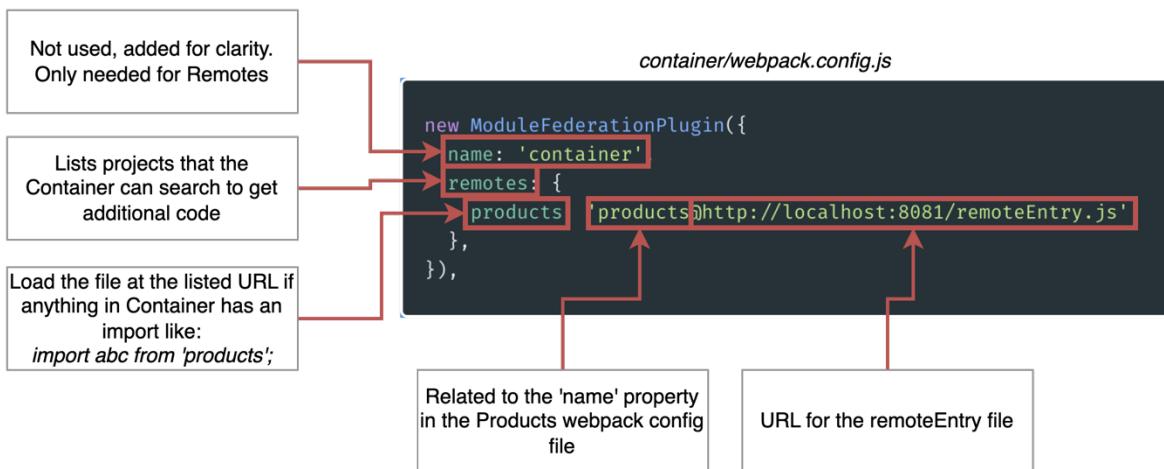
- Trình duyệt truy cập trang web tại `localhost:8080`.
- `main.js` được tải và thực thi trước.
- Sau đó, `main.js` yêu cầu tải và thực thi `bootstrap.js`.
- Tuy nhiên, `bootstrap.js` cần một tệp từ `Products`. Do đó, nó gửi yêu cầu tải `remoteEntry.js` từ `Products Webpack Dev Server` để biết cách tải các tệp khác.
- Sau khi `remoteEntry.js` được tải, nó cho biết rằng cần tải `src_index.js`, và `src_index.js` cũng yêu cầu thêm `faker.js`.

- Cuối cùng, khi tất cả các tệp cần thiết đã được tải, bootstrap.js được thực thi.

Quá trình này là một ví dụ về việc chia sẻ module giữa các hệ thống khác nhau trong một ứng dụng sử dụng Webpack Module Federation, giúp quản lý và chia sẻ các phần của mã nguồn giữa các ứng dụng hoặc dự án độc lập.

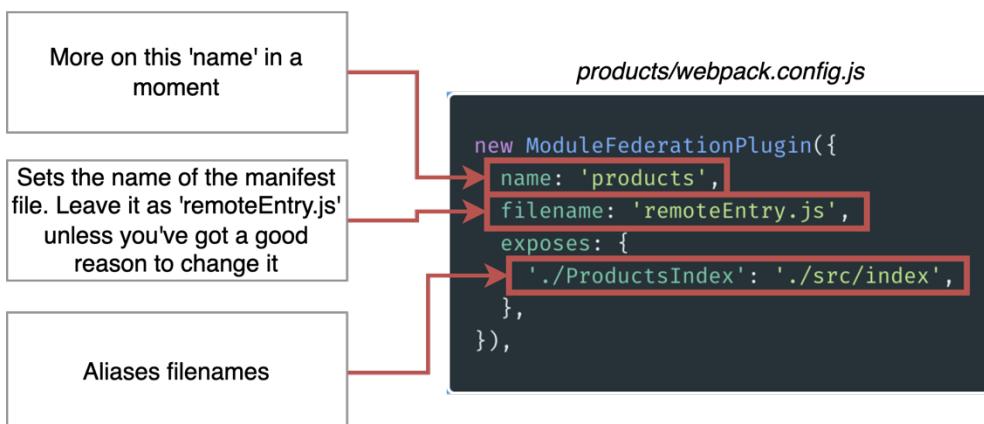
### 3.5. Hiểu hơn về các tùy chọn cấu hình

#### 3.5.1. Đối với Host:



- Không cần phải quá quan tâm về name.
- remotes: Danh sách dự án mà Host có thể tìm để lấy mã.
- remotes.products: tải tệp ở URL được liệt kê nếu mọi thứ trong Host có một import kiểu `import abc from 'products'`. Cái tên products này chính là name của Remote.

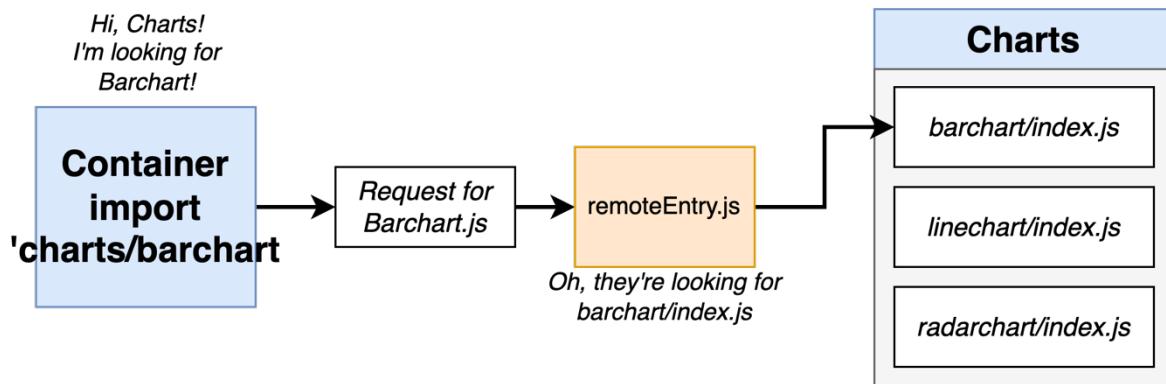
#### 3.5.2. Đối với Remote:



- name: đây là một thuộc tính vô cùng quan trọng vì nó quyết định tên remote mà cần được liệt kê ở Host.

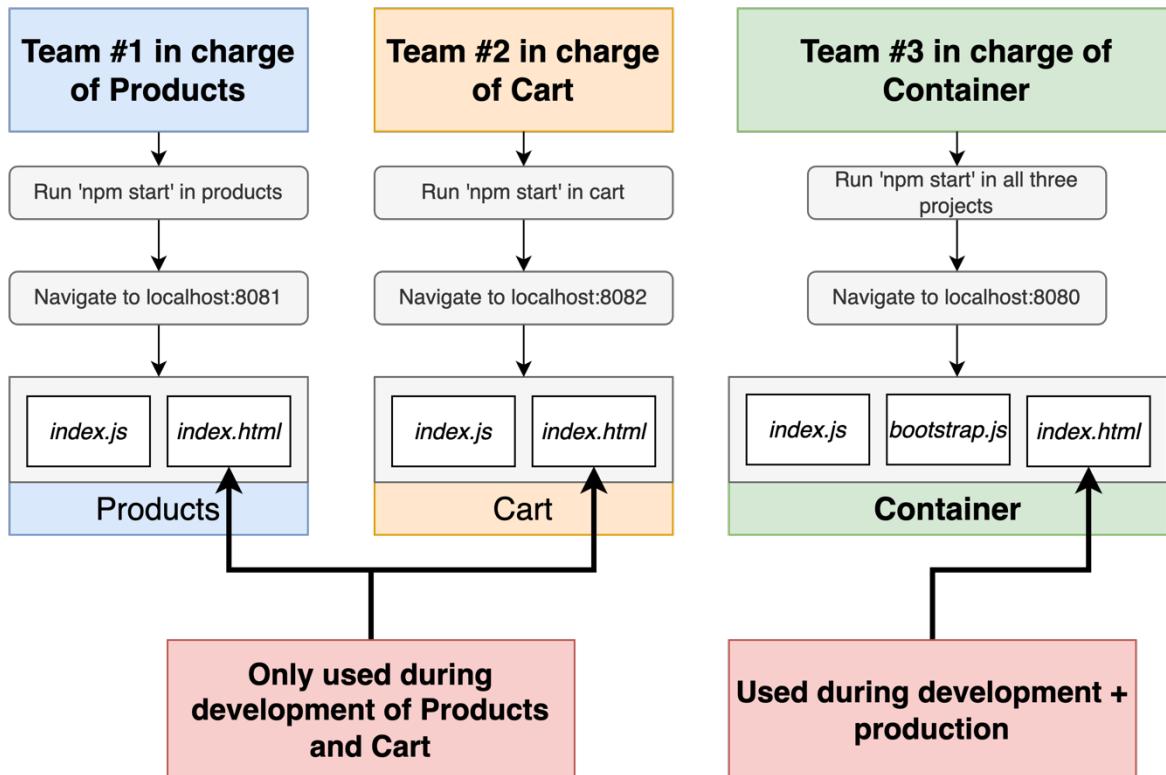
- filename: đây là tên file manifest. Mặc định nên để là remoteEntry.js.  
Đây sẽ là file mà host sẽ cần phải liệt kê và tìm tới.
- exposes: liệt kê danh sách các thành phần cần được lấy ra trong filename.  
Và đặt cho chúng những alias phù hợp.

Tại sao nên có sự phân biệt giữa các alias?

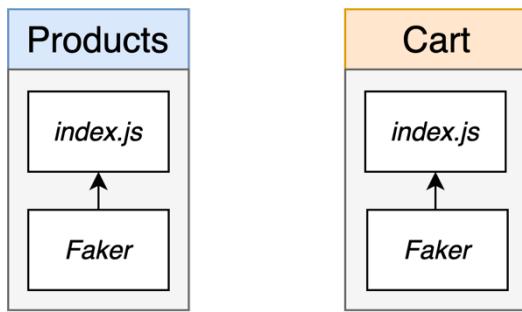


Tạo ra sự phân biệt đó để Host có thể tìm kiếm đúng và chính xác thành phần cần sử dụng từ Remote.

## 4. Quy trình phát triển



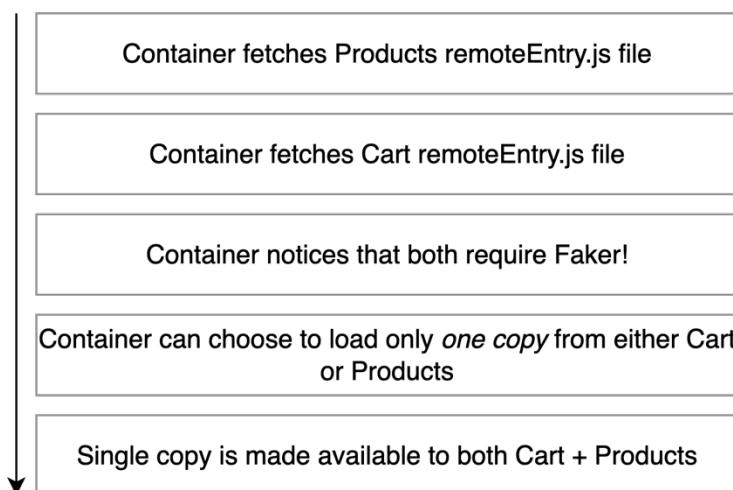
#### 4.1.Chia sẻ các phụ thuộc giữa các ứng dụng (sub-project)



Both Products + Cart need the faker module

Trong quá trình phát triển, giữa các team sẽ cài các phụ thuộc cho app của mình và sẽ xảy ra trường hợp các phụ thuộc giống nhau (về tên và phiên bản). Việc để trình duyệt tải cả các phụ thuộc tuy giống nhau nhưng mỗi phụ thuộc là một lần tải này sẽ làm giảm tốc độ tải trang (hiệu suất) và tốn tài nguyên. Chính vì thế cần Webpack cung cấp một cơ chế để chia sẻ các phụ thuộc này với nhau.

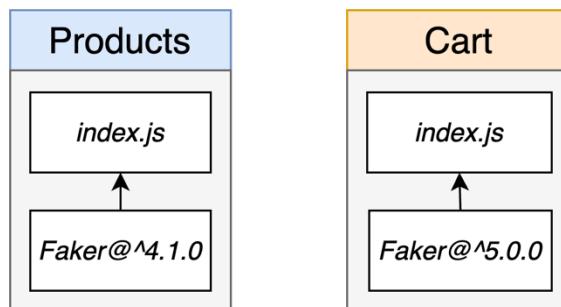
Quy trình chia sẻ module sẽ được diễn ra như sau:



- Host sẽ fetch remoteEntry.js file từ các Remote.
- Host nhận ra các Remote dùng một loại phụ thuộc giống nhau.
- Host có thể chọn để tải một bản sao chép từ phụ thuộc đó của các Remote (hoặc).
- Bản sao chép này sẽ được làm cho có sẵn ở cả các Remote.

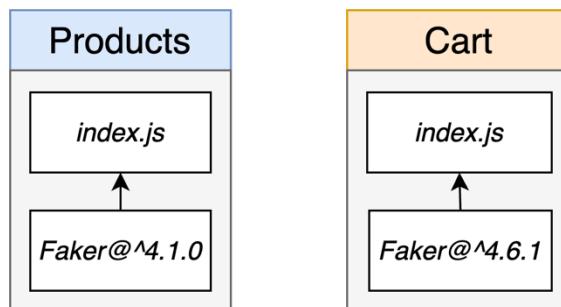
Vậy nếu các phụ thuộc tuy giống nhau về tên, nhưng có sự phân biệt về version thì sao? Nó sẽ có các vấn đề xảy ra như sau:

- Các phiên bản phụ thuộc ở các Remote khác nhau:



Trong trường hợp này, Webpack sẽ tự động load cả 2 module của 2 phụ thuộc có 2 phiên bản khác nhau này.

- Các phiên bản phụ thuộc ở các Remote khác nhau về phiên bản nhỏ:



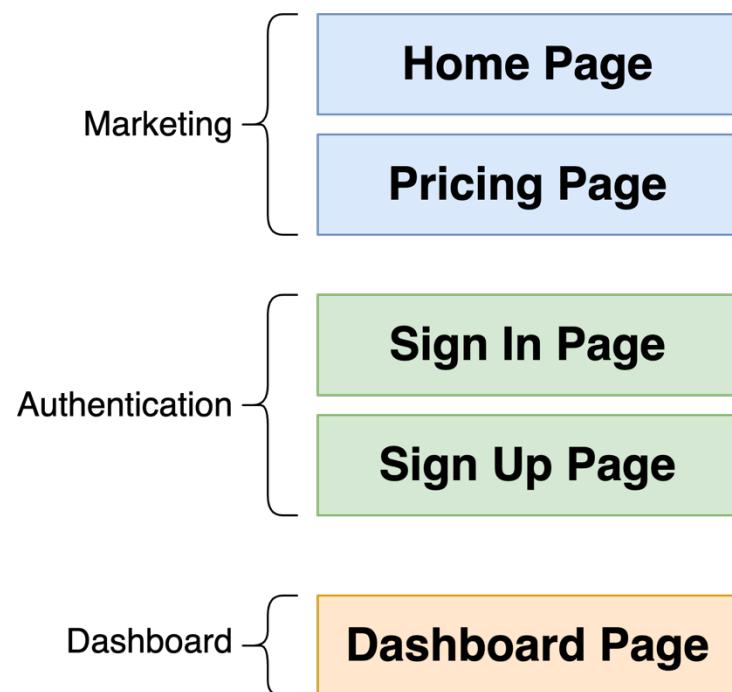
Trong trường hợp này, Webpack vẫn sẽ load một bản copy của bản phụ thuộc đó duy nhất.

## 4.2. Yêu cầu dẫn đến lựa chọn kiến trúc

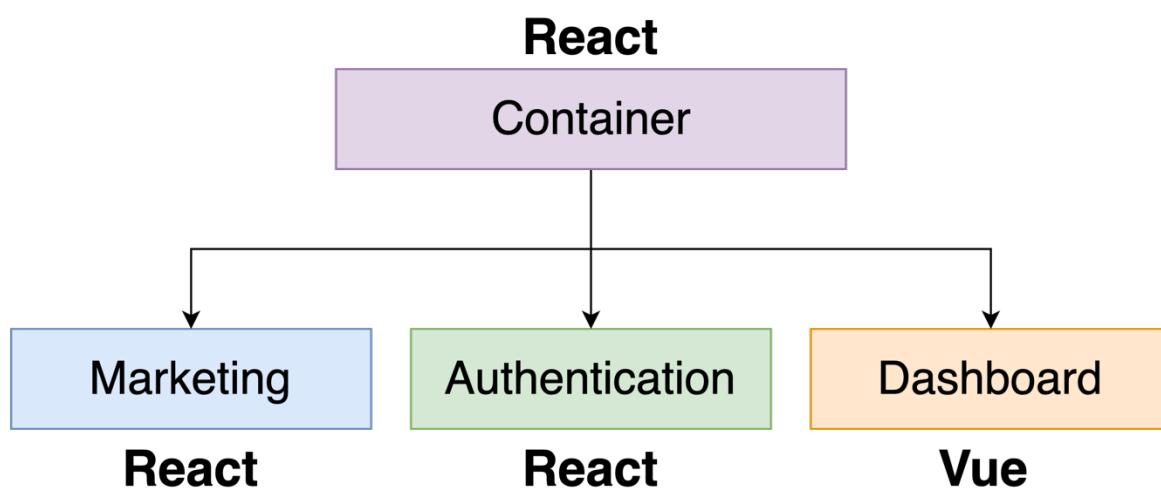
Giả sử ta có giao diện như sau:



Giao diện sẽ được chia thành 3 microfrontend chính:



Một cách tổng quan:



## *Huge Disclaimer*

Some blog posts, articles, videos, etc will tell you to do things differently



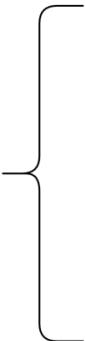
The architecture for this project is determined by its *requirements*



You need to think about the requirements of your app to decide if this architecture will work for you

## Examples

Some blog posts, articles, videos, etc will tell you to do things differently



*"Share state between apps with redux"*

*"The container must be written with webcomponents"*

*"Each microfrontend can be a React component that is directly used by another app"*

*"Only communicate between apps using xyz system"*

#### 4.2.1. Không có sự liên kết giữa các dự án con

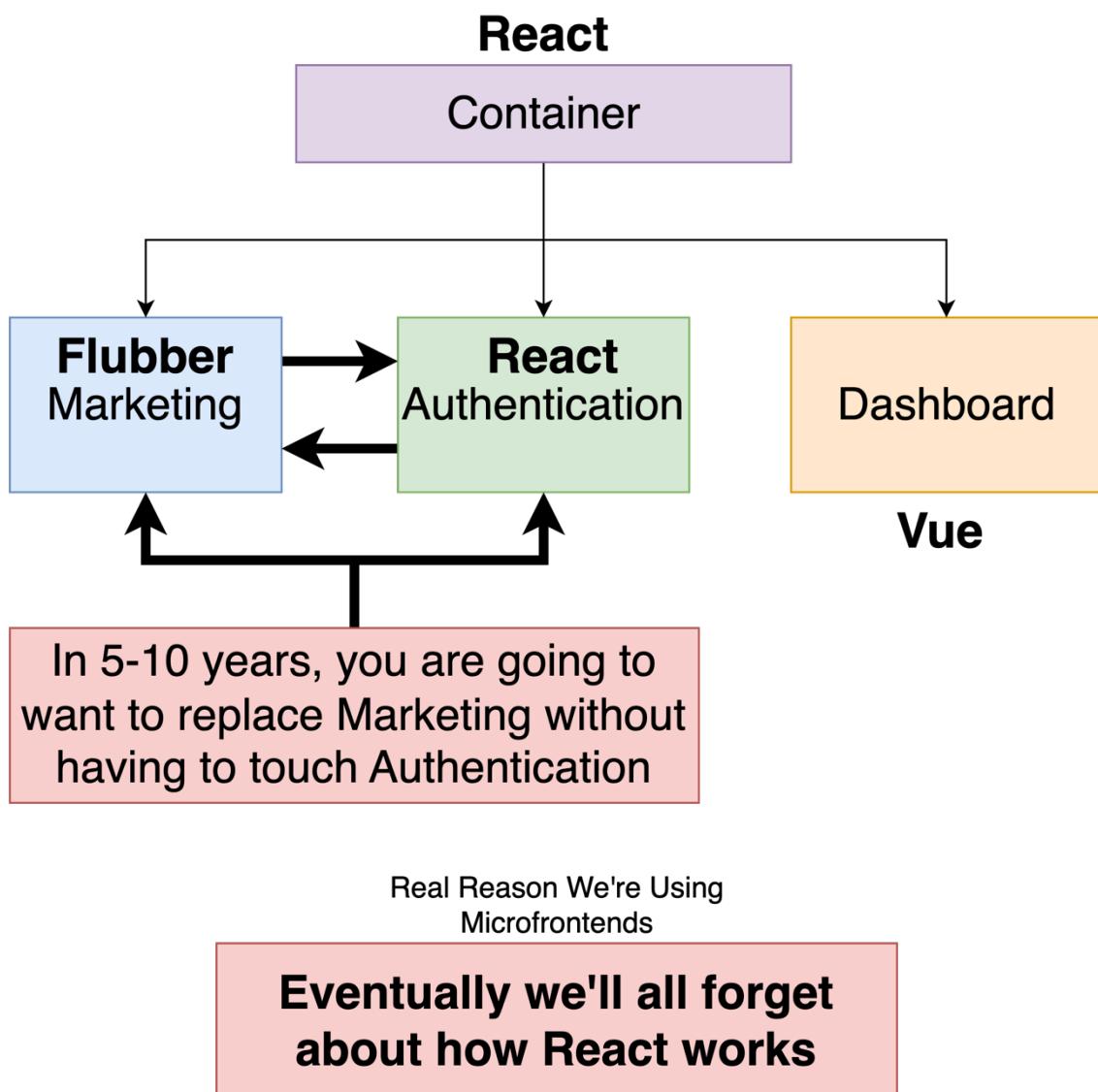
##### Inflexible Requirement #1

Zero coupling between child projects

No importing of functions/objects/classes/etc

No shared state

Shared libraries through MF is ok



#### 4.2.2. Gần như không có sự kết hợp giữa container và các ứng dụng con

##### Inflexible Requirement #2

Near-zero coupling between container and child apps

Container shouldn't assume that a child is using a particular framework

Any necessary communication done with callbacks or simple events

#### 4.2.3. CSS từ một dự án không nên ảnh hưởng đến dự án khác

#### 4.2.4. Kiểm soát phiên bản (monorepo so với separate) không nên có bất kỳ tác động nào đến toàn bộ dự án

##### Inflexible Requirement #4

Version control (monorepo vs separate) shouldn't have any impact on the overall project

Some people want to use monorepos

Some people want to keep everything in a separate repo

#### 4.2.5. Container phải có khả năng quyết định luôn sử dụng phiên bản mới nhất của microfrontend hoặc chỉ định một phiên bản cụ thể

##### Inflexible Requirement #5

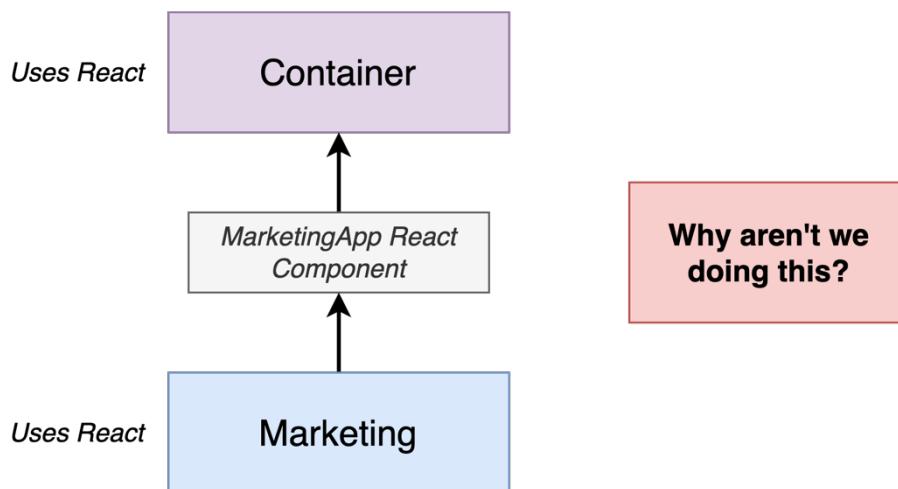
Container should be able to decide to always use the latest version of a microfrontend **or** specify a specific version

(1) Container will always use the latest version of a child app (doesn't require a redeploy of container)

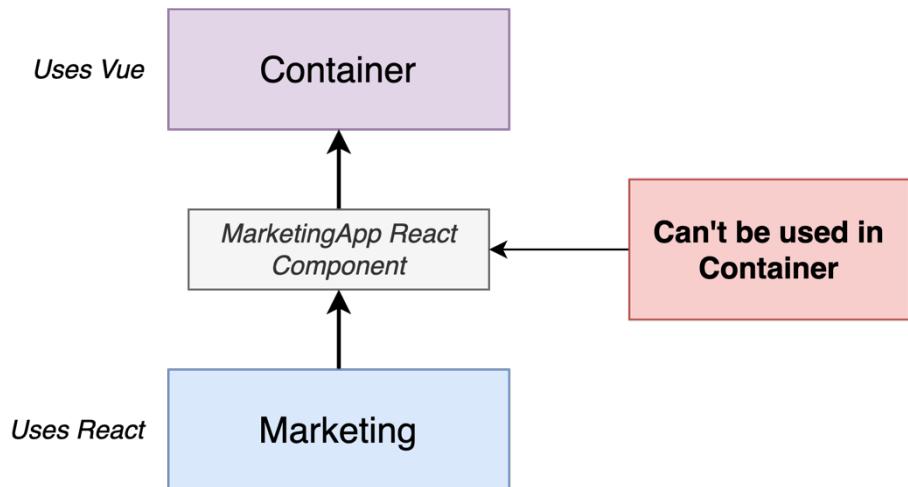
(2) Container can specify exactly what version of a child it wants to use (requires a redeploy to change)

#### 4.3. Tại sao phải sử dụng mount function để render UI?

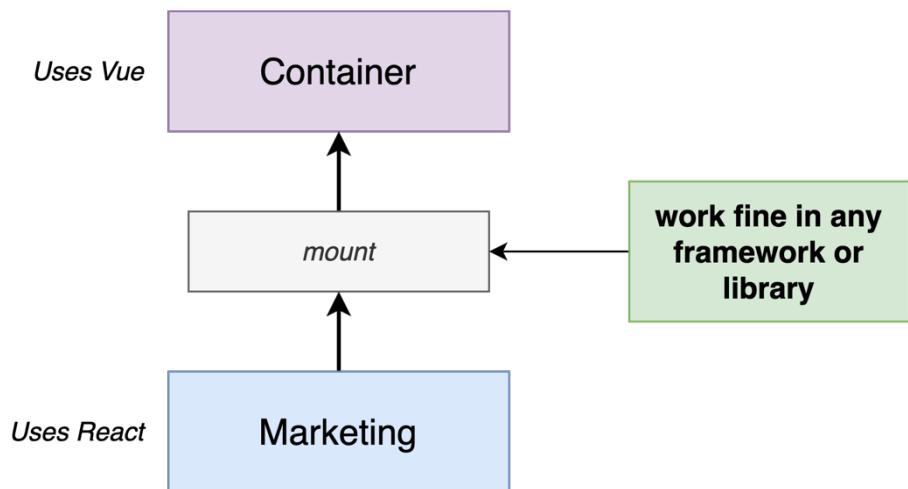
Giả sử ta Container sử dụng React và microfrontend Marketing cũng sử dụng React:



Lúc này, có sự tương thích giữa 2 thư viện (framework) với nhau, cho nên ta có thể expose microfrontend Marketing như một React Component. Nhưng điều này đã vi phạm với yêu cầu lựa chọn kiến trúc #2, nghĩa là sau này nếu như hoặc Marketing đổi framework, hoặc Container đổi framework, sẽ dẫn đến việc chỉnh sửa source code rất nhiều (xung đột với mục đích chọn kiến trúc) vì không có sự tương thích giữa các framework.



Chính vì thế, thay vì ta expose microfrontend như những component, thì ta sẽ expose nó như những mount function để có thể được sử dụng ở bất cứ đâu và tương thích với bất kỳ framework nào.



Mount function sẽ chọn phần tử (HTML DOM) thông qua id/class ở trang web và render lên màn hình thông qua logic được định sẵn trong nó.

## 5. Lợi ích của Microfrontend trong phát triển ứng dụng

Chia khối đơn thành các mô-đun được ghép nối lỏng lẻo là một nhiệm vụ phức tạp và chỉ có ý nghĩa khi các giao diện vi mô giải quyết được các vấn đề mà chúng được cho là giải quyết. Nếu bạn không gặp phải những vấn đề này, thì việc trải qua rắc rối khi áp dụng phương pháp giao diện vi mô là không có ý nghĩa gì.

Sau đây là những lợi ích mà phương pháp giao diện vi mô có thể mang lại:

## **5.1.Triển khai nhanh hơn và quản lý bản phát hành tốt hơn**

Trái ngược với giao diện đơn khôi, trong đó một nhóm duy nhất chịu trách nhiệm triển khai tất cả các chức năng và tính năng mới, giao diện vi mô cho phép các nhóm nhỏ hơn, tự chủ làm việc đồng thời để triển khai nhiều chức năng và tính năng khác nhau. Điều này giúp giảm đáng kể thời gian phát triển và đẩy nhanh quá trình phát hành.

## **5.2.Nhiều nhóm có trách nhiệm khác nhau**

Mỗi nhóm chịu trách nhiệm về giao diện vi mô của mình, là giao diện độc lập. Họ có thể xây dựng, thử nghiệm, triển khai và cập nhật giao diện này một cách độc lập. Với kiến trúc khôi đơn, nếu Nhóm A đã sẵn sàng phát hành và Nhóm B cần thời gian để chuẩn bị, Nhóm A phải đợi. Rào cản này đã được loại bỏ trong phương pháp giao diện vi mô, vì mỗi nhóm có quyền tự do làm những gì họ chọn và không phụ thuộc vào người khác.

## **5.3.Tự do công nghệ**

Mỗi micro frontend có thể được triển khai trên các ngăn xếp công nghệ khác nhau vì chúng là các phần mềm độc lập. Do đó, các nhóm làm việc trên chúng có thể tự do lựa chọn ngăn xếp công nghệ của riêng mình, dựa trên chuyên môn và kinh nghiệm của họ.

## **5.4.Dễ dàng mở rộng quy mô**

Micro frontend còn mang đến một lợi thế khác cho các nhà phát triển. Vì mỗi tính năng có thể được mở rộng quy mô độc lập, nên toàn bộ quy trình trở nên hiệu quả hơn về mặt chi phí và thời gian so với monolith.

## **5.5.Triển khai liên tục**

Bằng cách chia nhỏ frontend monolith thành các thành phần nhỏ hơn, độc lập, micro frontend cho phép cập nhật gia tăng mà không ảnh hưởng đến toàn bộ ứng dụng. Chúng cũng giúp dễ dàng khôi phục một thành phần duy nhất về phiên bản trước đó, có thể cải thiện sự cộng tác của nhóm bằng cách giảm tình trạng tắc nghẽn và tăng khả năng mở rộng quy mô, cùng nhiều lợi thế khác.

# **6. Ứng dụng Microfrontend trong lĩnh vực đặt phòng khách sạn**

## **6.1.Các module microfrontend chính:**

### **6.1.1. Team Khám phá**

- Module: Explore
- Chức năng:

- Gợi ý khách sạn dựa trên vị trí, xu hướng du lịch hoặc sở thích của người dùng
- Cung cấp các địa điểm du lịch, điểm đến phố biển/thịnh hành
- Cung cấp các loại chỗ nghỉ khác nhau (Khách sạn, căn hộ, resort,...)

#### **6.1.2. Team Quản lý đặt phòng**

- Module: Booking
- Chức năng:
  - Tìm kiếm khách sạn
  - Hiển thị danh sách khách sạn phù hợp (với bộ lọc)
  - Chi tiết phòng và giá cả
  - Chọn ngày và số lượng phòng, khách
  - Xác nhận đặt phòng

#### **6.1.3. Team Thanh toán**

- Module: Payment
- Chức năng:
  - Xác nhận thông tin thanh toán
  - Lựa chọn phương thức thanh toán (thẻ tín dụng, ví điện tử,...)
  - Xử lý giao dịch thanh toán
  - Hiển thị trạng thái giao dịch (thành công/thất bại)

#### **6.1.4. Team Quản lý người dùng**

- Module: User Profile
- Chức năng:
  - Đăng ký/Đăng nhập
  - Quản lý thông tin tài khoản
  - Hiển thị và quản lý lịch sử đặt phòng
  - Cập nhật thông tin cá nhân (email, số điện thoại,...)

#### **6.1.5. Team Đánh giá và Xếp hạng**

- Module: Review
- Chức năng:
  - Hiển thị đánh giá khách sạn từ người dùng
  - Cho phép người dùng thêm đánh giá và xếp hạng sau khi hoàn tất chuyến đi
  - Tính năng lọc đánh giá theo mức xếp hạng

### **6.1.6. Team Marketing**

- Module: Promotion
- Chức năng:
  - o Hiển thị các chương trình khuyến mãi
  - o Cung cấp các mã giảm giá và ưu đãi đặc biệt

## 6.2. Một số ví dụ trùu tượng hoá các microfrontend trên trang

### 6.2.1. Trang chủ:

The screenshot shows the main landing page of Booking.com. At the top, there's a navigation bar with the Booking.com logo, currency selection (VND), a user profile placeholder, and two buttons: "Đăng ký" (Sign up) and "Đăng nhập" (Log in). A red box highlights these buttons.

The main search area has a green border. It contains three input fields: "Bạn muốn đến đâu?", "Ngày nhận ph... - Ngày trả ph...", and "2 người lớn - 0 trẻ em - 1 phòng". To the right of these is a blue "Tim" (Search) button.

A yellow box highlights the search input fields and the search button.

Below the search area, there's a section titled "Ưu đãi" (Promotions) with a purple border. It features a heading "Vui là chính, không cần dài", a small text about savings, and a "Tim Ưu Đãi Cuối Năm" (Search Promotions) button next to a photo of two people in a boat.

A blue box highlights the "Ưu đãi" section.

Underneath is a "Promotion" section with a blue border. It has a heading "Tìm theo loại chỗ nghỉ" and four categories with images: Khách sạn (Hotel), Căn hộ (Apartment), Các resort (Resort), and Các biệt thự (Villa). Below each category is a caption: Khách sạn, Căn hộ, Các resort, and Các biệt thự.

An orange box highlights the "Promotion" section.

At the bottom, there's a "Explore" section with an orange border, featuring three cards: TP. Hồ Chí Minh (Saigon) showing city skyscrapers, Vũng Tàu showing a harbor with boats, Đà Nẵng showing a beach and coastline, Hà Nội showing the Tortoise Tower, and Đà Lạt showing a panoramic view of the city.

A yellow box highlights the "Explore" section.

## 6.2.2. Trang chi tiết

**User Profile**

Tuy  
232

Khách lưu trú ở đây thích  
"Khách sạn rất đẹp, si  
Phòng ốc rộng rãi, ti  
Nhân viên nhiệt tình, f  
này."

T Truong VIệt Ni

Vị trí tuyệt vời!

Gà Lai Gò Vấp Bến Tà  
HƯỚNG ĐI: QUỐC LỘ 1  
QUỐC LỘ 101

Hiển thị trên bản  
Google

Đặt ngay

**Điểm nổi bật của chỗ nghỉ**

Hoàn hảo cho kỳ nghỉ 2 đêm!  
Nằm ngay trung tâm TP. Hồ  
khách sạn này có điểm vị  
Có bãi đậu xe riêng miễn phí  
này.

**Đánh giá**

Đánh giá của khách về Apina Saigon - Truong Dinh

9,1 Tuyệt hảo Chúng tôi cố gắng mang đến 100% đánh giá thật! [Viết đánh giá](#)

**Hạng mục:**

Nhan vien phuc vu ↑	9,5	Tien nghi ↑	9,0	Sach se ↑	9,1
Thoi mai	9,0	Đang giá tiền ↑	9,1	Địa điểm ↑	9,6
WiFi miễn phí ↑	9,5	Điểm cao ở TP. Hồ Chí Minh			

**Bộ lọc**

Khách đánh giá Điểm đánh giá Ngôn ngữ Thời gian trong năm

Tất cả (233) Tất cả (233) Tất cả (233) Tất cả (233)

Chọn chủ đề để đọc đánh giá:  
+ Vị trí + Phòng + Sạch sẽ + Bữa sáng + Giường + Hiển thị thêm

**Đánh giá của khách**

Sắp xếp đánh giá theo: Phù hợp nhất

**Xuất sắc**

P Phú VIỆT NAM  
Phòng Tiêu Chuẩn Giường Đôi  
1 đêm - tháng 8/2024  
Cập nhật

Danh giá hàng đầu Ngày đánh giá: ngày 29 tháng 8 năm 2024  
Giá thành hơi cao so với các khách sạn khác, nhưng nhân viên thân thiện thoải mái, phòng sạch sẽ  
Hữu ích Không hữu ích

**Một ks đáng nhớ**

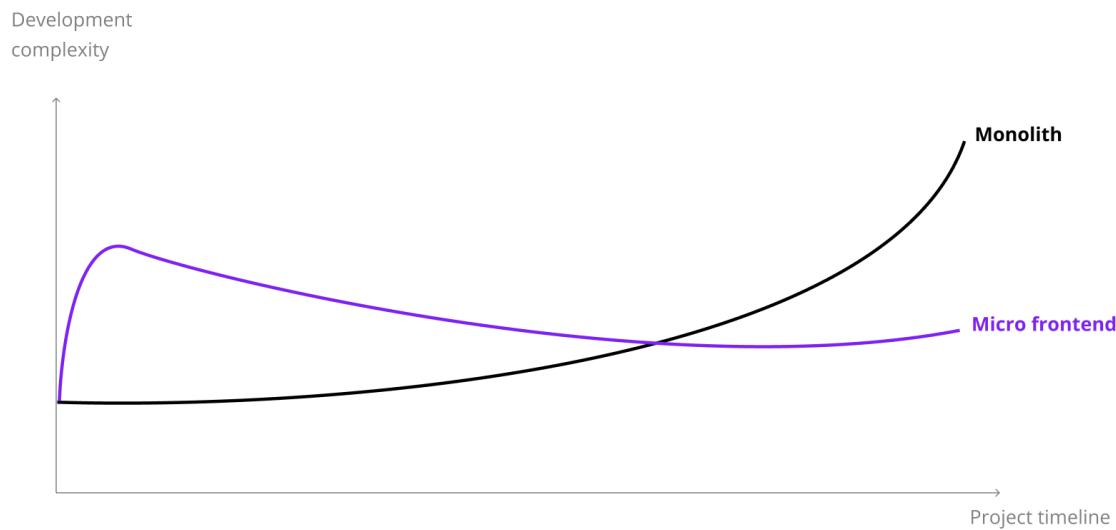
T Truong VIỆT NAM  
Phòng Deluxe Giá đình  
1 đêm - tháng 10/2024  
Cập nhật

Ngày đánh giá: ngày 18 tháng 10 năm 2024  
Khách sạn rất đẹp, sạch sẽ. Phòng ốc rộng rãi, tiện nghi. Nhân viên nhiệt tình. Rất thích ks này.  
Không có

**Review**

## 7. Thách thức khi áp dụng Microfrontend

Micro frontend có nhiều lợi ích, vì vậy, thật dễ dàng để cho rằng chúng luôn là giải pháp tốt nhất cho kiến trúc frontend của bạn. Trên thực tế, điều này không đúng. Phương pháp micro frontend không có khả năng giải quyết tất cả các vấn đề của bạn. Do đó, khi quyết định có nên áp dụng phương pháp micro frontend hay không, điều cần thiết là phải giải quyết các thách thức của nó trước.



### 7.1. Kích thước tải trọng

Micro frontend trùng lắp các phụ thuộc và các khung và thư viện bô, làm tăng kích thước tải trọng của ứng dụng web. Mặc dù yếu tố này là một thách thức đáng kể, nhưng trang web trên micro frontend sẽ tải xuống nhanh hơn so với trên kiến trúc monolith.

Một số giải pháp cục bộ bao gồm việc sử dụng tài nguyên cẩn thận, lựa chọn phụ thuộc tỉ mỉ và tách biệt cẩn thận các trang ít khi sử dụng.

### 7.2. Giao tiếp giữa các trang

Giao tiếp giữa các micro frontend rất khó triển khai và duy trì nhưng có những trường hợp các trang riêng biệt cần giao tiếp ở mức tối thiểu với nhau. Một phần của ứng dụng cần được máy chủ hoặc các micro frontend khác thông báo về tương tác của người dùng và cần thay đổi, làm mới hoặc kích hoạt hành động.

### 7.3. Sự khác biệt về thiết kế

Nếu các nhóm riêng biệt làm việc trên mỗi micro frontend, họ có thể không thấy được toàn cảnh. Do đó, các trang của trang web có thể trông giống như được tạo thành từ nhiều bản vá, không nhất quán về phong cách và UX/UI.

Ngoài ra, còn có nhiều cách để giải quyết vấn đề này – tạo các thành phần chung, hướng dẫn về phong cách mà tất cả các nhóm có thể tham khảo hoặc chỉ cần giao tiếp với nhau.

#### **7.4. Độ phức tạp về mặt vận hành**

Cuối cùng, việc áp dụng giao diện người dùng vi mô gây ra độ phức tạp về mặt vận hành vì cách tiếp cận kiến trúc giao diện người dùng như vậy đòi hỏi nhiều kho lưu trữ, công cụ, nhiều đường ống xây dựng/triển khai và cơ sở hạ tầng phức tạp hơn để tất cả có thể hoạt động cùng nhau.

### **8. Thực tiễn và ví dụ thành công**

#### **8.1. Các công ty đã áp dụng Microfrontend**

- Âm nhạc & Phương tiện: Spotify sử dụng micro frontend cho một nền tảng linh hoạt, theo mô-đun, cho phép phát triển nhanh nhẹn và cung cấp tính năng nhanh chóng.
- Bán lẻ: IKEA và Zalando khai thác micro frontend để mô-đun hóa nền tảng trực tuyến của họ, đảm bảo khả năng bảo trì mã và phát hành tính năng nhanh hơn.
- Tài chính: American Express và PayPal sử dụng micro frontend để nâng cao khả năng mở rộng và khả năng bảo trì của các ứng dụng web của họ.
- Thực phẩm & Đồ uống: Starbucks áp dụng micro frontend để hợp lý hóa nền tảng đặt hàng trực tuyến của mình, đảm bảo trải nghiệm người dùng mượt mà trong bối cảnh cơ sở mã ngày càng phát triển.
- Công nghệ: Upwork, HelloFresh và LambdaTest thúc đẩy kiến trúc web có thể mở rộng và bảo trì với micro frontend, thúc đẩy tính tự chủ của nhóm và chu kỳ phát triển nhanh hơn.

#### **8.2. Nghiên cứu điển hình về ứng dụng đặt phòng khách sạn (Case Study)**

Ví dụ về một ứng dụng đặt phòng khách sạn: Một công ty khởi nghiệp trong lĩnh vực du lịch đã áp dụng kiến trúc Microfrontend cho ứng dụng đặt phòng khách sạn của mình. Dưới đây là một số điểm nổi bật trong quy trình phát triển:

- Chia thành các microfrontend: Công ty này đã chia ứng dụng thành các microfrontend cho các tính năng chính như tìm kiếm, đặt phòng, thanh toán, và đánh giá. Mỗi nhóm phát triển phụ trách một microfrontend cụ thể, giúp tăng tốc độ phát triển và giảm thiểu sự phụ thuộc lẫn nhau.
- Sử dụng API để giao tiếp: Mỗi microfrontend được thiết kế để giao tiếp qua API, cho phép chúng dễ dàng tương tác và chia sẻ dữ liệu, chẳng hạn như thông tin về phòng đã đặt hoặc đánh giá của người dùng.

- Triển khai độc lập: Khi cần cập nhật hoặc sửa lỗi, các nhóm có thể triển khai các microfrontend mà không cần dừng toàn bộ ứng dụng, giúp tiết kiệm thời gian và đảm bảo tính liên tục trong dịch vụ.
- Cải thiện trải nghiệm người dùng: Kết quả là, ứng dụng đã cải thiện đáng kể về tốc độ và tính năng, giúp người dùng dễ dàng tìm kiếm và đặt phòng hơn. Đánh giá từ người dùng cũng tăng lên do trải nghiệm mượt mà và khả năng truy cập nhanh chóng đến các tính năng quan trọng.

## 9. Kết luận

### 9.1. Tóm tắt các điểm chính

Trong báo cáo này, chúng ta đã khám phá kiến trúc Microfrontend và cách áp dụng nó vào phát triển ứng dụng đặt phòng khách sạn. Microfrontend mang lại nhiều lợi ích như tính linh hoạt, khả năng mở rộng, quản lý đội ngũ phát triển, và tối ưu hóa trải nghiệm người dùng. Việc chia nhỏ ứng dụng thành các microfrontend giúp các nhóm phát triển hoạt động độc lập, cho phép họ nhanh chóng triển khai và cập nhật các tính năng mà không ảnh hưởng đến toàn bộ hệ thống.

Chúng ta cũng đã xem xét các yếu tố cần cân nhắc khi áp dụng Microfrontend, từ việc xác định các tính năng cho đến cách tổ chức nhóm phát triển và quản lý trạng thái. Các ví dụ thành công từ các công ty như Zalando và Spotify chứng minh tính hiệu quả của kiến trúc này trong việc phát triển các ứng dụng lớn và phức tạp.

### 9.2. Đề xuất và hướng phát triển tương lai

Dựa trên những lợi ích và thách thức đã được đề cập, có một số đề xuất cho các công ty và nhà phát triển khi áp dụng Microfrontend trong lĩnh vực đặt phòng khách sạn:

- Đào tạo và nâng cao kỹ năng: Để thành công trong việc áp dụng Microfrontend, các nhóm phát triển cần được đào tạo về các công nghệ và quy trình mới liên quan đến kiến trúc này. Việc nâng cao kỹ năng lập trình, kiểm thử, và quản lý trạng thái sẽ giúp cải thiện hiệu suất làm việc.
- Thiết lập quy trình kiểm tra rõ ràng: Cần có quy trình kiểm tra mạnh mẽ để đảm bảo rằng mọi microfrontend đều hoạt động ổn định và tương thích với nhau. Sử dụng các công cụ kiểm thử tự động có thể giúp phát hiện lỗi nhanh chóng và giảm thiểu rủi ro trong quá trình triển khai.
- Theo dõi và tối ưu hóa hiệu suất: Việc theo dõi hiệu suất của từng microfrontend là rất quan trọng để đảm bảo rằng ứng dụng luôn hoạt

động tron tru. Sử dụng các công cụ giám sát có thể giúp phát hiện và khắc phục các vấn đề kịp thời.

- Khám phá các công nghệ mới: Ngành công nghệ phần mềm liên tục phát triển, và việc áp dụng các công nghệ mới như serverless, containerization, hoặc các dịch vụ đám mây có thể giúp tối ưu hóa việc triển khai và quản lý các microfrontend.

## 10. Kế hoạch thực hiện

Giai đoạn	Tuần	Ngày	Công việc	Thành viên thực hiện
Lên kế hoạch	1, 2, 3, 4	30/09 – 20/10	<ul style="list-style-type: none"> <li>- Tìm hiểu các nội dung liên quan đến đề tài</li> <li>- Lập bảng kế hoạch sắp tới</li> <li>- Chuẩn bị dàn ý nội dung</li> </ul>	Kiệt, Mai
Đặc tả yêu cầu	5	21/10 – 26/10	<ul style="list-style-type: none"> <li>- Tổng hợp chức năng và use case</li> <li>- Xây dựng các sơ đồ use case và đặc tả sơ đồ use case</li> </ul>	Mai
Thiết kế	6	28/10 - 03/11	- Thiết kế giao diện	Mai
Xây dựng Backend	6, 7	28/10 – 10/11	- Xây dựng Backend cho toàn bộ ứng dụng	Kiệt
Xây dựng ứng dụng	8	11/11 – 17/11	Xây dựng các microfrontends: <ul style="list-style-type: none"> <li>- Explore</li> <li>- Promotion</li> </ul>	Mai
	9	18/11 – 25/11	Xây dựng các microfrontends: <ul style="list-style-type: none"> <li>- User Profile</li> </ul>	Mai
	10, 11	26/11 – 08/12	Xây dựng các microfrontends: <ul style="list-style-type: none"> <li>- Booking</li> <li>- Payment</li> </ul>	Kiệt

			- Review	
Triển khai ứng dụng	12	09/12 – 13/12	- Testing	Mai
	12	14/12 – 15/12	- Triển khai ứng dụng lên AWS S3 và AWS CloudFront	Kiệt
Báo cáo	13	16/12 – 23/12	- Viết báo cáo và hoàn thiện, chỉnh sửa (nếu có) - Chuẩn bị Slide thuyết trình	Mai, Kiệt

## 11. Tài liệu tham khảo

<https://www.udemy.com/course/microfrontend-course>

<https://euristiq.com/micro-frontend-architecture/>

<https://dev.to/buildwebcrumbs/companies-already-using-micro-frontends-and-why-o37>

<https://en.wikipedia.org/wiki/Webpack>

<https://levelup.gitconnected.com/micro-frontend-architecture-794442e9b325>

<https://blueprint.the-tractor.store/>

<https://speakerdeck.com/naltatis/micro-frontends-building-a-modern-webapp-with-multiple-teams>