Bài tập tuần 7

Thành viên:

Lê Đình Hoàng – 17021254
Nguyễn Đình Nhật Minh – 17021298

I. Micro Frontend là gì?

- Trong một hệ thống sử dụng web app, có hai bộ phận tối cơ bản cấu thành chính là frontend và backend, mỗi bộ phận này thường sẽ được phát triển bởi các đội riêng biệt, đặc biệt với các hệ thống lớn.
- Đối với hệ thống xây dựng theo kiểu monolith, tất cả các chức năng hiển thị UI để người dùng tương tác đều nằm trong một module frontend và module backend của nó.
- Khi chúng ta sử dụng Microservices để xây dựng hệ thống kiểu web app, từng chức năng riêng biệt được chia ra theo tiêu chí giảm liên quan giữa module và tăng liên kết giữa các chức năng trong một module. Do đó, frontend cũng được tách thành nhiều module frontend của các chức năng tương ứng, và tương tự với backend. Nói cách khác, các chức năng được chia theo công dụng và sự liên quan, và cấu trúc frontend backend cũng được chia theo chiều dọc như vậy. Mỗi chức năng có một module frontend và module backend tách biệt so với module của các chức năng khác, điều này gần giống việc module của các hệ thống khác nhau, điều này giúp cho việc phát triển và triển khai các module chức năng trên trở nên tách biệt và độc lập hơn.
- Từ việc chia module theo chức năng, ta thấy rõ rằng module frontend không còn là một module thống nhất nữa mà nó đã trở thành một tập các module riêng biệt phục vụ frontend của các chức năng riêng biệt. **Các module frontend riêng biệt này được gọi là Micro Frontend.**

II. Tại sao cần Micro Frontend?

- Sự phát triển của Micro frontend là một phần của sự phát triển của Micro services, do đó động cơ phát triển của Micro frontend cũng tương tự với Micro services:

+ Các hệ thống phần mềm càng ngày càng trở nên to lớn về kích thước lẫn số lượng chức năng. Mỗi lần thay đổi một chức năng hay mỗi lần có lỗi xảy ra với một chức năng sẽ gây lỗi với toàn bộ hệ thống. Điều đó bao gồm cả các chức năng frontend cũng cần được chia ra theo chức năng chính. Do đó khi một module lỗi sẽ không gây lỗi toàn bộ frontend.
+ Trong các hệ thống lớn, một đội phát triển frontend duy nhất có thể không thể hoàn thành toàn bộ các chức năng cho module frontend theo đúng tiêu chí đề ra. Vì vậy, chúng ta cần tách frontend thành nhiều module theo chức năng của nó, các module này được thực hiện bởi những đội riêng biệt.
+ Tách thành các Micro frontend khiến các đội phát triển các module khác nhau có thể sử dụng công nghệ và ngôn ngữ khác nhau. Ví dụ một đội có thể dùng AngularJS để làm, đội khác có thể dùng VueJS,... .
+ Khi gỡ lỗi, không cần phải khởi động lại toàn bộ frontend và không ảnh hưởng tới các bộ phận khác.

III. Ưu điểm:

- Hỗ trợ các công nghệ khác nhau
- Các nhóm chức năng chéo tự trị

- Phát triển độc lập, triển khai và quản lý và điều hành
- Khả năng kiểm tra tốt hơn
- Cải thiện khả năng cách ly, giải quyết lỗi
- Khả năng mở rộng cao.
- Khả năng "nhập cuộc" nhanh hơn: Mỗi khi một nhà phát triển tham gia vào nhóm phát triển, họ gần như hiểu ngay lập tức về hệ thống một cách dễ dàng.
- Cải thiện hiệu suất: Application shell tải các micro applications dựa trên lộ trình khi người dùng truy cập ứng dụng web

IV. Nhược điểm:

- Tăng kích thước trọng tải
- Các đoạn mã bị lặp lại
- Các micro-frontend chia sẻ dependency
- Khó đạt được tính nhất quán của trải nghiệm người dùng
- Việc theo dõi và gỡ lỗi các vấn đề trên toàn bộ hệ thống rất phức tạp
- Một số triển khai của Micro-Frontends, đặc biệt là xem xét việc nhúng iFrames, có thể gây ra những thách thức lớn về khả năng tiếp cận

V. Ứng dụng:

a. Nên dùng:

- Cho các hệ thống đủ lớn, có nhiều đội phát triển, mỗi đội đảm nhận một mảng "cắt dọc" của hệ thống theo chức năng của hệ thống. Các lát cắt dọc thường bao gồm từ frontend, backend đến database, và không liên quan nhiều đến các chức năng khác (có thể tách thành các hệ thống con và phát triển riêng biệt).
- Cho các hệ thống có thể được chia thành các hệ thống con mà các hệ thống con có bộ chức năng không liên quan tới nhau nhiều, nhưng các chức năng trong cùng một hệ thống lại liên hệ mật thiết với nhau.
- Do Micro frontend dùng nhiều công nghệ frontend khác nhau nên chúng bắt buộc phải giao tiếp bằng JS DOM, do đó framework bắt buộc phải hỗ trợ JS DOM một cách native.

b. Không nên dùng:

- Khi các hệ thống không đủ lớn, bộ phận frontend chỉ có một đội phát triển duy nhất. Với trường hợp này, xử lý nhiều hệ thống con gây tốn thời gian không cần thiết. Lưu ý rằng bộ phận frontend và backend có thể được phát triển bởi các đội riêng biệt, có thể với frontend nhẹ nhàng chỉ yêu cầu một đội lại có thể cần một bộ các hệ thống backend được phát triển bởi nhiều đội khác nhau, điều này khiến hệ thống vẫn giữ được tính chất Micro servies của nó mặc dù vẫn được áp dụng linh hoạt cho từng module.
- Cho các hệ thống không thể chia chia thành các hệ thống có chức năng tách biệt do sự liên kết và lệ thuộc lẫn nhau quá lớn. Khi này có thể chia theo các mô hình khác ngoài mô hình chia dọc như chia chức năng dùng chung thành một hệt thống,... hoặc không chia nữa giữ nguyên một hệ thống thuần nhất.
- Không sử dụng cho các hệ thống dùng các framework không hỗ trợ native JS DOM, ví dụ như ReactJS.