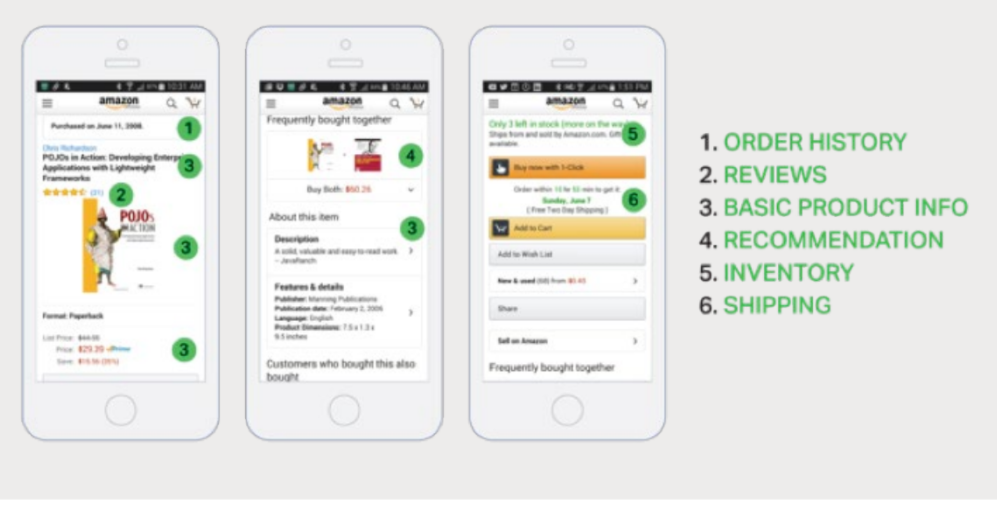
Các lưu ý khi triển khai Microservices Appliation

1. API Gateway

rat nhieu thong tin duoc hien thi tren 1 UI,

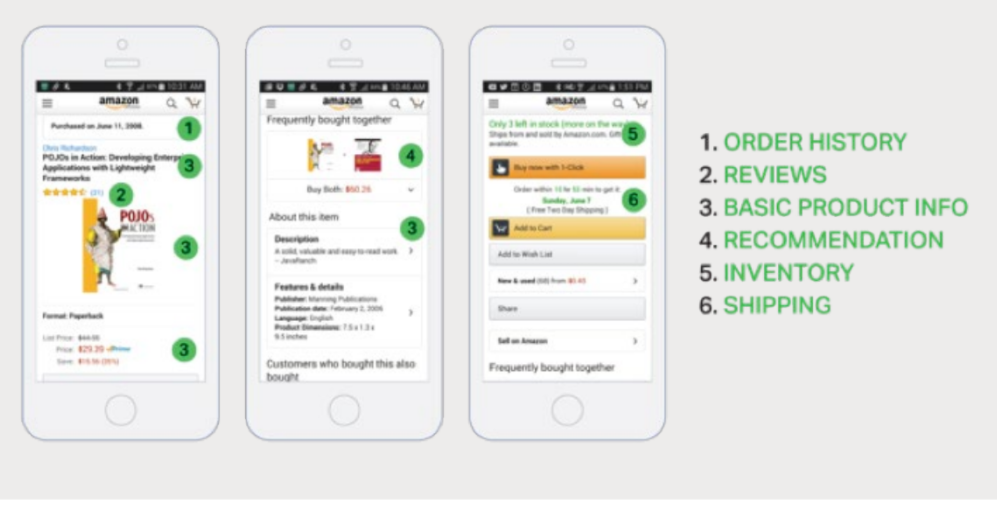
Va doi voi Monolithic thì nó sẽ được query data bằng cách:

GET: api.company.com/productdetails/productId

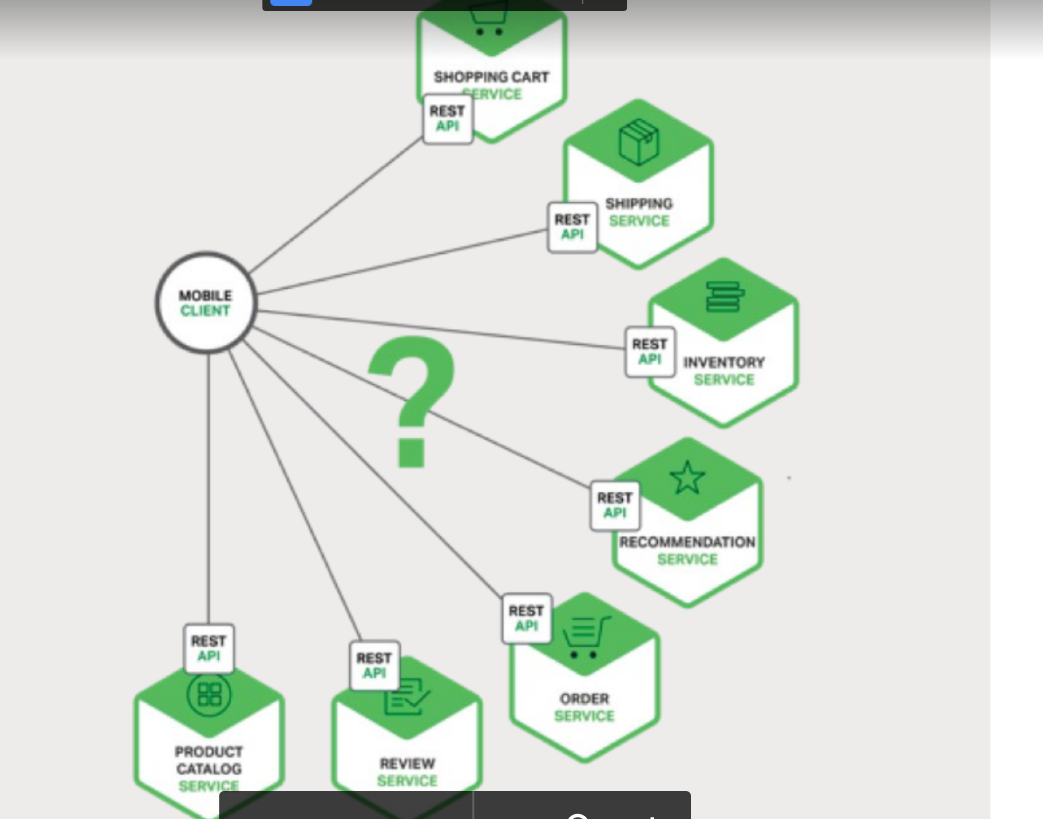
Tất cả thông tin liên quan đến product đó sẽ được query bằng 1 lệnh REST Call duy nhất - sau đó loadbalancer sẽ route cái request đó đến 1 trong các Application instance giống nhau. Application Instance nào nhận được request đó sẽ truy vấn database table và trả kết quả cho client

-------------

Nếu bạn sử dụng microservices architecture, data được hiển thị sẽ được hiển thị trên trang product details như hình dưới, được lấy từ rất nhiều microservice khác nhau



Mobile client sẽ lấy data từ nhiều service nhỏ khác nhau (microservices) rồi hiển thị lên trang chủ



Việc chúng ta cần làm là quyết định cách các mobile client kết nối đến các service này

-------------------------------

Giao tiếp trưc tiếp Client và Microservices

một Client có thể đưa ra yêu cầu trực tiếp đến từng microservices g

Mỗi microservice sẽ có một public endpoint

Vd: [https://serviceName.api.company.name](https://serviceName.api.company.name/)

Mỗi URL sẽ map với loadbalancer nhằm phân phối các gói đến đúng Server(chia tải)

Client gửi yêu cầu trực tiếp đến microservice gây ra:

\*\*\* Lưu ý : để truy xuất được hết tất cả thông tin trên trang product details như hình ở trên, client sẽ phải request hết tất cả các microservice liên quan để có thể lấy được data

\*\*\* Lưu ý : một vấn đề khi client request hết tất cả các microservice liên quan để lấy được data là sự mismatch giữa những thông tin mà khác hàng cần và những APIs nhỏ nhặt mà các microservice cung cấp.

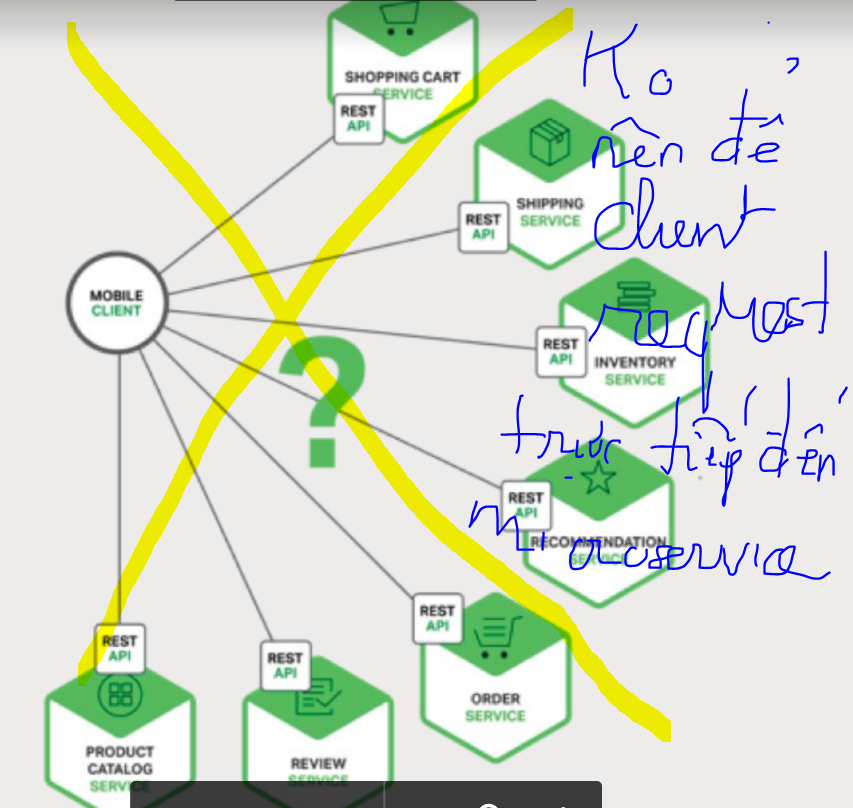
Trong ví dụ page trên, khác hàng phải tạo 7 yêu cầu riêng biệt. Trong các ứng dụng phức tạp hơn như Amazon nó có thể tạo hàng trăm request đến các microservice khác nhau để hiển thị lên trang sản phẩm của họ, Mặc dù một Client có thể thực hiện các request đó qua mạng LAN một cách ổn định, tuy nhiên nếu request đó qua mạng Internet công cộng thì đó sẽ là một vấn đề .Cách tiếp cận này làm cho client code phức tạp hơn

Một vấn đề khác với client là Application Client gọi trực tiếp các microservices một số Application Client có thể sử dụng các giao thức không thân thiện với Website như 1 service sử dụng : Thrift binary RFC, 1 service khác sử dụng: AMQP messaging protocol (không có cái nào là thân thiện với Browser hoạc firewall cả), Application nên sử dụng protocol như là HTTP và WebSocket bên ngoài Firewall

Một nhược điểm khác là nó gây khó khăn cho việc cấu trúc lại các microservice

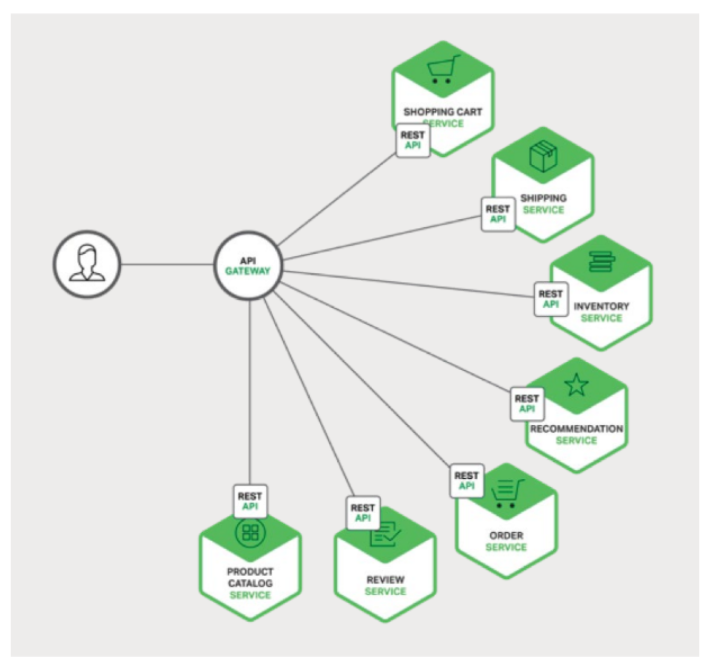
Vd: hợp 2 dịch vụ lại thành 1 dịch vụ, hoặc tách 1 dịch vụ làm 2 dịch vụ, nhưng đối với kiểu client request trực tiếp đến microservice sẽ không thể làm được điều đó.

Vì vậy trong khi code người ta thường không để Client request trực tiếp đến Microservices



---------------------------

Khác phục việc client request trực tiếp đến với Microservice ta có thể sử dụng API Gateway



Api gateway như là một server mà chỉ có 1 entry point (Điểm vào) duy nhất vào hệ thống.

Api gateway encapsulates the “Internal system architecture” và cung cấp 1 API mà phù hợp với từng máy khách, nó có thể có trách nhiệm khác như xác thực, giám sát, cân bằng tải, cache, quản lý các request và xử lý phản hồi tĩnh

Các thuận lợi của API gateway là không cần bàng cãi, tuy nhiên API gateway cũng có các nhược điểm:y

API gateway là một thành phần tính khả dụng cao mà được developed, deployed và quản lý. Cũng có một rủi ro rằng các cổng API sẽ gặp một tình huống nút cổ chai (dữ liệu trafic quá nhiều mà Server API Gateway không xử lý được hết). Các Developer phải cập các API để hiển thị ra (lộ ra) các Microservice’s Endpoint . Điều quan trọng là quá trình cập nhật cổng API càng nhanh càng tốt. Mặt khác, các developed sẽ bị buộc phải chờ xếp hàng để cập nhật

Cách thức triển khai API Gateway: (động lực và sự đánh đổi khi bạn triển khai API)

Động lực (lợi ích) khi sử dụng API Gateway

1. **Hiệu suất và khả năng mở rộng** là rất quan trọng khi xây dụng API GATEWAY . Có nhiều công nghệ có thể sử dụng để xây dụng Scalable API Gateway. treeb JVM bạn có thể sử dụng NIO-based Frameworks như là Netty, Vertx, Sprint Reactor or JBoss Undertow. Phổ biến nhất vẫn là Node.js hoặc NGINX Plus

2. **Sử dụng mô hình Reactive Programming** : Gateway xử lý một số request chỉ bằng cách định tuyến (routing) chúng đến service ở phía BackEnd

phù hợp, nó xử lý các request khác bằng cách gọi nhiều backend services và tổng hợp kết quả lại làm 1 để gửi cho enduser

Vd: page Proudct details, khi khách hàng vào trang, nó sẽ request đến API gateway, sau đó API gateway hiểu nó và chia thành các request nhỏ đến các microservice ở server phía sau. \*\*\* (các yêu cầu đến các Microservices ở phía backend ***độc lập với nhau*** ) \*\*\* , tuy nhiên để giảm thiểu thời gian đáp ứng, các API Gateway sẽ (nên) thực hiện các request tới các Microservices ở phía backend một cách đồng thời..

Tuy nhiên đôi khi cũng có sự phụ thuộc giữa các request đến các Microservices phía backend, ***\*\*\* lưu ý, trước khi API Gateway định tuyến gói request đến các Microservices, nó sẽ (nên) thực hiện xác thực trước \*\*\**** ***Bằng cách gọi một authentication service trước khi nó route request đến backend***.

Vd: về sự phụ thuộc có trong API Gateway

Page hiển thị danh sách wish list (danh sách các sp mong muốn mua của khách), để hiển thị các thông tin lên page này, đầu tiên nó sẽ phải truy suất 1 request đến hồ sơ của khách hàng để xem những item mà trong wish list của khách hàng , rồi lấy id của những item đó truy xuất thông tin chi tiết của sản phẩm để hiển thị lên trên khách hàng (sự thuộc)

vd khác về sự phụ thuộc trong API Gateway là :Netflix Video Grid -

***Để bạn viết code xây dựng các API Gateway kiểu phụ thuộc như vậy - nếu bạn sử phương pháp truyền thống để thiết kế là (traditional asynchronous callback) sẽ khiến bạn gặp tình trạng Callback***

Phương pháp tốt hơn là viết code xây dựng API Gateway kiểu phụ thuộc như vậy là xây dựng mã API Gateway sử dụng phương pháp ***REACTIVE***

Phương pháp REACTIVE là gì: vd: Examples of reactive abstractions include Future in Scala, CompletableFuture in Java 8, and Promise in JavaScript. Ngoài ra còn có Extension (còn được gọi là Rx hoặc ReactiveX), ban đầu được phát triển bởi Microsoft cho nền tảng “.NET”. Netflix đã tạo ra RxJava cho JVM để sử dụng trong API Gateway của họ

Sử dụng phương pháp REACTIVE cho phép bạn viết mã API Gateway đơn giản nhưng hiệu quả hơn

3. Service Invocation or Inter-process communication mechanism (lời mời dịch vụ ??)

Một ứng dụng được xây dựng theo kiểu microservice là một hệ thống phân tán và sử dụng cơ chế giao tiếp giao tiếp giữa các inter-process (liên qkh

có 2 loại giao tiếp giữa các inter-process

a. use an asynchronous, messaging-based mechanism

Một số triển khai sử dụng trình môi giới tin nhắn như là (JMS hoặc AMQP) - Zeromq không có môi giới và các dịch vụ giao tiếp trực tiếp

b. synchronous mechanism (such as HTTP or Thrift)

một hệ thống có thể sử dụng cả đồng bộ và không đồng bộ đồng . It might even use multiple implementations of each style. Do đó, API Gateway sẽ cần hỗ trợ nhiều cơ chế giao tiếp.

4.Service Discovery

Các API Gateway cần biết vi trí (IP address và port) của mỗi microservices mà nó giao tiếp với (or nó gửi request tới) , đôi với các ứng dụng truyền thống (ý là ko phải ứng dụng cloud) v***iệc xác định IP address và port của mỗi microservices là không quá khó. Tuy nhiên đối với các ứng dụng trên Cloud (Cloud-base microservices Applications) việc xác định các IP or Port của các Instance chạy trên cloud là một việc cũng tương đối khó***

các dịch vụ Infrastructure (vd: Message Broker), thường sẽ có một vị trí tĩnh (IP Tĩnh), có thể được chỉ định thông qua các biến OS. Tuy nhiên, để xác định location (IP,port) của những ứng dụng như vậy là không dễ dàng

Application Services có vị trí được chỉ định động, Ngoài ra, tập hợp các Intances của một dịch vụ thay đổi động do tính năng autoscaling and upgrades. Do đó, API Gateway, giống như bất kỳ ứng dụng khách dịch vụ nào khác trong hệ thống, cần sử dụng

Vì vậy, API Gateway, giống như các Client Service

khác trong hệ thống, cần sử dụng cơ chế ***Service Discovery*** : khám phá phía máy chủ và khám phá phía máy khác, C4 sẽ tìm hiểu kỹ hơn về vấn đề này

Hiện tại chỉ cần lưu ý:

Nếu hệ thống sử dụng tính năng khám phá phía Client, thì API Gateway phải có thể truy vấn được service registry (là một loại database lưu tất cả các microservice instances và vị trí (IP,Port) của chúng)

5.Xử lý lỗi một phần

Một vấn đề khác bạn phải giải quyết khi triển khai API Gateway là vấn đề lỗi một phần. Sự cố này phát sinh trong tất cả các hệ thống Distributed systems, bất cứ khi nào một dịch vụ gọi một dịch vụ đang phản hồi chậm hoặc không khả dụng, “The API Gateway should never block indefinitely waiting for a downstream service ???”

Tuy nhiên, cách nó xử lý lỗi phụ thuộc vào từng trường hợp cụ thể và dịch vụ nào bị lỗi

vd: nếu dịch vụ đề xuất không phản hồi trong page product details. API Gateway sẽ trả lại phần còn lại của chi tiết sản phẩm cho khách (vd: chi tiết sản phẩm, các sản phẩm cùng loại, ...) (không có phần đề xuất thôi) lên page product vì chúng vẫn hữu ích cho khách hàng, tuy nhiên nếu service thông tin sản phẩm (service core) không phản hồi nó sẽ trả về lỗi -> xử lý lỗi một phần

Netflix Hystrix là một thư viện cực kỳ hữu ích để viết mã gọi các dịch vụ từ xa. Nếu time out (nghĩa là service không phản hồi) Hystrix có thể gọi các cuộc gọi vượt quá ngưỡng chỉ định. Nó có thể ngắt mạch để user khỏi phải chờ đợi một cách không cần thiết với một dịch vụ không phản hồi.

Nếu tỷ lệ lỗi cho một dịch vụ vượt quá một ngưỡng được chỉ định, Hystrix ngắt bộ ngắt mạch và tất cả các yêu cầu sẽ không thành công ngay lập tức trong một khoảng thời gian nhất định. Hystrix cho phép bạn xác định hành động dự phòng khi yêu cầu không thành công, chẳng hạn như đọc từ bộ nhớ cache hoặc trả về giá trị mặc định. Nếu bạn đang sử dụng JVM, bạn chắc chắn nên cân nhắc sử dụng Hystrix. Và, nếu bạn đang chạy trong môi trường không phải JVM, bạn nên sử dụng một thư viện tương đương