

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**



Lê Tiến Chiến

HỆ THỐNG THỰC HIỆN A/B TEST TRONG QUẢNG CÁO

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công Nghệ Thông Tin

Hà Nội, 2022

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Tiến Chiến

HỆ THỐNG THỰC HIỆN A/B TEST TRONG QUẢNG CÁO

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Công Nghệ Thông Tin

Cán bộ hướng dẫn:

Cán bộ đồng hướng dẫn:

Hà Nội, 2022

Phê duyệt của giảng viên hướng dẫn

Tôi xác nhận rằng dự án đến hiện tại đã sẵn sàng cho hội đồng thẩm định để lấy bằng cử nhân ngành Khoa học Máy tính trường Đại học Công nghệ.

Ký tên:.....

Th.S Hồ Đắc Phương
Cán bộ hướng dẫn dự án
Khoa Công nghệ thông tin
Trường Đại học Công Nghệ
Đại học Quốc gia Hà Nội

Lời mở đầu

Trước tiên, tôi xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất đến thầy giáo, Th.S Hồ Đắc Phương, người đã tận tình chỉ bảo, hướng dẫn, động viên và giúp đỡ tôi trong suốt quá trình làm học phần dự án. Kiến thức và kinh nghiệm nghiên cứu tôi có được từ thầy sẽ rất quan trọng đối với tương lai của tôi.

Tiếp theo, em xin gửi đến quý thầy cô giáo trong ngành Khoa học máy tính, Khoa Công nghệ Thông tin nói riêng và trong Trường Đại học Công nghệ - Đại học Quốc Gia Hà Nội nói chung đã dành tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Các thầy cô không chỉ giảng dạy mà còn định hướng cho em rất nhiều trong quá trình phát triển bản thân và trong cuộc sống. Bên cạnh đó tôi cũng gửi lời cảm ơn đến anh, chị, em, bạn bè đã đồng hành cùng tôi suốt những năm dưới mái trường đại học Công Nghệ.

Tiếp đến, tôi xin được cảm ơn gia đình, đã sinh thành và nuôi dưỡng cũng như định hướng, động viên tôi trong suốt quá trình học tập.

Cuối cùng, báo cáo dự án này không tránh khỏi những thiếu sót. Tôi mong nhận được những nhận xét về những sai sót và ý kiến đóng góp để dự án hoàn thiện hơn.

Tôi xin chân thành cảm ơn!

Hà Nội, ngày tháng năm 2021
Sinh viên

Lê Tiên Chiến

Tóm tắt

Ngày nay có rất nhiều hệ thống được xây dựng dựa trên các mô hình tính toán khác nhau. Với mỗi mô hình tính toán lại có các ưu, nhược điểm khác nhau. Để đánh giá mỗi mô hình tính toán, cần triển khai mỗi mô hình trên hệ thống trong một thời gian dài. Sau khi triển khai xong, các doanh nghiệp tổ chức rất khó để so sánh giữa các mô hình tính toán khác nhau. Bởi vì mỗi mô hình được chạy trong các khoảng thời gian khác nhau và có tập người sử dụng khác nhau trong các thời điểm.

Do đó các doanh nghiệp nảy sinh ra nhu cầu cần có một hệ thống toàn diện để giải quyết bài toán trên. Hệ thống này có thể đánh giá các mô hình tính toán khác nhau một cách chính xác và công bằng. Từ đó doanh nghiệp có thể biết được mô hình tính toán nào tốt hơn mà không cần phải tốn quá nhiều thời gian thử nghiệm trên từng mô hình.

Hệ thống A/B Testing được ra đời để hoàn thành mục đích đó. Khi sử dụng hệ thống, bất cứ người dùng nào cũng có thể khởi tạo các mô hình với các thông số khác nhau để so sánh. Hệ thống A/B Testing cũng có thể tích hợp với các hệ thống khác để chạy thử nghiệm, từ đó thu thập dữ liệu để đánh giá các mô hình.

Từ khoá: A/B Testing, thử nghiệm, đánh giá.

Mục lục

PHÊ DUYỆT CỦA GIẢNG VIÊN HƯỚNG DẪN	i
LỜI MỞ ĐẦU	i
TÓM TẮT	ii
DANH SÁCH HÌNH VẼ	v
DANH SÁCH BẢNG	vi
1 GIỚI THIỆU TỔNG QUAN	1
1.1 A/B Testing	1
1.1.1 A/B Testing là gì	1
1.1.2 Các A/B testing hoạt động	2
1.1.3 Ứng dụng của A/B Testing	3
1.1.4 Quy trình thử nghiệm A / B	4
1.2 Thông số	5
1.2.1 Tỷ lệ chuyển đổi (Conversion Rate)	5
1.2.2 Tỷ lệ click (CTR)	7
1.3 Cấu trúc của thí nghiệm	7
1.3.1 Product	8
1.3.2 Layer	8
1.3.3 Experiment	8
1.3.4 Test Group	9
1.4 Công nghệ thực hiện	9
1.4.1 Golang	9
1.4.2 React	13
1.4.3 Redis	14
2 KIẾN THỨC HỆ THỐNG	19
2.1 Cơ sở lý thuyết của kiến trúc	19
2.1.1 Kiến trúc Onion	19
2.1.2 Kiến trúc mô-đun	21
2.2 Kiến trúc tổng quan hệ thống	22
2.2.1 Hệ thống frontend	22
2.2.2 Hệ thống backend	23
2.2.3 Biểu đồ tuần tự	24
2.3 Mô hình cơ sở dữ liệu	25

2.3.1	Các thực thể	25
2.3.2	Mối quan hệ giữa các thực thể	27
2.3.3	Id generator	28
3	TRIỂN KHAI VÀ THỰC NGHIỆM	29
3.1	Ứng dụng Domain Driven Design	29
3.2	Triển khai chi tiết Use case	29
3.2.1	Use case khởi tạo A/B Test	29
3.2.2	Use case sử dụng A/B Test	30
3.3	Triển khai kiểm thử thích hợp	30
3.3.1	Mô tả Integration test với một service cụ thể	31
3.4	Thực nghiệm	32
4	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	33
4.1	Kết quả đạt được	33
4.2	Hạn chế khi thực hiện đề tài	33
4.3	Hướng phát triển	33

Danh sách hình vẽ

1.1	A/B Testing	2
1.2	Biểu đồ kiểm soát biến	3
1.3	Quy trình thử nghiệm A / B	5
1.4	Conversions	6
1.5	Tổng quan của một A/B Testing	7
1.6	Product	8
1.7	Layer	9
1.8	Experiment	9
1.9	Test Group	10
1.10	Golang	10
1.11	Lịch sử của Golang	11
1.12	Goroutine	12
1.13	Channel	12
1.14	React	13
1.15	Virtual DOM	14
1.16	Redis	14
1.17	Tính năng của Redis	15
1.18	Cấu trúc dữ liệu của Redis	16
1.19	Ứng dụng của Redis	17
2.1	Các lớp của Kiến trúc Onion	20
2.2	Kiến trúc mô-đun	21
2.3	Hệ thống Backend	23
2.4	Biểu đồ tuần tự chung	24
2.5	Biểu đồ tuần tự khởi tạo A/B Test	25

Danh sách bảng

2.1	Hệ thống Trang	22
2.2	Hệ thống đường dẫn	24
2.3	Thuộc tính của Product	25
2.4	Thuộc tính của Layer	26
2.5	Thuộc tính của Experiment	26
2.6	Thuộc tính của Test Group	26

Chương 1

GIỚI THIỆU TỔNG QUAN

1.1 A/B Testing

A/B Testing (còn được biết đến như là bucket testing hay split-run testing) là một phương pháp nghiên cứu trải nghiệm người dùng. Một A/B test bao gồm một thí nghiệm ngẫu nhiên gồm có hai biến thể, A và B. Nó bao gồm ứng dụng của kiểm định giả thuyết thống kê hoặc "kiểm định giả thuyết hai mẫu" như được sử dụng trong lĩnh vực thống kê. A/B testing là một cách để so sánh hai phiên bản của một biến đơn lẻ, thường bằng cách kiểm tra phản ứng của đối tượng đối với biến thể A so với biến thể B và xác định xem biến thể nào trong hai biến thể hiệu quả hơn. Phiên bản hiệu quả hơn sẽ dùng để quảng cáo online. A/B Testing có thể dùng để test mọi thứ. Từ một hình banner cho đến các trang web hoặc các mẫu quảng cáo, email, CTA kêu gọi hành động...

1.1.1 A/B Testing là gì

A/B Testing chính là kỹ thuật chia đối tượng cần kiểm tra thành hai phiên bản A và B để đưa ra được phiên bản mà người dùng quan tâm hơn thông qua cách người dùng tương tác với mỗi phiên bản đó.

A/B Testing là cách viết tắt của một thử nghiệm ngẫu nhiên có đối chứng đơn giản, trong đó hai mẫu (A và B) của một biến vector duy nhất được so sánh. Các giá trị này tương tự nhau ngoại trừ một biến thể có thể ảnh hưởng đến hành vi của người dùng. A/B Testing được coi là hình thức đơn giản nhất của thử nghiệm có đối chứng. Tuy nhiên, bằng cách thêm nhiều biến thể hơn vào thử nghiệm, độ phức tạp của nó sẽ tăng lên.

A/B Testing hữu ích để hiểu mức độ tương tác của người dùng và mức độ hài lòng của các tính năng trực tuyến như một tính năng hoặc sản phẩm mới. Các trang web truyền thông xã hội lớn như LinkedIn, Facebook và Instagram sử dụng A/B Testing để làm cho trải nghiệm người dùng thành công hơn và như một cách để hợp lý hóa dịch vụ của họ.

Ngày nay, các A/B Testing cũng đang được sử dụng để thực hiện các thử nghiệm phức tạp trên các chủ đề như hiệu ứng mạng khi người dùng ngoại tuyến, cách các dịch vụ trực tuyến ảnh hưởng đến hành động của người dùng và cách người dùng ảnh hưởng đến nhau. Nhiều ngành nghề sử dụng dữ liệu từ các bài kiểm tra A / B. Điều này bao gồm các kỹ sư dữ liệu, nhà tiếp thị, nhà thiết kế, kỹ sư phần mềm và doanh nhân. Nhiều vị trí dựa vào dữ liệu từ các A/B Testing, vì chúng cho phép các công ty hiểu được sự tăng trưởng, tăng doanh thu và tối ưu hóa sự hài lòng của khách hàng.

Phiên bản A có thể là phiên bản được sử dụng hiện tại (do đó tạo thành nhóm kiểm soát),

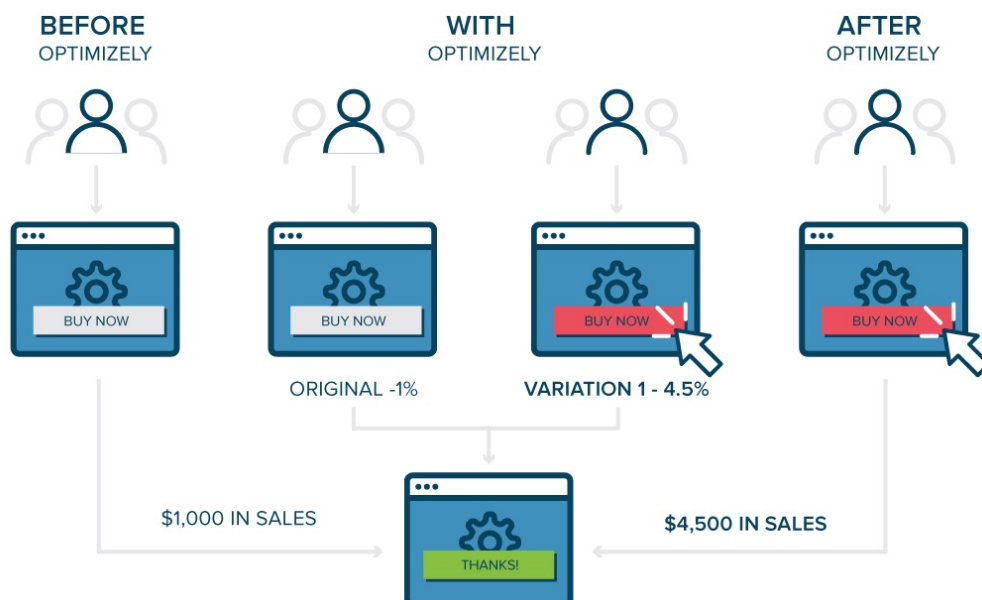
trong khi phiên bản B được sửa đổi ở một số khía cạnh so với phiên bản A (do đó hình thành nhóm điều trị). Ví dụ: trên một trang web thương mại điện tử, phiếu mua hàng thường là một ứng cử viên tốt cho A/B Testing, vì ngay cả tỷ lệ rút hàng giảm nhẹ cũng có thể thể hiện mức tăng đáng kể trong bán hàng. Những cải tiến đáng kể đôi khi có thể được nhìn thấy thông qua các yếu tố thử nghiệm như sao chép văn bản, bố cục, hình ảnh và màu sắc, nhưng không phải lúc nào cũng vậy. Trong các thử nghiệm này, người dùng chỉ thấy một trong hai phiên bản, vì mục đích là khám phá phiên bản nào trong hai phiên bản này thích hợp hơn.

Thử nghiệm đa biến hoặc thử nghiệm đa thức tương tự như A/B Testing, nhưng có thể thử nghiệm nhiều hơn hai phiên bản cùng một lúc hoặc sử dụng nhiều kiểm soát hơn. Các A/B Testing đơn giản không hợp lệ cho các tình huống quan sát, bán thực nghiệm hoặc phi thực nghiệm khác - thường xảy ra với dữ liệu khảo sát, dữ liệu ngoại tuyến và các hiện tượng khác, phức tạp hơn.

Một số người cho rằng A/B Testing là sự thay đổi triết lý và chiến lược kinh doanh trong một số lĩnh vực nhất định, mặc dù cách tiếp cận giống hệt với thiết kế giữa các chủ thể, thường được sử dụng trong nhiều truyền thống nghiên cứu. A/B Testing như một triết lý phát triển web đưa lĩnh vực này phù hợp với một phong trào rộng lớn hơn theo hướng thực hành dựa trên bằng chứng. Lợi ích của A/B Testing được coi là nó có thể được thực hiện liên tục trên hầu hết mọi thứ, đặc biệt là vì hầu hết các phần mềm tự động hóa tiếp thị hiện nay thường đi kèm với khả năng chạy A/B Testing liên tục.

1.1.2 Các A/B testing hoạt động

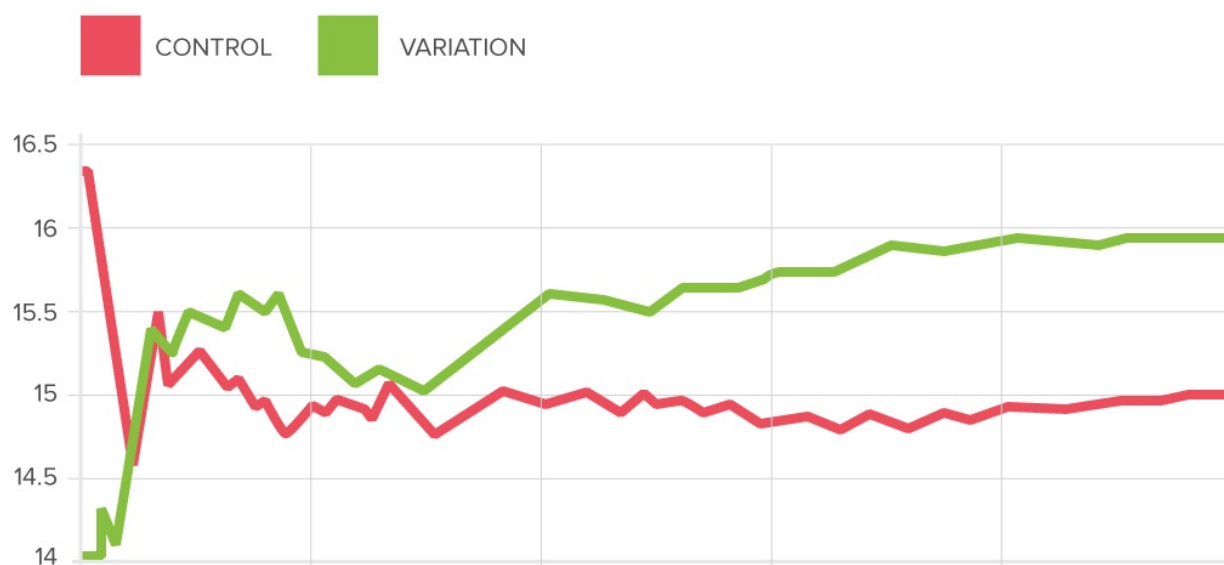
Trong thử nghiệm A / B, bạn chụp một trang web hoặc màn hình ứng dụng và sửa đổi nó để tạo phiên bản thứ hai của cùng một trang. Thay đổi này có thể đơn giản như một dòng tiêu đề, một nút hoặc là một thiết kế lại hoàn toàn của trang. Sau đó, một nửa lưu lượng truy cập của bạn được hiển thị phiên bản gốc của trang (được gọi là điều khiển) và một nửa được hiển thị là phiên bản đã sửa đổi của trang (biến thể).



Hình 1.1: A/B Testing

Khi khách truy cập được phục vụ hoặc kiểm soát hoặc biến thể, mức độ tương tác của họ với

từng trải nghiệm được đo lường và thu thập trong một bảng điều khiển và phân tích thông qua một công cụ thống kê. Sau đó, bạn có thể xác định xem việc thay đổi trải nghiệm có tác động tích cực, tiêu cực hay trung tính đến hành vi của khách truy cập hay không.



Hình 1.2: Biểu đồ kiểm soát biến

1.1.3 Ứng dụng của A/B Testing

A/B Testing tăng trải nghiệm người dùng, giúp quảng cáo online thu thập được các dữ liệu lý tưởng. Họ sẽ hiểu hơn về hành vi quảng cáo nào ảnh hưởng đến người dùng để cung cấp các dịch vụ tốt nhất.

A/B Testing giúp giảm tiền vào đầu tư các chiến dịch marketing. Chi phí thử nghiệm thấp nhưng hiệu quả mang về lại rất cao. Sales và marketing sẽ kết nối với nhau để tương tác hoạt động tốt hơn.

Xu hướng marketing online hiện nay đều chú trọng đến A/B Testing. Chạy A/B Testing cũng là một cách để kiểm tra cạnh tranh tốt hơn với các đối thủ. Để công việc kinh doanh của mình thành công bạn nên nghiên cứu về những lĩnh vực thử nghiệm này. Chúng sẽ giúp bạn thấu hiểu khách hàng, biết được hành vi của khách hàng để đưa ra các chiến lược kinh doanh quan trọng hơn.

Đối với website

Để tìm ra giao diện mà thu hút người sử dụng hơn thì chúng ta có thử nghiệm 2 phiên bản của giao diện đó, hai phiên bản sẽ khác nhau ở cách bố trí nội dung, vị trí đặt các button điều hướng, các hình ảnh,

Đối với Email marketing

Đã qua rồi cái thời mà đẩy hàng trăm ngàn email đi và nghĩ rằng người dùng sẽ đọc được những email của mình gửi. Các email clients ngày càng có các bộ lọc tinh xảo hơn, tổng tất cả

các spam email vào thùng rác. Điều quan trọng là làm thế nào để khách hàng chịu mở email của mình ra xem và tương tác với các email đó. Câu trả lời chính là A/B testing.

Bạn có thể làm A/B Testing để xác định được ngày nào trong tuần tỉ lệ mở mail cao nhất, gửi thời gian nào trong ngày là tốt nhất cho nội dung của bạn, tiêu đề email nào sẽ mang lại tỉ lệ mở mail cao hơn?...

Đối với quảng cáo và bán hàng

Đối với mảng online thì A/B testing thường được dùng để đo lường hiệu quả của các mẫu quảng cáo khác nhau. Ví dụ như khi bạn viết quảng cáo Adwords cho cùng 1 nhóm từ khóa (ad group) thì bạn nên viết 2 mẫu quảng cáo khác nhau và cho chạy song song để biết mẫu quảng cáo nào hiệu quả hơn sau một thời gian chạy. Tương tự với quảng cáo cho Facebook, sử dụng các thiết kế quảng cáo khác nhau cho cùng một chiến dịch để đo lường hiệu quả sau đó chọn mẫu thiết kế nào hiệu quả hơn để chạy tiếp. Việc tối ưu hóa quảng cáo thường xuyên bằng cách test các lựa chọn khác nhau sẽ giúp bạn liên tục cải thiện được tỷ lệ chuyển đổi và giúp quảng cáo chạy ngày càng hiệu quả hơn.

Đối với mảng offline thì A/B testing thường có thể được dùng để đánh giá hiệu quả của các kênh quảng cáo như báo giấy, tờ rơi, billboard... Chẳng hạn bằng cách sử dụng các mã coupon khác nhau cho từng mẫu quảng cáo trên báo, mẫu tờ rơi, hoặc billboard, nhà quảng cáo có thể nắm được mẫu quảng cáo nào hiệu quả hơn thông qua việc có nhiều người sử dụng mã coupon nào hơn.

Đối với ứng dụng di động

A/B testing cũng được ứng dụng trong việc phát triển ứng dụng di động và tương tự như website, chủ yếu nhằm cải thiện UI/UX của sản phẩm.

Với các ứng dụng điện thoại di động thì việc tiến hành testing thường khó khăn hơn nhiều cả về mặt kỹ thuật lẫn về hành vi người dùng.

Về mặt kỹ thuật thì để tiến hành test, thì phiên bản ứng dụng cần được cập nhật, được duyệt bởi AppStore hay Google Play rồi mới đến với người dùng do đó tốn nhiều thời gian hơn.

Về phương diện hành vi người dùng, không phải ai cũng sẽ cập nhật ngay phiên bản mới và trải nghiệm người dùng trên điện thoại di động hoàn toàn khác so với trên web.

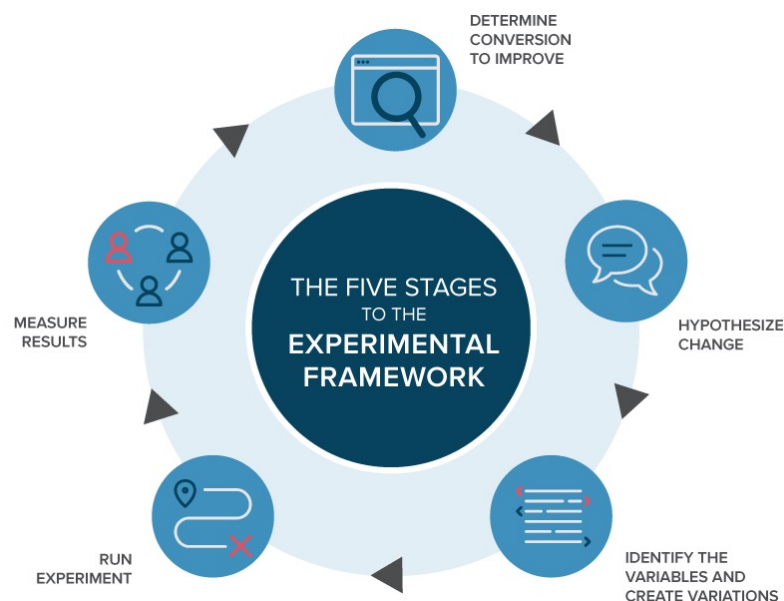
1.1.4 Quy trình thử nghiệm A / B

Sau đây là khung thử nghiệm A / B mà có thể sử dụng để bắt đầu chạy thử nghiệm:

- **Thu thập dữ liệu:** Các phân tích của bạn thường sẽ cung cấp thông tin chi tiết về nơi bạn có thể bắt đầu tối ưu hóa. Nó hữu ích để bắt đầu với các khu vực có lưu lượng truy cập cao trên trang web hoặc ứng dụng của bạn để cho phép bạn thu thập dữ liệu nhanh hơn. Tìm kiếm các trang có tỷ lệ chuyển đổi thấp hoặc tỷ lệ rời mạng cao có thể được cải thiện.
- **Xác định mục tiêu:** Mục tiêu chuyển đổi của bạn là số liệu mà bạn đang sử dụng để xác định xem biến thể có thành công hơn phiên bản gốc hay không. Mục tiêu có thể là bất cứ điều gì từ việc nhấp vào nút hoặc liên kết đến việc mua sản phẩm và đăng ký e-mail.
- **Tạo giả thuyết:** Khi bạn đã xác định được mục tiêu, bạn có thể bắt đầu tạo ra các ý tưởng và giả thuyết thử nghiệm A / B để biết lý do tại sao bạn cho rằng chúng sẽ tốt hơn phiên

bản hiện tại. Khi bạn đã có danh sách các ý tưởng, hãy sắp xếp thứ tự ưu tiên cho chúng về tác động dự kiến và mức độ khó thực hiện.

- **Tạo các biến thể:** Sử dụng phần mềm thử nghiệm A / B của bạn (như Optimizely), thực hiện các thay đổi mong muốn đối với một phần tử của trang web hoặc trải nghiệm ứng dụng dành cho thiết bị di động của bạn. Điều này có thể là thay đổi màu của nút, hoán đổi thứ tự của các phần tử trên trang, ẩn các phần tử điều hướng hoặc một cái gì đó hoàn toàn tùy chỉnh. Nhiều công cụ kiểm tra A / B hàng đầu có trình chỉnh sửa trực quan sẽ giúp những thay đổi này trở nên dễ dàng. Đảm bảo QA thử nghiệm của bạn để đảm bảo thử nghiệm hoạt động như mong đợi.
- **Chạy thử nghiệm:** Bắt đầu thử nghiệm của bạn và đợi khách tham gia! Tại thời điểm này, khách truy cập vào trang web hoặc ứng dụng của bạn sẽ được chỉ định ngẫu nhiên cho quyền kiểm soát hoặc biến thể trải nghiệm của bạn. Tương tác của họ với từng trải nghiệm được đo lường, tính toán và so sánh để xác định cách mỗi trải nghiệm hoạt động.
- **Phân tích kết quả:** Sau khi thử nghiệm của bạn hoàn tất, đã đến lúc phân tích kết quả. Phần mềm thử nghiệm A / B của bạn sẽ trình bày dữ liệu từ thử nghiệm và cho bạn thấy sự khác biệt giữa hai phiên bản trang của bạn hoạt động như thế nào và liệu có sự khác biệt có ý nghĩa thống kê hay không.



Hình 1.3: Quy trình thử nghiệm A / B

1.2 Thông số

1.2.1 Tỷ lệ chuyển đổi (Conversion Rate)

Hiểu một cách đơn giản, tỷ lệ chuyển đổi chính là tỷ lệ người truy cập vào website đã thực hiện hành động bạn mong muốn (click vào một trang quảng cáo, mua hàng, hay điền form khảo sát,...) trên tổng số lượt truy cập vào website

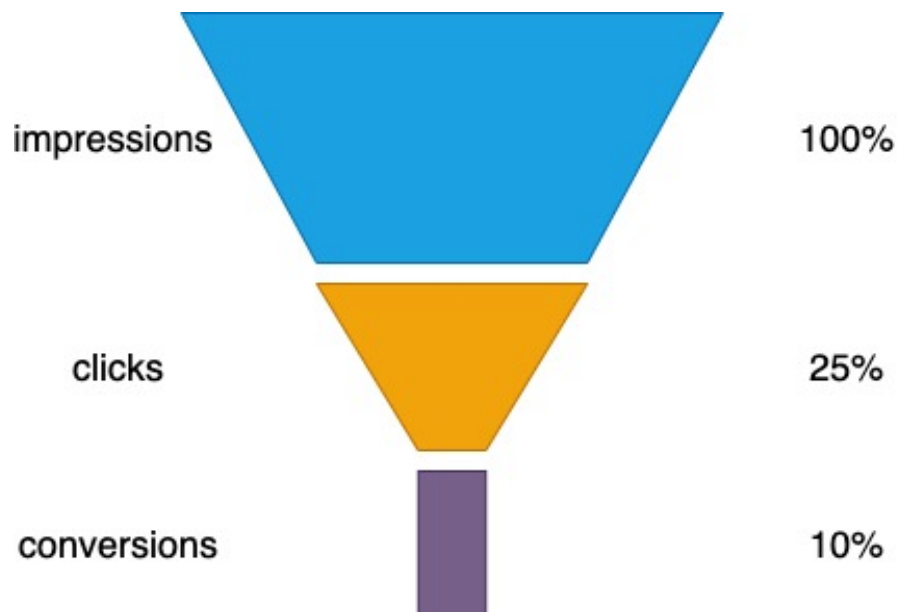
$$CR = \frac{Conversions}{Clicks} * 100\%$$

Lấy một ví dụ đơn giản như: Bạn có một website bán hàng, bạn muốn có thật nhiều người mua hàng trên website của bạn. Hôm nay website của bạn có 200 lượt truy cập trong đó có 5 lượt mua hàng thông qua website của bạn, khi đó bạn sẽ có con số về tỷ lệ chuyển đổi là $CR = (5/200) * 100\% = 2,5\%$. Việc của bạn là phải có các chiến lược để con số này càng cao càng tốt.

Tỷ lệ chuyển đổi tương tự như chỉ số doanh thu tăng lên vì trong thương mại điện tử, một chuyển đổi tương đương với một lần bán hàng. Tuy nhiên, việc đếm số lượng bán hàng khác với tính toán doanh thu: doanh thu cao hơn không cho thấy hoạt động tăng lên bởi vì nó dựa trên lợi nhuận chứ không chỉ dựa trên số lượng bán hàng. Biên lợi nhuận lớn có thể làm sai lệch kết quả; ví dụ, một lần bán thêm duy nhất với tỷ suất lợi nhuận lớn có thể cho thấy doanh thu tăng lên. Việc tính doanh số bán hàng đơn giản và rõ ràng hơn, nhưng điểm mạnh của nó cũng chứa một điểm yếu, như bạn sẽ thấy ở phần sau.

Lợi ích của việc sử dụng chuyển đổi làm phép đo là nó được tính ở quy mô nhỏ hơn: tìm kiếm. Do số lượng tìm kiếm dao động nhanh hơn số lượng chuyển đổi (khách hàng chuyển đổi ít thường xuyên hơn nhiều so với số lượng họ tìm kiếm), nên tỷ lệ tìm kiếm trên chuyển đổi có thể thay đổi nhanh chóng nếu khách hàng chuyển đổi thường xuyên hơn. Do đó, chuyển đổi có thể phát hiện các sửa đổi hệ thống dễ dàng hơn.

Do đó, tỷ lệ chuyển đổi ít mang tính định hướng hơn so với doanh thu nhưng có thể được phát hiện nhanh chóng - tức là tỷ lệ này nhạy cảm hơn. Thuộc tính này thích hợp hơn cho thử nghiệm A / B vì nó đạt được mức ý nghĩa nhanh hơn. Điều đó nói rằng, chuyển đổi không phải là một hình thức đo lường hoàn hảo vì nó coi tất cả doanh số bán hàng là bình đẳng, không mang lại giá trị kinh doanh giống nhau do tăng doanh thu.



Hình 1.4: Conversions

1.2.2 Tỷ lệ click (CTR)

CTR là tỷ lệ cho biết tần suất những người xem quảng cáo hoặc danh sách sản phẩm sẽ nhấp vào quảng cáo hoặc danh sách sản phẩm miễn phí của bạn. Tỷ lệ nhấp (CTR) có thể được sử dụng để đánh giá mức độ hoạt động của các từ khóa và quảng cáo cũng như danh sách miễn phí của bạn.

CTR là số nhấp chuột mà quảng cáo của bạn nhận được chia cho số lần quảng cáo của bạn được hiển thị: nhấp chuột ÷ hiển thị = CTR. Ví dụ: nếu bạn có 5 nhấp chuột và 100 hiển thị, thì CTR của bạn sẽ là 5

$$CTR = \frac{Clicks}{Impressions} * 100\%$$

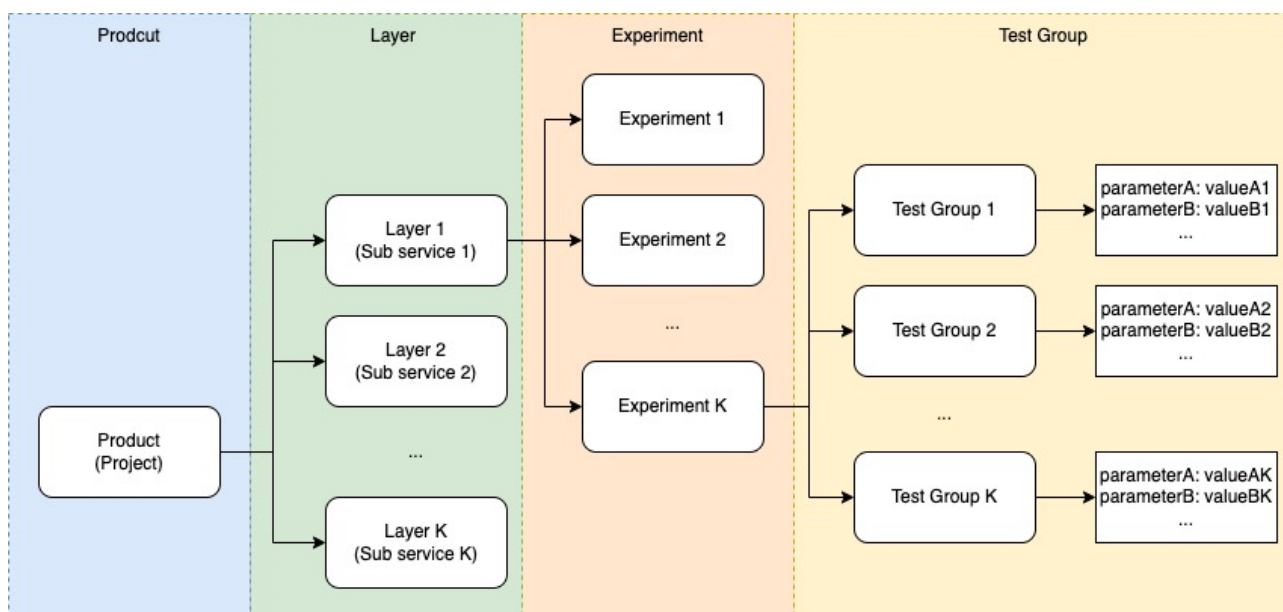
Mỗi quảng cáo, danh sách và từ khóa của bạn có CTR riêng mà bạn có thể thấy được liệt kê trong tài khoản của mình.

CTR cao là một dấu hiệu tốt cho thấy người dùng thấy quảng cáo và danh sách của bạn hữu ích và có liên quan. CTR cũng đóng góp vào CTR dự kiến của từ khóa của bạn, là một thành phần của Xếp hạng quảng cáo. Lưu ý rằng CTR tốt có liên quan đến những gì bạn đang quảng cáo và trên mạng nào.

Bạn có thể sử dụng CTR để đánh giá những quảng cáo, danh sách và từ khóa nào thành công với bạn và những quảng cáo nào cần được cải thiện. Từ khóa, quảng cáo và danh sách của bạn càng liên quan đến nhau và với doanh nghiệp của bạn, thì càng có nhiều khả năng người dùng nhấp vào quảng cáo hoặc danh sách của bạn sau khi tìm kiếm trên cụm từ khóa của bạn.

1.3 Cấu trúc của thí nghiệm

Một A/B Testing được chia làm thành nhiều tầng lớp. Mỗi tầng lớp đều có ý nghĩa riêng và được chia ra để có thể phục vụ cho logic business phức tạp.

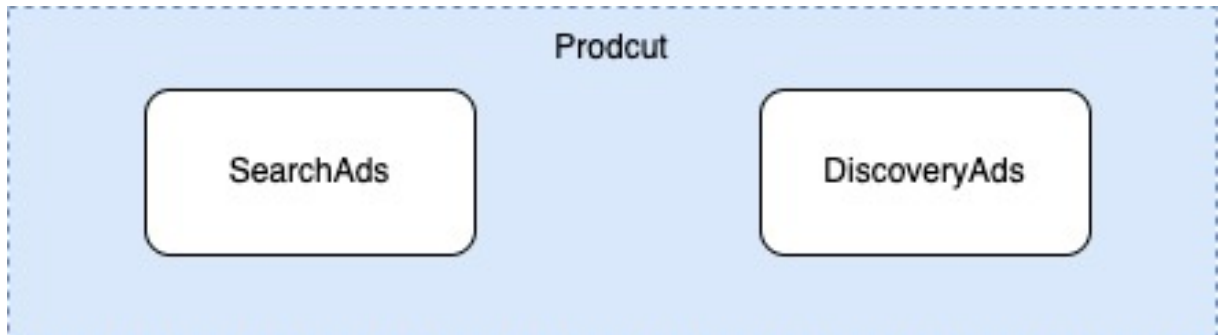


Hình 1.5: Tổng quan của một A/B Testing

1.3.1 Product

Product là thực thể cao nhất trong một A/B Testing. Mỗi product thường đại diện cho một tập thể lớn. Mỗi team thường có nhiều bộ phận cho các lĩnh vực khác nhau, mỗi bộ phận sẽ thường có một hoặc nhiều Product.

Mỗi Product được chia ra để tách biệt các A/B Testing giữa các team khác nhau. A/B Testing thuộc hai Product khác nhau sẽ không liên quan gì đến nhau, không để so sánh được cùng với nhau.



Hình 1.6: Product

Ví dụ: trong phòng quảng cáo thì sẽ có hai bộ phận lớn là quảng cáo tìm kiếm và quảng cáo gợi ý. Mỗi bộ phận sẽ có 2 product khác nhau để khi thực hiện A/B Testing không ảnh hưởng lẫn nhau.

1.3.2 Layer

Layer là thực thể thứ hai đứng sau thực thể Product. Một Product có thể có nhiều Layer. Mỗi Layer đại diện cho một logic nhỏ hơn của Product đấy. Mỗi layer sẽ có 100% lượng truy cập.

Ví dụ: trong mobile app có thể có 2 layer là layer giao diện cài đặt hay layer giao diện chơi game.

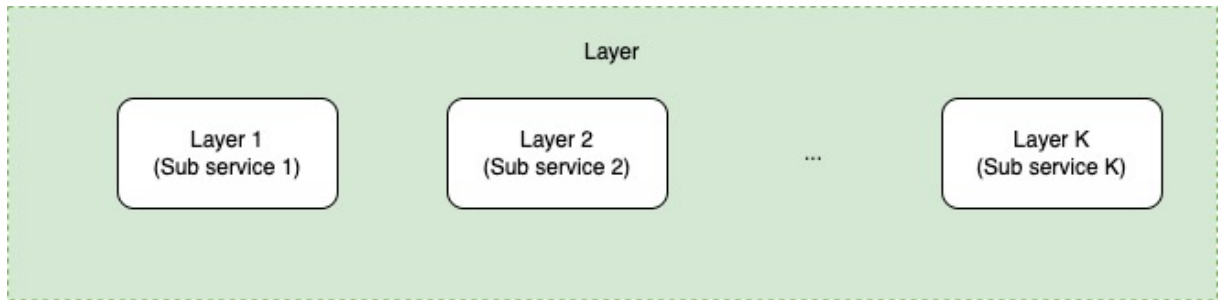
Ngoài ra, với mỗi layer, có thể cài đặt layer này sử dụng một cơ sở để phân chia lượng truy cập. Các cơ sở có thể là:

- **request_id**: sử dụng request_id để phân chia truy cập.
- **user_id**: sử dụng user_id để phân chia truy cập.
- **request_id và user_id**: sử dụng cả request_id và user_id để phân chia truy cập.

Khi một truy cập được thông qua, truy cập đó sẽ đi qua từng Layer. Với mỗi layer, nó sẽ sử dụng cơ sở để quyết định xem layer này sẽ sử dụng A/B Testing hay thí nghiệm nào.

1.3.3 Experiment

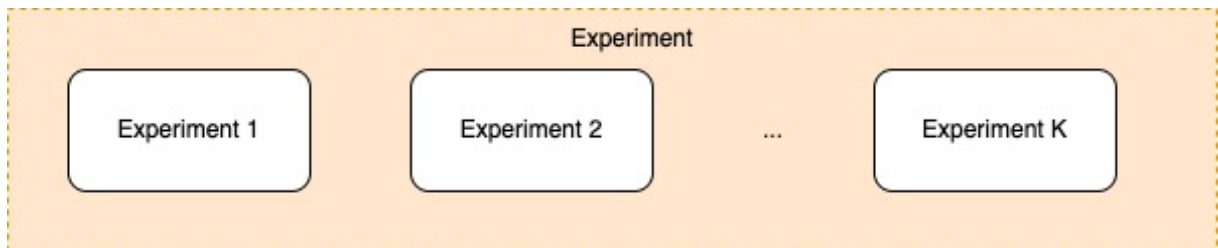
Experiment là thực thể tiếp theo đứng dưới một layer. Một experiment chính là một thí nghiệm mà được tạo ra để thử nghiệm một mô hình tính toán mới. Ví dụ: thử nghiệm thuật toán A mới để tính toán giá cả, thử nghiệm giao dịch màu mới để tăng trải nghiệm người dùng, v.v.



Hình 1.7: Layer

Mỗi experiment sẽ thuộc một Layer và chiếm một tỉ lệ phần trăm truy cập của layer đó. Một layer có thể có nhiều experiment. Ví dụ: layer tính toán giá cả sẽ có hai thí nghiệm thử nghiệm hai thuật toán A và B, mỗi thuật toán có 10% lượng truy cập để đánh giá.

Như vậy, với mỗi layer, khi một try cập đi đến, nó sẽ sử dụng không hoặc một experiment.



Hình 1.8: Experiment

1.3.4 Test Group

Test Group là thực thể nhỏ nhất trong một A/B Testing. Test Group đại diện cho một biến thể của một Experiment. Ví dụ khi sử dụng thuật toán A để tính toán giá cả của sản phẩm, thuật toán A có thể sử dụng hai chỉ số khác nhau để tính toán tương ứng với hai test group khác nhau.

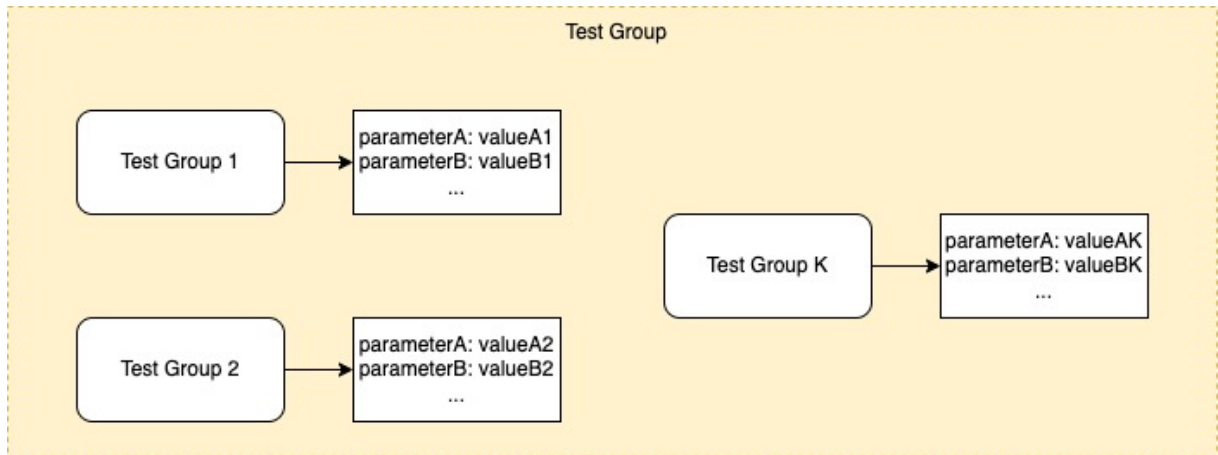
Khi thực hiện một Experiment, mỗi Test Group sẽ có một lượng truy cập giống nhau và kết quả sẽ được so sánh giữa hai Test Group. Ví dụ trong thuật toán A, hai biến thể X và Y sẽ có một biến thể mang lại kết quả tốt hơn khi so sánh với nhau.

1.4 Công nghệ thực hiện

1.4.1 Golang

Golang còn được gọi là Go là một ngôn ngữ lập trình open-source, compiled và statically-typed do Google phát triển.

Go là một ngôn ngữ lập trình đa năng với cú pháp đơn giản và một thư viện tiêu chuẩn mạnh mẽ. Go cải thiện các khái niệm như lập trình mệnh lệnh và hướng đối tượng và do đó đơn giản hóa môi trường làm việc cho các lập trình viên.



Hình 1.9: Test Group



Golang

Hình 1.10: Golang

Lịch sử của Golang

Ngôn ngữ lập trình Go ra đời bởi vì mọi thứ trở nên phức tạp hơn nhiều trong codebases của Google.

Nó được thiết kế bởi Robert Griesemer, Rob Pike và Ken Thompson, những người được cho là có chung sở thích với C++.

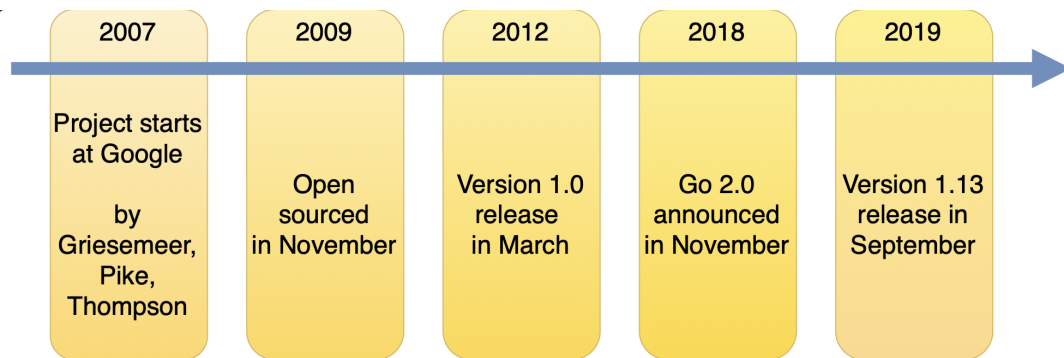
Go được công bố ra mắt công chúng vào năm 2009, và nó được làm mã nguồn mở vào năm 2012 khi phiên bản đầu tiên của nó, 1.0, được phát hành.

Go nhanh chóng trở nên phổ biến và trở thành lựa chọn đầu tiên của nhiều nhà phát triển do tính đơn giản, dễ đọc, hiệu quả và tính đồng thời của nó. Đồng thời có nghĩa là nó có thể chạy nhiều tác vụ cùng một lúc.

Go được sử dụng để lập trình phía máy chủ, phát triển trò chơi, lập trình dựa trên đám mây và thậm chí cả Khoa học dữ liệu. Nó cũng phổ biến để tạo các công cụ dòng lệnh.

Ngày nay, nhiều gã khổng lồ công nghệ sử dụng Go như Google, Netflix, Twitch, Ethereum, Dropbox, Kubernetes, Docker, Heroku, v.v.

Không có gì ngạc nhiên khi những người như Kubernetes, Docker và Heroku đang sử dụng cò vậy vì cloud-based programming là một trong những lý do chính khiến Go được thiết kế.



Hình 1.11: Lịch sử của Golang

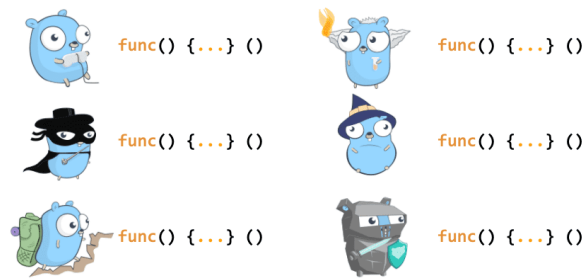
Ưu điểm của Golang

- **Cú pháp đơn giản:** Go có cú pháp ngắn gọn và đơn giản giúp viết mã dễ đọc và dễ bảo trì.
- **Ngôn ngữ biên dịch:**
- **Liên kết tĩnh:** Trình biên dịch hỗ trợ liên kết tĩnh có nghĩa là bạn có thể liên kết tĩnh dự án của mình thành một tệp nhị phân lớn và sau đó chỉ cần triển khai nó lên máy chủ.
- **Mã nguồn mở:** Go là mã nguồn mở nên bạn có thể đọc mã nguồn và đóng góp vào kho lưu trữ.
- **Ngôn ngữ Strong và Statically Type:** Strong có nghĩa là sau khi bạn tạo một số biến bằng cách sử dụng một số kiểu dữ liệu thì đối với toàn bộ ứng dụng, nó sẽ vẫn là kiểu dữ liệu đấy. Statically Type là tất cả các biến phải xác định tại thời điểm biên dịch.
- **Cộng đồng mạnh:** Golang có một cộng đồng tuyệt vời để theo dõi vấn đề và giải pháp.
- **Biên dịch nhanh:** Golang sẽ chỉ bao gồm các gói sẽ nhập trực tiếp trong mã.
- **Garbage Collector:** Đó là một tính năng chính của Go. Go vượt trội trong việc cung cấp nhiều quyền kiểm soát đối với việc phân bổ bộ nhớ và đã giảm đáng kể độ trễ trong các phiên bản gần đây nhất của Garbage Collector.
- **Concurrency tích hợp:** Go có tính năng concurrency được tích hợp sẵn. Sử dụng Go Routines và các kênh, bạn có thể xử lý đồng thời các tác vụ rất dễ dàng và hiệu quả.

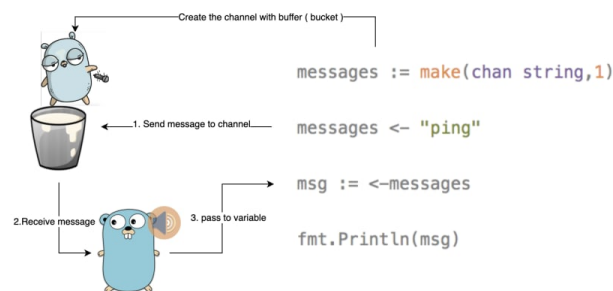
Ứng dụng của Golang

- **Dịch vụ đám mây & mạng:** Go giúp các doanh nghiệp xây dựng và mở rộng quy mô hệ thống điện toán đám mây. Khi các ứng dụng và quá trình xử lý chuyển sang đám mây, đồng thời trở thành một vấn đề rất lớn. Về bản chất, các hệ thống điện toán đám mây chia sẻ và mở rộng quy mô tài nguyên. Điều phối quyền truy cập vào các tài nguyên được chia sẻ là một vấn đề ảnh hưởng đến mọi quá trình xử lý ứng dụng trên đám mây và yêu cầu ngôn ngữ lập trình “hướng đến phát triển các ứng dụng đồng thời có độ tin cậy cao”.

Goroutines



Hình 1.12: Goroutine



Hình 1.13: Channel

- **Command-line Interfaces (CLIs):** Giao diện dòng lệnh (CLI), không giống như giao diện người dùng đồ họa (GUI), chỉ ở dạng văn bản. Các ứng dụng cơ sở hạ tầng và đám mây chủ yếu dựa trên CLI do khả năng tự động hóa và điều khiển từ xa dễ dàng.
- **Web Development:** Go được thiết kế để cho phép các nhà phát triển nhanh chóng phát triển các ứng dụng web có khả năng mở rộng và bảo mật. Go đi kèm với một máy chủ web dễ sử dụng, an toàn và hiệu quả và bao gồm thư viện tạo khuôn mẫu web riêng. Go có hỗ trợ tuyệt vời cho tất cả các công nghệ mới nhất từ HTTP / 2, cơ sở dữ liệu như MySQL, MongoDB và Elasticsearch, cho đến các tiêu chuẩn mã hóa mới nhất bao gồm TLS 1.3. Các ứng dụng web của Go chạy nguyên bản trên Google App Engine và Google Cloud Run (để dễ dàng mở rộng quy mô) hoặc trên bất kỳ môi trường, đám mây hoặc hệ điều hành nào nhờ tính di động cực cao của Go.
- **Development Operations & Site Reliability Engineering:** Thời gian khởi động và xây dựng nhanh. Thư viện tiêu chuẩn mở rộng của Go — bao gồm các gói cho các nhu cầu thông thường như HTTP, I / O tệp, thời gian, biểu thức chính quy, thực thi và các định dạng JSON / CSV — cho phép DevOps / SRE đi đúng vào logic kinh doanh của họ. Ngoài ra, hệ thống kiểu tĩnh và xử lý lỗi rõ ràng của Go giúp các tập lệnh nhỏ thậm chí còn mạnh mẽ hơn.



Hình 1.14: React

1.4.2 React

React là gì

React (còn được gọi là React.js hoặc ReactJS) là một thư viện JavaScript front-end mã nguồn mở và miễn phí để xây dựng giao diện người dùng dựa trên các thành phần UI. Nó được duy trì bởi Meta (trước đây là Facebook) và một cộng đồng các nhà phát triển và công ty cá nhân. React có thể được sử dụng làm cơ sở để phát triển các single-page, ứng dụng di động hoặc ứng dụng web với các framework như Next.js.

Tuy nhiên, React chỉ quan tâm đến việc quản lý trạng thái và hiển thị trạng thái đó cho DOM, vì vậy việc tạo các ứng dụng React thường yêu cầu sử dụng các thư viện bổ sung để định tuyến, cũng như một số chức năng phía client.

Đặc trưng của React

Declarative

React giúp tạo giao diện người dùng tương tác dễ dàng hơn. Thiết kế các khung nhìn đơn giản cho từng trạng thái trong ứng dụng của bạn và React sẽ cập nhật và hiển thị các thành phần phù hợp một cách hiệu quả khi dữ liệu của bạn thay đổi.

Chế độ xem Declarative làm cho code của bạn dễ đoán hơn và dễ debug hơn.

Component-Based

Xây dựng các thành phần được đóng gói quản lý trạng thái của riêng chúng, sau đó biên soạn chúng để tạo giao diện người dùng phức tạp.

Vì logic thành phần được viết bằng JavaScript thay vì các mẫu, bạn có thể dễ dàng chuyển dữ liệu phong phú thông qua ứng dụng của mình và giữ trạng thái không nằm trong DOM.

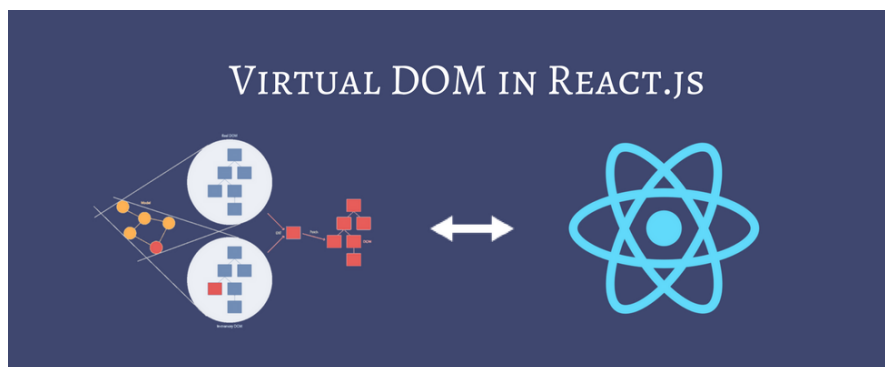
Learn Once, Write Anywhere

Chúng tôi không đưa ra giả định về phần còn lại của nền tảng công nghệ của bạn, vì vậy bạn có thể phát triển các tính năng mới trong React mà không cần viết lại mã hiện có.

React cũng có thể render trên server bằng Node và compile ra các ứng dụng di động bằng React Native.

Virtual DOM

Những Framework sử dụng Virtual-DOM như ReactJS khi Virtual-DOM thay đổi, chúng ta không cần thao tác trực tiếp với DOM trên View mà vẫn phản ánh được sự thay đổi đó. Do Virtual-DOM vừa đóng vai trò là Model, vừa đóng vai trò là View nên mọi sự thay đổi trên Model đã kéo theo sự thay đổi trên View và ngược lại. Có nghĩa là mặc dù chúng ta không tác động trực tiếp vào các phần tử DOM ở View nhưng vẫn thực hiện được cơ chế Data-binding.



Hình 1.15: Virtual DOM

Điều này làm cho tốc độ ứng dụng tăng lên đáng kể – một lợi thế không thể tuyệt vời hơn khi sử dụng Virtual-DOM.

Ứng dụng của React

1.4.3 Redis



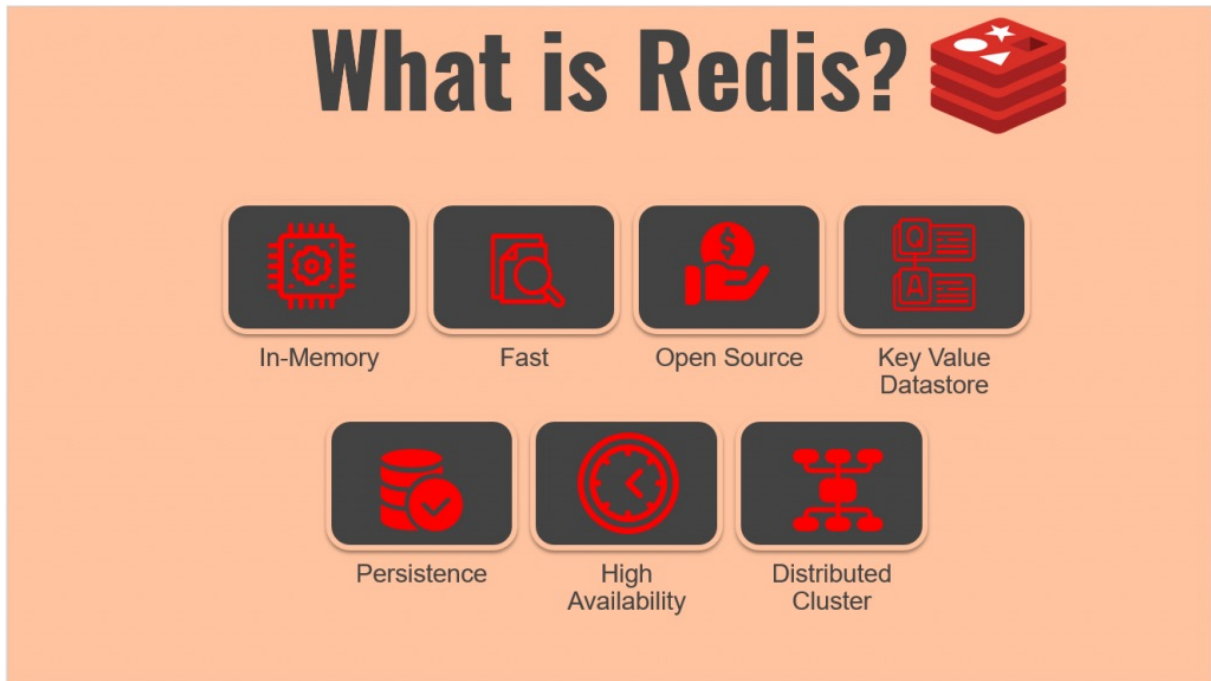
Hình 1.16: Redis

Redis là gì

Redis, viết tắt của Remote Dictionary Server, là một kho lưu trữ dữ liệu key-value, mã nguồn mở, in-memory và nhanh chóng. Dự án bắt đầu khi Salvatore Sanfilippo, nhà phát triển ban đầu của Redis, muốn cải thiện khả năng mở rộng của công ty khởi nghiệp của mình. Từ đó, anh phát triển Redis, hiện được sử dụng làm cơ sở dữ liệu, bộ nhớ đệm, message broker và hàng đợi.

Redis cung cấp thời gian phản hồi dưới mili giây, cho phép hàng triệu yêu cầu mỗi giây cho các ứng dụng thời gian thực trong các ngành như trò chơi, công nghệ quảng cáo, dịch vụ tài chính, chăm sóc sức khỏe và IoT. Ngày nay, Redis là một trong những engine mã nguồn mở phổ biến nhất hiện nay, được Stack Overflow đặt tên là cơ sở dữ liệu "Được yêu thích nhất" trong 5 năm liên tiếp. Do hiệu suất nhanh, Redis là một lựa chọn phổ biến cho bộ nhớ đệm, quản lý phiên, chơi trò chơi, bảng xếp hạng, phân tích thời gian thực, không gian địa lý, gọi xe, trò chuyện / nhắn tin, phát trực tuyến phương tiện và ứng dụng pub / sub.

Tính năng của Redis



Hình 1.17: Tính năng của Redis

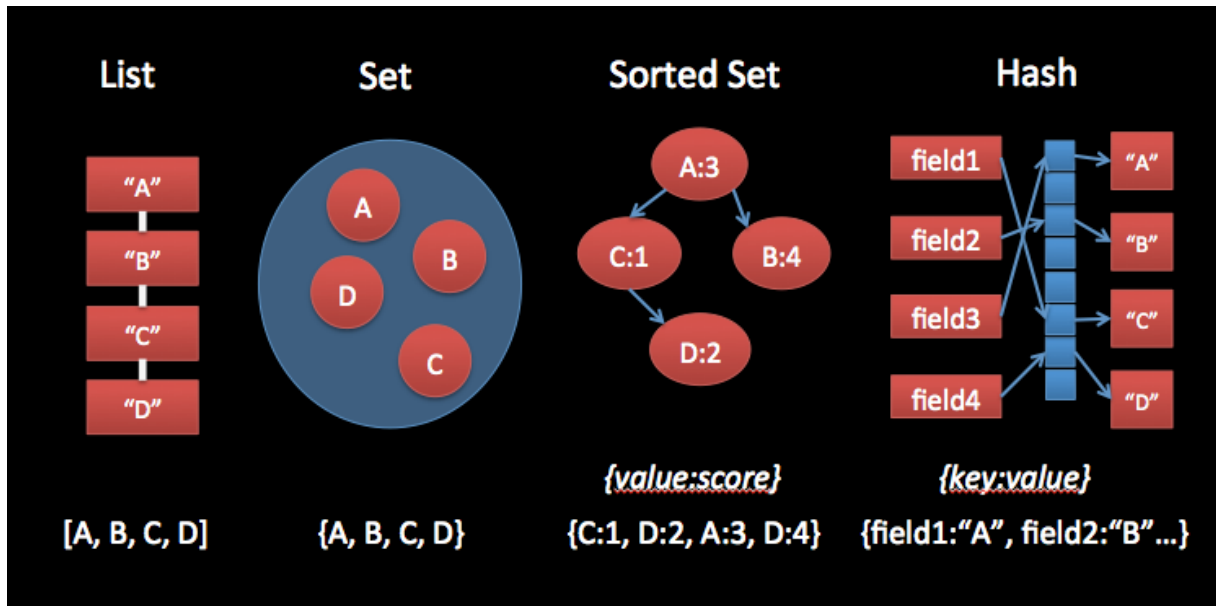
Performance

Tất cả dữ liệu Redis nằm trong bộ nhớ, cho phép truy cập dữ liệu có độ trễ thấp và thông lượng cao. Không giống như cơ sở dữ liệu truyền thống, kho lưu trữ dữ liệu trong bộ nhớ không yêu cầu chuyển tới đĩa, giảm độ trễ của động cơ xuống micro giây. Do đó, các kho lưu trữ dữ liệu trong bộ nhớ có thể hỗ trợ nhiều thao tác hơn và thời gian phản hồi nhanh hơn. Kết quả là mang lại hiệu suất cực nhanh với các thao tác đọc và ghi trung bình chỉ mất chưa đến một phần nghìn giây và hỗ trợ hàng triệu thao tác mỗi giây.

Cấu trúc dữ liệu linh hoạt

Không giống như các kho dữ liệu khóa-giá trị khác cung cấp cấu trúc dữ liệu hạn chế, Redis có rất nhiều cấu trúc dữ liệu để đáp ứng nhu cầu ứng dụng của bạn. Các kiểu dữ liệu của Redis bao gồm:

- **Strings**: dữ liệu văn bản hoặc dữ liệu nhị phân có kích thước lên đến 512MB
- **Lists**: tập hợp các Chuỗi theo thứ tự chúng được thêm vào
- **Sets**: một tập hợp các chuỗi không có thứ tự với khả năng giao nhau, kết hợp và khác các loại Bộ khác
- **Sorted Sets** : Các bộ được sắp xếp theo một giá trị
- **Hashes**: một cấu trúc dữ liệu để lưu trữ danh sách các trường và giá trị
- **Bitmap**: một kiểu dữ liệu cung cấp các hoạt động ở mức bit
- **HyperLogLogs**: cấu trúc dữ liệu xác suất để ước tính các mục duy nhất trong tập dữ liệu



Hình 1.18: Cấu trúc dữ liệu của Redis

- **Streams:** cấu trúc dữ liệu nhật ký Hàng đợi tin nhắn
- **Geospatial:** mục nhập dựa trên kinh độ / vĩ độ Bản đồ, "lân cận"
- **JSON:** một đối tượng lồng nhau, bán cấu trúc của các giá trị được đặt tên hỗ trợ số, chuỗi, Boolean, mảng và các đối tượng khác

Đơn giản và dễ sử dụng

Redis cho phép bạn viết mã phức tạp truyền thống với ít dòng hơn, đơn giản hơn. Với Redis, bạn viết ít dòng mã hơn để lưu trữ, truy cập và sử dụng dữ liệu trong các ứng dụng của mình. Sự khác biệt là các nhà phát triển sử dụng Redis có thể sử dụng cấu trúc lệnh đơn giản trái ngược với các ngôn ngữ truy vấn của cơ sở dữ liệu truyền thống. Ví dụ: bạn có thể sử dụng cấu trúc dữ liệu băm Redis để di chuyển dữ liệu đến kho dữ liệu chỉ với một dòng mã. Một tác vụ tương tự trên kho dữ liệu không có cấu trúc dữ liệu băm sẽ yêu cầu nhiều dòng mã để chuyển đổi từ định dạng này sang định dạng khác. Redis đi kèm với cấu trúc dữ liệu gốc và nhiều tùy chọn để thao tác và tương tác với dữ liệu của bạn. Hơn một trăm ứng dụng khách mã nguồn mở có sẵn cho các nhà phát triển Redis. Các ngôn ngữ được hỗ trợ bao gồm Java, Python, PHP, C, C++, C#, JavaScript, Node.js, Ruby, R, Go và nhiều ngôn ngữ khác.

Nhân rộng và bền bỉ

Redis sử dụng kiến trúc bản sao chính và hỗ trợ sao chép không đồng bộ, nơi dữ liệu có thể được sao chép sang nhiều máy chủ bản sao. Điều này cung cấp hiệu suất đọc được cải thiện (vì các yêu cầu có thể được phân chia giữa các máy chủ) và khôi phục nhanh hơn khi máy chủ chính gặp sự cố. Để bền bỉ, Redis hỗ trợ sao lưu theo thời gian (sao chép tập dữ liệu Redis vào đĩa).

Redis không được xây dựng để trở thành một cơ sở dữ liệu bền và nhất quán. Nếu bạn cần một cơ sở dữ liệu bền, tương thích với Redis, hãy xem xét Amazon MemoryDB cho Redis. Bởi vì MemoryDB sử dụng nhật ký giao dịch lâu dài để lưu trữ dữ liệu trên nhiều Vùng khả dụng (AZ), bạn có thể sử dụng nó làm cơ sở dữ liệu chính của mình. MemoryDB được xây dựng có mục đích cho phép các nhà phát triển sử dụng API Redis mà không phải lo lắng về việc quản lý bộ nhớ cache, cơ sở dữ liệu riêng biệt hoặc cơ sở hạ tầng bên dưới.

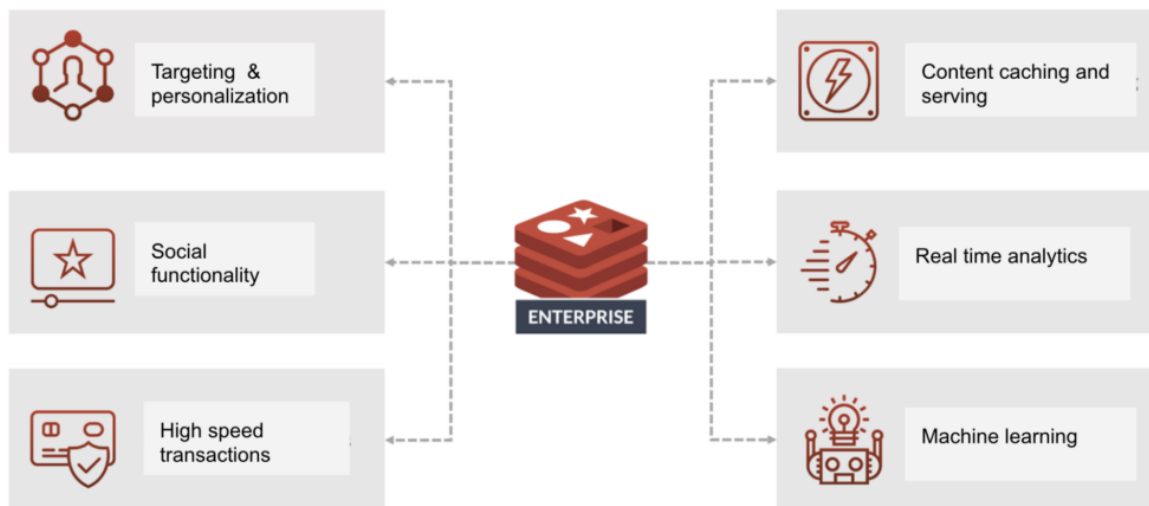
Tính khả dụng và khả năng mở rộng cao

Redis cung cấp một kiến trúc bản sao chính trong một nút chính hoặc một cấu trúc liên kết nhóm. Điều này cho phép bạn xây dựng các giải pháp có tính khả dụng cao, cung cấp hiệu suất và độ tin cậy nhất quán. Khi bạn cần điều chỉnh kích thước cụm của mình, các tùy chọn khác nhau để mở rộng và mở rộng quy mô trong hoặc ngoài cũng có sẵn. Điều này cho phép cụm của bạn phát triển theo nhu cầu của bạn.

Mã nguồn mở

Redis là một dự án mã nguồn mở được hỗ trợ bởi một cộng đồng sôi động, bao gồm AWS. Không có nhà cung cấp hoặc công nghệ nào bị khóa vì Redis dựa trên các tiêu chuẩn mở, hỗ trợ các định dạng dữ liệu mở và có một tập hợp khách hàng phong phú.

Ứng dụng của Redis



Hình 1.19: Ứng dụng của Redis

Bộ nhớ đệm

Redis là một lựa chọn tuyệt vời để triển khai bộ nhớ đệm trong bộ nhớ sẵn có cao để giảm độ trễ truy cập dữ liệu, tăng thông lượng và giảm tải cơ sở dữ liệu và ứng dụng NoSQL hoặc cơ sở dữ liệu quan hệ của bạn. Redis có thể phân phát các mặt hàng được yêu cầu thường xuyên ở thời gian phản hồi dưới mili giây và cho phép bạn dễ dàng mở rộng quy mô để có tải cao hơn mà không tăng phần phụ trợ tốn kém hơn. Bộ nhớ đệm kết quả truy vấn cơ sở dữ liệu, bộ nhớ đệm phiên liên tục, bộ nhớ đệm trang web và bộ nhớ đệm của các đối tượng được sử dụng thường xuyên như hình ảnh, tệp và siêu dữ liệu đều là những ví dụ phổ biến về bộ nhớ đệm với Redis.

Trò chuyện, nhắn tin và hàng đợi

Redis hỗ trợ Pub / Sub với tính năng so khớp mẫu và nhiều cấu trúc dữ liệu khác nhau như danh sách, tập hợp được sắp xếp và băm. Điều này cho phép Redis hỗ trợ các phòng trò chuyện hiệu suất cao, luồng bình luận trong thời gian thực, nguồn cấp dữ liệu mạng xã hội và giao tiếp liên máy chủ. Cấu trúc dữ liệu Redis List giúp dễ dàng triển khai một hàng đợi nhẹ. Danh sách cung cấp các hoạt động nguyên tử cũng như khả năng chặn, làm cho chúng phù hợp với nhiều ứng dụng khác nhau yêu cầu trình môi giới thông điệp đáng tin cậy hoặc danh sách vòng tròn.

Bảng thành tích trò chơi

Redis là một lựa chọn phổ biến trong số các nhà phát triển trò chơi đang tìm cách xây dựng bảng xếp hạng thời gian thực. Chỉ cần sử dụng cấu trúc dữ liệu Redis Sorted Set, cung cấp tính duy nhất của các phần tử trong khi vẫn duy trì danh sách được sắp xếp theo điểm số của người dùng. Việc tạo danh sách được xếp hạng theo thời gian thực cũng dễ dàng như việc cập nhật điểm của người dùng mỗi khi nó thay đổi. Bạn cũng có thể sử dụng Bộ đã sắp xếp để xử lý dữ liệu chuỗi thời gian bằng cách sử dụng dấu thời gian làm điểm số.

Lưu trữ session

Redis như một kho lưu trữ dữ liệu trong bộ nhớ với tính khả dụng cao và bền bỉ là lựa chọn phổ biến của các nhà phát triển ứng dụng để lưu trữ và quản lý dữ liệu phiên cho các ứng dụng quy mô internet. Redis cung cấp độ trễ dưới mili giây, quy mô và khả năng phục hồi cần thiết để quản lý dữ liệu phiên, chẳng hạn như hồ sơ người dùng, thông tin đăng nhập, trạng thái phiên và cá nhân hóa người dùng cụ thể.

Phát trực tuyến đa phương tiện

Redis cung cấp một kho lưu trữ dữ liệu trong bộ nhớ, nhanh chóng để cung cấp năng lượng cho các trường hợp sử dụng phát trực tiếp. Redis có thể được sử dụng để lưu trữ siêu dữ liệu về hồ sơ của người dùng và lịch sử xem, thông tin xác thực / mã thông báo cho hàng triệu người dùng và tệp kê khai để cho phép CDN truyền video tới hàng triệu người dùng thiết bị di động và máy tính để bàn cùng một lúc.

Không gian địa lý

Redis cung cấp các cấu trúc và toán tử dữ liệu trong bộ nhớ được xây dựng có mục đích để quản lý dữ liệu không gian địa lý theo thời gian thực ở quy mô và tốc độ. Các lệnh như GEOADD, GEODIST, GEORADIUS và GEORADIUSBYMEMBER để lưu trữ, xử lý và phân tích dữ liệu không gian địa lý trong thời gian thực giúp không gian địa lý trở nên dễ dàng và nhanh chóng với Redis. Bạn có thể sử dụng Redis để thêm các tính năng dựa trên vị trí như thời gian lái xe, khoảng cách lái xe và các điểm ưa thích vào ứng dụng của bạn.

Học máy

Các ứng dụng hướng dữ liệu hiện đại yêu cầu máy học để xử lý nhanh chóng một khối lượng lớn, sự đa dạng và tốc độ của dữ liệu cũng như tự động hóa việc ra quyết định. Đối với các trường hợp sử dụng như phát hiện gian lận trong trò chơi và dịch vụ tài chính, đặt giá thầu thời gian thực trong công nghệ quảng cáo và mai mối trong việc hẹn hò và chia sẻ chuyến đi, khả năng xử lý dữ liệu trực tiếp và đưa ra quyết định trong vòng hàng chục mili giây là vô cùng quan trọng. Redis cung cấp cho bạn một kho lưu trữ dữ liệu trong bộ nhớ nhanh chóng để xây dựng, đào tạo và triển khai các mô hình học máy một cách nhanh chóng.

Phân tích thời gian thực

Redis có thể được sử dụng với các giải pháp phát trực tuyến như Apache Kafka và Amazon Kinesis như một kho lưu trữ dữ liệu trong bộ nhớ để nhập, xử lý và phân tích dữ liệu thời gian thực với độ trễ dưới mili giây. Redis là một lựa chọn lý tưởng cho các trường hợp sử dụng phân tích thời gian thực như phân tích phương tiện truyền thông xã hội, nhắm mục tiêu quảng cáo, cá nhân hóa và IoT.

Chương 2

KIẾN THỨC HỆ THỐNG

2.1 Cơ sở lý thuyết của kiến trúc

2.1.1 Kiến trúc Onion

Onion Architecture (Kiến trúc củ hành) được xây dựng dựa trên ý tưởng đặt Domain vào trung tâm ứng dụng, mở rộng cơ chế phân phối (UI) và cơ sở hạ tầng được xây dựng bởi hệ thống.

Các tầng được biểu diễn thành dạng các vòng tròn đồng tâm. Đặc trưng của kiến trúc củ hành đó là một luật về sự ràng buộc giữa các tầng. Tầng bên ngoài chỉ phụ thuộc và gọi trực tiếp tầng bên trong. Các tầng thấp hơn không thể phụ thuộc vào các tầng bên ngoài.

Hầu hết các kiến trúc truyền thống đều nêu lên những vấn đề cơ bản về sự kết hợp giữa các thành phần một cách chặt chẽ và tách biệt. Kiến trúc Onion được Jeffrey Palermo giới thiệu để cung cấp một cách tốt hơn để xây dựng các ứng dụng theo quan điểm về khả năng kiểm tra, khả năng bảo trì và độ tin cậy tốt hơn.

Nguyên lý

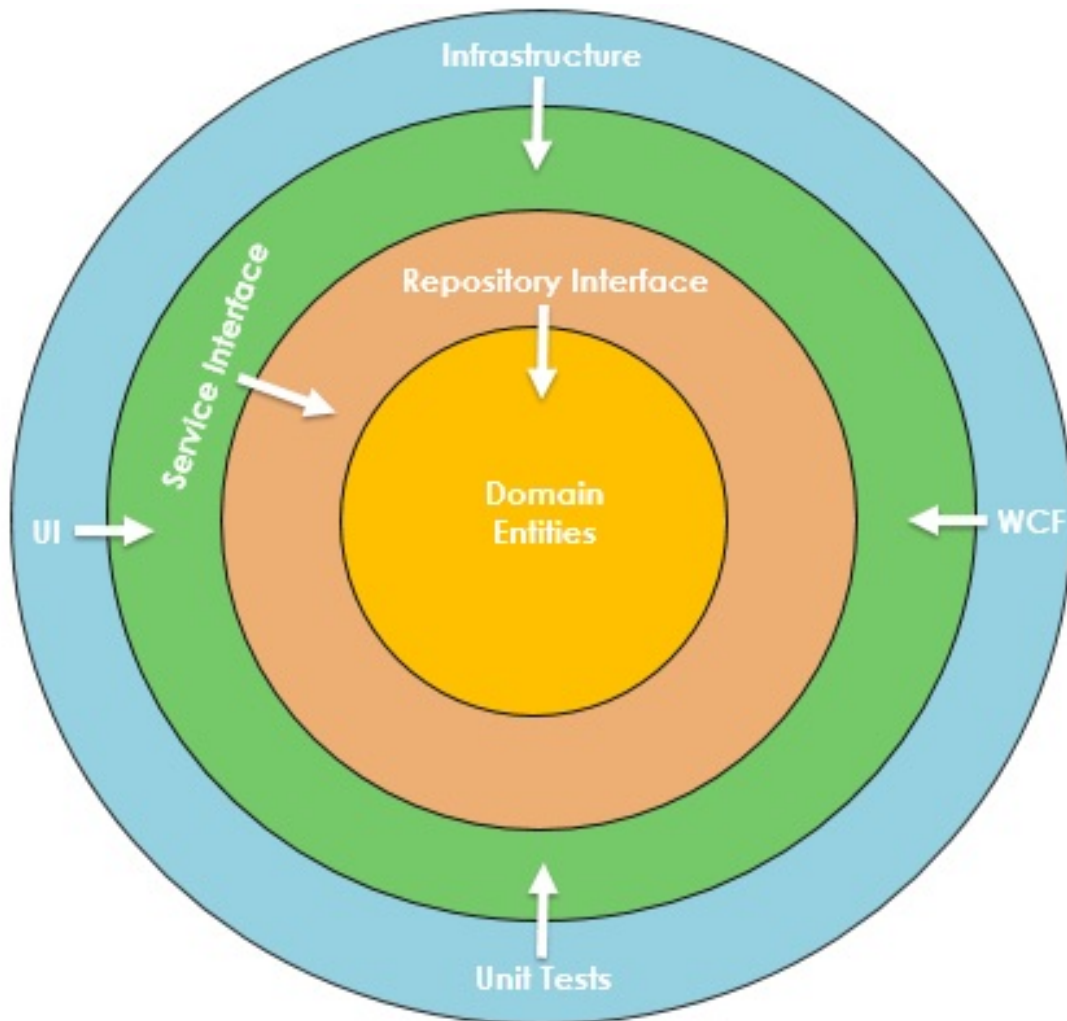
Kiến trúc Onion dựa trên sự đảo ngược của nguyên tắc điều khiển. Kiến trúc Onion bao gồm nhiều lớp đồng tâm giao nhau về phía lõi đại diện cho miền. Kiến trúc không phụ thuộc vào lớp dữ liệu như trong các kiến trúc nhiều tầng cổ điển, mà dựa trên các mô hình miền thực tế.

Theo kiến trúc truyền thống, lớp giao diện người dùng tương tác với logic nghiệp vụ và logic nghiệp vụ nói chuyện với lớp dữ liệu và tất cả các lớp được trộn lẫn và phụ thuộc nhiều vào nhau. Trong kiến trúc 3 tầng và n tầng, không có lớp nào là độc lập. Những hệ thống như vậy rất khó hiểu và khó bảo trì. Hạn chế của kiến trúc truyền thống này là khớp nối không cần thiết.

Kiến trúc Onion đã giải quyết vấn đề này bằng cách xác định các lớp từ lõi đến Cơ sở hạ tầng. Nó áp dụng quy tắc cơ bản bằng cách di chuyển tất cả các khớp nối về phía trung tâm. Kiến trúc này chắc chắn thiên về lập trình hướng đối tượng và nó đặt các đối tượng trước tất cả các đối tượng khác. Trung tâm của Kiến trúc Onion là mô hình miền, đại diện cho các đối tượng hành vi và nghiệp vụ. Xung quanh lớp miền là các lớp khác, với nhiều hành vi hơn.

Các lớp của Kiến trúc Onion

Domain Layer - Lớp tên miền



Hình 2.1: Các lớp của Kiến trúc Onion

Ở phần trung tâm của Kiến trúc Onion, lớp miền tồn tại; lớp này đại diện cho các đối tượng nghiệp vụ và hành vi. Ý tưởng là có tất cả các đối tượng miền của bạn ở cốt lõi này. Nó chứa tất cả các đối tượng miền ứng dụng. Bên cạnh các đối tượng miền, bạn cũng có thể có các giao diện miền. Các thực thể miền này không có bất kỳ phụ thuộc nào.

Repository Layer - Lớp kho lưu trữ

Lớp này tạo ra một sự trừu tượng giữa các thực thể miền và logic nghiệp vụ của một ứng dụng. Trong lớp này, thường thêm các giao diện cung cấp hành vi lưu và truy xuất đối tượng thường bằng cách liên quan đến cơ sở dữ liệu. Lớp này bao gồm mẫu truy cập dữ liệu, là một cách tiếp cận kết hợp lỏng hơn để truy cập dữ liệu. Chúng tôi cũng tạo một kho lưu trữ chung và thêm các truy vấn để truy xuất dữ liệu từ nguồn, ánh xạ dữ liệu từ nguồn dữ liệu sang một thực thể kinh doanh và duy trì những thay đổi trong thực thể kinh doanh đối với nguồn dữ liệu.

Services Layer - Lớp dịch vụ

Lớp Dịch vụ giữ các giao diện với các hoạt động phổ biến, chẳng hạn như Thêm, Lưu, Chính sửa và Xóa. Ngoài ra, lớp này được sử dụng để giao tiếp giữa lớp UI và lớp kho lưu trữ. Lớp Dịch vụ cũng có thể giữ logic nghiệp vụ cho một thực thể. Trong lớp này, các giao diện dịch vụ được giữ với quá trình triển khai của nó, giữ cho sự liên kết lỏng và tách biệt các mối quan tâm trong tâm trí.

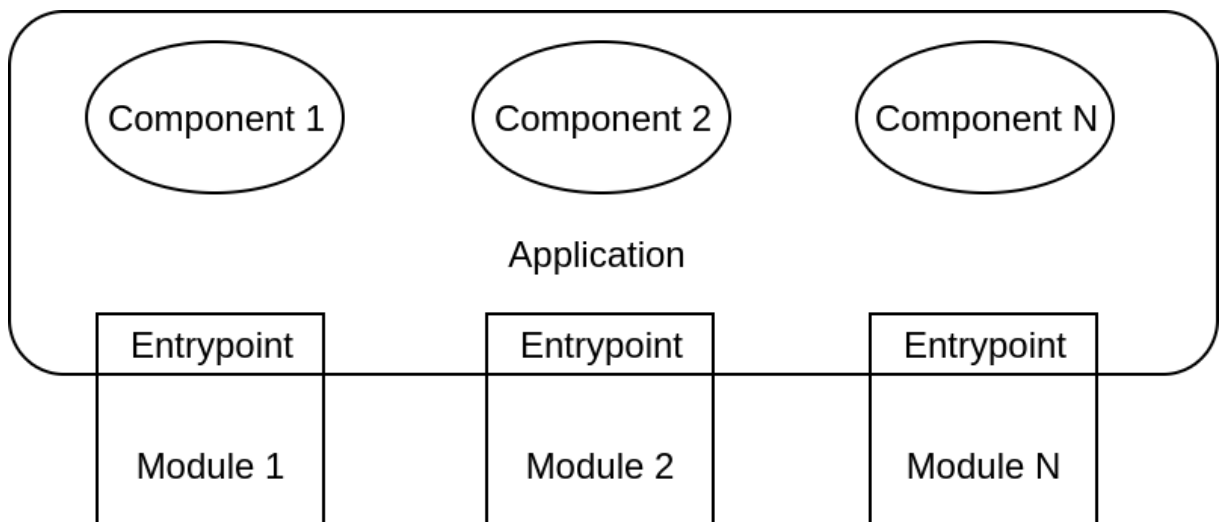
UI Layer - Lớp giao diện người dùng

Đây là lớp ngoài cùng và giữ các mối quan tâm ngoại vi như giao diện người dùng và các thử nghiệm. Đối với một ứng dụng Web, nó đại diện cho dự án Web API hoặc Unit Test. Lớp này có sự triển khai của nguyên tắc tiêm phụ thuộc để ứng dụng xây dựng một cấu trúc liên kết lỏng và có thể giao tiếp với lớp bên trong thông qua các giao diện.

2.1.2 Kiến trúc mô-đun

Kiến trúc mô-đun (Modular Architecture) là kiểu kiến trúc phần mềm cho phép quản lý sự phức tạp của một vấn đề bằng cách chia nhỏ chúng thành các mô-đun để dễ quản lý hơn với các nguyên tắc và mô hình. Mô-đun là một đơn vị phần mềm không trạng thái có thể triển khai, quản lý, tái sử dụng, kết hợp lại và cung cấp giao diện ngắn gọn cho người dùng.

Khi phát triển phần mềm, khi hệ thống càng lớn thì càng cần thêm nhiều component, dẫn tới sự thay đổi nhỏ trong một component có thể ảnh hưởng tới nhiều component khác. Trong hệ thống, module là những component được phát triển bên ngoài ứng dụng. Module giao tiếp với ứng dụng thông qua entry-point.



Hình 2.2: Kiến trúc mô-đun

Sử dụng kiến trúc mô-đun đem lại một số lợi ích:

- **Tùy chỉnh:** ứng dụng có thể hoạt động thực sự khác biệt bằng cách chỉ bật / tắt một số mô-đun.
- **Ít phụ thuộc hơn:** các mô-đun độc lập hơn với chính ứng dụng, cho đến khi các "điểm vào" tương thích, cả mô-đun và ứng dụng có thể phát triển độc lập.
- **Các phần mở rộng của bên thứ ba:** vì các mô-đun không phải là một phần của ứng dụng và các "điểm vào" được xác định rõ, việc phát triển các mô-đun có thể được thực hiện bởi các bên thứ ba.
- **Phát triển độc lập:** Vì ứng dụng và các mô-đun là độc lập, chúng có thể là:
 - được phát triển bởi các nhà phát triển bên ngoài
 - phát hành với các chu kỳ phát hành độc lập

- được phát triển tiềm năng với các công nghệ khác nhau
- **Ứng dụng nhỏ hơn:** Các ứng dụng nhỏ hơn (nhiều chức năng có thể được thực hiện thông qua các mô-đun) và ứng dụng nhỏ hơn được dịch để có khả năng bảo trì tốt hơn.

Mô-đun được hình thành dựa trên việc nhóm những lớp có mức độ liên quan cao thành một mô-đun để có tính tương liên cao nhất. Có 2 loại tương liên:

- Tương liên giao tiếp: có được khi 2 phần của mô-đun thao tác trên cùng dữ liệu.
- Tương liên chức năng: có được khi mọi phần của mô-đun làm việc cùng nhau để thực hiện tác vụ định rõ.

2.2 Kiến trúc tổng quan hệ thống

2.2.1 Hệ thống frontend

Hệ thống Trang

Hệ thống frontend có các hệ thống Trang sau đây: Product, Layer, Experiment và Test Group. Mỗi hệ thống Trang bao gồm có hệ thống Đường dẫn, mỗi Đường dẫn có một nhiệm vụ cụ thể được ghi chú trong Chú thích.

Mỗi thực thể đều có một hệ thống Trang nhằm mục đích cho người dùng có thể khởi tạo, cập nhập và thay đổi theo ý muốn. Ngoài ra người dùng có thể có cái nhìn toàn cảnh với mỗi thực thể.

Trang	Đường dẫn	Chú thích
Product	/product	Hiển thị toàn bộ Product
	/product/new	Khởi tạo một Product mới
	/product/:id	Hiển thị một Product
	/product/:id/update	Cập nhật một Product
Layer	/product/:id/layer	Hiển thị toàn bộ Layer của một Product
	/product/:id/layer/new	Khởi tạo một Layer mới
	/product/:id/layer/:id	Hiển thị một Layer
	/product/:id/layer/:id/update	Cập nhật một Layer
Experiment	/product/:id/layer/:id/exp	Hiển thị toàn bộ Experiment của một Layer
	/product/:id/layer/:id/exp/new	Khởi tạo một Experiment mới
	/product/:id/layer/:id/exp/:id	Hiển thị một Layer
	/product/:id/layer/:id/exp/:id/update	Cập nhật một Layer
Group	/product/:id/layer/:id/exp/:id/group	Hiển thị toàn bộ Test Group của một Experiment
	/product/:id/layer/:id/exp/:id/group/new	Khởi tạo một Test Group mới
	/product/:id/layer/:id/exp/:id/group/:id	Hiển thị một Test Group
	/product/:id/layer/:id/exp/:id/group/:id/update	Cập nhật một Test Group

Bảng 2.1: Hệ thống Trang

Các thành phần

Hệ thống frontend gồm các thành phần sau: Types, Pages, Features.

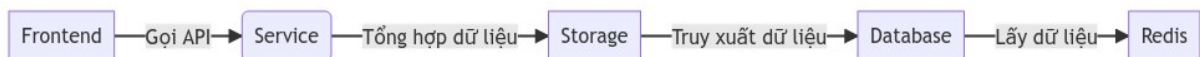
- **Types:** Mục đích là tổng hợp các thực thể ở tầng Frontend, bao gồm hai loại là Thực thể gốc và nội dung khi nhận/gửi API
- **Pages:** Là thành phần xử lý hệ thống Trang, mỗi file sẽ là một hệ thống Trang khác nhau
- **Features:** Là thành phần xử lý hệ thống thông tin cho toàn bộ hệ thống Frontend, bao gồm cả việc nhận, gửi dữ liệu từ Server

2.2.2 Hệ thống backend

Các thành phần

Hệ thống backend gồm các thành phần sau: Service, Storage, Database, Types

- **Service:** Là thành phần xử lý hệ thống HTTP, chuyên nhận và phản hồi các truy cập từ hệ thống Frontend
 - **Summary:** Gửi trả toàn bộ dữ liệu của các A/B Test
 - **Abtest:** Thực hiện và gửi trả A/B Test
 - **Entity:** Thực hiện khởi tạo hoặc cập nhật các thực thể, gửi trả kết quả
- **Storage:** Là thành phần xử lý hệ thống tổng hợp thông tin cho hệ thống Backend
 - **Summary:** Tổng hợp toàn bộ dữ liệu của các A/B Test
 - **Abtest:** Thực hiện A/B Test
 - **Entity:** Khởi tạo hoặc cập nhật các thực thể
- **Database:** Là thành phần xử lý hệ thống quản lý dữ liệu cho hệ thống Backend
 - **Entity:** Khởi tạo hoặc cập nhật các thực thể
- **Types:** Tổng hợp các thực thể ở tầng Backend, bao gồm 2 loại là: Thực thể gốc và nội dung khi nhận/gửi API



Hình 2.3: Hệ thống Backend

Hệ thống đường dẫn

Hệ thống đường dẫn bao gồm các thực thể sau: Summary, AbTest, Product, Layer, Experiment và Test Group.

Mỗi thực thể có chức năng riêng biệt nhau.

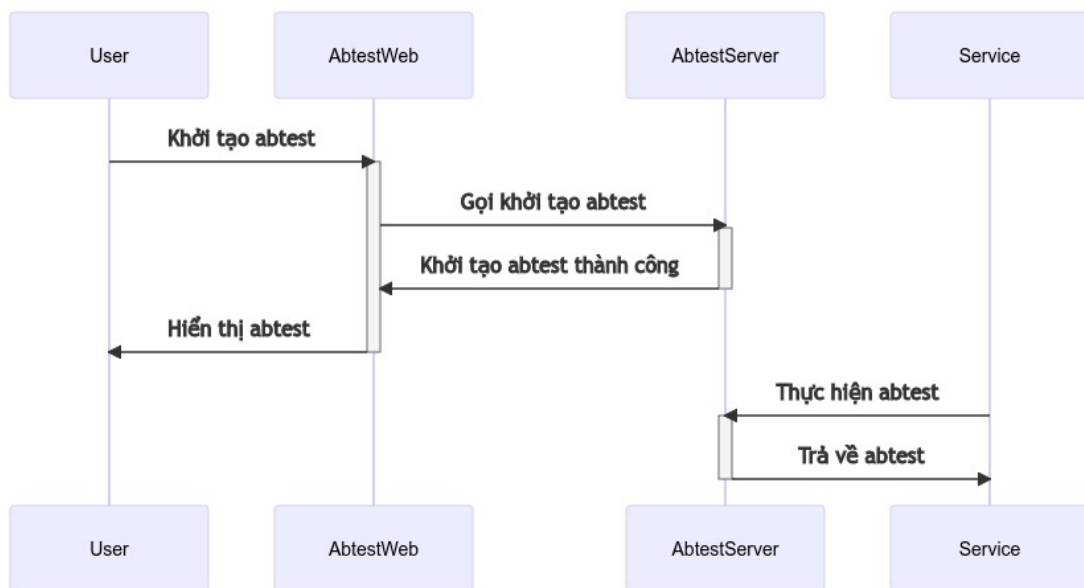
Thực thể	Đường dẫn	Chú thích
Summary	/summary	Gọi toàn bộ thông tin của hệ thống A/B Test
AbTest	/abtest	Thực hiện A/B Test
Product	/product/create	Khởi tạo Product mới
	/product/update	Cập nhật Product
Layer	/layer/create	Khởi tạo Layer mới
	/layer/update	Cập nhật Layer
Experiment	/exp/create	Khởi tạo Experiment mới
	/exp/update	Cập nhật Experiment
Test Group	/group/create	Khởi tạo Test Group mới
	/group/update	Cập nhật Test Group mới

Bảng 2.2: Hệ thống đường dẫn

2.2.3 Biểu đồ tuần tự

Biểu đồ tuần tự chung

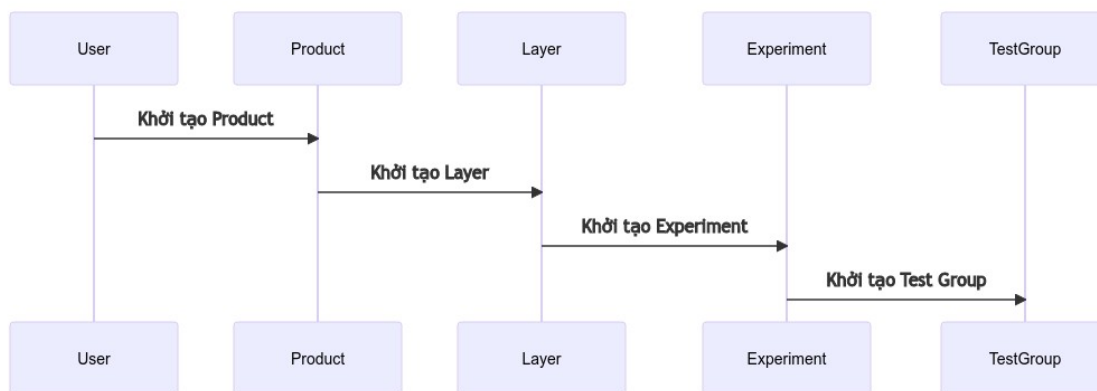
Biểu đồ tuần tự chung biểu thị việc khởi tạo và sử dụng A/B Test.



Hình 2.4: Biểu đồ tuần tự chung

Biểu đồ tuần tự khởi tạo A/B Test

Biểu đồ tuần tự khởi tạo A/B Test biểu thị



Hình 2.5: Biểu đồ tuần tự khởi tạo A/B Test

2.3 Mô hình cơ sở dữ liệu

Mô hình cơ sở dữ liệu của hệ thống mô tả cụ thể các đặc tính của mỗi thực thể, ngoài ra mô tả cách lưu trữ các thực thể và các mối quan hệ tương quan trong cơ sở dữ liệu.

2.3.1 Các thực thể

Các thực thể bao gồm: Product, Layer, Experiment và Test Group.

Product

Product là thực thể lớn nhất, chỉ có chức năng phân tách theo nhu cầu sử dụng nên chỉ có 2 trường như sau:

Trường	Loại	Chú thích
id	int	Unique id
name	string	Tên của Product

Bảng 2.3: Thuộc tính của Product

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** product::[id]
- **Value:** binary (nén bởi protobuf)
- **Command:** SET, GET

Layer

Trường	Loại	Chú thích
id	int	Unique id
name	string	Tên của Layer
type	int	Loại của Layer

Bảng 2.4: Thuộc tính của Layer

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** layer::[id]
- **Value:** binary (nén bởi protobuf)
- **Command:** SET, GET

Experiment

Trường	Loại	Chú thích
id	int	Unique id
name	string	Tên của Experiment
traffic	int	Số lượng traffic
start_time	int	Thời điểm bắt đầu
end_time	int	Thời điểm kết thúc
status	int	Trạng thái

Bảng 2.5: Thuộc tính của Experiment

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** exp::[id]
- **Value:** binary (nén bởi protobuf)
- **Command:** SET, GET

Test Group

Trường	Loại	Chú thích
id	int	Unique id
parameter	string	Parameter của Test Group
value	string	Value của Parameter

Bảng 2.6: Thuộc tính của Test Group

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** group::[id]
- **Value:** binary (nén bởi protobuf)
- **Command:** SET, GET

2.3.2 Môi quan hệ giữa các thực thể

Giữa các thực thể có các mối quan hệ lẫn nhau, vì vậy ngoài việc lưu trữ các thực thể dưới cơ sở dữ liệu, cũng cần lưu trữ mối quan hệ giữa chúng.

Tổng hợp các Product

Khi cần query tất cả các Product hiện có, chỉ cần lưu các id của product vào một chỗ. Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** product::ids
- **Value:** các id của product
- **Command:** SADD, SMEMBERS

Giữa Product và Layer

Mỗi Product có nhiều Layer, khi cần query tất cả các Layer dưới một Product, chỉ cần lưu các id của layer vào một chỗ.

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** product::[id]::layers
- **Value:** các id của Layer
- **Command:** SADD, SMEMBERS

Giữa Layer và Experiment

Mỗi Layer có nhiều Experiment, khi cần query tất cả các Experiment dưới một Layer, chỉ cần lưu các id của Experiment vào một chỗ.

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** layer::[id]::exps
- **Value:** các id của Experiment
- **Command:** SADD, SMEMBERS

Giữa Experiment và Test Group

Mỗi Experiment có nhiều Test Group, khi cần query tất cả các Test Group dưới một Experiment, chỉ cần lưu các id của Test Group vào một chỗ.

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** exp::::groups
- **Value:** các id của Test Group
- **Command:** SADD, SMEMBERS

2.3.3 Id generator

Mỗi thực thể đều yêu cầu một mã ID định danh, do đó cơ sở dữ liệu cũng cần được sử dụng để mỗi khi tạo một thực thể mới có thể có một mã ID định danh mới.

Cấu hình khi được lưu xuống cơ sở dữ liệu:

- **Key:** product::id, layer::id, exp::id, group::id
- **Value:** Mã ID đang sử dụng hiện tại
- **Command:** INCR

Chương 3

TRIỂN KHAI VÀ THỰC NGHIỆM

3.1 Ứng dụng Domain Driven Design

3.2 Triển khai chi tiết Use case

3.2.1 Use case khởi tạo A/B Test

Bước 1: Xác định ngữ cảnh

- Phía frontend gồm có: page để hiển thị và khởi tạo các thực thể
- Phía backend gồm có: Service, Storage, Database

Bước 2: Xác định các trường dữ liệu

- Product: tên product
- Layer: tên layer, loại của layer
- Experiment: tên experiment, số lượng traffic, ngày bắt đầu và ngày kết thúc
- Test Group: tên của parameter, giá trị của parameter

Bước 3: Xử lý nghiệp vụ

- Nhập dữ liệu trên website
- Frontend gọi API của Service
- Service gọi hàm tạo thực thể của Storage
- Storage gọi hàm tạo thực thể của Database
- Database thực hiện khởi tạo dữ liệu trên Cơ sở dữ liệu
- Cơ sở dữ liệu thực hiện thành công và gửi trả dữ liệu cho Database
- Database gửi trả dữ liệu cho Storage

- Storage gửi trả dữ liệu cho Service
- Service gửi trả dữ liệu cho Frontend
- Frontend thông báo thành công cho người dùng

3.2.2 Use case sử dụng A/B Test

Bước 1: Xác định ngữ cảnh

- Phía frontend gồm có: page để hiển thị và khởi tạo các thực thể
- Phía backend gồm có: Service, Storage, Database

Bước 2: Xác định các trường dữ liệu

- Product: Id của product
- UserID: Id của người dùng
- SessionID: session của request

Bước 3: Xử lý nghiệp vụ

- Từ phía người dùng, thực hiện một request bất kỳ đến abtest API của Service
- Service gọi hàm tạo thực hiện abtest của Storage
- Storage gọi truy xuất thông tin của Product từ Database
- Database thực hiện truy xuất những dữ liệu cần trên Cơ sở dữ liệu
- Cơ sở dữ liệu thực hiện thành công và gửi trả dữ liệu cho Database
- Database gửi trả dữ liệu cho Storage
- Storage thực hiện abtest dựa trên những dữ liệu đang có và trả về cho Service
- Service gửi trả dữ liệu cho người dùng

3.3 Triển khai kiểm thử thích hợp

Việc kiểm thử phần mềm là quan trọng vì những lý do sau:

- Kiểm thử chỉ ra sai sót của sản phẩm trong giai đoạn phát triển, đồng thời đảm bảo chất lượng sản phẩm trong tương lai.
- Kiểm thử đảm bảo kết quả cuối cùng đáp ứng các yêu cầu kinh doanh và người sử dụng.

Unit testing và integration testing:

Unit testing	Integration testing
Kiểm thử đơn vị là để kiểm tra từng phần của module riêng lẻ và quan sát rằng các bộ phận riêng lẻ đang hoạt động như mong đợi.	Kiểm thử tích hợp là tích hợp tất cả các module vào ứng dụng và kiểm tra chúng như một nhóm để xem chúng có hoạt động như mong đợi hay không
Kiểm thử đơn vị còn được gọi là kiểm thử hộp trắng vì nó đòi hỏi kiến thức về mã code và luồng điều khiển trong chương trình.	Kiểm thử tích hợp còn được gọi là kiểm thử hộp đen, chúng ta không động đến mã code mà chỉ tập trung vào đầu vào đã cho và đầu ra mong đợi.
Unit testing có thể được thực hiện bất cứ lúc nào nhưng luôn luôn thực hiện trước khi kiểm thử tích hợp.	Integration Testing được thực hiện sau khi kiểm thử đơn vị nhưng trước khi kiểm thử hệ thống.
Unit Testing chỉ kiểm tra chức năng của các module ở cấp độ đơn vị và không thể bắt được các lỗi mức tích hợp hoặc bất kỳ vấn đề nào trong toàn hệ thống.	Các lỗi trong kiểm thử tích hợp được phát hiện sau khi các module được tích hợp để xây dựng hệ thống tổng thể.
Chủ yếu tập trung vào chức năng của một module riêng lẻ.	Chủ yếu tập trung vào tích hợp các module.
Unit Testing thường được thực hiện bởi các developers.	Integration Testing được thực hiện bởi người kiểm thử(Tester).
Lỗi phát hiện thường đơn giản khi thử nghiệm đơn vị.	Lỗi phát hiện thường khó khăn khi thử nghiệm tích hợp.
Chi phí bảo trì kiểm thử đơn vị là rất ít vì nó được duy trì và mức đơn vị.	Kiểm thử tích hợp bảo trì là khá tốn kém vì nó đòi hỏi phải thiết lập môi trường riêng biệt.
Kiểm thử đơn vị không xác minh xem mã code có hoạt động như mong đợi với các phụ thuộc bên ngoài hay không.	Kiểm thử tích hợp giúp xác minh rằng mã code hoạt động như mong đợi với các phụ thuộc bên ngoài.

Quy trình kiểm thử gồm 5 bước triển khai: Khởi tạo môi trường, chuẩn bị dữ liệu người dùng, gọi API, kiểm tra dữ liệu trả về, xóa môi trường.

3.3.1 Mô tả Integration test với một service cụ thể

Sau đây sẽ thực hiện thử nghiệm với việc khởi tạo một A/B Test.

- Chuẩn bị môi trường
- Xác định danh sách các API cần test
 - Khởi tạo Product mới
 - Khởi tạo Layer mới
 - Khởi tạo Experiment mới
 - Khởi tạo Test Group mới
- Chuẩn bị các testcase bao gồm:

- API cần test
- Dữ liệu đầu vào
- Dữ liệu đầu ra mong muốn
- Kết quả test

Viết code cho từng ca kiểm thử theo tập ca kiểm thử được chuẩn bị. Bảng ca kiểm thử được trình bày file ngoài.

Xóa trắng cơ sở dữ liệu sau mỗi vòng test.

3.4 Thực nghiệm

Chương 4

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Trong chương cuối cùng, chúng tôi ngắn gọn kết luận kết quả đạt được của khóa luận, những hạn chế khi thực hiện đề tài và những hướng phát triển sắp tới.

4.1 Kết quả đạt được

Đề tài dự án đã tiến hành nghiên cứu về quy trình nghiệp vụ, các công nghệ và công cụ, đặc biệt là Domain Driven Design vào xây dựng hệ thống Abtest trong Quảng Cáo.

Việc thực hiện đề tài dự án giúp hiểu sâu hơn về nghiệp vụ quảng cáo, cách thức phân tích hệ thống và triển khai thực hiện các công nghệ cùng nhau.

4.2 Hạn chế khi thực hiện đề tài

Thứ nhất là do thiếu kiến thức quảng cáo nên khi triển khai nghiệp vụ còn nhiều sai sót và sửa đổi nhiều lần.

Thứ hai là do thời gian có hạn nên các module được cài đặt mới chỉ ở mức sơ khai. Còn tồn tại các hạn chế như chưa thực hiện việc quản lý kết quả abtest để thể hiện đúng chức năng nhất của một hệ thống abtest.

4.3 Hướng phát triển

Phát triển chức năng quản lý kết quả abtest để thành một sản phẩm trong hệ thống quảng cáo hoàn chỉnh.

Áp dụng các module đã xây dựng cho các hệ thống khác ngoài hệ thống abtest.