# Giới thiệu

* VNTT
* Dịch vụ Cloudbackup:

## Hiện trạng ứng dụng portal

Định nghĩa

* Dashboard là bảng gồm các hàng và các cột. Mỗi ô trong bảng là tổng hợp trạng thái của một một tài khoản manager (admin hoặc backupset) trên cloudbackup trong một thời gian cố định (một ngày hoặc một giờ). Các ô được hiển thị với màu sắc khác nhau tương ứng với trạng thái của tài khoản (backupset) đó trong đơn vị thời gian. Những trạng thái này được xét theo thứ tự ưu tiên nếu có nhiều backupjob trong cùng đơn vị thời gian. Mỗi dòng trong dashboard tương ứng với một tài khoản(manager, admin) hoặc backupset.
* Dashboard được thể hiện ở 2 dạng: dạng theo ngày (được sử dụng cho hầu hết các tài khoản trên cloudbackup gồm: manager, admin, vnttadmin, vnttdirector) và dạng trình bày theo giờ (được sử dụng cho tài khoản vnttagent).
* Dashboard là một tính năng mới, chưa xuất hiện trong console. Dashboard cho phép user có cái nhìn tổng quát về trạng thái của tài khoản (manager, admin) hoặc theo dõi trạng thái của khách hàng (vnttadmin, vnttagent, vnttdirector). Đây là điều rất quan trọng đặc biệt là đối với user mới vì có thẻ theo dõi được hoạt động của tài khoản. Đồng thời, công ty VNTT cũng có thể theo dõi tình trạng sử dụng của khách hàng một cách dễ hàng để có thể hỗ trợ nhanh chóng, nâng cao chất lượng dịch vụ.

Các chức năng

* Hiển thị trạng thái tổng hợp theo ngày (giờ) của user hiện hành (áp dụng đối với manager và admin).
* Hiển thị trạng thái tổng hợp theo ngày (giờ) của toàn bộ tài khoản khách hàng (áp dụng đối với vnttadmin, vnttagent và vnttdirector).
* Hiển thị popup tổng hợp thông tin khi rê chuột vào các ô trên dashboard (dữ liệu theo ngày hoặc giờ).
* Hiển thị popup tổng hợp thông tin khi rê chuột vào các ô header trên dashboard (dữ liệu theo ngày hoặc giờ).
* Hiển thị popup thông tin tài khoản, backupset khi rê chuột vào ô tương ứng.
* Tạo link liên kết đến các trang setting, backupset, report…
* Cho phép collapse và expand để xem chi tiết.

## Các vấn đề

* Lượng dữ liệu lớn: Với mỗi tài khoản manager sẽ có nhiều tài khoản admin, mỗi tài khoản admin lại có nhiều backupset. Do đó số lượng backupset trong mỗi tài khoản admin là khá lớn. Trong một tháng( hoặc một ngày) được hiển thị trên dashboard, số lượng các dòng(record) lưu thông tin của backup job là rất lớn.

Ví dụ: với 1 tài khoản manager có 5 tài khoản vguard pro và 5 tài khoản vguard lite, mỗi tài khoản có 5 backupset (riêng vguard lite chỉ có 1 backupset). Số lượng backupset là: 30 backupset. Mỗi backupset này có thể thực hiện nhiều lần backup dữ liệu. Cụ thể theo thiết kế, có thể lập lịch cho backupset thực hiện backupset mỗi 2 phút. Tuy nhiên, với điều kiện mỗi backupset thực hiện backup 1 lần/4h thì số backupjob là 180 backupjob.

Theo trung bình, nếu có 20 user thì số backupjob mỗi ngày là 180 x 20 = 36000 backupjob, một con số rất lớn.

* Tốc độ obs: Thời gian trả lời của obs khi truy xuất các record là khá lâu và API của Obs chỉ hỗ trợ truy xuất backup Job trên từng tài khoản admin vào một ngày xác định. Ngoài ra các thông tin cung cấp bởi chức năng truy xuất trên không thể cung cấp đầy đủ thông tin cho các backup job mà đòi hỏi với mỗi backupjob cần có một câu truy vấn obs khác để lấy thông tin. Tất cả dẫn đến thời gian truy vấn trực tiếp trên Obs để lấy dữ liệu về backupJob là rất dài.

Cụ thể: nếu muốn truy cập danh sách backupjob của một tài khoản admin trong một tháng thì phải thực hiện 30 truy vấn đến obs (giả định tháng có 30 ngày). Mỗi backupjob trong một ngày cần thực hiện thêm 2 lần truy vấn obs để có thông tin đầy đủ (truy vấn theo ngày chỉ trả về status) như danh sách files, tổng dung lượng thay đổi, thời gian bắt đầu, kết thúc… do đó, nếu có để thực hiện truy vấn được 36000 backupjob thì sẽ tốn rất nhiều thời gian, không thể đáp ứng nếu truy xuất trực tiếp trên obs.

* Vấn đề API: API của ashay cung cấp backupset được tạo theo thời gian (ID tính theo millisecond từ 1/1/1970) nên vẫn có thể trùng ID backupset. Một khó khăn khác là cập nhật dữ liệu từ obs để lưu trữ trên database, trong một số lần chạy demo thì xuất hiện có những backupjob cùng ID của cùng backupset nhưng khác status. Hai backupjob trên có một status là missed. Do đó, việc cập nhật dữ liệu và xác nhận, kiểm tra để thêm record vào database gặp khó khăn.
* Vấn đề về cập nhật: Do thời gian chạy backupjob trên các user là khác nhau, có thể có những backupjob chạy rất lâu, đăc biệt là chạy lần đầu khi mới tạo backupset. Do không có thông tin về lần cuối cùng chạy backupjob nên phải update từ lúc có backup chưa hoàn tất. Tuy nhiên số lượng backupjob lại rất lớn nên nếu chạy lại nhiều ngày sẽ rất chậm nên cần xác định thời gian giới hạn để cập nhật.
* Sự thay đổi: API của cloudbackup cung cấp chức năng rất hạn chế để có thể cập nhật dữ liệu backupjob. Việc cập nhật dữ liệu chỉ có thể thực hiện thông qua nhiều truy vấn nên thời gian rất chậm. Đồng thời, các backupjob có trạng thái là chưa hoàn thành phải đượ cập nhật lại. Có thể những backupjob kéo dài sẽ dẫn đến việc phải cập nhật dữ liệu lại nhiều ngày. Do đó, cần phải xác định một giới hạn thời gian để có thời tránh trường hợp backupjob không kết thúc hoặc kéo dài quá lâu.
* Tần suất xem: Các thông tin về tài khoản (manager, admin) , backupset được xem với tần số thấp và có thể được thay đổi trong một phiên nên không được lưu trữ trên web mà được truy vấn với mỗi khi có yêu cầu xem từ phí người dung.
* Vấn đề bất đồng bộ

## Giải pháp

Kiểm soát sự thay đổi

Cache (Database, Server, Web)

Bất đồng bộ + Predict + Detect tác vụ không có chủ đích + Loading stutus + Error

Build Tree

Build Table

Build Popup

## Làm việc nhóm

* Team Foudation

# Kết quả

## Đạt được

## Hạn chế

# Module 2: Tổng quát hóa phân tích dữ liệu

1. Định nghĩa

Các dữ liệu trả về ban đầu là khối lượng lớn dữ liêu thô, không có tổ chức và được lấy từ nhiều nguồn khác nhau, do đó cần một quá trình phân tích, biến đổi tổng quát các dữ liệu này thành các đối tượng để sử dụng một cách thuận tiện nhất trong các tác vụ khác liên quan đến dữ liệu.

1. Vấn đề

* Toàn bộ dữ liệu của chương trình được truy xuất thông qua các Obs server, dữ liệu gồm thông tin về users, backup set, backup job và các thông tin về hệ thống. Tuy nhiên các thông tin này được chia làm các API khác nhau và các services được viết trên các API tạo nên 32 services khác nhau. Bên cạnh số lượng services lớn mỗi service lại có các thông số đầu vào (paramaters) riêng và trả về các dữ liệu riêng. Dẫn đến việc tạo riêng các paramaters và các objects khác nhau để cho từng service sẽ tốn một chi và thời gian lớn. Ngoài ra, điều này sẽ làm quá trình kiểm tra tính đúng đắn dữ liệu khó khăn và khó kiểm soát trong các tác vụ liên quan đến dữ liệu.
* Dữ liệu các servers Obs cung cấp ở dạng dữ liệu thô XML với số lượng thuộc tính lớn, do đó việc này sẽ gây khó khăn trong quá trình kiểm tra lại dữ liệu khi truy xuất. Ngoài ra cấu trúc XML khá phức tạp gồm nhiều tập hợp và đối tượng ở các node phía trong và mỗi API lại trả cấu trúc khác do đó không thể áp dụng được các công nghệ serialization và deserialization XML của C#.
* Ngoài những khó khăn trên, trong dữ liệu trên Obs có một số dữ liệu cần được truy xuất thường xuyên, nếu không được xứ lý thích hợp chi phí và thời gian bỏ trong quá trình truy xuất lên Obs sẽ gây ra ảnh hưởng lớn tốc độ của chương trình.

1. Phân tích

* Tuy dữ khối lượng các thông số đầu vào và dữ liệu trả về của service rất lớn nhưng nó có nhiều thuộc tính chung và có một số service lấy dữ liệu cho chung một đối tượng.
* Dữ liệu XML trả về phức tạp nhưng cấu trúc nó chỉ lặp lại ở 1 số dạng cấu trúc nhất định, có thể xử lí linh hoạt để đọc dữ liệu.
* Các dữ liệu thường xuyên được sử dụng nhưng không phải lúc nào thông tin đó cũng thay đổi nên ta có thể sử dụng một số kĩ thuật để kiểm tra và truy xuất khi cần thiết.

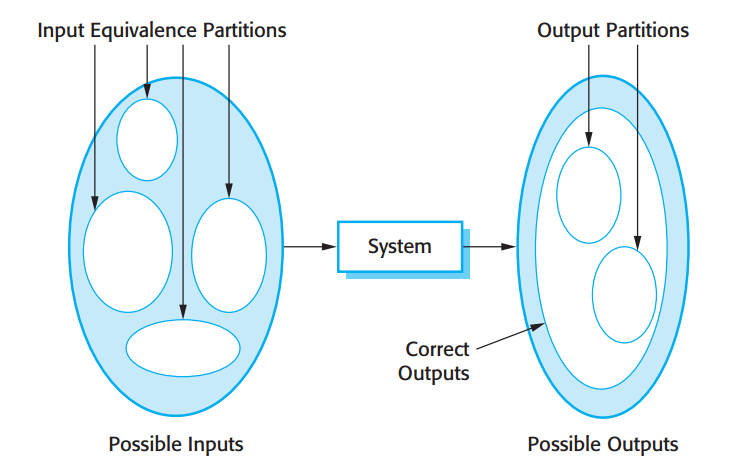
1. Giải pháp

* Phân tích tổng quát thành các đối tượng chính để dùng chung các services có các dữ liệu trả về chung cho một đối tượng,ví dụ backup set, user profile,… . Tương tự với dữ liệu đầu ta sẽ tạo một lớp gồm các đối tượng chung và các lớp khác sẽ thừa kế lại những lớp chung này. Tuy nhiên khi tạo các lớp này số lượng các thuộc tính vẫn còn lớn để đảm bảo chất lượng không sai xót khi tạo các lớp này, có một giải pháp khá hữu ích là viết một chương trình đơn giản để tự động lọc ra các tên thuộc tính của các API do Obs cung cấp và chuyển thành định dạng thuộc tính trong C#. Việc này giảm thời gian đáng kể và chất lượng các tên thuộc tính đảm bảo.
* Sau khi phân tích các dữ liệu XML gồm ba cấu trúc chính Object, Collection,Primitive và kết hợp của chúng, do đó ta tách ra làm 3 hàm con để xử lý riêng cho các dạng dữ liệu. Các generic method được viết để tăng tính linh hoạt phù hợp với tất cả các dữ liệu trả về. Bên cạnh đó một số kĩ thuật về tạo đối tượng bằng Activator Class để giải quyết vấn đề của các Objects và Collections.
* Các dữ liệu cần thiết ví dụ như thông tin Users cần được update kiểm tra liên tục, do đó chúng ta sẽ tạo ra cache để lưu lại danh sách users và chúng ta sẽ phân tích tính toán thời gian hợp lý để update lại danh sách. Do đó danh sách này chỉ được cập nhật khi User được kiểm tra chưa nằm trong danh sách hoặc sau khoảng thời gian nhất định. Ngoài ra, chúng ta có một phương thức force update để dành cho một số trường hợp như tạo tài khoản.

Unit Test.

1. Định nghĩa.

Unit Test là quá trình kiểm thử các thành phần của chương trình như các phương thức, các lớp đối tượng hoặc các đơn vị khác trong code. Unit Test sẽ giúp developers có khả năng để xác định chức năng của họ hoạt động đúng như mong đợi.



1. Lợi ích của Unit Test.

* Unit Test giúp xác định giải thuật, logic không đúng hoặc các lỗi trong quá trình lập trình từ sớm để sửa chữa và nâng cao chất lượng code.
* Unit Test giúp có thể kiểm tra lại code một cách nhanh chóng và chính xác khi thay đổi một phần nhỏ trong chương trình.
* Chi phí và thời gian hiện thực phần mềm giảm, bên cạnh đó chất lượng code được nâng cao và giảm rủi ro của từng chức năng một trong phần mềm.

1. Unit Test trong Dashboard.

Dữ liệu của dashboard được tổng hợp từ nhiều dữ liệu khác nhau trên server trả về, bên cạnh đó chức năng chính của dashboard là hiển thị chi tiết các dữ liệu tổng hợp cho khách hàng cũng như admin. Do đó, các thông tin này cần được đảm bảo chính xác khi lấy từ server và trong các phương thức tổng hợp.

* Tất cả các đối tượng lưu trữ thông lấy từ server và các phương thức xử lý dữ liệu đều phải thực hiện Unit Test.

1. Công việc thực hiện.

* Xác định và thiết kế các trường hợp input testcases có thể xảy ra và kết quả mong đợi của các testcase đó trước khi lập trình chi tiết các chức năng chính.
* Viết tất cả các unit test theo các thiêt kế trên, có thể bổ sung trong quá trình hiện thực các chức năng chính.
* Chạy test cases thường xuyên trong quá trình hiện thực và trong quá trình thay đổi và chỉnh sửa các chi tiết trong chương trình.