**Báo cáo thực tập**

**Produced for:**

**Produced by:**

Table of Contents

1 Giới thiệu 5

1.1 VNTT 5

1.2 Dịch vụ Cloudbackup: 5

1.3 Hiện trạng ứng dụng portal 5

2 Module 1: Dashboard 6

2.1 Định nghĩa 6

2.2 Các chức năng 7

2.3 Các vấn đề 8

2.4 Giải pháp 8

2.5 Build Table 9

2.6 Build popup 9

2.6.1 Thông báo loading: 9

Vấn đề 9

Giải pháp 9

2.6.2 Tối ưu vị trí hiện thị popup: 9

Vấn đề 9

Giải pháp 9

3 Module 2: Tổng quát hóa phân tích dữ liệu 11

4 Làm việc nhóm 16

5 Kết quả 17

5.1 Đạt được 17

5.2 Hạn chế 17

Table of Figures

# Giới thiệu

## VNTT

## Dịch vụ Cloudbackup:

vGuard Cloud Backup (Online Backup) là dịch vụ sao lưu dữ liệu trong máy tính KH (server, desktop, laptop) về lưu trữ tại Data Center VNTT thông qua kết nối Internet. Khi gặp sự cố mất mát/hư hỏng dữ liệu, KH có thể phục hồi lại từ các bản sao đang được lưu trữ tại Data Center VNTT.

KH (KH) sử dụng dịch vụ vGuard Cloud Backup theo cách như sau:

* Mỗi KH được cung cấp một (hoặc nhiều) tài khoản dịch vụ. Mỗi tài khoản có giới hạn dung lượng xác định (50GB, 100GB, 500GB,…)
* KH cài đặt và cấu hình phần mềm (vGuard Lite cho laptop/desktop, vGuard Pro cho server) trên máy tính cần sao lưu
* Đến thời điểm trong lịch sao lưu, phần mềm tự động nén, mã hóa và gửi dữ liệu về Data Center VNTT qua Internet. Bên cạnh sao lưu tự động, phần mềm cũng cho phép KH sao lưu thủ công. Để quản lý tài khoản và quá trình sao lưu, KH truy cập Web-console tại địa chỉ <https://sv.vguard.vn>
* Để phục hồi dữ liệu, KH sử dụng phần mềm để lấy lại, hoặc phục hồi qua web tại Web-console

## Hiện trạng ứng dụng portal

* Portal được bổ sung một loại tài khoản mới là tài khoản manager. Các tài khoản cũ được sử dụng là tài khoản admin. Tài khoản manager là tài khoản quản lý tất cả các tài khoản admin của một khách hàng (một khách hàng có thể có nhiều tài khoản admin), được quyền cấp phát dung lượng cho các tài khoản admin (dựa trên dung lượng tổng của tài khoản), tạo mới tài khoản admin, xóa tài khoản admin… Tài khoản này được tạo bởi VNTT.
* Tài khoản vnttadmin là tài khoản được tạo sẵn để VNTT có thể tạo khoản manager mới cho khách hàng.
* Tài khoản vnttagent là tài khoản dùng cho nhân viên VNTT có thể quan sát hoạt động của hệ thống cloudbackup qua dashboard dạng giờ.
* Tài khoản vnttdirector tương tự tài khoản vnttadmin nhưng có quyền tạo ra các tài khoản VIP.
* Trước khi bắt đầu xây dụng dashboard thì các tính năng về tạo tài khoản (manager, admin), các tính năng về backupset, các nội dung và report được hoàn thành.
* Tuy nhiên portal vẫn chưa có trang chủ. Do đó, dashboard được xây dụng như trang chủ của tất cả các loại tài khoản. Đồng thời, dashboard đóng vai trò như một bản tổng hợp để người dùng kiểm tra sự hoạt động của hệ thống cloudbackup một cách tiện lợi, nhanh chóng.

# **Dashboard**

## **Định nghĩa**

* Dashboard là bảng gồm các hàng và các cột.
  + Mỗi ô trong bảng là tổng hợp trạng thái của một tài khoản manager (admin hoặc backupset) 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: tổng hợp dữ liệu theo ngày, được hiển thị thành một tháng được chọn (manager, admin, vnttadmin, vnttdirector)
  + Dạng theo giờ: tông hợp dữ liệu theo giờ, được hiển thị thành một ngày tính từ thời điểm hiện hành, dạng này được sử dụng bởi tài khoản vnttagent.

Hình 1 Dashboard dạng ngày



Hình 2 Dashboard dạng giờ

* 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.
* Cho phép thay đổi thứ tự manager của dashboard với dạng giờ.
* Cho phép chọn tháng hiển thị (dạng ngày) hoặc thay đổi ngày dạng giờ).

## Các vấn đề

* Lượng dữ liệu lớn: Số backupset trong một tài khoản manager là

Trong đó:

n là số lượng account vguard pro

m là số lượng account vguard lite (mỗi tài khoản chỉ có tối đa một backupset).

ai là số backupset của tài khoản vguard pro thứ i

Với n = 5, ai = 5 ∀i, m = 5. Ta có S = 30.

Với 200 manager thì có 30 x 200 backupset = 6000 backupset.

Nếu mỗi backupset thực hiện backup mỗi 2h thì số backupjob là 6000 x 12 = 72000 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.
* 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:
* Bắt đầu cập nhật từ ngày đầu tiên có backupjob chưa hoàn thành.
* Nếu ngày này trước thời gian hiện tại quá 7 ngày thì sẽ bắt đầu lúc 7 ngày trước ngày hiện tại.
* Nếu không có backupjob nào chưa hoàn thành thì sẽ cập nhật từ ngày hiện tại.
* Cache (Database, Web):
* Database: Lưu trữ toàn bộ dữ các record backupjob trên database, do đó giảm số lượng truy vấn trên obs.
* Web: Dữ liệu tổng hợp của các ô được cache lại trên web để giảm thời gian truy vấn, tổng hợp theo ngày (giờ) của toàn bộ các admin khi được yêu cầu.

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

Thay đổi thứ tự manager: Dùng lệnh update trong SQL để cập nhật dữ liệu. Dựa trên các giá trị vị trí cũ và mới để cập nhật vị trí mới. Đồng thời, dựa trên số lượng tài khoản manager để có thể sửa lỗi các trường hợp nhập sai của người dùng (nhập số lớn hơn số tài khoản manager hoặc nhâp số bé hơn 1).

## Trải nghiệm người dùng

### Build Tree

**Mục đích**: cho phép người dùng đóng/mở các node để có thể xem chi tiết/ tổng quát hoạt các hoạt động trên cloudbackup.

## Dựa trên cấu trúc phân cấp từ cao đến thấp lần lượt là manager, admin, backupset. Khi đó sử dụng javascript và jquery để xử lý việc đóng mở các node trên cây. Với những nút đang mở thì ẩn tất cả các con (level thấp hơn) của nó. Đối với những nút đang đóng thì hiện tất cả những con của nó.

### Build popup

Mục đích: hiển thị thông tin về backup job.

Vấn đề: cần xây dựng một popup hiển thị đầy đủ thông tin cần thiết. Tuy nhiên cũng cần phải tối ưu về mặt truy vấn dữ liệu để cải thiện tốc độ đáp ứng người dùng từ đó cần phải giải quyết một số vấn đề.

### Thông báo loading:

Vấn đề: khi người sử dụng truy vấn nhưng hệ thống không thể đáp ứng ngay thì cần phải hiển thị thông báo đang tải cho người dùng. Tuy nhiên nếu thời gian đáp ứng nhỏ (dưới 3 giây) thì không nên hiển thị vì sẽ gây cảm giác hệ thống chậm chạp. Ngược lại, nếu đã đủ thời gian để hiện thị thì cần phải hiện thị ít nhất một khoản thời gian nhất định vì nếu không sẽ gây hiện tượng khó chịu cho người dùng.

Giải pháp:

### Tối ưu vị trí hiện thị popup:

Vấn đề: khi hiển thị popup cho người dùng, jQuery hỗ trợ 4 tùy chọn hiển thị cơ bản là top, bottom, left, right. Tuy nhiên nếu như vị trí hiện thị gần các biên (trên, dưới, trái, phải) thì cần phải tính toán cách hiển thị vị trí sao cho hợp lý.

Giải pháp:

Để hiển thị hợp lý cần tính được các yếu tố sau:

* Vị trí cần hiển thị: thông qua phương thức offset của jQuery. Phương thức offset sẽ trả về vị trí của thành phần được chọn tính theo vị trí trong thành phần document.
* Vị trí của cửa sổ hiện tại: dựa trên 4 phương thức của jQuery là:
  + scrollTop(): trả về kết và là vị trí của thanh cuộn dọc cho thành phần được chọn.
  + scrollLeft(): trả về kết quả là vị trí của thanh cuộn ngang cho thành phần được chọn.
  + $(window).width(): trả về chiều rộng của cửa sổ web hiện tại.
  + $(window).height(): trả về chiều cao của cửa sổ web hiện tại.



# Tổng quát hóa phân tích dữ liệu

## Định nghĩa

Các dữ liệu trả về ban đầu là khối lượng lớn dữ liêu thô, dưới định dạng XML và không có đặc tả cho dữ liệu rõ ràng, 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.

## 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.
* Thông số đầu vào (paramaters): mỗi service được định nghĩa với thông số đầu vào khác nhau. Do đó, số lượng dữ liệu đầu vào lớn dẫn đến gây khó ngăn trong quá trình kiểm soát và sai sót trong quá trình thao tác với các services.
* Dữ liệu trả về (Service Response): tương tự thông số đầu vào số lượng dữ liệu lớn ngoài ra dữ liệu được trả về với nhiều cấu trúc dữ liệu khác nhau. Do đó, nếu dữ liệu không được xử lý sẽ tốn chi phi và thời gian cho quá trình kiểm soát và tính chính xác của dữ liệu không được đảm bảo.
* Ngoài ra, quá trình test dữ liệu trong các tác vụ xử lý khác sẽ gặp khó khăn dẫn đến rủi ro cao.
* Dữ liệu các servers Obs cung cấp được định dạng dữ liệu kiểu 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 tuy dữ liệu XML là dạng dữ liệu có cấu trúc nhưng do không được cung cấp đặc tả cấu trúc dữ liệu rõ ràng, mỗi API lại trả cấu trúc khác và nhu cầu lưu trữ dự liệu khác nhau, do đó không thể áp dụng được các công nghệ serialization và deserialization XML của C#.



Hình Dữ liệu Obs trả về

* 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.

## Phân tích

* Số lượng các thông số đầu vào và dữ liệu trả về của service rất lớn nhưng chúng có các thuộc tính chung .
* Dữ liệu XML không được cung cấp đặc tả cụ thể nhưng ta có thể thấy cấu trúc lưu trữ dữ liệu được lặp lại một số dạng nhất định.
* 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.

## Giải pháp

* Chúng ta sẽ viết các phương thức để chuyển đổi dữ liệu dưới dạng XML thành các đối tượng trong C#, việc này sẽ làm cho các tác vụ liên quan đến dữ liệu được xử lý dễ dàng và độ chính xác được đảm bảo. 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 các thuộc tính trong C#. Việc này giảm thời gian đáng kể và đảm bảo tên thuộc tính chính xác.



* Cấu trúc dữ liệu XML được lưu dưới dạng các node bao gồm các thuộc tính và giá trị, sau khi được phân tích ta có thể thấy dữ liệu có một số cấu trúc chính như Object, Collection, Dictionary và Primitive. Sau đó, viết phương thức duyệt lần lượt các node phía ngoài sau đó lần lượt duyệt vào trong để lấy dữ liệu về các thuộc tính và giá trị của node. Để viết phương thức này ta áp dụng một số kỹ thuật của C# :
  + Generic method: để cho phương thức linh hoạt hơn, tránh lập lại code => code được trình bày dễ hiểu, ngắn gọn
  + Activator Class: bao gồm phương tạo ra các lớp đối tượng
  + Unit Test: Đảm bảo tính đúng đắn của phương thức.
* Các dữ liệu cần thiết ví dụ như thông tin Users cần được cập nhật liên tục, do đó,việc tạo ra cache để lưu lại danh sách users và phân tích tính toán thời gian hợp lý để cập nhật 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 cập nhật tức thời để dành cho một số trường hợp như tạo tài khoản.

# Unit Test.

## Định nghĩa.

* + Unit Test là quá trình kiểm thử tự động 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.

## Lợi ích của Unit Test.

* Unit Tests 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 Tests giúp có thể kiểm tra lại code một cách nhanh chóng và chính xác khi thay đổi một phần nhỏ trong chương trình.
* Unit Tests chứng minh rằng code của bạn đang hoạt động.
* Test-first buộc developers phải có kế hoạch trước khi lập 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.

## Giới thiệu một số Framework Unit tests trên các ngôn ngữ khác nhau.

### C#

* + - Visual studio: Visual Studio Unit Testing Framework lần đầu tiên được bao gồm trong Visual Studio Team System 2005, nơi nó được tích hợp với IDE, nhưng không có sẵn trong hầu hết được sử dụng Standard Edition. Từ Visual Studio 2008 nó có sẵn cũng Professional Edition. Bắt đầu với Visual Studio Express 2013, nó được bao gồm các phiên bản với Visual Studio Express.

### Java

* + - Junit là một framework unit test cho các ngôn ngữ lập trình Java. JUnit là một phần quan trọng trong sự phát triển của test-driven development, , dựa trên kiến ​​trúc xUnit.

### C++

* + - Google C++ Testing Framework: một thư viện Unit Test cho các ngôn ngữ lập trình C ++, dựa trên kiến ​​trúc xUnit

## Unit tests trong portal.

* Các phương thứcđược viết đều được thiết kế và hiện thực các test cases tự động trước. Các dữ liệu trong database được sử dụng kĩ thuật mock để giả lập cho các unit tests để đảm bảo hạn chế lỗi tối đa khi chạy trong môi trường thực tế. Bên cạnh đó, số lượng test cases nhiều đảm bảo các chức năng chạy đúng như mong muốn do đó số lượng bug được tìm ra và sửa chữa sớm. Trong quá trình sửa chữa, các test cases luôn được chạy lại để đảm bảo tính đúng đắn của chương trình.



* Một số kỹ thuật được sử dụng trong Unit Test:
  + Kỹ thuật Mock:
    - Mock là một phương thức/đối tượng mô phỏng hành vi của các phương thức/đối tượng thực tế.
    - Mocking chủ yếu được sử dụng trong Unit Test. Một đối tượng được kiểm tra có thể phải phụ thuộc (dependencies objects or methods) vào các đối tượng phức tạp khác. Để tách biệt các hành vi của đối tượng muốn kiểm tra ta đối tượng bằng các đối tượng Mocks ,điều này rất hữu ích nếu các đối tượng thực tế không thể tích hợp vào Unit Test (database,…).
  + Kỹ thuật multi test:
    - Trong thực tế số lượng test cases rất lớn nếu mỗi test cases là một Unit Test đơn sẽ gây nhiều khó khăn trong quá trình hiện thực Unit Test như khó kiểm soát các test cases, đặt tên, số lượng code lớn,…
    - Các test cases được phân tích thành một tập hợp cùng với kì vọng kết quả của mỗi phần tử trong tập hợp đó. Các test này sẽ được tự động thực hiện lần lượt chỉ với một hàm test.



## Công việc thực hiện.

* Tạo một kế hoạch thử nghiệm
* Tạo các test cases và test data
* Nếu có thể tạo ra các scripts để chạy các test cases.
* Chạy test cases cho mỗi khi hoàn thành code.
* Sửa chữa các lỗi nếu có và chạy lại test cases.
* Lặp lại chu kỳ test cho đến khi "đơn vị" không còn lỗi.

# Làm việc nhóm

* Team Foudation

# Kết quả

## Đạt được

## Hạn chế