|  |
| --- |
|  |
| NodeRED Expansion |
| Reviewing possible addons with NodeRED |

|  |
| --- |
| Tran Nguyen Hoang Huy  7-28-2025 |

# Mục lục

[Mục lục 1](#_Toc204870215)

[1. Giới thiệu 2](#_Toc204870216)

[1.1. Mục tiêu nghiên cứu 2](#_Toc204870217)

[1.2. Phạm vi nghiên cứu 2](#_Toc204870218)

[1.3. Phương pháp nghiên cứu 2](#_Toc204870219)

[1.4. Công cụ nghiên cứu 2](#_Toc204870220)

[2. Các vấn đề cần xem xét 3](#_Toc204870221)

[2.1. Đánh giá, đề xuất cách thức kết nối với datasource 3](#_Toc204870222)

[2.2. Khả năng cấu hình điều kiện phức tạp gồm nhiều điều kiện 3](#_Toc204870223)

[2.3. Khả năng alert dạng chuỗi (chain) 4](#_Toc204870224)

[2.4. Khả năng repeat alert n lần 4](#_Toc204870225)

[3. Kết luận 4](#_Toc204870226)

# 1. Giới thiệu

## 1.1. Mục tiêu nghiên cứu

Xem xét và đánh giá khả năng tích hợp NodeRED vào hệ thống Alert, từ input, khai báo điều kiện, các logic như lặp lại alert hoặc alert theo chuỗi.

## 1.2. Phạm vi nghiên cứu

Phạm vi nghiên cứu bao gồm:

* Đánh giá, đề xuất cách thức kết nối với datasource
* Khả năng cấu hình điều kiện phức tạp gồm nhiều điều kiện
* Khả năng repeat alert **n** lần
* Khả năng alert dạng chuỗi (chain)

## 1.3. Phương pháp nghiên cứu

Nghiên cứu sử dụng phương pháp thực hành triển khai và phân tích hệ thống. Tham khảo tài liệu chính thức của NodeRED, xây dựng một mô hình thử nghiệm gồm các flow trong NodeRED, mô phỏng tình huống cảnh báo thực tế, từ đó phân tích luồng xử lý và các quyết định thiết kế.

## 1.4. Công cụ nghiên cứu

* NodeRED
* MySQL
* Postman
* ChatGPT
* Google

# 2. Các vấn đề cần xem xét

## 2.1. Đánh giá, đề xuất cách thức kết nối với datasource

Với datasource thì sẽ có 2 cách chính:

* API qua các Node HTTP
* Database qua các Node kết nối database

|  |  |  |
| --- | --- | --- |
|  | **API** | **Database** |
| **Mục đích** | Giao tiếp với các hệ thống/dịch vụ bên ngoài | Lưu trữ, truy xuất và phân tích dữ liệu có cấu trúc |
| **Input** | Dữ liệu có thể được gửi đến ngay từ API ở dạng JSON object | Cần node **Function** đi trước truyền query, hoặc một node **HTTP** truyền vào query để thực hiện |
| **Output** | Sử dụng node **function** để lấy dữ liệu  Trả về ở dạng JSON object | Sử dụng node function để lấy dữ liệu  Trả về ở dạng array gồm các object |
| **Trigger** | Node **HTTP In** có thể bắt đầu flow sau khi nhận được request gửi đến | Cần **HTTP In** hoặc các node trigger khác bắt đầu flow để truyền query |

Với mục đích là dùng input để xử lý event cho Alert thì API sẽ hỗ trợ tốt nhất do có sự phản hồi lập tức. Có thể là mình cũng muốn lưu lại các log thời gian Alert thì ở đây ta cũng có thể dùng các node **function, database** để đưa write dữ liệu vào database

## 2.2. Khả năng cấu hình điều kiện phức tạp gồm nhiều điều kiện

**Switch Node và JSONata:**

JSONata là một ngôn ngữ truy vấn và biến đổi dữ liệu JSON, qua đó ta có thể viết điều kiện phức tạp như:

**alerts[type="temperature" and (value > 80 or location="office")]**

* Mỗi điều kiện đặt ra sẽ tạo ra một nhánh, vì vậy nó đã đáp ứng được các điều kiện ở dạng **ANY**
* Để có được điều kiện phức tạp ở dạng **ALL** thì cùng phải query nhiều điều kiện cùng một lúc trong JSONata

**Boolean Module:**

Module này cho mình 2 node mới là **AND** và **OR**, gần như là đáp ứng được các nhu cầu của mình, nhưng lại không hỗ trợ JSONata, chỉ có các operator đơn giản như ở **Switch node.**

Kết luận: Đối với nhóm điều kiện **ANY**, mình có thể dễ dàng đáp ứng qua **Switch node** với JSONata. Nhưng đối với nhóm điều kiện **ALL** thì càng scale lên thì sẽ càng khó quản lý.

## 2.3. Khả năng alert dạng chuỗi (chain)

**Ý tưởng:**

Xây dựng một luồng cảnh báo có nhiều mức độ (Stage 1 → Stage 2 → Stage 3), mỗi mức độ tương ứng với một hình thức cảnh báo nghiêm trọng hơn.

* **Giai đoạn 1:** Gửi cảnh báo tới ứng dụng web.
* **Giai đoạn 2:** Gửi thêm email thông báo.
* **Giai đoạn 3:** Kích âm thanh qua loa.

Mỗi cảnh báo sẽ được nâng cấp sang giai đoạn kế tiếp nếu vẫn chưa được phản hồi sau một khoảng thời gian định sẵn, tạo thành một **stage flow** giúp người không bỏ sót cảnh báo quan trọng và đảm bảo hệ thống phản ứng thích hợp tùy theo mức độ khẩn cấp.

Một node **function** sẽ kiểm soát việc tăng bậc stage, và một **flow** để kiểm tra trạng thái của alert. Nếu trạng thái vẫn active thì flow sẽ truyền về function kiểm soát để kiểm tra Stage hiện tại và tăng lên nếu cần.

Ngoài ra thì trong **flow** kiểm tra trạng thái sẽ có node để nhận API, mục đích của node này là nếu đã có người tiếp nhận được alert, stage sẽ được reset lại về trạng thái ban đầu.

## 2.4. Khả năng repeat alert n lần

**Ý tưởng:**

Khi hệ thống cố gắng gửi cảnh báo nhưng không nhận được phản hồi hợp lệ (mã lỗi HTTP hoặc timeout), thay vì bỏ qua, nó sẽ tự động thử gửi lại. Sẽ có một **retry flow** quản lý quá trình retry, nếu thành công request gặp lỗi thì **retry flow** sẽ trigger lại flow alert chính, kiểm tra lại **stage** hiện tại rồi mới tiếp tục.

Ngoài ra, trong **alert flow** sẽ có một cổng kiểm soát dữ liệu, nếu đang ở trong trạng thái retry thì cổng sẽ được đóng lại để tránh ảnh hưởng đến **stage flow.**

# 3. Kết luận

Qua nghiên cứu và thử nghiệm tạo ra một logic cảnh báo trong Node-RED, có thể rút ra một số kết luận như sau:

* **Node-RED có thể đáp ứng yêu cầu của một hệ thống cảnh báo nhiều bậc**, từ tiếp nhận dữ liệu đầu vào, kiểm soát trạng thái, kiểm tra điều kiện cho đến thực hiện các hành động cụ thể (gửi cảnh báo, email, phát loa…).
* **Logic cấu hình điều kiện linh hoạt** thông qua các node function, switch, và JSONata giúp hệ thống dễ dàng mở rộng và tùy biến cho từng tình huống cụ thể. Tuy nhiên, khi số lượng điều kiện tăng cao, việc quản lý logic có thể trở nên phức tạp nếu không có quy ước rõ ràng.
* **Cơ chế alert dạng chuỗi (chain alert):** cho phép nâng cấp mức độ cảnh báo tùy theo mức độ phản hồi, đảm bảo không bỏ sót cảnh báo quan trọng và phản ứng kịp thời với các tình huống nghiêm trọng hơn.
* **Cơ chế lặp lại cảnh báo (repeat alert):** Nếu gặp sự cố khi gửi alert thì có thể retry theo số lần cần thiết.
* Hạn chế lớn nhất nằm ở việc **quản lý trạng thái phức tạp khi scale lên**, đặc biệt là với nhiều stage hoặc nhiều luồng cảnh báo khác nhau diễn ra đồng thời.

Như vậy, **Node-RED** có thể đáp ứng được đa số các nhu cầu của hệ thống **Alert**. Khi triển khai thực tế ở quy mô lớn hơn, thì chặt chẽ trong việc thiết kế hệ thống và khả năng mô-đun hóa để đảm bảo hiệu quả lâu dài.