# AUTHENTICATION

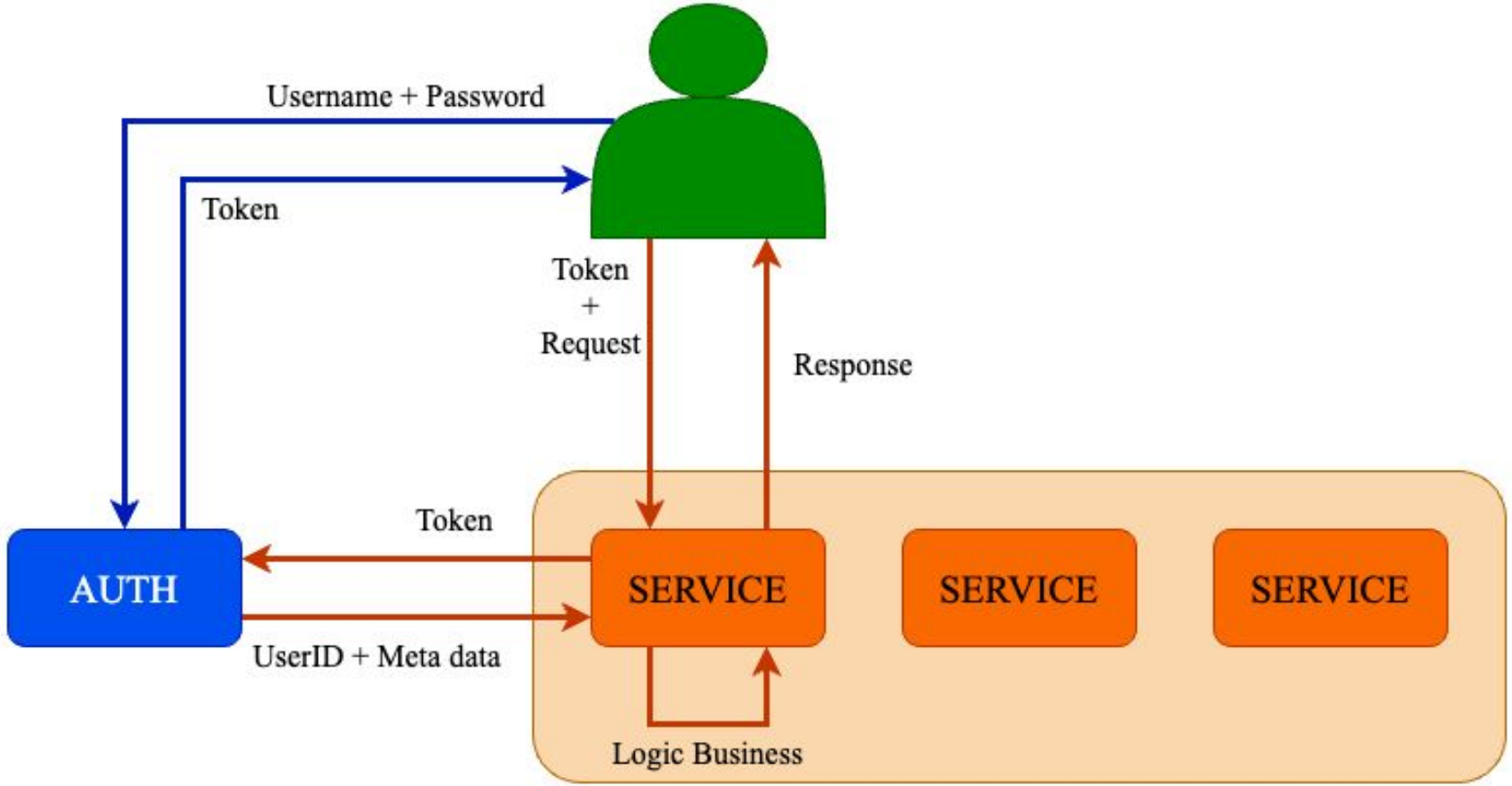
HUỲNH TRỌNG TIẾN

# MỤC LỤC

# Nguyên tắc chung

- Khi A gửi request cho B cần có thông tin (đặt trong header) để B xác thực danh tính và tin cậy A.
- Dù là Token hay JWT, luôn phải có thông tin thời gian tạo, có thể có hoặc không thời gian hết hạn.
- Hạn chế tối đa các thông tin quan trọng trong Token hoặc JWT.

| | |
|---|---|
| POST /echo/get HTTP/1.1<br>Host: google.com<br>Accept: application/json<br>Authorization: Bearer {token}<br><br>id=1&a=1 | GET /echo/get?id=1&a=1 HTTP/1.1<br>Host: google.com<br>Accept: application/json<br>Authorization: Bearer {token} |

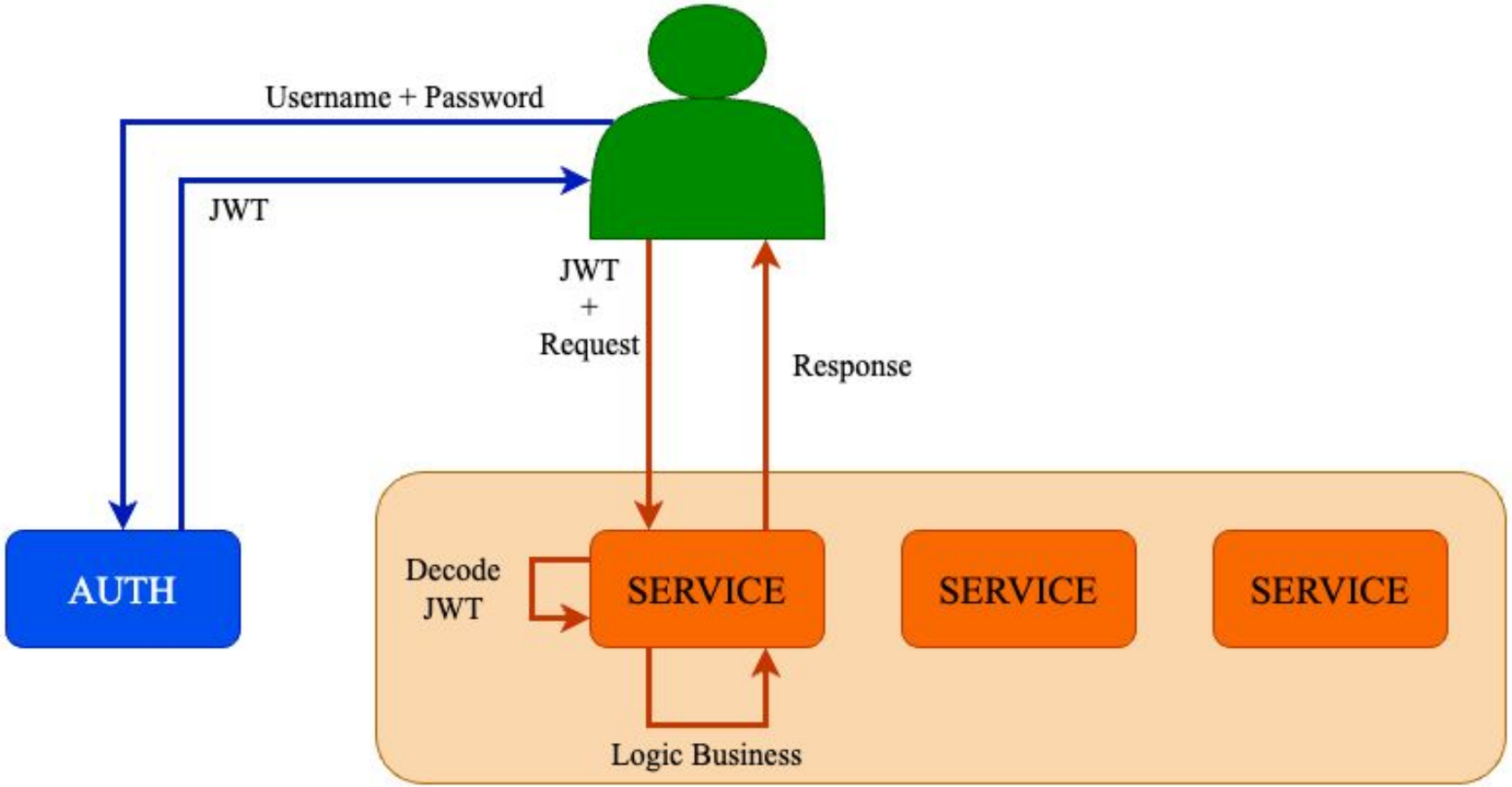

Username + Password
Token
Token
+
Request
Response
Token
AUTH
SERVICE
SERVICE
SERVICE
UserID + Meta data
Logic Business

# Ưu điểm

- Quản lí user tập trung.
- Dễ dàng bảo vệ khoá, tăng tính bảo mật.
- Các service khác không cần quan tâm thuật toán mã hoá.

# Nhược điểm

- Chỉ dùng cho trường hợp có nhiều gateway.
- Dễ trở thành thắt cổ chai của hệ thống.

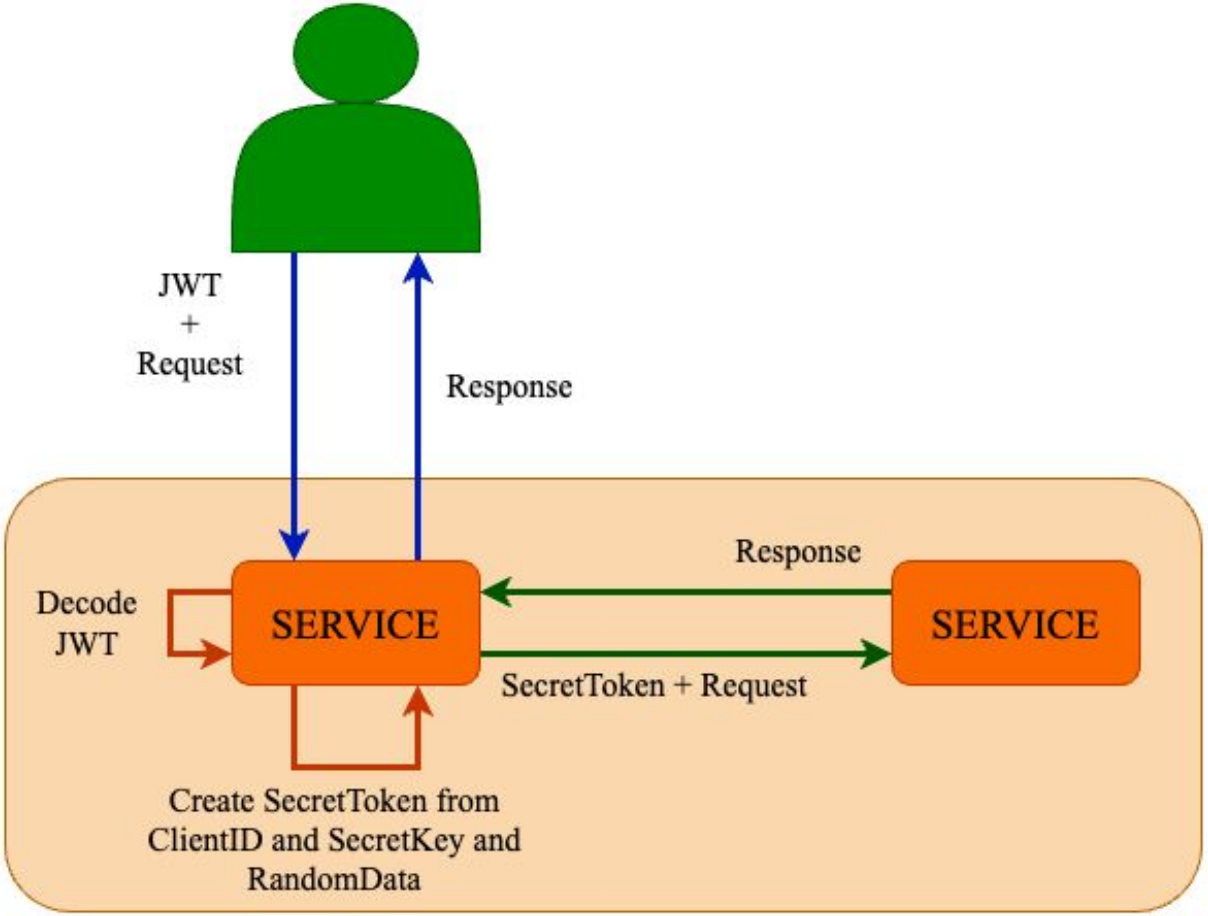

Username + Password
JWT
JWT
+
Request
Response
AUTH
Decode
JWT
SERVICE
SERVICE
SERVICE
Logic Business

# Ưu điểm

- Dễ sử dụng, tạo một lần, dùng nhiều nơi.
- Không phụ thuộc vào service session.

# Nhược điểm

- Khó khăn trong việc bảo mật key.
- Phải có service khác cung cấp thông tin cho trường hợp user bị khoá hoặc thay đổi quyền hạn.
- Thời gian sử dụng ngắn.


JWT
+
Request
Response
Decode
JWT
SERVICE
Response
SERVICE
SecretToken + Request
Create SecretToken from
ClientID and SecretKey and
RandomData

# Ví dụ

Token = md5(clientID + secretKey + TimestampOfNow)

# Ưu điểm

- Dễ sử dụng.
- Thời gian nhanh.
- Phù hợp dùng trong nội bộ hệ thống.
- Có thể tận dụng để monitor hệ thống, rate limit request từ một service khác.

# Nhược điểm

- Mỗi token chỉ dùng một lần.
- Phải có cơ chế quản lí danh sách ClientID và ClientKey.
- Không thể dùng cho môi trường mạng công cộng.


QUESTIONS?
ANSWERS!