Kiến Trúc Hệ Thống Xác Thực JWT Trong Microservices

Kiến Trúc Tổng Thể Hệ Thống

Client

Ứng dụng web/mobile của người dùng, gửi yêu cầu đăng nhập và lưu trữ JWT nhận được để sử dụng cho các yêu cầu tiếp theo.

API Gateway

Điểm vào duy nhất của hệ thống, có nhiệm vụ định tuyến yêu cầu và kiểm tra tính hợp lệ của token.

Auth Service

Xác thực người dùng, phát hành JWT theo chuẩn OAuth2, quản lý phiên làm việc.

Luồng Thực Thi

1. Người dùng gửi thông tin đăng nhập tới API Gateway
2. Gateway chuyển yêu cầu cho Auth Service
3. Auth Service xác thực và trả về JWT
4. Client đính kèm JWT vào các yêu cầu tiếp theo (`Authorization: Bearer <token>`)

1. Gateway xác minh JWT (chữ ký, thời gian sống, issuer/audience)
2. Sau khi xác minh, Gateway chuyển yêu cầu tới dịch vụ thích hợp
3. Các dịch vụ nghiệp vụ (Chart Service, AI Prediction, Backtesting) truy cập dữ liệu qua Message Broker
4. Kết quả được trả về cho client

Mã Minh Họa Hệ Thống Xác Thực

Endpoint Đăng Nhập

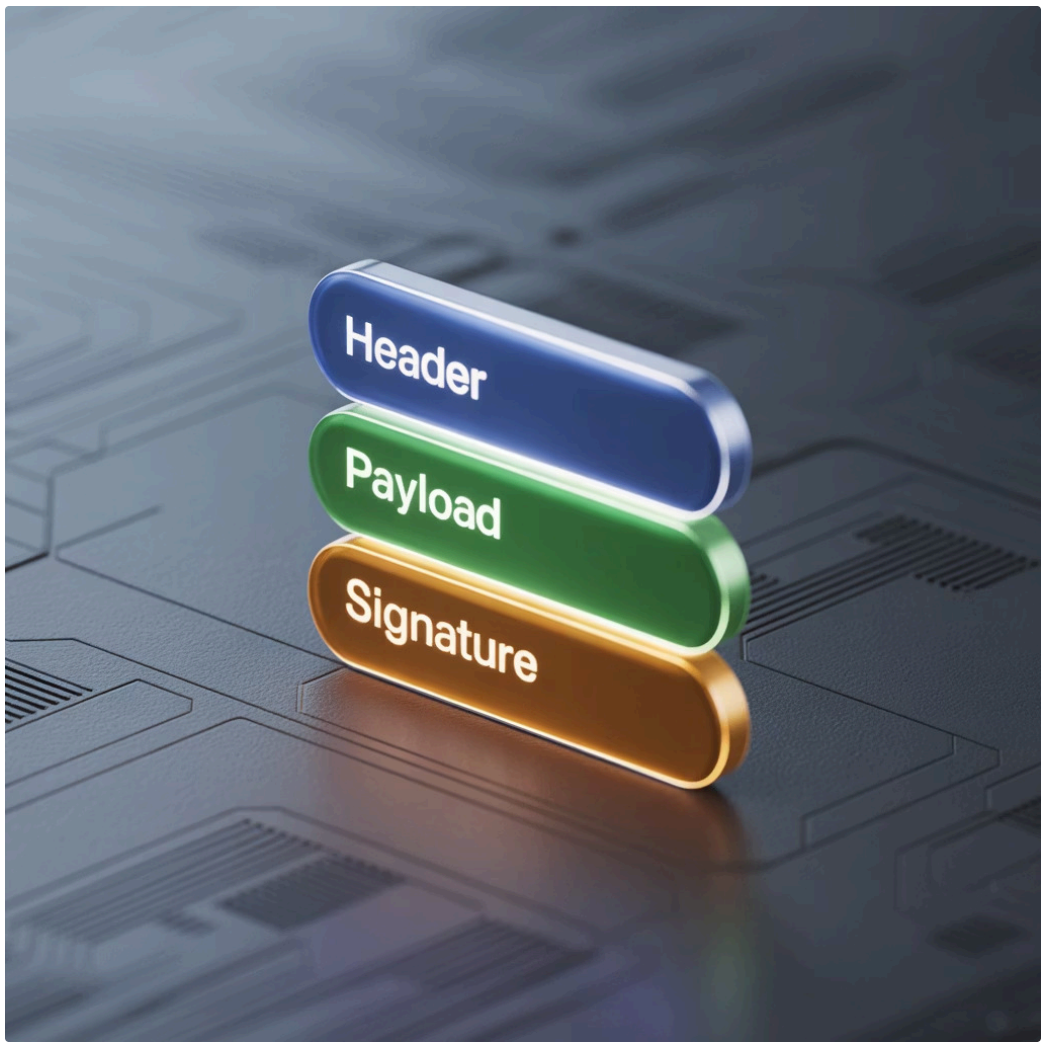
```
// Endpoint /login trong AuthService
app.MapPost("/login", async (HttpContext http) => {
  var loginRequest = await http.Request
    .ReadFromJsonAsync();

  // Kiểm tra tên đăng nhập/mật khẩu
  if (loginRequest.Username == "user" &&
      loginRequest.Password == "password") {

    // Tạo danh sách claim chứa thông tin người dùng
    var claims = new List {
      new Claim(ClaimTypes.NameIdentifier,
              loginRequest.Username),
      new Claim(ClaimTypes.Role, "User")
    };

    // Định nghĩa cấu trúc và thời hạn token
    var tokenDescriptor = new SecurityTokenDescriptor {
      Subject = new ClaimsIdentity(claims),
      Expires = DateTime.UtcNow.AddMinutes(30),
      Issuer = "ExampleAuthServer",
      Audience = "ExampleResourceServer",
      SigningCredentials = new SigningCredentials(
        new SymmetricSecurityKey(signingKey),
        SecurityAlgorithms.HmacSha256Signature)
    };

    // Tạo JWT và gửi lại cho client
    var tokenHandler = new JwtSecurityTokenHandler();
    var securityToken = tokenHandler.CreateToken(
                        tokenDescriptor);
    var jwt = tokenHandler.WriteToken(securityToken);

    return Results.Ok(new { access_token = jwt });
  }

  return Results.Unauthorized();
});
```

Endpoint Được Bảo Vệ

```
// Endpoint yêu cầu chứng thực
app.MapGet("/protected-resource",
  [Authorize] (ClaimsPrincipal user) => {
    // Hàm chỉ chạy khi token hợp lệ
    var username = user.Identity?.Name ?? "anonymous";
    return Results.Ok(
      $"Xin chào, {username}! Đây là tài nguyên được bảo vệ."
    );
  });
```

Giải Thích

Thuộc tính `[Authorize]` buộc ASP.NET Core kiểm tra JWT trước khi xử lý request. Nếu token hợp lệ, `ClaimsPrincipal` sẽ chứa thông tin người dùng từ JWT. Nếu token không hợp lệ hoặc thiếu, hệ thống tự động trả về mã lỗi 401/403.



Lợi Ích Của Kiến Trúc Xác Thực Phân Tán

Phân Tách Mối Quan Tâm (Separation of Concerns)

Dịch vụ xác thực (Auth Service) hoàn toàn tách biệt với các dịch vụ nghiệp vụ, giúp hệ thống dễ bảo trì và phát triển. Mỗi thành phần chỉ tập trung vào một chức năng cụ thể.

Khả Năng Mở Rộng (Scalability)

Auth Service và các dịch vụ nghiệp vụ có thể mở rộng độc lập theo nhu cầu sử dụng. Khi lưu lượng tăng, bạn chỉ cần tăng tài nguyên cho các dịch vụ cần thiết mà không ảnh hưởng đến toàn bộ hệ thống.

An Ninh Cao (Security)

Token được ký số bằng thuật toán mạnh (HMAC/RSA); API Gateway kiểm tra mọi yêu cầu trước khi chuyển tiếp, ngăn chặn các yêu cầu không hợp lệ tiếp cận các dịch vụ nội bộ. JWT không lưu trạng thái trên server, giảm nguy cơ tấn công session.

Khả Năng Mở Rộng Chức Năng (Extensibility)

Dễ dàng bổ sung thêm microservice mới (như dịch vụ cảnh báo, phân tích dữ liệu) mà không ảnh hưởng tới cơ chế xác thực hiện tại. Các dịch vụ mới chỉ cần tuân thủ cơ chế xác thực JWT đã được thiết lập.

Phù Hợp Với Tiêu Chuẩn

Kiến trúc này tuân thủ các tiêu chuẩn OAuth2 và JWT - được sử dụng rộng rãi trong các hệ thống tài chính và ứng dụng yêu cầu bảo mật cao. Điều này giúp tăng tính tương thích và dễ tích hợp với các hệ thống khác.