**Các thành phần trong mô hình phân quyền:**

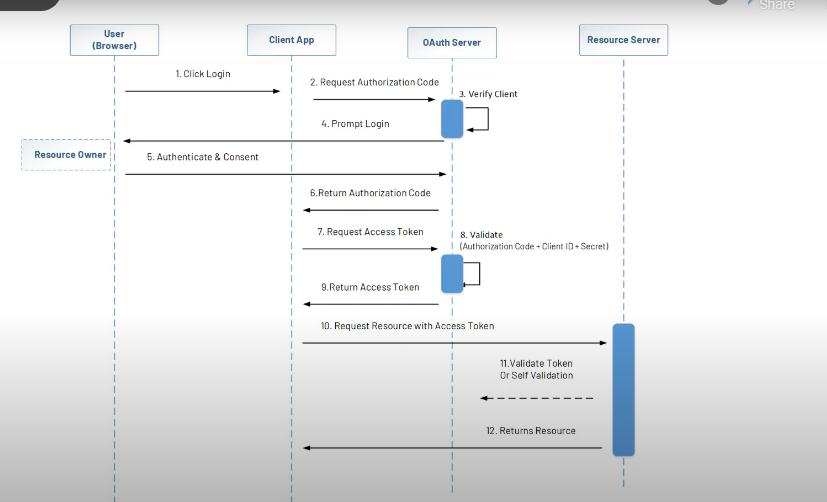
**1 Resource Server**: là các server tài nguyên chứa service, DB, process…

**2 Resource Owner:** Người nắm giữ tài nguyên **– Admin của hệ thống.** Có quyền cho phép client truy cập hay không vào các tài nguyên được bảo vệ

**3 Client:** Người dùng hệ thống: yêu cầu cập quyền truy cập vào hệ thốg, sd các thiết bị mobile, web browser

**4 Authorize Server**: Máy chủ xác thực, phân quyền cho **3.Client,**  là thành phần đang cần xây dựng.

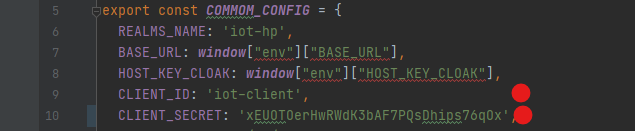
**Các luồng trong mô hình:**



1: Bước 1: user click login tới **client\_app** (ứng dụng front-end/ mobile app)

2: **client\_app** yêu cầu **authorization\_code** từ **Oauth\_server**

3: xác minh client (**id + secret**) đó có hợp lệ hay không, đăng ký hay **chưa (ứng dụng front-end luôn phải có 1 client đã tạo(xác minh hợp lệ) trước khi thực hiệntác vụ gì đó để yêu cầu access\_token)**

****

4: Khi client\_id và client-secret hợp lệ, **Oauth-server** sẽ trả ra kq hợp lệ và trả về **form login** cho user.

5: user điền ttin login và gửi cho **Oauth-server**

6 Nếu ttin login hợp lệ, **Oauth-server** trả về 1 **Authorization\_code**  cho **client app**

7 **Client\_app** sd **Authorization\_code** mới nhận được kèm mới **client\_id + secret**  để y/c access\_token từ **Oauth-server**.

8: **Oauth-server** validate (**Authorization\_code + client\_id + client\_code**)

9 nếu thỏa mãn. Trả lại **accesss\_token** cho **client\_APP**

10 **client\_app** thực hiện yêu tới **service** sd **access-token** nhận được tại 9

11 **service** xác nhận **access\_token** còn hợp lệ hay ko+

12: **service** trả lại kết quả cho **client-app**

**A-F token**

Access-token hay rêfresh-token là các string;

**Access-token** là string **đã được cấp quyền rồi cho client sử dụng để truy cập tài nguyên**. thường có thời gian tồn tại ngắn (5-10p) nếu dài sẽ có nguy cơ bị hacker tấn công

**Refresh-token**: là 1 token có thời gian sống lâu hơn accesstoken, thường trong khởng vài ngày-tháng. Có vai trò là yêu cầu cấp lại access-token sau khi access-token cũ đã hết hạn mà không cầ phải đăng nhập lại.

**Json-web-token(jwt)** là 1 thư việnd để encode tất cả các phần của access-token sang dạng encoded trc khi chuyển qua lại token giữa client-server

**Client registration**

Để làm việc với Oauth server. Client phải đăng ký, thường dùng name đăng ký website/

**Callback url, redirectUrl:**  sau khi ressource owner grant quyền thành công, server sẽ chuyển hướng tới các URL này để trả về **AUTHORATION\_CODE**  hoặc  **ACCESS\_TOKEN.**

**CLIENT\_ID , CLIENT\_SECRET** : client\_id là id cấp cho client (public), client\_secret là mật khẩu (private)

**SCOPE**: giới hạn cho client khi được **resource server** grant

**Grant\_type** là luồng get access\_token. Có các loại thường dùng sau: **authorization\_dode, client\_credentials, DeviceCode, RefreshToken, password** không thường dùng: implcit\_flow, **password**..

Vd user click login to client

Client ask for authorization code using client(id + secret)

Client use that authorization code + client(id + secret) to ask for access token.  
client request resource using that token + client info to reserved server (services)

Services check that token + client info : isValid, if valid, return data to client

**Trong oauth2, có 2 loại tài khoản: tài khoản END USER VÀ tài khoản CLIENT(SCOPE)**

**End user** là tài khoarnng dùn dùng để đăng nhập vào hệ thống autho-server (authorization-server, keycloak), chúng chứa **scopes** và **roles**, còn **tài khoản CLIENT** là tài khoản các reserved server(sevice) dùng khi chúng muốn gọi qua lại lẫn nhau , chúng chứa **scope**s.