**TRƯỜNG ĐẠI HỌC KHOA HỌC-ĐẠI HỌC HUẾ**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙞🙞🕮🙜🙜**



**TIỂU LUẬN**

**ĐỀ TÀI: TÌM HIỂU GIAO THỨC OAUTH**

**VÀ XÂY DỰNG ỨNG DỤNG THỬ NGHIỆM**

**GVHD: Th.s Võ Việt Dũng**

**Sinh viên thực hiện:**

1. Nguyễn Công Thành – 21T1020708
2. Nguyễn Phước Tuấn Kiệt – 21T1020460
3. Hồ Bảo Quốc – 21T1020636
4. Nguyễn Minh Trí – 21T1020773
5. Phạm Minh Tuấn – 21T1020806

**HUẾ, THÁNG 01 NĂM 2025**

# **LỜI CẢM ƠN**

Lời đầu tiên chúng em muốn gửi lời cảm ơn chân thành tới Thầy Võ Việt Dũng là giảng viên hướng dẫn đồ án của chúng em. Trong suốt thời gian thực hiện bài tiểu luận Thầy đã tạo điều kiện cho chúng em về thời gian và những sự giúp đỡ quý báu về kiến thức, sự chỉ dẫn, định hướng và tài liệu tham khảo.

Và sau đó chúng em muốn gửi lời cám ơn chân thành nhất tới toàn thể các thầy cô trong trường. Các thầy cô là những người có kiến thức sâu rộng, nhiệt tình với sinh viên giúp chúng em có thêm kiến thức để làm hoàn thành đề tài.

Mặc dù đã có những đầu tư nhất định trong quá trình làm bài song cũng khó có thể tránh khỏi những sai sót, em kính mong nhận được ý kiến đóng góp của quý thầy cô để bài tiểu luận được hoàn thiện hơn.

Cuối cùng em xin kính chúc các thầy cô và toàn thể các bạn sinh viên trường Đại học Khoa học có một sức khỏe dồi dào, đạt được những thành công trên con đường học tập và nghiên cứu khoa học của mình và đặc biệt là đón giao thừa bên gia đình thật vui vẻ và hạnh phúc.

**Chúng em xin chân thành cảm ơn!**

# **MỤC LỤC**

[**LỜI CẢM ƠN** 1](#_Toc187382317)

[**MỤC LỤC** 2](#_Toc187382318)

[**CHƯƠNG 1. TỔNG QUAN** 5](#_Toc187382319)

[**1.** **Mục tiêu** 5](#_Toc187382320)

[**2.** **Đặt vấn đề** 5](#_Toc187382321)

[**3.** **Kịch bản bài toán** 5](#_Toc187382322)

[**4.** **Bố cục bài toán** 8](#_Toc187382323)

[**CHƯƠNG 2. TÌM HIỂU OAUTH 1.0 VÀ OAUTH 2.0** 9](#_Toc187382324)

[**1.** **Lịch sử ra đời** 9](#_Toc187382325)

[**2.** **Khái niệm tổng quát** 9](#_Toc187382326)

[**3.** **Giải thích Oauth** 10](#_Toc187382327)

[**4.** **Một vài ví dụ về Oauth** 10](#_Toc187382328)

[**5.** **OAUTH 1.0** 12](#_Toc187382329)

[5.1. Một vài định nghĩa: 12](#_Toc187382330)

[5.2. Xác thực với Oauth 1.0: 13](#_Toc187382331)

[5.3. Nhận mã thông báo yêu cầu trái phép 14](#_Toc187382332)

[5.4. Lấy quyền của người dùng 14](#_Toc187382333)

[5.5. Nhận mã thông báo truy cập 14](#_Toc187382334)

[**6.** **OAUTH 2.0** 15](#_Toc187382335)

[6.1. Khái niệm: 15](#_Toc187382336)

[6.2. Các Role (vai trò) trong mô hình Oauth 2.0 15](#_Toc187382337)

[6.3. Luồng xử lý của giao thức 16](#_Toc187382338)

[6.4. Authorization Grant 17](#_Toc187382339)

[6.4.1. Authorization code 18](#_Toc187382340)

[6.4.2. Implicit 18](#_Toc187382341)

[6.4.3. Resource owner password credential 18](#_Toc187382342)

[6.5. Các khái niệm khác 18](#_Toc187382343)

[6.5.1. Access token 18](#_Toc187382344)

[6.5.2. Refesh token 18](#_Toc187382345)

[6.6. Phạm vi (Scope) của Access token 20](#_Toc187382346)

[7. SO SÁNH OAUTH 1.0 VÀ OAUTH 2.0 21](#_Toc187382347)

[**CHƯƠNG 3. THỰC NGHIỆM** 24](#_Toc187382348)

[1. Chuẩn bị 24](#_Toc187382349)

[2. Mô phỏng 24](#_Toc187382350)

[3. Tạo Database vào hệ thống qua công cụ SQL Server 24](#_Toc187382351)

[4. Xây dụng trang web local trước khi tiến hành chứng thực bằng Oauth 2.0 25](#_Toc187382352)

[4.1. Code trên Admin 25](#_Toc187382353)

[4.2. Code trên Frontend 26](#_Toc187382354)

[5. Triển khai chứng thực trên web local đã buil bằng Oauth 2.0 27](#_Toc187382355)

[5.1. Chạy file Admin 27](#_Toc187382356)

[5.2. Chạy file Frontend 27](#_Toc187382357)

[5.3. Tiến hành kết nối tới web localhost đã được thực thi 28](#_Toc187382358)

[**CHƯƠNG 4. TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN** 34](#_Toc187382359)

[1. Những vấn đề làm được 34](#_Toc187382360)

[2. Những vấn đề chưa làm được 34](#_Toc187382361)

[3. Hướng phát triển 34](#_Toc187382362)

**PHỤ LỤC HÌNH ẢNH**

[Hình 1: Sơ đồ mô tả kịch bản bài toán 9](#_Toc48213844)

[Hình 2: Ví dụ xác thực Oauth qua nhà cung cấp dịch vụ Facebook 13](#_Toc48213845)

[Hình 3: Ví dụ xác thực Oauth qua nhà cung cấp dịch vụ Google 14](#_Toc48213846)

[Hình 4: Sơ đồ xác thực của OAuth 1.0 16](#_Toc48213847)

[Hình 5: Sơ đồ thông số luồng xử lí có trong Oauth 1.0 17](#_Toc48213848)

[Hình 6: : Mô hình xử lí của giao thức Oauth 2.0 18](file:///C:\Users\ADMIN\Desktop\DOAN1TC%20-%20ChungPhuc.docx#_Toc48213849)

[Hình 7: : Luồng xử lí truy cập của Oauth 2.0 19](#_Toc48213850)

[Hình 8: Làm mới Access token đã hết hạn 21](#_Toc48213851)

[Hình 9: Cơ sở dữ liệu dùng để chia sẽ tài nguyên 26](#_Toc48213852)

[Hình 10: Cở sở dữ liệu dùng để đăng nhập hệ thống 26](#_Toc48213853)

[Hình 11: Vai trò của User trong hệ thống 26](#_Toc48213854)

[Hình 12: Code cấu hình máy chủ ủy quyền 27](#_Toc48213855)

[Hình 13: Code cấu hình chi tiết dịch vụ phía client 27](#_Toc48213856)

[Hình 14: Code cấu hình kết thúc ủy quyền của máy chủ 27](#_Toc48213857)

[Hình 15: Code cấu hình phần login 28](#_Toc48213858)

[Hình 16: Code cấu hình phần logout 28](#_Toc48213859)

[Hình 17: Code lấy token 28](#_Toc48213860)

[Hình 18: Code làm mới token 28](#_Toc48213861)

[Hình 19: Debug code phần admin 29](#_Toc48213862)

[Hình 20: Debug code phần frontend 29](#_Toc48213863)

[Hình 21: Giao diện của web localhost và phần login 30](file:///C:\Users\ADMIN\Desktop\DOAN1TC%20-%20ChungPhuc.docx#_Toc48213864)

[Hình 22: Giao diện form đăng kí tài khoản 30](#_Toc48213865)

[Hình 23: Giao diện form quên mật khẩu 31](#_Toc48213866)

[Hình 24: Giao diện sau khi đăng nhập thành công 31](file:///C:\Users\ADMIN\Desktop\DOAN1TC%20-%20ChungPhuc.docx#_Toc48213867)

[Hình 25: Token được sinh ra 32](#_Toc48213868)

[Hình 26: Làm mới Access token 32](#_Toc48213869)

[Hình 27: Thông tin User đăng nhập 33](#_Toc48213870)

[Hình 28: Giao diện khi chưa đăng nhập của phần local storage 34](#_Toc48213871)

[Hình 29: Sơ đồ luồng xử lí của bản dự thảo Oauth 2.1 36](#_Toc48213872)

# **CHƯƠNG 1. TỔNG QUAN**

1. **Mục tiêu**

Mục tiêu của đề tài là nghiên cứu về giao thức Oauth. Giao thức có thể ứng dụng trong việc đăng nhập tài khoản mà không cần phải nhập username và password. Bảo mật mọi phiên làm việc của bạn trước mối đe dọa tấn công từ việc đăng nhập tài khoản. Ví dụ: Keylogger, Virus, Trojan,..

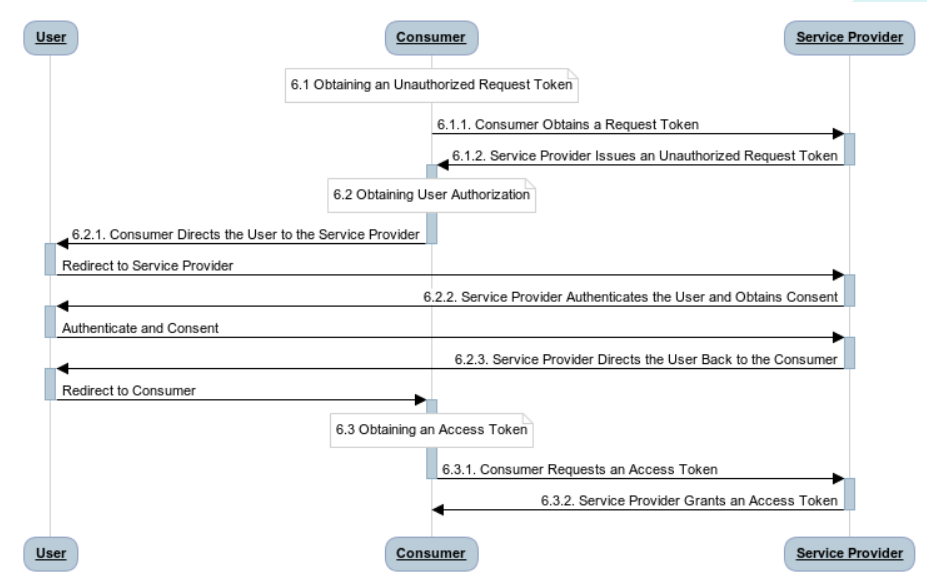
1. **Đặt vấn đề**

Chúng tôi sẽ bắt đầu bài toán bằng một ví dụ điển hình: Bạn đang muốn tải một tài liệu từ một trang web nào đó để **chia sẻ tài nguyên**, hệ thống sẽ yêu cầu bạn đăng nhập để tải văn bản mà bạn yêu cầu. Và tin chúng tôi đi, với những văn bản phù hợp với chất văn mà bạn muốn, nó đều mất phí khi bạn tải. Nếu không muốn trả một khoản tiền (dù vài nghìn đồng) cho một văn bản chỉ xài một lần với mục đích hoàn thành bài đồ án, thì bạn sẽ đi tìm trên các trang web khác để có một chất văn như ý và được miễn phí. Và bạn sẽ phải đi tìm nó, tìm đi tìm lại loay hoay một hồi rồi bạn nhận ra rằng: “ không có trang web nào đáp ứng đủ yêu cầu của bạn mà miễn phí cả”. Ở đây, một chuỗi hành động được lặp đi lặp lại như chính cách mà bạn đã làm trên trang web đó vậy. Sau cùng, bạn nhận ra top đầu trong hòm thư của mình là hàng loạt các Email yêu cầu kích hoạt tài khoản và bạn đã lãng phí hàng tiếng đồng hồ để lặp đi lặp lại cái hành động: Đăng Kí – Kích Hoạt – Đăng Nhập nhàm chán này..

Để tránh mất thời gian vào việc Đăng Kí – Kích Hoạt – Đăng Nhập rồi lại Đăng Kí – Kích Hoạt – Đăng Nhập thì các nhà cung cấp dịch vụ lớn trên thế giới như: Facebook, Twitter hay Google,.. đã giải quyết được khó khăn trên. Với khó khăn này nhóm chúng em xin giới thiệu giải pháp để giải quyết là **Oauth**. Đồng thời phạm vi giới thiệu của nhóm em sẽ tập trung nhiều hơn vào chuẩn Oauth mới nhất hiện nay là **Oauth 2.0**

1. **Kịch bản bài toán**

Trong ví dụ của chúng em: **Thành** là người dùng (**User**), **Kiệt** là người tiêu dùng (**Consumer**) và **Facebook** là dịch vụ được cung cấp (**Service Provider)**, và cũng có thể thêm một hacker **Quốc** là kẻ mạo danh Kiệt trong quá trình chia sẽ tài nguyên. Người kiểm soát tài nguyên an toàn của Thành là trang Facebook cá nhân của anh ấy. Giờ đây, Thành muốn Kiệt có thể đăng các liên kết rút ngắn lên trang Facebook của mình. Đây là cách nó hoạt động:



Hình 1: Sơ đồ mô tả kịch bản bài toán

* **Bước 1 - Người dùng hiển thị ý định**
  + - * **Thành (User):** Hey Kiệt!! tôi muốn bạn có thể đăng liên kết trực tiếp lên trang Facebook của tôi.
* **Kiệt (Consumer):** Tuyệt vời! Chờ tôi đi xin phép Facebook đã.
* **Bước 2 - Quyền của người tiêu dùng**
* **Kiệt (Consumer):** Tôi có một người dùng muốn tôi đăng lên trang cá nhân của anh ấy. Tôi có thể có mã thông báo yêu cầu không??
* **Facebook (Service Provider):** Vâng!! Chắc chắn. Đây là một mã thông báo của bạn và nó được bảo mật.
* **Bước 3 - Người dùng được chuyển hướng đến Nhà cung cấp dịch vụ**
* **Kiệt (Consumer):** OK Thành. Tôi đang gửi yêu cầu của bạn lên Facebook và đang chờ phê duyệt. Mã thông báo sớm được gửi đến bạn thôi mà!!
* **Thành (User):** À!! Có mã thông báo rồi Kiệt ơi.

**<Kiệt chỉ đạo Thành đến Facebook để ủy quyền>**

Đây là phần đáng sợ. Nếu có một người bạn tên **Quốc** là hacker chuyên nghiệp sẽ mạo danh Kiệt, ở đây **Quốc** có thể bật lên một cửa sổ trông giống như Facebook nhưng thực sự là lừa đảo cho tên người dùng và mật khẩu của bạn. Bạn phải chắc chắn rằng URL bạn hướng đến thực sự là nhà cung cấp dịch vụ Facebook (tên miền Facebook là: “*https://www.facebook.com*”).

* **Bước 4 - Người dùng cấp phép**
* **Thành (User):** Facebook Facebook, tôi muốn ủy quyền mã thông báo yêu cầu này mà Kiệt đã cho tôi.
* **Facebook (Service Provider):** chắc chắn, bạn muốn ủy quyền cho Kiệt thực hiện X, Y và Z bằng tài khoản Facebook của mình??
* **Thành (User):** Vâng!!
* **Facebook (Service Provider):** OK OK, bạn có thể quay lại với Kiệt và nói cho họ rằng họ có quyền sử dụng mã thông báo yêu cầu của họ.

Facebook đánh dấu mã thông báo yêu cầu là tốt, nên khi người tiêu dùng yêu cầu quyền truy cập, nó sẽ được chấp nhận (miễn là nó được ký bằng bí mật chung của họ).

* **Bước 5 - Người tiêu dùng có được mã thông báo truy cập**
* **Kiệt (Consumer):** Facebook Facebook, tôi có thể trao đổi mã thông báo yêu cầu này để lấy mã thông báo truy cập không?
* **Facebook (Service Provider):** chắc chắn. Đây là mã thông báo truy cập và bí mật của bạn.
* **Bước 6 - Người tiêu dùng truy cập tài nguyên được bảo vệ**
* **Kiệt (Consumer):** Tôi muốn đăng liên kết này lên trang cá nhân của Thành. Đây là mã thông báo truy cập của tôi!
* **Facebook (Service Provider) thông báo:** đã hoàn tất!!

Trong kịch bản của chúng em, Thành (**User**) không bao giờ phải chia sẻ thông tin đăng nhập Facebook (**Service Provider**) của mình với Kiệt (**Consumer**). Anh ta chỉ cần ủy quyền truy cập bằng OAuth (**Protocol**) một cách an toàn. Bất cứ lúc nào, Thành (**User**) có thể đăng nhập vào Facebook (**Service Provider**) và xem xét quyền truy cập mà anh ta đã cấp và thu hồi mã thông báo cho các ứng dụng cụ thể mà không ảnh hưởng đến người khác. OAuth (**Protocol)** cũng giới hạn mức độ truy cập cho phép. Bạn có thể cho Kiệt (**Consumer**) quyền đăng bài lên tài khoản Facebook (**Service Provider**) của mình, nhưng hạn chế việc đọc tin nhắn trên Messenger chẳng hạn.

1. **Bố cục bài toán**

* Chương 1: TỔNG QUAN

Phần này, chúng em nói về tổng quan của đề tài: Đặt vấn đề, mô tả bài toán,..

* Chương 2: TÌM HIỂU GIAO THỨC OAUTH 1.0 VÀ OAUTH 2.0

Phần này, chúng em sẽ tập trung các kiến thức tổng quan của Oauth 1.0 và Oauth 2.0

* Chương 3: THỰC NGHIỆM

Phần này là phần trọng tâm đề tài của chúng em. Ở đây, chúng em sẽ mô tả đầy đủ về cách thức chia sẽ tài quyền qua giao thức Oauth 2.0 và có demo cụ thể. Trong demo chúng em sẽ sử dụng ngôn ngữ chính là Java được viết trên nền tảng Spring Boot để buil hệ thống có sử dụng giao thức Oauth 2.0.

* Chương 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
* Phần cuối cùng, chúng em sẽ tóm tắt lại toàn bộ quá trình triển khai đề tài và nêu ra một hướng phát triển cụ thể cho giao thức Oauth
* Hướng phát triển nhóm em muốn nói tới là: Oauth 2.1 – Giao thức này sẽ phát triển trong tương lai.

# **CHƯƠNG 2. TÌM HIỂU OAUTH 1.0 VÀ OAUTH 2.0**

1. **Lịch sử ra đời**

Năm 2006, Facebook phát triển hệ thống OpenID phục vụ cho đăng nhập các ứng dụng trong hệ thống của Facebook – được coi là chuẩn Oauth đầu tiên, yêu cầu người dùng phải cung cấp thông tin cá nhân – dây cũng chính là điểm yếu của nó. Các ông hoàng của mạng xã thấy vậy đã ngồi lại với nhau và phát triển hệ thống Oauth lên một tầng cao mới.

Năm 2010, IETF – Tổ chức quản lý tiêu chuẩn mạng Internet đã phát hành phiên bản chính lần đầu tiên của Oauth 1.0. Sau đó, một lỗi bảo mật nghiêm trọng với cái tên Session Fixation xảy ra trên Oauth 1.0, cho phép các Hacker giả mạo ứng dụng bên thứ ba để trao quyền truy cập vào tài khoản và dữ liệu người dùng.

Năm 2012, Oauth 2.0 ra đời tuy vẫn còn nhiều lỗi bảo mật như: dùng chorm hack facebook nhưng hiện vẫn đang được sử dụng khá rộng rãi.

1. **Khái niệm tổng quát**

OAuth là một giao thức hoặc khung ủy quyền tiêu chuẩn cung cấp cho các ứng dụng khả năng truy cập được chỉ định an toàn. Ví dụ: bạn có thể nói với Facebook rằng ESPN.com có ​​thể truy cập hồ sơ của bạn hoặc đăng cập nhật lên dòng thời gian của bạn mà không cần phải cung cấp cho ESPN mật khẩu Facebook của bạn. Điều này giảm thiểu rủi ro theo một cách chính: Trong trường hợp ESPN bị vi phạm, mật khẩu Facebook của bạn vẫn an toàn.

OAuth không chia sẻ dữ liệu mật khẩu mà thay vào đó sử dụng mã thông báo ủy quyền để chứng minh danh tính giữa người tiêu dùng và nhà cung cấp dịch vụ. OAuth là một giao thức xác thực cho phép bạn phê duyệt một ứng dụng tương tác với ứng dụng khác mà không cần cung cấp mật khẩu của bạn.

OAuth là từ ghép của O – Open, và Auth – tượng trưng cho 2 nghĩa:

* Authentication: Xác thực người dùng.
* Authorization: Người dùng ủy quyền cho ứng dụng truy cập tài nguyên của họ.

1. **Giải thích Oauth**

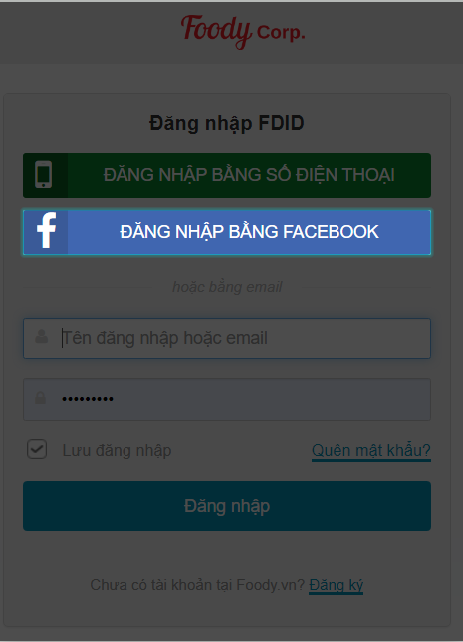
Oauth là cả về ***ủy quyền (Authorization)*** và ***xác thực (Authentication).*** Ủy quyền là yêu cầu cho phép bạn làm mọi công việc. Còn xác thực là việc chứng minh bạn là người chính xác. Oauth không chuyển dữ liệu xác thực giữa người tiêu dùng (consumer) và nhà cung cấp dịch vụ (service provider) mà thay vào đó là việc gửi mã mã thông báo (token) ủy quyền do nhà cung cấp dịch vụ tạo ra thay vì phải chia sẽ thông tin đăng nhập.

Sự phổ biến mà tôi đã thấy được sử dụng trong khi nghiên cứu OAuth là: “chìa khóa valet cho xe hơi của bạn”. Khóa valet cho phép người phục vụ khởi động và di chuyển xe nhưng không cho họ quyền truy cập vào cốp xe hoặc hộp đen của xe.

Mã thông báo OAuth giống như khóa valet. Là người dùng, bạn có thể nói với người tiêu dùng những gì họ có thể sử dụng và những gì họ không thể sử dụng từ mỗi nhà cung cấp dịch vụ. Bạn có thể cung cấp cho mỗi người tiêu dùng một khóa valet khác nhau và họ không bao giờ có đầy đủ tính năng của một cái khóa hoặc bất kỳ dữ liệu riêng tư nào cho phép họ truy cập vào đầy đủ tính năng của khóa.

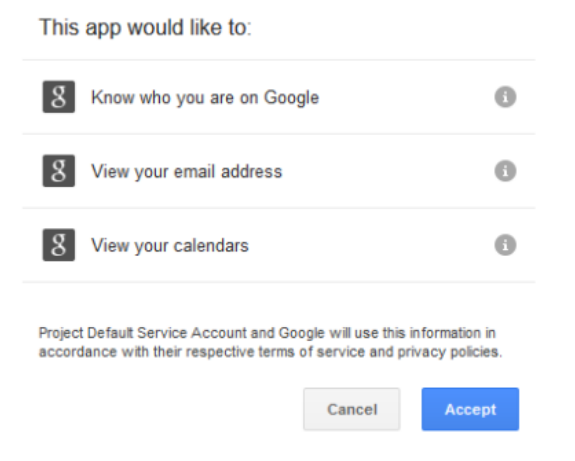
1. **Một vài ví dụ về Oauth**

* Ví dụ 1:  Ví dụ đơn giản nhất về OAuth đang hoạt động là một trang web nói rằng hey hey, bạn có muốn đăng nhập vào trang web của chúng tôi bằng thông tin đăng nhập của các nhà cung cấp dịch vụ khác không (Facebook)? Trong kịch bản này, hãy coi trang web đó là người tiêu dùng (consumer) - muốn biết rằng người dùng (*user*) là cùng một người trên cả hai trang web và đã đăng nhập thành công cho nhà cung cấp dịch vụ - đó là trang web của người dùng (*user*) đăng nhập ban đầu, chứ không phải là người tiêu dùng (consumer).
* Ví dụ 2: Ứng dụng Facebook là một ví dụ về trường hợp sử dụng OAuth tốt nhất. Giả sử bạn đang sử dụng một ứng dụng trên Facebook và nó yêu cầu bạn chia sẻ hồ sơ và hình ảnh của mình. Trong trường hợp này, Facebook là nhà cung cấp dịch vụ: nó có dữ liệu đăng nhập và hình ảnh của bạn. Ứng dụng là người tiêu dùng và là người dùng, bạn muốn sử dụng ứng dụng để làm gì đó với hình ảnh của mình. Bạn đặc biệt đã cấp cho ứng dụng này với quyền truy cập vào ảnh của bạn, mà OAuth đang quản lý ở chế độ nền.



Hình 2: Ví dụ xác thực Oauth qua nhà cung cấp dịch vụ Facebook

* Ví dụ 3: Các thiết bị nhà thông minh như: máy nướng bánh mì, máy điều hòa, hệ thống bảo mật,… có thể sử dụng một số loại dữ liệu đăng nhập để đồng bộ với nhau và cho phép bạn quản trị chúng từ trình duyệt hoặc thiết bị khác. Các thiết bị này sử dụng những gì có từ OAuth gọi là **ủy quyền bí mật**. Điều đó có nghĩa là họ nắm giữ thông tin khóa bí mật, vì vậy bạn không cần phải đăng nhập nhiều lần.
* Ví dụ 4: Trong ảnh chụp màn hình ở dưới, bạn có thể thấy hộp thoại xác nhận với Google sau khi được chuyển hướng từ ứng dụng bên thứ 3 (có thể là một số ứng dụng lịch trong trường hợp này) đến các máy chủ của Google, yêu cầu quyền truy cập vào tài khoản Google của người dùng (tên, email , lịch,…) thay mặt cho người dùng. Sau khi được chuyển hướng đến Google (hoặc bất kỳ máy chủ OAuth nào) nếu bạn chưa đăng nhập, bạn sẽ được nhắc làm như vậy, sau khi chấp nhận quyền truy cập được yêu cầu bạn sẽ được chuyển hướng trở lại ứng dụng của bên thứ 3. Sau đó ứng dụng này có thể truy cập vào các tài nguyên “được phép” từ tài khoản Google của bạn bằng cách sử dụng **access token.**



Hình 3: Ví dụ xác thực Oauth qua nhà cung cấp dịch vụ Google

1. **OAUTH 1.0**
   1. Một vài định nghĩa:

**Nhà cung cấp dịch vụ** (*Service Provider*): là một ứng dụng được cho phép truy cập được thông qua.

**Người dùng** (*User*): là một cá nhân có tài khoản với nhà cung cấp dịch vụ.

**Người tiêu dùng** (*Consumer*): là một trang web hoặc ứng dụng sử dụng Oauth để truy cập nhà cung cấp dịch vụ thay mặt cho người dùng.

**Tài nguyên được bảo vệ** (*Protected Resource*): là dữ liệu được kiểm soát bởi nhà cung cấp dịch vụ mà khách hàng có thể truy cập thông qua xác thực.

**Nhà phát triển** (*Consumer Developer*): là cá nhân hay tổ chức thực hiện như khách hàng.

**Khóa tiêu dùng** (*Consumer Key*): là một giá trị được sử dụng để chứng thực khách hàng với nhà cung cấp dịch vụ.

**Giá trị bí mật** (*Consumer Secret*): là một giá trị bí mật được khách hàng sử dụng để thiết lập quyền sở hữu khóa tiêu dùng.

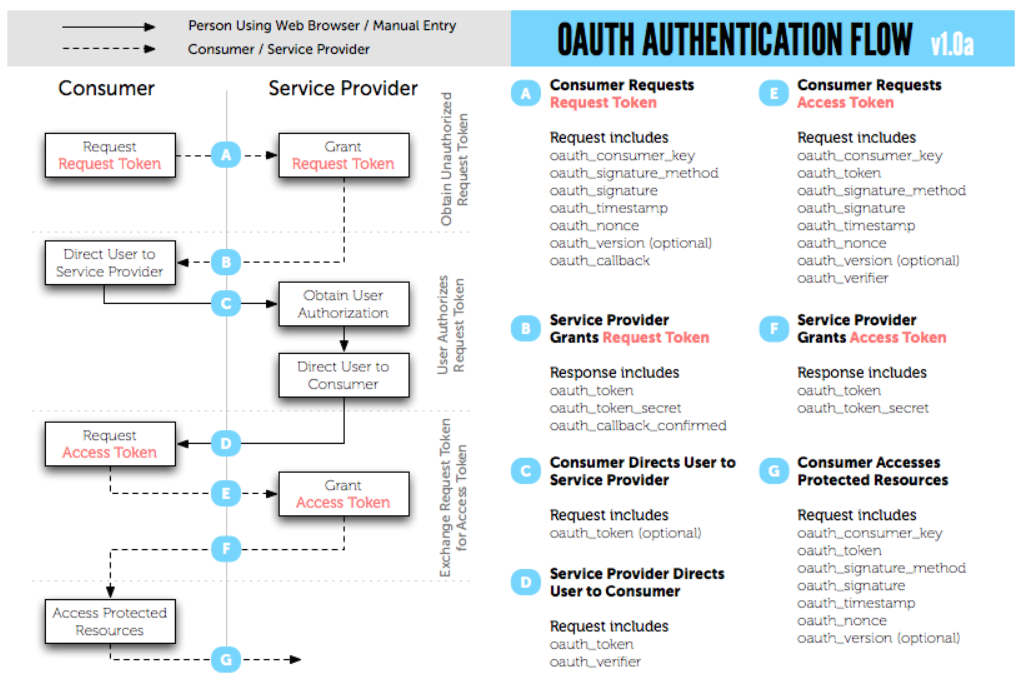
**Mã thông báo yêu cầu** (*Request Token*): là một giá trị được khách hàng sử dụng để nhận ủy quyền từ người dùng và đổi lấy mã thông báo truy cập.

**Mã thông báo truy cập** (*Access Token*): là một giá trị được khách hàng sử dụng để có quyền truy cập vào tài nguyên được bảo vệ thay mặt cho người dùng, thay vì phải sử dụng thông tin đăng nhập của nhà cung cấp dịch vụ.

**Mã thông báo bí mật** (*Token Secret*): là một giá trị bí mật được sử dụng cho khách hàng để thiết lập quyền sở hữu một Token được cấp.

**Tham số giao thức OAuth**: là các tham số có tên bắt đầu bằng: *oauth****\_***

* 1. Xác thực với Oauth 1.0:
     1. Khái niệm: Xác thực Oauth là quá trình User cấp quyền truy cập vào tài nguyên được bảo vệ (protected resources) của họ mà không cần phải chia sẻ thông tin đăng nhập của họ với người tiêu dùng (consumer). Oauth sử dụng Token do nhà cung cấp dịch vụ (provider) tạo ra thay vì chia sẻ thông tin đăng nhập của User trong các yêu cầu của tài nguyên bảo vệ (protected resources). Quá trình sử dụng hai loại mã token:
* Request Token: Được sử dụng bởi Consumer để yêu cầu User cho phép truy cập vào Protected Resources. Request Token yêu cầu do người dùng ủy quyền được trao đổi với Access Token, chỉ được sử dụng một lần và không được sử dụng cho bất kỳ mục đích nào khác. Điều này được muốn nói lên rằng Request Tokens có giới hạn về thời gian và phạm vi hoạt động.
* Access Token: Được sử dụng bởi Consumer để truy cập Protected Resources thay mặt cho User. Access Tokens có thể giới hạn quyền truy cập vào một số Protected Resources nhất định và có thể có thời gian giới hạn. Service Providers nên cho phép Users thu hồi Access Tokens. Chỉ Access Token mới được sử dụng để truy cập Protect Resources.
  + 1. Xác thực OAuth được thực hiện theo ba bước:
* Consumer có được Request Token trái phép.
* User ủy quyền Request Token.
* Consumer trao đổi Request Token cho Access Token.

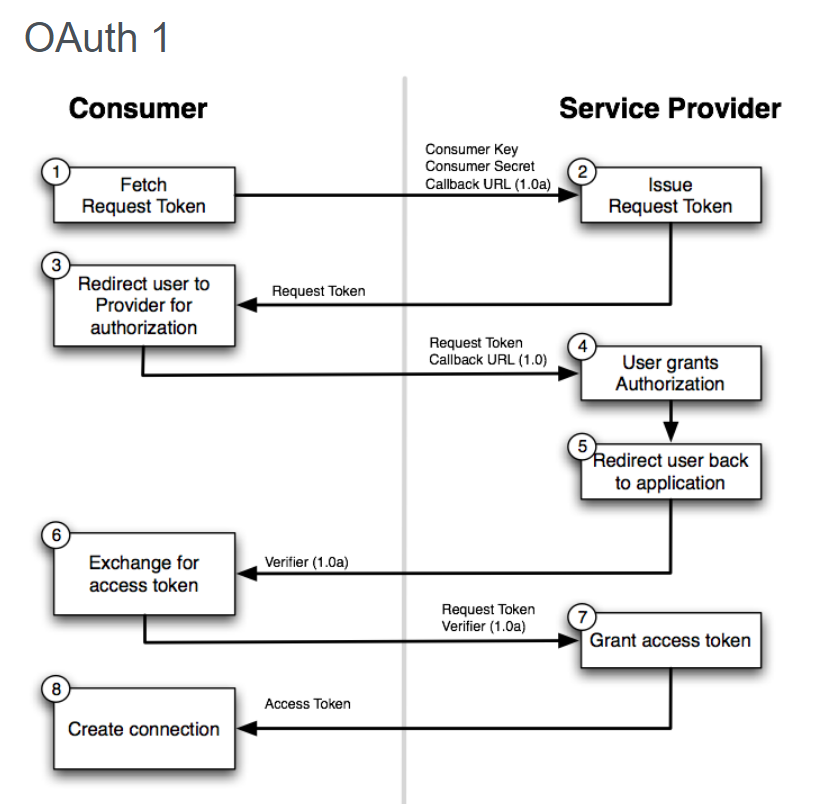


Hình 4: Sơ đồ xác thực của OAuth 1.0

* 1. Nhận mã thông báo yêu cầu trái phép

Consumer có được mã thông báo yêu cầu trái phép bằng cách yêu cầu Service Provider phát hành Token. Mục đích duy nhất của Request Token là nhận được sự chấp thuận của User và chỉ có thể được sử dụng để có được Access Token. Quá trình tạo Request Token diễn ra như sau:

* Consumer có được mã thôn báo yêu cầu (request token)
* Service Provider phát hành Request Token trái phép
  1. Lấy quyền của người dùng
* Consumer hướng User đến Service Provider
* Service Provider xác thực User và có được sự đồng ý
* Service Provider hướng User quay lại với Consumer
  1. Nhận mã thông báo truy cập
* Consumer yêu cầu Access Token
* Service Provider cấp Access Token



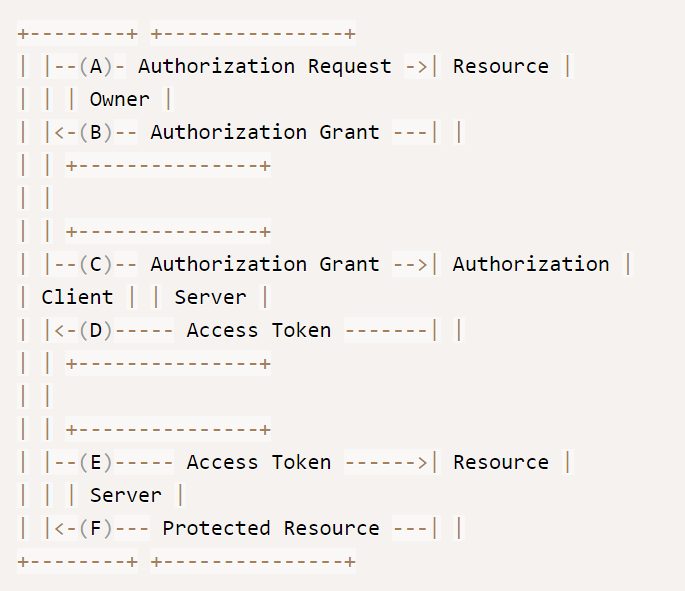
Hình 5: Sơ đồ thông số luồng xử lí có trong Oauth 1.0

1. **OAUTH 2.0**
   1. Khái niệm:

* Oauth 2.0 là bản nâng cấp của giao thức chứng thực Oauth 1.0
* Nó cho phép chia sẻ tài nguyên được lưu trữ trên một trang này sang trang khác mà không cần sử dụng thông tin đăng nhập của họ . Nó sử dụng mã thông báo tên người dùng và mật khẩu để thay thay thế.
  1. Các Role (vai trò) trong mô hình Oauth 2.0
* **Resource owner:** là đối tượng có khả năng cấp quyền truy cập tới tài nguyên (resource) được bảo vệ hay còn gọi là chủ sở hữu tài nguyên.
* **Resource server:** là một server lưu trữ các tài nguyên, có khả năng xử lý các yêu cầu truy cập tới tài nguyên được bảo vệ.
* **Client:** Ứng dụng muốn truy cập vào protected resource với tư cách của người sở hữu tài nguyên.
* **Authorization server:** là server chuyên cấp access token cho client sau khi resource owner đồng ý cấp phép cho client truy xuất vào resource được bảo vệ
  1. Luồng xử lý của giao thức



Hình : : Mô hình xử lí của giao thức Oauth 2.0



Hình 7: : Luồng xử lí truy cập của Oauth 2.0

* (A) Ứng dụng client yêu cầu resource owner cấp quyền
* (B) Client nhận được sự cấp phép từ resource owner tùy theo loại grant trong yêu cầu. Nếu được chứng thực, client sẽ được cấp một authorization grant
* (C) Client yêu cầu Authorization server xác thực authorization grant từ resource owner.
* (D) Authorization server sẽ kiểm tra và xác nhận authorization grant. Nếu được xác thực, Client sẽ nhận được access token
* (E) Client request resource từ resource server, dính kèm access token trong request.
* (F) Resource Server sẽ kiểm tra tính hợp lệ của access token. Nếu token hợp lệ, resource theo yêu cầu sẽ được cấp phát.
  1. Authorization Grant

Authorization grant là một chứng nhận xác định những resource nào đã được cấp quyền truy cập bởi chủ sở hữu của resource. Nó được sử dụng bởi ứng dụng client để đổi lấy access token. Có 4 loại authorization grant:

* Authorization code
* Implicit
* Resource owner password credential
* Client credential
  + 1. Authorization code

Authorization code được cấp phát trong mô hình mà server ủy quyền đóng vai trò trung gian giữa client và resource owner. Client sẽ chuyển resource owner tới server ủy quyền. Sau khi resource owner đồng ý, server ủy quyền sẽ trả về cho clien authorization code

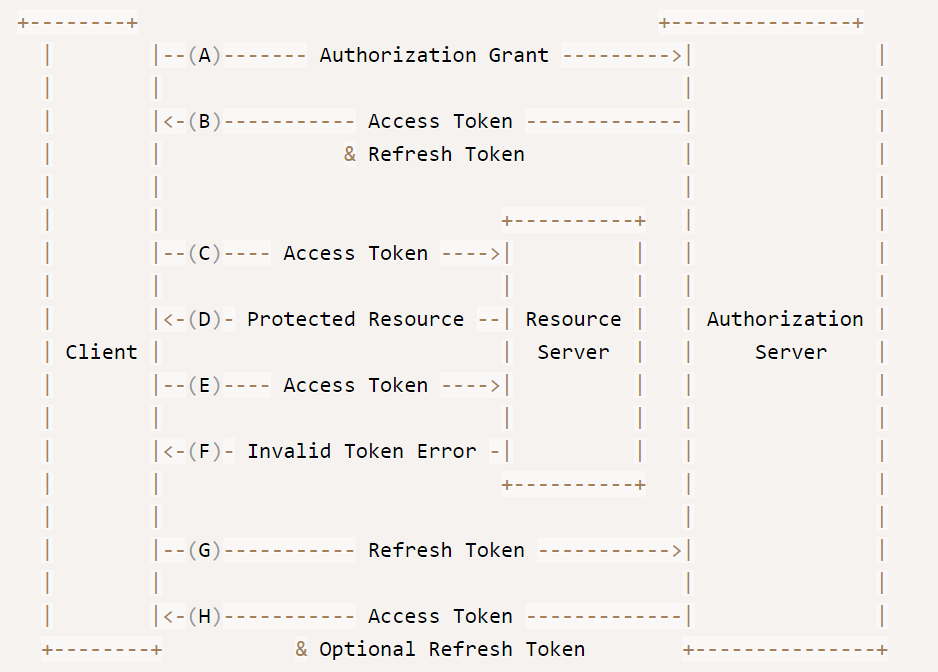
* + 1. Implicit

Implicit grant là sự bản đơn giản hóa của Authorization code, thường được dùng trên các ứng dụng không có khả năng bảo mật authorization grant. Ví dụ như web browser hoặc ứng dụng mobile. Thay vì tạo ra các authorization code, client được nhận trực tiếp access token. Do sự đơn giản hóa của nó nên implicit grant dễ bị tấn công, và các vấn đề về bảo mật.

* + 1. Resource owner password credential

Đây là phương thức xác thực trong đó ứng dụng clieownernt sẽ dùng username và password của người dùng/ resource owner để xác thực với server ủy quyền. Sau khi xác thực thì server ủy quyền sẽ trả về cho ứng dụng client access token và refresh token. Do đó ứng dụng client không cần phải lưu thông tin đăng nhập của resource owner. Kiểu chứng thực này chỉ được sử dụng trên những ứng dụng của đối tác có độ tin cậy cao.

* 1. Các khái niệm khác
     1. Access token
* **Access token**: là những chứng nhận dùng để truy cập vào những tài nguyên được bảo vệ. Access token là một chuỗi ký tự, chứa thông tin về chứng nhận được cấp cho ứng dụng client như: thời gian hết hạn, phạm vi tài nguyên được sử dụng.
* Access token có thể dùng để định danh người dùng. nó dùng để lấy thông tin của resource owner hoặc bản thân nó chứa thông tin của resource owner.
* Access token giúp cho việc xác thực đơn giản hơn. 4 cách yêu cầu cấp quyền từ resource owner đều tạo ra access token do đó resource server không cần phải biết phương thức chứng thực nào được sử dụng. Nó chỉ cần biết sử dụng access token là đủ.
  + 1. Refesh token
* **Refresh token**: là chứng chỉ được sử dụng để trao đổi lấy access token. Nó được trả về cùng với authorization grant, và được dùng để yêu cầu cấp mới access token khi access token hiện tại hết hạn.
* Refresh token được sử dụng với server ủy quyền, không sử dụng với resource server.



Hình 8: Làm mới Access token đã hết hạn

* (A): Client gửi authorization grant và yêu cầu server ủy quyền cấp access tocken và refresh token
* (B): Server ủy quyền xác thực client và authorization grant. Nếu hợp lệ, nó sẽ tạo ra access token và refresh token và trả về cho client
* (C): Client yêu cầu truy xuất tới resource được bảo vệ sử dụng access token
* (D): Resource server kiểm tra access token, nếu hợp lệ, thực hiện yêu cầu từ client.
* (E): Bước (C) và (D) được lặp lại cho tới khi access token hết hạn. Nếu client biết access token hết hạn, nó sẽ chủ động chuyển qua bước (G), nếu không nó sẽ gửi request mới.
* (F) Phát hiện access token hết hạn, resource server trả về lỗi access token không hợp lệ.
* (G) Client gửi refresh token lên server ủy quyền và yêu cầu cấp phát access token mới.
* (H) Server ủy quyền kiểm tra thông tin client và refresh token, nếu hợp lệ, nó sẽ cấp phát access token và có thể là cả refresh token mới.
  1. Phạm vi (Scope) của Access token
* Authorization endpoint và Token endpoint cho phép client xác định phạm vi/ giới hạn truy xuất của request thông qua tham số scope. Sau khi được cấp quyền, authorization server sử dụng scope để xác định các giới hạn/ phạm vi được quyền truy cập đối với access token được trả về.
* Scope được thể hiện dưới dạng một chuỗi gồm nhiều scope cách nhau bằng khoảng trắng.
* Nếu scope của access token được cấp phát giống với scope trong request của client thì server có thể bỏ qua tham số scope trong response. Nếu scope khác với trong request thì bắt buộc phải trả về tham số scope trong response.

1. SO SÁNH OAUTH 1.0 VÀ OAUTH 2.0

|  |  |  |
| --- | --- | --- |
| **Giao thức**  **Nội dung** | **Oauth 1.0** | **Oauth 2.0** |
| **Về cách bảo mật** | Bảo mật của giao thức OAuth 1.0 dựa trên giả định rằng một khóa bí mật được nhúng trong ứng dụng khách có thể được giữ bí mật. Tuy nhiên, giả định là ngây thơ. | Trong OAuth 2.0, một ứng dụng khách ngây thơ như vậy được gọi là ứng dụng khách bí mật . Mặt khác, một ứng dụng khách trong một môi trường khó giữ bí mật khóa bí mật được gọi là ứng dụng khách công khai. |
| **Về cách hoạt động** | OAuth 1.0 yêu cầu khách hàng gửi hai mã thông báo bảo mật cho mỗi lệnh gọi API và sử dụng cả hai để tạo chữ ký. Nó yêu cầu các điểm cuối tài nguyên được bảo vệ có quyền truy cập vào thông tin đăng nhập của khách hàng để xác thực yêu cầu. | Chữ ký OAuth 2.0 không bắt buộc đối với các lệnh gọi API thực tế sau khi mã thông báo được tạo. Nó chỉ có một mã thông báo bảo mật. |
| **Về sự phức tạp liên quan đến việc ký từng yêu cầu** | OAuth 1.0 bắt buộc khách hàng phải tạo chữ ký trên mỗi lệnh gọi API đến tài nguyên máy chủ bằng cách sử dụng bí mật mã thông báo. Ở đầu nhận, máy chủ phải tạo lại cùng một chữ ký và máy khách sẽ chỉ được cấp quyền truy cập nếu cả hai chữ ký khớp nhau. | OAuth 2.0 không yêu cầu máy khách cũng như máy chủ tạo bất kỳ chữ ký nào để bảo mật các tin nhắn. Bảo mật được thực thi thông qua việc sử dụng TLS/ SSL (HTTPS) cho tất cả các giao tiếp. |
| **Về giải quyết các ứng dụng khách không có trình duyệt** | Nhiều tính năng của OAuth 1.0 được thiết kế bằng cách xem xét cách ứng dụng khách ứng dụng web tương tác với các tin nhắn gửi đến và gửi đi. Điều này đã được chứng minh là không hiệu quả trong khi sử dụng nó với các máy khách không có trình duyệt như các ứng dụng di động trên thiết bị. | Giải quyết vấn đề này bằng cách cung cấp thêm các luồng ủy quyền phù hợp cho các nhu cầu khách hàng. Không sử dụng bất kỳ giao diện người dùng (web) nào, chẳng hạn như các ứng dụng di động trên thiết bị hoặc dịch vụ API. Điều này làm cho giao thức rất linh hoạt. |
| **Về vai trò** | Giao thức Oauth 1.0 không định rõ vai trò của tất cả các bên liên quan. | Xác định rõ vai trò của tất cả các bên liên quan đến giao tiếp, chẳng hạn như máy khách, chủ sở hữu tài nguyên, máy chủ tài nguyên và máy chủ ủy quyền. Thông số kỹ thuật rõ ràng về phần nào của giao thức dự kiến ​​sẽ được thực hiện bởi chủ sở hữu tài nguyên, máy chủ ủy quyền và máy chủ tài nguyên. |
| **Về thời gian thông báo mã truy cập** | Không có thời gian Time Life. Phần lớn người dùng không biết được rằng khi nào ta sẽ vào trình quản lý và hủy việc cấp ủy quyền này đi cả. | Mã thông báo truy cập trong OAuth 2.0 có thể chứa thời gian hết hạn, giúp cải thiện bảo mật và giảm cơ hội truy cập bất hợp pháp. |
| **Về mã thông báo được làm mới** | Khi hết hạn phiên truy cập Oauth 1.0 phải xin phép ủy quyền tài nguyên của chủ sở hữu một lần nữa… | OAuth 2.0 cung cấp mã thông báo làm mới có thể được sử dụng để nhận mã thông báo truy cập mới khi hết hạn mã hiện tại, mà không phải trải qua toàn bộ quy trình ủy quyền. |
| **Ưu điểm** | Không phải nhập username và password cho mỗi phiên làm việc | Nó có khả năng chia sẻ dữ liệu cho người dùng mà không phải tiết lộ thông tin cá nhân. |
| **Nhược điểm** | Bảo mật kém, phát sinh lỗi bảo mật Session Fixation  Phức tạp khi sinh ra access token chính và tạm thời. | Nếu các trang web yêu thích của bạn được kết nối với tài khoản trung tâm, và tài khoản trung tâm đó bị hack, thì nó dẫn đến hậu quả nghiệm trọng là các trang web bạn liên kết cũng bị hack thay vì một trang web chứa tài khoản trung tâm của bạn. |

Như vậy, qua bảng so sánh trên ta thấy sự khác biệt nhất trên Oauth 2.0 là tập trung chủ yếu vào việc đơn giản hóa phía máy khách (Client) - Giải quyết vấn đề này bằng cách cung cấp thêm các luồng ủy quyền phù hợp cho các nhu cầu khách hàng. Không sử dụng bất kỳ giao diện người dùng (web) nào, chẳng hạn như các ứng dụng di động trên thiết bị hoặc dịch vụ API. Điều này làm cho giao thức rất linh hoạt.

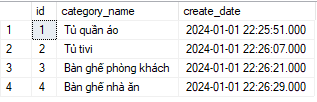
Trong giao thức Oauth 2.0, Client không chỉ được quyền truy xuất dữ liệu nhân danh chủ sở hữu tài nguyên (Resource Owern), mà Client còn có thể nhân danh chính bản thân nó. Qua đó, việc phân quyền cụ thể cho từng Client cũng có thể thực hiện được. Chính vì thế, giao thức Oauth 2.0 là một giao thức hoàn toàn mới và bản phát hành này không tương thích với phiên bản Oauth 1.0.

# **CHƯƠNG 3. THỰC NGHIỆM**

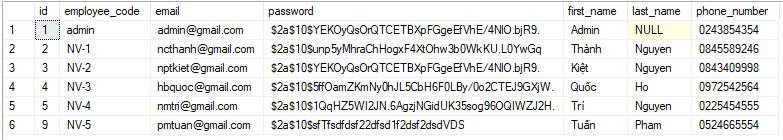
* + - 1. Chuẩn bị
* Hệ điều hành window 10
* Các công cụ hỗ trợ: IntelliJ IDEA 2024.2.4 x64, SQL Server, Node.js
  + - 1. Mô phỏng

Trong phần này chúng em sẽ demo trực tiếp việc chia sẽ tài nguyên bằng giao thức Oauth trên website localhost để mô phỏng quy trình và các bước chia sẽ tài nguyên bằng Oauth. Ở đây chúng em sẽ buil một website local được viết bằng ngôn ngữ lập trình phổ biến là: **Java (viết trên nền tảng Spring Boot)**

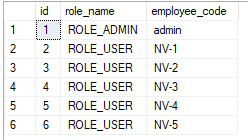
* + - 1. Tạo Database vào hệ thống qua công cụ SQL Server



Hình 9: Cơ sở dữ liệu dùng để chia sẽ tài nguyên

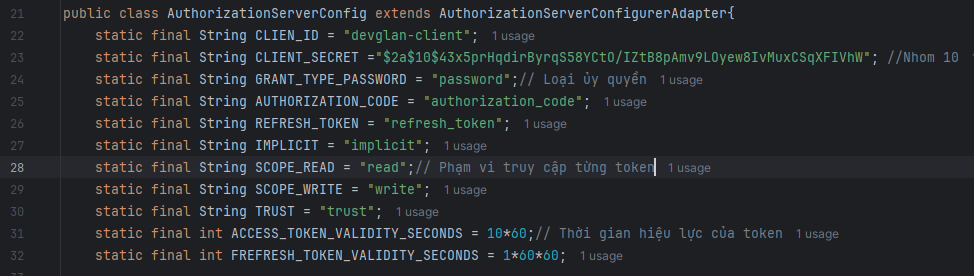


Hình 10: Cở sở dữ liệu dùng để đăng nhập hệ thống

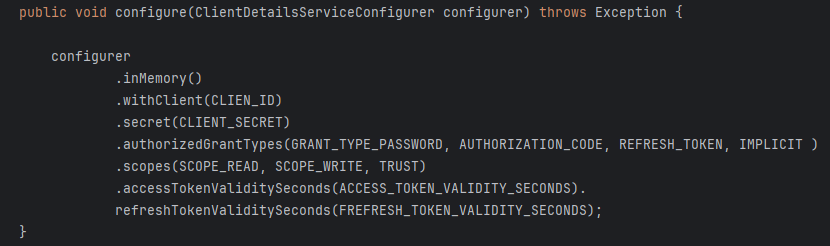


Hình 11: Vai trò của User trong hệ thống

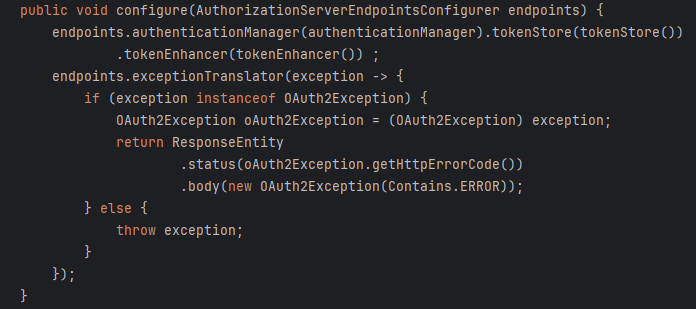
* + - 1. Xây dụng trang web local trước khi tiến hành chứng thực bằng Oauth 2.0
  1. Code trên Admin



Hình 12: Code cấu hình máy chủ ủy quyền

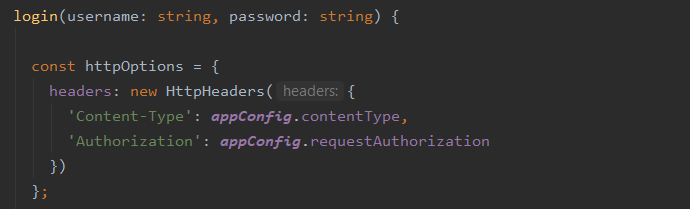


Hình 13: Code cấu hình chi tiết dịch vụ phía client

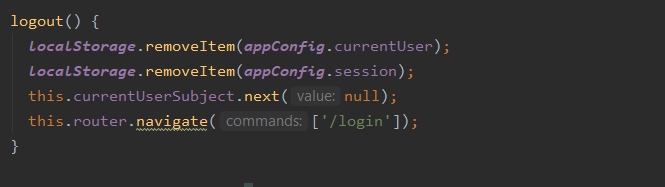


Hình 14: Code cấu hình kết thúc ủy quyền của máy chủ

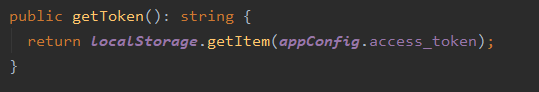
* 1. Code trên Frontend



Hình 15: Code cấu hình phần login



Hình 16: Code cấu hình phần logout



Hình 17: Code lấy token



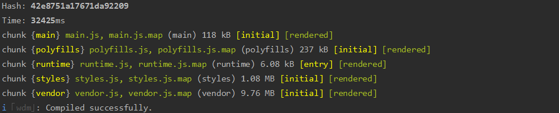
Hình 18: Code làm mới token

* + - 1. Triển khai chứng thực trên web local đã buil bằng Oauth 2.0
  1. Chạy file Admin



Hình 19: Debug code phần admin

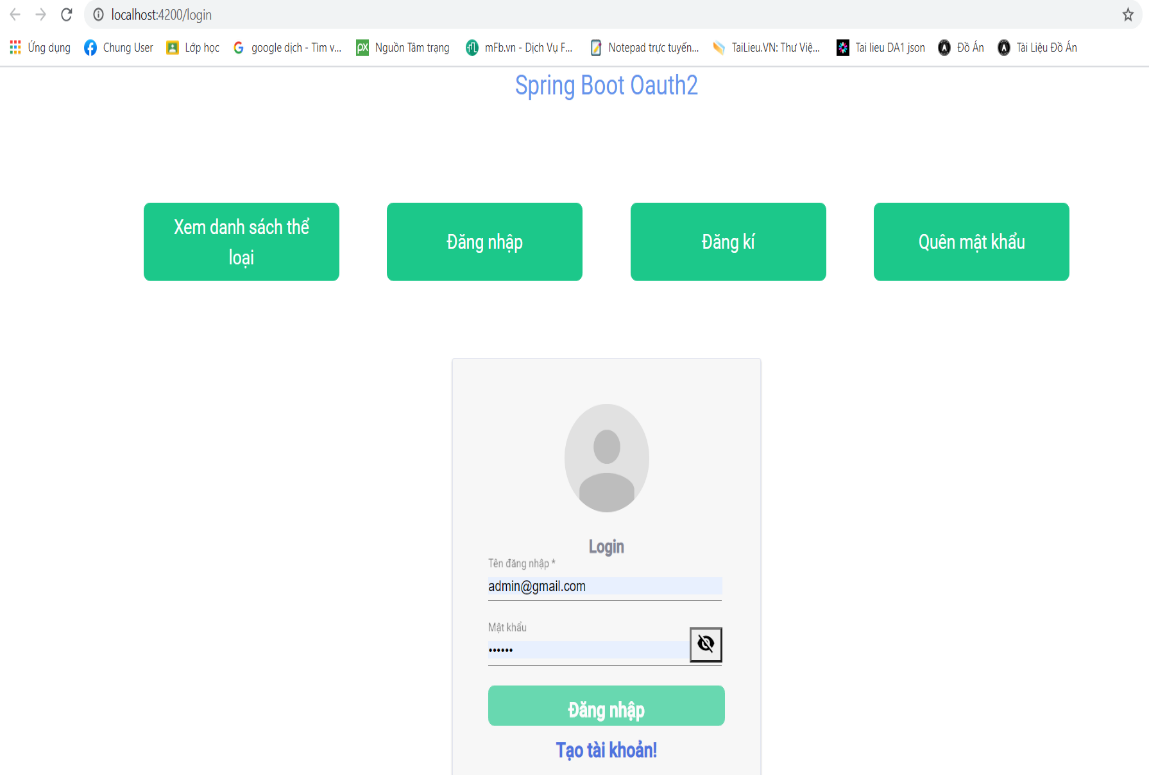
* 1. Chạy file Frontend



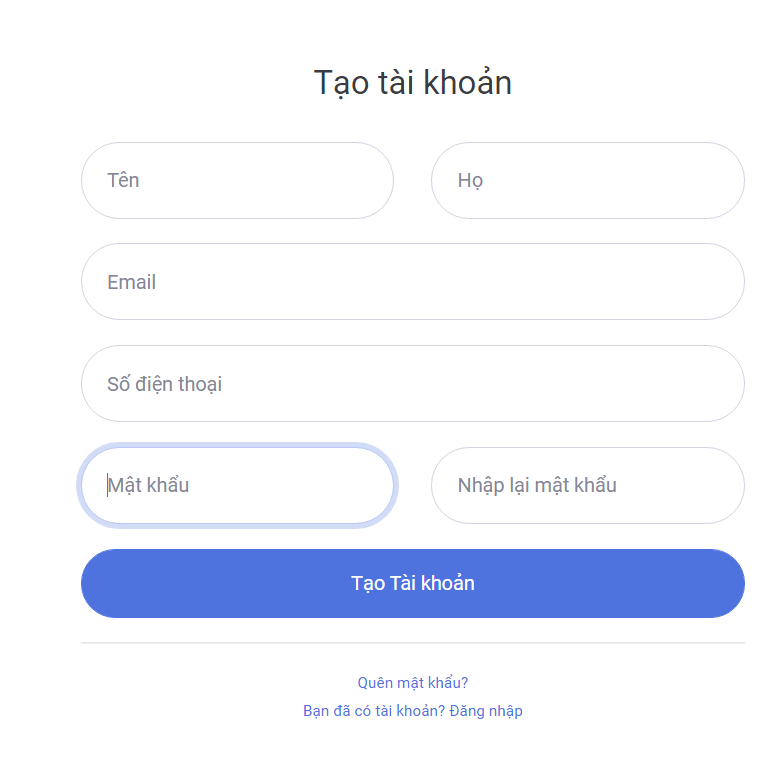
Hình 20: Debug code phần frontend

Sau khi debug thành công code phần frontend, chúng ta sẽ nhận được một đường dẫn kết nối tới web local

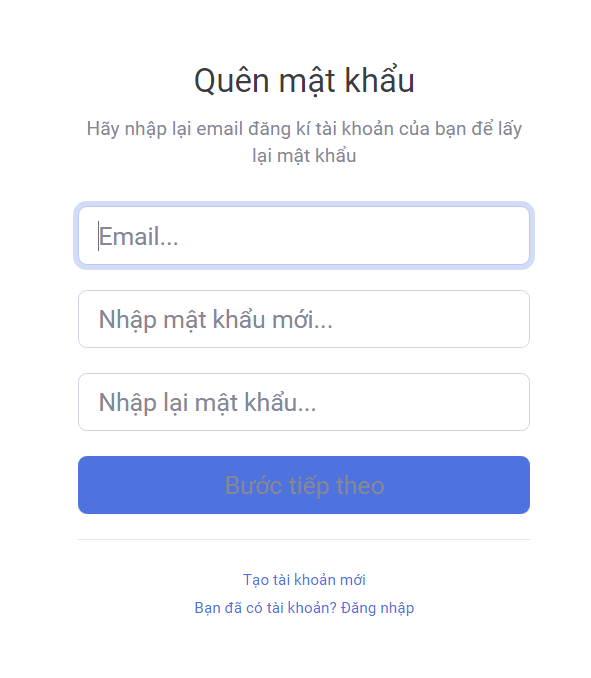
* 1. Tiến hành kết nối tới web localhost đã được thực thi



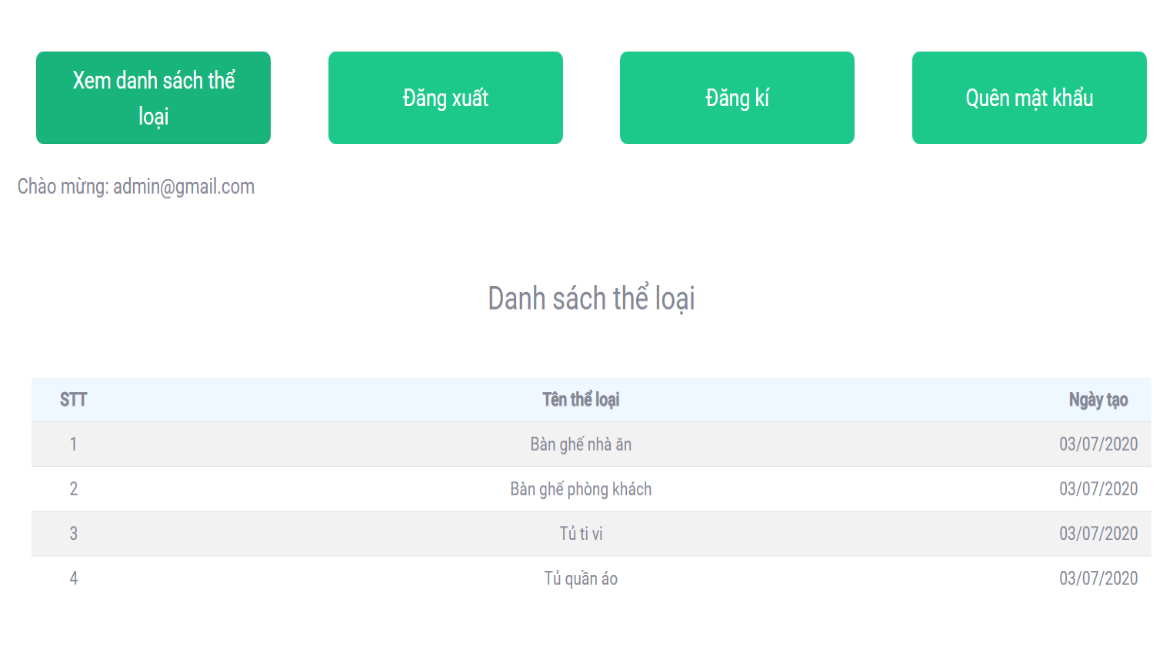
Hình : Giao diện của web localhost và phần login



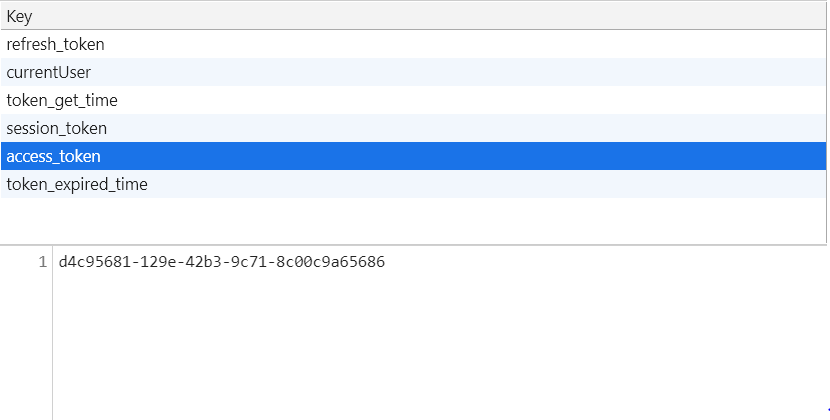
Hình 22: Giao diện form đăng kí tài khoản



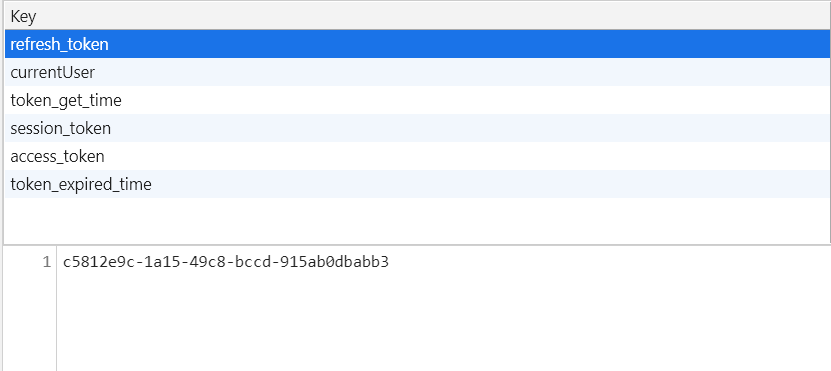
Hình 23: Giao diện form quên mật khẩu



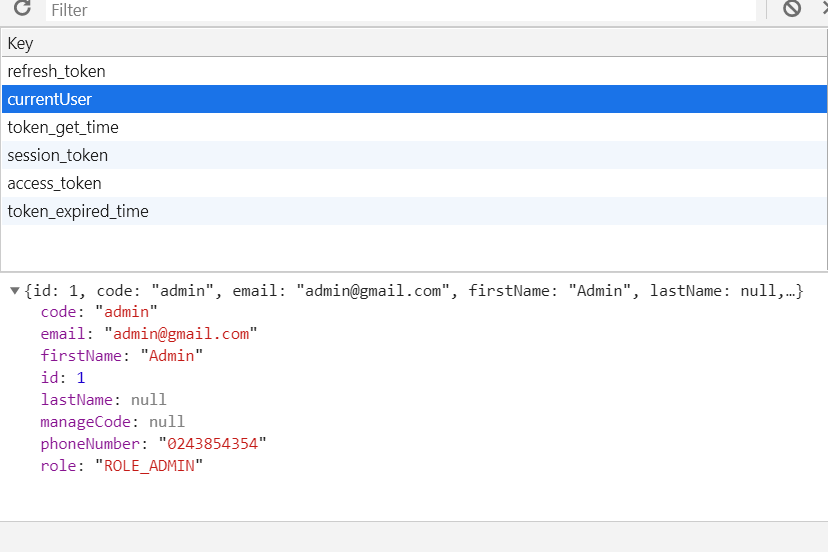
Hình : Giao diện sau khi đăng nhập thành công



Hình 25: Token được sinh ra



Hình 26: Làm mới Access token



Hình 27: Thông tin User đăng nhập

**Mã user:** code

**Email user:** admin@gmail.com

**Tên user:** Admin

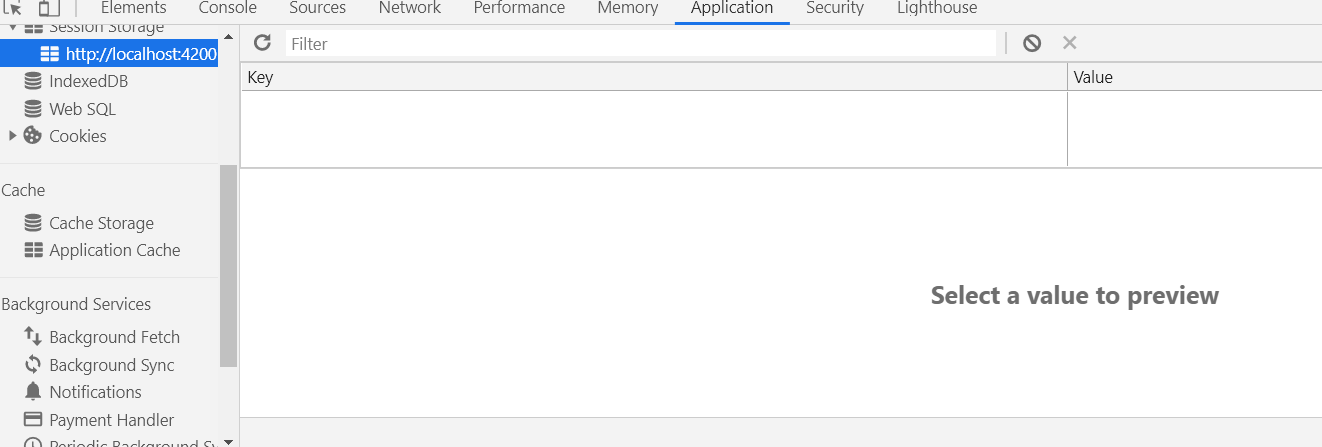
**Họ user:** Null

**ID user:** 1

**SĐT user:** 0243854354

**Vai trò user:** ROLE\_ADMIN

Toàn bộ thông tin trên sẽ được lưu trong Local Storage sau khi bạn đã đăng nhập thành công.



Hình 28: Giao diện khi chưa đăng nhập của phần local storage

Ngoài ra, khi bạn chưa đăng nhập hoặc đăng nhập không thành công, thì chủ sở hữu tài nguyên sẽ không chia sẽ được tài nguyên cho bạn. Phần lưu trữ cục bộ sẽ không nhận được access token từ chủ sở hữu tài nguyên và phần thông tin đăng nhập của user đó.

# **CHƯƠNG 4. TỔNG KẾT VÀ HƯỚNG PHÁT TRIỂN**

1. Những vấn đề làm được

* Triển khai được ý tưởng như đề tài đã đưa ra thành một sản phẩm thực tế có ứng dụng cao.
* Tìm hiểu framework Spring Boot, framework HeidiSQL
* Dowload được công cụ hỗ trợ viết web local

1. Những vấn đề chưa làm được

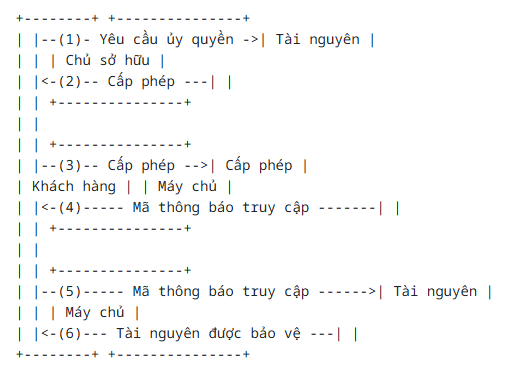
* Về mặt thẩm mỹ của sản phẩm thực tế chưa cao.
* Thao tác các chức năng chưa hoàn thiện.
* Có một số phần cần thiết trong hệ thống nhưng nhóm đã không đề cập đến.
* Còn nhiều thiếu sót.

1. Hướng phát triển

Giao thức ủy quyền Oauth 2.0 đã phát triển rộng rãi và được mọi người tin tưởng dùng – là một giao thức bảo mật không chia sẻ thông tin tài khoản đối với phía khách hàng. Và hiện nay nó đã phát triển lên trên điện toán đám mây.

Trong thời gian sắp tới, thế giới sẽ nhận được một Giao thức ủy quyền mới là Oauth 2.1 phát triển từ Oauth 2.0 (*theo bản dự thảo Internet về khung cấp giao thức ủy quyền Oautn 2.1 tháng 11 năm 2024*). Sau đây, nhóm em xin nói đôi chút về hướng phát triển theo bản dự thảo:

* Giao thức ủy quyên Oauth 2.1 cho phép ứng dụng bên thứ ba có đượ quyền truy cập hạn chế vào dịch vụ HTTP, thay mặt cho chủ sở hữu tài nguyên bằng cách sắp xếp tương tác phê duyệt giữa chủ sở hữu tài nguyên và dịch vụ HTTP hoặc cho phép ứng dụng bên thứ ba có được quyền truy cập thay mặt cho chính nó. Đặc điểm của kĩ thuật này là thay thế và hủy bỏ.
* OAuth 2.1 giải quyết những vấn đề này bằng cách giới thiệu lớp ủy quyền và tách biệt vai trò của khách hàng với vai trò của chủ sở hữu tài nguyên. Trong OAuth 2.1, ứng dụng khách yêu cầu quyền truy cập vào tài nguyên do chủ sở hữu tài nguyên kiểm soát và được lưu trữ bởi máy chủ tài nguyên và được cấp một bộ thông tin xác thực khác với thông tin của chủ sở hữu tài nguyên.
* Thay vì sử dụng thông tin đăng nhập của chủ sở hữu tài nguyên để truy cập vào tài nguyên được bảo vệ, máy khách nhận được mã thông báo truy cập - một chuỗi biểu thị phạm vi cụ thể, thời gian tồn tại và các thuộc tính truy cập khác như Oauth 2.0. Thì mã thông báo truy cập được máy chủ ủy quyền cấp cho khách hàng của bên thứ ba với sự chấp thuận của chủ sở hữu tài nguyên. Máy khách sử dụng mã thông báo truy cập để truy cập vào các tài nguyên được bảo vệ do máy chủ tài nguyên lưu trữ.
* **Ví dụ:** người dùng cuối (*chủ sở hữu tài nguyên*) có thể cấp cho một dịch vụ in (*máy khách*) quyền truy cập vào ảnh được bảo vệ của mình được lưu trữ tại dịch vụ chia sẻ ảnh (*máy chủ tài nguyên*) mà không cần chia sẻ tên người dùng của mình và mật khẩu với dịch vụ in. Thay vào đó, có sự xác thực trực tiếp bằng máy chủ được dịch vụ chia sẻ ảnh (*máy chủ ủy quyền*) tin cậy, nơi cấp thông tin xác thực dành riêng cho dịch vụ in ấn (*mã thông báo truy cập*).



Hình 29: Sơ đồ luồng xử lí của bản dự thảo Oauth 2.1

Như vậy, là bản dự thảo tháng 07 năm 2024 đã mở ra một diện mạo mới về giao thực xác thực ủy quyền.

**TÀI LIỆU THAM KHẢO**

[1] Oauth – Wikipedia: <https://en.wikipedia.org/wiki/OAuth>

[2] RFC5849 – Oauth 1.0: <https://tools.ietf.org/html/rfc5849#appendix-A>

[3] RFC6749 – Oauth 2.0: <https://tools.ietf.org/html/rfc6749>

[4] Bản dự thảo tháng 11 năm 2024 – Oauth 2.1: [https://datatracker.ietf.org](https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/)