**Tài liệu thiết kế**

1. **Giới thiệu**
2. **Mục đích tài liệu**

Tài liệu thiết kế này nhằm mục đích cung cấp một cái nhìn tổng quan và chi tiết về thiết kế hệ thống Single Sign-On (SSO) cho Dextrends. Tài liệu sẽ trình bày các khía cạnh kỹ thuật quan trọng, bao gồm luồng xác thực, quản lý phiên làm việc, cấu trúc cơ sở dữ liệu và các giao diện lập trình ứng dụng (API) liên quan.

1. **Phạm vi tài liệu**

Tài liệu thiết kế này tập trung vào các khía cạnh kỹ thuật phục vụ demo của hệ thống Single Sign-On (SSO) của Dextrends, bao gồm:

* Đăng nhập: Mô tả chi tiết luồng xử lý khi người dùng đăng nhập bằng các phương thức khác nhau (user/email, Facebook, Google).
* Đăng xuất: Giải thích quy trình hủy phiên làm việc khi người dùng đăng xuất.
* Cơ sở dữ liệu: Trình bày sơ đồ cấu trúc cơ sở dữ liệu tổ chức user/account tổng quát.
* API/Endpoint: liệt các API/Endpoint được cung cấp theo chuẩn OAuth để tích hợp 1 hệ thống con vào hệ thống SSO.

Trong phạm vi demo, tài liệu này giả định và không tập trung quá nhiều vào các vấn đề sau:

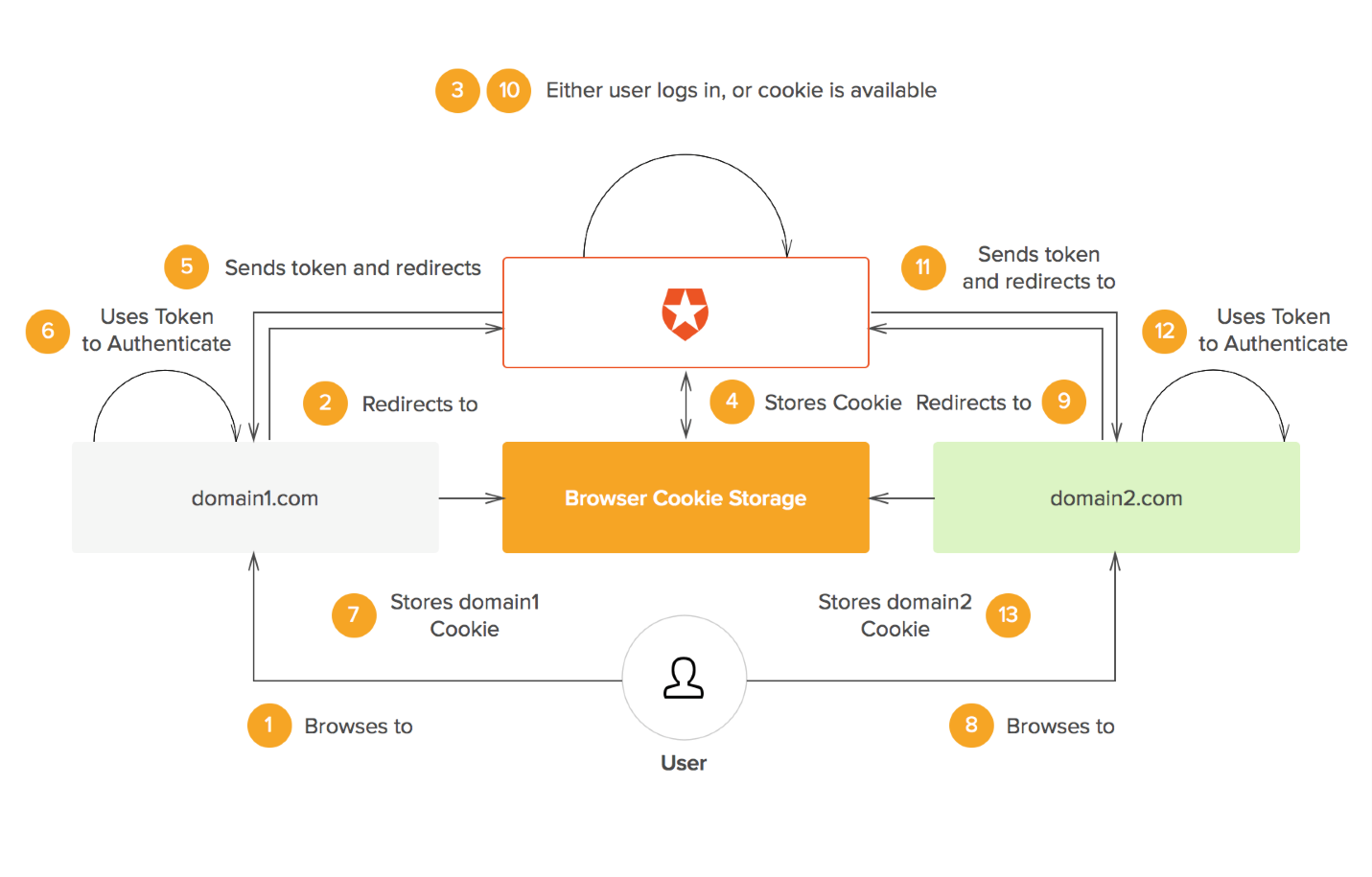
* Sử dụng giao diện người dùng mặc định.
* Không nêu tập trung mô tả cách triển khai kĩ thuật tại các hệ thống con (do các yếu tố về lập trình).
* Do hạn chế của người viết về các công nghệ trên mobile nên sẽ không có demo cho cách tích hợp trên mobile-app.

1. **Đối tượng**

Đối tượng sử dụng tài liệu bao gồm:

* Nhà phát triển: hiểu rõ các yêu cầu kỹ thuật, luồng xử lý, cấu trúc dữ liệu và API cần thiết.
* Software architect: đánh giá tính khả thi, hiệu quả và khả năng mở rộng của mô hình thiết kế SSO, đồng thời đưa ra các quyết định liên quan đến việc lựa chọn công nghệ và tích hợp với các hệ thống khác.

1. **Tổng quan về hệ thống SSO**
2. **Tổng quan luồng hoạt động của mô hình SSO**



Hình 1: Luồng hoạt động tổng quan theo mô hình SSO

Các điểm chính trong mô hình này:

* Sử dụng hệ thống xác thực tập trung cho các hệ thống con, bao gồm các chức năng:
  + Quản lý định danh & truy cập tập trung
  + Quản lý phiên làm việc
  + Hỗ trợ đăng nhập 1 lần trên 1 domain bất kì, và tự động xác thức khi đăng nhập trên các domain khác (Single Sign On - SSO)
* Để tích hợp, các hệ thống con cần:
  + Thực hiện redirect đến giao diện đăng nhập tập trung (thay vì giao diện đăng nhập tại mỗi hệ thống con)
  + Thực thi nghiệp vụ xác thực dựa theo token trả về từ hệ thống xác thực tập trung (thay vì sử dụng hệ thống xác thực truyền thống)

1. **Tổng quan các thành phần chính của mô hình xác thực tập trung**

Trong đề tài, Keycloak được chọn làm framework để phát triển hệ thống xác thực tập trung SSO. Vì vậy các thành phần chính của hệ thống có thể như sau:

1. **Các chức năng chính (trong phạm vi demo)**

* Login theo mô hình SSO
* Logout theo mô hình SSO
* Liên kết tài khoản với account mạng xã hội phổ biến khác (trong phạm vi đề tài này, hỗ trợ demo liên kết theo tài khoản Google)

1. **Tổ chức thông tin user tổng quan**

Trong bài toán đề ra, ngoài việc hỗ trợ xác thực user (hoặc email) và password theo cách truyền thống, hệ thống SSO đề ra đòi hỏi hỗ trợ xác thực qua các nền tảng xã hội phổ biến như Google, Facebook ….

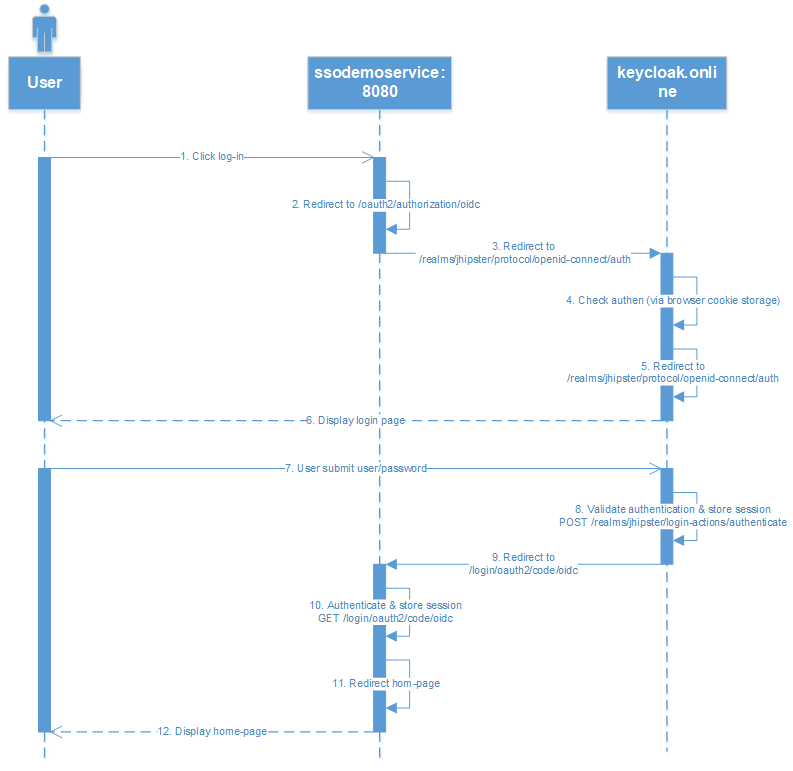
Vì vậy mô hình tổ chức user được đề xuất như sau:



Trong đó:

* **User**: bảng thông tin lưu trữ account chính trong hệ thống của công ty
* **Federated\_identity**: bảng lưu thông tin account từ các nền tảng mạng xã hội (hoặc hệ thống khác) có hỗ trợ liên kết
* 1 User có thể có nhiều thông tin account liên kết

1. **Luồng giao tiếp**
2. **Chức năng login**



**Kịch bản:**

Người dùng truy cập dịch vụ:

* Nếu chưa đăng nhập, hiển thị màn hình đăng nhập với các tùy chọn: email, Facebook, Google.
* Nếu đã đăng nhập, chuyển hướng đến trang chủ của dịch vụ.

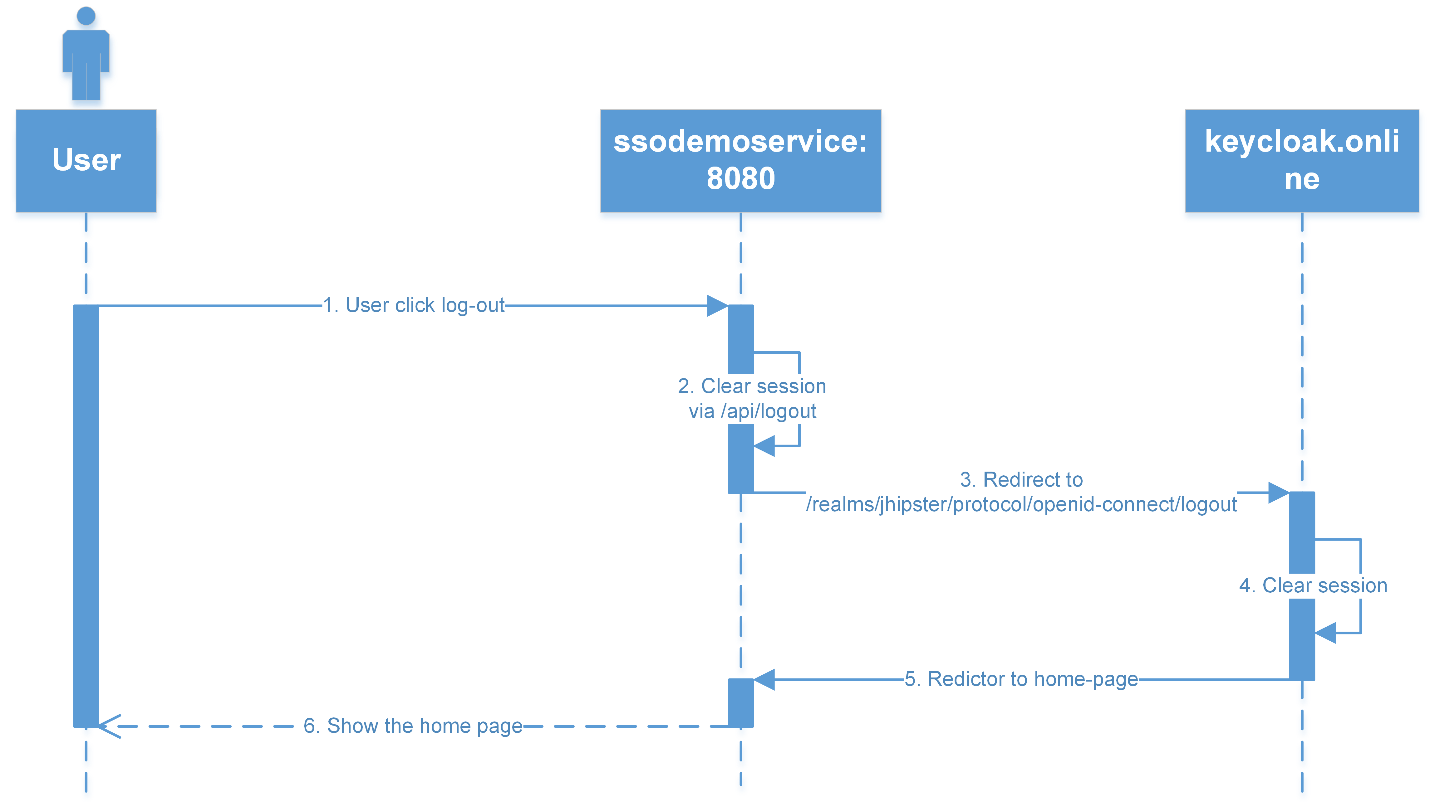
Người dùng đăng nhập:

* Xác thực thông tin đăng nhập.
* Tạo session đăng nhập và lưu trữ thông tin người dùng.

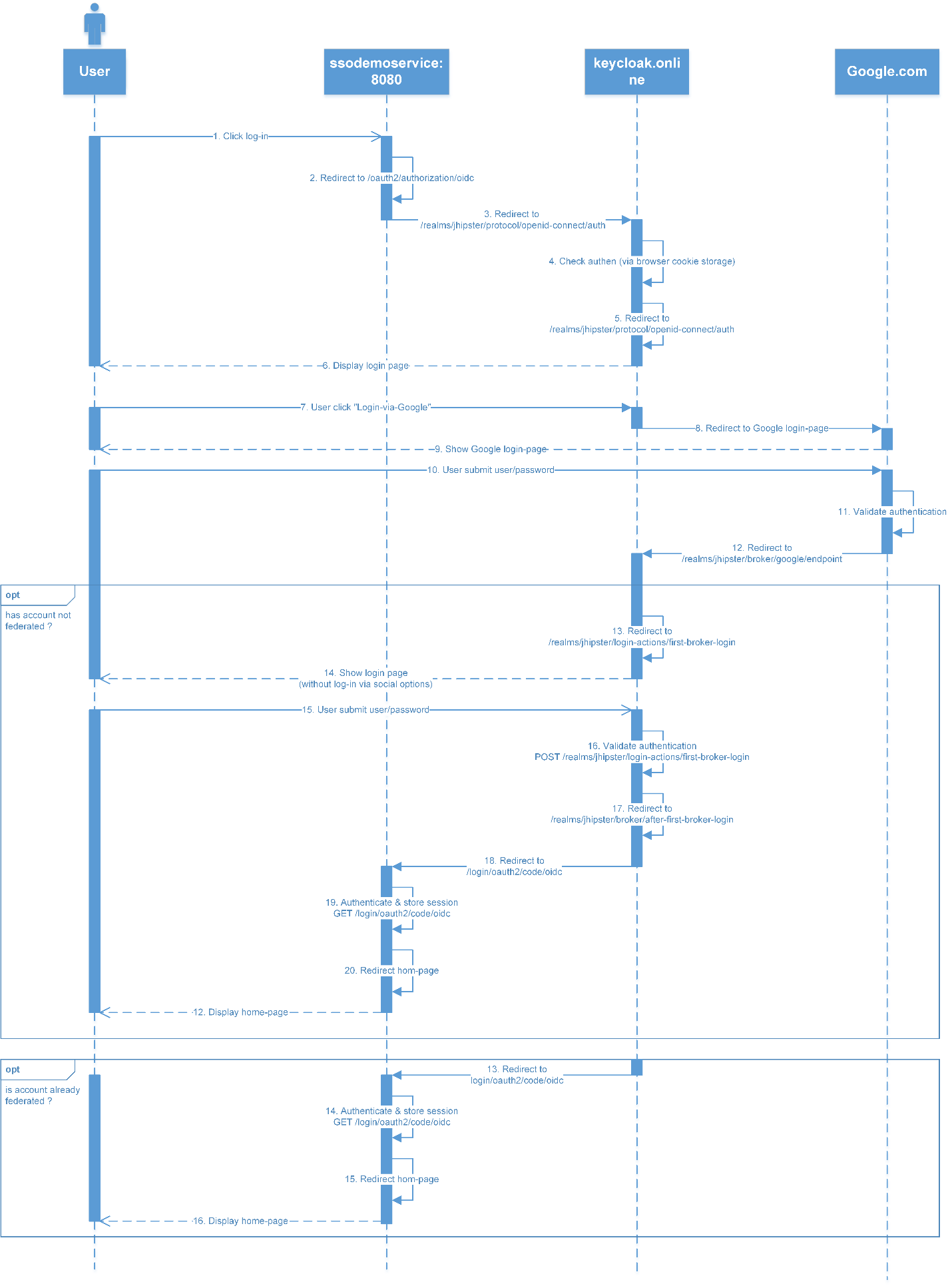
Người dùng truy cập các dịch vụ khác:

* Kiểm tra session đăng nhập.
* Nếu session hợp lệ, cho phép truy cập.
* Nếu session không hợp lệ, yêu cầu đăng nhập lại.

1. **Chức năng logout**



1. **Chức năng liên kết tài khoản**



**Kịch bản:**

Người dùng đăng nhập bằng dịch vụ khác (trong trường hợp này là Google):

* Hệ thống chuyển trới trang đăng nhập của Google.
* Người dùng đăng nhập theo chỉ dẫn của dịch vụ Google
* Hệ thống chuyển lại sang trang đăng nhập SSO (trong TH đăng nhập Google thành công)
* Người đăng nhập với tài khoản gốc
* Hệ thống tự động liên kết tài khoản gốc với tài khoản Google
* Tạo session đăng nhập và chuyển về trang chủ của dịch vụ

1. **Đặc tả API/Enpoint tích hợp**
2. **Login endpoint**

**Request Details:**

* *URL: /realms/{realm}/protocol/{protocal}/auth*
* *HTTP Version: HTTP/1.1*
* *Full URL: http://keycloak.online:9080/realms/jhipster/protocol/openid-connect/auth?response\_type=code&client\_id=ssodemoservice01&scope=openid profile email offline\_access&state=arGmrK8G2lmxQ6bdt9A2N0\_V2In8wB5MOYUjZhUMgNM=&redirect\_uri=http://ssodemoservice01:8080/login/oauth2/code/oidc&nonce=-rhKs5Dn8yFjpMsjfKeXRV4t3-AOw2hhXPY4RerfCF0*

**Example endpoint:**

* *GET /realms/jhipster/protocol/openid-connect/auth?response\_type=code&client\_id=ssodemoservice01&scope=openid%20profile%20email%20offline\_access&state=arGmrK8G2lmxQ6bdt9A2N0\_V2In8wB5MOYUjZhUMgNM=&redirect\_uri=http://ssodemoservice01:8080/login/oauth2/code/oidc&nonce=-rhKs5Dn8yFjpMsjfKeXRV4t3-AOw2hhXPY4RerfCF0 HTTP/1.1*

*Host: keycloak.online:9080*

**Explanation:**

* *URL Parameters:*
  + *response\_type=code: Indicates that the authorization server should return an authorization code.*
  + *client\_id=ssodemoservice01: The client identifier.*
  + *scope=openid profile email offline\_access: The scope of the access request.*
  + *state=arGmrK8G2lmxQ6bdt9A2N0\_V2In8wB5MOYUjZhUMgNM=: An opaque value used to maintain state between the request and the callback.*
  + *redirect\_uri=http://ssodemoservice01:8080/login/oauth2/code/oidc: The redirect URI where the authorization server will send the user once the access request is granted or denied.*
  + *nonce=-rhKs5Dn8yFjpMsjfKeXRV4t3-AOw2hhXPY4RerfCF0: A string value used to associate a client session with an ID token.*

1. **Logout enpoint**

**Request Details:**

* *URL: /realms/{realm}/protocol/{protocol}/logout*
* *HTTP Version: HTTP/1.1*

**Example endpoint:**

* *GET /realms/jhipster/protocol/openid-connect/logout?id\_token\_hint=eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI4cGJRSnljZHZJc1FrRzJybElNUFlLNFFNeXdYeW02Wk5meGJkQlpwUGp3In0..GBDUkPcnwav9hl317h5\_ay3gNT1FANKejHG8hULnOS6GHU7SWHN-2doUYotic5kg6LBBexAlpdlIxBz4cQOtAD2iaYSMu2gK-CuxvRdm3Z0xKMKKAWBxpU3qKxuKSr7AYD63LaYOrFo4uyWxK1NnTHGglc\_aJxCVjM9dZmih8g0PNSLWSVGLxYpFcjSauHDqpJwCJxqBytSmKBxwdxyc\_mU5WnUbAO4-jg0F8b5A8f\_MNhZP064fkeRgMEYlCyGndEhy5AD5Yk35LXMhRUAapR2CYFUQN3WirHL12EPuTyhYsF2Wzq4UB5jNGt77redSr3lzLXc0TCIq2QgAGYB9CQ&post\_logout\_redirect\_uri=http://ssodemoservice01:8080 HTTP/1.1*

*Host: keycloak.online:9080*

**Explanation:**

URL Parameters:

* *id\_token\_hint: The JWT token previously issued by the authorization server, representing the current session to be logged out.*
* *post\_logout\_redirect\_uri: The URL to which the user is redirected after logout.*

1. **API get access-token**

**Request Details:**

* *Method: POST*
* *URL: /realms/{realm}/protocol/{protocol}/token*
* *HTTP Version: HTTP/1.1*
* *Headers:*
  + *Accept: application/json;charset=UTF-8*
  + *Content-Type: application/x-www-form-urlencoded;charset=UTF-8*
  + *Authorization: Basic <encode of clientKey:clientSecretKey>*
  + *User-Agent: Java/11.0.22*
  + *Host: localhost:9080*
  + *Connection: keep-alive*
  + *Content-Length: 219*
* *Body*
  + *code: The authorization code received from the authorization server.*
  + *client\_id: The client identifier issued to the client during the registration process.*
  + *client\_secret: The client secret issued to the client during the registration process.*
  + *grant\_type: The type of grant being used, which in this case is authorization\_code.*
  + *redirect\_uri: The registered callback-uri when redirect to login-endpoint*

**Example Request:**

* *POST /realms/jhipster/protocol/openid-connect/token HTTP/1.1*
* *Accept: application/json;charset=UTF-8*
* *Content-Type: application/x-www-form-urlencoded;charset=UTF-8*
* *Authorization: Basic d2ViX2FwcF9kZXY6d2ViX2FwcF9kZXY=*
* *User-Agent: Java/11.0.22*
* *Host: localhost:9080*
* *Connection: keep-alive*
* *Content-Length: 219*
* *code=b3b495a9-665b-439c-9c28-7e523e8e35bd.cefe12dc-75ce-4058-881e-9657250e1278.6e8deddb-b4d6-4e2e-b389-b397d3f74fcd&client\_id=ssodemoservice01 &client\_secret=ssodemoservice01&grant\_type=authorization\_code*

**Response:**

*{*

*"access\_token": <jwt from authorize service>,*

*"expires\_in": <expired in minute>,*

*"refresh\_expires\_in": 0,*

*"refresh\_token": <refresh token for request a new access token when the old one is expired>,*

*"token\_type": "Bearer",*

*"id\_token": <id-token, using for logout endpoint>,*

*"not-before-policy": 1718801705,*

*"session\_state": < An opaque value used to maintain state between the request and the callback.>,*

*"scope": "openid email profile offline\_access"*

*}*

1. **API get user-info**

**Request Details:**

* *Method: GET*
* *URL: /realms/{realm}/protocol/{protocol}/*userinfo
* *HTTP Version: HTTP/1.1*
* *Headers:*
  + *Accept: application/json;charset=UTF-8*
  + *Content-Type: application/x-www-form-urlencoded;charset=UTF-8*
  + *Authorization: Bearer <access\_token>*
  + *User-Agent: Java/11.0.22*
  + *Host: localhost:9080*
  + *Connection: keep-alive*

**Example Request:**

* *GET /realms/jhipster/protocol/openid-connect/* *userinfo HTTP/1.1*
* *Accept: application/json;charset=UTF-8*
* *Content-Type: application/x-www-form-urlencoded;charset=UTF-8*
* *Authorization: Bearer eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICI4cGJRSnljZHZJc1FrRzJybElNUFlLNFFNeXdYeW02Wk5meGJkQlpwUGp3In0..TxmmSILWGCnoWDRN5bKohjQTIJEkf8ITNThXbeidTEBHLgvBxVhRlA67lSqEH2bvYYaSLEI5DkhSK7tERjzHJk2aMLTqzQakeDC1jks8DQqFLgIKH5\_3ucEcbNZMrlnRZBDPsx1IHNvUJdKWkKWh9BjAy89VXsCfOFOMzQmsn5olKR5T5tvb6fymOkn3jffX21F48bashNQj62UsH5hylh4CmbZbJ0t3Uu-syQQJs568XK6nq7AJZ8o4fgHEHrPzGRYIAeEJ1o0wPZXqKwbMb8J3onWFv1GXpRiZ8T8ZSUKla\_O2dRCLFHYZCy8bz2\_4eJ2ipaIx9pjpnOQqRGc1xg*
* *User-Agent: Java/11.0.22*
* *Host: localhost:9080*
* *Connection: keep-alive*

**Response:** (the information of loging-in user from authorize-server)

*{*

*"sub": "4c973896-5761-41fc-8217-07c5d13a004b",*

*"email\_verified": true,*

*"roles": [*

*"ROLE\_USER",*

*"offline\_access",*

*"ROLE\_ADMIN",*

*"uma\_authorization"*

*],*

*"name": "Admin Administrator",*

*"preferred\_username": "admin",*

*"given\_name": "Admin",*

*"family\_name": "Administrator",*

*"email": "admin@localhost"*

*}*

1. **Kết luận**
2. **Các điểm còn thiếu trong tài liệu:**

* chưa chỉ rõ các chuẩn được áp dụng trong project như: OAuth 2.0, OpenID Connect, SAML
* chưa có các luồng làm việc trong trường hợp người dùng đăng nhập SSO trên mobile
* chưa có so sánh với các framwork xác thực tập trung khác
* ứng dụng service con đang sử dụng cơ chế mặc định theo stateful (sử dụng session-id)

1. **Nội dung sắp tới (nếu được dự án được phê duyệt)**

* Hỗ trợ xác thực theo cơ chế stateless
* Chi tiết hoá các cơ chế mà keycloak hỗ trợ