**TÀI LIỆU HƯỚNG DẪN TÍCH HỢP BASEAPP VÀO SSO**

**Phiên bản: 2.4**

**LỊCH SỬ THAY ĐỔI**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 31/12/2021 | 1.0 | Phát hành tài liệu |
| 05/01/2022 | 1.1 | Cập nhật tài liệu |
| 07/01/2022 | 1.2 | Cập nhật tài liệu |
| 10/01/2022 | 1.3 | Cập nhật tài liệu |
| 17/01/2022 | 1.4 | Cập nhật tài liệu |
| 18/01/2022 | 1.5 | Cập nhật tài liệu |
| 20/01/2022 | 1.6 | Cập nhật tài liệu |
| 09/03/2022 | 1.7 | Cập nhật tài liệu |
| 15/03/2022 | 1.8 | Cập nhật tài liệu |
| 06/06/2022 | 1.9 | Cập nhật tài liệu |
| 22/02/2023 | 2.0 | Cập nhật tài liệu |
| 04/04/2023 | 2.1 | Cập nhật tài liệu |
| 10/04/2023 | 2.2 | Cập nhật tài liệu |
| 29/11/2023 | 2.3 | Cập nhật tài liệu |
| 17/05/2024 | 2.4 | Cập nhật tài liệu |

**MỤC LỤC**

[**I.** Hướng dẫn tích hợp cho .NET Framework 1](#_Toc91852712)

[**1.** Cấu hình đường đẫn lấy key giải mã token trong file web.config 1](#_Toc91852713)

[**2.** Tạo class lưu key giải mã token: key giải mã rất ít khi cập nhật, nên chỉ cần lấy 1 lần khi start service 1](#_Toc91852714)

[**3.** Lấy key giải mã token khi start service (trong Application\_Start()) 1](#_Toc91852715)

[**4.** Cập nhật Validate Token: 2](#_Toc91852716)

[**II.** **Hướng dẫn tích hợp cho Python** 4](#_Toc91852717)

[**1.** Cài đặt thư viện 5](#_Toc91852718)

[**2.** Cập nhật hàm validate token 5](#_Toc91852719)

[**III.** **Hướng dẫn tích hợp cho React** 5](#_Toc91852720)

[**1.** Cài đặt thư viện Oidc Client 5](#_Toc91852721)

[**2.** Cấu hình 5](#_Toc91852722)

[**3.** Thay đổi và thêm mới 1 số file 5](#_Toc91852723)

Lưu ý tích hợp: Các hệ thống tích hợp vào cần cung cấp cho đội phát triển SSO thông tin các tên miền domain các ứng dụng front-end, backend hoặc IP ( với localhost hoặc môi trường lab) để phía SSO cho phép truy cập tích hợp bao gồm các thông tin sau

* Link (IP) ứng dụng callback phía front-end sau login, logout thành công sẽ chuyển hướng đến màn hình giao diện nào
* Link (IP) ứng dụng back-end

# Hướng dẫn tích hợp cho .NET Framework

## Cấu hình đường đẫn lấy key giải mã token trong file web.config

<appSettings>

    <add key="AUTH\_DOMAIN" value="https://lab.connect247.vn/.well-known/openid-configuration" />

</appSettings>

## Tạo class lưu key giải mã token: key giải mã rất ít khi cập nhật, nên chỉ cần lấy 1 lần khi start service

public static class ConfigAuth

{

    public static OpenIdConnectConfiguration openIdConfig { get; set; }

}

## Lấy key giải mã token khi start service (trong Application\_Start())

* Using thư viện

using Microsoft.IdentityModel.Protocols;

using Microsoft.IdentityModel.Protocols.OpenIdConnect;

try

{

    string auth0Domain = WebConfigurationManager.AppSettings["AUTH\_DOMAIN"];

    IConfigurationManager<OpenIdConnectConfiguration> configurationManager =

        new ConfigurationManager<OpenIdConnectConfiguration>(auth0Domain, new OpenIdConnectConfigurationRetriever());

    ConfigAuth.openIdConfig = configurationManager.GetConfigurationAsync(CancellationToken.None).Result;

}

catch (Exception ex)

{

    // Log error

}

## Cập nhật Validate Token:

* Hàm GetPrincipalFromTokenAsync()

private static ClaimsPrincipal GetPrincipalFromTokenAsync(string token)

{

    try

    {

        var tokenValidationParameters = new TokenValidationParameters

        {

            ValidateAudience = false,

            ValidateIssuer = false,

            ValidateIssuerSigningKey = true,

            ValidateLifetime = false,

            IssuerSigningKeys = ConfigAuth.openIdConfig.SigningKeys //Key giải mã token

        };

        Microsoft.IdentityModel.Tokens.SecurityToken securityToken;

        var tokenHandler = new JwtSecurityTokenHandler();

        var tokenS = tokenHandler.ReadToken(token) as JwtSecurityToken;

        var exp = tokenS.Claims.First(claim => claim.Type == "exp").Value;

        long currentDate = ConvertToTimestamp(DateTime.UtcNow);

        if (long.Parse(exp) >= currentDate)

        {

            var principal = tokenHandler.ValidateToken(token, tokenValidationParameters, out securityToken);

            return principal;

        }

        return null;

    }

    catch (Exception ex)

    {

        return null;

    }

}

* Hàm ValidateToken(string token): Cập nhật cấu trúc get Clams
  + Các giá trị Clams sẽ được lưu theo định dạng json, với mỗi dịch vụ sẽ có key để get clams tương ứng

public class ClamModel

{

    public string name { get; set; }

    public string tenant\_id { get; set; }

    public string is\_administrator { get; set; }

    public string is\_rootuser { get; set; }

public string roles { get; set; }

}

public static bool ValidateToken(string token)

{

    try

    {

        \_logger = CommonFunc.CreateInstanceDJ<ILogger>();

        ClaimsPrincipal principal = GetPrincipalFromTokenAsync(token);

        if (principal == null) return false;

        ClaimsIdentity identity = null;

        identity = (ClaimsIdentity)principal.Identity;

        var clamModel = new ClamModel();

        var strValue = principal.Claims.FirstOrDefault(claim => claim.Type == "ic\_claims")?.Value??"";

        clamModel = JsonConvert.DeserializeObject<ClamModel>(strValue);

        SessionStore.Set(Constants.KEY\_SESSION\_TOKEN, token);

        SessionStore.Set(Constants.KEY\_SESSION\_EMAIL, clamModel?.name);

        SessionStore.Set(Constants.KEY\_SESSION\_TENANT\_ID, clamModel?.tenant\_id);

        SessionStore.Set(Constants.KEY\_SESSION\_IS\_ADMIN, clamModel?.is\_administrator);

        SessionStore.Set(Constants.KEY\_SESSION\_IS\_ROOT, clamModel?.is\_rootuser);

        return true;

    }

    catch (Exception ex)

    {

        \_logger.LogError(ex);

        return false;

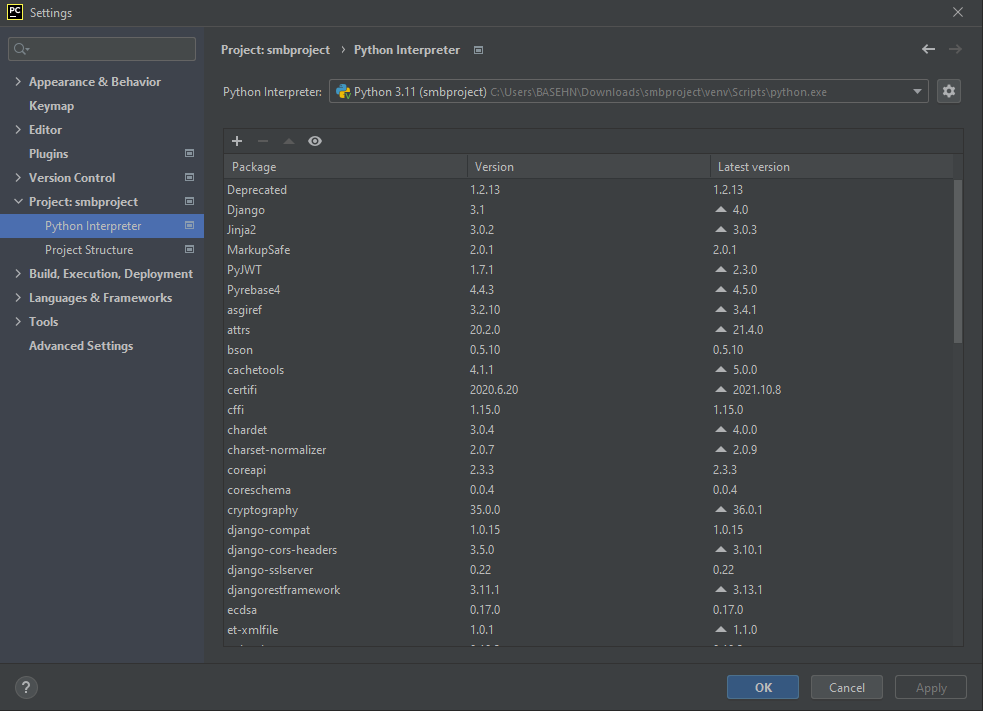
    }

}

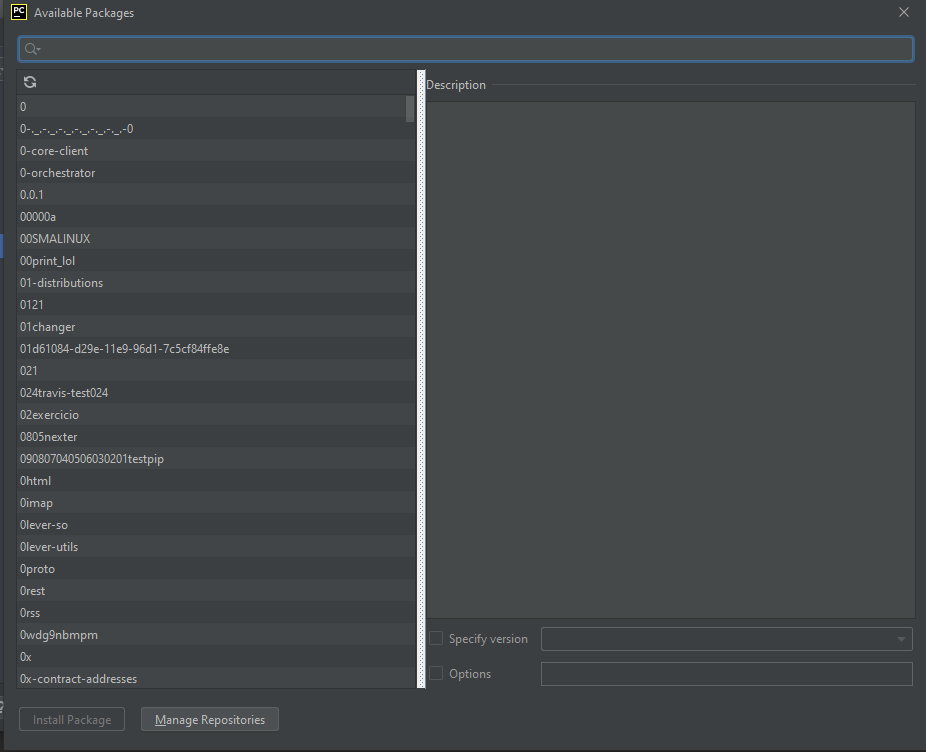
# **Hướng dẫn tích hợp cho Python**

## Cài đặt thư viện

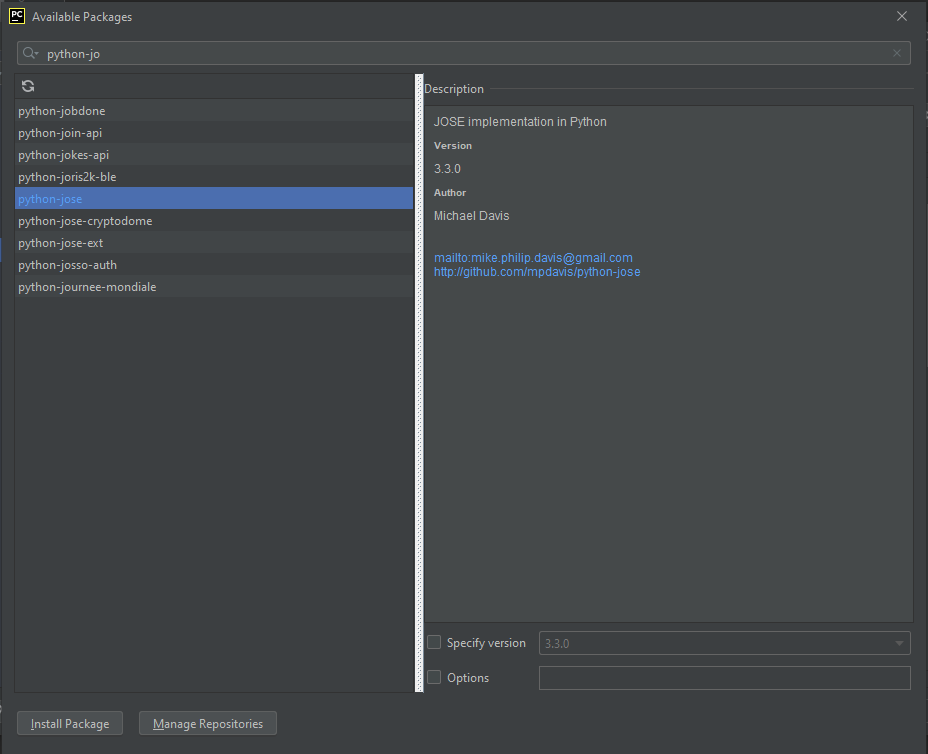
* Sử dụng JWT bằng Python\_jose.



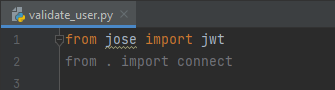
* Nhấn vào thêm interpreter



* Tìm kiếm gói python-jose, nhấn cài đặt.



* Sau khi cài đặt hoàn tất. Mở file validate\_user.py. Thực hiện thay đổi thư viện sử dụng JWT như bên dưới.



## Cập nhật hàm validate token

* Trong file validate\_user.py. Thực hiện thêm hàm như dưới

AUTH0\_DOMAIN = 'lab.connect247.vn'  
API\_AUDIENCE = 'sso-center'  
ALGORITHMS = ["RS256"]  
def get\_salt():  
 jsonurl = urlopen("https://"+AUTH0\_DOMAIN+"/.well-known/openid-configuration/jwks")  
 jwks = json.loads(jsonurl.read())  
 rsa\_key = {}  
 for key in jwks["keys"]:  
 rsa\_key = {  
 "kty": key["kty"],  
 "kid": key["kid"],  
 "use": key["use"],  
 "n": key["n"],  
 "e": key["e"],  
 "alg": key["alg"]  
 }  
 if(rsa\_key is not None):  
 return rsa\_key  
 else:  
 return "BASEBS"

* Cùng file trên. Thực hiện sửa hàm validate\_token và hàm validate\_only\_token

def validate\_token(request):  
 try:  
 \_ = request.headers['Authorization']  
 token = \_.replace('Bearer ', '')  
 decode\_token = jwt.decode(  
 token,  
 get\_salt(),  
 audience="https://lab.connect247.vn/resources",  
 algorithms=ALGORITHMS,  
 issuer="https://"+AUTH0\_DOMAIN  
 )  
 lcm\_data = json.loads(decode\_token["lcm\_claims"])  
  
 tenant\_id = lcm\_data['tenant\_id']  
 email\_user = decode\_token['Email']  
 user\_id = lcm\_data["user\_name"]  
 is\_admin = lcm\_data["is\_administrator"]

return {  
 "status": "success",  
 "tenants": tenant\_id,  
 "email": email\_user,  
 "user\_id": user\_id,  
 "is\_admin": is\_admin  
 }  
 except Exception as e:  
 return {"status": "error"}

def validate\_only\_token(\_token):  
 try:  
 token = \_token.replace('Bearer ', '')  
 decode\_token = jwt.decode(  
 token,  
 get\_salt(),  
 audience="https://lab.connect247.vn/resources",  
 algorithms=ALGORITHMS,  
 issuer="https://" + AUTH0\_DOMAIN  
 )  
 lcm\_data = json.loads(decode\_token["lcm\_claims"])  
  
 tenant\_id = lcm\_data['tenant\_id']  
 email\_user = decode\_token['Email']  
 user\_id = lcm\_data["user\_name"]  
 is\_admin = lcm\_data["is\_administrator"]  
 return {  
 "status": "success",  
 "tenants": tenant\_id,  
 "email": email\_user,  
 "user\_id": user\_id,  
 "is\_admin": is\_admin  
 }  
 except:  
 return {"status": "error"}

* Thực hiện gọi API bất kỳ để test nội dung vừa sửa dổi. Sử dụng chức năng đăng nhập của SSO

# **Hướng dẫn tích hợp cho React**

## Cài đặt thư viện Oidc Client

* Tại thư mục project chạy lệnh:
  + NPM: **npm install oidc-client@1.10.1 --save**
  + Yarn: **yarn add** [**oidc-client@1.10.1**](mailto:oidc-client@1.10.1)

## **(UPDATE 2.4)** Cấu hình

* Tại file env thêm các thuộc tính sau:

REACT\_APP\_STS\_AUTH = <https://lab.connect247.vn/>

* + - Link server SSO (Không cần thay đổi)

REACT\_APP\_CLIENT\_ID = sso\_center

* + - Cấu hình key phía server (Không cần thay đổi)

REACT\_APP\_CLIENT\_ROOT= link client

* + - Domain của client (cần cung cấp lại cho SSO để allow)

REACT\_APP\_CLIENT\_SCOPE = **openid profile offline\_access**

* + - openid profile offline\_access : 3 tham số này dữ nguyên

REACT\_APP\_CLIENT\_LOGOUT\_URL=https://domain-frontend/signin

* + - Link trang đăng nhập của từng ứng dụng
* Tại file constant (src/constants/index.js) thêm các đoạn code:

const stsAuthority = process.env.REACT\_APP\_STS\_AUTH

const clientId = process.env.REACT\_APP\_CLIENT\_ID

const clientRoot = process.env.REACT\_APP\_CLIENT\_ROOT

const clientScope = process.env.REACT\_APP\_CLIENT\_SCOPE

const apiRoot = process.env.REACT\_APP\_API\_ROOT

const clientLogoutURL = process.env.REACT\_APP\_CLIENT\_LOGOUT\_URL

export const oidcClient = {

    stsAuthority: stsAuthority,

    clientId: clientId,

    clientRoot: clientRoot,

    clientScope: clientScope,

    clientLogoutURL: clientLogoutURL

}

* **(UPDATE 2.4) Thêm folder services tại src, tạo file AuthService.js**

import { Log, UserManager } from 'oidc-client'

import { oidcClient } from '../constants'

const AuthService = (function () {

    const settings = {

        authority: oidcClient.stsAuthority,

        client\_id: oidcClient.clientId,

        redirect\_uri: `${oidcClient.clientRoot}signin-callback.html`,

        silent\_redirect\_uri: `${oidcClient.clientRoot}silent-renew.html`,

        post\_logout\_redirect\_uri: `${oidcClient.clientLogoutURL}`,

        response\_type: 'code',

        scope: oidcClient.clientScope,

        loadUserInfo: true,

        automaticSilentRenew: true,

        filterProtocolClaims: true,

        monitorSession: true,

        accessTokenExpiringNotificationTime: 180

    }

    var userManager = new UserManager(settings)

    Log.logger = console

    Log.level = Log.INFO

    function getUser() {

//v2.2 update

return new Promise((resolve, reject) => {

            userManager

                .getUser()

                .then((user) => {

                    if (!user) {

                        userManager

                            .signinSilent()

                            .then(() => {

                                userManager.getUser().then((user) => {

                                    resolve(user)

                                })

                            })

                            .catch(() => {

                                userManager.signinRedirect()

                            })

                    } else resolve(user)

                })

                .catch((error) => {

                    reject(error)

                })

        })

    }

    function login() {

        return userManager.signinRedirect()

    }

    function renewToken() {

        return userManager.signinSilent()

    }

    function logout() {

let paramLogout = {

id\_token\_hint: localStorage.getItem('setting\_idToken')

}

       return userManager.signoutRedirect(paramLogout)

    }

    function checkSession() {

        return userManager.querySessionStatus()

    }

    function eventSignOut(event) {

        return userManager.events.addUserSignedOut(() => {

            console.log('LOGOUT SSO')

            event()

        })

    }

    function eventAddAccessTokenExpiring() {

        return userManager.events.addAccessTokenExpiring(() => {

            console.log('AT expiring event success')

        })

    }

    function eventAddAccessTokenExpired() {

        return userManager.events.addAccessTokenExpired(() => {

            console.log('AT expired event success')

            setTimeout(() => {

                userManager.signinSilent()

            }, 1000)

        })

    }

    function eventAddSilentRenewError() {

        return userManager.events.addSilentRenewError(() => {

            setTimeout(() => {

                userManager.signinSilent()

            }, 1000)

        })

    }

    function eventAddUserLoaded() {

        return userManager.events.addUserLoaded(() => {

            userManager.getUser().then((user) => {

//v2.4 update

**window.localStorage.setItem(**[KEY\_ACCESS\_TOKEN]**, user.access\_token)**

window.localStorage.setItem('setting\_idToken', user.id\_token)

**window.localStorage.setItem('setting\_oidc', JSON.stringify(user))**

            })

        })

    }

    return {

        getUser: getUser,

        login: login,

        renewToken: renewToken,

        logout: logout,

        checkSession: checkSession,

        eventSignOut: eventSignOut,

        eventAddAccessTokenExpiring: eventAddAccessTokenExpiring,

        eventAddAccessTokenExpired: eventAddAccessTokenExpired,

        eventAddUserLoaded: eventAddUserLoaded,

        eventAddSilentRenewError: eventAddSilentRenewError

    }

})()

export default AuthService

* **Tại folder public, thêm file signin-callback.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <title></title>

</head>

<body>

    <noscript>

        You need to enable JavaScript to run this app.

    </noscript>

    <script src="oidc-client.min.js"></script>

    <script>

        new Oidc.UserManager({ response\_mode: "query" }).signinRedirectCallback().then(function (user) {

            window.localStorage.setItem('setting\_accessToken', user.access\_token)

window.localStorage.setItem('setting\_username', user?.profile?.name)

window.localStorage.setItem('setting\_idToken', user.id\_token)

window.localStorage.setItem('setting\_oidc', JSON.stringify(user))

            window.location = "path screen cần điều hướng đến sau khi login";

        }).catch(function (e) {

            console.error(e);

        });

    </script>

</body>

</html>

* Tại folder public, thêm file silent-renew.html:

<script src="oidc-client.min.js"></script>

<script>

    var mgr = new Oidc.UserManager();

    mgr.signinSilentCallback().then((user)=>{

    }).catch(function (error) {

        console.error(error);

    });

</script>

* Tại folder public, thêm file oidc-client.min.js

url tải file: https://github.com/IdentityModel/oidc-client-js/blob/dev/dist/oidc-client.min.js



## Thay đổi và thêm mới 1 số file

* Thêm folder Login tại src/containers, tạo file index.js: (Login)

import React, { useEffect } from 'react'

import AuthService from '../../services/AuthService'

const Login = (props) => {

    useEffect(() => {

        AuthService.login()

    }, [])

    return <div />

}

export default Login

* Tại file router.js (src/routers), thay đổi router của phần login

export const routersNotAuth = [

    {

        exact: true,

        path: '/signin',

        component: 'containers/Login'

    }

]

* Tại folder src/containers/Layout/Header, chỉnh sửa file index.js (Logout)

import AuthService from '../../../services/AuthService'

import { LocalStorage } from '../../../utils/localStorage'

……

const localStorageService = LocalStorage.getService()

    const handleLogout = () => {

        localStorageService.clearToken()

        AuthService.logout()

localStorage.removeItem('setting\_idToken')

    }

* **(UPDATE 2.4) Tại folder src/containers/Layout, chỉnh sửa file index.js (Event logout ở hệ thống khác, check user đã đăng nhập hay chưa)**

import AuthService from '../../services/AuthService'

import { oidcClient } from '../../constants'

……

    useEffect(() => {

//Check user đã đăng nhập hay chưa

        AuthService.checkSession().then(() => {

**AuthService.getUser().then((res)=>{})**

}).catch(() =>{

        })

//Sự kiện đã logout khỏi hệ thống

        AuthService().eventSignOut(() => {

        })

//Sự kiện refresh token

AuthService.eventAddAccessTokenExpiring(() => {

        })

//Sự kiện token đã hết hạn

        AuthService.eventAddAccessTokenExpired(() => {

        })

//Sự kiện get thông tin user sau khi refresh token + lưu token mới

        AuthService.eventAddUserLoaded(() => {

        })

//Sự kiện refresh token thất bại

        AuthService.eventAddSilentRenewError(() => {

        })

//Cập nhật lại thông tin user vào sessionStorage

        if (window.addEventListener)

            window.addEventListener(

                'storage',

                () => {

                    sessionStorage.setItem(

                        `oidc.user:${oidcClient.stsAuthority}:${oidcClient.clientId}`,

                        localStorage.getItem('setting\_oidc')

                    )

                },

                false

            )

    }, [dispatch])

* Tại folder src/containers/Layout, chỉnh sửa file index.js (Thêm iframe vào trong view)

<iframe src="{url}baseapp?userName={userName}&userID={userID}&baseAppName={baseAppName}" title="sso" style={{ display: 'none' }}/>

Với các tham số

url: REACT\_APP\_STS\_AUTH trong file env

userName: Tên đăng nhập

userID: ID user

baseAppName: Theo từng dịch vụ (VD: IC, LCM, CRM, Connect247 )

* Tại folder src/utils, chỉnh sửa file request.js (dung token lưu ở localstorage – key: setting\_accessToken trước để để truyền header API)

……

    const request = axios.create({

        baseURL: urlLink,

        headers: {

            'content-type': 'application/json'

        },

        paramsSerializer: (params) => queryString.stringify(params)

    })

    request.defaults.withCredentials = true

sendToken &&

        request.interceptors.request.use(

            (config) => {

                const token = localStorageService.getAccessToken()

                if (token) {

                    config.headers['Authorization'] = 'Bearer ' + token

                }

                return config

            },

            (error) => {

                Promise.reject(error)

            }

        )

* **(UPDATE 2.4) Nút Redirect đến các bass app**

Phía base app tự thiết kế nút và nhúng iframe này

<iframe src="{url}sso-fe/list-base-app" />

Với các tham số

url: REACT\_APP\_STS\_AUTH trong file env