ĐẠI HỌC QUỐC GIA HÀ NỘI
Trường Đại học Công nghệ
Khoa Công nghệ thông tin
🕮 🕮 🕮

BÁO CÁO THỰC TẬP

Đề tài

Nâng cấp back-end của hệ thống Sổ thu chi MISA

Giảng viên hướng dẫn: TS. Lê Đình Thanh
Cán bộ: Nguyễn Văn Mạnh
Công ty cổ phần MISA

Sinh viên: Nguyễn Thành Trung
MSV: 17021087 – Lớp: QH-2017-I-CQ-CK

Hà Nội, tháng 10 năm 2020

Lời cám ơn

Là một học sinh của khoa Công nghệ thông tin của trường ĐH Công Nghệ. Trong suốt 3 năm học vừa qua em đã biết thêm nhiều kiến thức nền tảng và chuyên sâu về lĩnh vực công nghệ thông tin để nâng cao tay nghề nhằm phát triển khả năng sửa chữa, xây dựng, cài đặt, bảo trì các phần cứng của máy tính cũng như nghiên cứu và phát triển các ứng dụng phần mềm. Ngoài ra cũng được trang bị kiến thức về an toàn và bảo mật thông tin mạng, một trong những lĩnh vực quan trọng được quan tâm hàng đầu trên thế giới hiện nay. Em đã rất hứng thú học tập và rèn luyện trong suốt quãng thời gian qua.

Em xin được gửi lời cảm ơn đến các thầy cô trong nhà trường, các thầy cô bộ môn của Khoa đã giảng dạy rất tâm huyết, truyền dạy cho chúng em nhiều kiến thức cũng như tình cảm với nghề nghiệp của mình. Với đề tài "Nâng cấp back-end của hệ thống Sổ thu chi MISA" cùng với sự giúp sức nhiệt tình của TS Lê Đình Thanh đã hướng dẫn giúp đỡ em hoàn thiện đợt thực tập này. Em muốn gửi lời cảm ơn đặc biệt nhất, sâu sắc nhất, thân thương nhất đến thầy.

Tiếp theo em xin cảm ơn công ty cổ phần MISA đã tài trợ và giúp đỡ em trong suốt quá trình em thực hiện và hoàn thiện đề tài này.

Trong suốt thời gian thực tập vừa qua em còn nhiều thiếu sót, em mong sẽ được thầy cô cũng như quý công ty góp ý để em có thể sửa đổi và hoàn thiện đề tài này nhanh nhất để có thể ứng dụng vào thực tế.

Em xin chân thành cảm ơn!

Hà Nội, tháng 10 năm 2020
Sinh viên thực tập

Nguyễn Thành Trung

Mục Lục

Phần I. Giới thiệu chung 1
1. Giới thiệu đơn vị thực tập: Công ty cổ phần MISA 1
2. Giới thiệu công việc 1
3. Giới thiệu bài toán 1
II. Yêu cầu bài toán 2
III. Tóm tắt lý thuyết, giải pháp, thuật toán 2
1. Lý thuyết, giải pháp, thuật toán liên quan 2
a. Asp.net Core Identity và IdentityServer 4 2
b. API Gateway 3
c. JWT 4
2. Hướng giải quyết 4
a. Xác định yêu cầu nghiệp vụ 4
b. Tìm hiểu các công nghệ mới 5
3. Liên hệ và so sánh 5
IV. Mô tả cài đặt phần mềm 6
1. Cài đặt và cấu hình Kong và Konga 6
a. Cài đặt Kong, Konga 6
b. Sinh bộ mã khoá công khai và khoá cá nhân 8
2. Cài đặt và cấu hình IdentityServer4 và Identity 8
a. Cấu hình IdentityServer4 8
b. Cấu hình Asp.net Core Identity 10
3. Cài đặt WebApi phục vụ cho việc đăng nhập, đăng ký 13
a. Đăng nhập với MISAID 13
b. Đăng ký với tài khoản mạng xã hội 13
c. Quản lý thông tin người dùng 14
d. Cơ chế lấy lại access_token 14
e. Xác thực đăng nhập hai lớp 15
V. Kết quả đạt được 16
VI. Hướng phát triển 18
Đánh giá của giảng viên hướng dẫn 19

Phần I. Giới thiệu chung

1. Giới thiệu đơn vị thực tập: Công ty cổ phần MISA

Công ty cổ phần MISA là công ty cung cấp các phần mềm quản lý cho các cơ quan, nhà nước, doanh nghiệp. MISA chuyên ở lĩnh vực quản lý công (như phần mềm kế toán, quản lý tài sản, quản lý trường học, quản lý hộ tịch, …) và quản trị doanh nghiệp (như phần mềm quản trị kế toán tài chính, nhân sự, bán hàng, v.v…). Hiện tại MISA có 5 văn phòng đại diện ở Hà Nội, Đà Nẵng, Buôn Mê Thuật, Hồ Chí Minh và Cần Thơ.

Bằng nỗ lực sáng tạo trong khoa học, công nghệ và đổi mới trong quản trị, MISA mong muốn trở thành công ty có nền tảng, phần mềm và dịch vụ được sử dụng phổ biến nhất trong nước và quốc tế. Hành trình 25 năm phát triển, xả thân, sáng tạo, vì sứ mệnh phụng sự xã hội. Mang trong mình sứ mệnh phát triển các nền tảng, phần mềm và dịch vụ công nghệ thông tin để thay đổi ngành kinh tế và giúp khách hàng thực hiện công việc theo phương thức mới, năng suất và hiệu quả hơn nhằm thúc đẩy sự phát triển của đất nước và các quốc gia trên thế giới.

2. Giới thiệu công việc

Sổ thu chi MISA là ứng dụng di động giúp các cá nhân và tổ chức ghi chép lại việc chi tiêu tài chính. Còn được biết đến với tên MISA MoneyKeeper.

Trong thời gian thực tập, em được phân công vào nhóm back-end của phần mềm Sổ thu chi MISA. Đảm nhận công việc nâng cấp back-end của hệ thống, sử dụng các công nghệ mới hiện năng nhằm cải thiện hiệu năng của hệ thống, bước đầu đặt nền tảng cho ứng dụng web được phát triển sau này.

3. Giới thiệu bài toán

Phần service của Sổ thu chi MISA hiện đang được xây dựng bằng Asp.net (.NET Framework) và WCF, VB. Hiện tại, các công nghệ đã cũ, hiệu năng của hệ thống không ổn định, không đáp ứng được nhu cầu dữ liệu ngày càng lớn của

người dùng. Vì thế, dự án cần nâng cấp hệ thống back-end lên sử dụng các công nghệ mới nhất hiện nay để tối ưu hoá hiệu năng.

II. Yêu cầu bài toán

Nâng cấp hệ thống back-end của sổ thu chi MISA, sử dụng các công nghệ mới hiện nay như: Api Gateway, OpenAPI Spec, Identity (Asp.net core 3.0).

- Database: Chuyển từ SqlServer sang Mariadb

- Ngôn ngữ: VB, WCF sang C#

- Framework: .NET sang .NET Core

- Áp dụng được các kỹ thuật công nghệ mới như viết API theo chuẩn OpenAPI; sử dụng Kong để định tuyến API; áp dụng Identity (hỗ trợ trên .NET core) để Authen việc đăng nhập đăng ký.

Nhiệm vụ của em được phân công là xây dựng microservice đăng nhập, đăng ký tài khoản. Sử dụng Identity và IdentityServer4 để quản lý việc đăng nhập, đăng ký của hệ thống. Khởi tạo cơ chế kiểm tra Authen cho các API bằng cách sinh AccessToken và cơ chế kiểm chứng token.

III. Tóm tắt lý thuyết, giải pháp, thuật toán

1. Lý thuyết, giải pháp, thuật toán liên quan

a. Asp.net Core Identity và IdentityServer 4

ASP.NET Core Identity là một thư viện quản lý việc chứng thực và cấp quyền được dùng cho mọi loại project từ MVC, WebForms cho đến WebAPI. Nó đáp ứng các nhu cầu chính như sau:

- Dễ dàng tùy chỉnh profile của user

- Lưu trữ thông tin user trong cơ sở dữ liệu sử dụng EF Code First.

- Hỗ trợ unit test

- Giới hạn quyền truy cập theo quyền

- Cung cấp cơ chế làm việc với claim

IdentityServer4 là một framework hỗ trợ OpenID Connect và OAuth 2.0 trên ASP.NET Core. Nó cũng là một gói thư viện trên Nuget được dùng trong ASP.NET Core như một middleware cho phép đăng nhập/đăng xuất, cấp token, chứng thực và các giáo thức chuẩn khác. Kiến trúc tổng quan của nó bao gồm:

- User: là con người, là bạn hoặc tôi đang sử dụng một client.

- Client: là một phần mềm ứng dụng kiểu như trình duyệt web, mobile app hay bất cứ cái gì đang cần gọi một API resource.

- Resources: là các API bạn cần bảo vệ bởi IdentityServer4

- Access Token: nó là token được sử dụng để truy cập vào API Resource

- Refresh Token: mỗi một token đều có một thời gian hết hạn. Refresh token là việc lấy lại token mới mà không cần tương tác của người dùng. Client nên cho phép làm điều này bằng cách setup AllowOfflineAccess giá trị là true ở phần cài đặt client trong IdentityServer4.

- Grant Type: nó là loại tương tác giữa client và IdentityServer. Dựa trên client của bạn, có thể chọn loại grant type phù hợp.

b. API Gateway

API Gateway có thể coi là một cổng trung gian, nó là cổng vào duy nhất tới hệ thống microservices của chúng ta, api gateway sẽ nhận các requests từ phía client, chỉnh sửa, xác thực và điều hướng chúng đến các API cụ thể trên các services phía sau. Với API Gateway, chúng ta có thể:

- Che dấu được cấu trúc của hệ thống microservices với bên ngoài

- Phần code phía frontend sẽ gọn gàng hơn

- Dễ dàng theo dõi và quản lý traffic.

- Requests caching và cân bằng tải.

- Thêm một lớp bảo mật nữa cho hệ thống.

- Thay thế authentication services

Tuy nhiên, cũng có một số nhược điểm cần lưu ý:

- Tăng thời gian response

- Thêm tác nhân gây lỗi

- Có thể gây nghẽn cổ chai

- Tốn thêm tiền

c. JWT

Json Web Token (JWT) là một phương tiện đại diện cho các yêu cầu chuyển giao giữa hai bên Client – Server, các thông tin trong chuỗi JWT được định dạng bằng JSON. Trong đó chuỗi Token phải có 3 phần là header, phần payload và phần signature được ngăn bằng dấu “.”

Vậy theo lý thuyết trên thì mình sẽ có một chuỗi Token như sau:

```
header.payload.signature
```

Vậy khi nào cần dùng JWT? Authentication: Đây là trường hợp phổ biến nhất thường sử dụng JWT. Khi người dùng đã đăng nhập vào hệ thống thì những request tiếp theo từ phía người dùng sẽ chứa thêm mã JWT. Điều này cho phép người dùng được cấp quyền truy cập vào các url, service, và resource mà mã Token đó cho phép. Phương pháp này không bị ảnh hưởng bởi Cross-Origin Resource Sharing (CORS) do nó không sử dụng cookie.

2. Hướng giải quyết

a. Xác định yêu cầu nghiệp vụ

Thông qua tìm hiểu và phân tích nghiệp vụ, yêu cầu được đặt ra bao gồm đăng nhập, đăng ký bằng 3 phương pháp:

- Thông qua MISAID: MisaID là một hệ thống của MISA quản lý tài khoản chung của MISA. Với tài khoản MISAID, người dùng có thể đăng nhập mọi ứng dụng của MISA. MISAID đã hỗ trợ đăng nhập, đăng ký, xác thực hai lớp, … qua API.

- Thông qua Facebook, Google: từ các tài khoản mạng xã hội có thể đăng nhập được với Sổ thu chi. Hiện tại, MISAID chưa hỗ trợ đăng nhập, đăng ký mạng xã hội qua API nên phải tự xử lý.

- Thông qua AppleId: tương tự như Facebook, Google. Tài khoản AppleId cũng phải có cơ chế xử lý riêng.

Các tài khoản mạng xã hội khi đăng nhập, đăng ký cần được quản lý token để có thể bảo vệ các API sau này. Vì thế phải có cơ chế quản lý access_token và refresh_token.

b. Tìm hiểu các công nghệ mới

Bắt đầu một công nghệ mới, cần phải tìm và hiểu rõ về nó, hiểu được cách thức hoạt động, nguyên lý, ưu nhược điểm của nó. Sau đó, bắt đầu xây dựng các ứng dụng thử nghiệm để biết cách cài đặt, cấu hình, tổ chức thực thi. Sau khi các ứng dụng thử nghiệm có thể hoạt động và hoạt động tốt (khả thi) thì mới tìm cách áp dụng nó vào bài toán của mình. Vì bài toán là nâng cấp, nên cần giữ lại sự ổn định của dữ liệu, kiểu dữ liệu, vì thế, khi áp dụng các công nghệ mới phải thực sự coi trọng đến yếu tố dữ liệu.

Sau khi có giải pháp hợp lý, mới bắt đầu tiến hành vào thi công. Vừa làm vừa kiểm tra, vừa làm vừa tìm hiểu. Và sau đó là tìm cách cải tiến hệ thống.

3. Liên hệ và so sánh

So với hệ thống cũ, service đăng nhập mới sẽ có các điểm mới:

- Các tài khoản đăng nhập bằng Facebook, Google, AppleId sẽ được quản lý bằng Identity và IdentityServer4 với cơ chế sinh token JWT thay vì quản lý trên Cookie (do Membership trên Asp.net chưa hỗ trợ đăng nhập với các tài khoản mạng xã hội)

- Sau khi đăng nhập, các tài khoản mạng xã hội sẽ được liên kết với tài khoản MISAID, nơi quản lý tập chung các tài khoản khách hàng của MISA.

- Tạo ra các Client với các cấu hình phân quyền truy cập API. Do đó các API ở sau sẽ được bảo vệ tốt hơn.

- Service này được đặt trước API Gateway (ở đây, được sử dụng Kong và Konga), từ đây, cấu hình cơ chế xác thực token bằng các thuật toán mã hoá bất đối xứng (sử dụng một cặp khoá công khai và khoá cá nhân để mã hoá và giải mã).

IV. Mô tả cài đặt phần mềm

1. Cài đặt và cấu hình Kong và Konga

Các ứng dụng đều được cài đặt trên Docker, hệ điều hành Windows 10. Docker là một nền tảng để cung cấp cách để building, deploying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers (trên nền tảng ảo hóa). Ban đầu viết bằng Python, hiện tại đã chuyển sang Golang. Vì thế, phần mềm được dùng là Docker Desktop.

a. Cài đặt Kong, Konga

Cài đặt network:

```
docker network create kong-net
```

Cài đặt kong-database:

```
docker run -d --name kong-database --network kong-net -p 5432:5432 -e "POSTGRES_USER=kong" -e "POSTGRES_PASSWORD=kong" -e "POSTGRES_DB=kong" postgres:9.6
```

Cài đặt kong mirgate:

```
docker run --rm --network kong-net -e "KONG_DATABASE=postgres" -e "KONG_PG_HOST=kong-database" -e "KONG_PG_USER=kong" -e "KONG_PG_PASSWORD=kong" kong:latest kong migrations bootstrap
```

Cài đặt kong:

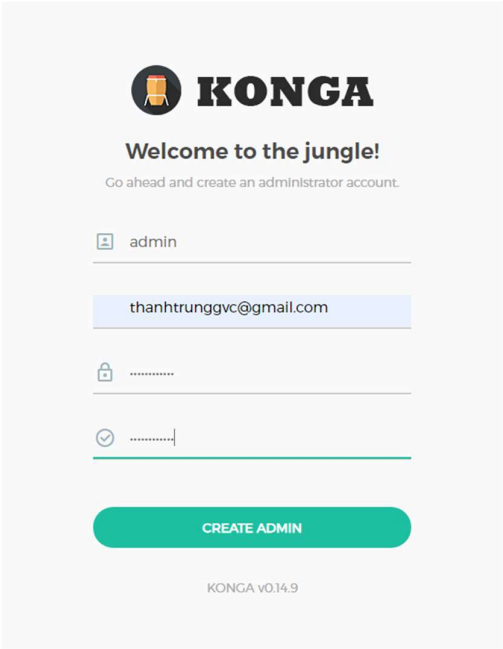
```
docker run -d --name kong --network kong-net -e "KONG_DATABASE=postgres" -e "KONG_PG_HOST=kong-database" -e "KONG_PG_USER=kong" -e "KONG_PG_PASSWORD=kong" -e "KONG_PROXY_ACCESS_LOG=/dev/stdout" -e "KONG_ADMIN_ACCESS_LOG=/dev/stdout" -e "KONG_PROXY_ERROR_LOG=/dev/stderr" -e "KONG_ADMIN_ERROR_LOG=/dev/stderr" -e "KONG_ADMIN_LISTEN=0.0.0.0:8001" -e "KONG_ADMIN_LISTEN_SSL=0.0.0.0:8444" -p 8000:8000 -p 8443:8443 -p 8001:8001 -p 8444:8444 –-link kong-database:kong-database kong:latest
```

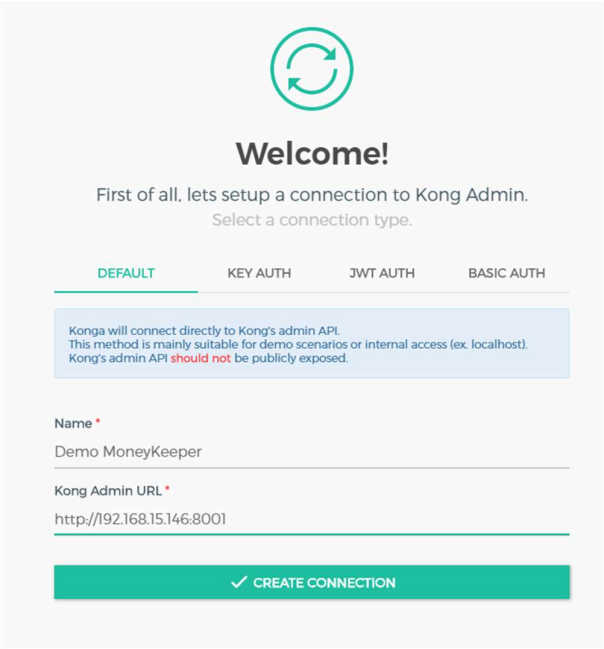
Cài đặt konga:

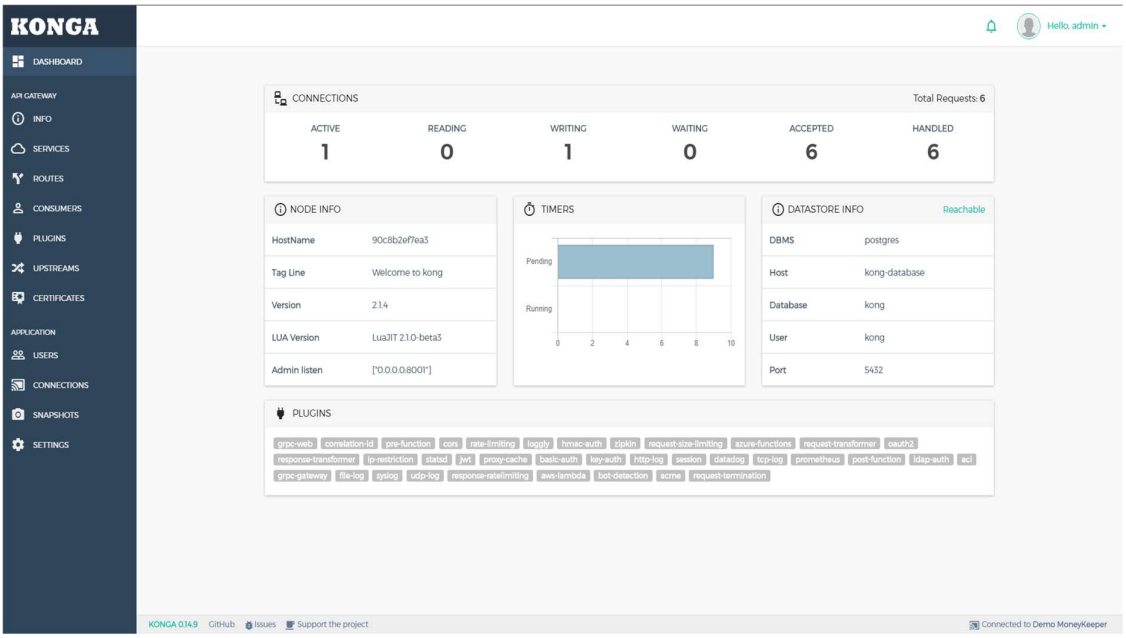
```
docker run -d -p 1337:1337 --name konga --network kong-net --link kong:kong -e "KONG_DATABASE=postgres" -e "KONG_PG_HOST=kong-database" -e "KONG_PG_PASSWORD=kong" -e "NODE_ENV=development" pantsel/konga
```

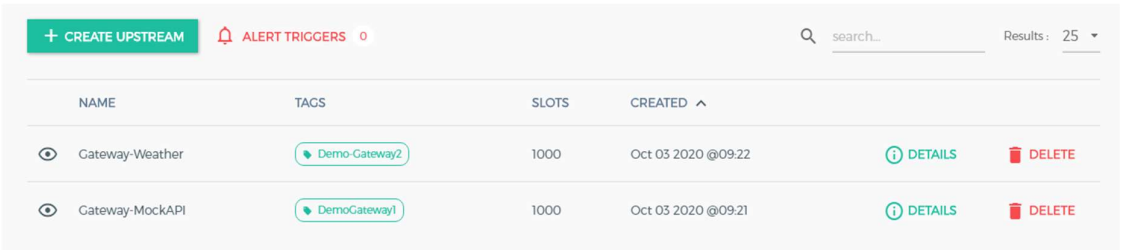
Konga là bản UI để thuận tiện cấu hình, cần phải kết nối Konga với Kong.
Database sử dụng để lưu trữ Kong là Postgres.

Một số hình ảnh của Kong và Konga sau khi cài đặt:









b. Sinh bộ mã khoá công khai và khoá cá nhân

Sử dụng OpenSLL để sinh bộ khoá certificate.

```
openssl req -x509 -newkey rsa:4096 -sha256 -nodes -keyout example.key -
out example.crt -subj "/CN=example.com" -days 3650
```

Chuyển đổi certificate sang file .pfx

```
openssl pkcs12 -export -out example.pfx -inkey example.key -in
example.crt -certfile example.crt
```

Sẽ có yêu cầu đặt mật khẩu cho file, đặt mật khẩu và nhấn Enter.

Chuyển file .pfx sang key-pair

```
openssl pkcs12 -in example.pfx -nocerts -nodes -out sample.key
```

Từ key-pair sẽ lấy được private-key

```
openssl rsa -in example.key -out sample_private.key
```

Lấy public-key từ key-pair

```
openssl rsa -in example.key -pubout -out sample_public.key
```

Khi lấy được public_key, sẽ dùng mã này để cấu hình trên Kong để "ký" token. Kong sẽ giúp xác thực token khi đi qua Kong. Ở đây, sử dụng plugin JWT của Kong. Cấu hình các tham số client_id, public_key tương ứng với plugin.

2. Cài đặt và cấu hình IdentityServer4 và Identity

a. Cấu hình IdentityServer4

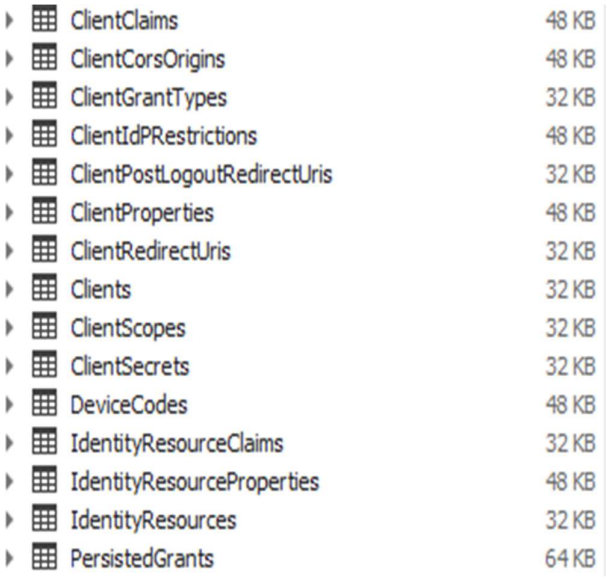
Tạo mới project Asp.net core Empty project. Cài đặt lệnh để tạo form IdentityServer. Ở đây, dùng IdentityServer kết hợp với Identity. Sử dụng Framework EntityFramework với cơ sở dữ liệu MariaDb. Chạy lệnh sau để cài bộ mẫu của IdentityServer4 hỗ trợ Entity Framework Core.

```
dotnet new is4ef
```

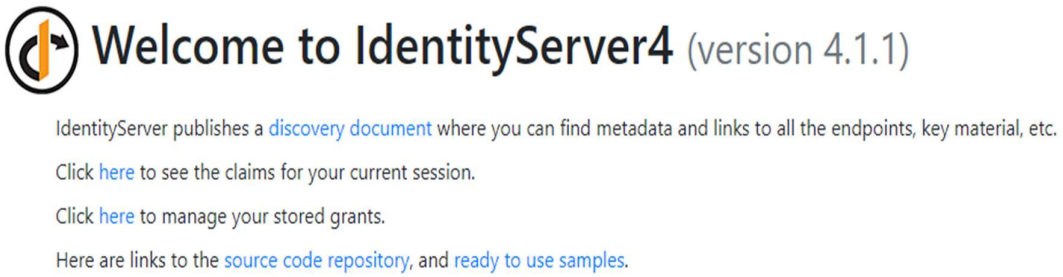
Cấu hình IdentityServer4 ở file Startup.cs.

```
var builder = services.AddIdentityServer()
    .AddConfigurationStore(options =>
    {
        options.ConfigureDbContext = builder => builder.UseMySql(connectionString, sql => sql.MigrationsAssembly(migrationsAssembly));
    })
    .AddOperationalStore(options =>
    {
        options.ConfigureDbContext = builder => builder.UseMySql(connectionString, sql => sql.MigrationsAssembly(migrationsAssembly));
    })
    .AddSigningCredential(cert)
    .AddInMemoryPersistedGrants()
    .AddInMemoryIdentityResources(Config.GetIdentityResources())
    .AddInMemoryApiResources(Config.GetApiResources())
    .AddInMemoryApiScopes(Config.ApiScopes)
    .AddInMemoryClients(Config.Clients)
    .AddAspNetIdentity<MoneyKeeperUser>();
```

Cấu hình IdentityServer4, với 2 DbContext: PersistedGrantDbContext (quản lý các thông tin Client, Scope, API, …) và ConfigurationDbContext (quản lý access_token, refresh_token, …).



Sau khi cấu hình, sẽ tạo ra được một server, chạy project, ta có kết quả tương tự như:



Với IdentityServer4, sẽ có các enpoint được hỗ trợ để quản lý tài khoản, sẽ tương tự như sau:


```
"issuer": "https://localhost:5001",
"jwks_uri": "https://localhost:5001/.well-known/openid-configuration/jwks",
"authorization_endpoint": "https://localhost:5001/connect/authorize",
"token_endpoint": "https://localhost:5001/connect/token",
"userinfo_endpoint": "https://localhost:5001/connect/userinfo",
"end_session_endpoint": "https://localhost:5001/connect/endsession",
"check_session_iframe": "https://localhost:5001/connect/checksession",
"revocation_endpoint": "https://localhost:5001/connect/revocation",
"introspection_endpoint": "https://localhost:5001/connect/introspect",
"device_authorization_endpoint": "https://localhost:5001/connect/deviceauthorization",
"frontchannel_logout_supported": true,
"frontchannel_logout_session_supported": true,
"backchannel_logout_supported": true,
"backchannel_logout_session_supported": true,
```


Cấu hình certificate đã sinh ở bên trên (example.pfx) vào identityServer4. Token mà identityServer4 sinh ra phải cùng một certificate (cặp public_key và secret) với cấu hình trên Kong. Vì phải giống nhau thì Kong mới có thể giải mã được token.

```
X509Certificate2 cert = null;
using (X509Store certStore = new X509Store(StoreName.My, StoreLocation.CurrentUser))
{
    certStore.Open(OpenFlags.ReadOnly);
    X509Certificate2Collection certCollection = certStore.Certificates.Find(
        X509FindType.FindByThumbprint,
        // Replace below with your cert's thumbprint
        "CB781679561914B7539BE120EE9C4F6780579A86",
        false);
    // Get the first cert with the thumbprint
    if (certCollection.Count > 0)
    {
        cert = certCollection[0];
        Log.Logger.Information($"Successfully loaded cert from registry: {cert.Thumbprint}");
    }
}

// Fallback to local file for development
if (cert == null)
{
    cert = new X509Certificate2(Path.Combine(Environment.ContentRootPath, "example.pfx"), "12345678@Abc", X509KeyStorageFlags.MachineKeySet);
    Log.Logger.Information($"Falling back to cert from file. Successfully loaded: {cert.Thumbprint}");
}
```

b. Cấu hình Asp.net Core Identity

```
// ===== Add our DbContext ========
services.AddDbContext<AccountDbContext>(options => options.UseMySql(connectionString, sql => sql.MigrationsAssembly(migrationsAssembly)));

// ===== Add Identity ========
services.AddIdentity<MoneyKeeperUser, IdentityRole>(options =>
        options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<AccountDbContext>();
```

Để tương tác với Database (ở đây sử dụng Mariadb) và Identity hỗ trợ thì EntityFramework là một lựa chọn phù hợp. Để sử dụng EF Core, trong dự án

cần cài đặt các Package như EntityFramework.Tool, EntityFramework.Design, Pomelo.EntityFramework.Mysql.

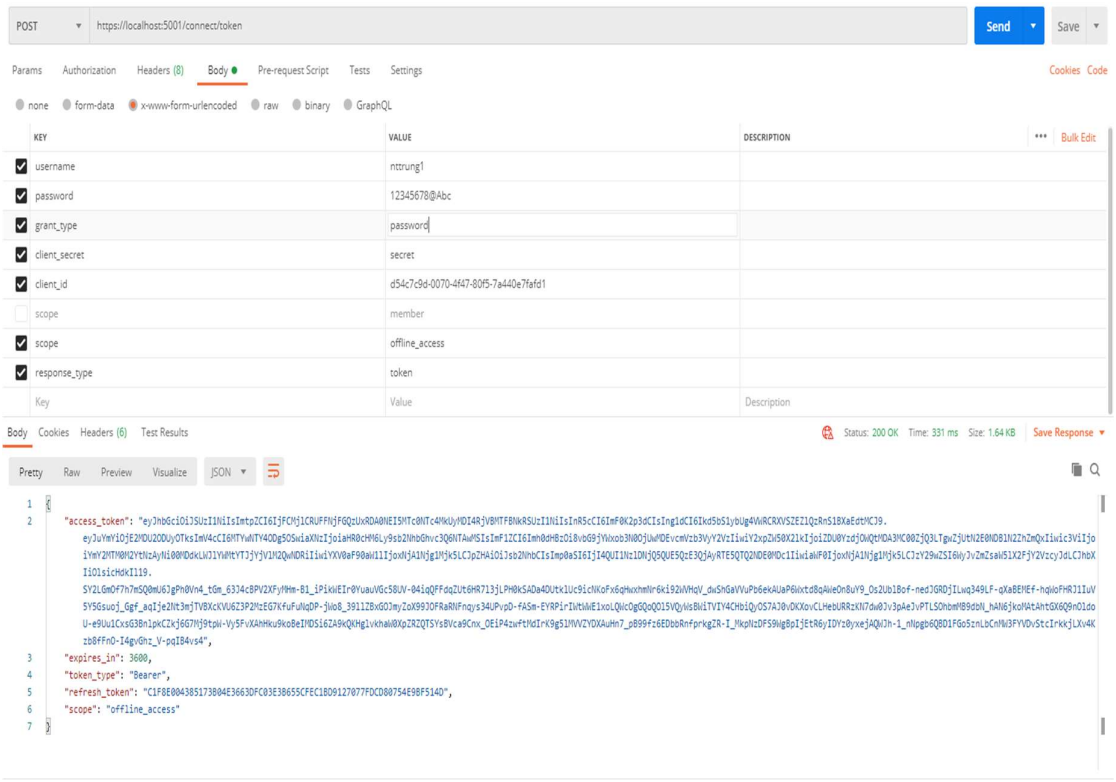
Cấu hình EntityFramework Core tạo ApplicationDbContext kế thừa Identity. DbContext này sẽ quản lý các thông tin liên quan đến tài khoản người dùng.

Đầu tiên, đăng ký một User (CreateUserAsync bằng Identity Asp.net core), khởi tạo được bắt đầu từ việc tạo mới một IdentityUser (hoặc ApplicationUser kế thừa IdentityUser), đặt Username, password, email cho người dùng đó. Nếu đây là người dùng external (các tài khoản Facebook, Google, Github, …) từ một bên thứ ba thì cần đặt trạng thái sẵn sàng mật khẩu và kích hoạt email sẵn, thường sẽ không yêu cầu xác thực rmail cho các tài khoản này. Sau đó ở trong bảng AspNetUsers được đăng ký:

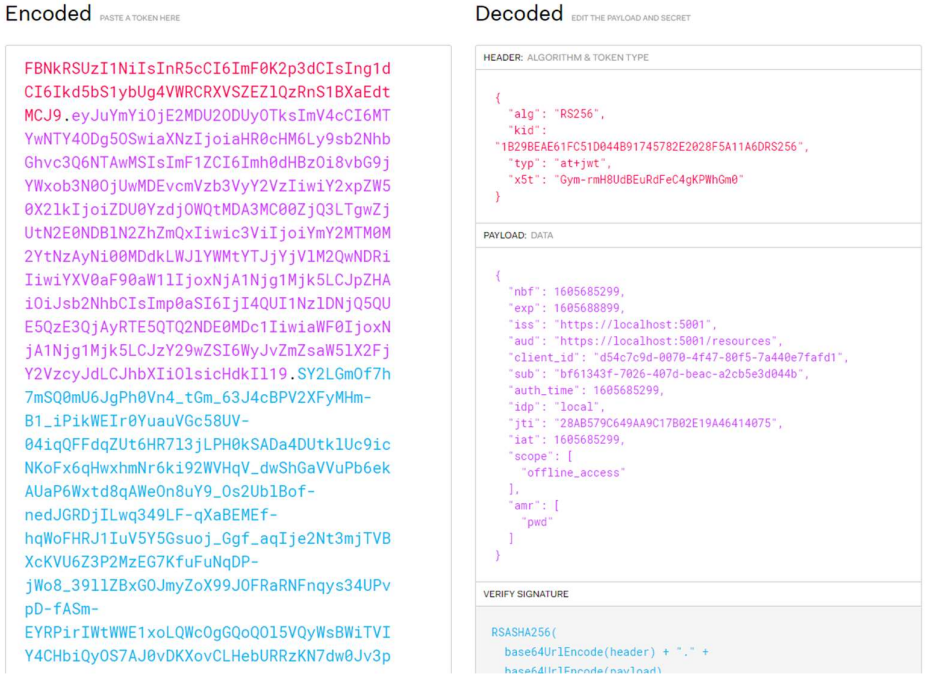
Id VARCHAR(255)	UserName VARCHAR(256)	NormalizedUserName VARCHAR(256)	Email VARCHAR(256)	NormalizedEmail VARCHAR(256)	EmailConfirmed TINYINT(1)	PasswordHash LONGTEXT
be807ed6-5458-4e34-9849-ede14f822e61	nttrung	NTTRUNG	nttrung@gmail.com	NTTRUNG@GMAIL.COM	1	AQAAAAEAACcQAAAAEKr4WLuPKRXPkhdAc1WK0PRq+7jITuVUJo/9QV7STfShc6iRj+fFGuG3VYHCH...
bf61343f-7026-407d-beac-a2cb5e3d044b	nttrung1	NTTRUNG1	nttrung@gmail.com	NTTRUNG@GMAIL.COM	1	AQAAAAEAACcQAAAAELpuX5RaEnYdi6BA/sWbftrygJ+g7EZhPNfTeTj+X2hLM1YCOaKg/NH8iq60e...

Với việc đăng nhập, sẽ gọi đến end-point token của IdentityServer4 (/connect/token) để lấy mã access_token và refresh_token, kết quả sẽ như sau:

Khi grant là Owner thì grant_type yêu cầu là password và các trường username và password là bắt buộc. client_id ở đây là định danh cho client đã đăng ký với IdentityServer4. Trường refresh_token được đưa ra vì trong cấu hình client *AllowOfflineAccess = true.*



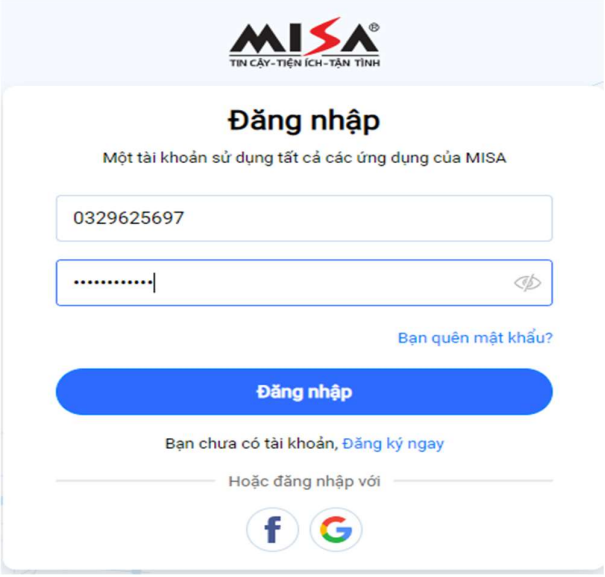
Với access_token có được, thì các Claims mặc định bao gồm:



3. Cài đặt WebApi phục vụ cho việc đăng nhập, đăng ký

a. Đăng nhập với MISAID

Do MISAID là một hệ thống độc lập, hỗ trợ tương đối đầy đủ cho việc xác thực tài khoản, nên công việc này khá đơn giản, gọi API của MISAID theo đúng đặc tả và xử lý lưu vết thông tin người dùng trên cơ sở dữ liệu.



Sau khi đăng ký, đăng nhập thành công, MISAID sẽ sinh ra một bộ access_token và refresh_token. Công việc của service sổ thu chi là phải giải mã được token của MISAID. Việc này sẽ được tiến hành qua mã x5c do MISAID cung cấp. Sau khi giải mã được token của MISAID, sẽ tiến hành kiểm tra Authen và Author. Nếu qua công đoạn kiểm tra thì thông tin tài khoản sẽ được Identity quản lý như một người dùng bình thường. Và sẽ thêm một bản ghi đánh dấu đăng nhập từ bên thứ 3. Đó là provider và providerkey (là misaid cung cấp) ở trong bảng Aspnet_UserLogins.

b. Đăng ký với tài khoản mạng xã hội

Tích hợp việc đăng ký với các tài khoản mạng xã hội bằng API (vì đây là nâng cấp hệ thống back-end cho mobile tích hợp). Nhận vào access_token của các mạng xã hội cung cấp, sau đó xử lý, kiểm tra authen, author, cuối cùng trả ra thông tin đăng nhập người dùng.

Từ acccess_token client gửi lên, dùng token này gọi lên api các trang mạng xã hội tương ứng để lấy về các thông tin người dùng. Như provider (tên nhà mạng xã hội cung cấp dịch vụ: Facebook, Google, Apple, …), providerKey (định danh của người dùng trên các trang mạng xã hội), DisplayName (là tên hiện thị của người dùng, thường là họ và tên của người dùng).

Từ một cặp provider và providerKey cho phép định danh ra một người dùng đăng nhập bằng tài khoản mạng xã hội. Sử dụng Asp.net Identity Core với API AddLoginAsync() sẽ tạo ra một người dùng (userId) với đầu vào là LoginUserInfo (là các thông tin lấy được từ các trang mạng xã hội).

Sau khi tạo được người dùng (nếu là lần đầu đăng nhập), ta sẽ dùng email và một mật khẩu mặc định để tạo mới người dùng quản lý trong IdentityServer. Khi có được user, sẽ gọi end-point token của IdentityServer để lấy mã access_token và refresh_token.

c. Quản lý thông tin người dùng

Là các dịch vụ cho phép lấy thông tin người dùng, đổi mật khẩu, quên mật khẩu, xác thực email, bật xác thực 2 lớp đăng nhập, …

Identity hỗ trợ các api quản lý thông tin người dùng như tạo mới người dùng (CreateUserAsync), tìm kiếm người dùng (FindUserByName). Thông tin người dùng Identity được lưu trữ ở bảng AspNetUsers, riêng các tài khoản đăng nhập bằng tài khoản mạng xã hội khác thì được lưu trữ thêm thông tin ở bảng AspNet_UserLogins với các trường Provider và ProviderKey kết hợp tạo thành một định danh cho người dùng. Trong các trường hợp, CreateUser sẽ được gọi trước, sẽ tạo ra một userId, từ UserId này sẽ là khoá ngoại cho các bảng khác.

Các dịch vụ này đều yêu cầu token, ở đây không phải là access_token. Ứng với mỗi nghiệp vụ, IdentityServer4 và Asp.net Identity Core sẽ có cơ chế sinh các token riêng.

d. Cơ chế lấy lại access_token

Một token sẽ có thời gian có hiệu lực, khi acess_token hết hiệu lực, người dùng phải gửi refresh_token lên để lấy lại access_token. Đây là việc giúp cho một tài khoản đăng nhập một lần, mà có thể sử dụng nhiều lần access_token (mặc dù access_token có thời gian hiệu lực khá ngắn). Cấu hình cho client để xét refresh_token:

AllowOfflineAccess = true,

AccessTokenLifetime = 3600,

RefreshTokenUsage = TokenUsage.OneTimeOnly,

RefreshTokenExpiration = TokenExpiration.Sliding,

SlidingRefreshTokenLifetime = 36000,

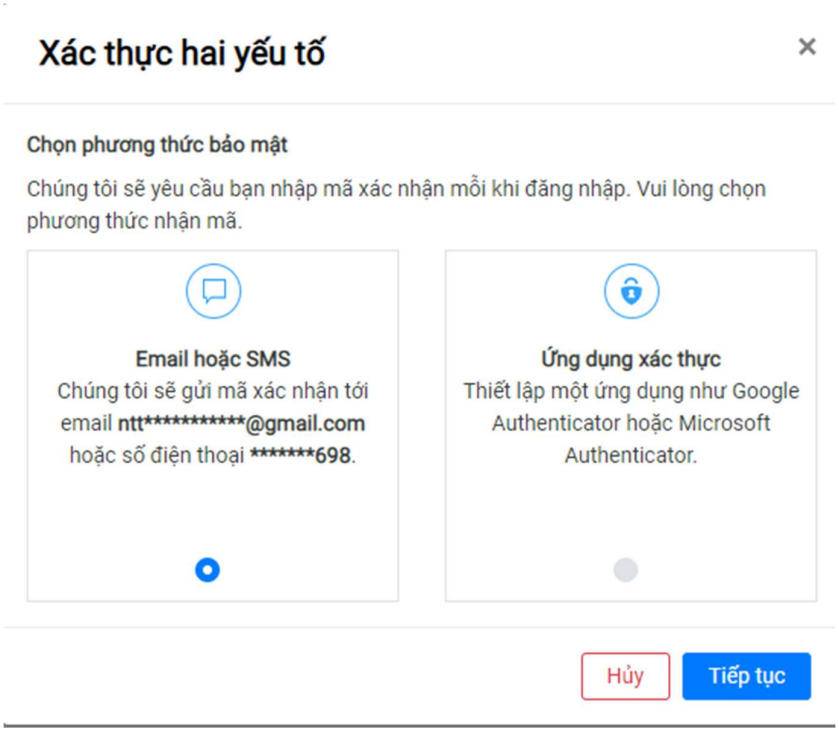
Để cấu hình cho việc lấy lại access_token trong IdentityServer4, đặt trường AllowOfflineAccess = true. Khi gọi API lên lấy access_token thì scope cần đặt thành refresh_token. Trong cấu hình AccessTokenLifetime đặt là 3600 giây (1 giờ). Có nghĩa là sau một giờ kể từ khi access_token được cung cấp, nó sẽ hết hạn. Sau một giờ này, access_token không còn hiệu lực, muốn lấy lại token, ta cần một refresh_token.

Với cài đặt RefreshTokenUsage đặt là OneTimeOnly có nghĩa là refresh_token chỉ được sử dụng một lần duy nhất, sau khi sử dụng, token này sẽ bị xoá bỏ, không có hiệu lực. Cũng có thể đặt là ReUse nếu muốn sử dụng lại nhiều lần refresh_token này.

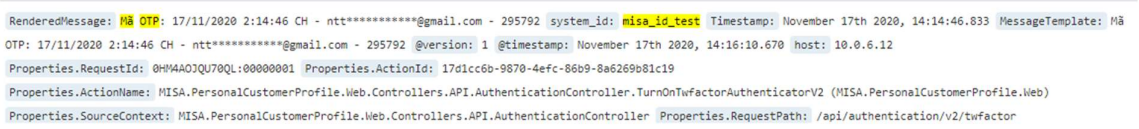
Với cài đặt Sliding là 36000 giây, tức là 10 giờ, có nghĩa là refresh_token này chỉ có hiệu lực trong vòng 10 tiếng. Sau khi hết thời gian này, token này cũng hết hiệu lực. Muốn cấp lại refresh_token, người dùng bằng cách đăng nhập lại tài khoản của mình.

e. Xác thực đăng nhập hai lớp

Identity và IdentityServer4 đều hỗ trợ việc xác thực hai lớp. Khi bật tính năng xác thực hai lớp, hệ thống sẽ gửi một mã OTP về mail hoặc số điện thoại (tuỳ thuộc vào việc xác thực khi đăng ký). Hoặc cũng có thể sử dụng ứng dụng xác thực của Google, Facebook để hoàn thành việc xác thực này.



Sử dụng mã OTP này để kích hoạt đăng nhập hai lớp. Trên đây là đăng ký một tài khoản test nên mã này sẽ được xử lý và lưu vào log ELK.

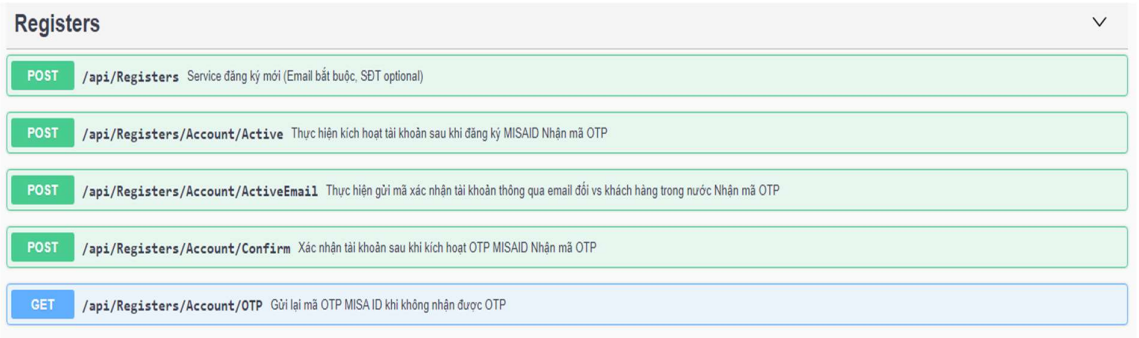


Đây là ví dụ về mã OTP xác thực đăng nhập hai lớp được ghi ra log trong khi đang trong quá trình phát triển. Mã OTP này được sinh ra dựa trên API yêu cầu xác thực hai lớp. Identity sẽ tạo ra một token quản lý việc xác thực này, base64 token này chính là mã OTP được gửi về cho người dùng. Mỗi mã OTP này sẽ có hiệu lực trong một khoảng thời gian xác định được định nghĩa trước. Và mỗi mã OTP chỉ được dùng một lần duy nhất.

V. Kết quả đạt được

Từ quá trình phân tích bài toán, nghiệp vụ. Sau khi được xem xét và chỉnh sửa lại mã nguồn code. Em cũng đã tạo dựng được một microservice quản lý tài khoản khách hàng của sổ thu chi. Service này được đặt tên là MISA MoneyKeeper Identity Service. Bao gồm các cụm API:

Đầu tiên, là cụm API liên quan đến việc đăng ký tài khoản MISAID:



Tạo liên kết đăng ký với MISAID

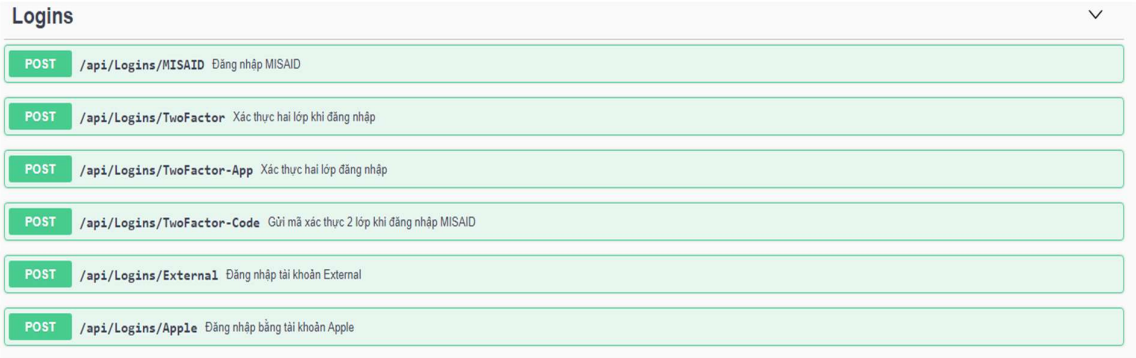
Lấy mã OTP kích hoạt tài khoản

Gửi mã OTP về email người dùng khi đăng ký bằng email

Thực hiện xác thực mã OTP

Gửi lại mã OTP (trong trường hợp OTP hết hạn)

Thứ hai, là cụm API phục vụ đăng nhập

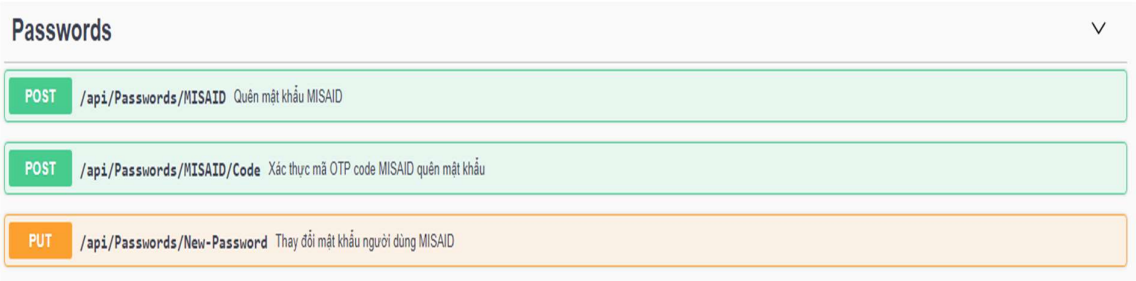


Có ba hình thức đăng nhập: Đăng nhập qua MISAID, External (Facebooke, Google) và AppleId

API lấy mã xác thực hai lớp khi người dùng bật tính năng và API xác thực mã.

Ngoài ra, còn cũng có API hỗ trợ xác thực qua app của Facebook và Google.

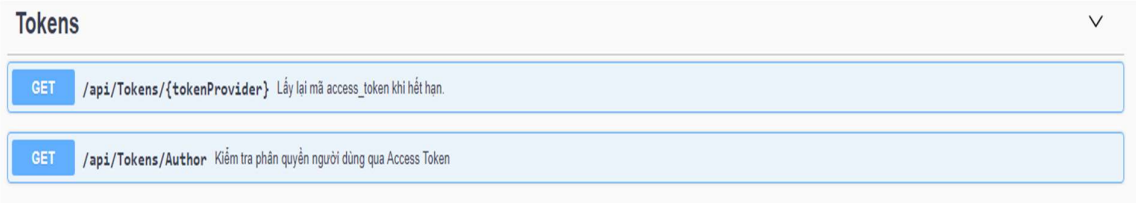
Tiếp theo là cụm API mật khẩu



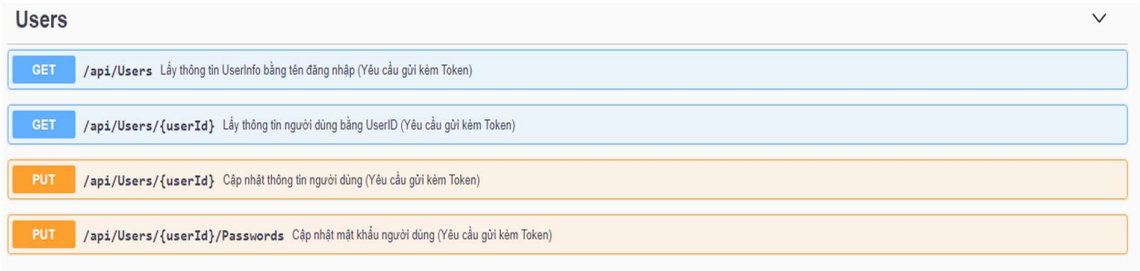
Các API liên quan đến mật khẩu hiện tại chỉ áp dụng cho tài khoản MISAID. Còn nội bộ của sổ thu chi, mật khẩu được đặt trong cụm API User. Bao gồm:

API gửi yêu cầu đổi mật khẩu MISAID. Sau đó MISAID sẽ cung cấp một mã OTP để xác thực. Người dùng gọi tiếp API để xác thực. Xác thực mật khẩu thành công thì gọi API để đổi mật khẩu.

Cụm API quản lý Token người dùng. Ở đây, có 2 API là lấy lại mã access_token khi mã này hết hạn, và một API kiểm tra phân quyền dựa trên access_token.



Và cụm API về thông tin người dùng. Gồm việc lấy thông tin người dùng, cập nhật thông tin người dùng.



VI. Hướng phát triển

Để sản phẩm mang được tính ứng dụng cao và được người dùng ưa thích, đáp ứng được nhu cầu dữ liệu ngày càng lớn, yêu cầu bảo mật ngày càng cao thì tương lai cần cải thiện lại một số vấn đề.

Một là, cơ chế xác thực Authen. Việc sinh access_token và refresh_token cần tuân thủ đúng quy trình. Mã public_key được ký gửi trên Kong cần có cơ chế bảo mật. Kong đã hỗ trợ xác thực Authen và Author cho API qua token. Nhưng cần kiểm tra lại ở service. Vì tuân theo cơ chế, việc mã hoá và giải mã dữ liệu cần được thực hiện ở các thiết bị hai đầu.

Hai là, cơ chế log lỗi. Giải pháp được lựa chọn là ELK. Cần kiểm tra lại và tối ưu việc log lỗi. Đảm bảo tuân thủ Restful API và OpenAPI Spec.

Ba là, thực hiện chuyển đổi các tài khoản đăng nhập Facebook, Google, AppleID lên MISAID. Nơi quản lý tập trung tài khoản của MISA. Tránh việc thất thoát dữ liệu, khó quản lý thông tin khách hàng.

Bốn là, chuyển đổi cơ chế authen cũ lên cơ chế authen mới. Vì một vài lý do, một số người dùng chưa thể chuyển đổi lên phiên bản mới. Nên cần phải tìm một giải pháp xác thực cho các tài khoản cũ.

Năm là, tối ưu hoá hiệu năng. Bảo mật là ưu tiên hàng đầu, nhưng hiệu năng cũng rất quan trọng. Cần cải tiến, tối ưu để năng hiệu năng cho hệ thống. Tinh chỉnh, giúp client cũng như việc tương tác giữa các service thuận tiện, an toàn hơn.

Đánh giá của giảng viên hướng dẫn

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Giảng viên hướng dẫn

TS. Lê Đình Thanh