Documentación API Account Test

Proyecto que permite la creación de una cuenta de usuario para realizar diferentes tipos de operaciones bancarias (depósitos, Retiros y transferencia entre cuentas). Este proyecto más allá de permitir la creación de una cuenta también incluirá la funcionalidad de que dos o mas usuarios puedan compartir un balance bancario, esto implica que dichos usuarios podrás realizar retiros y transferencias desde un mismo balance.

Configuración inicial

Cambiar en el archivo *JSON* el nombre de la instancia SQL donde se va a crear la base de datos.

Proyectos

Account.API: Capa principal de la aplicación, expone los endpoints necesarios para el funcionamiento de la aplicación.

Account.Entities: Capa que contiene las clases para obtener y enviar información desde y hacia la base de datos.

Account.DTO: Capa que contiene las clases para ingresar información o realizar algún tipo de operación en la aplicación y para retornarla cuando se necesita exponer dicha información ingresada.

Account.InfrastructureEF: Capa donde se manejan la configuración de las conexiones a la base de datos, usando las entidades en la capa de *Account.Entities.* Se usa EntityFrame Core 3.0 y se realizan las migraciones de manera automática.

Account.Interfaces: Capa donde se definen todas las interfaces necesarias para diseñar la arquitectura del proyecto.

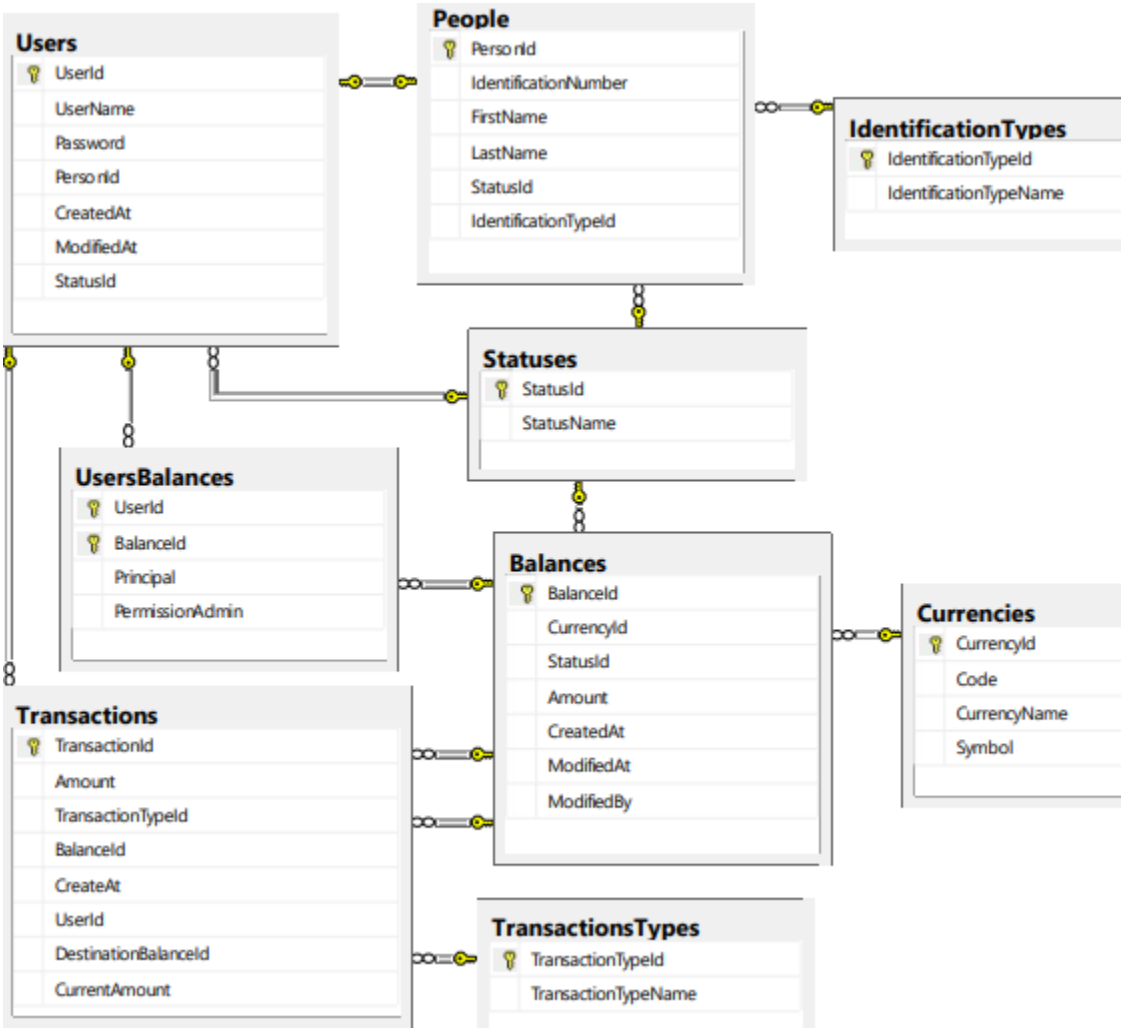
Account.DataAccessLayer: Capa donde se encuentras todos los métodos dedicados a ingresar, modificar, eliminar y consultar información hacia la base de datos. Es la capa que se encuentra entre la base de datos y la lógica del negocio.

Account.BusinessLayer: Capa que contiene toda la lógica de la aplicación. Donde se realizar las llamadas a la capa de acceso a datos, validaciones, se recibe la información desde los endpoint y se procesa toda la lógica necesaria.

API.Helper: Proyecto donde se ubican funciones que nos ayudan en el mantenimiento de la aplicación. Como manejo de excepciones y Logger.

Base de datos: Se usa base de datos SQL Server. Realizando toda la creación desde el proyecto con EntityFrame Core 3.1.

Esquema de base de datos



Balance: Esta tabla almacena los tipos de cuenta que puede tener un usuario. Y se le puede especificar una moneda, para saber en qué moneda se maneja un balance. Por los momentos solo esta preparado para usar Euros. En el futuro un usuario podría manejar distintas monedas (Euro, Dólar, Reales, Pesos etc).

UsersBalances: Esta tabla permite crear la funcionalidad que un puede tener diferentes balances, pero a la vez también permite pueda compartir los permisos de un balance con otros usuarios. Se podrían crean cuentas comunes entre usuarios. Ya sea el caso de familiar, cónyuge, negocio.

- **Principal:** Define quien creo el balance
- **PermisosAdmin:** Define el nivel de permiso que puede tener el usuario invitado a compartir el balance. El usuario creador de este balance siempre tendrá el mas alto nivel y no será posible cambiar dicho permiso.

Transactions: Esta tabla permite registrar los diferentes tipos de transacciones que se realicen a un balance (Depósitos, retiros, transferencias).

- **UserId:** Define cual fue el usuario que realizo una transición.
- **DestinationBalanceId:** Esta columna sirve para cuando se realiza una transacción de transferencia, identificar el balance de hacia donde va o de donde vino el dinero.
- **CurrentAmount:** Columna que permita guardar el balance después de haber realizado la transacción.

TransactionsTypes: Guarda los tipos de transacciones que se pueden realizar (Deposito, retiro, transferencia).

Autenticación y Autorización

Para la seguridad de la aplicación que usa la librería *Json Web Token,* realización la configuración en la clase Startup del proyecto. La misma usa una clave secreta que se encuentra almacenada en el archivo *JSON*. Para mayor seguridad dicha clave se puede almacenar en variables de ambientes o en algún repositorio de clases como *Azure key vault* en el caso de usar Azure.

Si la API se a alojar en una zona no publica y se va a acceder a ella a través de un API Gateway se podrían dejar dicha información en el archivo JSON.

EndPoint

Authenticate (SecurityController): Servicio utilizado para para la autenticación de un usuario y que regresa un token que permite realizar peticiones a los demás servicios.

Post (UserController): Servicio para registrar un nuevo usuario.

Get (UserController): Servicio que obtiene la información de un usuario.

Post (TransactionController): Servicio para realizar los diferentes tipos de transacciones que puede hacer la aplicación.