ALTERNATIVAS DIAGRAMA DE CLASES

Organización

1. Modelar la alternativa como un **Singleton** debido a que necesitamos una única instancia de la organización.
2. Podríamos también definir a la organización como una clase que puede representar no solo a organización sino como también a proveedores. No es del Scope para esta entrega.

Tipos de Organización

1. En el caso de las entidades jurídicas, definimos **CategoriaEntidadJuridica** como un enum, debido a que, según la información de la AFIP, por scope, no nos solicita ningún dato adicional (ventas totales anuales, actividades incluidas, excluidas, activos o personal ocupado).
2. Se podría definir **CategoriaEntidadJuridica** como una interfaz, y cada categoría (MICRO, PEQUEÑA, MEDIANA TRAMO 1, MEDIANA TRAMO 2, OSC) como una clase con los atributos ventas totales anuales, actividades incluidas, excluidas, activos o personal ocupado.

Egresos

1. Definimos que una Organización es una clase que tiene como atributo un listado de **operaciones**, que es una interfaz cuyas sub clases son **EGRESO** e **INGRESO**. Esto nos permite, que el día de mañana, podamos agregar N sub clases con diferentes métodos y atributos.
2. Otra alternativa es que exista la clase **EGRESO** que hereda de Organización que tiene un **DocumentoComercial**, un **Proveedor**, un **monto** y una **fechaDeCreacion**. Esto nos permite resolver el scope de la entrega pero no tiene escalabilidad, ya que, a la hora de existir otro tipo de Operación, debo forzar que los datos vengan de la manera DocumentoComercial, Proveedor, monto y fechaDeCreacion.
3. Una tercera alternativa fue agregar una clase TIPOOPERACION y que contenga las sub clases Egreso e Ingreso. Esta alternativa fue descartada porque lo correcto en esos casos, según method template, es usar una interfaz.

Referencia De Egreso

Es lo mismo que Egreso, decidimos hacer una interfaz **ReferenciaDePago** ya que existen N cantidad de egresos y todos pueden ser modelados de distinta manera. En el primer caso, modelamos la **compra** como una clase que contiene Items y Provedor.

Métodos De Pago

1. Optamos por generar una interfaz de la cual se desprenden las clases de los diferentes métodos de pago con sus respectivos atributos y métodos para simplificar el diagrama y no tener que agregar mas clases/interfaces en el medio.
2. Pensamos desprender de la interfaz MetodoDePagoYCobro, las clases **EFECTIVO** y **TARJETA** y que, de cada uno de ellos, se desprendan **enum** con los métodos propiamente dichos. En este caso, entendemos que los métodos de Efectivo o Tarjeta, van a ser similares para todos los contenidos.

Documentos Comerciales

1. Definimos DocumentosComerciales como una interfaz, nuevamente usando method template, para poder dividir en sub clases cada uno de los tipos de documentos comerciales, los cuales tendrán diferentes atributos.
2. Otra alternativa es definir a DocumentosComerciales como clase y tener atributos y métodos iguales para todos los tipos de documentos.

VALIDACION DE USUARIO

Chequeo de contraseñas contra listado de contraseñas no recomendadas (obligatorio)

1. Lectura de archivo e iteración buscando coincidencia
2. Verificar contraseñas vs un Array de Strings.
3. Validando contra algún servicio externo.

Validar Longitud (8 caracteres)

Validar que el String usuario sea diferente que String contraseña

Validar que la contraseña no tenga caracteres repetidos y/o consecutivos (aaaa o 1234abcd)

1. Expresión Regular

Validación de contraseña segura contra un servicio externo (<https://cmatskas.com/password-validation-and-creation-using-a-public-api/>)(<https://github.com/ankurpathak/password-validation>)