



GeSoc

Sistema de Gestión de Proyectos Sociales

Documento de Justificaciones - Entrega 4.2

Grupo 5

Apellidos	Nombres	Legajo	Mail
COVELLO	JORGE JULIO FERNANDO	163.661-3	jcovello@est.frba.utn.edu.ar
CUELI	JUAN FRANCISCO	163.279-6	jcueli@est.frba.utn.edu.ar
LEGUIZAMÓN	ROCÍO	163.609-1	rleguizamon@est.frba.utn.edu.ar
LESNIAK	MATIAS	162.992-0	mlesniak@est.frba.utn.edu.ar
URTEAGA NAYA	MARTIN IGNACIO	164.412-9	martin.un@est.frba.utn.edu.ar

Primera Sección

Para comenzar identificamos las siguientes entidades fundamentales:

- Usuario (pers_usuarios): es quien usará nuestro sistema, por ello es fundamental persistir sus datos como su nombre, su contraseña (esta será mapeada mediante el uso la biblioteca SHA256 para garantizar la seguridad de la misma) y su rol. Asimismo, el Usuario posee una Primary Key (PK) autogenerada con el objetivo de crear un identificador inequívoco y además, nos es más sencillo para la utilización del ORM.
- Egreso (dom_egresos): representa el dinero que la empresa gasta. Esta entidad es fundamental en nuestro dominio debido al hecho de que el sistema es de gestión contable. De forma similar al Usuario, el Egreso tiene una PK autogenerada para poder identificarlos inequívocamente y para facilitar la utilización del ORM. También cuenta con siete FK siendo ellas:
 - Proveedor (pers_proveedores): es a quién le damos el dinero para que nos brinde un determinado bien o servicio.
 - DocumentoComercial (dom_documentos_comerciales): respalda legalmente al Egreso.
 - Ingreso: representa el dinero que ingresa a la empresa y con dicho monto se puede afrontar los egresos de la misma.
 - Criterio (dom_criterios): tiene como objetivo clasificar los egresos de la organización.
 - Entidad (dom_entidades): es la empresa que usará nuestra sistema y por ende tendrá muchos egresos asociados.
 - Valor (mon_valor): representa el monto del egreso.

También posee un atributo para la fecha en el que se realizó el egreso, un boolean para saber qué egresos hay que validar, el tipo de medio de pago y su descripción.

- Ingreso (dom_ingresos): por el contrario al Egreso, el Ingreso representa el dinero que entra a la organización y es fundamental persistir esta entidad ya que es vital en nuestro sistema de gestión contable. Aquí también se utiliza una PK autogenerada inequívoca para poder identificar fácilmente cada Ingreso y para utilizar sin problemas el ORM. Además, posee dos FK, una para relacionarse con la tabla Entidad ya que cada una de ellas puede tener muchos ingresos y otra que se relaciona con la tabla Valor que representa el monto del Ingreso en cuestión. Finalmente, posee una fecha para saber cuándo sucedió el Ingreso y una descripción para datos extra del mismo.
- Proveedor (pers_proveedores): es quien le brinda un bien o servicio a la organización. Tiene una PK autogenerada e inequívoca para tener bien identificado a cada Proveedor. Inicialmente pensamos en modelar la dirección del proveedor con cuatro tablas (Dirección, DireccionPostal, Ciudad, Provincia y País) pero no era performante ya que para obtener la dirección completa era necesario realizar cinco JOINS. Por esta razón, decidimos desnormalizar la dirección del Proveedor y agregarle a dicha tabla la calle, la altura de dicha calle, el piso y el departamento (en caso de necesitarlos). Siguiendo con lo anterior, la tabla Proveedor posee un FK para relacionarse con la tabla Ciudad en caso de que se requiera más información de la dirección (es decir, la

ciudad, la provincia y el país). De esta manera, se reducen los posibles JOINS ya que el [Proveedor](#) posee todos los datos de su dirección en su propia tabla y sólo se realizaría un JOIN si surge la necesidad de obtener datos más específicos (de la tabla [Ciudad](#)) como los mencionados anteriormente. Asimismo, cuenta con atributos importantes como razón social, nombre y DNI/CUIT.

- [Presupuesto](#) (dom_presupuestos): representa el precio de un bien o servicio que la empresa deberá pagar en caso de adquirirlo. En esta tabla hay una PK autogenerada con el fin de poder identificar cada Presupuesto. Además, posee dos FK, una para el [DocumentoComercial](#) que acompaña al [Presupuesto](#) y otra para establecer una relación con los proveedores ya que un [Proveedor](#) puede presentar varios presupuestos en la organización.
- [Entidad](#) (dom_entidades): nos permite diferenciar entre entidades jurídicas y entidades base mediante el campo *tipoEntidad* (**B** si es Base y **J** si es Jurídica). Su PK es autogenerada y numérica para hacer buen uso del ORM.
- [TipoEntidadJuridica](#) (dom_entidades_juridicas): representa los datos de las entidades jurídicas dados por el dominio, deben ser almacenados para su futura consulta o modificación en caso de que se quiera agregar un nuevo egreso/ingreso o se quiera re-categorizar dicha entidad. Está siendo identificada por una PK autogenerada y numérica para facilitar las consultas a través del ORM. Asimismo, tiene una dirección que está formada por la calle, la altura de dicha calle, el piso y el departamento (en caso de necesitarlos). Análogamente con lo modelado en la tabla [Proveedor](#), [TipoEntidadJuridica](#) también tiene un FK a [Ciudad](#) en caso de que se necesiten más datos sobre su dirección, evitando así, JOINS innecesarios. Una FK a la [Categoría](#) a la cual pertenece dicha Entidad, y una FK al [Sector](#) de la entidad.
- [Categoría](#) (ent_categorias): representa los valores asociados (promedio de ventas anuales y personal máximo) de cada tipo de categoría. Contiene una PK autogenerada y numérica junto con los atributos previamente mencionados. Además, posee una FK a la tabla [Sector](#) debido a que un sector contiene muchas categorías. Es necesario aclarar que decidimos no modelar una relación muchos a muchos entre [Categoría](#) y [Sector](#) porque no existe más de una categoría que pertenezca a distintos sectores. En otras palabras, habría un muchos a muchos entre el tipoDeCategoría y el [Sector](#) pero como dicho atributo está desnormalizado dentro de la tabla [Categoría](#), no es necesario una relación de muchos a muchos.
- [Sector](#) (ent_sector): describe la actividad de una empresa y permite conocer en qué categoría está dicha empresa. Contiene una PK autogenerada y numérica junto con el nombre y la descripción del [Sector](#) en cuestión.
- [CategoríaCustom](#) (dom_categorias): representa la [categoría](#) que crea la propia organización para clasificar u ordenar sus egresos de manera que le sea más fácil operar con ellos. No tiene nada que ver con la [Categoría](#), ya que dicha tabla tiene una cantidad limitada de filas dadas por las regulaciones que se imponen en nuestro sistemas mientras que esta tabla tiene tantas filas como los usuarios lo deseen (por ahora). Esta tabla posee columnas de nombre, descripción y de id, siendo este último para almacenar el identificador de la fila. También tiene una columna con el [Criterio](#) al cual está asociado.

- Criterio (dom_criterios): es el criterio que crea dicha entidad para que se aplique a los diferentes egresos con el fin de que estos sean más fáciles de reconocer. Posee una PK numérica y autogenerada. También tiene una FK a la [Entidad](#) que creó dicho criterio, y otra al criterio que es padre. Así permitimos anidar criterios a un [egreso](#) y de esta manera se pueden crear jerarquías de [criterios](#). Además, incluimos nombre y descripción para que el usuario pueda reconocerlos más fácilmente.

Las tablas País, Provincia y Ciudad se mantienen porque buscamos persistir lo que se obtiene de la API de ML, y luego poder consultar a la base de datos directamente, ya que es mucho más eficiente. Tampoco perdemos la consistencia de estos datos, al dejarlos como tablas separadas. A su vez, podemos aprovechar las tablas para poder listar sus respectivos registros en un menú desplegable.

- Ciudad: posee un nombre (varchar) y está relacionada con la [Provincia](#) a la cual pertenece geográficamente. Su PK es numérica y autogenerada.
- Provincia: sus campos son parecidos a los de [Ciudad](#), pero en cuestiones de concepto son cosas totalmente distintas. Posee un nombre (varchar), un valor numérico autogenerado que la representa con respecto a otras tablas (PK) y su relación con el [País](#) al cual pertenece geográficamente (FK).
- País: tal como dijimos antes, se parece a las tablas previamente mencionadas pero su significado es totalmente diferente. Representa los países que reconocemos y a los cuales prestamos nuestro servicio de software. Está compuesto por un nombre (varchar) y un valor numérico que es su identificador.

Relaciones de Muchos a Muchos

- CriteriosXEgreso: como los criterios son personalizados, la empresa puede utilizarlos en más de un [Egreso](#), y a su vez, por requerimiento del enunciado, un [Egreso](#) puede tener más de un [Criterio](#) asociado.
- RevisorXEgreso: un mismo [Egreso](#) puede tener más de un revisor y los revisores se repiten en distintos egresos. Además, el [Revisor](#) puede tener muchos egresos asociados con el objetivo de revisar cada uno de ellos.

Mapeo de Herencia

- Entidad: en este caso mapeamos con la estrategia *Single Table* porque la diferencia entre “entidades” está dada por la relación entre entidades base y jurídicas que se refleja con la *relación autorreferencial*. Este campo será el único que podrá estar con valor nulo utilizando esta estrategia, por lo que consideramos que es un precio justo para el performance que nos brinda al evitar el uso de reiterados joins en las consultas a la base de datos.

- **TipoEntidadJuridica**: volvemos a usar la estrategia de mapeo *Single Table* ya que el único campo que podría estar con valores nulos es la relación con la **Categoría** correspondiente y puesto a que evitamos hacer joins, tal como aclaramos antes, nos parece correcto el uso de esta estrategia.

Queremos destacar que utilizamos composición entre las clases **TipoEntidadJuridica** y **Entidad** por lo que cuando mapeamos podría considerarse que utilizamos la estrategia de mapeo *Joined* pero como aclaramos antes surgió de la composición utilizada en el dominio de objetos y no en el mapeo de traer datos de mi dominio de objetos al mundo relacional. Por lo que nosotros **no** consideramos que fuese uso de dicha estrategia.

Ejemplos de las Tablas

Probamos la persistencia con algunos Tests y también con INSERTs directos a la base de datos.

• Proveedor

pers_proveedores

select * from pers_proveedores pp

	proveedor	apellido	altura	calle	departamento	piso	nro_documento	nombre	razon_social	ciudad
1	1	[NULL]	2454	Larralde	A	3	28-49367281-6	[NULL]	Razon Social	1

• País

geo_paises

select * from geo_paises gp

	pais	nombre
1	1	Argentina

• Provincia

geo_provincias

select * from geo_provincias gp

	provincia	nombre	pais
1	1	Buenos Aires	1

• Ciudad

geo_ciudades

select * from geo_ciudades gc

	ciudad	nombre	provincia
1	1	Adrogué	1

- Documento Comercial

dom_documentos_comerciales	
select * from dom_documentos_comerciale	
123 documento_comercial	ABC descripcion
1	N° 00003500
	[NULL]
	REMITO

- Valor

mon_valor	
select * from mon_valor mv	
123 moneda	123 importe
1	1.285
	Argentina
	\$
	Peso argentino

- Usuario

pers_usuarios	
select * from pers_usuarios pu	
123 usuario	ABC apellido
1	Palacio
	S4R4z@78P212EyR
	Lucía
	[NULL]

- Presupuesto

dom_presupuestos	
select * from dom_presupuestos dp	
123 presupuesto	fecha_creado
1	2020-05-09
	18.525
	123 documento_comercial
	123 proveedor
	1
	1

- Detalle

dom_detalle_presupuestos	
select * from dom_detalle_presupuestos ddp	
123 detalle	ABC descripcion
1	Descripción del Detalle 1
	18.250
	123 presupuesto
	1

- Mensaje

men_mensajes	
select * from men_mensajes mm	
123 mensaje	fecha_creado
1	2020-08-19
	2020-09-01
	leido
	ABC cuerpo_mensaje
	123 usuario
	1
	Mensaje 1 para e revisor
	1
	2
	2020-08-19
	[NULL]
	Mensaje 2 para el revisor
	1

- Ingreso

dom_ingresos					
select * from dom_ingresos di Enter a SQL expression to filter results (use Ctrl+Space)					
	123 ingreso	ABC descripcion	fecha	123 entidad	123 valor
1	1	Descripción del Ingreso 1	2019-09-03	1	1

- Egreso

dom_egresos						
select * from dom_egresos de Enter a SQL expression to filter results (use Ctrl+Space)						
	123 egreso	validado	fecha	ABC descripcion_metodo_pago	ABC metodo_pago	123 documento_comercial
1	1	<input type="checkbox"/>	2020-09-27	Cheque N° 0004	CHEQUE	1

- Entidad Base

dom_entidades					
select * from dom_entidades de Enter a SQL expression to filter results (use Ctrl+Space)					
	ABC tipo_entidad	123 entidad	ABC descripcion	ABC nombre	123 entidad_juridica_asociada
1	8	1	Entidad Base	Ejemplo	[NULL]

- Entidad Jurídica

dom_entidades_juridicas							
select * from dom_entidades_juridicas dej Enter a SQL expression to filter results (use Ctrl+Space)							
	123 entidad_juridica	123 cant_personal	ABC cod_igj	ABC cuil_cuil	ABC altura	ABC calle	ABC departamento
1	1	50	07/18	28-49367281-6	2454	Larralde	A
on to filter results (use Ctrl+Space)							
	ABC piso	123 prom_ventas_anuales	ABC razon_social	123 actividad	123 categoria	123 ciudad	
3		18.050	Razon Social	1	1	1	

- Sector

ent_sector			
select * from ent_sector es Enter a SQL expression to filter resu			
	123 sector	ABC descripcion	ABC nombre
1	1	Descripción del Sector 1	Nombre 1

- Categoría

ent_categorias				
select * from ent_categorias ec Enter a SQL expression to filter results (use Ctrl+Space)				
	123 categoria	ABC nombre	123 personal_maximo	123 ventas_anuales
1	1	Nombre 1	150	20.890

- Criterios

dom_criterios						
select * from dom_criterios dc Enter a SQL expression to filter results (use Ctrl+Space)						
	123 criterio	ABC descripcion	ABC nombre	123 criterio_padre	123 entidad	123 presupuesto
1	1	Descripción del Criterio 1	Nombre 1	[NULL]	1	1

- Categoría

dom_categorias				
select * from dom_categorias dc Enter a SQL expression to filter results (use Ctrl+Space)				
	123 categoria	ABC descripcion	ABC nombre	123 criterio
1	1	Descripción de la Categoría 1	Nombre 1	1

- CriteriosXEgreso

dom_egresos_dom_criterios		
select * from dom_egresos_dom_criterios ded Enter a SQL		
	123 dom_egresos_egreso	123 criterios_criterio
1	1	1

- Items

dom_items				
select * from dom_items di Enter a SQL expression to filter results (use Ctrl+Space)				
	123 item	ABC descripcion	ABC tipo	123 valor
1	1	Descripción Item 1	Producto	4.575
2	2	Descripción Item 2	Producto	14.500

- RevisorXEgreso

pers_usuarios_dom_egresos		
select * from pers_usuarios_dom_egresos pudi Enter a SQL		
	123 pers_usuarios_usuario	123 egresos_egreso
1	1	1

Segunda Sección

Para comenzar, detectamos las siguientes entidades fundamentales:

- SolicitarVinculacion: es la entidad encargada de generar la vinculación de egresos e ingresos.
- Importe: representa lo que nos interesa de un ingreso o egreso a vincular.
- Criterio: determina cómo se realizará la vinculación.
- Condicion: establece ciertos estándares para poder realizar la vinculación.
- Vinculacion: genera el JSON con el resultado de la vinculación.

Inicialmente, habíamos modelado una clase Egreso y otra Ingreso pero nos dimos cuenta que era innecesario porque ambas compartían la mayoría de sus atributos. Por esta razón, modelamos la clase **Importe** debido al hecho de que sólo nos interesa la fecha, el importe y la fecha de un ingreso o egreso. En consecuencia, ganamos mantenibilidad ya que reutilizamos código y es muy sencillo localizar la clase **Importe** en caso de tener que cambiar algo.

Por otro lado, la clase **SolicitarVinculacion** posee una lista de condiciones con el objetivo de poder agregar más de una **Condicion** a la hora de querer vincular egresos con ingresos. En consecuencia, nuestra solución es más extensible porque damos la posibilidad de incorporar más de una condición en el futuro. Cabe destacar, que elegimos modelar una interfaz **Condicion** con el fin de añadir nuevas clases que implementen el método *cumple()*. De esta manera, cumplimos con el requerimiento del enunciado que solicitaba poder agregar nuevas condiciones en un futuro, logrando así, una solución más extensible.

Por último, como se nos solicitaba la posibilidad de setear un criterio *Mix*, es decir, una combinación de otros criterios, decidimos agregarle una lista de criterios a la clase **SolicitarVinculacion**. De esta manera, si se escoge un solo **Criterio** la lista solamente tendrá un elemento mientras que si se desea un *Mix* la lista tendrá más de uno. En consecuencia, nos ahorramos el modelar una clase *Mix* logrando así una solución más consistente ya que se evitan posibles condiciones de recursividad entre criterios. Asimismo, modelamos una interfaz **Criterio** ya que ganabamos mayor mantenibilidad porque es fácil de localizar a cada Criterio y extensibilidad debido al hecho de que se pueden agregar más criterios simplemente haciendo que éstos implementen dicha interfaz.

En conclusión, con todo lo mencionado anteriormente nuestra clase **SolicitarVinculacion** posee mayor cohesión ya que se delega en dos interfaces (**Condicion** y **Criterio**) la responsabilidad de establecer ciertos estándares a la hora de realizar una vinculación y de determinar cómo se llevará a cabo dicha vinculación. Análogamente, se delega en la clase **Vinculacion** la responsabilidad de generar el JSON con la vinculación concretada que será enviada al módulo GeSoc, logrando así mayor cohesión en la clase **SolicitarVinculación** debido al hecho de que ésta última no debe preocuparse por la confección de dicho JSON.