

14 DE MAYO, 2023

TRABAJO PRACTICAS

DESAROLLO
DIRIGIDO
POR
MODELOS

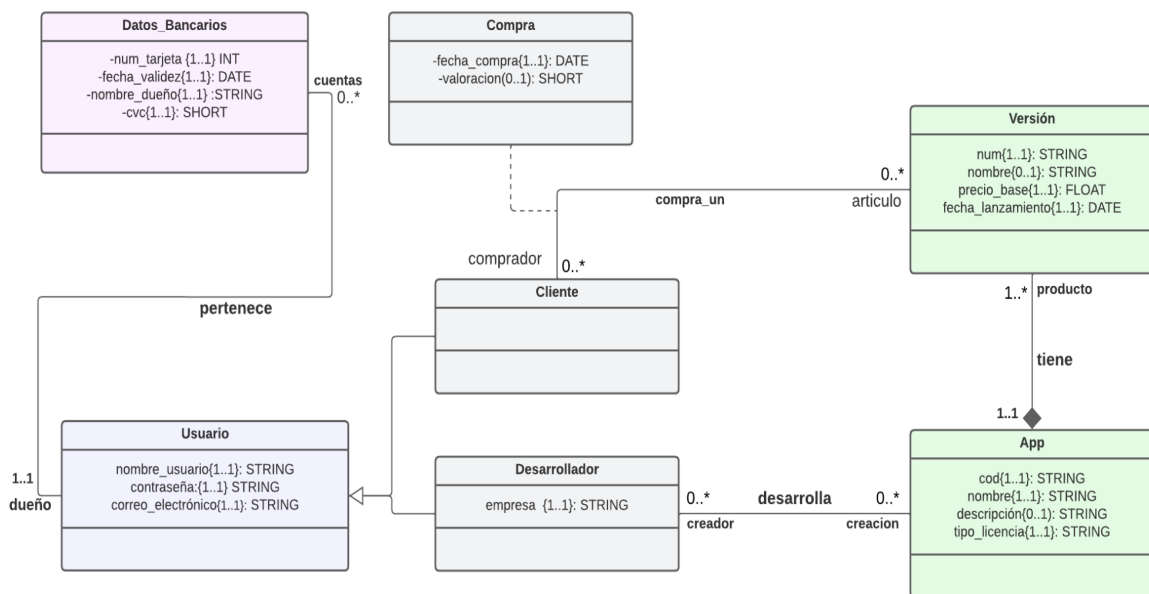
EDWIN MAKOVEEV ROUTSKAIA
IVAN ABAD GARCIA

Resumen

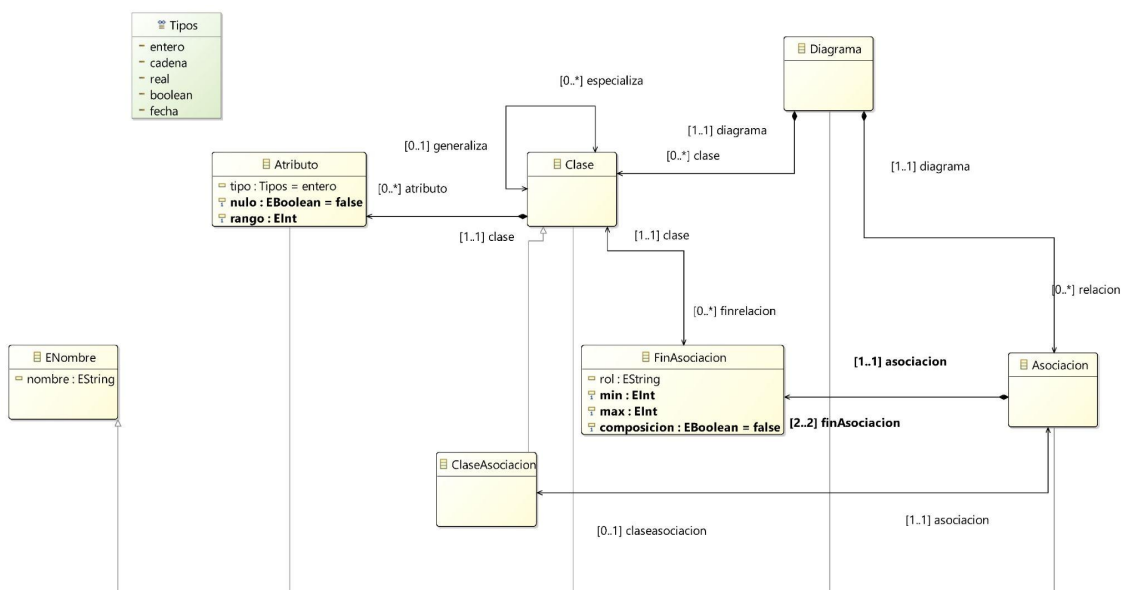
En un inicio como el metamodelo que hicimos en la entrega de la práctica anterior difiere en comparación con el metamodelo del profesor, hemos optado por utilizar el metamodelo proporcionado por el profesor, de modo que estamos utilizando tanto el DC como el ER.

Cabe recalcar, que hemos almacenado todos los pasos de las transformaciones en la carpeta backups, para que así podamos consultarlo si algo difiere. Por otro lado, hemos utilizado el mismo UML, para representar las instancias.

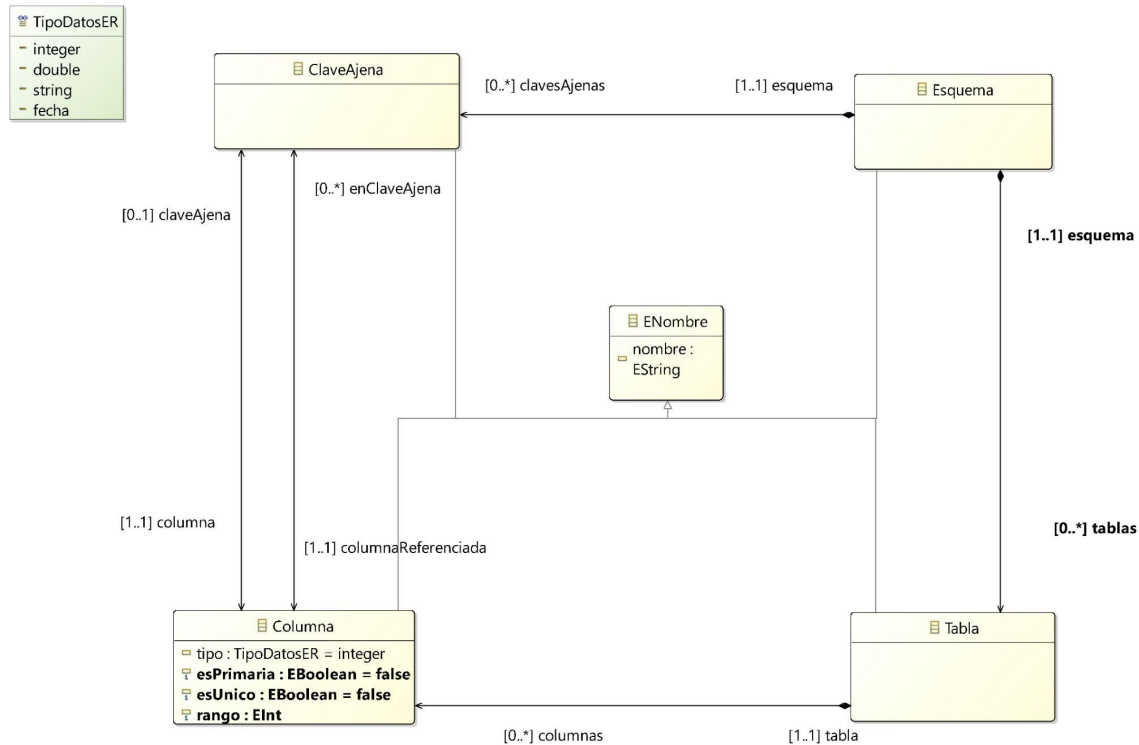
UML:



DC:



ER:



Incrementos Sugeridos

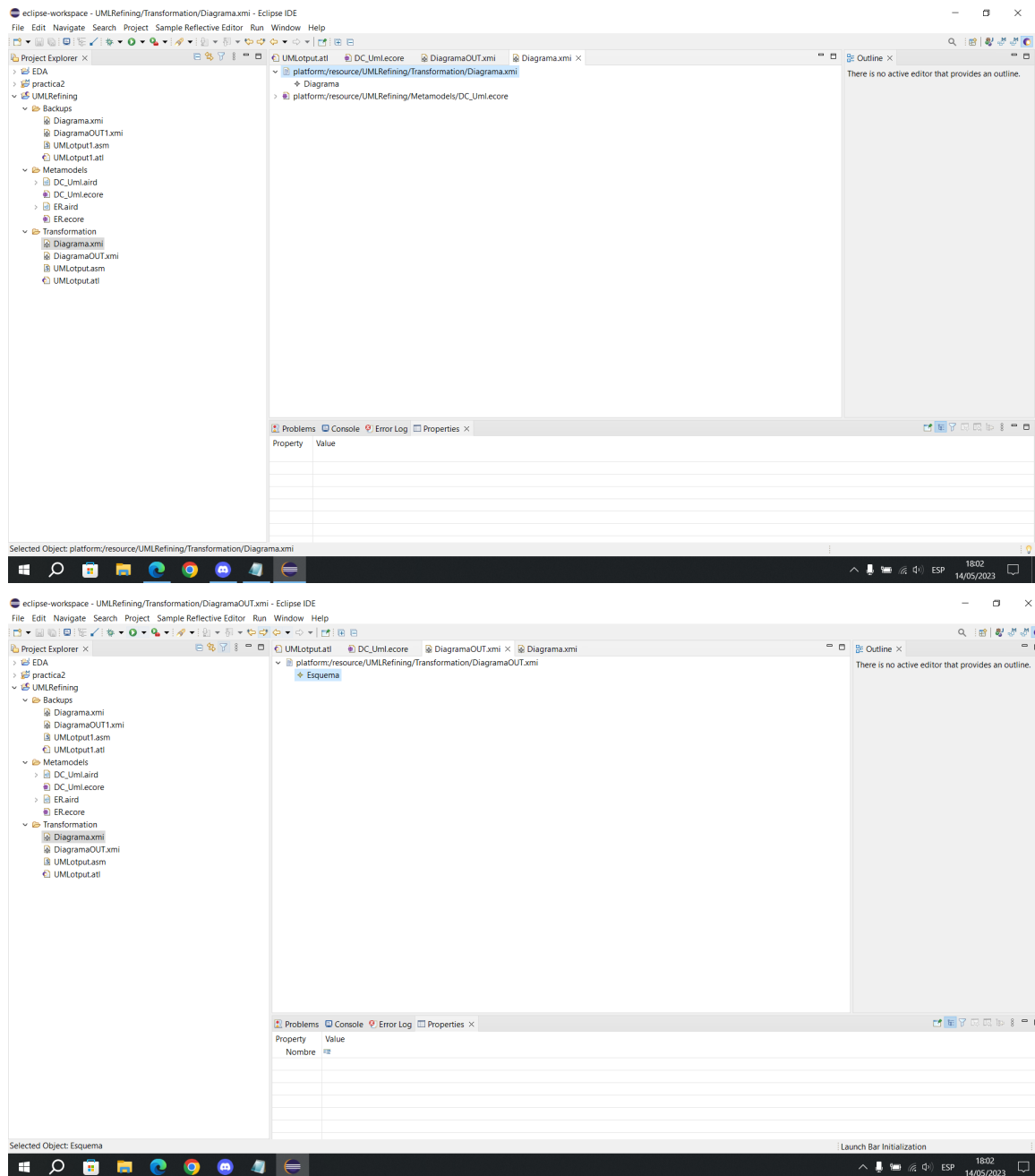
1.Diagrama de Clases → Esquema Relacional

A partir del diagrama de DC, ponemos que el esquema generado tenga el mismo nombre y la clase que el diagrama

```

module UMLotput;
create OUT : ERMM from IN : INMM;

rule diagram2esquema {
  from
    diagrama: INMM!Diagrama
  to
    esquema: ERMM!Esquema (
      nombre <- diagrama.nombre,
      tablas <- diagrama.clase
    )
}
  
```



2.Clases → Tablas (sin columnas)

A partir de la clase de metamodelado DC hacemos que las tablas que esten en el esquema despues de la transformación tengan los mismos nombres y pertenezcan al esquema correspondiente.

```

rule clase2tabla {
  from
    clase: INMM!Clase
  to
    tabla: ERMM!Tabla (
      nombre <- clase.nombre,
      esquema <- tabla.nombre
    )
}

```

- ▼ platform:/resource/UMLRefining/Transformation/Diagrama.xmi
 - ▼ ♦ Diagrama
 - ♦ Clase Usuario
 - ♦ Clase Cliente
 - ♦ Clase Desarrollador
 - ♦ Clase App
 - ♦ Clase Version
 - ♦ Clase Compra
 - ♦ Clase Datos_Bancarios
- > platform:/resource/UMLRefining/Metamodels/DC_Uml.ecore

- ▼ platform:/resource/UMLRefining/Transformation/DiagramaOUT.xmi
 - ▼ ♦ Esquema
 - ♦ Tabla Usuario
 - ♦ Tabla Cliente
 - ♦ Tabla Desarrollador
 - ♦ Tabla App
 - ♦ Tabla Version
 - ♦ Tabla Compra
 - ♦ Tabla Datos_Bancarios

3. Generación de columna Id en cada Tabla

Les añadimos a las tablas anteriores la columna, la cual sirve como identificador de la tabla. Será clave primaria , única, de tipo Integer y no nula.

```
rule clase2tabla {
  from
    clase: INMM!Clase
  to
    tabla: ERMM!Tabla (
      nombre <- clase.nombre,
      esquema <- clase.diagrama,
      columnas <- columna
    ),
    columna: ERMM!Columna (|
      nombre <- 'id_' + tabla.nombre,
      tipo <- #integer,
      esPrimaria <- true,
      esUnico <- true,
      rango <- 1,
      tabla <- tabla
    )
}
```





























- platform:/resource/UMLRefining/Transformation/Diagrama.xmi
 - Diagrama
 - Clase Usuario
 - Clase Cliente
 - Clase Desarrollador
 - Clase App
 - Clase Version
 - Clase Compra
 - Clase Datos_Bancarios
- platform:/resource/UMLRefining/Metamodels/DC_Uml.ecore

- ▼ platform:/resource/UMLRefining/Transformation/DiagramaOUT.xmi
 - ▼ Esquema
 - ▼ Tabla Usuario
 - ◆ Columna id_Usuario
 - ▼ Tabla Cliente
 - ◆ Columna id_Cliente
 - ▼ Tabla Desarrollador
 - ◆ Columna id_Desarrollador
 - ▼ Tabla App
 - ◆ Columna id_App
 - ▼ Tabla Version
 - ◆ Columna id_Version
 - ▼ Tabla Compra
 - ◆ Columna id_Compra
 - ▼ Tabla Datos_Bancarios
 - ◆ Columna id_Datos_Bancarios

4. Generación de Columnas en cada Tabla

Ahora le tenemos que añadir las variables/atributos que tengan las clases a las columnas de sus respectivas tablas que se han transformado, de manera que las variables tendran el mismo nombre, no seran primarias ni unicas, teniendo el mismo rango, eso si aun no se toma en consideración si el atributo es una cadena, un entero, etc...

```
rule atributos2tabla {
  from
    atributo : INMM!Atributo
  to
    columna : ERMM!Columna (
      nombre <- atributo.nombre,
      esPrimaria <- false,
      esUnico <- false,
      rango <- atributo.rango,
      tabla <- atributo.clase
    )
}
```

- ▼  platform:/resource/UMLRefining/Transformation/Diagrama.xml
 - ▼  Diagrama
 - ▼  Clase Usuario
 -  Atributo contraseña
 -  Atributo nombre_usuario
 -  Atributo correo_electronico
 -  Clase Cliente
 - ▼  Clase Desarrollador
 -  Atributo empresa
 - ▼  Clase App
 -  Atributo cod
 -  Atributo nombre
 -  Atributo descripcion
 -  Atributo tipo_licencia
 - ▼  Clase Version
 -  Atributo num
 -  Atributo nombre
 -  Atributo precio_base
 -  Atributo fecha_lanzamiento
 - ▼  Clase Compra
 -  Atributo fecha_compra
 -  Atributo valoracion
 - ▼  Clase Datos_Bancarios
 -  Atributo num_tarjeta
 -  Atributo fecha_validez
 -  Atributo nombre_dueño
 -  Atributo cvc
- >  platform:/resource/UMLRefining/Metamodels/DC_Uml.ecore

- ▼ ◆ Esquema
 - ▼ ◆ Tabla Usuario
 - ◆ Columna id_Usuario
 - ◆ Columna contraseña
 - ◆ Columna nombre_usuario
 - ◆ Columna correo_electronico
 - ▼ ◆ Tabla Cliente
 - ◆ Columna id_Cliente
 - ▼ ◆ Tabla Desarrollador
 - ◆ Columna id_Desarrollador
 - ◆ Columna empresa
 - ▼ ◆ Tabla App
 - ◆ Columna id_App
 - ◆ Columna cod
 - ◆ Columna nombre
 - ◆ Columna descripcion
 - ◆ Columna tipo_licencia
 - ▼ ◆ Tabla Version
 - ◆ Columna id_Version
 - ◆ Columna num
 - ◆ Columna nombre
 - ◆ Columna precio_base
 - ◆ Columna fecha_lanzamiento
 - ▼ ◆ Tabla Compra
 - ◆ Columna id_Compra
 - ◆ Columna fecha_compra
 - ◆ Columna valoracion
 - ▼ ◆ Tabla Datos_Bancarios
 - ◆ Columna id_Datos_Bancarios
 - ◆ Columna num_tarjeta
 - ◆ Columna fecha_validez
 - ◆ Columna nombre_dueño
 - ◆ Columna cvc

5. Generación de datos de columnas (Tipo y valor nulo no permitido)

Generamos un convertidor, para que el tipo de atributo se convierta en tipos de datos aceptables para nuestra transformación, que son entero -> integer, cadena -> string, real -> double, date -> fecha. Cabe recalcar, que los atributos boolean existen por defecto, así que no es necesario un convertidor para ellos, para que no haya valores no nulos el rango de la columna ha de ser mínimo 1.

```














> helper context INMM!Atributo def: convierteTipo() : Relational!TipoDatosER =
  if self.tipo = #entero then #integer else if self.tipo = #cadena then #string
  else if self.tipo = #real then #double else #fecha endif endif endif;








```

```
rule atributos2tabla {  
  from  
    atributo : INMM!Atributo  
  to  
    columna : ERMM!Columna (  
      nombre <- atributo.nombre,  
      tipo <- atributo.convierteTipo(),  
      esPrimaria <- false,  
      esUnico <- false,  
      rango <- atributo.rango,  
      tabla <- atributo.clase  
    )  
}
```

- ✦ Clase Usuario
 - ✦ Atributo contraseña
 - ✦ Atributo nombre_usuario
 - ✦ Atributo correo_electronico
 - ✦ Clase Cliente
 - ✦ Clase Desarrollador
 - ✦ Atributo empresa
 - ✦ Clase App
 - ✦ Atributo cod
 - ✦ Atributo nombre
 - ✦ Atributo descripcion
 - ✦ Atributo tipo_licencia
 - ✦ Clase Version
 - ✦ Atributo num
 - ✦ Atributo nombre
 - ✦ Atributo precio_base
 - ✦ Atributo fecha_lanzamiento
 - ✦ Clase Compra
 - ✦ Atributo fecha_compra
 - ✦ Atributo valoracion
 - ✦ Clase Datos_Bancarios
 - ✦ Atributo num_tarjeta
 - ✦ Atributo fecha_validez
 - ✦ Atributo nombre_dueño
 - ✦ Atributo cvc
- > platform:/resource/UMLRefining/Metamodels/DC_Uml.ecore

Problems Console Error Log Properties ×	
Property	Value
Clase	Clase Usuario
Nombre	correo_electronico
Nulo	false
Rango	1
Tipo	cadena

- ▼  platform:/resource/UMLRefining/Transformation/DiagramaOUT.xmi
 - ▼  Esquema
 - ▼  Tabla Usuario
 -  Columna id_Usuario
 -  Columna contraseña
 -  Columna nombre_usuario
 -  Columna correo_electronico
 - >  Tabla Cliente
 - >  Tabla Desarrollador
 - >  Tabla App
 - >  Tabla Version
 - >  Tabla Compra
 - >  Tabla Datos_Bancarios

Problems Console Error Log Properties ×	
Property	Value
En Clave Ajena	
Es Primaria	 false
Es Unico	 false
Nombre	 correo_electronico
Rango	 1
Tabla	 Tabla Usuario
Tipo	 string

6. Generación de FKs correspondientes a Asociaciones con multiplicidad > 1

En la siguiente transformación creamos las claves ajenas de aquellas Asociaciones cuya multiplicidad sea mayor a 1.

```
rule asociacion2ClaveAjena {  
  from  
    asociacion: INMM!Asociacion (asociacion.finAsociacion.first().max.intValue() = 1 and  
                                asociacion.claseasociacion.oclIsUndefined())  
  to  
    claveAjena: ERM!ClaveAjena (  
      nombre <- 'FK_' + asociacion.finAsociacion.last().rol + '_' + asociacion.finAsociacion.first().rol,  
      columna <- claveAjenaColumna,  
      columnaReferenciada <- thisModule.resolveTemp(asociacion.finAsociacion.first().clase, 'columna')  
    ),  
    claveAjenaColumna: ERM!Columna (  
      nombre <- 'id_FK_' + asociacion.finAsociacion.first().rol,  
      tabla <- asociacion.finAsociacion.last().clase  
    )  
}
```

platform:/resource/UMLRefining/Transformation/Diagrama.xml

Diagrama

> Clase Usuario

> Clase Cliente

> Clase Desarrollador

> Clase App

> Clase Version

> Clase Compra

> Clase Datos_Bancarios

> Asociacion pertenece

> Fin Asociacion dueño

> Fin Asociacion cuentas

> platform:/resource/UMLRefining/Metamodels/DC_Uml.ecore

platform:/resource/UMLRefining/Transformation/DiagramaOUT.xml

Esquema

> Tabla Usuario

> Tabla Cliente

> Tabla Desarrollador

> Tabla App

> Tabla Version

> Tabla Compra

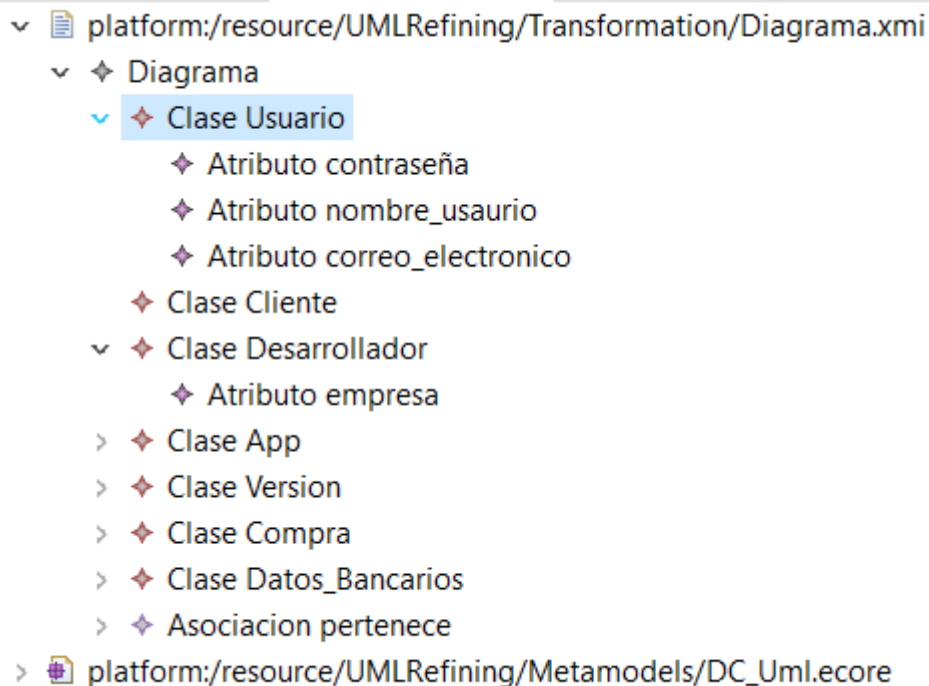
> Tabla Datos_Bancarios

> Clave Ajena FK_cuentas_dueño

7. Generación de FKs en tablas correspondientes a Clases Hijas

(!)En el caso de la séptima transformación, no nos funciona debidamente ya que no vemos que se generen 'Claves Ajenas' (FK) en las clases que heredan de usuario, que son cliente y desarrollador.

```
rule herencia2tabla {
  from
    clase: INMM!Clase (not clase.generaliza.oclIsUndefined())
  to
    tabla: ERM!Tabla (
      nombre <- clase.nombre,
      esquema <- clase.diagrama,
      columnas <- columna
    ),
    columna: ERM!Columna (
      nombre <- 'id_' + clase.nombre,
      tipo <- #integer,
      esPrimaria <- true,
      esUnico <- true,
      rango <- 1,
      enClaveAjena <- claveAjena
    ),
    claveAjena: ERM!ClaveAjena (
      nombre <- 'FK_' + clase.nombre + '_' + clase.generaliza.nombre,
      columna <- claveAjenaColumna,
      columnaReferenciada <- thisModule.resolveTemp(clase.generaliza, 'columnas')->any(c|c.nombre = 'id_' + clase.generaliza.nombre),
      esquema <- tabla.esquema
    ),
    claveAjenaColumna: ERM!Columna (
      nombre <- 'id_FK_' + clase.generaliza.nombre,
      tabla <- tabla
    )
}
```





- ▼ platform:/resource/UMLRefining/Transformation/DiagramaOUT.xml
 - ▼ Esquema
 - ▼ Tabla Usuario
 - ◆ Columna id_Usuario
 - ◆ Columna contraseña
 - ◆ Columna nombre_usuario
 - ◆ Columna correo_electronico
 - ▼ Tabla Cliente
 - ◆ Columna id_Cliente
 - ▼ Tabla Desarrollador
 - ◆ Columna id_Desarrollador
 - ◆ Columna empresa
 - > Tabla App
 - > Tabla Version
 - > Tabla Compra
 - > Tabla Datos_Bancarios
 - ◆ Clave Ajena FK_cuentas_dueño













8. Generación de FKs en tablas correspondientes a Clases Asociación


(!)En el caso de la octava transformación, no nos funciona debidamente ya que no vemos que se generen 'Claves Ajenas' (FK) en las clases asociación.


```
rule clavesAjenas2ClasesAsociacion {
  from
    asociacion: INMM!Asociacion (
      not asociacion.claseasociacion.oclIsUndefined()
    )
  to
    claveAjenaFirst: ERMM!ClaveAjena (
      nombre <- 'FK_' + asociacion.claseasociacion.nombre + '_' + asociacion.finAsociacion.first().rol,
      columnaReferenciada <- thisModule.resolveTemp(asociacion.finAsociacion.first().clase, 'columna'),
      columna <- columnaClaveAjenaFirst
    ),
    claveAjenaLast: ERMM!ClaveAjena (
      nombre <- 'FK_' + asociacion.claseasociacion.nombre + '_' + asociacion.finAsociacion.last().rol,
      columnaReferenciada <- thisModule.resolveTemp(asociacion.finAsociacion.last().clase, 'columna'),
      columna <- columnaClaveAjenaLast,
      esquema <- claveAjenaFirst.esquema
    ),
    columnaClaveAjenaFirst: ERMM!Columna (
      nombre <- 'id_FK_' + asociacion.finAsociacion.first().rol,
      tabla <- asociacion.claseasociacion
    ),
    columnaClaveAjenaLast: ERMM!Columna (
      nombre <- 'id_FK_' + asociacion.finAsociacion.last().rol,
      tabla <- asociacion.claseasociacion
    )
}
```


▼  platform:/resource/UMLRefining/Transformation/Diagrama.xmi













▼  Diagrama

- >  Clase Usuario
 -  Clase Cliente
- >  Clase Desarrollador
- >  Clase App
- >  Clase Version
- >  Clase Compra
- >  Clase Datos_Bancarios
 -  Clase Asociacion Compras
- >  Asociacion pertenece
- ▼  Asociacion compra_un
 -  Fin Asociacion producto
 -  Fin Asociacion comprador

>  platform:/resource/UMLRefining/Metamodels/DC_Uml.ecore

▼  platform:/resource/UMLRefining/Transformation/DiagramaOUT.xmi

▼  Esquema

- >  Tabla Usuario
- ▼  Tabla Cliente
 -  Columna id_Cliente
- >  Tabla Desarrollador
- >  Tabla App
- >  Tabla Version
- ▼  Tabla Compra
 -  Columna id_Compra
 -  Columna fecha_compra
 -  Columna valoracion
- >  Tabla Datos_Bancarios
-  Clave Ajena FK_cuentas_dueño