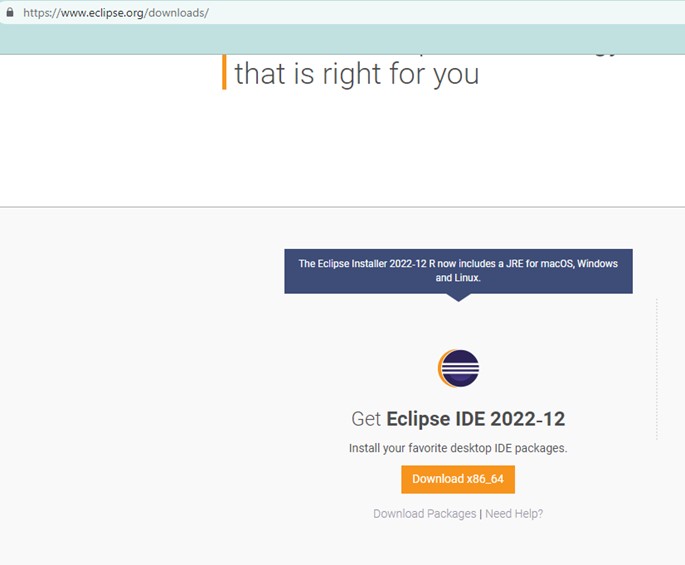
Actividad 2: Ingeniería de software dirigida por modelos

Recuerda que esta actividad deberá ser completada en la plataforma. Solo tendrás un intento. Una vez que la envíes, se dará por cerrada. Tu puntuación y las respuestas correctas se mostrarán cuando finalice el periodo de entrega de la tarea.

**Descripción de la actividad**

En esta actividad, vamos a profundizar en la ingeniería dirigida por modelos, creando una serie de lenguajes específicos de dominio, los cuales definiremos de forma gráfica mediante metamodelos y haciendo uso de la herramienta Eclipse Modeling Framework (EMF).

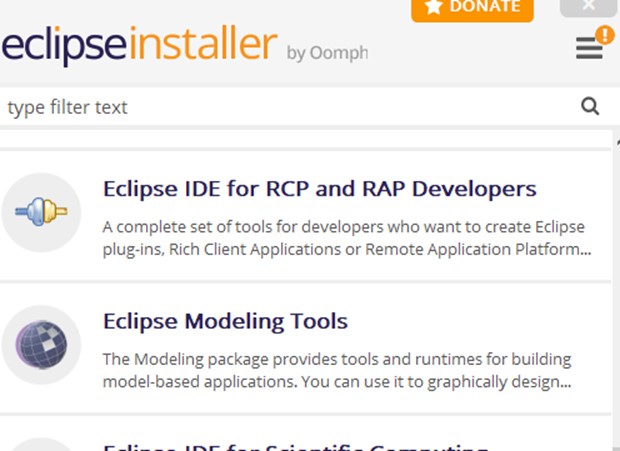
Por ello, lo primero será instalar EMF, para hacerlo iremos a [Eclipse Foundation](https://www.eclipse.org/downloads/) y descargaremos el Eclipse Installer en nuestro equipo.



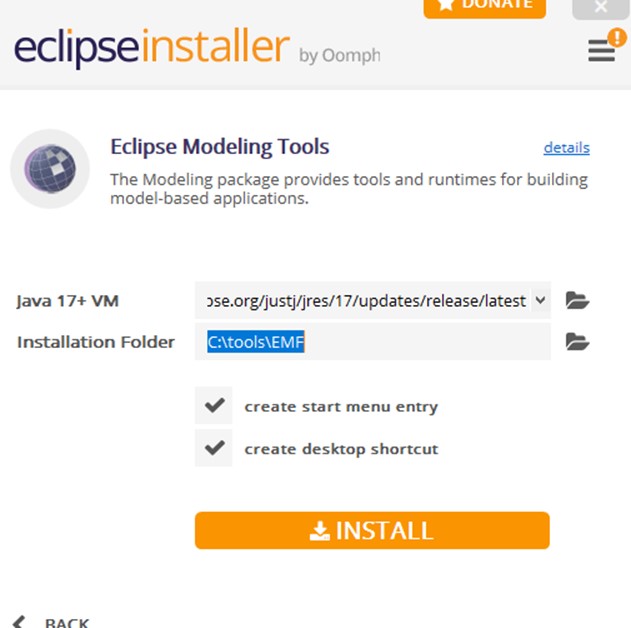
Iniciamos el instalador:



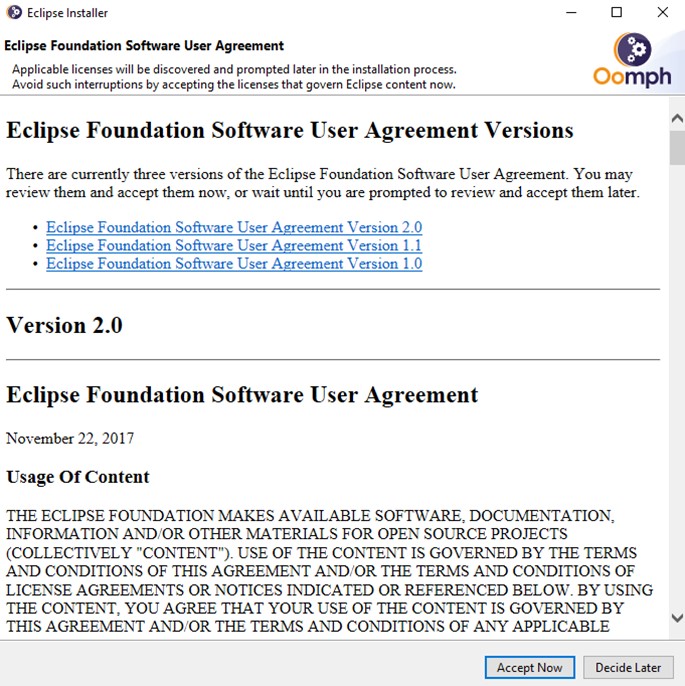
Bajaremos hasta seleccionar Eclipse Modeling Tools:



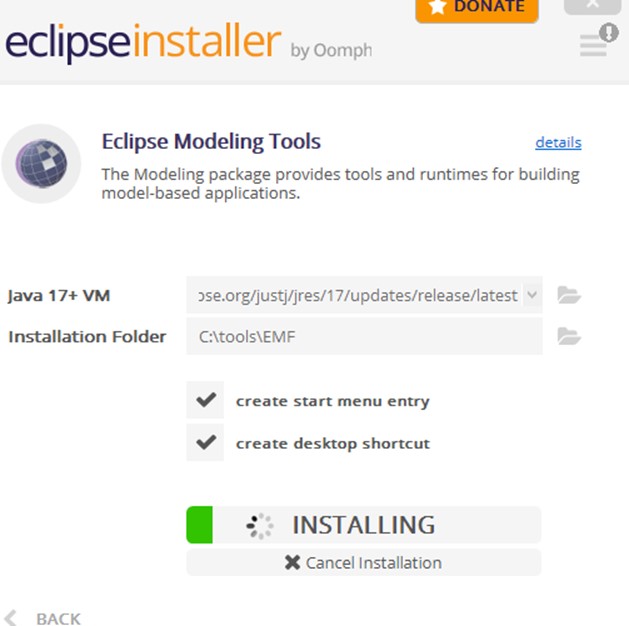
Seleccionaremos una carpeta de destino (se recomienda no ubicarlo en rutas que tengan espacios en blanco, como Archivos de Programa, mejor en C:\tools o en C:\tools\emf).



Aceptaremos los acuerdos de licencia:



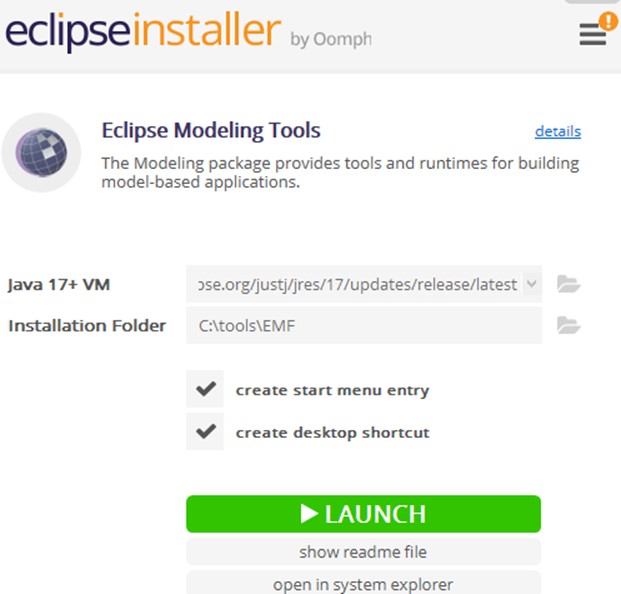
El instalador comenzará:

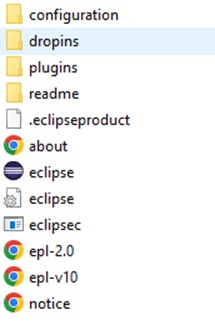


En función de la conectividad, es posible que muestre mensajes de advertencia o incluso que sea necesario repetir el proceso de instalación:



Una vez que ha finalizado la instalación, podemos lanzar el EFM desde el propio instalador, bien ejecutando el «eclipse.exe» que tendremos en la carpeta destino.



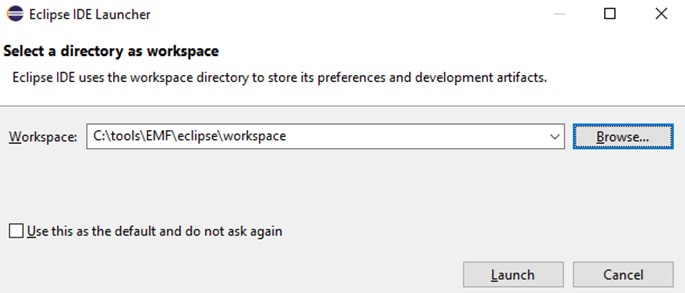


Como lo primero que nos va a pedir es un espacio de trabajo donde almacenar los proyectos, es recomendable tener una carpeta *workspace* creada.

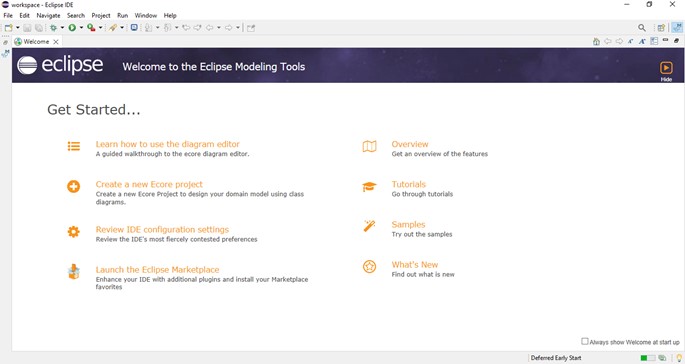
* Inicializamos eclipse:



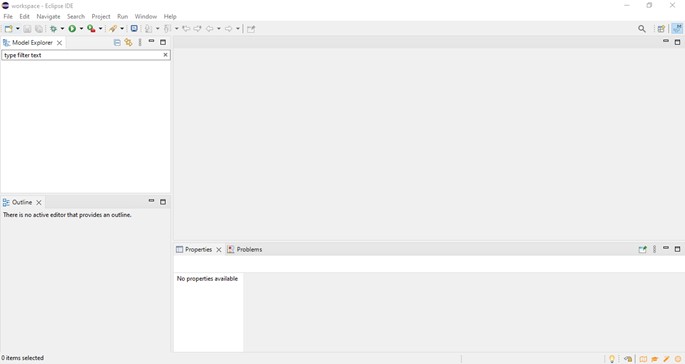
* Seleccionamos una carpeta como espacio de trabajo:



* Tras unos segundos, tendremos el entorno disponible:

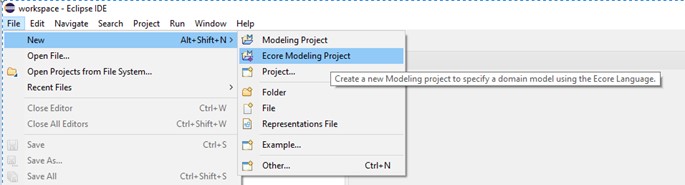


* Pulsaremos arriba a la derecha para ocultar (Hide) y tendremos la vista vacía:

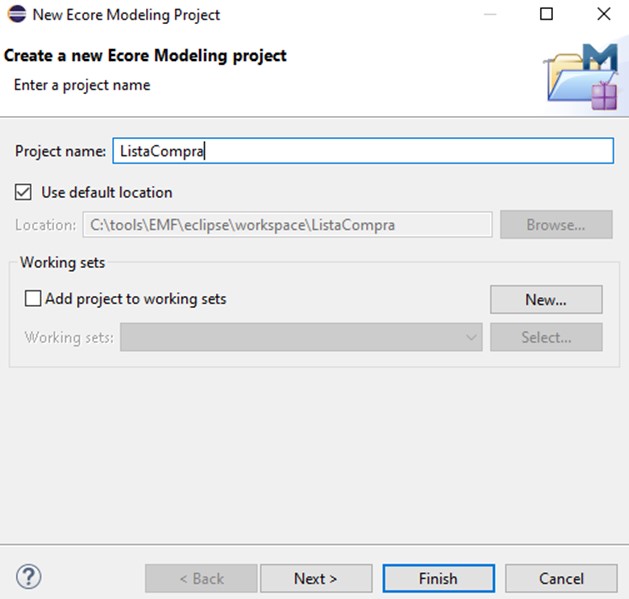


1. Vamos a empezar a familiarizarnos con el entorno EMF.

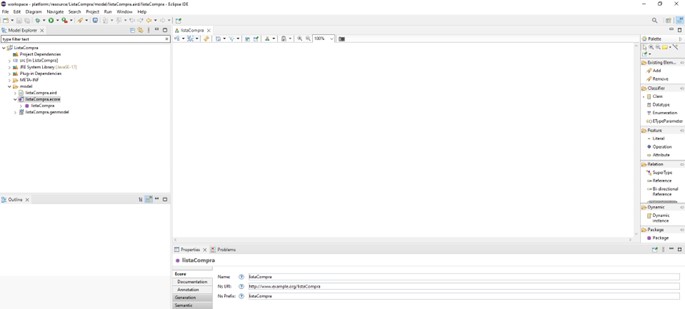
Crea tu primer EMF Modeling Project:



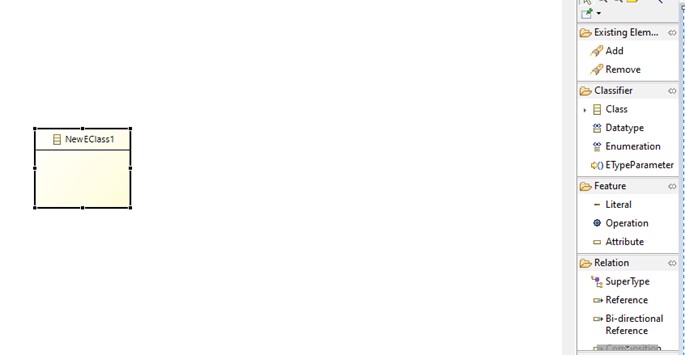
Ponle como nombre «ListaCompra» y pulsa Finish:



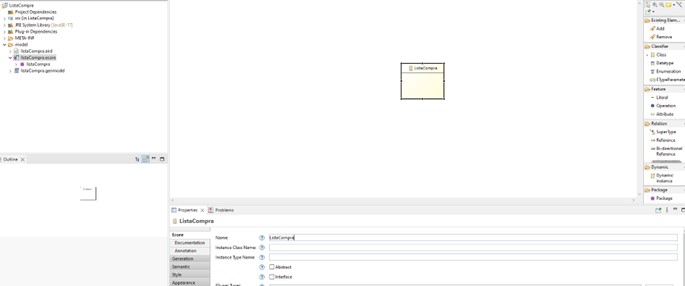
Veremos que se ha creado el proyecto y tenemos esta visualización para poder crear metamodelos en base a ECore, la cual es una especificación similar a [Essential MoF:](https://www.eclipse.org/modeling/emft/search/concepts/EMOF.html)



En el menú de la derecha, haz clic sobre Class y pulsa en el canvas blanco del medio para crear una nueva clase:



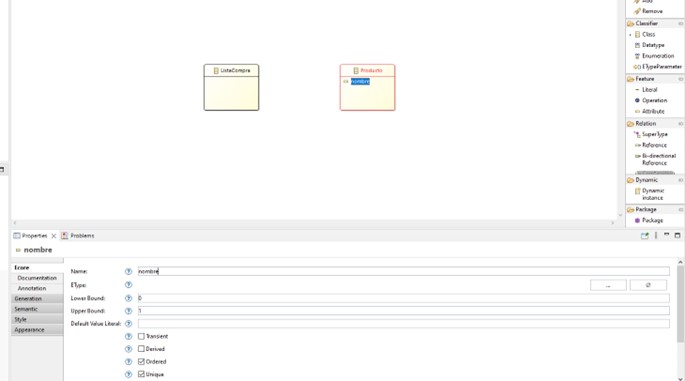
Denomínala «ListaCompra»:



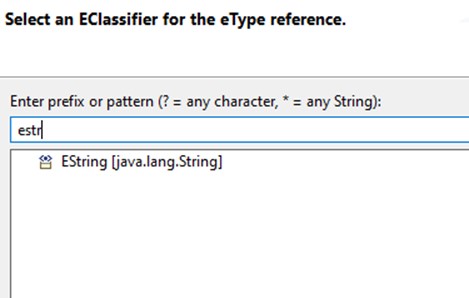
Haz lo mismo con otra que se llame «Producto»:



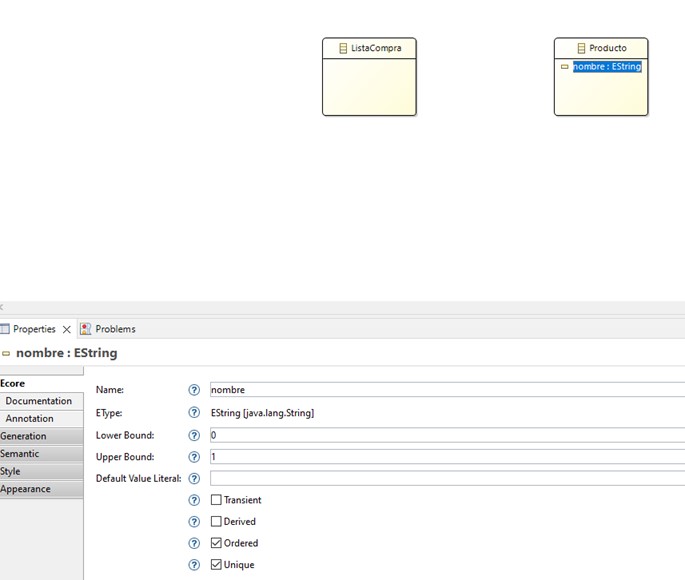
Pulsa en la derecha en Attribute y añade un nombre y una cantidad a «Producto»:



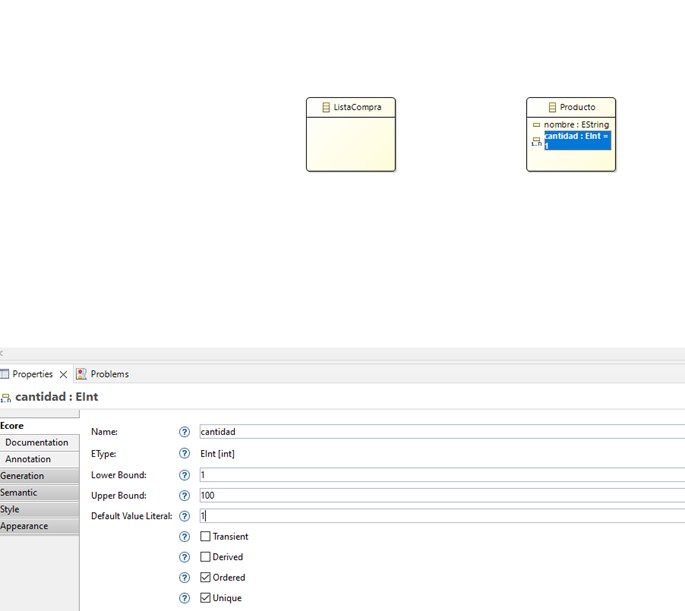
En la paleta de propiedades, seleccionaremos el EType como un *string* (EString):



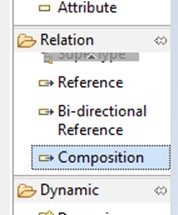
Tras añadir el nombre, nos quedará así:



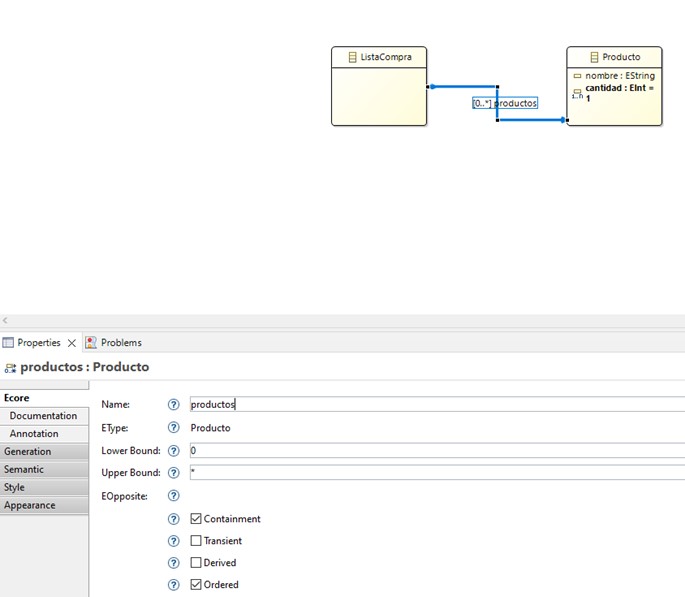
Añadiremos la cantidad:



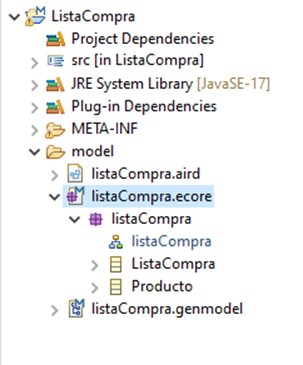
Ya solo nos falta relacionar la lista con el producto. Para esto, seleccionaremos Composition, que está bajo Relation, pulsaremos primero en «ListaCompra» y posteriormente en «Producto»:



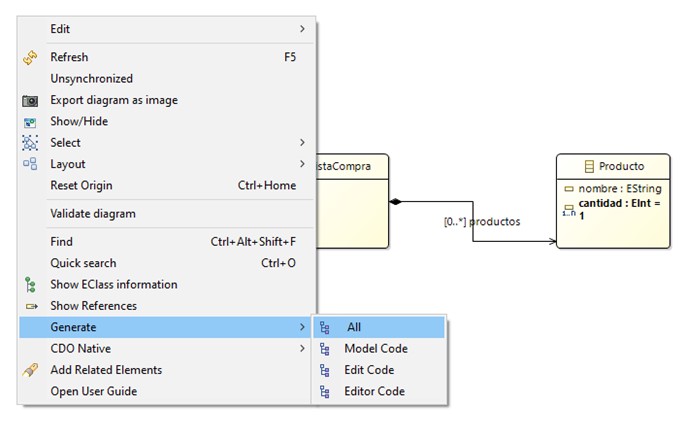
Y renombramos la relación a productos:



Ya tenemos nuestro primer metamodelo ECore desarrollado en el EMF, el «listaCompra.ecore»:



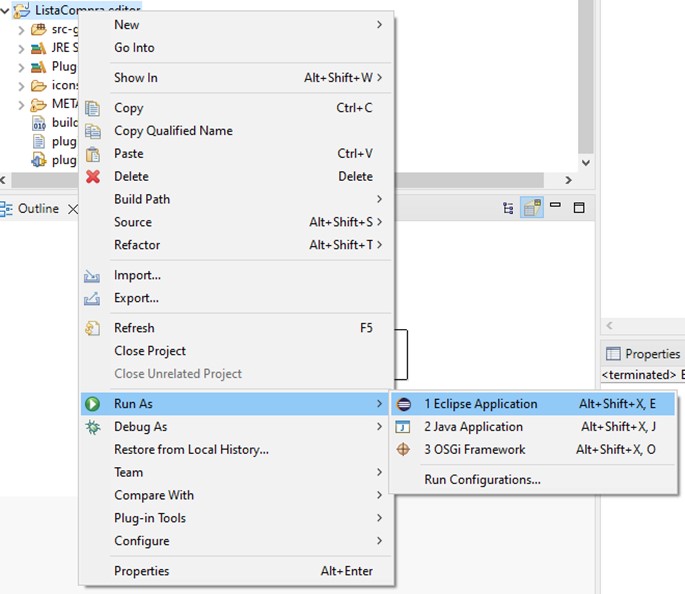
Sobre el canvas, con el botón derecho del ratón, pulsa Generate y luego All:



¿Qué es lo que ha ocurrido?

1. Nada.
2. Muestra en consola un mensaje con un error.
3. Ha creado código en mi proyecto en la carpeta src.
4. Ha creado código en mi proyecto en la carpeta src y, además, ha creado otros dos proyectos nuevos.
5. Ya has creado tu primer metamodelo para simplemente guardar la lista de la compra. Vamos a dar un paso más para crear nuestro primer modelo que es conforme al metamodelo anterior. Para ello, vamos a utilizar el proyecto «.editor» que en el paso anterior el EMF generó.

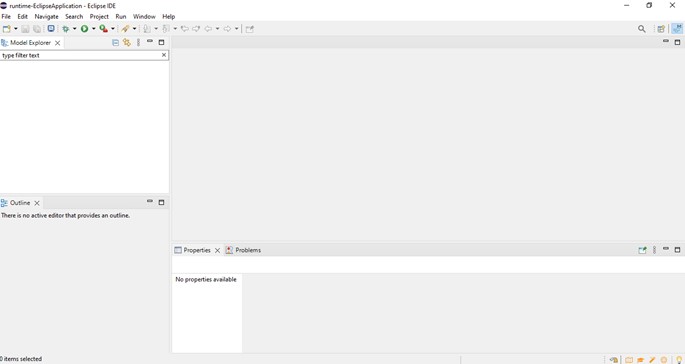
Pulsa sobre el proyecto «.editor» con el botón derecho del ratón, en el menú contextual, selecciona Run AS y Eclipse Application:



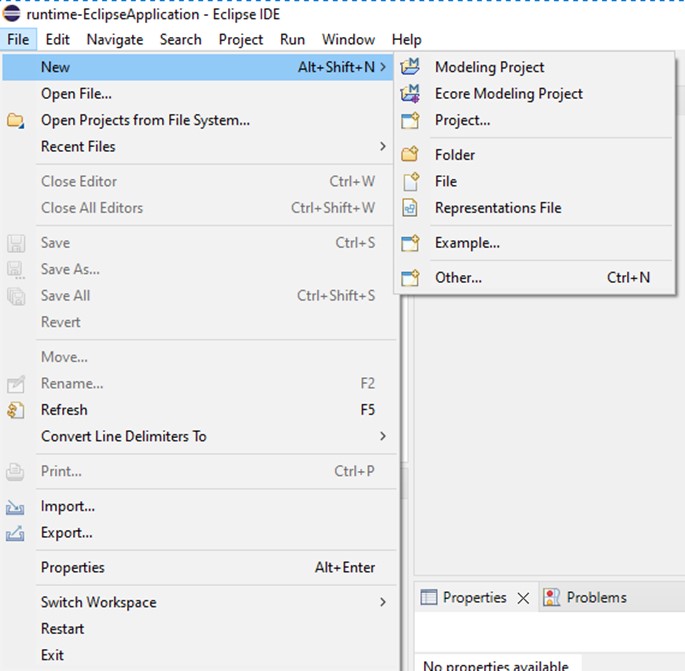
Al ser un *plugin* de eclipse, podemos ejecutarlo dentro del propio eclipse. Para no tener que generar una versión e instalarla, podemos, desde una instancia de eclipse (en la que estábamos editando el metamodelo), lanzar otra segunda instancia que ejecutará ese *plugin.* Saldrá de nuevo que está arrancado el IDE, esta vez no pedirá el *workspace*, sino que guardará todo por defecto bajo una carpeta «runtime-EclipseApplication» (por ejemplo C:\tools\EMF\eclipse\runtime-EclipseApplication si lo instalaste en C:\tools\EMF).



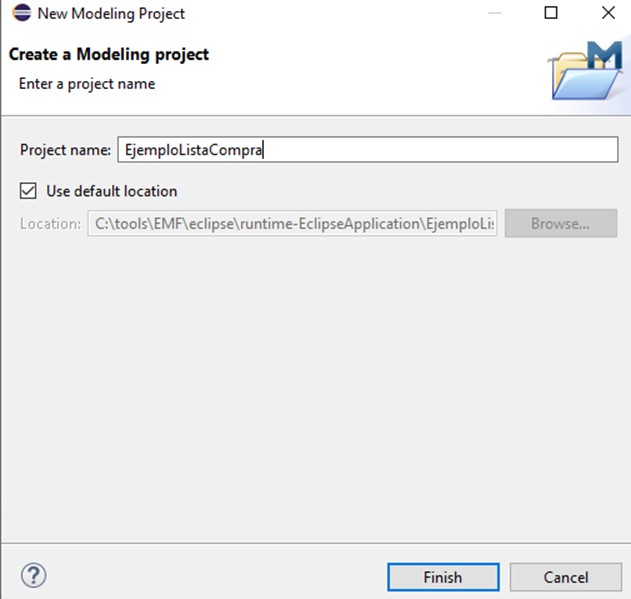
Al ser la primera ejecución, estará vacío, aunque en el título de la aplicación vemos que pone «runtime-EclipseApplication»:



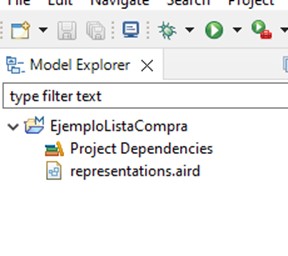
Haremos un New > Modeling Project:



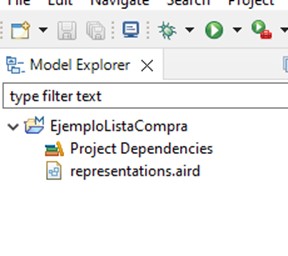
Le pondremos un nombre, por ejemplo, «EjemploListaCompra» y pulsaremos Finish:



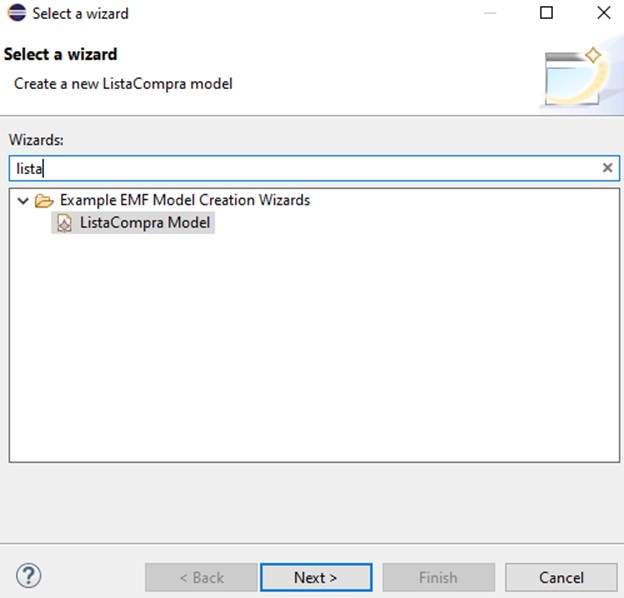
Nos crea un proyecto vacío:



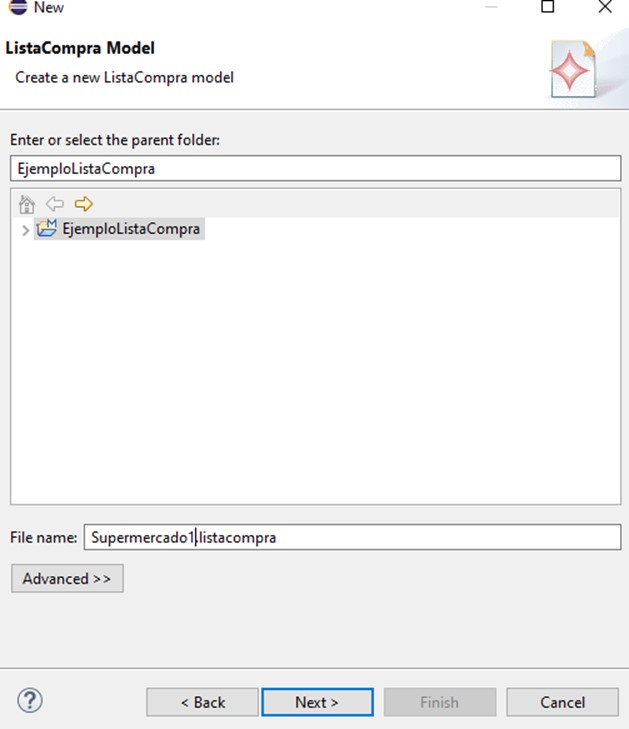
Sobre el proyecto, haremos un New > Other:



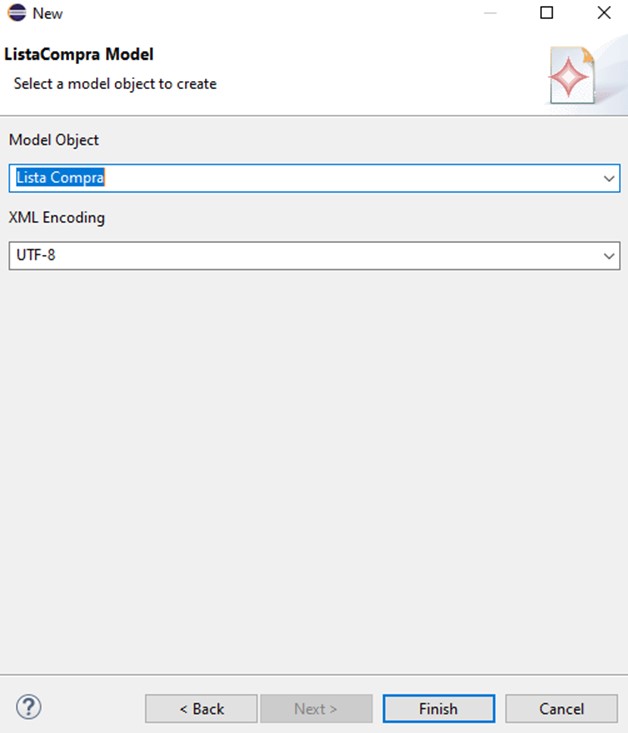
En el buscador pondremos lista y seleccionaremos el Wizard siguiente:



Le pondremos un nombre, por ejemplo, «Supermercado1»:

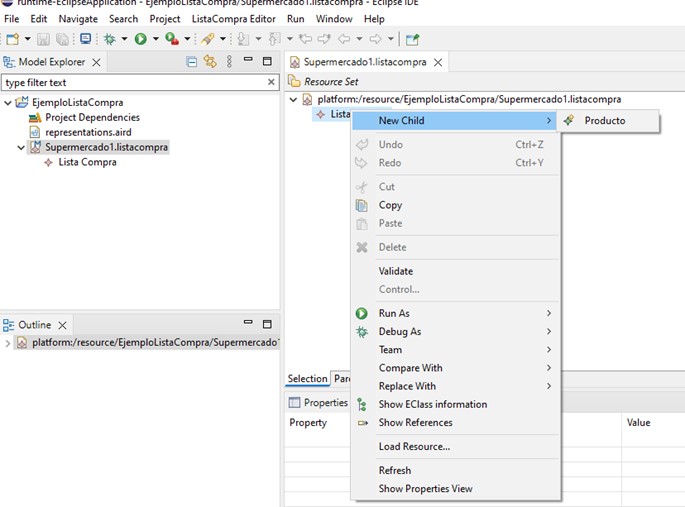


Le indicaremos el elemento raíz del modelo, en nuestro caso, la lista de la compra:

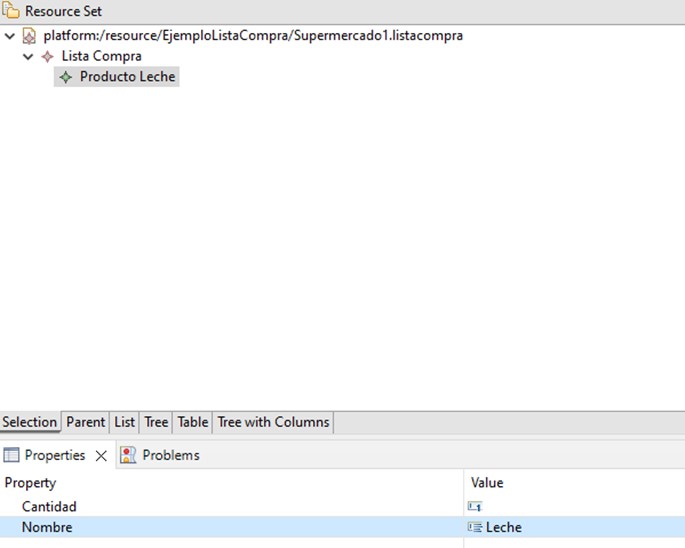


Tras pulsar Finish, ¿qué ha ocurrido?

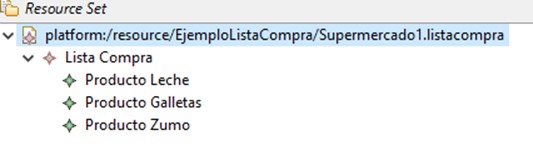
1. Ha generado código sobre el proyecto bajo src.
2. Ha generado el modelo y nos muestra una visualización en modo Selection-Parent-List-Tree-Table-Tree with Columns.
3. Ha generado el modelo y nos muestra una visualización en modo Tree-Table-Tree with Columns.
4. Genera un diagrama visual con un modelo de lista de compra de ejemplo.
5. Ya solo nos queda un paso, incluir datos en el modelo. Dentro de la lista de la compra, con el botón derecho hay que seleccionar New Child y Producto:



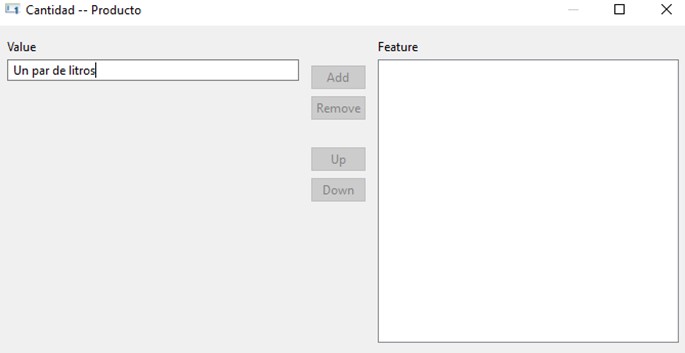
En el editor inferior pondremos el nombre, por ejemplo, «Leche»:



Hacemos este ejercicio varias veces modificando el nombre y la cantidad:



Intenta editar la cantidad de zumo y pon «un par de litros»:

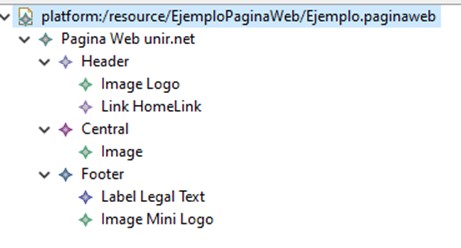


¿Qué ocurre?

1. Nada.
2. Salta una excepción y se cierra eclipse.
3. Muestra un mensaje de «NumberFormatException» y no deja asignar valor.
4. Deja el valor asociado correctamente.
5. En este metamodelo solo podemos tener una lista de la compra y, además, no podemos organizarla por zonas o supermercados. ¿Cómo extenderías el metamodelo para cumplir las siguientes extensiones?

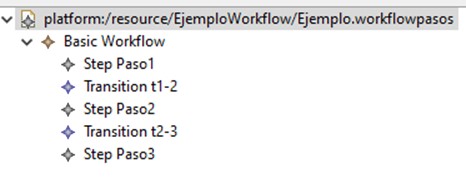
* Una lista de compra tiene sublistas.
* Incluyen una descripción del lugar.
* Cada una tiene productos como hasta ahora.

1. Se debe incluir un nuevo elemento al metamodelo sublista.
2. Se debe incluir un nuevo atributo a la lista y una composición reflexiva.
3. En metamodelos no se pueden anidar elementos de forma recursiva.
4. El modelo existente ya lo soportaba.
5. Vamos a crear el primer metamodelo propio que nos permita definir páginas web simplificadas. Crea el ecore que soporta la definición de una página web con zonas (*header,* central, *footer)* y elementos (*label, buton, image, link)* como para poder generar este modelo:



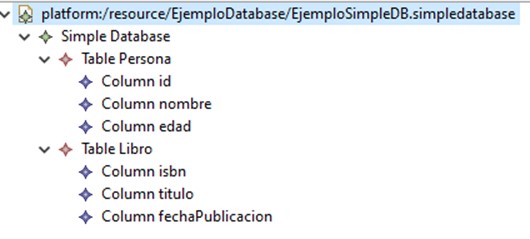
¿Qué contiene tu metamodelo?

1. 10 clases, 2 de ellas abstractas, 7 herencias y 2 composiciones.
2. 8 clases, 0 abstractas, 3 herencias y 2 composiciones.
3. 3 clases y 2 composiciones.
4. No se puede generar el metamodelo para cumplir esa especificación.
5. Crea un nuevo metamodelo para soportar un *workflow* de pasos simple, que permita registrar una secuencia de actividades secuencial unida por transiciones para soportar un modelo de este estilo:



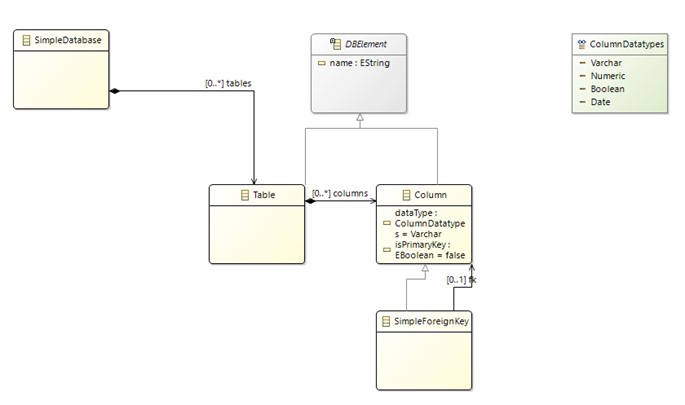
¿Qué contiene tu metamodelo?

1. 3 clases y 1 composición.
2. 4 clases, 1 abstracta, 1 composición, 2 herencias y 2 relaciones.
3. 4 clases, 0 abstractas y 3 composiciones.
4. No se puede generar el metamodelo para cumplir esa especificación.
5. Queremos generar un metamodelo de bases de datos. Vamos a comenzar definiendo las tablas y sus columnas, de momento sin relaciones, para poder cubrir un modelo de este tipo:

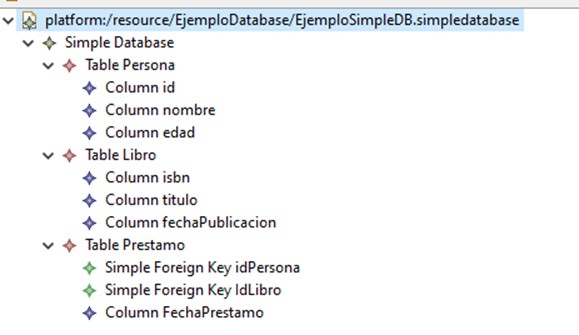


¿Qué contiene tu metamodelo?

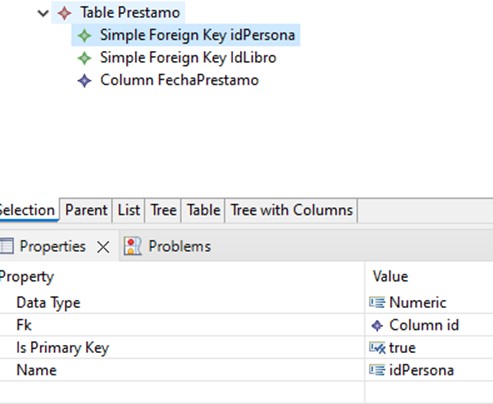
1. 2 clases y 1 composición.
2. 4 clases, 0 abstractas y 3 composiciones.
3. No se puede generar el metamodelo para cumplir esa especificación.
4. 4 clases, 1 abstracta, 2 composiciones y un tipo enumerado.
5. Extendemos el metamodelo de esta forma:

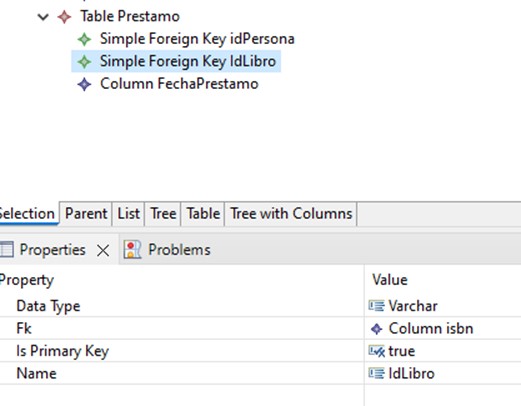


¿Es posible representar un modelo así?



Donde en el préstamo tenemos:





1. No, ya que Foreign Key tendría que ser otra entidad separada de la columna.
2. No, ya que no puede ser Foreign Key y clave primaria a la vez.
3. Sí, el metamodelo lo soporta.
4. No, porque, aunque el metamodelo lo soportara, el modelo sería erróneo.
5. ¿Cómo extenderías el metamodelo para incluir la restricción de valor único para una columna (de momento simplemente asociado a un campo)?
6. Hay que añadir un nuevo atributo *boolean* en Tabla.
7. Hay que añadir un nuevo elemento como ForeignKey que herede de Columna.
8. Hay que añadir un nuevo atributo *boolean* en Columna.
9. Hay que añadir un nuevo atributo *boolean* en Table.
10. ¿Qué tendríamos que modificar para soportar el modelado de que no podamos tener una persona que tiene el mismo nombre y teléfono, pero sí que tengamos dos personas con el mismo nombre o con el mismo teléfono?
11. Crear un nuevo elemento que se relacione con Column.
12. Crear un nuevo elemento que herede de Column.
13. Crear un nuevo elemento que herede de Foreign Key.
14. No es posible incluirlo en el metamodelo anterior.