Let’s start from the basics.

### Entity

The primary unit of Paragon Low-Code is called “*Entity*”. This is the most fundamental piece of the whole system. An entity is a unit that could contain some information, change it or operate with it using other elements of the system. We consider entities as “bricks” using which we build the whole “building”. Any piece of information that should be stored or shown to the user is keeped inside an Entity.

### Attribute

The additives which are used to colour a brick are called “*Attributes*”. Actually, there could be an entity without them at all, but who needs a brick with no colour?  
Attributes are used to store data inside an entity. We support some basic data types as:

* Address
* Attachment
* Flag
* Date and Time
* GUID
* Number (integer)
* Object
* String
* Long Text
* Time Span

There’s more information about Data Types and their structure here.

An Attribute could also contain *Flags* which are providing some features to that EntityType.  
Actually, a *Flag* in Paragon Low-Code is a True/False selector which may add a certain feature to something.

* IsFixed - means that there’s a set of fixed values for this attribute one of which must be chosen
* IsLocalizable - allows us to introduce various locale translations
* IsSystem - means that this Attribute has been created by the system itself (or the admin, for example) and cannot be changed or deleted at all.

Now, the question is “How to adjust an Entity and specify what it should be like?” Let’s find out.

### EntityType

We call “draft” Entities *EntityTypes.* An EntityType contains the fundamental structure of the future Entities that will be based on this EntityType. It is also possible to inherit an existing EntityType and create a new one with some additional parameters. Some of the EntityTypes can only be used as a parent to inherit them.

To be more precise, we specify attributes of certain data types which means that we’ll be able to create Entities with those attributes which would contain real data. However, adding an Attribute to an EntityType doesn’t mean it wouldn’t be possible to create an Entity of this Type without some of that Attributes. EntityType is only a draft which features the maximum number of Attributes that can be added later. We can create an Entity of a certain type without using all the Attributes of that type with an ability to add them later.

An EntityType could also contain *Flags* which are providing some features to that EntityType.

Here they are:

* isAbstract - means that it is impossible to create Entities of this EntityType. It can only be used as a parent for another EntityType.
* IsSearchable - this flag allows Entities of this EntityType to be found by Paragon.Searcher and displayed on UI. Always use this flag if you are going to work with the Entities on the front-end.
* IsSystem - means that this EntityType has been created by the system itself (or the admin, for example) and cannot be changed or deleted at all.

Each EntityType has its own set of *Permissions* which allow it to create, update, read or delete EntityTypes and Entities related to them. We are not going to talk about this feature much yet.

### References

*Reference* is a special link between two EntityTypes which allows them and their Entities access each other's data. In other words, References describe a set of special relations between EntityTypes which is possible to be transferred to the Entities created as “children” of that EntityTypes.

Let’s talk a bit about what *Reference Types* we have.

1. Reference direction

The reference is always added *From* one EntityType *To* another. This relation actually shows this direction.

E.g a reference called “REF1” which goes *From* EntityType “A” *To* EntityType “B”.

It doesn’t mean that EntityType “B” cannot access “A”, though. This relation just sets the perspective of the link - the “REF1” would be a “From” reference for EntityType “A” and a “To” reference for EntityType “B”. We need to remember this reference direction in order to configure the data transmission correctly.

1. Reference Relation Type

Which could be:

* OneToOne
* OneToMany
* ManyToOne
* ManyToMany

This relation indicates the number of Entities that could be linked to each other. Let’s take the previous example with EntityTypes “A” and “B” and say that their Relation Type is “OneToMany”.

It means that there could be only one Entity of EntityType “A” and many Entities of “B” type linked together.

For that “A” type Entity it would be a “From”, “OneToMany” reference, however

each Entity of “B” type would have a “To”, “ManyToOne” reference.

Of course, there could be multiple “A” type Entities each of which has its own pool of “B” types.

Other Reference Relation Types introduce the same logic which depends on the type itself.

Here’s a short summary to understand that better:

* OneToOne - A single “A” type Entity and a single “B” type Entity  
  Business example - A business and its only Bank Account (There’s only one business in the Low-Code service and it has only one official account)
* OneToMany - A single “A” type Entity and a set of “B” type Entities linked together  
  An organisation and its employees (the employees are working only in this exact organisation)
* ManyToOne - A set of “A” type Entities all linked to a single “B” type one  
  Various goods and the Shop Seller (each product is sold only by this Seller)
* ManyToMany - A set of “A” type Entities linked with a set of “B” typed ones all together  
  The employees and their skills (each skill or a number of skills could be linked to many employees)

As mentioned above, setting a reference between EntityTypes doesn’t mean that Entities created based on one of these Types would be automatically linked to a random (or certain) Entity of the other Type by default. We always choose what to link and how to do so. EntityType settings are just allowing us to use certain link types.

There is some good news by the way! We’re ready to look at all these things in practice to better get the idea.

### Showcase “EntityTypes, Entities, Attributes, References”

Now, let’s try to look at the process a bit closer and find out how to create basic Paragon Low-Code units and configure them. Actually, we already have enough knowledge to use it for a simple practical task such as creating a Low-Code database.

The EntityEditor which is a UI editor using which it is possible to operate with most of the units is situated here: <https://backstage-sandbox.integrity.paragonbox.com/entity-editor/>

One should have a configured *Business Organisation* with granted permissions to work with the system, but we are not going to talk about the Permission system for now. However there’s an article about it [*here.*](https://docs.google.com/document/u/0/d/1RGOmL0IgUWomq0Ns5k5gNWIb2RjY9tw-ggcQrK8n5-o/edit)

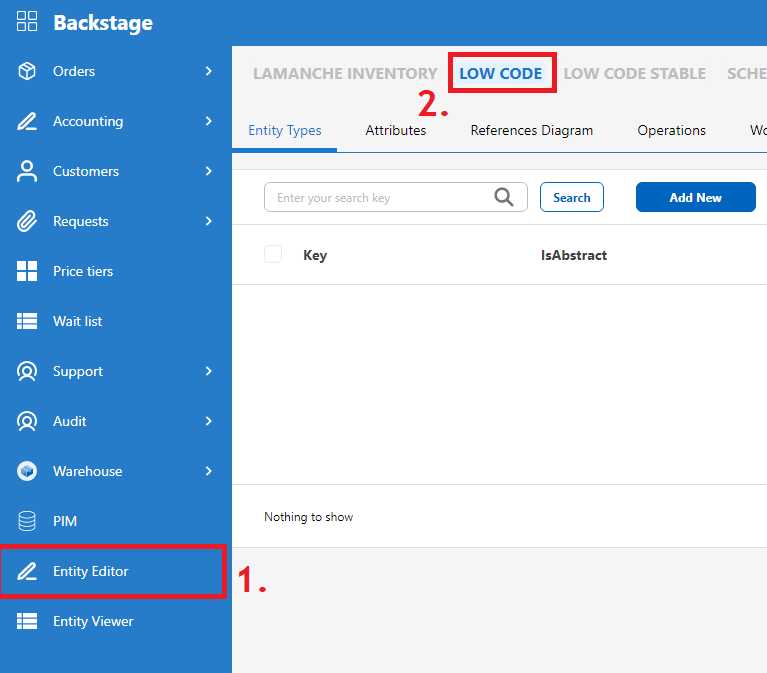
There’s a test Business designed for training:

Login: bs\_seller@paragon-software.auto

Password: Qwerty123

Organisation name: Low-Code Glossary Test

Log in to the Low-Code editor and switch to the “Entity Editor” tab, then choose “LOW CODE” service”. This is the instance of the Low-Code service we are going to use as a sandbox playground.



#### Showcase scenario

There’s a huge Ebay-like online shop where various sellers may sell their goods. Congratulations! You are the owner of the shop. The thing is before you start earning money and become a rich man you need to establish a database which includes all the information about the sellers and their goods. Seems difficult? Well, it’s not this hard with Paragon Low-Code!

Actually, the first thing you need to do is to think about what attributes your Entities are going to use. Some default ones such as “Creation date” of DateTime type or “Title” of String type would work, but usually there’s a need to add something else.

The testing environment is empty at all though. Okay, let’s think what Attributes we actually need to establish the shop. To do that we should first imagine what entries our database would contain.

I suggest this one:

1. Organisation - a list of all the sellers. We may store such information about the sellers as:

* Name - a string attribute
* Email address - a string attribute
* Address - an “address” type attribute
* Country - a string attribute
* VAT rate - a number type attribute

1. Product - something a seller could sell. To make it easier:

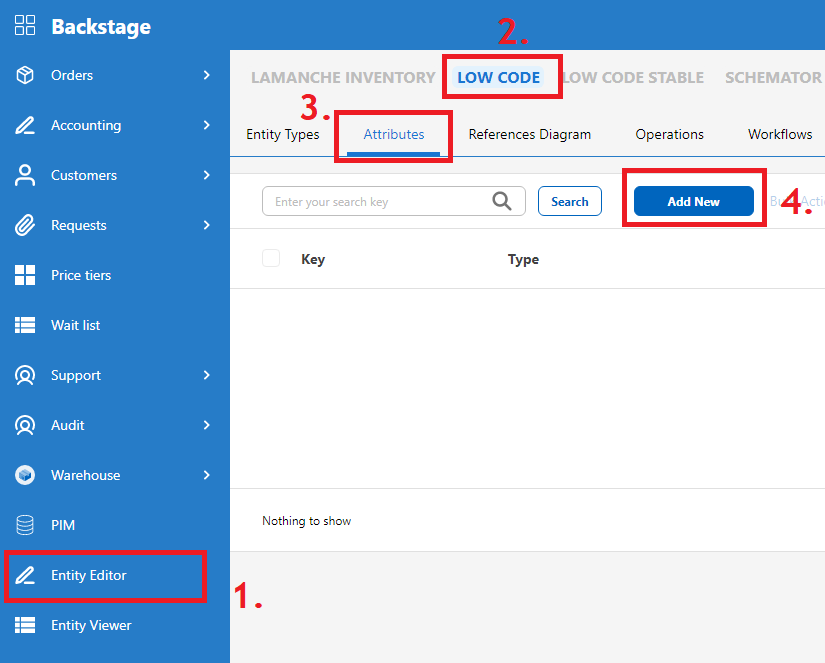
* Name - a string attribute
* Price - a number type attribute

1. Buyer - a list of users who bring us the money:

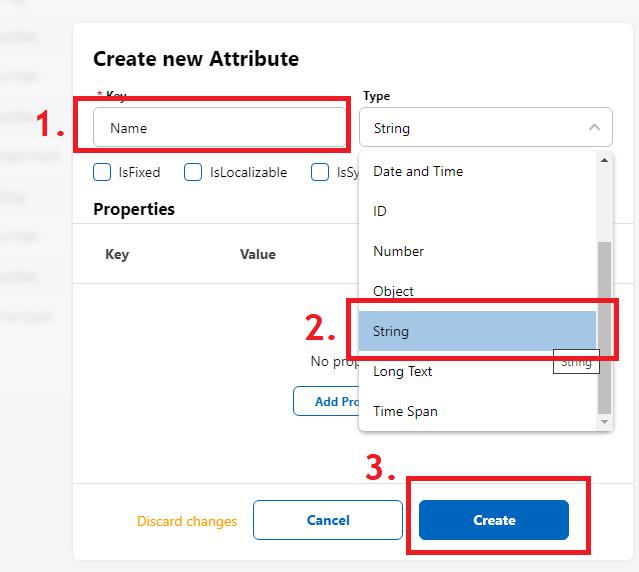
* Personal name - a string attribute
* Address - an “address” type attribute
* Country - a string attribute
* Date of birth - a datetime attribute

**We are now ready to create those attributes and EntityTypes that are going to use them.**

Get to the Entity Editor Attributes section and click on “Add New” button

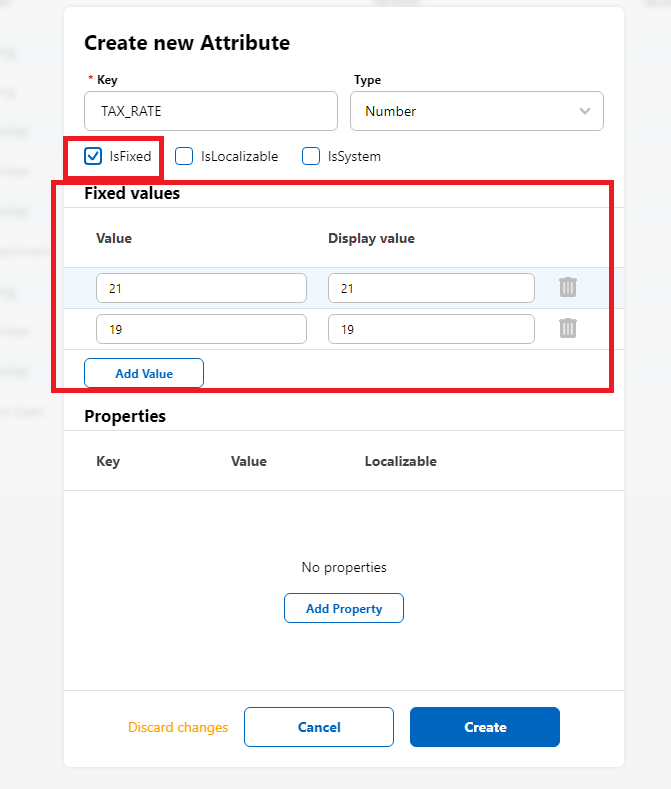


Fill in the fields and create each Attribute by typing in its Key name and choosing its Type.



The only Attribute we can do something more with is “VAT rate”. Why don’t we add some fixed values defining the tax rate? Let’s imagine we only work in 2 countries which use 18% and 21% tax rates.

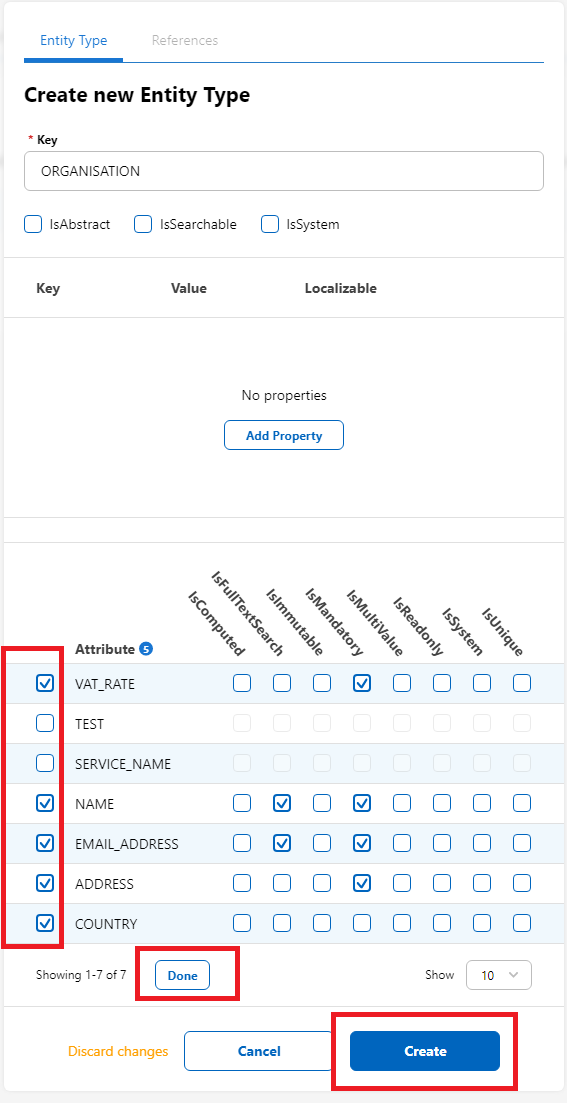
Enable the “IsFixed” flag and add the values needed.



**The next step is creating the EntityTypes.**

Switch to the “Entity Types” tab of the Entity Editor. Click “Create New”

Here’s the EntityType creation window.



Click the “Add Attribute” button and choose the Attributes that you want to add to the EntityType. Don’t forget to click “Done” before creating it!

Create Organisation, Product and Buyer EntityTypes by your own - use the logic described above and that piece of information about Flags below.

Important! Do not edit the EntityTypes using UI if you’ve added any additional properties that are not presented here (E.g. Computed Attributes). Everything else is going to disappear from the EntityType.

##### EntityType Attributes Flags

Take a look at the screenshot above - we’ve already used some of the Flags there. Here’s a short explanation that would help to understand the meaning of each Flag and why it has been used here.

* IsComputed - means that this Attribute could be calculated by its own. We’ll talk about such attributes separately a bit later.
* IsFullTextSearch - Attributes marked by this flag are going to be a part of full text searching. We marked the “NAME” and “EMAIl\_ADDRESS” attributes to easily search for organisations.
* IsImmutable - immutable attributes get their value only once - either on Entity creation or during Entity update if this attribute is added to the Entity. The value cannot be changed later whatsoever.
* IsMandatory - remember that we can create an Entity not using all the Attributes added to the corresponding EntityType? This Flag denies that ability meaning that an Entity must contain at least this exact Attribute.
* IsMultiValue - allows us to add multiple values to this Attribute.
* IsReadonly - this flag forbids to change the value of an Attribute manually, even during Entity creation. Only automatic calculation (we’ll talk about these features later) may change the value of such Attribute.
* IsSystem - means that this Attribute has been added here by the system itself (or the admin, for example) and cannot be changed or deleted at all
* IsUnique - IsUnique flag reserves the value of the Attribute by making it unique. No other Entities of this EntityType can get the same AttributeValue.

##### EntityType References

I bet you’ve noticed the “References” tab was disabled during the EntityType creation. It happens due to the fact we create References using a separate controller. We now can edit the EntityTypes and add the References we need.

Let’s remember the logic of References and think of what we should do now.

My suggestion is here:

ORGANISATION (seller) is the main part of the hierarchy. Each Organisation has its own set of Products.

At the same time, Buyers may buy products from different Organisations at the same time.

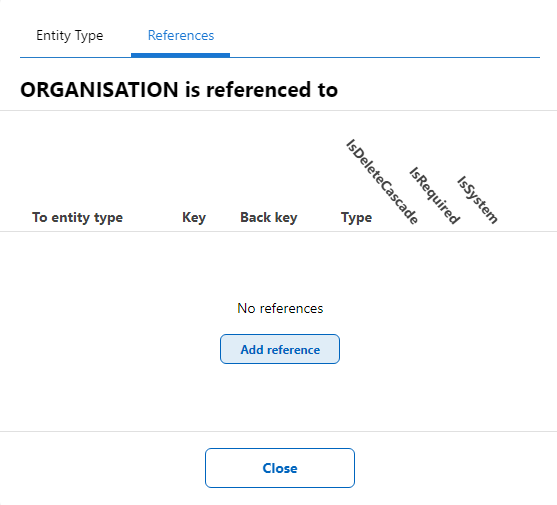
It means, the references should be like:

* From ORGANISATION To BUYER (ManyToMany)
* From ORGANISATION To PRODUCT (OneToMany)

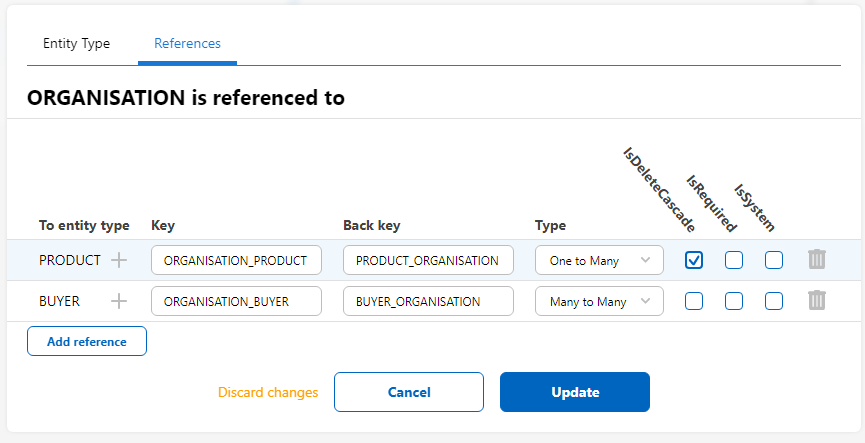
Buyers are going to have access to each Organisation Products through Reference Links of each Organisation links which means that a Buyer would see only that Products which are linked to the Seller Organisation as this Organisation’s goods.

Let’s Establish the Reference Links for now.

Edit the Organisation EntityType and click the “References” tab, then “Add reference”.



Here’s what it should look like:

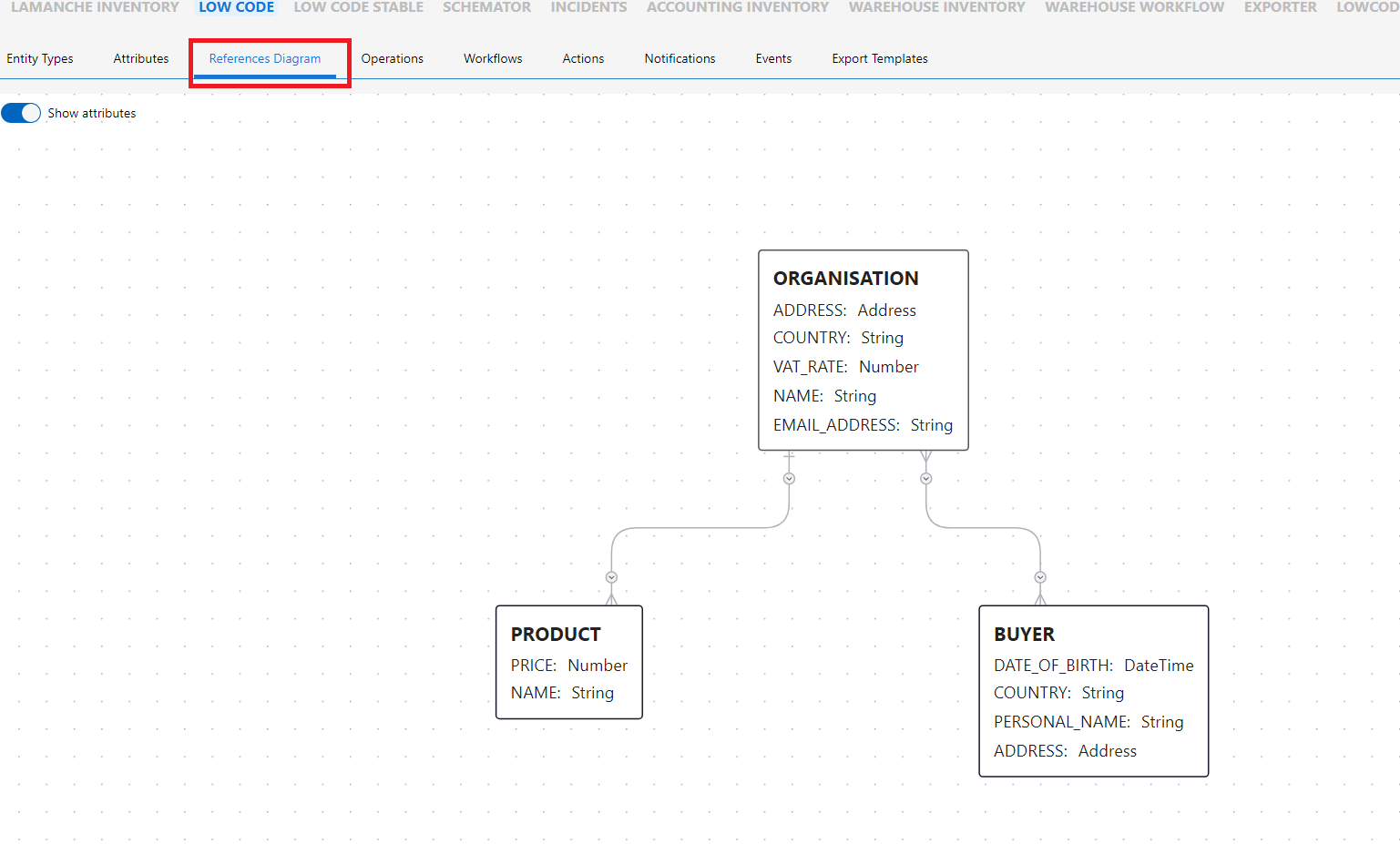


A few words about the Flags:

* IsDeleteCascade - deleting the From Entity (Organisation) would result in deleting all the To referenced Entities together with the From one
* IsRequired - indicates that the From Entity (Organisation) cannot be created without these references
* IsSystem - tells us that this reference is needed for the System to work properly

We’ve marked the IsDeleteCascade Flag for the Product reference to delete all the Organisation’s products automatically if an Organisation leaves our marketplace.

“References Diagram” tab now shows what we have just created and referenced all together.



Now, you may try to create some Entities and fill the Shop with Sellers, Buyers and Products!