CONSONO

Data engines are the same all over the world…

Except one.

For more detail see https://api.dynizer.com

The Dynizer

In brief

The Dynizer data harmonization engine's patented technology virtualizes, integrates and manages structured, unstructured and semi-structured data from any source, making it useful in any application.

The Dynizer was developed based on the realization that any data element was either a person, a thing, a location, or a time and place. A Who, a What, a Where, or a When.

The Dynizer benefits any sector where harmonizing data into one universally usable resource, and the ability to combine structured and unstructured data in real time, makes a recognisable improvement in operational outcomes. These include: Data Governance and compliance (eg: GDPR).

The Dynizer harmonizes all kinds of data, whether that's in rows and columns, like spreadsheets, or in plain text in anything from emails to documents. It treats all data the same way…

When data is harmonized into the Dynizer, the platform automatically detects direct and indirect links between data elements, finding without searching and telling you things you didn't know…

What does that mean for your data? It means you can use it anywhere you want without having to worry about the way specific data models were set up for particular applications. The model is less complicated, doesn't break, queries remain valid and no data is lost.

Autonomous

Re-use data for different purposes without building specific models, using separate microservices to ingest, store and query data, using auto-indexing, scaling and automatic de-duplication.

DQL

Use the SQL compliant Dynizer Query Language (DQL) enabling querying of both structured and unstructured data without having to know the data model.

All the data

Map all data in a universally useful structure and flexibly integrate new data into an existing context without breaking queries.

Fluid width columns

Go beyond any boundaries with flexible query results in the form of fluid width column tables with variable datatypes.

The Dynizer

Nuts and bolts

From the chapter The Dynizer in Michael Brands' book 'Data Harmonization in the Key of C (or How to Tune Your Data)

In terms of architecture, the Dynizer is made up of a limited number of microservices. Each task that we need to perform to get data into the engine, to store the data and to answer queries is implemented as a separate microservice.

Everything is handled by a single module. And within the data model we can load each function to the maximum before spreading the load through further distributed modules.

It can import load data through two dedicated pipelines. The first pipeline deals with structured data – from relational databases, document databases or graph databases - and the second handles unstructured data that uses bespoke Intelligent Indexing to abstract the Who, What, Where, and When elements in the data.

To query the database and we use lookup indices. We use Bleve for text, geoindexing for locations, time for date/time information and numeric for numbers.

The query service runs separately, which means it can run multiple queries that can branch off in order to guarantee the best possible performance in returning information to an application.

The query server looks only for the ID of a data element in the indices, and only when it has resolved that does it return the data. It means you can do data comparisons without having to look at the data itself, which is important in data security.

You can run also multiple applications against the same data store and if your data model changes, you just adapt it.

Fuel and spark

Entities can be built by combinations of Who, What, Where, and When. So, whatever the piece of data is, it's represented by one of the four.

Instead of having a problem of endless dimensions, where you create the Entities as you go, here you almost have something that's more like a four-dimensional problem. It's not exactly four-dimensional because they are combined into 'words' and then 'sentences' called Actions.

They can then be used in any kind of application for analysis, no matter what was the model in which the data was created. Even so, it's quite a reduction in relation to the infinite problem it was before and it's what gives us an advantage over classical systems.

If you can represent the different parts of your data as Whos, Whats, Wheres, or Whens, you can immediately see the overlap between the Actions, because they will have the same Who at some point, or the same What, Where, or When.

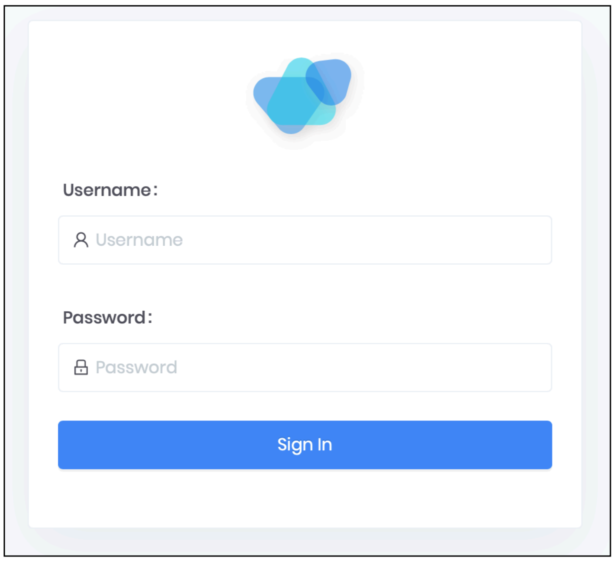
In that way you can build flexible schema that show you immediate links - the direct connections - but also indirect connections.

The Dynizer creates the links based on overlaps in the data. An added bonus is that you don't have to know where the data is stored to figure out how you can retrieve it.

You don't have to specify, for instance, that 'you need to go and look in the customer table', or 'you need to go and look in the body text of an email'. You only need to ask the question and the system will bring you the information, no matter where it comes from.

You will be able to figure out where it comes from based on the information the system brings back, but you don't have to know it up front. It helps because often information is there, but people don't know where to look for it.

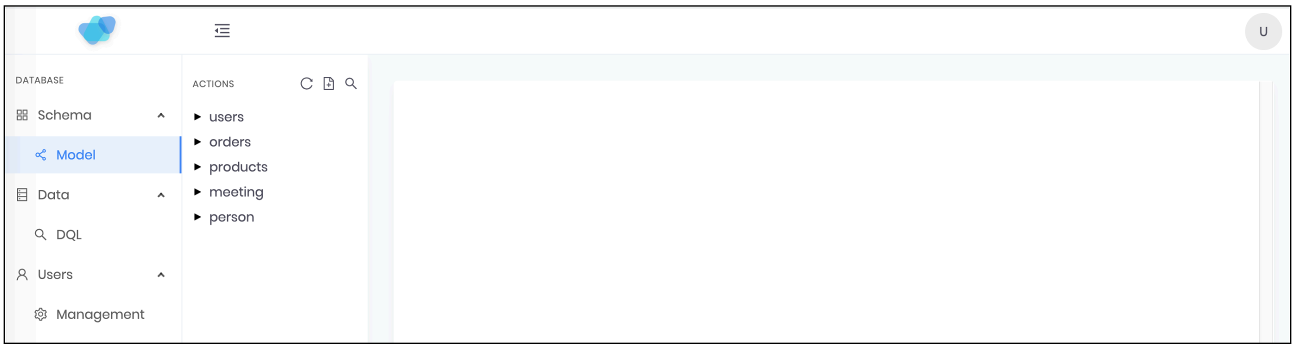
The Base UI https://api.dynizer.com/#/authentication/login



1. The Login Screen
Login screen has fields for Username and Password.

Click 'Sign In' button to log in.

2. Base UI Home Screen Default View



3. Home Button

Click here to return to main screen

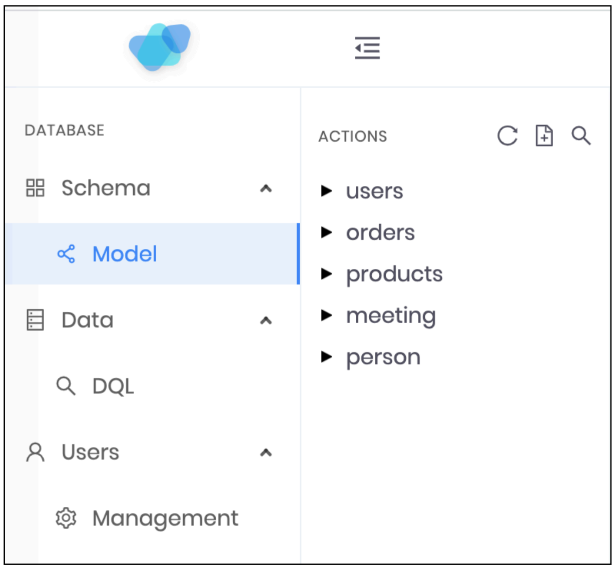
4. Concertina Button

Click here to collapse DATABASE column

5. User Button

Click to open User Profile window

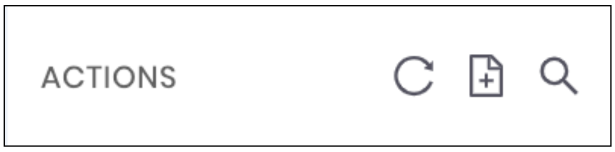
6. Base UI Home Screen Menu Detail



The Home Screen opens up with the elements of the DATABASE column expanded.

These elements are: Schema > Model; Data > DQL; Users > Management.

Schema > Model is automatically selected to reveal the ACTIONS column, which contains all the Actions in the model.



The top of the Actions menu shows three buttons, from left Refresh; New, Search

Clicking Refresh repopulates the Actions list.

Clicking 'New' creates a new Action.

The Actions listed here are:

users

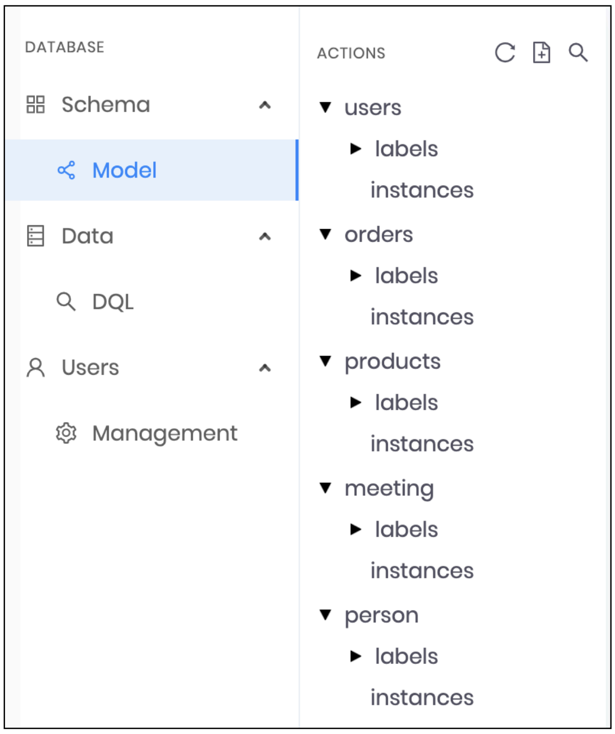
orders

products

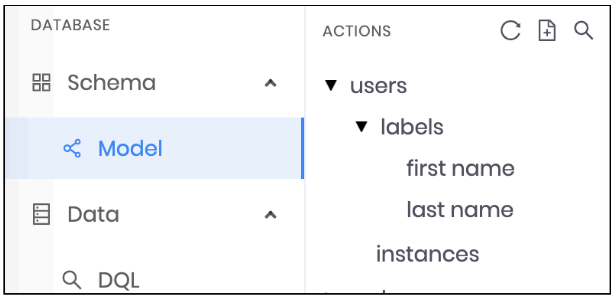
meeting

person

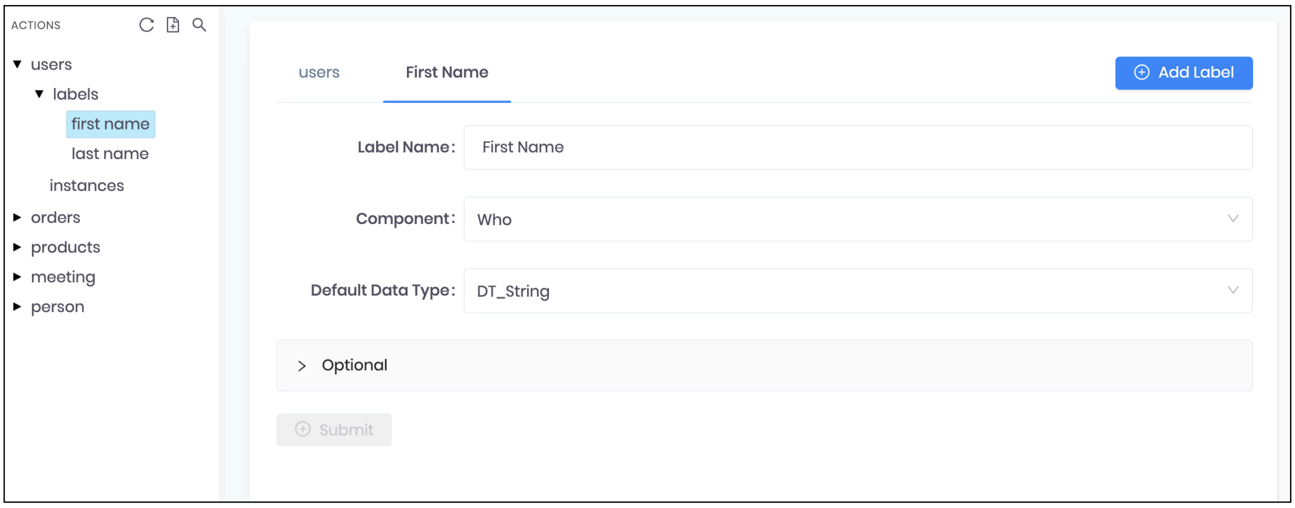
7. Actions Column Expanded



When the disclosure triangles against each Action are expanded, they reveal a second level; labels and instances.



Clicking the disclosure triangle against labels reveals the labels contained within the Action 'users' - in this case 'first name' and 'last name'



Label Name: First Name

Component: Who

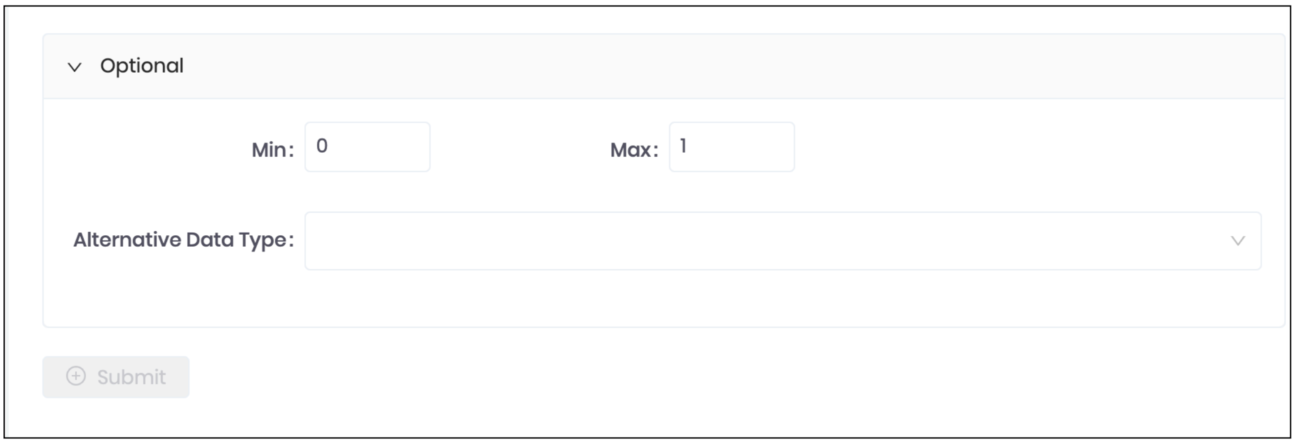
Default Data Type: DT_String

Clicking on the label 'first name' reveals an input window containing the details of the label 'users > first name.

NB: in this window First Name is capitalised, but in the ACTIONS column all the text is lower case.

Details include: Label Name [First Name]; Component [Who]; Default Data Type [DT_String]. The DT_ prefix in this case is optional.

Schema > Model is automatically selected to reveal the ACTIONS column, which contains all the Actions in the model.



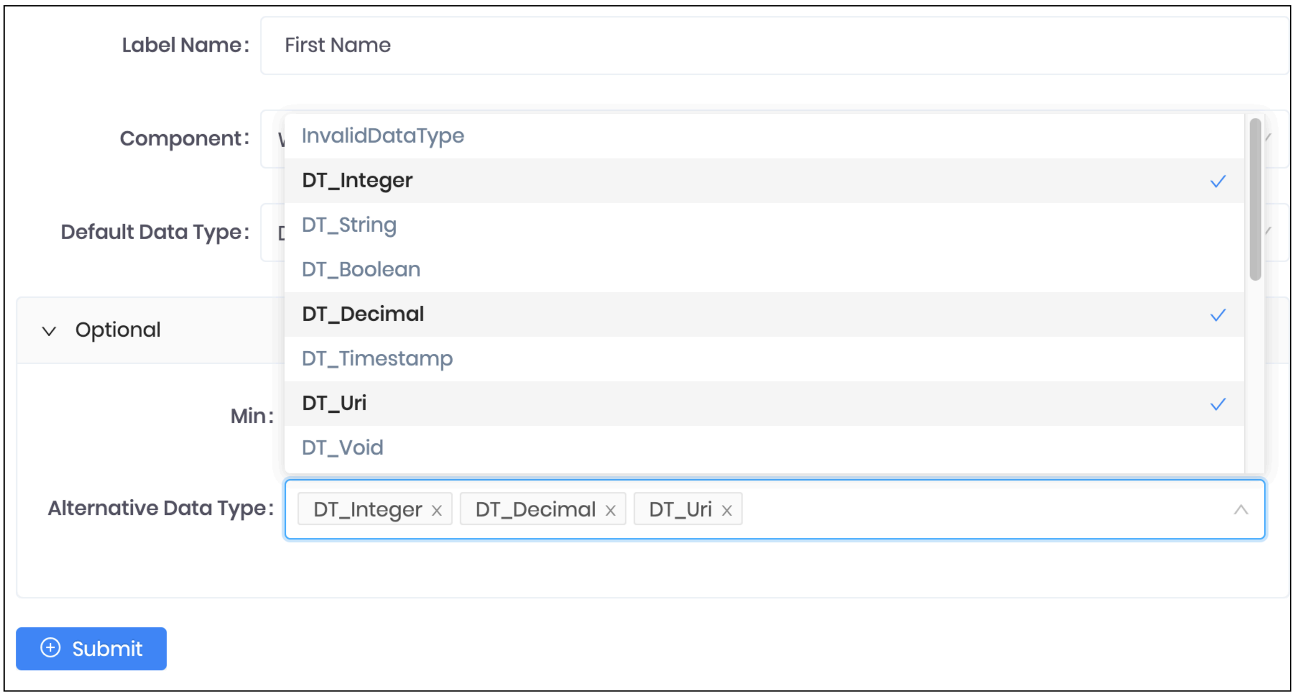
Min: 0

Max: 1

Alternative Data Type:

Below the details is an 'Optional' field, which can be turned down to reveal the Minimum and Maximum occurrences which can be assigned to each label. If Maximum is set to 0 (zero) it will be interpreted as open ended, otherwise the value can be equal to or greater than the Minimum. In this case, for the label 'First Name', the minimum occurrence is set to zero and the maximum to 1.

It is also possible to select an Alternative Data Type to attach to the label. Clicking in the field reveals a drop-down menu from which to select an alternative.


Label Name: First Name

Component: V

Default Data Type: D

Optional

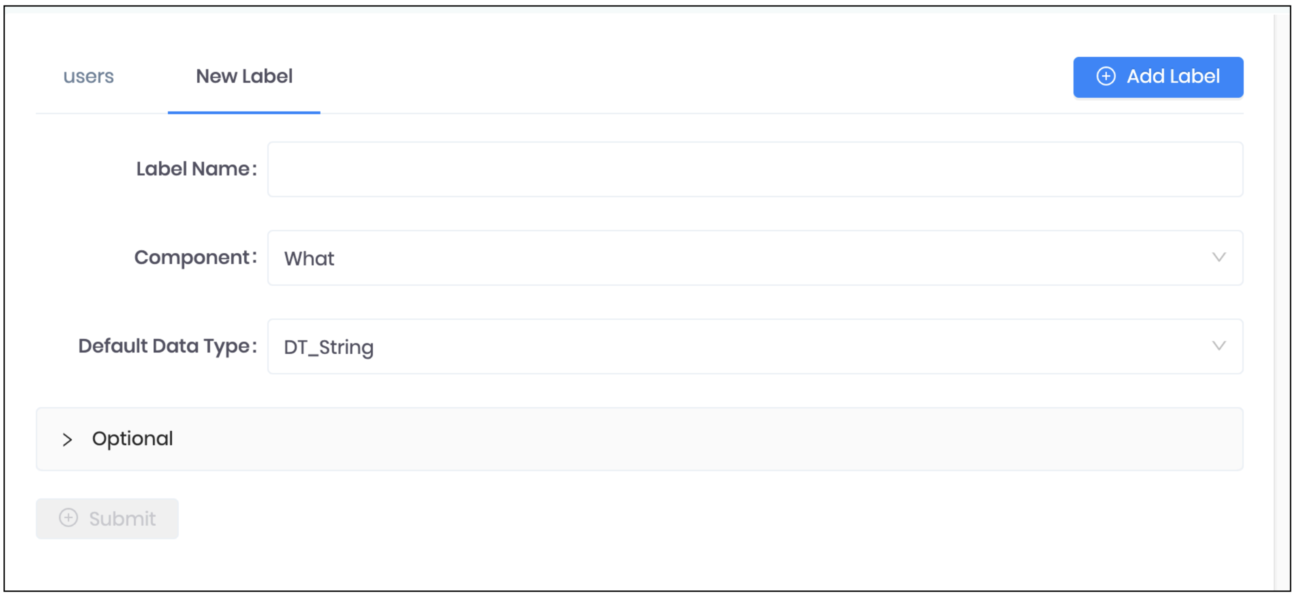
Min:

Alternative Data Type: DT_Integer × DT_Decimal × DT_Uri ×

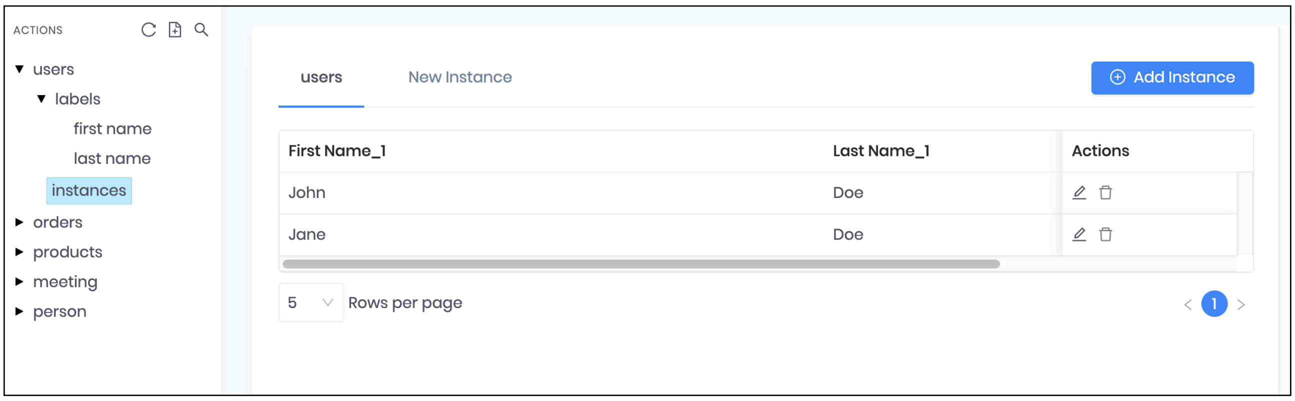
In this version, the list of Data Types is: InvalidDataType; DT_Integer (whole numbers in the range -2,147,483,647 to -2,147,483,647 for nine or 10 digits of precision, stored as a signed binary integer); DT_String (representing text rather than numbers); DT_Boolean (data that has one of two possible values, usually true or false); DT_Decimal (exact numeric data defined by its number of digits and scale - the digits to the right of the decimal point); DT_Timestamp (values that contain both date and time parts); DT_Uri (base64 encoded string that can be directly embedded into html or css); DT_Void (indicates no value); DT_Float (double-precision floating point numbers up to 17 significant digits); DT_ UnsignedInteger (can hold zero and positive numbers); DT_Binary (similar to char and varchar data types, contain byte strings; DT_Uuid (16-byte numbers used to uniquely identify records); DT_Text (variable length character data type with a default value of 1); DT_Blob (binary large object, a pointer to an object such as an image, video etc); DT_Clob (character data up to 2gb); DT_Any; DT_Delete

It is possible to make multiple selections from the drop down list. When selected they are added to the Alternative Data Type field. Clicking 'Submit' saves the changes.

Clicking the 'Add Label' button creates a new label window.

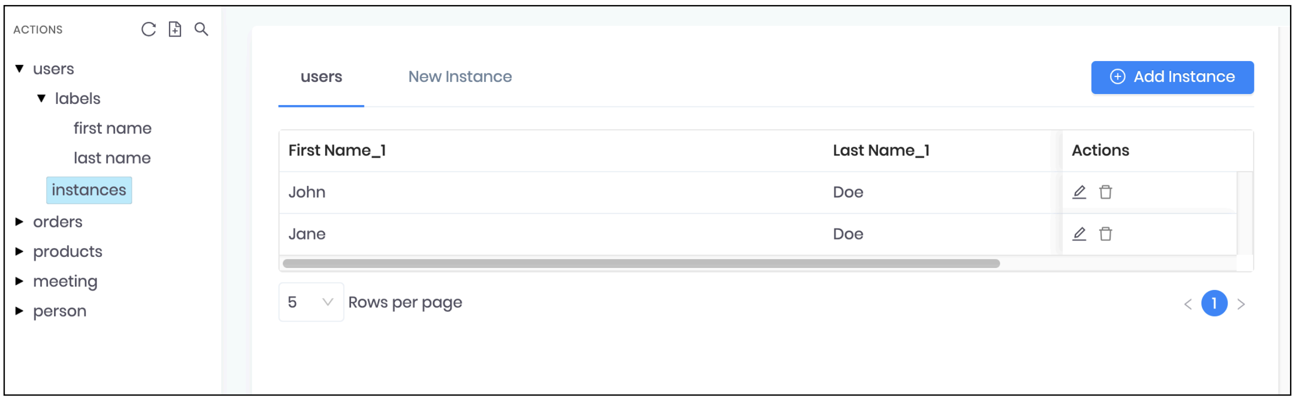


Label Name:	
Component:	What
Default Data Type:	DT_String



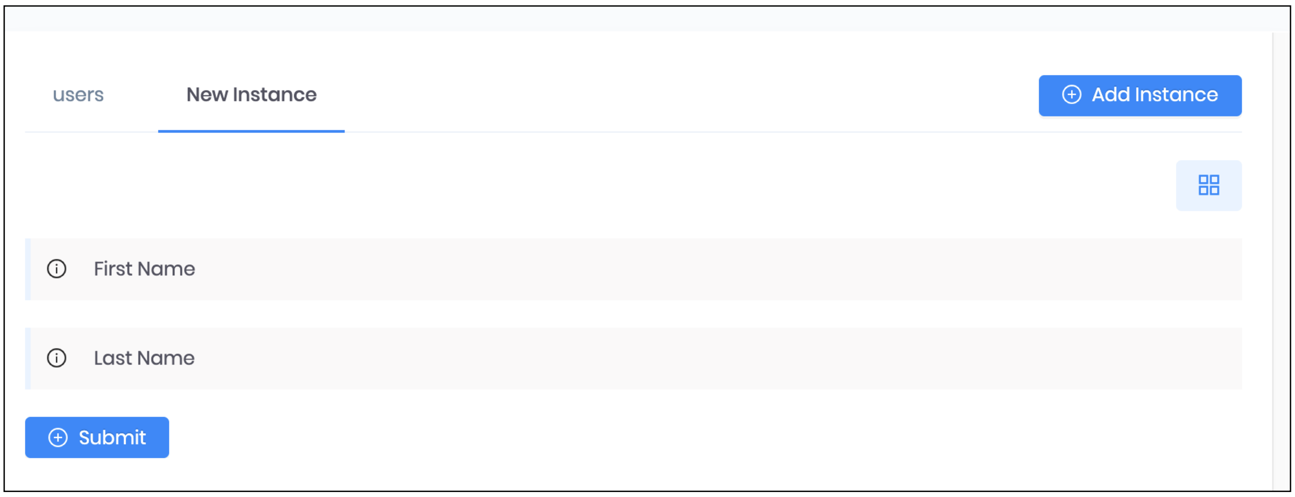
First Name_1	Last Name_1	Actions
John	Doe	
Jane	Doe	

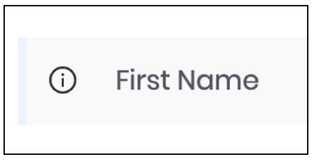
Selecting users > instances in the ACTIONS column reveals the 'users' window, with a table.



First Name_1	Last Name_1	Actions
John	Doe	
Jane	Doe	

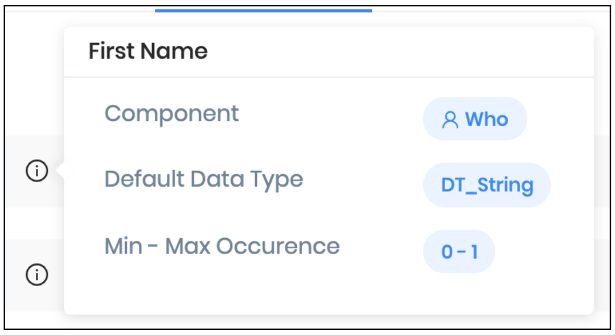
Clicking either 'New Instance' or the 'Add Instance' button opens a new window in which to add a new instance.





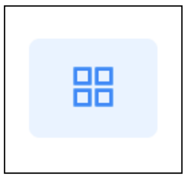
Next to each label is an Information button.

Clicking the information button reveals the details of the Component, Default Data Type and Minimum and Maximum Occurrence values.

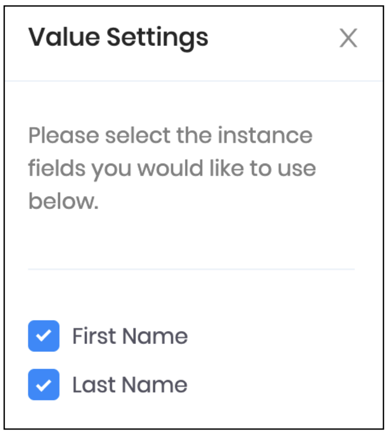


First Name

Component	Who
Default Data Type	DT_String
Min - Max Occurence	0 - 1



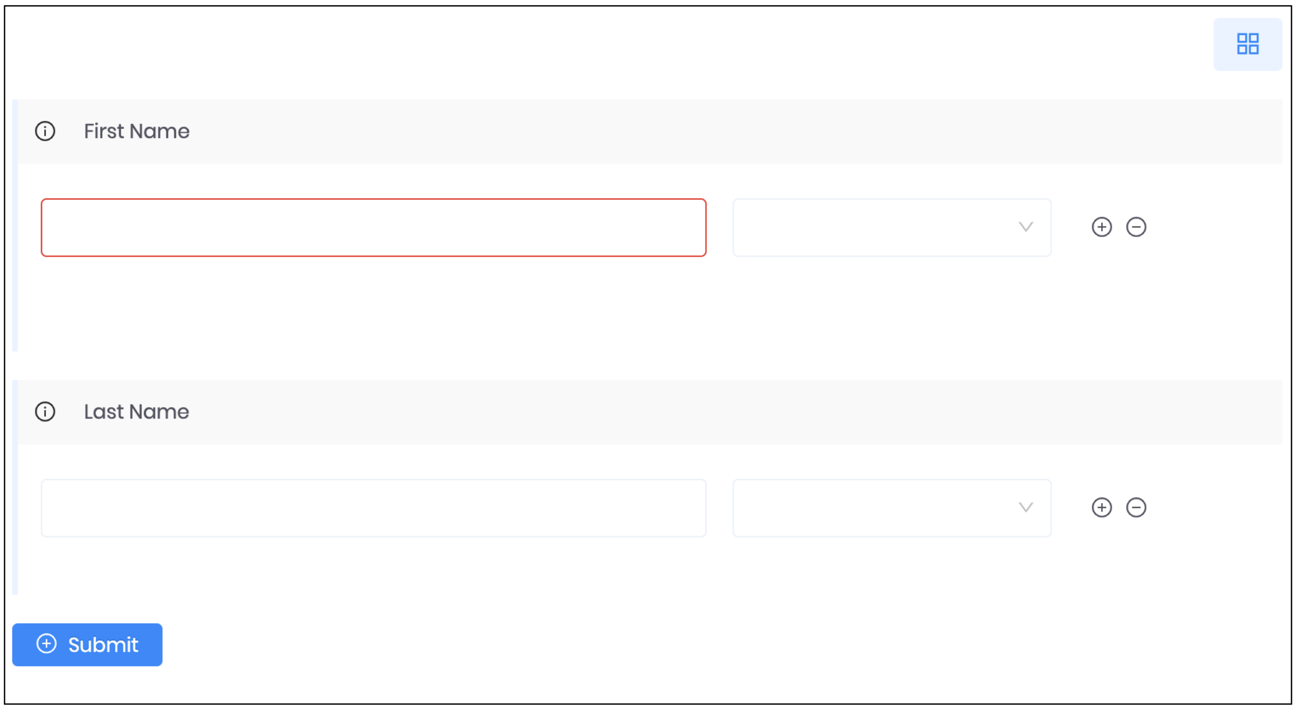
To the right of the window is a 'Value Settings' button.



Please select the instance fields you would like to use below.

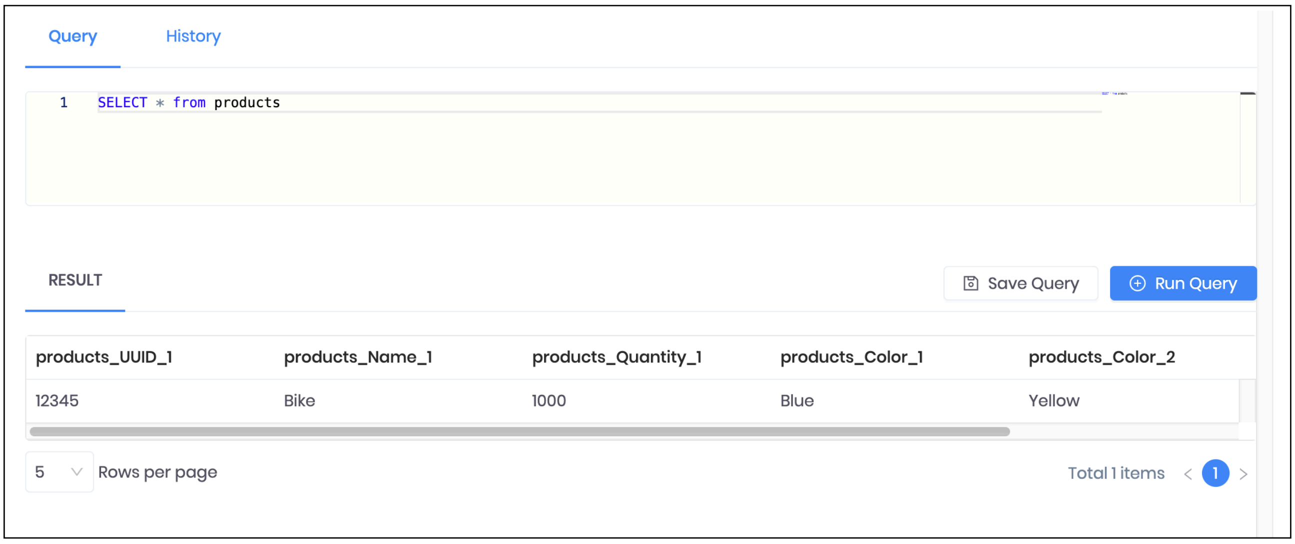
- [✓] First Name
- [✓] Last Name

This window allows users to select which labels they wish to edit.



Information can be inserted into each text field and saved by clicking the 'Submit' button.

8. The Data Menu



```
SELECT * from products
```

products_UUID_1	products_Name_1	products_Quantity_1	products_Color_1	products_Color_2
12345	Bike	1000	Blue	Yellow

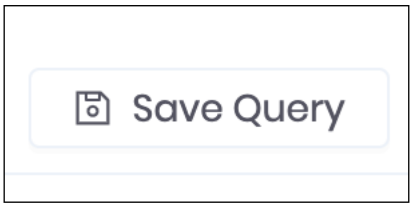
The Dynizer can be queried using Dynizer Query Language, which is an enriched version of SQL and is compliant with that standard.

Clicking on Data > DQL in the DATABASE column reveals the DQL Query Window, in which users can create multiple DQL queries, by hitting Return after completing each statement. The beginning of each DQL statement is automatically filled in as: SELECT * from

The simple statement above (SELECT * from products) reveals the result shown, when the blue 'Run Query' button is clicked.

You will notice that the column labels are displayed in the format [action_LABEL_increment]. This is to distinguish from labels of a similar name. In this case you will see, for instance: products_UUID_1.

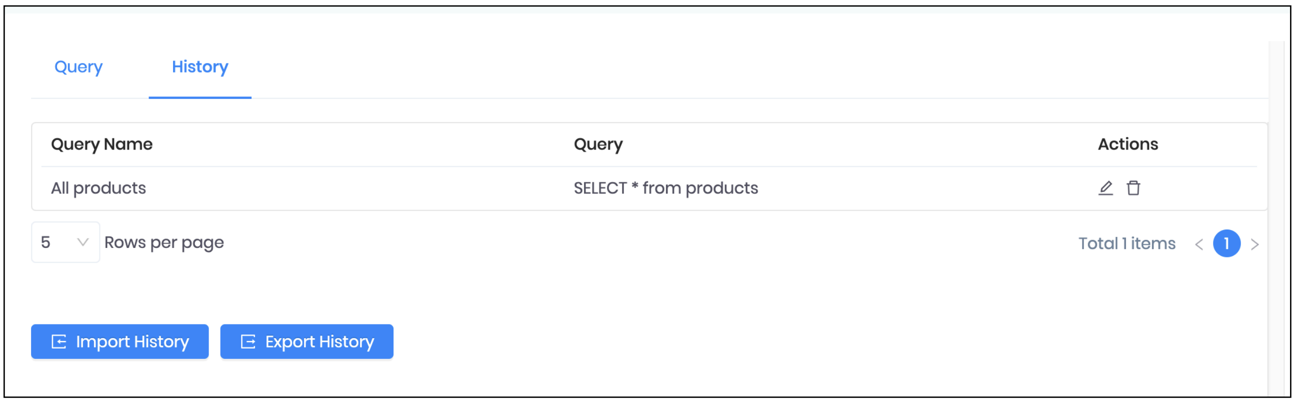
In this, as in other display windows, it is possible to display up to 30 rows in increments of 5, 10, 20, 30. The blue figure bottom right shows pagination against a view of the total number of items.



Clicking on this Save Query button stores the query for future use.

The saved queries appear in the History window, which can be displayed by clicking on History at the top left of the page.

The Data Menu continued

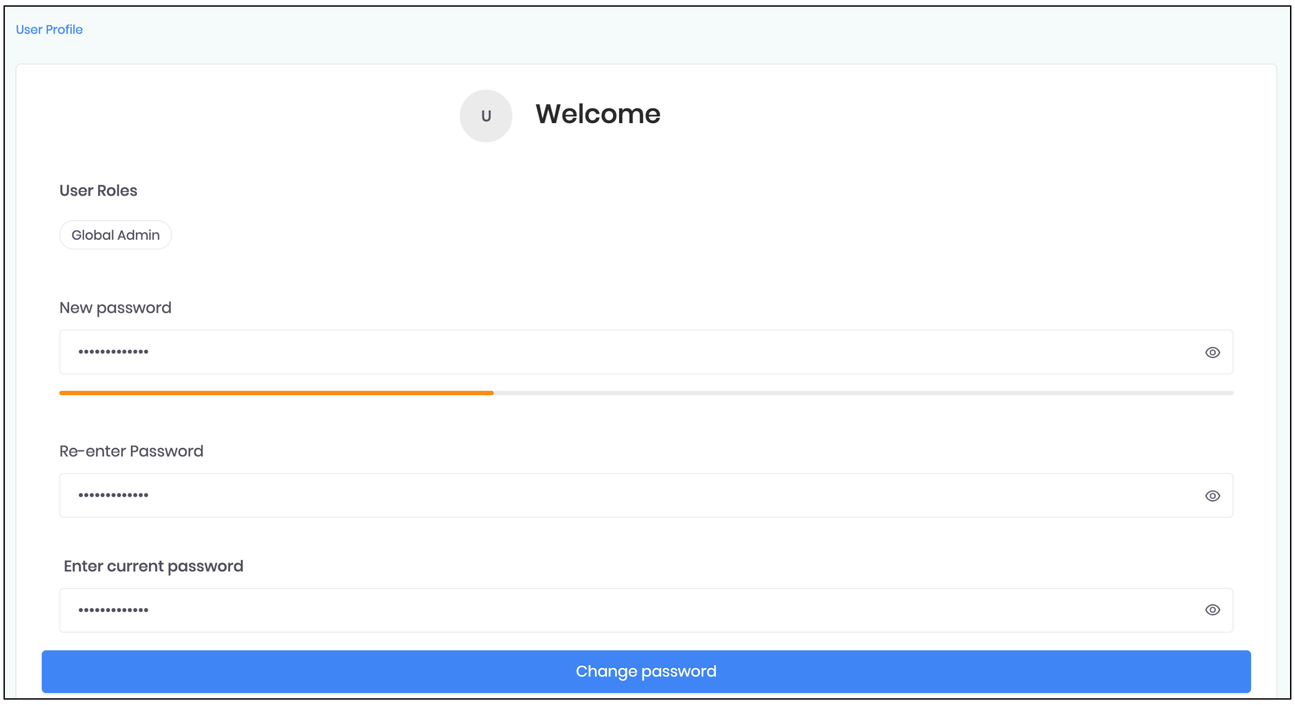


Query Name	Query	Actions
All products	SELECT * from products	

This shows the current history, with the saved query ‘All products’. The table gives a brief overview of the query (SELECT * from products) and offers the chance to edit or delete the query using the buttons on the right of the row.

It is possible to export queries using the ‘Export History’ button. Similarly, it is possible to import queries using the ‘Import History’ button. Queries are saved in JSON format.

9. User Profile window



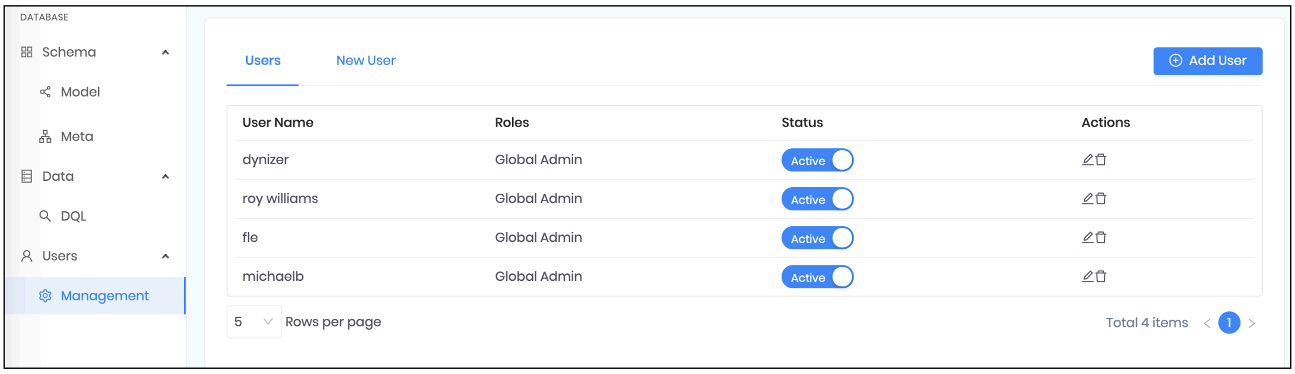
You can use this window to view User Roles (in this case Global Admin) and to create new passwords.

Clicking the 'eye' icon to the right of each field makes the text within the field visible. Unchecking makes the text invisible.

To change passwords, the user must fill in the 'New password' field then fill in the 'Re-enter Password' field. The input text will be validated at each point.

Filling in the 'Re-enter Password' field enables the 'Change password' button.

10. User Management



DATABASE

- Schema
  - Model
  - Meta
- Data
  - DQL
- Users
  - Management

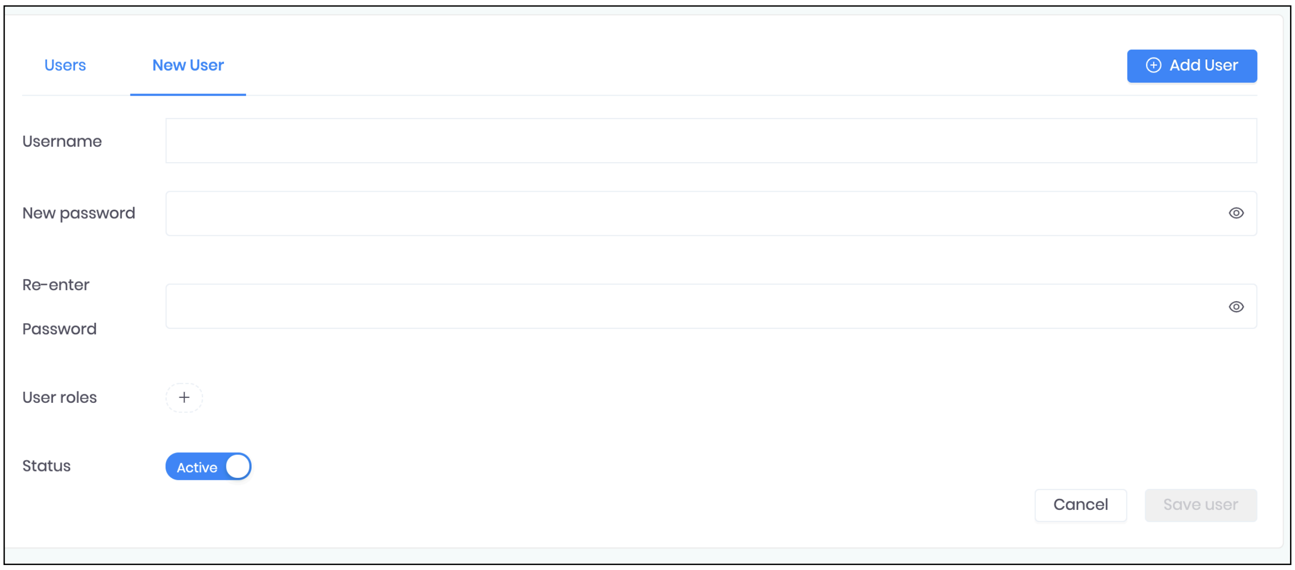
Users | New User | Add User

User Name	Roles	Status	Actions
dynizer	Global Admin	Active	
roy williams	Global Admin	Active	
fle	Global Admin	Active	
michaelb	Global Admin	Active	

5 Rows per page — Total 4 items — 1

Clicking on DATABASE > Users > Management reveals the list of users of the Dynizer with their Roles and Status. Status can be toggled as Active or Inactive. At the end of each row are Edit and Delete buttons.

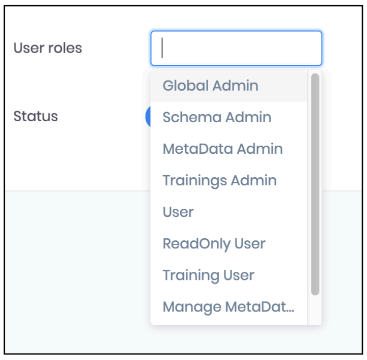
Clicking the blue ‘Add User’ button or on ‘New User’ reveals the New User Window



Users | New User | Add User

Username	
New password	
Re-enter Password	
User roles	+
Status	Active

Cancel | Save user



User roles	
Status	

Here administrators can complete new user registrations, including Username, Password, User role, and Status (Active or Inactive).

User roles can be added, or selected from the dropdown menu, by clicking on the ‘+’ button (see right).

Once completed the user registration can be saved by clicking the ‘Save user’ button, or cancelled using the ‘Cancel’ button.