



Administrator Manual

MINT Team — MaaS Project

Information about the document

Version	2.0.0
Editing	Michael Ogbuachi, Navid Taha, Tommaso Zagni
Verification	Enrico Canova, Fabiano Tavallini
Approval	Michael Ogbuachi
Use	External
Delivery	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	MINT Team

Description

This document is meant to guide the user's (Administrator or Owner of a Company) first approach to the MaaS system, developed by MINT. It is also intended as a document for the Administrators and Owner of a Company, that **being such describes actions that only the two roles can perform, and should be read alongside the *User Manual* in order to entirely understand its content.**



Changelog

Version	Date	Collaborators	Description
2.0.0	04-09-2016	Michael Ogbuachi (Person in charge)	document approval.
1.1.0	04-09-2016	Fabiano Tavallini (Verifier)	document verification.
1.0.2	03-09-2016	Tommaso Zagni (Administrator)	images upload
1.0.1	02-09-2016	Michael Ogbuachi (Person in charge)	several additions - draft.
1.0.0	02-08-2016	Michael Ogbuachi (Person in charge)	document approval.
0.1.1	01-08-2016	Fabiano Tavallini (Verifier)	images upload.
0.1.0	31-07-2016	Fabiano Tavallini (Verifier)	document verification.
0.0.2	29-07-2016	Enrico Canova (Administrator)	Chapter 2 - draft.
0.0.1	26-07-2016	Enrico Canova (Administrator)	Chapter 1 - draft.

Contents

1	Introduction	3
1.1	Aim of the document	3
1.2	Aim of the project	3
1.3	Prerequisites	3
1.4	Issues reporting	3
2	Account and authentication	4
2.1	Sign up - New company owner	4
2.2	Sign up - On invitation	4
2.3	Login	5
2.4	Password recovery	5
3	Role-specific management	6
3.1	Company and User management	6
3.2	Database management	8
3.3	DSL management	9
4	Language description	10
4.0.1	Cell configuration	10
4.0.2	Document configuration	13
4.0.3	Collection configuration	15
4.0.4	Dashboard configuration	18
4.0.5	Row configuration	21
4.0.6	Column configuration	22
4.0.7	Action configuration	22
	Glossary	24

List of Figures

1	Central view of the signup page for a new Company Owner	4
2	View of the signup page for an invited user	5
3	View of the login page	5
4	Central view of the password recovery page	6
5	View in which the members of the Company are shown	6
6	Main view of the database management page	8
7	View in which the DSL instruction sets are shown	9
8	Example of the visualization page of a Cell query	13
9	Example of the visualization page of a Document query	15
10	Example of the visualization page of a Collection query	18
11	Example of the visualization page of a Dashboard query	21

1 Introduction

1.1 Aim of the document

This document is intended for the user who has to learn how to properly interact with the MaaS system. For this reason it shows how the main operations have to be done, displaying all the procedures based on the UI of the service.

There are no knowledge prerequisites the user should meet to start using MaaS, as the interaction shall be performed through a web *Browser_G*, just as it is done with any other website. In addition to that, the technical aspects of the usage shall be thoroughly explained where necessary.

1.2 Aim of the project

The purpose of this project is the development of a system usable through the web called MaaS, aimed at at the so-called *Business-men_G*, namely people having a key-role in a company but lacking advanced technological knowledge, to help them making administrative and commercial decisions by providing a platform to visualize data stored in an easy to use *Database_G*, which is at the same time equipped with great capabilities.

1.3 Prerequisites

The user must have an internet connection and a web browser (Google Chrome v.51 or higher and Mozilla Firefox v.47 or higher are recommended). The system itself, being a service, shall take care of all its intrinsic necessities.

The user should also subscribe an account on the system to make an effective use of its functionalities.

1.4 Issues reporting

In the case of errors or malfunctions refer to the *Webmaster_G*, namely the Super-Administrator. The message containing the error report must include the following information:

- An effectual contact of the person encountering the problem;
- Date and approximate time in which the problem occurred;
- Error code if displayed;
- Actions and pages involved with the problem;
- Eventual notes that may help the comprehension of the issue.

A reply shall be sent within the shortest time possible.

2 Account and authentication

This section discusses all the steps needed to create an account and authenticate to the system. It is important to notice one important fact though. There are two different types of user registration to the system:

- **Sign up as a company owner**

The user should undertake this type of registration if he intends to build a new company in the system, of which he shall be the owner. (*Jump to 2.1 - New company owner*)

- **Sign up on invitation**

The user should undertake this type of registration if he has to join a company (become a member of it), having received a prior invitation by the same. (*Jump to 2.2 - On invitation*)

Anyway the user should first of all access the homepage of the system. This can be done by typing in the address bar on the browser the link to the system and pressing the Enter button on the keyboard.

The black bar on the top of the page contains the links to sign up and log in.

2.1 Sign up - New company owner

Clicking on the "Sign up" button, a sign up page will come up. Hence the user can create his own company by adding the chosen personal credentials (username, email and password).

At this point the user can just click on the confirmation button and the account (if the name is available) will be created.

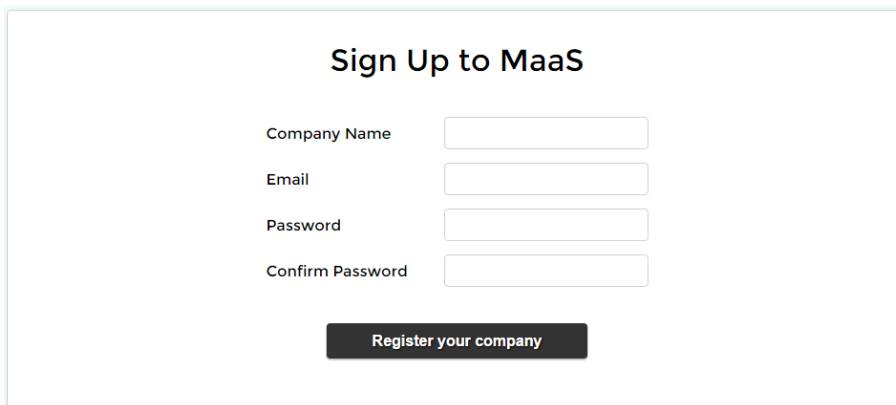


Figura 1: Central view of the signup page for a new Company Owner

2.2 Sign up - On invitation

This type of registration is only accessible through a link that will be sent by a company owner or administrator to the becoming user's email. Clicking on it will direct to a registration page in which the invitee shall type his username and password; he should then click on the confirmation button to make his choices effective.

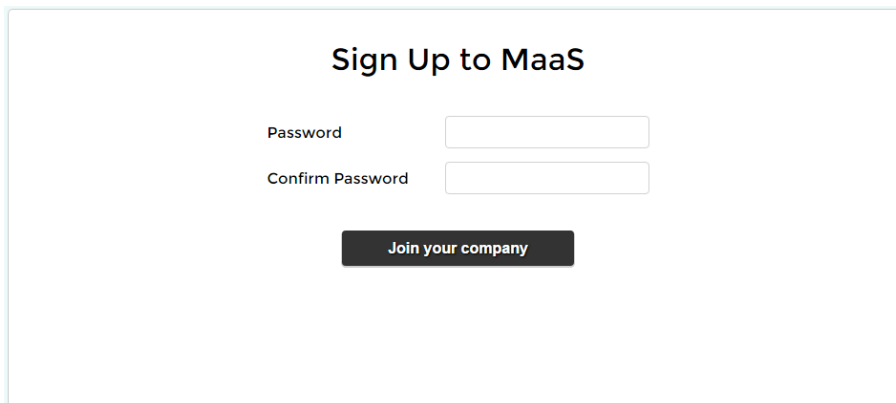
A screenshot of the 'Sign Up to MaaS' form. The form is centered on a light gray background. It has a title 'Sign Up to MaaS' in bold black text. Below the title, there are two input fields: 'Password' and 'Confirm Password'. Each field has a small icon on the left (a key for password and a checkmark for confirm). Below these fields is a dark gray button with white text that says 'Join your company'.

Figura 2: View of the signup page for an invited user

2.3 Login

Clicking on the "*Login_G*" button in the homepage will take the user to the authentication page. The correct access credentials must be inserted here to gain access to the personal account on the system. After typing-in the correct data, the user will have to click on the confirmation button and the system will direct him to his main dashboard.

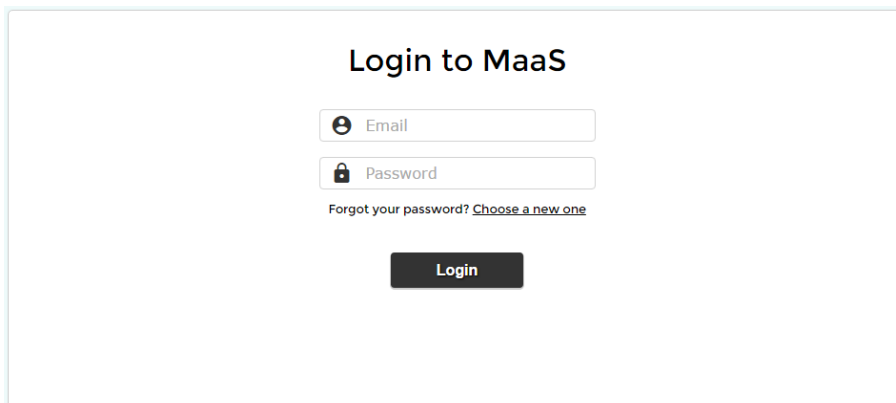
A screenshot of the 'Login to MaaS' form. The form is centered on a light gray background. It has a title 'Login to MaaS' in bold black text. Below the title, there are two input fields: 'Email' and 'Password'. Each field has a small icon on the left (an envelope for email and a key for password). Below these fields is a link that says 'Forgot your password? Choose a new one'. Below the link is a dark gray button with white text that says 'Login'.

Figura 3: View of the login page

2.4 Password recovery

This option is to be used only if the user has forgotten his access credentials. Clicking on the "*password recovery*" link under the login form will load a password recovery page, which contains a form in which the user shall insert his email (the one used for the registration process).

The next step is to click on the "*Send*" button: if the inserted email address corresponds to an account regularly registered to the system, a message will be sent to it, containing a link to a password recovery page.

After clicking the link the user will have access to the aforementioned page, which contains a form

to insert and confirm the new password. Tacking all the necessary steps as with the previous forms will change the user's password and save the new one in the company database.

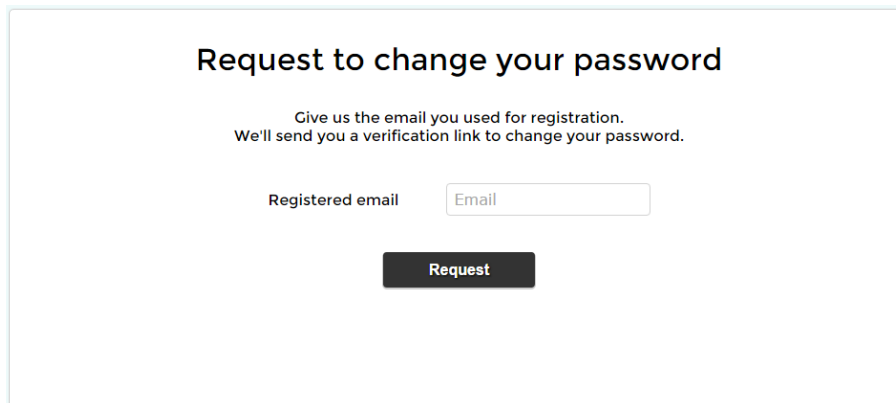


Figura 4: Central view of the password recovery page

3 Role-specific management

This section aims to explain some of the operations that can be performed exclusively by users having administrative privileges or above.

3.1 Company and User management

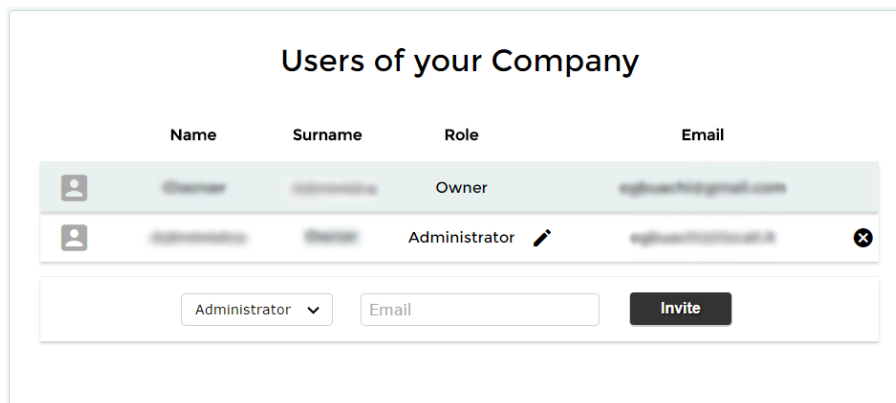


Figura 5: View in which the members of the Company are shown

After the log in, the Administrator or Owner can view a set of information regarding the Company to which he/she belongs, by clicking on the name of the company on the top bar of the system. Clicking on the "People" link on the left sidebar, a page showing a table which contains the list

of all the users of the Company shall be displayed. Each row represents a user, and for each of them two buttons can be found:

- A button in the "*Role*" column allows to modify the role of the corresponding user, giving him/her more or less privileges and capabilities;
- A button in the rightmost side of the table allows to delete the corresponding user from the database of the Company.

Beneath this table a little form is also displayed. This is the user invitation form, which requires the mail of the user that the Administrator or Owner intends to invite as a member of the Company.

This form also contains a drop-down menu, from which a role for the new user has to be selected. If all the data is set correctly, clicking the "*Invite*" button will send a join request to the specified email address. In this message a link can be found. Clicking on it the invited user will be directed to a particular sign up page in which he/she will choose his/her password to complete the subscription to the Company in the system.

Please note that a new Owner, having yet no member subscribed to is/her Company, won't see any user list. Instead a brief explanation on the invitation process shall be displayed.

It is also important to notice that the Owner cannot be canceled, neither can his role be changed.

The last option in the sidebar gives the user the possibility of deleting the entire Company from the MaaS system, it is available only to the Owner.

This process is not reversible and will also have as consequences:

- The deletion of all the users registered to the Company;
- The deletion of all the external data sources saved in the database of the Company;
- The deletion of all the DSL_G instruction sets belonging to the Company.

3.2 Database management

Manage databases

All

+
✕

<input type="checkbox"/>	Name <small>▼</small>	Status	
<input type="checkbox"/>	db1	●	↔ ✕
<input type="checkbox"/>	db2	●	↔ ✕
<input type="checkbox"/>	db3	●	↔ ✕
<input type="checkbox"/>	db4	●	↔ ✕

10 ▼

1

Figura 6: Main view of the database management page

Users having administrative privileges (or higher) can also manage the databases registered in the Company account.

The image above shows the main view of the database management page which displays, provided the Company has at least one external database registered in its account, the list of external data sources that can be used to query information through the DSL instruction sets.

The central column of the table, "*Status*", shows if the corresponding database is connected (green icon) or not (red icon). To change this state the user should just click on the button showing two arrows that can be found on the right column. The other button, next to it, allows the user to delete the relative database.

On the top of the page there are three other elements:

- A search bar to filter the results in the table through a specific name, useful when the number of external databases gets big;
- A button to add a new database to the system (see the *User Manual* for further information about this topic);
- A multi-delete button, that deletes all the databases selected in the table (through the checkboxes on the left side).

Another type of filtering can be performed using the sidebar, which allows the users to see

- All the databases;
- Only the connected databases;
- Only the disconnected databases.

3.3 DSL management

Manage your DSL definitions

All

+
✕

	Name	
<input type="checkbox"/>	cellOne	
<input type="checkbox"/>	cellTwo	
<input type="checkbox"/>	collectionOne	
<input type="checkbox"/>	DashboardOne	

10 ▾

1

Figura 7: View in which the DSL instruction sets are shown

The structure of the DSL management page is very similar to the previous one. On the center the user will find a table containing the list of the DSL instruction sets (henceforth described as DSLIS, DSLIS's for the plural). The left column shows the name of each DSLIS, while the right column contains three buttons for each row:

- An edit button to modify the corresponding DSLIS using the text editor provided by the system;
- An execution button to run the corresponding DSLIS and see the results it displays;
- A delete button to delete the corresponding DSLIS from the main database of the Company.

As before, on the top there are some other useful tools:

- A search bar to filter the results in the table through a specific name, useful when the number of DSLIS gets big;
- A button to add a new DSLIS to the main database of the Company (see the *User Manual* for further information about this topic);
- A multi-delete button, that deletes all the DSLIS's selected in the table (through the checkboxes on the left side).

Another type of filtering can also be performed using the sidebar, which allows the users to see

- All the DSLIS's;
- Only the DSLIS's that build *Dashboard_G*;
- Only the DSLIS's that build *Collection_G*;
- Only the DSLIS's that build *Document_G*;
- Only the DSLIS's that build *Cell_G*.

4 Language description

This section gives the user a thorough description of all the keywords and constructs of the system DSL.

First of all the main elements (Cell, Document, Collection and Dashboard) are described with all of their properties. These are the components that constitute the data visualization pages, building up the tabular display of the information queried from all the connected MongoDB databases.

Along with the details and descriptions some explanatory code snippets are also provided.

Then some functional keywords, used in the configuration of the main elements, are also explained.

Each component has two parts in its definition:

- **Identity:** this is the part of a component that is defined between round brackets ("()"). All the attributes (and possible functions) of this set are defined for each main element in the following sections;
- **Body:** this is the part of a component that is defined between curly brackets ("{}"). All the properties and sub-elements of this set are defined for each main element in the following sections.

Any other element-specific detail shall also be covered in the corresponding section.

4.0.1 Cell configuration

The `Cell` keyword defines the configuration of the Cell object of a table row. This is the smallest fundamental element of a DSL configuration, and its content is the first value of the result of a query. A Cell can also contain an image or a link. Its identity fields are:

- **name (required):** this attribute accepts a string. It contains the name of the MongoDB database column to which the query is going to make reference. When this is defined, alongside the `table` attribute, the `value` body property has to be left blank;
- **label (optional):** this attribute accepts a string. It contains the string that is going to be displayed as a title to the query result;
- **table (required):** this attribute accepts a string. It contains the name of the MongoDB database Collection to which the query is going to make reference. When this is defined, alongside the `name` attribute, the `value` body property has to be left blank;
- **type (required):** this attribute accepts a string. It contains the type of the data contained in the `value` body property or in the result of a query. Possible types are:
 - **Array:** an array of elements;
 - **Object:** a JSON object;
 - **Link:** a link to a specific resource;
 - **Image:** an image file;
 - **String:** a simple string;
 - **Number:** a simple number.

- **sortBy (optional)**: this attribute accepts a string. It contains the name of the MongoDB database column that the query will use as a reference to sort the result;
- **order (optional)**: this attribute accepts a string with value "asc" o "desc". It's the type of order for the sortBy parameter:
 - "asc" means that the order will be ascending (this is also the default value);
 - "desc" means that the order will be descending.
- **query (optional)**: this attribute accepts a JSON object. It contains the parameters and the values on which to execute the query;
- **transformation (optional)**: this attribute corresponds to the transformation function of an element. The function has to return a value that will overwrite the result of the query.
- **columnLabel (optional)**: this attribute contains the string that is going to be used as an alternate name to the MongoDB column.

Its body field is:

- **value (required)**: this parameter contains an arbitrary value to be shown as the result of the query. When this is defined the identity attributes **name** and **table** must be left blank;

The following code shows two syntax variations that allow the creation of valid Cell configurations.

Cell with value:

```
Cell (
  type: 'string',
  columnLabel: 'Name',
  label: 'User',
  transformation: function(val) {
    return val + " user2";
  }
){
  value: 'prova'
}
```

Cell without value:

```
Cell (
  name: 'name',
  type: 'string',
  table: 'users',
  columnLabel: 'Name',
  label: 'User',
  transformation: function(val) {
    return val + " user2";
  }
)
```



```
) {  
}
```

Two types of Cell:

- 1) Cell with value:
 Identity:
 type | required | "string"
 label | optional
 columnLabel | optional
 transformation | optional
 Body:
 value | required
- 2) Cell without value:
 Identity:
 name | required
 table | required
 label | optional
 transformation | optional
 columnLabel | optional
 type | required | "string"
 sortBy | optional
 order | optional | "asc"
 query | optional
 Body:
 empty

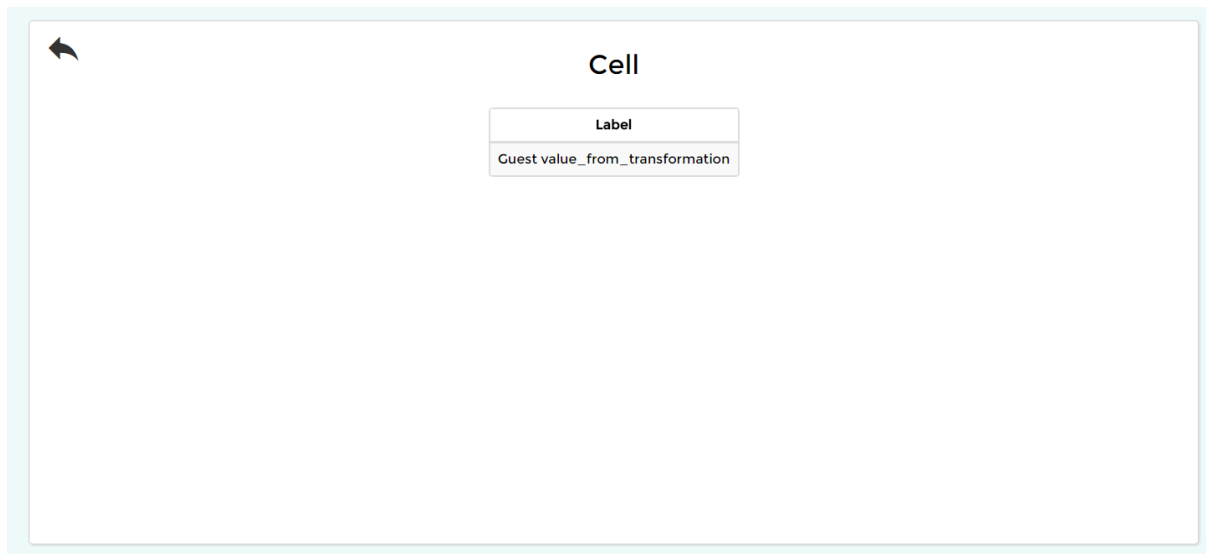


Figura 8: Example of the visualization page of a Cell query

4.0.2 Document configuration

The **Document** keyword defines the configuration of the Document object in a table. This is the element that most of all represents the concept of "document" as it is intended in a MongoDB database. Its identity fields are:

- **label (optional)**: this attribute accepts a string. It contains the string that is going to be displayed as a title to the query result;
- **table (required)**: this attribute accepts a string. It contains the name of the MongoDB database Collection to which the query is going to make reference while gathering the content of the Document;
- **sortby (optional)**: this attribute accepts a string. It contains the name of the MongoDB database column that the query will use as a reference to sort the result;
- **order (optional)**: this attribute accepts a string with value "asc" o "desc". It's the type of order for the sortby parameter:
 - "asc" means that the order will be ascending (this is also the default value);
 - "desc" means that the order will be descending.
- **query (optional)**: this attribute accepts a JSON object. It contains the parameters and the values on which to execute the query.

Its body fields are:

- **row (optional)**: this parameter accepts a string. It corresponds to the information of a row in the Document. Please view section 3.6.5 - *Row configuration* for further explanation about this field;



- **action (optional):** this parameter accepts a string. It contains the actions that can be invoked from the Document. Please view section 3.6.7 - *Action configuration* for further explanation about this field.

The following code shows an example of Document configuration and two syntax variations that allow the creation of valid Documents.

```
-----  
  
Document (  
    table: 'users',  
    label: 'Users',  
    sortby: 'surname',  
    order: 'asc',  
    query: {age : { $lt : 40}}  
) {  
    row(  
        name: 'surname',  
        label: 'Surname',  
        type: 'string'  
    )  
    row(  
        name: 'name',  
        label: 'Name',  
        type: 'string'  
    )  
    row(  
        name: 'orders',  
        label: 'Orders',  
        type: 'number',  
        transformation: function(val) { return val.length; }  
    )  
}
```

two types of independent Document:

```
1)  
    Identity:  
        table | required  
        label | optional  
        sortby | optional  
        order | optional | asc  
        query | optional  
    Body:  
        row  
        action | optional  
Row:  
    Identity:  
        name | required
```

```
label | optional
transformation | optional
type | required
```

2)

Identity:

```
table | required
label | optional
sortby | optional
order | optional | asc
query | optional
```

Body:

```
action | optional
```

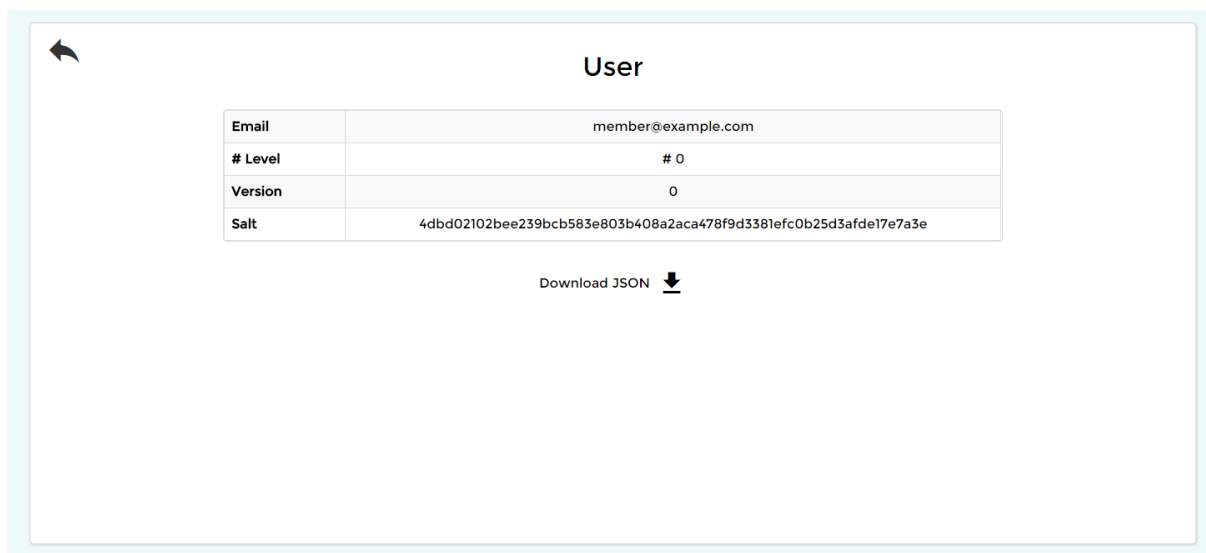


Figura 9: Example of the visualization page of a Document query

4.0.3 Collection configuration

The **Collection** keyword creates a new Collection object. The way information shall be displayed in a visualization page will be derived from it. Its identity fields are:

- **label (optional)**: this attribute accepts a string. It contains the string that is going to be displayed as a title to the query result;
- **table (required)**: this attribute accepts a string. It contains the name of the MongoDB database Collection to which the query is going to make reference;



- **sortby (optional)**: this attribute accepts a string. It contains the name of the MongoDB database column that the query will use as a reference to sort the result;
- **order (optional)**: this attribute accepts a string with value "asc" o "desc". It's the type of order for the sortby parameter:
 - "asc" means that the order will be ascending (this is also the default value);
 - "desc" means that the order will be descending.
- **query (optional)**: this attribute accepts a JSON object. It contains the parameters and the values on which to execute the query.

Its body fields are:

- **column (optional)**: this parameter accepts a string. It corresponds to the information of a column in the Collection. Please view section 3.6.6 - *Column configuration* for further explanation about this field;
- **action (optional)**: this parameter accepts a string. It contains the actions that can be invoked from the Collection. Please view section 3.6.7 - *Action configuration* for further explanation about this field.

The following code shows an example of Collection configuration and some syntax variations that allow the creation of valid Collections. The syntax for the creation of internal Documents (which are constituted inside another Collection) is also shown.

```
-----  
  
Collection(  
    table: "customers",  
    label: "JuniorCustomers",  
    -----id: "Junior",  
    -----Weight:"0",  
    perpage: "20",  
    sortby: "surname",  
    order: "asc",  
    query: {age: {$lt: 40}}  
) {  
    column(  
        name: "3"  
    )  
    action(  
        Export: "true",  
        SendEmail: "true"  
    )  
    column(  
        name: "4"  
    )  
    Document(  
        table: "prova"  
    ){  
        row(  

```



```
        name: "asd"
    )
    action(
        SendEmail: "true"
    )
}
column(
    name: "5"
)
}
```

Two types of Collection:

1)

```
Identity:
    table | required
    label | optional
    sortby | optional
    query | optional
    order | optional
    perpage | optional
Body:
    action | optional
    column | optional
```

2)

```
Identity:
    table | required
    label | optional
    sortby | optional
    query | optional
    order | optional
    perpage | optional
Body:
    action | optional
    column | optional
    document
```

```
Column:
    name | required
    selectable | optional | false
    sortable | optional | false
    type | required | false
    label | optional
    transformation | optional
```

Two types of Document inside a Collection (if defined):

1)

```

Identity:
  populate | optional
Body:
  row
Row:
  Identity:
    name | required
    label | optional
    transformation | optional
    type | required

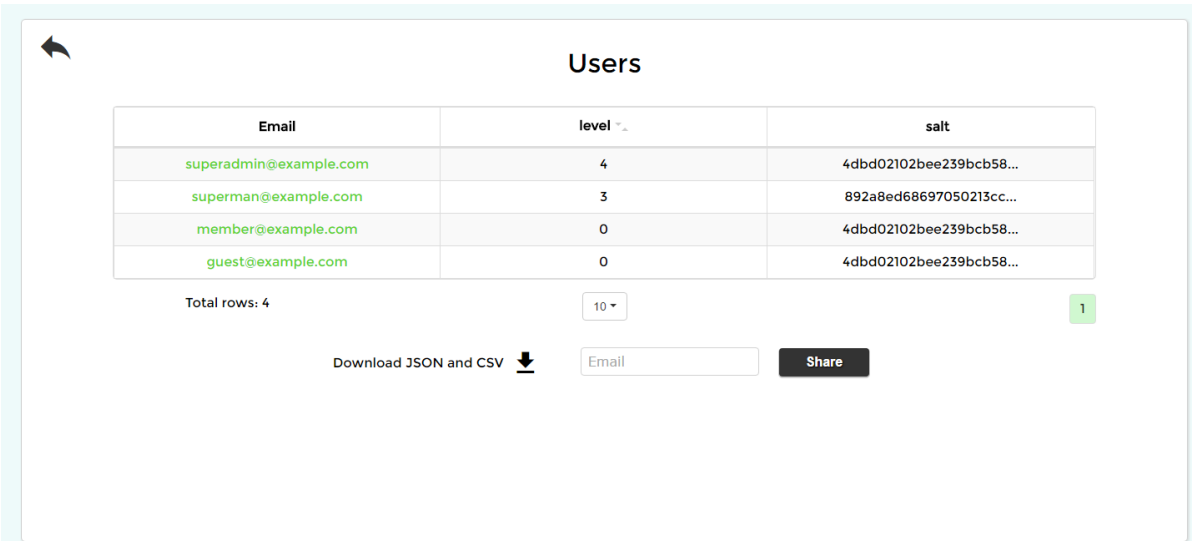
```

2)

```

Identity:
  populate | optional
Body:
  empty

```



Email	level	salt
superadmin@example.com	4	4dbd02102bee239bcb58...
superman@example.com	3	892a8ed68697050213cc...
member@example.com	0	4dbd02102bee239bcb58...
guest@example.com	0	4dbd02102bee239bcb58...

Total rows: 4

10

Download JSON and CSV

Email

Share

1

Figura 10: Example of the visualization page of a Collection query

4.0.4 Dashboard configuration

The **Dashboard** keyword creates a new Dashboard object. All the other configurations will derive from it. The fundamental unit of a Dashboard is the row, just like in a table structure. In each row various objects can be placed (apart from other Dashboards), each inserted object defines a column in the row. The Dashboard, if used, must be the first element of a DSL configuration. Its identity field is:



- **label (optional)**: this attribute accepts a string. It contains the string that is going to be displayed as a title to the Dashboard view;

Its body field is:

- **row (optional)**: this is a special parameter, different from the **row** described in section 3.6.5 - *Row configuration*. It corresponds to the information that has to be displayed in a row of the Dashboard and its content has to be defined through the language components:
 - **Cell**;
 - **Document**;
 - **Collection**.

The following code shows an example of Dashboard configuration and the syntax that allows the creation of valid Dashboards.

```
Dashboard(  
  label: "Dashboard"  
)  
{  
  row(  
    Document(  
      table: "prova"  
    )  
    {  
      row(  
        name: "email",  
        type: "string",  
        label: "Email"  
      )  
    }  
    Document(  
      table: "prova"  
    ){  
  }  
)  
  row(  
    Document(  
      table: "prova"  
    ){  
  }  
    Collection(  
      table: "users"  
    ){  
  }  
    Cell(  
      type: "string"  
    ){
```



```
    }
  )
}
```

Dashboard structure:

```
Dashboard(
  label | optional
){
  row | optional
}

Row(
  Cell() | optional
  Document() | optional
  Collection() | optional
)
```

The screenshot shows a dashboard titled "Dashboard" with a back arrow in the top left. The dashboard contains several sections:

- Prova**: A form with an "Email" field containing "superadmin@example.com".
- Doc**: A form with fields for "customer", "product", and "quantity".
- Users**: A table with columns "Email", "level", and "salt". It shows two rows: "superadmin@exa" with level 4 and salt "4dbd02102bee235", and "superman@exam" with level 3 and salt "892a8ed6869705". Below the table, it says "Total rows: 2".
- valore**: A form with a "prova cell" field.
- Table**: A table with columns "fullname", "avatar", "city", "email", "orders", and "age". It contains three rows of data:

fullname	avatar	city	email	orders	age
Adrienne Pitts	http://www.filltext.com/ima	Danbury	JNacita@pharetra.org	2	53
Minsoo Almanza	http://www.filltext.com/ima	Shorewood	RPsarros@tempor.org	9	16
Mayra Chenevert	http://www.filltext.com/ima	Cape Coral	DMauro@sed.org	8	62

Figura 11: Example of the visualization page of a Dashboard query

4.0.5 Row configuration

The **row** keyword defines the configuration of a table row of the visualization page. Its identity fields are:

- **name (required)**: this parameter accepts a string. It is the name of the reference row of a Collection in the MongoDB database, from which this Row is filled;
- **type (required)**: this parameter accepts a string. It specifies the type of data defined in a specific row, which can also be the result of a query. Possible types are:
 - **Array**: an array of elements;
 - **Object**: a JSON object;
 - **Link**: a link to a specific resource;
 - **Image**: an image file;
 - **String**: a simple string;
 - **Number**: a simple number.
- **label (optional)**: this parameter accepts a string. It is the header name of the table row of the visualization page. If not specified it is automatically set to the value of the **name** parameter;
- **transformation (optional)**: this parameter corresponds to the transformation function of an element. The function has to return a value that will overwrite the result of the query.

4.0.6 Column configuration

The **column** keyword defines the configuration of a table column of the visualization page. Its identity fields are:

- **name (required)**: this parameter accepts a string. It is the name of the reference collection in the MongoDB database;
- **type (required)**: this parameter accepts a string. It specifies the type of data defined in a specific row;
- **label (optional)**: this parameter accepts a string. It is the header name of a table column. If not specified it is automatically set to the value of the **name** parameter;
- **sortable (optional)**: this parameter accepts a boolean. It specifies whether or not a table could be sorted on the basis of this column. If not specified it is set to **false**;
- **selectable (optional)**: this parameter accepts a boolean. It specifies whether or not the element could be a link which redirects to a related document. If not specified it is set to **false**;
- **transformation (optional)**: this parameter corresponds to the transformation function of an element. The function has to return a value that will overwrite the result of the query.

4.0.7 Action configuration

The **action** keyword defines a specific operation that can be executed on a specific main element of the DSL. Its identity fields are the functions that the user may want to run on an element. These functions are defined as:

- **Export (optional)**: if enabled this function allows the user to export the result of the query performed by a specific main element of the DSL. The default activation value is set to **false** (which means that the function is disabled), the other activation values are defined below;
- **SendEmail (optional)**: if enabled this function allows the user to send to his own mail an exported result of the query performed by a specific main element of the DSL; as a consequence the action performed by this function also implies the execution of the Export function. The default activation value is set to **false** (which means that the function is disabled), the other activation values are defined below.

Both functions accept some particular values called *activation values*, which specify if the function is enabled and what type of result their execution must produce. These values are:

- **"false"** means that the corresponding function is disabled, thus cannot be executed;
- **"true"** means that the corresponding function is enabled. The function enabled through this value will produce two results: a JSON output file and a CSV (comma-separated values) output file;
- **"json"** means that the corresponding function is enabled. The function enabled through this value will produce only a JSON output file;
- **"csv"** means that the corresponding function is enabled. The function enabled through this value will produce only a CSV output file.

Glossary

Browser Also called Web browser. Digital Technology. a software program that allows the user to find and read encoded documents in a form suitable for display, especially such a program for use on the World Wide Web. 3

Business-men People regularly employed in business, especially white-collar workers, executives, or company owners. 3

Cell With regard to the MaaS project, the term identifies a system element whose function is to display a single value, which may be of type String, Numeric, Link, or Image Data. It can take as input the result of a query or any arbitrary value. In the first case, if the result is a set of values, then the one shown is the first (the results can be ranked according to an attribute field). 9

Collection With regard to the MaaS project, the term refers to one of the elements of the system, which has the task of showing a list of documents extracted from a DSL program in tabular format. A Collection is identified by a name and a label, and allows you to define which subsets of documents you want to display. 9

Dashboard Relatively to the MaaS project, this term identifies a layout grid defined by said row sub-elements, which correspond to the rows of the structure. The number of columns in each row is implicitly defined by the number of elements it contains. 9

Database A comprehensive collection of related data organized for convenient access, generally in a computer. 3

Document With regard to the MaaS project, the term indicates an element that specifies how to display a single document in a Collection. It can however be stated independently of other entities. 9

DSL *Domain specific Language*, in software development and domain engineering it indicates a programming language (or a specification language) dedicated to the particular problems of a domain, to a particular technique of representation and/or in a particular technical solution. MaaS The project incorporates a DSL built for structuring queries for viewing data contained in a company registered in the system database. 7

Login In computer security, logging in, is the process by which an individual gains access to a computer system by identifying and authenticating themselves. The user credentials are typically some form of "username" and a matching "password", and these credentials themselves are sometimes referred to as a login, (or a logon or a sign in or a sign on). 5

Webmaster A webmaster (from web and master), also called a web architect, web developer, site author, website administrator, website coordinator, or website publisher is a person responsible for maintaining one or many websites. The duties of the webmaster may include: ensuring that the web servers, hardware and software are operating correctly, designing the website, generating and revising web pages, A/B testing, replying to user comments, and examining traffic through the site. As a general rule, professional webmasters "must also be well-versed in Web transaction software, payment-processing software, and security software". 3