User 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 first approach to the MaaS system, developed by MINT. It is therefore advisable to follow it as a step-by-step guide, covering each chapter in the written order.

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	02-09-2016	Tommaso Zagni (Administrator)	images upload
1.0.1	01-09-2016	Michael Ogbuachi (Person in charge)	several additions - draft.
1.0.0	02-08-2016	Michael Ogbuachi (Person in charge)	document approval.
0.1.1	02-08-2016	Fabiano Tavallini (Verifier)	images upload.
0.1.0	01-08-2016	Fabiano Tavallini (Verifier)	document verification.
0.0.2	29-07-2016	Tommaso Zagni (Administrator)	Chapter 3 - draft.
0.0.2	28-07-2016	Tommaso Zagni (Administrator)	Chapter 2 - draft
0.0.1	26-07-2016	Tommaso Zagni (Administrator)	Chapter 1 - draft

Contents

List of Figures

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_G 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

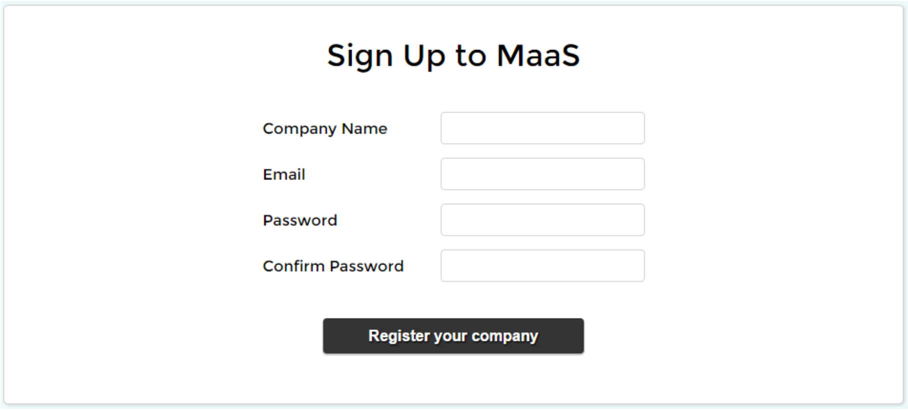
This section discusses all the steps needed to create an account and authenticate to the system. It is important to notice one important fact though. There are two different types of user registration to the system:

- **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.


Sign Up to MaaS

Company Name	
Email	
Password	
Confirm Password	

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 ivitee shall type his username and password;he should then click on the confirmation button to make his choices effective.

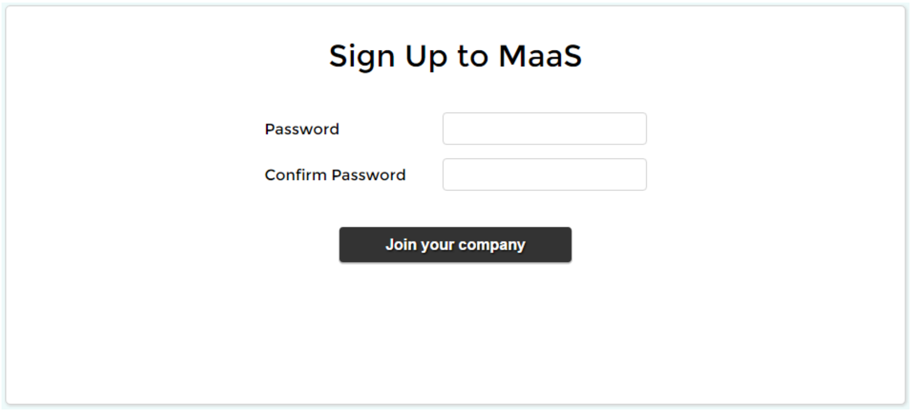


Figura 2: View of the signup page for an invited user

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 sistemm wil direct him to his main *Dashboard$_G$*.

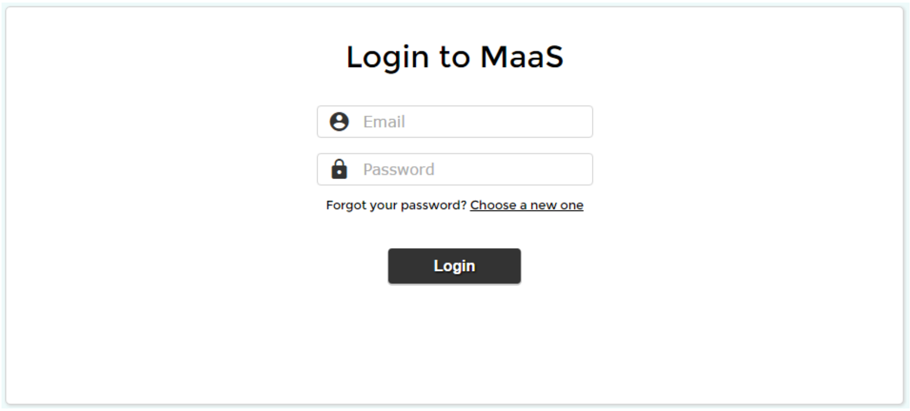
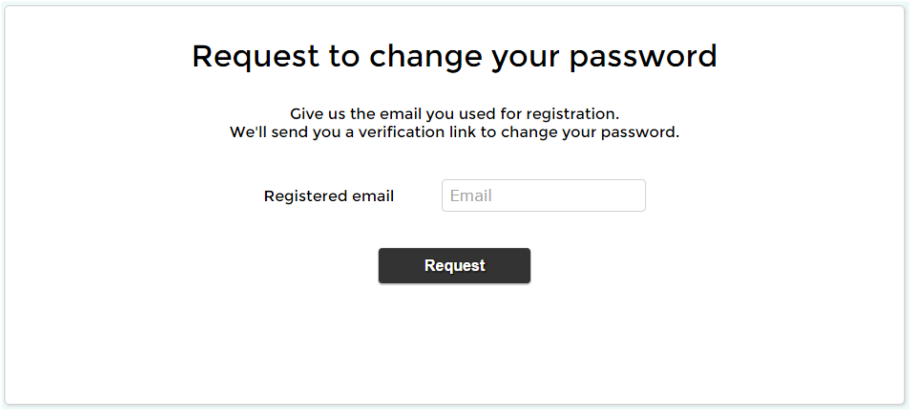


Figura 3: View of the login page

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.


Request to change your password

Give us the email you used for registration.
We'll send you a verification link to change your password.

Registered email: Email

Figura 4: Central view of the password recovery page

3 Interacting with the interface of the system

Once the user has successfully completed the log in procedure, he will find some changes in the main page of the system. In fact new elements shall show up on the horizontal dark bar on the top.

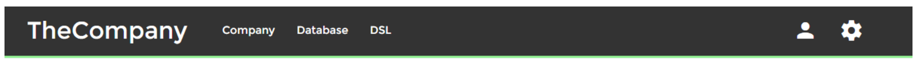


Figura 5: Top bar of all the pages after login

All those are useful links to access the various features MaaS offers. Ordered left to right these elements are:

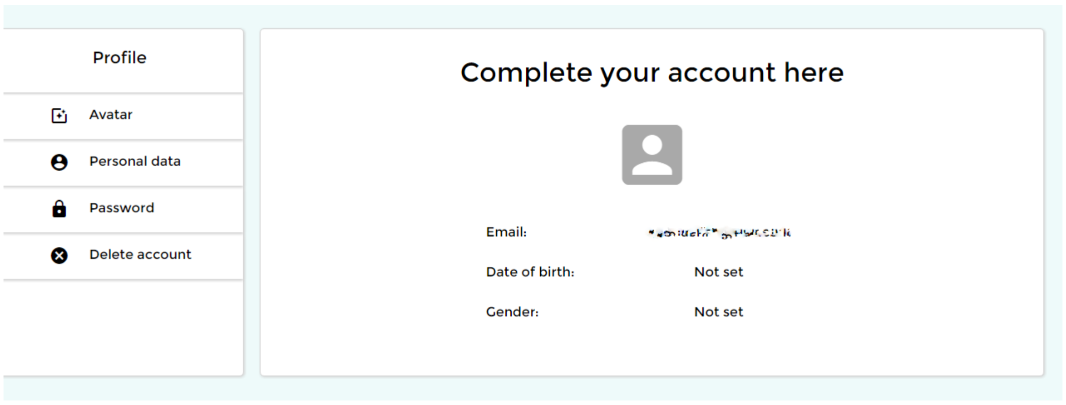
- Name of the affiliated Company (it also serves as a link to the user's home page);
- Link to the page showing information about the affiliated Company;
- Link to the database management page;
- Link to the DSL_G management page;
- Icon serving as a link to the personal profile page;
- Icon showing (on click) a drop down menu for settings and log out

Another change in the home page shall affect its central body, showing the main Dashboard (hereinafter referred to as *"Active Dashboard"*), which shall be capable of displaying the list of $Collection_G$ containing various types of information and data queried by the user.
These data rely on the $MongoDB_G$ databases that will be connected to the system. By clicking on any of the available Collections (provided at least one Collection has been stored through an

adequate DSL) the user will be sent to a view page containing other elements that make up the data visualization, such as *Document*$_G$ and *Cell*$_G$.
From this Active Dashboard page the user will be able to perform other operations.

3.1 Personal profile

By clicking on the user icon on the right side of the upper bar the user will be taken to a page that will show him his profile, besides giving him the possibility to change his personal data.
In the image below the structure of the page can be seen. The center block at first describes some basic information about the user, while on the left side a vertical menu provides some options that (on click) will change the content of the central area.



Complete your account here

Email: [illegible]
Date of birth: Not set
Gender: Not set

Figura 6: Main view of the user profile

This menu contains five links.

- **Profile**: shows the default profile view page, which contains some basic information about the user;
- ***Avatar***$_G$: shows a page that allows the user to change his avatar. Through the form the user can easily perform this operation, he/she should just drag a new image in the provided space and click on the "Change Avatar" button to save the new setting;

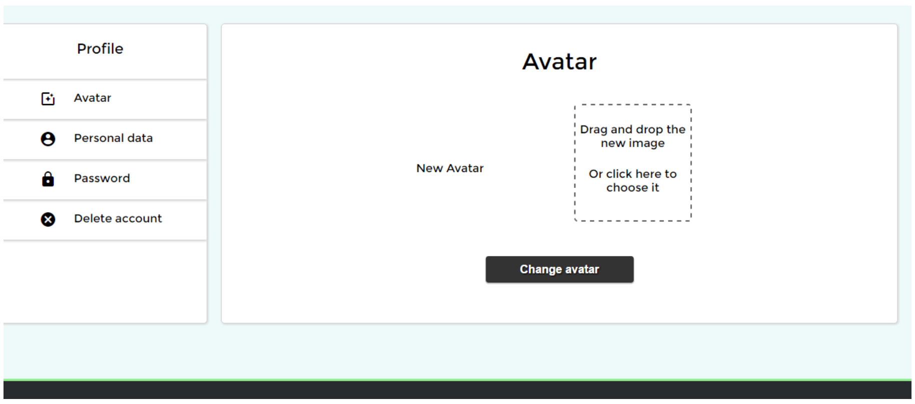


Figura 7: Change user avatar

- **Personal data**: shows the page that allows the user to display and edit his/her personal data. Information such as the name, surname, date of birth and gender can be set; to save the changes the user must click on the "Save changes" button;

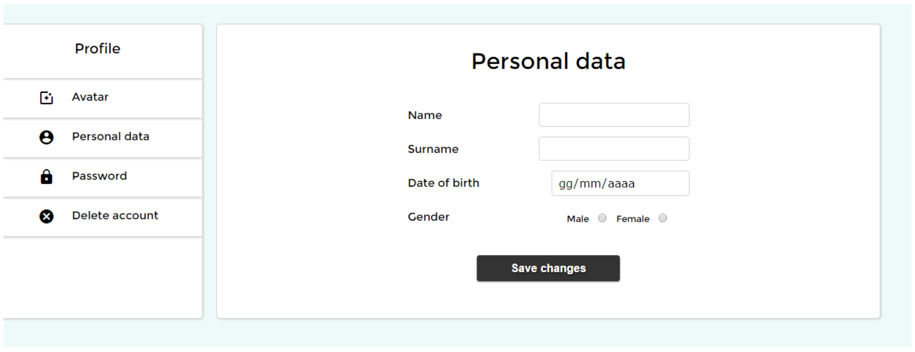


Figura 8: Edit user's personal data

- **Password**: displays a form through which the user can change the password he/she uses to authenticate to the system. The form is made of two input fields, the first takes in the new password, while the second is used to confirm the correctness of the content of the first field. By clicking the "Set password" button the user accepts and saves the changes;

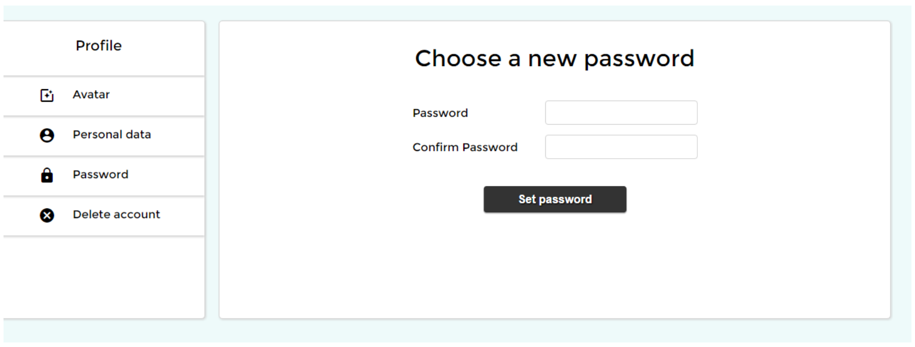
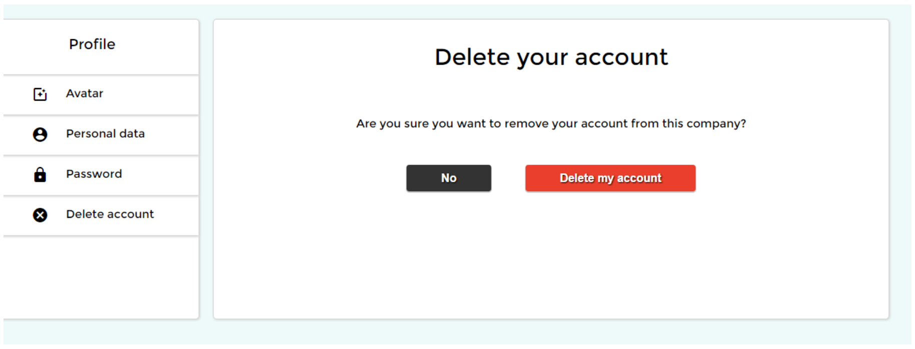


Figura 9: Change the user's login password

- **Delete account**:shows a page from which the user can delete his/her personal account. It is sufficient to click on the red "Delete my account" button to carry on the deletion, or on the "No" button to go back to the default visualization page. It is important to notice that **the deletion of an account is not a reversible operation**.



Are you sure you want to remove your account from this company?

Figura 10: Delete user's account

3.2 Settings

The gear icon on the rightmost side of the dark upper bar is a button which, if clicked, opens a small drop-down menu containing three options.

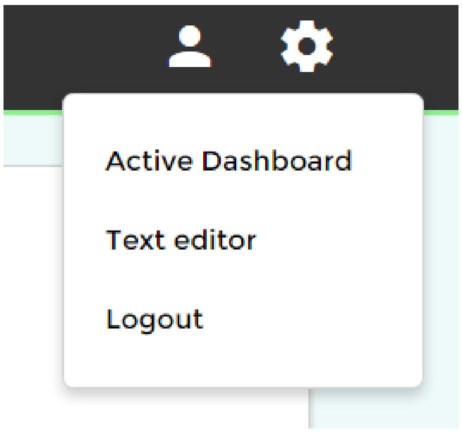
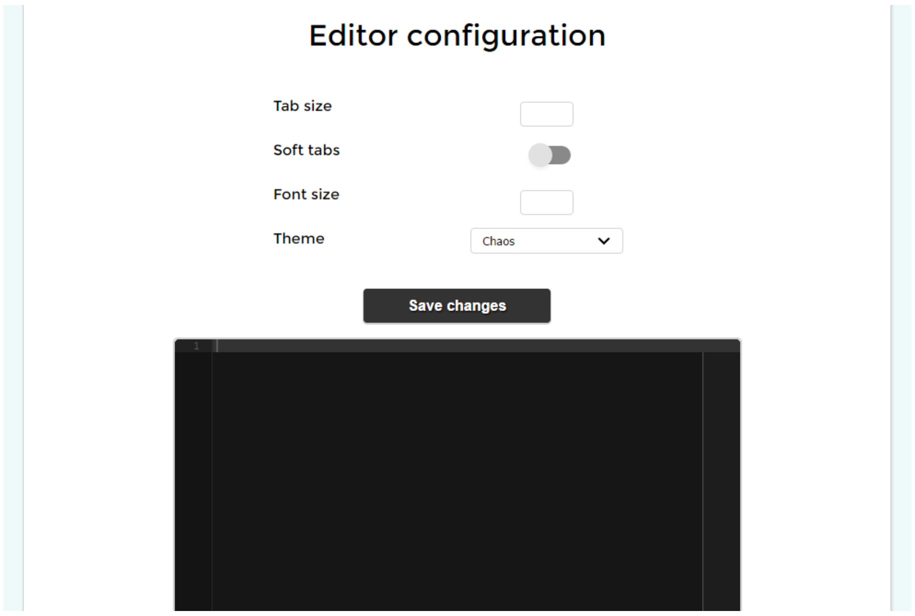


Figura 11: Menu that appears by clicking on the settings button

- **Active Dashboard**: when clicked, this icon takes the user to the page in which he/she can choose an Active Dashboard or edit the current one (if there's any);
- **Editor configuration**: when clicked, this link takes the user to a page containing a form, through which it is possible to change the settings of the text editor;


Editor configuration

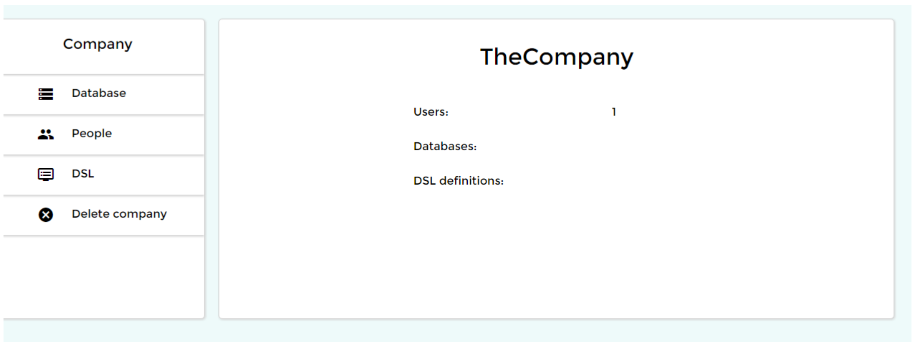
Tab size	
Soft tabs	
Font size	
Theme	Chaos

Figura 12: Change the configuration of the DSL editor

- ***Logout***$_G$: this button simply performs, if clicked, all the necessary steps to log out successfully from the current user account. This means that clicking on it results in the current user session being cleared and all the changes that weren't saved being lost.

3.3 Company information

By clicking on the Company name near the top left corner of the page the user shall be directed to a Company visualization page. This by default displays basic information about the affiliated Company, such as the number of subscribed users, the number of databases and the number of DSL code files.

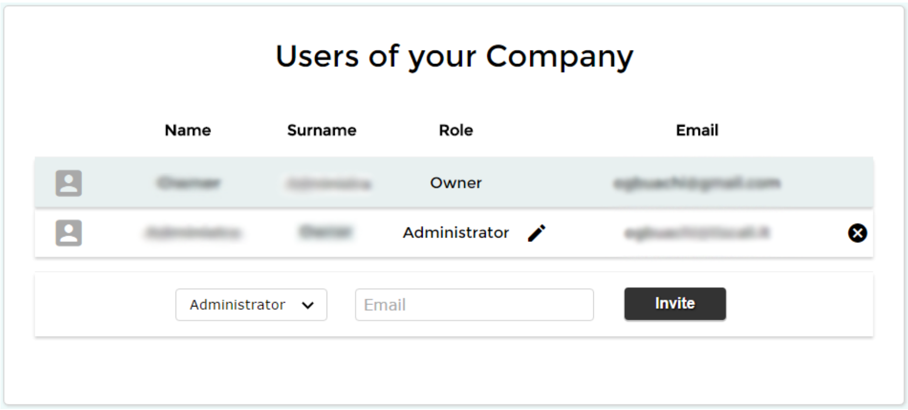


Users:	1
Databases:	
DSL definitions:	

Figura 13: Show some basic information about the Company

On the left a vertical pane can be found, containing some other links to features of the system.

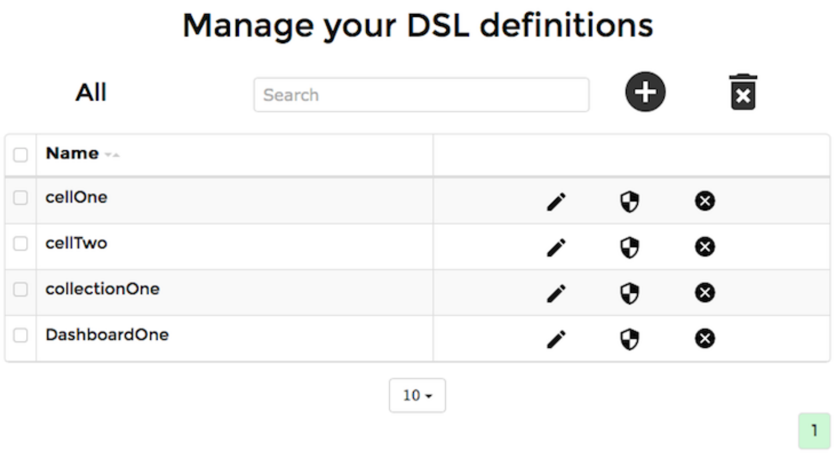
- **Database**: this link directs to the Database management page (later explained);
- **People**: shows a page containing the list of subscribed users and a form to invite a new member to the Company. Most of the functions of this page are available only to the Administrators and Owner of the Company. The user should refer to the *Administrator Manual* as the case may be he falls in one of these categories;



Name	Surname	Role	Email
[illegible]	[illegible]	Owner	[illegible]
[illegible]	[illegible]	Administrator	[illegible]

Figura 14: View in which the members of the Company are shown

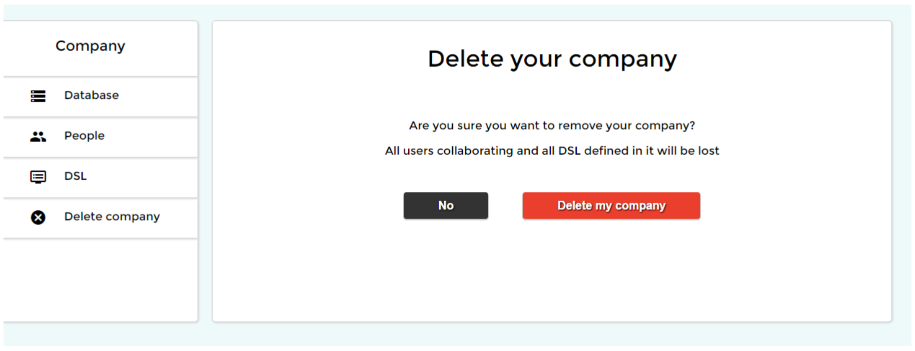
- **DSL**: shows a page on which a text editor is displayed, giving the user the possibility of writing his/her own DSL files to query information from the Company databases. As a result, from this page the user can also save various types of queries like new Dashboards, Collections, Documents and Cells. The user can also edit, delete, view or execute already existing DSL files, through the buttons in the page. Some of the functions of this page are not available to the Guests of the Company.



Name	
cellOne	
cellTwo	
collectionOne	
DashboardOne	

Figura 15: View in which the DSL instruction sets are shown

- **Delete company**: shows a page from which the Owner can delete his/her Company. It is sufficient to click on the red "Delete my company" button to carry on the deletion, or on the "No" button to go back to the default visualization page. It is important to notice that **the deletion of a Company is not a reversible operation** and that **all the data belonging to the deleted Company shall be lost if not saved elsewhere**.



Delete your company

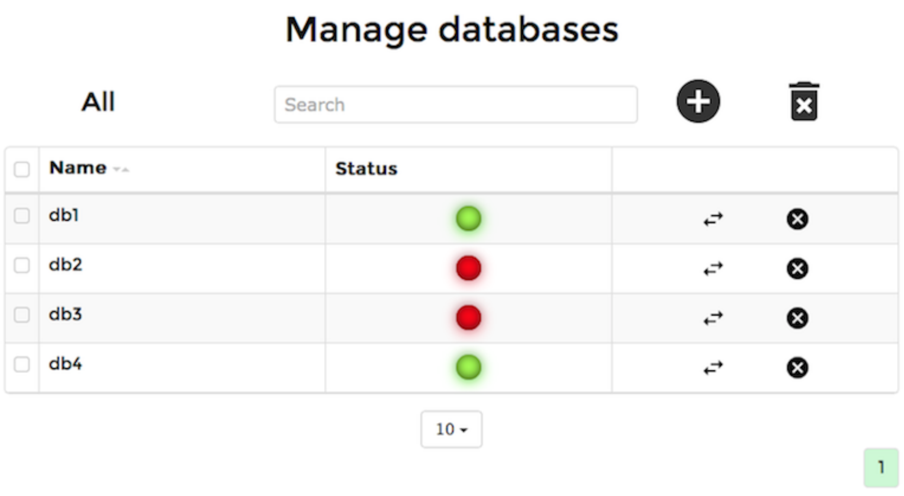
Are you sure you want to remove your company?

All users collaborating and all DSL defined in it will be lost

Figura 16: Delete the current Company (only the Owner is allowed to do this)

3.4 Database management

At this point in time there are two ways to access the database management page. The first (and most immediate) way is by clicking on the "Database" link on the top bar of each page, while the second consists in accessing it from the left pane of the Company page.



Name	Status		
db1	🟢	⇄	⊗
db2	🔴	⇄	⊗
db3	🔴	⇄	⊗
db4	🟢	⇄	⊗

Figura 17: Main view of the database management page

The user should notice that most of the functionalities offered by this page are role-dependent, therefore a Guest will not be able to use them effectively. Once entered in the page the user shall see a table containing the name and status of each database owned by the Company. The word "status" here means the property of a database of being connected or disconnected.
There are also three buttons above the aforementioned table:

- **Add database** (+ button): allows the user to add a new MongoDB database to the data sources of his/her Company. Clicking on this button will show a new section in the page, which contains a form comprised of three input fields and a button. To add a new database the user must fill the form inserting the database name, password and connection string. The next step is to click on the "Confirm" button, the new database shall be automatically inserted in the table and allowed for the utilization. Is is important to notice that the connection string must be a valid one, pointing to a truly existing database provided by some MongoDB *Storage Host*$_G$;

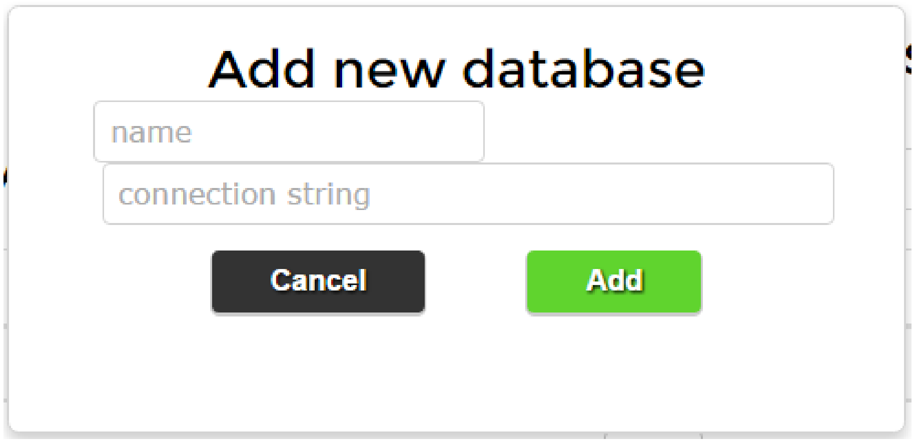


Figura 18: Add a new database to the Company

- **Delete database**: allows the user to delete one or more databases from the Company data sources. First of all the databases that have to be deleted must be selected through the checkboxes located on the leftmost column of the table. The user then has to click on the "Delete database" button in order to complete the operation. It is important to notice that **this operation is not reversible**;
- **Enable/Disable**: this button allows the user to set the status of the database to be enabled or disabled, switching the value of the property accordingly. As in the previous operation, the user must first select the databases to change in the table, by clicking on the corresponding checkboxes.

3.5 DSL management

This page embodies all the necessary instruments to write, view, edit and execute new DSL query files.
(Please see the section *Company information - DSL* for further explanation about the main DSL management page).

Clicking on the + button to add a new DSL instruction set will open another page containing a text editor.
The main part of the page is constituted by the text editor on the center, which allows all the writing and editing process.

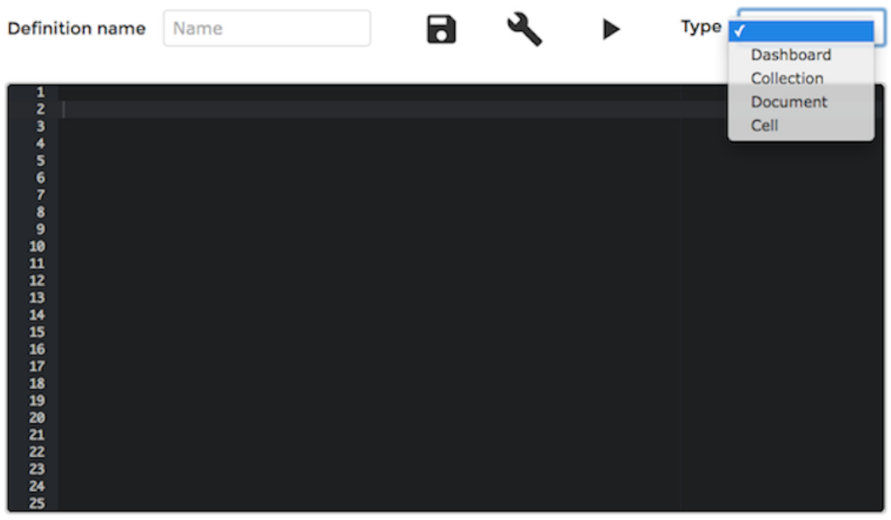


Figura 19: Main view of the editor page

When the server starts, the system reads the directory and sequentially get all the files with extension .dsl inside it. The DSL interpreter parses these files and interfaces with the API_G of MaaS to generate all the classes and the models required to configure the given collections.

If during this process some errors occur about the file interpretation and the execution of the code, they will be reported in the application. Each expression accepts lists of parameters similarly to the *JavaScritp*$_G$ *JSON*$_G$ style, like: `parameterName: parameterValue`. Some parameters are required and others are optional, which are set to default values if none is given.

3.6 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.

3.6.1 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.

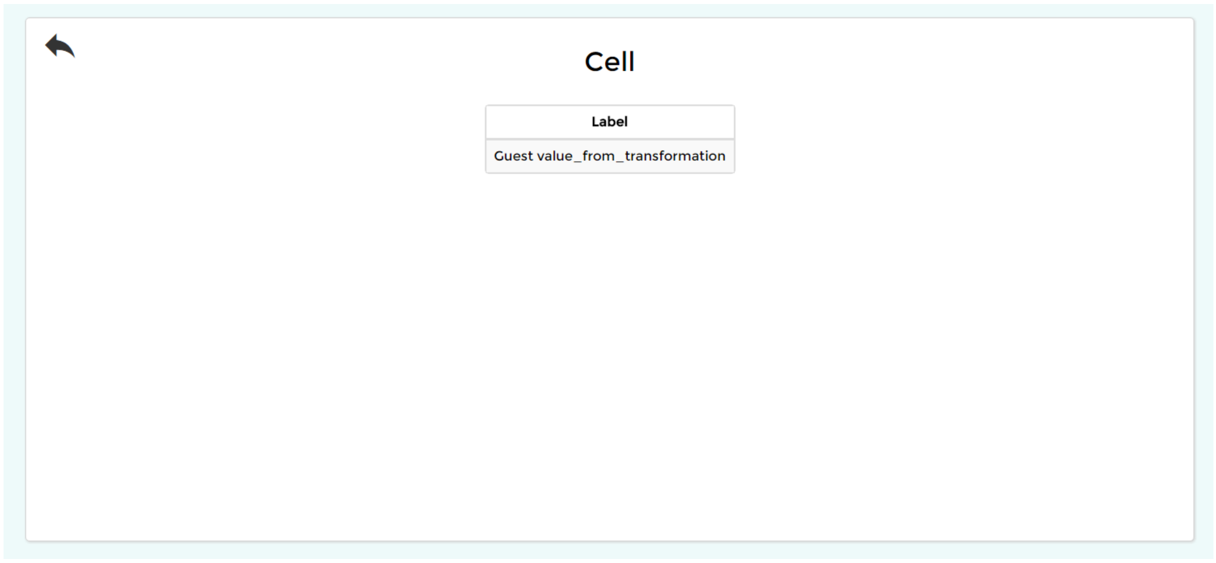
- **sortby (optional)**: this attribute accepts a string. It contains the name of the MongoDB database column that the query will use as a reference to sort the result;
- **order (optional)**: this attribute accepts a string with value "asc" o "desc". It's the type of order for the sortby parameter:
  - **"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

-------------------------------------------------------------------
```



Label
Guest value_from_transformation

Figura 20: Example of the visualization page of a Cell query

3.6.2 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.

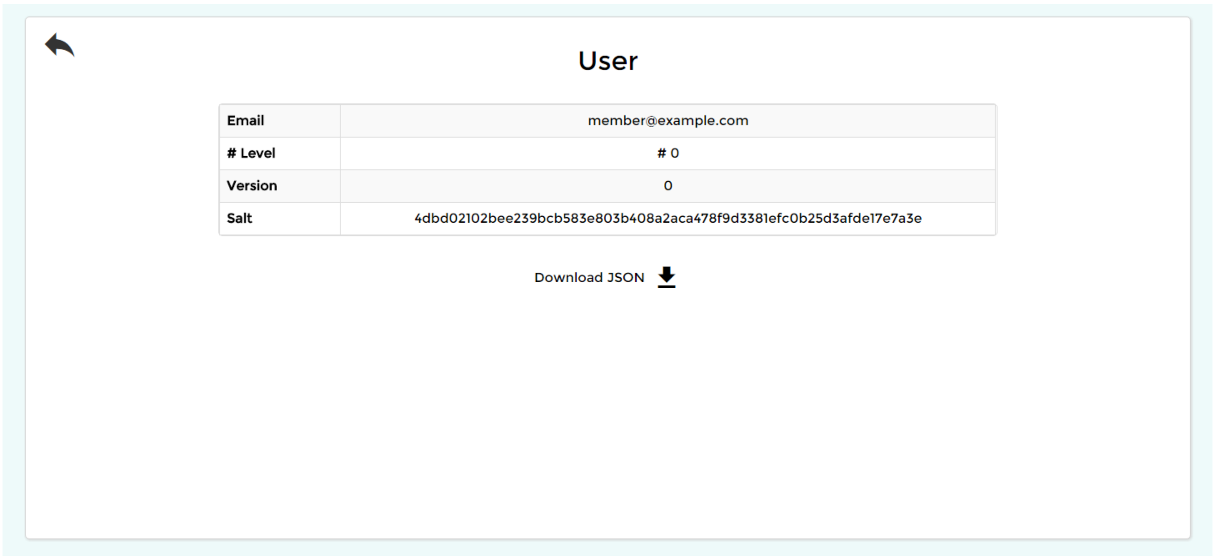
Its body fields are:

- **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

-------------------------------------------------------------------
```



User

Email	member@example.com
# Level	# 0
Version	0
Salt	4dbd02102bee239bcb583e803b408a2aca478f9d3381efc0b25d3afde17e7a3e

Figura 21: Example of the visualization page of a Document query

3.6.3 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.

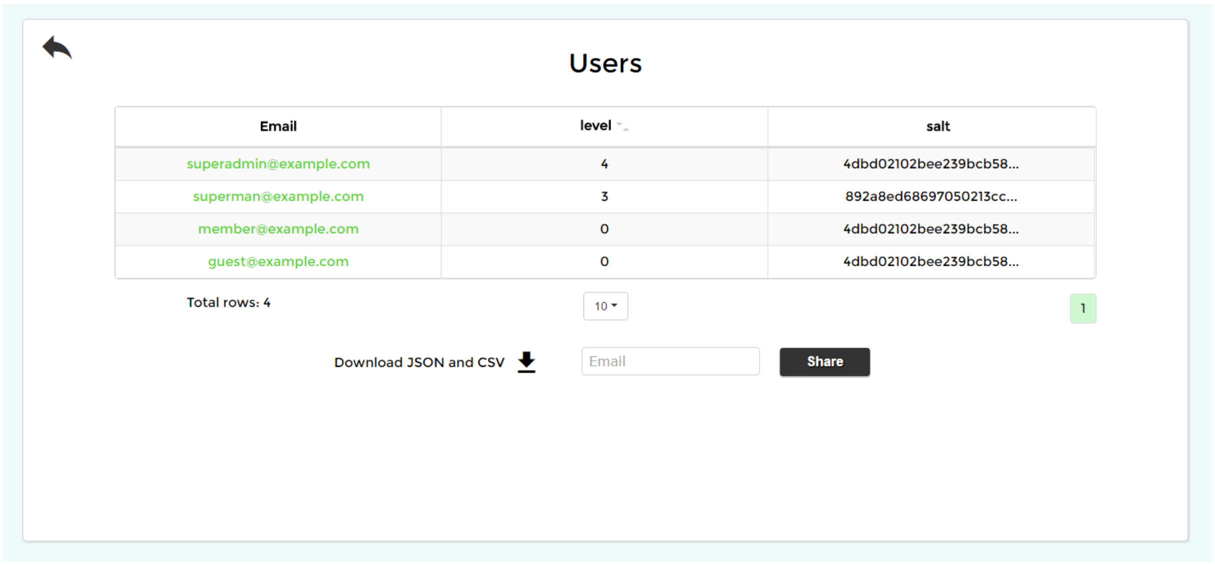
Its body fields are:

- **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...

Figura 22: Example of the visualization page of a Collection query

3.6.4 Dashboard configuration

The `Dashboard` keyword creates a new Dashboard object. Al 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:

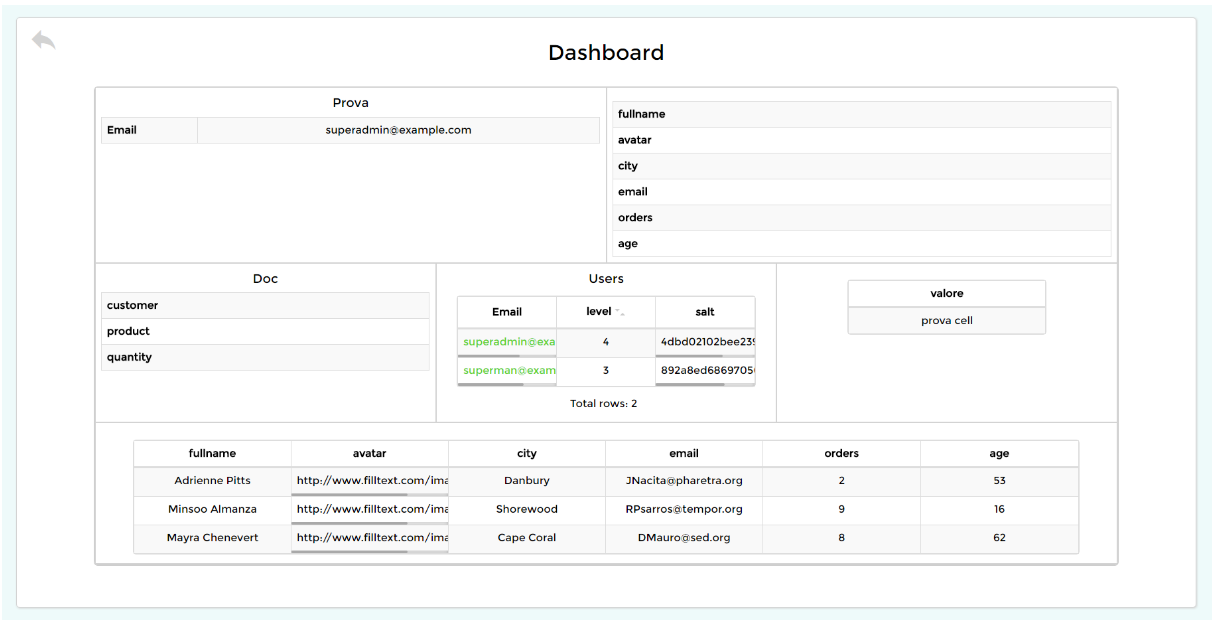
- **label (optional)**: this attribute accepts a string. It contains the string that is going to be displayed as a title to the Dashboard view;

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
    )

-------------------------------------------------------------------
```



Dashboard

Prova

Email	superadmin@example.com

fullname
avatar
city
email
orders
age

Doc

customer
product
quantity

Users

Email	level	salt
superadmin@exa	4	4dbd02102bee23
superman@exam	3	892a8ed6869705

Total rows: 2

valore
prova cell

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 23: Example of the visualization page of a Dashboard query

3.6.5 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.

3.6.6 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.

3.6.7 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

API Application programming (or program) interface: a set of protocols used by programmers to create applications for a specific operating system or to interface between the different modules of an application. 15

Avatar User graphical representation. It is supplied through an image that will complement the user's profile. 8

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. 4

Business-men People regularly employed in business, especially white-collar workers, executives, or company owners. 4

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). 8

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. 7

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. 6

Database A comprehensive collection of related data organized for convenient access, generally in a computer. 4

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. 8

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

JavaScritp JavaScript is a high-level, dynamic, untyped, and interpreted programming language. It is one of the core technologies of World Wide Web content production. The majority of websites employ it and it is supported by all modern Web browsers without plug-ins. 16

JSON Is an open-standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. JSON is a language-independent data format. It derives from JavaScript, but as of 2016, code to generate and parse JSON-format data is available in many programming languages. 16

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). 6

Logout In computer security, logging out, is the process to end a session at a computer system. 11

MongoDB MongoDB (from humongous) is a free and open-source cross-platform document-oriented database. Classified as a NoSQL database, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. 7

Storage Host Storage Host refers to the practice of storing electronic data with a third party service accessed via the Internet. It is an alternative to traditional local storage (such as disk or tape drives) and portable storage (such as optical media or flash drives). It can also be called "Internet storage" or "cloud storage". 14

UI The *User Interface* in the industrial design field of human–machine interaction, is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process. 4

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". 4