

Tailored IoT & BigData Sandboxes and Testbeds for Smart,  
Autonomous and Personalized Services in the European  
Finance and Insurance Services Ecosystem



D4.16 – Visualization Front-End for  
Aggregated Information - I

<b>Revision Number</b>	3.0
<b>Task Reference</b>	T4.6
<b>Lead Beneficiary</b>	ENG
<b>Responsible</b>	Susanna Bonura – Domenico Messina
<b>Partners</b>	Participating partners in Task according to DOA
<b>Deliverable Type</b>	Report (R)
<b>Dissemination Level</b>	Public (PU)
<b>Due Date</b>	2021-03-31
<b>Delivered Date</b>	2021-04-12
<b>Internal Reviewers</b>	CTAG, JSI
<b>Quality Assurance</b>	CCA
<b>Acceptance</b>	WP Leader Accepted and Coordinator Accepted
<b>EC Project Officer</b>	Pierre-Paul Sondag
<b>Programme</b>	HORIZON 2020 - ICT-11-2018
	This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 856632

## Contributing Partners

Partner Acronym	Role <sup>1</sup>	Author(s) <sup>2</sup>
ENG	Lead Beneficiary	
CTAG	Internal Reviewer	
JSI	Internal Reviewer	Tanja Zdolsek Draksler
CCA	Quality Assurance	

## Revision History

Version	Date	Partner(s)	Description
0.1	2020-12-31	ENG	ToC Version
0.2	2021-01-10	ENG	Added section 1
0.3	2021-01-15	ENG	Added section 2
0.4	2021-01-21	ENG	Added section 3
0.5	2021-02-01	ENG	Added section 4
0.6	2021-19-02	ENG	Added section 5
0.7	2021-03-05	ENG	Updated section 4
0.8	2021-03-19	ENG	Updated section 5
0.9	2021-03-30	ENG	Version ready for Internal Review and Quality Assurance
1.0	2021-04-02	CTAG, JSI, CCA	Version with internal reviews and quality check
2.0	2021-04-03	ENG	Feedback coming from internal review and quality check implemented
3.0	2021-04-12	GFT	Ready to be submitted

<sup>1</sup> Lead Beneficiary, Contributor, Internal Reviewer, Quality Assurance

<sup>2</sup> Can be left void

## Executive Summary

This software deliverable reports the work conducted in T4.6 of the INFINITECH project. Towards this end, the scope of the current report is to present the visualization front-end tool's high-fidelity mock-ups that were designed for all the main functionalities of the application.

At first, the main technical requirements and the design architecture of the first version of the tool to be tested and validated within the Pilot#10 are presented. Then the adopted design process for the user interaction that is tailored to the needs of the task is elaborated. Following this design process, a comprehensive description of the high-fidelity mock-ups is provided and encapsulates the design choices in terms of color schemes, layouts, typography, iconography, spacing and navigation visuals, as well as the overall atmosphere of the tool. Moreover, for each mock-up presented, the expected user interaction with the corresponding mock-up is defined.

# Table of Contents

1	Introduction .....	7
1.1	Objective of the Deliverable .....	8
1.2	Insights from other Tasks and Deliverables .....	8
1.3	Structure .....	9
2	Technical requirements.....	10
3	Real time visualization architecture.....	12
4	User interface design process.....	14
4.1	Mock-ups .....	15
4.1.1	Landing page.....	15
4.1.2	Connections Section.....	16
4.1.3	Charts Section.....	21
4.1.4	Dashboards.....	26
4.1.5	Security Section.....	35
4.1.6	Account Section.....	38
5	Conclusions .....	41

## List of Figures

Figure 1 - INFINITECH work breakdown structure .....	9
Figure 2 - General Agile Development Process ( <a href="http://empireone.com.au/agile-iterative-leandevlopment-what-does-it-all-mean">http://empireone.com.au/agile-iterative-leandevlopment-what-does-it-all-mean</a> ).....	12
Figure 3 – Real time dashboard architecture.....	13
Figure 4 – Design process.....	15
Figure 5 – Login page .....	16
Figure 6 –Burger button .....	16
Figure 7 – New connection form .....	17
Figure 8 - Connection settings .....	18
Figure 9 – Connection list.....	19
Figure 10 – Connection details.....	20
Figure 11 – ‘Delete connection’ option.....	21
Figure 12 – New chart form.....	22
Figure 13 – New chart details .....	23
Figure 14 – Visualization type selection.....	24
Figure 15 – Chart list .....	25
Figure 16 – Chart customization.....	26
Figure 17 – Dashboards list .....	27
Figure 18 – ‘New dashboard’ form .....	28
Figure 19 – Dashboard details .....	29
Figure 20 - ‘Delete dashboard’ option.....	30
Figure 21 – Dashboard customization.....	31
Figure 22 - Dashboard frames and components .....	32
Figure 23 – Tab content.....	33
Figure 24 – Dashboard with Raw Data” and “Graphic Insights” as an example .....	34
Figure 25 – Dashboard result.....	35
Figure 26 – User list .....	36
Figure 27 – Roles list .....	37
Figure 28 – Action log list.....	38
Figure 29 – User menu .....	39
Figure 30 – User details .....	40

## List of Tables

Table 1 – Veesualive technical requirements .....	10
---	----

## Abbreviations/Acronyms

Abbreviation	Definition
API	Application Programming Interface
JDBC	Java Database Connectivity
UI	User Interface
UX	User Experience

# 1 Introduction

This document describes the work done in T4.6 of the INFINITECH project and reports the first result of the web-based framework, named Veesimalive, for the visualization of aggregated results developed in the scope of the project, and more generally of all information of relevance.

In general, visualization tools are widely used to outline and describe datasets as they effectively expose and communicate the structure, patterns, and trends of data and interconnections between them. Financial and insurance organizations' stakeholders require dashboards to utilize visual analytics that incorporate interactive graphics (e.g., gauges, bar charts, and graphs), mapping components and enhanced landscapes to display information on performance, structure, patterns, and trends.

Key data on resources relating to financial and insurance services and infrastructure need to be displayed on a single screen, updated as new data are fed into the database. Furthermore, it must be possible to interact with data (e.g., data selection, filtering and querying data, zooming in/out, data visualization in a number of ways).

Such actions are primarily useful for making sense of very large data sets, revealing context, clusters, gaps, and outliers that may persist invisibly. In addition, data extraction and statistical modelling, such as forecasting, simulation and optimization, can be performed and output via visual interfaces.

By exploiting the data, for example banks can now use the information on a customer's transactions to continuously monitor their behavior in real time, providing the exact type of resource needed at all times. This real-time timeliness increases the overall performance and profitability of the market, pushing it to continue towards a cycle of incessant growth. At this aim, analytics dashboards have proven to have a broader reach, and the main one is uncovering the details behind models and results.

In most analytic dashboards, indicators are involved in monitoring and visualizing the performance of the financial domain. An indicator is a measure or set of measures to evaluate the performance of certain business objectives. Indicators are mapped to uncover patterns by visualizing, mapping and validating analytical results.

The main goal of the work conducted in T4.6 is to deliver a visualization front-end software component to be integrated wherever financial data need to be visualized and analyzed with special emphasis on data streaming. In accordance with the INFINITECH specifications (described in WP2 and WP3 deliverables) the implementation of such a component will be based on microservices approach.

A development approach based on microservices makes components created be composed of services independent of small volumes that interact thanks to specific application programming interfaces (hereinafter APIs), that is function libraries that allow to make calls to parts of a program to shorten the developer's work.

The microservices architectures give the possibility to reach scalability such as to develop the applications in a much faster way than the methodologies traditional, thus allowing to accelerate the deployment of new components. They differ from the monolithic architectures used before the advent of cloud computing, in which all processes are connected to each other, taking the form of a single service.

On the other hand, with an architecture built on microservices, the elements are developed so to be independent, making them single processing flows that each carry out a separate application process. Communication between these components is done through APIs that act as useful interfaces to various services to communicate with each other.

The services are carried out by small separate and autonomous work teams for business functions and each service performs only one function. Being structurally isolated, each service can be freely updated or resized to fill specific functional needs of the program.

## 1.1 Objective of the Deliverable

Within task T4.6, Veesualive, the visualization front-end framework aims to be used to visualize and monitor trends and mine financial big data. It will be able to display real-time data and the most important information necessary to achieve one or more objectives in various ways within 24 hours a day, seven days a week. It aims to be capable of producing graphics from the combination of data extracted from various resources (database, files, API, etc.) suitable for different Fintech/Insurance Tech applications.

This deliverable describes the development process and reports significant results that led to the first version of the visualization front-end for aggregated information: the list of the main high-fidelity mock-ups that were designed for all the main functionalities of the application.

## 1.2 Insights from other Tasks and Deliverables

The WP4 – Interoperable Data Exchange and Semantic Interoperability focuses on establishing the foundation for a common, shared meaning across the several data sources and message and event feeds within the INFINITECH platform while facilitating the technical implementation of the INFINITECH principles. In this landscape, WP4 sets the following objectives:

1. Defined shared semantics (ontologies) for semantic interoperability of BigData and IoT streams in the finance/insurance sectors.
2. Provide the means for scalable the massive analytics over linked semantic streams.
3. Provide a permissioned blockchain solution for exchange data across different organizations in the finance and insurance supply chains.
4. Enhance the permissioned blockchain of the project with tokenization functionalities, as means of enabling digital assets trading.
5. Implement techniques for secure querying of encrypted personal data over a blockchain.

In particular, T4.6 Situation Awareness Front-End over Aggregated Information, will provide a web-based framework for the visualization of the aggregated results of analytic algorithms developed in the scope of the project, and more generally of all information of relevance. The framework will be based on open-source solutions that will be extended and customized in order to support specific data models and persistence technologies. The visualization functionality will allow users to assemble personalized dashboards for situation awareness, wiring together related information from different sources. Special emphasis will be paid in visualizing information from distributed ledgers.

The deliverable builds on top of the work reported in WP2 and WP3 as shown in Figure 1.

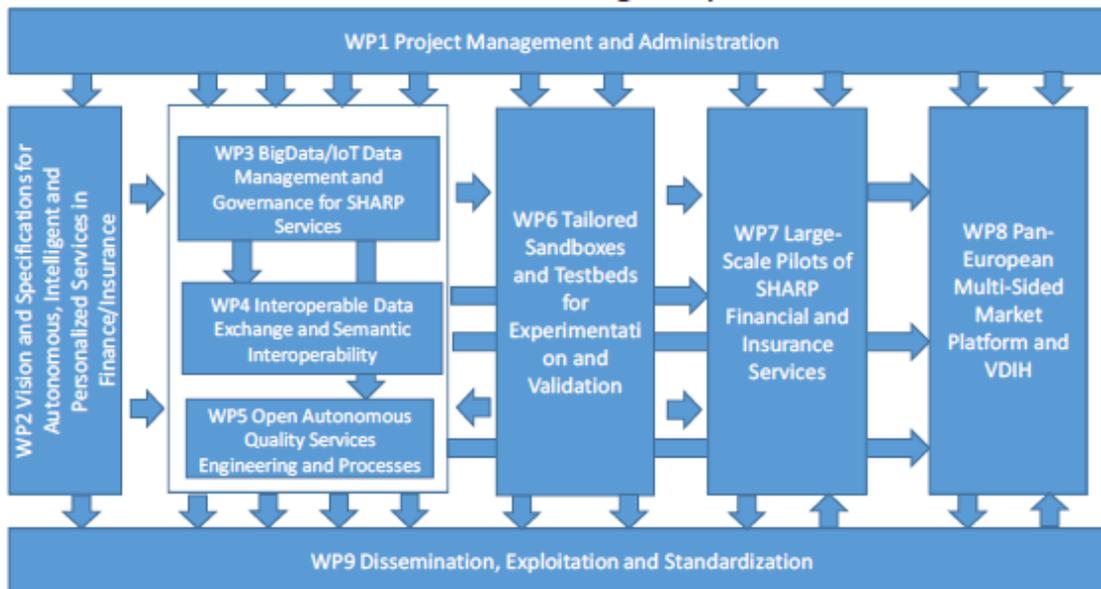


Figure 1 - INFINITECH work breakdown structure

Within WP4, the deliverables to be considered as main inputs for T4.6 mainly come from tasks T4.1 and T4.2.

### 1.3 Structure

Deliverable D4.16 is organized in five main sections as indicated in the table of contents.

- The first section introduces the deliverable. It documents the objectives of the deliverable and the relation of the current deliverable with the other deliverables by describing how the outcomes of other deliverables and work-packages serves as input to the current deliverable. Finally, a brief description is provided on how the document is structured.
- The second section presents the main technical requirements elicited by taking into account the deliverable coming from WP2 and WP3 and thanks to a first round of feedbacks coming from T4.2 and Pilot#10 design and development.
- The third section provides a high-level architecture of Veesimalive: in summary it describes the capability to ‘consume’ and visualize real-time data produced and stored as incoming data flows in a Kafka topic.
- In the fourth section, the high-fidelity mock-ups are presented. At first, the design process that was adopted and the tool that was utilized in the process are presented. Following the process and tools presentation, the designed mock-ups are presented accompanied by a brief description of the user’s interactions with the platform.
- The fifth section concludes the deliverable. It outlines the main findings of the deliverable which will guide the future research and technological efforts.

## 2 Technical requirements

Based on the requirements elicited within the Pilot#10 design and implementation, as well as on the specifications provided in WP2 and WP3, this section explains the requirements currently addressed by the visualization front-end component and the implementation details currently envisioned for this component.

Such requirements are listed in Table 1.

Table 1 – Veesimalive technical requirements

Technical requirements		
FE_01	Stream data ingestion	Ability to ingest and visualize data streaming.
FE_02	Multiple results backend	Allowance of use of a backend like S3, Redis, Memcached.
FE_03	Multiple message queue	Usage of message queues like Kafka, Redis, RabbitMQ, SQS.
FE_04	Multiple metadata database engine	Adoption of metadata database engine (MySQL, Postgres, MariaDB, ...).
FE_05	High availability	Ability to scale out in large, distributed environments, and works inside containers.
FE_06	Portability	Deployment of each single component of the architecture as a micro service that is extensible, independent and altogether contributes to a modular and scalable architecture, thus it must be possible to deploy it to any other system where Docker is running.
FE_07	Dashboard customization	Ability to assemble personalized dashboards for situation awareness
FE_08	Support of several chart types	Presence of a rich set of charts (Line chart, time series, table, bar chart, are chart, heatmap, pie chart, treemap, box plot, map chart, bubble chart).
FE_09	Organization authentication (OpenID, LDAP, OAuth...)	Easy access for users to allow them to use existing credentials obtained from official existing services. The intention is to request to user as little information as possible. This will start from the login option that will integrate with the eID standard to try to recover the user information through the already delivered digital identities.
FE_10	User Authentication and Permissions by groups and roles	Support of user groups management with different authentication schema.
FE_11	Lightweight semantic support	Ability to control how data sources are exposed to the user by defining dimensions and metrics.
FE_12	Isolation	Containerization: it must be made available as a docker container. A Docker container that contains one of your applications also includes the relevant versions of any supporting software that your application requires. If other Docker containers contain applications that require different versions of the same supporting software, that is not a problem because the different Docker containers are totally independent of one other. This also means that as you move through the various stages of your development

		lifecycle, you can have total confidence that an image you create during development will perform exactly the same as it moves through testing and potentially to your users.
FE_13	Interactivity	Presence of a visual interface which the user can interact with. User should be able to intuitively comprehend the data, perceive the underlying patterns and query the data visually, without the need for programming knowledge.
FE_14	Data Reduction	Ability to reduce data. In case of data at-rest, it should be considered that computer screens have a limited number of pixels onto which data points can be rendered. There are several ways to reduce the data, including filtering, clustering, and sampling techniques.
FE_15	Custom visualizations	Exposure of a single diagram or chart or dashboard as an API call or embedding, since the final experience that the project wants to offer to the end user, on some frontend, is a unified interface detached from the dashboarding visualization tool.
FE_16	Authentication	Custom authentication option to be able to integrate authentication within the overall system; this should include a single sign on feature to allow users to shift from an application to dashboarding design with a uniform user experience.

### 3 Real time visualization architecture

Veesualive is the component enabling the visualization capabilities of the INFINITECH visualization tool for the output of the direct data acquisition from different data sources as well as the output of the querying and filtering results that will come from the Query Builder (tasks T4.2). More specifically, the Visualization Front-End is undertaking the necessary actions to address the advanced visualization requirements by offering a variety of visualization formats, which span from simple static charts to interactive charts offering several layers of information and customization.

Veesualive will consist of a set of functionalities which support the execution of the visualization process. This set of functionalities includes the dataset selection, the dataset preview generation, the visualization type selection, the visualization configuration, the visualization generation and the interactive dashboard.

In the following list, the functionalities envisioned for Veesualive are described:

- Dataset selection: The list of available datasets is presented to the user for the selection.
- Dataset preview generation: upon selecting the dataset, a preview of the dataset is displayed.
- Visualization type selection: the list of available visualizations for the selected dataset is presented to the user for selection.
- Visualization configuration: based on the visualization type selected for the desired dataset a set of parameters are displayed to the user to get the visualization.
- Visualization generation: once the visualization type along with the parameters are set for the desired dataset, the visualization generation is triggered. The results can be used in the current session for creating an interactive dashboard.
- Dashboard: the result of the visualization generation is presented to the user into an interactive dashboard. This dashboard can contain also multiple generated visualization results.

To address the development activities that involve the abovementioned functionalities implementation, methods and instruments of the standard agile methodology and their adaptations to the needs of the development of Veesualive have been taken into account. Figure 2 describes the core parts of all agile development processes in which the recurring execution of a defined sequence of process phases is presented.

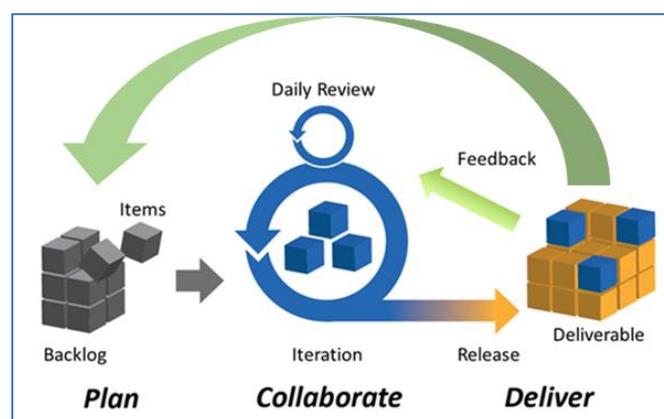


Figure 2 - General Agile Development Process (<http://empireone.com.au/agile-iterative-leandevlopment-what-does-it-all-mean>)

User requests for a new feature or the enhancement of an existing feature of the product are managed in a so called backlog. For each iteration, a set of these requests will be determined and brought into the development. The resulting release will be delivered to the stakeholders of the project, who will evaluate it and provide feedback. Within the task T4.6 the development process embraces this process with the necessary adjustments to fit the needs of the task.

In the current ongoing iteration for the next version of the visualization front-end a variety of visualization types are considered to be supported and will be further extended in the upcoming versions. The list of supported visualization types includes scatter plot, bar chart, pie chart, bubble chart, time series, heat map, map, box plot and histogram. Veesimalive is following the microservices architecture. The designed microservices will enable the advanced visualization capabilities and will be orchestrated towards the execution of the visualization process, as described in the abovementioned item list. Each microservice will be assigned with a specific functionality within the visualization process.

The next step will involve the implementation and integration of a stream data source connector to visualize real time data. More in detail the architecture (Figure 3) presents a sink connector to continuously ingest data streams from Kafka<sup>3</sup> and serve fast queries; it takes every event in the topics being watched so that users can then build real-time dashboards or data APIs on top of the data.

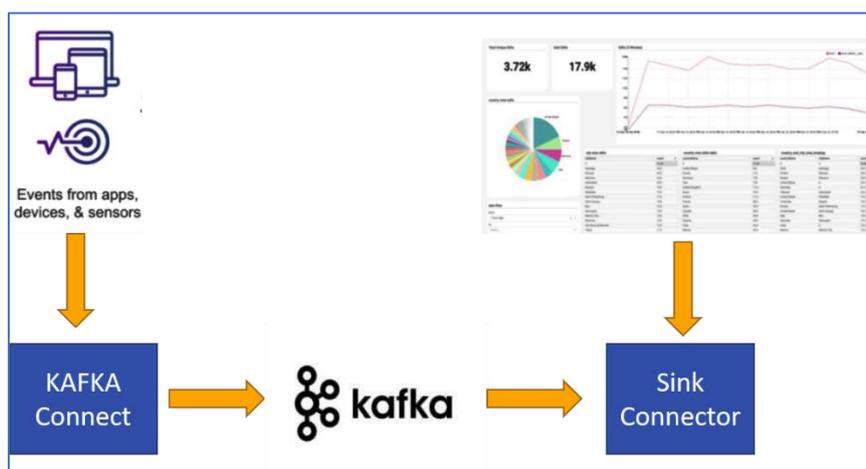


Figure 3 – Real time dashboard architecture

Apache Kafka is an event streaming platform that combines messages, storage, and data processing. The Apache Kafka project includes two additional components: Kafka Connect for integration and Kafka Streams for stream processing. Kafka’s ecosystem also includes other valuable components, which are used in most mission-critical projects. Among these are Confluent Schema Registry, which ensures the right message structure, and ksqlDB for continuous stream processing on data streams, such as filtering, transformations, and aggregations using simple SQL commands. Not only can Kafka be used for both real-time and batch applications, but it can also integrate with non-event-streaming communication paradigms like files, REST, and Java database connectivity (hereinafter JDBC). In addition, it is often used for smaller datasets (e.g., bank transactions) to ensure reliable messaging and processing with high availability, exactly once semantics, and zero data loss.

<sup>3</sup> <https://kafka.apache.org/>

## 4 User interface design process

To have a clear idea of how a software application should look and behave before starting time-consuming tasks like coding, development, debugging and installation is crucial. Creating a sketch first, then a wireframe, then a mock-up and finally the prototype, can eliminate any communication gaps early, and start the software development process with confidence.

To this aim within the task T4.6, the design process that was followed included the following steps (Figure 4):

- **Sketch:** this is typically the first step in the design process. In this step, several brainstorming sessions took place to exchange initial ideas on how the front-end should be and how the various operations could be designed towards the easy and straightforward execution of the various tasks performed. This step produced the first low-fidelity representation of the visualization front-end component in the form of sketch drawing on a whiteboard and pieces of paper. Sketching provided a quick and easy way to focus and organize the work that needs to be done in the wireframe step.
- **Wireframe:** this step includes the visual representation of the content layout in a design. It produced the skeleton or structure of the user interface. It made it easier to organize and simplify the elements and the content within a space. Each produced wireframe is used to describe the functionalities and the relation between the different views that will be developed. In this stage, the decisions on what to include, in terms of content and features were taken. The wireframes provided the first overview of the work that needs to be implemented.
- **Mock-up:** in this step, the high-fidelity mock-ups were created based on the final version of the wireframes. In general mock-ups reflect the design choices for the color schemes, layouts, typography, iconography, spacing, the navigation visuals and the overall atmosphere of the visualization tool. The mock-ups easily provide the look and feel of the final product to the stakeholders thanks to their high-fidelity nature. Moreover, the mock-ups provide a realistic perspective of the visual decisions that were made and unveil any problems that were initially unnoticed.
- **Prototype:** The final step of the design process is the high-fidelity representation in the form of prototype. While the wireframes handle the structure and the mock-ups handle the visuals, the prototypes handle the usability aspect. The prototype is early internal release of parts of the designed tool that are used to test the user interactions within the user interface. Prototypes provide valuable input for the actual implementation of the user interface.

The mock-ups were designed using Figma<sup>4</sup>. Figma is a browser-based user interface (hereinafter UI) and user experience (hereinafter UX) design application with powerful design and prototyping tools. It is considered to be one of the dominant interface design tools, with multiple robust features that support the design process on every step of the design process. Moreover, Figma provides real-time collaboration for teams with version history.

---

<sup>4</sup> [www.figma.com](http://www.figma.com)

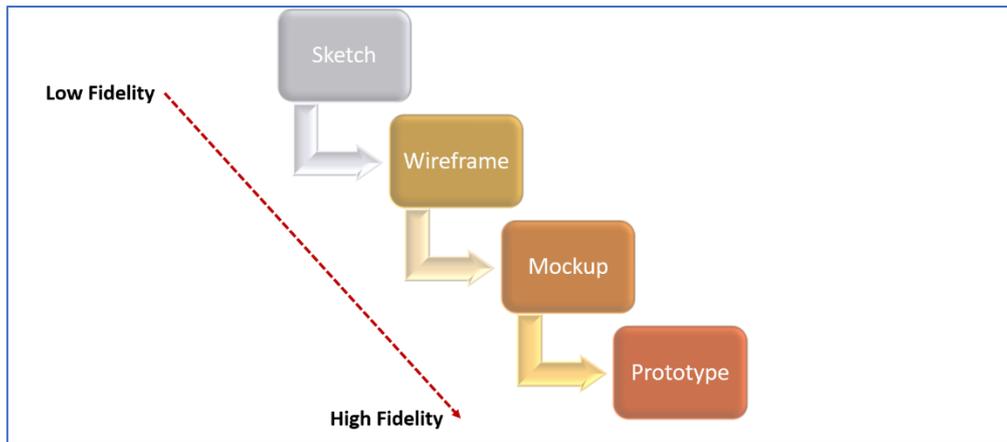


Figure 4 – Design process

## 4.1 Mock-ups

Mock-ups were designed following the design process that was elaborated in the previous section. For each feature, a set of figures is provided along with a brief description of the corresponding interactions of the user with the front-end. It should be noted that the following sections present the main set of functionalities of each feature of the front-end and the respective interactions of the user. However, the produced mock-ups cover the full set of functionalities of each feature and all the possible interactions of the user.

The complete list of mock-ups is available online in the following URL:

<https://www.figma.com/file/A6IA9guZeSbFFtsbgwOhQv/Infinittech-D4.16-Mockups>

### 4.1.1 Landing page

The access to the tool is regulated with the user authentication control. In order to access the services of the visualization tool, the user is required to enter his/her credentials as depicted in Figure 5, provided that the organization he/she belongs has been already registered. In the case where the organization has not yet been registered, the organization registration process needs to be followed. Additionally, in the case where the user does not have an account, she must first register to the platform where the visualization tool is integrated and deployed using a register form.

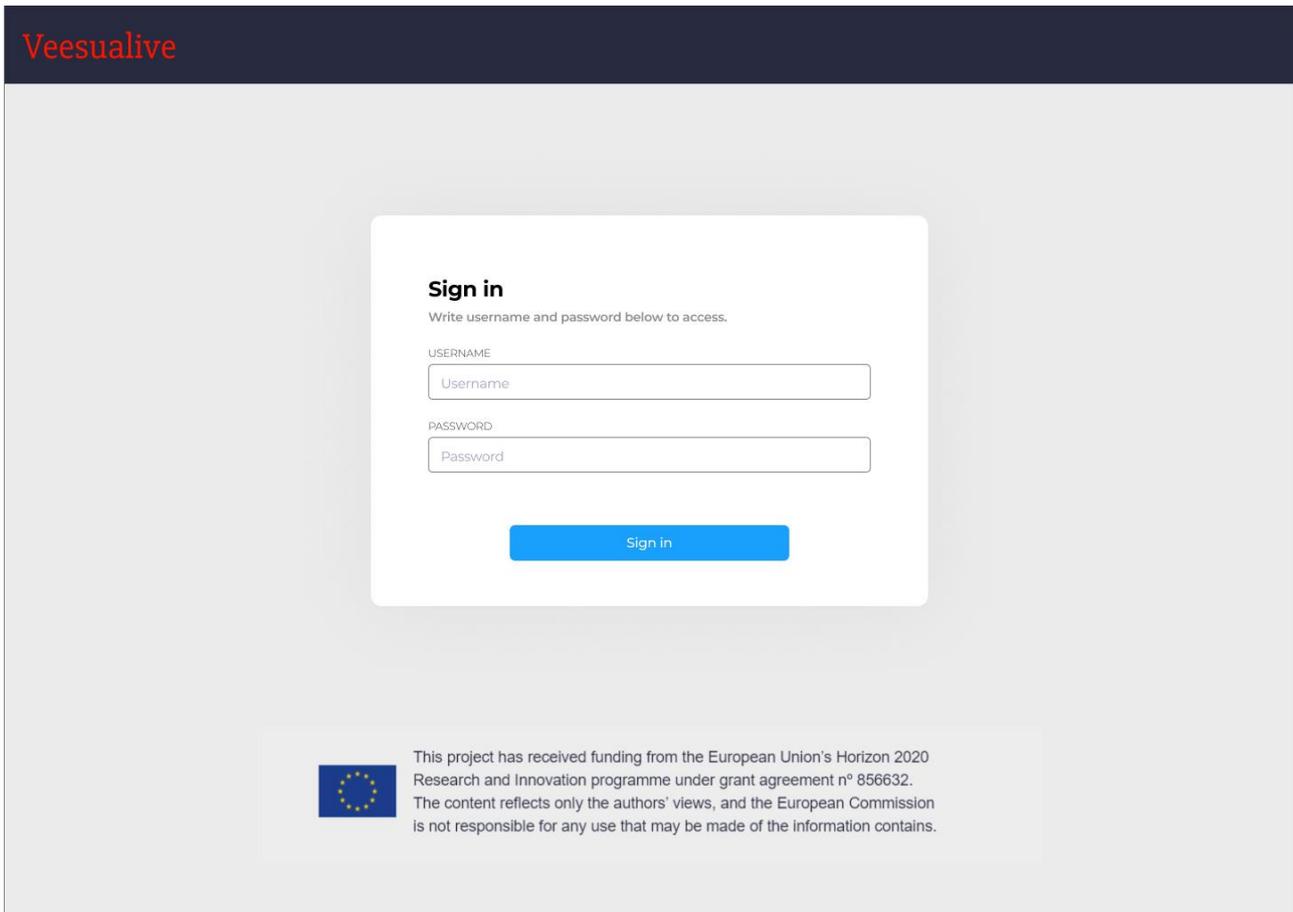


Figure 5 – Login page

Upon successful login, the user is able to access to the visualization tool, divided in four main sections: security, connections, charts and dashboards.



Figure 6 –Burger button

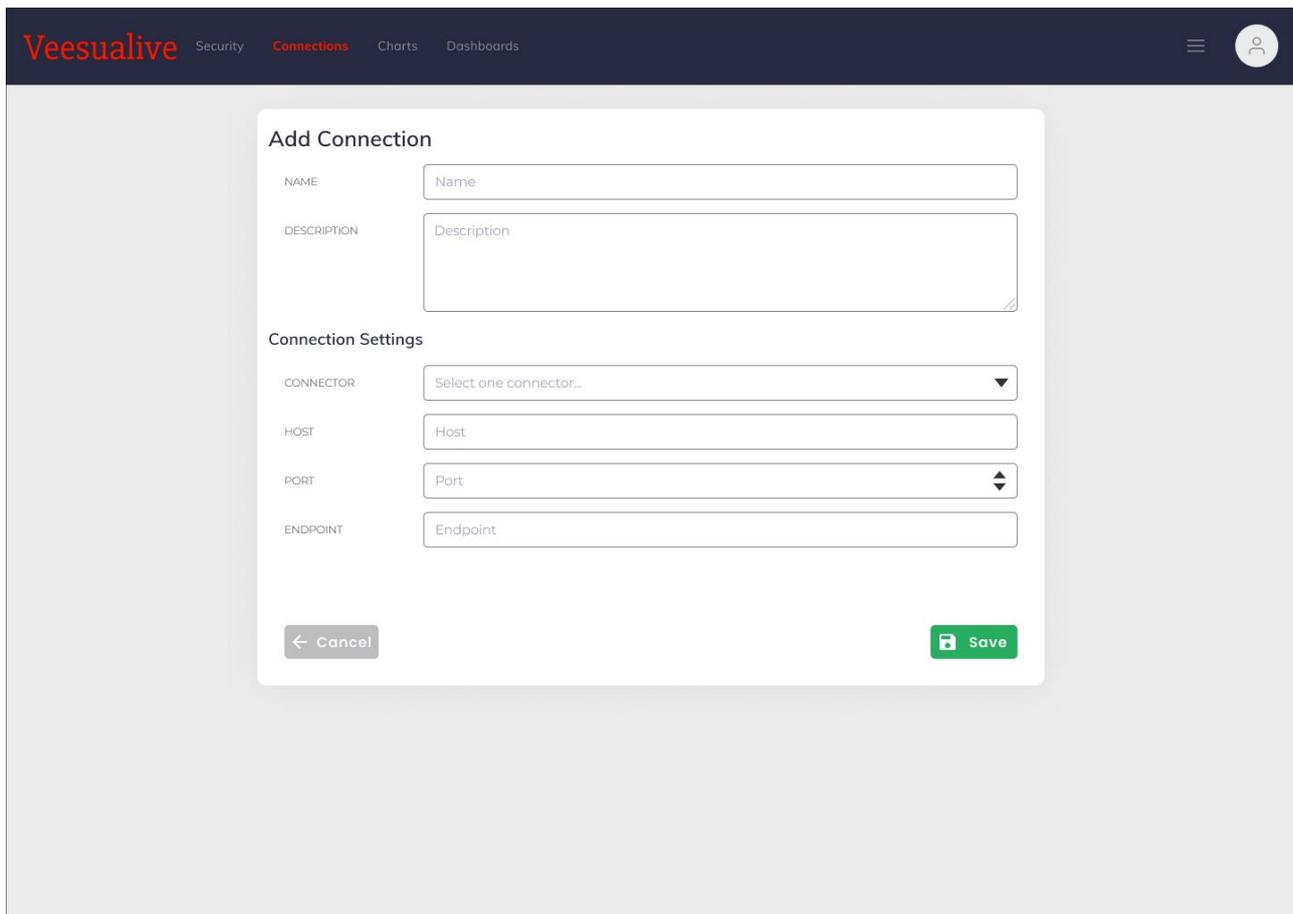
The navbar on top is provided also with a menu burger button to access additional options and a user menu, where he/she can manage account settings or sign out (Figure 6).

## 4.1.2 Connections Section

Connections represent the mean which allows the user to reach his/her own data, that can be either stored on a database or available through alternative mechanisms, such as streaming message queues or topics.

### 4.1.2.1 New Connection

As soon as the user decides to add a connection to the ones available, a form shows up where she must fill in some details about the new connection (Figure 7).



The screenshot shows the 'Add Connection' form in the Veesimalive application. The form is titled 'Add Connection' and is located in the center of the page. It has a white background and a dark border. The form is divided into two main sections: 'NAME' and 'DESCRIPTION', and 'Connection Settings'. The 'NAME' section has a text input field with the placeholder text 'Name'. The 'DESCRIPTION' section has a larger text area with the placeholder text 'Description'. The 'Connection Settings' section includes a dropdown menu for 'CONNECTOR' with the text 'Select one connector...', a text input field for 'HOST' with the placeholder text 'Host', a dropdown menu for 'PORT' with the placeholder text 'Port', and a text input field for 'ENDPOINT' with the placeholder text 'Endpoint'. At the bottom left of the form is a 'Cancel' button with a left-pointing arrow, and at the bottom right is a 'Save' button with a floppy disk icon.

Figure 7 – New connection form

Along with a name and optional description, the user is also asked to provide the actual connection settings, represented by the connector, that could be any streaming or batch source, e.g. Kafka, Redis or Mongo, the connection host and port, together with the endpoint, which is the last part of the connection string that specifies the exact location to look at to find the data (Figure 8).

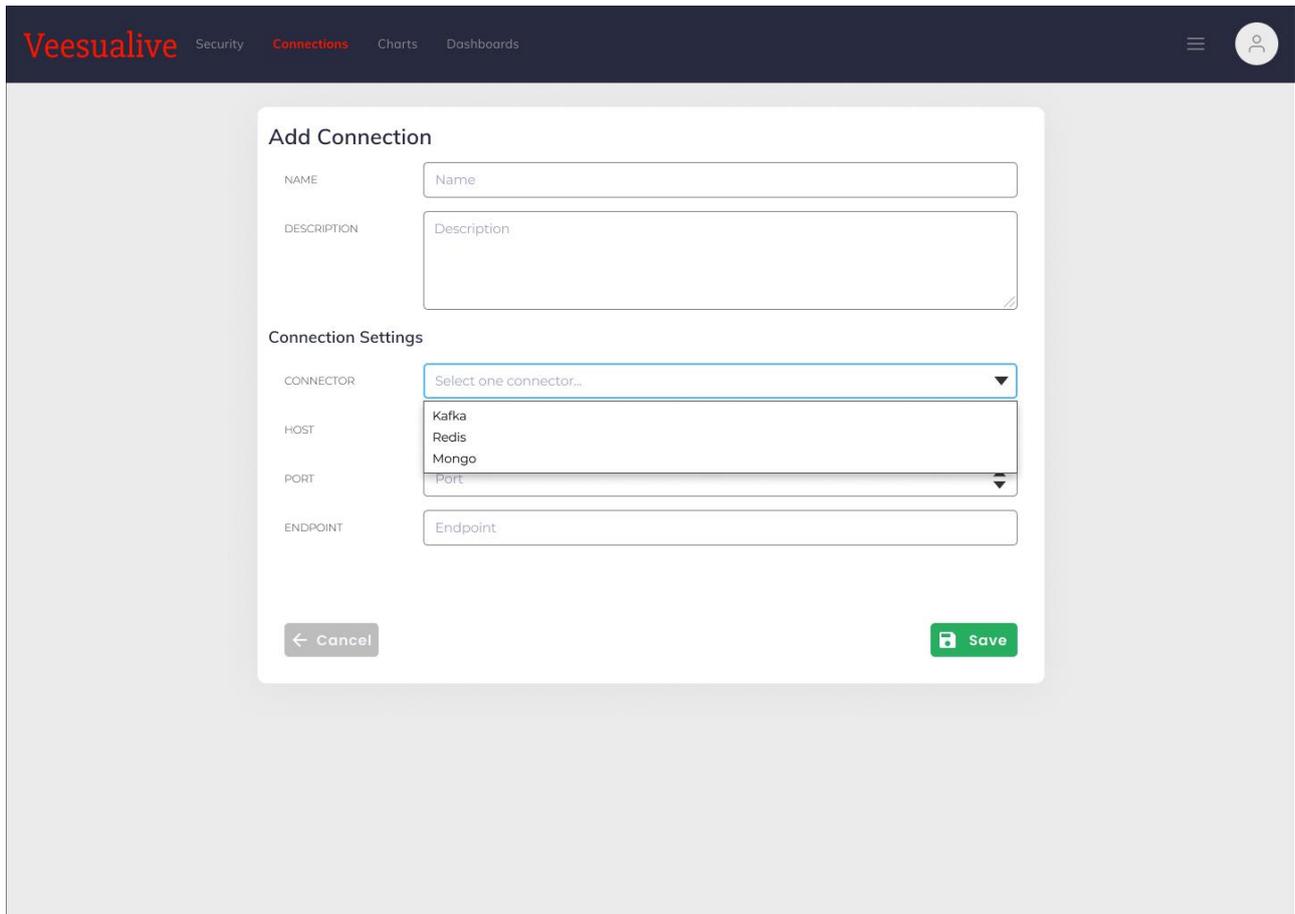


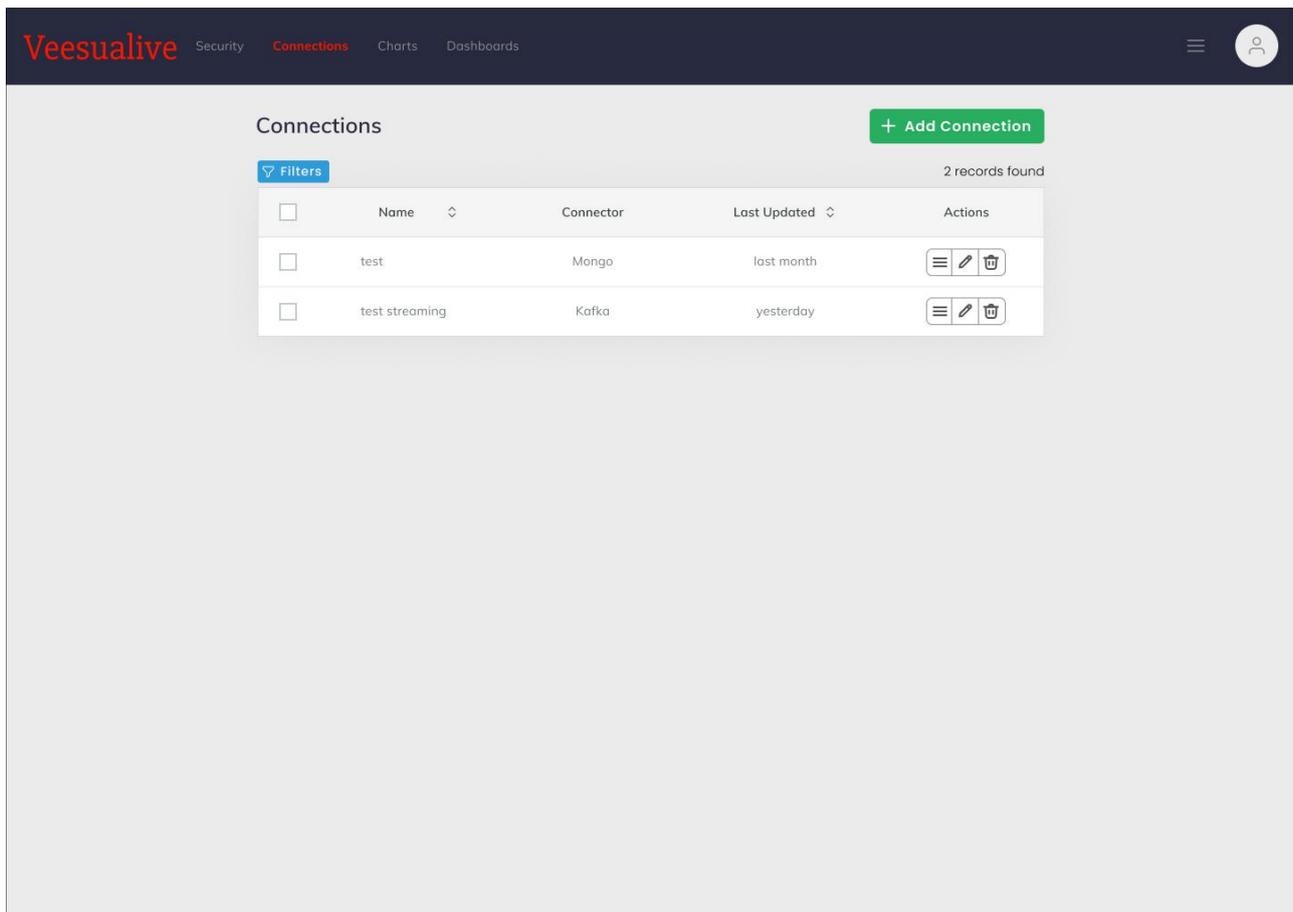
Figure 8 - Connectionsettings

#### 4.1.2.2 Connections List

When the user has finished filling in the required information and submits the form, the new connection is added to connections list page (Figure 9).

On the upper part of the page there are also some control buttons, one on the far right to create a new connection, one above the connections, on the left, to filter the list based on the user preferences.

The list of connections is in the shape of a table featuring connection name, the time when it was last updated and the connector, that is the technology behind it, the type of database or message broker used to establish the connection. Each entry is provided also with a set of action buttons that allow the user to view the connection details, edit the connection with the same form used to create a new one or delete it.



The screenshot shows the 'Connections' page in the Veesimalive application. The page has a dark blue header with the 'Veesimalive' logo and navigation links for 'Security', 'Connections', 'Charts', and 'Dashboards'. A user profile icon is visible in the top right corner. Below the header, the 'Connections' section features a green '+ Add Connection' button and a 'Filters' button. A table displays two records, with a '2 records found' indicator. The table columns are 'Name', 'Connector', 'Last Updated', and 'Actions'. Each row includes a checkbox and a set of action icons (list, edit, delete).

<input type="checkbox"/>	Name	Connector	Last Updated	Actions
<input type="checkbox"/>	test	Mongo	last month	  
<input type="checkbox"/>	test streaming	Kafka	yesterday	  

Figure 9 – Connection list

### 4.1.2.3 Connection Details

The details button on the connection entry gets the user to the connection details page, represented by a card containing a summary of that connection (Figure 10).

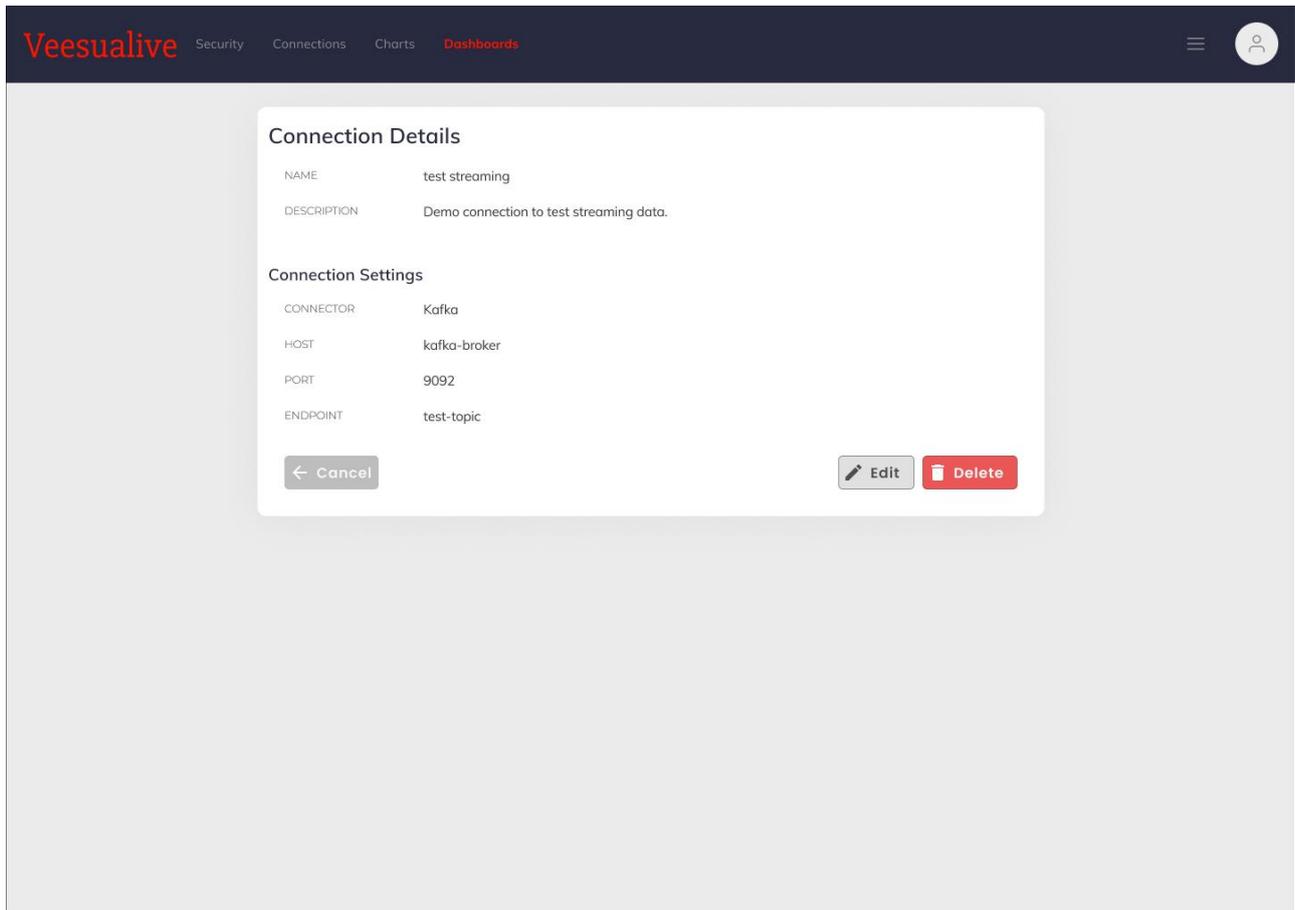


Figure 10 – Connection details

Along with the information, at the bottom of the card there is a set of buttons to come back to the list, to edit the connection or to delete it.

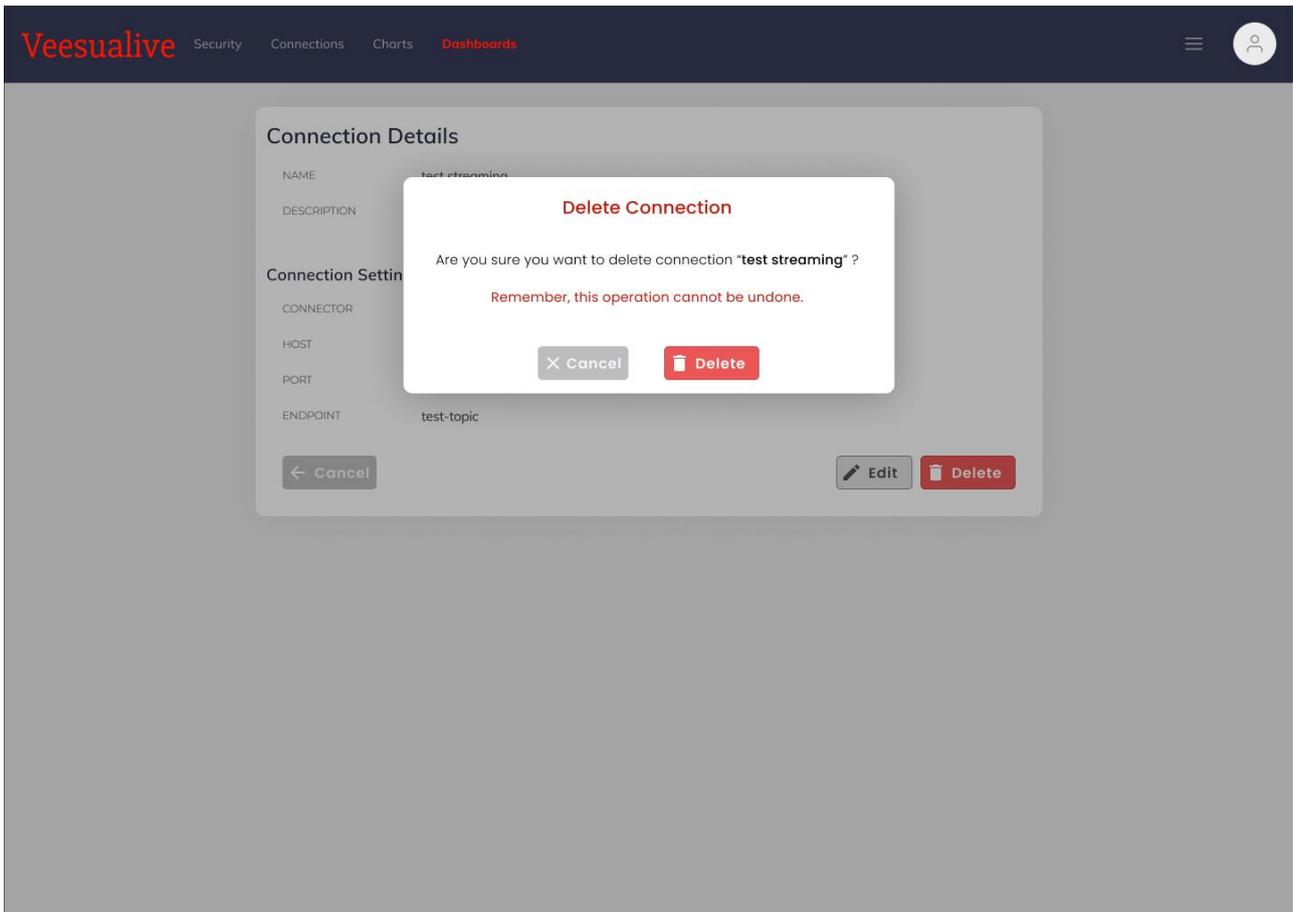


Figure 11 – ‘Delete connection’ option.

The delete button triggers a confirmation dialog that warns the user that the action cannot be reverted (Figure 11).

### 4.1.3 Charts Section

Charts (or graphs) are defined as graphical representations for data visualization, in which data is represented in the form of lines, shapes, dots, bars, slices and other simpler or more complex forms, based on the type of diagram, map or graph.

Veesualive charts can be configured during the chart customization to compose wonderful walls called Dashboards, where the user can have insights on his/her own data at first sight.

#### 4.1.3.1 New Chart

To add a new chart, the user must provide the mandatory information in the “Add Chart” form and fill in the chart metadata (Figure 12).

The screenshot shows the 'Add Chart' form in the Veesimalive application. The form is centered on a light gray background. It has a title 'Add Chart' at the top left. Below the title are four sections: 'NAME' with a text input field containing 'Name'; 'DESCRIPTION' with a larger text area containing 'Description'; 'CONNECTION' with a dropdown menu showing 'Select one connection...'; and 'VISUALIZATION TYPE' with a radio button labeled 'TABLE'. At the bottom left is a 'Cancel' button with a left arrow, and at the bottom right is a green 'Save' button with a save icon. The top navigation bar includes the Veesimalive logo and links for Security, Connections, Charts, and Dashboards.

Figure 12 – New chart form

Together with the chart name and optional description, it is enough to provide the connection and the visualization type.

The screenshot shows the 'Add Chart' form in the Veesimalive application. The form is titled 'Add Chart' and is located in the 'Charts' section of the dashboard. It contains the following fields and options:

- NAME:** A text input field with the placeholder text 'Name'.
- DESCRIPTION:** A larger text input field with the placeholder text 'Description'.
- CONNECTION:** A dropdown menu with the placeholder text 'Select one connection...'. The dropdown is currently open, showing two options: 'Mongo / test' and 'Kafka / test streaming'.
- VISUALIZATION TYPE:** A dropdown menu that is currently closed.

At the bottom of the form, there are two buttons: a grey 'Cancel' button with a left-pointing arrow, and a green 'Save' button with a save icon.

Figure 13 – New chart details

Chart connection can be chosen among the list of the ones that were previously created by the user (Figure 13).

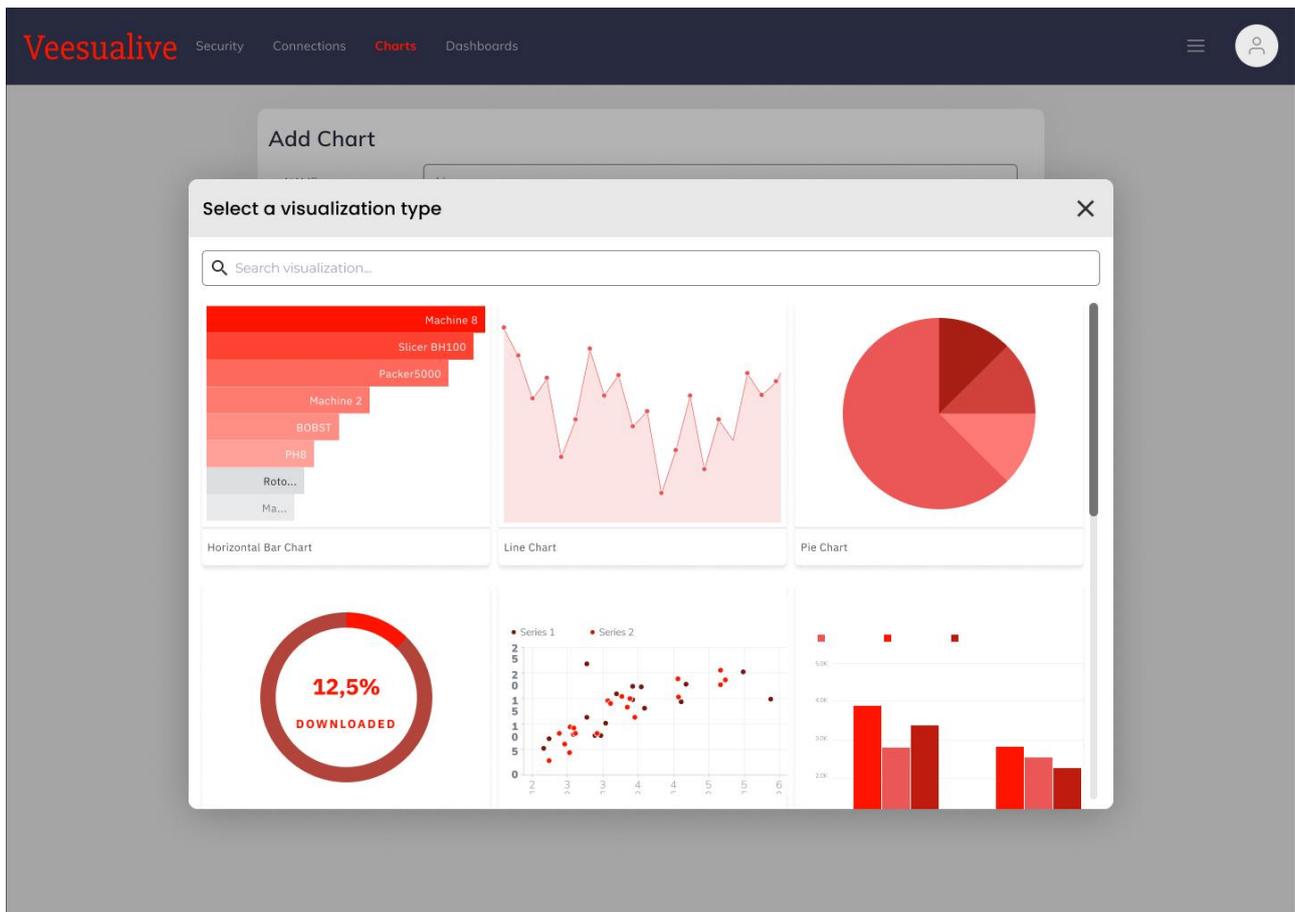


Figure 14 – Visualization type selection

Default visualization type is “Table”, that just prints the raw data, but can be changed with a simple click on the pill; this opens a modal which contains the list of all the chart types available, from the traditional histograms and pies to the more sophisticated Gantt and Scatter Plot, and can be browsed with ease thanks to the search bar above (Figure 14).

#### 4.1.3.2 Charts List

Upon the form submission, the new chart is added to the others and becomes available in the chart list page. Each entry of the list is indicated by its name, the visualization type, the data source, i.e. the connection, the user that created it and the date when the chart was last updated (Figure 15).

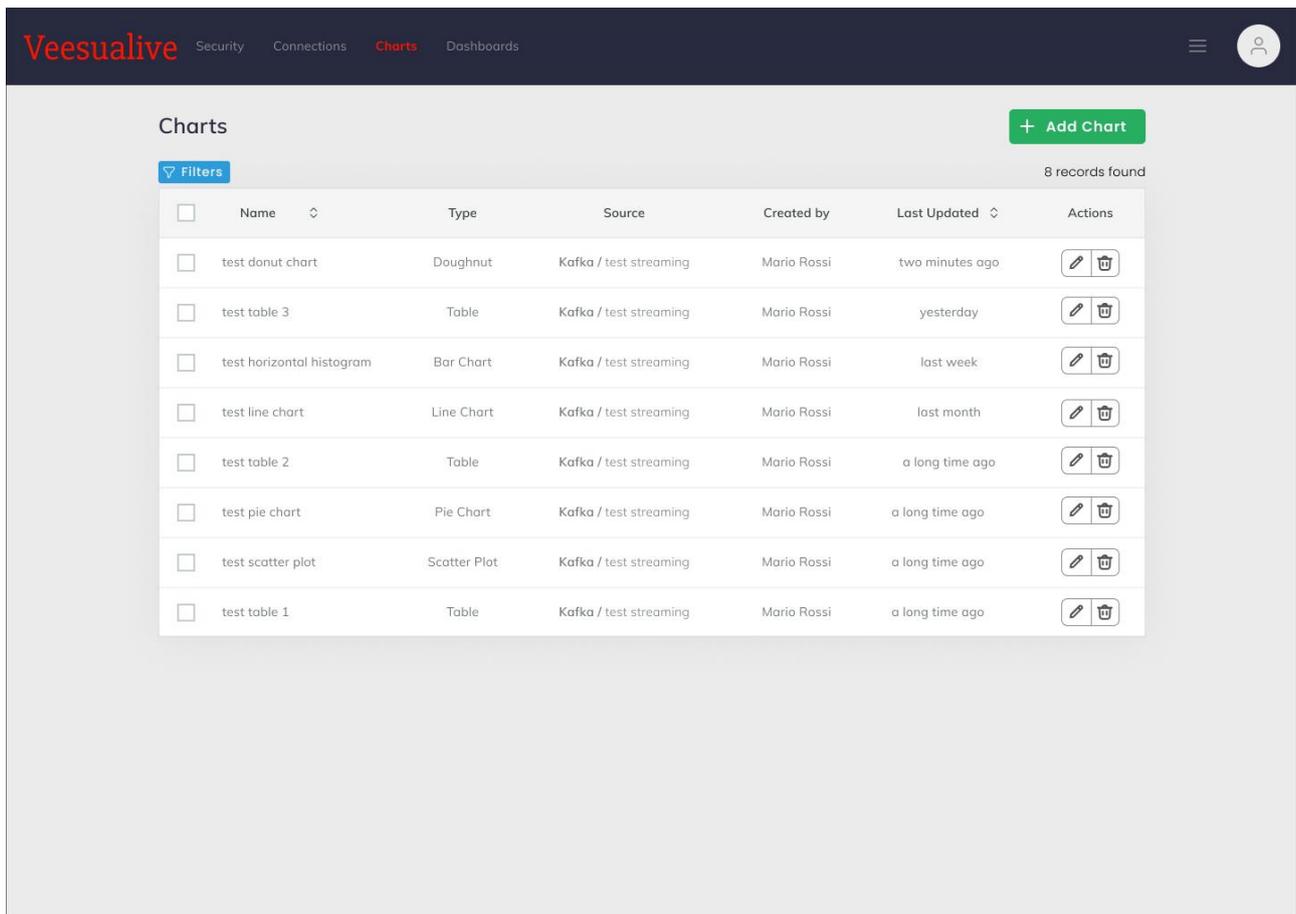


Figure 15 – Chart list

Together with the mentioned information, every chart can be edited or deleted, thanks to the buttons on the right.

Above the list there are also two control buttons, one to get to the form to add a new chart, and one to improve the chart browsing thanks to some filters.

Apart from the metadata, the actual chart can be configured in the chart customization, accessible through a simple click on the chart title.

### 4.1.3.3 Chart Customization

Chart customization lets the user tailor data visualization in every aspect (Figure 16).

The top main toolbar contains the chart title and a set of utility buttons on the far right to export the chart in the form of an iframe to embed in an external web page or via a URL, download it in JSON or CSV formats and other options and settings.

Below, the page is divided in three columns. The column on the far left contains the general information about the source, data structure and the dataset available metrics, such as average, sum and count.

The second column is about the actual settings that affect the final look of the chart, divided in two tabs, one strictly regarding data, the other groups graph-specific properties that can be fine-tuned according to the user needs and taste.

Data customization includes settings about the visualization type, time metrics and the query logic to get data, which can be standard or aggregated, can include all fields or just some of them and can limit the rows or sort them.

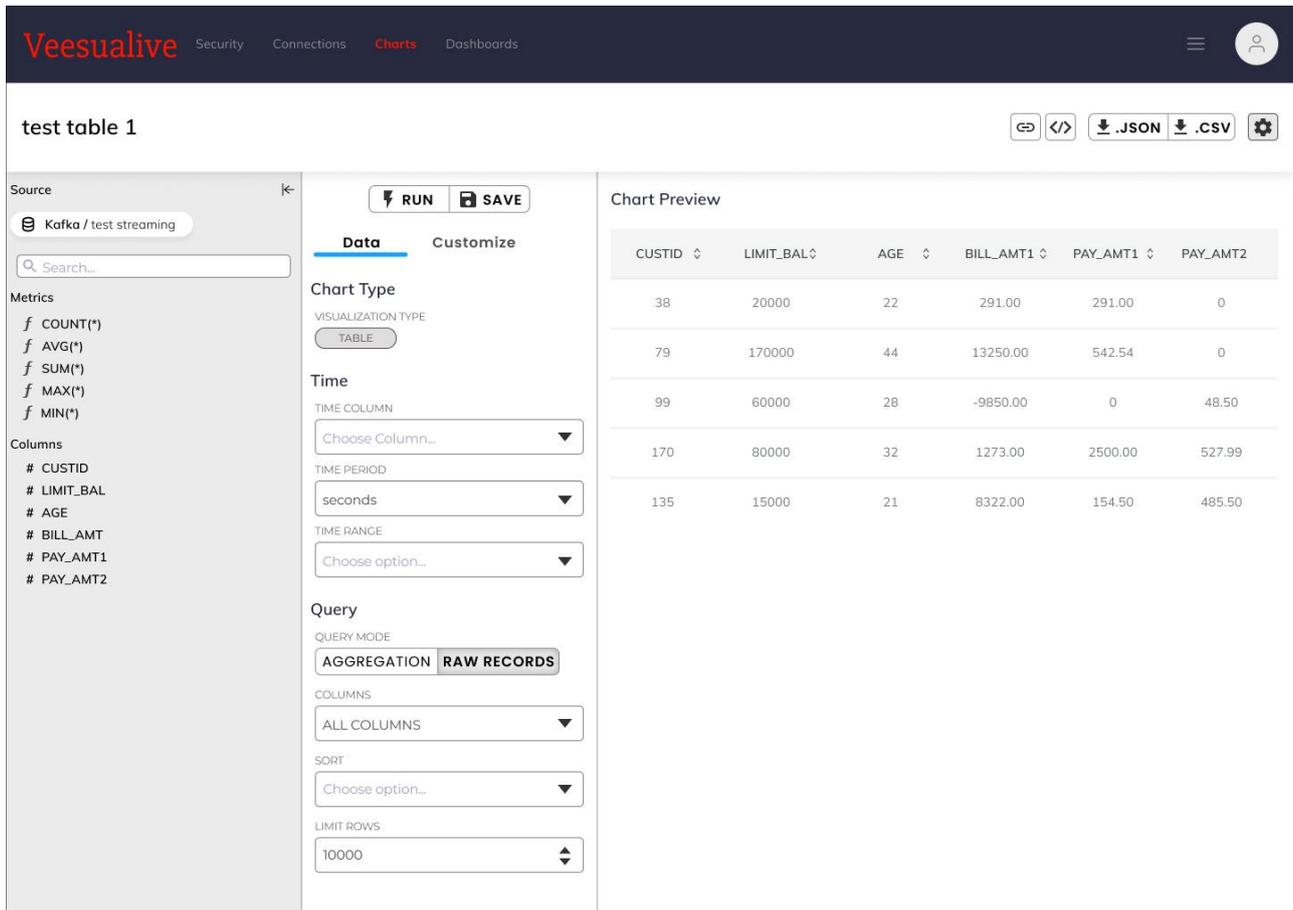


Figure 16 – Chart customization

## 4.1.4 Dashboards

A dashboard may be defined as an aggregation of charts and tables which is useful for keeping the information of interest on a single panel, to always have the main insights about data at a glance.

### 4.1.4.1 Dashboards List

Veesimalive home page is represented by the dashboards list page, as illustrated in Figure 17. Here a list of dashboards is showed and for each of them it is possible to edit, show or delete it.

Above the table listing the available dashboards, two control buttons allow the user to register a new dashboard's metadata or to filter the entries below.

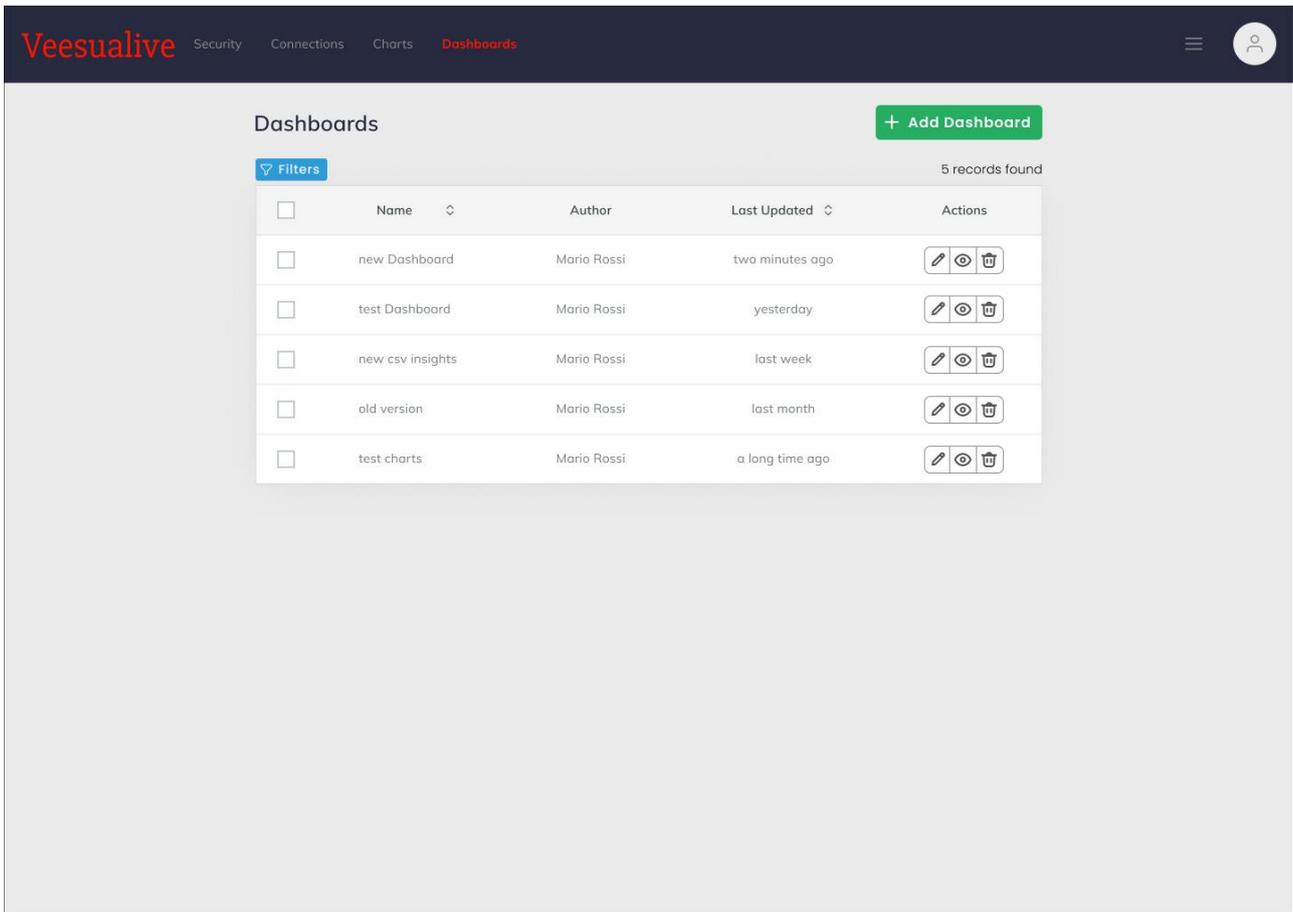


Figure 17 – Dashboards list

#### 4.1.4.2 New Dashboard

As soon as the user clicks on the “Add Dashboard” button, he/she is redirected to a form where basic metadata must be provided, i.e. a title, an optional URL to make the dashboard resource available on the web in a more readable way, as well as a flag to indicate whether the dashboard should be published or not, that means if other users can see it or if it still needs to be updated/improved and remains private for the author (Figure 18).

The screenshot shows the 'Add Dashboard' form in the Veesimalive application. The form is centered on a light gray background. It has a title 'Add Dashboard' and three input fields: 'TITLE' with placeholder 'Dashboard name', 'URL' with placeholder 'Dashboard readable URL', and 'PUBLISH' with an unchecked checkbox. At the bottom left is a 'Cancel' button with a left arrow, and at the bottom right is a green 'Save' button with a floppy disk icon. The top navigation bar shows 'Veesimalive' in red, followed by 'Security', 'Connections', 'Charts', and 'Dashboards' in white. On the right of the navigation bar are a hamburger menu icon and a user profile icon.

Figure 18 – 'New dashboard' form

#### 4.1.4.3 Dashboard Details

Upon successful submit, the new dashboard is added to the list among the available ones and its details can be visualized through the detail button provided for every entry (Figure 19).

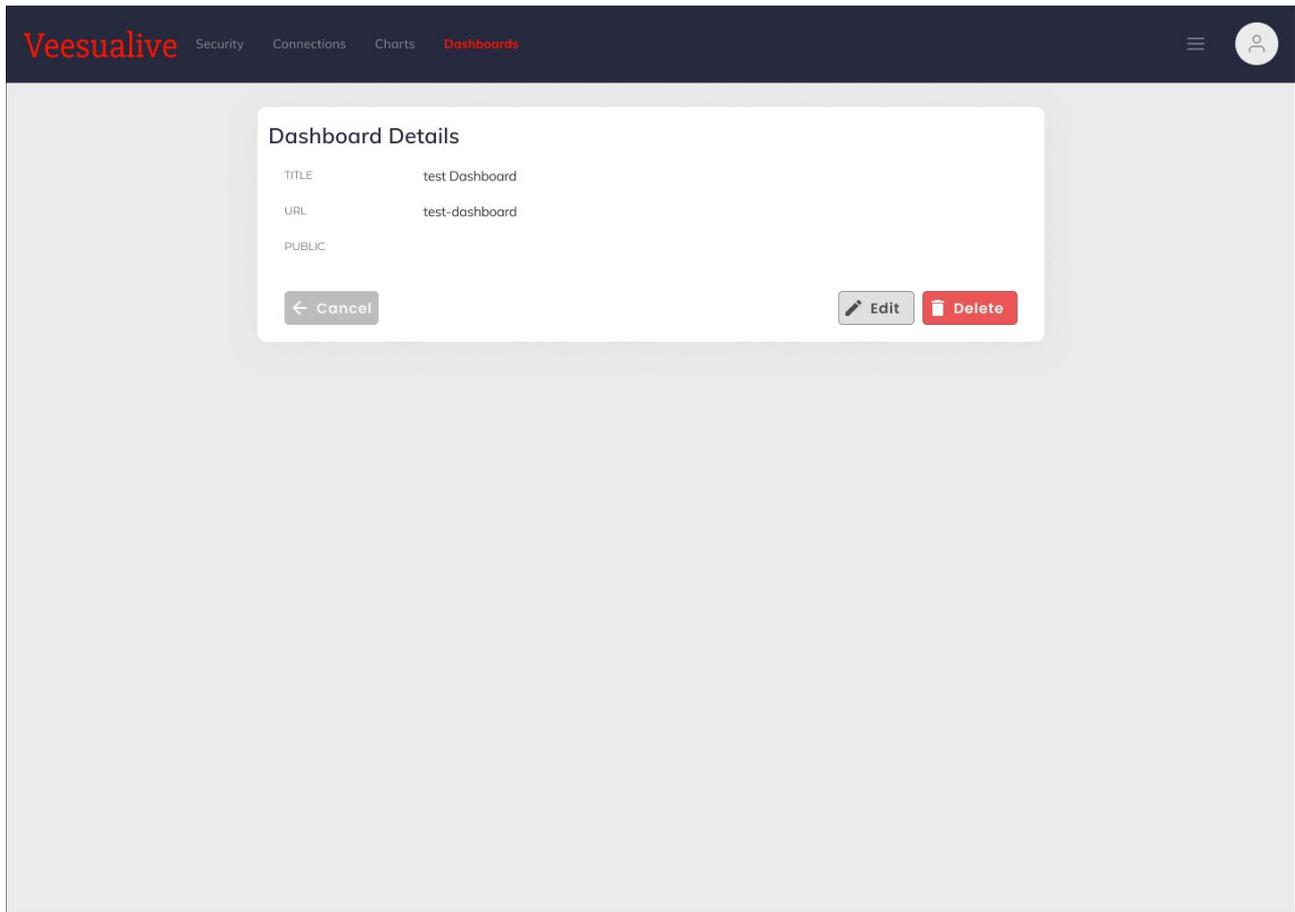


Figure 19 – Dashboard details

From here, the user can also edit or delete it (Figure 20).

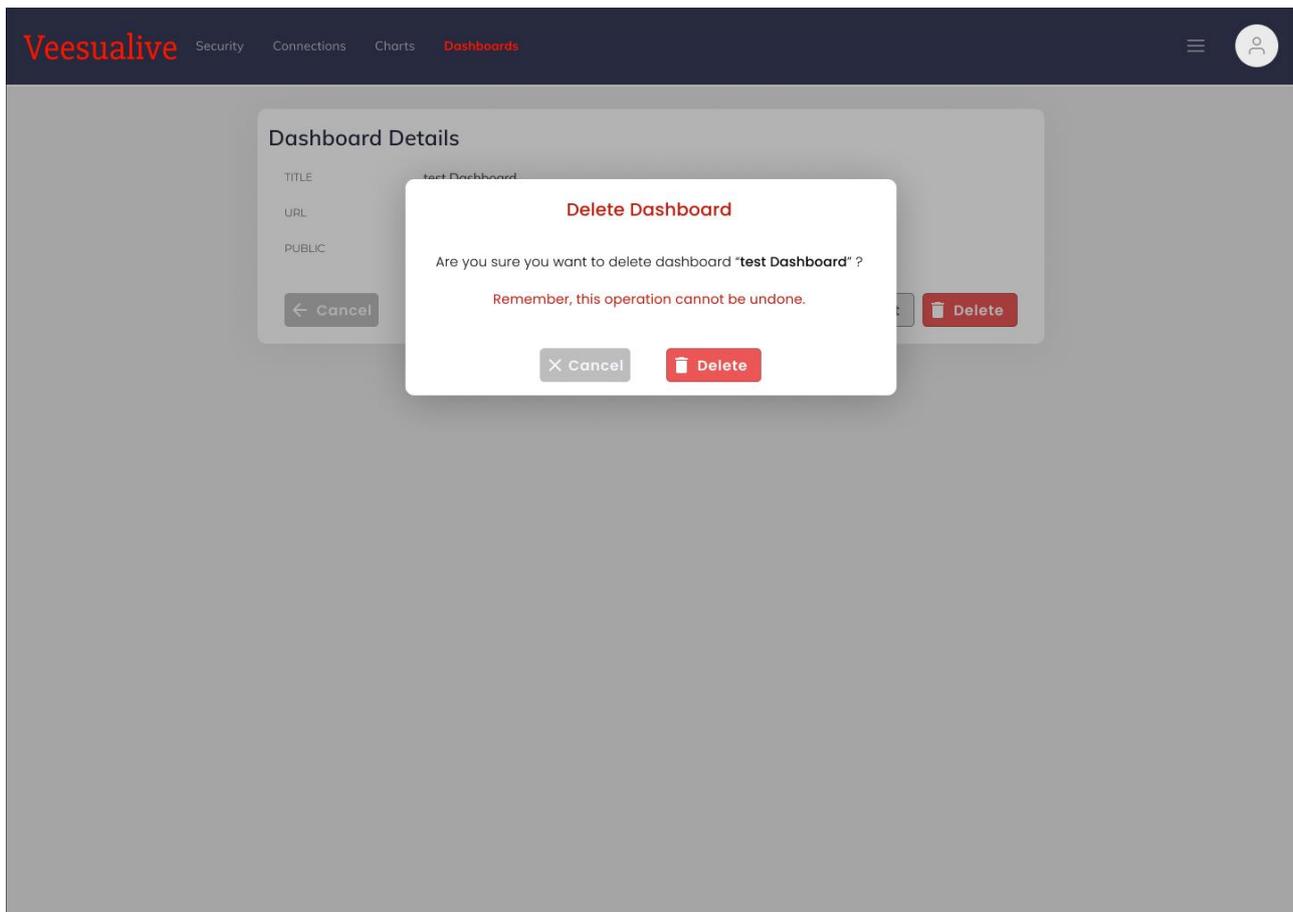


Figure 20 - 'Delete dashboard' option.

#### 4.1.4.4 Dashboard Customization

The actual dashboard composition happens in the Dashboard Editor, which can be reached clicking on the dashboard title.

The top of the Dashboard Customization page is represented by a toolbar containing the dashboard title, its published status, together with a set of control buttons to undo/redo the actions performed, cancel and discard all the changes, save the current status or access further customizations (Figure 21).

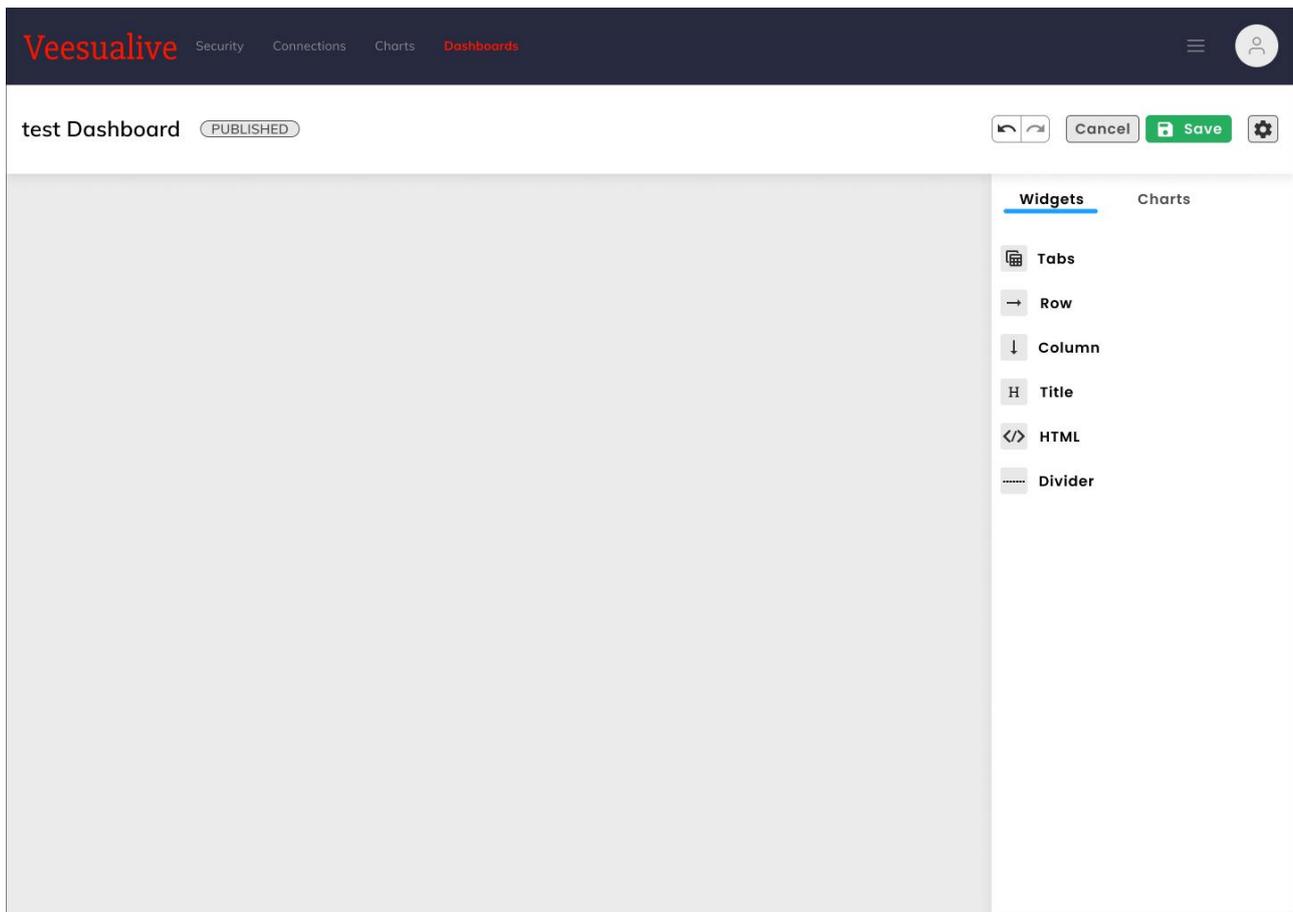


Figure 21 – Dashboard customization

Most part of the page is occupied by the Dashboard Editor. This editor is a live canvas in which the user can drag and drop the components that can be found on the right (Figure 22).

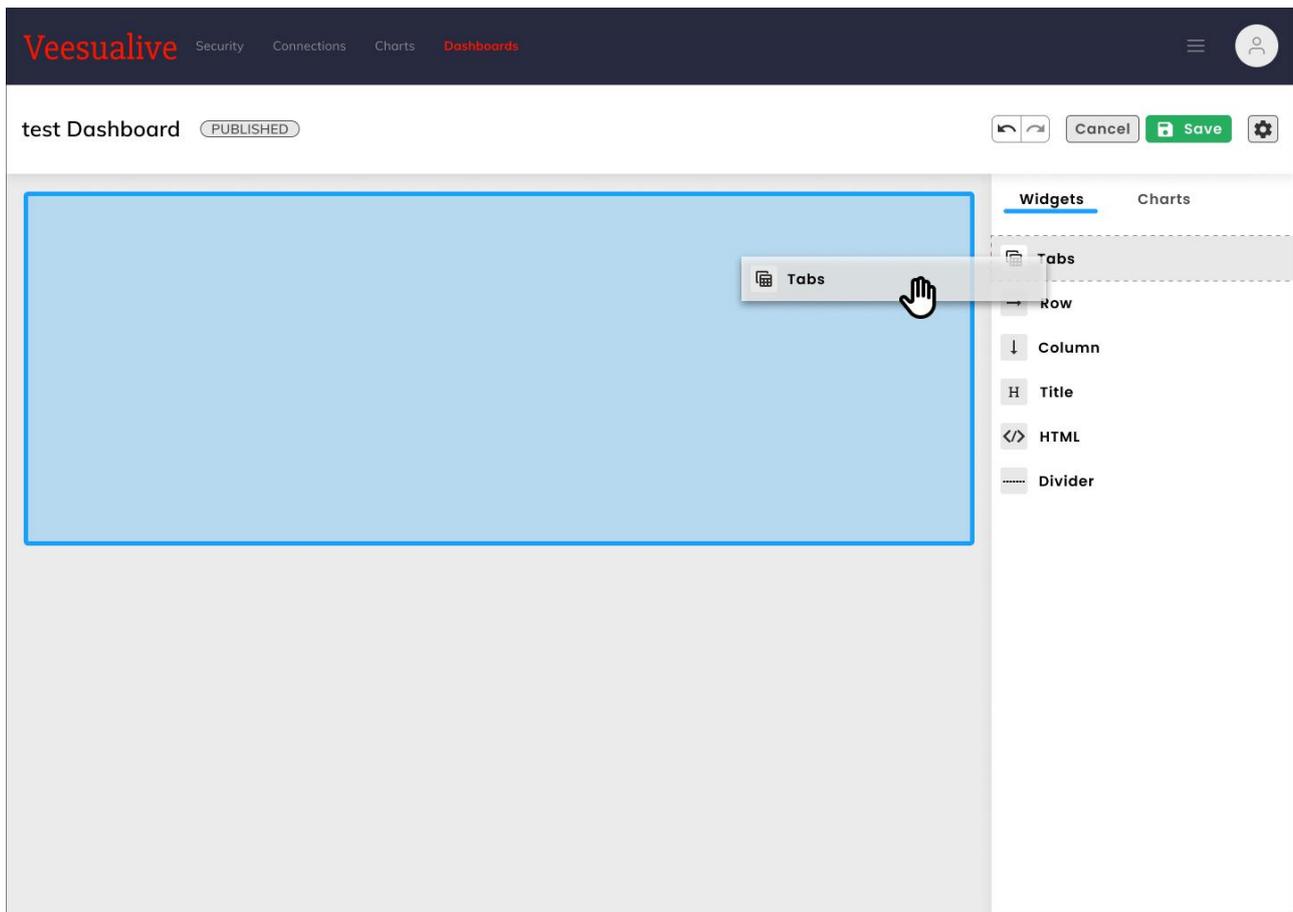


Figure 22 - Dashboard frames and components

Dashboard frames and components are divided into “Widgets”, which take care of the actual layout of the puzzle, e.g. Tabs, Columns and Dividers, and “Charts” the user has previously created (Figure 23).

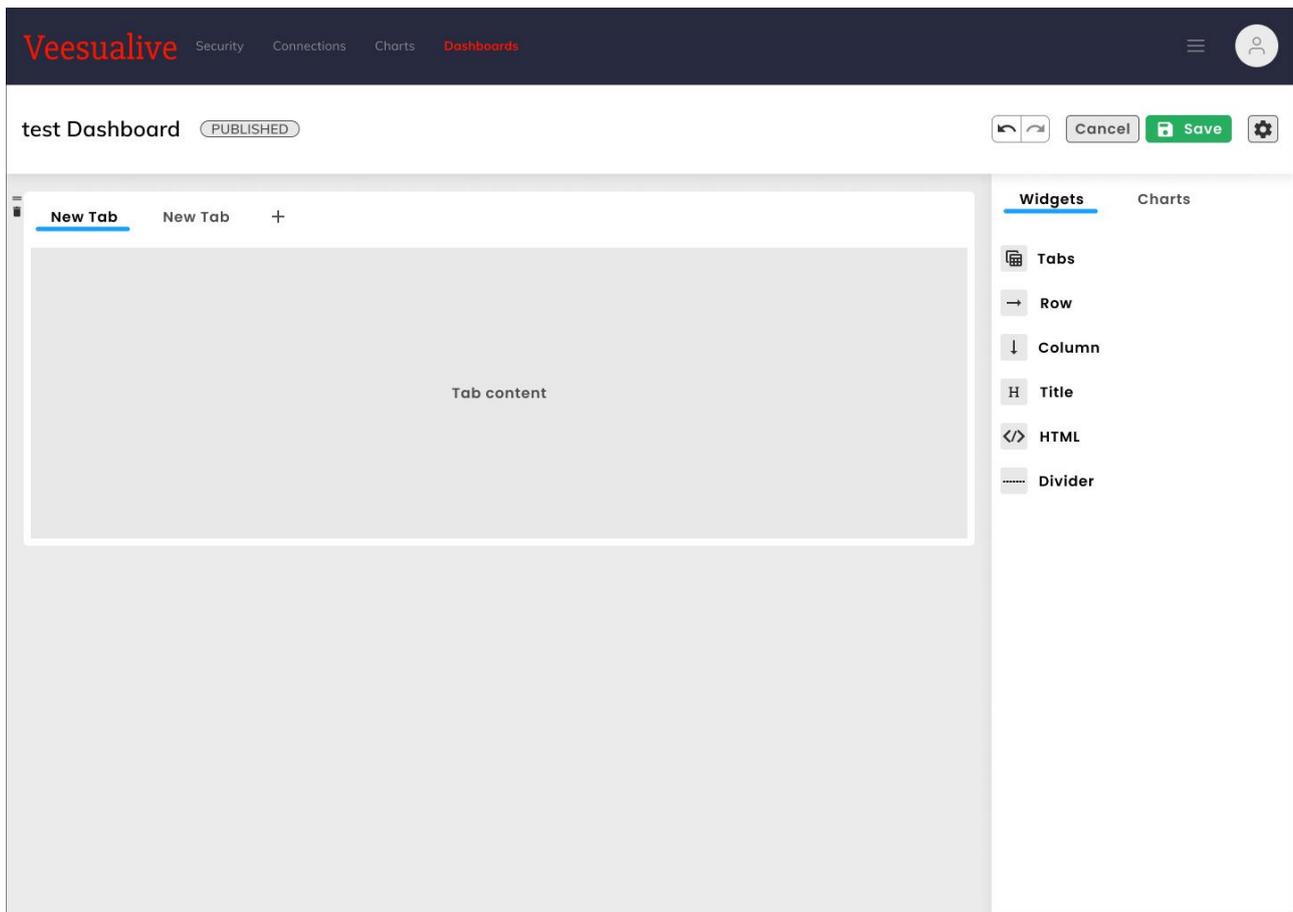


Figure 23 – Tab content

Once the user is happy with the overall structure, can add all the tables and graphs he/she wants among the ones at his/her disposal (Figure 24).

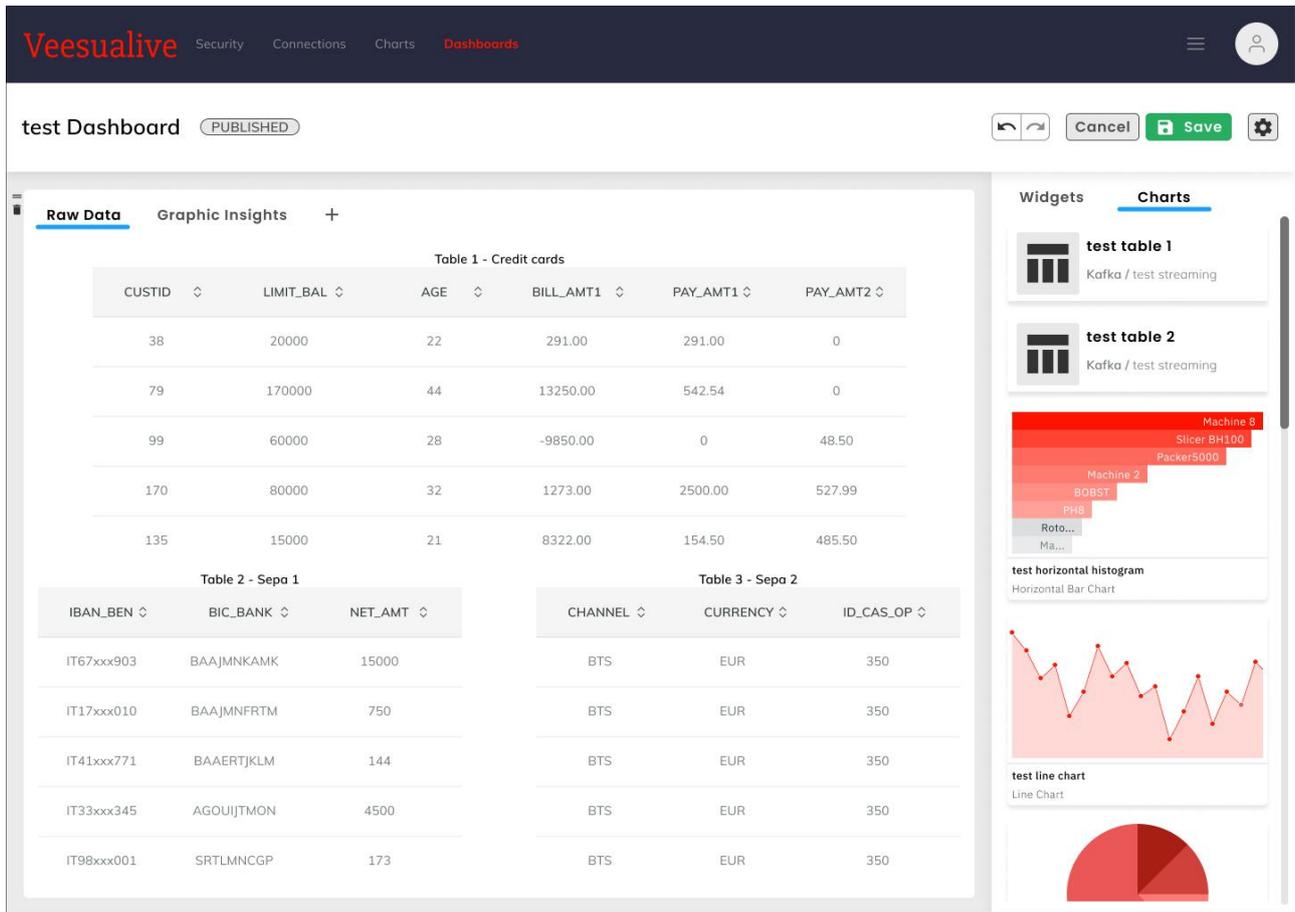


Figure 24 – Dashboard with “Raw Data” and “Graphic Insights” as an example

The end result shown in the example, is a dashboard demo where tables and charts are split among two tabs, “Raw Data” and “Graphic Insights” (Figure 25).

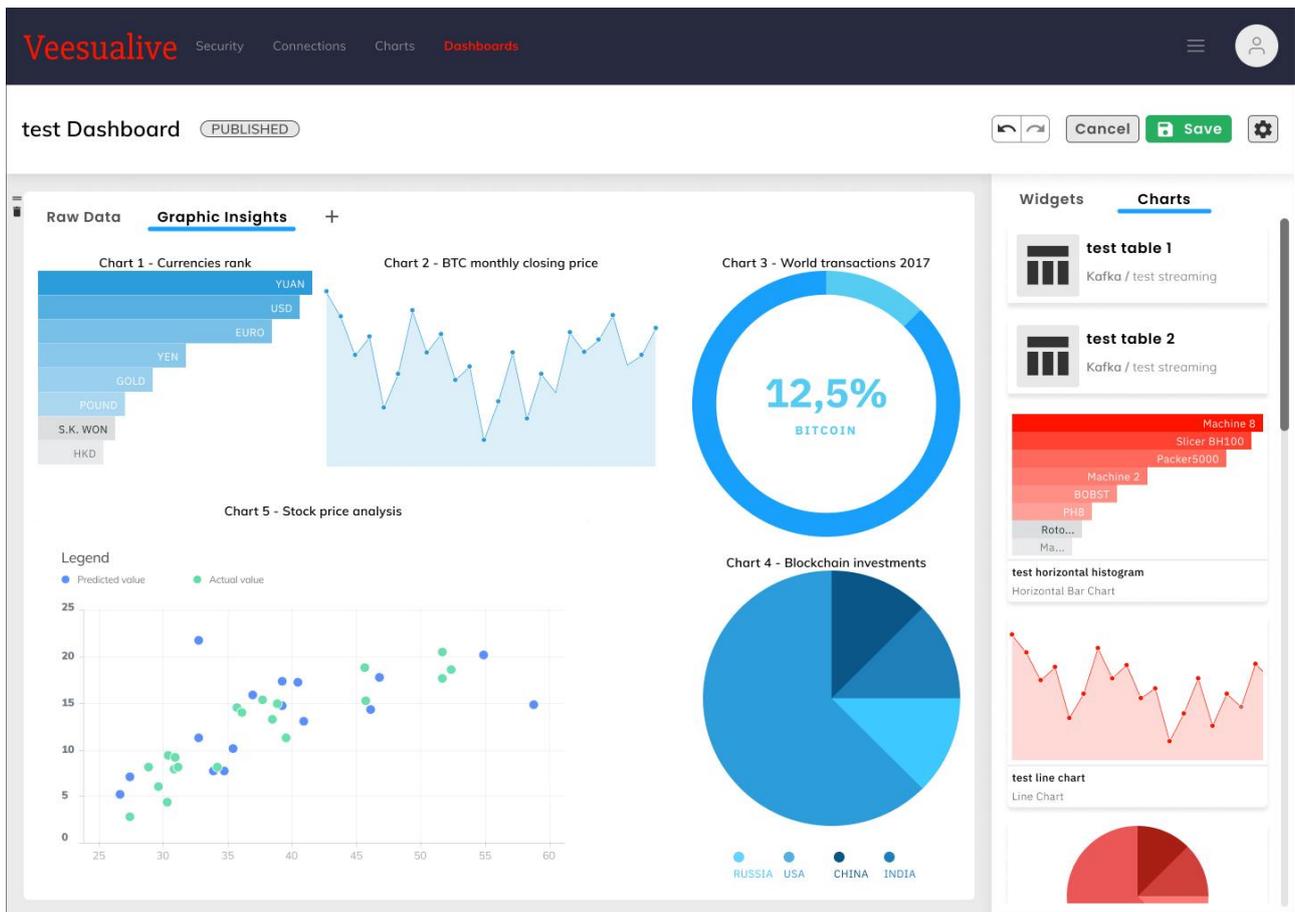


Figure 25 – Dashboard result

## 4.1.5 Security Section

Veesimalive security allows the user to monitor the overall visualization tool status. This section is divided in Users List, Roles List and Action Logs pages, accessible anytime thanks to the navigation tree on the left.

### 4.1.5.1 Users List

This page lists all the users registered in Veesimalive, their personal information, their role and account status (Figure 26).

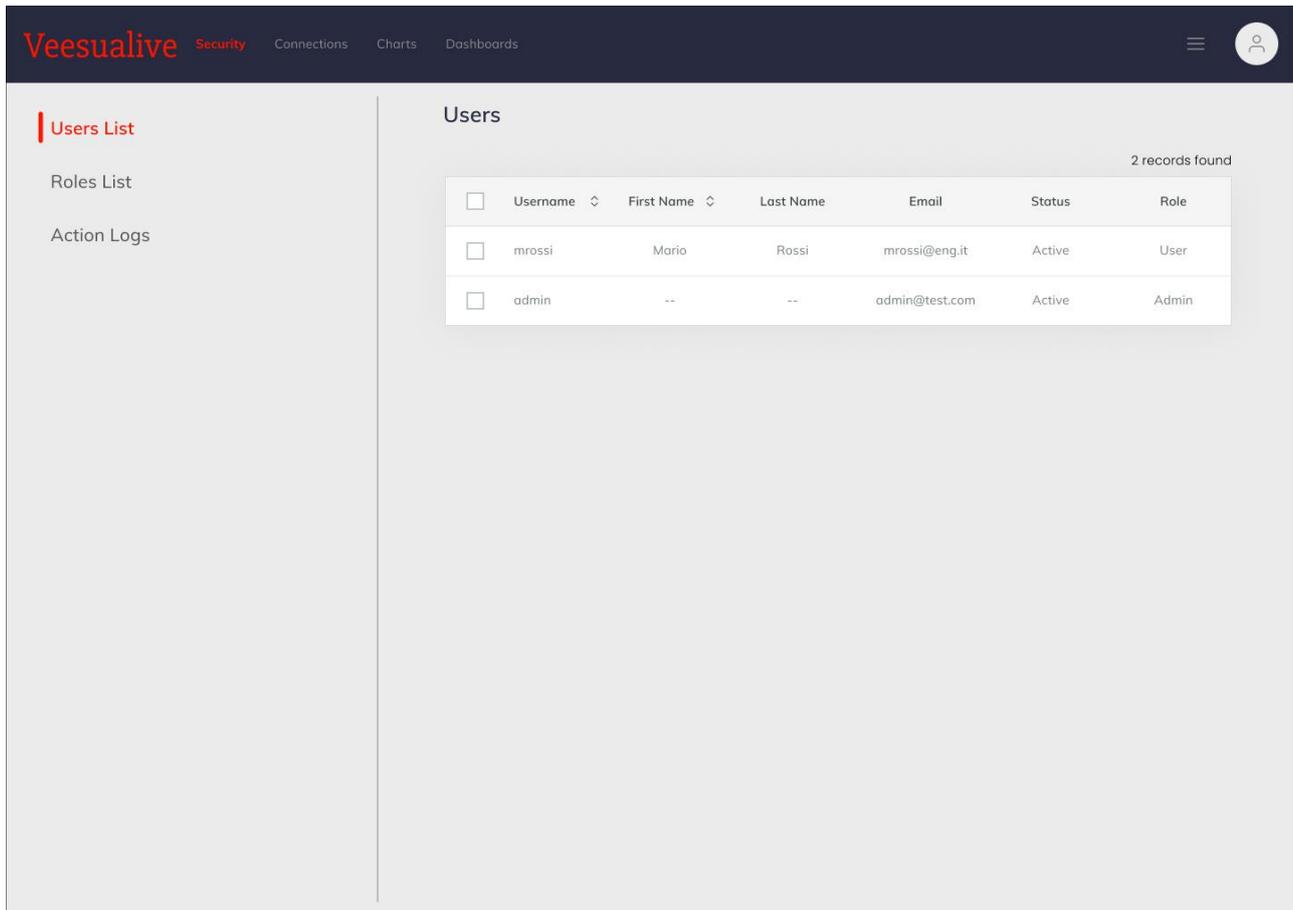


Figure 26 – User list

#### 4.1.5.2 Roles List

Roles list displays all the roles available in the platform (Figure 27).

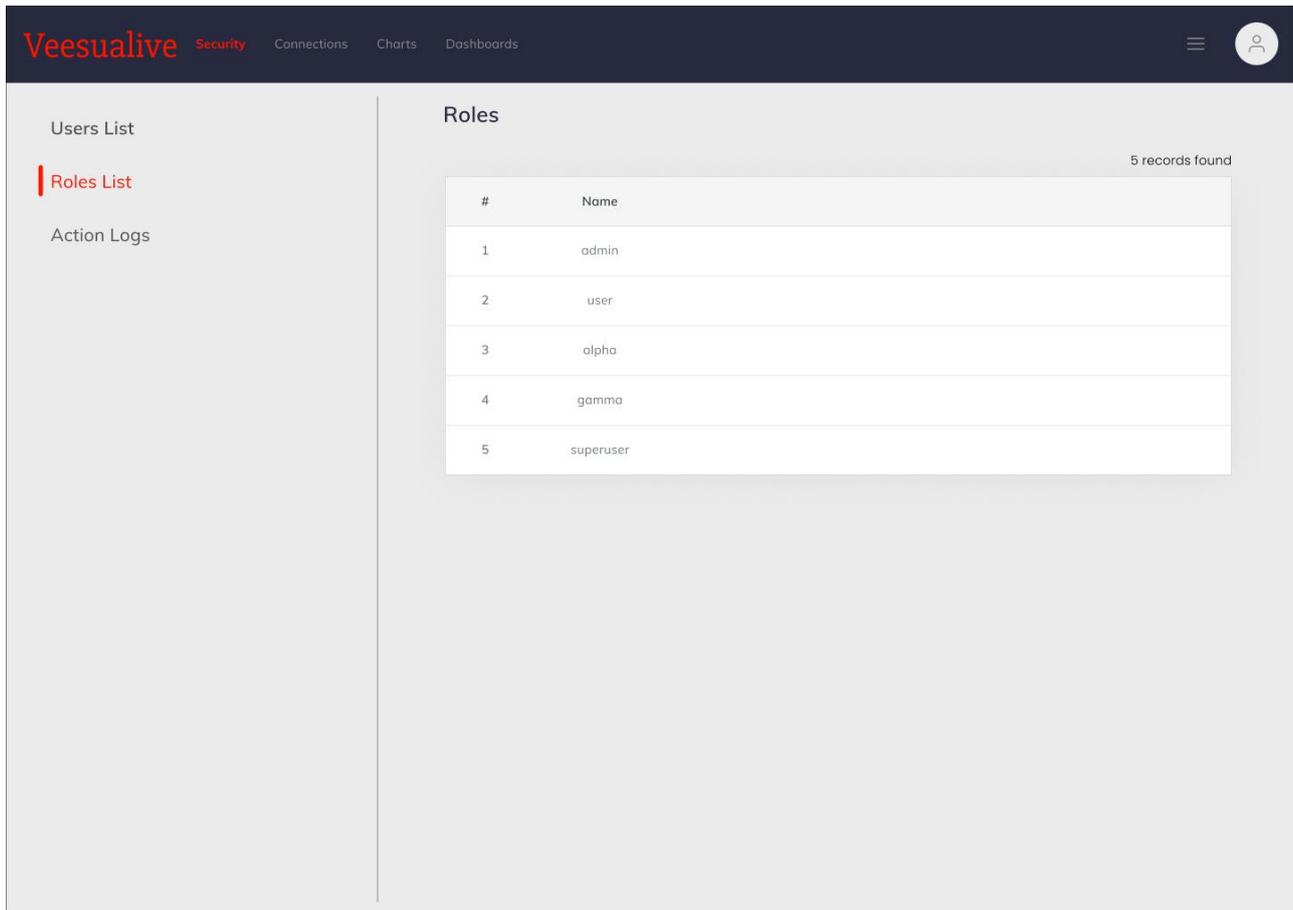


Figure 27 – Roles list

### 4.1.5.3 Action Logs

Action logs is the journal of the platform, where the user can check all the history of activities (Figure 28).

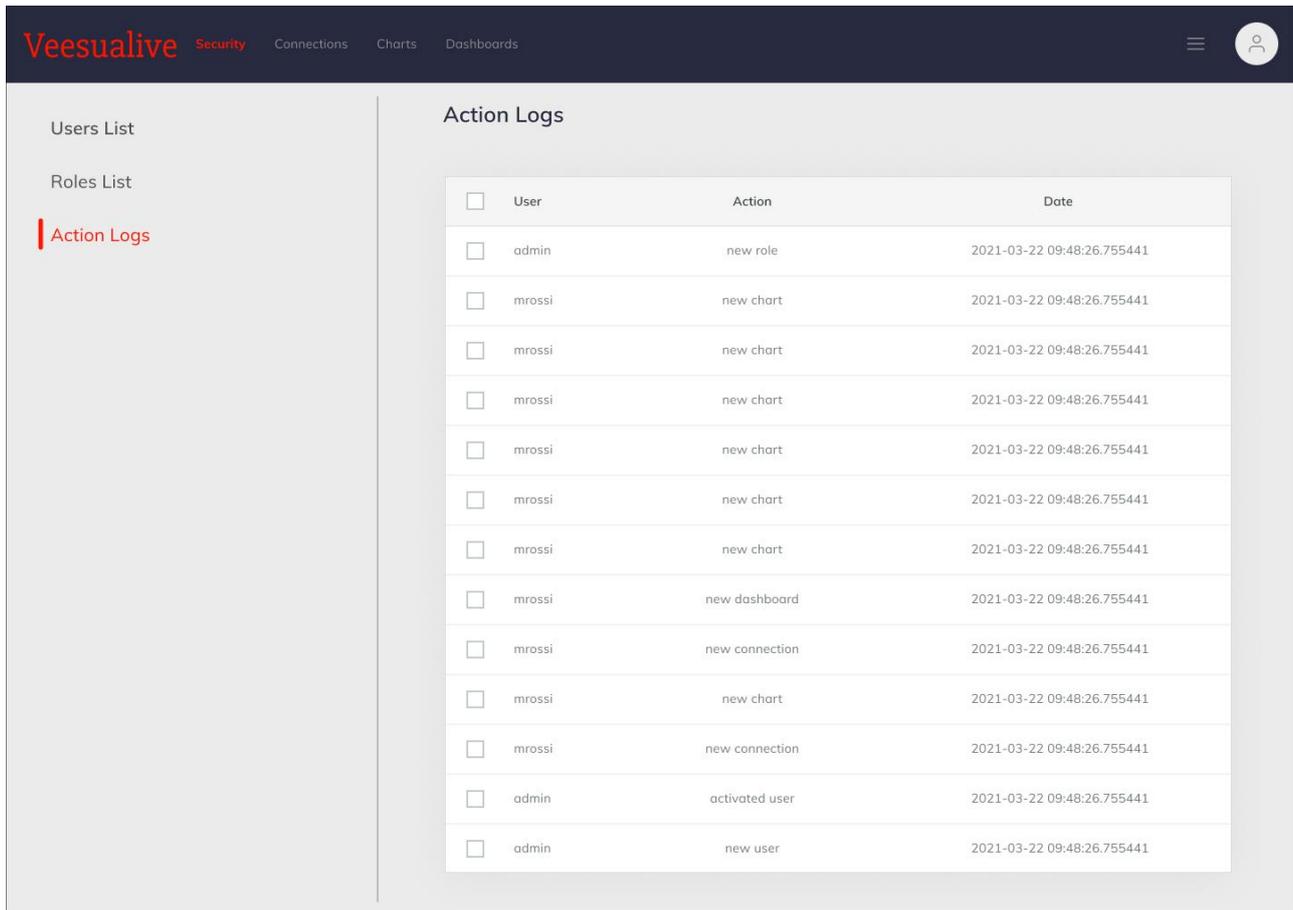


Figure 28 – Action log list

## 4.1.6 Account Section

Account section lets the user manage his/her own personal information, as well as account preferences and credentials.

### 4.1.6.1 User Menu

User menu is accessible through the application navbar, always on top of the window, with just one click on the avatar on the far right. Then, this drop-down menu opens, showing the user main info and two links, one to get to the account page, the other to perform the logout (Figure 29).

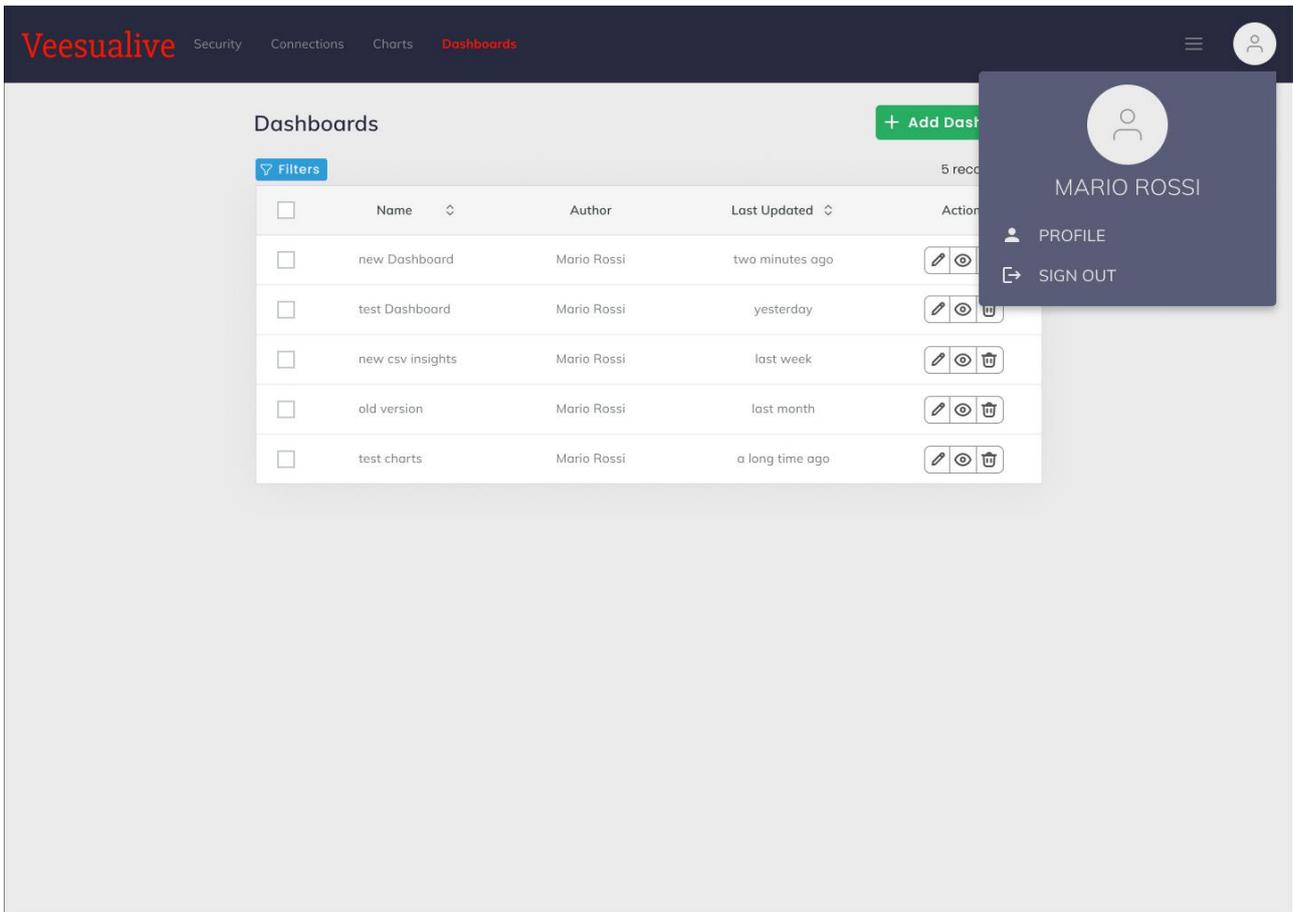


Figure 29 – User menu

#### 4.1.6.2 User Details

The profile information is organized in personal information and avatar.

User details and personal information showed in Figure 30 may be modified and the 'Reset Password' button allows the user to request the password modification.

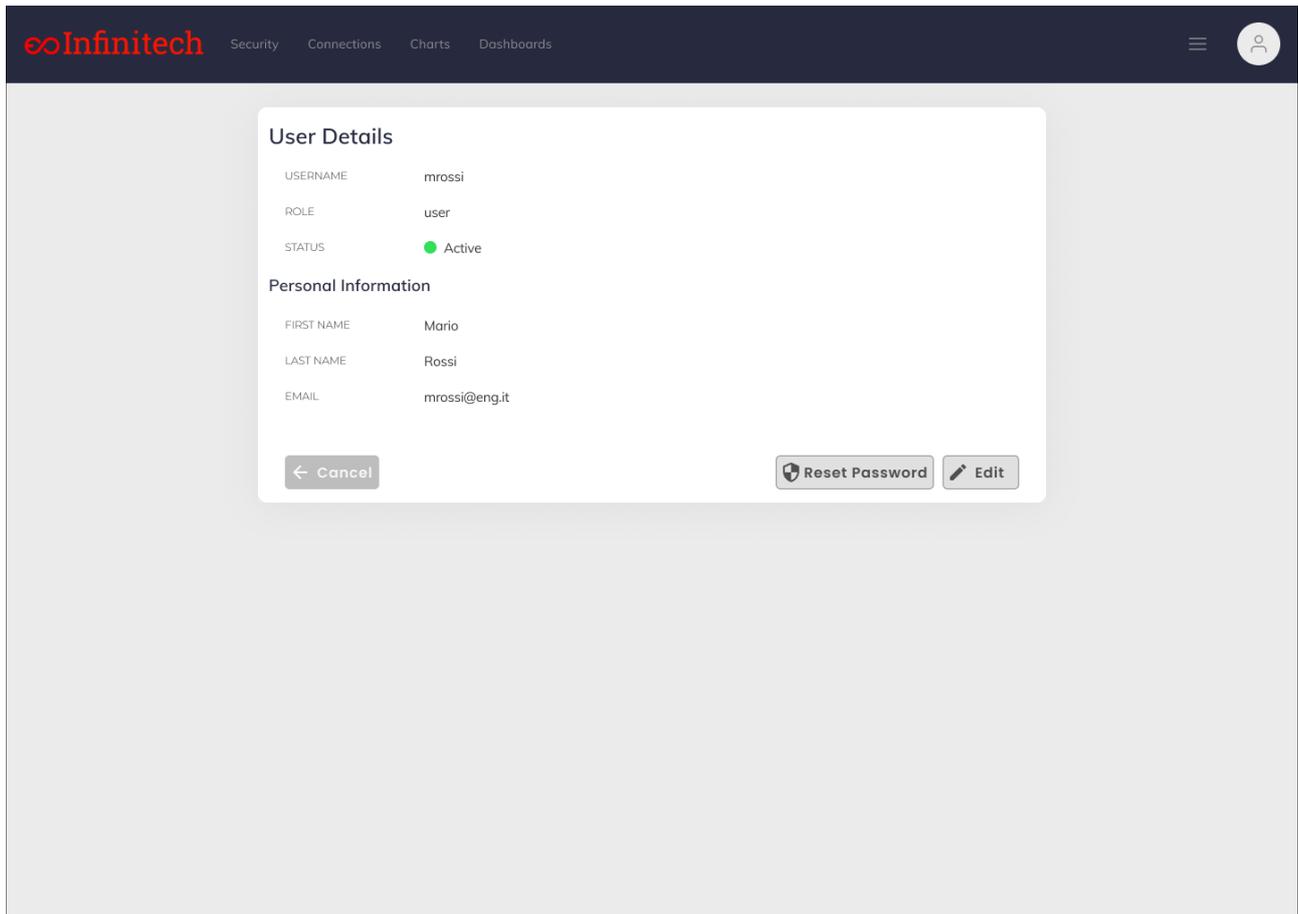


Figure 30 – User details

## 5 Conclusions

The purpose of this deliverable has been to deliver the first low fidelity, functional mock-up version of the visualization front-end component for aggregated information.

The deliverable is built on top of the main outcomes and the knowledge extracted from the WP2-3-4 and Pilot#10 feedback (within WP7) in order to provide the first version of Veesimalive, the visualization front-end component that will be tested and validated within the Pilot#10 sandbox.

Following an iterative design process, the high-fidelity mock-ups were designed. These mock-ups are providing the basis for the implementation of the visualization front-end as a customizable tool that allows a straightforward design of dashboards and an easy connectivity and fast visualization of data streams. Within these mock-ups, the main functionalities are presented. For each functionality, the design choices and the envisioned user interaction were presented.

The implementation of Veesimalive is a continuing process that will last till M30 when the final version will be delivered. This first version is provided for early assessment by the users and based upon the evaluation that will be received for the necessary enhancements and updates.