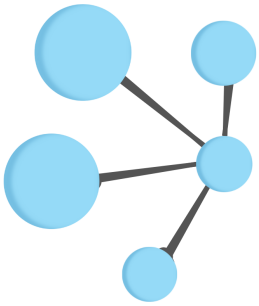


Telecommunications
S9 Advanced Project
Project Report

GraphVision

Authors:
Mohamed Bigi
Mohamed Lamghari
Soufyane Rahhal
Raja Safraoui
Alexandre Braudy
Asma Kadiri

Supervisors:
Frederic Denis
Alexis Carrelbilliard

January 31, 2024

Contents

Glossaire **2**

1 Introduction **3**
1.1 Origin of the Project . . . 3
1.2 Objectives and Avenues of Work . . . 3
1.3 Customer requirements . . . 3

2 Architecture **4**
2.1 General architecture . . . 4
2.2 Technologies . . . 4
2.2.1 Backend . . . 4
2.2.2 Frontend . . . 4

3 Development and Results **6**
3.1 Server side . . . 6
3.1.1 Conversion to JSON . . . 6
3.1.2 Conversion to pivot format . . . 6
3.2 Client side . . . 7
3.3 Graph construction . . . 7
3.4 Filters construction . . . 9
3.5 Technical Issues and Directions . . . 9

4 Project Management **10**

5 Conclusion **10**

Annex - Web app **12**
5.1 Application overview . . . 12

Glossary

- **JSON :** A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format. It is primarily used for transmitting data between a web application and a server.
- **Db :** database
- **id :** identity
- **API :**An API or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate with each other.
- **NoSQL :** NoSQL database is a type of non-relational database that allows for flexible, high-performance storage and retrieval of unstructured data.
- **Vis.js :** javascript library to display the graphs.
- **vue.js :** JavaScript framework for building user interfaces.
- **v-bind :** the "v-bind" directive is used to bind data from the Vue instance to an element's attribute.
- **$emit :** the "$emit" method is used to trigger an event on the parent component from a child component.
- **$root :** In Vue.js, the $root property is a reference to the root Vue instance of the application. It can be used to access the data and methods of the root Vue instance from any component in the application.

1 Introduction

1.1 Origin of the Project

The need comes from an Orange team in charge of operating hundreds of applications, physical, virtual or containerized. The diversity of applications, infrastructures, interlocutors, analysis tools and deployment on the servers makes it difficult for one person to apprehend the entire perimeter, particularly in an emergency.

The team wishes to have a graphical visualization tool to display the different information (applications, servers, etc.) in the form of nodes linked together, which represent the different relationships. Thus being able to display all the useful information of each node in order to facilitate access to information despite its quantity.

1.2 Objectives and Avenues of Work

The project objectives are:

1. Displaying user information as nodes
2. Representing relationships as links between nodes
3. Importing any type of data repository by the administrator
4. Filtering the data by various filters chosen by the user
5. Displaying the size of a node according to the number of its relations
6. Displaying different groups of information by different colour

1.3 Customer requirements

In order to successfully complete this project and meet the client's needs, some functional and non-functional requirements were specified, as the following :

Functional requirements:

1. Graphic visualization
2. Multi-user interface (admin / user)
3. Processing of the user's repository before its display
4. Filters (color/size/data)

Non-Functional requirements:

1. Demonstration
2. application package
3. source code
4. generic product
5. non-adherent product
6. understandable deliverable

2 Architecture

2.1 General architecture

The project is structured as shown in the graph below

The web interface handles interactions with the user, and is also in charge of visualizing the graphs stored in the database.
The main goal of the project, is to be able to not only visualize multiple graphs, but to allow certain flexibility to the user, where they can either upload graphs for visualization, choose certain graphs to visualize, and also customize their visualization using certain filters.
Thus, in order to guarantee these functionalities, the web interface has to interact with another entity, that would be in charge of implementing them, and that is the role of the server
The server, which represents the core of the entire project, is responsible for implementing most of the functionalities that the project relies on. It communicates directly with the web interface, in order to give instruction as to what to visualize based on the user's choices.
The server also communicates directly with the database. In fact, the server receives data from the web interface, and relays it to the database for storage. Only to retrieve the same data later on, and relay it to the Client for visualization.
The database has the basic functionality of storing the desired data, to be accessed by the user whenever he desires to do so.

2.2 Technologies

2.2.1 Backend

Backend technologies play a crucial role in the development of web applications as they handle the server-side operations such as database management, API creation and processing HTTP requests. The backend of the system utilizes Node.js and MongoDB as its primary technologies [3].

Node.js is a popular technology used for building the backend of web applications. It is a JavaScript runtime built on Chrome's V8 JavaScript engine, which allows developers to run JavaScript on the server-side. Node.js is open-source, cross-platform, and lightweight, making it a great choice for building scalable and high-performance web applications. [6]

MongoDB is a NoSQL database that is also open-source and cross-platform. It is a document-oriented database, which means that it stores data in a JSON-like format. This allows for a more flexible and scalable data model, as it can handle unstructured data and handle data changes without affecting the existing schema. MongoDB also has built-in sharding and replication, which allows for easy horizontal scaling and high availability. [1]

The combination of Node.js and MongoDB was chosen for this project as it allows for efficient and fast data processing and storage, making it ideal for building dynamic and responsive web applications, Node.js allows for the creation of high-performance server-side logic, while MongoDB provides the flexibility and scalability needed to handle large amounts of data.

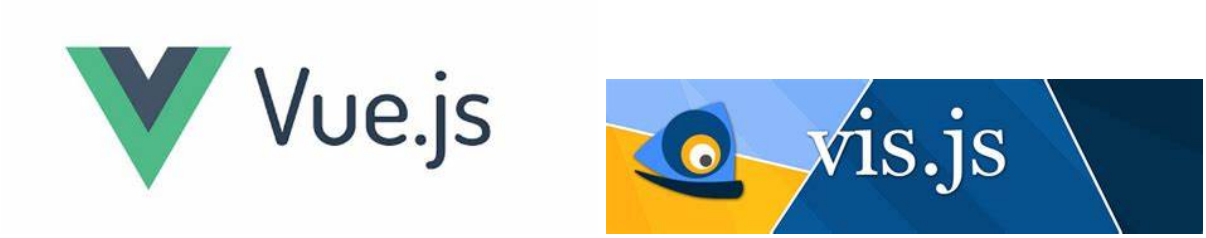
2.2.2 Frontend

Front-end is the part of web development that deals with the user interface, or the client-side of a web application. It involves using HTML, CSS, and JavaScript to create the visual and interactive elements of a website or web application that users interact with [4]. To build this interface, the framework **Vue.js**

was chosen (Figure 1).

Vue.js is an open-source JavaScript framework for building user interfaces. One of Vue's key features is its ability to reactively update the view when the underlying data changes, making it a popular choice for building dynamic user interfaces. Vue also provide a powerful Single-File Components that allows you to define the template, script, and styles for a component all in one file, making it easy to organize and reuse code [5].

As the Orange group demanded that their data is to be presented in the form of nodes and link them together, a javaScript library called **Vis.js** ((Figure 1) was considered the most suitable to achieve this goal.



(a) Vue.js (b) Vis.js

Figure 1: Vuejs Visjs

Vis.js can be used to create interactive network diagrams, which are useful for visualizing connections between various entities such as people, organizations, or systems. The library provides a variety of options for customizing the appearance and behavior of the network diagram, such as the ability to change node and edge colors and shapes, add labels, and define custom interactions [1].

The library provides a Network class that allows initialization and customization of the network diagram. The programmer can set the data for the nodes and edges, define the layout, and add event handlers for user interactions. The library also provides methods for manipulating the network data and layout in real-time, such as adding, removing or updating nodes, edges and groups.

3 Development and Results

3.1 Server side

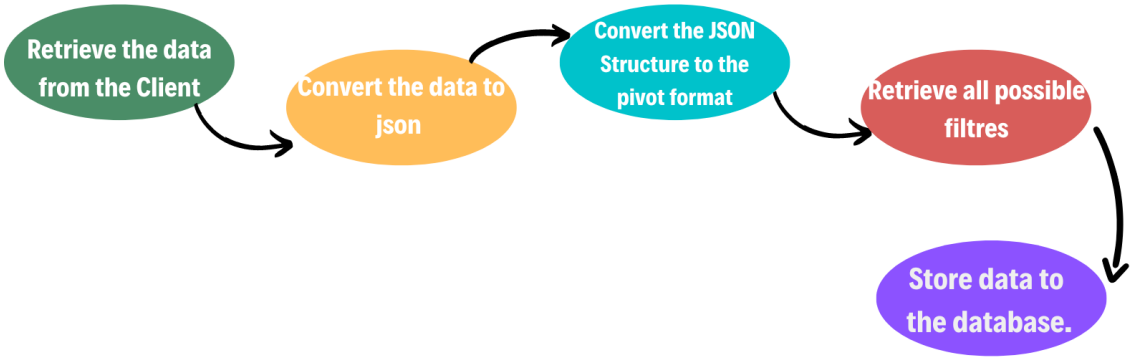


Figure 2: Backend services

The main challenge of the project, is to have a generic product, able to visualize any type of graph, represented under any format.
To remedy this issue, the data sent from the users, via the web interface, to the server and vice versa, must have a unified format.

3.1.1 Conversion to JSON

With *JSON* being the most used and reliable format for exchanging data, it was chosen as the basic format, in which data will be sent from the users to the server, and vice versa.
However, in order to avoid hindering the user experience, the web interface allows the users to upload their desired files in the format they wish, *(i.e. JSON, excel or csv)*, and an extra layer is added that converts the uploaded files, no matter their format, to a *JSON file*, that is then sent to the server.
This conversion allows flexibility to the user, and expands the range of data types that can be visualized by the web interface.

3.1.2 Conversion to pivot format

In general, a format describes the manner in which data is structured within a file. And a *pivot* format, is one of the many ways data can be organized inside a file.
In order to successfully visualize the graphs, the client side must receive data from the server, not only as a *JSON* file, but the data within the *JSON* file must be structured in a specific format, i.e. the *pivot format* mentioned above.
Thus, an extra layer of file processing is added within the server. This layer is in charge of receiving the uploaded *JSON* files, sent by the user, and converting them to the *pivot format*, before being stored in the database to then be sent to the web interface for visualization.

3.2 Client side

The user interface is built in such a way that it will be easily usable by the user. The project consists of three pages :

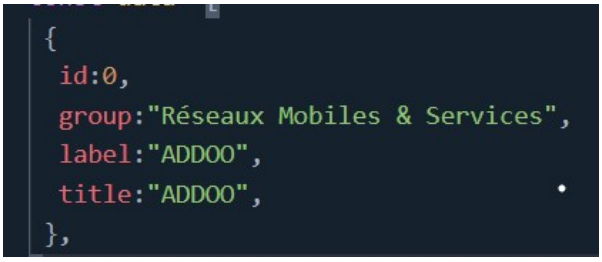
- **The main page (figure 8)** : The main page is divided into three areas :
  - **The header** : the header bar is a common component between all pages, since it contains the buttons to navigate from one page to another. The first button is GRAPHVISION, it allows navigating to the main page. The second button is the **ADMINISTRATOR** button which allows navigating to the Admin page and finally the **LOGIN** button to navigate to the authentication page.
  - **The filter area** : It is the component responsible for the construction of the filters chosen by the user. It is essentially composed of **selection bars** that allow displaying the possible values for each filter once the bar is clicked, and a **Search** button that triggers an API to the back to receive the filtered data.
  - **The network area** : It is the Network component provided by the **vis.js** library and used to display the graph. In fact, this component needs two types of data; an array of edges and an array of nodes. Once these two arrays are implemented by the back and sent to the front, the component displays the graph.
- **Login page (figure 9)** : The Login page allows the authentication of the admins who have the possibility to import files and to choose the parameters of each graph.
- **Admin Page (figure 10 et 11)** : This is the page where the user can import a file whatever its extension and once he clicks on the **SUBMIT** button, he will be redirected to the parameters page to choose the fields of his graph, as well as the filters to filter his data

The construction of the front-end is done in such a way to respect the dynamic aspect of the project. Therefore, the communication between components was essential in order to transmit the data entered by the user as well as the events from one component to another. In this context, the **vue.js** framework was very practical thanks to the easy communication it provided either between father-son components with the **v-bind and $emit** options or between any components with the **this.$root** function. [7]

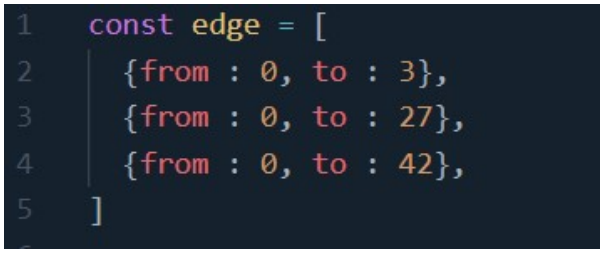
3.3 Graph construction

To display the graph, the first step is to build the nodes and edges.To do this, it is necessary to reform the initial data that received from the orange team, which is an array of objects and each object represents an application(Figure 3), to two arrays; the first for nodes and the second for edges (Figure 4).

Figure 3: Initial data



(a) table of nodes



(b) table of edges

Figure 4: Nodes Edges

For the table of nodes, each node has four fields, the first one is its ID, the second one is its group, because each group has a specific color in the graph, the third one is its label and the last one is the text that is displayed when the user click on the node. For edges, each one has two fields, the first one is the ID of a node and the second one is the ID of another node that links to the first one.

After the preparation of these two tables in the back-end, they will be sent to the front-end to display the graph using the **vis.js** library. This communication is done with the help of **API** according to the following figure (Figure 5).

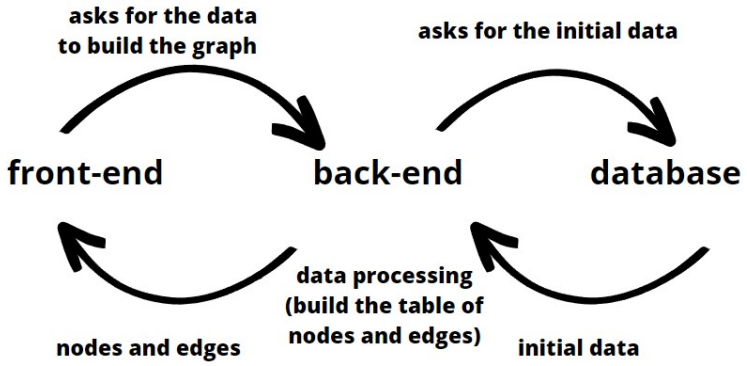


Figure 5: Communication between back, front and db entities using API

The front-end sends an API to the back-end to ask for the necessary data to build the graph, then

the back-end asks the db for the initial stored data and does the necessary processing to build the table of edges and nodes. Finally, it sends them to the front end to display the graph using vis.js.

3.4 Filters construction

The user can choose the filters he wants to filter his data. In fact, once the data file is downloaded, an API is sent to the back-end to read all the possible fields and send them to the front-end so that the user can choose his filters.

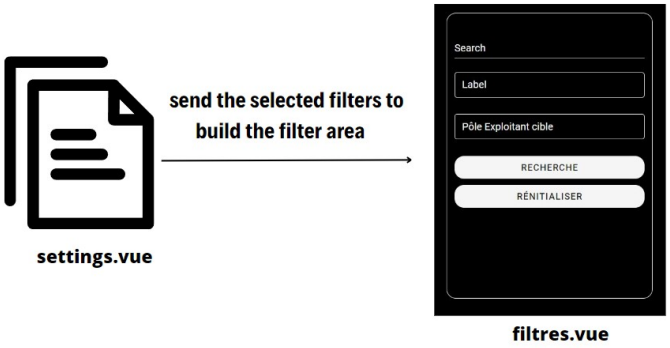


Figure 6: construction of the filter area

The choice of filters is done in the **settings.vue** file, then an event is sent to the **filters.vue** file in order to build the filters area. Figure6 shows an example where the user has chosen two filters : **Label et Pôle exploitant cible**.

After the user chooses his filters, this data is sent to the backend level to filter the initial data and build new tables of nodes and edges and build the new graph (figure 7).

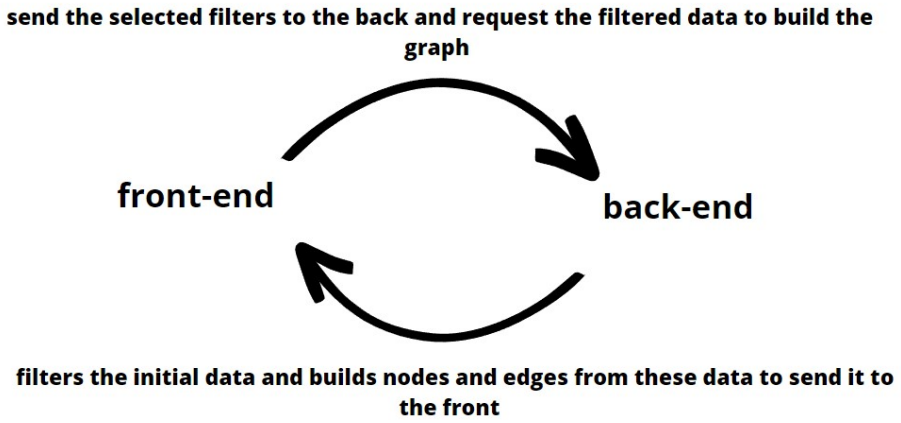


Figure 7: transmission of filters

3.5 Technical Issues and Directions

Initially, the team had considered using Neo4j as the graph database for the project. Neo4j is a powerful and flexible JavaScript library for visualizing the type of data that is being used. However, after conducting further research, testing, evaluating the available documentation and resources, it became apparent that Neo4j would not be the best choice. Despite its powerful capabilities and visually appealing results, the lack of a comprehensive documentation and community support made it difficult to implement

and maintain the database effectively.

As a result, it has been decided to pivot and explore alternative options. After evaluating various options, MongoDB was ultimately chosen to be used as the primary database. While MongoDB is not a traditional graph database, it does offer similar functionalities and was able to effectively meet the project's needs. One of the key benefits of MongoDB is its extensive documentation and active community. This made it much easier to understand and implement the database into the system. Additionally, MongoDB offers powerful querying capabilities and allows for easy scaling, which are both essential for the project.

4 Project Management

In order to effectively manage the development of the web application for the client, the agile method was implemented. This approach allows for flexibility and responsiveness to the changing needs of the project, and also helps to continuously monitor the progress. To accomplish this, weekly meetings were held with the client to review progress and receive feedback. During these meetings, demonstrations of the current state of the application were presented, which allowed the client to see the progress made and to provide valuable insights. Additionally, the client sets new objectives for the next week in the form of user stories, which describe the features that need to be developed from the point of view of the user. These user stories are then assigned a complexity level, which helps with prioritizing the development efforts and ensure that the most important tasks are being developed. Overall, this approach helps with staying on track and ensure that the project is moving in the right direction.

In addition to the weekly meetings with the client, the team has a strong organizational structure. Trello is used to organize and subdivide the tasks, which helps with keeping track of the progress of each individual task and see how it is being developed. We also use Microsoft'Teams for internal meetings, which allows for constant connectivity and effective collaboration, even when working remotely. This with staying on the top of the desired progress and making sure that client's expectations are met. Overall, the organizational structure is designed to help staying on track and deliver a high-quality web application that meets the needs of the client.

5 Conclusion

The S9 projects are an opportunity for most students to take part in group projects for the second time. The GraphVision team developed a team spirit and achieved its goals through dedication and hard work.

In conclusion, the development of this web application has been a challenging but rewarding experience. We have successfully implemented the agile method to manage the project, which has allowed us to stay flexible and responsive to the changing needs of the client. We have also implemented strong organizational structures within our team, using tools like Trello and Teams, which have helped us to stay on track and deliver a high-quality web application that meets the client's needs. The graphical visualization tool that we have developed will enable the Orange team to manage their workload more efficiently and effectively, and will help them to have a better understanding of the entire system in emergency situations. Although the project is completed, it lacks some last features that we were planning to implement. However, we believe that the final product will still be beneficial for the client and will meet their initial needs. Overall, we are proud of the work that we have accomplished, and we are confident that our client will be satisfied with the end result.

All GraphVision members sincerely thank the Orange team, especially Fréderic and Alexis who have been very supportive and has given a lot of advice throughout the project. Finally, we truly thank the school for giving us the chance to work on this project.

References

[1] Literature on VisJs website : `https://visjs.org/`

[2] Literature on MongoDB website : `https://www.mongodb.com/docs/`

[3] Literature on techtarget website : `https://www.techtarget.com/whatis/definition/front-end`

[4] Literature on front-end masters : `https://frontendmasters.com/guides/front-end-handbook/2018/what-is-a-FD.html`

[5] Literature on wikipedia : `https://fr.wikipedia.org/wiki/Vue.js`

[6] Literature on nodejs.org : `https://nodejs.org/en/about/`

[7] Literature on vuejs doc : `https://vuejs.org/guide/introduction.html`

Annex - Web app

5.1 Application overview

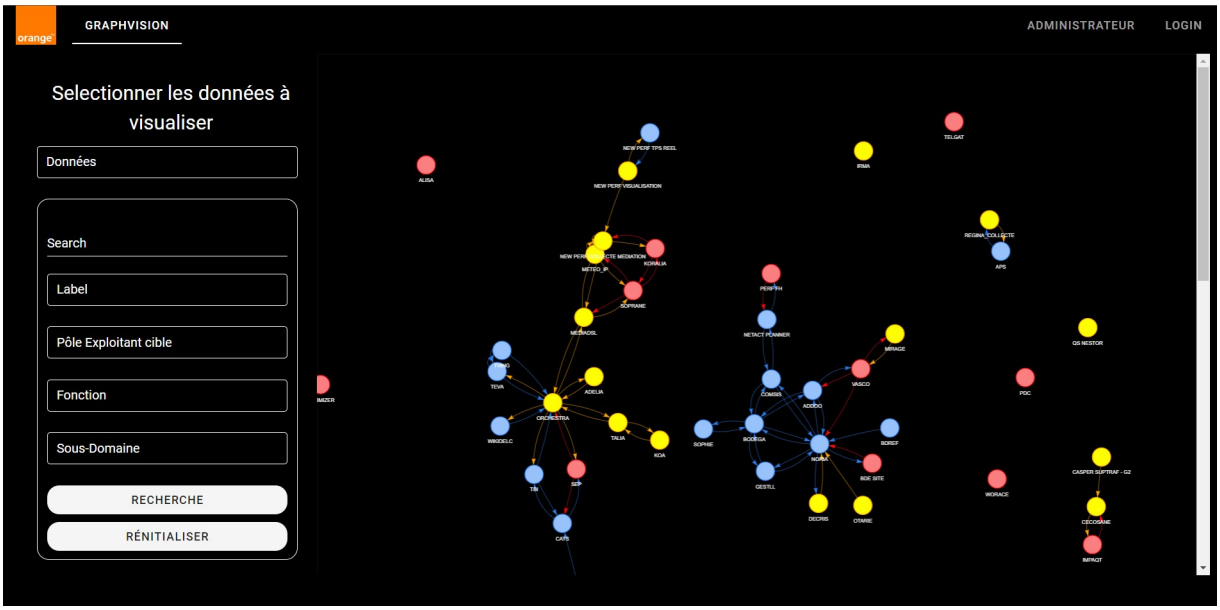


Figure 8: main page

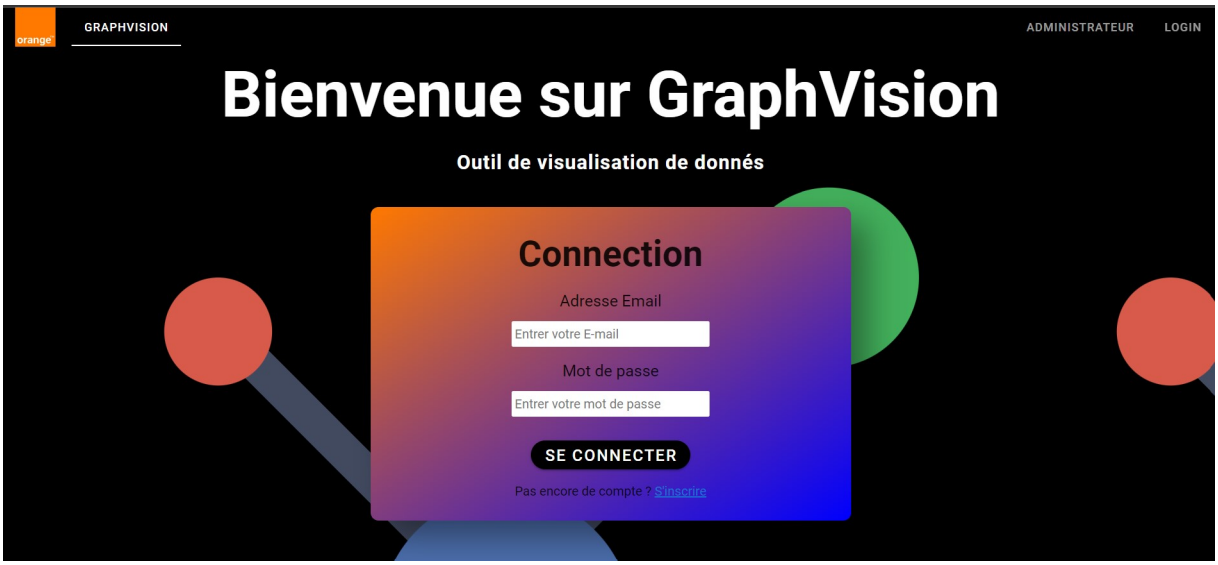


Figure 9: Login page

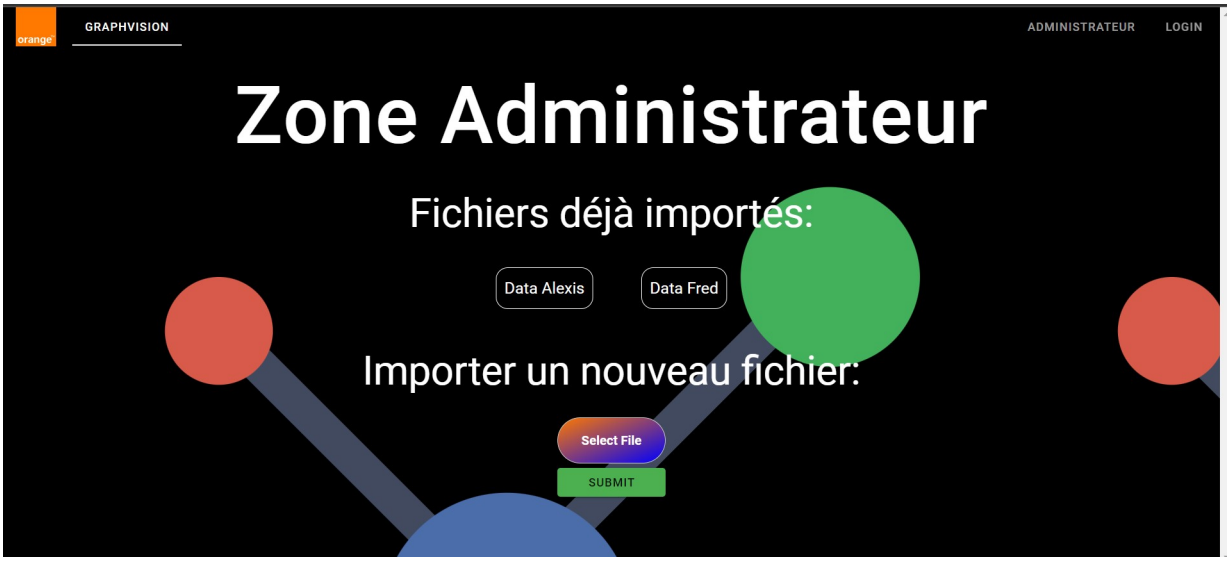


Figure 10: Login page

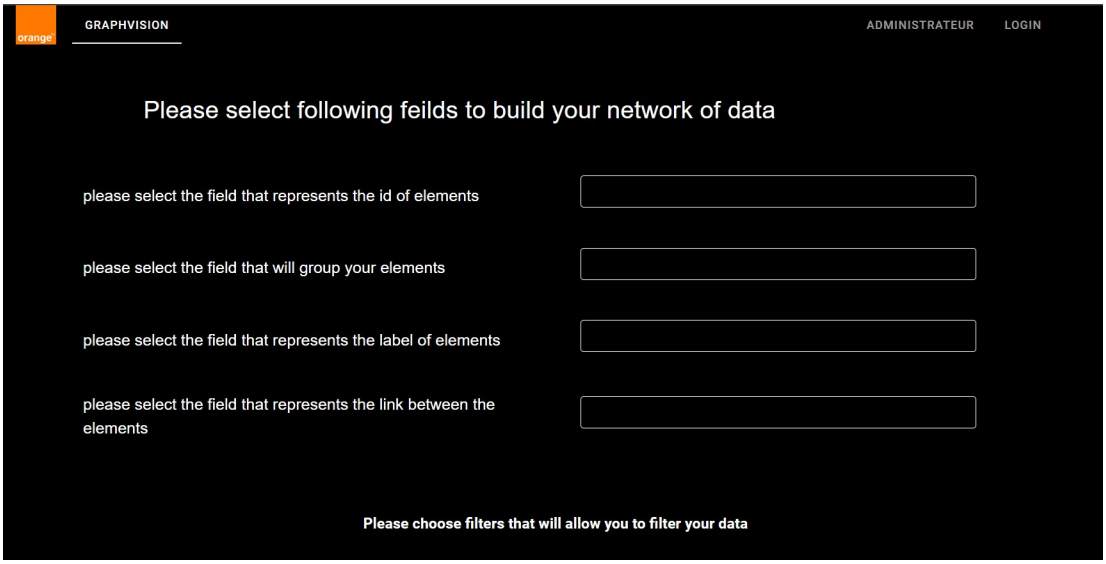


Figure 11: Login page