Universidad Zaragoza

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

REAL-TIME EARTHQUAKE MONITOR

Sistemas y tecnologías web (2021/2022)

Mathis BOSTON - 867063@unizar.es

Summary

1 Access

1.1 Online

The app is hosted on Netlify and Heroku. The front end is hosted on Netlify and the back end on Heroku.

The backend is hosted at the following URL: https://earthquake-monitor-server.herokuapp.com/

The app is accessible from the following URL: https://main--earthquake-monitor-stw.netlify.app/

A demo account is available : Username : test Password : testtest

1.2 Local

The project can be downloaded and run locally.

$ git clone https://github.com/Tiss971/earthquake-monitor

- Run Backend

$ cd earthquake monitor/server
$ npm run dev

- Run FrontEnd

$ cd earthquake monitor/client
$ npm run dev

2 Description of the idea :

A French singer (Nekfeu) said : " Et si les cyclones étaient la somme de tous les derniers soupirs de la terre ? Et si les tremblements de terre étaient la somme de tous nos premiers pas ? Ca voudrait dire qu'on est tous liés." which can be translated by : "What if the cyclones were the sum of all the last sighs of the earth? What if the earthquakes were the sum of all our first steps? That would mean that we are all bound". These words inspired the genesis of this project.

1.3 Aims :

The goal of this project is to make it possible **to visualize all the earthquakes which take place on all the surface of the globe**. The user can then adjust certain parameters to navigate through the data and visualize what interests him.

These earthquakes will be the way to connect people living on both sides of the globe who would never have had the opportunity to discuss otherwise. Indeed, the goal is also to **allow discussion with people located more or less far from a recent earthquake**.

Finally, the site will **provide information** on these natural phenomena and possibly **educate** some people on the instructions to follow during one of these phenomena.

1.4 Targeted audience :

The site is intended for **all audiences**. Visualizing earthquakes may interest only **curious people and occasional visitors**. While some people looking to chat will have the opportunity to talk with people from all over the world and they will **regularly use as a message box and a place to discover new profiles.** The educational part will probably be **aimed more at users living in countries that are infrequently subject to natural disasters.**

1.5 Impact :

First, it is expected to have at least an **educational impact** by allowing some people who have had no training in the reaction to a disaster phenomenon to acquire the basics of survival. This knowledge could be decisive if something happened during a trip for example. Otherwise, it's never a bad thing to accumulate knowledge even if you don't use it.

Secondly, it can have a **positive cultural impact**. It's a great way to **connect with people from all walks of life that you would never have met any other way**. It's a very good way to communicate with people from different cultures and to get to know them and interact with them. Often, virtual discussions are limited to people we already know or who look like us. Algorithms offer us people who are likely to please us. Here, people will be proposed only based on the location and the user can learn to discover, if he wishes, women and men.

3 Similar sites:

Others websites offering an interactive real-time map of earthquake around the world already exist, like :

- https://earthquake.usgs.gov/earthquakes/map/
- https://seismo.berkeley.edu/seismo.real.time.map.html

Some applications already allow you to chat with people from all over the world, but often completely anonymously without necessarily sharing position. Others focus precisely on connecting people living in the same geographical area only, which gives them a common ground and facilitates discussion.

However, none of them use events to allow people to chat with each other. In fact, I don't know of any site allowing direct discussion between users around the world based on trivial events. This project is trying to mix the two by bringing together people from all over the world and using earthquakes to facilitate discussion.

4 Data source :

3.1 Provider :

The provider is a **very reliable source**. It is the USGS Earthquake Hazards Program of the U.S. Geological Survey (USGS) . It is the US government agency responsible for monitoring seismic activity on its territory and around the world using the Global Seismographic Network (GSN). It's a 152 station, globally distributed, state-of-the-art digital seismic network that provides free, realtime, open access data through the IRIS DMC.

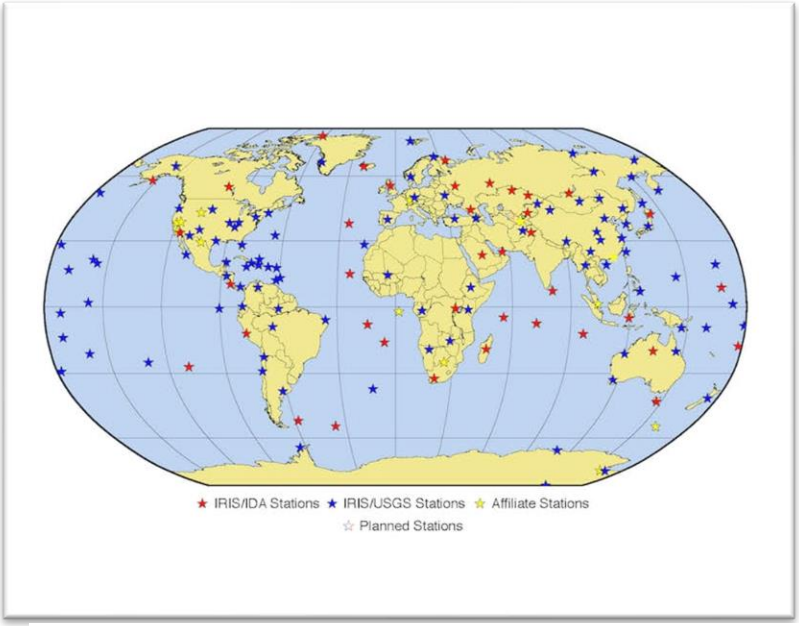


Figure 2 Distribution of the current station network

Figure 1 Logo of the United States Geological Survey (USGS)

3.2 Feeds

The USGS also provide several feeds updated every minutes which could be accessed for free.

Documentation can be accessed at the following URL :

- https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php

There is several feeds which give information according to the time interval compared to the current date and the magnitude of the earthquakes.

	Past Hour	Past Day	Past 7 Days	Past 30 Days
All	*x*	*x*	*x*	*x*
Significant	*x*	*x*	*x*	*x*
Magnitude > 4.5	*x*	*x*	*x*	*x*
Magnitude > 2.5	*x*	*x*	*x*	*x*
Magnitude > 1.0	*x*	*x*	*x*	*x*

Feeds could be accessed at the following URL :

- https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/[MAGNITUDE]_[INTERVAL].geojson

No methods or parameters could be applied.

3.3 API Documentation :

The USGS also provide an open API and allows custom searches for earthquake Information using a variety of parameters.

Documentation can be accessed at the followings URLs :

- https://earthquake.usgs.gov/fdsnws/event/1/
- https://earthquake.usgs.gov/earthquakes/feed/v1.0/geojson.php

Data could be accessed at the followings URLs :

- https://earthquake.usgs.gov/fdsnws/event/1/[METHOD[?PARAMETERS]]
- https://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/[MAGNITUDE]_[INTERVAL].geojson

We will mainly use the **QUERY method** which will allow us to add parameters that allow us to customize the results obtained by modifying the parameters.

We will mainly use the following **parameters** :

- **Format** : can be csv, geojson, kml, quakeml, text, xml. We'll use geojson for displaying on a list and 2D map and probably kml for 3D map.
- **Time** : with starttime and endtime mainly to filter by time.
- **Location/circle**: Using latitude, longitude, maxradiuskm for creating a circle area of a certain size around coordinates.
- **Limit** : for limiting the maximum of results.
- **Maxdepth** and **Mindepth** : to filter by the depth of the epicenter of the earthquake.
- **Maxmagnitude** and **Minmagnitude**: to filter by the magnitude of the earthquake.

There is a query example :

https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime=2022-01-01&endtime=2022-02-01

The API has the following limitation :

- Number of call : unlimited
- Result cap : limited to maximum 20.000 earthquake result.

3.4 Utilization

Feed will be preferred over the use of query with the API when possible (for default and preset filter for example). For more complex selection of earthquake, we'll user parameters of the API. Most useful ones have been descried on the last section.

We'll use this result to list and display recent earthquake and allow user to filter by time, location, depth, magnitude and maybe more. Data will also feed statistics and records which also could be filtered.

More details will be presented in the "Main features" section.

3.5 Problems or inconvenience

The large number of earthquakes each month (around 10,000) will make it difficult to record these events over the long term. For example, the data for all earthquakes in February 2022 represents more than 10,000 lines and 7mb of storage. We quickly realize that we are between 50 and 100 mb per year and therefore very approximately between 500mb and 1gb of storage for 10 years. If we want to archive and host earthquake information over the long term, **storage will have to be taken into account.**

5 Main features:

It should propose at least the same functionalities that the similar earthquake monitor website :

- Real Time world 2D map.
- Clickable earthquake point with detailed information about it.
- Link to 'Did you feel it' form or other links collecting data to help science
- Information about earthquake and safety instructions

We would improve it with :

- Chat with users located near the selected earthquake : first phrase should be : "Did you feel it ? "
- Real-time notification when an earthquake occur in 1 selected area (radius to be determined)
- 3D world map (use of cesium.js) https://sandcastle.cesium.com/?src=Geometry%20and%20Appearances.html

This is a more detailed list of functionalities : functionalities can be assessed according to the role of the users. We can distinguish 3 commons users roles :

- Visitor : User is not logged into an existing account.
- Logged: User has created an account and logged in
- Logged & Located: User logged in and provided their location and agreed to share it publicly.

Account gestion :

- **Visitor can register** : information are mail, username, password
- **Visitor can login** : information are mail, username, password (If login, visitor become logged user)
- **Logged can modify** his information. *(Password is advanced)*
- **Logged can input** his location.
- **Logged can choose to share or not** his location. (If location shared, logged user become located user)
- **Logged can unlogin**.
- **Logged can delete** his account.

Latest Earthquake functions :

- **All user can see X latest earthquake on a list and a map**.
  - Earthquake are linked from list to map. Focus one will focus the other.
  - Each earthquake will be represented as a point.
  - Each point is clickable which allow to display more information as : title, date & time, latitude, longitude, depth, magnitude, link to 'did you feel it' form, list of nearest users
- **All user can see X latest earthquake on a 2D map only**. List will be hidden.

    - Each earthquake will be represented as a point.
    - Each point is clickable which allow to display more information as : title, date & time, latitude, longitude, depth, magnitude, link to 'did you feel it' form, list of nearest users
- **Logged user can see latest earthquake on a 3D map**. *(advanced)*
    - Each earthquake will be represented as a point.

Filter function :

- **All user can filter by area** : Focus on an area to see more earthquake
    - Choose latitude, longitude and radius (Max radius because of 20.000 results limit).
- **All user can filter by magnitude** : Only show earthquakes within a certain magnitude range
    - Choose minimum and/or maximum.
- **Logged user can filter by time interval** : Focus on time interval to see more earthquake
    - Choose start time and end time (Max interval to be determined because of 20.000 results limit).
- **Logged user can filter by depth** : Only show earthquakes within a certain epicenter depth range
    - Choose minimum and/or maximum.

Chat function :

- **Logged user can start a chat** with located users who are near the earthquake selected.
    - A link to communicate will be available in the list of nearest users for each of them, when the details of the selected earthquake are displayed.
- **Located user will appear in the list of nearest users.**

Stats function :

- **All user can view statistics about earthquake**.
    - Country mas touched by earthquake. *(advanced)*
    - Closest located user of the latest earthquake
    - Average depth, magnitude
    - Daily, weekly, monthly, yearly, total earthquakes by area *(yearly, total & By area are advanced)*
    - Etc.
- **Located user can view more statistics about earthquake according to his location** (Reset when he changes it)
    - Nearest earthquake record
    - Number of earthquakes that occurred within a customizable radius and time interval.
    - Etc.

Third-Party / External function :

- **Visitor can register and login with Google or Twitter**.
- **Maps will automatically by centered** according to the position indicated by the user's browser.
- **Located user can ask for notification** or email when earthquakes occurs near his location.
- **All user can access to embed Twitter timeline** of official USGS accounts. *(maybe)*
- **Logged user could download long time archive** in a compressed archive format. *(maybe)*

Summary :

All user

- Register & login (with Twitter & Google too)
- See the earthquakes of the last 7 days.
- Show earthquake on a list or a 2D map.
- Earthquake statistics, detailed information, "*Did you feel it"* form
- Filter by :
  - Area
  - Magnitude
- View embed timeline of recent earthquake
- View information about earthquake (how they are born, how to protect against them,etc.)

Logged user :

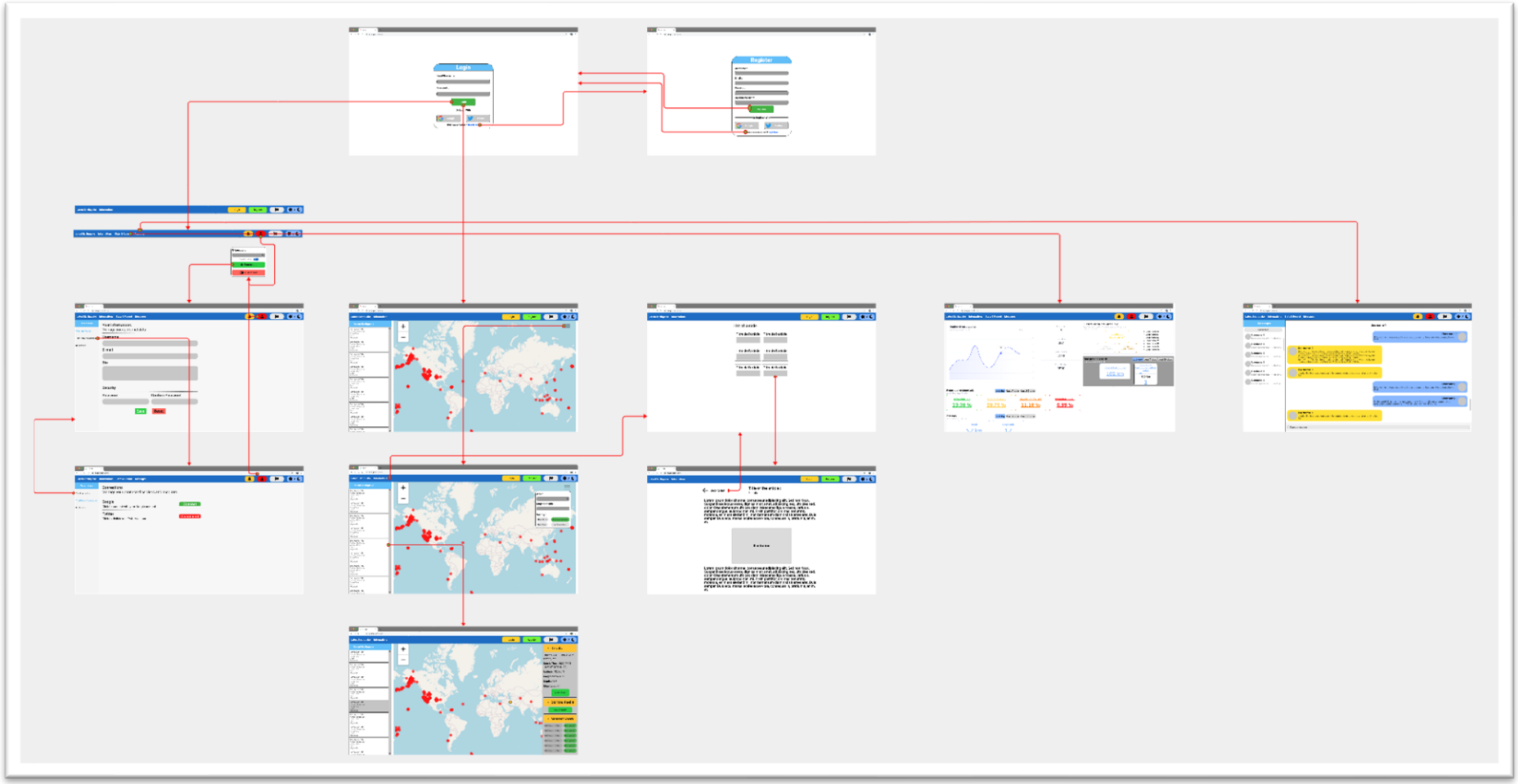
- Modify personal information
- Share location
- Unlogin
- Start chat with located users near recent earthquake selected.
- Show earthquake on a 3D map. (maybe)
- Add filter by :
  - Time
  - Depth
- Download long time archive
- View basic statistics

Located user :

- Be notified when an earthquake occurs nearby. (maybe)
- Be proposed to other users when nearby earthquake occurs
- Add statistics based on location :
  - Nearest earthquake record
  - Number of earthquakes that occurred within a radius of X km
  - Etc.

Figure 3 App requirements

6 Prototype, interface, views[1]



[1] Each component is available and visible independently in the following folder : https://github.com/Tiss971/earthquake-monitor/tree/main/entregables/prototipo
And at the following link: https://www.figma.com/proto/kHS2VhlpmdONVB2pXVKYan /STW?node-id=13249%3A5322&scaling=scale-down&page-id=13249%3A5066

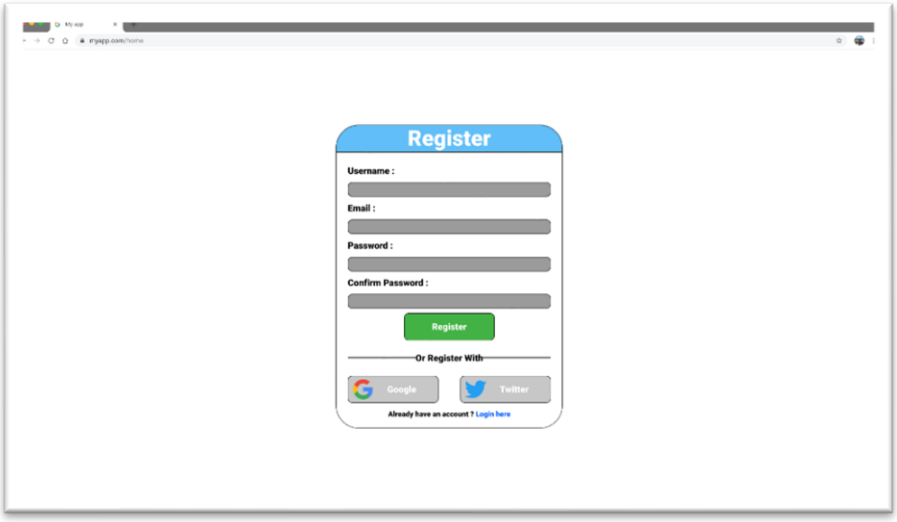


Figure 4 Register View

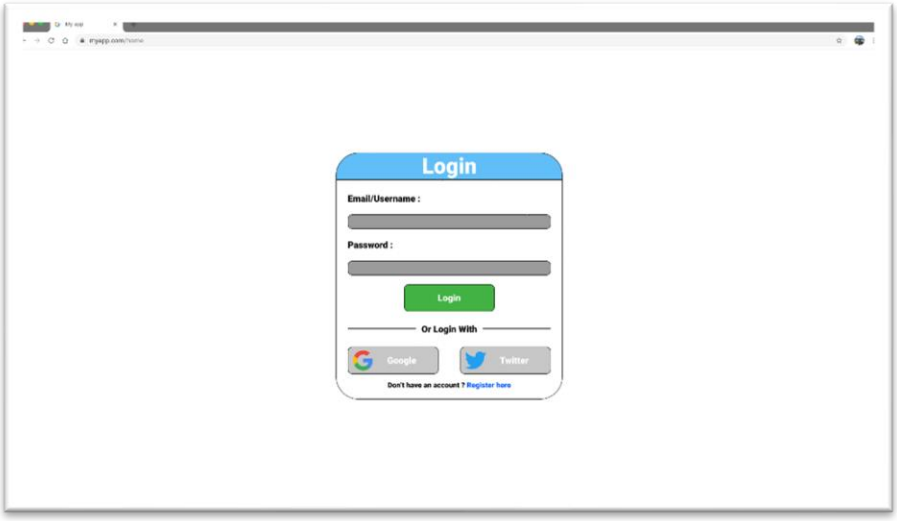


Figure 5 Login View

- User registration and login will be integrated with the use of Google or Twitter account.

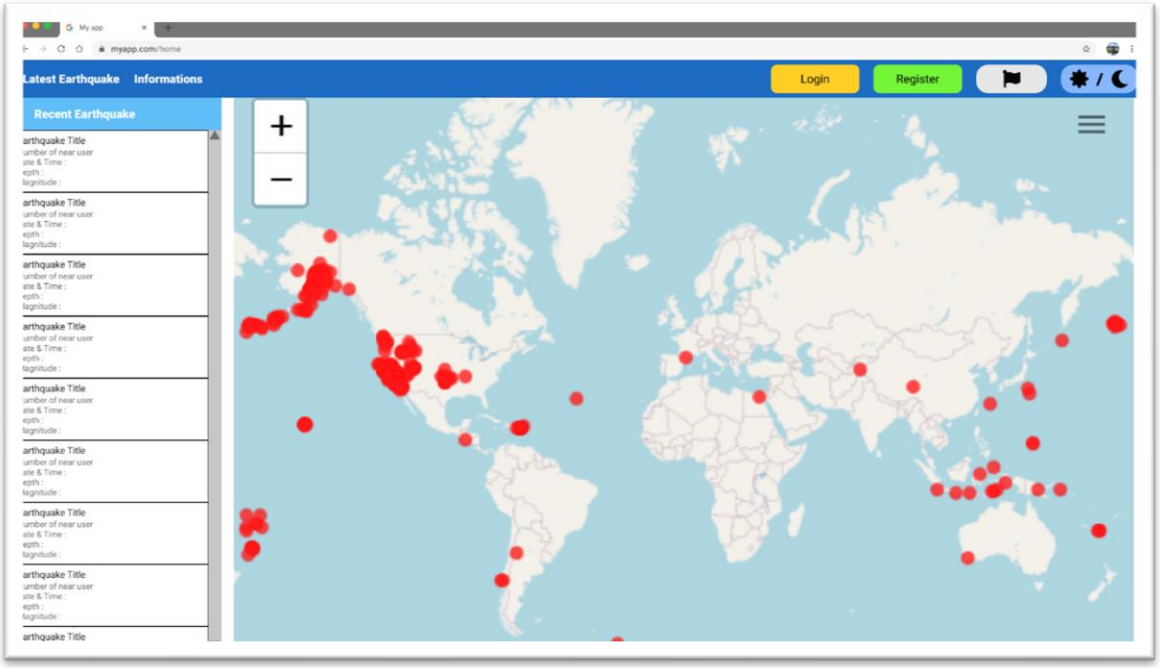


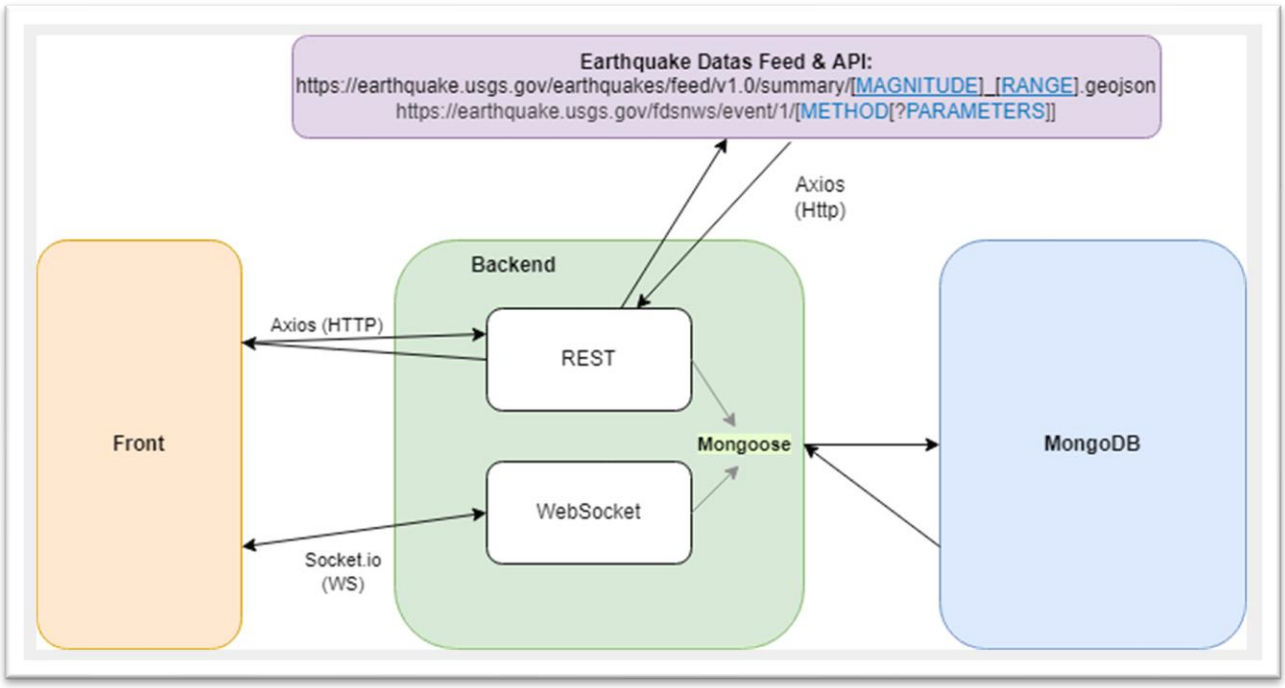
Figure 6 2D Map

- 2D map will be integrated with Leaflet and OpenStreetMap. The browser will obtain the user's position (more or less precisely if the user agrees to share his position) to automatically center the map.

Others :

- A twitter timeline of the recent earthquake may be created and embedded and display in the bottom right corner under the list of 'Recent Earthquake

7 High Level Architecture



When user log a webSocket connection is initialized and user is considerate as an onlineUsers and set in database. On logout he is erased from the onlineUsers collection.

- WebSocket are used for the chat part.
- Mongoose takes care of managing access to MongoDB
- All earthquake functions get the datas from the provider source usgov. No earthquake data is currently stored in mongoDB.

8 Data Models

8.1 Users :

- **_id** *(UNIQUE NOT NULL)*
- **username** (NOT NULL)
- **email** *(NOT NULL)*
- **password** *(NOT NULL)*
- **third_party_auth** *(CAN BE NULL)*
    - **provider_name**
    - **provider_id**
    - **provider_data**
- **lastVisit**
- **address**
- **public**
- **location** (CAN BE NULL)
    - **latitude**

- **longitude**

Users can log with username/password authentication, Google Oauth or Facebook Oauth.

User's username don't have to be unique because during registration an user can unfortunately select the name provided by third authorization, if he have the same name for example.

Location can be null because user aren't forced to share and set their location.

We store address in addition to coordinates in order to facilitate display and calculation.

8.2 OnlineUsers

- _id
- socketId
- userId
- name

All attributes are required. "userId" and "name" referred to user's informations.

SocketId referred to an existing socket.

8.3 Messages

- _id
- From
- To
- Message
- Timestamp

All attributes are required. "From" and "To" referred to user's _id.

8.4 Sessions

Passport use mongoDB to store persistent session.

9 API REST

API REST is detailed here: https://earthquake-monitor-server.herokuapp.com/docs/

10 Navigation & Implementation & Analytics

The app can be divided into 5 principal parts :

10.1 Log In & Register

The connection can be made in several different ways.

- **Register with username/email & password then log in**
- **Log in with Google**
- **Log in with Facebook**

We use Passport Js to manage the connection. Passport Js is initialized in server "app.js" and stores users in sessions using MongoDB.

Passport Local strategy handle login connection but register isn't currently handled by Passport.

A Google and Facebook developer account has been created and secret tokens and id has been set in .env files. Passport Js dedicated strategies use these information to login with OAuth and to provide user's information from these third app account.

All strategies can be found in the "Passport" folder.

10.2 User management

- By default, user location isn't set. User has to explicitly choose to set his location. In order to do it, we provide an address geocoding service. User can type an address and the geocoding service will propose places with name and coordinates. This means :
  - Users can't set latitude and longitude by his own to avoid range error.
  - Users can't write an inexistent address.
  - Users can locate him everywhere, which mean he can lie but we trust him 😊

The geocoding service is provided by ArcGIS which propose a free tier for a limited number of queries. The service is divided in 2 steps:

1. User's start typing. We send the input of the user to ArcGIS and it replies a list of places that could correspond. In order to limit the amount of query we use throttled query for waiting that the user has stopped writing.
2. User's select the address he want or change his input (back to step 1). The list provided contain a magic string which we can send to ArcGIS again to directly get the coordinate of the chosen place. We set these coordinate in the user data in database.

These service has been chosen because it's free, provide suggestions and can be easily integrated with Leaflet.

- User can choose to share his position or not. An attribute exist in user database to quickly detect user's agreement.
- Users can change his username, email if it isn't already taken and the user can change his password.

10.3 View earthquake

It is the main functionality. Accessible to all users (logged or not), It displays on a map the fetched earthquake according to a certain range of time or magnitude.

We used Leaflet to display the map and React-leaflet is the React wrapper that facilitate the implementation. The map is instanced and we center in user location according to the navigator information. If user deny the location request of the navigator, we currently use a default location but we should search for user's location in database before.

Automatically, earthquake data are fetched from the provider source. Each earthquake are displayed on the Leaflet map thanks to latitude and longitude.

At the same time, we displayed a list of these earthquake. Clicking on a list item will move the map to the corresponding earthquake and highlight it.

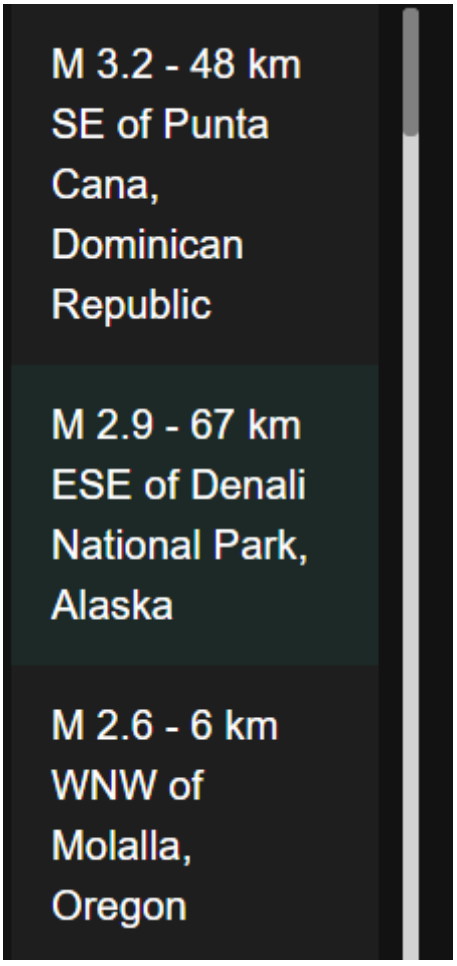


Figure 5 Earthquake List

Figure 6 Highlighted earthquake marker

Clicking on an earthquake on the map will display more information about the selected earthquake.

Like title, location, date & time, coordinates, depth and magnitude. But also, direct links to the provider source which takes care of collecting the feelings and displays more information.

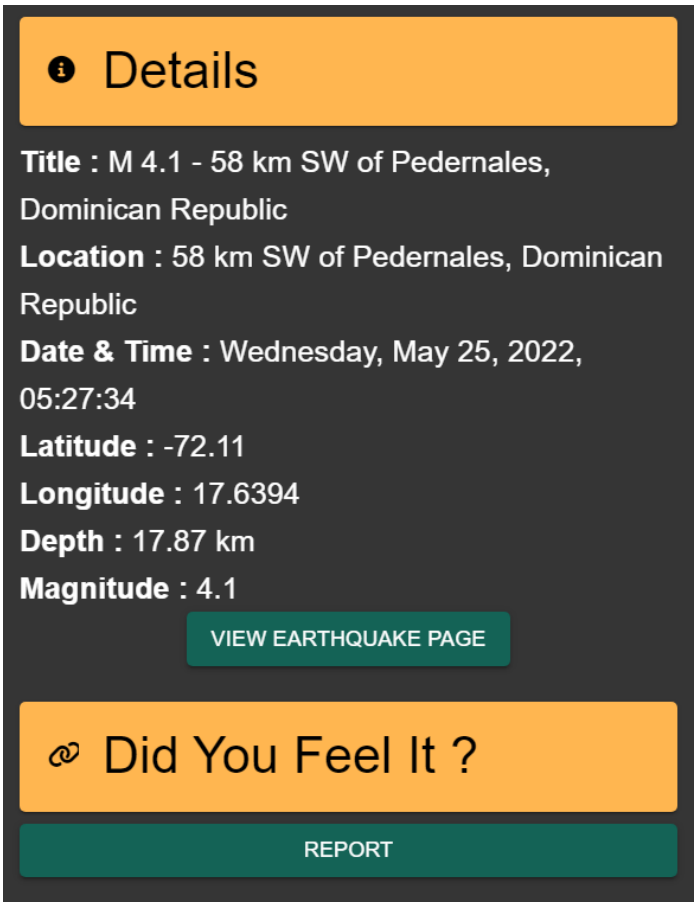


Figure 6 Earthquake details

All user can filter fetched earthquake by range time and magnitude. Logged users can add other filtering, in frontend, by depth over the fetched data by range and magnitude.

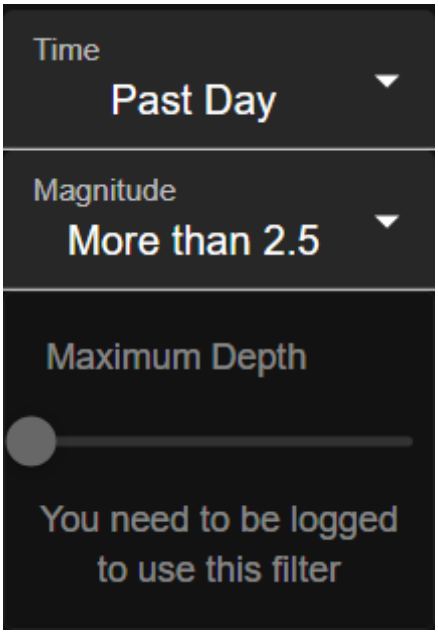


Figure 7 Filters

One more thing, logged users can see the 5 nearest users of the selected earthquake. In order to do that, we send to backend the coordinates and we use geonear query provided by mongoDB to order the users by the nearest position, then we send the 5 first one. The problem is that the $near operator in queries don't provide the calculated distance. Another way to get it will be to used $near aggregate function but it isn't accessible in free tier of MongoAtlas.

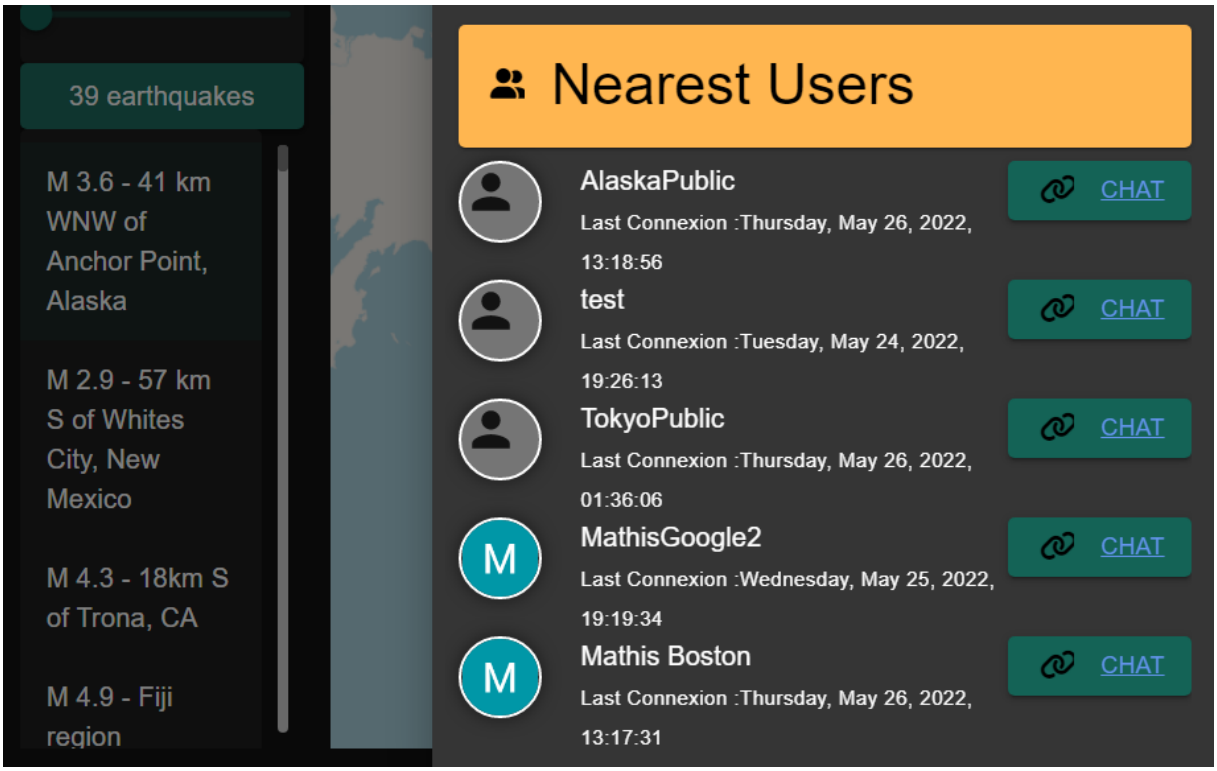


Figure 8 Nearest users of an earthquake in Alaska

Each of the nearest users have a direct link to a chat to them.

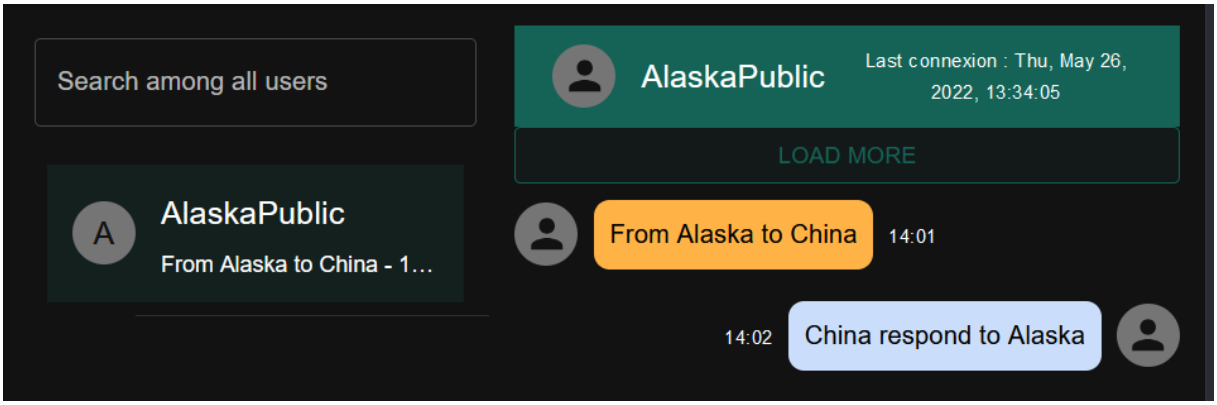
10.4 Chat

In order to create a private persistent social messaging app, we used WebSocket to send and receive message event and we store all messages in a mongoDB collection.

If a user is an online user, it means his userID and socketID are known and we directly emit a WebSocket event to the socketID all messages that refers to his userID. The message contains :

- From : the sender user ID
- To : the receiver user ID
- Message : the message content
- Timestamp : the exact date and time of the message

Then the front will interpret all messages and displayed it depending on whether the "from" id matches the current user id or not.



The "To User Id" is displayed in the URL : *localhost:3000/chat/628e567b80c243b577f0643c*

It allows us to directly accessed to the desired user and each time this URL parameters change we fetch the history of messages between this id and the current user ID in order to display the old conversation. By default, a static amount of message are fetched (20) but we can go back further in history by loading more messages because the mongoose query is set with a skip and limit parameters that allows us setting a paginations system. Each time we clicked on the "LOAD MORE" button we fetch another page of old messages between current and selected user.

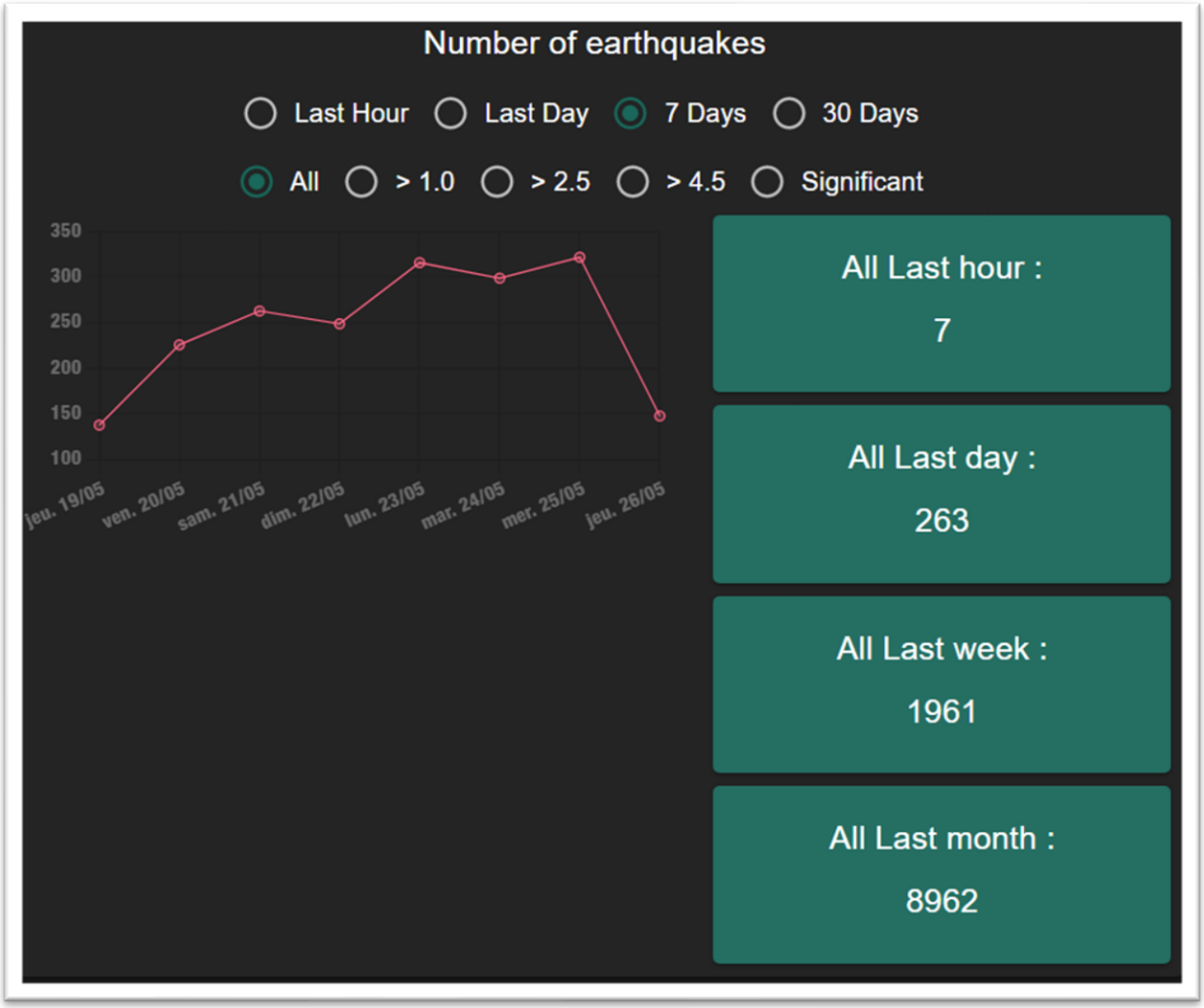
We provide an autocomplete text field where we can search among all users in the database. Also, under this autocomplete, there is a list of all the existing conversations the current user is linked to. To do that, we fetched messages where the current user ID is the same as the "FROM" or the "TO" attributes and we grouped by ID keeping the last timestamp and ordered result by timestamp.

10.5 Stats

This last part is about the statistics and graphics about earthquakes. To display charts we used charts and react-chartjs-2 as a wrapper.

We choose to display :

- A line graphic of the evolution of number of earthquakes

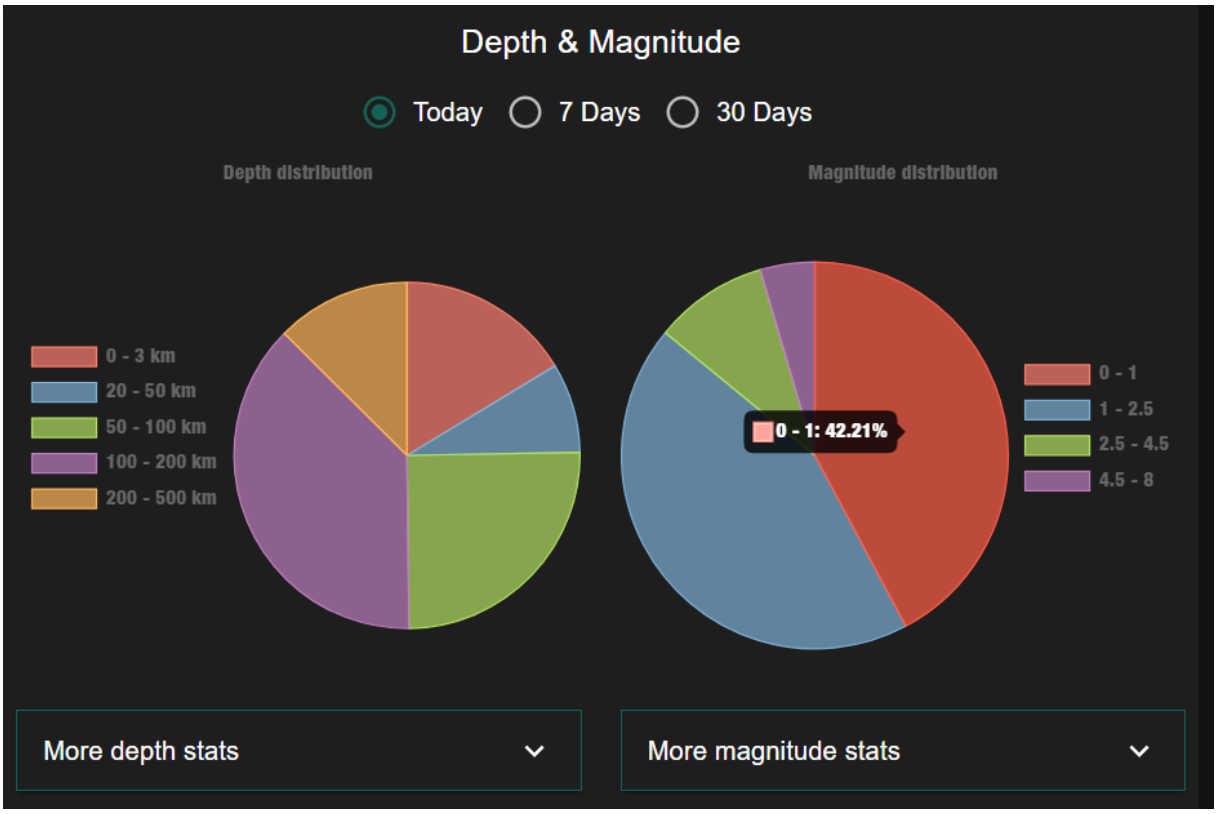


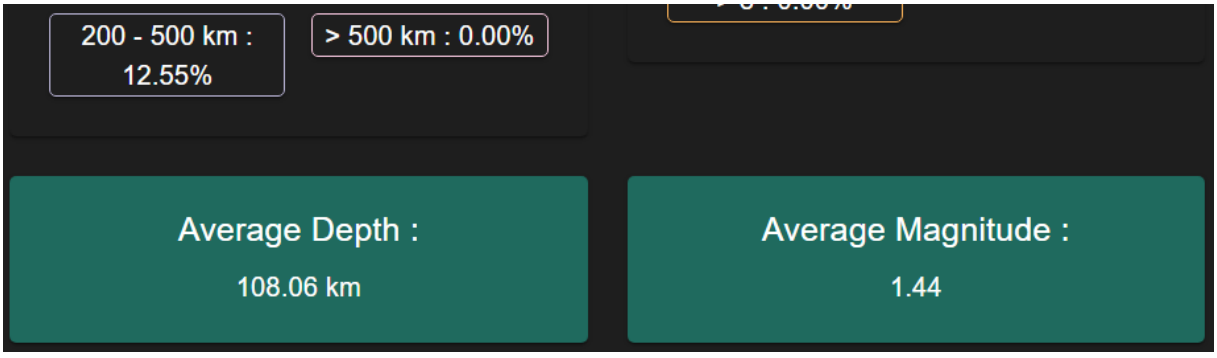
We can change the desired range and magnitude and it will refresh data :

- For the hour range => earthquakes are grouped by minute of occurrence
- For day range => earthquakes are grouped by hour
- For week and month range => earthquakes are grouped by days.

The total number is also displayed

- Repartition and mean of depth and magnitude of the earthquakes





We can change the range and it will refresh data. Clicking on More stats buttons will display percentage for all labels.

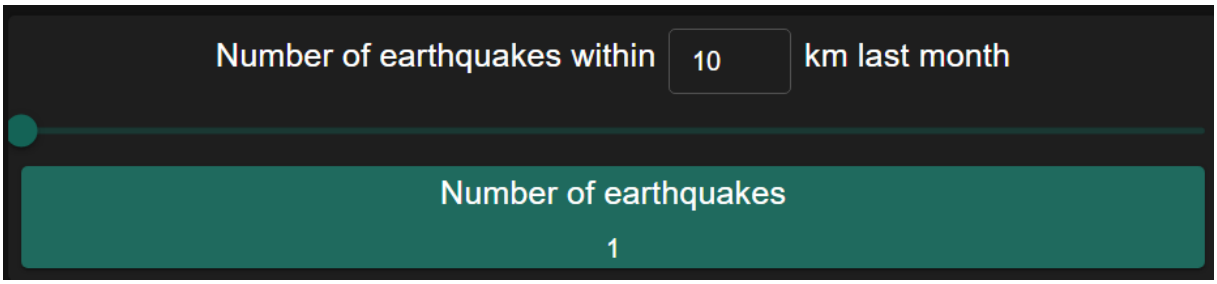
Depth is split in these parts :

- 0 - 3 km
- 3 - 10 km
- 10 - 20 km
- 20 - 50 km
- 50 - 100 km
- 100 - 200 km
- 200 - 500 km
- > 500 km

Magnitude is split in these parts :

- 0 - 1
- 1 - 2.5
- 2.5 - 4.5
- 4.5 - 8
- > 8

• Number of earthquake in a certain range of the user location last month



Only logged users can accessed to this statistics and the user also must have set up his location.

We use the API to request all earthquake in a circle center on latitude and longitude of user with a selected max radius.

https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&latitude=${latitude}&longitude=${longitude}&maxradiuskm=${maxRadius}

It gives us all the concerned earthquakes and we can count or display them.

11 Deployment

11.1 My Automatic Deployment

To deploy the app, we use Netlify setup which will build and deploy automatically the react front-end code each time a commit his made in the main branch of the git repository.

The same things is set up for deploying the backend code in Heroku. In each host, we set up the environment variables used in the code.

The mongoDB database is host with MongoAtlas and already set up with some fake account located across the globe.

11.2 Clone and Setup

In order to copy the site, it's needed to :

- Download the code from git repository
- Create a .env in the root of client folder and set up variables for front-end
    - REACT_APP_API_ENDPOINT=http://mysuperapi.com
    - REACT_APP_ARCGIS_API_KEY=my_token_string_provided_by_argis_developper_account[2]
    - REACT_APP_RECAPTCHA_SITE_KEY= my_token_SITE_KEY_provided_by_google_developper_account_for_recaptcha_v2[3]
- Create a .env in the root of server folder and set up variables for back-end
    - DB_URI=mongodb+srv://[user]:[password]@cluster0.hngu0.mongodb.net/[database]?retryWrites=true&w=majority[4]
    - GOOGLE_CLIENT_ID=my_token_CLIENTID_provided_by_google_for_OAuth_2.0[5]
    - GOOGLE_CLIENT_SECRET= my_token_SECRET_provided_by_google_for_OAuth_2.0
    - FACEBOOK_CLIENT_SECRET= my_token_SECRET_provide_by_facebook_for_OAuth[6]
    - CLIENT_HOME=https://mysuperfrontendUrl.com
    - RECAPTCHA_SECRET_KEY= my_token_SECRET_KEY_provided_by_google_developper_account_for_recaptcha_v2[3]
- Run ***npm run dev*** in client folder and ***npm run dev*** in server folder
-

---

[2] https://developers.arcgis.com/documentation/mapping-apis-and-services/security/api-keys/

[3] https://help.salesforce.com/s/articleView?id=sf.workdotcom_quemgmt_set_up_captcha_googlekeys.htm&type=5

[4] https://www.mongodb.com/docs/atlas/getting-started/

[5] https://support.google.com/cloud/answer/6158849?hl=en

[6] https://developers.facebook.com/docs/development/create-an-app

12 Development problems

12.1 Twitter OAuth

A problem occurred with authentication with Twitter and Passport, which is why we opted for Facebook. Twitter's developer account was set and API Keys set in each .env files. Passport Twitter strategy was set up as indicated in many tutorials but the same error occurred and the same page of Twitter was displayed indicating that the keys do not match. The problems seems to be related to the difference in OAuth version between Twitter APIs, Passport for example. Quickly, it was decided to try using Facebook instead and its implementation went smoothly.

12.2 Can't get distance with mongoDB

A lot of features was based on calculations over geodatas of users and earthquakes. Nevertheless, as indicated in the part of this document over nearest users, I were not able to get the distance calculated by the $near argument in query and using the $near argument with a "distance field" attribute is not allowed in a free tier of MongoDb Atlas.

But the solution is finally to manually calculate the distance for the 5 nearest users using the haversine formula.

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$
$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$
$$d = R \cdot c$$

Doing this manually for all users, for all earthquakes for statistics must cost too much.

13 Potential problems

When selecting the range of a month and all magnitudes, there are often tens of thousands of earthquakes recorded. Leaflet and navigator cannot handle such a number of animations and information to process and the page slows down to freezing and being unusable.

➔ The imagined solution is to only display the visible markers according to what the map displays. It is theoretically possible to access the instance of the map and obtain the coordinates of the "edges" of the map displayed. Using these edges, we can filter the markers that are in this area and display only these.

But, the React wrapper around Leaflet makes this solution a bit complicated to implement.

14 Project Time Distribution

Since the project was carried out alone, the organization of the work was lighter. No Gantt chart has been set up. The working method was based on the analysis of the needs of the application described in *Figure 4* and on the implementation point by point. A Notion document follow each requirement by category, user scope and state of implementation. It is visible by clicking on the following link : http://tiny.cc/p00suz

All the needs could not be implemented because the dedicated time could not be sufficient to achieve everything. From May, tasks that seemed compromised were discarded and the focus was on achievable tasks, this results to a lot more of commits and small additions during this month.

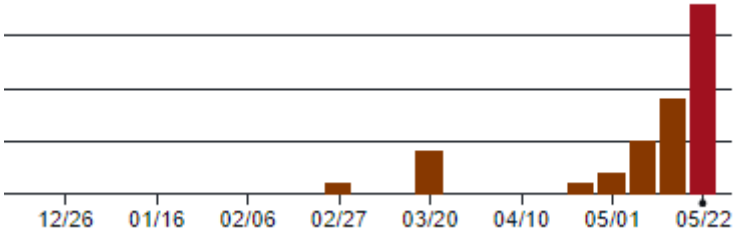
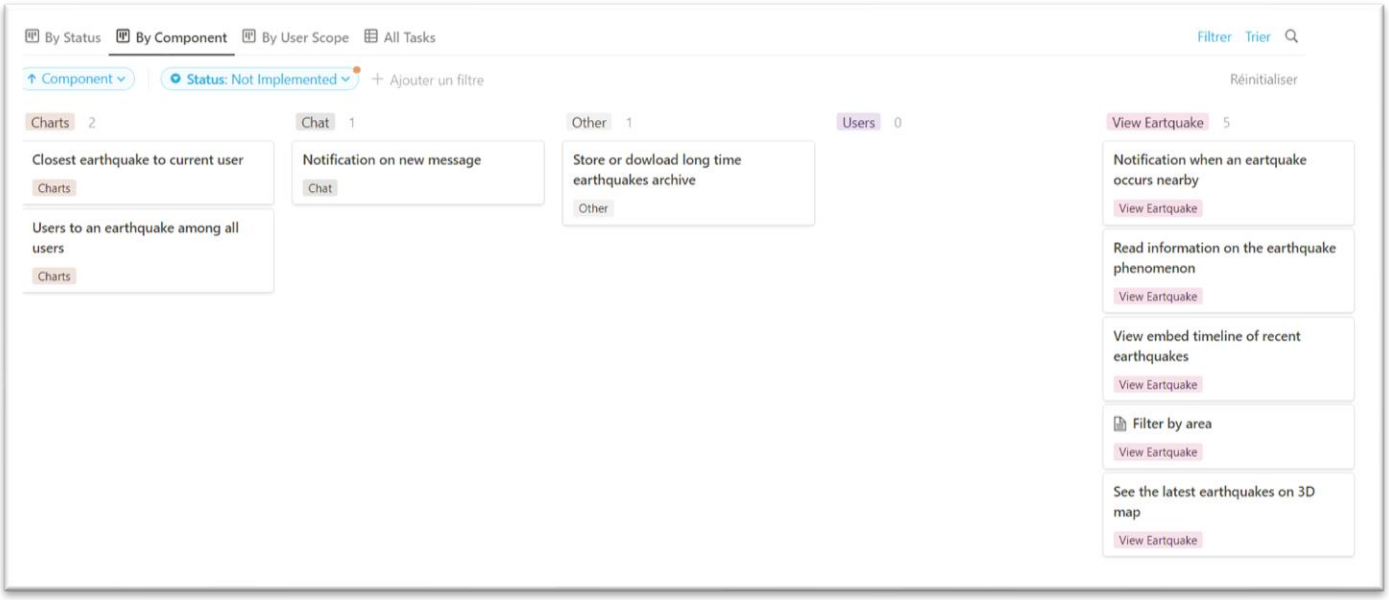


Figure 9 Number of commits

Figure 10 Addition and Deletion in code by week

15 Missing requirements



- They are due to the problems listed in the "development problems" section and are due to an amount of work that cannot be managed by a single person.

- The use of a 3D map is hardly possible as long as we are not able to manage tens of thousands of markers on a 2D map.

- Storing long time data archive could be implemented easily but the storage over years could be an issue with free tier web host.

- Charts and functionalities related to distance were compromised by the lack of automatic distance calculations by mongo DB. Once this problem is solved, their implementation will be easy.

16 Annexes

- **Component by component view**

https://www.figma.com/proto/kHS2VhlpmdONVB2pXVKYan/STW?node-id=13249%3A5105&scaling=scale-down&page-id=13249%3A5066

- **Navigation overview**

https://www.figma.com/proto/kHS2VhlpmdONVB2pXVKYan/STW?node-id=13242%3A1436&scaling=scale-down&page-id=1%3A14&starting-point-node-id=13242%3A1436