FILE EXPLORER

Isma and Bernat

This project simulates an explorer of files where directories and files can be uploaded, created, moved and even deleted.

ORGANIZATION

In order to organize the project, we first wrote the key points that would present a difficulty throughout the project. With that, we created code snippets, used as microenvironments where we could test and experiment with one specific problem only. Once we had a solid idea on how to solve them, we started developing

In order to track the tasks, we wrote down the requirements in a sheet , and divided them by days. The tasks were assigned by writing down the name of the person responsible for it, at the end of the task name. Once the task was finished, it was marked in green color.

The requirements this project fulfills are listed below.

REQUIREMENTS

- How will the **interface** be

    - You will have to **design a wireframe** of your application taking into account the requirements.
- What **actions** can be executed by the user
    - You will have to **design a use case diagram**
- **Analyze** and **understand** what brings more value to the user
- **Analyze** how you will organize the project at the level of **directories** and **files**

IMPLEMENTATION

- **Create**, **modify** and **delete directories**
- **Browse through directories** from an **initial path**

    The **initial path** will start from a **folder** inside the **project repository** whose name will be "**root**".

- **Search directories** and **files** by **name**

    In the case of **searching** for **files by name**, you must also be able to specify **their extension as part of the name**.

- **Navigate through the initial path** established and all the folders created from that path. Therefore the **user will not be able** to see or **navigate** to the **parent folders of the “root” folder**.
- **Upload a file** to a directory
- **See** the following **information** of **files** and **directories**

    Creation date, Last Modified Date, Extension (if it’s a file), Size (MB/KB)

- **Show** the **icon** of the main **file extensions** such as: Solo estos permitidos.

    Doc,csv, jpg, png, txt, ppt, odt, pdf, zip, rar, exe, svg, mp3, Mp4

EXTRAS

In addition to the **requirements** mentioned above, you can add the following **extra functionalities** so that the **user is able to**:

- **Show the information** of the uploaded **“.csv”** files on the screen.
- **Move files** and **directories between folders**
- When **deleting** a **file** or **folder**, it will be moved to a **specific folder** called "**trash**", so if you want to **delete it completely** you must delete it from the mentioned folder.

DESIGN

Navbar: project logo, search bar, options for uploading or creating folder / files.

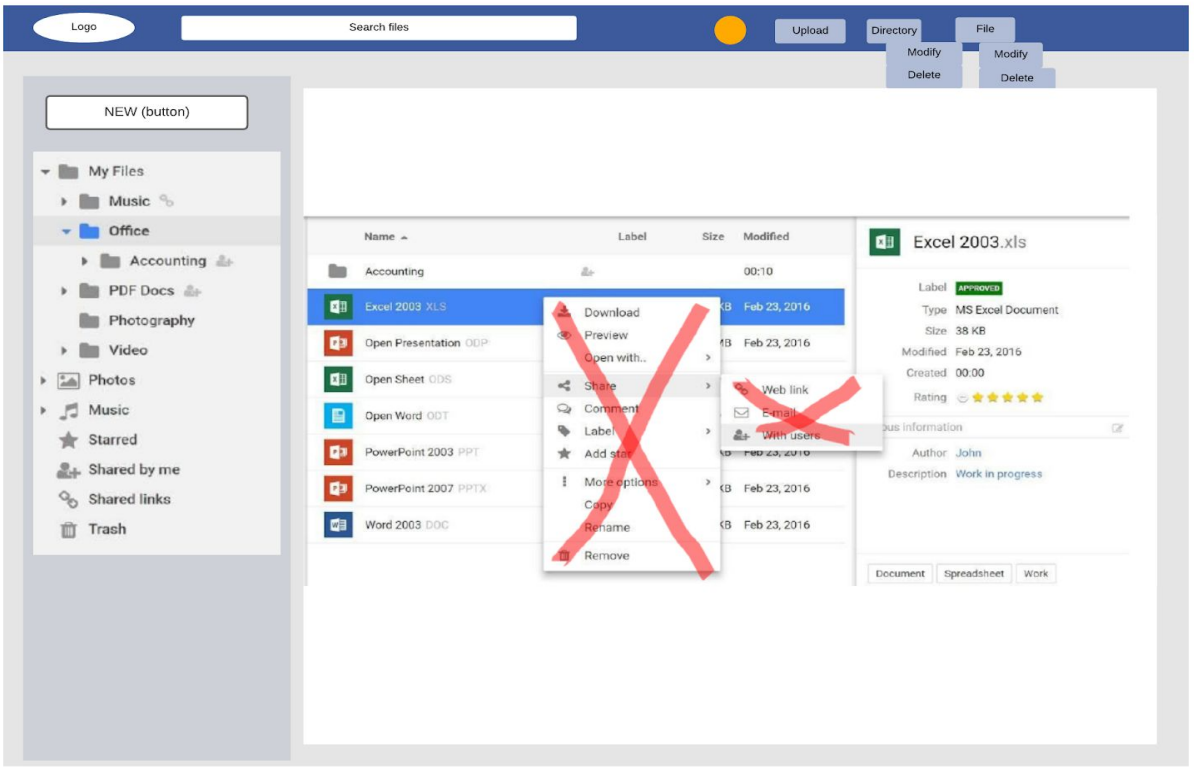
The website body will be structured in three main columns:

- Left: folder structure

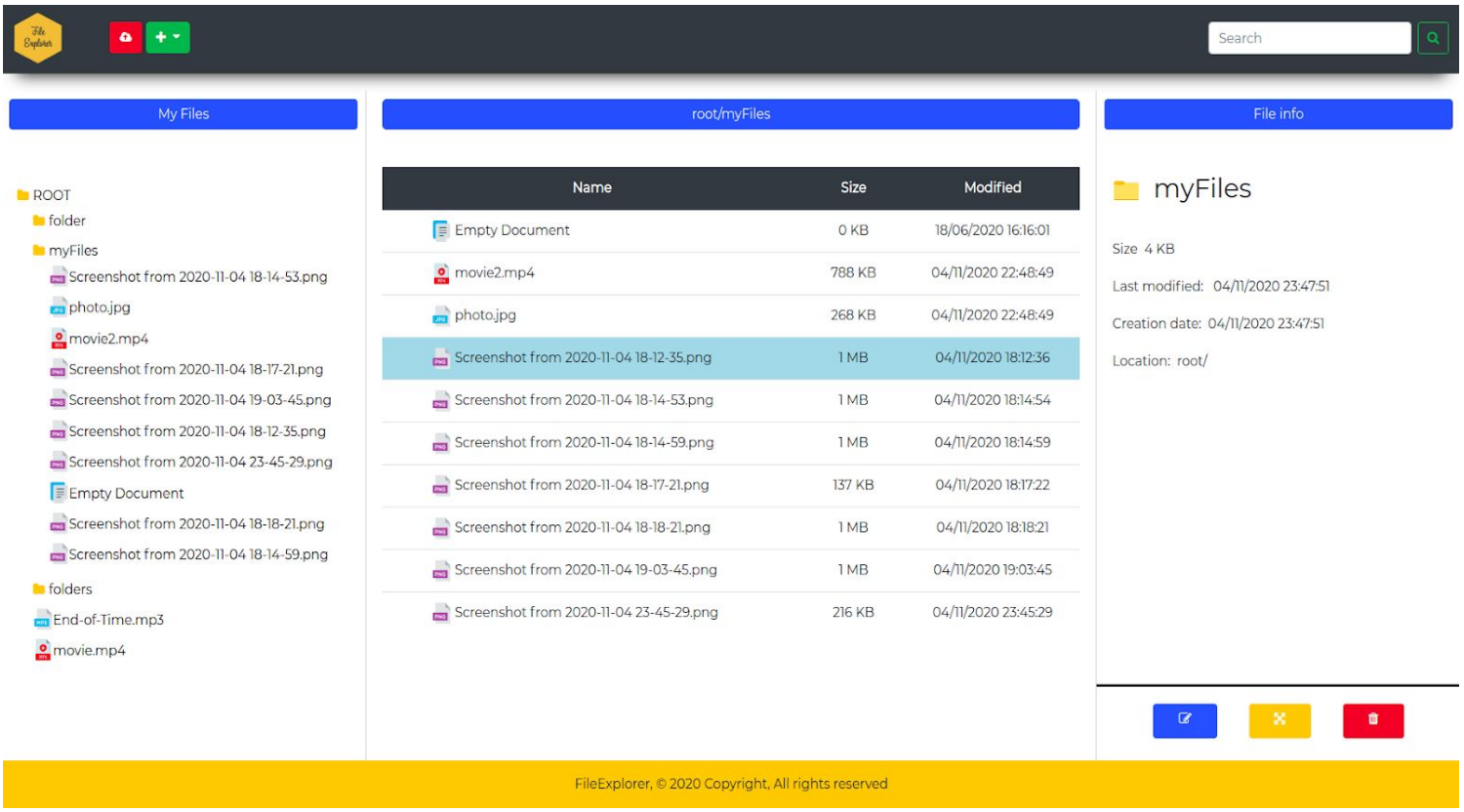
- Middle : table listing the content of the folder being selected. In case an image, video or audio file is been selected, it will be displayed in the middle, hiding the contents table

- Right : Properties of the folder or file being selected.

The following wireframe gives an idea of how the initial design was.



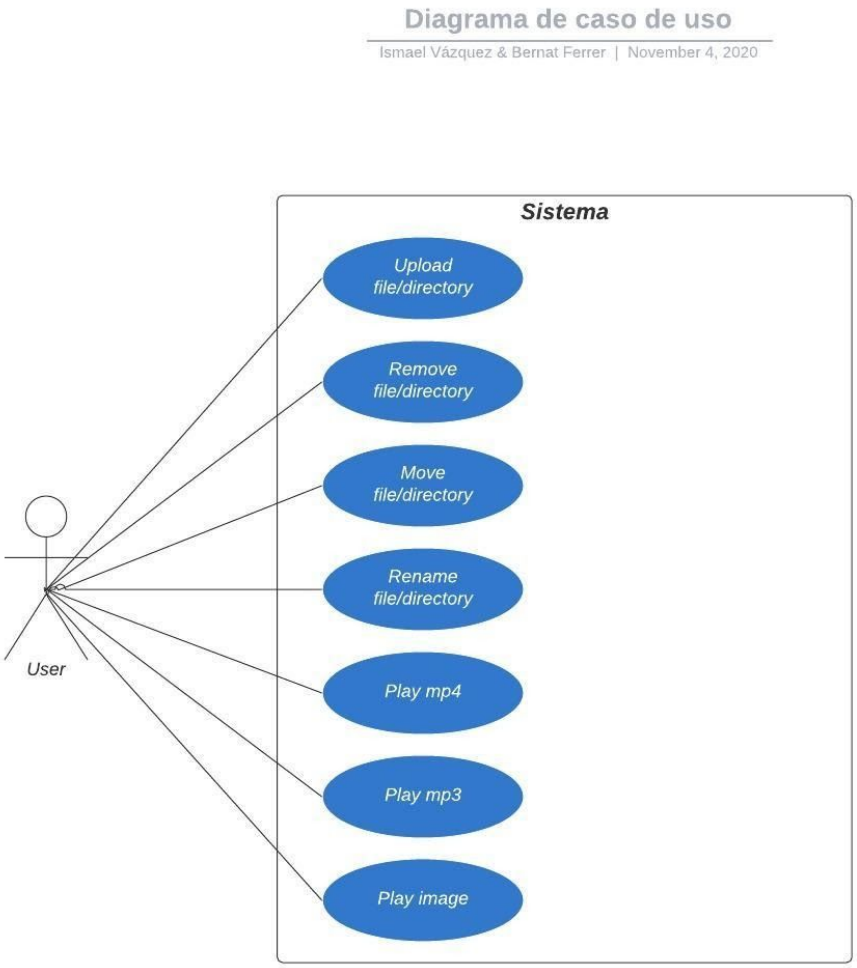
And the end result:



Hence, with the initial wireframe we set the basic structure, namely the navbar and the body structure with three columns . On the other hand, we left room for improvisation with styles and hence the result was clearly different from the starting point.

USE CASE DIAGRAM

The user can perform all the actions covered in the use case diagram



LESSONS

- Be careful with bootstrap (it uses “important!” sometimes)
- Create code snippets of what we do not know how to do
- Plan which are the most difficult points of the project
- Separate code with different files.
- Do not overplan, especially if you do not know how to do some of the things.
- Built in functions in PHP

PROBLEMS

- Bootstrap has unpredictable behaviour, many hours spent refactoring
- Bugs regarding specific edge cases (move/upload/edit) not taken into account at first. Examples
    - Not allowing for a folder to be moved inside itself
    - Disabling edit/move/delete for the root folder
    - Validating names (no “/”)
- Linux permissions.
- Refactoring code. Whenever we refactored code, some problems would appear involving functionalities already implemented.