FileSystem Explorer

Alberto González Sánchez & Jordi Arnau Esteban

Project requirements

The requirements for this project are:

- Create a basic interface for a system file explorer.
- Users must be able to **create a folder**.
- Users must be able to **modify** (rename) a file or folder.
- Users must be able to **delete** a file or folder.
- Users must be able to navigate through directories starting from a folder named ***root***.
- Users **can't access** the root parent folders.
- Users must be able to **search** files and folders by name and extension.
- Users must be able to **upload** a file to a directory.
- Users must be able to see some **details** about the files and folders: creation date, extension (only files) and size.
- The interface must **show an icon** for the basic file extensions (doc, csv, jpg, png, txt, ppt, odt, pdf, zip, rar, exe, svg, mp3, mp4).
- Users must be able to **view** images.
- Users must be able to **play** videos.
- Users must be able to **play** audios.

Technical requirements

- Mandatory use of **PHP v8 or higher**.
- Mandatory use of **GIT**.
- Mandatory use of **camelCase** and **English** for the code and documentation.
- **Inline styles** are totally forbidden.
- **Global PHP variables** are totally forbidden.
- **PHP third party libraries** are totally forbidden.
- You might use **third party libraries** for HTML, CSS or JavaScript.

Wireframes

After analyzing the project we created wireframes that included the different uses and cases that the users might find when using our Filesystem Explorer.

Basic wireframe

Wireframe of the main features and interface of the project. It includes:

- Search bar
- Breadcrumb panel to navigate through previous directories
- Create folder button
- Upload file button
- Directories section with the existing directories
- Files sections with the existing files.

Details modal wireframe

Wireframe showing the modals where the files or directories details are shown. It includes:

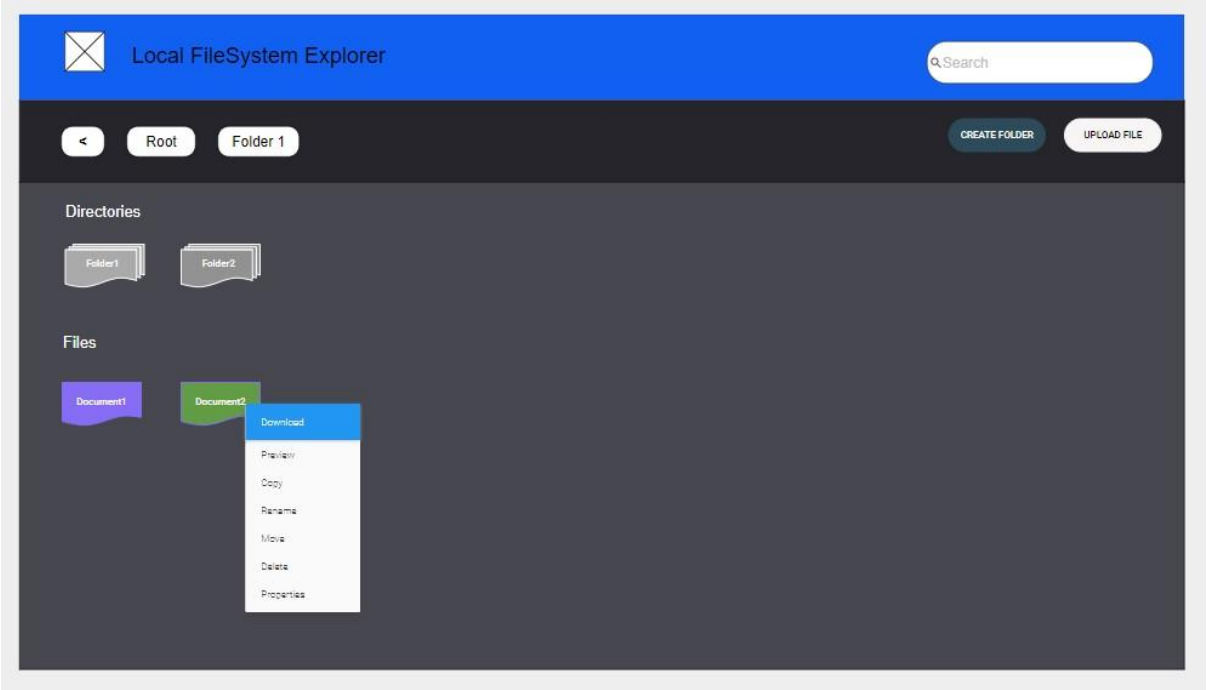
- Creations date
- Last modification date
- Extension (only with files)
- Size

Media player modal wireframe

Mockup modal wireframe showing how the videos and audio should play on the site.

Includes the name of the file and the media player.

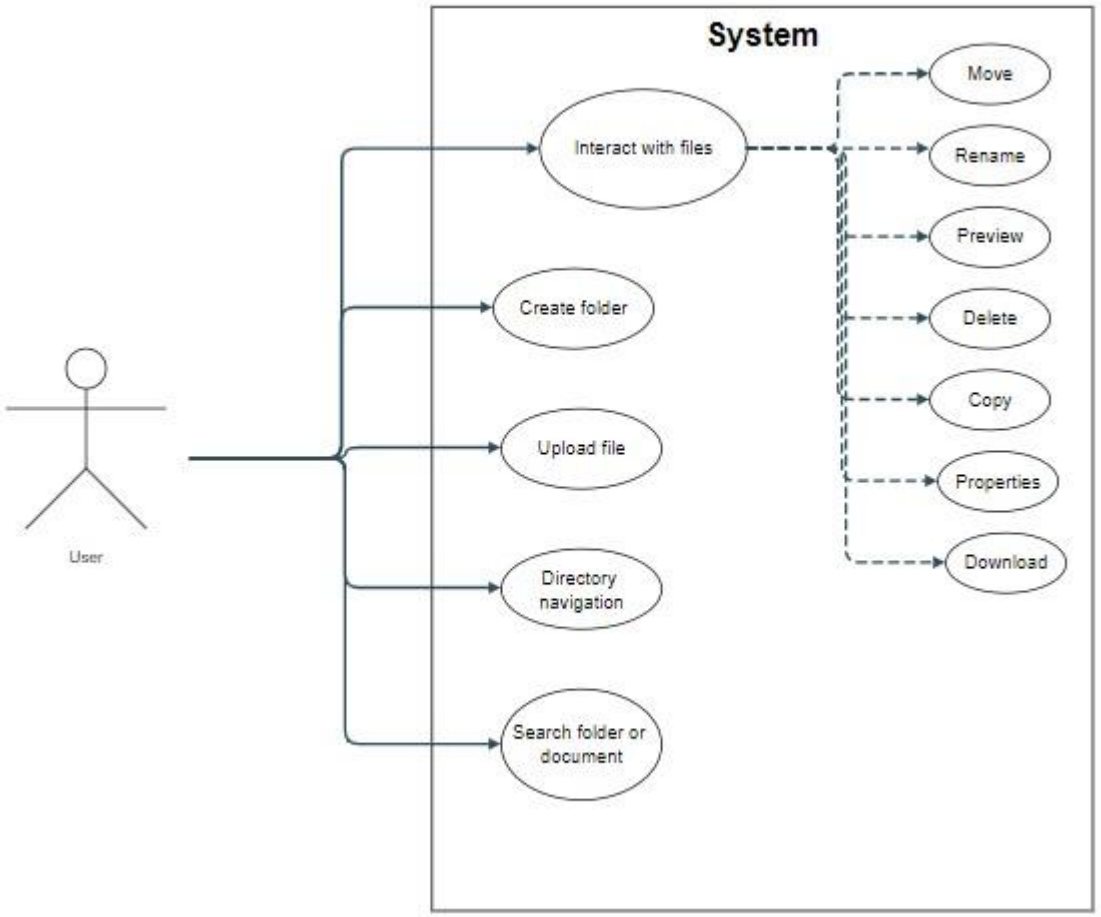
Contextual menu wireframe



Example on how the custom contextual menu and its options should show on the file system explorer. The menu contains these options:

- Download
- Preview
- Copy
- Rename
- Move
- Delete
- Details

Use case diagram



Use case diagram with all the possible actions that the users should be able to do after our thoughtful analysis on the project. These are:

- Directory navigation
- Upload files
- Create folders
- Search folder or document
- Interaction with files:
    - Rename
    - Preview
    - Delete
    - Copy
    - See details or properties
    - Download files
    - Move

Observed incidents

Windows modification date:

On the Windows OS it's only possible to obtain the creation date and not the modification date through the function **filectime()**.

```
echo "<p class='text-break'><strong>Name: </strong>";
echo basename($currentElement);
echo "</p>
    <p><strong>Creation date: </strong>";
echo date("d F Y - H:i", filemtime($currentElement));
echo "</p>
    <p><strong>Size: </strong>";
echo human_filesize(filesize($currentElement));
echo "</p>";
```

Relative paths:

Paths for all functions and features must be always relative in relation to the *index.php* element, which is the origin of the project.

Because of this, we've learned the specific use of "../", "./" and other options.

```
$currentPath = "../" . substr($completePath, strpos($completePath, "root"));

include_once 'fillContent.php';
```

Bootstrap modals not working:

We've used plenty of modals on this project, so first we thought that it would be a good idea to use automatically generated bootstrap modals. But they didn't work on our project!

We tried some solutions, but they didn't work. So unwilling to lose more time, we made the modals and their functionalities manually.

```
document.getElementById("folderBtn").addEventListener("click", function () {
  modal.classList.toggle("hidden");
  shadow.classList.toggle("hidden");
  modalInput.select();
});      You, a week ago • feat: function created with all validation

closing.forEach((element) => {
  element.addEventListener("click", function (e) {
    e.preventDefault();
    modal.classList.toggle("hidden");
    shadow.classList.toggle("hidden");
    modalInput.value = "Unnamed folder";
    modalInput.select();
  });
});
```

Lessons learned

Server petitions and PHP interactions:

We've learned three main methods to send information through HTTP petitions:

- **POST**: best way to send information when having a form.

```php
if (isset($_POST['renameInput'])) {
    $newFileName = $_POST['renameInput'];
    $oldFileName = $_POST['oldNameInput'];
    $directory = scandir("../" . $currentPath);
    $file_parts = pathinfo($newFileName);
```

- **GET:** if you can modify the URL you can send information using GET.

```php
<?php
function getFilter()
{
    return isset($_GET['search']) ? $_GET['search'] : '';
}
```

- **Ajax** or **xmlhttp:** best option when you don't want to redirect or refresh the actual page.

```javascript
function ajaxPetition(responseContainer, endpoint, e) {
  var xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById(responseContainer).innerHTML = this.responseText;
    }
  };
  xmlhttp.open(
    "GET",
    "src/" + endpoint + "?n=" + e.target.getAttribute("data-title"),
    true
  );
  xmlhttp.send();
}
```

Array walk:

We've learned a new way to iterate through an array and modify it with a custom function: the **array_walk()** function.

More information: https://www.php.net/manual/en/function.array-walk.php

```php
$files = getFiles($_SESSION['currentPath']);
if ($files) {
  array_walk($files, function ($filePath) {
    $fileIcon = getFileIcon($filePath);
    $fileName = basename($filePath);
```

PHP callbacks:

We've learned how to pass callbacks to PHP internal functions with the **use()** syntax.

```php
array_walk($pathArr, function($folderName) use($currentPath) {
    $originalPath = substr($currentPath, 0, strpos($currentPath, $folderName));
    $folderPath = $originalPath . $folderName;
```

Require and include:

In our effort to modularize the project we've learned how to include different PHP into index and into other files with the **require()**, **require_once()**, **include()**, and **include_once()**.

```php
require_once('./src/folderNavigation.php');
include_once './src/fillContent.php';
```

Plenty of PHP internal functions:

Of course using PHP we've found and learned about plenty of PHP internal functions that gave us a lot of options to programm our features and functions. Some of these are:

- mkdir
- copy
- rmdir
- unlink
- basename
- is_dir
- glob
- scandir
- str_contains
- count
- substr
- strpos
- in_array
- fopen
- fgetcsv

...