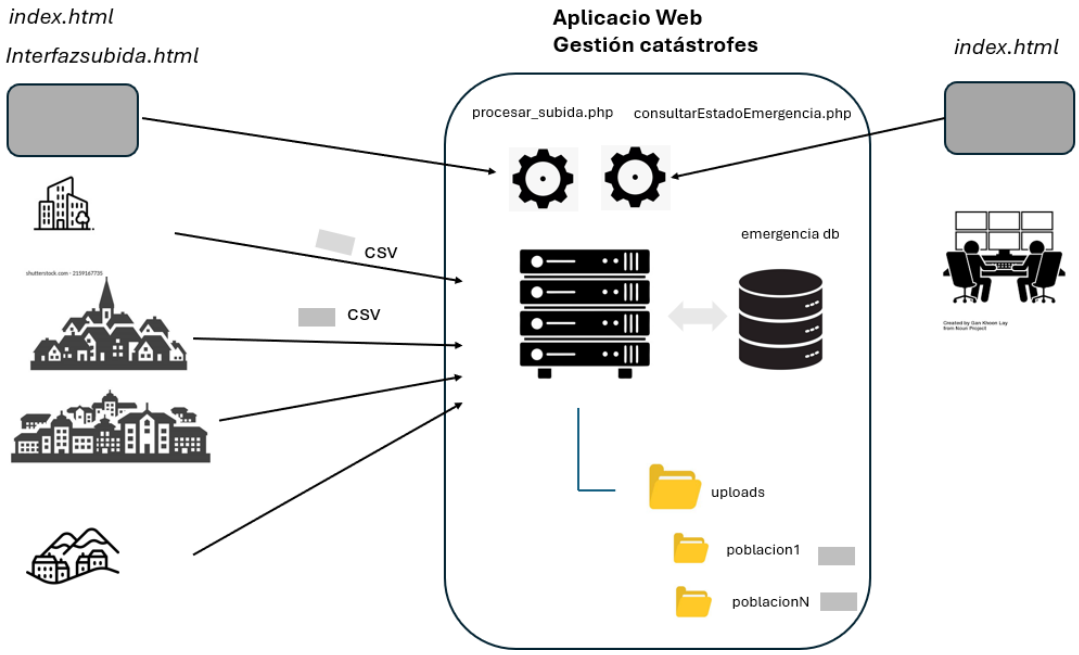
Actividad de desarrollo en Servidor.
Aplicación de soporte a la gestión de catástrofes naturales

Índice

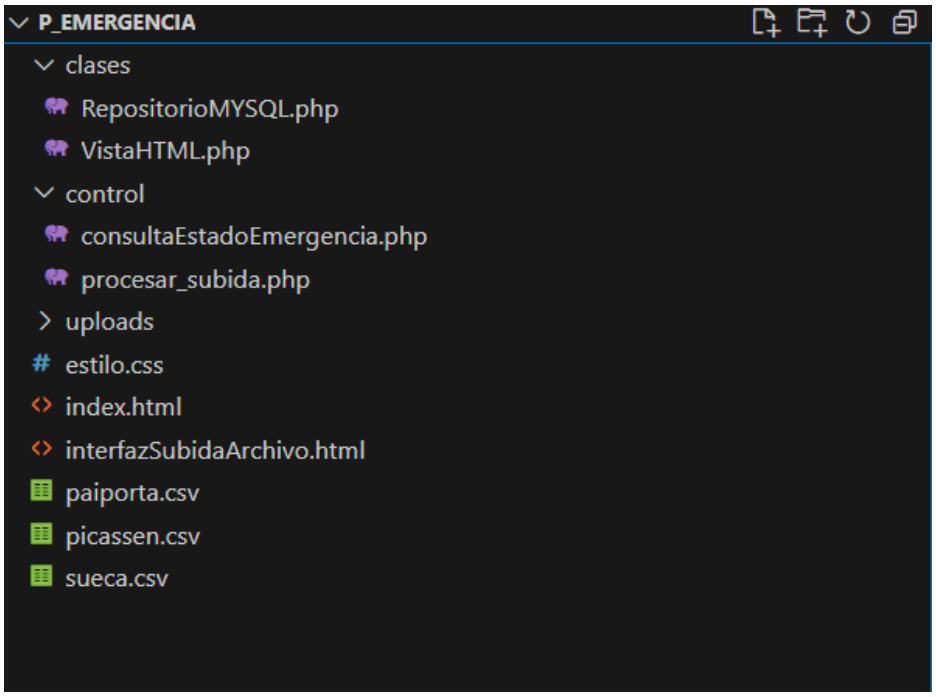
Descripción de la actividad

Vamos a **desarrollar una aplicación web que de soporte a la actualización y comunicación de datos en catástrofe natural.**

La estructura general es la siguiente:



La estructura de componentes que tendrá el proyecto será la siguiente:



IMPORTANTE :Como ves las clases se organizan en le directorio clases y los módulos de control en el directorio control

Interfaz de zona afectada (index.php)

Esta interfaz permite que cada municipio afectado actualice la información del estado de sus necesidades, para ello debe utilizar la opción de la izquierda


```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Consulta de Municipios Afectados</title>
    <link rel="stylesheet" href="estilo.css">
</head>
<body>
    <header>
        <h1>Comunicación de Municipios Afectados (castastrofe
natural)</h1>
    </header>

    <nav>
        <a href="interfazSubidaArchivo.html"> Actualización de
información de zona mediante archivo cvs</a>
        <a href="control/consultaEstadoEmergencia.php">Ver Información
de Municipios Afectados</a>
    </nav>
</body>
</html>
```

El formato del archivo csv que se debe subir es fijo y tiene la siguiente estructura.

```
municipo, personasAfectadas, comunicacionesCortadas, necesidades
```

Un ejemplo de csv válido sería el siguiente:

```
"picassen",2500,SI, "agua, luz, víveres de primera necesidad"
```

Donde se indica que el municipio es Picassen, hay 2500 personas afectadas, no hay comunicaciones y se necesita agua, luz y víveres de primera necesidad

IMPORTANTE: Fíjate en el último campo porque es un campo de texto que recoge las necesidades separadas por.

Interfaz gráfica interfazSubidaArchivo.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Subir Datos de Catástrofe</title>
    <link rel="stylesheet" href="estilo.css">
</head>
<body>
    <header>
        <h1>Subida de Archivo CSV - Datos de Inundación</h1>
    </header>
    <main>
        <form action="control/procesar_subida.php" method="POST"
enctype="multipart/form-data" class="form-subida">
            <input type="file" name="archivo_csv" accept=".csv" required>
            <button type="submit">Subir Archivo</button>
        </form>
    </main>
</body>
</html>
```

Lógica de control de la aplicación

La aplicación tienes dos módulos de control principales

Módulo procesar_archivo.php

Este módulo realiza lo siguiente:

1. Procesa el archivo de subida

2. Si es la primera vez que el municipio afectado sube el archivo , la aplicación crea un subdirectorio en **"uploads/nombre_municipio/ultimo_reporte.csv,** guarda el archivo csv , lo procesa **y guarda los datos de necesidades actualizados en la base de datos emergencia** en la **tabla estado_emergencia**

3. Si el archivo ya se ha subido anteriormente, se hará lo mismo **pero actualizando la base de datos, con la nueva información**

Lo anterior está reflejado en el siguiente esquema de código y acciones que deben hacerse

Modulo procesar_subida.php

```
//1.- Abrir el archivo CSV y leer el nombre del municipio (primer campo)

//2.- Crear carpeta del municipio si no existe

$folderPath = "uploads/" . $municipio; // El nombre del municipio se poner

//2.1 Si no existe creamos la carpeta

        if (!file_exists($folderPath)) {

            mkdir($folderPath, 0777, true);

        }

// Ruta final del archivo en la carpeta del municipio

        $fileDestination = $folderPath . "/ultimo_reporte.csv";

//3. Mover el archivo a la carpeta del municipio, sobrescribiendo el archivo anterior

/4.- Verificar la última fecha de actualización en la base de datos

 $lastUpdate = $repositorio->obtenerUltimaFechaReporte($municipio);

 $newUpdate = filemtime($fileDestination); // Fecha de subida del archivo

// 4.- Buscar palabras clave en el campo necesidades de ayuda con la función strpos NOTA: ESTO SE DEBE HACER POR CADA NECESIDAD ESTABLECIDA
```

```
                if (stripos($necesidades_ayuda, 'agua') !== false) {
                    $agua = 1;
                }
 // Asignar a $otros si no se encuentra ninguna palabra clave
                if ($agua === 0 && $productos_limpieza === 0 &&
$viveres === 0 && $medicinas === 0) {
                    $otros = $necesidades_ayuda;
                }
//5.- Insertar o actualizar los datos en la base de datos
                $repositorio->guardarEstadoMunicipio($municipio,
$personas_afectadas, $comunicaciones_cortadas, $agua,
$productos_limpieza, $viveres, $medicinas, $otros);
//6.- Cerrar el archivo
                fclose($file);
```

Módulo consultarEstadoEmergencia.php

Este módulo accede a la base de datos central (usando la clase RepositoryMSQL) y obtiene la información de todos los municipios , sumando el total de personas afectadas y mostrando las distintas necesidades que se han clasificado para cada una.

La siguiente tabla muestra la información que obtiene este módulo

Necesidades de Ayuda - Municipios Afectados

Nombre Población	Personas Afectadas	Comunicaciones Cortadas	personas_afectadas	comunicaciones_cortadas	agua	productos_limpieza	viveres	medicinas	otros	Fecha de Reporte
picassen	2500	0	2500	0	1	0	1	0		2024-11-05 10:42:42
paiporta	1500	0	1500	0	0	0	0	0	luz	2024-11-05 10:11:00
sueca	600	0	600	0	0	0	0	0	internet ,wifi	2024-11-05 11:08:08

Total afectados 4600

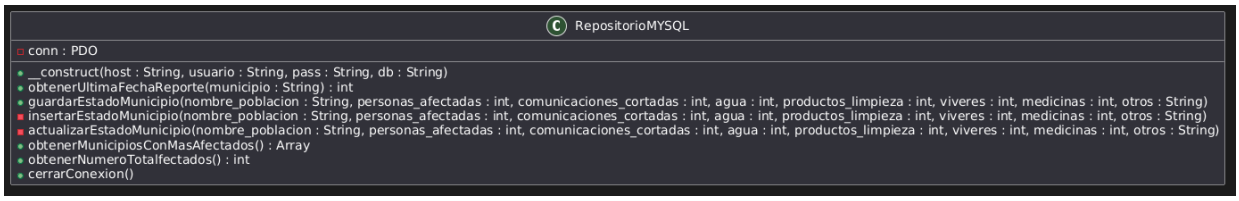
Módulo **consultarEstadoEmergencia.php**

```
<?php
//COMPLETAR ESTE MODULO
//1.-importar clases de soporte necesarias
//2.- Obtener los municipios Afectados
//3.- Obtener el número total de afectados
//4.-Mostrar el resultado con la vista
```

Clases de Soporte

RepositoryMSQL.php

Esta clase **encapsula toda la lógica de acceso a la base de datos** emergencia



Resumen de métodos de la clase

```
+ __construct(host : String, usuario : String, pass : String, db : String) +
obtenerUltimaFechaReporte(municipio : String) : int

+ guardarEstadoMunicipio(nombre_poblacion : String, personas_afectadas : int,
comunicaciones_cortadas : int, agua : int, productos_limpieza : int, viveres : int,
medicinas : int, otros : String)

- insertarEstadoMunicipio(nombre_poblacion : String, personas_afectadas : int,
comunicaciones_cortadas : int, agua : int, productos_limpieza : int, viveres : int,
medicinas : int, otros : String)

- actualizarEstadoMunicipio(nombre_poblacion : String, personas_afectadas : int,
comunicaciones_cortadas : int, agua : int, productos_limpieza : int, viveres : int,
medicinas : int, otros : String)

+ obtenerMunicipiosConMasAfectados() : Array

+ obtenerNumeroTotalfectados() : int

 + cerrarConexion()
```

NOTA: Fíjate en el método `guardarEstadoMunicipio`

```
class RepositorioMYSQL

{

    private $conn;

    public function __construct($host, $usuario, $pass, $db)

     {

         //COMPLETAR

     }

    public function obtenerUltimaFechaReporte($municipio)

    {
```

```
        // Este select solo puede dar un registro
        $query = "SELECT fecha_reporte FROM estado_municipios
WHERE nombre_poblacion = '$municipio' ORDER BY fecha_reporte
DESC LIMIT 1";
        $result = $this->conn->query($query);


        if ($result && $row = $result->fetch()) {
            return strtotime($row['fecha_reporte']);
        } else
            return null;
    }



    // Método para guardar el estado del municipio
    public function guardarEstadoMunicipio($nombre_poblacion,
$personas_afectadas, $comunicaciones_cortadas, $agua,
$productos_limpieza, $viveres, $medicinas, $otros)
    {
        // Primero, comprueba si el municipio ya existe
        $queryCheck = "SELECT COUNT(*) AS total FROM
estado_municipios WHERE nombre_poblacion = :nombre_poblacion";
        …
        if ($result['total'] > 0) {
            // Si el municipio existe, actualiza el registro
            $this->actualizarEstadoMunicipio($nombre_poblacion,
$personas_afectadas, $comunicaciones_cortadas, $agua,
$productos_limpieza, $viveres, $medicinas, $otros);
        } else {
            // Si el municipio no existe, inserta un nuevo
registro
            $this->insertarEstadoMunicipio($nombre_poblacion,
$personas_afectadas, $comunicaciones_cortadas, $agua,
$productos_limpieza, $viveres, $medicinas, $otros);
        }
```

```
    }


    // Método para insertar un nuevo estado del municipio
    private function insertarEstadoMunicipio($nombre_poblacion, $personas_afectadas, $comunicaciones_cortadas, $agua, $productos_limpieza, $viveres, $medicinas, $otros)
    {
        $queryInsert = "INSERT INTO estado_municipios (nombre_poblacion, personas_afectadas, comunicaciones_cortadas, agua, productos_limpieza, viveres, medicinas, otros)
                        VALUES (:nombre_poblacion, :personas_afectadas, :comunicaciones_cortadas, :agua, :productos_limpieza, :viveres, :medicinas, :otros)";


        $stmtInsert = $this->conn->prepare($queryInsert);


        // Vincular parámetros
        $stmtInsert->bindParam(':nombre_poblacion', $nombre_poblacion);
        $stmtInsert->bindParam(':personas_afectadas', $personas_afectadas);
        $stmtInsert->bindParam(':comunicaciones_cortadas', $comunicaciones_cortadas);
        $stmtInsert->bindParam(':agua', $agua);
        $stmtInsert->bindParam(':productos_limpieza', $productos_limpieza);
        $stmtInsert->bindParam(':viveres', $viveres);
        $stmtInsert->bindParam(':medicinas', $medicinas);
        $stmtInsert->bindParam(':otros', $otros);


        // Ejecutar la consulta de inserción
        $stmtInsert->execute();
    }
```

```
    // Método para actualizar el estado del municipio
    private function
actualizarEstadoMunicipio($nombre_poblacion,
$personas_afectadas, $comunicaciones_cortadas, $agua,
$productos_limpieza, $viveres, $medicinas, $otros)
    {
        $queryUpdate = "UPDATE estado_municipios SET
                            personas_afectadas =
:personas_afectadas,
                            comunicaciones_cortadas =
:comunicaciones_cortadas,
                            agua = :agua,
                            productos_limpieza =
:productos_limpieza,
                            viveres = :viveres,
                            medicinas = :medicinas,
                            otros = :otros
                        WHERE nombre_poblacion =
:nombre_poblacion";


        $stmtUpdate = $this->conn->prepare($queryUpdate);


        // Vincular parámetros
        $stmtUpdate->bindParam(':nombre_poblacion',
$nombre_poblacion);
        $stmtUpdate->bindParam(':personas_afectadas',
$personas_afectadas);
        $stmtUpdate->bindParam(':comunicaciones_cortadas',
$comunicaciones_cortadas);
        $stmtUpdate->bindParam(':agua', $agua);
        $stmtUpdate->bindParam(':productos_limpieza',
$productos_limpieza);
        $stmtUpdate->bindParam(':viveres', $viveres);
        $stmtUpdate->bindParam(':medicinas', $medicinas);
```

```
        $stmtUpdate->bindParam(':otros', $otros);


        // Ejecutar la consulta de actualización
        $stmtUpdate->execute();
    }







    public function obtenerMunicipiosConMasAfectados()
    //COMPLETAR
        // Devuelve un array asociativo de información de municipios
        return $municipios;
    }


    public function obtenerNumeroTotalfectados(): int
    {
        $query = "SELECT SUM(personas_afectadas) as suma FROM estado_municipios  ";
        $result = $this->conn->query($query);


        // Verificar si la consulta se ejecutó correctamente
        if ($result !== false) {
            $row = $result->fetch(PDO::FETCH_ASSOC);
            return (int)$row['suma']; // Retorna la suma como un entero
        }


        // En caso de que la consulta falle, devolver 0 o lanzar una excepción
        return 0;
```

```
    }
    public function cerrarConexion()
    {
        $conn = null;
    }
}
```

Clase VistaHTML.php

```
<?php
class VistaHTML
{
    public function __construct() {}

    public function mostrarEstadoMunicipios($municipios, $total)
    {
        echo '
    <!DOCTYPE html>
    <html lang="es">
    <head>
        <meta charset="UTF-8">
        <title>Consulta de Ayuda - Protección Civil</title>
    </head>
    <body>
        <h2>Necesidades de Ayuda - Municipios Afectados</h2>
        <table border="1">
            <thead>
                <tr>
                    <th>Nombre Población</th>
                    <th>Personas Afectadas</th>
```

```
                    <th>Comunicaciones Cortadas</th>
                    <th>personas_afectadas </th>
                    <th>comunicaciones_cortadas</th>
                    <th>agua</th>
                    <th>productos_limpieza</th>
                    <th>viveres</th>
                    <th>medicinas</th>
                   <th>otros</th>

                    <th>Fecha de Reporte</th>
                </tr>
            </thead>
            <tbody>';


        foreach ($municipios as $registro) {
            echo "<tr>
                                <td>{$registro['nombre_poblacion']}<
/td>
                                <td>{$registro['personas_afectadas']
}</td>
                                <td>{$registro['comunicaciones_corta
das']}</td>

                                <td>{$registro['personas_afectadas']
}</td>
                                <td>{$registro['comunicaciones_corta
das']}</td>
                                 <td>{$registro['agua']}</td>
                                <td>{$registro['productos_limpieza']
}</td>
                                <td>{$registro['viveres']}</td>
                                <td>{$registro['medicinas']}</td>
```

```
                            <td>{$registro['otros']}</td>
                           <td>{$registro['fecha_reporte']}</td
>
                        </tr>";
        }
        echo '
            </tbody>
        </table>
    </body>';
        echo '<h1> Total afectados ' . $total . '</h1>';
        echo '</html>';
    }
}
```

Cuestiones

1. Desarrolla la aplicación siguiendo los requisitos planteados y completando las partes que faltan

2. Prueba la aplicación con los siguientes archivos y después prueba a cambiarlos y ver si se actualiza la información en el servidor y la base de datos . Por ejemplo cambia la necesidades de Paiporta a “luz,agua y víveres”

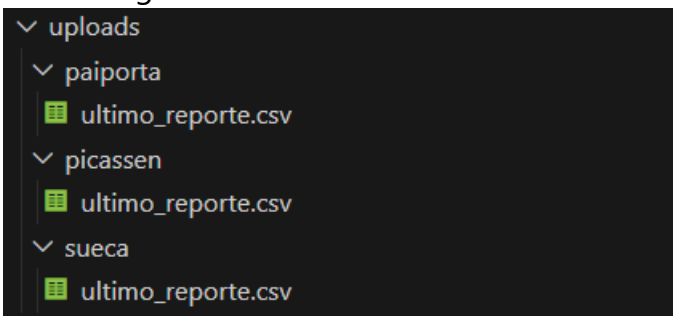
```
paiporta.csv
municipo,personasAfectadas,comunicacionesCortadas,
necesidades
"paiporta",1500,NO, "luz"
```

```
picassen.csv
municipo,personasAfectadas,comunicacionesCortadas,
necesidades
"picassen",2500,SI, "agua, luz, viveres de primera
necesidad"
```

sueca.csv

```
municipo,personasAfectadas,comunicacionesCortadas,
necesidades
"sueca",600,SI, "internet ,wifi"
```

Si tú código funciona deberas obtener una estructura como esta:



3. Modifica el diseño de la aplicación para que autentique a los usuarios que acceden a la parte de consulta. Para ello, diseña una nueva tabla dentro de la base de datos denominada usuarios que guarde id y contraseña

Anexo 1.

Script de base de datos

```
CREATE DATABASE IF NOT EXISTS emergencia;

-- Usar la base de datos

USE emergencia;


-- Crear la tabla estado_municipios
CREATE TABLE IF NOT EXISTS estado_municipios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_poblacion VARCHAR(100) NOT NULL,
    personas_afectadas INT NOT NULL,
    comunicaciones_cortadas INT NOT NULL,
    agua TINYINT(1) NOT NULL DEFAULT 0,
    productos_limpieza TINYINT(1) NOT NULL DEFAULT 0,
    viveres TINYINT(1) NOT NULL DEFAULT 0,
    medicinas TINYINT(1) NOT NULL DEFAULT 0,
```

```
    otros VARCHAR(255),
    fecha_reporte DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    UNIQUE KEY (nombre_poblacion)
);
```

Estilo.css

/* Estilos generales */

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    align-items: center;
}
/* Estilos para el encabezado */
header {
    width: 100%;
    background-color: #4CAF50;
    padding: 10px 0;
    text-align: center;
}

header h1 {
    color: white;
    margin: 0;
}

/* Estilos de la barra de navegación */
nav {
```

```
    margin-top: 20px;
    display: flex;
    gap: 20px;
}

nav a {
    text-decoration: none;
    color: white;
    padding: 10px 20px;
    background-color: #4CAF50;
    border-radius: 5px;
    transition: background-color 0.3s ease;
}

nav a:hover {
    background-color: #45a049;
}

/* Estilos para el contenido principal */
main {
    margin-top: 30px;
    width: 80%;
}

/* Estilos para la tabla */
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}
```

```
th, td {
    padding: 12px;
    border: 1px solid #ddd;
    text-align: center;
}

th {
    background-color: #f4f4f4;
    font-weight: bold;
}

/* Estilos generales */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    flex-direction: column;
    align-items: center;
}

/* Estilos para el encabezado */
header {
    width: 100%;
    background-color: #4CAF50;
    padding: 10px 0;
    text-align: center;
}
```

```
header h1 {
    color: white;
    margin: 0;
}

/* Estilos para el contenido principal */
main {
    margin-top: 30px;
    width: 80%;
    display: flex;
    justify-content: center;
}

/* Estilos del formulario de subida */
.form-subida {
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 5px;
    background-color: #f9f9f9;
    width: 100%;
    max-width: 400px;
}

/* Estilos para los elementos del formulario */
.form-subida input[type="file"] {
    margin: 15px 0;
    padding: 8px;
```

```
    width: 100%;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 16px;
}

.form-subida button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.form-subida button:hover {
    background-color: #45a049;
}
/* Estilos del formulario de subida */
.form-subida {
    display: flex;
    flex-direction: column;
    align-items: center;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 5px;
    background-color: #f9f9f9;
    width: 100%;
```

```
    max-width: 400px;
}

/* Estilos para los elementos del formulario */
.form-subida input[type="file"] {
    margin: 15px 0;
    padding: 8px;
    width: 100%;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 16px;
}

.form-subida button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.form-subida button:hover {
    background-color: #45a049;
}
```